BAB II
TINJAUAN PUSTAKA

2.1 *Machine Learning*

Machine Learning telah menjadi topik hangat di dunia dalam beberapa tahun terakhir karena mampu mengubah dan menyederhanakan cara orang hidup dan bekerja. *Machine Learning* adalah salah satu disiplin ilmu dalam kecerdasan buatan (*Artificial Intelligence*) dengan mengembangakan sistem komputer sehingga mampu meningkatkan kinerja. *Machine Learning* terbagi menjadi tiga kategori antara lain *unspervised learning, supervised learning, reinforcement learning*. Adapun penjelasan dari ketiga kategori *Machine Learning* adalah sebagai berikut (Roihan, 2020).

2.1.1 *Unsupervised Learning*

Unsupervised learning merupakan kategori *Machine Learning* bersifat deskriptif sehingga berguna dalam mengelompokkan dan mengkategorikan data dan membutuhkan pembelajaran dari data yang telah ada. Proses penggunaan *unsupervised learning* yaitu tidak membutuhkan *training* data dan tidak ada pemberian label sehingga hasil tidak mengidentifikasi kelas kelas yang telah ditentukan yang dapat digunakan pada permasalahan *clustering*. *Clustering* adalah pengelompokkan data dalam sejumlah kelompok dengan memiliki kesamaan karakteristik pada kelompok yang sama. Algoritma yang dapat digunakan dalam c*lustering* yaitu algoritma *K-Means Clustering, Maen Shift Clustering,* DBSCAN *Clustering, Agglomerative Hierarchical Clustering,* dan *Gaussian Mixture* (Roihan, 2020)

2.1.2 *Supervised Learning*

Supervised learning merupakan kategori *Machine Learning* bersifat prediktif sehingga hasil pembelajaran mampu digunakan sebagai sebuah prediks dan mampu menyelesaikan permasalahan data secara linear, multilinear, dan polinomial. Proses penggunaan *supervised learning* didasarkan pada kumpulan sampel data yang memiliki label untuk membangun sebuah model yadeng tingkat keakurasiannya semakin tinggi dari waktu ke waktu yang mampu digunakan pada

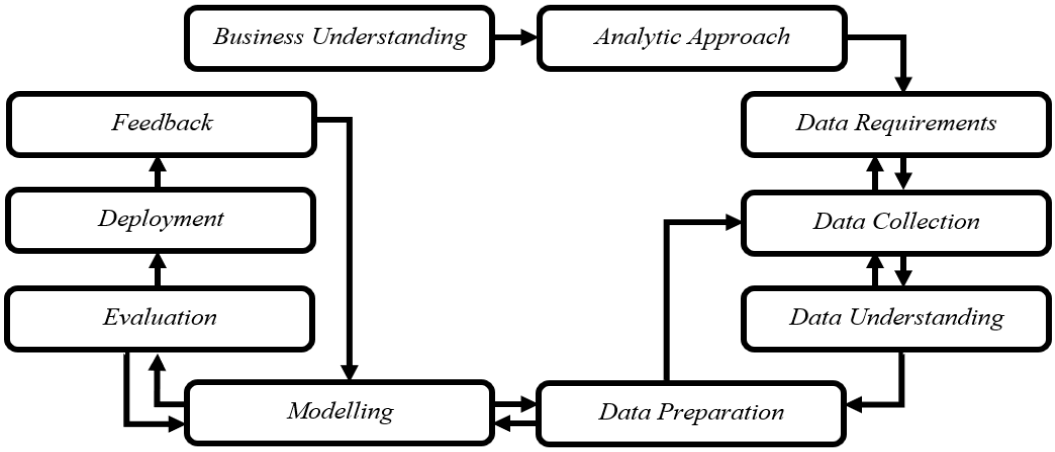
permasalahan regresi dan klasifikasi. Regresi adalah metode dalam memodelkan suatu hubungan antara variabel *input* dan variabel output kontinu sehingga mampu memprediksi hasil berupa nilai kontinu dalam rentang tertentu. Algoritma yang dapat digunakan dalam regresi yaitu *Linear Regression, Neural Network Regression, Support Vector Regression, Decision Tree Regression, Lasso Regression*, dan *Ridge Regression*. Klasifikasi adalah proses mengelompokkan dengan memisahkan data berdasarkan fitur-fitur tertentu sehingga mampu memprediksi sampel data berdasarkan satu kelas tertentu dengan hasil berupa dua kelas ataupun lebih dari dua kelas. Algoritma yang dapat digunakan dalam klasifikasi *Decision Tree, Random Forest, Support Vector Machine, Naive Bayes,* KNN (K-*Nearest Neighbor*), *XGBoost,* MLP *Classifier,* dan *Logistic Regression* (Abijono, 2021)

2.1.3 *Reinforcement Lerning*

Reinforcement learning merupakan kategori *Machine Learning* yang tidak membutuhkan pembelajaran sebelumnya sehingga mampu dengan mandiri mendapatkan hasil dari pembelajaran yang dilakukan secara berulang ulang dengan interaksi lingkungan yang dinamis sehingga cukup berbeda dengan algoritma *unsupervised learning* dan *supervised learning.* Proses penggunaan *reinforcement learning* yaitu dilakukan *trial and error* untuk mempelajari pengalaman baru. Metode r*einforcement learning* berdasarkan model proses pengambilan keputusan Markov meliputi metode berbasis model seperti algoritma SARSA (*State Action Reward State Action*) yaitu r*einforcement learning* terlebih dahulu mempelajari pengetahuan model dan kemudian memperoleh strategi optimal dari pengetahuan model tersebut dan metode terkait model seperti algoritma *Temporal Difference* dan algoritma *Q-learning,* di mana pembelajaran penguatan secara langsung menghitung strategi optimal tanpa pengetahuan model (Roihan, 2020).

2.2 Tahapan *Machine Learning*

Machine Learning memiliki beberapa tahapan dalam pembuatan model hingga pengembangan aplikasi. Tahapan *Machine Learning* antara lain *business understanding, analytic approach, data requirements, data collection, data understanding, data preparation, modelling, evaluation,* dan *deployment*. Berikut gambar skema dalam tahapan *Machine Learning* (Smith, 2021).



Gambar 1. Skema *Machine Learning*

Berdasarkan Gambar 1. terdapat 10 tahapan dalam melakukan *Machine Learning*. Dalam skema juga terlihat terdapat panah bolak balik menunjukkan bahwa proses dapat kembali ke tahapan sebelumnya apabila hasil yang diperoleh pada tahap sebelumnya belum mencukupi dan perlu beberapa tahapan ulang untuk mendapatkan hasil yang lebih dari hasil awal. Adapun penjabaran terkait tahapan *Machine Learning* adalah sebagai berikut (Smith, 2021).

2.2.1 *Business Understanding*

Business understanding adalah tahapan pertama dalam *Machine Learning* untuk memahami masalah dan menentukan tujuan permasalahan. *Business understanding* menjadi dasar utama dalam suatu penelitian. Pemahaman yang jelas tentang permasalahan akan membuat penyelesaian penelitian dengan mudah. Tidak ada perumusan matematis untuk *business understanding* namun peneliti harus dapat ber*brainstorming* dengan topik yang telah ditentukan (Smith, 2021).

2.2.2 *Analytic Approach*

Analytic approach adalah tahapan kedua dalam *Machine Learning* untuk mengetahui pendekatan apa yang ingin digunakan sehingga mampu menentukan kategori dan model dalam penyelesaian permasalahan suatu penelitian. *Analytic approach* terdapat beberapa jenis pendekatan antara lain *descriptive approach, diagnostic approach, predictive approach,* dan *precriptive approach*. *Descriptive approach* adalah pendekatan untuk memberikan gambaran tentang kondisi saat ini dengan mendeskripsikan peristiwa, situasi, perilaku, subjek atau fenomena yang sedang terjadi di masyarakat. *Diagnostic approach* adalah pendekatan untuk

memahami penyebab di balik peristiwa seperti mengidentifikasi hubungan sebab-akibat. *Predictive aproach* adalah pendekatan untuk meramalkan atau memprediksi kejadian di masa mendatang seperti memprediksi waktu kelulusan mahasiswa. *Presciptipive approach* adalah pendekatan untuk memberikan rekomendasi atau solusi optimal dalam menangami masalah seperti memberikan rekomendasi produk kepada pengguna (Smith, 2021).

2.2.3 *Data Requirements*

Data requirements adalah tahapan ketiga dalam *Machine Learning* untuk mengidentifikasi data yang dapat dibutuhkan dalam menyelesaikan permasalahan. Setelah mengidentifikasi permasalahan dan tujuan yang ingin dicapai, peneliti melakukan identifikasi jenis data yang diperlukan, sumber data, dan volume data yang cukup untuk menghasilkan analisis yang akurat (Smith, 2021).

2.2.4 *Data Collection*

Data collection adalah tahapan keempat dalam *Machine Learning* untuk mengumpulkan data-data yang dibutuhkan untuk menyelesaikan permasalahan. Pengumpulan data dapat dilakukan beberapa cara antara lain menyebarkan kuisioner, mengamati peristiwa secara langsung, meminta data tertulis atau dokumen yang telah dikumpulkan pihak lain, melakukan wawancara dengan narasumber, ataupun melakukan eksperimen mandiri (Smith, 2021).

2.2.5 *Data Understanding*

Data understanding adalah tahapan kelima dalam *Machine Learning* untuk memahami data yang telah dikumpulkan. *Data understanding* dapat dilakukan dengan melakukan visualisasi data dalam bentuk histogram atau grafik, menentukan statistika deskriptif, ataupun menentukan korelasi data (Smith, 2021).

2.2.6 *Data Preparation*

Data preparation adalah tahapan keenam dalam *Machine Learning* untuk mempersiapkan data untuk dilakukan pemodalan atau *modelling*. *Data preparation* menjadi langkah penting dikarenakan harus memastikan haswa data berada dalam format yang benar untuk algoritma. Apabila terdapat data yang berada dengan format yang tidak sesuai dalam *modelling*, maka harus harus kembali ke tahap *data preparation*. *Data preparation* dapat dilakukan dengan melakukan identifikasi *missing value*, *imbalnced data*, ataupun normalisaasi (Smith, 2021).

Missing value dapat terjadi pada suatu data ketika data tidak ada atau tidak lengkap dalam data. *Missing value* dapat mempengaruhi hasil analisis dan menghasilkan kesimpulan yang tidak akurat jika diatasi dengan benar. Dalam mengatasi *missing value* dapat dilakukan dengan cara menghapus data yang hilang, menggantikan *missing value* dengan *mean* data, dan menyandikan *missing value* dengan suatu label (Emmanuel, 2021).

Imbalanced data dapat terjadi pada suatu data ketika banyaknya kelas tidak seimbang atau jumlah antara kelas mayoritas dan kelas minoritas tidak sama. *Imbalanced data* dapat mempengaruhi hasil analisis karena hasil prediksi akan lebih condong ke kelas mayoritas. Dalam mengatasi *imbalanced data* dapat dilakukan *handling imbalanced data* dengan cara *random over sampling* (menambahkan beberapa data acak ke kelas minoritas sehingga jumlah kelas minoritas akan sama dengan jumlah kelas mayoritas), *random under sampling* (mengurangi beberapa data dari kelas mayoritas sehingga jumlah kelas mayoritas akan sama dengan jumlah kelas minoritas), dan SMOTE (*Synthetic Minority Over Samppling Technique* atau menciptakan data baru yang mirip dengan data minoritas untuk ditambahkan ke kelas minoritas sehingga jumlah kelas minoritas akan sama dengan jumlah kelas mayaritas) (Indrawati, 2020).

2.2.7 *Modelling*

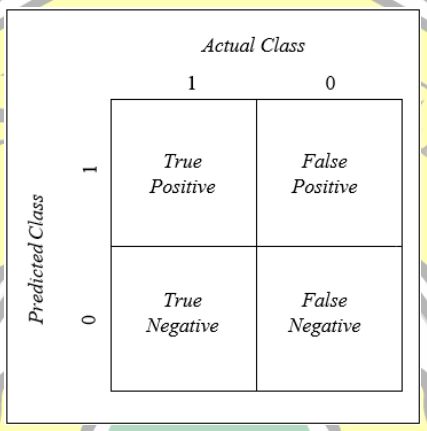
Modelling adalah tahapan ketujuh dalam *Machine Learning* untuk proses pembuatan dan pengembangan model untuk menyelesaikan permasalahan. Model yang dibuat berdasarkan pendekatan yang telah ditentukan. Dalam beberapa model, perlu dilakukan beberapa iterasi pengujian untuk mendapatkan hasil sesuai dengan yang ditentukan. Selain itu, dengan menentukan parameter parameter tertentu dapat membuat hasil menjadi lebih optimal (Smith, 2021).

Proses *modelling* dapat dilakukan dengan *cross calidation. Cross validation* dilakukan untuk membagi data menjadi data *training* dan data *testing* menjadi beberapa *fold* atau lipatan. *Cross validation* dilakukan dengan memanfaatkan data yang terbatas sehingga data yang kecil mampu membuat prediksi yang tepat. Pengujian model menggunakan *cross validation* diharapkan mampu memperoleh hasil maksimal pada kinerja model. *Cross validation* dapat dilakukan dengan beberapa cara antara lain K-*Fold Cross Validation, Hold-Out Cross Validation*, dan

Stratified K-*Fold Cross Validation.* Pada *cross validation*, data *training* dan data *testing* akan dilakukan *crossover* sehingga semua data memiliki kesempatan untuk tervalidasi. (Saifudin, 2018).

2.2.8 *Evaluation*

Evaluation atau evaluasi adalah tahapan kedelapan dalam *Machine Learning* untuk mengukur sejauh mana suatu model mampu mencapai tujuan, standar, dan harapan yang telah ditetapkan. Beda kategori algoritma maka berbeda pula metrik evaluasi yang dihasilkan. Dalam evaluasi terdapat *confusion matrix* sebagai tabel matriks. (Smith, 2021).



Gambar 2. *Confusion Matrix*

Confusion matrix adalah tabel matriks yang digunakan untuk mengukur kinerja suatu model klasifikasi terdiri dari *True Positive*, *False Positive*, *True Negative*, dan *False Negative*. *True Positive* adalah kondisi data bernilai positif pada keadaan sebenarnya dan bernilai positif pada hasil prediksi. *False Positive* adalah kondisi data bernilai negatif pada keadaan sebenarnya namun bernilai positif pada hasil prediksi. *True Negative* adalah kondisi data bernilai positif pada keadaan sebenarnya namun bernilai negatif pada hasil prediksi. *False Negative* adalah kondisi data bernilai negatif pada keadaan sebenarnya dan bernilai negatif pada hasil prediksi (Normawati, 2021).

Dalam kategori algoritma klasifikasi, evaluasi menggunakan metrik-metrik tertentu yaitu *accuracy*, *precision*, *recall*, dan, F1-*score*. *Accuracy* adalah perbandingan antara jumlah prediksi yang benar secara keseluruhan (*True Positive* + *True Negative*) dibandingan jumlah total data. *Precision* adalah perbandingan jumlah prediksi positif yang benar (*True Positive)* dibandingkan dengan jumlah

prediksi positif secara keseluruhan *(True Positive + False Positives*). *Recall* adalah perbandingan antara jumlah prediksi positif yang benar (*True Positives)* dibandingkan dengan jumlah semua kasus yang seharusnya positif *(True Positive + False Negatives*). Berikut rumus dari *confusion matrix* (Normawati, 2021)

$$Precision\ (P) = \frac{TP}{TP+FP} \quad \text{.....(1)}$$

$$Recall\ (R) = \frac{TP}{TP+FN} \quad \text{.....(2)}$$

$$F1\ Score = 2\ x\ \frac{P\ x\ R}{P+R} \quad \text{.....(3)}$$

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad \text{.....(4)}$$

Pada kasus yang memiliki keseimbangan pada kelas mayoritas dan minoritas, evaluasi lebih tepat digunakan mengggunakan *Precision* dan *Accuracy*. Namun, pada kasus yang memiliki ketidakseimbangan pada kelas mayoritas dan minoritas, evaluasi model lebih tepat menggunakan ROC-AUC. ROC-AUC (*Area Under the Receiver Operatting Characteristics Curve*) adalah metrik evaluasi mengukur kinerja model dengan memisahkan antara kelas positif dan negatif berkisar dari 0 hingga 1 (Puad, 2023). Apabila nilai ROC-AUC diatas 0,6 maka klasifikasi dianggap berhasil dan masuk ke dalam kategori *Excellent, Good, Fair*, atau *Poor* sedangkan nilai evaluasi dibawah 0,6 dianggap proses klasifikasi gagal dan hasil klasifikasi tidak bisa diterapkan (Andriani, 2017). Level pengukuran kualitas klasifikasi menggunakan ROC-AUC dilihat berdasarkan akurasi dengan rentang sebagai berikut (Puad, 2023).

a. 1,00 – 0,90 = *Excellent Classification*
b. 0,90 – 0,80 = *Good Classification*
c. 0,80 – 0,70 = *Fair Classification*
d. 0,70 – 0,60 = *Poor Classification*
e. 0,60 – 0,50 = *Failure Classification*

2.2.9 *Deployment*

Deployment adalah tahapan kesembilan dalam *Machine Learning* untuk membangun, mengimplementasikan, mengembangkan, dan siap digunakan oleh

pengguna akhir menggunakan aplikasi atau *website* sesuai dengan tujuan permasalahan dan kemampuan pengembang. Ada beberapa metode yang dapat digunakan antara lain aplikasi berbasis *web* seperti Streamlit, Flask, dan Django, *Application Programming Interface* seperti penggunaan HTTP, ataupun aplikasi *mobile* seperti Java (Android) dan Swift (iOS) (Smith, 2021).

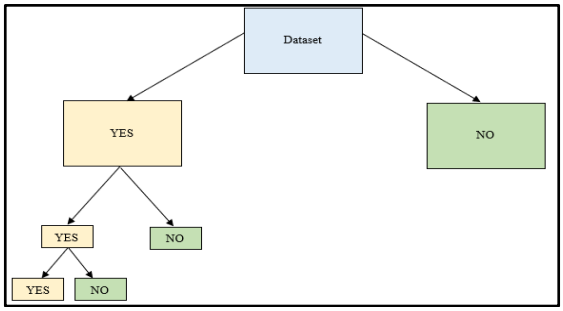
2.3 Algoritma Klasifikasi *Machine Learning*

Machine Learning adalah salah satu disiplin ilmu dalam kecerdasan buatan (*Artificial Intelligence*) dengan mengembangakan sistem komputer sehingga mampu meningkatkan kinerja. Salah satu kategori pada *Machine Learning* yaitu klasifikasi. Berikut penjelasan dari algoritma klasifikasi *Machine Learning*. (Buslim, 2019)

2.3.1 Decision Tree

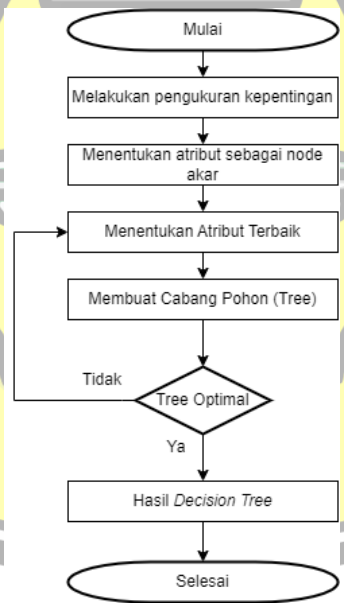
Decision Tree adalah algoritma klasifikasi *Machine Learning* untuk memrepresentasikan secara sederhana sejumlah kelas terdiri dari simpul dan cabang digunakan ntuk mengambil keputusan atau membuat prediksi. Setiap simpul atau *node* pohon merepresentasikan variabel yang telah diuji. Setiap cabang merepresentasikan pembagian hasil uji. Setiap *node* daun (*leaf*) merepresentasikan kelompok kelas tertentu. Level *node* teratas dari sebuah *Decision Tree* adalah akar (*root*) yang biasanya berupa variabel yang paling memiliki pengaruh terbesar pada suatu kelas tertentu (Nasrullah, 2021).

Penggunaan algoritma *Decision Tree* memiliki beberapa kelebihan. Kelebihan yang dimiliki *Decision Tree* yaitu memiliki sifat yang fleksibel sehingga mampu meningkatkan kualitas keputusan yang dihasilkan, hasil *output* mudah dipahami oleh orang awam, dan dapat digunakan pada klasifikasi dan regresi sedangkan kekurangan dari algoritma ini adalah akan terjadi *overfitting* jika menggunakan data yang memiliki kelas dan kriteria dengan jumlah yang banyak (Permana, 2021).



Gambar 3. Skema *Decision Tree*

Decision Tree pada perhitungan diawali dengan menentukan nilai *entropy* dan *gain*. *Entropy* dan *gain* digunakan untuk menentukan *leaf node* atau ujung pohon yang hanya memiliki satu *input* dan tidak ada *output*. *Entropy* adalah jumlah data yang tidak termasuk ke dalam informasi dataset sedangkan *gain* adalah informasi yang didapatkan dari *entropy*. Nilai yang dihasilkan dari *gain* maka dapat membentuk suatu *Decision Tree*. Semakin tinggi nilai *gain* maka akan memeiliki kedudukan yang lebih tinggi dalam *Decision Tree*. (Harryanto, 2017). Berikut *flowchart Decision Tree* adalah sebagai berikut (Abdillah, 2020)



Gambar 4. *Flowchart Decision Tree*

Adapun penjelasan *flowchart Decision Tree* adalah sebagai berikut (Abdillah, 2020).

1. Mulai

   Memulai perhitungan dengan memasukkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Melakukan pengukuran kepentingan

   Menghitung sebarapa baik pengukuran dalam pemisahan kelas dengan perhitungan *entrophy* dan *gain*

3. Menentukan atribut terbaik

   Atribut terbaik dipilih berdasarkan pemisahan kelas yang paling baik untuk dijadikan sebagai *node* atau simpul

4. Membuat Cabang Pohon (*Tree*)

   Setelah simpul akar dibuat, selanjutnya membuat cabang pohon menggambarkan kelas dari klasifikasi yang dibuat dabn setiap cabang mengarah pada simpul anak

5. *Tree* Optimal

   Apabila *tree optimal* masih dapat dilakukan pencabangan maka kembali ke tahapan menentukan atribut terbaik untuk membuat simpul simpul selanjutnya. Namun, apabila sudah tidak ada lagi pencabangan yang dibuat maka dapat ditentukan hasil klasifikasi *Decision Tree*.

6. Hasil *Decision Tree*

   Hasil berupa gambar pohon klasifikasi yang terdiri dari *node*, cabang, dan daun

7. Selesai

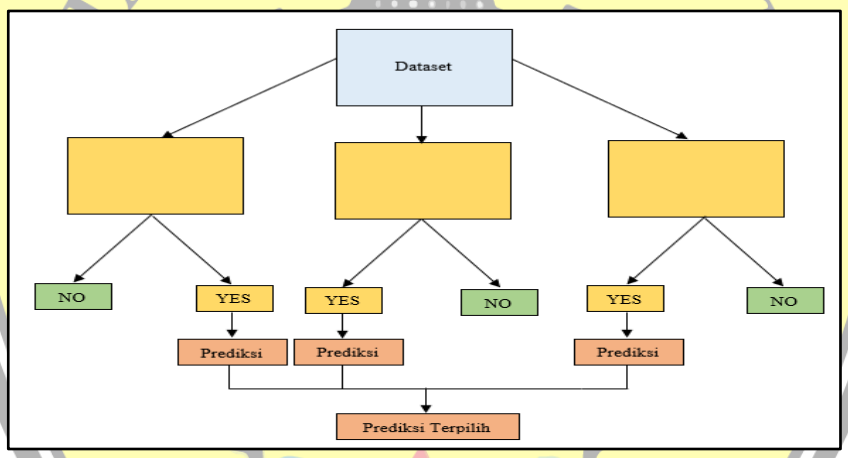
   Pembuatan pohon keputusan atau *Decision Tree* telah selesai dan dapat digunakan untuk menentukan klasifikasi yang sesuai terhadap data baru yang ingin diklasifikasikan.

2.3.2 *Random Forest*

Random Forest diperkenalkan oleh seorang ilmuwan komputer dan inovator *Machine Learning* pada tahun 2001 yaitu Leo Breiman. *Random Forest* merupakan salah satu algoritma *Machine Learning* dalam pendekatan *predictive* dapat digunakan untuk kategori klasifikasi dan regresi. Breiman juga penemu konseop *bagging* yang merupakan dasar dari *Random Forest* yaitu teknik *ensemble* atau menggabungkan hasil dari beberapa model yang dibangun dengan data yang diambil secara acak. *Random Forest* tidak memiliki rumus matematis tunggal dalam menggambarkan seluruh prosesnya dikarenakan melibatkan banyak pohon keputusan yang bekerja bersama-sama maka dalam pembuatan model dilakukan

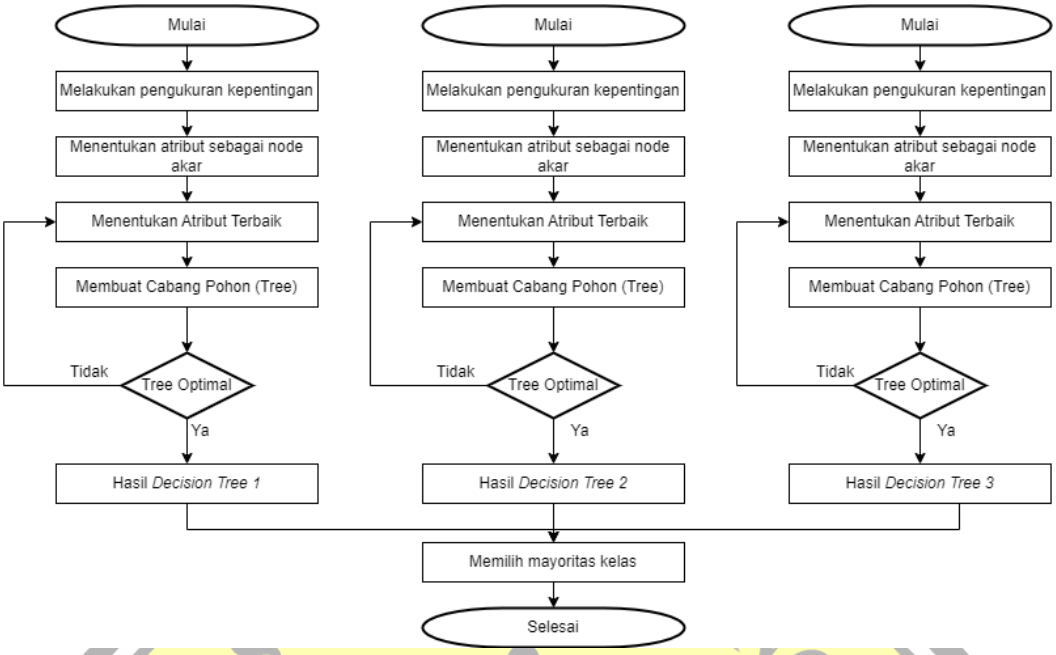
dalam bantuan bahasa pemrograman Python. Hasil dari semua pohon keputusan (*Decision Tree)* akan digabungkan dengan teknik *ensemble* untuk menghasilkan prediksi akhir yang optimal (Hestie, 2008).

Penggunaan algoritma *Random Forest* lebih sering digunakan karena beberapa kelebihannya. Kelebihan algoritma *Random Forest* antara lain mampu menghasilkan klasifikasi yang lebih akurat, mampu mengatasi data yang hilang atau tidak lengkap, mampu mengidentifikasi banyak variabel dengan baik, tidak memerlukan *preprocessing* yang rumit, cocok untuk data dengan jumlah yang besar, hasil cenderung lebih stabil karena memiliki kemampuan menggabungkan model. Kekurangan algoritma *Random Forest* adalah data kelas memiliki perbedaan signifikan mampu mempengaruhi hasil akurasi. Data kelas yang dimaksud merupakan pengelompokkan klasifikasi yang ingin dilakukan (Kember, 2011). Berikut adalah contoh skema dari *Random Forest* (Erlin, 2022).



Gambar 5. Skema *Random Forest*

Berikut *flowchart Random Forest* dalam melakukan klasifikasi adalah sebagai berikut (Sulistyaningrum, 2019).



Gambar 6. *Flowchart Random Forest*

Adapun penjelasan *flowchart Random Forest* adalah sebagai berikut

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Melakukan pengukuran kepentingan

   Menghitung sebarapa baik pengukuran dalam pemisahan kelas dengan perhitungan *entrophy* dan *gain*

3. Menentukan atribut terbaik

   Atribut terbaik dipilih berdasarkan pemisahan kelas yang paling baik untuk dijadikan sebagai *node* atau simpul

4. Membuat Cabang Pohon (*Tree*)

   Setelah simpul akar dibuat, selanjutnya membuat cabang pohon menggambarkan kelas dari klasifikasi yang dibuat dabn setiap cabang mengarah pada simpul anak

5. *Tree* Optimal

   Apabila *tree optimal* masih dapat dilakukan pencabangan maka kembali ke tahapan menentukan atribut terbaik untuk membuat simpul simpul selanjutnya. Namun, apabila sudah tidak ada lagi pencabangan yang dibuat maka dapat ditentukan hasil klasifikasi *Decision Tree*.

6. Hasil *Decision Tree*

   Hasil berupa gambar pohon klasifikasi yang terdiri dari *node*, cabang, dan daun

7. Memilih mayoritas kelas

   Memilih mayoritas kelas dengan menghitung frekuensi kemunculan setiap kelas pada hasil prediksi yang diberikan oleh semua *Decision Trees*. Kelas yang mendapatkan suara terbanyak atau mayoritas dianggap sebagai klasifikasi akhir dari *Random Forest*.

8. Selesai

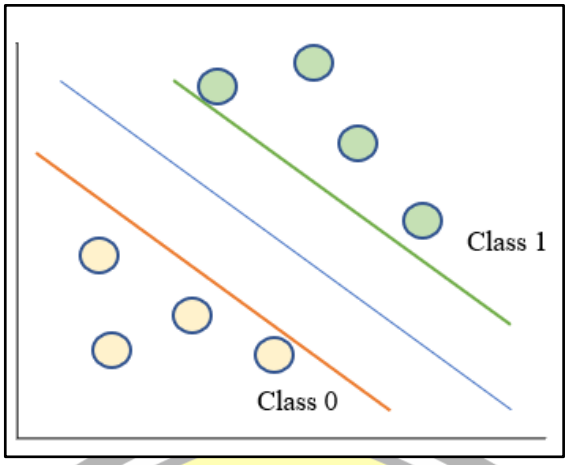
   Langkah *Random Forest* telah selesai dan dapat digunakan untuk menentukan klasifikasi yang sesuai terhadap data baru yang ingin diklasifikasikan.

2.3.3 *Support Vector Machine*

Support Vector Machine adalah algoritma klasifikasi *Machine Learning* yang mampu melakkan klasifikasi dan regresi. *Support Vector Machine* terdapat prinsip dasar *classifier* dalam penggunaannya yaitu kasus klasifikasi secara linier dapat dipisahkan namun pengembangan *Support Vector Machine* dilakukan agar mampu bekerja pada permasalahan non-linier dengan menggabungkan konsep *hyperplane* yaitu memaksimlkan jarak antara kelas data. *Hyperplane* terbaik dapat dicari dengan menggunakan metode *Quadratic Programming Problem* (Pushpita, 2014)

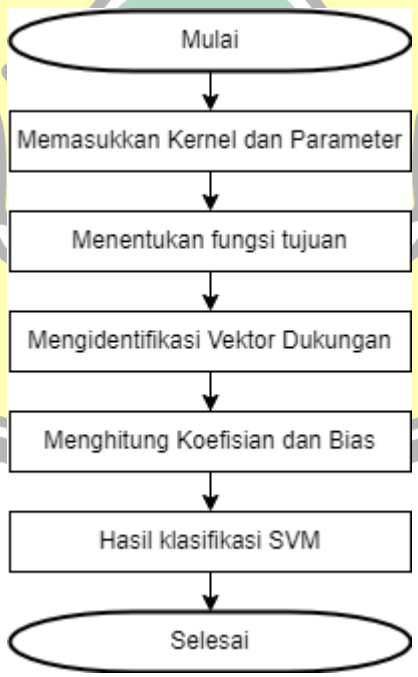
Penggunaan algoritma *Support Vector Machine* memiliki beberapa kelebihan. Kelebihan yang dimiliki *Support Vector Machine* yaitu mampu mempercepat komputasi dengan penentuan jarak *Support Vector Machine* dan mampu memberikan model yang baik dengan penggunaan data relatif sedikit. Kekurangan algoritma *Support Vector Machine* adalah pengaplikasian agak sulit dilakukan untuk jumlah data relatif besar pada beberapa kasus dan membutuhkan tambahan teknik apabila ingin mengklasifikasi lebih dari dua kelas (Arifin. 2021).

Berikut contoh skema *Support Vector Machine* (Husada, 2021).



Gambar 7. Skema *Support Vector Machine*

Berdasarkan Gambar 7. terdapat 3 garis diantara kelas 0 dan 1. Pada proses *Support Vector Machine* mencari *Hyperplane* yaitu jarak margin maksimal antara kedua kelas agar dapat mengklasifikasikan titik data dalam ruang N-dimensi secara jelas yang dijadikan sebagai pembatas. Pada Gambar 7, *Hyperplane* ditandai dengan garis berwarna biru. Garis berwarna jingga dan hijau yaitu garis pembantu untuk membatasi data terluar dari kedua kelas. Berikut *flowchart Support Vector Machine* dalam melakukan klasifikasi adalah sebagai berikut (Octaviani, 2014).



Gambar 8. *Flowchart Support Vector Machine*

Adapun penjelasan *flowchart Support Vector Machine* adalah sebagai berikut (Octaviani, 2014).

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Memasukkan kernel dan parameter

   Kernel dan parameter yang digunakan adalah kernel RBF (*Radial Basis Function*) dan parameter *C* diatur ke nilai 1.0. Kernel digunakan untuk memetakan data ke dimensi yang lebih tinggi dalam memisahkan kelas kelas sedangkan parameter *C* digunakan untuk meminimasi kesalahan dalam klasifikasi.

3. Menentukan fungsi tujuan

   Fungsi tujuan yang ingin dioptimalkan terkait dengan menemukan *hiperplane* dengan memaksimalkan margin dan meminimalkan jumlah kesalahan klasifikasi

4. Mengidentifikasi Vektor Dukungan

   Mengidentifikasi data *training* yang paling dekat dengan *hiperplane* dan berkontribusi pada pembentukan margin

5. Menghitung Koefisian dan Bias

   Koefisien yang dihitung berupa vektor bobot yang menggambarkan arah dan panjang *hiperplane* dan nilai bisa berupa nilai posisi *hiperplane* dalam ruang. Apabila di dalam pustaka pemodelan Scikit-learn, perhitungan ini sudah diintregasikan dalam metode *predict*

6. Hasil Klasifikasi SVM

   Hasil klasifikasi secara manual berupa fungsi keputusan yang digunakan untuk melakukan klasifikasi pada data baru selanjutnya

7. Selesai

   Support Vestor Machine akan menghasilkan sebuah fungsi keputusan yang digunakan untuk klasifikasi data baru. Model berupa fungsi tujuan yang ingin dioptimalkan

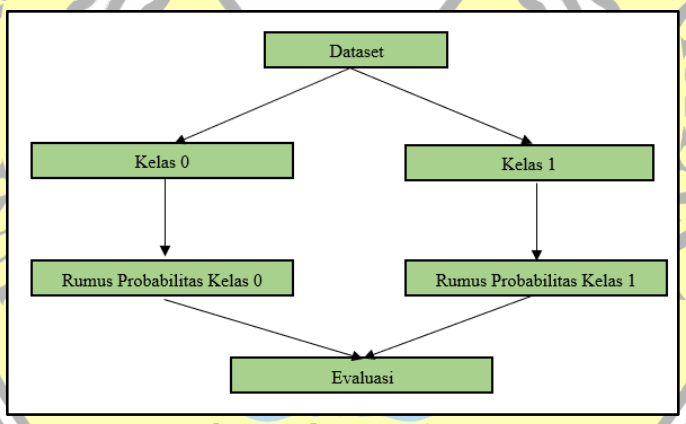
2.3.4 *Naïve Bayes*

Naïve Bayes diperkenalkan oleh seorang ilmuwan Inggris Thomas Bayes. *Naïve Bayes* merupakan algoritma klasifikasi *Machine Learning* untuk memprediksi masa depan menggunakan data historis masa lalu dimana

pengklasifikasian sederhana dengan menghitung sejumlah probababilitas lalu menjumlahkan frekuensi dan kombinasi nilai. *Naive Bayes* menggunakan teorema Bayes dengan mengasumsikan bahwa semua variabel independen dan tidak saling berhubungan dengan variabel kelas. Dasar *Naïve Bayes* dilakukan dengan penyederhanaan nilai variabel saling bebas apabila diberikan nilai *output* (Kawani, 2019).

Penggunaan algoritma *Naïve Bayes* memiliki beberapa kelebihan. Kelebihan yang dimiliki *Naïve Bayes* yaitu cocok diterapkan untuk data dengan jumlah banyak, mampu menangani data yang memeliki *missing value*, mampu menangani variabel yang tidak sama, dapat bekerja dengan baik dalam data pelatihan yang kecil, dan mampu bekerja lebih baik terhadap situasi dunia yang kompleks dan fleksibel. Kekurangan *Naïve Bayes* adalah tidak mampu menghitung besar akurasi kalsfikasi dengan probabilitasnya dan pemilihan variabel yang tidak tepat mampu mempengaruhi nilai akurasi juga (Arifin, 2019).

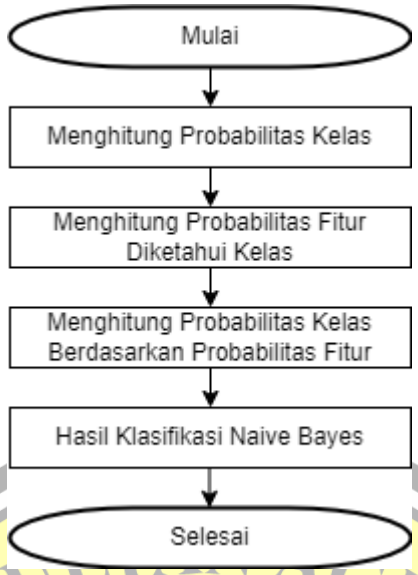
Berikut contoh skema Naïve Bayes



Gambar 9. Skema *Naive Bayes*

Berdasarkan Gambar 9, dataset telah diberikan label berupa kelas seperti kelas 0 dan 1. Masing masing kelas akan dihitung probabilitas nya sesuai dengan rumus yang ada. Lalu dilakukan perbandingan probabilitas hasil kelas 0 dan 1 untuk menentukan nilai evaluasi. Evaluasi dilakukan untuk mengukur kinerja model agar mampu melakukan klasifikasi yang akurat (Muin, 2016).

Berikut *flowchart Naïve Bayes* dalam melakukan klasifikasi adalah sebagai berikut (Ashari, 2016).



Gambar 10. *Flowchart Naive Bayes*

Adapun penjelasan *flowchart Naïve Bayes* adalah sebagai berikut (Ashari, 2016)

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.
2. Menghitung Probabilitas Kelas

   Menghitung seberapa sering suatu kelas muncul dalam data
3. Menghitung Probabilitas Fitur Diketahui Kelas

   Menghitung seberapa sering suatu nilai fitur muncul dalam data jika diketahui bahwa data berada dalam suatu kelas
4. Menghitung Probabilitas Kelas Berdasarkan Probabilitas Fitur

   Menghitung probabilitas kelas berdasarkan probabilitas fitur dengan menggunakan teorema bayes
5. Hasil Klasifikasi *Naïve Bayes*

   Hasil klasifikasi secara manual berupa nilai probabilitas dari kedua kelas dengan fitur yang berbeda lalu dipilih kelas yang nilai probabilitas lebih besar dengan cara membandingkannya
6. Selesai

   Naïve Bayes akan menghasilkan sebuah nilai probabilitas untuk setiap fitur kelas yang dapat digunakan untuk klasifikasi data baru.

2.3.5 *K-Nearest Neighbors Classifier*

K-*Nearest Neighbors Classifier* adalah algoritma klasifikasi Machine Learning untuk mengklafikasikan data berdasarkan penentuan nilai K atau jarak terpendek. K-*Nearest Neighbors Classifier* menggunakan algoritma berbasis memori dengan beberapa iterasi sehingga mampu menemukan parameter terdekat yang dapat digunakan. Jarak minimum data *testing* akan diproses dengan membandingkan data *training* jarak terdekat. K-*Nearest Neighbors Classifier* dapat digunakan pada beberapa permasalahan seperti pengenalan teks, pengenalan objek, pengenalan pola, dan lainnya (Cholil, 2021).

Penggunaan algoritma K-*Nearest Neighbors Classifier* memiliki beberapa kelebihan. Kelebihan yang dimiliki K-*Nearest Neighbors Classifier* yaitu dengan menggunakan satu parameter K dapat dengan mudah diimplementasikan, nilai parameter K tinggi mampu mengurangi efek *noise*, sederhana, mudah dipelajari, dan efektif jika memeiliki data pelatihan yang relatif besar. Kekurangan K-*Nearest Neighbors Classifier* adalah sulit dilakukan pengklasifikasian untuk pola-pola tersebar acak, harus menentukan nilai parameter K terlebih dahulu, dan membutuhkan biaya komputasi yang tinggi karena membutuhkan perhitungan jarak tiap data *training* (Mutrofin, 2019). Berikut *flowchart* K-*Nearest Neighbor* dalam melakukan klasifikasi adalah sebagai berikut (Wijaya, 2021).

Gambar 11. *Flowhart* K-*Nearest Neighbor*

Adapun penjelasan *flowchart* K-*Nearest Neighbor* adalah sebagai berikut (Wijaya, 2021).

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Menentukan Parameter k

   Penentuan nilai parameter k disesuaikan dengan penelitian dan rentang nilai antara 1 hingga 10 mampu memberikan variasi yang cukup baik dan memberikan pemahaman awal terhadap model KNN yang dibuat

3. Menghitung Jarak Euclidean

   Menghitung jarak antara data yang akan diklasifikasi dengan semua data

4. Memilih *Neighbor* Terdekat

   Memilih *neighbor* terdekat dengan memperhatikan urutan jarak Euclidean yang telah dihitung sesuai dengan nilai parameter yang dipilih di awal

5. Melakukan *Majority Voting*

   Dari urutan jarak euclidean dilakukan pemilihan kelas yang paling sering muncul

6. Hasil Klasifikasi KNN

   Hasil klasifikasi secara manual ditentukan dari kelas yang sering muncul pada *majority voting*

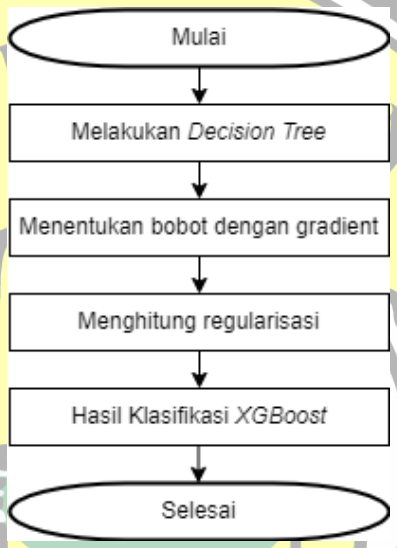
7. Selesai

   K-*Nearest neighbor* telah selesai melakukan klasifikasi.

2.3.6 *XGBoost*

XGBoost merupakan pengembangan oleh salah satu dosen University of Washington pada tahun 2014 dari *Gradient Boosting* yaitu Dr. Tianqi Chen. *Gradient Boosting* merupakan algoritma yang mampu menemukan solusi optimal pada permasalahan klasifikasi atau regresi. Pada *XGBoost* penggunaan model yang lebih teratur mampu membangun struktur pada permahasalahan regresi sehingga hasil kinerja lebih baik dan dapat mengurangi komplektisitas model. Konsep dasar algoritma *XGBoost* merupakan penyesuaian parameter dengan menurunkan *loss function*. Fungsi objektif dibutuhkan dalam *XGBoost* untuk menilai seberapa bagus model apabila telah adanya data latih (Yulianti, 2022).

Penggunaan algoritma *XGBoost* memiliki beberapa kelebihan. Kelebihan yang dimiliki *XGBoost* yaitu efektif digunakan dengan data berskala besar, memiliki beberapa fitur tambahan berupa parameter untuk mencegah *overfitting* dan mempercepat skala perhitungan. Kekurangan *XGBoost* adalah menggunakan banyak parameter mampu mengakibatkan hasil akan sulit fleksibel pada keadaan dunia nyata yang kompleks dan fleksibel dan dengan parameter yang banyak memerlukan waktu komputasi lebih lama dibandingkan algoritma lainnya (Siringoringo, 2021). Berikut *flowchart XGBoost* dalam melakukan klasifikasi adalah sebagai berikut (Elina, 2022).



Gambar 12. *Flowchart XGBoost Classifier*

Adapun penjelasan *flowchart XGBoost* adalah sebagai berikut (Elina, 2022)

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Melakukan *Decision Tree*

   Penggunaan *Decision Tree* dalam *XGBoost* dikarenakan pohon pohon keputusan yang secara berurutan mampu meningkatkan kinerja model.

3. Menentukan Bobot dengan Gradien

   Penggunaan gradient pada penentuan bobot dilakukan dengan perhitungan turunan parsian dari fungsi kerugian (L) untuk meminimalkan fungsi

kerugian dan membantu memperbarui model agar mendekati nilai yang lebih optimal.

4. Menghitung Regularisasi

   Perhitungan regularisasi untuk membatasi kompleksitas model dan mempelajari detail detail kecil dari data.

5. Hasil Klasifikasi *XGBoost*

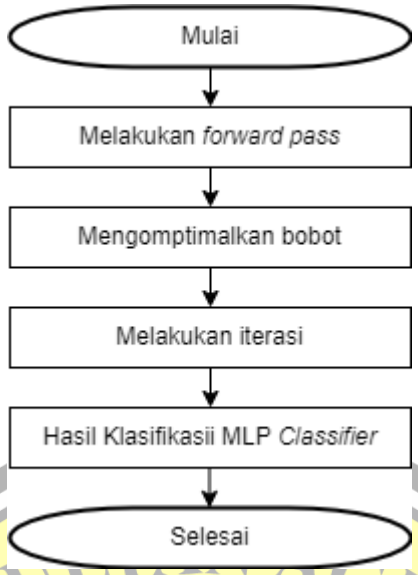
   Hasil klasifikasi secara manual berupa penentuan kelas yang sesuai dengan data

6. Selesai

   XGBoost telah selesai melakukan klasifikasi.

2.3.7 MLP *Classifier*

MLP *Classifier* atau *Multilayer Perceptron* adalah klasifikasi *Machine Learning* yang tersusun dari banyak lapisan *perceptron* dengan menggunakan fungsi aktivasi di setiap lapisan untuk menghasilkan *output* dari beberapa lapisan. Proses MLP *Classifier* menggunakan *backpropagation* dengan menggunakan metode optimasi *gradient descent*. MLP *Classifier* termasuk ke dalam *neural network* dalam algoritma klasifikasi *Machine Learning*. Penggunaan algoritma MLP *Classifier* memiliki beberapa kelebihan. Kelebihan yang dimiliki MLP *Classifier* yaitu mampu memodelkan data yang kompleks, memiliki fleksibilitas dalam tugas klasifikasi biner, multikelas, hingga regresi. Kekurangan penggunaan algoritma MLP *Classifier* yaitu membutuhkan jumlah data yang cukup banyak untuk mencegah *overfitting* dan membutuhkan parameter yang banyak untuk mencapai kinerja yang optimal (Handayani, 2021). Berikut *flowchart* MLP *Classifier* dalam melakukan klasifikasi adalah sebagai berikut (Khoirudin, 2018).



Gambar 13. *Flowchart* MLP *Classifier*

Adapun penjelasan *flowchart* MLP *Classifier* adalah sebagai berikut

1. Mulai

   Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Melakukan *Forward Pass*

   Forward pass atau pemrosesan maju merupakan tahapan membuat tiga lapisan yaitu lapisan input, lapisan tersembunya, dan lapisan *output*.

3. Mengoptimalkan bobot

   Mengoptimalkan bobot dengan menggunakan *backpropagation* atau propagasi balik dengan menentukan gradient kesalahan terhadap bobot menggunakan suatu metrik eror

4. Melakukan iterasi

   Tahapan *forward pass* dan pengoptimalan bobt menggunakan *backpropagation* dilakukan secara berulang ulang melaluui beberapa iterasi untuk mencapai tingkat kinerja yang memadai

5. Hasil Klasifikasi MLP *Classifier*

   Hasil klasifikasi berupa probabilitas untuk setiap kelas yang digunakan untuk melakukan klasifikasi pada data baru selanjutnya

6. Selesai

   MLP *Classifier* telah selesai melakukan klasifikasi

2.3.8 *Logistic Regression*

Logistic Regression adalah klasifikasi *Machine Learning* yang menghitung probabilitas keanggotaan kelas untuk salah satu dari kategori pada kumpulan data. Nilai *input* pada proses *Logistic Regression* digabungkan secara linier menggunakan fungsi sigmoid dengan estimasi kemungkinan data probabilitas diberikan antara 0 sampai 1. Apabila ambang keputusan digunakan maka terjadilah maslaha klasifikasi. Penggunaan algoritma *Logistic Regression* memiliki beberapa kelebihan. Kelebihan yang dimiliki *Logistic Regression* yaitu menghasilkan *output* yang sederhana dan mudah dipahami, jika hubungan variabel *input* dan *output* saling linear maka akan memberikan hasil kinerja yang baik, dan membutuhkan waktu komputasi yang lebih rendah dibandingkan dengan model lainnya. Kekurangan penggunaan algoritma *Logistic Regression* yaitu asumsi linearitas memiliki hubungan yang mempengaruhi hasil sehingga apabila asumsi tidak terpenuhi maka peforma kinerja yang dihasilkan akna buruk dan memiliki keterbatasan apabila ingin mnghasilkan *output* berupa multikelas (Amriza, 2021). Berikut *flowchart Logistic Regression* dalam melakukan klasifikasi adalah sebagai berikut (Putri, 2021).

Gambar 14. *Flowchart Logistic Regression*

Adapun penjelasan *flowchart Logistic Regression* adalah sebagai berikut (Putri, 2021).

1. Mulai

Memulai perhitungan dengan mempersiapkan data yang digunakan untuk melakukan klasifikasi dan telah dilabeli untuk setiap kolomnya.

2. Menentukan koefisien dan *intercept* terbaik

   Penentuan koefisien atau bobot dengan menggunakan nilai acak dari distribusi normal dan penentuan *intercept* atau bias dengan penambahan nilai ke hasil fungsi linear berguna untuk meminimalkan kesalahan prediksi model.

3. Menghitung fungsi logistik

   Fungsi logistik untuk memetakan nilai kombinasi linear dari bobot dan nilai fitur ditambah dengan *intercept* ke dalam probabilitas kelas.

4. Melakukan iterasi

   Tahapan perhitungan fungsi logistik dilakukan secara berulang ulang melaluui beberapa iterasi untuk mencapai tingkat kinerja yang memadai

5. Hasil Klasifikasi *Logistic Regression*

   Hasil klasifikasi berupa probabilitas untuk setiap kelas yang digunakan untuk melakukan klasifikasi pada data baru selanjutnya

6. Selesai

   Logistic Regression telah selesai melakukan klasifikasi