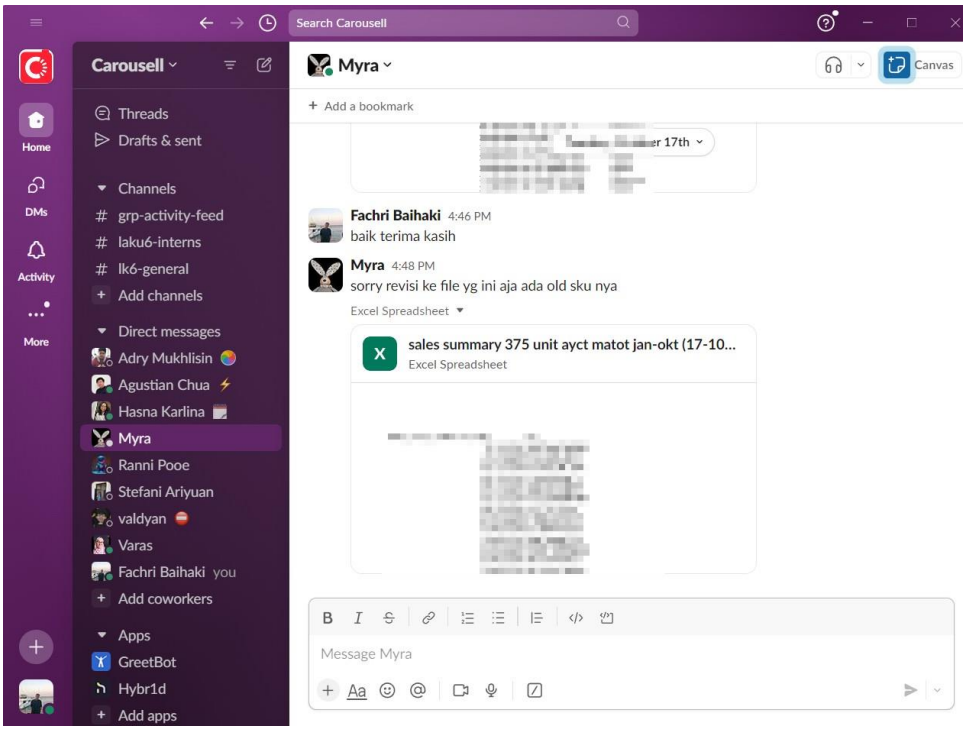
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Posisi sebagai *business data analyst intern* berada pada divisi Retail BuyBack (RBB) dalam departemen Business. Bapak Ferdy Winopo, yang menjabat sebagai Head of Retail Buy-Back (RBB), memegang peran sebagai supervisor dalam program magang. Beliau bertanggung jawab untuk memberikan tugas dan proyek kepada peserta magang yang harus dilaksanakan selama periode magang. Selain *supervisor*, terdapat mentor yang membimbing peserta magang dalam melakukan pekerjaan dan menjalankan proyek. Mentor tersebut adalah Ka Ranni Pooe yang memiliki *role* sebagai RBB Support. Koordinasi dalam pekerjaan dilakukan secara *offline* maupun *online* dengan rekan divisi RBB maupun divisi lain. Untuk koordinasi secara online, perusahaan menggunakan aplikasi Slack dan WhatsApp.



Gambar 3. 1 Aplikasi Slack untuk Koordinasi

3.2 Tugas dan Uraian Kerja Magang

Dalam tugasnya sebagai *business data analyst intern*, peserta magang bertanggung jawab untuk melakukan analisis dan prediksi data. Proses dimulai dengan mengimpor data dari web yang menyimpan database perusahaan, khususnya dalam bentuk laporan 'Sales Summary'. Setelah mengimpor data, langkah selanjutnya melibatkan proses pembersihan data, pemodelan, dan visualisasi data. Selama menjalankan tugasnya, *business data analyst* intern menggunakan dengan Python sebagai bahasa pemrograman utama.

Dalam pelaksanaan kegiatan magang, peserta magang terlibat dalam berbagai proyek yang telah direncanakan oleh pengguna (*user*) perusahaan, masing-masing dengan tujuan yang berbeda. Selain dari proyek-proyek tersebut, *business data analyst intern* juga melibatkan diri dalam tugas-tugas harian, mingguan, dan berkala sesuai dengan permintaan tim lain. Selama proses magang, peserta juga memperoleh pembelajaran melalui pengalaman, yang membantu dalam pengembangan *soft skills* serta pemahaman mendalam terkait alur kerja perusahaan, termasuk aspek seperti perolehan koleksi untuk stok dan proses pembelian oleh pelanggan. Program magang ini dibagi menjadi tiga fase utama, yakni fase pengenalan, fase pelaksanaan/pengerjaan proyek, dan fase presentasi

Dalam peran internship sebagai *business data analyst*, tugas utama adalah menjalankan serangkaian proses analisis dan prediksi data. Langkah awal dalam melakukan proses tersebut yaitu impor data dari web, database perusahaan yang tersimpan dalam laporan menu 'Sales Summary'. Setelah itu, dilakukan pembersihan data, pemodelan, dan visualisasi menggunakan framework *Cross Industry Standard Process for Data Mining* (CRISP-DM). Visualisasi dilakukan menggunakan Google Data Studio pada awalnya, namun diputuskan menggunakan Python digunakan sebagai bahasa pemrograman utama dengan menggunakan Visual Studio Code: Jupyter Notebook. Selama masa *internship*, *business data analyst intern* terlibat dalam pengerjaan proyek yang disusun oleh *user*, yaitu Head of RBB dengan tujuan yang beragam. Selain tugas proyek, ada juga tugas harian, mingguan, dan berkala sesuai dengan permintaan tim. Selama magang, peserta

memperoleh pengalaman, mengembangkan keterampilan sosial, serta memahami alur kerja *company*, dari pengumpulan *collection stock* hingga pembelian oleh *customer*. Program magang dibagi menjadi tiga fase, yakni pengantar, pelaksanaan proyek, dan presentasi.

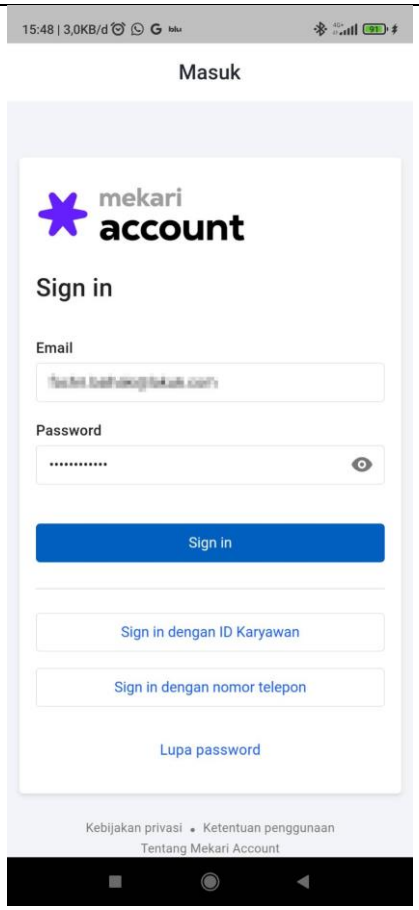
Tabel 3. 1 Uraian Kerja Magang

NO	Pekerjaan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
1	*On-boarding* dan pengenalan perusahaan Laku6.	1	14-08-2023	18-08-2023
2	Pengenalan sistem, aplikasi dan web yang digunakan perusahaan, serta eksplorasi data dan proyek.	2-4	21-08-2023	08-09-2023
3	Proyek 1: *Eksploratory Data Analysis* (EDA)	5-7	11-09-2023	29-09-2023
4	Proyek 2: *Clustering* produk Laku6 berdasarkan frekuensi penjualan dan rata-rata keuntungan.	8-11	02-10-2023	27-10-2023
5	Proyek 3: model f*orecasting sales*	12-14	30-10-2023	17-11-2023
6	Proyek 4: Prediksi total profit per product	15-18	20-11-2023	15-12-2023

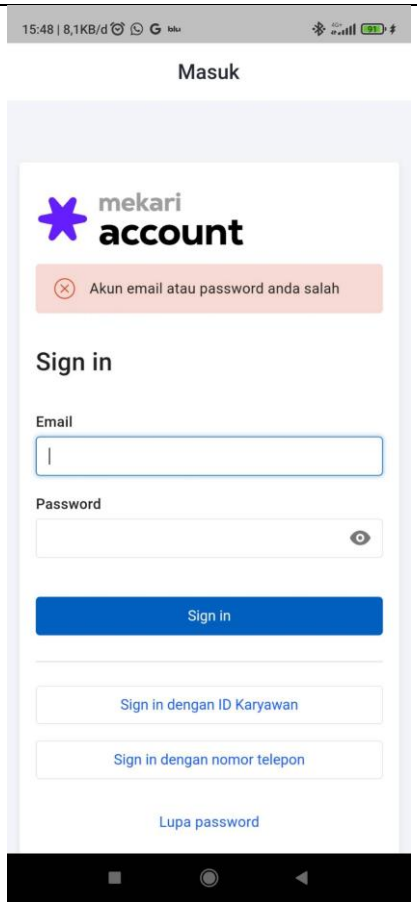
3.2.1 Minggu 1: *On-boarding* dan Pengenalan Perusahaan Laku6

Proses magang di Laku6 dimulai dengan partisipasi dalam sesi presentasi yang memberikan gambaran umum tentang perusahaan pada tanggal 14 Agustus 2023. Presentasi tersebut mencakup informasi terkait struktur organisasi, peraturan perusahaan, dan sistem kerja yang dijelaskan oleh tim Human Resource (HR) Laku6. Setelah presentasi, HR menyelenggarakan tur untuk memperkenalkan berbagai divisi dan fasilitas di gedung perusahaan yang terletak di Jl. Lapangan Bola No.5, RT.7/RW.1, Kb. Jeruk, Kec. Kb. Jeruk, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta. HR secara rinci memperkenalkan berbagai ruangan di kantor dan memperkenalkan aplikasi yang digunakan oleh karyawan, termasuk fitur-fitur seperti absensi, pengecekan slip gaji, dan pengajuan kepada departemen HR. Aplikasi yang digunakan dalam konteks ini adalah Talenta by Mekari.

Tabel 3. 2 Tampilan menu login Talenta



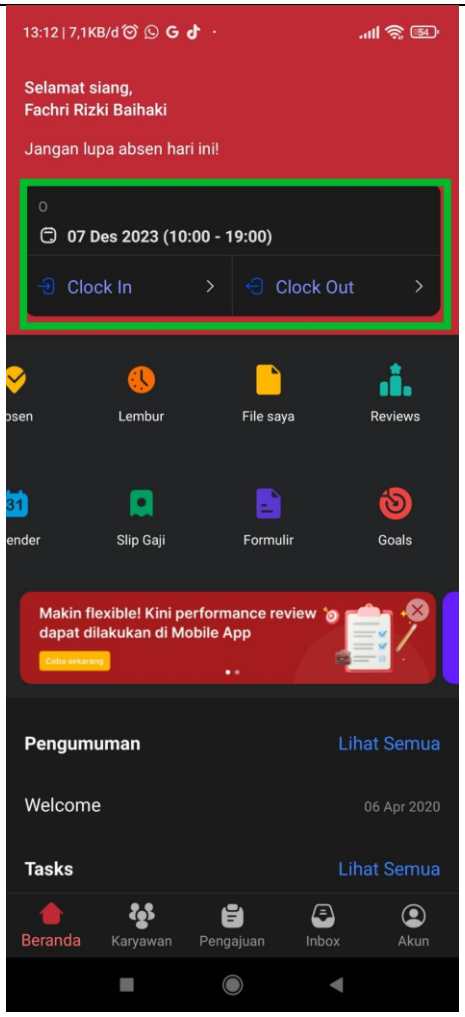
Gambar 3. 2 Display Login



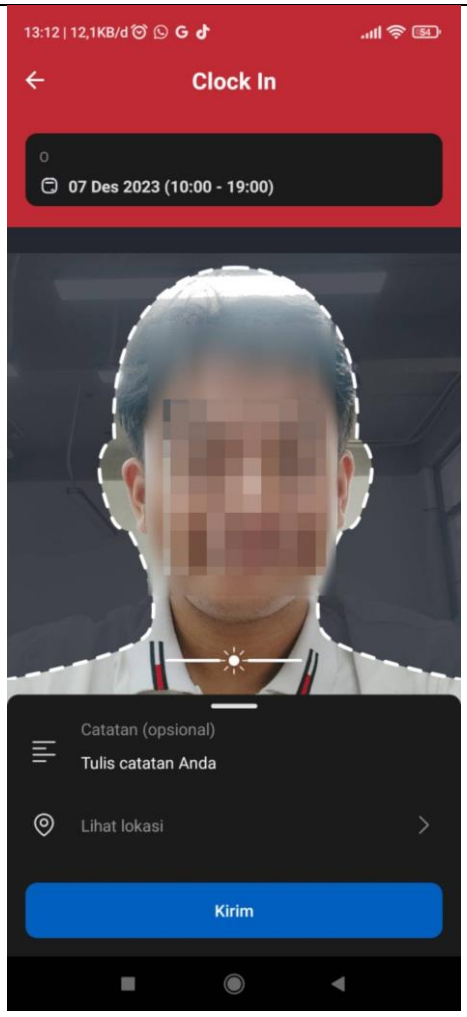
Gambar 3. 3 Display Error Login

Gambar 3.2 adalah *display* untuk melakukan *sign-in* ke dalam aplikasi Talenta untuk melakukan berbagai aktivitas yang berkaitan dengan HR. Ketika email yang dimasukkan tidak terdaftar ataupun salah maka akan ditampilkan output error sehingga harus memasukkan ulang akunnya seperti pada gambar 3.3.

Tabel 3. 3 Display Menu Talenta



Gambar 3. 4 Display Menu



Gambar 3. 5 Display Absen dengan Foto

Gambar 3.4 menunjukkan berbagai menu ketika sudah berhasil login. Karyawan juga melakukan absensi dengan klik menu ‘Clock-in’ seperti yang ditandai kotak berwarna hijau pada gambar. Untuk melakukan absensi maka akan dilakukan foto dan pengecekan lokasi seperti pada gambar 3.5. Setelah jam kerja telah selesai, maka dapat melakukan ‘Clock-out’.

3.2.2 Minggu 2-4: Pengenalan Sistem, Aplikasi dan Web yang Digunakan Perusahaan, erta Eksplorasi Data dan Proyek.

Dalam melakukan pekerjaan di Laku6, Excel masih menjadi salah satu *software* dipilih karena selain tampilannya yang sudah dikenal secara luas di kalangan masyarakat umum, juga telah menjadi perangkat yang salah satu sering digunakan karena kepraktisan penggunaannya. Excel, sebagai aplikasi *wokrsheet* dari Microsoft, menyusun data dalam bentuk sel-sel yang terstruktur dalam baris dan kolom. Data yang dimasukkan dalam *wokrsheet* dapat dilakukan penghitungan dan diproses dengan tingkat akurasi tinggi melalui penggunaan berbagai rumus yang mempermudah pengguna. Kelebihan lainnya adalah kemampuan Excel untuk memvisualisasikan hasil olahan data, termasuk pembuatan tabel, diagram, dan grafik garis. Walaupun memiliki sejumlah keunggulan, Excel juga memiliki keterbatasan dalam pemrosesan data, sehingga diperlukan perangkat lunak yang lebih canggih untuk pemrosesan data yang bersifat advance

Dalam pengolahan data sederhana dan pengambilan data, perusahaan menggunakan Google Data Studio atau LookerStudio namun dikarenakan kenyamanan maka Visual Studio Code juga digunakan dengan *environment* Jupyter Notebook dan *framework* CRISP-DM dengan Python sebagai bahasa pemrogramannya.

CRISP-DM, yang merupakan singkatan dari *Cross Industry Standard Process for Data Mining*, adalah sebuah model yang telah diakui secara standar dan bersifat terbuka. Model ini memiliki fase-fase yang terstruktur, dapat disesuaikan dengan fleksibilitas sehingga dapat memenuhi standar kebutuhan seorang *business data analyst* walaupun merupakan *framework* sederhana. Oleh karena itu, CRISP-DM sering dipilih sebagai *framework* utama dalam proses *data mining*. Dengan menyajikan serangkaian tahapan yang diakui oleh industri, model ini menjadi panduan yang dapat diandalkan dalam menjalankan aktivitas *data mining*.

Sebagai perangkat pembuatan kode yang dikembangkan oleh Microsoft, Visual Studio Code (VSC) unggul dengan fitur-fitur dan ekstensi yang

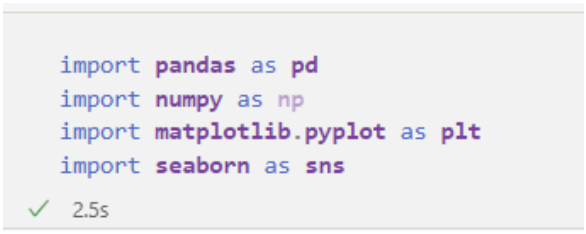
komprehensif, menjadikannya pilihan utama bagi para *developer*. VSC didesain untuk mendukung semua *operating system*, termasuk Linux, Mac OS, dan Windows, sehingga mempermudah penggunaan perangkat dalam pembuatan kode. Kelebihan lainnya adalah VSC dirancang dapat dijalankan dengan ringan dan memberikan kenyamanan kepada penggunanya, tanpa memerlukan spesifikasi perangkat yang *high-end*. Tools tersebut mempunyai *support* bawaan dalam *coding* menggunakan JavaScript dan menyediakan berbagai ekstensi, termasuk yang berkaitan dengan bahasa pemrograman Python.

Alasan penggunaan Python sebagai bahasa pemrograman adalah karena sifatnya yang *open source*. Python diaplikasikan mulai dari proses analisis data hingga evaluasi, dan memberikan dukungan bagi *business data analyst* agar dapat melakukan perbaikan. Selain itu, bahasa pemrograman ini mendukung Object-Oriented Programming (OOP), serta dapat untuk membuat skrip dan *Application Programming Interface* (API) yang diperlukan oleh *user*. Python diterapkan di *environment* Jupyter Notebook berbagai *library* dapat ditampung dan dipanggil untuk proses *coding*. Pemilihan *environment* tersebut didasarkan pada kenyamanan dan kebiasaan pribadi, membuatnya menjadi *environment* kerja yang familiar dan sesuai dengan preferensi pengguna.

3.2.3 Minggu 5-7: Proyek 1: *Eksploratory Data Analysis* (EDA)

Seorang *business data analyst* memiliki tanggung jawab utama dalam melakukan analisis data, pemodelan data, dan mendukung perusahaan dalam menerapkan *data analyst* sebagai solusi sesuai dengan permintaan yang diberikan untuk keperluan bisnis perusahaan. Dalam menjalankan tugasnya, seorang *business data analyst* terlibat dalam proyek-proyek yang memiliki tenggat waktu tertentu. Setiap proyek didapat dari permintaan pengguna (*user*) berdasarkan kebutuhan spesifik yang mereka miliki. *Supervisor*, yang berperan sebagai user langsung, menugaskan proyek setelah proses *onboarding* dilakukan.

Salah satu proyek yang ditugaskan kepada *business data analyst* adalah memvisualisasikan *rank* tertinggi dari beberapa variabel kunci. Proyek ini bertujuan untuk memberikan wawasan tentang visualisasi dari data perusahaan. *Supervisor* menugaskan proyek ini dengan harapan dapat memberikan bantuan dalam menetapkan target yang ingin dicapai dalam penjualan untuk periode selanjutnya. Target penjualan tersebut mencakup pemilihan produk yang akan dijual, mengumpulkan *collection*, dan menentukan *event* program. Pelaksanaan proyek ini dijadwalkan dalam rentang waktu antara 11 September 2023 hingga 29 September 2023.



Gambar 3. 6 Import Library EDA

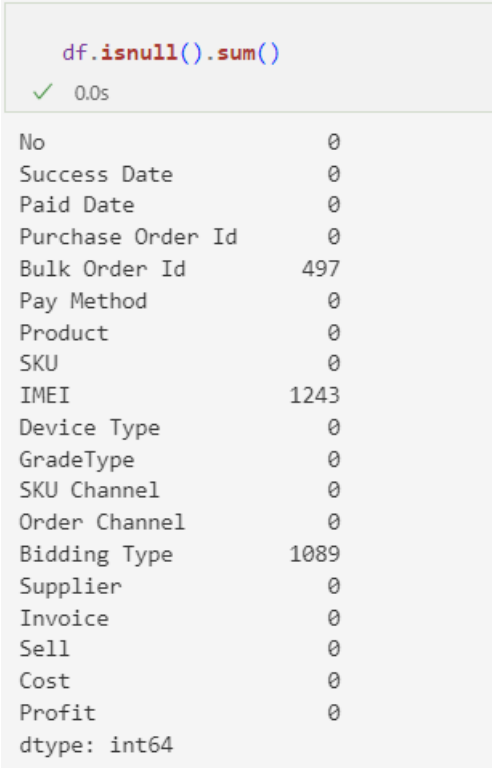
Kode pada gambar 3.6 merupakan impor modul-modul yang umumnya digunakan dalam analisis data dan visualisasi di lingkungan pemrograman Python dengan fokus pada pengolahan data menggunakan Pandas, manipulasi array dengan NumPy, serta pembuatan visualisasi dengan Matplotlib dan Seaborn.

```
df = pd.read_excel("data penjualan 2023.xlsx", sheet_name='sales')
✓ 12.6s

df.head(3)
✓ 0.0s
```

	No	Success Date	Paid Date	Purchase Order Id	Bulk Order Id	Pay Method	Product	SKU	IMEI	Device Type	...	SKU Channel
0	1	2023-01-02	2023-01-02	[illegible]	[illegible]	[illegible]	Samsung [illegible]	[illegible]	[illegible]	[illegible]		[illegible]
1	2	2023-01-02	2023-01-02	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		[illegible]
2	3	2023-01-02	2023-01-02	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		[illegible]

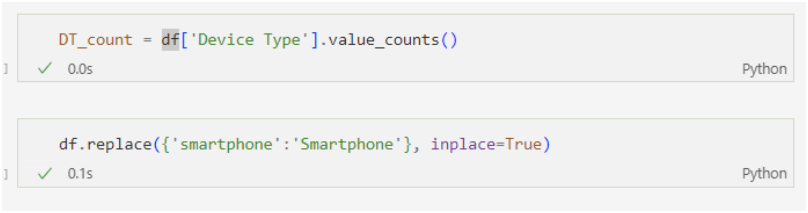
Gambar 3. 7 Read Data Excel

Kode pada gambar 3.7 menggunakan modul Pandas untuk membaca file Excel yang disimpan dalam format .xlsx. Dalam hal ini, file tersebut dinamai "data penjualan 2023.xlsx", dan data dari lembar Excel yang bernama 'sales' akan dibaca. Hasil dari pembacaan data disimpan dalam *dataframe* ‘df’. Kemudian data ditambilkan sebanyak 3 baris untuk memberikan gambaran singkat dari struktur dan isi data.



Gambar 3. 8 Mengecek Nilai Null

Baris kode pada gambar 3.8 digunakan untuk menghitung jumlah nilai null atau missing dalam setiap kolom dari *dataframe* ‘df’. Kode tersebut menghasilkan informasi berguna dalam pemahaman awal mengenai kekosongan data dalam dataset, dan dapat membantu dalam proses pembersihan atau imputasi data yang hilang sebelum melakukan analisis lebih lanjut.



Gambar 3. 9 Menghitung Nilai Unik dan Replace Data Redundancy

Baris kode pertama dari gambar 3.9 digunakan untuk menghitung jumlah kemunculan setiap nilai unik dalam kolom 'Device Type' dari *dataframe* ‘df’. Hasilnya disimpan dalam variabel DT_count, yang kemudian berisi Seri Pandas yang berisi nilai unik sebagai indeks dan jumlah kemunculannya sebagai nilai.

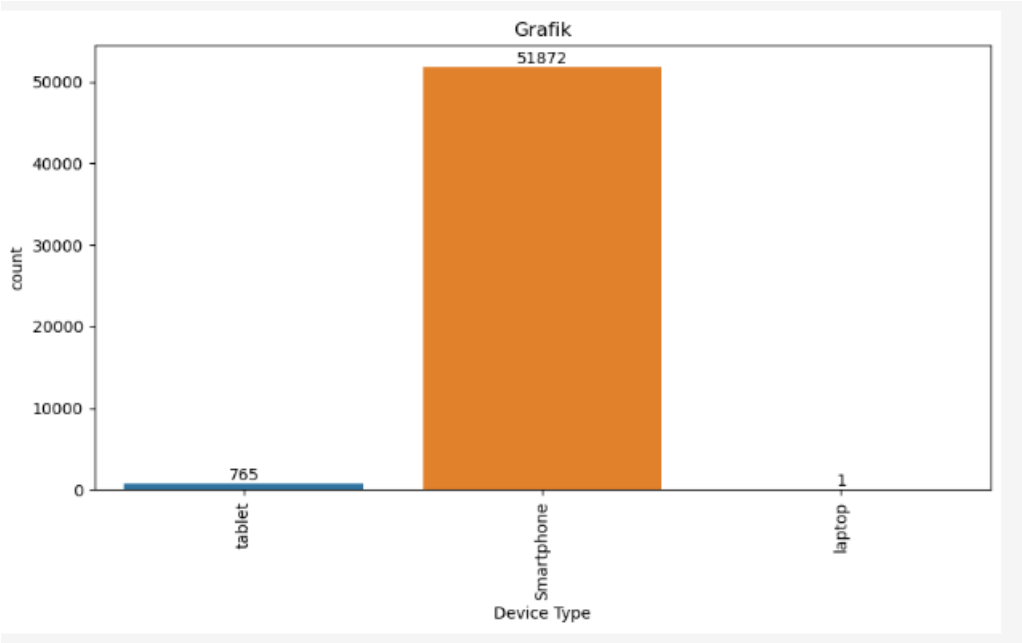
Baris kode kedua digunakan untuk mengganti nilai tertentu dalam kolom 'Device Type'. Dalam hal ini, nilai 'smartphone' diubah menjadi 'Smartphone' dikarenakan data dengan nilai tersebut mengalami reducancy dengan tulisan yang berbeda.

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

✓ 0.1s

Gambar 3. 10 Kode Menampilkan Grafik Device Type

Baris kode pada gambar 3.10 digunakan untuk membuat dan menampilkan grafik batang (*bar plot*) yang menunjukkan distribusi jumlah kemunculan setiap nilai unik dalam kolom 'Device Type' dari *dataframe* ‘df’. Kode tersebut menghasilkan grafik batang dengan label angka yang menunjukkan frekuensi masing masing kategori.



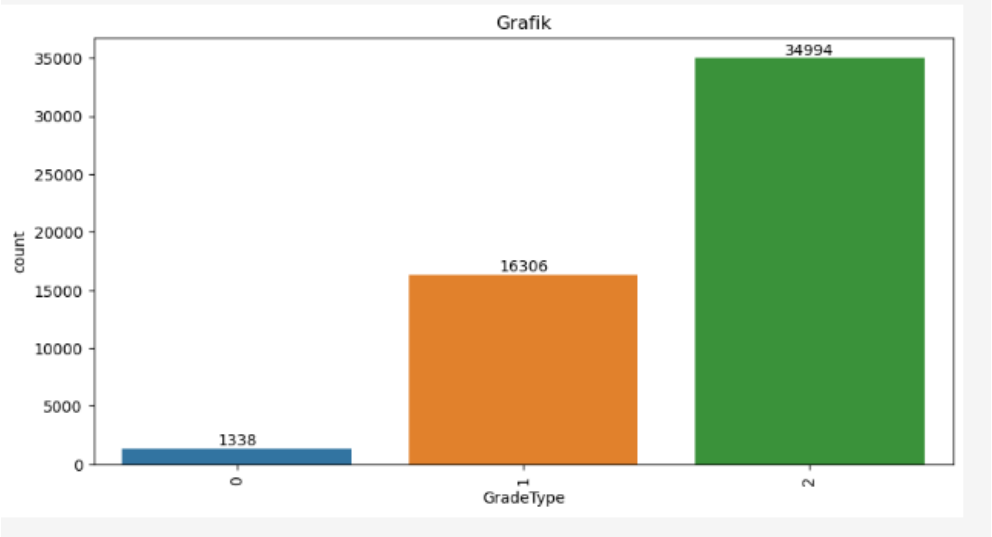
Gambar 3. 11 Visualisasi Grafik Batang Device Type

Pada gambar 3.11 dapat dilihat hasil visualisasi *bar plot* dari kolom data ‘Device Type’. Hasil visualisasi tersebut menunjukkan ada terdapat 3 tipe *device* yang terjual di Laku6 dalam periode waktu sebelas bulan dalam tahun 2023.

Terdapat perbedaan signifikan yaitu tipe *smartphone* memiliki jumlah penjualan terbanyak dibandingkan dengan tipe *device* lainnya.

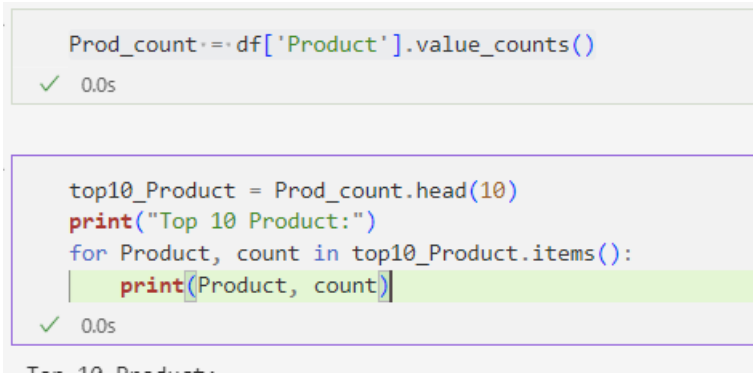
```
plt.figure(figsize=(10,5))
ax = sns.countplot(x='GradeType', data=df)
ax.bar_label(ax.containers[0])
plt.title('Grafik')
plt.xticks(rotation=90)
plt.show()
```

✓ 0.1s



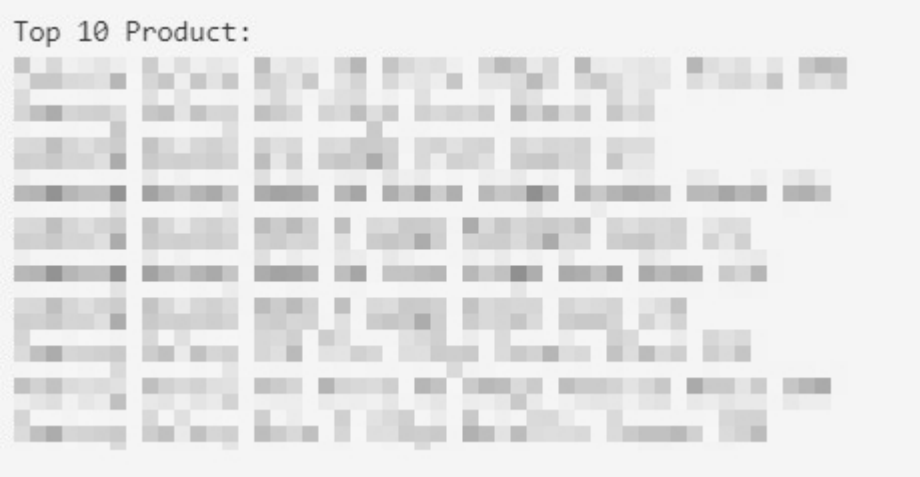
Gambar 3. 12 Menampilkan Visualisasi Grafik Batang Grade Type

Gambar 3.12 menunjukkan kode dan juga hasil visualisasi berupa *bar plot* dari kolom data ‘GradeType’. Hasil visualisasi tersebut menunjukan ada tiga *grade* dari penjualan unit di Laku6, tiga grade ini masing masing menunjukkan kategori unit dari mati total yang ditunjukkan kategori dengan angka 0 (nol), kemudian ada *grade* dengan kategori angka 1 (satu) yaitu unit yang butuh diperbaiki, dan angka 2 (dua) adalah tipe *grade* unit yang sudah siap jual dalam kondisi bagus ataupun hanya memiliki sedikit kecacatan.



Gambar 3. 13 Kode Identifikasi Top 10 Product

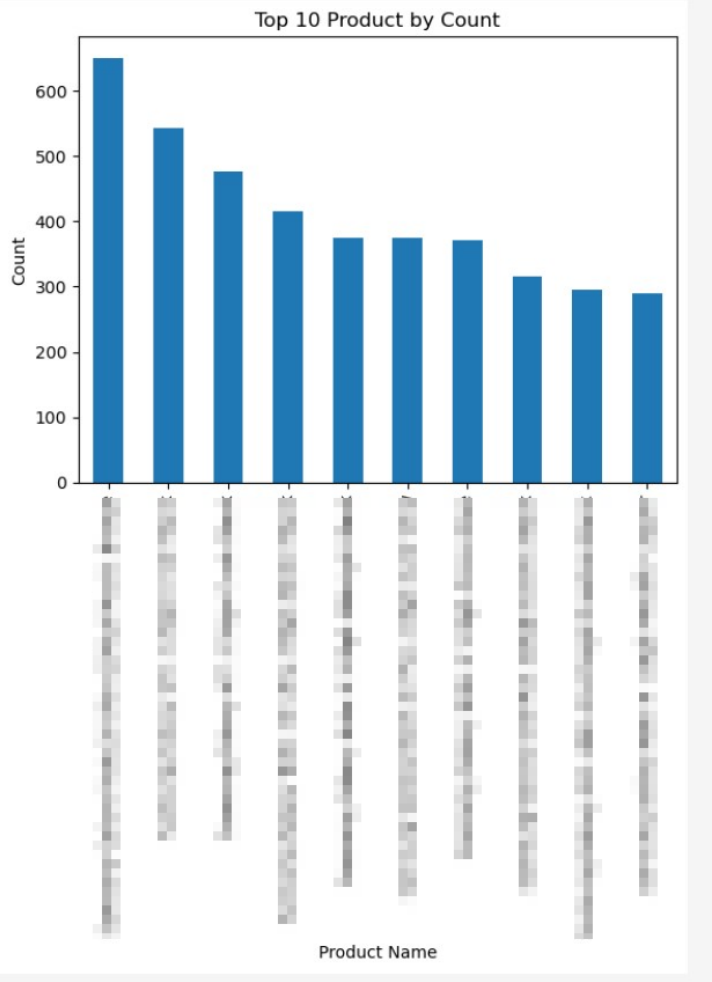
Kode pada gambar 3.13 berfokus pada mengidentifikasi dan menampilkan sepuluh produk teratas beserta jumlah kemunculannya dalam *dataframe*. Dari hasil sepuluh produk teratas ini dapat dijadikan *insight* untuk mengoptimalkan produk-produk yang akan diperjual-belikan kedepannya.



Gambar 3. 14 Output List Top 10 Product

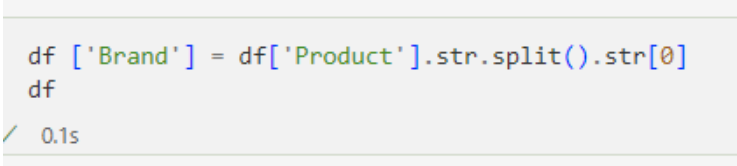
Gambar 3.14 menunjukkan daftar sepuluh produk teratas dalam bentuk sebuah list sehingga dapat diketahui produk produk yang paling banyak terjual serta jumlahnya. Namun untuk kerahasiaan perusahaan, data tersebut akan diberlakukan sensor.

```
top10_Product.plot(kind="bar")
plt.title("Top 10 Product by Count")
plt.xlabel("Product Name")
plt.ylabel("Count")
plt.show()
```
✓ 0.1s



Gambar 3. 15 Visualisasi Grafik Batang Top 10 Product

Dalam visualisasi data, hasil pengolahan data sepuluh produk teratas dapat dilihat dalam bentuk *bar plot*. Hal ini mempermudah jika ingin melihat perbandingan dari masing-masing produk yang terjual dilihat dari frekuensi jumlah terjualnya produk tersebut.



Gambar 3. 16 Kode Pembuatan Kolom Brand

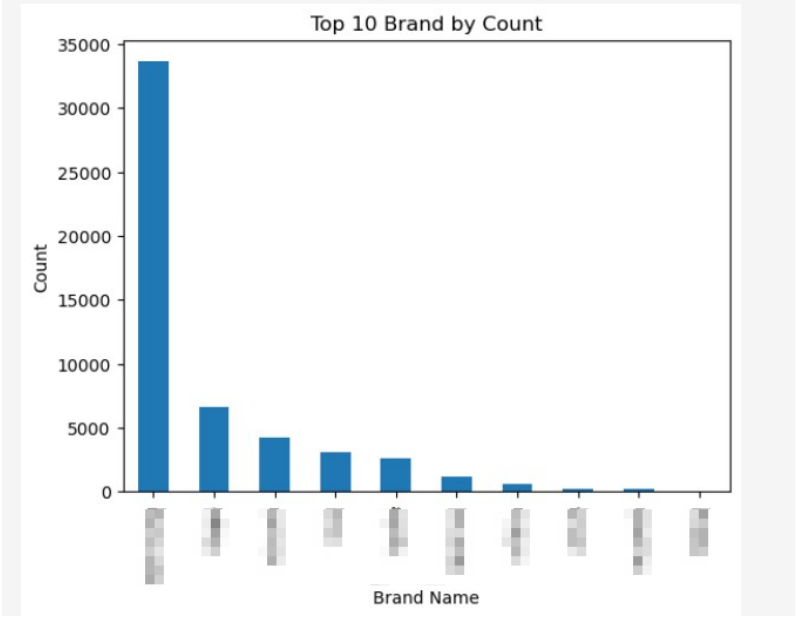
Kode pada gambar 3.16 digunakan untuk membuat kolom baru dalam *dataframe* ‘df’ yang disebut 'Brand'. Fungsi str.split**()** digunakan untuk membagi setiap nilai dalam kolom 'Product' berdasarkan spasi, sehingga membentuk daftar kata-kata. Fungsi str[0] diaplikasikan untuk setiap elemen daftar tersebut, menghasilkan elemen pertama (kata pertama) dari setiap nilai dalam kolom 'Product' sehingga kolom ‘Brand’ berisi kata pertama dari produk.

Gambar 3. 17 Visualisasi List Top 10 Brand

Setelah kolom ‘Brand’ sudah terdapat di dalam *dataframe*, maka dapat dihitung jumlah kemunculan nilai unik dari kolom tersebut dalam *dataframe*. Yang dilanjutkan dengan visualisasi berupa daftar berbentuk *list* untuk menunjukan sepuluh *brand* teratas yang terjual dalam data penjualan perusahaan.

```
top10_Brand.plot(kind="bar")
plt.title("Top 10 Brand by Count")
plt.xlabel("Brand Name")
plt.ylabel("Count")
plt.show()
```

✓ 0.1s Python

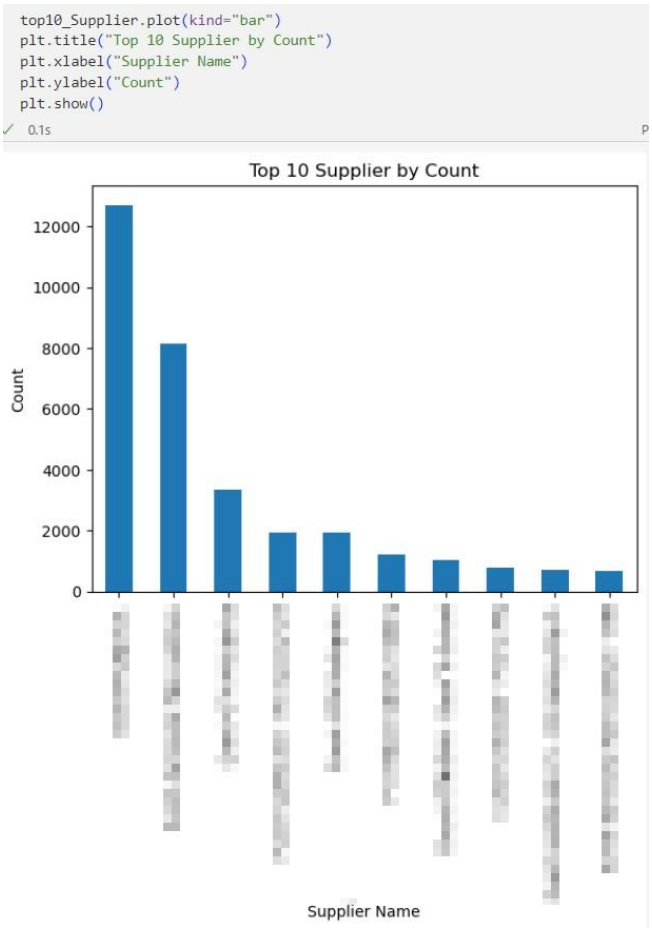


Gambar 3. 18 Visualisasi Grafik Batang Top 10 Brand

Bar plot juga digunakan untuk memberikan kejelasan perbedaan dari perbandingan tiap *brand* yang terjual. Berdasarkan gambar hasil visualisasi, terdapat sebuah *brand* yang memiliki jumlah penjualan jauh di atas *brand* lainnya dengan mencapai lebih dari tiga puluh ribu unit terjual.

Gambar 3. 19 Visualisasi List Top 10 Supplier

Gambar 3.19 menunjukkan kode untuk menampilkan sepuluh *supplier* teratas yang menghasilkan jumlah penjualan tertinggi. Supplier dalam data ini mengartikan *event* program reguler maupun bersama partner yang digunakan dalam transaksi pembelian atau tukar tambah *smartphone*.



Gambar 3. 20 Visualisasi Grafik Batang Top 10 Supplier

Berikut merupakan hasil visualisasi berupa *bar plot* untuk menunjukkan sepuluh *supplier* teratas yang memiliki penjualan tertinggi. Berdasarkan gambar visualisasi hasil *coding* menunjukkan bahwa terdapat satu *supplier* yang jumlah penjualannya lebih tinggi secara signifikan dibandingkan *supplier* lainnya dengan jumlah diatas dua belas ribu penjualan.

3.2.4 Minggu 8-11: Proyek 2: *Clustering* Produk Laku6 Berdasarkan Frekuensi Penjualan dan Rata-Rata Keuntungan.

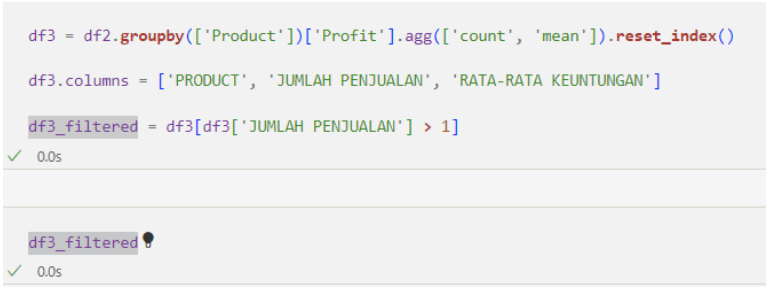
Proyek *clustering* produk Laku6 berdasarkan frekuensi penjualan dan rata-rata keuntungan diberikan untuk mengetahui kelompok *product* yang memiliki jumlah penjualan dengan rata-rata keuntungan tertinggi. Periode pengerjaan proyek ini yaitu 02 Oktober 2023 – 27 Oktober 2023.

Gambar 3. 21 Pembuatan Sampel Data

Sebelum kode pada gambar 3.21 dijalankan, impor *library* yang dibutuhkan dalam proses *coding*. Kemudian *read* data yang ingin digunakan, lalu dikarenakan jumlah data yang terlalu banyak, dapat dibuat sample seperti pada baris pertama dalam gambar. Pada proyek *clustering* ini, sample data yang digunakan sejumlah lima ribu data.

Gambar 3. 22 Pengecekan Jumlah Row Column dan Jumlah Nilai Unik pada Kolom Product

Dalam dua baris kode pada gambar 3.22, df2.shape memberikan dimensi dari *dataframe* df2, yaitu jumlah baris dan kolom setelah sebelumnya sample data yang ingin digunakan dibuat dalam *dataframe* baru. Selanjutnya, kode df2['Product'].nunique() digunakan untuk menghitung jumlah nilai unik dalam kolom 'Product'. Fungsi nunique() berguna untuk mengetahui seberapa banyak produk yang berbeda dalam kolom 'Product', memberikan gambaran tentang keragaman produk dalam dataset. Informasi ini dapat menjadi dasar penting dalam analisis dan pemahaman lebih lanjut tentang data yang sedang diolah.



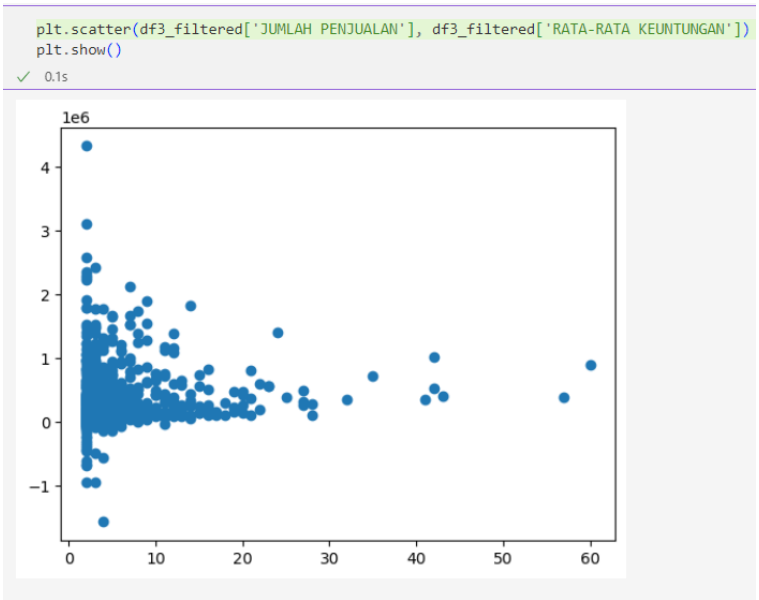
Gambar 3. 23 Pengelompokkan Kolom Product dan Profit

Gambar 3.23 menunjukkan kode yang menggunakan fungsi ‘groupby’ dan terdapat juga fungsi ‘count’ dan ‘mean’. Fungsi ini berguna untuk mengelompokkan data dalam *dataframe* berdasarkan kolom ‘Product’ lalu menghitung jumlah penjualan dan rata rata keuntungan untuk setiap produk lalu disimpan dalam *dataframe* baru. Dilanjutkan dengan mengubah nama kolom dalam *dataframe* baru untuk memberikan label yang mendeskripsikan data tersebut.

	PRODUCT	JUMLAH PENJUALAN	RATA-RATA KEUNTUNGAN
1		3	
3		2	
8		2	
13		4	
15		4	
...		...	
1319		6	
1321		5	
1322		2	
1323		4	
1324		4	

Gambar 3. 24 Hasil Print Dataframe

Gambar 3.24 merupakan hasil *print* dari *dataframe* setelah dilakukan 'groupby' dan memasukkan kolom ke *dataframe* baru. Gambar tersebut menunjukkan visualisasi *dataframe* dengan kolom 'PRODUCT', 'JUMLAH PENJUALAN', dan 'RATA RATA KEUNTUNGAN'.



Gambar 3. 25 Scatter Plot Jumlah Penjualan dan Rata Rata Keuntungan

Gambar 3.25 menunjukkan visualisasi berupa diagram *scatter plot* dengan menggunakan data dari *dataframe*. Sumbu X merupakan 'JUMLAH

PENJUALAN' dan sumbu Y merupakan 'RATA-RATA KEUNTUNGAN'. *Scatter plot* ini berguna untuk melihat data persebaran hubungan antara jumlah antar sumbu X dan Y.

```
#pembentukan model cluster membutuhkan setidaknya 2 input numerik
#pada kasus ini, digunakan variabel jumlah penjualan dan rata2 keuntungan
#tujuan:melihat model mana yg memiliki jumlah penjualan & keuntungan rendah, sedang, tinggi

X = df3_filtered[['JUMLAH PENJUALAN', 'RATA-RATA KEUNTUNGAN']]
```
✓ 0.0s

Gambar 3. 26 Pembuatan Variabel Berisi Subset

Variabel X dalam kode pada gambar 3.26 berisi subset dari *dataframe* yang mencakup dua kolom. Menyimpan data *subset* dalam variabel terpisah ini memungkinkan dan memudahkan untuk untuk melakukan operasi atau pemrosesan data lebih lanjut tanpa mengubah *dataframe* asli.

```
from sklearn.cluster import KMeans

Kmean = KMeans(n_clusters=3)
Kmean.fit(X)
```
✓ 0.1s

```
KMeans(n_clusters=3)
```

```
Kmean.cluster_centers_
```
✓ 0.0s

```
array([[5.51190476e+00, 1.59320515e+05],
       [7.24637681e+00, 7.66225927e+05],
       [5.02702703e+00, 1.83079205e+06]])
```

Gambar 3. 27 Clustering dengan algoritma K-Means dan Menentukan Titik Sentroid

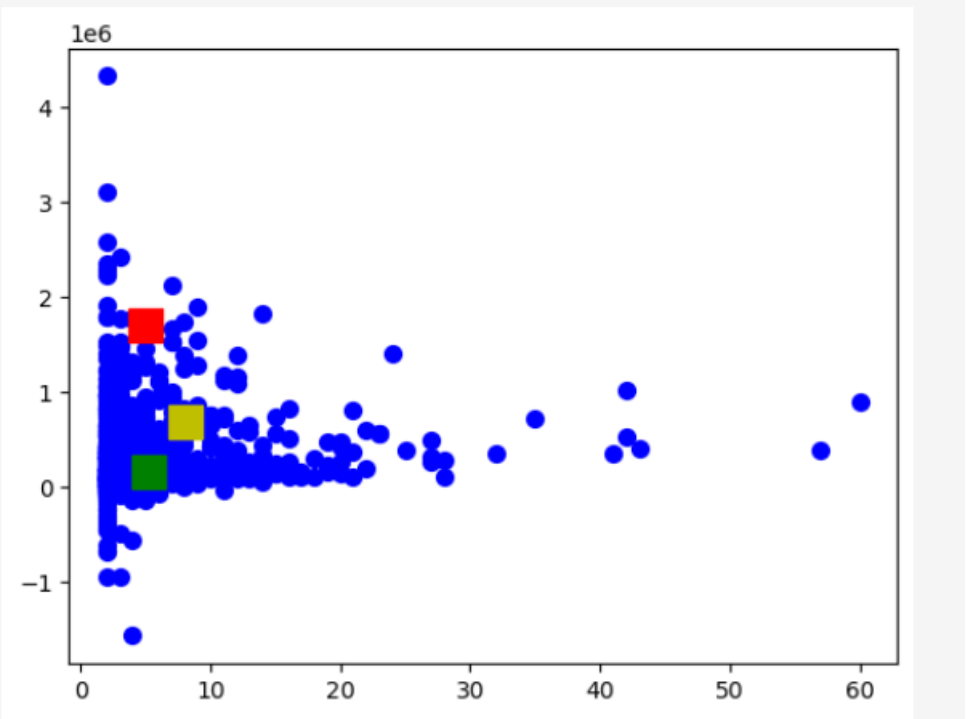
Gambar 3.27 menunjukkan model *clustering* yang dipakai adalah K-Means dengan melakukan impor dari *library* scikit-learn dengan menunjukkan parameter yang berarti mengelompokkan menjadi tiga *cluster*. K-Means dipilih karena selain merupakan algoritma favorit yang digunakan dalam *clustering* karena merupakan salah satu yang paling banyak dipakai, algoritma ini juga relatif mudah diimplementasikan untuk *dataset* yang cukup besar. Kode pada *line* kedua

digunakan agar dapat menampilkan array agar titik tengah dari *cluster* yang terbentuk dapat terlihat.

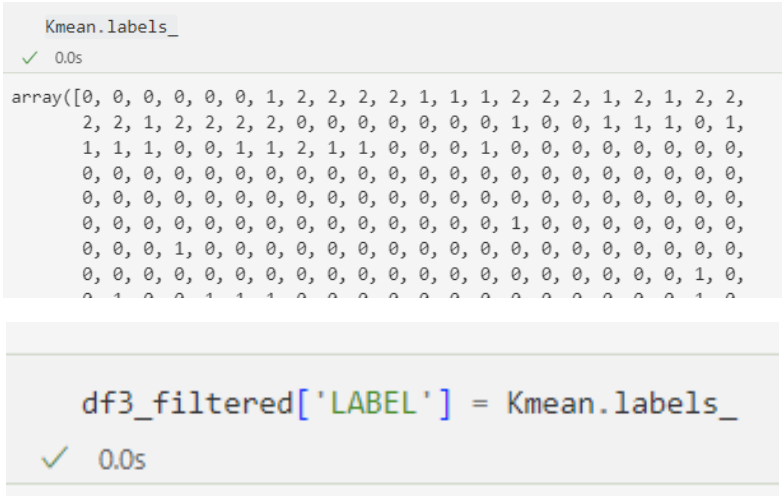
Gambar 3.28 adalah hasil dari pembentukan *scatter plot* dengan adanya tanda nilai titik tengah setelah memasukkan array yang didapatkan dari ‘Kmean.cluster_centers_’ untuk mengetahui bagian titik sentral dari tiap cluster dengan ditandai tiap titik sentroid dengan warna yang berbeda.

```
plt.scatter(df3_filtered['JUMLAH PENJUALAN'], df3_filtered['RATA-RATA KEUNTUNGAN'], s =50, c='b')
plt.scatter(5.18971631e+00, 1.53241241e+05, s=200, c='g', marker='s')
plt.scatter(5.02040816e+00, 1.70061906e+06, s=200, c='r', marker='s')
plt.scatter(8.02531646e+00, 6.80414967e+05, s=200, c='y', marker='s')
plt.show()
```

✓ 0.1s

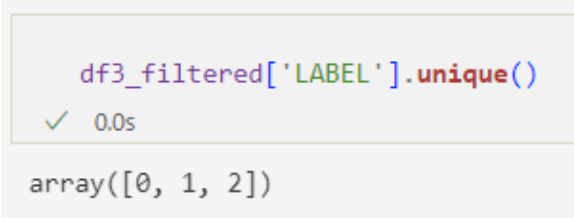


Gambar 3. 28 Titik Sentroid pada Scatter Plot



Gambar 3. 29 Labelling Cluster dan Disimpan dalam Kolom Label

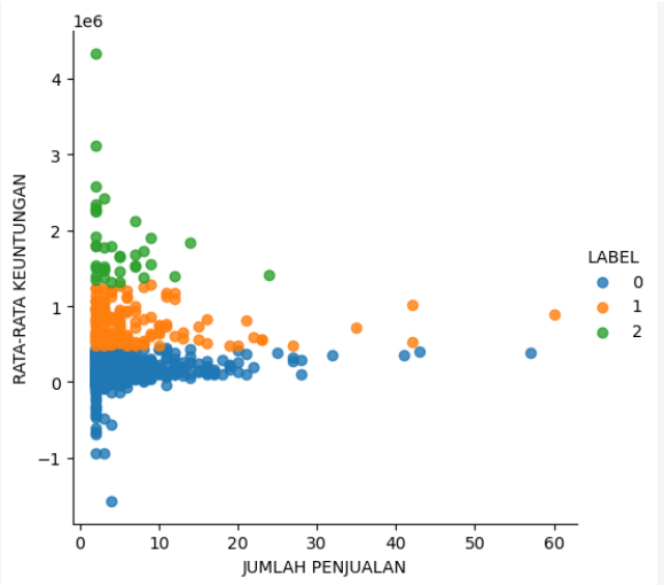
Kode pada gambar 3.29 digunakan untuk mendapatkan label *cluster* yang ditentukan oleh algoritma K-Means untuk setiap data dalam variabel X. Label-label tersebut kemudian disimpan dalam kolom ‘LABEL’ pada *dataframe*.



Gambar 3. 30 Mendapatkan Nilai Unik dari Kolom LABEL

Kode pada gambar 3.30 digunakan untuk mendapatkan nilai unik dari kolom ‘LABEL’. Hal ini dilakukan untuk dapat mengetahui *cluster* yang telah dibentuk oleh algoritma K-Means. Setiap nilai unik dalam kolom tersebut mencerminkan *cluster* tertentu yang telah diberikan oleh algoritma, dalam dataset ini *cluster* dibagi menjadi ‘0’, ‘1’, dan ‘2’.

```
import seaborn as sns
sns.lmplot('JUMLAH PENJUALAN', 'RATA-RATA KEUNTUNGAN', data=df3_filtered, hue='LABEL', fit_reg=False)

plt.show()

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



Gambar 3. 31 Scatter Plot Labelling

Kode yang digunakan pada gambar 3.31, Menggunakan fungsi ‘lmplot’ dari Seaborn untuk membuat scatter plot. Parameter pertama dan kedua ('JUMLAH PENJUALAN' dan 'RATA-RATA KEUNTUNGAN') menentukan variabel yang akan diplot pada sumbu x dan y. Kemudian menunjukkan *clustering* dengan memberikan warna yang berbeda.

```
df3_filtered['LABEL'].replace({0:'RENDAH',
                               1:'SEDANG',
                               2: 'TINGGI'}, inplace=True)
```

0.0s

Gambar 3. 32 Mengganti Label Angka dengan Karakter

Dilanjutkan dengan mengganti nilai numerik dalam kolom tersebut dengan label kategori yang lebih deskriptif agar memudahkan pengertian dari clustering dilihat dari gambar visualisasi yang nantinya akan dihasilkan dengan menambahkan label tersebut. Nilai 0 akan diganti dengan 'RENDAH', nilai 1 dengan 'SEDANG', dan nilai 2 dengan 'TINGGI'

df3_filtered

0.0s Python

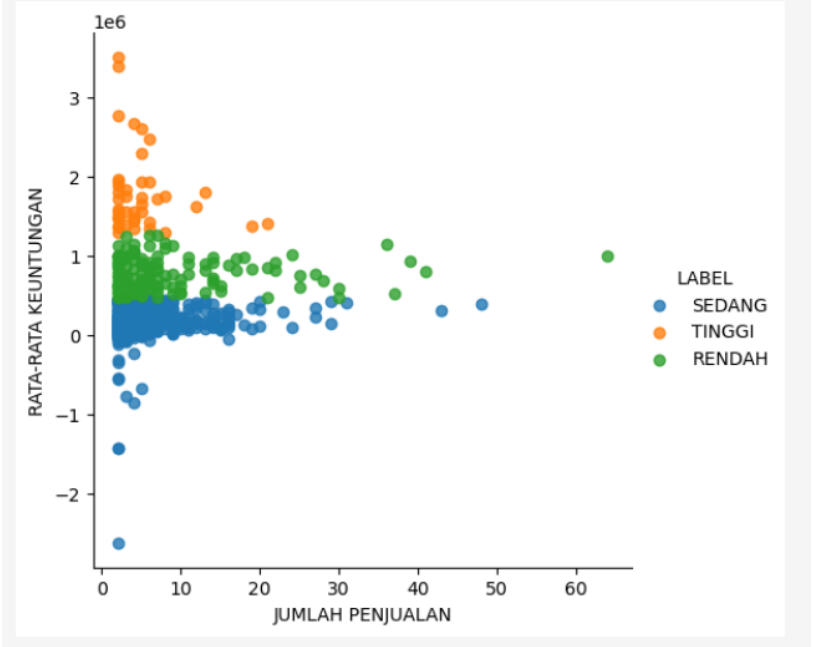
	PRODUCT	JUMLAH PENJUALAN	RATA-RATA KEUNTUNGAN	LABEL
1	[illegible]	3	[illegible]	RENDAH
3	[illegible]	2	[illegible]	RENDAH
8	[illegible]	2	[illegible]	RENDAH
13	[illegible]	4	[illegible]	RENDAH
15	[illegible]	4	[illegible]	RENDAH
...	...	...	...	...
1319	[illegible]	6	[illegible]	SEDANG
1321	[illegible]	5	[illegible]	RENDAH
1322	[illegible]	2	[illegible]	RENDAH
1323	[illegible]	4	[illegible]	RENDAH
1324	[illegible]	4	[illegible]	RENDAH

Gambar 3. 33 Hasil Print Dataframe setelah Labelling dengan Karakter

Gambar 3.33 menunjukkan hasil print dari *dataframe* yang menunjukkan kolom baru Bernama ‘LABEL’ untuk menunjukkan deskripsi clustering dari masing masing ‘Product’ secara manual untuk mempermudah interpretasi, analisis data dan visualisasi.

```
import seaborn as sns
sns.lmplot('JUMLAH PENJUALAN', 'RATA-RATA KEUNTUNGAN', data=df3_filtered, hue='LABEL', fit_reg=False)

plt.show()
```

0.3s



Gambar 3. 34 Hasil Visualisasi Scatter Plot dengan Labelling Karakter

Gambar 3.34 merupakan visualisasi setelah plot dibentuk ulang dengan menggunakan label yang sudah diubah dengan karakter yang lebih deskriptif. Dengan menggunakan warna dan label yang berbeda untuk setiap kategori ('RENDAH', 'SEDANG', 'TINGGI'), scatter plot tersebut memvisualisasikan distribusi produk berdasarkan jumlah penjualan dan rata-rata keuntungan dengan lebih jelas, memudahkan pemahaman dan analisis data.

3.2.5 Minggu 12-14: Proyek 3: Model F*orecasting Sales*

```
df2 = df.groupby(df['Success Date'].dt.date).size().reset_index()
df2.columns = ['TANGGAL', 'SALES']
```
0.0s

```
df2.head(3)
```
0.0s

	TANGGAL	SALES
0	2023-01-02	[illegible]
1	2023-01-03	[illegible]
2	2023-01-04	[illegible]

Gambar 3. 35 Pembuatan Dataframe dan Kolom Baru

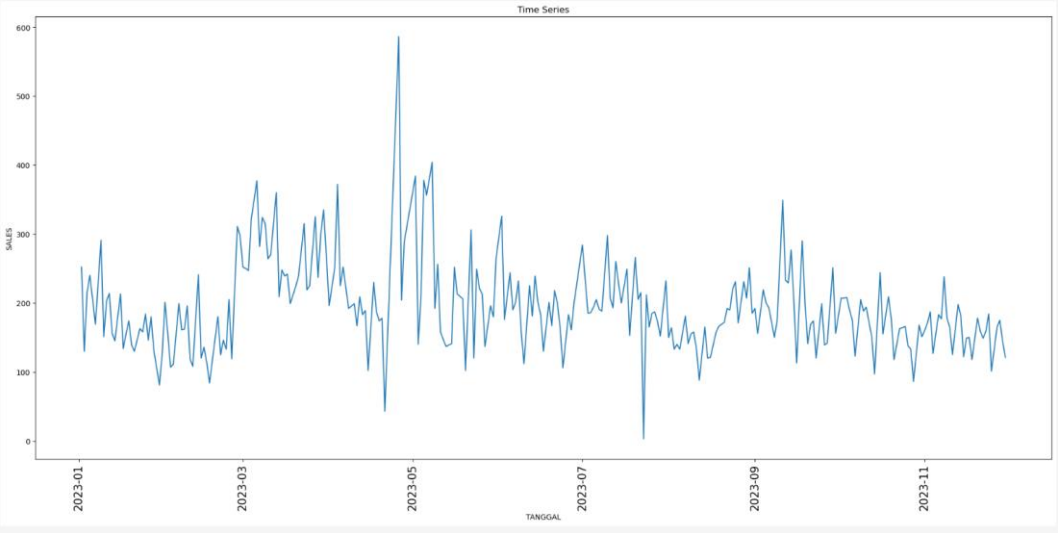
Dengan menggunakan kode tersebut, kita dapat membuat *dataframe* ‘df2’ yang berisi informasi tentang jumlah penjualan pada setiap tanggal yang ada dalam data asli ‘df’. Data ini dapat digunakan untuk melakukan analisis dan visualisasi tren penjualan sepanjang waktu. Nama kolom hasil pengelompokkan diganti menjadi 'TANGGAL' untuk kolom tanggal dan 'SALES' untuk kolom jumlah penjualan.

Gambar 3.36 menggunakan kode visualisasi menggunakan ‘seaborn’ agar dapat melihat deret waktu dari data penjualan per tanggal yang dihasilkan dari *dataframe* dalam bentuk *time series*. Hasil visualisasi berbentuk grafik ini membantu dalam memahami tren dan pola penjualan sepanjang waktu. Dari hasil

grafik, ditunjukkan lonjakan pada periode tertentu mengalami puncak penjualan tertinggi maupun terendah, serta trend tiga bulan terakhir yang mengalami penurunan dari segi jumlah penjualan.

```
import seaborn as sns
import matplotlib.pyplot as plt

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

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

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

plt.show()
```

0.2s

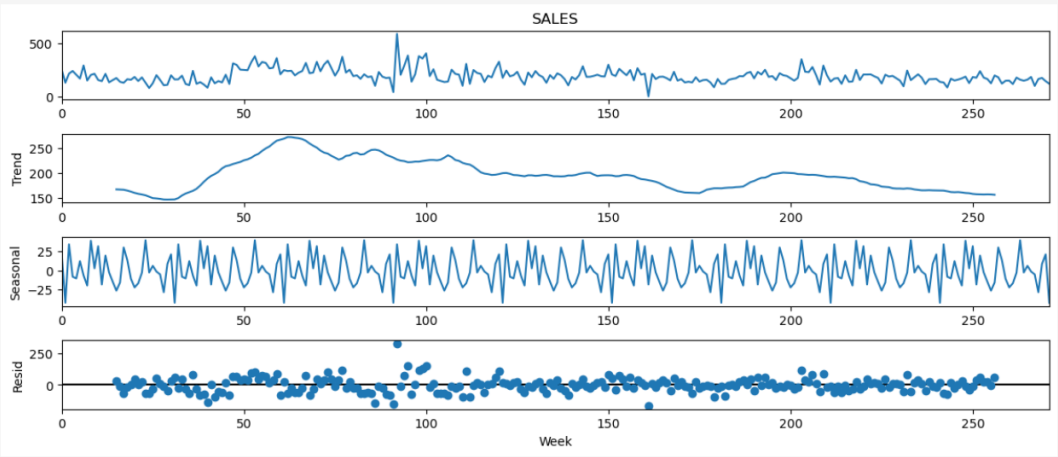


Gambar 3. 36 Visualisasi Time Series Data TANGGAL dan SALES

```
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. 37 Kode Seasonal Decompose

Baris kode ini menggunakan *library* 'statsmodels' untuk melakukan *seasonal decompose* pada deret waktu penjualan. Dengan menggunakan dekomposisi musiman, kita dapat memvisualisasikan komponen-komponen utama dari deret waktu penjualan, yaitu tren, musiman, dan residu. Hal ini membantu untuk mengidentifikasi pola musiman yang mungkin ada dalam data dan memahami struktur dasar dari tren dan fluktuasi musiman. Dalam konteks ini, periode musiman diestimasi dengan panjang 30 (mewakili jumlah hari dalam sebulan).



Gambar 3. 38 Visualisasi Hasil Seasonal Decompose

Hasil visualisasi yang dilakukan yaitu adalah tiga komponen *Trend*, *Seasonal*, dan Residu. Dengan memahami ketiga komponen tersebut, maka dapat mendapatkan wawasan lebih mendalam tentang struktur dasar dari deret waktu dan membuat prediksi atau interpretasi yang lebih akurat terkait perubahan dan pola yang terkandung dalam data.

```
sales_train = df2[:int(0.9*(len(df2)))]
sales_test = df2[int(0.9*(len(df2))):]
```
0.0s

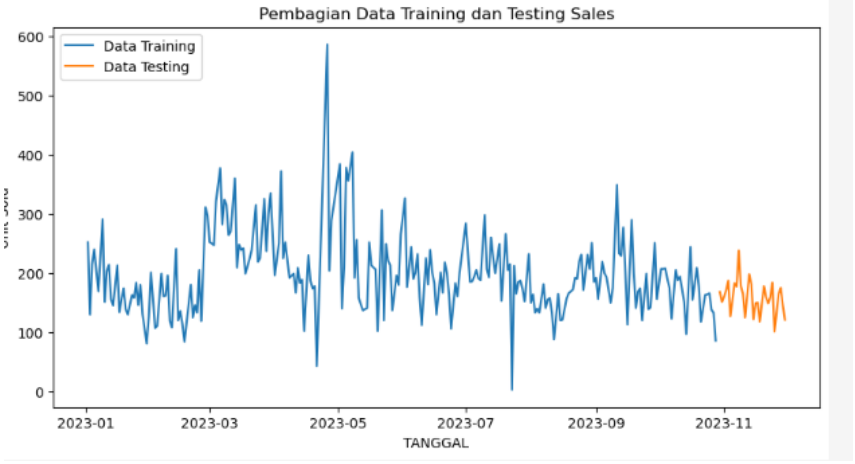
Gambar 3. 39 Split Data Train dan Test Rasio 90:10

Untuk melanjutkan proses berikut, dilakukan *split* data menjadi data *train* dan data *test* yang nantinya digunakan untuk menguji performa model pada data yang akan diuji untuk membantu mengukur sejauh mana model dapat melakukan

prediksi yang baik. Dalam kode pada gambar 3.39, *split* data dilakukan dengan rasio 90% untuk *training* dan 10% untuk testing.

```
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()
```

0.2s Python



Gambar 3. 40 Hasil Visualisasi Split Data

Setelah *split* data dilakukan, maka dapat dilakukan visualisasi dari hasil tersebut. Visualisasi dilakukan dengan menggunakan *line chart*, hal ini dapat dilihat perbedaan penjualan antara data *train* dan data *test* dalam rentang waktu tertentu. Pembagian data *split* dan data *test* dipisah menjadi dua warna, biru dan oranye.

Memasuki tahap pemodelan *forecasting*, model yang digunakan yaitu ARIMA dalam membuat prediksi terhadap data penjualan yang disediakan dalam 'sales_train'. Dari hasil *summary*, dapat ditampilkan ringkasan statistik model tersebut yang mencakup informasi seperti nilai koefisien, p-*value* dan statistik lain yang membantu dalam mengevaluasi kualitas model sehingga dapat memberikan wawasan untuk seberapa baik suatu model dengan data *train*.

```
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()
```

✓ 0.0s

SARIMAX Results

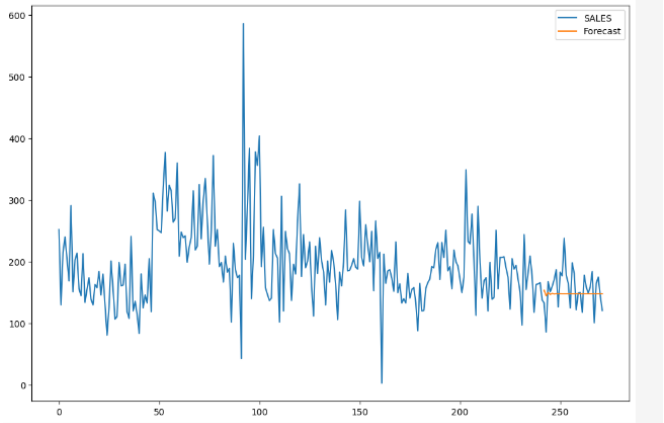
Dep. Variable:	SALES			No. Observations:	244	
Model:	ARIMA(1, 1, 0)			Log Likelihood	-1372.635	
Date:	Wed, 06 Dec 2023			AIC	2749.270	
Time:	01:32:42			BIC	2756.256	
Sample:	0			HQIC	2752.084	
	- 244					
Covariance Type:	opg					
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5562	0.047	-11.773	0.000	-0.649	-0.464
sigma2	4713.3325	214.011	22.024	0.000	4293.879	5132.786
Ljung-Box (L1) (Q):	10.44		Jarque-Bera (JB):	832.76		
Prob(Q):	0.00		Prob(JB):	0.00		
Heteroskedasticity (H):	0.67		Skew:	1.41		
Prob(H) (two-sided):	0.08		Kurtosis:	11.62		

Gambar 3. 41 Pemodelan Forecasting Menggunakan ARIMA

```
df2['Forecast']=results.predict(start=242,end=272,dynamic=True)
df2[['SALES','Forecast']].plot(figsize=(12,8))
```

✓ 0.1s Python

```
AxesSubplot:>
```

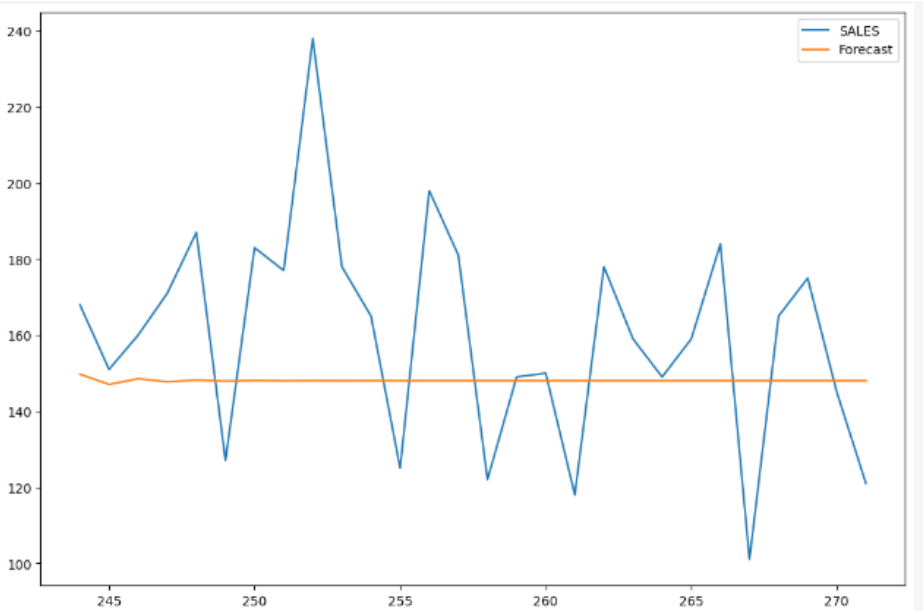


Gambar 3. 42 Hasil Visualisasi Forecasting Menggunakan ARIMA

Berikut merupakan visualisasi bagaimana *forecasting* menggunakan model ARIMA. Membandingkan garis prediksi dengan data asli (SALES), sehingga hal ini dapat membantu untuk mengevaluasi sejauh mana model dapat mereplikasi pola dan tren dalam data sebenarnya.

```
sales_test['Forecast']=results.predict(start=242,end=272,dynamic=True)
sales_test[['SALES','Forecast']].plot(figsize=(12,8))
```

0.1s



Gambar 3. 43 Gambaran Lebih Jelas Visualisasi Forecasting ARIMA

Gambar 3.43 menunjukkan hasil visualisasi yang lebih jelas dari menggunakan model ARIMA pada *forecasting*. Melihat garis prediksi tersebut mengartikan bahwa hasil prediksi tidak baik dikarenakan tidak menggambarkan replikasi dari data sebenarnya.

Exponential Smoothing

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

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

0.0s

Gambar 3. 44 Kode Pembuatan Forecasting Menggunakan Model Exponential Smoothing

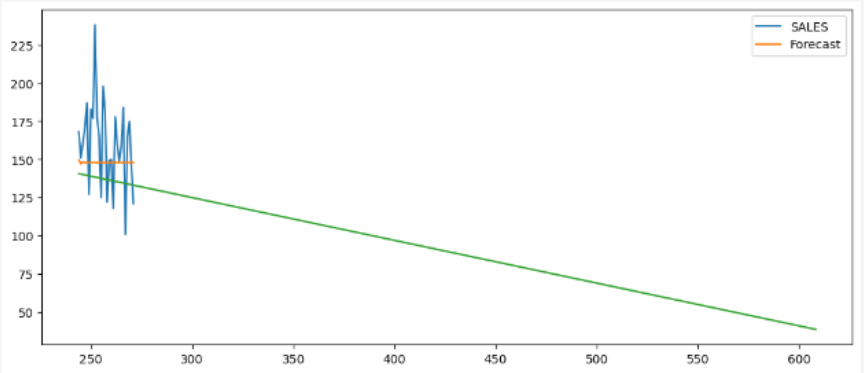
Kode tersebut menggunakan model Exponential Smoothing dari *package* statsmodels untuk melakukan *forecasting*. Menggunakan Model Exponential Smoothing untuk menangkap pola tren dan musiman dalam data dengan menggunakan pendekatan eksponensial. Penggunaan parameter trend='add' menunjukkan bahwa tren akan diakumulasi secara aditif, dan parameter

seasonal_periods=365 menunjukkan bahwa model akan memperhitungkan pola musiman setiap tahunnya.

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

✓ 0.1s Pytho

```
<AxesSubplot:>
```



Gambar 3. 45 Hasil Visualisasi Prediksi Model Exponential Smoothing

Gambar 3.45 menunjukkan plot dari hasil prediksi model Exponential Smoothing untuk 365 periode waktu ke depan. Dengan memplot data uji dan hasil prediksi pada grafik yang sama, maka dapat memvisualisasikan sejauh mana model Exponential Smoothing dapat mereplikasi pola dan tren dalam data uji. Berdasarkan hasil tersebut, model Exponential Smoothing tidak dapat mereplikasi pola dan menunjukkan hasil prediksi yang kurang baik.

```
from prophet import Prophet

model = Prophet()
```

✓ 0.0s

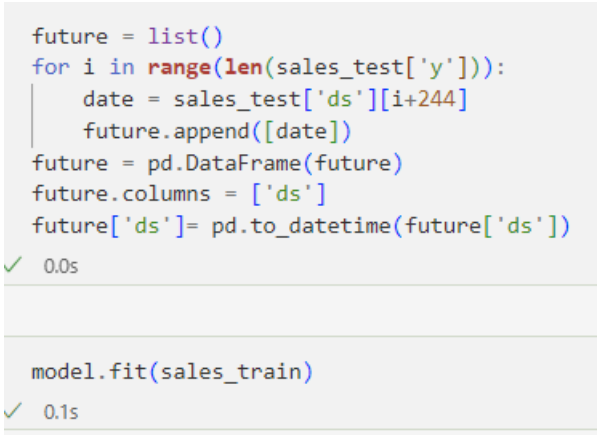
```
if 'Forecast' in sales_test.columns:
    del sales_test['Forecast']

# Mengubah nama kolom 'TANGGAL' menjadi 'ds' dan 'SALES' menjadi 'y'
sales_test.rename(columns={'TANGGAL': 'ds', 'SALES': 'y'}, inplace=True)
sales_train.rename(columns={'TANGGAL': 'ds', 'SALES': 'y'}, inplace=True)
```

✓ 0.0s

Gambar 3. 46 Kode Memanggil Model Prophet dan Penyiapan Data

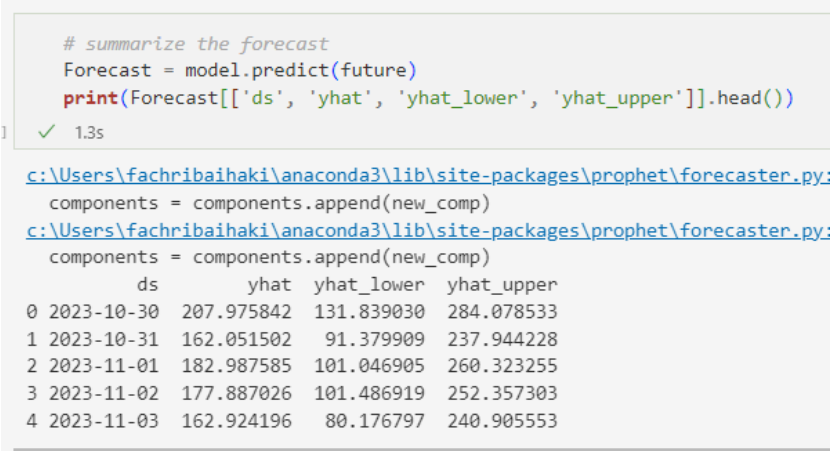
Pengecekan model berikutnya menggunakan model Prophet yang merupakan *library open-source* yang dikembangkan oleh Facebook untuk peramalan *time series* dengan menggunakan model yang dapat menangkap tren

musiman harian dan tahunan. Kode pada baris selanjutnya digunakan untuk melakukan pengecekan apakah kolom 'Forecast' sudah ada dalam *dataframe* ‘sales_test’. Jika ya, maka kolom tersebut dihapus. Ini dilakukan untuk memastikan bahwa tidak ada kolom prediksi yang tersisa atau bertumpang tindih sebelum menambahkan hasil prediksi baru. Lalu, mengubah nama kolom ‘TANGGAL’ menjadi ‘ds’, dan kolom ‘SALES’ menjadi ‘y’.



Gambar 3. 47 Menyesuaikan Format untuk Digunakan Model Prophet

Kode tersebut membuat *dataframe* ‘future’ yang akan digunakan sebagai *input* untuk membuat prediksi menggunakan model Prophet. Selanjutnya mengubah objek pada kolom ‘ds’ menjadi *datetime* sehigga dapat membuat format yang sesuai untuk digunakan sebagai input pada model Prophet untuk membuat prediksi pada periode waktu tertentu.

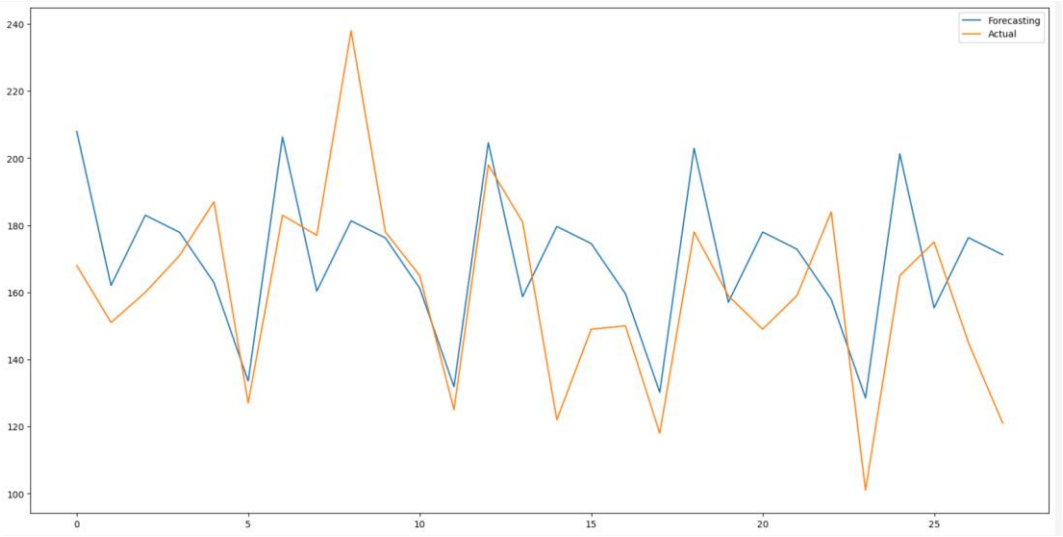


Gambar 3. 48 Ringkasan Hasil Prediksi Model Prophet

Kode ini menyajikan ringkasan hasil prediksi dari model Prophet untuk periode waktu tertentu. ‘yhat’ merupakan nilai prediksi utama (trend), ’yhat_lower’ merupakan batas bawah dari rentang prediksi dan ‘yhat_upper’ merupakan batas atas. Ringkasan ini memberikan gambaran tentang nilai prediksi utama serta tingkat keyakinan yang diberikan oleh model Prophet.

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

✓ 0.2s



Gambar 3. 49 Visualisasi Hasil Prediksi Model Prophet

Hasil dari visualisasi pada gambar 3.49 memberikan visualisasi tentang sejauh mana hasil prediksi dari model Prophet cocok dengan nilai aktual pada periode waktu yang diuji. Perbandingan visual ini dapat membantu dalam mengevaluasi kinerja model dan melihat sejauh mana prediksi tersebut mereplikasi pola dan tren yang terdapat pada data aktual.

Dari hasil visualisasi *forecasting* yang dilakukan dengan 3 model, hasil dari model ARIMA dan Exponential Smoothing tidak dapat digunakan karena garis prediksi tidak mendekati garis aktual dari data. Dengan menggunakan model Prophet, data prediksi dapat mereplika dan mendekati data aktual yang membuat

model Prophet merupakan model terbaik untuk digunakan dalam *forecasting* data penjualan.

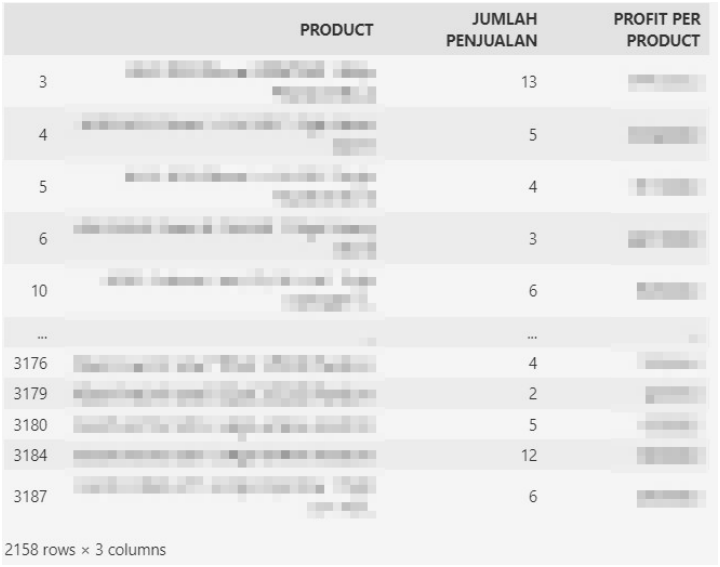
3.2.6 Minggu 15-18: Proyek 4: Prediksi Total Profit per Product

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

✓ 0.0s
```

Gambar 3. 50 Import Library yang Digunakan untuk Prediksi

Pada proyek keempat, dilakukan prediksi total profit per product. Untuk melakukan prediksi tersebut diperlukan beberapa *package* yang harus di-impor seperti LinearRegression yang merupakan model regresi dari scikit-learn yang nantinya akan diuji serta mean_squared_error untuk mengukur kesalahan prediksi model dan r2_score yaitu metrik evaluasi untuk mengukur sejauh mana variabilitas variabel target dapat dijelaskan oleh model.

```
df2 = df.groupby(['Product'])['Profit'].agg(['count', 'sum']).reset_index()

df2.columns = ['PRODUCT', 'JUMLAH PENJUALAN', 'PROFIT PER PRODUCT']

df2_filtered = df2[df2['JUMLAH PENJUALAN'] > 1]
df2_filtered
```



	PRODUCT	JUMLAH PENJUALAN	PROFIT PER PRODUCT
3	[illegible]	13	[illegible]
4	[illegible]	5	[illegible]
5	[illegible]	4	[illegible]
6	[illegible]	3	[illegible]
10	[illegible]	6	[illegible]
...		...	
3176	[illegible]	4	[illegible]
3179	[illegible]	2	[illegible]
3180	[illegible]	5	[illegible]
3184	[illegible]	12	[illegible]
3187	[illegible]	6	[illegible]

2158 rows × 3 columns

Gambar 3. 51 Hasil Print Dataframe Baru dari Pengelompokkan Data

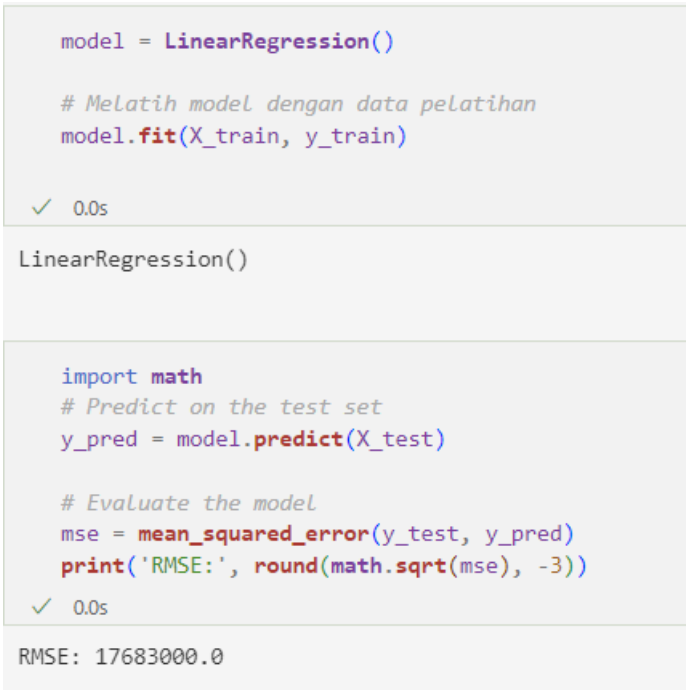
Gambar 3.51 menunjukkan kode yang digunakan untuk mengelompokkan data berdasarkan kolom 'Product' pada *dataframe* df dan menghitung jumlah penjualan dan total keuntungan ('Profit') untuk setiap produk lalu mengubah format dan diberi nama kolom yang lebih deskriptif dan disimpan dalam *dataframe* baru.

```
X = df2_filtered[['JUMLAH PENJUALAN']]
y = df2_filtered['PROFIT PER PRODUCT']

# Membagi data menjadi data pelatihan dan data pengujian (misalnya, 80% data untuk pelatihan, 20% untuk pengujian)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 0.0s

Gambar 3. 52 Mempersiapkan Data Sebelum Pemodelan

Kode pada gambar 3.52 digunakan untuk mempersiapkan data sebelum dilakukan pemodelan dengan melakukan *define* variabel X yang berisikan data ‘JUMLAH PENJUALAN’ dan variabel y berisikan data ‘PROFIT PER PRODUCT’. Setelah itu lakukan *split* data dengan rasio 80% data *train* dan 20% data *test*.

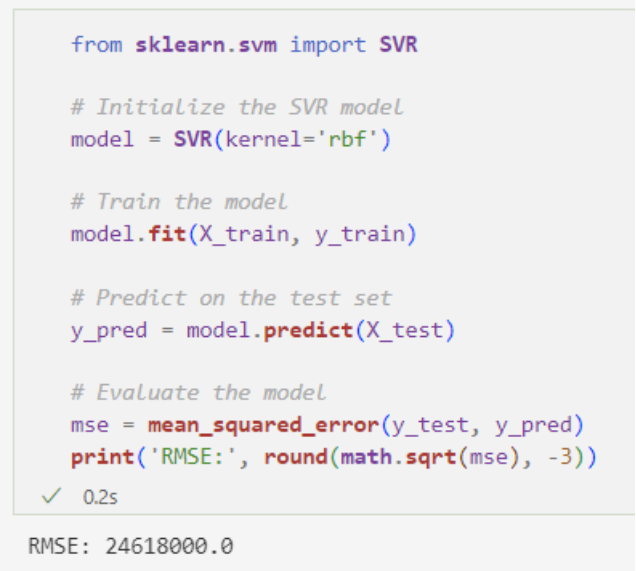


Gambar 3. 53 Pemodelan Menggunakan Linear Regression

Pemodelan pertama dilakukan dengan menggunakan model Linear Regression dengan menggunakan model yang telah dilatih untuk melakukan prediksi pada data pengujian. Impor *library* ‘math’ yang menyediakan berbagai fungsi matematika dan konstanta matematika yang dapat digunakan dalam Python.

Regresi linear adalah suatu metode dalam statistika yang digunakan untuk membuat model persamaan guna memperkirakan atau memprediksi nilai. Pendekatan ini bertujuan untuk mengidentifikasi pola pada data numerik, sehingga data yang digunakan harus berupa nilai numerik agar dapat diolah oleh algoritma ini. [8]

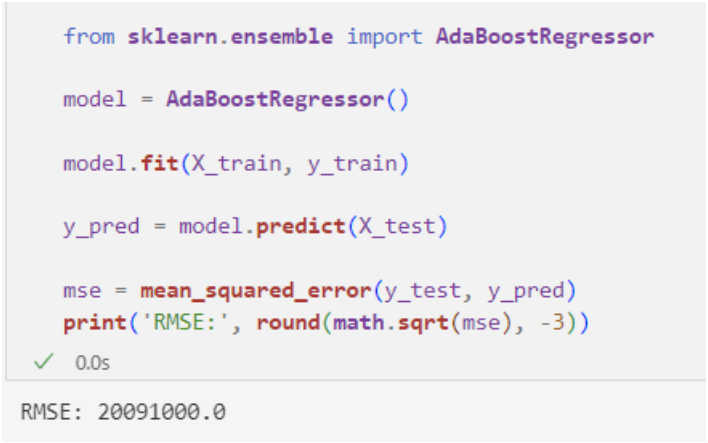
Selanjutnya mengukur performa model menggunakan metrik *Root Mean Squared Error* (RMSE). RMSE mengukur seberapa baik model memprediksi nilai target dengan membandingkan nilai prediksi dengan nilai sebenarnya. Dengan melakukan evaluasi ini, dapat dinilai seberapa baik model regresi linear dapat memprediksi nilai 'PROFIT PER PRODUCT' berdasarkan fitur 'JUMLAH PENJUALAN'. Hasil RMSE pada model Linear Regression adalah 17683000.0.



Gambar 3. 54 Pemodelan Menggunakan SVR

Gambar 3.54 menunjukkan pemodelan dengan algoritma Support Vector Regression (SVR) dari *library* Scikit-learn. SVR adalah algoritma regresi yang menggunakan mesin vektor dukungan.

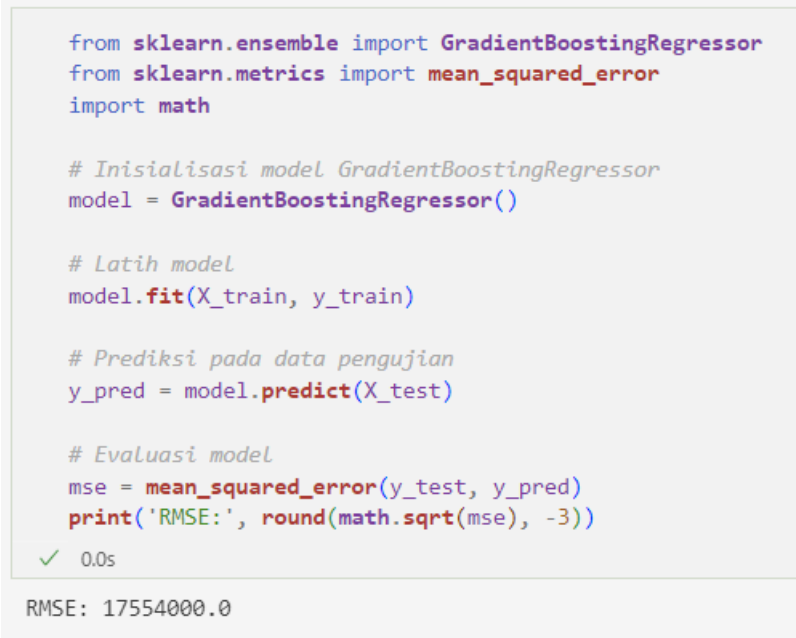
SVR (Support Vector Regression) digunakan untuk mengidentifikasi suatu fungsi yang memiliki deviasi paling besar dari target aktual y. Tujuan dari SVR adalah untuk menemukan fungsi sebagai fungsi regresi, di mana pada semua input data memiliki deviasi maksimum dari target aktual. Kelebihan penggunaan SVR adalah kecocokannya untuk data set dengan dimensi tinggi.[9] Dengan menerapkan pemodelan menggunakan algoritma SVR, nilai Root Mean Squared Error (RMSE) yang diperoleh adalah sebesar 24618000.0.



Gambar 3. 55 Pemodelan Menggunakan AdaBoostRegressor

Gambar 3.55 menunjukkan pemodelan menggunakan algoritma AdaBoostRegressor. AdaBoostRegressor adalah salah satu metode dalam machine learning yang termasuk dalam kategori ensemble learning, yang berfokus pada tugas regresi.

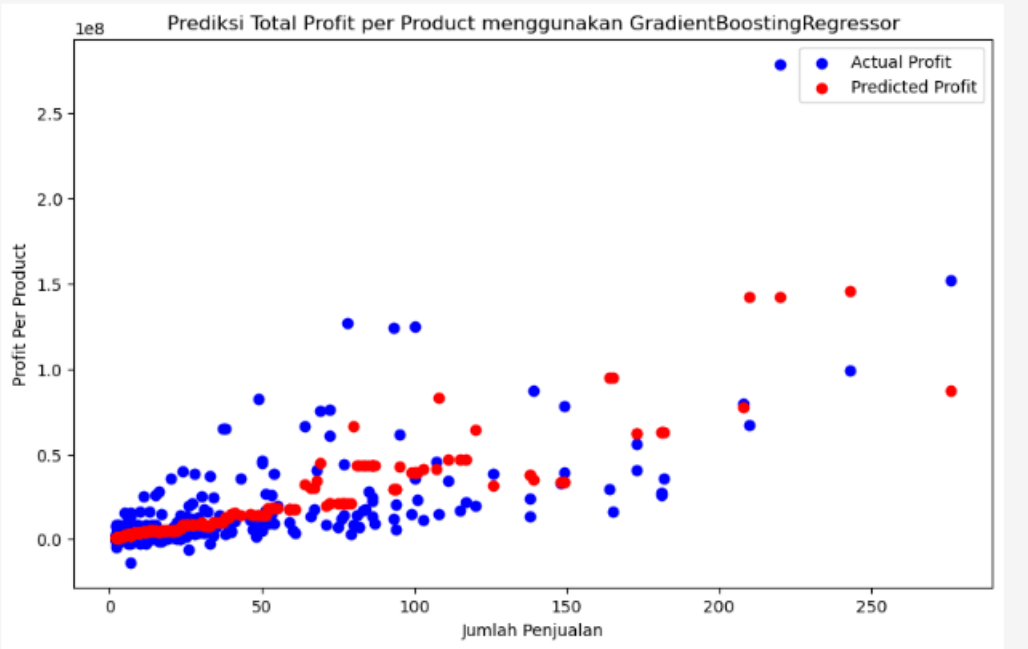
Adaboost merupakan suatu algoritma pembelajaran mesin yang dirancang untuk meningkatkan performa pohon keputusan. Algoritma Adaboost mampu mengeliminasi fitur-fitur tidak penting dalam data pelatihan dan fokus pada data yang krusial untuk pelatihan.[10] Dengan menerapkan pemodelan menggunakan algoritma AdaBoostRegressor, nilai Root Mean Squared Error (RMSE) yang diperoleh adalah sebesar 20091000.0.



Gambar 3. 56 Pemodelan Menggunakan Gradient Boosting Regressor

Gambar 3.56 menunjukkan pemodelan menggunakan algoritma Gradient Boosting Regressor. Gradient Boosting Regressor adalah salah satu metode dalam machine learning yang termasuk dalam kategori ensemble learning, yang berfokus pada tugas regresi dan model ini memanfaatkan peningkatan akurasi suatu nilai dan juga model regresi gradient boosting mampu menangani pola-pola yang rumit.[11] Dengan menerapkan pemodelan menggunakan algoritma Boosting Regressor, nilai Root Mean Squared Error (RMSE) yang diperoleh adalah sebesar 17554000.0.

Berdasarkan hasil RMSE, penggunaan algoritma Gradient Boosting Regressor merupakan algoritma yang memiliki nilai *error* atau RMSE paling kecil dibandingkan dengan 3 algoritma pemodelan lainnya yang diuji dengan jumlah RMSE sebesar 17554000.0. Gambar 3.57 menunjukkan visualisasi menggunakan *scatter plot* dari hasil pemodelan menggunakan Gradient Boosting Regressor, hasil sebaran data prediksi mendekati data aktual ditandai dengan titik yang berdekatan.



Gambar 3. 57 Visualisasi Scatter Plot Pemodelan Gradient Boosting Regressor

3.3 Kendala yang Ditemukan

Proses magang di Laku6 secara umum dapat dilakukan dengan lancar, tetapi selama menjalani program kerja magang terdapat beberapa kendala sehingga terhambatnya waktu pengerjaan dalam tugas ataupun pekerjaan yang dilakukan, berikut antara lainnya:

a. Kesulitan memahami alur bisnis, data, dan penggunaan serta fitur dari web yang digunakan perusahaan.
b. Terbatasnya kemampuan menggunakan Google Data Studio karena merupakan sebelumnya hanya sedikit mempelajari untuk pengolahan data untuk *clustering* dan *forecasting*.
c. Update harga untuk *event* yang masih manual sehingga memakan waktu yang cukup banyak dalam proses pengerjaannya dan harus dilakukan sebelum jam kerja.

3.4 Solusi atas Kendala yang Ditemukan

a. Bertanya kepada karyawan lain untuk pengertian-pengertian terhadap data yang terdapat dalam perusahaan, serta cara penggunaan dan fitur dari web perusahaan.
b. Menggunakan Virtual Studio Code: Jupyter Notebook dengan bahasa pemrograman Python untuk melakukan olah data dan visualisasi untuk proyek.
c. Mengajukan kepada pihak IT untuk membuatkan fitur otomasi dalam update harga sehingga hanya perlu memasukkan matriks harga *event.*