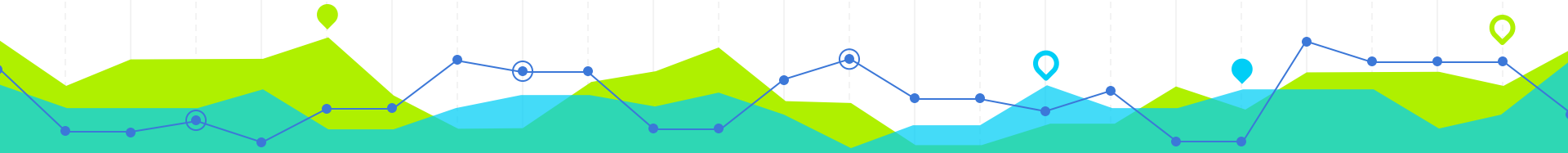


Airline Passenger Satisfaction Classification

INFO6105 Data Science Methods and Tools

Pooja Yendhe
NUID : 002922341

Introduction

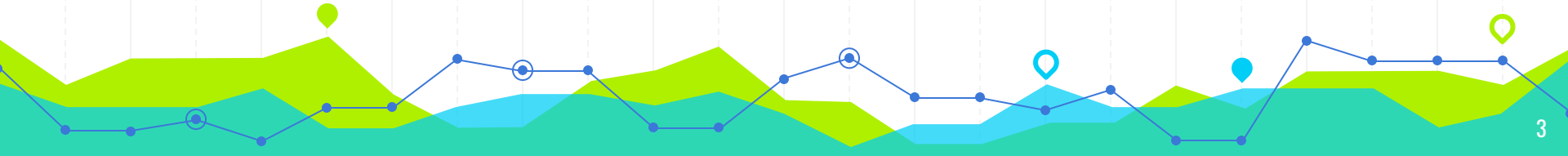
Data from airline passenger is collected to know what factors leads to satisfactory airline travel.

Here, the interest lies in predicting if a passenger satisfied with the airline services which could help airline improve their services and gain advantages in difficult situation and competition.

BUSINESS GOAL

What factors are important in estimating the satisfaction of a airline passenger tp help make future investments?

How effectively do those factors capture the satisfaction of a airline passenger so airline can continue doing improvements in those areas?



OBJECTIVE

We must use the available independent criteria to illustrate the satisfaction of passenger.

It could be used to determine how accurately satisfaction change when considering the free features. To adhere to predetermined satisfactory level they can suitably regulate their services.

STEPS

- Understanding the data
- Data Preparation and cleaning
- Data visualisation
- Model Fitting
- Evaluation of model
- Important features
- Hyperparameter Tuning of selected model
- Summary

Understanding the dataset

The dataset consist of scores of customer satisfaction from more than 120,000 passengers aboard airplanes, including details about each passenger, their flight, and the nature of their trip, as well as their assessments of various aspects like cleanliness, comfort, service, and overall experience.

The data consists of numerical as well as categorical features.

Data set reference :- Kaggel.com

Understanding the dataset

Details of data:

```
In [3]: df.shape
Out[3]: (129880, 24)

In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 24 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   ID                                      129880 non-null  int64
 1   Gender                                  129880 non-null  object
 2   Age                                     129880 non-null  int64
 3   Customer Type                           129880 non-null  object
 4   Type of Travel                          129880 non-null  object
 5   Class                                   129880 non-null  object
 6   Flight Distance                         129880 non-null  int64
 7   Departure Delay                         129880 non-null  int64
 8   Arrival Delay                           129487 non-null  float64
 9   Departure and Arrival Time Convenience  129880 non-null  int64
 10  Ease of Online Booking                  129880 non-null  int64
 11  Check-in Service                        129880 non-null  int64
 12  Online Boarding                         129880 non-null  int64
 13  Gate Location                           129880 non-null  int64
 14  On-board Service                        129880 non-null  int64
 15  Seat Comfort                            129880 non-null  int64
 16  Leg Room Service                        129880 non-null  int64
 17  Cleanliness                             129880 non-null  int64
 18  Food and Drink                          129880 non-null  int64
 19  In-flight Service                       129880 non-null  int64
 20  In-flight Wifi Service                  129880 non-null  int64
 21  In-flight Entertainment                 129880 non-null  int64
 22  Baggage Handling                        129880 non-null  int64
 23  Satisfaction                            129880 non-null  object
dtypes: float64(1), int64(18), object(5)
```

Understanding the dataset

Details of data:

memory usage: 23.8+ MB

```
In [5]: df.head(5)
```

Out[5]:

Class	Flight Distance	Departure Delay	Arrival Delay	Departure and Arrival Time Convenience	...	On-board Service	Seat Comfort	Leg Room Service	Cleanliness	Food and Drink	In-flight Service	In-flight Wifi Service	In-flight Entertainment	Baggage Handling	Satisfaction
Business	821	2	5.0	3	...	3	5	2	5	5	5	3	5	5	Neutral or Dissatisfied
Business	821	26	39.0	2	...	5	4	5	5	3	5	2	5	5	Satisfied
Business	853	0	0.0	4	...	3	5	3	5	5	3	4	3	3	Satisfied
Business	1905	0	0.0	2	...	5	5	5	4	4	5	2	5	5	Satisfied
Business	3470	0	1.0	3	...	3	4	4	5	4	3	3	3	3	Satisfied

Data Preparation

Checking for duplicates:

```
dtype: int64

In [8]: df.duplicated().sum()
Out[8]: 0

In [9]:
```

There are no duplicate values.

Checking for null values:

```
In [7]: df.isnull().sum()
Out[7]: ID                                          0
        Gender                                      0
        Age                                         0
        Customer Type                               0
        Type of Travel                              0
        Class                                       0
        Flight Distance                             0
        Departure Delay                             0
        Arrival Delay                             393
        Departure and Arrival Time Convenience      0
        Ease of Online Booking                      0
        Check-in Service                            0
        Online Boarding                             0
        Gate Location                               0
        On-board Service                            0
        Seat Comfort                                0
        Leg Room Service                            0
        Cleanliness                                 0
        Food and Drink                              0
        In-flight Service                           0
        In-flight Wifi Service                      0
        In-flight Entertainment                     0
        Baggage Handling                            0
        Satisfaction                                0
```

Data Preparation

Handling null values: The null values are filled with median of the column.

```
: arrival_median = df['Arrival Delay'].median()
  print(arrival_median)
```

```
0.0
```

```
: df['Arrival Delay'].fillna(arrival_median,inplace= True)
```

```
: df.isnull().sum()
```

```
: ID                                          0
  Gender                                      0
  Age                                         0
  Customer Type                               0
  Type of Travel                              0
  Class                                       0
  Flight Distance                             0
  Departure Delay                             0
  Arrival Delay                               0
  Departure and Arrival Time Convenience      0
  Ease of Online Booking                      0
  Check-in Service                            0
  Online Boarding                             0
  Gate Location                               0
  On-board Service                            0
  Seat Comfort                                0
  Leg Room Service                            0
  Cleanliness                                 0
  Food and Drink                              0
  In-flight Service                           0
```

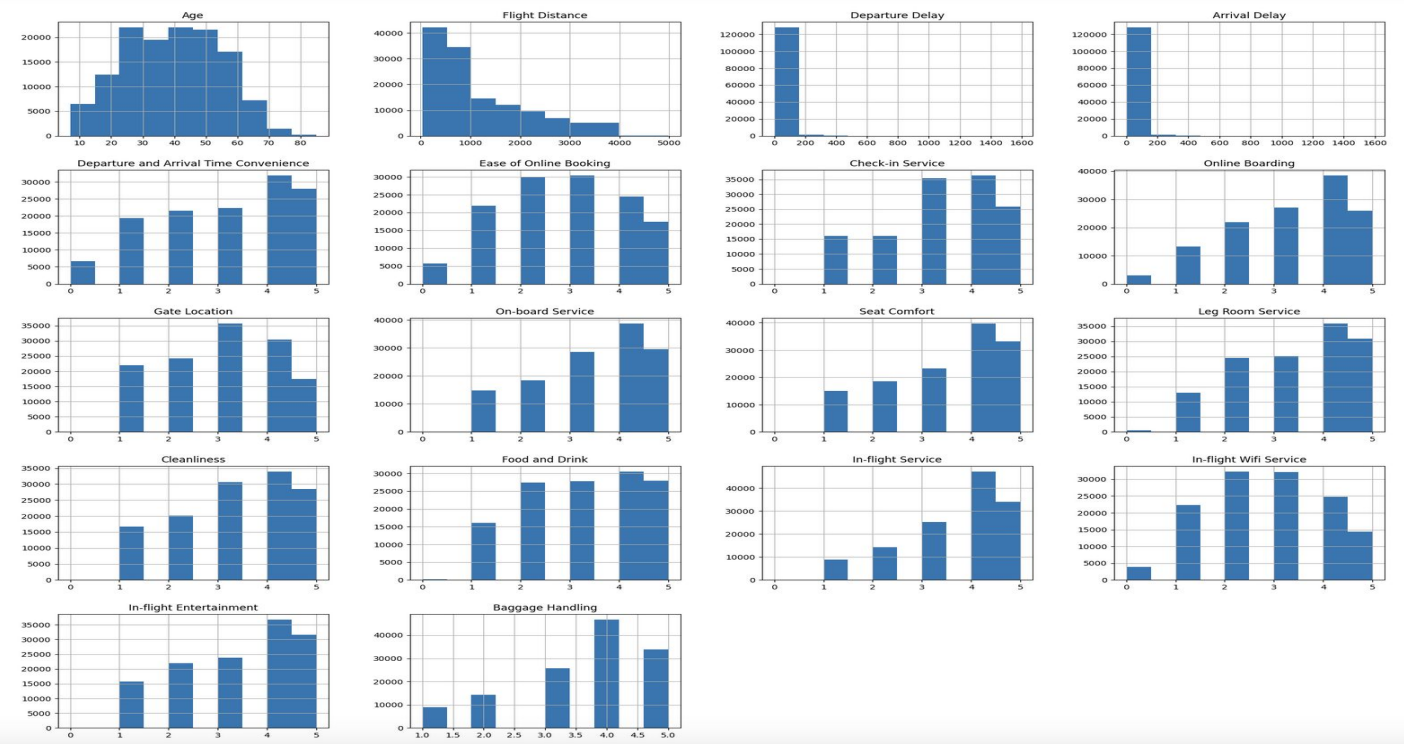
Data Preparation

Dropping ID column: The column ID is an irrelevant feature. Hence, removing it from data/

```
In [13]: df= df.drop(['ID'], axis = 1)
         df.info()
```

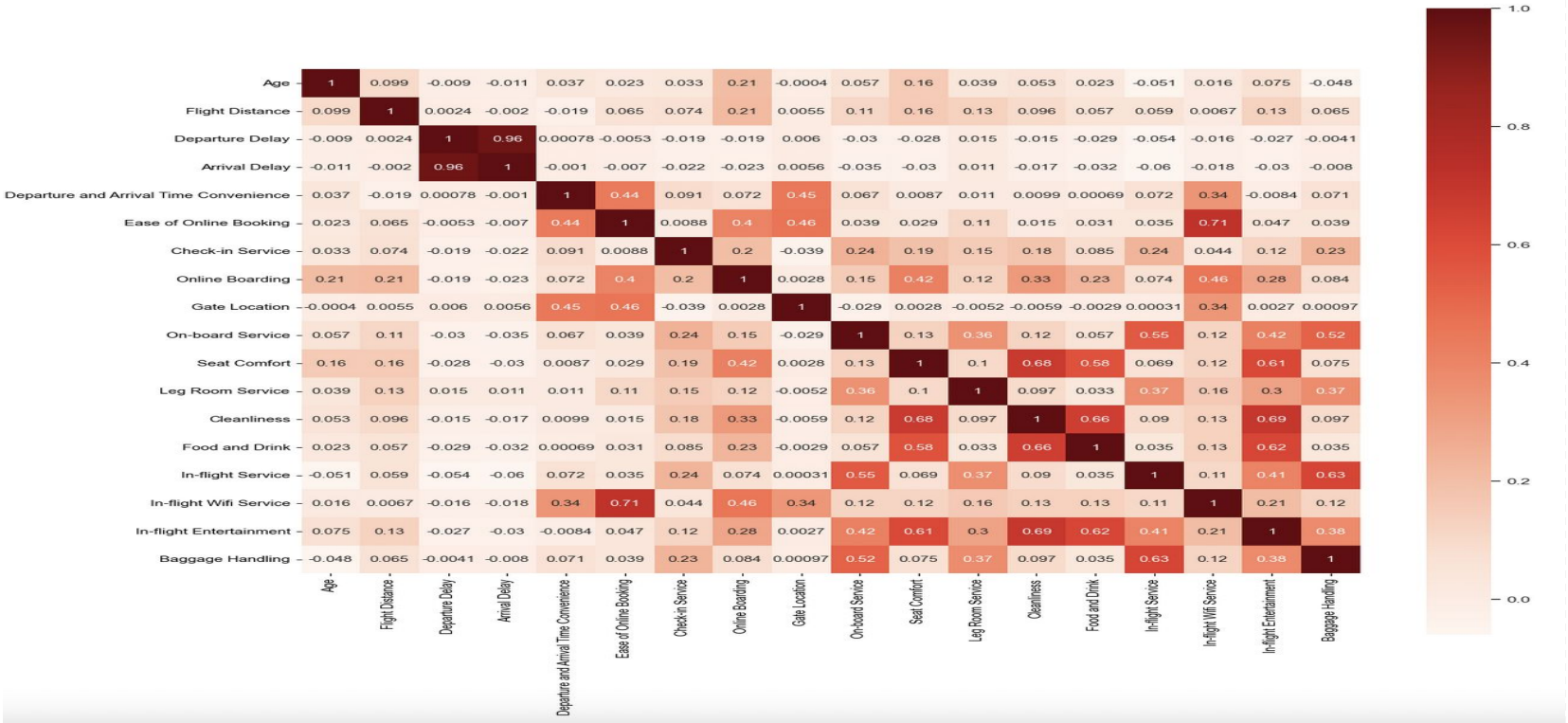
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 23 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   Gender                                  129880 non-null  object 
 1   Age                                     129880 non-null  int64  
 2   Customer Type                           129880 non-null  object 
 3   Type of Travel                          129880 non-null  object 
 4   Class                                   129880 non-null  object 
 5   Flight Distance                         129880 non-null  int64  
 6   Departure Delay                         129880 non-null  int64  
 7   Arrival Delay                           129880 non-null  float64
 8   Departure and Arrival Time Convenience  129880 non-null  int64  
 9   Ease of Online Booking                  129880 non-null  int64  
 10  Check-in Service                        129880 non-null  int64  
 11  Online Boarding                         129880 non-null  int64  
 12  Gate Location                           129880 non-null  int64  
 13  On-board Service                        129880 non-null  int64  
 14  Seat Comfort                            129880 non-null  int64  
 15  Leg Room Service                        129880 non-null  int64  
 16  Cleanliness                             129880 non-null  int64  
 17  Food and Drink                          129880 non-null  int64  
 18  In-flight Service                       129880 non-null  int64  
 19  In-flight Wifi Service                  129880 non-null  int64  
 20  In-flight Entertainment                 129880 non-null  int64  
 21  Baggage Handling                        129880 non-null  int64  
 22  Satisfaction                            129880 non-null  object 
dtypes: float64(1), int64(17), object(5)
memory usage: 22.8+ MB
```

Data Visualization



It is observed that the data in dataset is skewed for eg, departure delay and arrival delay.

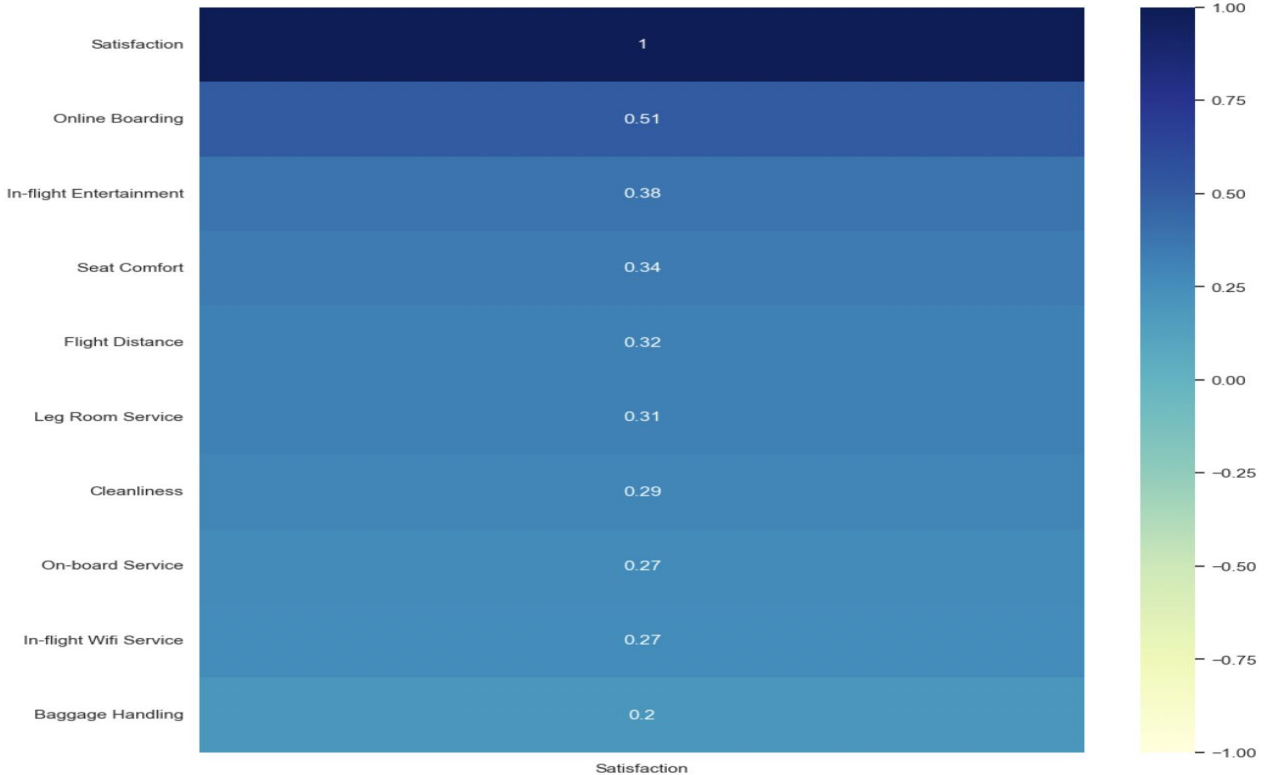
Data Visualization



From above data, we could see that features arrival delay and departure delay are highly correlated.

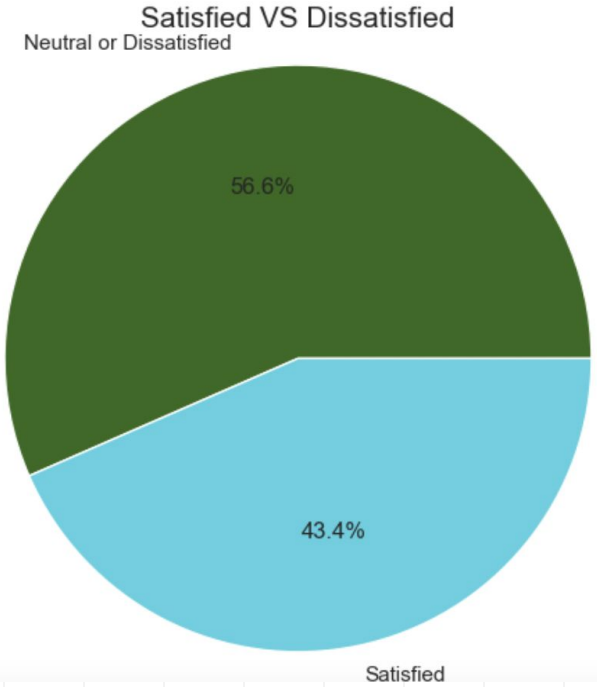
Data Visualization

Correlation of target feature ‘ Satisfaction’ with other features in dataset



Data Visualization

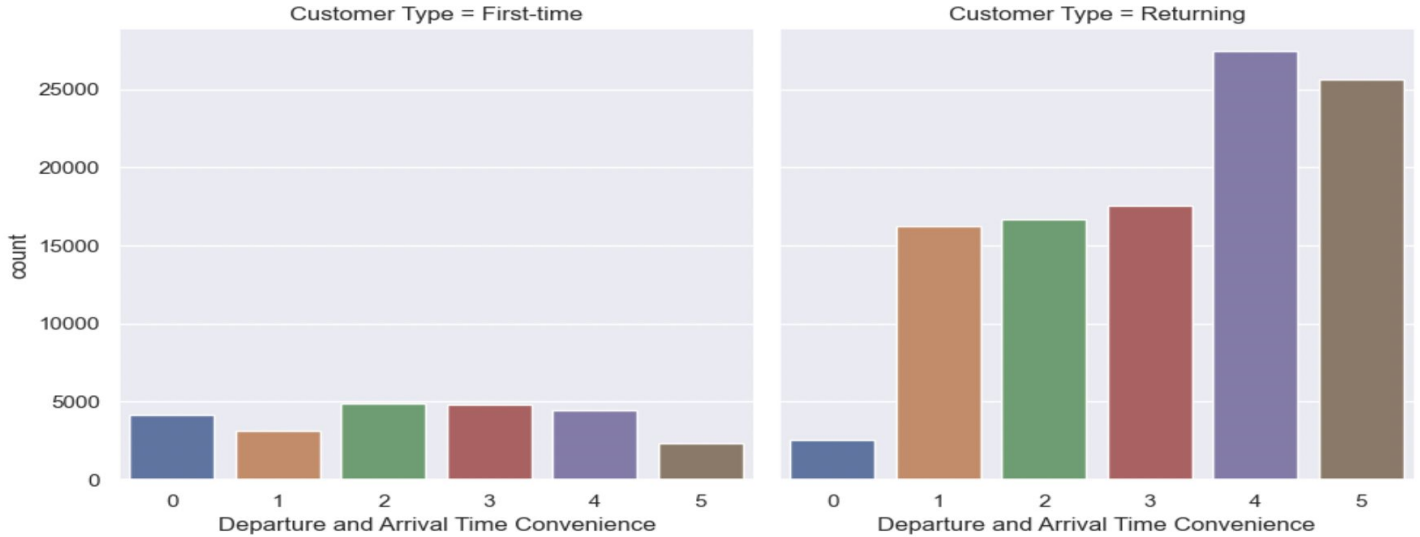
From the data, it can be seen that 56.6 % of passengers are neutral or dissatisfied whereas 43.4% of passengers are satisfied.



Data Visualization

From below graph, it is clear that people who travel frequently ie. returning passengers are more happy than first timers with Departure & Arrival Time Convenience.

```
In [28]: sns.catplot(kind='count',data=df,x='Departure and Arrival Time Convenience',col='Customer Type')
Out[28]: <seaborn.axisgrid.FacetGrid at 0x13a91beb0>
```



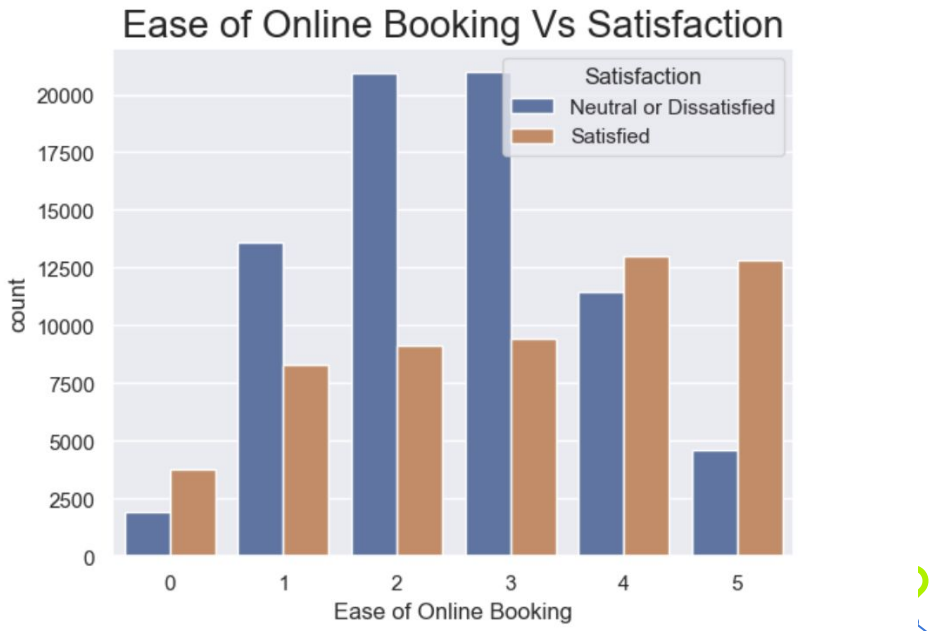
We see that the people who travel frequently, most of them are happy about the arrival and departure dates, as for the first time people, the largest percentage ranges from 2 to 4

Data Visualization

It is found that the most satisfied people are happy with the online booking and the largest percentage of the people are satisfied with the reservation average

```
sns.countplot(x='Ease of Online Booking',data=df,hue='Satisfaction')
plt.title(label='Ease of Online Booking Vs Satisfaction',fontsize=20)
```

```
Text(0.5, 1.0, 'Ease of Online Booking Vs Satisfaction')
```



Data Visualization

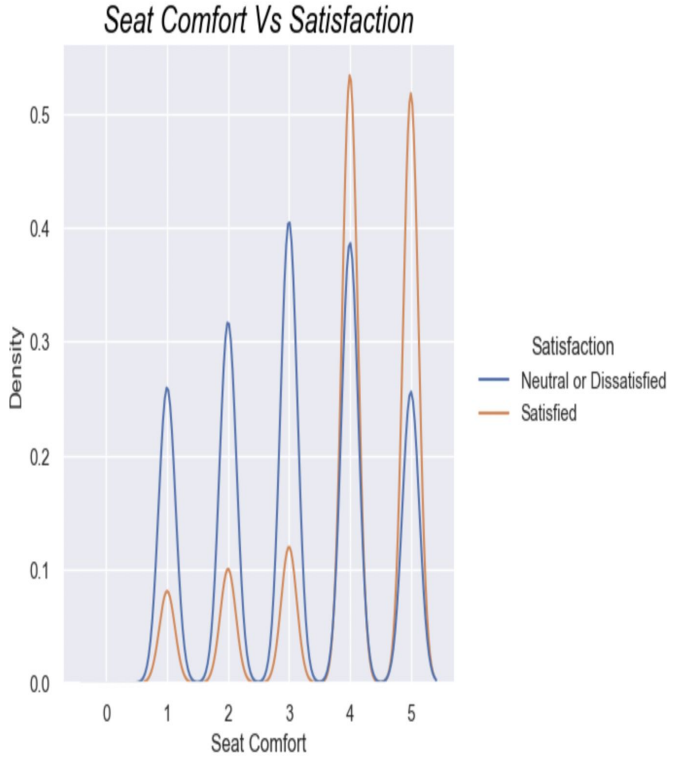
It is observed that, there is a large percentage of people who are satisfied category happy with the service as well as people in dissatisfied or neutral category seems to be variable. Hence, scope of improvement.

```
In [32]: sns.catplot(kind='count',data=df,x='On-board Service',hue='Satisfaction')
         plt.title(label='On-board Service Vs Satisfaction',fontsize=18,color='black',fontstyle='italic')
Out[32]: Text(0.5, 1.0, 'On-board Service Vs Satisfaction')
```

There is a large percentage of people who are satisfied category happy with the service as well as people in dissatisfied or neutral category seems to be variable.

Data Visualization

It is found that most of the people think that the seats are comfortable. Like the rest of the previous comparisons, a good number of dissatisfied people are happy with the seats, so the problem is not here




```
In [35]: df2 = df.groupby(['Satisfaction', 'Seat Comfort']).agg({'Seat Comfort': 'sum'})
         df3 = df2.groupby(level=0,group_keys=False).apply(lambda x:100 * x / float(x.sum()))
         print(df3)
```

		Seat Comfort
Satisfaction	Seat Comfort	
Neutral or Dissatisfied	0	0.000000
	1	5.258417
	2	12.853411
	3	24.648326
	4	31.335398
	5	25.904449
Satisfied	1	1.507482
	2	3.742343
	3	6.693862
	4	39.809308
	5	48.247005

Most of the people think that the seats are comfortable. Like the rest of the previous comparisons, a good number of dissatisfied people are happy with the seats, so the problem is not here

Data Visualization

It is observed that, around 41% of satisfied people and around 25% of Neutral or Dissatisfied are happy with highest amount cleanliness(5).

```
sns.displot(df,x='Cleanliness',hue='Satisfaction',kind='kde')
plt.title(label='Cleanliness/Satisfaction',fontsize=20,color='black',fontstyle='ital
plt.show()
sns.countplot(x='Cleanliness', hue="Satisfaction", data=df).set_title('Cleanliness V
```


```
df2 = df.groupby(['Satisfaction', 'Cleanliness']).agg({'Cleanliness': 'sum'})
df3 = df2.groupby(level=0,group_keys=False).apply(lambda x:100 * x / float(x.sum()))
print(df3)
```

		Cleanliness
Satisfaction	Cleanliness	
Neutral or Dissatisfied	0	0.000000
	1	6.236190
	2	14.702442
	3	24.189955
	4	29.312042
	5	25.559372
Satisfied	1	1.558598
	2	4.046204
	3	18.829467
	4	34.403618
	5	41.162113

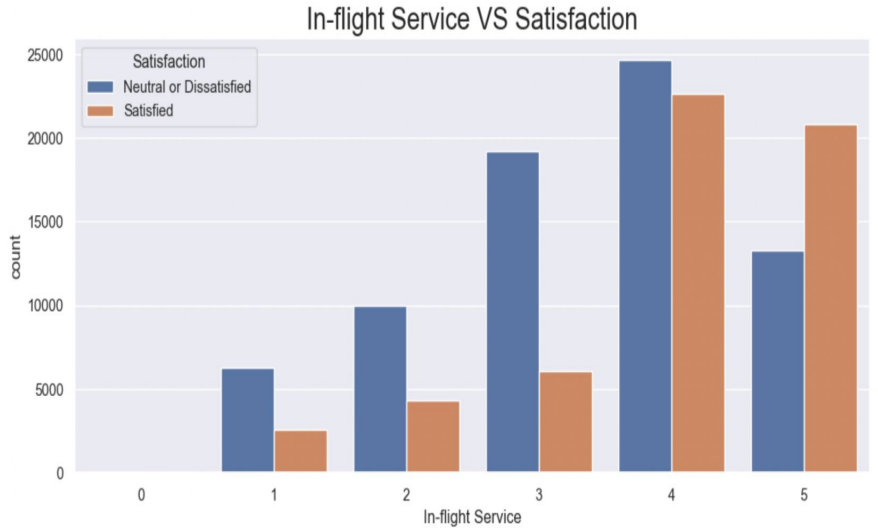
Around 41% of satisfied people and around 25% of Neutral or Dissatisfied are happy with highest amount cleanliness(5).

Data Visualization

From the graph, it is observed that most people who are satisfied or are neutral & dissatisfied are happy with In-flight Service.

The relationship of the satisfaction with in-flight service

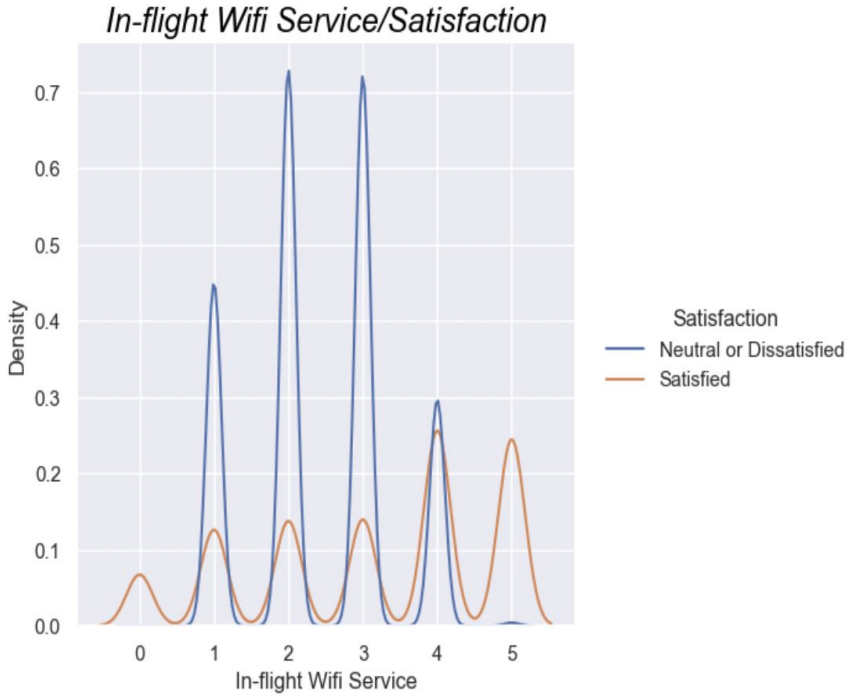
```
plt.figure(figsize=(12,5))
sns.countplot(x='In-flight Service', hue="Satisfaction", data=df).set_title('In-flight Service VS Satisfaction', fon
```



From the above graph, it is observed that most people who are satisfied or are neutral & dissatisfied are happy with In-flight Service.

Data Visualization

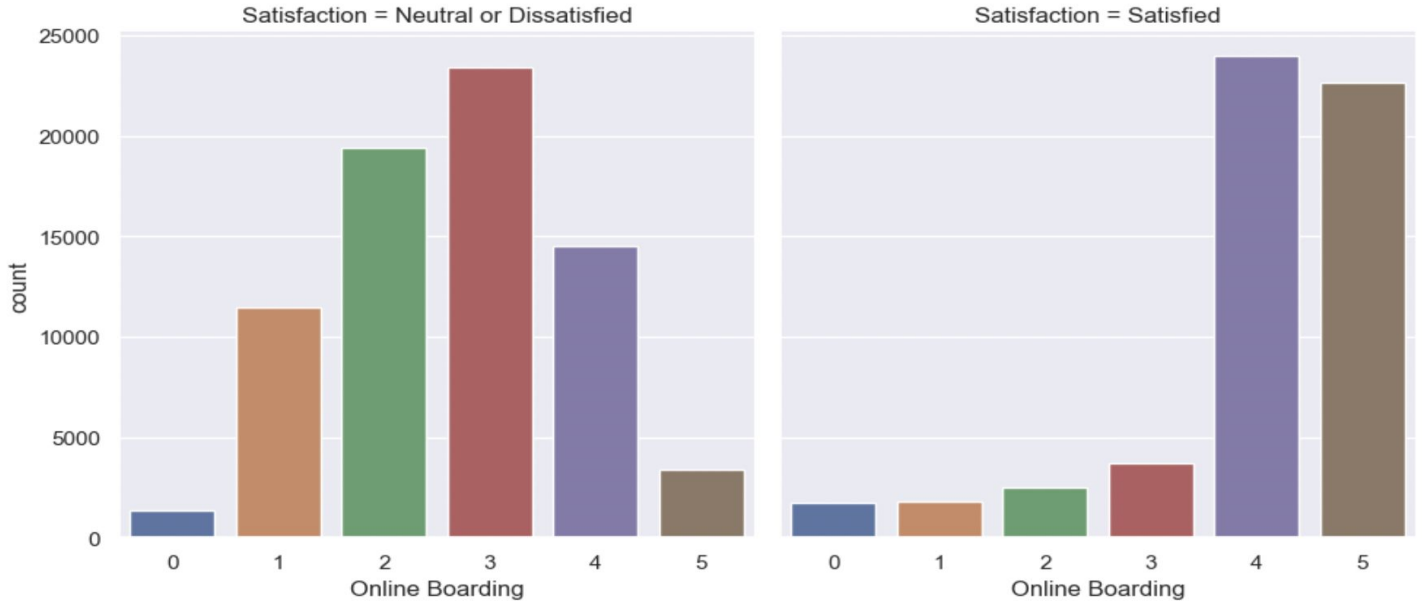
It is observed from below data that most of the people are not happy with the In-flight Wifi Service especially people who are neutral or dissatisfied around 45%.



		Gender	Age	Customer Type
Satisfaction	In-flight Wifi Service			
Neutral or Dissatisfied	0	0.007699	0.007699	0.007699
	1	11.549122	11.549122	11.549122
	2	18.732676	18.732676	18.732676
	3	18.540191	18.540191	18.540191
	4	7.615491	7.615491	7.615491
	5	0.108562	0.108562	0.108562
Satisfied	0	3.007391	3.007391	3.007391
	1	5.642131	5.642131	5.642131
	2	6.151832	6.151832	6.151832
	3	6.240376	6.240376	6.240376
	4	11.459809	11.459809	11.459809
	5	10.944718	10.944718	10.944718

Data Visualization

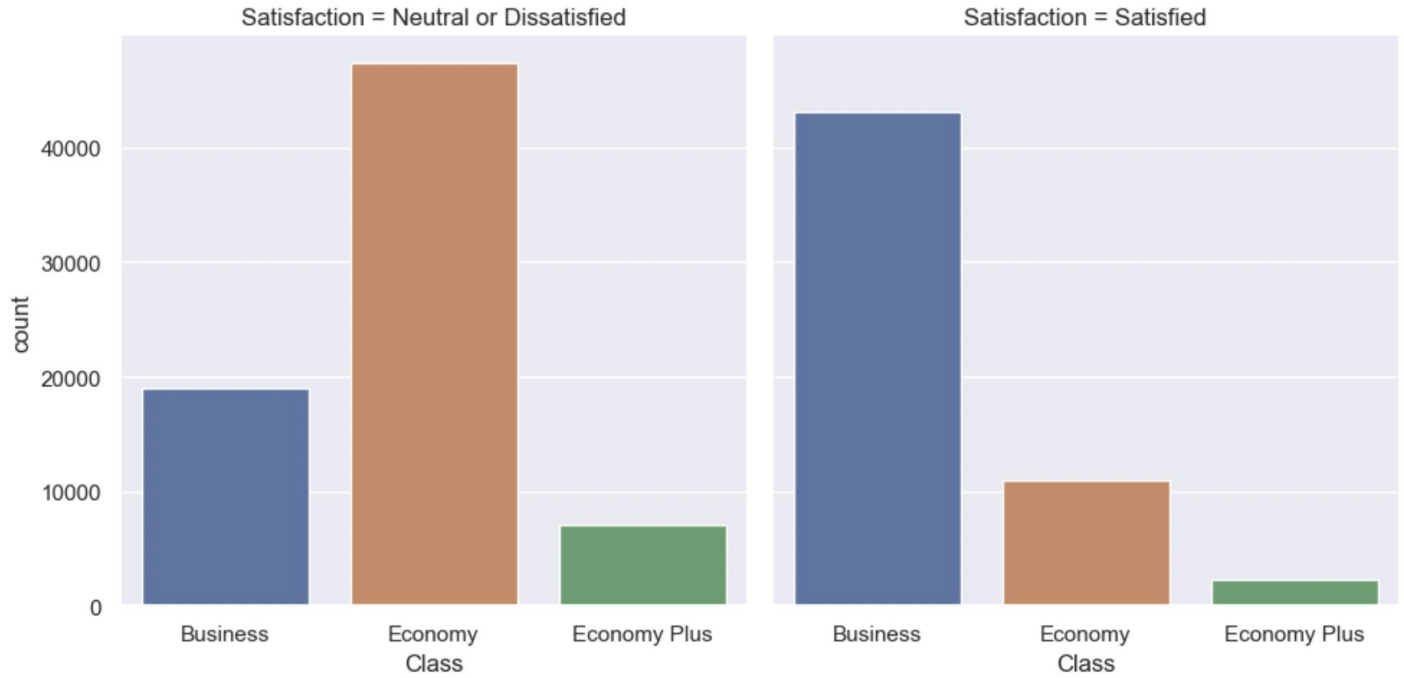
It is observed that, most of satisfied people who used online-boarding are happy whereas most of the people who unsatisfied are mostly neutral(3) or inclined to unhappy. Hence, it could be one of the factor for people to be satisfied or unsatisfied.



Most of satisfied people who viewed the entertainment are happy whereas most of the people who unsatisfied are mostly neutral(3) or inclined to unhappy. Hence, it could be one of the factor for people to be satisfied or unsatisfied.

Data Visualization

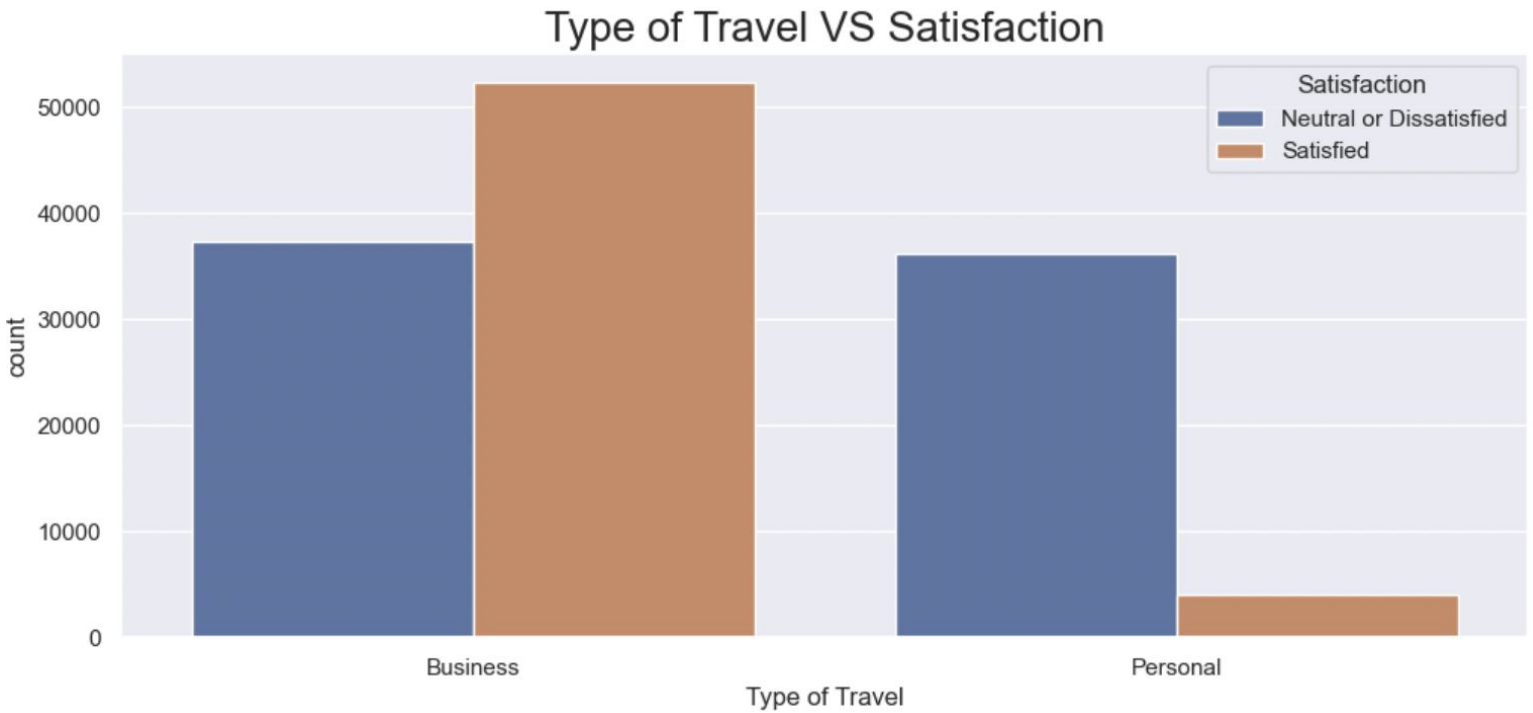
It is observed that, in general people travelling in business class are more happy compared to other classes and people travelling in Economy plus class are least happy



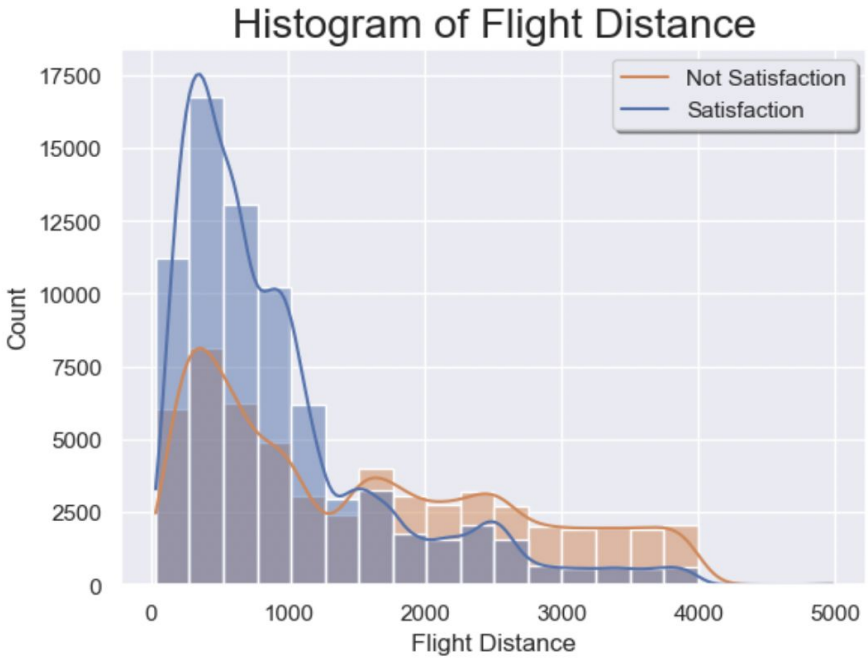
In general people travelling in business class are more happy compared to other classes and people travelling in Economy plus class are least happy

Data Visualization

It is observed that, People mostly travelling are with business purpose than with personal type of travel.



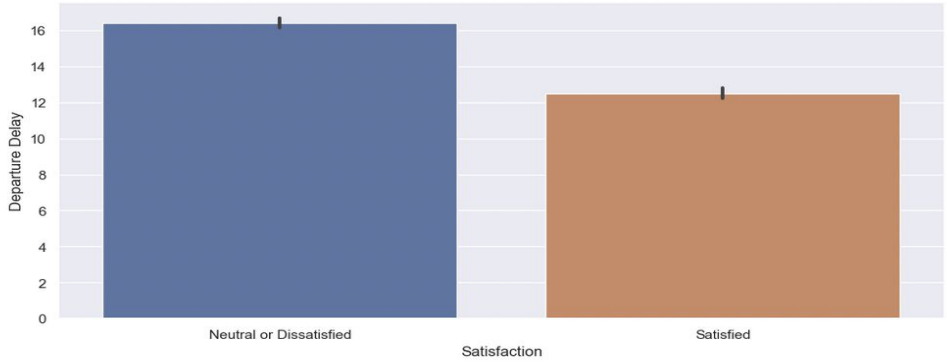
Data Visualization

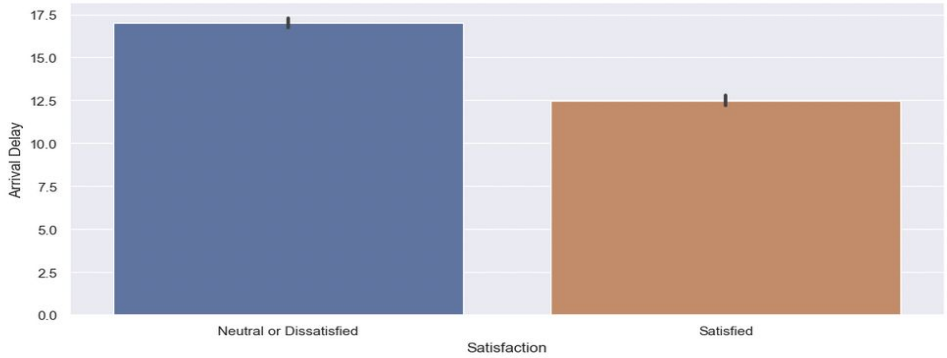


In this chart where we check all customers, you can see that a large proportion of the flights are in the range of 0-1250 (distance). Twice or more clients, depending on the specific distance, are neutral or dissatisfied. Most of them are in the range of 250 - 750.

Data Visualization

It is observed that, people in general are unhappy when there is delay in departure or arrival.





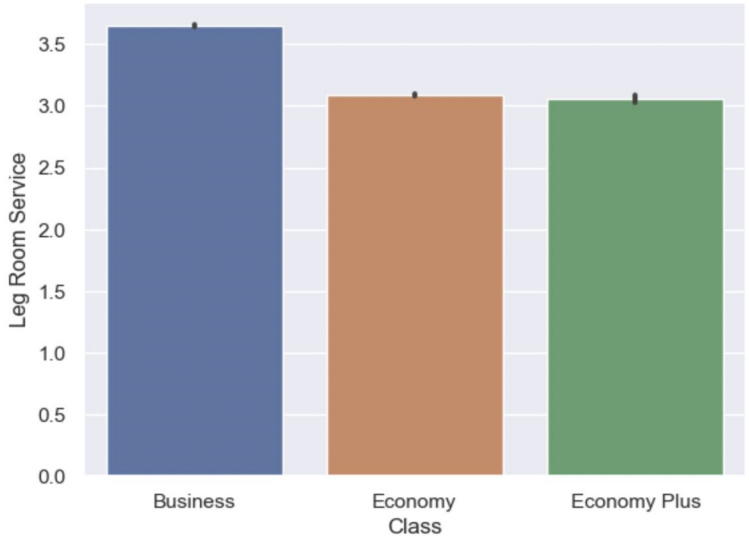
People are unhappy when there is delay in departure or arrival.

Data Visualization

We can see here that the business class probably have more leg space as the average of business class rating in this service is slightly higher

Relationship between Leg Room Service and Class

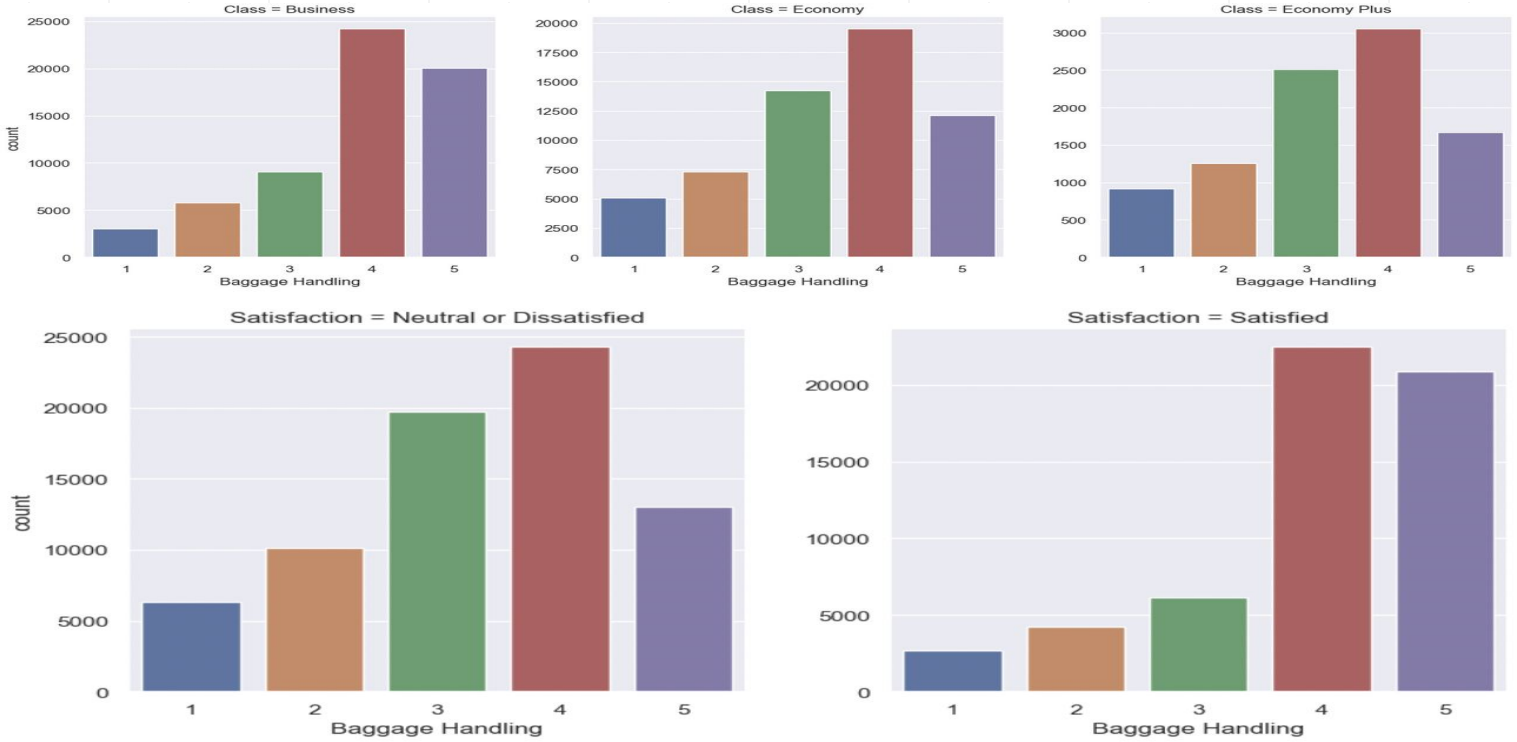
```
: sns.barplot(x='Class',y='Leg Room Service',data=df)
: <AxesSubplot: xlabel='Class', ylabel='Leg Room Service'>
```



We can see here that the businesss class probably have more leg space as the average of business class rating in this service is slightly higher

Data Visualization

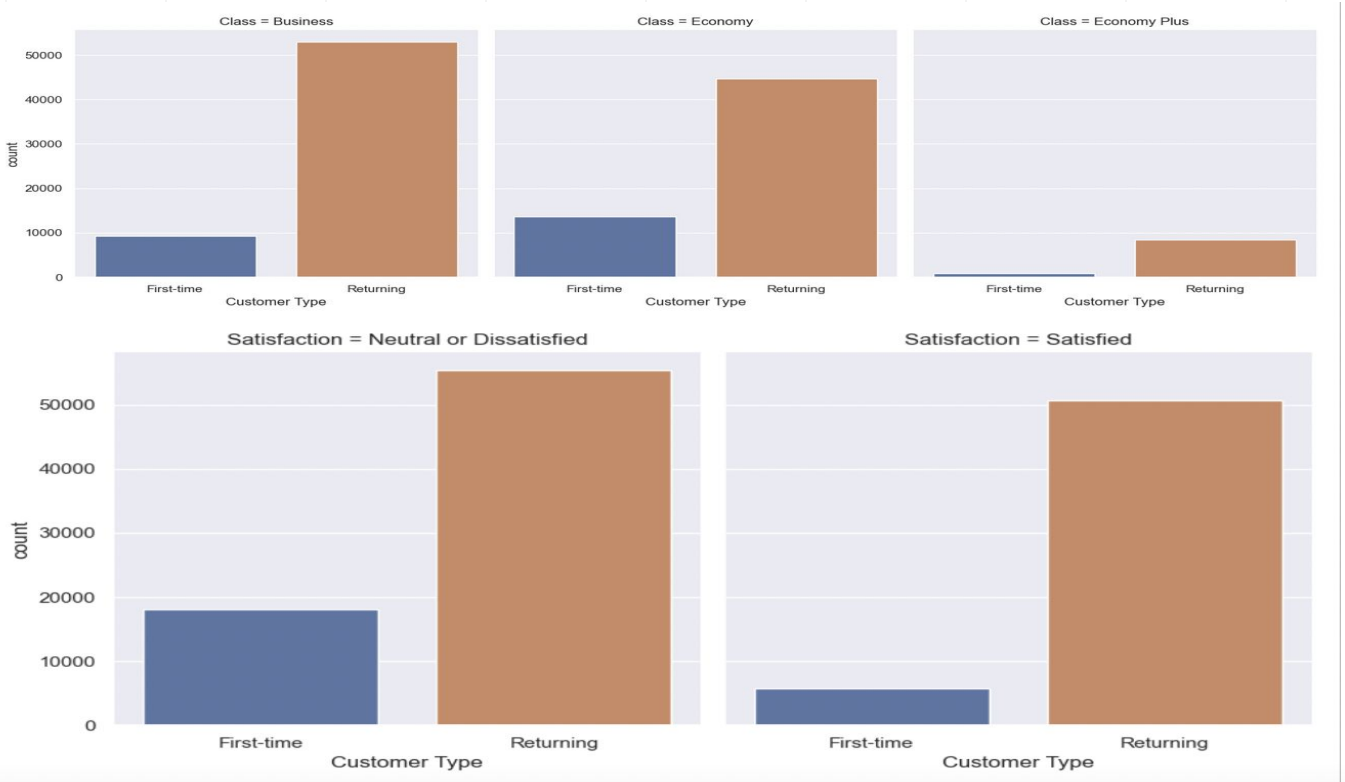
It Looks like most of the people are pleased with this service irrespective of class



Looks like most of the people are pleased with this service irrespective of class

Data Visualization

We can see here that the returning passengers are larger than the first-time passengers especially in business class following by economy class.



Data Preparation & Cleaning

There were outliers present in data, hence removing those using IQR

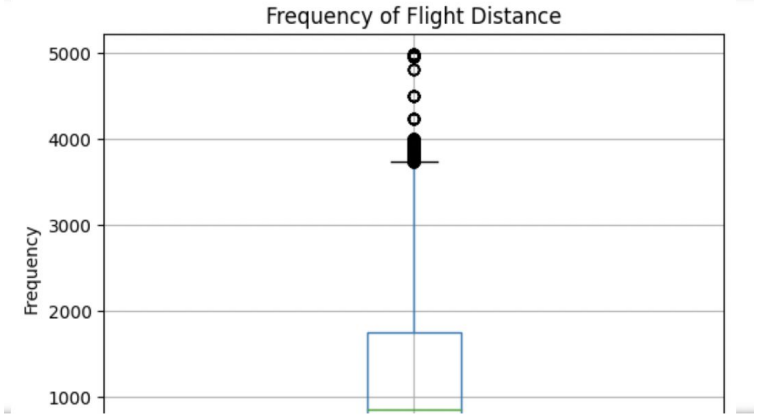
Remove the outlier datapoints from the dataset.

```
def remove_outliers(columns):
    Q1 = df[columns].quantile(0.25)
    Q3 = df[columns].quantile(0.75)
    IQR = Q3 - Q1    #IQR is interquartile range.

    filter = (df[columns] >= Q1 - 1.5 * IQR) & (df[columns] <= Q3 + 1.5 *IQR)
    return df.loc[filter]



showfliers=False
for col in columns:
    df = remove_outliers(col)
```

```
df.shape
```

```
(74621, 23)
```

```
def plotboxplt(col):
    plt.ylabel("Frequency")
    plt.title(f"Frequency of {col}")
    df.boxplot(column=[col])
    plt.show()

num_df = df.select_dtypes(include='number')
columns = num_df.columns
for col in columns:
    plotboxplt(col)
```



Data Preparation & Cleaning

As one can see, there were categorical data present in dataset, hence, using LabelEncoder to encode them accordingly. Eg. Class feature has 3 types - Business, Economy & Economy Plus. It will be encoded into 0, 1, & 2 respectively.

```
df['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
df['Customer Type'].unique()
```

```
array(['First-time', 'Returning'], dtype=object)
```

```
df['Type of Travel'].unique()
```

```
array(['Business', 'Personal'], dtype=object)
```

```
df['Class'].unique()
```

```
array(['Business', 'Economy', 'Economy Plus'], dtype=object)
```

```
df['Satisfaction'].unique()
```

```
array(['Neutral or Dissatisfied', 'Satisfied'], dtype=object)
```

Encoding data

```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
df['Gender'] = labelencoder.fit_transform(df['Gender'])
df['Customer Type'] = labelencoder.fit_transform(df['Customer Type'])
df['Type of Travel'] = labelencoder.fit_transform(df['Type of Travel'])
df['Class'] = labelencoder.fit_transform(df['Class'])
df['Satisfaction'] = labelencoder.fit_transform(df['Satisfaction'])
```

```
print(df[['Gender', 'Customer Type', 'Type of Travel', 'Class', 'Satisfaction']].head(5))
```

	Gender	Customer Type	Type of Travel	Class	Satisfaction
0	1	0	0	0	0
2	1	1	0	0	1
3	1	1	0	0	1
4	0	1	0	0	1
6	1	1	0	0	1

```
df['Class'].unique()
```

```
array([0, 1, 2])
```

Model fitting & evaluation of model

Using KNN Model:

It is observed that, using KNN as classifier the model gives an accuracy of 92.16%. Also, from classification report, it is seen that precision is about 90 & 95 percent which is quite decent.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

mymodelknm = KNeighborsClassifier()

mymodelknm.fit(X_train , y_train)
knn_train_score=mymodelknm.score(X_train , y_train)
knn_test_score=mymodelknm.score(X_test , y_test)
knny_pred = mymodelknm.predict(X_test)
print(knn_test_score)
```

```
0.9216750418760469
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,knny_pred))
print('accuracy',accuracy_score(y_test, knny_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      7369
           1       0.95      0.89      0.92      7556

    accuracy                           0.92     14925
   macro avg       0.92      0.92      0.92     14925
weighted avg       0.92      0.92      0.92     14925

accuracy 0.9216750418760469
```

Model fitting & evaluation of model

Using Decision Tree Model:

It is observed that, using Decision Tree as classifier the model gives an accuracy of 94.37%.
Also, from classification report, it is seen that precision is about 94 & 95 percent which is quite decent and much better than KNN model.

```
from sklearn.tree  import DecisionTreeClassifier

mymodeldt = DecisionTreeClassifier()

mymodeldt.fit(X_train , y_train)
dt_train_score=mymodeldt.score(X_train , y_train)
dt_test_score=mymodeldt.score(X_test , y_test)
dty_pred = mymodeldt.predict(X_test)
print(dt_test_score)
```

```
0.9437185929648241
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,dty_pred))
print('accuracy',accuracy_score(y_test, dty_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      7369
           1       0.95      0.94      0.94      7556

    accuracy                           0.94     14925
   macro avg       0.94      0.94      0.94     14925
weighted avg       0.94      0.94      0.94     14925

accuracy 0.9437185929648241
```

Model fitting & evaluation of model

Using Random Forest Classifier Model:

It is observed that, using Random Forest Classifier as classifier the model gives an accuracy of 95.65%.
Also, from classification report, it is seen that precision is about 94 & 97 percent which is much better than KNN model and Decision Tree model.

Hence, random forest classifier gives more accuracy and precision. Hence, moving forward with random forest classifier to check if its performance could be increased more.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

mymodelrf = RandomForestClassifier()

mymodelrf.fit(X_train , y_train)
rfc_train_score=mymodelrf.score(X_train , y_train)
rfc_test_score=mymodelrf.score(X_test , y_test)
rfcy_pred = mymodelrf.predict(X_test)
print(rfc_test_score)
```

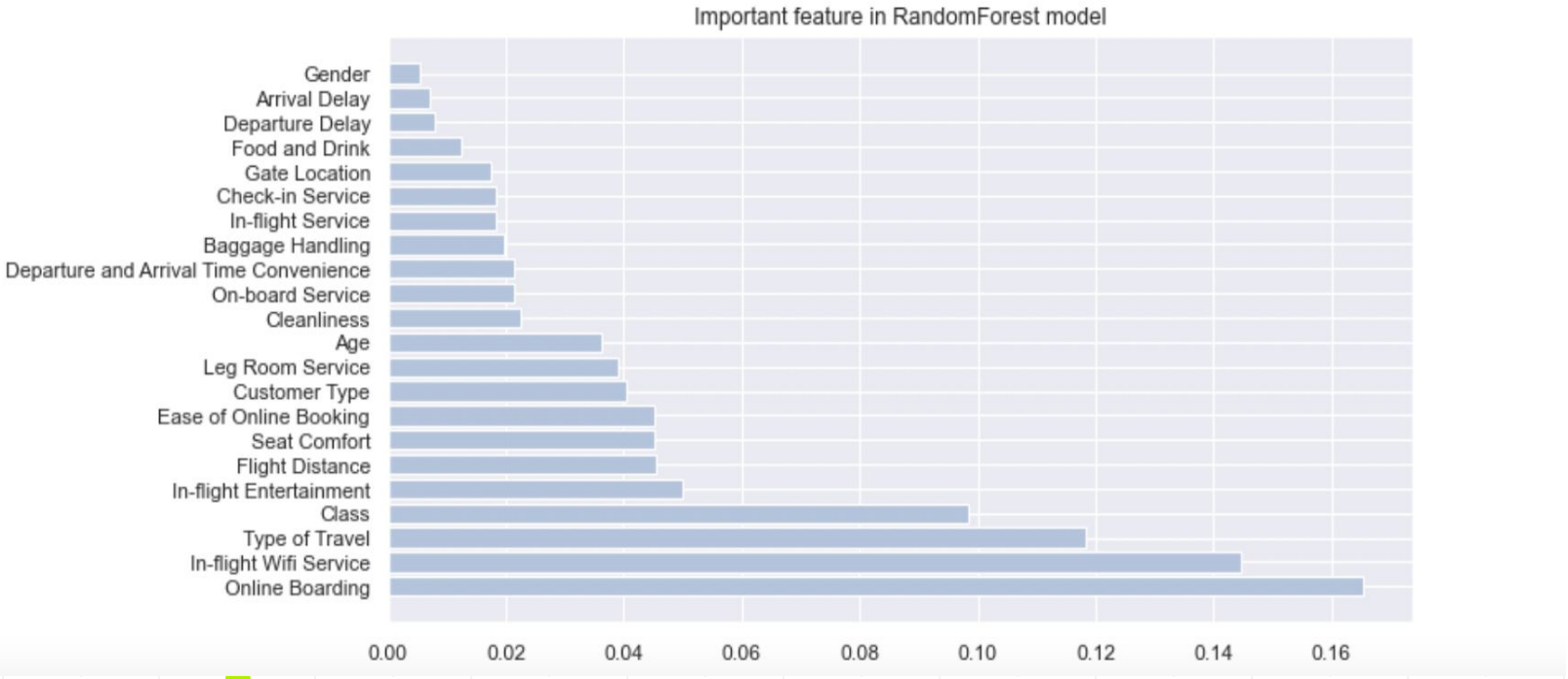
```
0.9565159128978225
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,rfcy_pred))
print('accuracy',accuracy_score(y_test, rfcy_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.96      7369
           1       0.97      0.94      0.96      7556

    accuracy                           0.96     14925
   macro avg       0.96      0.96      0.96     14925
weighted avg       0.96      0.96      0.96     14925

accuracy 0.9565159128978225
```

Model fitting & evaluation of model

Feature importance using Random Forest Classifier



Model fitting & evaluation of model

Using RFE, retrieved the 11 most important features. It is observed that these features match the most important features from previous graph.
After using 11 most important features to train model using random forest classifier, got an accuracy of 94.35 % which is less than that of base model of random forest classifier.

I is clear that the model built using random forest classifier with all features has better performance than using important features.

Hence, for further processing, using base model.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

mymodelrf1 = RandomForestClassifier()

mymodelrf1.fit(X_new_train , y_new_train)
rfc_train_score1=mymodelrf1.score(X_new_train , y_new_train)
rfc_test_score1=mymodelrf1.score(X_new_test , y_new_test)
rfcy_pred1 = mymodelrf1.predict(X_new_test)
print(rfc_test_score1)
```

```
0.9435845896147403
```

```
print('accuracy',accuracy_score(y_new_test,rfcy_pred1))
print(classification_report(y_test,rfcy_pred1))
```

```
accuracy 0.9435845896147403
              precision    recall  f1-score   support

           0       0.93      0.96      0.94      7369
           1       0.96      0.93      0.94      7556

    accuracy                           0.94     14925
   macro avg       0.94      0.94      0.94     14925
weighted avg       0.94      0.94      0.94     14925
```

HyperParameter Tuning

Using RandomizedSearchCV for tuning the hyperparameter of RandomForestClassifier to check if the performance improves after hyperparameter tuning.

After processing, got the below hyperparameters for tuning.

```
rf_random.best_params_
```

```
{'n_estimators': 733,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 80,
 'bootstrap': False}
```

HyperParameter Tuning

After building a model using Random Forest Classifier and the hyperparameter obtained from RandomSearchCV, the accuracy 95.93 % which is better than previous model.

There is a improvement of 0.3% accuracy but computational time went high.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

mymodelrf1 = RandomForestClassifier(max_depth = 80, max_features = 'sqrt', n_estimators = 733, min_samples_leaf =1,
                                    min_samples_split = 5,bootstrap = False)
mymodelrf1.fit(X_train , y_train)
rfc_train_score1=mymodelrf1.score(X_train , y_train)
rfc_test_score1=mymodelrf1.score(X_test , y_test)
rfcy_predh = mymodelrf1.predict(X_test)
print(rfc_test_score1)
```

```
0.9591959798994975
```

```
print('accuracy',accuracy_score(y_test, rfcy_predh))
print(classification_report(y_test,rfcy_predh))
```

```
accuracy 0.9593299832495812
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      7369
           1       0.97      0.95      0.96      7556

    accuracy                           0.96     14925
   macro avg       0.96      0.96      0.96     14925
weighted avg       0.96      0.96      0.96     14925
```

```
print('There is improvement of ',(accuracy_score(y_test,rfcy_predh) - accuracy_score(y_test,rfcy_pred))*100,'%')
```

```
There is improvement of  0.3082077051926313 %
```

HyperParameter Tuning

Actual & predicted values
after hyperparameter tuning

```
data = pd.DataFrame({"Y_test": y_test , "Y_pred" : rfcy_predh})
data.head(20)
```

	Y_test	Y_pred
12247	1	1
40165	0	0
68949	1	1
103146	1	1
1108	0	0
23184	0	0
71331	1	1
48507	0	0
33101	0	0
8325	0	0
53509	0	0
55610	1	1
127253	1	1
63720	1	1
95075	0	0
21718	1	1
59419	1	1
77065	0	0
109513	0	0
115488	0	0

Summary

It is observed that , random forest with hyperparameter tuning gives the best performance among above model like using KNN, Decision Tree, basic random forest, random forest with important features & random forest with hyperparameter tuning.

But given the computational cost of hyperparameter tuning and the improvement in accuracy is not that high; basic random forest classifier would be appropriate and better model for this dataset from the above model evaluation.

Future Improvement

- Using another classifiers like XGBoost, AdaBoost to improve accuracy, precision and also computation.

Thank you