Optimizing Flight Booking Decisions through Machine Learning Price Predictions

**ABSTRACT:**

A lot of factors that affect the overall price of airline tickets, including the airline, the date of travel, source, destination, route, duration, and so on. Each provider seems to have its own unique set regulations and methods for determining pricing. Recent breakthroughs in Artificial Intelligence (AI) and Machine Learning (ML) allow for the inference of such principles as well as the modelling of price volatility. This article is a study conducted on predicting flight prices. Utilizing two datasets for testing and training, this study analyses various machine learning methods for predicting flight prices.

**INTRODUCTION:**

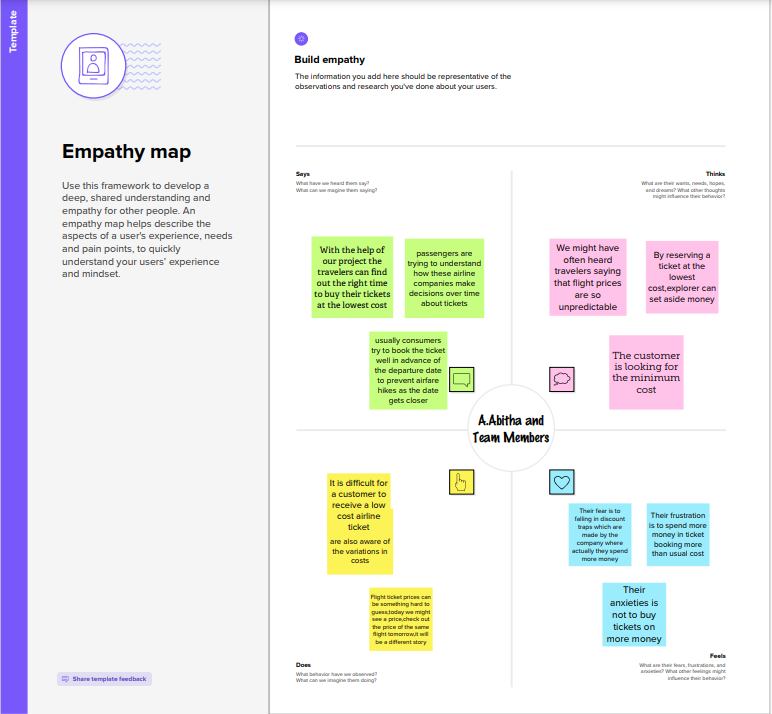
Perfect time for purchasing plane ticket by the passenger’s view is difficult since passengers get very less information of future business price rates. Different models figure out future business price on plane and categorise the best time to obtain flight ticket. Airlines use different strategies of pricing for their tickets, later taking the decision on price because order shows higher value for the approximation models. The causes behind the difficult system is each Planes has limited number of seats to be filed, so airlines must regulate demand. Suppose when demand is expected to increase capacity, the airline may increase prices, to decrease the rate at which seats fill. Also, seating arrangements in flight which is not occupied shows the loss of the amount invested for the business airline companies and making them purchase the ticket to fill the seats for any price this would be the best idea to get profit in loss too. Passengers should be compatible with the airline companies to get adjusted for the increase and decrease of the price. Passengers or customers should make their own planning to get the best offers available on different airlines and travel through less price. Planes ticket prices changes as time passes, pulling out the elements which creates the difference. Reporting the correlated and models which is used to price the flight tickets. Then, using thatinformation, building the model which helps passengers to make pull out the ticket to buy and predicting air ticket prices which progresses in the future. Duration, Arrival time, Price, Source, Destination and much more these are the attribute used for flight price prediction.

**Purpose Statement:**

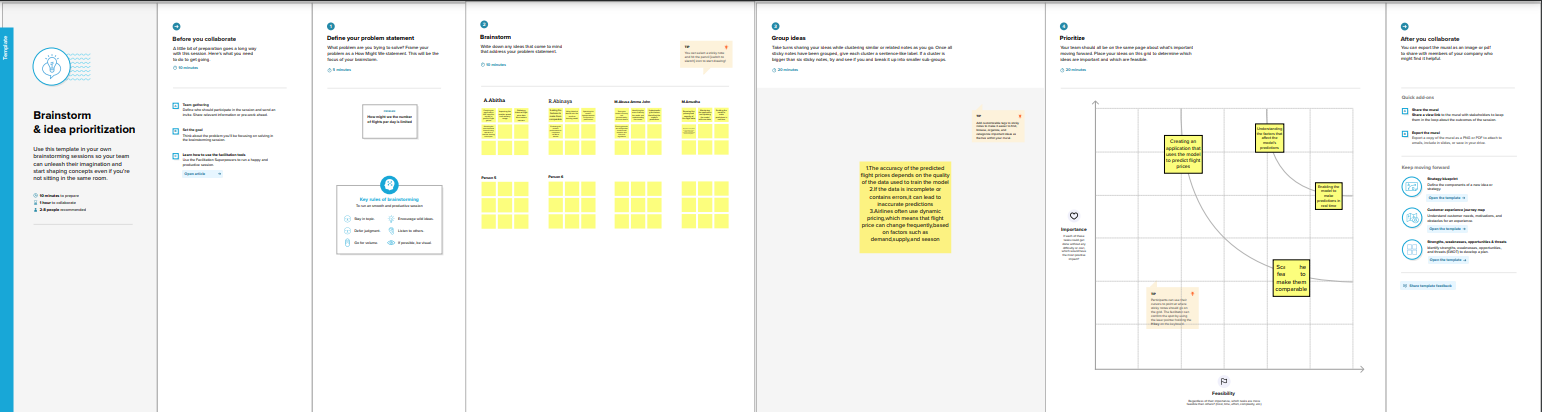
* A Flight price prediction application which predicts fares of flight for a particular date based on various parameters like Source, Destination, Stops & Airline.
* With consideration of some features like arrival time, departure time as well as time to purchase the ticket using these factors prices can be predict.
* Due to this factors there may be change in airline fare prices and also detect how factors are related to being change of Flight ticket.
* In short, distance and demand are pivotal factors that determine flight ticket prices.
* Travel is just like any other valuable commodity, and it's priced so that when demand is low, prices are low, to stimulate sales. Likewise, when demand is high, prices are high to capitalise on the interest.

**Problem Defining & Design Thinking:**

Empathy Map:



**Ideation & Brainstorming Map:**



**RESULTS:**

* Optimizing flight booking decisions through machine learning price predictions can result in several benefits for both customers and airlines. By accurately predicting flight prices, customers can make informed decisions about when to book their flights and potentially save money on their travel expenses. For airlines, this can lead to increased sales and revenue, as well as more efficient pricing strategies.
* Machine learning algorithms can large amounts of historical data on flight prices, as well as current market trends and external factors such as weather and holidays, to make accurate predictions on future prices. By leveraging this technology, airlines can adjust their pricing in real-time to match demand and optimize their revenue.
* In addition, customers can benefit from personalized recommendations based on their travel preferences and past booking history.
* Machine learning algorithms can take into account factors such as preferred airlines, travel dates, and destinations to provide tailored recommendations for each customer.
* Overall, optimizing flight booking decisions through machine learning price predictions can lead to more efficient and personalized travel experiences for both customers and airlines.

**Advantages Of Flight Price Prediction:**

* Cost savings: One of the biggest advantages of using machine learning for flight booking decisions is cost savings. By predicting the price of a flight, a traveler can choose the most cost-effective option. This can lead to significant savings, especially for frequent travelers.
* Improved decision-making: Machine learning algorithms can analyze vast amounts of data to identify patterns and trends that humans may not be able to see. This means that travelers can make more informed decisions about when to book flights and which airlines to choose.
* Time savings: Machine learning algorithms can quickly analyze large amounts of data and provide insights that would take humans much longer to find. This means that travelers can make decisions faster and spend less time researching flight options.
* Personalization: Machine learning algorithms can take into account a traveler's past booking history, preferences, and other factors to provide personalized flight recommendations. This can help travelers find flights that best meet their needs and preferences.
* Competitive advantage: Businesses that use machine learning for flight booking decisions can gain a competitive advantage by offering more personalized and cost-effective options to their customers. This can help businesses attract and retain customers and increase revenue

**Disadvantages Of Flight Price Prediction:**

* Inaccuracy: Machine learning algorithms are only as good as the data they are trained on. If the algorithm is not trained on a large enough and diverse enough dataset, it may not provide accurate predictions. Additionally, unforeseen events such as weather disruptions, political unrest or other unexpected factors may affect the accuracy of the algorithm.
* Lack of transparency: Machine learning algorithms can be difficult to interpret, which can make it hard to understand how the algorithm arrived at a particular prediction. This can be particularly concerning if the algorithm is used to make important decisions, such as which flight to book.
* Limited control: Travelers may feel like they have less control over their flight booking decisions if they rely solely on machine learning algorithms. Some travelers may prefer to have more control over their flight booking decisions, such as choosing specific airlines or routes.
* Over-reliance: Travelers may become too reliant on machine learning algorithms and fail to consider other important factors, such as flight times, layovers, and personal preferences. This can result in poor travel experiences and even missed flights

**Applications for Flight Price Prediction:**

* Machine learning can be used to optimize flight booking decisions by predicting prices of airline tickets. By analyzing historical data on flight prices, weather conditions, fuel costs, demand, and other relevant factors, machine learning algorithms can predict the likelihood of price changes for specific routes and dates. This information can help travelers make informed decisions about when to book flights to get the best possible price.
* Here are some specific applications of machine learning for optimizing flight booking decisions:
* Price prediction: Machine learning algorithms can analyze historical flight data to predict future price trends for specific routes and dates. This information can be used by travelers to make informed decisions about when to book flights to get the best possible price.
* Demand prediction: By analyzing historical flight data and other factors such as holidays, events, and weather conditions, machine learning algorithms can predict future demand for specific routes and dates. This information can be used by airlines to adjust prices and seat availability accordingly.
* Route optimization: Machine learning algorithms can analyze data on flight routes, travel times, and other factors to optimize airline schedules and routes. This can help airlines reduce costs, improve efficiency, and offer better prices to customers.
* Personalized pricing: Machine learning algorithms can analyze customer data such as past purchases, search history, and preferences to offer personalized pricing and recommendations. This can help airlines increase customer loyalty and improve their bottom line.
* Overall, machine learning can help airlines and travelers alike optimize flight booking decisions and get the best possible prices and experiences.

**Conclusion:**

Machine learning models were examined in this case study 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 selectionstrategies .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 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.

**Future Scope:**

In Upcoming days when huge amount of information is accessed as in detailed information in the dataset, the expected results in future are highly correct. For further research anyone desire to expand upon it ought to request different sources of historical data or be a lot of organized in collection knowledge manually over amount of your time to boot, a lot of different combination of plane are going to be traversed. There is whole possibility that planes differ their execution ideas consisting characteristics of the plane. At last, it is curious to match our model accuracy with that of the business models accuracy offered nowadays. Machine Learning algorithms are applied on the dataset to predict the dynamic fare of flights. This gives the predicted values of flight fare to get a flight ticket at minimum cost. Data is collected from the websites which sell the flight tickets so only limited information can be accessed. 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.

**APPENDIX:**

SOURCE CODE:

Milestone2:

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, GradientBoostingClassifier

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

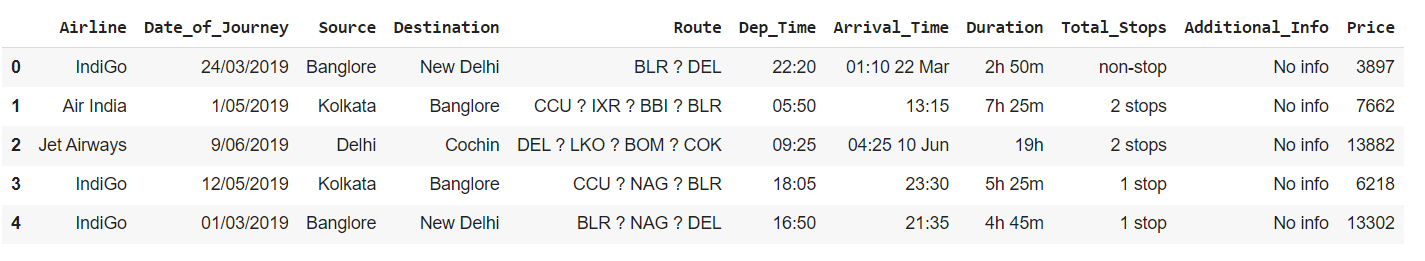
warnings.filterwarnings('ignore')

plt.style.use('fivethirtyeight')

**Read the data set**

data=pd.read\_csv("/content/Data\_Train.csv")

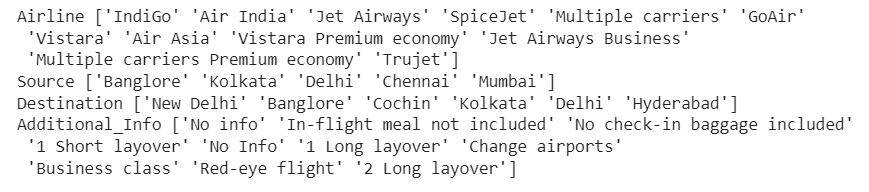
data.head()



**Data Preparation**

for i in category:

    print(i, data[i].unique())



category\_cols=data.select\_dtypes(include=['object']).columns

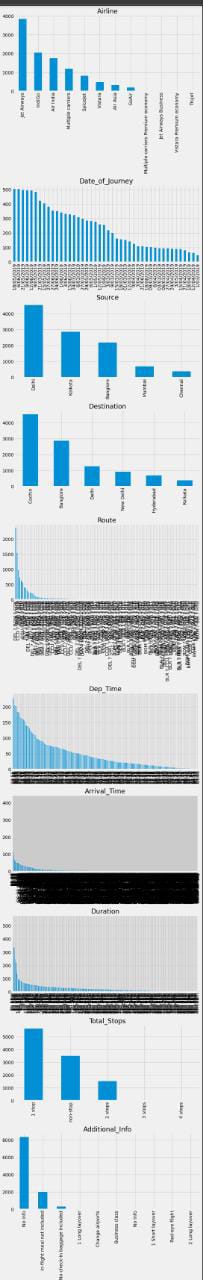
category\_cols

for column in category\_cols:

    plt.figure(figsize=(20,4))

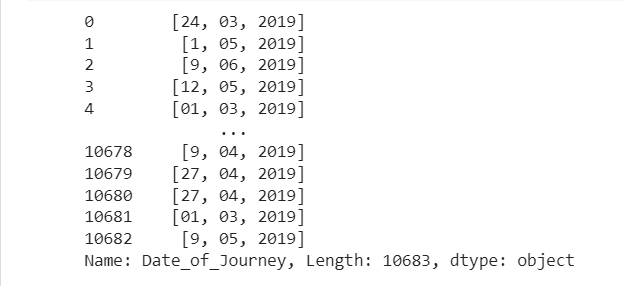
    plt.subplot(121)

    data[column].value\_counts().plot(kind='bar')



data.Date\_of\_Journey=data.Date\_of\_Journey.str.split('/')

data.Date\_of\_Journey



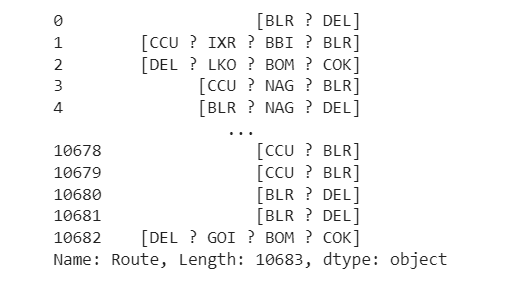
data['Date']=data.Date\_of\_Journey.str[0]

data['Month']=data.Date\_of\_Journey.str[1]

data['Year']=data.Date\_of\_Journey.str[2]

data.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['City5']=data.Route.str[4]

data['City6']=data.Route.str[5]

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')

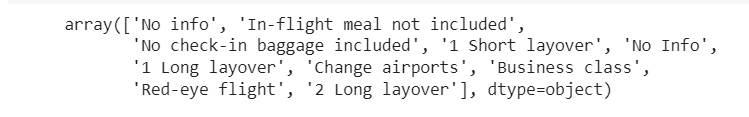
data.Travel\_Mins=data.Travel\_Mins.str[0]

data.Total\_Stops.replace('non\_stop',0,inplace=True)

data.Total\_Stops=data.Total\_Stops.str.split(' ')

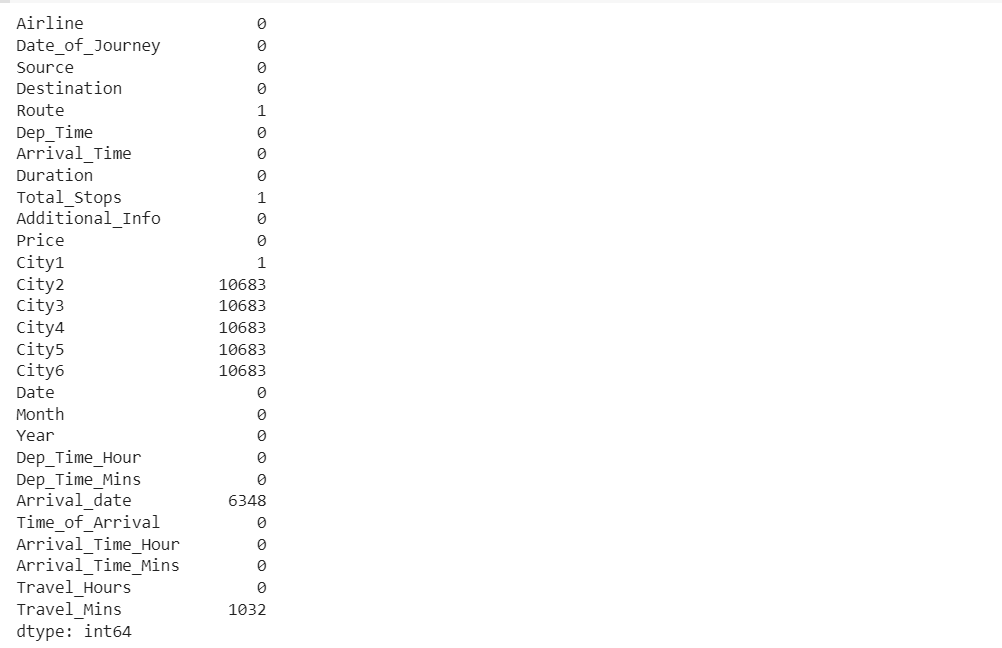
data.Total\_Stops=data.Total\_Stops.str[0]

data.Additional\_Info.unique()



data.Additional\_Info.replace('No Info','No info',inplace=True)

data.isnull().sum()

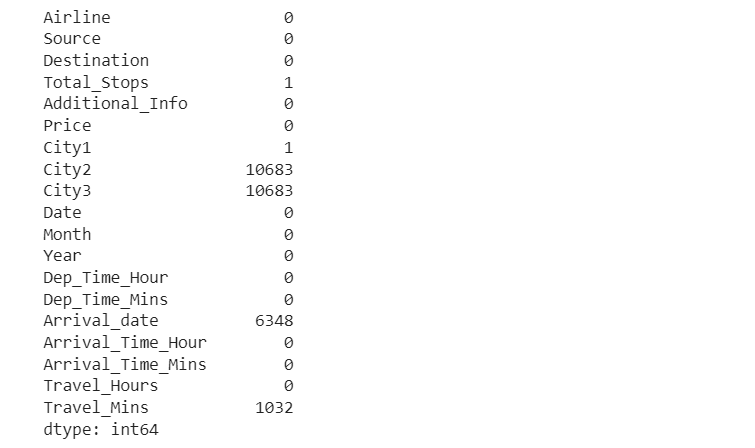


data.drop(['City4','City5','City6'],axis=1,inplace=True)

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()

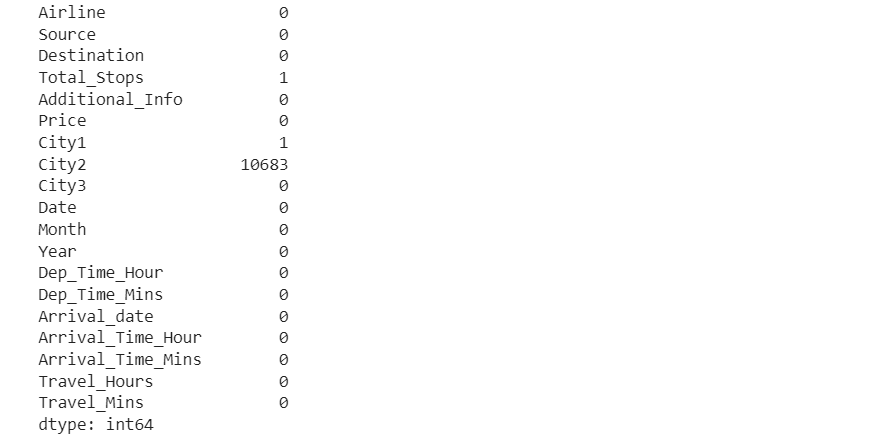


data['City3'].fillna('None',inplace=True)

data['Arrival\_date'].fillna(data['Date'],inplace=True)

data['Travel\_Mins'].fillna(0,inplace=True)

data.isnull().sum()



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')

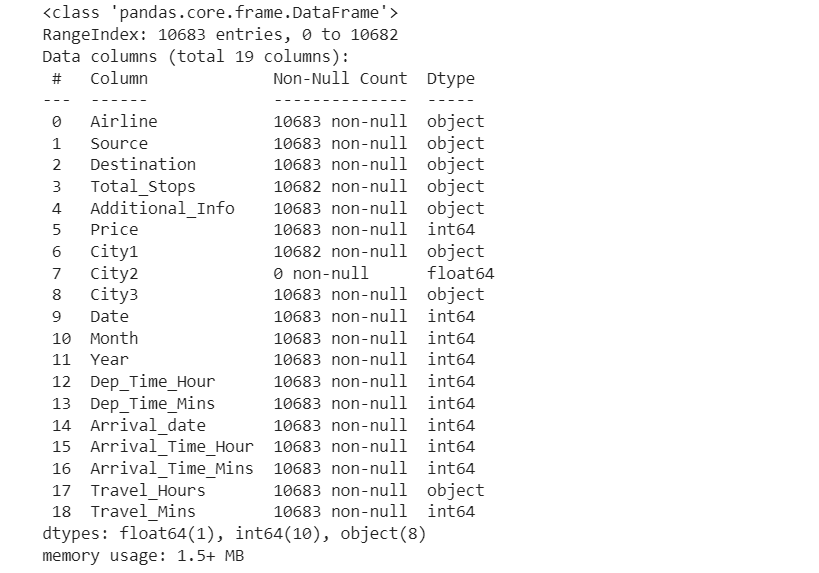
data.Arrival\_date=data.Arrival\_date.astype('int64')

data.Arrival\_Time\_Hour=data.Arrival\_Time\_Hour.astype('int64')

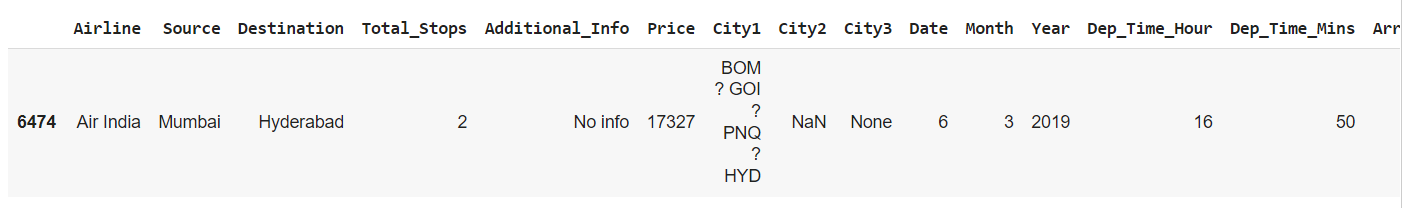
data.Arrival\_Time\_Mins=data.Arrival\_Time\_Mins.astype('int64')

data.Travel\_Mins=data.Travel\_Mins.astype('int64')

data.info()



data[data['Travel\_Hours']=='5m']

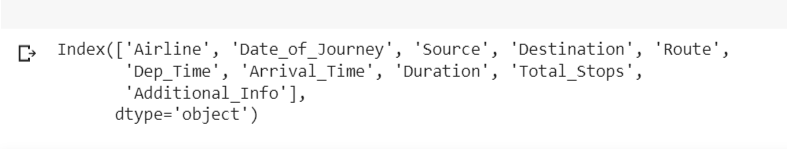


data.drop(index=6474,inplace=True,axis=0)

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\_Time\_Mins','Travel\_Hours','Travel\_Mins']



**Milestone3**

import seaborn as sns

c=1

plt.figure(figsize=(20,45))

for i in categorical:

    plt.subplot(6,3,c)

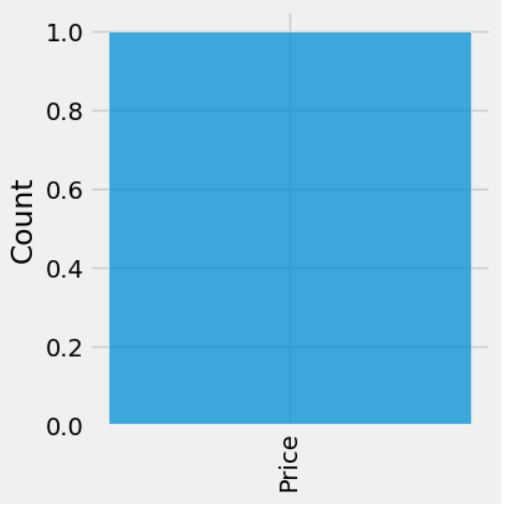
    sns.displot("Price")

    plt.xticks(rotation=90)

    plt.tight\_layout(pad=3.0)

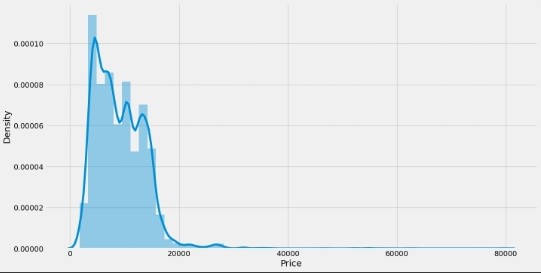
    c=c+1

plt.show()

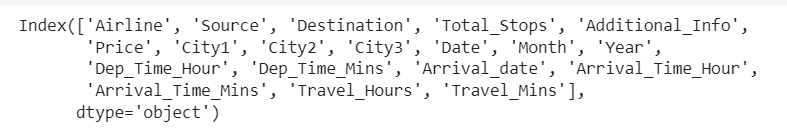


plt.figure(figsize=(15,8))

sns.distplot(data.Price)



data.columns



import seaborn as sns

c=1

for i in categorical:

  plt.figure(figsize = (10,20))

  plt.subplot(6,3,c)

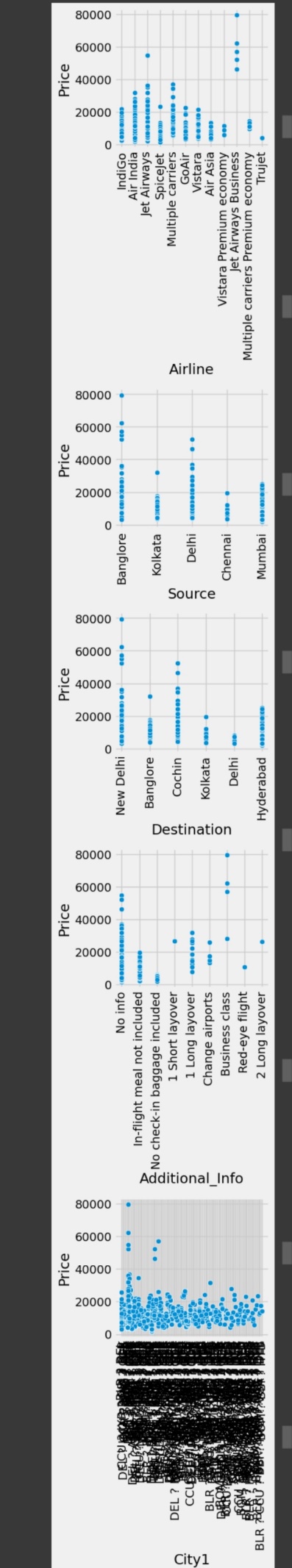
  sns.scatterplot(x=data[i],y=data.Price)

  plt.xticks(rotation=90)

  #plt.tight\_layout(pad=3.0)

  c=c+1

  plt.show()

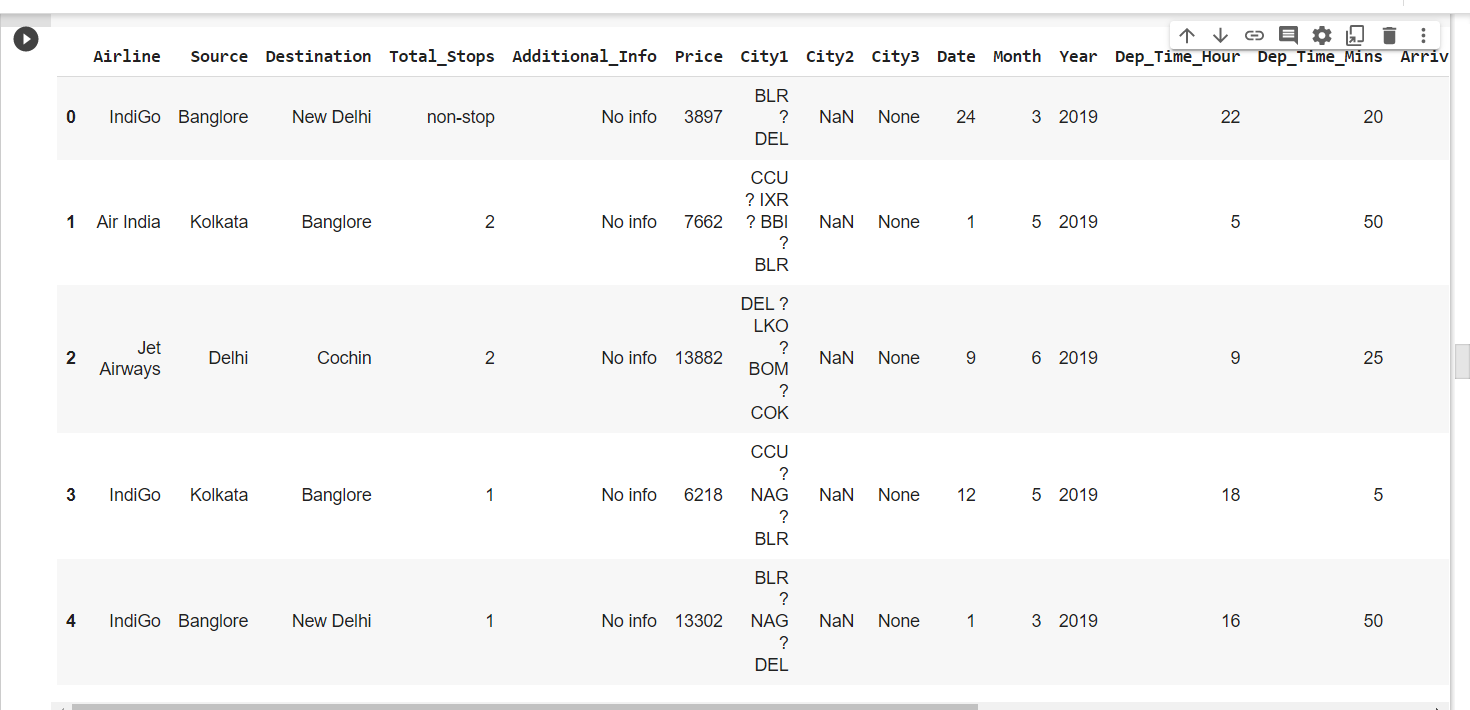


data[data.Price>50000]

data.head()

pd.set\_option('display.max\_columns',25)

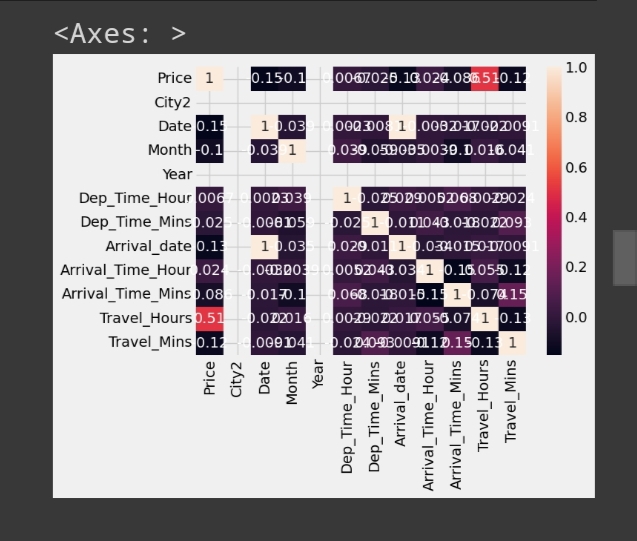
data.head()



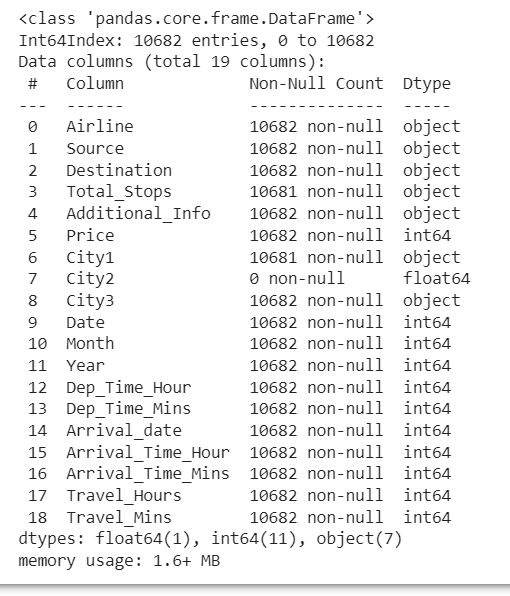
data['Year'].max()

2019

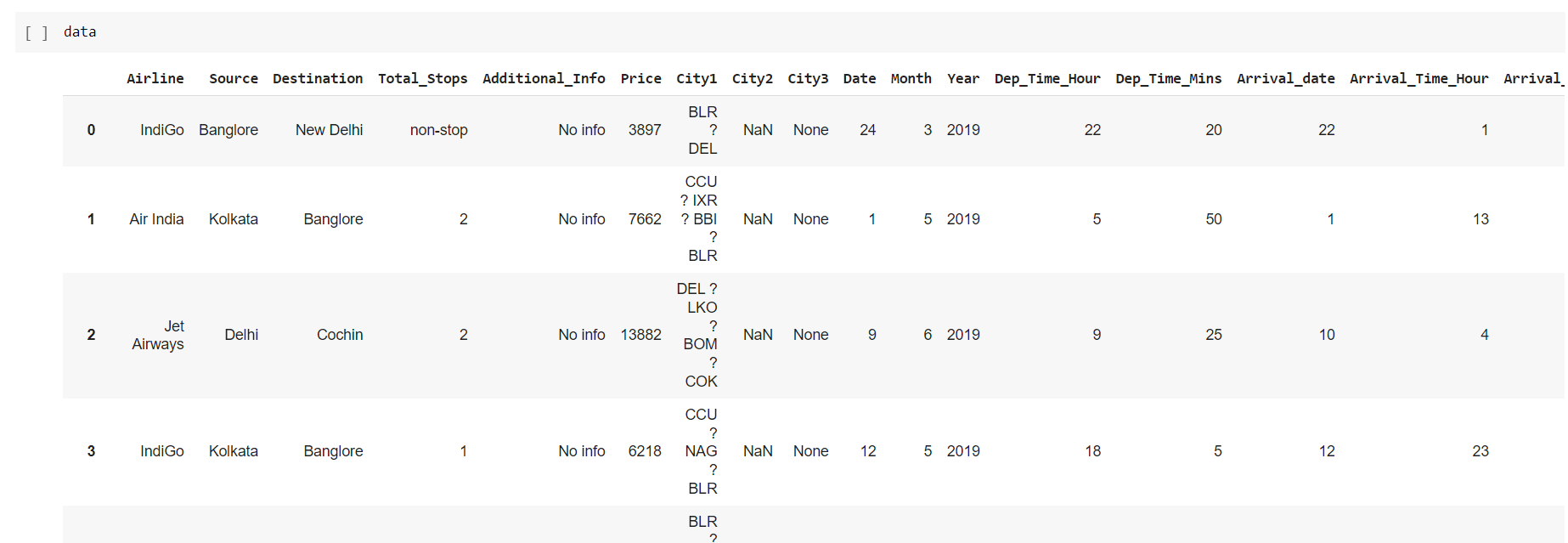
sns.heatmap(data.corr(),annot=True)



data.info()



Data



c=1

for i in numerical:

  plt.figure(figsize=(10,20))

  plt.subplot(6,3,c)

  sns.scatterplot(x = data[i], y=data.Price)

  plt.xticks(rotation=90)

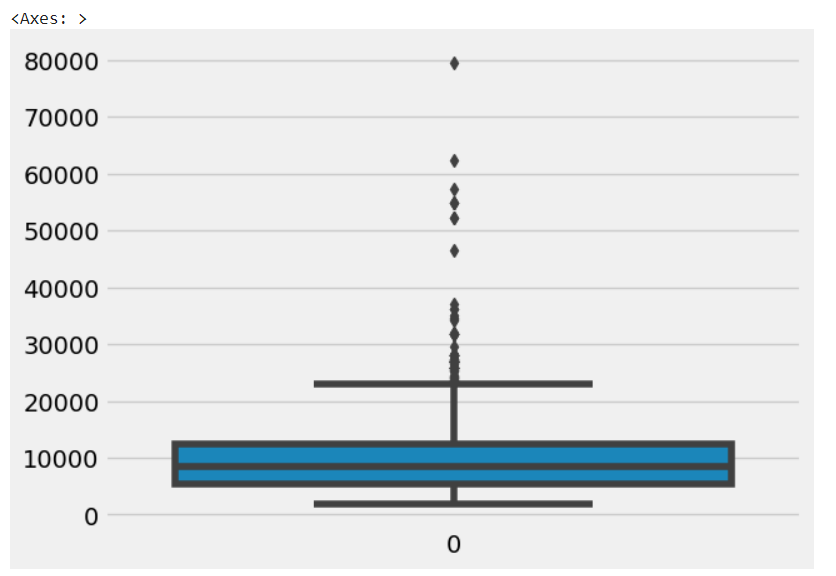
  #plt.tight\_layout(pad=3.0)

  c=c+1

  plt.show()

import seaborn as sns

sns.boxplot(data['Price'])



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)

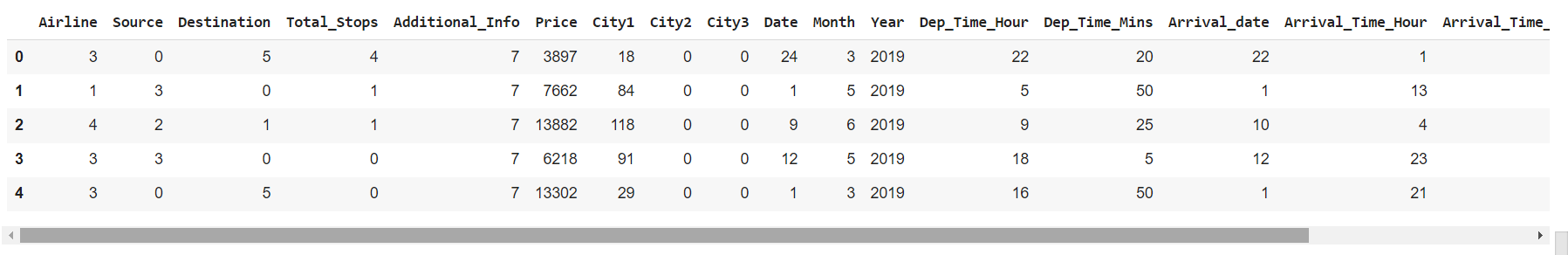
data.City1=le.fit\_transform(data.City1)

data.City2=le.fit\_transform(data.City2)

data.City3=le.fit\_transform(data.City3)

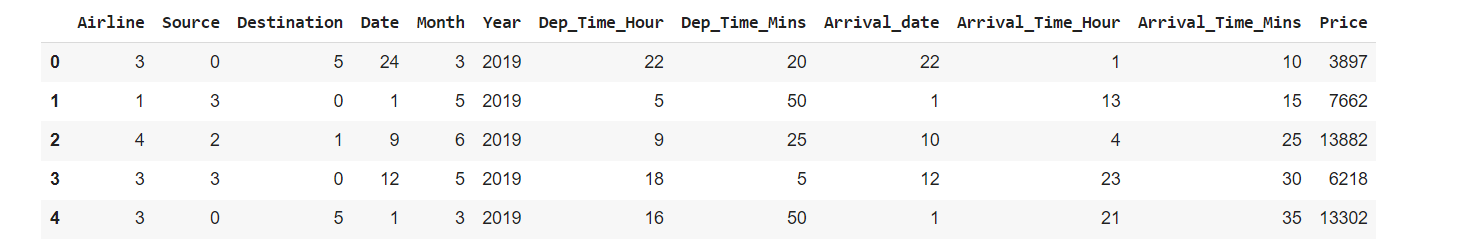
data.Additional\_Info=le.fit\_transform(data.Additional\_Info)

data.head()



data = data[['Airline','Source','Destination','Date','Month','Year','Dep\_Time\_Hour','Dep\_Time\_Mins','Arrival\_date','Arrival\_Time\_Hour','Arrival\_Time\_Mins','Price']]

data.head()



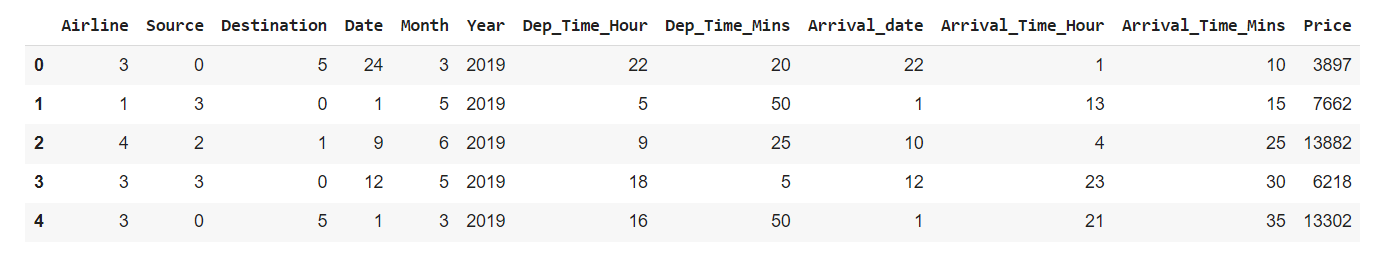
from sklearn.preprocessing import StandardScaler

ss=StandardScaler()

data1 = ss.fit\_transform(data)

data1 = pd.DataFrame(data1,columns=data.columns)

data.head()



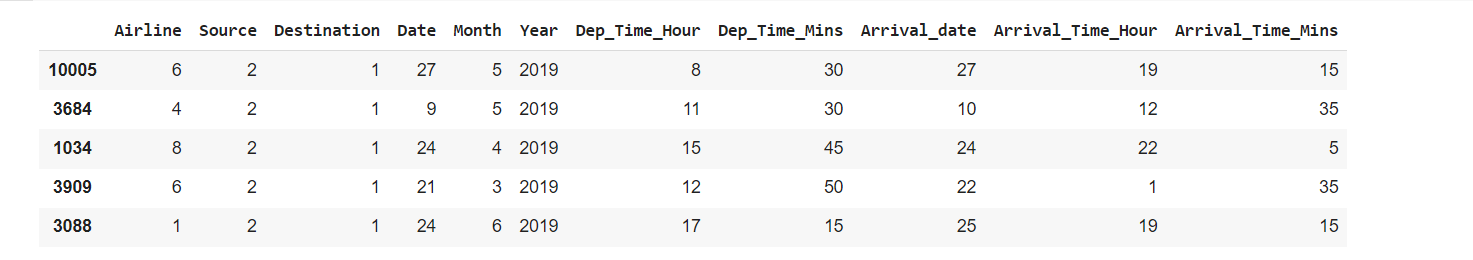
y = data['Price']

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

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()



x\_train.shape



**Milestone4:**

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)

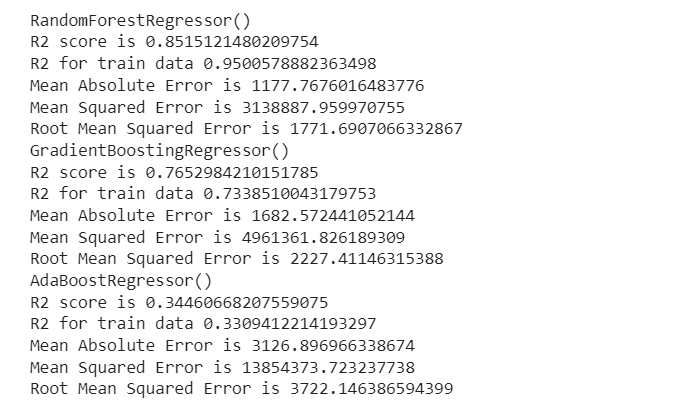
       print("R2 score is",r2\_score(y\_test,y\_pred))

       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 Squared Error is",(mean\_squared\_error(y\_pred,y\_test,squared=False)))



import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Dense, Activation, Dropout

from tensorflow.keras.optimizers import Adam

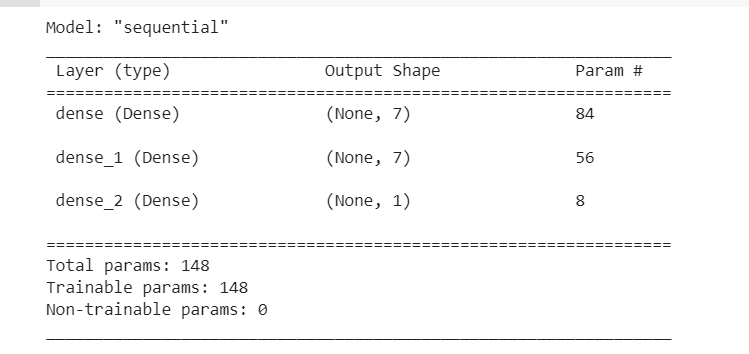
model=keras.Sequential()

model.add(Dense(7,activation ='relu',input\_dim=11))

model.add(Dense(7,activation='relu'))

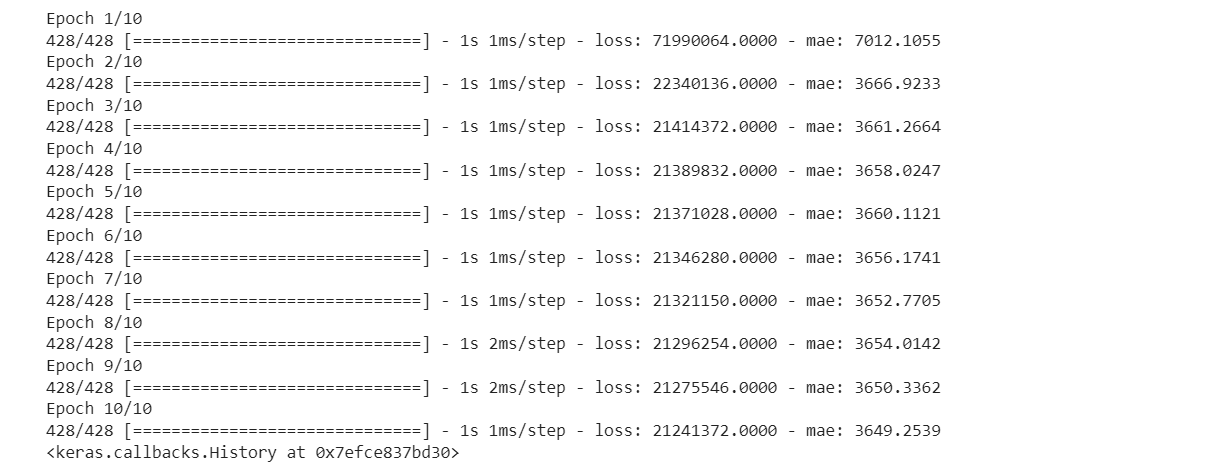
model.add(Dense(1,activation='linear'))

model.summary()



model.compile(loss = 'mse', optimizer = 'rmsprop',metrics = ['mae'])

model.fit(x\_train, y\_train, batch\_size = 20, epochs = 10)

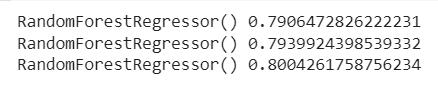


from sklearn.model\_selection import cross\_val\_score

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

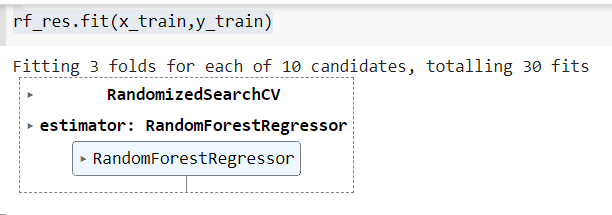
param\_grid={'n\_estimators':[10,30,50,70,100],'max\_depth':[None,1,2,3],

            'max\_features':['auto','sqrt']}

rfr=RandomForestRegressor()

rf\_res=RandomizedSearchCV(estimator=rfr,param\_distributions=param\_grid,cv=3,verbose=2,n\_jobs=-1)

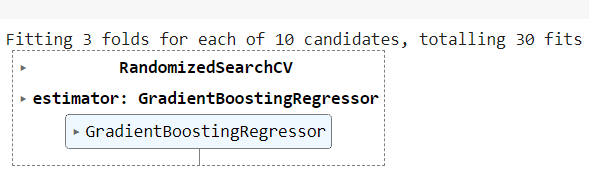
rf\_res.fit(x\_train,y\_train)



gb=GradientBoostingRegressor()

gb\_res=RandomizedSearchCV(estimator=gb,param\_distributions=param\_grid,cv=3,verbose=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))

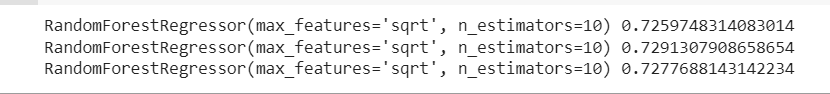


from sklearn.model\_selection import cross\_val\_score

for i in range(2,5):

  cv=cross\_val\_score(gb,x,y,cv=i)

  print(rfr,cv.mean())



gb=GradientBoostingRegressor(n\_estimators=10,max\_features='sqrt',max\_depth=None)

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

y\_train\_pred=gb.predict(x\_train)

y\_test\_pred=gb.predict(x\_test)

print("train accuracy",r2\_score(y\_train\_pred,y\_train))

print("test accuracy",r2\_score(y\_test\_pred,y\_test))



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)

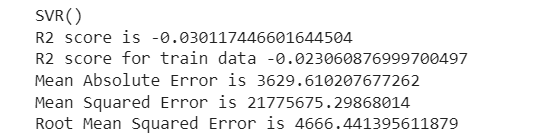
       print("R2 score is",r2\_score(y\_test,y\_pred))

       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)))



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))

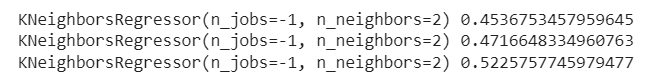


from sklearn.model\_selection import cross\_val\_score

for i in range(2,5):

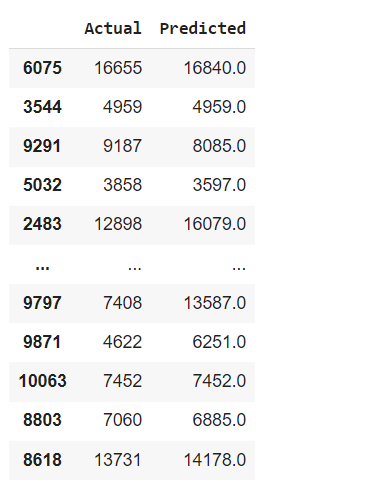
    cv=cross\_val\_score(knn,x,y,cv=i)

    print(knn,cv.mean())



predicted\_values=pd.DataFrame({'Actual':y\_test,'Predicted':y\_pred})

predicted\_values



prices=rfr.predict(x\_test)

price\_list=pd.DataFrame({'Price':prices})

price\_list

