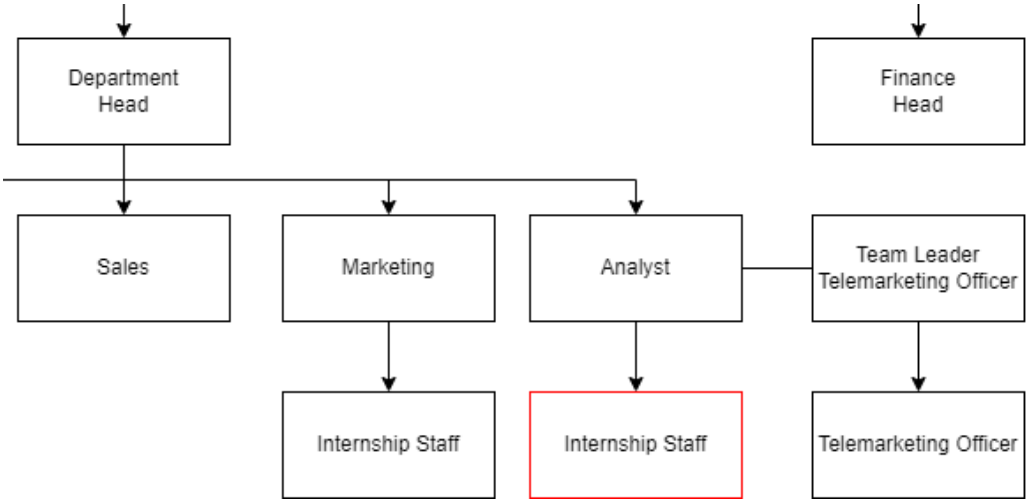
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3. 1 Bagan Penempatan Mahasiswa Magang

Posisi magang sebagai *data analyst intern* berada pada departemen Setir Kanan di dalam divisi *Integrated Supply Chain*. Selama kegiatan program magang dilaksanakan, Bapak Christian Eka selaku *Department Head* berperan sebagai *supervisor*. Bapak Christian Eka juga merupakan pihak yang memberikan tugas dan proyek untuk dilaksanakan oleh peserta magang. Hasil proyek yang dikerjakan guna mengetahui *customer behaviour* sehingga dapat membantu dalam meningkatkan penjualan yang nantinya hasil tersebut akan dikirimkan kepada *Telemarketing Officer*

3.2 Tugas dan Uraian Kerja Magang

Data Analyst internship staff dalam pekerjaannya bertugas dalam menjalankan proses analisis dan prediksi data, hal yang pertama dilakukan yaitu memperoleh data dari divisi yang bertugas dalam menyimpan database perusahaan, membersihkan data, modelling, dan visualisasi data. Seluruh proses dalam pelaksanaan pekerjaannya, data analyst intern menggunakan *framework Cross Industry Standard Process for Data Mining* atau lebih dikenal sebagai *framework* CRISP-DM dengan Python sebagai bahasa yang digunakan dalam proses

pemrograman. Selama proses kegiatan kerja magang berlangsung, *data analyst intern* mengerjakan proyek-proyek yang telah direncanakan *user* dengan tujuan dari proyek tersebut yang berbeda-beda. Selain dari pengerjaan proyek yang ditugaskan, *data analyst intern* juga mendapat pembelajaran berupa pengalaman guna mendapatkan *soft skills* dan mempelajari cara kerja perusahaan mulai dari supply sampai purchasing by customer. Kegiatan program magang terdiri dari 3 fase, yaitu pengenalan, pelaksanaan/pengerjaan proyek, dan presentasi.

Tabel 3. 1 Uraian Kerja Magang

NO	Pekerjaan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
1	Pengenalan dan *on boarding*.	1	27-02-2023	04-03-2023
2	Pengenalan sistem dan *software* yang digunakan perusahaan.	2-3	06-03-2023	25-03-2023
3	Pengerjaan proyek 1: *eksploratory data analysis* (EDA)	4-6	27-03-2023	08-04-2023
4	Pengerjaan proyek 2: Pengerjaan model *forecasting sales*	7-10	10-04-2023	06-05-2023
5	Pengerjaan proyek 3: *Clustering* jenis mobil berdasarkan frekuensi penjualan dan keuntungan	11-14	08-05-2023	03-06-2023
6	Pengerjaan proyek 4: Prediksi harga jual mobil berdasarkan spesifikasi mobil	15-18	05-06-2023	30-06-2023

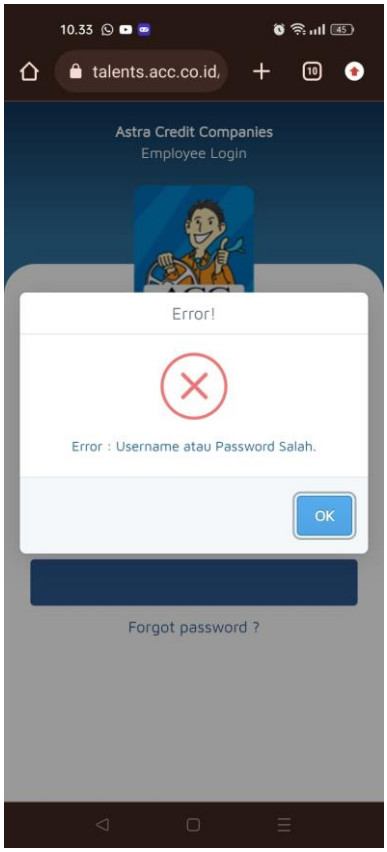
3.2.1 Minggu 1: Pengenalan dan *on boarding*

Proses kerja magang diawali dengan melakukan pengenalan selama proses *on boarding* yang diatur oleh HC Astra Credit Companies pada tanggal 27 Februari 2023. Perusahaan diperkenalkan dengan diadakan *tour* untuk melihat-lihat dari dua gedung kantor utama yang berlokasi di Tanjung Barat, TB Simatupang, Jakarta Selatan. Pihak perusahaan memperkenalkan ruangan serta fungsinya dan ruangan yang digunakan untuk bekerja oleh peserta *intern*, selain itu diperkenalkan web “Talents.co.acc.id” bagi para karyawan untuk melakukan

berbagai aktivitas dalam web perusahaan tersebut, seperti melihat gaji, *claim* lembur, asuransi dan juga presensi.

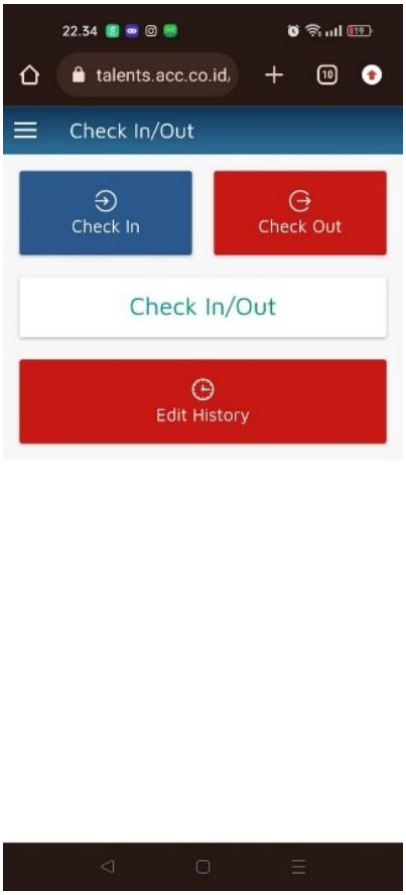
Tabel 3. 2 Tampilan Menu login

Gambar 3. 2 Tampilan Login Employee



Gambar 3. 3 Tampilan Error Login

Pada gambar 3.2 merupakan tampilan login bagi para karyawan untuk melakukan berbagai aktivitas dalam web perusahaan Astra Credit Companies. Terdapat tampilan *forgot password* atau lupa kata sandi yang berguna bagi para karyawan yang lupa akunnya. Ketika karyawan memasukkan *email* atau *password* salah, maka akan tampil output error seperti pada gambar 3.3 sehingga harus memasukkan ulang akunnya.

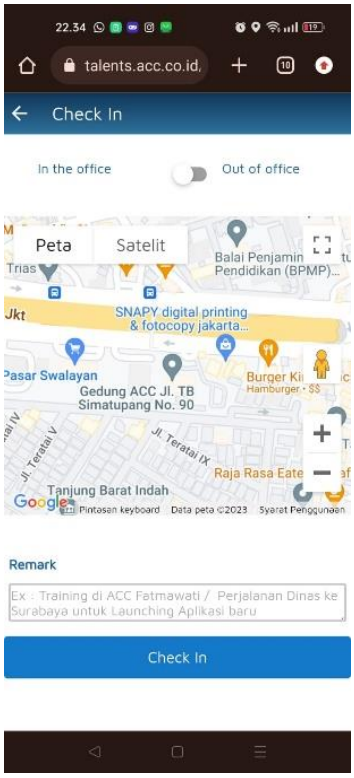


Gambar 3. 4 Presensi Check-in dan Check-out

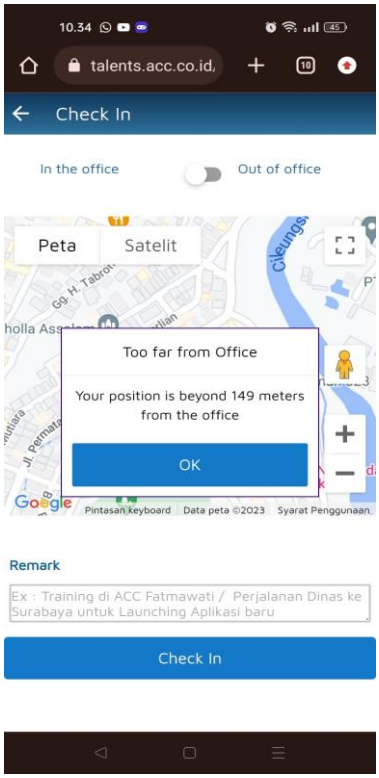
Gambar 3.4 ini menunjukkan menu yang digunakan untuk presensi yaitu menu *check-in* pada saat masuk jam kerja di pagi hari dan *check-out* pada saat jam kerja telah selesai. Pada saat melakukan presensi *check-in* hanya bisa dilakukan di kantor dengan menggunakan *location map*. Jika ingin mengubah *history* presensi, maka dapat klik *edit history* dengan menyertakan alasan dan juga *approval* dari atasan.

Gambar 3.5 tampilan saat *check-in* ingin dilakukan yaitu dengan *tracking location* karyawan. Proses *check-in* hanya dapat dilakukan di dalam wilayah kantor perusahaan. Sistem akan membaca lokasi dari karyawan, sehingga ketika *check-in* ingin dilakukan tidak di dalam wilayah kantor maka akan mengeluarkan *output* seperti pada gambar 3.6 dengan kegagalan untuk melakukan presensi sehingga karyawan harus melakukannya di dalam wilayah kantor.

Tabel 3. 3 Tampilan Check-in by Location



Gambar 3. 5 Tampilan ketika Check-In



Gambar 3. 6 Tampilan ketika Check-In Tidak Dilakukan di Kantor

3.2.2 Minggu 2-3: Pengenalan sistem dan *software* yang digunakan perusahaan

Pada perusahaan Setir Kanan – Astra Credit Companies, *database* yang dimiliki masih terpusat ke kantor gedung utama Astra Credit Companies sehingga segala keperluan data harus mengajukan terlebih dahulu. Ketika data sudah diperoleh dalam bentuk ASCII yang dimuat dalam Excel, maka data dapat digunakan untuk diolah.

Excel digunakan karena selain tampilannya yang sudah familiar di mata masyarakat awam, excel memang sudah menjadi salah satu *tools* yang diandalkan karena kemudahan dalam penggunaannya. Excel merupakan lembar kerja Microsoft yang berbentuk kumpulan sel yang disusun menjadi baris dan kolom. Data yang terdapat di lembar kerja dapat dihitung dan diolah secara akurat yang dapat dibantu dengan rumus-rumus yang memudahkan bagi penggunanya. Excel juga dapat memvisualisasikan hasil olahan data seperti

tabel, diagram dan grafik bergaris namun masih terdapat keterbatasan dalam pengolahan data, sehingga dibutuhkan *software* untuk pengolahan data yang lebih *advance.* [10] Data yang digunakan untuk proyek adalah data Sales Setir Kanan 2022.

Dalam pengolahan data, tidak ada *software* khusus dari perusahaan yang harus digunakan sehingga dapat dilakukan dengan langsung menggunakan Microsoft Excel ataupun *software* pengolahan data lain seperti Jupyter *Notebook* dan Visual Studio Code dan dalam pengerjaan proyek yang dilakukan *framework* yang dipilih yaitu adalah *Framework Cross Industry Standard Process for Data Mining* (CRISP-DM).

CRISP-DM adalah model yang terstandarisasi dan bersifat terbuka dan fase yang dimiliki terstuktur, dapat didefinisikan secara efisien, fleksibel sesuai kebutuhan, dan sederhana namun tetap memeunhi *standard* kebutuhan *data analyst* sehingga ini merupakan salah satu *framework* yang banyak digunakan untuk proses *data mining*. CRISP-DM menurut IBM yaitu adalah tahapan proses yang dipercaya industri untuk menjadi panduan dalam *data mining*. [11]

Visual Studio Code merupakan *tools* yang digunakan untuk pembuatan *code* buatan Microsoft. VSC memiliki fitur dan ekstensi dalam pembuatan *code* yang lengkap sehingga menjadi pilihan utama untuk para *developer* serta *tools* ini dipilih karena mendukung semua sistem operasi seperti Windows, Mac OS dan Linux sehingga memudahkan penggunaan *device* dalam pembuatan *code.* VSC juga dibuat agar dapat digunakan dengan ringan dan nyaman sehingga tidak perlu memiliki perangkat yang memiliki spesifikasi terlalu tinggi. *Tools* ini memiliki *built-in* dukungan untuk JavaScript dan memiliki array beragam *ekstensi,* salah satunya bahasa Python. [12]

Pemilihan bahasa pemrograman Python yaitu karena bahasanya yang memiliki sifat *open source.* Python dapat digunakan untuk menjalankan proses analisis data hingga evaluasi sehingga membantu *data analyst* untuk melakukan *improvement.* Selain itu bahasa pemrograman yang dapat digunakan dalam OOP

atau *Object-Oriented Programing* ini juga dapat digunakan untuk membuat *script* dan API yang dibutuhkan oleh pengguna. [13] Python digunakan pada *environment* Jupyter Notebook karena dapat menampung *library* dan fungsi yang akan dipanggil pada saat pengkodean berjalan. Jupyter Notebook digunakan dalam pengerjaan karena alasan kenyamanan dan kebiasaan berdasarkan preferensi pribadi.

3.2.3 Minggu 4-6: Pengerjaan proyek 1: *eksploratory data analysis* (EDA)

Data Analyst dalam pekerjaannya bertugas untuk melakukan analisis, membuat *data modeling* dan membantu instansi dalam mengimplementasikan *data analyst* sesuai *request*. Dalam mengerjakan tugas, *data* analyst dibagi pekerjaannya yang dimuat dalam beberapa proyek dan memiliki tenggat waktu dalam tiap proyek. Proyek tersebut diberikan oleh user berdasarkan kebutuhan.

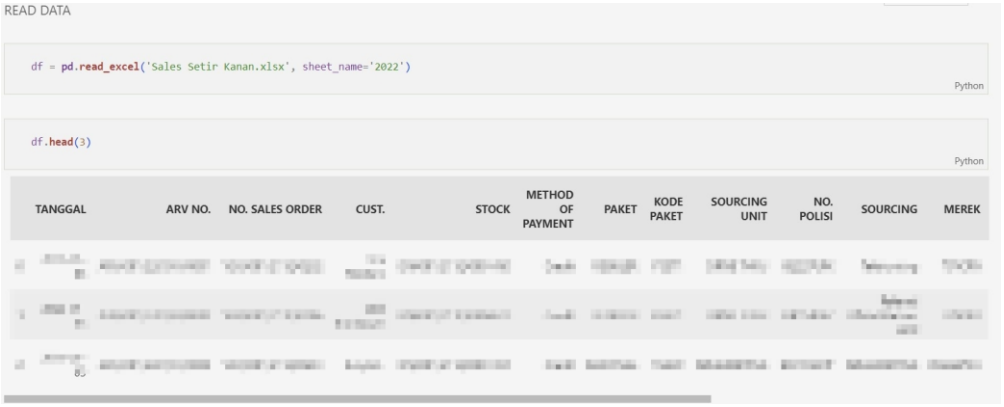
Mentor selaku user secara langsung menugaskan proyek setelah proses *on boarding* dilakukan untuk memvisualisasikan *top ranking* beberapa variabel untuk mengetahui visualisasi data yang dimiliki perusahaan tersebut. Proyek ini bertujuan agar dapat membantu menentukan target penjualan berikutnya mulai dari produk yang dijual dan target customernya. Proyek ini dikerjakan dalam rentang waktu 27 Maret 2023 – 8 April 2023.

Gambar 3. 7 Import Library

Dalam pengolahan data, setelah mendapatkan data yang diolah maka dapat melakukan proses *coding*. Hal pertama yang dilakukan yaitu adalah *import library* yang dibutuhkan untuk membantu proses pengolahan data, beberapa *library* yang sering digunakan contohnya yaitu adalah Pandas yang berguna untuk analisis, modifikasi dataframe, memanipulasi tabel yang berbentuk

numerik atau angka dan *time series* dan juga menambahkan ataupun menghapus kolom dan *replace value*. [14]

Selanjutnya ada Numpy merupakan *library* yang berfungsi untuk memungkinkan *user* memanipulasi data umum, perhitungan angka-angka seperti aljabar linier, matriks, dan transformasi Fourier.[15] Library selanjutnya yaitu adalah Matplotlib yang berfungsi untuk membuat suatu *plots, histograms, barchart, scatterplot, soectra* agar lebih mudah dilakukan.[16] Selanjutnya merupakan *library* Seaborn yaitu adalah *library* yang berfungsi untuk membuat grafik dan statistik dengan menggunakan Python yang dibangun berdasarkan library matplotlib yang datanya terintegrasi dengan terstruktur pada pandas. Seaborn juga bertujuan agar hasil yang ditampilkan dari visualisasi data dapat meningkatkan dalam memahami suatu data yang diolah. Berdasarkan hal tersebut, *library* Seaborn berorientasi pada fungsi *plotting* berdasarkan dataset yang diproses pada dataframe dan array untuk menghasilkan grafik yang informatif. [17]



Gambar 3. 8 Read Data Excel

Selanjutnya, *read* data excel yang sudah diberikan dan membuat variabel yang berguna untuk memanggil data excel yang ingin diolah beserta *sheet name*-nya. Kemudian *show* tabel untuk mengecek tabel data, menggunakan *head* untuk melihat data teratas dan angka tiga melambangkan 3 baris data saja yang ditampilkan.

```python
df.isnull().sum()
```

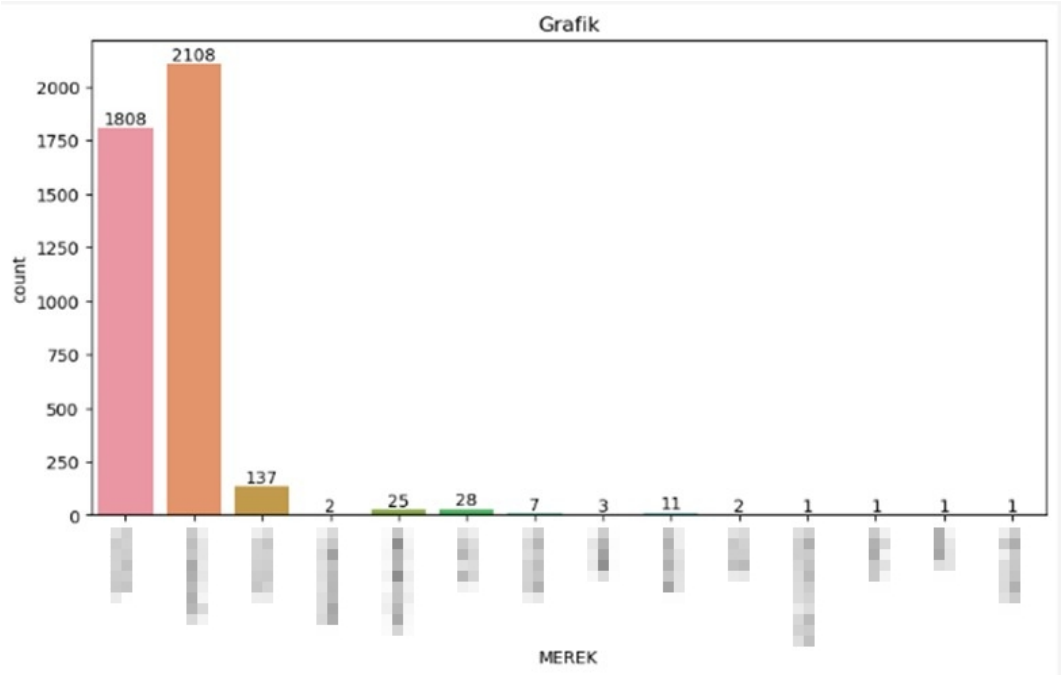
```
TANGGAL                    0
ARV NO.                    0
NO. SALES ORDER            0
CUST.                      0
STOCK                      0
METHOD OF PAYMENT          0
PAKET                      0
KODE PAKET                 0
SOURCING UNIT              0
NO. POLISI                 0
SOURCING                   0
MEREK                      0
MODEL                      0
WARNA                      0
TIPE UNIT                  0
TAHUN                      0
TIPE UNIT.1                0
HARGA MODAL                0
HARGA JUAL (INC. TAX 10%)  0
dtype: int64
```

Gambar 3. 9 Mengecek Nilai Null pada Data

Langkah yang dilakukan selanjutnya yaitu adalah mengecek apakah terdapat data kosong atau *null* dalam dataset yang akan diolah, karena jika terdapat data *null* maka dapat membuat suatu visualisasi ataupun pemodelan data tidak sesuai dengan yang diinginkana. Ketika terdapat data yang memiliki nilai *null* maka dapat dihapus ataupun menghitung nilai pengganti.

Gambar 3.10 merupakan visualisasi data dengan merek terbanyak yang terjual pada tahun 2022 pada Sales Setir Kanan. Visualisasi menggunakan *library* Matplotlib dengan Sumbu X menunjukkan data merek mobil yang terjual, sedangkan sumbu Y merupakan jumlah unit yang terjual. Ini menandakan 2 variabel paling kiri merupakan mayoritas merek yang terjual terbanyak dengan jumlah 2108 dan 1808 unit.

```
plt.figure(figsize=(10,5))
ax = sns.countplot(x='MEREK', data=df)
ax.bar_label(ax.containers[0])
plt.title('Grafik')
plt.xticks(rotation=90)
plt.show()
```



Gambar 3. 10 Visualisasi Merek Terbanyak

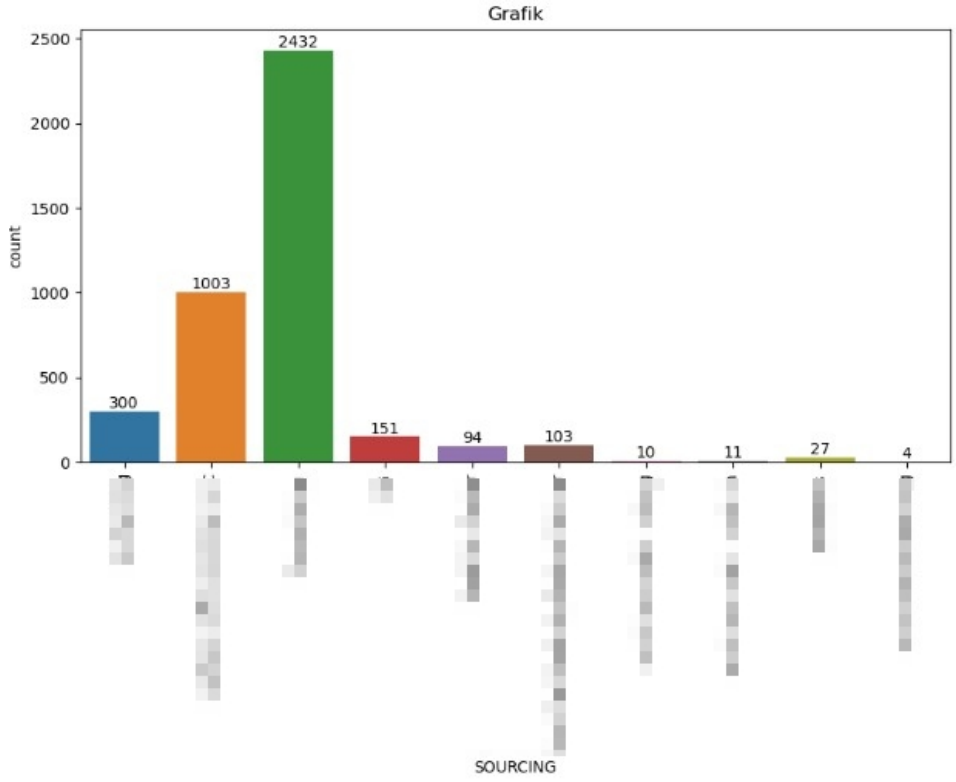
```
df.replace({'REMARKETING':'Remarketing'}, inplace=True)
```

Python

Gambar 3. 11 Replace Data Redudancy

Dalam data Sales Setir Kanan, terdapat data redundancy yang dimiliki dikarenakan penulisan variabel yang berbeda dalam suatu kolom. Untuk mengatasi hal ini maka diperlukan *command replace* agar dapat disesuaikan menjadi satu data yang sama. Setelah data di-*replace* dengan menjadi suatu data yang sama sehingga data yang dimiliki tidak terpisah dengan perbedaan penulisan.

```
plt.figure(figsize=(10,5))
ax = sns.countplot(x='SOURCING', data=df)
ax.bar_label(ax.containers[0])
plt.title('Grafik')
plt.xticks(rotation=90)
plt.show()
```



Gambar 3. 12 Visualisasi Jumlah Sourcing

Gambar 3.12 merupakan visualisasi berupa grafik *bar chart* untuk melihat jumlah *count* pada *sourcing* atau sumber unit yang terjual. Dapat dilihat bahwa *bar* berwarna hijau merupakan *sourcing* unit yang paling banyak terjual, terdapat perbedaan yang signifikan dibandingkan dengan sumber unit lainnya. Dari 10 sumber unit yang dimiliki Setir Kanan, hanya dua sumber yang memiliki jumlah terjual lebih dari seribu penjualan.

```
customer_count = df['CUST.'].value_counts()
```

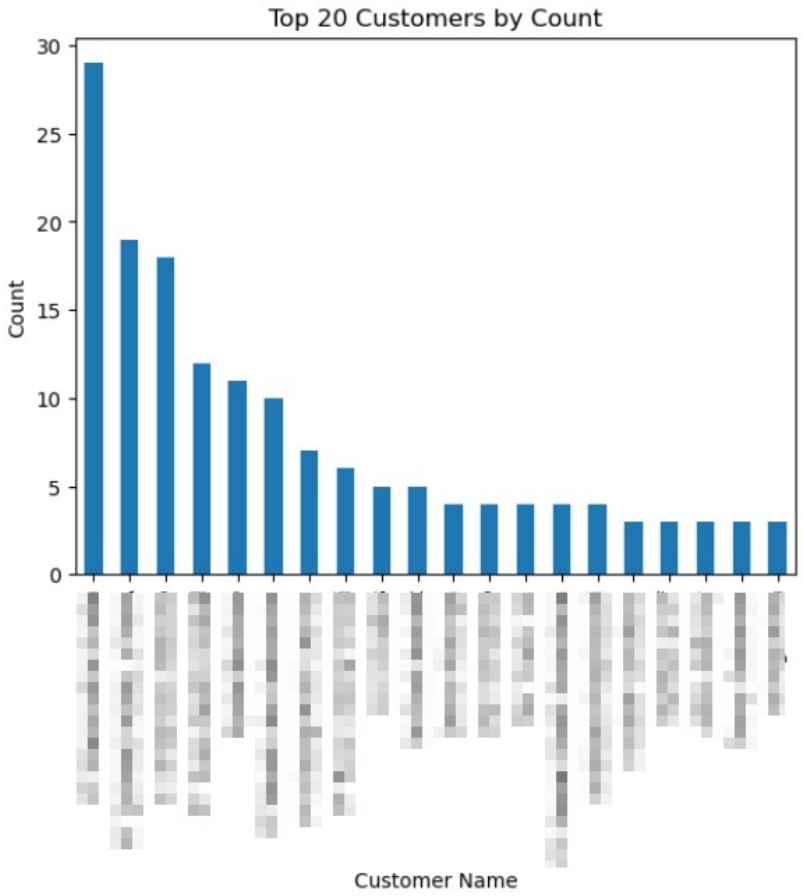
```
top20_customers = customer_count.head(20)
print("Top 20 customers:")
for customer, count in top20_customers.items():
    print(customer, count)
```

```
Top 20 customers:
PT Stacomitra Graha 29
PT ZIRANG MOBIL UTAMA 19
Ir. Antonious Wibowo 18
PT Trans Pasific Global 12
Djoko Martono 11
PT Mobil Laku Indonesia 10
PT Zirang Mobil Utama 7
PT Balai Lelang Serasi 6
Eric Charles 5
Sri Budi Astutik 5
Bonar Ritonga 4
Dwi Hariyatno 4
Nur Alam, SE 4
PT. Kawan Mobil Nusantara 4
Andriana Puspitasari 4
I Kadek Candiasa 3
Mahfud Arief 3
Arbie Fathony 3
Derry Setiawan 3
Sri Agustina 3
```

Gambar 3. 13 Top 20 Customer

Dalam *dataset* Sales Setir Kanan, data dapat diolah untuk menunjukkan jumlah pembelian unit mobil yang dilakukan tiap customer sepanjang tahun 2022. Dalam data tersebut, customer tidak hanya melakukan pembelian sekali saja namun beberapa kali sehingga visualisasi dapat menunjukkan berapa jumlah pembelian yang dilakukan oleh masing-masing customer. Dengan banyaknya data tersebut, dapat dibuat top 20 customer untuk melihat 20 pelanggan yang melakukan pembelian terbanyak yang ditunjukkan pada gambar 3.13 dalam bentuk list.

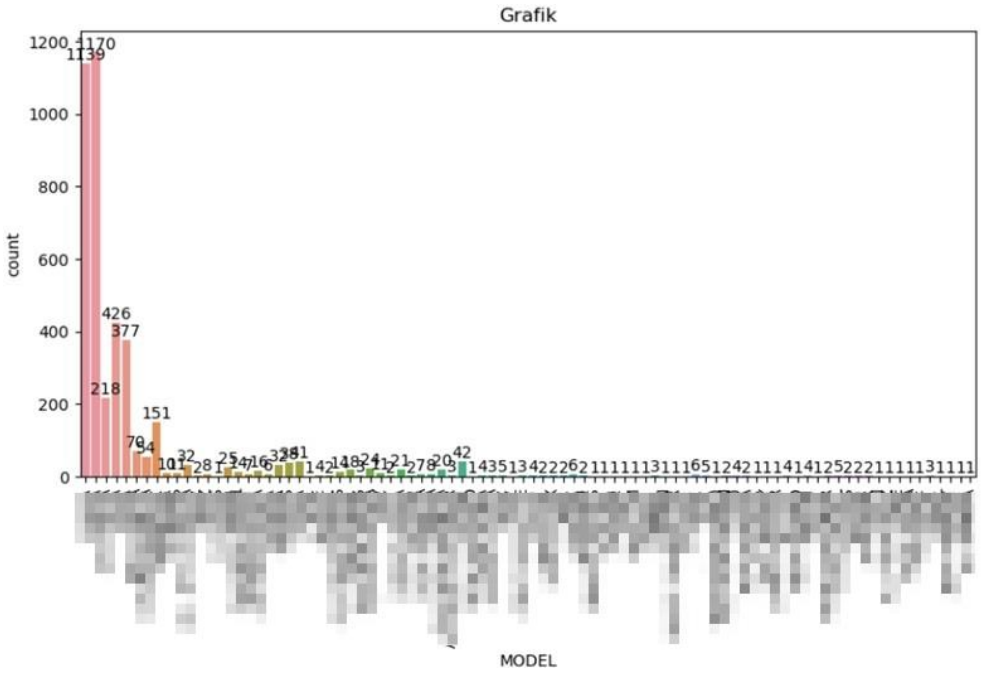
```
top20_customers.plot(kind="bar")
plt.title("Top 20 Customers by Count")
plt.xlabel("Customer Name")
plt.ylabel("Count")
plt.show()
```



Gambar 3. 14 Visualisasi Data Top 20 Customer

Dari list data top 20 customer yang sudah dibuat, dalam memudahkan pemahaman data untuk dipresentasikan dan dilihat oleh *audience* maka hal tersebut dapat divisualisasikan agar dapat melihat dalam bentuk grafik *barchart.* Dalam visualisasi *barchart*, hasil data dapat dilihat lebih jelas dari perbedaan jumlah pembelian yang customer lakukan. Terdapat perbedaan yang signifikan dari satu customer dibandingkan customer lainnya.

```
plt.figure(figsize=(10,5))
ax = sns.countplot(x='MODEL', data=df)
ax.bar_label(ax.containers[0])
plt.title('Grafik')
plt.xticks(rotation=90)
plt.show()
```



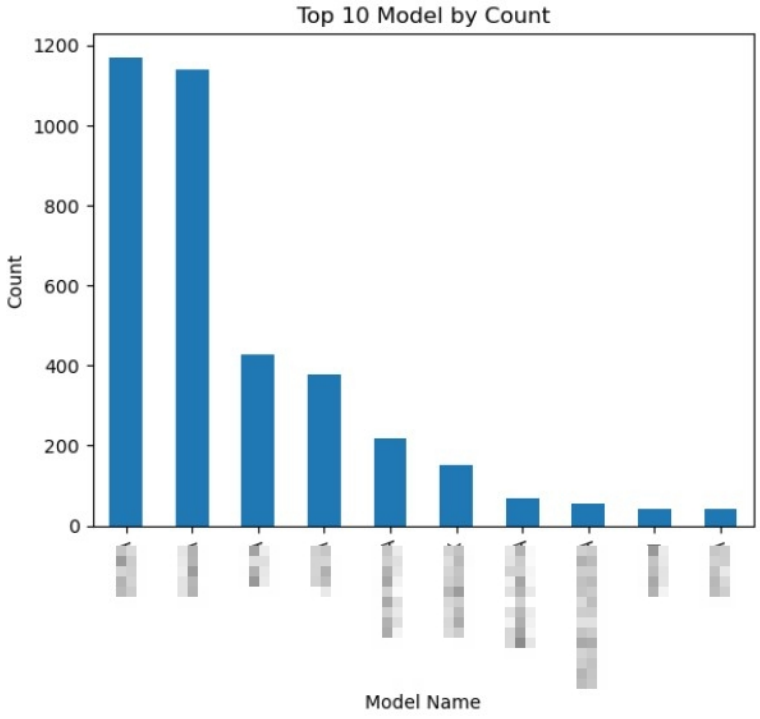
Gambar 3. 15 Visualisasi Jumlah Model Mobil Terjual

Gambar 3.15 merupakan visualisasi dari seluruh jumlah model mobil yang terjual berupa grafik *barchart*. Dalam *dataset* yang dimiliki, memiliki banyak model yang terjual, namun dengan hanya menampilkan seperti ini akan membuat visualisasi yang menumpuk sehingga sulit untuk dimengerti *audience*, maka dari hasil visualisasi keseluruhan akan dibuat visualisasi berupa *ranking* dari 10 model yang dijual terbanyak dalam data Sales Setir Kanan pada tahun 2022.

Gambar 3. 16 List Top 10 Model

Dari bentuk visualisasi yang ditampilkan pada gambar 3.15, *barchart* terlalu penuh. Maka dari itu untuk mempermudah pemahaman dalam menentukan 10 jumlah model yang paling banyak terjual dibuat dalam grafik top 10 model. *Output* data berbentuk list untuk menentukan 10 model terbanyak terlebih dahulu, hal yang dilakukan dalam pengkodean untuk mencari 10 model terbanyak yaitu dengan di-*define* terlebih dahulu variabel yang digunakan dan menampilkan list dengan mengambil data dari kolom MODEL dan menggunakan *value.count* lalu tampilkan *list* tersebut.

```
top10_model.plot(kind="bar")
plt.title("Top 10 Model by Count")
plt.xlabel("Model Name")
plt.ylabel("Count")
plt.show()
```



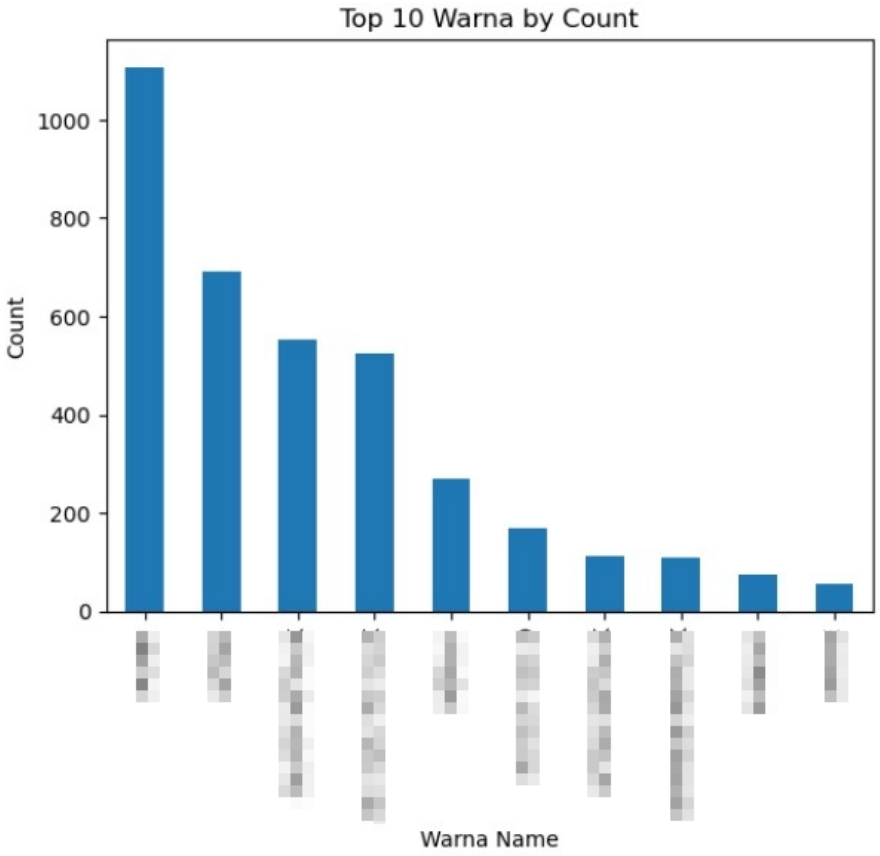
Gambar 3. 17 Visualisasi Top 10 Model

Setelah list dibuat, maka visualisasi data dapat dilakukan dengan menggunakan grafik *barchart*. Berdasarkan visualisasi dapat dilihat terdapat dua model yang memiliki jumlah penjualan tertinggi dibandingkan delapan lainnya. Dari hasil ini juga dapat diketahui bagi perusahaan untuk mencari unit tertentu yang memiliki jumlah penjualan terbanyak.

Gambar 3. 18 List Top 10 Warna

Sama seperti visualisasi sebelumnya pada gambar 3.16, namun visualisasi pada gambar 3.18 merupakan list dari top 10 warna mobil yang terjual dalam *dataset* Sales Setir Kanan pada tahun 2022. Warna unit mobil yang dimiliki dalam penjualan Setir Kanan memiliki banyak warna dan untuk mengetahui jumlah unit mobil yang paling banyak terjual berdasarkan warna untuk memudahkan *audience* menggunakan visualisasi top 10.

```
top10_warna.plot(kind="bar")
plt.title("Top 10 Warna by Count")
plt.xlabel("Warna Name")
plt.ylabel("Count")
plt.show()
```



Gambar 3. 19 Visualisasi Top 10 Warna

Gambar 3.19 adalah bentuk hasil visualisasi berupa *bar chart* dari list data top 10 warna mobil yang terjual pada tahun 2022. Dari 10 warna yang ditampilkan, terdapat 1 warna yang memiliki jumlah penjualan terbanyak dengan jumlah lebih dari seribu unit berhasil terjual. Perbedaan jumlah penjualan dari warna terbanyak signifikan dibandingkan warna mobil lain yang dijual pada Setir Kanan.

3.2.4 Minggu 7-10: Pengerjaan proyek 2: model *forecasting sales*

Sales adalah jumlah transaksi penjualan yang dilakukan atas produk yang dilakukan oleh perusahaan. Karena perusahaan Setir Kanan bergerak di bidang jual-beli mobil bekas, maka produknya yaitu adalah unit mobil dari berbagai tipe, model dan juga warna. Proyek kedua ini bertujuan untuk memprediksi penjualan mobil bekas di masa mendatang. Pengerjaan proyek ini yaitu dengan rentang waktu dari 10 April 2023 – 06 Mei 2023.

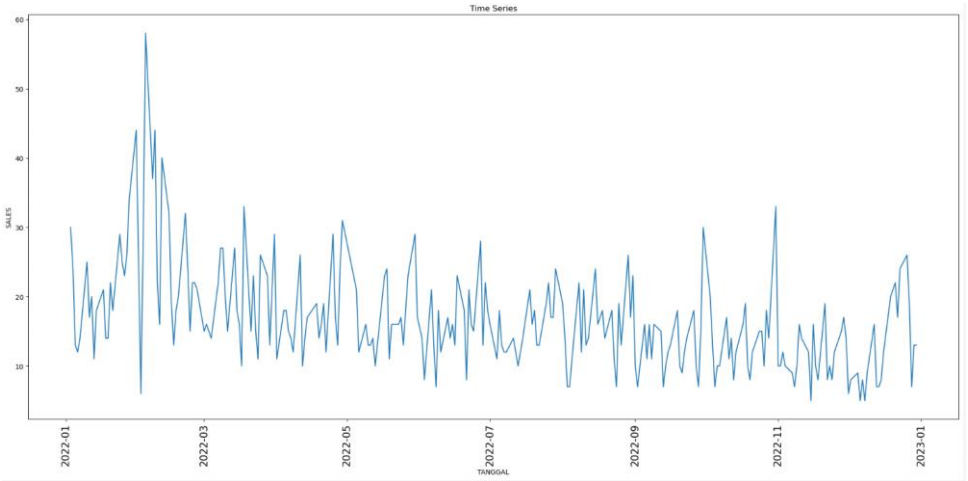
Gambar 3. 20 Penggabungan Data

Untuk memulai pengolahan data pada proyek *forecasting*, yang harus dilakukan yaitu menyiapkan data dengan *import* library yang dibutuhkan seperti *library* Pandas, Numpy, Matplotlib, dan Seaborn lalu *read* data. Selanjutnya pengolahan data dapat dilanjutkan dengan Groupby yang berfungsi untuk menggabungkan data berdasarkan variabel tertentu dengan tujuan untuk menggabungkan jumlah sales berdasarkan tanggal. Hasil yang didapat yaitu output yang menunjukkan jumlah penjualan pada tiap tanggal yang ada di *dataset*.

```
plt.figure(figsize=(20,10))
sns.lineplot(x=df2['TANGGAL'], y=df2['SALES'])

# Labelling
plt.xlabel("TANGGAL")
plt.ylabel("SALES")
plt.title("Time Series")

# Setting Ticks
plt.tick_params(axis='x',labelsize=15,rotation=90)
plt.tight_layout()

# Display
plt.show()
```

Gambar 3. 21 Pengkodean Labelling

Pada gambar 3.21 menggunakan pengkodean model *time series* dan menggunakan sns yaitu panggilan untuk *library* seaborn yang digunakan untuk menampilkan grafik guna membuat visualisasi menggunakan kolom tanggal dan sales. Lalu hasil dari pengkodean akan ditampilkan dalam bentuk *time series.*



Gambar 3. 22 Visualisasi Time Series

Gambar 3.22 merupakan hasil visualisasi dari pengkodean yang menggunakan model *time series*. Hasil tersebut menunjukkan adanya penurunan berdasarkan jumlah penjualan dengan frekuensi menggunakan kolom tanggal.

Forecast

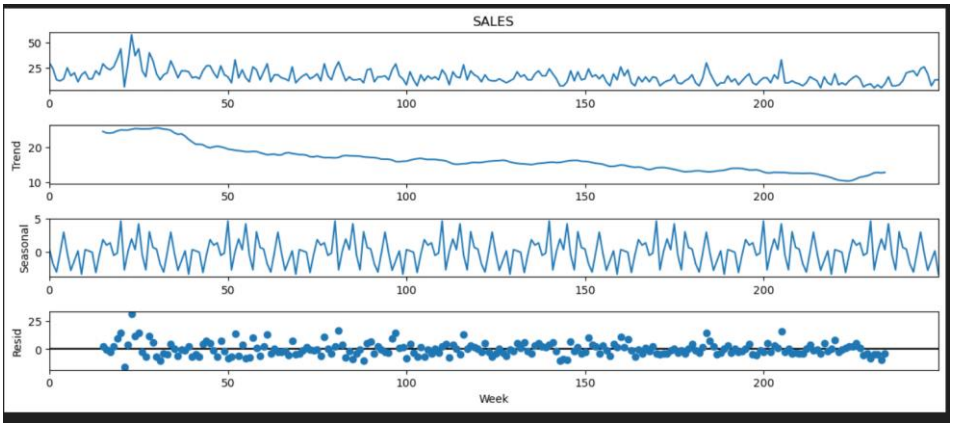
ARIMA, EXPONENTIAL SMOOTHING, FbPROPHET

ARIMA

```
from statsmodels.tsa.seasonal import seasonal_decompose

sales_ts = pd.Series(df2['SALES'].values, index=df2['TANGGAL'])
decomposition = seasonal_decompose(df2['SALES'], period=30, model='additive')
plt.rcParams['figure.figsize'] = 12, 5
decomposition.plot()
plt.xlabel("Week")
plt.show();
```

Gambar 3. 23 Seasonal Decompose

Selanjutnya pada gambar 3.23 merupakan pengkodean untuk menganalisa data menggunakan seasonal decompose hingga menghasilkan residu, seasonal, dan trend dari data untuk melihat ketertarikan *customer* pada penjualan di periode waktu tertentu.



Gambar 3. 24 Visualisasi Seasonal Decompose

Hasil visualisasi dari menggunakan seasonal decompose pada gambar 3.24 menunjukkan hasil bahwa trend menurun dan terdapat pola *seasonality*. Trend menunjukkan penjualan tertinggi berada pada awal-awal tahun yang menunjukkan banyak *customer* melakukan pembelian unit mobil di awal tahun 2022.

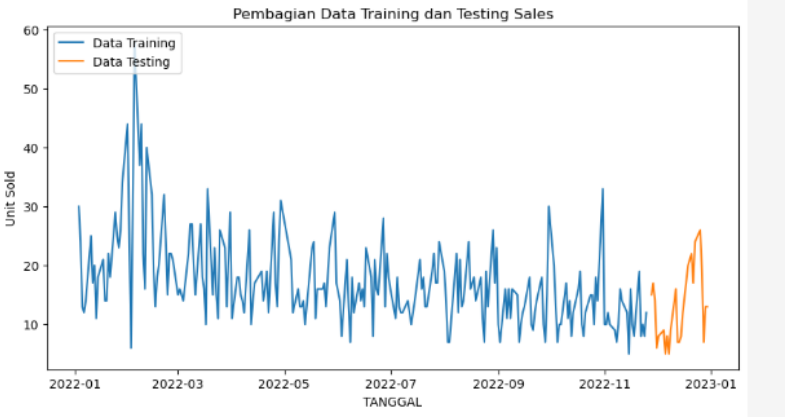
```
sales_train = df2[:int(0.9*(len(df2)))]
sales_test = df2[int(0.9*(len(df2))):]
```

Gambar 3. 25 Data Split

Hal yang dilakukan selanjutnya yaitu adalah *split* data *train* dan data *test* dengan rasio 90% untuk *testing* dan 10% untuk *train*. Data *training* adalah sekumpulan data yang memiliki atribut label yang digunakan untuk mengenal karakteristik data hingga menghasilkan pola. Sedangkan data *testing* merupakan sekumpulan data yang juga memiliki atribut label yang digunakan untuk menguji keakuratan pola dalam klasifikasi data *testing*. Data split digunakan untuk mengevaluasi atau menguji data dan membandingkan hasil klasifikasi sebagai tolak ukur dalam melihat akurasi suatu model.

```
plt.figure(figsize=(10,5))
sns.lineplot(x=sales_train['TANGGAL'], y=sales_train['SALES
sns.lineplot(x=sales_test['TANGGAL'], y=sales_test['SALES']
plt.legend(['Data Training','Data Testing'],
           loc='upper left')
plt.ylabel('Unit Sold')
plt.title('Pembagian Data Training dan Testing Sales')

plt.show()
```

Python



Gambar 3. 26 Visualisasi Data Split

Gambar 3.26 menunjukan hasil dari visualisasi pembagian data *training* dan data *testing* pada kolom *sales*. Dari hasil visualisasi berdasarkan gambar, data *training* merupakan yang berwarna biru dan data *testing* berwarna oranye.

```python
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm

model = sm.tsa.ARIMA(sales_train['SALES'],order=(1,1,0))
results = model.fit()
results.summary()
```

Python

SARIMAX Results

Dep. Variable:	SALES	No. Observations:	225
Model:	ARIMA(1, 1, 0)	Log Likelihood	-778.800
Date:	Sat, 27 May 2023	AIC	1561.600
Time:	15:54:13	BIC	1568.423
Sample:	0	HQIC	1564.354
	- 225		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.3954	0.048	-8.318	0.000	-0.489	-0.302
sigma2	61.2477	3.741	16.371	0.000	53.915	68.580

Ljung-Box (L1) (Q):	4.74	Jarque-Bera (JB):	100.48
Prob(Q):	0.03	Prob(JB):	0.00
Heteroskedasticity (H):	0.39	Skew:	0.35
Prob(H) (two-sided):	0.00	Kurtosis:	6.21

Gambar 3. 27 Pengkodean Model Arima

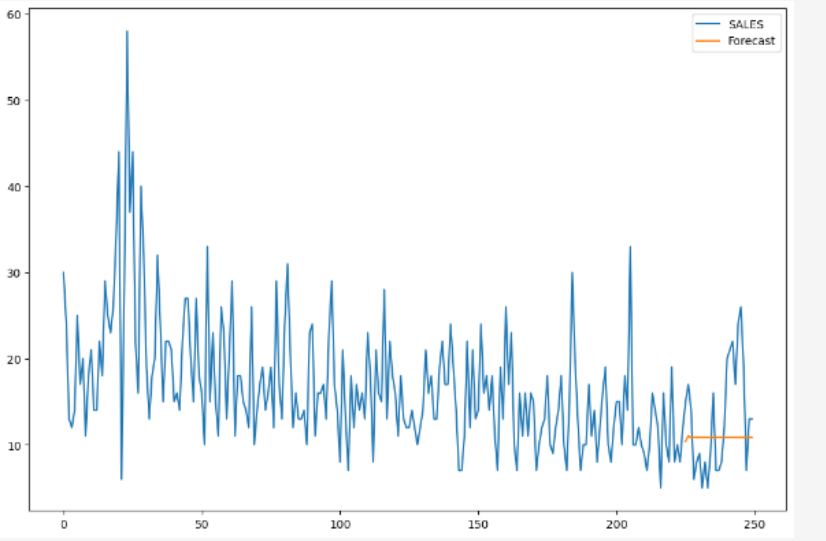
Berdasarkan pengkodean di gambar 3.27, proses pengkodean menggunakan model arima dari *library* statsmodel. Model arima disimpan ke dalam variabel yang Bernama model. Penjelasan model tertampil pada gambar. Dependant data yaitu sales berjumlah 225 data.

Gambar 3.28 merupakan hasil model *forecasting* dengan model arima, dari visualisasi tersebut garis yang berwarna biru merupakan data asli dan garis berwarna oranye merupakan data hasil prediksi. Dengan menggunakan model arima pada *forecasting*, hasil yang didapat kurang bagus sehingga tidak cocok untuk digunakan karena hasilnya tidak sesuai dengan data *testing*.

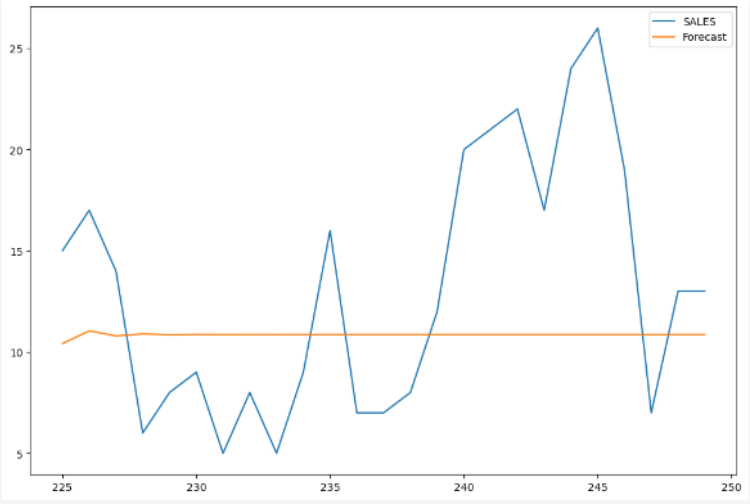
```
df2['Forecast']=results.predict(start=225,end=249,dynamic=T
df2[['SALES','Forecast']].plot(figsize=(12,8))
```

Python

```
<AxesSubplot:>
```



Gambar 3. 28 Visualisasi Hasil Model Forecasting Arima



Gambar 3. 29 Perbandingan Data Testing dan Hasil Prediksi

Gambar 3.29 merupakan setelah diperbesar untuk melihat lebih jelas data *forecasting* model arima, hasil visualisasi menunjukkan bagaimana perbandingan data *testing* dengan hasil prediksi yang mengakibatkan hasil ini

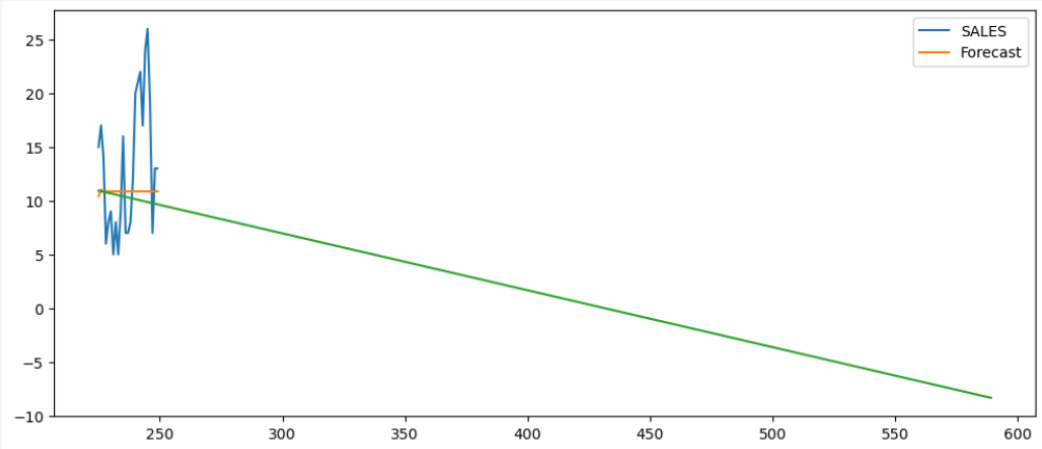
kurang baik. Dapat dikatakan kurang baik dikarenakan data prediksi yang tidak mendekati data SALES.

Exponential Smoothing

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

model = ExponentialSmoothing(sales_train['SALES'], trend='add', seasonal_periods=365).fit()
```

```
sales_test.plot()
model.forecast(365).plot()
```

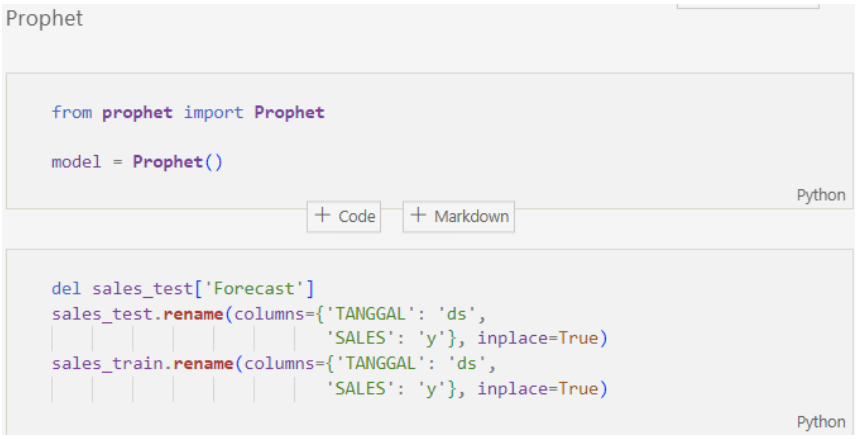
```
<AxesSubplot:>
```



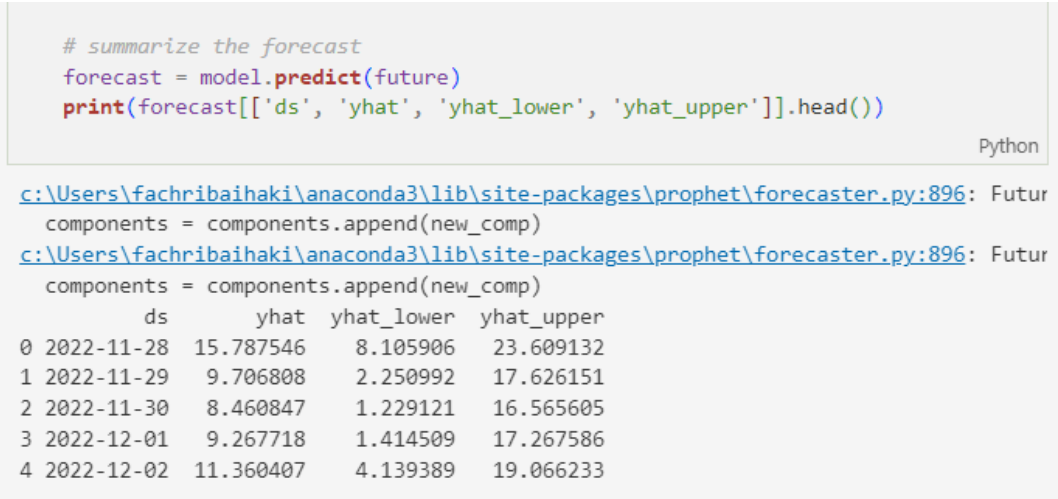
Gambar 3. 30 Visualisasi Hasil Model Forecasting Exponential Smoothing

Selanjutnya menggunakan *forecasting* model exponential smoothing dengan menggunakan parameter selama 365 hari ke depan, hasil prediksi dari model ini juga kurang bagus sehingga tidak cocok untuk digunakan karena tidak sesuai atau tidak mendekati dengan data *testing*.

Pada model *forecasting* yang selanjutnya yaitu menggunakan model prophet, pertama-tama *import library* prophet lalu mengganti variabel tanggal menjadi ds, dan sales menjadi y. Kemudian *building* model dengan data *train*.



Gambar 3. 31 Pengkodean Forecasting Model Prophet



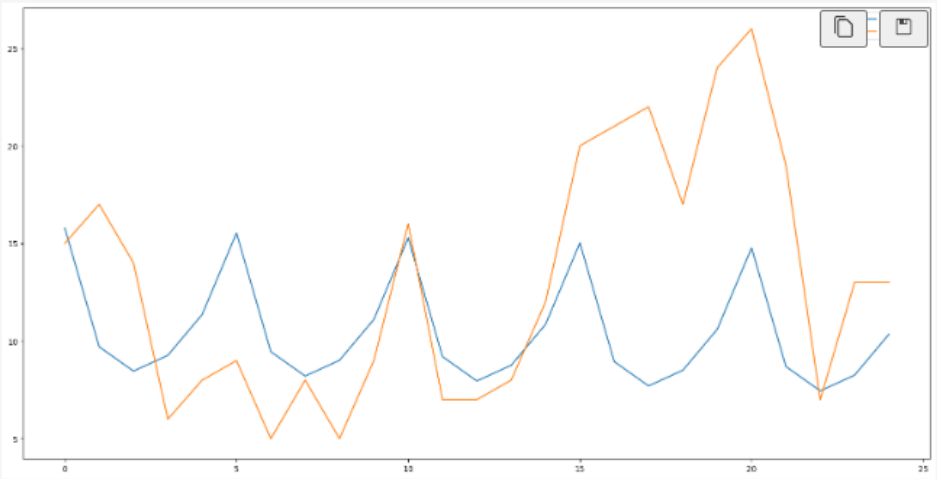
Gambar 3. 32 Hasil Forecasting Model Prophet

Gambar 3.32 menunjukkan hasil *forecasting* model prophet dalam bentuk angka. Variabel ds adalah tanggal yang diprediksi, yhat adalah hasil prediksinya. Prophet memberi batas atas & batas bawah prediksi, sehingga jika terjadi error dalam prediksi, hasilnya masih berada dalam rentang batas bawah dan batas atas.

```
slst = sales_test
slst.reset_index(drop=True, inplace=True)
comparation = pd.DataFrame({'Prediction':forecast['yhat'], 'Actual':slst['y']})
comparation.plot(figsize=(20,10))
```

Python

```
<AxesSubplot:>
```



Gambar 3. 33 Visualisasi Komparasi Data Prediksi dan Testing Model Prophet

Gambar 3.33 merupakan visualisasi setelah memasukan data prediksi ke dalam *dataframe testing* untuk komparasi. Hasil visualisasi ditampilkan setelah komparasi dari prediksi data ke dalam *dataframe*. Hasil yang didapat adalah baik karena mendekati dengan hasil *testing*, hal ini yang membuat model prophet adalah model yang paling baik diantara 3 model *forecasting* yang diuji.

3.2.4 Minggu 11-14: Pengerjaan proyek 3: *Clustering* jenis mobil berdasarkan frekuensi penjualan dan keuntungan

```
#Hilangkan transaksi yang harganya minus

df = df[df['HARGA MODAL'] > 0]
df.reset_index(inplace=True, drop=True)
```

✓ 0.0s Python

Gambar 3. 34 Filter Data Harga Modal

Pengkodean pada gambar untuk menghilangkan transaksi yang harganya minus, transaksi yang minus dianggap tidak valid untuk analisis karena harga tidak mungkin untuk minus. Namun hal ini tetap ada pengecualian, jika

terjadinya retur barang atau gagal bayar sehingga unit ditarik kembali tapi tetap saja harga jual tidak mungkin minus.

```
df['HARGA JUAL (EX. TAX 10%)'] = np.nan
df['HARGA JUAL (EX. TAX 10%)'] = round(df['HARGA JUAL (INC. TAX 10%)']/1.01, -6)


'''harga sesudah tax = harga sebelum tax + (harga sebelum tax x 10%)
sesudah = sebelum + 0.01sebelum
sesudah = 1.01 sebelum tax
sesudah / 1.01 = sebelum tax
'''
✓ 0.0s
```

Gambar 3. 35 Membuat Variabel Harga Jual Tanpa Tax

Pada gambar 3.35 pengkodean dilakukan untuk menghitung nilai harga jual tanpa *tax* atau pajak, pajak dihilangkan karena tidak masuk ke dalam keuntungan perusahaan.

```
df['KEUNTUNGAN'] = df['HARGA JUAL (EX. TAX 10%)'] - df['HARGA MODAL']
✓ 0.0s

df.head(2)
✓ 0.0s
```

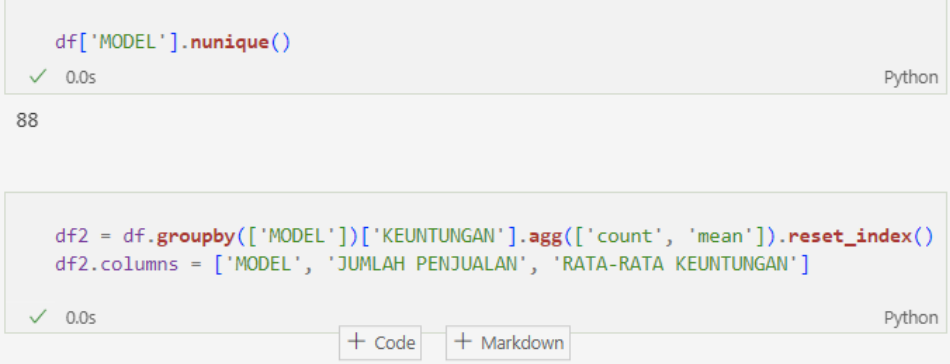
Gambar 3. 36 Membuat Variabel Keuntungan

Pengkodean digunakan untuk menghitung nilai keuntungan untuk setiap transaksi, setelah dijalankan maka data teratas ditampilkan untuk dicek *dataframe* baru dari hasil kalkulasi.

```
df[df['KEUNTUNGAN'] < 0].head(2)
✓ 0.0s
```

Gambar 3. 37 Mengecek Data Keuntungan dari Dataframe

Gambar 3.37 yaitu menunjukan kode untuk mengecek apakah ada keuntungan yang minus atau rugi. Diketaui bahwa terdapat beberapa transaksi mobil dengan keuntungan yang minus.



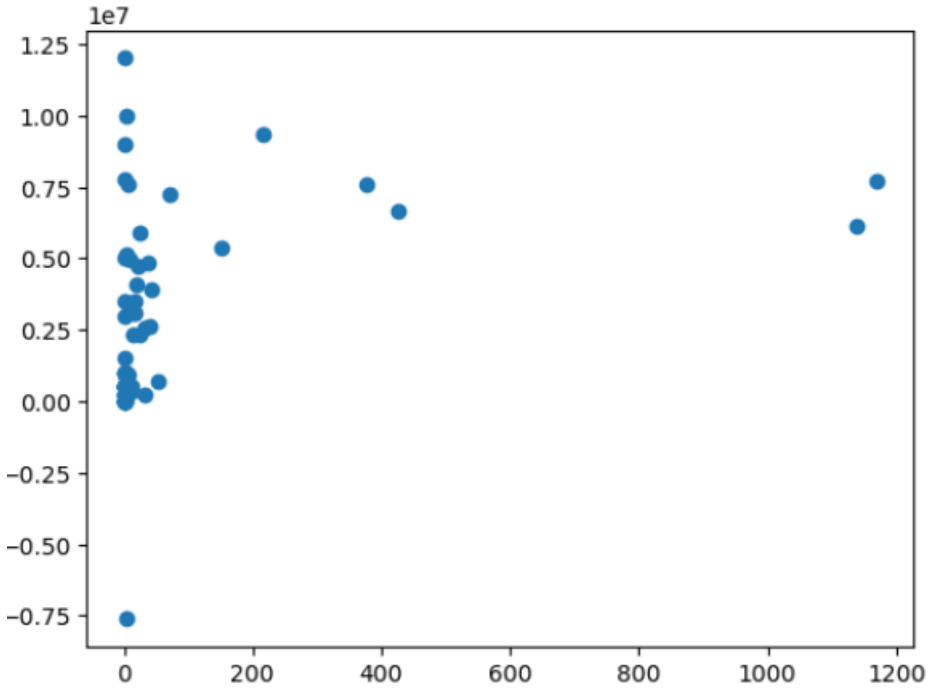
Gambar 3. 38 Melihat Jumlah Model Terjual dan Rata Rata Keuntungan

Kode yang ditunjukkan pada gambar 3.38 yaitu untuk melihat berapa banyak model yang dijual perusahaan lalu membentuk *dataframe* baru dengan mencari rata-rata keuntungan dan menghitung jumlah penjualan untuk setiap model mobil. Setiap model mobil memiliki rata rata keuntungan dan jumlah penjualan, kedua variabel tersebut yang akan digunakan dalam modeling.

Gambar 3. 39 Visualisasi Dataframe Terbaru

Dataframe dicek lagi untuk melihat *dataframe* terbaru yang menunjukan jumlah penjualan dan rata-rata keuntungan pada model mobil yang berhasil dijual oleh perusahaan.

```
import matplotlib.pyplot as plt

plt.scatter(df2['JUMLAH PENJUALAN'], df2['RATA-RATA KEUNTUNGAN'])
plt.show()
```



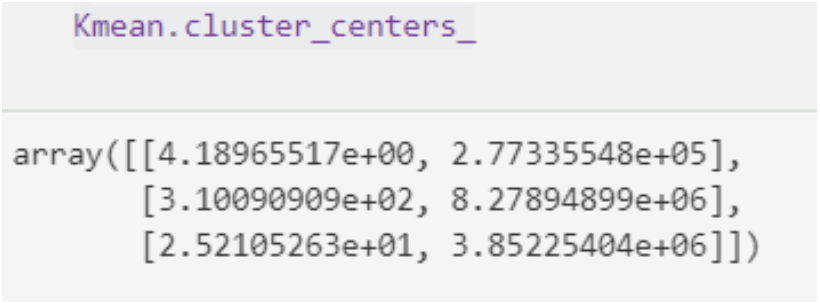
Gambar 3. 40 Visualisasi Hasil Sebaran Data

Gambar 3.40 merupakan hasil dari membentuk scatterplot untuk melihat sebaran data jumlah penjualan dan rata rata keuntungan untuk setiap model yang ada.

Gambar 3. 41 Pembentukan Model Cluster

Gambar 3.41 menunjukan kode dalam pembentukan model cluster yang membutuhkan setidaknya 2 input numerik, pada proyek ini menggunakan variabel jumlah penjualan dan rata-rata keuntungan dengan tujuan melihat model mobil yang memiliki jumlah penjualan dan keuntungan rendah, sedang, dan tinggi.

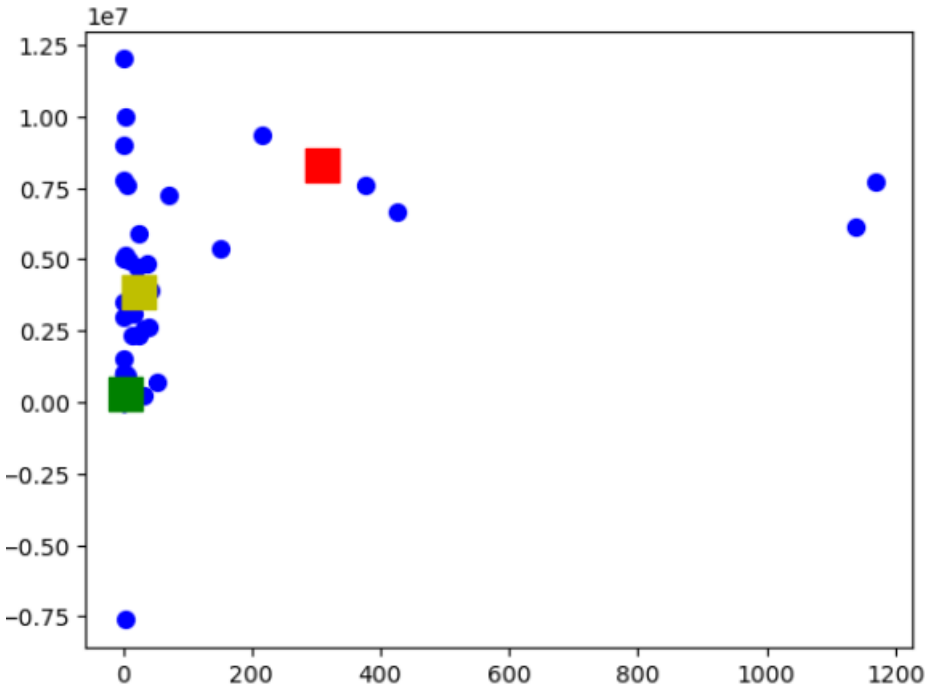
KMeans digunakan dalam *building model*, algoritma ini digunakan karena merupakan algoritma yang paling banyak digunakan dalam pengerjaan *clustering*.



Gambar 3. 42 Melihat Sentroid Cluster

Kode Kmean.cluster_centers_ yang ditampilkan pada gambar 3.42 digunakan untuk melihat sentroid atau yang diketahui sebagai titik tengah dari setiap *cluster* yang terbentuk.

```
plt.scatter(df2['JUMLAH PENJUALAN'], df2['RATA-RATA KEUNTUNGAN'], s =50, c='b')
plt.scatter(4.18965517e+00, 2.77335548e+05, s=200, c='g', marker='s')
plt.scatter(3.10090909e+02, 8.27894899e+06, s=200, c='r', marker='s')
plt.scatter(2.52105263e+01, 3.85225404e+06, s=200, c='y', marker='s')
plt.show()
```



Gambar 3. 43 Visualisasi Hasil Scatter Plot dan Sentroid

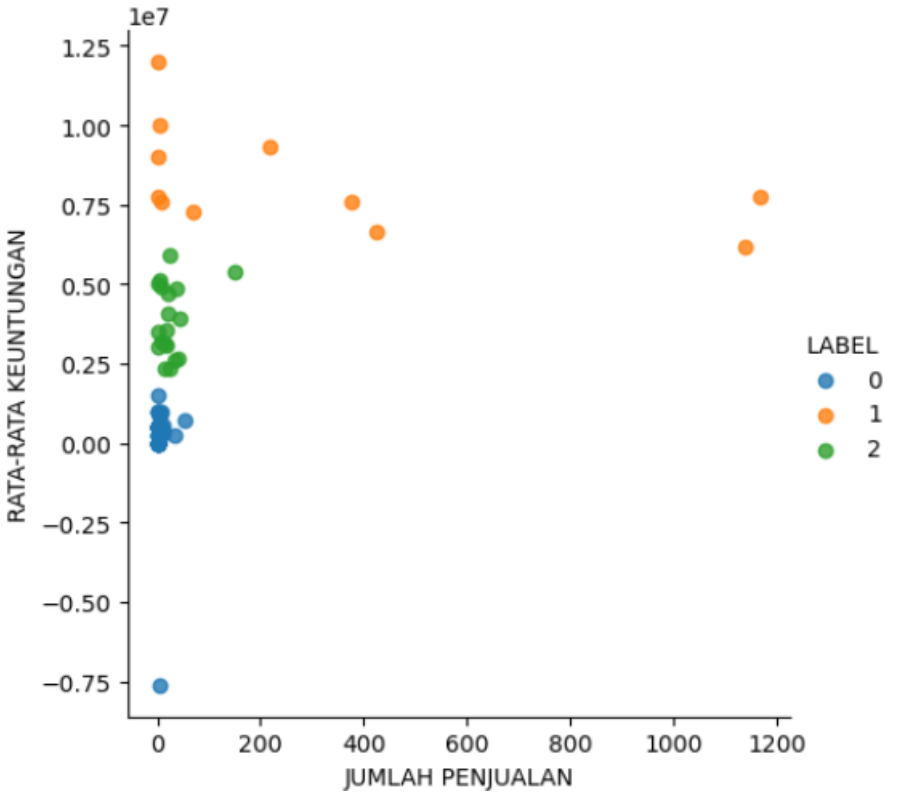
Gambar 3.43 merupakan hasil dari pembentukan scatter plot dengan variabel penjualan dan keuntungan, pada hasil visualisasi juga terdapat penandaan sentroid untuk mengetahui lokasi bagian titik sentral cluster.

Gambar 3. 44 Melihat Hasil Prediksi Label

Gambar 3.44 untuk melihat hasil prediksi label/cluster. Label terbagi menjadi 3 yaitu ada 0,1, dan 2. *Default* label dari model *cluster* adalah angka dan dimulai dari 0, hal ini dikarenakan mesin hanya mengerti angka sehingga agar dapat mengubahnya menjadi karakter harus dilakukan secara manual.

Label hasil dari prediksi *cluster* dimasukkan ke dalam *dataframe* awal untuk mengetahuinya dan juga model mobil dikategorikan sebagai antara tinggi/rendah/ atau sedang lalu dipastikan bahwa cluster yang terbentuk yaitu adalah tiga cluster.

```
import seaborn as sns
sns.lmplot('JUMLAH PENJUALAN', 'RATA-RATA KEUNTUNGAN', data=df2, hue='LABEL', fit_reg=False)

plt.show()

#1 rendah
#2 sedang
#0 tinggi
```



Gambar 3. 45 Membentuk Scatter Plot Baru

Gambar 3.45 menunjukan yang digunakan untuk membentuk *scatter plot* baru berdasarkan variabel penjualan dan keuntungan, terdapat pewarnaan berdasarkan cluster. Berdasarkan plot ini juga dapat diketahui apa arti dari 0, 1, dan 2 yang merupakan label tinggi/rendah atau sedang. Setelah pengkodean dilakukan, gambar visualisasi *scatter plot* merupakan hasil dari scatter plot berdasarkan jumlah penjualan dan keuntungan dan terdapat perbedaan warna dan keterangan label yang ditunjukkan.

```
df2['LABEL'].replace({0:'TINGGI',
                      1:'RENDAH',
                      2:'SEDANG'}, inplace=True)
```

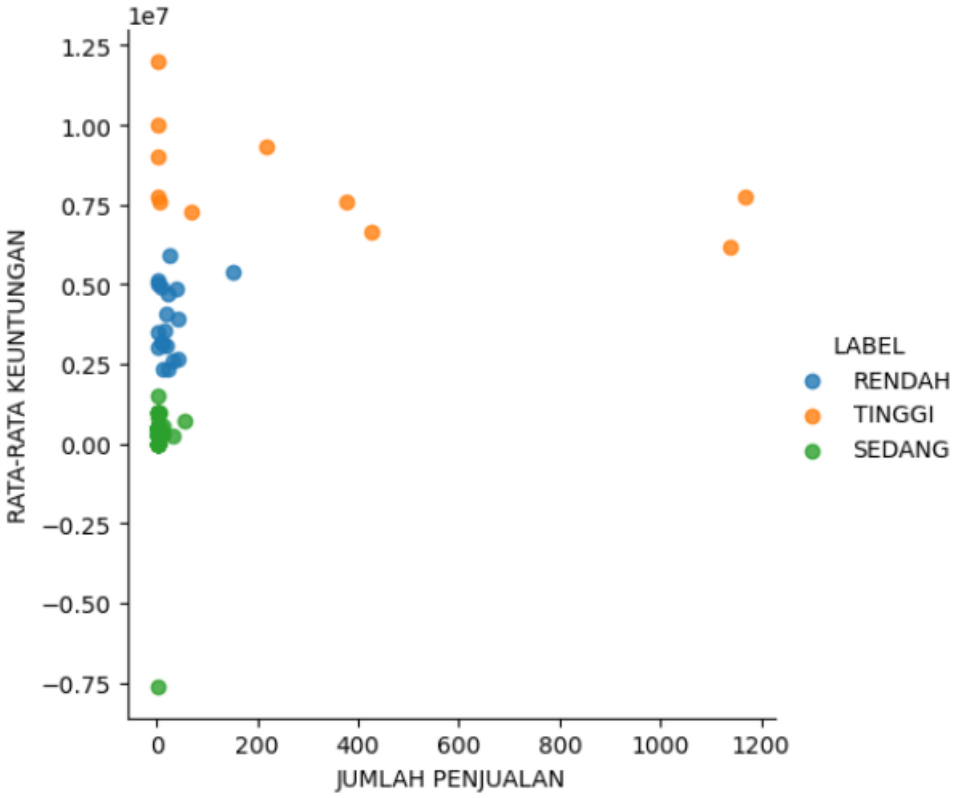
	MODEL	JUMLAH PENJUALAN	RATA-RATA KEUNTUNGAN	LABEL
0		1	3.000000e+06	SEDANG
1		377	7.578249e+06	RENDAH
2		20	4.075000e+06	SEDANG
3		32	2.500000e+05	TINGGI
4		1	1.200000e+07	RENDAH
...		...	...	...
83		1	0.000000e+00	TINGGI
84		2	5.000000e+05	TINGGI
85		41	2.646951e+06	SEDANG
86		2	7.750000e+06	RENDAH
87		11	5.454545e+05	TINGGI

88 rows × 4 columns

Gambar 3. 46 Mengubah Label Menjadi Karakter

Gambar 3.46 merupakan pengkodean untuk mengubah label secara manual agar lebih mudah dipahami dengan menggunakan karakter, tidak seperti sebelumnya yang hanya berbentuk angka saja. Berdasarkan pengkodean dibuat angka 0 (nol) berarti tinggi, 1 (satu) berarti rendah, dan 2 (dua) berarti sedang.

```
import seaborn as sns
sns.lmplot('JUMLAH PENJUALAN', 'RATA-RATA KEUNTUNGAN', data=df2, hue='LABEL', fit_reg=False)

plt.show()
```



Gambar 3. 47 Visualisasi Scatter Plot Setelah Label Diperbaharui

Gambar 3.47 adalah hasil visualisasi setelah membentuk ulang plot dengan label yang telah diperbaharui menggunakan karakter sehingga dapat diketahui warna yang ditampilkan dengan keterangan dari arti warna tersebut.

3.2.5 Minggu 15-18: Pengerjaan proyek 4: Prediksi harga jual mobil berdasarkan spesifikasi mobil

```
#filter data

df = df[df['HARGA MODAL'] > 0]
df = df[df['WARNA'] != 0]
df = df[df['TIPE UNIT'] != 0]
```

0.0s

Gambar 3. 48 Filter Data Menhindari Nilai Minus

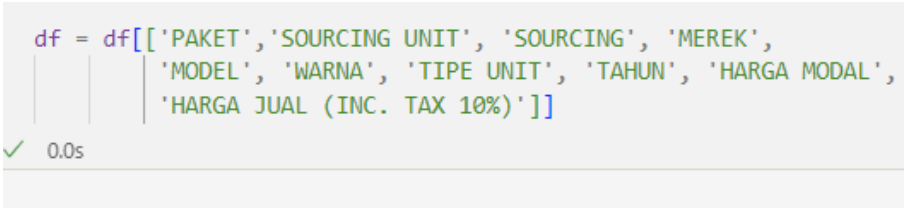
Gambar 3.48 bertujuan untuk mem-filter data agar harga tidak diperbolehkan minus, menggunakan data warna dan tipe unit selain dari nilai nol. Hal ini dikarenakan dalam data terdapat nilai nol pada kedua kolom variabel tersebut.

```
df.drop_duplicates(['MEREK', 'MODEL', 'WARNA',
                    'TIPE UNIT', 'TAHUN'], inplace=True)
df.reset_index(drop=True, inplace=True)
```

✓ 0.0s

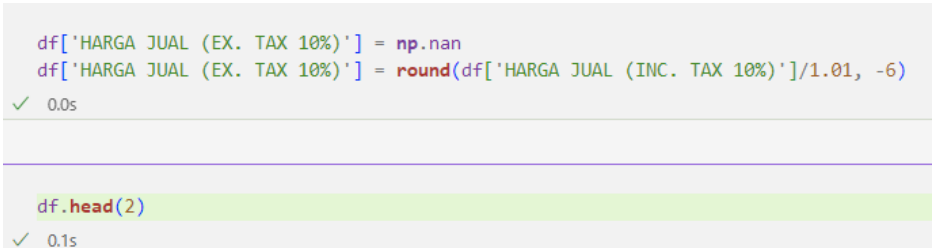
Gambar 3. 49 Menghapus Variabel Duplikat

Pada gambar 3.49 dilakukan pengkodean untuk men-*drop* atau menghapus data yang duplikat. Data yang terdpulikat yaitu diantaranya adalah data MEREK, MODEL, WARNA, TIPE UNIT, dan TAHUN. Hal ini dilakukan agar tidak adanya *redundancy* data sehingga tidak terjadi *error* atau kesalahan membaca yang dilakukan *tools* pengolahan data.



Gambar 3. 50 Menginisialisasi Data yang Diperlukan

Pada gambar 3.50 yaitu untuk mengambil data yang diperlukan saja dan men-*define* dalam variabel df untuk penggunaan pengolahan data selanjutnya. Hal ini agar tidak memenuhi *output* agar pemahaman dalam hasil olahan data dapat dilakukan dengan lebih mudah karena data yang ditampilkan hanya data yang ingin digunakan saja.



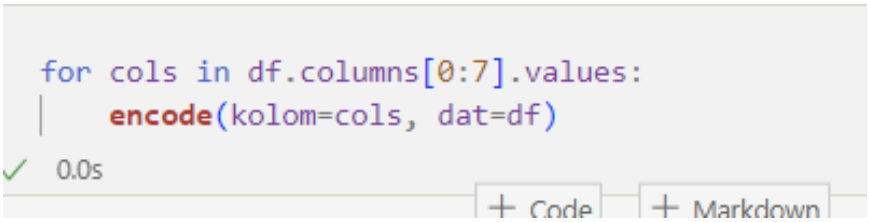
Gambar 3. 51 Mencari Harga Jual Sebelum Pajak

Gambar 3.51 menunjukkan kode untuk mencari harga jual sebelum atau tidak termasuk pajak 10% sebagai angka murni yang dimiliki oleh perusahaan. Kemudian tampilkan data akhir yang sudah menampilkan variabel data harga jual sebelum *tax* pada *dataframe.*

Gambar 3. 52 Proses Label Encoding

Kode pada gambar 3.52 digunakan untuk melakukan label *encoding* atau mengubah karakter menjadi angka dikarenakan *machine learning* tidak dapat

memproses karakter dan hanya dapat memproses angka. Objek LabelEncoder diinisialisasi dan fungsi dipanggil lalu dimasukkan ke variabel label_encoder. Lalu fungsi label encoding dilakukan pada kolom ‘Kolom’



Gambar 3. 53 Proses Looping

Kode pada gambar 3.53 digunakan untuk membuat looping atau proses berulang setiap kolom dan diubah menjadi fungsi encode() agar dapat dibaca oleh *maching learning*.

	PAKET	SOURCING UNIT	SOURCING	MEREK	MODEL	WARNA	TIPE UNIT	TAHUN	HARGA MODAL	HARGA JUAL (INC. TAX 10%)	HARGA JUAL (EX. TAX 10%)	PA
0	REGULER	DRIVE THRU	Telehunting									
1	REGULER	DRIVE THRU	Referral Official Partner ACC									

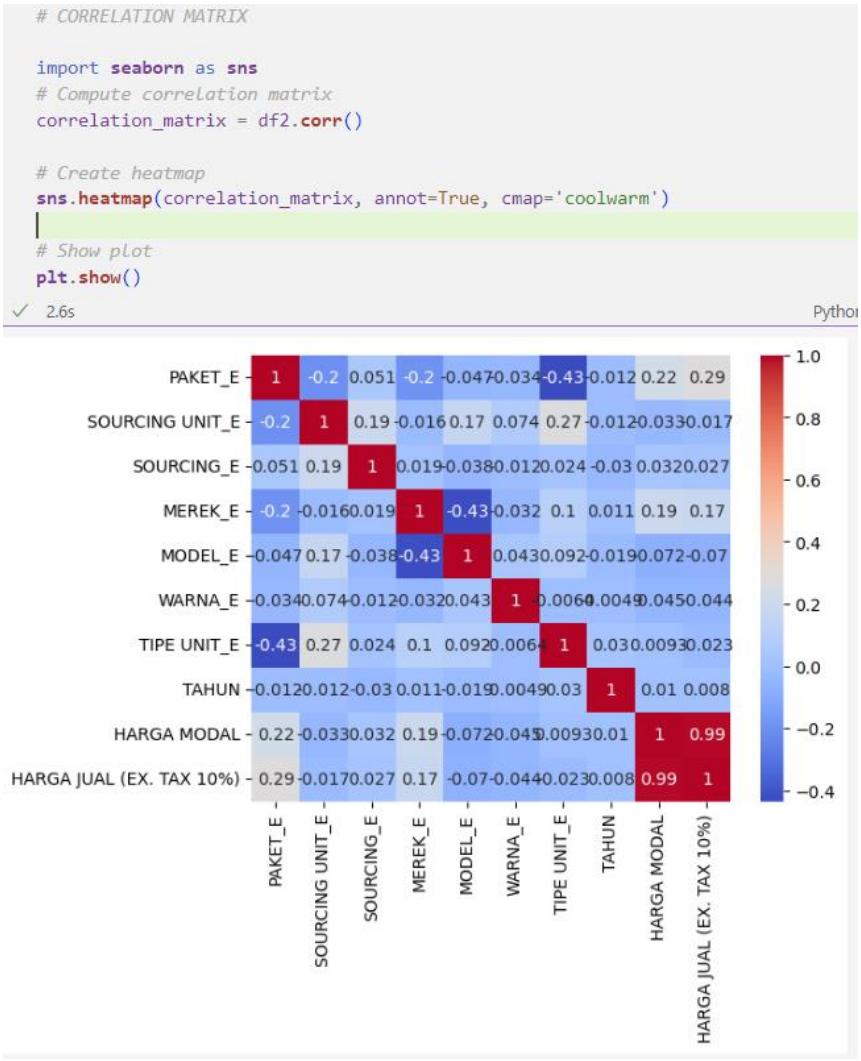
```
df2 = df[['PAKET_E','SOURCING UNIT_E', 'SOURCING_E', 'MEREK_E', 'MODEL_E', 'WARNA_E', 'TIPE UNIT_E', 'TAHUN', 'HARGA MODAL',
df2.head(2)
```

0.0s

	PAKET_E	SOURCING UNIT_E	SOURCING_E	MEREK_E	MODEL_E	WARNA_E	TIPE UNIT_E	TAHUN	HARGA MODAL	HARGA JUAL (EX. T
0	3	0	8	13	21	5	33			
1	3	0	3	13	21	73	33			

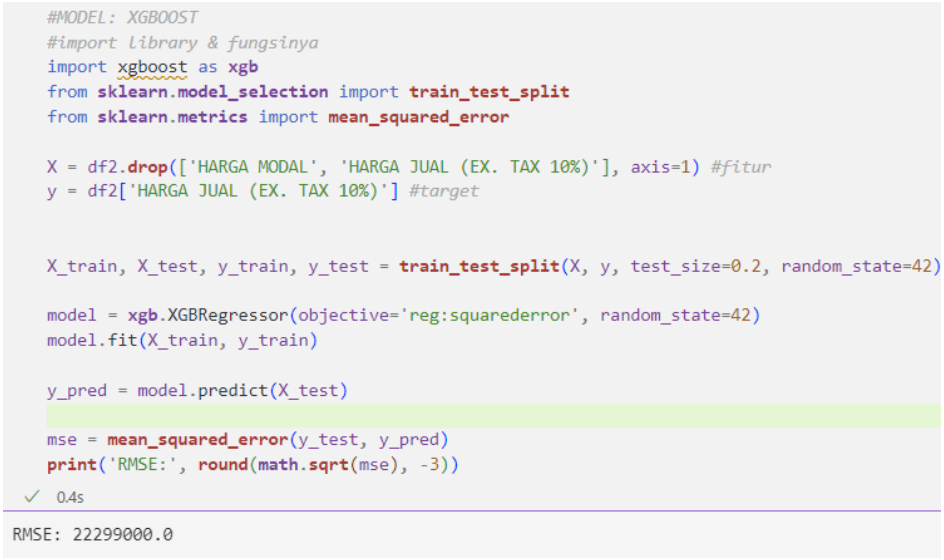
Gambar 3. 54 Penyeleksian Kolom

Ini merupakan seleksi untuk kolom yang akan digunakan, yaitu kolom terbaru yang sudah diganti menjadi angka. Variabel ditambahkan agar dibedakan dengan data asli.



Gambar 3. 55 Correlation Matrix

Gambar 3.55 merupakan correlation matrix dengan menggunakan sns.heatmap. Ini merupakan plot untuk melihat hubungan antar kolom, semakin mendekati 0 maka artinya semakin tidak berhubungan, sedangkan semakin mendekati 1 maka hubungan antar variabel makin kuat. Nilai minus pada plot menunjukkan hubungan yang terbalik, dan nilai plus melambangkan hubungan searah. Kesimpulannya yaitu hubungan antara kolom-kolom dengan variabel yang ingin diprediksi yaitu variabel harga adalah lemah cenderung tidak ada hubungan sehingga tidak ada pengaruh yang signifikan pada harga jual.

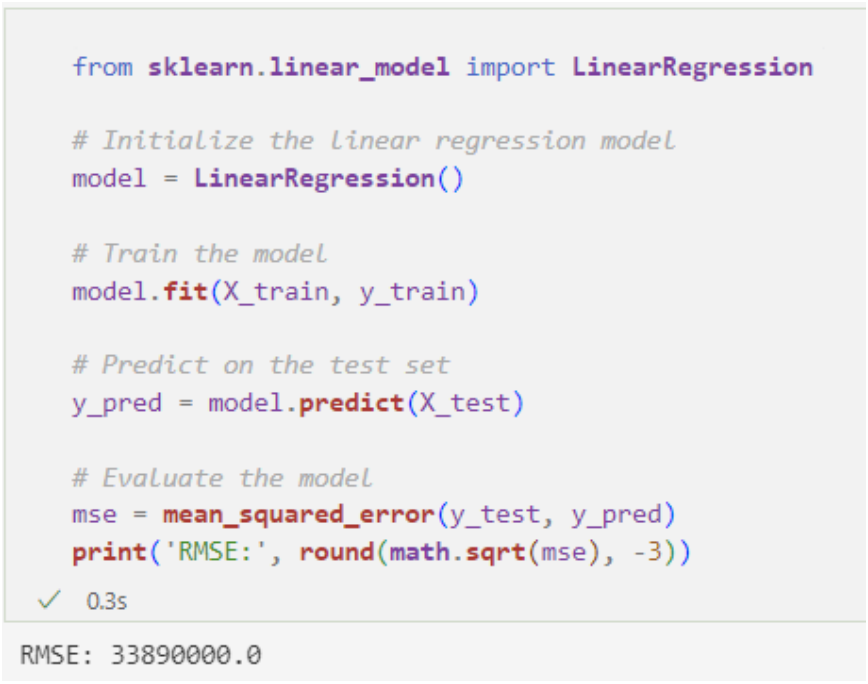


Gambar 3. 56 Build Model Menggunakan XGBoost

Gambar 3.56 merupakan pemodelan menggunakan algoritma XGBoost. XGBoost merupakan algoritma yang ditingkatkan berdasar dari *gradient boosting decision tree* dan dapat membangun *boosted trees* dengan efisien dan dijalankan dengan parallel. XGBoost adalah salah satu teknik *machine learning* untuk mengatasi permasalahan *regression* dan *classification* dari *Gradient Boosting Decision Tree* (GBDT). XGBoost dipilih karena dengan membuat model yang baru berdasarkan model yang lama dapat menghasilkan kesalahan prediksi (*error)* yang lebih rendah dari model sebelumnya.[18]

Dalam melakukan pemodelan XGBoost, pertama yang harus dilakukan yaitu *import library* dan fungsinya. Lalu *load dataset* menjadi suatu *dataframe* berupa X dan Y. X merupakan fitur dan Y adalah target variabel. Kemudian bagi data untuk mengetahui data target dan data fitur. Data target adalah harga yaitu variabel yang ingin diprediksi, dan data fitur yaitu variabel yang membantu untuk mendapatkan hasil target.

Split data menjadi *train* dan *test*. Data *training* berguna untuk build model dan data *testing* berguna untuk mengetes bagus atau tidaknya suatu model. Pembagiannya yaitu data *test* 20% dan 80% data *train*. Hasil yang didapat yaitu RMSE atau nilai error prediksi sebesar 22299000.0 yaitu artinya rata-rata prediksi dapat mengalami error 22 juta.

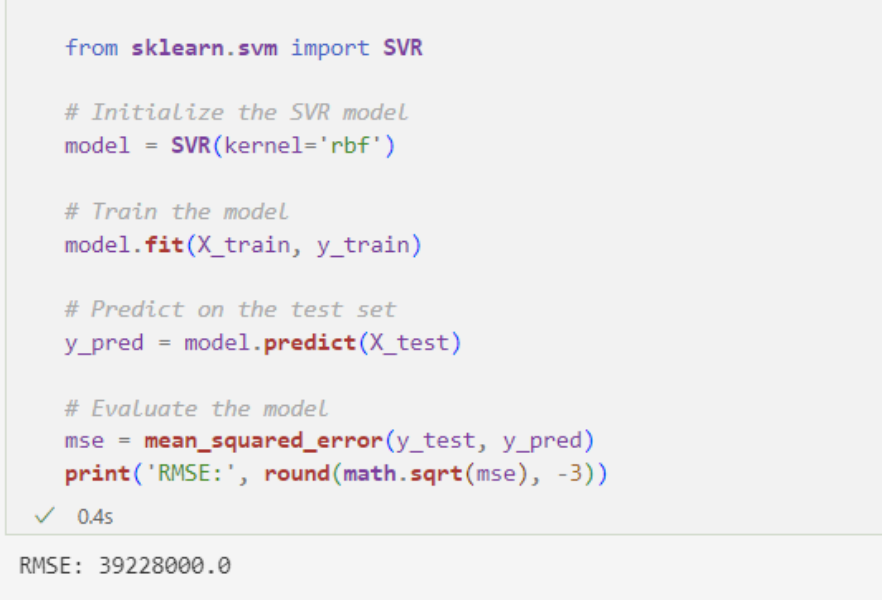


Gambar 3. 57 Build Model Menggunakan Linear Regression

Gambar 3.57 merupakan pemodelan dengan algoritma *Linear Regression*. *Linear Regression* merupakan metode untuk pemodelan hubungan antara satu variabel *dependent* dan satu variabel *independent*. Variabel *independent* adalah variabel yang mempengaruhi dan variabel dependen adalah variabel yang dipengaruhi. Di dalam model regresi, variabel independent menerangkan variabel dependennya. Hubungan antara variabel dalam analisis regresi bersifat linier yaitu perubahan variabel A akan diikuti oleh perubahan yang terjadi pada variabel B secara konstan.[19] Setelah menggunakan data yang telah dibagi, hasil yang didapat dari pemodelan menggunakan *Linear Regression* yaitu RMSE yang dimiliki sebesar 33890000.0 atau berarti prediksi dapat mengalami *error* hingga 33 juta. Hal ini berarti prediksi dapat mengalami *error* lebih banyak dibandingkan dengan menggunakan model XGBoost.

Gambar 3. 58 Build Model Menggunakan Logistic Regression

Gambar 3.58 merupakan pemodelan dengan algoritma *Logistic Regression. Logistic Regression* adalah metode yang paling umum untuk digunakan dalam pembuatan model prediksi *probability* dalam suatu kejadian. Model ini digunakan jika variabel *output* dari model yang digunakan terdefinisi sebagai kategori biner.[20] Dalam pemodelan menggunakan algoritma *Logistirc Regression,* hasil yang didapat yaitu RMSE sebesar 45298000.0 atau berarti prediksi dapat mengalami error hingga 45 juta. Hasil RMSE ini merupakan yang terbesar dibandingkan dengan algoritma lainnya.



Gambar 3. 59 Build Model Menggunakan SVR

Gambar 3.59 merupakan pemodelan dengan algoritma SVR. SVR atau *Support Vector Regression* adalah bentuk pendekatan dari SVM. Metode yang digunakan untuk mengatasi *overfitting.* Karakteristik dari SVR yaitu mengecilkan batas kesalahan yang digeneralisasi. [21] Dari menggunakan pemodelan dengan algoritma *Support Vector Regression,* hasil yang didapat yaitu RMSE sebesar 39228000.0 atau berarti prediksi dapat mengalami error hingga 39 juta.

Berdasarkan pemodelan dari 4 algoritma yang diuji, XGBoost merupakan pemodelan yang memiliki RMSE terendah. RMSE atau *Root Mean Square Error* adalah salah satu cara untuk evaluasi model regresi dengan mengukur tingkat akurasi hasil suatu model. Nilai *error* pada RMSE menunjukkan performa dari model yang dibuat, dari nilai RMSE yang dihasilkan semakin kecil atau mendekati nol maka hasil prediksi semakin akurat.

3.3 Kendala yang Ditemukan

Proses magang pada Astra Credit Companies dapat dilaksanakan dengan lancar secara umum, namun selama proses magang berlangsung terdapat beberapa kendala yang mengakibatkan penundaan pekerjaan serta menghambat pengerjaan proyek, yaitu:

a. Perusahaan memiliki praktisi *data analyst* namun hanya menggunakan excel saja, sehingga tidak ada mentor khusus dalam pelaksanaan analisis data perusahaan.
b. Proyek yang dilaksanakan menggunakan data yang bersifat kredensial sehingga tidak dapat menggunakan *device* laptop sendiri dan harus bergantian dengan praktisi *data analyst* yang bekerja di sini.
c. Kurangnya kelengkapan data sehingga harus menunggu dengan waktu yang cukup lama untuk melanjutkan pengerjaan proyek.

3.4 Solusi atas Kendala yang Ditemukan

a. Materi dan mencari penyelesaian masalah terkait *data analysis* dipelajari melalui internet, seperti menonton video tutorial untuk pembelajaran, membaca artikel dan jurnal yang membahas teknis dalam proses analisis data dan penerapannya.
b. Pengerjaan proyek dilakukan selama jam kerja berlangsung yaitu di kantor, membagi waktu dengan praktisi *data analyst* yang bersangkutan.
c. Mengerjakan proyek yang ada terlebih dahulu dan menganalisa proyek lebih jauh