# BAB III
# METODOLOGI PENELITIAN

## 3.1 Pengumpulan Data

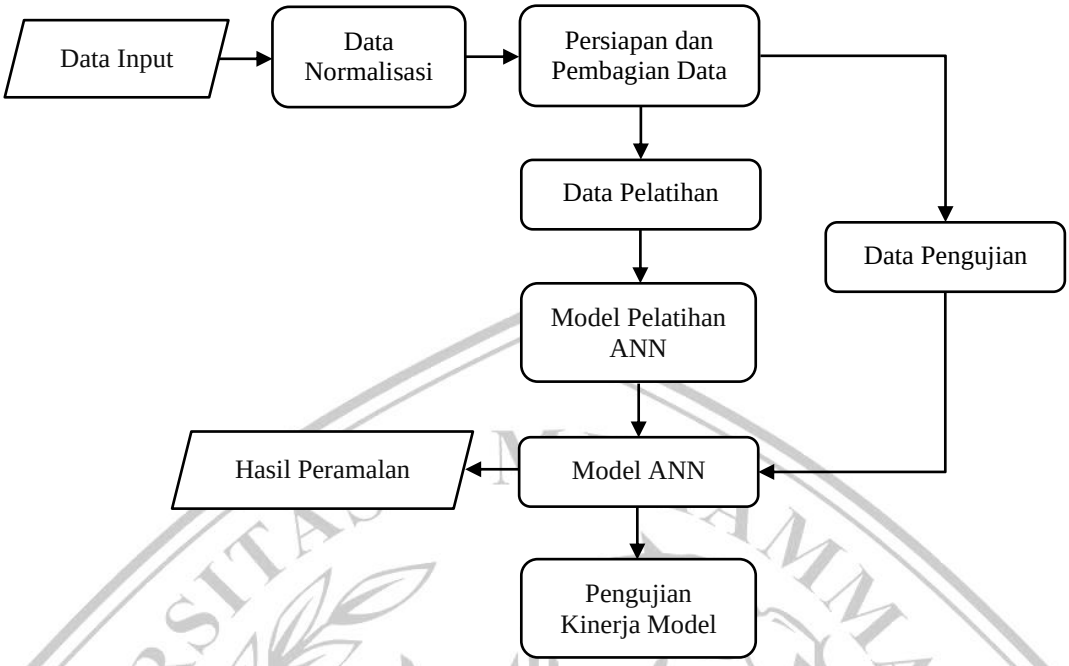
Data beban listrik yang digunakan dalam penelitian ini berasal dari PT. PLN UP3 Tanjung Pinang dan mencakup pada rentang waktu mulai dari Januari 2018 hingga Oktober 2023. Berikut ini adalah ringkasan mengenai data beban listrik selama periode Januari 2018 hingga Oktober 2023, yang secara keseluruhan akan terdapat pada Lampiran 1.

**Tabel 3.1 Data Beban Listrik PT. PLN UP3 Tanjung Pinang**
**Januari 2018 – Oktober 2023**

| Bulan – Tahun | Beban Listrik (VA) |
|---|---|
| **Jan-18** | 479.629.446 |
| **Feb-18** | 481.940.296 |
| **Mar-18** | 484.401.546 |
| **Apr-18** | 495.333.146 |
| **May-18** | 495.974.586 |
| ⋮ | ⋮ |
| **Jun-23** | 700.162.966 |
| **Jul-23** | 701.152.816 |
| **Aug-23** | 705.230.616 |
| **Sep-23** | 709.459.516 |
| **Oct-23** | 714.106.916 |

## 3.2 Perancangan Sistem

Dalam penelitian ini, peramalan beban listrik bulanan dilakukan berdasarkan data historis beban listrik yang bersumber dari PT. PLN UP3 Tanjung Pinang. Metode yang digunakan untuk melakukan peramalan beban listrik adalah *Artificial Neural Network* dengan algoritma *Backpropagation* sebagai metode pembelajarannya. Pemrograman yang digunakan dalam penelitian ini adalah Python. Gambaran umum mengenai model penelitian dapat dilihat pada **Gambar 3.1** di bawah ini.




**Gambar 3.1 Blok Diagram Peramalan Beban Listrik Metode ANN**

Dari blok diagram yang terdapat pada **Gambar 3.2,** alur penelitian dimulai dengan memperoleh data beban listrik dari PT. PLN UP3 Tanjung Pinang. Langkah pertama melibatkan pengimporan data menggunakan library yang relevan. Selanjutnya, proses melibatkan normalisasi data beban listrik untuk menyesuaikan data ke dalam skala yang konsisten. Normalisasi ini menjadi tahap awal untuk memastikan konsistensi dan keseimbangan karakteristik dataset. Pengolahan data berlanjut dengan melatih model *Artificial Neural Network* (ANN) hingga diperoleh output model ANN. Pendekatan ini memungkinkan penggunaan metode *Artificial Neural Network* (ANN) dengan algoritma *Backpropagation* untuk melakukan peramalan beban listrik dengan akurat. Adapun penjelasan terkait masing-masing tahapan sebagai berikut.

### 3.2.1 Input Data

Pada **Gambar 3.1** data beban listrik yang telah diperoleh akan dilakukan proses menginput data dengan cara impor data. Impor data dilakukan dengan menggunakan *library* yang sesuai sebagai awal dari seluruh proses analisis. Pemilihan *library* yang tepat memungkinkan pengguna untuk membaca data dari berbagai sumber dengan efisien. Adapun *library* ANN dalam lingkup

pemrograman Python yang digunakan dalam penelitian ini terdapat pada **Gambar 3.2** sebagai berikut.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from tabulate import tabulate
from datetime import datetime, timedelta
import calendar
```

**Gambar 3.3 Listing Program *Library* ANN**

### 3.2.2 Normalisasi Data

Setelah tahap impor data, langkah selanjutnya dalam pengolahan data adalah normalisasi data. Normalisasi merupakan proses penting dalam persiapan data sebelum digunakan untuk analisis lebih lanjut agar variabel beban listrik memiliki rentang nilai dan skala yang konsisten dan sama. Pada penelitian ini data dinormalisasikan pada interval (0,1) atau rentan nilai 0 sampai 1. Hal ini memungkinkan model ANN yang akan digunakan dapat memahami dan memproses data dengan lebih baik. Adapun untuk melakukan normalisasi data maka digunakan Persamaan 3.1 [1].

$$X' = \frac{0.8\,(X-a)}{(b-a)} + 0.1 \qquad \textbf{(3.1)}$$

dimana :

X’ = Data Normalisasi

X = Data Aktual

a = Data Aktual Minimum

b = Data Aktual Maksimum

Dari Persamaan 3.1 tersebut, kemudian dimasukkan kedalam listing program untuk dilakukan normalisasi data dengan rentan 0 sampai 1 seperti pada **Gambar 3.3** berikut.

```
# Normalisasi data
scaler = MinMaxScaler()
data['load_normalized']
scaler.fit_transform(data['Load'].values.reshape(-1, 1))
```

**Gambar 3. 3 Listing Program Normalisasi Data**

### 3.2.3 Persiapan dan Pembagian Data

Persiapan data adalah tahap penting dalam pengembangan model peramalan beban listrik menggunakan *Artificial Neural Network* (ANN) dengan algoritma *Backpropagation*. Pada tahap ini, data yang sudah dinormalisasi disiapkan untuk pelatihan dan pengujian model ANN. Untuk itu, digunakan variabel yang menentukan jumlah data yang digunakan sebagai input untuk meramal data berikutnya dalam rangkaian waktu (*time series*). Semakin besar nilai variabel tersebut, semakin banyak data masa lalu yang digunakan sebagai input yang memengaruhi kemampuan model dalam memahami pola dalam data time series.

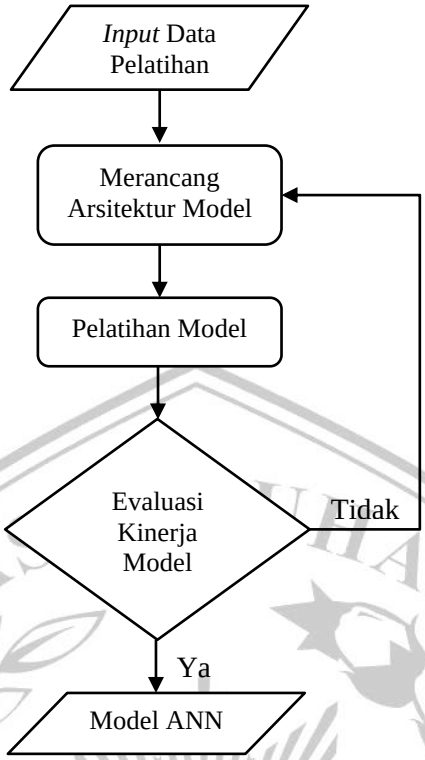
Selanjutnya, terdapat tahap pembagian data yang melibatkan pemisahan dataset menjadi dua subset utama: data pelatihan (*training data*) dan data uji (*testing data*). Data pelatihan digunakan untuk melatih model ANN, sementara data uji digunakan untuk mengukur sejauh mana model mampu melakukan peramalan pada data yang belum pernah dilihat sebelumnya. Pembagian data ini bertujuan untuk mengukur kemampuan model dalam menggeneralisasi pola dari data pelatihan ke data baru. Terdapat dua variasi rasio pembagian data yang digunakan, yaitu rasio 80% data pelatihan dan 20% data uji (80:20), serta rasio 70% data pelatihan dan 30% data uji (70:30). Berikut listing program pembagian data pada **Gambar 3.4**.

```
# Persiapan data untuk pelatihan

look_back = 12
X, y = [], []

for i in range(look_back, len(data)):
    X.append(data['load_normalized'][i-look_back:i])
    y.append(data['load_normalized'][i])

X = np.array(X)
y = np.array(y)

# Pembagian data untuk pelatihan

TRAIN_SIZE = 0.7  # Rasio 70%
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=TRAIN_SIZE, test_size=0.3, shuffle=False)

TRAIN_SIZE = 0.8  # Rasio 80%
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=TRAIN_SIZE, test_size=0.2, shuffle=False)
```

**Gambar 3.4 Listing Program Pembagian Data**

### 3.2.4 Pelatihan Model ANN

Setelah menyelesaikan tahap persiapan dan pembagian data untuk data pelatihan dan data uji, pengolahan data dilanjutkan dengan melatih model Artificial Neural Network (ANN) menggunakan metode backpropagation. Pada tahap pelatihan, terdapat serangkaian langkah yang melibatkan pembangunan dan pelatihan model ANN. Proses ini mencakup perancangan arsitektur, pelaksanaan proses pelatihan, dan evaluasi kinerja model hingga menghasilkan output dari model ANN. Flowchart yang menggambarkan proses pelatihan model ANN dapat ditemukan pada **Gambar 3.5** berikut.




**Gambar 3.5 Flowchart Proses Pelatihan Metode ANN-*Backpropagation***

### 3.2.4.1 Rancang Arsitektur Model

Rancang bangun arsitektur model adalah langkah awal dalam pengembangan ANN. Model arsitektur menentukan cara lapisan dan komponen-komponen model ditempatkan serta dihubungkan untuk memungkinkan model memproses data. Arsitektur model ANN terdiri dari beberapa komponen inti, seperti lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*). Lapisan masukan berfungsi untuk menerima data masukan, lapisan tersembunyi bertugas memproses data dan mengidentifikasi pola, sementara lapisan keluaran menghasilkan prediksi atau output model.

Dalam setiap lapisan, terdapat fungsi aktivasi untuk memungkinkan model memahami hubungan yang kompleks dalam data. Fungsi aktivasi ini berperan penting dalam proses pembelajaran dan perlu divariasikan selama pelatihan model untuk mencapai hasil yang optimal. Pada penelitian ini, digunakan fungsi aktivasi ReLU (*Rectified Linear Unit*) pada lapisan masukan dan lapisan tersembunyi serta fungsi aktivasi linier pada lapisan keluaran. Fungsi ReLU mengaktifkan neuron jika masukan lebih besar dari

nol dan mematikannya jika kurang dari nol. Sedangkan fungsi aktivasi linier merupakan output dari hasil perkalian sederhana dari input, bobot, dan penambahan bias.

Semetara itu, terdapat fungsi kerugian (*loss function*) yang menggunakan *Mean Squared Error* (MSE) untuk mengukur sejauh mana prediksi model berbeda dari nilai sebenarnya. Tujuan utama fungsi kerugian adalah untuk meminimalkan nilai fungsi kerugian agar model memberikan prediksi yang akurat. Dalam arsitektur model, terdapat pula parameter optimisasi yang berfungsi untuk mengubah bobot dan bias dalam model, dengan tujuan meminimalkan kesalahan selama pelatihan. Algoritma optimisasi yang digunakan adalah *Adam*, yang mengatur laju pembelajaran secara otomatis selama proses pelatihan yang memungkinkan model untuk cepat mengkonvergensi ke solusi yang optimal. Berikut listing program arsitektur model ANN pada **Gambar 3.6**.

```
model = Sequential()
model.add(Dense(64, input_dim=look_back,
activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='linear'))

model.compile(loss='mean_squared_error', optimizer='adam',
metrics=[ tf.keras.metrics.MeanSquaredError(),
```

**Gambar 3.6 Listing Program *Input* dan *Output* Data**

#### 3.2.4.2 Pelatihan Model

Setelah merancang arsitektur model, langkah berikutnya adalah tahap pelatihan model. Dalam tahap ini, model ANN-*backpropagation* akan mempelajari pola-pola yang terkandung dalam data pelatihan untuk dapat membuat prediksi yang akurat di masa depan. Proses pelatihan dimulai dengan inisialisasi model, yang berarti model disiapkan dengan konfigurasi yang telah ditentukan sebelumnya, termasuk jumlah lapisan dan neuron di setiap lapisan.

Selama pelatihan, model melakukan iterasi (epoch) pada data pelatihan dengan memperbaiki diri saat terdapat perbedaan antara prediksi

dan nilai sebenarnya. Proses pelatihan juga melibatkan batch data yang lebih kecil untuk menghindari masalah ukuran data. Model terus memperbaiki parameter hingga mencapai performa yang diinginkan, dan durasi pelatihan bervariasi berdasarkan kompleksitas model dan ukuran dataset. Adapun listing program untuk pelatihan model seperti pada **Gambar 3.7** berikut.

```
model.fit(X_train, y_train, epochs=100, batch_size=16,
verbose=1, validation_data=(X_test, y_test))
```

**Gambar 3.7 Listing Program Parameter Pelatihan Model ANN**

Apabila perancangan arsitektur dan pelatihan model telah ditentukan, maka proses berikutnya adalah pelatihan. Proses pelatihan adalah langkah krusial dalam pengembangan model peramalan. Pada tahap ini, sejumlah percobaan dilakukan dengan variasi nilai *look_back*, *train_size* dan arsitektur model yang berbeda untuk mencari parameter optimal yang menghasilkan *Mean Squared Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE) terkecil.

Model akan menjalankan beberapa iterasi dengan parameter internal yang diperbarui untuk meningkatkan performa peramalan. Hasil dari proses pelatihan menjadi signifikan sebab parameter yang optimal akan menghasilkan prediksi yang lebih akurat, mengurangi tingkat kesalahan, dan meningkatkan kehandalan model untuk memprediksi data di masa depan. Proses pelatihan berulang kali dilakukan hingga ditemukan parameter terbaik yang menghasilkan MSE dan MAPE minimal. Adapun parameter model jaringan ANN divariasikan dalam **Tabel 3.2** berikut.

**Tabel 3.2 Parameter Model Jaringan ANN-*Backpropagation***

| No. | *Look_back* | *Train_size* | *Build Model (Input, Hidden, Output)* |
|---|---|---|---|
| **1.** | 12 | 0.7 | 64,32,1 |
| **2.** | 24 | 0.7 | 64,32,1 |
| **3.** | 36 | 0.7 | 64,32,1 |

| | | | |
|---|---|---|---|
| **4.** | 12 | 0.7 | 64,64,1 |
| **5.** | 24 | 0.7 | 64,64,1 |
| **6.** | 36 | 0.7 | 64,64,1 |
| **7.** | 12 | 0.8 | 64,32,1 |
| **8.** | 24 | 0.8 | 64,32,1 |
| **9.** | 36 | 0.8 | 64,32,1 |
| **10.** | 12 | 0.8 | 64,64,1 |
| **11.** | 24 | 0.8 | 64,64,1 |
| **12.** | 36 | 0.8 | 64,64,1 |

#### 3.2.4.3 Evaluasi Kinerja Model

Setelah selesai tahap prose pelatihan, langkah selanjutnya adalah evaluasi kinerja model. Evaluasi ini bertujuan untuk mengukur seberapa baik model ANN-*backpropagation* yang telah dilatih dalam melakukan prediksi menggunakan data yang belum pernah dilihat sebelumnya. Evaluasi kinerja model adalah tahap penting untuk memahami sejauh mana model dapat memberikan hasil yang akurat dan sesuai dengan tujuan pengembangan model peramalan.

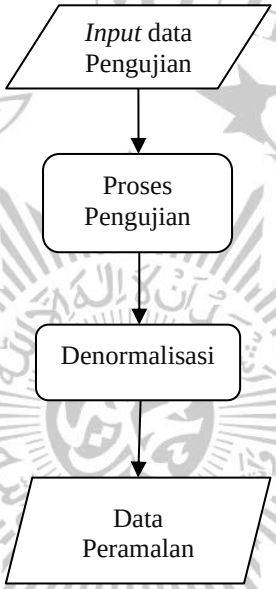
Metrik evaluasi yang digunakan adalah *Mean Squared Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE). MSE mengukur seberapa baik model ANN dapat memprediksi nilai dengan akurat. Semakin rendah nilai MSE, semakin baik model dalam meminimalkan kesalahan prediksi. Sedangkan MAPE memberikan gambaran persentase rata-rata kesalahan prediksi terhadap nilai sebenarnya. MAPE membantu dalam memahami sejauh mana model sesuai dengan ekspektasi, terutama dalam konteks peramalan. Semakin rendah nilai MAPE, semakin baik model dalam melakukan prediksi. Adapun listing program evaluasi kinerja model dapat dilihat pada **Gambar 3.8** berikut.

```
loss, mse, mape = model.evaluate(X_test, y_test,
batch_size=BATCH_SIZE)
```

**Gambar 3.8 Listing Program Evaluasi Kinerja Model**

### 3.2.5 Pengujian Model

Setelah tahap pelatihan, langkah selanjutnya adalah tahap pengujian model. Tahap pengujian memiliki peran dalam mengukur sejauh mana model ANN yang telah dilatih mampu memberikan prediksi yang akurat pada data yang belum pernah dilihat sebelumnya. Evaluasi kinerja pada tahap ini diperlukan untuk memastikan bahwa model ANN siap digunakan dalam situasi dunia nyata. Tahap pengujian melibatkan serangkaian proses mulai dari proses pengujian, denormalisasi hingga diperoleh data *output* peramalan seperti pada **Gambar 3.9** berikut.




**Gambar 3.9 Flowchart Proses Pengujian Metode ANN-*Backpropagation***

#### 3.2.5.1 Proses Pengujian

Tahap pengujian merupakan langkah akhir dalam pengembangan model *Artificial Neural Network* (ANN) dengan metode *Backpropagation*, bertujuan untuk memastikan kehandalan model dalam situasi dunia nyata. Proses pengujian melibatkan beberapa tahap, diawali dengan pemisahan antara data pengujian dan data pelatihan. Data pengujian bersifat independen dan belum pernah digunakan sebelumnya untuk melatih model. Model ANN yang telah dilatih kemudian digunakan untuk membuat prediksi pada data pengujian. Evaluasi hasil prediksi dilakukan dengan menggunakan metrik evaluasi seperti *Mean Squared Error* (MSE) dan *Mean Absolute Percentage Error* (MAPE). Langkah ini bertujuan untuk

membantu menilai kesiapan model untuk digunakan dalam dunia nyata atau mengidentifikasi kebutuhan perbaikan lebih lanjut jika diperlukan. Adapun listing program untuk proses pengujian dapat ditemukan pada **Gambar 3.10** berikut.

```
y_pred_normalized = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred_normalized)

y_test_actual = scaler.inverse_transform(y_test.reshape(-
1, 1))
mse = mean_squared_error(y_test_actual, y_pred)
mape = np.mean(np.abs((y_test_actual - y_pred) /
y_test_actual)) * 100
```

**Gambar 3.10 Listing Program Proses Pengujian**

Evaluasi hasil prediksi dalam model ANN-*Backpropagation* adalah langkah penting dalam menentukan sejauh mana model mampu memodelkan dan memprediksi data dengan akurasi. Dalam upaya ini, metrik evaluasi seperti MSE dan MAPE digunakan untuk meminimalkan kesalahan prediksi sebanyak mungkin. Namun, perbedaan antara data yang diprediksi oleh model dan data aktual juga memiliki signifikansi penting dalam proses evaluasi.

Persentase selisih adalah metrik kunci dalam evaluasi model prediksi. Metrik ini bertujuan untuk memahami sejauh mana perbedaan antara data prediksi dan data aktual. Untuk menghitung persentase selisih, digunakan Persamaan 3.2 sebagai berikut:

$$\text{Persentase Selisih} = \frac{(a-b)}{b} \times 100 \quad \textbf{(3.2)}$$

dimana :

a = Data Aktual

b = Data Prediksi

Pada Persamaan 3.2 memberikan persentase yang menggambarkan perbedaan antara nilai prediksi model dan nilai aktual. Semakin kecil persentase selisih, semakin baik kinerja model. Selain itu, pemahaman

tentang persentase selisih membantu mengidentifikasi pola kesalahan model, seperti kecenderungan *over-predict* (prediksi terlalu tinggi) atau *under-predict* (prediksi terlalu rendah), yang dapat digunakan untuk perbaikan model. Hal ini penting dalam memastikan bahwa model dapat diandalkan dalam aplikasi dunia nyata.

#### 3.2.5.2 Denormalisasi

Setelah proses pengujian telah dilakukan dan memperoleh hasil peramalan beban yang sesuai, maka tahap selanjutnya yang perlu dilakukan yaitu denormalisasi data. Denormalisasi data adalah proses yang melibatkan pengembalian data ke dalam satuan asli atau skala semula setelah data dinormalisasi. Untuk melakukan denormalisasi data maka digunakan Persamaan 3.3 [2]:

$$X = \frac{(X'-0.1)(b-a)}{0.8} + a \quad \textbf{(3.3)}$$

dimana :

X’ = Data Normalisasi
X = Data Aktual
a = Data Aktual Minimum
b = Data Aktual Maksimum

Pada Persamaan 3.3 diatas kemudian dimasukkan kedalam listing program untuk dilakukan denormalisasi data dengan mengubah ke bilangan real seperti pada **Gambar 3.11** berikut.

```
data_denormalized =
scaler.inverse_transform(data['load_normalized'].valu
es.reshape(-1, 1))
```

**Gambar 3.11 Listing Program Denormalisasi**