BAB II
LANDASAN TEORI

2.1. *Machine Learning* (ML)

Machine Learning (ML) adalah cabang dari kecerdasan buatan (*Artificial Intelligence*) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. ML bekerja berdasarkan prinsip bahwa sistem dapat belajar dari data, mengenali pola, dan mengambil keputusan dengan intervensi manusia yang minimal. Berbeda dengan pemrograman tradisional yang memerlukan aturan yang jelas dan tetap, ML memberikan fleksibilitas bagi komputer untuk memperbaiki performanya secara otomatis seiring bertambahnya data atau pengalaman.

Proses pembelajaran dalam ML umumnya melibatkan tiga komponen utama, yaitu data pelatihan (training data), model, dan algoritma pembelajaran. Data pelatihan digunakan sebagai sumber informasi agar model dapat mengenali pola atau hubungan antarvariabel. Algoritma pembelajaran berperan untuk menyesuaikan parameter dalam model agar prediksi yang dihasilkan sesuai dengan kenyataan. Semakin banyak dan berkualitas data yang digunakan, maka semakin baik pula akurasi dari model yang dibentuk. Oleh karena itu, data menjadi unsur yang sangat krusial dalam pengembangan model ML.

Dalam implementasinya, ML terbagi ke dalam beberapa jenis pendekatan, di antaranya *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Supervised learning merupakan pendekatan yang paling umum digunakan, di mana

sistem dilatih menggunakan data yang telah diberi label. Dalam pendekatan ini, model belajar dari pasangan input-output dan digunakan untuk memprediksi hasil pada data baru. Sedangkan unsupervised learning digunakan ketika data tidak memiliki label, dan sistem ditugaskan untuk menemukan struktur tersembunyi dalam data. Reinforcement learning, di sisi lain, berfokus pada pembelajaran berbasis umpan balik berupa hadiah atau penalti dalam proses interaksi dengan lingkungan.

Salah satu metode yang sangat populer dalam ML adalah *Artificial Neural Network* (ANN) atau dalam bahasa Indonesia dikenal sebagai *Jaringan Syaraf Tiruan* (JST). JST meniru cara kerja otak manusia dengan membentuk jaringan neuron buatan yang saling terhubung. Setiap neuron memiliki bobot (weight) yang disesuaikan selama proses pelatihan untuk meminimalkan kesalahan antara hasil prediksi dan nilai aktual. JST sangat efektif digunakan untuk memodelkan hubungan yang kompleks dan non-linear dalam data, yang membuatnya cocok digunakan pada banyak kasus seperti pengenalan suara, pengolahan citra, dan juga prediksi numerik.

Salah satu algoritma pembelajaran yang paling sering digunakan dalam JST adalah *Backpropagation*. Algoritma ini bekerja dengan menghitung kesalahan dari hasil prediksi model, lalu menyebarkan kesalahan tersebut kembali ke seluruh jaringan untuk memperbarui bobot secara bertahap. Proses ini dilakukan berulang-ulang melalui metode pelatihan yang dikenal sebagai *gradient descent*, hingga model mencapai tingkat kesalahan yang seminimal mungkin. Dengan pendekatan

ini, JST menjadi semakin presisi dalam mengolah data dan menghasilkan prediksi yang akurat.

Dalam konteks penelitian ini, Machine Learning diterapkan untuk memprediksi jumlah penduduk di Kabupaten Labuhanbatu dengan menggunakan algoritma Backpropagation dalam JST. Prediksi jumlah penduduk merupakan permasalahan regresi yang menuntut model mampu mengenali tren pertumbuhan dari data historis. Data yang digunakan adalah data jumlah penduduk per kecamatan dari tahun 2018 hingga 2023 yang diperoleh dari BPS. Dengan menggunakan pendekatan ML, diharapkan sistem mampu memahami pola pertumbuhan populasi yang mungkin tidak terdeteksi melalui pendekatan statistik konvensional.

Penerapan JST dalam penelitian ini sangat relevan mengingat model ini mampu menangkap hubungan non-linier antara waktu dan pertumbuhan jumlah penduduk. Algoritma Backpropagation berperan penting dalam mengoptimalkan model agar kesalahan prediksi dapat diminimalkan. Dengan mencoba beberapa variasi arsitektur JST—misalnya jumlah neuron tersembunyi atau jumlah lapisan tersembunyi—penelitian ini mengevaluasi mana konfigurasi model yang memberikan tingkat akurasi prediksi terbaik. Hal ini menjadi penting untuk memastikan model tidak hanya akurat terhadap data pelatihan, tetapi juga mampu melakukan generalisasi pada data di masa depan.

Hasil dari model prediksi jumlah penduduk ini tidak hanya bermanfaat dari sisi akademik, tetapi juga memberikan kontribusi praktis bagi pemerintah daerah. Dengan memperoleh prediksi jumlah penduduk yang lebih akurat, pemerintah Kabupaten Labuhanbatu dapat merencanakan kebijakan yang lebih terukur dan

tepat sasaran di berbagai sektor seperti pendidikan, kesehatan, perumahan, dan infrastruktur. Oleh karena itu, integrasi metode Machine Learning dalam penelitian ini diharapkan menjadi solusi inovatif dalam menyelesaikan persoalan perencanaan berbasis data yang semakin kompleks.

2.2. Jaringan Saraf Tiruan (JST)

Jaringan Syaraf Tiruan (JST) atau *Artificial Neural Network* (ANN) adalah salah satu pendekatan dalam kecerdasan buatan yang terinspirasi dari cara kerja otak manusia. JST terdiri dari sejumlah unit pemrosesan kecil yang disebut neuron buatan, yang saling terhubung dan bekerja secara paralel untuk memproses informasi. Neuron-neuron ini tersusun dalam lapisan-lapisan, yaitu lapisan input, lapisan tersembunyi (hidden layer), dan lapisan output. Setiap koneksi antar neuron memiliki bobot yang dapat disesuaikan selama proses pelatihan, dengan tujuan agar jaringan mampu menghasilkan output yang sesuai dengan target yang diinginkan. JST bekerja dengan prinsip pembelajaran melalui data, yaitu dengan mengidentifikasi pola dalam data input dan menyesuaikan bobot koneksi secara iteratif untuk meminimalkan kesalahan hasil prediksi.

Salah satu keunggulan utama JST adalah kemampuannya dalam memodelkan hubungan non-linear yang kompleks, yang tidak dapat ditangani secara optimal oleh metode statistik tradisional seperti regresi linear. Oleh karena itu, JST banyak digunakan dalam berbagai bidang, termasuk pengenalan suara, klasifikasi gambar, pengolahan bahasa alami, serta prediksi numerik dan deret waktu.

Dalam penelitian ini, JST digunakan untuk memprediksi jumlah penduduk di Kabupaten Labuhanbatu dengan memanfaatkan data historis dari tahun 2018

hingga 2023. Penggunaan JST menjadi sangat relevan mengingat pola pertumbuhan penduduk tidak selalu linier dan bisa dipengaruhi oleh banyak faktor yang saling berkaitan. JST mampu menangkap kompleksitas data tersebut dan memprosesnya untuk membentuk model prediktif yang lebih akurat dibandingkan metode tradisional. Dengan bantuan algoritma Backpropagation, proses pelatihan jaringan menjadi lebih efisien karena bobot neuron diperbarui secara sistematis berdasarkan kesalahan output yang dihasilkan.

Penelitian ini juga mengeksplorasi berbagai arsitektur JST untuk menemukan konfigurasi model terbaik dalam hal akurasi prediksi jumlah penduduk. Dengan menguji beberapa kombinasi jumlah neuron tersembunyi dan jumlah lapisan, peneliti dapat mengevaluasi bagaimana struktur jaringan mempengaruhi performa model. Hasil dari prediksi ini diharapkan dapat memberikan informasi yang akurat dan bermanfaat bagi pemerintah daerah dalam menyusun kebijakan pembangunan yang berkelanjutan. Dengan demikian, pemanfaatan JST dalam penelitian ini tidak hanya menunjukkan kekuatan teknologi dalam pengolahan data, tetapi juga kontribusi nyata dalam mendukung perencanaan berbasis data.

2.3. *Backpropagation*

2.4.1. Pengertian *Backpropagation*

Backpropagation adalah suatu model dari Jaringan Saraf Tiruan (JST). Model *Backpropagation* ini menggunakan pembelajaran terawasi untuk memecahkan masalah yang kompleks dan dilatih menggunakan metode pembelajaran (Matondang, 2013). Metode tersebut terdiri dari proses pembelajaran maju dan koreksi kesalahan mundur. Model jaringan *Backpropagation* banyak digunakan dalam proses prediksi, identifikasi dan peramalan (Candra Dewi, 2013).

Backpropagation pertama kali diterbitkan oleh Paul Werbos pada tahun 1974 memperkenalkan untuk melatih perceptron dengan banyak lapisan. Kemudian dirumuskan ulang oleh David Parker pada tahun 1982, dan dipopulerkan oleh Rumelhar dan McCelland pada tahun 1986.

2.4.2. Komponen *Backpropagation*

Berdasarkan Gambar 2.1, dapat dilihat bahwa arsitektur algoritma *Backpropagation* memiliki beberapa komponen yaitu:

1. Lapisan Masukan (*Input Layer*)

Input layer yaitu layer atau lapisan yang berisi neuron-neuron masukan bagi JST *Backpropagation*. Banyaknya neuron yang digunakan disesuaikan dengan banyaknya fitur data yang akan diproses. Neuron input layer pada arsitektur JST *Backpropagation* disimbolkan dengan Xi dengan i = 1, 2, … , n dimana n yaitu banyak neuron di input layer. Banyaknya neuron di input layer sesuai dengan banyaknya fitur atau kelompok pada data yang akan digunakan, jika terdapat tujuh fitur atau kelompok pada data masukan, maka jumlah neuron di input layer yaitu sebanyak tujuh neuron.

2. Lapisan Tersembunyi (*Hidden Layer*)

Hidden layer merupakan layer atau lapisan yang menjadi penghubung antara input layer dengan output layer. Lapisan ini berisi neuron-neuron hidden layer yang disimbolkan dengan Zj dengan j = 1, 2, … , m dimana m adalah banyaknya neuron di hidden layer. Menurut Heaton (2008), terdapat beberapa metode untuk menentukan banyak neuron di hidden layer yaitu sebagai berikut:

1. Banyak neuron tersembunyi harus berada di antara ukuran input layer dan ukuran output layer.

2. Banyak neuron tersembunyi harus $\frac{2}{3}$ dari ukuran input layer, ditambah output layer.
3. Banyak neuron tersembunyi harus kurang dari dua kali ukuran input layer

Jika banyak neuron yang dipilih lebih sedikit, hal ini akan menyebabkan *underfitting* dan bias statistik yang tinggi. Sedangkan jika kita memilih terlalu banyak neuron, hal ini dapat menyebabkan *overfitting*, varians tinggi, dan meningkatkan waktu yang diperlukan untuk melatih jaringan.

3. Lapisan Keluaran (*Output Layer*)

Output layer merupakan layer atau lapisan yang berisi neuron-neuron untuk keluaran JST *Backpropagation*. Neuron-neuron ini disimbolkan dengan Yk yaitu dengan $k = 1, 2, \ldots, p$ dimana p adalah banyaknya neuron di output layer. Banyaknya neuron di output layer tergantung pada jenis masalah yang ingin dipecahkan. Misalnya, jika masalah yang ingin dipecahkan adalah klasifikasi gambar menjadi 10 kategori, maka output layer akan terdiri dari 10 neuron.

4. Bobot V

Untuk setiap neuron pada input layer dan hidden layer akan dihubungkan dengan bobot dan bias. Bobot V yaitu bobot yang menjadi penghubung antara neuron-neuron pada input layer dan hidden layer. Bias merupakan komponen yang berfungsi untuk mempercepat laju pembelajaran, bobot boleh diabaikan atau ditiadakan. Bias V biasanya disimbolkan dengan $V0$. Banyaknya bobot V tergantung pada banyak neuron di input layer dan hidden layer yaitu.

$$bobot\ V = \mathrm{n}_{\mathrm{input}} \times \mathrm{n}_{\mathrm{hidden}}$$

Apabila menggunakan bias, maka banyaknya bobot V yaitu

$$bobot\ V = (n_{input} + 1) \times n_{hidden}$$

keterangan:

n_{input} : Banyak neuron pada *input layer*

n_{hidden} : Banyak neuron pada *hidden layer*

5. Bobot W

Bobot W yaitu bobot yang menjadi penghubung antara neuron-neuron yang terdapat pada hidden layer dan output layer. Bias W biasanya disimbolkan dengan $W0$. Banyaknya bobot W tergantung pada banyak neuron di input layer dan hidden layer yaitu.

$$bobot\ W = n_{hidden} \times n_{output}$$

Apabila menggunakan bias, maka banyaknya bobot W yaitu

$$bobot\ W = (n_{hidden} + 1) \times n_{output}$$

keterangan:

n_{hidden} : Banyak neuron pada *hidden layer*

n_{output} : Banyak neuron pada *output layer*

2.4.3. Algoritma *Backpropagation*

Algoritma *Backpropagation* adalah algoritma yang digunakan untuk melatih jaringan saraf tiruan, khususnya dalam menghitung gradien dari fungsi kerugian terhadap bobot jaringan, yang kemudian digunakan untuk memperbarui bobot tersebut. Algoritma yang digunakan yaitu dengan menyesuaikan bobot berdasarkan selisih keluaran dan target yang diinginkan untuk mengurangi nilai error. Kesalahan atau perbedaan antara nilai target dengan data sebenarnya akan ditanggung oleh neuron masukan. Bobot tersebut kemudian dihitung kembali, neuron mengarahkan

sinyal ke hidden layer neuron keluaran, dan proses penghitungan error antara target dan data aktual diulangi lagi hingga error mencapai batas yang ditentukan di awal.

Penentuan bobot dilakukan dengan memilih angka random dengan rentang dari 0 dan 1, atau -1 dan 1. Angka tersebut dipilih dari rentang yang kecil agar nilai bobot awal tidak terlalu besar, sehingga membantu dalam memastikan bahwa jaringan neural mulai pelatihan dari kondisi yang lebih seimbang. Ini bisa mempercepat konvergensi algoritma pembelajaran. Ketika bobot awal terlalu besar, jaringan bisa terjebak dalam pola pembelajaran yang tidak efisien.

Algoritma *Backpropagation* dapat digunakan untuk menyelesaikan permasalahan yang rumit dan kompleks, yaitu dengan menggunakan pelatihan *multi-layer perceptron* (MLP) untuk proses pelatihan input dan output dengan tujuan menghitung bobot untuk mencapai hasil yang diinginkan. Algoritma ini menggunakan lebih sedikit memori dibandingkan dengan algoritma lainnya. Algoritma ini menerima suatu nilai masukan dan melakukan perhitungan berdasarkan bobot yang diperoleh secara acak yang diterima di awal. Apabila nilai keluaran masih belum sesuai dengan nilai target yang diinginkan, maka bobot yang ada pada saat itu disesuaikan [24]. Proses ini berlanjut hingga nilai keluaran sesuai dengan nilai target yang diinginkan. Proses pembelajaran juga terhenti jika keluaran dan target mencapai nilai bobot atau tingkat kesalahan atau nilai toleransi lebih kecil.

Algoritma *Backpropagation* menggunakan fungsi aktivasi pada hidden layer dengan tujuan untuk mengurangi nilai error dari output yang dihasilkan neuron. Fungsi sigmoid, Tangen hyperbolicus (Tanh), dan Rectified Linear Unit (ReLu)

adalah beberapa fungsi aktivasi yang paling umum digunakan pada algoritma *Backpropagation*. Dalam penelitian ini fungsi aktivasi yang akan digunakan yaitu ReLu dan Tanh, karena fungsi ReLu dan Tanh mampu memecahkan masalah non-linier, serta memerlukan sumber daya komputasi yang jauh lebih sedikit. Berikut yaitu rumus dari fungsi ReLu dan Tanh.

1. ReLu

$$f(x) = \max(0,x) \quad \text{.....(1)}$$

2. Tanh

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad \text{.....(2)}$$

Keterangan:

$f(x)$: Hasil Aktivasi

x : Nilai *input*

Pada algoritma *Backpropagation* juga digunakan fungsi optimisasi untuk memperbarui bobot secara iteratif yang didasarkan pada data training. Tujuan dari fungsi optimisasi yaitu untuk meminimalkan kerugian dengan proses training. Terdapat beberapa fungsi optimisasi yang biasa digunakan pada *Neural Network* yaitu Stochastic Gradient Descent (SGD), dan Adapting Moment Estimation (Adam). Pada penelitian ini akan menggunakan fungsi optimisasi Adam karena fungsi ini merupakan kombinasi antara RMSprop dan SGD, sehingga hasil dari fungsi ini lebih baik dibanding dengan kedua fungsi sebelumnya. Pada umumnya nilai Adam yang digunakan yaitu nilai default sebesar 0,0001

2.4.4. Tahapan *Backpropagation*

Terdapat beberapa tahap untuk menggunakan algoritma *Backpropagation* yaitu (R Sovia, 2018).

1. Tahap inisialisasi

   Langkah ini merupakan langkah untuk menentukan nilai awal bagi variabel yang dibutuhkan.

2. Tahap Aktivasi

   Menghitung keluaran aktual pada *hidden layer* dan keluaran aktual pada *output layer* adalah dua tugas yang dilakukan pada langkah ini.

3. Tahap *weight training*

   Pada langkah ini, dua tugas dilakukan yaitu menghitung *gradient error* pada *output layer* dan menghitung *gradient error* pada *hidden layer.*

4. Tahap iterasi

   Pada langkah terakhir ini, pengujian dilakukan. Jika tidak ditemukan kesalahan, maka akan kembali ke tahap kedua yaitu aktivasi.

Hamdan Wadi (2021) menyatakan bahwa proses prediksi data dengan algoritma *Backpropagation* memerlukan dua tahap: pelatihan dan pengujian.

1. Proses Pelatihan

Tujuan dari proses pelatihan adalah untuk mengenali pola pada data dengan melatih bobot agar dapat mengenalinya. Proses ini menghasilkan bias yang terlatih, bobot V, dan bobot W. Dalam proses ini terdapat dua tahap yaitu propagasi maju dan propagasi mundur. Proses pelatihan diartikan sebagai perulangan terhadap propagasi maju dan propagasi mundur dengan tujuan memperoleh bobot V, bobot W, dan bias terlatih.

2. Proses Pengujian

Pengujian adalah proses klasifikasi dan prediksi data uji atau data yang akan datang. Dalam proses ini, algoritma *Backpropagation* hanya melakukan propagasi maju.

Tahapan *Backpropagation*, menurut Andrian dan Wahyadi (2014), adalah sebagai berikut.

1. Langkah 1: Menentukan epoch, target error, tingkat pembelajaran, dan inisialisasi bobot random
2. Langkah 2: Inisialisasi kondisi berhenti. Kondisi berhenti dimulai dengan iterasi (epoch < epoch maksimum) atau minimum error.
3. Langkah 3: Melakukan langkah 4-10 pada setiap data training, Tahap I: Feed Forward
4. Langkah 4: Sinyal input Xi diterima oleh setiap neuron input Xi dengan i = 1,2,3, … n dan didistribusikan ke neuron hidden layer.
5. Langkah 5: Sinyal input terbobot dan biasnya dijumlahkan oleh setiap neuron tersembunyi Zj dengan j = 1,2,3, … m dengan menggunakan Persamaan dibawah.

$$Z_{input\,j} = v_{0j} + \sum_{i=1}^{n} X_i \; V_{ij} \quad \text{.....(3)}$$

Keterangan

$Z_{input\,j}$: Sinyal informasi dari input layer ke neuron hidden layer ke-j

v_{0j} : Bias pada hidden layer ke-j

X_i : Neuron input layer ke-i

V_{ij} : Bobot yang menghuungkan neuron input layer ke-i dan neuron hidden layer ke-j

Dengan menggunakan fungsi aktivasi, dihitung sinyal output-nya mengikuti rumus pada Persamaan dibawah.

$$Z_j = f\,(z_{input\ j}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Keterangan

Z_j : Neuron hidden layer ke-j

$f\,(z_{input\ j})$: Fungsi aktivasi terhadap nilai $z_{input\ j}$

Unit output kemudian menerima sinyal dari seluruh neuron input.

6. Langkah 6: Unit Yk, yang memiliki nilai k = 1,2,3, . . . p, menerima sinyal output dari hidden layer, dan kemudian menjumlahkan sinyal output yang terbobot dan biasnya dengan menggunakan Persamaan dibawah.

$$Y_{input\ k} = W_{0k} + \sum_{j=1}^{n} Z_j W_{jk} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Keterangan

$Y_{input\ k}$: Sinyal informasi dari hidden layer ke neuron output layer ke-k

W_{0k} : Bias pada neuron output layer ke-k

Z_j : Neuron hidden layer ke-j

W_{jk} : Bobot yang menghubungkan neuron hidden layer ke-j dan neuron output layer ke-k

Hitung sinyal output dengan menggunakan fungsi aktivasi yang telah ditentukan mmengikuti rumus dari Persamaan dibawah.

$$Y_{output\ k} = f\,(Y_{input\ k}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Keterangan

$Y_{output\ k}$: Keluaran dari nilai $Y_{input\ k}$

$f(Y_{\text{input k}})$: Fungsi aktivasi terhadap nilai $Y_{\text{input k}}$

Setelah itu, sinyal yang dihasilkan ditransmisikan ke seluruh unit di layer atasnya.

Tahap II: *Feed Backward*

7. Langkah 7: Untuk menghitung kesalahan error pada output unit, setiap unit Y output k dengan k = 1,2,3, . . . p menerima output yang diinginkan, yang sesuai dengan input data pelatihan dengan menggunakan rumus pada Persamaan dibawah.

$$\delta_k = (t_k - Y_{\text{output k}})\, Y_{\text{output k}}\, (1 - Y_{\text{output k}}) \quad \text{.....(7)}$$

Keterangan

δ_k : Faktor koreksi dari neuron output layer ke-k

t_k : Target output pada neuron output layer ke-k

$Y_{\text{output k}}$: Keluaran pada output unit

8. Kemudian hitung faktor koreksi bobot (yang akan digunakan untuk memperbarui bobot baru). Faktor error δk digunakan untuk mengoreksi nilai error pada bobot antara unit tersembunyi dan unit keluaran ΔWjk yang nantinya digunakan untuk memperbarui bobot Wjk dengan menggunakan Persamaan dibawah.

$$\Delta W_{jk} = \alpha \delta_k Z_j \quad \text{.....(8)}$$

Keterangan

ΔW_{jk} : Koreksi bobot yang menghubungkan antara neuron output layer ke-k dan neuron hidden layer ke-j

α : Tingkat pembelajaran (learning rate)

Langkah 8: Setiap unit tersembunyi Zj dengan j = 1,2,3, … . m menjumlahkan input delta yang sudah terbobot seperti pada Persamaan dibawah.

$$\delta_{input\,j} = \sum_{k=1}^{m} \delta_k W_{jk} \text{.....(9)}$$

Keterangan

$\delta_{input\,j}$: Sinyal faktor koreksi dari output layer ke neuron hidden layer ke-j Selanjutnya, hasil delta input dikalikan dengan turunan fungsi aktivasi yang telah ditentukan dan menghasilkan faktor koreksi error δ_j seperti pada Persamaan dibawah.

$$\delta_j = \delta_{input\ j} Z_j \left(1 - Z_j\right) \text{.....(10)}$$

Keterangan

δ_j : Faktor koreksi dari neuron hidden layer ke-j

Hitung koreksi error ΔV_{ji} yang nantinya akan digunakan untuk memperbarui V_{ji} dengan menggunakan rumus pada Persamaan dibawah.

$$\Delta V_{ji} = \alpha \delta_j X_i \text{.....(11)}$$

Keterangan

ΔV_{ji} : Koreksi bobot yang menghubungkan antara neuron hidden layer ke-j dan neuron input layer ke-i

Tahap III: Perubahan Bobot dan Bias

Langkah 9:

Bobot dan bias diperbarui untuk setiap unit output Y_k yang memiliki k = 1,2,3, . . . , p dengan menggunakan rumus pada Persamaan dibawah.

$$W_{jk_{(baru)}} = W_{jk_{(lama)}} + \Delta W_{jk} \text{.....} (12)$$

Bobot dan bias diperbarui untuk setiap unit tersembunyi Z_j (j = 1,2,3, . . . , m) dengan menggunakan rumus pada Persamaan dibawah.

$$V_{ij_{(baru)}} = V_{ij_{(lama)}} + \Delta V_{ij} \text{.....} (13)$$

Langkah 10:

Tes kondisi berhenti. Iterasi akan berhenti ketika hasil keluaran sudah memenuhi target atau nilai error dari hasil prediksi dan target sudah melebihi batas inisialisasi error.

2.4. Metode *Neural Network*

Neural Network, atau jaringan saraf tiruan, adalah salah satu metode dalam kecerdasan buatan yang meniru cara kerja otak manusia dalam memproses informasi. Metode ini terdiri dari sejumlah node atau "neuron" yang tersusun dalam lapisan-lapisan (*layer*), yaitu input layer, hidden layer, dan output layer. Setiap neuron saling terhubung melalui bobot (*weights*) yang dapat disesuaikan selama proses pelatihan (*training*). *Neural Network* mampu mempelajari pola dan hubungan kompleks dalam data melalui proses pembelajaran berbasis data latih, sehingga sangat efektif untuk tugas-tugas klasifikasi, prediksi, dan pengenalan pola.

Dalam praktiknya, metode *Neural Network* sering digunakan dalam berbagai bidang seperti pengenalan suara, pengenalan wajah, deteksi penyakit, hingga peramalan pasar. Keunggulan utamanya adalah kemampuannya untuk menangani

data nonlinear dan menghasilkan model yang fleksibel serta akurat. Namun, metode ini juga memiliki kekurangan, seperti kebutuhan terhadap data dalam jumlah besar, waktu pelatihan yang lama, dan kesulitan dalam menjelaskan logika keputusan yang dihasilkan (*black box*). Meskipun begitu, dengan perkembangan teknologi dan peningkatan daya komputasi, *Neural Network* terus berkembang dan menjadi fondasi utama dalam pengembangan kecerdasan buatan modern, termasuk *Deep Learning*.

2.5. Model Klasifikasi

Model klasifikasi merupakan salah satu metode dalam pembelajaran mesin (*Machine Learning*) yang digunakan untuk memprediksi label atau kategori dari suatu data berdasarkan pola yang telah dipelajari dari data latih. Dalam prosesnya, model ini mempelajari hubungan antara fitur-fitur atau atribut-atribut dari data dengan kelas target, sehingga dapat digunakan untuk mengklasifikasikan data baru ke dalam salah satu kelas yang telah ditentukan. Model klasifikasi sering diterapkan dalam berbagai bidang, seperti deteksi spam, diagnosis medis, pengenalan wajah, dan analisis perilaku konsumen.

Beberapa algoritma yang umum digunakan dalam model klasifikasi antara lain adalah *Decision Tree* (pohon keputusan), *Naive Bayes, K-Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), dan algoritma berbasis ensemble seperti *Random Forest* dan *Gradient Boosting*. Kinerja model klasifikasi biasanya diukur menggunakan metrik seperti akurasi, presisi, *recall*, dan *F1-score*. Pemilihan algoritma yang tepat serta proses preprocessing data yang baik sangat memengaruhi keakuratan hasil klasifikasi. Oleh karena itu, pemahaman yang mendalam mengenai

karakteristik data dan tujuan klasifikasi menjadi hal penting dalam membangun model yang efektif dan andal.

2.6. Alat Bantu Program Aplikasi Orange

Orange adalah sebuah alat bantu program aplikasi berbasis visual yang dirancang untuk analisis data dan pembelajaran mesin (*Machine Learning*). Aplikasi ini menyediakan antarmuka grafis yang memudahkan pengguna dalam membangun alur kerja analisis data tanpa harus menulis kode secara manual. Orange sangat populer di kalangan akademisi, peneliti, dan praktisi data karena menyediakan berbagai komponen (*widget*) untuk visualisasi data, klasifikasi, regresi, clustering, dan pemrosesan data secara interaktif. Dengan sistem drag-and-drop, pengguna dapat menyusun proses analisis secara logis dan cepat, menjadikan Orange sebagai alat yang ideal untuk pembelajaran dan eksperimen data.

Salah satu keunggulan Orange adalah kemampuannya dalam mengintegrasikan berbagai algoritma *Machine Learning* seperti *Naive Bayes*, *Decision Tree (C4.5)*, *k-Nearest Neighbor*, dan lainnya hanya dengan beberapa klik. Selain itu, Orange juga mendukung visualisasi hasil analisis dalam bentuk grafik dan tabel yang informatif, sehingga memudahkan interpretasi data dan pengambilan keputusan. Dengan dukungan pustaka Python di latar belakang, Orange juga dapat diperluas kemampuannya untuk pengguna tingkat lanjut yang ingin menggabungkan analisis visual dengan skrip pemrograman. Kombinasi kemudahan penggunaan dan fleksibilitas membuat Orange menjadi pilihan yang menarik dalam dunia data science dan pendidikan.

2.7. Kelebihan dan Kekurangan Metode *Neural Network*

Metode *Neural Network*, khususnya Artificial *Neural Network* (ANN), memiliki sejumlah kelebihan yang menjadikannya populer dalam berbagai bidang seperti pengenalan pola, prediksi, klasifikasi, dan pengolahan citra. Salah satu kelebihan utamanya adalah kemampuannya dalam mempelajari pola-pola kompleks dan *non-linear* dari data. *Neural Network* dapat membangun model dari data yang tidak memiliki hubungan linier secara eksplisit, yang mana metode statistik konvensional sering kali kesulitan dalam mengenalinya. Hal ini membuat *Neural Network* sangat efektif digunakan dalam sistem yang membutuhkan pemrosesan informasi kompleks, seperti pengenalan suara, pengenalan wajah, dan analisis gambar medis.

Selain itu, metode ini memiliki kemampuan pembelajaran adaptif, yaitu dapat menyesuaikan bobot-bobot internalnya berdasarkan data pelatihan yang diberikan. Semakin banyak data yang digunakan, biasanya performa jaringan akan semakin baik karena jaringan dapat mengenali lebih banyak pola. *Neural Network* juga bersifat toleran terhadap kesalahan dan noise dalam data, sehingga masih mampu memberikan output yang akurat walaupun data masukan tidak sepenuhnya bersih. Keunggulan lainnya adalah kemampuan generalisasi yang cukup baik, artinya jaringan dapat memprediksi output dari data baru yang belum pernah dilihat sebelumnya, asalkan proses pelatihan dilakukan dengan benar dan tidak terjadi *overfitting*.

Meskipun memiliki banyak kelebihan, *Neural Network* juga memiliki beberapa kekurangan yang perlu dipertimbangkan sebelum diterapkan dalam suatu

permasalahan. Salah satu kekurangan utama adalah kebutuhan terhadap jumlah data yang sangat besar untuk menghasilkan model yang akurat dan andal. Tanpa jumlah data pelatihan yang cukup, model *Neural Network* cenderung mengalami *underfitting* atau tidak mampu mengenali pola dengan baik. Selain itu, proses pelatihan jaringan sering kali memerlukan waktu yang lama dan memerlukan daya komputasi yang tinggi, terutama untuk jaringan yang memiliki banyak lapisan (deep *Neural Network*s), sehingga membutuhkan perangkat keras khusus seperti GPU.

Kelemahan lainnya adalah sifatnya yang seperti "*black box*", artinya sulit untuk menafsirkan bagaimana jaringan mengambil keputusan berdasarkan input yang diberikan. Ini menjadi tantangan dalam bidang yang memerlukan transparansi dan penjelasan, seperti bidang medis dan hukum. Selain itu, model *Neural Network* sangat sensitif terhadap pemilihan parameter seperti jumlah neuron, jumlah lapisan, fungsi aktivasi, dan *learning rate*. Kesalahan dalam menentukan parameter dapat menyebabkan performa model menurun secara drastis. Oleh karena itu, meskipun *Neural Network* menawarkan kemampuan analisis data yang canggih, penggunaannya memerlukan pemahaman yang baik mengenai struktur dan prinsip kerja jaringan agar hasil yang diperoleh optimal.

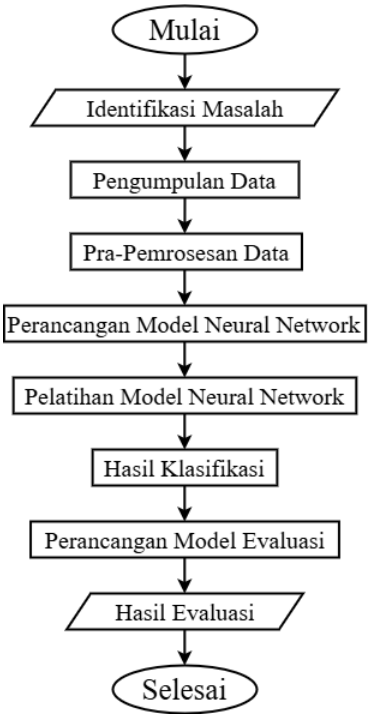
2.8. Evaluasi Model *Neural Network*

Evaluasi metode *Neural Network* dilakukan untuk menilai seberapa baik model yang dibangun mampu mempelajari dan memprediksi data secara akurat. Proses evaluasi ini umumnya dilakukan dengan membagi data menjadi dua bagian, yaitu data pelatihan (*training data*) dan data pengujian (*testing data*). Selama

pelatihan, model belajar dari data yang diberikan, sedangkan pada tahap pengujian, model diuji kemampuannya dalam memprediksi data yang belum pernah dilihat sebelumnya. Beberapa metrik yang sering digunakan untuk mengevaluasi performa *Neural Network* antara lain akurasi, presisi*, recall, F1-score*, dan *Mean Squared Error* (MSE) tergantung pada jenis masalahnya, apakah klasifikasi atau regresi.

Selain metrik kuantitatif, evaluasi juga mencakup analisis terhadap *overfitting* dan *underfitting*. *Overfitting* terjadi ketika model terlalu baik dalam mengenali data pelatihan namun gagal dalam menggeneralisasi data baru, sedangkan *underfitting* terjadi ketika model tidak mampu mengenali pola dalam data pelatihan itu sendiri. Untuk mencegah kedua hal tersebut, teknik seperti validasi silang (cross-validation), regularisasi, dan penggunaan data yang seimbang sangat penting dalam proses evaluasi.

2.9. Kerangka Kerja Penelitian



Gambar 2. 1. Kerangka Kerja Penelitian