Rakamin Academy

W3.Solutions()

Dokumen Laporan Final Project

Latar Belakang Masalah

Problem :

Sebuah perusahaan *e-commerce* berbasis internasional ingin menemukan *insight* dari data pelanggan.. Berdasarkan data dari perusahaan tersebut, terdapat kurang lebih 60% yang mengalami keterlambatan dalam penerimaan barang. Berdasarkan studi dari voxware yang melakukan survey terhadap 600 orang, sebanyak 62% pelanggan akan cenderung berkurang atau berhenti berbelanja dari retailer online jika barang yang mereka beli terlambat 2-3 hari dari tanggal yang dijanjikan. Maka dari itu pihak *e-commerce* ingin menjaga dan meningkatkan customer retention dan meningkatkan performa logistik dikarenakan banyak dari pelanggan yang melakukan komplain mengenai ketepatan waktu pengiriman. (https://www.supplychainbrain.com/articles/14912-impact-of-late-or-inaccurate-deliveries-can-be-disastrous-study-shows)

Role :

Sebagai konsultan data scientist untuk perusahaan *e-commerce*, kami diminta memprediksi apakah penerimaan tersebut tepat waktu atau tidak berdasarkan data yang tersedia dan kami diminta untuk menganalisis faktor-faktor yang mempengaruhi ketepatan waktu penerimaan serta memberikan insight dan rekomendasi berdasarkan hasil analisis.

Goal :
Menurunkan persentase keterlambatan

Objective :
Membuat model *machine learning* untuk memprediksi ketepatan waktu pengiriman barang agar mencegah keterlambatan agar persentase keterlambatan menurun. Perusahaan diharapkan dapat menggunakan model tersebut untuk menentukan keputusan bisnis sehingga customer retention dan tingkat kepuasan pelanggan tetap atau meningkat.

Business Metrics :
- Persentase keterlambatan

Data Exploration

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   10999 non-null  int64 
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64 
 4   Customer_rating      10999 non-null  int64 
 5   Cost_of_the_Product  10999 non-null  int64 
 6   Prior_purchases      10999 non-null  int64 
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64 
 10  Weight_in_gms        10999 non-null  int64 
 11  Reached.on.Time_Y.N  10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

Pengamatan:

1. Data terdiri dari 10999 baris
2. Tidak terdapat data yang null atau missing value
3. Sepertinya tidak ada issue yang mencolok pada tipe data untuk setiap kolom (sudah sesuai)

Dan dari beberapa kali sample

Sepertinya tidak ada anomali pada setiap entri kolom sudah sesuai

```
[4] df.sample(5)
```

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
1440	1441	D	Ship	3	2	273	3	low	M	41	3464	1
6032	6033	A	Flight	6	3	283	5	medium	M	10	1483	0
3838	3839	C	Flight	4	1	190	4	medium	F	5	5570	1
9812	9813	A	Ship	4	3	253	3	low	F	5	5703	0
8148	8149	D	Ship	3	2	176	2	medium	F	1	5292	0

Dan dari beberapa kali sample :

Sepertinya tidak ada anomali pada setiap entri kolom sudah sesuai

Exploratory Data Analysis

```
[6]  df[nums].describe()
```

	Customer_care_calls	Customer_rating	Prior_purchases	Discount_offered	Cost_of_the_Product	Weight_in_gms	Reached.on.Time_Y.N
count	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	4.054459	2.990545	3.567597	13.373216	210.196836	3634.016729	0.596691
std	1.141490	1.413603	1.522860	16.205527	48.063272	1635.377251	0.490584
min	2.000000	1.000000	2.000000	1.000000	96.000000	1001.000000	0.000000
25%	3.000000	2.000000	3.000000	4.000000	169.000000	1839.500000	0.000000
50%	4.000000	3.000000	3.000000	7.000000	214.000000	4149.000000	1.000000
75%	5.000000	4.000000	4.000000	10.000000	251.000000	5050.000000	1.000000
max	7.000000	5.000000	10.000000	65.000000	310.000000	7846.000000	1.000000

Beberapa pengamatan:

1. Kolom Customer_care_calls, customer_rating, dan Cost_of_the_Product tampak sudah cukup simetrik distribusinya (mean dan median tak berbeda jauh)
2. Kolom Discount_offered dan Prior_purchases tampaknya skew ke kanan (long-right tail)
3. Kolom Reached.on.Time_Y.Nbernilai boolean/binary

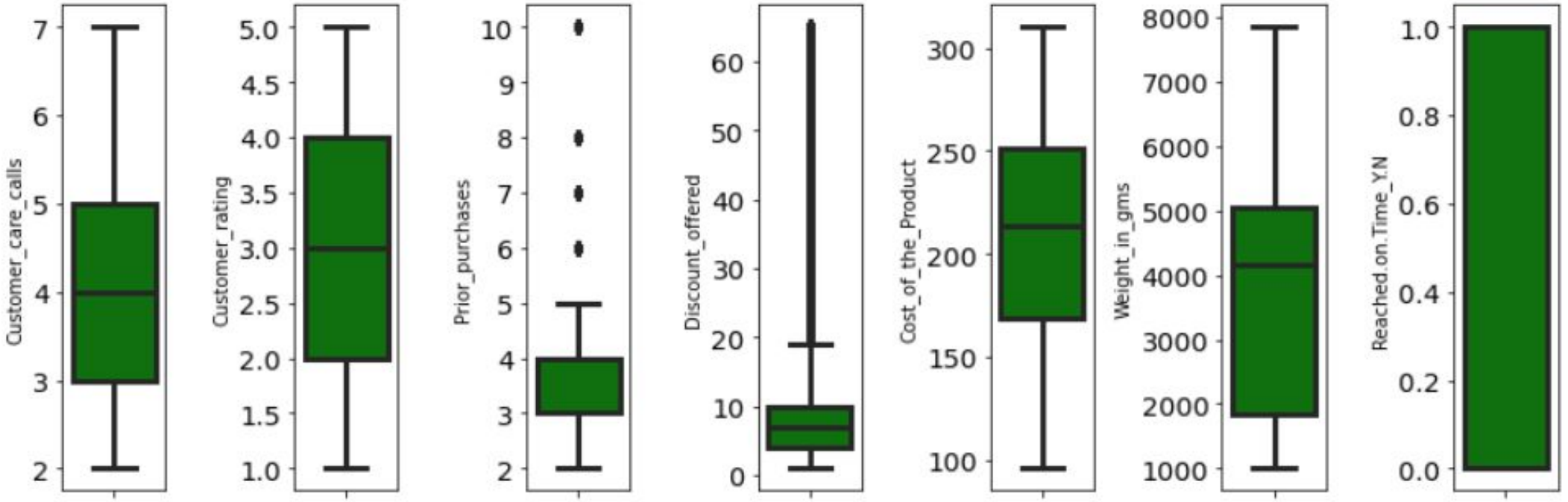
```
df[cats].describe()
```

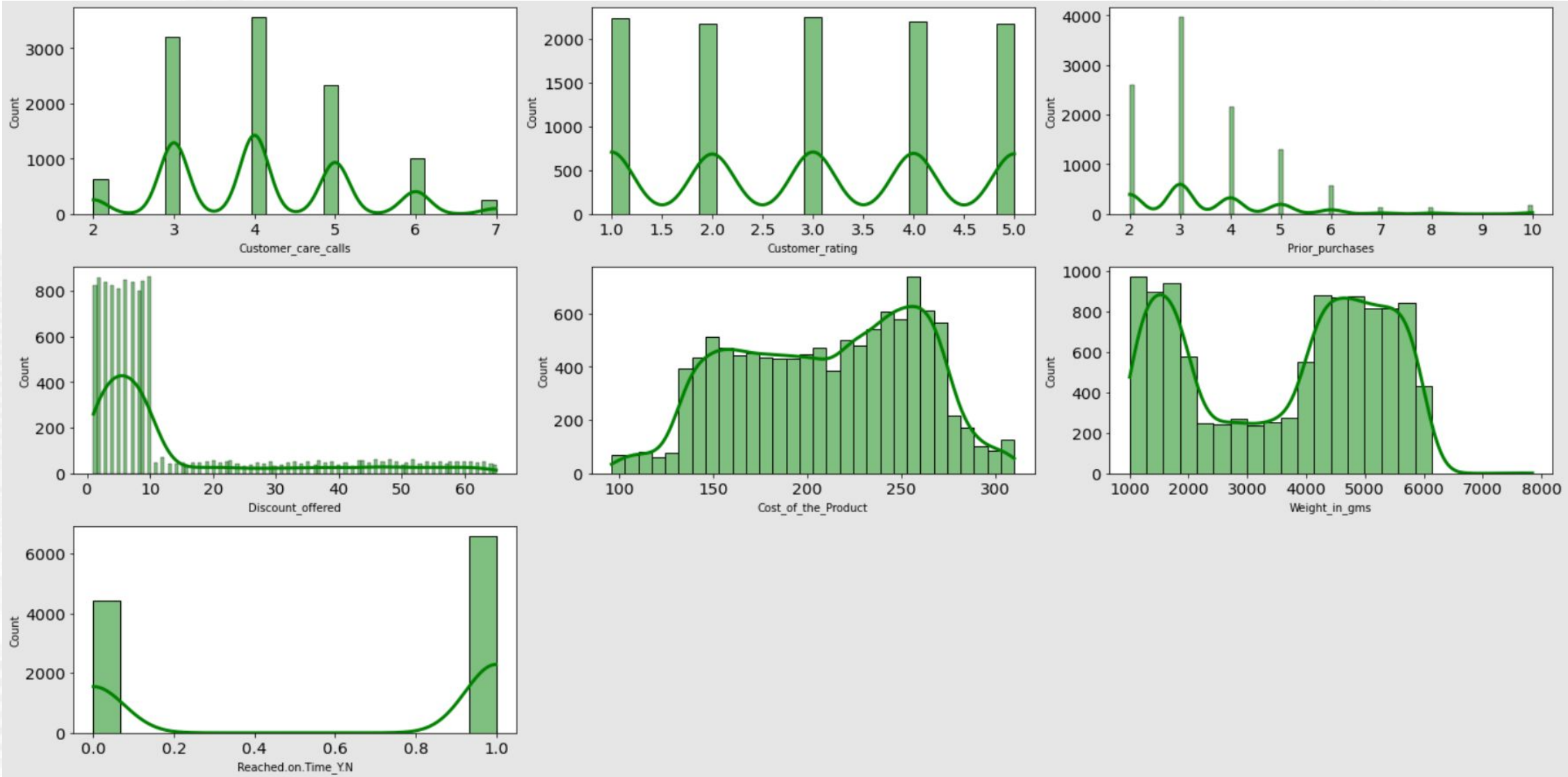
	Mode_of_Shipment	Product_importance	Gender	Warehouse_block
count	10999	10999	10999	10999
unique	3	3	2	5
top	Ship	low	F	F
freq	7462	5297	5545	3666

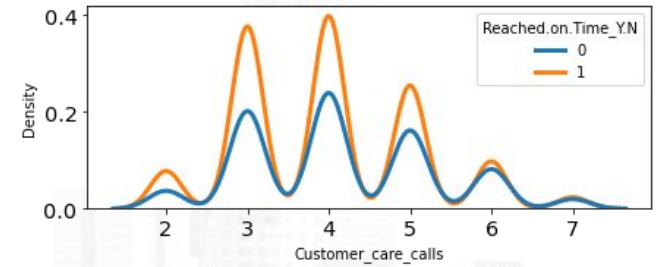
Beberapa pengamatan:

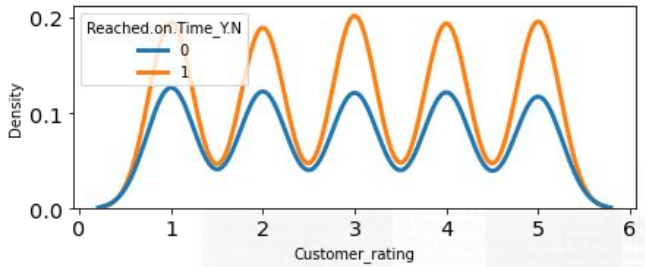
- Untuk kategori gender **perempuan** lebih dominan,
- untuk kategori product importance di dominasi oleh **kategori low**
- untuk kategori mode pengiriman di dominasi oleh **pengiriman menggunakan kapal (ship)**
- untuk warehouse_block didominasi oleh **block F**
- Semua unique value tiap kategori masih dalam kategori normal sekitar **2-5 unique values**

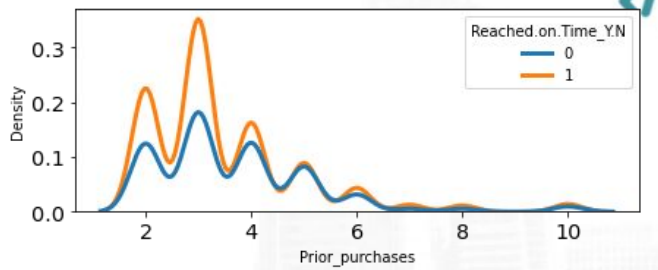
Exploratory Data Analysis - Univariate Analysis

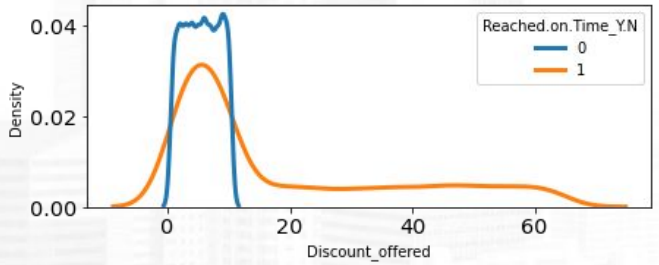


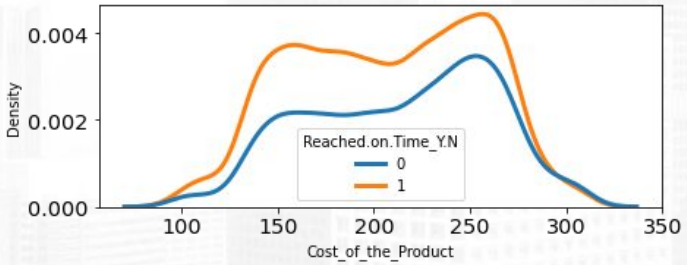

Count
Customer_care_calls
Customer_rating
Prior_purchases
Discount_offered
Cost_of_the_Product
Weight_in_gms
Reached.on.Time_Y.N

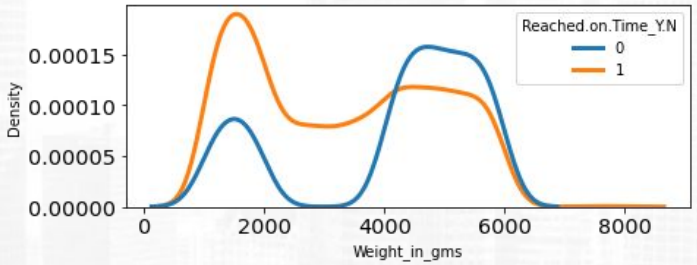


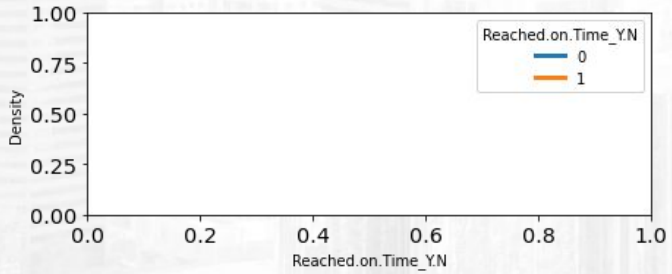






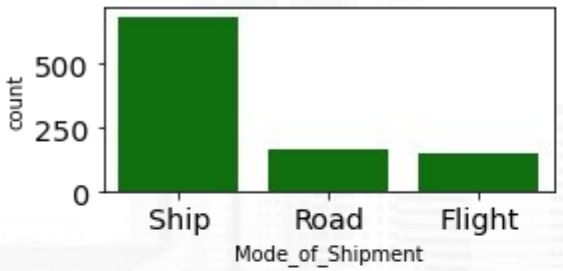


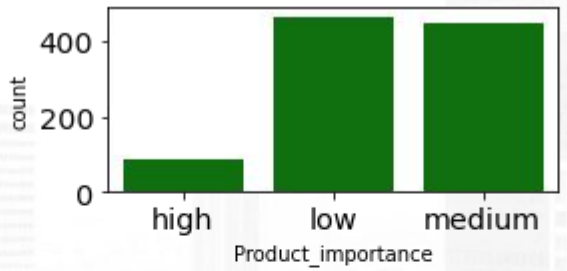


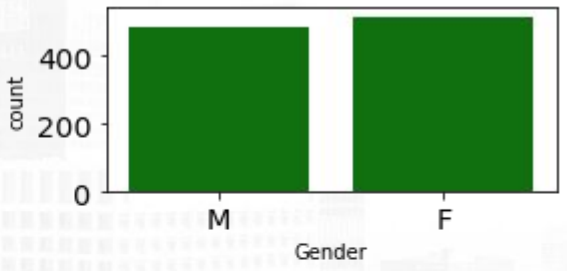


Dari distribution plot terlihat bahwa:

- Kolom cost_of_the_product tampak sudah mendekati distribusi normal
- Seperti dugaan kita ketika melihat boxplot di atas, kolom Prior_purchases, dan Discount_offered sedikit skewed Berarti ada kemungkinan kita perlu melakukan sesuatu pada kolom2 tersebut nantinya
- Kolom-kolom Reached.on.Time sejatinya adalah biner, sehingga tidak perlu terlalu diperhatikan bentuk distribusinya
- Untuk kolom weigh_in_gms terdapat ketidakpastian distribusi karena berbentuk u-shape.
- untuk kolom customer_care_calls dan customer_rating distibusi merata







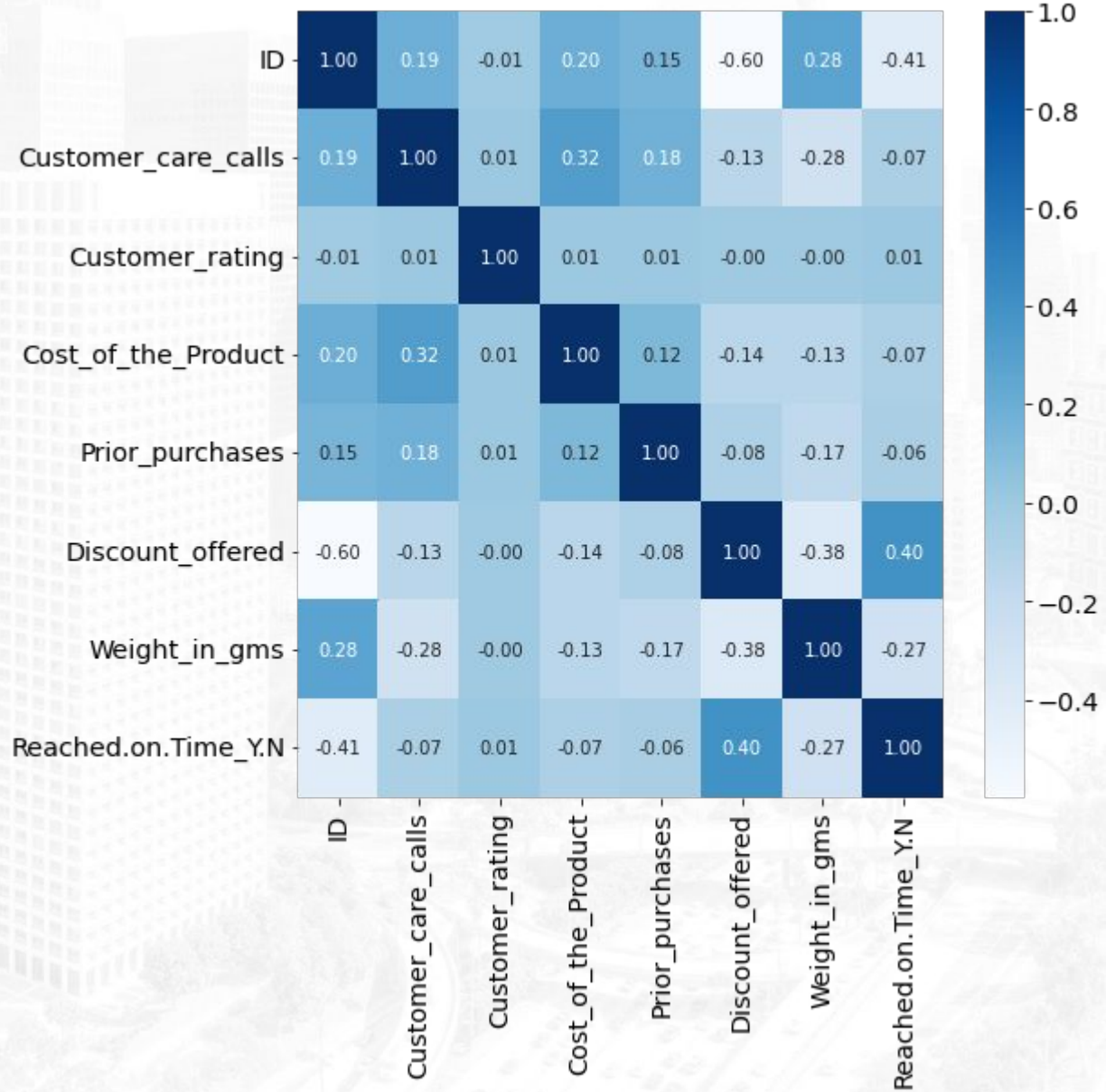
Seperti pengamatan kita sebelumnya, distribusi kategori low (Product_importance), gudang penyimpanan(warehouse_block) dan kategori Ship (Mode_of_Shipment) didominasi 1-2 value.

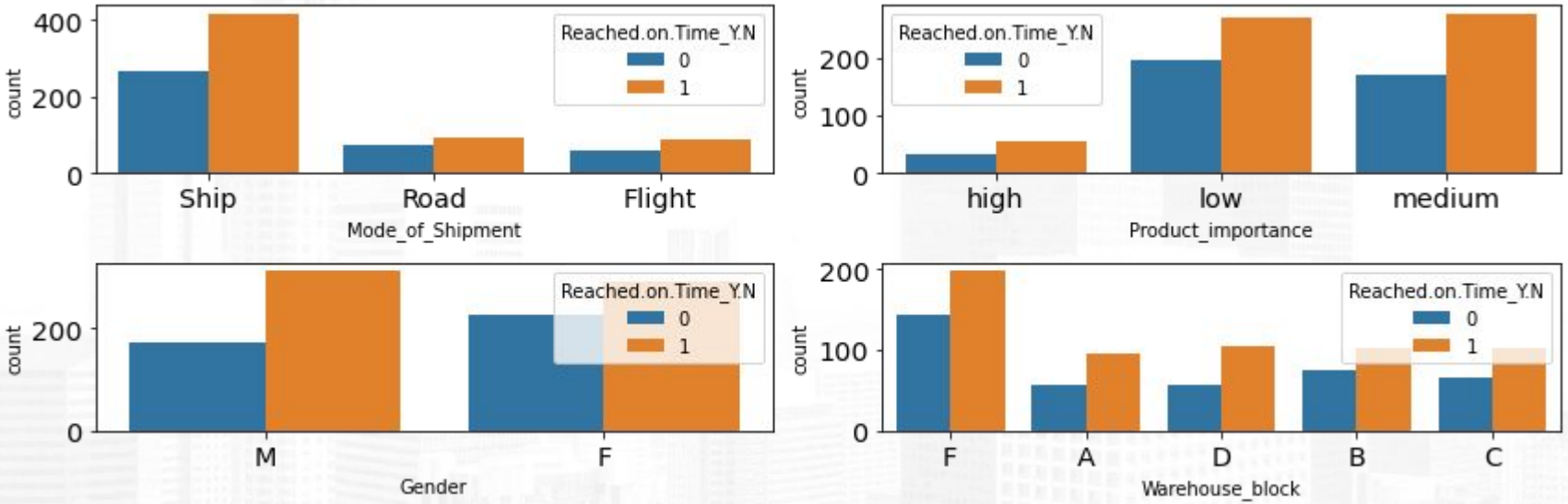
Exploratory Data Analysis - Bivariate Analysis

Rakamin Academy

Dari correlation heatmap di atas dapat dilihat bahwa:

- Target kita Reached.on.Time_Y.N memiliki korelasi positif lemah dengan customer_rating, cost_of_the_product, customer_care_calls dan prior_purchases
- Ia juga memiliki korelasi positif cukup kuat dengan Discount_offered
- Ia juga memiliki korelasi negatif cukup kuat dengan weight_in_gms





Pengamatan

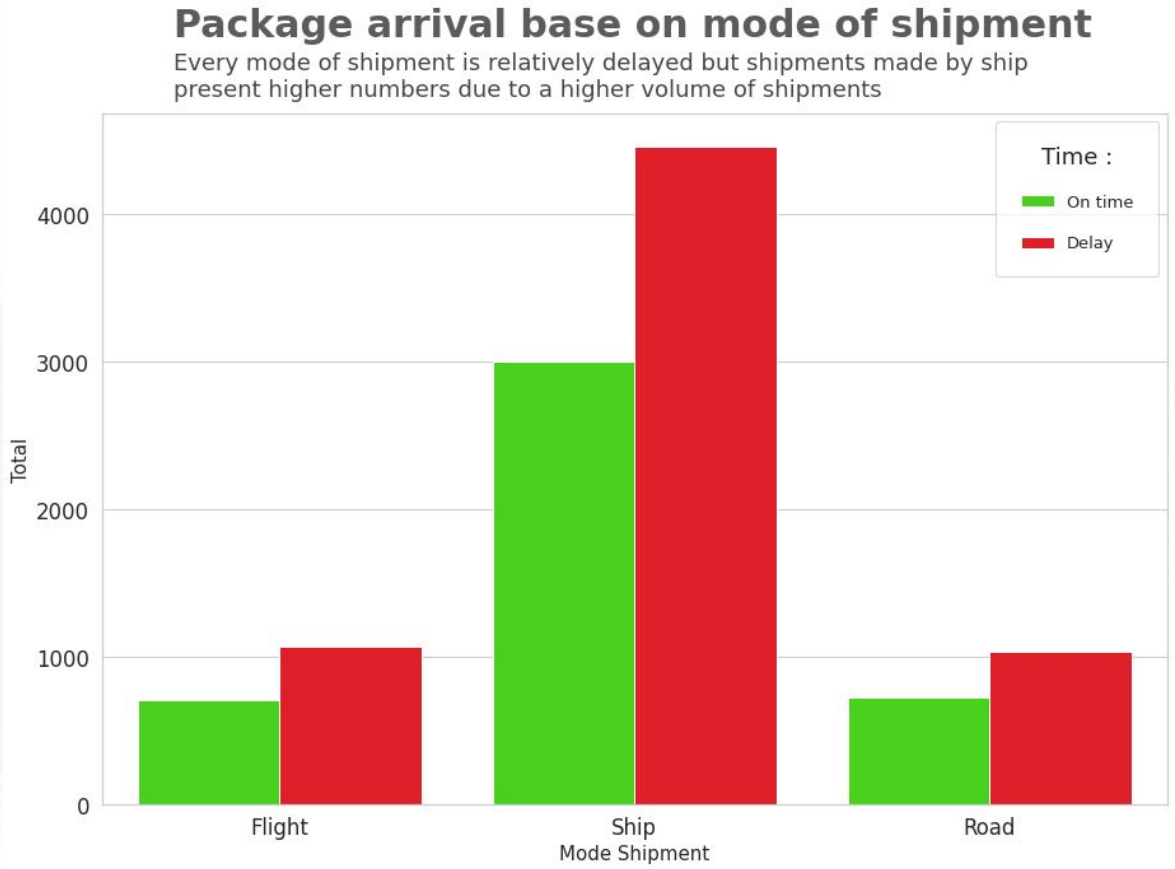
- shipment dengan ship cenderung akan mengalami telat pengiriman
- untuk produk_importance dengan kategori low dan medium cenderung akan mengalami telat pengiriman
- untuk warehouse_block dengan kategori F cenderung mengalami telat pengiriman

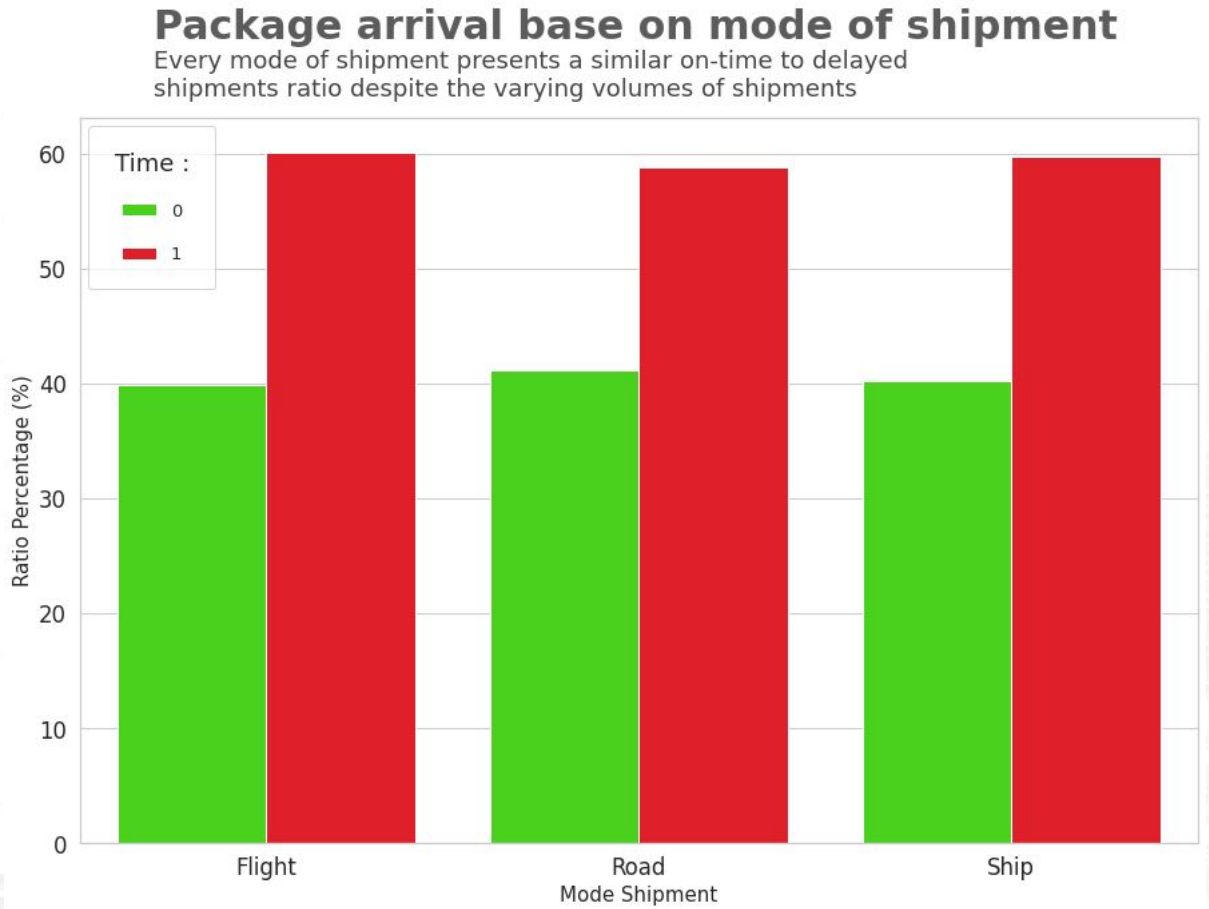
EDA Conclusion

Beberapa hal yang kita temukan dari EDA dataset ini adalah:

- Data terlihat valid dan tidak ada kecacatan yang major/signifikan
- Ada beberapa distribusi yang sedikit skewed, hal ini harus diingat apabila kita ingin melakukan sesuatu atau menggunakan model yang memerlukan asumsi distribusi normal
- Beberapa feature memiliki korelasi yang jelas dengan target, mereka akan dipakai
- Beberapa feature terlihat sama sekali tidak berkorelasi, mereka sebaiknya diabaikan
- Dari fitur kategorikal, “mode_of_shipment” ,” warehouse_block”, dan “product_importance” sepertinya berguna untuk menjadi prediktor model

Insight and Visualization

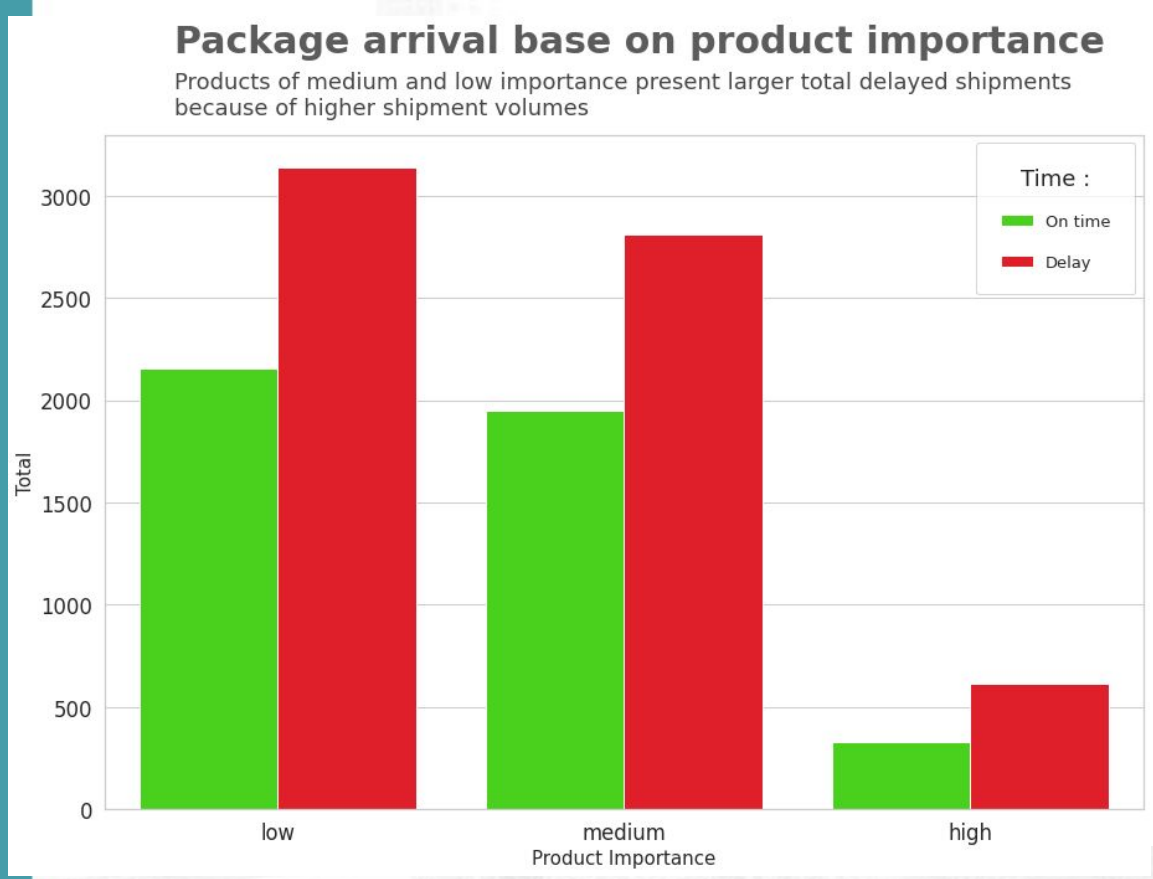


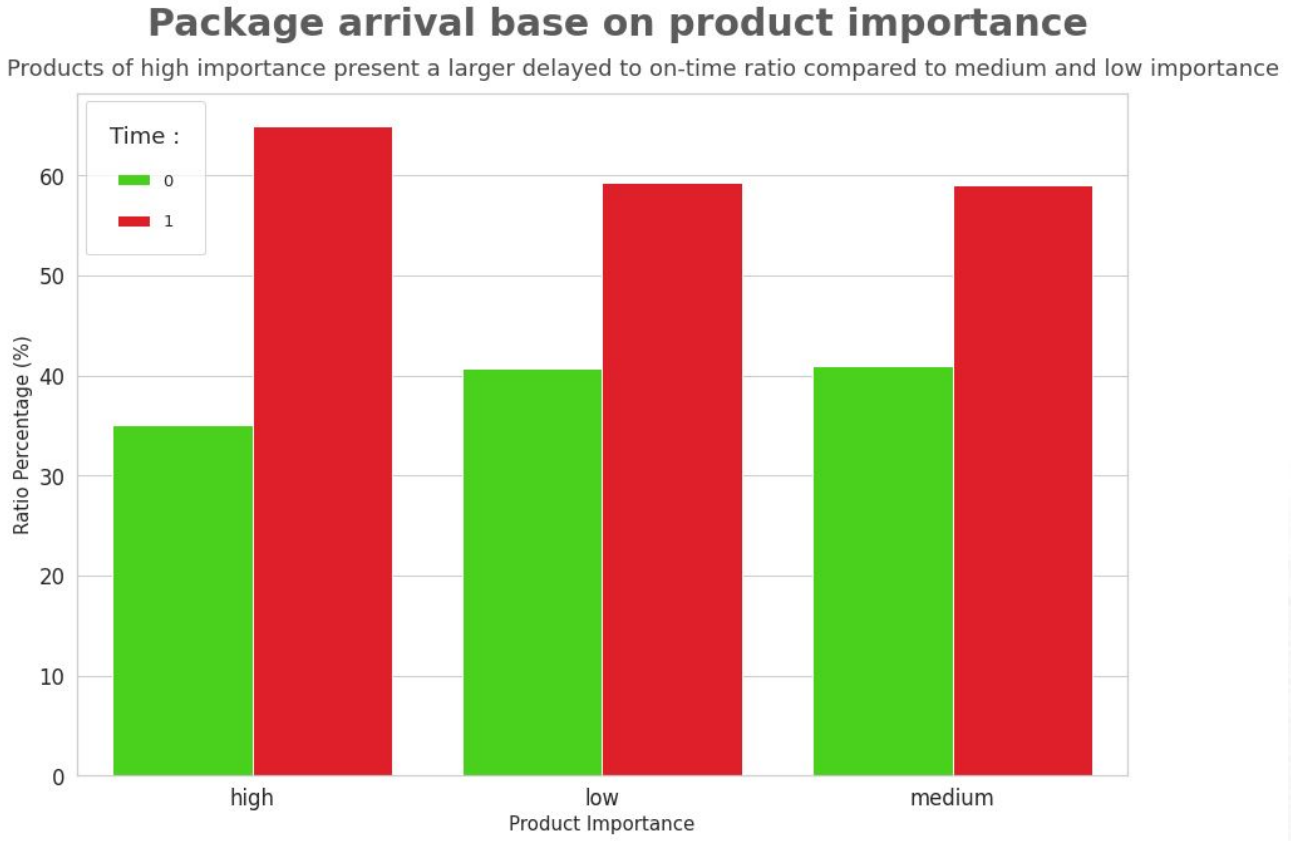


Insight: Pada awalnya kami berasumsi bahwa metode shipment ‘Ship’ berperforma paling buruk di antara ketiga metode shipping. Namun, saat dibandingkan berdasarkan ratio on-time dan delay, dapat dilihat bahwa ketiga-tiganya underperform. Namun, yang harus lebih diselidiki lagi adalah metode shipment ‘Flight’ karena seharusnya adalah metode shipment paling cepat.

Rekomendasi bisnis:

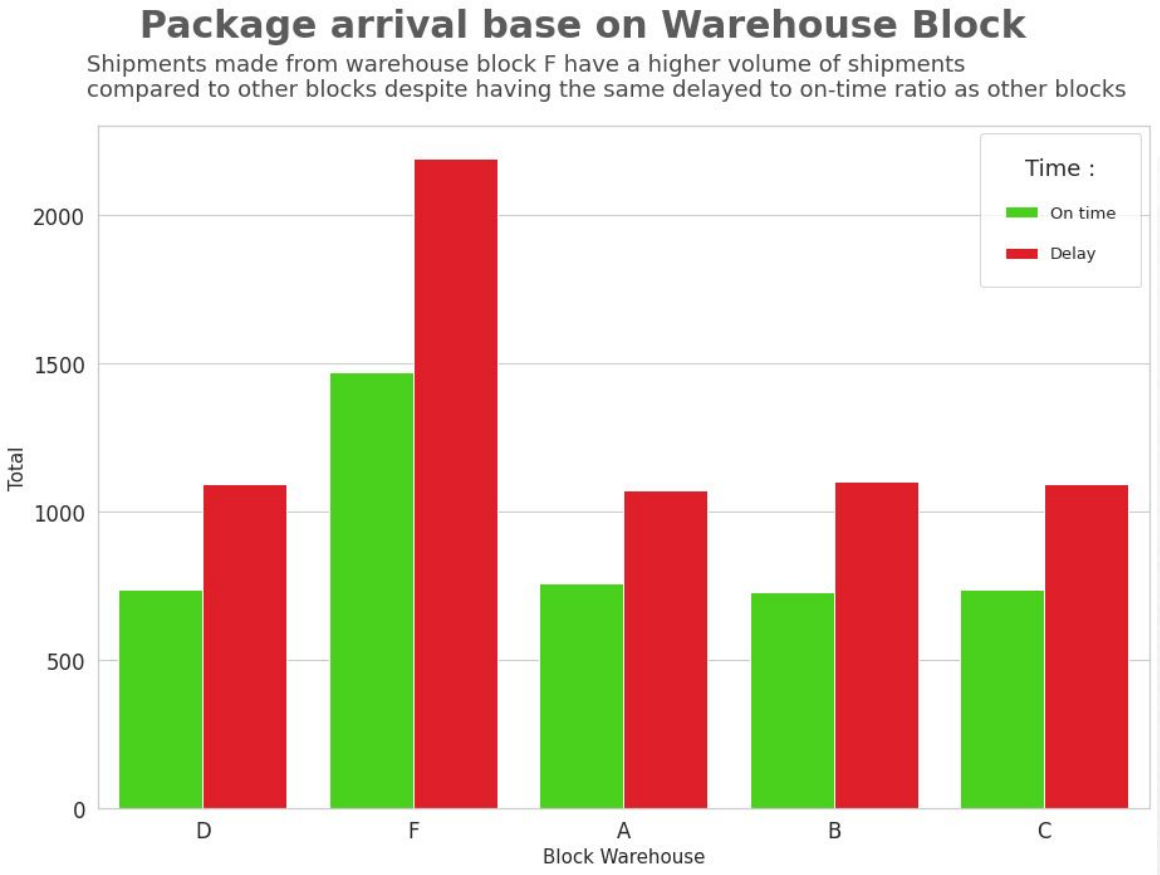
Client dapat meningkatkan performa logistik dan evaluasi internal untuk semua metode pengiriman terutama flight karena dapat dilihat underperformance dari ketiga-tiganya

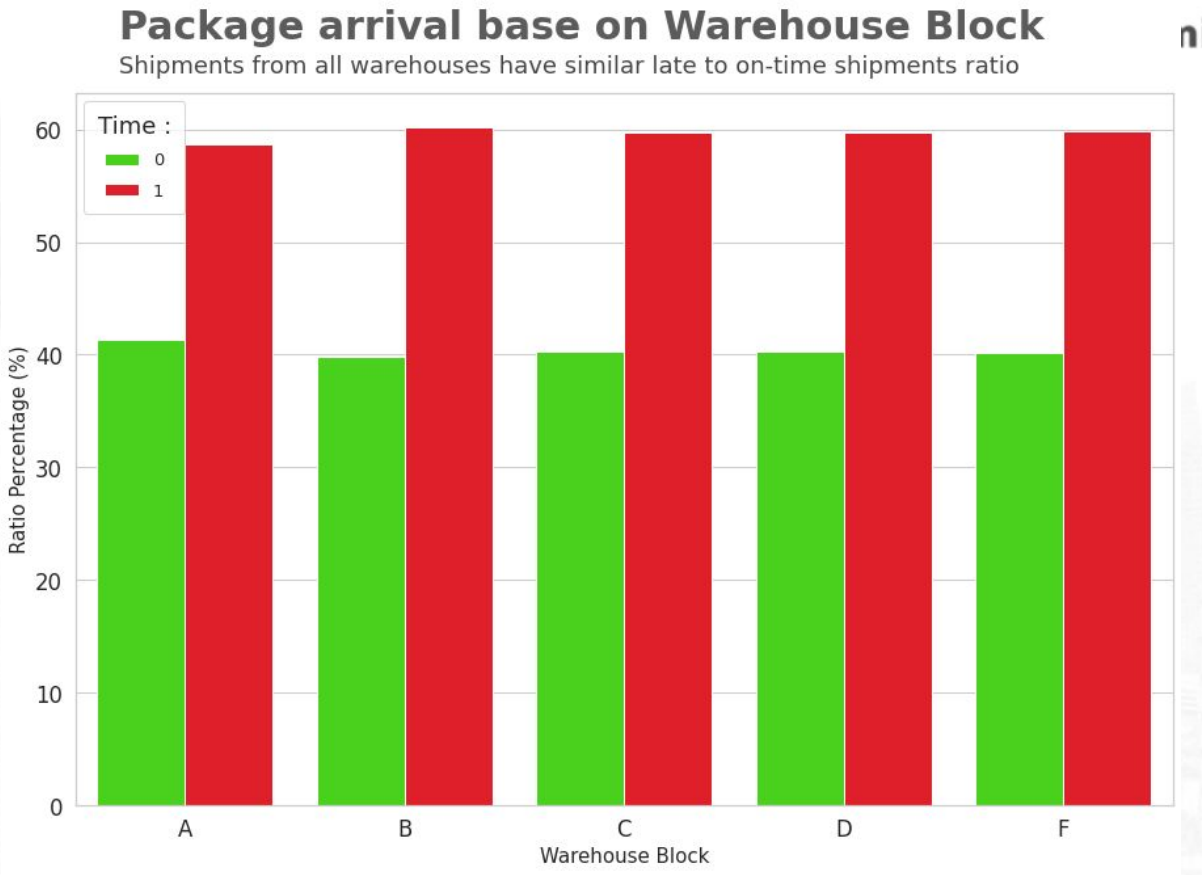




Insight: Kami berasumsi di awal bahwa low product importance mempunyai performance paling buruk. Namun saat dibandingkan secara persentase terlihat bahwa produk high importance memiliki persentase delayed tertinggi di antara ketiganya dimana seharusnya produk high importance diprioritaskan

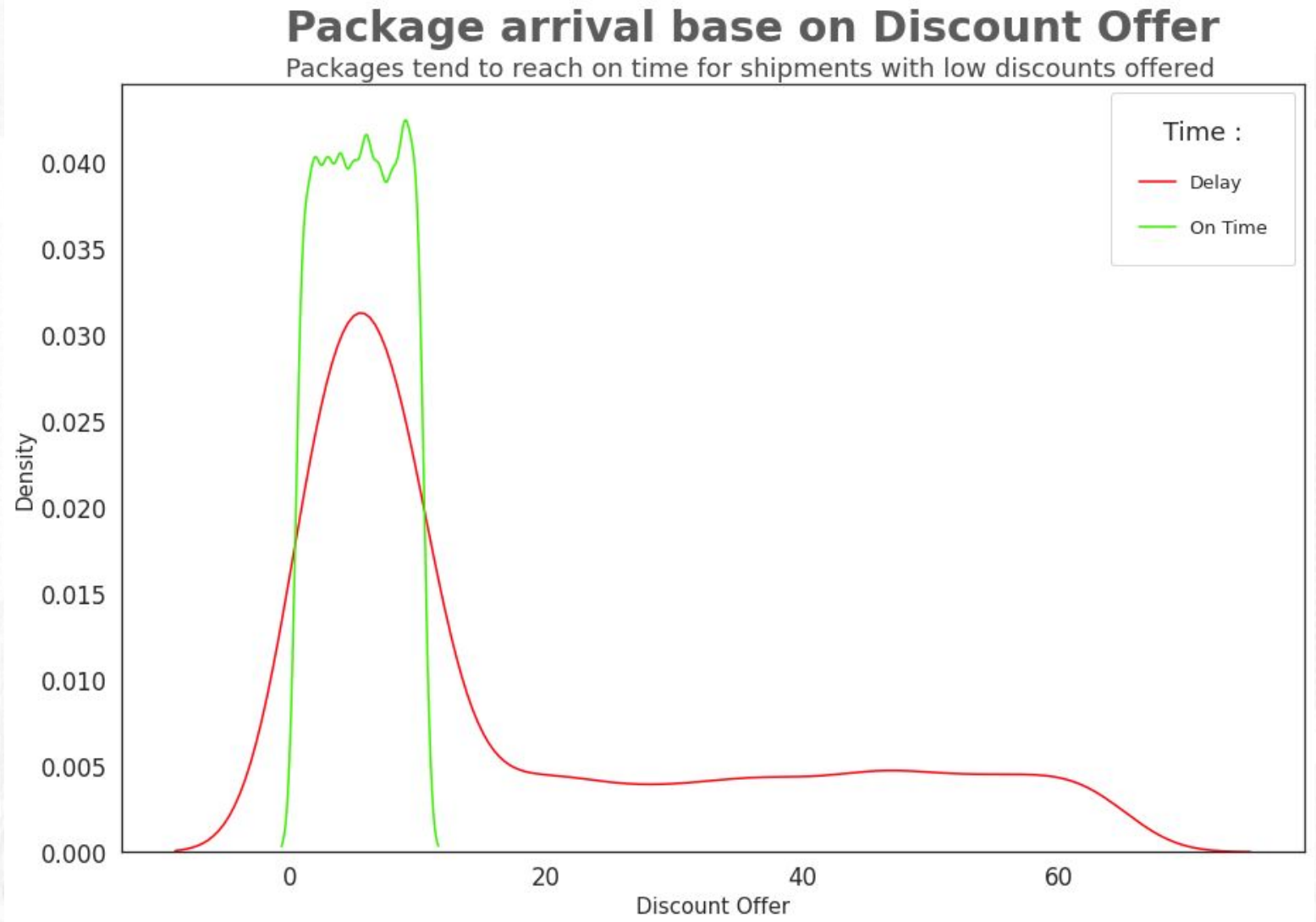
Rekomendasi bisnis:
Client dapat meningkatkan performa logistik untuk semua *product importance* terutama ‘high’ karena seharusnya diprioritaskan dan terlihat underperformance dari ketiga-tiganya





Insight: Asumsi awal adalah block F mempunyai performa paling buruk di antara 5 warehouse. Namun, saat dibandingkan secara persentase terlihat bahwa kelima-limanya underperform

Rekomendasi bisnis:
Client dapat meningkatkan performa logistik dan evaluasi internal untuk semua warehouse block karena dapat dilihat underperformance dari ketiga-tiganya



Insight: Di saat discount antara 0 dan < 20 shipment akan on time, namun saat discount yang lebih besar ditawarkan shipment yang delayed pun meningkat

Rekomendasi bisnis:
Client dapat meningkatkan SDM dalam pengiriman saat customer diberikan banyak discount seperti ‘Black Friday’, Harbolnas, dan event-event khusus lainnya

Data Cleansing - Handle Missing Value

```
df.isna().sum()
```

```
ID                      0
Warehouse_block         0
Mode_of_Shipment        0
Customer_care_calls     0
Customer_rating         0
Cost_of_the_Product     0
Prior_purchases         0
Product_importance      0
Gender                  0
Discount_offered        0
Weight_in_gms           0
Reached.on.Time_Y.N     0
dtype: int64
```

penjelasan : Missing values tidak perlu dihandle karena tidak ada missing values pada setiap feature

Data Cleansing - Handle Duplicated Data

```
df.duplicated().sum()
```

```
0
```

```
df.duplicated(subset=['Customer_care_calls', 'Customer_rating', 'Prior_purchases', 'Discount_offered', 'Cost_of_the_Product',
                      'Weight_in_gms', 'Reached.on.Time_Y.N', 'Mode_of_Shipment', 'Product_importance', 'Gender','Warehouse_block']).sum()
```

```
0
```

penjelasan : Duplicated Data tidak perlu dihandle karena tidak ada duplicated data pada setiap feature

Data Cleansing - Handle Missing Value

```
[13] #code
     df2 = df.copy()
     print(f'Jumlah baris sebelum memfilter outlier: {len(df2)}')

     filtered_entries1 = np.array([True] * len(df2))

     for col in nums:
         zscore = abs(stats.zscore(df2[col])) # hitung absolute z-scorenya
         filtered_entries1 = (zscore < 3) & filtered_entries1 # keep yang kurang dari 3 absolute z-scorenya

     df2 = df2[filtered_entries1] # filter, cuma ambil yang z-scorenya dibawah 3

     print(f'Jumlah baris setelah memfilter outlier: {len(df2)}')
```

```
Jumlah baris sebelum memfilter outlier: 10999
Jumlah baris setelah memfilter outlier: 10642
```

penjelasan : dengan menggunakan z-score untuk setiap feature yang ada membuang sekitar 3% data outlier jadi data menjadi 10642. karena kami menganggap setiap data berharga jadi kami menggunakan z-score untuk tidak membuang terlalu banyak data

Data Cleansing - Handle Outliers

```
df2 = df.copy()
print(f'Jumlah baris sebelum memfilter outlier: {len(df2)}')

filtered_entries1 = np.array([True] * len(df2))

for col in nums:
    zscore = abs(stats.zscore(df2[col])) # hitung absolute z-scorenya
    filtered_entries1 = (zscore < 3) & filtered_entries1 # keep yang kurang dari 3 absolute z-scorenya

df2 = df2[filtered_entries1] # filter, cuma ambil yang z-scorenya dibawah 3

print(f'Jumlah baris setelah memfilter outlier: {len(df2)}')
```

```
Jumlah baris sebelum memfilter outlier: 10999
Jumlah baris setelah memfilter outlier: 10642
```

penjelasan : dengan menggunakan z-score untuk setiap feature yang ada membuang sekitar 3% data outlier jadi data menjadi 10642. karena kami menganggap setiap data berharga jadi kami menggunakan z-score untuk tidak membuang terlalu banyak data

Data Cleansing - Feature Transformation

```
df2.describe()
```

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
count	10642.000000	10642.000000	10642.000000	10642.000000	10642.000000	10642.000000	10642.000000	10642.000000
mean	5570.347773	4.065683	2.989194	210.578557	3.463447	12.528660	3646.514189	0.590303
std	3159.806013	1.145348	1.412344	48.077818	1.288855	14.992539	1639.849048	0.491801
min	1.000000	2.000000	1.000000	96.000000	2.000000	1.000000	1001.000000	0.000000
25%	2881.250000	3.000000	2.000000	170.000000	3.000000	4.000000	1837.000000	0.000000
50%	5604.500000	4.000000	3.000000	215.000000	3.000000	7.000000	4172.000000	1.000000
75%	8300.750000	5.000000	4.000000	251.000000	4.000000	10.000000	5063.750000	1.000000
max	10999.000000	7.000000	5.000000	310.000000	8.000000	61.000000	7846.000000	1.000000

```
# Normalisasi :
df2['Customer_rating'] = MinMaxScaler().fit_transform(df2['Customer_rating'].values.reshape(len(df2), 1))

#Standarisasi :
df2['Customer_care_calls'] = StandardScaler().fit_transform(df2['Customer_care_calls'].values.reshape(len(df2), 1))
df2['Cost_of_the_Product'] = StandardScaler().fit_transform(df2['Cost_of_the_Product'].values.reshape(len(df2), 1))
df2['Prior_purchases'] = StandardScaler().fit_transform(df2['Prior_purchases'].values.reshape(len(df2), 1))
df2['Discount_offered'] = StandardScaler().fit_transform(df2['Discount_offered'].values.reshape(len(df2), 1))
df2['Weight_in_gms'] = StandardScaler().fit_transform(df2['Weight_in_gms'].values.reshape(len(df2), 1))
```

Hasil Normalisasi dan Standarisasi

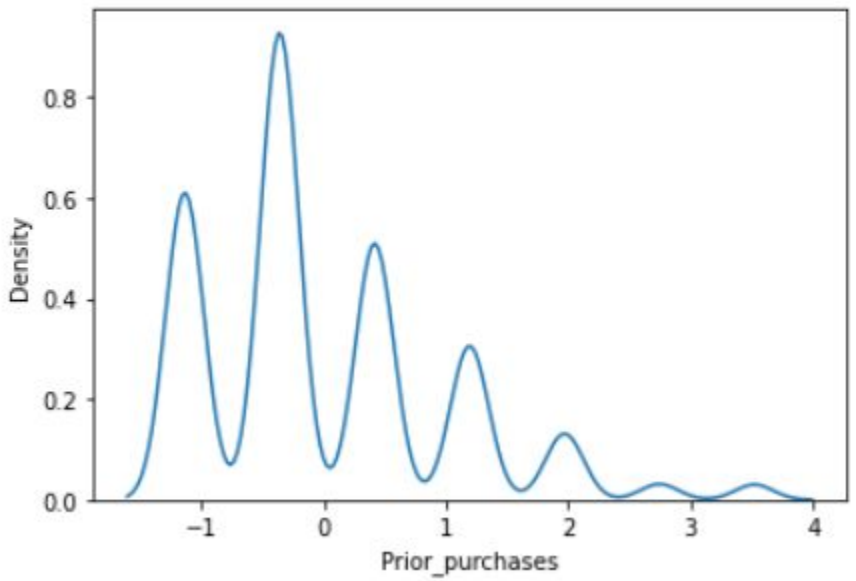
```
df2.describe()
```

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
count	10642.000000	1.064200e+04	10642.000000	1.064200e+04	1.064200e+04	1.064200e+04	1.064200e+04	10642.000000
mean	5570.347773	6.986956e-15	0.497298	5.756426e-16	-6.094584e-15	2.935728e-14	1.339529e-17	0.590303
std	3159.806013	1.000047e+00	0.353086	1.000047e+00	1.000047e+00	1.000047e+00	1.000047e+00	0.491801
min	1.000000	-1.803627e+00	0.000000	-2.383302e+00	-1.135516e+00	-7.689960e-01	-1.613343e+00	0.000000
25%	2881.250000	-9.304889e-01	0.250000	-8.440579e-01	-3.595969e-01	-5.688870e-01	-1.103516e+00	0.000000
50%	5604.500000	-5.735048e-02	0.500000	9.196863e-02	-3.595969e-01	-3.687781e-01	3.204627e-01	1.000000
75%	8300.750000	8.157880e-01	0.750000	8.407899e-01	4.163217e-01	-1.686692e-01	8.642883e-01	1.000000
max	10999.000000	2.562065e+00	1.000000	2.068025e+00	3.519996e+00	3.233183e+00	2.561018e+00	1.000000

penjelasan : beberapa fitur dilakukan standarisasi agar lebih mudah untuk pemodelan dan juga agar fitur mendekati distribusi normal. terkhusus 'customer_rating' di normalisasi karena kita sudah mengetahui batasan dari rating yaitu 1-5 jadi hanya perlu di normalisasi

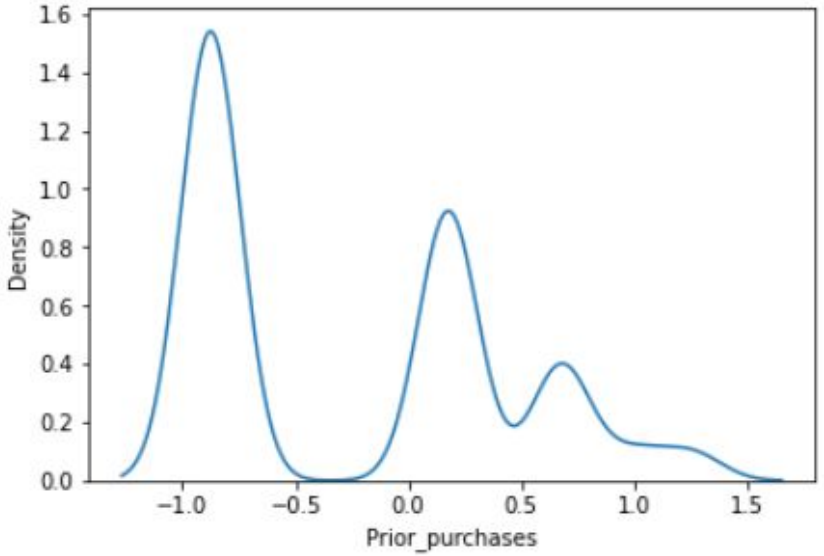
Log Transformation

```
sns.kdeplot(df2['Prior_purchases']);
```

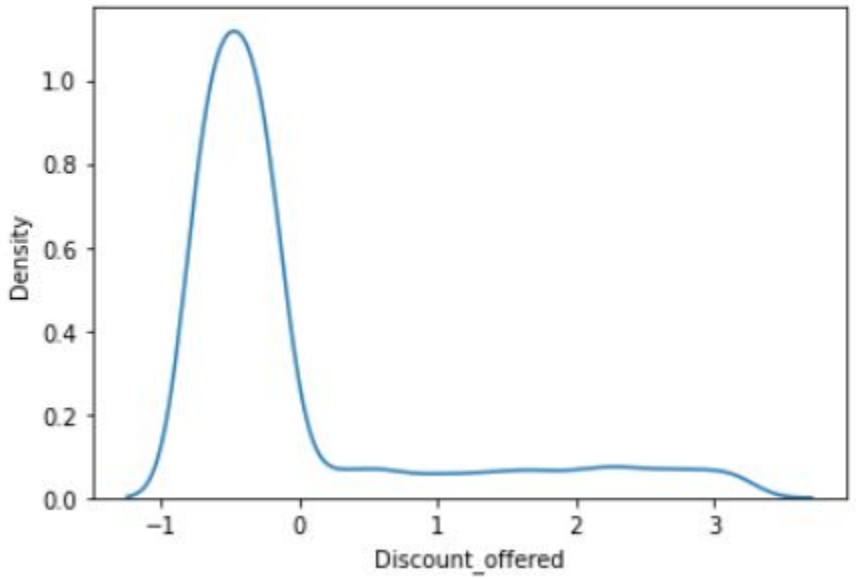



```
sns.kdeplot(np.log(df2['Prior_purchases']));
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arra
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

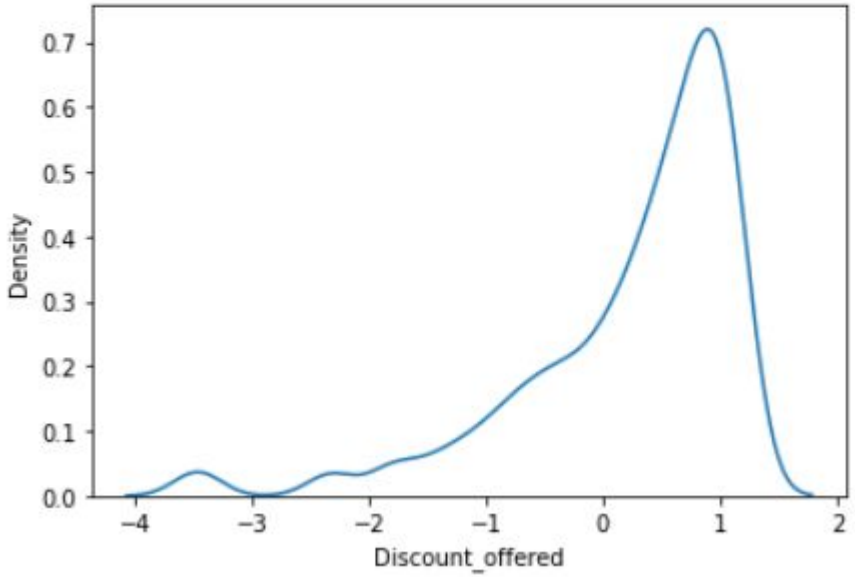


```
sns.kdeplot(df2['Discount_offered']);
```

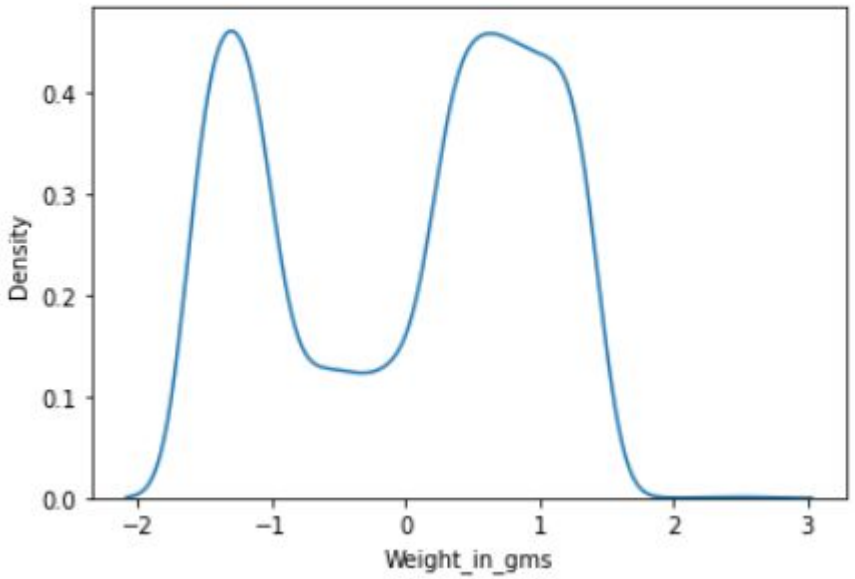



```
sns.kdeplot(np.log(df2['Discount_offered']));
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arr
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

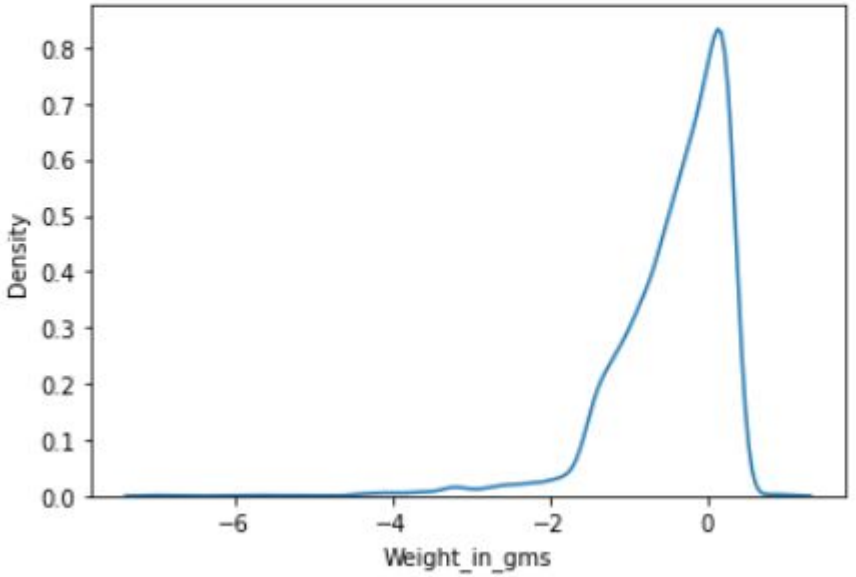


```
sns.kdeplot(df2['Weight_in_gms']);
```

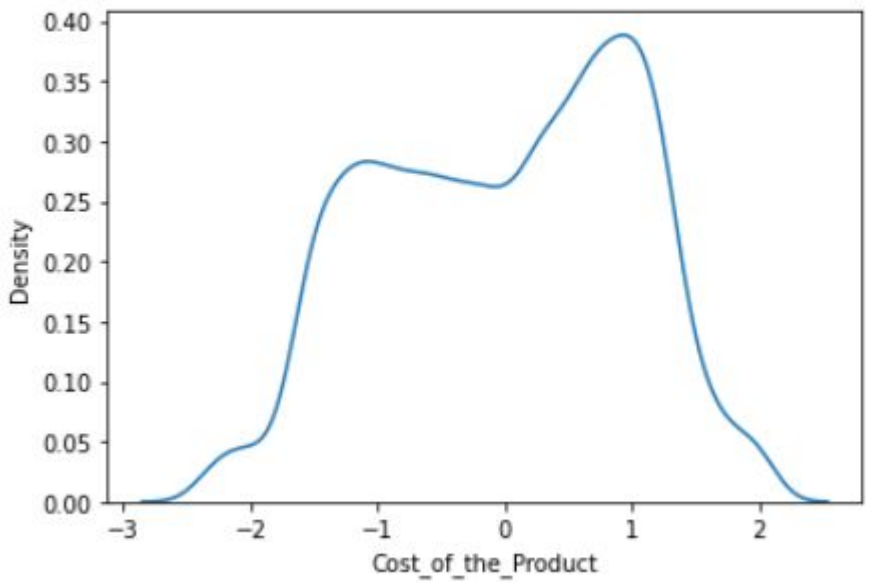



```
sns.kdeplot(np.log(df2['Weight_in_gms']));
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arra
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

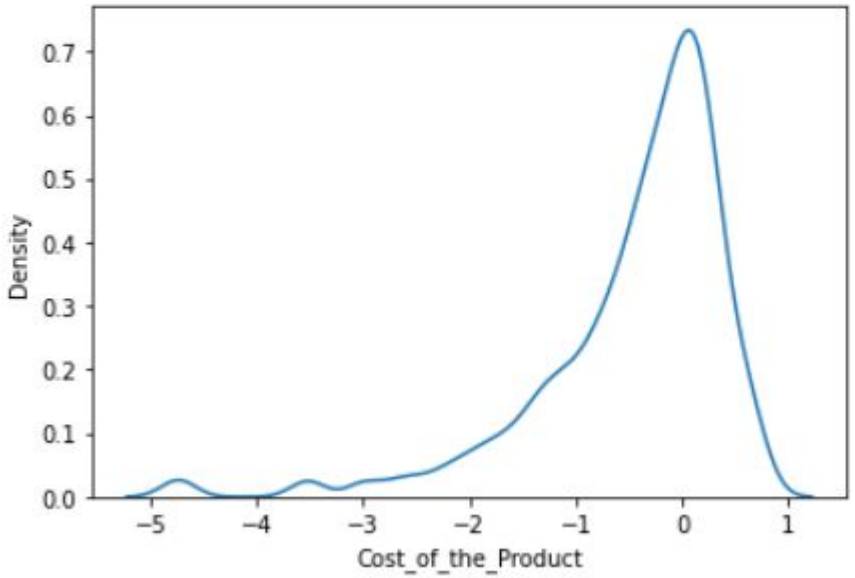


```
sns.kdeplot(df2['Cost_of_the_Product']);
```

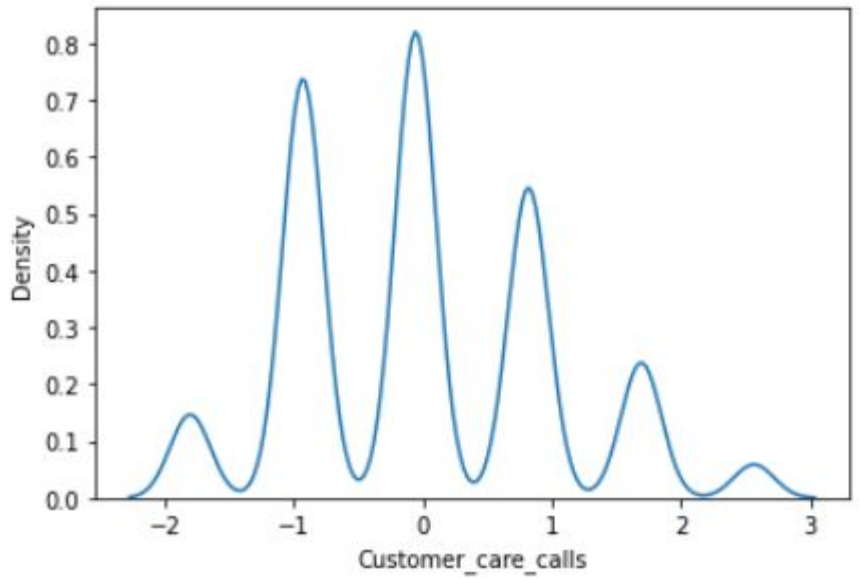



```
sns.kdeplot(np.log(df2['Cost_of_the_Product']));
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arra
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

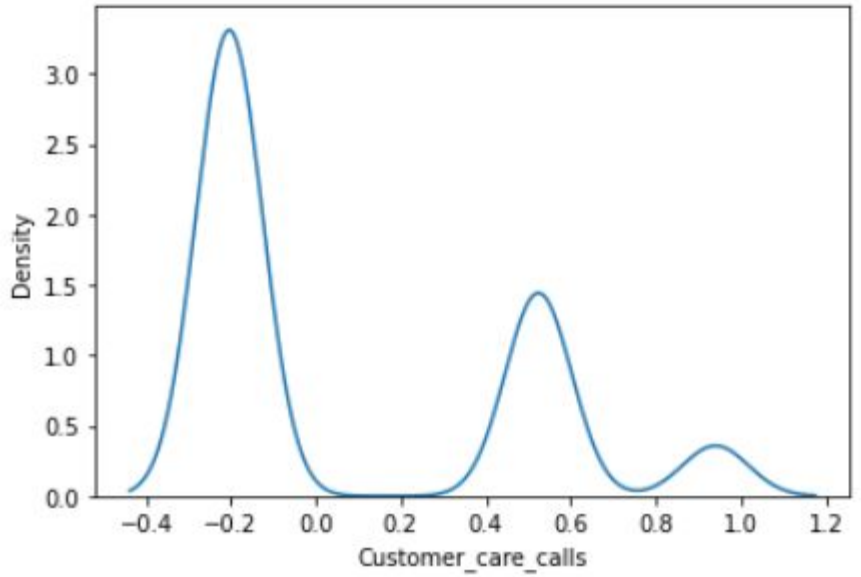


```
sns.kdeplot(df2['Customer_care_calls']);
```




```
sns.kdeplot(np.log(df2['Customer_care_calls']));
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arra
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



```
df2['log_prior_purchases'] = np.log(df2['Prior_purchases'])
df2['log_discount_offered'] = np.log(df2['Discount_offered'])
df2['log_weight_in_gms'] = np.log(df2['Weight_in_gms'])
df2['log_cost_of_the_Product'] = np.log(df2['Cost_of_the_Product'])
df2['log_customer_care_calls'] = np.log(df2['Customer_care_calls'])

df2 = df2.drop(columns=['ID', 'Prior_purchases', 'Discount_offered', 'Weight_in_gms','Cost_of_the_Product', 'Customer_care_calls'])
df2.describe()
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/arraylike.py:364: RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

	Customer_rating	Reached.on.Time_Y.N	log_prior_purchases	log_discount_offered	log_weight_in_gms	log_cost_of_the_Product	log_customer_care_calls
count	10642.000000	10642.000000	4212.000000	2304.000000	6145.000000	5573.000000	3521.000000
mean	0.497298	0.590303	-0.232526	0.224030	-0.425354	-0.464535	0.081788
std	0.353086	0.491801	0.694371	0.952796	0.724980	0.914966	0.399492
min	0.000000	0.000000	-0.876297	-3.459681	-7.006352	-4.736844	-0.203601
25%	0.250000	0.000000	-0.876297	-0.184072	-0.764418	-0.808381	-0.203601
50%	0.500000	1.000000	-0.876297	0.530048	-0.252014	-0.198464	-0.203601
75%	0.750000	1.000000	0.175834	0.916071	0.101796	0.142193	0.524093
max	1.000000	1.000000	1.258460	1.173467	0.940405	0.726594	0.940814

Penjelasan : Terdapat beberapa perubahan dari beberapa fitur agar mendekati distribusi normal dengan log transformation seperti Prior_purchases, Discount_offered, Weight_in_gms, Cost_of_the_Product , Customer_care_calls

Data Cleansing - Feature Encoding

```
value counts of column Mode_of_Shipment
Ship      7462
Flight    1777
Road      1760
Name: Mode_of_Shipment, dtype: int64
------------------------------

value counts of column Product_importance
low       5297
medium    4754
high       948
Name: Product_importance, dtype: int64
------------------------------

value counts of column Gender
F    5545
M    5454
Name: Gender, dtype: int64
------------------------------

value counts of column Warehouse_block
F    3666
D    1834
A    1833
B    1833
C    1833
Name: Warehouse_block, dtype: int64
------------------------------
```

Penjelasan :

- Label encoding : mengubah distinct values dengan nilai tertentu. Digunakan pada data kategorikal dengan jumlah distinct values = 2 atau data benbentuk ordinal(dapat diurutkan)
- One hot encoding : mengubah distinct values menjadi feature tersendiri. Digunakan pada data nominal(tidak dapat diurutkan)

Teknik encoding

- Product_importance & Gender : Label Encoding
- Mode_of_Shipment & Warehouse_block : One Hot Encoding

Data Cleansing - Feature Encoding

Label Encoding

Mengubah value

'low' : 0,
'medium' : 1,
'high' : 2

'F' : 0,
'M' : 1

Product_importance	Gender
medium	F
medium	F
low	F
medium	M
low	F

Product_importance	Gender	I
1	0	
1	0	
0	0	
1	1	
0	0	

Data Cleansing - Feature Encoding

One Hot Encoding

Mengubah menjadi feature tersendiri

Warehouse_block	Mode_of_Shipment	C
A	Ship	
B	Ship	
C	Ship	
F	Ship	
D	Ship	

Mode_of_Shipment_Flight	Mode_of_Shipment_Road	Mode_of_Shipment_Ship	Warehouse_block_A	Warehouse_block_B	Warehouse_block_C	Warehouse_block_D	Warehouse_block_F
0	0	1	1	0	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	1
0	0	1	0	0	0	1	0

Data Cleansing - Handle Class Imbalance

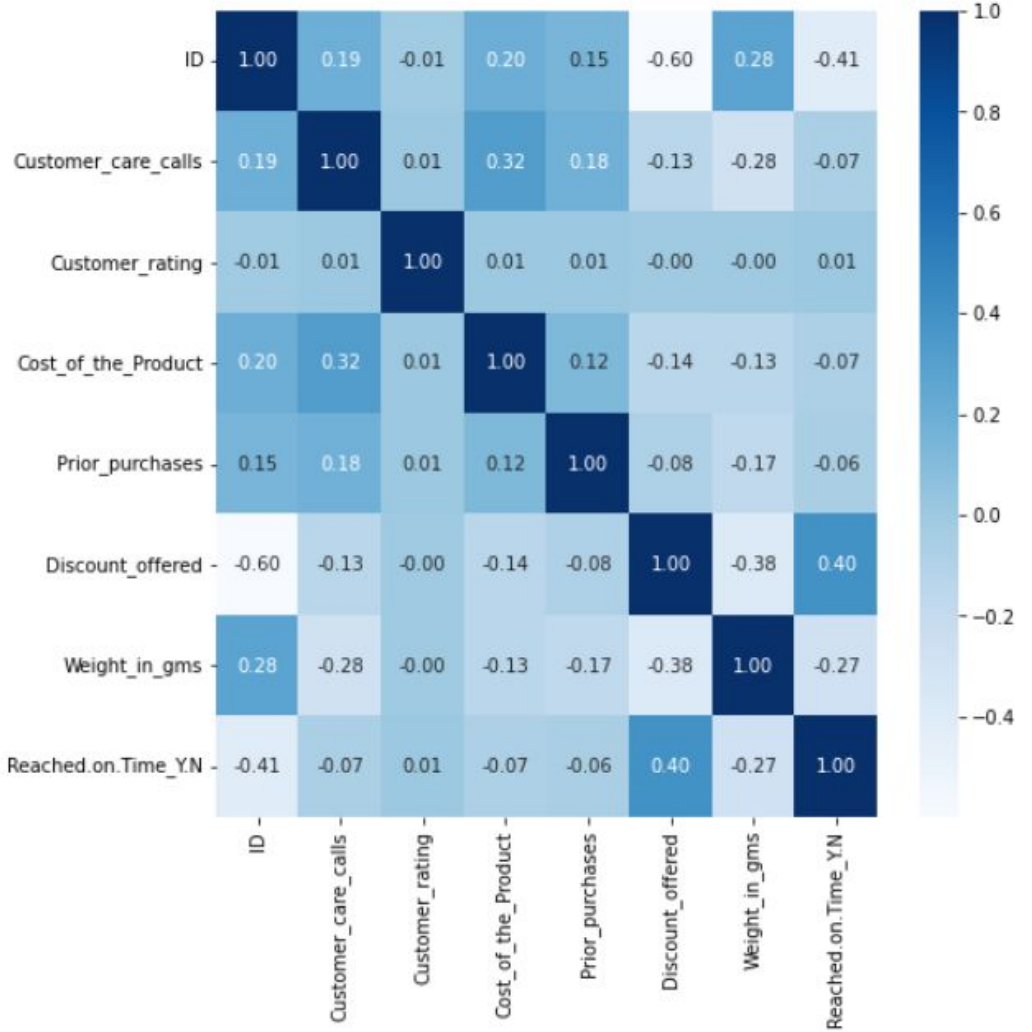
```
[33] df['Reached.on.Time_Y.N'].value_counts()
0s

1    6563
0    4436
Name: Reached.on.Time_Y.N, dtype: int64
```

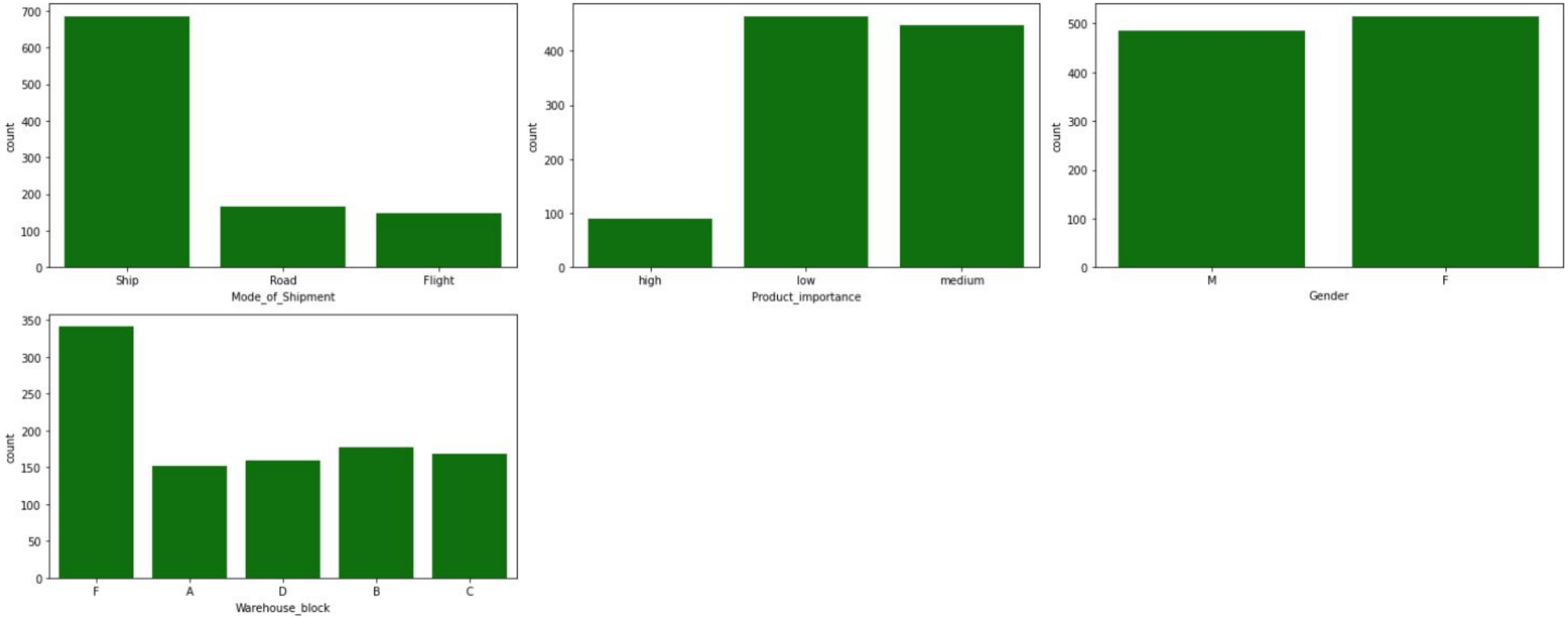
penjelasan :

rasio target prediksi sudah cukup balance, jadi tidak perlu over/undersampling

Feature Engineering - Feature Selection



penjelasan: From the figure above, some features that passed the threshold (0,05) according to the targeted feature (Reached.on.Time_Y.N) are chosen to be processed, such as Customer_care_calls, Cost_of_the_Products, Prior_Purchases, Discount_offered, and Weight_in_grms



Penjelasan : From the figure above can be seen that all features have a quite bigdifference in terms of sum in counts, except Gender, that features is excluded because have the same numbers in counts

Feature Engineering - Feature Extraction

	Warehouse	Reached.on.Time_Y.N	Count	Total_Warehouse	Persen
0	A	0	758	1833	41.35
1	A	1	1075	1833	58.65
2	B	0	729	1833	39.77
3	B	1	1104	1833	60.23
4	C	0	739	1833	40.32
5	C	1	1094	1833	59.68
6	D	0	738	1834	40.24
7	D	1	1096	1834	59.76
8	F	0	1472	3666	40.15
9	F	1	2194	3666	59.85

Penjelasan : untuk menghitung persentase tiap warehouse yang on time dan tidak

	Rating	Reached.on.Time_Y.N	Count	Total Rating	Persen
0	1	0	922	2235	41.25
1	1	1	1313	2235	58.75
2	2	0	892	2165	41.20
3	2	1	1273	2165	58.80
4	3	0	882	2239	39.39
5	3	1	1357	2239	60.61
6	4	0	886	2189	40.48
7	4	1	1303	2189	59.52
8	5	0	854	2171	39.34
9	5	1	1317	2171	60.66

Penjelasan : untuk menghitung persentase tiap rating yang on time dan tidak

	Importance	Reached.on.Time_Y.N	Count	Total importance	Persen
0	high	0	332	948	35.02
1	high	1	616	948	64.98
2	low	0	2157	5297	40.72
3	low	1	3140	5297	59.28
4	medium	0	1947	4754	40.95
5	medium	1	2807	4754	59.05

Penjelasan : untuk menghitung persentase tiap prorduct importance yang on time dan tidak

	Purchase	Reached.on.Time_Y.N	Count	Total purchase	Percentage
0	2	0	974	2599	37.48
1	2	1	1625	2599	62.52
2	3	0	1421	3955	35.93
3	3	1	2534	3955	64.07
4	4	0	984	2155	45.66
5	4	1	1171	2155	54.34
6	5	0	645	1287	50.12
7	5	1	642	1287	49.88
8	6	0	247	561	44.03
9	6	1	314	561	55.97
10	7	0	44	136	32.35
11	7	1	92	136	67.65
12	8	0	45	128	35.16
13	8	1	83	128	64.84
14	10	0	76	178	42.70
15	10	1	102	178	57.30

Penjelasan : untuk menghitung persentase jumlah pembelian yang on time dan tidak

Feature Engineering - Feature Tambahan

4 feature tambahan yang mungkin akan sangat membantu membuat performansi model semakin bagus:

1. Date time - Agar mengetahui waktu keberangkatan paket
2. Destination - Agar mengetahui tempat tujuan paket
3. Type of package - Agar mengetahui tipe package yang akan dikirim
4. Distance - Agar mengetahui jarak pengiriman barang

MODELING

- Model yang akan dibuat adalah **model Supervised Learning** Jenis **Klasifikasi**
- Model digunakan untuk memprediksi suatu barang atau *product* yang dikirim akan telat atau tepat waktu pada saat pengiriman

Langkah-langkah modelling :

- Membuat fungsi untuk model evaluasi, feature importance dan Best Parameters
- Memisahkan antara fitur dan target dari dataset
- Memisahkan antara data train dan data test
- Memanggil fungsi atau mengimport library sesuai dengan algoritma yang ingin dipakai
- Melakukan pemodelan dan mengevaluasinya

Perbandingan Hasil dari berbagai algoritma

No	Algortima Model	Accuracy	Precision	Recall	F1-Score	AUC
1.	Decision Tree	0.62	0.67	0.70	0.68	0.61
2.	Logistic Regression	0.64	0.69	0.67	0.68	0.63
3.	Lightgbm	0.66	0.76	0.60	0.67	0.725
4.	KNN	0.63	0.69	0.64	0.67	0.63
5.	Random Forest	0.65	0.72	0.64	0.68	0.65
6.	XGBoost	0.67	0.87	0.51	0.64	0.70

EVALUATION

Evaluation Metric yang digunakan **AUC (Area Under Curve)** karena Area Under the Curve (AUC) sendiri mengukur kemampuan classifier untuk membedakan masing-masing class hasil prediksi.

Jadi , Score AUC yang lebih besar lebih baik, karena dapat membedakan prediksi label satu dengan lainnya

Hyperparameter Tuning - Decision Tree

```python
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform

# List of hyperparameter
max_depth = [int(x) for x in np.linspace(1, 110, num = 30)] # Maximum number of levels in tree
min_samples_split = [2, 5, 10, 100] # Minimum number of samples required to split a node
min_samples_leaf = [1, 2, 4, 10, 20, 50] # Minimum number of samples required at each leaf node
max_features = ['auto', 'sqrt'] # Number of features to consider at every split
criterion = ['gini', 'entropy']
splitter = ['best', 'random']

hyperparameters = dict(max_depth=max_depth,
                       min_samples_split=min_samples_split,
                       min_samples_leaf=min_samples_leaf,
                       max_features=max_features,
                       criterion=criterion,
                       splitter=splitter
                      )

# Inisialisasi Model
dt = DecisionTreeClassifier(random_state=42)
model = RandomizedSearchCV(dt, hyperparameters, cv=5, random_state=42, scoring='roc_auc')
model.fit(X_train, y_train)

# Predict & Evaluation
y_pred = model.predict(X_test)#Check performa dari model
eval_classification(model, y_pred, X_train, y_train, X_test, y_test)
```

```
Accuracy (Test Set): 0.65
Precision (Test Set): 0.72
Recall (Test Set): 0.66
F1-Score (Test Set): 0.69
AUC: 0.65
```

```
Best max_depth: 110
Best min_samples_split: 100
Best min_samples_leaf: 4
Best max_features: sqrt
Best criterion: entropy
Best splitter: best
```

Hyperparameter Tuning - Logistic Regression

```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV

# List Hyperparameters yang akan diuji
penalty = ['l2','l1','elasticnet']
C = [0.0001, 0.001, 0.002] # Inverse of regularization strength; smaller values specify stronger regularization.
solver = ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
hyperparameters = dict(penalty=penalty, C=C , solver=solver)

# Inisiasi model
logres = LogisticRegression(random_state=42) # Init Logres dengan Gridsearch, cross validation = 5
model = RandomizedSearchCV(logres, hyperparameters, cv=5, random_state=42, scoring='roc_auc')

# Fitting Model & Evaluation
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
eval_classification(model, y_pred, X_train, y_train, X_test, y_test)
```

```
Accuracy (Test Set): 0.58
Precision (Test Set): 0.58
Recall (Test Set): 1.00
F1-Score (Test Set): 0.73
AUC: 0.50
```

```
print('Best algorithm:', model.best_estimator_.get_params()['penalty'])
print('Best C:', model.best_estimator_.get_params()['C'])
print('Best solver:', model.best_estimator_.get_params()['solver'])
```

```
Best algorithm: l2
Best C: 0.0001
Best solver: saga
```

Hyperparameter Tuning - lightgbm

```
model = ltb.LGBMClassifier(learning_rate=0.09,num_leaves = 3, max_depth=2,random_state=42, num_iterations=132)
model.fit(xtrain,ytrain,eval_set=[(xtest,ytest),(xtrain,ytrain)],eval_metric='auc')
print(); print(model)

expectedy  = ytest
predictedy = model.predict(xtest)
```

```
LGBMClassifier(learning_rate=0.09, max_depth=2, num_iterations=132,
               num_leaves=3, random_state=42)
```

```
from sklearn import metrics
print(metrics.classification_report(expectedy, predictedy))
print(metrics.confusion_matrix(expectedy, predictedy))
```

```
              precision    recall  f1-score   support

           0       0.58      0.89      0.70       897
           1       0.87      0.52      0.65      1232

    accuracy                           0.68      2129
   macro avg       0.72      0.71      0.68      2129
weighted avg       0.75      0.68      0.67      2129

[[802  95]
 [589 643]]
```

Hyperparameter Tuning - KNN

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import uniform

# List of hyperparameter
n_neighbors = list(range(2,30))
p=[1,2]
algorithm = ['auto', 'ball_tree', 'kd_tree', 'brute']
hyperparameters = dict(n_neighbors=n_neighbors, p=p, algorithm=algorithm)

# Init model
knn = KNeighborsClassifier()
model = RandomizedSearchCV(knn, hyperparameters, cv=5, random_state=42, scoring='roc_auc')

# Fit Model & Evaluasi
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
eval_classification(model, y_pred, X_train, y_train, X_test, y_test)
```

```
Accuracy (Test Set): 0.66
Precision (Test Set): 0.78
Recall (Test Set): 0.56
F1-Score (Test Set): 0.65
AUC: 0.67
```

Hyperparameter Tuning - Random Forest

```
# Hyperparameter Tuning Random Forest
parameter_range = np.arange(1, 10, 1)

train_score, test_score = validation_curve(
                                RandomForestClassifier(random_state = 42),
                                X = X_train, y = y_train,
                                param_name = 'n_estimators',
                                param_range = parameter_range,
                                cv = 5,
                                scoring = 'roc_auc')
```

```
Accuracy : 0.67
Precision : 0.88
Recall : 0.50
F-1Score : 0.64
ROC AUC : 0.70
AP : 0.73
```

Hyperparameter Tuning - XGBoost

```
[ ]  from sklearn.model_selection import RandomizedSearchCV
     import numpy as np

     #Menjadikan ke dalam bentuk dictionary
     hyperparameters = {
                        'max_depth' : [int(x) for x in np.linspace(10, 110, num = 9)],
                        'min_child_weight' : [int(x) for x in np.linspace(1, 20, num = 11)],
                        'gamma' : [float(x) for x in np.linspace(0, 1, num = 11)],
                        'tree_method' : ['auto', 'exact', 'approx', 'hist'],

                        'colsample_bytree' : [float(x) for x in np.linspace(0, 1, num = 11)],
                        'eta' : [float(x) for x in np.linspace(0, 1, num = 100)],

                        'lambda' : [float(x) for x in np.linspace(0, 1, num = 11)],
                        'alpha' : [float(x) for x in np.linspace(0, 1, num = 11)]
                        }

     # Init
     xg = XGBClassifier(random_state=42)
     xg_tuned1 = RandomizedSearchCV(xg, hyperparameters, cv=5, random_state=42, scoring='roc_auc')
     xg_tuned1.fit(X_train,y_train)

     # Predict & Evaluation
     y_pred = xg_tuned1.predict(X_test)#Check performa dari model
     eval_classification(xg_tuned1, y_pred, X_train, y_train, X_test, y_test)
```

```
Accuracy (Test Set): 0.64
Precision (Test Set): 0.68
Recall (Test Set): 0.72
F1-Score (Test Set): 0.70
AUC: 0.63
```

Perbandingan Hasil dari berbagai algoritma dengan Hyperparameter

No	Algortima Model	Accuracy	Precision	Recall	F1-Score	AUC
1.	Decision Tree	0.65	0.72	0.66	0.69	0.65
2.	Logistic Regression	0.58	0.58	1.00	0.73	0.50
3.	Lightgbm	0.66	0.76	0.60	0.67	0.739
4.	KNN	0.66	0.78	0.56	0.65	0.67
5.	Random Forest	0.61	0.62	0.86	0.72	0.57
6.	XGBoost	0.64	0.68	0.72	0.70	0.63

Perbandingan Hasil dari berbagai Algoritma dengan Feature Selection

Rakamin Academy

No	Algortima Model	Accuracy	Precision	Recall	F1-Score	AUC
1.	Decision Tree	0.63	0.67	0.70	0.69	0.62
2.	Logistic Regression	0.63	0.68	0.67	0.68	0.62
3.	Lightgbm	0.66	0.76	0.60	0.67	0.725
4.	KNN	0.63	0.69	0.64	0.67	0.63
5.	Random Forest	0.57	0.58	0.98	0.73	0.50
6.	XGBoost	0.68	0.88	0.51	0.65	0.71

Perbandingan Hasil dari berbagai Algoritma dengan Feature Selection dengan Hyperparameter

No	Algortima Model	Accuracy	Precision	Recall	F1-Score	AUC
1.	Decision Tree	0.65	0.72	0.66	0.69	0.65
2.	.Logistic Regression	0.58	0.58	1.00	0.73	0.50
3.	Lightgbm	0.66	0.76	0.60	0.67	0.739
4.	KNN	0.66	0.78	0.56	0.65	0.67
5.	Random Forest	0.67	0.88	0.50	0.64	0.70
6.	XGBoost	0.65	0.71	0.67	0.69	0.65

MODEL AKHIR

- Berdasarkan hasil dari tiap algoritma model dan diskusi kelompok, kami memutuskan untuk menggunakan model **Desicion Tree**

- Karena model tersebutmemiliki kombinasi **Skor AUC dan Recall** yang cukup balance dan lebih bisa diinterpretasikan hasilnya