Fundamentals of Data Mining – IT3051

Airline Passenger Satisfaction Prediction

Group Number: 37

Group Members:

Name	Registration Number
Rupasinghe J.D	IT20024772
Ariyasinghe P.A.D.N.I	IT20033828
Dahanayake U.S	IT20043650

Video Presentation Link: https://drive.google.com/file/d/1fcXhl7jtxPQqqBItooi_bg9zZkQTp-ta/view?usp=sharing

Table of Contents

1. Background

The service sector has surpassed manufacturing as the most important sector of the global economy. Several studies have been conducted to improve service efficiency and to identify the factors that influence customer satisfaction and loyalty in various industries.

Customer satisfaction occurs in the airline industry when passengers' expectations are met by the airline experience. They will be content and think the service quality is very good if it meets their expectations, but if it exceeds their expectations, customers will be very delighted and think the service quality is excellent. As a result, increasing service efficiency is heavily dependent on the airline's ability to consistently meet the demands and desires of passengers.

The dataset used in our project contains an airline passenger satisfaction survey responses as well as passenger and flight information from US airlines. This dataset contains nearly 130,000 survey responses. Including details like gender, age, customer type, seat comfort, inflight services, etc. Fourteen of the attributes are based on survey responses, where passengers rate their flight experiences from 1 to 5.

In this project, we have two goals. The initial goal is to build a classification model that predicts the satisfaction of a customer with the services provided by an airline. The second goal is to analyze airline customer data to discover the relationship between various factors influencing customer satisfaction.

Identified Problem – The airline company requires a system to identify whether or not the customer had an overall satisfactory experience with their services based on the customer's survey responses. They also need to identify possible relationships between services and customer satisfaction.

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

2. Description about the Original Datasets

The chosen dataset contains 129,878 customer ratings for the services provided by an airline. Each row corresponds to survey data from a single passenger. Passengers get to rate 14 services and at the end state the overall satisfaction.
The dataset was already divided into train and test csv files. 80% of the total dataset is in training.csv and 20% are in testing.csv.

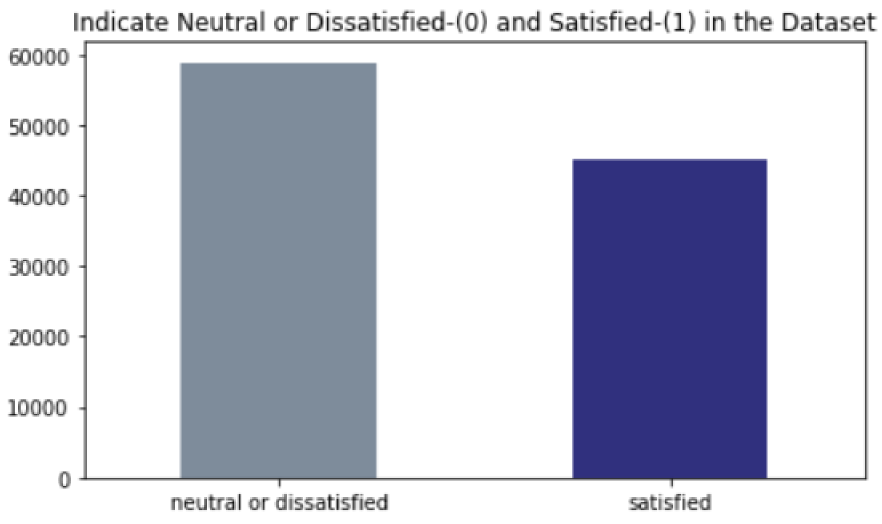
Column Name	**Description**	**Data Type**
Airport Satisfaction ID	Airport Satisfaction Unique ID	Int
Gender	Gender of the passengers (Female, Male)	String
Customer Type	The Customer type (Loyal Customer /Disloyal Customer)	String
Age	Age of the passenger	Int
Travel Type	Purpose of the flight of the passengers (Personal Travel, Business Travel)	String
Class	Travel class in the plane of the passengers (Business, Eco Plus, Eco)	String
Flight Distance	The flight distance of the journey	Int
Inflight Wi-Fi service	Rating Inflight Wi-Fi service	Int
Departure/Arrival time convenient	Rating Departure/Arrival time convenient	Int
Ease of Online booking	Rating Ease of Online booking	Int
Gate location	Rating Gate location	Int
Food and drink	Rating Food and drink	Int
Online Boarding	Rating Online Boarding	Int
Seat comfort	Rating Seat comfort	Int
Inflight entertainment	Rating Inflight entertainment	Int
On-board service	Rating On-board service	Int
Leg room service	Rating Leg room service	Int
Baggage handling	Rating Baggage handling	Int
Check in service	Rating Check in service	Int
Inflight service	Rating Inflight service	Int
Cleanliness	Rating Cleanliness	Int
Departure Delay in Minutes	Minutes delayed when departure	Int
Arrival Delay in Minutes	Minutes delayed when Arrival	Int
Satisfaction	Airline satisfaction level (Satisfaction, Neutral or Dissatisfaction)	String

3. Data Identification

3.1. Bar Chart

The plot depicts a distribution of neutral/dissatisfied passengers and satisfied passengers. The data appears to be well balanced, with somewhat more neutral or disgruntled customers. There is no need for any extra treatment or resampling.

```
# Checking whether the dataset is balanced or imbalanced
fig = plt.figure(figsize = (7,4))
train.satisfaction.value_counts().plot(kind='bar', color= ['slategrey','midnightblue'], alpha = 0.9, rot=0)
plt.title('Indicate Neutral or Dissatisfied-(0) and Satisfied-(1) in the Dataset')
plt.show()
```

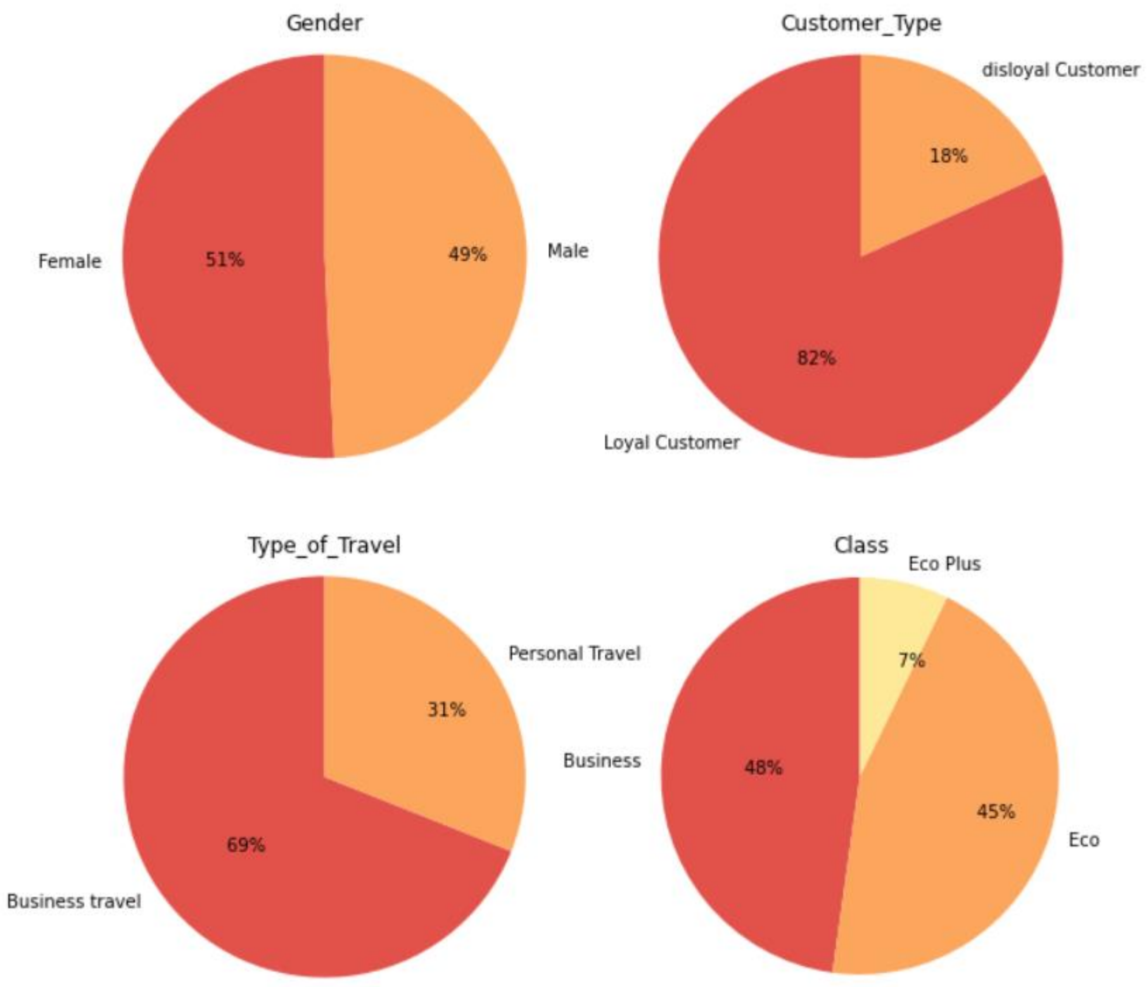


3.2. Pie Charts

This pie chart provides a quick overview of the data set. We can get some information about the data sample.

- The number of men and women in this sample is roughly equal.
- The vast majority of the airline's customers are repeat customers.
- Majority of the passengers flew for business rather than for personal reasons.
- Approximately half of the passengers flew in business class.

```
#display the insight of the data sample
figure = [0, 1, 3, 4]
count = train.iloc[:,figure]
fig, axes = plt.subplots(2, 2, figsize = (10, 10))
for i, columns in enumerate(count):
    column_values = train[columns].value_counts()
    label = column_values.index
    total = column_values.values
    axes[i//2, i%2].pie(total, labels = label, colors = sns.color_palette("Spectral"), autopct = '%1.0f%%', startangle = 90)
    axes[i//2, i%2].axis('equal')
    axes[i//2, i%2].set_title(columns)
plt.show()
```



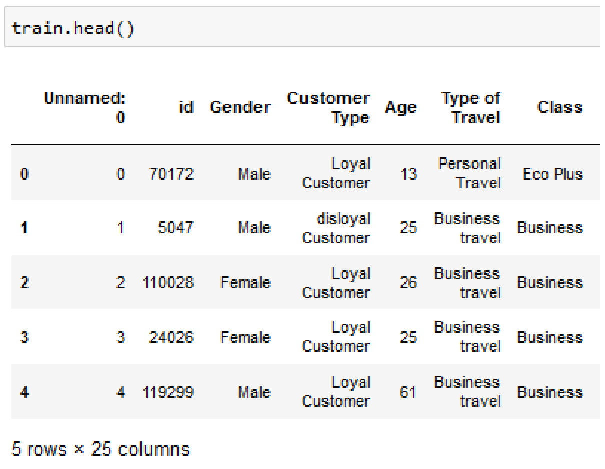
4. Data Preprocessing

4.1. Classification Data Preprocessing

- The classification won't be affected by the first two features, therefore remove the unnecessary columns.

```
# Drop unnecessary columns
train = train.drop('Unnamed: 0', axis=1)
train = train.drop('id', axis=1)

test = test.drop('Unnamed: 0', axis=1)
test = test.drop('id', axis=1)
```

- Before removing columns

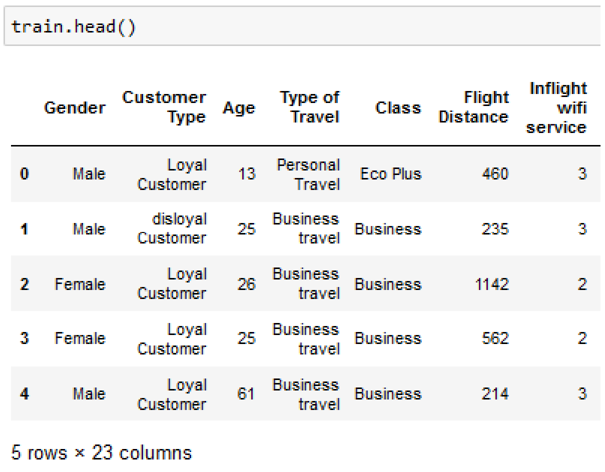


```
train.head()
```

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus
1	1	5047	Male	disloyal Customer	25	Business travel	Business
2	2	110028	Female	Loyal Customer	26	Business travel	Business
3	3	24026	Female	Loyal Customer	25	Business travel	Business
4	4	119299	Male	Loyal Customer	61	Business travel	Business

5 rows × 25 columns

- After removing columns



```
train.head()
```

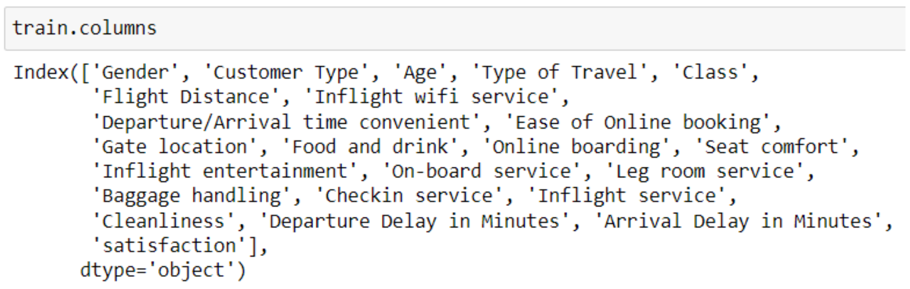
	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3
1	Male	disloyal Customer	25	Business travel	Business	235	3
2	Female	Loyal Customer	26	Business travel	Business	1142	2
3	Female	Loyal Customer	25	Business travel	Business	562	2
4	Male	Loyal Customer	61	Business travel	Business	214	3

5 rows × 23 columns

- Use underscore to replace spaces between column names.

```
# Replace spaces in the column names with underscore
train.columns = [c.replace(' ', '_') for c in train.columns]
test.columns = [c.replace(' ', '_') for c in test.columns]
```

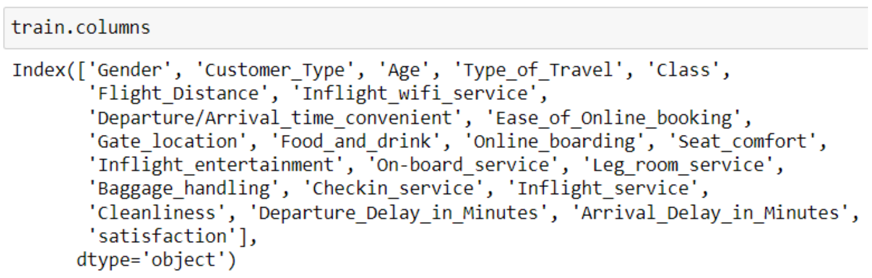
- Before replacing spaces



```
train.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')
```

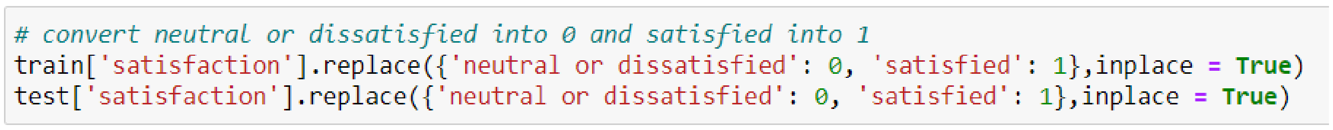
- After replacing spaces



```
train.columns
```

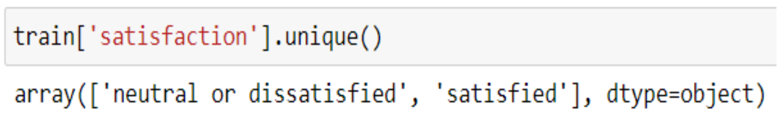
```
Index(['Gender', 'Customer_Type', 'Age', 'Type_of_Travel', 'Class',
       'Flight_Distance', 'Inflight_wifi_service',
       'Departure/Arrival_time_convenient', 'Ease_of_Online_booking',
       'Gate_location', 'Food_and_drink', 'Online_boarding', 'Seat_comfort',
       'Inflight_entertainment', '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 the satisfaction column with, satisfied: 1, neutral or dissatisfied: 0.



```
# convert neutral or dissatisfied into 0 and satisfied into 1
train['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1},inplace = True)
test['satisfaction'].replace({'neutral or dissatisfied': 0, 'satisfied': 1},inplace = True)
```

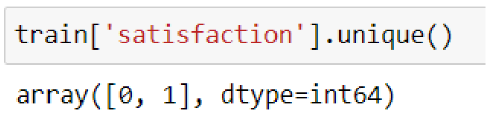
- Before replacing



```
train['satisfaction'].unique()
```

```
array(['neutral or dissatisfied', 'satisfied'], dtype=object)
```

- After replacing

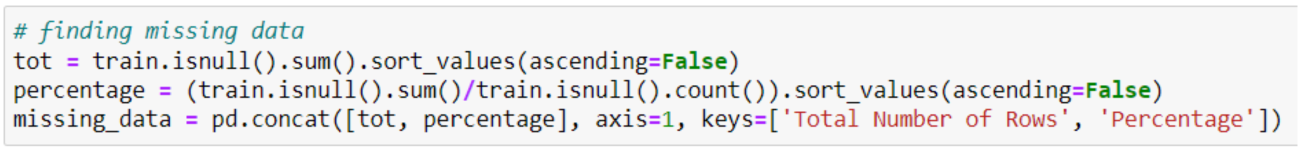


```
train['satisfaction'].unique()
```

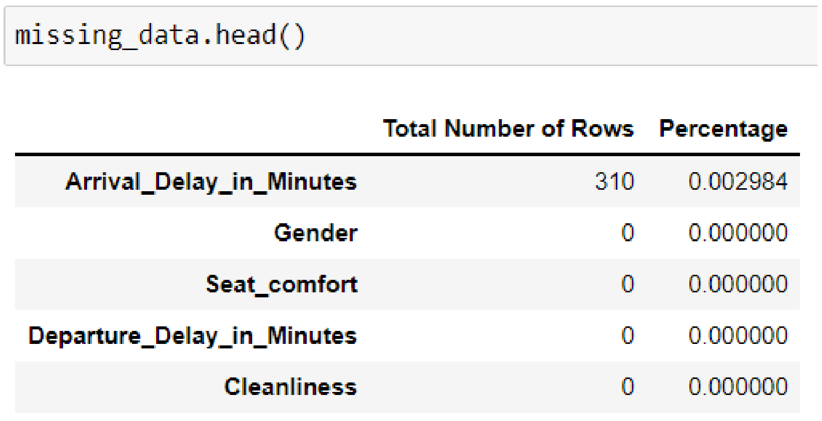
```
array([0, 1], dtype=int64)
```

- Finding the missing values in the data set.

  There are 393 missing values found in the Arrival_Delay_in_Minutes.



```
# finding missing data
tot = train.isnull().sum().sort_values(ascending=False)
percentage = (train.isnull().sum()/train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([tot, percentage], axis=1, keys=['Total Number of Rows', 'Percentage'])
```



```
missing_data.head()
```

	Total Number of Rows	Percentage
Arrival_Delay_in_Minutes	310	0.002984
Gender	0	0.000000
Seat_comfort	0	0.000000
Departure_Delay_in_Minutes	0	0.000000
Cleanliness	0	0.000000

- Fill up the missing values in the columns corresponding to measurements with means.

```
# filling missing values with the mean value
train['Arrival_Delay_in_Minutes'] = train['Arrival_Delay_in_Minutes'].fillna(train['Arrival_Delay_in_Minutes'].mean())

test['Arrival_Delay_in_Minutes'] = test['Arrival_Delay_in_Minutes'].fillna(test['Arrival_Delay_in_Minutes'].mean())
```

- Fill up the missing values in the categorical columns with 0.

```
# filling missing values of categorical data
train.select_dtypes(include=['object']).columns

train['Gender'] = train['Gender'].fillna(train['Gender'].mode()[0])
train['Customer_Type'] = train['Customer_Type'].fillna(train['Customer_Type'].mode()[0])
train['Type_of_Travel'] = train['Type_of_Travel'].fillna(train['Type_of_Travel'].mode()[0])
train['Class'] = train['Class'].fillna(train['Class'].mode()[0])

test['Gender'] = test['Gender'].fillna(test['Gender'].mode()[0])
test['Customer_Type'] = test['Customer_Type'].fillna(test['Customer_Type'].mode()[0])
test['Type_of_Travel'] = test['Type_of_Travel'].fillna(test['Type_of_Travel'].mode()[0])
test['Class'] = test['Class'].fillna(test['Class'].mode()[0])
```

- Transforming categorical information into numerical information.
  - Data transformation in the Gender with, Male: 1, Female: 0.
  - Data transformation in the Customer Type with, Loyal Customer: 0, Disloyal Customer: 1.
  - Data transformation in the Type of Travel with, Personal Travel: 1, Business Travel: 0.
  - Data transformation in the Class with, Eco Plus: 2, Business: 0, Eco: 1.

```
# encoding categorical variables
lencoders = {}
for c in train.select_dtypes(include=['object']).columns:
    lencoders[c] = LabelEncoder()
    train[c] = lencoders[c].fit_transform(train[c])

lencoders_t = {}
for c in test.select_dtypes(include=['object']).columns:
    lencoders_t[c] = LabelEncoder()
    test[c] = lencoders_t[c].fit_transform(test[c])
```

- Before encoding

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

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

```
train['Customer_Type'].unique()
```

```
array(['Loyal Customer', 'disloyal Customer'], dtype=object)
```

```
train['Type_of_Travel'].unique()
```

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

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

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

- After encoding

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

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

```
train['Customer_Type'].unique()
```

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

```
train['Type_of_Travel'].unique()
```

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

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

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

- Detection and Removal of Outliers.

```
# finding outliers
Quantile1 = train.quantile(0.25)
Quantile3 = train.quantile(0.75)
IQR = Quantile3 - Quantile1
```

```
# removing outliers
train = train[~((train < (Quantile1 - 1.5 * IQR)) |(train > (Quantile3 + 1.5 * IQR))).any(axis=1)]
```

- Before removing outliers

```
train.shape
```

```
(103904, 23)
```

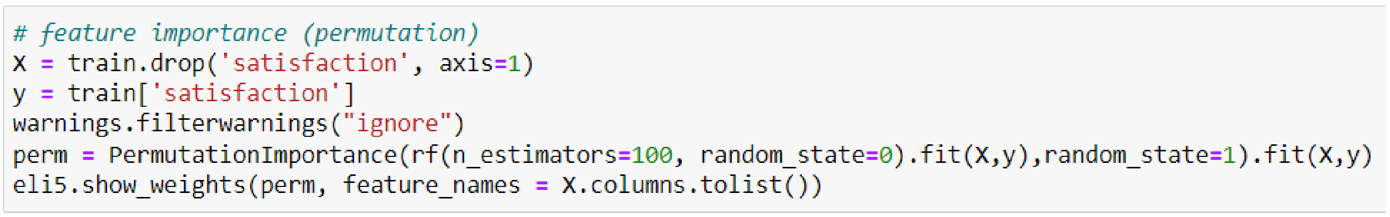
- After removing outliers

```
train.shape
```

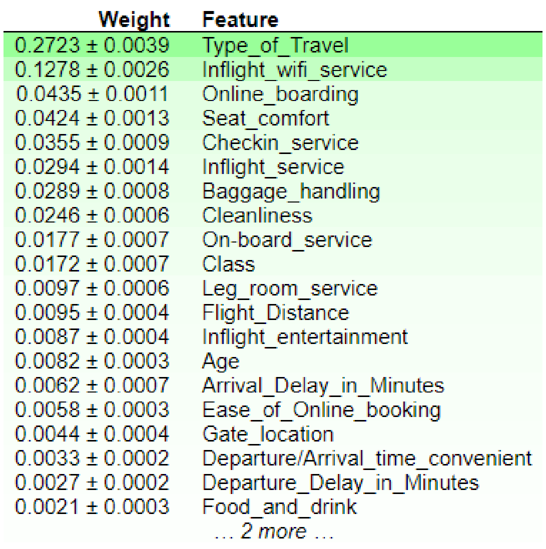
```
(61197, 23)
```

(When comparing, 42,707 outliers were removed.)

- Finding the top five features that have the highest impact on passenger satisfaction.



```python
# feature importance (permutation)
X = train.drop('satisfaction', axis=1)
y = train['satisfaction']
warnings.filterwarnings("ignore")
perm = PermutationImportance(rf(n_estimators=100, random_state=0).fit(X,y),random_state=1).fit(X,y)
eli5.show_weights(perm, feature_names = X.columns.tolist())
```



Weight	Feature
0.2723 ± 0.0039	Type_of_Travel
0.1278 ± 0.0026	Inflight_wifi_service
0.0435 ± 0.0011	Online_boarding
0.0424 ± 0.0013	Seat_comfort
0.0355 ± 0.0009	Checkin_service
0.0294 ± 0.0014	Inflight_service
0.0289 ± 0.0008	Baggage_handling
0.0246 ± 0.0006	Cleanliness
0.0177 ± 0.0007	On-board_service
0.0172 ± 0.0007	Class
0.0097 ± 0.0006	Leg_room_service
0.0095 ± 0.0004	Flight_Distance
0.0087 ± 0.0004	Inflight_entertainment
0.0082 ± 0.0003	Age
0.0062 ± 0.0007	Arrival_Delay_in_Minutes
0.0058 ± 0.0003	Ease_of_Online_booking
0.0044 ± 0.0004	Gate_location
0.0033 ± 0.0002	Departure/Arrival_time_convenient
0.0027 ± 0.0002	Departure_Delay_in_Minutes
0.0021 ± 0.0003	Food_and_drink
	… 2 more …

The above results allow us to get a list of the five most important features.

- Type_of_Travel
- Inflight_wifi_service
- Online_boarding
- Seat_comfort
- Checkin_service

Then, for the derived features, we perform classification to predict customer satisfaction

4.2. Clustering Data Preprocessing

- Drop unnecessary columns.

```python
# drop unnecessary columns
train.drop(['Unnamed: 0', 'id'], axis=1, inplace=True)
```

- Convert categorical values in the satisfaction column into numerical values.

```python
# a function to convert neutral or dissatisfied into 0 and satisfied into 1 else into -1
def satisfaction_Transform(s):
    if s == 'satisfied':
        return 1
    elif s == 'neutral or dissatisfied':
        return 0
    else:
        return -1

# calling function
train['satisfaction'] = train['satisfaction'].apply(satisfaction_Transform)
```

- Find columns which contains null values.

```python
# find columns with null values
train.isnull().sum().sort_values(ascending=False)
```

```
Arrival Delay in Minutes             310
Gender                                 0
Seat comfort                           0
Departure Delay in Minutes             0
Cleanliness                            0
Inflight service                       0
Checkin service                        0
```

- Fill null values with mean values.

```python
# filling null values in 'Arrival Delay in Minutes' column with mean value
train['Arrival Delay in Minutes'] = train['Arrival Delay in Minutes'].fillna(train['Arrival Delay in Minutes'].mean())
```

- Replace categorical variables with dummy values

```python
# replace categorical data with numbers usin pandas library 'get_dummies'
categorical_cols = ['Gender', 'Customer Type', 'Type of Travel', 'Class']
new_train = pd.get_dummies(train, columns=categorical_cols, drop_first=True)
train_df = new_train.copy()
```

```python
train_df.columns
```

```
Index(['Age', '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', 'Gender_Male', 'Customer Type_disloyal Customer',
       'Type of Travel_Personal Travel', 'Class_Eco', 'Class_Eco Plus'],
      dtype='object')
```

- The categorical columns have been replaced by the dummy columns with numerical values as shown below.

```
tab = train_df[["Gender_Male", "Customer Type_disloyal Customer","Type of Travel_Personal Travel","Class_Eco","Class_Eco Plus"]]
tab.head()
```

	Gender_Male	Customer Type_disloyal Customer	Type of Travel_Personal Travel	Class_Eco	Class_Eco Plus
0	1	0	1	0	1
1	1	1	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	1	0	0	0	0

- Scale continuous columns.

```
# scaling columns with 'StandardScaler()' to give all features same importance
train_cols = train_df[['Age', 'Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']]
scaler = StandardScaler()
scaled_train = pd.DataFrame(scaler.fit_transform(train_cols), columns = train_cols.columns)

x_cols = ['Age', 'Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']
train_df[x_cols] = scaled_train
```

- Before scaling.

```
tab = train_df[['Age', 'Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']]
tab.head()
```

	Age	Flight Distance	Departure Delay in Minutes	Arrival Delay in Minutes
0	13	460	25	18.0
1	25	235	1	6.0
2	26	1142	0	0.0
3	25	562	11	9.0
4	61	214	0	0.0

- After scaling.

```
tab = train_df[['Age', 'Flight Distance', 'Departure Delay in Minutes', 'Arrival Delay in Minutes']]
tab.head()
```

	Age	Flight Distance	Departure Delay in Minutes	Arrival Delay in Minutes
0	-1.745279	-0.731539	0.266393	0.073014
1	-0.951360	-0.957184	-0.361375	-0.237539
2	-0.885200	-0.047584	-0.387532	-0.392816
3	-0.951360	-0.629246	-0.099805	-0.159901
4	1.430397	-0.978244	-0.387532	-0.392816

5. Proposed Data Mining Solutions

5.1. Classification

A classification model is built to draw conclusions from the input values given for training. Based on the model, the provided input values can predict the class for the new inputs.

To create an airline customer satisfaction prediction model, we used three classification algorithms:

- Decision Tree classifier
- Random Forest classifier
- K-Nearest classifier

5.1.1. Classification Model Evaluation

Evaluating model performances to test how accurate a model performs and predicts the outcome is an important part of modeling classification models. A variety of techniques are used to evaluate the models.

- **Confusion Matrix:**

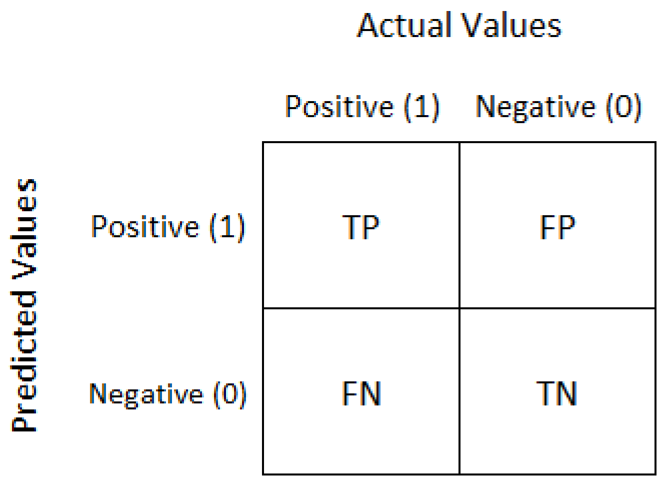
A confusion matrix can be used as an illustration technique to summarize an algorithm's performance. It can provide a better understanding of the accuracy with which classes are predicted by the models developed.
There are four possible outcomes when performing classification predictions.

True positives (TP): A true positive is when you predict an observation belongs to a class and it in fact does belong to that class.

True negatives (TN): A true negative is when you predict an observation does not belong to a class and it in fact does not belong to that class.

False positives (FP): A false positive is when you predict that an observation belongs to a class when in fact, it does not.

False negatives (FN): A false negative when you predict that an observation does not belongs to a class when in fact, it does.



- **Accuracy:** Accuracy tells us how close the result is to the actual value we were trying to achieve. How accurate is the modelbuilt.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

- **Precision:** Precision shows us how many of the correctly predicted cases turned out to be positive.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- **Recall:** Recall tells us how many of the truly positive cases we were able to predict properly with the model.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- **F1 Score:** Recall indicates the balance between precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.1.2. Classification Model Building

```
# Define the features and the target of Models
features = ['Type_of_Travel', 'Inflight_wifi_service','Online_boarding','Seat_comfort', 'Checkin_service']
target = ['satisfaction']

# Split into test and train
X_train = train[features]
y_train = train[target].to_numpy()
X_test = test[features]
y_test = test[target].to_numpy()

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

pickle.dump(scaler, open('scaler.pkl', 'wb'))
```

To avoid code repetition 2 functions where defined,

- calculate_time_taken function calculate the time taken for the execution.

```
def calculate_time_taken(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train.ravel(), verbose=0)
    else:
        model.fit(X_train,y_train.ravel())
    time_taken = time.time()-t0
    print("Time taken = {}".format(time_taken))

    return time_taken
```

- classification_model function calculates the accuracy, precision, recall, F1 score and plot the confusion matrix.

```
def classification_model(model, X_train, y_train, X_test, y_test):
    # train the model using the training set
    model.fit(X_train, y_train)
    # predictions
    y_pred = model.predict(X_test)
    # Model Accuracy
    accuracy = metrics.accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)
    #Plot the Confusion Matrix
    plt.figure(figsize=(6, 6))
    plot_confusion_matrix(model_dt, X_test, y_test,cmap=plt.cm.bone, normalize = 'all')
    plt.title('Confusion matrix of the Model',size = 16)
    plt.ylabel('Actual Class',size = 14)
    plt.xlabel('Predicted Class',size = 14)
    # Text summary of the precision, recall, F1 score
    print(classification_report(y_test,y_pred,digits=5))

    return model, accuracy
```

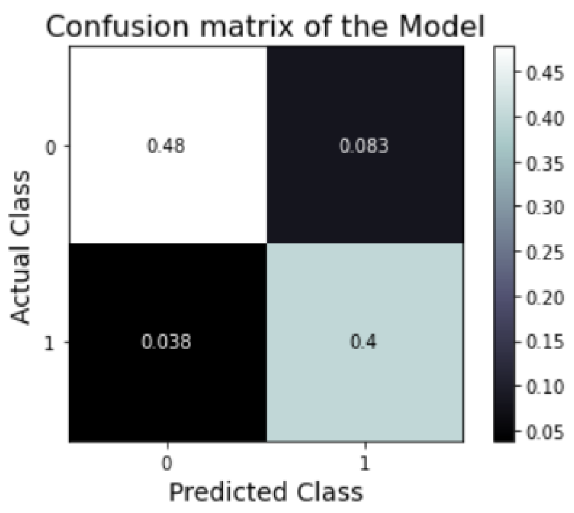
5.1.2.1. Decision Tree Classifier

Decision trees have an architecture similar to a flow chart, with a tree structure that categorizes labels based on their feature values. A node in a decision tree represents an instance, with branches representing test results and the leaf node representing the predicted class label.

```
# Decision Tree Model
model_dt = DecisionTreeClassifier(criterion = "gini",random_state = 100)

# Accuracy, confusion matrix and text summary (precision, recall, F1 score) according to the function
model_dt, accuracy_dt = classification_model(model_dt, X_train, y_train, X_test, y_test)

# Time taken to execute the model
tt_dt = calculate_time_taken(model_dt, X_train, y_train, X_test, y_test)
```

```
# save model
pickle.dump(model_dt, open('model.pkl', 'wb'))
```

```
Accuracy: 0.8787342161995688
              precision    recall  f1-score   support

           0    0.92556   0.85240   0.88748     14573
           1    0.82867   0.91239   0.86852     11403

    accuracy                        0.87873     25976
   macro avg    0.87712   0.88239   0.87800     25976
weighted avg    0.88303   0.87873   0.87915     25976

Time taken = 0.03965950012207031
```

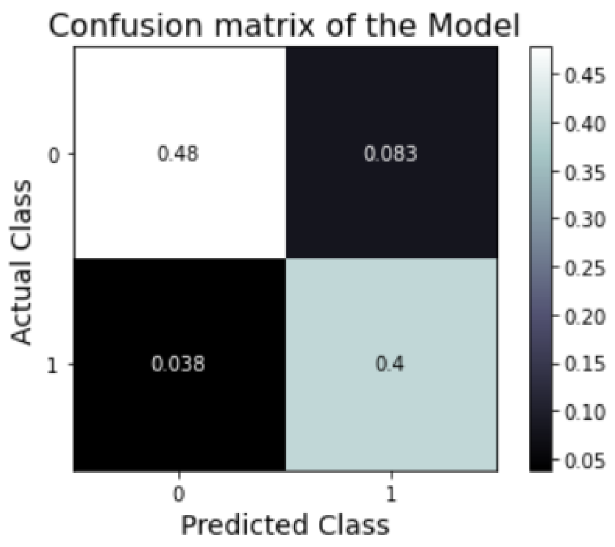


5.1.2.2. Random Forest Classifier

The random forest algorithm is a supervised machine learning algorithm composed of decision tree algorithms. Random Forest increases the model's randomness while growing trees. When splitting a node, it looks for the best feature from a random subset of features rather than the most important feature. This algorithm determines the outcome based on the decision trees' predictions. To obtain a more precise and stable prediction, it predicts the class using the average or mean of the output from various trees.

```
# Random Forest Model
model_rf = RandomForestClassifier()

# Accuracy, confusion matrix and text summary (precision, recall, F1 score) according to the function
model_rf, accuracy_rf = classification_model(model_rf, X_train, y_train, X_test, y_test)

# Time taken to execute the model
tt_rf = calculate_time_taken(model_rf, X_train, y_train, X_test, y_test)
```

```
Accuracy: 0.8760394210040037
              precision    recall  f1-score   support

           0    0.92562   0.84711   0.88463     14573
           1    0.82372   0.91301   0.86607     11403

    accuracy                        0.87604     25976
   macro avg    0.87467   0.88006   0.87535     25976
weighted avg    0.88089   0.87604   0.87648     25976

Time taken = 1.8972373008728027
```

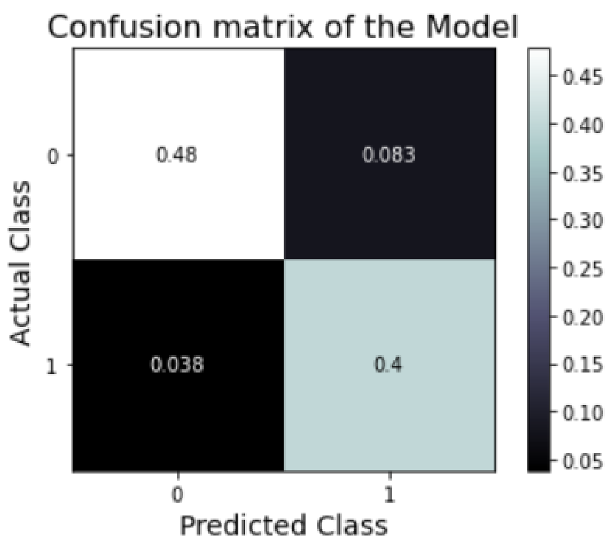


5.1.2.3. K-Nearest Classifier

The K neighbor classifier decides and classifies a test pattern using patterns from the training set. To make predictions and pass the final output, these classifiers look for similarities between the test pattern and every pattern in the training set.

```
# K Nearest Neighbour Model
model_kn = KNeighborsClassifier(n_neighbors=10, algorithm= 'kd_tree', n_jobs=4)

# Accuracy, confusion matrix and text summary (precision, recall, F1 score) according to the function
model_kn, accuracy_kn = classification_model(model_kn, X_train, y_train, X_test, y_test)

# Time taken to execute the model
tt_kn = calculate_time_taken(model_kn, X_train, y_train, X_test, y_test)
```

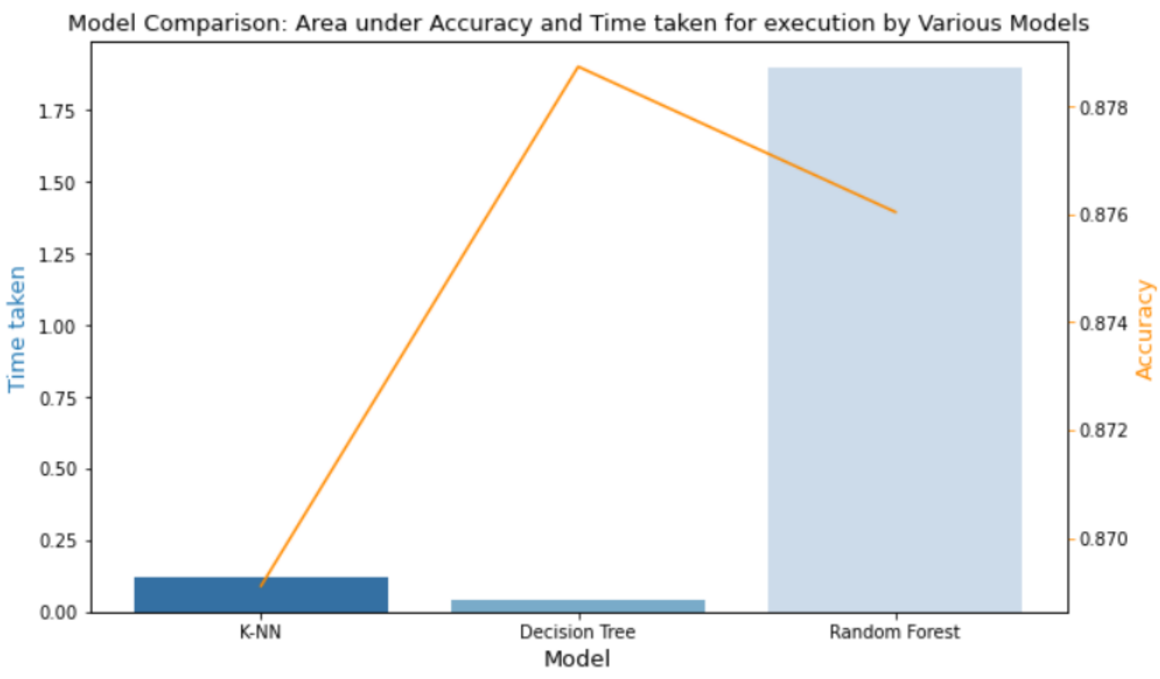
```
Accuracy: 0.8691099476439791
              precision    recall  f1-score   support

           0    0.90786   0.85329   0.87973     14573
           1    0.82588   0.88933   0.85643     11403

    accuracy                        0.86911     25976
   macro avg    0.86687   0.87131   0.86808     25976
weighted avg    0.87187   0.86911   0.86950     25976

Time taken = 0.12000894546508789
```



5.1.3. Classification Model Selection

The evaluation metrics were used to select the classification model.

```
accuracy = [accuracy_kn, accuracy_dt, accuracy_rf]
tt = [tt_kn, tt_dt, tt_rf]

model_data = {'Model': ['K-NN','Decision Tree','Random Forest'],
              'Accuracy': accuracy,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(10,6))
ax1.set_title('Model Comparison: Area under Accuracy and Time taken for execution by Various Models', fontsize=13)
color = 'tab:blue'
ax1.set_xlabel('Model', fontsize=13)
ax1.set_ylabel('Time taken', fontsize=13, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='Blues_r')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'darkorange'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
ax2.tick_params(axis='y', color=color)
```



- Among the three classification models, the Decision tree takes the least amount of time and has highest accuracy. Therefore, it was selected as the best model to be used for the prediction.

5.2. Clustering

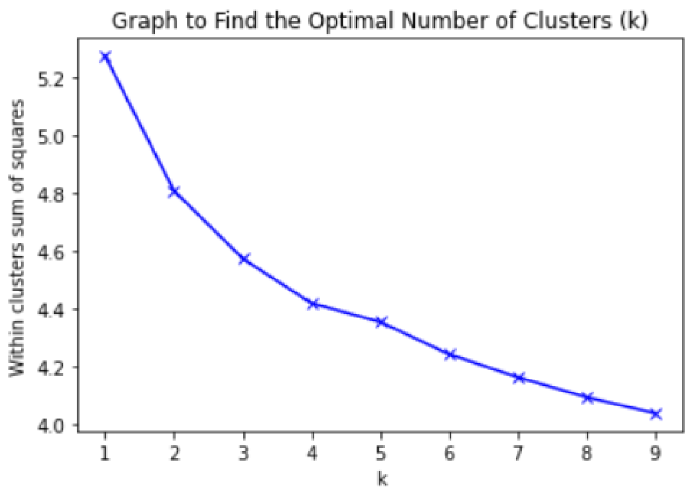
Clustering is the classification of a group of different data objects as similar objects. In the cluster analysis, data sets are divided into different groups based on their similarity. A label is assigned to each group of data after it has been classified into various groups. It helps in adapting to changes by categorizing them.

5.2.1 Clustering Model: K-means Clustering

K-Means Clustering is an unsupervised iterative machine learning model, which groups the unlabeled dataset into k number of different clusters in such a way that each dataset belongs only one group that has similar properties.

- Find the optimal number of clusters by defining an "Elbow Function".

```
# elbow function to find the optimal number of clusters
RSEED = 42
def elbow_function(df):
    sse = []
    K = range(1,10)
    for k in K:
        # in each iteration create a model with clusters=k
        kModel = KMeans(n_clusters=k, init = 'k-means++', random_state=RSEED).fit(df)
        kModel.fit(df)
        sse.append(sum(np.min(cdist(df, kModel.cluster_centers_, 'euclidean'), axis=1)) / df.shape[0])

    # plot the graph
    plt.plot(K, sse, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Within clusters sum of squares')
    plt.title(f'Graph to Find the Optimal Number of Clusters (k)')
    plt.show()

elbow_function(train_df)
```



From the above plot we have identified the value k at the "elbow" point as 4.
Thus, the Number of Initial Clusters = 4

- Define Model: Compute cluster centers and predict cluster index for each sample.

```
# kmeans = KMeans(n_clusters=4, random_state=RSEED)
kmeans = KMeans(n_clusters=4, init = 'k-means++', random_state=RSEED)
y = kmeans.fit_predict(train_df)
```

- Assign observations to k = 4 clusters which have created from the above model.

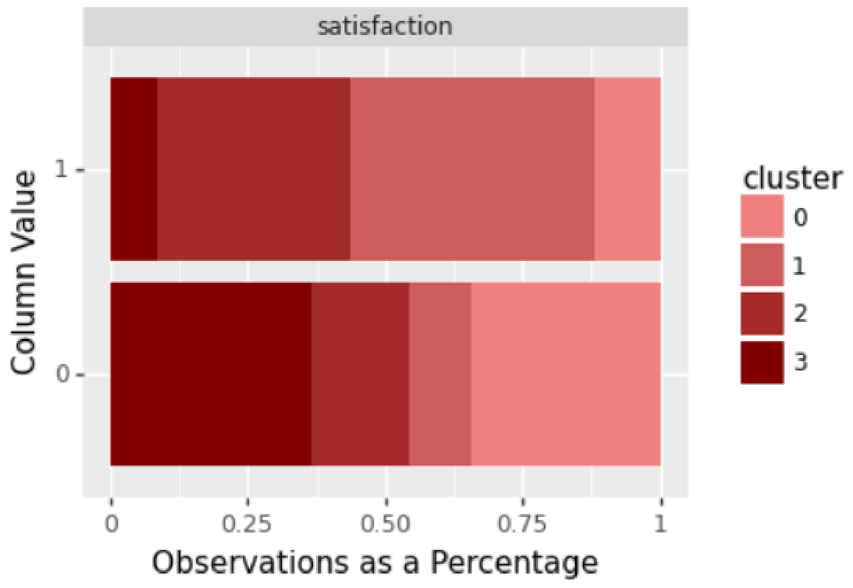
```
# function to assign the observations to the clusters created by K-Means
def get_divided_clusters(labels):
    outcome_df = dm.copy()

    outcome_df["cluster"] = labels
    divided_cluster = outcome_df.groupby("cluster").mean().reset_index().melt(id_vars="cluster")
    divided_cluster = divided_cluster["variable"].str.split("_" , expand=True).join(divided_cluster)
    divided_cluster = divided_cluster.rename({0 : "baseVar" , 1:"outcome"} , axis=1)
    divided_cluster["cluster"] = divided_cluster["cluster"].astype("category")
    return divided_cluster
```

- Visualize the clusters using "ggplot" grapgs.

```
# function to return ggplots
def get_ggplots():

  plot = (
    ggplot(cluster_distribution[~cluster_distribution["outcome"].isna()], aes(x="outcome", y="value", fill="cluster"))
    + geom_col(position="fill")
    + labs(x='Column Value', y='Observations as a Percentage')
    + coord_flip()
    + facet_wrap("~ baseVar")
    + theme(figure_size=(4, 3))
    + scale_fill_manual(values=('lightcoral', 'indianred', 'brown', 'maroon'))
  )

  return plot
```

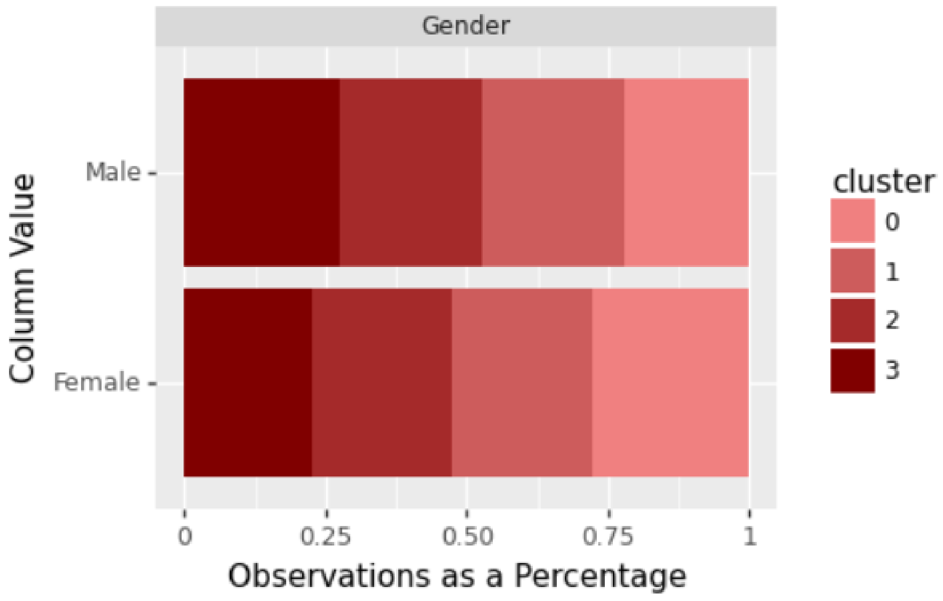
❖ Cluster distribution of the "Satisfaction" column

```
# satisfaction plot
dm = pd.get_dummies(train, columns=['satisfaction'])
cluster_distribution = get_divided_clusters(kmeans.labels_)
get_ggplots()
```



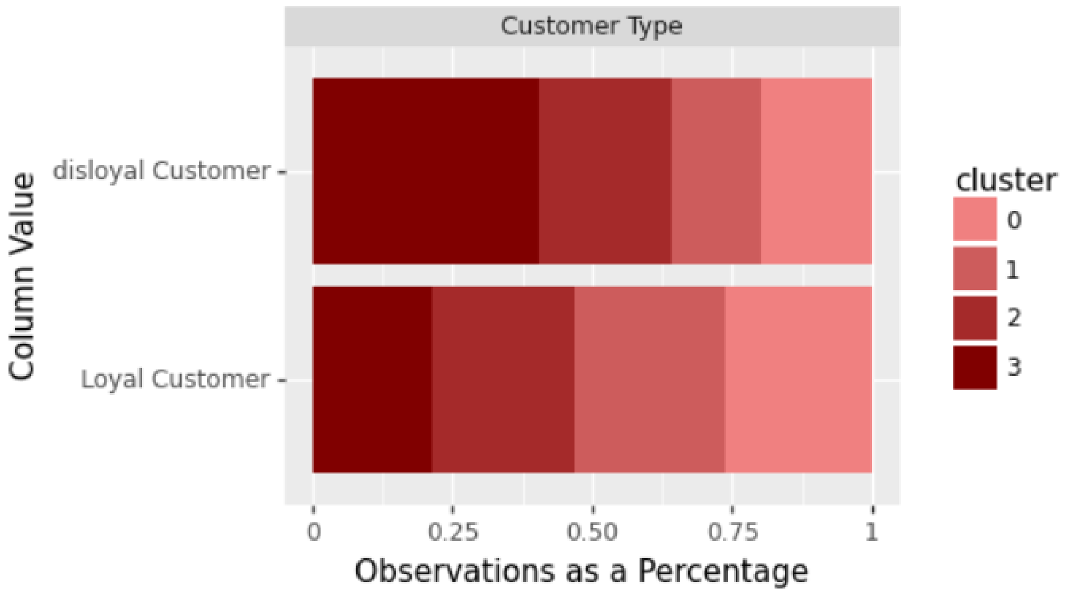
❖ Cluster distribution of the "Gender" column

```
# gender plot
dm = pd.get_dummies(train, columns=['Gender'])
cluster_distribution = get_divided_clusters(kmeans.labels_)
get_ggplots()
```



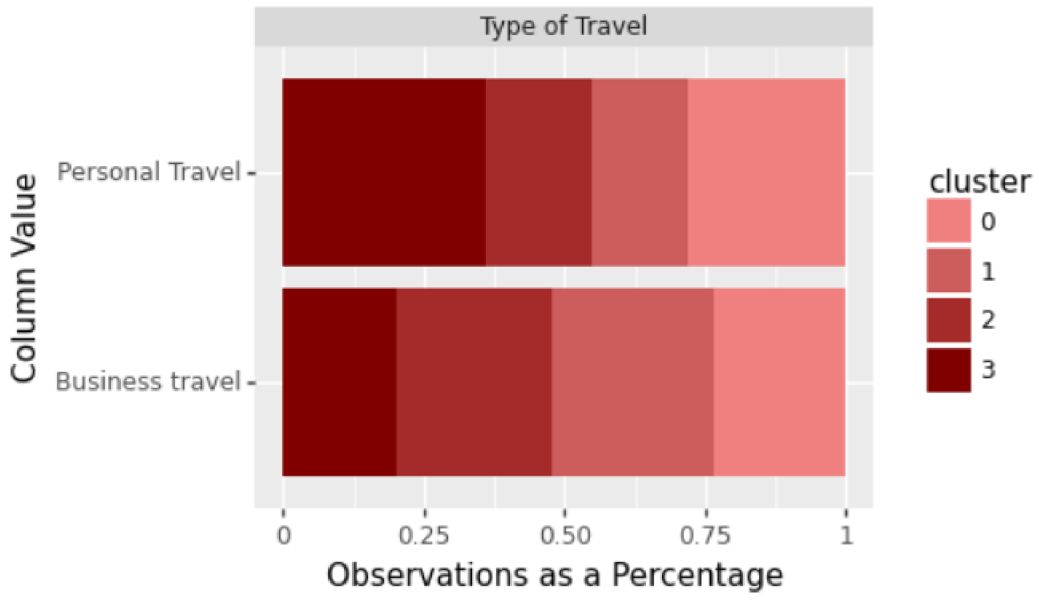
❖ Cluster distribution of the "Customer Type" column

```
# customer type plot
dm = pd.get_dummies(train, columns=['Customer Type'])
cluster_distribution = get_divided_clusters(kmeans.labels_)
get_ggplots()
```



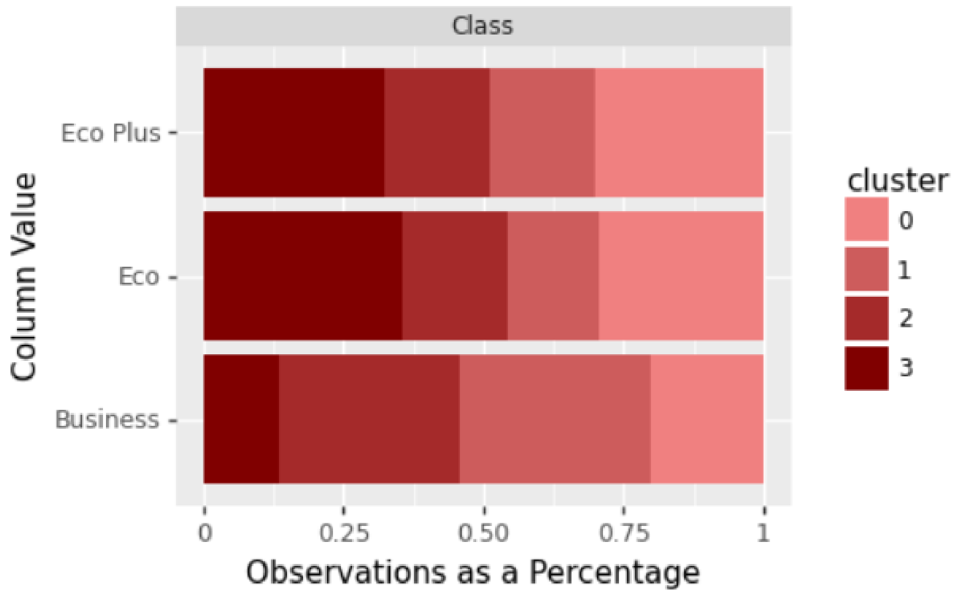
❖ Cluster distribution of the "Travel Type" column

```
# travel type plot
dm = pd.get_dummies(train, columns=['Type of Travel'])
cluster_distribution = get_divided_clusters(kmeans.labels_)
get_ggplots()
```



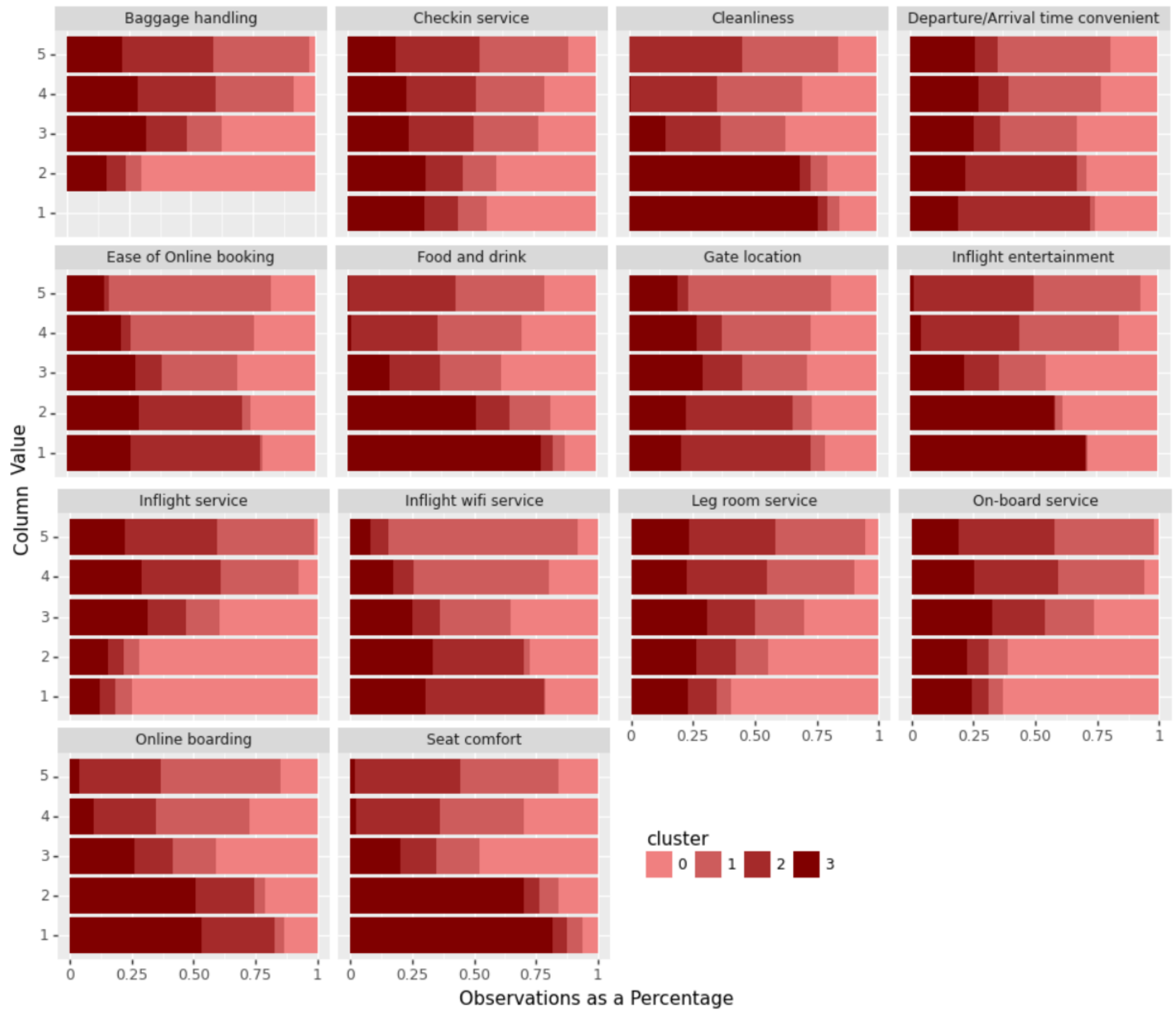
❖ Cluster distribution of the "Class" column

```
# class plot
dm = pd.get_dummies(train, columns=['Class'])
cluster_distribution = get_divided_clusters(kmeans.labels_)
get_ggplots()
```



❖ Cluster distribution of the passenger rating columns which are defined in the “cols” list.

```
# cluster distribution for passenger rating columns
cols = ['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']

dm = pd.get_dummies(train, columns=cols, drop_first=True)
cluster_distribution = get_divided_clusters(kmeans.labels_)

(
    ggplot(cluster_distribution[~cluster_distribution["outcome"].isna()], aes(x="outcome", y="value", fill="cluster"))
    + geom_col(position="fill")
    + labs(x='Value', y='Percentage')
    + coord_flip()
    + facet_wrap("~ baseVar")
    + theme(figure_size=(12, 10), legend_position = (0.6, 0.2))
    + scale_fill_manual(values=('lightcoral', 'indianred', 'brown', 'maroon'))
)
```



- Result Interpretation:

- Cluster 0:
  - Mostly dissatisfied
  - Mostly customers in Eco/Eco plus class for personal travel
  - Lowest rated services:
    - Inflight service
    - On board service
    - Baggage handling
- Cluster 1:
  - Satisfied
  - Mostly loyal customers in Business class for business travel
  - Highest rated services:
    - Inflight Wi-Fi service
    - Ease of online booking

- Cluster 2:
  - Mostly satisfied
  - Mostly customers in Business class for business travel
  - Lowest rated services:
    - Departure/arrival time convenience
    - Ease of online booking
    - Gate location
  - Highest rated services:
    - Inflight entertainment
    - Cleanliness
    - Food and drink

- Cluster 3:
  - Not satisfied
  - Mostly disloyal customers in Eco/ Eco Plus class for personal travel
  - Lowest rated services:
    - Cleanliness
    - Food and drink
    - Seat Comfort

- Conclusions:

Customers are more likely to feel “satisfied” if they have a high satisfaction with,

- Inflight Wi-Fi service, Cleanliness, Ease of online booking, and Food and drink

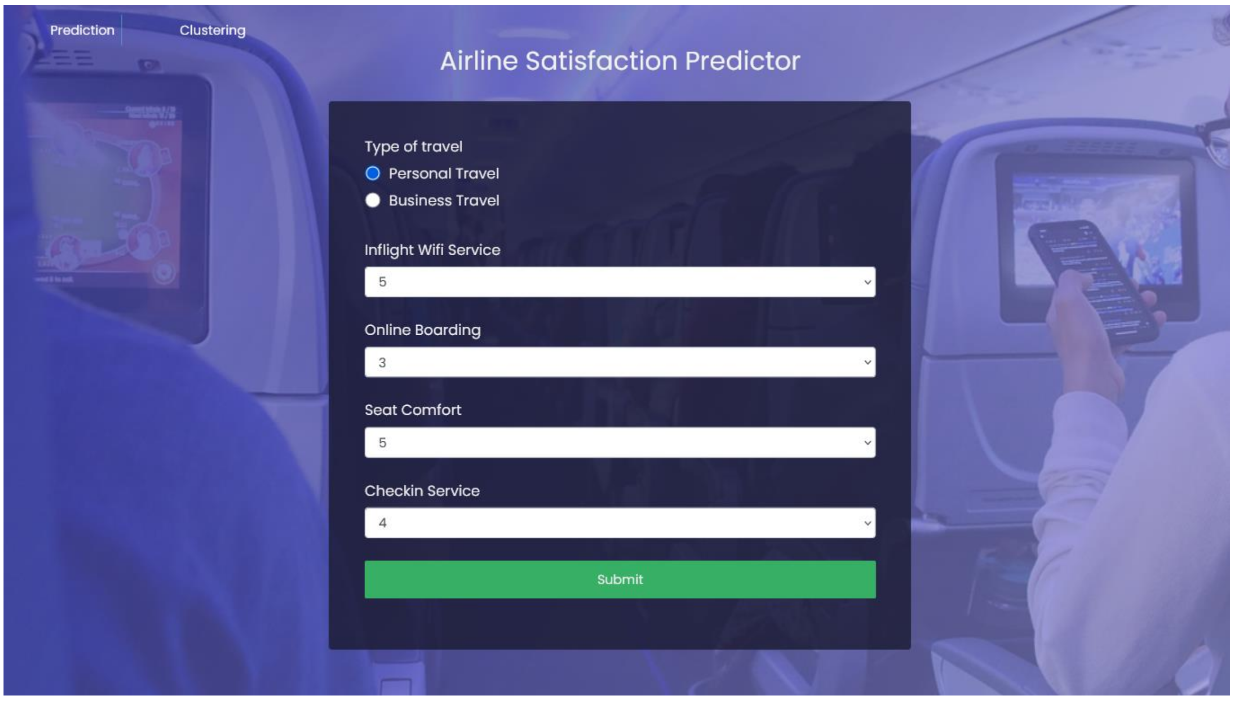
Customers are more likely to feel “dissatisfied” if they have a low satisfaction with,

- Cleanliness, Food and drink, Seat comfort, and Inflight service

6. UI Implementation

When the Type of Travel and scores for Inflight Wi-Fi Services, Online Boarding, Seat Comfort, and Check-in Service are entered into the prediction, it will show whether the passenger is satisfied or not.

- Situation where the passenger was satisfied with the service.



Airline Satisfaction Predictor

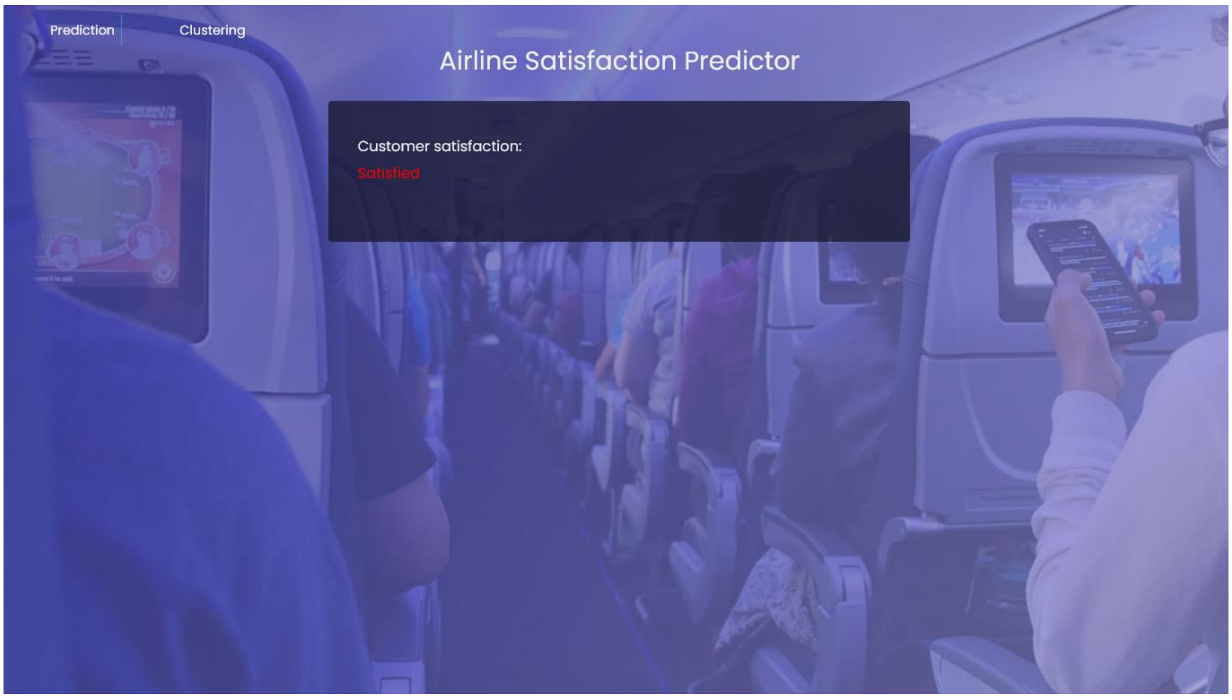
Type of travel
[✓ Personal Travel]
Business Travel

Inflight Wifi Service: 5

Online Boarding: 3

Seat Comfort: 5

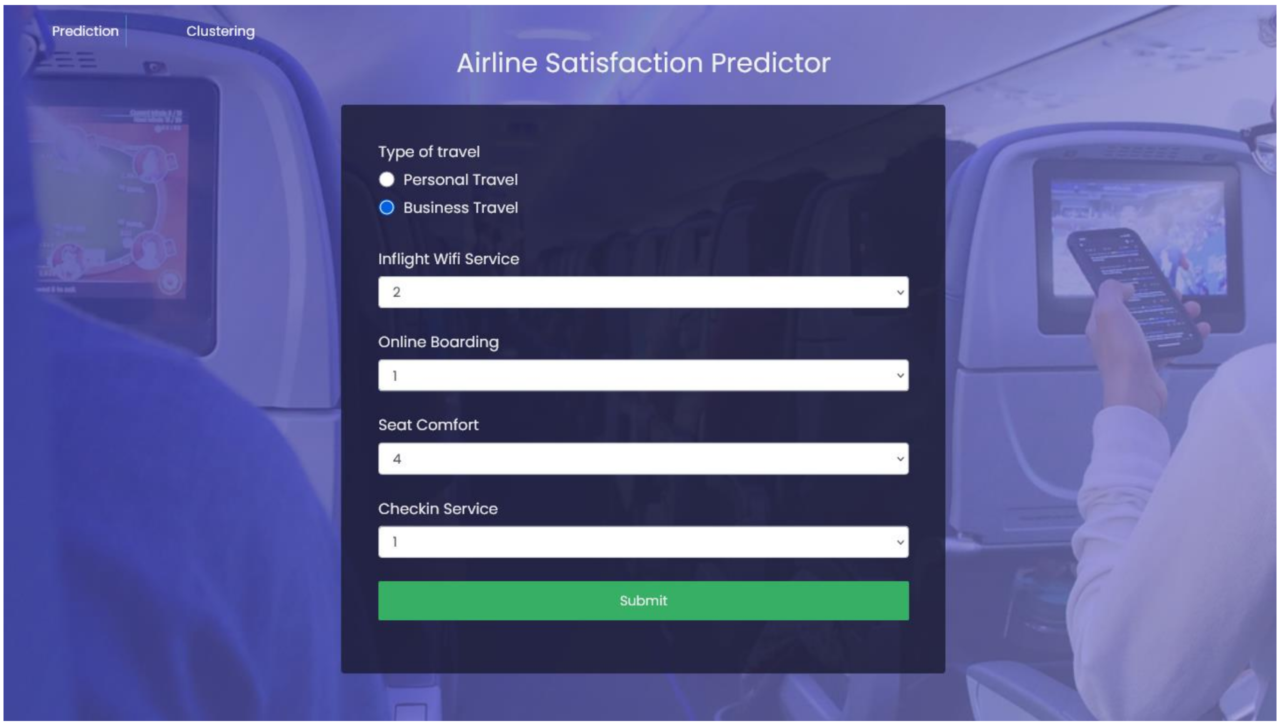
Checkin Service: 4



Airline Satisfaction Predictor

Customer satisfaction:
Satisfied

- Situation where the passenger was dissatisfied with the service.



Airline Satisfaction Predictor

Type of travel

○ Personal Travel

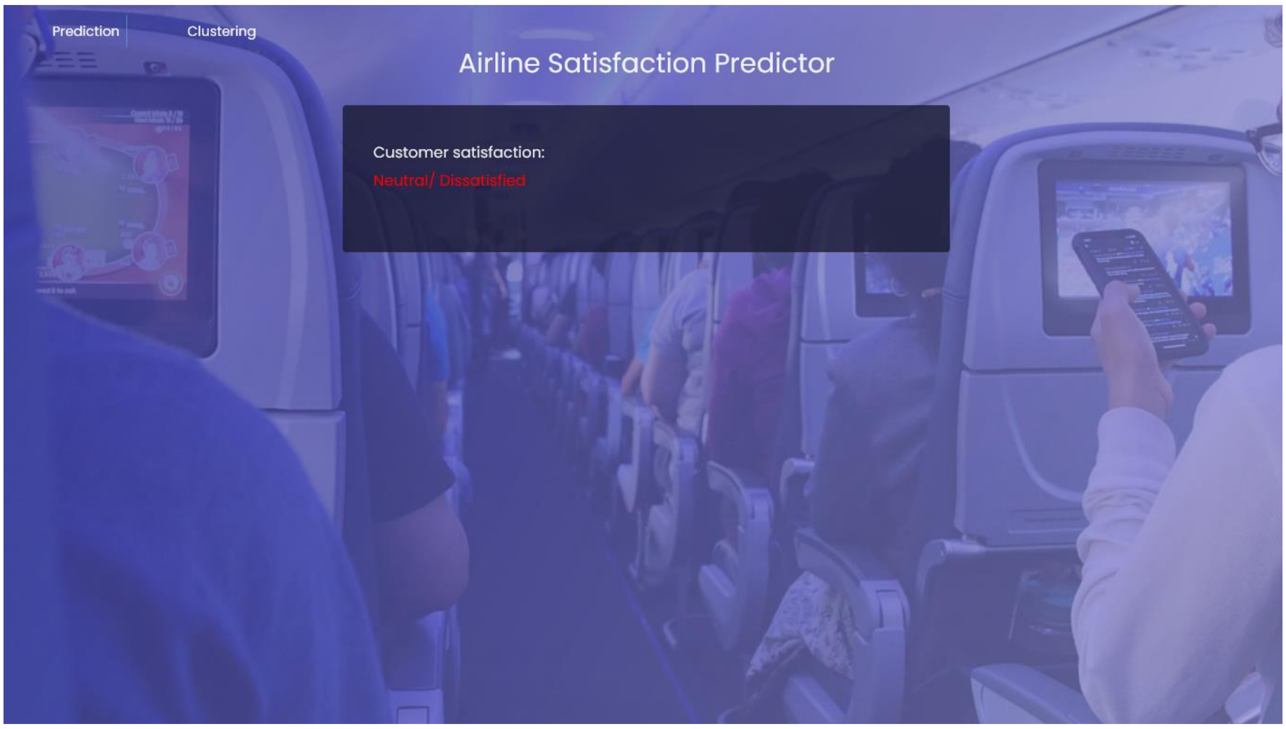
[✓ Business Travel]

Inflight Wifi Service: 2

Online Boarding: 1

Seat Comfort: 4

Checkin Service: 1

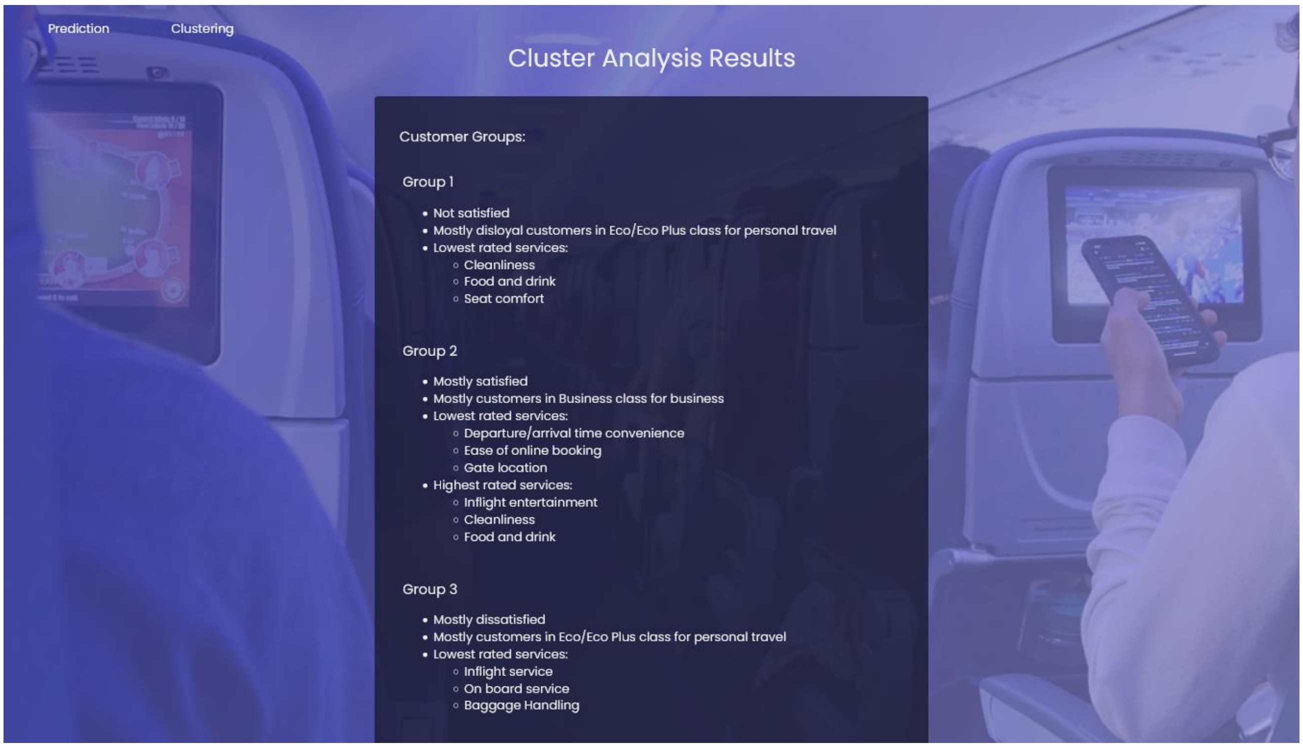


Airline Satisfaction Predictor

Customer satisfaction:

Neutral/ Dissatisfied

Information derived from clustering flight satisfaction details



Cluster Analysis Results

Customer Groups:

Group 1

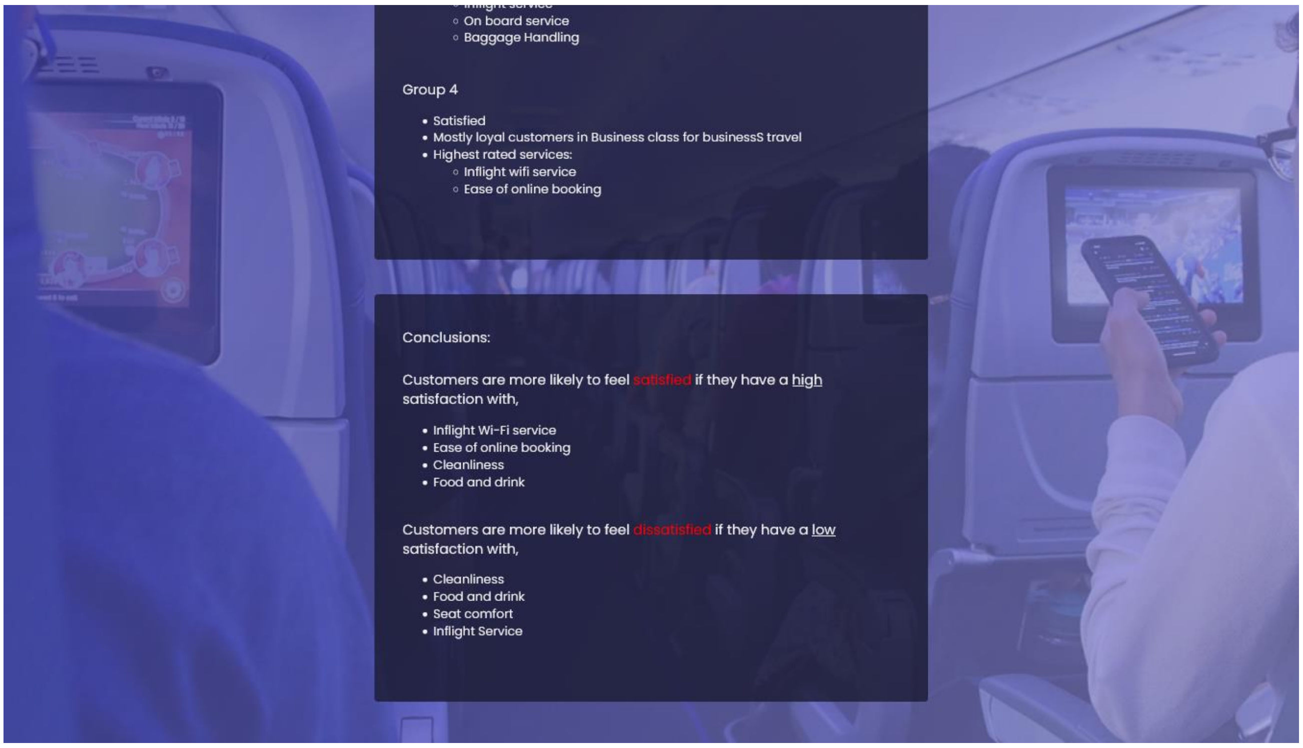
- Not satisfied
- Mostly disloyal customers in Eco/Eco Plus class for personal travel
- Lowest rated services:
  - Cleanliness
  - Food and drink
  - Seat comfort

Group 2

- Mostly satisfied
- Mostly customers in Business class for business
- Lowest rated services:
  - Departure/arrival time convenience
  - Ease of online booking
  - Gate location
- Highest rated services:
  - Inflight entertainment
  - Cleanliness
  - Food and drink

Group 3

- Mostly dissatisfied
- Mostly customers in Eco/Eco Plus class for personal travel
- Lowest rated services:
  - Inflight service
  - On board service
  - Baggage Handling



  - Inflight service
  - On board service
  - Baggage Handling

Group 4

- Satisfied
- Mostly loyal customers in Business class for businessS travel
- Highest rated services:
  - Inflight wifi service
  - Ease of online booking

Conclusions:

Customers are more likely to feel satisfied if they have a high satisfaction with,

- Inflight Wi-Fi service
- Ease of online booking
- Cleanliness
- Food and drink

Customers are more likely to feel dissatisfied if they have a low satisfaction with,

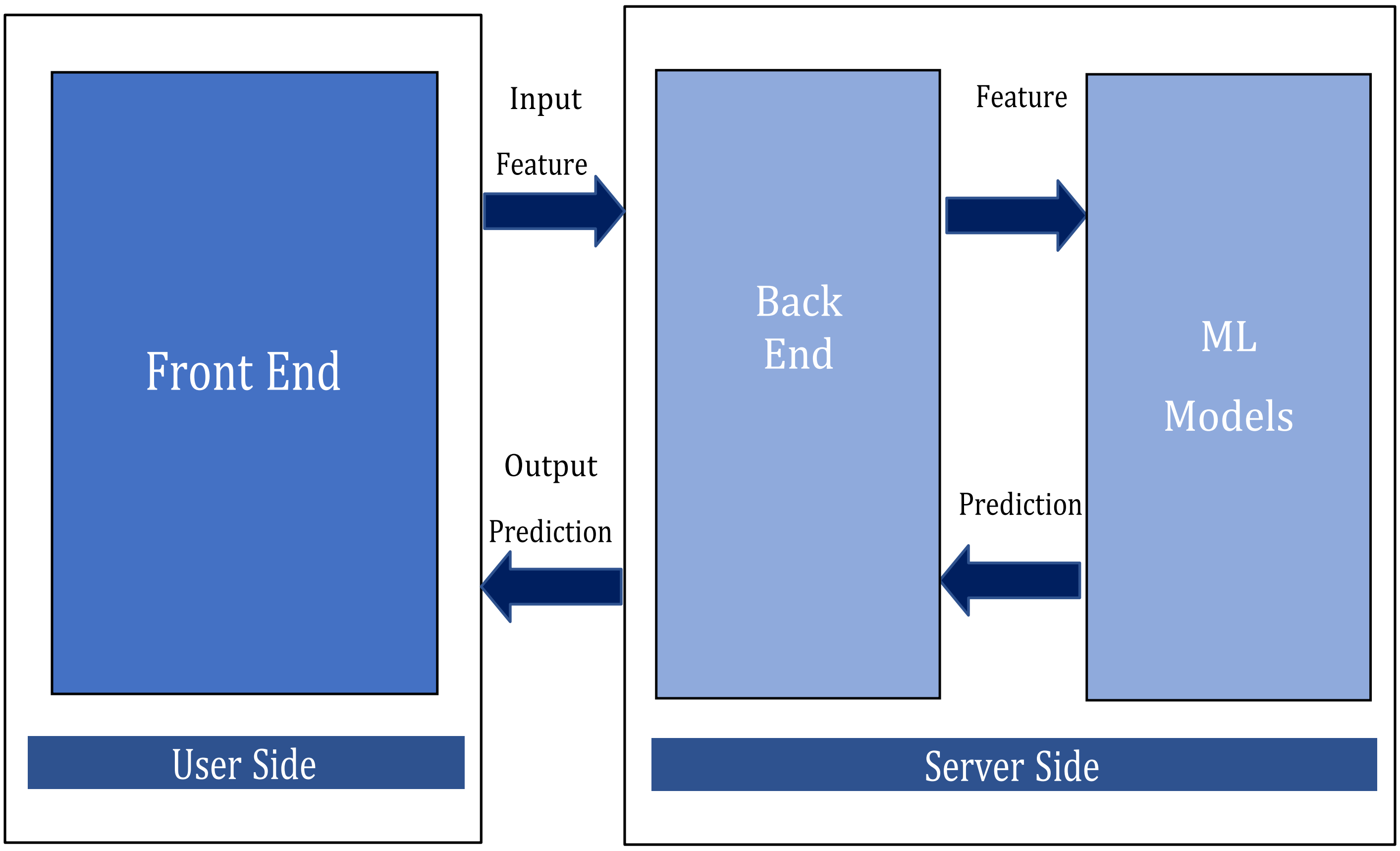
- Cleanliness
- Food and drink
- Seat comfort
- Inflight Service

7. Deployment Implementation

Python was used to create and train all classification and clustering models. The web application was built with Flask, an open-source Python framework. The developed application was deployed using Heroku, a container-based cloud Platform as a Service (PaaS).

GitHub Link: https://github.com/it20024772/FDMproject

Deployment Link: https://airlinesatisfaction.herokuapp.com/



8. Test Cases

Test Scenario ID	01 (Classification)		
Action	Enter positive rating to check if customer satisfaction is displayed as Satisfied.		
Inputs	**Expected Output**	**Actual Output**	**Result**
Type of travel: Personal Travel Inflight Wi-Fi Service: 5 Online Boarding: 3 Seat Comfort: 5 Check in Service: 4	Display customer satisfaction as "Satisfied"	Display customer satisfaction as "Satisfied"	Pass

Test Scenario ID	02 (Classification)		
Action	Enter negative rating to check if customer satisfaction is displayed as Neutral/ Dissatisfied.		
Inputs	**Expected Output**	**Actual Output**	**Result**
Type of travel: Business Travel Inflight Wi-Fi Service: 1 Online Boarding: 2 Seat Comfort: 2 Check in Service: 1	Display customer satisfaction as "Neutral/ Dissatisfied"	Display customer satisfaction as "Neutral/ Dissatisfied"	Pass

Test Scenario ID	03 (Classification)		
Action	Submitting the form with an empty field.		
Inputs	**Expected Output**	**Actual Output**	**Result**
Type of travel: Personal Travel Inflight Wi-Fi Service: 2 Online Boarding: Seat Comfort: 4 Check in Service: 3	Display an error message – "Please select an item in the list"	Display an error message – "Please select an item in the list"	Pass

9. Benefits of Proposed Solution

- **Classification:**

The classification models created achieve the initial goal of developing an application that predicts customer satisfaction with an airline's services. Airlines have the ability to focus on the main factors that influence customer satisfaction and increase the number of satisfied customers. The proposed prediction models are expected to assist airlines in developing new business strategies while saving them time from analyzing thousands of survey records.

- **Clustering:**

K-mean clustering is used to identify similar customer groups and gain insight into the characteristics that influence customer satisfaction. We use clustering to organize large amounts of airline passenger satisfaction survey responses in order to generate competitive insights about the airlines. Using these insights, airlines can succeed by maintaining a competitive edge by doing their utmost to establish and maintain reliable service, which can lead to consumer loyalty.

10. Conclusion

To predict the overall satisfaction of the customers with the airline services, three classification models were built, and the Decision Tree model has been selected as the best model due to its high accuracy and lower execution time. Five features were identified as the ones affecting the prediction the most and a prediction is made using the model on whether a customer is satisfied or dissatisfied with the airline services based on the customer’s survey responses to those features.

Finding the relationship between customer satisfaction and airline services was done using cluster analysis. The customers were grouped into four clusters and from their analysis it was concluded that customers having a higher satisfaction with Inflight Wi-Fi service, Ease of online booking, Cleanliness, and Food and drink are more likely to be satisfied with their experience while customers having a lower satisfaction with Cleanliness, Food and drink, Seat comfort and Inflight Service are less likely to be satisfied with their experience.

11. Project Team and Workload

Name	**RegistrationNumber**	**Responsibilities**
Rupasinghe J.D	IT20024772	• Model development • Web application frontend development • Data visualization • Documentation
Ariyasinghe P.A.D.N.I	IT20033828	• Model development • Web application backend development • Data analysis • Documentation
Dahanayake U.S	IT20043650	• Model development • Web application backend development • Data visualization • Documentation

12. References

[1] “A Grammar of Graphics for Python” [Online]. Available: https://plotnine.readthedocs.io/en/stable/

[2] "numpy.ravel-NumPy v1.23 Manual" [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.ravel.html

[3] “Permutation Importance” [Online]. Available: https://eli5.readthedocs.io/en/latest/blackbox/permutation_importan ce.html

[4] “sklearn.preprocessing.LableEncoder” [Online]. Available:

https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.h tml

[5] “pandas.get_dummies – pandas 1.5.1 documentation” [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html