IMPLEMENTASI METODE
EXTREME GRADIENT BOOSTING (XGBOOST)
UNTUK KLASIFIKASI PADA DATA BIOINFORMATIKA

(Studi Kasus : Penyakit Ebola, GSE122692)

TUGAS AKHIR



Gregy Addis Shafila

16611022

JURUSAN STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAUHAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2020

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul	:	Implementasi Metode *Extreme Gradient Boosting (XGBoost)* Untuk Klasifikasi Pada Data Bioinformatika (Studi Kasus : Penyakit Ebola, GSE122692)
Nama Mahasiswa	:	Gregy Addis Shafila
Nomor Mahasiswa	:	16611022

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK

DIUJIKAN

Yogyakarta, 2 September 2020

Menyetujui,

Dosen Pembimbing I



Dr. techn. Rohmatul Fajriyah, S.Si., M.Si

HALAMAN PENGESAHAN

TUGAS AKHIR

IMPLEMENTASI METODE

***EXTREME GRADIENT BOOSTING* (*XGBOOST*)**

UNTUK KLASIFIKASI PADA DATA BIOINFORMATIKA

(Studi Kasus : Penyakit Ebola, GSE122692)

Nama Mahasiswa : Gregy Addis Shafila

Nomor Mahasiswa : 16611022

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL 2 SEPTEMBER 2020

	Nama Penguji	**Tanda Tangan**
1.	**Dr. techn. Rohmatul Fajriyah, S.Si., M.Si.**	.....
2.	**Ayundyah Kesumawati, S.Si., M.Si.**	.....
3.	**Dina Tri Utari, S.Si., M.Sc.**	.....

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Prof. Riyanto, S.Pd., M.Si., Ph.D.

KATA PENGANTAR

Asslamualaikum Wr.Wb

Alhamdulillahirobbilalamin, Puji Syukur kehadirat Allah SWT yang telah melimpahkan ramat dan hidayahnya selama melaksanakan Tugas Akhir sehingga dapat terlselesaikan. Tidak lupa shalawat serta salam tercurahkan kepada Nabi Muhammad SAW beserta keluarga dan para pengikutnya, sehingga dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi Metode *eXtreme Gradient Boosting (XGBoost)* Untuk Klasifikasi Data Penyakit Ebola, GSE122692" ini sebagai salah satu syarat untuk memperoleh gelar sarjana Program Studi Statistika di Universitas Islam Indonesia dengan baik.

Selama menulis Tugas Akhir, penulis telah banyak mendapat bimbingan serta bantuan dari berbagai pihak, baik itu berupa saran, kritik, motivasi, bimbingan serta bantuan lainnya. Oleh karena itu, pada kesempatan ini penulis bermaksud menyampaikan ucapan terimakasih kepada :

1. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia, Yogyakarta beserta seluruh jajarannya.
2. Bapak Dr. Edy, S.Si., M.Si., selaku Ketua Program Studi Statistika beserta seluruh jajarannya.
3. Ibu Dr. techn, Rohmatul Fajriyah, S.Si., M.Si., selaku Dosen Pembimbing 1 penulis yang sangat berjasa atas ilmu, saran, kritik, memberi motivasi dan bimbingan bagi penulis dalam penyelesaikan Tugas Akhir ini sehingga dapat selesai dengan baik.
4. Dosen-dosen Statistka Universitas Islam Indonesia yang telah mendidik dan memberikan ilmunya kepada penulis serta selalu menginspirasi.
5. Kedua Orang tua, ayahanda tercinta Sudarmo dan Ibunda tersayang Imih yang telah memberikan dukungan moril maupun materil kepada penulis

dalam hal apapun untuk menjadi yang terbaik serta doa yang tiada henti-hentinya kepada penulis.

6. Keluarga Bimbingan SBRC : Anas, Iqbal, Rima, Widia, Vivid, Rahma dan Revika yang selalu memberikan bantuan, kritk, saran, motivasi dan do'a dalam menyelesaikan Tugas Akhir ini.
7. Teman-teman Statistika UII angkatan 2016 (ARTCOS) yang selalu saling mendo'akan dan memotivasi selama masa perkuliahan selama 4 tahun ini.
8. Sahabaat UKM Basket FMIPA yang selalu memberikan kenyamanan baru setelah sibuk aktivitas kuliah, dan cerita-cerita yang terbentuk selama latihan.
9. Sahabat Hadroh Artcos yang selalu memberikan pelajaran dan pengalaman baru dalam hal kebaikan, yuk ngopi bareng lagi dan berkumpul lagi.
10. Sahabat Crocodile : Heri, Alif, Rafik, Fiqri, Fadhel dan Rananda yang selalu memberikan pengalaman suka dan duka selama di jogja, tidak ada uang dan akhir bulan, teman untuk jalan-jalan dan teman dikala gabut melanda, dan masih banyak hal baik yang dilakukan bersama.
11. Elfara Rahmadhika, yang telah menemani dan membantu memberikan semangat setiap harinya dalam penyelesaian skripsi ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna, oleh karena itu segala kritik dan saran yang bersifart membangun selalu penulis harapkan. Semoga Tugas Akhir ini dapat bermanfaat bagu penulis khusunya dan bagi semua yang membutuhkan. Akhir kata, semoga Allah SWT selalu melimpahkan rahmat serta hidayahnya kepada kita semua, Aamiin ya rabbal'alamin.

Wassalamu'alaikum. Wr.Wb

Yogyakarta, 20 Agustus 2020



Gregy Addis Shafila

DAFTAR ISI

HALAMAN PERSETUJUAN PEMBIMBING ii
HALAMAN PENGESAHAN iii
KATA PENGANTAR iv
DAFTAR ISI vi
DAFTAR TABEL viii
DAFTAR GAMBAR ix
DAFTAR LAMPIRAN x
PERNYATAAN xi
INTISARI xii
ABSTRACT xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 4
1.6 Sistematika Penelitian 5
BAB II TINJAUAN PUSTAKA 7
2.1 Penelitian Terkait Penyakit Ebola 7
2.2 Penelitian Terkait Metode *eXtreme Gradient Boosting* (*XGBoost*) 8
2.3 Penelitian dalam Bidang Bioinformatika Menggunakan Metode *eXtreme Gradient Boosting* (*XGBoost*) 10
BAB III DASAR TEORI 16
3.1 Bioinformatika 16
3.2 Ebola Viruses 17
3.3 *Gene Expression* 18
3.4 *Microarray* 18
3.5 *Oligo* 20
3.6 *Preprocessing* 20
3.7 *Filtering* 23

3.8 ***Feature selection*** 23
3.9 Uji Signifikansi Simultan (Uji Statistik F) 24
3.10 Analisis Deskriptif 25
3.11 ***Machine Learning*** 25
3.12 Klasifikasi 26
3.13 ***Ensemble Learning*** 27
3.13.1 ***Boosting*** 27
3.13.2 ***eXtreme Gradient Boosting*** **(*XGBoost*)** 29
3.13.3 ***Tunning Hyperparameter*** 34
3.13.4 ***Confusion Matrix*** 35
BAB IV METODOLOGI PENELITIAN 38
4.1 Data 38
4.2 Tempat dan Waktu Penelitian 38
4.3 Variabel dan Definisi Operasional Variabel 38
4.4 Metode Analisis Data 39
4.5 Langkah Penelitian 40
BAB V HASIL DAN PEMBAHASAN 42
5.1 Analisis Deskriptif 42
5.2 Pengolahan Data Bioinformatik 43
5.2.1 ***Visualization*** **dan** ***QC Tools*** 44
5.2.2 ***Preprocessing*** 46
5.2.3 ***Filtering*** 47
5.2.4 ***Feature Selection*** 48
5.2.5 Mengolah Data 48
5.3 Klasifikasi ***Boosting XGBoost*** 49
5.3.1 Model ***XGBoost*** 50
5.3.2 Prediksi ***XGBoost*** 51
5.3.3 ***Optimasi Hyperparameter*** 52
BAB VI PENUTUP 59
6.1 Kesimpulan 59
6.2 Saran 60
DAFTAR PUSTAKA 61
LAMPIRAN 65

DAFTAR TABEL

Tabel 2.1 Perbandingan penelitian 13
Tabel 3.1 Metode Pre-Processing 22
Tabel 3.2 Confusion Matrix 36
Tabel 3.3 Klasifikasi AUC 37
Tabel 4.1 Definisi Operasional Variabel 38
Tabel 5.1 Dataset pasien Ebola Virus 42
Tabel 5.2 Informasi Lainnya Pada Data 43
Tabel 5.3 Jumlah data setelah filtering non spesific filter 48
Tabel 5.4 Jumlah data untuk dianalisis 48
Tabel 5.5 Ratio Data Training dan Testing 49
Tabel 5.6 Parameter Model XGBoost 50
Tabel 5.7 Confussion Matrix Sebelum Optimasi Hyperparameter 51
Tabel 5.8 Overall statistic Sebelum Optimasi Hyperparameter 52
Tabel 5.9 Kombinasi Nilai Parameter Tunning Hyperparameter 53
Tabel 5.10 Hasil Nilai Parameter Tunning Hyperparameter 53
Tabel 5.11 Confusion Matrix Setelah Tunning Hyperparameter 54
Tabel 5.12 Overall Statistic Setelah Tunning Hyperparameter 56
Tabel 5.13 Variabel Importance 57

DAFTAR GAMBAR

Gambar 3.1 Interaksi disiplin ilmu yang berhubungan dengan Bioinformatika.. 16
Gambar 3.2 Langkah – Langkah dalam Percobaan Microarray 19
Gambar 3.3 Contoh Classification and Regression Trees (CART) 30
Gambar 3.4 Model Tree Ensemble 31
Gambar 3.5 Skor Struktur..... 32
Gambar 3.6 Struktur Blok..... 34
Gambar 4.1 Flow Chart 40
Gambar 5.1 Pie Chart Kelas Pasien Virus Ebola 42
Gambar 5.2 Pheno Data..... 44
Gambar 5.3 Pseudo-Image..... 44
Gambar 5.4 MA-Plot sebelum Normalisasi..... 45
Gambar 5.5 MA-Plot Sesudah Normalisasi..... 45
Gambar 5.6 Boxplot sebelum preprocessing 46
Gambar 5.7 Boxplot setelah preprocessing 47
Gambar 5.8 Logarithmic loss Training dan Testing 57
Gambar 5.9 Histogram Feature importance..... 58

DAFTAR LAMPIRAN

Lampiran 1 Syntaks Membaca data microarray 65
Lampiran 2 Melihat Phenodata 66
Lampiran 3 Syntaks Preprocessing 66
Lampiran 4 Syntaks Filtering 67
Lampiran 5 Syntaks Membuat label klasifikasi 67
Lampiran 6 Syntaks Feature Selection 67
Lampiran 7 Syntaks Membuat Data Training dan Testing 69
Lampiran 8 Syntaks Analisis XGBoost Cross Validation 70
Lampiran 9 Syntaks Analisis XGBoost dengan Tunning Hyperparameter 72
Lampiran 10 Output Hasil Preprocessing 73
Lampiran 11 Output Hasil Filtering 74
Lampiran 12 Output Hasil Feature Selection 74
Lampiran 13 Output Hasil Training dan Testing 75
Lampiran 14 Output Confusion matrix Sebelum Tunning Hyperparameter 76
Lampiran 15 Output Confusion matrix Sesudah Tunning Hyperparameter 76
Lampiran 16 Data Set 77
Lampiran 17 Data Training 77
Lampiran 18 Data Testing 77

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali diacu dalam naskah ini dan diterbitkan dalam daftar pustaka.

Yogyakarta, 2 September 2020



Gregy Addis Shafila

IMPLEMENTASI METODE

***EXTREME GRADIENT BOOSTING* (*XGBOOST*)**

UNTUK KLASIFIKASI PADA DATA BIOINFORMATIKA

(Studi Kasus : Penyakit Ebola, GSE122692)

Oleh : Gregy Addis Shafila

Program Studi Statistika Fakultas MIPA

Universitas Islam Indonesia

INTISARI

Bioinformatika mempunyai kajian yang tak lepas dari perkembangan biologi molekuler modern yang diketahui dengan kemampuan manusia untuk memahami genom. Pada bidang kesehatan, bioinformatika digunakan dalam penerapan diagnosis, prediksi maupun pencegahan sebuah penyakit salah satunya yaitu virus Ebola atau EBOV. Berdasarkan data, dari Maret 2014 hingga Maret 2016, Afrika barat menderita wabah virus Ebola (EBOV) terbesar hingga saat ini, dengan 28.652 kasus dan 11.325 kematian. Data microarray pasien virus Ebola yang terdiri dari sampel jaringan Fatalities, Viremic suvivors, Suvivors in recovery phase dan Healthy control yang diperoleh dari website National Center of Biotechnology Information (NCBI), platform GPL16686 - GSE 122692, digunakan dalam penelitian ini. Pengolahan data microarray, dimulai dengan preprocessing dan filtering. Machine learning banyak digunakan pada analisis data microarray, salah satu diantaranya adalah eXtreme Gradient Boosting (XGBoost). Pembagian data set Train dan Test dilakukan dengan ratio 80% : 20%,. Berdasarkan model yang dipilih, diperoleh nilai accuracy 69.23% dan Kappa 0.05702. Melalui *grid search didapatkan hasil nilai terbaik accuracy adalah 76.92% dan nilai Kappa 0.675 dengan nilai AUC\ sebesar 0,85.*

***Kata Kunci** : Bioinformatika, Ebola, Microarray, Klasifikasi, XGBoost, Tunning Hyperparameter, Grid search.*

IMPLEMENTATION OF

EXTREME GRADIENT BOOSTING (XGBOOST) METHOD FOR

CLASSIFICATION IN BIOINFORMATIC DATA

(*Case Study: Ebola Disease, GSE122692*)

Gregy Addis Shafila

Department of Statistics, Faculty Mathematics and Natural Sciences Islamic University of Indonesia

ABSTRACT

Bioinformatics has studies that cannot be separated from the development of modern molecular biology which is known by the human ability to understand the genome. In the health sector, bioinformatics is used in the application of diagnosis, prediction and prevention of a disease, one of which is the Ebola virus or EBOV. Based on data, from March 2014 to March 2016, West Africa suffered from the largest Ebola virus (EBOV) outbreak to date, with 28,652 cases and 11,325 deaths. Ebola virus patient microarray data consisting of tissue samples of Fatalities, Viremic suvivors, Suvivors in recovery phase and Healthy control obtained from the National Center of Biotechnology Information (NCBI) website, platform GPL16686 - GSE 122692, were used in this study. Microarray data processing, starting with preprocessing and filtering. Machine learning is widely used in microarray data analysis, one of which is eXtreme Gradient Boosting (XGBoost). The distribution of Train and Test data sets is carried out with a ratio of 80%: 20% ,. Based on the selected model, the accuracy value is 69.23% and the Kappa value is 0.05702. Through grid search, the best accuracy value is 76.92% and the Kappa value is 0.675 with an AUC value of 0.85.

Keywords *: Bioinformatics, Ebola, Microarray, Classification, XGBoost, Tunning Hyperparameter, Grid search.*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya teknologi komputasi dan algoritma belajar oleh mesin berimplikasi pada pesatnya teknologi diberbagai bidang termasuk dalam bidang penelitian bioinformatika. Teknologi ini telah banyak diimplementasikan pada banyak hal terkait genetika dan genomik karena dinilai mampu melakukan interprestasi *dataset* genom yang berukuran sangat besar dan telah digunakan untuk mendeskripsikan berbagai varietas yang luas dari bagian urutan genomik (Ranganathan, Nakai, & Schonbac, 2018). Bioinformatika mempunyai kajian yang tak lepas dari perkembangan biologi molekul *modern* yang diketahui dengan kemampuan manusia untuk memahami genom, yaitu cetak biru informasi genetik yang menentukan sifat setiap makhluk hidup. Dengan menggunakan teknik komputasi untuk data bioinformatika berskala besar dan analisis data biologi yang rumit. Berbagai macam teknik komputasi, terutama teknik *Machine Learning* yang digunakan, contoh untuk memilih gen atau protein yang menghubungkan sifat yang terkait dan untuk mengklasifikasikan perbedaan tipe dari sampel ekspresi gen dari data *microarrays* (Yang P. , Yang, Zhou, & Zomaya, 2016).

Pada bidang kesehatan, bioinformatika digunakan dalam penerapan diagnosis, prediksi maupun pencegahan sebuah penyakit salah satunya yaitu virus Ebola atau EBOV pertama kali ditemukan pada tahun 1976 di Sudan. Virus ini termasuk dalam famili *Filoviridae* dan genus *Ebolavirus.* Terdapat 5 spesies *Ebolavirus,* 4 diantaranya menyebabkan penyakit pada manusia yaitu: *Zaire ebolavirus, Sudan ebolavirus, Taï Forest ebolavirus, Côte d'Ivoire ebolavirus* dan *Bundibugyo ebolavirus.* Virus jenis kelima adalah virus Ebola yang menyerang primata yaitu: *Reston ebolavirus* (Li & Chen , 2013). Penyebaran dan penularan virus Ebola pada manusia masih belum diketahui tapi dicurigai merupakan penyakit yang ditularkan oleh hewan yaitu kelelawar. Penularan virus Ebola dari manusia ke

manusia mudah terjadi. Menurut para ahli, virus Ebola dapat ditularkan melalui kontak dengan cairan tubuh penderita seperti darah, urin, cairan semen, air liur dan muntahan. Virus dapat masuk ke tubuh manusia melalui kulit atau mukosa yang tidak intak (Bausch , et al., 2007).

Dari Maret 2014 hingga Maret 2016, Afrika barat menderita wabah virus Ebola (EBOV) terbesar hingga saat ini, menghasilkan 28.652 kasus dan 11.325 kematian (Vernet & et al, 2017). Menyoroti kurangnya pengetahuan tentang patogenisitas virus dan faktor-faktor yang bertanggung jawab untuk hasil. Performa dan diagnosis yang cepat sangat penting, seperti mengatasi kesulitan menyediakan manajemen pasien berkualitas tinggi selama wabah yang luas. Di sini, mengusulkan untuk mempelajari peran mediator kekebalan selama penyakit virus Ebola dan untuk menentukan beberapa molekul yang penting dalam hasilnya (Reynard, Journeaux, Gloaguen, Schaeffer, & et al, 2019).

Machine learning (ML) yang tangguh dibutuhkan untuk memenuhi kebutuhan pengenalan pola data bioinformatika. Salah satu metode *boosting* yang sering digunakan adalah *gradient boosting* (GB) di mana metode ini menggunakan pendekatan *boosting* secara *gradient descent*. Mekanisme *gradient descent* dilakukan dalam rangka evaluasi untuk membuat model berikutnya. *Gradient boosting* pertama kali diperkenalkan oleh Friedman (2001) algoritma ini kemudian dikembangkan lebih lanjut, salah satu bentuk implementasinya adalah *eXtreme Gradient Boosting* atau *XGBoost* oleh Chen & Guestrin (2016). Algoritma ini merupakan pengembangan dari algortima *Gradient Boosting Machine* klasik dan hanya digunakan untuk data yang memiliki label dalam proses latihnya (Syahrani, 2019).

XGBoost mampu mengerjakan berbagai fungsi seperti regresi, klasifikasi, dan ranking. *XGBoost* merupakan *tree ensembles algorithm* yang terdiri atas kumpulan beberapa *classification and regression trees* (CART). Faktor yang paling penting di balik kesuksesan *XGBoost* adalah *scalability* dalam berbagai skenario. *Scalability* ini disebabkan karena optimasi dari algoritme sebelumnya. Inovasi yang diberikan meliputi *novel tree learning algorithm* untuk menangani *sparse data* (Chen & Guestrin, 2016). Kesuksesan tersebut terbukti saat *XGBoost* menjadi salah

satu metode yang sedang banyak diterapkan untuk berbagai kasus pada *machine learning*. *XGBoost* pertama kali dikenalkan dalam *Higgs Boson Competition*. Pada akhir kompetisi, *XGBoost* menjadi metode yang paling banyak digunakan oleh sebagian besar tim yang mengikuti kompetisi tersebut. Hal ini juga terjadi pada kompetisi *machine learning* yang diselenggarakan oleh situs Kaggle selama tahun 2015. Dari 29 *winning solution*, 17 di antaranya menggunakan algoritme *XGBoost*. Dari 17 solusi tersebut, sebanyak delapan solusi menggunakan *XGBoost* untuk model *training*, sedangkan sisanya mengombinasikan algoritme *XGBoost* dengan algoritme k-Nearest. Neighbor (k-*NN*) (Handayani, Jamal, & Septiandri, 2017).

Berdasarkan uraian latar belakang diatas penulis ingin melakukan penelitian menggunakan metode *XGBoost* sebagai representasi dari metode *boosting* untuk mengklasifikasikan terhadap data Parameter dan hasil kekebalan selama penyakit virus Ebola kemudian dilakukan optimasi dengan *Tuning hyperparameter* untuk menyeimbangkan *overfitting* dengan keakuratan dan kompleksitas komputasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan di atas, terdapat empat pokok masalah pada penelitian ini yaitu:

1. Bagaimana langkah-langkah untuk mengolah data ekspresi gen status kekebalan pasien virus ebola pada data *affymetrix human gene 2.0 st by array*?
2. Bagaimana gambaran data-data ekspresi gen status kekebalan pasien virus ebola pada data *affymetrix human gene 2.0 st by array*?
3. Bagaimana hasil klasifikasi pada data ekspresi gen status kekebalan pasien virus ebola menggunakan metode *eXtreme Gradient Boosting*?
4. Bagaimana hasil nilai optimasi *tuning hyperparameter* untuk meningkatkan hasil akurasi pada metode *eXtreme Gradient Boosting*?

1.3 Batasan Masalah

Dalam penelitian ini memiliki beberapa batasan untuk di teliti, berikut batasan masalah penelitian yang dilakukan yaitu :

1. Data yang digunakan merupakan data sekunder dari website NCBI (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE122692 diakses pada tanggal 10 Januari 2020. Mengenai data *immune parameters and outcome during ebola virus disease*.
2. Metode statistik yang digunakan yaitu salah satu teknik *machine learning* dengan metode *ensemble* yaitu *eXtreme Gradient Boosting* (*XGBoost*).
3. *Software* yang digunakan dalam penelitian ini adalah aplikasi *R Studio* dengan menggunakan beberapa *package* bioinformatika dan *XGBoost.*

1.4 Tujuan Penelitian

Berdasarkan rumusan dan batasan masalah di atas, maka tujuan dalam penelitian ini adalah untuk:

1. Mengetahui langkah-langkah untuk mengolah data ekspresi gen status kekbalan pasien virus ebola pada data *affymetrix human gene 2.0 st by array*.
2. Mengetahui gambaran data data ekspresi gen status kekbalan pasien virus ebola pada data *affymetrix human gene 2.0 st by array*.
3. Mengetahui hasil klasifikasi pada data ekspresi gen status kekbalan pasien virus ebola menggunakan metode *eXtreme Gradient Boosting* (*XGBoost*).
4. Mengetahui hasil nilai optimasi *tuning hyperparameter* untuk meningkatkan hasil akurasi pada metode *eXtreme Gradient Boosting* (*XGBoost*).

1.5 Manfaat Penelitian

Berdasarkan tujuan di atas, maka manfaat yang diharapkan dalam penelitian ini adalah sebagai berikut:

1. Bagi Mahasiswa Jurusan Statistika

   Penelitian ini bermanfaat sebagai bahan referensi bagi penelitian selanjutnya atau bagi pihak-pihak yang akan melakukan penelitian tentang data bioinformatika dan juga metode *eXtreme Gradient Boosting* serta sebagai literatur untuk menambah ilmu pengetahuan.
2. Bagi Masyarakat

Dapat mengetahui peran data *bioinformatika* dalam machine learning sangat perlu di kembangkan, serta untuk mengetahui beberapa informasi dalam bidang *bioinformatika*.

3. Bagi Peneliti Selanjutnya

   Untuk ikut serta mengembangkan metode *eXtreme Gradient Boosting* serta menjadi pembanding dengan metode klasifikasi yang lainnya.

1.6 Sistematika Penelitian

Bab pertama dari skripsi adalah pendahuluan yang memiliki gambaran umum dalam penyusunan sesuai dengan judul. Penulis menyusun pembabakan dari ringkasan setiap isi dari bab per bab yang dibagi dalam lima bab yaitu latar belakang masalah, perumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

Bab dua merupakan tinjauan pustaka yang memuat tinjauan penelitian terdahulu yang melandasi penulisan skripsi ini, yaitu penelitian tentang virus ebola, analisis *XGBoost* dan analisis *XGBoost* dalam bidang bioinformatika. Dalam bab ini juga akan diuraikan perbedaan dengan penelitian sebelumnya.

Bab tiga merupakan landasan teori yang memuat landasan teori skripsi ini, yaitu teori tentang Bioinformatika, Ebola *viruses*, *Gene expression*, *Microarray*, *Oligo*, *Preprocessing*, *Filtering*, *Feature selection*, Uji Anova, Analisis Deskriptif, *machine learning*, klasifikasi, *ensemble learning XGBoost*, *Tunning hyperparameter* dan *Confusion matrix*. Dalam bab ini juga akan diuraikan rerangka pemikiran teori yang digunakan dalam penulisan skripsi ini.

Bab empat merupakan metode penelitian yang memuat tempat dan waktu penelitian, metode pengambilan sampel, jenis dan sumber data. Dalam bab ini juga akan diuraikan teknik pengambilan data, variabel penelitian, definisi operasional, metode analisis data dan langkah penelitian.

Bab lima merupakan analisis data dan pembahasan yang memuat penyajian dan analisis data. Dalam bab ini juga akan dijelaskan mengenai hasil analisis pembahasannya.

Bab enam adalah penutup yang memberikan uraian mengenai kesimpulan dari hasil penelitian yang diambil dari bab analisis data dan pembahasan penelitian. Selain itu, juga dikemukakan keterbatasan penelitian serta saran-saran yang bermanfaat bagi pihak-pihak lain dikemudian hari.

BAB II
TINJAUAN PUSTAKA

Penelitian terdahulu sebagai kajian bagi penulis sangat penting untuk mengetahui hubungan antara penelitian yang dilakukan sebelumnya dengan penelitian yang penulis lakukan saat ini serta dapat menghindari adanya duplikasi. Hal ini bermanfaat untuk menunjukkan bahwa penelitian yang dilakukan mempunya arti penting sehingga dapat diketahui kontribusi penelitian terhadap ilmu pengetahuan. Beberapa penelitian tentang penerapan metode *Extreme Gradient Boosting* (*XGBoost*), yaitu :

2.1 Penelitian Terkait Penyakit Ebola

Penelitian yang dilakukan Reynard, Journeaux, Gloaguen, Schaeffer, & et al (2019) dengan judul : *Immune Parameters and Outcome During Ebola Virus Disease* pada penelitian ini mengevaluasi pola kekebalan tergantung pada hasil penyakit. Pasien menunjukkan respon imun yang efisien dan seimbang, sedangkan kematian ditandai dengan respon inflamasi yang intens, ekspresi berlebih dari beberapa sitokin, dan "badai kemokin." Konsentrasi plasma sebagian besar parameter yang diuji meningkat hingga mati. Analisis statistik juga memungkinkan kami untuk menentukan panel penanda yang sangat memprediksi hasil. Penelitian ini melibatkan 44 sampel dari 42 pasien yang meninggal karena penyakit ini dan 58 sampel dari 33 yang selamat, sesuai dengan tingkat kematian 56%, konsisten dengan yang diamati selama epidemi (62%). Tidak ada perbedaan signifikan antara usia rata-rata, rasio jenis kelamin, atau waktu antara timbulnya gejala dan masuk ketika membandingkan kematian dan orang yang selamat. Hanya sampel pertama dari setiap pasien (diperoleh saat masuk) yang selanjutnya dipertimbangkan untuk analisis statistik untuk menghindari bias dari jumlah sampel yang bervariasi di antara pasien. Sampel kontrol diambil dari 9 pasien EBOV-negatif dan 8 donor sehat.

Dengan menggunakan koefisien regresi logistik dan meninggalkan satu validasi silang pada protein dengan koefisien tidak nol, risiko kematian diperkirakan untuk setiap pasien dihitung. Kami merangkum nilai prediktif model dengan menghitung *area under the receiver operating curve* (AUROC). Analisis komponen utama dilakukan untuk meringkas informasi yang terkandung dalam data dari model ini dengan mengurangi dimensi data ini dan menambahkan kontrol pasien. Analisis statistik dilakukan dengan perangkat lunak R 3.4.3. Analisis komponen utama dilakukan dengan menggunakan paket factoextra dan perangkat lunak R serta ROC dan AUROC menggunakan paket-R plotROC.

2.2 Penelitian Terkait Metode *eXtreme Gradient Boosting* (*XGBoost*)

Penelitian yang dilakukan Chen dan Gustrin (2016) dengan judul : *XGBoost: A Scalable Tree Boosting System*. Pada penelitian ini menggunakan metode *Boosting* (*XGBoost). Tree boosting* adalah mesin yang sangat efektif dan banyak digunakan dalam metode pembelajaran. Penulis menggambarkan *a scalable end to end tree boosting system* disebut *XGBoost*, yang digunakan secara luas oleh para ilmuwan data untuk mencapai hasil-hasil canggih pada banyak tantangan *machine learning*. Penulis mengusulkan sebuah novel Algoritma sparsity aware untuk *sparse data* dan *weighted quantile sketch* untuk perkiraan *tree learning*. Lebih penting, penulis memberikan wawasan tentang pola akses *cache, data compression* dan *sharding* untuk membangun *scalable tree boosting system*. Dengan menggabungkan wawasan ini, *XGBoost* meningkatkan skala melampaui miliaran contoh menggunakan sumber daya jauh lebih sedikit daripada sistem yang ada. Kesimpulan dalam tulisan ini, menggambarkan pelajaran yang penulis pelajari saat itu membangun *XGBoost, scalable tree boosting system* yang dapat diskalakan banyak digunakan oleh para ilmuwan data dan memberikan keadaan seni hasil pada banyak masalah. Penulis mengusulkan novel *sparsity aware algorithm for handling sparse data and a theoretically justifed weighted quantile sketch for approximate*

learning. Pengalaman penulis menunjukkan bahwa pola akses *cache*, *data compression* dan *sharding* adalah elemen penting untuk membangun *a scalable end to end tree boosting system*. Pelajaran ini dapat diterapkan pada sistem *machine learning* lainnya juga. Dengan menggabungkan wawasan ini, *XGBoost* mampu menyelesaikan masalah skala menggunakan jumlah sumber daya yang minimal.

Salah satu algoritma *machine learning* yang baru-baru ini mendapatkan tolok ukur dalam berbagai keadaan seni masalah dalam *machine learning* adalah *eXtreme Gradient Boosting (XGBoost)*. Algoritma ini telah dipecahkan banyak di berbagai tantangan penambangan data dan *machine learning* yang telah diterapkan dalam berbagai masalah dan memberikan hasil yang bagus. Penelitian yang dilakukan oleh Salamah dan Ramayanti (2018) dengan judul : *Supervised Classification of Indonesian Text Document Using Extreme Gradient Boosting (XGBoost)* yang berisi tentang mengklasifikasikan data teks menggunakan *XGBoost* untuk memprediksi teks apakah diklasifikasikan sebagai keluhan atau non-keluhan berdasarkan data yang ada di media sosial. Dalam percobaan ini, penulis memasang *package XGBoost* pada sistem untuk digunakan dalam *Python*. *XGBClassifier* diimpor ke kode penulis untuk mendukung menggunakan Pencarian *Grid sklearn* untuk parameter *tuning* dengan pemrosesan paralel. Algoritma *XGBoost* menggunakan beberapa parameter sehingga dapat ditingkatkan dengan menyetel parameter, *mis. eta* atau tingkat belajar, *gamma, max_depth, min_child_weight, subsample, colsample_bytree,* dan *alpha*. Sebagai hasilnya, nilai terbaik untuk setiap parameter adalah '*reg_alpha*': 0,01, '*colsample_bytree*': 0,9, '*learning_rate*': 0,5, '*min_child_weight*': 1, '*subsample*': 0.8, '*max_depth*': 3, '*gamma*': 0.0. Bahkan waktu komputasi yang dibutuhkan untuk tune parameter tersebut adalah 13870.012468 dan akurasi terbaik yang dicapai adalah 0,927943760984.

Penelitian yang dilakukan Prasetyo, Christianto, & Hartomo (2019) dengan judul : Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan *Machine Learning* di Wilayah Kabupaten Boyolali dan Purworejo. Pada penelitian ini digunakan metode *XGBoost* dan *Random Forest* untuk Penggunaan data citra satelit Landsat 8 OLI sebagai media informasi vegetasi dan pendekatan *Machine Learning* untuk menganalisa data ekstraksi pada citra satelit berupa indeks vegetasi. Indeks vegetasi yang di gunakan yaitu NDVI, VCI, VHI, dan TCI. Oleh karena itu penelitian ini ditunjukan khusus untuk membantu Bencana kekeringan merupakan salah satu bencana yang tidak dapat di hindari lagi keberadaannya. Berdasarkan data dari tahun 1815 sampai tahun 2015 telah terjadi 382 kejadian. Berdasarkan kajian BNPB Kabupaten Boyolali dan Kabupaten Purworejo memiliki resiko tinggi terpapar bencana kekeringan. Untuk itu perlu adanya informasi wilayah resiko bencana kekeringan. Pada proses pengujian hasil yang diperoleh dengan menggunakan perhitungan metode *XGBoost* ada total 9 kecamatan yang diprediksi terkena bencana kekeringan sangat parah, dan 9 kecamatan dengan metode *Random Forest* terindikasi kekeringan sangat parah. metode *XGBoost* memiliki nilai akurasi 0.8286 dan nilai kappa 0.6477 dan metode *Random Forest* memiliki nilai akurasi 0.6857 dan Nilai Kappa 0.3699. dimana semakin tinggi nilai akurasi dan kappa semakin tepat hasil prediksi yang dilakukan.

2.3 Penelitian dalam Bidang Bioinformatika Menggunakan Metode *eXtreme Gradient Boosting* (*XGBoost*)

Penelitian yang dilakukan Torlay, Perrone-Bertolotti, Thomas, & Baciu (2017) dengan judul : *Machine learning XGBoost analysis of language networks to classify patients with epilepsy*. Tujuan penulis adalah untuk menerapkan pendekatan statistik untuk memungkinkan identifikasi pola bahasa atipikal dan untuk membedakan pasien dengan epilepsi dari subyek sehat, berdasarkan aktivitas otak mereka, sebagaimana dinilai oleh MRI fungsional (fMRI). Analisis fMRI memungkinkan identifikasi pola atipikal

jaringan bahasa pada pasien tidak cukup kuat dan memerlukan pendekatan statistik tambahan. Dalam penelitian ini, penulis menjelaskan penggunaan klasifikasi *machine learning* nonlinear statistik, algoritma *Extreme Gradient Boosting (XGBoost)*, untuk mengidentifikasi pola atipikal dan mengklasifikasikan 55 peserta sebagai subyek sehat atau pasien dengan epilepsi. Analisis *XGBoost* didasarkan pada fitur neurofisiologis di lima wilayah bahasa (tiga frontal dan dua temporal) di kedua belahan otak dan diaktifkan dengan fMRI untuk tugas bahasa fonologis (PHONO) dan semantik (SEM). Fitur-fitur ini digabungkan menjadi 135 himpunan bagian yang masuk akal secara kognitif dan selanjutnya diajukan ke seleksi dan klasifikasi biner. Kinerja klasifikasi dinilai dengan area di bawah kurva karakteristik pengoperasian penerima (AUC). Hasil kami menunjukkan bahwa subset SEM_LH BA_47-21 (Aktivasi fronto-temporal kiri yang disebabkan oleh tugas SEM) memberikan diskriminasi terbaik antara kedua kelompok (AUC sebesar 91 ± 5%). *Machine Learning XGBoost* adalah metode statistik klasifikasi yang kuat yang mendeteksi pola nonlinier dalam kumpulan data dengan nilai yang hilang. Ini menunjukkan potensi signifikan untuk mengklasifikasikan pasien dengan epilepsi berdasarkan wilayah otak, belahan dan pemrosesan representasi bahasa mereka. Satu subset, atau kombinasi fitur tertentu, SEM_LH BA_47-21, adalah yang paling kuat, untuk mengidentifikasi pasien. Pentingnya subset khusus ini masuk akal mengingat pengamatan kognitif dan klinis yang dilakukan dengan pasien ini.

Penelitian yang dilakukan Syahrani (2019) dengan judul : Analisis Perbandingan Teknik Ensamble Secara *Boosting (XGBoost)* dan *Bagging (Random Forest)* Pada Klasifikasi Sambatan Skuens DNA. Data biogenetika berupa urutan nukleotida atau DNA merupakan data yang sering dianalisa berupa bentuk polanya salah satunya sebagai pengidentifikasi jenis protein. Penelitian ini menggunakan *dataset Splice-junction* yang merupakan urutan nukleotida (DNA) di mana di dalamnya

terdapat gen yang dianggap tidak berguna dan akan ditiadakan selama proses pembentukan protein pada organisme tingkat tinggi. *Exon* merupakan bagian dari urutan DNA yang tetap dipertahankan setelah proses *splicing* dan *intron* merupakan bagian DNA yang disambung keluar. *Dataset* berisi data yang mencirikan kategori berdasarkan urutan apakah berupa batas *ekson* terhadap *intron* (EI) ataukah berupa batas *intron* terhadap *ekson* (IE). Terdapat kategori lain yang bukan merupakan keduanya yaitu *neither(N).* Data ekstraksi ciri sekuens DNA berupa frekuensi k-mers juga dianalisa sebagai perbandingan. Penelitian ini menggunakan metode *ensemble* secara *boosting* dan *bagging* mampu menangani klasifikasi dengan akurasi yang baik. Optimasi dengan *tuning hyperparameter* yang tepat terbukti mampu meningkatkan nilai akurasi. Pada studi kasus ini *XGBoost* mencapai tingkat akurasi lebih baik dibandingkan *random forest* namun dalam hal waktu latih dan uji *random forest* lebih unggul. Setelah dikeluarkan *output* berupa nilai *gain* dari model xgboost dan RF hasil dari proses uji sekuens DNA dan frekuensi k-mers, terlihat nilai *gain* tertinggi pada sekuens DNA berada di urutan 30 s.d. 34. Sebagai pembandingan digunakan nilai *gain* dari klasifikasi *dataset* frekuensi k-mers dengan nilai k=5. Jika merujuk pada Lampiran 5, nilai *gain* tertinggi frekuensi 5-mers ada pada kombinasi urutan GGTGA dan GGTAA. Penghitungan manual dari *dataset* sekuens DNA, diperoleh kedua urutan ini mayoritas berada di urutan ke 30 sampai 34 pada tiap data observasi. Sehingga dapat disimpulkan kedua metode ini mampu mengenali penciri dominan dari data observasi untuk diklasifikasikan dalam kategori jenis sambatannya.

Tabel 2.1 Perbandingan penelitian

Judul	Peneliti, Tahun	Metode	Kasus	Hasil / Kesimpulan
Immune Parameters and Outcome During Ebola Virus Disease	Reynard, Journeaux, Gloaguen, Schaeffer, & et al, 2019	*ROC* dan *AUROC* menggunakan *package*-R *plotROC*	Mengevalu-asi pola kekebalan tergantung pada hasil penyakit Virus Ebola.	Tidak ada perbedaan signifikan antara usia rata-rata, rasio jenis kelamin, atau waktu antara timbulnya gejala dan masuk ketika membanding-kan kematian dan orang yang selamat.
XGBoost: A Scalable Tree Boosting System	Chen & Guestrin, 2016	*A scalable end to end tree boosting system* disebut *XGBoost*	Pola akses *cache*, *data compression* dan *sharding* untuk membangun *scalable tree boosting system.*	*XGBoost* mampu menyelesaikan masalah skala menggunakan jumlah sumber daya yang minimal, pola akses *cache*, *data compression* dan *sharding* adalah elemen penting untuk membangun *a scalable end to end tree boosting system.*
Supervised Classification of Indonesian Text Document Using *Extreme Gradient Boosting (XGBoost)*	Salamah & Ramayanti, 2018	klasifikasi *machine learning* menggunakan *Extreme Gradient Boosting (XGBoost)*	Memprediksi data teks apakah diklasifikasi-kan sebagai keluhan atau non-keluhan berdasarkan data yang ada di media sosial.	Sebagai hasilnya memiliki nilai terbaik untuk setiap parameter dan waktu komputasi yang dibutuhkan untuk tune parameter tersebut adalah

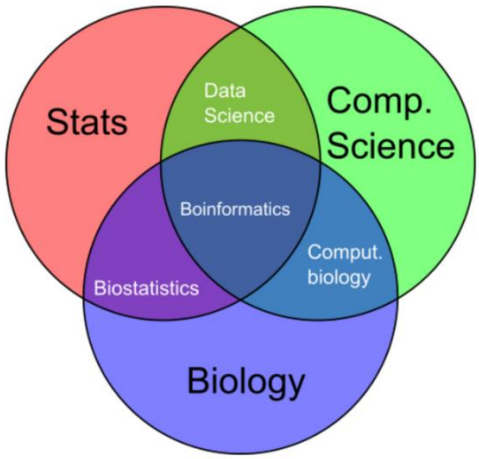
				13870.012468 dan akurasi terbaik yang dicapai adalah 0,92794376098.
Judul	**Peneliti, Tahun**	**Metode**	**Kasus**	**Hasil / Kesimpulan**
Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan *Machine Learning* di Wilayah Kabupaten Boyolali dan Purworejo	Prasetyo, Christianto, & Hartomo, 2019	Klasifikasi *XGBoost* dan *Random Forest*	Penggunaan data citra satelit Landsat 8 OLI sebagai media informasi vegetasi dan pendekatan *Machine Learning* untuk menganalisa data ekstraksi pada citra satelit.	Metode *XGBoost* memiliki nilai akurasi lebih tinggi dibandingkan metode *Random Forest.* dimana semakin tinggi nilai akurasi dan kappa semakin tepat hasil prediksi yang dilakukan.
Machine learning XGBoost analysis of language networks to classify patients with epilepsy	Torlay, Perrone-Bertolotti, Thomas, & Baciu, 2017	Klasifikasi *machine learning* menggunakan *Extreme Gradient Boosting (XGBoost).*	mengidentifikasi dan mengklasifikasikan pasien dengan epilepsi.	Menunjukkan potensi signifikan untuk mengklasifikasikan pasien dengan epilepsi berdasarkan wilayah otak, belahan dan pemrosesan representasi bahasa mereka. Satu subset, atau kombinasi fitur tertentu, SEM_LH BA_47-21, adalah yang paling kuat, untuk mengidentifikasi pasien.

Judul	Peneliti, Tahun	Metode	Kasus	Hasil / Kesimpulan
Analisis Perbandingan Teknik Ensamble Secara *Boosting (XGBoost)* dan *Bagging (Random Forest)* Pada Klasifikasi Sambatan Skuens DNA	Syahrani, 2019	Perbandingan *Boosting (XGBoost)* dan *Bagging (Random Forest*	Data biogenetika berupa urutan nukleotida atau DNA sebagai pengidentifikasi jenis protein	*XGBoost* mencapai tingkat akurasi lebih baik dibandingkan *random forest* namun dalam hal waktu latih dan uji *random forest* lebih unggul
Pembeda dari penelitian sebelumnya				
Implementasi metode *eXtreme Gradient Boosting (XGBoost)* untuk Klasifikasi pada Data Bioinformatika	Gregy Addis Shafila. 2020	Klasifikasi XGBoost dengan *Tunning Hyperparameter* secara *Grid Search*	Parameter dan Hasil kekebalan selama penyakit virus Ebola	Hasil akurasi pada boosting *XGBoost* sebelum dilakukan *tunning hyperparameter* sebesar 69,23% dan 76,92%. setelah dilakukan *tunning hyperparameter* secara *grid search*.

BAB III
LANDASAN TEORI

3.1 Bioinformatika

Bioinformatika, sesuai dengan asal katanya yaitu “bio” dan “informatika” adalah gabungan antara ilmu biologi dan ilmu teknik informasi (TI). Pada umumnya, Bioinformatika didefenisikan sebagai aplikasi dari alat komputasi dan analisa untuk menangkap dan menginterpretasikan data-data biologi. Ilmu ini merupakan ilmu baru yang yang merangkup berbagai disiplin ilmu termasuk ilmu komputer, matematika dan fisika, biologi, dan ilmu kedokteran (Gambar 2.1), dimana kesemuanya saling menunjang dan saling bermanfaat satu sama lainnya (Bayat, 2002). Ilmu bioinformatika lahir atas insiatif para ahli ilmu komputer berdasarkan *artificial intelligence*. Mereka berpikir bahwa semua gejala yang ada di alam ini bisa dibuat secara *artificial* melalui simulasi dari gejala-gejala tersebut. Untuk mewujudkan hal ini diperlukan data-data yang menjadi kunci penentu tindak-tanduk gejala alam tersebut, yaitu gen yang meliputi DNA atau RNA. Bioinformatika ini penting untuk manajemen data-data dari dunia biologi dan kedokteran modern (Utama, 2003).



Gambar 3.1 Interaksi disiplin ilmu yang berhubungan dengan Bioinformatika
Sumber: (Ramadhan, 2020)

Menurut Aprijani & Elfaizi (2004) Di Indonesia, Bioinformatika masih belum dikenal oleh masyarakat luas. Hal ini dapat dimaklumi karena penggunaan komputer sebagai alat bantu belum merupakan budaya. Bahkan di kalangan peneliti sendiri, barangkali hanya para peneliti biologi molekul yang sedikit banyak mengikuti perkembangannya karena keharusan menggunakan perangkat-perangkat Bioinformatika untuk analisa data. Sementara di kalangan TI masih kurang mendapat perhatian. Ketersediaan database dasar (DNA, protein) yang bersifat terbuka/gratis merupakan peluang besar untuk menggali informasi berharga daripadanya. Database genom manusia sudah disepakati akan bersifat terbuka untuk seluruh kalangan, sehingga dapat digali/diketahui kandidat-kandidat gen yang memiliki potensi kedokteran/farmasi. Dari sinilah Indonesia dapat ikut berperan mengembangkan Bioinformatika. Kerjasama antara peneliti bioteknologi yang memahami makna biologis data tersebut dengan praktisi TI seperti programmer, dan sebagainya akan sangat berperan dalam kemajuan Bioinformatika Indonesia nantinya.

3.2 Ebola Viruses

Pada bulan Maret 2014 WHO mengumumkan adanya wabah dari penyakit menular yang ditandai adanya demam, diare berat dan muntah dengan tingkat keparahan yang tinggi di Guinea. Wabah ini disebabkan oleh virus Ebola (EBOV) dan Marburg (MARV). Virus Ebola menyebabkan tingkat keparahan mencapai 30-90%, bergantung pada spesies virus itu sendiri (Baize, et al., 2014). Wabah penyakit ini telah diidentifikasi setiap tahun selama 3 tahun terakhir di Afrika Tengah. Laporan terakhir didapatkan di Republik Congo dan menurut WHO terdapat sekitar lebih dari 125 kasus yang fatal (Sullivan, Yang, & Nabel, 2003).

EBOV merupakan patogen agresif yang menyebabkan gejala demam dengan perdarahan yang letal pada manusia dan hewan. Adanya penyakit ini pertama kali ditemukan di dekat sungai Ebola dengan wabah di Zaire pada tahun 1976. *Outbreak* sudah terjadi di Afrika selama 27 tahun dengan mortalitas berkisar antara 50-90% (Sullivan, Yang, & Nabel, 2003). Penyebaran virus Ebola tidak hanya terjadi di Afrika. Jenis virus yang baru Reston ebolavirus (REBOV) ditemukan pada kera-

kera yang diimpor dari Manila (Filipina) ke Amerika pada tahun 1989. (Mayumbe-Tamfum, et al., 2012).

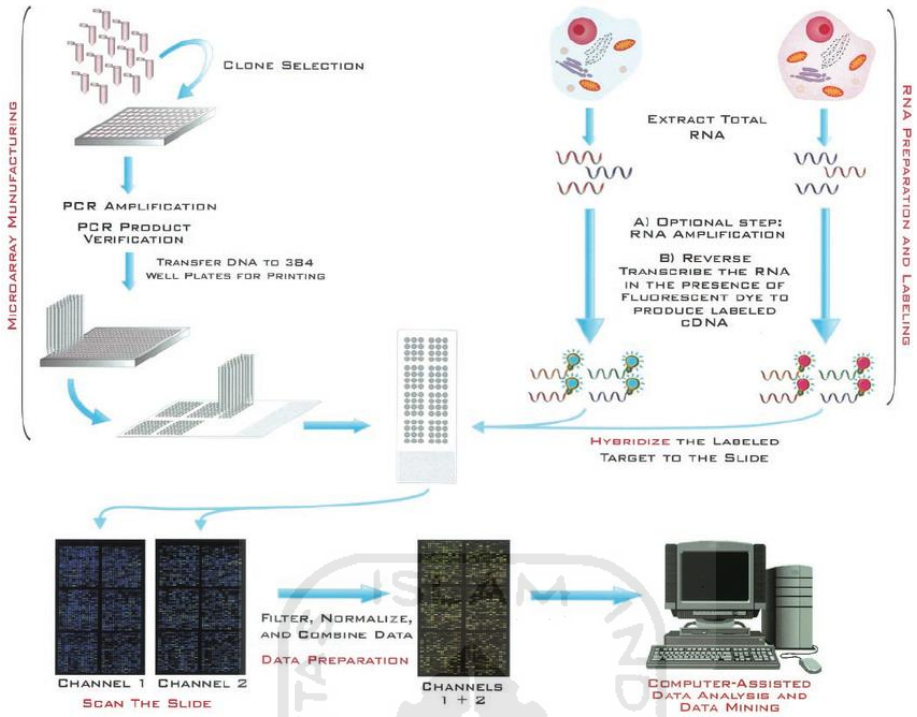
Di Afrika terdapat 3 jenis virus Ebola yang menyebabkan wabah terbesar, yaitu: EBOV, Sudan ebolavirus dan Bundibugyo ebolavirus. Nilai fatalitas EBOV dapat mencapai 30-90% tergantung pada jenis virus. (Baize, et al., 2014) Virus Ebola berasal dari golongan *Filoviridae*. Jenis ini merupakan *virionpleomorfik* yang dapat berbentuk huruf U, angka 6, atau lingkaran, tetapi yang paling sering terlihat di mikroskop elektron ialah struktur tubular panjang. Virus Ebola mengandung 1 molekul *linear single stranded* dengan *negative-sense* RNA yang hampir mirip dengan *Paramyxoviridae* (Sullivan, Yang, & Nabel, 2003).

3.3 *Gene Expression*

Menurut Kumar (2010) Ekspresi gen atau *Gene Expression* adalah serangkaian peristiwa atau tahapan pada tingkat yang berbeda hingga pelepasan protein viamRNA. Tingkat utama dari itu adalah transkripsi yang merupakan sintesis mRNA dan translasi mengarah ke sintesis protein, molekul fungsional gen. Seluruh pelengkap protein dalam suatu organisme disebut *Proteome* dan analisis sampel biologis untuk kadar protein, modifikasi dan aktivitas dikenal sebagai *proteomik*. Membandingkan tingkat mRNA dari seluruh set gen dalam sampel biologis dengan cDNA berlabel pewarna *fluorescent* dilakukan oleh ekspresi Gen *microarrays* juga dikenal sebagai *microDays* DNA. Melalui *microarray*, dimungkinkan untuk mengetahui gen mana yang diekspresikan mengingat level apa dalam kondisi tertentu.

3.4 *Microarray*

Microarray adalah sebuah perangkat yang dirancang untuk mengukur tingkat ekspresi ribuan gen di jenis jaringan atau sel tertentu. (Bolstad, 2004). Sedangkan menurut (Hovatta, et al., 2005) *Microarray* adalah suatu *chip* atau *slide* mikroskop yang berisi rangkaian sampel jaringan, protein, RNA dan DNA. Pada gambar 3.2 data *microarray* dengan platform *Affymetrix* berbentuk *affybatch* objek dengan melalui beberapa tahapan untuk mendapatkan data *microarray*.



Gambar 3.2 Langkah – Langkah dalam Percobaan Microarray
Sumber: (Leon & Squire, 2002)

Keberhasilan penerapan *microarray* untuk analisis ekspresi dapat ditelusuri kembali ke dua dekade lalu. Sejak itu, teknik *microarray* telah banyak digunakan untuk profil ekspresi di hampir setiap bidang penelitian biologi. Selain analisis transkrip, teknik alternatif berbasis *microarray* juga telah dirancang untuk tujuan lain seperti genotipe, pemetaan DNA, pengikatan protein, dan studi epigenetik. Karena karakteristik *throughput* yang tinggi dari teknik *microarray*, telah mengubah studi biologi dari gen spesifik ke tingkat transkriptome dan sangat mendorong banyak bidang studi biologi (Sun , Shao, & Wang, 2018).

Menurut (Vicente, 2009) *Microarray* mempunyai dua perbedaan yang mengarah pada jumlah gen yang dimasukkan pada setiap *chip*:

1. *One-channel Microarray*

   adalah *microarray* yang berisi urutan gen dari suatu sampel untuk diproses hibridisasi, contoh dari jenis ini adalah *Affymetrics GeneChip*. Jenis ini membutuhkan dua chip untuk perbandingan sampel dan dilakukan dengan cara komputasi.

2. *Two-channel Microarray*

   adalah *microarray* yang berisi urutan gen dari dua sampel yang berbeda untuk diproses hibdridisasi kompetitif. Jenis ini mempunyai warna neon yang berbeda yaitu hijau atau merah. Untuk menampilkan sebuah sampel control vs sampel percobaan penyakit hanya membutuhkan satu chip.

3.5 *Oligo*

Beberapa platform saat ini tersedia, termasuk array *Affymetrix GeneChip®* berbasis *Oligonucleotide* (*Oligo*) tinggi yang umum di gunakan. *Oligo* adalah paket *Bioconductor* untuk *preprocessing oligonucleotide microarrays*. Saat ini mendukung chip yang diproduksi oleh *Affymetrix* dan *NimbleGen*. Paket oligo memungkinkan pengguna untuk *preprocessing* data *microarray*. *Oligo* dapat mengidentifikasi file CEL atau XYS, ini dilakukan dengan fungsi `list.celfiles` atau `list.xysfiles`. Fungsi-fungsi ini memiliki argument yang sama dengan `list.files` yaitu mengimpor file CEL dan XYS. Dari fungsi ini dapat divisualisasikan melalui strategi yang berbeda seperti, *pseudo-image, smoothed histogram, boxplot* dan *MAplot* (Carvalho & Irizarry , 2010).

Menurut (Carvalho & Irizarry , 2010) Paket *Oligo* dan *Affy* memiliki keterkaitan yang erat. Yang pertama, menggunaka pengetahuan yang diperoleh untuk memberikan solusi bagi keterbatasan yang ditemukan sejauh ini. Peningkatan utama yang ditawarkan *Oligo* adalah:

1. Dukungan untuk banyak vendor dan platform.
2. Skema penyimpanan dan akses yang efisien untuk anotasi *array throughput* tinggi saat ini, yang metadata-nya telah menjadi sangan besar.
3. Dukungan asli ke file pabrikan.

3.6 *Preprocessing*

Menurut (Bolstad, 2004) *Preprocessing* merupakan tahap untuk penyesuaian serta konversi data *affybatch* ke dalam bentuk *expression set*. Tidak hanya melakukan konversi data *affybatch*, dalam tahap *preprocessing* dilakukan proses *background correction, normalization* dan *summarization*. Istilah *background*

correction mengacu pada penyesuaian berbagai macam metode, serta yang harus dilakukan meliputi:

1. Memperbaiki background noise dan efek pengolahan.
2. Menyesuaikan hibridisasi pengikat DNA non-spesifik pada array.
3. Menyesuaikan estimasi ekspresi sehingga bersekala tepat atau berhubungan linear.

Penting untuk dicatat bahwa definisi tersbut merupakan hal yang umum digunakan. *Background correction* secara umum mengacu pada penegrtian pertama. Tahap setelah *background correction* yaitu *normalization*. *Normalization* adalah proses penghapusan akhiran non-biologis yang tidak diinginkan diantara chip dalam eksperimen *microarray*. Tahap setelah hal tersebut ialah tahap dilakukan *summarization*, yaitu untuk mengukur *gene expression*. Paket Bioconductor "oligo" menawarkan sejumlah alat untuk *preprocessing* file *Affymetrix CEL*, termasuk *data import, background correction, normalization, data summarization and visualization*. Selain itu, mungkin perlu menginstal dan memuat paket anotasi probeset (mis., Pd.hugene.2.0.st untuk platform HuGene2.0-ST). *Robust Multi-Array* (RMA) adalah algoritma *normalisasi* yang paling banyak digunakan untuk menormalkan ekspresi gen dan membuat objek *ExpressionSet* (Sun , Shao, & Wang, 2018).

(Gautier, Cope, Bolstad, & Irizarry, 2004) menjelaskan empat langkah *Preprocessing* sebagai berikut:

1. *Background correction.* Ini menghilangkan intensitas latar belakang yang tidak spesifik dari *scanner image*. Tiga algoritma yang dapat digunakan adalah *Robust Multichip Average* (RMA), *Convolution* dan algoritma *MAS* dan *Gcrma*.
2. *Normalization*. Hal ini dimaksudkan untuk mengurangi sebagian besar perbedaan non-biologis antara chip yang memberikan intensitas sinyal yang di normalisasi untuk setiap chip. Beberapa metode nya adalah *Scaling, Quantile, Loess, Invariant Set* dan *Variance Stabilization Method* (VSN).

3. *Perfect Match* (PM) *Correction*. Intensitas sinyal PM harus disesuaikan untuk memperhitungkan sinyal yang tidak spesifik. Ada dua kemungkinan algoritma: MAS dan PMonly (Pilihan khusus untuk platform Affymetrix).
4. *Summarization*. Ini adalah tahap terakhir dalam *Preprocessing* data *Affymetrix GeneChip*® dan menghitung nilai ekspresi dari semua replikasi dalam chip dengan menggunakan intensitas probe 11-20 untuk menghasilkan nilai ekspresi tunggal untuk sebuah gen. Ada beberapa metode yang sering digunakan dalam literatur: *Median Polish*, *Tukey Biweight*, *Li-Wong MBEI expression index* dan *Avgdiff*.

Algoritma yang dijelaskan diatas dapat ditemukan dalam paket *Affy Bioconductor*, dalam *library* R yaitu *affyPLM* dan *vsn*. Dalam table 3.1 adalah ringkasan metode *Preprocessing* data *microarray* dan fungsi R yang digunakan.

Tabel 3.1 Metode *Pre-Processing*

Method	***Background Correction***	***Normalization***	***PM correction***	***Summarization***	***Funciton***
RMA	rma	quantiles	pmonly	medianpolish	expresso(), rma(), threestep()
GCRMA	gcrma	quantiles	pmonly	medianpolish	gcrma(), threestep()
MAS5	mas	mas	mas	mas	expresso(), mas5()
VSN		vsn	pmonly	medianpolish	expresso(), vsnmas()
dChip		invariantset	pmonly	liwong	expresso()
Loess 1	mas	loess	mas	mas	expresso()
Loess 2		loess	pmonly	avgdiff	expresso()

Sumber: (Pomares, Calvo, & Gonzalo, 2009)

Hasil dari *pre-processing* adalah data yang siap untuk dianalisis berbentuk kelas *Affybatch* untuk data yang berasal dari platform *Affymetrix*. Data hasil *pre-processing* yang berbentuk yang berbentuk data matriks atau berbentuk data frame, barisnya merupakan informasi tentang gen dan kolom adalah informasi mengenai sampel. Cel matriks atau data frame berupa sebuah angka, yaitu nilai intensitas yang

menggambarkan besarnya ekspresi dari suatu gen. Data intensitas ini berasal dari pengukuran image, dalam pixel dan chip yang telah dihibridasi dengan objek dari sampel.

3.7 *Filtering*

Filtering adalah salah satu operasi dasar untuk menghapus sebagaian data yang tidak diikutsertakan dalam analisis atau data yang tidak valid. *Filtering* dapat meningkatkan kualitas model, juga dapat membuat proses pemodelan lebih efesien. Menurut (Bourgon, Gentleman, & Huber, 2010) fungsi `nsFilter` (*non specific filter*) menyediakan *one-stop shop* (serba ada) untuk berbagai pilihan *filtering* (penghapusan) fitur dari *expression set*. *Filtering* berguna untuk menghapus sebagian *probe* yang tidak dibutuhkan, hal ini dapat berguna untuk analisis selanjutnya. Dalam *function* `nsFilter` terdapat beberapa perintah yang dilakukan yaitu `var.cutoff` untuk memfilter gen dengan nilai tertentu yang berguna untuk menghapus data dengan nilai IQR dibawah quartil. Variasi *default* `var.cutoff` adalah 0.5 dan dimotivasi oleh aturan praktis bahwa dibanyak jaringan hanya 40% gen yang diekspresikan sesuai dengan data. `Require.entrez TRUE` berfungsi untuk memfilter tanpa anotasi *Entrez Gene ID*, dimana sistem pengenal selain ID pusat, maka ID tersebut yang akan diperlukan. `Remove.DupEntrez TRUE` akan menghapus ID *gen Entrez* yang sama atau terduplikat. `Feature.exclude` berfungsi untuk menyaring *probe control* yaitu AFFX supaya tidak dimasukkan kedalam analisis.

3.8 *Feature selection*

Pemilihan fitur atau biasa disebut *Feature selection* adalah proses untuk membuat pengklasifikasian lebih efektif dengan cara mengurangi data-data yang tidak relevan, sehingga dapat meringkas waktu pengklasifikasian dan meningkatkan akurasi (Karabulut, Ozel, & Ibrikci, 2011). Menggunakan bantuan dari *packages* `multtest`, dalam melakukan *feature selection* dapat menggunakan perintah `mt.teststat` yang mempunyai fungsi untuk menghitung statistik uji misalnya seperti uji t-test, wicoxon, uji F, untuk setiap baris kerangka data. Pada

penelitian ini digunakan uji statistic F karena memiliki 4 sampel. Uji F dilakukan dengan langkah membandingkan dari F_{hitung} dengan F_{tabel}. Nilai F_{hitung} dapat dilihat dari hasil pengolahan data bagian Anova (Pollard, et al., 2019). Hasil dari *Feature selection* merupakan data yang sudah siap untuk dianalisis lebih lanjut untuk mendapati model dan akurasi yang lebih baik serta mengefesiensikan waktu.

3.9 Uji Signifikansi Simultan (Uji Statistik F)

Uji keberartian model regresi atau disebut dengan uji F, yaitu pengujian terhadap variable independent secara bersama (simultan) yang ditunjukan untuk mengetahui apakah semua variable independen secara bersama-sama dapat berpengaruh terhadap variabel dependen (Santoso, 2006).

Uji F dilakukan untuk melihat pengaruh variabel X1, X2, X3 dan variabel X4 secara keseluruhan terhadap variabel Y. untuk menguji hipotesis H_0 : b = 0, maka langkah – langkah yang akan digunakan untuk menguji hipotesis tersebut dengan uji F adalah sebagai berikut (Ghozali, 2011):

1. Menentukan Hipotesis:

   H_0: $E_1 = 0$ (tidak terdapat pengaruh yang signifikan antara variabel independent dan variabel dependen)

   H_1: $E_1 \neq 0$ (terdapat pengaruh yang signifikan antara variabel independent dan variabel dependen)
2. Menentukan *Level of Significance*

   Sebesar 5% atau $\alpha = 0.05$
3. Daerah Kritis
   1. Bila F hitung < F tabel, variabel bebas secara bersama-sama tidak berpengaruh terhadap variabel dependen, H0 diterima.
   2. Bila F hitung > F tabel, variabel bebas secara bersama-sama berpengaruh terhadap variabel dependen, H0 ditolak.
4. Statistik Uji

$$\mathbf{F} = \frac{R^2/k}{(1-R^2)-(n-k-1)} \tag{3.1}$$

Keterangan:

R^2 = Koefisien Korelasi Ganda

k = Jumlah variabel independent

n = Jumlah anggota sampel

5. Kesimpulan

   Kesimpulan untuk tolak atau gagal tolak H_0 berdasarkan perumusan hipotesis.

3.10 Analisis Deskriptif

Statistika deskriptif merupakan cara mendeskripsikan atau menggambarkan data yang telah terkumpul sebagaimana adanya tanpa memberikan kesimpulan yang berlaku untuk umum. Penyajian analisis deskriptif lebih ditekankan dalam bentuk tabel, grafik dan ukuran statistik seperti persentase, rata-rata, variansi dan angka indeks. Selain itu analisis deskriptif tidak menggunakan uji signifikansi dan taraf kesalahan karena tidak ada kesalahan generalisasi (Purwoto, 2007).

3.11 *Machine Learning*

Machine Learning atau pembelajaran mesin merupakan pendekatan dalam *Artifical Intelegent* yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi. Sesuai namanya, mencoba menirukan bagaimana proses manusia atau makhluk cerdas belajar dan mengeneralisasi (Tanaka & Okutomi, 2014). Setidaknya ada dua aplikasi utama dalam *Machine Learning* yaitu, klasifikasi dan prediksi. Ciri khas dari *Machine Learning* adalah adanya proses pelatihan dan pembelajaran, oleh karena itu, membutuhkan data untuk dipelajari yang disebut sebagai data *training* dan *testing*. Klasifikasi adalah metode dalam *Machine Learning* yang digunakan oleh mesin untuk memilah atau mengklasifikasikan obyek berdasarkan ciri tertentu sebagaimana manusia mencoba membedakan benda satu dengan yang lain. (Abu, 2017).

Membagi dataset menjadi dua bagian (*training* dataset dan *testing* dataset) adalah langkah persiapan untuk membuat model klasifikasi yang valid dengan akurasi yang dapat diandalkan dalam fase pemodelan. Dataset *training* digunakan untuk menghasilkan model klasifikasi melalui pembelajaran mesin algoritma,

sedangkan dataset *testing* bertujuan untuk mengukur kemampuan model klasifikasi untuk mengekstrapolasi label baru untuk observasi baru yang tidak termasuk dalam dataset *training*, jadi bisa diketahui apakah algoritma tersebut dapat diandalkan atau tidak untuk menghasilkan prediksi yang akurat. Proporsi dataset *training* adalah lebih besar dari dataset *testing* karena membangun model klasifikasi dengan varians rendah (Hanif, 2019).

Proporsi untuk data *training* dan *testing* menurut (Thalib, 2018) adalah tidak mengikat, tetapi agar variasi dalam model tidak terlalu besar maka disarankan data latih lebih besar dari data uji. Biasanya yang menghasilkan keakuratan model klasifikasi dengan proporsi 80%:20% dan 90%:10% untuk data latih dan data uji.

3.12 Klasifikasi

Klasifikasi adalah metode dalam *machine learning* yang digunakan oleh mesin untuk memilah atau mengklasifikasikan obyek berdasarkan ciri tertentu sebagaimana manusia mencoba membedakan benda satu dengan benda lainnya (Ahmad, 2017). Klasifikasi adalah bentuk analisis data yag mengekstraksi model yang menggambarkan kelas data atau memprediksi label kelas kategorikal. Klasifikasi memiliki banyak aplikasi, baik deteksi penipuan, target pemasaran, prediksi kinerja dan diagnosis medis (Han, Kamber, & Pei, 2012).

Klasifikasi menurut (Gorunescu, 2011) adalah proses menempatkan objek tertentu dalam satu set kategori, berdasarkan sifat objek yang bersangkutan. Berdasarkan empat komponen klasifikasi yaitu:

1. Kelas

   Kelas merupakan variabel dependen berupa data kategorikal yang merepresentasikan ‘label’ yang terdapat pada objek setelah klasifikasi.

   Contoh: kelas hujan dan kelas panas.

2. Prediktor

   Prediktor merupakan variabel independen yang diwakili oleh karakteristik data yang akan diklasifikasikan dan berdasarkan klasifikasi yang dibuat.

   Contohnya: Musim, waktu dan lokasi.

3. *Training Dataset*

Training dataset merupakan kumpulan data yang berisi nilai nilai komponen sebelumnya dan digunakan untuk melatih mdoel untuk mencari kelas yang sesuai didasarkan pada prediksi yang sesuai.

4. *Testing Dataset*

   Testing dataset merupakan data baru yang akan diklasifikasikan oleh model yang dibangun sebelumnya dan akurasi klasifikasi dapat dievaluasi.

3.13 *Ensemble Learning*

Berbagai metode masih belum sepenuhnya sempurna, banyak yang memiliki kekurangan, yaitu kurang stabil dan mudah terjebak pada kondisi *overfit*, ketika digunakan pada suatu kasus tertentu yang kurang sesuai dengan karakter metode-metode tersebut (Suyanto, 2018). Oleh karena itu, sejumlah peneliti mengusulkan teknik pembelajaran gabungan (*Ensemble learning*). *Ensemble learning* adalah pembelajaran mesin di mana banyak model sering disebut (*weak leaners*) dilatih untuk memecahkan masalah dan kemudian digunakan untuk membuat prediksi. Ketika memprediksi variabel target menggunakan teknik pembelajaran mesin, penyebab utama dalam perbedaan nilai aktual dan prediksi adalah *noise*, *varians*, dan bias. teknik *Ensemble* membantu mengurangi faktor-faktor ini (kecuali *noise*, yang merupakan kesalahan yang tidak dapat direduksi) (Rocca, 2019).

3.13.1 *Boosting*

Boosting pertama kali diusulkan oleh Robert E. Schapire pada tahun 1990 (Schapire , 1990). Sesuai dengan namanya, metode *boosting* bekerja dengan cara memperkuat (*boost*) sebuah model klasifikasi awal yang lemah. Konsep ensemble dengan boosting bekerja dengan cara melatih kelompok model secara sekuensial dan kemudian menggabungkan keseluruhan model tersebut untuk melakukan prediksi, model yang dihasilkan belajar dari kesalahan model sebelumnya (Zhou , 2012). Boosting mengubah model prediksi yang lemah menjadi prediktor kompleks yang handal. Tahapan dari proses belajar ini adalah memprediksi untuk regresi kemudian dilakukan kalkulasi atas kesalahan dari residu dan terakhir proses belajar untuk mengolah residu (Syahrani, 2019).

Menurut (Suyanto, 2018) *Pseudocode* metode *boosting* diilustrasikan pada Algoritma dibawah ini:

Masukan:

- Sebuah himpunan data sumber

$$S = ((x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)) \quad (3.2)$$

Inisialisasi:

- Maksimum iterasi *T*
- Inisialisasi distribusi bobot

$$\forall i \in \{1, \dots, m\}, D^{(1)}(i) = \frac{1}{m} \textbf{ for } t = 1 \; sampai \; T \textbf{ do} \quad (3.3)$$

- Bangunlah sebuah model klasifikasi

$$f_t : \Re^d \rightarrow \{-1, +1\} \quad (3.4)$$

menggunakan distribusi $D^{(t)}$

- Hitung

$$\varepsilon_t = \sum_{i:f_t(x_i) \neq y_i} D^{(t)}(i) \quad (3.5)$$

- Pilih

$$\alpha_t = \frac{1}{2} \, ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \quad (3.6)$$

- Perbarui distribusi bobot seluruh data

$$\forall i \in \{1, \dots, m\}, D^{(t+1)}(i) = \frac{D^{(t)}(i) e^{-a_t y_i f_t(x_i)}}{Z^{(t)}} \quad (3.7)$$

Dimana

$$Z^{(t)} = \sum_{i=1}^{m} D^{(t)}(i) e^{-a_t y_i f_t(x_i)} \quad (3.8)$$

adalah sebuah factor normalisasi sedemikian hingga $D^{(t+1)}$ menyisakan sebuah distribusi.

Keluaran:

- Model klasifikasi hasil kombinasi

$$\forall x, F(\mathrm{x}) = sign(\sum_{t=1}^{T} \alpha_t f_t(x)) \quad (3.9)$$

Keputusan akhir pada *Boosting* dilakukan dengan skema pembobotan tertentu, dimana model yang memiliki akurasi tinggi diberi bobot yang besar sedangkan model yang memiliki akurasi rendah diberi bobot yang kecil. Performansi *boosting* relative tinggi dibanding model tunggal yang berdiri sendiri. Secara umum performansi *boosting* lebih tinggi dibanding *bagging* (Quinlan, 1996).

3.13.2 *eXtreme Gradient Boosting* (*XGBoost*)

XGBoost adalah singkatan *eXtreme Gradient Boosting* diusulkan oleh (Chen & Guestrin, 2016) adalah proyek *open source* untuk menerapkan sistem pembelajaran mesin yang efisien, cepat dan terukur yang disebut *Gradient Tree Boosting* untuk berbagai masalah pembelajaran berdasarkan pada makalah *Greedy Function Approximation*: *A Gradient Boosting Machine* oleh (Friedman, 2001). *XGBoost* digunakan untuk masalah *supervised learning* dimana menggunakan data latih dengan beberapa fitur $\mathcal{X}_i$ untuk memprediksi variabel target $\mathcal{Y}_i$. Sebelum mempelajari pohon secara khusus, dimulai dengan meninjau elemen dasar dalam *supervised learning*. Model dan Parameter Model dalam *supervised learning* biasanya mengacu pada struktur matematika di mana prediksi $\mathcal{Y}_i$ dibuat dari input $\mathcal{X}_i$. Contoh umum adalah model linier, di mana prediksi diberikan sebagai $\hat{\mathcal{Y}}_i = \sum_j \theta_j \mathcal{X}_{ij}$, kombinasi linier fitur masukan bobot. Nilai prediksi dapat memiliki interpretasi yang berbeda, tergantung pada tugasnya, yaitu regresi atau klasifikasi. Parameter adalah bagian yang belum di tentukan yang perlu kita pelajari dari data. Dalam masalah regresi linier, parameter adalah koefisien θ. Biasanya digunakan θ untuk menunjukkan parameter (Chen & Guestrin, 2016).

Dengan pilihan bijak untuk $\mathcal{Y}_i$ dapat mengekspresikan berbagai tugas, seperti regresi, klasifikasi dan peringkat. Tugas melatih model adalah menemukan parameter terbaik θ yang paling sesuai dengan data pelatihan $\mathcal{X}_i$ dan label $\mathcal{Y}_i$. Untuk melatih model, perlu mendefinisikan fungsi tujuan untuk mengukur seberapa cocok model tersebut dengan data pelatihan. Karakteristik penting dari fungsi objektif adalah bahwa mereka terdiri dari dua bagian: *training loss* dan *regularization*:

$$obj(\theta) = L(\theta) + \Omega(\theta) \tag{3.10}$$

Dimana L adalah fungsi *training loss* dan Ω adalah istilah *regularization. Training loss* mengukur seberapa prediktif model terkait dengan data pelatihan. *L* adalah kesalahan kuadrat rata-rata, yang diberikan oleh:

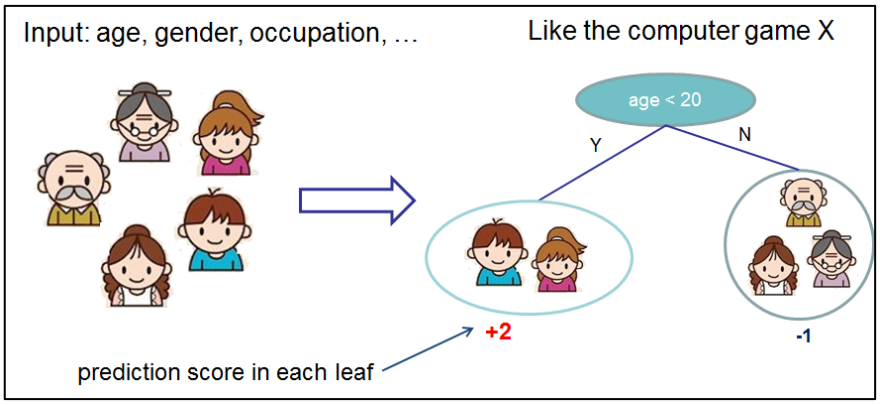
$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \quad (3.11)$$

Fungsi kerugian lain yang umum digunakan adalah kerugian logistik, yang akan digunakan untuk regresi logistik:

$$L(\theta) = \left[y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{-\hat{y}_i}) \right] \quad (3.12)$$

Istilah *regularization* adalah istilah yang biasanya lupa ditambahkan. Istilah *regularization* mengontrol kompleksitas model, yang membantu untuk menghindari overfitting (Chen & Guestrin, 2016).

Setelah memperkenalkan elemen-elemen supervised learning, mari kita mulai dengan pohon yang sebenarnya. *XGBoost* dikembangkan dari *Classification and Regression Trees* (CART) atau juga dengan nama lain *Decision Trees.* Dimana metode ini merupakan gabungan dari dua jenis pohon, yaitu *classification tree* dan juga *regression tree*. Jika variabel dependen yang dimiliki bertipe kategorik maka CART menghasilkan pohon klasifikasi (*classification tree*) sedangkan jika variabel dependen yang dimiliki bertipe kontinu atau numerik maka CART menghasilkan pohon regresi (*regression tree*). Berikut adalah contoh sederhana dari CART yang mengklasifikasikan apakah seseorang akan menyukai permainan komputer hipotetis X.



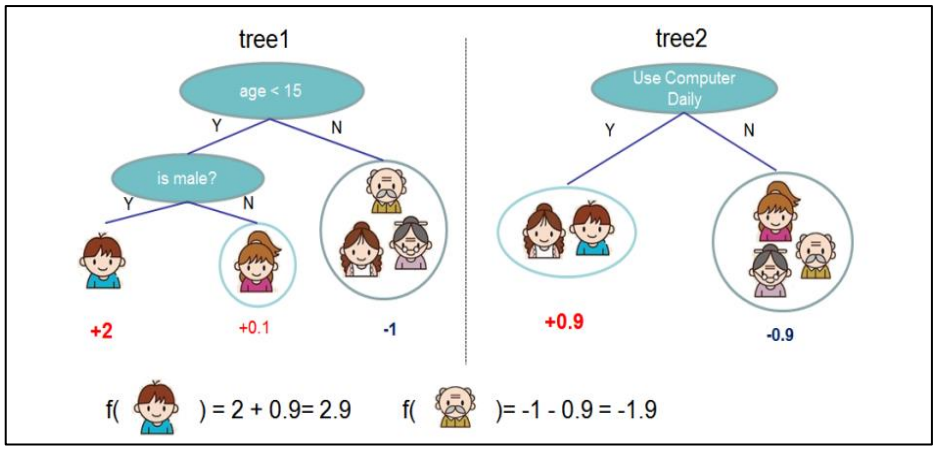
Gambar 3.3 Contoh *Classification and Regression Trees* (CART)
Sumber: (Chen & Guestrin, 2016)

Dari gambar 3.3 diatas mengklasifikasikan anggota keluarga ke dalam daun yang berbeda, dan memberikan skor pada daun yang sesuai. CART sedikit berbeda dari *decision trees*, di mana daunnya hanya berisi nilai keputusan. Di CART, skor nyata dikaitkan dengan masing-masing daun, yang memberi kita interpretasi yang lebih kaya yang melampaui klasifikasi. Ini juga memungkinkan untuk pendekatan berprinsip dan terpadu untuk pengoptimalan.

Pada gambar 3.4 adalah contoh tree ensemble dari dua pohon. Skor prediksi masing-masing pohon dijumlahkan untuk mendapatkan skor akhir. $\{R_1(\mathcal{X}_i, \mathcal{Y}_i), \dots, R_k(\mathcal{X}_i, \mathcal{Y}_i)\}$ dan klasifikasi C dimana $\mathcal{X}_i$ dan $\mathcal{Y}_i$ adalah label data training dan class. Skor prediksi dievaluasi dengan menggunakan fungsi aditif k sebagai berikut:

$$\hat{\mathcal{Y}}_i = \sum_{k=1}^{C} f_k(x_i)\,, f_k \in F \tag{3.13}$$

Dimana f_k adalah struktur pohon independent dengan skor daun dan $F = \{f(x) = w_{q(x)}\}(q : R^m \rightarrow T, w \in R^T)$ adalah ruang pohon regresi juga dikenal sebagai CART. Disini q mewakili struktur setiap pohon yang memetakan contoh ke index daun yang sesuai. T adalah jumlah daun di pohon. Setiap f_k sesuai dengan struktur pohon independent q dan bobot daun w. Tidak seperti pohon keputusan, setiap pohon regresi berisi skor terus menerus pada setiap daun, menggunakan w_i untuk mewakili skor pada daun ke-1. Sebagai contoh, kita akan menggunakan aturan keputusan di pohon (diberikan oleh q) untuk mengklasifikasikan.



Gambar 3.4 Model Tree Ensemble

Sumber: (Chen & Guestrin, 2016)

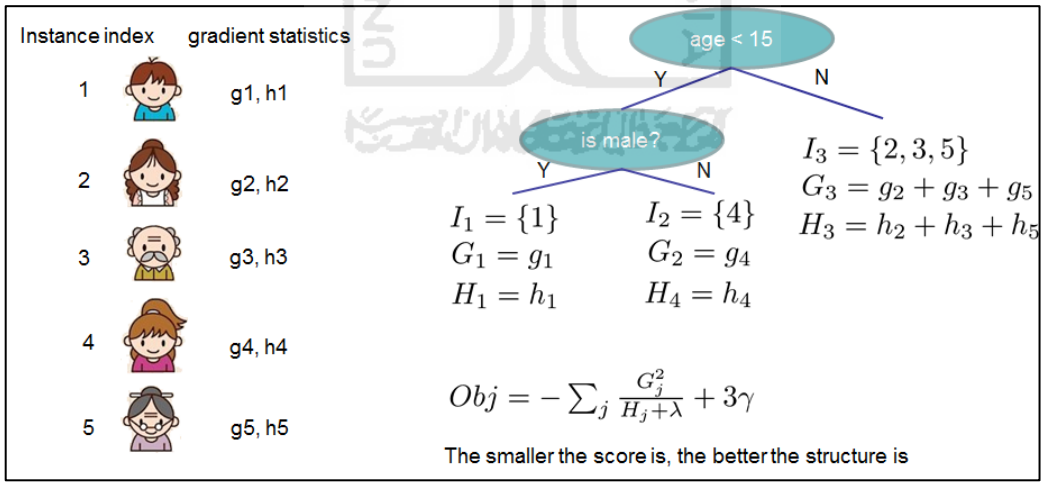
Prediksi terakhir untuk contoh yang diberikan adalah jumlah prediksi dari setiap pohon. Ke dalam daun dan hitung prediksi akhir dengan menjumlahkan skor dalam daun yang sesuai (diberikan oleh *w*). Kemudian sekumpulan fungsi yang digunakan dalam model diminimalkam seperti berikut ini.

$$\mathcal{L}(\emptyset) = \sum_i \iota(\hat{y}_i, y_i) + \sum_\kappa \Omega(f_\kappa) \quad (3.14)$$

$$\text{Dimana } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Disini ι adalah fungsi kerugian cembung yang dapat dibedakan perbedaan antara $\hat{y}_i$ prediksi dan target y_i istiah kedua Ω menghukum kompleksitas model (misalnya fungsi pohon regresi) istilah regulasi tambahan membantu memuluskan bobot akhir yang dipelajari untuk dihindari terlalu pas. Secara intuitif, tujuan yang diatur akan cenderung untuk memilih model yang menggunakan fungsi sederhana dan prediktif. Teknik regulasi serupa telah digunakan dalam model *Regularization Greedy Forest* (RGF) (Zhang & Johnson, 2014).

Pada gambar 3.5 adalah Perhitungan Skor Struktur. Dengan menjumlahkan gradien dan statistik gradien urutan kedua pada setiap daun, lalu menerapkan rumus penilaian untuk mendapatkan skor kualitas.



Gambar 3.5 Skor Struktur

Sumber: (Chen & Guestrin, 2016)

Dari gambar 3.5 dan lihat bagaimana skor dapat dihitung. Pada dasarnya, untuk struktur pohon tertentu, mendorong statistik g_i dan h_i ke daunnya, menjumlahkan statistiknya dan menggunakan rumus untuk menghitung seberapa

bagus pohon tersebut. Skor ini seperti ukuran ketidakmurnian dalam pohon keputusan, kecuali bahwa itu juga memperhitungkan kompleksitas model. Akhirnya mendapatkan informasi dari fungsi objektif setelah setiap pemisahan adalah:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda} + \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}\right] - \gamma \qquad (3.15)$$

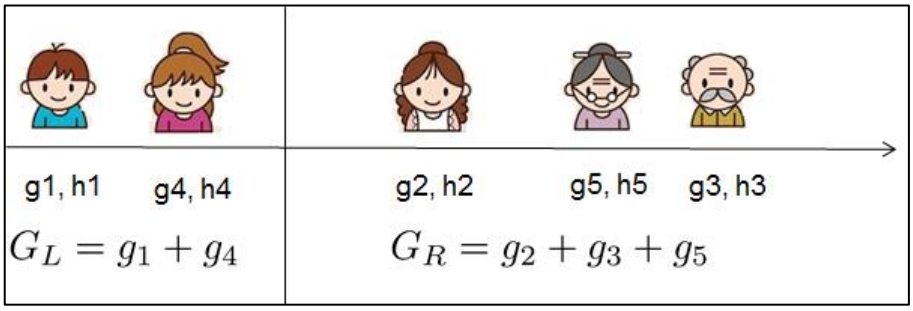
Seperti yang dapat dilihat dari (3.15), untuk menekan pertumbuhan pohon dan mencegah model overfitting, ditambahkan ambang pemisah γ. Node daun diijinkan untuk membagi jika dan hanya jika informasi yang diperoleh lebih besar dari γ. Ini sama dengan pra-penetapan harga pohon sambil mengoptimalkan fungsi objektif, Selain itu, terdapat dua teknik terbaik *XGBoost* berikut untuk menghindari overfitting dalam percobaan:

1. Jika semua bobot sampel pada node daun kurang dari ambang batas, pemisah dihentikan. Ini mencegah model dari mempelajari sampel pelatihan khusus.
2. Fitur sampel secara acak ketika membangun setiap pohon.

Semua metode ini membuat *XGBoost* lebih di generalisasikan dan mendapatkan kinerja yang lebih baik dalam aplikasi praktis. Sekarang memiliki cara untuk mengukur seberapa bagus pohon itu, idealnya akan menghitung semua pohon yang mungkin dan memilih yang terbaik. Dalam praktiknya, hal ini sulit dilakukan, jadi mencoba mengoptimalkan satu tingkat pohon pada satu waktu. Secara khusus dengan mencoba membelah daun menjadi dua daun, dan skor yang diperolehnya adalah:

$$Gain = \frac{1}{2}\left[\frac{{G^2}_L}{H_L+\lambda} + \frac{{G^2}_R}{H_R+\lambda} + \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \qquad (3.16)$$

Rumus ini dapat diuraikan sebagai 1) skor pada daun kiri baru 2) skor pada daun kanan baru 3) skor pada daun asli 4) regularisasi pada daun tambahan. dapat melihat fakta penting di sini: jika gain lebih kecil dari γ, sebaiknya tidak menambahkan cabang tersebut. Inilah teknik pemangkasan dalam model berbasis pohon dengan menggunakan prinsip-prinsip supervised learning. Untuk data bernilai nyata, biasanya dicari pemisahan yang optimal.



Gambar 3.6 Struktur Blok

Sumber: (Chen & Guestrin, 2016)

Untuk melakukannya secara efisien, dengan menempatkan semua instance dalam urutan yang diurutkan, seperti gambar 3.6 skema pemilihan pemisahan terbaik pemindaian kiri ke kanan cukup untuk menghitung skor struktur dari semua solusi pemisahan yang mungkin dan dapat menemukan pemisahan terbaik secara efisien.

3.13.3 *Tunning Hyperparameter*

Algoritma pembelajaran mesin seperti *deep neural networks, gradient boosting models*, dan lainnya. Terdiri dari berbagai *hyperparameter* yang membutuhkan penyetelan untuk kinerja algoritma yang optimal. *Optimasi Hyperparameter* adalah ilmu menyetel *hyperparameter* dari algoritma untuk mendapatkan kinerja yang optimal Algoritma yang berbeda memiliki tipe *hyperparameter* yang berbeda pula. Dampak *optimasi hyperparameter* pada kinerja dari algoritma pembelajaran mesin telah terbukti keduanya secara teoritis dan empiris oleh banyak penelitian yang terdapat dalam literatur. Itu adalah tugas yang membosankan dan memakan waktu jika dilakukan dengan pencarian manual, beberapa pendekatan umum untuk mengatasi ini termasuk *grid search, random search* dan alternatif lain adalah melakukan optimasi Bayesian (Putatunda & Rama, 2018). Seperti beberapa parameter utama hyperparameter untuk meningkatkan algoritma *eXtreme Gradient Boosting* (*XGBoost*) yaitu:

1. *n_estimator*

   n_estimator adalah jumlah iterasi dalam *training*. *n_estimator* yang terlalu kecil dapat menyebabkan *underfitting*, yang membuat model tidak

sepenuhnya melakukan kemampuan belajarnya. Namun, *n_estimator* yang terlalu besar juga biasanya tidak bagus karena akan menyebabkan *overfitting*.

2. *min_child_weight*

   adalah untuk menentukan sampel berat node daun terkecil untuk mencegah *overfitting*.

3. *max_depth*

   adalah kedalaman maximum pohon. Semakin besar kedalaman pohon, semakin kompleks model pohonnya, dan semakin kuat kemampuan pemasangannya, tetapi pada saat yang sama, modelnya jauh lebih mudah untuk dikenakan.

4. *subsample*

   parameter ini berarti laju sampling dari semua sampel *training*.

5. *colsample_bytree*

   adalah laju pengambilan sample fitur ketika membangun setiap pohon. Dalam tugas ini, setara dengan laju pengambilan sampel *landmark gene*.

6. *learning_rate*

   adalah parameter yang sangat penting yang perlu di sesuaikan, juga dalam *XGBoost*. Ini sangat mempengaruhi kinerja model. Kita bias mengurangi bobot setiap langkah untuk membuat model lebih kuat.

Grid search adalah salah satu metode yang paling umum digunakan untuk *optimasi hyperparameter*, ini adalah jenis teknik *brute force*. Idenya adalah untuk mencari melalui subset dari *hyperparameter* untuk algoritma *machine learning* yang bersangkutan. Sehingga ruang pencarian ditentukan dan tujuannya adalah untuk mengoptimalkan kinerja algoritma *machine learning*. Ini adalah metode yang sangat sederhana tetapi dengan peningkatan *hyperparameter*, biaya komputasi meningkat secara *eksponensial*. Beberapa keuntungan lainnya dari metode *grid search* adalah kemudahan implementasi dan paralelisme (Schaer, Muller, & Depeursinge, 2016).

3.13.4 *Confusion Matrix*

Metode yang digunakan dalam menilai tingkat akurasi dari model adalah *confusion matrix* (CM). CM memiliki informasi pembanding antara hasil klasifikasi

dari keluaran model dan hasil klasifikasi sebenarnya. Empat komponen yang merepresentasikan hasil klasifikasi dalam CM yakni *True Positive* (TP), *True Negative* (TN), *False Postive* (FP) dan *False Negatif* (FN). TN berupa jumlah data negatif yang bernilai benar, FP merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, TP merupakan data positif yang terdeteksi benar. FN merupakan kebalikan dari *True Positive*, sehingga datanya bernilai positif, namun terdeteksi sebagai data negatif (Syahrani, 2019). Isi dari *confusion matrix* dapat dilihat pada tabel 3.2.

Tabel 3.2 *Confusion Matrix*

Confusion Matrix		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	*TP (True Positive) Correct result*	*FP (False Positive) Unexpected result*
	FALSE	*FN (False Negative) Missing result*	*TN (True Negative) Correct absence of result*

Sumber : (Sofi & Jajuli, 2015)

Berikut formulasi untuk menghitung nilai *accuracy*, *precision*, *recall/sensitivity*, *specificity*, FPR dan *AUC* pada pembentukan model klasifikasi:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}\ x\ 100\% \quad (3.17)$$

$$Precision = \frac{TP}{TP+FP}\ x\ 100\% \quad (3.18)$$

$$Recall/Sensitivity = \frac{TP}{TP+FN}\ x\ 100\% \quad (3.19)$$

$$Specificity = \frac{TN}{TN+FP}\ x\ 100\% \quad (3.20)$$

$$FPR = 1 - Specificity \tag{3.21}$$

$$AUC = \frac{1+sensitivity-FPR}{2} \tag{3.22}$$

AUC mengukur kinerja diskriminatif dengan memperkirakan probabilitas *output* dari sampel yang dipilih secara acak dari populasi positif atau negatif.

Tabel 3.3 Klasifikasi *AUC*

Nilai *AUC*	Keterangan
0.91 – 1.00	Klasifikasi sangat baik
0.81 – 0.90	Klasifikasi baik
0.71 – 0.80	Klasifikasi cukup
0.61 – 0.70	Klasifikasi buruk
≤ 0.60	Klasifikasi salah

Sumber: Sofi & Jajuli (2015)

Dengan keterangan klasifikasi seperti pada tabel 3.2. Nilai *AUC* akan selalu diantara 0 dan 1, semakin besar nilai *AUC* maka semakin kuat klasifikasi yang digunakan.

BAB IV
METODOLOGI PENELITIAN

4.1 Data

Sumber data dalam penelitian ini berupa data sekunder yang diperoleh dari *website National Center of Biotechnology Information* (NCBI) yang bersumber dari penelitian (Reynard S, Baize S, Varet H, Pietrosemoli N, 2019) dalam bentuk jenis percobaan *Expression profiling by array* dengan Platform data yang di akses adalah GPL16686 dengan *series* GSE122692 yang berjudul "*Immune parameters and outcome durring Ebola virus disease"* diakses pada tanggal 10 Januari 2020 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE122692) dimana data yang digunakan berisi gene expression dari parameter dan hasil pola kekebalan pada pasien virus Ebola dengan jumlah sampel sebanyak 59.

4.2 Tempat dan Waktu Penelitian

Penelitian ini dilakukan di kampus terpadu Universitas Islam Indonesia yang bertempat di Yogyakarta. Waktu pengambilan data dan penelitian ini dilakukan pada bulan Februari 2020 hingga bulan Agustus 2020.

4.3 Variabel dan Definisi Operasional Variabel

Beberapa variabel yang digunakan dalam penelitian ini dijelaskan pada tabel 4.1 sebagai berikut :

Tabel 4.1 Definisi Operasional Variabel

Variabel	Definisi Operasional Variabel
Gen	Nama gen yang mempunyai nilai intensitas ekspresi gen jaringan Fatalities, Viremic suvivors, Suvivors in recovery phase dan Healthy control pada pasien virus Ebola.

Variabel	Definisi Operasional Variabel
Jaringan	Variabel sekumpulan sel yang terdapat pada pasien virus Ebola dan memiliki empat jaringan berbeda yaitu jaringan jaringan Fatalities, Viremic suvivors, Suvivors in recovery phase dan Healthy control.

4.4 Metode Analisis Data

Metode yang diterapkan dalam penelitian ini adalah metode klasifikasi *machine learning* dengan menggunakan algoritma *Boosting* (*XGBoost*). Dengan menggunakan aplikasi *RStudio* dengan versi 4.0.0. Ada beberapa langkah yang harus dilakukan sebelum melakukan analisis terhadap data ekspresi gen:

1. Membaca data

   Paket *Bioconductor* untuk analisis data yang muncul dari genomic dan biologi mulekuler dengan menggunakan *package* `oligo` untuk membaca data file *Affymetrix CEL*.
2. *Preprocessing*

   Dengan menggunakan algoritma *Robust Multi-Array* (*RMA*) untuk meringkas ekspresi level gen, anotasi probe untuk *array* spesifik normalisasi harus diperlukan.
3. *Gene Annotation*

   Untuk interpretasi lebih lanjut dari hasil. Diperlukan dua paket *Bioconductor* yaitu `biobase` dan `hugene20sttranscriptcluster.db`.
4. *Gene Filtering*

   Untuk menghapus set probe dengan varian rendah di semua *array* dengan perintah `varFilter` dan `eset.filt` yang berasal dari *packages* `genefilter`.
5. *Feature Selection*

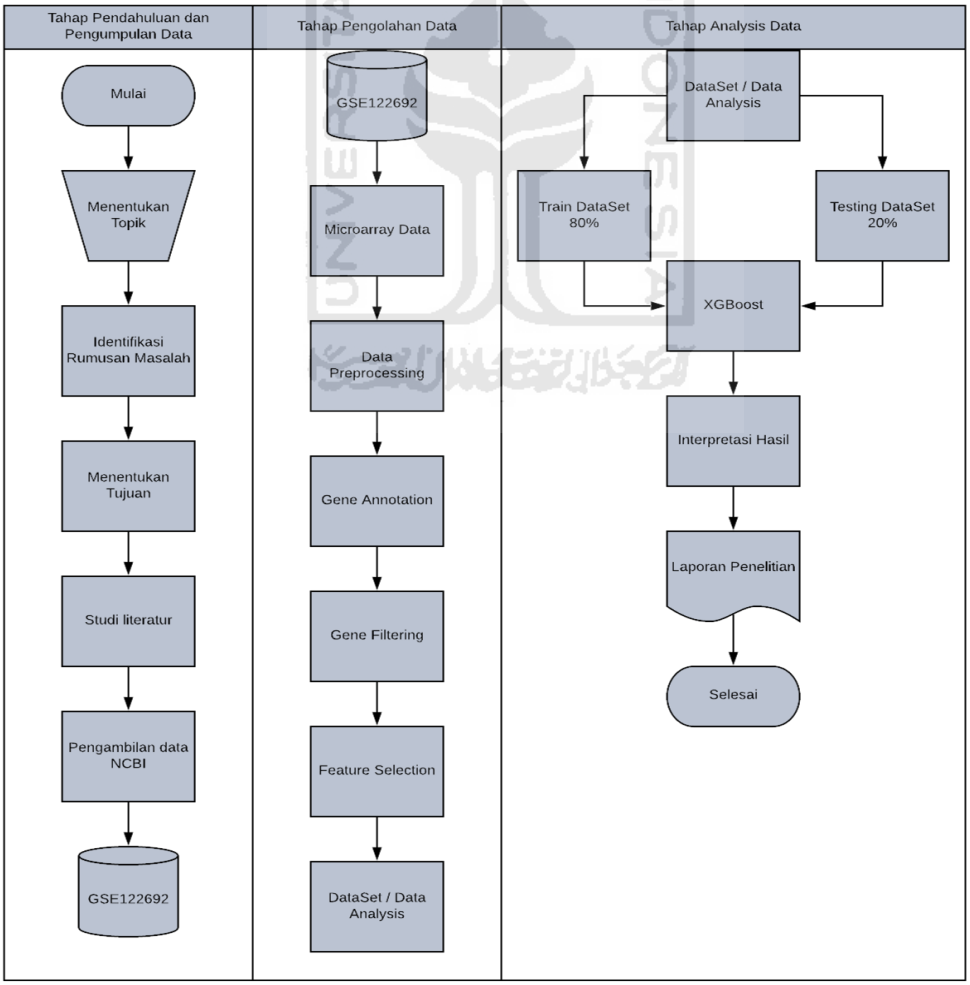
Dengan menggunakan bantuan *package* `multtes` yang ada pada Bioconductor, dan peneliti menggunakan uji t karena ingin membandingkan rata-rata dua sampel.

6. Data Analisis

   Peneliti menggunakan analisis klasifikasi *machine learning* dengan metode *Extreme Gradient Boosting* menggunakan *package* `xgboost` untuk membangun model dan `caret` untuk *cross-validation* dan *grid search*.

4.5 Langkah Penelitian

Tahapan pada penelitian ini di mulai dari menentukan topik penelitian hingga diperoleh hasil dan membuat laporan penelitian seperti yang digambarkan pada *flowchart* pada gambar 4.1 dibawah ini berikut:



Gambar 4.1 *Flow Chart*

Keterangan:

1. Langkah pertama yang dilakukan peneliti adalah menentukan topik yaitu tentang bioinformatika, mencari studi pustaka dengan mempelajari penelitian-penelitian terdahulu mengenai bioinformatika dan mendapatkan referensi penelitian yang akan dilakukan oleh peneliti dalam penerapan metode *XGBoost* dan menentukan tujuan dari penelitian.
2. Peneliti mengambil data di *website National Center of Biotechnology Information* (NCBI) dan mendapatkan data dari parameter dan hasil pola kekebalan pada pasien virus Ebola yang diberi kode akses GSE122692.
3. Kemudian masuk kedalam tahapan penginputan data, dengan menggunakan *software Rstudio* dibantu dengan beberapa *package* bioinformatika dari *Bioconductor* pada tahap ini pembacaan data dibantu dengan menggunakan *package* `oligo` untuk membaca data file *Affymetrix CEL* dan `pd.hugene.2.0.st` untuk mengambil data microarray.
4. Setelah membaca data, peneliti melakukan *preproessing* yaitu tahap dimana pemilihan variabel dalam data dan menyeleksi fitur yang akan digunakan sebelum melakukan analisis klasifikasi dengan menggunakan algoritma *Robust Multi-Array* (*RMA*), *gene annotation* untuk interpretasi lebih lanjut dari hasil, *filtering* untuk menghapus set probe dengan varian rendah di semua *array* dan *feature selection* untuk membuat pengklasifikasian lebih efektif dengan cara mengurangi data-data yang tidak relevan.
5. Tahap selanjutnya adalah memasukkan data yang telah diproses kedalam data expression set yang baru dan merubah kelas menjadi numerik, sehingga data berbentuk frame. sebelum melakukan analisis klasifikasi *XGBoost* adalah pembagian data menjadi dua yaitu pembagian data *train* dan data *test*, dengan *ratio* 80% : 20%.
6. Pada tahap analisis klasifikasi *XGBoost* yang pertama adalah ubah data *training* dan *testing* set menjadi matrix, kemudian mendapatkan hasil dengan metode *cross-validation* dan setelah dilakukan *tunning hyperparameter* secara *grid search*.

BAB V
HASIL DAN PEMBAHASAN

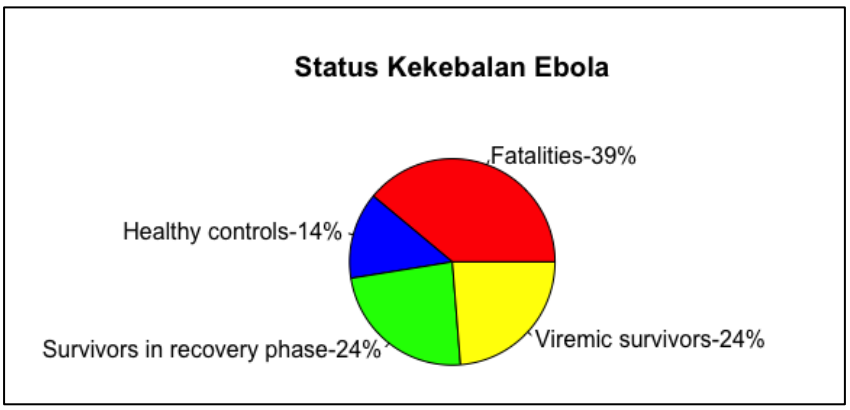
5.1 Analisis Deskriptif

Data GSE122692 merupakan kumpulan sampel data ekspresi array dari pola kekebalan pada pasien Ebola ditandai tergantung pada hasil penyakit. Data GSE122692 dijelaskan pada tabel 5.1 didapatkan melalui perintah `list.celfiles` untuk mendapatkan daftar semua CEL di direktori dan `read.celfiles` digunakan untuk membaca data yang berbentuk cel ke dalam memori.

Tabel 5.1 *Dataset* pasien Ebola Virus

Informasi	Keterangan
Anotasi	pd.hugene.2.0.st
Gen	2598544
Sampel	59
Kelas	4

seperti annotation, jumlah gen dan jumlah sampel gen merupakan kumpulan data ekspresi gen sebanyak 2598544 nama gen dan 59 sampel pola kekebalan pasien virus ebola yang dibedakan berdasarkan sampel jaringan *fatalities, viremic suvivors, suvivors in recovery phase* dan *healthy contro*l.



Gambar 5.1 *Pie Chart* Kelas Pasien Virus Ebola

Pasien virus Ebola yang berasal dari *Guinea* (Afrika Barat) yang diteliti dimana seperti gambar 5.1 dapat diperoleh informasi mengenai imun selama

penyakit virus Ebola. Eksperimen chip RNA dilakukan pada 59 sampel dan dibagi menjadi 4 kelas yaitu jaringan *fatalities*, *viremic suvivors*, *suvivors in recovery phase* dan *healthy contro*l. Sebanyak 23 sampel mempunyai jaringan *fatalities* pada pasien virus Ebola (39%), 14 sampel mempunyai jaringan *viremic suvivors* pada pasien virus Ebola (24%), 14 sampel mempunyai jaringan *suvivors in recovery phase* pada pasien virus Ebola (24%) dan 8 sampel mempunyai jaringan *healthy contro*l pada pasien virus Ebola (14%). Pola kekebalan tergantung pada hasil penyakit. Korban yang selamat menunjukan respon imun yang efisien dan seimbang, sedangkan kematian ditandai dengan respon inflamasi yang intens, ekspresi berlebih dari berbagai sitokin, dan "*chemokine storm*". Tidak ada perbedaan yang signifikan antara usia rata-rata, rasio jenis kelamin, atau waktu timbulnya gejala dan masuk ketika membandingkan kematian dan orang yang selamat.

Tabel 5.2 Informasi Lainnya Pada Data

Informasi	Keterangan
Sample_type	RNA
Sample_source_name_ch1	blood cells
Sample_organism_ch1	Homo sapiens
Sample_characteristics_ch1	tissue: leukocytes

Berdasarkan Tabel 5.2 menunjukkan tentang variabel-variabel yang ada pada series data, jumlah pheno dalam data sebanyak 29 variabel tetapi variabel lain hanya penjelasan tentang catatan-catatan informasi tentang pasien. variabel terpilih yang akan menjadi fokusan terhadap analisis yang akan dilakukan adalah variabel Gen, Sample dan Kelas.

5.2 Pengolahan Data Bioinformatik

Data Bioinformatik dalam proses pengolahannya dengan program RStudio memerlukan bantuan *package* dari Bioconductor. Data yang dapat diolah dengan menggunakan RStudio adalah data *vector*, *matrix*, *data frame* atau yang lainnya. Untuk membaca data bionformatik tersebut memerlukan bantuan *package* `oligo`

untuk membaca data file *Affymetrix CEL*. Data tersebut menyimpan berbagai macam informasi dari pasien yang dijadikan sampel penelitan. Informasi mengenai pasien tersimpan didalam pheno data seperti gambar berikut :

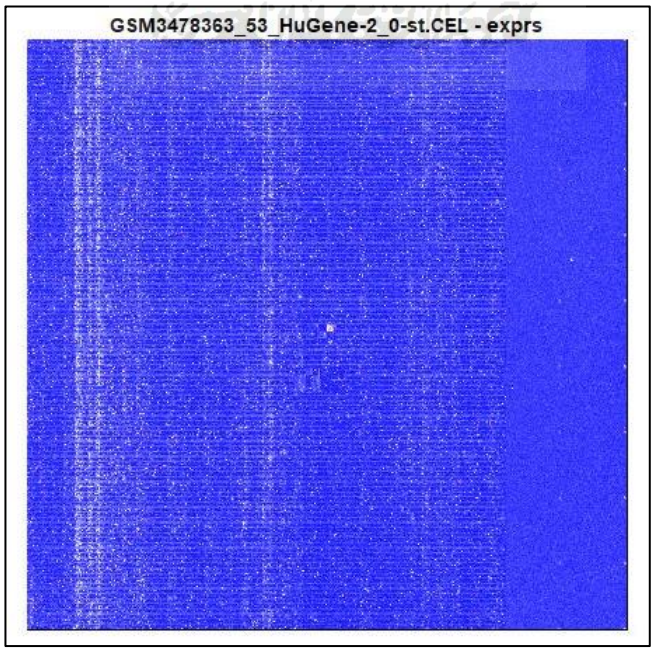
```
> varLabels(phenoData(gset[[1]]))
 [1] "title"                "geo_accession"          "status"                 "submission_date"
 [5] "last_update_date"     "type"                   "channel_count"          "source_name_ch1"
 [9] "organism_ch1"         "characteristics_ch1"    "characteristics_ch1.1"  "molecule_ch1"
[13] "extract_protocol_ch1" "label_ch1"              "label_protocol_ch1"     "taxid_ch1"
[17] "hyb_protocol"         "scan_protocol"          "data_processing"        "platform_id"
[21] "contact_name"         "contact_institute"      "contact_address"        "contact_city"
[25] "contact_zip/postal_code" "contact_country"     "supplementary_file"     "data_row_count"
[29] "group:ch1"            "tissue:ch1"
```

Gambar 5.2 *Pheno* Data

Dari gambar 5.2 tersebut dapat diperoleh informasi tentang pasien yang dijadikan sampel penelitian, dimana pasien diberikan kode GSM yang mencakup geo_accession, status, hingga jaringan pasien. Keseluruhan gambar 5.2 mengenai pheno data mempunyai sampel sebanyak 59 dengan 29 variabel yang diketahui.

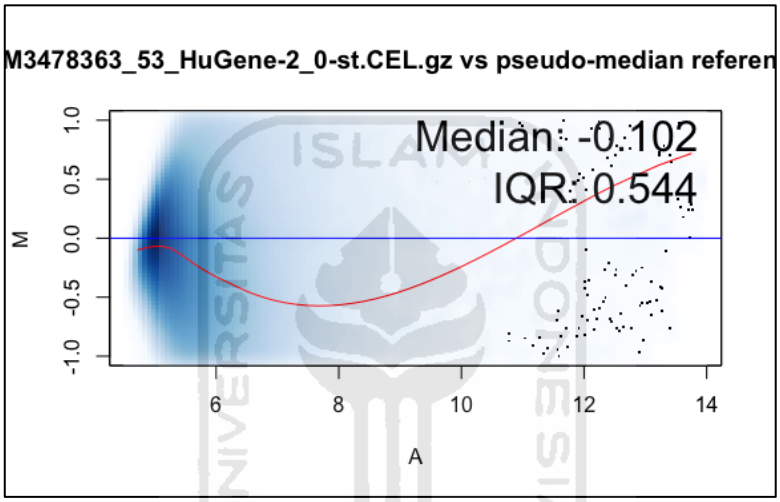
5.2.1 *Visualization* dan *QC Tools*

Chip *pseudo-image* di *affy* sangat berguna untuk mendeteksi perbedaan spasial pada *array* invidual tetapi tidak untuk membandingkan antar *array*. *pseudo-image* berdasarkan residu atau bobot yang dihasilkan dari perbandingan model (data ideal, tanpa adanya *nois*) ke data aktual. Bobot atau residu ini dapat ditampilkan secara grafis menggunakan fungsi image () di *Bioconductor.*

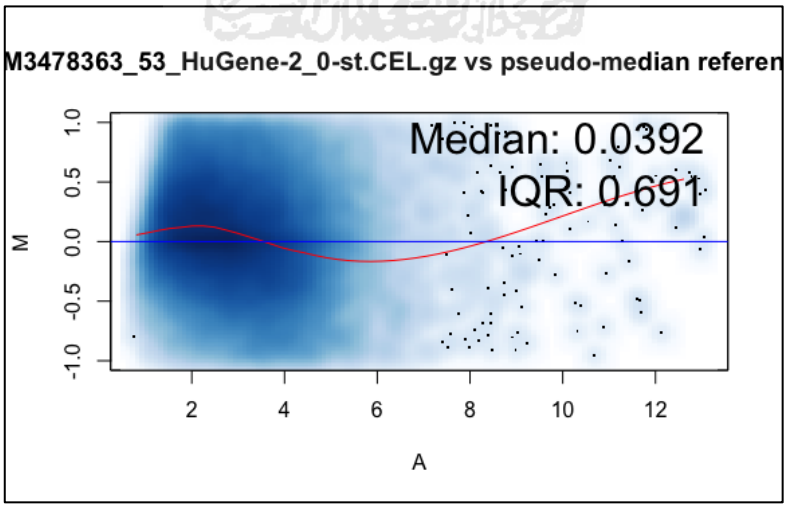


Gambar 5.3 *Pseudo-Image*

Gambar 5.3 menunjukan bagaimana metode gambar untuk menghasilkan sampel *pseudo-image*, dalam plot ini peneliti menggunakan sampel pertama sebagai contoh GSM3478363_53_HuGene-2_0-st.CEL.gz_exprs. *Pseudo-image* digunakan untuk penilaian visual dari array, untuk sampel *Pseudo-image* dihasilkan dengan memasang *Probe Level Model* (PLM) ke data yang mengasumsikan bahwa semua probe set beperilaku sama dalam sampel yang berbeda. Peneliti menggunakan $\log_2 - scale$ *Pseudo-image* untuk *rank of the observation* dengan mengatur argument *transfo* untuk metode gambar.



Gambar 5.4 *MA-Plot* sebelum Normalisasi



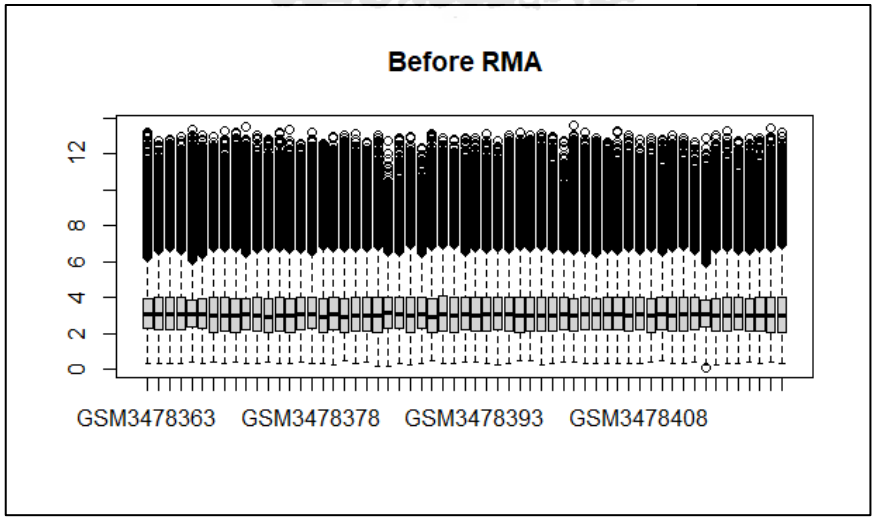
Gambar 5.5 *MA-Plot* Sesudah Normalisasi

MA-Plot adalah plot rasio intensitas log (nilai-M) versus rata-rata intensitas log (nilai-A) untuk dua objek. Titik data harus dipusatkan di sekitar M = 0 (garis

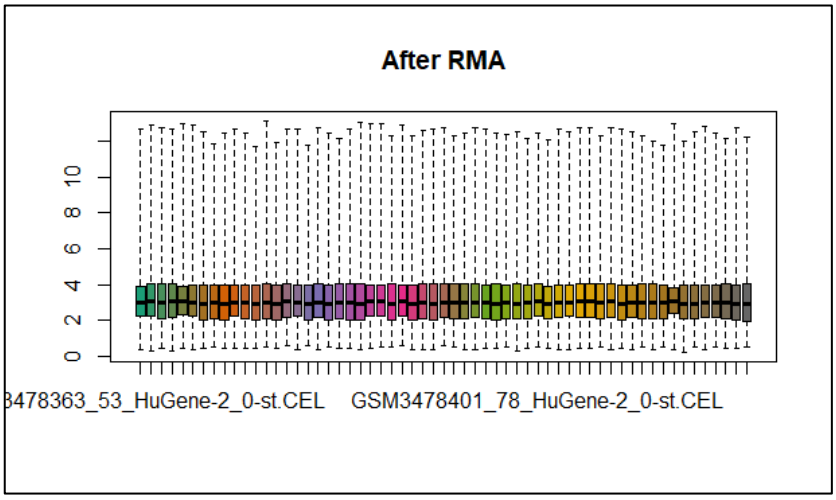
biru). Pada gambar 5.4 dan gambar 5.5 diatas didapatkan *MA-Plot* untuk GSM3478363 membandingkan hasilnya dengan *pseudo-image reference chip* yang menjelaskan bahwa GSM3478363 adalah faktor dan memberikan gambaran singkat tentang distribusi data. Pada diagram tersebut terlihat garis merah berjalan dibawah tanda nol sebelum normalisasi, maka perlu dilakukan normalisasi pada data sehingga garis merah lurus mendekati pada garis biru.

5.2.2 *Preprocessing*

Pada tahap *preprocessing* digunakan untuk menghilangkan nilai non biologis dan *noise* pada data. Untuk meringkas ekspresi level gen, Tahap *preprocessing* memerlukan bantuan *package* `affyPLM`. Paket yang memperluas dan meningkatkan fungsionalitas paket *affy* dasar. Beberapa anotasi probset diperlukan untuk array yang spesifik. Pada penelitian ini data microarray hugene.2.0.st yang memerlukan *package Bioconductor* `pd.hugene.2.0.st` kemudian dilakukan normalisasi *Robust Multi-Array Average* (RMA) adalah algoritma normalisasi yang paling banyak digunakan. dengan fungsi `rma()`. Fungsi ini mengubah objeck *AffyBatch* menjadi objek *ExpressionSet* dengan menggunakan RMA yang kuat. Karena pada data memiliki anotasi *Gene ST Affymetrix* maka memerlukan bantuan fungsi anotasi `getNetAffx` ini hanya tersedia untuk *array ExonST* dan *Gene ST*.



Gambar 5.6 *Boxplot* sebelum *preprocessing*



Gambar 5.7 *Boxplot* setelah *preprocessing*

Argumen "*type*" harus sesuai target *summarization* yang digunakan untuk menghasilkan "objek". Metode "RMA" memungkinkan untuk dua target yaitu "probset" dan "transcript". Pada gambar 5.6 merupakan *boxplot* ekspresi gen yang belum dilakukan *preprocessing* dan gambar 5.7 merupakan *boxplot* hasil dari *preprocessing*. Dimana *boxplot* digunakan untuk mengetahui sebaran nilai data pengamatan. Tahapan *preprocessing* pada data ekspresi gen digunakan untuk menghapus nilai non biologis, sehingga yang ada pada *boxplot* gambar 5.7 hanya ada data yang bersifat biologis berupa nilai ekspresi gen, menyebabkan *boxplot* mempunyai nilai yang rata-rata sama.

5.2.3 *Filtering*

Filtering data *microarray* adalah proses pemilihan subset dari *probe* yang tersedia untuk pengecualian atau penyertaan dalam analisis, dengan menggunakan program R dapat menggunakan bantuan *packages* `genefilter` untuk melakukan penyaringan, dan bantuan *function* `varFilter`. Untuk melakukan *filtering* ini peneliti melakukan cara menghapus set *probe* dengan ekspresi rendah dan varians rendah di semua *aray* dengan varian dibawah 0,25 kuantil.

Setelah dilakukan *filtering* dengan menghapus data dengan nilai IQR tertentu, menghapus ID gen Entrez yang terduplikat dan menyaring probe kontrol dimana pada data ekspresi gen ini menggunakan probe kontrol AFFX untuk tidak diikutsertakan dalam analisis.

Tabel 5.3 Jumlah data setelah *filtering non spesific filter*

nsFilter	
Gen	Sampel
40212	59

Data pada tabel 5.3 yang telah difilter dengan *function* `nsFilter` dengan nilai `var.cutoff` sebesar 0.25 karena perbedaan standar adalah versi statistik (Cohen, 1998) untuk ukuran nilai `cutoff` , menghasilkan data berdimensi 40212 x 59.

5.2.4 *Feature Selection*

Untuk melakukan *feature selection* dapat menggunakan bantuan *package* `multtest` yang dapat diolah jika data berbentuk *matrix* atau *data frame*. Dengan cara merubah data *AffyBatch* menjadi *matrix* atau *data frame* dengan menggunakan *function* `exprs`, sehingga data yang sebelumnya berbentuk *AffyBatch* menjadi data *matrix* atau *data frame*, kemudian dilakukan *multiple testing* dengan bantuan *function* `mt.teststat`.

Tabel 5.4 Jumlah data untuk dianalisis

mt.teststat	
Gen	Sampel
184	59

Pada table 5.4 tahapan *multiple testing* klasifikasi data pada penelitian diasumsikan berdistribusi normal, yang dilihat pada qqnorm. Fungsi ini memberikan cara mudah untuk menghitung nilai statistik uji dengan menggunakan uji Anova karena data mempunyai empat kelas antara jaringan *fatalities, viremic suvivors, suvivors in recovery phase* dan *healthy contro*l pada pasien virus ebola, dengan menggunakan nilai signifikansi sebesar 10^{-11}. Hasil filtering akhir memberikan data dengan dimensi 184 x 59. Data inilah yang akan digunakan kedalam klasifikasi *XGBoost* dengan *tunning hyperparameter* secara *grid search.*

5.2.5 Mengolah Data

Selesai melakukan *preprocessing* dan *filtering* yang telah dijelaskan menghasilkan suatu data *expression set*. Tahap selanjutnya adalah merubah data

expression set menjadi data *matrix* atau *data frame* dengan bantuan *function* `exprs` agar dapat dianalisis lebih lanjut dengan klasifikasi *Boosting XGBoost.* Data yang sudah berbentuk *data frame* atau data *matrix* merupakan syarat untuk mengolah data sebelum melakukan klasifikasi, penulis harus melakukan pembagian data *training* dan data *testing*.

Tabel 5.5 Ratio Data *Training* dan *Testing*

	Training	*Testing*
Ratio	80%	20%
Jumlah	46	13

Dimana data *training* merupakan kumpulan dataset yang akan digunakan untuk membentuk model klasifikasi dalam memprediksi kelas data baru dan data *testing* merupakan kumpulan dataset baru yang akan digunakan untuk mengukur sistem dapat melakukan klasifikasi dengan benar. Penulis menggunakan ratio untuk data latih dan data uji sebesar 80% : 20%. dengan jumlah sample sebanyak 59 dari data. Jumlah masing-masing pembagian data *training* dan *testing* dijelaskan pada table 5.5.

5.3 Klasifikasi *Boosting XGBoost*

Pada penelitian ini bertujuan untuk mengklasifikasian sampel pasien virus Ebola berdasarkan nilai ekspresi gen. *XGBoost* (*Extreme Gradient Boosting*) adalah algoritma *boosting* berdasarkan pada *Gradient Boosting Machine*. *XGBoost* menerapkan teknik regulasi untuk mengurangi *overfitting* dan itu adalah salah satu pembeda dari *Gradient Boosting*, keuntungan yang lainnya adalah kecepatan eksekusi yang cepat dan berkinerja baik dalam permodelan prediktif klasifikasi. Pada proses pengklasifikasian ini peneliti mengimplementasikan nya menggunakan RStudio 4.0.0 dengan bantuan *package* `keras`, `caret`, `dplyr` dan `xgboost` untuk melakukan klasifikasi *XGBoost.* Setelah didapatkan data *training* dan data *testing* kemudian memisahkan bagian fitur-X dan label-Y, Algoritma *XGBoost* membutuhkan data dengan tipe matriks. Disini peneliti menggunakan fungsi `xgb.DMatrix()` untuk membuat dataset kelas *xgb.DMatrix*. Keuntungan dari

matriks ini adalah dapat mengirimkannya variabel dan label serta mengidentifikasi kolom mana yang merupakan label. Dataset *testing* tidak digunakan dalam pemasangan model karena mendapatkan *cross-validation error estimate* dari data *training*, sementara dataset *testing* digunakan sebagai *hold-out validation* untuk model akhir yang cocok untuk semua data *training*. Kemudian peneliti membandingkan dengan parameter yang diambil setelah dilakukan *tunning hyperparameter* secara *grid search*.

5.3.1 Model *XGBoost*

Peneliti dapat mendefinisikan model *XGBoost* dengan fungsi *XGBoost* dengan mengubah beberapa parameter. Pada model *XGBoost* digunakan fungsi *training* jadi perlu memasukan dataset *train*, kemudian menyesuaikan serangkaian model *XGBoost* untuk setiap `cv.nvold` Pada *k-folds cross validation* nilai `cv.nfold` dipilih secara acak, jumlah `cv.nfold` biasanya akan menjadi 5 atau 10, tetapi itu tergantung pada ukuran data, rasio *noise* dan waktu *training*. Parameter catatan lainnya adalah *nround* dan *prediksi*. Pada penelitian ini menggunakan `cv.nfold` = 5. Parameter *nround* memberitahu *XGBoost* berapa kali untuk mengulangi, pada penelitian ini menggunakan *nround* = 50. Ini secara inheren terkait dengan *learning rate eta* dan memunginkan membutuhkan *tunning hyperparameter* karena sangat penting untuk memperbaikinya.

Tabel 5.6 Parameter Model *XGBoost*

Parameter *XGBoost*	
Objective	Multi:softprob
Eval_metric	Mlogloss
Num_class	4
Method	xgbTree

Berdasarkan Table 5.6 hasil mengatur pemasangan parameter dan kemudian menyesuaikan serangkaian model *XGBoost.* Dimulai dengan menetapkan jumlah kelas, sekelompok parameter untuk kesesuaian *XGBoost* dan keluar parameter nya. Daftar *xgb_params* menampung beberapa informasi penting untuk prediksi multi-kelas. Disini peneliti menetapkan tujuan ke `multi:softprob` dan `eval_metric` =

`mlogloss`. Dua parameter ini memberitahu algoritma *XGBoost* bahwa ingin mengklasifikasi probabilitas dan menggunakan *multiclass logloss* sebagai evaluasi metrik. Penggunaan `multi:softprob` juga mengharuskan kita tahu adala jumlah kelas yang kita miliki dengan `num_class`.

Untuk menilai kesalahan prediksi menggunakan data yang ada pada objek model *cross validation*, objek berisi nilai prediksi untuk setiap pengamatan dalam dataset *train* berada di lipatan K^{th} yang dikenal sebagai *Out Of Fold* (OOF) sampel. Sampel ini tidak digunakan untuk *train* model sehingga bertindak sebagai sample independent. Objek prediksi dari `xgb.cv()` berisi kolom untuk masing masing kelas dan probabilitas bahwa setiap pengamatan milik masing-masing kelas. Jika kesalahan yang dinilai dari *cross validation* dapat di terima dalam kisaran variasi yang masuk akal misalnya semua lipatan menunjukan tingkat kesalahan yang relatif konsisten, kemudian menyesuaikan dengan dataset *training*. Agar sesuai dengan model *training,* fungsi `xgb.train()` digunakan dengan parameter yang ditetapkan diatas. Mengikuti model fit, dataset *testing* diprediksi menggunakan bst_model. Hasilnya diubah menjadi matriks prediksi kelas, probabilitas maksimal diambil dan label sebenarnya ditambahkan semuanya dengan cara yang sama seperti yang dilakukan untuk data OOF.

5.3.2 Prediksi *XGBoost*

Hasil dari pengujian model dianalisis dengan melakukan pencocokan kelas predisksi dengan kelas sebenarnya yang dapat memberikan informasi berupa *accuracy, 95% CI, No Information Rate, Kappa, Sensitivity, Specificity* dan *Precision* dengan menggunakan co*nfusion matrix.*

Tabel 5.7 *Confussion Matrix* Sebelum *Optimasi Hyperparameter*

Prediksi	**Aktual**			
	Kelas 0	Kelas1	Kelas 2	Kelas 3
Kelas 0	1	0	0	0
Kelas1	0	4	0	1
Kelas 2	1	1	3	1
Kelas 3	0	0	0	1

Confusion matrix memberi informasi mengenai kinerja dan kemampuan model untuk menggeneralisasi, tetapi harus membutuhkan label kelas yang diprediksi bukan probabilitas. Oleh karena itu, perlu menetapkan nilai *threshold*. Nilai *threshold* adalah *cutoff* di mana menentukan prediksi positif dan prediksi negatif. Tabel 5.7 dan tabel 5.8 merupakan hasil *confusion matrix* model tanpa menggunakan *tunning hyperparameter.*

Tabel 5.8 *Overall statistic* Sebelum *Optimasi Hyperparameter*

Overall statistic	
Accuracy	0.6923 (69.23%)
95% CI	(0.3857, 0.9091)
No Information Rate	0.3846
P-Value [ACC > NIR]	0.0245
Kappa	0.5702

Dari Tabel 5.8 diketahui Kelas 0 = *fatalities*, Kelas 1 = *viremic suvivors*, Kelas 2 = *suvivors in recovery phase* dan Kelas 3 = *healthy contro*l kemudian menghitung nilai akurasi untuk *confusion matrix multiclass* diketahui bahwa set *testing* memberikan akurasi keseluruhan 69.23% dengan 95% CI dari 38.57% hingga 90.91% serta nilai Kappa 0.5702.

5.3.3 *Optimasi Hyperparameter*

Pada pembahasan model dan prediksi diatas, hanya menggunakan parameter *cross validation* tanpa menggunakan parameter *tunning hyperparameter* untuk meningkatkan model, sehingga pada pembahasan ini peneliti melakukan proses peningkatan model menggunakan metode *tunning hyperparameter* secara *grid search*. Pada Tabel 5.9 merupakan uraian untuk mendapatkan nilai *Hyperparameter* yang digunakan pada tunning secara *grid search* yang akan dikonfigurasi untuk *XGBoost*. *Hyperparameter* lainnya menggunakan nilai *default*. Keseluruhan kombinasi nilai *hyperparameter* tersebut dijalankan hingga didapatkan nilai terbaik sebagai model akhir.

Tabel 5.9 Kombinasi Nilai Parameter *Tunning Hyperparameter*

Keterangan	*Accuracy*
nrounds	500
eta	(0.001, 0.005, 0.01, 0.05, 0.1, 0.5)
max_depth	(2, 4, 6, 8)
gamma	(1, 2, 4)
subsample	(0.25, 0.5, 0.75)
min_child_weight	(1, 2, 3)
colsample_bytree	1

Setelah melakukan tunning hyperparameter dengan menggunakan fungsi `expand.grid` untuk membuat data frame dari semua kombinasi vektor atau faktor. Setelah itu menggunakan fungsi `train` untuk memperkirakan parameter untuk model yang diberikan dari sekumpulan data, maka ringkasan hasil tentang nilai parameter terbaik untuk setiap parameter disajikan pada table 5.10 dibawah.

Tabel 5.10 Hasil Nilai Parameter *Tunning Hyperparameter*

Keterangan	*Accuracy*
nrounds	500
eta	0.005
max_depth	2
gamma	1
subsample	0.75
min_child_weight	1
colsample_bytree	1

Setelah didapatkan hasil nilai terbaik *tunning hyperparameter* secara *grid search*, kemudian hasil tersebut di inputkan kedalam parameter model *XGBoost* yang terbaik sehingga di dapatkan model dan prediksi yang maksimal. Pada tabel 5.11 merupakan hasil confusion matrix setelah menggunakan parameter terbaik.

Tabel 5.11 *Confusion Matrix* Setelah *Tunning Hyperparameter*

Prediksi	Aktual			
	Kelas 0	Kelas1	Kelas 2	Kelas 3
Kelas 0	1	0	0	1
Kelas1	0	5	0	1
Kelas 2	1	0	3	0
Kelas 3	0	0	0	1

Berdasarkan Tabel 5.11 di atas, data yang digunakan pada proses uji data sebesar 13 sampel. Untuk menghitung nilai akurasi untuk *confusion matrix multiclass* dengan menggunakan persamaan berikut.

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}\ x\ 100\%$$

$$= \frac{1 + 5 + 3 + 1}{13}\ x\ 100\%$$

$$= 76\%$$

Dari perhitungan nilai *accuracy*, dapat diketahui bahwa tingkat kedekatan antara nilai prediksi dan nilai aktual adalah sebesar 76%, yang mempunyai arti bahwa sistem dapat melakukan klasifikasi antara kelas prediksi dan kelas aktual dengan baik dimana dari 13 sampel terdapat 10 sampel yang tepat pengklasifikasiannya.

$$\textit{Precision} = \frac{TP}{TP + FP}\ x\ 100\%$$

$$\textit{P(A)} = \frac{1}{1 + 1} = 0.5$$

$$\textit{P(B)} = \frac{5}{5 + 0} = 1$$

$$\textit{P(C)} = \frac{3}{3 + 0} = 1$$

$$\textit{P(D)} = \frac{1}{1 + 2} = 0.33$$

$$\textit{All Precision} = \frac{0.5 + 1 + 1 + 0.33}{4}\ x\ 100\%$$

= 70%

Dari perhitungan nilai *precision*, dapat diketahui nilai presisi untuk pengklasifikasian mempunyai hasil prediksi yang tepat dengan nilai presisi sebesar 70%.

Recall / Sensitivity = $\frac{TP}{TP + FN}\ x\ 100\%$

R(A) = $\frac{1}{1 + 1} = 0.5$

R(B) = $\frac{5}{5 + 1} = 0.83$

R(C) = $\frac{3}{3 + 1} = 0.75$

R(D) = $\frac{1}{1 + 0} = 1$

All Recall = $\frac{0.5 + 0.83 + 0.75 + 1}{4}\ x\ 100\%$

= 77%

Dari perhitungan nilai *sensitivity* yang diperoleh nilainya sebesar 77% yang berarti bahwa proporsi kelas positif yang diklasifikasikan dengan benar oleh sistem masih terdapat kesalahan.

Specificity = $\frac{TN}{TN + FP}\ x\ 100\%$

S(A) = $\frac{10}{10 + 1} = 0.90$

S(B) = $\frac{7}{7 + 0} = 1$

S(C) = $\frac{9}{9 + 0} = 1$

S(D) = $\frac{10}{10 + 2} = 0.83$

All Specificity = $\frac{0.90 + 1 + 1 + 0.83}{4}\ x\ 100\%$

= 93%

FPR = $1 - specificity$

$$= 1 - 0{,}93$$

$$= 0{,}07$$

Dari perhitungan *specificity* yang diperoleh nilainya sebesar 93% yang berarti bahwa proporsi kelas negatif yang diklasifikasikan dengan benar oleh sisem masih belum sempurna karena terdapat kesalah pada klasifikasi, dengan nilai FPR yang dihitung bernilai 0.07.

$$AUC = \frac{1 + sensitivitas - FPR}{2}$$

$$= \frac{1 + 0{,}77 - 0.07}{2}$$

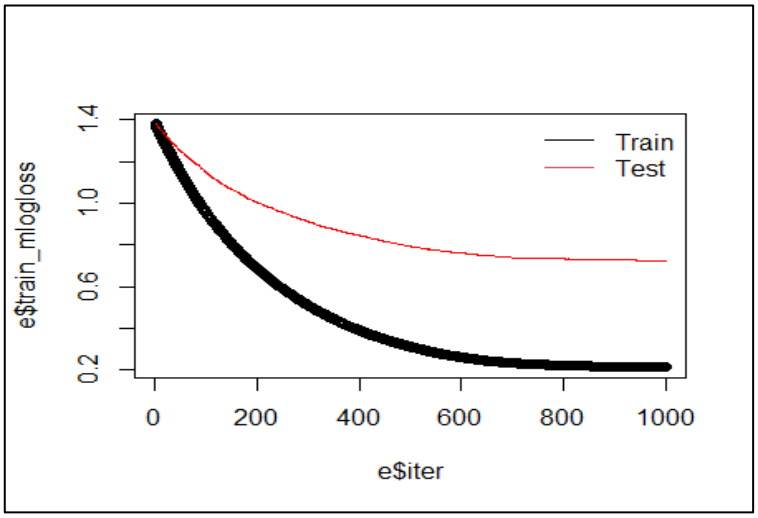
$$= 0{,}85$$

Dari analisis, diperoleh akurasi untuk model dengan menggunakan data uji sebesar 76.92%. Kemudian ditinjau dari nilai *AUC* untuk melihat nilai kinerja pengklasifikasian secara umum dimana niali *AUC* berada pada rentang nilai 0 hingga 1, dimana jika nilai *AUC* mendekati 1 dapat dikatakan bahwa hasil prediksi semakin akurat. Dari perhitungan nilai *AUC*, didapatkan nilai sebesar 0,85 yang berarti bahwa klasifikasi yang diperoleh cukup.

Tabel 5.12 *Overall Statistic* Setelah *Tunning Hyperparameter*

Overall statistic	
Accuracy	0.7692 (76.92%)
95% CI	(0.4619, 0.9496)
No Information Rate	0.3846
P-Value [ACC > NIR]	0.005614
Kappa	0.675

Dari Tabel 5.12 diketahui bahwa set *testing* memberikan akurasi keseluruhan 76.92% dengan 95% CI dari 46% hingga 94% serta nilai Kappa 0.675.



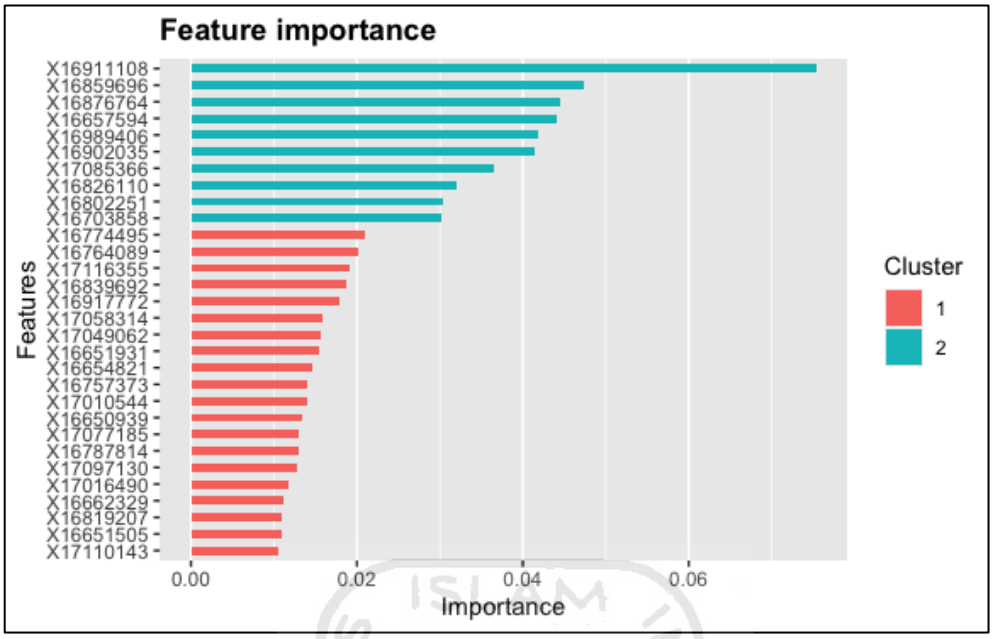
Gambar 5.8 *Logarithmic loss Training* dan *Testing*

Pada gambar 5.8 merupakan plot *logarithmic loss* dari model *XGBoost* untuk setiap periode *training* dan *testing* dataset, garis hitam merupakan garis *train logloss* sementara garis yang merah yaitu *test logloss*. *XGBoost* mensyaratkan penggunaan alat pengukuran berupa *logloss* untuk melakukan klasifikasi lebih dari 2 kategori. *Logloss* merupakan metode pengukuran performa model klasifikasi di mana probabilitas prediksi bernilai antara 0 dan 1. Dari plot tersebut menunjukkan iterasi ke-500 nilai logloss cenderung stabil sehingga penentuan nround dapat dimulai dari iterasi 500.

Tabel 5.13 *Variabel Importance*

Feature	*Gain*	*Cover*	*Frequency*
X16911108	0.0753242574	0.0863280132	0.0814527504
X16859696	0.0474133267	0.0472060028	0.0363187588
X16876764	0.0444142484	0.0433621134	0.0430183357
X16657594	0.0439936794	0.0387197454	0.0377291961
X16989406	0.0418547134	0.0394002164	0.0468970381

Pada table 5.13 merupakan nilai *variabel Importance* yang berisi nilai *Gain, Cover* dan *Frequency* dari masing-masing *Feature* dari nilai tersebut akan digunakan untuk membuat plot `xgb.importance()` yang dapat dilihat pada gambar 5.8.



Gambar 5.9 Histogram *Feature importance*

Pada tahapan *feature importance* menggunakan semua splits di pohon *XGBoost* untuk menilai tingkat kepentingan variabel dan memahami seberapa akurat klasifikasi berdasarkan pada splits. Diperoleh dari fungsi `xgb.importance()` mengambil informasi Gain dan memplotnya menggunakan *ggplot2*. Variabel juga dikelompokkan menggunakan *k-means clustering* yang mengoptimalkan k berdasarkan *Bayesian Information Criteria* (BIC) untuk k=1 hingga k=30. Dari gambar 5.8 dapat melihat bahwa elemen X16911108, X16859696, X16876764, X16657594, X16989406, X16902035, X17085366, X16826110, X16802251 dan X16703858 ada pada cluster ke 2 tampaknya yang paling penting dalam mengklasifikasi setiap artefak ke akurasi yang ditunjukan oleh *best_model* mengingat data dan *hyperparameter* sementara untuk elemen X16774495, X16764089 sampai X17110143 yang terlihat pada gambar berada di cluster 1.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian yang diperoleh, maka peneliti dapat memberikan beberapa kesimpulan yang dihasilkan sebagai berikut:

1. Langkah pengolahan dan membaca data bionformatik pada penelitian memerlukan bantuan *package* `oligo` untuk membaca data file *Affymetrix CEL*. Pada proses pengolahan selanjutnya dengan 2 tahap, yaitu tahap *preprocessing* dan tahap *filtering*. Pada tahap *preprocessing* terdapat tiga hal yang harus dilakukan yaitu *background correction*, *normalization* dan *summarization*. Sedangkan untuk filtering terdapat dua tahap, yaitu *non specific filter* dan *feature selection*. Setelah melewati semua proses diatas, data tereduksi dari jumlah awal 2598544 gen menjadi 184 gen.
2. Gambaran data eksperimen chip RNA mengenai imun selama penyakit virus Ebola dilakukan pada 59 sampel pasien dan dibagi menjadi 4 kelas yaitu jaringan *fatalities, viremic suvivors, suvivors in recovery phase* dan *healthy contro*l. Sebanyak 23 sampel mempunyai jaringan *fatalities* pada pasien virus Ebola (39%), 14 sampel mempunyai jaringan *viremic suvivors* pada pasien virus Ebola (24%), 14 sampel mempunyai jaringan *suvivors in recovery phase* pada pasien virus Ebola (24%) dan 8 sampel mempunyai jaringan *healthy contro*l pada pasien virus Ebola (14%).
3. Pengolahan data ekspresi gen jaringan *fatalities*, *viremic suvivors*, *suvivors in recovery phase* dan *healthy contro*l dengan menggunakan metode klasifikai *eXtreme Gradient Boosting* memberikan akurasi keseluruhan 69.23% dengan 95% CI dari 38.57% hingga 90.91% serta nilai Kappa 0.5702.
4. Didapatkan hasil nilai terbaik *tunning hyperparameter* secara *grid search* yaitu nround = 500, eta = 0.005, max_depth = 2, gamma = 1, sub_sampel = 0.75, min_child_weight = 1, colsample_bytree = 1, memberikan akurasi

keseluruhan 76.92% dengan 95% CI dari 46% hingga 94% serta nilai Kappa 0.675. Dari perhitungan menggunakan persamaan *confusion matrix* didapatkan nilai *accuracy*, *precision*, *sensitifity*, *specificity* dan *AUC* adalah sebesar 76.92%, 70%, 77%, 93% dan 85% yang berarti bahwa klasifikasi yang diperoleh cukup.

6.2 Saran

Berdasarkan hasil analisis dan kesimpulan yang diperoleh, peneliti dapat memberikan saran-saran sebagai berikut:

1. Penelitian selanjutnya dapat melakukan perbandingan metode untuk pengklasifikasian data ekspresi gen Parameter Dan Hasil Kekebalan Selama Penyakit Virus Ebola untuk mencari metode klasifikasi terbaik, misalnya membandingkan antara metode Bagging dengan Boosting.
2. Pada penelitian selanjutnya dapat menerapkan metode tunning hyperparameter yang lainnya, untuk mencapai nilai tunning yang terbaik, salah satunya dengan menggunakan metode Random search.

DAFTAR PUSTAKA

Abu, A. (2017). Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *Jurnal Teknologi Indonesia*.

Ahmad, A. (2017). Mengenal Artificial Intelligence, Machine learning, Neural Network, dan Deep Learning. *Yayasan Cahaya Islam Jurnal Teknologi Indonesia*.

Aprijani, D., & Elfaizi, M. (2004). BIOINFORMATIKA: Perkembangan, Disiplin Ilmu dan Penerapannya di Indonesia. *diakses dari ftp://202.125*, 94.

Baize, S., Pannetier , D., Oestereich, L., Rieger, T., Koivogui, L., Magassouba, N., & et al. (2014). Emergence of Zaire ebola virus disease in Guinea – Preliminary report. *N Eng J Med*, 1-8.

Bausch , D., Towner, J., Dowell , S., Kaducu, F., Lukwiya, M., & Sanchez, A. (2007). Assessment of the risk of Ebola virus transmission from bodily fluids and fomites. *J Infect Dis*, 196:S142-7.

Bayat, A. (2002). Bioinfromatics: Science, medicine, and the future. *BMJ 324*, 1018-1022.

Bolstad, B. M. (2004). Low level Analysis of High density Oligonucleotide Array Data : Background, Normalization and Summarization. *University Of California*.

Bourgon, R., Gentleman, R., & Huber, W. (2010). Independent filtering increases detection power for high-throughput experiments. *PNAS*, 107 (21) 9546-9551.

Carvalho, B., & Irizarry , R. (2010). A Framework for Oligonucleotide Microarray Preprocessing. *Bioinformatics*, 2363-7.

Chen , T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Knowledge discovery and data mining*, ACM: 785-794.

Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Gautier, L., Cope, L., Bolstad, B., & Irizarry, R. (2004). affy—analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, Volume 20, Issue 3 ,Pages 307–315.

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

Han, J., Kamber, M., & Pei, J. (2012). *Third Editiom Data Mining Concepts and Techniques.* USA: Morgn Kaufmann.

Handayani, A., Jamal, A., & Septiandri, A. A. (2017). Evaluasi Tiga Jenis Algoritme Berbasis Pembelajaran Mesin untuk Klasifikasi Jenis Tumor Payudara. *JNTETI*, Vol. 6, No. 4.

Hanif, I. (2019). Implementing Extreme Gradient Boosting (XGBoost) Classifier to Improve Customer Churn Prediction. *Proceedings of the 1st International Conference on Statistics and Analytics.* Bogor, Indonesia: EAI.

Hovatta, I., Kimppa, K., Lehmussola, A., Pasanen, T., Saarela, J., Saarikko, I., . . . Wong, G. (2005). DNA Microarray Data Analysis. *Scientific Computing*.

Karabulut, E. M., Ozel, S. A., & Ibrikci, T. (2011). A comparative study on the effect of feature selection on classification accuracy. *Procedia Technology*, 323-327.

Kumar, A. (2010). *Gene : Expression and Regulation (in Recent Advances in Life Sciences).* New Delhi, India: IK international publisher.

Li , Y., & Chen , S. (2013). Evolutionary history of Ebola virus. *Epidemiol Infect*, 16:1-8.

Mayumbe-Tamfum, J., Mulangu, S., Masumu, J., Kayambe, J., Kemp, A., & Paweska, J. (2012). bola virus outbreaks in Africa: Past and present. *Onderstepoort Journal of Veterinary Research*, 79(2).

Pomares, H., Calvo, J., & Gonzalo, C. (2009). On Selecting the Best Pre-processing Method for Affymetix Genechips. *ResearchGate.*

Prasetyo, S., Christianto, Y., & Hartomo, K. (2019). *Analisis Data Citra Landsat 8 OLI Sebagai Indeks.* Salatiga: Indonesian Jurnal of Computing and Modeling.

Purwoto, A. (2007). *Panduan Laboratorium Statistik Inferensial.* Jakarta: Gramedia Widiasarana Indonesia.

Putatunda, S., & Rama, K. (2018). A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost. *SPML '18: Proceedings of the 2018 International Conference on Signal Processing and Machine Learning* (pp. Pages 6-10). New York, United States: Association for Computing Machinery.

Quinlan, J. (1996). Bagging, Boosting and C4.5. *In AAAI*, 725-730.

Ramadhan, W. (2020, Maret 20). *Pengantar Komputasi Modern* . Retrieved from Komputasi pada bidang biologi :

https://adhaniscuber.files.wordpress.com/2017/04/pengantarkomputasimodern-komputasibiologi.pdf

Ranganathan, S., Nakai, K., & Schonbac, C. (2018). *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics.* Elsevier.

Reynard, S., Journeaux, A., Gloaguen, E., Schaeffer, J., & et al. (2019). Immune parameters and outcomes during Ebola virus disease. *JCI Insight*, 10;4(1).

Rocca, J. (2019). *Ensemble methods: bagging, boosting and stacking*. Retrieved from Towards Data Science: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

Salamah , U., & Ramayanti, D. (2018). Supervised Classification of Indonesian Text Document Using Extreme Gradient Boosting (XGBoost). *International Journal of Computer Techniques*, Volume 5 Issue 5.

Schaer, R., Muller, H., & Depeursinge, A. (2016). Optimized Distributed Hyperparameter Search and Simulation for Lung Texture Classification in CT Using Hadoop. *J.Imaging*, vol. 2, no. 19, pp. 1–20.

Schapire , R. (1990). A Brief Introduction to Boosting. *IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence, 1999* (pp. Pages 1401–1406). 180 Park Avenue, Room A279, Florham Park, NJ 07932, USA: AT&T Labs, Shannon Laboratory.

Sofi, D., & Jajuli. (2015). Integrasi Metode Klasifikasi dan Clustering dalam Data Mining. *Jurnal Teknik Informatika Fakultas Ilmu Komputer*, Universitas Singaperbangsa Karawang.

Sullivan, N., Yang, Y., & Nabel, G. (2003). Ebola virus pathogenesis: implications for vaccines and therapies. *J Virol*, 77(18):9733-7.

Sun , M.-a., Shao, X.-j., & Wang, Y. (2018). Microarray Data Analysis for Transcriptome Profiling. *Methods in molecular biology (Clifton, N.J.)*, 1751:17-33.

Suyanto. (2018). *Machine Learning Tingkat Dasar dan Lanjut* . Bandung: Informatika Bandung.

Syahrani, I. (2019). *Analisis Perbandingan Teknik Ensemble Secara Boosting (XGBOOST) Dan Bagging (RANDOM FOREST) Pada Klasifikasi Kategori Sambatan Sekuens DNA.* Sekolah Pasca Sarjana Institut Pertanian Bogor.

Tanaka, M., & Okutomi, M. (2014). *PatternRecognition (ICPR)* (pp. pp 1526–1531). 2014 22nd International Conference on.

Thalib, A. K. (2018). *Analisis klasifikasi Hoax pada Unstructured Data Text di Situs Portal Berita Detik.com dan Turnbackhoax.id.* Yogyakarta: Skripsi Statistika FMIPA UII.

Torlay, L., Perrone-Bertolotti, M., Thomas, E., & Baciu, M. (2017). Machine learning–XGBoost analysis of language networks to classify patients with epilepsy. *Brain Inform*, 4(3): 159–169.

Utama, A. (2003). Peranan Bioinformatika Dalam Dunia Kedokteran . *IlmuKomputer.Com*.

Vernet, M., & et al. (2017). Clinical, virological, and biological parameters associated with outcomes of Ebola virus infection in Macenta, Guinea. *JCI Insight*, 2(6):e88864.

Vicente, R. S. (2009). Visual Analysis Of Gene Expression Data By Means Of Biclustering. *University Of Salamanca*.

Yang, P., Yang, Y. H., Zhou, B. B., & Zomaya, A. Y. (2016). A Review of ensemble methods in Bioinformatics. 1.

Zhang, T., & Johnson, R. (2014). Learning nonlinear functions using regularized greedy forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5).

Zhou , Z. (2012). *Ensemble methods: foundations and algorithms*. Retrieved from Chapman and Hall/CRC: http://www2.islab.ntua.gr/attachments/article/86/Ensemble%20methods%20-%20Zhou.pdf

LAMPIRAN

Lampiran 1 Syntaks Membaca data *microarray*

```
### Install package Biocmanager ###
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(version = "3.10")
##Package
BiocManager::install(c("affy", "GEOquery", "Biobase",
"simpleaffy", "affyPLM", "u133x3pcdf", "u133x3p.db",
"genefilter", "AnnotationDbi"))
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
library(affy)
library(GEOquery)
library(Biobase)
library(simpleaffy)
library(affyPLM)
library(u133x3pcdf)
library(u133x3p.db)
library(genefilter)
library(AnnotationDbi)
library(oligo)

#listing the files from directory using special CEL file
read function
gse<-
list.celfiles("/Users/labstatistika8/Documents/GSE122692_RAW
", full.names=T)
gse
#reading data from cellist and setting annotation package to
approiate one for this microarray
rawData <- read.celfiles(gse, pkgname='pd.hugene.2.0.st')
library(oligoData)
rawData
class(rawData)
#boxplot
boxplot(rawData)
#pseudo-images of the samples
image(rawData, 1, col=gray((64:0)/64))
#histogram
hist(rawData)
#MA plot
MAplot(rawData, which=1, ylim=c(-1, 1))
```

Lampiran 2 Melihat Phenodata

```
#Get pheno data
gset <- getGEO(GEO="GSE122692",GSEMatrix =TRUE)
gset
data.gse <- exprs(gset[[1]])
data.gse
class(data.gse)
dim(data.gse)
data.gse[1:5,1-4]
pheno <- pData(phenoData(gset[[1]]))
pheno
varLabels(phenoData(gset[[1]]))
dim(pheno)
str(pheno)

#pie chart
des <-table(pheno$'group:ch1')
des
persen <- round(des/sum(des)*100)
des <- as.data.frame(des)
lbls <- paste(des$Var1,'-',persen, '%', sep='')
pie(des$Freq, label= lbls,
col=c('red','blue','green','yellow'))
```

Lampiran 3 Syntaks *Preprocessing*

```
#normalizing the data using RMA algorithm
library(Biobase)
normData <- rma(rawData, target="core")
#retreaving feature data
featureData(normData) <- getNetAffx(normData, "transcript")
class(normData)
show(normData)
#boxplot and histogram after RMA
boxplot(normData, transfo=identity, main="After RMA")
hist(normData, transfo=identity, main="After RMA")
#Gene Annotation
library(hugene20sttranscriptcluster.db)
keytypes(hugene20sttranscriptcluster.db)
gns <- select(hugene20sttranscriptcluster.db,
keys(hugene20sttranscriptcluster.db), c("ENTREZID",
"SYMBOL"))
gns <- gns[!duplicated(gns[,1]),]
gns = gns[,-1]
row.names(gns) = keys(hugene20sttranscriptcluster.db)
expr <- data.frame(exprs(normData))
expr.anno <- merge(gns, expr, by.x=0, by.y=0, all=TRUE)
write.table(expr.anno, file = "rma_norm_expr.anno.txt", sep
= "\t", row.names = FALSE, col.names = TRUE, quote = FALSE)
```

Lampiran 4 Syntaks *Filtering*

```
#Gene Filtering
library(genefilter)
eset.filt = varFilter(normData, var.func=IQR, var.cutoff =
0.25, filterByQuantile = TRUE)
nrow(eset.filt)
eset.filt = eset.filt[featureNames(eset.filt) %in%
row.names(gns)[!is.na(gns$ENTREZID)], ] nrow(eset.filt)
eset.filt = eset.filt[featureNames(eset.filt) %in%
row.names(gns)[!duplicated(gns$SYMBOL)],] nrow(eset.filt)
nrow(eset.filt)
log <-eset.filt$filter.log
eset <- eset.filt$index
#Melihat hasil filter (eset)
dim(eset)
head(eset)
class(eset)
eset
#expression set: Membuat data microarray menjadi matrix/data
frame
databaru <- exprs(eset.filt)
dim(databaru)
class(databaru)
head(databaru)
View(databaru)
```

Lampiran 5 Syntaks Membuat label klasifikasi

```
### Differential Expression Analysis ###
#merubah kelas menjadi numerik
datacl <- c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,
            1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,
            2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,
            3,3,3,3,3,3,3,3)
length(datacl)
class(datacl)
datafac <- factor(datacl,levels=0:3, labels=
c("HC","FA","SR","SV"))
datafac
```

Lampiran 6 Syntaks *Feature Selection*

```
#filtering setelah mengubah kelas menjadi numerik
BiocManager::install("multtest")
library(multtest)
?mt.teststat
```

```
datattest <- mt.teststat(databaru,datacl,test="f")
class(datattest)
length(datattest)
qqnorm(datattest)
qqline(datattest)
length(datattest)

#Adjusting p-value (untuk melihat p-value yg sesuai dan
tidak sesuai)
#filtering dg metode penyesuaian
rawp = 2 * (1 - pnorm(abs(datattest)))
length(rawp)
prosedur = c("Bonferroni", "Holm", "Hochberg", "BH", "BY")
adjusted = mt.rawp2adjp(rawp, prosedur)
data <- adjusted$adjp[,]
data1 <- data[order(adjusted$index), ]
head(data1)
dim(data1)
#mengambil data p-value adjp (p-value adjusted)/mengambil
kolom rawp
ffs <- data1[,1]
class(ffs)
length(ffs)
ffs[1 : 10]
View(ffs)
#Adjusting rawp
#bainding data hasil filtering dengan p-value
adjusted/adjusting rawp
datarawp <- data.frame(databaru, ffs)
row.names(datarawp) <- row.names(databaru)
class(datarawp)
head(datarawp)
dim(datarawp)
library(dplyr)
#datadatarawpfilter <- filter(datarawp, ffs < 0.000001)
datarawpfilterfinal <- subset(datarawp, ffs <
0.000000000001)
#dim(datadatarawpfilter)
rownames(datarawpfilterfinal)
class(datarawpfilterfinal)
dim(datarawpfilterfinal)
head(datarawpfilterfinal)
## mendefinisikan data baru setelah filter
datadef <- datarawpfilterfinal[,1:59]
head(datadef)
dim(datadef)
summary(datadef)
colnames(datadef)
#Usepackages
install.packages("pROC")
library(pROC)
```

```
data_siap   = as.data.frame (t(datadef))
head(data_siap)
dim(data_siap)
dataY = as.factor(datacl)
dataY
data_use = as.data.frame(cbind(data_siap,dataY))
data_use
dim(data_use)
head(data_use)
```

Lampiran 7 Syntaks Membuat Data *Training* dan *Testing*

```
#### Training dan Testing #####

HC<-dplyr::filter(data_use, data_use$dataY==0)
dim(HC)
FA<-dplyr::filter(data_use, data_use$dataY==1)
dim(FA)
SR<-dplyr::filter(data_use, data_use$dataY==2)
dim(SR)
SV<-dplyr::filter(data_use, data_use$dataY==3)
dim(SV)

#penentuan ukuran data traning dan testing
set.seed(1000)
rasio=8/10
n<-round(nrow(HC)*0.8)
sampel_HC<-sample(1:nrow(HC),n)
m<-round(nrow(FA)*0.8)
sampel_FA<-sample(1:nrow(FA),m)
x<-round(nrow(SR)*0.8)
sampel_SR<-sample(1:nrow(SR),x)
o<-round(nrow(SV)*0.8)
sampel_SV<-sample(1:nrow(SV),o)

#data training dan testing
#training
training_HC=HC[sampel_HC,]
dim(training_HC)
training_FA=FA[sampel_FA,]
dim(training_FA)
training_SR=SR[sampel_SR,]
dim(training_SR)
training_SV=SV[sampel_SV,]
dim(training_SV)
#testing
testing_HC=HC[-sampel_HC,]
dim(testing_HC)
testing_FA=FA[-sampel_FA,]
```

```
dim(testing_FA)
testing_SR=SR[-sampel_SR,]
dim(testing_SR)
testing_SV=SV[-sampel_SV,]
dim(testing_SV)

data_training=rbind(training_HC,training_FA,training_SR,trai
ning_SV)
data_testting=rbind(testing_HC,testing_FA,testing_SR,testing
_SV)
dim(data_training)
dim(data_testting)
View(data_training)

write.csv(data_training, file
="/Users/labstatistika8/Documents/train1.csv" )
write.csv(data_testting, file
="/Users/labstatistika8/Documents/test1.csv" )
?write.csv
```

Lampiran 8 Syntaks Analisis *XGBoost Cross Validation*

```
install.packages("xgboost")
library(xgboost)
library(keras)
library(tensorflow)
library(caret)
library(dplyr)
data_training = read.csv(file.choose(), header = T)
dim(data_training)
data_testting=read.csv(file.choose(), header = T)
dim(data_testting)
## coba
library(Matrix)
dim(data_training)
dim(data_testting)
data_train<-data_training[,2:186]
dim(data_train)
data_test<-data_testting[,2:186]
dim(data_test)

trainm <- sparse.model.matrix(data_train$dataY~.,
data=data_train)
head(trainm)
train_label <-data_train[,"dataY"]
train_matrix <-xgb.DMatrix(data =
as.matrix(trainm),label=train_label)
```

```
testm<-sparse.model.matrix(data_test$dataY~., data =
data_test)
test_label<-data_test[,"dataY"]
test_matrix <-xgb.DMatrix(data = as.matrix(testm),
label=test_label)

#Parameter  ie no of class
numberOfClasses <- length(unique(train_label))
xgb_params <- list("objective" = "multi:softprob",
                   "eval_metric" = "mlogloss",
                   "num_class" = numberOfClasses)
nround    <- 50 # number of XGBoost rounds
cv.nfold  <- 5

# Fit cv.nfold * cv.nround XGB models and save OOF
predictions
cv_model <- xgb.cv(params = xgb_params,
                   data = train_matrix,
                   nrounds = nround,
                   nfold = cv.nfold,
                   verbose = FALSE,
                   prediction = TRUE)
OOF_prediction <- data.frame(cv_model$pred) %>%
  mutate(max_prob = max.col(., ties.method = "last"),
         label = train_label + 1)
head(OOF_prediction)
confusionMatrix(factor(OOF_prediction$max_prob),
                factor(OOF_prediction$label),
                mode = "everything")
bst_model <- xgb.train(params = xgb_params,
                       data = train_matrix,
                       nrounds = nround)
# Predict hold-out test set
test_pred <- predict(bst_model, newdata = test_matrix)
test_prediction <- matrix(test_pred, nrow = numberOfClasses,

ncol=length(test_pred)/numberOfClasses) %>%
  t() %>%
  data.frame() %>%
  mutate(label = test_label + 1,
         max_prob = max.col(., "last"))
# confusion matrix of test set
confusionMatrix(factor(test_prediction$max_prob),
                factor(test_prediction$label),
                mode = "everything")
```

Lampiran 9 Syntaks Analisis *XGBoost* dengan *Tunning Hyperparameter*

```
#optimasi hyperparameter grid search
xgb_grid_par = expand.grid(
  nrounds = 500,
  eta = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5),
  max_depth = c(2,4,6,8),
  gamma = c(1,2,4),
  subsample = c(0.25, 0.5, 0.75),
  min_child_weight = c(1, 2, 3),
  colsample_bytree = 1)
xgb_control<- trainControl(number=5, verboseIter=TRUE)
library(data.table)
tr<-data.table(data_train, keep.rownames = F)
modFitxgb <- train(form=factor(train_label)~.,
                   data = tr,
                   method = "xgbTree",
                   metric = "Accuracy",
                   trControl = xgb_control,
                   tuneGrid = xgb_grid_par)
print(modFitxgb)
#Parameter  ie no of class
nc <- length(unique(train_label))
xgb_params <- list("objective"="multi:softprob",
                   "eval_metric"="mlogloss",
                   "num_class"=nc, method="xgbTree")
watchlist <- list(train=train_matrix,test=test_matrix)
print(watchlist)

#XGB Model
set.seed(0)
gridsearch_model <- xgb.train(params = xgb_params,
                        data = train_matrix,
                        nrounds = 500,
                        watchlist = watchlist,
                        eta=0.005,max_depth=2,
                        gamma=1,colsample_bytree=1,
                        min_child_weight=1, subsample=0.75)
print(gridsearch_model)
#training & test error plot
e<-data.frame(gridsearch_model$evaluation_log)
plot(e$iter, e$train_mlogloss, col='blue')
lines(e$iter, e$test_mlogloss, col='red')
min(e$test_mlogloss)
e[e$test_mlogloss == 0.834809,]

#feature importance
imp<-xgb.importance(colnames(train_matrix),
                    model = gridsearch_model)
print(imp)
xgb.plot.importance(imp)
```

```
# plot
install.packages("Ckmeans.1d.dp")
library(Ckmeans.1d.dp)
gp = xgb.ggplot.importance(imp)
print(gp)

#prediction
p<-predict(gridsearch_model, newdata = test_matrix)
predik<-matrix(p, nrow = nc, ncol = length(p)/nc) %>%
  t()%>%
  data.frame()%>%
  mutate(label=test_label, max_prob=max.col(., "last")-1)
head(predik)
table(Prediction= predik$max_prob, Actual=predik$label)
10/13
# confusion matrix of test set
confusionMatrix(factor(predik$max_prob),
                factor(predik$label),
                mode = "everything")
```

Lampiran 10 Output Hasil *Preprocessing*

```
> normData <- rma(rawData, target="core")
Background correcting
Normalizing
Calculating Expression
> featureData(normData) <- getNetAffx(normData, "transcript")
> class(normData)
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
> show(normData)
ExpressionSet (storageMode: lockedEnvironment)
assayData: 53617 features, 59 samples
  element names: exprs
protocolData
  rowNames: GSM3478363_53_HuGene-2_0-st.CEL.gz GSM3478364_54_HuGene-2_0-st.CEL.gz ...
    GSM3478421_32_HuGene-2_0-st.CEL.gz (59 total)
  varLabels: exprs dates
  varMetadata: labelDescription channel
phenoData
  rowNames: GSM3478363_53_HuGene-2_0-st.CEL.gz GSM3478364_54_HuGene-2_0-st.CEL.gz ...
    GSM3478421_32_HuGene
  varLabels: index
  varMetadata: labelDescription channel
featureData
  featureNames: 16650001 16650003 ... 17127721 (53617 total)
  fvarLabels: transcriptclusterid probesetid ... category (18 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: pd.hugene.2.0.st
```

varFilter(eset, var.func = IQR, var.cutoff = 0.5, filterByQuantile = TRUE)

Lampiran 11 Output Hasil *Filtering*

```
> head(eset)
[1] 1 2 3 4 5 6
> class(eset)
[1] "integer"
> eset
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
[36] 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
> databaru <- exprs(eset.filt)
> dim(databaru)
[1] 40212    59
> class(databaru)
[1] "matrix"
> head(databaru)
        GSM3478363_53_HuGene-2_0-st.CEL.gz GSM3478364_54_HuGene-2_0-st.CEL.gz
> dim(eset)
NULL
> head(eset)
[1] 1 2 3 4 5 6
> class(eset)
[1] "integer"
> eset
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
[36] 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
> databaru <- exprs(eset.filt)
> dim(databaru)
[1] 40212    59
> class(databaru)
[1] "matrix"
```

Lampiran 12 Output Hasil *Feature Selection*

```
> rawp = 2 * (1 - pnorm(abs(datattest)))
> length(rawp)
[1] 40212
> prosedur = c("Bonferroni", "Holm", "Hochberg", "BH", "BY")
> adjusted = mt.rawp2adjp(rawp, prosedur)
> data <- adjusted$adjp[,]
> data1 <- data[order(adjusted$index), ]
> head(data1)
           rawp Bonferroni Holm  Hochberg        BH BY
[1,] 0.52834431          1    1 0.9997115 0.7542451  1
[2,] 0.06426832          1    1 0.9997115 0.2563225  1
[3,] 0.70735037          1    1 0.9997115 0.8439789  1
[4,] 0.17603829          1    1 0.9997115 0.4522939  1
[5,] 0.53598463          1    1 0.9997115 0.7588919  1
[6,] 0.38566410          1    1 0.9997115 0.6578014  1
> dim(data1)
[1] 40212     6
```

```
> library(dplyr)
> datarawpfilterfinal <- subset(datarawp, ffs < 0.000000000001)
> rownames(datarawpfilterfinal)
  [1] "16650511" "16650887" "16650939" "16651145" "16651505" "16651931" "16652595" "16654821" "16655323"
 [10] "16655367" "16657594" "16662329" "16666485" "16677544" "16679871" "16680031" "16686271" "16696723"
 [19] "16703858" "16711793" "16716443" "16716459" "16719492" "16725227" "16730308" "16731806" "16732891"
 [28] "16733409" "16736703" "16737434" "16741659" "16745628" "16745866" "16755046" "16757373" "16764089"
 [37] "16766860" "16774495" "16774766" "16783389" "16786470" "16787362" "16787814" "16791555" "16798138"
 [46] "16798148" "16801239" "16802251" "16802918" "16805190" "16805490" "16805492" "16807271" "16818359"
 [55] "16818701" "16819207" "16825425" "16826110" "16828732" "16837101" "16839692" "16841455" "16841689"
 [64] "16843113" "16852622" "16857736" "16859696" "16860778" "16865058" "16865066" "16865074" "16865100"
 [73] "16866374" "16872803" "16876764" "16880579" "16882332" "16882696" "16883715" "16885976" "16890055"
 [82] "16896442" "16902035" "16906335" "16911108" "16916483" "16917129" "16917552" "16917772" "16921064"
 [91] "16921706" "16924408" "16933774" "16936452" "16948569" "16954217" "16959505" "16960186" "16965773"
[100] "16966125" "16967321" "16972658" "16976360" "16981041" "16981219" "16984940" "16988913" "16989406"
[109] "16989408" "16995863" "16996345" "17005092" "17005276" "17010544" "17010618" "17010632" "17014208"
[118] "17014709" "17016254" "17016490" "17017173" "17019924" "17044046" "17046772" "17046836" "17049062"
[127] "17051524" "17053707" "17053850" "17058259" "17058314" "17073259" "17076573" "17077185" "17080060"
[136] "17083670" "17085366" "17086361" "17087395" "17087926" "17089117" "17094662" "17097130" "17097780"
[145] "17100197" "17100663" "17100671" "17100691" "17100695" "17100699" "17109394" "17110143" "17113062"
[154] "17114407" "17116355" "17117092" "17118496" "17118504" "17118898" "17119552" "17119578" "17119600"
[163] "17119654" "17119806" "17120122" "17120616" "17120908" "17120954" "17121922" "17122178" "17122186"
[172] "17122490" "17122700" "17123218" "17123256" "17123440" "17123740" "17123742" "17124060" "17124066"
[181] "17124068" "17124142" "17124624" "17125968"
> class(datarawpfilterfinal)
[1] "data.frame"
> dim(datarawpfilterfinal)
[1] 184  60
```

Lampiran 13 Output Hasil *Training* dan *Testing*

```
> dim(training_HC)
[1]   6 185
> training_FA=FA[sampel_FA,]
> dim(training_FA)
[1]  18 185
> training_SR=SR[sampel_SR,]
> dim(training_SR)
[1]  11 185
> training_SV=SV[sampel_SV,]
> dim(training_SV)
[1]  11 185
> testing_HC=HC[-sampel_HC,]
> dim(testing_HC)
[1]   2 185
> testing_FA=FA[-sampel_FA,]
> dim(testing_FA)
[1]   5 185
> testing_SR=SR[-sampel_SR,]
> dim(testing_SR)
[1]   3 185
> testing_SV=SV[-sampel_SV,]
> dim(testing_SV)
[1]   3 185
> data_training=rbind(training_HC,training_FA,training_SR,training_SV)
> data_testting=rbind(testing_HC,testing_FA,testing_SR,testing_SV)
> dim(data_training)
[1]  46 185
> dim(data_testting)
[1]  13 185
> View(data_training)
> write.csv(data_training, file ="/Users/labstatistika8/Documents/train1.csv" )
> write.csv(data_testting, file ="/Users/labstatistika8/Documents/test1.csv" )
```

Lampiran 14 *Output Confusion matrix* Sebelum *Tunning Hyperparameter*

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3 4
         1 1 0 0 0
         2 0 4 0 1
         3 1 1 3 1
         4 0 0 0 1

Overall Statistics

               Accuracy : 0.6923
                 95% CI : (0.3857, 0.9091)
    No Information Rate : 0.3846
    P-Value [Acc > NIR] : 0.0245

                  Kappa : 0.5702

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity           0.50000   0.8000   1.0000  0.33333
Specificity           1.00000   0.8750   0.7000  1.00000
Pos Pred Value        1.00000   0.8000   0.5000  1.00000
Neg Pred Value        0.91667   0.8750   1.0000  0.83333
Precision             1.00000   0.8000   0.5000  1.00000
Recall                0.50000   0.8000   1.0000  0.33333
F1                    0.66667   0.8000   0.6667  0.50000
Prevalence            0.15385   0.3846   0.2308  0.23077
Detection Rate        0.07692   0.3077   0.2308  0.07692
Detection Prevalence  0.07692   0.3846   0.4615  0.07692
Balanced Accuracy     0.75000   0.8375   0.8500  0.66667
```

Lampiran 15 *Output Confusion matrix* Sesudah *Tunning Hyperparameter*

```
Tuning parameter 'nrounds' was held constant at a value of 500
Tuning parameter 'colsample_bytree'
 was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 500, max_depth = 2, eta = 0.005, gamma =
 1, colsample_bytree = 1, min_child_weight = 1 and subsample = 0.75.
```

```
Confusion Matrix and Statistics

          Reference
Prediction 0 1 2 3
         0 1 0 0 1
         1 0 5 0 1
         2 1 0 3 0
         3 0 0 0 1

Overall Statistics

               Accuracy : 0.7692
                 95% CI : (0.4619, 0.9496)
    No Information Rate : 0.3846
    P-Value [Acc > NIR] : 0.005614

                  Kappa : 0.675

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity           0.50000   1.0000   1.0000  0.33333
Specificity           0.90909   0.8750   0.9000  1.00000
Pos Pred Value        0.50000   0.8333   0.7500  1.00000
Neg Pred Value        0.90909   1.0000   1.0000  0.83333
Precision             0.50000   0.8333   0.7500  1.00000
Recall                0.50000   1.0000   1.0000  0.33333
F1                    0.50000   0.9091   0.8571  0.50000
Prevalence            0.15385   0.3846   0.2308  0.23077
Detection Rate        0.07692   0.3846   0.2308  0.07692
Detection Prevalence  0.15385   0.4615   0.3077  0.07692
Balanced Accuracy     0.70455   0.9375   0.9500  0.66667
```

Lampiran 16 Data Set

Gen	GSM3478363	GSM3478364	GSM3478365	…	…	GSM3478421
16657436	2.98	2.47	2.89	…	…	2.65
16657440	3.54	3.40	4.08			3.74
16657445	1.90	2.59	2.84	…	…	3.09
16657447	2.81	1.66	1.81	…	…	5.86
16657450	5.11	6.64	6.31	…	…	8.60
16657469	4.50	4.28	4.11	…	…	5.38
16657473	1.85	2.39	1.52	…	…	2.10
16657476	3.10	3.67	3.56	…	…	3.90
16657480	2.79	3.05	2.93	…	…	2.88
:	:	:	:	…	…	:
:	:	:	:	…	…	:
16657514	2.48	3.44	2.88	…	…	2.50

Lampiran 17 Data *Training*

16650511	16650887	16650939	16651145	…	…	dataY
3.54	2.21	2.74	1.34	…	…	0
2.25	2.56	2.88	2.22	…	…	0
5.03	1.64	0.89	2.82	…	…	0
5.12	2.56	3.76	2.87	…	…	0
3.42	2.46	4.37	1.95	…	…	0
5.71	5.26	6.25	2.75	…	…	0
3.22	2.93	1.75	1.20	…	…	1
2.83	1.53	0.97	2.91	…	…	1
:	:	:	:	…	…	:
:	:	:	:	…	…	:
2.77	2.78	2.23	1.78	…	…	3

Lampiran 18 Data *Testing*

16650511	16650887	16650939	16651145	…	…	dataY
4.63	2.33	3.11	2.00	…	…	0
5.22	5.01	4.51	2.84	…	…	0
2.62	1.87	2.23	1.26	…	…	1
4.14	1.55	4.80	2.07	…	…	1
1.71	0.80	1.94	1.62	…	…	1
2.60	1.94	0.87	1.44	…	…	1
3.53	3.10	3.52	1.47	…	…	1
3.24	5.56	4.68	2.01	…	…	2
:	:	:	:	…	…	:
:	:	:	:	…	…	:
4.48	1.37	5.62	1.55	…	…	3