COMS-W4995 AML Final Project Deliverable #2

Group #25: Longyuan Gao (lg3230), Ken Xiong (kx2175), Samya Ahsan (ska2138)

Initial Data Exploration

Some preliminary findings (on train & test combined dataset):

1. There are 24 feature columns, and the target is the “satisfaction” column;
2. The features are all floats and ints except for "Gender", "Customer Type", "Type of Travel", and "Class", as they are objects;
3. The target column, "satisfaction", is an object;

Imbalanced Dataset?

1. We have an imbalanced dataset**,** since we have 73452 observations with class "neutral or dissatisfied" and 56428 "satisfied" observations， as shown in fig 1 below

Missing Values?

1. The only column with missing values is “Arrival Delay in Minutes” column with 393 missing values (but later we will drop this column);

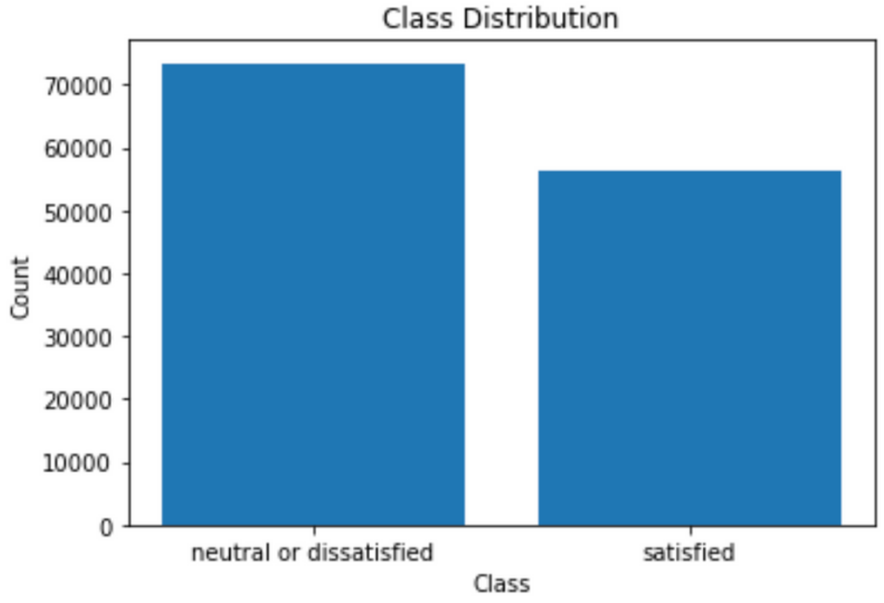
Highly Correlated Columns?

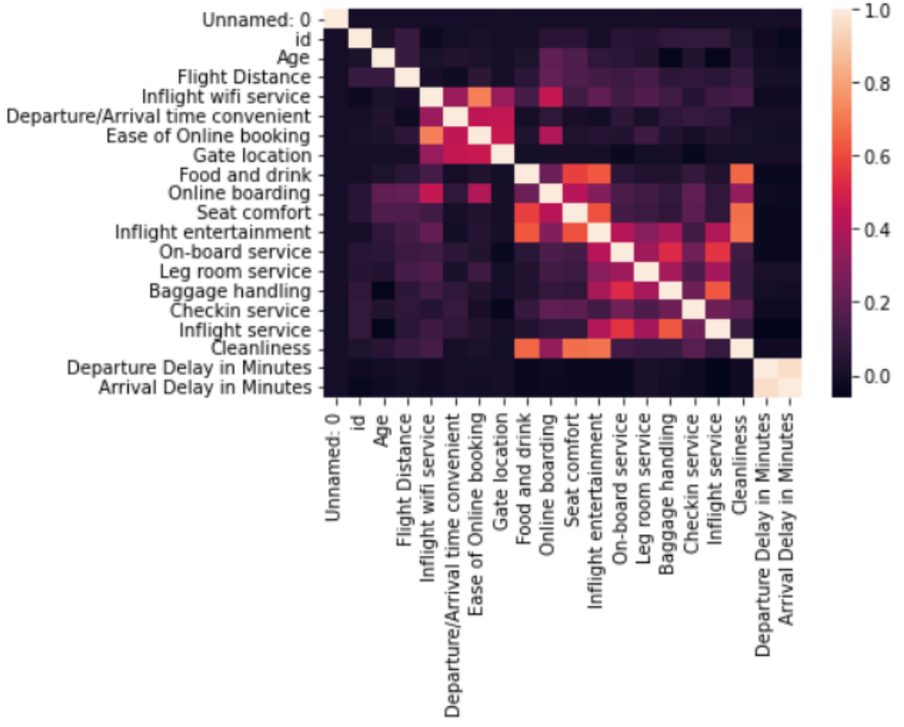
1. For numeric features, if we set the threshold to be 0.90, then The only highly correlated columns are "Departure Delays in Minutes" and "Arrival Delay in Minutes" with coefficient 0.965, as shown in fig2;

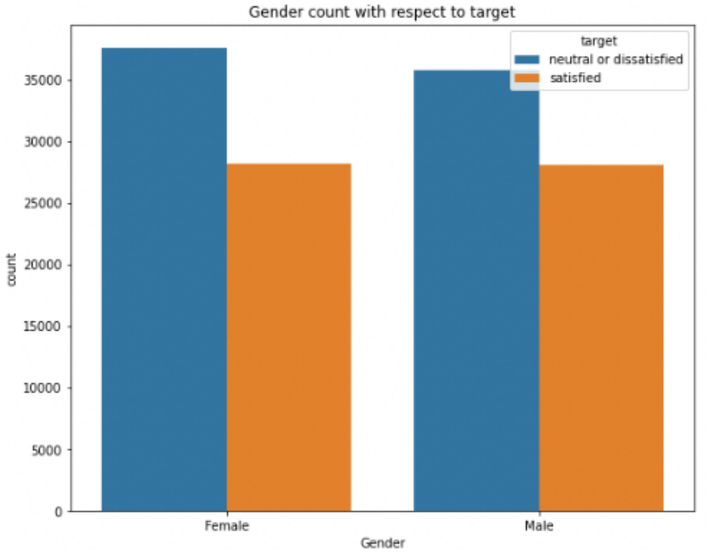
Categorical Features (fig 3)?

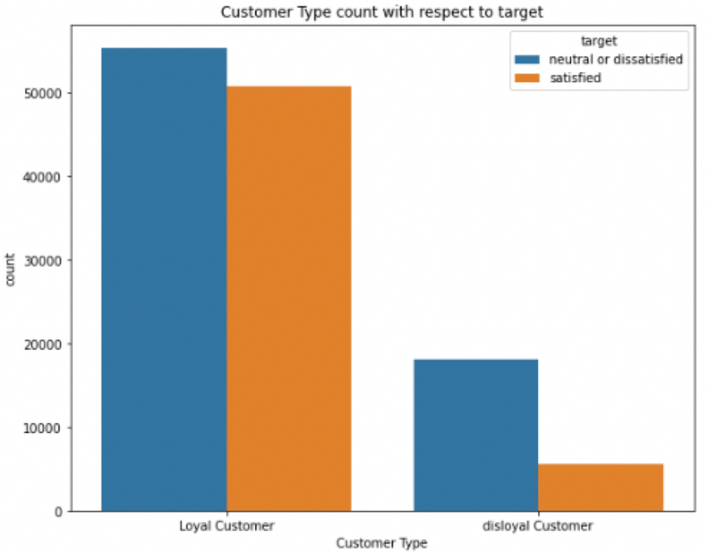
1. For *Gender* column, there are more female observations than male observations, and for both genders, there are more answers with 0 (neutral or dissatisfied) than answers with 1 (satisfied).

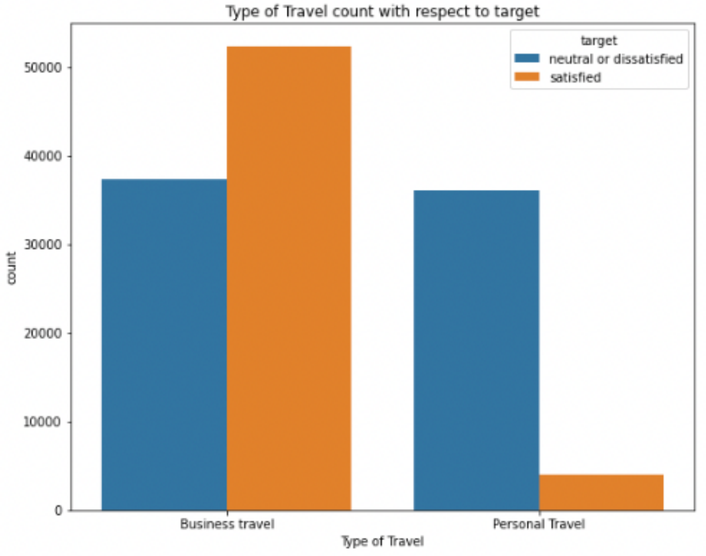
For "*Customer Type*" column, there are considerably more observations with loyal customers than disloyal customers, and again, more observations are with answer 0.
For "T*ype of Travel*" column, more observations are made with business travel than personal travel, and for business travel, 1 answers outnumber 0 answers, but the opposite is true for personal travels.
For *Class* column, eco plus is the class with least number of observations; the other two classes are with considerably more observations. For eco plus and eco classes, more 0 observations are made, and the opposite is true for business class.

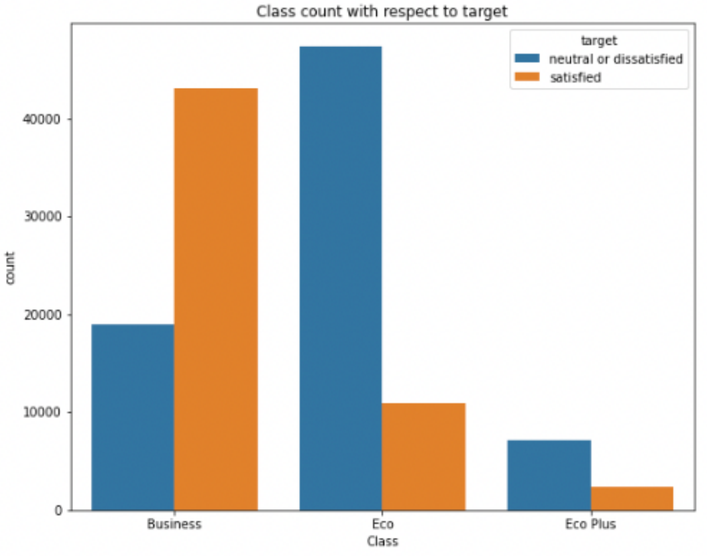

Class Distribution
Count
70000
60000
50000
40000
30000
20000
10000
0
neutral or dissatisfied
satisfied
Class


Unnamed: 0
id
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
1.0
0.8
0.6
0.4
0.2
0.0


Gender count with respect to target
target
neutral or dissatisfied
satisfied
count
Female
Male
Gender


Customer Type count with respect to target
target
neutral or dissatisfied
satisfied
count
Loyal Customer
disloyal Customer
Customer Type


Type of Travel count with respect to target
target
neutral or dissatisfied
satisfied
count
Business travel
Personal Travel
Type of Travel


Class count with respect to target
target
neutral or dissatisfied
satisfied
count
Business
Eco
Eco Plus
Class

PREPROCESSING

- Encoding categorical data using **ordinal encoding**
  - Many of the features have an inherent ordering, such as that present in the “Class” feature
- **Scaling** features using StandardScaler
  - Normalize data set such that no feature dominates
- **Dropping** irrelevant columns
  - id
  - Unnamed 0th column

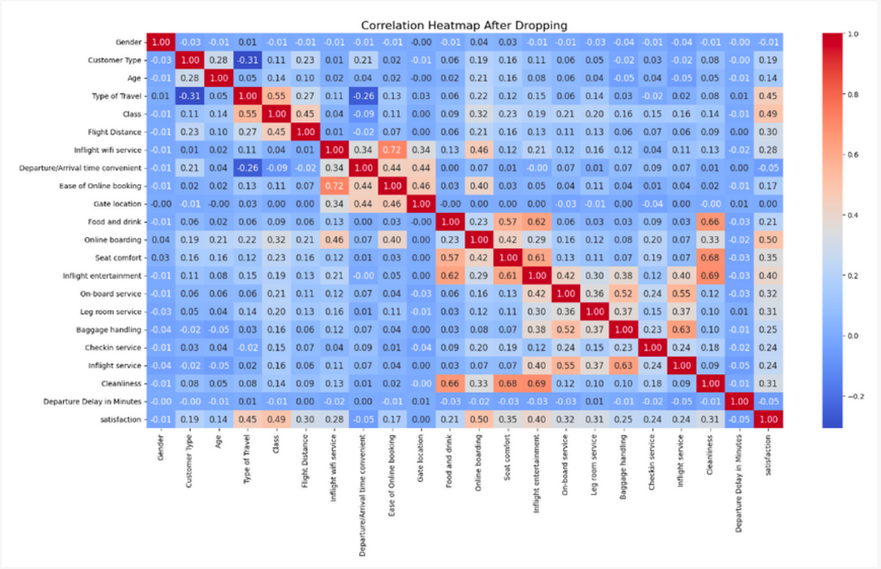
```
[1] import numpy as np
    import pandas as pd
```

```
[2] #load dataset
    train = pd.read_csv('train.csv')
    test = pd.read_csv('test.csv')
```

```
from sklearn.preprocessing import OrdinalEncoder
gender_encoder = OrdinalEncoder(categories = [["Male", "Female"]])
train['Gender'] = gender_encoder.fit_transform(train['Gender'].to_numpy().reshape(-1, 1))

customer_encoder = OrdinalEncoder(categories=[["disloyal Customer", "Loyal Customer"]])
train['Customer Type'] = customer_encoder.fit_transform(train['Customer Type'].to_numpy().reshape(-1, 1))

travel_encoder = OrdinalEncoder(categories=[["Personal Travel", "Business travel"]])
train['Type of Travel'] = travel_encoder.fit_transform(train['Type of Travel'].to_numpy().reshape(-1, 1))

class_encoder = OrdinalEncoder(categories=[["Eco", "Eco Plus", "Business"]])
train['Class'] = class_encoder.fit_transform(train['Class'].to_numpy().reshape(-1, 1))

satisfaction_encoder = OrdinalEncoder(categories=[["neutral or dissatisfied", "satisfied"]])
train['satisfaction'] = satisfaction_encoder.fit_transform(train['satisfaction'].to_numpy().reshape(-1, 1))

#Now encoding test set
test['Gender'] = gender_encoder.fit_transform(test['Gender'].to_numpy().reshape(-1, 1))
test['Customer Type'] = customer_encoder.fit_transform(test['Customer Type'].to_numpy().reshape(-1, 1))
test['Type of Travel'] = travel_encoder.fit_transform(test['Type of Travel'].to_numpy().reshape(-1, 1))
test['Class'] = class_encoder.fit_transform(test['Class'].to_numpy().reshape(-1, 1))
test['satisfaction'] = satisfaction_encoder.fit_transform(test['satisfaction'].to_numpy().reshape(-1, 1))
```

```
train = train.drop("Arrival Delay in Minutes", axis=1)
test = test.drop("Arrival Delay in Minutes", axis=1)

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

```
from sklearn.preprocessing import StandardScaler
# Normalize dataset
features = ['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']
target = ['satisfaction']

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

X_test = test[features]
y_test = test[target].to_numpy()

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



INSIGHTS FROM DATA EXPLORATION

We have an imbalanced dataset. So, we should better employ methods such as stratified split, over or undersampling methods when doing data preprocessing.

The dataset has highly correlated columns, and we decided to drop one of them.

The missing values are all located in one of the highly correlated columns, so we drop the one with missing values.

For categorical features, the class distribution is not balanced in most subgroups, as we observe in fig 3. For example, there are far more loyal customers than disloyal customers in the customer type column. Therefore, we should keep this in mind when doing data preprocessing and training.

Machine Learning Techniques Proposed to be Implemented

1. Data preprocessing
    a. Encoding categorical features
    b. Identifying highly collinear features
    c. Sampling strategy
    d. Missing value analysis
2. Model Selection
    a. Linear model vs. tree-based model
3. Hyper-parameter Tuning
    a. Grid search
4. Model Performance Evaluation
    a. K-fold cross-validation

Data preprocessing

1. Encoding categorical features
    a. Ordinal encoding
2. Identifying highly collinear features
    a. Simply remove “Arrival Delay in Minutes” to avoid multicollinearity
3. Sampling strategy
    a. Stratified sampling
    b. Create stratum based on “Satisfaction”
4. Missing value analysis
    a. “Arrival Delay in Minutes” is dropped in the previous step. No more missing values

Model Selection and Hyper-parameter Tuning

1. Train one linear model (Logistic regression) and one tree-based model (Random forest)
2. Use k-fold cross-validation where k is 5 to compare model performance
3. We will use f1 score to evaluate model performance and pick the best model
4. We will use grid search to perform hyper-parameter tuning
    a. Intercept, coefficient, and threshold if logistic model is chosen in model selection
    b. Number of trees, maximum depth, and number of features if random forest is chosen in model selection
    c. Performance will be measured using f1 score

Model Performance Evaluation

We will use k-fold cross-validation again to measure the performance of the chosen model after hyper-parameter tuning. The performance is measure using the f1 score metric.