# Introduction

The flight ticket buying system is to purchase a ticket many days prior to flight take-off so as to stay away from the effect to the most extreme charge. Mostly, aviation routes don’t agree this procedure. Plane organizations may diminish the cost at the time, they need to build the market and at the time when the tickets are less accessible. They may maximize the costs. So, the cost may rely upon different factors. To foresee the costs this venture uses AI to exhibit the ways of flight tickets after some time. All organization have the privilege and opportunity to change booking.

# OVER VIEW

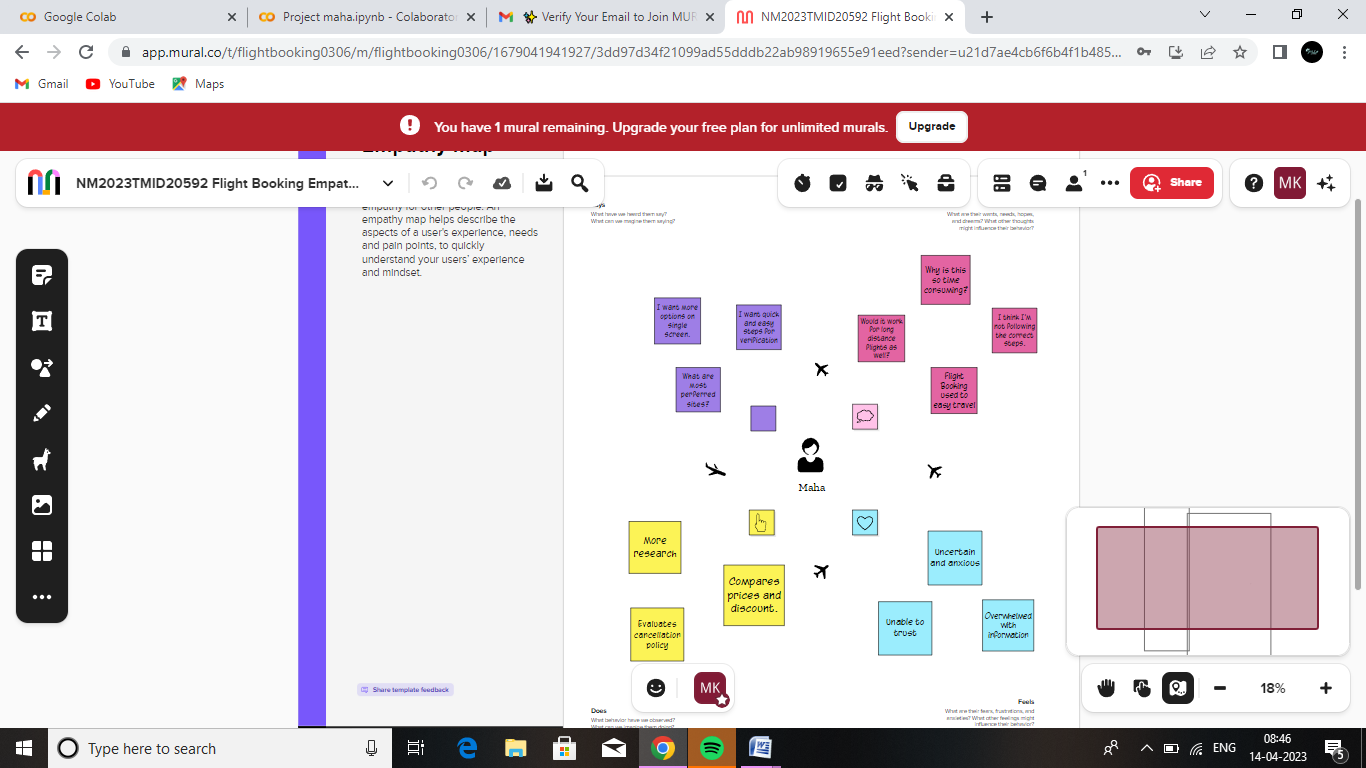
The Flight ticket prices increase or decrease every now and then depending on various factors like timing of the flights, destination, duration of flights. In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights. Optimal timing for airline ticket purchasing from the consumer’s perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we majorly targeted to uncover underlying tends of flight prices in India using historical data and also to suggest the best time to buy a flight ticket.

## PURPOSES

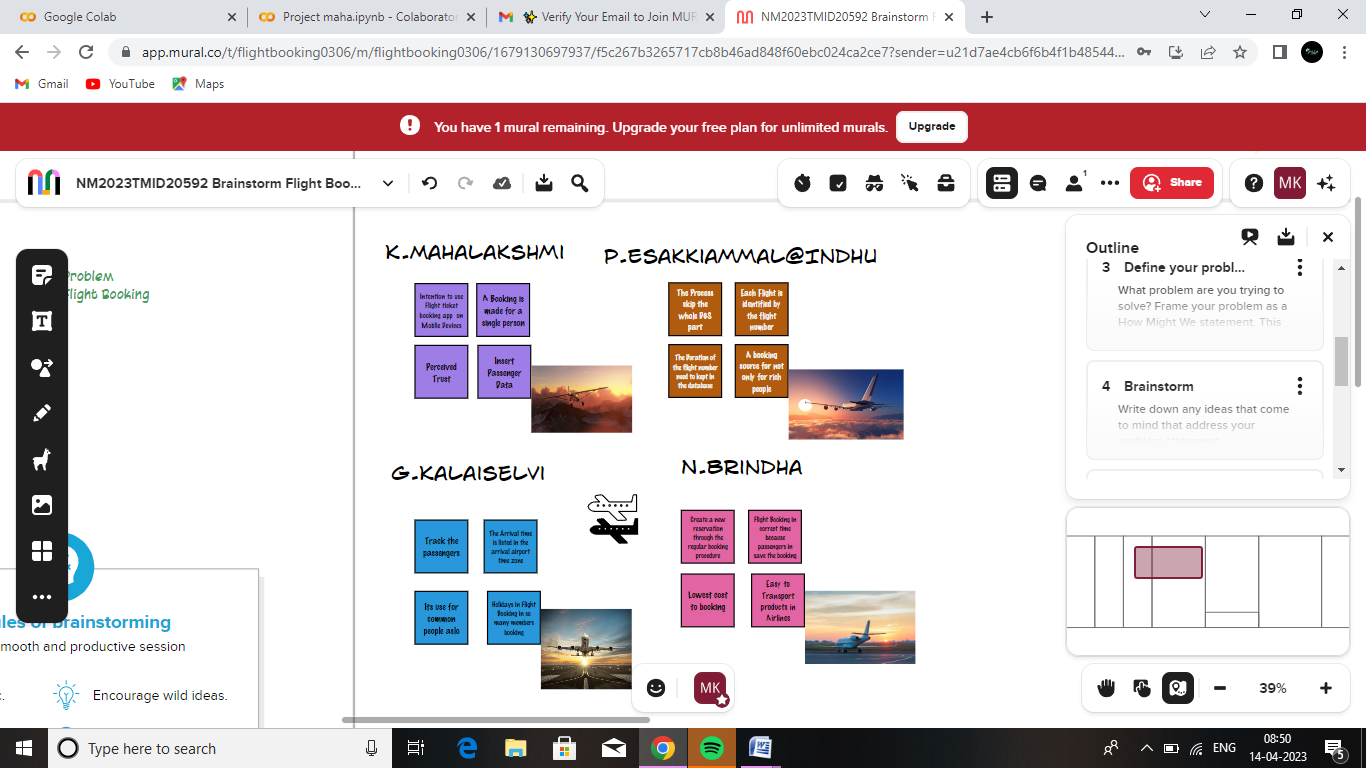
* Motivation is to help people who tends to pay more for the flight fare ticket and for those who are naive to this booking tickets process. This will also help us to get more exposure to the machine learning techniques that will help us to excel and improve in the existing skills.
* To get effective price for the customers.
* Make UI user friendly.
* Use of various ML methods to know more about dataset and get accurate results.

### PROBLEM DEFINITION & DESIGN THINKING

EMPATHY MAP:



## BRAINSTROM:



## ADVANTAGES

* Traveler get the fare prediction handy using which it’s easy to decide the airlines.
* Saves time in searching /deciding for airlines.

## DISADVANTAGES

* Improper data will result in incorrect fare predictions.

## CONCLUSION

Machine Learning algorithms are applied on the dataset to predicted the dynamic fare of flights. This gives the predicted values of flight fare to get a flight ticket at minimum cost. The values of R-squared obtained from the algorithm give the accuracy of the model. In the future, If more data could be accessed such as the current availability of seats, the predicted results will be more accurate. Finally, we conclude that this methodology is not preferred for performing this project. We can add more methods, more data for more accurate results.

## APPENDIX

#### SOURCE CODE

# 

IMPORTING PACKAGES

import matplotlib .pyplot as plt

import seaborn as sns

import pandas as pd

IMPORTING DATASET

train\_data = pd.read\_excel("/content/Data\_Train (1).xlsx")

pd.set\_option('display.max\_columns', None)

train\_data.head()

HEAD

train\_data.head()

TAIL

train\_data.head()

DATA PREPROCESSING

train\_data.info()

train\_data.isnull().sum()

ANALYISING THE CORRELTION FLIGHT ATTRIBUTES WITH OTHER ATTRIBUTES

plt.figure(figsize = (18,18))

sns.heatmap(train\_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()from sklearn.model\_selection import train\_test\_split

IMPORTING RANDOMFORESTREGRESSOR

from sklearn.ensemble import RandomForestRegressor

reg\_rf = RandomForestRegressor()

reg\_rf.fit(X\_train, y\_train)

TRAINING AND TESTING THE RECORDS OF DATASET FOR PREDICTION

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42

y\_pred = reg\_rf.predict(X\_test)

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()tances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show() ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show() VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

IMPORTING DATASET

test\_data = pd.read\_excel("/content/Data\_Train (1).xlsx")

HEAD

data\_train ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show().head()

TAIL

data\_train.shape

DATA PREPROCESSING

train\_data.info()

train\_data.isnull().sum()

ANALYSING THE CORRELATION OF FLIGHT ATTRIBUTESWITH OTHER ATTRIBUTES

y = data\_t

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()rain.iloc[:, 1]

y.head()

SPLITTING RECORDS FOR TRAING AND TESTING

from sklearn.model\_selection import train\_test\_split

TRAINING AND TESTING THE RECORDSS OF DATASET FOR PREDICTION

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

IMPORT DECISIONTREEREGRESSOR

from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor(random\_state = 0)

regressor.fit(X, y)

TRAINING AND TESTING THE RECORDS OF DATASET FOR PREDICTION

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42

y\_pred = reg\_rf.predict(X\_test)

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

IMPORTING DATASET

test\_data = pd.read\_excel("/content/Data\_Train (1).xlsx")

HEAD

data\_train.head()

TAIL

data\_train.shape

DATA PREPROCESSING

train\_data.info()

train\_data.isnull().sum()

ANALYSING THE CORRELATION OF FLIGHT ATTRIBUTESWITH OTHER ATTRIBUTES

y = data\_train.iloc[:, 1]

y.head()

SPLITTING RECORDS FOR TRAING AND TESTING

from sklearn.model\_selection import train\_test\_split

TRAINING AND TESTING THE RECORDSS OF DATASET FOR PREDICTION

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

ERROR OF PREDICTIOIN

print('MAE:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)

)

CLASSIFICATION REPORT

from sklearn import metrics

PRINTING THE FINAL ACCURACY SCORE OF PREDICTION

y\_pred = regressor.predict(X\_test)

VISUALIZING THE ACCURACY OF PREDICTED RESULT

plt.figure(figsize = (12,8))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()

## RESULT

