AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH

Assignment Cover Sheet

Students must complete all details except the faculty use part.

Please submit all assignments to your subject lecturers or the office of the concerned lecturer.

Assignment Title: Classification Based Machine Learning Model Development

Assignment Number: 01 Due Date: 13/12/2022 Semester: 2022-2023,Fall
Subject Code: CSC4162 Subject Name: Programming in Python Section: A

Course Instructor: Akinul Islam Jony Degree Program: BSc CSE

Declaration and Statement of Authorship:

1. I/we hold a copy of this assignment, which can be produced if the original is lost/ damaged.
2. This assignment is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this assignment has been written for me/us by any other person except where such collaboration has been authorized by the lecturer/teacher concerned and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the School for review and comparison, including review by external examiners.

I/we understand that

7. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.
8. Enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy your work

Group Name (if applicable):

No.	Student Name	Student Number	Student Signature	Date
1	A.S.M. FAZLE RABBI	19-39714-1		13/12/22
2				
3				
4				
5				
6				

For faculty use only:

Total Marks: _____ Marks Obtained: _____

Faculty comments_____

_____

_____

1. Project Overview

In this project our goal is to develop five classification-based machine learning model. These models are Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbor (KNN), Logistic regression (LR) and Naive Bayes (NB). In classification-based machine learning after loading the dataset and our first job is to preprocess the data i.e., is cleaning the data set if there are any empty cells, wrong data, data in wrong format or duplicate data. For empty cells depending on size of the dataset we either refill the value or remove the rows or columns. We usually use Mean, Median, Mode value to refill the empty cells. Depending on the size of dataset if the number of missing values is more than 70% it's wise to remove the rows or columns since the data provided is insufficient to train and test the model. If there is wrong data, Wrong data means the data value is wrong. For example, "Duration" column should have values 30 to 60, but someone written 450, which is wrong. Sometimes you can spot wrong data by looking at the data set, because you have an expectation of what it should be. We can fix these value iterating through the columns. In case of data in wrong format, we can either remove the columns, or convert all cells in the columns into the same format. Since, cells with data of wrong format can make it difficult, or even impossible, to analyze data. For duplicate data we simply remove them from the dataset so they cannot manipulate the model. Exploratory Data Analysis is a very important step before training the model. We will use some visualizations to understand the relationship of the target variable with other feature variables. Next, we create a correlation matrix that measures the linear relationships between the variables. Features that correlate together may make interpretability of their effectiveness difficult). We can also use parallel plot; a parallel plot allows to compare the feature of several individual observations (series) on a set of numeric variables. Interestingly, Pandas is probably the best way to plot a parallel coordinate plot with python. When we develop model, we create feature matrix, with relevant columns to find the optimize output. We also need to set the target which will be our goal to predict. Then we split the dataset for training and testing the model. Depending on the size of model it's a standard practice to split the training and testing dataset to 80-20 or 70-30. Finally, we fit the machine learning model with testing dataset. To train the model means we need to give both features and target so that machine can develop a mathematical formula or an algorithm which find relation between the features and target. For predicting, we need to provide the testing dataset that only contain the feature which will predict the target. And to measure the accuracy of the model we compare the predicted results with the testing target. The higher the accuracy the better it is.

2. Dataset Overview

Source: https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

Scan QR code to visit source

Description:

This dataset contains an airline passenger satisfaction survey. What factors are highly correlated to a satisfied (or dissatisfied) passenger. It has 25 different columns and 103904 instances.

Gender: Gender of the passengers (Female, Male)
Customer Type: The customer type (Loyal customer, disloyal customer)
Age: The actual age of the passengers
Type of Travel: Purpose of the flight of the passengers (Personal Travel, Business Travel)
Class: Travel class in the plane of the passengers (Business, Eco, Eco Plus)
Flight distance: The flight distance of this journey
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: Minutes delayed when departure
Arrival Delay in Minutes: Minutes delayed when Arrival
Satisfaction: Airline satisfaction level(Satisfaction, neutral or dissatisfaction)

3. Data Preprocessing and Exploratory data analysis

Setting Up Library

```
import numpy as np
import pandas as pd

# Visualization Libraries
import matplotlib.pyplot as plt
import seaborn as sns

# for checking the model accuracy
from sklearn import metrics

#To plot the graph embedded in the notebook
%matplotlib inline
```

[1] ✓ 0.8s Python

Loading Dataset

```
passenger_df = pd.read_csv('Airline.csv')
passenger_df
```

[5] ✓ Python

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	bo serv
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	
103899	103899	94171	Female	disloyal Customer	23	Business travel	Eco	192	2	1	...	2	
103900	103900	73097	Male	Loyal Customer	49	Business travel	Business	2347	4	4	...	5	
103901	103901	68825	Male	disloyal Customer	30	Business travel	Business	1995	1	1	...	4	
103902	103902	54173	Female	disloyal Customer	22	Business travel	Eco	1000	1	1	...	1	
103903	103903	62567	Male	Loyal Customer	27	Business travel	Business	1723	1	3	...	1	

103904 rows × 25 columns

Checking Information about Data

```
passenger_df.info()
```

[803] Python

```
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   Unnamed: 0                        103904 non-null  int64
 1   id                                103904 non-null  int64
 2   Gender                            103904 non-null  object
 3   Customer Type                     103904 non-null  object
 4   Age                               103904 non-null  int64
 5   Type of Travel                    103904 non-null  object
 6   Class                             103904 non-null  object
 7   Flight Distance                   103904 non-null  int64
 8   Inflight wifi service             103904 non-null  int64
 9   Departure/Arrival time convenient 103904 non-null  int64
 10  Ease of Online booking            103904 non-null  int64
 11  Gate location                     103904 non-null  int64
 12  Food and drink                    103904 non-null  int64
 13  Online boarding                   103904 non-null  int64
 14  Seat comfort                      103904 non-null  int64
 15  Inflight entertainment            103904 non-null  int64
 16  On-board service                  103904 non-null  int64
 17  Leg room service                  103904 non-null  int64
 18  Baggage handling                  103904 non-null  int64
 19  Checkin service                   103904 non-null  int64
...
 23  Arrival Delay in Minutes          103594 non-null  float64
 24  satisfaction                      103904 non-null  object
dtypes: float64(1), int64(19), object(5)
memory usage: 19.8+ MB
```

```
passenger_df.describe()
```

[6] ✓ 0.1s Python

	Unnamed: 0	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000
mean	51951.500000	64924.210502	39.379706	1189.448375	2.729683	3.060296	2.756901	2.976883
std	29994.645522	37463.812252	15.114964	997.147281	1.327829	1.525075	1.398929	1.277621
min	0.000000	1.000000	7.000000	31.000000	0.000000	0.000000	0.000000	0.000000
25%	25975.750000	32533.750000	27.000000	414.000000	2.000000	2.000000	2.000000	2.000000
50%	51951.500000	64856.500000	40.000000	843.000000	3.000000	3.000000	3.000000	3.000000
75%	77927.250000	97368.250000	51.000000	1743.000000	4.000000	4.000000	4.000000	4.000000
max	103903.000000	129880.000000	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000

Data Preprocessing:

Droping Irrelevant Columns

```
#Droping Unnamed Column and id Column
passenger_df = passenger_df.drop(passenger_df.iloc[:,[0,1]], axis = 1)
passenger_df
```

[804] Python

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	O boa servi
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	...	5	
1	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	...	1	
2	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	...	5	
3	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	...	2	
4	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	...	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	
103899	Female	disloyal Customer	23	Business travel	Eco	192	2	1	2	3	...	2	
103900	Male	Loyal Customer	49	Business travel	Business	2347	4	4	4	4	...	5	
103901	Male	disloyal Customer	30	Business travel	Business	1995	1	1	1	3	...	4	
103902	Female	disloyal Customer	22	Business travel	Eco	1000	1	1	1	5	...	1	
103903	Male	Loyal Customer	27	Business travel	Business	1723	1	3	3	3	...	1	

103904 rows × 23 columns

Dropping Duplicate Rows

```
#Droping duplicate rows and printing updated Data Frame
passenger_df.drop_duplicates(inplace= True)
passenger_df
```

[11] ✓ 0.1s Python

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inf se
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	...	5	4	3	4	4	
1	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	...	1	1	5	3	1	
2	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	...	5	4	3	4	4	
3	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	...	2	2	5	3	1	
4	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	...	3	3	4	4	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
103899	Female	disloyal Customer	23	Business travel	Eco	192	2	1	2	3	...	2	3	1	4	2	
103900	Male	Loyal Customer	49	Business travel	Business	2347	4	4	4	4	...	5	5	5	5	5	
103901	Male	disloyal Customer	30	Business travel	Business	1995	1	1	1	3	...	4	3	2	4	5	
103902	Female	disloyal Customer	22	Business travel	Eco	1000	1	1	1	5	...	1	4	5	1	5	
103903	Male	Loyal Customer	27	Business travel	Business	1723	1	3	3	3	...	1	1	1	4	4	

103904 rows × 23 columns

Finding Null Values

```
#Finding the sum of null value in each column
passenger_df.isnull().sum()
```

[808] Python

```
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         0
Class                                  0
Flight Distance                        0
Inflight wifi service                  0
Departure/Arrival time convenient      0
Ease of Online booking                 0
Gate location                          0
Food and drink                         0
Online boarding                        0
Seat comfort                           0
Inflight entertainment                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Inflight service                       0
Cleanliness                            0
Departure Delay in Minutes             0
Arrival Delay in Minutes             310
satisfaction                           0
dtype: int64
```

```
#Finding the percentage of null value in each column
passenger_df.isnull().sum()/len(passenger_df)
```

[809] Python

```
Gender                                 0.000000
Customer Type                          0.000000
Age                                    0.000000
Type of Travel                         0.000000
Class                                  0.000000
Flight Distance                        0.000000
Inflight wifi service                  0.000000
Departure/Arrival time convenient      0.000000
Ease of Online booking                 0.000000
Gate location                          0.000000
Food and drink                         0.000000
Online boarding                        0.000000
Seat comfort                           0.000000
Inflight entertainment                 0.000000
On-board service                       0.000000
Leg room service                       0.000000
Baggage handling                       0.000000
Checkin service                        0.000000
Inflight service                       0.000000
Cleanliness                            0.000000
Departure Delay in Minutes             0.000000
Arrival Delay in Minutes               0.002984
satisfaction                           0.000000
dtype: float64
```

Droping Rows with Null Value

```
#Droping Null Value and loading Updated Dataset
passenger_df = passenger_df.dropna()
passenger_df
```

[18] ✓ 0.1s Python

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inf se
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3	1	...	5	4	3	4	4	
1	Male	disloyal Customer	25	Business travel	Business	235	3	2	3	3	...	1	1	5	3	1	
2	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2	2	...	5	4	3	4	4	
3	Female	Loyal Customer	25	Business travel	Business	562	2	5	5	5	...	2	2	5	3	1	
4	Male	Loyal Customer	61	Business travel	Business	214	3	3	3	3	...	3	3	4	4	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
103899	Female	disloyal Customer	23	Business travel	Eco	192	2	1	2	3	...	2	3	1	4	2	
103900	Male	Loyal Customer	49	Business travel	Business	2347	4	4	4	4	...	5	5	5	5	5	
103901	Male	disloyal Customer	30	Business travel	Business	1995	1	1	1	3	...	4	3	2	4	5	
103902	Female	disloyal Customer	22	Business travel	Eco	1000	1	1	1	5	...	1	4	5	1	5	
103903	Male	Loyal Customer	27	Business travel	Business	1723	1	3	3	3	...	1	1	1	4	4	

103594 rows × 23 columns

Current Shape and Columns of the Data Frame

```
#Printing Shape and Column of Data Frame
print('Shape: ',passenger_df.shape)
print('Columns: ',passenger_df.columns)
```

[20] ✓ 0.1s Python

```
Shape:  (103594, 23)
Columns:  Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

Replacing String Value with Integer

```
#Replacing String Value with integer and printing updated Dataset
passenger_df['Gender'] = passenger_df['Gender'].replace(['Female','Male'], [0,1])
passenger_df['Customer Type'] = passenger_df['Customer Type'].replace(['Loyal Customer','disloyal Customer'], [0,1])
passenger_df['Type of Travel'] = passenger_df['Type of Travel'].replace(['Business travel','Personal Travel'], [0,1])
passenger_df['Class'] = passenger_df['Class'].replace(['Business','Eco','Eco Plus'], [0,1,2])
passenger_df['satisfaction'] = passenger_df['satisfaction'].replace(['neutral or dissatisfied','satisfied'], [0,1])
passenger_df
```

[38] ✓ 0.1s Python

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service
0	1	0	13	1	2	460	3	4	3	1	...	5	4	3	4	4	5
1	1	1	25	0	0	235	3	2	3	3	...	1	1	5	3	1	4
2	0	0	26	0	0	1142	2	2	2	2	...	5	4	3	4	4	4
3	0	0	25	0	0	562	2	5	5	5	...	2	2	5	3	1	4
4	1	0	61	0	0	214	3	3	3	3	...	3	3	4	4	3	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
103899	0	1	23	0	1	192	2	1	2	3	...	2	3	1	4	2	3
103900	1	0	49	0	0	2347	4	4	4	4	...	5	5	5	5	5	5
103901	1	1	30	0	0	1995	1	1	1	3	...	4	3	2	4	5	5
103902	0	1	22	0	1	1000	1	1	1	5	...	1	4	5	1	5	4
103903	1	0	27	0	0	1723	1	3	3	3	...	1	1	1	4	4	3

103594 rows × 23 columns

Exploratory Data Analysis:

Finding Size of Each Group

```
#Printing each some important group size information
print(passenger_df.groupby('Gender').size(),"\n")
print(passenger_df.groupby('Customer Type').size(),"\n")
print(passenger_df.groupby('Type of Travel').size(),"\n")
print(passenger_df.groupby('Class').size(),"\n")
print(passenger_df.groupby('satisfaction').size(),"\n")
```

[37] ✓ 0.5s Python

```
Gender
0    52576
1    51018
dtype: int64

Customer Type
0    84662
1    18932
dtype: int64

Type of Travel
0    71465
1    32129
dtype: int64

Class
0    49533
1    46593
2     7468
dtype: int64

satisfaction
0    58697
1    44897
dtype: int64
```

Count Plot

```
#Count Plot
plt.figure(figsize=(17,8))
sns.set_style ('whitegrid')
plt.subplot(2, 3, 1)
sns.countplot(data=passenger_df,x='Gender',palette='twilight')
plt.subplot(2, 3, 2)
sns.countplot(data=passenger_df,x='Customer Type',palette='twilight')
plt.subplot(2, 3, 3)
sns.countplot(data=passenger_df,x='Type of Travel',palette='twilight')
plt.subplot(2, 3, 4)
sns.countplot(data=passenger_df,x='Class',palette='twilight')
plt.subplot(2, 3, 5)
sns.countplot(data=passenger_df,x='satisfaction',palette='twilight')
```

[151] ✓ 0.7s Python

```
<AxesSubplot: xlabel='satisfaction', ylabel='count'>
```

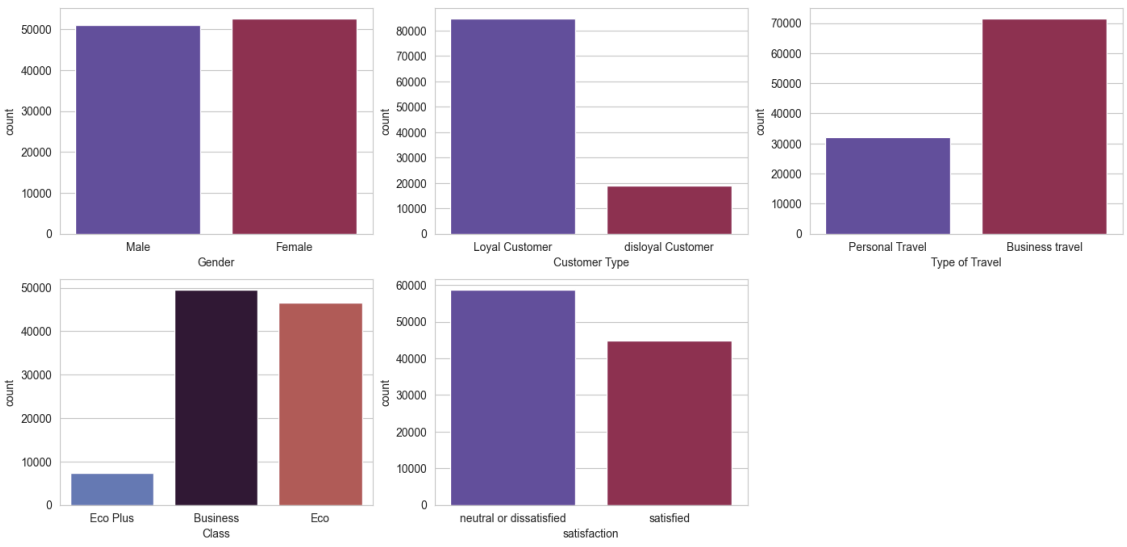

Figure: Count Plot

Distribution Plot

```
#Distribution Plot of Age and Flight Distance
sns.set_style ('whitegrid')
plt.figure(figsize=(17,8))

sns.displot(passenger_df['Age'], bins=30, color='black')
plt.title("Data distribution of Age");

sns.displot(passenger_df['Flight Distance'], bins=30, color='black')
plt.title("Data distribution of Flight Distance");
```

[180] ✓ 1.1s Python

```
<Figure size 1700x800 with 0 Axes>
```

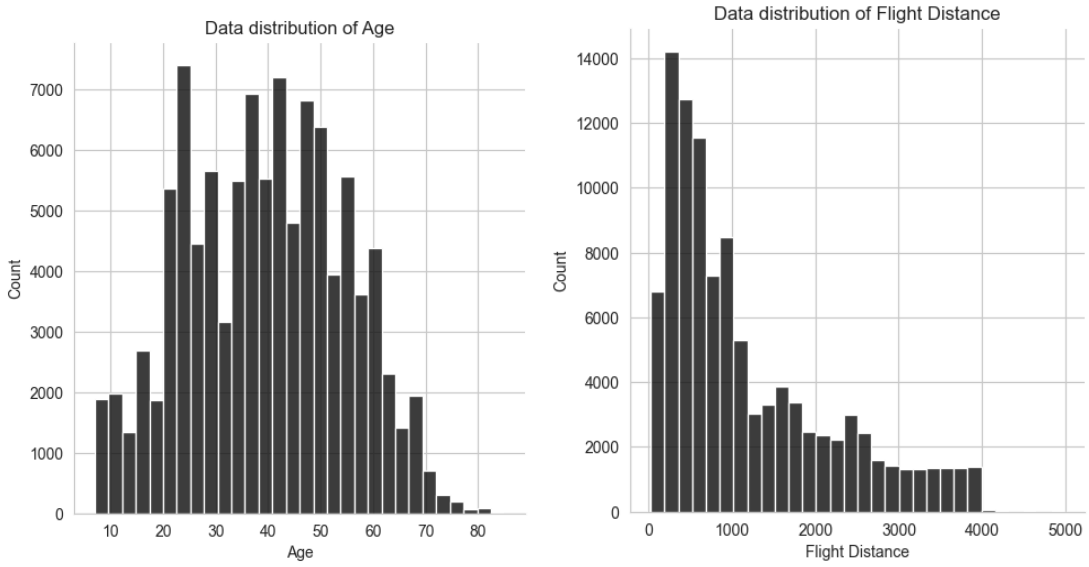

Figure: Distribution Plot

Paraller Co-ordinates

```
#Ploting Parallel cordinates
from pandas.plotting import parallel_coordinates
plt.figure(figsize = (45, 8))
parallel_coordinates(passenger_df, "satisfaction", color = ['blue', 'green']);
```

[215] ✓ 2m 0.9s Python

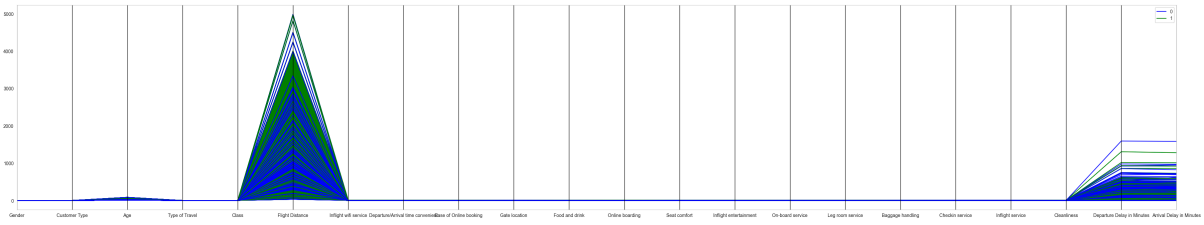

Figure: Parallel Co-ordinates

correlation Matrix

```
# corr() to calculate the correlation between variables
correlation_matrix = passenger_df.corr().round(2)
# changing the figure size
plt.figure(figsize = (15, 10))
# "annot = True" to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True);
```

[ ] Python

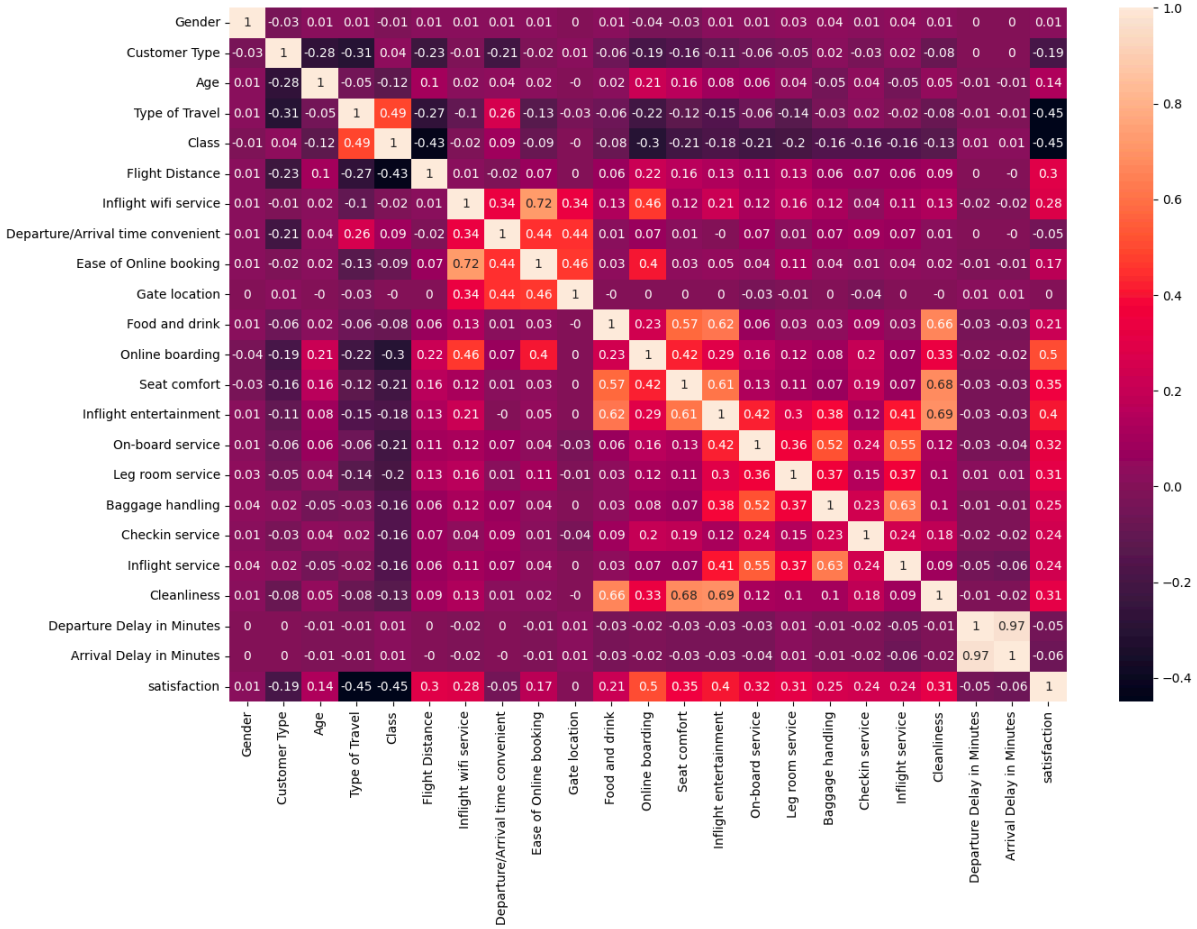

Figure: Correlation Matrix

```
# corr() to calculate the correlation between variables
correlation_matrix = passenger_df.corr().round(2)
# Steps to remove redundant values
# Return a array filled with zeros
mask = np.zeros_like(correlation_matrix)
# Return the indices for the upper-triangle of array
mask[np.triu_indices_from(mask)] = True
# changing the figure size
plt.figure(figsize = (12, 8))
# "annot = True" to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True, mask=mask);
```

[201] ✓ 1.2s Python

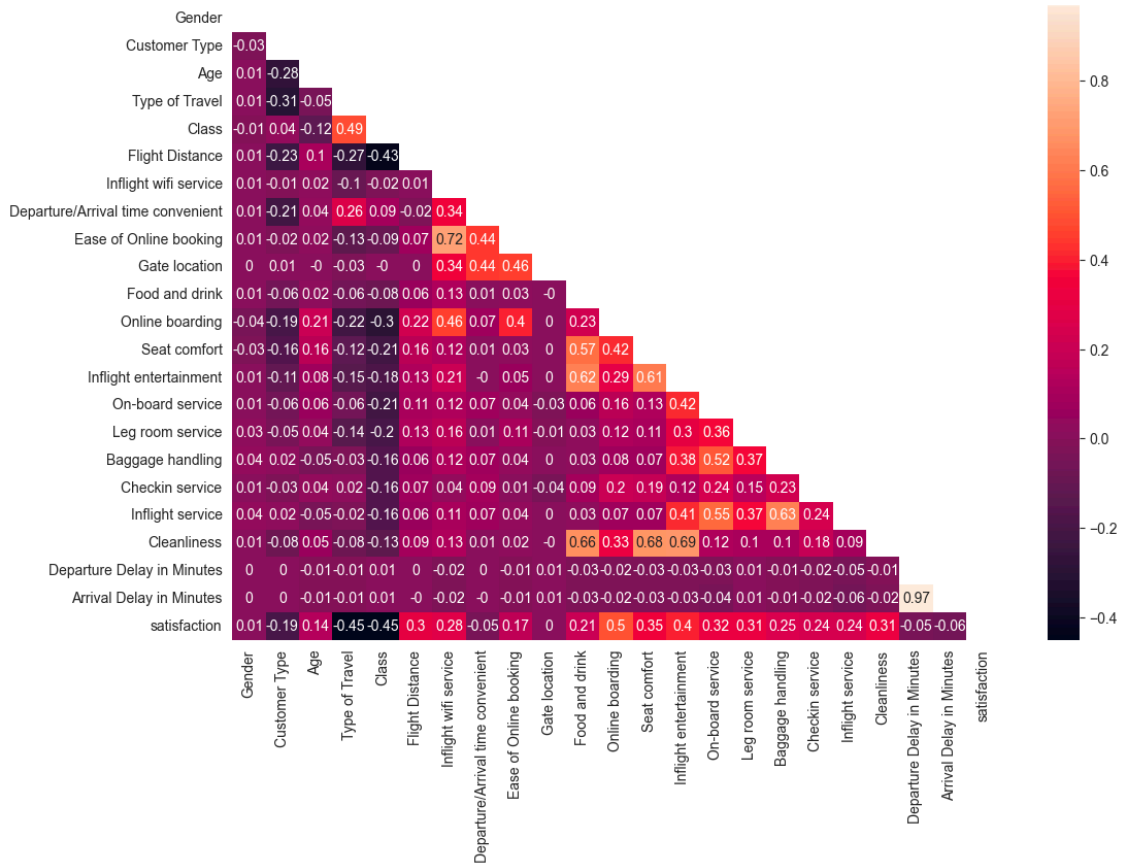

Figure: Correlation Matrix

4. Model Development

For model development we have remove features which has either high correlation or no correlation with the help of correlation matrix. We only considered relevant variable such as Type of Travel, Inflight wifi service, Online boarding, Seat comfort, Inflight entertainment, On-board service, Leg room service, Baggage handling, Checkin service, Inflight service, Cleanliness' for feature matrix. And our Target variable is satisfaction.

Create Features Matrix & Target Variable

```
#Feature matrix

X = passenger_df[['Type of Travel','Inflight wifi service','Online boarding', 'Seat comfort','Inflight entertainment', 'On-board service',
        'Leg room service','Baggage handling', 'Checkin service', 'Inflight service','Cleanliness']]
X
```

[237] ✓ 0.5s Python

	Type of Travel	Inflight wifi service	Online boarding	Seat comfort	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkin service	Inflight service	Cleanliness
0	1	3	3	5	5	4	3	4	4	5	5
1	0	3	3	1	1	1	5	3	1	4	1
2	0	2	5	5	5	4	3	4	4	4	5
3	0	2	2	2	2	2	5	3	1	4	2
4	0	3	5	5	3	3	4	4	3	3	3
...	...	...	...	...	...	...	...	...	...	...	...
103899	0	2	2	2	2	3	1	4	2	3	2
103900	0	4	4	5	5	5	5	5	5	5	4
103901	0	1	1	5	4	3	2	4	5	5	4
103902	0	1	1	1	1	4	5	1	5	4	1
103903	0	1	1	1	1	1	1	4	4	3	1

103594 rows × 11 columns

```
#Target Variable
y= passenger_df['satisfaction']
y
```

[238] ✓ 0.2s Python

```
0         0
1         0
2         1
3         0
4         1
         ..
103899    0
103900    1
103901    0
103902    0
103903    0
Name: satisfaction, Length: 103594, dtype: int64
```

To split the dataset for training and testing the model we used train test split module from scikit learn, we split the dataset into 70-30 ratio for training and testing. Among 103594 rows 72515 rows were randomly selected for training and 31079 were selected for testing the model's prediction accuracy.

Spliting the Dataset

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 10)
print("Feature train shape: ", X_train.shape)
print("Feature test shape: ", X_test.shape)
print("Target train shape: ", y_train.shape)
print("Target test shape: ", y_test.shape)
```

[239] ✓ 0.3s Python

```
Feature train shape:  (72515, 11)
Feature test shape:  (31079, 11)
Target train shape:  (72515,)
Target test shape:  (31079,)
```

Create Model: Support Vector Machine (SVM)

```
# importing the necessary package to use the classification algorithm
from sklearn import svm #for Support Vector Machine (SVM) Algorithm

model_svm = svm.SVC() #select the algorithm
model_svm.fit(X_train, y_train) #train the model with the training dataset

y_prediction_svm = model_svm.predict(X_test) # pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_svm = metrics.accuracy_score(y_prediction_svm, y_test).round(4)
print("-------------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("-------------------------------------")

# save the accuracy score
score = set()
score.add(('SVM', score_svm))
```

[240] ✓ 36.7s Python

```
-------------------------------------
The accuracy of the SVM is: 0.9354
-------------------------------------
```

For Support Vector Machine (SVM) we imported the model from scikit learn and we fitted the model with both features and target for training and once the model is ready it's tested with only the testing features and the predicted outcome is then compared with the actual outcome to find the accuracy. For Support Vector Machine (SVM) the accuracy is almost 93.5% which is excellent.

```
#Confusion Matrix for Support Vector Machine
confusion_matrix = metrics.confusion_matrix(y_prediction_svm, y_test)
confusion_matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
```

[289] ✓ 0.2s Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x3184efa30>
```

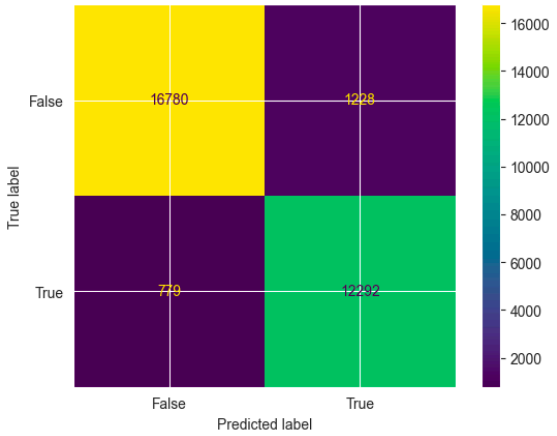

Figure: Confusion Matrix for Support Vector Machine (SVM)

Create Model: Decision Tree (DT)

```
# importing the necessary package to use the classification algorithm
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
model_dt = DecisionTreeClassifier(random_state=4)
model_dt.fit(X_train, y_train) #train the model with the training dataset
y_prediction_dt = model_dt.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_dt = metrics.accuracy_score(y_prediction_dt, y_test).round(4)
print("-----------------------------------")
print('The accuracy of the DT is: {}'.format(score_dt))
print("-----------------------------------")
# save the accuracy score
score.add(('DT', score_dt))
```

[241] ✓ 0.1s Python

```
-----------------------------------
The accuracy of the DT is: 0.9316
-----------------------------------
```

For Decision Tree (DT) we imported the model from scikit learn and we fitted the model with both features and target for training and once the model is ready it's tested with only the testing features and the predicted outcome is then compared with the actual outcome to find the accuracy. For Decision Tree (DT) the accuracy is almost 93 % which is also excellent.

```
#Confusion Matrix for Decision Tree
confusion_matrix = metrics.confusion_matrix(y_prediction_dt, y_test)
confusion_matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
```

[288] ✓ 0.3s Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x293305840>
```

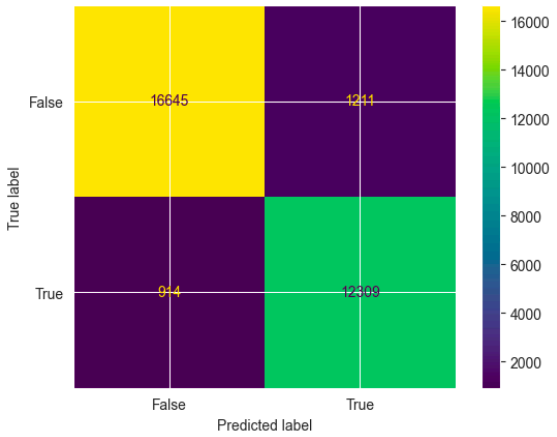

Figure: Confusion Matrix for Decision Tree (DT)

Create Model: K Nearest Neighbours (KNN)

```
# importing the necessary package to use the classification algorithm
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_knn = KNeighborsClassifier(n_neighbors=3) # 3 neighbours for putting the new data into a class
model_knn.fit(X_train, y_train) #train the model with the training dataset
y_prediction_knn = model_knn.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_knn = metrics.accuracy_score(y_prediction_knn, y_test).round(4)
print("-----------------------------------")
print('The accuracy of the KNN is: {}'.format(score_knn))
print("-----------------------------------")
# save the accuracy score
score.add(('KNN', score_knn))
```

[242] ✓ 2.4s Python

```
-----------------------------------
The accuracy of the KNN is: 0.9236
-----------------------------------
```

For K Nearest Neighbors (KNN) we imported the model from scikit learn and we fitted the model with both features and target for training and once the model is ready it's tested with only the testing features and the predicted outcome is then compared with the actual outcome to find the accuracy. For K Nearest Neighbors (KNN) the accuracy is almost 92 % which can be considered excellent.

```
#Confusion Matrix for K Nearest Neighbour
confusion_matrix = metrics.confusion_matrix(y_prediction_knn, y_test)
confusion_matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
```

[287] ✓ 0.2s Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x318344910>
```

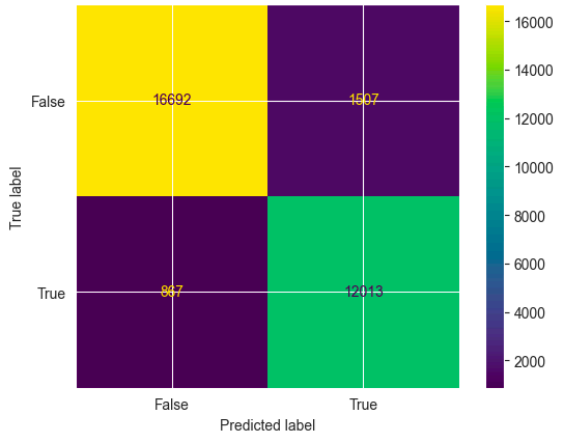

Figure: Confusion Matrix for K Nearest Neighbors (KNN)

Create Model: Logistic Regression

```
# importing the necessary package to use the classification algorithm
from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_lr = LogisticRegression()
model_lr.fit(X_train, y_train) #train the model with the training dataset
y_prediction_lr = model_lr.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_lr = metrics.accuracy_score(y_prediction_lr, y_test).round(4)
print("----------------------------------")
print('The accuracy of the LR is: {}'.format(score_lr))
print("----------------------------------")
# save the accuracy score
score.add(('LR', score_lr))
```

[243] ✓ 0.2s Python

```
----------------------------------
The accuracy of the LR is: 0.8538
----------------------------------
```

For Logistic Regression (LR) we imported the model from scikit learn and we fitted the model with both features and target for training and once the model is ready it's tested with only the testing features and the predicted outcome is then compared with the actual outcome to find the accuracy. For Logistic Regression (LR) the accuracy is 85% which is very good.

```
#Confusion Matrix for Logistic Regression
confusion_matrix = metrics.confusion_matrix(y_prediction_lr, y_test)
confusion_matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
```

[286] ✓ 0.2s Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x293289540>
```

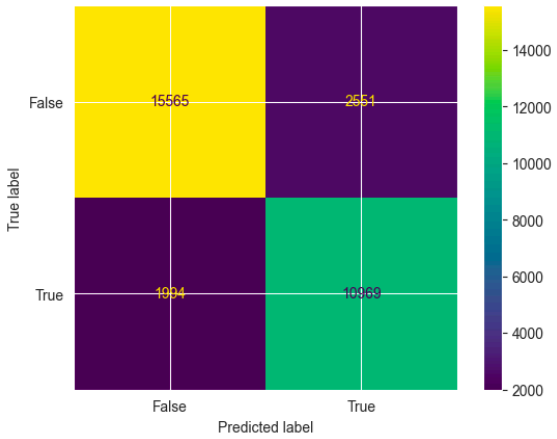

Figure: Confusion Matrix for Logistic Regression (LR)

Create Model: Naive Bayes

```
# importing the necessary package to use the classification algorithm
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
model_nb.fit(X_train, y_train) #train the model with the training dataset
y_prediction_nb = model_nb.predict(X_test) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
score_nb = metrics.accuracy_score(y_prediction_nb, y_test).round(4)
print("-----------------------------------")
print('The accuracy of the NB is: {}'.format(score_nb))
print("-----------------------------------")
# save the accuracy score
score.add(('NB', score_nb))
```

[244] ✓ 0.8s Python

```
-----------------------------------
The accuracy of the NB is: 0.8509
-----------------------------------
```

For Naïve Bayes (NB) we imported the model from scikit learn and we fitted the model with both features and target for training and once the model is ready it's tested with only the testing features and the predicted target is then compared with the actual outcome to find the accuracy. Naïve Bayes (NB) the accuracy is 85% which can be considered as pretty good.

```
#Confusion Matrix for Naive Bayes Model
confusion_matrix = metrics.confusion_matrix(y_prediction_nb, y_test)
confusion_matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
```

[285] ✓ 0.3s Python

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2900181f0>
```

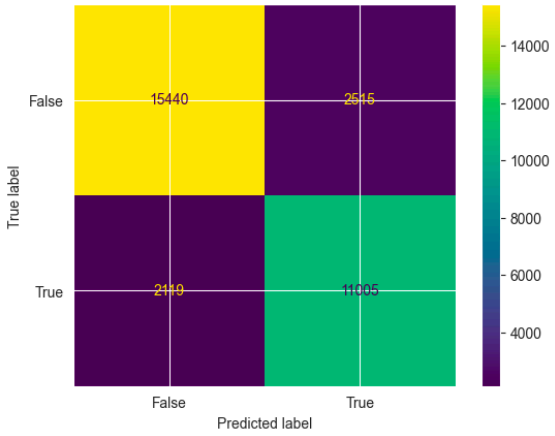

Figure: Confusion Matrix for Naïve Bayes (NB)

5. Discussion and Conclusion

Acurrecy of Five Different Machine Learning Model

```
#Printing the accurecy of 5 different models from score set using for loop
print("The accuracy scores of different Models:")
print("----------------------------------------")

for s in score:
  print(s)
```

[245] ✓ 0.4s Python

```
The accuracy scores of different Models:
----------------------------------------
('KNN', 0.9236)
('DT', 0.9316)
('NB', 0.8509)
('LR', 0.8538)
('SVM', 0.9354)
```

```
#Inserting the score into a dataset and arrenging it in assending order
accuracy_df = pd.DataFrame(score)
accuracy_df.rename(columns = {0:'Model',1:'Accuracy'}, inplace = True)
accuracy_df = accuracy_df.sort_values('Accuracy')
accuracy_df
```

[281] ✓ 0.4s Python

	Model	Accuracy
2	NB	0.8509
3	LR	0.8538
0	KNN	0.9236
1	DT	0.9316
4	SVM	0.9354

```
#Ploting model vs accuracy barplot
sns.set_style ('darkgrid')
sns.barplot(data= accuracy_df,x='Model',y='Accuracy',palette='twilight')
plt.title('Model Vs Accuracy')
```

[282] ✓ 0.2s Python

```
Text(0.5, 1.0, 'Model Vs Accuracy')
```

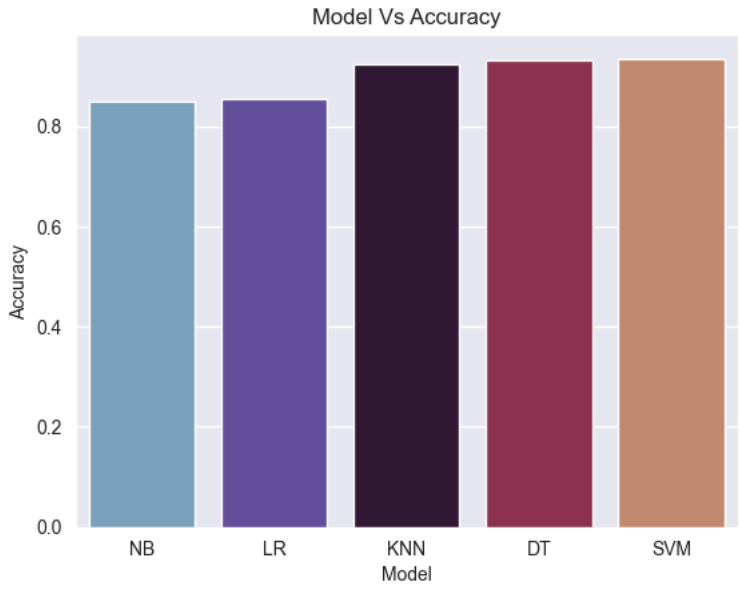

Figure: Bar Plot for Model vs Accuracy

From the given numbers and the bar plot for model vs accuracy we can say Naïve Bayes is the least accurate among the other five with the accuracy is 85% which can be considered as pretty decent. Logistic Regression's accuracy is just fraction better than Naïve Bayes which is very good but not as accurate as next three. For K Nearest Neighbors the accuracy is almost 92 % which can be considered excellent. Decision Tree model's accuracy is almost 93 % which slightly better than K Nearest Neighbors. Support Vector Machine with 93.5% accuracy is the best among all. The prediction accuracy of K Nearest Neighbors, Decision Tree, Support Vector Machine are more than 90% which can be considered excellent for the type of dataset we worked on.