***<*** ***Predicting Customer Satisfaction >***

**<5/26/2024>**

*Prepared by*

<Vijay Laxmi Tyagi>

**<A01317295>**

Table of Contents

[Predicting Airline Passenger Satisfaction with Python 3](#_Toc167656857)

[Exploratory data analysis 3](#_Toc167656858)

[Attributes 5](#_Toc167656859)

[Visualization 6](#_Toc167656860)

[Correlation 6](#_Toc167656861)

[Impute Null Values 8](#_Toc167656862)

[Binning 8](#_Toc167656863)

[Convert non-numeric columns into dummy variables 8](#_Toc167656864)

[Model Selection 9](#_Toc167656865)

[Model Comparison Table 11](#_Toc167656866)

[Scaling 11](#_Toc167656867)

[Scaling Comparison Table 13](#_Toc167656868)

[K-Fold Technique 13](#_Toc167656869)

[K-Fold Comparison Table 16](#_Toc167656870)

[Final Code for K Fold 5 is as follows 16](#_Toc167656871)

# Predicting Airline Passenger Satisfaction with Python

In this assignment my aim is to build a model to predict if the customer is satisfied or not with the help of python libraries.

## Exploratory data analysis

import pandas as pd  
# Import data.  
PATH = "/Users/vipul/Documents/bcit/Comp4254AdvancedDataAnalytics/Data/"  
FILE = "airlinePassengerSatisfaction.csv"  
df = pd.read\_csv(PATH + FILE)  
pd.set\_option('display.max\_columns', None)  
#Exploratory data Analysis  
print(df.shape) # check dimension  
print(df.info()) # check data type and Missing Values  
print(df.describe()) # summary statistics  
print(df.head(10)) # display first 10 rows  
print(df.tail(10)) # display last 10 rows

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**data type and Missing Values:-**

A screenshot of a computer

Description automatically generated

### Attributes

Data set has 23 Attributes which are as follows: -

**Gender:** Gender of passengers (Female, Male)

**Customer Type:** Types of Customers (Loyal customer, disloyal customer)

**Age:** The actual age of the passengers

**Type of Travel:** Type of Travel of the passengers (Personal Travel, Business Travel)

**Class:** Class of Travel in the plane of the passengers (Business, Eco, Eco Plus)

**Flight distance:** The flight distance from Source to Destination

**Inflight wifi service:** Satisfaction level of the inflight wifi service (0:Not Applicable;1-5)

**Departure/Arrival time convenient:** Satisfaction level of Departure/Arrival time convenient

**Ease of Online booking:** Satisfaction level of online booking

**Gate location:** Satisfaction level of Gate location

**Food and drink:** Satisfaction level of Food and drink

**Online boarding:** Satisfaction level of online boarding

**Seat comfort:** Satisfaction level of Seat comfort

**Inflight entertainment:** Satisfaction level of inflight entertainment

**On-board service:** Satisfaction level of On-board service

**Leg room service:** Satisfaction level of Leg room service

**Baggage handling:** Satisfaction level of baggage handling

**Check-in service:** Satisfaction level of Check-in service

**Inflight service:** Satisfaction level of inflight service

**Cleanliness:** Satisfaction level of Cleanliness

**Departure Delay in Minutes:** Delayed time when departure

**Arrival Delay in Minutes:** Delayed Arrival time

**Satisfaction:** Airline satisfaction level(Satisfaction, neutral or dissatisfaction)

## Visualization

import matplotlib.pyplot as plt  
import pandas as pd  
import seaborn as sns  
# Import data.  
PATH = "/Users/vipul/Documents/bcit/Comp4254AdvancedDataAnalytics/Data/"  
FILE = "airlinePassengerSatisfaction.csv"  
df = pd.read\_csv(PATH + FILE)  
pd.set\_option('display.max\_columns', None)  
palette = ['#46cdcf','#3d84a8','#48466d' ]  
sns.set\_palette(palette)  
df.hist(bins=24, figsize=(25,10), color= palette[1])  
plt.show()

A screenshot of a computer

Description automatically generated

## Correlation

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.preprocessing import LabelEncoder  
# Import data  
PATH = "/Users/vipul/Documents/bcit/Comp4254AdvancedDataAnalytics/Data/"  
FILE = "airlinePassengerSatisfaction.csv"  
df = pd.read\_csv(PATH + FILE)  
  
# Impute null values with mean for Arrival Delay in Minutes  
mean\_arrival\_delay = df['Arrival Delay in Minutes'].mean()  
df['Arrival Delay in Minutes'].fillna(mean\_arrival\_delay, inplace=True)  
  
# Encode 'satisfaction' column  
label\_encoder = LabelEncoder()  
df['satisfaction\_encoded'] = label\_encoder.fit\_transform(df['satisfaction'])  
  
# Drop non-numeric columns before calculating correlation  
numeric\_df = df.select\_dtypes(include=['number'])  
  
# Plot the heatmap  
heatmap = sns.heatmap(numeric\_df.corr().loc[['satisfaction\_encoded'], :], vmin=-1, vmax=1, annot=True)  
heatmap.set\_title('Correlation with Satisfaction', fontdict={'fontsize':18}, pad=16)  
plt.show()

**# Calculate sorted correlation values**sorted\_correlation = df.\_get\_numeric\_data().corr()['satisfaction\_encoded'].drop('satisfaction\_encoded').abs().sort\_values(ascending=False)  
print(sorted\_correlation)

A graph of different colors

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| Online boarding | 0.505766 |
| Inflight entertainment | 0.398088 |
| Seat comfort | 0.351974 |
| On-board service | 0.323120 |
| Leg room service | 0.307331 |
| Cleanliness | 0.302745 |
| Flight Distance | 0.291604 |
| Inflight wifi service | 0.285709 |
| Baggage handling | 0.250012 |
| Inflight service | 0.247294 |
| Checkin service | 0.237167 |
| Food and drink | 0.211287 |
| Ease of Online booking | 0.170895 |
| Age | 0.134781 |
| Arrival Delay in Minutes | 0.058467 |
| Departure Delay in Minutes | 0.051549 |
| Departure/Arrival time convenient | 0.050156 |
| Gate location | 0.003748 |

## Impute Null Values

We found out that Arrival Delay in minutes has 51795 values whereas other attributes have 51952 values so to impute null values in Arrival delay in minutes we use mean arrival delay time. Code to do the same is as follows: -

# Impute null values with mean for Arrival Delay in Minutes  
mean\_arrival\_delay = df['Arrival Delay in Minutes'].mean()  
df['Arrival Delay in Minutes'] = df['Arrival Delay in Minutes'].fillna(mean\_arrival\_delay)

## Binning

# Bin the 'Age' attribute  
bins = [0, 20, 40, 60, 100]  
labels = ['0-20', '21-40', '41-60', '61-100']  
df['Age\_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

## Convert non-numeric columns into dummy variables

# Convert non-numeric columns into dummy variables  
non\_numeric\_columns = ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfaction', 'Age\_Group']  
df = pd.get\_dummies(df, columns=non\_numeric\_columns, drop\_first=True)

## Model Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R\_adj^2 | AIC | BIC | RMSE |
| Model A(all Predictors) | 0.552184 | 24609.425721 | 24840.826343 | 0.331901 |
| Model B(high-correlation predictors only) | 0.370787 | 37840.529964 | 37900.522718 | 0.389379 |
| Model C(highest correlation predictors) | 0.323536 | 40657.916920 | 40683.628100 | 0.404743 |
| Model D(Best P-Value Predictors) | 0.550918 | 24711.350737 | 24874.188212 | 0.332483 |

From above 4 models, Model D seems promising to Predict Customer Satisfaction, so I choose Model D.

## Scaling

Out of StandardScaler, MinMaxScaler, RobustScaler I compare and choose the promising one.

## Scaling Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
|  | MinMaxScaler | RobustScaler | StandardScaler |
| Accuracy | 0.8671850939328611 | 0.8671850939328611 | 0.8671850939328611 |
| Precision | 0.8533188641707361 | 0.8533188641707361 | 0.8533188641707361 |
| Recall | 0.8377130681818182 | 0.8377130681818182 | 0.8377130681818182 |
| F1 Score | 0.8454439566347102 | 0.8454439566347102 | 0.8454439566347102 |

## K-Fold Technique

## K-Fold Comparison Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | K-fold: 3 | K-fold: 4 | K-fold: 5 | K-fold: 6 | K-fold: 7 |
| RMSE | 0.32913984670757734 | 0.33778646555140374 | 0.3209382354382673 | 0.32795539219925635 | 0.33402126813175603 |
| BIC | 30994.10914520416 | 30813.079879720623 | 31160.508270591876 | 31017.475137229834 | 30892.221104999757 |
| R^2 | 0.5502069599697921 | 0.551758257588866 | 0.5490512755466295 | 0.5500350426223373 | 0.5511610803981357 |
| Accuracy | 0.8692263279445728 | 0.8611431870669746 | 0.8796189376443418 | 0.8744226327944573 | 0.8747113163972287 |
| Precision | 0.856 | 0.852882703777336 | 0.8616796047988708 | 0.8655804480651731 | 0.8698023176550784 |

Out of above tried K- Fold Techniques K-fold 5 seems promising. So, I choose k-fold 5.

## Final Code for K Fold 5 is as follows

import numpy as np  
import pandas as pd  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix  
from sklearn.model\_selection import train\_test\_split, KFold  
import statsmodels.api as sm  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn import metrics  
  
# Load the data  
PATH = "/Users/vipul/Documents/bcit/Comp4254AdvancedDataAnalytics/Data/"  
FILE = "airlinePassengerSatisfaction.csv"  
df = pd.read\_csv(PATH + FILE)  
pd.set\_option('display.max\_columns', None)  
  
# Impute null values with mean for Arrival Delay in Minutes  
mean\_arrival\_delay = df['Arrival Delay in Minutes'].mean()  
df['Arrival Delay in Minutes'] = df['Arrival Delay in Minutes'].fillna(mean\_arrival\_delay)  
  
# Bin the 'Age' attribute  
bins = [0, 20, 40, 60, 100]  
labels = ['0-20', '21-40', '41-60', '61-100']  
df['Age\_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)  
  
# Convert non-numeric columns into dummy variables  
non\_numeric\_columns = ['Gender', 'Customer Type', 'Type of Travel', 'Class', 'satisfaction', 'Age\_Group']  
df = pd.get\_dummies(df, columns=non\_numeric\_columns, drop\_first=True)  
  
# Define the predictor variables (features) and the target variable  
features = ['Age\_Group\_61-100', 'Inflight wifi service', 'Online boarding', 'Inflight entertainment',  
 'Seat comfort', 'On-board service', 'Departure/Arrival time convenient', 'Ease of Online booking',  
 'Leg room service', 'Baggage handling', 'Checkin service', 'Inflight service', 'Cleanliness',  
 'Arrival Delay in Minutes', 'Customer Type\_disloyal Customer', 'Type of Travel\_Personal Travel',  
 'Class\_Eco', 'Class\_Eco Plus']  
X = df[features]  
  
# Scale the features using StandardScaler  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# Adding an intercept to the scaled features  
X\_scaled = sm.add\_constant(X\_scaled)  
  
# Define the target variable  
y = df['satisfaction\_satisfied']  
  
# Split the data into training and testing sets (75% train, 25% test)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.25, random\_state=0)  
  
# Fit Model A: Include all predictors  
model = sm.OLS(y\_train.astype(float), X\_train.astype(float)).fit()  
predictions = model.predict(X\_test.astype(float))  
threshold = 0.5 # Threshold for converting predicted probabilities to binary predictions  
binary\_predictions = (predictions > threshold).astype(int)  
  
# K-fold cross-validation  
kfold = KFold(n\_splits=5, shuffle=True, random\_state=0)  
rmseList = []  
bicList = []  
rsquareLst = []  
accuracyList = []  
precisionList = []  
count = 1  
for train\_index, test\_index in kfold.split(X\_scaled):  
 X\_train\_cv = X\_scaled[train\_index]  
 X\_test\_cv = X\_scaled[test\_index]  
 y\_train\_cv = y.iloc[train\_index]  
 y\_test\_cv = y.iloc[test\_index]  
  
 # Perform linear regression  
 model\_cv = sm.OLS(y\_train\_cv.astype(float), X\_train\_cv.astype(float)).fit()  
 y\_pred\_cv = model\_cv.predict(X\_test\_cv.astype(float)) # Make the predictions by the model  
 mse = metrics.mean\_squared\_error(y\_test\_cv, y\_pred\_cv)  
 rmse = np.sqrt(mse)  
 rmseList.append(rmse)  
 bic = model\_cv.bic  
 bicList.append(bic)  
 rsqr = model\_cv.rsquared  
 rsquareLst.append(rsqr)  
  
 # Convert predicted probabilities to binary predictions  
 binary\_predictions\_cv = (y\_pred\_cv > threshold).astype(int)  
  
 # Calculate accuracy and precision  
 accuracy = accuracy\_score(y\_test\_cv, binary\_predictions\_cv)  
 precision = precision\_score(y\_test\_cv, binary\_predictions\_cv)  
 accuracyList.append(accuracy)  
 precisionList.append(precision)  
 print("\n\*\*\*K-fold: " + str(count))  
 print("RMSE: " + str(rmse))  
 print("BIC: " + str(bic))  
 print("R^2: " + str(rsqr))  
 print("Accuracy: " + str(accuracy))  
 print("Precision: " + str(precision))  
 count += 1  
  
# Show averages of scores over multiple runs.  
print("\nAverage Scores Across All Folds:")  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
print("Average RMSE: " + str(np.mean(rmseList)))  
print("RMSE std: " + str(np.std(rmseList)))  
print("Average BIC: " + str(np.mean(bicList)))  
print("BIC std: " + str(np.std(bicList)))  
print("Average R^2: " + str(np.mean(rsquareLst)))  
print("R^2 std: " + str(np.std(rsquareLst)))  
print("Average accuracy: " + str(np.mean(accuracyList)))  
print("Accuracy std: " + str(np.std(accuracyList)))  
print("Average precision:" + str(np.mean(precisionList)))  
print("Precision std: " + str(np.std(precisionList)))  
print(model.summary())