Project documentation

Ecommerce Shipping Prediction Using Machine Learning

1. Introduction

1.1. Project overviews

Machine Learning forecasts precise delivery dates for e-commerce, enhancing sales and customer satisfaction.

1.2. Objectives

Create a machine learning model to forecast shipping times for e-commerce orders. Utilize historical order data, real-time carrier information, and external data sources to predict either delivery ranges or specific delivery dates.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

E-commerce customers often feel frustrated by inaccurate or unreliable shipping estimates, resulting in a negative shopping experience. Project Proposal (Proposed Solution)

Train machine learning models on historical data to predict future commerce delivery times.

2.2. Initial Project Planning

The plan outlines 3 phases:

1. Define & Understand: Identify business needs, gather relevant data, and explore it to understand factors affecting shipping times.
2. Model Development & Training: Engineer features from data, choose a suitable ML model, train it, and evaluate its performance.
3. Deployment & Monitoring: Deploy the model for use, monitor its performance, and retrain it periodically to maintain accuracy.

The plan emphasizes defining success criteria, acquiring resources, and establishing communication channels for asmooth project execution.

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

This project uses data from a public source called Kaggle andthis have 10000 records in it.

With the data in hand, we'll explore it using various methods to gain better insights. These methods include visualizations, such as charts, and analysis techniques, like pattern detection. While there are numerous ways to explore data, we'll concentrate on a few common approaches here. Importing the libraries

Import the necessary libraries as shown in the image.

```
[1]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder , StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from keras.models import Sequential
from keras.layers import Dense
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

3.2. Data Quality Report

A Data Quality Report is basically a checkup for your data! Itassesses how well your data meets your needs.

Here,Many data analysis tools, like pandas, can handle differentfile types you might encounter, including CSV, Excel, TXT, and JSON. In pandas, you can use the read_csv() function specifically for CSV files. Just point it to the location of your CSVfile, and pandas will do the magic! It will take the data from that file and organize it into a neat table format called a DataFrame, making it much easier to work with and analyze.

```
[9]: #Information about the columns
dataset.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
```

This gives us the information about the Dataset.

3.3. Data Exploration and Preprocessing

Our data might be messy right now, kind of like a dirty window. We can't see clearly through it, which is bad for machine learningmodels. So before we use the data, we need to clean it up. This cleaning process is called pre-processing. Here are some things we might do:

Handling missing values Handling

categorical dataHandling Outliers

Handling missing values

Firstly we will check whether there are null values in thedataset and then handle those null values.

```
[11]: #Checking if there is any null values in the dataset
dataset.isnull().sum()
```

```
[11]: 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
```

Since there are no null values in the dataset here there is noneed of handling the null values.

Data preprocessing

```
1]: #Dropping the ID column because of high cardinality
data=dataset.drop(['ID'],axis=1)
data.head()
```

1]:

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	We
0	D	Flight	4	2	177	3	low	F	44	
1	F	Flight	4	5	216	2	low	M	59	
2	A	Flight	2	2	183	4	low	M	48	
3	B	Flight	3	3	176	4	medium	M	10	
4	C	Flight	2	2	184	3	medium	F	46	


```
[54]: #Dropping the ID column because of high cardinality
data=dataset.drop(['ID'],axis=1)
data.head()
```

[54]:

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered	W
0	D	Flight	4	2	177	3	low	F	44	
1	F	Flight	4	5	216	2	low	M	59	
2	A	Flight	2	2	183	4	low	M	48	
3	B	Flight	3	3	176	4	medium	M	10	
4	C	Flight	2	2	184	3	medium	F	46	

Handling categorical data

Chi square test of independence

```
[56]: from sklearn.feature_selection import chi2
from scipy.stats import chi2_contingency
```

```
[58]: #Warehouse_block
crossTab = pd.crosstab(data['Warehouse_block'], data['Reached.on.Time_Y.N'])
ChiSqResult = chi2_contingency(crossTab)
print("p-value ",ChiSqResult[1])
```

```
p-value  0.8959524278243698
```

```
[60]: crossTab = pd.crosstab(data['Mode_of_Shipment'],data['Reached.on.Time_Y.N'])
ChiSqResult = chi2_contingency(crossTab)
print("p-value ",ChiSqResult[1])
```

```
p-value  0.6895487627593786
```

```
[62]: # Product_importance
crossTab = pd.crosstab(data['Product_importance'],data['Reached.on.Time_Y.N'])
ChiSqResult = chi2_contingency(crossTab)
print("p-value ",ChiSqResult[1])
```

```
p-value  0.002230383104745087
```

```
[64]: # Gender
crossTab = pd.crosstab(data['Gender'],data['Reached.on.Time_Y.N'])
ChiSqResult = chi2_contingency(crossTab)
print("p-value ",ChiSqResult[1])
```

```
p-value  0.6367032124181522
```

From chi square test, we can conclude that three of the independent categorical features are not related to the response variable

Dropping unwanted columns

beacause the p value is greater than 0.05. So, these features can be removed and only product importance feature can be included

[66]:
```
#Renaming the column Reached.on.Time_Y.N
data.rename(columns={'Reached.on.Time_Y.N':'Reached on Time'}, inplace=True)
```

[68]:
```
X=data.drop(['Reached on Time','Warehouse_block','Mode_of_Shipment','Gender'],axis=1)
y=data['Reached on Time']
X
```

[68]:

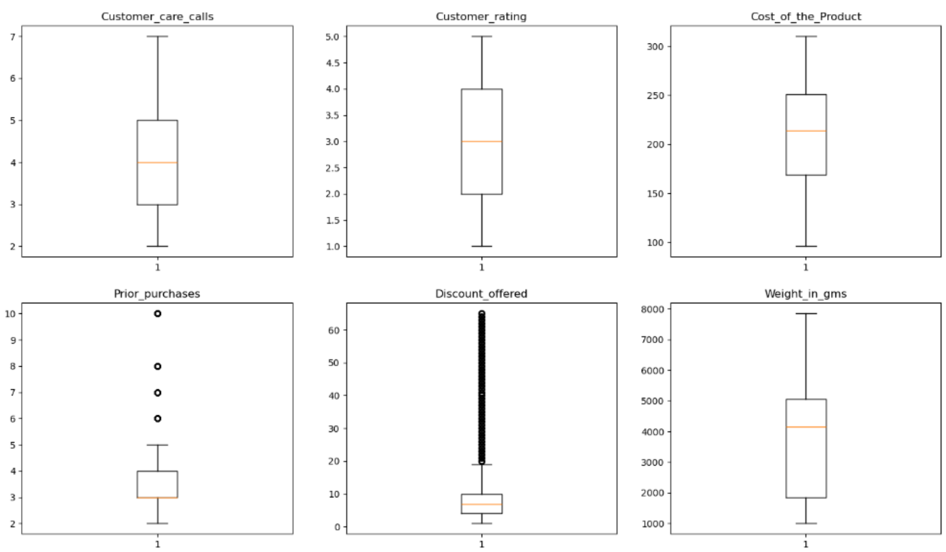
	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Discount_offered	Weight_in_gms
0	4	2	177	3	low	44	1233
1	4	5	216	2	low	59	3088
2	2	2	183	4	low	48	3374
3	3	3	176	4	medium	10	1177
4	2	2	184	3	medium	46	2484
...	...	...	...	...	...	...	...
10994	4	1	252	5	medium	1	1538
10995	4	1	232	5	medium	6	1247
10996	5	4	242	5	low	4	1155
10997	5	2	223	6	medium	2	1210
10998	2	5	155	5	low	6	1639

10999 rows × 7 columns

Handling Outliers

Handling outliners

[48]:
```
C=0
plt.figure(figsize=(18, 10))
for i in dataset.drop(columns=[
    'Warehouse_block', 'Mode_of_Shipment', 'Gender', 'Reached.on.Time_Y.N', 'ID' ]).columns:
    if str(dataset[i].dtype)=='object':
        continue
    plt.subplot(2, 3, C+1)
    plt.boxplot(dataset[i])
    plt.title(i)
    C+=1
plt.show()
```



```
[58]: def check_outliers(arr):
    Q1 = np.percentile(arr, 25, interpolation = 'midpoint')
    Q3 = np.percentile(arr, 75, interpolation = 'midpoint')
    IQR = Q3 - Q1

#Above Upper bound
    upper=Q3+1.5*IQR
    upper_array=np.array(arr>=upper)
    print(' '*3,len(upper_array[upper_array == True]), 'are over the upper bound:', upper)

    #Below Lower bound
    lower=Q1-1.5*IQR
    lower_array=np.array(arr<=lower)
    print(' '*3, len(lower_array[lower_array == True]), 'are less than the lower bound:', lower, '\n')

for i in dataset.drop(columns=[ 'Warehouse_block', 'Mode_of_Shipment', 'Gender', 'Reached.on.Time_Y.N', 'ID' ]).columns:
    if str(dataset[i].dtype)=='object':
        continue
    print(i)
    check_outliers(dataset[i])
```

```
Customer_care_calls
    0 are over the upper bound: 8.0
    0 are less than the lower bound: 0.0

Customer_rating
    0 are over the upper bound: 7.0
    0 are less than the lower bound: -1.0

Cost_of_the_Product
    0 are over the upper bound: 374.0
    0 are less than the lower bound: 46.0

Prior_purchases
    1003 are over the upper bound: 5.5
    0 are less than the lower bound: 1.5

Discount_offered
    2262 are over the upper bound: 19.0
    0 are less than the lower bound: -5.0

Weight_in_gms
    0 are over the upper bound: 9865.75
    0 are less than the lower bound: -2976.25
```

4. Model Development Phase

4.1. Feature Selection Report

A Feature Selection Report dives into the process of choosing themost informative and relevant features from your dataset for building a machine learning model. It's like picking the most important ingredients for a recipe – you wouldn't throw everythingin the kitchen sink, right? So here we remove the unwanted columns and keep the required columns only in the dataset and encode the Alphabetical values to the numeric values. Using the Encoders we can process the data into the required form.

```
[63]: #Renaming the column Reached.on.Time_Y.N
data.rename(columns={'Reached.on.Time_Y.N':'Reached on Time'}, inplace=True)
```

```
[65]: X=data.drop(['Reached on Time','Warehouse_block','Mode_of_Shipment','Gender'],axis=1)
y=data['Reached on Time']
X.head()
```

[65]:

	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Discount_offered	Weight_in_gms
0	4	2	177	3	low	44	1233
1	4	5	216	2	low	59	3088
2	2	2	183	4	low	48	3374
3	3	3	176	4	medium	10	1177
4	2	2	184	3	medium	46	2484

```
[78]: from sklearn.preprocessing import LabelEncoder
```

```
[80]: le=LabelEncoder()
```

```
[82]: y = le.fit_transform(y)
```

```
[84]: y
```

```
[84]: array([1, 1, 1, ..., 0, 0, 0])
```

```
[86]: X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.3, random_state=42)

      #Scaling the data
      ms = MinMaxScaler()
      X_train = ms.fit_transform(X_train)
      X_test = ms.fit_transform(X_test)
```

Let us see the total discrimination about the data that we havepreprocessed.

```
[13]: #Basic summary statistics
      dataset.describe()
```

[13]:

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000	10999.000000
mean	5500.00000	4.054459	2.990545	210.196836	3.567597	13.373216	3634.016729	0.596691
std	3175.28214	1.141490	1.413603	48.063272	1.522860	16.205527	1635.377251	0.490584
min	1.00000	2.000000	1.000000	96.000000	2.000000	1.000000	1001.000000	0.000000
25%	2750.50000	3.000000	2.000000	169.000000	3.000000	4.000000	1839.500000	0.000000
50%	5500.00000	4.000000	3.000000	214.000000	3.000000	7.000000	4149.000000	1.000000
75%	8249.50000	5.000000	4.000000	251.000000	4.000000	10.000000	5050.000000	1.000000
max	10999.00000	7.000000	5.000000	310.000000	10.000000	65.000000	7846.000000	1.000000

Exploratory Data Analysis

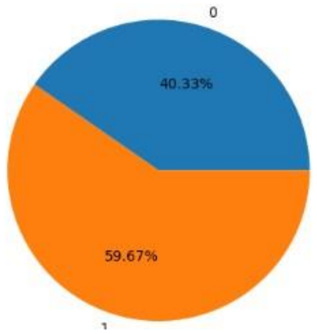
```
[15]: dataset.head()
```

[15]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
0	1	D	Flight	4	2	177	3	low	F	44
1	2	F	Flight	4	5	216	2	low	M	59
2	3	A	Flight	2	2	183	4	low	M	48
3	4	B	Flight	3	3	176	4	medium	M	10
4	5	C	Flight	2	2	184	3	medium	F	46

```
[17]: delay=pd.DataFrame(dataset.groupby(['Reached.on.Time_Y.N'])['ID'].count()/len(dataset)).reset_index()
      plt.pie(delay['ID'],labels=delay['Reached.on.Time_Y.N'],autopct='%1.2f%%')
```

```
[17]: ([<matplotlib.patches.Wedge at 0x21cf0389f50>,
        <matplotlib.patches.Wedge at 0x21cf03e8650>],
       [Text(0.32902377754583456, 1.0496396304491695, '0'),
        Text(-0.32902377754583423, -1.0496396304491697, '1')],
       [Text(0.17946751502500063, 0.5725307075177288, '40.33%'),
        Text(-0.17946751502500047, -0.5725307075177288, '59.67%')])
```



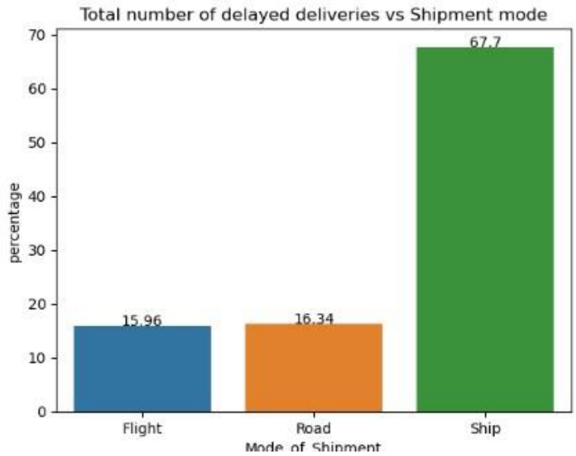
[19]:

```
data_v1 = dataset[dataset['Reached.on.Time_Y.N']==0]
```

[21]:

```
data_v2=pd.DataFrame((data_v1.groupby(['Mode_of_Shipment'])['ID'].count())/len(data_v1)*100)
data_v2=data_v2.reset_index()
visual=sns.barplot(x="Mode_of_Shipment", y="ID", data=data_v2 )
for index, row in data_v2.iterrows():
    visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total number of delayed deliveries vs Shipment mode')
plt.ylabel('percentage')
```

[21]: Text(0, 0.5, 'percentage')



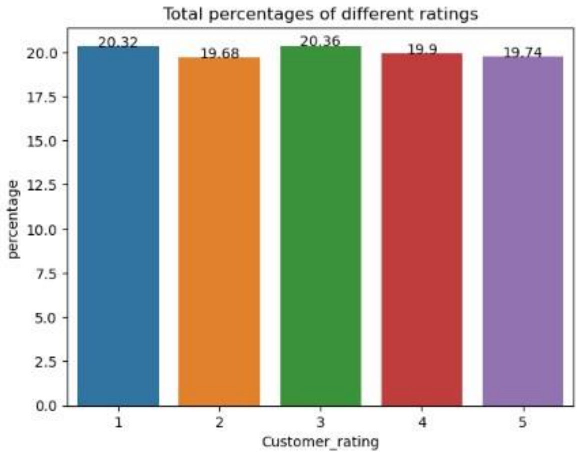
[23]:

```
data_v3=pd.DataFrame((data_v1.groupby(['Warehouse_block'])['ID'].count())/len(data_v1)*100)
data_v3=data_v3.reset_index()
visual=sns.barplot(x="Warehouse_block", y="ID", data=data_v3 )
for index, row in data_v3.iterrows():
    visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total number of delayed deliveries vs Warehouse block')
plt.ylabel('percentage')
```

[23]: Text(0, 0.5, 'percentage')

```
[25]: data_v4=pd.DataFrame((dataset.groupby(['Customer_rating'])['ID'].count())/len(dataset)*100)
      data_v4=data_v4.reset_index()
      visual=sns.barplot(x="Customer_rating", y="ID", data=data_v4 )
      for index, row in data_v4.iterrows():
          visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
      plt.title('Total percentages of different ratings')
      plt.ylabel('percentage')
```

```
[25]: Text(0, 0.5, 'percentage')
```




```
[27]: dataset['Reached.on.Time_Y.N'].replace({1 : "Yes", 0: "No"}, inplace = True)
```

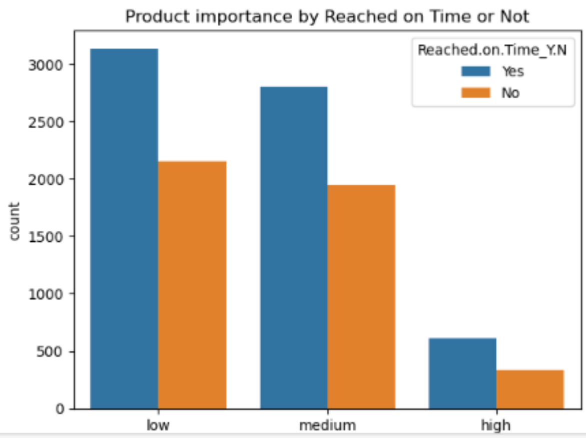
```
[29]: dataset.head()
```

[29]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
0	1	D	Flight	4	2	177	3	low	F	44
1	2	F	Flight	4	5	216	2	low	M	59
2	3	A	Flight	2	2	183	4	low	M	48
3	4	B	Flight	3	3	176	4	medium	M	10
4	5	C	Flight	2	2	184	3	medium	F	46

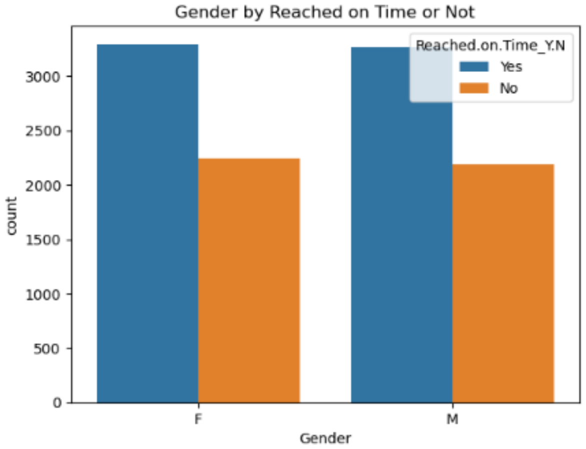
```
[31]: sns.countplot(x = "Product_importance", data = dataset, hue="Reached.on.Time_Y.N")
      plt.title("Product importance by Reached on Time or Not")
```

```
[31]: Text(0.5, 1.0, 'Product importance by Reached on Time or Not')
```



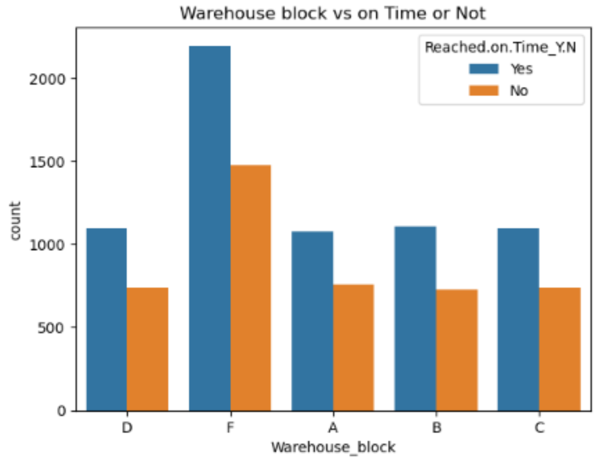
```
[33]: sns.countplot(x = "Gender", data = dataset, hue="Reached.on.Time_Y.N")
plt.title("Gender by Reached on Time or Not")
```

[33]: Text(0.5, 1.0, 'Gender by Reached on Time or Not')



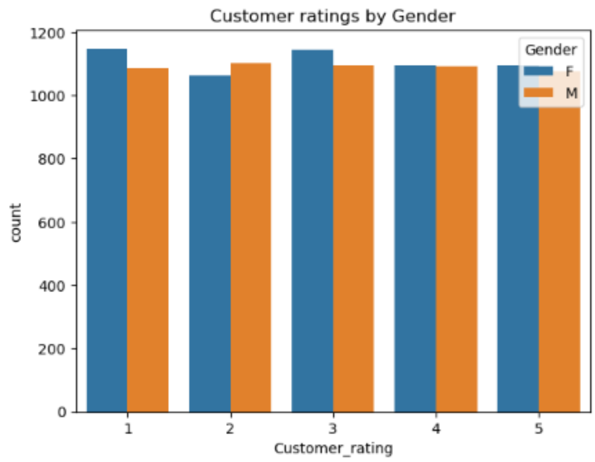

```
[35]: sns.countplot(x = "Warehouse_block", data = dataset, hue = 'Reached.on.Time_Y.N' )
plt.title("Warehouse block vs on Time or Not")
```

[35]: Text(0.5, 1.0, 'Warehouse block vs on Time or Not')



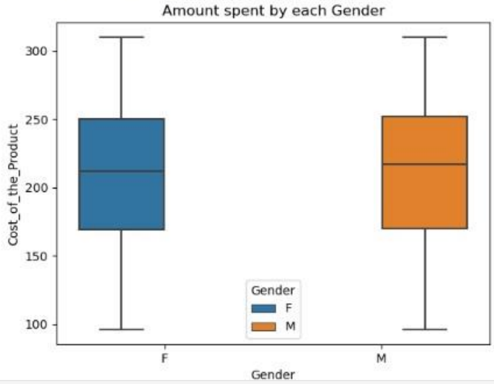

```
[37]: sns.countplot(x = "Customer_rating", data = dataset, hue="Gender")
plt.title("Customer ratings by Gender")
```

[37]: Text(0.5, 1.0, 'Customer ratings by Gender')



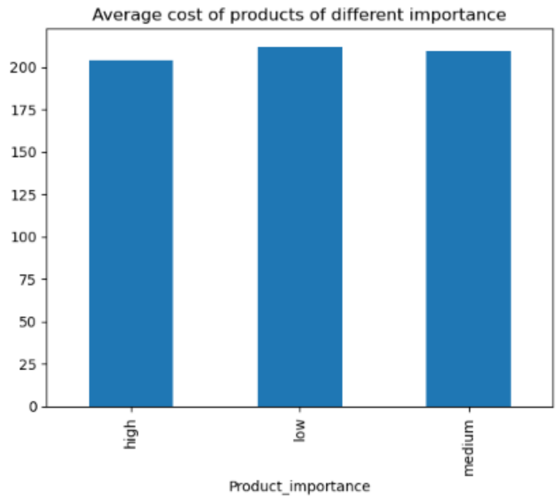
[39]:
```
sns.boxplot(x='Gender',y='Cost_of_the_Product',data=dataset,hue='Gender')
plt.title("Amount spent by each Gender")
```

[39]:
```
Text(0.5, 1.0, 'Amount spent by each Gender')
```



[41]:
```
dataset.groupby(['Product_importance'])['Cost_of_the_Product'].mean().plot.bar()
plt.title("Average cost of products of different importance")
```

[41]:
```
Text(0.5, 1.0, 'Average cost of products of different importance')
```



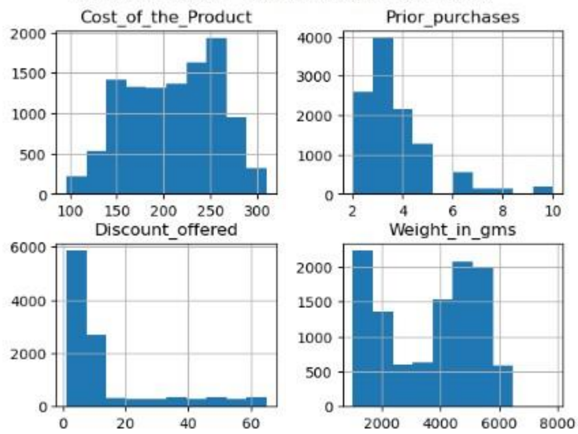
[43]:
```
dataset.head()
```

[43]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
0	1	D	Flight	4	2	177	3	low	F	44
1	2	F	Flight	4	5	216	2	low	M	59
2	3	A	Flight	2	2	183	4	low	M	48
3	4	B	Flight	3	3	176	4	medium	M	10
4	5	C	Flight	2	2	184	3	medium	F	46

[45]:
```
dataset[['Cost_of_the_Product','Prior_purchases','Discount_offered','Weight_in_gms']].hist()
```

[45]:
```
array([[<Axes: title={'center': 'Cost_of_the_Product'}>,
        <Axes: title={'center': 'Prior_purchases'}>],
       [<Axes: title={'center': 'Discount_offered'}>,
        <Axes: title={'center': 'Weight_in_gms'}>]], dtype=object)
```



```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=42)

#Scaling the data
ms = MinMaxScaler()
X_train = ms.fit_transform(X_train)
X_test = ms.fit_transform(X_test)
```

The process that is done here is,

Splitting the Data: Preparing for Learning andAssessment

The preprocessed data serves as the fuel for our model. However, we don't simply throw it all at the model at once. Astrategic data split is crucial:

Training Set : This serves as the basis for the learning process ofthe model. The features (such as order weight, distance) and theircorrelation with the goal variable (such as on-time vs. late delivery, expected delivery time) are presented to the model. The model gains the ability to recognize links and patterns via this exposure, which will help it forecast data that has not yet been observed.

Testing Set : The last test for the generalizability of the model isconducted on this untested set. Select metrics are used to evaluate the model on the testing set after it has been trained onthe training data and maybe fine-tuned using the validation set.
This gives an objective evaluation of the model's performance ondata that it has never seen before.

Crucially important is the size of each split (training, validation,and testing). Typically, 60–80% of the data are set aside for training, 10–20% for validation, and 10–20% for testing.
Depending on the size and features of the dataset, the preciseallocation can be changed.

4.2. Model Selection Report

A Model Selection Report acts like a guide for choosing the **champion** among machine learning models for your project. Imagine you're training a bunch of athletes (models) for a competition (solving a specific task). The Model Selection Reporthelps pick the one who will perform the best. Here There are many models that we used like,

Support Vector ClassifierLogistic
Regression Decision Tree
classifier KNN
Naive Bayes
XGBoost Ada Boost
Gradient Boosting, Decision Tree
classifer Artificial Neural
Network

Here are the particular codes for the models that are trained.

1. Support Vector Classifier

```
[88]: svm_model = svm.SVC(gamma='auto',C=5,kernel='rbf')
      svm_model.fit(X_train,y_train)
      y_pred = svm_model.predict(X_test)
      print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.54      0.85      0.66      1312
           1       0.84      0.53      0.65      1988

    accuracy                           0.66      3300
   macro avg       0.69      0.69      0.66      3300
weighted avg       0.72      0.66      0.65      3300

[90]: print(confusion_matrix(y_test,y_pred))

      [[1119  193]
       [ 940 1048]]
```

2. Logistic Regression

```
[92]: from sklearn.linear_model import LogisticRegression
```

```
[94]: lr=LogisticRegression()
      lr.fit(X_train,y_train)
      predLR=lr.predict(X_test)
      print(classification_report(y_test,predLR))
```

```
              precision    recall  f1-score   support

           0       0.54      0.59      0.56      1312
           1       0.71      0.67      0.69      1988

    accuracy                           0.64      3300
   macro avg       0.63      0.63      0.63      3300
weighted avg       0.64      0.64      0.64      3300
```

```
[96]: print(confusion_matrix(y_test,predLR))
```

```
[[ 771  541]
 [ 651 1337]]
```

3. Decision Tree classifier

```
[98]: from sklearn.tree import DecisionTreeClassifier
```

```
[100]: df=DecisionTreeClassifier(criterion='entropy',random_state=0)
       df.fit(X_train,y_train)
       preddf=df.predict(X_test)
       print(classification_report(y_test,preddf))
```

```
              precision    recall  f1-score   support

           0       0.56      0.59      0.58      1312
           1       0.72      0.70      0.71      1988

    accuracy                           0.66      3300
   macro avg       0.64      0.64      0.64      3300
weighted avg       0.66      0.66      0.66      3300
```

```
[102]: print(confusion_matrix(y_test,preddf))
```

```
[[ 769  543]
 [ 593 1395]]
```

4. KNN

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
predknn=knn.predict(X_test)
print(classification_report(y_test,predknn))
```

```
              precision    recall  f1-score   support

           0       0.55      0.61      0.58      1312
           1       0.72      0.67      0.70      1988

    accuracy                           0.65      3300
   macro avg       0.64      0.64      0.64      3300
weighted avg       0.65      0.65      0.65      3300
```

```
print(confusion_matrix(y_test,predknn))
```

```
[[ 802  510]
 [ 655 1333]]
```

5. Naive Bayes

```
[110]: from sklearn.naive_bayes import GaussianNB
       nb = GaussianNB()
```

```
[112]: nb.fit(X_train,y_train)
       prednb = nb.predict(X_test)
       print(classification_report(prednb,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.53      0.69      2415
           1       0.43      0.97      0.60       885

    accuracy                           0.65      3300
   macro avg       0.70      0.75      0.64      3300
weighted avg       0.83      0.65      0.66      3300
```

```
[114]: print(confusion_matrix(prednb,y_test))
```

```
[[1283 1132]
 [  29  856]]
```

6.XGBoost

```
[118]: import xgboost as xgb
       xg=xgb.XGBClassifier()
       xg.fit(X_train,y_train)
       predxg = xg.predict(X_test)
       print(classification_report(prednb,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.53      0.69      2415
           1       0.43      0.97      0.60       885

    accuracy                           0.65      3300
   macro avg       0.70      0.75      0.64      3300
weighted avg       0.83      0.65      0.66      3300
```

```
[120]: print(confusion_matrix(prednb,y_test))
```

```
[[1283 1132]
 [  29  856]]
```

7. Ada Boost and Gradient Boosting,

Initial Model Training Code, Model Validation and EvaluationReport.

```
[142]: model_list = {
           'Logistic Regression':lr,
           'XGBoost':xg,
           'Ada Boost':ab,
           'Gradient Boosting' : gb,
           'Support Vector Classifier': svm_model,
           'Naive Bias' : nb,
           'KNN' : knn,
           'Decision Tree' : df,

       }


       model_eval_info = []
       for i in model_list.keys():
           model_eval_info.append(eval(i,model_list[i]))
       model_eval_info = pd.DataFrame(model_eval_info, columns=['Name', 'Accuracy', 'f1_score', 'Recall', 'Precision'])
       model_eval_info.to_csv('model_eval.csv')
       model_eval_info
```

Evaluation before Tuning

[140]:
```
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score

def eval(name, model):
    y_pred = model.predict(X_test)
    result = []
    result.append(name)
    result.append("{:.2f}".format(accuracy_score(y_test, y_pred) * 100))
    result.append("{:.2f}".format(f1_score(y_test, y_pred) * 100))
    result.append("{:.2f}".format(recall_score(y_test, y_pred) * 100))
    result.append("{:.2f}".format(precision_score(y_test, y_pred) * 100))
    return result
```

[130]:
```
pred2= ab.predict(X_test)
pred3= gb.predict(X_test)
```

[132]:
```
print(classification_report(y_test, pred2))
```

```
              precision    recall  f1-score   support

           0       0.57      0.76      0.65      1312
           1       0.79      0.62      0.69      1988

    accuracy                           0.67      3300
   macro avg       0.68      0.69      0.67      3300
weighted avg       0.70      0.67      0.68      3300
```

[134]:
```
print(confusion_matrix(y_test, pred2))
```

```
[[ 995  317]
 [ 765 1223]]
```

[136]:
```
print(classification_report(y_test, pred3))
```

```
              precision    recall  f1-score   support

           0       0.56      0.87      0.68      1312
           1       0.87      0.55      0.67      1988

    accuracy                           0.68      3300
   macro avg       0.71      0.71      0.68      3300
weighted avg       0.75      0.68      0.68      3300
```

[138]:
```
print(confusion_matrix(y_test, pred3))
```

```
[[1142  170]
 [ 889 1099]]
```

Evaluation report:

[142]:

	Name	Accuracy	fl_score	Recall	Precision
0	Logistic Regression	63.88	69.17	67.25	71.19
1	XGBoost	66.76	70.75	66.75	75.27
2	Ada Boost	67.21	69.33	61.52	79.42
3	Gradient Boosting	67.91	67.49	55.28	86.60
4	Support Vector Classifier	65.67	64.91	52.72	84.45
5	Naive Bias	64.82	59.59	43.06	96.72
6	KNN	64.70	69.59	67.05	72.33
7	Decision Tree	65.58	71.06	70.17	71.98

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

Hyperparameter tuning is a critical step in the machine learning workflow that involves adjustingthe settings of your model to achieve the best possible performance. It's like fine-tuning the dials on a radio to get the clearest signal. In this documentation, we'll delve into the world of hyperparameter tuning, explaining its importance, common techniques, and best practices.

Random Forest Classifier

Hyperparametric Tuning

```
[144]: params = {'n_estimators':[150,500], 'criterion':['gini', 'entropy'], 'max_depth' : [7],
                'max_features' : [60,80,100]
               }
       #Hyper parameter tuning
       rf_model =GridSearchCV(estimator=RandomForestClassifier(),param_grid=params,scoring='accuracy', n_jobs = -1, cv=7, verbose = 3)
       rf_model.fit(X_train,y_train)
       y_pred=rf_model.predict(X_test)
       print(classification_report(y_test,y_pred))
```

```
Fitting 7 folds for each of 12 candidates, totalling 84 fits
              precision    recall  f1-score   support

           0       0.56      0.93      0.70      1312
           1       0.92      0.52      0.67      1988

    accuracy                           0.68      3300
   macro avg       0.74      0.73      0.68      3300
weighted avg       0.78      0.68      0.68      3300
```

```
[146]: print(confusion_matrix(y_test,y_pred))
```

```
[[1224   88]
 [ 953 1035]]
```

XG Boost Classifier

```
[148]: params2 = {'min_child_weight' : [10,20],
                 'gamma' : [1.5,2.0,2.5],
                 'colsample_bytree' : [0.6,0.8,0.9],
                 'max_depth' : [4,5,6]
                 }
       xgb1 = xgb.XGBClassifier(learning_rate=0.5, n_estimators = 100 , objective = 'binary:logistic', nthread=3)
       fitmodelXGB = GridSearchCV(xgb1,param_grid = params2, cv=5, refit = True , scoring = "accuracy", n_jobs = -1, verbose = 3)
       fitmodelXGB.fit(X_train,y_train)
       y_pred1=fitmodelXGB.predict(X_test)
       print(classification_report(y_test,y_pred1))
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
              precision    recall  f1-score   support

           0       0.56      0.85      0.68      1312
           1       0.85      0.56      0.68      1988

    accuracy                           0.68      3300
   macro avg       0.71      0.71      0.68      3300
weighted avg       0.73      0.68      0.68      3300
```

```
[150]: print(confusion_matrix(y_test,y_pred1))
```

```
[[1113  199]
 [ 868 1120]]
```

Logistic Regression

```
[152]: lg = LogisticRegression(n_jobs= -1 , random_state = 2)
params3 = {
    'C' : [6,8,10,15,20],
    'max_iter' : [60,80,100]
}
fitmodelLR = GridSearchCV(lg,param_grid = params3, cv=5, refit = True , scoring = "accuracy", n_jobs = -1, verbose = 3)
fitmodelLR.fit(X_train,y_train)
y_pred2=fitmodelLR.predict(X_test)
print(classification_report(y_test,y_pred2))

print("Best Score:")
print(fitmodelXGB.best_score_)
```

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
              precision    recall  f1-score   support

           0       0.54      0.59      0.56      1312
           1       0.71      0.67      0.69      1988

    accuracy                           0.64      3300
   macro avg       0.63      0.63      0.63      3300
weighted avg       0.64      0.64      0.64      3300

Best Score:
0.6824271115499185
```

```
[154]: print(confusion_matrix(y_test,y_pred2))
```

```
[[ 774  538]
 [ 658 1330]]
```

Support Vector Machine

```
svc = svm.SVC(random_state =3)
params4 = {
    'kernel' : ['ploy','rbf'],
    'C' : [10,13],
    'gamma' : [4,5],
    'tol' : [1e-1,1e-2,1e-3]
}
fitmodelSVC = GridSearchCV(svc,param_grid = params4, cv=5, refit = True , scoring = "accuracy", n_jobs = -1, verbose = 3)
fitmodelSVC.fit(X_train,y_train)
y_pred3=fitmodelLR.predict(X_test)
print(classification_report(y_test,y_pred3))
```

Artificial Neural Network

```
[156]: ann = Sequential()
```

```
[158]: ann.add(Dense(14,activation='relu'))
ann.add(Dense(26,activation='relu'))
ann.add(Dense(26,activation='relu'))
ann.add(Dense(1,activation='sigmoid'))
ann.compile(loss="binary_crossentropy", optimizer='adam',metrics=['accuracy'])
```

```
[160]: 
ann.fit(X_train, y_train, epochs=50, batch_size=15)
```

```
Epoch 1/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 3s 805us/step - accuracy: 0.6064 - loss: 0.6383
Epoch 2/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 1s 921us/step - accuracy: 0.6379 - loss: 0.5391
Epoch 3/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 811us/step - accuracy: 0.6599 - loss: 0.5256
Epoch 4/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 835us/step - accuracy: 0.6600 - loss: 0.5283
Epoch 5/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 902us/step - accuracy: 0.6663 - loss: 0.5259
Epoch 6/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 831us/step - accuracy: 0.6569 - loss: 0.5228
Epoch 7/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 837us/step - accuracy: 0.6687 - loss: 0.5204
Epoch 8/50
514/514 ━━━━━━━━━━━━━━━━━━━━ 0s 860us/step - accuracy: 0.6658 - loss: 0.5210
```

```
[162]: predictions = (ann.predict(X_test) > 0.5)
       print(classification_report(y_test,predictions))

104/104 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
              precision    recall  f1-score   support

           0       0.55      0.85      0.67      1312
           1       0.85      0.54      0.66      1988

    accuracy                           0.66      3300
   macro avg       0.70      0.70      0.66      3300
weighted avg       0.73      0.66      0.66      3300
```

```
[164]: print(confusion_matrix(y_test,predictions))

[[1120  192]
 [ 915 1073]]
```

5.2. Performance Metrics Comparison Report

A performance metrics comparison report evaluates a business's performance against benchmarks, competitors, or internal targets. It uses key performance indicators (KPIs) to analyze trends, identify strengths and weaknesses, and suggest improvements.

```
def hyper_eval(name, model):
    y_pred_prob = model.predict(X_test)

    if y_pred_prob.ndim == 2 and y_pred_prob.shape[1] > 1:
        y_pred = y_pred_prob.argmax(axis=1)
    else:
        y_pred = (y_pred_prob >= 0.5).astype(int)  # For binary

    result = []
    result.append(name)
    result.append("{:.2f}".format(accuracy_score(y_test, y_pred) * 100))
    result.append("{:.2f}".format(f1_score(y_test, y_pred, average='weighted') * 100))
    result.append("{:.2f}".format(recall_score(y_test, y_pred, average='weighted') * 100))
    result.append("{:.2f}".format(precision_score(y_test, y_pred, average='weighted') * 100))
    return result

# Example usage:
model_list = {
    'Logistic Regression(Hyper)': fitmodelLR,
    'XGBoost(Hyper)': fitmodelXGB,
    'Random Forest(Hyper)': rf_model,
    'ANN': ann,
    'SVC' : fitmodelSVC
}

model_hyper_eval_info = []
for name, model in model_list.items():
    model_hyper_eval_info.append(hyper_eval(name, model))

model_hyper_eval_info = pd.DataFrame(model_hyper_eval_info, columns=['Name', 'Accuracy', 'F1_Score', 'Recall', 'Precision'])
model_hyper_eval_info.to_csv('model_hyper_eval.csv', index=False)
model_hyper_eval_info
```

	Name	Accuracy	F1_Score	Recall	Precision
0	Logistic Regression(Hyper)	63.97	64.19	63.97	64.57
1	XGBoost(Hyper)	68.24	67.98	68.24	76.09
2	Random Forest(Hyper)	68.52	68.06	68.52	77.82
3	ANN	65.94	66.30	65.94	67.89
4	SVC	66.85	67.11	66.85	70.53

5.3. Final Model Selection Justification

	Name	Accuracy	fl_score	Recall	Precision
0	Logistic Regression	63.79	69.15	67.35	71.03
1	XGBoost	66.76	70.89	67.20	75.01
2	Ada Boost	67.21	69.33	61.52	79.42
3	Gradient Boosting	68.39	68.21	56.29	86.54
4	Support Vector Classifier	66.27	64.52	50.91	88.08
5	Naive Bias	64.73	59.04	42.20	98.24
6	KNN	64.70	69.59	67.05	72.33
7	Decision Tree	65.52	70.92	69.82	72.07

	Name	Accuracy	F1_Score	Recall	Precision
0	Logistic Regression(Hyper)	63.97	64.19	63.97	64.57
1	XGBoost(Hyper)	68.24	67.98	68.24	76.09
2	Random Forest(Hyper)	68.52	68.06	68.52	77.82
3	ANN	65.94	66.30	65.94	67.89
4	SVC	66.85	67.11	66.85	70.53

The Random Forest model is chosen as the best model due to its superior performance and efficiency.

High accuracy: It consistently outperforms other models in predicting outcomes.

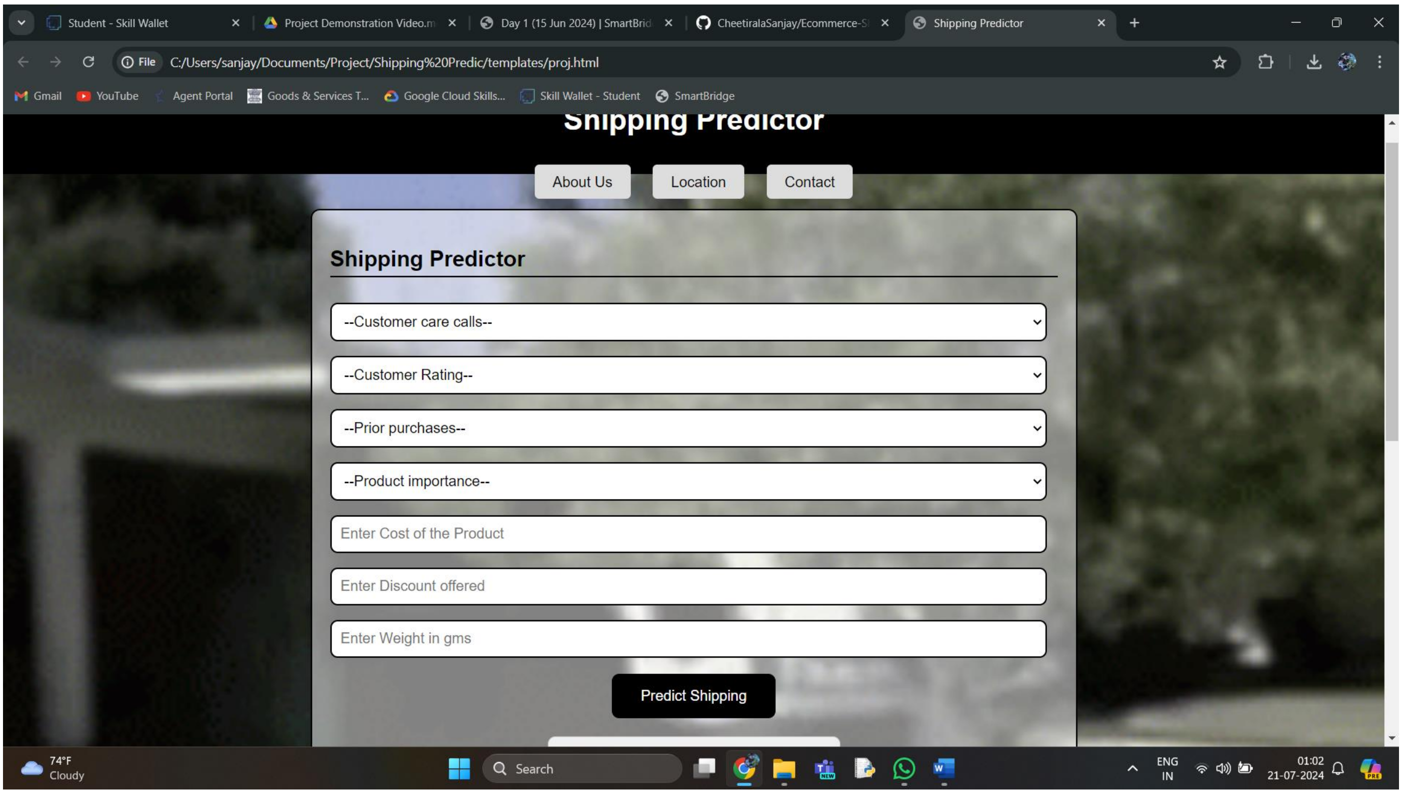
Efficiency: It handles large datasets effectively and requires reasonable computational resources.

Compared to other models, Random Forest demonstrated advantages in terms of accuracy androbustness. Hyperparameter tuning, feature engineering, and rigorous validation were employed to optimize the model's performance.

Overall, Random Forest is deemed reliable for accurate predictions and informed decision- making.

6. Results

Output Screenshots



7. Advantages & Disadvantages

Advantages :

- **Better Customer Experience**: Precise shipping estimates helpclients set reasonable expectations, which lowers annoyance andboosts confidence in your company.

- **Increased Operational Efficiency:** You may more effectivelydeploy resources for fulfillment and delivery by forecasting shipment timeframes, which can result in more seamless operations and possibly cheaper costs.

- **Lower Shipping Costs**: ML can assist in determining the mosteconomical shipping solutions depending on variables such as weight, destination, and delivery time.

- **Higher Sales:** When consumers know exactly when toanticipate their orders, they are more likely to finish their transactions when provided with clear and accuratedelivery information.

- **Proactive Exception Handling:** You may proactively engagewith clients and take corrective action to prevent interruptions when possible delays are identified in advance.

Disadvantages:

While promising, there are also some challenges to consider:

Data Quality: The completeness and quality of your data havea significant impact on the accuracy of machine learning models. Predictions that contain missing or inconsistent data may not be trustworthy.

Model Complexity: Complex machine learning models demand knowledge and computing power to develop and maintain, which may not be possible for all types of enterprises.

External Factors: Variations may arise that machine learningalgorithms are not always able to fully capture, such as weather disruptions, carrier problems, or high seasons.

Costs of Implementation: There may be upfront expenditurerequired when integrating machine learning technologies into current systems.

Over-reliance on Technology: Although machine learning (ML) is an effective tool, human oversight in the tracking and management of the shipping process shouldn't be replaced byML.

Overall, using machine learning for e-commerce shipping predictions provides significant benefits for businesses aiming to enhance customer experience, streamline operations, and gain a competitive edge. However, it's essential to recognize the challenges and ensure you have the necessary resources and expertise to implement and sustain an effective ML solution.

8. Conclusion

In this study, we explored the potential of using machine learning to predict shipping times for online orders. We began by understanding the data through visualization and analysis techniques. Following this, we preprocessed the data to ensure its accuracy and suitability for machine learning models.

We employed feature selection strategies to identify the most relevant factors influencing shipping times, which helped develop a more precise and effective model. We then compared several machine learning models to determine which performed best in terms of accuracy and generalizability.

By evaluating the model's performance on hypothetical data, we gained insights into its predictive power. This study demonstrated how machine learning could create realistic shipping expectations and improve the e-commerce experience for customers.

Overall, this research has significantly advanced the use of machine learning for predicting e-commerce delivery times. By continuously refining the model and exploring new strategies, we can achieve even more accurate shipping predictions, leading to increased customer satisfaction and enhanced operational efficiency for e-commerce businesses.

9. Future Scope

This project has successfully explored the potential of machinelearning for predicting e-commerce shipping times. Here are some exciting avenues for further development:

1. Advanced Machine Learning Techniques:

• Examine modeling the sequential nature of the shipping processusing recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, which may be able to capture intricateconnections between variables.

• Examine how deep learning methods might be included for even more advanced feature extraction and model functionality.

2. Real-time Data Integration:

• To increase prediction accuracy, use real-time data feeds from shipment monitoring systems or logistics providers to account fordynamic factors like carrier delays or weather disruptions.

• Create an interactive dashboard that enables proactive exception management and shows anticipated shipment times. This will facilitate quicker customer communication in the event ofany delays.

3. Multimodal Shipping and Personalization:

• To give clients a more complete view of their options, expand the model to take into account a larger range of shipping options(air, ground, and express), as well as the charges and delivery periods that go along with them.

• For even more individualized arrival estimates, personalize shipping projections by adding customer-specific variables likelocation, preferred shipping methods, and previous purchase history.

We can advance the field of machine learning-based e- commerce shipment prediction by investigating these potential future avenues. This will result in increased operational efficiency,more precise and dependable shipment estimates, and an informed and contented clientele for e-commerce companies.

10. Appendix

Source code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder , StandardScaler,
MinMaxScalerfrom sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn import svm
from sklearn.ensemble import
```

```
RandomForestClassifierfrom keras.models

import Sequential

from keras.layers import Dense

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

dataset =

pd.read_csv(r"C:\Users\sanjay\OneDrive\Desktop\Train.csv")

dataset.head()


dataset.info()


#Checking if there is any null values in the dataset

dataset.isnull().sum()

#Basic summary statistics

dataset.describe()


delay=pd.DataFrame(dataset.groupby(['Reached.on.Time_Y.N'])['ID'].count()/le
n(dataset)).reset
_index()

plt.pie(delay['ID'],labels=delay['Reached.on.Time_Y.N'],autopct=

'%1.2f%%')


data_v2=pd.DataFrame((data_v1.groupby(['Mode_of_Shipment'])['ID'].count
())/len(data_v1)*10 0)
```

```
data_v2=data_v2.reset_index()

visual=sns.barplot(x="Mode_of_Shipment", y="ID",

data=data_v2 )for index, row in data_v2.iterrows():

   visual.text(row.name,row.ID, round(row.ID,2), color='black',

ha="center")plt.title('Total number of delayed deliveries vs

Shipment mode') plt.ylabel('percentage')


data_v3=pd.DataFrame((data_v1.groupby(['Warehouse_block'])['ID'].count())/l

en(data_v1)*100) data_v3=data_v3.reset_index()

visual=sns.barplot(x="Warehouse_block", y="ID",

data=data_v3 )for index, row in data_v3.iterrows():

   visual.text(row.name,row.ID, round(row.ID,2), color='black',

ha="center")plt.title('Total number of delayed deliveries vs

Warehouse block') plt.ylabel('percentage')


data_v4=pd.DataFrame((dataset.groupby(['Customer_rating'])['ID'].count())/le

n(dataset)*100) data_v4=data_v4.reset_index()

visual=sns.barplot(x="Customer_rating", y="ID",

data=data_v4 )for index, row in data_v4.iterrows():

   visual.text(row.name,row.ID, round(row.ID,2), color='black', ha="center")
plt.title('Total percentages of different ratings')

plt.ylabel('percentage')
```

```
dataset['Reached.on.Time_Y.N'].replace({1 : "Yes", 0: "No"}, inplace = True)

sns.countplot(x = "Product_importance", data = dataset,
hue="Reached.on.Time_Y.N")plt.title("Product importance by Reached on
Time or Not")

sns.countplot(x = "Gender", data = dataset,
hue="Reached.on.Time_Y.N")plt.title("Gender by Reached on
Time or Not")

sns.countplot(x = "Warehouse_block", data = dataset, hue =
'Reached.on.Time_Y.N' )plt.title("Warehouse block vs on Time or Not")

sns.countplot(x = "Customer_rating", data = dataset,
hue="Gender")plt.title("Customer ratings by Gender")

sns.boxplot(x='Gender',y='Cost_of_the_Product',data=dataset,h
ue='Gender') plt.title("Amount spent by each Gender")

dataset.groupby(['Product_importance'])['Cost_of_the_Product'].mean().plot
.bar() plt.title("Average cost of products of different importance")

dataset[['Cost_of_the_Product','Prior_purchases','Discount_offered','Weight_i
n_gms']].hist()
```

```
#Handling outliners
def check_outliers(arr):
  Q1 = np.percentile(arr, 25, interpolation =
  'midpoint')Q3 = np.percentile(arr, 75,
  interpolation = 'midpoint')
  IQR = Q3 - Q1

#Above Upper bound
  upper=Q3+1.5*IQR
  upper_array=np.array(arr>=upper
  )
  print(' '*3,len(upper_array[upper_array == True]), 'are over the upper bound:',
  upper)

  #BeLow Lower bound
  lower=Q1-1.5*IQR
  lower_array=np.array(arr<=lower)
  print(' '*3, len(lower_array[lower_array == True]), 'are less than the lower
  bound:', lower, '\n')

for i in dataset.drop(columns=[ 'Warehouse_block',
'Mode_of_Shipment', 'Gender','Reached.on.Time_Y.N', 'ID'
]).columns:
  if str(dataset[i].dtype)=='object':
    continue
  print(i)
 check_outliers(dataset[i])


from sklearn.preprocessing import
LabelEncoderle=LabelEncoder()
y = le.fit_transform(y)
```

```
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.3,

random_state=42)#Scaling the data
ms = MinMaxScaler()
X_train =
ms.fit_transform(X_train)X_test
= ms.fit_transform(X_test)


svm_model = svm.SVC(gamma='auto',C=5,kernel='rbf')
svm_model.fit(X_train,y_train)
y_pred = svm_model.predict(X_test)
print(classification_report(y_test,y_pred))

from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
lr.fit(X_train,y_train)
predLR=lr.predict(X_test)
print(classification_report(y_test,predLR))

from sklearn.tree import DecisionTreeClassifier

df=DecisionTreeClassifier(criterion='entropy',random_state=0)
df.fit(X_train,y_train)
preddf=df.predict(X_test)
print(classification_report(y_test,preddf))

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
predknn=knn.predict(X_test)
print(classification_report(y_test,predknn))

from sklearn.naive_bayes import
GaussianNBnb = GaussianNB()
```

```
nb.fit(X_train,y_train)
prednb =
nb.predict(X_test)
print(classification_report(prednb,y_test))


import xgboost as xgb
xg=xgb.XGBClassifier()
xg.fit(X_train,y_train)
predxg = xg.predict(X_test)
print(classification_report(prednb,y_test))

from sklearn.ensemble import AdaBoostClassifier,

GradientBoostingClassifierab.fit(X_train,y_train)

gb.fit(X_train,y_train)

pred2= ab.predict(X_test)
pred3= gb.predict(X_test)

print(classification_report(y_test,  pred3))

#Evalution before Tuning

from sklearn.metrics import accuracy_score, f1_score, recall_score,

precision_score def eval(name, model):
  y_pred =
  model.predict(X_test)result
  = [] result.append(name)
  result.append("{:.2f}".format(accuracy_score(y_test, y_pred) *
  100)) result.append("{:.2f}".format(f1_score(y_test, y_pred) *
  100)) result.append("{:.2f}".format(recall_score(y_test,
  y_pred) * 100))
```

```
  result.append("{:.2f}".format(precision_score(y_test, y_pred) *
  100))
 return result

model_list = {
  'Logistic Regression':lr,
  'XGBoost':xg,
  'Ada Boost':ab,
  'Gradient Boosting' : gb,
  'Support Vector Classifier': svm_model,
  'Naive Bias' : nb,
  'KNN' : knn,
  'Decision Tree' : df,

}


model_eval_info = []
for i in model_list.keys():
  model_eval_info.append(eval(i,model_list[i]))
model_eval_info = pd.DataFrame(model_eval_info, columns=['Name',
'Accuracy', 'fl_score','Recall', 'Precision'])
model_eval_info.to_csv('model_eval.csv')
model_eval_info


#Hyper paramerter tuning

params = {'n_estimators':[150,500], 'criterion':['gini', 'entropy'],
     'max_depth' : [7], 'max_features' : [60,80,100]
     }
#Hyper parameter tuning
rf_model
=GridSearchCV(estimator=RandomForestClassifier(),param_grid=params,scoring
```

```
='accuracy',n_jobs = -1, cv=7, verbose = 3)
rf_model.fit(X_train,y_train)
y_pred=rf_model.predict(X_test)
print(classification_report(y_test,y_pred))

params2 = {'min_child_weight' :
        [10,20],'gamma' : [1.5,2.0,2.5],
        'colsample_bytree' : [0.6,0.8,0.9],
        'max_depth' : [4,5,6]
        }
xgb1 = xgb.XGBClassifier(learning_rate=0.5, n_estimators = 100 , objective =
'binary:logistic', nthread=3)
fitmodelXGB = GridSearchCV(xgb1,param_grid = params2, cv=5, refit = True ,
scoring ="accuracy", n_jobs = -1, verbose = 3)
fitmodelXGB.fit(X_train,y_train)
y_pred1=fitmodelXGB.predict(X_test)
print(classification_report(y_test,y_pred1))

lg = LogisticRegression(n_jobs= -1 ,
random_state = 2)params3 = {
  'C' : [6,8,10,15,20],
  'max_iter' : [60,80,100]
}
fitmodelLR = GridSearchCV(lg,param_grid = params3, cv=5, refit = True , scoring
= "accuracy", n_jobs = -1, verbose = 3)
fitmodelLR.fit(X_train,y_train)
y_pred2=fitmodelLR.predict(X_test)
print(classification_report(y_test,y_pred2))

print("Best Score:")
print(fitmodelXGB.best_score_)

#Evalution after tuning
```

```
def hyper_eval(name, model):
  y_pred_prob =
  model.predict(X_test)
  if y_pred_prob.ndim == 2 and y_pred_prob.shape[1]
    > 1:y_pred = y_pred_prob.argmax(axis=1)
  else:
    y_pred = (y_pred_prob >= 0.5).astype(int) # For binary

  result = []
  result.append(name)
  result.append("{:.2f}".format(accuracy_score(y_test, y_pred) * 100))
  result.append("{:.2f}".format(f1_score(y_test, y_pred, average='weighted')
  * 100)) result.append("{:.2f}".format(recall_score(y_test, y_pred,
  average='weighted') * 100))
  result.append("{:.2f}".format(precision_score(y_test, y_pred,
  average='weighted') * 100))return result

# Example usage:
model_list = {
  'Logistic Regression(Hyper)':
  fitmodelLR,'XGBoost(Hyper)':
  fitmodelXGB, 'Random
  Forest(Hyper)': rf_model, 'ANN':
  ann,
}

model_hyper_eval_info = []
for name, model in model_list.items():
  model_hyper_eval_info.append(hyper_eval(name, model))

model_hyper_eval_info = pd.DataFrame(model_hyper_eval_info,
columns=['Name', 'Accuracy', 'F1_Score', 'Recall', 'Precision'])
model_hyper_eval_info.to_csv('model_hyper_eval.csv',
index=False) model_hyper_eval_info
```

#Saving the model

import pickle as pkl

pkl.dump(ms,open('ship_scaler.pkl','wb'))

pkl.dump(le,open('ship_label.pkl','wb'))

import joblib
joblib.dump(ct,"shipct")pkl.dump
(rf_model,open('Shipping.pkl','w
b'))

Demo Link:

https://drive.google.com/file/d/1heKgRUpFJ-eaPRJGn8x0NPcwx57Yjgl_/view?usp=drive_link

Git Link:

https://github.com/CheetiralaSanjay/Ecommerce-Shipping-Prediction