INTEGRASI ANALISIS DATA ULASAN *ONLINE* BERBASIS *MACHINE LEARNING* DENGAN *QUALITY FUNCTION DEPLOYMENT* GUNA MENDUKUNG PROSES PENGEMBANGAN PRODUK (STUDI KASUS: VENTELA PUBLIC LOW)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Strata-1 Pada Jurusan Teknik Industri Fakultas Teknologi Industri

Nama : Syafira Mahardiningtyas
No. Mahasiswa : 16 522 242

PROGRAM STUDI TEKNIK INDUSTRI
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2020

PERNYATAAN KEASLIAN

Demi Allah, saya akui karya ini adalah karya hasil kerja saya sendiri kecuali nukilan dan ringkasan yang setiap satunya telah saya jelaskan sumbernya. Jika dikemudian hari ternyata terbukti pengakuan saya ini tidak benar dan melanggar peraturan yng sah dalam karya tulis dan hak kekayaan intelektual maka saya bersedia ijazah yang telah saya terima untuk ditarik kembali oleh Universitas Islam Indonesia.

Yogyakarta, 18 September 2020



Syafira Mahardiningtyas

165242242

LEMBAR PENGESAHAN DOSEN PEMBIMBING

INTEGRASI ANALISIS DATA ULASAN *ONLINE* BERBASIS *MACHINE LEARNING* DENGAN *QUALITY FUNCTION DEPLOYMENT* GUNA MENDUKUNG PROSES PENGEMBANGAN PRODUK (STUDI KASUS: VENTELA PUBLIC LOW)

TUGAS AKHIR

Oleh

Nama : Syafira Mahardiningtyas
No. Mahasiswa : 16 522 242

Yogyakarta, 18 September 2020

Menyetujui,
Pembimbing

(Abdullah 'Azzam, S.T., M.T)

LEMBAR PENGESAHAN DOSEN PENGUJI

INTEGRASI ANALISIS DATA ULASAN *ONLINE* BERBASIS *MACHINE LEARNING* DENGAN *QUALITY FUNCTION DEPLOYMENT* GUNA MENDUKUNG PROSES PENGEMBANGAN PRODUK (STUDI KASUS: VENTELA PUBLIC LOW)

TUGAS AKHIR

Oleh

Nama : Syafira Mahardiningtyas
No. Mahasiswa : 16 522 242

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 Teknik Industri

Yogyakarta, 07 September 2020

Tim Penguji

Abdullah 'Azzam, S.T., M.T.
Ketua Tim Penguji

Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D
Anggota I

Atyanti Dyah Prabaswari, S.T., M.Sc
Anggota II

Mengetahui
Ka. Prodi Teknik Industri
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Taufiq Immawan, ST., MM.)

HALAMAN PERSEMBAHAN

Alhamdulillahirabbil' alamin

Kepada yang saya cintai kedua orang tua saya
(Edi Sutopo & Ngadini)
Terimakasih atas semua dukungan dan doa

Kepada yang saya hormati Dosen Pembimbing saya, Abdullah 'Azzam, S.T., M.T
Terimaksih atau pengetahuan dan bimbingan yang telah diberikan

Dan kepada sahabat saya dan Asisten Laboratorium SIMAN

Dan kepada Jurusan Teknik Industri Universitas Islam Indonesia

Jazakumullah Khairan Katsira

HALAMAN MOTTO

“ Apakah sama antara orang-orang yang mengetahui dengan orang yang tidak mengetahui? ....”

(Az-Zumar:9)

“Sebaik–baik manusia diantaramu adalah yang paling banyak memberi manfaat bagi orang lain.”

(HR. Bukhari Muslim)

HALAMAN PENGANTAR

Asalamualaikum Warahmatullahi Wabarakatuh

Syukur kehadiran Allah SWT dengan berbagai limpahan karunia yang kita terima. Shalawat dan salam tidak lupa kita sampaikan kepada junjungan kita Nabi Muhammad SAW, kepada keluarga, dan para sahabat beliau. Semoga kita termasuk umat beliau yang giat dalam mengamalkan ajaran beliau. Amin.

Alhamdulillah atas karunia-Nya dan doa semangat orang terdekat. Laporan Tugas Akhir ini diketahui wujudnya. Tebal tidak menjadi ukuran, namun manfaat bagi penulis dan lingkunganlah yang menjadi ukuran pentingnya. Dalam mengolah dan menganalisis ilmu di dalam buku ini tidak lepas dari bantuan beberapa cendekiawan dan dermawan. Diantaranya penulis berterimakasih kepada:

1. Hari Purnomo, Prof., Dr., Ir., M.T. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. M. Ridwan Andi Purnomo, S.T., M.Sc., Ph.D.. selaku Kepala Jurusan Teknik Industri Universitas Islam Indonesia.
3. Dr. Taufiq Immawan, S.T., M.M. selaku Kepala Program Studi S1 Teknik Industri Universitas Islam Indonesia.
4. Abdullah ‘Azzam, S.T., M.T. selaku dosen pembimbing yang selalu membimbing, memberikan ilmu, dan menyediakan waktu untuk penulis sampai akhirnya dapat menyelesaikan Laporan Tugas Akhir.
5. Ayah, Ibu, dan Alfian selaku keluarga atas doa, semangat, dan dukungan yang senantiasa mengiringi.
6. Kepala Laboratorium, Laboran dan seluruh Asisten Laboratorium Sistem Manufaktur Terintegrasi Universitas Islam Indonesia yang telah memberikan kesempatan bagi penulis untuk mengembangkan ilmu dan memperoleh pengalaman yang berharga.
7. Teman dekat penulis, atas dukungan, doa dan keikhlasan bantuan yang diberikan.

8. Semua yang tidak dapat penulis sebutkan satu per satu yang membantu dalam penyelesaian Laporan Tugas Akhir ini.

Meskipun banyak kekurangan dalam penulisan, terdapat harapan semoga laporan ini dapat bermanfaat dan mendatangkan berkah ilmu bagi pembaca pada umumnya dan Teknik Industri Universitas Islam Indonesia pada khususnya. Sebagai refleksi, ilmu tanpa disebarkan dan dikembangkan hanya akan menuju sia-sia. Mulai saat yang tepat, saya selaku penulis berkeinginan lebih keras berbagi dan belajar, agar nilai luhur Ulil Albab dapat tertanam dalam diri saya, dan kemudian dapat memberikan kontribusi kepada lingkungan

Wasalamualaikum Warahmatullahi Wabarakatuh

Yogyakarta, 06 Agustus 2020

(Syafira Mahardiningtyas)
16522242

ABSTRAK

Proses pengembangan produk dilakukan dengan langkah awal yaitu perencanaan desain baik dari segi fisik maupun kualitas. Kebutuhan konsumen atau *Voice of Customer* menjadi *input* penting bagi proses pengembangan desain. Perkembangan teknologi dan media saat ini menjadikan data *review online* sebagai sumber data yang menjanjikan. Produk Ventela Public Low merupakan salah satu produk yang terkenal karena banyaknya pendapat/*review* tentang produk yang disampaikan melalui media sosial. Melalui *machine learning* dengan tahap *text processing* dilakukan analisa terhadap data *review* produk Ventela Public Low. Analisis sentimen akan digunakan, untuk menemukan kelompok kelas dari masing-masing data. Algoritma klasifikasi yang digunakan adalah *Naïve bayes* dan *SVM*. Dari serangkaian percobaan, didapatkan model dengan nilai performansi tertinggi yaitu sebesar 71. Hasil validasi model tersebut dengan *10-cross fold validation* diperoleh nilai rata-rata sebesar 0,850463 atau dalam persen 85,0463%. Selanjutnya mengukur keakuratan model dengan menghitung *ROC AUC*, dengan nilai 0,858485 dan 0,866732 yang berarti model termasuk kedalam *good classification*. Model ini akan digunakan pada tahap selanjutnya untuk memproses data *input QFD*. Dalam beberapa tahap *QFD* diperlukan peran dari *stakeholder* seperti tim desain, tim pemasaran, dan bagian produksi, maka pada beberapa tahap *QFD* akan digunakan data *dummy* (subjektif peneliti). Dari hasil simulasi *QFD* yang dilakukan peneliti, menunjukkan bahwa integrasi antara analisis data *review* menggunakan *machine learning* dangan *QFD* menghasilkan informasi yang layak sesuai dengan ketentuan metode *QFD* dan dapat mendukung proses pengembangan produk dari segi jumlah data, topik data yang beragam, dan mengurangi subjektivitas konsumen yaitu pada tahap penentuan *VoC* dan nilai kinerja produk dan pesainng.

Keyword: Machine Learning, Sentimen, *QFD, HoQ*

DAFTAR ISI

HALAMAN JUDUL i
PERNYATAAN KEASLIAN ii
LEMBAR PENGESAHAN DOSEN PEMBIMBING iii
LEMBAR PENGESAHAN PENGUJI iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTTO vi
HALAMAN PENGANTAR vii
ABSTRAK ix
DAFTAR ISI x
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 6
1.3 Tujuan Penelitian 6
1.4 Batasan Penelitian 6
1.5 Manfaat Penelitian 7
1.6 Sistematika Penulisan 7
BAB II KAJIAN PUSTAKA 9
2.1 Kajian Deduktif 9
2.1.1 Produk 9
2.1.2 Pengembangan Produk 10
2.1.3 *Big Data* 10
2.1.4 *Review*/Ulasan *Online* 11
2.1.5 *Machine Learning (ML)* 11
2.1.6 *Natural Language Processing (NLP)* 12
2.1.7 Analisis Sentimen 13
A. *Term Frequency Inverse Document Frequency (TFIDF)* 14
B. *Bag of Words (BoW)* 15
C. *Mutual Information (MI)* 16
D. *Naïve Bayes (NB)* 17
E. *Support Vector Machine (SVM)* 18
2.1.8 Asosiasi Kata 19
2.1.9 *Quality Function Deployment (QFD)* 21
2.2 Kajian Induktif 21
BAB III METODE PENELITIAN 30
3.1 Objek Penelitian 30
3.2 Identifikasi Masalah 30
3.3 Kajian Literatur 31
3.4 Metode Pengumpulan Data 31
3.4.1 Data Primer 31
3.4.2 Data Sekunder 31

3.4.3 Teknik Pengumpulan Data 32
3.5 Pengolahan Data 32
3.5.1 Pelabelan Data 32
3.5.2 Data *Pre-processing* 33
3.5.3 Pembagian Data 34
3.6 Analisis dan Pembahasan 42
3.7 Rekomendasi 42
3.8 Kesimpulan dan Saran 42
3.9 *Flow Chart* Kerangka Penelitian 43
BAB IV PENGUMPULAN DAN PENGOLAHAN DATA 44
4.1 Pengumpulan Data 44
4.1.1 Twitter *Scraping* 45
4.1.2 Youtube *Scraping* 45
4.2 Pelabelan Data 46
4.3 Proses Pengolahan *Dataset* 48
4.3.1 *Preprocessing* 48
4.3.2 Pembobotan Kata 53
4.3.3 Seleksi fitur 57
4.4 Pembentukan Data *Training* dan *Testing* 60
4.5 Proses Pengolahan Data *Training* 61
4.5.1 Proses Klasifikasi 61
4.5.2 Analisa dan Evaluasi Algoritma 71
4.6 Prediksi dan Asosiasi Kata 73
4.7 Tahap *QFD* 77
4.7.1 Identifikasi *Voice of Customer* (Kebutuhan Konsumen) 77
4.7.2 Analisa Tingkat Kinerja Produk dan Pelayanan 79
4.7.3 Penentuan Tingkat Kepentingan *(Important Ratings)* 81
4.7.4 Menghitung Rasio Perbaikan *(Improvement Ratio)* 84
4.7.5 Menentukan Titik Penjualan *(Sales Point)* 85
4.7.6 Menentukan Skala Kepentingan Konsumen 86
4.7.7 Menentukan Daftar Kebutuhan Teknis 87
4.7.8 Analisa Hubungan Antara Kebutuhan Konsumen dengan Kebutuhan Teknis dan Kesulitan Teknis 88
4.7.9 Menghitung Skor Kebutuhan Teknis 88
4.7.10 Menentukan Korelasi Antar Kebutuhan Teknis 89
4.7.11 Menyusun Semua Data Perhitungan dan Analisa Kedalam Matriks *House of Quality* 89
BAB V HASIL DAN PEMBAHASAN 91
5.1 Pengumpulan Data 91
5.2 Analisa dan Evaluasi Eksperimen Model 94
5.3 Analisa Model Terpilih 98
5.4 Prediksi dan Asosiasi Kata 100
5.5 Analisa *QFD* 101
5.5.1 Identifikasi *Voice of Customer* (Kebutuhan Konsumen) 101
5.5.2 Analisa Tingkat Kinerja Produk dan Pelayanan 102
5.5.3 Penentuan Tingkat Kebutuhan dan Keinginan Relatif *(IR)* Sesuai Kebutuhan Konsumen 103
5.5.4 Menentukan Rasio Perbaikan *(Improvement Ratio)* 104
5.5.5 Menentukan *Sales Point* 105

5.5.6 Menentukan Skala Kepentingan Konsumen 105
5.5.7 Pembuatan Daftar Kebutuhan Teknis dan Analisa Hubungan Antara Kebutuhan Konsumen dengan Kebutuhan Teknis 105
5.5.8 Analisa Kesulitan Teknis 106
5.5.9 Hasil Simulasi *QFD* 106
BAB VI KESIMPULAN DAN SARAN 108
6.1 Kesimpulan 108
6.2 Saran 113
DAFTAR PUSTAKA 114
LAMPIRAN 119

DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu 26
Tabel 3. 1 *Confusion Matrix* 39
Tabel 4. 1 Deskripsi *User* 46
Tabel 4. 2 Contoh Pengambilan Keputusan Proses Pelabelan 47
Tabel 4. 3 Hasil *BoW* Dokumen B 55
Tabel 4. 4 Hasil *10-Cross Validation* 71
Tabel 4. 5 Hasil Asosiasi Data Prediksi *(Bigram)* 75
Tabel 4. 6 Hasil Asosiasi Data Prediksi *(Trigram)* 75
Tabel 4. 7 Tabel Evaluasi Metode Collocations 76
Tabel 4. 8 Data Review Sesuai Peringkat Nilai Asosiasi sebagai *Input QFD* 78
Tabel 4. 9 Data Review yang Telah Dicocokkan 78
Tabel 4. 10 Proses Pengelompokkan 79
Tabel 4. 11 Perhitungan Kinerja Produk 80
Tabel 4. 12 Nilai Kinerja Produk Masing-Masing Fitur 80
Tabel 4. 13 Tingkat Kinerja Produk 81
Tabel 4. 14 Pembentukan Fitur/Kriteria 82
Tabel 4. 15 Fitur yang Menjadi Kebutuhan Konsumen 82
Tabel 4. 16 Nilai *IR* Masing-Masing Fitur 83
Tabel 4. 17 Nilai Rasio Perbaikan 84
Tabel 4. 18 Nilai *Sales Point* 85
Tabel 4. 19 Hasil Skala Kepentingan Konsumen 86
Tabel 4. 20 Target Perbaikan Produk 87
Tabel 4. 21 Respon Teknis/Kebutuhan Teknis 87
Tabel 4. 22 Skor Kebutuhan Teknis 89
Tabel 5. 1 Klasifikasi Tanpa Normalisasi 94
Tabel 5. 2 Klasifikasi Dengan Penambahan Normalisasi 94
Tabel 5. 3 Klasifikasi Dengan Normalisasi dan *Hyperparameter Turning* 95
Tabel 5. 4 Hasil Percobaan Pertama dengan *Split* 70% dan 30% 96
Tabel 5. 5 Hasil Percobaan Kedua dengan *Split* 70% dan 30% 96
Tabel 5. 6 Hasil Percobaan Ketiga dan Keempat dengan *Split* 70% dan 30% 96
Tabel 5. 7 Hasil Percobaan Pertama dengan *Split Stratified* 96
Tabel 5. 8 Hasil Percobaan Kedua dengan *Split Stratified* 96
Tabel 5. 9 Hasil Percobaan Ketiga dan Keempat dengan *Split Stratified* 97
Tabel 5. 10 Hasil Rata-Rata Performansi Tiap Percobaan 99
Tabel 5. 11 *Top Words* Masing-Maing Kelas 100
Tabel 5. 12 Perbandingan Nilai *IR* 104
Tabel 6. 1 Perbandingan dengan *QFD* dan Penelitian Terdahulu 111

DAFTAR GAMBAR

Gambar 1.1 Grafik Pengguna Internet 2
Gambar 2. 1 Sistem *NLP* 13
Gambar 3. 1 Pembobotan *TFIDF* 36
Gambar 3. 2 Contoh Kata *TFIDF* 36
Gambar 3. 3 Matriks *HoQ* 40
Gambar 3. 4 *House of Quality* 41
Gambar 3. 5 Diagram Alur Penelitian 43
Gambar 4. 1 Hasil *Scraping* dari *Twitter* 45
Gambar 4. 2 Hasil *Scraping* dari *Youtube* 46
Gambar 4. 3 Data yang Telah Digabungkan 47
Gambar 4. 4 Hasil Proses *Pre-Processing* 49
Gambar 4. 5 Hasil *Case Folding* Dokumen B 49
Gambar 4. 6 Hasil Menghilangkan Tanda Baca Dokumen B 50
Gambar 4. 7 Hasil Menghilangkan *Punctuation* Dokumen B 50
Gambar 4. 8 Hasil *Tokenization* Dokumen B 51
Gambar 4. 9 Hasil Menghilangkan *Stopword* Dokumen B 52
Gambar 4. 10 Hasil Normalisasi Dokumen B 52
Gambar 4. 11 Hasil *Stemming* Dokumen B 53
Gambar 4. 12 Hasil *TFIDF* Dokumen B (Tanpa Normalisasi) 56
Gambar 4. 13 Hasil *TFIDF* Dokumen B (Dengan Normalisasi) 57
Gambar 4. 14 Nilai *TFIDF L1* Dokumen B 58
Gambar 4. 15 Nilai *MI* Dokumen B 59
Gambar 4. 16 Grafik Perbandingan Setelah Penyeleksian Fitur 60
Gambar 4. 17 *Naïve* Percobaan Pertama Pada Dokumen B dengan *BoW* 63
Gambar 4. 18 *SVM* Percobaan Pertama Pada Dokumen B dengan *BoW* 64
Gambar 4. 19 *Naïve* Percobaan Pertama Pada Dokumen B dengan *TFIDF* 64
Gambar 4. 20 *SVM* Percobaan Pertama Pada Dokumen B dengan *TFIDF* 64
Gambar 4. 21 *Naive* Percobaan Kedua Pada Dokumen B dengan *BoW* 65
Gambar 4. 22 *SVM* Percobaan Kedua Pada Dokumen B dengan *BoW* 65
Gambar 4. 23 *Naive* Percobaan Kedua Pada Dokumen B dengan *TFIDF* 65
Gambar 4. 24 *SVM* Percobaan Kedua Pada Dokumen B dengan *TFIDF* 65
Gambar 4. 25 *Naive* Percobaan Ketiga Pada Dokumen B dengan *TFIDF L1* 66
Gambar 4. 27 *Naive* Percobaan Ketiga Pada Dokumen B dengan *MI* 66
Gambar 4. 28 *SVM* Percobaan Ketiga Pada Dokumen B dengan *MI* 66
Gambar 4. 29 *Naive* Percobaan Keempat Pada Dokumen B 68
Gambar 4. 30 *SVM* Percobaan Keempat Pada Dokumen B (*TFIDF*) Dengan *Kernel* Linear 68
Gambar 4. 31 *SVM* Percobaan Keempat Pada Dokumen B (*BoW*) Dengan *Kernel* Linear 69
Gambar 4. 32 *SVM* Percobaan Keempat Pada Dokumen B (*TFIDF*) Dengan *Kernel RBF* 69

Gambar 4. 33 *SVM* Percobaan Keempat Pada Dokumen B (*BoW*) Dengan *Kernel RBF* 69
Gambar 4. 34 *SVM* Percobaan Keempat Pada Dokumen B *(TFIDF L1)* Dengan *Kernel* Linear 70
Gambar 4. 35 *SVM* Percobaan Keempat Pada Dokumen B *(TFIDF L1)* Dengan *Kernel RBF* 70
Gambar 4. 36 *SVM* Percobaan Keempat Pada Dokumen B *(MI)* Dengan *Kernel* Linear 70
Gambar 4. 37 *SVM* Percobaan Keempat Pada Dokumen B *(MI)* Dengan *Kernel RBF* .. 71
Gambar 4. 38 Hasil *ROC AUC* Model Terpilih 73
Gambar 4. 39 Hasil *Pre-processing* 74
Gambar 4. 40 Algoritma Prediksi 74
Gambar 4. 41 Grafik Proporsi Kelas Hasil dari Prediksi 75
Gambar 4. 42 Produk Ventela Low Public 77
Gambar 4. 43 Produk Converse Chuck Taylor 70s 78
Gambar 4. 44 *House of Quality* Produk Ventela Public Low 90
Gambar 5. 1 Jumlah Pengguna Media Sosial di Indonesia 92
Gambar 5. 2 Peringkat Pengguna *Marketplace* di Indonesia 93
Gambar 5. 3 *Confusion Matrix* Model Terpilih 97
Gambar 5. 4 Parameter *Hyperparameter* yang Menghasilkan Model Terpilih 98
Gambar 6. 1 Visualisasi Bagian Hasil Integrasi 112

BAB I

PENDAHULUAN

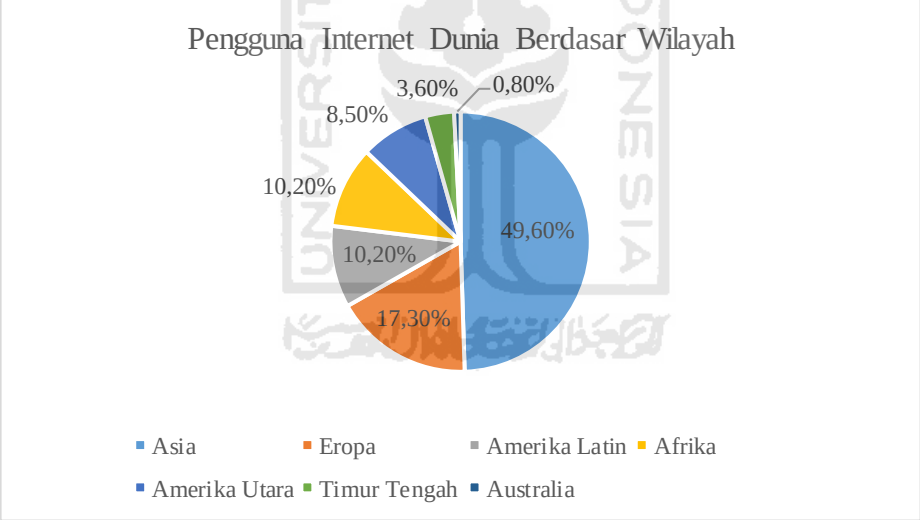
1.1 Latar Belakang

Banyak produk termasuk gagal di pasar salah satunya karena menggunakan asumsi saat pembuatannya dan kurangnya basis data saat merancangnya. Beberapa tahun lalu, *Microsoft* mencoba memperoleh kesempatan dengan merilis seri *windows phone*, yang diharapkan dapat memudahkan komunikasi dan dapat bersaing dengan *Android* dan *iOS*. Namun, pada kenyataannya *windows phone* tidak dapat mengalahkan dominasi kedua pasar tersebut. Pada bulan Juli tahun 2017 *Microsoft* menetapkan untuk mengakhiri produksi *windows Phone* karena tidak laku di pasaran (indoworx.com, 2018).

Pemberhentian usaha juga dialami oleh *platform Qlapa*, yang merupakan *platform daring* untuk menjual produk kerajinan tangan yang beroperasi pada tahun 2015. Tahun 2019 *Qlapa* memutuskan memberhentikan operasionalnya secara menyeluruh. Dalam keterangan pers pihak *Qlapa* mengatakan pemberhentian perusahaan dengan alasan bisnis yang dinilai tidak menguntungkan dan tidak berkelanjutan (iPrice Indonesia, 2019). Pada bidang transportasi, beberapa moda transportasi sudah tidak dapat dijumpai di akibatkan ekspetasi pengguna yang semakin tinggi, yaitu tentang keterjangkauan, cepat dan biaya yang murah.

Di masa sekarang berbicara dengan pengguna tentang apa yang mereka inginkan dan mereka butuhkan sangat mudah. Dengan gelombang konektivitas yang terbentuk, kehadiran dan semakin kuatnya mesin pencari seperti Google. Selain Google, pada masa ini juga ramai berkembang pesatnya media sosial atau jejaring sosial yang dapat memberikan efek mengorganisasi diri (Kasali, 2019).

Penggunaan internet telah menjadi bagian dari kegiatan dalam keseharian bagi mayoritas masyarakat dunia. *Internet World Stats* pada tahun 2017 mengungkapkan ada lebih dari 4 miliar pengguna internet di seluruh dunia dari berkisar total 7 miliar lebih populasi penduduk dunia. Dari 4 miliar pengguna internet tersebut, 3 miliar diantaranya merupakan pengguna aktif jejaring sosial. Asia merupakan daerah yang terdaftar memiliki jumlah pengguna internet terbesar yaitu mencapai 49,6% (IMARKETOLOGY, 2020).



(Sumber: (IMARKETOLOGY, 2020))

Gambar 1.1 Grafik Pengguna Internet

Dari penggunaan internet tersebut, situs Google menjadi situs yang paling banyak dikunjungi yaitu sebesar 16.38%, kemudian Facebook sebesar 5.89%, Youtube sebesar 2,94%, dan akumulatif situs lain sebesar 74.79%. Dari akumulatif situs yang ada, Amazon merupakan situs yang meraih posisi ke-5 yang paling banyak dikunjungi masyarakat. Hal ini dapat dikatakan bahwa banyak pengguna internet yang mempertimbangkan media

digital sebagai sarana untuk berbelanja produk atau jasa yang diinginkan dan dibutuhkan (IMARKETOLOGY, 2020).

Menurut data *vpnmentor*, jumlah transaksi penjualan *Business to Costumer* (*B2C) online* hampir di setiap wilayah dunia meningkat setiap tahunnya. Bahkan, total penjualan retail *e-commerce* di seluruh dunia bertambah setiap tahunnya. Diperkirakan akan dapat terus meningkat dan mencapai 4,479 triliun US Dollar pada tahun 2021. Hal tersebut juga terlihat di pasar Indonesia, data yang diungkapkan oleh KataData menunjukkan bahwa pada tahun 2016, jumlah konsumen online di Indonesia sudah mencapai sekitar 144 triliun Rupiah (IMARKETOLOGY, 2020). Persentase belanja *online* pengguna internet Indonesia adalah yang tertinggi dibanding negara di kawasan ASEAN. Di sisi lain, terdapat penelitian yang hasil dari Google Search pada tahun 2018 menobatkan Indonesia adalah pasar *e-commerce* terbesar dan paling cepat berkembang di Asia Tenggara, dengan rata-rata peningkatan majemuk belanja konsumen naik sebesar 49% pertahun (CNN Indonesia, 2019).

Dalam menentuan keputusan saat membeli suatu produk banyak dipengaruhi beberapa faktor. Pada penelitian (Nugraha, W, & Widayanto, 2017), menunjukkan bahwa secara simultan kualitas produk, promosi, dan layanan purna jual memiliki pengaruh terhadap pembelian sebesar 27,3% dan dipengaruhi faktor lain sebesar 72,7%. Ditemukan juga pada penelitian (Ramadhan Zulkarnain, 2014), bahwa variabel kualitas produk dan desain produk memiliki kontribusi terhadap keputusan pembelian sebesar 35,7%.

Pengembangan produk adalah salah satu hal yang krusial dan perlu dilakukan oleh pelaku bisnis agar produk yang ditawarkan dapat diterima oleh konsumen dan sebagai bentuk usaha dalam memenuhi kebutuhan dan keinginan konsumen. Dalam mengeksekusi pengembangan produk dan inovasi, terdapat beberapa model dan metode yang dapat digunakan perusahaan. Perusahaan juga perlu memutuskan tahapan untuk mendukung kelancaran dan kesuksesan proses pengembangan dan inovasi produk (Wijaya & Mustamu, 2013).

Kedekatan mengakses dan mudahnya dalam membagikan informasi pada saat ini, merupakan peluang besar sekaligus tantangan bagi pihak yang dapat memanfaatkannya. Dalam beberapa hal yang dilakukan atau keputusan yang dibuat, dipengaruhi oleh opini atau pendapat dari orang lain. Didukung dengan ketersediaan dan familiarnya sumber opini, diantaranya melalui *website review*/ulasan secara *online*, *blog* pribadi, dan media sosial. Ulasan dapat dikatakan merupakan pendapat dari seseorang dan bukan sebuah iklan dan merupakan salah satu faktor yang menentukan keputusan terkait daya beli seseorang (Farki et al., 2016).*Review*/ulasan termasuk ke dalam *big data* yaitu istilah yang berkembang yang mendeskripsikan volum secara terstruktur, *semistructured* dan tidak terstruktur dalam kumpulan data yang memiliki potensi untuk diolah menjadi informasi (Haviluddin, 2016).

Dalam pengembangan produk memahami kebutuhan dan permasalahan pengguna merupakan hal yang diperlukan untuk kemudian dapat memberikan solusi yang nyata. Meningkatnya pemanfaatan teknologi seperti internet, *review*/ulasan konsumen pada suatu produk yang di *posting* melalui media sosial atau *platform marketplace* merupakan informasi yang berharga dan dapat menjadi data masukan bagi pengembangan suatu produk. Seiring perkembangan dan kompleksitas dari suatu produk yang ada di pasaran, analisis data secara manual memiliki banyak keurangan dan memerlukan waktu yang lama. Mesin pintar *(Machine learning)* merupakan metodologi yang digunakan dan memungkinkan komputer untuk mengerjakan tugas yang cerdas hingga menyerupai kerja manusia seperti menyelesaikan tugas-tugas prediksi, deteksi, klasifikasi, dan klasterisasi. Dengan kata lain mesin dapat melakukan tugas analisis statistik banyak dengan banyak data (Haviluddin, 2016).

Pada penelitian ini akan dikembangkan model analisis data dengan pendekatan *machine learning* dan diintegrasi metode desain yaitu *Quality Function Deployment (QFD)* dengan menggunakan data kasus suatu produk buatan Indonesia yaitu sepatu *Ventela Public Low*. *Machine learning* dengan konsep *Natural language processing (NLP)* atau pengolahan bahasa alami merupakan salah satu bidang ilmu *Artificial intelligence* (kecerdasan buatan) yang mempelajari komunikasi antara manusia dengan komputer melalui bahasa alami.

Analisis sentimen akan dilakukan untuk memetakan data dengan algoritma yang digunakan merupakan algoritma yang popular digunakan dalam data berbahasa Indonesia dan memiliki performansi yang baik, yaitu *Naïve Bayes* dan *Support Vector Machine (SVM)*. Pemilihan produk Ventela Public Low dengan pertimbangan yaitu besarnya minat konsumen Indonesia terhadap produk tersebut dibuktikan dengan banyaknya ulasan yang tersedia, baik dijumpai pada *e-commerce* ataupun media sosial. Dikutip dari Krisdamarjati (2020) selama empat tahun terakhir, industri sepatu dalam negeri kian meningkat, dengan munculnya beragam merek baru, dan Ventela merupakan salah satu merek yang banyak disorot. Pihak Ventela menyatakan bahwa jumlah penjualan sepatu dapat menyentuh ratusan pasang dalam satu bulan, dan diestimasikan mencapai 700 hingga 1000 pasang per bulan (Resmi, 2019). Model dan hasil yang didapatkan dari penelitian ini diharapkan dapat membantu *designer* dalam pembuatan suatu produk dan memenuhi kebutuhan konsumen dengan data yang lebih *real-time* dan pada akhirnya dapat meningkatkan kesempatan produk tersebut dapat sukses di pasaran.

1.2 Rumusan Masalah

Melalui latar belakang yang telah dijabarkan, maka rumusan masalah yang tepat adalah sebagai berikut:

Bagaimana hasil dari integrasi model *machine learning* dengan *Quality Function Deployment (QFD)* yang dikembangkan guna membantu pengembangan produk selanjutnya?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dijabarkan sebelumnya, dapat diketahui bahwa tujuan penelitian ini adalah sebagai berikut:

Mengetahui hasil integrasi model *machine learning* dengan *Quality Function Deployment (QFD)* yang dikembangkan guna membantu pengembangan produk selanjutnya.

1.4 Batasan Penelitian

Agar penelitian senantiasa fokus pada objek yang akan diteliti, maka ditentukan batasan masalah sebagai berikut:

1. Penelitian hanya dilakukan dengan data ulasan *online* produk sepatu Ventela Public Low, tidak bisa dibandingkan dengan produk lainnya.
2. Pengklasifikasian data menggunakan metode *Term Frequency Inverse Document Frequency (TFIDF), Bag of Words, Mutual Information, Naïve Bayes* dan *Support Vector Machine (SVM),* tidak sama atau tidak dapat dibandingkan dengan metode lain.
3. Pengembangan produk menggunakan metode *Quality Function Deployment (QFD),* tidak sama atau tidak dapat dibandingkan dengan metode lain.
4. Data ulasan *online* produk sepatu Ventela Public Low dan Converse Chuck Taylor 70s belum dilakukan pada semua situs, namun dilakukan pada *platform* Shopee, Twitter, dan Youtube.

5. Penelitian tidak dilakukan dalam waktu lama dan berkelanjutan, yaitu hanya dilakukan dalam waktu kurang lebih empat bulan.

1.5 Manfaat Penelitian

Dengan jumlah data yang banyak dan kompleks terkait ulasan suatu produk. Melakukan analisis secara manual memerlukan waktu yang cukup lama. Dengan adanya pengembangan model analisis data pada penelitian ini dapat menganalisis data ulasan dengan lebih cepat dan terkini. Penelitian ini dapat membantu *designer*/industri dalam menentukan dan menganalisis data kebutuhan dan keinginan konsumen dalam pembuatan/pembaharuan suatu produk.

1.6 Sistematika Penulisan

Berikut merupakan sistematika penulisan secara umum:

1. BAB I PENDAHULUAN

Pada bab ini membahas latar belakang masalah terkait data ulasan suatu produk, masalah yang ada dalam penelitian yang dilakukan, serta pentingnya penelitian dilakukan. Bab pendahuluan juga berisi rumusan masalah, tujuan penelitian, batasan masalah dari penelitian yang dilakukan, manfaat dari penelitian dan sistematika penulisan.

2. BAB II KAJIAN PUSTAKA

Bab kajian pustaka mengajikan tentang kajian induktif dan deduktif yang digunakan dalam penelitian yang dilakukan. Kajian induktif membahas mengenai penelitian-penelitian yang terdahulu sebagai acuan dalam penelitian ini. Kajian induktif juga dilakukan untuk mencari kesempatan topik penelitian agar tidak mirip dengan penelitian sebelumya. Kajian deduktif mengenai pengertian serta penjabaran keilmuan yang digunakan sebagai panduan penelitian. Kajian pustaka yang dilakukan pada penelitian ini berfokus kepada *machine learning*, pengembangan produk, dan *Quality Function Deployment (QFD).*

3. BAB III METODOLOGI PENELITIAN

Bab metode penelitian membahas mengenai alur penelitian yang dilakukan dalam diagram *flowchart*. Pada bab ini juga secara umum dibahas beberapa hal seperti sumber data, teknik pengumpulan data, pengolahan data, dan usulan strategi.

4. BAB IV PENGUMPULAN DAN PENGOLAHAN DATA

Bab iv ini membahas tentang hasil pengumpulan data yang dilakukan melalui *scraping*. Data yang digunakan pada penelitian ini merupakan data ulasan *online* produk Ventela Public Low, serta keterangan yang dapat membantu dalam penelitian ini.

5. BAB V HASIL DAN PEMBAHASAN

Bab hasil dan pembahasan membahas mengenai data-data yang diolah pada bab iv dan melakukan analisa terhadap data tersebut. Pada bab ini juga dibahas mengenai pengembangan model pengklasifikasian dan kemudian pengembangan produk melalui *Quality Function Deployment (QFD).*

6. BAB VI PENUTUP

Bab penutup berisi mengenai kesimpulan yang membahas tujuan dari penelitian yang dilakukan secara garis besar. Kemudian saran/rekomendasi untuk penelitian selanjutnya.

7. DAFTAR PUSTAKA

Daftar pustaka berisikan semua referensi yang digunakan sebagai pedoman dan acuan penelitian.

8. LAMPIRAN

Lampiran berisikan data-data pendukung maupun dokumen yang dianggap penting untuk dilampirkan dalam laporan penelitian ini.

BAB II

KAJIAN PUSTAKA

2.1 Kajian Deduktif

2.1.1 Produk

Produk merupakan segala sesuatu yang digunakan dalam pemenuhan kebutuhan atau keinginan. Barang terwujud, jasa, *events*, tempat, organisasi, ide atau kombinasi dari beberapa hal tersebut termauk ke dalam produk (Yuliarty et al., 2008). Produk dapat dibutuhkan oleh perorangan, rumah tangga, lembaga, organisasi, pemerintah, atau mungkin lingkungan yang lebih luas lagi. Produk yang telah dimanfaatkan seperti halnya benda lain akan usang dan kemudian digantikan dengan produk yang baru (Sihombing & Sitanggang, 2018). Terdapat dua jenis umur produk yaitu: umur teknis dan umur ekonomis. Umur teknis merupakan umur yang berhubungan dengan daya tahan pemakaian fisik. Umur ekonomis merupakan kemampuan suatu produk yang digunakan memberikan manfaat (nilai) yang lebih besar dibandingkan dengan biaya pemeliharaan produk tersebut (Kosasih, 2009).

2.1.2 Pengembangan Produk

Menciptakan sebuah produk harus selalu disesuaikan dengan kebutuhan dan keinginan pasar yang terus berganti. Dalam hal tersebut menuntut kemampuan perusahaan secara terus menerus menemukan produk baru untuk didesain, dikembangkan, dan dipasarkan. Proses pengembangan produk secara umum merupakan rangkaian langkah dasar yang dibutuhkan dalam melakukan proses desain pada suatu produk. Proses tersebut menggambarkan urutan dasar dari langkah-langkah atau aktivitas yang diberlakukan oleh perusahaan guna menyusun, mendesain, dan menjual produk ke pasar (Sihombing & Sitanggang, 2018). Menurut (Kosasih, 2009), tahapan dalam pengembangan produk yaitu:

1. Pemilihan produk
2. Rancangan awal
3. Konstruksi *prototype*
4. *Testing*
5. *Final* desain

2.1.3 *Big Data*

Big data yang berarti data besar merupakan istilah yang dikenal oleh masyarakat, menggambarkan volum secara terstruktur, *semistructured* dan tidak terstruktur sekumpulan data yang memiliki potensi untuk ditambang/diolah menjadi informasi. *Big data* sering dikaitkan dengan *3V's* yaitu kapasitas data *extreme (volume),* jenis/tipe data (*variety*), dan kecepatan proses data (*velocity*). Aspek *volume* berkaitan dengan asal sumber yang dapat berasal dari beragam sumber, seperti catatan *real-time sensors IoT* dan lain-lain. Aspek *variety*, data yang besar mungkin melibatkan beberapa sumber data simultan yang tidak mungkin jika diintegrasikan. Sedangkan aspek *velocity*, mengacu pada kecepatan data yang besar yang harus dianalisa, di mana proses analisis meniru persepsi dengan mencari dan menggunakan pola dalam data yang dikumpulkan (Haviluddin, 2016).

2.1.4 *Review*/Ulasan *Online*

Ulasan *online* berkaitan pada contoh spesifik di mana pengulas membuat artikel atau kalimat yang berfokus pada produk, dengan tujuan memberikan informasi, membujuk, atau mengingatkan konsumen (Berthon et al., 2008). Ulasan *online* yang bermanfaat didefinisikan sebagai kemampuan yang dirasakan dari ulasan *online* yang membantu pembuat keputusan dalam memahami dan mengevaluasi kualitas dari kinerja produk yang dijual secara *online* (Li et al., 2013).

2.1.5 *Machine Learning (ML)*

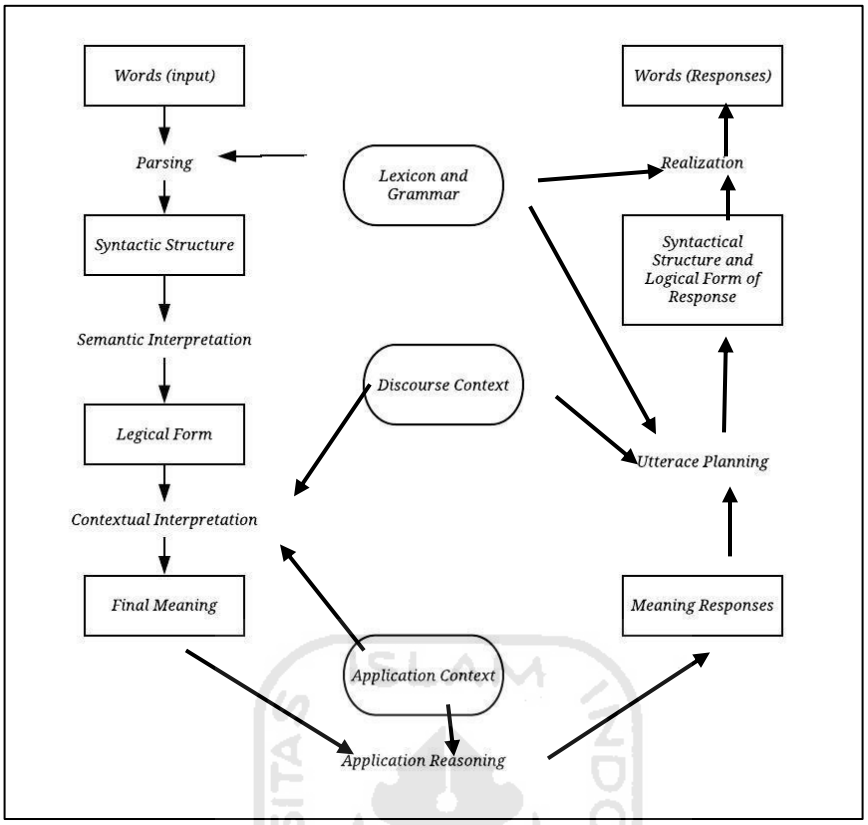
Beberapa langkah dasar dalam *machine learning* yaitu (Sultana et al., 2019):

1. Filtrasi Data, yaitu menyaring data yang mengimpor semua positif dan *dataset* negatif dari *file* dan menggabungkannya ke dalam satu *file*. *Set* data mentah dapat berisi banyak simbol yang tidak diinginkan dan angka. Hal ini perlu diperbaiki atau diselesaikan untuk meningkatkan efisiensi. Dalam proses ini simbol, angka, dan faktor lain yang tidak diinginkan dihapus. Dalam analisis, *dataset* akan dibagi menjadi *dataset* itu sendiri, yaitu data yang digunakan membuat *classifier* belajar dan memprediksinya. *Training dataset*, yaitu data yang memperoleh fitur dari *dataset* pelatihan dan membentuk logika klasifikasi berdasarkan fitur yang diekstraksi untuk mengklasifikasikan ulasan tes yang diberikan sebagai ulasan yang positif atau negatif. *Test dataset* yaitu kumpulan data yang digunakan untuk pengujian.
2. Model pelatihan, *dataset* yang telah disaring kemudian diekstrak dengan mengambil semua kata yang sesuai fitur. Kemudian *dataset* diberi label masing-masing apakah positif , negatif atau netral. Selanjutnya algoritma diberikan sesuai dengan yang diinginkan.
3. Model pengujian, merupakan proses menguji dan menganalisis masing-masing model dengan melakukan *processing* atas *input* data.

2.1.6 *Natural Language Processing (NLP)*

Natural language processing (NLP) atau pengolahan bahasa alami merupakan salah satu bidang ilmu *Artificial intelligence* (kecerdasan buatan) yang mempelajari komunikasi antara manusia dengan komputer melalui bahasa alami. *NLP* tidak memiliki tujuan untuk mengubah bahasa yang diterima dalam bentuk suara menjadi data digital dan/atau sebalinya. Namun memiliki tujuan untuk memahami arti dari ucapan yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, seperti dengan melakukan aksi tertentu atau menampilkan data tertentu.

Untuk meraih tujuan tersebut diperlukan tiga tahap proses. Proses pertama ialah *parsing* atau analisa sintaksis yang memeriksa kebenaran struktur kalimat berdasarkan suatu *grammar* (tata bahasa) dan *lexicon* (kosakata) tertentu. Proses selanjutnya yaitu *semantic interpretation* atau interpretasi semantik yang bertujuan untuk merepresentasikan arti dari kalimat secara *context-independent* guna kepentingan yang lebih lanjut. Proses ketiga yaitu *contextual interpretation* atau interpretasi kontekstual yang bertujuan untuk merepresentasikan arti secara *context-dependent* dan menentukan maksud dari penggunaan kalimat. Gambaran organisasi dari sebuah sistem *NLP* secara lengkap ditunjukkan pada gambar 2.1 berikut.



Gambar 2. 1 Sistem *NLP*
(Sumber: Suciadi, 2001)

2.1.7 Analisis Sentimen

Sentimen berarti pikiran, sikap, atau penilaian yang diungkapkan dengan perasaan. Analisis sentimen dikenal juga sebagai penambangan opini, yang mempelajari sentimen orang terhadap suatu entitas. Analisis sentimen didalamnya terdapat proses pemahaman dan mengklasifikasikan sentimen yang telah ditentukan (Singh et al., 2018). Tujuan dari analisis sentimen adalah untuk menentukan apakah teks yang dihasilkan oleh pengguna bermaksud menyampaikan pendapat positif, negatif, atau netral. Klasifikasi sentimen dapat dilakukan pada tiga tingkat ekstraksi, yaitu tingkat aspek atau fitur, tingkat kalimat, dan tingkat dokumen. Saat ini terdapat tiga pendekatan yang dapat digunakan dalam analisis sentimen, yaitu teknik berbasis leksikon, *machine learning*, dan pendekatan *hybird* (Dang et al., 2020).

Teknik berbasis leksikon merupakan teknik awal dalam analisis sentimen. Teknik ini terbagi menjadi dua pendekatan: berbasis kamus dan berbasis *corpus* (Salas Zarate et al., 2017). Dalam tipe sebelumnya, klasifikasi sentimen dilakukan dengan menggunakan kamus istilah, seperti yang dapat dijumpai di *SentiWordNet* dan *WordNet*. Namun demikian, analisis sentimen berbasis *corpus* tidak bergantung pada kamus yang telah ditentukan tetapi pada analisis statistik dari isi beberapa dokumen, yaitu menggunakan teknik *seperti k-nearest neighbors (k-NN), conditional random field (CRF),* dan *hidden Markov models (HMM)* (Dang et al., 2020).

Teknik berbasis *machine learning* yang digunakan dalam analisis sentimen terbagi menjadi dua kelompok yaitu model tradisional dan *deep learning models*. Model tradisional merupakan model *machine learning* yang klasik seperti *naïve Bayes classifier, maximum entropy,* atau *support vector machine (SVM). Input* untuk algoritma tersebut termasuk fitur leksikal, fitur berbasis sentimen leksikon, *parts of speech*, atau kata sifat dan kata keterangan. Keakuratan dari model ini tergantung pada fitur yang dipilih. Model *deep learning* dapat memberikan performa yang lebih baik daripada model tradisional. Berbagai jenis model *deep learning* yang dapat digunakan untuk analisis sentimen yaitu *convolutional neural network (CNN), deep neural network (DNN),* dan *recurement neural network (RNN).* Pendekatan tersebut mengatasi masalah klasifikasi di level dokumen, level kalimat, atau level aspek. Sedangkan pada pendekatan *hybrid* merupakan menggabungan dari pendekatan berbasis leksikon dan *machine learning*. (Dang et al., 2020).

Berikut merupakan metode *machine learning* di dalam penelitian ini yang termasuk dalam proses analisis sentimen:

A. *Term Frequency Inverse Document Frequency (TFIDF)*

TFIDF dapat diartikan sebagai ukuran statistik yang mencerminkan seberapa penting suatu kata bagi dokumen dalam koleksi atau *corpus*. Metrik ini mempertimbangkan frekuensi kata dalam dokumen target, serta frekuensi dalam dokumen lain dari *corpus*. Semakin tinggi frekuensi kata dalam suatu target tercatat dan semakin rendah frekuensinya di dokumen lain, maka semakin besar kepentingannya (Dang et al., 2020).

Merupakan langkah untuk memberikan skor terhadap frekuensi kemunculan sebuah kata. Pada penelitian ini *TFIDF* digunakan. Pada *term frequency*, setiap dokumen memiliki panjang yang berbeda-beda, bisa saja terjadi sebuah kata muncul lebih banyak di dokumen yang panjang dibandingkan dengan dokumen yang pendek. Pada *term frequency* sering dibagi dengan panjangnya dokumen (total kata yang ada di dokumen tersebut). Pada perhitungan *term frequency*, semua kata dianggap sama pentingnya, namun terdapat kata yang kurang penting dan tidak perlu diperhitungkan, seperti "di-", "ke-", "dan" dan kata yang sejenisnya. Skor *TFIDF* dapat diperoleh menggunakan persamaan 2.1 berikut (Hadna, Santosa, & Winarno, 2016).

$$tf - idf_{t_i d} = tf_{t_i d} \times idf_i \quad \text{....(2.1)}$$

Nilai *TFIDF* berkisar dari nol sampai satu satu dengan presisi sepuluh digit. Setelah dihitung, kata-kata ini diurutkan dalam urutan menurun berdasarkan nilainya. Kemudian, itu dikompilasi ke dalam kamus kata baru dan nilainya. Penyortiran ini penting untuk menganalisis peringkat nilai *TFIDF* dari semua kata untuk memeriksa hasil ringkasan. Setelah mengetahui nilai *TFIDF* dari setiap kata, dapat menghitung nilai pentingnya suatu kalimat. Nilai pentingnya suatu kalimat adalah jumlah dari nilai setiap kalimat kata benda dan kata kerja dalam kalimat. Setiap kalimat dalam dokumen diurutkan dalam urutan menurun (Christian et al., 2016).

B. *Bag of Words (BoW)*

Dalam melakukan pemodelan suatu dokumen, salah satu cara dapat dilakukan melalui *Bag of Words* (kantong huruf), yaitu mengasumsikan bahwa setiap kata yang muncul secara independen dan urutan kata-kata tersebut tidak dipedulikan. Pada penggunaan *BoW*, setiap kata (*term*) akan dihitung bobotnya. Cara sederhana adalah dengan menghitung frekuensi kemunculan *term* dalam dokumen dan kemudian menggunakannya sebagai bobot *term* tersebut. Dokumen tersebut kemudian digambarkan sebagai suatu vektor t_d. Jika $tf_{(d,t)}$, menyatakan frekuensi dari *term* t dalam dokumen d, maka representasi vektor suatu dokumen d dinyatakan sesuai persamaan 2.2.

$$t_d = (tf_{d,t_1}, \dots, tf_{(d,t_m)}) \quad \text{....(2.2)}$$

Jika D = {d_1, … , d_n} merupakan satu kesatuan dokumen, T = {t_1, … , t_m} merupakan satu kesatuan *term* unik yang muncul dalam *set* dokumen D, dan tf(d,t) frekuensi *term* t € T dalam dokumen d € D, maka representasi vektor dokumen d dinyatakan dengan persamaan di atas (Ryansyah & Andayani, 2017)

C. *Mutual Information (MI)*

Seleksi fitur merupakan salah satu faktor yang cukup dapat diperhitungkan dalam mempengaruhi tingkat akurasi klasifikasi karena jika *dataset* berisi sejumlah fitur, dimensi ruang akan menjadi besar, kemudian berakibat membuat tingkat akurasi menjadi rendah (Liu, et al., 2011). Dalam proses seleksi fitur beberapa aspek yang mempengaruhi yaitu pola klasifikasi, akurasi klasifikasi, waktu yang diperlukan untuk pembelajaran fungsi klasifikasi, jumlah contoh yang dibutuhkan untuk pembelajaran dan biaya yang berhubungan dengan fitur (Zhao et al., 2011).

Tiga model pendekatan dalam seleksi fitur yaitu *filter method, wrapper method*, dan *embedded method*. Pada *filter method* mengevaluasi fitur dengan langkah membentuk *ranking* fitur secara independen terhadap kelas dalam *training set*. *Wrapper method* menggunakan metode-metode *Artificial Intelligence* untuk mencari fitur yang terbaik dengan proses *searching* secara *iterative*. Kemudian *embedded method* menggunakan pendekatan prediksi linear secara simultan untuk meningkatkan *goodness-of-fit* dan menurunkan jumlah fitur masukkannya (Khalid et al., 2015). Dalam sejumlah alat dan teknik yang telah digunakan untuk pemrosesan bahasa antara lain: *entropy, perplexity, mutual information, traditional statistics, connectionism, genetic algorithms, formal language theory, Markov Modelling*, dan *non-linear dinamical modelling*. Diantara teknik-teknik tersebut terdapat hubungan secara matematika (Ziveria, 2014).

Mutual information termasuk kedalam pendekatan seleksi fitur *wrapper method. Mutual information* membandingkan peluang pengamatan x dan y secara bersama dengan peluang pengamatan secara bebas. Apabila ada hubungan yang kuat antara pengamatan x dan y, maka P(x,y) akan lebih besar dari P(x).P(y) dan sebagai akibatnya I(x,y) > 0. Apabila tidak ada hubungan keterikatan antara pengamatan x dan y maka P(x,y) = P(x).P(y) dan I(x,y) = 0. Dalam penerapannya, peluang kata P(x) dan P(y) diperkirakan

dengan menghitung jumlah kata x dan jumlah kata y yang ada pada suatu artikel, yaitu f(x) dan f(y)., lalu dinormalkan dengan N, yaitu jumlah seluruh artikel. Peluang bersama P(x,y) diperkirakan dengan menghitung berapa kali kata x yang diikuti oleh kata y yang dinotasikan dengan f_w.(x,y) dalam suatu dokumen dengan jumlah w kata-kata, lalu dinormalkan dengan N (Widiastuti, 2018). Persamaan 2.3 berikut merupakan representasi dari metode *mutual information*.

$$I(x,y) = \log \frac{f_w(x,y)/N}{(\frac{f(x)}{N})(\frac{f(y)}{N})} \quad \ldots.(2.3)$$

D. *Naïve Bayes (NB)*

Metode *Naïve Bayes* merupakan metode yang dipakai dalam pengklasifikasian. Metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris Thomas Bayes, yaitu melalui prediksi probabilitas dimasa depan berdasarkan pengalaman dimasa sebelumnya. Konsep dasar teori *bayes* yaitu peluang bersyarat P(X|C), dimana X adalah *posterior* dan C adalah *prior*. *Prior* mengartikan pengetahuan tentang karakteristik suatu parameter atau pengalaman dimasa lalu, sedangkan posterior adalah karakteristik yang diduga pada kejadian yang datang (Negara et al., 2020).

$$p(C_j|d_j) = \frac{p(c_j|d_j)_{p(C_j)}}{p(d_i)} \quad \ldots.(2.4)$$

Persamaan 2.4 diilustrasikan data *training* yang dikategorikan menjadi beberapa kategori C_j={C_1, C_2, C_a, … , C_k}, dan prior probability untuk setiap kategori adalah p(C_j), dimana j=1,2,3, … ,k. Sedangkan untuk simbol d_i=(W_1, …, W_j, … , W_m), dan kata yang terdapat pada dokumen berupa W_j, dimana j=1,2,3, … m, masuk ke dalam kategori C_j. Untuk mengklasifikasi dokumen d_i, dilakukan dengan menghitung nilai probabilitas dari semua dokumen *(posterior probability).* Sehingga *posterior probability* suatu dokumen pada suatu kategori dapat dihitung dengan persamaan 2.4 (Chen et al., 2009).

E. *Support Vector Machine (SVM)*

SVM digunakan dalam analisis regresi, dan klasifikasi untuk satu *set* perangkat pelatihan dengan berbagai bentuk dan kategori. *SVM* termasuk klasifikasi yang memberi aturan dalam mengubah dokumen teks menjadi *vector* sebelum diklasifikasi. Langkah dasar dalam SV*M* yaitu (Sadhasivam & Kalivaradhan, 2019):

1. Menemukan berat untuk setiap titik pelatihan, dan titik-titik yang beratnya menjadi nol bukan merupakan *support vector*, seperti kepentingannya nol selama waktu pengujian, dan sisanya adalah *support vector*.
2. Mengubah setiap vektor menjadi beberapa vektor dimensi tinggi lainnya, kemudian ambil titik produk antara dua *support vector* berdimensi tinggi yang kompleks.
3. Menemukan maksimum vektor sebagai nilai sentimen.

Metode *SVM* secara umum memiliki kelebihan dan kekurangan. Kelebihan *SVM* yaitu: (1) dapat menangani dokumen dengan *input space* yang berdimensi tinggi dan memilih banyak *irrelevant* fitur, (2) memiliki kemampuan untuk menghasilkan akurasi klasifikasi tinggi dibandingkan dengan algoritma klasifikasi yang lain. Sedangkan kekurangan *SVM* yaitu: (1) waktu dan memori dari *traning* dan klasifikasi membutuhkan persyaratan yang sebanding dengan *decision trees*, tapi memang lebih mahal daripada *naïve bayes* dan algoritma lain, (2) sulit dalam menentukan parameter *SVM* (Choudhari & Dhari, 2017).

Dalam metode *SVM* klasifikasi dilakukan dengan mencari *hyperplane* atau garis pembatas *(decision boundary)* yang memisahkan antara satu kelas dengan kelas yang lain. *SVM* berusaha untuk mencari *hyperplane* terbaik diantara fungsi yang tidak terbatas jumlahnya. Fungsi yang tidak terbatas dalam pencarian *hyperplane* dalam metode *SVM* merupakan sebuah kelebihan, karena pemrosesan akan selalu bias dilakukan bagaimanapun data yang dimiliki. Data yang digunakan pada metode *SVM* dengan $x_i \in R^d$ dan label dinotasikan dengan $y_i \in \{1,2\}$ untuk $i = 1,2,\ldots.l$ dimana l adalah banyaknya data. *SVM* dapat diformulasikan dalam persamaan 2.5 untuk $y_i = +1$ dan persamaan 2.6 untuk $y_i = -1$ (Hadna et al., 2016).

$$x_i \times w + b \geq +1 \quad \text{....(2.5)}$$

$$x_i \times w + b \leq 11 \quad \text{....(2.6)}$$

2.1.8 Asosiasi Kata

Data dengan kelas negatif akan dicari asosiasi katanya bertujuan untuk memudahkan proses dalam *QFD* dan didapatkan informasi yang lebih baik yang mewakili fitur yang perlu diperbaiki. Proses asosiasi dilakukan dengan metode yaitu perhitungan frekuensi, *Pointwise Mutual Information (PMI)*, dan pengujian hipotesis *t-test*, pengujian hipotesis *chi-square* dan *likelihood* (Ruchirawat, 2018).

1. Memberikan Peringkat Kolokasi (Frekuensi)

Menghitung frekuensi kata yang berdekatan, dengan metode paling sederhana adalah memberi peringkat *bigram* atau *trigram*.

2. *Pointwise Mutual Information*

Intuisi utamanya adalah mengukur seberapa besar kemungkinan kata-kata itu muncul bersama daripada jika mereka independen. Namun, sangat sensitif terhadap kombinasi kata yang langka. Misalnya, jika *bigram* acak 'abc xyz' muncul, dan baik 'abc' maupun 'xyz' tidak muncul di tempat lain dalam teks, 'abc xyz' akan diidentifikasi sebagai *bigram* yang sangat signifikan ketika itu bisa berupa kesalahan ejaan acak atau ungkapan itu terlalu jarang digeneralisasikan sebagai *bigram*. Rumus *PMI* dalam menghitung *bigram* seperti pada persamaan 2.7 dan persamaan 2.8 untuk menghitung *trigram*.

$$PMI\ (w^1, w^2) = log_2 \frac{P\ (w^1, w^2)}{P\ (w^1) P(w^2)} \quad \text{....(2.7)}$$

$$PMI\ (w^1, w^2, w^3) = log_2 \frac{P\ (w^1, w^2, w^3)}{P\ (w^1) P(w^2) P(w^3)} \quad \text{....(2.8)}$$

3. Pengujian Hipotesis

a. *T-test*

Pertimbangkan jika kita memiliki korpus dengan N kata, sebagai contoh pada kata sosial media memiliki jumlah kata C (sosial) dan C (media) masing-masing. Dengan asumsi hipotesis nol dengan *social* dan media menjadi independen. Persamaan dalam *t-test* seperti pada persamaan berikut.

H_0 : *'social media'* muncul dengan probabilitas =

$$1\mu = P(social)\ P(media) = \frac{C(sosial)}{N} \times \frac{C(media)}{N} \quad \ldots.(2.9)$$

H_a : *'social media'* tidak muncul sesuai dengan probabilitas μ yang diprediksi

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} \quad \ldots.(2.10)$$

$$\bar{x} = \frac{C(social\ media)}{N} \quad \ldots.(2.11)$$

Untuk Bernoulli trial : $s^2 = p(1-p) \sim p \sim \bar{x}$

b. *Chi-square*

Uji *chi-square* mengasumsikan dalam hipotesis nol bahwa kata-kata independen, sama seperti dalam uji-t. Statistik uji *chi-square*. Persamaan dalam *chi-square* seperti pada persamaan berikut.

$$x^2 = \sum_{i,j} \frac{(O_{ij} - O_{ij})^2}{E_{ij}} \quad \ldots.(2.12)$$

Keterangan:

i dan j adalah semua baris dan kolom pada tabel

O_{ij} merupakan nilai yang diobservasi pada tabel i dan kolom j

E_{ij} Merupakan nilai yang diperkirakan jika kata adalah independen, yaitu

$E(social\ media) = \frac{C(social)}{N} \times \frac{C(media)}{N} \times N$

Dimana N merupakan total jumlah kata dalam korpus

4. *Likelihood*

Menghitung probabilitas tabel kontingensi yang teramati (atau bagian dari) menurut non hipotesis asosiasi lengkap (biasanya hipotesis independen terhadap H_0)

2.1.9 *Quality Function Deployment (QFD)*

Quality Function Deployment (*QFD*) adalah sebuah metode perencanaan dan pengembangan produk atau jasa secara terstruktur. Dimana memungkinkan tim pengembang mendefinisikan kebutuhan dan harapan secara jelas dan dapat mengevaluasi kemampuan produk atau jasa secara sistematik untuk memenuhi kebutuhan dan harapan tersebut yang berasal dari konsumen (Arini, 2003). Metode *QFD* memberikan sejumlah manfaat bagi organisasi yang mencoba untuk meningkatkan daya saingnya dengan memperbaiki kualitas dan produktivitasnya. Beberapa manfaatnya diantara lain: fokus pada pelanggan, efisiensi waktu, orientasi kerjasama tim, dan orientasi pada dokumentasi (Goetsch & Davis, 2000).

QFD sebagai suatu proses atau mekanisme terstruktur untuk menentukan kebutuhan pelanggan dan menerjemahkan kebutuhan-kebutuhan tersebut ke dalam kebutuhan teknis yang relevan, dimana masing-masing area fungsional dan level organisasi dapat mengerti dan bertindak (Gasperz, 1998). *QFD* merupakan suatu pendekatan disiplin namun fleksibel terhadap pengembangan produk. Titik awal dari *QFD* yaitu pelanggan serta keinginan dan kebutuhan dari pelanggan tersebut (Goestch & Davis, 2000).

2.2 Kajian Induktif

Berikut merupakan pengkajian penelitian terdahulu, bertujuan untuk mengetahui posisi dan petunjuk atas topik penelitian yang akan dilakukan.

Pada penelitian yang dilakukan oleh Irish et al. (2017) membahas tentang analisis sentimen menggunakan *review*/umpan balik pengguna dalam aplikasi belanja *online* dan mengklasifikasikan menggunakan *Support Vector Machine (SVM)*. Dalam penelitian tersebut diusulkan pembelajaran kata sentimen khusus yang kemudian dinamakan sebagai *sentiment embeddings*, yaitu dengan mempertahankan keefektifan konteks kata dan memanfaatkan analisis sentimen teks untuk menentukan kata berkelanjutan yang lebih kuat representasinya. Eksperimen dilakukan dengan satu *set* data besar dengan berbagai ulasan dan kemudian diberikan sebagai data pelatihan ke *classifier*. Hasil dari penelitian ini yaitu model yang dikembangkan dapat secara signifikan mengurangi waktu

yang digunakan untuk analisis ulasan dan memberikan lebih banyak hasil yang akurat dan dapat diandalkan. Namun, terdapat batasan yaitu *classifier trained* tidak bekerja efisien di *domain* lain dan pada penelitian tersebut hanya terdapat dua klasifikasi yaitu positif dan *negative*.

Penelitian yang membahas tentang pendekatan pembelajaran mesin untuk disambiguasi ulasan produk berdasarkan klasifikasi fungsi, bentuk, dan perilaku, yang dilakukan oleh Singh dan Tucker (2016), memiliki tujuan yaitu mengusulkan pendekatan pembelajaran mesin yang memisahkan ulasan pelnggan *online* dengan mengklasifikasikan menjadi salah satu dari tiga karakteristik produk langsung (yaitu bentuk, fungsi, atau perilaku) dan dua karakteristik produk tidak langsung (yaitu layanan dan lainnya), kemudian menemukan algoritma pembelajaran mesin yang memberikan hasil tertinggi dan paling dapat digeneralisasi dalam meraih tujuan, dan untuk mengukur korelasi antara peringkat produk dan karakteristik produk langsung dan tidak langsung. Pendekatan yang digunakan yaitu pendekatan *Multilayerd*, dengan penggunaan *Naïve Bayes, Support Vector Machines, Decision Tress* dan *IBk Classification Algorithms*. Dengan data yang diperoleh dari Amazon.com didapatkan tingkat akurasi klasifikasi yaitu 82,44% untuk klasifikasi produk, 80,84% untuk seluruh klasifikasi produk, 79,03% untuk seluruh jenis produk klasifikasi dan 80,64% untuk seluruh klasifikasi domain produk. Penelitian ini memiliki keterbatasan yaitu mengabaikan sinonim dan polisemi karena dapat mengurangi kinerja model.

Penelitian yang dilakukan oleh Sultana, Kumar et al. (2019) membahas tentang survei rinci berbagai teknik atau pendekatan yang digunakan dalam analisis sentimen dan mengusulkan teknik baru. Pada penelitian tersebut menggunakan pendekatan tingkat dokumen untuk menemukan polaritas kalimat atau dokumen dalam hal sentimen positif atau negatif. Setelah melalui proses perbandingan teknik, diusulkan model dengan menggunakan *dataset* Stanford berupa 50000 ulasan film, diperoleh hasil bahwa kombinasi kata sifat, kata keterangan dan kata kerja menjadi kombinasi terbaik di antara berbagai kombinasi.

Kumar et al. (2020) dengan penelitian berjudul *Sentiment Analysis of Amazon Product Reviews using Machine* digunakan teknik *supervised gaining information* pada data

Amazon yang berskala besar untuk mempolarisasi dan mendapatkan akurasi yang baik. Dalam mengekstrak *skill* data, digunakan campuran teknik yaitu *bag of phase techniques dan tf-idf & chi square* dan teknik pengklasifikasian yang digunakan yaitu *naïve bayesian* dan *SVM*. Didapatkan hasil memungkinkan untuk analisis, klasifikasi yang masuk akan untuk volum data *file* heterogen, membantu memotong proses statistik dalam penulisan analisis, analisis yang memerlukan kualitas akun dari suatu pandangan publik , distribusi secara mekanis polaritas sentimen, dan memiliki akses ke salah satu atau semua kemampuan teks yang menarik.

Sadhasivam & Kalivaradhan (2019) dengan penelitian yang berjudul *Sentiment Analysis of Amazon Products Using Ensemble Machine Learning Algorithm* memiliki tujuan membantu memberikan akurasi yang lebih baik daripada algoritma yang ada saat ini. Penelitian tersebut mengusulkan sebuah metode *Ensemble*, dan kemudian menggabungkan metode tersebut dengan *Naïve bayes* dan *SVM*. Hasil yang diperoleh metode yang diusulkan, mampu menghasilkan akurasi yang lebih baik daripada sistem yang ada, namun tingkat akurasi bervariasi dengan jumlah total pengklasifikasi yang digabungkan oleh peneliti dalam mendapatkan prediksi *output* ulasan. Pada penelitian ini merekomendasikan untuk klasifikasi tingkat aspek.

Siringoringo & Jamaluddin (2019) dengan penelitian yang berjudul *Text Mining* dan Klasterisasi Sentimen Pada Ulasan Produk Toko *Online* dengan data ulasan produk TrendyShoes sepatu anak merek denim dan teknik klasifikasi yang digunakan yaitu *SVM* dan *K- Nearest Neighor (KNN)*, pembobotan *TF&IDF*, dan *Coefficient feature selection (CFS)*. Hasil yang diperoleh dengan pengujian pada tiga jenis data bahwa *SVM* dapat bekerja dengan baik pada data ulasan yang tidak seimbang. Klasifikasi sentimen menggunakan *SVM* lebih unggul dari *KNN*.

Kouw et al. (2016) pada penelitian yang berjudul *Feature-Level Domain Adaption* peneliti mengusulkan pendekatan yang diberi nama sebagai *Feature-Level Domain Adaptation (FLDA)*, model tersebut memodelkan ketergantungan antara dua *domain* dengan menggunakan model transfer level-fitur yang dilatih untuk menggambarkan perpindahan dari sumber ke *domain* target. Pada penelitian ini digunakan *set data* ulasan

alat dapur. Dari eksperimen pada penelitian *FLDA* bekerja setara dengan keadaan metode saat ini untuk adaptasi *domain*.

Pada penelitian oleh Santosh (2018) dengan judul *A Combined Approach for Effective Features Extraction from Online Product Reviews* memiliki tujuan membuat model yang mencakup lebih banya fitur produk sehingga memaksimalkan kepuasan pelanggan dengan *set* fitur yang dimiliki. Dalam penelitian tersebut digunakan penggabungan *Larent Dirichet Allocation (LDA)* dengan *Feature Ontology Tree (FOT)*. Hasil yang didapatkan yaitu penggabungan kedua pendekatan *NLP* dan *LDA+FOT* sukses dilakukan dan berguna dalam menentukan kesamaan antara produk yang dicari dan basis *case* yang disimpan pada saat rekomendasi produk.

Penelitian oleh Xu, Zhang, & DeGroof (2018), peneliti mengusulkan pendekatan pembelajaran untuk menganalisis secara efisien dan efektif ulasan produk *online* dan mengidentifikasi kekuatan dan kelemahan suatu prosuk oleh fitur produknya. Pendekatan yang digunakan adalah *Feature-based Sentence Model (FSM)*. Dengan menggunakan data yang diperoleh dari Amazon dengan produk yaitu "Kamera Digital Nikon Coolpix L340 (hitam)", dalam dua studi kasus pendekatan yang diusulkan lebih efektif dibandingkan pendekatan yang ada, dan memberi konsumen cara yang jauh lebih mudah untuk menemukan fitur produk pada produk *online* yang paling diinginkan.

Penelitian oleh Ma, Zhang et al. (2013) yang memiliki judul *An LDA and Synonym Lexicon Lexicon Based Approach to Product Feature Extraction from Online Costumer Product Reviews*. Metode yang digunakan mengidentifikasi frase kata benda dan kata benda dalam ulasan konsumen dan membuat fitur-fitur dengan *LDA*, dan kemudian lebih lanjut dengan sinonim leksikon. Pada penelitian tersebut *LDA* digunakan dengan cara yang berbeda, dan membuat pendekatan yang diusulkan mampu mengekstraksi fitur produk yang tidak ada dalam deskripsi produk. Data yang digunakan adalah tiga kamera digital yang berbeda dan satu ponsel. Penelitian tersebut menjadi langkah awal mengeksplorasi ekstraksi fitur untuk ulasan produk konsumen *online* untuk sepenuhnya memanfaatkan media sosial dan meningkatkan pembuatan keputusan oleh konsumen.

Rintyarna et al. (2019) dalam penelitian yang berjudul *Evaluating the performance of sentence level features (SLF) and domain sensitive features (DSF) of product reviews on supervised sentiment analysis tasks*, dalam penelitian tersebut peneliti mengusulkan metode untuk mengekstraksi fitur teks bernama *SLF* dan *DSF* yang mempertimbangkan semantik kata dalam tingkat kalimat dan tingkat domain dari ulasan produk. Hasil dari percobaan yang dilakukan bahwa fitur semantik yang diusulkan dapat meningkatkan kinerja analisis sentimen (*supervised*) pada ulasan produk.

Penelitian yang dilakukan oleh Adetunmbi et al. (2019), mengenai analisis sentimen pada ulasan film dengan menggunakan *Naïve Bayes, Support Vector Machine(SVM),* dan *K-Nearest Neighboors(KNN)* dan teknik ekstraksi fitur *Term Inverse Document Frequency (TFIDF).* Hasil yang diperoleh menunjukkan bahwa *KNN* dengan *TFIDF* memiliki kinerja senilai 95,9% dan merupakan yang lebih baik dibandingkan *Naïve* dan *SVM*.

Penelitian oleh Widyawan et al. (2019) yang berisi tentang perbandingan pembobotan fitur dalam kinerja *SVM*, dalam penelitian tersebut memanfaatkan *TFIDF, TF-CHI (Term Frequency Chi-Square), TF-RF (Term Frequency Relevansi Frekuensi)* dan *TF-OR (Term Frequency Odd Ratio)*. Data yang digunakan adalah ulasan tentang Jakarta BRT yang didapatkan dari Twitter. Hasil dari percobaan yang dilakukan, *TFIDF* memiliki kinerja tertinggi dan memberikan efek pada kinerja yang lebih baik.

Penelitian oleh Najib et al. (2019), menggunakan *dataset* yang digunakan sebanyak 1050 *tweet* terkait topik kampanye pilpres Indonesia. Berdasar hasil penelitian yang dilakukan, akurasi berdasarkan metode *lexicon-based* sebesar 39% dan metode *SVM* sebesar 83%. Hasil *lexicon-based* menunjukkan bahwa *sentiment* pada mayoritas atribut adalah *negative*. Sedangkan hasil *SVM* menunjukkan bahwa sentimen pada mayoritas atribut (finansial dan kesejahteraan) termasuk positif dan sisanya (lapangan kerja) berupa netral. Kekurangan dari penelitian tersebut adalah pembuatan ontologi masih dilakukan secara manual dan belum adanya proses ekstraksi yang baik.

Tabel 2. 1 Penelitian Terdahulu

No	Penelitian, tahun	Kata Kunci																												
		Data Ulasan	**Sentimen**	**Produk**	**Jasa**	***Naive***	***SVM***	***Decision Tree***	***TF&IDF***	***Sentiment Embeddings***	***IBk Classification***	***Linear Model Algoritma***	***Bag of Phase***	***Chi square***	***Ensemble Learning***	***KNN***	***CFS***	***FLDA***	***LDA***	***FOT***	***FSM***	***Sinonim Leksikon***	***SLF***	***DSF***	***TF-RF***	***TF-OR***	***QFD***	***Mutual Information***	***Bag of Words***
1	(Irish, Sherin, Surya, Vidhya, & Ramamoorthy, 2017)	Ingris	√	√						√																				
2	(Singh & Tucker, 2016)	Ingris	√	√		√	√	√			√																			
3	(Sultana, Kumar, Patra, Chandra, & Alam, 2019)	Ingris	√		√	√	√	√				√																		
4	(Kumar, Jagadeesh, Kshirsagar, & Marve, 2020)	Ingris	√	√		√	√		√				√	√																
5	(Sadhasivam &	Ingris	√	√		√	√								√															

No	Penelitian, tahun	Kata Kunci																										
		Data Ulasan	**Sentimen**	**Produk**	**Jasa**	***Naive***	***SVM***	***Decision Tree***	***TF&IDF***	***Sentiment Embeddings***	***IBk Classification***	***Linear Model Algorithma***	***Bag of Phase***	***Chi square***	***Ensemble Learning***	***KNN***	***CFS***	***FLDA***	***LDA***	***FOT***	***FSM***	***Sinonim Leksikon***	***SLF***	***DSF***	***TF-RF***	***TF-OR***	***QFD***	***Mutual Information***	***Bag of Words***
	Kalivaradhan, 2019)																											
6	(Siringoringo & Jamaluddin, 2019)	Indonesia	√	√			√		√							√	√											
7	(Kouw, van der Maaten, Krijthe, & Loog, 2016)	Ingris	√	√		√	√											√										
8	(Santosh, 2018)	Ingris		√															√	√								
9	(Xu, Zhang, & DeGroof, 2018)	Ingris	√	√																	√							
10	(Ma, Zhang, Yan, & Kim, 2013)	Ingris		√			√												√			√						
11	(Rintyarna,	Ingris	√	√		√																	√	√				

No	Penelitian, tahun	Kata Kunci																											
		Data Ulasan	**Sentimen**	**Produk**	**Jasa**	***Naive***	***SVM***	***Decision Tree***	***TF&IDF***	***Sentiment Embeddings***	***IBk Classification***	***Linear Model Algorithma***	***Bag of Phase***	***Chi square***	***Ensemble Learning***	***KNN***	***CFS***	***FLDA***	***LDA***	***FOT***	***FSM***	***Sinonim Leksikon***	***SLF***	***DSF***	***TF-RF***	***TF-OR***	***QFD***	***Mutual Information***	***Bag of Words***
	Sarno, & Faticah, 2019)																												
12	(Adetunmbi, Sarumi, Olutomilola, & Boyinbode, 2019)	Inggris	√		√	√	√		√							√													
13	(Widyawan, Damayanti, Adji, & Putra, 2019)	Indonesia	√		√		√		√					√											√	√			
14	(Najib, Irsyad, Qandi, & Rakhmawati, 2019)	Indonesia	√		√		√												√										
15	(Maharditingtyas, 2020)	Indonesia	√	√		√	√		√																		√	√	√

Analisis data melalui sentimen adalah bidang riset yang kian berkembang. Tujuan dari analisis sentimen yaitu untuk menemukan opini/pendapat yang ada dalam tulisan teks, mendapatkan sentimen yang ada dalam opini tersebut, dan akhirnya mendapatkan polaritas/klasifikasi atas sentimen tersebut apakah positif/mendukung atau negatif/tidak mendukung. Algoritma yang sering digunakan terbagi menjadi dua yaitu algoritma berbasis kamus dan berbasis pembelajaran mesin. *Support Vetor Machine* dan *Naïve Bayes* merupakan algoritma yang umum digunakan pada analisis sentimen dengan data yang digunakan berbahasa Indonesa, dari beberapa penelitian yang telah dilakukan algoritma klasifikasi dapat ditingkatkan kinerjanya salah satunya dengan *mutual information* yang berperan sebagai seleksi fitur.

Pada penelitian ini akan dikembangkan model analisis data ulasan *online* dengan pendekatan *mutual information* sebagai pemilihan fitur, *TFIDF* dan *Bags of Words (BoW)* sebagai pembobotan kata, kemudian *SVM* dan Naïve sebagai *classifier*. Dengan beberapa pendekatan tersebut, akan dilakukan serangkaian percobaan untuk mendapatkan model dengan performa paling baik dan diharapkan dapat meningkatkan proses analisis sentimen pada data ulasan *online*. Selanjutnya, untuk melengkapi proses pengembangan produk kaitannya dengan desain produk, hasil analisis sentimen tersebut akan berkedudukan sebagai *Voice of Costumer* yang digunakan sebagai masukan dalam metode *Quality Function Deployment (QFD)*. Integrasi analisis data menggunakan *machine learning* dengan metode *QFD*, diusulkan peneliti dalam rangka membantu pemilik usaha pada proses desain produk dengan mengoptimalkan data yang ada dan dengan proses yang lebih baik.

BAB III

METODE PENELITIAN

3.1 Objek Penelitian

Penelitian Analisis Data pada Ulasan Produk *Online* Berbasis *Machine Learning* untuk Menunjang Proses Awal Desain Produk menggunakan studi kasus produk asal Indonesia yaitu sepatu Ventela. Sepatu Ventela merupakan salah satu produk yang populer baik secara penjualan maupun *branding*. Penelitian ini berfokus pada varian sepatu Ventela yaitu *Public Low*.

3.2 Identifikasi Masalah

Sepatu Ventela merupakan salah satu produk lokal yang mulai dikenalkan pada tahun 2017. Sepatu Ventela diproduksi oleh pabrik bernama PT. Sinar Runerindo yang berlokasi di Bandung. Dalam mengembangkan bisnis, dapat dilakukan salah satunya melalui pengembangan produk. Desain produk merupakan aspek penting yang mendukung suksesnya produk di pasaran. Identifikasi masalah pada penelitian ini dilakukan berdasarkan kajian studi dan tren tentang *machine learning* sebagai metode yang dapat membantu meningkatkan nilai suatu produk dan kemudian akan di integrasikan dengan metode *QFD* sebagai simulasi perolehan spesifikasi teknis.

3.3 Kajian Literatur

Pada penelitian ini memiliki dua jenis kajian literatur yakni kajian induktif dan kajian deduktif. Kajian induktif berisikan tentang penelitian-penelitian sebelumnya yang telah dilakukan oleh para peneliti untuk memperoleh informasi mengenai metode, *tools* yang digunakan, dan juga dapat digunakan untuk melakukan *benchmarking* atau mencari perbedaan antara penelitian-penelitian yang telah dilakukan. Kajian deduktif membahas terkait ilmu atau teori dasar untuk melakukan penelitian agar dapat mendalami landasan-landasan teoritis dan tahapan yang digunakan pada penelitian ini. Kajian literatur ini dapat berupa buku, jurnal, *paper*, metode, dan juga *guidance* yang telah resmi dan dapat dipercaya. Studi literatur yang ada pada penelitian ini berkonsentrasi kepada beberapa topik yaitu *machine learning*, analisis sentimen, *TFIDF, Bags of Words, Mutual Information, SVM, Naïve Bayes* dan metode pengembangan produk yaitu *QFD*.

3.4 Metode Pengumpulan Data

Pada metode pengumpulan data, digunakan untuk mengategorikan data atau sumber informasi perolehan data.

3.4.1 Data Primer

Data yang termasuk ke dalam data primer yaitu data yang diambil secara langsung melalui kuesioner. Kuesioner dilakukan dengan tanpa adanya perantara atau responden memberikan langsung data kepada pengumpul data. Pada penelitian ini data kuesioner digunakan untuk memperoleh data berkaitan dengan media sosial dan *marketplace* yang digunakan pengguna produk dan berkaitan dengan penilaian tingkat kepentingan kriteria pada metode *QFD*.

3.4.2 Data Sekunder

Data sekunder merupakan data yang dihasilkan melalui perantara ataupun data-data yang didapatkan dalam dokumen-dokumen organisasi, artikel-artikel, serta buku. Data sekunder berfungsi untuk memperkuat deskripsi kualitatif dan kuantitatif dalam

penelitian. Pada penelitian ini *dataset* ulasan produk sepatu Ventela dan Converse yang diperoleh melalui media sosial Twitter dan Youtube termasuk ke dalam data sekunder.

3.4.3 Teknik Pengumpulan Data

Terdapat beberapa teknik dalam pengumpulan data yang telah disesuaikan dengan rangkaian penelitian yang dilakukan. Berikut merupakan teknik pengumpulan data pada penelitian ini.

1. *Scraping*

Merupakan proses ekstrasi data dari sebuah *website*. Proses ini dilakukan dengan semakin berkembangnya *big data*, jumlah data yang tersedia sudah tidak terhitung lagi. Melalui proses *scraping* dapat membantu mengumpulkan data dengan lebih cepat karena dapat melakukan *automation*. Pada penelitian ini *scraping* dilakukan untuk memperoleh data ulasan produk Ventela public low. Sebanyak 10412 data ulasan didapatkan melalui *web* Twitter dan Youtube, data yang diambil pada rentang waktu maret 2018 sampai dengan maret 2020, dengan menggunakan *keyword* ventela saat pencarian.

2. Studi Literatur

Proses studi literatur berguna dalam pelaksanaan penelitian dengan mencari informasi-informasi dengan sumber dari penelitian terdahulu, buku, jurnal, *paper*, serta *guidance*.

3.5 Pengolahan Data

3.5.1 Pelabelan Data

Data yang telah dikumpulkan selanjutnya akan dilakukan pelabelan. Proses ini bertujuan untuk membagi data menjadi beberapa kelas sentimen yang akan digunakan dalam penelitian. Pada penelitian ini akan dibagi menjadi tiga kelas yaitu positif, netral, dan negatif. Proses pelabelan juga berfungsi membagi *dataset* menjadi 2 bagian yaitu *training*

data dan *testing data*. Pada penelitian ini pelabelan tidak dilakukan dengan melibatkan ahli atau pakar bahasa, dikarena keterbatasan waktu dan biaya.

3.5.2 Data *Pre-processing*

Adalah teknik *data mining* yang melibatkan transformasi data mentah menjadi bentuk yang mudah dimengerti. Tahapan data *pre processing* dibutuhkan untuk menyelesaikan beberapa jenis masalah yang muncul seperti *noisy data*, data redundansi, nilai data yang hilang, dan lain-lain (Kotsiantis et al., 2006). Berikut merupakan tahapan dalam melakukan data *pre-processing*:

1. *Case Folding*

Dalam proses ini dilakukan pengubahan semua huruf dalam dokumen atau kalimat menjadi huruf kecil. *Case folding* digunakan untuk mempermudah pencarian.. Dengan alasan tidak semua data konsisten dalam penggunaan huruf kapital (Gunawan et al., 2018).

2. Tokenisasi

Merupakan proses pemisahan suatu karakter berdasarkan karakter spasi, dan juga dapat pada waktu yang bersamaan proses penghapusan karakter tertentu dapat dilakukan, seperti penghapusan tanda baca atau emoji (Amin, 2012).

3. *Stemming*

Setelah melalui proses pengubahan menjadi huruf kecil, maka perlu dilakukan pengecekan. Prosess *stemming* digunakan untuk menyeragamkan kata sehingga mengurangi daftar kata yang ada pada data *training* (Manning et al., 2008).

4. *Filtering*

Filtering atau dikenal sebagai eliminasi *stopwords* memiliki banyak kelebihan, yaitu akan mengurangi *space* pada tabel *term index* hingga 40% atau lebih. Proses *stopword removal* merupakan proses penghapusan *term* yang tidak memiliki arti atau tidak relevan (Gunawan et al., 2018).

3.5.3 Pembagian Data

Terdapat dua bagian data yang dibutuhkan yaitu *training data* dan *testing data*. *Training data* adalah data yang digunakan untuk melatih model agar mampu mengenali pola yang sedang dicari. Sedangkan *testing data* adalah data yang digunakan untuk menguji hasil pelatihan yang sudah dilakukan. Pada penelitian ini akan dibagi *training* data sebesar 80% dan *testing* data sebesar 20%, persentase pembagian tersebut sesuai dengan *rule of thumb* (aturan umum). Menurut Suthaharan (2015), meskipun penelitian eksklusif belum dilakukan dalam pemilihan rasio yang optimal antara kumpulan data ini, terdapat beberapa praktik umum dalam memilih ukuran kumpulan data ini salah satunya adalah *rule of thumb*. Kemudian parameter lainnya adalah *random_state* yang merupakan *random number generator (RNG)*. *Random_state* yang digunakan pada model adalah 77, *random_state* digunakan agar didapatkan hasil pembagian *train* dan *test* yang sama meski dilakukan berulang kali.

3.5.4 Pembobotan Kata

Pada proses ini dilakukan pengurutan dokumen menggunakan representasi *vector space model* dari kumpulan *dataset*. Dokumen dalam *vector space model* direpresentasikan dalam matriks yang berisi bobot kata. Bobot tersebut menyatakan kepentingan/kontribusi kata terhadap suatu dokumen dan sekumpulan dokumen. Kepentingan suatu kata dalam dokumen dilihat dari frekuensi kemunculannya terhadap dokumen. Hal umum terjadi adalah kata yang berbeda memiliki frekuensi yang berbeda. Berikut merupakan metode pembobotan kata yang digunakan:

1. *TFIDF*

Dalam menghitung nilai *TFIDF* pada penelitian ini akan menggunakan *library Scikit-Learn*. *Sckit Learn* sering disebut juga *SKLearn* adalah *library* dalam *machine learning* yang memiliki beragam *feature* yang mendukung algoritma klasifikasi, *regression*, dan *clustering*, diantaranya *support vector machines*, *random forest*, *gradient boosting*, *k-means*, dan *DBSCAN*. *SKLearn* telah didesain untuk dapat digunakan bersama *library Numpy* dan *Scipy*. Untuk dapat menghitung *TFIDF* modul yang digunakan adalah

TfidfVectorizer. Untuk setiap istilah dalam dokumen, matiks dihitung dengan melakukan 3 langkah berikut:

a. Menghitung frekuensi suatu istilah dalam dokumen, yang dikenal sebagai *Term Frequency*. Proses ini dicapai dengan membagi berapa kali suatu istilah muncul dalam suatu dokumen.
b. Menghitung *inverse* dari frekuensi dokumen suatu istilah. Proses ini dihitung dengan membagi jumlah total dokumen dengan jumlah dokumen yang mengandung istilah. Kebalikannya dihitung sehingga dapat menghitung nilai log positif. Oleh karena itu, hitung log dari nilai yang dihitung akan menghasilkan nilai positif yang dikenal dengan *Inverse Document Frequency (IDF)*
c. Langkah terakhir adalah dengan mengalikan hasil 1 dan 3 dan dikenal dengan *TFIDF*. Baris dari matriks mewakili istilah dan kolom dari matriks adalah nama dokumen.

2. *Bag of Words*

BoW merupakan teknik untuk membuat matriks kata-kata dimana kata-kata (istilah) direpresentasikan sebagai baris dan kolom yang mewakili nama dokumen. Kemudian mengisi matriks dengan frekuensi setiap istilah dalam dokumen, mengabaikan tata bahasa dan urutan persyaratan yang ada. Matriks tersebut dinamakan sebagai *Term Document Matrix (TDM)*. Dalam melakukan proses *vectorization* beberapa langkah umum yang disediakan algoritma *BoW* yaitu: (1) tokenisasi *string* dan memberikan *id integer* untuk setiap *token* yang muncul, dapat menggunakan spasi dan tanda baca sebagai pemisah *token*, (2) menghitung kemunculan token di setiap dokumen, (3) normalisasi dan pembobotan dengan *token* penting yang semakin berkurang yang terjadi pada sebagian besar sampel atau dokumen.Untuk mengkonversi teks menjadi angka dalam *BoW*, digunakan *library SKLearn* dengan modul *CountVectorizer*. Didalam algoritma *BoW* terdapat fitur dan sampel yang didefinisikan sebagai berikut:

a. Setiap frekuensi kejadian *token* individu (dinormalisasi atau tidak) akan diperlakukan sebagai fitur.
b. Vektor dari semua frekuensi *token* untuk dokumen yang diberikan akan dianggap sebagai sampel multivariat.

c. Korpus dokumen atau kamus dokumen akan berisi matriks dengan satu baris per dokumen dan satu kolom per *token* yang ada di korpus.

3. *Seleksi Fitur*

a. *TFIDF L1 Normalisasi*

Ketika menghitung *TF* pada fungsi *TFidfVectorizer()* pada *SKLearn* tidak dilakukan normalisasi, sehingga bentuk formula *TF* pada *SKLearn* dirumuskan sebagai berikut.

$$Tf_{ij} = f_d\,(i) \qquad \ldots.(3.1)$$

Sebagai contoh, bila terdapat dua buah kalimat dokumen sebagai berikut.

‘hari ini cerah setelah kemarin mendung’,
‘jakarta hari kemarin mendung’

Maka hasil *vector TF* pada *SKLearn* dan nama fitur adalah seperti pada gambar 3.1 dan 3.2.

```
[1, 1, 1, 0, 1, 1, 1],
[0, 1, 0, 1, 1, 1, 0]
```

Gambar 3. 1 Pembobotan *TFIDF*

```
['cerah', 'hari', 'ini', 'jakarta', 'kemarin', 'mendung',
'setelah']
```

Gambar 3. 2 Contoh Kata *TFIDF*

Untuk mengatasi masalah tersebut tidak cukup hanya menggunakan *TfidfVectorizer*, maka akan ditambahkan fungsi *CountVectorizer()* yang dapat menghasilkan *vector TF*, kemudian diterapkan *L1* normalisasi dengan fungsi *normalize()*.

b. *Mutual Information*

Pada fungsi ini akan memperkirakan informasi timbal balik untuk *variable* yang bertarget diskrit. Informasi timbal balik antara dua *variable* acak adalah nilai non-negatif, yang mengukur ketergantungan antara *variable*. Sama dengan nol jika dan hanya ada dua variabel acak independen, dan nilai yang lebih tinggi berarti ketergantungan lebih tinggi. Fungsi ini bergantung pada metode *nonparametric* yang didasarkan estimasi entropi dari jarak terdekat.

3.5.5 *Naïve Bayes*

Pada proses pengklasifikasian menggunakan algoritma *Naïve bayes*, menggunakan *library SKLearn* dengan modul *MultinomialNB*. Multinomial diterapkan karena sesuai dengan kasus data yang digunakan yaitu menghitung berapa kali setiap kata muncul dalam dokumen. Algoritma membantu model dengan memberikan informasi, tidak hanya seperti apakah suatu kata terdapat pada dokumen atau tidak. Algoritma ini mengasumsikan bahwa fitur diambil dari distribusi multinomial.

3.5.6 *Support Vector Machine (SVM)*

Pengklasifikasian algoritma *SVM* dalam *python* menggunakan *library SKLearn* dan modul *SVC*. Pada pengklasifikasian awal akan digunakan parameter C sebesar 1, dengan *kernel* yang digunakan adalah linear, dan dengan *gamma* yaitu *scale*.

3.5.7 *Hyperparameter Turning*

Pada tahap klasifikasi, langkah ini cukup krusial untuk dilakukan, karena kinerja suatu algoritma bisa sangat tergantung pada pilihan *hyperparameter* yang digunakan. Penyetelan parameter secara manual membutuhkan waktu dari langkah dalam *machine learning*, *feature engineering* dan hasil interpretasi. Penyetelan *hyperparameter* secara otomatis dilakukan dengan tujuan menemukan *hyperparameter* optimal dalam waktu yang lebih singkat, melalui pencarian informasi yang diperlukan.

3.5.8 Pengukuran Kualitas Hasil Uji

Pada tahap ini kinerja dari model yang telah dilakukan akan diuji dengan dua parameter, yaitu *accuracy*, *precision* dan *recall*. *Precision (P)* merupakan seberapa banyak hasil pemrosesan yang cocok terhadap informasi yang ingin dicari atau dengan kata lain *precision* adalah klasifikasi positif yang benar *(true positive)* dan keseluruhan data yang diprediksikan sebagai kelas positif. *Precision* dapat diperoleh dengan penggunaan persamaan 3.2 (Hadna et al., 2016).

$$P = \frac{tp}{tp+fp} \quad \text{....(3.2)}$$

Recall (R) adalah berapa banyak dokumen relevan dalam *collection* yang dihasilan oleh sistem, atau dapat berarti *recall* adalah jumlah dokumen yang memiliki klasifikasi positif yang benar *(true positive)* dari semua dokumen yang benar-benar positif (termasuk di dalamnya *false negative*). *Recall* dapat diperoleh dengan menggunakan persamaan 3.3 (Hadna et al., 2016).

$$R = \frac{tp}{tp+fn} \quad \text{....(3.3)}$$

Accuracy (A) merupakan jumlah dokumen yang diklasifikasikan dengan benar, baik itu *true positive* maupun *true negative*. Berikut merupakan persamaan yang digunakan untuk memperoleh nilai *accuracy*.

$$P = \frac{(tp+tn)}{(tp+fp+tn+fn)} \quad \text{....(3.4)}$$

Confusion matrix berisi variabel seperti *tn, tp, fn,* dan *fp*. *Tn* merupakan singkatan dari *true negative*, yaitu data negatif yang diklasifikasikan sebagai negatif. *Tp* adalah akronim dari *true positive*, data positif yang diklasifikasikan sebagai positif. *Fn* merupakan data positif yang diklasifikasikan sebagai negatif. *Fp* adalah data negatif yang diklasifikasikan sebagai positif. tabel 3.1 berisi visualisasi lebih jelas.

Tabel 3. 1 *Confusion Matrix*

	Mesin Katakan Ya	Mesin Katakan Tidak
Manusia Katakan Ya	tp	fn
Manusia Katakan Tidak	fp	tn

(Sumber: (Prabowo & Thelwall, 2009))

Selanjutnya dilakukan proses *cross validation*, proses ini akan dilakukan dengan menggunakan metode *k-fold cross validation*. *K-fold validation* akan membagi keseluruhan dokumen menjadi n bagian sama besar, kemudian proses *training* dan *testing* dilakukan sejumlah n kali, dan setiap kali pemrosesan menggunakan bagian yang berbeda sebagai *testing data* (Hadna, Santosa, & Winarno, 2016). Dalam penelitian ini akan digunakan *10-fold cross validation*. Metode ini dipilih karena *10-fold cross validation* merupakan salah satu *k-fold cross validation* yang direkomendasikan untuk pemilihan model terbaik dengan alasan condong memberikan estimasi akurasi yang kurang bias dibadingkan dengan *cross validation* biasa, *leave-one-out- cross validation* dan *bootstrap* (Wibowo, 2017).

3.5.9 Asosiasi Kata

Asosiasi kata akan dihitung menggunakan fungsi *collocations* pada *library nltk*. Dengan konsep yang sama *collocations* adalah frasa yang terdiri dari lebih dari satu kata tetapi kata-kata ini lebih sering muncul bersamaan dalam konteks yang diberikan daripada bagian kata individualnya. Misalnya, dalam satu *set* dokumen terkait rumah sakit, frasa *'CT scan'* lebih mungkin terjadi secara bersamaan daripada *'CT'* dan *'scan'* secara individual. *'CT scan'* juga merupakan ungkapan yang bermakna. Dua jenis kolokasi yang paling umum adalah *bigram* dan *trigram*. *Bigram* adalah dua kata yang berdekatan, seperti *'CT scan'*, *'machine learning'*, atau 'media sosial'. *Trigram* adalah tiga kata yang berdekatan, seperti 'keluar dari bisnis' (Ruchirawat, 2018). Pada *nltk*, *script* menerapkan *collocations* yaitu sebagai berikut, selengkapnya dapat dilihat pada lampiran.

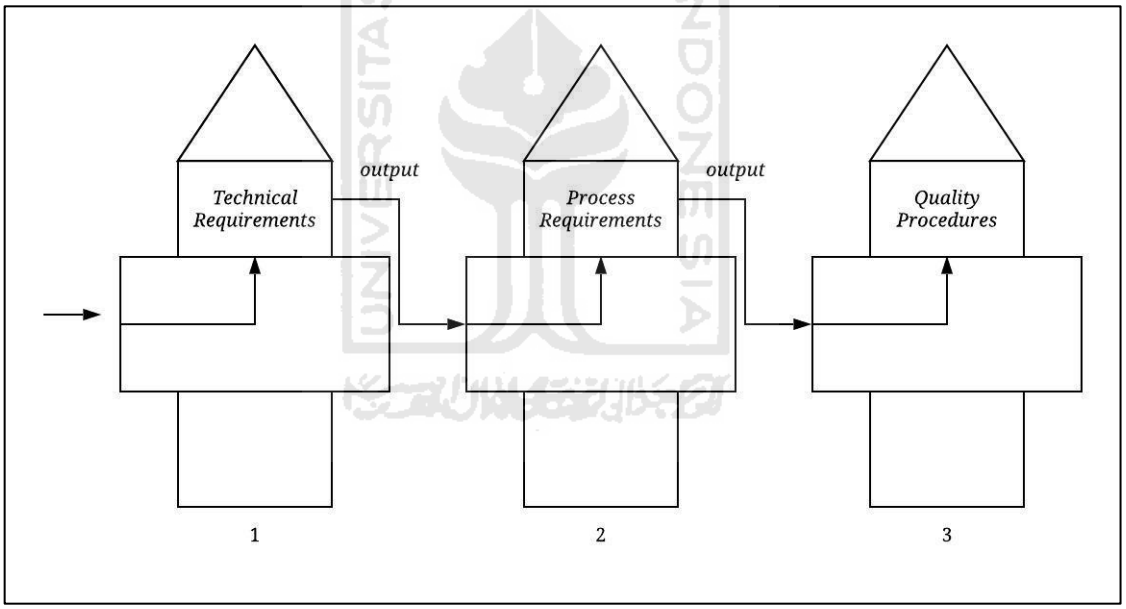
```
bigrams = nltk.collocations.BigramAssocMeasures()
trigrams = nltk.collocations.TrigramAssocMeasures()
```

3.5.10 *Quality Function Deployment (QFD)*

Berikut merupakan tahapan yang dilakukan dalam aplikasi *QFD* (Suhendar & Suroto, 2014):

1. Matriks *House of Quality (HOQ)*

Matriks *House of Quality* (*HOQ*) merupakan *output* yang dihasilkan dari metode *QFD*. Dalam *QFD*, suatu matriks yang saling berhubungan dikembangkan untuk menetapkan hubungan antara keinginan pelanggan dan parameter teknik dari produk atau jasa. Matriks ini menjelaskan apa saja yang menjadi kebutuhan dan harapan pelanggan dan bagaimana cara memenuhinya. gambar 3.3 di bawah ini merupakan Matriks *House of Quality*.

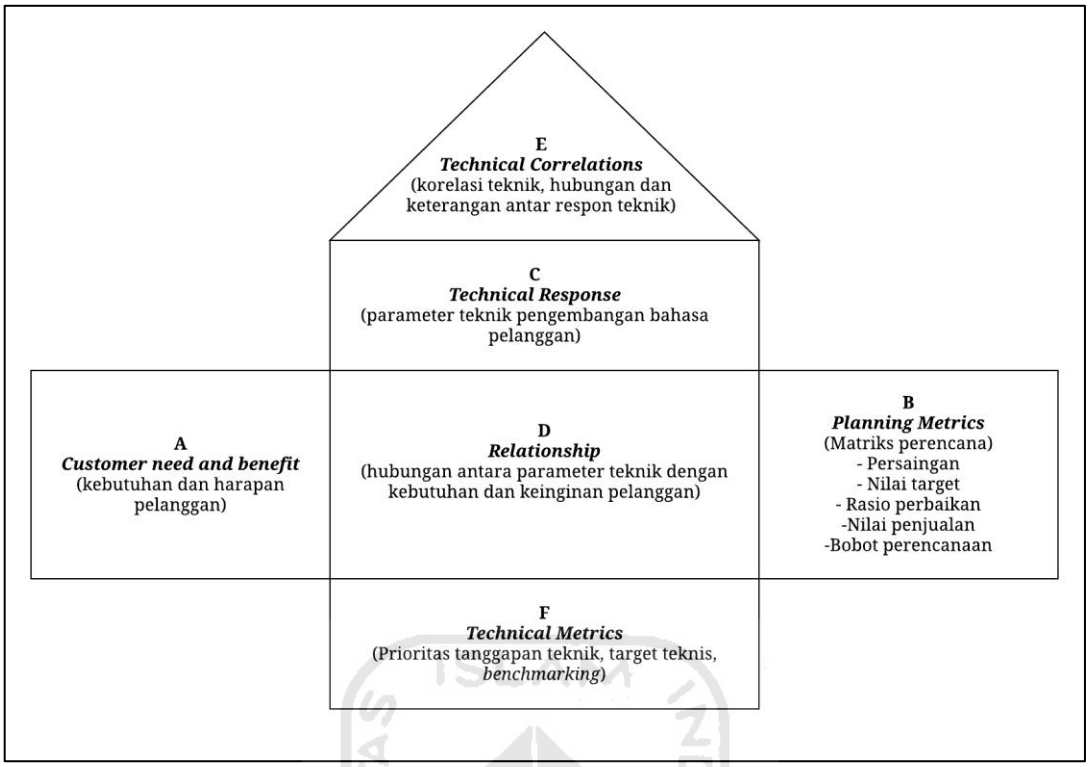


Gambar 3. 3 Matriks *HoQ*

Pada sisi kiri (bagian 1), terdiri dari daftar *input* yang berisi keinginan dari konsumen. *Input* diterjemahkan ke dalam *output* teknis, dimana akan dimasukkan ke sisi tengah (bagian 2). Kemudian, *output* pada matriks 2 menjadi *input* untuk matriks bagian 3.

2. Tahapan-tahapan Implementasi *Quality Function Deployment*

Di bawah ini merupakan data yang dibutuhkan dalam matriks *House of Quality*.



Gambar 3. 4 *House of Quality*

(Sumber: Cohen, 1995)

Terdapat tiga fase dalam pengimplementasian *Quality Function Deployment*, yaitu:

a. Fase Pertama (Mengumpulkan Suara Konsumen (*Voice of Customer*). Prosedur dalam mengumpulkan suara konsumen adalah:
   1) Menentukan atribut yang dianggap penting oleh konsumen (berupa data kualitatif) dan data ini diperoleh dari wawancara dan observasi terhadap konsumen.
   2) Mengukur tingkat kepentingan dari atribut-atribut.

b. Fase Kedua (Menyusun Rumah Kualitas/ *House of Quality*) . Langkah-langkah dalam pembuatan rumah kualitas meliputi:
   1) Pembuatan matriks keinginan konsumen.
   2) Pembuatan parameter teknik.
   3) Menentukan hubungan parameter teknik dengan keinginan konsumen.
   4) Korelasi teknis.
   5) *Benchmarking* dan penetapan target.

c. Fase Ketiga (Analisis dan Intepretasi)

Merupakan analisis dari tahapan-tahapan di atas. Selain ketiga tahapan di atas, ada tahapan yang pertama kali dilakukan yaitu tahapan perencanaan dan persiapan (fase 0/prafase). Adapun tahapan yang dilakukan, yaitu:

1) Menyiapkan dukungan organisasional meliputi dukungan dari pihak manajemen, dukungan fungsional dan dukungan teknis *QFD*.
2) Menentukan tujuan ataupun keuntungan yang diharapkan dari kegiatan *QFD*.
3) Menentukan siapa pelanggan. Karena dalam proses *QFD* penilaian banyak dilakukan oleh pelanggan.
4) Menentukan cakupan produk. Dalam hal ini harus ditentukan dahulu bagian mana dari produk atau jasa yang termasuk dan tidak termasuk dalam aktivitas *QFD*.
5) Melengkapi fasilitas dan material yang mendukung bagi pelaksanaan.

3.6 Analisis dan Pembahasan

Hasil analisis dan pembahasan dimaksudkan menjadi proses pengolahan data tahap akhir yang dikerjakan dengan berdasarkan data-data yang diambil dengan metode *scraping*. Hasil analisis dan pembahasan menyediakan informasi terkait perkembangan model *machine learning* yang dilakukan dengan menilai sentimen dan kriteria penting produk sepatu ventela, sesuai dengan *dataset* ulasan yang dimiliki. Selanjutnya kriteria penting akan dikembangkan sebagai *input* dalam metode pengembangan produk yaitu *QFD,* sehingga didapatkan spesifikasi teknis yang mewakili *voice of customer* tersebut.

3.7 Rekomendasi

Menurut hasil analisis dan pembahasan, dapat dilakukan rekomendasi berupa kekurangan dan kelebihan dari metode integrasi analisis data berbasis *machine learning* dengan *QFD* yang telah dilakukan.

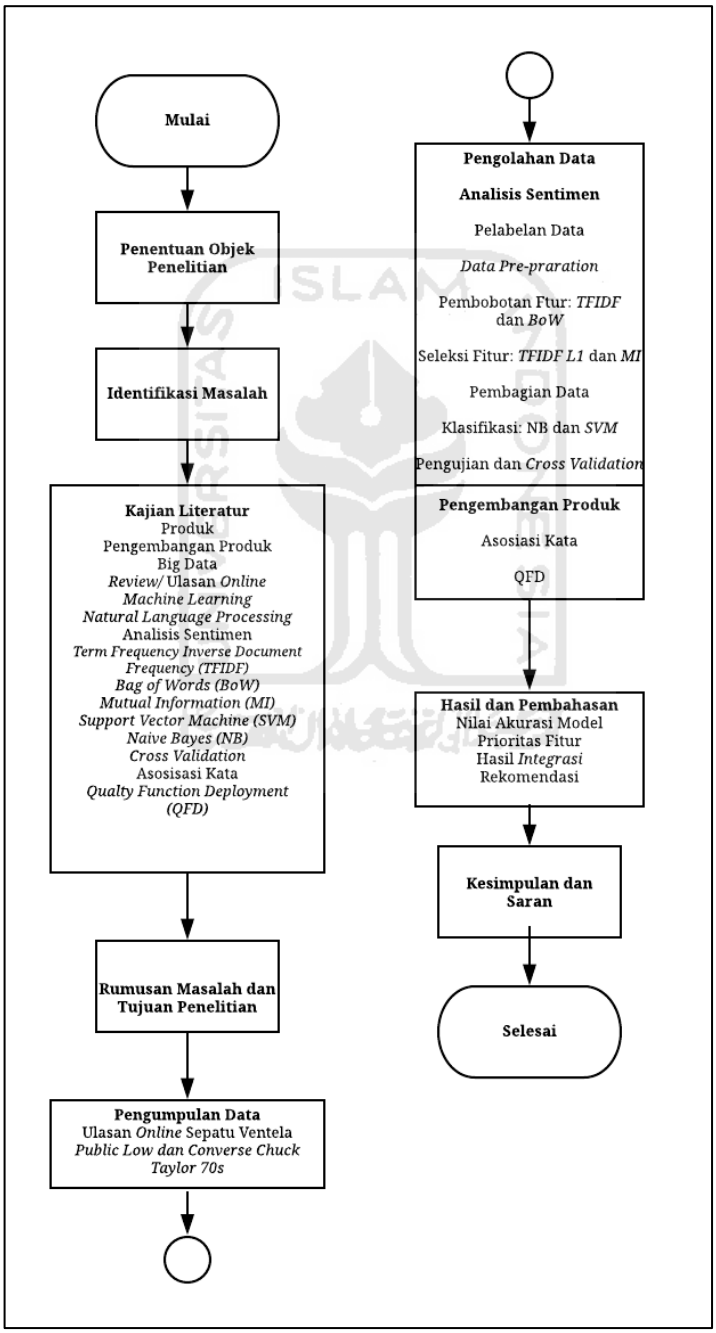
3.8 Kesimpulan dan Saran

Pada akhirnya tahap kesimpulan dan saran, akan menjadi jawaban atas rumusan masalah dan tujuan yang telah tertulis pada bab 1. Pada tahap ini guna meningkatkan proses

analisis data, akan direkomendasikan terkait metode *machine learning* dan pengembangan produk yang dapat diterapkan oleh peneliti selanjutnya serta kepada perusahaan.

3.9 *Flow Chart* Kerangka Penelitian

Gambar 3.5 berikut merupakan alur dalam kerangkan penelitian yang dilakukan.



Gambar 3. 5 Diagram Alur Penelitian

BAB IV

PENGUMPULAN DAN PENGOLAHAN DATA

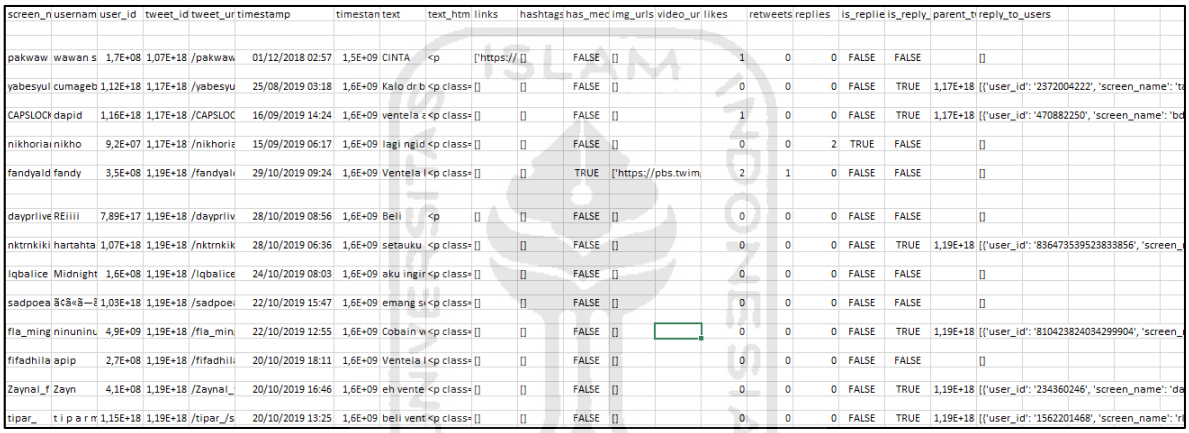
4.1 Pengumpulan Data

Proses *scraping* dilakukan pada media sosial Twitter dan Youtube dan *marketplace* Shopee, dengan *software* yang digunakan yaitu python dan dua *library* utama yaitu *pandas* dan *numpy*. *Pandas* adalah alat analisis dan manipulasi data yang termasuk kedalam *open source* yang cepat, kuat, fleksibel dan mudah digunakan, dan dibangun diatas bahasa pemrograman *python*. Salah satu fitur dari *pandas* adalah *dataframe*. Melalui fitur *dataframe* memungkinkan kita membaca sebuah *file* dan mengubahnya menjadi tabel, dapat juga melakukan manipulasi degan mengolah data dengan menggunakan operasi seperti *join, distinct, group by,* agregasi, dan teknik lain yang terdapat pada *SQL*. Format yang dapat dibaca oleh *pandas* ada banyak seperti .txt, .csv, .json, .tsv dan lainnya.

Numpy (numerical python) adalah *library python* yang fokus pada *scientific computing*. *Numpy* memiliki kemampuan untuk membentuk objek *N-dimentional array*, yang mirip dengan *list* pada *python*. Kelebihan *numpy array* dibandingkan dengan *list* pada *python* adalah konsumsi memori yang lebih kecil serta *runtime* yang lebih cepat. *Numpy* juga memudahkan pada aljabar linear terutama operasi pada *vector (1-d array)* dan *matrix (2-day array).*

4.1.1 Twitter *Scraping*

Pada pengambilan data di Twitter, *Twitterscraper* digunakan karena tidak menggunakan tampilan *web* dan tidak ada limitasi 1000 *tweet* dan data dapat disimpan pada *file* .json atau .csv. Kata kunci yang digunakan pada proses ini yaitu ventela, dengan data tang diambil mulai *tweet* pada tanggal 1 Maret 2018 sampai dengan tanggal 31 Maret 2020 yang disimpan dalam bentuk csv. Beberapa variabel yang didapatkan yaitu *screen_name*, *username*, *user_id*, *tweet_id*, *tweet_url*, *timestamp*, *text*, dan lain-lain, dengan data yang diperoleh sebanyak 996 *tweet*. Gambar 4.1 merupakan *output* yang diperoleh dari *twitter scraping*.



screen_n	usernam	user_id	tweet_id	tweet_ur	timestamp	timestan	text	text_htm	links	hashtags	has_mec	img_urls	video_ur	likes	retweets	replies	is_replie	is_reply_	parent_t	reply_to_users
pakwaw	wawan s	1,7E+08	1,07E+18	/pakwaw	01/12/2018 02:57	1,5E+09	CINTA	<p	['https://	[]	FALSE	[]		1	0	0	FALSE	FALSE		[]
yabesyul	cumageb	1,12E+18	1,17E+18	/yabesyu	25/08/2019 03:18	1,6E+09	Kalo dr b	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	TRUE	1,17E+18	[{'user_id': '2372004222', 'screen_name': 't
CAPSLOCK	dapid	1,16E+18	1,17E+18	/CAPSLOC	16/09/2019 14:24	1,6E+09	ventela z	<p class=	[]	[]	FALSE	[]		1	0	0	FALSE	TRUE	1,17E+18	[{'user_id': '470882250', 'screen_name': 'bd
nikhoria	nikho	9,2E+07	1,17E+18	/nikhoriz	15/09/2019 06:17	1,6E+09	lagi ngid	<p class=	[]	[]	FALSE	[]		0	0	2	TRUE	FALSE		[]
fandyald	fandy	3,5E+08	1,19E+18	/fandyal	29/10/2019 09:24	1,6E+09	Ventela l	<p class=	[]	[]	TRUE	['https://pbs.twim		2	1	0	FALSE	FALSE		[]
dayprlive	REiiii	7,89E+17	1,19E+18	/dayprliv	28/10/2019 08:56	1,6E+09	Beli	<p	[]	[]	FALSE	[]		0	0	0	FALSE	FALSE		[]
nktrnkiki	hartahta	1,07E+18	1,19E+18	/nktrnkik	28/10/2019 06:36	1,6E+09	setauku	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	TRUE	1,19E+18	[{'user_id': '836473539523833856', 'screen_
Iqbalice	Midnight	1,6E+08	1,19E+18	/Iqbalice	24/10/2019 08:03	1,6E+09	aku ingir	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	FALSE		[]
sadpoea	ãĉã«ã—ã	1,03E+18	1,19E+18	/sadpoe	22/10/2019 15:47	1,6E+09	emang si	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	FALSE		[]
fla_ming	ninuninu	4,9E+09	1,19E+18	/fla_min	22/10/2019 12:55	1,6E+09	Cobain v	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	TRUE	1,19E+18	[{'user_id': '810423824034299904', 'screen_
fifadhila	apip	2,7E+08	1,19E+18	/fifadhil	20/10/2019 18:11	1,6E+09	Ventela l	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	FALSE		[]
Zaynal_f	Zayn	4,1E+08	1,19E+18	/Zaynal_	20/10/2019 16:46	1,6E+09	eh vente	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	TRUE	1,19E+18	[{'user_id': '234360246', 'screen_name': 'da
tipar_	t i p a r m	1,15E+18	1,19E+18	/tipar_/s	20/10/2019 13:25	1,6E+09	beli vent	<p class=	[]	[]	FALSE	[]		0	0	0	FALSE	TRUE	1,19E+18	[{'user_id': '1562201468', 'screen_name': 'r

Gambar 4. 1 Hasil *Scraping* dari *Twitter*

4.1.2 Youtube *Scraping*

Proses pengambilan data melalui *Youtube* dilakukan dengan *Youtube Comment Scraper*. Pengambilan data dilakukan berdasar video yang ada, kata kunci yang digunakan dalam mencari video yang akan diambil komentarnya yaitu *ventela low public*. Data yang diperoleh yaitu sebanyak 9412 komentar. Gambar 4.2 berikut merupakan hasil keluaran dari proses ini.



Gambar 4. 2 Hasil *Scraping* dari *Youtube*

4.2 Pelabelan Data

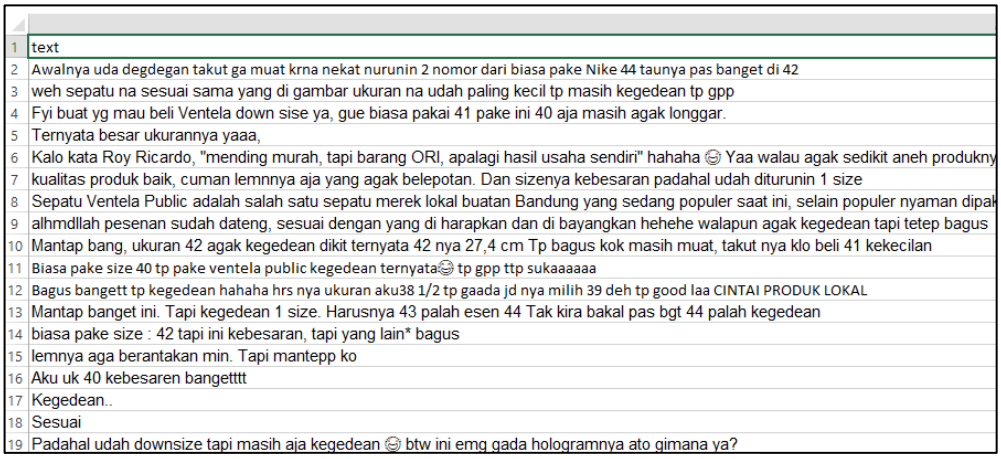
Analisis sentimen pada penelitian ini termasuk ke dalam *supervised learning*, yaitu algoritma pada model perlu diberikan *training dataset* berupa informasi *input* dan *output* yang diinginkan sehingga model akan mempelajari berdasarkan data yang telah ada, sistem akan mencari pola data dari *dataset* kemudian pola tersebut akan dijadikan sebagai acuan untuk data-data selanjutnya. Menurut Sebastiani (2002), pembuatan konstruksi kategorisasi teks secara otomatis memerlukan ilmu dari seorang ahli *domain*. Hal ini berguna untuk melakukan perbandingan analisa sistem dengan analisa ahli bahasa. Namun dalam penelitian ini proses klasifikasi yang dilakukan dengan pelabelan secara manual dan dilakukan lebih dari satu kali oleh orang dengan latar belakang pendidikan yang berbeda. Tabel 4.1 berisikan data *user* yang terlibat dalam proses pelabelan.

Tabel 4. 1 Deskripsi *User*

User	Usia (tahun)	Jenis Kelamin	Latar Belakang Pendidikan
User 1	22	Perempuan	Ilmu Komunikasi
User 2	18	Laki-laki	Teknik Mesin
User 3	50	Perempuan	Umum
User 4	22	Perempuan	Dokter Gigi
User 5	22	Perempuan	Ilmu Hukum

Proses pelabelan ini menghasilkan data terbagi ke dalam tiga kelas kata, yaitu positif, netral, dan negatif dengan porsi yang sama. Setiap dokumen yang diklasifikasi akan dikategorikan ke dalam kelas yang paling dominan. Sebelumnya, data telah melalui proses *filter*, dengan mengambil kolom *'text'* pada data yang telah diberi label. Kemudian

menggabungkan dua *file*, yaitu *file* yang berisi data *review twitter* dan *shopee* dengan *file* yang berisi data *review youtube*. Gambar 4.3 merupakan hasil penggabungan *filter* data.



	text
2	Awalnya uda degdegan takut ga muat krna nekat nurunin 2 nomor dari biasa pake Nike 44 taunya pas banget di 42
3	weh sepatu na sesuai sama yang di gambar ukuran na udah paling kecil tp masih kegedean tp gpp
4	Fyi buat yg mau beli Ventela down sise ya, gue biasa pakai 41 pake ini 40 aja masih agak longgar.
5	Ternyata besar ukurannya yaaa,
6	Kalo kata Roy Ricardo, "mending murah, tapi barang ORI, apalagi hasil usaha sendiri" hahaha 😊 Yaa walau agak sedikit aneh produkny
7	kualitas produk baik, cuman lemnnya aja yang agak belepotan. Dan sizenya kebesaran padahal udah diturunin 1 size
8	Sepatu Ventela Public adalah salah satu sepatu merek lokal buatan Bandung yang sedang populer saat ini, selain populer nyaman dipak
9	alhmdllah pesenan sudah dateng, sesuai dengan yang di harapkan dan di bayangkan hehehe walapun agak kegedean tapi tetep bagus
10	Mantap bang, ukuran 42 agak kegedean dikit ternyata 42 nya 27,4 cm Tp bagus kok masih muat, takut nya klo beli 41 kekecilan
11	Biasa pake size 40 tp pake ventela public kegedean ternyata😊 tp gpp ttp sukaaaaaa
12	Bagus bangett tp kegedean hahaha hrs nya ukuran aku38 1/2 tp gaada jd nya milih 39 deh tp good laa CINTAI PRODUK LOKAL
13	Mantap banget ini. Tapi kegedean 1 size. Harusnya 43 palah esen 44 Tak kira bakal pas bgt 44 palah kegedean
14	biasa pake size : 42 tapi ini kebesaran, tapi yang lain* bagus
15	lemnya aga berantakan min. Tapi mantepp ko
16	Aku uk 40 kebesaren bangetttt
17	Kegedean..
18	Sesuai
19	Padahal udah downsize tapi masih aja kegedean 😊 btw ini emg gada hologramnya ato gimana ya?

Gambar 4. 3 Data yang Telah Digabungkan

Kategori kelas akhir dari masing-masing data *review* ditentukan dengan mengambil hasil klasifikasi yang paling banyak dipilih oleh kelima *user*. Dokumen yang berisi dua variabel (*text* dan sentimen) kemudian dikonversikan menjadi tipe json. Dalam penelitian ini, terdapat beberapa percobaan untuk mendapatkan hasil yang optimal salah satunya adalah jumlah data yang berbeda. Dokumen A berisi total 1062 data masing-masing kelas sentimen berisi 354 *text*. Dokumen B berisi total 1176 data masing-masing kelas sentimen berisi 392 *text*. Dan dokumen C berisi total 1326 data dengan masing-masing kelas sentimen berisi 442 *text*. Tabel 4.2 menunjukkan salah satu contoh klasifikasi manual yang dilakukan oleh masing-masing *user*.

Tabel 4. 2 Contoh Pengambilan Keputusan Proses Pelabelan

No	*Text*	*User1*	*User2*	*User3*	*User4*	*User5*	Hasil
35	gw iseng hajar main basket 2 minggu berturut turut, kuat aja sol nya, compass kalah masalah kenyamanan *and build quality*	negatif	positif	negatif	positif	positif	positif
416	Tapi ini ko lemnya keliatan yaaa? *Reject* atau gmn ya gan :(((	negatif	negatif	negatif	negatif	netral	negatif

No	*Text*	*User1*	*User2*	*User3*	*User4*	*User5*	Hasil
496	Bangga si sama produk lokal tpi plagiat jdi rada gmna gtu	netral	negatif	netral	netral	netral	netral

4.3 Proses Pengolahan *Dataset*

Pada proses pengolahan *dataset*, terdapat beberapa *library* yang digunakan seperti *Regrex, Nltk, SKLearn,* dan *matplotlib*. *Regrex (regular expression)* merupakan urutan karakter khusus yang membantu mencocokkan atau menemukan *string* atau *set* lainnya yang menggunakan sintaksis khusus yang disimpan dalam suatu pola. Modul *re* akan muncul *exception re.error* apabila terjadi kesalahan saat menggunakan modul tersebut. *Library Nltk* merupakan *platform* yang dikenal untuk membangun program dalam *python* untuk dapat bekerja dengan data bahasa manusia. *Nltk* menyediakan lebih dari 50 korpora dan sumber daya leksikal yang mudah digunakan seperti *WordNet*, dan serangkaian pustaka pemrosesan teks untuk klasifikasi, *tokenization, stemming, tagging, parsing,* dan penalaran semantik. Pada bab ini akan ditampilkan hasil dengan menggunakan dokumen B, karena dokumen B merupakan *dataset* yang menghasilkan performa paling baik dari serangkaian percobaan yang dilakukan. Untuk hasil pada dokumen A dan B dapat dilihat pada lampiran.

4.3.1 *Preprocessing*

Data yang digunakan untuk klasifikasi ini adalah data pembelajaran *(data training)* dan data validasi *(data testing)*. Sebelum dilakukan proses pembelajaran, dilakukan *preproccesing data*. Tujuan dilakukannya *preprocessing* ini untuk menyeragamkan kata dan mengurangi *volume* kata yang tidak perlu. Pada proses ini menggunakan *library regex* pada *python*. Hasil dari proses *pre-processing* (dengan proses normalisasi) dapat dilihat pada gambar 4.4

	text	sentimen	text_clean	text_clean1	text_clean_stopword	text_normalized	text_clean_stemmed
0	Asliii sih parah parahh, bagus bgt ternyataaa?...	positif	asliii sih parah parahh bagus bgt ternyataaa l...	[asliii, sih, parah, parahh, bagus, bgt, terny...	[asliii, parah, parahh, bagus, bgt, kali, mo, ...	[asli, parah, parah, bagus, sangat, sekali, ma...	[asli, parah, parah, bagus, sangat, sekali, ma...
1	Masyaa Allah, inj sepwtu lokal tapi kualitasny...	positif	masyaa allah inj sepwtu lokal tapi kualitasnya...	[masyaa, allah, inj, sepwtu, lokal, tapi, kual...	[masyaa, lokal, kualitasnya, bersaing, produk,...	[mantap, lokal, kualitasnya, bersaing, produk,...	[mantap, lokal, kualitas, saing, produk, luar,...
2	Ventela tetap terbaik	positif	ventela tetap terbaik	[ventela, tetap, terbaik]	[tetap, terbaik]	[tetap, terbaik]	[tetap, baik]
3	Local brand tp bagus. Dipakenya juga enak. Bah...	positif	local brand tp bagus dipakenya juga enak bahan...	[local, brand, tp, bagus, dipakenya, juga, ena...	[local, brand, tp, bagus, dipakenya, enak, bah...	[lokal, merek, tapi, bagus, dipakainya, enak, ...	[lokal, merek, tapi, bagus, pakai, enak, bahan...
4	Kualitas bagus. Solnya empuk. Saya pikir dapat...	positif	kualitas bagus solnya empuk saya pikir dapat b...	[kualitas, bagus, solnya, empuk, saya, pikir, ...	[kualitas, bagus, solnya, empuk, pikir, bonus,...	[kualitas, bagus, solnya, empuk, pikir, bonus,...	[kualitas, bagus, sol, empuk, pikir, bonus, ta...
...	...	...	...	...	...	...	...
1171	Kenapa yaa saya mikirnya ventela ni sengaja ba...	netral	kenapa yaa saya mikirnya ventela ni sengaja ba...	[kenapa, yaa, saya, mikirnya, ventela, ni, sen...	[mikirnya, sengaja, balik, bikin, siluet, miri...	[memikirkan, sengaja, balik, buat, siluet, mir...	[pikir, sengaja, balik, buat, siluet, mirip, v...
1172	Bodo amat gw mah, beli pake duit gw sendiri. G...	netral	bodo amat gw mah beli pake duit gw sendiri ga ...	[bodo, amat, gw, mah, beli, pake, duit, gw, se...	[bodo, beli, pake, duit, sendiri, ga, ngerugii...	[bodoh, beli, pakai, duit, sendiri, tidak, rug...	[bodoh, beli, pakai, duit, sendiri, tidak, rug...
1173	Baru kemaren beli ventela gegara pingin tau	netral	baru kemaren beli ventela gegara pingin tau	[baru, kemaren, beli, ventela, gegara, pingin,...	[baru, kemaren, beli, gegara, pingin, tau]	[baru, kemarin, beli, karena, ingin, tahu]	[baru, kemarin, beli, karena, ingin, tahu]
1174	nikmati aja,soalnya duit aku pas pasan wkwkkwkw	netral	nikmati aja soalnya duit aku pas pasan wkwkkwkw	[nikmati, aja, soalnya, duit, aku, pas, pasan,...	[nikmati, soalnya, duit, pas, pasan]	[nikmati, karena, duit, pas, pasan]	[nikmat, karena, duit, pas, pas]
1175	semakin lo hujat mirip vans/converse semakin v...	negatif	semakin lo hujat mirip vans converse semakin v...	[semakin, lo, hujat, mirip, vans, converse, se...	[semakin, hujat, mirip, vans, converse, semaki...	[semakin, hujat, mirip, vans, converse, semaki...	[makin, hujat, mirip, vans, converse, makin, v...

Gambar 4. 4 Hasil Proses *Pre-Processing*

Berikut merupakan tahapan yang dilakukan dalam *pre-processing*:

1. *Case Folding*

Salah satu bentuk keberagaman dalam data adalah penulisan huruf besar dan huruf kecil. Untuk mengatasi hal ini teks akan diubah dalam huruf kecil melalui fungsi *lower()*. Hasil data setelah melewati proses ini dapat dilihat pada gambar 4.5 berikut.

	text	sentimen	text_clean
0	Asliii sih parah parahh, bagus bgt ternyataaa?...	positif	asliii sih parah parahh bagus bgt ternyataaa l...
1	Masyaa Allah, inj sepwtu lokal tapi kualitasny...	positif	masyaa allah inj sepwtu lokal tapi kualitasnya...
2	Ventela tetap terbaik	positif	ventela tetap terbaik
3	Local brand tp bagus. Dipakenya juga enak. Bah...	positif	local brand tp bagus dipakenya juga enak bahan...
4	Kualitas bagus. Solnya empuk. Saya pikir dapat...	positif	kualitas bagus solnya empuk saya pikir dapat b...
...	...	...	...
1171	Kenapa yaa saya mikirnya ventela ni sengaja ba...	netral	kenapa yaa saya mikirnya ventela ni sengaja ba...
1172	Bodo amat gw mah, beli pake duit gw sendiri. G...	netral	bodo amat gw mah beli pake duit gw sendiri ga ...
1173	Baru kemaren beli ventela gegara pingin tau	netral	baru kemaren beli ventela gegara pingin tau
1174	nikmati aja,soalnya duit aku pas pasan wkwkkwkw	netral	nikmati aja soalnya duit aku pas pasan wkwkkwkw
1175	semakin lo hujat mirip vans/converse semakin v...	negatif	semakin lo hujat mirip vans converse semakin v...

Gambar 4. 5 Hasil *Case Folding* Dokumen B

2. *Document Filtering*

Proses *document filtering* bertujuan untuk membersihkan data dari kata-kata atau simbol yang tidak diperlukan untuk mengurangi *noise* saat proses klasifikasi. Adapun kata-kata yang dihilangkan antara lain:

a. Tanda baca
b. Alamat situs *(url)*
c. Angka
d. *Tab, new line, as back slice*
e. *Non ASCII (emoticon, chinese word, etc)*
f. *Hashtag*

	text	sentimen	text_clean
0	Asliii sih parah parahh, bagus bgt ternyataaa?...	positif	asliii sih parah parahh bagus bgt ternyataaa l...
1	Masyaa Allah, inj sepwtu lokal tapi kualitasny...	positif	masyaa allah inj sepwtu lokal tapi kualitasnya...
2	Ventela tetap terbaik	positif	ventela tetap terbaik
3	Local brand tp bagus. Dipakenya juga enak. Bah...	positif	local brand tp bagus dipakenya juga enak bahan...
4	Kualitas bagus. Solnya empuk. Saya pikir dapat...	positif	kualitas bagus solnya empuk saya pikir dapat b...
...	...	...	...
1171	Kenapa yaa saya mikirnya ventela ni sengaja ba...	netral	kenapa yaa saya mikirnya ventela ni sengaja ba...
1172	Bodo amat gw mah, beli pake duit gw sendiri. G...	netral	bodo amat gw mah beli pake duit gw sendiri ga ...
1173	Baru kemaren beli ventela gegara pingin tau	netral	baru kemaren beli ventela gegara pingin tau
1174	nikmati aja,soalnya duit aku pas pasan wkwkkwkw	netral	nikmati aja soalnya duit aku pas pasan wkwkkwkw
1175	semakin lo hujat mirip vans/converse semakin v...	negatif	semakin lo hujat mirip vans converse semakin v...

Gambar 4. 6 Hasil Menghilangkan Tanda Baca Dokumen B

	text	sentimen	text_clean
0	Asliii sih parah parahh, bagus bgt ternyataaa?...	positif	asliii sih parah parahh bagus bgt ternyataaa l...
1	Masyaa Allah, inj sepwtu lokal tapi kualitasny...	positif	masyaa allah inj sepwtu lokal tapi kualitasnya...
2	Ventela tetap terbaik	positif	ventela tetap terbaik
3	Local brand tp bagus. Dipakenya juga enak. Bah...	positif	local brand tp bagus dipakenya juga enak bahan...
4	Kualitas bagus. Solnya empuk. Saya pikir dapat...	positif	kualitas bagus solnya empuk saya pikir dapat b...
...	...	...	...
1171	Kenapa yaa saya mikirnya ventela ni sengaja ba...	netral	kenapa yaa saya mikirnya ventela ni sengaja ba...
1172	Bodo amat gw mah, beli pake duit gw sendiri. G...	netral	bodo amat gw mah beli pake duit gw sendiri ga ...
1173	Baru kemaren beli ventela gegara pingin tau	netral	baru kemaren beli ventela gegara pingin tau
1174	nikmati aja,soalnya duit aku pas pasan wkwkkwkw	netral	nikmati aja soalnya duit aku pas pasan wkwkkwkw
1175	semakin lo hujat mirip vans/converse semakin v...	negatif	semakin lo hujat mirip vans converse semakin v...

Gambar 4. 7 Hasil Menghilangkan *Punctuation* Dokumen B

3. *Tokenization*

Setiap entitas adalah bagian dari pembagian berdasar aturan yang dibuat. Misalnya, ketika kalimat "dipisah" menjadi kata-kata, setiap kata adalah sebuah tag, jika membagi paragraf menjadi kalimat, setiap kalimat bisa menjadi penanda. *Tokenization* merupakan proses dimana teks yang berisi banyak kuantitas akan dibagi atau dipisah menjadi bagian yang lebih kecil. Proses ini menjadi penting untuk memahami pola dalam teks untuk dapat mencapai tujuan yang akan dicapai. *Token* akan sangat berguna untuk menemukan pola sesuai tujuan dan dianggap sebagai langkah dasar untuk *stemming* dan *lemmatization*. Digunakan *word_tokenize()* untuk membagi kalimat menjadi kata-kata. Hasil dari *tokenization* kata dapat dikonversi ke dalam *DataFrame* untuk pemahaman teks yang lebih baik dalam penggunaan *machine learning*. Dalam *machine learning* akan membutuhkan data numerik untuk dilatih dan membuat prediksi. *Token* data menjadi penting dari teks (*string*) untuk konversi data numerik. Kekurangan dari *nltk* adalah belum tersedia dalam bahasa indonesia. Sehingga beberapa kata majemuk (dua atau lebih kata yang bermakna tunggal) akan terbaca menjadi kata tunggal. Modul ini akan memecah kata dengan tanda baca, sebagai contoh hasil *tokenization* dari tahap sebelumnya dapat dilihat di gambar 4.8 berikut

```
Tokenizing Result :

0    [asliii, sih, parah, parahh, bagus, bgt, terny...
1    [masyaa, allah, inj, sepwtu, lokal, tapi, kual...
2                          [ventela, tetap, terbaik]
3    [local, brand, tp, bagus, dipakenya, juga, ena...
4    [kualitas, bagus, solnya, empuk, saya, pikir, ...
Name: text_clean1, dtype: object
```

Gambar 4. 8 Hasil *Tokenization* Dokumen B

4. Eliminasi *Stopwords*

Salah satu bentuk utama dari *preprocessing* adalah menyaring data yang tidak berguna. Dalam *NLP*, kata-kata yang tidak berguna disebut *stopwords*. Apabila tidak ingin kata-kata yang tidak berguna tersebut memenuhi ruang dalam basis data atau menghabiskkan waktu pemrosesan. Tidak ada aturan pasti dalam menentukan *stopwords* yang akan digunakan, penentuan *stopword*s ini disesuaikan dengan kasus yang sedang dihadapi.

Proses ini dapat dilakukan dengan menyimpan daftar kata yang menurut peneliti adalah *stopwords*. Pada saat ini akan memperlakukan *stopwords* sebagai kata yang tidak mengandung makna apa pun, dan akan dihapus dalam data. Dalam kelas *StopWordRemoverFactory* pada *library* Sastrawi telah terdapat beberapa kata *stopwords* isi dari kamus data tersebut selengkapnya terdapat di lampiran. Kemudian akan ditambahkan beberapa kata yang diperlukan. Berikut merupakan contoh hasil eliminasi *stopwords* dari tahap sebelumnya dapat dilihat di gambar 4.9.

```
0       [asliii, parah, parahh, bagus, bgt, kali, mo, ...
1       [masyaa, lokal, kualitasnya, bersaing, produk,...
2                                        [tetap, terbaik]
3       [local, brand, tp, bagus, dipakenya, enak, bah...
4       [kualitas, bagus, solnya, empuk, pikir, bonus,...
                              ...
1171    [mikirnya, sengaja, balik, bikin, siluet, miri...
1172    [bodo, beli, pake, duit, sendiri, ga, ngerugii...
1173           [baru, kemaren, beli, gegara, pingin, tau]
1174                 [nikmati, soalnya, duit, pas, pasan]
1175    [semakin, hujat, mirip, vans, converse, semaki...
Name: text_clean_stopword, Length: 1176, dtype: object
```

Gambar 4. 9 Hasil Menghilangkan *Stopword* Dokumen B

5. Normalisasi

Proses normalisasi merupakan proses penggantian kata tidak baku atau tidak sesuai menjadi kata baku dan telah sesuai. Pada proses ini bahasa asing juga diubah menjadi Bahasa Indonesia. Pada proses ini harus menyiapkan terlebih dahulu *dataset* untuk *mapping term* akan diseragamkan. Berikut merupakan contoh hasil eliminasi *stopwords* dari tahap sebelumnya dapat dilihat di gambar 4.10.

```
0    [asli, parah, parah, bagus, sangat, sekali, ma...
1    [mantap, lokal, kualitasnya, bersaing, produk,...
2                                     [tetap, terbaik]
3    [lokal, merek, tapi, bagus, dipakainya, enak, ...
4    [kualitas, bagus, solnya, empuk, pikir, bonus,...
5    [lama, tahu, produknya, baru, sempat, beli, te...
6    [bagus, sekali, perempuan, mau, beli, saran, u...
7    [nyaman, dipakainya, empuk, harga, merakyat, k...
8                   [harga, kualitas, semuanya, baik]
9    [suka, sangat, parah, saran, buat, teman, tema...
Name: text_normalized, dtype: object
```

Gambar 4. 10 Hasil Normalisasi Dokumen B

6. *Stemming*

Mentransformasikan kata menjadi kata dasar (*root*) dengan menghilangkan semua imbuhan kata (*affixes*) meliputi awalan kata (*prefixes*), sisipan kata (*infixes*), akhiran kata (*suffixes*) dan atau mengilangkan awalan dan akhiran kata (*confixes*) pada kata turunan. Konsep *stemming* hadir dengan alasan mengekstrak kata dasar dapat mempersingkat waktu pencarian karena banyak variasi kata yang memiliki arti yang sama dan menormalkan kalimat. Proses *stemming* pada penelitian ini menggunakan *library Sastrawi* yang dibantu dengan *swifter* agar proses *stemming* dapat lebih cepat.

```
0       [asli, parah, parah, bagus, sangat, sekali, ma...
1       [mantap, lokal, kualitas, saing, produk, luar,...
2                                          [tetap, baik]
3       [lokal, merek, tapi, bagus, pakai, enak, bahan...
4       [kualitas, bagus, sol, empuk, pikir, bonus, ta...
                              ...
1171    [pikir, sengaja, balik, buat, siluet, mirip, v...
1172    [bodoh, beli, pakai, duit, sendiri, tidak, rug...
1173          [baru, kemarin, beli, karena, ingin, tahu]
1174                [nikmat, karena, duit, pas, pas]
1175    [makin, hujat, mirip, vans, converse, makin, v...
Name: text_clean_stemmed, Length: 1176, dtype: object
```

Gambar 4. 11 Hasil *Stemming* Dokumen B

4.3.2 Pembobotan Kata

Algoritma pembobotan kata didasarkan bahwa teks perlu dikonversi menjadi angka sebelum dapat diterapkan ke dalam algoritma matematika dikenal sebagai *vectorization*. Ketika mengonversi teks menjadi angka, terdapat beberapa teknik yang dapat diterapkan. Salah satu teknik adalah menghitung kemunculan kata dalam sebuah dokumen. Dalam penelitian ini akan digunakan dua metode pembobotan kata yang sering digunakan yaitu *Bag of Words (BoW)* dan *TFIDF*.

1. *Bag of Words*

Modul *CountVectorizer* pada *SKLearn* memiliki beberapa parameter yaitu:

a. *Input* : *string {filename, file, content}*, *input* diharapkan sesuai urutan dari *item* dan memiliki tipe *string* atau *byte*. Dalam penelitian ini digunakan data pada kolom *text_clean_stemmed* sebagai *input*.

b. *Encoding*, digunakan secara *default* yaitu menggunakan *utf-8*.
c. *Decode_error* : perintah ini menginstruksikan apa yang harus diberikan apabila data mengandung karakter yang tidak sesuai dengan *encoding*. Pada penelitian ini digunakan secara *default* yaitu *strict* yang berarti *UnicodeDecodeError* akan muncul ketika terdapat karakter yang tidak sesuai.
d. *Strip_accents* : perintah untuk menghapus aksen *(asci, unicode)* dan memunculkan karakter lain yang telah dinormalisasi selama proses *preprocessing*. Pada penelitian disetel secara *default* yang berarti tidak melakukan perintah ini.
e. *Lowercase* : disetel secara *default* yaitu *true* sehingga perintah ini dilakukan.
f. *Preprocessor* : disetel secara *default* sehingga tidak dilakukan perintah ini.
g. *Tokenizer* : disetel secara *default* sehingga tidak dilakukan perintah ini.
h. *Stop_words* : disetel secara *default* sehingga tidak dilakukan perintah ini.
i. Token_pattern : Ekspresi reguler yang menunjukkan apa yang merupakan "token", hanya digunakan jika analyzer == 'word'. Token pilih regexp default dari 2 atau lebih karakter alfanumerik (tanda baca diabaikan sepenuhnya dan selalu diperlakukan sebagai pemisah token).
j. *Ngram_range* : disetel secara *default* maka akan ditampilkan sebagai (1,1) atau *unigram*.
k. *Analyzer* : perintah ini menginstruksikan untuk membuat fitur menggunakan *word* atau *character n-grams*. Disetel secara *default* maka ditampilkan sebagai *word*.
l. *Max_df* : perintah ini menginstruksikan ketika membangun kamus untuk mengabaikan *terms* yang memiliki *document frequency* lebih tinggi daripada *threshold*. Pada penelitian disetel secara *default* yaitu 1.0, *float* parameter merepresentasikan proporsi dari dokumen.
m. *Min_df* : perintah ini menginstruksikan ketika membangun kamus untuk mengabaikan *terms* yang memiliki *document frequency* lebih rendah daripada *threshold*. Pada penelitian disetel secara *default* yaitu 1, integer parameter merupakan perhitungan absolut.
n. *Max_features* : perintah ini menginstruksikan untuk membangun kamus hanya berdasarkan *max_features* yang tertinggi sesuai dengan *term frequency* yang ada pada *corpus*. Disetel secara *default* maka tidak melakukan perintah ini.
o. Vocabulary:Merupakan kunci adalah istilah dan nilai-nilai adalah indeks dalam matriks fitur, atau *iterable* atas istilah. Secara *default* diatur secara *none* yang berarti

tidak diberikan, kosakata ditentukan dari dokumen *input*. Indeks dalam pemetaan tidak boleh diulang dan tidak boleh memiliki kesenjangan antara 0 dan indeks terbesar.

p. *Binary*: perintah ini menginstruksikan untuk mengubah *all non zero counts* menjadi 1, dan akan berguna untuk probabilitas diskrit. Disetel secara *default* maka tidak dilakukan proses ini.

q. *Dtype* : menunjukkan tipe matriks yang akan dikembalikan oleh *fit_transform()* atau *transform()* secara *default* akan disetel *np.int64*.

Hasil pembobotan kata secara *bag of words*, dapat dilihat pada tabel 4.3 berikut.

Tabel 4. 3 Hasil *BoW* Dokumen B

BoW tanpa normalisasi			*BoW* dengan normalisasi		
asliii	0,21	1	asli	0,7	1
parah	0,124	1	parah	0,61	1
parahh	0,167	1	bagus	0,77	1
bagus	0,254	1	sangat	0,82	1
ternyataaa	0,609	1	sekali	0,118	1
kali	0,81	1	mau	0,133	1
mo	0,827	1	beli	0,444	1
beli	0,1162	1	sini	0,493	1
sini	0,1167	1	model	0,508	1
model	0,1176	1	agak	0,731	1
aga	0,1629	1	besar	0,759	1
gede	0,1631	1	sedikit	0,837	1
dikit	0,1991	1	baik	0,85	2
gapapa	0,1992	1	pakai	0,851	1
kaos	0,1996	1	kaos	0,1016	1
kaki	0,2384	1	kaki	0,1033	1
tebel	0,2549	1	tebal	0,1041	1
pas	0,2586	1	jadi	0,1094	1
			pas	0,1175	1

2. *TFIDF*

Pada *tfidfvectorizer* yang dilakukan parameter a-q adalah sama dengan yang dilakukan dengan *countvectorizer*. Berikut merupakan tambahan parameter yang dilakukan pada *tfidfvectorizer*:

a. *Dtype*: disetel secara *default* yaitu menampilakan *float64*.
b. *Norm*: Setiap baris *output* akan memiliki norma satuan, baik: * 'l2': Jumlah kuadrat elemen vektor adalah 1. Kesamaan cosinus antara dua vektor adalah produk titik mereka ketika norma l2 telah diterapkan. * 'L1': Jumlah nilai absolut elemen vektor adalah 1.
c. *Use_idf* : mengaktifkan *inverse-document-frequency* untuk pembobotan. Disetel secara *default* yaitu *true*, maka proses ini dilakukan.
d. *Smooth_idf*: bobot idf dengan menambahkan satu ke frekuensi dokumen, seolah-olah dokumen tambahan terlihat berisi setiap istilah dalam koleksi tepat sekali. Mencegah pembagian nol.
e. *Sublinear_tf* : menerapkan *sublinear tf scaling*, dengan merubah formula tf menjadi 1 + log (tf). Secara *default* yaitu *false* maka tidak dilakukan perintah ini.

Hasil pembobotan kata secara *TFIDF*, dapat dilihat pada gambar 4.12 dan 4.13 berikut.

term	TF	TF-IDF
asliii	0.05555555555555555	0.35426260821659034
parah	0.05555555555555555	0.20966873680301348
parahh	0.05555555555555555	0.2932285921794731
bagus	0.05555555555555555	0.11705719434437026
ternyataaa	0.05555555555555555	0.35426260821659034
kali	0.05555555555555555	0.22919084162734066
mo	0.05555555555555555	0.30335756755691506
beli	0.05555555555555555	0.10978466264732277
sini	0.05555555555555555	0.3157544315188156
model	0.05555555555555555	0.18783303746359087
aga	0.05555555555555555	0.30335756755691506
gede	0.05555555555555555	0.17770406208614897
dikit	0.05555555555555555	0.22104620417223644
gapapa	0.05555555555555555	0.2932285921794731
kaos	0.05555555555555555	0.30335756755691506
kaki	0.05555555555555555	0.17770406208614897
tebel	0.05555555555555555	0.28466466552240327
pas	0.05555555555555555	0.19368639944458124

Gambar 4. 12 Hasil *TFIDF* Dokumen B (Tanpa Normalisasi)

term	TF	TF-IDF
asli	0.05	0.14961683422764266
parah	0.1	0.3514526066969159
bagus	0.05	0.08857783809552677
sangat	0.05	0.07843079589569227
sekali	0.05	0.1666102255087602
mau	0.05	0.13829045176272012
beli	0.05	0.09641052859779563
sini	0.05	0.2730218108012236
model	0.05	0.16781510308771322
agak	0.05	0.22154083994216567
besar	0.05	0.13077634161025323
sedikit	0.05	0.1690497337172318
baik	0.05	0.15222612188617116
pakai	0.05	0.09641052859779563
kaos	0.05	0.2730218108012236
kaki	0.05	0.1609863263374257
tebal	0.05	0.22924837393352862
jadi	0.05	0.14713698717067403
pas	0.05	0.17294781079071742

Gambar 4. 13 Hasil *TFIDF* Dokumen B (Dengan Normalisasi)

4.3.3 Seleksi fitur

Berikut merupakan hasil perhitungan menggunakan algortima seleksi fitur yaitu *TFIDF L1 Norm* dan *Mutual Information*.

1. *TFIDF L1 Norm*

Serupa dengan *TFIDF* tahap sebelumnya, namun *TFIDF L1* akan menambahkan fungsi *normalize()* yang ada pada *library scikitlearn.preprocesing*. Pada perhitungan *TFIDF L1 Norm* akan diterapkan *n_gram* yaitu *unigram* (satu *term*), *bigram* (dua *term*), dan *triagram* (tiga *term*). Pada fungsi ini terdapat beberapa parameter yaitu:

a. X : merupakan data yang akan di *normalize* dengan bentuk format CSR.
b. Norm : normalisasi yang digunakan untuk *normalize* setiap *non zero* sampel (atau setiap *non zero* fitur jika axis adalah 0). Pada penelitian ini akan digunakan *norm l1*.
c. Axis : digunakan untuk menormalkan data. Pada penelitian digunakan 1 yang artinya akan menormalkan setiap sampel.
d. *Copy*: *default* adalah *true* maka akan melakukan salinan (tidak menimpa baris saat normalisasi).

e. *Return_norm* : perintah ini menginstruksikan untuk mengembalikan perhitungan normalisasi. Secara default adalah *false* sehingga proses ini tidak dilakukan.

Contoh hasil *TFIDF L1 Norm* dapat dilihat sebagai berikut.

```
Show TFIDF sample ke-0

['asli', 'parah', 'parah', 'bagus', 'sangat', 'sekali', 'mau', 'beli', 'sini', 'model', 'agak', 'besar', 'sedikit', 'baik', 'pa
kai', 'kaos', 'kaki', 'tebal', 'jadi', 'pas']

                        TF              IDF             TF-IDF          Term

array position 7        0.050000        5.504925        0.275246        agak
array position 61       0.050000        4.009431        0.200472        asli
array position 77       0.050000        2.776569        0.138828        bagus
array position 82       0.050000        4.026823        0.201341        baik
array position 115      0.050000        2.934076        0.146704        beli
array position 128      0.050000        3.615527        0.180776        besar
array position 364      0.050000        3.942740        0.197137        jadi
array position 397      0.050000        4.157851        0.207893        kaki
array position 408      0.050000        6.683580        0.334179        kaos
array position 540      0.050000        3.765809        0.188290        mau
array position 560      0.050000        4.380995        0.219050        model
array position 607      0.050000        2.934076        0.146704        pakai
array position 613      0.100000        4.543514        0.454351        parah
array position 614      0.050000        4.486355        0.224318        pas
array position 708      0.050000        2.572706        0.128635        sangat
array position 723      0.050000        4.406312        0.220316        sedikit
array position 731      0.050000        4.356302        0.217815        sekali
array position 784      0.050000        6.683580        0.334179        sini
array position 865      0.050000        5.671979        0.283599        tebal
```

Gambar 4. 14 Nilai *TFIDF L1* Dokumen B

2. *Mutual Information*

Perhitungan *TFIDF* dengan *L1* yang disimpan dalam bentuk *sparse matrix* dan kemudian akan digunakan untuk mencari nilai *Mutual Information (MI)*. Fungsi ini memiliki beberapa parameter yatu:

a. X : yaitu *feature matrix* yang berupa *array_like* atau *sparse matrix*. Pada penelitian akan digunakan *sparse matrix* yang telah dihitung *tfidf* secara l1.
b. Y: yaitu target *vector* berupa *array_like*. Pada penelitian menggunakan kolom sentimen
c. *Discreate_features*: disetel secara *default* yaitu *auto* yang berarti penugasan akan bernilai *false* untuk *dense* X dan bernilai *true* untuk *sparse* X.
d. *N_neighbor*: jumlah tetangga yang digunakan untuk estimasi *MI* untuk variabel kontinyu. Nilai yang lebih tinggi akan mengurangi varian estimasi, tetapi bias menimbulkan bias. Secara *default n-neighbors* bernilai 3.
e. *Copy*: secara *default* adalah *true* maka akan membuat salinan data.

f. *Random_state*: menentukan sejumlah angka acak untuk menambahkan *noise* kecil ke variabel kontinu untuk menghapus nilai berulang. Secara *default* diatur sebagai *none*.

Pada penelitian ini akan menerapkan *threshold MI* sebesar 0,01. Sehingga fitur yang akan dipilih hanya yang nilai *MI* nya ≥ 0,01. Pada perhitungan *MI* akan diterapkan *n_gram* yaitu *unigram* (satu *term*), *bigram* (dua *term*), dan *triagram* (tiga *term*). Contoh hasil *MI* dapat dilihat sebagai berikut.

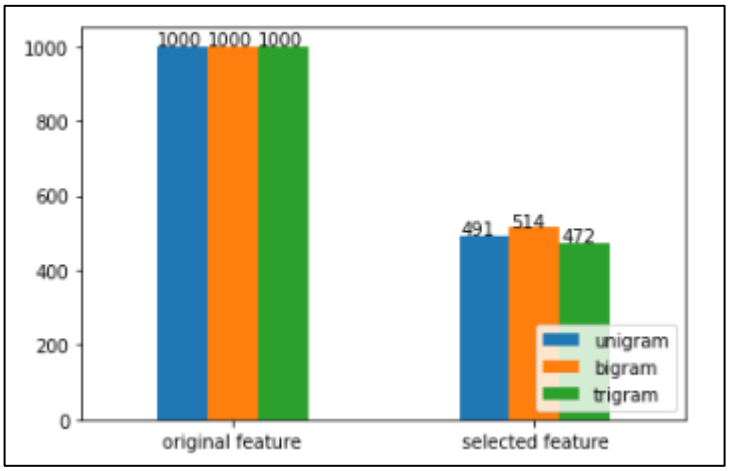
	term	rank	MI_unigram
77	bagus	107.853348	0.896436
708	sangat	97.417852	0.734197
938	ukur	86.297728	0.720080
896	tidak	83.593544	0.000000
528	mantap	80.758121	1.000000
153	buat	68.858971	0.313723
208	converse	67.970710	0.065826
607	pakai	67.215039	0.000000
437	keren	66.250206	0.142933
501	lokal	61.733140	0.457545

	term	rank	MI_bigram
14	bagus sangat	67.706647	0.000000
547	merek lokal	54.214087	0.290160
510	lokal bangga	46.543247	0.000000
972	ukur besar	45.584478	0.446503
686	produk lokal	37.152235	0.270420
173	buat ukur	32.996719	0.000000
568	mirip vans	32.562247	0.375425
236	converse vans	32.506161	0.000000
866	suka sangat	30.465545	0.173973
478	lebih bagus	28.319789	0.000000

	term	rank	MI_trigram
48	buat desain sendiri	41.973048	0.373524
158	mirip sangat converse	26.734319	0.000000
117	kualitas bagus sangat	24.765005	0.000000
974	tidak sesal beli	18.304569	0.159420
381	pakai duduk jauh	17.428155	0.000000
87	jadi buat desain	17.428155	0.420863
12	bagus sangat tidak	17.428155	0.401825
994	vans dengan converse	16.266278	0.143933
57	butuh buat desain	16.266278	0.252516
55	buat ukur besar	15.595019	0.017153

Gambar 4. 15 Nilai *MI* Dokumen B

Dari nilai *MI* yang telah diperoleh, akan disaring dengan nilai *MI* ≥ 0,01. Hasil pengurangan fitur dapat dilihat pada gambar 4.16 berikut. Setelah melalui proses penyaringan ini akan didapatkan *sparse matrix* baru yang memiliki *feature* lebih kecil.



Gambar 4. 16 Grafik Perbandingan Setelah Penyeleksian Fitur

4.4 Pembentukan Data *Training* dan *Testing*

Proses pembentukkan data *training* dan *testing* dikenal sebagai *splitting*. Fungsi yang digunakan dalam proses *split* adalah *model_selection.train_test_split*. Dengan menggunakan fungsi ini tidak lagi diperlukan untuk membagi *dataset* secara manual. Secara pengaturan, *SKLearn model_selection.train_test_split* akan membuat bagian dari dua *dataset* secara acak. Dalam penggunaannya, juga dapat disesuaikan *random state* secara spesifik yang akan diterapkan pada operasi. Dalam fungsi ini terdapat beberapa parameter yang dapat dilihat pada *syntax* berikut.

model_selection.train_test_split (X, y, test_size=0.2, random_state=77)

a. *Array* yaitu urutan *index* dengan panjang atau bentuk yang sama dan *input* yang diizinkan adalah *list*, *array numpy*, matriks *scipy-sparse*, atau kerangka data *pandas*. Pada penelitian digunakan X yang mewakili teks review dan y yang mewakili sentimen.
b. *Test size* yaitu proporsi dari *dataset* termasuk dalam pemecahan data uji. Dalam penelitian ini digunakan sebesar 0.2.
c. *Train size* yaitu proporsi *dataset* termasuk dalam pemecahan data latih. Dan secara otomatis mengikuti ukuran sisa dari *test size*.
d. *Random_state* digunakan untuk mengontrol pemilihan acak yang diterapkan pada data sebelum menerapkan pemisahan. Dengan memberikan int untuk *output* yang

dapat dihasilkan melalui beberapa panggilan fungsi. Setelah mencoba beberapa kali percobaan digunakan *random_state* yaitu 77.

e. *Shuffle* yaitu mengacak data atau tidak sebelum memisahkan. Pada penelitian digunakan *default* sehingga *shuffle* diterapkan pada data.
f. *Stratfy* pada penelitian di atur sebagai *default* sehingga data tidak dibagi secara bertingkat.

Dalam pemprosesan *dataset* dikenal kasus *overfitting*, yaitu ketika model menunjukkan akurasi yang hampir sempurna saat melatih data *training*. Kasus ini terjadi ketika model memiliki beberapa aturan yang kompleks. Ketika suatu model *overfitting*, bisa jadi menjadi tidak akurat ketika melatih data baru. Sebaliknya kasus *underfitting*, yaitu ketika model tidak sesuai dengan data *training* karena serangkaian aturan yang terlalu sederhana. Pada model yang kurang tepat untuk membuat prediksi tentu tidak dapat diandalkan. Menggunakan *dataset* yang sama untuk pelatihan dan pengujian dapat meningkatkan peluang dalam kesalahan perhitungan, sehingga meningkatkan kemungkinan prediksi yang tidak akurat. Melalui proses *split* menggunakan *model_selection.train_test_split* sangat membantu membagi *dataset* untuk mendekatkan kepada model yang ideal (BitDegree, 2020).

4.5 Proses Pengolahan Data *Training*

4.5.1 Proses Klasifikasi

Sesuai dengan hasil kajian pustaka pada bab ii, penelitian ini akan menggunakan dua algoritma klasifikasi yaitu *SVM* dan *Naïve Bayes*. Pada penggunaan modul *Mutinomialnb* di *SKLearn* terdapat beberapa parameter, yaitu:

a. *Alpha* : merupakan parameter penghalusan aditif *(Laplace atau Lidstone)*. Pada penelitian disetel *default* yaitu bernilai 1.0.
b. *Fit_prior* : perintah ini menginstruksikan untuk mempelajari probabilitas kelas sebelumnya atau tidak. Secara *default* diatur secara *true* maka proses ini dilakukan.
c. *Class_prior* : diatur secara *default* yaitu *none* yang berarti *prior* probabilitas kelas sebelumnya disesuaikan menurut data.

Sedangkan pada modul *SVM.SVC* juga terdapat beberapa parameter dalam penggunaannya yaitu:

a. C : merupakan parameter regulasi. Kekuatan regulasi berbanding terbalik dengan C. Untuk percobaan awal akan disetel secara *default* yaitu bernilai C = 1.0.
b. Kernel : perintah ini akan menentukan jenis kernel yang akan digunakan dalam algoritma. Pada percobaan awal akan diatur kernel = linear.
c. *Degree* : merupakan tingkat fungsi pada kernel polynomial *('poly').* Secara *default* bernilai 3, tetapi akan diabaikan bila kernel bukan *poly*.
d. Gamma : merupakan koefisien kernel untuk *rbf*, *poly*, dan *sigmoid*. Pada percobaan awal digunakan gamma *auto* yang berarti akan menggunakan 1 / n_fitur. Secara *default* akan diatur yaitu *'scale'.*
e. *Coef0* : merupakan independen *term* dalam fungsi kernel. Akan bernilai signifikan bila kernel adalah *poly* dan *sigmoid*. Secara *default* diatur bernilai 0.0
f. *Shrinking* : menentukan apakah akan menggunkan *shrinking heuristic*. Secara *default* akan diatur yaitu *true*.
g. *Probability* : menentukan apakah akan mengaktifkan estimasi probabilitas. Secara internal akan menggunakan *5-fold-cross-validation* dan hasil *predict_proba* mungkin tidak konsisten dengan *predict*. Pada percobaan pertama disetel secara *default* yaitu *false*.
h. Tol : merupakan toleransi untuk menghentikan kriteria. Secara *default* diatur sebesar 1e-3.
i. *Cache_size* : menentukan ukuran *cache* kernel (dalam MB). Secara *default* diatur sebesar 200.
j. *Class_weight* : disetel secara *default* yaitu *none* yang berarti semua kelas memiliki bobot satu.
k. *Verbose* : untuk mengaktifkan *output verbose*. Secara *default* diatur secara *false*.
l. *Max_iter* : yaitu batas pada iterasi dalam penyelesaiannya atau -1 untuk *no limit*. Secara *default* diatur bernilai -1.
m. *Decision_function_shape* : secara *default* diatur secara *ovr* yang berarti akan mengembalikan fungsi keputusan dalam bentuk *one-vs-rest (n_samples, n_classes)* sebagai semua pengklasifikasi.

n. *Break_ties* : secara *default* diatur secara *false* yang berarti kelas pertama diantara kelas terikat akan dikembalikan (tidak memutus hubungan).
o. *Random_state* : mengontrol pembuatan angka acak pseudo untuk mengacak data saat estimasi probabilitas. Secara *default* diatur secara *none*.

Dalam memilih model dengan performansi paling baik dan sesuai dengan *dataset* maka dilakukan serangkaian percobaan yaitu menemukan jumlah data optimal, melakukan tahapan proses *pre-processing*, dan menggunakan *hyperparameter*. Hasil yang diperoleh dari serangkaian percobaan tersebut yaitu sebagai berikut.

1. Tanpa proses normalisasi kata pada *pre-processing*

Percobaan pertama dilakukan dengan mengklasifikasi data tanpa melakukan proses normalisasi pada *pre-processing*. Berikut merupakan hasil performansi pada percobaan pertama.

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul unt

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.57      0.51      0.54        77
      netral       0.44      0.42      0.43        66
     positif       0.78      0.88      0.83        93

    accuracy                           0.63       236
   macro avg       0.60      0.60      0.60       236
weighted avg       0.62      0.63      0.62       236
```

Gambar 4. 17 *Naïve* Percobaan Pertama Pada Dokumen B dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  62.71186440677966
```

Gambar 4. 18 *SVM* Percobaan Pertama Pada Dokumen B dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  61.86440677966102
```

Gambar 4. 19 *Naïve* Percobaan Pertama Pada Dokumen B dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  59.32203389830508
```

Gambar 4. 20 *SVM* Percobaan Pertama Pada Dokumen B dengan *TFIDF*

2. Menambahkan normalisasi kata pada *pre-processing*

Percobaan kedua dilakukan dengan mengklasifikasi data dengan melakukan proses normalisasi pada *pre-processing*. Berikut merupakan hasil performansi pada percobaan kedua.

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul unt

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.67      0.52      0.58        77
      netral       0.53      0.61      0.57        66
     positif       0.83      0.90      0.87        93

    accuracy                           0.69       236
   macro avg       0.68      0.68      0.67       236
weighted avg       0.69      0.69      0.69       236
```

Gambar 4. 21 *Naive* Percobaan Kedua Pada Dokumen B dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  68.22033898305084
```

Gambar 4. 22 *SVM* Percobaan Kedua Pada Dokumen B dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  66.94915254237289
```

Gambar 4. 23 *Naive* Percobaan Kedua Pada Dokumen B dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  65.2542372881356
```

Gambar 4. 24 *SVM* Percobaan Kedua Pada Dokumen B dengan *TFIDF*

3. Menambahkan normalisasi dan menggunakan seleksi fitur

Percobaan ketiga dilakukan dengan mengklasifikasi data dengan melakukan proses normalisasi pada *pre-processing* dan menerapkan seleksi fitur. Berikut merupakan hasil performansi pada percobaan ketiga.

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)

Naive Bayes Accuracy Score -> 66.94915254237289
```

Gambar 4. 25 *Naive* Percobaan Ketiga Pada Dokumen B dengan *TFIDF L1*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)

SVM Accuracy Score ->  68.22033898305084
```

Gambar 4. 26 *SVM* Percobaan Ketiga Pada Dokumen B dengan *TFIDF L1*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)

Naive Bayes Accuracy Score -> 61.86440677966102
```

Gambar 4. 27 *Naive* Percobaan Ketiga Pada Dokumen B dengan *MI*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)

SVM Accuracy Score ->  63.983050847457626
```

Gambar 4. 28 *SVM* Percobaan Ketiga Pada Dokumen B dengan *MI*

4. Menambahkan normalisasi dan menggunakan *hyperparameter turning*.

Library SKLearn memfasilitasi pencarian lengkap atas nilai parameter yang ditentukan untuk *estimator* atau yang disebut juga proses *hyperparameter turning*. Pada proses ini akan digunakan fungsi *GridSearchCV* untuk klasifikasi menggunakan *SVM.SVC*. *GridSearchCV* mengimplementasikan metode *"fit"* dan *"score"*. Algoritma ini mengimplementasikan *"predict"* dan *"predict_proba", "decision_function", "transform"* dan *"inverse_transform"* yang diterapkan dalam *estimator* yang digunakan. Parameter *estimator* yang digunakan untuk menerapkan metode ini dioptimalkan dengan pencarian jaringan yang divalidasi silang melalui parameter *grid*. Fungsi ini memiliki beberapa parameter yaitu:

a. *Estimator* : perintah ini mengasumsikan untuk menggunakan *sklearn estimator*.
b. *Param_grid* : merupakan kamus yang berisi daftar pengaturan parameter untuk digunakan sebagai nilai. Pencarian akan melalui urutan sesuai kamus ini.
c. *Scoring* : perintah ini menginstruksikan untuk mengevaluasi beberapa *metric*, dengan memberikan daftar *string* atau *dict* sebagai kunci dan *callable* sebagai nilai. Secara *default* akan diatur secara *none* yang berarti metode skor *estimator* akan digunakan.
d. *N_jobs* : merupakan jumlah pekerjaan yang dijalankan secara parallel. Secara *default* diatur secara *none* yang berarti 1 kecuali dalam konteks *joblib.parallel_backend*.
e. *Pre_dispatch* : mengontrol jumlah pekerjaan yang dikirim selama eksekusi parallel. Secara *default* diatur secara *n_jobs*.
f. *Iid* : secara *default* diatur secara *false* yang berarti tidak diasumsikan terdistribusi secara identik di seluruh *fold* (iterasi).
g. *Cv* : menentukan strategi pemisahan validasi silang. Secara *default* diatur secara *none* yang berarti menggunakan *5-fold cross validation*.
h. *Refit* : perintah untuk memasang kembali *estimator* dengan menggunakan parameter terbaik yang ditemukan pada seluruh *dataset*.
i. *Verbose* : menginstruksikan untuk mengontrol verbositas, semakin tinggi, semakin banya pesan. Pada model akan diatur sebesar 3.
j. *Error_score* : yaitu nilai yang dierikan pada skor jika terjadi kesalahan pada *fitting estimator*. Secara *default* diatur secara *np.nan* maka *FitFailedWarning* akan muncul.

k. *Return_train_score* : secara *default* diatur sebagai *false* yang berarti atribut *cv_results_* tidak akan menyertakan skor pelatihan.

Sedangkan untuk klasifikasi *MultinomialNB*, dalam menerapkan *hyperparameter turning* akan menggunakan *pipeline*. *Pipeline* merupakan fungsi yang mentransformasikan *final estiomator*. *Pipeline* akan secara berurutan menerapkan daftar transformasi dan *final estimator*. Tujuan *pipeline* adalah mengumpulkan beberapa langkah yang dapat divalidasi silang bersama-sama sambil mengatur parameter yang berbeda. Pada penelitian ini *pipeline* akan berisi pengklasifikasi yang terdiri dari *CountVectorizer*, *Tfidftransformer* dan *MultinomialNB*. Kemudian menentukan kombinasi parameter yang ingin diuji pada *pipeline* untuk mendapatkan hasil terbaik. Dalam model akan digunakan parameter *n_gram*, *idf*, *tfidf*, normalisasi dan nilai *alpha*.

a. Tanpa Seleksi Fitur

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

     negatif     0.6379    0.4805    0.5481        77
      netral     0.4941    0.6364    0.5563        66
     positif     0.8817    0.8817    0.8817        93

    accuracy                         0.6822       236
   macro avg     0.6713    0.6662    0.6621       236
weighted avg     0.6938    0.6822    0.6819       236
```

Gambar 4. 29 *Naive* Percobaan Keempat Pada Dokumen B

```
: grid_predictions = grid.predict(Test_X_Tfidf)

  # print classification report
  print(classification_report(Test_Y, grid_predictions))

              precision    recall  f1-score   support

           0       0.56      0.45      0.50        77
           1       0.47      0.65      0.54        66
           2       0.94      0.82      0.87        93

    accuracy                           0.65       236
   macro avg       0.65      0.64      0.64       236
weighted avg       0.68      0.65      0.66       236
```

Gambar 4. 30 *SVM* Percobaan Keempat Pada Dokumen B (*TFIDF*) Dengan *Kernel* Linear

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.54	0.61	0.57	77
netral	0.54	0.53	0.53	66
positif	0.89	0.81	0.85	93
accuracy			0.67	236
macro avg	0.66	0.65	0.65	236
weighted avg	0.68	0.67	0.67	236

Gambar 4. 31 *SVM* Percobaan Keempat Pada Dokumen B (*BoW*) Dengan *Kernel* Linear

```
grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

	precision	recall	f1-score	support
0	0.59	0.47	0.52	77
1	0.48	0.65	0.55	66
2	0.93	0.85	0.89	93
accuracy			0.67	236
macro avg	0.67	0.66	0.65	236
weighted avg	0.69	0.67	0.67	236

Gambar 4. 32 *SVM* Percobaan Keempat Pada Dokumen B (*TFIDF*) Dengan *Kernel RBF*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.64	0.58	0.61	77
netral	0.56	0.68	0.62	66
positif	0.91	0.84	0.87	93
accuracy			0.71	236
macro avg	0.70	0.70	0.70	236
weighted avg	0.72	0.71	0.72	236

Gambar 4. 33 *SVM* Percobaan Keempat Pada Dokumen B (*BoW*) Dengan *Kernel RBF*

b. Dengan Seleksi Fitur

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.60      0.51      0.55        77
      netral       0.52      0.68      0.59        66
     positif       0.89      0.82      0.85        93

    accuracy                           0.68       236
   macro avg       0.67      0.67      0.67       236
weighted avg       0.69      0.68      0.68       236
```

Gambar 4. 34 *SVM* Percobaan Keempat Pada Dokumen B *(TFIDF L1)* Dengan *Kernel* Linear

```
]: grid_predictions = grid.predict(X_test)

   # print classification report
   print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.67      0.40      0.50        77
      netral       0.47      0.80      0.60        66
     positif       0.94      0.78      0.85        93

    accuracy                           0.67       236
   macro avg       0.69      0.66      0.65       236
weighted avg       0.72      0.67      0.67       236
```

Gambar 4. 35 *SVM* Percobaan Keempat Pada Dokumen B *(TFIDF L1)* Dengan *Kernel RBF*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.64      0.36      0.46        77
      netral       0.47      0.71      0.57        66
     positif       0.84      0.84      0.84        93

    accuracy                           0.65       236
   macro avg       0.65      0.64      0.62       236
weighted avg       0.67      0.65      0.64       236
```

Gambar 4. 36 *SVM* Percobaan Keempat Pada Dokumen B *(MI)* Dengan *Kernel* Linear

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.62      0.45      0.53        77
      netral       0.51      0.74      0.60        66
     positif       0.90      0.82      0.86        93

    accuracy                           0.68       236
   macro avg       0.68      0.67      0.66       236
weighted avg       0.70      0.68      0.68       236
```

Gambar 4. 37 *SVM* Percobaan Keempat Pada Dokumen B *(MI)* Dengan *Kernel RBF*

4.5.2 Analisa dan Evaluasi Algoritma

Evaluasi berfungsi untuk mengetahui akurasi dari model algoritma yang diusulkan. Validasi digunakan untuk melihat perbandingan hasil akurasi dari model yang digunakan dengan hasil yang telah ada sebelumnya. Teknik validasi yang digunakan adalah *10-Fold Cross Validation*. Pada saat melakukan validasi, urutan dari kumpulan dokumen yang ada akan diacak. Hal ini bertujuan untuk menghindari adanya pengelompokkan dokumen yang berasal dari ketegori tertentu. Hasil pengujian masing-masing dapat dilihat pada tabel dibawah ini.

Tabel 4. 4 Hasil *10-Cross Validation*

K		*precision*	*recall*	*f1-score*	*support*
1	negatif	0,56	0,53	0,55	34
	netral	0,51	0,68	0,58	31
	positif	0,93	0,79	0,86	53
	accuracy			0,69	118
2	negatif	0,71	0,63	0,67	43
	netral	0,62	0,69	0,65	35
	positif	0,88	0,9	0,89	40
	accuracy			0,74	118
3	negatif	0,82	0,89	0,85	46
	netral	0,83	0,69	0,76	36
	positif	0,89	0,94	0,92	36
	accuracy			0,85	118
4	negatif	0,81	0,91	0,86	33
	netral	0,94	0,81	0,87	37
	positif	0,98	1	0,99	48
	accuracy			0,92	118

K		*precision*	*recall*	*f1-score*	*support*
5	negatif	0,84	0,86	0,85	43
	netral	0,87	0,85	0,86	40
	positif	0,97	0,97	0,97	35
	accuracy			0,89	118
6	negatif	0,92	0,83	0,88	42
	netral	0,87	0,87	0,87	47
	positif	0,85	0,97	0,9	29
	accuracy			0,88	118
7	negatif	0,82	0,89	0,86	37
	netral	0,93	0,83	0,88	47
	positif	0,94	1	0,97	33
	accuracy			0,9	117
8	negatif	0,79	0,79	0,79	39
	netral	0,81	0,83	0,82	36
	positif	0,95	0,93	0,94	42
	accuracy			0,85	117
9	negatif	0,86	0,97	0,91	33
	netral	0,98	0,86	0,92	51
	positif	0,91	0,97	0,94	33
	accuracy			0,92	117
10	negatif	0,84	0,88	0,86	42
	netral	0,84	0,84	0,84	32
	positif	0,93	0,88	0,9	43
	accuracy			0,87	117
rata-rata *accuracy*				0,850463	

Nilai akurasi yang dihasilkan pada setiap *fold* memiliki hasil yang bervariatif. Akurasi tertinggi pada *fold*-3, 4 dan 9 dengan akurasi yaitu 0,92. Akurasi terendah pada *fold*-1 yaitu dengan akurasi 0,69. Perbedaan akurasi pada setiap *fold* disebabkan karena karakteristik data uji dan data latih pada setiap *fold* berbeda-beda. Pada *fold*-4 dan 9 menghasilkan akurasi tinggi karena pada data uji dan data latih memiliki kedekatan karakteristik, sedangkan pada *fold*-1 antara data latih dan data uji memiliki karakteristik yang cukup jauh. Rata-rata akurasi yang diperoleh yakni 0.850463.

Selain itu evaluasi algoritma juga dilakukan dengan kurva *ROC (Receiver Operating Characteristic)*. Kurva *ROC* dapat digunakan untuk mendapatkan nilai *AUC*. Nilai *AUC* digunakan untuk menentukan keakuratan klasifikasi pengujian *diagnostic*. Menurut (Gorunescu, 2011) tingkat nilai diagnosa adalah sebagai berikut:

a. Akurasi bernilai 0,90-1,00 = *excellent classification*

b. Akurasi bernilai 0,80-0,90 = *good classification*
c. Akurasi bernilai 0,70-0,80 = *fair classification*
d. Akurasi bernilai 0,60-0,70 = *poor classification*
e. Akurasi bernilai 0,50-0,60 = *failure*

Berdasar hasil yang diperoleh maka secara nilai *AUC* akurasi model ada pada tingkat *good classification*. Hasil nilai *ROC AUC* seperti pada gambar 4.38.

```
One-vs-One ROC AUC scores:
0.845645 (macro),
0.854584 (weighted by prevalence)
One-vs-Rest ROC AUC scores:
0.858485 (macro),
0.866732 (weighted by prevalence)
```

Gambar 4. 38 Hasil *ROC AUC* Model Terpilih

4.6 Prediksi dan Asosiasi Kata

Sesuai hasil yang diperoleh, selanjutnya akan dilakukan proses prediksi data dengan model terpilih. Data *input* yang akan diprediksi yaitu data *review* keseluruhan yang berjumlah 1673 data *review*. Sebelumnya data 1673 akan melalui proses *pre-processing* yang telah disesuaikan dengan karakter data. Hasil *pre-processing* dan prediksi disajikan pada gambar berikut.

	text	text_clean	text_clean1	text_clean_stopword	text_normalized	text_clean_stemmed
0	Asliii sih parah parahh. bagus bgt ternyataaa....	asliii sih parah parahh bagus bgt ternyataaa l...	[asliii, sih, parah, parahh, bagus, bgt, terny...	[asliii, parah, parahh, bagus, bgt, kali, mo, ...	[asli, parah, parah, bagus, sangat, sekali, ma...	[asli, parah, parah, bagus, sangat, sekali, ma...
1	Masyaa Allah. inj sepwtu lokal tapi kualitasny...	masyaa allah inj sepwtu lokal tapi kualitasnya...	[masyaa, allah, inj, sepwtu, lokal, tapi, kual...	[masyaa, lokal, kualitasnya, bersaing, produk,...	[mantap, lokal, kualitasnya, bersaing, produk,...	[mantap, lokal, kualitas, saing, produk, luar,...
2	oiya buat tmn2 yg mau beli ventela klo bisa si...	oiya buat tmn yg mau beli ventela klo bisa siz...	[oiya, buat, tmn, yg, mau, beli, ventela, klo,...	[buat, tmn, mau, beli, size, naik, biasa, pke,...	[buat, teman, mau, beli, ukuran, naik, biasa, ...	[buat, teman, mau, beli, ukur, naik, biasa, pa...
3	Local brand tp bagus. Dipakenya juga enak. Bah...	local brand tp bagus dipakenya juga enak bahan...	[local, brand, tp, bagus, dipakenya, juga, ena...	[local, brand, tp, bagus, dipakenya, enak, bah...	[lokal, merek, tapi, bagus, dipakainya, enak, ...	[lokal, merek, tapi, bagus, pakai, enak, bahan...
4	Kualitas bagus. Solnya empuk. Saya pikir dapat...	kualitas bagus solnya empuk saya pikir dapat b...	[kualitas, bagus, solnya, empuk, saya, pikir, ...	[kualitas, bagus, solnya, empuk, pikir, bonus,...	[kualitas, bagus, solnya, empuk, pikir, bonus,...	[kualitas, bagus, sol, empuk, pikir, bonus, ta...
...	...	...	...	...	...	...
1668	Kalau punya desain sendiri bakal lebih bagus	kalau punya desain sendiri bakal lebih bagus	[kalau, punya, desain, sendiri, bakal, lebih, ...	[punya, desain, sendiri, lebih, bagus]	[punya, desain, sendiri, lebih, bagus]	[punya, desain, sendiri, lebih, bagus]
1669	Sunggug sungguh worth it . dibanding beli conv...	sunggug sungguh worth it dibanding beli conver...	[sunggug, sungguh, worth, it, dibanding, beli,...	[sunggug, sungguh, worth, it, dibanding, beli,...	[sungguh, sungguh, nilai, itu, dibanding, beli...	[sungguh, sungguh, nilai, itu, banding, beli, ...
1670	Diliat dari segi mana pun itu mirip converse	diliat dari segi mana pun itu mirip converse	[diliat, dari, segi, mana, pun, itu, mirip, co...	[diliat, segi, mirip, converse]	[dilihat, segi, mirip, converse]	[lihat, segi, mirip, converse]
1671	Kykny bntukn logony yg prlu ad refisi disain. ...	kykny bntukn logony yg prlu ad refisi disain ...	[kykny, bntukn, logony, yg, prlu, ad, refisi, ...	[bntukn, logony, prlu, refisi, disain, bikn, l...	[bentuknya, logo, perlu, revisi, desain, bikn,...	[bentuk, logo, perlu, revisi, desain, bikn, lo...
1672	Iya logo nya agak kurang sreg sih gua	iya logo nya agak kurang sreg sih gua	[iya, logo, nya, agak, kurang, sreg, sih, gua]	[logo, kurang, sreg]	[logo, kurang, cocok]	[logo, kurang, cocok]

Gambar 4. 39 Hasil *Pre-processing*

```
# Model yang sudah jadi bisa kita gunakan seperti ini

text = count_vect.transform(ubah)
# String ke Angka

# Kita prediksi sentimennya
result = SVM.predict(text)[0:1674]
print("Sentiment:")
print()
print(result)
```

```
Sentiment:

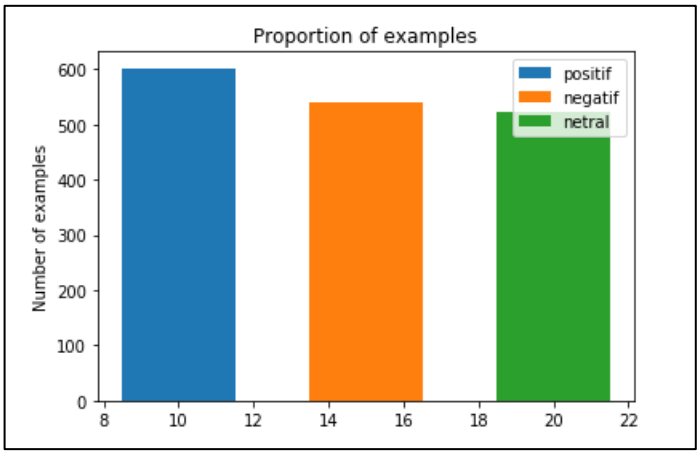
['positif' 'positif' 'positif' ... 'netral' 'negatif' 'negatif']
```

```
df = pd.DataFrame(result, index=result, columns=["result"])
```

```
df
```

	result
positif	positif
positif	positif
positif	positif
positif	positif
positif	positif
...	...
netral	netral
positif	positif
netral	netral
negatif	negatif
negatif	negatif

Gambar 4. 40 Algoritma Prediksi



Gambar 4. 41 Grafik Proporsi Kelas Hasil dari Prediksi

Sebelum memasuki proses *QFD*, hasil prediksi yang akan digunakan sebagai *input QFD* adalah data dengan kelas negatif karena memerlukan perbaikan atau evaluasi. Hasil perhitungan seperti dapat dilihat pada tabel 4.5 dan 4.6.

Tabel 4. 5 Hasil Asosiasi Data Prediksi *(Bigram)*

Rank	*Frequency*		*PMI*		*T-test*		*Chi-square*		*Likelihood Ratio*	
0	(merek, lokal)	29	(abai, terima)	12,250	(merek, lokal)	4,981	(abai, terima)	4872.0	(lokal, bangga)	131,483
1	(lokal, bangga)	25	(pijak, bumi)	12,250	(lokal, bangga)	4,794	(pijak, bumi)	4872.0	(merek, lokal)	112,576
2	(buat, desain)	21	(sering, tekuk)	12,250	(buat, desain)	4,010	(sering, tekuk)	4872.0	(jazz, strip)	70,312
3	(ukur, besar)	14	(sana, sini)	12,250	(ukur, besar)	3,562	(sana, sini)	4872.0	(ciri, khas)	68,000
4	(buat, ukur)	13	(saing, online)	12,250	(merek, luar)	3,190	(saing, online)	4872.0	(ukur, besar)	66,817

Tabel 4. 6 Hasil Asosiasi Data Prediksi *(Trigram)*

Rank	*Frequency*		*PMI*		*T-test*		*Chi-square*		*Likelihood Ratio*	
0	(buat, desain, sendiri)	5	(besok, edar, facebook)	24,5	(buat, desain, sendiri)	2,229	(besok, edar, facebook)	2,37E+13	(merek, lokal, bangga)	389,984
1	(tidak, seperti, compass)	4	(slimfit, estetika, streetwear)	24,5	(tidak, seperti, compass)	1,992	(slimfit, estetika, streetwear)	2,37E+13	(bangga, lokal, bangga)	236,736
2	(bilang, bangga, lokal)	3	(transparan, chelsea, boots)	24,5	(ciri, khas, sendiri)	1,732	(transparan, chelsea, boots)	2,37E+13	(kata, lokal, bangga)	216,057

Rank	*Frequency*		*PMI*		*T-test*		*Chi-square*		*Likelihood Ratio*	
3	(lebih, bagus, buat)	3	(tombol, jebol, bahkan)	24,5	(vans, kamu, terlalu)	1,732	(tombol, jebol, bahkan)	2,37E+13	(lokal, bangga, umpat)	214,585
4	(tidak, ada, ukur)	3	(toecap, half, moon)	24,5	(cinta, produk, indonesia)	1,732	(toecap, half, moon)	2,37E+13	(bukan, lokal, bangga)	214,221

Pada *bigram* masing-masing metode memiliki 3959 term dan pada *trigram* masing-masing memiliki 4743 term. Dari segi informasi, pada penelitian akan menggunakan *trigram*. Proses selanjutnya adalah memilih metode mana yang datanya akan digunakan sebagai *input QFD*. Penelitian oleh (Evert & Krenn, 2001 (a)) mengatakan bahwa prosedur standar dalam mengevaluasi pengukuran asosiasi adalah menggunakan penilaian manual dari kandidat n-terbaik yang telah diidentifikasi. Penilaian manual yang diusulkan adalah menilai *true positive (tp)*, dalam hal ini kaitannya dengan term yang masuk akal pada 50 atau 100 term yang memiliki ranking tertinggi. Penelitian lain oleh (Evert, 2005 (b)) melakukan perbandingan dan evaluasi dengan metode *log-likelihood*, *t-score*, *chi-squared*, *MI* dan *frequency* secara manual pada n-terbaik yang berbeda, dan diperoleh bahwa n = 100 memiliki hasil *precision values* yang paling baik diantara n-terbaik lain. Maka pada penelitian ini pemilihan metode asosiasi akan berdasarkan *true positive* pada peringkat 1-100 masing-masing metode. Metode *t-test* mendapatkan nilai *true positive* terbesar yaitu 88. Maka sebagai *input QFD* akan digunakan data dengan metode *t-test*. Hasil evaluasi seperti pada tabel 4.7.

Tabel 4. 7 Tabel Evaluasi Metode *Collocations*

	Frequency	*PMI*	*T-test*	*Chi-square*	*Likelihood*
True Positive	83	58	88	59	50

4.7 Tahap *QFD*

Pengolahan data selanjutnya dilakukan dengan mengacu pada metode implementasi *Quality Function Deployment (QFD)*. Pada penelitian ini akan diteliti sejauh mana data hasil proses *machine learning* dapat digunakan dalam metode *QFD*. Dengan batasan bahwa peneliti tidak dapat melakukan tahapan *QFD* dengan perusahaan atau *stakeholder* terkait, maka pada tahap *QFD* akan muncul data *dummy* (subjektif peneliti). Hasil dari tahap *QFD* ini lebih lanjut akan dibahas dalam pembahasan.

4.7.1 Identifikasi *Voice of Customer* (Kebutuhan Konsumen)

Identifikasi fitur kebutuhan konsumen dilakukan dengan menggunakan data *review* produk Ventela Public Low peringkat 1-100 dari hasil prediksi yang telah diasosiasikan dengan metode yang telah dipilih sebelumnya. Sebagai produk *benchmark* akan digunakan data *review* produk Converse Chuck Taylor 70s peringkat 1-100 dari hasil dengan tahapan proses yang sama dengan yang diterapkan pada data *review* produk Ventela Public Low.



Gambar 4. 42 Produk Ventela Low Public

(Sumber: ventela.com)



Gambar 4. 43 Produk Converse Chuck Taylor 70s
(Sumber: converse.id)

Proses selanjutnya adalah mencocokkan fitur kebutuhan konsumen Ventela Public Low dangan produk pesaing. Pencocokkan dilakukan per data, dan pemilihan data produk pesaing didasarkan kepada kesamaan kategori dengan data yang dipilih adalah adalah data dengan nilai tertinggi pada kategori tersebut, dengan alasan produk pesaing adalah sebagai pembanding. Tabel 4.8 berisi hasil dari tahapan proses sebelumnya yang akan digunakan dalam tahapan *QFD*. Hasil data yang telah melalui proses pencocokkan dapat dilihat pada tabel 4.9.

Tabel 4. 8 Data *Review* Sesuai Peringkat Nilai Asosiasi sebagai *Input QFD*

Ventela Public Low		Converse Chuck Taylor 70s	
Trigram	t	*Trigram*	*Freq*
('buat', 'desain', 'sendiri')	2,22860361	('lebih', 'beli', 'vans')	3
('tidak', 'seperti', 'compass')	1,99212736	('makin', 'buluk', 'makin')	3
('ciri', 'khas', 'sendiri')	1,732023663	('harga', 'jual', 'gila')	2
('vans', 'kamu', 'terlalu')	1,731879571	('ventela', 'merek', 'lokal')	2
('cinta', 'produk', 'indonesia')	1,731700185	('ukur', 'tidak', 'ingin')	2

Tabel 4. 9 Data *Review* yang Telah Dicocokkan

Ventela Public Low		Converse Chuck Taylor 70s	
Trigram	t	*Trigram*	*Freq*
('buat', 'desain', 'sendiri')	2,22860361	('makin', 'buluk', 'makin')	3
('tidak', 'seperti', 'compass')	1,99212736	('makin', 'buluk', 'makin')	3
('ciri', 'khas', 'sendiri')	1,732023663	('makin', 'buluk', 'makin')	3
('vans', 'kamu', 'terlalu')	1,731879571	('lebih', 'beli', 'vans')	3
('cinta', 'produk', 'indonesia')	1,731700185	('ventela', 'merek', 'lokal')	2

Data yang telah melalui proses pencocokkan kemudian dikelompokkan. Pengelompokkan berlaku untuk produk Ventela Public Low maupun produk pesaing. Pada produk Ventela Public Low pengelompokkan berdasarkan dengan adanya term yang sama, sedangkan untuk produk pesaing pengelompokkan dilakukan dengan mengikuti hasil dari pengelompokkan Ventela Public Low. Proses pengelompokkan seperti dapat dilihat pada tabel 4.10. Lebih lengkap dapat dilihat pada lampiran.

Tabel 4. 10 Proses Pengelompokkan

	Ventela Public Low		Converse	
Kelompok 1	('buat', 'desain', 'sendiri')	2,22860361	('makin', 'buluk', 'makin')	3
	('punya', 'desain', 'sendiri')	1,730218524	('makin', 'buluk', 'makin')	3
	('butuh', 'buat', 'desain')	1,413071414	('makin', 'buluk', 'makin')	3
	('produk', 'indonesia', 'desain')	1,411637026	('makin', 'buluk', 'makin')	3
	('cari', 'buat', 'desain')	1,406218523	('makin', 'buluk', 'makin')	3
	('buat', 'desain', 'baru')	1,404314943	('makin', 'buluk', 'makin')	3

4.7.2 Analisa Tingkat Kinerja Produk dan Pelayanan

Pada tahap ini akan dihitung tingkat kinerja produk dan pelayanan produk Ventela Public. Hasil dari proses asosiasi tiap data telah memiliki nilai, yang mewakili kedekatan antar kata dan seberapa sering term tersebut muncul secara bersamaan. Nilai tersebut akan digunakan untuk menghitung tingkat kinerja produk dan pesaing, mengingat kesamaan konsep yaitu pada kelas *negative* data yang paling sering muncul atau paling sering dibicarakan maka data.tersebut memiliki kinerja yang buruk Perhitungan tingkat kinerja produk dimulai dengan normalisasi pada nilai. Dari 100 data dan merupakan kelas *negative* maka nilai minimum yaitu 2,228604 dan nilai maksimum yaitu 1,400317. Masing-masing hasil pengelompokkan kemudian akan dihitung rata-rata normalisasinya dari nilai data pembentuknya, seperti dapat dilihat pada tabel 4.11. Hasil kinerja produk akhir masing-masing fitur didapatkan dari rata-rata kinerja produk dikalikan dengan bobot masing-masing fitur, seperti dapat dilihat pada tabel 4.12

Tabel 4. 11 Perhitungan Kinerja Produk

	VENTELA		Kinerja Produk	Rata-rata
Kelompok 1	('buat', 'desain', 'sendiri')	2,22860361	0	0,7601147
	('punya', 'desain', 'sendiri')	1,730218524	0,601705995	
	('butuh', 'buat', 'desain')	1,413071414	0,984601314	
	('produk', 'indonesia', 'desain')	1,411637026	0,986333067	
	('cari', 'buat', 'desain')	1,406218523	0,992874887	
	('buat', 'desain', 'baru')	1,404314943	0,995173102	

Tabel 4. 12 Nilai Kinerja Produk Masing-Masing Kelompok

Kelompok	Rata-rata	Bobot	Kinerja Produk	*(roundup)*
Kelompok 1	0,761	0,06	0,04566	0,0457
Kelompok 2	0,796	0,04	0,03184	0,0319
Kelompok 3	0,89	0,04	0,0356	0,0356
Kelompok 4	0,948	0,1	0,0948	0,0948
Kelompok 5	0,793	0,02	0,01586	0,0159
Kelompok 6	0,989	0,08	0,07912	0,0792
Kelompok 7	0,95	0,1	0,095	0,095
Kelompok 8	0,989	0,02	0,01978	0,0198
Kelompok 9	0,964	0,15	0,1446	0,1446
Kelompok 10	0,857	0,03	0,02571	0,0258
Kelompok 11	0,891	0,04	0,03564	0,0357
Kelompok 12	0,286	0,01	0,00286	0,0029
Kelompok 13	0,984	0,04	0,03936	0,0394
Kelompok 14	0,985	0,05	0,04925	0,0493
Kelompok 15	0,984	0,02	0,01968	0,0197
Kelompok 16	0,915	0,1	0,0915	0,0915
Kelompok 17	0,984	0,02	0,01968	0,0197
Kelompok 18	0,984	0,02	0,01968	0,0197
Kelompok 19	0,986	0,01	0,00986	0,0099
Kelompok 20	0,986	0,01	0,00986	0,0099
Kelompok 21	0,984	0,01	0,00984	0,0099
Kelompok 22	0,984	0,01	0,00984	0,0099
Kelompok 23	0,984	0,01	0,00984	0,0099
Kelompok 24	0,984	0,01	0,00984	0,0099

Pada produk pesaing dilakukan langkah perhitungan yang sama, namun terdapat perbedaan karena pada produk pesaing yang digunakan adalah semua kelas. Maka untuk proses normalisasi, nilai minimum yaitu 1 dan nilai maksimum yaitu 3. Hasil tingkat kinerja produk dan pesaing, seperti pada tabel 4.13 berikut.

Tabel 4. 13 Tingkat Kinerja Produk

Kelompok	Ventela	Converse	hasil
Kelompok 1	0,0457	0,06	*bad*
Kelompok 2	0,0319	0,04	*bad*
Kelompok 3	0,0356	0,03	*bad*
Kelompok 4	0,0948	0,05	*bad*
Kelompok 5	0,0159	0,01	*bad*
Kelompok 6	0,0792	0,04	*bad*
Kelompok 7	0,095	0,075	*bad*
Kelompok 8	0,0198	0,01	*bad*
Kelompok 9	0,1446	0,075	*good*
Kelompok 10	0,0258	0,0201	*good*
Kelompok 11	0,0357	0,04	*bad*
Kelompok 12	0,0029	0,01	*bad*
Kelompok 13	0,0394	0,02	*bad*
Kelompok 14	0,0493	0,025	*bad*
Kelompok 15	0,0197	0,01	*bad*
Kelompok 16	0,0915	0,075	*bad*
Kelompok 17	0,0197	0,01	*good*
Kelompok 18	0,0197	0,02	*bad*
Kelompok 19	0,0099	0,01	*bad*
Kelompok 20	0,0099	0,01	*bad*
Kelompok 21	0,0099	0,005	*bad*
Kelompok 22	0,0099	0,005	*bad*
Kelompok 23	0,0099	0,005	*bad*
Kelompok 24	0,0099	0,01	*bad*

4.7.3 Penentuan Tingkat Kepentingan *(Important Ratings)*

Sebelum memasuki tahap penentuan tingkat kepentingan. Hasil pengelompokkan akan dilakukan pembentukan fitur/kriteria. Fitur/kriteria disesuaikan dengan karakteristik produk Ventela, dan mempertimbangkan nilai asosiasi dari produk pesaing. Apabila nilai kinerja produk lebih rendah dari nilai kinerja produk pesaing maka fitur bermakna *negative*, dan sebaliknya bila nilai kinerja produk lebih baik dari kinerja pesaing maka fitur bermakna positif seperti dapat dilihat pada tabel 4.14. Tabel 4.15 berisi 24 fitur kebutuhan konsumen yang akan menempati sisi kiri dari matriks *House of Quality (HoQ)*.

Tabel 4. 14 Pembentukan Fitur/Kriteria

Fitur	Ventela	Converse	hasil
Desain produk	0,0457	0,06	*bad*
Tingkat orisinalitas produk	0,0319	0,04	*bad*
Kemiripan dengan produk yang ada di pasaran	0,0356	0,03	*bad*
Ketersediaan ukuran perlu ditingkatkan	0,0948	0,05	*bad*
Merubah *sizing*	0,0159	0,01	*bad*
Membuat ukuran besar	0,0792	0,04	*bad*
Meniru *Vans*	0,095	0,075	*bad*
Mirip *Converse*	0,0198	0,01	*bad*
Bangga produk lokal	0,1446	0,075	*good*
Produk Indonesia	0,0258	0,0201	*good*
Model Produk	0,0357	0,04	*bad*
Lebih baik dari Compass	0,0029	0,01	*bad*
Ketersediaan warna perlu ditingkatkan	0,0394	0,02	*bad*
Ketersediaan produk perlu ditingkatkan	0,0493	0,025	*bad*
Penjualan perlu ditingkatkan	0,0197	0,01	*bad*
Membentuk identitas produk	0,0915	0,075	*bad*
Harga produk	0,0197	0,01	*good*
ketahanan material	0,0197	0,02	*bad*
Desain siluet	0,0099	0,01	*bad*
Desain logo	0,0099	0,01	*bad*
Perlu promosi produk	0,0099	0,005	*bad*
Bagian kaki depan	0,0099	0,005	*bad*
Penggunaan dengan kaos kaki	0,0099	0,005	*bad*
Hak cipta	0,0099	0,01	*bad*

Tabel 4. 15 Fitur yang Menjadi Kebutuhan Konsumen

No	Fitur / Kriteria Kebutuhan Konsumen
1	Desain produk
2	Tingkat orisinalitas produk
3	Kemiripan dengan produk yang ada di pasaran
4	Ketersediaan ukuran perlu ditingkatkan
5	Merubah *sizing*
6	Membuat ukuran besar
7	Meniru *Vans*
8	Mirip *Converse*
9	Bangga produk lokal
10	Produk Indonesia
11	Model Produk
12	Lebih baik dari Compass
13	Ketersediaan warna perlu ditingkatkan
14	Ketersediaan produk perlu ditingkatkan
15	Penjualan perlu ditingkatkan

No	Fitur / Kriteria Kebutuhan Konsumen
16	Membentuk identitas produk
17	Harga produk
18	ketahanan material
19	Desain siluet
20	Desain logo
21	Perlu promosi produk
22	Bagian kaki depan
23	Penggunaan dengan kaos kaki
24	Hak cipta

Tahap selanjutnya adalah mengukur tingkat kepentingan konsumen *(IR)* terhadap fitur yang diperoleh. Berbeda dengan tingkat kinerja produk yang dapat menggunakan data lain yang tersedia (laporan dan publikasi), pada tahap ini mengukur kepentingan fitur tidak dapat dilakukan dengan input nilai asosiasi, karena tingkat pentingnya suatu fitur lebih baik dilakukan penilaian secara langsung terhadap fitur tersebut (Cudney & Elrod, 2011). Maka pada penelitian ini, nilai *IR* didapatkan dari hasil kuesioner yang telah dibagikan kepada 55 responden pengguna produk Ventela. Untuk menyamakan ukuran, maka proses normalisasi akan dilakukan pada hasil *IR*, dengan nilai maksimum yaitu 5 dan minimum yaitu 1. Tabel 4.16. berisi nilai *IR* masing-masing fitur.

Tabel 4. 16 Nilai *IR* Masing-Masing Fitur

Fitur	Rata-rata *IR*	Normalisasi	*IR (roundup)*	Peringkat
Desain produk	4,709091	0,927273	0,9273	2
Tingkat orisinalitas produk	4,563636	0,890909	0,891	4
Kemiripan dengan produk yang ada di pasaran	3,654545	0,663636	0,6637	18
Ketersediaan ukuran perlu ditingkatkan	4,472727	0,868182	0,8682	6
Merubah *sizing*	3,854545	0,713636	0,7137	16
Membuat ukuran besar	3,690909	0,672727	0,6728	17
Meniru *Vans*	2,8	0,45	0,45	21
Mirip *Converse*	2,963636	0,490909	0,491	20
Bangga produk lokal	4,345455	0,836364	0,8364	10
Produk Indonesia	4,418182	0,854545	0,8546	9
Model Produk	4,309091	0,827273	0,8273	11
Lebih baik dari Compass	3,345455	0,586364	0,5864	19
Ketersediaan warna perlu ditingkatkan	4,436364	0,859091	0,8591	8
Ketersediaan produk perlu ditingkatkan	4,454545	0,863636	0,8637	7
Penjualan perlu ditingkatkan	4,163636	0,790909	0,791	13

Fitur	Rata-rata *IR*	Normalisasi	*IR (roundup)*	Peringkat
Membentuk identitas produk	4,527273	0,881818	0,8819	5
Harga produk	4,618182	0,904545	0,9046	3
ketahanan material	4,854545	0,963636	0,9637	1
Desain siluet	4,145455	0,786364	0,7864	14
Desain logo	3,927273	0,731818	0,7319	15
Perlu promosi produk	4,218182	0,804545	0,8046	12
Bagian kaki depan	4,618182	0,904545	0,9046	3
Penggunaan dengan kaos kaki	4,345455	0,836364	0,8364	10
Hak cipta	4,527273	0,881818	0,8819	5

4.7.4 Menghitung Rasio Perbaikan *(Improvement Ratio)*

Pada tahap ini perlu dilakukan penentuan *goals* dari masing-masing fitur. Terdapat dua cara penentuan *goals*, yaitu sesuai dengan keinginan konsumen dan kebijakan perusahaan sesuai kemampuan yang dimiliki perusahaan. Pada penelitian ini, parameter nominal *goals* mempertimbangkan hasil *benchmark* nilai kinerja antara produk dengan pesaing secara subyektif. Rasio perbaikan (RB) dihitung dengan membandingkan *goals (G)* dengan kinerja saat ini (KS). Hasil rasio perbaikan dengan menggunakan *goals* sesuai dengan keinginan konsumen dapat dilihat pada tabel 4.17.

Tabel 4. 17 Nilai Rasio Perbaikan

Fitur	KS	*G*	RB
Desain produk	0,046	0,05	1,094092
Tingkat orisinalitas produk	0,032	0,05	1,567398
Kemiripan dengan produk yang ada di pasaran	0,036	0,001	0,02809
Ketersediaan ukuran perlu ditingkatkan	0,095	0,001	0,010549
Merubah *sizing*	0,016	0,001	0,062893
Membuat ukuran besar	0,079	0,001	0,012626
Meniru Vans	0,095	0,001	0,010526
Mirip Converse	0,02	0,001	0,050505
Bangga produk lokal	0,145	0,05	0,345781
Produk Indonesia	0,026	0,05	1,937984
Model Produk	0,036	0,05	1,40056
Lebih baik dari Compass	0,003	0,001	0,344828
Ketersediaan warna perlu ditingkatkan	0,039	0,001	0,025381
Ketersediaan produk perlu ditingkatkan	0,049	0,001	0,020284
Penjualan perlu ditingkatkan	0,02	0,001	0,050761
Membentuk identitas produk	0,092	0,001	0,010929

Fitur	KS	*G*	RB
Harga produk	0,02	0,05	2,538071
ketahanan material	0,02	0,05	2,538071
Desain siluet	0,01	0,05	5,050505
Desain logo	0,01	0,05	5,050505
Perlu promosi produk	0,01	0,001	0,10101
Bagian kaki depan	0,01	0,05	5,050505
Penggunaan dengan kaos kaki	0,01	0,05	5,050505
Hak cipta	0,01	0,05	5,050505

4.7.5 Menentukan Titik Penjualan *(Sales Point)*

Titik penjualan ditetapkan untuk megetahui seberapa besar *benefit* yang dapat diperoleh bila terdapat perubahan pada atribut-atribut tertentu. Dengan berdasarkan keinginan konsumen, hasil titik penjualan *(SP)* pada tiap fitur dapat dilihat pada tabel 4.18.

Tabel 4. 18 Nilai *Sales Point*

Fitur	*SP*
Desain produk	1,5
Tingkat orisinalitas produk	1,5
Kemiripan dengan produk yang ada di pasaran	1,5
Ketersediaan ukuran perlu ditingkatkan	1,5
Merubah *sizing*	1,5
Membuat ukuran besar	1,5
Meniru *Vans*	1,5
Mirip *Converse*	1,5
Bangga produk lokal	1,5
Produk Indonesia	1,5
Model Produk	1,5
Lebih baik dari Compass	1,5
Ketersediaan warna perlu ditingkatkan	1,5
Ketersediaan produk perlu ditingkatkan	1,5
Penjualan perlu ditingkatkan	1,5
Membentuk identitas produk	1,5
Harga produk	1,5
ketahanan material	1,5
Desain siluet	1,5
Desain logo	1,5
Perlu promosi produk	1,5
Bagian kaki depan	1,5
Penggunaan dengan kaos kaki	1,5
Hak cipta	1,5

4.7.6 Menentukan Skala Kepentingan Konsumen

Pada proses ini akan diketahui tingkat kepentingan relatif masing-masing fitur sekaligus menunjukkan prioritas fitur yang membutuhkan perbaikan. Hasil perhitungan skala kepentingan konsumen (skk) seperti pada tabel 4.19.

Tabel 4. 19 Hasil Skala Kepentingan Konsumen

Fitur	*IR*	RB	*SP*	SKK	NKK	Prioritas
Desain produk	0,9273	1,0940919	1,5	1,52182713	3,20497257	11
Tingkat orisinalitas produk	0,891	1,56739812	1,5	2,09482759	4,41171326	9
Kemiripan dengan produk yang ada di pasaran	0,6637	0,02808989	1,5	0,02796489	0,05889414	19
Ketersediaan ukuran perlu ditingkatkan	0,8682	0,01054852	1,5	0,01373734	0,02893088	22
Merubah *sizing*	0,7137	0,06289308	1,5	0,06733019	0,14179758	16
Membuat ukuran besar	0,6728	0,01262626	1,5	0,01274242	0,02683558	23
Meniru *Vans*	0,45	0,01052632	1,5	0,00710526	0,01496371	24
Mirip *Converse*	0,491	0,05050505	1,5	0,03719697	0,07833693	18
Bangga produk lokal	0,8364	0,34578147	1,5	0,43381743	0,91362082	12
Produk Indonesia	0,8546	1,9379845	1,5	2,48430233	5,23194825	8
Model Produk	0,8273	1,40056022	1,5	1,73802521	3,66028638	10
Lebih baik dari Compass	0,5864	0,34482759	1,5	0,30331034	0,63877251	14
Ketersediaan warna perlu ditingkatkan	0,8591	0,02538071	1,5	0,03270685	0,06888073	13
Ketersediaan produk perlu ditingkatkan	0,8637	0,02028398	1,5	0,0262789	0,05534345	20
Penjualan perlu ditingkatkan	0,791	0,05076142	1,5	0,06022843	0,12684125	17
Membentuk identitas produk	0,8819	0,01092896	1,5	0,01445738	0,03044728	21
Harga produk	0,9046	2,53807107	1,5	3,44390863	7,25288204	7
ketahanan material	0,9637	2,53807107	1,5	3,66890863	7,72673273	6
Desain siluet	0,7864	5,05050505	1,5	5,95757576	12,5466727	4
Desain logo	0,7319	5,05050505	1,5	5,54469697	11,6771487	5
Perlu promosi produk	0,8046	0,1010101	1,5	0,12190909	0,25674092	15
Bagian kaki depan	0,9046	5,05050505	1,5	6,8530303	14,4325026	1
Penggunaan dengan kaos kaki	0,8364	5,05050505	1,5	6,33636364	13,3444011	3
Hak cipta	0,8819	5,05050505	1,5	6,68106061	14,0703339	2

4.7.7 Menentukan Daftar Kebutuhan Teknis

Dalam memenuhi kepentingan/kebutuhan konsumen, perlu dirancang target dan respon secara teknis sesuai dengan kemampuan perusahaan. Tabel 4.20 berisi target dan tabel 4.21 berisi respon teknis atas target secara berurutan.

Tabel 4. 20 Target Perbaikan Produk

No	Target
1	Lebih menonjolkan kesan klasik dan unik
2	Mengurangi adopsi desain
3	Persediaan ukuran di semua wilayah merata
4	*Size chart* mampu diterima dan sesuai dengan kondisi konsumen
5	Informasi tentang *size chart* yang ada dipahami konsumen secara optimal
6	Terakomodasinya konsumen dengan ukuran kaki 45-50 cm
7	Memiliki komponen dekoratif dengan fungsi yang kuat dan estetik yang disukai konsumen
8	Harga kompetitif
9	Kerapihan produk terkendali
10	Menghasilkan produk dengan desain variatif dan mampu mengokohkan keberadaan brand
11	Konsisten menghadirkan produk sebagai pelengkap penampilan
12	Memiliki ciri khas yang tidak terikat waktu dan fleksibel
13	Insole amblas dalam waktu satu bulan
14	Memberikan informasi keberadaan *brand* dengan mudah
15	Brand dikenal masyarakat luas
16	Kenyamanan saat pemakaian
17	Menjadi *brand* yang membawa inspirasi bagi *brand* lain

Tabel 4. 21 Respon Teknis/Kebutuhan Teknis

No	Respon Teknis
1	Aksen jahitan dibuat lebih minimalis
2	Optimasi parameter desain yaitu bentuk, dimensi, dan material
3	Klusterisasi wilayah berdasarkan total penjualan dan permintaan
4	Menggunakan standar ukuran sistem standar dari Prancis
5	Aktif memberi informasi terkait size produk pada *website*, akun media sosial, *packaging* produk dan toko
6	Membuat produk dengan ukuran nomor sepatu dalam centimeter 45-50cm dengan jumlah yang sesuai permintaan
7	Melakukan uji paramater secara berkala untuk memilih parameter mana yang paling berpengaruh signifikan terhadap luaran desain yang diinginkan
8	Menjaga kestabilan harga bahan baku dan terus melakukan perbaikan proses produksi untuk mengurangi biaya
9	Membuat dan melaksanakan prosedur produksi dan *QC* dengan optimal
10	Mengeluarkan banyak produk kolaborasi dengan tujuan eksplorasi desain dan riset kecenderungan desain yang diminati konsumen
11	Membangun sistem dan model pengembangan produk yang *update*
12	Menetapkan tema atau konsep produk secara mendalam dan aktif berkomunikasi dengan konsumen melalui media yang ada

No	Respon Teknis
13	Perbaikan secara berkelanjutan dengan menerapkan *hybrid* metode atau pendekatan yang mutakhir
14	Logo dibuat secara *emboss*
15	Memperbaiki konten media sosial dan website dan melakukan kolaborasi dengan *infuencer* serta *content creator*
16	Mengutamakan aspek ergonomis
17	Memiliki hak cipta yang terdaftar dengan desain orisinil

4.7.8 Analisa Hubungan Antara Kebutuhan Konsumen dengan Kebutuhan Teknis dan Kesulitan Teknis

Analisa hubungan antara kebutuhan konsumen dengan kebutuhan teknis dilakukan untuk mengetahui seberapa kuat pengaruh atribut produk yang didesain perusahaan mampu memberikan kepuasan atau kontribusi terdahap atribut produk yang diinginkan konsumen. Bernilai 9 apabila respon berpengaruh kuat dalam memberikan kepuasan konsumen, bernilai 3 apabila respon teknis memberikan pengaruh sedang pada kepuasan konsumen, dan bernilai 1 apabila dianggap lemah memberikan kepuasan kepada konsumen. Sebagai contoh, respon aksen jahitan dibuat minimalis merupakan langkah yang dianggap berpengaruh kuat pada desain produk. Kemudian untuk persiapan pihak perusahaan, dapat dianalisa kesulitan teknis tersebut. Hasil dari proses ini dapat dilihat pada gambar 4.44.

4.7.9 Menghitung Skor Kebutuhan Teknis

Skor kebutuhan teknis akan menunjukkan kebutuhan teknis yang perlu segera dilakukan untuk menjawab kepentingan konsumen. Perhitungan ini juga dinormalkan untuk mengetahui prioritas kegiatan. Makin tinggi nilai kebutuhan teknis maka semakin perlu untuk dilaksanakan. Hasil perhitungan skor kebutuhan teknis (SKB), dapat dilihat pada tabel 4.22.

Tabel 4. 22 Skor Kebutuhan Teknis

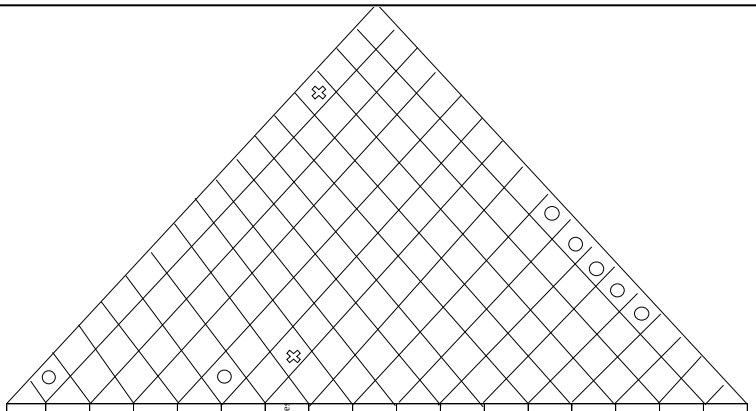
No Kebutuhan Teknis	SKB	NKB
1	32,7339	3,542328
2	93,2792	10,09429
3	43,3254	4,688496
4	19,3966	2,09902
5	23,4334	2,535866
6	26,9609	2,917598
7	111,3983	12,05507
8	38,2432	4,138522
9	62,2081	6,731904
10	81,2193	8,789218
11	112,0302	12,12345
12	81,933	8,866452
13	49,8485	5,394399
14	6,9142	0,748226
15	27,6156	2,988446
16	43,3345	4,689481
17	70,2045	7,597242

4.7.10 Menentukan Korelasi Antar Kebutuhan Teknis

Pada tahap ini akan dianalisa untuk melihat pola hubungan antar respon teknis, apakah terdapat respon teknis yang telah dirancang memberikan sinergis atau justru saling menghambat. Hasil penentuan korelasi antar respon teknis dapat dilihat pada gambar 4.44.

4.7.11 Menyusun Semua Data Perhitungan dan Analisa Kedalam Matriks *House of Quality*

Setelah melalui beberapa proses, data dan perhitungan akan disusun dalam matriks *House of Quality(HoQ)* mengikuti kaidah dan ketentuan yang telah ditetapkan. Hasil *HoQ* seperti pada gambar 4.44.



A : Meningkatkan kualitas produk
B : Mempertahankan kualitas produk dan melakukan inovasi produk secara kontinyu
C : Mempertahankan kualitas produk

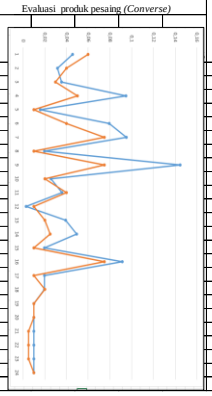
Kebutuhan
:produk ventela
:produk pesaing

: korelasi positif
: korelasi negatif

Relationship Key
● : kuat (9)
○ : sedang (3)
△ : lemah (1)

Sales point
● : 1.5
◉ : 1.2
○ : 1

No	Fitur	*Important Rating*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	*Goals*	*Sales point*	*Improvement Ratio*	Bobot baris	Bobot Baris (%)	Tindakan
1	Desain produk	0,9273	●	●				△	●	△	○	●	●	●	○	△		●	●	0,05	1,5	1,0940919	1,521827133	3,204972569	B
2	Tingkat orisinalitas produk	0,891	○	●				△	●	△	△	●	●	●	○	○		○	●	0,05	1,5	1,5673981	2,094827586	4,411713264	B
3	Kemiripan dengan produk yang ada di pasaran	0,6637	○	●					●	△	△	●	●	●	○	△		△	●	0,001	1,5	0,0280899	0,027964888	0,058894138	A
4	Ketersediaan ukuran perlu ditingkatkan	0,8682		△	●	●	●	●	○	△	○	△	○	△	●			○		0,001	1,5	0,0105485	1,37E-02	0,028930884	A
5	Merubah *sizing*	0,7137				●	●	●	△	△	△		○	△	○			●		0,001	1,5	0,0628931	0,067330189	0,141797582	A
6	Membuat ukuran besar	0,6728		△	●	○	●	●	△											0,001	1,5	0,0126263	1,27E-02	0,026835584	A
7	Meniru *Vans*	0,45	○	●					●	△	○	●	●	●	○	△			●	0,001	1,5	0,0105263	7,11E-03	0,014963706	A
8	Mirip *Converse*	0,491	○	●					●	△	○	●	●	●	○	△				0,001	1,5	0,0505051	3,72E-02	0,078336931	A
9	Bangga produk lokal	0,8364		●	○	△	△	△	●	●	●	●	●	●	○		●		●	0,05	1,5	0,3457815	0,433817427	0,913620821	A
10	Produk Indonesia	0,8546		●	○	△	△	△	●	●	●	●	●	●	○		●		●	0,05	1,5	1,9379845	2,484302326	5,231948249	B
11	Model Produk	0,8273	●	●				△	●	△	○	●	●	●	○	△		●	●	0,05	1,5	1,4005602	1,73802521	3,660286375	B
12	Lebih baik dari Compass	0,5864		●	○	△	△	△	●	●	●	●	●	●	○		●		●	0,001	1,5	0,3448276	0,303310345	0,638772508	B
13	Ketersediaan warna perlu ditingkatkan	0,8591			●						△				△					0,001	1,5	0,0253807	3,27E-02	0,068880731	A
14	Ketersediaan produk perlu ditingkatkan	0,8637			●	△	△	△					○		△		●			0,001	1,5	0,020284	2,63E-02	0,055343453	A
15	Penjualan perlu ditingkatkan	0,791			●															0,001	1,5	0,0507614	6,02E-02	0,126841249	A
16	Membentuk identitas produk	0,8819	●	●				△	●	△	○	●	●	●	○	△		●	●	0,001	1,5	0,010929	0,014457377	0,03044728	A
17	Harga produk	0,9046		○					○	●	△				○					0,05	1,5	2,5380711	3,44E+00	7,252882041	A
18	ketahanan material	0,9637		●					●		●					●				0,05	1,5	2,5380711	3,668908629	7,726732725	A
19	Desain siluet	0,7864	△	●					●			●	●	●	○					0,05	1,5	5,0505051	5,957575758	12,54667265	A
20	Desain logo	0,7319	△	●					●			●	●	●	○					0,05	1,5	5,0505051	5,54469697	11,67714867	A
21	Perlu promosi produk	0,8046																●		0,001	1,5	0,1010101	1,22E-01	0,256740916	A
22	Bagian kaki depan	0,9046							●		●		●							0,05	1,5	5,0505051	6,853030303	14,43250265	A
23	Penggunaan dengan kaos kaki	0,8364							●		●		●							0,05	1,5	5,0505051	6,336363636	13,34440108	A
24	Hak cipta	0,8819																	●	0,05	1,5	5,0505051	6,681060606	1,85465783	A
Target			Lebih menonjolkan kesan klasik dan unik	Menguangi adopsi desain	Persediaan ukuran di semua wilayah merata	Size chart mampu diterima dan sesuai dengan kondisi konsumen	Informasi tentang*size chart* yang ada dipahami konsumen secara optimal	Terakomodasinya konsumen dengan ukuran kaki 45-50 cm	Memiliki komponen dekoratif dengan fungsi yang kuat dan estetik yang disukai konsumen	Harga kompetitif	Ketepatan produk terkendali	Menghasilkan produk dengan desain variatif dan mampu mengakomodasi keberadaan*brand*	Konsisten menghadirkan produk sebagai pelengkap penampilan	Memiliki ciri khas yang tidak terikat waktu dan fleksibel	Insole ambles dalam waktu satu bulan	Memberikan informasi keberadaan*brand* dengan mudah	*brand* dikenal masyarakat luas	Kenyamanan saat pemakaian	Menjadi*brand* yang membawa inspirasi bagi*brand* lain						
Tingkat Kesulitan Teknis			1	1	3	1	1	3	2	1	1	2	2	2	1	1	1	1	2						
Skor Kebutuhan Teknis			32,7339	93,2792	43,3254	19,3966	23,4334	26,9609	111,3983	38,2432	62,2081	81,2193	112,0302	81,933	49,8485	6,9142	27,6156	43,3345	70,2045						
Skor Kebutuhan Teknis (%)			3,54233	10,09429	4,688496	2,09902	2,535866	2,917598	12,05507	4,138522	6,731904	8,789218	12,12345	8,866452	5,394399	0,748226	2,988446	4,689481	7,597242						



Gambar 4. 44 *House of Quality* Produk Ventela Public Low

BAB V

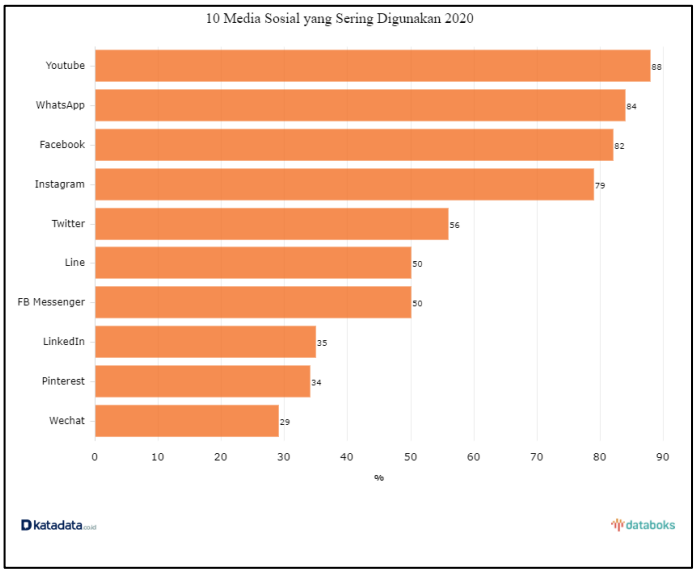
HASIL DAN PEMBAHASAN

Pada Bab ini akan berisi perbandingan terhadap setiap dokumen dan percobaan yang telah dilakukan dan kemudian dipilih model yang paling baik untuk diimplementasikan ke tahap pengembangan produk melalui *QFD*. Selain itu juga akan dibahas secara rinci hasil dari integrasi model dengan *QFD* untuk mengembangkan produk *Ventela Pulic Low*.

5.1 Pengumpulan Data

Pada proses pengumpulan data, media sosial yang digunakan pada penelitian ini yaitu Twitter dan Youtube, dan *marketplace* yang digunakan adalah Shopee. Kuesioner dengan responden merupakan pengguna sepatu Ventela sebanyak 55 orang dan informasi dari beberapa sumber akan digunakan sebagai pertimbangan. Hasil kuesioner secara lengkap dapat dilihat pada lampiran. Berikut merupakan alasan terpilihnya Twitter, Youtube, dan Shopee sebagai *platform* yang akan diambil datanya untuk membangun model.

1. Twitter dan Instagram termasuk kedalam 10 media sosial yang paling sering digunakan di Indonesia. Twitter menempati peringkat 5 dengan presentase 56% digunakan oleh masyarakat Indonesia dan Instagram menempati peringkat 4 dengan presentase digunakan sebesar 79% (Jayani & Fitra, 2020 (a))

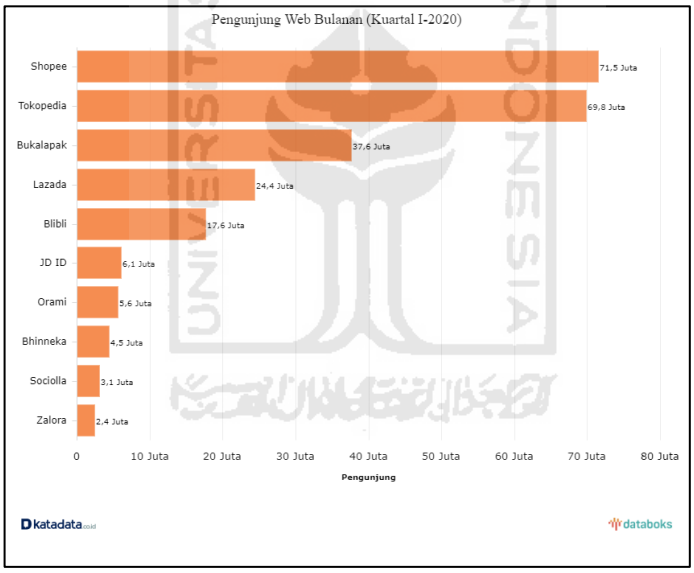


Gambar 5. 1 Jumlah Pengguna Media Sosial di Indonesia
(Sumber: Jayani & Fitra, 2020 (a))

2. Beberapa media sosial seperti WhatApp, FB Messenger, dan Wechat merupakan media sosial yang bersifat semi terbuka, yang mana konten yang dibagikan dapat dilihat hanya kepada beberapa orang tertentu yaitu dalam lingkup kontak yang disimpan. Sedangkan Youtube, Facebook, Instagram, Twitter, LinkedIn, Line dan Pinterest merupakan *platform* media sosial yang berbasis *sharing* secara terbuka, yaitu konten dapat dilihat oleh semua pengguna media sosial tersebut. Dari segi kemudahan pengambilan data, hal ini merupakan keunggulan dari media sosial yang berbasis sharing secara terbuka. Pada media sosial Facebook dan Instagram terdapat batasan dalam mengambil data secara otomatis *(scraping)* seperti yang disebutkan pada peraturan penggunaan oleh instagram nomor 10 (Instagram, 2013) dan peraturan pengambilan data otomatis oleh facebook (Facebook, 2010).
3. Hasil dari kuesioner yang dilakukan, media sosial secara berurutan yang paling sering digunakan untuk menuliskan atau menuangkan *review* yaitu Twitter dan Instagram bernilai sama, WhatsApp, Youtube, Facebook, Line dan Pinterest bernilai sama, Tiktok, Tumblr dan Blog bernilai sama. Beberapa alasan responden memilih Instagram dan Twitter yaitu dari segi mudah digunakan, tampilan media sosial yaitu Instagram unggul pada visual dan Twitter fokus pada teks, bebas dalam menyampaikan ulasan, dan beragamnya konten yang tersedia.
4. Pada media sosial secara berurutan yang paling sering digunakan untuk melihat atau mengumpulkan *review* yaitu Instagram, Youtube, Twitter, Tiktok, Blog, WhatsApp,

dan Facebook. Alasan dalam memilih media sosial tersebut yaitu kemudahan dalam mengakses, konten yang beragam, ulasan disampaikan secara detail dan media sosial tersebut banyak digunakan oleh *influencer*.

5. Berdasar hal tersebut maka akan dipilih Twitter dan Youtube dengan pertimbangan kemudahan pengambilan data dan termasuk media sosial yang populer digunakan konsumen. Sesuai jumlah pengguna Twitter dan Youtube secara umum dengan populernya media sosial tersebut digunakan konsumen, maka data yang diambil dari kedua media sosial dianggap mampu mewakili data ulasan produk Ventela Public Low dan Converse yang ada.
6. Dari hasil kuesioner *marketplace* yang paling sering digunakan konsumen untuk berbelanja produk ventela, yaitu 60% menggunakan shopee, 21,8% menggunakan *website reseller*, 16,4% menggunakan tokopedia, dan 1,8% menggunakan lazada. Dengan penggunaan umum *marketplace* seperti dapat dilihat pada gambar 5.2.



Gambar 5. 2 Peringkat Pengguna *Marketplace* di Indonesia

(Sumber: (Jayani & Fitra, 2020 (b))

7. Alasan konsumen sesuai hasil kuesioner menggunakan *marketplace* tersebut yaitu, dari segi promosi yang ditawarkan menarik, mudah digunakan, dan mudah ditemui penjual yang terpercaya. Berdasar beberapa keunggulan dan jumlah pengguna, maka pada penelitian ini akan menggunakan data *marketplace* Shopee.

5.2 Analisa dan Evaluasi Eksperimen Model

Sesuai dengan algoritma dan penyetelan parameter seperti yang dapat dilihat pada bab iv, perbandingan hasil performasi adalah seperti pada tabel berikut 5.1.

Tabel 5. 1 Klasifikasi Tanpa Normalisasi

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	67	61,03	63	62,71	63	59,77
2	*tfidf*	70	64,32	62	59,32	63,15	60,9

Pada percobaan pertama, yaitu tidak menerapkan proses normalisasi pada *pre-processing* dan menggunakan dua klasifikasi dengan dua metode dalam proses *vectorization* yaitu *BoW* dan *TFIDF*. Berdasar tabel 5.1 dapat dilihat bahwa performansi terbesar didapatkan sebesar 70 yang didapatkan dokumen A dengan klasifikasi niave dan penggunaan *tfidf*. Sedangkan performansi terkecil sebesar 59,32 yang didapatkan dokumen B dengan klasifikasi *svm* dan penggunaan *tfidf*.

Tabel 5. 2 Klasifikasi Dengan Penambahan Normalisasi

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	70	65,26	69	68,22	67	62,03
2	*tfidf*	68,07	65,26	66,95	65,25	66,17	65,41
3	*tfidf L1*	68,54	65,73	66,95	68,22	66,54	61,65
4	*3+feature MI*	68,07	61,97	61,86	63,98	61,27	58,27

Pada percobaan kedua, yaitu menerapkan proses normalisasi pada *pre-processing* dan menggunakan dua klasifikasi dengan empat metode dalam proses *vectorization* yaitu *BoW*, *TFIDF*, *TFIDF L1*, dan *MI*. Sebelum melihat perbandingan antara penggunaan *TFIDF L1* dan *MI*, berdasar tabel 5.2 performansi terbesar didapatkan sebesar 70 yang didapatkan dokumen A dengan klasifikasi *naive* dan penggunaan *BoW*. Sedangkan performansi terkecil sebesar 62,03 yang didapatkan dokumen C dengan klasifikasi *svm* dan penggunaan *BoW*. Maka dapat disimpulkan bahwa penambahan proses normalisasi pada *pre-processing* dapat meningkatkan performansi pada setiap kombinasi. Penambahan normalisasi pada proses *pre-processing* selanjutnya akan selalu digunakan untuk setiap dokumen. Kemudian percobaan ketiga ditambahkan dengan mencoba dua

metode seleksi fitur yaitu *TFIDF L1* dan *MI*. Pada *TFIDF L1* fitur diseleksi menjadi 1000 fitur untuk kemudian digunakan dalam klasifikasi sentimen, hasil membuktikan bahwa seleksi fitur *TFIDF L1* belum mengalahkan nilai performansi terbesar. Sedangkan pada penggunaan *MI* yang digunakan secara *unigram*, fitur diseleksi menjadi 491 fitur, namun dapat dilihat bahwa hasil performansi menggunakan seleksi fitur *MI* mendapatkan nilai performansi yang lebih rendah dari sebelumnya, dan terdapat nilai performansi terendah yaitu sebesar 58,27.

Tabel 5. 3 Klasifikasi Dengan Normalisasi dan *Hyperparameter Turning*

		Dokumen A			Dokumen B			Dokumen C		
		naïve	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*
1	*bow*	65,26	64	63	68,22	**71**	67	67.67	63	60
2	*tfidf*		65	65		67	65		66	65
3	*tfidf L1*	-	65	63	-	67	68	-	65	65
4	*3+feature MI*	-	59	58	-	68	65	-	60	58

Pada percobaan keempat akan digunakan *hyperparameter turning* hasil seperti dapat dilihat pada tabel 5.3. Pada proses ini terdapat beberapa kombinasi yang meningkat performansinya dan terdapat kombinasi yang nilai performansi lebih rendah dari percobaan kedua maupun ketiga. Dokumen B dengan klasifikasi *svm* secara *kernel rbf* dan penggunaan *BoW* mendapatkan nilai performansi tertinggi dari percobaan lainnya yaitu sebesar 71.

Dengan kombinasi algoritma dan parameter yang sama tambahan percobaan dilakukan dengan mengubah variasi *split* data menjadi 70 data training dan 30 data *testing*. Namun hasil performansi tidak dapat melebihi 71, seperti dapat dilihat pada tabel 5.4, 5.5, dan 5.6. Kemudian percobaan terakhir melakukan *split* secara bertingkat *(stratified)*, dimana pada fungsi tersebut data akan terbagi sesuai proporsi awal (dalam hal ini adalah sama rata antara kelas positif, *negative*, dan netral) untuk 80 data *training* dan 20 *testing* data, pada percobaan ini nilai tertinggi yang diperoleh sebesar 70, yang berarti belum melebihi performansi sebelumnya yaitu 71. Hasil klasifikasi dengan *stratified* dapat dilihat pada tabel 5.7, 5.8, dan 5.9.

Tabel 5. 4 Hasil Percobaan Pertama dengan *Split* 70% dan 30%

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	65	62,69	59	60,9	63	60,3
2	*tfidf*	63	66,77	61,47	57,79	62,56	59,54

Tabel 5. 5 Hasil Percobaan Kedua dengan *Split* 70% dan 30%

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	67	64,89	63	62,6	66	64,57
2	*tfidf*	67,39	65,83	63,17	64,02	64,57	64,07
3	*tfidf L1*	68	63,32	64	63,17	67	63,06
4	*3+feature MI*	60	55,48	61	61,8	63,53	63,15

Tabel 5. 6 Hasil Percobaan Ketiga dan Keempat dengan *Split* 70% dan 30%

		Dokumen A			Dokumen B			Dokumen C		
		naïve	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*
1	*bow*	66,46	65	62	66,29	66	62	66,08	65	63
2	*tfidf*		69	66		64	64		66	64
3	*tfidf L1*		66	63		63	65		64	61
4	*3+feature MI*		57	55		63	62		61	62

Tabel 5. 7 Hasil Percobaan Pertama dengan *Split Stratified*

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	67	61,03	63	62,71	63	59,77
2	*tfidf*	70	64,31	61,86	59,32	63,15	60,9

Tabel 5. 8 Hasil Percobaan Kedua dengan *Split Stratified*

		Dokumen A		Dokumen B		Dokumen C	
		naïve	*svm*	*naïve*	*svm*	*naïve*	*svm*
1	*bow*	70	62,91	63	63,55	62	59,77
2	*tfidf*	68,07	65,25	65,67	62,71	63,9	64,28
3	*tfidf L1*	68,07	66,19	63,56	64,4	64,66	65,04
4	*3+feature MI*	65,73	64,79	61,44	59,74	59,02	60,53

Tabel 5. 9 Hasil Percobaan Ketiga dan Keempat dengan *Split Stratified*

		Dokumen A			Dokumen B			Dokumen C		
		naïve	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*	*naïve*	*svm rbf*	*svm linear*
1	*bow*	70,89	62	67	66,1	64	64	62,03	59	59
2	*tfidf*		65	65		62	63		63	64
3	*tfidf L1*		66	66		61	64		62	61
4	*3+feature MI*		63	65		61	60		60	61

Confusion matrix akan menggambarkan hasil akurasi prediksi positif yang benar, prediksi positif yang salah, prediksi negatif yang benar, prediksi negatif yang salah, prediksi netral yang benar, dan prediksi netral yang salah. Perhitungan akurasi berdasarkan seluruh prediksi yang benar dibandingkan dengan seluruh data *testing*. Semakin tinggi nilai akurasi, semakin baik pula model yang dihasilkan. Penilaian performansi diukur berdasarkan nilai performansi antara *precision*, *recall*, dan *f-measure*. Suatu model dikatakan memiliki nilai yang baik apabila *precision* dan *recall* sama-sama memiliki nilai yang tinggi, hal tersebut menunjukkan bahwa model terhindar dari bias (Powers, 2007). Dapat dilihat pada gambar 5.3. Pada kelas *negative* memiliki nilai *precision*, *recall*, dan *f1 score* yang tidak terlalu berbeda, begitu pula pada kelas netral dan positif. Pada hasil *macro avg* dan *weighted avg* juga menunjukkan nilai yang baik dan seragam.

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.64      0.58      0.61        77
      netral       0.56      0.68      0.62        66
     positif       0.91      0.84      0.87        93

    accuracy                           0.71       236
   macro avg       0.70      0.70      0.70       236
weighted avg       0.72      0.71      0.72       236
```

Gambar 5. 3 *Confusion Matrix* Model Terpilih

Seperti yang telah diketahui sebelumnya model terpilih merupakan algoritma yang telah melalui proses *hyperparameter* dengan klasifikasi *SVM*. Gambar 5.4 berikut merupakan parameter lengkap yang diterapkan oleh proses *hyperparameter* pada data.

```
{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
SVC(C=10, break_ties=False, cache_size=200, class_weight=None, co
ef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.01, kernel
='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=T
rue, tol=0.001,
    verbose=False)
```

Gambar 5. 4 Parameter *Hyperparameter* yang Menghasilkan Model Terpilih

5.3 Analisa Model Terpilih

Sesuai dengan kriteria yang didapatkan yaitu nilai performansi tertinggi, hasil rata-rata *10-fold cross validation* yang menunjukkan 0,850463, dan hasil *ROC* yang menunjukkan 0,858485 dan 0,866732. Maka Dokumen B dengan performansi 71 merupakan model yang akan dipilih. Sebelumnya beberapa percobaan telah dilakukan untuk mengoptimalkan performansi dari *dataset* yang ada. Beberapa analisa atas didapatkan performansi tertinggi sebesar 71 adalah sebagai berikut:

1. Hasil *scraping* Twitter didapatkan 996 *tweet* dan hasil *scraping* Youtube didapatkan 9412 komentar. Total data mentah yang ada adalah sebanyak 10408, dari data mentah yang ada *review* yang dapat digunakan hanya sebanyak 1673 *review*, atau yang berarti 16,074% dari data mentah. Alasan tidak dapat digunakannya data yaitu (1) pada *twitter* data dengan *keyword* ventela terdapat juga *tweet* Bahasa asing dengan arti bukan merupakan merek sepatu, (2) terdapat beberapa *tweet* hanya berupa balasan *tweet*, tidak membicarakan tentang ventela, (3) pada *scraping* Shopee, beberapa data tidak mengulas sepatu ventela melainkan mengulas performa toko atau performa kurir pengiriman, (4) pada *youtube* terdapat komentar yang hanya berupa komentar, tidak membicarakan tentang ventela, dan (4) terdapat pula beberapa komentar tidak mengulas sepatu ventela tetapi mengulas atribut video, pembawaan pembicara, dan menawarkan produk.
2. Dari hasil literasi yang telah dilakukan dan menurut Putra (2019), distribusi pada masing-masing *dataset* sebaiknya seimbang *(balanced)* dan memuat seluruh kasus. Misal, sebuah *dataset binary classification* sebaiknya memuat 50% kasus positif dan 50% kasus negatif. Jumlah data sebanyak 1673 (positif sebanyak 746 data, negative sebanyak 476 data, dan netral sebanyak 451 data) *review* tersebut, sebelumnya telah

mengalami penambahan bertahap sampai akhirnya mendapat nilai tersebut. Terdapat tiga dokumen yaitu A, B, dan C dengan jumlah *dataset* yang berbeda dan masing-masing memiliki proporsi kelas positif, netral dan *negative* yang sama. Ketiganya kemudian dilakukan proses yang sama untuk mengetahui kecenderungan performansi model terhadap banyaknya data. Sesuai hasil yang didapatkan, yang dapat dilihat pada tabel 5.10. Rata-rata performansi optimal pada dokumen B dan menurun pada dokumen C. Maka tidak dilakukan penambahan data lagi dan beranjak ke tahap optimalisasi performansi model selanjutnya yaitu pada tahap *pre-processing*.

Tabel 5. 10 Hasil Rata-Rata Performansi Tiap Percobaan

Rata-rata	Dokumen A	Dokumen B	Dokumen C
Tanpa normalisasi	66	62	61,705
Dengan normalisasi	66,6125	66,303	63,5425
Normalisasi+*hyperparameter*	63,0288	67,357	62,75
Keseluruhan	65	65	63

3. Pada *pre-processing* selain proses penambahan *stopword* dan normalisasi yang dapat menaikkan nilai performansi, terdapat proses yang dapat mengurangi *noise* dalam proses klasifikasi sehingga dapat meningkatkan performansi yaitu proses stemming. Namun pada *library* popular seperti *nltk* dan *spacy* belum tersedia dalam Bahasa Indonesia, hal ini mengakibatkan beberapa kata menjadi tidak sesuai ketika dirubah menjadi kata dasarnya.
4. Proses *vectorization* dengan *TFIDF* dan *BoW* tidak mempertimbangkan kata majemuk dan susunan kata.
5. Berdasarkan *information theory* atau *entropy*, seleksi fitur dilakukan pada fitur yang dianggap tidak memiliki nilai informasi yang tinggi. Dengan kata lain, fitur tersebut tidak dapat membedakan satu kelas dan lain bersifat tidak informatif. Namun penggunaan seleksi fitur popular yaitu *MI* dan *TFIDF L1* tidak cocok dengan *dataset*, hal ini dikarenakan karena ada beberapa fitur yang dianggap tidak diinginkan pada proses *MI* maupun *TFIDF L1*, namun sebenarnya dibutuhkan pada proses klasifikasi.
6. Banyak teknik data mining atau *machine learning* yang dikembangkan dengan asumsi kelinieran, sehingga algoritma yang dihasilkan terbatas untuk kasus-kasus yang linier (Santosa, 2007). *SVM* dapat bekerja pada data non-linier dengan menggunakan pendekatan *kernel* pada fitur data awal himpunan data. Fungsi kernel yang digunakan untuk memetakan dimensi awal (dimensi yang lebih rendah) himpunan data ke

dimensi baru (dimensi yang relatif lebih tinggi). Fungsi *kernel* mana yang harus digunakan untuk subtitusi *dot product* di *feature space* sangat tergantung pada data karena fungsi *kernel* ini akan menentukan fitur baru di mana *hyperplane* akan dicari. Pada percobaan pertama, kedua, dan ketiga *kernel* yang digunakan adalah linear karena saat digunakan *kernel* rbf dengan parameter *default* yang digunakan, performansi akan menurun. Namun, dengan kombinasi parameter pada hyperparameter, kernel rbf memunculkan nilai performansi tertinggi.

7. Model merupakan *multi-label classification*, klasifikasi yang bersifat *soft* yaitu dapat mengklasifikasikan suatu term ke dalam beberapa kelas, misal term x memiliki 70% probabilitas sebagai kelas A dan 30% sebagai kelas B. Pada data dijumpai beberapa kata tersebut tidak spesifik dalam satu kelas, sehingga dapat ditemukan dalam lebih dari satu kelas dan hal ini mempengaruhi algoritma dalam memprediksi data. Beberapa kata dapat dilihat seperti pada tabel 5.11.

Tabel 5. 11 *Top Words* Masing-Maing Kelas

Positif		Netral		Negatif	
Kata	Frekuensi	Kata	Frekuensi	Kata	Frekuensi
sangat	264	tidak	214	buat	142
bagus	234	beli	186	tidak	137
mantap	144	buat	141	merek	114
lokal	129	converse	130	converse	105
pakai	129	pakai	122	ukur	99
tidak	121	lebih	98	lokal	93
empuk	119	ukur	90	desain	90
beli	110	sangat	88	vans	88
keren	109	merek	84	pakai	67
buat	104	mirip	81	sangat	64

5.4 Prediksi dan Asosiasi Kata

Pada 1673 data *review* dengan proses pelabelan secara manual, terbagi menjadi kelas positif sebanyak 746 data, *negative* sebanyak 476, dan netral sebanyak 451 data. Sedangkan melalui proses prediksi dengan model terpilih, didapatkan hasil kelas positif sebanyak 602 data, kelas *negative* sebanyak 547 data, dan netral sebanyak 524 data. Melalui proses evaluasi, terdapat 475 data hasil prediksi yang tidak sesuai dengan hasil

pelabelan secara manual, yang berarti akurasi dari proses prediksi dengan 1673 data *review* yaitu sebesar 71,60789.

Hasil asosiasi produk Ventela menggunakan metode terpilih yaitu *t-test* memiliki nilai teratas sebesar 2,22860361 dengan term (buat, desain, sendiri) dan nilai terendah sebesar 0,906567235 dengan term (buat, merek, tidak). Perbedaan antar term teratas dan terendah sebesar 1,322036. Pada hasil asosiasi produk Converse, dari jumlah *review* total sebanyak 14346 dari *scraping* twitter dengan rentang waktu 01 Juli 2010-01 Juli 2020, jumlah data yang layak yang akan digunakan sebagai pembanding sebanyak 469 data review. Berbeda dengan produk Ventela yang hanya diambil kelas *negative* sebagai *input QFD*, pada produk converse 469 data akan digunakan semua dan menghasilkan 3234 term *trigram*. Pada produk Converse metode asosiasi terbaik pada peringkat 1-100 adalah metode *frequency* yang memiliki nilai teratas sebesar 3 dengan term (lebih, beli, vans) dan (makin, buluk, makin). Sedangkan nilai terendah yaitu 1 dan dimiliki oleh banyak *term*, menandakan bahwa data sangat bervariatif.

5.5 Analisa *QFD*

Telah disebutkan sebelumnya bahwa, pada tahapan *QFD* di penelitian ini terdapat beberapa data *dummy* (subjektif peneliti), hal tersebut dimaksudkan untuk menyimulasikan dan mengetahui sejauh mana analisis data dengan *machine learning* yang telah dilakukan dapat digunakan pada metode *QFD*. Maka pada tahap *QFD* ini tidak membahas tentang saran dan analisa dari hasil *QFD*, tetapi akan dibahas tentang proses dan potensi *QFD* yang menggunakan data dengan *machine learning* yang telah dilakukan.

5.5.1 Identifikasi *Voice of Customer* (Kebutuhan Konsumen)

QFD secara konvensional mendapatkan data kepentingan konsumen melalui cara wawancara, kuesioner, atau dokumentasi atas pesanan dan saran konsumen. Dalam menggunakan cara tersebut terdapat hal yang harus dilakukan yaitu penentuan sampel dan uji validitas dan realibilitas atas pertanyaan yang diajukan. Dengan penggunaan *machine learning* hal ini dapat diringkas. Data *review* merupakan pendapat jujur dari konsumen dan pada masa sekarang sudah menjadi hal yang sangat umum.

Sebagai *input QFD*, data *review* dapat disajikan dalam kelas yang diinginkan, telah memiliki peringkat sesuai dengan kemunculan dan keterikatan antar kata, dan dalam jumlah yang mencukupi. Setelah memilih metode yang menghasilkan *term* asosiasi paling baik, data akan melalui proses pengelompokkan. Banyaknya data awal yang akan dikelompokkan dapat disesuaikan dengan kemampuan perusahaan, apakah akan mengambil 100 peringkat atas, atau semua data untuk dikelompokkan. Mulai dari pemutusan banyaknya data ini, peran *designer* akan dibutuhkan.

5.5.2 Analisa Tingkat Kinerja Produk dan Pelayanan

Penentuan tingkat kinerja produk secara konvensional dilakukan dengan menggunakan skala *likert*. Sesuai ketentuan, kebutuhan akan terbagi menjadi lima kelompok dari tidak penting (bernilai 1) dan sangat penting (bernilai 5). Pada penelitian ini, tingkat kinerja produk, dinilai sesuai dengan nilai asosiasinya. Bila pada konvensional sangat penting mendapat nilai 5, maka pada data sangat penting akan bernilai 1. Perhitungan *kinerja produk* dengan nilai asosiasi nilai memiliki rentang yang lebih luas.

Berikut merupakan analisis kelemahan dan kelebihan menggunakan perhitungan *kinerja produk* menggunakan *input* analisis data *machine learning*.

1. Pada penentuan *voice of customer* terdapat proses pengelompokkan data yang bertujuan untuk meringkas data dan mendapatkan kebutuhan konsumen yang utuh. Kekurangan dari data *review* adalah masih banyak data mentah yang berupa kata-kata yang kurang mencerminkan sebuah informasi. Dengan konsep asosiasi kata, terdapat pula *term* yang memiliki informasi sama namun memiliki nilai yang sangat berbeda dikarenakan kemunculan *term* yang berbeda. Hal ini akan berpengaruh pada proses pengelompokkan, karena nilai dihitung secara rata-rata maka bila terdapat *term* sama dengan nilai yang berbeda signifikan dapat menurunkan nilai. Maka digunakan, pembobotan berdasarkan jumlah data penyusunnya.
2. Proses pengelompokkan berdasarkan kepada kesamaan kata masih dilakukan secara manual. Peneliti telah melakukan percobaan pengelompokkan secara otomatis melalui algoritma *clustering k-means* dan *topic modelling*. Didapatkan hasil yang tidak sesuai, hal ini karena *k-means* dan *topic modelling* fokus kepada kemunculan

kata dalam dokumen bukan kepada kesamaan kata. Namun, pengelompokkan secara otomatis dengan *input* data asosiasi sangat mungkin dilakukan karena data hasil asosiasi telah spesifik.

3. Metode skala *likert* dengan skala 1 sampai 5 yang biasa digunakan pada metode konvensional merupakan metode yang kurang logis dalam menilai kinerja produk. Dalam beberapa penelitian disebutkan bahwa ketidaklogisan ditunjukkan karena kategori pada nilai tengah masih bersifat bias seperti perbedaan antara sesuai dan lebih sesuai yang kurang signifikan.
4. Setelah melalui proses pengelompokkan, perlu dibentuk fitur yang mewakili data-data penyusunnya dan akan berperan sebagai kebutuhan konsumen. Dalam beberapa data yang memuat informasi tidak utuh, fitur akan sulit muncul maka dalam hal ini perlu melihat reputasi *brand* terkait data-data tersebut. Pencarian data *review* produk pesaing akan berguna untuk menemukan fitur yang sesuai. Data *review* produk pesaing dapat dijadikan pembanding masing-masing data yang ada. Pada penelitian ini, fitur yang didapatkan terbagi menjadi dua jenis yaitu fitur bermakna positif dan fitur bermakna negatif. Pada tahap identifikasi kebutuhan konsumen dengan menggunakan data *review* tetap memerlukan analisis dari pihak *designer* seperti pada halnya menganalisis kebutuhan konsumen dengan cara konvensional. Namun, data *review* memiliki kelebihan dari segi informasi yaitu beragamnya topik, telah memiliki nilai, dan jumlah yang mencukupi.

5.5.3 Penentuan Tingkat Kebutuhan dan Keinginan Relatif *(IR)* Sesuai Kebutuhan Konsumen

Terdapat beberapa fitur yang apabila nilai *IR* dihitung dengan nilai asosiasi terdapat perbedaan peringkat kepentingannya, karena nilai asosiasi merupakan nilai tidak langsung, sedangkan penilaian tingkat kebutuhan diperlukan penilaian secara langsung. Berikut merupakan perbandingan nilai *IR* yang menggunakan nilai asosiasi dengan hasil kuesioner, seperti dapat dilihat pada table 5.12.

Tabel 5. 12 Perbandingan Nilai *IR*

Fitur	*IR (kuesioner)*	*IR (asosiasi)*
Desain produk	0,9273	0,0144
Tingkat orisinalitas produk	0,891	0,0082
Kemiripan dengan produk yang ada di pasaran	0,6637	0,0045
Ketersediaan ukuran perlu ditingkatkan	0,8682	0,0053
Merubah *sizing*	0,7137	0,0042
Membuat ukuran besar	0,6728	0,0009
Meniru *Vans*	0,45	0,0051
Mirip *Converse*	0,491	0,0003
Bangga produk lokal	0,8364	0,0055
Produk Indonesia	0,8546	0,0044
Model Produk	0,8273	0,0044
Lebih baik dari Compass	0,5864	0,0072
Ketersediaan warna perlu ditingkatkan	0,8591	0,0007
Ketersediaan produk perlu ditingkatkan	0,8637	0,0008
Penjualan perlu ditingkatkan	0,791	0,0004
Membentuk identitas produk	0,8819	0,0086
Harga produk	0,9046	0,0004
ketahanan material	0,9637	0,0004
Desain siluet	0,7864	0,0002
Desain logo	0,7319	0,0002
Perlu promosi produk	0,8046	0,0002
Bagian kaki depan	0,9046	0,0002
Penggunaan dengan kaos kaki	0,8364	0,0002
Hak cipta	0,8819	0,0002

5.5.4 Menentukan Rasio Perbaikan *(Improvement Ratio)*

Rasio perbaikan dihitung dari hasil membandingkan *goals* dengan kinerja saat ini. Maka penentuan *goals* atau tingkat perbaikan produk dimasa mendatang perlu dilakukan terlebih dahulu. Penentuan *goals* secara umum dapat dilakukan dengan dua cara yaitu sesuai dengan keinginan konsumen (menentukan dengan nilai maksimal) atau sesuai dengan kebijakan perusahaan berdasarkan kemampuan perusahaan. Pada *QFD* konvensional nilai rasio bernilai ± 1,00 berarti tidak ada perubahan/perbaikan karena kinerja yang ditunjukkan hampir mencapai nilai terbaik, nilai ± 1,20 berarti tingkat perbaikan sedang karena mendekati nilai kinerja terbaik, dan nilai ± 1,50 berarti tingkat perbaikan sulit karena kinerja jauh dari nilai ideal (Sadono et al., 2000). Pada penelitian ini, nilai *goals* disesuaikan dengan masing-masing fitur apabila fitur adalah *positive goals* terbaik adalah bernilai besar, sedangkan pada fitur *negative goals* terbaik adalah bernilai

kecil. Berbeda dengan *QFD* konvensional *goals* memiliki *range* 1-5, pada penelitian ini *goals* memiliki *range* yang yang lebih luas sehingga lebih fleksibel.

5.5.5 Menentukan *Sales Point*

Pada proses ini perlu melibatkan bagian pemasaran, berkaitan dengan berbagai macam atribut yang diinginkan oleh konsumen dan mempertimbangkan rasio perbaikan, bagaimana pengaruhnya bagian pemasaran terhadap perubahan. Kriteria yang dapat digunakan adalah bernilai ± 1,00 bila tidak ada pengaruh terhadap penjualan, bernilai ± 1,20 apabila berpengaruh sedang terhadap penjualan, dan bernilai ± 1,50 apabila memiliki pengaruh kkuat terhadap penjualan.

5.5.6 Menentukan Skala Kepentingan Konsumen

Skala kepentingan konsumen merupakan besaran nominal yang merepresentasikan kinerja kepuasan pelanggan, upaya implementasi perbaikan kinerja perusahaan, dan potensi penjualan. Dengan alasan tersebut SKK memberikan suatu perspektif bisnis strategis secara keseluruhan tentang pentingnya kebutuhan konsumen terhadap kesuksesan perencanaan produk dimasa depan. Nilai SKK didapatkan dengan mengalikan *IR*, rasio perbaikan, dan *sales point*. Semakin besar SKK setiap fitur maka berarti fitur tersebut semakin penting bagi tim pengembangan produk di perusahaan.

5.5.7 Pembuatan Daftar Kebutuhan Teknis dan Analisa Hubungan Antara Kebutuhan Konsumen dengan Kebutuhan Teknis

Menjawab fitur yang dibutuhkan konsumen, diperlukan daftar secara teknis baik kualitatif maupun kuantitatif. Tahap ini biasa dilakukan tim pengembangan produk atau *stakeholder* terkait dengan memperhatikan segala aspek yang ada di perusahaan. Hubungan antara kebutuhan konsumen dengan kebutuhan teknis ditunjukkan dengan *impact symbol* dan memiliki nilai secara kuantitatif. Nilai 1 untuk hubungan yang lemah, nilai 3 untuk hubungan yang sedang dan nilai 9 untuk hubungan yang kuat.

5.5.8 Analisa Kesulitan Teknis

Analisa kesulitan teknis disusun dengan melibatkan bagian produksi. Kesulitan teknis dalam melakukan desain kualitas dapat dikelompokkan menjadi 0 bila termasuk tidak ada masalah, 1 bila kesulitan rendah, 2 bila kesulitan menengah, dan 3 bila sangat sulit.

5.5.9 Hasil Simulasi *QFD*

Dengan data yang ada dan beberapa tambahan data secara *dummy* dengan maksud menyimulasikan integrasi *QFD* dengan analisis data *review machine learning*. Berikut merupakan hasil simulasi *QFD* sebagai gambaran penerapan.

1. Diperoleh 24 fitur dari hasil proses pengelompokkan dan pembentukkan.
2. Berdasar hasil IR melalui kuesioner, fitur yang mendapat peringkat 1 dalam hal dibutuhkan konsumen adalah ketahanan karena memiliki nilai *IR* terbesar, dan meniru Vans menjadi fitur yang mendapat peringkat terakhir dengan *IR* terkecil.
3. Berdasar tingkat kinerja produk, fitur desain produk, tingkat orisinilitas produk, kemiripan dengan produk yang ada di pasaran, ketersediaan ukuran perlu ditingkatkan, merubah *sizing*, membuat ukuran besar, meniru vans, mirip converse, model produk, ketersediaan warna perlu ditingkatkan, ketersediaan produk perlu ditingkatkan, penjualan produk perlu ditingkatkan, membentuk identitas produk, ketahanan material, desain siluet, desain logo, perlu promosi produk, dan hak cipta merupakan fitur yang tingkat kinerja lebih rendah dibandingkan dengan tingkat kinerja produk pesaing.
4. Kebutuhan teknis yang memiliki tingkat kesulitan adalah klusterisasi wilayah berdasarkan total penjualan dan permintaan dan membuat produk dengan ukuran sepatu dalam centimeter 45-50cm dengan jumlah sesuai permintaan.
5. Pada *sales point*, semua atribut memiliki pengaruh besar terhadap penjualan dan kepuasan konsumen, berdasarkan nilai sales *point* sebesar 1,5.
6. Tiga fitur teratas yang paling perlu mendapat perhatian menurut konsumen adalah bagian kaki depan, hak cipta, dan penggunaan dengan kaos kaki.
7. Kebutuhan teknis yang harus segera dilaksanakan untuk menjawab kepentingan konsumen, dua tingkat teratas yaitu membangun *system* dan model pengembangan

produk yang *update* dan meningkatkan sistem, riset, produksi, dan melakukan uji parameter secara berkala untuk memilih parameter mana yang paling berpengaruh signifikan terhadap luaran desain yang diinginkan.

8. Pada tingkat sinergis antar kebutuhan teknis terdapat dua relasi negatif. Pertama hubungan antara dan menerima pesanan khusus *(custom)* untuk produk dengan ukuran besar dan menjaga kestabilan harga bahan baku dan terus melakukan perbaikan untuk mengurangi biaya. Kedua yaitu hubungan aksen jahitan dibuat minimalis dan logo dibuat secara *emboss*.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan penelitian yang dilakukan, diperoleh kesimpulan untuk menjawab rumusan masalah, yaitu sebagai berikut.

Proses pengembangan produk dilakukan dengan langkah awal yaitu perencanaan desain baik dari segi fisik maupun kualitas. Kebutuhan konsumen atau *Voice of Customer* menjadi *input* penting bagi proses pengembangan desain. Perkembangan teknologi dan media saat ini menjadikan data *review online* sebagai sumber data yang menjanjikan. Produk Ventela Public Low merupakan salah satu produk yang terkenal karena banyaknya pendapat/*review* tentang produk yang disampaikan melalui media sosial. Melalui *machine learning* dengan tahap *text processing* dilakukan analisa terhadap data *review* produk Ventela Public Low. Data didapatkan dari media sosial *Twitter, Youtube* dan *marketplace Shopee,* dengan total data 10412. Setelah melalui proses penyaringan yaitu pemilihan data yang merupakan data review diperoleh data sebanyak 1673 data. Analisis sentimen akan digunakan, untuk menemukan kelompok kelas dari masing-masing data. Proses pertama adalah pelabelan secara manual, hal ini dilakukan untuk memberikan identitas awal pada algoritma yang bersifat *supervised learning*.

Algoritma klasifikasi yang digunakan adalah *Naïve bayes* dan *SVM*, sebagai algoritma yang banyak digunakan berdasar kajian literatur yang dilakukan. Algoritma pembobotan kata yang digunakan yaitu *TFIDF* dan *BoW*, bilangan *random* yag digunakan yaitu 77, dan setiap dokumen memiliki jumlah proporsi kelas positif, *negative*, dan netral yang sama. Dalam mengoptimalkan model, dilakukan serangkaian percobaan. Percobaan pertama melakukan penambahan data sampai diperoleh nilai performansi yang stagnan. Hasil didapatkan bahwa performansi mengalami peningkatan pada dokumen B (1176 data) dan mengalami penurunan pada dokumen C (1376 data). Percobaan kedua dilakukan dengan memaksimalkan proses *pre-processing* atau pembersihan data. Hal ini dilakukan dengan penambahan *stopword* dan penggunaan normalisasi yaitu mengubah kata baku menjadi tidak baku, proses ini meningkatkan performansi pada semua model maka akan diterapkan pada model selanjutnya. Percobaan ketiga dengan memilih jumlah data *training* dan *testing* yang tepat *(split), split* dengan perbandingan data *training*:data *testing* sebesar 80:20 memiliki nilai performansi yang lebih baik dari 70:30 dan melakukan *split* secara *stratified* (bertingkat). Percobaan keempat dilakukan dengan menerapkan seleksi fitur menggunakan algoritma *MI* dan *TFIDF L1*. Namun proses seleksi fitur tidak meningkatkan nilai performansi pada semua model. Percobaan terakhir yaitu dengan menggunakan *hyperparameter* baik pada algoritma *Naïve Bayes* maupun *SVM*. Pada percobaan ini menghasilkan model yang memiliki nilai performansi tertinggi yaitu diperoleh sebesar 71. Validasi dengan *10-cross fold validation* dilakukan pada model terpilih untuk mengetahui apakah model memiliki generalisasi yang baik. Hasil *10-cross fold validation* diperoleh nilai rata-rata sebesar 0,850463 atau dalam persen 85,0463%. Selanjutnya mengukur keakuratan model dengen menghitung *ROC AUC*, dengan nilai 0,858485 dan 0,866732 maka model termasuk kedalam *good classification*. Maka model ini akan digunakan pada tahap selanjutnya untuk memproses data *input QFD*. Sebagai *input QFD*, data yang berjumlah 1673 akan diprediksi dengan model yang telah dipilih, dan memperoleh keakuratan dalam memprediksi data sesuai dengan hasil pelabelan manual sebesar 71,60789.

Tahap *QFD* akan menggunakan kelas negatif sebagai fitur akan kebutuhan konsumen. Dalam memperoleh informasi dari data yang telah diprediksi, dilakukan asosiasi kata dengan fungsi *collocations*. Dari lima metode *collocations* yang digunakan *t-test* dipilih karena memiliki nilai *true positive* terbesar. Sebagai pembanding, dilakukan proses

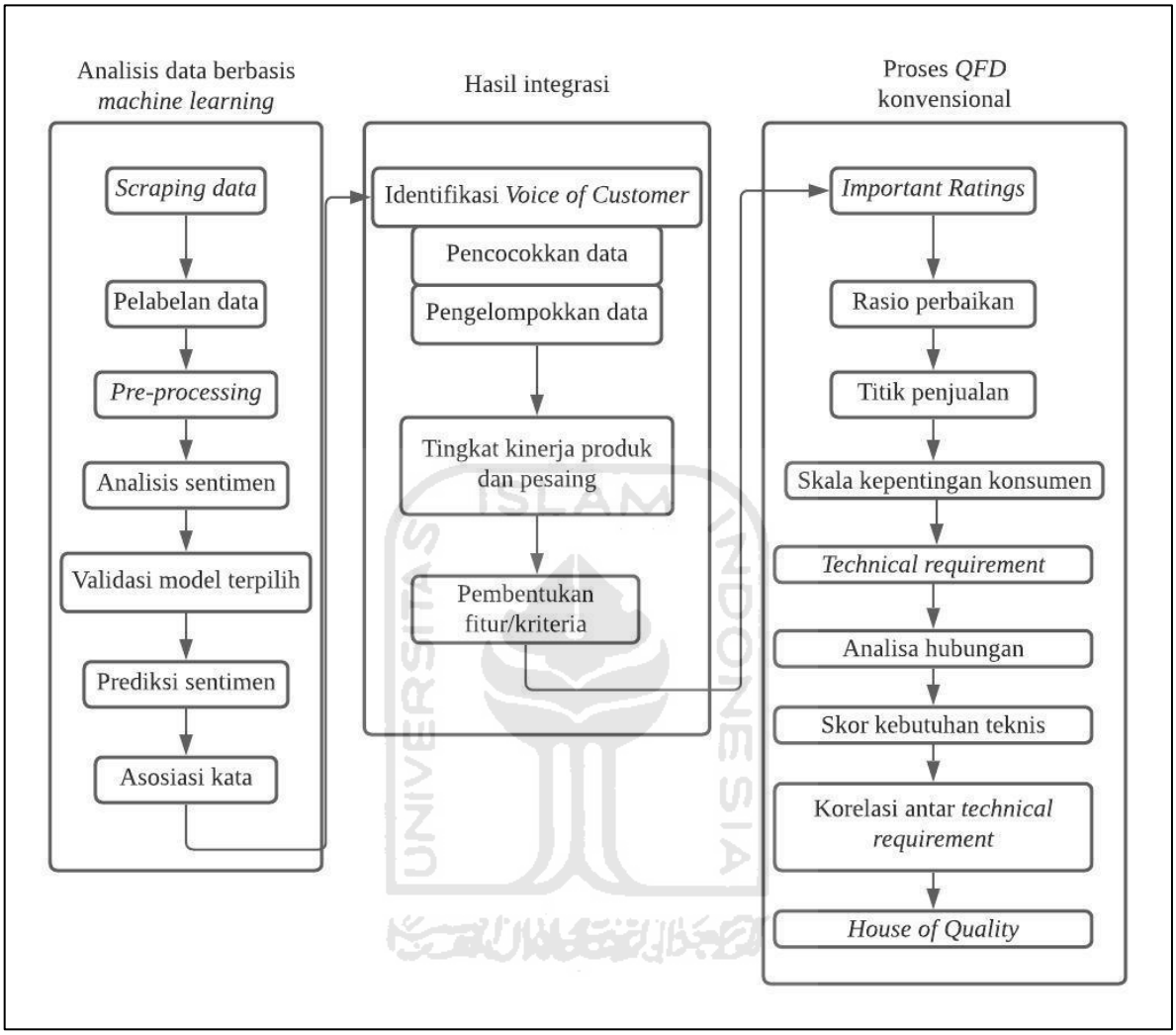
pencarian data pada produk pesaing, yaitu Converse Chuck Taylor 70s Low. Data produk pesaing melalui proses yang sama dengan data *review* produk Ventela Public Low. Proses pertama pada *QFD* dilakukan dengan pencocokkan data dan pengelompokkan data asosiasi yang memiliki term dengan informasi yang sama. Proses kedua yaitu menghitung kinerja produk, pada model integrasi ini nilai kinerja produk akan dihitung dari nilai asosiasinya. Kelemahan pada perhitungan ini yaitu apabila ada data dengan term yang memiliki informasi sama namun memiliki nilai asosisasi yang sangat berbeda dikarenakan kemunculan term yang berbeda. Hal ini akan berpengaruh karena nilai dihitung secara rata-rata maka bila terdapat *term* sama dengan nilai yang berbeda signifikan dapat menurunkan nilai. Untuk memperoleh nilai yang seimbang, dilakukan pembobotan untuk masing-masing fitur berdasarkan jumlah data penyusunnya. Selanjutnya membuat fitur atau atribut atau kriteria yang mewakili data penyusunnya, mulai pada proses ini peran *designer* diperlukan untuk menganalisis hasil data. Tahap berikutnya perhitungan *IR*, dikarenakan pentingnya suatu fitur perlu dinilai secara langsung, maka nilai asosiasi tidak dapat digunakan untuk proses ini.

Pada proses selanjutnya yaitu rasio perbaikan, menentukan titik penjualan, menentukan skala kepentingan konsumen, menentukan daftar kebutuhan teknis, analisa hubungan antara kebutuhan konsumen dengan kebutuhan teknis, menghitung skor kebutuhan teknis, menentukan korelasi antar kebutuhan teknis dan penyusunan semua data perhitungan dan analisa ke dalam matriks *HoQ*, diperlukan peran dari *stakeholder* seperti tim desain, tim pemasaran, dan bagian produksi. Dari hasil simulasi peneliti menggunakan data *dummy*, menunjukkan bahwa integrasi antara analisis data *review* menggunakan *machine learning* dangan *QFD* menghasilkan informasi yang layak sesuai dengan ketentuan metode *QFD* dan dapat mendukung proses pengembangan produk dari sisi jumlah data, topik data yang beragam, dan mengurangi subjektivitas pada tahap penentuan *VoC* dan kinerja produk dan pesaing. Tabel 6.1. menunjukkan rangkuman perbandingan, kelebihan dan kekurangan atas metode yang diteliti dengan penggunaan *QFD* secara konvensional.

Tabel 6. 1 Perbandingan dengan *QFD* dan Penelitian Terdahulu

No	Konvensional atau penelitian yang ada	Penelitian yang dilakukan
1	Mengajukan kuesioner terbuka tentang apa yang dibutuhkan konsumen atau apa pendapat tentang produk	Data yang digunakan sebagai *input* awal adalah hasil dari pengolahan data ulasan
2	Jumlah data kuesioner terbatas	Jumlah data dapat disesuaikan sesuai kebutuhan dan sangat beragam
3	Dalam menentukan *VoC* dari kuesioner biasa menggunakan *affinity diagrams* atau *tree diagrams*	*VoC* ditentukan dari hasil nilai asosiasi dan pertimbangan nilai kinerja produk dan pesaing
4	Dalam menentukan *VoC* melakukan proses pengelompokkan data	Dalam menentukan *VoC* melakukan proses pencocokkan dan pengelompokkan data. Data dapat dikembangkan untuk proses pengelompokkan dan pencocokkan secara otomatis dengan lebih mudah karena data telah spesifik secara informasi
5	Dalam mengelompokkan data melibatkan beberapa subjekif	Penentuan kriteria/fitur berdasarkan data penyusunnya dan diambil dari kesamaan kata dan mempertimbangkan data produk pesaing
7	Mengajukan kuesioner untuk menilai dari rentang 1-5 kinerja produk dan kinerja produk pesaing	Kinerja produk dan kinerja produk pesaing dihitung dengan nilai asosiasi kata dan pembobotan. Dengan asumsi, pada kelas negatif semakin sering dibicarakan maka nilai kinerja produk tersebut rendah. Dan untuk produk pesaing dapat menggunakan semua kelas karena berkedudukan sebagai pembanding. Perbedaan perhitungan terdapat pada nilai minimum dan maksimum pada proses normalisasi.
8	Penilaian rentang 1-5 berpotensi bias	Penilaian dengan nilai asosiasi dipengaruhi oleh nilai tiap penyusun fitur/kriteria. Apabila ada term yang nilai asosiasi berbeda secara signifikan dapat menurunkan nilai. Maka dilakukan pembobotan berdasarkan jumlah masing-masing penyusun.
9	Penelitian terdahulu dalam meningkatkan *QFD* berfokus pada mengurangi subjektivitas setelah mendapat *VoC*	*VoC* dan kinerja produk dan pesaing didapatkan melalui proses yang telah mengurangi subjektivitas yaitu melalui hasil pengolohan data *review*

Gambar 6.1 berikut merupakan visualisasi masing-masing proses dan bagian hasil integrasi sesuai dengan penelitian yang telah dilakukan.



Gambar 6. 1 Visualisasi Bagian Hasil Integrasi

6.2 Saran

Beberapa saran sesuai dengan penelitian yang telah dilakukan, yaitu:

1. Perusahaan dapat menginisiasi *platform* khusus sebagai wadah ulasan produk, sehingga informasi didapatkan secara lebih utuh dan *focus*.
2. Untuk penelitian selanjutnya, dapat dikembangkan algoritma *text processing* dalam Bahasa Indonesia sehingga dapat menangkap kata sesuai dengan karakteristik Bahasa Indonesia dan penggunaan algoritma klasifikasi lain seperti regresi linear, *decision tree*, dan *logistic regression*,
3. Mengintegrasikan penelitian ini dengan metode lain untuk mengoptimalkan dan mengurangi subjektivitas pada *QFD* seperti *fuzzy logic, artificial neural network, the taguchi method,* dan metode lain. Mengembangkan pendekatan berupa suatu formulasi perhitungan yang melibatkan hasil nilai *benchmarking* untuk dapat menerapkannya kedalam keputusan peringkat prioritas kriteria yang perlu diperbaiki.

DAFTAR PUSTAKA

Adetunmbi, A., Sarumi, O. A., Olutomilola, O., & Boyinbode, O. (2019). MACHINE LEARNING APPROACH TO SENTIMENT ANALYSIS OF USERS MOVIE REVIEWS. *i-manager's Journal on Computer Science, Vol. 7*, 9-15.

Aini, S. A., Sari, Y. A., & Arwan, A. (2018). Seleksi Fitur Information Gain untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode K-Nearest Neighbor dan Naive Bayes. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2546-2554.

Amin, F. (2012). Sistem Temu Kembali Informasi dengan Metode Vector Space Model. *Jurnal Sistem Informasi Bisnis 02*.

Arini, W. (2003). *Manajemen Kualitas*. Jakarta: Ghalia Indonesia.

Berthon, P., Pitt, L., & Campbell, C. (2008). Ad lib: When Customers Create Ad. *California Management Review, Vol.50 No.4*, 642-653.

BitDegree. 2020. *Splitting Dataset With the SKLearn train_test_spli Function*. *(online):* https://www.bitdegree.org/learn/train-test-split (18 Mei 2020)

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature Selection for Text Classification with Naive Bayes. *Expert System With Application*, 5432-5435.

Chormunge, S., & Jena, S. (2016). Efficient Feature Subset Selection Algorithm for High Dimentional Data. *International Journal of Electrical and Computer Engineering (IJECE)*, 1880-1888.

Choudhari, P., & Dhari, V. (2017). Sentiment Analysis and Machine Learning Based Sentiment Classification: A Review. *International Journal of Advanced Research in Computer Science*, 1051-1056.

Christian, H., Agus, M. P., & Suhartono, D. (2016). Single Document Automatic Text Summarization Using Term Frequency-Inverse Document Frequency (TF-IDF). *ComTech*, 285-294.

CNN Indonesia. 2019. *Netizen Indonesia Paling Gemar Belanja Online*. *(online):* https://www.cnnindonesia.com/teknologi/20190201173813-185-365769/netizen-indonesia-paling-gemar-belanja-online (21 April 2020)

Converse.2020. *(online):* converse.id (06 Agustus 2020)

Cudney, E. A., & Elrod, C. C. (2011). Quality Function Deployment in Continuous Improvement. *intechopen.com*, 3-78.

Dang, N. C., Moreno-Garcia, M. N., & De la Prieta, F. (2020). Sentiment Analysis Based on Deep Learning: A Comparative Study. *Electronics*, 1-29.

Evert, S. (2005). *The Statistics of Word Cooccurrences Word Pairs and Collocation*. Stuttgart: University of Stuttgart.

Evert, S., & Krenn, B. (2001). Methods for the Qualitative Evaluation of Lexical Association Measures. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (pp. 188-195). Toulouse: Association for Computational Linguistics.

Facebook. (2010). *Automated Data Collection Terms*. *(online):* https://www.facebook.com/apps/site_scraping_tos_terms.php (01 September 2020)

Farki, A., Baihaqi, I., & Wibawa, B. M. (2016). Pengaruh Online Customer Review dan Rating Terhadap Kepercayaan dan Minat Pembelian pada Online Marketplace di Indonesia. *JURNAL TEKNIK ITS Vol. 5, No. 2*, A614-A619.

Gasperz, V. (1998). *Manajemen Kualitas Dalam Industri Jasa*. Jakarta: Gramedia Pustaka Umum.

Goestch, D. L., & Davis, S. B. (2000). *The Total Quality Approach to Quality Management 3rd edition*. New Jersey: Prentice Hall.

Goetsch, D. L., & Davis, S. B. (2000). *Quality Management: Introduction to Total Quality Management for Production Processing and Services.* New Jersey: Pretince Hall.

Gorunescu, F. (2011). *Data Mining Concept, Models, and Techniques.* Berlin: Springer.

Gunawan, B., Pratiwi, H. S., & Pratama, E. E. (2018). Sistem Analisis Sentimen pada Ulasan Produk Menggunakan Metode Naive Bayes. *JEPIN Jurnal Edukasi dan Penelitian Informatika*, 113-118.

Hadna, N. S., Santosa, P. I., & Winarno, W. W. (2016). Studi Literatur Tentang Perbandingan Metode Untuk Proses Analisis Sentimen di Twitter. *Seminar Nasional Teknologi Informasi dan Komunikasi 2016 (SENTIKA 2016)*, 57-64.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques third edition*. Waltham: Elsevier.

Haviluddin. (2016). Big Data Analisis: Sebuah Peluang dan Tantangan Lulusan Informatika Dalam Menghadapi Pasar Global di Indonesia Timur. *Proceedings Seminar Nasional Riset Ilmu Komputer (SNRIK)* (pp. 1-5). Makassar: Universitas Muslim Indonesia.

IMARKETOLOGY. 2020. *Apa yang Diungkapkan Data Tentang Pasar Online Indonesia dan Global? (online):* https://academy.getcraft.com/id/blog/apa-yang-diungkapkan-data-tentang-pasar-online-indonesia-dan-global (21 April 2020)

indoworx.com. 2018. *Inilah 7 Produk Gagal Dari Perusahaan Terkenal. (online):* https://www.indoworx.com/produk-gagal-dari-perusahaan-terkenal/ (21 April 2020)

Instagram. 2013. *Term of Use*. *(online)*: https://www.instagram.com/about/legal/terms/before-january-19-2013/ (01 September 2020)

iPrice Indonesia. 2019. *Kumpulan 17 Perusahaan E-commerce Indonesia yang Gagal Eksis Sejak Era 2000-an*. *(online):* https://dailysocial.id/post/kumpulan-17-perusahaan-e-commerce-indonesia-yang-gagal-eksis-sejak-era-2000-an (21 April 2020)

Irish, S. S., Sherin, M. S., Surya, R., Vidhya, Y., & Ramamoorthy, V. (2017). Opinion Poll: Big Data Implementation of Unstructured Data Analytics of Social Network Reviews Using Sentiment Analysis & SVM. *International Journal of Advanced Engineering, Management and Science (IJAEMS)*, 288-291.

Jayani, D. H., & Fitra, S. (2020). *10 Media Sosial yang Paling Sering Digunakan di Indonesia*. *(online)*: https://databoks.katadata.co.id/datapublish/2020/02/26/10-media-sosial-yang-paling-sering-digunakan-di-indonesia# (01 September 2020)

Jayani, D. H., & Fitra, S. (2020). *Pengunjung Situs Shopee Terbesar di Indonesia*. *(online)*: https://databoks.katadata.co.id/datapublish/2020/07/06/pengunjung-situs-shopee-terbesar-di-indonesia# (01 September 2020)

Kasali, R. (2019). *Distruption, Tak Ada yang Tak Bisa Diubah Sebelum Dihadapi. Motivasi Saja Tidak Cukup*. Jakarta: PT Gramedia Pustaka Utama.

Khalid, Rintyarna, B. S., & Arifin, A. Z. (2015). Seleksi Fitur Dua Tahap Menggunakan Information Gain dan Artificiaal Bee Colony untuk Kategorisasi Teks Berbasis Support Vector Machine. *SYSTEMIC*, 22-26.

Kosasih, S. (2009). *Manajemen Operasi Internasional Edisi Pertama*. Jakarta: Mitra Wacana Media.

Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data Preprocessing for Supervised Learning. *International Journal of Computer Science*.

Kouw, W. M., van der Maaten, L. J., Krijthe, J. H., & Loog, M. (2016). Feature-Level Domain Adaptation. *Journal of Machine Learning Research 17*, 1-32.

Krisdamarjati, Y. A. 2020. *Ventela dan Tantangan Popularitas Sepatu Lokal di Indonesia*. *(online):* https://kompas.id/baca/riset/2020/02/29/ventela-dan-tantangan-popularitas-sepatu-lokal-di-indonesia/ (06 Agustus 2020)

Kumar, K. A., Jagadeesh, C., Kshirsagar, P., & Marve, S. M. (2020). Sentiment Analysis of Amazon Product Reviews using Machine Learning. *Engineering & Management (TEST)*, 5245-5254.

Li, M., Huang, L., Tan, C., & Wei, K. (2013). Helpfulness of online product reviews as seen by consumers: source and content features. *International Journal of Electronic Commerce, Vol. 17 No.4*, 101-136.

Liu, Y., Wang, G., Chen, H., Dong, H., Zhu, X., & Wang, S. (2011). An Improved Particle Swarm Optimization for Feature Selection. *Journal of Bionic Engineering*, 191-200.

Ma, B., Zhang, D., Yan, Z., & Kim, T. (2013). AN LDA AND SYNONYM LEXICON BASED APPROACH TO PRODUCT FEATURE EXTRACTION FROM ONLINE CONSUMER PRODUCT REVIEWS. *Journal of Electronic Commerce Research, VOL 14, NO 4*, 304-314.

Manning, Christopher, D., Prabhakar, R., & Hinrich, S. (2008). *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Najib, A. C., Irsyad, A., Qandi, G. A., & Rakhmawati, N. A. (2019). Perbandingan Metode Leexicon-based untuk Analisis Sentimen Berbasis Ontologi pada Kampanye Pilpres Indonesia Tahun 2019 di Twitter. *Fountain of Informatics Journal*, 41-48.

Negara, A. P., Muhardi, H., & Putri, I. M. (2020). Analisis Sentimen Maskapai Penerbangan Menggunakan Metode Naive Bayes dan Seleksi Fitur Information Gain. *JTIIK*, 599-606.

Nugraha, A. A., W, H. D., & Widayanto. (2017). Pengaruh Citra Merek, Kualitas Produk, dan Promosi terhadap Keputusan Pembelian Laptop Acer. *Jurnal Ilmu Administrasi Bisnis S1 Undip*, 291-300.

Powers, D. M. (2007, Desember). Evaluation: From Precision, Recall, and F-Factor. *Technical Report SIE-07-001*, pp. 1-22.

Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. *Journal of Infometrics*, 143-157.

Putra, J. G. (2019). *Pengenalan Konsep Pembelajaran Mesin dan Deep Learning Edisi 1.3*. Tokyo: self-published work.

Ramadhan Zulkarnain, G. R. (2014). Pengaruh Kualitas Produk dan Desain Produk terhadap Keputusan Pembelian Konsumen Tas Carrier Eiger di Jakarta. *Jurnal Ilmiah Universitas Bakrie*.

Resmi, C. 2019. *Ventela, Sepatu Lokal Bandung yang Mulai Hits di Kalangan Milenial*. *(online):* https://www.urbanasia.com/ventela-sepatu-lokal-bandung-yang-mulai-hits-di-kalangan-milenial-U5995 (06 Agustus 2020)

Rintyarna, B. S., Sarno, R., & Fatichah, C. (2019). Evaluating the performance of sentence level features and domain sensitive features of product reviews on supervised sentiment analysis tasks. *Journal of Big Data*, 1-19.

Ruchirawat, N. 2018. *Collocations — identifying phrases that act like single words in Natural Language Processing*. *(online):* https://medium.com/@nicharuch/collocations-identifying-phrases-that-act-like-individual-words-in-nlp-f58a93a2f84a (15 Juli 2020)

Ryansyah, A., & Andayani, S. (2017). Implementasi Algoritma TF-IDF Pada Pengukuran Kesamaan Dokumen. *Jurnal Sistem & Teknologi Informasi Komunikasi, Vol 1 No 1*, 1-10.

Sadhasivam, J., & Kalivaradhan, R. B. (2019). Sentiment Analysis of Amazon Products Using Ensemble Machine Learning Algorithm. *International Journal of Mathematical, Engineering and Management Sciences, Vol. 4, No. 2*, 508-520.

Sadono, M. S., Mulyani, G. T., & Purwanto, W. (2000). Implementasi Konsep Quality Function Deployment (QFD) untuk Meningkatkan Kualitas Produk Usaha Bakery. *AgriTech* , 197-208.

Salas Zarate, M., Moreira, J. M., Ortiz, K. L., Aveiga, H. L., Garcia, M. R., & Garcia, R. V. (2017). *Sentiment analysis on tweets about diabetes: An aspect-level approach.* Comput, Math, Method.

Santosa, B. (2007). *Data Mining Teknik Pemanfaatn Data Untuk Keperluan Bisnis*. Yogyakarta: Graha Ilmu.

Santosh, D. T. (2018). A Combined Approach for Effective Features Extraction from Online Product Reviews. *I.J. Education and Management Engineering,*, 11-21.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 1-49.

Sihombing, S., & Sitanggang, D. (2018). Analisis Desain Produk dan Jasa. *JRAK Vol 4*, 137-156.

Singh, A. K., Singh, A. K., Aman, & Anil. (2018). Sentiment Analysis for Product Reviews. *International Journal of Advanced Research in Computer Science, Vol 9 No. 3*, 23-26.

Singh, A., & Tucker, C. S. (2016). A machine learning approach to product review disambiguation. *Decision Support Systems (DECSUP)*, 1-37.

Siringoringo, R., & Jamaluddin. (2019). Text Mining dan Klasterisasi Sentimen Pada Ulasan Produk Toko Online. *Jurnal Penelitian Teknik Informatika Universitas Prima Indonesia (UNPRI), Vol 2 Nomor 2*.

Suciadi, J. (2001). Studi Analisis Metode-Metode Parsing dan Interpretasi Semantik pada Natural Language Processing. *Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra*, 13-22.

Suhendar, E., & Suroto. (2014). Penerapan Metode Quality Function Deployment (QFD) dalam Upaya Peningkatan Kualitas Pelayanan Akademik pada UB.

Sultana, N., Kumar, P., Patra, M. R., Chandra, S., & Alam, S. S. (2019). SENTIMENT ANALYSIS FOR PRODUCT REVIEW. *ICTACT JOURNAL ON SOFT COMPUTING, VOLUME: 09, ISSUE: 03*, 1913-1919.

Suthaharan, S. (2015). Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning. *Springer ISBN 9781489976413.*, 10.

Ventela. 2020. *(online):* https://www.ventela.com/ (06 Agustus 2020)

Wibowo, A. 2017. *10 Fold-Cross Validation*. *(online):*: https://mti.binus.ac.id/2017/11/24/10-fold-cross-validation/#:~:text=Cross%2Dvalidation%20(CV)%20adalah,pembelajaran%20dan%20data%20validasi%20%2F%20evaluasi. (15 Juli 2020)

Widiastuti, L. (2018). Pemilihan Fitur Pada Analisis Sentimen Review Travel Online Menggunakan Algoritma Naive Bayes Dalam Penerapan Mutual Information dan Particle Swarm Optimization (PSO). *IJCIT*, 91-100.

Widyawan, Damayanti, N. R., Adji, T. B., & Putra, G. D. (2019). Comparison of Feature Weighting in SVM Performance for Sentiment Analysis of Jakarta BRT. *IOP Conf. Series: Journal of Physics*, 1-11.

Wijaya, W. S., & Mustamu, R. H. (2013). Analisis Pengembangan Produk Pada Perusahaan Tepung Terigu di Surabaya. *AGORA Vol.1 No.1*, 294-303.

Xu, H., Zhang, Y., & DeGroof, R. (2018). A FEATURE-BASED SENTENCE MODEL FOR EVALUATION OF SIMILAR ONLINE PRODUCTS. *Journal of Electronic Commerce Research, VOL 19, NO 4*, 320-355.

Yuliarty, P., Permana, T., & Pratama, A. (2008). Pengembangan Desain Produk Papan Tulis dengan Metode Quality Function Deployment (QFD). *Jurnal Ilmiah PASTI Volume VI Edisi 1*, 1-13.

Zhao, M., Fu, C., Ji, L., Tang, K., & Zhou, M. (2011). Feature Selection and Parameter Optimization for Support Vector Machines: A New Approach Based on Genetic Algorithm with Feature Chromosomes. *Expert Systems With Applications*, 5197-5204.

Ziveria, M. (2014). Secara Semantik Menggunakan Teori Mutual. *Seminar Nasional Sistem Informasi Indonesia*, 203-210.

LAMPIRAN

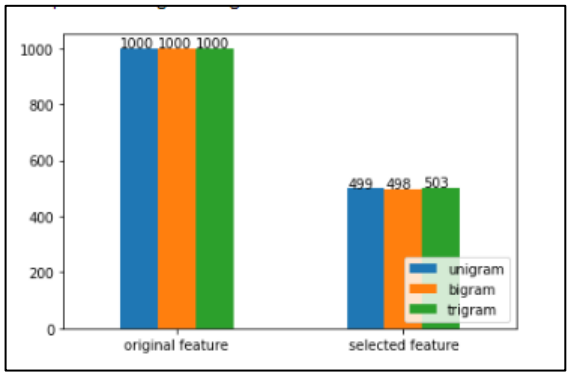
A-1 Hasil Klasifikasi Pada Dokumen A

	term	rank	MI_unigram
73	bagus	99.530042	1.000000
712	sangat	92.390493	0.728046
527	mantap	84.208407	0.497577
943	ukur	83.030463	0.418373
902	tidak	76.857328	0.049023
442	keren	60.801477	0.127973
152	buat	60.474318	0.199583
619	pakai	60.342665	0.000000
501	lokal	58.295449	0.000000
205	converse	57.698207	0.000000

	term	rank	MI_bigram
47	bagus sangat	65.223165	0.047112
571	merek lokal	47.740478	0.063536
500	lokal bangga	43.178148	0.155974
955	ukur besar	40.027801	0.264049
691	produk lokal	32.682258	0.020706
851	suka sangat	31.962463	0.437279
253	converse vans	31.826006	0.276197
196	buat ukur	31.694311	0.000000
465	lebih bagus	27.655963	0.081500
589	mirip vans	27.447276	0.067778

	term	rank	MI_trigram
36	buat desain sendiri	41.230970	0.000000
132	mirip sangat converse	26.326459	0.405508
975	tidak sesal beli	19.075414	0.063855
96	kualitas bagus sangat	18.647909	0.000000
5	bagus sangat tidak	17.173242	0.000000
31	bilang bangga lokal	17.173242	0.083856
43	buat ukur besar	16.454037	0.424034
994	vans dengan converse	15.455918	0.000000
193	nyaman sangat pakai	14.808633	0.000000
314	padu converse vans	14.549524	0.403525

Hasil *MI* Dokumen A



Grafik Banyakya Fitur Sebelum dan Sesudah *MI*

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul un

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.64      0.64      0.64        76
      netral       0.49      0.51      0.50        59
     positif       0.84      0.81      0.82        78

    accuracy                           0.67       213
   macro avg       0.66      0.65      0.65       213
weighted avg       0.67      0.67      0.67       213
```

Klasifikasi *Naive* Tanpa Normalisasi Dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  69.95305164319248
```

Klasifikasi *Naive* Tanpa Normalisasi Dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  61.03286384976526
```

Klasifikasi *SVM* Tanpa Normalisasi Dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  64.31924882629107
```

Klasifikasi *SVM* Tanpa Normalisasi Dengan *TFIDF*

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul unt

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.67      0.68      0.68        76
      netral       0.57      0.51      0.54        59
     positif       0.82      0.86      0.84        78

    accuracy                           0.70       213
   macro avg       0.68      0.68      0.68       213
weighted avg       0.69      0.70      0.70       213
```

Klasifikasi *Naive* Dengan Normalisasi Dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  68.07511737089203
```

Klasifikasi *Naive* Dengan Normalisasi Dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  65.25821596244131
```

Klasifikasi *SVM* Dengan Normalisasi Dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  65.25821596244131
```

Klasifikasi *SVM* Dengan Normalisasi Dengan *TFIDF*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

```
Naive Bayes Accuracy Score -> 68.54460093896714
```

Klasifikasi *Naive* Dengan Seleksi Fitur *TFIDF L1*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  65.72769953051643
```

Klasifikasi *SVM* Dengan Seleksi Fitur *TFIDF L1*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

```
Naive Bayes Accuracy Score -> 68.07511737089203
```

Klasifikasi *Naive* Dengan Seleksi Fitur *MI*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  61.97183098591549
```

Klasifikasi *SVM* Dengan Seleksi Fitur *MI*

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

     negatif     0.6615    0.5658    0.6099        76
      netral     0.4634    0.6441    0.5390        59
     positif     0.8788    0.7436    0.8056        78

    accuracy                         0.6526       213
   macro avg     0.6679    0.6511    0.6515       213
weighted avg     0.6862    0.6526    0.6619       213
```

Klasifikasi *Naive* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.63      0.53      0.58        76
           1       0.48      0.66      0.55        59
           2       0.88      0.77      0.82        78

    accuracy                           0.65       213
   macro avg       0.66      0.65      0.65       213
weighted avg       0.68      0.65      0.66       213
```

Klasifikasi *SVM* Linear *TFIDF* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.58	0.57	0.57	76
netral	0.48	0.49	0.48	59
positif	0.81	0.81	0.81	78
accuracy			0.63	213
macro avg	0.62	0.62	0.62	213
weighted avg	0.63	0.63	0.63	213

Klasifikasi *SVM* Linear *BoW* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

	precision	recall	f1-score	support
0	0.63	0.53	0.58	76
1	0.48	0.66	0.55	59
2	0.88	0.77	0.82	78
accuracy			0.65	213
macro avg	0.66	0.65	0.65	213
weighted avg	0.68	0.65	0.66	213

Klasifikasi *SVM RBF TFIDF* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.61	0.58	0.59	76
netral	0.46	0.56	0.51	59
positif	0.86	0.77	0.81	78
accuracy			0.64	213
macro avg	0.64	0.64	0.64	213
weighted avg	0.66	0.64	0.65	213

Klasifikasi *SVM Linear BoW* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.71	0.47	0.57	76
netral	0.43	0.76	0.55	59
positif	0.93	0.69	0.79	78
accuracy			0.63	213
macro avg	0.69	0.64	0.64	213
weighted avg	0.71	0.63	0.65	213

Klasifikasi *SVM Linear TFIDF L1* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.65	0.55	0.60	76
netral	0.47	0.68	0.55	59
positif	0.90	0.72	0.80	78
accuracy			0.65	213
macro avg	0.67	0.65	0.65	213
weighted avg	0.69	0.65	0.66	213

Klasifikasi *SVM RBF TFIDF L1* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.58	0.41	0.48	76
netral	0.41	0.66	0.50	59
positif	0.83	0.68	0.75	78
accuracy			0.58	213
macro avg	0.61	0.58	0.58	213
weighted avg	0.62	0.58	0.58	213

Klasifikasi *SVM Linear MI* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.58      0.49      0.53        76
      netral       0.42      0.61      0.50        59
     positif       0.83      0.68      0.75        78

    accuracy                           0.59       213
   macro avg       0.61      0.59      0.59       213
weighted avg       0.63      0.59      0.60       213
```

Klasifikasi *SVM RBF MI* Dengan *Hyperparameter*

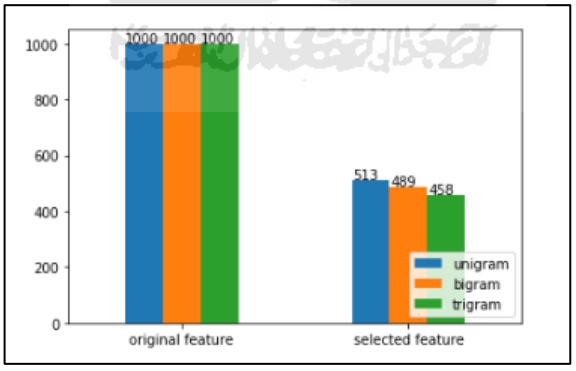
A-2 Hasil Klasifikasi Pada Dokumen C

	term	rank	MI_unigram
51	bagus	114.529344	1.000000
681	sangat	108.341464	0.648671
891	tidak	93.739733	0.104801
938	ukur	93.152511	0.208363
449	mantap	89.873889	0.433195
106	buat	77.833472	0.103454
535	pakai	76.368656	0.000000
141	converse	75.132575	0.094962
77	beli	73.210676	0.043263
355	keren	70.422496	0.284361

	term	rank	MI_bigram
20	bagus sangat	76.335447	0.998146
559	merek lokal	58.551862	0.000000
511	lokal bangga	48.350536	0.000000
967	ukur besar	45.267627	0.000000
702	produk lokal	42.969294	0.128616
241	converse vans	36.712660	0.000000
568	mirip sangat	34.252360	0.000000
569	mirip vans	34.147563	0.159124
346	insole empuk	34.042556	0.467931
544	mau beli	33.548415	0.000000

	term	rank	MI_trigram
55	buat desain sendiri	40.181076	0.000000
142	kualitas bagus sangat	27.725372	1.000000
191	mirip sangat converse	27.214511	0.563508
968	tidak sesal beli	26.321937	0.000000
457	pakai duduk jauh	21.273930	0.084913
89	dukung produk lokal	19.407760	0.241673
931	strip mirip vans	19.276945	0.000000
929	sol empuk sangat	18.369795	0.000000
83	daripada beli kw	17.728275	0.030944
13	bagus sangat tidak	17.728275	0.416733

Hasil *MI*



Grafik Perbandingan Sebelum dan Sesudah *MI*

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul un

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.52      0.59      0.55        81
      netral       0.58      0.44      0.50        95
     positif       0.78      0.87      0.82        90

    accuracy                           0.63       266
   macro avg       0.62      0.63      0.62       266
weighted avg       0.63      0.63      0.62       266
```

Klasifikasi *Naïve* Tanpa Normalisasi Dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  63.1578947368421
```

Klasifikasi *Naïve* Tanpa Normalisasi Dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  59.77443609022557
```

Klasifikasi *SVM* Tanpa Normalisasi Dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  60.902255639097746
```

Klasifikasi *SVM* Tanpa Normalisasi Dengan *TFIDF*

```
# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul un

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))
```

```
Testing selesai
              precision    recall  f1-score   support

     negatif       0.56      0.60      0.58        81
      netral       0.59      0.48      0.53        95
     positif       0.82      0.91      0.86        90

    accuracy                           0.67       266
   macro avg       0.66      0.67      0.66       266
weighted avg       0.66      0.67      0.66       266
```

Klasifikasi *Naïve* Dengan Normalisasi Dengan *BoW*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)
```

```
Naive Bayes Accuracy Score ->  66.16541353383458
```

Klasifikasi *Naïve* Dengan Normalisasi Dengan *TFIDF*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  62.03007518796993
```

Klasifikasi *SVM* Dengan Normalisasi Dengan *BoW*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  65.41353383458647
```

Klasifikasi *SVM* Dengan Normalisasi Dengan *TFIDF*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

```
Naive Bayes Accuracy Score -> 66.54135338345864
```

Klasifikasi *Naive* Dengan Seleksi Fitur *TFIDF L1*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  61.65413533834586
```

Klasifikasi *SVM* Dengan Seleksi Fitur *TFIDF L1*

```
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

```
Naive Bayes Accuracy Score -> 61.278195488721806
```

Klasifikasi *Naive* Dengan Seleksi Fitur *MI*

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
SVM Accuracy Score ->  58.27067669172933
```

Klasifikasi *SVM* Dengan Seleksi Fitur *MI*

```
Detailed classification report:

The model is trained on the full development set.
The scores are computed on the full evaluation set.

              precision    recall  f1-score   support

     negatif     0.5618    0.6173    0.5882        81
      netral     0.6154    0.5053    0.5549        95
     positif     0.8283    0.9111    0.8677        90

    accuracy                         0.6767       266
   macro avg     0.6685    0.6779    0.6703       266
weighted avg     0.6711    0.6767    0.6709       266
```

Klasifikasi *Naive* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.52      0.57      0.54        81
           1       0.56      0.55      0.55        95
           2       0.89      0.84      0.87        90

    accuracy                           0.65       266
   macro avg       0.66      0.65      0.66       266
weighted avg       0.66      0.65      0.66       266
```

Klasifikasi *SVM* Linear BoW Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

     negatif       0.45      0.68      0.54        81
      netral       0.56      0.33      0.41        95
     positif       0.82      0.82      0.82        90

    accuracy                           0.60       266
   macro avg       0.61      0.61      0.59       266
weighted avg       0.62      0.60      0.59       266
```

Klasifikasi *SVM Linear TFIDF* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.50	0.62	0.55	81
netral	0.57	0.51	0.54	95
positif	0.86	0.78	0.82	90
accuracy			0.63	266
macro avg	0.64	0.63	0.63	266
weighted avg	0.65	0.63	0.64	266

Klasifikasi *SVM RBF BoW* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

	precision	recall	f1-score	support
0	0.54	0.59	0.56	81
1	0.56	0.54	0.55	95
2	0.88	0.84	0.86	90
accuracy			0.66	266
macro avg	0.66	0.66	0.66	266
weighted avg	0.66	0.66	0.66	266

Klasifikasi *SVM RBF TFIDF* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.57	0.53	0.55	81
netral	0.54	0.63	0.58	95
positif	0.89	0.79	0.84	90
accuracy			0.65	266
macro avg	0.67	0.65	0.66	266
weighted avg	0.67	0.65	0.66	266

Klasifikasi *SVM Linear TFIDF L1* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.54	0.54	0.54	81
netral	0.56	0.58	0.57	95
positif	0.88	0.83	0.86	90
accuracy			0.65	266
macro avg	0.66	0.65	0.65	266
weighted avg	0.66	0.65	0.66	266

Klasifikasi *SVM RBF TFIDF L1* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.43	0.35	0.38	81
netral	0.49	0.61	0.54	95
positif	0.82	0.76	0.79	90
accuracy			0.58	266
macro avg	0.58	0.57	0.57	266
weighted avg	0.58	0.58	0.58	266

Klasifikasi *SVM Linear MI* Dengan *Hyperparameter*

```
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

	precision	recall	f1-score	support
negatif	0.49	0.33	0.40	81
netral	0.49	0.67	0.57	95
positif	0.85	0.77	0.81	90
accuracy			0.60	266
macro avg	0.61	0.59	0.59	266
weighted avg	0.61	0.60	0.60	266

Klasifikasi *SVM RBF MI* Dengan *Hyperparameter*

A-3 *Script* Data *Pre-processing*

```
import string
import re #regex library
# import word_tokenize & FreqDist from NLTK
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

# Langkah pertama kita harus memanggil library "pandas" dan menyingkatnya (memberikan
alias) dengan sebutan "pd"
import pandas as pd
import numpy as np

# Script berikut akan menyimpan data csv yang sudah dipanggil oleh Pandas ke dalam variabel
"data"
data = pd.read_json('data-label3.json')

# Ketika mengetik variabel "data" seperti di bawah, maka akan muncul visualisasi berbentuk
tabel
Data

import string
import re #regex library
# import word_tokenize & FreqDist from NLTK
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
```

```
def cleansing(sent):
    # Mengubah kata menjadi huruf kecil semua dengan menggunakan fungsi lower()
    string = str(sent).lower()
    #menghapus angka
    string = re.sub(r"\d+", "", string)
    #menghapus tanda baca
    string = re.sub("\W+","  ", string )
    #menghapus url
    string = re.sub(r"http\S+",  " ", string)
    return string
data["text_clean"] = data.text.apply(cleansing)


def remove_tweet_special(text):
    # remove tab, new line, ans back slice
    text = text.replace('\\t'," ").replace('\\n'," ").replace('\\u'," ").replace('\\',"")
    # remove non ASCII (emoticon, chinese word, .etc)
    text = text.encode('ascii', 'replace').decode('ascii')
    # remove mention, link, hashtag
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)","  ", text).split())
    # remove incomplete  URL
    return text.replace("http://",  " ").replace("https://",  " ")
data['text_clean'] = data['text_clean'].apply(remove_tweet_special)


#remove punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans('','',string.punctuation))
data['text_clean'] = data['text_clean'].apply(remove_punctuation)


#remove whitespace leading  & trailing
def remove_whitespace_LT(text):
    return text.strip()
data['text_clean'] = data['text_clean'].apply(remove_whitespace_LT)
```

```
#remove multiple  whitespace into single  whitespace
def remove_whitespace_multiple(text):
    return re.sub('\s+',' ',text)
data['text_clean'] = data['text_clean'].apply(remove_whitespace_multiple)


# remove single  char
def remove_singl_char(text):
    return re.sub(r"\b[a-zA-Z]\b", "", text)
data['text_clean'] = data['text_clean'].apply(remove_singl_char)


# import  word_tokenize  & FreqDist from NLTK
from nltk.tokenize  import  word_tokenize
from nltk.probability  import  FreqDist

# import  word_tokenize  & FreqDist from NLTK
from nltk.tokenize  import  word_tokenize
from nltk.probability  import  FreqDist
# NLTK word rokenize
def word_tokenize_wrapper(string):
    return word_tokenize(string)

data["text_clean1"] = data["text_clean"].apply(word_tokenize_wrapper)

print('Tokenizing  Result : \n')
print(data["text_clean1"].head())
print('\n\n\n')

from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory, StopWordRemover, ArrayDictionary
from nltk.tokenize  import  word_tokenize
```

```
# ambil stopword bawaan
stop_factory = StopWordRemoverFactory().get_stop_words()
more_stopword = ['ventela', 'lah', 'ya', 'udh', 'w', 'gw', 'wkwk', 'si', 'sih', 'lu', 'kok', 'aja', 'deh', 'dh', 'mah', 'wkwkwkwk',
'ni', 'lagi', 'gini', 'ane', 'jd', 'sm', 'sama', 'ny', 'nya', 'dok', 'om', 'kak', 'cuk', 'aj', 'ih', 'mba', 'mas', 'ntar', 'biar', 'aku', 'hahaha',
'ternyata', 'yang', 'yg', 'dari', 'mana', 'sepatu', 'dong', 'kalo', 'udah', 'gua', 'apa', 's', 'cuma', 'gue', 'gitu', 'gt', 'begitu', 'segitu',
'segitunya', 'dah', 'gue', 'kalau', 'kalo', 'emg', 'emang', 'wah', 'hehe', 'sepatunya', 'sepatuny', 'dr', 'dulu', 'dl', 'semua',
'smua', 'bakal', 'bkl', 'bakalan', 'bklan', 'bakl', 'bakln', 'duluan', 'dokter', 'tirta', 'nih', 'nah', 'nh', 'pokok', 'pokoknya',
'pkoknya', 'pkokny', 'dpt', 'pic', 'twitter', 'com', 'ken', 'dahhhhh', 'gan', 'aamiin', 'wkwkkwk', 'lahh', 'euyy', 'trnyata',
'anjirrrrrrrrr', 'awkowkowkowkow', 'dehhh', 'jga', 'yaa', 'd', 'ade', 'broh', 'sidesta', 'tns', 'wk', 'iniiii', 'nbtw', 'haha', 'a', 'fyi',
'siii', 'omdo', 'sumpah','mungkin', 'segini', 'mwehehe', 'an' 'dehhhhhh', 'nno', 'sumpahh', 'emng', 'jg', 'dahh', 'sihh', 'plis',
'pliss', 'aghu', 'dng', 'hahahaha', 'yupppp', 'tu', 'sepokat', 'kawan', 'yah', 'kayak', 'huhu', 'doang', 'yu', 'man', 'sepatutnya',
'men', 'tjoyyy', 'boskuuu', 'ahh', 'cuuuuyyy', 'awww', 'kaaa', 'woooii', 'ceunaah', 'bro', 'bree', 'shoe', 'd', 'di', 'oiya',
'hehehe', 'loh', 'jg', 'pars', 'nyaa', 'nihhhh', 'bosss', 'kyk', 'guys', 'ituu', 'yaa' 'trus', 'nyaaaa', 'jugaaaa', 'sepatunyaaaa', 'lo',
'ayok', 'min', 'ehh', 'ku', 'cuy', 'barangnye', 'emamg', 'eu', 'cooy', 'klo', 'bdg', 'at', 'kicks', 'nder', 'njir', 'jempul', 'tok', 'sak',
'e', 'lur', 'yo', 'seh', 'kan', 'dlm', 'cui', 'beuh', 'ns', 'siah', 'yayyyy', 'cmp', 'ssss', 'bing', 'gih', 'cipeng', 'persepatuan', 'bang',
'gor', 'huft', 'akutu', 'wkwkwkw', 'hahahaha', 'anjir', 'btw', 'mun', 'ulah', 'yampun', 'sepatuu', 'broo', 'sihh', 'laf', 'gabegitu',
'do', 'aaaa', 'co', 'wes', 'mboh', 'patu', 'sok', 'yh', 'a', 'uga', 'boim', 'nek', 'wae', 'iniii', 'nm', 'skdar', 'yup', 'in', 'maupum', 'jp',
'polar', 'x', 'yha', 'inisiiii', 'ko', 'yaaa', 'gan', 'elah', 'lg', 'opo', 'to', 'yaampun', 'jiiiiir', 'spt', 'ampeee', 'skrg', 'sik', 'kk', 'k',
'ae', 'wkwkwk', 'da', 'ah', 'hahahaha', 'wkwkw', 'ak', 'tuh', 'kaye', 'alo', 'kiwi', 'ntah', 'heheh', 'kwkwkwkwkwk', 'dear',
'amsyong', 'tu', 'venteka', 'udhh', 'vantella', 'gtu', 'ahahahahahahah', 'hmmmm', 'woiiii', 'cireng', 'cipeeeeeeeng', 'donk',
'hr', 'pliss', 'qo', 'bm', 'please', 'nyg', 'gitulah', 'dooongg', 'we', 'ituuu', 'kq', 'plissss', 'onoh', 'ventelaaaa', 'ajaa', 'in', 'guee',
'hihi', 'donggg', 'n', 'hahahahaha', 'aje', 'gpp', 'hhh', 'aplgi', 'tuch', 'haduhh', 'malah', 'ohh', 'nua', 'ndak', 'ajalah', 'na', 'weh',
'yaaa', 'ato', 'gpapa', 'yaudahlah', 'nge', 'wlwl', 'nichh', 'perna', 'aduhh', 'ah', 'gadeng', 'yauda', 'yaudah', 'cip', 'iyaa', 'yaaa',
'eosjeu', 'hiyahiya', 'btwww', 'atw', 'dngan', 'q', 'hehehehe', 'spatu', 'ehe', 'gwa', 'ajaaa', 'wta', 'vantela', 'ndan', 'don',
'qtela', 'ventala', 'ventella', 'nventela', 'venetla', 'entela', 'hh', 'laa', 'slur', 'ge', 'an', 'dehhhhhh', 'wkwkwkwkkw',
'wkwkwkkw', 'its', 'maah', 'getzke', 'cbnghxnk', 'livgvnmai', 'usliqibdt', 'coy', 'sepatuuu', 'aaaa', 'inisiiiii', 'yak', 'ki',
'woiiiii', 'cmon', 'ayoo', 'mulu', 'blahh', 'dongg', 'cipeeeeeeeeeng', 'wkw', 'wahai', 'sk', 'sss', 'adek', 'uss', 'dongs',
'ventelaaaaa', 'kkkkkk', 'huffttt', 'hayo', 'lau', 'bla', 'aya', 'euy', 'ter', 'meng', 'tt', 'apaan', 'nwkwkwk', 'yaudalah', 'kqoxbgr',
'au', 'lohh', 'bentar', 'dir', 'yeeeyy', 'yahh', 'sis', 'lahhh', 'nye', 'ye', 'eui', 'mu', 'noh', 'halah', 'rii', 'mom', 'nawkwkwkwk',
'loe' 'kaa' 'hwaa', 'wkkwkw', 'samaa', 'oalah', 'oy', 'lord', 'sepatuku', 'hahahhhah', 'dheee', 'abangnye', 'la', 'suatu', 'gaes',
'yaela', 'soan', 'wkwkkwkw', 'ntr', 'you', 'paste', 'patch', 'tanyadok', 'on', 'bosku', 'bos', 'aaaaa', 'ventelashoes', 'shoes',
'lho', 'halo', 'ayo', 'gee', 'hehhehehe', 'ventelanya', 'huffttt', 'duh', 'ng', 'hehee', 'wkakkaka', 'gituan', 'heheheh', 'akbarrr',
'ahhahahah', 'kaa', 'hwa', 'kkwkwkwkw', 'hahahhhhah', 'hwaa', 'iyaaa', 'that', 'loe', 'cem', 'shoesnya', 'darisananya', 'the',
'sepatunyaa', 'sepatunyaaa', 'pokonya', 'pokoke', 'pokoknyaaa', 'pokoe', 'ncuma', 'ternyataaa', 'inj', 'sepwtu', 'deng',
'itupun', 'seginiii', 'mak', 'eh', 'palah', 'njirrre', 'hmm', 'jierr', 'jah', 'wkwkkwkwk', 'spatune', 'rek', 'nu', 'ugaa', 'yaah',
'septu', 'ventelaa', 'njirrr', 'ehehehe', 'ahp', 'cuyy', 'mat', 'nboss', 'hiya', 'borr','yng', 'ajah', 'pak', 'dhe', 'nihh', 'dk', 'laaa', 'is',
'nini', 'cmiiw', 'klu', 'fi', 'gonta', 'situ', 'situnya', 'siiii', 'ja', 'hadeh', 'nok', 'sy', 'broooooo', 'hmmm', 'etela', 'huh',
'wkwkwkwkwk', 'nya', 'hha', 'awokwok', 'doank', 'kn', 'gasetu', 'ngue', 'mang', 'woi', 'wuihhh', 'coyy', 'ahha', 'apalah', 'kl',
'mr', 'kucing', 'ventntela', 'hahahahahah', 'allah', 'aku', 'saya', 'ak', 'sy', 'cuman', 'cmn', 'sumpaah', 'dngn', 'trs', 'jdi', 'sma',
'trus', 'nyang', 'lgi', 'dgm', 'nterus', 'nmending', 'nkalau', 'nkalo', 'dlu', 'dengn', 'sing', 'mgkin', 'dri', 'nitu', 'of', 'lagian', 'ari',
'mangkanya', 'kalo', 'dor', 'denga', 'dengan', 'decit', 'kalau', 'mungkin', 'dari', 'yang', 'terus', 'lagi', 'too', 'mendingan',
'bntr', 'smpe', 'sampai', 'hehwhw', 'uwe', 'ploduk', 'vetntela', 'heuheu', 'ix', 'wkkwkwkwkw', 'kwkwkwkwkw', 'emnk',
'nk', 'bre', 'eug', 'tw', 'po', 'hihih', 'bosque', 'pp', 'idih', 'poko', 'kaka', 'huffffft', 'hihihiii', 'cuman', 'jug', 'mbak', 'ffjcdktem',
'btrtpvow', 'atuh', 'banyusadewa', 'ua', 'ad', 'segituu', 'hadehhh', 'teroooooosss', 'produc', 'keknya', 'paleng', 'wkwkkw',
'hemm', 'nsing', 'sepatumu', 'sepatau', 'nkayaknya', 'kwkwkw', 'bng', 'venetela', 'nw', 'cuuuuy', 'wkkw', 'huahahaha', 'ii',
'kayaknyaa', 'tira', 'jugaa', 'yuhuuu', 'ahhh', 'denis', 'mahh', 'laaah', 'oz', 'lol', 'ngw', 'pokoknyaa', 'kykny', 'sptuny', 'wahh',
'boss', 'dhafinelka', 'muntiiaraa', 'nya', 'iya', 'itu']
```

```
# menggabungkan stopword
data1 = stop_factory + more_stopword

dictionary = ArrayDictionary(data1)
str = StopWordRemover(dictionary)

def stopwords_removal(words):
    return [word for word in words if word not in data1]
data["text_clean_stopword"] = data["text_clean1"].apply(stopwords_removal)
print(data["text_clean_stopword"])


normalizad_word = pd.read_excel("normalisasi.xlsx")
normalizad_word_dict = {}
for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]

def normalized_term(document):
    return [normalizad_word_dict[term] if term in normalizad_word_dict else term for term in
document]
data['text_normalized'] = data['text_clean_stopword'].apply(normalized_term)
data['text_normalized'].head(10)


conda install -c conda-forge swifter
```

```
# import Sastrawi package
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import swifter

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
    return stemmer.stem(term)

term_dict = {}

for document in data["text_normalized"]:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '

print(len(term_dict))
print("------------------------")

for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    print(term,":"  ,term_dict[term])

print(term_dict)
print("------------------------")

# apply stemmed term to dataframe
def get_stemmed_term(document):
    return [term_dict[term] for term in document]

data["text_clean_stemmed"] = data["text_normalized"].swifter.apply(get_stemmed_term)
print(data["text_clean_stemmed"])

# Pada akhirnya setelah proses cleansing sudah selesai, langkah terakhir mengexport-nya
kembali dalam bentuk csv
data.to_csv('Text_Preprocessing-1176b.csv')

data.to_excel('"Text_Preprocessing-fix-1176b.xlsx")
```

A-4 Script Klasifikasi Tanpa Feature

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
import pickle
import re

# Mengambil dataset twitter berbahasa indonesia
datareview = pd.read_csv('Text_Preprocessing-1176b.csv')

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = datareview.text_clean_stemmed.tolist()

# Kita proses Feature Extraction
count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

pickle.dump(count_vect, open("feature.p", "wb"))

X = count_vect.transform(data_preprocessed)
print ("Feature Extraction selesai")
```

```
# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=datareview.sentimen

X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)

# Langkah selanjutnya adalah "Training"
# "MultinomialNB" merupakan modul untuk metode "Naive Bayes"

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Save model
pickle.dump(clf, open("model-sentiment.p", "wb"))
print ("Training selesai")

# Langkah selanjutnya adalah "Testing"
# "classification_report" merupakan salah satu modul untuk melakukan "Testing & Validation"

from sklearn.metrics import classification_report

test = clf.predict(X_test)

print ("Testing selesai")

print(classification_report(y_test, test))

# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)
```

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

Corpus = pd.read_csv('Text_Preprocessing-1176b.csv')

Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(Corpus['text_clean_stemmed'],Corpus['sentimen'],test_size=0.2,
random_state=77)

Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)

Tfidf_vect = TfidfVectorizer(max_features=10000)
Tfidf_vect.fit(Corpus['text_clean_stemmed'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)

# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*100)

# Classifier - Algorithm - SVM
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

A-5 Script Proses *TFIDF L1*

```
import pandas as pd
import numpy as np

datareview = pd.read_csv("Text_Preprocessing-1176b.csv", usecols=["sentimen", "text_clean_stemmed"])
datareview.columns = ["sentimen", "text"]

datareview.head()

# join list of token as single document string
import ast

def join_text_list(texts):
    texts = ast.literal_eval(texts)
    return ' '.join([text for text in texts])
datareview["text_join"] = datareview["text"].apply(join_text_list)

datareview["text_join"].head()

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import normalize

max_features = 1000


def generate_tfidf_mat(min_gram, max_gram):
    cvect = CountVectorizer(max_features=max_features, ngram_range=(min_gram, max_gram))
    counts = cvect.fit_transform(datareview["text_join"])

    normalized_counts = normalize(counts, norm='l1', axis=1)

    tfidf = TfidfVectorizer(max_features=max_features, ngram_range=(min_gram, max_gram), smooth_idf=False)
    tfs = tfidf.fit_transform(datareview["text_join"])

    tfidf_mat = normalized_counts.multiply(tfidf.idf_).toarray()

    TF = normalized_counts.toarray()
    IDF = tfidf.idf_
    TF_IDF = tfidf_mat
    return TF, IDF, TF_IDF, tfidf.get_feature_names()

# ngram_range (1, 1) to use unigram only
tf_mat_unigram, idf_mat_unigram, tfidf_mat_unigram, terms_unigram = generate_tfidf_mat(1,1)

# ngram_range (2, 2) to use bigram only
tf_mat_bigram, idf_mat_bigram, tfidf_mat_bigram, terms_bigram = generate_tfidf_mat(2,2)

# ngram_range (3, 3) to use trigram only
tf_mat_trigram, idf_mat_trigram, tfidf_mat_trigram, terms_trigram = generate_tfidf_mat(3,3)
```

```
# ---------- check sparse data ------------------
idx_sample = 0

print("Show  TFIDF sample ke-" + str(idx_sample), "\n")
print(datareview["text"][idx_sample],  "\n")

print("\t\t\t",  "TF",  "\t\t",  "IDF",  "\t\t",  "TF-IDF",  "\t",  "Term\n")
for i, item in enumerate(zip(tf_mat_unigram[idx_sample],  idf_mat_unigram,
tfidf_mat_unigram[idx_sample],  terms_unigram)):
   if(item[2] != 0.0):
      print  ("array position  " + str(i) + "\t",
           "%.6f" % item[0], "\t",
           "%.6f" % item[1], "\t",
           "%.6f" % item[2], "\t",
           item[3])

def get_TF_unigram(row):
   idx = row.name
   return [tf for tf in tf_mat_unigram[idx]  if  tf != 0.0]

datareview["TF_UNIGRAM"] = datareview.apply(get_TF_unigram, axis=1)

def get_IDF_unigram(row):
   idx = row.name
   return [item[1] for item in zip(tf_mat_unigram[idx],  idf_mat_unigram) if item[0] != 0.0]

datareview["IDF_UNIGRAM"] = datareview.apply(get_IDF_unigram, axis=1)

def get_TFIDF_unigram(row):
   idx = row.name
   return [tfidf  for tfidf  in tfidf_mat_unigram[idx]  if  tfidf  != 0.0]

datareview["TFIDF_UNIGRAM"] = datareview.apply(get_TFIDF_unigram, axis=1)

datareview[["text", "TF_UNIGRAM", "IDF_UNIGRAM", "TFIDF_UNIGRAM"]].head()

# save TFIDF Unigram to Excel

datareview[["text", "TF_UNIGRAM", "IDF_UNIGRAM",
"TFIDF_UNIGRAM"]].to_excel("TFIDF_Unigram-1176b.xlsx")
```

```
def get_TF_bigram(row):
    idx = row.name
    return [tf for tf in tf_mat_bigram[idx] if tf != 0.0]

datareview["TF_BIGRAM"] = datareview.apply(get_TF_bigram, axis=1)

def get_IDF_bigram(row):
    idx = row.name
    return [item[1] for item in zip(tf_mat_bigram[idx], idf_mat_bigram) if item[0] != 0.0]

datareview["IDF_BIGRAM"] = datareview.apply(get_IDF_bigram, axis=1)

def get_TFIDF_bigram(row):
    idx = row.name
    return [tfidf for tfidf in tfidf_mat_bigram[idx] if tfidf != 0.0]

datareview["TFIDF_BIGRAM"] = datareview.apply(get_TFIDF_bigram, axis=1)

def get_Term_bigram(row):
    idx = row.name
    return [item[1] for item in zip(tf_mat_bigram[idx], terms_bigram) if item[0] != 0.0]

datareview["text_BIGRAM"] = datareview.apply(get_Term_bigram, axis=1)

datareview[["text_BIGRAM", "TF_BIGRAM", "IDF_BIGRAM", "TFIDF_BIGRAM"]].head()


# save TFIDF Bigram to Excel

datareview[["text_BIGRAM", "TF_BIGRAM", "IDF_BIGRAM", "TFIDF_BIGRAM"]].to_excel("TFIDF_Bigram-
1176b.xlsx")

def get_TF_trigram(row):
    idx = row.name
    return [tf for tf in tf_mat_trigram[idx] if tf != 0.0]

datareview["TF_trigram"] = datareview.apply(get_TF_trigram, axis=1)

def get_IDF_trigram(row):
    idx = row.name
    return [item[1] for item in zip(tf_mat_trigram[idx], idf_mat_trigram) if item[0] != 0.0]

datareview["IDF_trigram"] = datareview.apply(get_IDF_trigram, axis=1)

def get_TFIDF_trigram(row):
    idx = row.name
    return [tfidf for tfidf in tfidf_mat_trigram[idx] if tfidf != 0.0]

datareview["TFIDF_trigram"] = datareview.apply(get_TFIDF_trigram, axis=1)
```

```
def get_Term_trigram(row):
    idx = row.name
    return [item[1] for item in zip(tf_mat_trigram[idx], terms_trigram) if item[0] != 0.0]

datareview["text_TRIGRAM"] = datareview.apply(get_Term_trigram, axis=1)

datareview[["text_TRIGRAM", "TF_trigram", "IDF_trigram", "TFIDF_trigram"]].head()


# save TFIDF Trigram to Excel

datareview[["text_TRIGRAM", "TF_trigram", "IDF_trigram",
"TFIDF_trigram"]].to_excel("TFIDF_Trigram-1176b.xlsx")

from scipy import sparse

# save sparse matrix unigram, bigram and trigram to .npz file
sparse.save_npz("tfidf_mat_unigram-1176b.npz", sparse.csr_matrix(tfidf_mat_unigram))
sparse.save_npz("tfidf_mat_bigram-1176b.npz", sparse.csr_matrix(tfidf_mat_bigram))
sparse.save_npz("tfidf_mat_trigram-1176b.npz", sparse.csr_matrix(tfidf_mat_trigram))

import json

def get_dict_feature_name(terms):
    feature_name = {}
    feature_name['feature'] = terms
    return feature_name


def save_to_json(Data, json_filename):
    with open(json_filename, mode='w') as json_config:
        json.dump(Data, json_config)

# save feature name unigram, bigram and trigram as json file
save_to_json( get_dict_feature_name(terms_unigram),"tfidf_feature_name_unigram-
1176b.json")
save_to_json( get_dict_feature_name(terms_bigram),"tfidf_feature_name_bigram-1176b.json")
save_to_json( get_dict_feature_name(terms_trigram),"tfidf_feature_name_trigram-1176b.json")
```

A-6 Script Proses *MI*

```
import numpy as np
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

from sklearn.feature_selection import mutual_info_classif
from scipy import sparse
import pandas as pd
import numpy as np
np.set_printoptions(suppress=True)

# read sparse matrix yang sebelumnya disimpan pada file .npz
tfidf_mat_unigram = sparse.load_npz("tfidf_mat_unigram-1176b.npz").toarray()
tfidf_mat_bigram = sparse.load_npz("tfidf_mat_bigram-1176b.npz").toarray()
tfidf_mat_trigram = sparse.load_npz("tfidf_mat_trigram-1176b.npz").toarray()

# read label/class dari csv preprocessing untuk keperluan fiture selection
tags = pd.read_csv("Text_Preprocessing-1176b.csv", usecols=["sentimen"])

# calculate mutual info (MI) / information gain (GI) untuk tiap feature
mi_unigram = mutual_info_classif(tfidf_mat_unigram, tags.sentimen)
mi_bigram = mutual_info_classif(tfidf_mat_bigram, tags.sentimen)
mi_trigram = mutual_info_classif(tfidf_mat_trigram, tags.sentimen)

# normalisasi (L1) untuk semua hasil perhitungan MI/GI
norm_mi_unigram = mi_unigram/np.max(mi_unigram)
norm_mi_bigram = mi_bigram/np.max(mi_bigram)
norm_mi_trigram = mi_trigram/np.max(mi_trigram)

# read json feature name yang sebelumnya disimpan ditahap feature engineering tfidf
import json

def read_json_feature_name(file_name, Keys):
    with open(file_name) as json_config:
        json_object = json.load(json_config)

    param = []
    for item in json_object:
        if item in Keys:
            param.append(json_object[item])

    return param

terms_unigram = read_json_feature_name("tfidf_feature_name_unigram-1176b.json", "feature")
terms_bigram = read_json_feature_name("tfidf_feature_name_bigram-1176b.json", "feature")
terms_trigram = read_json_feature_name("tfidf_feature_name_trigram-1176b.json", "feature")
```

```
# sum tfidf frequency of each term through documents
sums_unigram = tfidf_mat_unigram.sum(axis=0)
sums_bigram = tfidf_mat_bigram.sum(axis=0)
sums_trigram = tfidf_mat_trigram.sum(axis=0)

def generate_ranking(terms, sums):
    # connecting term to its sums frequency
    data = []
    for col, term in enumerate(terms):
        data.append((term, sums[col] ))
    ranking = pd.DataFrame(data, columns=['term','rank'])
    return ranking.sort_values('rank', ascending=False)


ranking_unigram = generate_ranking(terms_unigram[0], sums_unigram)
ranking_bigram = generate_ranking(terms_bigram[0], sums_bigram)
ranking_trigram = generate_ranking(terms_trigram[0], sums_trigram)


# apply norm_mi to Dataframe rangking
ranking_unigram["MI_unigram"] = ranking_unigram.apply(lambda x:
norm_mi_unigram[x.name], axis=1)
ranking_bigram["MI_bigram"] = ranking_bigram.apply(lambda x: norm_mi_bigram[x.name],
axis=1)
ranking_trigram["MI_trigram"] = ranking_trigram.apply(lambda x: norm_mi_trigram[x.name],
axis=1)

# show top 10 rank (total tfidf) unigram
print(ranking_unigram.head(10))

# show top 10 rank (total tfidf) unigram
print(ranking_bigram.head(10))

# show top 10 rank (total tfidf) unigram
print(ranking_trigram.head(10))
```

```
# get all column index on norm_mi below 0.01
def get_column_idx(norm_mi, threshold):
    return [i for i, mi_item in enumerate(norm_mi) if mi_item < threshold]

column_idx_unigram = get_column_idx(norm_mi_unigram, 0.01)
column_idx_bigram = get_column_idx(norm_mi_bigram, 0.01)
column_idx_trigram = get_column_idx(norm_mi_trigram, 0.01)


# delete sparse matrix tfidf_mat where column=column_id, save into new array
tfidf_mat_selection_unigram = np.delete(tfidf_mat_unigram, column_idx_unigram ,1)
tfidf_mat_selection_bigram = np.delete(tfidf_mat_bigram, column_idx_bigram ,1)
tfidf_mat_selection_trigram = np.delete(tfidf_mat_trigram, column_idx_trigram ,1)

# number of feature after reduction
max_features_unigram = tfidf_mat_selection_unigram.shape[1]
max_features_bigram = tfidf_mat_selection_bigram.shape[1]
max_features_trigram = tfidf_mat_selection_trigram.shape[1]

# plot number of feature on bar chart

groups = [[tfidf_mat_unigram.shape[1],
          tfidf_mat_bigram.shape[1],
          tfidf_mat_trigram.shape[1]],

         [tfidf_mat_selection_unigram.shape[1],
          tfidf_mat_selection_bigram.shape[1],
          tfidf_mat_selection_trigram.shape[1]]]

group_labels = ['original feature', 'selected feature']

# Convert data to pandas DataFrame.
df = pd.DataFrame(groups, index=group_labels, columns=["unigram", "bigram", "trigram"])

# plot into bar chart
ax = df.plot(kind='bar', rot=0)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))

ax.legend(loc='lower right')
```

```
from scipy import sparse
import json

#-----------------------------------------------------------------------------------------------------
# save ranking data to excel
ranking_unigram.to_excel("Feature_Selection_Information_Gain_Unigram-1176b.xlsx")
ranking_bigram.to_excel("Feature_Selection_Information_Gain_Bigram-1176b.xlsx")
ranking_trigram.to_excel("Feature_Selection_Information_Gain_Trigram-1176b.xlsx")


#-----------------------------------------------------------------------------------------------------
# save sparse matrix unigram, bigram and trigram to .npz file
sparse.save_npz("tfidf_mat_selection_unigram-1176b.npz", sparse.csr_matrix(tfidf_mat_selection_unigram))
sparse.save_npz("tfidf_mat_selection_bigram-1176b.npz", sparse.csr_matrix(tfidf_mat_selection_bigram))
sparse.save_npz("tfidf_mat_selection_trigram-1176b.npz", sparse.csr_matrix(tfidf_mat_selection_trigram))


#-----------------------------------------------------------------------------------------------------
# save feature name after feature selection to JSON
def get_dict_feature_name(terms):
    feature_name = {}
    feature_name['feature'] = terms
    return feature_name


def save_to_json(Data, json_filename):
    with open(json_filename, mode='w') as json_config:
        json.dump(Data, json_config)

def get_selected_feature(norm_mi, column_idx, terms):
    selected_idx = [j for j in range(len(norm_mi)) if j not in column_idx]
    selected_features = []
    for idx in selected_idx:
        selected_features.append(terms[0][idx])
    return selected_features

# get feature name after feature selection
terms_selection_unigram = get_selected_feature(norm_mi_unigram, column_idx_unigram, terms_unigram)
terms_selection_bigram = get_selected_feature(norm_mi_bigram, column_idx_bigram, terms_bigram)
terms_selection_trigram = get_selected_feature(norm_mi_trigram, column_idx_trigram, terms_trigram)

# save feature name unigram, bigram and trigram as json file
save_to_json( get_dict_feature_name(terms_selection_unigram),"tfidf_feature_name_selection_unigram-1176b.json")
save_to_json( get_dict_feature_name(terms_selection_bigram),"tfidf_feature_name_selection_bigram-1176b.json")
save_to_json( get_dict_feature_name(terms_selection_trigram),"tfidf_feature_name_selection_trigram-1176b.json")
```

A-7 Script Klasifikasi Dengan Seleksi Fitur *MI1*

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from scipy import sparse
import pandas as pd
import pickle
import numpy as np

# ---------------------- Get Sparse Matrix & Tags , Unigram, BIgram and Trigram ------------------
def get_sparse_n_tags(file_names, dataset_name):

    # load preprocessed dataset as dataframe
    LABEL_DATA = pd.read_csv(dataset_name, encoding = "ISO-8859-1", usecols=["sentimen"])
    tags = LABEL_DATA.sentimen

    # load sparse matrix
    unigram = sparse.load_npz(file_names[0]).toarray()
    bigram = sparse.load_npz(file_names[1]).toarray()
    trigram = sparse.load_npz(file_names[2]).toarray()

    return [unigram, bigram, trigram, tags]

sparse_file_names = [
                "tfidf_mat_selection_unigram-1176b.npz",
                "tfidf_mat_selection_bigram-1176b.npz",
                "tfidf_mat_selection_trigram-1176b.npz"]
preprocessed_dataset_name = "Text_Preprocessing-1176b.csv"

sparse_unigram, sparse_bigram, sparse_trigram, tags = \
                    get_sparse_n_tags(sparse_file_names, preprocessed_dataset_name)

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = sparse_unigram.tolist()

X = (data_preprocessed)
print ("Feature Extraction selesai")

# Selanjutnya kita split dataset menjadi data training dan data testing
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=tags
X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)
```

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)


# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

A-8 Script Klasifikasi Dengan Seleksi Fitur *TFIDF L1*

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from scipy import sparse
import pandas as pd
import pickle
import numpy as np

# ---------------------- Get Sparse Matrix & Tags , Unigram, BIgram and Trigram ------------------
def get_sparse_n_tags(file_names, dataset_name):

    # load preprocessed dataset as dataframe
    LABEL_DATA = pd.read_csv(dataset_name, encoding = "ISO-8859-1", usecols=["sentimen"])
    tags = LABEL_DATA.sentimen

    # load sparse matrix
    unigram = sparse.load_npz(file_names[0]).toarray()
    bigram = sparse.load_npz(file_names[1]).toarray()
    trigram = sparse.load_npz(file_names[2]).toarray()

    return [unigram, bigram, trigram, tags]

sparse_file_names = [
                "tfidf_mat _unigram-1176b.npz",
                "tfidf_mat _bigram-1176b.npz",
                "tfidf_mat _trigram-1176b.npz"]
preprocessed_dataset_name = "Text_Preprocessing-1176b.csv"

sparse_unigram, sparse_bigram, sparse_trigram, tags = \
                    get_sparse_n_tags(sparse_file_names, preprocessed_dataset_name)

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = sparse_unigram.tolist()

X = (data_preprocessed)
print ("Feature Extraction selesai")

# Selanjutnya kita split dataset menjadi data training dan data testing
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=tags
X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)
```

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', gamma='scale')
SVM.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)


# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(X_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(X_test) # Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score ->",accuracy_score(predictions_NB, y_test)*100)
```

A-9 Script Klasifikasi *Hyperparameter Naive*Tanpa *Feature*

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
import pickle
import re

# Mengambil dataset twitter berbahasa indonesia
datareview = pd.read_csv('Text_Preprocessing-1176b.csv')

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
X = datareview.text_clean_stemmed.tolist()

# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
y=datareview.sentimen

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=77)
```

```
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV

text_clf = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])

tuned_parameters = {
   'vect__ngram_range': [(1, 1), (1, 2), (2, 2)],
   'tfidf__use_idf': (True, False),
   'tfidf__norm': ('l1', 'l2'),
   'clf__alpha': [1, 1e-1, 1e-2]
}

from sklearn.metrics import classification_report

score = 'f1_macro'
print("# Tuning hyper-parameters for %s" % score)
print()
np.errstate(divide='ignore')
clf = GridSearchCV(text_clf, tuned_parameters, cv=10, scoring=score)
clf.fit(X_train, y_train)

print("Best parameters set found on development set:")
print()
print(clf.best_params_)
print()
print("Grid scores on development set:")
print()
for mean, std, params in zip(clf.cv_results_['mean_test_score'],
                             clf.cv_results_['std_test_score'],
                             clf.cv_results_['params']):
   print("%0.3f (+/-%0.03f) for %r" % (mean, std * 2, params))
print()

print("Detailed classification report:")
print()
print("The model is trained on the full development set.")
print("The scores are computed on the full evaluation set.")
print()
print(classification_report(y_test, clf.predict(X_test), digits=4))
print()
```

A-10 Script Klasifikasi Hyperparameter SVM Tanpa Feature

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
import pickle
import re

# Mengambil dataset twitter berbahasa indonesia
datareview = pd.read_csv('Text_Preprocessing-1176b.csv')

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = datareview.text_clean_stemmed.tolist()

# Kita proses Feature Extraction
count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

pickle.dump(count_vect, open("feature.p", "wb"))

X = count_vect.transform(data_preprocessed)
print ("Feature Extraction selesai")
```

```
# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=datareview.sentimen

X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)

from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))

import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

Corpus = pd.read_csv('Text_Preprocessing-1176b.csv')

Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(Corpus['text_clean_stemmed'],Corpus['sentimen'],test_size=0.2, random_state=77)
```

```
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)

Tfidf_vect = TfidfVectorizer(max_features=10000)
Tfidf_vect.fit(Corpus['text_clean_stemmed'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)

from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(Train_X_Tfidf, Train_Y)

# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

grid_predictions = grid.predict(Test_X_Tfidf)

# print classification report
print(classification_report(Test_Y, grid_predictions))
```

A-11 Script Klasifikasi Hyperparameter SVM Dengan Seleksi Feature TFIDF L1

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from scipy import sparse

import pandas as pd
import pickle
import numpy as np

import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score


# ---------------------- Get Sparse Matrix & Tags , Unigram, BIgram and Trigram ------------------
def get_sparse_n_tags(file_names, dataset_name):

    # load preprocessed dataset as dataframe
    LABEL_DATA = pd.read_csv(dataset_name, encoding = "ISO-8859-1", usecols=["sentimen"])
    tags = LABEL_DATA.sentimen

    # load sparse matrix
    unigram = sparse.load_npz(file_names[0]).toarray()
    bigram = sparse.load_npz(file_names[1]).toarray()
    trigram = sparse.load_npz(file_names[2]).toarray()

    return [unigram, bigram, trigram, tags]


sparse_file_names = [
                "tfidf_mat_unigram-1176b.npz",
                "tfidf_mat_bigram-1176b.npz",
                "tfidf_mat_trigram-1176b.npz"]
preprocessed_dataset_name = "Text_Preprocessing-1176b.csv"

sparse_unigram, sparse_bigram, sparse_trigram, tags = \
                    get_sparse_n_tags(sparse_file_names, preprocessed_dataset_name)
```

```
# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = sparse_unigram.tolist()

X = (data_preprocessed)
print ("Feature Extraction selesai")

# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=tags

X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)


from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

A-12 Script Klasifikasi Hyperparameter SVM Dengan Seleksi Feature MI

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

from scipy import sparse

import pandas as pd
import pickle
import numpy as np

import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score


# ---------------------- Get Sparse Matrix & Tags , Unigram, BIgram and Trigram ----------------------
def get_sparse_n_tags(file_names, dataset_name):

    # load preprocessed dataset as dataframe
    LABEL_DATA = pd.read_csv(dataset_name, encoding = "ISO-8859-1", usecols=["sentimen"])
    tags = LABEL_DATA.sentimen

    # load sparse matrix
    unigram = sparse.load_npz(file_names[0]).toarray()
    bigram = sparse.load_npz(file_names[1]).toarray()
    trigram = sparse.load_npz(file_names[2]).toarray()

    return [unigram, bigram, trigram, tags]


sparse_file_names = [
                "tfidf_mat_selection_unigram-1176b.npz",
                "tfidf_mat_selection _bigram-1176b.npz",
                "tfidf_mat_selection _trigram-1176b.npz"]
preprocessed_dataset_name = "Text_Preprocessing-1176b.csv"

sparse_unigram, sparse_bigram, sparse_trigram, tags = \
                    get_sparse_n_tags(sparse_file_names, preprocessed_dataset_name)
```

```
# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = sparse_unigram.tolist()

X = (data_preprocessed)
print ("Feature Extraction selesai")

# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=tags

X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)


from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['linear']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)

grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

A-13 Script 10-fold Cross Validation

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=10, kernel='rbf', gamma=0.01)
SVM.fit(X_train, y_train)
# predict the labels on validation dataset
predictions_SVM = grid.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)

# Untuk lebih menyakinkan lagi, kita juga bisa melakukan "Cross Validation"

from sklearn.metrics import accuracy_score

from sklearn.model_selection import KFold

kf = KFold(n_splits=10,random_state=77,shuffle=True)

accuracies = []

y = classes

for iteration, data in enumerate(kf.split(X), start=1):

    data_train   = X[data[0]]
    target_train = y[data[0]]

    data_test    = X[data[1]]
    target_test  = y[data[1]]

    SVM = svm.SVC(C=10, kernel='rbf', gamma=0.01)
    SVM.fit(X_train, y_train)

    preds = grid.predict(data_test)

    # for the current fold only
    accuracy = accuracy_score(target_test,preds)

    print("Training ke-", iteration)
    print(classification_report(target_test,preds))
    print("==================================================")

    accuracies.append(accuracy)

# this is the average accuracy over all folds
average_accuracy = np.mean(accuracies)

print()
print()
print()
print("Rata-rata Accuracy: ", average_accuracy)
```

A-14 Script ROC AUC

```
import matplotlib
from matplotlib import pyplot as plt

from sklearn.metrics import roc_curve, auc

y_prob = grid.predict_proba(X_test)

macro_roc_auc_ovo = roc_auc_score(y_test, y_prob, multi_class="ovo",
                                  average="macro")
weighted_roc_auc_ovo = roc_auc_score(y_test, y_prob, multi_class="ovo",
                                     average="weighted")
macro_roc_auc_ovr = roc_auc_score(y_test, y_prob, multi_class="ovr",
                                  average="macro")
weighted_roc_auc_ovr = roc_auc_score(y_test, y_prob, multi_class="ovr",
                                     average="weighted")
print("One-vs-One ROC AUC scores:\n{:.6f} (macro),\n{:.6f} "
      "(weighted by prevalence)"
      .format(macro_roc_auc_ovo, weighted_roc_auc_ovo))
print("One-vs-Rest ROC AUC scores:\n{:.6f} (macro),\n{:.6f} "
      "(weighted by prevalence)"
      .format(macro_roc_auc_ovr, weighted_roc_auc_ovr))
```

A-15 Script Model Prediksi

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
import pickle
import re

# Mengambil dataset twitter berbahasa indonesia
datareview = pd.read_csv('Text_Preprocessing-1176b.csv')


from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

# Untuk melakukan Feature Extraction, kita menggunakan library "Sklearn atau scikit-learn".
# Sklearn adalah library untuk melakukan task-task Machine Learning.
# "CountVectorizer" merupakan salah satu modul untuk melakukan "BoW"

from sklearn.feature_extraction.text import CountVectorizer

# Kita simpan teks ke dalam sebuah variabel
data_preprocessed = datareview.text_clean_stemmed.tolist()

# Kita proses Feature Extraction
count_vect = CountVectorizer()
count_vect.fit(data_preprocessed)

pickle.dump(count_vect, open("feature.p", "wb"))

X = count_vect.transform(data_preprocessed)
print ("Feature Extraction selesai")
```

```
# Selanjutnya kita split dataset menjadi data training dan data testing
# "train_test_split" merupakan salah satu modul untuk melakukan split dataset
from sklearn.model_selection import train_test_split

# Kita simpan sentiment (class) ke dalam sebuah variabel
classes=datareview.sentimen

X_train, X_test, y_train, y_test = train_test_split(X, classes, test_size = 0.2, random_state=77)

# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=10, kernel='rbf', gamma=0.01)
SVM.fit(X_train, y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(X_test)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, y_test)*100)

prediksi = pd.read_csv('Text_Preprocessing-prediksi.csv')

ubah = prediksi.text_clean_stemmed



# Model yang sudah jadi bisa kita gunakan seperti ini

text = count_vect.transform(ubah)
# String ke Angka

# Kita prediksi sentimennya
result = SVM.predict(text)[0:1674]
print("Sentiment:")
print()
print(result)

df = pd.DataFrame(result, index=result, columns=["result"])

df.to_excel("hasil-prediksi-1673-fix.xlsx")
```

A-16 Script Mencari Top Word

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
import pickle
import re

# Mengambil dataset twitter berbahasa indonesia
hasilprediksi = pd.read_json('hasil-prediksi-fix.json')

import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score

Corpus = pd.read_json('hasil-prediksi-fix.json')

Train_X, Test_X, Train_Y, Test_Y =
model_selection.train_test_split(Corpus['text_clean_stemmed'],Corpus['result'],test_size=0.2,
random_state = 77)
```

```
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)

Tfidf_vect = TfidfVectorizer(max_features=1000)
Tfidf_vect.fit(Corpus['text_clean_stemmed'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)

topkata = Corpus.text_clean_stemmed

import re
import numpy as np
import pandas as pd
from sklearn import preprocessing, model_selection, naive_bayes, svm
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score
import matplotlib
from matplotlib import pyplot as plt

# extracting the number of examples of each class
positif_len = Corpus[Corpus['result'] == 'positif'].shape[0]
negatif_len = Corpus[Corpus['result'] == 'negatif'].shape[0]
netral_len = Corpus[Corpus['result'] == 'netral'].shape[0]

plt.bar(10,positif_len,3, label="positif")
plt.bar(15,negatif_len,3, label="negatif")
plt.bar(20,netral_len,3, label="netral")
plt.legend()
plt.ylabel('Number of examples')
plt.title('Proportion of examples')
plt.show()


positif_data = Corpus[Corpus['result'] == 'positif']
negatif_data = Corpus[Corpus['result'] == 'negatif']
netral_data = Corpus[Corpus['result'] == 'netral']

freq = pd.Series((
' '.join(positif_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending = False)
top_vocab = freq.head(50)
top_vocab.plot(kind = 'barh', figsize=(5,7), xlim= (0, 40), title = 'Top Word kategori Positif')
```

```
freq = pd.Series((' '.join(negatif_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending  = False)
top_vocab = freq.head(50)
top_vocab.plot(kind = 'barh', figsize=(5,7), xlim=  (0, 40), title = 'Top Word kategori  Negatif')

freq = pd.Series((' '.join(netral_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending  = False)
top_vocab = freq.head(50)
top_vocab.plot(kind = 'barh', figsize=(5,7), xlim=  (0, 40), title = 'Top Word kategori  Netral')

df2 = pd.DataFrame(positif_data)

df2.to_csv("positif_data_1673-fix.csv")

df = pd.DataFrame(negatif_data)

df.to_csv("negatif_data_1673-fix.csv")

df1 = pd.DataFrame(netral_data)

df1.to_csv("netral_data_1673-fix.csv")

freq = pd.Series((' '.join(negatif_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending  = False)
top_vocab = freq.head(50)

dt = pd.DataFrame(freq)

dt.to_csv("top-negatif-data-fix.csv")

dt.to_excel("top-negatif-data-fix.xlsx")

freq2 = pd.Series((' '.join(positif_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending = False)
top_vocab1 = freq.head(50)

dt2 = pd.DataFrame(freq2)

dt2.to_excel("top-positif-data-fix.xlsx")

dt2.to_csv("top-positif-data-fix.csv")

freq3 = pd.Series((' '.join(netral_data['text_clean_stemmed']).split())).value_counts().sort_values(ascending = False)
top_vocab2 = freq.head(50)

dt3 = pd.DataFrame(freq3)

dt3.to_excel("top-netral-data-fix.xlsx")

dt3.to_csv("top-netral-data-fix.csv")
```

A-17 Script Asosiasi Kelas Negatif

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag

import string
import nltk
import re #regex library

# import word_tokenize & FreqDist from NLTK
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

documents = pd.read_csv('negatif_data_1673-fix.csv')

# import word_tokenize & FreqDist from NLTK
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

# import word_tokenize & FreqDist from NLTK
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
# NLTK word rokenize
def word_tokenize_wrapper(string):
   return word_tokenize(string)

documents["text_clean_stemmed1"] =
documents["text_clean_stemmed"].apply(word_tokenize_wrapper)

print('Tokenizing Result : \n')
print(documents["text_clean_stemmed1"].head())
print('\n\n\n')

unlist_comments = [item for items in documents["text_clean_stemmed1"] for item in items]


bigrams = nltk.collocations.BigramAssocMeasures()
trigrams = nltk.collocations.TrigramAssocMeasures()


bigramFinder = nltk.collocations.BigramCollocationFinder.from_words(unlist_comments)
trigramFinder = nltk.collocations.TrigramCollocationFinder.from_words(unlist_comments)
```

```
bigram_freq = bigramFinder.ngram_fd.items()

bigramFreqTable = pd.DataFrame(list(bigram_freq),
columns=['bigram','freq']).sort_values(by='freq', ascending=False)

bigramFreqTable.head().reset_index(drop=True)


trigram_freq = trigramFinder.ngram_fd.items()

trigramFreqTable = pd.DataFrame(list(trigram_freq),
columns=['trigram','freq']).sort_values(by='freq', ascending=False)

trigramFreqTable.head().reset_index(drop=True)

freq_bi = bigramFreqTable[:].bigram.values

freq_tri = trigramFreqTable[:].trigram.values

bigramPMITable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.pmi)),
columns=['bigram','PMI']).sort_values(by='PMI', ascending=False)


trigramPMITable = pd.DataFrame(list(trigramFinder.score_ngrams(trigrams.pmi)),
columns=['trigram','PMI']).sort_values(by='PMI', ascending=False)

pmi_bi = bigramPMITable[:].bigram.values

pmi_tri = trigramPMITable[:].trigram.values

#t-test

bigramTtable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.student_t)),
columns=['bigram','t']).sort_values(by='t', ascending=False)

trigramTtable = pd.DataFrame(list(trigramFinder.score_ngrams(trigrams.student_t)),
columns=['trigram','t']).sort_values(by='t', ascending=False)

t_bi = bigramTtable[:].bigram.values

t_tri = trigramTtable[:].trigram.values
```

```
df = pd.DataFrame(trigramTtable)

df.to_csv("asosiasi-negatif-data-fix.csv")

df.to_excel("asosiasi-negatif-data-fix.xlsx")

df1 = pd.DataFrame(t_tri)

df1.to_csv("asosiasi-negatif-data-fix-1.csv")

df1.to_excel("asosiasi-negatif-data-fix-1.xlsx")

#chi square

bigramChiTable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.chi_sq)), columns=['bigram','chi-sq']).sort_values(by='chi-sq', ascending=False)

trigramChiTable = pd.DataFrame(list(trigramFinder.score_ngrams(trigrams.chi_sq)), columns=['trigram','chi-sq']).sort_values(by='chi-sq', ascending=False)

chi_bi = bigramChiTable[:].bigram.values

chi_tri = trigramChiTable[:].trigram.values

#likelihood

bigramLikTable = pd.DataFrame(list(bigramFinder.score_ngrams(bigrams.likelihood_ratio)), columns=['bigram','likelihood ratio']).sort_values(by='likelihood ratio', ascending=False)

trigramLikTable = pd.DataFrame(list(trigramFinder.score_ngrams(trigrams.likelihood_ratio)), columns=['trigram','likelihood ratio']).sort_values(by='likelihood ratio', ascending=False)

lik_bi = bigramLikTable[:].bigram.values

lik_tri = trigramLikTable[:].trigram.values

#comparison bigram

bigramsCompare = pd.DataFrame([freq_bi, pmi_bi, t_bi, chi_bi, lik_bi]).T

bigramsCompare.columns = ['Frequency With Filter', 'PMI', 'T-test With Filter', 'Chi-Sq Test', 'Likeihood Ratio Test With Filter']

bigramsCompare.to_excel("asosiasi-negatif-bigram.xlsx")

#compare triagram

trigramsCompare = pd.DataFrame([freq_tri, pmi_tri, t_tri, chi_tri, lik_tri]).T

trigramsCompare.columns = ['Frequency With Filter', 'PMI', 'T-test With Filter', 'Chi-Sq Test', 'Likeihood Ratio Test With Filter']

trigramsCompare.to_excel("asosiasi-negatif-trigram.xlsx")
```

A-18 Script Scraping Twitter dan Youtube (Pada Prompt)

```
#Twitter Scraper

python -m pip install twitterscraper

twitterscraper "ventela" --output data-teview.csv --limit 1000000 --begindate 2018-03-01 --enddate 2020-03-01 –
csv

#Getoldtweet

!pip install GetOldTweets3

import GetOldTweets3 as got
import csv

# Script penyimpanan di atas perlu di-custom untuk disesuaikan sesuai kebutuhan kita

# Pertama kita setup parameternya seperti ini:
tweetCriteria = got.manager.TweetCriteria().setQuerySearch('converse')\
                                .setMaxTweets(1000000)\
                                .setNear('Jakarta')\
                                 .setTopTweets(False)\
                                .setSince("2018-03-02")\
                                .setUntil("2020-07-30")

# Setelah itu kita ambil datanya dengan menjalankan script di bawah:
data = got.manager.TweetManager.getTweets(tweetCriteria)

# Membuat file csv dengan nama "data.csv"
with open('data.csv','w', encoding="utf-8") as file:
   writer = csv.writer(file)

   # Definisikan nama kolom
   writer.writerow(["text"])

   # Loop data yang sudah diambil tadi
   for d in data:
      writer.writerow ([d.text])

   print("Data tersimpan!")

#Youtube Scraper

npm install -g youtube-comment-scraper-cli

youtube-comment-scraper –f csv –outputFile (nama file.csv) <VideoID>
```

A-19 Proses Pengelompokkan Data

Kriteria/Fitur	VENTELA	
DESAIN PRODUK	('buat', 'desain', 'sendiri')	2,22860361
	('punya', 'desain', 'sendiri')	1,730218524
	('butuh', 'buat', 'desain')	1,413071414
	('produk', 'indonesia', 'desain')	1,411637026
	('cari', 'buat', 'desain')	1,406218523
	('buat', 'desain', 'baru')	1,404314943
TINGKAT ORISINILITAS PRODUK	('jiplak', 'merek', 'luar')	1,728856466
	('jiplak', 'jiplak', 'jiplak')	1,725674566
	('jiplak', 'karya', 'orang')	1,413698791
	('jiplak', 'vans', 'kw')	1,410019131
KEMIRIPAN DENGAN PRODUK YANG ADA DI PASARAN	('mirip', 'merek', 'luar')	1,729205842
	('lebih', 'tepat', 'mirip')	1,413938482
	('bilang', 'mirip', 'compass')	1,411792172
	('mirip', 'sangat', 'dengan')	1,411279367
KETERSEDIAAN UKURAN PERLU DITINGKATKAN	('temu', 'ukur', 'kemarin')	1,414160477
	('temu', 'temu', 'ukur')	1,414107391
	('ukur', 'kemarin', 'umpat')	1,414080848
	('ukur', 'panjang', 'centimeter')	1,413800673
	('cari', 'ukur', 'warna')	1,413532295
	('benar', 'cari', 'ukur')	1,412851028
	('tidak', 'ada', 'ukur')	1,728751821
	('susah', 'cari', 'ukur')	1,412417494
	('ukur', 'susah', 'benar')	1,412331967
	('ukur', 'ukur', 'susah')	1,405746382
MERUBAH SIZING	('harus', 'turun', 'ukur')	1,414036661
	('biasa', 'pakai', 'ukur')	1,730598772

Kriteria/Fitur	VENTELA	
MEMBUAT UKURAN BESAR	('besar', 'ukur', 'besar')	1,411001873
	('ukur', 'besar', 'kaki')	1,410223282
	('ukur', 'kaki', 'besar')	1,410223282
	('jarang', 'ukur', 'besar')	1,413726943
	('kasihan', 'kaki', 'besar')	1,414012033
	('kaki', 'besar', 'seperti')	1,412440104
	('ukur', 'besar', 'ukur')	1,404578495
	('buat', 'ukur', 'besar')	1,400393566
TIRU VANS	('mirip', 'vans', 'old')	1,413615856
	('besar', 'mirip', 'vans')	1,409282484
	('mirip', 'sangat', 'vans')	1,404650259
	('pakai', 'vans', 'kw')	1,409822517
	('ikut', 'converse', 'vans')	1,40953415
	('vans', 'dengan', 'converse')	1,406781554
	('vans', 'kamu', 'terlalu')	1,731879571
	('vans', 'old', 'sekolah')	1,414150646
	('terlalu', 'vans', 'kamu')	1,414003841
	('tiru', 'vans', 'jadi')	1,411020551
BANGGA PRODUK LOKAL	('judul', 'merek', 'lokal')	1,413581895
	('kata', 'lokal', 'bangga')	1,413016719
	('bangga', 'dengan', 'produk')	1,412867114
	('merek', 'lokal', 'harus')	1,412634395
	('produk', 'lokal', 'tapi')	1,412495871
	('dengan', 'produk', 'lokal')	1,411894679
	('lokal', 'bangga', 'karena')	1,410922244
	('merek', 'lokal', 'mudah')	1,410107726
	('cinta', 'merek', 'lokal')	1,409476059
	('lokal', 'bangga', 'lebih')	1,406134873
	('bangga', 'lokal', 'bangga')	1,406134873

Kriteria/Fitur	VENTELA	
	('merek', 'lokal', 'punya')	1,405686057
	('banyak', 'merek', 'lokal')	1,40505439
	('seperti', 'merek', 'lokal')	1,400316887
	('bilang', 'bangga', 'lokal')	1,729241305
MODEL PRODUK	('buat', 'model', 'baru')	1,411683915
	('lebih', 'bagus', 'buat')	1,72776103
	('saran', 'kritik', 'buat')	1,412995271
	('saran', 'mau', 'buat')	1,412453808
MIRIP CONVERSE	('seperti', 'converse', 'kw')	1,410772818
	('sangat', 'dengan', 'converse')	1,408808466
PRODUK INDONESIA	('cinta', 'produk', 'indonesia')	1,731700185
	('hak', 'cipta', 'indonesia')	1,414198787
	('bangga', 'produk', 'indonesia')	1,412667664
LEBIH BAIK DARI COMPASS	('tidak', 'seperti', 'compass')	1,992127336
KETERSEDIAAN WARNA PERLU DITINGKATKAN	('public', 'kuning', 'hijau')	1,41420677
	('cari', 'public', 'kuning')	1,414177904
	('kuning', 'hijau', 'tidak')	1,414164588
	('hijau', 'tidak', 'temu')	1,414115613
KETERSEDIAAN PRODUK PERLU DITINGKATKAN	('dapat', 'tidak', 'seperti')	1,410442506
	('mudah', 'dapat', 'tidak')	1,413099387
	('susah', 'benar', 'cari')	1,413814436
	('sekarang', 'mulai', 'susah')	1,414078792
	('tidak', 'temu', 'temu')	1,414066638
PENJUALAN PRODUK PERLU DITINGKATKAN	('juta', 'pasang', 'tahun')	1,414213086
	('produksi', 'juta', 'pasang')	1,414210941
MEMBENTUK IDENTITAS PRODUK	('lokal', 'bangga', 'umpat')	1,411969482
	('buat', 'merek', 'lokal')	1,695432273
	('ciri', 'khas', 'sendiri')	1,732023663
	('bukan', 'lokal', 'bangga')	1,410174218

Kriteria/Fitur	VENTELA	
	('tidak', 'tahu', 'merek')	1,409560961
	('pakai', 'tidak', 'tahu')	1,411479139
	('orang', 'tidak', 'tahu')	1,412989194
	('jadi', 'tahu', 'bagaimana')	1,413975838
	('merek', 'tidak', 'punya')	1,401651538
	('tidak', 'punya', 'ide')	1,41377279
HARGA PRODUK	('converse', 'mahal', 'karena')	1,413387784
	('jangan', 'bilang', 'murah')	1,413966901
KETAHANAN MATERIAL	('cuci', 'jadi', 'tahu')	1,414188539
	('jadi', 'tidak', 'enak')	1,413013681
DESIGN SILUET	('buat', 'siluet', 'baru')	1,412013869
DESIGN LOGO	('tidak', 'suka', 'logo')	1,412303547
BAGIAN KAKI DEPAN	('kaki', 'bagi', 'depan')	1,414092645
PERLU PROMOSI PRODUK	('kenal', 'banyak', 'orang')	1,413902555
PENGGUNAAN DENGAN KAOS KAKI	('pakai', 'kaos', 'kaki')	1,414049896
HAK CIPTA	('paten', 'hak', 'cipta')	1,414212132

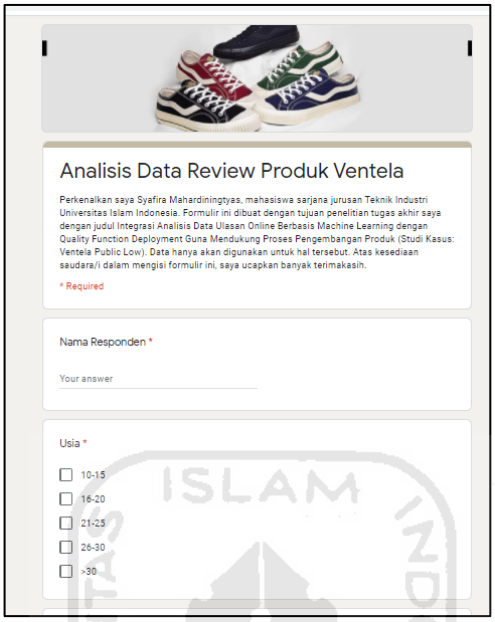
A-20 Proses Pencocokkan Data

Ventela	Converse
('buat', 'desain', 'sendiri')	('makin', 'buluk', 'makin')
('punya', 'desain', 'sendiri')	('makin', 'buluk', 'makin')
('butuh', 'buat', 'desain')	('makin', 'buluk', 'makin')
('produk', 'indonesia', 'desain')	('makin', 'buluk', 'makin')
('cari', 'buat', 'desain')	('makin', 'buluk', 'makin')
('buat', 'desain', 'baru')	('makin', 'buluk', 'makin')
('jiplak', 'merek', 'luar')	('makin', 'buluk', 'makin')
('jiplak', 'jiplak', 'jiplak')	('makin', 'buluk', 'makin')
('jiplak', 'karya', 'orang')	('makin', 'buluk', 'makin')
('jiplak', 'vans', 'kw')	('makin', 'buluk', 'makin')
('mirip', 'merek', 'luar')	('makin', 'buluk', 'makin')
('lebih', 'tepat', 'mirip')	('beli', 'chuck', 'taylor')
('bilang', 'mirip', 'compass')	('makin', 'buluk', 'makin')
('mirip', 'sangat', 'dengan')	('pakai', 'kira', 'ventela')
('temu', 'ukur', 'kemarin')	('ukur', 'tidak', 'ingin')
('temu', 'temu', 'ukur')	('ukur', 'tidak', 'ingin')
('ukur', 'kemarin', 'umpat')	('ukur', 'tidak', 'ingin')
('ukur', 'panjang', 'centimeter')	('ukur', 'tidak', 'ingin')
('cari', 'ukur', 'warna')	('beli', 'chuck', 'taylor')
('benar', 'cari', 'ukur')	('ukur', 'tidak', 'ingin')
('tidak', 'ada', 'ukur')	('ukur', 'tidak', 'ingin')
('susah', 'cari', 'ukur')	('ukur', 'tidak', 'ingin')
('ukur', 'susah', 'benar')	('ukur', 'tidak', 'ingin')
('ukur', 'ukur', 'susah')	('ukur', 'tidak', 'ingin')
('harus', 'turun', 'ukur')	('ukur', 'tidak', 'ingin')
('biasa', 'pakai', 'ukur')	('ukur', 'tidak', 'ingin')
('besar', 'ukur', 'besar')	('ukur', 'tidak', 'ingin')
('ukur', 'besar', 'kaki')	('ukur', 'tidak', 'ingin')
('ukur', 'kaki', 'besar')	('ukur', 'tidak', 'ingin')
('jarang', 'ukur', 'besar')	('ukur', 'tidak', 'ingin')
('kasihan', 'kaki', 'besar')	('ukur', 'tidak', 'ingin')
('kaki', 'besar', 'seperti')	('ukur', 'tidak', 'ingin')
('ukur', 'besar', 'ukur')	('ukur', 'tidak', 'ingin')
('buat', 'ukur', 'besar')	('ukur', 'tidak', 'ingin')
('mirip', 'vans', 'old')	('makin', 'buluk', 'makin')
('besar', 'mirip', 'vans')	('ingin', 'sangat', 'beli')
('mirip', 'sangat', 'vans')	('makin', 'buluk', 'makin')
('pakai', 'vans', 'kw')	('makin', 'buluk', 'makin')
('ikut', 'converse', 'vans')	('pakai', 'kira', 'ventela')
('vans', 'dengan', 'converse')	('pakai', 'kira', 'ventela')

Ventela	Converse
('vans', 'kamu', 'terlalu')	('lebih', 'beli', 'vans')
('vans', 'old', 'sekolah')	('adidas', 'nike', 'vans')
('terlalu', 'vans', 'kamu')	('beli', 'chuck', 'taylor')
('tiru', 'vans', 'jadi')	('makin', 'buluk', 'makin')
('judul', 'merek', 'lokal')	('ventela', 'merek', 'lokal')
('kata', 'lokal', 'bangga')	('ventela', 'merek', 'lokal')
('bangga', 'dengan', 'produk')	('ventela', 'merek', 'lokal')
('merek', 'lokal', 'harus')	('ventela', 'merek', 'lokal')
('produk', 'lokal', 'tapi')	('ventela', 'merek', 'lokal')
('dengan', 'produk', 'lokal')	('ventela', 'merek', 'lokal')
('lokal', 'bangga', 'karena')	('ventela', 'merek', 'lokal')
('merek', 'lokal', 'mudah')	('ventela', 'merek', 'lokal')
('cinta', 'merek', 'lokal')	('ventela', 'merek', 'lokal')
('lokal', 'bangga', 'lebih')	('ventela', 'merek', 'lokal')
('bangga', 'lokal', 'bangga')	('ventela', 'merek', 'lokal')
('merek', 'lokal', 'punya')	('ventela', 'merek', 'lokal')
('banyak', 'merek', 'lokal')	('ventela', 'merek', 'lokal')
('seperti', 'merek', 'lokal')	('ventela', 'merek', 'lokal')
('bilang', 'bangga', 'lokal')	('ventela', 'merek', 'lokal')
('buat', 'model', 'baru')	('makin', 'buluk', 'makin')
('lebih', 'bagus', 'buat')	('makin', 'buluk', 'makin')
('saran', 'kritik', 'buat')	('makin', 'buluk', 'makin')
('saran', 'mau', 'buat')	('makin', 'buluk', 'makin')
('seperti', 'converse', 'kw')	('pakai', 'kira', 'ventela')
('sangat', 'dengan', 'converse')	('pakai', 'kira', 'ventela')
('cinta', 'produk', 'indonesia')	('ventela', 'merek', 'lokal')
('hak', 'cipta', 'indonesia')	('makin', 'buluk', 'makin')
('bangga', 'produk', 'indonesia')	('ventela', 'merek', 'lokal')
('tidak', 'seperti', 'compass')	('makin', 'buluk', 'makin')
('public', 'kuning', 'hijau')	('ingin', 'sangat', 'beli')
('cari', 'public', 'kuning')	('ingin', 'sangat', 'beli')
('kuning', 'hijau', 'tidak')	('ingin', 'sangat', 'beli')
('hijau', 'tidak', 'temu')	('ingin', 'sangat', 'beli')
('dapat', 'tidak', 'seperti')	('ingin', 'sangat', 'beli')
('mudah', 'dapat', 'tidak')	('ingin', 'sangat', 'beli')
('susah', 'benar', 'cari')	('ingin', 'sangat', 'beli')
('sekarang', 'mulai', 'susah')	('beli', 'sport', 'station')
('tidak', 'temu', 'temu')	('ingin', 'sangat', 'beli')
('juta', 'pasang', 'tahun')	('ingin', 'sangat', 'beli')
('produksi', 'juta', 'pasang')	('ingin', 'sangat', 'beli')
('lokal', 'bangga', 'umpat')	('makin', 'buluk', 'makin')
('buat', 'merek', 'lokal')	('makin', 'buluk', 'makin')
('ciri', 'khas', 'sendiri')	('makin', 'buluk', 'makin')
('bukan', 'lokal', 'bangga')	('pakai', 'kira', 'ventela')

Ventela	Converse
('tidak', 'tahu', 'merek')	('pakai', 'kira', 'ventela')
('pakai', 'tidak', 'tahu')	('pakai', 'kira', 'ventela')
('orang', 'tidak', 'tahu')	('pakai', 'kira', 'ventela')
('jadi', 'tahu', 'bagaimana')	('pikir', 'lebih', 'beli')
('merek', 'tidak', 'punya')	('makin', 'buluk', 'makin')
('tidak', 'punya', 'ide')	('makin', 'buluk', 'makin')
('converse', 'mahal', 'karena')	('harga', 'jual', 'gila')
('jangan', 'bilang', 'murah')	('harga', 'jual', 'gila')
('cuci', 'jadi', 'tahu')	('makin', 'buluk', 'makin')
('jadi', 'tidak', 'enak')	('makin', 'buluk', 'makin')
('buat', 'siluet', 'baru')	('makin', 'buluk', 'makin')
('tidak', 'suka', 'logo')	('makin', 'buluk', 'makin')
('kaki', 'bagi', 'depan')	('kaos', 'kaki', 'lumayan')
('kenal', 'banyak', 'orang')	('kaos', 'putih', 'hitam')
('pakai', 'kaos', 'kaki')	('kaos', 'kaki', 'lumayan')
('paten', 'hak', 'cipta')	('makin', 'buluk', 'makin')

A-21 Kuesioner



Analisis Data Review Produk Ventela

Perkenalkan saya Syafira Mahardiningtyas, mahasiswa sarjana jurusan Teknik Industri Universitas Islam Indonesia. Formulir ini dibuat dengan tujuan penelitian tugas akhir saya dengan judul Integrasi Analisis Data Ulasan Online Berbasis Machine Learning dengan Quality Function Deployment Guna Mendukung Proses Pengembangan Produk (Studi Kasus: Ventela Public Low). Data hanya akan digunakan untuk hal tersebut. Atas kesediaan saudara/i dalam mengisi formulir ini, saya ucapkan banyak terimakasih.

* Required

Nama Responden *

Your answer

Usia *

- 10-15
- 16-20
- 21-25
- 26-30
- >30

Bagian 1 Pertanyaan Pada Kuesioner



Media sosial apa yang sangat sering anda gunakan untuk MENULISKAN atau MENUANGKAN review atau pendapat anda mengenai produk Ventela? *

- Twitter
- Instagram
- Facebook
- Youtube
- Tiktok
- WhatsApp
- Line
- Snapchat
- Tumblr
- Pinterest
- Blog

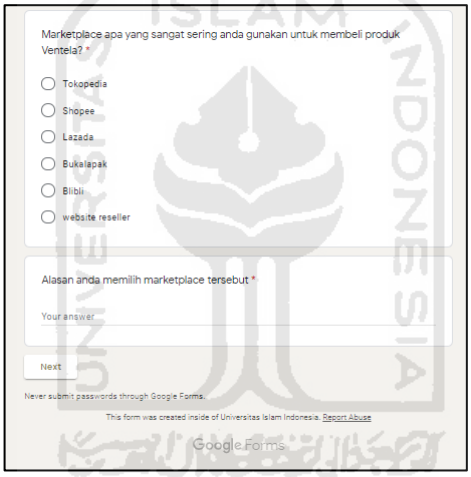
Alasan anda memilih media sosial tersebut *

Your answer

Bagian 2 Pertanyaan Pada Kuesioner



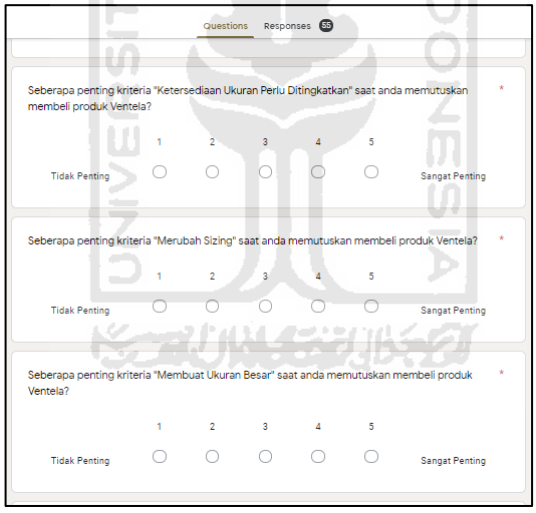
Bagian 3 Pertanyaan Pada Kuesioner



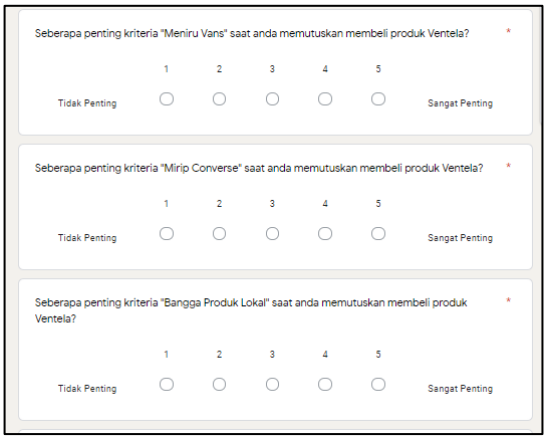
Bagian 4 Pertanyaan Pada Kuesioner



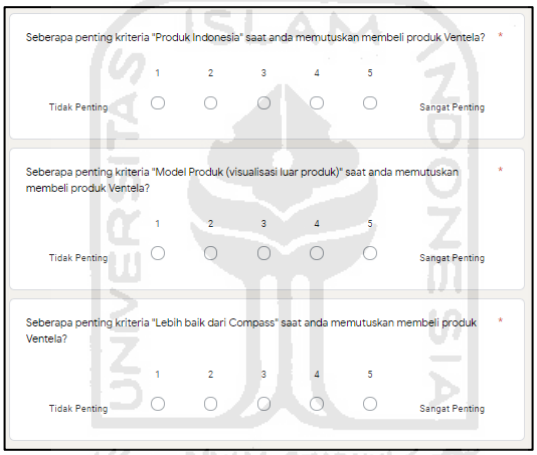
Bagian 5 Pertanyaan Pada Kuesioner



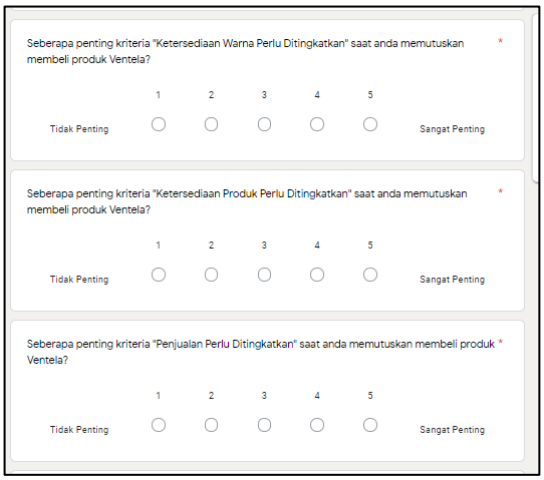
Bagian 6 Pertanyaan Pada Kuesioner



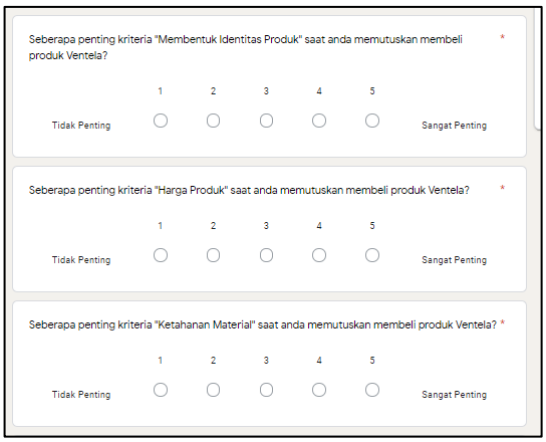
Bagian 7 Pertanyaan Pada Kuesioner



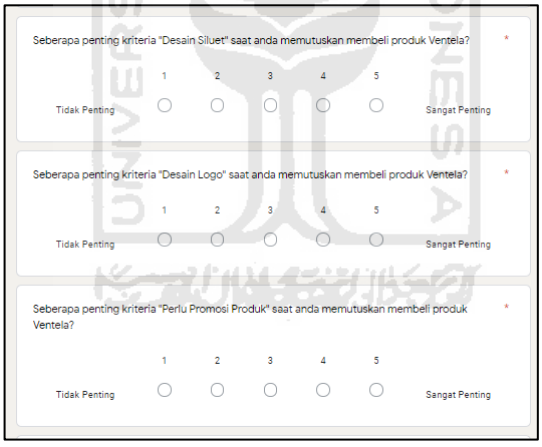
Bagian 8 Pertanyaan Pada Kuesioner



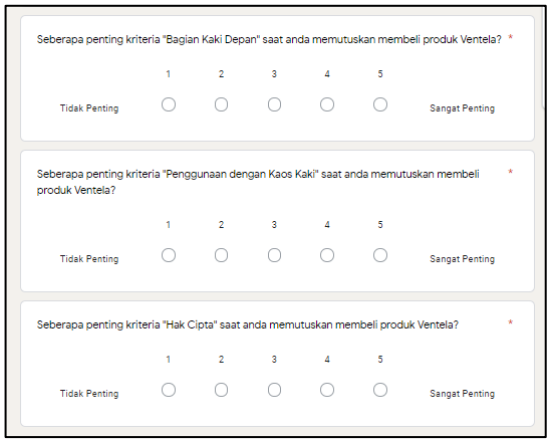
Bagian 9 Pertanyaan Pada Kuesioner



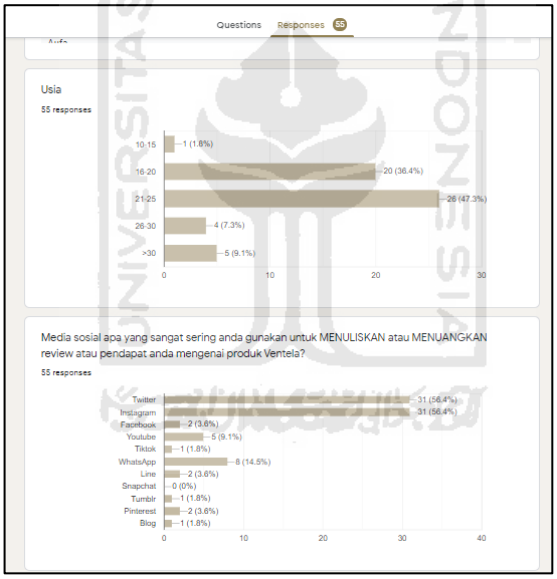
Bagian 10 Pertanyaan Pada Kuesioner



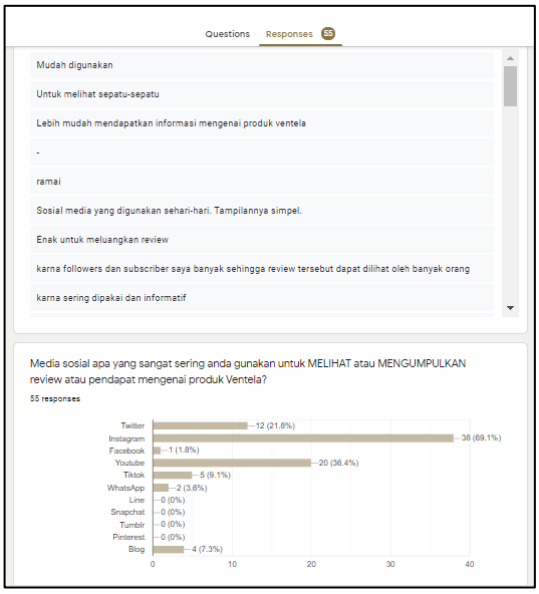
Bagian 11 Pertanyaan Pada Kuesioner



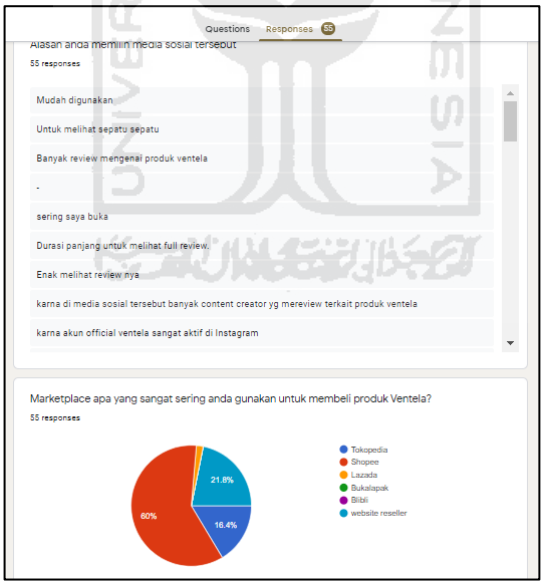
Bagian 12 Pertanyaan Pada Kuesioner



Bagian 1 Respon Pada Kuesioner



Bagian 2 Respon Pada Kuesioner



Bagian 3 Respon Pada Kuesioner



Bagian 4 Respon Pada Kuesioner