Airline passenger Satisfaction Detection
CSE 587

Ranganadh Neelakantham
ranganad@buffalo.edu
50456403

Sarthak jain
sjain34@buffalo.edu
50485197

Problem Statement:

In this Project we will analyze the US Airline Passenger Satisfaction Dataset [1] and use it to train the model to determine the most significant factors affecting passenger satisfaction and provide recommendations to airlines on how to enhance their overall customer experience and also provide the dashboard to the customer where we can predict if customer is satisfied or not by entering rating values of each service they are providing thereby he can forecast which services needs improvement

Discuss the background of the problem leading to your objectives. Why is it a significant problem?

The airline industry's prosperity relies heavily on maintaining high levels of passenger satisfaction. Despite the airlines' attempts to improve their services and facilities to attract customers, passenger satisfaction remains a persistent issue. Flight cancellations and delays are among the primary factors contributing to this problem. In the United States, 23% of all domestic flights were delayed, while 1.9% were canceled in 2019, resulting in millions of passengers experiencing travel disruptions that lead to frustration, stress, and decreased satisfaction with the airline [2]. Another major concern affecting passenger satisfaction is the mishandling or loss of luggage. Passengers expect their luggage to be treated with care and arrive on time at their destination. However, when luggage is misplaced, delayed, or damaged during transit, it creates significant inconvenience and frustration. Poor customer service, uncomfortable seating arrangements, and inadequate in-flight amenities are additional issues that negatively impact passenger satisfaction. It is crucial for airlines to tackle the fundamental problems that cause passenger discontent to guarantee their long-term success and competitiveness.

Explain the potential of your project to contribute to your problem domain. Discuss why this contribution is crucial.

Machine learning models play a crucial role in identifying and improving airline passenger satisfaction by enabling data-driven decision-making. By identifying emerging trends, predicting passenger satisfaction, and offering personalized services, airlines can boost customer loyalty, revenue, and reputation in the competitive airline industry. Moreover, machine learning models aid in optimizing airline operations, reducing expenses, and maximizing profitability, ensuring their long-term success.

Data Sources

The dataset used in this project was acquired from Kaggle, where it was collected via a survey conducted by US airlines to measure passenger satisfaction levels. The dataset consists of 25 columns with 129897 data samples, including features such as age, gender, married or not, travel class, arrival and departure delays, as well as other factors that may impact customer satisfaction, such as on-board service, cleanliness, seat comfort, and baggage handling. Of particular importance in the dataset is the "satisfaction" column, which provides an overall Rating of the customer's satisfaction level. This column has two values: "neutral or dissatisfied" and "satisfied." It is considered the label feature because it represents the customer's overall experience based on their ratings for the other features.

Overall data points in dataset:

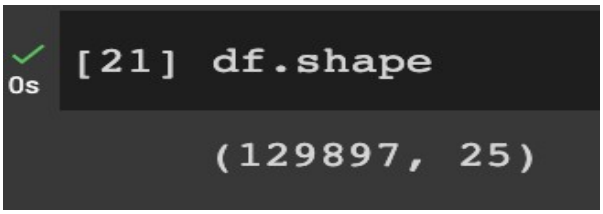


Fig 1.1. Sample size (rows , columns)

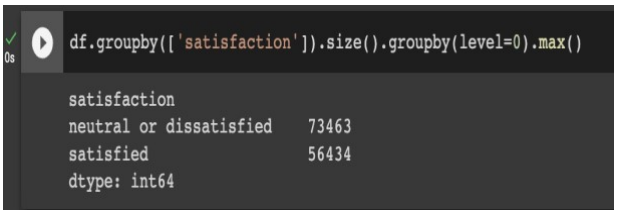


Fig 1.2. Distribution of output sample

Data column description

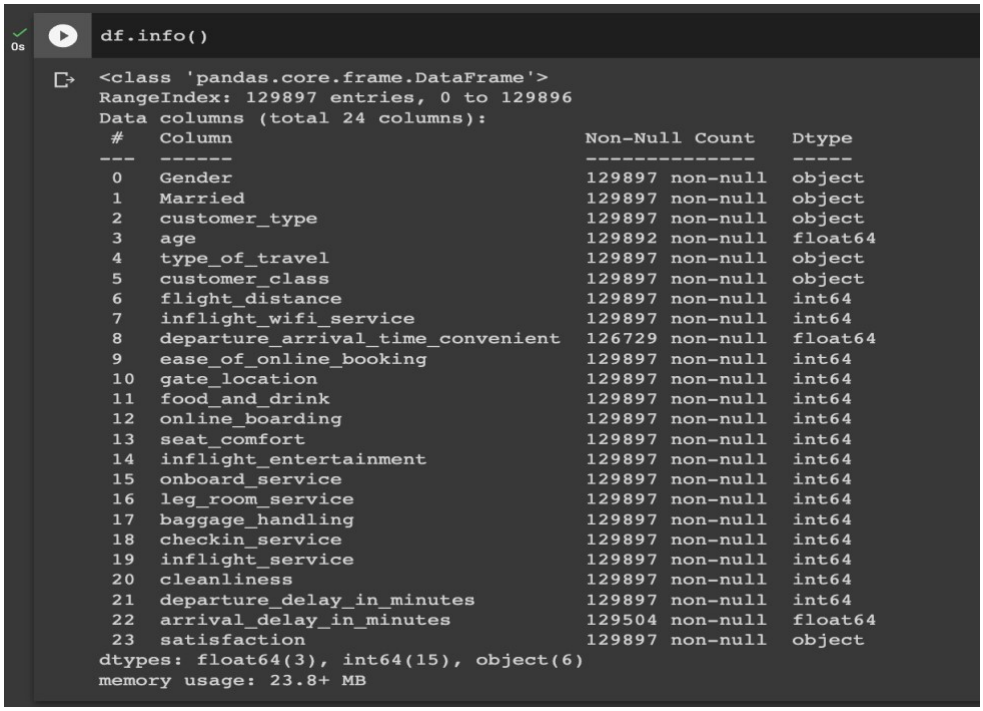


Fig 1.3. data column description

I. Data Cleaning/Processing

1) Remove Irrelevant data

When analyzing the data, we found that columns Sno and Married does not have any effect on outcome of satisfaction. So, we are removing those columns

Before removing,

```
[71] df = pd.read_csv("airline_passenger_satisfaction (5).csv")
```

```
df.head()
```

	Sno	Gender	Married	customer_type	age	type_of_travel	customer_class	flight_distance	inflight_wifi_service	departure_arrival_time_convenient	...	inflight_entertainment
0	1	Male	No	Loyal. Customer	13.0	Personal Travel	Eco Plus	460	3	4.0	...	5
1	2	Male	No	disloyal. Customer	25.0	Business travel	Business	235	3	2.0	...	1
2	3	Female	No	Loyal. Customer	26.0	Business travel	Business	1142	2	2.0	...	5
3	4	Female	No	Loyal. Customer	25.0	Business travel	Business	562	2	5.0	...	2
4	5	Male	Yes	Loyal. Customer	61.0	Business travel	Business	214	3	3.0	...	3

5 rows × 25 columns

Fig 1.4. Data before Irrelevant data After removing,

```
[67] # Remove Irrelevant data
     df = df.drop(['Sno', 'Married'], axis=1)
```

```
df.head()
```

	Gender	customer_type	age	type_of_travel	customer_class	flight_distance	inflight_wifi_service	departure_arrival_time_convenient	ease_of_online_booking	gate_location	..
0	Male	Loyal. Customer	13.0	Personal Travel	Eco Plus	460	3	4.0	3	1	
1	Male	disloyal. Customer	25.0	Business travel	Business	235	3	2.0	3	3	
2	Female	Loyal. Customer	26.0	Business travel	Business	1142	2	2.0	2	2	
3	Female	Loyal. Customer	25.0	Business travel	Business	562	2	5.0	5	5	
4	Male	Loyal. Customer	61.0	Business travel	Business	214	3	3.0	3	3	

5 rows × 23 columns

Fig 1.5. Data after removing Irrelevant data

2) Checking for missing data

Empty cells in the data can lead to inaccurate analysis results, thus it becomes necessary to remove any rows that contain empty cells.

```
#Check if any missing values
df.isnull().sum()
```

```
Sno                                    0
Gender                                 0
Married                                0
customer_type                          0
age                                    5
type_of_travel                         0
customer_class                         0
flight_distance                        0
inflight_wifi_service                  0
departure_arrival_time_convenient   3169
ease_of_online_booking                 0
gate_location                          0
food_and_drink                         0
online_boarding                        0
seat_comfort                           0
inflight_entertainment                 0
onboard_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             393
satisfaction                           0
dtype: int64
```

Fig 1.6. Data description showing missing data

Evaluating missing data

Despite the large size of the dataset, removing a few rows is unlikely to significantly affect the Outcome. So we will remove all the rows from the dataset which contains empty values

```
[57] #2. Evaluating missing data
     df=df.dropna(inplace = True)
```

```
#Check after dropping null rows
df.isnull().sum()
```

```
Sno                                    0
Gender                                 0
Married                                0
customer_type                          0
age                                    0
type_of_travel                         0
customer_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
onboard_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               0
satisfaction                           0
dtype: int64
```

Fig 1.7. data description after removing missing data

3) Checking for duplicate values

Duplicate rows add no significance to the data. So, we need to find out if we have duplicate rows in our data set . We found that there are 32 duplicate rows

```
[95] #check for duplicates
     duplicates = df[df.duplicated()]

     duplicates.shape

     (32, 23)
```

Fig 1.8. data description before

Removing duplicates

Since these duplicate rows add no significance, we need to get rid of these duplicate rows. So removing these duplicate rows from data set

```
[109] df = df.drop_duplicates(inplace=True)
```

Fig 1.9 Code to remove duplicate records

4) Remove extra Spaces

There may be some extra spaces by mistake that need to be cleaned up.

```
# remove extra spaces
df = df.replace(' ', '')
```

Fig 1.10 Code to replace extra spaces with empty string

5) Remove Punctuations

Punctuation marks don't add any meaning, so they need to be removed.

```
#remove punctuation from data
df['customer_type'] = df['customer_type'].str.replace(r'[^\w\s]+', '')
df['satisfaction'] = df['satisfaction'].str.replace(r'[^\w\s]+', '')
```

```
<ipython-input-123-50c18f3ef517>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  df['customer_type'] = df['customer_type'].str.replace(r'[^\w\s]+', '')
<ipython-input-123-50c18f3ef517>:3: FutureWarning: The default value of regex will change from True to False in a future version.
  df['satisfaction'] = df['satisfaction'].str.replace(r'[^\w\s]+', '')
```

+ Code + Text

Fig 1.11 Code to remove punctuations

6) Range Constraint validation

From survey description, we have found that all survey values will range between 1-5 where 1- extremely poor and 5 - Excellent. But we have found that some values with 0 rating. So, we Need to replace 0 with 1 rating.

```
#Range Constraints validation
df = df.replace(0, 1)
```

Fig 1.12 Code to replace 0 with 1

7) Standardizing capitalization

In order to handle the text, we must convert all cases to lowercase

```
[127] #Standardize Capitalization
      df = df.applymap(lambda s: s.lower() if type(s) == str else s)
```

Fig 1.13 Code to convert to lower case

8) Converting categorical data into numerical data

numerical data allows for mathematical operations and statistical analysis to be performed on

The data easily. So, we need to get rid of categorical data and convert them into numerical data

```
#Converting categorical data into Numerical data
df['satisfaction']=df['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1},inplace = True)
df['gender']=df['gender'].replace({'male': 0, 'female': 1},inplace = True)
df['customer_type']=df['customer_type'].replace({'loyal customer': 0, 'disloyal customer': 1},inplace = True)
```

Fig 1.14 Code to convert categorical data into Numerical

9) Standardizing column datatypes

Upon analyzing the data, it was discovered that the data types of two columns were incongruous. with the other columns that store the same data. To maintain consistency throughout the project, these two columns were modified to match the data types of the other columns.

```
#standardizing all column datatype which store same values
df['departure_delay_in_minutes'] = df['departure_delay_in_minutes'].astype(float)
df['flight_distance'] = df['flight_distance'].astype(float)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 23 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Gender                             0 non-null      object
 1   customer_type                      0 non-null      object
 2   age                                0 non-null      float64
 3   type_of_travel                     0 non-null      object
 4   customer_class                     0 non-null      object
 5   flight_distance                    0 non-null      float64
 6   inflight_wifi_service              0 non-null      int64
 7   departure_arrival_time_convenient  0 non-null      float64
 8   ease_of_online_booking             0 non-null      int64
 9   gate_location                      0 non-null      int64
 10  food_and_drink                     0 non-null      int64
 11  online_boarding                    0 non-null      int64
 12  seat_comfort                       0 non-null      int64
 13  inflight_entertainment             0 non-null      int64
 14  onboard_service                    0 non-null      int64
 15  leg_room_service                   0 non-null      int64
 16  baggage_handling                   0 non-null      int64
 17  checkin_service                    0 non-null      int64
 18  inflight_service                   0 non-null      int64
 19  cleanliness                        0 non-null      int64
 20  departure_delay_in_minutes         0 non-null      float64
 21  arrival_delay_in_minutes           0 non-null      float64
 22  satisfaction                       0 non-null      object
dtypes: float64(5), int64(13), object(5)
memory usage: 0.0+ bytes
```

Fig 1.15 Code to convert columns datatypes

10) Fixing structural problems

Upon analyzing the data, it was discovered that the customer class column was abbreviated with short forms. To enhance comprehension and readability, the column data has been expanded to its full form.

```
#Fixing Structural Problems
Class_Dict = {'eco':'economy' ,
              'eco plus' : 'economy plus' ,
              'business' : 'business'}
df['customer_class'] = df['customer_class'].map(Class_Dict)
df['customer_class'].unique()
```

```
array(['economy plus', 'business', 'economy'], dtype=object)
```

Fig 1.16 Code to expand short forms

11) Sorting

Sorting the data showing all satisfied customer and neutral or dissatisfied customer together.

```
df = df.sort_values(by=['satisfaction'])
df
```

...	gate_location	inflight_entertainment	inflight_service	inflight_wifi_service	leg_room_service	onboard_service	online_boarding	satisfaction	seat_comfort	type_of_travel
...	1	5	5	3	3	4	3	neutral or dissatisfied	5	personal travel
...	4	2	2	2	2	2	3	neutral or dissatisfied	3	business travel
...	1	2	5	2	5	2	2	neutral or dissatisfied	4	personal travel
...	3	1	3	1	4	1	1	neutral or dissatisfied	1	personal travel
...	3	1	1	3	3	1	4	neutral or dissatisfied	5	personal travel
...	...	...	...	...	...	...	...	...	...	...
...	4	5	5	5	5	5	5	satisfied	4	business travel
...	1	4	4	1	4	4	4	satisfied	5	business travel
...	3	5	2	1	5	3	1	satisfied	5	personal travel
...	3	5	1	5	5	5	5	satisfied	5	business travel
...	5	5	4	5	2	3	5	satisfied	5	business travel

Fig 1.17 Code to Sort the data by data

12) Reordering

Reordering the columns based on alphabetical order for better readability and understanding

```
df = df.reindex(sorted(df.columns), axis=1)
df.head()
```

	Gender	age	arrival_delay_in_minutes	baggage_handling	checkin_service	cleanliness	customer_class	customer_type	departure_arrival_time_convenient	departure_delay_in_min
0	male	13.0	18.0	4	4	5	economy plus	loyal customer	4.0	
1	male	25.0	6.0	3	1	1	business	disloyal customer	2.0	
2	female	26.0	1.0	4	4	5	business	loyal customer	2.0	
3	female	25.0	9.0	3	1	2	business	loyal customer	5.0	
4	male	61.0	1.0	4	3	3	business	loyal customer	3.0	

5 rows × 23 columns

Fig 1.18 Code to reorder the columns

II. Exploratory Data Analysis:

To get started with the EDA first we figured out the types of data present in the dataset. We found that we have 2 categories of data present in the dataset:

1) **Categorical** ['Gender', 'customer_type', 'type_of_travel', 'customer_class', 'satisfaction']
2) **Numerical**
   a) Continuos ['age', 'flight_distance', 'departure_delay_in_minutes', 'arrival_delay_in_minutes']
   b) Discrete ['inflight_wifi_service', 'departure_arrival_time_convenient', 'ease_of_online_booking', 'gate_location', 'food_and_drink', 'online_boarding', 'seat_comfort', 'inflight_entertainment', 'onboard_service', 'leg_room_service', 'baggage_handling', 'checkin_service', 'inflight_service', 'cleanliness']

After defining our variable and there data types now lets pick all the feature one by one and analyze them.

Categorical Variables:

1) Gender

We can see gender does not influence customer satisfaction. We have mixed set of opinions regarding the customer satisfaction.

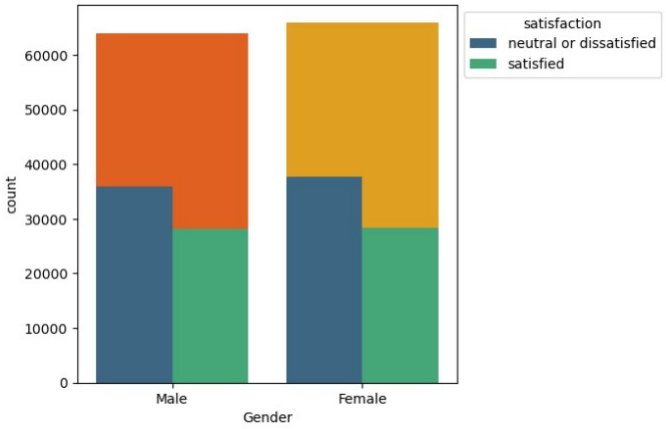


Fig 2.1 countplot for the variable gender with hue as satisfaction

2) Customer Type

Loyal customers tends to be more satisfied than the disloyal customer.

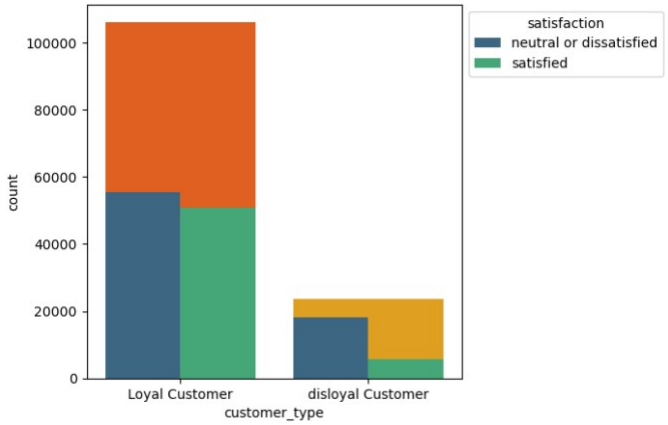


Fig 2.2 countplot for the customer_type with hue as satisfaction

3) Type Of travel

People travelling for personal reasons tend to have greater level of dissatisfaction as compared to people travelling for business visits. This may be due to people travelling for business purposes travel more on business class flights.

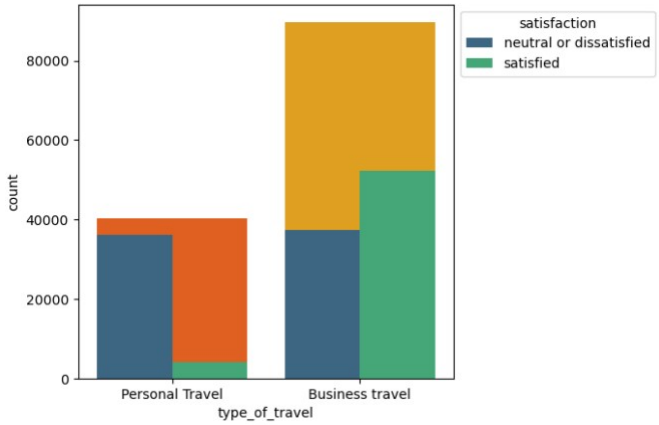


Fig 2.3 countplot for the variable type_of_travel with hue as satisfaction

Lets plot a countplot of customer class with hue as type of travel to corroborate the above assumption.

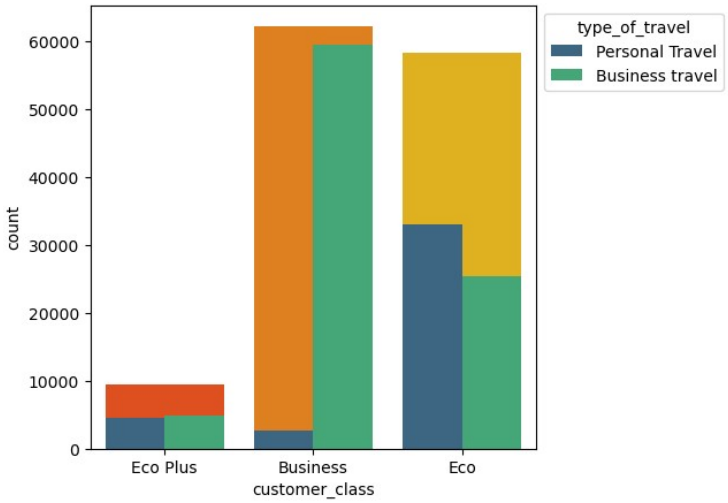


Fig 2.4 countplot for the variable customer_class with hue as type_of_travel

4) Customer Class

Business class travelers are more satisfied than other economy classes. This indicates that the economy classes need more attention as compare to business class.

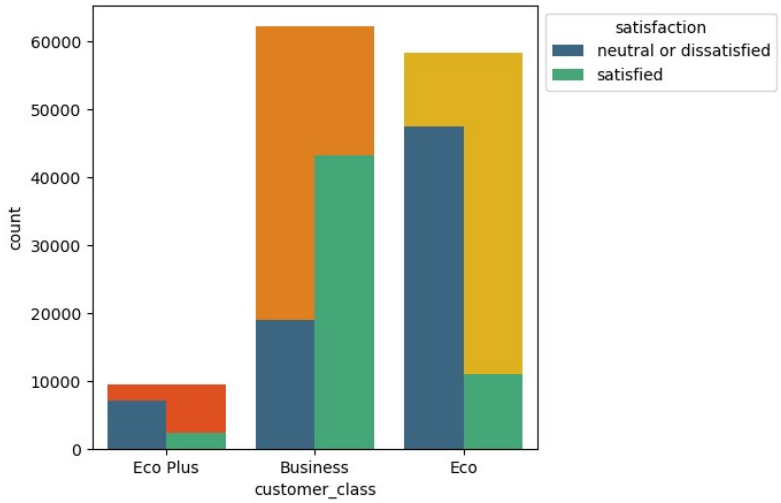


Fig 2.5 countplot for the variable customer_class with hue as satisfaction

After categorical data, let's visualize the Numerical variables now.

Numerical Variables:

1) Inflight Wifi service

Flyers who rated wifi service with highest score are satisfied with the overall flight experience.

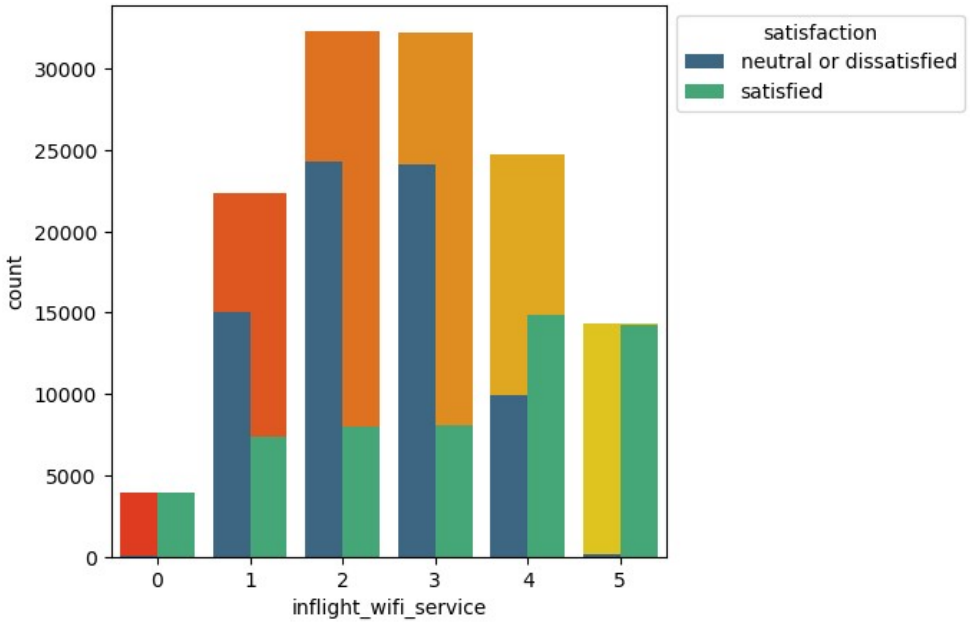


Fig 2.6 countplot for the variable inflight_wifi_service with hue as satisfaction

2) Departure arrival time convenient

The departure Arrival time convenience do not have any significant impact on the overall satisfaction of the customer.

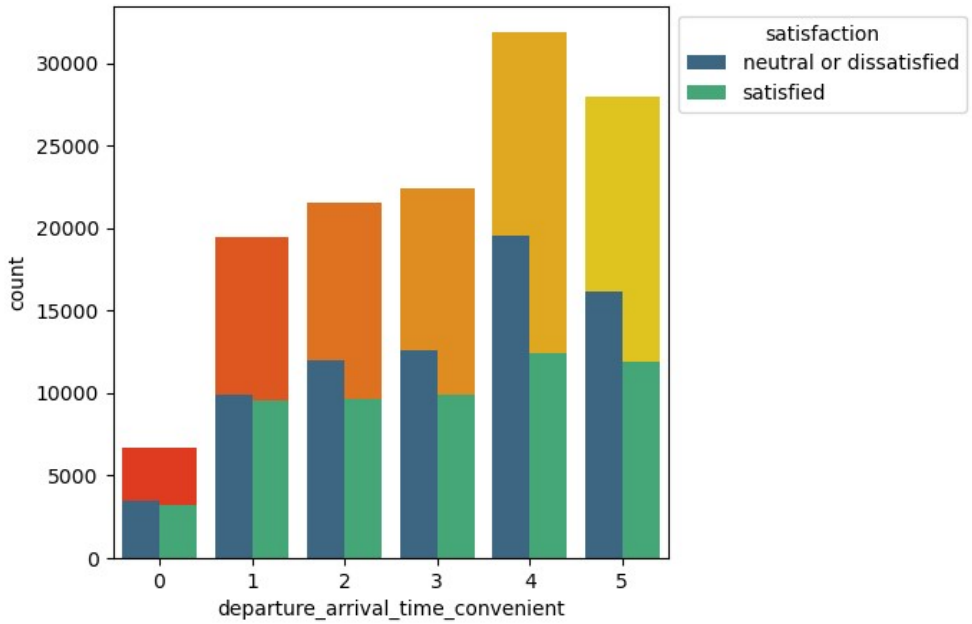


Fig 2.7 Countplot for the variable departure_arrival_time_convenient with hue as satisfaction

3) Ease of online booking

Those customers who are satisfied with the online booking service are inclined towards satisfactory experience.

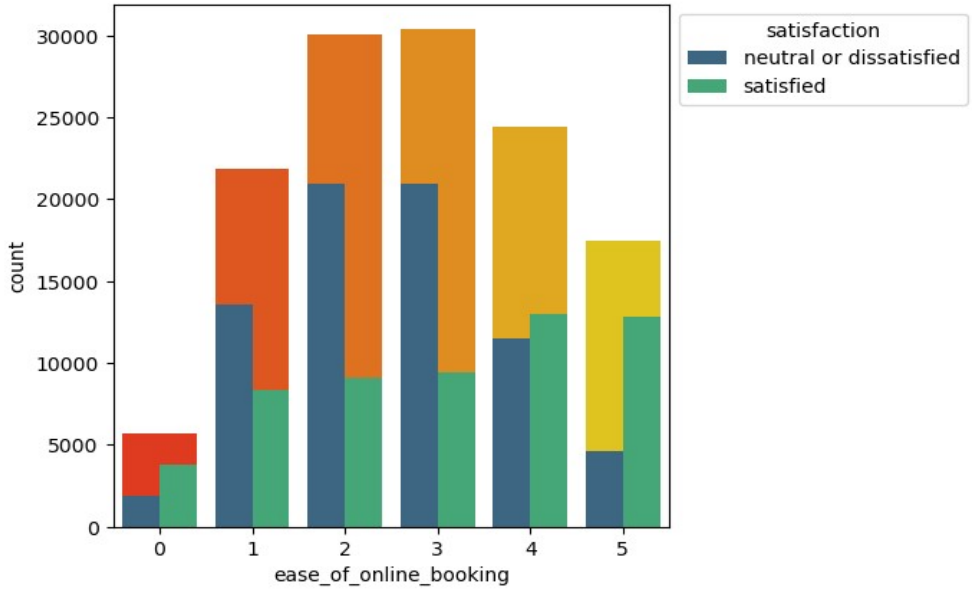


Fig 2.8 Countplot for the variable ease_of_online_booking with hue as satisfaction

4) Gate Location

It tends to have less impact on the overall satisfaction of the passengers.

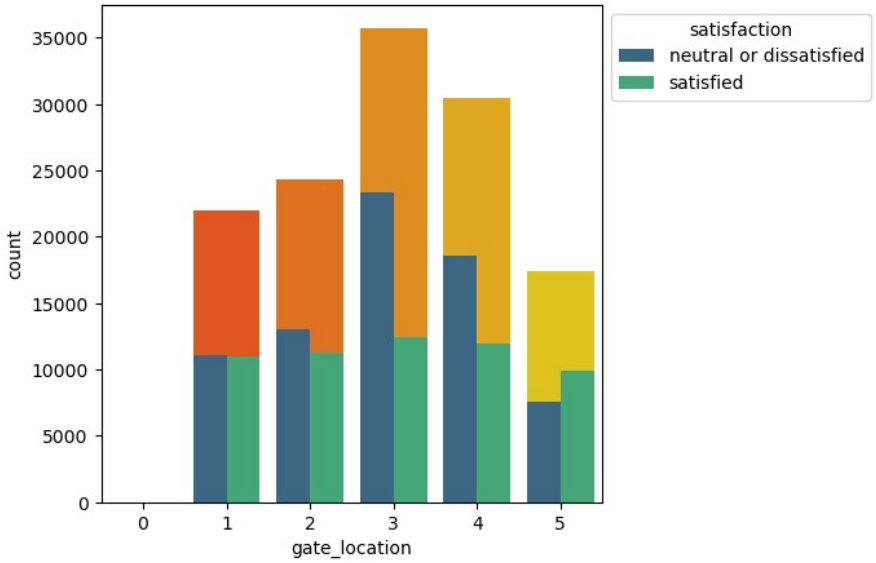


Fig 2.9 Countplot for the variable gate_location with hue as satisfaction

5) Food and drink

Food and drink also contribute less to the satisfaction of the customer however the food is rated the poorest then customer tends to have a dissatisfactory experience.

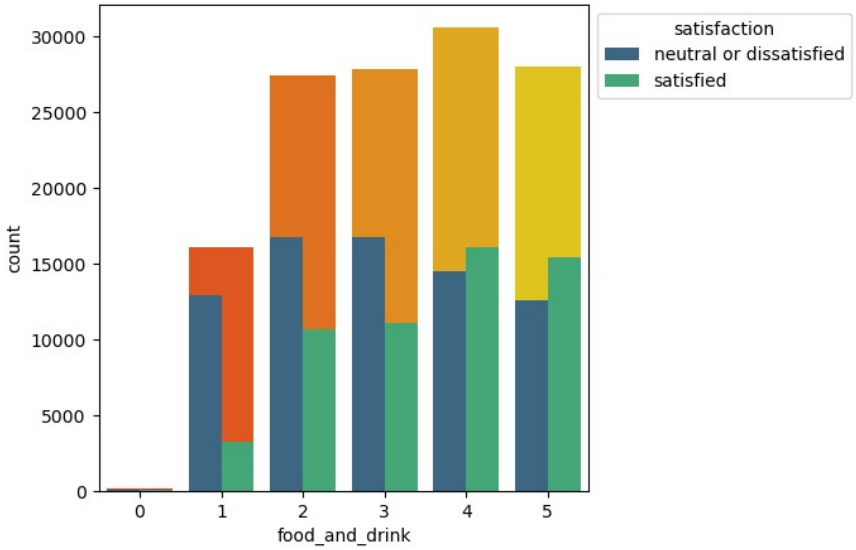


Fig 2.10 Countplot for the variable food_and_drink with hue as satisfaction

6) Online boarding

It tend to have a significant impact on the overall satisfaction of the customer. The customers who have rated the online boarding experience as good are tend to have a satisfactory experience overall. While the customer who rated the online service as 3 or below are dissatisfied with the airline.

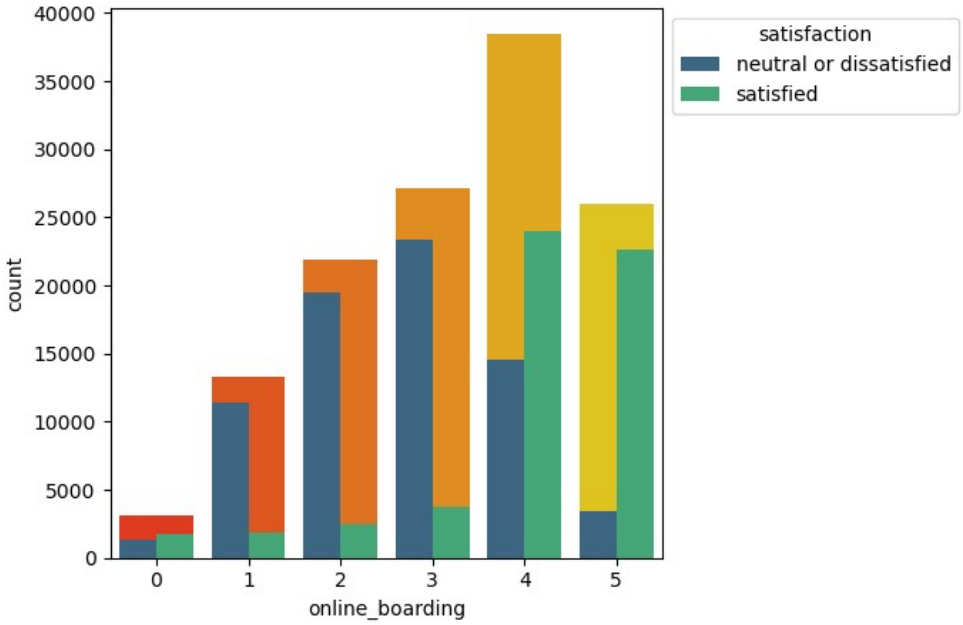


Fig 2.11 Countplot for the variable online_boarding with hue as satisfaction

7) Seat Comfort

Just like online boarding, Seat comfort also have a very huge impact on the overall customer satisfaction of the customer. The customers with good seating experience are satisfied overall.

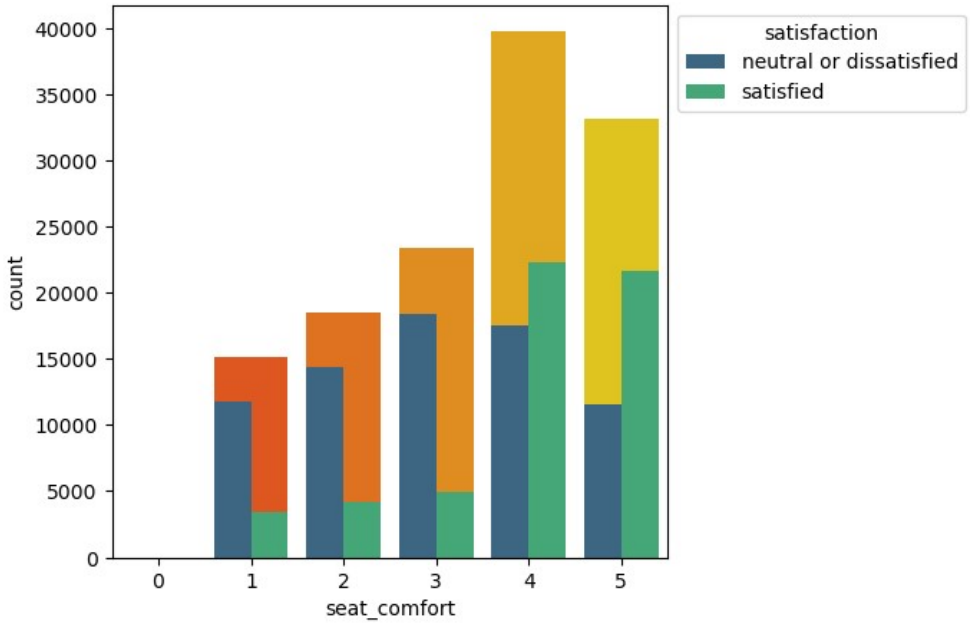


Fig 2.12 Countplot for the variable seat_comfort with hue as satisfaction

8) Inflight entertainment

The same pattern as seat comfort, wifi and some other variables are also observed in the 'inflight_entertainment'. Travelers care about the inflight entertainment service provided to them.

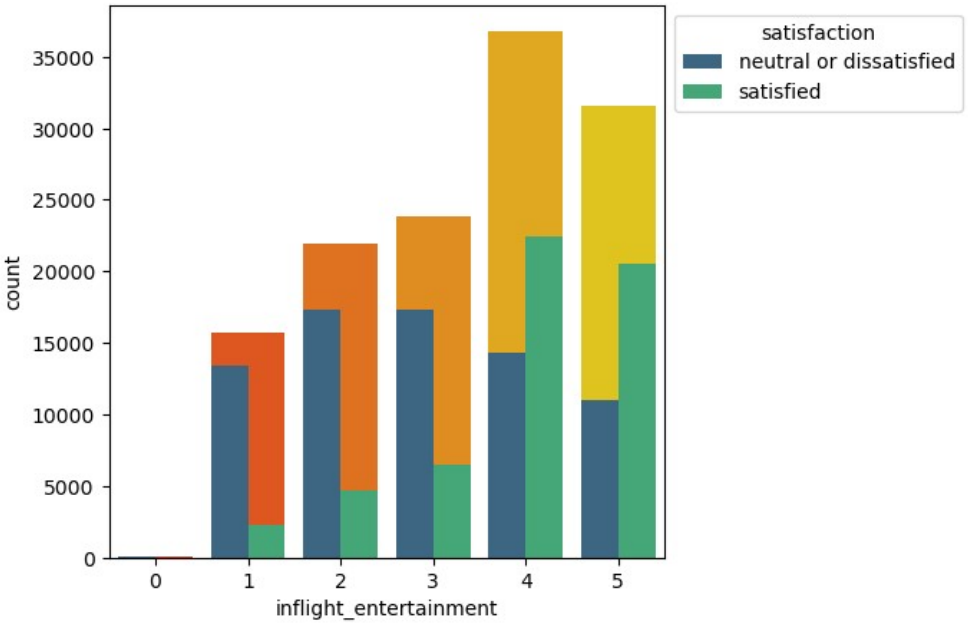


Fig 2.13 Countplot for the variable inflight_variable with hue as satisfaction

9) Onboard services:

The passengers who rated 4 and 5 stars for the onboarding service seems to have a satisfactory experience with the airline. And a rating of 1 to 3 shows a dissatisfactory customer behavior. This can be concluded as Passengers mostly want a 4 or 5 start onboarding services.

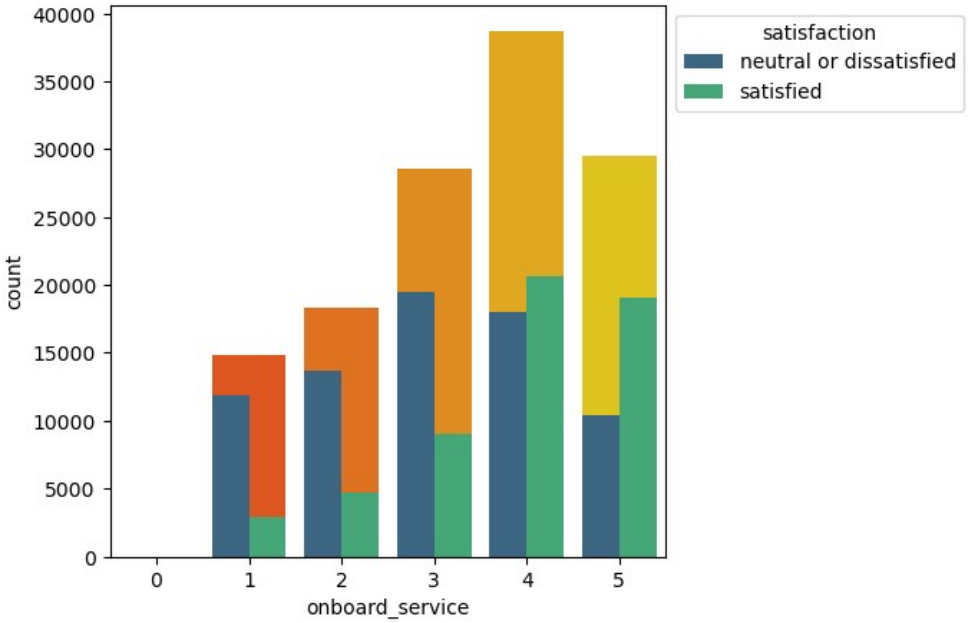
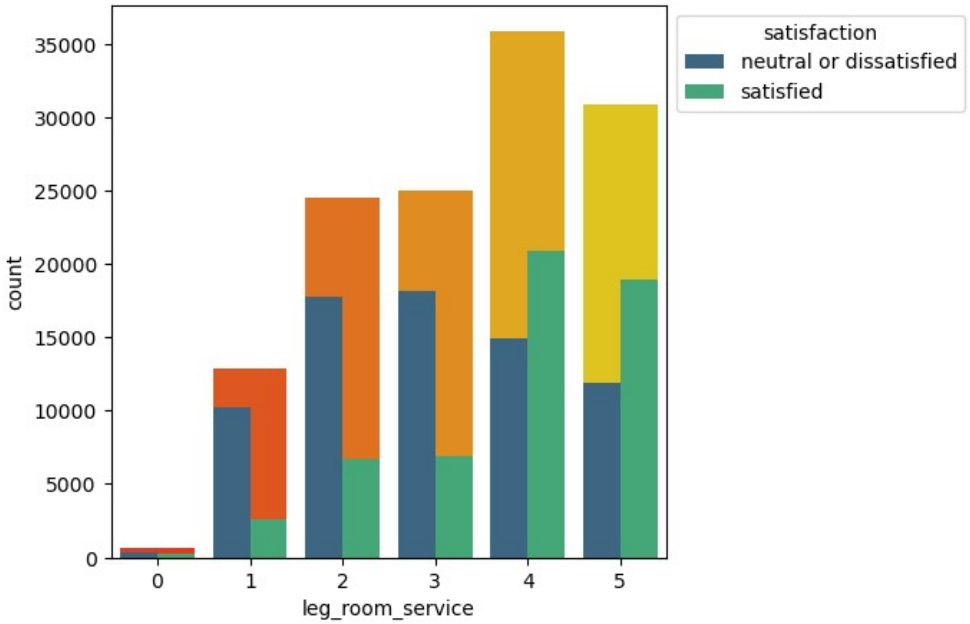


Fig 2.14 Countplot for the variable onboard_service with hue as satisfaction

10) Legroom Service

The same behavior as onboard services goes with the legroom service and baggage handling services also. Passenger do not expect anything less than 4 or 5 stars in these services.



10) Baggage Handling, Check in service and Inflight service

These services also follows the same pattern as the onboard service. Below are the countplots for these service with hue as satisfaction.

Fig 2.15 Countplot for the variable leg_room_service with hue as satisfaction

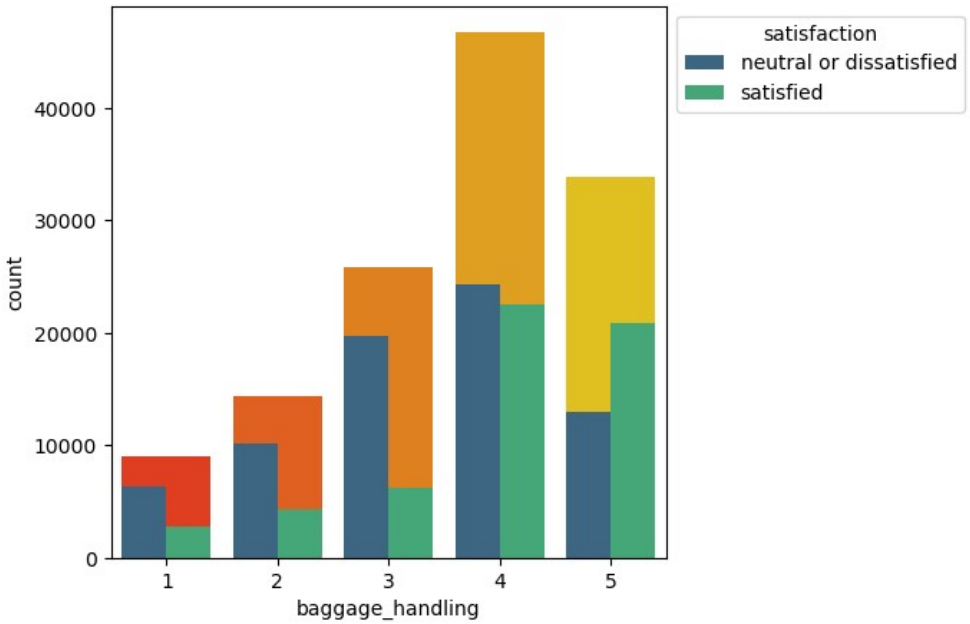


Fig 2.16 Countplot for the variable baggage handling with hue as satisfaction

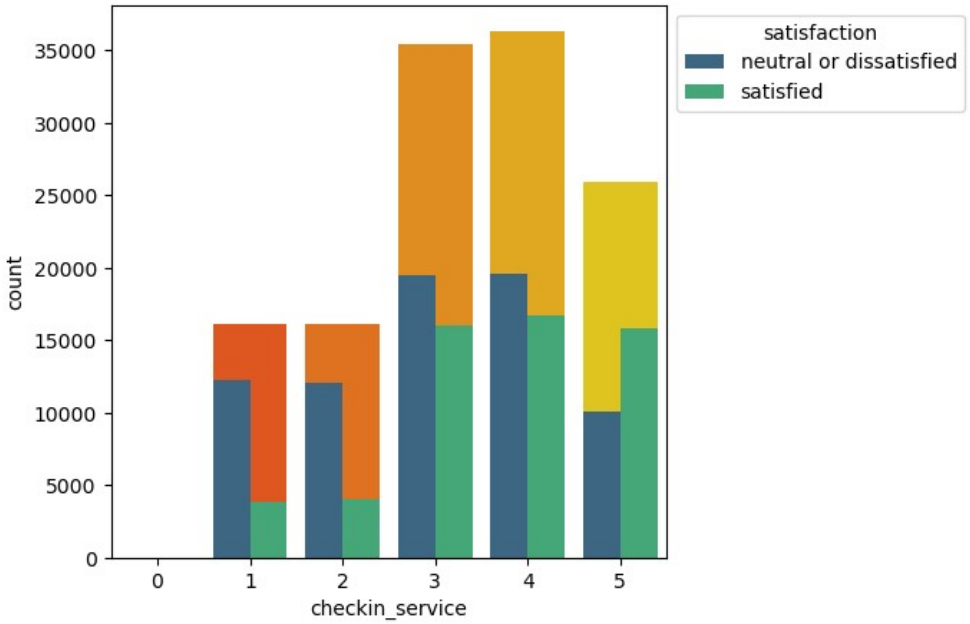


Fig 2.17 Countplot for the variable checkin_service with hue as satisfaction

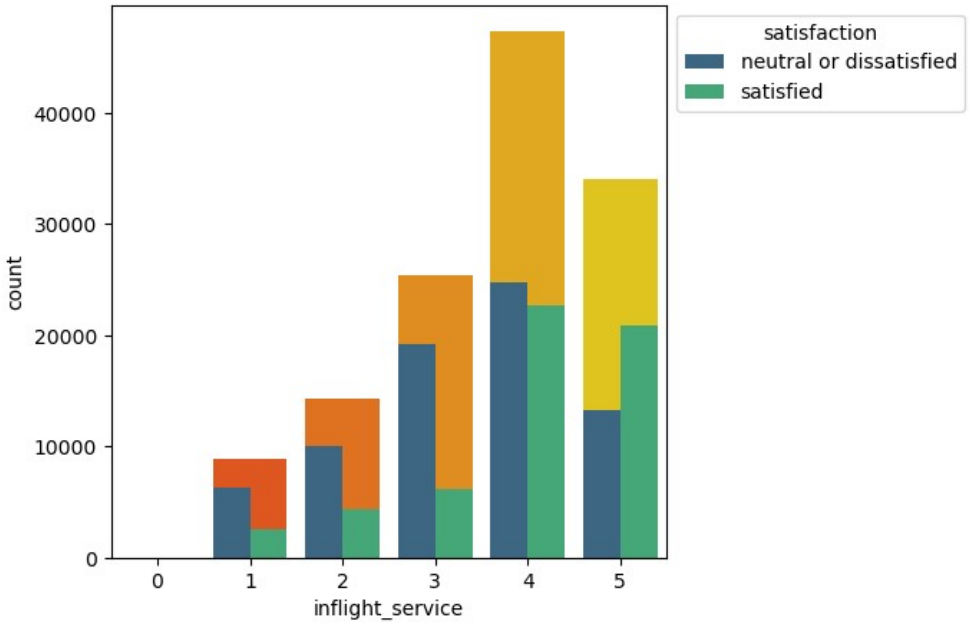


Fig 2.18 Countplot for the variable inflight_service with hue as satisfaction

11) Cleanliness

It is very clear from the count plot that the passengers who have rated cleanliness as 1 and 2 are not at all satisfied with the airlines.

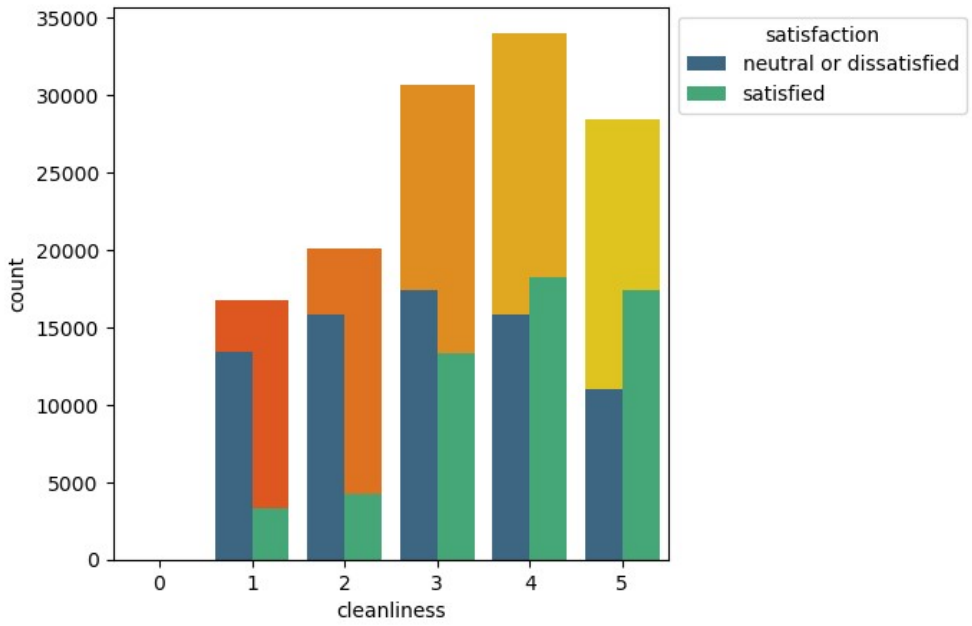


Fig 2.19 Countplot for the variable cleanliness with hue as satisfaction

12) Age

We found that the distribution of the age to be uniformly distributed. we have sufficient entries to represent all groups of ages in the dataset.

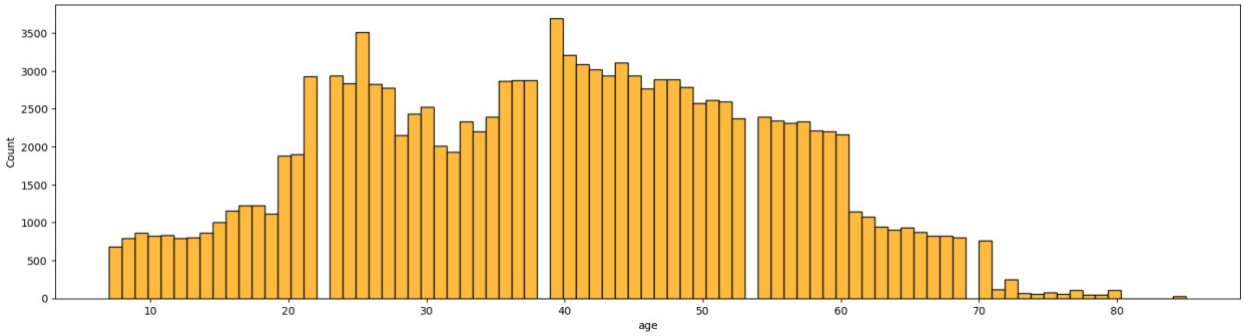


Fig. 2.20 countplot of the variable ages

We can also conclude from fig 2.21 that elder people with age greater than 40 are more satisfied with the airlines as compared to the people aged less than 40.

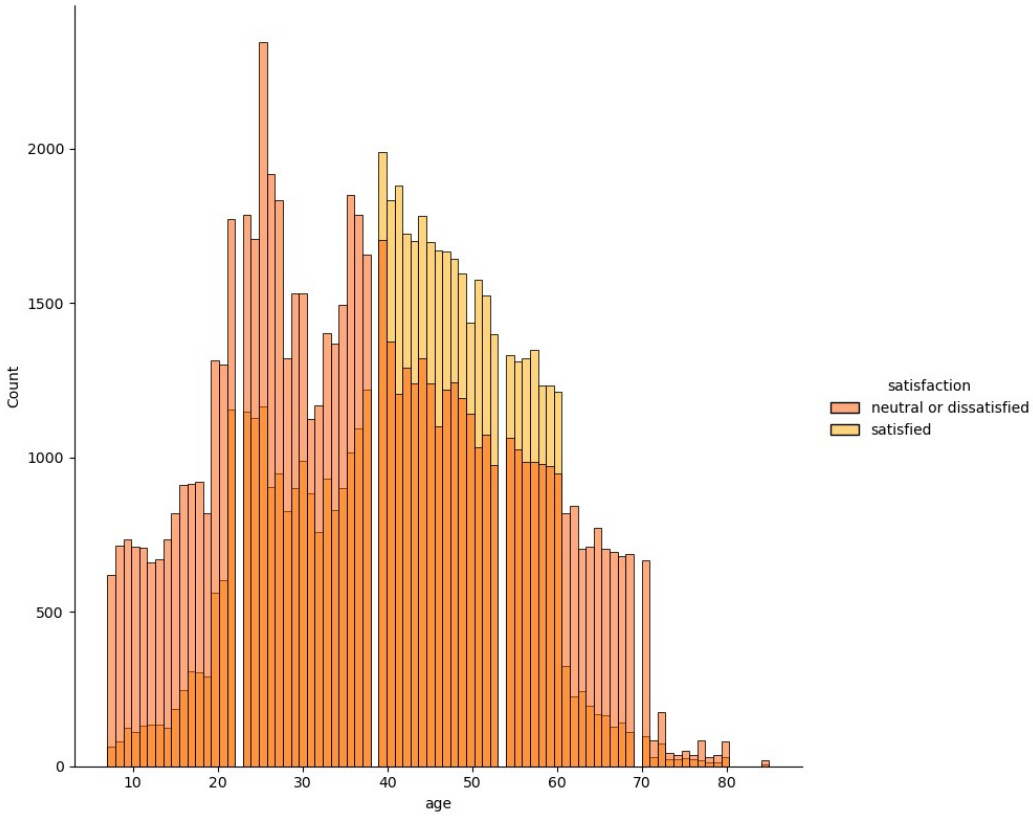


Fig. 2.21 countplot of the variable ages with hue as satisfaction

13) Flight distance:

For the flight_distance variable we have the data right skewed which means we have less entries for larger flight distance as compared to

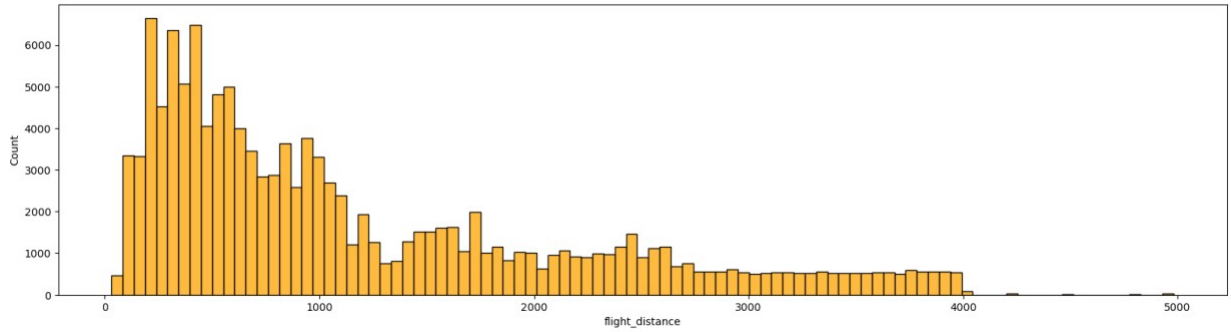


Fig. 2.22 countplot of the variable flight_distance

It is evident that flyer with larger flight distances are more satisfied than flyers with short distance flights.

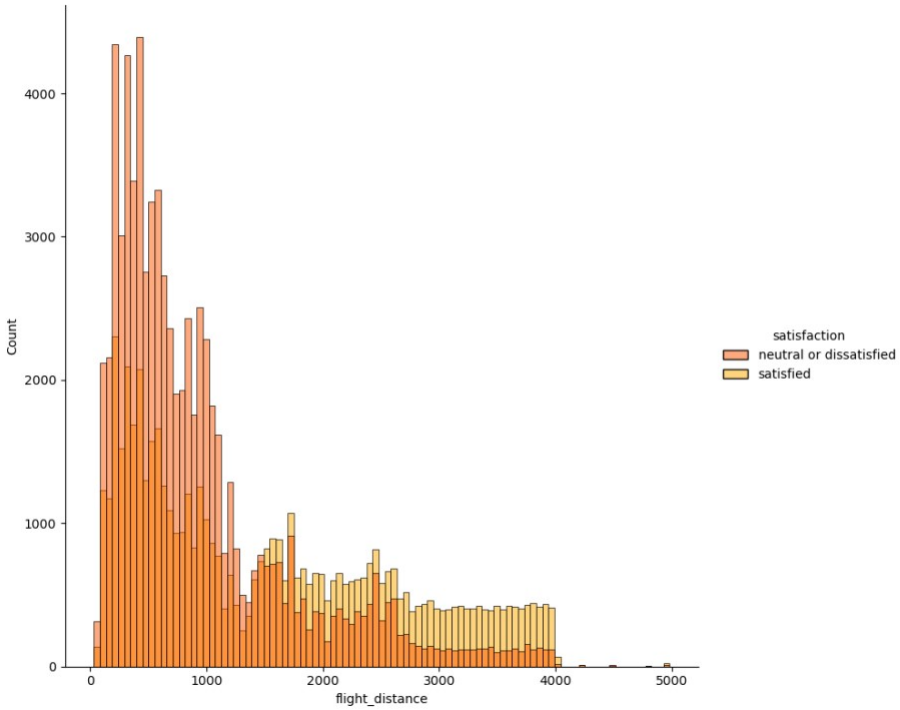


Fig. 2.23 countplot of the variable flight_distance with hue as satisfaction

Relationship between the travel type and the flight distance with hue as satisfaction can be found in the figure 2.24

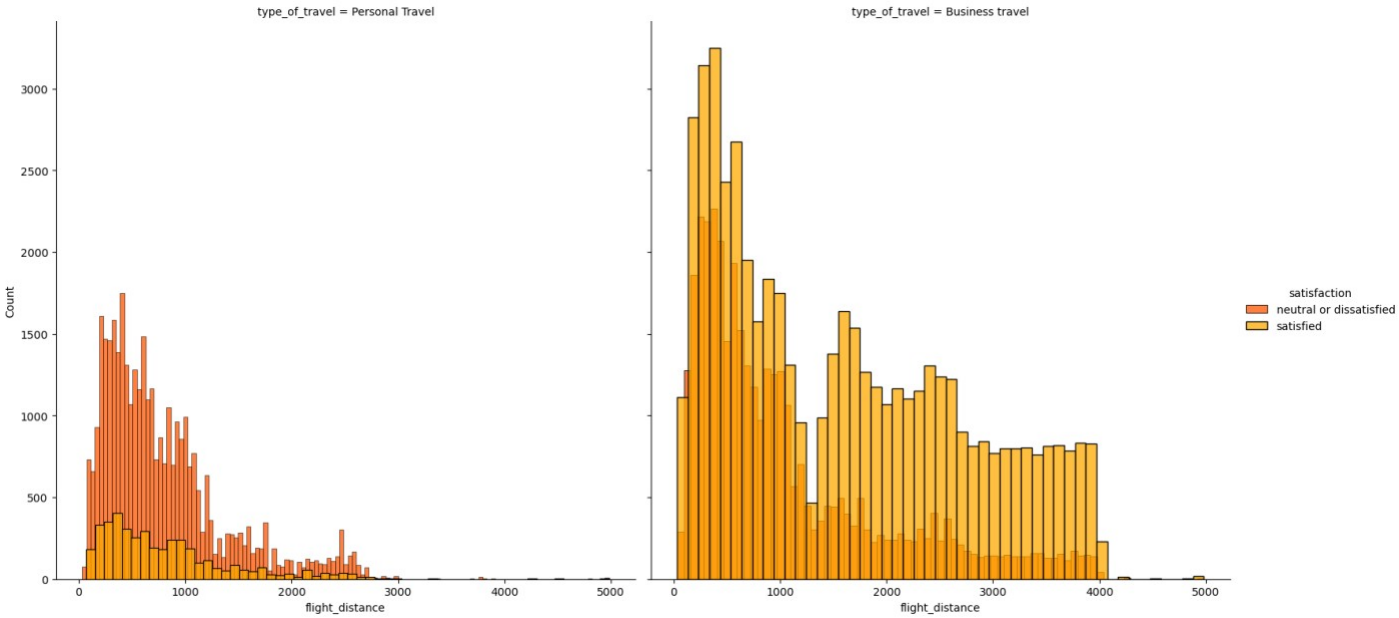


Fig 2.24 countplot of variable flight disrtance with hue as satisfaction and separated based on type_of_travel

14) Departure delay in minutes:

It is an obvious observation that longer departure delay in flights tends to have a negative impact on the satisfaction of the passenger.

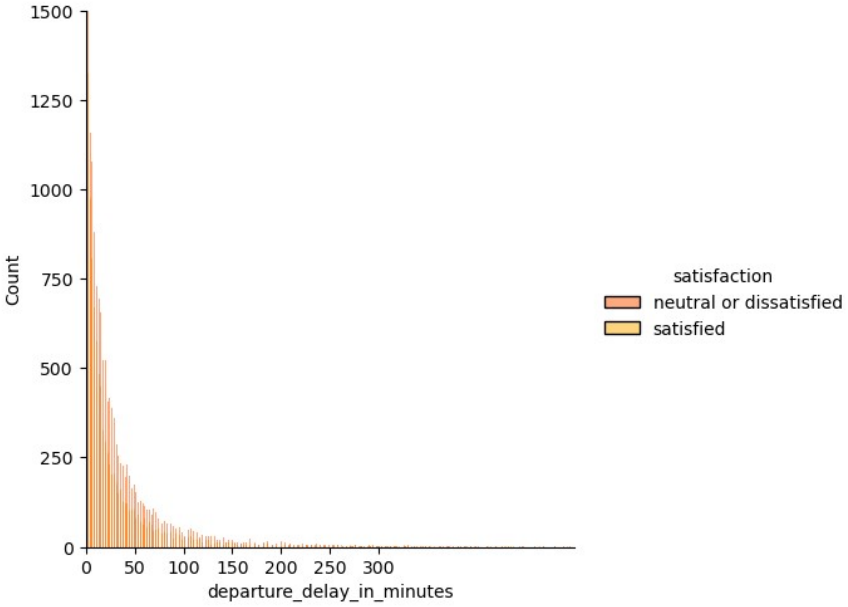


Fig 2.25 countplot of variable departure_delay_in_minutes with hue as satisfaction

15) Arrival delay in minutes:

Like departure delay longer arrival delays can also be attributed to dissatisfaction of the customers.

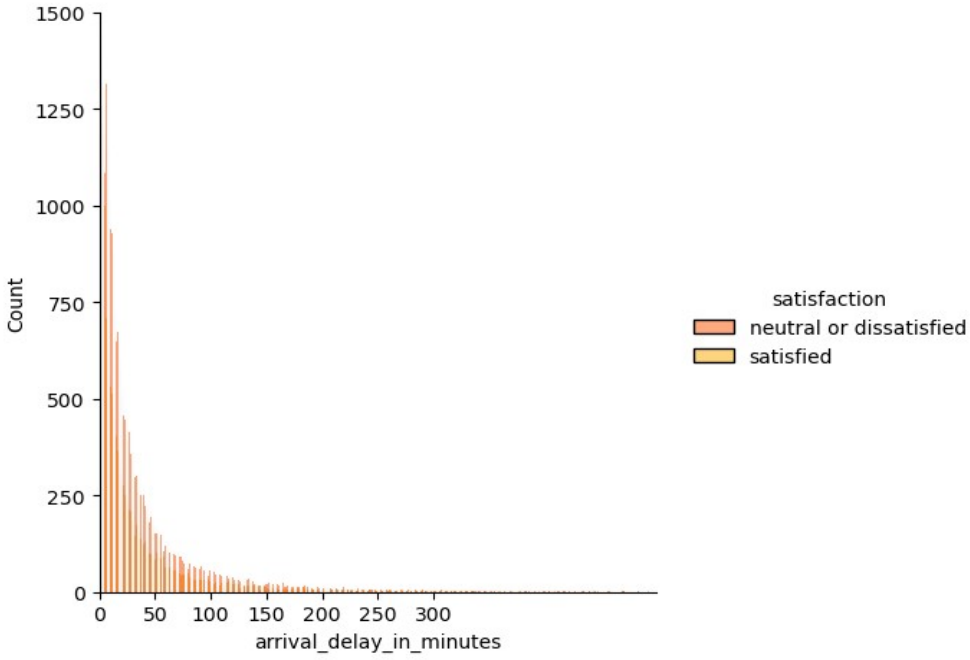


Fig 2.25 countplot of variable arrival_delay_in_minutes with hue as satisfaction

Correlation between variables

In order to get the correlation between different variables we plotted a heatmap of the numerical features.

HeatMap

We also plotted a heatmap to find the correlation among the features. From the heatmap it is very clear that most of the features are less correlated with each other except for the 'departure_delay_in_minutes' and 'arrival_delay_in_minutes'. They are showing a very strong correlation of .97. To dig further deep into it we will plot a scatterplot between both the variables.

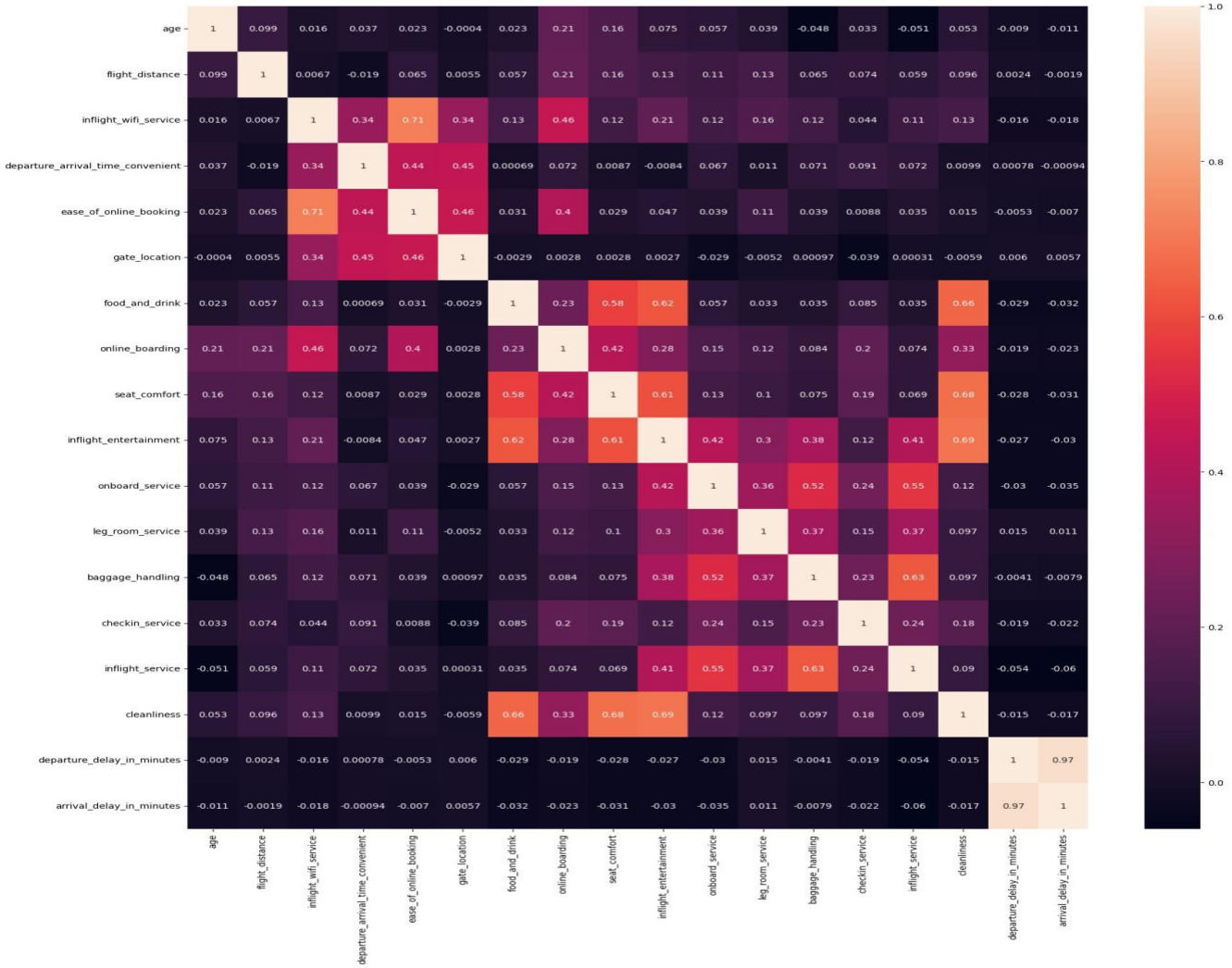


Fig. 2.26 Heatmap to show correlation between variables

It is pretty evident from the scatterplot that both the variables share a linear relationship. It is very intuitive that if a flight arrives late it will have a late departure, hence we can eliminate anyone of them as a part of our feature selection strategy.

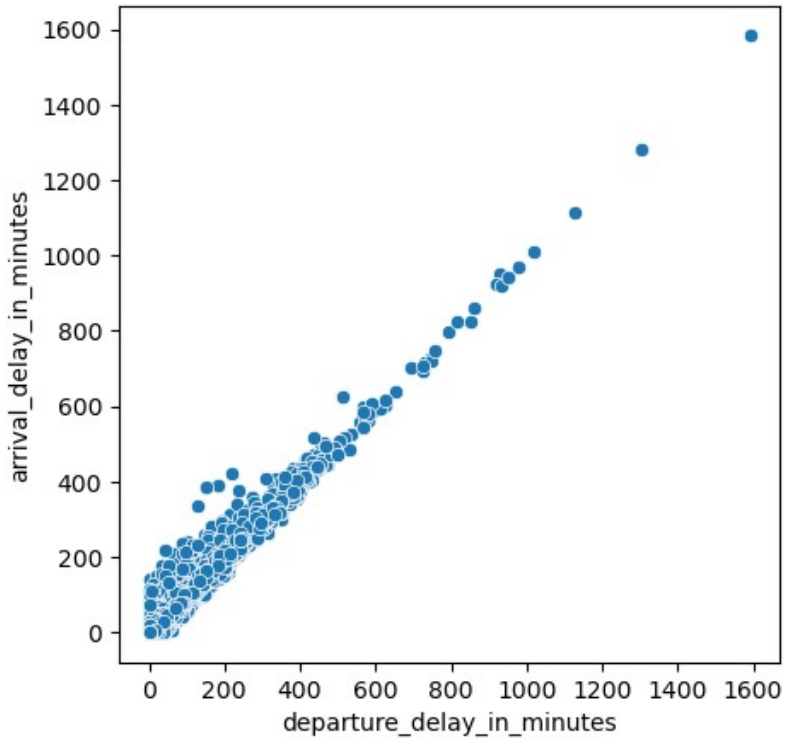


Fig. 2.26 Scatterplot between variables arrival_delay_in_minutes and variable departure_delay_in_minutes

Phase -2

Classification:

Classification is a type of supervised learning in machine learning where the goal is to predict the category or class of a given input data point based on a set of training data. The input data is usually represented as a set of features or variables that describe the characteristics of the data. The classification algorithm then learns to map these features to the corresponding output class by analyzing a set of labeled training examples. Once the model has been trained, it can be used to predict the output class of new, unseen input data.

The project on analyzing Airline passenger satisfaction comes under classification problem to be more specific binary classification problem as we have only two classifications i.e satisfied and unsatisfied or neutral. Based on input data samples, the

known samples will be used as training data and once the model learns based on the input sample, we will detect if the passenger is satisfied or not.

Dataset Split:

Before applying the machine learning models, as part of the initial process we need to transform features of our dataset to fit as train and test samples for our model. Majority of machine learning models require numeric values as input instead of strings. This lead us to encode our data samples into numbers instead of text or strings. Also, for any machine learning model, we first need to train the model on some input samples and finally test some known test samples. To do it we need to split the input samples into training and testing data.

Let's break the dataset into 2 subsets, one for training the model and the other is to test the model.To do this,we used the train_test_split **method from sklearn.model_selection**. Below is the screenshot from the code

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

Fig. 3.0 Screenshot of code for data splitting into training and testing data

The outcome after running this code would be 4 different portions X_train, X_test, y_train, y_test. We have splitted the dataset into 2 parts with size of 75% as training data sample testing data sample as 25% of the whole data set.
Lets now apply the various classification algorithms on the training data subset and use the remaining test data subset to test our model. Upon research, although there are various classification algorithms available to analyze airline passenger satisfaction data, we will be demonstrating the top 6 optimized and efficient algorithms in this report. We will also try to explain metrics of each model along with explanations on what it means.
We have obtained the results using certain libraries in the sklearn.

Feature Scaling:

Feature scaling, also referred to as data normalization, is a technique employed to standardize the range of independent variables or features in a given dataset. Typically implemented during the data preprocessing phase, this method is necessary as raw data can have widely varying ranges of values. Failure to normalize the data may result in objective functions malfunctioning in some machine learning algorithms.

```
from sklearn.preprocessing import StandardScaler

# Normalize Features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Fig. 3.1 Screenshot of code for scaling the data using standard scaler

Model Evaluation:

We have used primarily four import library functions to get the model evaluated. Below are

a. Confusion matrix:

A confusion matrix is a 2d array which summarizes the whole classification summary of prediction results.

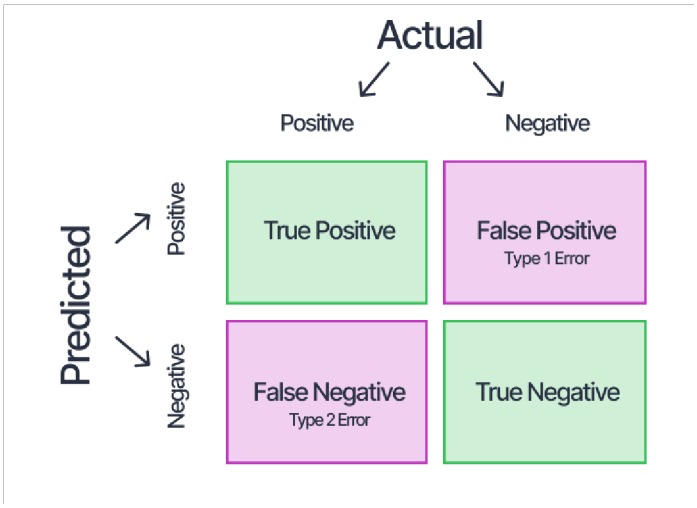


Fig. 3.2 A generic confusion matrix

b. Accuracy: measure of how often a model correctly predicts the outcome of a given data point

c. ROC Curve and AUC: It is a plot of the true positive rate against the false positive rate and AUC is the area under this curve, which is a measure of the model's ability to distinguish between positive and negative classes

d. Classification Report: is a summary of the performance of a classification model, which includes metrics such as precision, recall, F1 score, and support for each class. The screenshot provided below gives the code we have implemented to calculate the above mentioned metrics. We have used prebuilt libraries from sklearn.

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix, roc_auc_score, accuracy_score, classification_report

#confusion matrix visualization
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import RocCurveDisplay
```

Fig. 3.3 Screenshot of code for importing libraries used to calculate various metrics

```
def ModelResults(model, X_train, y_train, X_test, y_test,modelName, verbose=True):

#     Fitting the model in to train data
    model.fit(X_train,y_train)

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

#     getting the roc_auc score
    roc_auc = roc_auc_score(y_test, y_pred)
    print("ROC AUC = {}".format(roc_auc))

#     getting the classification report
    print(classification_report(y_test,y_pred,digits=5))

#     getting the confusion matrix
    cm=confusion_matrix(y_test,y_pred)

#     plotting the confusion matrix
    f, ax=plt.subplots(figsize=(5,5))
    sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    title = "confusion matrix for "+modelName
    plt.title(title,y=-0.2)
    plt.show()

#     generating the ROC curve
    y_proba = model.predict_proba(X_test)

    # Plots the ROC curve using the sklearn methods
    plot_sklearn_roc_curve(y_test, y_proba[:, 1])
    return roc_auc
```

Fig. 3.4 Screenshot of code to calculate various metrics for different models

```
# Plots the ROC curve using the sklearn methods
def plot_sklearn_roc_curve(y_real, y_pred):
    fpr, tpr, _ = roc_curve(y_real, y_pred)
    roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
    roc_display.figure_.set_size_inches(5,5)
    plt.plot([0, 1], [0, 1], color = 'g')
```

Fig. 3.5 Screenshot of code for plotting roc curve

Models and their respective performance metrices

1. Logistic Regression:

Logistic regression is a statistical method used to model the relationship between a binary dependent variable and one or more independent variables.

Reason to choose Logistic regression: To start with, we have chosen logistic regression as our initial model as it is one of the very popular and specially designed for modeling probabilities of binary outcome problems. In our project, the outcome is binary classification with only two outcomes with satisfied and unsatisfied/neutral results. Besides, this since logistic regression is very easy to implement and interpret the results

Model Building:

This algorithm is easy to implement. We have imported the LogisticRegression method from sklearn library and fit our training data subset into this model. Below is the screenshot:

```
#  Logistic regression
from sklearn.linear_model import LogisticRegression
modellr = LogisticRegression()
modellr, rocauclr, cm = ModelResults(modellr, X_train, y_train, X_test, y_test,'Logistic Regression')
```

Fig. 3.6 Screenshot of code for applying logistic regression classifier on the training dataset and getting different metrics on test data

Model Evaluation:

We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

- True Positive: satisfied satisfaction and predicted as satisfied satisfaction
- True Negative: Not satisfied satisfaction and predicted as not satisfied satisfaction
- False Positive: Not a satisfied satisfaction and predicted as satisfied satisfaction
- False Negative: satisfied satisfaction and predicted as not satisfied satisfaction

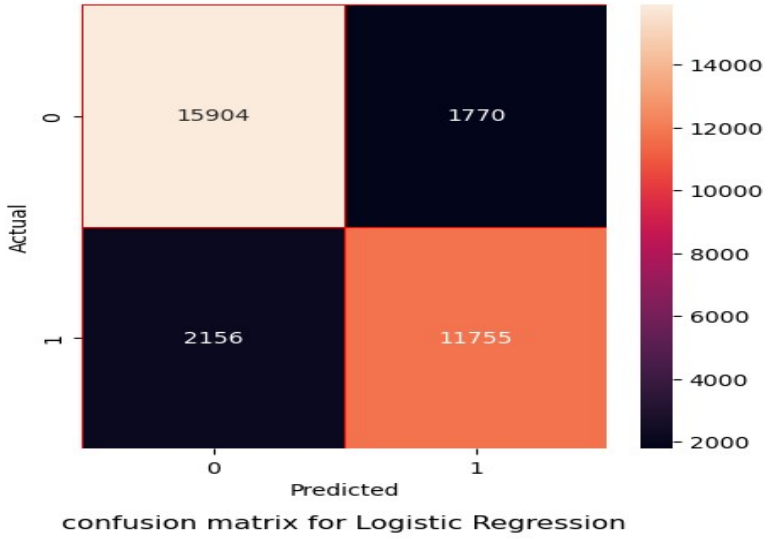


Fig. 3.7 confusion matrix for Logistic Regression

From the above matrix, it is clear that 15904 satisfied satisfaction is predicted as satisfied and 11755 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. In comparison with true positives and true negatives, false positives and false negatives are insignificant. So, therefore we can conclude that this model is a good fit for predicting satisfied or unsatisfied/neutral satisfaction.

b. Accuracy:

accuracy	0.92129	31585

Since the accuracy is 0.9219 , i.e 92.19% we can conclude that this model is good fit for our data samples

c. ROC Curve and AUC:

ROC:

Receiver operating characteristic curve is a graph that shows the performance of a classification model at all classification thresholds. ROC plots will have 2 parameters. True Positive rate False Positive rate.

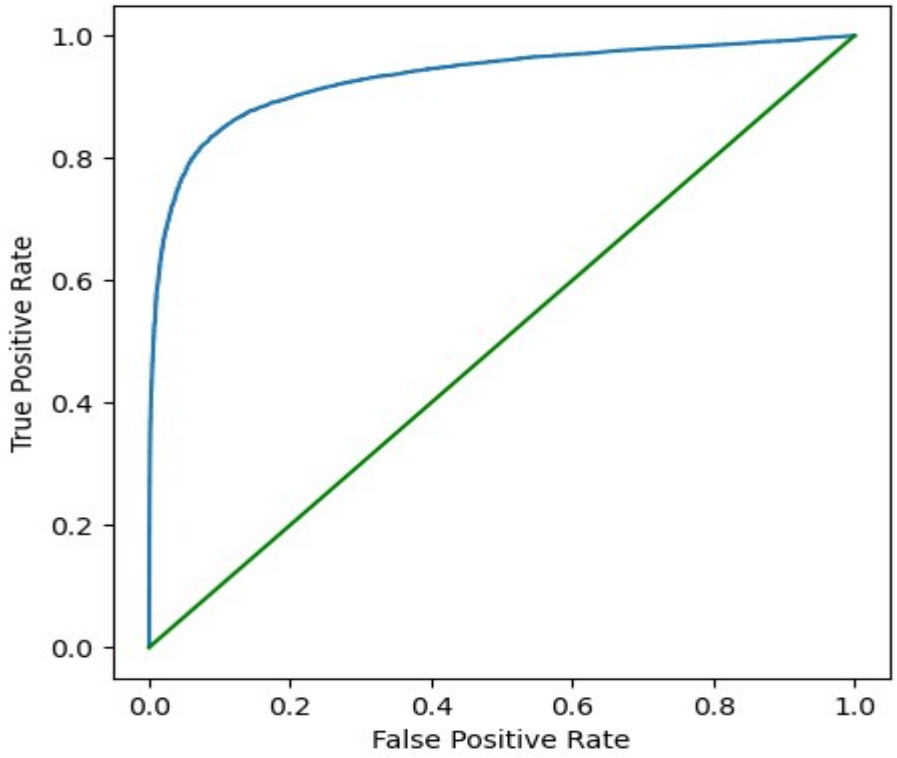


Fig. 3.8 ROC Curve for Logistic Regression

AUC:

Area under the ROC Curve, that measures the entire 2 dimensional area under the ROC curve. It provides a measure of performance for all the classification thresholds.

```
ROC AUC = 0.9160566643830373
```

AUC score is 0.916 This evaluates our model is pretty good at predicting the label as satisfied or unsatisfied/neutral

d. Classification Report:

This is the report that shows Precision, recall, f1 scores.

	precision	recall	f1-score	support
0	0.90510	0.96000	0.93174	17674
1	0.94493	0.87212	0.90707	13911
accuracy			0.92129	31585
macro avg	0.92502	0.91606	0.91940	31585
weighted avg	0.92264	0.92129	0.92087	31585

Fig. 3.9 Screenshot of code output for classification report

In the above report, we can clearly say that Precision is 0.9051 to predict satisfaction and 0.94493 to predict unsatisfied or neutral satisfaction. Recall is 0.96 for satisfaction and 0.8721 for unsatisfied or neutral satisfaction. The weighted harmonic mean of precision and recall is the f1-score. As precision and recall are embedded in their computation, f1-score is always lower than accuracy measurements. To compare classifier models, we should consider the weighted average of f1-score rather than global accuracy. Weighted average f1-score is 0.93 and 0.90 respectively.

Therefore, we can conclude that the relationship between the input features and the output variable is approximately linear on the log-odds scale and with very few irrelevant samples i.e outliers.

2. KNN (K-Nearest Neighbor)

KNN works by computing the distances between a given input and all the data points in the training set, then selecting the K closest data points based on those distances.

Reason to choose KNN: we have chosen KNN as our second model since KNN is a good fit if we have a small dataset with sparse data in it. It also doesn't need to understand the relationship between data points as it makes no assumptions about the underlying distribution of the data. Also, the best part of this algorithm is there is no need to build algorithm and fine tune many parameters unlike other models

Model Building:

We used the imported KNeighborsClassifier method from sklearn library and fit our training data subset into this model. Below is the screenshot:

```
# KNN
from sklearn.neighbors import KNeighborsClassifier
modelknn = KNeighborsClassifier()
modelknn, rocaucknn = ModelResults(modelknn, X_train, y_train, X_test, y_test, 'KNN classifier')
```

Fig. 3.10 Screenshot of code for applying KNN classifier on the training dataset and getting different metrics on test data

Model Evaluation:

We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy here as well. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

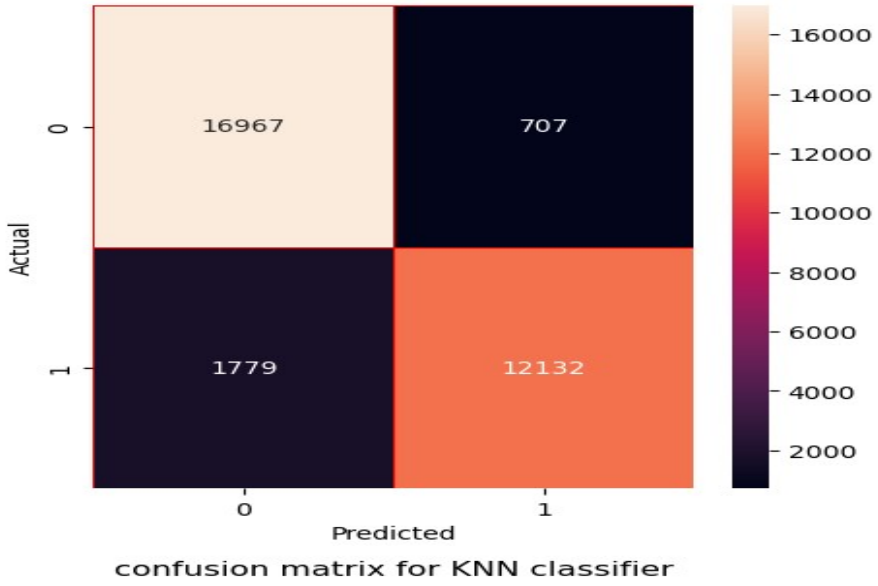


Fig. 3.11 confusion matrix for KNN classifier

From the above matrix, it shows that 16967 satisfied satisfaction is predicted as satisfied and 12132 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. false positives and false negatives are insignificant as it has very few in number. In comparison with logistic regression we can see very few false positives and false negatives So, therefore we can conclude that this model is a good fit for predicting satisfied or unsatisfied/neutral satisfaction and has better performance than logistic regression

b. Accuracy:

```
accuracy                    0.94941
```

Since the accuracy is 0.9491 , i.e 94.91% , there is increase in model accuracy on comparison with logistic regression we can conclude that this model will have better performance than logistic regression

c. ROC Curve and AUC: ROC Curve:

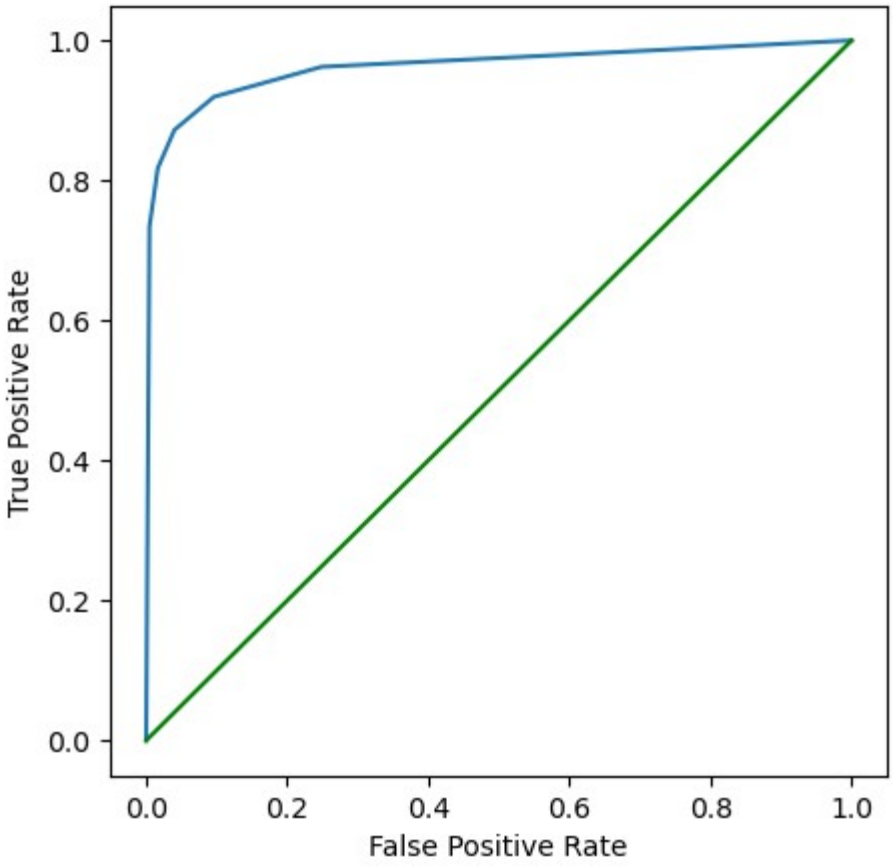


Fig. 3.12 ROC Curve for KNN classifier

AUC:

```
ROC AUC = 0.9458081564069658
```

AUC score is 0.9458 This evaluates our model is pretty good at predicting the label of the news as satisfied or unsatisfied/neutral

d. Classification Report:

Precision, recall, f1 scores are shown in the screenshot.

	precision	recall	f1-score	support
0	0.93628	0.97601	0.95573	17674
1	0.96778	0.91561	0.94097	13911
accuracy			0.94941	31585
macro avg	0.95203	0.94581	0.94835	31585
weighted avg	0.95015	0.94941	0.94923	31585

Fig. 3.13 Screenshot of code output for classification report

In the above report, we got Precision as 0.93625 to predict satisfaction and 0.96778 to predict unsatisfied or neutral satisfaction. Recall is 0.976 for satisfaction and 0.91561 for unsatisfied or neutral satisfaction. Weighted average f1-score is 0.955 and 0.94 respectively.
Overall, based on the above evaluation metrics, the KNN model appears to perform better than the logistic regression model, the reason for being insensitive to outliers when capturing non-linear relationships between the features and the target variable. As a result we can see there has been increase of ~2% accuracy on comparison with
logistic regression

3. Random Forest

Random Forest is a machine learning algorithm that creates multiple decision trees.
Each decision tree is built using a random subset of the available data and features. The algorithm then combines the predictions of all the trees to arrive at a final prediction.[6]
Reason to choose random forest: we have chosen random forest as our third model since it is highly effective at solving complex problems because it can handle non-linear relationships and interactions between features. Because of the large number of decision trees involved in the process, random forests are regarded as a highly accurate and robust method. It does not have the overfitting problem.
Model Building:
Just like previous models, we imported RandomForestClassifier method from sklearn.ensemble library and fit our training data subset into this model. Below is the screenshot:

```
#  Random Forest
from sklearn.ensemble import RandomForestClassifier
modelrf = RandomForestClassifier()
modelrf, rocaucrf = ModelResults(modelrf, X_train, y_train, X_test, y_test, 'Random Forest Classifier')
```

Fig. 3.14 Screenshot of code for applying Random Forest classifier on the training dataset and getting different metrics on test data

Model Evaluation:
We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy here as well. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

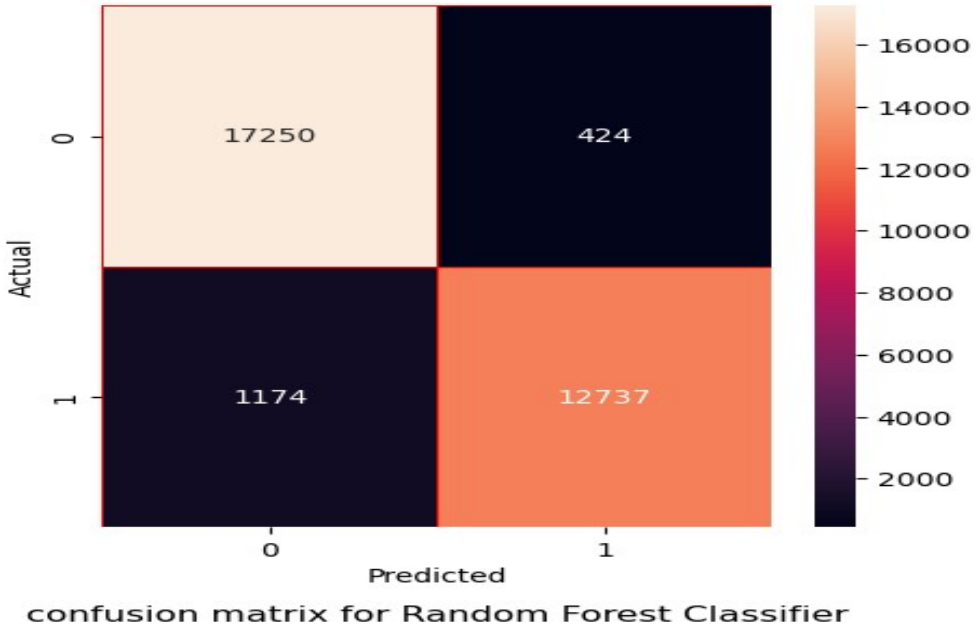


Fig. 3.15 confusion matrix for Random forest clasifier

From the above matrix, it shows that 17250 satisfied satisfaction is predicted as satisfied and 12737 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. false positives and false negatives are insignificant as it has very few in number. In comparison with logistic regression and KNN we can see very few false positives and false negatives So, therefore we can conclude that this model is a good fit for predicting satisfied or unsatisfied/neutral satisfaction and has better performance than other two models

b. Accuracy

```
accuracy                          0.95036
```

Since the accuracy is 0.9503 , i.e 95.03% , there is increase in model accuracy on comparison with logistic regression we can conclude that this model will have better performance than logistic regression

c. ROC Curve and AUC

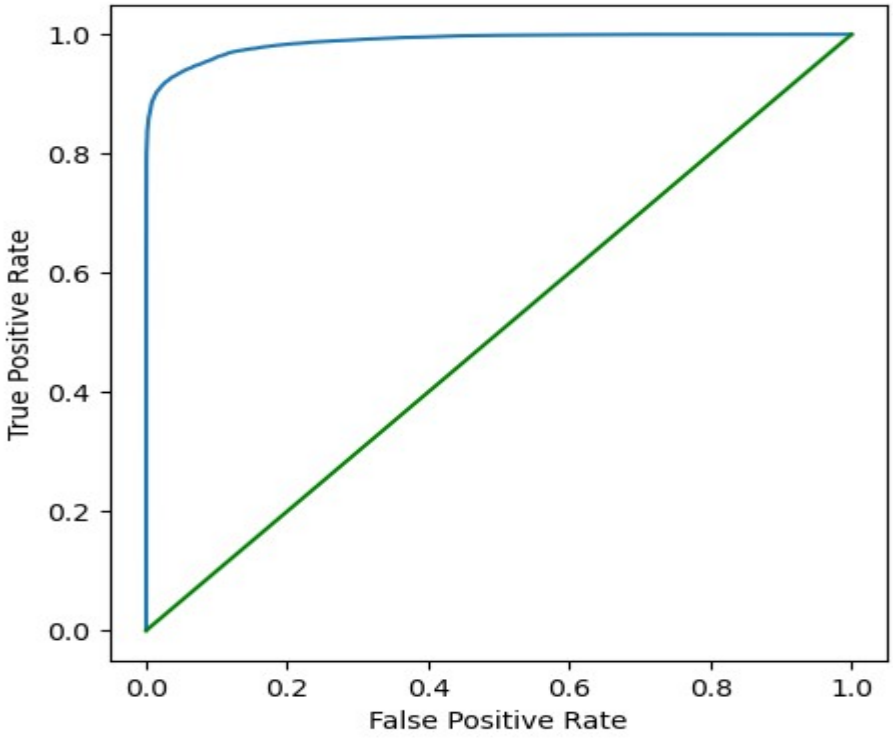


Fig. 3.16 ROC Curve for Random forest classifier

AUC:

```
ROC AUC = 0.9471313241120521
```

AUC score is 0.9471 This evaluates our model is pretty good at predicting the label of the news as satisfied or unsatisfied/neutral

d. Classification Report:

	precision	recall	f1-score	support
0	0.93933	0.97420	0.95645	17674
1	0.96560	0.92006	0.94228	13911
accuracy			0.95036	31585
macro avg	0.95247	0.94713	0.94937	31585
weighted avg	0.95090	0.95036	0.95021	31585

Fig. 3.17 Screenshot of code output for classification report In the above report, we got Precision as 0.93933 to predict satisfaction and 0.9656 to predict unsatisfied or neutral satisfaction. Recall is 0.9742 for satisfaction and 0.92 for unsatisfied or neutral satisfaction. Weighted average f1-score is 0.956 and 0.942 respectively.
Overall, based on the evaluation metrics, the random forest model appears to perform better than the logistic regression model but almost equal performance in comparison with KNN because random forest captures complex non-linear relationships between the input features and the target variable. We can conclude that our data samples are having some non-linear relationship

4. Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm used for classification tasks. It assumes that the features are independent of each other, given the class label. It calculates the posterior probability of each class for a given set of features and chooses the class with the highest probability as the predicted class.[5][7]

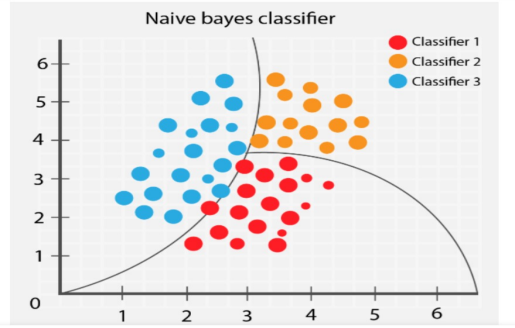


Fig. 3.18 Generic Naive Baye's classifier algorithm

Reason to choose naive bayes: we have chosen naive bayes as our fourth model since it is highly effective when working with high-dimensional datasets. Naive Bayes assumes independence between features, which makes it a good choice. The good part of this algorithm is it is robust to irrelevant features, which makes it a good choice when working with noisy data and easy to implement and interpret the results
Model Building:
Just like previous models, we imported GaussianNB method from sklearn.naive bayes library and fit our training data subset into this model. Below is the screenshot:

```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
modelnv = GaussianNB()
modelnv, rocaucnv = ModelResults(modelnv, X_train, y_train, X_test, y_test, 'Naive Bayes CLassifier')
```

Fig. 3.19 Screenshot of code for applying Naive Baye's classifier on the training dataset and getting different metrics on test data

Model Evaluation:

We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy here as well. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

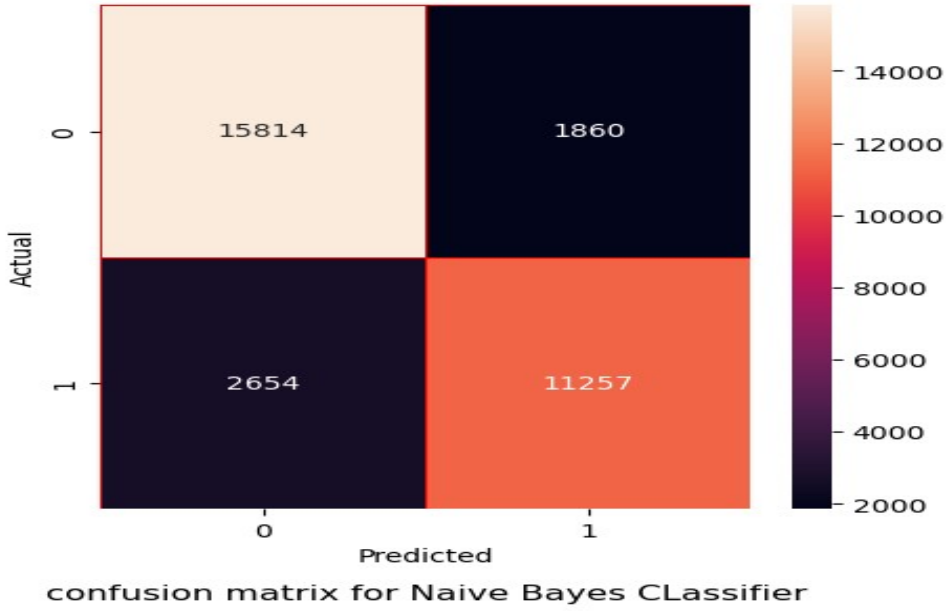


Fig. 3.20 confusion matrix for Naive Baye's Classifier

From the above matrix, it shows that 15814 satisfied satisfaction is predicted as satisfied and 11257 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. false positives and false negatives are insignificant as it has very few in number. In comparison with logistic regression and KNN we can see very few false positives and false negatives **b. Accuracy**

```
accuracy                    0.85708
```

Since the accuracy is 0.8570 , i.e 85.70% , there is a decrease in model accuracy in comparison with all other models. we can conclude that this model will have least performance in comparison with all other models.

c. ROC Curve and AUC

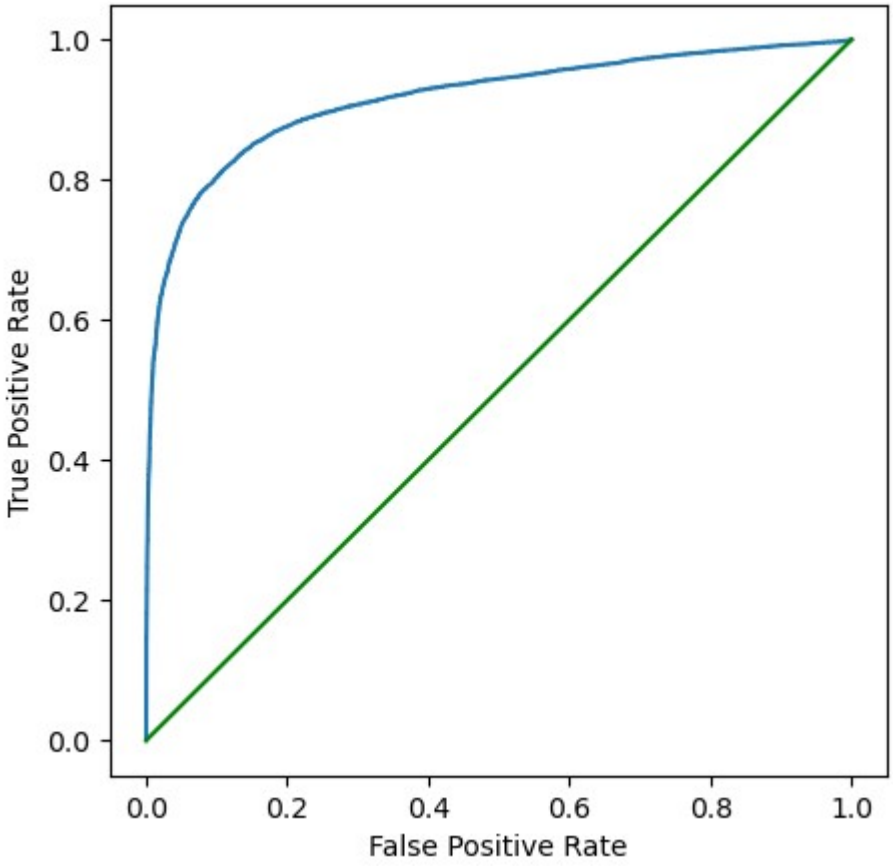


Fig. 3.21 ROC Curve for Naive Baye's classifier

AUC:

```
ROC AUC = 0.8519881969721562
```

AUC score is 0.8519 This evaluates our model is pretty good at predicting the label of the news as satisfied or unsatisfied/neutral

d. Classification Report:

	precision	recall	f1-score	support
0	0.85629	0.89476	0.87510	17674
1	0.85820	0.80922	0.83299	13911
accuracy			0.85708	31585
macro avg	0.85725	0.85199	0.85405	31585
weighted avg	0.85713	0.85708	0.85655	31585

Fig. 1.22 Screenshot of code output for classification report

In the above report, we got Precision as 0.85629 to predict satisfaction and 0.8582 to predict unsatisfied or neutral satisfaction. Recall is 0.8947 for satisfaction and 0.809 for unsatisfied or neutral satisfaction. Weighted average f1-score is 0.875 and 0.832 respectively.

Overall, based on the evaluation metrics, the naive bayes appear to perform worse than the other models. This may be because of the non-linear relationship between input and output samples and also naive bayes perform worse if there are any outliers in it.

5. Extreme Gradient Boosting (XGBoost)

XGBoost is a machine learning algorithm that leverages a group of decision trees to generate predictions. It excels at processing structured data with well-defined features and has become well-liked for its swiftness and precision. By integrating gradient boosting principles with diverse regularization strategies, XGBoost helps counteract overfitting and enhances generalization.[5][9][10]

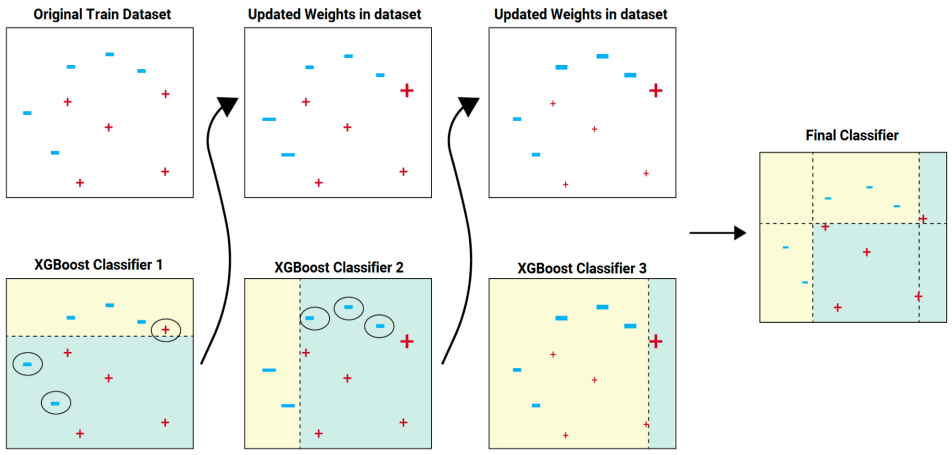


Fig. 3.23 General Xgboost algorithm process

Reason to choose naive bayes: we have chosen XG Boosting as our fifth model since it is considered as one of the algorithms with highest accuracy and better performances. It can handle large datasets with inbuilt regularization like L1, L2, L3 for early stopping to prevent overfitting problems. It also boosts the important features which helps to identify the important features.

Model Building:

Just like previous models, we imported xgboost method and fit our training data subset into this model. Below is the screenshot:

```
# XGB
import xgboost as xgb
modelxgb = xgb.XGBClassifier()
modelxgb, rocaucxgb = ModelResults(modelxgb, X_train, y_train, X_test, y_test, 'XG boost classifier')
```

Fig. 1.24 Screenshot of code for applying XG Boost classifier on the training dataset and getting different metrics on test data

Model Evaluation:

We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy here as well. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

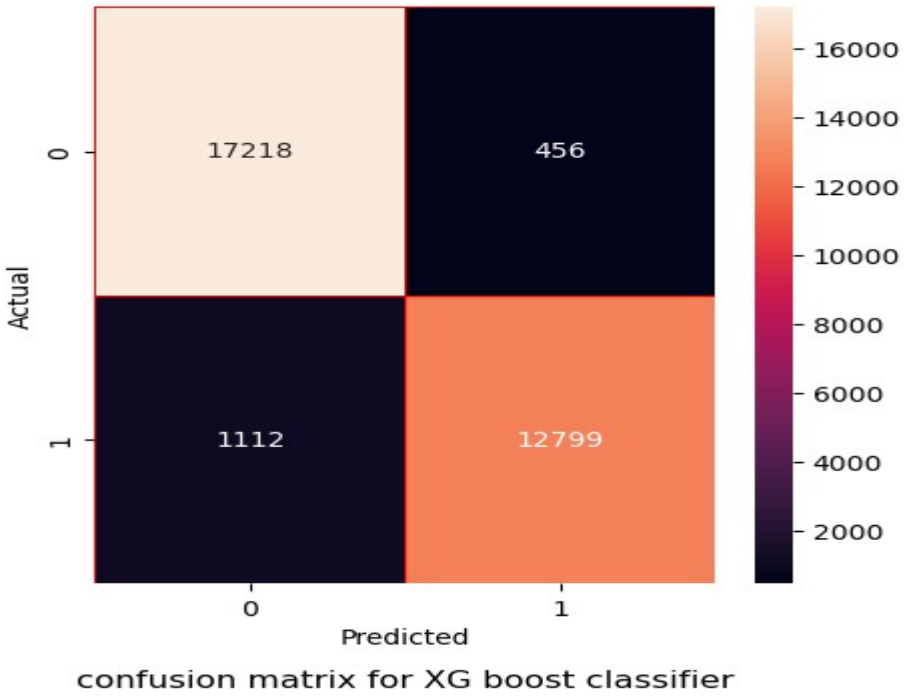


Fig. 3.25 confusion matrix for XG Boost classifier

From the above matrix, it shows that 17218 satisfied satisfaction is predicted as satisfied and 12799 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. false positives and false negatives are insignificant as it has very few in number. In comparison with other models, we can see very few false positives and false negatives

b. Accuracy:

```
accuracy                                    0.87570
```

Since the accuracy is 0.8757 , i.e 87.57% , there is a decrease in model accuracy in comparison with all other models. we can conclude that this model will have better performance than naive bayes but least performance on comparison with all other models

c. ROC Curve and AUC:

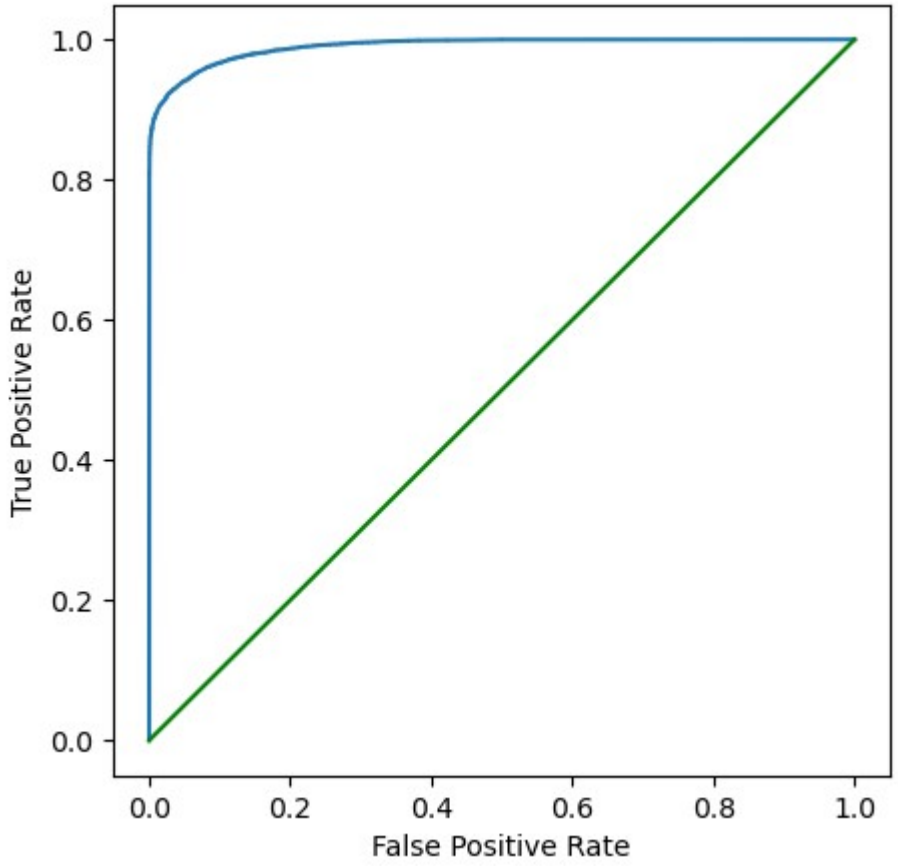


Fig. 3.26 ROC Curve for XG Boost classifier

AUC:

```
ROC AUC = 0.8724338138960583
```

AUC score is 0.8724 This evaluates our model is pretty good at predicting the label of the news as satisfied or unsatisfied/neutral but least performance than all other models except naive bayes

d. Classification Report:

Precision, recall, f1 scores are shown in the screenshot.

	precision	recall	f1-score	support
0	0.88062	0.89985	0.89013	17674
1	0.86913	0.84501	0.85690	13911
accuracy			0.87570	31585
macro avg	0.87488	0.87243	0.87352	31585
weighted avg	0.87556	0.87570	0.87550	31585

Fig. 3.27 Screenshot of code output for classification report

In the above report, we got Precision as 0.8806 to predict satisfaction and 0.8691 to predict unsatisfied or neutral satisfaction. Recall is 0.8998 for satisfaction and 0.8450 for unsatisfied or neutral satisfaction. Weighted average f1-score is 0.8901 and 0.8569 respectively.

Overall, based on the evaluation metrics, the XGBoost appears to perform better than the naive bayes but less than all other models above. This is because XGBoost doesn't perform better incase of non-linear relationship between input sample and output. Based on all above scenarios, we can conclude our input dataset samples are having non-linear relation and any model which needs linear relation show less performance.

6. Support Vector Machines

Support Vector Machine (SVM) is a powerful machine learning technique for classification and regression problems. SVM constructs a hyperplane to separate the data into distinct classes, with the largest possible margin between them. SVM is especially useful in scenarios involving complex, high-dimensional data, and can address non-linear relationships by applying kernel functions[5][11]

Reason to choose SVM: we have chosen SVM as our sixth model since it is considered as working effectively for high dimensional spaces. Apart from this, It can work well with non-linear data with effective handling of imbalance data samples in it.

Model Building:

Just like previous models, we imported SVC method from sklearn.svm library and fit our training data subset into this model. Below is the screenshot:

```
# SVM
from sklearn.svm import SVC
modelsvm = SVC(probability=True)
modelsvm, rocaucsvm = ModelResults(modelsvm, X_train, y_train, X_test, y_test,'SVM Classifier')
```

Fig. 3.28 Screenshot of code for applying SVM classifier on the training dataset and getting different metrics on test data

Model Evaluation:

We have used different evaluation techniques like confusion matrix, ROC curve and AUC and Accuracy here as well. The calculation for these models uses final test output and actual label of output dataset.

a. Confusion matrix:

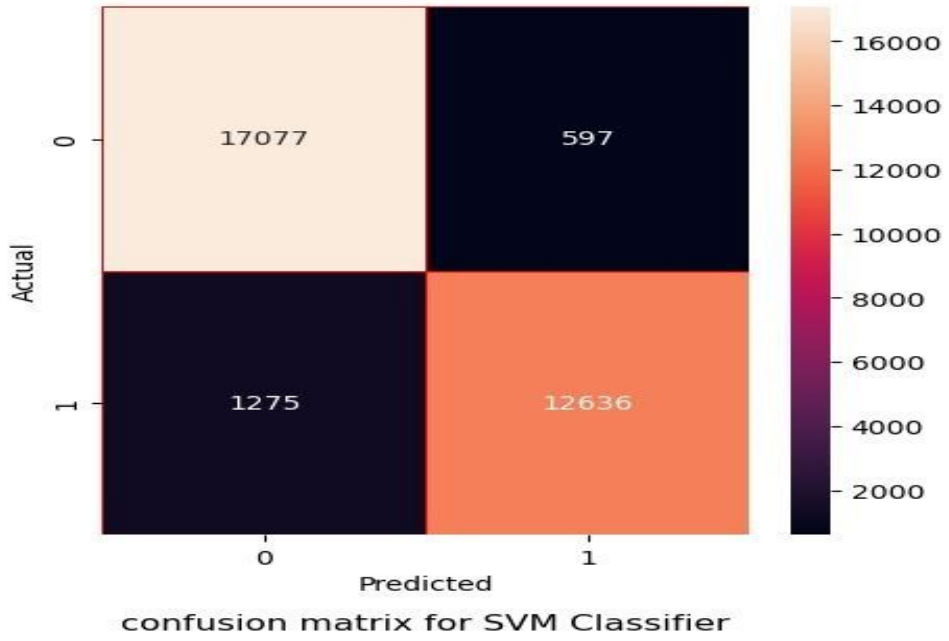


Fig. 3.29 confusion matrix for SVM classifier

From the above matrix, it shows that 17077 satisfied satisfaction is predicted as satisfied and 12636 unsatisfied satisfaction is predicted as unsatisfied and very few false positives and false negatives. false positives and false negatives are insignificant as it has very few in number. In comparison with other models, we can see very few false positives and false negatives **b. Accuracy**

accuracy 0.94073

Since the accuracy is 0.9407 , i.e 94.07% , there is a decrease in model accuracy in comparison with all other models. we can conclude that this model will have better performance than naive bayes but least performance on comparison with all other models

c. ROC Curve and AUC

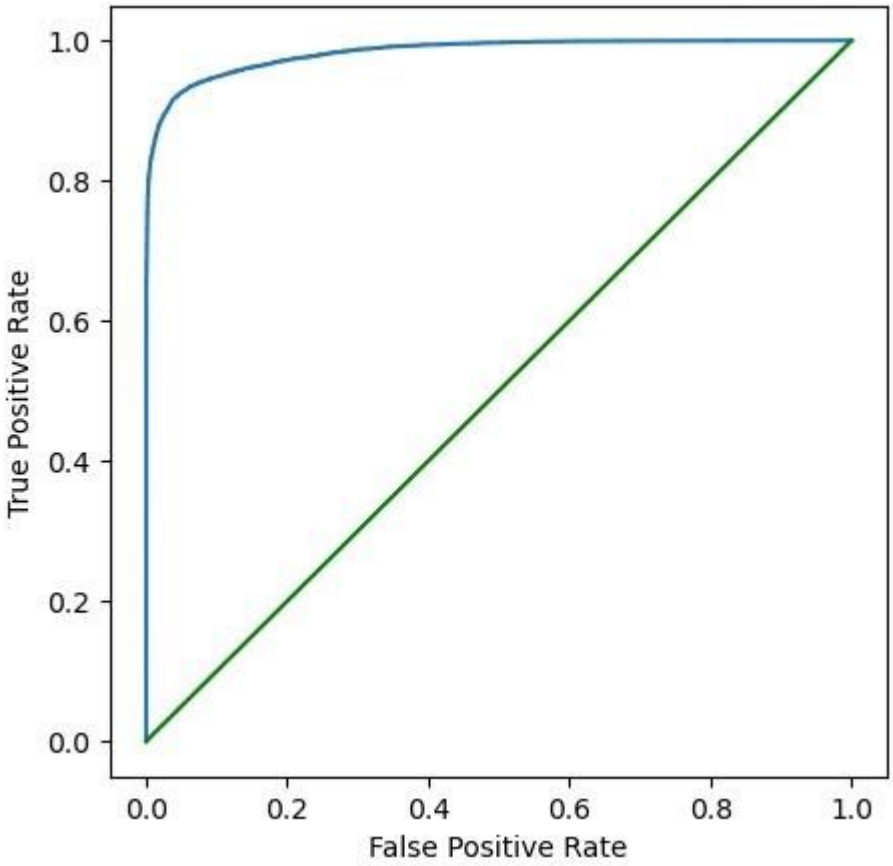


Fig. 3.30 ROC Curve for SVM classifier

AUC:

```
ROC AUC = 0.9372837408557921
```

AUC score is 0.9372 This evaluates our model is pretty good at predicting the label of the news as satisfied or unsatisfied/neutral but least performance than all other models except naive bayes

d. Classification Report:

Precision, recall, f1 scores are shown in the screenshot.

	precision	recall	f1-score	support
0	0.93053	0.96622	0.94804	17674
1	0.95489	0.90835	0.93103	13911
accuracy			0.94073	31585
macro avg	0.94271	0.93728	0.93954	31585
weighted avg	0.94125	0.94073	0.94055	31585

Fig. 1.31 Screenshot of code output for classification report

In the above report, we got Precision as 0.9305 to predict satisfaction and 0.9548 to predict unsatisfied or neutral satisfaction. Recall is 0.9662 for satisfaction and 0.9083 for unsatisfied or neutral satisfaction. Weighted average f1-score is 0.9480 and 0.9310 respectively.

Overall, based on the evaluation metrics, the SVM also appears to perform better than the naive bayes and XGBooster but almost having equal performance in comparison with all other models above.

The ability of SVMs to handle non-linear correlations between the characteristics and the target variable makes them useful for datasets on airline satisfaction. SVMs can also handle high-dimensional data, which is typical of datasets on passenger satisfaction in airlines where there may be a variety of factors that affect passenger satisfaction.

Conclusion:

Based on the accuracy metrics of various models, it is apparent that we have the ability to create a predictive system capable of measuring customer satisfaction and integrating it into their business operations. Our exploratory data analysis has yielded results that can assist airline businesses in implementing measures that enhance customer experience. Our system has demonstrated a high level of accuracy, achieving a predictive accuracy rate of 95%. Below are the accuracies of different model used in this project.

MODEL	**Accuracy**
Logistic regression	92.12

KNN	94.91
Random Forest	95.03
XGB	85.7
SVM	87.57
Naive Bayes	94.073

Phase 3:

In this phase, we use the models that were trained in phase 2 to predict if a customer is satisfied or not in the real-world in a real time. As part of phase-2 we have done Random Forest Classifier, Linear Support Vector Machine, Logistic regression, Naive bayes, XGBoost models have performed well to predict if a customer is satisfied or not for a provided service. We have chosen the Logistic Regression as default model to predict the services by the company in the real-world and let's also provide the flexibility of selecting a particular trained model to the user when he provides the real world satisfaction levels like rating for each service.

In order to predict the user input, we need to use the trained model, instead of running the entire code of phase 2 every time. This reduces the running time as we are training only required model

So, if we export the trained vectorizer to a pickle file, we can load the same file as our model in the code on run time of the data product to predict the given input.

we need to store all the trained models, to load them in a data product server. Since we have an option of selecting the model in the front end screen, instead of loading the entire models as a pickle file, we are loading only the selected model into our code on run time. Below is the code to perform the above operation:
Sample screenshot of how front end looks like for a user to select models on their need

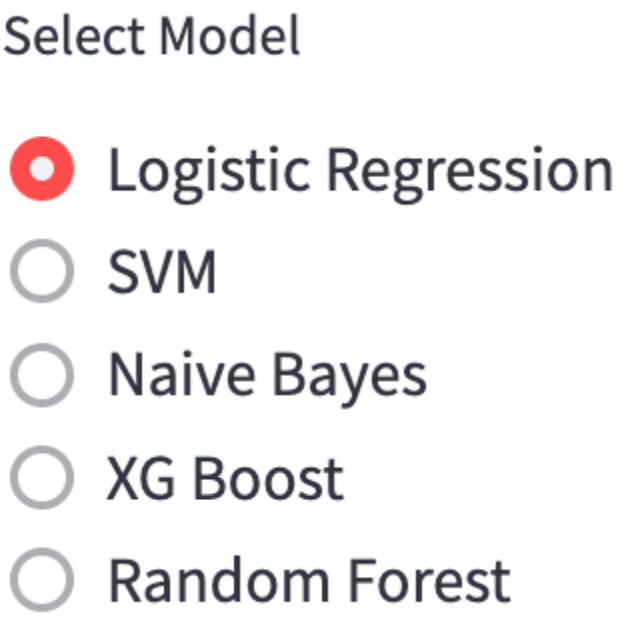


Fig 4.1User selection of model

Based on selected model, we will import that corresponding model into our code on run time

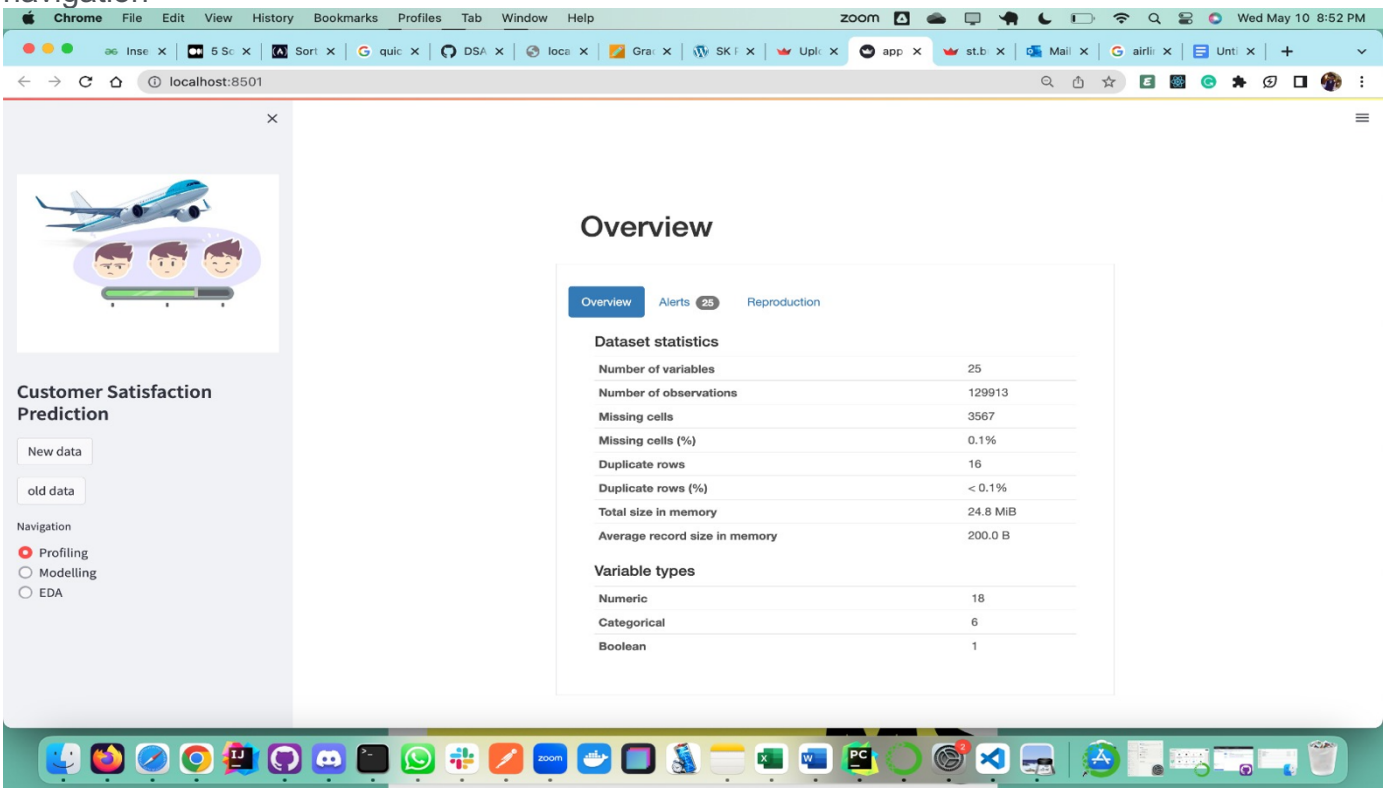
```
def chooseModel(modelName):
    loaded_model = pickle.load(open(modelName+'.pkl','rb'))
    return loaded_model
```

To predicted the satisfaction level in the real-time, we have developed a web application in python and rendered the UI using HTML and CSS
The application contains three main pages on the left side with a radio buttons

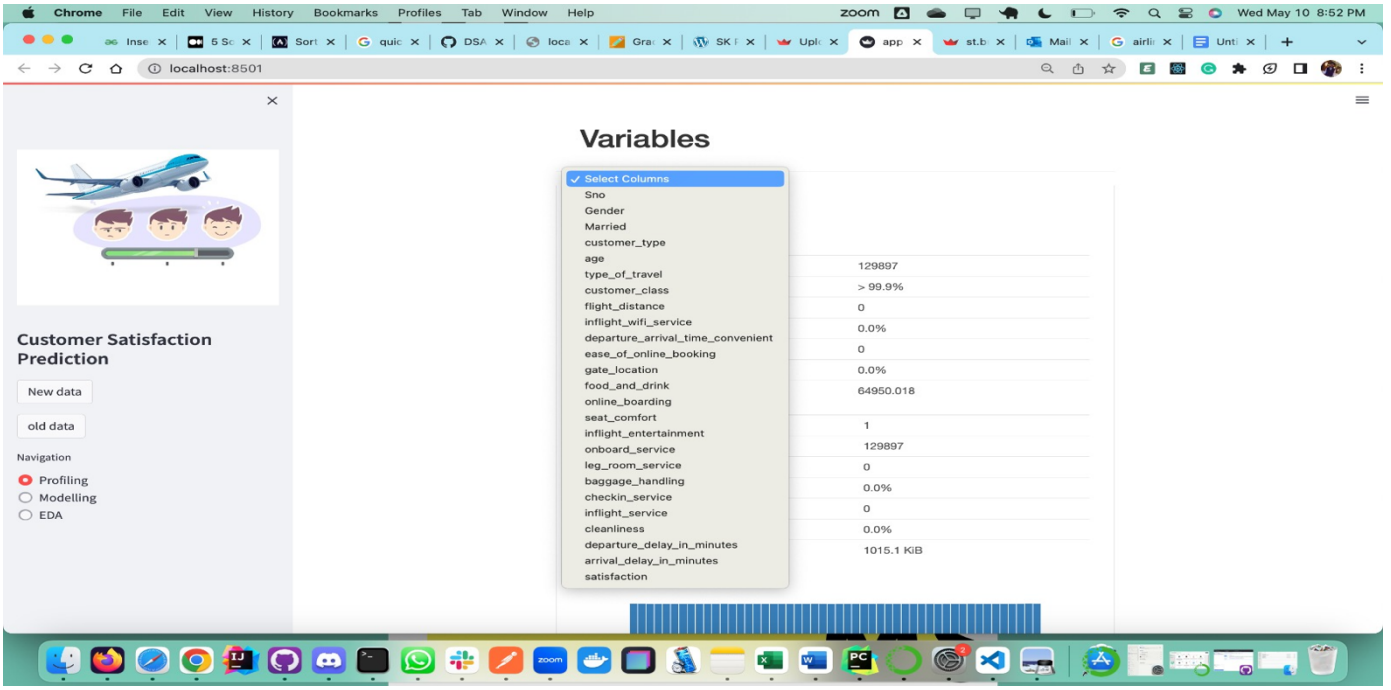
1. Profiling
2. Modeling
3. EDA

Profiling: In this page we are examining, analyzing, and creating useful summaries of data based on the dataset we have uploaded, this includes mean, number of missing values, number of zeros, memory size this feature occupies and number of distinct values in the entire dataset. We have also given an option to the user to select if he needs a only particular feature to analyze and examine. Below screenshot depicts how to use profiling tab in the UI

Below figure shows the analysis of the entire dataset upon click of Profiling button in navigation

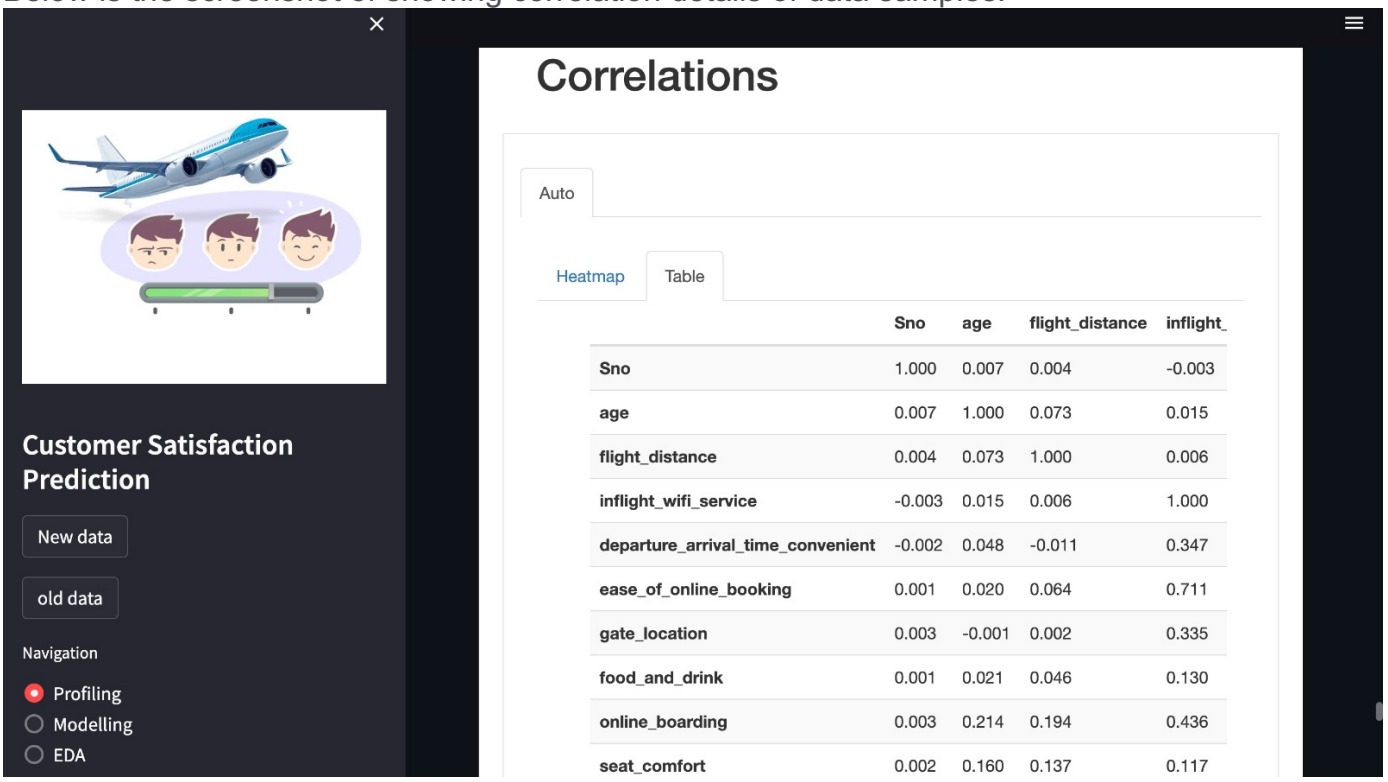


We can analyse the individual feature by selecting the required column in drop down as shown below:



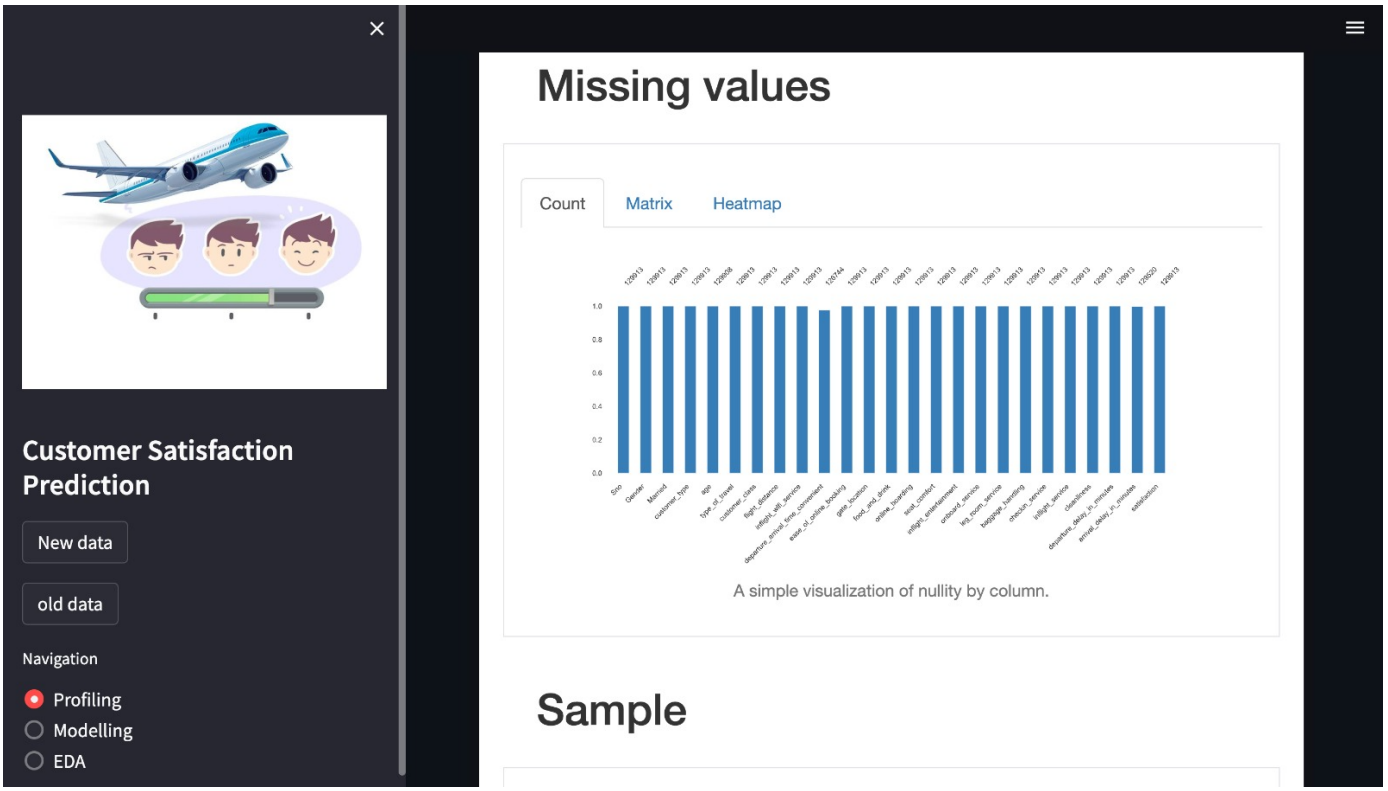
By doing so, will display results only to the particular column

Finally, we also show the correlation details of the entire features, Missing values , sample view of dataset for first and last 10 rows and their duplicate details in dataset provided.
Below is the screenshot of showing correlation details of data samples:

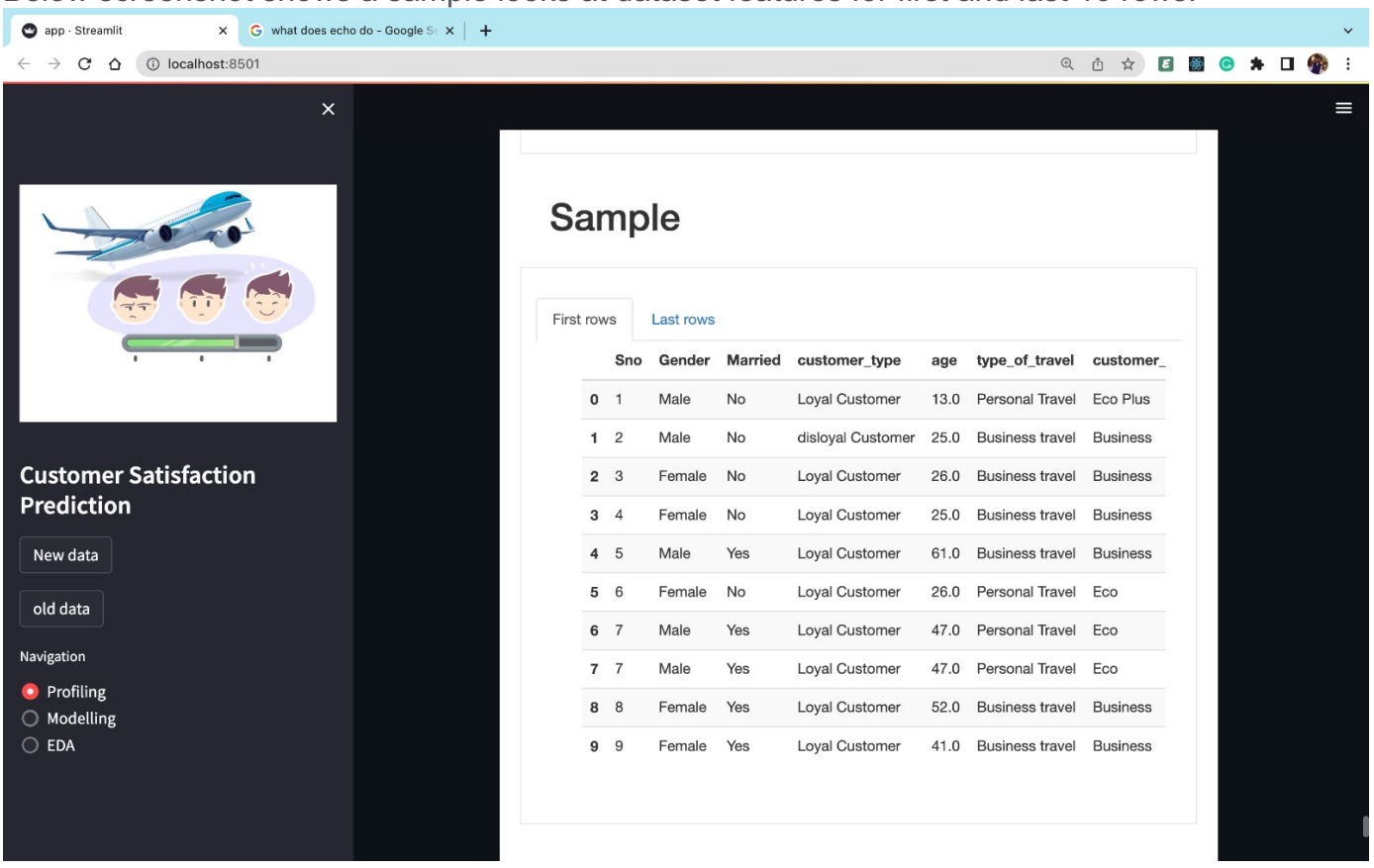


	Sno	age	flight_distance	inflight_
Sno	1.000	0.007	0.004	-0.003
age	0.007	1.000	0.073	0.015
flight_distance	0.004	0.073	1.000	0.006
inflight_wifi_service	-0.003	0.015	0.006	1.000
departure_arrival_time_convenient	-0.002	0.048	-0.011	0.347
ease_of_online_booking	0.001	0.020	0.064	0.711
gate_location	0.003	-0.001	0.002	0.335
food_and_drink	0.001	0.021	0.046	0.130
online_boarding	0.003	0.214	0.194	0.436
seat_comfort	0.002	0.160	0.137	0.117

Below screenshot shows missing values in the features in provided dataset

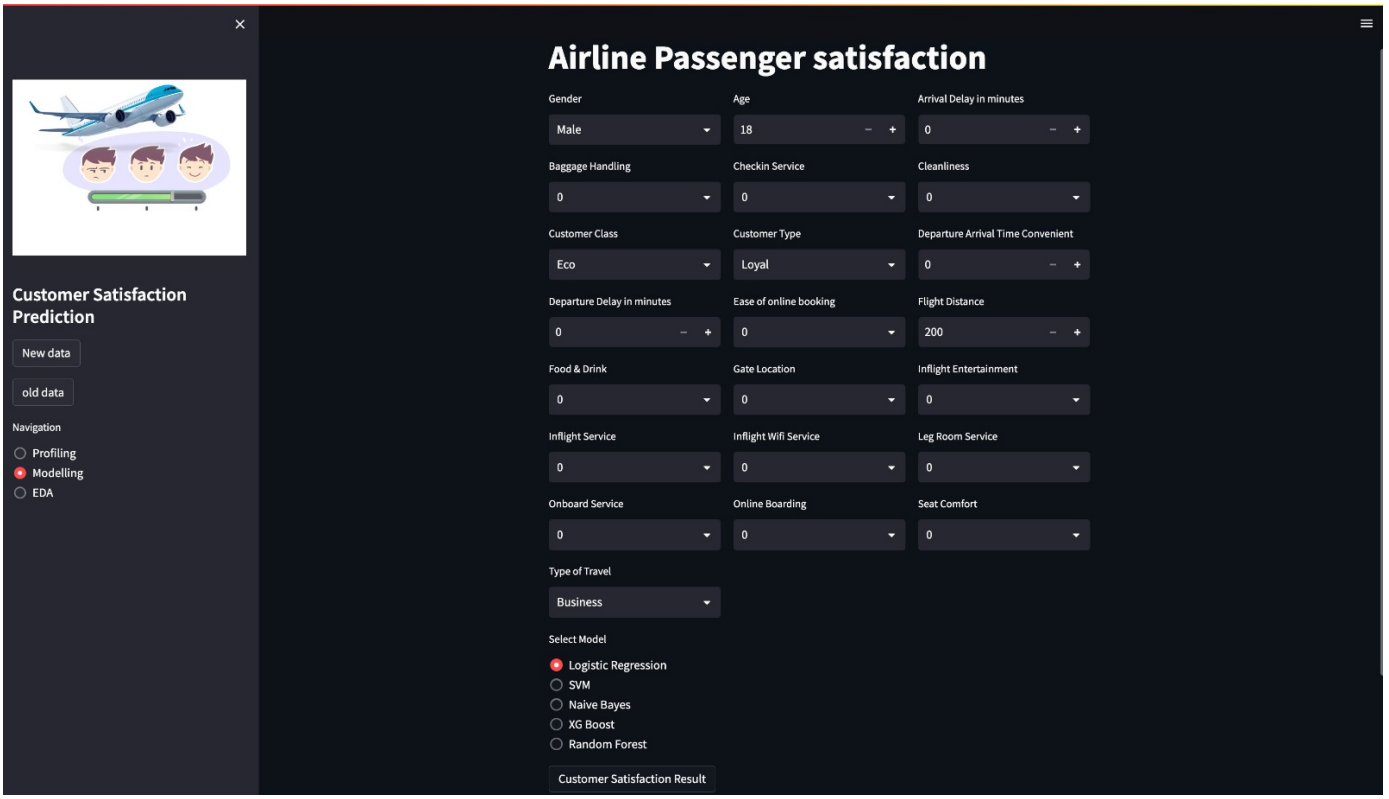


Below screenshot shows a sample looks at dataset features for first and last 10 rows:



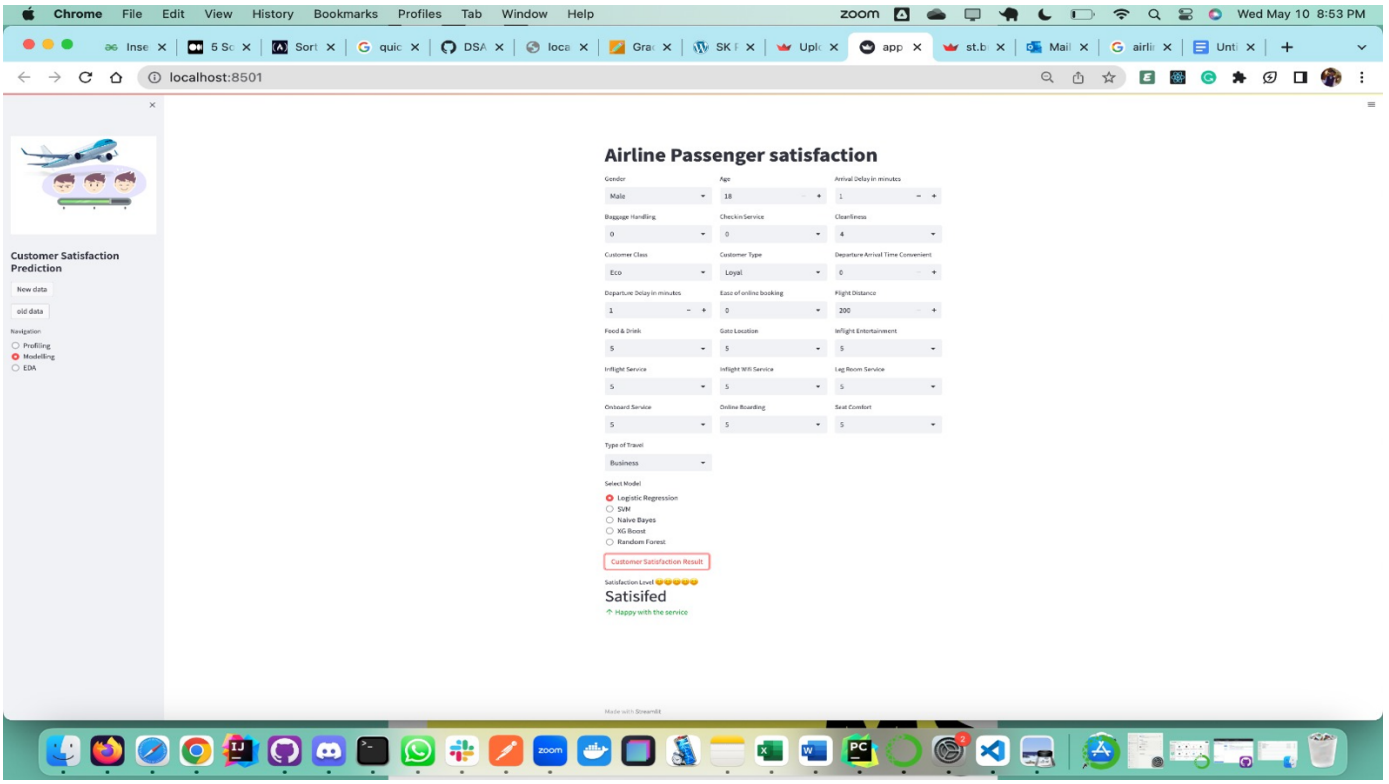
	Sno	Gender	Married	customer_type	age	type_of_travel	customer_
0	1	Male	No	Loyal Customer	13.0	Personal Travel	Eco Plus
1	2	Male	No	disloyal Customer	25.0	Business travel	Business
2	3	Female	No	Loyal Customer	26.0	Business travel	Business
3	4	Female	No	Loyal Customer	25.0	Business travel	Business
4	5	Male	Yes	Loyal Customer	61.0	Business travel	Business
5	6	Female	No	Loyal Customer	26.0	Personal Travel	Eco
6	7	Male	Yes	Loyal Customer	47.0	Personal Travel	Eco
7	7	Male	Yes	Loyal Customer	47.0	Personal Travel	Eco
8	8	Female	Yes	Loyal Customer	52.0	Business travel	Business
9	9	Female	Yes	Loyal Customer	41.0	Business travel	Business

Modeling: In this page, we are training a machine learning algorithm to predict the labels from the features, tuning it for the business need, and validating it on holdout data. As part of it we have designed an UI for a user where he can enter satisfaction levels of each service they are providing as a form of input and predict if the customer is satisfied or not overall. We have given the flexibility to an user to select which model he want to apply, final test result will be displayed based on the training that data model
Below is the screenshot from UI where user can enter the satisfaction levels:



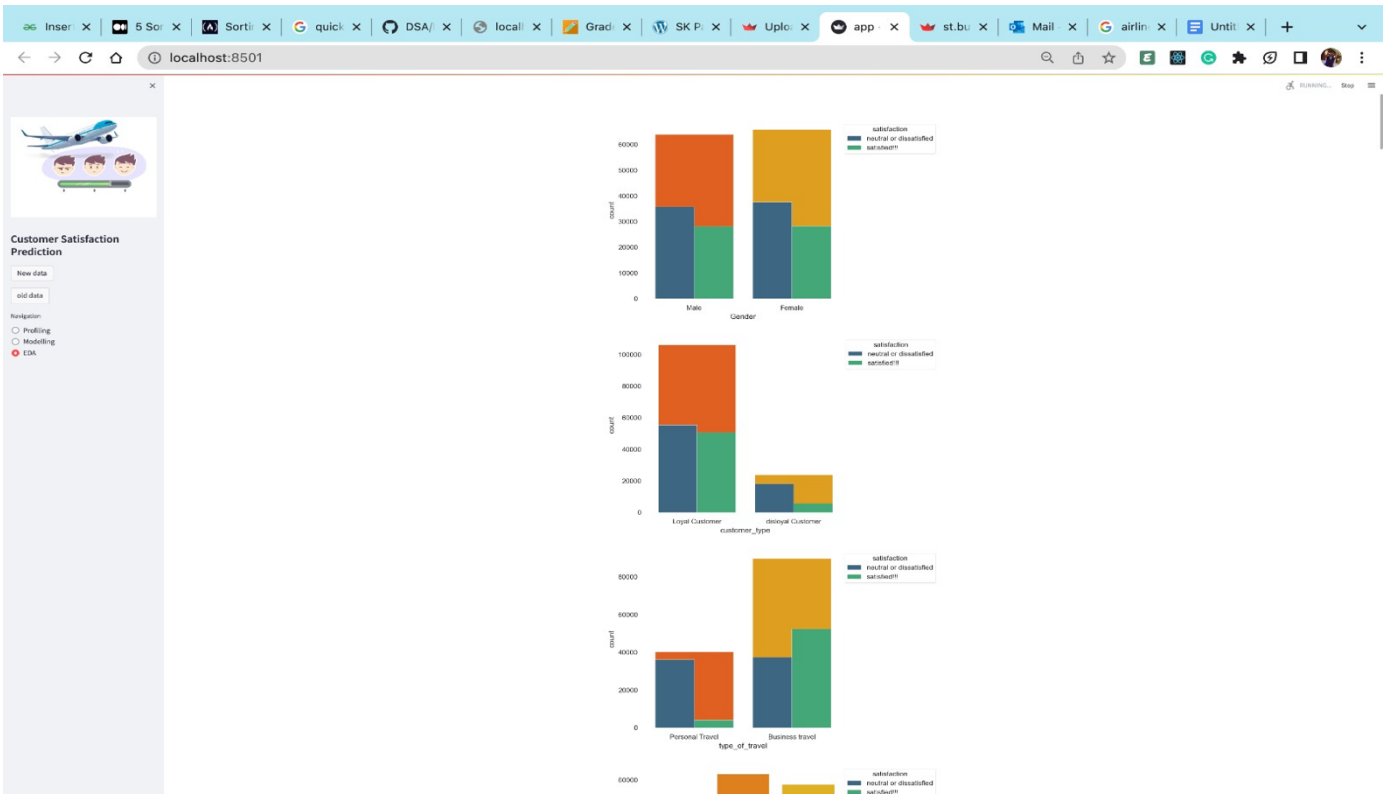
Upon entering all inputs, user has to select which model he want to apply, and click on the customer satisfaction result button to view if the custom is satisfied or not

Below is the screenshot that shows how the overall result looks like:



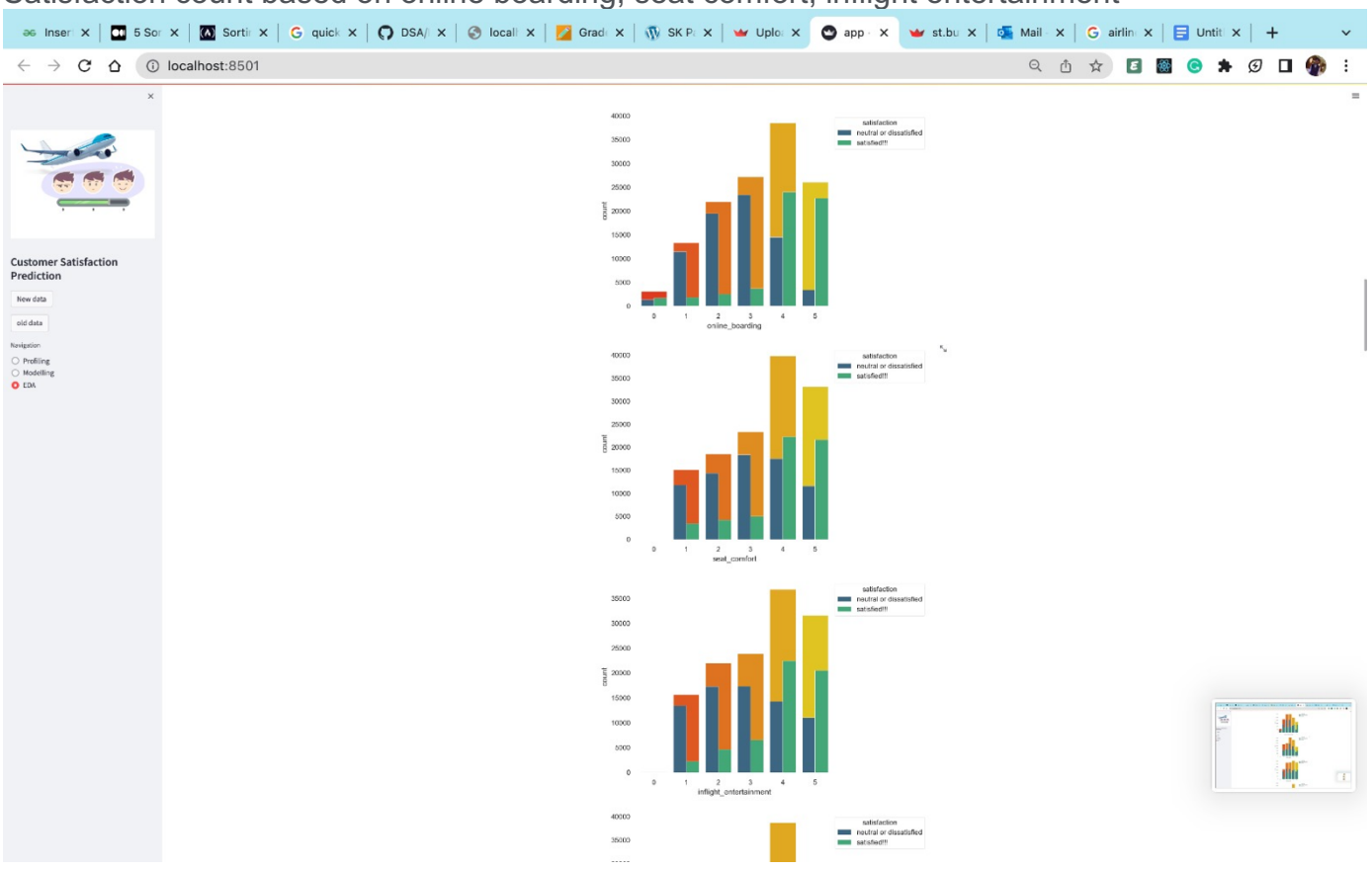
EDA: In this page, we analyze and investigate our dataset and summarize their main characteristics, by employing data visualization methods.
This page specifically helps the busines to tackle the challenges they are facing and to take necessary steps to tackle them. It shows what effect each feature haver on customer satisfaction. We have plotted graphs for each feature and investigated their satisfaction for each feature. We have also plotted the box plots to check outliers of each feature in our dataset.Below are some screenshot of graphs:
Below screenshots feature wise satisfaction as part of EDA:

Satisfaction count based on gender, customer type, travel type



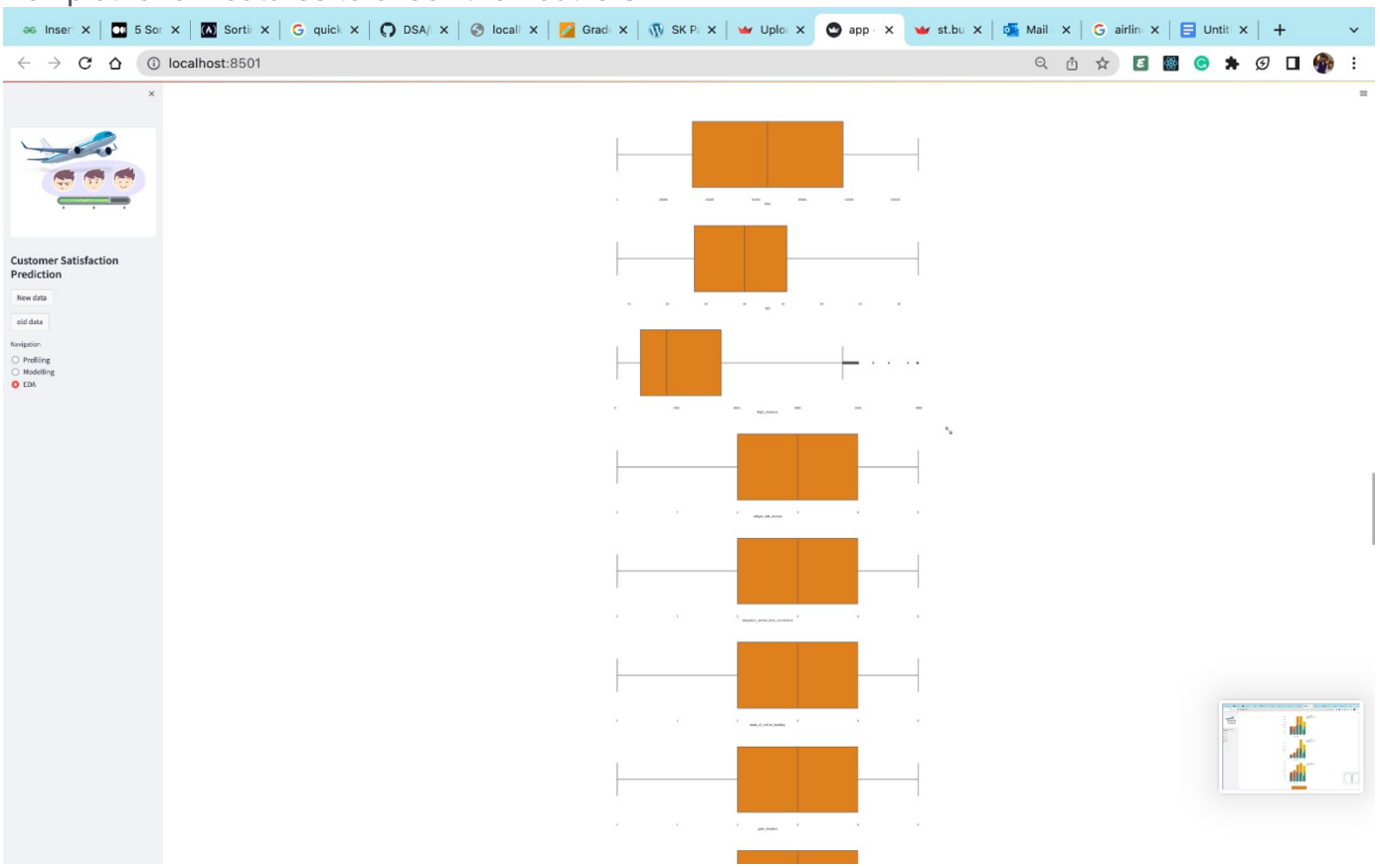
Satisfaction count based on customer class, wifi service, departure arrival time

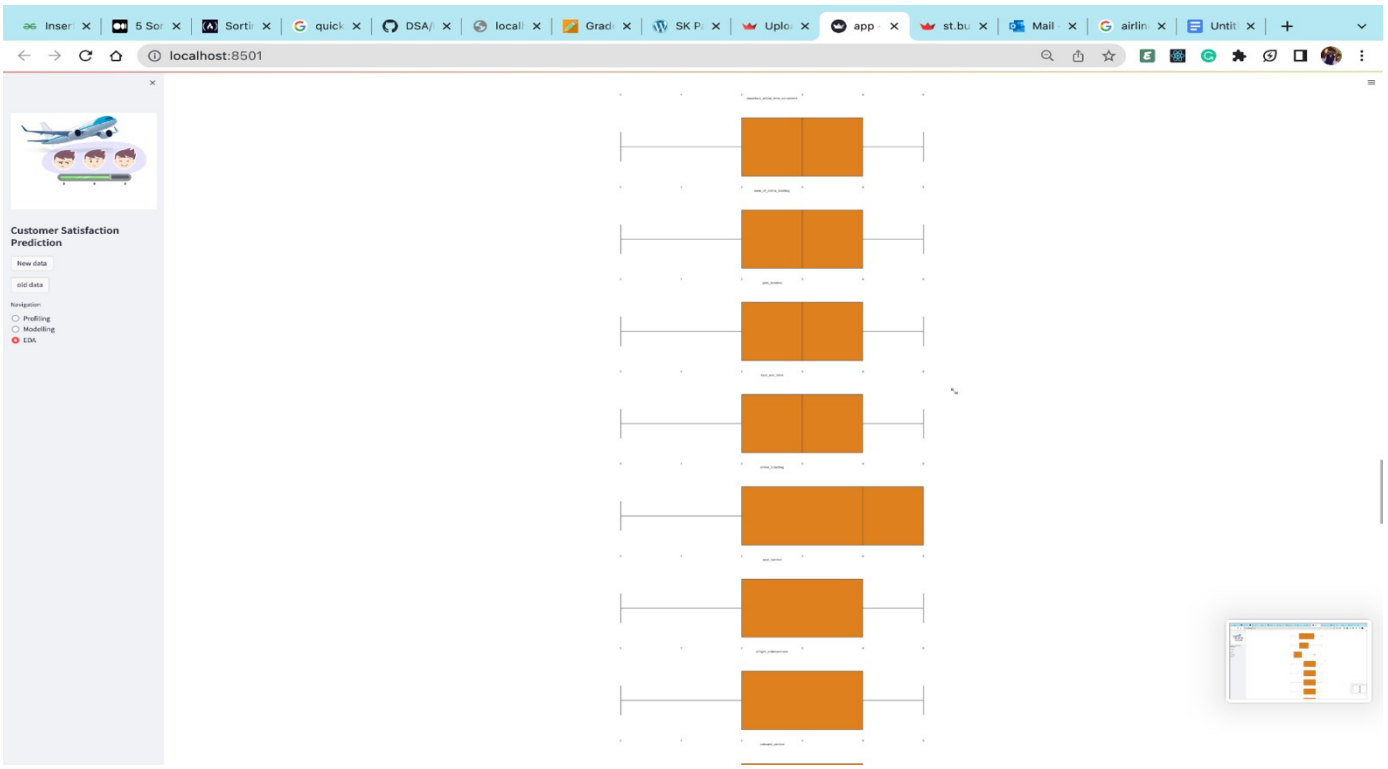
Satisfaction count based on online boarding, seat comfort, inflight entertainment



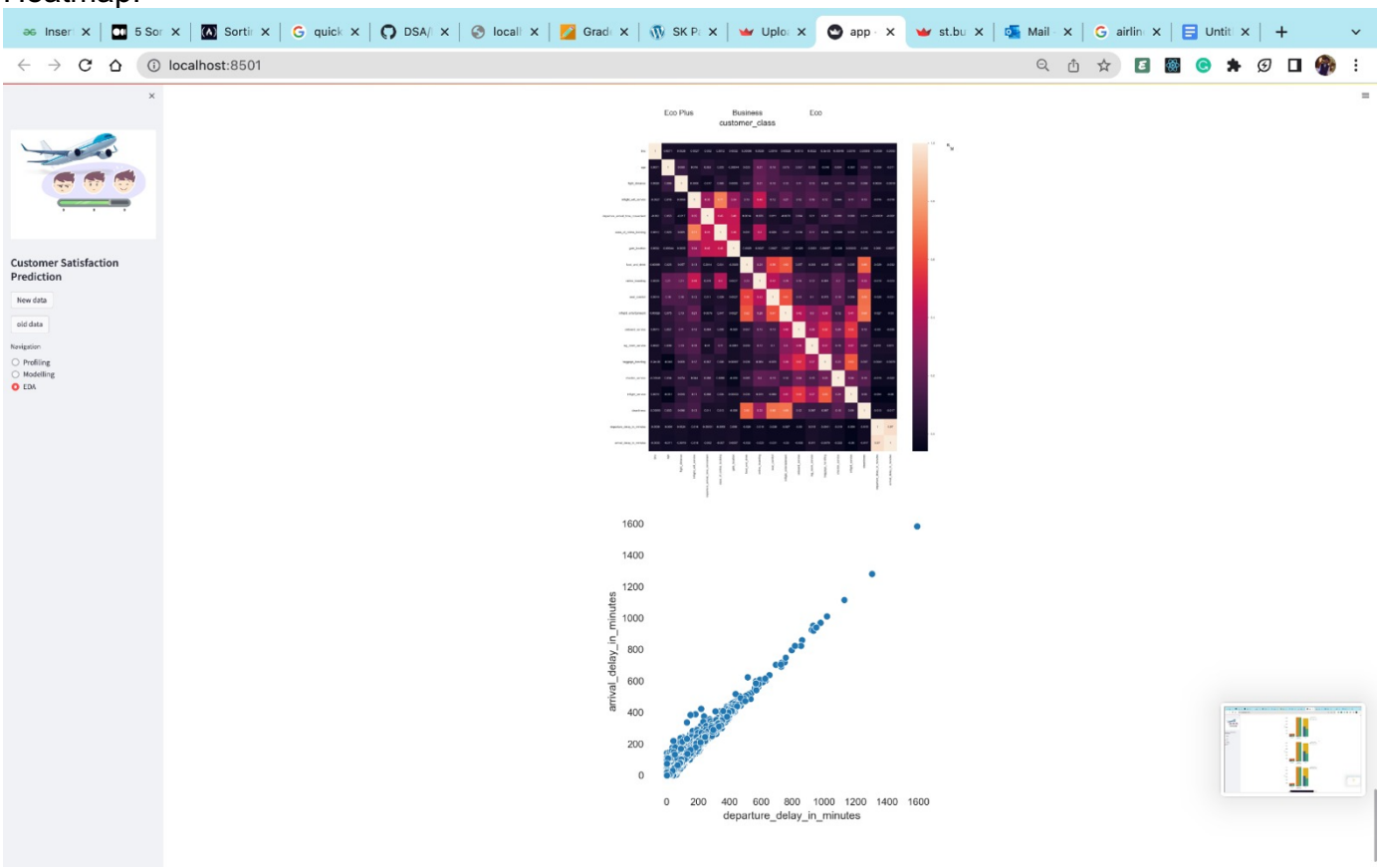
Satisfaction count based on checkIn service, Inflight service, cleanliness

Box plot for all features to check their outliers




localhost:8501
Customer Satisfaction Prediction
New data
old data
Navigation
Profiling
Modelling
EDA

Heatmap:


localhost:8501
Customer Satisfaction Prediction
New data
old data
Navigation
Profiling
Modelling
EDA
Eco Plus
Business
Eco
customer_class
arrival_delay_in_minutes
departure_delay_in_minutes
0
200
400
600
800
1000
1200
1400
1600

Conclusion:

In conclusion, the front-end screen we created in our project helps users to add input samples in easier manner. This UI interface makes it easier for those who may not have a technical background to utilize by streamlining the data input process and enhancing user experience. In order to ensure that the data is precisely gathered and processed, the front-end screen is also made to integrate easily with the back-end data processing system. Overall, this initiative has the potential to streamline the data collection and analysis process, providing businesses and organizations with more precise insights and better decision-making.

References:

[1] Binary J. (2021 January). Airline Passenger Satisfaction Version 2. Febrauary 2023 from https://www.kaggle.com/datasets/binaryjoker/airline-passenger-satisfaction

[2] Bureau of Transportation Statistics. (2020). Airline On-Time Performance Data. Retrieved from https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp

[3] Python development guide : https://seaborn.pydata.org/generated/seaborn.scatterplot.html

[4] Pandas documentation: https://pandas.pydata.org/docs/user_guide/index.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[6] https://seaborn.pydata.org/tutorial.html

[7] https://scikit-learn.org/stable/

[8] https://plotly.com/python/

[9] The Hundred page machine learning Book by Andriy Burkov :

[10] http://ema.cri-info.cm/wp-content/uploads/2019/07/2019BurkovTheHundred-pageMachineLearning.pdf

[11] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[12] https://scikit-learn.org/stable/modules/naive_bayes.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[14] https://docs.streamlit.io/