Capstone Project

Exploration and Modelling Documentation

Yixuan Chen 301178375

Centennial College

BA 723001 Business Analytics Capstone

William Au

David Parent

August 12, 2022

Table of Contents

Executive Summary

0.1. Executive Introduction

This report is mainly about an airline customer satisfaction survey analysis. Business can acquire insights from this analysis for improving customer experience and satisfaction to seek long-term growth.

Python was the only programming language used throughout the study, and the Google Colab platform was used to write and run Python code. This document introduces the background of the research, problems and objectives of the enterprise. The process of data processing and application of the model are described, and final results are analyzed and explained. Based on the data analysis results, suggestions are put forward to enterprises.

0.2. Executive Objective

The main purpose of this paper is to describe the process of analyzing the company's historical data, identify and explain the best prediction model. Conclude suggestions based on key elements of impacting customers' satisfaction to help company concentrate sources on improving services or hardware facilities to improve customer experience and increase company revenue.

0.3. Executive Model Description

In this project, six models were applied and compared: Decision Tree, Random Forest, Logistic Regression, Neural Network, KNN and Max Voting Ensemble. The final recommended model is Random Forest, since it has the highest accuracy and F1 score, and the calculation and time cost are relatively low among these models.

0.4. Executive Recommendations

Under the Random Forest model, the important variables affecting customer satisfaction are successively determined as Inflight Entertainment, Seat comfort, Ease of Online Booking, Online support, etc. Therefore, relevant behaviors to improve corporate performance are recommended, such as enriching in-flight entertainment items, providing items for improving seat comfort, upgrading the online booking system, and so on.

Introduction

1.0. Background

Air CC is a young Canadian airline, mainly undertaking intra-Canadian routes and a few important international routes. As competition among aerospace companies becomes more intense, Air CC is facing huge competitive adjustments, with operational issues such as customer loss, underutilization of resources, and marketing strategies.

Major competitors of Air CC include Air Canada, Westjet, Air Transat, Emirates, Air France and others. According to statista data, in 2020, air Canada held 43.7% of the Canadian domestic market share, WestJet held 36.5% and other air companies held only less than 20% totally (statista, 2022). The entire aircraft industry has been hit hard by the impact of the epidemic， during which people travel much less, and flights for leisure travel and visiting relatives and friends will increase significantly after COVID-19 (Bouwer, Saxon, & Wittkamp, 2021). Remote working patterns and flexible work arrangements during the pandemic are likely to remain in place for a long time, making it difficult for business travel to return to pre-pandemic proportions (Bouwer, Saxon, & Wittkamp, 2021).

In order to change the situation, Air CC conducted a survey on the flying experience of historical customers, asking about various aspects of satisfaction during the flight and whether they were satisfied with the flight in general. Customer profile information including gender and age is also covered.

By implementing this project, the key factors that determine customer satisfaction are identified, and the company can strategically adjust the status quo based on these factors, such as optimizing the online ticketing system, training flight attendants, improving in-flight meals, and other measures to enhance the customer experience. As a result, customer loyalty will be improved and thus long-term growth is achieved. Without this project, the company may not target to improve some aspects that are truly important enough, which will result in a waste of resources and time.

2.0. Problem Statement

Customer churn has a huge negative impact on profit growth. The longer the customer stays with the airline, the more profit the airline will engage. When the customer has high satisfaction, the same airline will continue to be chosen the next time. But to get new customers, it requires marketing techniques and a lot of money to attract customers. And new customers can be recommended by old customers. In this context, in order to retain customers, the most relevant factors for satisfaction need to be identified based on the analysis of historical customer satisfaction survey data.

Due to the general environment of the epidemic, there is a significant decline in revenue in 2020 and 2021 compared to previous years, and the airline industry should see a

growth after the end of the epidemic. The company hopes to seize this opportunity to retain and attract customers and increase its overall revenue.

3.0. Objectives & Measurement

Objectives of model includes identifying key elements of influencing customers' satisfaction to help improve business performance and predicting whether future customers are satisfied or not.

In order to measure how successful of a model, the model selection statistic including accuracy and f1 score will be applied.

4.0. Assumptions and Limitations

Assumptions:

- This dataset was built 2 years ago, which is representative.
- These customers are randomly selected.

Limitations:

This survey is interviewed and recorded by humans, and the results may contain human error and multiple bias.

Data Sources

5.0. Data Set Introduction

This satisfaction feedback dataset is from Kaggle collected by an airline company (Jana, 2020). The dataset contains 129880 observations and 23 features including customer profile variables, like gender, age, customer type and so on. Airline service-related variables include satisfaction level of seat comfort, inflight entertainment, food and drink, etc. Among these variables, there are only 4 continues features including "Age", Flight Distance",

“Departure Delay in Minutes” and “Arrival Delay in Minutes”. Five variables are categorical, which are “satisfaction”, “Gender”, “Customer Type”, “Type of Travel” and “Class”. The left 14 discrete variables are ordinal.

6.0. Data Dictionary

Descriptions of all variables are as following:

Variables	Descriptions	Type
Gender	Male or Female	Object
Customer Type	Loyal or disloyal customer	Object
Age	Age ranges from 7 to 85	Integer
Type of Travel	Personal or Business travel	Object
Class	Class level including eco, eco plus, business	Object
Flight Distance	The length of a flight	Integer
Seat comfort	The satisfaction level of seat comfort (0-5 scale)	Integer
Departure / Arrival time convenient	The satisfaction level of Departure / Arrival time convenient (0-5 scale)	Integer
Food and drink	The satisfaction level of Food and drink (0-5 scale)	Integer
Gate location	The satisfaction level of Gate location (0-5 scale)	Integer
Inflight wifi service	The satisfaction level of Inflight wifi service (0-5 scale)	Integer
Inflight entertainment	The satisfaction level of Inflight entertainment (0-5 scale)	Integer
Online support	The satisfaction level of Online support (0-5 scale)	Integer
Ease of Online booking	The satisfaction level of Ease of Online booking (0-5 scale)	Integer

On-board service	The satisfaction level of On-board service (0-5 scale)	Integer
Leg room service	The satisfaction level of Leg room service (0-5 scale)	Integer
Baggage handling	The satisfaction level of Baggage handling (0-5 scale)	Integer
Checkin service	The satisfaction level of Checkin service (0-5 scale)	Integer
Cleanliness	The satisfaction level of Cleanliness (0-5 scale)	Integer
Online boarding	The satisfaction level of Online boarding (0-5 scale)	Integer
Departure Delay in Minutes	Flight departure delay time	Float
Arrival Delay in Minutes	Flight arrival delay time	Float
satisfaction	Satisfied or dissatisfied with the flight	Object

Data Exploration and Preparation

7.0. Data Exploration

Before data cleaning and variable engineering, this dataset is need to be explored to find duplicate rows, missing values, outliers, special correlations, distributions and so on. Plots and interpretation will be presented in this part.

Necessary libraries and the complete dataset were imported into Google Colab. The overview of this dataset was showed as the output including each feature's name, records count and datatype.

```
[ ]  %matplotlib inline
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt

[ ]  df = pd.read_csv("Airline.csv")
     print(df.info())
     df.head()
```

```
[ ]  df.columns=df.columns.str.replace(' ','_')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 23 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   Gender                            129880 non-null  object 
 1   Customer Type                     129880 non-null  object 
 2   Age                               129880 non-null  int64  
 3   Type of Travel                    129880 non-null  object 
 4   Class                             129880 non-null  object 
 5   Flight Distance                   129880 non-null  int64  
 6   Seat comfort                      129880 non-null  int64  
 7   Departure/Arrival time convenient 129880 non-null  int64  
 8   Food and drink                    129880 non-null  int64  
 9   Gate location                     129880 non-null  int64  
 10  Inflight wifi service             129880 non-null  int64  
 11  Inflight entertainment            129880 non-null  int64  
 12  Online support                    129880 non-null  int64  
 13  Ease of Online booking            129880 non-null  int64  
 14  On-board service                  129880 non-null  int64  
 15  Leg room service                  129880 non-null  int64  
 16  Baggage handling                  129880 non-null  int64  
 17  Checkin service                   129880 non-null  int64  
 18  Cleanliness                       129880 non-null  int64  
 19  Online boarding                   129880 non-null  int64  
 20  Departure Delay in Minutes        129880 non-null  int64  
 21  Arrival Delay in Minutes          129487 non-null  float64
 22  satisfaction                      129880 non-null  object 
dtypes: float64(1), int64(17), object(5)
memory usage: 22.8+ MB
None
```

```
[ ]  df.isnull().sum()
```

```
Gender                                0
Customer_Type                         0
Age                                   0
Type_of_Travel                        0
Class                                 0
Flight_Distance                       0
Seat_comfort                          0
Departure/Arrival_time_convenient     0
Food_and_drink                        0
Gate_location                         0
Inflight_wifi_service                 0
Inflight_entertainment                0
Online_support                        0
Ease_of_Online_booking                0
On-board_service                      0
Leg_room_service                      0
Baggage_handling                      0
Checkin_service                       0
Cleanliness                           0
Online_boarding                       0
Departure_Delay_in_Minutes            0
Arrival_Delay_in_Minutes            393
satisfaction                          0
dtype: int64
```

```
[ ]  df.duplicated().sum()
```

As the “isnull()” function result showed, there are 393 missing values existing in variable “ Arrival_Delay_in_ Minutes”, which is required to be imputed. In addition, there was no duplicated rows in this dataset.

In order to find whether there are values out of theoretical range or not, statistical description of all numerical variables is displayed.

```
df.describe()
```

	Age	Flight_Distance	Seat_comfort	Departure/Arrival_time_convenient	Food_and_drink	Gate_location	Inflight_wifi_service	Inflight_entertainment	Online_support
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000
mean	39.427957	1981.409055	2.838597	2.990645	2.851994	2.990422	3.249130	3.383477	3.519703
std	15.119360	1027.115606	1.392983	1.527224	1.443729	1.305970	1.318818	1.346059	1.306511
min	7.000000	50.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	27.000000	1359.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	3.000000
50%	40.000000	1925.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000
75%	51.000000	2544.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000
max	85.000000	6951.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

Ease_of_Online_booking	On-board_service	Leg_room_service	Baggage_handling	Checkin_service	Cleanliness	Online_boarding	Departure_Delay_in_Minutes	Arrival_Delay_in_Minutes
129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129487.000000
3.472105	3.465075	3.485902	3.695673	3.340807	3.705759	3.352587	14.713713	15.091129
1.305560	1.270836	1.292226	1.156483	1.260582	1.151774	1.298715	38.071126	38.465650
0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2.000000	3.000000	2.000000	3.000000	3.000000	3.000000	2.000000	0.000000	0.000000
4.000000	4.000000	4.000000	4.000000	3.000000	4.000000	4.000000	0.000000	0.000000
5.000000	4.000000	5.000000	5.000000	4.000000	5.000000	4.000000	12.000000	13.000000
5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1592.000000	1584.000000

There could be outliers only in the variable “Flight Distance”, since the minimum value is 50 and the maximum value is 6951, but the unit of distance is not given. According to flight distance calculator, the flight distance from Toronto to Beijing is 6596 miles and it’s 57 miles between Toronto and Kitchener (DISTANCE CALCULATOR, 2022). As a result, I can assume the unit of “Flight Distance” is miles and there are no outliers theoretically.

Descriptions of nonnumerical features were presented, including Gender, Customer_Type, Type_of_Travel, Class and satisfaction. Class imbalance is one of key issues impacting model accuracy. Effective solutions like resampling or SMOTE are often used to deal with this problem. However, in this dataset, classes are balanced with frequency of

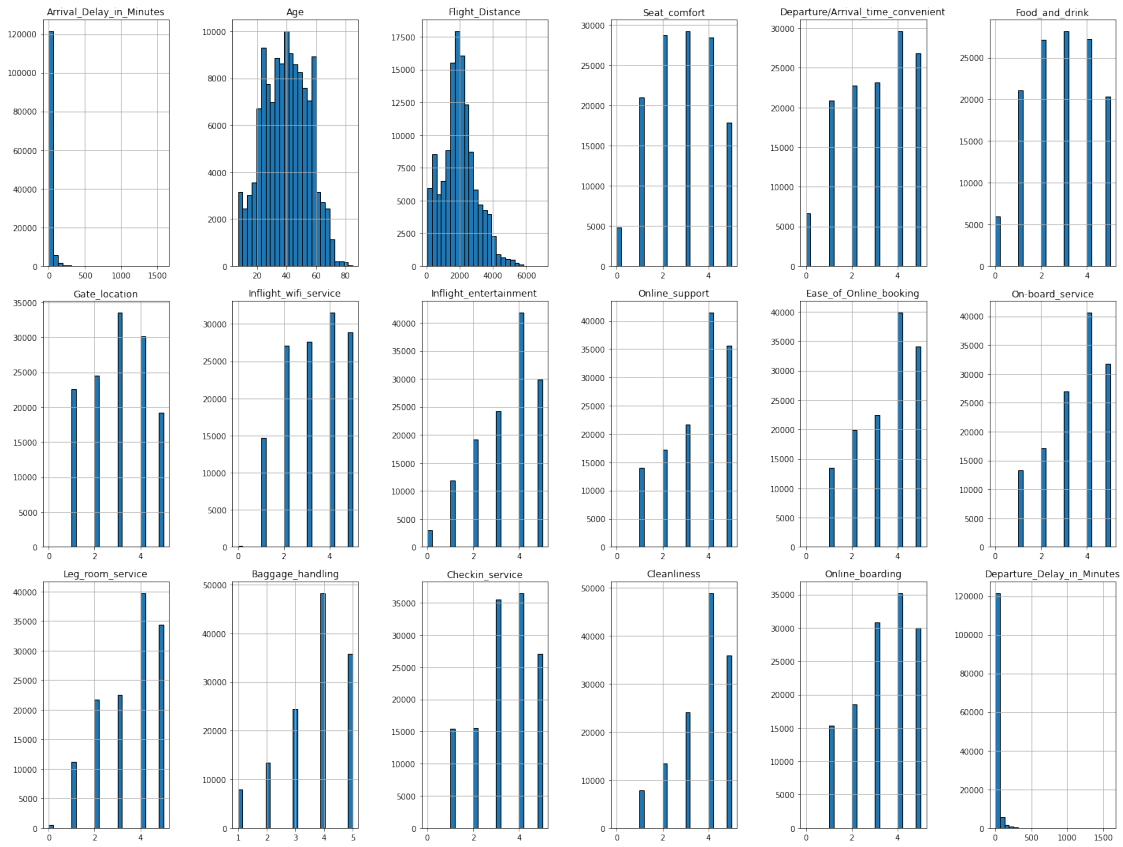
“satisfied” being 71087, accounting for about 55% of total records. No further action dealing with imbalance needed.

```
[ ]  df.describe(exclude='number')
```

	Gender	Customer_Type	Type_of_Travel	Class	satisfaction
count	129880	129880	129880	129880	129880
unique	2	2	2	3	2
top	Female	Loyal Customer	Business travel	Business	satisfied
freq	65899	106100	89693	62160	71087

Plots of all numerical features were drawn to see distribution of these variables. Continues and discreate features are separated to see correlates.

```
[ ]  # histograms for each numerical feature
     df_impu.hist(bins=25, figsize=(20, 15), layout=(-1, 6), edgecolor="black")
     plt.tight_layout();
```

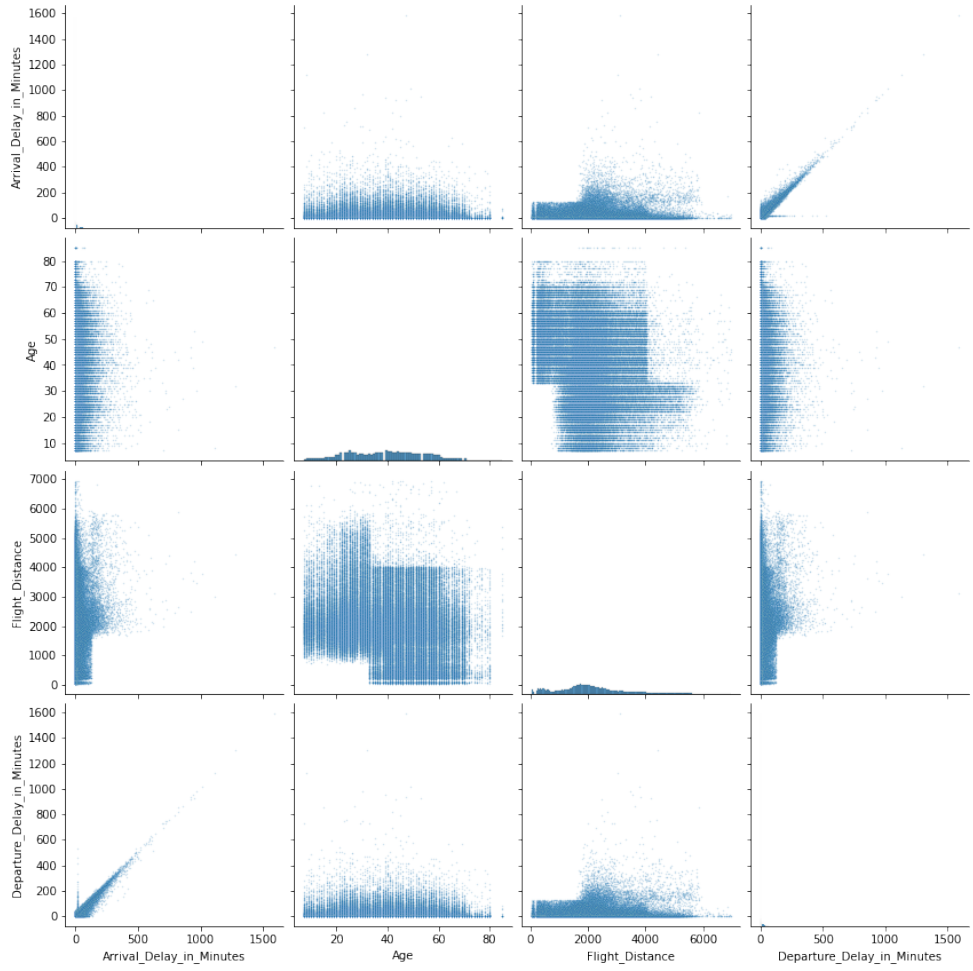


```
[ ]  # numerical features with more and equal than 7 unique features
     cols_continuous = df_impu.select_dtypes(include="number").nunique() >= 7

[ ]  # New dataframes only contains the continuous features or discrete features
     df_continuous = df_impu[cols_continuous[cols_continuous].index]
     df_discrete = df_impu[cols_continuous[~cols_continuous].index]
     print(df_discrete.shape)
     print(df_continuous.shape)

     (129880, 14)
     (129880, 4)
```

```
[ ]  # pair plot for continuous variables
     sns.pairplot(df_continuous, height=3, plot_kws={"s": 2, "alpha": 0.2});
     #a strong linear relationship between departure delay time and arrival delay time
```



According to histograms of numerical features and pair plot of continuous variables, the “Departure Delay in Minutes” and “Arrival Delay in Minutes” have a strong linear

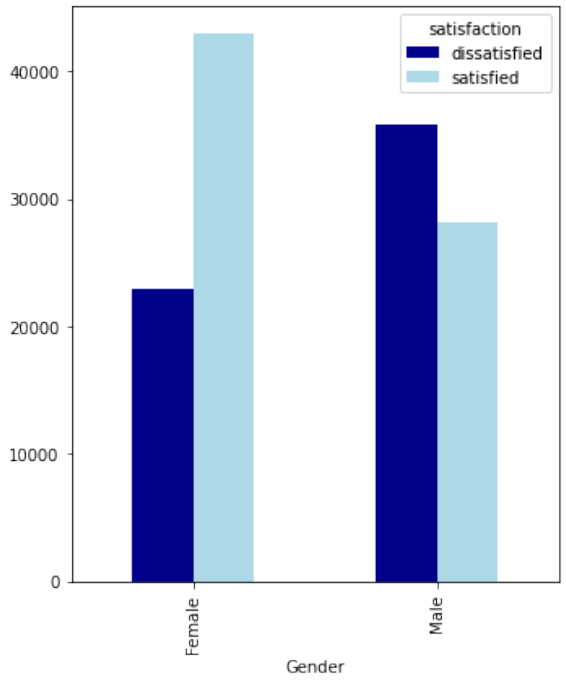
relationship. Theoretically, these two variables can be interpreted as highly correlated, since a delay in departure time leads directly to a delay in arrival time.

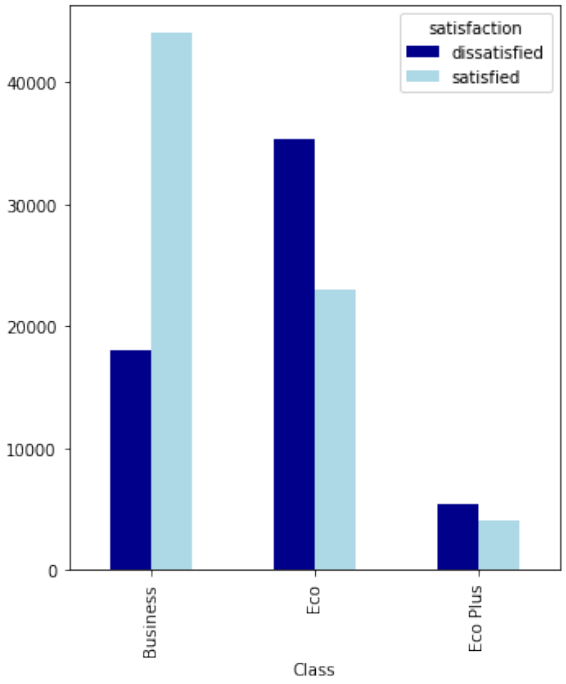
```
[ ] pd.crosstab(df_impu['Gender'],df_impu['satisfaction']).plot.bar(figsize=(5,6), color=('DarkBlue','LightBlue','Teal'))
    plt.tight_layout()
    #Femal-more satisfied
```

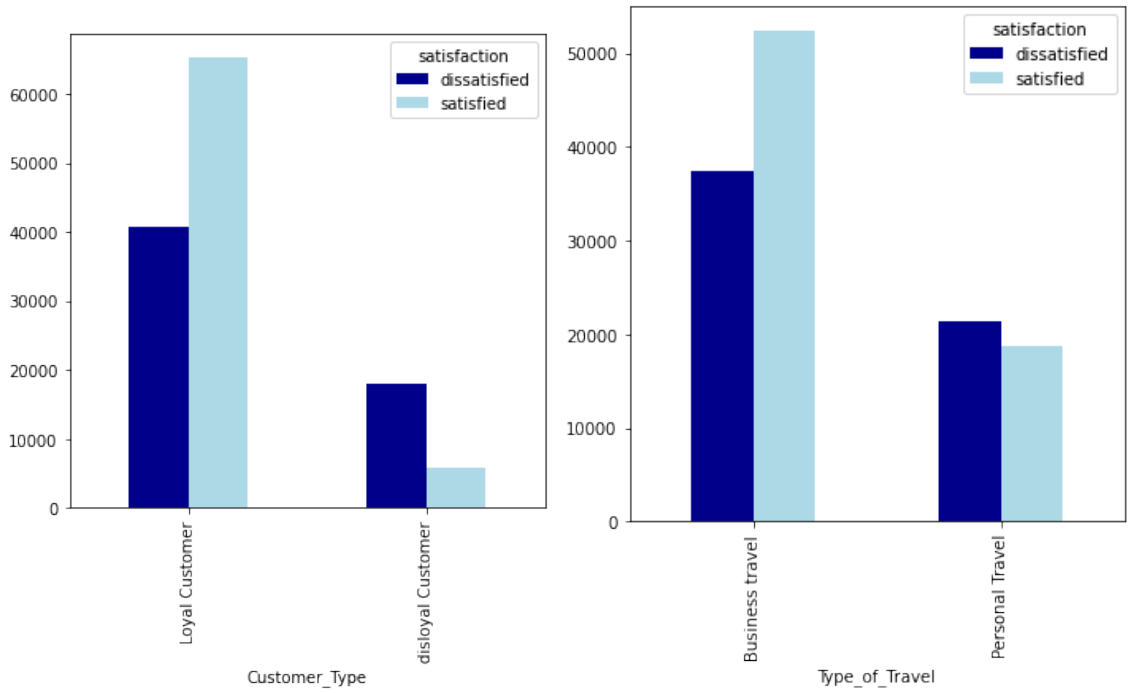
```
[ ] pd.crosstab(df_impu['Customer_Type'],df_impu['satisfaction']).plot.bar(figsize=(5,6), color=('DarkBlue','LightBlue','Teal'))
    plt.tight_layout()
    #Loyal customer-more satisfied
```

```
[ ] pd.crosstab(df_impu['Class'],df_impu['satisfaction']).plot.bar(figsize=(5,6), color=('DarkBlue','LightBlue','Teal'))
    plt.tight_layout()
    #Business class: Highest satisfaction
    #Eco: Lowest satisfaction
```

```
[ ] pd.crosstab(df_impu['Type_of_Travel'],df_impu['satisfaction']).plot.bar(figsize=(5,6), color=('DarkBlue','LightBlue','Teal'))
    plt.tight_layout()
    #business travel-more satisfied
```







Different satisfaction distributions were studied based on gender, customer type, cabin class, and travel type. I found that:

- More male passengers were dissatisfied with the airline, while most female passengers were satisfied with airline.
- Loyal customers have a high probability of satisfaction, and most non-loyal customers expressed dissatisfaction in the survey.
- Among the different cabin classes, business class passengers have a higher satisfaction rate, while economy class passengers express more dissatisfaction, and Eco Plus passengers have a slightly higher dissatisfaction rate than satisfaction rate.
- For groups of people with different purpose, the satisfaction rate of travelers on business trips is higher, and the dissatisfaction rate of travelers on personal trips is slightly higher than the satisfaction rate.

8.0. Summary

- There are 393 missing values in variable “Arrival_Delay_in_Minutes”.
- Variable “Arrival_Delay_in_Minutes” and “Departure_Delay_in_Minutes” are highly correlated.
- Highly skewed variables exist: “Arrival_Delay_in_Minutes”, “Departure_Delay_in_Minutes”.
- No duplicated rows found.
- No values out of the theoretical range.
- No class imbalance issue.

9.0. Data Cleansing

9.1. Imputation of Missing Data

A new variable was added by indicating the missing value of “Arrival_Delay_in_Minutes”. If there is a blank in a record, there is a “True” indicator showing in the new feature column “Arrival_Delay_in_Minutes”.

```
[ ]  #indicator
     df['Arrival_Delay_Imputation'] = df['Arrival_Delay_in_Minutes'].isnull()
     df
```

Departure_Delay_in_Minutes	Arrival_Delay_in_Minutes	satisfaction	Arrival_Delay_Imputation
0	0.0	satisfied	False
310	305.0	satisfied	False
0	0.0	satisfied	False
0	0.0	satisfied	False
0	0.0	satisfied	False

For missing data, the KNN imputation was applied. The KNN algorithm matches a point with its nearest K neighbors and is effective for all types of missing data (Obadia, 2017).

Traditional imputation methods, such as filling null values with average values, is not applicable in this situation.

```
[ ]  #KNN imputation
     from sklearn.impute import KNNImputer

     imputer = KNNImputer(n_neighbors=2)
     imputed = imputer.fit_transform(df[['Arrival_Delay_in_Minutes']])
     df_impu = pd.DataFrame(imputed, columns=df[['Arrival_Delay_in_Minutes']].columns)
```

```
[ ]  df_impu = pd.concat([df_impu, df.drop('Arrival_Delay_in_Minutes', axis=1)],axis=1)
```

```
[ ]  df_impu.isnull().sum().sum()

     0
```

9.2. Transformation of Skewed Variables

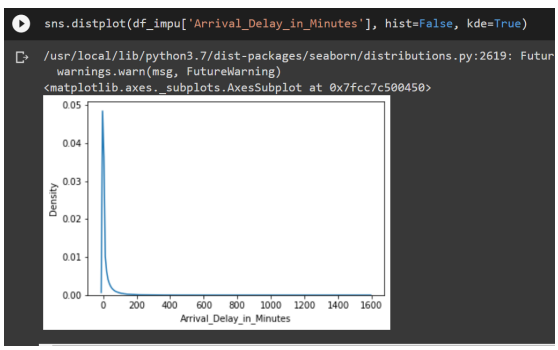
The skewness level of "Arrival_Delay_in_Minutes" and "Departure_Delay_in_Minutes" were both calculated to be more than 6.5. The Box-Cox transformation was applied to reduce skewness level in this case. I have also tried other transformation methods, such as log transformation, but the effect of skewness processing is not as good as the Box-Cox transformation based on skewness score.

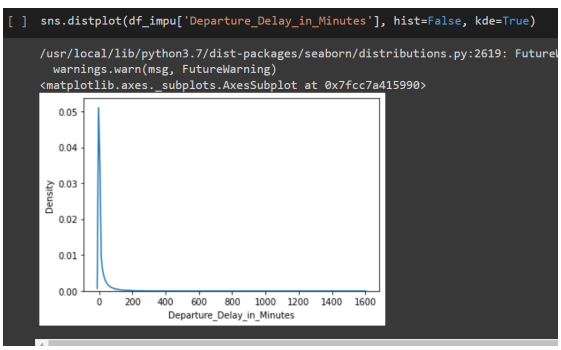
```
[ ]  df_impu['Arrival_Delay_in_Minutes'].skew()

     6.680238800959987
```

```
[ ]  df_impu['Departure_Delay_in_Minutes'].skew()

     6.821980310173458
```





```
[ ]  #boxcox transform
     from scipy.special import boxcox1p

     df_A = df_impu['Arrival_Delay_in_Minutes'].apply(lambda x: boxcox1p(x,0.25))
     df_D = df_impu['Departure_Delay_in_Minutes'].apply(lambda x: boxcox1p(x,0.25))
```

```
[ ]  df_updated = pd.concat([df_A, df_impu.drop( 'Arrival_Delay_in_Minutes',axis=1)],axis=1)

[ ]  df_updated = pd.concat([df_D, df_updated.drop( 'Departure_Delay_in_Minutes',axis=1)],axis=1)

[ ]  df_updated['Arrival_Delay_in_Minutes'].skew()

     1.3646865457342778

[ ]  df_updated['Departure_Delay_in_Minutes'].skew()

     1.4138001008967958
```

9.3. Encode Categorical Variables

All categorical data are required to transformed to numerical, since categorical variables cannot be supported by the scikit-learn.

I transferred categorical variables to dummy columns with “get_dummies()” function, including "Gender”, “Customer_Type”, “Class”, “Type_of_Travel”, “Arrival_Delay_Imputation”. In order to reduce new correlated features, “drop_first=True” was add to drop the first column that is not necessary when creating dummy variables. In addition, the target variable “satisfaction” was integer mapped with LabelEncoder.

```
[ ]  cat_coll=['Gender','Customer_Type','Class','Type_of_Travel','Arrival_Delay_Imputation']

[ ]  df_updated[cat_coll]=df_updated[cat_coll].astype('category')

[ ]  df_dum=pd.get_dummies(df_updated[cat_coll],drop_first=True)

[ ]  df_dum=pd.concat([df_dum, df_updated.drop(axis=1, columns=df_updated[cat_coll])],axis=1)
     df_dum
```

```
[ ]  #integer mapping with LabelEncoder
     from sklearn.preprocessing import LabelEncoder

     df_dum['satisfaction']=df_dum[['satisfaction']].apply(LabelEncoder().fit_transform)
```

9.4. Drop High Correlated Variable

In order to increase the stability of models, one of highly correlated feature could be eliminated. Base on the correlation heatmap of all variables, “Arrival_Delay_in_Minutes” and “Departure_Delay_in_Minutes” are highly correlated, but “Departure_Delay_in_Mi”utes" is more related to the target variable “satisfaction”. As a result, I dropped variable “Arrival_Delay_in_Minutes”.

```
[ ]  corr=df_dum.corr()
     plt.figure(figsize=(20,20))
     sns.heatmap(corr,annot=True,cmap='PuBu')
     plt.title('Correlation Matrix')
     plt.show()
```


```
[ ]  del df_dum['Arrival_Delay_in_Minutes']
     df_dum
```

	Gender_Male	Customer_Type_disloyal Customer	Class_Eco	Class_Eco Plus	Type_of_Travel_Personal Travel	Arrival_Delay_Imputation_True	Departure_Delay_in_Minutes	Age
0	0	0	1	0	1	0	0.000000	65
1	1	0	0	0	1	0	12.797710	47
2	0	0	1	0	1	0	0.000000	15
3	0	0	1	0	1	0	0.000000	60
4	0	0	1	0	1	0	0.000000	70
...	...	...	...	...	...	...	...	...
129875	0	1	1	0	1	0	0.000000	29
129876	1	1	0	0	1	0	10.548543	63
129877	1	1	1	0	1	0	10.136475	69
129878	1	1	1	0	1	0	10.928302	66
129879	0	1	1	0	1	0	10.771964	38

129880 rows × 24 columns

The dataset was finished cleaning with 24 features before fitting to models, with new dummy variables: Gender_Male, Customer_Type_disloyal Customer, Class_Eco, Class_Eco Plus and Type_of_Travel_Personal Travel.

Model Exploration

10.0. Modeling Introduction

In this analysis, multiple models are applied including Decision Tree, Random Forest, Logistic Regression, Neural Network, KNN and Max Voting Ensemble. Finally, the model accuracy and F1 score were compared to determine the best model and explain the important variables under the best model.

In order to optimize models and avoid overfitting, cross validation is used. Random search and grid search are used to tune hyperparameter. Both methods objectively search for different values of model hyperparameters, and finally select a set of hyperparameters with good performance to achieve the best performance. Random search is good for finding and retrieving unknown combinations of hyperparameters, but takes more time to execute (Brownlee, 2020).

11.0. Decision Tree

Classification Tree can predict the classification results of target variable by learning data features. Decision tree is easy to understand and interpret, but are prone to overfitting. This can be avoided by pruning, setting the depth of the tree and other ways.

```
[ ]  from sklearn import metrics
     from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
     from sklearn.metrics import plot_confusion_matrix,accuracy_score, confusion_matrix, mean_squared_error,classification_report,precision_score, recall_score, f1_score
     from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV, GridSearchCV
     !pip install dmba
     from dmba import plotDecisionTree, classificationSummary, regressionSummary
```

Cleaned dataset was divided into training and testing sets, with a ratio of 8 to 2.

Randoms Search was used to obtain the optimal hyperparameter combination:

{'min_samples_split': 8, 'min_samples_leaf': 3, 'max_depth': 17}.

```
[ ]  #split 2 sets 8:2
     X = df_dum.drop(columns=['satisfaction'])
     y = df_dum[['satisfaction']]

     train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
[ ]  param_space = {
         'max_depth':list(range(2, 20)),
         'min_samples_leaf':list(range(1, 30)),
         'min_samples_split':list(range(2, 20))}
```

```
[ ]  randomsearch = RandomizedSearchCV(DecisionTreeClassifier(random_state=1),param_space,
                                       n_iter=50,scoring="accuracy", cv=10, n_jobs=-1,return_train_score=True)
     randomsearch.fit(train_X, train_y)
```

```
[ ]  print("Optimal hyperparameter combination:", randomsearch.best_params_)
     print()
     print("Mean cross-validated training accuracy score:",
           randomsearch.best_score_)
```

```
Optimal hyperparameter combination: {'min_samples_split': 15, 'min_samples_leaf': 3, 'max_depth': 15}

Mean cross-validated training accuracy score: 0.9426970756232096
```

Optimal Decision Tree model was created to fit into the training dataset.

```
[ ]  randomsearch.best_estimator_.fit(train_X, train_y)
     y_pred = randomsearch.best_estimator_.predict(test_X)
```

```
[ ]  besttree=randomsearch.best_estimator_
```

```
[ ]  classificationSummary(test_y, y_pred)
```

```
Confusion Matrix (Accuracy 0.9476)

       Prediction
Actual     0     1
     0 11256   548
     1   814 13358
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))

     Accuracy: 0.9475669849091469
     Precision: 0.9605925499784266
     Recall: 0.9425627998871013
     F1 Score: 0.9514922715293112
     RMSE:  0.2289825650368453
```

The accuracy and F1 score are high in predicting the satisfaction outcome.

12.0. Random Forest

Random forest is composed of many randomly generated decision trees with low correlation, and the result of classification prediction will be more accurate.

```
[ ]  rf = RandomForestClassifier(random_state=1, n_estimators=500, max_depth=None,  min_samples_split=2)
     rf.fit(train_X, train_y)
```

```
[ ]  y_pred = rf.predict(test_X)
```

```
[ ]  classificationSummary(test_y, y_pred)

     Confusion Matrix (Accuracy 0.9600)

            Prediction
     Actual     0     1
          0 11404   400
          1   638 13534
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))

     Accuracy: 0.9600400369571912
     Precision: 0.9712932395579159
     Recall: 0.9549816539655659
     F1 Score: 0.9630683839749521
     RMSE:  0.1998999825482615
```

The accuracy and F1 score are higher than that under optimal Decision Tree model.

13.0. Logistic Regression

Logistic Regression can estimate the probability of customer satisfaction or dissatisfaction and thus predict whether the outcome is satisfied or not.

```
[ ]  from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
     from sklearn.metrics import accuracy_score, roc_curve, auc
     from sklearn.model_selection import RepeatedStratifiedKFold
     !pip install scikit-plot
     import scikitplot as skplt
     from scipy.stats import loguniform
     !pip install dmba
     from dmba import regressionSummary, classificationSummary, liftChart, gainsChart, adjusted_r2_score,
                     exhaustive_search, backward_elimination, forward_selection, AIC_score, BIC_score
```

```
[ ]  lr=LogisticRegression()

[ ]  #repeated stratified k-fold cross-validation
     cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
```

Based on random search, the optimal hyperparameter combination is {'C': 0.030673510835226868, 'penalty': 'l2', 'solver': 'newton-cg'}.

```
[ ]  param_space = dict()
     param_space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']
     param_space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']
     param_space['C'] = loguniform(1e-5, 100)

[ ]  randomsearch = RandomizedSearchCV(lr, param_space, n_iter=50, scoring='accuracy',
                                       n_jobs=-1, cv=cv, random_state=1)
     randomsearch.fit(train_X, train_y)
```

```
[ ]  print("Optimal hyperparameter combination:", randomsearch.best_params_)
     print()
     print("Mean cross-validated training accuracy score:",
           randomsearch.best_score_)

     Optimal hyperparameter combination: {'C': 0.030673510835226868, 'penalty': 'l2', 'solver': 'newton-cg'}

     Mean cross-validated training accuracy score: 0.8352100601483594
```

Optimal Logistic Regression model was created to fit into the training dataset.

```
[ ]  randomsearch.best_estimator_.fit(train_X, train_y)
     y_pred = randomsearch.best_estimator_.predict(test_X)
```

```
[ ]  logistic=randomsearch.best_estimator_
```

```
[ ]  regressionSummary(test_y, y_pred)
```

```
    Regression statistics

                Mean Error (ME) : -0.0027
    Root Mean Squared Error (RMSE) : 0.4023
         Mean Absolute Error (MAE) : 0.1618
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
    Accuracy: 0.8381582999692023
    Precision: 0.8499508495997753
    Recall: 0.8541490262489416
    F1 Score: 0.8520447666643205
    RMSE:  0.4022955381691396
```

Logistic regression performed not as well as optimal Decision Tree and Random Forest.

14.0. Neural Network

MLPClassifier is one of the Neural Networks used for classification.

```
[ ]  from sklearn.neural_network import MLPClassifier
     from sklearn.preprocessing import MinMaxScaler
     from sklearn import preprocessing
```

Before training Neural Network, data sets should be normalized, because variables are in different units and have different data amplitudes. Data is converted to 0 to 1 by MinMaxScaler normalization.

```
[ ]  #Normalize
     scaler = preprocessing.MinMaxScaler()
     names = df_dum.columns
     scaled_df_dum = pd.DataFrame(scaler.fit_transform(df_dum), columns=names)
     scaled_df_dum.head()
```

```
[ ]  X = scaled_df_dum.drop(columns=['satisfaction'])
     y = scaled_df_dum[['satisfaction']]

     train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=1)
```

The best hidden_layer_sizes obtained by Grid search is 15.

```
[ ]  param_grid = {'hidden_layer_sizes':list(range(1, 20))}
```

```
[ ]  gridsearch=GridSearchCV(MLPClassifier(activation='logistic',solver='lbfgs',
                                           random_state=1,max_iter=500),
                             param_grid=param_grid,cv=5,n_jobs=-1,return_train_score=True)
     gridsearch.fit(train_X,train_y)
```

```
[ ]  print("Optimal hyperparameter", gridsearch.best_params_)
     print()
     print("Mean cross-validated training accuracy score:",
           gridsearch.best_score_)
```

```
     Optimal hyperparameter {'hidden_layer_sizes': 15}

     Mean cross-validated training accuracy score: 0.94565175516856
```

Optimal Neural Network model was created to fit into the training dataset.

```
[ ]  gridsearch.best_estimator_.fit(train_X, train_y)
     y_pred = gridsearch.best_estimator_.predict(test_X)
```

```
[ ]  nn=gridsearch.best_estimator_
```

```
[ ]  classificationSummary(y,nn.predict(X))
```

```
     Confusion Matrix (Accuracy 0.9437)

            Prediction
     Actual     0     1
          0 55172  3621
          1  3686 67401
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9448721897135818
Precision: 0.951005380911923
Recall: 0.9477843635337285
F1 Score: 0.9493921402318349
RMSE:  0.23479312231498228
```

Neural Network has higher accuracy and F1 score than Logistic Regression, but still less than Random Forest.

15.0. KNN

The KNeighborsClassifier algorithm assumes that similar things are close, and uses proximity to classify groups of individual data points (Harrison, 2018).

```
[ ]  from sklearn.neighbors import KNeighborsClassifier

[ ]  knn = KNeighborsClassifier()
```

The dataset feed in this model is also scaled data, same as Neural Network. The best n_neighbors obtained by Grid search is 5.

```
[ ]  param_grid = {'n_neighbors':list(range(1, 30))}

[ ]  gridsearch = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy',
                               return_train_score=False,verbose=1)
     gridsearch.fit(train_X,train_y)
```

```
[ ]  print("Optimal hyperparameter", gridsearch.best_params_)
     print()
     print("Mean cross-validated training accuracy score:",
           gridsearch.best_score_)
```

```
Optimal hyperparameter {'n_neighbors': 5}

Mean cross-validated training accuracy score: 0.9250942544952417
```

Optimal KNN model was created to fit into the training dataset.

```
[ ]  gridsearch.best_estimator_.fit(train_X, train_y)
     y_pred = gridsearch.best_estimator_.predict(test_X)
```

```
[ ]  knn=gridsearch.best_estimator_
```

```
[ ]  classificationSummary(y,knn.predict(X))

     Confusion Matrix (Accuracy 0.9445)

            Prediction
     Actual     0     1
          0 56222  2571
          1  4637 66450
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))

     Accuracy: 0.9290498922081922
     Precision: 0.9518434361943854
     Recall: 0.9163138583121648
     F1 Score: 0.9337407873449577
     RMSE:  0.266364614376249
```

KNN has lower accuracy and F1 score than that in Neural Network and Random Forest.

16.0. Max Voting Ensemble

The max voting ensemble model combines multiple algorithms, where each model makes a prediction and a vote, and the prediction with the most votes will be the final prediction, which is applied to the model of the prediction class label (Brownlee, How to Develop Voting Ensembles With Python, 2020). The advantage of this model is that it can combine various algorithms to improve the prediction accuracy.

```
[ ]  from numpy import mean
     from numpy import std
     from sklearn import model_selection
     from sklearn.datasets import make_classification
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import RepeatedStratifiedKFold
     from sklearn.ensemble import VotingClassifier
     from matplotlib import pyplot
```

I combined all five of the previous models into the Max Voting Model, including Optimal Decision Tree, Random Forest, Logistic Regression, Neural Network and KNN.

```
[ ]    model_deicision_tree = besttree
       model_random_forest_ = rf
       model_logistic_regression = logistic
       model_neural_network = nn
       model_knn = knn
```

```
[ ]  models = []
     models.append(('deicision_tree', besttree))
     models.append(('random_forest', rf))
     models.append(('logistic_regression', logistic))
     models.append(('neural_network', nn))
     models.append(('knn', knn))
```

Voting Model was created to fit into the training dataset.

```
[ ]  ensemble = VotingClassifier(
         estimators=models, voting='hard')
```

```
[ ]  ensemble.fit(train_X, train_y)
```

```
[ ]  y_pred = ensemble.predict(test_X)
```

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9554203880505081
Precision: 0.9650514579759862
Recall: 0.9527942421676545
F1 Score: 0.9588836812952706
RMSE:  0.21113884519313786
```

The Max Voting model has a high accuracy and F1 score, but still not performed as well as random forest.

17.0. Model Comparison

Multiple statistics were calculated under each model. All models have high accuracy and F1 score except for logistic regression. Random forest has the highest accuracy and F1 score.

besttree(decision tree with CV)

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9475669849091469
Precision: 0.9605925499784266
Recall: 0.9425627998871013
F1 Score: 0.9514922715293112
RMSE:  0.2289825650368453
```

random forest

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9600400369571912
Precision: 0.9712932395579159
Recall: 0.9549816539655659
F1 Score: 0.9630683839749521
RMSE:  0.1998998825482615
```

logistic regression

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.8381582999692023
Precision: 0.8499508495997753
Recall: 0.8541490262489416
F1 Score: 0.8520447666643205
RMSE:  0.4022955381691396
```

neural network

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9448721897135818
Precision: 0.951005380911923
Recall: 0.9477843635337285
F1 Score: 0.9493921402318349
RMSE:  0.23479312231498228
```

KNN

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9290498922081922
Precision: 0.9518434361943854
Recall: 0.9163138583121648
F1 Score: 0.9337407873449577
RMSE:  0.266364614376249
```

voting ensemble

```
[ ]  print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
     print("Precision:",metrics.precision_score(test_y, y_pred, average="binary"))
     print("Recall:",metrics.recall_score(test_y, y_pred, average="binary"))
     print("F1 Score:",metrics.f1_score(test_y, y_pred, average="binary"))
     print("RMSE: ", np.sqrt(mean_squared_error(test_y, y_pred)))
```

```
Accuracy: 0.9554203880505081
Precision: 0.9650514579759862
Recall: 0.9527942421676545
F1 Score: 0.9588836812952706
RMSE:  0.21113884519313786
```

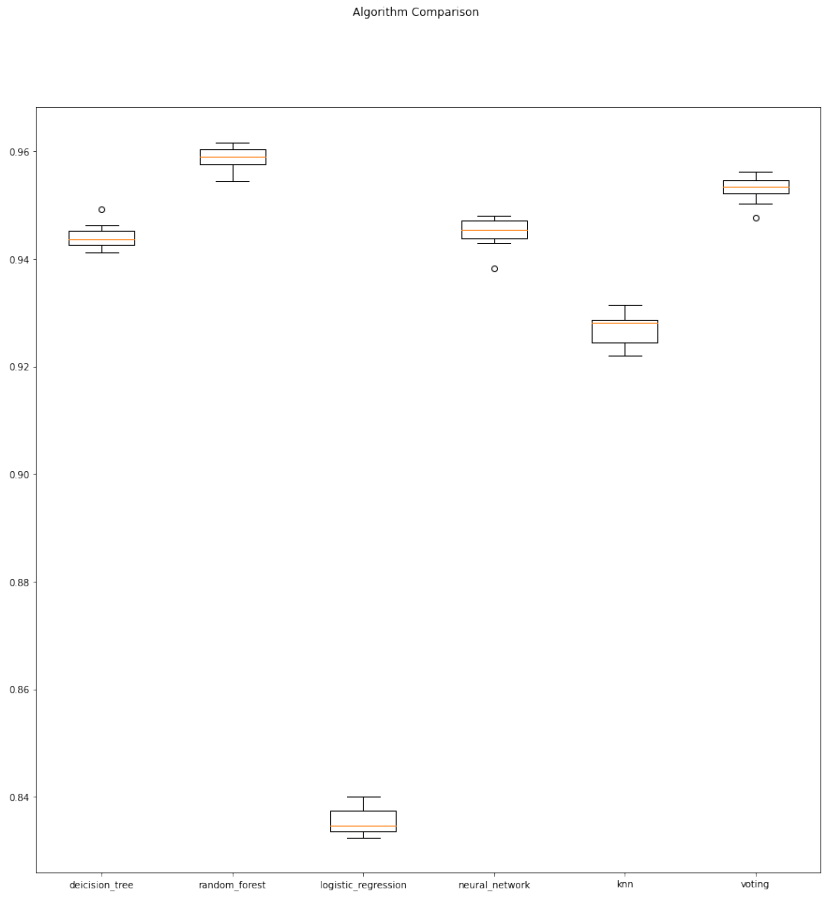
Model Recommendation

18.0 Model Selection

```
[ ]  models = []
     models.append(('deicision_tree', besttree))
     models.append(('random_forest', rf))
     models.append(('logistic_regression', logistic))
     models.append(('neural_network', nn))
     models.append(('knn', knn))
     models.append(('voting', ensemble))
```

```
[ ]  seed=5
     results = []
     names = []
     scoring = 'accuracy'
     for name, model in models:
       kfold = model_selection.KFold(n_splits=10, random_state=seed,shuffle=True)
       cv_results = model_selection.cross_val_score(model, X, y, cv=kfold, scoring=scoring)
       results.append(cv_results)
       names.append(name)
       msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
       print(msg)
```

```
[ ]  fig = plt.figure(figsize=(15,15))
     fig.suptitle('Algorithm Comparison')
     ax = fig.add_subplot(111)
     plt.boxplot(results)
     ax.set_xticklabels(names)
     plt.show()
```



According to the boxplot comparing between models, the accuracy of Random Forest is the highest, but the accuracy of Neural Network and voting Ensemble were also similar. However, even if their accuracy is higher than that of Random Forest, Random Forest will still be selected in this case from the perspective of cost. Because both the Neural Network and the Voting Ensemble took a long time to run, which is expensive in computation and resource. In Voting Ensemble, 5 models are needed to maintain and run, which requires more model development.

19.0 Model Assumptions and Limitations

Random forests have no formal distributional assumptions, and it's nonparametric, which makes it able to handle skewed, multimodal, and ordinal data (Richmond, 2016).

Limitations of random forests include the possibility of overfitting with noisy data sets. In the data set with different levels of categorical predictor variables, random forest tends to have more levels of predictor variables, which is biased (Richmond, 2016).

20.0 Model Sensitivity to Key Drivers

In random forests, Gini importance is used to determine the importance of variables. Gini importance calculates the importance of each variable as the sum of the number of splits that contain that variable, proportional to the number of samples it splits (Perrie, 2015).

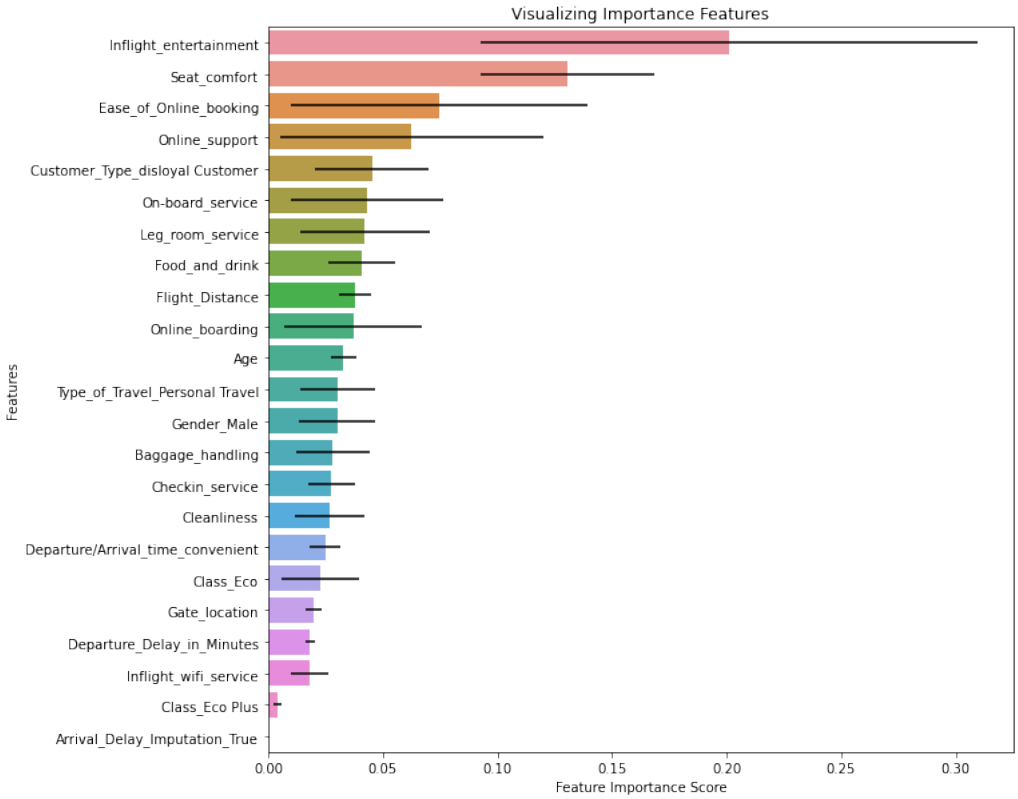
```
[ ]  importance = rf.feature_importances_
     std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
```

```
[ ]  df = pd.DataFrame({'feature': train_X.columns,
                        'importance': importance,
                        'std':std})
     print(df.sort_values('importance', ascending=False))
```

```
                              feature  importance       std
14            Inflight_entertainment    0.201031  0.108629
9                      Seat_comfort    0.130374  0.037851
16           Ease_of_Online_booking    0.074840  0.064634
15                   Online_support    0.062625  0.057308
1    Customer_Type_disloyal Customer    0.045240  0.024824
17                 On-board_service    0.043111  0.033255
18                 Leg_room_service    0.042146  0.028275
11                   Food_and_drink    0.040671  0.014651
8                     Flight_Distance    0.037961  0.007183
22                   Online_boarding    0.037190  0.029939
7                                Age    0.032896  0.005593
4     Type_of_Travel_Personal Travel    0.030330  0.016054
0                        Gender_Male    0.030175  0.016649
19                 Baggage_handling    0.028187  0.016077
20                  Checkin_service    0.027699  0.010033
21                      Cleanliness    0.027071  0.015153
10  Departure/Arrival_time_convenient    0.024872  0.006556
2                         Class_Eco    0.022732  0.017078
12                    Gate_location    0.019955  0.003458
6         Departure_Delay_in_Minutes    0.018343  0.001857
13           Inflight_wifi_service    0.018290  0.008179
3                     Class_Eco Plus    0.004048  0.001888
5      Arrival_Delay_Imputation_True    0.000215  0.000092
```

```
[ ]  feature_imp=pd.Series(rf.feature_importances_,index=train_X.columns).sort_values(ascending=False)
     tf = pd.DataFrame({'feature': train_X.columns,'importance': rf.feature_importances_,'std':std})
     tf = tf.sort_values('importance',ascending = False)

     fig = plt.figure(figsize=(10,10))
     sns.barplot(x = tf['importance'], y=feature_imp.index, xerr = tf['std'])
     plt.xlabel('Feature Importance Score')
     plt.ylabel('Features')
     plt.title('Visualizing Importance Features')
```



Based on the importance of features plot, inflight entertainment, set comfort, ease of online booking, online support and so on are identified to be key drivers of customer satisfaction. The company could concentrate on these important features to improve customer experience by adopting targeted actions.

Validation and Governance

21.0. Variable Level Monitoring

```
df.describe()
```

	Age	Flight_Distance	Seat_comfort	Departure/Arrival_time_convenient	Food_and_drink	Gate_location	Inflight_wifi_service	Inflight_entertainment	Online_support
count	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000
mean	39.427957	1981.409055	2.838597	2.990645	2.851994	2.990422	3.249130	3.383477	3.519703
std	15.119360	1027.115606	1.392983	1.527224	1.443729	1.305970	1.318818	1.346059	1.306511
min	7.000000	50.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	27.000000	1359.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	3.000000
50%	40.000000	1925.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000
75%	51.000000	2544.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000
max	85.000000	6951.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

Ease_of_Online_booking	On-board_service	Leg_room_service	Baggage_handling	Checkin_service	Cleanliness	Online_boarding	Departure_Delay_in_Minutes	Arrival_Delay_in_Minutes
129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129880.000000	129487.000000
3.472105	3.465075	3.485902	3.695673	3.340807	3.705759	3.352587	14.713713	15.091129
1.305560	1.270836	1.292226	1.156483	1.260582	1.151774	1.298715	38.071126	38.465650
0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2.000000	3.000000	2.000000	3.000000	3.000000	3.000000	2.000000	0.000000	0.000000
4.000000	4.000000	4.000000	4.000000	3.000000	4.000000	4.000000	0.000000	0.000000
5.000000	4.000000	5.000000	5.000000	4.000000	5.000000	4.000000	12.000000	13.000000
5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	1592.000000	1584.000000

The acceptable range can be determined based on this historical data. The 14 Ordinal data including Food and drink, Gate Location, Inflight Entertainment, Online support and so on still need to be 0-5 scale.

The age of passengers should be around 10 to 95, ensuring that they are able to complete the survey independently. The unit of Flight Distance is not indicated in this data, and the new data should remain the same unit and similar range. Delays in departure and arrival times should remain in minutes.

If there is a loss of data, KNN method can be used to impute. For values out of tolerance, cap and floor can be used to transform outlier. The box - cox variables used for the transformation of highly skewed variable.

22.0. Model Health & Stability & Drift

Various methods can be adopted to monitor model's robustness and AUC of a ROC curve is preferred for classification model.

```
from sklearn.metrics import roc_auc_score
roc_auc_score(test_y, y_pred)
```

Prediction score of area under the receiver operating characteristics curve of each model was calculated as below:

Models	roc_auc_score
Decision Tree	0.9447
Random Forest	0.9605
Logistic Regression	0.8366
Neural Network	0.9446
KNN	0.9303
Voting Ensemble	0.9556

After comparison, the recommended model random forest still has the highest roc_auc_score.

The model performance could get worse overtime and the accuracy should be lower if more recent data was fit in this model. Since this dataset is from two years ago and there is no more recent dataset, a synthetic dataset can be created to rescore for identifying the drift. If I got more time, I will use “random.normal” from numpy to create new synthetic test data. Only numerical variable will be changed randomly, including “Flight distance”, “Arrival Delay in Minutes” and “Departure Delay in Minutes”, and other features will be left unchanged.

Based on the rescoring result of synthetic data, different actions will be required under different situations.

- If the accuracy and f1 score didn’t decrease a lot, I will be confident about this model. There is minimal risk and no action is needed.
- If the accuracy and f1 score got lower, the drift should be a problem. A more robust model should be built or the company is required to keep monitor model and rebuild or refresh model overtime.

23.0. Initial Model Fit Statistics

23.1. Accuracy

Accuracy is used as one parameter, since the "satisfied" and "dissatisfied" categories both are important in this case. Accuracy is a common measure for the correct classification of models. Accuracy metrics are easy to understand but not applicable to imbalanced data.

Accuracy = (True Positive + True Negative) / (True Positive + True Negative + False Positive + False Negative)

23.2. F1-score

As another indicator, F1-score highlights the misclassification of the model. F1-score works even when the data is not balanced.

F1-score = 2*(Precision*Recall) / (Precision + Recall)

24.0. Risk Tiering

This data is updated before pandemic, which is mainly about passengers' subjective evaluation of various services and experiences in the journey. The model based on historical data has room for error when applied to future forecasts.

Airlines should pay special attention to the costs of false positive and false negative. If a group of passengers are satisfied with the airline, but are predicted to be dissatisfied, the company may need to spend unnecessary resources and time to change the unsatisfactory status. If a group of passengers are unhappy with the airline but are predicted to be satisfied, the company may not notice them and may lose the group of customers forever.

In the aftermath of the pandemic, people's travel frequency and consumption habits may have changed significantly, as will their subjective feedback before, during and after a

flight. The proportion of people traveling for purposes may change, for example, the proportion of personal travel increases significantly after the pandemic.

Risk grades that affect model performance and company decisions can be classified as limited or high. The company needs to monitor and refit model overtime.

Conclusion and Recommendations

25.0. Impacts on Business Problem

According to the previous analysis, the Random Forest trained on these historical data is the most effective model in terms of prediction accuracy. This model can help the company to predict the future customer satisfaction results with high reliability. It helps companies attract, retain and cultivate valuable customers. In addition, according to the variable importance obtained from this model, targeted adjustment and improvement of the current situation can help the company avoid the waste of resources and maximize benefits with as little cost as possible.

The main variables affecting customer satisfaction include in-flight entertainment, seat comfort, ease of booking online, online support, etc. Providing more varied in-flight entertainment programs and upgrading the entertainment system facilities will be the first action taken by the company. Providing items or services that help passengers more comfortable can help increase satisfaction with seat comfort. Upgrading and improving existing online booking systems and services will also greatly improve customers' consumption experience. These important variables reflect the consumption demands of customers. Only by trying to meet these needs, more customers will be retained and long-term benefits will be achieved.

26.0. Recommended Next Steps

In order to make the model more representative and accurate, more and updated data need to be collected continuously, and new data can be used to test and improve model.

For continuously improving customer satisfaction rate and achieve profit growth, more analysis of different customer groups can be carried out. For example, key factors affecting the satisfaction of disloyal customers can be analyzed, focusing on improving their concerns and attracting more customers to become loyal customers, which is conducive to the company's long-term profit. After the COVID-19 pandemic, personal travel to visit friends and relatives has a greater growth trend and space than business travel. Focusing on analyzing the influencing factors of personal travel satisfaction and making targeted improvements will play an important role in catering to specific consumer groups in the short term and realizing rapid profits

References

Bouwer, J., Saxon, S., & Wittkamp, N. (2021, April 2). *Back to the future? Airline sector poised for change post-COVID-19*. Retrieved from McKinsey & Company: https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/back-to-the-future-airline-sector-poised-for-change-post-covid-19

Brownlee, J. (2020, April 17). *How to Develop Voting Ensembles With Python*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/voting-ensembles-with-python/

Brownlee, J. (2020, September 4). *Hyperparameter Optimization With Random Search and Grid Search*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

DISTANCE CALCULATOR. (2022). Retrieved from DISTANCE CALCULATOR: https://distancecalc.com/how-far-from-toronto-canada-to-kitchener-canada

Harrison, O. (2018, September 10). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Retrieved from Medium: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Jana, S. (2020). *Kaggle*. Retrieved from Airlines Customer satisfaction: https://www.kaggle.com/datasets/sjleshrac/airlines-customer-satisfaction?select=Invistico_Airline.csv

Obadia, Y. (2017, Janaury 31). *The use of KNN for missing values*. Retrieved from Medium: https://towardsdatascience.com/the-use-of-knn-for-missing-values-

cf33d935c637#:~:text=Why%20using%20KNN%20%3F,all%20kind%20of%20missing%20data.

Perrie, A. (2015, August 27). *Feature Importance in Random Forests*. Retrieved from Alexis Perrier - Data Science: https://alexisperrier.com/datascience/2015/08/27/feature-importance-random-forests-gini-accuracy.html

Richmond, S. (2016, Mar 21). *Algorithms Exposed: Random Forest*. Retrieved from bccvl: https://bccvl.org.au/algorithms-exposed-random-forest/

statista. (2022, July 12). *Major airlines' domestic market share in Canada in 2020**. Retrieved from statista: https://www-statista-com.centennial.idm.oclc.org/statistics/545642/air-carrier-canada-domestic-market-share/