E-Commerce Shipping Dataset :

Product Shipment Delivered on time or not ; To Meet E-Commerce Customer Demand

The data contains the following information:

1. **ID** : ID Number of Customers.
2. **Warehouse block** : The Company have big Warehouse which is divided in to block such as A,B,C,D,E.
3. **Mode of shipment** :The Company Ships the products in multiple way such as Ship, Flight and Road.
4. **Customer care calls** : The number of calls made from enquiry for enquiry of the shipment.
5. **Customer rating** : The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best).
6. **Cost of the product** : Cost of the Product in US Dollars.
7. **Prior purchases** : The Number of Prior Purchase.
8. **Product importance** : The company has categorized the product in the various parameter such as low, medium, high.
9. **Gender** : Male and Female.
10. **Discount offered** : Discount offered on that specific product.
11. **Weight in gms** : It is the weight in grams.
12. **Reached on time** : It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

Data Pre-processing

Load & Describe Data

Import library

In [53]:
```
import numpy as np
import pandas as pd

#Visualization
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('fivethirtyeight') # use style fivethirtyeight
import seaborn as sns
from matplotlib import rcParams
import warnings
warnings.filterwarnings("ignore")

# Scaling
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Selection
from scipy.stats import chi2_contingency

# Splitting the data into Train and Test
from sklearn.model_selection import train_test_split
```

```python
# Algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

# Evaluation metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report

# Hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
```

Import file

In [54]:
```python
df = pd.read_csv(r"C:\Users\HP\Downloads\Train.csv")
```

Rename column target

In [55]:
```python
df.rename(columns={'Reached.on.Time_Y.N':'is_late'}, inplace=True)
df.head()
```

Out[55]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Pr
0	1	D	Flight	4	2	177	
1	2	F	Flight	4	5	216	
2	3	A	Flight	2	2	183	
3	4	B	Flight	3	3	176	
4	5	C	Flight	2	2	184	

Because of the target's name is too long, so we simplify the name to ease the next step.

Get the shape of dataset

In [56]:
```python
df.shape
```

Out[56]:
```
(10999, 12)
```

Get list of columns

In [57]:
```python
df.columns
```

Out[57]:
```
Index(['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',
       'is_late'],
      dtype='object')
```

Change all column names to lower case

In [58]:
```python
df.columns = df.columns.str.lower()
```

Get dataset information

In [59]:
```python
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  is_late              10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

In [60]:
```
df.describe()
```

Out[60]:

	id	customer_care_calls	customer_rating	cost_of_the_product	prior_purchases	discount_o
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.0
mean	5500.00000	4.054459	2.990545	210.196836	3.567597	13.3
std	3175.28214	1.141490	1.413603	48.063272	1.522860	16.2
min	1.00000	2.000000	1.000000	96.000000	2.000000	1.0
25%	2750.50000	3.000000	2.000000	169.000000	3.000000	4.0
50%	5500.00000	4.000000	3.000000	214.000000	3.000000	7.0
75%	8249.50000	5.000000	4.000000	251.000000	4.000000	10.0
max	10999.00000	7.000000	5.000000	310.000000	10.000000	65.0

Based on the information above :

1. Dataframe has 10999 rows and 12 columns.
2. No missing values are found.
3. There are only 2 data types, integer and object.
4. Classification target `is_late` and others we call features.

Separate numeric & categorical column

In [61]:
```
# Categorical data
categorical = ['warehouse_block','mode_of_shipment','product_importance', 'gender', '
# Numerical data
numeric = ['customer_care_calls', 'cost_of_the_product', 'prior_purchases', 'discount_
```

Data Cleansing & Feature Engineering

Reload dataset

In [62]:
```
df_dt = df.copy()
```

Identify missing values

In [63]:
```
df_dt.isna().values.any() # Missing value detection
```

Out[63]: False

In [64]:
```
df_dt.isna().sum()  # Calculate missing values
```

Out[64]:
```
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
is_late                0
dtype: int64
```

Just for making sure that no missing values are found.

Identify duplicated values

In [65]:
```
# Select all duplicate rows based on all columns
df_dt[df_dt.duplicated(keep=False)]
```

Out[65]:

id	warehouse_block	mode_of_shipment	customer_care_calls	customer_rating	cost_of_the_product	prior_

In [66]:
```
# Select all duplicate rows based on selected column
df_dt[df_dt.duplicated(subset=['id'],keep=False)] # Display all duplicated rows based
```

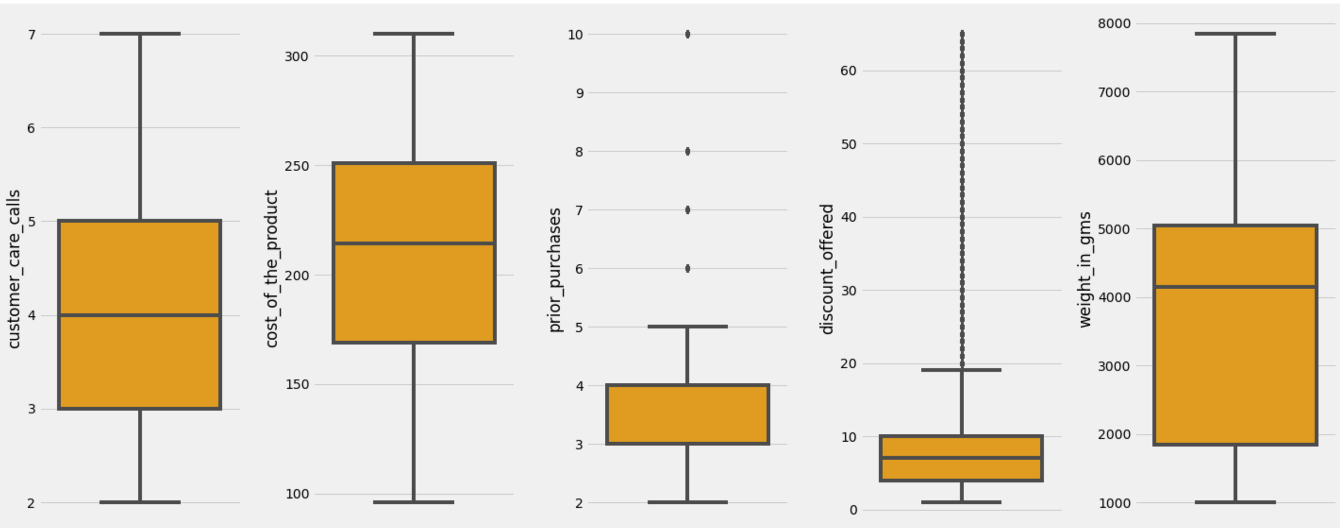
Out[66]:

id	warehouse_block	mode_of_shipment	customer_care_calls	customer_rating	cost_of_the_product	prior_

Luckily, there is no duplicated value in the dataframe.

Identify outliers

In [67]:
```
# Identify using boxplot
plt.figure(figsize=(20,8))
for i in range(0,len(numeric)):
    plt.subplot(1, len(numeric), i+1)
    sns.boxplot(y=df_dt[numeric[i]], color='orange')
    plt.tight_layout()
```



In [68]:
```
# Identify outlier using IQR
for col in numeric:

    # Menghitung nilai IQR
```

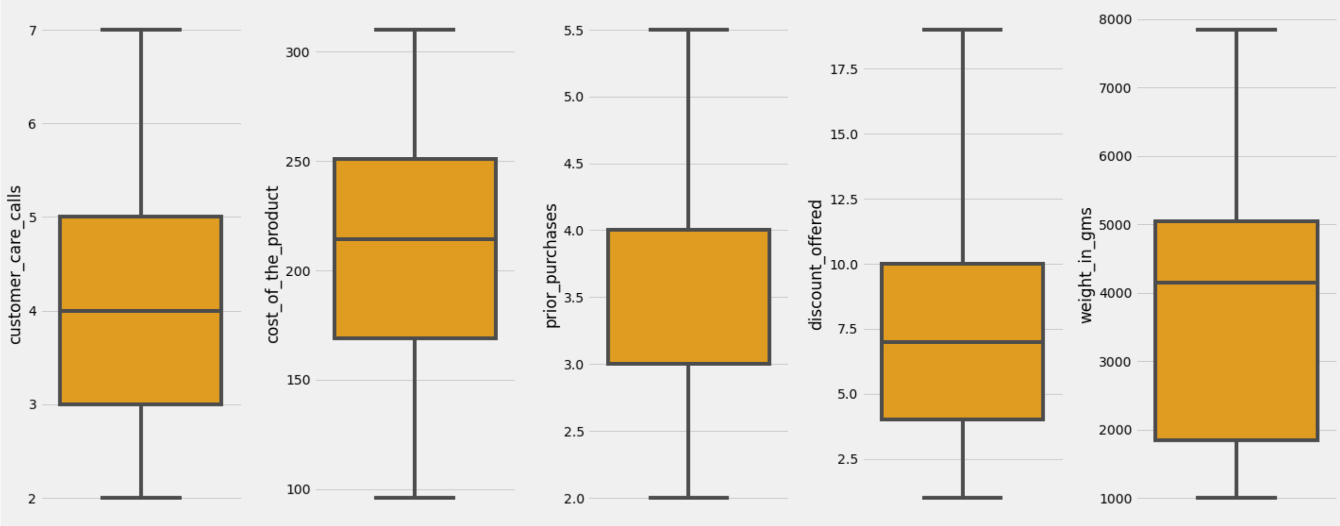
```
    Q1 = df_dt[col].quantile(0.25)
    Q3 = df_dt[col].quantile(0.75)
    IQR = Q3 - Q1

    # Define value
    nilai_min = df_dt[col].min()
    nilai_max = df_dt[col].max()
    lower_lim = Q1 - (1.5*IQR)
    upper_lim = Q3 + (1.5*IQR)

    # Identify low outlier

    if (nilai_min < lower_lim):
        print('Low outlier is found in column',col,'<', lower_lim,'\n')
        #display total low outlier
        print('Total of Low Outlier in column',col, ':', len(list(df_dt[df_dt[col] <
    elif (nilai_max > upper_lim):
        print('High outlier is found in column',col,'>', upper_lim,'\n')
        #display total high outlier
        print('Total of High Outlier in column',col, ':', len(list(df_dt[df_dt[col] >

    else:
        print('Outlier is not found in column',col,'\n')
```

```
Outlier is not found in column customer_care_calls

Outlier is not found in column cost_of_the_product

High outlier is found in column prior_purchases > 5.5

Total of High Outlier in column prior_purchases : 1003

High outlier is found in column discount_offered > 19.0

Total of High Outlier in column discount_offered : 2209

Outlier is not found in column weight_in_gms
```

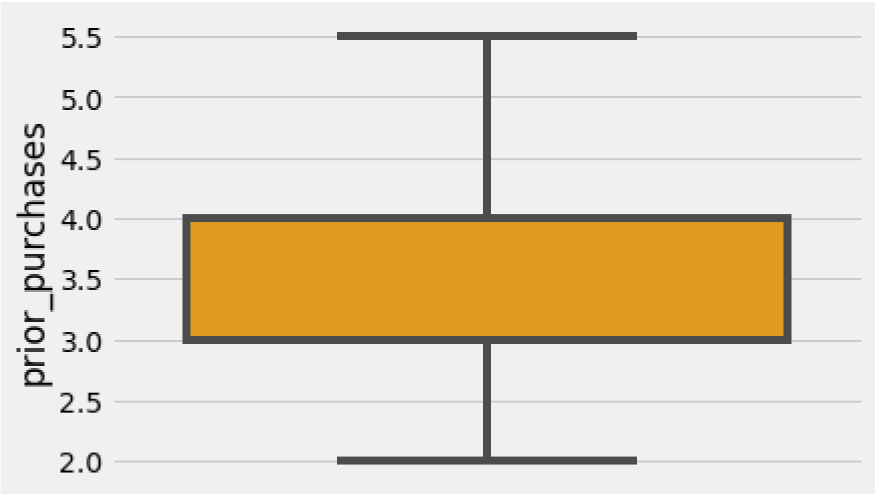
We found outliers in `discount_offered` & `prior_purchases` with almost 30% of data.

In [69]:
```
# We handle outlier with replace the value with upper_bound or lower_bound
for col in ['prior_purchases', 'discount_offered']:
    # Initiate Q1
    Q1 = df_dt[col].quantile(0.25)
    # Initiate Q3
    Q3 = df_dt[col].quantile(0.75)
    # Initiate IQR
    IQR = Q3 - Q1
    # Initiate lower_bound & upper_bound
    lower_bound = Q1 - (IQR * 1.5)
    upper_bound = Q3 + (IQR * 1.5)

    # Filtering outlier & replace with upper_bound or lower_bound
    df_dt[col] = np.where(df_dt[col] >= upper_bound,
                          upper_bound, df_dt[col])
    df_dt[col] = np.where(df_dt[col] <= lower_bound,
                          lower_bound, df_dt[col])
```

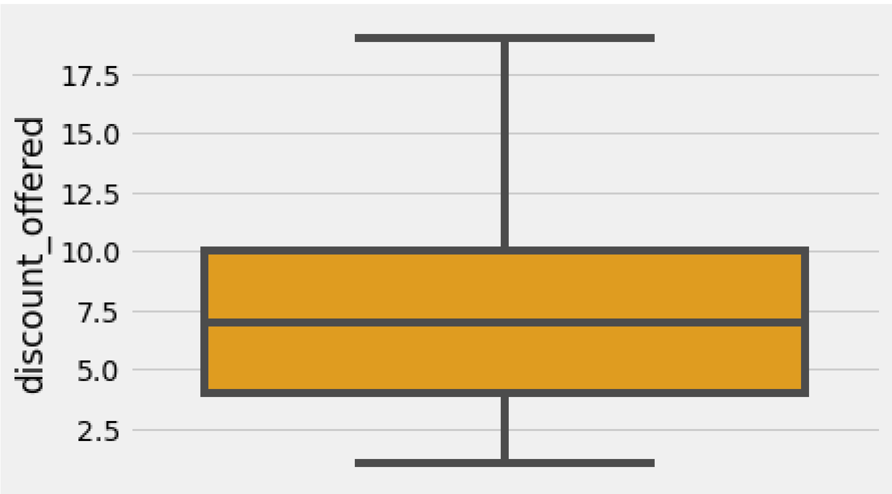
In [70]:
```
# Identify using boxplot
plt.figure(figsize=(20,8))
for i in range(0,len(numeric)):
    plt.subplot(1, len(numeric), i+1)
    sns.boxplot(y=df_dt[numeric[i]], color='orange')
    plt.tight_layout()
```



In [71]:
```
sns.boxplot(y= df_dt['prior_purchases'], color = 'orange', orient = 'h');
```



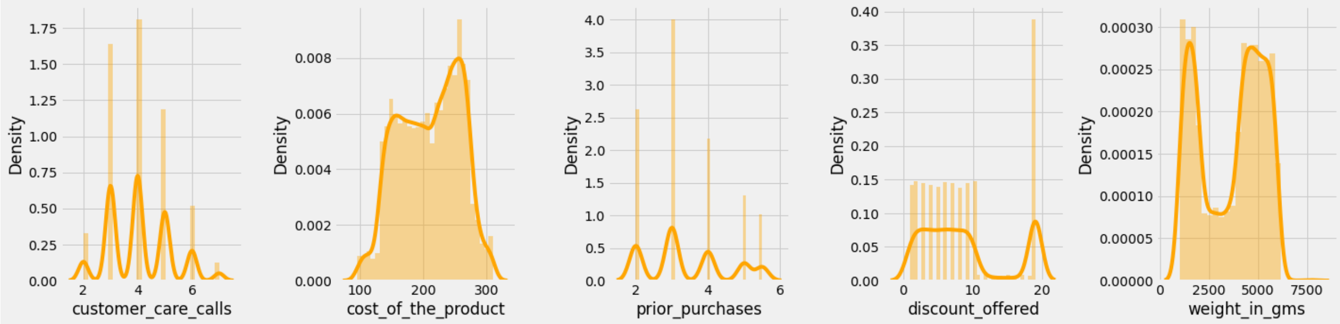
In [72]:
```
sns.boxplot(y= df_dt['discount_offered'], color = 'orange', orient = 'h');
```



We didn't remove the outliers, but replacing with upper bound and lower bound. And we can see in the visualization above, there is no outliers detected.

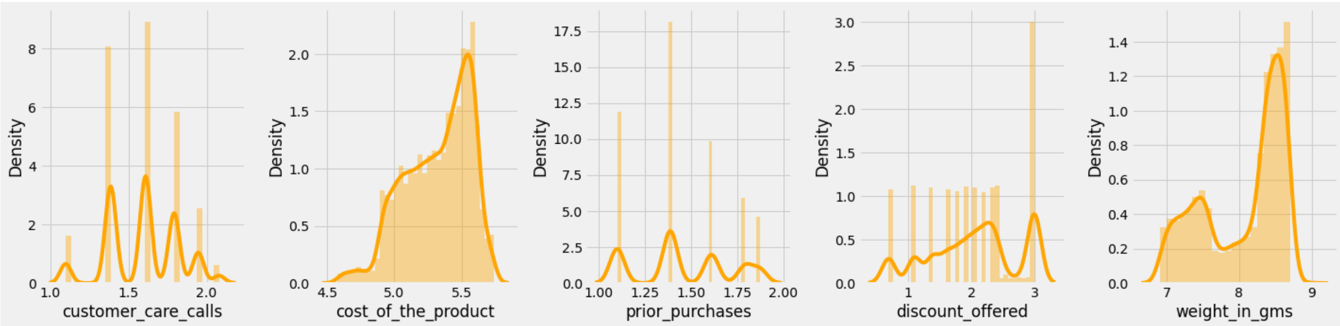
Feature Transformation : Log transform

In [73]:
```
# Check data distribution
plt.figure(figsize=(20,5))
for i in range(0,len(numeric)):
    plt.subplot(1, len(numeric), i+1)
    sns.distplot(df_dt[numeric[i]], color='orange')
    plt.tight_layout()
```



In [74]:
```python
# Apply log transformation
for col in numeric:
    df_dt[col] = (df_dt[col]+1).apply(np.log)
```

In [75]:
```python
# Visualize after log transformation
plt.figure(figsize=(20,5))
for i in range(0,len(numeric)):
    plt.subplot(1, len(numeric), i+1)
    sns.distplot(df_dt[numeric[i]], color='orange')
    plt.tight_layout()
```



Feature Scaling : Standardization

In [76]:
```python
# Apply standardization
for col in numeric:
    df_dt[col]= StandardScaler().fit_transform(df_dt[col].values.reshape(len(df_dt),
```

In [77]:
```python
df_dt.describe()
```

Out[77]:

	id	customer_care_calls	customer_rating	cost_of_the_product	prior_purchases	discount_o
count	10999.00000	1.099900e+04	10999.000000	1.099900e+04	1.099900e+04	1.09990
mean	5500.00000	-6.375198e-15	2.990545	-6.128706e-15	-3.886356e-15	-5.67272
std	3175.28214	1.000045e+00	1.413603	1.000045e+00	1.000045e+00	1.00004
min	1.00000	-2.177630e+00	1.000000	-3.086779e+00	-1.387728e+00	-1.94765
25%	2750.50000	-9.146463e-01	2.000000	-7.773403e-01	-2.636003e-01	-6.23428
50%	5500.00000	6.500001e-02	3.000000	1.892606e-01	-2.636003e-01	5.58237
75%	8249.50000	8.654293e-01	4.000000	8.428454e-01	6.083412e-01	5.16055
max	10999.00000	2.128413e+00	5.000000	1.708704e+00	1.633539e+00	1.38005

Feature Selection : Chi squared method

In [78]:
```python
# Selection for categorial feature
# Import module
from scipy.stats import chi2_contingency

category = ['warehouse_block','mode_of_shipment','product_importance',
            'gender','customer_rating']
chi2_check = []
```

```
# Iteration
for col in category:
    # If pvalue < 0.05
    if chi2_contingency(pd.crosstab(df_dt['is_late'], df_dt[col]))[1] < 0.05 :
        chi2_check.append('Reject Null Hypothesis')
    # If pvalue > 0.05
    else :
        chi2_check.append('Fail to Reject Null Hypothesis')

# Make the result into dataframe
res = pd.DataFrame(data = [category, chi2_check]).T
# Rename columns
res.columns = ['Column', 'Hypothesis']
res
```

Out[78]:

	Column	Hypothesis
0	warehouse_block	Fail to Reject Null Hypothesis
1	mode_of_shipment	Fail to Reject Null Hypothesis
2	product_importance	Reject Null Hypothesis
3	gender	Fail to Reject Null Hypothesis
4	customer_rating	Fail to Reject Null Hypothesis

In [79]:

```
# Adjusted P-Value use the Bonferroni-adjusted method

# Initiate empty dictionary
check = {}
# Iteration for product_importance column
for i in res[res['Hypothesis'] == 'Reject Null Hypothesis']['Column']:
    # One hot encoding product_importance column
    dummies = pd.get_dummies(df_dt[i])
    # Initiate Bonferroni-adjusted formula
    bon_p_value = 0.05/df_dt[i].nunique()
    for series in dummies:
        if chi2_contingency(pd.crosstab(df_dt['is_late'], dummies[series]))[1] < bon_
            check['{}-{}'.format(i, series)] = 'Reject Null Hypothesis'
        else :
            check['{}-{}'.format(i, series)] = 'Fail to Reject Null Hypothesis'
# Make the result into dataframe
res_chi_ph = pd.DataFrame(data=[check.keys(), check.values()]).T
# Rename the columns
res_chi_ph.columns = ['Pair', 'Hypothesis']
res_chi_ph
```

Out[79]:

	Pair	Hypothesis
0	product_importance-high	Reject Null Hypothesis
1	product_importance-low	Fail to Reject Null Hypothesis
2	product_importance-medium	Fail to Reject Null Hypothesis

From the result above, `product_importance` with **high** category has a correlation with our target.

Feature Encoding : One hot encoding

In [80]:

```
# one hot encoding feature product_importance and keep high category
onehots = pd.get_dummies(df_dt['product_importance'], prefix = 'product_importance')
df_dt = df_dt.join(onehots)

# drop all categorical columns & 'id, except product_importance_high
df_dt.drop(columns=['warehouse_block','gender','mode_of_shipment',
                    'product_importance', 'product_importance_low',
                    'product_importance_medium','id'], inplace = True)
```

```
# check dataframe after encoding
df_dt.info()
```

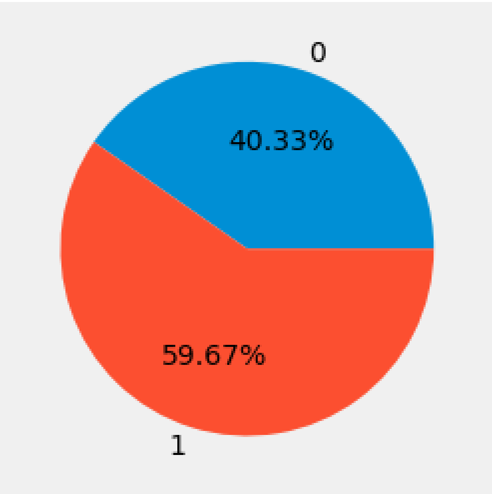
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   customer_care_calls      10999 non-null  float64
 1   customer_rating          10999 non-null  int64  
 2   cost_of_the_product      10999 non-null  float64
 3   prior_purchases          10999 non-null  float64
 4   discount_offered         10999 non-null  float64
 5   weight_in_gms            10999 non-null  float64
 6   is_late                  10999 non-null  int64  
 7   product_importance_high  10999 non-null  uint8  
dtypes: float64(5), int64(2), uint8(1)
memory usage: 612.4 KB
```

Exploratory Data Analysis (EDA)

In [81]:
```
# Copy dataset
df_eda = df.copy()
```

Target Visualization

In [82]:
```
delay = pd.DataFrame(df_eda.groupby(['is_late'])['id'].count()/len(df_eda)).reset_ind
plt.pie(delay['id'],labels=delay['is_late'],autopct='%.2f%%');
```



The class of target looks balance.

Descriptive Statistic

Categorical values

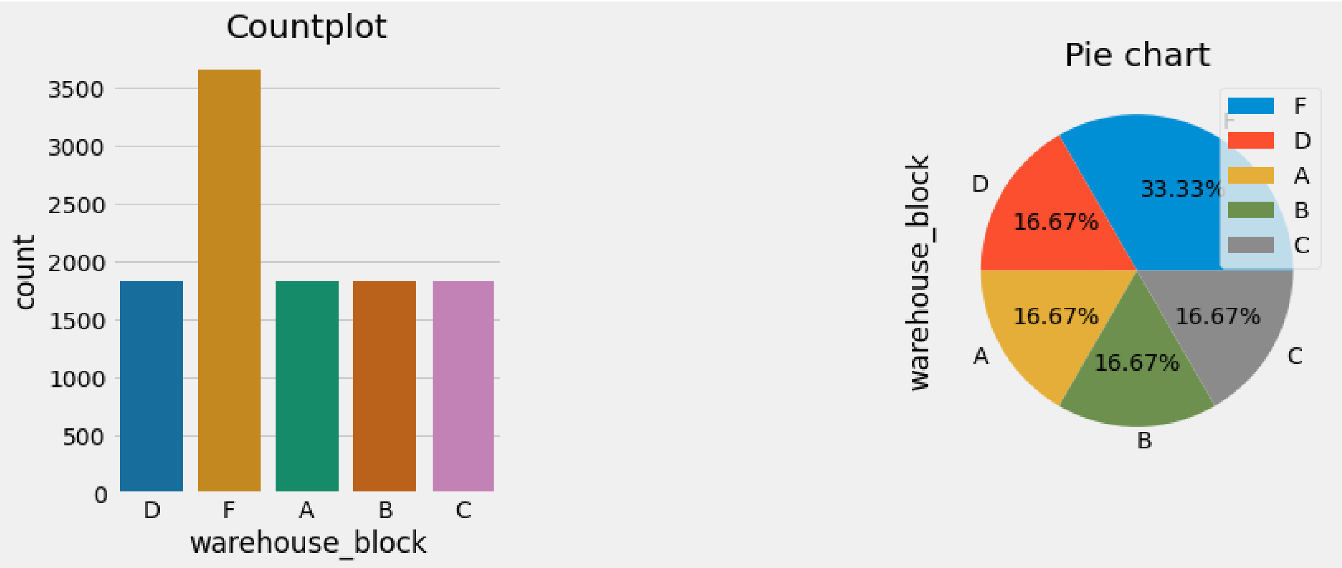
In [83]:
```
# for categorical column
for col in categorical:
    print('Value count kolom', col, ':')
    print(df_eda[col].value_counts())
    print()
```

```
Value count kolom warehouse_block :
F    3666
D    1834
A    1833
B    1833
C    1833
Name: warehouse_block, dtype: int64
```

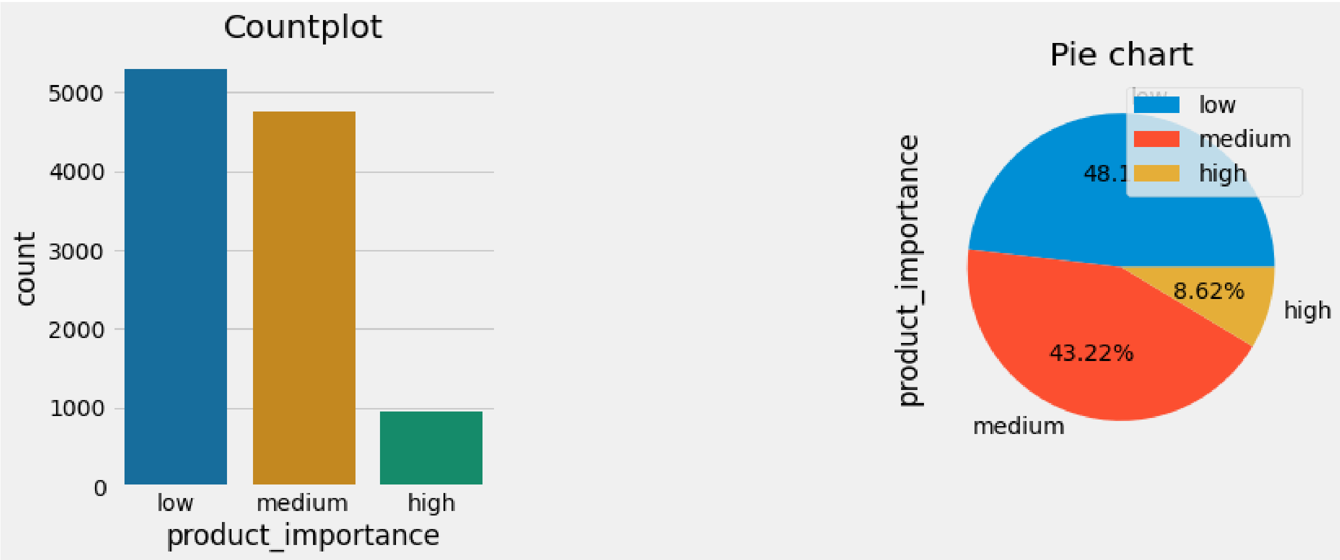
```
Value count kolom mode_of_shipment :
Ship       7462
Flight     1777
Road       1760
Name: mode_of_shipment, dtype: int64

Value count kolom product_importance :
low        5297
medium     4754
high        948
Name: product_importance, dtype: int64

Value count kolom gender :
F    5545
M    5454
Name: gender, dtype: int64

Value count kolom is_late :
1    6563
0    4436
Name: is_late, dtype: int64

Value count kolom customer_rating :
3    2239
1    2235
4    2189
5    2171
2    2165
Name: customer_rating, dtype: int64
```

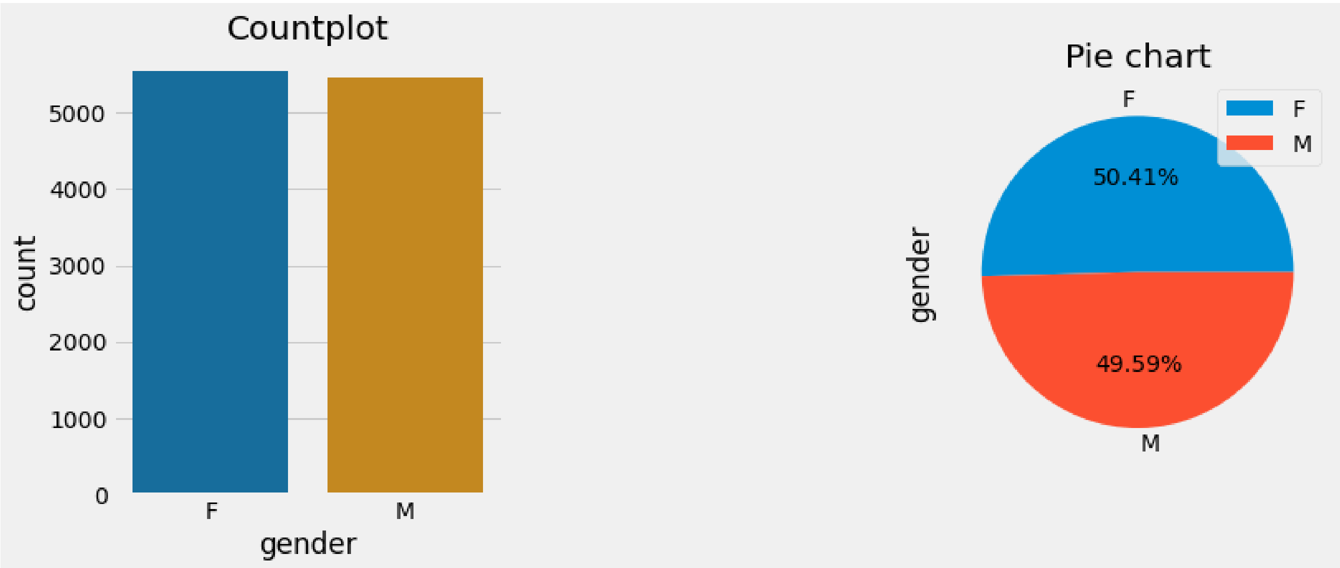
In [84]:

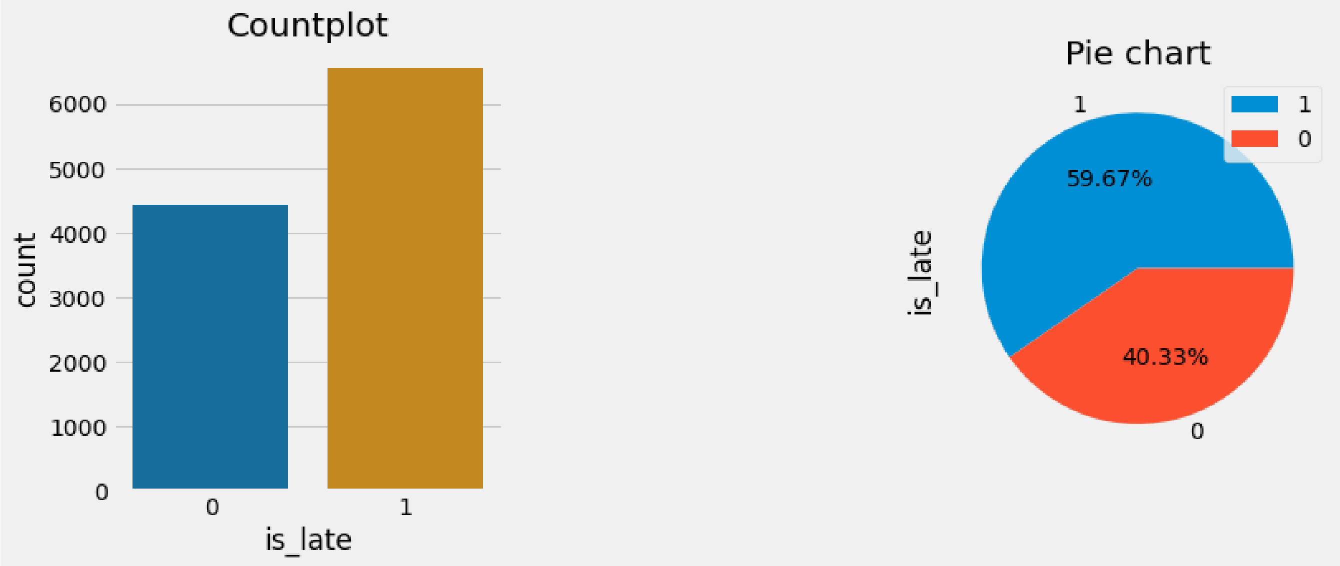
```
# Plot categorical columns
for col in categorical:
    plt.figure(figsize=(15, 5))

    plt.subplot(141);
    sns.countplot(df_eda[col], palette = 'colorblind', orient='v');
    plt.title('Countplot')
    plt.tight_layout();


    plt.subplot(143);
    df_eda[col].value_counts().plot.pie(autopct='%1.2f%%');
    plt.title('Pie chart')
    plt.legend()
```

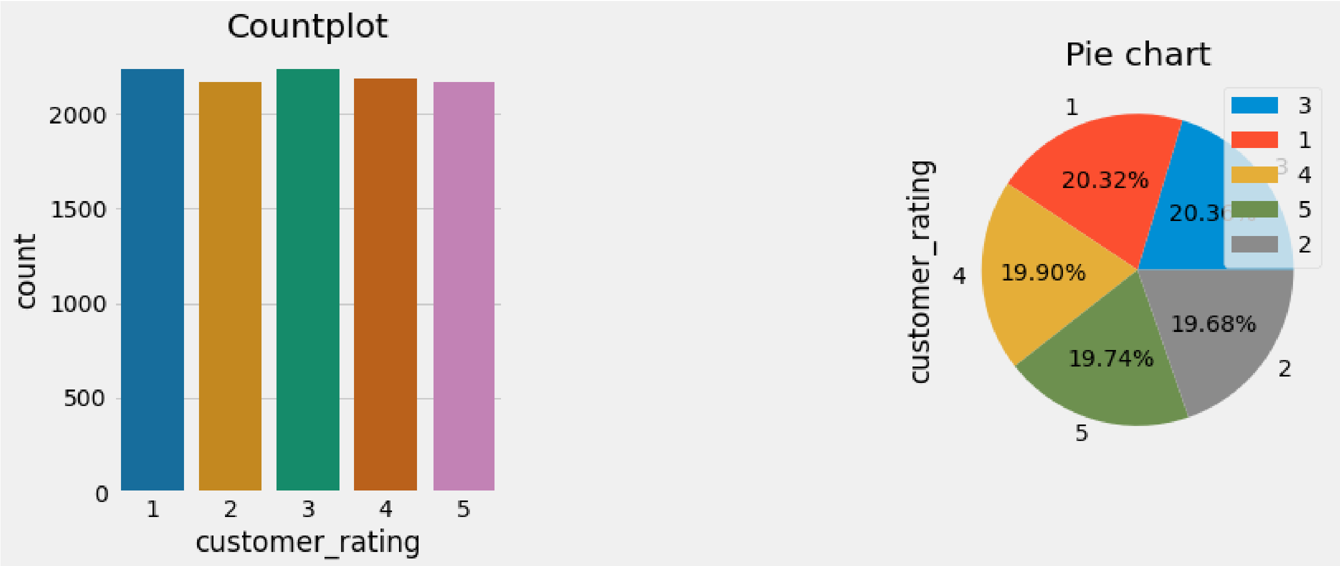



Countplot
count
7000
6000
5000
4000
3000
2000
1000
0
Flight
Ship
Road
mode_of_shipment
Pie chart
mode_of_shipment
Ship
Flight
Road
Ship
67.84%
16.00%
16.16%
Road
Flight


Countplot
count
5000
4000
3000
2000
1000
0
low
medium
high
product_importance
Pie chart
product_importance
low
medium
high
8.62%
43.22%
high
medium


Countplot
count
5000
4000
3000
2000
1000
0
F
M
gender
Pie chart
gender
F
F
M
50.41%
49.59%
M





Summary :

- **Warehouse_Block** has 5 unique values and dominated with `Warehouse_block_f`.
- **Mode_of_Shipment** has 3 unique values and mostly used ship.
- **Product_importance** has 3 unique values and mostly priority of products are low.
- Female customers are often shopping than male.

Numeric values

In [85]:
```
df_eda[numeric].describe()
```

Out[85]:

	customer_care_calls	cost_of_the_product	prior_purchases	discount_offered	weight_in_gms
count	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	4.054459	210.196836	3.567597	13.373216	3634.016729
std	1.141490	48.063272	1.522860	16.205527	1635.377251
min	2.000000	96.000000	2.000000	1.000000	1001.000000
25%	3.000000	169.000000	3.000000	4.000000	1839.500000
50%	4.000000	214.000000	3.000000	7.000000	4149.000000
75%	5.000000	251.000000	4.000000	10.000000	5050.000000
max	7.000000	310.000000	10.000000	65.000000	7846.000000

Summary :

- Distribution of **customer_care_calls**, **Customer_rating**, **Cost_of_the_Product**, **Prior_purchases** look normal, beacuse the mean and the median are close, while **discount_offered** and **weight_in_grams** are indicated skewed.

Correlation Heatmap

In [86]:
```python
plt.figure(figsize=(7,6));
sns.heatmap(df_eda.corr(), annot = True, fmt = '.2f', cmap = 'Reds');
```

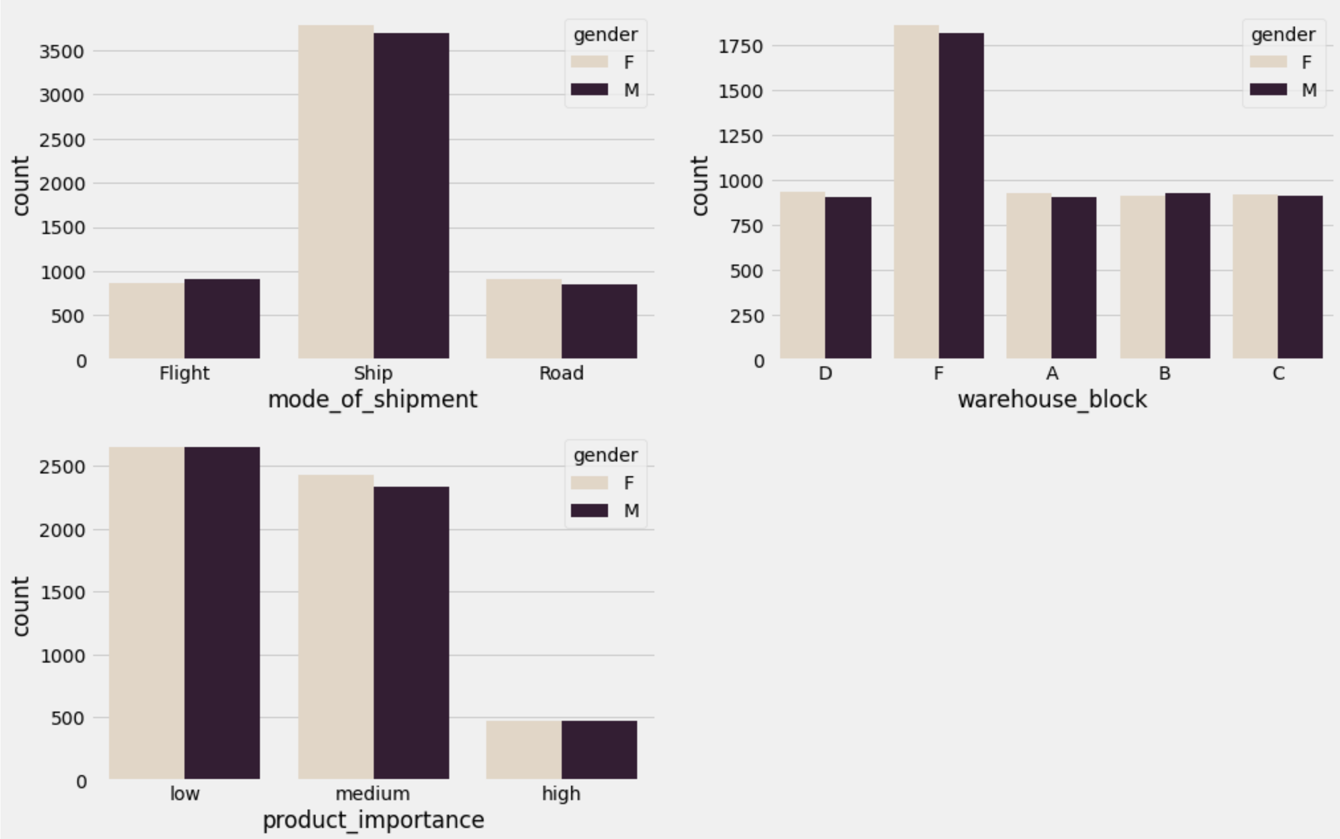
Based on the *Correlation heatmap* above :

1. Target *is_late* has a moderate positive correlation with *discount_offered* & weak negative correlation with *weight_in_gms*.
2. Feature *customer_care_calss* has a weak positive correlation with *cost_of_the_product* and negative correlation with *weight_in_gms*.
3. Feature *discount_offered* has a moderate negative correlation with *weight_in_gms*.

Categorical - Categorical

Based on Gender

In [87]:
```python
i=1
plt.figure(figsize=(15,10))
for col in ['mode_of_shipment', 'warehouse_block', 'product_importance']:
    plt.subplot(2,2,i)
    sns.countplot(df_eda[col], hue=df_eda['gender'], palette="ch:.25")
    i+=1
```

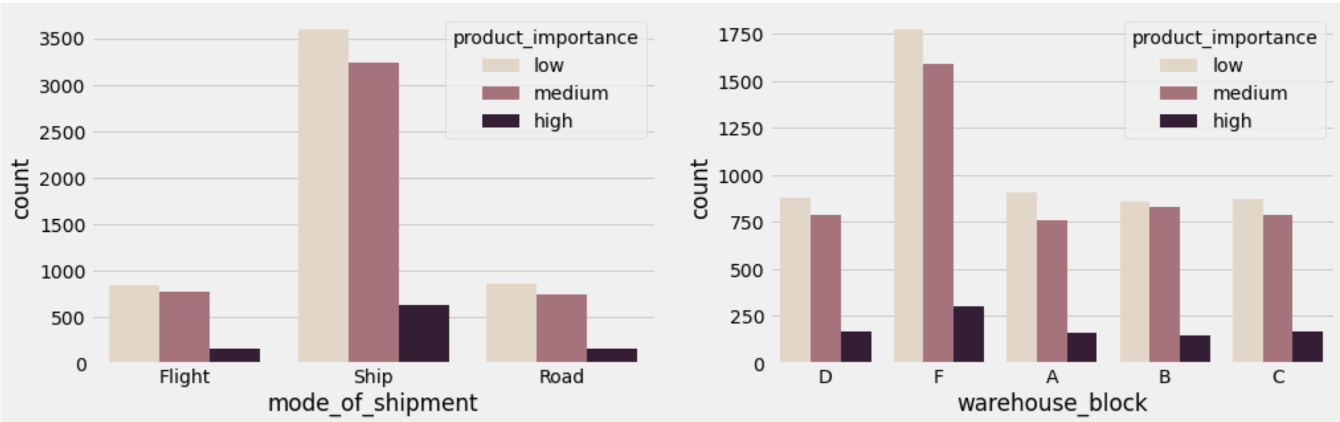


Summary :

- Total parcels of female customers in the warehouse_block are more dominant than male customers, except in warehouse_block B.

Based on Product Importance

In [88]:
```
i=1
plt.figure(figsize=(15,10))
for col in ['mode_of_shipment', 'warehouse_block']:
    plt.subplot(2,2,i)
    sns.countplot(df_eda[col], hue=df_eda['product_importance'], palette="ch:.25")
    i+=1
```

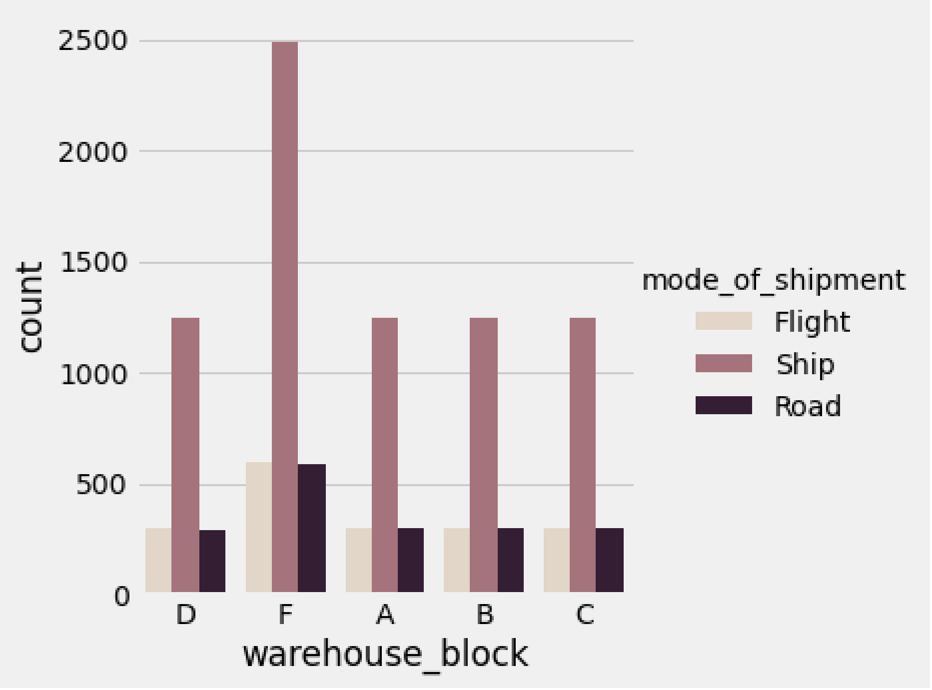


Summary :

- Mostly high & low priority parcels used ship.

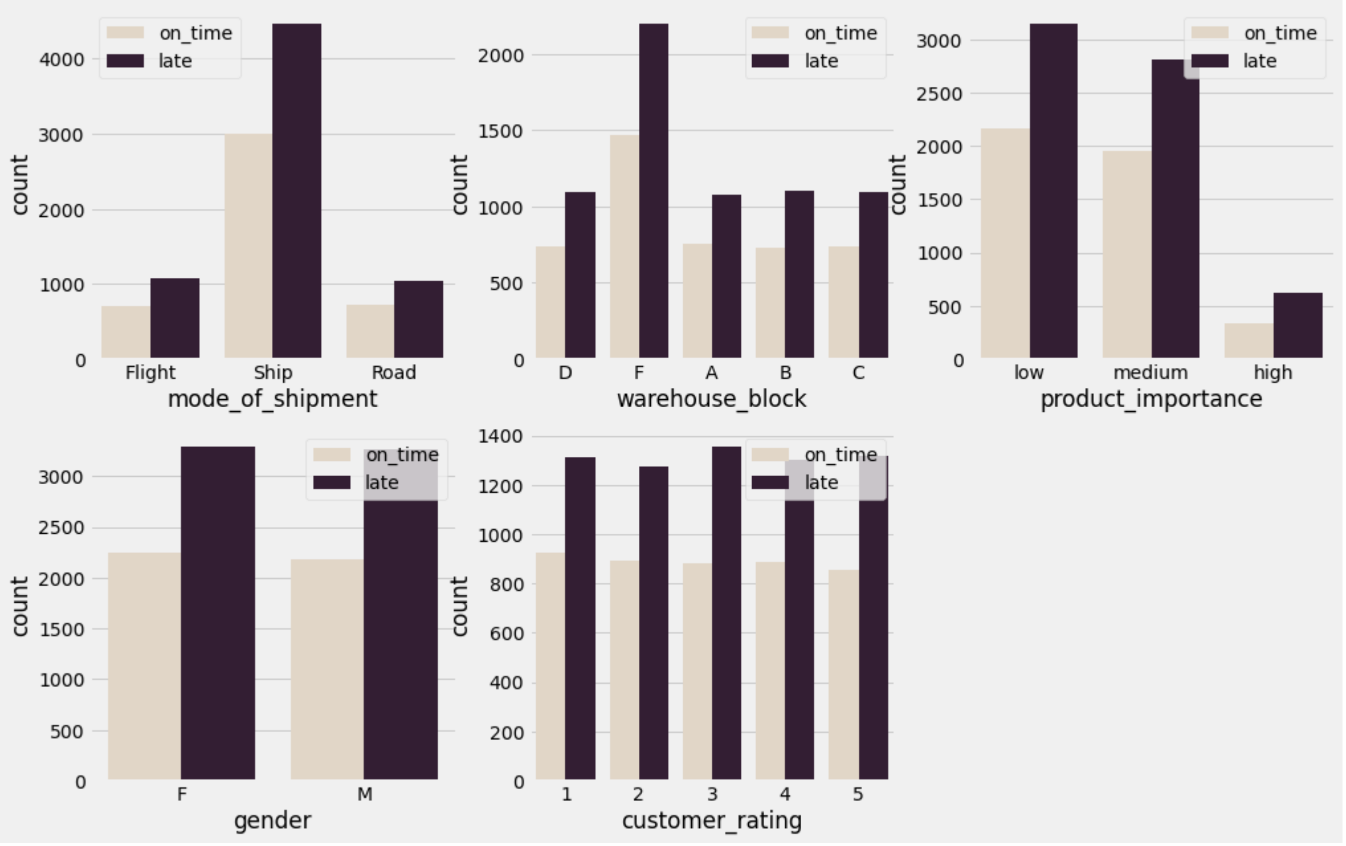
Warehouse block - Mode of Shipment

In [89]:
```
sns.catplot(x="warehouse_block", kind="count", hue='mode_of_shipment',
            palette="ch:.25", data=df_eda);
```



Based on target 'is_late'

In [90]:
```
i=1
plt.figure(figsize=(15,10))
for col in ['mode_of_shipment', 'warehouse_block', 'product_importance',
            'gender','customer_rating']:
    plt.subplot(2,3,i)
    sns.countplot(df_eda[col], hue=df_eda['is_late'], palette="ch:.25")
    i+=1
    plt.legend(['on_time','late']);
```



Summary :

- Most of parcels are stored in warehouse_block F.
- The ship contributes the most late delivery.
- Most of parcels in all shipment priority are delivered late.

Machine Learning Modelling & Evaluation

Separate feature & target column

In [91]:
```
# Inititate feature & target
X = df_dt.drop(columns = 'is_late')
y = df_dt['is_late']
```

Split train & test data

In [92]:
```
# Split Train & Test Data
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.30, random_state=42
```

Fit & Evaluation Model

In [93]:
```
# Create function to fit model & model evaluation
def fit_evaluation(Model, Xtrain, ytrain, Xtest, ytest):
    model = Model # initiate model
    model.fit(Xtrain, ytrain) # fit the model
    y_pred = model.predict(Xtest)
    y_pred_train = model.predict(Xtrain)
    train_score = model.score(Xtrain, ytrain) # Train Accuracy
    test_score = model.score(Xtest, ytest)    # Test Accuracy
    fpr, tpr, thresholds = roc_curve(ytest, y_pred, pos_label=1)
    AUC = auc(fpr, tpr) # AUC
    return round(train_score,2), round(test_score,2), round(precision_score(ytest, y_
           round(recall_score(ytrain, y_pred_train),2),round(recall_score(ytest, y_pr
           round(f1_score(ytest, y_pred),2), round(AUC,2)
```

Default Parameter

In [94]:
```
# Inititate algorithm
lr = LogisticRegression(random_state=42)
dt = DecisionTreeClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)
knn = KNeighborsClassifier(n_neighbors=5)
svc = SVC(random_state=42)

# Create function to make the result as dataframe
def model_comparison_default(X,y):

    # Logistic Regression
    lr_train_score, lr_test_score, lr_pr, lrtr_re, lrte_re, lr_f1, lr_auc = fit_evalu
        lr, Xtrain, ytrain, Xtest, ytest)
    # Decision Tree
    dt_train_score, dt_test_score, dt_pr, dttr_re, dtte_re, dt_f1, dt_auc = fit_evalu
        dt, Xtrain, ytrain, Xtest, ytest)
    # Random Forest
    rf_train_score, rf_test_score, rf_pr, rftr_re, rfte_re, rf_f1, rf_auc = fit_evalu
        rf, Xtrain, ytrain, Xtest, ytest)
    # KNN
    knn_train_score, knn_test_score, knn_pr, knntr_re, knnte_re, knn_f1, knn_auc = fi
        knn, Xtrain, ytrain, Xtest, ytest)
    # SVC
    svc_train_score, svc_test_score, svc_pr, svctr_re, svcte_re, svc_f1, svc_auc = fi
        svc, Xtrain, ytrain, Xtest, ytest)


    models = ['Logistic Regression','Decision Tree','Random Forest',
              'KNN','SVC']
    train_score = [lr_train_score, dt_train_score, rf_train_score,
```

```
                    knn_train_score, svc_train_score]
    test_score = [lr_test_score, dt_test_score, rf_test_score,
                  knn_test_score, svc_test_score]
    precision = [lr_pr, dt_pr, rf_pr, knn_pr, svc_pr]
    recall_train = [lrtr_re, dttr_re, rftr_re, knntr_re, svctr_re]
    recall_test = [lrte_re, dtte_re, rfte_re, knnte_re, svcte_re]
    f1 = [lr_f1, dt_f1, rf_f1, knn_f1, svc_f1]
    auc = [lr_auc, dt_auc, rf_auc, knn_auc, svc_auc]

    model_comparison = pd.DataFrame(data=[models, train_score, test_score,
                                          precision, recall_train, recall_test,
                                          f1,auc]).T.rename({0: 'Model',
                                                             1: 'Accuracy_Train',
                                                             2: 'Accuracy_Test',
                                                             3: 'Precision',
                                                             4: 'Recall_Train',
                                                             5: 'Recall_Test',
                                                             6: 'F1 Score',
                                                             7: 'AUC'
                                                                                 },

    return model_comparison
```

In [95]:
```
model_comparison_default(X,y)
```

Out[95]:

	Model	Accuracy_Train	Accuracy_Test	Precision	Recall_Train	Recall_Test	F1 Score	AUC
0	Logistic Regression	0.63	0.63	0.67	0.75	0.75	0.71	0.6
1	Decision Tree	1.0	0.66	0.72	1.0	0.71	0.71	0.64
2	Random Forest	1.0	0.67	0.76	1.0	0.66	0.71	0.67
3	KNN	0.76	0.64	0.72	0.77	0.67	0.69	0.64
4	SVC	0.68	0.66	0.88	0.52	0.51	0.65	0.7

From the result above, only **Logistic Regression and SVC which are neither overfitting nor underfiting**. Logistic Regression has the highest recall. Let's see with tuned parameter.

Hyperparameter

Logistic Regression

In [96]:
```
# List Hyperparameters
penalty = ['l2','l1','elasticnet']
C = [0.0001, 0.001, 0.002] # Inverse of regularization strength; smaller values speci
hyperparameters = dict(penalty=penalty, C=C)

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

# Fitting Model & Evaluation
model.fit(Xtrain, ytrain)
y_pred = model.predict(Xtest)
model.best_estimator_
```

Out[96]:
```
LogisticRegression(C=0.0001, random_state=42)
```

Decision Tree

In [97]:
```
# Let's do hyperparameter tuning using RandomizesearchCV
```

```python
# Hyperparameter lists to be tested
max_depth = list(range(1,10))
min_samples_split = list(range(5,10))
min_samples_leaf = list(range(5,15))
max_features = ['auto', 'sqrt', 'log2']
criterion = ['gini','entropy']
splitter = ['best','random']

# Initiate hyperparameters
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)

# Initiate model
dt_tun = DecisionTreeClassifier(random_state=42)
model = RandomizedSearchCV(dt_tun, hyperparameters, cv=10, scoring='recall',random_st
model.fit(Xtrain, ytrain)
y_pred_tun = model.predict(Xtest)
model.best_estimator_
```

Out[97]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, max_features='sqrt',
                       min_samples_leaf=12, min_samples_split=6,
                       random_state=42)
```

Random Forest

In [98]:
```python
# Initiate hyperparameters
params = {'max_depth':[50],'n_estimators':[100,150],
          'criterion':['gini', 'entropy']}
# Initiate model
model = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                     param_grid=params,scoring='recall', cv=5)
# Fit model
model.fit(Xtrain,ytrain)
y_pred = model.predict(Xtest)
# Get best estimator
model.best_estimator_
```

Out[98]:
```
RandomForestClassifier(max_depth=50, random_state=42)
```

KNN

In [99]:
```python
#List Hyperparameters that we want to tune.
leaf_size = list(range(1,50))
n_neighbors = list(range(1,30))
p=[1,2]

#Convert to dictionary
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)

#Create new KNN object
KNN_2 = KNeighborsClassifier()

#Use RandomizedSearchCV
clf = RandomizedSearchCV(KNN_2, hyperparameters, cv=10, scoring = 'recall')

#Fit the model
best_model = clf.fit(X,y)
# Get best estimator
clf.best_estimator_
```

Out[99]:
```
KNeighborsClassifier(leaf_size=36, n_neighbors=3)
```

SVC

In [100…

```
# Hyperparameter lists to be tested
kernel = ['linear', 'poly', 'rbf', 'sigmoid']
C = [0.0001, 0.001, 0.002]
gamma = ['scale', 'auto']

#Convert to dictionary
hyperparameters = dict(kernel=kernel, C=C, gamma=gamma)

# Initiate model
svc = SVC(random_state=42)
model = RandomizedSearchCV(svc, hyperparameters, cv=5, random_state=42,
                           scoring='recall')

# Fitting Model & Evaluation
model.fit(Xtrain, ytrain)
y_pred = model.predict(Xtest)
model.best_estimator_
```

Out[100]:

```
SVC(C=0.0001, kernel='linear', random_state=42)
```

Tuned Parameter

In [103…

```
# Inititate best estimator
lr_tune = LogisticRegression(C=0.0001, random_state=42)
dt_tune = DecisionTreeClassifier(criterion='entropy', max_depth=4, max_features='sqrt
                       min_samples_leaf=12, min_samples_split=6,
                       random_state=42)
rf_tune = RandomForestClassifier(max_depth=50, random_state=42)

knn_tune = KNeighborsClassifier(leaf_size=24, n_neighbors=3, p=1)

svc_tune = SVC(C=0.0001, kernel='linear', random_state=42)

# Create function to make the result as dataframe
def model_comparison_tuned(X,y):

    # Logistic Regression
    lr_train_score, lr_test_score, lr_pr, lrtr_re, lrte_re, lr_f1, lr_auc = fit_evalu
        lr_tune, Xtrain, ytrain, Xtest, ytest)
    # Decision Tree
    dt_train_score, dt_test_score, dt_pr, dttr_re, dtte_re, dt_f1, dt_auc = fit_evalu
        dt_tune, Xtrain, ytrain, Xtest, ytest)
    # Random Forest
    rf_train_score, rf_test_score, rf_pr, rftr_re, rfte_re, rf_f1, rf_auc = fit_evalu
        rf_tune, Xtrain, ytrain, Xtest, ytest)
    # KNN
    knn_train_score, knn_test_score, knn_pr, knntr_re, knnte_re, knn_f1, knn_auc = fi
        knn_tune, Xtrain, ytrain, Xtest, ytest)
    # SVC
    svc_train_score, svc_test_score, svc_pr, svctr_re, svcte_re, svc_f1, svc_auc = fi
        svc_tune, Xtrain, ytrain, Xtest, ytest)


    models = ['Logistic Regression','Decision Tree','Random Forest',
              'KNN','SVC']
    train_score = [lr_train_score, dt_train_score, rf_train_score,
                   knn_train_score, svc_train_score]
    test_score = [lr_test_score, dt_test_score, rf_test_score,
                  knn_test_score, svc_test_score]
    precision = [lr_pr, dt_pr, rf_pr, knn_pr, svc_pr]
    recall_train = [lrtr_re, dttr_re, rftr_re, knntr_re, svctr_re]
    recall_test = [lrte_re, dtte_re, rfte_re, knnte_re, svcte_re]
    f1 = [lr_f1, dt_f1, rf_f1, knn_f1, svc_f1]
    auc = [lr_auc, dt_auc, rf_auc, knn_auc, svc_auc]
```

```
    model_comparison = pd.DataFrame(data=[models, train_score, test_score,
                                          precision, recall_train, recall_test,
                                          f1,auc]).T.rename({0: 'Model',
                                                             1: 'Accuracy_Train',
                                                             2: 'Accuracy_Test',
                                                             3: 'Precision',
                                                             4: 'Recall_Train',
                                                             5: 'Recall_Test',
                                                             6: 'F1 Score',
                                                             7: 'AUC'
                                                                                     },

    return model_comparison
```

In [104…

```
model_comparison_tuned(X,y)
```

Out[104]:

	Model	Accuracy_Train	Accuracy_Test	Precision	Recall_Train	Recall_Test	F1 Score	AUC
0	Logistic Regression	0.59	0.6	0.6	1.0	1.0	0.75	0.5
1	Decision Tree	0.67	0.68	0.8	0.62	0.62	0.7	0.69
2	Random Forest	1.0	0.67	0.76	1.0	0.66	0.71	0.67
3	KNN	0.81	0.65	0.72	0.82	0.69	0.7	0.64
4	SVC	0.59	0.6	0.6	1.0	1.0	0.75	0.5

Decision Tree algorithm with hyper-parameter tuning has a good balance between its score, also neither underfitting nor overfitting.

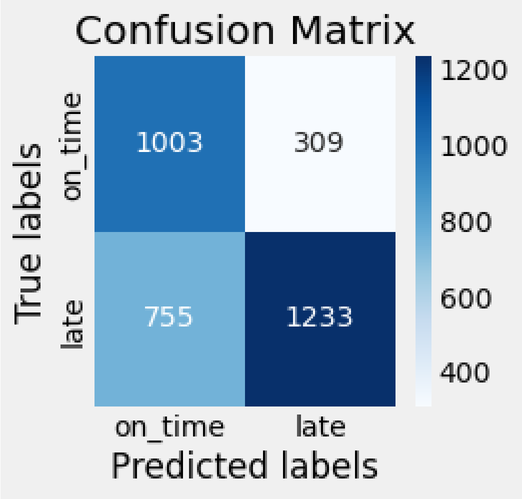
Confusion matrix

In [105…

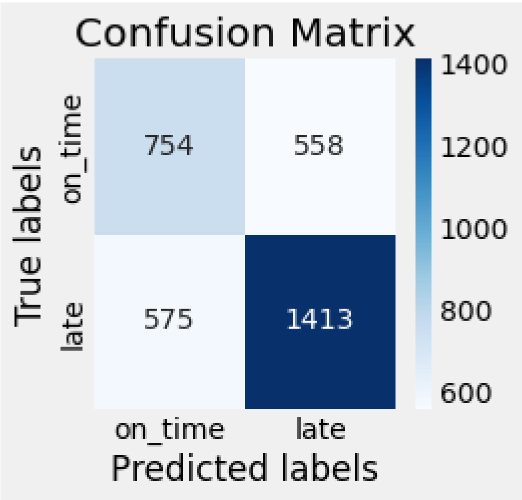
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def get_confusion_matrix(model, X_train, y_train, X_test, y_test, labels=None):
    # Train the model on the training data
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)

    # Create confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Set display labels
    if labels is None:
        labels = ['Negative', 'Positive']

    # Plot the confusion matrix using Seaborn heatmap
    plt.figure(figsize=(3, 3))
    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=labels, yticklabel
    plt.xlabel('Predicted labels')
    plt.ylabel('True labels')
    plt.title('Confusion Matrix')
    plt.show()
```

In [106…

```
# After hyperparameter tuning
get_confusion_matrix(dt_tune, Xtrain, ytrain, Xtest, ytest,labels=['on_time','late'])
```

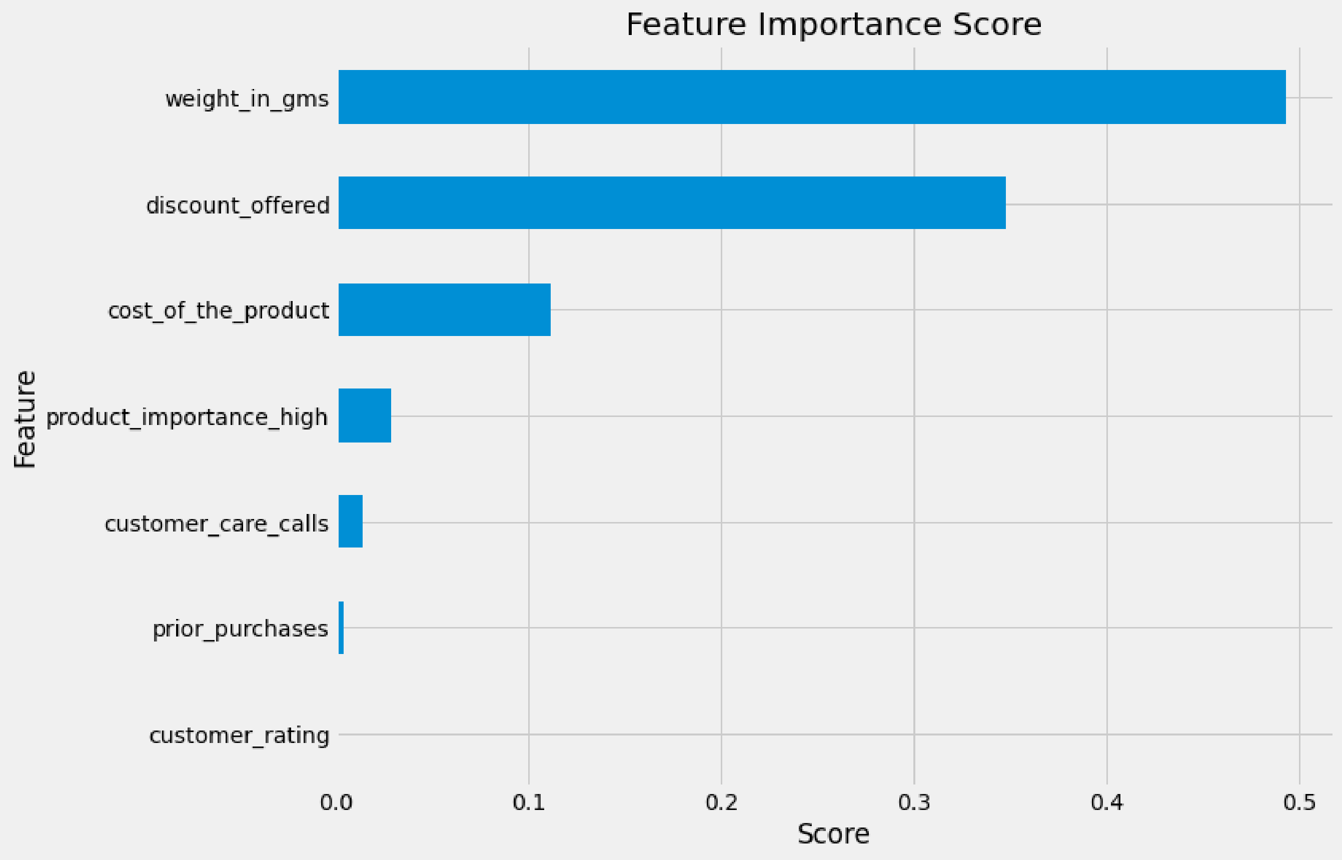



```
In [107…
# Before hyperparameter tuning
get_confusion_matrix(dt, Xtrain, ytrain, Xtest, ytest,labels=['on_time','late'])
```



Feature Importance

```
In [108…
feat_importances = pd.Series(dt_tune.feature_importances_, index=X.columns)
ax = feat_importances.nlargest(25).plot(kind='barh', figsize=(10, 8))
ax.invert_yaxis()

plt.xlabel('Score');
plt.ylabel('Feature');
plt.title('Feature Importance Score');
```



Recommendation for E-Commerce :

- The operation team should add more manpower when there is a sale program, especially for the discount more than 10% and the parcel weight is 1 - 4 Kg.
- The parcel should not be centralized in the warehouse block F, so that the handling is not too crowded which can cause the late shipment.
- Adding more features can imporve model performance, such as delivery time estimation, delivery date, customer address, and courier.

In [ ]:

Name of the Project- Product Shipment Delivered on time or not ?

The E-Commerce Shipping Problem

By Pooja Keer

Batch- PGA19(THANE)

E-COMMERCE INDUSTRY-

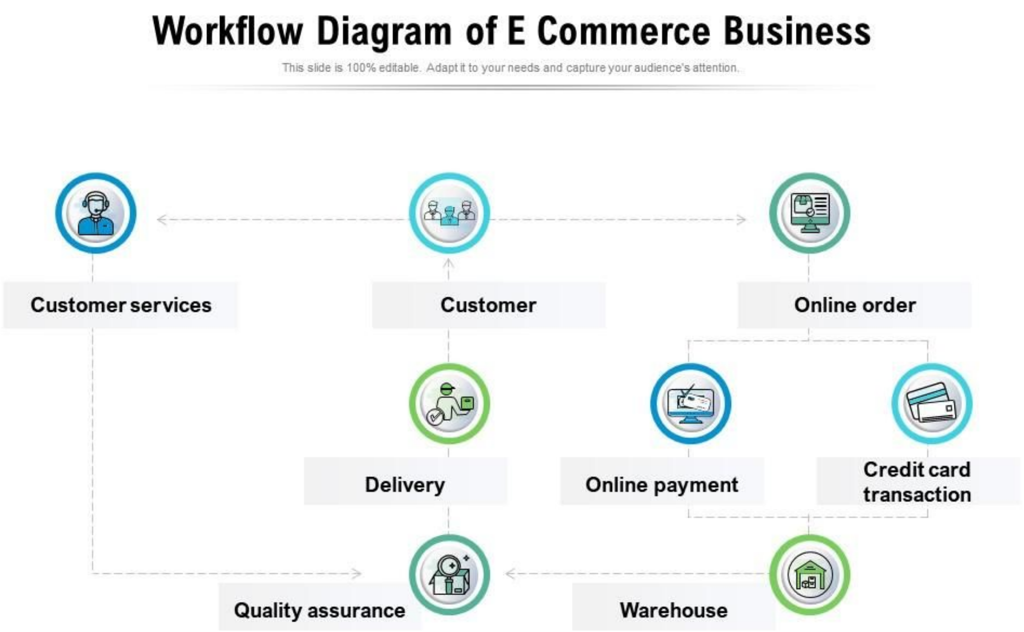
"Ecommerce" or "electronic commerce" is **the trading of goods and services on the internet**.

The e-commerce industry has seen significant growth in recent years, with more and more people shopping online. As a result, there is demand for efficient and reliable shipment services to deliver goods to customers in a timely manner.

One of the major challenges in e-commerce shipment is the management of the delivery process. The shipment process involves multiple stages, from receiving the order to packaging the goods and finally delivering them to the customer.

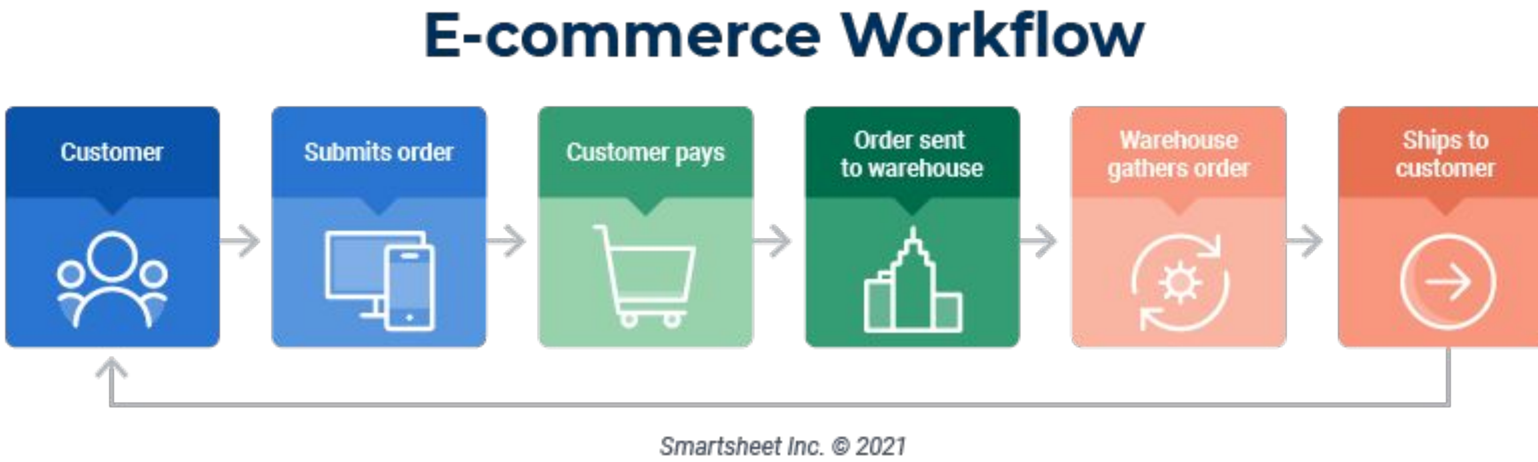
Each stage of the process must be carefully coordinated to ensure timely delivery and minimize the risk of errors or delays.

E-COMMERCE SHIPPING PROCESS



The shipping process involves everything from receiving a customer order to preparing it for last-mile delivery. The shipping process can be broken down into three primary stages:

- Order receiving: make sure items are in stock to fulfill the order
- Order processing: verify order data and make sure it's accurate (e.g., verifying the shipping address)
- Order fulfillment: a picking list is generated and items are picked, packed, and prepared to be shipped



HOW IT AFFECTS INDUSTRY?

- **<u>Customer dissatisfaction</u>**: Customers expect their orders to be delivered on time. If products are not delivered on time, customers may become dissatisfied with the service and the e-commerce platform. This can lead to negative reviews and decreased customer loyalty.
- **<u>Loss of sales:</u>** Delayed product delivery can lead to canceled orders, which can result in a loss of sales for the e-commerce platform. Customers may also choose to shop with competitors who are better at delivering products on time.
- **<u>Increased shipping costs</u>**: If products are not delivered on time, the e-commerce platform may need to pay for expedited shipping or other additional costs to meet the delivery deadline. This can increase the shipping costs, which can negatively impact the company's bottom line.
- **<u>Reputation damage</u>**: The timely delivery of products is critical to maintaining the e-commerce platform's reputation. If products are not delivered on time, it can damage the company's reputation and lead to decreased trust from customers.

WHAT COULD BE BENEFITS OF DELIVERING PRODUCT ON TIME ?

- **Increased customer satisfaction**: Timely delivery of products is essential for meeting customer expectations and delivering a positive shopping experience.
- **Improved brand reputation**: Consistently delivering products on time can help build a positive brand reputation for the e-commerce platform.
- **Increased efficiency and cost savings**: An efficient logistics system that ensures timely delivery can help reduce shipping and handling costs, reduce product returns and exchanges, and streamline order fulfillment processes.
- **Competitive advantage**: Timely delivery of products can be a competitive advantage for e-commerce platforms in a crowded market.
- **Improved operational performance**: Delivering products on time can help e-commerce platforms improve their operational performance.

Solution towards the business problem

- I have worked on the dataset called e-commerce shipping Data of about 11000 of records having features as below -
- **ID** : ID Number of Customers.
- **Warehouse block** : The Company have big Warehouse which is divided in to block such as A,B,C,D,E.
- **Mode of shipment** :The Company Ships the products in multiple way such as Ship, Flight and Road.
- **Customer care calls** : The number of calls made from enquiry for enquiry of the shipment.
- **Customer rating** : The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best).
- **Cost of the product** : Cost of the Product in US Dollars.
- **Prior purchases** : The Number of Prior Purchase.
- **Product importance** : The company has categorized the product in the various parameter such as low, medium, high.
- **Gender** : Male and Female.
- **Discount offered** : Discount offered on that specific product.
- **Weight in gms** : It is the weight in grams.
- **Reached on time** : It is the target variable, where 1 Indicates that the product has NOT reached on time and 0 indicates it has reached on time.

The following steps are carried out –

1. Data Preprocessing
2. Data Visualization
3. Model Fitting
4. Prediction
5. Hyperparameter Tuning

1.Data Pre-processing-

—Imported file in CSV format by renaming the target column 'Reached.on.Time_Y.N':'is_late'.

—Obtained the shape of dataset i.e (10999 rows , 12 columns) by changing the all columns into lower case as it is case sensitive.

—Described the data as info below & separated the numeric & categorical columns

Dataframe has 10999 rows and 12 columns.

1. No missing values are found.
2. There are only 2 data types, integer and object.
3. Classification target is_late and others we call features.

2.Data Visualization-

—Data Cleansing done by checking the missing values & duplicates in dataset

—No any missing values/duplicates are found in data.

—Identification of outlier was carried out for numeric columns using boxplot & IQR (interquartile range) method where prior_purchase column has identified outlier which is replaced by the upper & lower bound of itself.

—so , now no outlier is identified in any numeric column.

3.Featuring Engineering-

—Log transformation is carried out for numeric columns in which the plots are normalized i.eskewed to normal distribution & Numeric Values are standardized using StandardScaler

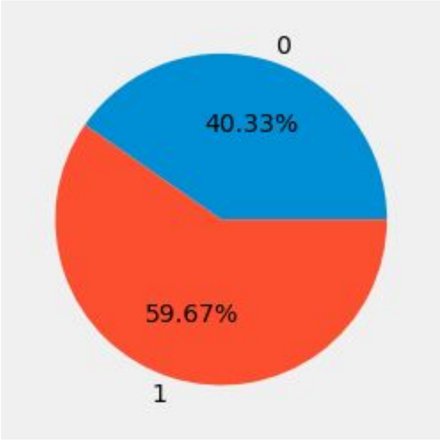
—Feature selection is carried-out by ‘ CHI-SQUARE METHOD’ in which the categorical column who ‘rejects the null hypothesis ‘ is the important feature for further feature encoding.

—In this, the Product_Importance column rejected the null hypothesis ,thus its the feature importance for the data & other columns 'warehouse_block',' mode_of_shipment','gender','customer_rating' “Failed to reject the null hypothesis”.

—Further, one-hot encoding is done(feature encoding)by dropping the categorical column ‘who failed to reject the null hypothesis’.

4.Exploratory Data Analysis-

—For carrying out EDA, the copy of data is used where 'TARGET VISUALISATION' is done & target of class looks balanced.

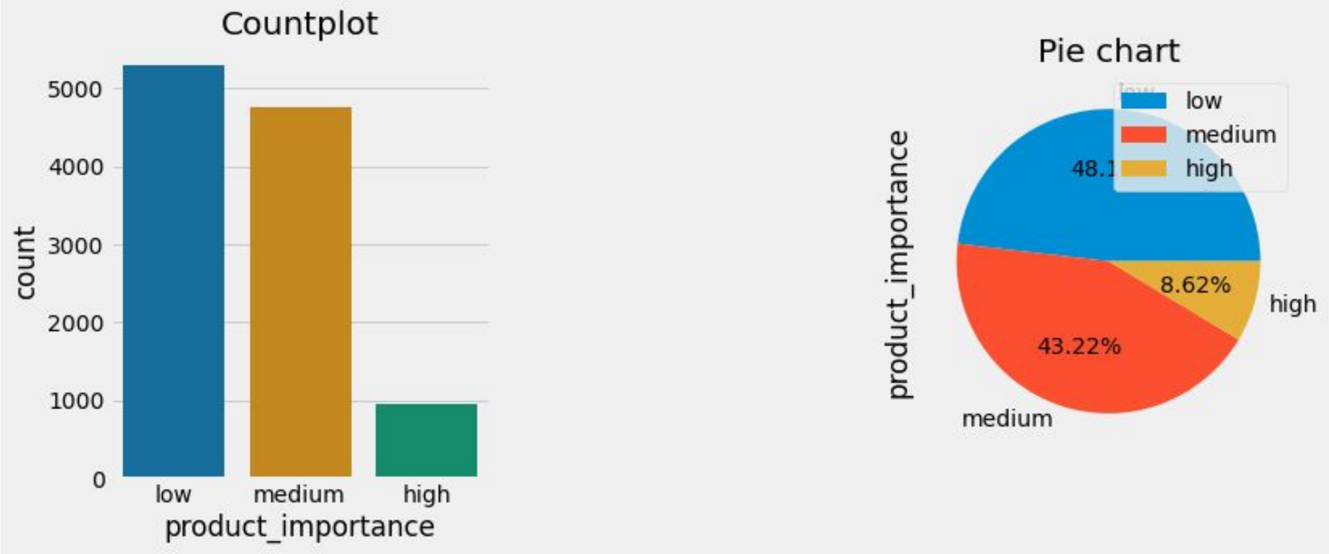


—Descriptive Statistic is carried out for categorical column to get the value counts of each column & plotted the countplot as shown below-

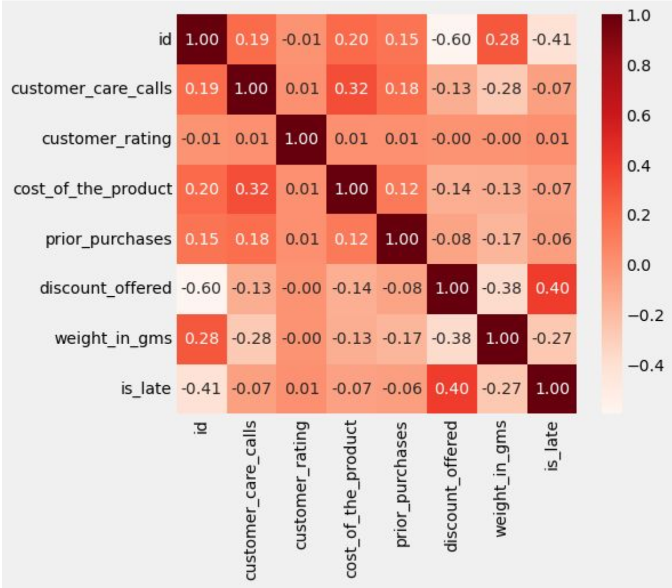
I) **Warehouse_Block** has 5 unique values and dominated with `Warehouse_block_f`.

II)Mode_of_Shipment has 3 unique values and mostly used ship.

III)Product_importance has 3 unique values and mostly priority of products are low.



- After doing EDA of numeric ,Distribution of **customer_care_calls**, **Customer_rating**, **Cost_of_the_Product**, **Prior_purchases** is normal, because the mean and the median are close, while **discount_offered** and **weight_in_grams** are indicated skewed.
- The Correlation Heatmap is obtained for studying the correlation with target variable as follows-



<u>Based on the *Correlation heatmap* above :</u>

1. Target *is_late* has a moderate positive correlation with *discount_offered* & weak negative correlation with *weight_in_gms*.
2. Feature *customer_care_calss* has a weak positive correlation with *cost_of_the_product* and negative correlation with*weight_in_gms*.
3. Feature *discount_offered* has a moderate negative correlation with *weight_in_gms*.

—Further, graphs were plot based on categorical columns'mode_of_shipment',

'warehouse_block', 'product_importance', 'gender','customer_rating'-

<u>where we got the output-</u>

- Most of parcels are stored in warehouse_block F.
- The ship contributes the most late delivery.
- Most of parcels in all shipment priority are delivered late.

5.Model Fitting-

–Through ‘Def Fit_evaluation ‘ function the model were initiated & evaluated.
& through ‘Model Comparison’ function the following models were passes through dataframe -
LogisticRegression,DecisionTreeClassifier,RandomForestClassifier,
KNeighborsClassifier,Support Vector Classifier.
with Accuracy_Train','Accuracy_Test','Precision','Recall_Train,'Recall_Test',
'F1 Score','AUC': - where the result is as-
Only **Logistic Regression and SVC which are neither overfitting nor underfiting**. Logistic Regression has the highest recall.

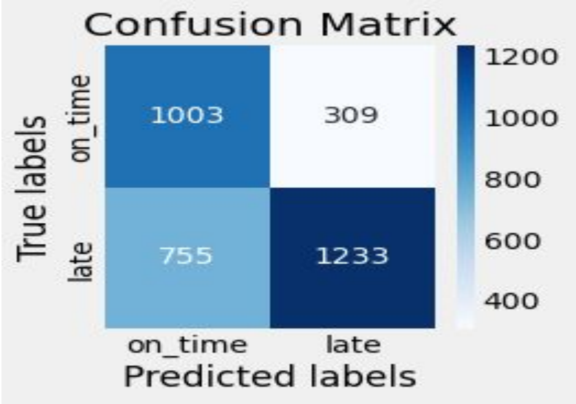
	Model	Accuracy_Train	Accuracy_Test	Precision	Recall_Train	Recall_Test	F1 Score	AUC
0	Logistic Regression	0.63	0.63	0.67	0.75	0.75	0.71	0.6
1	Decision Tree	1.0	0.66	0.72	1.0	0.71	0.71	0.64
2	Random Forest	1.0	0.67	0.76	1.0	0.66	0.71	0.67
3	KNN	0.76	0.64	0.72	0.77	0.67	0.69	0.64
4	SVC	0.68	0.66	0.88	0.52	0.51	0.65	0.7

6. Parameter tuning-

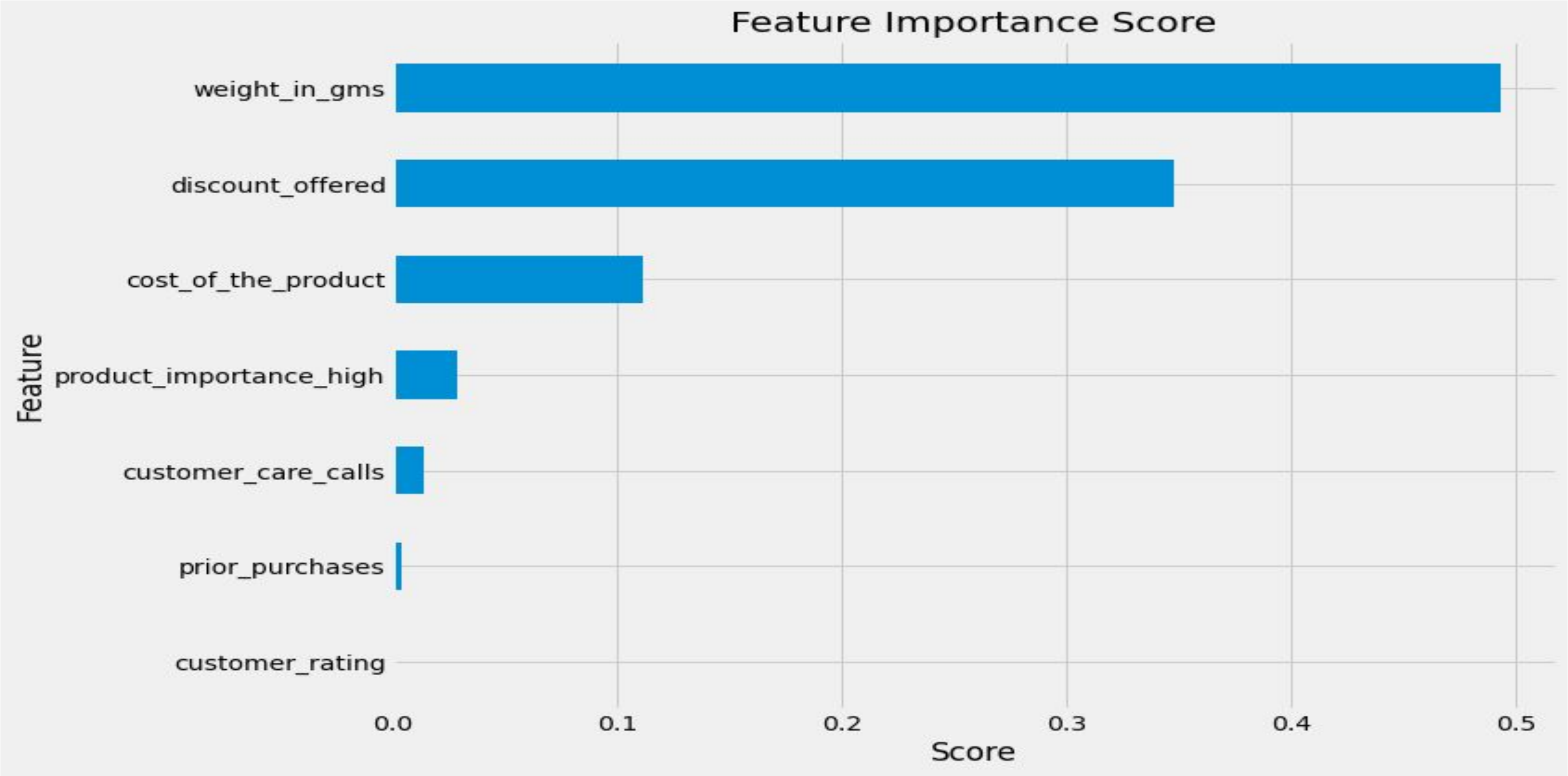
	Model	Accuracy_Train	Accuracy_Test	Precision	Recall_Train	Recall_Test	F1 Score	AUC
0	Logistic Regression	0.59	0.6	0.6	1.0	1.0	0.75	0.5
1	Decision Tree	0.67	0.68	0.8	0.62	0.62	0.7	0.69
2	Random Forest	1.0	0.67	0.76	1.0	0.66	0.71	0.67
3	KNN	0.81	0.65	0.72	0.82	0.69	0.7	0.64
4	SVC	0.59	0.6	0.6	1.0	1.0	0.75	0.5

–Decision Tree algorithm with hyper-parameter tuning has a good balance between its score, also neither underfitting nor overfitting.

–Confusion Matrix after parameter tuning-



7.Feature Importance-



- The operation team should add more manpower when there is a sale program, especially for the discount more than 10% and the parcel weight is 1 - 4 Kg.
- The parcel should not be centralized in the warehouse block F, so that the handling is not too crowded which can cause the late shipment.
- Adding more features can imporve model performance, such as delivery time estimation, delivery date, customer address, and courier.

CONCLUSION

- The timely delivery of products can also have a significant financial impact on the e-commerce business. If products are delivered on time, it can lead to increased sales, repeat business, and revenue growth. However, if products are consistently delivered late, it can lead to canceled orders, lost sales, and increased shipping costs, which can negatively impact the company's bottom line.
- Late delivery of products can have legal consequences, particularly if there are contractual obligations regarding delivery times. Failure to meet these obligations can result in breach of contract lawsuits, financial penalties, and damage to the business reputation.
- Timely delivery of products is critical to maintaining a positive business reputation. If products are consistently delivered on time, it can lead to a positive brand reputation and increased customer trust. However, if products are frequently delivered late, it can damage the business reputation and lead to decreased customer trust.

FURTHER ENHANCING MODEL

- The model Applied for the business problem were machine learning based. By maintaining accuracy ,further we can enhance by several deep learning models that could be useful for predicting delivery of a product on time.
- Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, Autoencoders.
- Time-series models are useful for analyzing data that changes over time, such as the frequency and timing of deliveries. You could use time-series models to predict the delivery time based on historical data, identifying patterns and trends that can help you make accurate predictions.
- RapidMiner can be useful for predicting the delivery of products on time. RapidMiner is a data science platform that offers a variety of tools and techniques for data preprocessing, modeling, and evaluation. By leveraging RapidMiner's predictive modeling capabilities, you can build and train a model that can predict the likelihood of delivery delays based on various factors such as shipping method, location, product type, and weather conditions.

THANK YOU