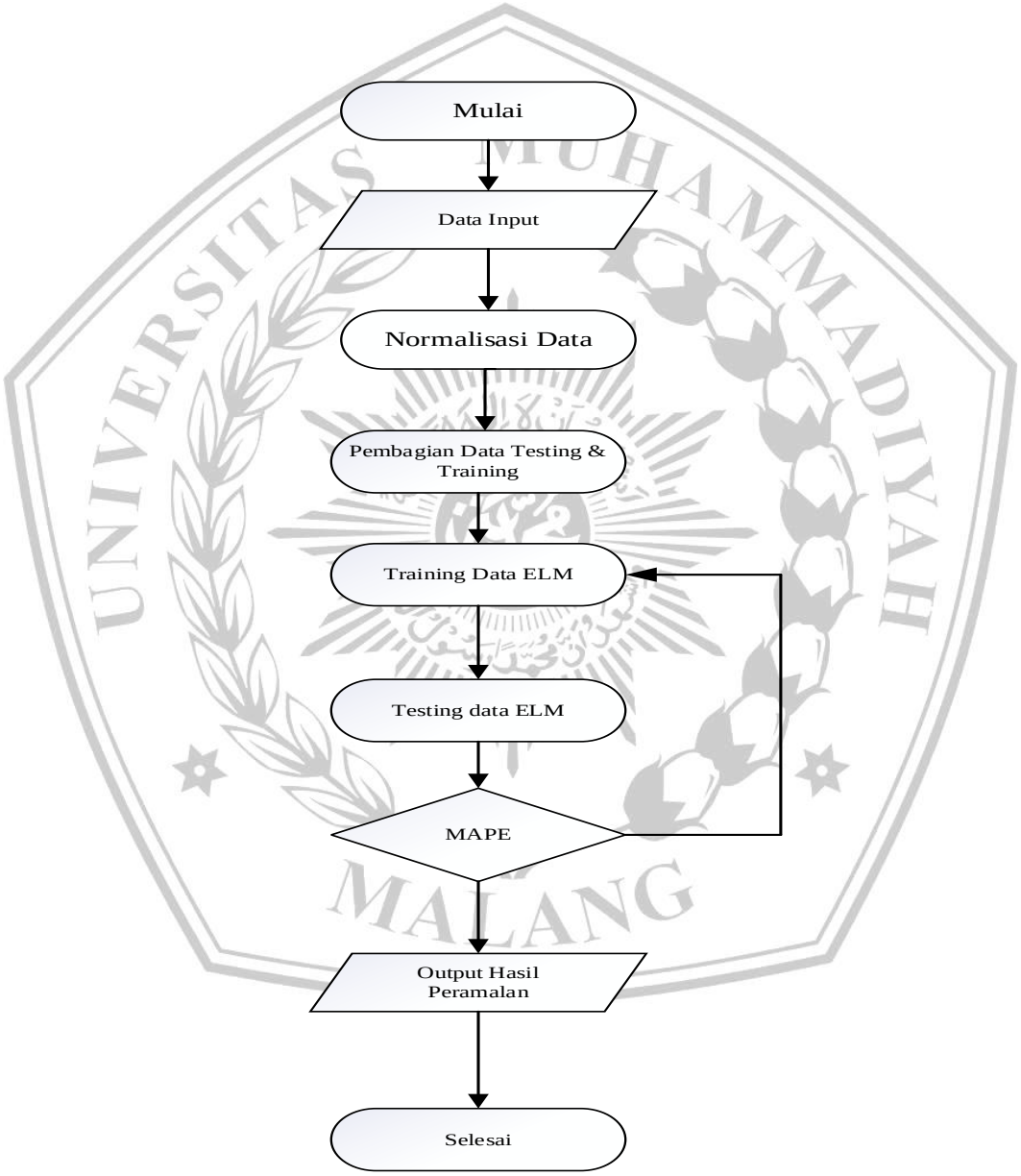
# BAB III

# PERANCANGAN SISTEM

## 3.1 Metodologi Penelitian

Pada penelitian yang akan dilakukan adalah peramalan beban listrik jangka panjang dengan objek penelitian PLN Kota Bima, metode penelitian yang digunakan untuk peramalan beban listrik jangka panjang di kota Bima adalah *Extreme Learning Machine*. Penelitian ini menggunakan bahasa pemrograman Paython. Berikut model penelitian yang secara garis besar digambarkan pada flowchart 3.1 berikut.




Gambar 3.1 Flowchart Metode ELM

### 3.2 Studi Literatur

Pada studi literatur ini dilakukan beberapa pengambilan informasi yang menunjang untuk menyelesaikan penelitian ini. Informasi yang dikumpulkan berupa literasi mulai dari jurnal, buku ataupun skripsi yang berguna sebagai penunjang pemahaman untuk menyelesaikan peramalan beban listrik jangka panjang menggunakan *Extreme Learning Machine.*

### 3.3 Pengumpulan Data

Data yang digunakan pada penelitian ini diambil dari PT. PLN (Persero) ULP Bima. Untuk data yang digunakan ialah data jumlah pelanggan, dan KWH yang terjual.

Tabel 3.1 Data Jumlah Pelanggan PT. PLN (Persero) ULP Bima

| **Jumlah Pelanggan** | | | | | |
|---|---|---|---|---|---|
| **Bulan/ Tahun** | **2014** | **2016** | **2017** | **2019** | **2020** |
| Januari | 28711 | 31894 | 36806 | 169843 | 194426 |
| Febuari | 28791 | 32181 | 37012 | 149895 | 173671 |
| Maret | 29195 | 32536 | 37154 | 149996 | 164868 |
| April | 29437 | 32800 | 37351 | 152738 | 193013 |
| Mei | 29714 | 33084 | 37596 | 156659 | 184484 |
| Juni | 30415 | 33338 | 37756 | 150287 | 181341 |
| Juli | 30570 | 33423 | 37907 | 150029 | 182040 |
| Agustus | 30590 | 33668 | 38069 | 143180 | 173967 |
| September | 30729 | 34003 | 38585 | 151580 | 186659 |
| Oktober | 31095 | 34131 | 38798 | 155332 | 183058 |
| November | 31394 | 34313 | 38989 | 167313 | 195677 |
| Desember | 31720 | 34550 | 39231 | 166285 | 184630 |

Tabel 3.2 Data KWH Penjualan PT. PLN (Persero) ULP Bima

| KWH Penjualan | | | | | |
|---|---|---|---|---|---|
| **Bulan/ Tahun** | **2014** | **2016** | **2017** | **2019** | **2020** |
| Januari | 4663188 | 5375395 | 4757482 | 21953684 | 25131215 |
| Febuari | 4511278 | 4697120 | 5051813 | 20459407 | 23704699 |
| Maret | 4662130 | 5205176 | 5718052 | 23084515 | 25373275 |
| April | 5226814 | 5217227 | 5510874 | 22535466 | 28477786 |
| Mei | 4801726 | 5289587 | 5755099 | 23980927 | 28240362 |
| Juni | 4730825 | 5210606 | 5638158 | 22442609 | 27079898 |
| Juli | 4874716 | 5287876 | 5585429 | 22106037 | 26822696 |
| Agustus | 4961496 | 5144166 | 6060684 | 22794659 | 27696048 |
| September | 4951354 | 5111803 | 5823389 | 22876957 | 28171230 |
| Oktober | 5335679 | 5663857 | 6307585 | 25253180 | 29760742 |
| November | 5350580 | 5944055 | 5962784 | 25588030 | 29925846 |
| Desember | 5024202 | 5801215 | 6167922 | 26143365 | 29027621 |

### 3.4 Normalisasi Data

Normalisasi data merupakan sebuah teknik untuk mengubah data sesuai dengan kebutuhan, Normalisasi dilakukan dengan metode *Max Normalization* (MN) yang memperbolehkan nilai berada di antara -1 dan 1. Cara metode MN. Hal ini dibutuhkan untuk *preprocessing* pada data *mining.* data akan di ubah ke interval yang lebih kecil dengan persamaan 3.1.

$$X_i' = \frac{X_i}{\max(X)} \qquad (3.1)$$

dimana :

$X_i$ = Jumlah Pelanggan pada tahun dan bulan tertentu

$X$ = Jumlah *dataset*

$X_i'$ = Hasil normalisasi

```
# normalisasi
X = jumlah_pelanggan / np.max(jumlah_pelanggan)
```

Gambar 3.2 Listing Program Normalisasi ELM

Pada penelitian ini data jumlah pelanggan dan kwh penjualan akan dinormalisasikan pada interval (0,1) atau rentan nilai 0 sampai 1.

### 3.5 Pembagian Data Training & Testing

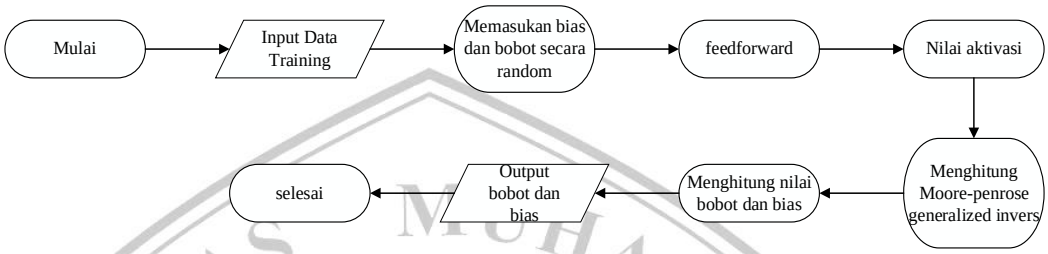
Pada proses ini data akan dibagi menjadi dua variabel yaitu jumlah pelanggan dan KWH yang terjual. Pada proses pengambilan data akan diambil secara random dimana untuk pengambilan data training dan testing yaitu 70% dan 30 % (sesuai pengujian di bab 4).

```
#pemisahan training dan testing
X_train, X_test, Y_train, Y_test = train_test_split(X, kwh_penjualan, test_size=0.3,random_state=None,shuffle=False)
_, X_test1, _, _ = train_test_split(jumlah_pelanggan, kwh_penjualan, test_size=0.3,random_state=None,shuffle=False)
print('Jumlah Pelanggan')
df = pd.DataFrame(jumlah_pelanggan, columns=['2014','2016','2017','2019','2020'])
print(tabulate(df, headers='keys', tablefmt='pretty', showindex=False))
print('kWh Penjualan')
df = pd.DataFrame(kwh_penjualan, columns=['2014','2016','2017','2019','2020'])
print(tabulate(df, headers='keys', tablefmt='pretty', showindex=False))
print('Data Normalisasi Jumlah Pelanggan')
df = pd.DataFrame(X, columns=['2014','2016','2017','2019','2020'])
print(tabulate(df, headers='keys', tablefmt='pretty', showindex=False))
print('\n')
print('Jumlah Pelanggan untuk Prediksi')
df = pd.DataFrame(X_test1, columns=['2014','2016','2017','2019','2020'])
print(tabulate(df, headers='keys', tablefmt='pretty', showindex=False))
```

Gambar 3.3 Listing Program Pemisahan Training & Testing ELM

### 3.6 Training Data ELM

Setelah selesai pada proses normalisasi data dan pembagian data, tahap selanjutnya adalah melakukan proses training data yang kemudian dilanjutkan pada proses testing data. Dari proses training bertujuan untuk mendapatkan nilai hasil output bobot dan bias. Dimana dalam proses ini akan dilakukan beberapa langkah yaitu :




Gambar 3.4 Flowchat Training Data ELM

```
def train(self, X_train, y_train):
    if self.activation_func == "sigmoid":
        H = self.sigmoid(np.dot(X_train, self.weights) + self.bias)
    elif self.activation_func == "relu":
        H = self.relu(np.dot(X_train, self.weights) + self.bias)
    elif self.activation_func == "tanh":
        H = self.relu(np.dot(X_train, self.weights) + self.bias)
    self.beta = np.dot(np.linalg.pinv(H), y_train)
```

Gambar 3.5 Listing Program Training ELM

1. Inisialisasi nilai bobot dan bias yang di tentukan secara random.
2. selanjutnya menghitung hasil *hidden* pada proses feedforward dengan persamaan berikut.

$$H = Xt.W^T\ b$$

Keterangan :

$H$ = hasil keluaran *hidden* layer.

$Xt$ = hasil dari training

$W^T$ = hasil transpose input wight

$b$ = hasil nilai bias

3. Tahap berikutnya menghitung matrik dengan aktivitas sigmoid H(x)
4. Tahap selanjutnya menghitung Transpose matriks *hidden layer* menggunakan fungsi aktivasi rules $H^T$

5. Berikutnya menghitung nilai fungsi aktivasi rules

6.selanjutnya menghitung matrik *Moore-penrose generalized invers*

7. Langkah terakhir menghitung output *beban* dengan persamaan.

### 1.6.1 Penetuan Bobot, bias dan Beta Secara acak

Pada proses ini menentukan proses Training untuk memulai mencari Bobot, Bias dan beta.

```
self.input_size = input_size
self.hidden_size = hidden_size
self.activation_func = activation_func
self.weights = np.random.randn(input_size, hidden_size)
self.bias = np.random.randn(hidden_size)
```

Gambar 3.6 Listing Program  Penentuan Bobot bias dan betas ELM

Pada proses ini Bobot, bias dan betas akan di tentukan secara random dari data aktual yang ada setelah proses ini selesai akan di lanjutkan pada proses selanjutnya yaitu Testing.

## 3.7 Testing Data ELM

Pada tahap selanjutnya yaitu testing data yang sudah dihitung menggunakan matriks H dengan menggunakan bobot dan bias yang diperoleh  dari proses sebelumnya yaitu training. Untuk fungsi aktivasi yang digunakan menggunakan rumus yang sama seperti tahap training. Tujuan dari proses testing sendiri melakukan uji coba data dari proses training, sehingga dapat diketahui akurasi dari program.


Gambar 3.7 Flowchat Testing Data ELM

```
def predict(self, X_test):
    if self.activation_func == "sigmoid":
        H = self.sigmoid(np.dot(X_test, self.weights) + self.bias)
    elif self.activation_func == "relu":
        H = self.relu(np.dot(X_test, self.weights) + self.bias)
    elif self.activation_func == "tanh":
        H = self.tanh(np.dot(X_test, self.weights) + self.bias)
    y_predict = np.dot(H, self.beta)
    return y_predict
```

Gambar 3.8 Listing Program Testing ELM

### 3.8 MAPE

Setelah selesai pada proses sebelumnya dari proses algoritma ELM , maka selanjutnya akan di uji coba pada proses *Mean Absolute Percentage Error* (MAPE), pada pemrograman mape ini data aktual akan dijadikan data yang digunakan untuk melakukan seberapa jauh hasil *error* dari proses ELM yang di lakukan.

```
MAPE = abs(((Y_pred-Y_test)/Y_test)*100)/a
print('MAPE')
df = pd.DataFrame(MAPE, columns=['2014','2016','2017','2019','2020'])
```

Gambar 3.9 Listing program MAPE