IMPLEMENTASI METODE SVM, MLP DAN XGBOOST PADA DATA EKSPRESI GEN

(Studi Kasus: Klasifikasi Data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe-2 GSE18732)

TUGAS AKHIR

Iqbal Fathur Rahman

16611120

PROGRAM STUDI STATISTIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2020

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul	:	Implementasi Metode SVM, MLP Dan XGBoost Pada Data Ekspresi Gen. (Studi Kasus: Klasifikasi Data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe-2 GSE18732)
Nama Mahasiswa	:	Iqbal Fathur Rahman
Nomor Mahasiswa	:	16611120

TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK DIUJIKAN

Yogyakarta, 9 Mei 2020

Menyetujui,

Dosen Pembimbing 1

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

HALAMAN PENGESAHAN

TUGAS AKHIR

IMPLEMENTASI METODE SVM, MLP DAN XGBOOST PADA DATA EKSPRESI GEN

(Studi Kasus: Klasifikasi Data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe-2 GSE18732)

Nama Mahasiswa : Iqbal Fathur Rahman

NIM : 16611120

TUGAS AKHIR INI TELAH DIUJIKAN
PADA TANGGAL: 16 MEI 2020

Nama Penguji:	**Tanda Tangan**
1. Dr. techn. Rohmatul Fajriyah, S.Si., M.Si	……..
2. Muh. Hasan Sidiq Kurniawan, M.Sc	……..
3. Dina Tri Utari, S.Si., M.Sc.	……..

Mengetahui,
Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Islam Indonesia

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

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Assalamu'alaikum Warahmatullaahi Wabarakaatuh

Alhamdulillahi Robbil 'Alamin, segala puji bagi Allah SWT Tuhan Semesta Alam. Karena atas berkat rahmat, hidayah, karunia dan nikmat-Nya penulis dapat menyusun dan menyelesaikan tugas akhir yang berjudul **"Implementasi Metode SVM, MLP Dan XGBoost Pada Data Bioinformatics"** Studi Kasus: Klasifikasi Data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe 2 dengan lancar dan sebaik-baiknya sebagai salah satu persyaratan dalam menyelesaikan jenjang strata satu di Jurusan Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Shalawat serta salam tak lupa penulis haturkan kepada junjungan Nabi Agung, Nabi Muhammad SAW beserta keluarga, sahabat dan umatnya yang selalu membimbing ke jalan yang penuh berkah ini.

Penyusunan tugas akhir ini tidak terlepas dari banyak dukungan, bantuan, arahan, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Kedua Orang Tua saya yakni Ibu (Basemi) dan Bapak (Sumantri) yang telah banyak berkorban dan berjuang mulai dari tenaga, materi, pikiran untuk dapat menyekolahkan saya diperguruan tinggi, dan selalu memotivasi, menghibur, dan mendoakan yang terbaik sehingga saya selalu mendapatkan hidayah dari Allah SWT selama masa perkuliahan hingga saat ini.
2. Kakak terbaik saya yakni Mas Adi yang telah banyak berkorban juga untuk membantu saya, sehingga saya dapat kuliah dan menyelesaikan kuliah saya ini. Adik-adik saya yakni Dek Silvi Yasmine Faradila dan Alya Shafa Kamila yang telah memberi saya semangat dan motivasi untuk dapat menjadi anak yang bisa diandalkan kelak, dan dapat memperbaiki perekonomian keluarga untuk membantu sesama.

3. Si mbah (Nenek) saya yang selalu memberikan semangat kepada saya untuk kuliah dan telah membantu bapak-ibu saya dalam menyekolahkan anak-anaknya sampai pada jenjang tertinggi.
4. Bapak Prof. Riyanto S.Pd., M.Si., Ph.D, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia
5. Bapak Dr. Edy Widodo, S.Si., M.Si, selaku Ketua Program Studi Statistika, Universitas Islam Indonesia
6. Ibu Dr. techn. Rohmatul Fajriyah, S.Si., M.Si, selaku Dosen Pembimbing tugas akhir yang telah mengorbankan waktu luangnya untuk sabar dalam membimbing penyusunan tugas akhir ini, dan berbagi nikmat rezeki kepada kami para bimbingannya, serta selalu memotivasi atas kegagalan yang kami peroleh saat proses penyusunan tugas akhir
7. Bapak Ibu Dosen Program Studi Statistika, Universitas Islam Indonesia yang telah mengajari saya dikelas dari semester awal hingga akhir, sehingga saya dapat memperoleh banyak pengalaman dan ilmu terbaik yang dapat saya tuangkan dalam penyusunan tugas akhir ini
8. Pengurus inti IKS FMIPS UII Periode 2018/2019, Dept Keilmuan IKS FMIPA UII Periode 2017/2018, Trial Lem UII, Fungsionaris IKS FMIPA UII Periode 2018/2019 yang telah membantu saya dalam menyelesaikan amanah-amanah dan tanggung jawab saya selama berada di lembaga/organisasi serta memberikan pengalaman baru tentang kepemimpinan dalam berorganisasi, berkapanitiaan, berbicara di depan umum dan mementingkan kepentingan umat diatas kepentingan pribadi agar dapat menjadi orang yang bermanfaat bagi orang lain.
9. Sarno Family, Agen Venus, KKN Unit 203, Kontrakan Ningrat, Hadroh artcos yang selalu memberikan semangat, motivasi, hiburan, pengingat dalam hal kebaikan dan tim sukses atas segala pencapaian saya.
10. Sahabat seperjuangan bimbingan TA bu ema, yang telah bekerjasama dengan baik, menemani penyusunan tugas akhir dan bertukar pikiran dalam penyempurnaan penyusunan skripsi hingga terlaksananya sidang pendadaran.

11. Seluruh teman-teman jurusan statistika angkatan 2016 yang telah banyak memberi informasi terkait akademik dan banyak bertukar ilmu selama 4 tahun berjuang bersama untuk menyandang gelar sarjana
12. Serta seluruh pihak-pihak lainnya yang tidak dapat disebutkan satu per satu, yang telah banyak membantu terselesaikannya tugas akhir ini dengan penuh drama dan perjuangan

Meskipun tugas akhir ini telah disusun dengan sebaik-baiknya, namun penulis menyadari bahwa tugas akhir ini masih banyak membutuhkan bantuan koreksi dari berbagai pihak, baik berupa saran, kritik, pembaharuan, dan penyempurnaan. Penulis mengucapkan terima kasih dan mohon maaf atas segala kekurangan yang ada, karena kesempurnaan hanyalah milik Allah SWT. Semoga tugas akhir ini dapat bermanfaat tidak untuk penulis saja, namun juga dapat bermanfaat sebagai bahan penelitian yang dapat diaplikasikan dimasa mendatang yang berguna bagi banyak orang. Akhirul kata, wabillahi taufik walhidayah

Wassalamu'alaikum Warahmatullahi Wabarakaatuh

Yogyakarta, 06 Maret 2020

(Iqbal Fathur Rahman)

DAFTAR ISI

DAFTAR ISTILAH

Akurasi	:	Tingkat ketepatan hasil pengukuran terhadap nilai sebenarnya
Batch Size	:	Jumlah sampel data yang disebarkan ke *Neural network* atau ukuran dari satuan kecil *Epoch* yang dimasukkan ke dalam *computer.*
Boosting	:	Teknik *ensemble* atau penggabungan beberapa model baik klasifikasi ataupun regresi
Class	:	Variabel pada penelitian yang menjadi target klasifikasi
Data Training	:	Data yang digunakan untuk melatih model
Data Testing	:	Data yang digunakan untuk menguji model
Diabetes Melitus	:	Penyakit kronis yang ditandai dengan tingginya kadar gula darah diatas kadar gula darah normal
Epoch	:	Perulangan pelatihan pada neural network sebagai proses pelatihan model
Hyperplane	:	Suatu bidang pemisah yang digunakan
Iterations	:	Jumlah *batch* yang diperlukan untuk menyelesaikan satu *epoch* atau *epoch* tanpa rambatan balik, contoh ilustrasi 2 kali iterasi a-b-c-d-a-b-c-d
IGT	:	Keadaan transisi antara kadar gula darah normal dan diabetes melitus
Loss	:	Tingkat ketidaktepatan hasil pengukuran/prediksi dengan nilai sebenarnya
Machine Learning	:	Suatu sistem yang dibangun berdasarkan pembelajaran menggunakan data-data besar
Margin	:	Jarak antara bidang pemisah (*hyperplane*) dengan data terdekatnya
Microarray	:	suatu teknologi yang digunakan untuk mengukur tingkat ekspresi dari gen yang ada pada suatu jaringan atau sel tertentu

Neuron	:	Biasa disebut dengan node/unit yaitu simpul dalam *neural network* yang biasanya menggunakan beberapa nilai *input* dan menghasilkan satu nilai *output*
Parameter	:	Tempat untuk nilai variabel yang dimasukkan kedalam *function* pada model
Perceptron	:	Sel saraf buatan untuk memodelkan sebuah sel saraf manusia
Supervised Learning	:	Pembelajaran pada machine learning untuk data yang telah diketahui kelasnya
Support Vector	:	Objek data terluar yang paling dekat dengan bidang pemisah (*hyperplane*)
Tuning	:	Proses penentuan parameter agar mendapatkan model yang terbaik
Unsupervised Learning	:	Pembelajaran pada *machine learning* untuk data yang belum diketahui kelasnya

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

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 juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 10 April 2020

(Iqbal Fathur Rahman)

IMPLEMENTASI METODE SVM, MLP DAN XGBOOST PADA DATA EKSPRESI GEN

(Studi Kasus: Klasifikasi Data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe-2 GSE18732)

Iqbal Fathur Rahman

Program Studi Statistika, Fakultas MIPA

Universitas Islam Indonesia

INTISARI

Diabetes Melitus (DM) adalah penyakit yang berlangsung lama atau kronis yang ditandai dengan naiknya kadar gula (glucose) pada darah yang tinggi. Diabetes melitus diklasifikasikan menjadi diabetes melitus tipe 1 (DM tipe 1), diabetes melitus tipe 2 (DMT2), dan diabetes melitus yang terjadi pada saat kehamilan. Menurut organisasi kesehatan dunia World Health Organization (WHO), pada tahun 2000 terdapat 171 juta orang menderita diabetes melitus dan diprediksi akan mengalami peningkatan 2 kali lipat menjadi 366 juta jiwa pada tahun 2030. Berdasarkan data dari IDF 95% dari total penderita diabetes melitus merupakan penderita diabetes melitus tipe 2 (DMT2). Diabetes melitus tipe 2 merupakan penyakit metabolik dengan karakteristik hiperglikemia yang terjadi karena kelainan sekresi insulin, kerja insulin atau kedua-duanya. Berbeda dengan diabetes melitus tipe 1 yang disebabkan karena kerusakan pankreas sehingga tidak dapat cukup memproduksi insulin, penderita diabetes melitus tipe 2 biasanya disebabkan karena pola makan yang tidak sehat dan jarang berolahraga, sedangkan diabetes melitus tipe 1 biasanya menyerang anak-anak dikarenakan kelainan genetik sejak lahir. Seiring majunya perkembangan tekhnologi, kini telah berkembang suatu bidang ilmu baru yaitu bioinformatika. Salah satu implementasi dari bioinformatika adalah digunakannya metode-metode komputasi, matematika, dan statistika dalam membantu menyelesaikan permasalahan-permasalahan biologi melalu analisis data ekspresi gen. Pada penelitian ini dilakukan analisis klasifikasi pada data microarray hasil dari ekspresi gen pada sampel diabetes melitus tipe 2, Impaired Glucose Tolerance (IGT), dan sampel dengan kadar gula darah normal (NGT) dengan kode series GSE18732. Pada penelitian ini digunakan metode klasifikasi Support Vector Machine (SVM), Arsitektur Multilayer Perceptron (MLP), dan Xtreme Gradient Boosting (XGBoost) untuk menganalisis data tersebut. Hasil dari analisis yang dilakukan, didapatkan bahwa metode klasifikasi Support Vector Machine dengan menggunakan kernel Linear mampu mendapatkan nilai akurasi terbesar yaitu sebesar 91,30%. Sedangkan arsitektur Multilayer Perceptron dengan satu hidden layer dan 100 hidden node mampu mendapatkan

nilai akurasi sebesar 78,26% dan metode klasifikasi terakhir Xtreme Gradient Boosting mampu mendapatkan nilai akurasi sebesar 71,39%.

Kata Kunci: Diabetes Melitus, Bioinformatika, *Microarray*, Klasifikasi, SVM, MLP, XGBoost

IMPLEMENTATION OF SVM, MLP, AND XGBOOST METHOD FOR GENE EXPRESSION DATA

(*Case Study: Classification of Gene Expression Data for NGT, IGT, and Type 2 Diabetes Mellitus, GSE18732*)

Iqbal Fathur Rahman

Department of Statistics, Faculty of Mathematics and Natural Sciences

Islamic University of Indonesia

ABSTRACT

Diabetes mellitus (DM) is a long-lasting or chronic disease characterized by elevated blood sugar (glucose) levels. Diabetes mellitus is classified into type 1 diabetes mellitus (DM type 1), type 2 diabetes mellitus (DMT2), and diabetes mellitus that occurs during pregnancy. According to the world health organization World Health Organization (WHO), in 2000 there were 171 million people suffering from diabetes mellitus and is predicted to experience a two-fold increase to 366 million by 2030. Based on data from IDF 95% of the total diabetes mellitus sufferers are sufferers type 2 diabetes mellitus (T2DM). Type 2 diabetes mellitus is a metabolic disease characterized by hyperglycemia which occurs due to abnormal insulin secretion, insulin action or both. Unlike type 1 diabetes mellitus caused by pancreatic damage so that it cannot produce enough insulin, people with type 2 diabetes mellitus are usually caused by unhealthy eating patterns and rarely exercise, whereas type 1 diabetes mellitus usually attacks children due to genetic abnormalities from birth. Along with the development of technology, now has developed a new field of science, namely bioinformatics. One implementation of bioinformatics is the use of computational, mathematical and statistical methods to help solve biological problems through analysis of gene expression data. In this study a classification analysis was performed on the microarray data resulting from gene expression in type 2 diabetes mellitus, Impaired Glucose Tolerance (IGT) samples, and samples with normal blood sugar levels (NGT) with GSE18732 series code. In this study the classification method Support Vector Machine (SVM), Multilayer Perceptron Architecture (MLP), and Xtreme Gradient Boosting (XGBoost) are used to analyze the data. The results of the analysis conducted, it was found that the Support Vector Machine classification method using Linear kernel was able to get the greatest accuracy value that is equal to 91.30%. While the Multilayer Perceptron architecture with one hidden layer and 100 hidden nodes is able to get an accuracy value of 78.26% and the final classification method Xtreme Gradient Boosting is able to get an accuracy value of 71.39%.

Keywords: *Diabetes Mellitus, Bioinformatics, Microarray, Classification, SVM, MLP, XGBoost*

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Diabetes Melitus (DM) adalah penyakit yang berlangsung lama atau kronis yang ditandai dengan kadar gula (*glucose*) darah yang tinggi diatas nilai normal. Kadar gula darah normal pada tubuh adalah:

- Sebelum makan: sekitar 70-130 mg/dL
- Dua jam setelah makan: kurang dari 140 mg/dL
- Saat tidak makan (puasa) setidaknya selama 8 jam: kurang dari 100 mg/dL
- Menjelang tidur: 100-140 mg/dL

Diabetes melitus menjadi permasalahan kesehatan dunia karena prevalensi dan kejadian penyakit ini terus meningkat baik dinegara maju maupun negara berkembang seperti Indonesia. Menurut organisasi kesehatan dunia *World Health Organizatiom* (WHO), pada tahun 2000 terdapat 171 juta orang menderita diabetes melitus dan diprediksi akan meningkat 2 kali lipat menjadi 366 jiwa pada tahun 2030. Penderita diabetes melitus ini selalu mengalami kenaikan setiap tahunnya hal ini terbukti pada awal tahun 2018 WHO mencatat sekitar 422 juta orang di seluruh dunia menderita diabetes melitus dimana angka ini telah jauh berada diatas perkiraan WHO pada tahun 2030 (WHO, 2018). Hal yang sama juga dijelaskan oleh *International Diabetes Federation* (IDF), menurut catatan dari IDF jumlah penderita diabetes pada tahun 2015 mencapai 415 juta dan pada tahun 2017 mencapai 425 juta angka ini akan diprediksi naik menjadi 625 juta jiwa pada tahun 2045.

Di Indonesia kejadian diabetes melitus juga selalu mengalami kenaikan setiap tahunnya, berdasarkan Riset Kesehatan Dasar (RISKESDAS) dari tahun 2013 hingga tahun 2018 penderita diabetes melitus meningkat dari 6,9% menjadi 8,5% yang berarti terdapat sekitar 22,9 juta penduduk di Indonesia menderita diabetes melitus. Dirjen P2P Kemenkes RI dr Anung Sugihantono M.Kes, mengatakan

“Peningkatan prevalensi DM disebabkan antara lain karena Indonesia saat ini dalam masa transisi, demografi, teknologi, epidemiologi, budaya, perilaku ekonomi, dan lain-lain. Hal ini tentunya dapat menimbulkan terjadinya ledakan peningkatan kasus-kasus komplikasi DM seperti gagal ginjal, penyakit jantung koroner, stroke dan lain-lain.” (suara.com, 2019)

Peningkatan prevalensi diabetes melitus menjadikannya penyebab kematian urutan ke tujuh didunia, pada tahun 2012 data dari IDF menunjukkan jumlah kejadian diabetes melitus didunia adalah sebanyak 371 juta jiwa dan 95% dari total penderita diabetes melitus merupakan penderita diabetes melitus tipe 2 (DMT2). Diabetes melitus tipe 2 merupakan penyakit metabolik dengan karakteristik hiperglikemia yang terjadi karena kelainan sekresi insulin, kerja insulin atau kedua-duanya. Berbeda dengan diabetes melitus tipe 1 yang disebabkan karena kerusakan pankreas sehingga tidak dapat cukup memproduksi insulin, penderita diabetes melitus tipe 2 biasanya disebabkan karena pola makan yang tidak sehat dan jarang berolahraga, sedangkan diabetes melitus tipe 1 biasanya menyerang anak-anak dikarenakan kelainan genetik sejak lahir.

Saat ini perkembangan teknologi informasi merupakan suatu hal yang tidak dapat dihindari. Seiring dengan perkembangan teknologi informasi telah memunculkan berbagai kajian ilmu lain salah satunya adalah bioinformatika. Menurut Fredj Tekaian dari Institut Pasteur, bioinformatika adalah “metode matematika, statistika dan komputasi yang bertujuan untuk menyelesaikan masalah-masalah biologi dengan menggunakan sekuen DNA dan Asam Amino dan informasi-informasi yang terkait didalamnya”. Bioinformatika merupakan aplikasi dari alat komputasi dan analisis yang digunakan untuk menginterpretasikan data-data biologi (Ardeshir, 2002). Saat ini bioinformatika telah diaplikasikan dalam berbagai bidang salah satunya pada bidang klinis yang digunakan untuk mengidentifikasi *agent* penyakit baru, diagnosis penyakit baru, dan penemuan obat-obatan. Analisis pada bioinformatika difokuskan pada tiga jenis dataset, yaitu urutan genom, struktural makromolekul dan percobaan *genomic fungsional* (Wargasetia, 2006).

Salah satu teknologi yang berkembang pada bioinformatika adalah teknologi *microarray*, sebuah perangkat berupa *chip* yang didalamnya berisi ribuan gen. Analisis *microarray* merupakan proses analisis ekspresi gen (*gene expression*) melalui *microarray*. *Gene expression* adalah proses penerjemahan informasi dari gen menjadi produk gen yang berguna yaitu, protein (Sanchez & de Villa, 2008). Data *microarray* dapat digunakan dalam mendeteksi dan mengklasifikasikan suatu jaringan penyakit di manusia. *Microarray* menghasilkan ekspresi gen yang berisi informasi gen-gen yang kemudian dicocokkan dengan suatu penyakit tertentu salah satunya adalah DMT2 (Trevino dkk, 2007). Teknik komputasi seperti teknik *machine learning* dapat digunakan untuk menganalisis pada pemilihan gen atau protein yang mempunyai sifat yang terkait dan untuk mengklasifikasikan tipe dari sampel ekspresi gen pada data *microarray* (Yang dkk, 2016). Selain itu *machine learning* juga membantu menyelesaikan permasalahan-permasalahan pada data-data dengan dimensi tinggi seperti data *microarray*.

Machine learning merupakan suatu sistem yang dibangun berdasarkan pembelajaran pada data-data yang besar. *Machine learning* terbagi menjadi 2 yaitu *supervised learning* dan *unsupervised learning*, *supervised learning* digunakan untuk serangkaian pengamatan dimana hasil yang telah didapatkan telah diketahui kelasnya sedangkan *unsupervised learning* digunakan untuk serangkaian pengamatan dimana hasil yang didapatkan belum diketahui kelasnya. Contoh metode pada *supervised learning* adalah pemodelan yang digunakan pada klasifikasi dimana pada penelitian ini akan digunakan beberapa metode klasifikasi seperti *Support Vector Machine* (SVM), *Multilayer Perceptron* (MLP) dan *Xtreme Gradient Boosting* (XGBoost).

Support Vector Machine (SVM) adalah suatu pemodelan klasifikasi dengan konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. SVM digunakan untuk mencari *hyperplane* terbaik dengan memaksimalkan jarak antar kelas. Algoritma SVM mempunyai 4 tipe kernel yang berbeda dalam mengklasifikasikan suatu data, 4 tipe kernel tersebut adalah *Linier, Polynomial, Sigmoid* dan *Radial Basis Function* (RBF).

Klasifikasi menggunakan algoritma SVM umumnya secara langsung memilih tipe kernel tertentu tanpa melihat akurasi pada tipe kernel lainnya (Suyanto, 2017).

Multilayer Perceptron (MLP) merupakan suatu algoritma pada pembuatan model *neural network*. MLP merupakan *perceptron* dengan satu atau lebih *hidden layer*, dimana *input* sebuah *perceptron* adalah *output* dari *perceptron* sebelumnya. Tidak seperti *perceptron* yang hanya dapat memodelkan permasalahan linear, MLP juga dapat menyelesaikan permasalahan non-linear (Suyanto, 2017).

Xtreme Gradient Boosting (XGBoost) merupakan suatu metode pada machine learning dimana XGBoost merupakan algoritma regresi dan klasifikasi dengan metode ensemble yang merupakan suatu varian dari algoritma *Tree Gradient Boosting* yang dikembangkan dengan optimasi 10 kali lebih cepat dibandingkan *Gradient Boosting* lainnya (Chen & Guestrin, 2016).

Berdasarkan uraian latar belakang diatas, maka peneliti melakukan analisis dengan judul **Implementasi Metode SVM, MLP Dan XGBoost Pada Data Bioinformatics** dengan studi kasus: Klasifikasi data Ekspresi Gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe 2, yang mana harapannya dari hasil penelitian ini dapat diketahui metode apa yang paling baik dalam mengklasfikasikan gen pada seseorang dengan kadar gula darah normal, IGT dan diabetes melitus tipe 2 berdasarkan nilai akurasi.

1.2 Rumusan Masalah

Adapun rumusan masalah yang diangkat dari latar belakang diatas adalah sebagai berikut:

a. Bagaimana gambaran data ekspresi gen *skeletal muscle* NGT, IGT dan diabetes melitus tipe 2 pada data *microarray*?
b. Bagaimana pengklasifikasian pada data ekspresi gen *Skeletal Muscle* NGT, IGT dan Diabetes Melitus Tipe 2 dengan menggunakan metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting*?

c. Bagaimana evaluasi terhadap hasil klasifikasi dengan menggunakan metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting*?

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini antara lain:

a. Data yang digunakan pada penelitian ini adalah data dari *website* NCBI dengan series GSE18732.
b. Data terdiri atas 3 kelas yaitu sampel dengan kadar gula darah normal, *impaired glucose tolerance,* dan diabetes melitus tipe 2.
c. *Software* yang digunakan dalam penelitian ini adalah aplikasi *R Studio* 3.6.2 dengan menggunakan beberapa *package* bioinformatika.
d. Melakukan klasifikasi dengan menggunakan metode *Support Vector Machine, Xtreme Gradient Boosting* dan *Multilayer Perceptron*.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

a. Mengetahui gambaran data ekspresi gen *skeletal muscle* NGT, IGT dan diabetes melitus tipe 2 pada data *microarray*.
b. Melakukan pengklasifikasian pada data ekspresi gen *Skeletal muscle* NGT, IGT dan Diabetes Melitus Tipe 2 dengan menggunakan metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting*.
c. Melakukan evaluasi dari hasil klasifikasi dengan menggunakan metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan beberapa manfaat diantaranya adalah:

a. Bagi Penulis

Dapat menerapkan ilmu yang diperoleh di perkuliahan dan menambah pengetahun baru dalam ilmu bioinformatika.

b. Bagi Pembaca

Dapat memperoleh informasi dan pengetahuan mengenai implementasi dari metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting* yang digunakan untuk mengklasifikasikan data bioinformatika.

c. Bagi Penelitian Lanjutan

Dapat mengembangkan teknologi dengan metode lainnya untuk mengoptimalkan pengklasifikasian dalam bidang kesehatan khususnya untuk memprediksi penyakit diabetes melitus tipe 2.

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan dipaparkan beberapa penelitian sebelumnya yang mempunyai keterkaitan terhadap beberapa metode dan acuan yang digunakan oleh peneliti. Penelitian terdahulu digunakan sebagai kajian yang bertujuan mengetahui keterkaitan penelitian terdahulu dengan penelitian yang akan dilakukan untuk mengurangi kesalahan dan menghindari terjadinya duplikasi. Berikut beberapa penelitian tentang penerapan metode *Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting*

2.1 Penelitian Menggunakan Metode SVM

Penelitian lainnya dilakukan oleh (Handayani dkk, 2017) berjudul "Evaluasi Tiga Jenis Algoritme Berbasis Pembelajaran Mesin untuk Klasifikasi Jenis Tumor Payudara". Penelitian ini membandingkan tiga algoritma yaitu *Xtreme Gradient Boosting* (XGBoost), *Support Vector Machine Kernel Radial Basic Function* (SVM-RBF), dan *Multilayer Perceptron* (MLP) berdasarkan nila *Area Under Curve* (AUC). Data yang digunakan pada penelitian ini adalah data sekunder yang didapatkan dari *UCI Machine Learning Repository*. Dari hasil yang didapatkan pada penelitian ini, SVM merupakan algoritma yang paling tepat digunakan untuk mengklasifikasikan jenis tumor payudara, algoritma SVM mampu menghasilkan nilai AUC sebesar 99,23 dimana nilai ini lebih tinggi dari nilai AUC yang didapatkan oleh kedua algoritma lainnya.

Dalam penelitian lain yang dilakukan oleh (Puspitasari dkk, 2018) berjudul "Klasifikasi Penyakit Gigi dan Mulut Menggunakan Metode *Support Vector Machine* (SVM)". Penelitian yang dilakukan ini bertujuan untuk mengklasifikasi jenis penyakit gigi dan mulut untuk membantu masyarakat dalam melakukan diagnosa awal terhadap penyakit gigi dan mulut. Metode klasifikasi yang digunakan yaitu SVM, karena dapat mengatasi masalah klasifikasi dan regresi dengan *Linear* maupun non *Linear*. Pada proses klasifikasi digunakan kernel *trick*

untuk menentukan kernel terbaik yang akan digunakan. Kernel yang digunakan yaitu kernel *Polynomial, Linear* dan *Radial Basis Function* (RBF). Hasil dari masing-masing kernel yaitu pada kernel *polynomial degree* 32,216%, *Linear* 56,66% dan RBF 93,328%. Dari hasil kernel, maka digunakan kernel RBF karena proses yang akan dilakukan bersifat non linear. Pencapaian hasil akurasi yang baik dapat diterapkan untuk membantu melakukan klasifikasi penyakit gigi dan mulut dengan metode SVM.

(Sugara & Subekti, 2019) melakukan penelitian yang berjudul "Penerapan *Support Vector Machine* (SVM) Pada *Small Dataset* Untuk Deteksi Dini Gangguan Autisme". Dalam penelitian ini peneliti menggunakan algoritma *Support Vector Machine* dan *k-fold cross validation* untuk menguji nilai keakuratan *small dataset* menggunankan algoritma serta menggunakan teknik *ensemble* terhadap algoritma *Support Vector Machine*. Data yang digunakan dalam penelitian deteksi dini gangguan autism ini merupakan data primer yang diperoleh dari hasil wawancara disebuah lembaga autisme di Bekasi Jawa Barat, selain wawancara data penelitian ini juga berasal dari hasil sebara kuisioner online yang dapat diakses menggunakan url https://forms.gle/YSvYcnRxi21dLzfP9. Data terdiri atas 2 parameter yaitu data gejala autisme yang digunakan sebagai atribut pada penelitian ini sedangkan data gangguan autisme digunakan sebagai *class*. Hasil dari penelitian ini menunjukkan bahwa model SVM dengan *poly kernel* dan teknik *ensemble Bagging* mempunyai nilai akurasi tertinggi yaitu sebesar 91%. Nilai akurasi ini lebih tinggi jika dibandingkan dengan nilai akurasi yang dihasilkan dari model SVM dengan *normalized poly kernel* tanpa menggunakan teknik *ensemble* yang mendapatkan nilai akurasi sebesar 85%.

(Audina dkk, 2019) juga melakukan penelitian yang berjudul "Analisis Perubahan Lahan Dan Sebaran Mangrove Menggunakan Algoritma Support Vector Machine (SVM) Dengan Citra Landsat Di Kabupaten Bintan Kepulauan Riau". Penelitian ini bertujuan untuk memetakan dan menganalisis perubahan lahan dan tutupan mangrove di Pesisir Bintan, Kepulauan Riau yang nantinya dapat bermanfaat bagi masyarakat luas. Data yang digunakan pada penelitian ini adalah data citra satelit Landsat 5 dan Landsat 8. Sedangkan data *insitu* yang digunakan

adalah data jenis mangrove, substrat dasar perairan dan kecepatan arus di Desa Berakit dan Pengundang. Hasil penelitian menunjukkan mangrove, permukiman dan perkebunan mengalami pertambahan luasan dan hasil klasifikasi tutupan lahan menunjukkan algoritma SVM kernel Radial Basis Function (RBF) memberikan akurasi yang tinggi, yaitu 70,42% dengan koefisien kappa 0,61, sedangkan hasil uji signifikansi menunjukkan bahwa SVM dengan kernel RBF tidak memiliki perbedaan yang signifikan dengan kernel sigmoid.

2.2 Penelitian Menggunakan Metode MLP

Penelitian lain yang dilakukan oleh (Ardilla dkk, 2014) berjudul "Deteksi Penyakit Epilepsi dengan Menggunakan Entropi Permutasi, K-means *Clustering*, dan *Multilayer Perceptron*" bertujuan untuk memprediksi apakah seseorang menderita epilepsy bebas kejang, atau epilepsy kejang. Pada penelitian ini digunakan 3 metode untuk memprediksi yaitu entropi permutasi, K-means *clustering*, dan *multilayer perceptron*. Dari hasil penelitian yang dilakukan didapatkan hasil bahwa model klasifikasi *multilayer perceptron* akan menghasilkan performa maksimal ketika dilakukan eksekusi pada algoritma entropi permutasi dengan jumlah dimensi antara 3 hingga 6, dan eksekusi pada algoritma k-means *clustering* yang telah dimodifikasi dengan jumlah *cluster* antara 3 serta pada jumlah *hidden layer neuron* antara nilai 3. Model ini mendapatkan nilai performa dengan akurasi 96,5%, *specificity* sebesar 95,45% dan *sensitivity* sebesar 97,97%.

(Purwaningsih, 2016) melakukan penelitian yang berjudul "Penerapan Multilayer Perceptron Untuk Klasifikasi Jenis Kulit Sapi Tersamak". Penelitian ini bertujuan untuk melakukan klasifikasi jenis kulit sapi tersamak berdasarkan analisis tekstur permukaannya. Manfaat yang bisa diperoleh dari penelitian ini adalah pengetahuan mengenai perbedaan karakteristik tekstur kulit sapi tersamak berdasarkan jenisnya. Dalam penelitian ini MLP yang dibentuk terdiri atas tiga buah lapisan (*layer*) sebagai berikut:

a. Satu *input layer* terdiri atas lima neuron.
b. Satu *hidden layer* terdiri atas empat neuron.
c. Satu *output layer* terdiri atas empat neuron.

Hasil dari penelitian ini menunjukkan tingkat ketepatan klasifikasi kulit tersamak dengan menggunakan MLP mencapai 87,83% dengan tingkat akurasi sebesar 98,75%.

(Naf'an & Arifin, 2017) melakukan penelitian yang berjudul "Identifikasi Tanda Tangan Berdasarkan *Grid Entropy* Menggunakan *Multi Layer Perceptron*". Tanda tangan merupakan salah satu bukti pengesahan dokumen yang sering digunakan. Pentingnya mengenal bentuk tanda tangan seseorang diperlukan untuk melakukan verifikasi terhadap dokumen apakah benar yang memberikan tanda tangan adalah orang yang bersangkutan atau orang lain. Data yang digunakan pada penelitian ini adalah dataset tanda tangan menggunakan citra tanda tangan asli dari penelitian lain, yaitu tanda tangan yang dikumpulkan dari 30 responden dan setiap responden akan menuliskan tanda tangan pada kertas sebanyak 30 kali. Hasil penelitian menunjukkan bahwa pengujian terbaik adalah untuk pengujian ukuran *grid* 8x8 dan menggunakan *citra outline*, yaitu dengan tingkat akurasi sebesar 97,78%, nilai korelasi 0,981 dan nilai kappa 0,977.

2.3 Penelitian Menggunakan Metode XGBoost

Penelitian yang dilakukan oleh (Prasetyo dkk, 2019) berjudul "Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan *Machine Learning* di Wilayah Kabupaten Boyolali dan Purworejo. Hasil dari penelitian ini diharapkan dapat digunakan sebagai pertimbangan pemerintahdaerah dalam pengelolaan lahan serta sumber air yang ada untuk mengantisipasi bencana kekeringan perubahan iklim maupun kebutuhan SDM daerah. Dalam penelitian ini data yang digunakan adalah data citra satelit Landsat 8 OLI pada Kabupaten Boyolali dan Kabupaten Purworejo yang kemudian dianalisis menggunakan dua metode yaitu metode XGBoost dan *Random Forest*. Hasil analisis menunjukkan bahwa metode XGBoost mendapatkaa nilai akurasi 0,8286 dan nilai kappa 0,6477 sedangkan metode *Random Forest* mempunyai nilai akurasi 0,6857 dan nilai kappa 0,3699. Hal ini menunjukkan bahwa metode *XGBoost* lebih baik digunakan dibandingkan dengan metode *Random Forest* karena mempunyai nilai akurasi dan nilai kappa yang lebih tinggi.

Penelitian tentang XGBoost selanjutnya adalah penelitian yang dilakukan oleh (Maalik dkk, 2019) berjudul "Comparation Analysis of Ensemble Technique With Boosting (XGBOOST) and Bagging (Random Forest) for Classify Splice Junction DNA Sequence Category". Penelitian dilakukan untuk membandingkan dua teknik *ensemble* yaitu *boosting* menggunakan metode XGBoost dan Bagging dengan menggunakan metode *random forest* yang digunakan untuk mengklasifikasikan kategori sambatan sekuens DNA. Hasil dari penelitian menunjukkan nilai akurasi sebesar 96,24% untuk metode XGBoost dan 95,11% untuk metode *Random Forest*.

2.4 Penelitian Tentang Diabetes Melitus

Diabetes melitus merupakan faktor risiko dari berbagai penyakit kardiovaskuler. Gaya hidup yang tidak sehat dan kepatuhan terapi merupakan salah satu factor yang mempengaruhi peningkatan kadar gula darah penderita diabetes melitus. (Toharin dkk, 2015) melakukan penelitian untuk mengetahui modifikasi gaya hidup dan kepatuhan konsumsi obat anti diabetik dengan kadar gula darah pada penderita DM tipe 2. Penelitian dilakukan dengan menggunakan sampel sebanyak 58 dan uji *chi-square* dan *fisher*. Berdasarkan hasil penelitian disarankan perlunya bimbingan dan penyuluhan pada penderita tentang latihan jasmani, diet yang sehat, berhenti merokok, penggunaan obat antidiabetic dan efek sampingnya, serta pentingnya control gula darah, meningkatkan informasi tentang DM, komplikasi, dan penanggulangannya.

(Susilawati & Muljati, 2016) melakukan penelitian tentang hubungan antara DM dan toleransi glukosa terganggu dengan riwayat Tuberkulosis (TB) paru dewasa di Indonesia. Desain penelitian potong lintang dengan menggunakan data sekunder dari data Riset Kesahatan Dasar (Riskesdas) tahun 2013, Badan Litbangkes RI. Responden yang menderita DM tidak ada hubungannya dengan riwayat TB, namun jika terdiagnosis intoleransi glukosa berpeluang mengalami TB paru sebesar 42% atau *odds* 1,4 kali dibandingkan responden yang mempunyai glukosa normal ($p<0,05$; OR 1,36; 95% CI 1,06-1,61).

Penelitian lain yang dilakukan (Vidyanto & Arifuddin, 2019) menggunakan analisis regresi linier berganda menunjukkan bahwa terdapat beberapa faktor yang mempengaruhi peningkatan kadar gula diantaranya adalah aktivitas fisik, obesitas, total kolestrol dan merokok. Peningkatan kadar gula pada tubuh akan meningkatkan resiko seseorang menderita diabetes mellitus, dengan mengetahui factor-faktor yang mempengaruhinya akan dapat mengantisipasi resiko menderita diabetes mellitus. Data pada penelitian ini menggunakan data primer dengan jumlah responden sebanyak 117 sampel yang diperoleh dengan menggunakan kuisioner.

2.5 Perbedaan Penelitian Sebelumnya dengan Penelitian ini

Setelah mempelajari dan memahami beberapa penelitian terdahulu yang berkaitan dengan metode dan objek yang digunakan pada penelitian ini maka dapat diketahui perbedaan yang dimiliki dari penelitian ini dengan penelitian-penelitian sebelumnya yang disajikan dalam Tabel 2.1

Tabel 2.1. Perbedaan Penelitian ini dengan Penelitian Sebelumnya

Nama peneliti	Judul Penelitian	Tahun	Isi
Yunita Ardilla, Handayani Tjandrasa & Isye Arieshanti	Deteksi Penyakit Epilepsi dengan Menggunakan Entropi Permutasi, K-means *Clustering*, dan *Multilayer Perceptron*	2014	Bertujuan untuk memprediksi apakah seseorang menderita epilepsy bebas kejang, atau epilepsy kejang. Pada penelitian ini digunakan 3 metode untuk memprediksi yaitu entropi permutasi, K-means *clustering*, dan *multilayer perceptron*.
Syamsi Nur Rahman Toharin, Widya Hary Cahyati & Intan Zainafree	Hubungan Modifikasi Gaya Hidup Dan Kepatuhan Konsumsi Obat Antidiabetik Dengan Kadar Gula Darah Pada Penderita	2015	penelitian untuk mengetahui modifikasi gaya hidup dan kepatuhan konsumsi obat anti diabetik dengan kadar gula darah pada penderita DM tipe 2. Penelitian dilakukan dengan menggunakan sampel sebanyak 58 dan uji *chi-square* dan *fisher*.

	Diabetes Melitus Tipe 2 Di RS QIM Batang Tahun 2013		
Nunik Purwaningsih	Penerapan Multilayer Perceptron Untuk Klasifikasi Jenis Kulit Sapi Tersamak	2016	Penelitian ini bertujuan untuk melakukan klasifikasi jenis kulit sapi tersamak berdasarkan analisis tekstur permukaannya. Manfaat yang bisa diperoleh dari penelitian ini adalah pengetahuan mengenai perbedaan karakteristik tekstur kulit sapi tersamak berdasarkan jenisnya
Made Dewi Susilawati & Sri Muljati	Hubungan Antara Intoleransi Glukosa dan Diabetes Melitus dengan Riwayat Tuberkulosis Paru Dewasa di Indonesia	2016	Penelitian tentang hubungan antara DM dan toleransi glukosa terganggu dengan riwayat Tuberkulosis (TB) paru dewasa di Indonesia. Desain penelitian potong lintang dengan menggunakan data sekunder dari data Riset Kesahatan Dasar (Riskesdas) tahun 2013, Badan Litbangkes RI.
Annisa Handayani, Ade Jamal & Ali Akbar Septiandri	Evaluasi Tiga Jenis Algoritme Berbasis Pembelajaran Mesin untuk Klasifikasi Jenis Tumor Payudara	2017	Penelitian ini membandingkan tiga algoritma yaitu *Xtreme Gradient Boosting* (XGBoost), *Support Vector Machine Kernel Radial Basic Function* (SVM-RBF), dan *Multilayer Perceptron* (MLP) berdasarkan nila *Area Under Curve* (AUC). Data yang digunakan pada penelitian ini adalah data sekunder yang didapatkan dari *UCI Machine Learning Repository*.

Muhammad Zidny Naf'an & Jaenal Arifin	Identifikasi Tanda Tangan Berdasarkan *Grid Entropy* Menggunakan *Multi Layer Perceptron*".	2017	Data yang digunakan pada penelitian ini adalah dataset tanda tangan menggunakan citra tanda tangan asli dari penelitian lain, yaitu tanda tangan yang dikumpulkan dari 30 responden dan setiap responden akan menuliskan tanda tangan pada kertas sebanyak 30 kali.
Ana Mariyam Puspitasari, Dian Eka Ratnawati & Agus Wahyu Widodo	Klasifikasi Penyakit Gigi dan Mulut Menggunakan Metode *Support Vector Machine* (SVM)	2018	Penelitian yang dilakukan ini bertujuan untuk mengklasifikasi jenis penyakit gigi dan mulut untuk membantu masyarakat dalam melakukan diagnosa awal terhadap penyakit gigi dan mulut. Metode klasifikasi yang digunakan yaitu SVM, karena dapat mengatasi masalah klasifikasi dan regresi dengan *Linear* maupun non *Linear*.
Sugara & Subekti	Penerapan *Support Vector Machine* (SVM) Pada *Small Dataset* Untuk Deteksi Dini Gangguan Autisme	2019	Penelitian ini menggunakan algoritma *Support Vector Machine* dan *k-fold cross validation* untuk menguji nilai keakuratan *small dataset* menggunankan algoritma serta menggunakan teknik *ensemble* terhadap algoritma *Support Vector Machine*. Data yang digunakan dalam penelitian deteksi dini gangguan autism ini merupakan data primer yang diperoleh dari hasil wawancara disebuah lembaga autisme di Bekasi Jawa Barat, selain wawancara data penelitian ini juga berasal dari hasil sebaran kuisioner online.

NurAudina, Vicentius P.Siregar & I Wayan Nurjaya	Analisis Perubahan Lahan Dan Sebaran Mangrove Menggunakan Algoritma Support Vector Machine (SVM) Dengan Citra Landsat Di Kabupaten Bintan Kepulauan Riau	2019	Penelitian ini bertujuan untuk memetakan dan menganalisis perubahan lahan dan tutupan mangrove di Pesisir Bintan, Kepulauan Riau yang nantinya dapat bermanfaat bagi masyarakat luas. Data yang digunakan pada penelitian ini adalah data citra satelit Landsat 5 dan Landsat 8. Sedangkan data *insitu* yang digunakan adalah data jenis mangrove, substrat dasar perairan dan kecepatan arus di Desa Berakit dan Pengundang.
Sri Yulianto Joko Prasetyo, Yansen Bagas Christianto & Kristoko Dwi Hartomo	Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan *Machine Learning* di Wilayah Kabupaten Boyolali dan Purworejo	2019	Dalam penelitian ini data yang digunakan adalah data citra satelit Landsat 8 OLI pada Kabupaten Boyolali dan Kabupaten Purworejo yang kemudian dianalisis menggunakan dua metode yaitu metode XGBoost dan *Random Forest*.
Iswaya Maalik, Wisnu Ananta Kusuma & Sri Wahjuni	*Comparation Analysis of Ensemble Technique With Boosting (XGBOOST) and Bagging (Random Forest) for Classify Splice Junction DNA Sequence Category*	2019	Penelitian dilakukan untuk membandingkan dua teknik *ensemble* yaitu *boosting* menggunakan metode XGBoost dan Bagging dengan menggunakan metode *random forest* yang digunakan untuk mengklasifikasikan kategori sambatan sekuens DNA.
Penelitian yang penulis lakukan			

Iqbal Fathur Rahman	Implementasi *Metode Support Vector Machine, Multilayer Perceptron* dan *Xtreme Gradient Boosting* Pada Data *Bioinformatics*	2020	Perbedaan penelitian ini dengan penelitian-penelitian sebelumnya yakni terletak pada objek yang digunakan adalah data ekspresi gen pada *skeletal muscle* atau otot rangka yang diklasifikasikan menggunakan 3 metode klasifikasi yaitu *Support Vector Machine* (SVM), *Multilayer Perceptron* (MLP) dan *Xtreme Gradient Boosting* (XGBoost) yang kemudian hasil dari ketiga metode tersebut dibandingkan metode mana yang paling baik dalam memprediksi gen sampel dengan kadar gula darah normal (NGT), IGT dan menderita diabetes melitus tipe 2.

BAB III
LANDASAN TEORI

3.1 Bioinformatika

Bioinformatika merupakan bidang multidisiplin dari biologi, medis, matematika, statistika, ilmu komputer, dan ilmu lain yang berkaitan. Bioinformatka menggunakan gabungan berbagai teknik dari matematika terapan, statistika, informatika, ilmu komputer, sistem informasi, intelegensi buatan, kimia dan biokimia untuk memecahkan permasalahan biologi pada level molekuler (Pramana dkk, 2018). Menurut Fredj Tekaian dari Institut Pasteur, bioinformatika adalah "metode matematika, statistika dan komputasi yang bertujuan untuk menyelesaikan masalah-masalah biologi dengan menggunakan sekuen DNA dan Asam Amino dan informasi-informasi yang terkait didalamnya". Bioinformatika telah digunakan dan diaplikasikan dalam bidang lain untuk membantu menyelesaikan permasalahan-permasalahan yang ada seperti dalam bidang klinis, bioinformatika digunakan untuk mengidentifikasi *agent* penyakit baru, diagnosis penyakit baru dan penemuan obat-obatan. Salah satu analisis pada bioinformatika adalah analisis ekspresi gen yaitu sejumlah gen dapat ditentukan dengan mengukur levelnya dengan menggunakan berbagai teknik seperti *microarray* (Raza, 2012).

3.1.1 *Microarray*

Microarray merupakan suatu tekhnologi yang digunakan untuk mengukur tingkat ekspresi dari gen yang ada pada suatu jaringan atau sel tertentu. *Microarray* berbentuk seperti suatu chip atau *slide* mikroskop yang berisi rangkaian sampel jaringan, protein, RNA dan DNA (Hovatta dkk, 2005). Teknologi *microarray* biasa digunakan untuk *genotype* dengan skala besar, profil ekspresi gen, hibridasi genomic, dan penyeimbang diantara penggunaan aplikasi lainnya. *Microarray* sendiri merupakan hasil dari pengkombinasian bidang tekhnologi dan penelitian seperti mekanik, pembuatan mikro, kimia, perilaku DNA, mikrofluida, enzim, optik, dan Bioinformatika (Dufva, 2009). Data *microarray* dapat digunakan dalam mendeteksi dan mengklasifikasikan pada suatu jaringan penyakit di manusia.

Microarray menghasilkan ekspresi gen yang berisi informasi gen-gen yang kemudian dicocokkan dengan suatu penyakit (Trevino dkk, 2007).

3.1.2 *Gene Expression*

Gene Expression atau Ekspresi gen adalah proses transkripsi DNA di dalam sel menjadi RNA (Madigan dkk, 2008). DNA atau *deoxyribonucleic acid* adalah asam nukleat yang menyimpan informasi-informasi tentang genetika. Informasi pada DNA disimpan dalam bentuk kode yang terdiri atas empat basis kimia yaitu adenine (A), guanine (G), sitosin (C), dan timin (T). urutan DNA yang mengkode protein disebut dengan gen, protein merupakan pengendali sifat fisik sel misalnya warna dari mata atau rambut. Berbeda dengan DNA, RNA adalah untau tunggal yang panjangnya hanya 75-5000 nukleotida. Suatu sel mengandung beberapa jenis RNA yaitu: messenger RNA (mRNA), RNA transfer (tRNA), dan RNA ribosom (rRNA) (Bolstad, 2004).

3.2 Diabetes Melitus

Diabetes Melitus (DM) merupakan penyakit gangguan metabolik menahun yang diakibatkan oleh pankreas yang tidak dapat memproduksi cukup insulin atau tubuh tidak dapat menggunakan insulin yang diproduksi secara efektif sehingga mengakibatkan terjadinya peningkatan konsentrasi glukosa di dalam darah (hiperglikemia). Gejala yang biasa menjadi keluhan para penderita DM yaitu polydipsia, polyuria, polifagia, penurunan berat badan, dan kesemutan. DM merupakan penyakit yang disebabkan oleh adanya kekurangan insulin secara relative atau absolut. Insulin adalah hormon yang mengatur keseimbangan kadar gula darah. Defisiensi insulin sendiri dapat terjadi melalui 3 tahap, yaitu: (Fatimah, 2015)

1. Rusaknya sel-sel B pankreas karena pengaruh dari luar (virus, zat kimia dll)
2. Desensitasi atau penurunan reseptor glukosa pada kelenjar pankreas.
3. Desensitasi atau kerusakan reseptor insulin di jaringan perifer.

Terdapat 2 kategori utama dari DM yaitu diabetes tipe 1 dan diabetes tipe 2 dengan diabetes tipe 2 merupakan 90% dari keseluruhan diabetes. Selain 2 kategori diabetes tersebut terdapat juga diabetes gestasuonal yakni hiperglikemia yang

didapatkan saat kehamilan. *Impaired Glucose Tolerance* (IGT) merupakan kondisi atau masa transisi antara keadaan normal dengan diabetes, orang dengan keadaan seperti ini beresiko tinggi untuk menderita diabetes melitus tipe 2 (InfoDATIN).

Diabetes Melitus Tipe 2 (DMT2) atau biasa disebut *non-insulin-dependent* merupakan penyakit hiperglikemia yang disebabkan oleh insensivitas sel terhadap insulin. DMT2 ditandai oleh kenaikan gula darah akibat penurunan sekresi insulin oleh sel beta pancreas atau gangguan funsgsi insulin (resistensi insulin). Pada patofisioligi DMT2 terdapat beberapa keadaan yang berperan yaitu:

1. Resistensi insulin
2. Disfungsi sel B pankreas

3.3 *Pre-processing*

Pre-processing adalah proses yang dilakukan untuk menghilangkan atau tidak mengikutsertakan efek non biologis yang terdapat pada data sehingga dapat memberikan hasil yang lebih baik dalam analisis (Serin, 2011).

Proses *pre-processing* terbagi menjadi 3 tahapan yaitu *background correction, normalization,* dan *summarization*.

1. *Background correction* merupakan tahapan yang digunakan untuk menghilangkan *background noise* (data *noise* merupakan data yang berisi nilai-nilai yang salah atau anomali yang biasanya disebut juga dengan *outlier,* misalnya umur dari sampel adalah -30), menyesuaikan *cross hibdrization* yang merupakan pengikat dari DNA non spesifik yang melekat pada *array*.
2. *Normalization* merupakan tahapan untuk menghilangkan variansi *non* biologis tidak diinginkan dan dimungkinkan terdapat pada *array* sehingga data menjadi homogen.
3. *Summarization* adalah proses yang digunakan untuk mengukur gen yang terdapat pada chip, sehingga akan menghasilkan nilai ekspresi gen.

Proses *pre-processing* dapat dilakukan dengan menggunakan bantuan *packages* `AffyPLM` dan dengan menggunakan fungsi perintah `Threestep`. Perintah `Threestep` pada tahap *pre-processing* merupakan alternatif yang dapat digunakan

pada perhitungan ukuran ekspresi gen, yang dilakukan dengan menggunakan RMA.2 pada tahapan *background correction*, *quantile* pada tahapan *normalization* dan *median* pada tahapan *summarization* (Bolstad, 2004).

3.4 *Filtering*

Filtering atau proses penyaringan data adalah suatu proses yang dilakukan untuk mengurangi jumlah gen dengan memilih data yang berpengaruh sehingga dapat meningkatkan kualitas statistik analisis ekspresi gen. Dengan proses *filtering* akan mengurangi ruang fitur untuk analisis prediksi, sehingga menyebabkan suatu model yang akan dilatih lebih cepat dan variansinya cenderung sama (Dozmorov, 2016).

Proses *filtering* pada bioinformatika dapat dilakukan dengan bantuan *packages* genefilter dengan menggunakan fungsi perintah `nsFilter`. Dengan perintah `nsFilter`, proses *filter* data akan dilakukan dengan menggunakan perintah *Var.cutoff* yang digunakan untuk memfilter gen dengan nilai tertentu yang berguna untuk menghapus data dengan melihat nilai *Inter Quartil Range* (IQR) dibawah quartil. Perintah *Require.entrez* berfungsi untuk memfilter tanpa anotasi Entrez Gene ID, dimana sistem pengenal selain ID pusat, maka ID tersebut yang akan diperlukan. Perintah *Remove.DupEntrez* berfungsi untuk menghapus ID gen Entrez yang sama. Dan perintah *Feature.exclude* yang berfungsi untuk menyaring *probe control* yaitu AFFX agar tidak dimasukkan kedalam proses analisis (Gentleman dkk, 2019).

3.5 Analisis Deskriptif

Analisis deskriptif merupakan suatu analisis yang digunakan untuk menggambarkan suatu data atau memberikan gambaran umum dari suatu data. Penyajian dari analisis deskriptif dapat berupa tabel, grafik ataupun ukuran-ukuran statistik seperti mean, median, modus, presentase, variansi, dan ukuran-ukuran statistik lainnya (Purwoto, 2007).

3.6 *Machine Learning*

Awal mula *machine learning* muncul diperkenalkan oleh Arthur Samuel pada tahun 1959 melaluli jurnal yang berjudul "Some Studies in Machine Learning Using the Game of Chekers" yang dipublikasikan oleh IBM Journal of Research and Development pada juli 1959. Pada saat itu Samuel mencoba mengajari program komputer untuk dapat bermain catur tujuannya agar komputer dapat bermain catur lebih baik dari dirinya. Tujuannya tercapai pada tahun 1962 dimana program buatannya dapat mengalahkan juara catur dari negara bagian *Connecticut*.

Proses pembelajaran *machine learning* sama dengan data mining dimana dimana sistem membutuhkan data untuk menemukan pola, sehingga dapat meningkatkan pemahaman program sendiri. Program *machine learning* mendeteksi pola dalam data dan menyesuaikan tindakan program yang sesuai. Jadi, secara sederhana dapat dijelaskan bahwa *machine learning* adalah pemrograman komputer untuk mencapai kriteria/performa tertentu dengan menggunakan sekumpulan data training atau pengalaman di masa lalu (Primartha, 2018).

3.7 Klasifikasi

Klasifikasi merupakan contoh dari algoritma *machine learning* yaitu *supervised learning*. *Supervised learning* adalah algoritma yang digunakan untuk machine learning yang menggunakan data-data latihan (*training*) yang telah diberi label dengan jawaban yang benar, algoritma sendiri adalah urutan langkah-langkah yang digunakan untuk perhitungan atau untuk menyelesaikan sutu masalah yang ditulis secara berurutan (Primartha, 2018). Berdasarkan banyaknya kelas atau label pada data, klasifikasi dapat terbagi menjadi dua yaitu klasifikasi biner dan klasifikasi multiclass. Pada klasifikasi biner, tugas dapat lebih baik dipahami karena hanya dua kelas yang terlibat sedangkan klasifikasi *multiclass* melibatkan penugasan objek ke salah satu dari beberapa kelas (Budiharto, 2016).

Klasifikasi adalah proses penempatan objek tertentu dalam suatu set kategori berdasarkan pada empat komponen klasfikasi yaitu (Gorunescu, 2011):

1. Kelas

Merupakan variabel dependen berupa data kategorikal yang merepresentasikan label dari suatu objek.

2. Predictor

   Merupakan variabel independen yang merepresentasikan karakteristik dari suatu objek yang akan diklasfikasikan

3. *Training* dataset

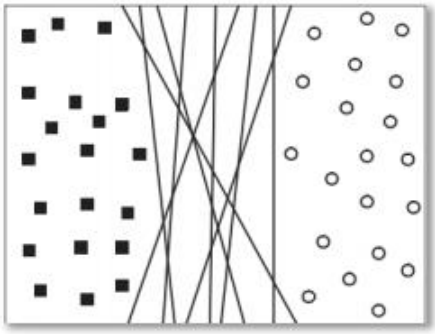
   Merupakan sekumpulan data yang terdiri atas 2 komponen sebelumnya yaitu kelas dan predictor yang digunakan untuk melatih model dalam mengklasifikasikan objek sehingga didapatkan kelas yang sesuai didasarkan pada hasil prediksi dari model.

4. *Testing* dataset

   Merupakan data baru yang digunakan untuk klasifikasi dari model yang didapatkan pada *training* dataset dan akurasi hasil klasifikasi dapat dievaluasi.

3.8 *Support Vector Machine*

Support Vector Machine (SVM) dikembangkan oleh Boser, Guyon, dan Vapnik pertama kali dipresentasikan pada tahun 1992 di *Annual Workshop on Computational Learning Theory*. Konsep dasar SVM merupakan kombinasi dari teori-teori komputasi yang telah ada pada tahun-tahun sebelumnya seperti *margin hyperplane* (Vapnik & N, 1999). Cara kerja dari SVM adalah dengan menemukan suatu bidang pemisah yang disebut dengan *hyperplane* terbaik yang membagi data menjadi 2 kelas yang berbeda. Pada perkembangannya SVM dapat diperluas untuk klasifikasi lebih dari dua kelas atau mutlti kelas. Berbeda dengan ANN, SVM akan menentukan *hyperplane* paling optimum dimana *hyperplane* dikatakan optimum apabila berada tepat ditengah-tengah kedua kelas sehingga memiliki jarak paling jauh ke data-data terluar di kedua kelas.



Gambar 3.1. Contoh *Hyperplane* Pada Klasifikasi SVM

Sumber : (Tan dkk, 2006)

Berdasarkan Gambar 3.1 diketahui bahwa terdapat banyak kemungkinan *hyperplane* yang terbentuk, dengan menggunakan SVM akan ditemukan *hyperplane* yang terbaik menggunakan *support vector* dan *margin*. *Hyeperplane* terbaik adalah *hyperplane* yang mempunyai *margin* yang paling luas dan nilai *support vector* atau data yang berada paling dekat dengan *hyperplane* akan digunakan untuk membentuk model yang digunakan untuk klasifikasi.

Suatu *hyperplane* pada SVM dapat dinotasikan dengan:

$$w.x + b = 0 \tag{3.1}$$

Dimana w adalah sebuah bobot suatu vector dan b adalah nilai bias. Konsep dari kerja SVM yaitu dengan mencari *hyperplane* terbaik untuk memisahkan dua kelas, yaitu kelas +1 (positif) dan kelas -1 (negatif). Fungsi klasifikasi pada SVM didefinisikan dengan persamaan:

$$f(x) = sign(w.x + b) \tag{3.2}$$

Apabila nilai dari $w.x+b > 0$ maka data akan diklasifikasikan kedalam kelas +1 (positif) dan apabila nilai dari $w.x+b < 0$ maka data akan diklasifikasikan kedalam kelas -1 (negatif). Seperti yang telah dijelaskan sebelumnya, *hyperplane* terbaik adalah *hyperplane* yang mempunyai *margin* paling besar atau maksimum. *Margin* maksimum dapat didapatkan dengan memaksimalkan jarak antara *hyperplane* dengan titik terdekatnya yaitu $\frac{1}{\|w\|}$. Permasalahan seperti ini biasa disebut dengan *Quadratic Programming* (QP), yaitu dengan menemukan titik minimal dari:

$$\min \tau(w) = \frac{1}{2}\|w\|^2 \tag{3.3}$$

Dengan batasan *constraint*:

$$y_i(x_i.w + b) - 1 \geq 0, i = 1,2,3,...,l \tag{3.4}$$

Permasalahan seperti ini dapat diselesaikan dengan *lagrange multiplier* dengan persamaan:

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} a_i y_i (x_i.w + b) - 1, i = 1,2,3,...,l \tag{3.5}$$

Dimana $a_i \geq 0$ adalah nilai koefisien *lagrange multipliers*. Selanjutnya meminimumkan nilai L terhadap *w* dan *b*, sehingga diperoleh:

$$\sum_{i=1}^{l} a_i y_i = 0 \tag{3.6}$$

$$w = \sum_{i=1}^{l} a_i y_i x_i \tag{3.7}$$

Selanjutnya persamaan 3.5 dimodifikasi sebagai maksimalisasi *problem* yang mengandung nilai a_i sehingga diperoleh persamaan:

$$L(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1,j=1}^{n} a_i a_j y_i y_j x_i x_j \tag{3.8}$$

$$\sum_{i=1}^{n} a_i y_i = 0\,, a_i \geq 0 \tag{3.9}$$

Dari proses yang telah dilakukan, maka akan diperoleh $a_i > 0$ yang disebut dengan *support vector* dan sisanya memiliki nilai $a_i = 0$. Fungsi keputusan yang dihasilkan hanya dipengaruhi oleh nilai *support vector*.

Pada umumnya permasalahan-permasalahan di dunia nyata sangat jarang ditemukan data terpisah secara linear, sehingga dalam menyelesaikan permasalahan nonlinear SVM menggunakan fungsi *kernel*. *Kernel* memetakan sampel data kedalam ruang dimensi yang lebih tinggi sehingga dapat menyelesaikan kasus dengan hubungan antara kelas dan atributnya tidak linear. SVM mempunyai 4 kernel yang dapat digunakan untuk menyelesaikan permasalahan linear dan nolinear diantaranya adalah:

1. Kernel Linear

$$K(x, x_k) = x_k^T x \tag{3.10}$$

2. Kernel *Polynomial*

$$K(x, x_k) = (x_k^T x + 1)^d \quad (3.11)$$

3. Kernel *Gaussian Radial Basis Function* (RBF)

$$K(x, x_k) = \exp\{-\|x - x_k\|_2^2 / \sigma^2\} \quad (3.12)$$

4. *Sigmoid*

$$K(x, x_k) = \tanh[\kappa x_k^T x + \theta] \quad (3.13)$$

Nilai akurasi dari model yang dihasilkan dengan menggunakan SVM sangat tergantung pada fungsi *kernel* dan nilai parameter yang digunakan. Parameter yang digunakan pada algoritma SVM adalah parameter *Cost* (C) dan *Gamma* (γ). Semakin besar nilai C akan menghasilkan *penalty* yang besar pula terhadap klasifiikasinya. Pada fungsi *kernel* RBF parameter *gamma* (γ) digunakan untuk metransformasikan data *train* ke ruang fitur yang kemudian dioptimasi menggunakan metode *Lagrange Multipliers* sehingga menghasilkan nilai *α* yang digunakan untuk menentukan *support vector* dan memperkirakan koefisien *w* (bobot) ataupun *b* (bias) pada model klasifikasi (Handayani dkk, 2017).

3.8.1 *Soft Margin*

Secara umum metode SVM memisahkan suatu kelas terpisah secara sempurna dengan *hyperplane*, namun pada kenyataannya suatu kelas tidak selalu terpisah secara sempurna sehingga dapat menyebabkan *constraint* pada persamaan 3.4 tidak terpenuhi. Untuk mengatasi permasalahan tersebut, SVM dapat dimodifikasi dengan menggunakan teknik *soft margin*, yaitu dengan menambahkan variabel *slack* ξ ($\xi_i > 0$) sehingga didapatkan persamaan baru sebagai berikut:

$$y_i(x_i.w + b) \geq 1 - \xi i \quad (3.14)$$

Dengan memasukkan variable *slack*, maka persamaan 3.4 dapat diubaha menjadi persamaan:

$$min_w = \tau(w) = \frac{1}{2\|w\|^2} + C \sum_{i=1}^{l} \varepsilon_i \quad (3.15)$$

Parameter C digunakan untuk mengontrol *trade off* antara *margin* dan *error* pada proses klasifikasi ξ. Nilai parameter C ditentukan dari percobaan nilai yang dievaluasi nilai dampaknya terhadap nilai akurasi yang didapatkan dari model klasifikasi SVM, salah satunya dengan cara *cross validation* (Nugroho dkk, 2003).

3.8.2 *Multi Class* SVM

Pada prinsipnya *multi class* SVM dapat diimplementasikan dengan cara menggabungkan beberapa SVM dengan dua kelas (biner). Ada beberapa metode yang dapat digunakan untuk SVM dengan lebih dari dua kelas atau *multi class*, salah satunya adalah metode *One-Against-One*. Dengan menggunakan metode ini, terdapat beberapa model SVM biner yang harus dibangun untuk membandingkan satu kelas dengan kelas lainnya. Misalkan akan dilakukan klasifikasi data kedalam *k* kelas maka harus dibangun model sebanyak $\frac{k(k-1)}{2}$ model SVM biner. Misal untuk klasifikasi data untuk 4 kelas maka model yang harus dibangun adalah $\frac{3(3-1)}{2}$ = 3 model SVM biner. SVM biner pertama akan dibangun dengan menggunakan data *train* dari kelas pertama dan kelas kedua untuk mengklasifikasikan data kedalam kelas pertama atau kelas kedua. Model SVM biner kedua akan dibangun dengan menggunakan data *train* dari kelas pertama dan kelas ketiga untuk mengklasifikasikan data kedalam kelas pertama atau kelas ketiga. SVM biner ketiga dibangun dengan menggunakan data *train* dari kelas kedua dan kelas ketiga untuk mengklasifikasikan data kedalam kelas kedua atau ketiga. Untuk mendapatkan kelas keputusan, akan dilakukan *voting* sehingga kelas yang paling banyak menang adalah kelas keputusan (Suyanto, 2018).

3.9 Jaringan Syaraf Tiruan

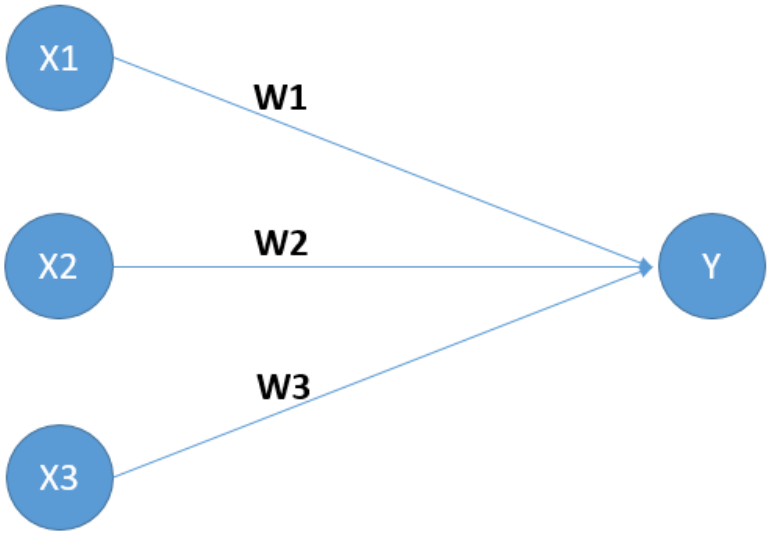
Jaringan Syaraf Tiruan (JST) merupakan sistem pemprosesan komputasi yang terinspirasi dari sistem sel syaraf biologi pada otak manusia. Menurut Sekarwati (2005), JST merupakan sistem komputasi yang didasarkan atas pemodelan sistem bilogis (neuron) melalui pendekatan dari sifat-sifat biologis (*biological computation*). Secara umum menurut (Budiharto, 2016), JST terdiri atas:

1. *Input layer*, berisi beberapa neuron *input* yang berfungsi seperti dendrit.
2. *Hidden layer*, merupakan tempat terjadinya proses utama dalam JST.
3. *Output layer*, berisi beberapa *output* neuron yang berfungsi seperti akson.
4. Fungsi aktivasi, berfungsi seperti sinapsis.

Hasil dari JST ditentukan atas 3 hal, yaitu:

1. Pola hubungan antar neuron (arsitektur jaringan)
2. Metode yang digunakan untuk menentukan bobot penghubung (algoritma *training/learning*)
3. Fungsi aktivasi

Model-model JST ditentukan oleh arsitektur jaringan serta algoritma pelatihan yang digunakan. Arsitektur jaringan menjelaskan arah perjalanan dari data di dalam jaringan, sedangkat algoritma pembelajaran menjelaskan bagaimana bobot koneksi diubah agar pasangan *input-output* yang diinginkan dapat tercapai.



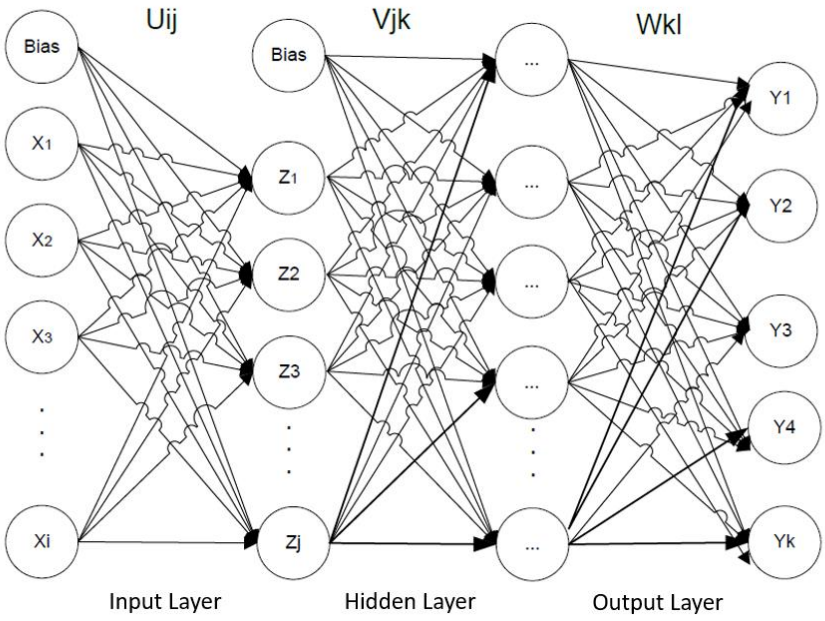
Gambar 3.2. Sel Jaringan Syaraf Tiruan

Pada gambar diatas diketahui bahwa sebuah sel syaraf tiruan sebagai elemen penghitung. Simpul Y menerima masukan dari 3 neuron x1, x2, dan x3 dengan bobot dari masing-masing masukan w1, w2, dan w3. Fungsi aktivasi adalah net (jejaring) masukan yaitu kombinasi linear masukan dan bobotnya, sehingga *net=x1w1 + x2w2 + x3w3*. Besarnya sinyal yang diterima oleh Y mengikuti nilai fungsi aktivasi *y=f(net)*.

3.9.1 *Multilayer Perceptron*

Multilayer Perceptron atau MLP merupakan arsitektur pada JST yang paling banyak digunakan untuk permasalahan-permasalahan klasifikasi.

Sesuai dengan namanya MLP memiliki lapisan ganda yang terdiri atas tiga lapisan yaitu *input layer, hidden layer,* dan *output layer*.



Gambar 3.3. Arsitektur *Multilayer Perceptron* (S.E Fahlman, 1987)

Dari Gambar 3.2 koneksi antar *layer* pada arsitektur MLP dihubungkan dengan bobot U_{ij} untuk koneksi antara *input layer* (x_i) menuju *hidden layer* (Z_j), bobot V_{jk} untuk koneksi antara *hidden layer* (Z_j) menuju *hidden layer* (Z_j) berikutnya, dan bobot W_{kl} untuk koneksi antara *hidden layer* (Z_j) menuju *output layer* (Y_k). Proses pembelajaran pada MLP bertujuan untuk menemukan bobot-bobot sinaptik yang paling optimum untuk mengklasifikasikan himpunan data latih dan data validasi. Algoritma pembelajaran yang banyak digunakan pada MLP adalah *Backpropagation*.

3.9.2 Algoritma *Backpropagation*

Algoritma *backpropagation* merupakan algoritma pelatihan yang terwasi dengan menggunakan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang terdapat ada *hidden layer* (Haryati dkk, 2016). Algoritma ini melakukan pelatihan pada MLP melalui dua tahap yaitu perhitungan maju *(Feedforward)* untuk menghitung galat *(loss function)* antara prediksi target dan nilai aktual. Selanjutnya perhitungan

mundur *(Backpropagation)* yang mempropragasikan balik galat untuk memprediksi bobot-bobot sinaptik pada semua neuron yang ada. Pada dasarnya proses pembelajaran pada MLP adalah menemukan bobot-bobot sinaptik yang menghasilkan *hyperplane* dengan kemiringan dan posisi yang tepat sehingga mampu mengklasifikasikan pola-pola pada himpunan data latih dengan kesalahan sekecil mungkin (Suyanto, 2018).

Berikut ini adalah tahapan pada *Feedforward* dan *Backpropagation* pada Algoritma *Backpropagation:*

Tahap *Feedforward*

1. Setiap unit lapisan *input* (x_i, i=1,2,3, …, n) menerima sinyal *input* (x_i) yang kemudian diteruskan ke unit-unit pada lapisan tersembunyi (*hidden layer*).
2. Setiap unit pada lapisan *hidden* (z_j, j=1,2,3, …, p) menjumlahkan sinyal *input* yang telah berbobot dengan persamaan:

$$Z\_in_{jk} = V_{oj} + \sum_{i=0}^{n} X_i V_{ij} \tag{3.16}$$

Selanjutnya menghitung sinyal *hidden layer* dengan menggunakan fungsi aktivasi pada persamaan:

$$Z_j = f(Z\_in_j) \tag{3.17}$$

3. Setiap unit pada lapisan *output* (Y_k, k=1,2,3, …, m) dikalikan dengan bobot dan dijumlahkan dengan biasnya menggunakan persamaan:

$$Y\_in_k = W_{oj} + \sum_{j=1}^{p} Z_j W_{jk} \tag{3.18}$$

Selanjutnya menghitung sinyal *output* dengan menggunakan fungsi aktivasi pada persamaan berikut:

$$Y_k = f(Y\_in_k) \tag{3.19}$$

Tahap *Backpropagation*

4. Setiap unit pada lapisan *output* (Y_k, k=1,2,3, …, m) menerima pola target (t_k) yang sesuai dengan pola *input* saat pelatihan, kemudian

informasi *error* pada lapisan *output* (δ_k) didefinisikan dengan persamaan sebagai berikut:

$$\delta_k = (t_k - y_k) f'(y_k)(1 - y_k) \tag{3.20}$$

Kumidan hitung koreksi bobot yang selanjutnya digunakan untuk memperbaiki nilai W_{jk} dengan persamaan sebagai berikut:

$$\Delta W_{jk} = \alpha \delta_k Z_j \tag{3.21}$$

Selanjutnya menghitung koreksi bias untuk memperbaiki nilai W_{0k} dengan persamaan sebagai berikut:

$$\Delta W_{0k} = \alpha \delta_k \tag{3.22}$$

5. Setiap unit pada lapisan *hidden* (j, j=1,2,3, …, p) menjumlahkan delta *input* dari unit-unit yang berada pada lapisan sebelumnya denga persamaan:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \tag{3.23}$$

Selanjutnya hasil dikalikan dengan turunan dari fungsi aktivasi yang digunakan pada jaringan untuk menghitung informasi kesalahan *error* δ_j dengan persamaan:

$$\delta_j = \delta\_in_j f'(Z\_in_j) \tag{3.24}$$

Faktor δ_j digunakan untuk menghitung korekai *error* Δv_{ij} yang selanjutnya digunakan untuk memperbarui v_{ij} menggunakan persamaan:

$$\Delta v_{ij} = \alpha \delta_j x_i \tag{3.25}$$

Selanjutnya menghitung koreksi bias Δv_{0j} yang digunakan untuk memperbarui v_{0j} dengan persamaan:

$$\Delta v_{0j} = \alpha \delta_j \tag{3.26}$$

6. Setiap unit *output* (k=1,2,3, …, m) digunakan untuk memperbaiki bias dan bobotnya memperbarui bobotnya (j=0,1,2,3, …, p) sehingga menghasilkan bobot dan bias baru dengan persamaan:

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \tag{3.27}$$

Demikian juga untuk bobot pada unit *input* ke unit *hidden*, diperbaiki dengan menggunakan bobot yang telah dihitung sebelumnya menggunakan persamaan:

$$v_{jk}(baru) = v_{jk}(lama) + \Delta v_{jk} \tag{3.28}$$

7. Selanjutnya adalah periksa kondisi berhenti. Jika kondisi berhenti telah terpenuhi maka pelatihan jaringan dapat dihentikan.

Terdapat 2 cara untuk menentukan kondisi berhenti, salah satunya adalah dengan membatasi banyaknya *epoch* yang digunakan.

3.9.3 Fungsi Aktivasi

Fungsi aktivasi adalah suatu fungsi yang digunakan untuk mentransformasikan suatu *input* menjadi *output*, sehingga dapat dikatakan fungsi aktivasi adalah operasi matematika yang digunakan pada perhitungan *output*. Salah satu fungsi aktivasi yang dapat digunakan pada arsitektur MLP adalah fungsi aktivasi *Rectified Linear Unit* (ReLu). Fungsi aktivasi ini menjadi fungsi yang sangat populer karena fungsi ini mudah untuk dioptimalkan. Selain itu fungsi aktivasi ini tidak mudah jenuh karena tidak *asymptotic* (Jin dkk, 2015). Fungsi ReLU didefinisikan sebagai berikut:

$$h(x_i) = \max(0, x_i) \tag{3.29}$$

Dimana x_i merupakan *input* dan $h(x_i)$ adalah *output*.

3.10 *Boosting*

Boosting merupakan salah satu teknik *ensemble* yang biasa digunakan dalam proses-proses analisis klasifikasi ataupun prediksi. Teknik *ensemble* merupakan metode algoritma pembelajaran yang dibangun berdasarkan beberapa model klasifikasi ataupun prediksi yang selanjutnya digunakan untuk mengklasifikasikan data baru berdasarkan bobot prediksi yang dihasilkan sebelumnya (Dietterich, 2000). Konsep *ensemble* pada *boosting* yaitu dengan melatih model-model secara sekuensial dan kemudaian model-model tersebut digabungkan untuk melakukan prediksi, model baru yang dihasilkan belajar berdasarkan kesalahan dari model sebelumnya (Zhou, 2012).

Langkah-langkah pada teknik *gradient boosting*, dimulai dari memuat model ke dalam data yang didefinisikan dengan persamaan:

$$F_1(x) = y \tag{3.30}$$

Selanjutnya, model digunakan untuk menghitung *residual* pada proses sebelumnya, seperti pada persamaan berikut:

$$h_1(x) = y - F_1(x) \tag{3.31}$$

Selanjutnya buat model baru dengan persamaan:

$$F_2(x) = F_1(x) + h_1(x) \tag{3.32}$$

Dari proses yang telah dilakukan didapatkan model final yang merupakan gabungan dari kumpulan model-model sebanuak n iterasi sampai dengan dihasilkan nilai *error* terkecil dari *residual.* Model final didefiniskan dengan persamaan:

$$\hat{F}(x) = F_1(x) \rightarrow F_2(x) = F_1(x) + h_1(x) \ldots \rightarrow F_M(x) = F_{M-1}(x) + h_{M-1}(x) \tag{3.33}$$

Model akhir dari *boosting* didefinisikan dengan persamaan sebagai berikut:

$$f(x) = \sum_{m-1}^{M} f_m(x) \tag{3.34}$$

Atau dapat didefinisikan dengan persamaan:

$$f(x) = \gamma_0 + \sum_{m=1}^{M} \gamma_m h_m(x) \tag{3.35}$$

Dimana $f_x = \gamma_0$ dan $f_m(x) = \gamma_m h_m(x)$ untuk m = 1,2,3, …, M dengan nilai $h_m(x) \in \{-1,1\}$. $\gamma_m(x)$ merupakan pengklasifikasian lemah sedangkan γ_m adalah bobot pada tiap pengklasifikasi (Syahrani, 2019).

3.10.1 Xtreme Gradient Boosting

Xtreme Gradient Boosting (XGBoost) merupakan metode yang mengkombinasikan *boosting* dengan *gradient boosting*. Metode ini pertama kali diperkenalkan oleh Friedman, pada penelitiannya digunakan hubungan antara *boosting* dan optimasi untuk membuat *Gradient Boosting Machine* (GBM). Model dibangun dengan menggunakan metode *boosting*, yaitu dengan membuat model baru untuk memprediksi *error* dari model sebelumnya. Algoritma seperti ini dinamakan *gradient boosting* karena digunakan *gradient descent* untuk memperkecil *error* saat membentuk model baru. Tujuan akhir dari proses ini adalah

didapatkannya fungsi paling mendekati $\hat{F}(x)$ terhadap fungsi-fungsi pembangunnya *f(x)* dengan meminimumkan nilai *loss function* L(*y, f(x)* didefinisikan dengan persamaan:

$$\hat{F} = \arg\min_f E_{x,y}[L(y, f(x))] \tag{3.36}$$

Dalam proses *training* tiap iterasi untuk meminimalkan nilai *loss function* berdasarkan fungsi awalnya $F_0(x)$. Secara umum algoritma *gradient boosting* mempunyai persamaan sebagai berikut:

$$\{\gamma_m, h_m\} = \arg\min \sum_{m=1}^{M} L(y_i, f^{(m-1)}(x_i) + \gamma_m h_m(x_i)) \tag{3.37}$$

XGBoost merupakan versi dari *Gradient Boosting Method* (GBM) yang lebih efisien dan *scalable* karena mampu menyelesaikan berbagai fungsi seperti regresi, klasifikasi, dan ranking. XGBoost pertama kali dikenalkan pada *Higgs Boson Competition*, dimana pada kompetisi ini metode XGBoost menjadi metode yang paling banyak digunakan oleh sebagian besar tim. Selain kompetisi tersebut, metode XGBoost juga menjadi metode yang banyak digunakan pada kompetisi *machine learning* yang diselenggarakan oleh Kaggle pada tahun 2015. XGBoost merupakan *tree ensembles algorithm* yang terdiri atas beberapa *classification and regression trees* (CART). Algoritma XGBoost melakukan optimasi 10 kali lebih cepat dibandingkan dengan implementasi dari GBM lainnya (Chen & Guestrin, 2016).

Nilai akurasi dari hasil klasifikasi dengan menggunakan XGBoost tergantung pada paremeter-parameter yang digunakan. Berikut ini adalah parameter-paremeter yang dapat disesuaikan nilainya pada XGBoost agar mendapatkan nilai akurasi yang lebih baik:

Tabel 3.1. Parameter Pada *Xtreme Gradient Boosting*

Parameter	**Keterangan**
Eta	*Learning rate* pada proses pelatihan
Gamma	Parameter *penalty* pada *regularization*
Max_depth	Tingkat kedalaman suatu pohon, semakin dalam pohon maka akan semakin kompleks

Min_child_weight	Nilai minimal bobot yang dibutuhkan *child node*
Subsample	Jumlah sampel yang digunakan untuk proses pelatihan. Misal 0,5 berarti menggunakan setengah dari data secara acak dalam membuat pohon baru
Colsample_bytree	Jumlah sampel kolom untuk membuat *tree* baru

3.11 *Confussion Matrix*

Confussion matrix merupakan suatu alat ukur berbentuk *matrix* yang digunakan untuk mengetahui nilai ketepatan klasifikasi terhadap kelas dengan algoritma yang digunakan. *Confussion matrix* dapat digunakan untuk mempermudah pencarian ukuran-ukuran performa hasil klasifikasi. Tabel 3.2 memberikan contoh *confussion matrix* dengan 2 klasifikasi benar dan salah.

Tabel 3.2. *Confussion 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)

Hasil dari pengukuran pada *confussion matrix* dapat digunakan untuk menghitung nilai akurasi. Akurasi merupakan suatu nilai yang digunakan untuk mengevaluasi hasil dari model klasifikasi. Akurasi didapatkan dari hasil pembagian antara jumlah prediksi model yang benar dengan jumlah total yang diprediksi, perhitungannya didefinisikan sebagai berikut (Klasifikasi: Akurasi, 2018):

$$Akurasi = \frac{Jumlah\ prediksi\ benar}{Jumlah\ total\ prediksi} \tag{3.38}$$

Hasil dari penggunaan *confussion matrix* selain dapat digunakan untuk menghitung akurasi, dapat juga digunakan untuk menghitung evaluasi model lainnya seperti *precision*, *recall/sensitivity*, *specificity*, FPR dan *AUC*.

$$Precision = \frac{TP}{TP+FP}\ x\ 100\% \tag{3.39}$$

$$Recall/Sensitivity = \frac{TP}{TP+FN}\ x\ 100\% \tag{3.40}$$

$$Specificity = \frac{TN}{TN+FP}\ x\ 100\% \tag{3.41}$$

$$FPR = 1 - Specificity \tag{3.42}$$

$$AUC = \frac{1+sensitivity-FPR}{2} \tag{3.43}$$

Nilai *AUC* digunakan untuk mengukur kinerja diskriminatif dengan memperkirakan probabilitas *output* dari sampel yang dipilih secara acak. Nilai *AUC* berada pada rentang 0-1, semakin besar nilai *AUC* maka semakin kuat klasifikasi yang digunakan, dengan keterangan pengukuran klasifikasi seperti pada Tabel 3.3.

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)

BAB IV
METODOLOGI PENELITIAN

4.1. Sumber dan Jenis Data

Data yang peneliti gunakan adalah data sekunder yang peneliti dapatkan dari website NCBI dengan url https://www.ncbi.nlm.nih.gov/ yang diakses pada tanggal 12 Februari 2020. Objek penelitian adalah data *microarray* yang telah dimuat server berisi tentang informasi bioteknologi seperti DNA, protein, senyawa aktif dan taksonomi. Data yang digunakan adalah data *microarray* dengan series GSE18732_RAW dengan banyaknya sampel sebanyak 118.

4.2. Metode Analisis Data

Metode analisis data yang digunakan pada penelitian ini adalah klasifikasi menggunakan 3 metode klasifikasi yaitu *Support Vector Machine* (SVM), *Multilayer Perceptron* (MLP) dan *Xtreme Gradient Boosting* (XGBoost). Analisis ini digunakan untuk mengklasifikasikan gen hasil ekspresi gen *skeletal muscle* normal (NGT), *Impaired Glucose Tolerance* (IGT) dan Diabetes melitus tipe 2. *Software* yang digunakan untuk penelitian ini adalah *software* R studio dengan versi 3.6.2.

4.3. Variabel Penelitian

Adapun macam-macam variabel yang digunakan pada penelitian ini adalah sebagai berikut:

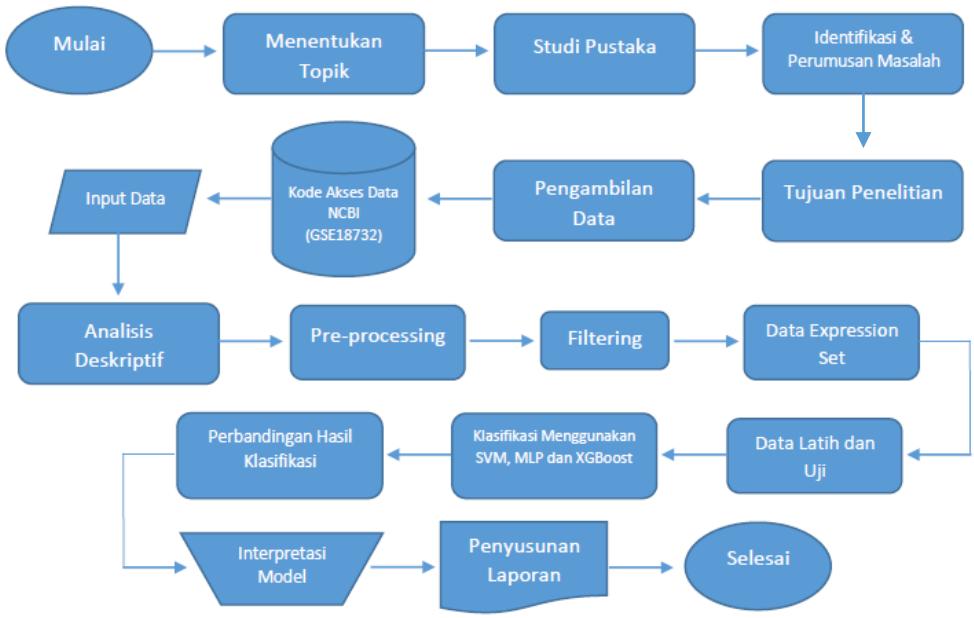
Tabel 4.1. Definisi Variabel Penelitian

Nama Variabel	Definisi Operasional Variabel
Gen	Gen yang digunakan pada penelitian ini adalah hasil dari ekspresi gen pada *skeletal muscle* normal (NGT), IGT dan manusia yang menderita penyakit diabetes melitus tipe 2 (DMT2).

Tissue **(Jaringan)**	Adalah variabel yang menunjukkan sekumpulan sel yang menyusun setiap tubuh manusia. variabel ini memiliki 3 jaringan yang berbeda pada data yaitu jaringan *skeletal mescle* normal (NGT), IGT dan DMT2.

4.4. Langkah-langkah Penelitian

Berikut ini adalah langkah-langkah atau tahapan yang dilakukan pada penelitian yang disajikan dalam bentuk *flowchart* berikut:



Gambar 4.1. Diagram Alir Penelitian

Keterangan:

1. Tahap pertama yang dilakukan peneliti setelah memulai penelitian adalah menentukan topik penelitian yang akan dilakukan kemudian mencari studi pustaka atau referensi tentang penelitin-penelitian yang pernah dilakukan sebelumnya sesuai dengan topik yang telah ada dan kemudian mengidentifikasi rumusan masalah penelitian dan tujuan dari penelitian yang akan dilakukan
2. Setelah tahapan pertama dilakukan selanjutnya adalah tahapan pencarian data yang sesuai dengan topik penelitian yang akan dilakukan dimana pada penelitian ini peneliti mencari data pada website NCBI.

3. Setelah mendapatkan data yang sesuai dengan topik penelitian, selanjutnya adalah tahapan penginputan data pada *software* yang digunakan oleh peneliti untuk melakukan analisis.
4. Setelah melakukan penginputan data, selanjutnya adalah melakukan analisis deskriptif yaitu melihat gambaran data yang telah didapatkan. Pada tahapan ini peneliti menggunakan bantuan *package* `GEOquery` dan `affy` untuk melihat informasi apa saja yang ada pada data tersebut.
5. Setelah mengetahui informasi apa saja yang ada pada data yang dimiliki, selanjutnya adalah tahapan *pre-processing* dan filtering. Tahapan *pre-processing* ini akan dilakukan untuk pemilihan variabel pada data dan menyeleksi fitur yang akan digunakan untuk pengklasifikasian dan *filtering* meliputi standar deviasi dan nilai rata-rata (*mean*).
6. Setelah melakukan tahap *pre-processing* dan *filtering,* selanjutnya data hasil proses tersebut akan dimasukkan kedalam data *expression set* yang baru, sehingga akan terdapat data baru berbentuk *frame* yang siap untuk dianalisis.
7. Proses selanjutnya adalah membagi data yang baru kedalam 2 data yaitu data latih *(train)* dan data uji *(test)* dengan jumlah data latih sebesar 80% dan data uji 20% dari total keseluruhan data.
8. Selanjutnya adalah menganalisis data latih menggunakan metode klasifikasi SVM, MLP dan XGBoost dan model yang didapatkan dari data latih digunakan untuk memprediksi data uji sehingga akan didapatkan nilai akurasi prediksi dari masing-masing metode yang digunakan untuk mengklasifikasikan data.
9. Setelah mendapatkan nilai akurasi dari masing-masing metode, selanjutnya adalah membandingkan metode mana yang mempunyai nilai akurasi terbaik dalam memprediksi atau mengklasifikasikan data uji dan kemudian dilakukan interpretasi dari model dengan metode terbaik yang telah didapatkan.
10. Hasil dari analisis yang telah dilakukan selanjutnya di proses kedalam laporan akhir penelitian.

Tabel 4.2. Keterangan *Flowchart*

Simbol	Keterangan
	Terminator simbol (Permulaan dari suatu kegiatan atau Pengakhiran)
	Simbol yang menyatakan proses *input* dan *output*
	*Magnetik Disk* (Hasil *Input* atau *Output* berupa Disk Magnetik)
	Proses Pengolahan Data atau Perhitungan Data
	Manual *Operation* (Pekerjaan atau Operasi Secara Manual)
	*Document* (Mencetak Hasil)
	Simbol Arus (Menghubungkan antara simbol yang satu dan lainnya)

BAB V
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil-hasil yang telah didapatkan dari hasil komputasi klasifikasi menggunakan metode *support vector machine*, *multilayer perceptron* dan *xtreme gradient boosting*. Pembahasan meliputi analisis deskriptif, *pre-processing*, *filtering*, klasifikasi menggunakan metode svm, mlp dan xgboost serta evaluasi dari ketiga metode klasifikasi yang digunakan.

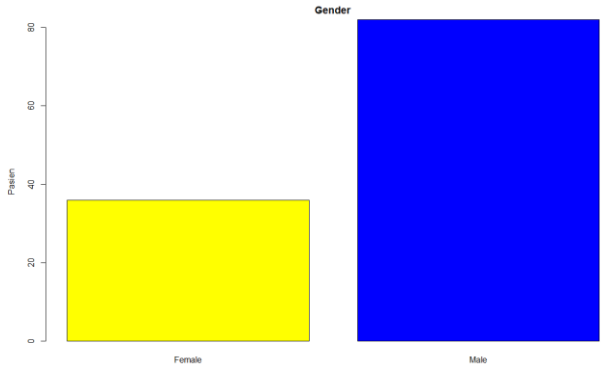
5.1 Analisis Deskriptif

Pada penelitian ini data yang digunakan adalah data ekspresi gen pada *skeletal muscle* atau otot rangka manusia yang didapatkan pada website NCBI dengan series GSE18732 pada *platform* GPL9486.

Tabel 5.1. *Dataset* GSE18732

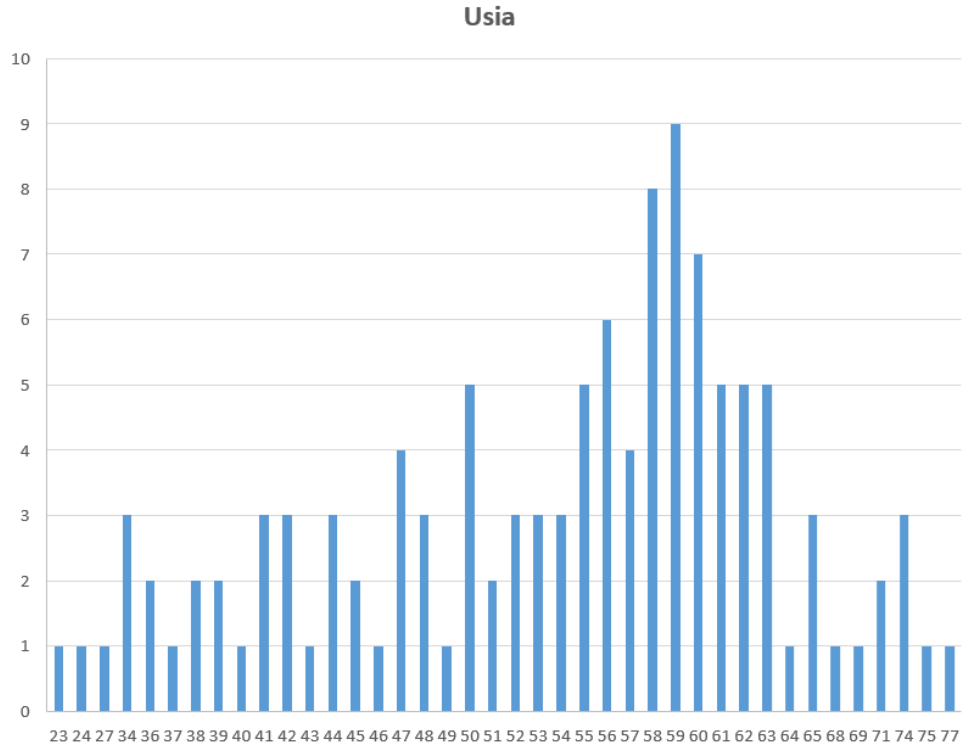
Informasi	Keterangan
Jumlah Gen	54675
Jumlah Sampel	118
Jumlah Kelas	3

Berdasarkan **Tabel 5.1** diketahui bahwa data terdiri atas 118 sampel dengan jumlah gen sebanyak 54675 gen dan terdiri atas 3 kelas yaitu ekspresi gen *skeletal muscle* Normal (NGT), IGT dan Diabetes Melitus tipe 2. Sampel pada data tersebut diambil di Faculty of Health Sciences and Sports, University of Stirling, kota Stirling, United Kingdom. Jumlah sampel normal sebanyak 47 orang, 26 sampel IGT dan 45 sisanya adalah penderita DMT2.



Gambar 5.1. *Barplot* Jenis Kelamin

Berdasarkan **Gambar 5.1** diketahui bahwa dari 118 sampel sebanyak 82 sampel berjenis kelamin laki-laki dan sisanya sebanyak 36 sampel berjenis kelamin perempuan. Dari 82 sampel laki-laki terdapat sebanyak 33 sampel dengan kadar gula darah tergolong kedalam diabetes melitus tipe 2, 20 sampel kadar gula darah *impaired glucose tolerance*, dan 29 sampel dengan kadar gula darah normal. Sedangkan dari 36 sampel perempuan terdapat 12 sampel dengan kadar gula darah tergolong kedalam diabetes melitus tipe 2, 6 sampel kadar gula darah *impaired glucose tolerance*, dan 18 sisanya dengan kadar gula darah normal.

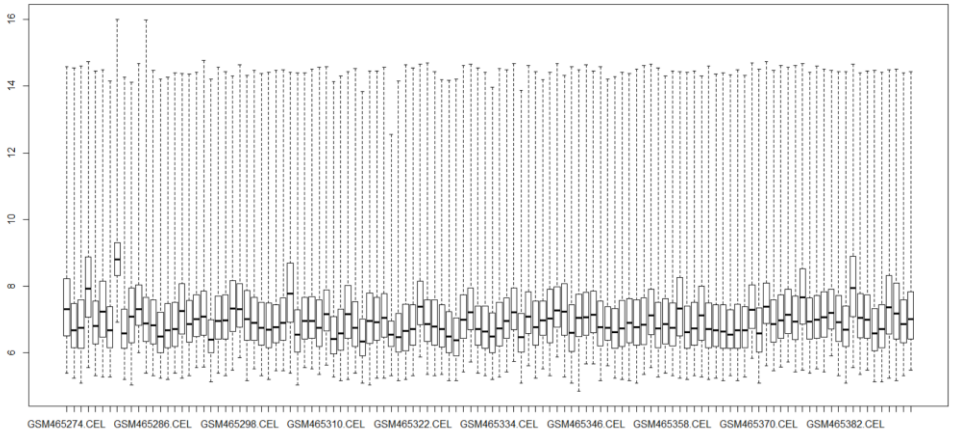


Gambar 5.2. *Barplot* Usia Sampel

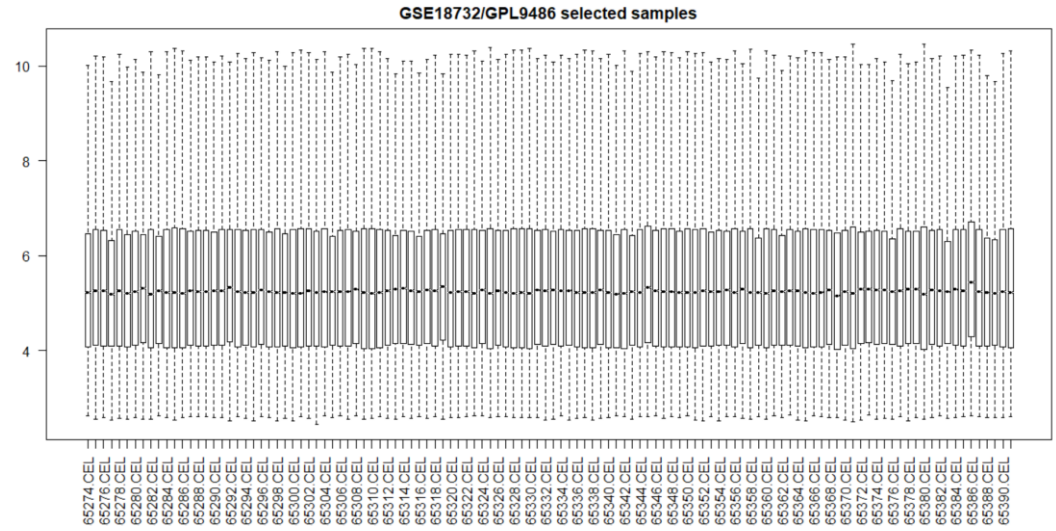
Berdasarkan **Gambar 5.2** diketahui bahwa sampel dari data ekspresi gen GSE18732 berada pada rentang 23 sampai umur 77 tahun. Sampel merupakan orang-orang dewasa dikarenakan diabetes melitus tipe 2 memang lebih banyak menyerang usia 20 keatas sedangkan usia 20 tahun kebawah biasanya lebih banyak menderita diabetes melitus tipe 1 dibandingkan diabetes melitus tipe 2.

5.2 *Pre-Processing* Data

Pada tahapan *pre-processing* ini hasil dapat dilihat pada **Gambar 5.3** dan **Gambar 5.4** dalam bentuk *boxplot*. *Boxplot* adalah salah satu cara dalam statistik untuk menggambarkan secara grafik data. Terdapat beberapa ukuran yang dapat diketahui dari *boxplot* diantaranya adalah kuartil pertama (Q1), median/kuartil tengah (Q2), kuartil ketiga (Q3) dan *interquartile range* (IQR). Nilai IQR menentukan panjang dari *box* pada *boxplot*, semakin tinggi atau lebar bidang IQR menandakan semakin menyebar data yang dimiliki. Berikut adalah *boxplot* sebelum dan setelah dilakukan *pre-processing*.



Gambar 5.3. *Boxplot* Sebelum *Pre-processing*



Gambar 5.4. *Boxplot* Setelah *Pre-processing*

Dari **Gambar 5.3** dan **Gambar 5.4** terlihat perbedaan yang signifikan pada *boxplot* sebelum dan sesudah *pre-processing* dimana *boxplot* data setelah *pre-processing* lebih baik daripada sebelum *pre-processing*. Tapahan *pre-processing* ini digunakan untuk menghapus nilai *non biologis*, sehingga yang tertera pada *boxplot* setelah dilakukan *pre-processing* hanya data-data yang bersifat *biologis* saja, sehingga *boxplot* tersebut mempunyai rata-rata yang sama.

5.3 *Filtering* Data

Pada tahapan *filtering* akan dilakukan tahapan penentuan atau pemilihan gen yang akan siap untuk dianalisis. Tahapan *filtering* terbagi menjadi dua tahapan yaitu tahapan *filtering* dan tahapan *feature selection*. Tahapan *filtering* data akan menggunakan fungsi `nsFilter` *(Non Speciefied Filtering)*, dengan fungsi ini akan digunakan untuk menghapus atau mengeluarkan variabel yang mempunyai nilai *interquartile range* (IQR) tinggi, sehingga dari tahapan pertama *filtering* ini dihasilkan data dengan dimensi 10064x118.

Data hasil *filtering* selanjutnya akan dilakukan tahapan *feature selection* dengan menggunakan fungsi multtest. Multtest adalah fungsi yang berasal dari t-test yang digunakan untuk membandingkan dua sampel yang diambil dari populasi yang mempunyai variansi yang sama. Karena data yang dianalisis terdiri atas tiga sampel kelas maka pada penelitian ini akan digunakan fungsi *F-test*. Pada tahapan multtest klasifikasi data pada penelitian diasumsikan berdistribusi normal yang

dapat dilihat dengan menggunakan perintah *qqnorm*. Hasil dari *feature selection* ini terbentuk data dengan dimensi 118x167 yang dapat dilihat pada tabel berikut ini.

Tabel 5.2. Hasil *Filtering* dan *Feature Selection*

NsFilter	
Sampel	Gen
118	10064
Multtest	
Sampel	Gen
118	80

5.4 Pembagian Data *Training* dan Data *Testing*

Dari tahap *filtering* didapatkan data yang siap dianalisis berdimensi 118x81 selanjutnya data tersebut akan dibagi menjadi data *training* (latih) dan data *testing* (uji) untuk selanjutnya dilakukan klasifikasi. Pembagian data *training* dan *testing* dibagi dengan rasio 80% untuk data *training* dan 20% untuk data *testing* dari total data yang siap dianalisis untuk masing-masing kelas.

Tabel 5.3. Pembagian Data *Training* dan Data *Testing*

Pembagian Data	
Data *Training*	Data *Testing*
95	23

5.5 Support Vector Machine *(SVM)*

Pada analisis klasifikasi menggunakan metode SVM terdapat empat kernel yang dapat digunakan untuk mengklasifikasikan data yaitu kernel *linear*, *Polynomial*, *Sigmoid* dan RBF. Masing-masing kernel mempunyai parameter-parameter dalam pembuatan modelnya, pada penelitian ini akan dilakuka Berikut ini adalah hasil nilai akurasi dari masing-masing kernel pada percobaan klasifikasi menggunakan data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan diabetes melitus tipe 2.

Tabel 5.4. Perbandingan Nilai Akurasi Pada Kernel

Kernel	**Akurasi**

Linear	91,30%
Polynomial	69,56%
Sigmoid	43,47%
RBF	39,13%

Berdasarkan **Tabel 5.4** diketahui bahwa model SVM dengan kernel *linear* merupakan model dengan nilai akurasi yang terbaik dibandingkan model SVM dengan kernel-kernel lain yaitu dengan nilai akurasi sebesar 91,30%.

5.6 Merancang Arsitektur *Multilayer Perceptron* (MLP)

Pada proses klasifikasi data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan Diabetes Melitus Tipe 2 menggunakan JST dengan arsitektur MLP data *input* terdiri atas 80 gen hasil dari proses *pre-processing* dan *filtering* dengan 3 luaran (*output*). Data yang digunakan untuk proses *train* atau latih dengan menggunakan 95 sampel yang kemudian akan di uji dengan 23 sampel. Pembuatan arsitektur MLP digunakan beberapa perbandingan jumlah *hidden layer*, jumlah node pada masing-masing *hidden layer*, *optimizer* dan jumlah *epoch* untuk mendapatkan hasil klasifikasi dengan nilai akurasi terbaik.

5.6.1. Membandingkan jumlah *hidden layer* dan *node*

Untuk mendapatkan arsitektur MLP, dilakukan dua percobaan untuk menentukan jumlah *hidden layer* dan jumlah *hidden node* dari setiap *layer*-nya. Percobaan pertama dilakukan dengan membandingkan jumlah *hidden node* dengan jumlah kelipatan dari angka 10 yaitu 20, 40, 60, 80, dan 100 untuk 1 *hidden layer*. Sedangkan percobaan kedua dengan 2 *hidden layer* dan mevariasikan jumlah *hidden node* pada masing-masing *hidden layer*.

Tabel 5.5. Perbandingan Jumlah *Hidden Layer* dan *Hidden Note*

Jumlah *Hidden Layer*	**Jumlah *Hidden Node***	***Loss***	**Akurasi**
1	20	0,7197	0,6086
	40	0,7233	0,6956
	60	0,7401	0,5652
	80	0,7063	0,6521
	100	0,5269	0,7391

2	100 & 50	0,8218	0,5652
	80 & 40	0,7702	0,6086
	60 & 30	0,6249	0,5652
	40 & 20	0,5861	0,7391

Dari hasil perbandingan jumlah *hidden layer* dan jumlah *hidden node* diketahui bahwa arsitektur MLP dengan jumlah *hidden layer* 1 dan jumlah *hidden node* 100 mempunyai nilai akurasi yang sama dengan 2 *hidden layer* dan *hidden node* 40 dan 20. Kedua arsitektur MLP tersebut mempunyai nilai akurasi yang sama besar yaitu 0,7391.

5.6.2. Membandingkan Jumlah *Epoch*

Setelah mendapatkan struktur dari arsitektur MLP, selanjutnya adalah membandingkan nilai *Epoch* untuk melatih model dalam mengklasifikasikan data. Berikut adalah perbandingan *epoch* 200 dan *epoch* 400:

Tabel 5.6. Perbandingan Jumlah *Epoch*

Epoch	**Jumlah *Hidden Layer***	**Jumlah *Hidden Node***	***Loss***	**Akurasi**
200	1	100	0,5269	0,7391
	2	40 & 20	0,5861	0,7391
400	1	100	0,4969	0,7826
	2	40 & 20	0,6076	0,6521

Berdasarkan pada **Tabel 5.6** dapat diketahui bahwa akurasi terbaik didapatkan arsitektur MLP dengan 1 *hidden layer* dan 100 *hidden node* dengan jumlah *epoch* 400.

Berdasarkan hasil pembuatan arsitektur MLP yang didapatkan arsitektur MLP terbaik dengan nilai akurasi tertinggi adalah dengan 1 *input layer* yang terdiri atas 80 *input node*, 1 *hidden layer* dengan 100 *hidden node*, 1 *output layer* dengan 3 *output node* dan dengan jumlah *epoch* 400. Arsitektur MLP tersebut mampu mengklasifikan data ekspresi gen pada manusia Normal, IGT, dan Diabetes Melitus Tipe 2 dengan akurasi sebesar 0,7826 atau sebesar 78,26%.

5.7 Xtreme Gradient Boosting *(XGBoost)*

Klasifikasi menggunakan *Xtreme Gradient Boosting* (XGBoost) dimulai dari menentukan nilai dari parameter-parameter yang digunakan pada XGBoost. Parameter-paremeter yang paneliti gunakan adalah *eta, max_depth, gamma, subsample, min_child_weight,* dan *colsample_by_tree.* Untuk mendapatkan model dengan parameter terbaik, maka dilakukan *tuning parameter* agar mendapatkan nilai dari masing-masing parameter yang digunakan.

Berdasarkan hasil *tuning parameter* didapatkan nilai 0, 5 untuk parameter *eta,* nilai terbaik parameter *max_depth* adalah 2, nilai terbaik parameter *gamma* adalah 2, nilai parameter *subsample* terbaik adalah 0,5, nilai parameter *min_child_weight* adalah 3, dan untuk parameter *colsample_by_tree* adalah 1. Model dengan nilai parameter-parameter tersebut mampu mendapatkan nilai akurasi sebesar 0,7391 dalam mengklasifikasikan data ekspresi gen pada manusia normal, IGT, dan diabetes melitus tipe 2.

5.8 Membandingkan Model dari SVM, MLP, dan XGBoost

Setelah didapatkan hasil klasifikasi dengan menggunakan metode SVM, MLP dan XGBoost selanjutnya adalah membandingkan nilai akurasi dari hasil klasifikasi metode-metode tersebut. Tabel 5.7 menunjukkan nilai akurasi dari masing-masing metode yang digunakan untuk mengklasifikasikan data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan Diabetes Melitus Tipe 2.

Tabel 5.7. Nilai Akurasi Metode SVM, MLP, dan XGBoost

Metode Klasifikasi	Akurasi
Support Vector Machine	91,30%
Multilayer Perceptron	78,26%
Xtreme Gradient Boosting	73,91%

Berdasarkan **Tabel 5.7** dapat diketahui bahwa dari ketiga metode yang digunakan untuk mengklasifikasikan data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan Diabetes Melitus Tipe 2 diperoleh metode klasifikasi dengan

menggunakan *Support Vector Machine* mampu mengklasifikasikan data ekspresi gen dengan nilai akurasi tertinggi yaitu sebesar 91,30%. Sedangkan metode klasifikasi *Multilayer Perceptron* mampu mengklasifikasikan data ekspresi gen dengan nilai akurasi sebesar 78,26% Metode klasifikasi terakhir *Xtreme Gradient Boosting* mampu mengklasifikasikan data ekspresi gen dengan nilai akurasi sebesar 73,91%.

5.9 Hasil Klasifikasi Model Terbaik

Berdasarkan subbab sebelumnya diketahui bahwa metode klasifikasi *Support Vector Machine* (SVM) merupakan metode klasifikasi dengan nilai akurasi tertinggi dalam mengklasifikasikan data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan Diabetes Melitus Tipe 2. Berikut ini adalah hasil klasifikasi dari model SVM dengan menggunakan kernel linear:

Tabel 5.8. Hasil Klasifikasi Data *Training*

Prediksi	**Aktual**		
	NGT	**IGT**	**DMT2**
NGT	34	5	4
IGT	1	13	1
DMT2	3	3	31

Berdasarkan Tabel 5.8. hasil klasifikasi pada data latih atau *train* didapatkan hasil dari sebanyak 95 sampel dengan keadaan Normal (NGT), *Impaired Glucose Tolerance* (IGT), dan Diabetes Melitus Tipe 2 (DMT2) model klasifikasi dapat mengklasifikasikan/memprediksi sebanyak 78 sampel sesuai dengan kelas aslinya. Sebanyak 34 sampel hasil prediksi normal sesuai dengan kelas aslinya, sebanyak 13 sampel hasil prediksi IGT sesuai dengan kelas aslinya, dan sebanyak 31 sampel prediksi DMT2 dapat sesuai dengan kelas aslinya. Sedangkan untuk hasil klasifikasi pada data uji atau *test* ditampilkan pada tabel berikut:

Tabel 5.9. Hasil Klasifikasi Data *Testting*

Prediksi	**Aktual**		
	NGT	**IGT**	**DMT2**
NGT	9	1	0
IGT	0	3	0
DMT2	0	1	9

Dari hasil klasifikasi pada Tabel 5.9. dapat diketahui nilai akurasi dari hasil klasifikasi pada data uji dengan menggunakan rumus perhitungan sebagai berikut:

$$Akurasi = \frac{Jumlah\ prediksi\ benar}{Jumlah\ total\ prediksi} = \frac{21}{23} = 91{,}30\%$$

Nilai akurasi dari hasil klasifikasi pada data uji didapatkan nilai sebesar 91,30% dimana terdapat 21 sampel hasil prediksi yang tepat sesuai dengan kelas aslinya. Dari 3 kelas yang ada diketahui pada kelas normal terdapat 9 sampel yang diprediksi tepat sesuai dengan kelas normal aslinya, pada kelas *impaired glucose tolerance* terdapat 3 sampel yang diprediksi sesuai dengan kelas *impaired glucose tolerance* aslinya, dan pada kelas diabetes melitus tipe 2 terdapat 9 sampel yang diprediksi sesuai dengan kelas diabetes melitus tipe 2 pada data aslinya.

Selain nilai akurasi untuk mengukur seberapa baik model klasifikasi yang didapatkan, digunakan juga ukuran-ukuran sebagai berikut:

$$Precision = \frac{9}{9+1}\ x\ 100\% = 0{,}9130 = 91{,}30\%$$

$$Recall/Sensitivity = \frac{9}{9+1}\ x\ 100\% = 0{,}9 = 90\%$$

$$Specificity = \frac{9}{9+1}\ x\ 100\% = 0{,}9 = 90\%$$

$$FPR = 1 - Specificity = 0{,}1 = 10\%$$

$$AUC = \frac{1+0{,}9-0{,}1}{2} = 0{,}9$$

Berdasarkan perhitungan diatas, didapatkan nilai AUC sebesar 0,9 yang berarti dapat dikatakan bahwa model klasifikasi dengan menggunakan metode SVM kernel linear merupakan model klasifikasi yang baik untuk digunakan mengklasifikasikan data ekspresi gen *skeletal muscle* normal (NGT), IGT, dan diabetes melitus tipe 2 yang didapatkan dari website NCBI dengan series GSE 18732 RAW.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil tahapan analisis yang telah dilakukan, diperoleh beberapa kesimpulan antara lain:

1. Berdasarkan gambaran data ekspresi gen *skeletal muscle* normal (NGT), IGT dan diabetes melitus tipe 2, menunjukkan bahwa penderita diabetes melitus tipe 2 banyak menyerang usia 20 tahun keatas dengan proporsi jenis kelamin laki-laki lebih besar menderita diabetes melitus tipe 2 dibandingkan dengan perempuan.
2. Metode klasifikasi *support vector machine* dengan menggunakan kernel linear mampu mengklasifikasikan data dengan nilai akurasi sebesar 91,30% sedangkan metode klasifikasi dengan menggunakan arsitektur *multilayer perceptron* dan *xtreme gradient boosting* mampu mengklasifikasikan data dengan nilai akurasi masing-masing sebesar 78,26% dan 73,91%.
3. Metode klasifikasi *support vector machine* lebih baik dalam mengklasifikasikan data ekspresi gen *skeletal muscle* normal (NGT), *Impaired Glucose Tolerance* (IGT), dan diabetes melitus tipe 2 dibandingkan 2 metode klasifikasi lainnya yang menggunakan arsitektur *multilayer perceptron* dan *xtreme gradient boosting* berdasarkan dari nilai akurasi yang didapatkan oleh masing-massing metode. Metode *support vector machine* mampu mengklasifikasikan sampel *impaired glucose tolerance* dengan tepat dan pada kelas sampel dengan kadar gula darah normal dan kelas diabetes melitus tipe 2, dari 10 sampel pada masing-masing kelas di data uji, metode ini mampu mengklasifikasikan 9 sampel dengan benar pada masing-masing kelas.

6.2 Saran

Adapun saran pada penelitian ini sebagai upaya perbaikan dan pengembangaan atau penelitian lanjutan adalah sebagai berikut:

1. Mengimplementasikan ketiga metode pada data ekspresi gen yang lainnya sebagai bahan perbandingan hasil klasifikasi.
2. Menggunakan teknik *ensemble* yang lainnya seperti *bagging* sebagai bahan perbandingan pada teknik *ensemble boosting.*
3. Menggunakan metode-metode penanganan *imbalance* data atau data yang tak seimbang seperti SMOTE dan yang lainnya.

DAFTAR PUSTAKA

Ardeshir, B. (2002). Science, Medicine, and the Future : Bioinformatics. *BMJ*, 1018-1022.

Ardilla, Y., Tjandrasa, H., & Arieshanti, I. (2014). Deteksi Penyakit Epilepsi dengan Menggunakan Entropi Permutasi, K-means Clustering, dan Multilayer Perceptron. *JURNAL TEKNIK POMTIS Vol. 3, No. 1*, A70 - A74.

Audina, N., P.Siregar, V., & Nurjaya, I. W. (2019). Analisis Perubahan Lahan Dan Sebaran Mangrove Menggunakan Algoritma Support Vector Machine (SVM) Dengan Citra Landsat Di Kabupaten Bintan Kepulauan Riau. *Jurnal Ilmu dan Teknologi Kelautan Tropis Vol. 11 No.1*, 49-63.

Bergeron, M. B. (2003). *Bioinformatics Computing.* New Delhi: Prentice Hall of India PVT LTD.

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

Budiharto, W. (2016). *Machine Learning & Computational Intelligence.* Yogyakarta: C.V ANDI OFFSET.

Chen, T., & Guestrin, C. (2016). XGBoost : A Scalable Tree Boosting System. *Vol.42, no. 8*, 665.

Dietterich, T. G. (2000). Multiple Classifier Systems. *International Workshop on Multiple Classifier Systems* (pp. 1-15). Berlin: Springer.

Dozmorov, M. (2016, October 3). *Filtering.* Retrieved from https://mdozmorov.github.io/BIOS567/assets/presentation_Bioconductor/Filtering.pdf.

Dufva, M. (2009). *DNA Microarrays for Biomedical Research.* switzerland: Springer.

Fatimah, R. N. (2015). DIABETES MELITUS TIPE 2. *J MAJORITY Volume 4 Nomor 5*, 101.

Gentleman, R., Carey, V., Huber, W., & Hahne, F. (2019). Package 'genefilter'. 19.

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

Handayani, A., Jamal, A., & Septiandri, A. A. (2017). Evaluasi Algoritme Berbasis Pembelajaran Mesin untuk Klasifikasi Jenis Tumor Payudara. *JNTETI, Vol. 6, No.4*, 394-403.

Haryati, D. F., Abdillah, G., & Hadiana, A. I. (2016). Klasifikasi Jenis Batubara Menggunakan Jaringan Syaraf Tiruan Dengan Algoritma Backpropagation. *Seminar Nasional Teknologi Informasi dan Komunikasi 2016 (SENTIKA 2016)*, (pp. 557-562). Yogyakarta.

Hofmann, M. (2006). Support Vector Machines — Kernels and the Kernel Trick. *http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf*.

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

Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2015). Deep Learning with S-shaped Rectified Linear Activation Units. *arXiv*.

Klasifikasi: Akurasi. (2018, 12 6). Retrieved from developers.google.com: https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=id

Maalik, I., Kusuma, W. A., & Wahjuni, S. (2019). Comparation Analysis of Ensemble Technique With Boosting (XGBOOST) And Bagging (Random Forest) For Classify Splice Junction DNA Sequence Category. *Jurnal Penelitian Pos dan Informatika*, 27-36.

Madigan, M. T., Martinko, J. M., V, D. P., & Clark, D. P. (2008). *Brock Biology of Microorganisms (12th Edition).* San Fransisco: Benjamin Cummings.

Naf'an, M. Z., & Arifin, J. (2017). Identifikasi Tanda Tangan Berdasarkan Grid Entropy Menggunakan Multi Layer Perceptron. *Jurnal INFOTEL Vol.9 NO.2*, 172-176.

Nugroho, A. S., Witarto, A. B., & Handoko, D. (2003). Support Vector Machine Teori dan Aplikasinya dalam Bioinformatika.

Pramana, S., Yuniarto, B., Mariyah, S., Santoso, I., & Nooraeni, R. (2018). *Data Mining dengan R*. Bogor: IN MEDIA.

Prasetyo, S. Y., Christianto, Y. B., & Hartomo, K. D. (2019). Analisis Data Citra Landsat 8 OLI Sebagai Indeks Prediksi Kekeringan Menggunakan Machine Learning di Wilayah Kabupaten Boyolali dan Purworejo. *Indonesian Journal of Computing and Modeling Volume 2 Nomor 2*, 25-36.

Primartha, R. (2018). *Belajar Machine Learning Teori Dan Praktik*. Bandung: Informatika Bandung.

Purwaningsih, N. (2016). PENERAPAN MULTILAYER PERCEPTRON UNTUK KLASIFIKASI JENIS KULIT SAPI TERSAMAK. *Jurnal TEKNOIF*, 1-6.

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

Puspitasari, A. M., Ratnawati, D. E., & Widodo, A. W. (2018). Klasifikasi Penyakit Gigi dan Mulut Menggunakan Metode Support Vector Mahine. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Vol. 2, No.2*, 802-810.

Raza, K. (2012). Application of Data Mining in Bioinformatics. *Indian Journal of Computer Science and Engineering*, 114-118.

Sanchez, A., & de Villa, M. C. (2008). *A Tutorial Review of Microarray Data Analysis*. Barcelona.

Serin, A. (2011). *Biclustering Analysis for Large Scale Data*. Jerman: Universitas Berlin.

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

suara.com. (2019, November 11). *health*. Retrieved from suara.com: https://www.suara.com/health/2019/11/11/183000/jelang-hari-diabetes-sedunia-mengapa-jumlah-pasien-terus-meningkat

Sugara, B., & Subekti, A. (2019). Penerapan Support Vector Machine (SVM) Pada Small Dataset Untuk Deteksi Dini Gangguan Autisme. *PILAR Nusa Mandiri Vol.15, No.2 September*, 177-182.

Susilawati, M. D., & Muljati, S. (2016). Hubungan Antara Intoleransi Glukosa dan Diabetes Melitus dengan Riwayat Tuberkulosis Paru Dewasa di Indonesia (Analisis Lanjut Riskesdas 2013). *Media Litabngkes, Vol. 26 No. 2*, 71-76.

Suyanto. (2017). *Data Mining* . Bandung: Informatika Bandung.

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

Syahrani, I. M. (2019). Analisis Perbandingan Teknik Ensemble Secara Boosting (XGBOOST) dan Bagging (RANDOM FOREST) Pada Klasifikasi Kategori Sambatan Sekuens DNA.

Tan, P. N., Steinbach, & Kumar, V. (2006). *Introduction to Data Mining.* USA: Pearson.

Toharin, S. N., Cahyati, W. H., & Zainafree, I. (2015). Hubungan Modifikasi Gaya Hidup Dan Kepatuhan Konsumsi Obat Antidiabetik Dengan Kadar Gula Darah Pada Penderita Diabetes Melitus Tipe 2 Di RS QIM Batang Tahun 2013. *Unnes Journal of Public Health*, 153-161.

topepo. (2020, 05 1). *caret*. Retrieved from github.com: https://github.com/topepo/caret/issues/336

Trevino, V., Falciani, F., & Barrera-Saldana, H. A. (2007). DNA Microarrays : a Powerful Genomic Tools for Biomedical and Clinical Research. *Molecular Medicine*, 527-541.

Vapnik, & N, V. (1999). *The Nature of Statistical Learning Theory 2nd.* New York Berlin Heidelberg: Springer.

Vidyanto, & Arifuddin, A. (2019). Determinan Peningkatan Kadar Gula Darah Pasien Interna Rumah Sakit Umum (RSU) Anutapura Palu. *Jurnal Kesehatan Tadulako Vol.5 No.1*, 1-62.

Walia, A. S. (2020, Mei 1). *Adaline-vs-MLP-vs-Perceptron*. Retrieved from github: https://github.com/anishsingh20/Adaline-vs-MLP-vs-Perceptron

Wargasetia, T. L. (2006). Peran Bioinformatika Dalam Bidang Kedokteran. *Maranatha Journal of Medicine and Health*, 59-70.

WHO. (2018, October 30). *News Room*. Retrieved from World Health Organization: https://www.who.int/health-topics/diabetes

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

Zhou, Z.-H. (2012). *Ensemble Methods: Foundation and Algorithms.* Boca Raton: CRC Press.

LAMPIRAN

Lampiran 1 *Script* Program *Software* R

```
# Packages-packages
library(affy)
library(GEOquery)
library(Biobase)
library(simpleaffy)
library(affyPLM)
library(u133x3pcdf)
library(u133x3p.db)
library(genefilter)
library(AnnotationDbi)
## Input Data D:\Iqbal\Skripsi\Data\GSE18732_RAW
GSE18732<- list.celfiles("D:/Iqbal/Skripsi/Data/GSE18732_RAW",
                         full.names=T)
GSE18732.AFFY = ReadAffy(filenames=GSE18732)
GSE18732.AFFY
class(GSE18732.AFFY)
### get pheno data ###
GSET_GSE18732 <- getGEO(GEO="GSE18732",GSEMatrix =TRUE)
str(GSET_GSE18732)
data.GSE18732 <- exprs(GSET_GSE18732[[1]])
class(data.GSE18732)
dim(data.GSE18732)
data.GSE18732[1:5,1:4]
pheno_GSE18732 <- pData(phenoData(GSET_GSE18732[[1]]))
str(pheno_GSE18732)
varLabels(phenoData(GSET_GSE18732[[1]]))
dim(pheno_GSE18732)
View(pheno_GSE18732)
setwd('D://Iqbal/Skripsi/Draft Skripsi')
#save pheno in csv file
write.csv(pheno_GSE18732, file ="D://Iqbal/Skripsi/
          Draft Skripsi/pheno data/pheno_GSE18732.csv" )

#### analisis deskriptif ####
##BOXPLOT
library(dplyr)
dev.new(width=4+dim(GSET_GSE18732)[[2]]/5, height=6)
par(mar=c(2+round(max(nchar(sampleNames(GSET_GSE18732))
                      )/2),4,2,1))
title <- paste ("GSE18732", '/','GPL9486',
                " selected samples", sep ='')
boxplot(data.gse_coba, boxwex=0.7, notch=T, main=title,
        outline=FALSE, las=2)
#Melihat Pheno
table(pheno_GSE18732$`gender:ch1`)
table(pheno_GSE18732$`age:ch1`)
#Barplot Gender
```

```
View(pheno_GSE18732)
des <-table(pheno_GSE18732$`gender:ch1`)
des <- as.data.frame(des)
barplot(des$Freq, main='Gender',col=c('yellow','blue'),
        xlab= 'Gender', ylab='Pasien',
        names.arg= c("Female","Male"))
des<-count(pheno_GSE18732, "pheno_GSE18732$age")
colnames(pheno_GSE18732)
#Pie Chart Age
library(plyr)
des2<-count(pheno_GSE18732, "characteristics_ch1.11")
des2$characteristics_ch1.11 = as.character(gsub(
  "age:", "", des2$characteristics_ch1.11))
percent <- round(des2$freq/sum(des2$freq)*100)
des2 <- as.data.frame(des2)
lbls2 <- paste(des2$characteristics_ch1.11, percent,
               '%', sep=' ')
pie(des2$freq,main ='Age',label= lbls2, col=c(
  "red","orange","yellow","blue","green"),border="brown",
  cex=0.5)

#### pre processing data ####
library(affyPLM)
eset.dChip=threestep(GSE18732.AFFY,
                     background.method = "RMA.2",
                     normalize.method="quantile",
                     summary.method="median.polish")
Ekspres.GSE18732 <- exprs(eset.dChip)
dim(eset.dChip)
#BOXPLOT
#set parameter and draw the plot
dev.new(width=4+dim(GSET_GSE18732)[[2]]/5, height=6)
par(mar=c(2+round(max(nchar(sampleNames(GSET_GSE18732))
                      )/2),4,2,1))
title <- paste ("GSE18732", '/','GPL9486',
                " selected samples", sep ='')
boxplot(Ekspres, boxwex=0.7, notch=T, main=title,
        outline=FALSE, las=2)

##### filtering #####
library(hgu133plus2.db)
filterdata <- nsFilter(eset.dChip, require.entrez =T,
                       var.func = IQR,
                       remove.dupEntrez = T,
                       var.cutoff = 0.5,
                       feature.exclude = "^AFFX")
log <-filterdata$filter.log
eset <- filterdata$eset
class(eset)
featureNames(eset) <- make.names(featureNames(eset))
#Melihat hasil filter (eset)
dim(eset)
head(eset)
class(eset)
eset
# Membuat data microarray menjadi matrix/data frame
```

```
databaru <- exprs(eset)
dim(databaru)
class(databaru)
head(databaru)
View(databaru)
#Kelas Vektor ada 3(Normal, Impaired Glucose Tolerance dan
Diabetes Melitus)
datacl <- c(1,0,1,0,2,0,2,2,2,1,0,1,2,1,1,0,2,2,0,2,
            0,0,2,0,2,1,2,2,0,0,2,2,1,2,1,0,0,2,2,0,
            2,1,0,2,2,0,0,1,2,0,2,2,2,1,1,0,2,2,0,2,
            2,2,2,2,1,0,1,2,2,2,1,2,0,2,2,2,2,0,2,0,
            0,0,1,0,1,0,1,0,2,0,0,0,0,1,0,0,2,0,0,0,
            0,1,0,1,2,0,2,0,1,1,0,0,0,1,2,1,0,0)
length(datacl)
class(datacl)
datafac <- factor(datacl,levels=0:2,
                  labels= c("NGT","IGT","DM"))
datafac
#### Filtering feature selection with multtest ####
library(multtest)
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)
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 kolom rawp
ffs <- data1[,1]
class(ffs)
length(ffs)
ffs[1 : 10]
#Adjusting rawp
datarawp <- data.frame(databaru, ffs)
row.names(datarawp) <- row.names(databaru)
class(datarawp)
head(datarawp)
dim(datarawp)
library(dplyr)
datarawpfilterfinal <- subset(datarawp, ffs < 0.00000001)
rownames(datarawpfilterfinal)
class(datarawpfilterfinal)
dim(datarawpfilterfinal)
head(datarawpfilterfinal)
## mendefinisikan data baru setelah filter
datadef <- datarawpfilterfinal[,1:118]
head(datadef)
```

```
dim(datadef)
colnames(datadef)
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))
dim(data_use)
summary(data_use)
#Usepackages(analisis)
library(e1071)
library(pROC)
library(devtools)
library(caret)
NGT<-dplyr::filter(data_use, data_use$dataY==0)
IGT<-dplyr::filter(data_use, data_use$dataY==1)
DTM<-dplyr::filter(data_use, data_use$dataY==2)
#penentuan ukuran data traning dan testing
set.seed(123)
rasio=8/10
n<-round(nrow(NGT)*rasio)
sampel_NGT<-sample(1:nrow(NGT),n)
m<-round(nrow(IGT)*rasio)
sampel_IGT<-sample(1:nrow(IGT),m)
x<-round(nrow(DTM)*rasio)
sampel_DTM<-sample(1:nrow(DTM),x)
#data training dan testing
#training
training_NGT=NGT[sampel_NGT,]
dim(training_NGT)
training_IGT=IGT[sampel_IGT,]
dim(training_IGT)
training_DTM=DTM[sampel_DTM,]
dim(training_DTM)
#testing
testing_NGT=NGT[-sampel_NGT,]
dim(testing_NGT)
testing_IGT=IGT[-sampel_IGT,]
dim(testing_IGT)
testing_DTM=DTM[-sampel_DTM,]
dim(testing_DTM)
data_training=rbind(training_NGT,training_IGT,training_DTM)
data_testting=rbind(testing_NGT,testing_IGT,testing_DTM)
dim(data_training)
dim(data_testting)
View(data_training)
#save data training dan testing .csv
setwd("D://Iqbal/Skripsi/Draft Skripsi")
write.csv(data_training,
          file      ="D://Iqbal/Skripsi/Draft      Skripsi/data
skripsi/train80gen.csv" )
write.csv(data_testting,
          file      ="D://Iqbal/Skripsi/Draft      Skripsi/data
skripsi/test80gen.csv" )
#### analisis klasifikasi menggunakan metode SVM ####
```

```
### 1. svm kernel linier
# 1.1.tuning (best parameter)
set.seed(12345)
tuning_lin <- tune(svm, dataY~. ,
                  data = data_training, kernel="linear",
                  types = "C-clasification", ranges= list(
                    cost = c(0.1,0.01, 0.001,1 , 10 , 100)))
summary(tuning_lin)
# 1.2. model svm kernel linier
# model_lin <- svm(dataY~. , data_training, kernel = "linear")
model_lin <- svm(dataY~. , data_training, kernel = "linear",
                  cost=0.01,scale=T,types = "C-clasification",
                  decision.value=T)
pred_train_lin <- predict(model_lin, data_training)
table(pred_train_lin, data_training$dataY)
pred_test_lin<-predict(model_lin, data_testting)
table(pred_test_lin, data_testting$dataY)
mean(pred_test_lin==data_testting$dataY)

### 2. svm kernel polynomial
# 2.1. tuning (best parameter)
set.seed(12345)
tuning_pol <- tune(svm, dataY~. , data = data_training,
                    kernel="polynomial",types        =        "C-
clasification",
                    ranges=         list(         cost         =
c(10,100,200,300,400,500)))
summary(tuning_pol)
# 2.2. model svm kernel polynomial
model_pol <- svm(dataY~., data_training, kernel="polynomial",
                  degree=3, cost=10,
                  types="C-clasification", decision.value=T)
pred_train_pol <- predict(model_pol, data_training)
table(pred_train_lin, data_training$dataY)
pred_test_pol <- predict(model_pol, data_testting)
table(pred_test_pol, data_testting$dataY)
mean(pred_test_pol==data_testting$dataY)

### 3. svm kernel sigmoid
# 3.1 tuning (best parameter)
set.seed(12345)
tuning_sig <- tune(svm, dataY~. , data = data_training,
                    kernel="sigmoid",
                    types = "C-clasification",
                    ranges= list(cost = c(
                      0.1,1,10,100,200,300),
                      gamma=c(0.1,1,2,3,4,5)))
summary(tuning_sig)
# 3.2 model svm kernel sigmoid
model_sig <- svm(dataY~., data_training,
                  kernel="sigmoid", cost=300,gamma=1,
                  types="C-clasification", decision.value=T)
pred_train_sig <- predict(model_sig, data_training)
table(pred_train_sig, data_training$dataY)
pred_test_sig <- predict(model_sig, data_testting)
table(pred_test_sig, data_testting$dataY)
```

```
mean(pred_test_sig==data_testting$dataY)

### 4. svm kernel RBF
# 4.1. tuning (best parameter)
set.seed(12345)
tuning_RBF <- tune(svm, dataY~. , data = data_training,
             kernel="radial",types = "C-clasification",
             ranges= list(
               cost = c(0.1,0.01, 0.001,1 , 10 , 100),
               gamma=c(1,2,3,4,5)))
summary(tuning_RBF)
# 4.2. model svm kernel RBF
model_rbf <- svm(dataY~., data_training,
                 kernel="radial", cost=0.1, gamma=1,
                 types="C-clasification", decision.value=T)
pred_train_rad <- predict(model_rbf, data_training)
table(pred_train_rad, data_training$dataY)
pred_test_rad <- predict(model_rbf, data_testting)
table(pred_test_rad, data_testting$dataY)
mean(pred_test_rad==data_testting$dataY)

##### Analisis Klasifikasi menggunakan metode MLP ####
library(keras)
library(tensorflow)
library(mlbench)
dim(data_training)
dim(data_testting)
#rubah data ke matrix
trainx <- data_training[,1:80]
trainx <- as.matrix(trainx)
testx <-data_testting[,1:80]
testx <- as.matrix(testx)

target_trainy<-as.numeric(as.factor(data_training[,81]))
target_testy<-as.numeric(as.factor(data_testting[,81]))
target_trainy<-target_trainy-1
target_testy<-target_testy-1
# one hot encoding
train_target <- to_categorical(target_trainy)
test_target <- to_categorical(target_testy)
# Epoch 200
# create sequential model
model.a <- keras_model_sequential()
model.a %>%
  layer_dense(units =20, activation = 'relu' ,
              input_shape = c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.a)
#compile
model.a %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.a <-model.a %>% fit(trainx, train_target,
                            epochs= 200, batch_size=10,
```

```
                              validation_split=0.2)
plot(history.a)
#evaluated model with data test
model1<- model.a %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.a%>%
  predict_proba(testx)
pred.a<-model.a%>%
  predict_classes(testx)
tabel1<-table(Predicted=pred.a, Actual=target_testy)

## create sequential model
model.b <- keras_model_sequential()
model.b %>%
  layer_dense(units =40, activation = 'relu' ,
              input_shape = c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.b)
#compile
model.b %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.b <-model.b %>% fit(trainx, train_target,
                            epochs= 200, batch_size=10,
                            validation_split=0.2)
plot(history.b)
#evaluated model with data test
model2<- model.b %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.b%>%
  predict_proba(testx)
pred.b<-model.b%>%
  predict_classes(testx)
tabel2<-table(Predicted=pred.b, Actual=target_testy)

## create sequential model
model.c <- keras_model_sequential()
model.c %>%
  layer_dense(units =60, activation = 'relu',
              input_shape = c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.c)
#compile
model.c %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.c <-model.c %>% fit(trainx, train_target,
                            epochs= 200,batch_size=10,
                            validation_split=0.2)
plot(history.c)
```

```
#evaluated model with data test
model3<- model.c %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.c%>%
  predict_proba(testx)
pred.c<-model.c%>%
  predict_classes(testx)
tabel3<-table(Predicted=pred.c, Actual=target_testy)

## create sequential model
model.d <- keras_model_sequential()
model.d %>%
  layer_dense(units =80, activation = 'relu' ,
              input_shape = c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.d)
#compile
model.d %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.d <-model.d %>% fit(trainx, train_target,
                            epochs= 200, batch_size=10,
                            validation_split=0.2)
plot(history.d)
#evaluated model with data test
model4<- model.d %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.d%>%
  predict_proba(testx)
pred.d<-model.d%>%
  predict_classes(testx)
tabel4<-table(Predicted=pred.d, Actual=target_testy)

## create sequential model
model.e <- keras_model_sequential()
model.e %>%
  layer_dense(units =100, activation = 'relu',
              input_shape = c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.e)
#compile
model.e %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.e <-model.e %>% fit(trainx, train_target,
                            epochs= 200, batch_size=10,
                            validation_split=0.2)
plot(history.e)
#evaluated model with data test
model5<- model.e %>%
```

```
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.e%>%
  predict_proba(testx)
pred.e<-model.e%>%
  predict_classes(testx)
tabel5<-table(Predicted=pred.e, Actual=target_testy)

## create sequential model
model.f <- keras_model_sequential()
model.f %>%
  layer_dense(units =100, activation = 'relu',
              input_shape = c(80)) %>%
  layer_dense(units =50, activation = 'relu') %>%
  layer_dense(units = 3) %>%
  layer_activation('softmax')
summary(model.f)
#compile
model.f %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.f <-model.f %>% fit(trainx, train_target,
                            epochs= 200,batch_size=10,
                            validation_split=0.2)
plot(history.f)

#evaluated model with data test
model6<- model.f %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.f%>%
  predict_proba(testx)
pred.f<-model.f%>%
  predict_classes(testx)
tabel6<-table(Predicted=pred.f, Actual=target_testy)
## create sequential model
model.g <- keras_model_sequential()
model.g %>%
  layer_dense(units =80, activation = 'relu',
              input_shape = c(80)) %>%
  layer_dense(units =40) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.g)
#compile
model.g %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.g <-model.g %>% fit(trainx, train_target,
                            epochs=200, batch_size=10,
                            validation_split=0.2)
plot(history.g)
#evaluated model with data test
model7<- model.g %>%
```

```
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.g%>%
  predict_proba(testx)
pred.g<-model.g%>%
  predict_classes(testx)
tabel7<-table(Predicted=pred.g, Actual=target_testy)
## create sequential model
model.h <- keras_model_sequential()
model.h %>%
  layer_dense(units =60, activation = 'relu',
              input_shape = c(80)) %>%
  layer_dense(units =30, activation = 'relu') %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.h)
#compile
model.h %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.h <-model.h %>% fit(trainx, train_target,
                            epochs=200,batch_size=10,
                            validation_split=0.2)
plot(history.h)
#evaluated model with data test
model8<- model.h %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.h%>%
  predict_proba(testx)
pred.h<-model.h%>%
  predict_classes(testx)
tabel8<-table(Predicted=pred.h, Actual=target_testy)
## create sequential model
model.i <- keras_model_sequential()
model.i %>%
  layer_dense(units =40, activation = 'relu' ,
              input_shape = c(80)) %>%
  layer_dense(units =20, activation = 'relu') %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.i)
#compile
model.i %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.i <-model.i %>% fit(trainx, train_target,
                            epochs= 200, batch_size=10,
                            validation_split=0.2)
plot(history.i)
#evaluated model with data test
model9<- model.i %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
```

```
prob<-model.i%>%
  predict_proba(testx)
pred.i<-model.i%>%
  predict_classes(testx)
tabel9<-table(Predicted=pred.i, Actual=target_testy)

######epoch 400
## create sequential model
model.e <- keras_model_sequential()
model.e %>%
  layer_dense(units =100, activation = 'relu' ,input_shape =
c(80)) %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.e)
#compile
model.e %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.e <-model.e %>% fit(trainx, train_target, epochs= 400,
                            batch_size=10, validation_split=0.2)
plot(history.e)
#evaluated model with data test
model5<- model.e %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.e%>%
  predict_proba(testx)
pred.e<-model.e%>%
  predict_classes(testx)
tabel5<-table(Predicted=pred.e, Actual=target_testy)

## create sequential model
model.i <- keras_model_sequential()
model.i %>%
  layer_dense(units =40, activation = 'relu' ,input_shape =
c(80)) %>%
  layer_dense(units =20, activation = 'relu') %>%
  layer_dense(units =3) %>%
  layer_activation('softmax')
summary(model.i)
#compile
model.i %>% compile(loss= 'categorical_crossentropy',
                    optimizer='adam',
                    metrics=c('accuracy'))
#fit model
history.i <-model.i %>% fit(trainx, train_target, epochs= 400,
                            batch_size=10, validation_split=0.2)
plot(history.i)
#evaluated model with data test
model9<- model.i %>%
  evaluate(testx, test_target)
#prediction & confussion matrix test data
prob<-model.i%>%
  predict_proba(testx)
```

```
pred.i<-model.i%>%
  predict_classes(testx)
tabel9<-table(Predicted=pred.i, Actual=target_testy)

##### Analisis Klasifikasi menggunakan metode XGBoost ####
library(xgboost)
library(keras)
library(tensorflow)
library(caret)
library(plyr)
library(Matrix)
## convert data to matrix
dim(data_training)
trainx <- data_training[,1:80]
trainx <- as.matrix(trainx)
testx <-data_testting[,1:80]
testx<-as.matrix(testx)
target_trainy<-as.numeric(as.factor(data_training[,81]))
target_testy<-as.numeric(as.factor(data_testting[,81]))
target_trainy<-target_trainy-1
target_testy<-target_testy-1
data_train<-data.frame(trainx,target_trainy)
data_test<-data.frame(testx, target_testy)
dim(data_train)
dim(data_test)
trainm <- sparse.model.matrix(data_train$target_trainy~.-1,
                              data=data_train)
head(trainm)
train_label <-data_train[,"target_trainy"]
train_matrix <-xgb.DMatrix(data = as.matrix(trainm),
                           label=train_label)
testm<-sparse.model.matrix(data_test$target_testy~.-1,
                           data = data_test)
test_label<-data_test[,"target_testy"]
test_matrix <-xgb.DMatrix(data = as.matrix(testm),
                          label=test_label)
# tuning parameter
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(target_trainy)~.,
                   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)
#XGB Model
set.seed(1000)
bst_model <- xgb.train(params = xgb_params,
                       data = train_matrix,
                       nrounds = 500,
                       watchlist = watchlist,
                       eta=0.5,max_depth=2,
                       gamma=2,colsample_bytree=1,
                       min_child_weight=3, subsample=0.5)
#training & test error plot
e<-data.frame(bst_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$test_mlogloss == 0.692789,]
#feature importance
imp<-xgb.importance(colnames(train_matrix),
                    model = bst_model)
print(imp)
xgb.plot.importance(imp)
library(dplyr)
#prediction
p<-predict(bst_model, newdata = test_matrix)
predik<-matrix(p, nrow = nc, ncol = length(p)/nc) %>%
  t()%>%
  data.frame()%>%
  mutate(label=test_label, max_prob=max.col(., "last")-1)
table(Prediction= predik$max_prob, Actual=predik$label)
```

Lampiran 2 Data Final untuk Analisis Klasfikasi

X1562921_at	**X237154_at**	**X214705_at**	**…**	**…**	**dataY**
3.303	6.690	4.311	…	…	**0**
3.800	7.299	5.023	…	…	**0**
3.429	6.797	4.610	…	…	**0**
3.485	6.875	4.466	…	…	**0**
3.289	7.244	5.928	…	…	**0**
3.963	7.262	4.319	…	…	**0**
4.261	7.272	4.562	…	…	**0**
3.779	6.849	4.149	…	…	**0**
3.669	6.949	4.369	…	…	**0**
3.317	6.899	4.546	…	…	**0**
4.695	6.922	4.122	…	…	**0**
3.859	7.232	4.606	…	…	**0**
4.038	6.873	4.207	…	…	**0**
:	:	:	…	…	**:**
:	:	:	…	…	**:**
3.841	7.768	4.617	…	…	**2**

Lampiran 3 Hasil Tuning Parameter Metode SVM Kernel Linear, *Polynomial*, RBF dan *Sigmoid*

```
> #### analisis klasifikasi menggunakan metode SVM ####
> ### 1. svm kernel linier
> # 1.1.tuning (best parameter)
> set.seed(12345)
> tuning_lin <- tune(svm, dataY~. ,
+                  data = data_training, kernel="linear",
+                  types = "C-clasification", ranges= list(
+                    cost = c(0.1,0.01, 0.001,1 , 10 , 100)))
> summary(tuning_lin)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 0.01

- best performance: 0.3444444

- Detailed performance results:
   cost     error dispersion
1 1e-01 0.3677778 0.13057328
2 1e-02 0.3444444 0.16480441
3 1e-03 0.5555556 0.19549101
4 1e+00 0.4322222 0.09078986
5 1e+01 0.4322222 0.09078986
6 1e+02 0.4322222 0.09078986
```

```
> ### 2. svm kernel polynomial
> # 2.1. tuning (best parameter)
> set.seed(12345)
> tuning_pol <- tune(svm, dataY~. , data = data_training,
+                    kernel="polynomial",types = "C-clasification",
+                    ranges= list( cost = c(10,100,200,300,400,500)))
> summary(tuning_pol)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
   10

- best performance: 0.3855556

- Detailed performance results:
  cost     error dispersion
1   10 0.3855556  0.2461807
2  100 0.3855556  0.2461807
3  200 0.3855556  0.2461807
4  300 0.3855556  0.2461807
5  400 0.3855556  0.2461807
6  500 0.3855556  0.2461807
```

```
> ### 3. svm kernel sigmoid
> # 3.1 tuning (best parameter)
> set.seed(12345)
> tuning_sig <- tune(svm, dataY~. , data = data_training,
+                    kernel="sigmoid",
+                    types = "C-clasification",
+                    ranges= list(cost = c(
+                      0.1,1,10,100,200,300),
+                      gamma=c(0.1,1,2,3,4,5)))
> summary(tuning_sig)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  300     1

- best performance: 0.3777778

- Detailed performance results:
   cost gamma     error dispersion
1   0.1   0.1 0.5077778  0.1777045
2   1.0   0.1 0.4766667  0.1838060
3  10.0   0.1 0.4955556  0.1805494
```

```
> ### 4. svm kernel RBF
> # 4.1. tuning (best parameter)
> set.seed(12345)
> tuning_RBF <- tune(svm, dataY~. , data = data_training,
+              kernel="radial",types = "C-clasification",
+              ranges= list(
+                cost = c(0.1,0.01, 0.001,1 , 10 , 100),
+                gamma=c(1,2,3,4,5)))
> summary(tuning_RBF)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  0.1     1

- best performance: 0.7022222

- Detailed performance results:
   cost gamma     error dispersion
1 1e-01     1 0.7022222  0.0942518
2 1e-02     1 0.7022222  0.0942518
3 1e-03     1 0.7022222  0.0942518
4 1e+00     1 0.7022222  0.0942518
5 1e+01     1 0.7022222  0.0942518
6 1e+02     1 0.7022222  0.0942518
```

Lampiran 4 Hasil Klasifikasi Metode SVM Kernel Linear, *Polynomial*, RBF, dan *Sigmoid*

```
> # 1.2. model svm kernel linier
> # model_lin <- svm(dataY~. , data_training, kernel = "linear")
> model_lin <- svm(dataY~. , data_training, kernel = "linear",
+                  cost=0.01,scale=T,types = "C-clasification",
+                  decision.value=T)
> pred_train_lin <- predict(model_lin, data_training)
> table(pred_train_lin, data_training$dataY)

pred_train_lin  0  1  2
             0 34  5  4
             1  1 13  1
             2  3  3 31
> pred_test_lin<-predict(model_lin, data_testting)
> table(pred_test_lin, data_testting$dataY)

pred_test_lin 0 1 2
            0 9 1 0
            1 0 3 0
            2 0 1 9
> mean(pred_test_lin==data_testting$dataY)
[1] 0.9130435
>
```

```
> # 2.2. model svm kernel polynomial
> model_pol <- svm(dataY~., data_training, kernel="polynomial",
+                  degree=3, cost=10,
+                  types="C-clasification", decision.value=T)
> pred_train_pol <- predict(model_pol, data_training)
> table(pred_train_lin, data_training$dataY)

pred_train_lin  0  1  2
             0 34  5  4
             1  1 13  1
             2  3  3 31
> pred_test_pol <- predict(model_pol, data_testting)
> table(pred_test_pol, data_testting$dataY)

pred_test_pol 0 1 2
            0 9 1 3
            1 0 1 0
            2 0 3 6
> mean(pred_test_pol==data_testting$dataY)
[1] 0.6956522
>
```

```
> # 3.2 model svm kernel sigmoid
> model_sig <- svm(dataY~., data_training,
+                  kernel="sigmoid", cost=300,gamma=1,
+                  types="C-clasification", decision.value=T)
> pred_train_sig <- predict(model_sig, data_training)
> table(pred_train_sig, data_training$dataY)

pred_train_sig  0  1  2
             0 22 11 16
             1  8  7  9
             2  8  3 11
> pred_test_sig <- predict(model_sig, data_testting)
> table(pred_test_sig, data_testting$dataY)

pred_test_sig 0 1 2
            0 6 2 5
            1 3 1 1
            2 0 2 3
> mean(pred_test_sig==data_testting$dataY)
[1] 0.4347826
>
```

```
> # 4.2. model svm kernel RBF
> model_rbf <- svm(dataY~., data_training,
+                  kernel="radial", cost=0.1, gamma=1,
+                  types="C-clasification", decision.value=T)
> pred_train_rad <- predict(model_rbf, data_training)
> table(pred_train_rad, data_training$dataY)

pred_train_rad  0  1  2
             0 38 21 36
             1  0  0  0
             2  0  0  0
> pred_test_rad <- predict(model_rbf, data_testting)
> table(pred_test_rad, data_testting$dataY)

pred_test_rad 0 1 2
            0 9 5 9
            1 0 0 0
            2 0 0 0
> mean(pred_test_rad==data_testting$dataY)
[1] 0.3913043
>
```

Lampiran 5 Hasil Klasifikasi Arsitektur MLP *Epoch* 200

```
> tabel1<-table(Predicted=pred.a, Actual=target_testy)
> model1
$loss
[1] 0.734879

$accuracy
[1] 0.6086956

> tabel1
         Actual
Predicted 0 1 2
        0 8 2 2
        1 1 3 4
        2 0 0 3
>
```

```
> tabel2<-table(Predicted=pred.b, Actual=target_testy)
> model2
$loss
[1] 0.7233058

$accuracy
[1] 0.6956522

> tabel2
         Actual
Predicted 0 1 2
        0 9 2 4
        1 0 3 1
        2 0 0 4
>
```

```
> tabel3<-table(Predicted=pred.c, Actual=target_testy)
> model3
$loss
[1] 0.8643424

$accuracy
[1] 0.5652174

> tabel3
         Actual
Predicted 0 1 2
        0 9 3 6
        1 0 2 1
        2 0 0 2
>
```

```
> tabel4<-table(Predicted=pred.d, Actual=target_testy)
> model4
$loss
[1] 0.7063335

$accuracy
[1] 0.6521739

> tabel4
         Actual
Predicted 0 1 2
        0 8 2 5
        1 1 3 0
        2 0 0 4
>
```

```
> tabel5<-table(Predicted=pred.e, Actual=target_testy)
> model5
$loss
[1] 0.5269752

$accuracy
[1] 0.7391304

> tabel5
         Actual
Predicted 0 1 2
        0 4 0 0
        1 4 5 1
        2 1 0 8
>
```

```
> tabel6<-table(Predicted=pred.f, Actual=target_testy)
> model6
$loss
[1] 0.8370701

$accuracy
[1] 0.5652174

> tabel6
         Actual
Predicted 0 1 2
        0 9 3 6
        1 0 2 1
        2 0 0 2
>
```

```
> tabel7<-table(Predicted=pred.g, Actual=target_testy)
> model7
$loss
[1] 0.7702811

$accuracy
[1] 0.6086956

> tabel7
         Actual
Predicted 0 1 2
        0 8 1 3
        1 1 4 4
        2 0 0 2
>
```

```
> tabel8<-table(Predicted=pred.h, Actual=target_testy)
> model8
$loss
[1] 0.7163099

$accuracy
[1] 0.5652174

> tabel8
         Actual
Predicted 0 1 2
        0 9 3 6
        1 0 2 1
        2 0 0 2
>
```

```
> tabel9<-table(Predicted=pred.i, Actual=target_testy)
> model9
$loss
[1] 0.5861403

$accuracy
[1] 0.7391304

> tabel9
         Actual
Predicted 0 1 2
        0 8 2 2
        1 0 3 1
        2 1 0 6
>
```

Lampiran 6 *Hasil Klasifikasi Arsitektur MLP Epoch 400*

```
> tabel5<-table(Predicted=pred.e, Actual=target_testy)
> model5
$loss
[1] 0.4969557

$accuracy
[1] 0.7826087

> tabel5
         Actual
Predicted 0 1 2
        0 8 1 2
        1 0 4 1
        2 1 0 6
>
```

```
> tabel9<-table(Predicted=pred.i, Actual=target_testy)
> model9
$loss
[1] 0.6289572

$accuracy
[1] 0.6521739

> tabel9
         Actual
Predicted 0 1 2
        0 8 3 3
        1 0 2 1
        2 1 0 5
>
```

Lampiran 7 Hasil *Tuning* Parameter Metode Klasifikasi XGBoost

```
> print(modFitxgb)
eXtreme Gradient Boosting

95 samples
80 predictors
 3 classes: '0', '1', '2'

No pre-processing
Resampling: Bootstrapped (5 reps)
Summary of sample sizes: 95, 95, 95, 95, 95
Resampling results across tuning parameters:

  eta    max_depth  gamma  min_child_weight  subsample  Accuracy   Kappa
  0.001  2          1      1                 0.25       0.5610526  0.3216787
  0.001  2          1      1                 0.50       0.5603027  0.3172798
  0.001  2          1      1                 0.75       0.5613794  0.3248885
  0.001  2          1      2                 0.25       0.5683385  0.3254541
  0.001  2          1      2                 0.50       0.5650740  0.3296116
  0.001  2          1      2                 0.75       0.5790609  0.3462221
  0.001  2          1      3                 0.25       0.5665153  0.3198008
  0.001  2          1      3                 0.50       0.5417062  0.2939737
  0.001  2          1      3                 0.75       0.5451324  0.2991390
  0.001  2          2      1                 0.25       0.5610526  0.3135859
  0.001  2          2      1                 0.50       0.5661851  0.3273587
  0.001  2          2      1                 0.75       0.5456244  0.2968938
  0.001  2          2      2                 0.25       0.5663158  0.3245776
  0.001  2          2      2                 0.50       0.5367699  0.2827811
  0.001  2          2      2                 0.75       0.5623598  0.3208414
  0.001  2          2      3                 0.25       0.5668077  0.3211060
  0.001  2          2      3                 0.50       0.5469694  0.3009853
  0.001  2          2      3                 0.75       0.5507224  0.3024067
  0.001  2          4      1                 0.25       0.5429481  0.2832784
  0.001  2          4      1                 0.50       0.5302683  0.2756301
  0.001  2          4      1                 0.75       0.5397420  0.2877302
  0.001  2          4      2                 0.25       0.5475920  0.2954622
  0.001  2          4      2                 0.50       0.5414138  0.2919486
  ...
  0.005  2          2      3                 0.75       0.5417062  0.2978425
  0.005  2          4      1                 0.25       0.5488304  0.2929038
  0.005  2          4      1                 0.50       0.5403027  0.2896614
  0.005  2          4      1                 0.75       0.5352047  0.2837346
  0.005  2          4      2                 0.25       0.5545855  0.3014816
  0.005  2          4      2                 0.50       0.5477537  0.2977263
  0.005  2          4      2                 0.75       0.5352047  0.2840027
  0.005  2          4      3                 0.25       0.5504334  0.2918030
  0.005  2          4      3                 0.50       0.5603027  0.3140624
  0.005  2          4      3                 0.75       0.5226901  0.2614964
  0.005  4          1      1                 0.25       0.5710870  0.3387147
  0.005  4          1      1                 0.50       0.5720330  0.3424183
  0.005  4          1      1                 0.75       0.5609219  0.3233078
  0.005  4          1      2                 0.25       0.5900103  0.3586928
  0.005  4          1      2                 0.50       0.5586997  0.3245579
  0.005  4          1      2                 0.75       0.5668043  0.3313995
  0.005  4          1      3                 0.25       0.5777537  0.3441842
 [ reached getOption("max.print") -- omitted 506 rows ]

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 = 0.5, gamma =
 2, colsample_bytree = 1, min_child_weight = 3 and subsample = 0.5.
>
```

Lampiran 8 Hasil Klasifikasi Metode XGBoost

```
> #prediction
> p<-predict(bst_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)
> table(Prediction= predik$max_prob, Actual=predik$label)
          Actual
Prediction 0 1 2
         0 7 1 3
         1 2 4 0
         2 0 0 6
> 17/23
[1] 0.7391304
>
```

Lampiran 9 *Output Session* info *software* R

```
> sessionInfo()
R version 3.6.2 (2019-12-12)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows >= 8 x64 (build 9200)
```