Rakamin Academy X KALBE Nutritionals

Time Series Forecasting and Customer Segmentation for Inventory and Marketing Optimization at Kalbe Nutritionals

Kalbe Nutritionals Data Scientist Virtual Internship Program

Presented by
Muhammad Alvero Johansyah

Muhammad Alvero Johansyah

About Me:

A third-year Undergraduate Mathematics Student at the Universitas Indonesia with a passion for data science. Experienced as a computer laboratory assistant in the Department of Mathematics, I have a basic understanding and experience in the field of data science with a few project experience. I have a keen interest in Programming Language Scripting and SQL Operation, and I am enthusiastic about applying these skills to solve complex data problems.

Rakamin Academy

Experience & Projects:

- Computer Lab Assistant on computing field in Department of Mathematics, Universitas Indonesia
- Assistant Lecturer of Numerical Method in Department of Mathematics, Universitas Indonesia
- Project: Classification of Cardiotocography to Model the Diagnosis of Problems in the Fetus using the Artificial Neural Network (ANN) Model with the Keras Library and Multilayer Perceptron (MLP) and the Decision Tree Model.
- Project: Handling Missing Value on Online Administration Universities form using Mean, Median, & KNN Method.

http://www.linkedin.com/in/muhammadalvero

Case Study

Case Study

Kamu adalah seorang Data Scientist di Kalbe Nutritionals dan sedang mendapatkan project baru dari tim inventory dan tim marketing.

Dari tim inventory, kamu diminta untuk dapat membantu memprediksi jumlah penjualan (quantity) dari total keseluruhan product Kalbe.

- Tujuan dari project ini adalah untuk mengetahui perkiraan quantity product yang terjual sehingga tim inventory dapat membuat stock persediaan harian yang cukup.
- Prediksi yang dilakukan harus harian.

Dari tim marketing kamu diminta untuk membuat cluster/segment customer berdasarkan beberapa kriteria.

- Tujuan dari project ini adalah untuk membuat segment customer.
- Segment customer ini nantinya akan digunakan oleh tim marketing untuk memberikan personalized promotion dan sales treatment.

Tools yang akan kamu gunakan dalam project ini adalah

- Python
- Jupyter Notebook
- Tableau
- Dbeaver
- PostgreSQL

Challenge: Query Database

Rakamin Academy

Data Ingestion ke dalam dbeaver

- Pilih connect to database di pojok kiri dan pilih postgreSQL
- Test connection ke postgreSQL
- Di menu drop down postgreSQL pilih table dan klik kanan (import data)
- Pilih CSV file dan upload 4 file dari Kalbehttps://drive.google.com/drive/folders/1_rQrauVW2OvLle2zd54Vcwnr2EY-vnnR?usp=sharing

Exploratory Data Analysis ke dalam dbeaver

- query 1 : Berapa rata-rata umur customer jika dilihat dari marital statusnya ?
- query 2 : Berapa rata-rata umur customer jika dilihat dari gender nya ?
- query 3 : Tentukan nama store dengan total quantity terbanyak!
- query 4 : Tentukan nama produk terlaris dengan total amount terbanyak!

Challenge: Tableau Public

Rakamin Academy

Data Ingestion ke dalam Tableau Public

- Pilih connect to database di pojok kiri dan pilih postgreSQL
- Test connection ke postgreSQL
- Di menu drop down postgreSQL pilih table dan klik kanan (import data)
- Pilih CSV file dan upload 4 file dari Kalbehttps://drive.google.com/drive/folders/1_rQrauVW2OvLle2zd54Vcwnr2EY-vnnR?usp=sharing

Membuat Dashboard kedalam tableau public

- Worksheet 1 Jumlah qty dari bulan ke bulan
- Worksheet 2 Jumlah total amount dari hari ke Hari
- Worksheet 3 Jumlah penjualan (qty) by Product
- Worksheet 4 Jumlah penjualan (total amount) by store name
- Setelah itu bisa membuat dashboard dengan menggabungkan 4 worksheet.

Challenge: Regression Model for Time Series Forecasting

Model ini diharapkan dapat membantu memprediksi jumlah penjualan (quantity) yang terjual sehingga tim inventory dapat membuat stock persediaan harian yang cukup dari total keseluruhan product Kalbe. Prediksi yang dilakukan adalah harian.

Challenge:

- Membaca data CSV
- Melakukan Data Cleansing
- Menggabungkan semua data menjadi 1 data
- Membuat model machine learning regression (time series)

Challenge: Clustering Segment Customer

Segment customer ini nantinya akan digunakan oleh tim marketing untuk memberikan personalized promotion dan sales treatment.

Challenge:

- Membaca data CSV
- Melakukan Data Cleansing
- Menggabungkan semua data menjadi 1 data
- Membuat model machine learning clustering

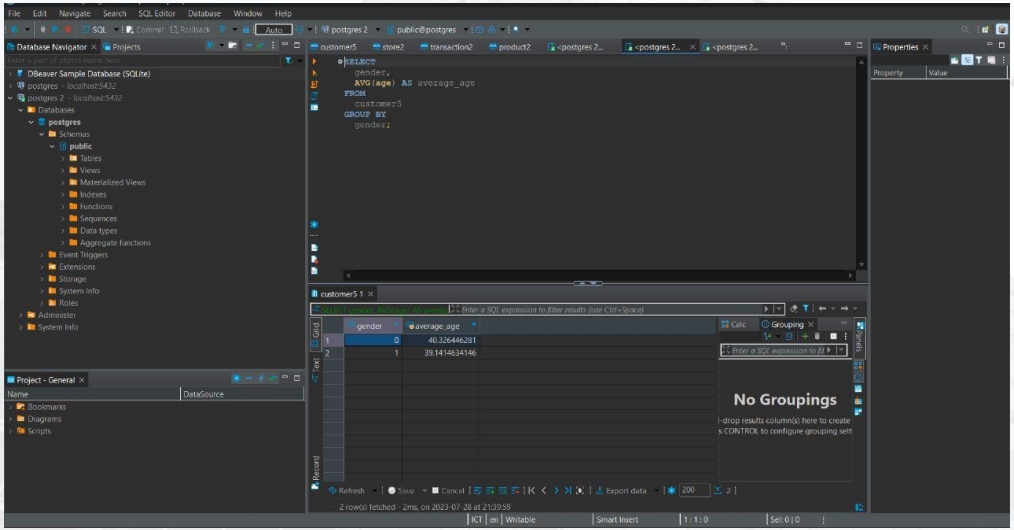
Result of the Challenge
&
Completion of the Case Study

Link Folder:
https://drive.google.com/drive/folders/1ezwrfXFsw_twOqWzjI7LVcGWVrHoHsMl?usp=sharing

Rakamin Academy X KALBE Nutritionals

Data ingestion & Exploratory Data Analysis using DBeaver

Connect Database dari Postgresql ke DBeaver, setelah itu import ke 4 data csv dari kalbe:



Data ingestion & Exploratory Data Analysis using DBeaver

Rakamin Academy

Query 1: Berapa rata-rata umur customer jika dilihat dari marital statusnya?

Code:

customer5 | store2 | transaction2 | product2 | <postgres 2> Script-Query1.sql

```
SELECT
    maritalstatus,
    AVG(age) AS average_age
FROM
    customer5
GROUP BY
    maritalstatus;
```

Output:

customer5 1

SELECT maritalstatus, AVG(age) AS average_age FROM customer5

Grid | Text

	maritalstatus	average_age
1	Married	43.008797654
2	Single	29.4056603774

Query 2: Berapa rata-rata umur customer jika dilihat dari gendernya?

Code:

customer5 | store2 | transaction2 | product2 | <postgres 2> Script-Query2.sql

```
SELECT
    gender,
    AVG(age) AS average_age
FROM
    customer5
GROUP BY
    gender;
```

Output:

customer5 1

SELECT gender, AVG(age) AS average_age FROM customer5 GROU

Grid | Text

	gender	average_age
1	0	40.326446281
2	1	39.1414634146

*Catatan:

0 = wanita

1 = pria

Data ingestion & Exploratory Data Analysis using DBeaver

Rakamin Academy

Query 3: Tentukan nama store dengan total quantity terbanyak!

Code:

customer5 | store2 | transaction2 | product2 | <postgres 2> Script-Query3.sql

```
select
  SUM(t.qty) AS total_quantity,
  s.storename AS store_name
FROM
  transaction2 t
JOIN
  store2 s ON t.storeid = s.storeid
GROUP BY
  s.storename
ORDER BY
  total_quantity DESC
LIMIT 1;
```

Output:

store2 1

```
select SUM(t.qty) AS total_quantity, s.storename AS store_name FR
```

Grid

	total_quantity	store_name
1	2,777	Lingga

Query 4: Tentukan nama produk terlaris dengan total amount terbanyak!

Code:

customer5 | store2 | transaction2 | product2 | <postgres 2> Script-Query4.sql

```
select
  SUM(t.totalamount) AS total_amount,
  s.productname AS product_name
FROM
  transaction2 t
JOIN
  product2 s ON t.productid = s.productid
GROUP BY
  s.productname
ORDER BY
  total_amount DESC
LIMIT 1;
```

Output:

product2 1

```
select SUM(t.totalamount) AS total_amount, s.productname AS pro
```

Grid

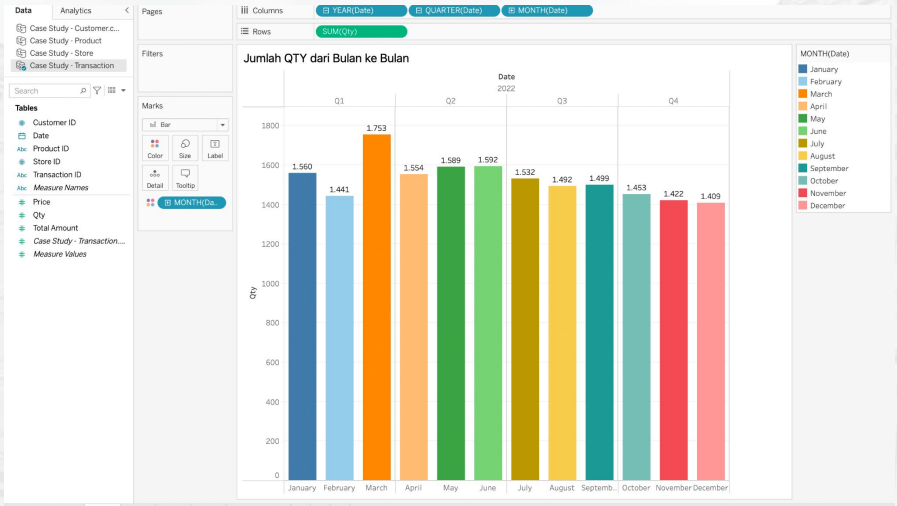
	total_amount	product_name
1	27,615,000	Cheese Stick

Data ingestion & Making Dashboard using Tableau Public

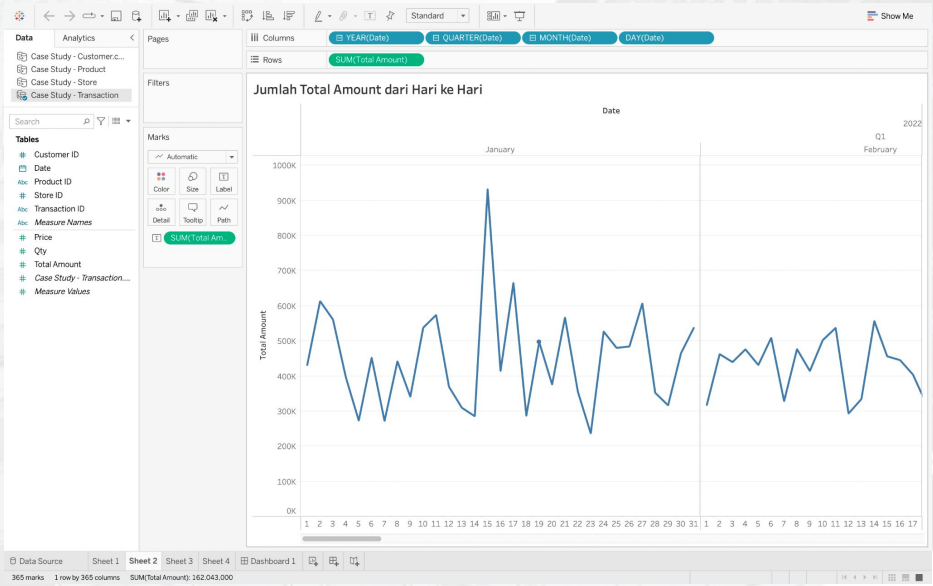
Rakamin Academy

Link Dashboard Tableau:
https://public.tableau.com/views/DashboardRakaminAcademy/Dashboard1?:language=en-US&:display_count=n&:origin=viz_share_link

Worksheet 1: Jumlah qty dari bulan ke bulan

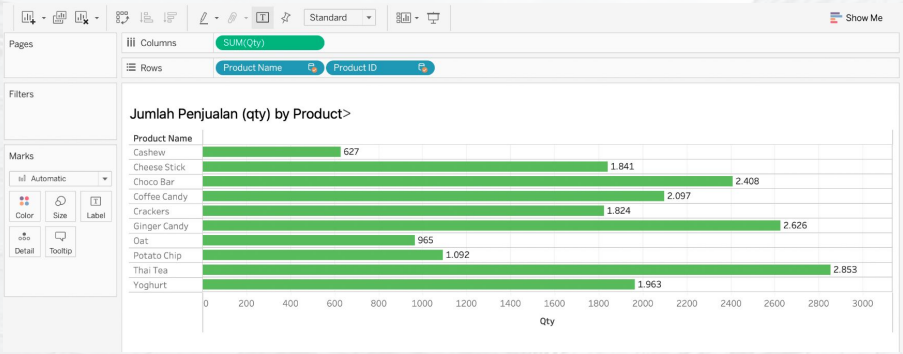


Worksheet 2: Jumlah total amount dari hari ke hari

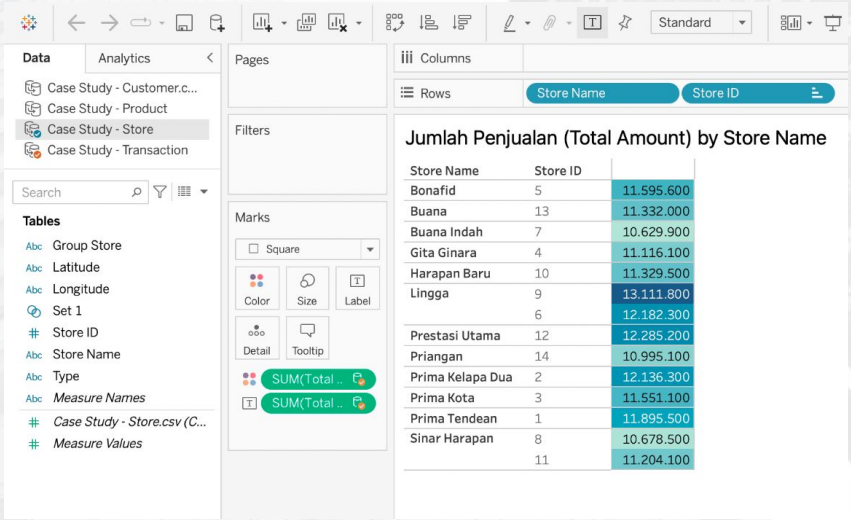


Data ingestion & Making Dashboard using Tableau Public

Worksheet 3: Jumlah penjualan (qty) by product



Worksheet 4: Jumlah penjualan (total amount) by store name



Dashboard using Tableau Public

Link Dashboard Tableau:
https://public.tableau.com/views/DashboardRakaminAcademy/Dashboard1?:language=en-US&:display_count=n&:origin=viz_share_link

Rakamin Academy

end-to-end Machine Learning Regression & Clustering Model

Link Google Colab:
https://colab.research.google.com/drive/16p_aIkvwqEIRuyBBpKL0QA5_sFPF8DA7?usp=sharing

Read data CSV

```
[ ]  import pandas as pd

     customer = pd.read_csv('Case Study - Customer.csv', sep=';')
     product = pd.read_csv('Case Study - Product.csv', sep=';')
     store = pd.read_csv('Case Study - Store.csv', sep=';')
     transaction = pd.read_csv('Case Study - Transaction.csv', sep=';')
```

[ ] transaction

	TransactionID	CustomerID	Date	ProductID	Price	Qty	TotalAmount	StoreID
0	TR11369	328	01/01/2022	P3	7500	4	30000	12
1	TR16356	165	01/01/2022	P9	10000	7	70000	1
2	TR1984	183	01/01/2022	P1	8800	4	35200	4
3	TR35256	160	01/01/2022	P1	8800	7	61600	4
4	TR41231	386	01/01/2022	P9	10000	1	10000	4
...	...	...	...	...	...	...	...	...
5015	TR54423	243	31/12/2022	P10	15000	5	75000	3
5016	TR5604	271	31/12/2022	P2	3200	4	12800	9
5017	TR81224	52	31/12/2022	P7	9400	6	56400	9
5018	TR85016	18	31/12/2022	P8	16000	3	48000	13
5019	TR85684	55	31/12/2022	P8	16000	1	16000	6

5020 rows × 8 columns

[ ] customer

	CustomerID	Age	Gender	Marital Status	Income
0	1	55	1	Married	5,12
1	2	60	1	Married	6,23
2	3	32	1	Married	9,17
3	4	31	1	Married	4,87
4	5	58	1	Married	3,57
...	...	...	...	...	...
442	443	33	1	NaN	9,28
443	444	53	0	Married	15,31
444	445	51	0	Married	14,48
445	446	57	0	Married	7,81
446	447	54	1	Married	20,37

447 rows × 5 columns

[ ] product

	ProductID	Product Name	Price
0	P1	Choco Bar	8800
1	P2	Ginger Candy	3200
2	P3	Crackers	7500
3	P4	Potato Chip	12000
4	P5	Thai Tea	4200
5	P6	Cashew	18000
6	P7	Coffee Candy	9400
7	P8	Oat	16000
8	P9	Yoghurt	10000
9	P10	Cheese Stick	15000

store

	StoreID	StoreName	GroupStore	Type	Latitude	Longitude
0	1	Prima Tendean	Prima	Modern Trade	-6,2	106,816666
1	2	Prima Kelapa Dua	Prima	Modern Trade	-6,914864	107,608238
2	3	Prima Kota	Prima	Modern Trade	-7,797068	110,370529
3	4	Gita Ginara	Gita	General Trade	-6,966667	110,416664
4	5	Bonafid	Gita	General Trade	-7,250445	112,768845
5	6	Lingga	Lingga	Modern Trade	-5,135399	119,42379
6	7	Buana Indah	Buana	General Trade	3,316694	114,590111
7	8	Sinar Harapan	Harapan Baru	General Trade	5,54829	95,323753
8	9	Lingga	Lingga	Modern Trade	-3,654703	128,190643
9	10	Harapan Baru	Harapan Baru	General Trade	3,597031	98,678513
10	11	Sinar Harapan	Prestasi	General Trade	0,533505	101,447403
11	12	Prestasi Utama	Prestasi	General Trade	-2,990934	104,756554
12	13	Buana	Buana	General Trade	-1,26916	116,825264
13	14	Priangan	Priangan	Modern Trade	-5,45	105,26667

Data Cleansing - Customer

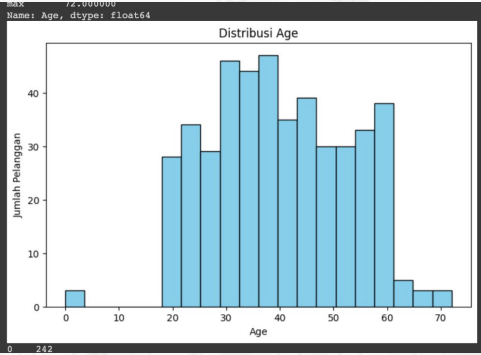
Data Profiling:

```
[ ] customer.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 447 entries, 0 to 446
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CustomerID      447 non-null    int64
 1   Age             447 non-null    int64
 2   Gender          447 non-null    int64
 3   Marital Status  444 non-null    object
 4   Income          447 non-null    object
dtypes: int64(3), object(2)
memory usage: 17.6+ KB
```

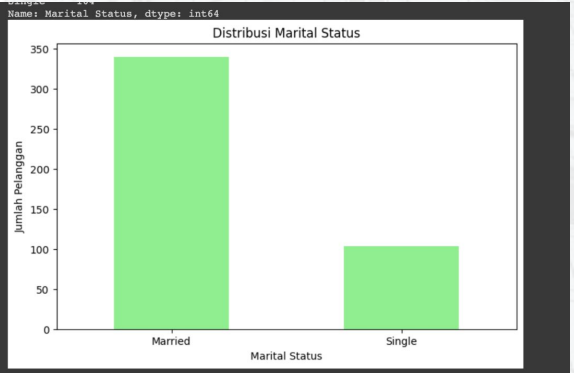
```
customer.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 447 entries, 0 to 446
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CustomerID      447 non-null    int64
 1   Age             447 non-null    int64
 2   Gender          447 non-null    int64
 3   Marital Status  444 non-null    object
 4   Income          447 non-null    object
dtypes: int64(3), object(2)
memory usage: 17.6+ KB

[ ] customer.shape

(447, 5)

[ ] customer.describe()
```

	CustomerID	Age	Gender
count	447.000000	447.000000	447.000000
mean	224.000000	39.782998	0.458613
std	129.182042	12.848719	0.498842
min	1.000000	0.000000	0.000000
25%	112.500000	30.000000	0.000000
50%	224.000000	39.000000	0.000000
75%	335.500000	50.500000	1.000000
max	447.000000	72.000000	1.000000

```
max      72.000000
Name: Age, dtype: float64
```

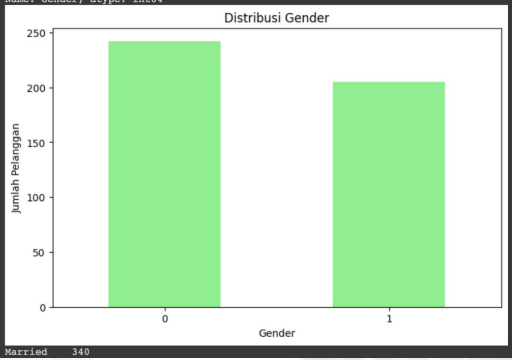



```
0   242
Name: Marital Status, dtype: int64
```

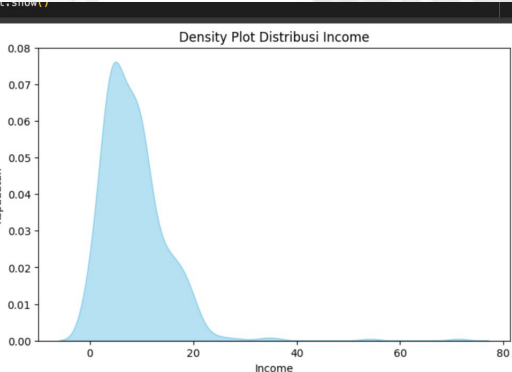


EDA:

```
1    205
Name: Gender, dtype: int64
```




```
Married    340
```



Data Cleansing - Customer

Data Imputation:

Data Imputation - Customer

```
[ ] customer['Marital Status'] = customer['Marital Status'].fillna(customer['Marital Status'].mode()[0])

[ ] customer.isna().sum()

CustomerID        0
Age               0
Gender            0
Marital Status    0
Income            0
dtype: int64
```

Data Transformation:

```
[ ] customer.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 447 entries, 0 to 446
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CustomerID      447 non-null    int64
 1   Age             447 non-null    int64
 2   Gender          447 non-null    int64
 3   Marital Status  447 non-null    object
 4   Income          447 non-null    object
dtypes: int64(3), object(2)
memory usage: 17.6+ KB

[ ] customer['Income'] = customer['Income'].str.replace(',', '.').astype(float)
    customer['Gender'] = customer['Gender'].astype('category')
    customer['Marital Status'] = customer['Marital Status'].astype('category')

    print(customer.head())

   CustomerID  Age Gender Marital Status  Income
0           1   55      1        Married    5.12
1           2   60      1        Married    6.23
2           3   32      1        Married    9.17
3           4   31      1        Married    4.87
4           5   58      1        Married    3.57
```

Data Cleansing - Product

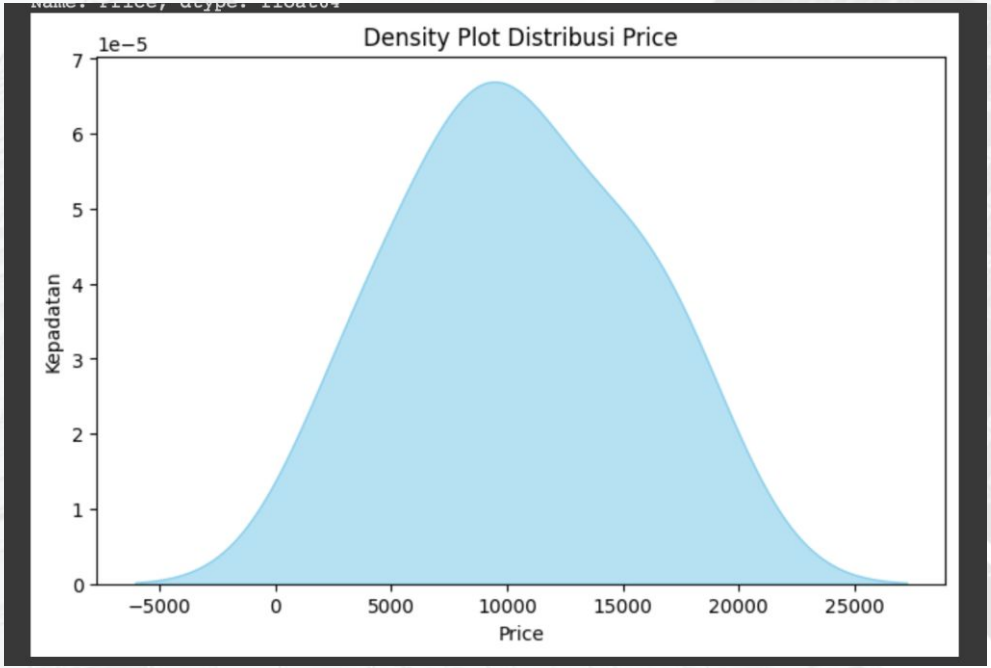
Rakamin Academy

Data Profiling:

```
[ ] product.columns

    Index(['ProductID', 'Product Name', 'Price'], dtype='object')

[ ] product.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 10 entries, 0 to 9
    Data columns (total 3 columns):
     #   Column        Non-Null Count  Dtype
    ---  ------        --------------  -----
     0   ProductID     10 non-null     object
     1   Product Name  10 non-null     object
     2   Price         10 non-null     int64
    dtypes: int64(1), object(2)
    memory usage: 368.0+ bytes

[ ] product.shape

    (10, 3)
```

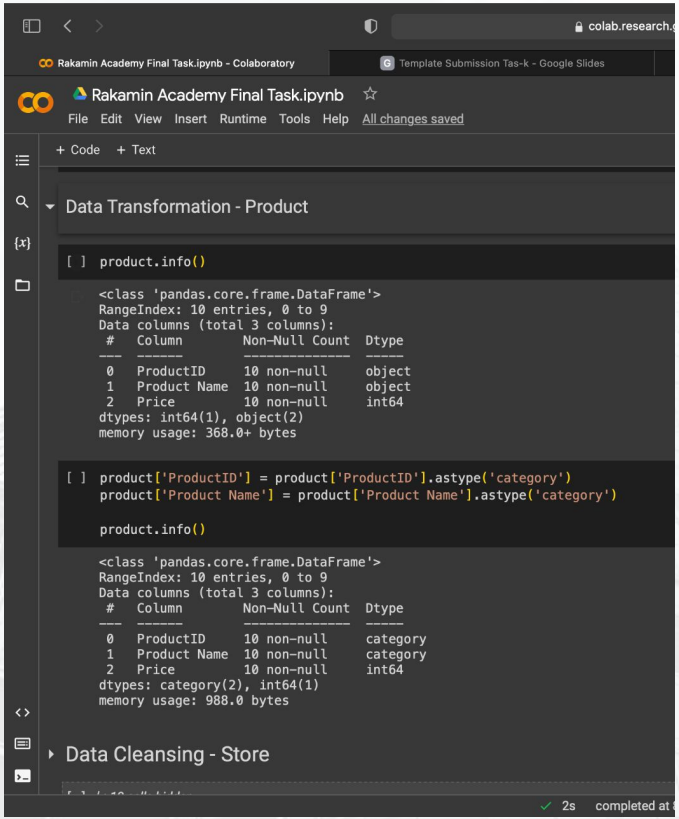
```
[ ] product.isna().sum()

    ProductID       0
    Product Name    0
    Price           0
    dtype: int64

[ ] product.duplicated().sum()

    0
```

EDA:



Data Cleansing - Product

Data Transformation:



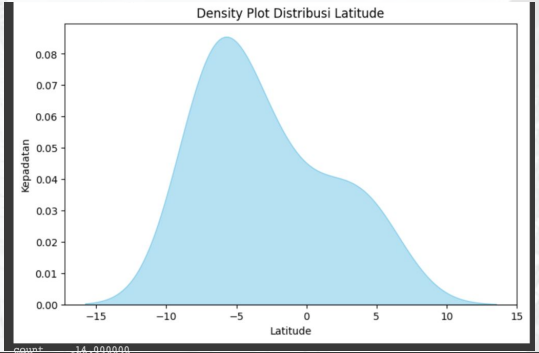
Data Cleansing - Store

Data Profiling:

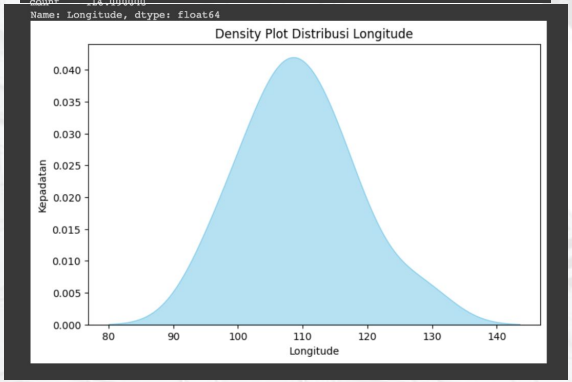
```
[ ] store.columns

    Index(['StoreID', 'StoreName', 'GroupStore', 'Type', 'Latitude', 'Longitude'], dtype='object')

[ ] store.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 14 entries, 0 to 13
    Data columns (total 6 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   StoreID     14 non-null     int64
     1   StoreName   14 non-null     object
     2   GroupStore  14 non-null     object
     3   Type        14 non-null     object
     4   Latitude    14 non-null     object
     5   Longitude   14 non-null     object
    dtypes: int64(1), object(5)
    memory usage: 800.0+ bytes

[ ] store.shape

    (14, 6)
```

```
[ ] store.isna().sum()

    StoreID       0
    StoreName     0
    GroupStore    0
    Type          0
    Latitude      0
    Longitude     0
    dtype: int64

[ ] store.duplicated().sum()

    0
```

EDA:




```
count    14.000000
Name: Longitude, dtype: float64
```



Data Cleansing - Store

Rakamin Academy

Data Transformation:

```
[ ]  store['Latitude'] = store['Latitude'].str.replace(',', '.').astype(float)
     store['Longitude'] = store['Longitude'].str.replace(',', '.').astype(float)

     store['Latitude'] = store['Latitude'].astype(float)
     store['Longitude'] = store['Longitude'].astype(float)
     store['Type'] = store['Type'].astype('category')
     store['GroupStore'] = store['GroupStore'].astype('category')
     store['StoreName'] = store['StoreName'].astype('category')

     store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   StoreID     14 non-null     int64
 1   StoreName   14 non-null     category
 2   GroupStore  14 non-null     category
 3   Type        14 non-null     category
 4   Latitude    14 non-null     float64
 5   Longitude   14 non-null     float64
dtypes: category(3), float64(2), int64(1)
memory usage: 1.3 KB
```

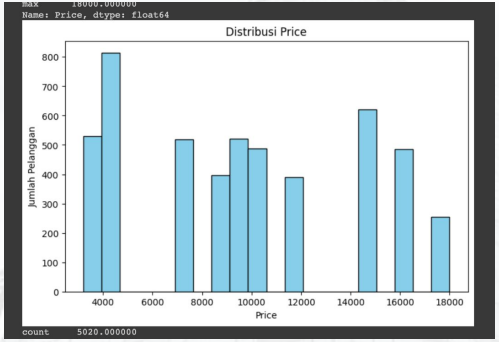
Data Cleansing - Transaction

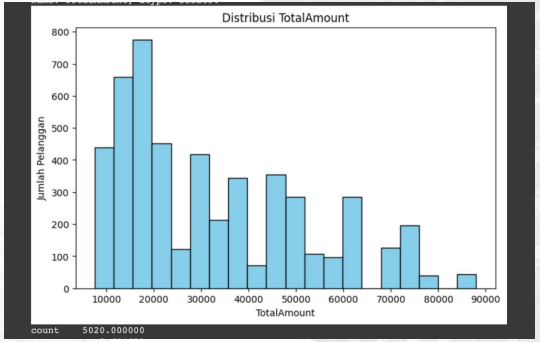
Data Profiling:

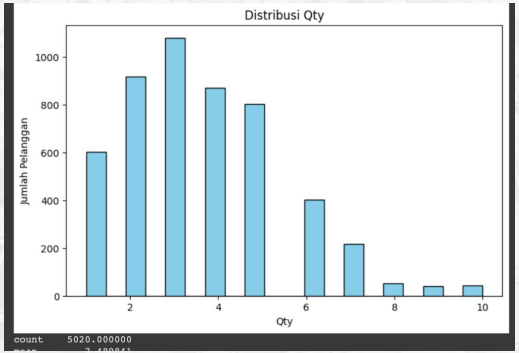
```
[ ] transaction.columns

Index(['TransactionID', 'CustomerID', 'Date', 'ProductID', 'Price', 'Qty',
       'TotalAmount', 'StoreID'],
      dtype='object')

transaction.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5020 entries, 0 to 5019
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   TransactionID  5020 non-null   object
 1   CustomerID     5020 non-null   int64
 2   Date           5020 non-null   object
 3   ProductID      5020 non-null   object
 4   Price          5020 non-null   int64
 5   Qty            5020 non-null   int64
 6   TotalAmount    5020 non-null   int64
 7   StoreID        5020 non-null   int64
dtypes: int64(5), object(3)
memory usage: 313.9+ KB

[ ] transaction.shape

(5020, 8)
```

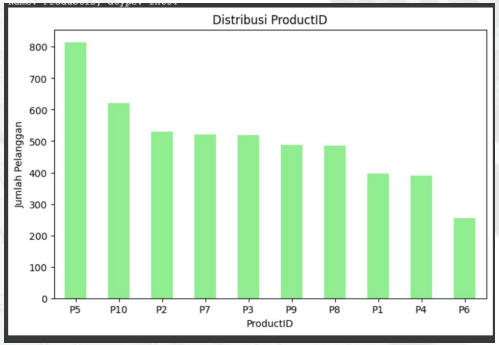
```
[ ] transaction.isna().sum()

TransactionID    0
CustomerID       0
Date             0
ProductID        0
Price            0
Qty              0
TotalAmount      0
StoreID          0
dtype: int64

[ ] transaction.duplicated().sum()

0
```

EDA:









Data Cleansing - Transaction

Rakamin Academy

Data Transformation:

```
[ ]  transaction['ProductID'] = transaction['ProductID'].astype('category')
     transaction['Date'] = transaction['Date'].astype('datetime64')

     transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5020 entries, 0 to 5019
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   TransactionID  5020 non-null   object
 1   CustomerID     5020 non-null   int64
 2   Date           5020 non-null   datetime64[ns]
 3   ProductID      5020 non-null   category
 4   Price          5020 non-null   int64
 5   Qty            5020 non-null   int64
 6   TotalAmount    5020 non-null   int64
 7   StoreID        5020 non-null   int64
dtypes: category(1), datetime64[ns](1), int64(5), object(1)
memory usage: 279.9+ KB
```

Merge All Data

Setelah dilakukan data observation dan data imputasi serta data transformasi agar semua tipe data sama antara dataset 1 dengan lainnya, akan dilakukan merge dari ke 4 data.

```
[ ]  # Gabungkan dataset transaction dengan dataset customer berdasarkan 'CustomerID'
     merged_df = pd.merge(transaction, customer, on='CustomerID', how='left')

     # Gabungkan dataset merged_df dengan dataset product berdasarkan 'ProductID'
     merged_df = pd.merge(merged_df, product, on='ProductID', how='left')

     # Gabungkan dataset merged_df dengan dataset store berdasarkan 'StoreID'
     merged_df = pd.merge(merged_df, store, on='StoreID', how='left')

     # Tampilkan DataFrame hasil penggabungan
     print(merged_df.head())
```

```
  TransactionID  CustomerID       Date ProductID  Price_x  Qty  TotalAmount  \
0       TR11369         328 2022-01-01        P3     7500    4        30000
1       TR16356         165 2022-01-01        P9    10000    7        70000
2        TR1984         183 2022-01-01        P1     8800    4        35200
3       TR35256         160 2022-01-01        P1     8800    7        61600
4       TR41231         386 2022-01-01        P9    10000    1        10000

   StoreID  Age Gender Marital Status  Income Product Name  Price_y  \
0       12   36      0        Married   10.53     Crackers     7500
1        1   44      1        Married   14.58      Yoghurt    10000
2        4   27      1         Single    0.18    Choco Bar     8800
3        4   48      1        Married   12.57    Choco Bar     8800
4        4   33      0        Married    6.95      Yoghurt    10000

         StoreName GroupStore          Type  Latitude   Longitude
0  Prestasi Utama   Prestasi  General Trade -2.990934  104.756554
1   Prima Tendean      Prima   Modern Trade -6.200000  106.816666
2     Gita Ginara       Gita  General Trade -6.966667  110.416664
3     Gita Ginara       Gita  General Trade -6.966667  110.416664
4     Gita Ginara       Gita  General Trade -6.966667  110.416664
```

Modelling: ARIMA Regression Model for Time Series Forecasting

Membuat data baru untuk Regresi:

```
[ ]  # Membuat data baru untuk regression
     data_reg = merged_df.groupby('Date')['Qty'].sum().reset_index()

[ ]  data_reg.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 365 entries, 0 to 364
     Data columns (total 2 columns):
      #   Column  Non-Null Count  Dtype
     ---  ------  --------------  -----
      0   Date    365 non-null    datetime64[ns]
      1   Qty     365 non-null    int64
     dtypes: datetime64[ns](1), int64(1)
     memory usage: 5.8 KB
```

Knowing ARIMA Regression Model

Rakamin Academy

ARIMA, yang merupakan kependekan dari Model AutoRegressive Integrated Moving Average, adalah sebuah algoritma peramalan yang berdasarkan gagasan bahwa informasi dari nilai-nilai masa lalu dari rangkaian waktu bisa digunakan sendiri untuk memprediksi nilai-nilai di masa depan.

Model ARIMA digunakan untuk meramalkan nilai-nilai masa depan dari suatu rangkaian waktu yang "non-musiman" dan menunjukkan pola-pola tertentu, bukan kebisingan putih acak. Model ARIMA dapat dijelaskan dengan tiga parameter urutan: (p, d, q),

di mana,

p adalah urutan dari istilah AR (AutoRegressive)

q adalah urutan dari istilah MA (Moving Average)

d adalah jumlah pendiferensian yang diperlukan untuk membuat rangkaian waktu menjadi stationer.

Knowing ARIMA Regression Model

AR(p) Autoregression - sebuah model regresi yang memanfaatkan hubungan dependen antara observasi saat ini dan observasi pada periode sebelumnya. Komponen auto regresi (AR(p)) mengacu pada penggunaan nilai-nilai masa lalu dalam persamaan regresi untuk rangkaian waktu.

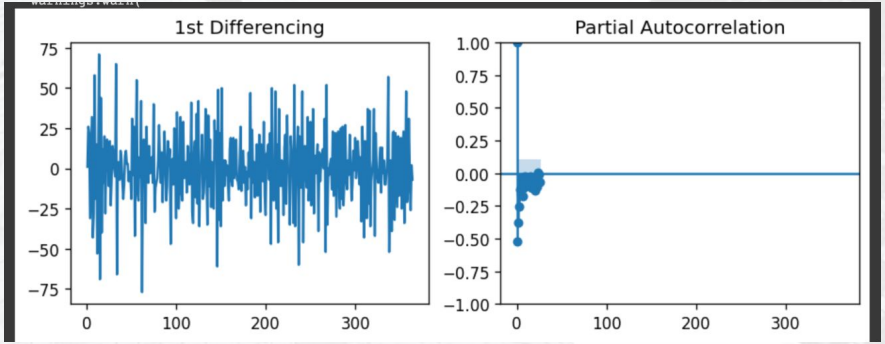
I(d) Integrasi - menggunakan pendiferensian observasi (mengurangkan observasi dari observasi pada langkah waktu sebelumnya) untuk membuat rangkaian waktu menjadi stationer. Pendiferensian melibatkan pengurangan nilai-nilai saat ini dari suatu rangkaian dengan nilai-nilai sebelumnya sebanyak d kali.

MA(q) Moving Average - sebuah model yang menggunakan ketergantungan antara observasi dan kesalahan residual dari model rata-rata bergerak yang diterapkan pada observasi tertinggal. Komponen rata-rata bergerak menggambarkan kesalahan dari model sebagai kombinasi dari kesalahan sebelumnya. Urutan q mewakili jumlah istilah yang akan dimasukkan dalam model.

Modelling: ARIMA Regression Model for Time Series Forecasting

Menentukan nilai parameter (p,d,q) untuk model regresi ARIMA untuk Time Series Forecasting:

```
[ ]  #MENENTUKAN PARAMETER P
     from statsmodels.graphics.tsaplots import plot_pacf
     import matplotlib.pyplot as plt

     # 1st Differencing
     data_reg['Qty_diff_1'] = data_reg['Qty'].diff()

     # 2nd Differencing
     data_reg['Qty_diff_2'] = data_reg['Qty_diff_1'].diff()

     # 3rd Differencing
     data_reg['Qty_diff_3'] = data_reg['Qty_diff_2'].diff()

     # Plot PACF for 1st Differencing
     plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})
     fig, axes = plt.subplots(1, 2, sharex=True)
     axes[0].plot(data_reg['Qty_diff_1'].dropna())
     axes[0].set_title('1st Differencing')
     axes[1].set(ylim=(0, 1.2))
     plot_pacf(data_reg['Qty_diff_1'].dropna(), ax=axes[1])
     plt.show()
```




```
[ ]  #MENENTUKAN PARAMETER D
     from statsmodels.tsa.stattools import adfuller
     from numpy import log
     result = adfuller(data_reg.Qty.dropna())
     print('ADF Statistic: %f' % result[0])
     print('p-value: %f' % result[1])

     ADF Statistic: -19.018783
     p-value: 0.000000

[ ]  # dari hasil p-value yang didapatkan, order D yang sekiranya cukup baik adalah = 0
```

Modelling: ARIMA Regression Model for Time Series Forecasting

Rakamin Academy

Digunakan order: (2,0,3), lalu dilakukan modelling dan visualisasi hasil dari model dengan beberapa plot.

```
[ ] from statsmodels.tsa.arima.model import ARIMA

    # 2,0,3 ARIMA Model
    model = ARIMA(data_reg['Qty'], order=(2, 0, 3))
    model_fit = model.fit()
    print(model_fit.summary())
```

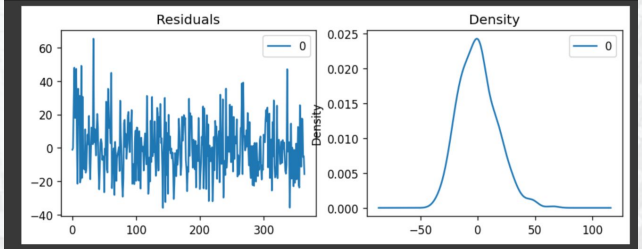
SARIMAX Results

Dep. Variable:	Qty	No. Observations:	365
Model:	ARIMA(2, 0, 3)	Log Likelihood	-1541.238
Date:	Fri, 28 Jul 2023	AIC	3096.475
Time:	12:59:23	BIC	3123.775
Sample:	0	HQIC	3107.324
	- 365		
Covariance Type:	opg		

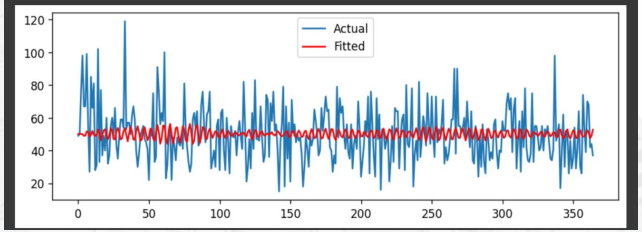
	coef	std err	z	P>|z|	[0.025	0.975]
const	50.0937	0.922	54.359	0.000	48.288	51.900
ar.L1	0.4140	0.025	16.570	0.000	0.365	0.463
ar.L2	-0.9690	0.025	-38.553	0.000	-1.018	-0.920
ma.L1	-0.4254	0.058	-7.274	0.000	-0.540	-0.311
ma.L2	0.9672	0.042	22.935	0.000	0.885	1.050
ma.L3	0.0321	0.057	0.567	0.571	-0.079	0.143
sigma2	268.0660	20.243	13.242	0.000	228.390	307.742

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	19.76
Prob(Q):	0.95	Prob(JB):	0.00
Heteroskedasticity (H):	0.68	Skew:	0.53
Prob(H) (two-sided):	0.04	Kurtosis:	3.40

```
# Plot residual errors
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()
```




```
# Prediksi dengan model ARIMA
pred = model_fit.predict()

# Plot Actual vs Fitted
plt.plot(data_reg.index, data_reg['Qty'], label='Actual')
plt.plot(data_reg.index, pred, label='Fitted', color='red')
plt.legend()
plt.show()
```



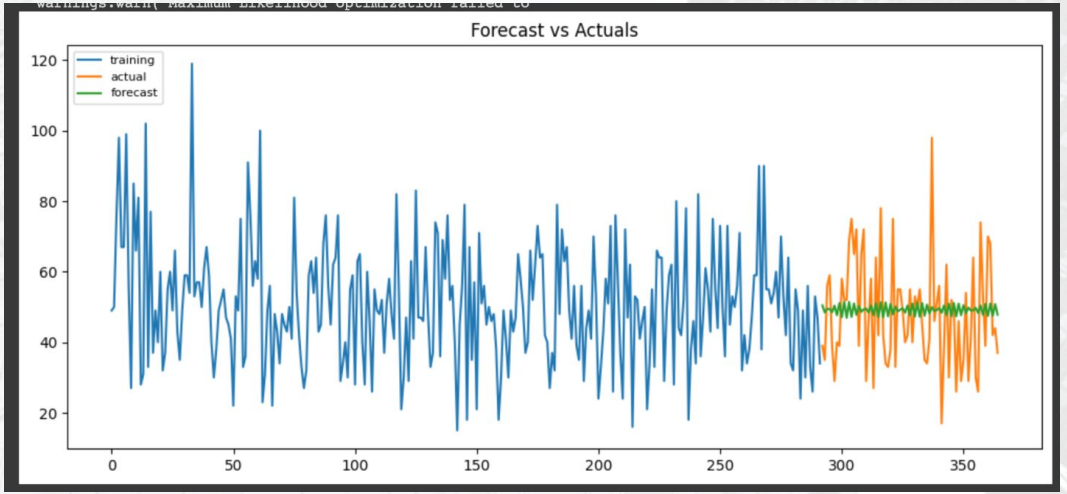
Modelling: ARIMA Regression Model for Time Series Forecasting

Rakamin Academy

Out-of-Time Cross validation Untuk Model ARIMA yang Optimal

```
[ ]  from statsmodels.tsa.stattools import acf

     # Create Training and Test
     train = data_reg.Qty[:292]
     test = data_reg.Qty[292:]
```

```
[ ]  from statsmodels.tsa.arima.model import ARIMA

     model = ARIMA(train, order=(2, 1, 3))
     fitted = model.fit()

     # Forecast
     forecast_result = fitted.get_forecast(steps=73, alpha=0.35)

     # Get forecast values
     fc = forecast_result.predicted_mean
     conf = forecast_result.conf_int()

     # Make as pandas series
     fc_series = pd.Series(fc, index=test.index)
     conf_df = pd.DataFrame(conf, index=test.index, columns=['lower', 'upper'])
     lower_series = conf_df['lower']
     upper_series = conf_df['upper']

     # Plot
     plt.figure(figsize=(12, 5), dpi=100)
     plt.plot(train, label='training')
     plt.plot(test, label='actual')
     plt.plot(fc_series, label='forecast')
     plt.fill_between(lower_series.index, lower_series, upper_series,
                      color='k', alpha=.15)
     plt.title('Forecast vs Actuals')
     plt.legend(loc='upper left', fontsize=8)
     plt.show()
```



Modelling: ARIMA Regression Model for Time Series Forecasting

Rakamin Academy

Metriks akurasi dari Time Series Forecasting:

```
import numpy as np

def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    acf1 = acf(fc-test)[1]                      # ACF1
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse, 'acf1':acf1,
            'corr':corr, 'minmax':minmax})

forecast_accuracy(fc, test.values)
```

```
<ipython-input-74-fa5a2e2a7bd7>:10: FutureWarning: Support for multi-di
  mins = np.amin(np.hstack([forecast[:,None],
<ipython-input-74-fa5a2e2a7bd7>:12: FutureWarning: Support for multi-di
  maxs = np.amax(np.hstack([forecast[:,None],
{'mape': 0.3067274123996467,
 'me': 1.1248567597534553,
 'mae': 12.807577478617448,
 'mpe': 0.1357186518917499,
 'rmse': 15.360688428728313,
 'acf1': -0.03157770230817815,
 'corr': 0.032026013114150785,
 'minmax': 0.2267894837231158}
```

Dari metriks akurasi yang didapatkan, Mean Absolute Percentage Error (MAPE) sebesar 0.3067 menunjukkan bahwa rata-rata persentase kesalahan prediksi terhadap nilai aktual adalah sekitar 30.67%, Mean Absolute Error (MAE) sebesar 12.8076 menunjukkan bahwa rata-rata kesalahan prediksi dalam satuan kuantitas penjualan adalah sekitar 12.81 atau sekitar 13 unit.

Rata-rata dari kuantitas yang telah diprediksi / forecasting perhari nya untuk tim inventory kalbe:

```
[ ]  average_fc = np.mean(fc)
     print(average_fc)

     49.22074717071237
```

Modelling: Clustering Model using KMeans Method

Membuat data baru untuk Clustering:

Membuat Dataset Baru

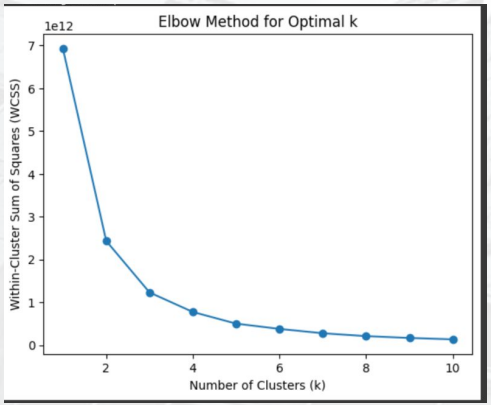
```
[ ]  import pandas as pd

     # Langkah 1: Membuat dataset baru untuk clustering
     clustering_df = merged_df[['CustomerID', 'TransactionID', 'Qty', 'TotalAmount']].copy()

     # Langkah 2: Mengelompokkan data berdasarkan customerID menggunakan groupby
     grouped_df = clustering_df.groupby('CustomerID')

     # Langkah 3: Melakukan agregasi terhadap fitur-fitur 'TransactionID', 'Qty', dan 'TotalAmount'
     agg_df = grouped_df.agg({
         'TransactionID': 'count',
         'Qty': 'sum',
         'TotalAmount': 'sum'
     })

     # Mengganti nama kolom hasil agregasi
     agg_df.columns = ['TransactionCount', 'TotalQty', 'TotalAmount']

     # Langkah 4: Siapkan data untuk clustering
     # Di sini kita menggunakan agg_df sebagai data yang akan di-cluster
     data_clus = agg_df.values
```

Modelling: Clustering Model using KMeans Method

Elbow Method untuk menentukan jumlah kluster:

K-Means Clustering

```
[ ] from sklearn.cluster import KMeans
    # Inisialisasi daftar WCSS (Within-Cluster Sum of Squares)
    wcss = []

    # Range jumlah kluster yang akan diuji (misalnya dari 1 hingga 10)
    k_range = range(1, 11)

    # Melakukan elbow method untuk mencari jumlah kluster yang optimal
    for k in k_range:
        kmeans_model = KMeans(n_clusters=k, init='k-means++', random_state=42)
        kmeans_model.fit(data_clus)
        wcss.append(kmeans_model.inertia_)

    # Plot grafik elbow untuk melihat elbow point
    plt.plot(k_range, wcss, marker='o')
    plt.title('Elbow Method for Optimal k')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
    plt.show()
```



KMeans Clustering:

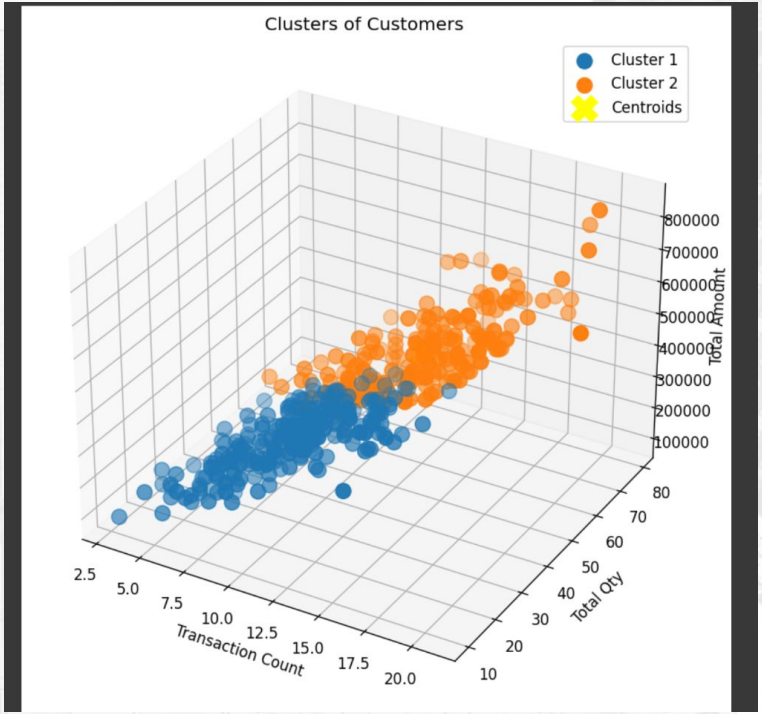
```
[ ] from sklearn.cluster import KMeans

    # Jumlah kluster yang diinginkan (2) sesuai dengan elbow method
    num_clusters = 2

    # Membuat model kmeans
    kmeans_model = KMeans(n_clusters=num_clusters, random_state=42)

    # Melakukan proses clustering
    kmeans_model.fit(data_clus)

    # Menambahkan kolom hasil clustering ke dalam dataframe
    agg_df['Cluster'] = kmeans_model.labels_

    # Melihat hasil clustering
    print(agg_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.p
  warnings.warn(
            TransactionCount  TotalQty  TotalAmount  Cluster
CustomerID
1                         17        60       623300        1
2                         13        57       392300        1
3                         15        56       446200        1
4                         10        46       302500        0
5                          7        27       268600        0
...                      ...       ...          ...      ...
443                       16        59       485100        1
444                       18        62       577700        1
445                       18        68       587200        1
446                       11        42       423300        1
447                       13        42       439300        1

[447 rows x 4 columns]
```

Modelling: Clustering Model using KMeans Method

Visualisasi Cluster dalam 3 Dimensi:

```
[ ] import matplotlib.pyplot as plt
    from mpl_toolkits.mplot3d import Axes3D

    labels = kmeans_model.labels_
    # Visualisasi 3D
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Plot data untuk setiap kluster
    for i in range(num_clusters):
        ax.scatter(data_clus[labels == i, 0], data_clus[labels == i, 1], data_clus[labels == i, 2],
                   label=f'Cluster {i+1}', s=100)

    # Plot centroid
    centroids = kmeans_model.cluster_centers_
    ax.scatter(centroids[:, 0], centroids[:, 1], centroids[:, 2], c='yellow', marker='X', s=300, label='Centroids')

    ax.set_xlabel('Transaction Count')
    ax.set_ylabel('Total Qty')
    ax.set_zlabel('Total Amount')
    ax.set_title('Clusters of Customers')
    plt.legend()
    plt.show()
```



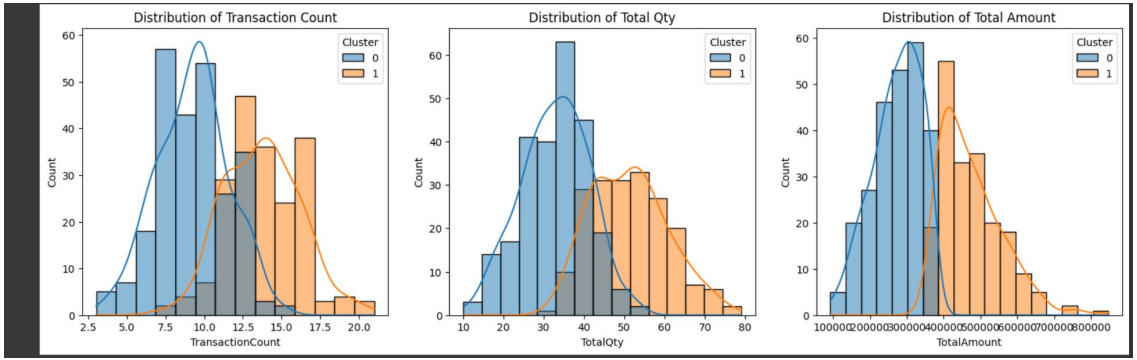
Modelling: Clustering Model using KMeans Method

Rakamin Academy

Rata-rata untuk setiap fitur berdasarkan cluster dan plot distribusi:

```
# Mencari rata-rata dari 'Transaction Count', 'Total Qty', dan 'Total Amount' untuk
cluster_means = agg_df.groupby('Cluster').mean()

# Mencetak rata-rata untuk setiap fitur berdasarkan cluster
print("Rata-rata untuk setiap fitur berdasarkan cluster:")
print(cluster_means)

# Visualisasi distribusi fitur 'Transaction Count', 'Total Qty', dan 'Total Amount'
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# Plot distribusi Transaction Count
sns.histplot(data=agg_df, x='TransactionCount', kde=True, hue='Cluster', ax=axes[0])
axes[0].set_title('Distribution of Transaction Count')

# Plot distribusi Total Qty
sns.histplot(data=agg_df, x='TotalQty', kde=True, hue='Cluster', ax=axes[1])
axes[1].set_title('Distribution of Total Qty')

# Plot distribusi Total Amount
sns.histplot(data=agg_df, x='TotalAmount', kde=True, hue='Cluster', ax=axes[2])
axes[2].set_title('Distribution of Total Amount')

plt.show()
```

```
Rata-rata untuk setiap fitur berdasarkan cluster:
         TransactionCount   TotalQty   TotalAmount
Cluster
0                9.252000  32.796000  273511.20000
1               13.741117  51.253807  475457.86802
```



Conclusion of the Case Study

Rakamin Academy

Setelah dilakukannya model regresi ARIMA dengan Time Series Forecasting, Tim Inventory Kalbe dapat memperkirakan sekiranya 49 Kuantitas produk per harinya. Namun, dapat menjadi catatan bahwa rata-rata kesalahan prediksi dalam satuan kuantitas penjualan adalah sekitar 13 unit, Bisa lebih dari hasil prediksi dan juga bisa kurang dari hasil prediksi. Berarti setidaknya, Tim Inventory Kalbe harus melebihkan 13 unit produk sebagai bentuk antisipasi.
Saran dalam model prediksi ini adalah coba untuk menggunakan metode lain yang lebih akurat sesuai dengan tipe persebaran kuantitas penjualan kalbe untuk memperbesar akurasi.

Setelah dilakukannya model clustering menggunakan KMeans Clustering, dapat dilihat bahwa terbentuk 2 cluster untuk mengoptimalkan menggunakan metode KMeans dengan elbow method. Selain itu, dapat dilihat rata-rata untuk setiap fitur berdasarkan cluster adalah sebagai berikut:

Rata-rata untuk setiap fitur berdasarkan cluster:

Cluster	TransactionCount	TotalQty	TotalAmount
0	9.252000	32.796000	273511.20000
1	13.741117	51.253807	475457.86802

Saran dalam model clustering ini adalah dapat dilakukan eksplorasi analisis data lebih lanjut, seperti bagaimana rata-rata umur dari masing-masing cluster, apa saja store yang lebih dominan dari masing-masing cluster, bagaimana gender yang lebih dominan dari masing-masing cluster, dll.

Rakamin Academy

Link Video Presentasi:

https://drive.google.com/file/d/1MB50i3q7_1aGdEFZFh_g1gu0sQ2F_RJm/view?usp=sharing

Thank You