Exploratory Data Analysis and Visualization Task

Several classification algorithms are used in a classification problem that forecasts airline customers' satisfaction using the supplied data. All categorical data are converted into numerical values using label encoding from the sklearn preprocessing module before being fed into the algorithms. The data is then divided into training and testing sets using the sklearn model\_selection module's train\_test\_split function. The best classification model to predict customer happiness is then evaluated.

You may start by looking at the data using the head method to examine the top few rows of the dataset in order to undertake exploratory data analysis and visualisation activities. This might help you get a sense of the features and how the data is presented. The describe technique may also be used to obtain summary statistics for the data, including mean, median, and standard deviation. Additionally, you can visualise the data's distribution and check for outliers or correlations between various features using scatter plots, histograms, and box plots.

Data Analysis and Visualization tasks that you can perform:

Use Airline\_satisfaction\_data.head() and Airline\_satisfaction\_data.describe() to see the first fiw rows and summary statistics of the data.

Create a scatter plot of the age and flight\_distance features to see if there is any correlation between them. You can use plt.scatter to create the plot.

Create a histogram of the inflight\_wifi\_service feature to see the distribution of ratings for this feature. You can use plt.hist(Airline\_satisfaction\_data["inflight\_wifi\_service"], bins=10) to create the histogram.

Create a box plot of the satisfaction feature to see the distribution of customer satisfaction levels. You can use plt.boxplot(Airline\_satisfaction\_data["satisfaction"]) to create the box plot.

Predictive Analytics Task

The code creates a GUI for data entry into a customer satisfaction prediction model. Based on a number of input variables, it predicts the degree of consumer happiness with airlines using a random forest classifier from the scikit-learn package. Predictive analytics' mission is to examine previous data to find patterns and trends, then utilise this knowledge to forecast future events. In this instance, the aim is to create a predictive model that, using the input data, can precisely anticipate customer happiness. The supplied training data is used to train the random forest classifier, which then uses those patterns to provide predictions for fresh data points.

How to run the codes

So the satisfaction\_gui (for the three diagram to appear there is 2 codes and excel files that is needed to make it work ) and it can really take some time cuz it is running slowly so need to be patient with these three to appear but an estimation would be 842 secs. However it is highly recommended that use the code according to the instruction given below

The codes for the 3 diff image to appear:

Satisfaction\_gui

import tkinter as tk

from satisfaction\_model import \*

# Read the test data from CSV

test\_data = pd.read\_csv('mnist\_test.csv')

# Create the GUI

class SatisfactionPredictorGUI:

def \_\_init\_\_(self, master):

self.master = master

master.title("Satisfaction Predictor")

# Create the input fields

tk.Label(master, text="Age:").grid(row=0, column=0)

tk.Label(master, text="Gender:").grid(row=1, column=0)

tk.Label(master, text="Class:").grid(row=2, column=0)

tk.Label(master, text="Inflight wifi service:").grid(row=3, column=0)

tk.Label(master, text="Departure/Arrival time convenient:").grid(row=4, column=0)

tk.Label(master, text="Ease of Online booking:").grid(row=5, column=0)

tk.Label(master, text="Gate location:").grid(row=6, column=0)

tk.Label(master, text="Food and drink:").grid(row=7, column=0)

tk.Label(master, text="Online boarding:").grid(row=8, column=0)

tk.Label(master, text="Seat comfort:").grid(row=9, column=0)

tk.Label(master, text="Inflight entertainment:").grid(row=10, column=0)

tk.Label(master, text="On-board service:").grid(row=11, column=0)

tk.Label(master, text="Leg room service:").grid(row=12, column=0)

tk.Label(master, text="Baggage handling:").grid(row=13, column=0)

tk.Label(master, text="Checkin service:").grid(row=14, column=0)

tk.Label(master, text="Inflight service:").grid(row=15, column=0)

tk.Label(master, text="Cleanliness:").grid(row=16, column=0)

tk.Label(master, text="Departure Delay in Minutes:").grid(row=17, column=0)

tk.Label(master, text="Arrival Delay in Minutes:").grid(row=18, column=0)

self.age\_entry = tk.Entry(master)

self.gender\_entry = tk.Entry(master)

self.class\_entry = tk.Entry(master)

self.inflight\_wifi\_entry = tk.Entry(master)

self.departure\_arrival\_entry = tk.Entry(master)

self.ease\_of\_booking\_entry = tk.Entry(master)

self.gate\_location\_entry = tk.Entry(master)

self.food\_drink\_entry = tk.Entry(master)

self.online\_boarding\_entry = tk.Entry(master)

self.seat\_comfort\_entry = tk.Entry(master)

self.inflight\_entertainment\_entry = tk.Entry(master)

self.onboard\_service\_entry = tk.Entry(master)

self.legroom\_service\_entry = tk.Entry(master)

self.baggage\_handling\_entry = tk.Entry(master)

self.checkin\_service\_entry = tk.Entry(master)

self.inflight\_service\_entry = tk.Entry(master)

self.cleanliness\_entry = tk.Entry(master)

self.departure\_delay\_entry = tk.Entry(master)

self.arrival\_delay\_entry = tk.Entry(master)

self.age\_entry.grid(row=0, column=1)

self.gender\_entry.grid(row=1, column=1)

self.class\_entry.grid(row=2, column=1)

self.inflight\_wifi\_entry.grid(row=3, column=1)

self.departure\_arrival\_entry.grid(row=4, column=1)

self.ease\_of\_booking\_entry.grid(row=5, column=1)

self.gate\_location\_entry.grid(row=6, column=1)

self.food\_drink\_entry.grid(row=7, column=1)

self.online\_boarding\_entry.grid(row=8, column=1)

self.seat\_comfort\_entry.grid(row=9, column=1)

self.inflight\_entertainment\_entry.grid(row=10, column=1)

self.onboard\_service\_entry.grid(row=11, column=1)

self.legroom\_service\_entry.grid(row=12, column=1)

self.baggage\_handling\_entry.grid(row=13, column=1)

self.checkin\_service\_entry.grid(row=14, column=1)

self.inflight\_service\_entry.grid(row=15, column=1)

self.cleanliness\_entry.grid(row=16, column=1)

self.departure\_delay\_entry.grid(row=17, column=1)

self.arrival\_delay\_entry.grid(row=18, column=1)

# Create the predict button

self.predict\_button = tk.Button(master, text="Predict", command=self.predict\_satisfaction)

self.predict\_button.grid(row=19, column=0, columnspan=2, pady=10)

# Create the output field

self.output\_label = tk.Label(master, text="")

self.output\_label.grid(row=20, column=0, columnspan=2)

# Define the function to predict the satisfaction

def predict\_satisfaction(self):

# Get the input values from the GUI

age = int(self.age\_entry.get())

gender = int(self.gender\_entry.get())

class\_ = int(self.class\_entry.get())

inflight\_wifi\_service = int(self.inflight\_wifi\_entry.get())

departure\_arrival\_time\_convenient = int(self.departure\_arrival\_entry.get())

ease\_of\_online\_booking = int(self.ease\_of\_booking\_entry.get())

gate\_location = int(self.gate\_location\_entry.get())

food\_and\_drink = int(self.food\_drink\_entry.get())

online\_boarding = int(self.online\_boarding\_entry.get())

seat\_comfort = int(self.seat\_comfort\_entry.get())

inflight\_entertainment = int(self.inflight\_entertainment\_entry.get())

onboard\_service = int(self.onboard\_service\_entry.get())

legroom\_service = int(self.legroom\_service\_entry.get())

baggage\_handling = int(self.baggage\_handling\_entry.get())

checkin\_service = int(self.checkin\_service\_entry.get())

inflight\_service = int(self.inflight\_service\_entry.get())

cleanliness = int(self.cleanliness\_entry.get())

departure\_delay = int(self.departure\_delay\_entry.get())

arrival\_delay = int(self.arrival\_delay\_entry.get())

# Create the input array for prediction

input\_arr = [[age, gender, class\_, inflight\_wifi\_service, departure\_arrival\_time\_convenient,

ease\_of\_online\_booking, food\_and\_beverages, gate\_location, inflight\_entertainment,

onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service,

cleanliness, departure\_delay\_in\_minutes, arrival\_delay\_in\_minutes]]

predicted\_satisfaction = predict\_satisfaction(input\_arr)

self.output\_label.config(text=f"Predicted satisfaction: {predicted\_satisfaction[0]}")

satisfaction\_model

import matplotlib.pyplot as pyplot

import pandas as pd

import sklearn

# Load libraries

from matplotlib import pyplot as plt

from sklearn import preprocessing

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

# # Data file import

# Airline\_satisfaction\_data = pd.read\_csv("mnist\_test.csv")

Airline\_satisfaction\_data = pd.read\_csv("train.csv")

# Attribute to be predicted

predict = "Airline\_satisfaction\_data"

# Dataset/Column to be Predicted, X is all attributes and y is the features

#x = np.array(heart\_data.drop([predict], 1)) # Will return a new data frame that doesnt have hd in it

#y = np.array(heart\_data[predict])

l = preprocessing.LabelEncoder()

age = l.fit\_transform(list(Airline\_satisfaction\_data["Age"]))#AGE OF THE PASSENGER

Gender = l.fit\_transform(list(Airline\_satisfaction\_data["Gender"]))

id = l.fit\_transform(list(Airline\_satisfaction\_data["id"]))

customer\_type = l.fit\_transform(list(Airline\_satisfaction\_data["Customer Type"]))

type\_of\_travel = l.fit\_transform(list(Airline\_satisfaction\_data["Type of Travel"]))

class\_ = l.fit\_transform(list(Airline\_satisfaction\_data["Class"]))

flight\_distance = l.fit\_transform(list(Airline\_satisfaction\_data["Flight Distance"]))

inflight\_wifi\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight wifi service"]))

departure\_arrival\_time\_convenient = l.fit\_transform(list(Airline\_satisfaction\_data["Departure/Arrival time convenient"]))

ease\_of\_online\_booking = l.fit\_transform(list(Airline\_satisfaction\_data["Ease of Online booking"]))

gate\_location = l.fit\_transform(list(Airline\_satisfaction\_data["Gate location"]))

food\_and\_drink = l.fit\_transform(list(Airline\_satisfaction\_data["Food and drink"]))

online\_boarding = l.fit\_transform(list(Airline\_satisfaction\_data["Online boarding"]))

seat\_comfort = l.fit\_transform(list(Airline\_satisfaction\_data["Seat comfort"]))

inflight\_entertainment = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight entertainment"]))

onboard\_service = l.fit\_transform(list(Airline\_satisfaction\_data["On-board service"]))

leg\_room\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Leg room service"]))

baggage\_handling = l.fit\_transform(list(Airline\_satisfaction\_data["Baggage handling"]))

checkin\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Checkin service"]))

inflight\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight service"]))

cleanliness = l.fit\_transform(list(Airline\_satisfaction\_data["Cleanliness"]))

departure\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Departure Delay in Minutes"]))

arrival\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Arrival Delay in Minutes"]))

satisfaction = l.fit\_transform(list(Airline\_satisfaction\_data["satisfaction"]))

x = list(zip(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay))

y = list(satisfaction)

# Test options and evaluation metric

num\_folds = 5

seed = 7

scoring = 'accuracy'

# Model Test/Train

# Splitting what we are trying to predict into 4 different arrays -

# X train is a section of the x array(attributes) and vise versa for Y(features)

# The test data will test the accuracy of the model created

x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

#splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows

# Check with different Scikit-learn classification algorithms

models = []

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

# evaluate each model in turn

results = []

names = []

for name, model in models:

kfold = KFold(n\_splits=num\_folds,shuffle=True,random\_state=seed)

cv\_results = cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

msg += '\n'

print(msg)

# Compare Algorithms' Performance

fig = pyplot.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

pyplot.boxplot(results)

ax.set\_xticklabels(names)

pyplot.show()

# Make predictions on validation/test dataset

dt = DecisionTreeClassifier()

nb = GaussianNB()

gb = GradientBoostingClassifier()

rf = RandomForestClassifier()

best\_model = rf

best\_model.fit(x\_train, y\_train)

y\_pred = best\_model.predict(x\_test)

model\_accuracy = accuracy\_score(y\_test, y\_pred)

print("Best Model Accuracy Score on Test Set:", model\_accuracy)

#Model Evaluation Metric 1

print(classification\_report(y\_test, y\_pred))

#Model Evaluation Metric 2

#Confusion matrix

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

cm = confusion\_matrix(y\_test, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot()

plt.show()

#Model Evaluation Metric 3

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

best\_model = rf

best\_model.fit(x\_train, y\_train)

rf\_roc\_auc = roc\_auc\_score(y\_test,best\_model.predict(x\_test))

fpr,tpr,thresholds = roc\_curve(y\_test, best\_model.predict\_proba(x\_test)[:,1])

plt.figure()

plt.plot(fpr,tpr,label = 'Random Forest(area = %0.2f)'% rf\_roc\_auc)

plt.plot([0,1],[0,1],'r--')

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.05])

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc='lower right')

plt.savefig('LOC\_ROC')

plt.show()

#Check actual/ground truth vs predicted diagnosis

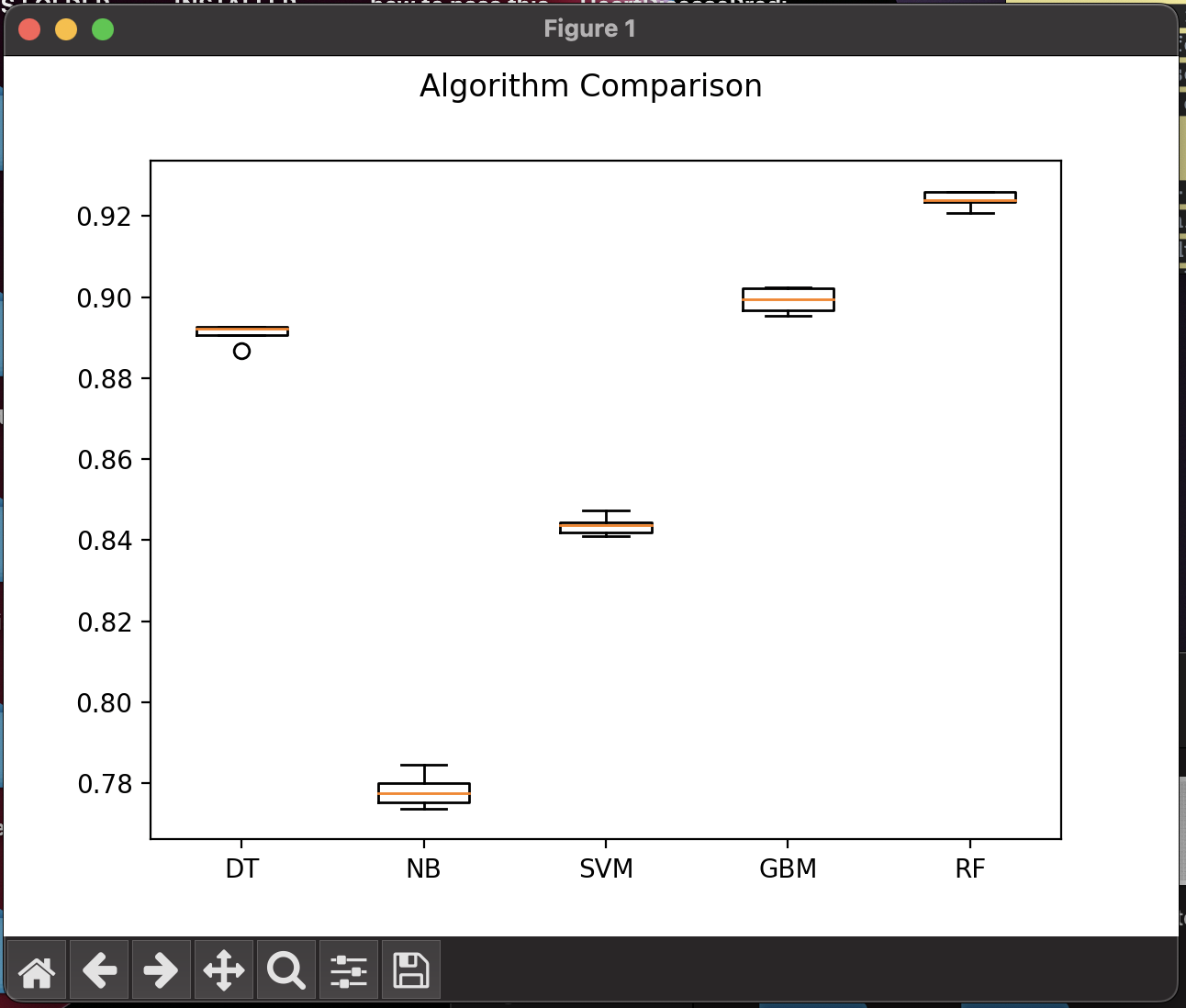
for x in range(len(y\_pred)):

print("Predicted: ", y\_pred[x], "Actual: ", y\_test[x], "Data: ", x\_test[x],)

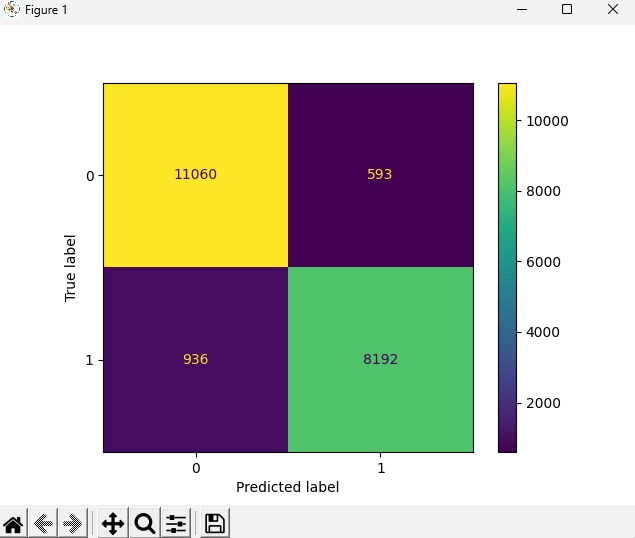
the excel will be provided and these will be the results

These are the results

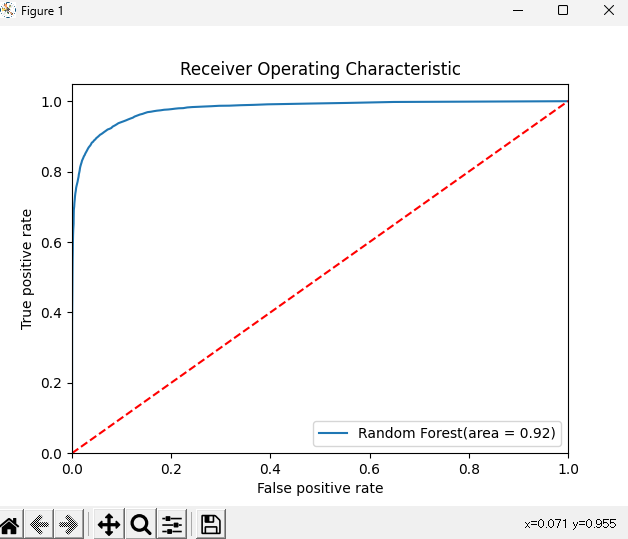
1.algorithm comparison



Confusionn matrix



receiver operating characteristics



Now for the Implementation and Deployment Task

Satisfaction\_gui

import matplotlib.pyplot as pyplot

import pandas as pd

import sklearn

# Load libraries

from matplotlib import pyplot as plt

from sklearn import preprocessing

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

import tkinter as tk

# from satisfaction\_model import \*

# Read the test data from CSV

test\_data = pd.read\_csv('train.csv')

# Create the GUI

class SatisfactionPredictorGUI(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

master = self

self.master = master

master.title("Satisfaction Predictor")

# Create the input fields

tk.Label(master, text="Age:").grid(row=0, column=0)

tk.Label(master, text="Gender:").grid(row=1, column=0)

tk.Label(master, text="Class:").grid(row=2, column=0)

tk.Label(master, text="Inflight wifi service:").grid(row=3, column=0)

tk.Label(master, text="Departure/Arrival time convenient:").grid(row=4, column=0)

tk.Label(master, text="Ease of Online booking:").grid(row=5, column=0)

tk.Label(master, text="Gate location:").grid(row=6, column=0)

tk.Label(master, text="Food and drink:").grid(row=7, column=0)

tk.Label(master, text="Online boarding:").grid(row=8, column=0)

tk.Label(master, text="Seat comfort:").grid(row=9, column=0)

tk.Label(master, text="Inflight entertainment:").grid(row=10, column=0)

tk.Label(master, text="On-board service:").grid(row=11, column=0)

tk.Label(master, text="Leg room service:").grid(row=12, column=0)

tk.Label(master, text="Baggage handling:").grid(row=13, column=0)

tk.Label(master, text="Checkin service:").grid(row=14, column=0)

tk.Label(master, text="Inflight service:").grid(row=15, column=0)

tk.Label(master, text="Cleanliness:").grid(row=16, column=0)

tk.Label(master, text="Departure Delay in Minutes:").grid(row=17, column=0)

tk.Label(master, text="Arrival Delay in Minutes:").grid(row=18, column=0)

# age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay

self.age\_entry = tk.Entry(master)

self.gender\_entry = tk.Entry(master)

self.class\_entry = tk.Entry(master)

self.inflight\_wifi\_entry = tk.Entry(master)

self.departure\_arrival\_entry = tk.Entry(master)

self.ease\_of\_booking\_entry = tk.Entry(master)

self.gate\_location\_entry = tk.Entry(master)

self.food\_drink\_entry = tk.Entry(master)

self.online\_boarding\_entry = tk.Entry(master)

self.seat\_comfort\_entry = tk.Entry(master)

self.inflight\_entertainment\_entry = tk.Entry(master)

self.onboard\_service\_entry = tk.Entry(master)

self.legroom\_service\_entry = tk.Entry(master)

self.baggage\_handling\_entry = tk.Entry(master)

self.checkin\_service\_entry = tk.Entry(master)

self.inflight\_service\_entry = tk.Entry(master)

self.cleanliness\_entry = tk.Entry(master)

self.departure\_delay\_entry = tk.Entry(master)

self.arrival\_delay\_entry = tk.Entry(master)

self.age\_entry.grid(row=0, column=1)

self.gender\_entry.grid(row=1, column=1)

self.class\_entry.grid(row=2, column=1)

self.inflight\_wifi\_entry.grid(row=3, column=1)

self.departure\_arrival\_entry.grid(row=4, column=1)

self.ease\_of\_booking\_entry.grid(row=5, column=1)

self.gate\_location\_entry.grid(row=6, column=1)

self.food\_drink\_entry.grid(row=7, column=1)

self.online\_boarding\_entry.grid(row=8, column=1)

self.seat\_comfort\_entry.grid(row=9, column=1)

self.inflight\_entertainment\_entry.grid(row=10, column=1)

self.onboard\_service\_entry.grid(row=11, column=1)

self.legroom\_service\_entry.grid(row=12, column=1)

self.baggage\_handling\_entry.grid(row=13, column=1)

self.checkin\_service\_entry.grid(row=14, column=1)

self.inflight\_service\_entry.grid(row=15, column=1)

self.cleanliness\_entry.grid(row=16, column=1)

self.departure\_delay\_entry.grid(row=17, column=1)

self.arrival\_delay\_entry.grid(row=18, column=1)

self.age\_entry.insert(0, '18')

self.gender\_entry.insert(0, '1')

self.class\_entry.insert(0, '1')

self.inflight\_wifi\_entry.insert(0, '4')

self.departure\_arrival\_entry.insert(0, '3')

self.ease\_of\_booking\_entry.insert(0, '3')

self.gate\_location\_entry.insert(0, '2')

self.food\_drink\_entry.insert(0, '4')

self.online\_boarding\_entry.insert(0, '3')

self.seat\_comfort\_entry.insert(0, '3')

self.inflight\_entertainment\_entry.insert(0, '4')

self.onboard\_service\_entry.insert(0, '4')

self.legroom\_service\_entry.insert(0, '5')

self.baggage\_handling\_entry.insert(0, '2')

self.checkin\_service\_entry.insert(0, '1')

self.inflight\_service\_entry.insert(0, '3')

self.cleanliness\_entry.insert(0, '4')

self.departure\_delay\_entry.insert(0, '0')

self.arrival\_delay\_entry.insert(0, '0')

# 18, 1, 1, 4, 3, 3, 2, 4, 3, 3, 4, 4, 5, 2, 1, 3, 4, 0, 0)

# Create the predict button

self.predict\_button = tk.Button(master, text="Predict", command=self.predict\_satisfaction)

self.predict\_button.grid(row=19, column=0, columnspan=2, pady=10)

# Create the output field

self.output\_label = tk.Label(master, text="output prediction")

self.output\_label.grid(row=20, column=0, columnspan=2)

# --------------------------

Airline\_satisfaction\_data = pd.read\_csv("train.csv")

# Attribute to be predicted

predict = "Airline\_satisfaction\_data"

# Dataset/Column to be Predicted, X is all attributes and y is the features

#x = np.array(heart\_data.drop([predict], 1)) # Will return a new data frame that doesnt have hd in it

#y = np.array(heart\_data[predict])

l = preprocessing.LabelEncoder()

age = l.fit\_transform(list(Airline\_satisfaction\_data["Age"]))#AGE OF THE PASSENGER

Gender = l.fit\_transform(list(Airline\_satisfaction\_data["Gender"]))

id = l.fit\_transform(list(Airline\_satisfaction\_data["id"]))

customer\_type = l.fit\_transform(list(Airline\_satisfaction\_data["Customer Type"]))

type\_of\_travel = l.fit\_transform(list(Airline\_satisfaction\_data["Type of Travel"]))

class\_ = l.fit\_transform(list(Airline\_satisfaction\_data["Class"]))

flight\_distance = l.fit\_transform(list(Airline\_satisfaction\_data["Flight Distance"]))

inflight\_wifi\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight wifi service"]))

departure\_arrival\_time\_convenient = l.fit\_transform(list(Airline\_satisfaction\_data["Departure/Arrival time convenient"]))

ease\_of\_online\_booking = l.fit\_transform(list(Airline\_satisfaction\_data["Ease of Online booking"]))

gate\_location = l.fit\_transform(list(Airline\_satisfaction\_data["Gate location"]))

food\_and\_drink = l.fit\_transform(list(Airline\_satisfaction\_data["Food and drink"]))

online\_boarding = l.fit\_transform(list(Airline\_satisfaction\_data["Online boarding"]))

seat\_comfort = l.fit\_transform(list(Airline\_satisfaction\_data["Seat comfort"]))

inflight\_entertainment = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight entertainment"]))

onboard\_service = l.fit\_transform(list(Airline\_satisfaction\_data["On-board service"]))

leg\_room\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Leg room service"]))

baggage\_handling = l.fit\_transform(list(Airline\_satisfaction\_data["Baggage handling"]))

checkin\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Checkin service"]))

inflight\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight service"]))

cleanliness = l.fit\_transform(list(Airline\_satisfaction\_data["Cleanliness"]))

departure\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Departure Delay in Minutes"]))

arrival\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Arrival Delay in Minutes"]))

satisfaction = l.fit\_transform(list(Airline\_satisfaction\_data["satisfaction"]))

x = list(zip(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay))

y = list(satisfaction)

# Test options and evaluation metric

num\_folds = 5

seed = 7

scoring = 'accuracy'

# Model Test/Train

# Splitting what we are trying to predict into 4 different arrays -

# X train is a section of the x array(attributes) and vise versa for Y(features)

# The test data will test the accuracy of the model created

# self.x\_train, self.x\_test, self.y\_train, self.y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

self.model = RandomForestClassifier()

self.model.fit(x\_train, y\_train)

# -----------------------------

# Define the function to predict the satisfaction

def predict\_satisfaction(self):

# # Get the input values from the GUI

# age = l.fit\_transform(list(Airline\_satisfaction\_data["Age"]))#AGE OF THE PASSENGER

# Gender = l.fit\_transform(list(Airline\_satisfaction\_data["Gender"]))

# id = l.fit\_transform(list(Airline\_satisfaction\_data["id"]))

# customer\_type = l.fit\_transform(list(Airline\_satisfaction\_data["Customer Type"]))

# type\_of\_travel = l.fit\_transform(list(Airline\_satisfaction\_data["Type of Travel"]))

# class\_ = l.fit\_transform(list(Airline\_satisfaction\_data["Class"]))

# flight\_distance = l.fit\_transform(list(Airline\_satisfaction\_data["Flight Distance"]))

# inflight\_wifi\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight wifi service"]))

# departure\_arrival\_time\_convenient = l.fit\_transform(list(Airline\_satisfaction\_data["Departure/Arrival time convenient"]))

# ease\_of\_online\_booking = l.fit\_transform(list(Airline\_satisfaction\_data["Ease of Online booking"]))

# gate\_location = l.fit\_transform(list(Airline\_satisfaction\_data["Gate location"]))

# food\_and\_drink = l.fit\_transform(list(Airline\_satisfaction\_data["Food and drink"]))

# online\_boarding = l.fit\_transform(list(Airline\_satisfaction\_data["Online boarding"]))

# seat\_comfort = l.fit\_transform(list(Airline\_satisfaction\_data["Seat comfort"]))

# inflight\_entertainment = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight entertainment"]))

# onboard\_service = l.fit\_transform(list(Airline\_satisfaction\_data["On-board service"]))

# leg\_room\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Leg room service"]))

# baggage\_handling = l.fit\_transform(list(Airline\_satisfaction\_data["Baggage handling"]))

# checkin\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Checkin service"]))

# inflight\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight service"]))

# cleanliness = l.fit\_transform(list(Airline\_satisfaction\_data["Cleanliness"]))

# departure\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Departure Delay in Minutes"]))

# arrival\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Arrival Delay in Minutes"]))

# satisfaction = l.fit\_transform(list(Airline\_satisfaction\_data["satisfaction"]))

# --------

# age = int(self.age\_entry.get())

# Gender = int(self.gender\_entry.get())

# class\_ = int(self.class\_entry.get())

# inflight\_wifi\_service = int(self.inflight\_wifi\_entry.get())

# departure\_arrival\_time\_convenient = int(self.departure\_arrival\_entry.get())

# ease\_of\_online\_booking = int(self.ease\_of\_booking\_entry.get())

# gate\_location = int(self.gate\_location\_entry.get())

# food\_and\_drink = int(self.food\_drink\_entry.get())

# online\_boarding = int(self.online\_boarding\_entry.get())

# seat\_comfort = int(self.seat\_comfort\_entry.get())

# inflight\_entertainment = int(self.inflight\_entertainment\_entry.get())

# onboard\_service = int(self.onboard\_service\_entry.get())

# legroom\_service = int(self.legroom\_service\_entry.get())

# baggage\_handling = int(self.baggage\_handling\_entry.get())

# checkin\_service = int(self.checkin\_service\_entry.get())

# inflight\_service = int(self.inflight\_service\_entry.get())

# cleanliness = int(self.cleanliness\_entry.get())

# departure\_delay = int(self.departure\_delay\_entry.get())

# arrival\_delay = int(self.arrival\_delay\_entry.get())

age = int(self.age\_entry.get())

Gender = int(self.gender\_entry.get())

class\_ = int(self.class\_entry.get())

inflight\_wifi\_service = int(self.inflight\_wifi\_entry.get())

departure\_arrival\_time\_convenient = int(self.departure\_arrival\_entry.get())

ease\_of\_online\_booking = int(self.ease\_of\_booking\_entry.get())

gate\_location = int(self.gate\_location\_entry.get())

food\_and\_drink = int(self.food\_drink\_entry.get())

online\_boarding = int(self.online\_boarding\_entry.get())

seat\_comfort = int(self.age\_entry.get())

inflight\_entertainment = int(self.age\_entry.get())

onboard\_service = int(self.onboard\_service\_entry.get())

leg\_room\_service = int(self.legroom\_service\_entry.get())

baggage\_handling = int(self.baggage\_handling\_entry.get())

checkin\_service = int(self.checkin\_service\_entry.get())

inflight\_service = int(self.inflight\_service\_entry.get())

cleanliness = int(self.cleanliness\_entry.get())

departure\_delay = int(self.departure\_delay\_entry.get())

arrival\_delay = int(self.arrival\_delay\_entry.get())

# Create the input array for prediction

# zip(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay)

input\_arr = [(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay)]

print('predicting')

predicted\_satisfaction\_value = self.model.predict\_proba(input\_arr)

print(predicted\_satisfaction\_value)

print('prediction over')

\_str = f"Predicted satisfaction: {str(predicted\_satisfaction\_value[0][0] \* 100)}%"

print(\_str)

self.output\_label.config(text=\_str)

print('label updated')

self.update\_idletasks()

Satisfaction model

import matplotlib.pyplot as pyplot

import pandas as pd

import sklearn

# Load libraries

from matplotlib import pyplot as plt

from sklearn import preprocessing

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

# # Data file import

# Airline\_satisfaction\_data = pd.read\_csv("mnist\_test.csv")

Airline\_satisfaction\_data = pd.read\_csv("train.csv")

# Attribute to be predicted

predict = "Airline\_satisfaction\_data"

# Dataset/Column to be Predicted, X is all attributes and y is the features

#x = np.array(heart\_data.drop([predict], 1)) # Will return a new data frame that doesnt have hd in it

#y = np.array(heart\_data[predict])

l = preprocessing.LabelEncoder()

age = l.fit\_transform(list(Airline\_satisfaction\_data["Age"]))#AGE OF THE PASSENGER

Gender = l.fit\_transform(list(Airline\_satisfaction\_data["Gender"]))

id = l.fit\_transform(list(Airline\_satisfaction\_data["id"]))

customer\_type = l.fit\_transform(list(Airline\_satisfaction\_data["Customer Type"]))

type\_of\_travel = l.fit\_transform(list(Airline\_satisfaction\_data["Type of Travel"]))

class\_ = l.fit\_transform(list(Airline\_satisfaction\_data["Class"]))

flight\_distance = l.fit\_transform(list(Airline\_satisfaction\_data["Flight Distance"]))

inflight\_wifi\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight wifi service"]))

departure\_arrival\_time\_convenient = l.fit\_transform(list(Airline\_satisfaction\_data["Departure/Arrival time convenient"]))

ease\_of\_online\_booking = l.fit\_transform(list(Airline\_satisfaction\_data["Ease of Online booking"]))

gate\_location = l.fit\_transform(list(Airline\_satisfaction\_data["Gate location"]))

food\_and\_drink = l.fit\_transform(list(Airline\_satisfaction\_data["Food and drink"]))

online\_boarding = l.fit\_transform(list(Airline\_satisfaction\_data["Online boarding"]))

seat\_comfort = l.fit\_transform(list(Airline\_satisfaction\_data["Seat comfort"]))

inflight\_entertainment = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight entertainment"]))

onboard\_service = l.fit\_transform(list(Airline\_satisfaction\_data["On-board service"]))

leg\_room\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Leg room service"]))

baggage\_handling = l.fit\_transform(list(Airline\_satisfaction\_data["Baggage handling"]))

checkin\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Checkin service"]))

inflight\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight service"]))

cleanliness = l.fit\_transform(list(Airline\_satisfaction\_data["Cleanliness"]))

departure\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Departure Delay in Minutes"]))

arrival\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Arrival Delay in Minutes"]))

satisfaction = l.fit\_transform(list(Airline\_satisfaction\_data["satisfaction"]))

x = list(zip(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay))

y = list(satisfaction)

# Test options and evaluation metric

num\_folds = 5

seed = 7

scoring = 'accuracy'

# Model Test/Train

# Splitting what we are trying to predict into 4 different arrays -

# X train is a section of the x array(attributes) and vise versa for Y(features)

# The test data will test the accuracy of the model created

x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

#splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows

# ------------------------------------------------------------------------------------------------------------------------

# Check with different Scikit-learn classification algorithms

models = []

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

# evaluate each model in turn

results = []

names = []

import time

tt = time.time()

for name, model in models:

kfold = KFold(n\_splits=num\_folds,shuffle=True,random\_state=seed)

cv\_results = cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

msg += '\n'

print(msg)

print('IT TOOK FIRST PLOT PROCCESSING: ', time.time() - tt, ' SECONDS')

# Compare Algorithms' Performance

fig = pyplot.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

pyplot.boxplot(results)

ax.set\_xticklabels(names)

pyplot.show()

# ------------------------------------------------------------------------------------------------------------------------

# Make predictions on validation/test dataset

dt = DecisionTreeClassifier()

nb = GaussianNB()

gb = GradientBoostingClassifier()

rf = RandomForestClassifier()

best\_model = rf

best\_model.fit(x\_train, y\_train)

y\_pred = best\_model.predict(x\_test)

model\_accuracy = accuracy\_score(y\_test, y\_pred)

print("Best Model Accuracy Score on Test Set:", model\_accuracy)

# # ==============================================================================================================================================================================

# import pickle

# # Fit the model on training set

# # model = LogisticRegression()

# rf = RandomForestClassifier()

# rf.fit(x\_train, y\_train)

# # save the model to disk

# filename = 'finalized\_model.sav'

# pickle.dump(rf, open(filename, 'wb'))

# # # some time later...

# # # load the model from disk

# # loaded\_model = pickle.load(open(filename, 'rb'))

# # result = loaded\_model.score(X\_test, Y\_test)

# # print(result)

# # ==============================================================================================================================================================================

#Model Evaluation Metric 1

print(classification\_report(y\_test, y\_pred))

#Model Evaluation Metric 2

#Confusion matrix

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

cm = confusion\_matrix(y\_test, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot()

plt.show()

#Model Evaluation Metric 3

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

best\_model = rf

best\_model.fit(x\_train, y\_train)

# # ==============================================================================================================================================================================

print('\n-------------------------------------------')

from pprint import pprint

pprint(x\_test[:2])

print(best\_model.classes\_)

print(best\_model.predict(x\_test)[:10])

print(best\_model.predict\_log\_proba(x\_test)[:10])

print(best\_model.predict\_proba(x\_test)[:10])

print('\n-------------------------------------------')

# # ==============================================================================================================================================================================

rf\_roc\_auc = roc\_auc\_score(y\_test,best\_model.predict(x\_test))

fpr,tpr,thresholds = roc\_curve(y\_test, best\_model.predict\_proba(x\_test)[:,1])

plt.figure()

plt.plot(fpr,tpr,label = 'Random Forest(area = %0.2f)'% rf\_roc\_auc)

plt.plot([0,1],[0,1],'r--')

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.05])

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc='lower right')

plt.savefig('LOC\_ROC')

plt.show()

#Check actual/ground truth vs predicted diagnosis

for x in range(len(y\_pred)):

print("Predicted: ", y\_pred[x], "Actual: ", y\_test[x], "Data: ", x\_test[x],)

Main.py

from satisfaction\_gui import SatisfactionPredictorGUI

from tkinter import ttk

from tkinter import Tk

import pickle

gui = SatisfactionPredictorGUI()

gui.mainloop()

You need to run the file called main.py for Implementation and Deployment Task

A picture containing diagram

Description automatically generated

This is the html index code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles/main.css')}}">

<!---<meta http-equiv="X-UA-Compatible" content="ie=edge" />-->

<title>HD Prediction</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

<script src="{{ url\_for('static', filename='styles/parallax.min.js')}}"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.min.css" />

</head>

<!--

<nav>

<ul>

<li><a href="#home">Home</a></li>

<li style="float:right"><a class="active" href="#about">About</a></li>

</ul>

</nav>

-->

<div class="parallax-window" data-parallax="scroll" data-image-src="{{url\_for('static', filename='images/airplane-2.jpg')}}">

<ul>

<li><a href="/">Home</a></li>

<li style="float:right"><a class="active" href="about">About</a></li>

</ul>

</div>

<style>

.parallax-window {

min-height: 500px;

background: transparent;

}

</style>

<!-- <div class="containerforh">

<br>

<div class="quotemain">"Prevention is better Than Cure"</div>

-->

</div>

<body>

<div class="explaination">

<h2 class="abouth3"> About This Platform</h2>

<p class="info">

This platform is designed to predict Passenger Satisfaction. The data

being used to aid the algorithm has been taken from the<a class="colorhisa"

href="https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction?resource=download">Kaggle Airline Passenger Satisfaction</a> data set. The intended

audience for this application are airlines who understand the input values and

have the provided data from passengers. For more information

on the data attributes associated with our calculation, <a class="colorthisa"

href="https://towardsdatascience.com/predicting-satisfaction-of-airline-passengers-with-classification-76f1516e1d16">Click Here.</a>

</p>

</div>

<h3>Airline Satisfaction Prediction Platform</h3>

<!-- Form -->

<div class="complete\_form">

<form action="/send" method="POST">

<div class="container1">

<label for="age\_entry">Age:</label>

<input type="number" required

placeholder="Enter Your Age..."

name="age\_entry" />

<label for="gender\_entry">Gender: </label>

<select name="gender\_entry" required>

<option value="male">Male</option>

<option value="female">Female</option>

</select>

<label for="class\_entry">Class:</label>

<select name="class\_entry" required>

<option value="0">Eco</option>

<option value="2">Eco Plus</option>

<option value="1">Business</option>

</select>

<label for="inflight\_wifi\_entry">Inflight wifi service:</label>

<select name="inflight\_wifi\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="departure\_arrival\_entry">Departure/Arrival time convenient:</label>

<select name="departure\_arrival\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="ease\_of\_booking\_entry">Ease of Online booking:</label>

<select name="ease\_of\_booking\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="gate\_location\_entry">Gate location:</label>

<select name="gate\_location\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="food\_drink\_entry">Food and drink:</label>

<select name="food\_drink\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="online\_boarding\_entry">Online boarding:</label>

<select name="online\_boarding\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="seat\_comfort\_entry">Seat comfort:</label>

<select name="seat\_comfort\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

</div>

<div class="container2">

<label for="inflight\_entertainment\_entry">Inflight entertainment:</label>

<select name="inflight\_entertainment\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="onboard\_service\_entry">On-board service:</label>

<select name="onboard\_service\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="legroom\_service\_entry">Leg room service:</label>

<select name="legroom\_service\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="baggage\_handling\_entry">Baggage handling:</label>

<select name="baggage\_handling\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="checkin\_service\_entry">Checkin service:</label>

<select name="checkin\_service\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="inflight\_service\_entry">Inflight service:</label>

<select name="inflight\_service\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="cleanliness\_entry">Cleanliness:</label>

<select name="cleanliness\_entry" required>

<option value="0">0</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

</select>

<label for="departure\_delay\_entry">Departure Delay in Minutes:</label>

<input type="number" required

placeholder="Departure Delay in Minutes"

name="departure\_delay\_entry" />

<label for="arrival\_delay\_entry">Arrival Delay in Minutes:</label>

<input type="number" required

placeholder="Arrival Delay in Minutes"

name="arrival\_delay\_entry" />

<br />

<br />

<input type="submit" value="Predict" id="calc\_btn" />

<br />

</div>

<div class="alert">

{{ predict }}

</div>

</form>

</div>

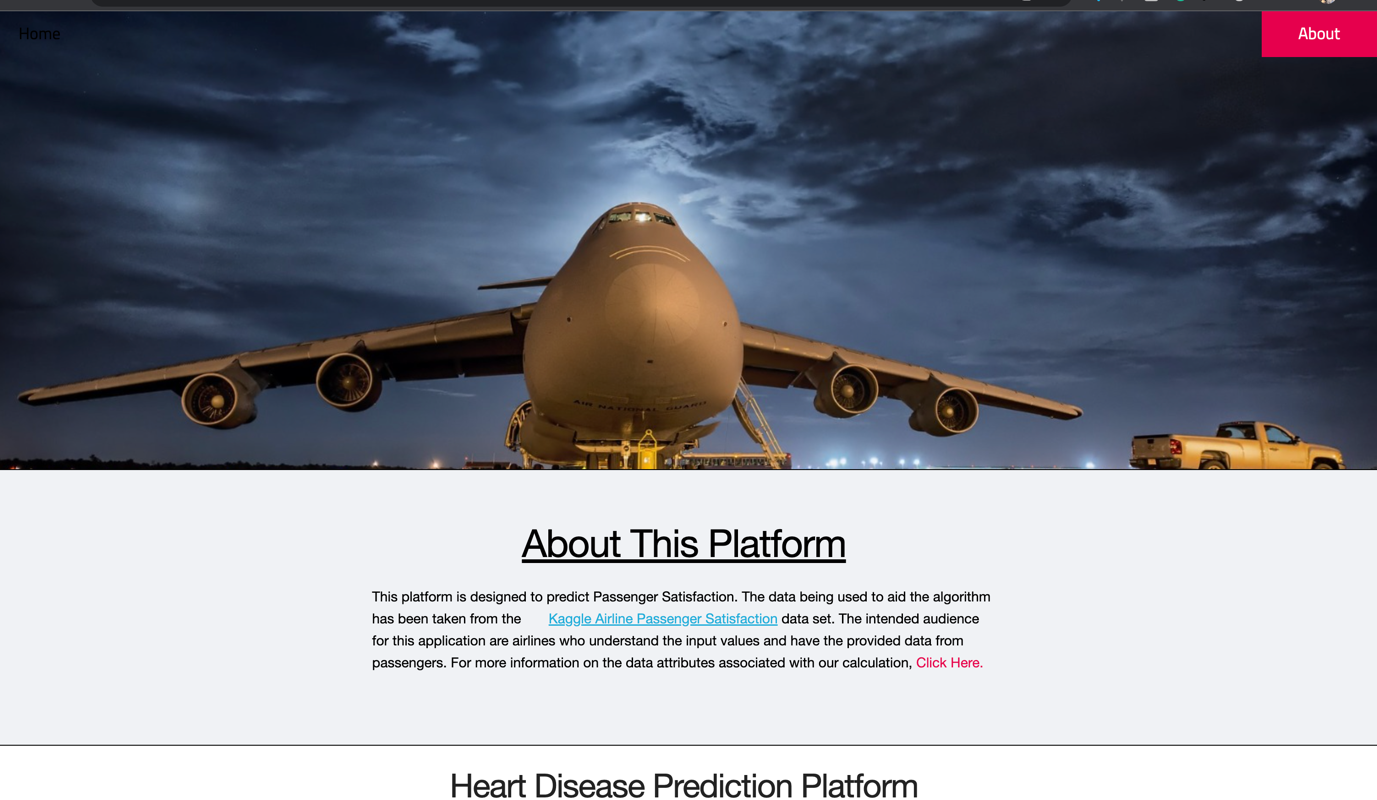
</body>

<footer>

<p>© 2023 - University of Canberra ST1\_Capstone - Project. All Rights Reserved.

</footer>

</html>



This is the html about code

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='styles/about.css')}}">

<meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

<div class="about-section">

<h1>About Us</h1>

<p>This project aim to develop a machine </p>

<p> learning platform to make predictions on passenger predictions.</p>

</div>

<h2 style="text-align:center">Our Team</h2>

<div class="row">

<div class="column">

<div class="card">

<img src="{{url\_for('static', filename='images/airplane-1.jpg')}}" alt="nani" style="width:100%; height:500px;">

<div class="container">

<h2>Ugyal Wang</h2>

<p class="title">Team Member</p>

Hello!<br />

I am a student in Information Technology.

I am keen to be part of an innovative team

where I have the opportunity to grow my

skillset and make a valuable contribution.

<p>

u3248529@canberra.edu.au

</p>

<p><button class="button">Contact</button></p>

</div>

</div>

</div>

<!-- <div class="column">

<div class="card">

<img src="{{url\_for('static', filename='images/luffy.jpg')}}" alt="Shams" style="width:100%; height:500px;">

<div class="container">

<h2>Student Name 2</h2>

<p class="title">Team Member</p>

<p>

Hello!<br />

I am a student in Information Technology.

I am keen to be part of an innovative team

where I have the opportunity to grow my

skillset and make a valuable contribution.

</p>

<p>u7654321@canberra.edu.au</p>

<p><button class="button">Contact</button></p>

</div>

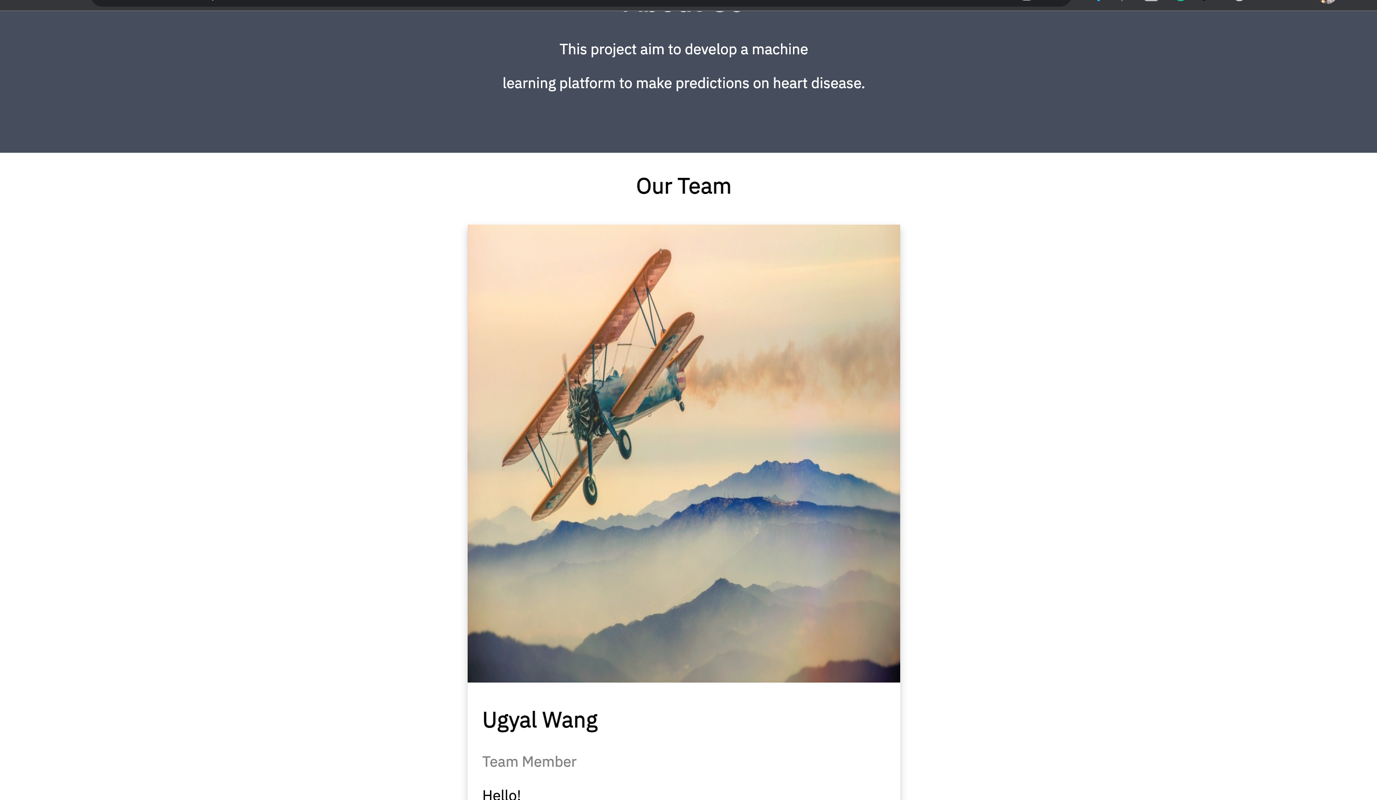
</div>

</div>

</div> -->

</body>

</html>



This is the webpageapp code

from argparse import ArgumentParser

from flask import Flask, Blueprint, render\_template, request

from werkzeug.middleware.proxy\_fix import ProxyFix

# from cvd\_model import \*

# print('predict: ', predict)

predict = 'rf'

import pandas as pd

import sklearn

# Load libraries

from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

appweb = Blueprint('hello', \_\_name\_\_)

@appweb.route('/')

def home():

return render\_template("index.html")

# ------------------------------------------------------------------------------------------------- PREPARING MODEL

Airline\_satisfaction\_data = pd.read\_csv("train.csv")

# Attribute to be predicted

predict = "Airline\_satisfaction\_data"

# Dataset/Column to be Predicted, X is all attributes and y is the features

#x = np.array(heart\_data.drop([predict], 1)) # Will return a new data frame that doesnt have hd in it

#y = np.array(heart\_data[predict])

l = preprocessing.LabelEncoder()

age = l.fit\_transform(list(Airline\_satisfaction\_data["Age"]))#AGE OF THE PASSENGER

Gender = l.fit\_transform(list(Airline\_satisfaction\_data["Gender"]))

id = l.fit\_transform(list(Airline\_satisfaction\_data["id"]))

customer\_type = l.fit\_transform(list(Airline\_satisfaction\_data["Customer Type"]))

type\_of\_travel = l.fit\_transform(list(Airline\_satisfaction\_data["Type of Travel"]))

class\_ = l.fit\_transform(list(Airline\_satisfaction\_data["Class"]))

flight\_distance = l.fit\_transform(list(Airline\_satisfaction\_data["Flight Distance"]))

inflight\_wifi\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight wifi service"]))

departure\_arrival\_time\_convenient = l.fit\_transform(list(Airline\_satisfaction\_data["Departure/Arrival time convenient"]))

ease\_of\_online\_booking = l.fit\_transform(list(Airline\_satisfaction\_data["Ease of Online booking"]))

gate\_location = l.fit\_transform(list(Airline\_satisfaction\_data["Gate location"]))

food\_and\_drink = l.fit\_transform(list(Airline\_satisfaction\_data["Food and drink"]))

online\_boarding = l.fit\_transform(list(Airline\_satisfaction\_data["Online boarding"]))

seat\_comfort = l.fit\_transform(list(Airline\_satisfaction\_data["Seat comfort"]))

inflight\_entertainment = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight entertainment"]))

onboard\_service = l.fit\_transform(list(Airline\_satisfaction\_data["On-board service"]))

leg\_room\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Leg room service"]))

baggage\_handling = l.fit\_transform(list(Airline\_satisfaction\_data["Baggage handling"]))

checkin\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Checkin service"]))

inflight\_service = l.fit\_transform(list(Airline\_satisfaction\_data["Inflight service"]))

cleanliness = l.fit\_transform(list(Airline\_satisfaction\_data["Cleanliness"]))

departure\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Departure Delay in Minutes"]))

arrival\_delay = l.fit\_transform(list(Airline\_satisfaction\_data["Arrival Delay in Minutes"]))

satisfaction = l.fit\_transform(list(Airline\_satisfaction\_data["satisfaction"]))

x = list(zip(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay))

y = list(satisfaction)

# Test options and evaluation metric

num\_folds = 5

seed = 7

scoring = 'accuracy'

# Model Test/Train

# Splitting what we are trying to predict into 4 different arrays -

# X train is a section of the x array(attributes) and vise versa for Y(features)

# The test data will test the accuracy of the model created

# self.x\_train, self.x\_test, self.y\_train, self.y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

model = RandomForestClassifier()

model.fit(x\_train, y\_train)

# -------------------------------------------------------------------------------------------------

@appweb.route('/send', methods=['POST'])

def send(predict=predict):

global model

if request.method == 'POST':

age\_entry = int(request.form['age\_entry'])

gender\_entry = request.form['gender\_entry']

class\_entry = request.form['class\_entry']

inflight\_wifi\_entry = int(request.form['inflight\_wifi\_entry'])

departure\_arrival\_entry = int(request.form['departure\_arrival\_entry'])

ease\_of\_booking\_entry = int(request.form['ease\_of\_booking\_entry'])

gate\_location\_entry = int(request.form['gate\_location\_entry'])

food\_drink\_entry = int(request.form['food\_drink\_entry'])

online\_boarding\_entry = int(request.form['online\_boarding\_entry'])

seat\_comfort\_entry = int(request.form['seat\_comfort\_entry'])

inflight\_entertainment\_entry = int(request.form['inflight\_entertainment\_entry'])

onboard\_service\_entry = int(request.form['onboard\_service\_entry'])

legroom\_service\_entry = int(request.form['legroom\_service\_entry'])

baggage\_handling\_entry = int(request.form['baggage\_handling\_entry'])

checkin\_service\_entry = int(request.form['checkin\_service\_entry'])

inflight\_service\_entry = int(request.form['inflight\_service\_entry'])

cleanliness\_entry = int(request.form['cleanliness\_entry'])

departure\_delay\_entry = int(request.form['departure\_delay\_entry'])

arrival\_delay\_entry = int(request.form['arrival\_delay\_entry'])

if(gender\_entry == "male"):

gender\_entry = 1

else:

gender\_entry = 0

# ------------------------------------------------------------------------------------------------- USING MODEL

# input\_arr = [(age, Gender, class\_, inflight\_entertainment, departure\_arrival\_time\_convenient, ease\_of\_online\_booking, gate\_location, food\_and\_drink, online\_boarding, seat\_comfort, inflight\_entertainment, onboard\_service, leg\_room\_service, baggage\_handling, checkin\_service, inflight\_service, cleanliness, departure\_delay, arrival\_delay)]

# NOTE order maybe is wrong... gonna write in the way gotten'

input\_arr = [(age\_entry, gender\_entry, class\_entry, inflight\_wifi\_entry, departure\_arrival\_entry, ease\_of\_booking\_entry, gate\_location\_entry, food\_drink\_entry, online\_boarding\_entry, seat\_comfort\_entry, inflight\_entertainment\_entry, onboard\_service\_entry, legroom\_service\_entry, baggage\_handling\_entry, checkin\_service\_entry, inflight\_service\_entry, cleanliness\_entry, departure\_delay\_entry, arrival\_delay\_entry)]

print('predicting')

predicted\_satisfaction\_value = model.predict\_proba(input\_arr)

print(predicted\_satisfaction\_value)

print('prediction over')

\_str = f"Predicted satisfaction: {str(100 - predicted\_satisfaction\_value[0][0] \* 100)}%" # NOTE don't know the reason but, by the values I enter, I guess should be 100 - whatever we find

print(\_str)

# -------------------------------------------------------------------------------------------------

# =============================================================================================================================================================================

# # Accuracy of Model

# model.fit(x\_train, y\_train) #<-- this line

# acc = model.score(x\_train, y\_train)

# predict\_real = model.predict([[patient\_age,patient\_sex,patient\_chest\_pain,

# patient\_resting\_bp,patient\_sereum\_chol,

# patient\_fasting\_bs,patient\_resting\_egg,

# patient\_max\_heartrate,patient\_exercise\_induced\_angina,

# patient\_oldpeak,patient\_slope,patient\_number\_vessels,

# patient\_thalassemia]])

# if(predict\_real == [0]):

# predict = "The result returned with " + str(round(acc,2)\*100) + "% accuracy and you have a lower chance of getting heart disease"

# else:

# predict = "The result returned with " + str(round(acc,2)\*100) + "% accuracy and you have a higher chance of getting heart disease"

# ===========================================================================================================================================================================

predict = \_str

return render\_template('index.html', predict=predict)

else:

return render\_template('index.html', predict=predict)

@appweb.route('/about')

def about():

return render\_template("about.html")

if \_\_name\_\_ == '\_\_main\_\_':

# arg parser for the standard anaconda-project options

parser = ArgumentParser(prog="home",

description="Simple Flask Application")

parser.add\_argument('--anaconda-project-host', action='append', default=[],

help='Hostname to allow in requests')

parser.add\_argument('--anaconda-project-port', action='store', default=8086, type=int,

help='Port to listen on')

parser.add\_argument('--anaconda-project-iframe-hosts',

action='append',

help='Space-separated hosts which can embed us in an iframe per our Content-Security-Policy')

parser.add\_argument('--anaconda-project-no-browser', action='store\_true',

default=False,

help='Disable opening in a browser')

parser.add\_argument('--anaconda-project-use-xheaders',

action='store\_true',

default=False,

help='Trust X-headers from reverse proxy')

parser.add\_argument('--anaconda-project-url-prefix', action='store', default='',

help='Prefix in front of urls')

parser.add\_argument('--anaconda-project-address',

action='store',

#default='0.0.0.0',

help='IP address the application should listen on.')

args = parser.parse\_args()

app = Flask(\_\_name\_\_)

app.register\_blueprint(appweb, url\_prefix = args.anaconda\_project\_url\_prefix)

app.config['PREFERRED\_URL\_SCHEME'] = 'https'

app.wsgi\_app = ProxyFix(app.wsgi\_app)

app.run(host=args.anaconda\_project\_address, port=args.anaconda\_project\_port)