

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590018, Karnataka



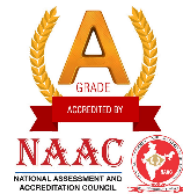
Python Application Programming
A mini project report on
Flight Ticket Price Predictor

Submitted By:

Pavithra K Tantry
Nisha Bhat Balanja

1GA18CS189
1GA18CS192

Under the Guidance of
Ms. Sushmitha S
Assistant Professor



Department of Computer Science and Engineering

(Accredited by NBA 2019-2022)

Global Academy of Technology

Rajarajeshwarinagar, Bengaluru – 560098

2020-2021

Global Academy of Technology
Department of Computer Science and Engineering



Certificate

This is to certify that the project entitled **“Flight Ticket Price Predictor”** is a bonafide work carried out by **Pavithra K Tantry (1GA18CS189), Nisha Bhat Balanja (1GA18CS192)** as a part of assignment in Python Applications Programming in Computer Science and Engineering during the year 2020-2021.

Max. marks	Marks obtained	Faculty Name and Signature
10		

ABSTRACT

Flight ticket prices are usually hard to guess. The price we see today might not be the same tomorrow. It can vary due to many factors like flight distance, competition, timing of flight, timing of purchase, number of flights, empty middle seats and many more. We can't keep all these factors in mind everytime we want to purchase tickets. This project is going to help us find flight tickets with the best price considering all our needs.

TABLE OF CONTENTS

TOPIC	PAGE NO
1. INTRODUCTION	2
1.1 Python Programming Language	2
1.2 Applications of Python	3
2. SYSTEM REQUIREMENTS	4
2.1 Software Requirements	4
2.2 Hardware Requirements	4
3. IMPLEMENTATION AND RESULTS	5-23
3.1 Project Code	5
3.2. Results	23
CONCLUSION	25
REFERENCES	25

ORGANIZATION OF THE REPORT

The report is divided into various chapters and is organized as follows:

Chapter 1: Introduction

This chapter includes a brief introduction to Python Programming Language and its applications.

Chapter 2: System requirements

This chapter includes details of hardware and software requirements necessary for the execution of the project.

Chapter 3: Implementation and Results

This chapter includes the program code of the project and the results of successful runs of the code.

Conclusion

This section includes the conclusion about the project.

References

This section includes the bibliographical references used for the development of the project.

CHAPTER 1

INTRODUCTION

1.1. Python Programming language

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

The foundation for Python was laid in the late 1980s. Its implementation was started in December 1989 by Guido van Rossum in Netherland. In February 1991, he published the code to alt.sources.

In 1994, Python 1.0 was released with new features like lambda, map, filter and reduce.

Python 2.0 added new features such as list comprehensions, garbage collection systems.

Python 3.0 was released on December 3, 2008 and it was designed to rectify fundamental flaws of the language.

Python programming language is being updated regularly with new features and supports. There have been around 20 versions of Python updates released since 1994.

1.2. Applications of Python programming language

Python is a general purpose, open source, high level programming language and also provides a number of libraries and frameworks. Python has gained popularity because of its simplicity, easy syntax and user-friendly environment.

Since 2003, Python has consistently ranked in the top ten most popular programming languages. As of February 2020, it is the third most popular language behind Java and C. It was selected as Programming Language of the Year in 2007, 2010 and 2018.

Some of the applications of Python are:

- Desktop applications
- Web applications
- Data Science
- Artificial intelligence
- Machine learning
- Scientific Computing
- Robotics
- Internet of things (IOT)
- Gaming
- Mobile apps
- Data Analysis and Preprocessing

Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify and many more. Sites like Reddit are written entirely in Python.

Python can serve as a scripting language for web applications.

Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing with specialized libraries such as Biopython and Astropy providing domain-specific functionality. OpenCV has python bindings with a rich set of features for computer vision and image processing.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

Jupyter Notebook / Google Colab

Front end : HTML

Back end : Python

2.2 Hardware Requirements

Processor : Intel core i3 and above

Operating system : Windows 10

RAM : 8GB

CHAPTER 3

IMPLEMENTATION AND RESULTS

3.1 About Project

Predicting flight prices without having proper ideas about particular airline company is near too impossible, especially when we want to book any flight real quick. Using Machine Learning approach it becomes quite easy as the model predicts how much you need to spend on flight expenses by getting rid of all unnecessary calculations and brain scratching thoughts. From the user's point of view this somehow proves to be a profitable project model as users can select from a wide range of airline companies of their choice and decide the budget of their ongoing journey without spending anything extra.

3.2 Project code

flight_price.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

train_data = pd.read_excel(r"Data_Train.xlsx")
pd.set_option('display.max_columns', None)
train_data.head()
train_data.info()
train_data.dropna(inplace = True)
train_data.isnull().sum()

train_data["Journey_day"] =
pd.to_datetime(train_data.Date_of_Journey, format="%d/%m/%Y").dt.day
```

```
train_data["Journey_month"] =
pd.to_datetime(train_data["Date_of_Journey"], format =
                                     "%d/%m/%Y").dt.month
train_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

train_data["Dep_hour"] =
pd.to_datetime(train_data["Dep_Time"]).dt.hour
train_data["Dep_min"] =
pd.to_datetime(train_data["Dep_Time"]).dt.minute
train_data.drop(["Dep_Time"], axis = 1, inplace = True)

train_data["Arrival_hour"] =
pd.to_datetime(train_data["Arrival_Time"]).dt.hour
train_data["Arrival_min"] =
pd.to_datetime(train_data["Arrival_Time"]).dt.minute
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
duration = list(train_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep = "m")[0]
                                              .split()[-1]))

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins
train_data.drop(["Duration"], axis = 1, inplace = True)
```

```
train_data["Airline"].value_counts()
# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data =
train_data.sort_values("Price", ascending = False), kind="boxen",
height = 6, aspect = 3)
plt.show()
```

```
# As Airline is Nominal Categorical data we will perform
OneHotEncoding
Airline = train_data[["Airline"]]
Airline = pd.get_dummies(Airline, drop_first= True)
Airline.head()
```

```
train_data["Source"].value_counts()
```

```
# Source vs Price
sns.catplot(y = "Price", x = "Source", data =
train_data.sort_values("Price", ascending = False), kind="boxen",
height = 4, aspect = 3)
plt.show()
```

```
# As Source is Nominal Categorical data we will perform
OneHotEncoding
Source = train_data[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
Source.head()
```

```
train_data["Destination"].value_counts()
```

```
# As Destination is Nominal Categorical data we will perform
OneHotEncoding
```

```
Destination = train_data[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first = True)
```

```
Destination.head()
train_data.drop(["Route", "Additional_Info"], axis = 1, inplace =
True)

train_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3
stops": 3, "4 stops": 4}, inplace = True)

# Concatenate dataframe --> train_data + Airline + Source +
Destination
data_train = pd.concat([train_data, Airline, Source, Destination],
axis = 1)
data_train.drop(["Airline", "Source", "Destination"], axis = 1,
inplace = True)

test_data = pd.read_excel(r"Test_set.xlsx")
test_data.head()

# Preprocessing
print("Test data Info")
print("-"*75)
print(test_data.info())
print()
print()
print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())

test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
```

```
test_data["Journey_month"] =
pd.to_datetime(test_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] =
pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)

test_data["Arrival_hour"] =
pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] =
pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)

duration = list(test_data["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep =
"m")[0].split()[-1]))
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)
```

```
# Categorical data
print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
print()

print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)
print()

print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first =
True)

test_data.drop(["Route", "Additional_Info"], axis = 1, inplace =
True)
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3
stops": 3, "4 stops": 4}, inplace = True)

data_test = pd.concat([test_data, Airline, Source, Destination], axis
= 1)
data_test.drop(["Airline", "Source", "Destination"], axis = 1,
               inplace = True)

print()
print()
print("Shape of test data : ", data_test.shape)

X = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Journey_month',
```

```
'Dep_hour', 'Dep_min', 'Arrival_hour', 'Arrival_min',
'Duration_hours', 'Duration_mins', 'Airline_Air India',
'Airline_GoAir', 'Airline_IndiGo',
'Airline_Jet Airways', 'Airline_Jet Airways Business',
'Airline_Multiple carriers',
'Airline_Multiple carriers Premium economy',
'Airline_SpiceJet', 'Airline_Trujet', 'Airline_Vistara',
'Airline_Vistara Premium economy', 'Source_Chennai',
'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
'Destination_Cochin', 'Destination_Delhi',
'Destination_Hyderabad', 'Destination_Kolkata',
'Destination_New Delhi']]

X.head()

y = data_train.iloc[:, 1]
y.head()

# Finds correlation between Independent and dependent attributes
plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")
plt.show()

# Important feature using ExtraTreesRegressor
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
print(selection.feature_importances_)

#plot graph of feature importances for better visualization
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
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 42)
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
y_pred = reg_rf.predict(X_test)
reg_rf.score(X_train, y_train)
reg_rf.score(X_test, y_test)
sns.distplot(y_test-y_pred)
plt.show()
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
from sklearn import metrics

# RMSE/(max(DV)-min(DV))
2090.5509/(max(y)-min(y))
metrics.r2_score(y_test, y_pred)

from sklearn.model_selection import RandomizedSearchCV

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200,
num = 12)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]

random_grid = {'n_estimators': n_estimators,
```



```
        'max_features': max_features,
        'max_depth': max_depth,
        'min_samples_split': min_samples_split,
        'min_samples_leaf': min_samples_leaf}

rf_random = RandomizedSearchCV(estimator = reg_rf,
param_distributions = random_grid,scoring='neg_mean_squared_error',
n_iter = 10, cv = 5, verbose=2, random_state=42, n_jobs = 1)
rf_random.fit(X_train,y_train)
prediction = rf_random.predict(X_test)
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,
prediction)))

import pickle
file = open('flight_rf.pkl', 'wb')
pickle.dump(rf_random, file)
model = open('flight_rf.pkl','rb')
forest = pickle.load(model)
y_prediction = forest.predict(X_test)
metrics.r2_score(y_test, y_prediction)
```

predictor.py

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
```

```
import sklearn
import pickle
import pandas as pd

app = Flask(__name__)
model = pickle.load(open("flight_rf.pkl", "rb"))
@app.route("/")
@cross_origin()
def home():
    return render_template("frontend.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)

        # 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)

        # 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)
```

```
# Duration
```

```
dur_hour = abs(Arrival_hour - Dep_hour)
```

```
dur_min = abs(Arrival_min - Dep_min)
```

```
# Total Stops
```

```
Total_stops = int(request.form["stops"])
```

```
# Airline
```

```
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
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
```



```
# Destination
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('frontend.html',prediction_text="Your
                           Flight price is ₹ {}".format(output))

return render_template("frontend.html")

if __name__ == "__main__":
    app.run(debug=True)
```

3.3 Results


Snapshot 3.1:

FLIGHT TICKET PRICE PREDICTOR

Departure Date and Time

10-01-2021 10:36 

Arrival Date and Time

10-01-2021 20:36 

Source

Bangalore 

Destination

Delhi 

Stops

Non-Stop 

Select an airline

SpiceJet 

Your Flight price is ₹ 10830.62

Snapshot 3.2:

FLIGHT TICKET PRICE PREDICTOR

Departure Date and Time

12-04-2021 21:18 

Arrival Date and Time

19-04-2021 18:22 

Source

Chennai 

Destination

Delhi 

Stops

1 

Select an airline

Jet Airways 

Your Flight price is ₹ 6488.51

Snapshot 3.3:

FLIGHT TICKET PRICE PREDICTOR

Departure Date and Time

16 - 10 - 2021 05 : 57 

Arrival Date and Time

18 - 10 - 2021 06 : 00 

Source

Mumbai 

Destination

Cochin 

Stops

Non-Stop 

Select an airline

Vistara 

Submit

Your Flight price is ₹ 5356.94

Snapshot 3.4:

FLIGHT TICKET PRICE PREDICTOR

Departure Date and Time

25 - 12 - 2021 21 : 04 

Arrival Date and Time

31 - 12 - 2021 21 : 05 

Source

Bangalore 

Destination

Hyderabad 

Stops

3 

Select an airline

IndiGo 

Submit

Your Flight price is ₹ 8277.45

CONCLUSION

The project is basically machine learning. We used Python for the implementation of the models & automation. We gathered airfare data from the web and showed that it is feasible to predict prices for flights based on historical fare data. The experimental results show that ML models are a satisfactory tool for predicting airfare prices.

Other important factors in airfare prediction are the data collection and feature selection from which we drew some useful conclusions.

From the experiments we concluded which features influence the airfare prediction at most.

REFERENCES

Bibliography:

- [1] “Automate the Boring Stuff with Python” by Al Sweigart
- [2] “Think Python: How to Think Like a Computer Scientist” by Allen B. Downey
- [3] “Python Data Science Handbook: Essential Tools for Working with Data” by Jake VanderPlas

Websites:

- [1] <https://github.com>
- [2] <https://kaggle.com>
- [3] <https://realpython.com>