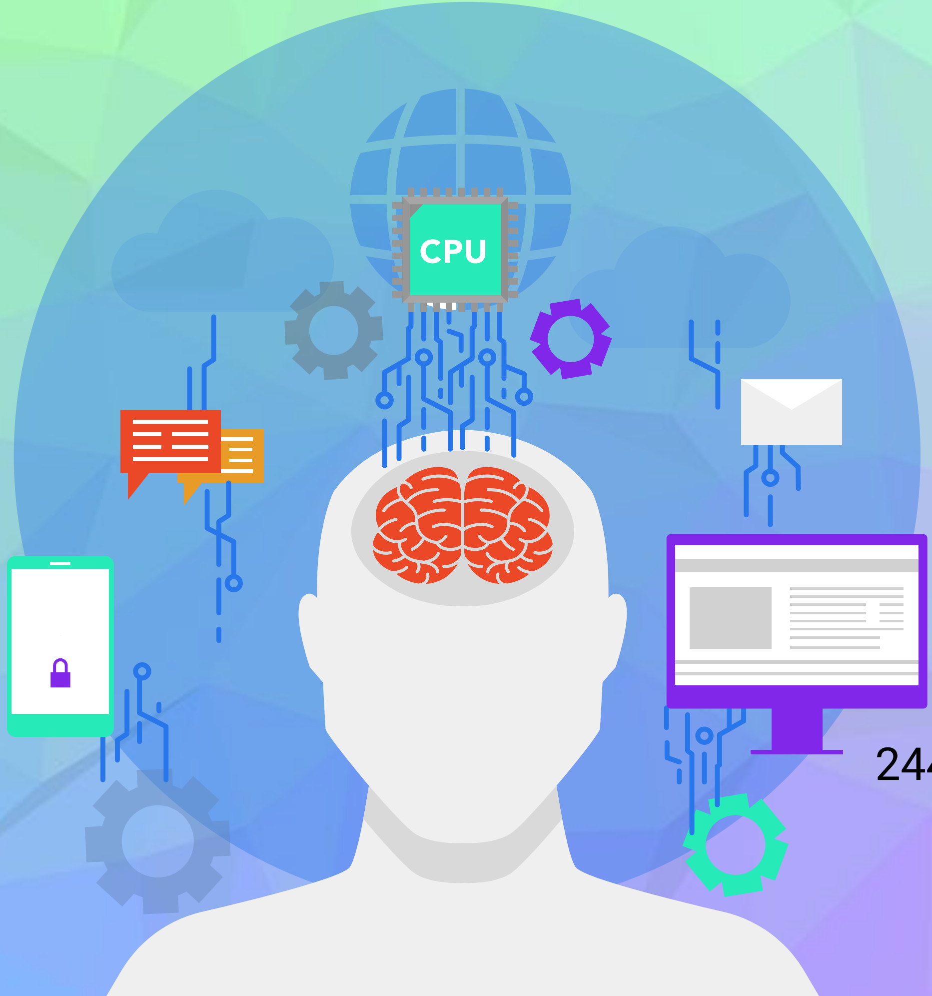
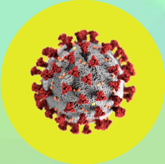


Airline Passenger Satisfaction

2440009654 - Darnell Kikoo
2440009162 - Matthew Adrianus Mulyono
2440076450 - Winson Allan Wijaya

Table of Contents

Problem Definition



COVID-19

Due to the rampant pandemic, the airline industry experienced plummeting profits.

Living Standards

However, living standards have in turn, improved as people took care of themselves more

Demand

These two factors results in airline industries desiring to improve their services.

Data Preparation

Dataset:
Airline Passenger Satisfaction by TJ Klein*

Remove Outliers

We remove columns that we deem as unnecessary

Handle Missing Values

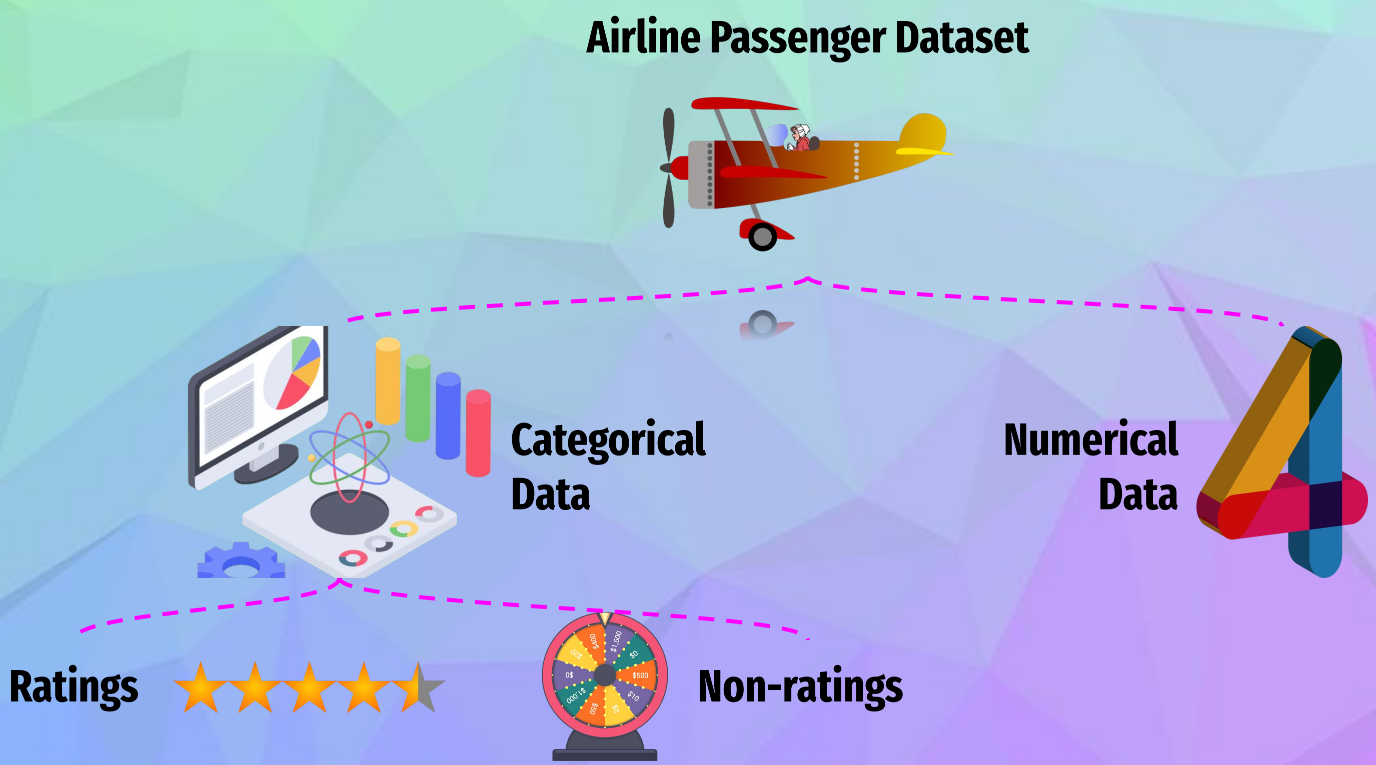
The categorical ones are replaced with its **mode**, while numericals are replaced with its **mean**.

Data Conversion

We convert categorical data to numerical and vice versa.

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

Exploratory Data Analysis



Exploratory Data Analysis

1.

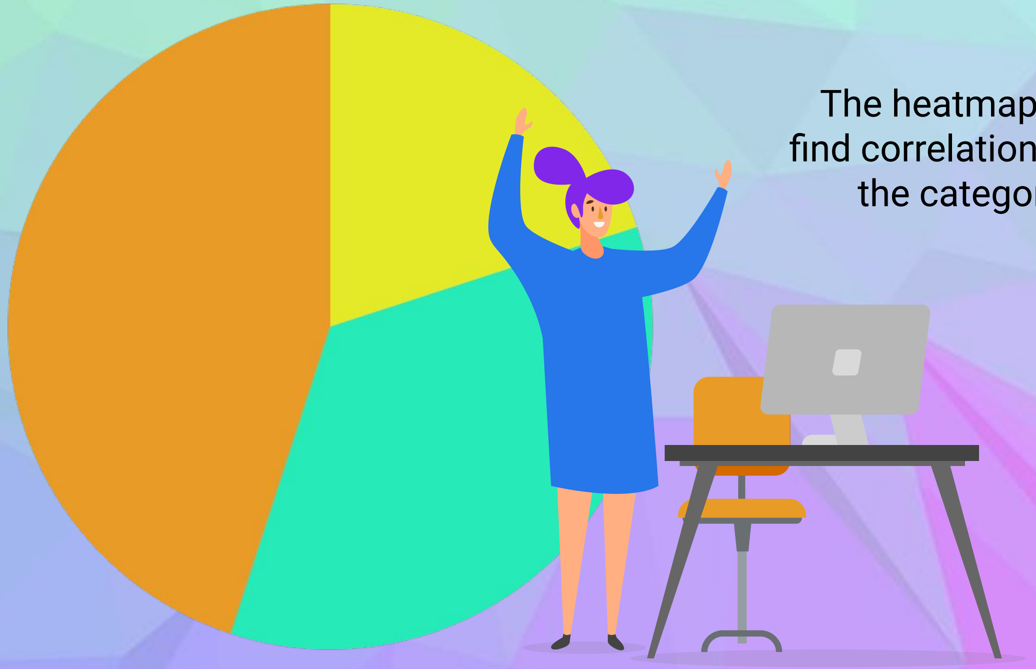
Categorical Label Distribution

We find out the label distribution in both the ratings and non ratings categorical labels using bar graphs

2.

Numerical Label Distribution

This is done using *kdeplot* and *boxplot*



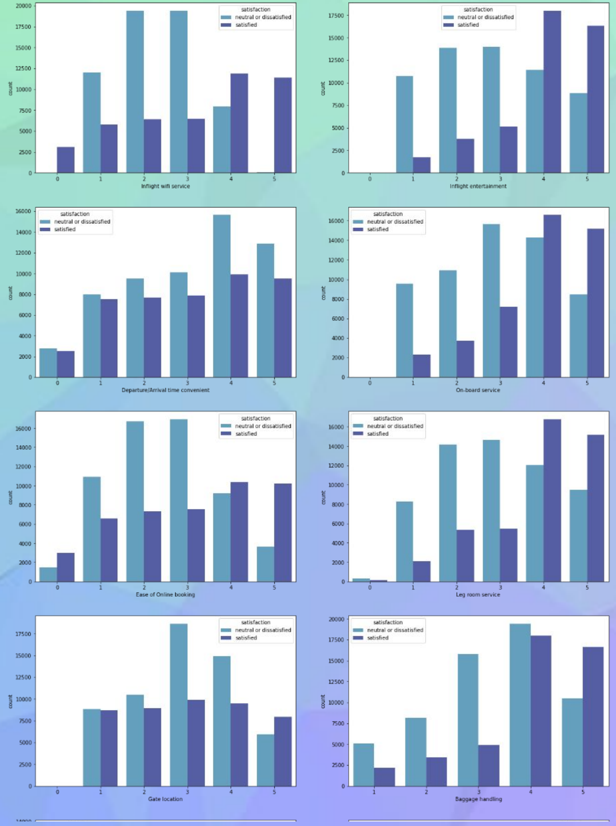
3.

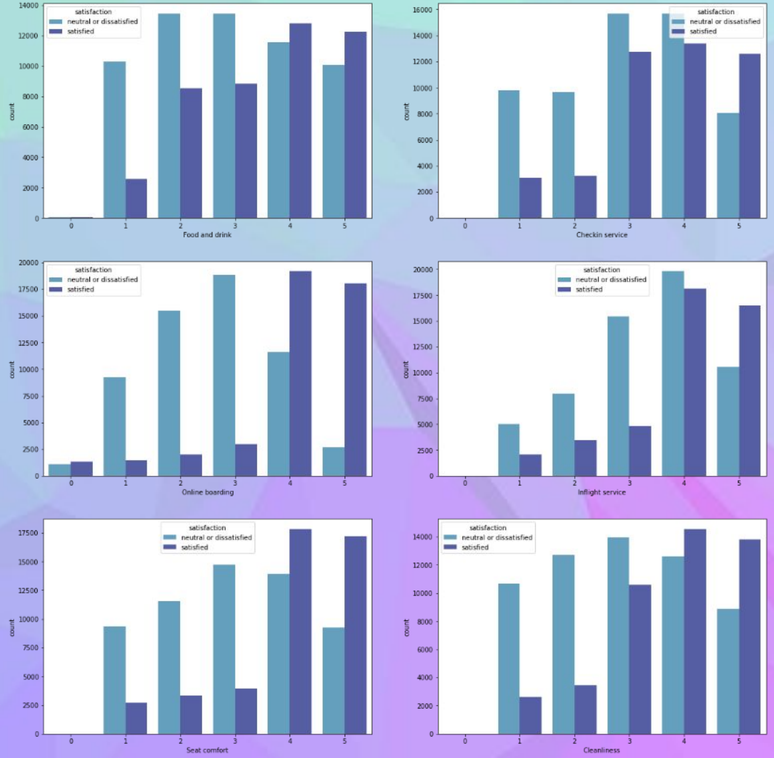
Heatmap

The heatmap is used to find correlations between the categorical rating labels

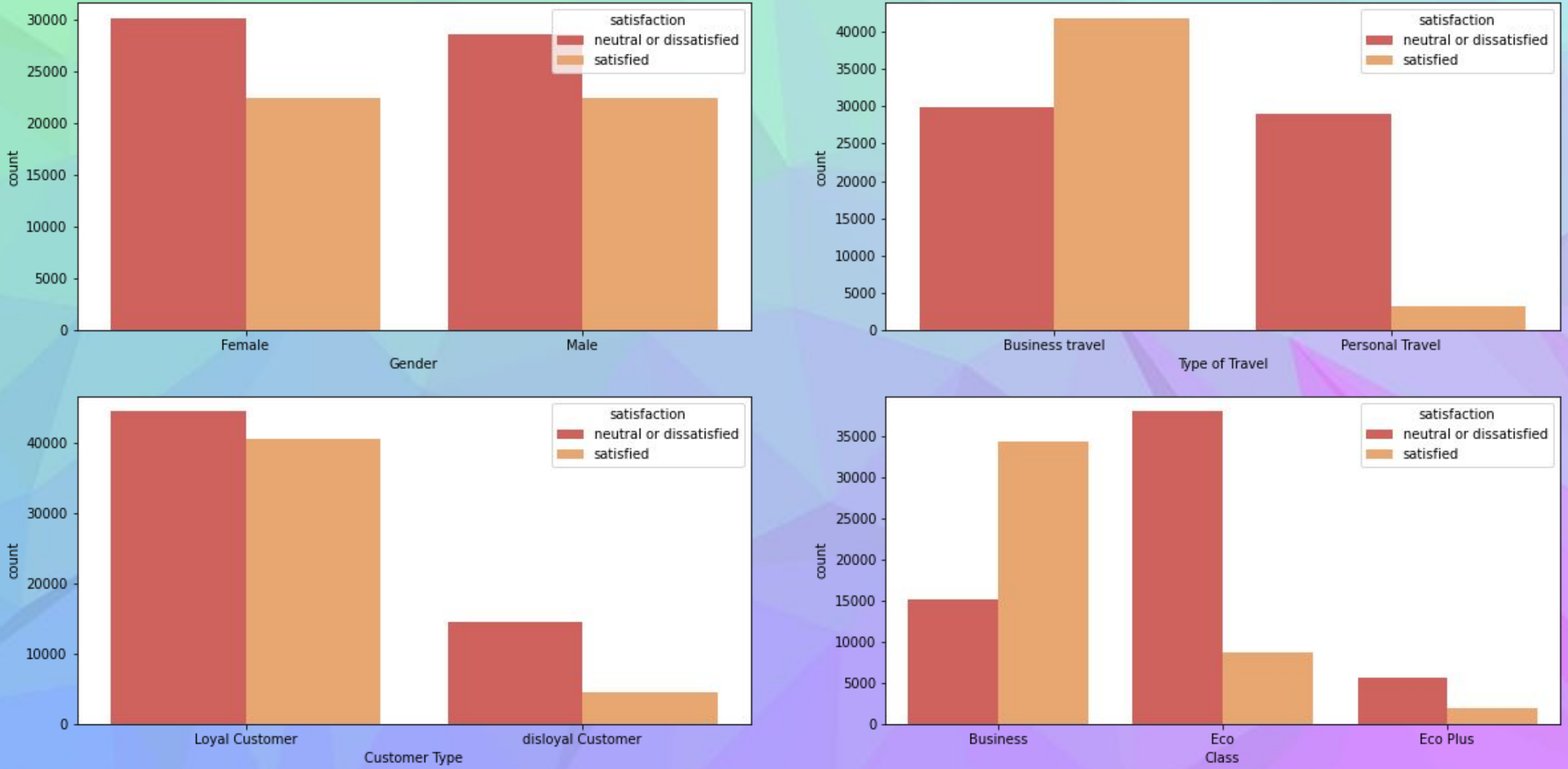
Check out the results @
https://colab.research.google.com/drive/1ozDpPvwTh44hDEafooDxac9SjwNuupHl#scrollTo=_qPqw3_BLsrS

CATEGORICAL LABEL DISTRIBUTION (RATINGS)

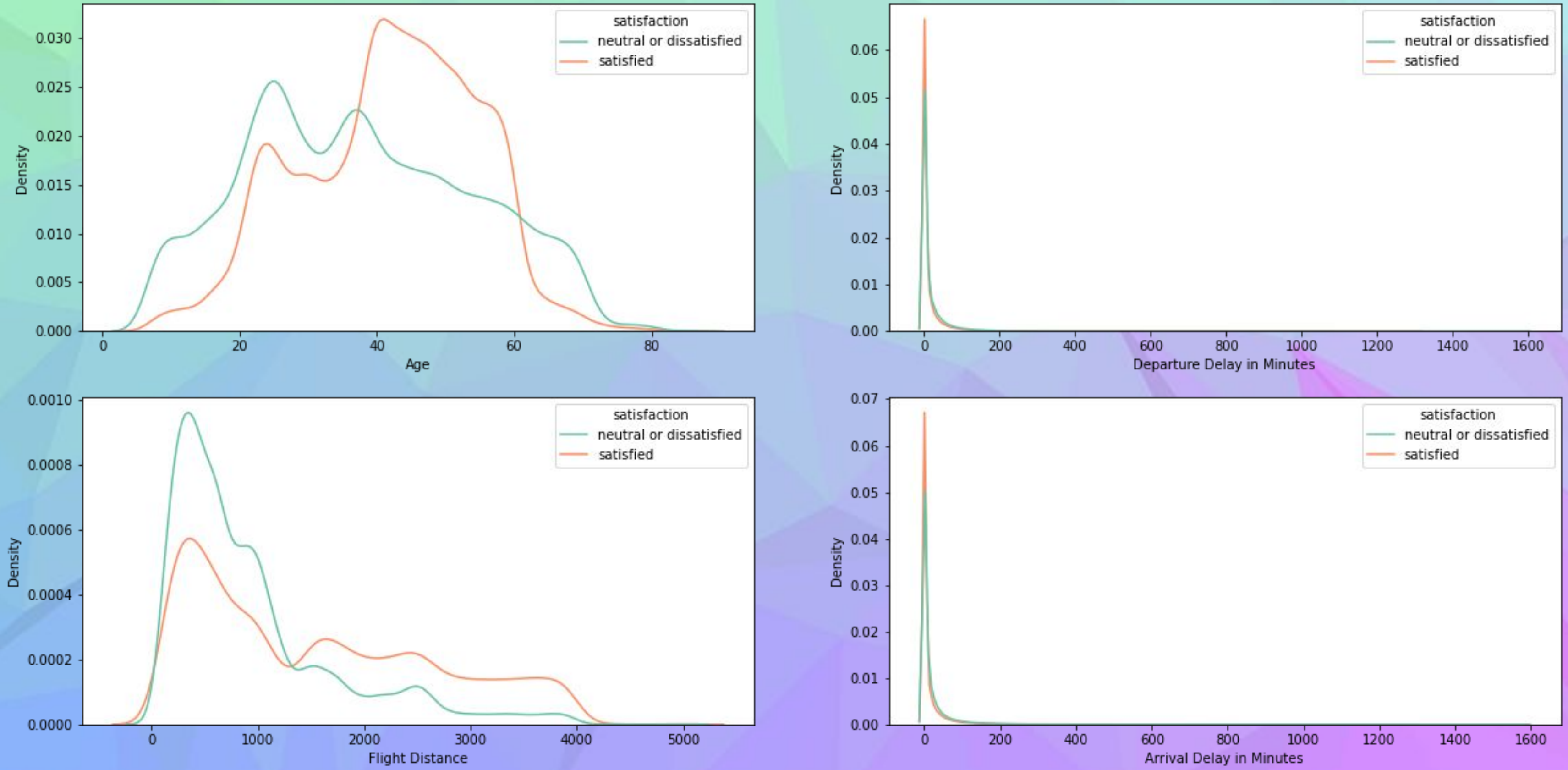




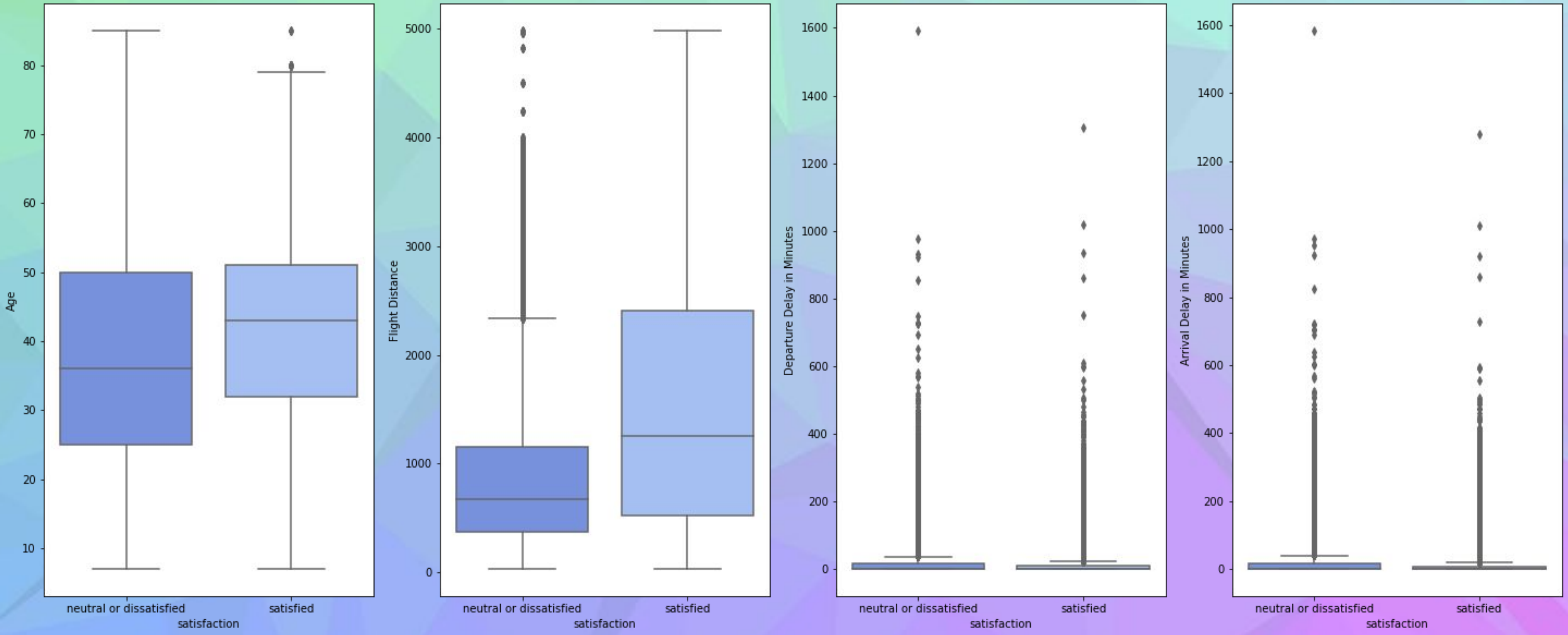
CATEGORICAL LABEL DISTRIBUTION (NON-RATINGS)



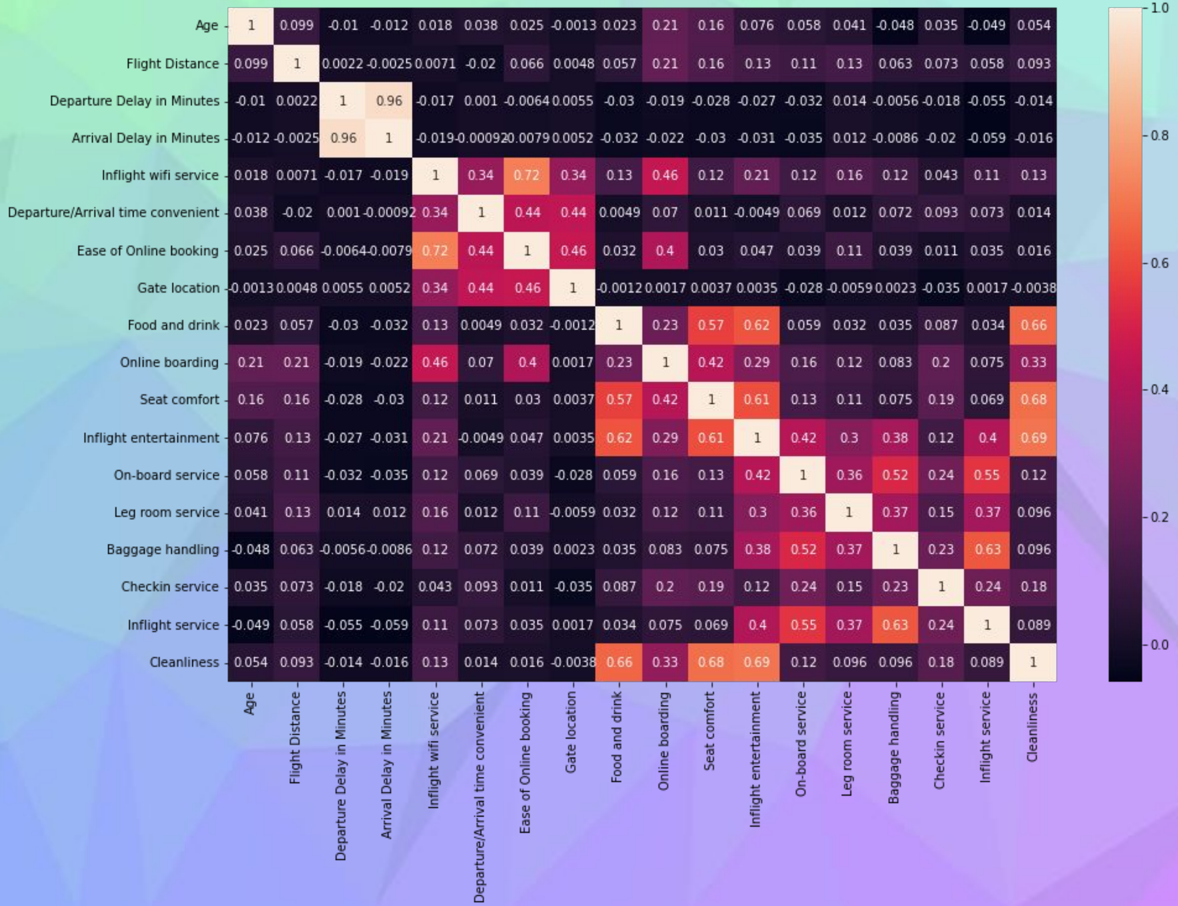
Numerical Label Distribution (Kdeplot)



Numerical Label Distribution (Boxplot)



HEATMAP


Age 1 0.099 -0.01 -0.012 0.018 0.038 0.025 -0.0013 0.023 0.21 0.16 0.076 0.058 0.041 -0.048 0.035 -0.049 0.054
Flight Distance 0.099 1 0.0022 -0.0025 0.0071 -0.02 0.066 0.0048 0.057 0.21 0.16 0.13 0.11 0.13 0.063 0.073 0.058 0.093
Departure Delay in Minutes -0.01 0.0022 1 0.96 -0.017 0.001 -0.0064 0.0055 -0.03 -0.019 -0.028 -0.027 -0.032 0.014 -0.0056 -0.018 -0.055 -0.014
Arrival Delay in Minutes -0.012 -0.0025 0.96 1 -0.019 -0.00092 -0.0079 0.0052 -0.032 -0.022 -0.03 -0.031 -0.035 0.012 -0.0086 -0.02 -0.059 -0.016
Inflight wifi service 0.018 0.0071 -0.017 -0.019 1 0.34 0.72 0.34 0.13 0.46 0.12 0.21 0.12 0.16 0.12 0.043 0.11 0.13
Departure/Arrival time convenient 0.038 -0.02 0.001 -0.00092 0.34 1 0.44 0.44 0.0049 0.07 0.011 -0.0049 0.069 0.012 0.072 0.093 0.073 0.014
Ease of Online booking 0.025 0.066 -0.0064 -0.0079 0.72 0.44 1 0.46 0.032 0.4 0.03 0.047 0.039 0.11 0.039 0.011 0.035 0.016
Gate location -0.0013 0.0048 0.0055 0.0052 0.34 0.44 0.46 1 -0.0012 0.0017 0.0037 0.0035 -0.028 -0.0059 0.0023 -0.035 0.0017 -0.0038
Food and drink 0.023 0.057 -0.03 -0.032 0.13 0.0049 0.032 -0.0012 1 0.23 0.57 0.62 0.059 0.032 0.035 0.087 0.034 0.66
Online boarding 0.21 0.21 -0.019 -0.022 0.46 0.07 0.4 0.0017 0.23 1 0.42 0.29 0.16 0.12 0.083 0.2 0.075 0.33
Seat comfort 0.16 0.16 -0.028 -0.03 0.12 0.011 0.03 0.0037 0.57 0.42 1 0.61 0.13 0.11 0.075 0.19 0.069 0.68
Inflight entertainment 0.076 0.13 -0.027 -0.031 0.21 -0.0049 0.047 0.0035 0.62 0.29 0.61 1 0.42 0.3 0.38 0.12 0.4 0.69
On-board service 0.058 0.11 -0.032 -0.035 0.12 0.069 0.039 -0.028 0.059 0.16 0.13 0.42 1 0.36 0.52 0.24 0.55 0.12
Leg room service 0.041 0.13 0.014 0.012 0.16 0.012 0.11 -0.0059 0.032 0.12 0.11 0.3 0.36 1 0.37 0.15 0.37 0.096
Baggage handling -0.048 0.063 -0.0056 -0.0086 0.12 0.072 0.039 0.0023 0.035 0.083 0.075 0.38 0.52 0.37 1 0.23 0.63 0.096
Checkin service 0.035 0.073 -0.018 -0.02 0.043 0.093 0.011 -0.035 0.087 0.2 0.19 0.12 0.24 0.15 0.23 1 0.24 0.18
Inflight service -0.049 0.058 -0.055 -0.059 0.11 0.073 0.035 0.0017 0.034 0.075 0.069 0.4 0.55 0.37 0.63 0.24 1 0.089
Cleanliness 0.054 0.093 -0.014 -0.016 0.13 0.014 0.016 -0.0038 0.66 0.33 0.68 0.69 0.12 0.096 0.096 0.18 0.089 1
Age
Flight Distance
Departure Delay in Minutes
Arrival Delay in Minutes
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
1.0
0.8
0.6
0.4
0.2
0.0

Data Preprocessing

01

Split the data

Using train_test_split
(train:test =4:1)

02

Remove outliers

Using Interquartile Range (IQR) to detect outliers

03

Scale the data

Scale numerical data using MinMaxScaler

04A

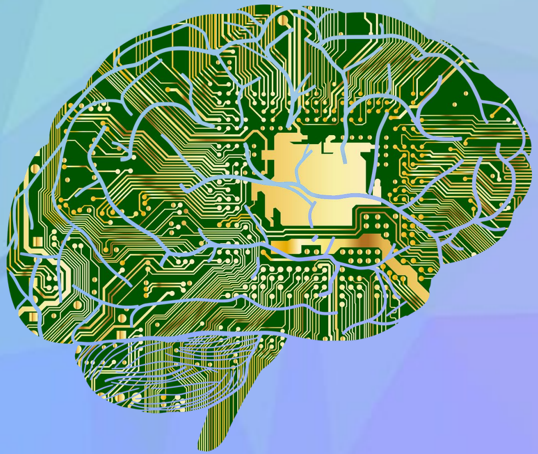
Encode Nominal Data

One-Hot encode nominal columns using pandas.get_dummies

04B

Encode Ordinal Data

Encode ordinal columns using map



Split the Dataset

```
train_df, test_df = train_test_split(df, random_state = 42, test_size = 0.2)
```

Remove Outliers

```
def removeOutlier(df, column):
    Q1=df[column].quantile(0.25)
    Q3=df[column].quantile(0.75)
    IQR=Q3-Q1
    df_final=df[~((df[column]<(Q1-1.5*IQR)) | (df[column]>(Q3+1.5*IQR)))]
    return df_final
```

Scale Numerical Data

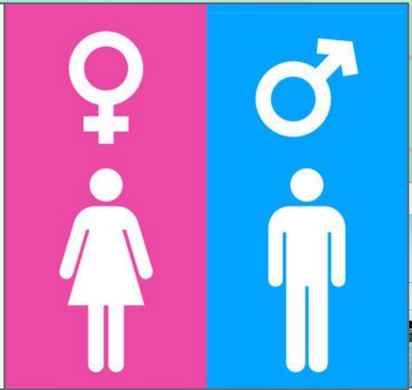
```
def preprocessData(scaler, df, column, type):
  if type == 'train':
    df[column] = scaler.fit_transform(df[[column]])
  elif type == 'test':
    df[column] = scaler.transform(df[column])
  return df
```

```
minmax = MinMaxScaler()
for i in num:
  print(i)
  train_df = preprocessData(minmax, train_df, i, 'train')
  test_df = preprocessData(minmax, test_df, [i], 'test')
```

Encode Categorical Data

```
def getDummies(df2, column):
  df2_ex = pd.DataFrame()
  df2_ex = pd.get_dummies(df2[column])

  df2 = pd.concat([df2, df2_ex], axis = 1)
  df2 = df2.drop(column, axis = 1)
  return df2
```




```
def ordinalEncode(df, map, column):
  df[column] = df[column].map(map)
  return df
class_map = {'Business' : 2,
             'Eco Plus' : 1,
             'Eco': 0}
cType_map = {'Loyal Customer': 1,
             'disloyal Customer': 0}
```

Modeling & Evaluation

Evaluation Metrics
The metrics used are accuracy, precision, recall, F1-Score, and ROC

Model redefinition
04
Fit the model with tuned parameters

Hyperparameter tuning
03
Use GridSearchCV

Filter models
02
Choose the model with best performance

Prepare models
01
Define 11 different classifier models, fit each of them and summarize it's metrics

Prepare Models

```
models = {
    "Logistic Regression" : LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "LDA" : LinearDiscriminantAnalysis(),
    "SGD" : SGDClassifier(),
    "Gaussian": GaussianNB(),
    "Random Forest" : RandomForestClassifier(),
    "Gradient Boosting" : GradientBoostingClassifier(),
    "XGBoost" : XGBClassifier(),
    "CatBoost" : CatBoostClassifier(),
    "LGBM" : LGBMClassifier(),
    "KNN" : KNeighborsClassifier(n_neighbors=3)
}
```

First of all, we initialize the models which performances we would like to compare.

```
X_train = train_df.drop("satisfaction", axis='columns')
y_train = train_df['satisfaction']
X_test = test_df.drop('satisfaction', axis='columns')
y_test = test_df['satisfaction']
```

We then split the datasets into one that only contains the target column, and another that doesn't contain the target column.

Prepare Models

```
scores = []
probability = {}
for model in models:
  classifier = models[model]
  classifier.fit(X_train, y_train)
  predicts = classifier.predict(X_test)
  try:
    score = classifier.predict_proba(X_test)[:,1]
    roc = roc_auc_score(y_test, score, average='weighted')
    probability[model] = score
  except:
    roc = 0
  scores.append([
    model,
    accuracy_score(y_test, predicts),
    f1_score(y_test, predicts, average ='weighted'),
    precision_score(y_test, predicts),
    recall_score(y_test, predicts),
    roc
  ])
```

We then use a loop to fit the data into each model. The scores for each model are also calculated and stored in the ***scores*** array.

Filter Models

	Model	Accuracy	F1	Precision	Recall	ROC
0	Logistic Regression	0.866753	0.865966	0.877955	0.806793	0.916526
1	Decision Tree	0.941100	0.941135	0.927886	0.937914	0.940740
2	LDA	0.866176	0.865526	0.872320	0.812197	0.918199
3	SGD	0.869256	0.867931	0.899685	0.788266	0.000000
4	Gaussian	0.841105	0.840107	0.847517	0.775364	0.879621
5	Random Forest	0.961166	0.961084	0.971249	0.938796	0.993430
6	Gradient Boosting	0.943314	0.943213	0.947787	0.920820	0.987916
7	XGBoost	0.939656	0.939531	0.945496	0.914424	0.987658
8	CatBoost	0.962466	0.962394	0.971445	0.941663	0.994979
9	LGBM	0.962273	0.962195	0.972292	0.940340	0.994717
10	KNN	0.919542	0.919086	0.942450	0.868659	0.953444

We then compile the scores into a dataframe, then display them for comparison purposes. Here, we saw that the CatBoost model has the highest accuracy in all of the metrics (Accuracy, F1 Score, Precision, Recall, and ROC).

Hyperparameter Tuning

In order to improve the performance of the model, we tuned the hyperparameters using ***GridSearchCV***.

```
grid = {'max_depth': [5, 6, 7],
        'n_estimators':[300, 400, 500],
        'learning_rate' : [0.01, 0.05, 0.1, 0.15]}

cbc = CatBoostClassifier()

gscv = GridSearchCV (estimator = cbc, param_grid = grid, scoring ='accuracy', cv = 5)

gscv.fit(X_train, y_train)
```

We then output the best estimator, score, and parameters.

```
#returns the estimator with the best performance
print(gscv.best_estimator_, end = '\n\n')

#returns the best score
print(gscv.best_score_, end = '\n\n')

#returns the best parameters
print(gscv.best_params_)
```

The resulting output can be seen below, with the best parameters being utilized in the next stage (Model Redefinition).

```
<catboost.core.CatBoostClassifier object at 0x7f00d6544550>

0.9627804851804577

{'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 400}
```

Model Redefinition

After tuning the hyperparameters, we redefine the model with the newly acquired values.

```
final_model = CatBoostClassifier(learning_rate=0.1, max_depth = 7, n_estimators= 400)
final_model.fit(X_train, y_train)
```

We then calculate the performance of the now redefined model using the same metrics as before (Accuracy score, F1 Score, Precision Score, and Recall Score).

```
predicts = final_model.predict(X_test)
print(f"Final Model's accuracy  : {accuracy_score(y_test, predicts)}")
print(f"Final Model's F1 Score  : {f1_score(y_test, predicts, average ='weighted')}")
print(f"Final Model's Precision : {precision_score(y_test, predicts)}")
print(f"Final Model's Recall    : {recall_score(y_test, predicts)}")
```

Results

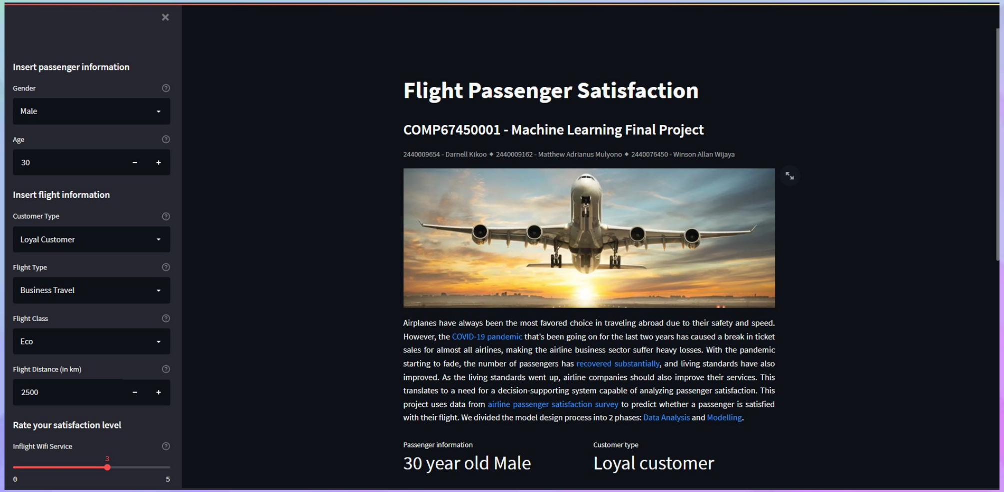
Catboost Model Performance

Metric	Performance
Accuracy	0.9628
Precision	0.9708
Recall	0.9431
F1 Score	0.9627

The model achieved 96.28% accuracy, supported by 97% precision, 94% recall, and 96% F1 Score.

Deployment

We deployed our model in the form of a web-application, which can be accessed through the link: https://flight-satisfaction.herokuapp.com/.




Спасибо
Gracias
شكرا
Obrigado
Спасибо
Dank U
Grazie
Ευχαριστώ
Danke
Ευχαριστώ
شكرا
Merci
Obrigado
Diolch
Ngiyabonga
Dank U
Thank You
Ngiyabonga
Tack
Merci
Ευχαριστώ
Dziękі
Dzięki
Danke
Grazie
תודה
謝謝
Terima Kasih
Diolch
Merci
Dank U
Grazie
Tack
Ευχαριστώ
Tack
감사합니다
Diolch