SUBMITTED BY

OPTIMIZING FLIGHT BOOKING DECISIONS THROUGH MACHINE LEARNING PRICE PREDICTIONS

FLIGHT BOOKING DECISION

**K.SANDHIYA**

**S.SANTHIYA**

**R.SHAKTHIPRIYA**

**A.V.SHAMEENA**

|  |  |
| --- | --- |
| **S.NO** | **CONTENT** |
| 1 | **INTRODUCTION** |
| 2 | PROBLEM DEFINITION AND DESIGN THINKING |
| **3** | RESULT |
| **4** | **ADVANTAGES AND DISADVANTAGES** |
| **5** | APPLICATIONS |
| 6 | **CONCLUSION** |
| **7** | **FUTURE SCOPE** |
| **8** | **APPENDIX** |

**1.INTRODUCTION**

People who work frequently travel through flight will have better knowledge on best discount

and right time to buy the ticket. For the business purpose many airline companies change prices

according to the seasons or time duration. They will increase the price when people travel

more. Estimating the highest prices of the airlines data for the route is collected with features

such as Duration, Source, Destination, Arrival and Departure. Features are taken from chosen

dataset and in the price wherein the airline price ticket costs vary overtime.

**1.1 OVERVIEW**

The average price for an airline ticket in the united states in November 2022 was $280,about

35% higher than November 2021,according to statistics from the U.S.Bureau of Labor statistics

.November’s average price was down from may 2022 when the average price for a domestic

flight hit an all-time high of $336.

A flighty price prediction application which predicts fares of flight for a particular date based

on various parameters like source,destination,stops & airline.

**1.2 PURPOSE**

The main objective of the project is , Features are taken from chosen dataset and in the price

wherein the airline price ticket costs vary overtime. we have implemented flight price

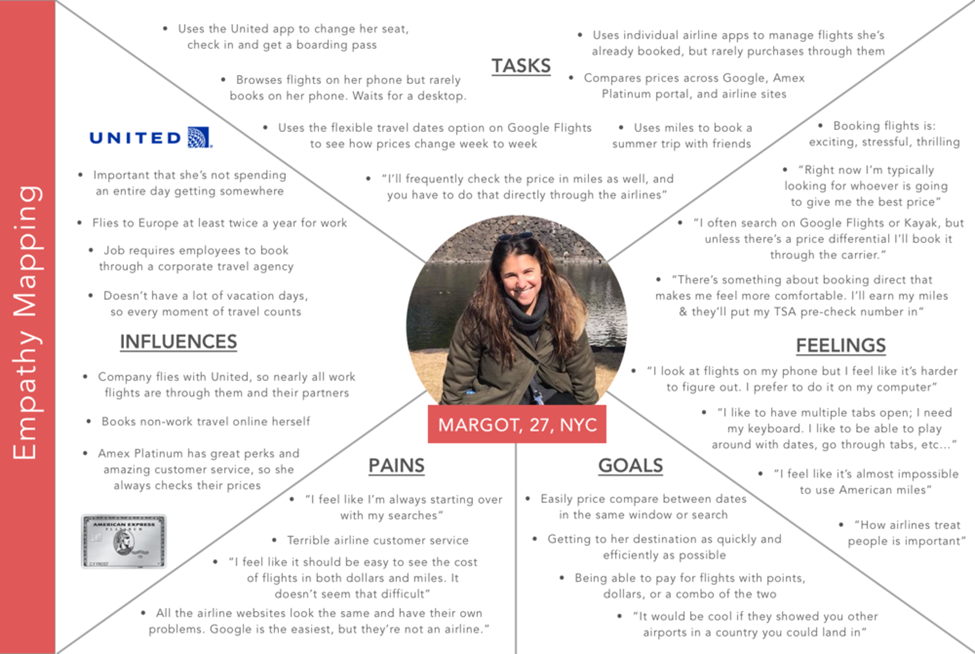
prediction for users by using KNN, decision tree and random forest algorithms. Random Forest

shows the best accuracy of 80% for predicting the flight price. also, we have done correlation

tests and metrics for the statistical analysis.

**2.PROBLEM DEFINITION AND DESIGN THINKING**

2.1 EMPATHY MAP



**3.RESULT**

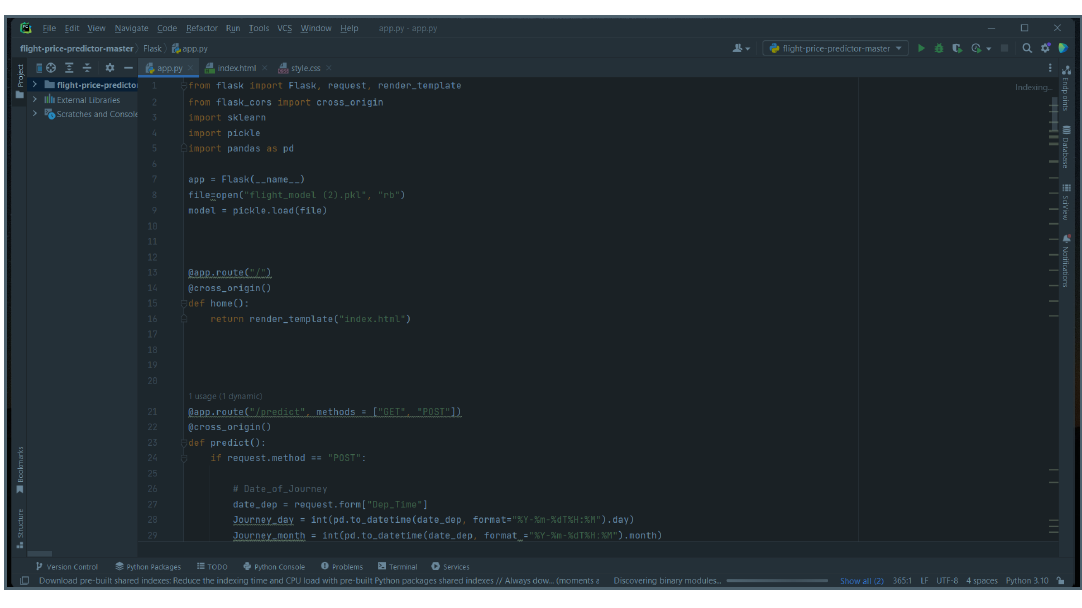


Fig 3.1 Flask code

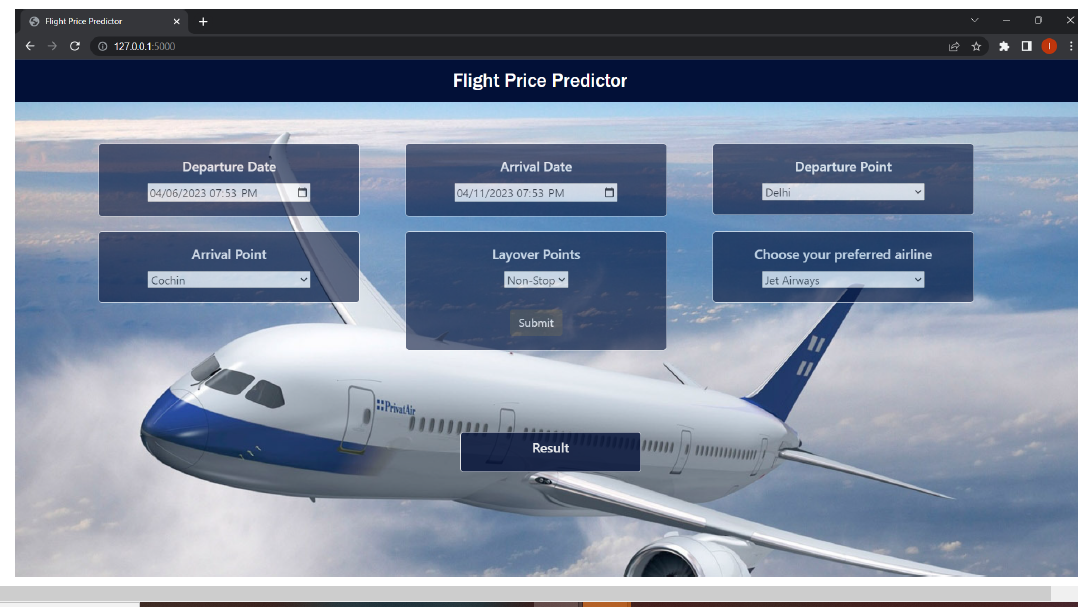


Fig 3.2 Home page for Flight Price Prediction

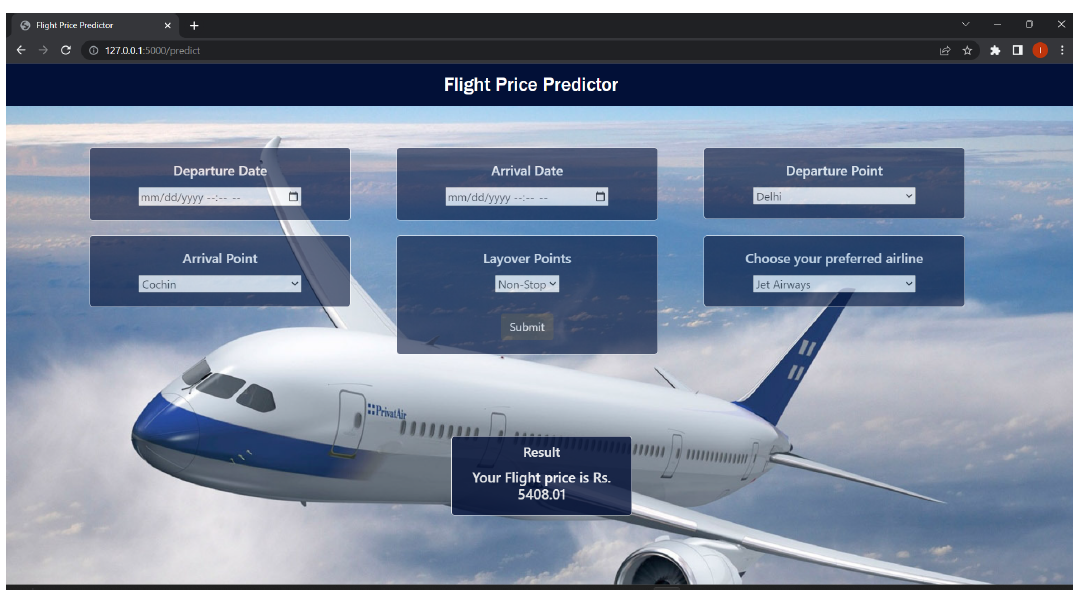


Fig 3.3 Predicting page of Flight Price Prediction

**4.ADVANTAGES AND DISADVANTAGES**

**ADVANTAGES**

1. Traveler get the fare prediction handy using which it’s easy to decide the airlines.

2. Saves time in searching / deciding for airlines.

**DISADVANTAGES**

3. Improper data will result in incorrect fare predictions.

**5.APPLICATIONS**

1. Make traveling easier

2. Airfare tracking

3. flight search and airfare prediction

4. Airfare tracking and hotel booking.

**6.CONCLUSION**

In this project is to forecast the average flight price at the business segment level. We used

training data to train the training data and test data to test it. These records were used to extract

a number of characteristics. Our suggested model can estimate the quarterly average flight

price using attribute selection strategies. To the highest possible standard, much prior studies

into flight price prediction using the large dataset depended on standard statistical approaches,

which have their own limitations in terms of underlying issue estimates and hypotheses. To our

knowledge, no other research has included statistics from holidays, celebrations, stock market

price fluctuations, depression, fuel price, and socioeconomic information to estimate the air

transport market sector; nonetheless, there are numerous restrictions. As example, neither of

the databases provide precise information about ticket revenue, including such departing and

arrival times and days of the week. This framework may be expanded in the future to also

include airline tickets payment details, that can offer more detail about each area, such as

timestamp of entry and exit, seat placement, covered auxiliary items, and so on. By merging

such data, it is feasible to create a more robust and complete daily and even daily flight price

forecast model. Furthermore, a huge surge of big commuters triggered by some unique events

might alter flight costs in a market sector. Thus, incident data will be gathered from a variety of

sources, including social media sites and media organizations, to supplement our forecasting

models. We will also examine specific technological Models, such as Deeper Learning

methods, meanwhile striving to enhance existing models by modifying their hyper-parameters

to get the optimum design for airline price prediction.

**7.FUTURESCOPE**

1. More routes can be added and the same analysis can be expanded to major

airports and travel routes in india.

2. The analysis can be done by increasing the data points and increasing the

historical data used. That will train the model better giving better accuracies

and more savings.

3. More rules can be added in the rule-based learning based on our

understanding of the industry, also incorporating the offer periods given by the

airlines .

4. Developing a more user-friendly interface for various routes giving more

flexibility to the users.

**APPENDIX**

**A Source Code of Flask:**

from flask import Flask, request, render\_template

from flask\_cors import cross\_origin

import sklearn

import pickle

import pandas as pd

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

file=open("flight\_model (2).pkl", "rb")

model = pickle.load(file)

@app.route("/")

@cross\_origin()

def home():

return render\_template("index.html")

@app.route("/predict", methods = ["GET", "POST"])

@cross\_origin()

def predict():

if request.method == "POST":

# Date\_of\_Journey

date\_dep = request.form["Dep\_Time"]

Journey\_day = int(pd.to\_datetime(date\_dep, format="%Y-%m-%dT%H:%M").day)

Journey\_month = int(pd.to\_datetime(date\_dep, format ="%Y-%m-%dT%H:%M").month)

# print("Journey Date : ",Journey\_day, Journey\_month)

# Departure

Dep\_hour = int(pd.to\_datetime(date\_dep, format ="%Y-%m-%dT%H:%M").hour)

Dep\_min = int(pd.to\_datetime(date\_dep, format ="%Y-%m-%dT%H:%M").minute)

# print("Departure : ",Dep\_hour, Dep\_min)

# Arrival

date\_arr = request.form["Arrival\_Time"]

Arrival\_hour = int(pd.to\_datetime(date\_arr, format ="%Y-%m-%dT%H:%M").hour)

Arrival\_min = int(pd.to\_datetime(date\_arr, format ="%Y-%m-%dT%H:%M").minute)

# print("Arrival : ", Arrival\_hour, Arrival\_min)

# Duration

dur\_hour = abs(Arrival\_hour - Dep\_hour)

dur\_min = abs(Arrival\_min - Dep\_min)

# print("Duration : ", dur\_hour, dur\_min)

# Total Stops

Total\_stops = int(request.form["stops"])

# print(Total\_stops)

# Airline

# AIR ASIA = 0 (not in column)

airline=request.form['airline']

if(airline=='Jet Airways'):

Jet\_Airways = 1

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='IndiGo'):

Jet\_Airways = 0

IndiGo = 1

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Air India'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 1

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Multiple carriers'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 1

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='SpiceJet'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 1

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Vistara'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 1

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='GoAir'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 1

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Multiple carriers Premium economy'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 1

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Jet Airways Business'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 1

Vistara\_Premium\_economy = 0

Trujet = 0

elif (airline=='Vistara Premium economy'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 1

Trujet = 0

elif (airline=='Trujet'):

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 1

else:

Jet\_Airways = 0

IndiGo = 0

Air\_India = 0

Multiple\_carriers = 0

SpiceJet = 0

Vistara = 0

GoAir = 0

Multiple\_carriers\_Premium\_economy = 0

Jet\_Airways\_Business = 0

Vistara\_Premium\_economy = 0

Trujet = 0

Source = request.form["Source"]

if (Source == 'Delhi'):

s\_Delhi = 1

s\_Kolkata = 0

s\_Mumbai = 0

s\_Chennai = 0

elif (Source == 'Kolkata'):

s\_Delhi = 0

s\_Kolkata = 1

s\_Mumbai = 0

s\_Chennai = 0

elif (Source == 'Mumbai'):

s\_Delhi = 0

s\_Kolkata = 0

s\_Mumbai = 1

s\_Chennai = 0

elif (Source == 'Chennai'):

s\_Delhi = 0

s\_Kolkata = 0

s\_Mumbai = 0

s\_Chennai = 1

else:

s\_Delhi = 0

s\_Kolkata = 0

s\_Mumbai = 0

s\_Chennai = 0

Source = request.form["Destination"]

if (Source == 'Cochin'):

d\_Cochin = 1

d\_Delhi = 0

d\_New\_Delhi = 0

d\_Hyderabad = 0

d\_Kolkata = 0

elif (Source == 'Delhi'):

d\_Cochin = 0

d\_Delhi = 1

d\_New\_Delhi = 0

d\_Hyderabad = 0

d\_Kolkata = 0

elif (Source == 'New\_Delhi'):

d\_Cochin = 0

d\_Delhi = 0

d\_New\_Delhi = 1

d\_Hyderabad = 0

d\_Kolkata = 0

elif (Source == 'Hyderabad'):

d\_Cochin = 0

d\_Delhi = 0

d\_New\_Delhi = 0

d\_Hyderabad = 1

d\_Kolkata = 0

elif (Source == 'Kolkata'):

d\_Cochin = 0

d\_Delhi = 0

d\_New\_Delhi = 0

d\_Hyderabad = 0

d\_Kolkata = 1

else:

d\_Cochin = 0

d\_Delhi = 0

d\_New\_Delhi = 0

d\_Hyderabad = 0

d\_Kolkata = 0

prediction=model.predict([[

Total\_stops,

Journey\_day,

Journey\_month,

Dep\_hour,

Dep\_min,

Arrival\_hour,

Arrival\_min,

dur\_hour,

dur\_min,

Air\_India,

GoAir,

IndiGo,

Jet\_Airways,

Jet\_Airways\_Business,

Multiple\_carriers,

Multiple\_carriers\_Premium\_economy,

SpiceJet,

Trujet,

Vistara,

Vistara\_Premium\_economy,

s\_Chennai,

s\_Delhi,

s\_Kolkata,

s\_Mumbai,

d\_Cochin,

d\_Delhi,

d\_Hyderabad,

d\_Kolkata,

d\_New\_Delhi

]])

output=round(prediction[0],2)

return render\_template('index.html', prediction\_result="Your Flight price is Rs.

{}".format(output))

return render\_template("index.html")

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)