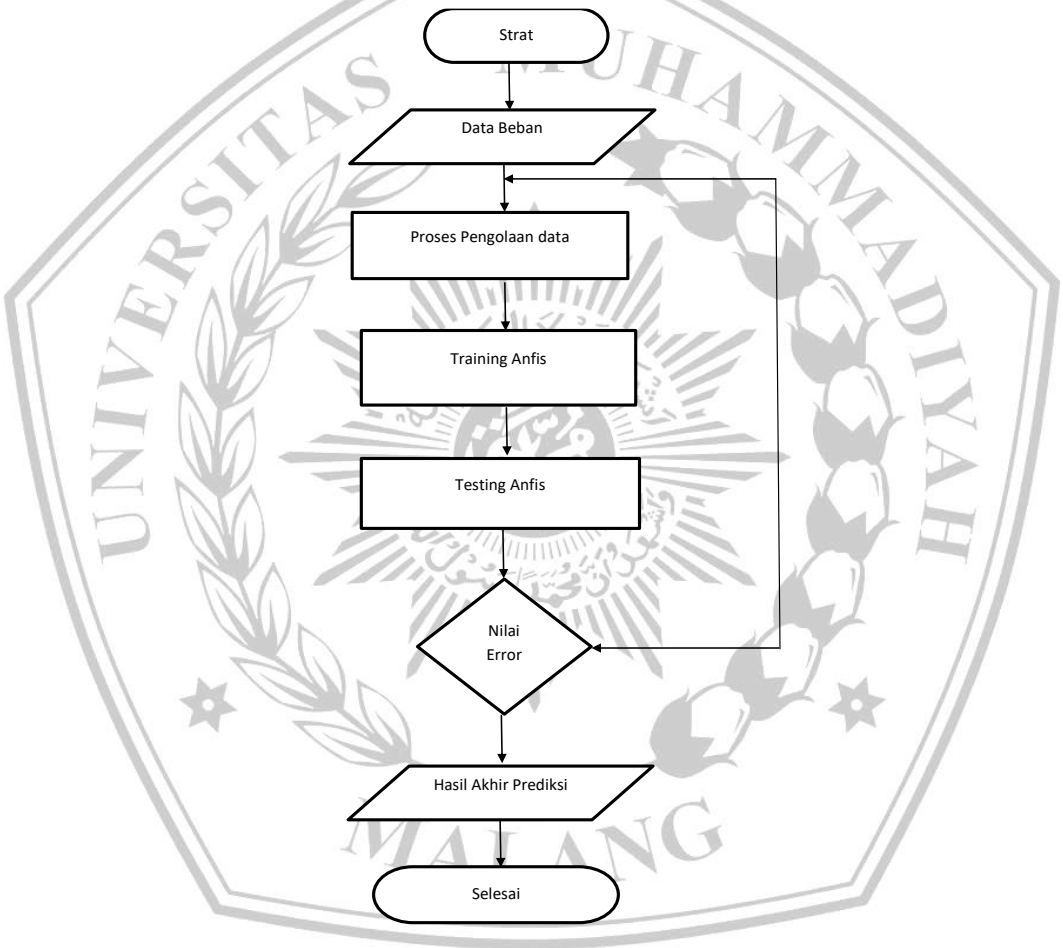
BAB III
METODOLOGI PENELITIAN

3.1 METODE

Bab ini menjelaskan tentang metodologi penelitian peramalan beban listrik jangka pendek, meliputi sumber data, variabel penelitian, metode analisis data untuk mengatasi permasalahan, Penelitian ini menggunakan bahasa pemrograman Paython, metode yang diusulkan secara umum ditunjukkan pada followchart pada Gambar 3.1



3.1 Gambar Flowchart pengolahan data

Gambar 3.1 menggambarkan peramalan beban listrik jangka pendek dengan menggunakan model ANFIS. Identifikasi variabel masukan dan variabel keluaran.

Tabel 3.1 Data beban listrik PT. PLN (PERSERO) Wilayah Surabaya Utara dari bulan Januari 2019 – Desember 2023

Tabel 3.1.1 Jumlah pelanggan dan Daya terpasang 2019

Tahun 2019	Jumlah Pelanggan (PLG)	Daya Terpasang (VA)
Januari	34.229	244.583.100
Februari	34.199	244.482.550
Maret	34.164	244.790.650
April	34.124	244.736.050
Mei	34.099	245.056.750
Juni	34.079	250.500.600
Juli	34.038	250.738.650
Agustus	34.000	250.826.250
September	31.372	255.282.350
Oktober	33.909	250.893.700
Nopember	33.867	250.845.600
Desember	33.827	251.127.750

Tabel 3.1.2 Jumlah pelanggan dan Daya terpasang 2020

Tahun 2020	Jumlah Pelanggan (PLG)	Daya Terpasang (VA)
Januari	33.784	251.455.050
Februari	33.739	251.852.200
Maret	33.707	252.035.000
April	33.666	252.305.900
Mei	33.619	251.948.000
Juni	33.653	251.862.100
Juli	33.655	251.712.200
Agustus	33.616	251.875.000
September	33.567	252.093.900
Oktober	33.518	251.994.750
Nopember	33.453	252.122.350
Desember	33.409	252.156.300

Tabel 3.1.3 Jumlah pelanggan dan Daya terpasang 2021

Tahun 2021	Jumlah Pelanggan (PLG)	Daya Terpasang (VA)
Januari	33.325	252.318.300
Februari	33.259	252.427.450
Maret	33.209	252.396.950
April	33.154	252.503.950
Mei	33.084	252.544.300
Juni	33.000	252.604.850
Juli	32.923	252.512.150
Agustus	32.872	252.725.450
September	32.787	252.245.350
Oktober	32.725	252.359.400
Nopember	32.640	253.403.650
Desember	32.564	253.337.050

Tabel 3.1.4 Jumlah pelanggan dan Daya terpasang 2022

Tahun 2022	Jumlah Pelanggan (PLG)	Daya Terpasang (VA)
Januari	32.500	253.312.950
Februari	32.445	253.200.400
Maret	32.400	253.380.800
April	32.353	254.278.100
Mei	32.299	254.118.550
Juni	32.198	253.906.000
Juli	32.124	253.489.700
Agustus	32.050	253.530.250
September	31.995	254.145.150
Oktober	31.916	253.772.450
Nopember	31.857	253.959.550
Desember	31.785	254.371.850

Tabel 3.1.5 Jumlah pelanggan dan Daya terpasang 2023

Tahun 2023	Jumlah Pelanggan (PLG)	Daya Terpasang (VA)
Januari	31.734	254.613.800
Februari	31.684	254.887.700
Maret	31.617	254.794.100
April	31.579	254.949.650
Mei	31.540	254.748.950
Juni	31.516	254.961.000
Juli	31.462	254.725.000
Agustus	31.414	255.120.800
September	33.965	251.054.150
Oktober	31.317	255.421.500
Nopember	31.263	255.750.450
Desember	31.734	254.613.800

Data mentah tersebut digabungkan menjadi satu untuk dimasukkan ke dalam program anfis, supaya diolah dan digunakan sebagai input. Ada 2 tahapan yang diklasifikasi dengan cara dikelompokkan menjadi masing-masing pertahun atau 60 bulan dari bulan januari sampai dengan bulan Desember selama 5 tahun ke belakang.

3.2 Skema Desain Anfis

Class parameter anfis

Pada class ini berfungsi untuk memasukan beberapa input parameter digunakan untuk mengontrol pelatihan dan inferensi yang akan diterima oleh ANFIS

```
class fis_parameters():

  def __init__(self, n_input: int = 3, n_memb: int = 3, batch_size: int = 16, n_epochs: int = 25, memb_func: str = 'gaussian', optimizer: str = 'sgd', loss: str = 'mse'):

    self.n_input = n_input  # no. of Regressors

    self.n_memb = n_memb  # no. of fuzzy memberships

    self.batch_size = batch_size

    self.n_epochs = n_epochs

    self.memb_func = memb_func  # 'gaussian' / 'gbellmf'

    self.optimizer = optimizer   # sgd / adam /

    self.loss = loss  # mse / mae
```

Gambar 3.2 Listing Program Class parameter anfis

Class ANFIS

Pada class ini berfungsi untuk membangun model ANFIS sebagai pelatihan dengan menggunakan objek pada inferensi fuzzy

```
class ANFIS:
    def __init__(self, n_input: int, n_memb: int, batch_size: int = 16, memb_func: str =
'gaussian', name: str = 'MyAnfis'):
        self.n = n_input
        self.m = n_memb
        self.batch_size = batch_size
        self.memb_func = memb_func
        input_ = keras.layers.Input(
            shape=(n_input), name='inputLayer', batch_size=self.batch_size)
        L1 = FuzzyLayer(n_input, n_memb, memb_func, name='fuzzyLayer')(input_)
        # norms_layer = tf.keras.layers.LayerNormalization(axis=1, center=True,
name='normalize')(L1)
        L2 = RuleLayer(n_input, n_memb, name='ruleLayer')(L1)
        L3 = NormLayer(name='normLayer')(L2)
        L4 = DefuzzLayer(n_input, n_memb, name='defuzzLayer')(L3, input_)
        L5 = SummationLayer(name='sumLayer')(L4)
        # L6 = tf.keras.layers.Dense(10, activation='relu', name='hidden_layer')(L5)
        # L7 = tf.keras.layers.Dense(1, name='outputs')(L6)
        self.model = keras.Model(inputs=[input_], outputs=[L5], name=name)
        self.update_weights()
    def __call__(self, X):
        return self.model.predict(X, batch_size=self.batch_size)
    def update_weights(self):
        # tạo các tham số tiền đề (mu&sigma for gaussian // a/b/c for bell-shaped)
        if self.memb_func == 'gaussian':
            self.mus, self.sigmas = self.model.get_layer(
                'fuzzyLayer').get_weights()
```

Gambar 3.3 Listing Program Class ANFIS

Fuzzy Layer

Pada code block ini digunakan untuk menggunakan identasi atau membangun fuzzy layer dalam ANFIS. Pada gambar code block ini memakai fungsi keanggotaan Gaussian untuk memetakan input data ke nilai fuzzy.

```
class FuzzyLayer(keras.layers.Layer):
    def __init__(self, n_input, n_memb, memb_func='gaussian', **kwargs):
        super(FuzzyLayer, self).__init__(**kwargs)
        self.n = n_input
        self.m = n_memb
        self.memb_func = memb_func

    def build(self, batch_input_shape):
        self.batch_size = batch_input_shape[0]

        if self.memb_func == 'gbellmf':
            self.a = self.add_weight(name='a',
                                     shape=(self.m, self.n),
                                     initializer=keras.initializers.RandomUniform(
                                        minval=.7, maxval=1.3, seed=1),
                                     #initializer = 'ones',
                                     trainable=True)
            self.b = self.add_weight(name='b',
                                     shape=(self.m, self.n),
                                     initializer=keras.initializers.RandomUniform(
                                        minval=.7, maxval=1.3, seed=1),
                                     #initializer = 'ones',
                                     trainable=True)
```

Gambar 3.4 Listing program Fuzzy Layer

Rule Layer

Code block di bawah ini menjelaskan bahwa class Rule layer mendefinisikan aturan dalam system inferensi ANFIS untuk menggabungkan nilai fuzzy dari class fuzzy layer untuk menghasilkan konsekuensi fuzzy.

```
class RuleLayer(keras.layers.Layer):
  def __init__(self, n_input, n_memb, **kwargs):
    super(RuleLayer, self).__init__(**kwargs)
    self.n = n_input
    self.m = n_memb
    self.batch_size = None

  def build(self, batch_input_shape):
    self.batch_size = batch_input_shape[0]
    # self.batch_size = tf.shape(batch_input_shape)[0]
    super(RuleLayer, self).build(batch_input_shape)

  def call(self, input_):
    if self.n == 2:
      L2_output = tf.reshape(input_[:, :, 0], [self.batch_size, -1, 1]) * \
        tf.reshape(input_[:, :, 1], [self.batch_size, 1, -1])
```

Gambar 3.5 Listing Program Rule Layer

Normalisasi Layer

Pada code block di bawah ini menunjukkan bahwa berfungsi untuk menormalisasi nilai – nilai ke dalam sebuah di sepanjang sumbu 1. Hasilnya kemudian diubah menjadi dimensi (-1, 1) yang akan dibagi oleh parameter *w* dengan hasil normalisasi pada layer.

```
class NormLayer(keras.layers.Layer):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def call(self, w):
        w_sum = tf.reshape(tf.reduce_sum(w, axis=1), (-1, 1))
        w_norm = w / w_sum
        return w_norm
```

Gambar 3. 6 Listing Program Normalisasi Layer

Layer Defuzzification

```
class DefuzzLayer(keras.layers.Layer):
    def __init__(self, n_input, n_memb, **kwargs):
        super().__init__(**kwargs)
        self.n = n_input
        self.m = n_memb

        self.CP_bias = self.add_weight(name='Consequence_bias',
                                       shape=(1, self.m ** self.n),
                                       initializer=keras.initializers.RandomUniform(
                                           minval=-2, maxval=2),
                                       # initializer = 'ones',
                                       trainable=True)
```

Gambar 3.7 Listing Program Layer Defuzzification

Pada code block di atas menunjukkan proses pengubahan nilai output dari system fuzzy menjadi nilai yang dapat dipahami. Pada proses penghitungan outputnya mengalikan w_{norm} dengan hasil perkalian $input\_$ dan cp_{weight} dan menambahkan cp_bias kemudian dikalikan dengan output layer.

Layer Summation

Pada code block ini layer summation digunakan untuk menjumlahkan nilai – nilai dalam sebuah tensor kemudian dikembalikan sebagai nilai output terakhir pada fuzzy dengan menjumlahkan nilai – nilai pada tensor di sepanjang sumbu 1 yang akan menghasilkan bentuk tensor output ke dalam dimensi $(-1, 1)$.

```
class SummationLayer(keras.layers.Layer):
  def __init__(self, **kwargs):
    super().__init__(**kwargs)

  def build(self, batch_input_shape):
    self.batch_size = batch_input_shape[0]
    #self.batch_size = tf.shape(batch_input_shape)[0]
    # gọi
    super(SummationLayer, self).build(batch_input_shape)

  def call(self, input_):
    L5_L2_output = tf.reduce_sum(input_, axis=1)
    L5_L2_output = tf.reshape(L5_L2_output, (-1, 1))
```

Gambar 3.8 Listing program Layer Summation

3.3 Import Pustaka

Memasukkan kode dari pustaka dalam program.

```
import pandas as pd

from anfis import fis_parameters, ANFIS

from sklearn.preprocessing import LabelEncoder, MaxAbsScaler,
StandardScaler

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np
```

Gambar 3.9 Listing program Import Pustaka

3.4 Load data set dan preprocessing

Memuat dataset dan melakukan pra-pemrosesan (preprocessing) dalam pembangunan model dengan data bulan. Serta memuat dataset, memisahkan fitur dan label, membagi dataset menjadi data latih dan data uji, serta melakukan normalisasi r untuk analisis data dan pengembangan model machine learning yang lebih lanjut.

```
df = pd.read_excel("dataset01.xlsx")

df['Bulan'] = sorted(LabelEncoder().fit_transform(df['Bulan']))

normscaler = MaxAbsScaler().fit_transform(df)

df.describe()
```

Gambar 3.10 Listing program Load data set dan preprocessing

3.5 Pemilihan fitur dan label

Pemilihan fitur dan label. langkah penting dalam analisis data dan pembangunan model machine learning.

```
selc =10
X = normscaler[:,0:selc]
Y = normscaler[:,-1]
Y.shape
```

Gambar 3.11 Listing program Pemilihan fitur dan label

3.6 Pembagian data training dan testin

Pembagian data menjadi data latih (training data) dan data uji (testing data) merupakan langkah kritis dalam pembangunan model machine learning. Pembagian ini bertujuan untuk menguji kinerja model yang dikembangkan pada data yang belum pernah dilihat sebelumnya, sehingga dapat memberikan perkiraan yang lebih baik tentang kinerja model di dunia nyata

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
                X, Y, test_size=0.2, random_state=42)
print(f'X_train shape: {X_train.shape}')
print(f'X_test shape: {X_test.shape}')

print(f'y_train shape: {y_train.shape}')
print(f'y_test shape: {y_test.shape}')
# clf.fit(X_train, y_train)
# print(clf.score(X_test, y_test))
```

Gambar 3.12 Listing program Pembagian data training dan testing

3.7 Training Anfis

Training Adaptive Neuro-Fuzzy Inference System (ANFIS) melibatkan penyesuaian parameter model agar sesuai dengan data latih.

```
param = fis_parameters(
        n_input = selc,              # no. of Regressors
        n_memb = 5,                  # no. of fuzzy memberships
        batch_size = 1,              # 16 / 32 / 64 / ...
        memb_func = 'sigmoid',       # 'gaussian' / 'gbellmf' / 'sigmoid'
        optimizer = tf.keras.optimizers.Adam(learning_rate=0.001,
                                    beta_1=0.99,
                                     beta_2=0.999,
                                     epsilon=1e-07,),   # sgd / adam / ...
        loss = tf.keras.losses.MeanAbsolutePercentageError(),
        # loss = 'huber_loss',              # mse / mae / huber_loss / mean_absolute_percentage_error /
...
        n_epochs = 100             # 10 / 25 / 50 / 100 / ...
        )
fis = ANFIS(n_input = param.n_input,
                    n_memb = param.n_memb,
                    batch_size = param.batch_size,
                    memb_func = param.memb_func,
                    name = 'myanfis'
                    )
  # compile modelm
fis.model.compile(optimizer=param.optimizer,
                  loss=param.loss,
                  # metrics=['accuracy']
                  metrics=[tf.keras.metrics.MeanAbsolutePercentageError(),
tf.keras.metrics.RootMeanSquaredError()]  # ['mae', 'mse']
                  )
history = fis.fit(X_train, y_train,
                  epochs=param.n_epochs,
                  batch_size=param.batch_size,
                  validation_data = (X_test, y_test),
                   #callbacks = [tensorboard_callback]  # for tensorboard
                  )
histories = []
histories.append(history)
```

Gambar 3.13 Listing Program Training Anfis

3.8 Visualisasi terhadap nilai error setelah training

Pada visualisasi nilai error setelah training dalam machine learning, sehingga dapat menggunakan grafik untuk melacak bagaimana error atau loss berubah seiring dengan iterasi atau epoch selama proses training a.

```
#Plot the loss history
# plt.figure(figsize=(12, 5))
plt.plot(history.history['loss'], color='r')
plt.plot(history.history['val_loss'], color='b')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()
```

Gambar 3.15 Listing program Visualisasi terhadap nilai error setelah training

3.9 Visualisasi nilai validasi terhadap model

Visualisasi nilai validasi terhadap model sangat penting dalam proses pengembangan model machine learning. Visualisasi sejuah mana memahami seberapa baik model yang melakukan generalisasi pada data yang tidak terlihat selama proses training.

```
# plt.figure(figsize=(12, 5))
plt.plot(history.history['root_mean_squared_error'], color='r')
plt.plot(history.history['val_root_mean_squared_error'], color='b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['RMSE', 'val_RMSE'])
plt.show()
```

Gambar 3.16 Listing program Visual nilai validasi terhadap modal

3.10 Visualisasi prediksi terhadap data testing

Visualisasi prediksi pada data testing langkah penting dalam evaluasi kinerja model machine learning pada data yang belum pernah dilihat sebelumnya. membantu memahami seberapa baik model yang di pake mampu melakukan generalisasi pada data baru.

```
column_values= ['Real', 'Prediction']
df = pd.DataFrame(data = co,columns = column_values)
df.reset_index(inplace = True)


# ## Evaluate Model
# fis.model.evaluate(X_test, y_test)
plot_prediction=True
if plot_prediction:
```

Gambar 3.17 Listing program Visualisasi prediksi terhadap data testing

3.11 Visualisasi terhadap distribusi data testing dan nilai RMSE dan MAPE

melakukan visualisasi terhadap distribusi data testing dan nilai Root Mean Squared Error (RMSE) serta Mean Absolute Percentage Error (MAPE), dapat menggunakan beberapa teknik visualisasi untuk memberikan

pemahaman yang komprehensif tentang kinerja model terhadap data testing. Sehingga dapat dilihat sejuah mana nilai RMSE dan MAPE pada model ini

```
import seaborn as sns


from sklearn.metrics import root_mean_squared_error,mean_absolute_error,
mean_absolute_percentage_error

# from tensorflow.keras.metrics import mean_absolute_percentage_error

print("MAPE Score :",mean_absolute_percentage_error(y_test, y_pred))

print("RMSE Score :",root_mean_squared_error(y_test, y_pred))

print("MAE Score :",mean_absolute_error(y_test, y_pred))

sns.set_theme(color_codes=True)

ax = sns.regplot(x="Real", y="Prediction", marker='o', data=df)
```

Gambar 3.18 Listing program Visualisasi nilai RMSE dan Mape

3.12 Grafik pendistribusian data pada data nyata dan data prediksi

```
column_values= ['Real', 'Prediction']

df = pd.DataFrame(data = co,columns = column_values)

df.reset_index(inplace = True)


# ## Evaluate Model

# fis.model.evaluate(X_test, y_test)

plot_prediction=True

if plot_prediction:

y_pred = fis(X_test)

f, axs = plt.subplots(2, 1, figsize=(8, 10))

# f.suptitle(f'{gen.get_data_name(data_id)} time series',
  size=16)

axs[0].plot(y_pred)

axs[0].plot(y_test, alpha=.7)

axs[0].legend(['Real', 'Predicted'])

axs[0].grid(True)

axs[0].set_title('Real vs. Predicted values')

axs[1].plot(y_test, y_test - y_pred)

axs[1].legend(['pred_error'])

axs[1].grid(True)

axs[1].set_title('Prediction Error')

plt.show()
```

Gambar 3.19 listing Program grafik pendistribusian data pada data nyata dan data prediksi

```
import seaborn as sns


from sklearn.metrics import
root_mean_squared_error,mean_absolute_error,
mean_absolute_percentage_error

# from tensorflow.keras.metrics import mean_absolute_percentage_error

print("MAPE Score :",mean_absolute_percentage_error(y_test, y_pred))

print("RMSE Score :",root_mean_squared_error(y_test, y_pred))

print("MAE Score :",mean_absolute_error(y_test, y_pred))

sns.set_theme(color_codes=True)

ax = sns.regplot(x="Real", y="Prediction", marker='o', data=df)
```

Gambar 3.20 listing Program grafik pendistribusian data pada data nyata dan data prediksi

3.13 Hasil Presiksi

```
maxP, minP = df_1['pelanggan-23'].max(), df_1['pelanggan-23'].min()

maxD, minD = df_1['daya-23'].max(), df_1['daya-23'].min()

pelanggan = (y1_pred * (maxP - minP) + minP)

daya = (y1_pred * (maxD - minD) + minD)

concate1 = np.concatenate((pelanggan, daya), axis=1)

# concate2 = np.concatenate(df_1['pelanggan-23'], df_1['daya-23'])


pd.DataFrame(concate1,  columns=['pelanggan 2024', 'daya 2024'])
```

Gambar 3.21 Listing Program Hasil prediksi