OPTIMIZING FLIGHT BOOKING DECISIONS

THROUGH MACHINE LEARNING

PRICE PREDICTION

TEAM SIZE : 4

TEAM LEADER : 1.AJAY KUMAR C(20UCS2497)

TEAM MEMBERS : 1.ABDUL AASHICK(20UCS2496)

2.ARUNACHALAM(20UCS2498)

3.DHANUSH(20UCS2499)

INTRODUCTION

OVERVIEW

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. 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.

PURPOSE

The purpose of flight price prediction through machine learning is to provide customers with an estimate of the future price of a flight ticket. This can help travelers make informed decisions about when to book their flights, based on the predicted price trends.

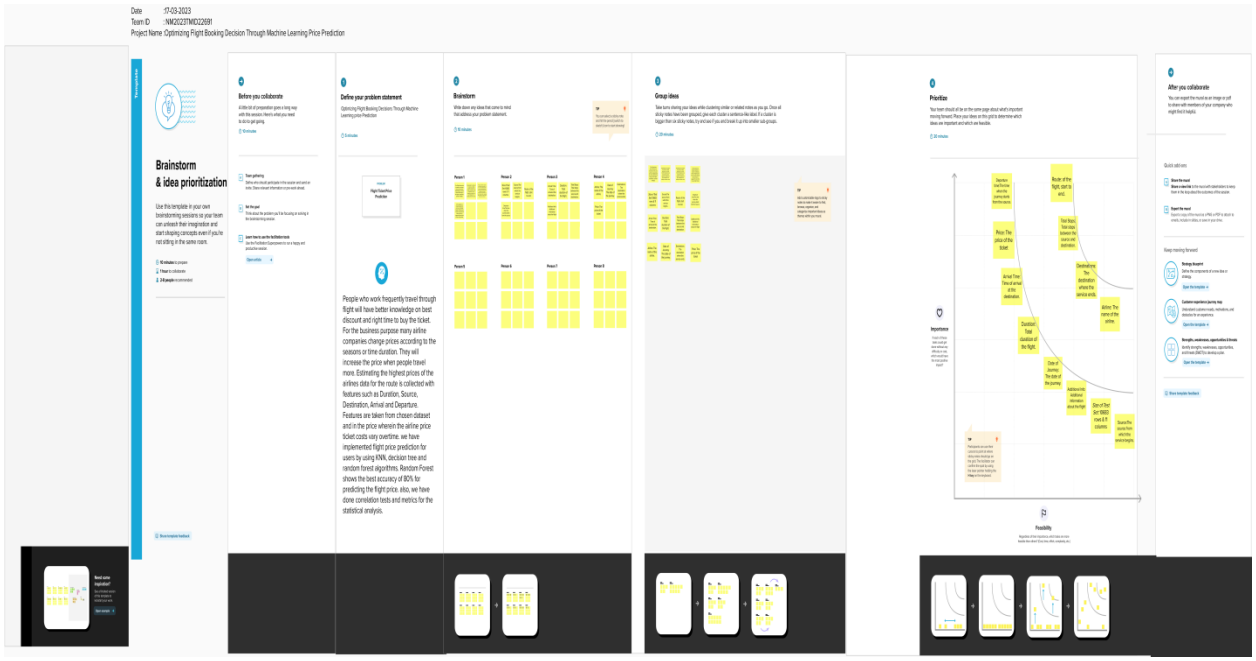
By using historical data on flight prices, weather patterns, and other relevant factors, machine learning models can analyze patterns and make predictions about future prices. These predictions can be used by airlines and travel companies to adjust their pricing strategies, and by customers to find the best deals on flights.

Flight price prediction through machine learning can help to increase efficiency and reduce costs for airlines and travel companies, as well as improve the customer experience by providing more accurate pricing information. It can also help to reduce the amount of time and effort required for customers to search for the best flight deals, as they can rely on the predictions made by the machine learning models.

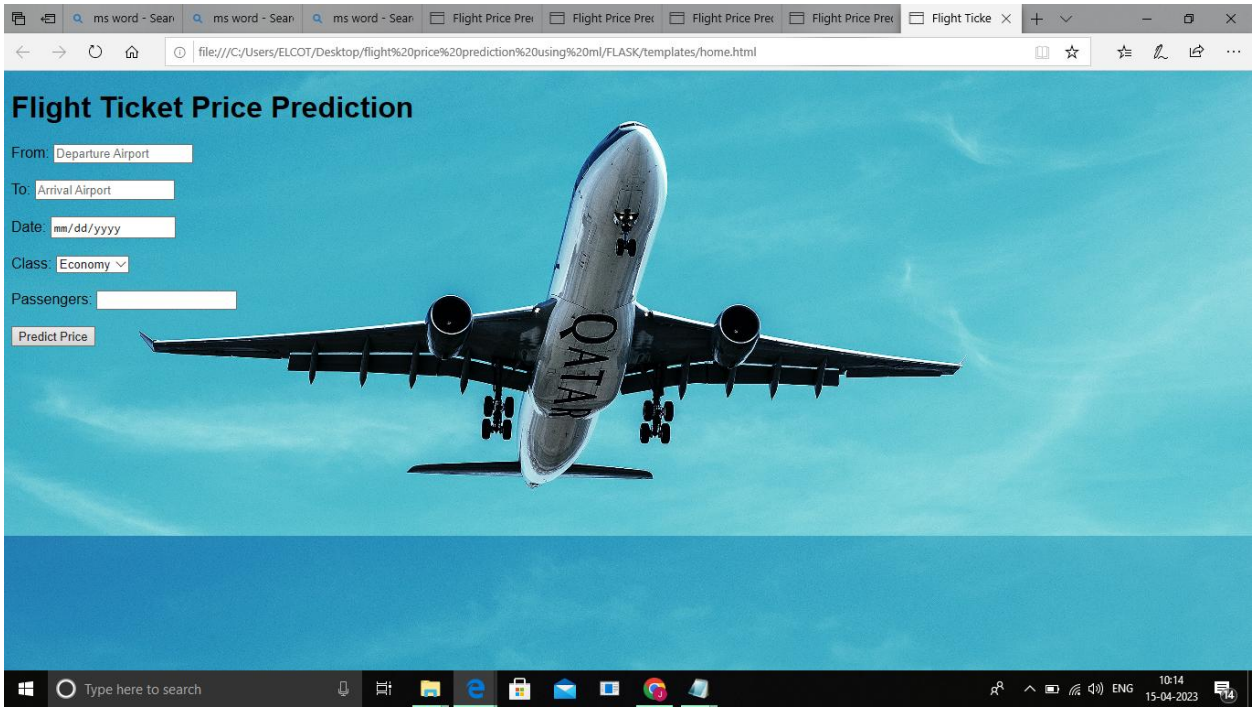
PROBLEM DEFINITION & DESIGN THINKING

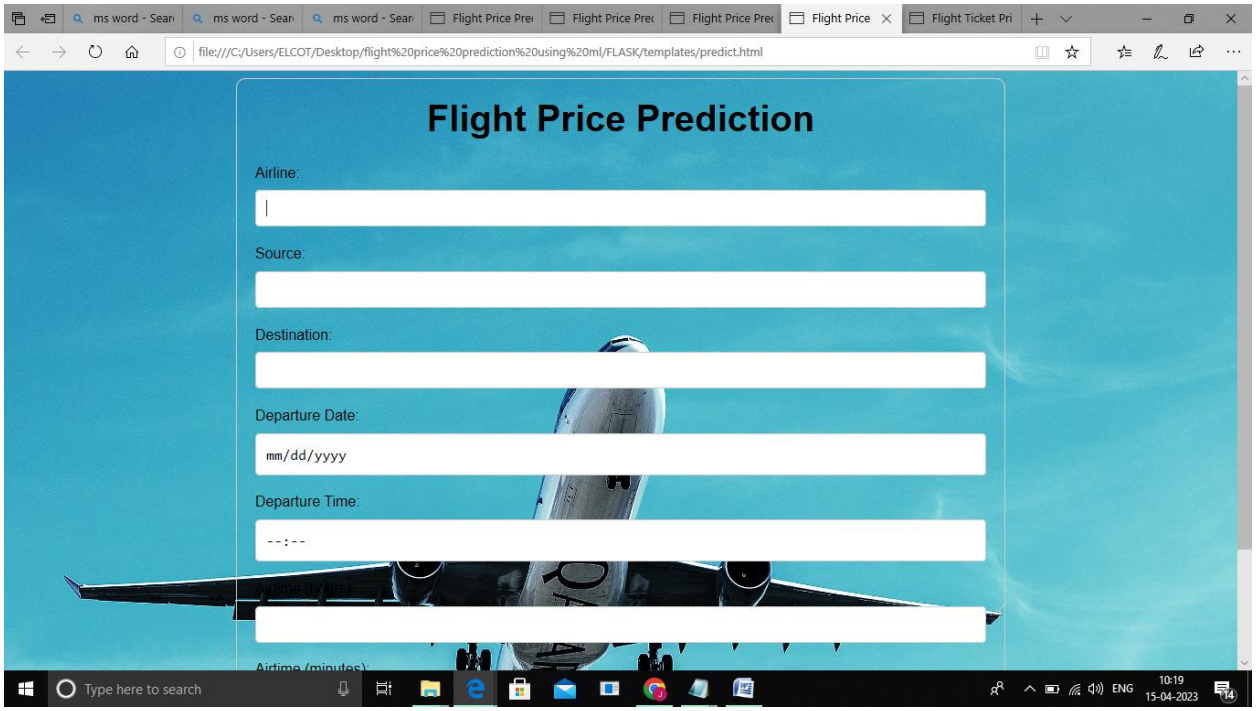
EMPATHY MAP

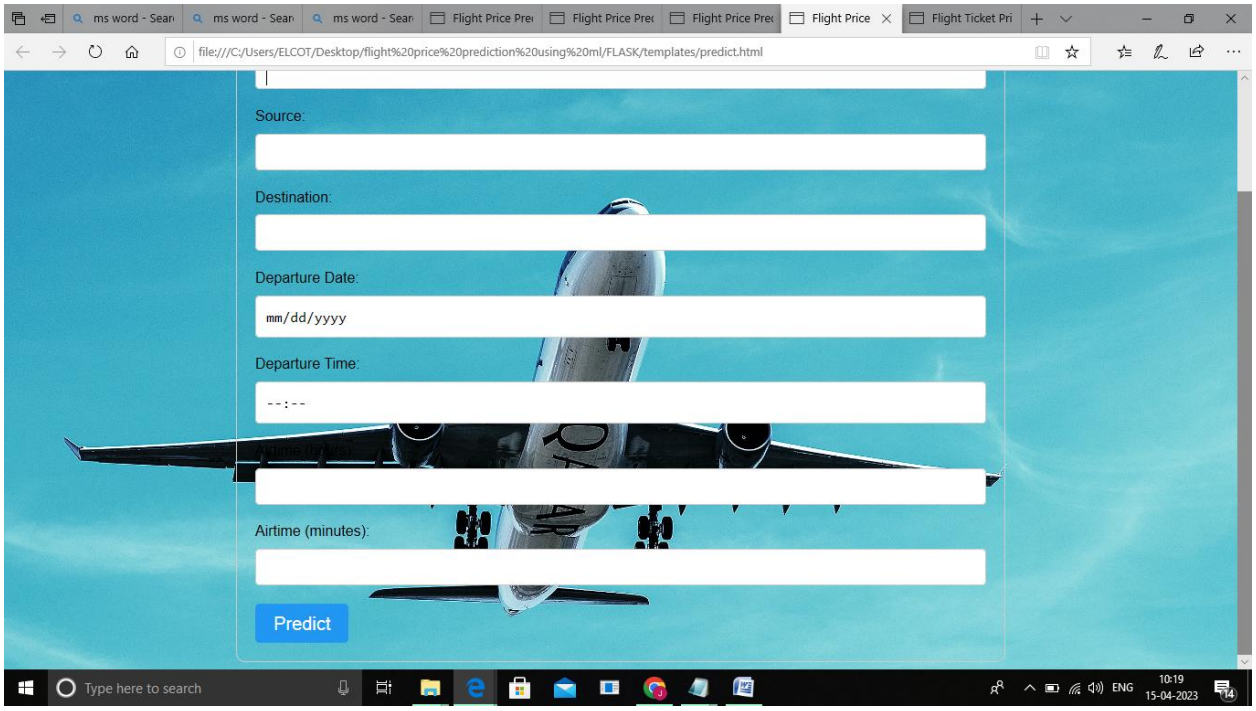
IDEATION & BRAISTROMING MAP


Date 17-03-2023
Team ID NM2022TMID22691
Project Name Optimizing Flight Booking Decision Through Machine Learning Price Prediction
Brainstorm
& idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
After you collaborate
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. we have implemented flight price prediction for users by using RNN, 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.

RESULT


Flight Ticket Price Prediction
From: Departure Airport
To: Arrival Airport
Date: mm/dd/yyyy
Class: Economy
Passengers:
Predict Price
file:///C:/Users/ELCOT/Desktop/flight%20price%20prediction%20using%20ml/FLASK/templates/home.html
Type here to search
ENG
10:14
15-04-2023


Flight Price Prediction
Airline:
Source:
Destination:
Departure Date:
mm/dd/yyyy
Departure Time:
--:--
Airtime (minutes):
Type here to search
ENG
10:19
15-04-2023


Source:
Destination:
Departure Date:
mm/dd/yyyy
Departure Time:
--:--
Airtime (minutes):
Predict
Type here to search
ENG
10:19
15-04-2023

ADVANTAGES

1. **Cost savings: Flight price prediction can help travelers save money by identifying the best time to buy tickets. Predicting flight prices ahead of time helps travelers to make more informed decisions on when to buy a ticket and how much they should expect to pay. This can lead to significant savings on travel expenses.**
2. **Improved travel planning: Flight price prediction can help travelers to plan their trips more effectively. By knowing when and where to book flights, travelers can better plan their itineraries and make more efficient use of their time.**
3. **Reduced stress: By knowing the expected cost of their travel, travelers can avoid the stress and anxiety associated with last-minute bookings and price fluctuations. They can make more informed decisions about their travel plans and avoid the uncertainty that comes with not knowing how much they will have to pay for flights.**
4. **Increased competition among airlines: By providing travelers with more transparency on flight prices, flight price prediction can increase competition among airlines. This can lead to lower prices and better deals for travelers.**

DISADVANTAGES

1. **Inaccuracy: While machine learning algorithms can be trained on large amounts of data to make accurate predictions, there is always a margin of error. Flight prices**

can be affected by many variables, such as changes in demand, weather conditions, and fuel prices, which can be difficult to predict with complete accuracy.

2. **Limited data availability: Machine learning algorithms require a large amount of data to make accurate predictions. However, data on flight prices and demand may not always be available or accessible, particularly for smaller airlines or less popular routes.**

3. **Over-reliance on technology: While technology can be a valuable tool for predicting flight prices, it should not be the only factor considered when making travel decisions. Over-reliance on technology can lead to missed opportunities, such as last-minute deals or alternative travel options.**

4. **Ethical concerns: There are also ethical concerns related to the use of personal data for flight price prediction. Some travelers may be uncomfortable with the amount of personal data that is collected and used by airlines or third-party booking sites.**

APPLICATIONS

- **Airline pricing strategy: Airlines can use flight price prediction to optimize their pricing strategy and set competitive fares that reflect the demand for flights. By understanding customer behavior and market trends,**

airlines can make data-driven decisions about pricing and improve their revenue management.

- **Travel planning and booking: Travelers can use flight price prediction to plan their trips and book flights at the most favorable prices. By knowing when and where to book flights, travelers can save money and avoid last-minute price hikes.**

- **Business travel management: Companies that manage business travel can use flight price prediction to reduce travel costs and optimize their travel budgets. By using predictive analytics, they can identify the most cost-effective travel options and negotiate better deals with airlines and travel agencies.**

- **Market analysis and trend monitoring: Travel agencies and other industry stakeholders can use flight price prediction to monitor market trends and identify patterns in customer behavior. By analyzing the data on flight prices and demand, they can make informed decisions about marketing, inventory management, and customer service.**

CONCLUSION

Predicting flight prices can be a complex task that involves various factors such as airline, route, season, and demand. Machine learning models can be trained on historical data to make

predictions on future prices.There are various techniques that can be used to build predictive models such as linear regression, decision trees, and neural networks. Each technique has its strengths and weaknesses, and the choice of method depends on the specific problem at hand.To build an accurate model, it is important to collect and preprocess relevant data, perform feature engineering to extract meaningful features, and tune the model's hyperparameters to optimize performance.In conclusion, predicting flight prices using machine learning is a challenging task but can be achieved with careful data preparation, model selection, and parameter tuning. The accuracy of the model can be improved over time as more data becomes available and new techniques are developed.

FUTURE SCOPE

Real-time pricing: Real-time pricing is an emerging area where airlines dynamically adjust their fares based on current demand and availability. Machine learning models can be used to predict future demand and help airlines optimize their pricing strategies in real-time.

Customer segmentation: Machine learning can be used to segment customers based on their travel patterns, preferences, and buying behavior. Airlines can then offer personalized pricing and promotions to different segments, leading to increased customer loyalty and revenue.

Multi-city itineraries: Many travelers book multi-city itineraries, which can be challenging to price accurately. Machine

learning models can be used to predict the optimal pricing for multi-city itineraries, taking into account the availability of flights and the overall demand.

APPENDIX

SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestclassifier,GradientBoostin
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report,confusion_matrix
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')

data=pd.read_csv("Data Train.csv")
data.head()

for i in categorical:
    print(i, data[i].unique())

data.Date_of_Journey=data.Date_of_Journey.str.split("/")
data.Date_of_Journey

data['Date']=data.Date_of_Journey.str[0]
data['Month']=data.Date_of_Journey.str[1]
data['Year']=data.Date_of_Journey.str[2]

data.Total_Stops.unique()
#array(["non-stop"," 2 stops", "1 stop", "3 stops", "4 stops"],dtype=object)
```

```
data.Route=data.Route.str.split("->")
data.Route

data['City1']=data.Route.str[0]
data['City2']=data.Route.str[1]
data['City3']=data.Route.str[2]
data['City4']=data.Route.str[3]

data.Dep_Time=data.Dep_Time.str.split(":")
data['Dep_Time_Hour']=data.Dep_Time.str[0]
data['Dep_Time_Mins']=data.Dep_Time.str[1]

data.Arrival_Time=data.Arrival_Time.str.split(' ')

data['Arrival_date']=data.Arrival_Time.str[1]
data['Time_of_Arrival']=data.Arrival_Time.str[0]

data["Time of Arrival"]-data.Time_of_Arrival.str.split(":")

data['Arrival_Time_Hour' ]=data.Time_of_Arrival.str[0]
data['Arrival_Time_Mins' ]=data.Time_of_Arrival.str[1]

data.Duration=data.Duration.str.split(' ')

data['Travel_Hours']=data.Duration.str[0]
data['Travel_Hours']=data['Travel_Hours'].str.split('h')
data['Travel_Hours' ]=data['Travel_Hours'].str[0]
data.Travel_Hours=data.Travel_Hours
data['Travel_Mins']=data.Duration.str[1]

data.Travel_Mins=data.Travel_Mins.str.split('m')
data.Travel_Mins=data.Travel_Mins.str[0]


data.Total_Stops.replace('non_stop',0,inplace=True)
data.Total_Stops=data.Total_Stops.str.split('')
data.Total_Stops=data.Total_stops.str[0]

data.Additional_Info.unique()

data.drop(['city4','city5','city6'],axis=1,inplace=True)

data.drop(['Date_of_Journey','Route','Dep_Time','Arrival_Time','Duration'],axis=1
,inplace=True)
data.drop(['Time_of_Arrival'],axis=1,inplace=True)
```

```
data.isnull().sum()

data['City3'].fillna('None',inplace=True)
data['Arrival_date'].fillna(data['Date'],inplace=True)
data['Travel_Mins'].fillna(0,inplace=True)

data.info()


data.Date=data.Date.astype('int64')
data.Month=data.Month.astype('int64')
data.Year=data.Year.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Mins=data.Dep_Time_Mins.astype('int64')
data.Arrival_date=data.Arrival_date.astype('int64')
data.Arrival_Time_Hour=data.Arrival_Time_Hour.astype('int64')
data.Arrival_Time_Mins=data.Arrival_Time_Mins.astype('int64')
data.Travel_Mins=data.Travel_Mins.astype('int64')

data[data['Travel_Hours']=='5m']
data.drop(index=6474,inplace=True,axis=0)
data.Travel_Hours=data.Travel_Hours.astype('int64')

categorical=['Airline','Source','Destination','Additional_Info','City1']
numerical=['Total
Stops','Date','Month','Year','Dep_Time_Hour','Dep_Time_Mins','Arrival_date','Arri
val_Time_Hour','Arrival_Time_Mins','Travel_Hours','Travel_Mins']


from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

data.Airline=le.fit_transform(data.Airline)
data.Source=le.fit_transform(data.Source)
data.Destination=le.fit_transform(data.Destination)
data.Total_Stops=le.fit_transform(data.Total_Stops)
data.Cityl=le.fit_transform(data.city1)
data.City2=le.fit_transform(data.city2)
data.City3-le.fit_transform(data.City3)
data.Additional_Info=le.fit_transform(data.Additional_Info)
data.head()

data.head()
```

```
data=data[['Airline','Source','Destination','Date','Month','year','dep_Time_Mins'
,'Arrival_date','Arrival_Time_Hour','Arrival_Time_Mins']]

data.head()

data.describe()

import seaborn as sns
C=1
plt.figure(figsize=(20,45))

for i in categorical:
    plt.subplot(6,3,c)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.tight_layout(pad=3.0)
    C=C+1

plt.show()

plt.figure(figsize=(15,8))
sns.distplot(data.Price)

#<AxesSubplot:xlabel='Price',ylabel='Destiny'>

sns.heatmap(data.corr(),annot=True)
#<AxesSubplot:>

import seaborn as sns
sns.boxplot(data['Price'])

#<AxesSubplot:xlabel='Price'>

y = data['Price']
x= data.drop(columns=["Price"],axis=1)

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()

x_scaled = ss.fit_transform(x)

x_scaled = pd.DataFrame(x_scaled,columns=x.columns)
x_scaled.head()

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)

x_train.head()

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
AdaBoostRegressor
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

for i in [rfr, gb,ad]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.2:
        print(i)

print("R2 score is",r2_score(y_test,y_pred))
print("R2 for train data",r2_score(y_train, i.predict(x_train)))
print("Mean Absolute Error is",mean_absolute_error(y_pred,y_test))
print("Mean Squared Error is",mean_squared_error(y_pred,y_test))
print("Root Mean Sqaured Error is",
(mean_squared_error(y_pred,y_test,squared=False)))

from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

knn=KNeighborsRegressor()
svr=SVR()
dt=DecisionTreeRegressor()


for i in [knn,svr,dt]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.1:
        print(i)
```

```
print('R2 Score is',r2_score(y_test,y_pred))
print('R2 Score for train data',r2_score(y_train,i.predict(x_train)))
print('Mean Absolute Error is',mean_absolute_error(y_test,y_pred))
print('Mean Squared Error is', mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error
is',(mean_squared_error(y_test,y_pred,squared=False)))

from sklearn.model_selection import cross_val_score
for i in range(2,5):
    cv=cross_val_score(rfr,x,y,cv=i)
    print(rfr,cv.mean())

from sklearn.model_selection import RandomizedSearchCV

param_grid={'n_estimators':[10,30,50,70,100],'max_depth':[None,1,2,3],
'max features':['auto','sqrt']}

rfr=RandomForestRegressor()
rf_res=RandomizedSearchCV(estimator=rfr,param_distributions=param_grid,cv=3,verbo
se=2,n_jobs=-1)

rf_res.fit(x_train,y_train)

gb=GradientBoostingRegressor()
gb_res=RandomizedSearchCV(estimator=gb,param_distributions=param_grid,cv=3,verbos
e=2,n_jobs=-1)

gb_res.fit(x_train,y train)


rfr=RandomForestRegressor(n_estimators=10,max_features="sqrt",max_depth=None)
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))

knn=KNeighborsRegressor(n_neighbors=2,algorithm='auto',metric_params=None,n_jobs=
-1)
knn.fit(x_train,y_train)
y_train_pred=knn.predict(x_train)
y_test_pred=knn.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))

price_list=pd.DataFrame({'Price':prices})

price_list

import pickle
pickle.dump(rfr,open('model1.pkl', 'wb'))

import pickle
pickle.dump(rfr,open('model1.pkl', 'wb'))
```