BAB IV
HASIL DAN PEMBAHASAN

4.1 DATA *INPUT*

Data yang digunakan merupakan data sales pada runtun Bulan November 2017 s/d Mei 2021. Dataset ini berisi numerikal data yang menjelaskan detail sales yang terjadi pada kurun waktu per Bulan.

Pada metode ELM, data yang ada diolah dengan menentukan beberapa parameter kriteria, sehingga mencapai tingkat *error* yang kecil & akurasi yang optimal. Pada tabel 4.1 dijelaskan beberapa parameter yang akan digunakan dalam pengolahan dataset, diantaranya *input layer, output layer, hidden Neuron,* & fungsi aktivasi.

Tabel 4. 1 Parameter Kebutuhan Metode ELM

Parameter	**Jumlah**	**Deskripsi**
Input layer	6	Runtun Bulan, Data Stock (Label) & 4 Brand Historical sales
Output layer	1 *neuron*	Hasil Peramalan
Hidden layer	10 Pengujian	11 - 30 neuron
Fungsi aktivasi	1	*Sigmoid biner*

4.2 FORECAST METODE PENGEMBANGAN *EXTREME LEARNING MACHINE*

Langkah awal yang dilakukan adalah mencari nilai optimal dari beberapa parameter *input* yang diuji, hal ini dilakukan agar proses didalam metode ELM dapat menghasilkan prediksi yang baik saat dilakukan *training*. Setelah dilakukan pengujian MSE pada hidden neuron, selanjutnya dilakukan proses inti metode yaitu dengan me-normalisasikan dataset kedalam range 0 – 1, lalu dilakukan proses training data, testing data, denormalisasi data & dilakukan prediksi penjualan pada bulan selanjutnya.

4.2.1 Pengujian Jumlah Jaringan *Hidden Neuron*

Pengujian jumlah hidden neuron dilakukan untuk mengetahui pengaruh jumlah hidden neuron terhadap nilai evaluasi nilai akurasi dalam implementasi algoritma ELM. Jumlah hidden neuron yang akan digunakan pada pengujian ini antara lain 11, 12, 13, 14, 15, 17, 24, 26, 27, 30

```
# Menentukan Jumlah Hidden Layer
h = 12

# Nilai Bobot
  W1 = np.matrix(np.random.uniform(0, 1, (h, n)))
  W1 = np.random.uniform(0, 1, (h, n))

# Nilai Bias
b = np.random.uniform(0, 1, (1, h))

# Sigmoid Biner
def sig(x):
    return 1 / (1 + np.exp(-x))
```

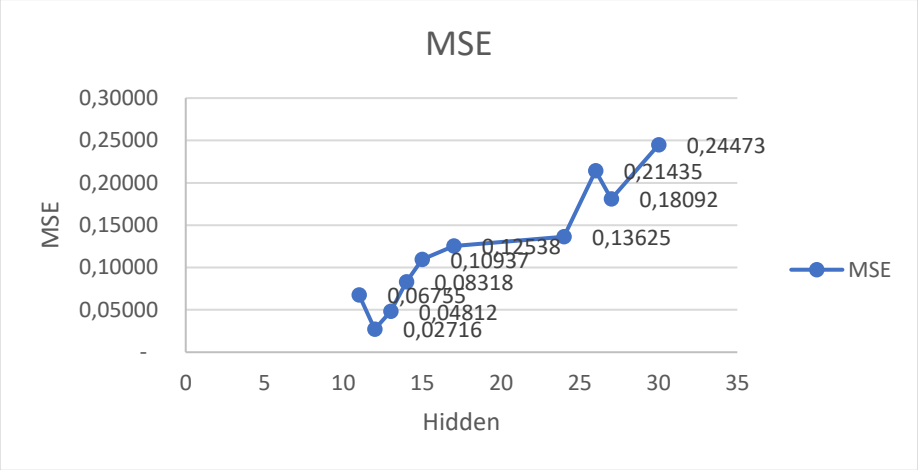
Gambar 4. 1 *Pseudocode* Uji *Hidden Neuron*

Pengujian dilakukan sebanyak 10 kali uji, jumlah jaringan hidden neuron diproses kedalam *pseudocode python*, pengujian pada jumlah neuron Hidden Neuron dilakukan secara berulang dengan mengganti jumlah neuron seperti tabel 4.2.

Tabel 4. 2 Hasil Pengujian Hidden Neuron

No	Jumlah Hidden	MSE	Waktu (detik)
1	**11**	0,06755	0,0515
2	**12**	**0,02716**	**0,0525**
3	**13**	0,04812	0,0528
4	**14**	0,08318	0,0545
5	**15**	0,10937	0,0526
6	**17**	0,12538	0,0532
7	**24**	0,13625	0,0550
8	**26**	0,21435	0,0548
9	**27**	0,18092	0,0545
10	**30**	0,24473	0,0547

Pada grafik gambar 4.1 adanya perubahan terjadi jika ada perubahan jumlah hidden layer. sehingga didapatkan nilai optimal jaringan adalah pada pengujian ke-2 yaitu dengan nilai *MSE* sebesar 0,02716 dengan waktu 0,00525 detik.



Gambar 4. 2 Plot Grafik *Hidden Network* Optimal

4.2.2 *Preprocessing*

Setelah jaringan hidden neuron dibuat, selanjutnya dilakukan *pre-processing* data. Data yang telah dikumpulkan dan dinormalisasi menggunakan normalisasi *Min-Max* atau ditransformasi ke dalam range 0 – 1. Hal ini dilakukan untuk mendapatkan nilai pada setiap variabel bantu.

Variable yang digunakan untuk proses masukan adalah variabel deret waktu per bulan, variabel stock dan variable penjualan berdasarkan brand. Penulis membagi data menjadi data *Testing* dan data *Training* melalui fungsi `train_test_split python` kemudian membandingkan kinerja model yang digunakan dalam melakukan peramalan dari data sales tersebut. Pembagian masing-masing data *Training* sebanyak 80% dan data *Testing* sebanyak 20%.

```
def ELM(x, csvarray, j):

 #Parameter Random Numpy yang digunakan dalam metode ELM
    np.random.seed(379)
```

```
    #Mengambil Data Fitur
    train_x = np.array(csvarray)[1:, 2:8]

    #Mengambil Data Target
    train_y = np.array(csvarray)[1:, 1]


    #Membagi data Training dan data Testing
    # 80% data training 20% data testing
    X_tr, X_te, y_tr, y_te = train_test_split(
        train_x.astype(float), train_y.astype(float), test_siz
e=0.2)
    X_tr_input = X_tr
    X_te_input = X_te
    y_tr_target = y_tr
    y_te_target = y_te

# Normalisasi
 std = np.std(X_tr, dtype=np.float64)
    transferdata = (X_tr - np.mean(X_tr))/(10*(std/(np.dot(2,
3.14))))
    X_tr = (1/(1+np.exp(-transferdata)))

    std = np.std(X_te, dtype=np.float64)
    transferdata = (X_te - np.mean(X_te))/(10*(std/(np.dot(2,
3.14))))
    X_te = (1/(1+np.exp(-transferdata)))

    m, n = X_tr.shape

    std = np.std(y_tr, dtype=np.float64)
    transferdata = (y_tr - np.mean(y_tr))/(10*(std/(np.dot(2,
3.14))))
    y_tr_bin = (1/(1+np.exp(-transferdata)))

    std = np.std(y_te, dtype=np.float64)
    transferdata = (y_te - np.mean(y_te))/(10*(std/(np.dot(2,
3.14))))
    y_te_bin = (1/(1+np.exp(-transferdata)))
```

Gambar 4. 3 *Pseudocode* Normalisasi & *Split* Dataset

Pada tabel 4.3 & tabel 4.4 dapat dilihat *output* data yang sudah terbagi / *split* menjadi 2 bagian dan membentuk range angka 0 – 1 data yang sudah disamakan rangenya ke-angka 0 – 1.

Tabel 4. 3 Pembagian Data *Training* 80% Setelah Normalisasi

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6
0,3552	0,3497	0,3589	0,3498	0,7480	0,7270
0,5395	0,3603	0,4630	0,3677	0,7573	0,7980
0,4781	0,3504	0,4128	0,3667	0,6723	0,7193
0,5268	0,3705	0,5047	0,3696	0,6141	0,6674
0,5049	0,3655	0,4704	0,3642	0,7085	0,5706
0,4635	0,3553	0,3978	0,3701	0,9262	0,6889

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6
0,5078	0,3709	0,4193	0,3669	0,5677	0,6189
0,4899	0,3538	0,4004	0,3754	0,6889	0,6091
0,4777	0,3507	0,3950	0,3784	0,6787	0,6886
0,4399	0,3614	0,4304	0,3541	0,5897	0,6004
0,4511	0,3546	0,3994	0,3606	0,7527	0,6141
0,4414	0,3625	0,4619	0,3610	0,6547	0,5677
0,4758	0,3505	0,4119	0,3642	0,7157	0,6713
0,4537	0,3506	0,3798	0,3679	0,6708	0,6994
0,5057	0,3518	0,3929	0,3873	0,7114	0,6157
0,4960	0,3660	0,4483	0,3751	0,6091	0,6229
0,4642	0,3492	0,3900	0,3710	0,7454	0,6838
0,4565	0,3532	0,4274	0,3662	0,6004	0,6152
0,4644	0,3634	0,4306	0,3645	0,6813	0,6149
0,4722	0,3555	0,4084	0,3745	0,6229	0,6813
0,4781	0,3703	0,4697	0,3596	0,6935	0,7085
0,4448	0,3570	0,4504	0,3636	0,6432	0,5754
0,4375	0,3494	0,3671	0,3833	0,7270	0,7254
0,4868	0,3598	0,4206	0,3821	0,6713	0,9262
0,4590	0,3533	0,4360	0,3651	0,6152	0,6547
0,4251	0,3489	0,3942	0,3684	0,7193	0,7157
0,4282	0,3561	0,4971	0,3613	0,5754	0,6646
0,4595	0,3654	0,4490	0,3643	0,7980	0,7527
0,4700	0,3495	0,3972	0,3666	0,6838	0,6723
0,4789	0,3494	0,4008	0,3781	0,7254	0,7114
0,4622	0,3492	0,3834	0,3693	0,7294	0,7454
0,5021	0,3523	0,4390	0,3694	0,6457	0,7310
0,4964	0,3505	0,3859	0,3760	0,6886	0,7294
0,4651	0,3507	0,3809	0,3786	0,6157	0,6708

Tabel 4. 4 Pembagian Data *Testing* 20% Setelah Normalisasi

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6
0,48451	0,35353	0,42715	0,34968	0,69842	0,72659
0,43501	0,35957	0,42615	0,34449	0,64512	0,36723
0,45851	0,34328	0,40150	0,36011	0,69533	0,67470
0,46529	0,36314	0,48058	0,46193	0,76199	0,76036
0,47845	0,34694	0,45601	0,35457	0,76622	0,79335
0,49427	0,36567	0,50819	0,35478	0,59095	0,61250
0,48471	0,35135	0,46971	0,36826	0,64059	0,67189
0,52766	0,34101	0,40406	0,37653	0,73279	0,71063
0,49897	0,35415	0,41395	0,35304	0,36723	0,78375

4.2.3 *Training*

Proses *Training* dilakukan dengan menggunakan 80% dataset. Tabel 4.5 merupakan hasil data yang sudah dinormalisasikan dan dibentuk suatu tabel *output training* dengan ordo 12 x 6 (12 jaringan Hidden Layer & 6 Fitur Masukan).

```
   # Menentukan Nilai Bobot
    # W1 = np.matrix(np.random.uniform(0, 1, (h, n)))
    W1 = np.random.uniform(0, 1, (h, n))


    # Menentukan Nilai Bias
    b = np.random.uniform(0, 1, (1, h))

    #Menghitung Sigmoid Biner
    def sig(x):
        return 1 / (1 + np.exp(-x))

    # output weight
    H = (X_tr @ W1.T)+b

    W2 = np.linalg.pinv(sig(H)) @ y_tr_bin.reshape(-
1, 1)
    # print(W2)
```

Gambar 4. 4 Pseudocode Menentukan Nilai Bias, *Output Weight & Moore Penrose Inverse*

Tabel 4. 5 *Training* Ordo 12 x 6

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6
0,0558	0,8430	0,6263	0,4151	0,6627	0,0252
0,0896	0,6381	0,5356	0,7703	0,3389	0,7839
0,1758	0,3225	0,5412	0,7633	0,4020	0,5811
0,4652	0,5609	0,1406	0,4446	0,7704	0,6104
0,5294	0,6224	0,4036	0,7573	0,0588	0,1303
0,6689	0,3863	0,4126	0,8024	0,8661	0,5111
0,7164	0,6305	0,5334	0,3697	0,0297	0,4240
0,7621	0,1383	0,3566	0,2204	0,9654	0,0216
0,8333	0,2910	0,8613	0,3733	0,3512	0,2307
0,8409	0,7427	0,2408	0,0436	0,0281	0,1970
0,9815	0,1281	0,4837	0,3220	0,0628	0,5891
0,9848	0,5148	0,9862	0,9852	0,2135	0,8389

Data *training* yang telah dirandom secara acara akan menghasilkan data berupa keluaran *weight* yang akan dibutuhkan pada saat proses *testing* data.

Tabel 4. 6 *Output Weight*

Output Weight (Beta)
419,442941
414,794705
713,993594
-149,313835
-473,334959
30,800049
-260,807955
-146,883101
-248,036551
-35,957942
-295,455812
32,925058

4.2.4 *Testing*

Proses *testing* pada tahap ini bertujuan untuk mengukur performa model jaringan yang telah dibangun pada proses training. Langkah-langkah yang digunakan sama dengan proses *training*. Akan tetapi, pada proses *testing* seluruh bobot diambil dari hasil training. Sehingga dalam proses testing tidak ada lagi penghitungan bobot β. Data yang digunakan juga berbeda dengan data yang digunakan pada proses training sebagaimana yang telah dijelaskan pada tahap *prepocessing* data. Setelah itu, tingkat akurasinya dihitung dengan cara yang sama seperti pada proses training.

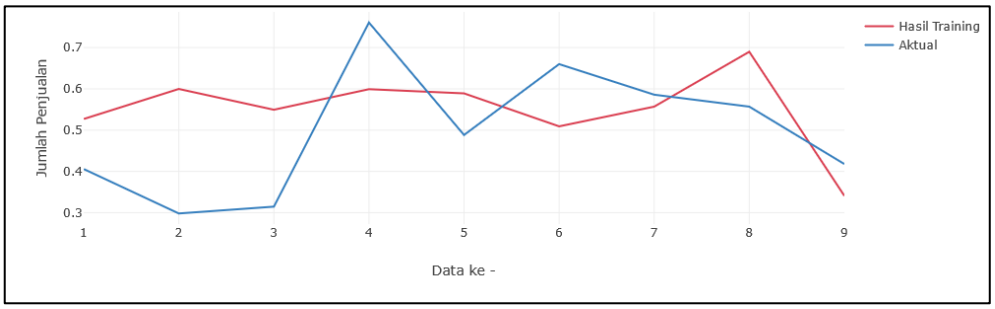
Tabel 4. 7 *Output Data Testing*

Fitur 1	Fitur 2	Fitur 3	Fitur 4	Fitur 5	Fitur 6
0,4845	0,3535	0,4272	0,3497	0,6984	0,7266
0,4350	0,3596	0,4261	0,3445	0,6451	0,3672
0,4585	0,3433	0,4015	0,3601	0,6953	0,6747
0,4653	0,3631	0,4806	0,4619	0,7620	0,7604
0,4785	0,3469	0,4560	0,3546	0,7662	0,7934
0,4943	0,3657	0,5082	0,3548	0,5909	0,6125
0,4847	0,3513	0,4697	0,3683	0,6406	0,6719
0,5277	0,3410	0,4041	0,3765	0,7328	0,7106
0,4990	0,3542	0,4139	0,3530	0,3672	0,7837

Pada tabel 4.9 dijelaskan bahwa jika dibandingkan target data *testing*, hasil prediksi yang dihasilkan dari proses *training* menghasilkan rata-rata prediksi sebesar 111%.

Tabel 4. 8 *Output* Target *Testing* vs Hasil Prediksi *Testing*

Data Ke-	**20% data target (Y)**	**Hasil Prediksi (y) Akurasi**	
Data ke-1	0,4057	0,5270	130%
Data ke-2	0,2980	0,5998	201%
Data ke-3	0,3145	0,5494	175%
Data ke-4	0,7608	0,5992	79%
Data ke-5	0,4882	0,5891	121%
Data ke-6	0,6600	0,5091	77%
Data ke-7	0,5859	0,5569	95%
Data ke-8	0,5570	0,6899	124%
Data ke-9	0,4179	0,3409	82%
Avearge	**0,4987**	**0,5513**	**111%**



Gambar 4. 5 Plot Testing

```
 pred = (sig((X_te @ W1.T)+b)@W2)

    MSEnumpy = np.square(np.subtract(y_te_bin.reshape(-
1, 1), pred)).mean()

    mse = mean_squared_error(y_te_bin.reshape(-
1, 1), pred)

    # pred = np.squeeze(np.asarray(pred))

    pred[pred < 0] = 0
    pred[pred > 1] = 1
```

Gambar 4. 6 *Pseudocode Target Testing*

4.2.5 Denormalisasi

Pada tahap denormalisasi, data *testing* yang sudah diprediksi dalam range output 0 – 1 diubah kedalam nilai sebenarnya atau dikonversi dalam satuan Kilogram sehingga data target dan hasil prediksi dapat dibaca dalam skala baca yang lebih luas melalui nilai aslinya.

```
pred_denormalilsasi = ((-(np.log((1/pred)-
1)))*((10*(std/(np.dot(2, 3.14))))))+np.mean(y_te)
    pred_denormalilsasi = np.round(pred_denormalilsasi)
```

Gambar 4. 7 *Pseudocode* Denormalisasi

Data hasil denormalisasi pada tabel 4.10 merupakan hasil dari 9 data yang ditest dan diprediksi untuk mengatahui perbandingan dari masing-masing akurasi per barisnya. Dan didapatkan akurasi dari rata-rata baris data akhir adalah sebsar 101 %.

Tabel 4. 9 Denormalisasi Target *Testing*

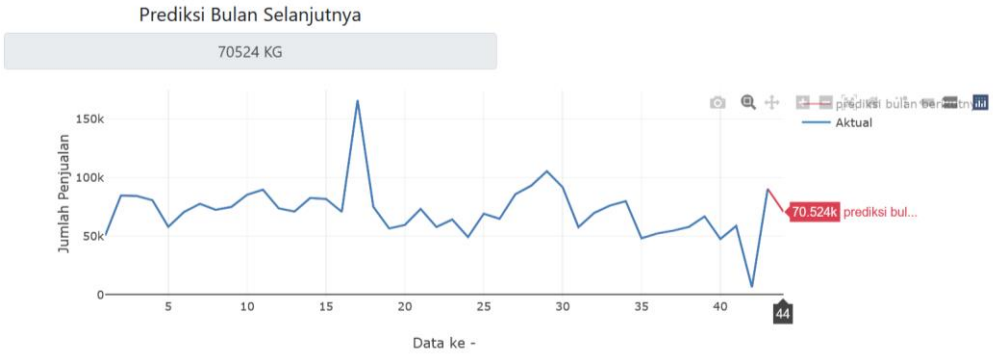
Data	**20% data target (Y)**	**Hasil Prediksi (y)**	%
Data ke-1	57.521	67.931	118%
Data ke-2	47.425	74.235	157%
Data ke-3	49.077	69.851	142%
Data ke-4	90.227	74.183	82%
Data ke-5	64.634	73.293	113%
Data ke-6	79.734	66.413	83%
Data ke-7	73.014	70.497	97%
Data ke-8	70.499	82.638	117%
Data ke-9	58.587	51.616	88%
Average Data ke-7 sd 9	**67.367**	**68.250**	**101%**

4.2.6 Hasil Prediksi Bulan Selanjutnya

Dari hasil prediksi yang dilakukan melalui *Pseudocode Python* nilai yang diprediksi pada pada data ke-44 atau bulan selanjutnya dari dataset yang diproses adalah sebesar 70,524 Kilogram

```
parameter = np.array(csvarray)[1, 2:8]
    parameter = parameter.astype(float)

    stdparameter = np.std(parameter, dtype=np.float64)

    transferdata = (parameter - np.mean(parameter)) / \
        (10*(stdparameter/(np.dot(2, 3.14))))
    parameter = (1/(1+np.exp(-transferdata)))

    parameter_penjualan = (sig((parameter @ W1.T)+b)@W2)

    parameter_denormalilsasi = ((-
(np.log((1/parameter_penjualan)-
1)))*((10*(std/(np.dot(2, 3.14))))))+np.mean(y_te)
    parameter_denormalilsasi = np.round(parameter_denorma
lilsasi)
```

Gambar 4. 8 *Pseudocode* Prediksi Bulan Selanjutnya



Gambar 4. 9 Plot Hasil Prediksi Bulan Selanjutnya

Dari hasil plot gambar 4.8 dapat dilihat garis merah pada sumbu y menjelaskan sales yang akan terjadi pada data ke-44 akan mengalami penurunan sales dari bulan sebelumnya.

4.3 *FORECAST EXIST* di PT. XYZ

PT. XYZ memiliki suatu *forecast* sales untuk mengetahui sales yang akan terjadi pada dalam waktu 3 bulan kedepan. Hal ini diketahui melalui suatu analisa *traditional time series forecasting* menggunakan *tools* Microsoft Excel. Pada tabel 4.11 dijabarkan perbandingan forecast yang dibuat dan aktual yang terjadi, dapat dilihat pada gambar 4.11 dari total peramalan terdapat adanya *gap* dan terdapat nilai akurasi sebesar 86%.

Tabel 4. 10 *Forecast Existing*

Brand	Maret	April	Mei	Total
Promina	8.996	751	15.558	8.435
SUN	36.524	3.577	33.461	24.520
Govit	1.439	122	1.962	1.174
Gowell	2.844	1.233	21.270	8.449
Total	**49.802**	**5.682**	**72.251**	**42.579**

Tabel 4. 11 Aktual Sales

Brand	Maret	April	Mei	Total
Promina	9.697	904	22.218	10.940
SUN	36.771	3.708	54.311	31.597
Govit	1.108	64	1.348	840
Gowell	11.010	1.553	12.349	8.304
Total	**58.587**	**6.229**	**90.227**	**51.681**

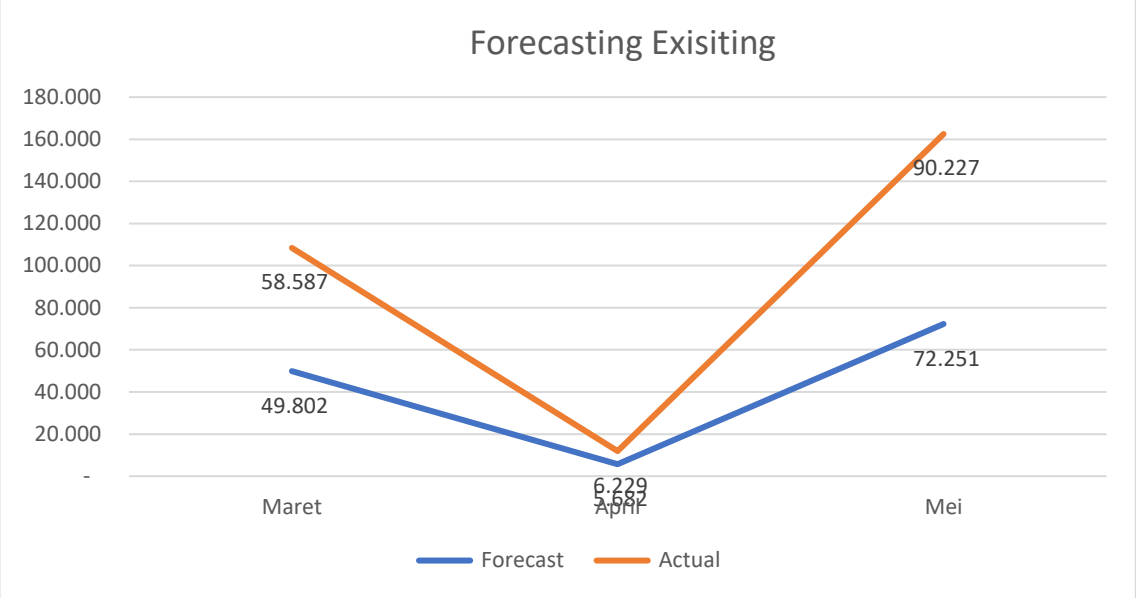
Tabel 4. 12 Perbandingan *Forecasting Exist* vs Aktual

Brand	Maret	April	Mei	*Average*
Forecast	49.802	5.682	72.251	**42.579**
Actual	58.587	6.229	90.227	**51.681**
Akurasi	**85%**	**91%**	**80%**	**82%**

Diketahui

Forecast : Hasil yang peramalan yang dilakukan oleh Perusahaan PT. XYZ dalam satuan volume Kilogram
Aktual : Sales yang terjadi pada Bulan Maret, April & Mei
Akurasi : *Forecast* / Aktual Sales

Dari Gambar 4.10 diperoleh informasi bahwa *Forecasting* yang telah dibangun perusahaan memiliki tingkat akurasi yang fluktuative pada Bulan Maret tingkat akurasi sebesar 85%, pada Bulan April 91% dan pada Bulan Mei tingkat akurasinya sebesar 80%.



Gambar 4. 10 Plot *Forecast Existing*

Dapat diambil *learning* juga terhadap rata-rata data pada 3 Bulan hasil prediksi, bahwa rata-rata akurasi yang dihasilkan hanya sebesar 82%. Plot akurasi yang kurang dari aktual menandakan peringatan kepada perusahan bahwa tidak ada stok yang cukup untuk mensuplai penjualan selanjutnya yang berarti penjualan akan tidak maksimal.

4.4 PERBANDINGAN METODE *EXSIT* VS PENGEMBANGAN *EXTREME LEARNING MACHINE*

4.4.1 Matrix Perbandingan Metode ELM vs *Exist*

Dari proses *Extreme Learning Machine* yang sudah dilakukan, dapat dibuat suatu matrix perbandingan antara metode *Extreme Learning Machine* dan Metode *Exist* sebagai berikut :

Tabel 4. 13 Perbandingan Metode ELM vs Metode *Exist*

Matrix	Metode Pembanding	Average Satuan Kg	Penjelasan
Forecast	***Exist* (Tradisional)**	42.579	Nilai rata-rata 3 Bulan *forecast* yang dilakukan perusahaan melalui metode *exist* atau tradisional adalah 42.579 Kilogram. nilai ini didapatkan dari tabel 4.11 .
	ELM	68.250	Nilai rata-rata 3 Bulan *forecast* yang dilakukan melalui metode *ELM* adalah 68.250 Kilogram. nilai ini didapatkan dari tabel 4.12
Aktual *Average* 3 Bulan		51.681	Nilai rata-rata 3 Bulan yang digunakan sebagai pembanding aktual sales adalah sebesar 51.681 Kilogram, nilai ini didapatkan dari tabel 4.12
Akurasi	***Exist* (Tradisional)**	82%	**Perbandingan Nilai Forcast Exist / Tradisional dibandingkan Aktual.** Nilai yang dhitung menggunakan Ms. Excel mendapatkan nilai akurasi yang relative baik namun belum mencapai nilai optimal, hal ini dikarenakan *forecast* yang dihitung masih menerapkan estimasi manual sebagai dasar ukur dengan melakukan pembagian rata-rata sales per Bulan dan melakukan penambahan perkiran sales sesuai pertumbuhan perkiraan
	ELM	132%	**Perbandingan Nilai Forcast ELM dibandingkan Aktual.** Sesuai dengan metode pembuktiaan yang dijelaskan pada pembahasan sebelumnya nilai akurasi metode ELM dapat mencapai nilai akurasi yang sangat baik, hal ini dikarenakan adanya *feature stock* yang menjadi label acuan perhitungan dan dilakukan beberapa kali uji untuk mendapatkan nilai error terkecil serta akurasi terbaik Beberapa *feature* tersebut membantu dalam memberikan rangsangan untuk mencapai akurasi optimal