DAYANANDA SAGAR UNIVERSITY

A MINOR PROJECT REPORT

ON

"AIRLINE PRICE PREDICTION MODEL"

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by

DEVANSH AWASTHI(ENG17CS0065)

BVM ANIRUDH(ENG17CS0048)

DEEPAK PUROHIT(ENG17CS0062)

DHRUVA SANTOSH(ENG17CS0070)

VI Semester, 2019

Under the supervision of

Dr. Suha Aimen
AssociateProfessor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF ENGINEERING

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that Mr. BVM Anirudh bearing USN ENG17CS0048, Mr. Deepak R Purohit bearing USN ENG17CS0062, Mr. Devansh Awasthi bearing USN ENG17CS0055 and Mr. Dhruva Santosh bearing USN ENG17CS0070 has satisfactorily completed their Minor Project Report as prescribed by the University for the Sixth semester B.Tech. Program in Computer Science & Engineering during the year 2019 at the School of Engineering, Dayananda Sagar University, Bangalore.

Prof Suha Aimen

(Associate Professor)

Guide **External Examiner**

Dr Reeja S R Dr. M K Banga

Project Co-Ordinator **Chairman**

ABSTRACT

Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination. From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price. In this paper, we present a review of customer side and airlines side prediction models. Our review analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date. Features extracted from external factors such as social media data and search engine query are not considered. Therefore, we introduce and discuss the concept of using social media data for ticket/demand prediction.

Devansh Awasthi(ENG17CS0065)

Deepak Purohit(ENG17CS0062)

Bvm Anirudh(ENG17CS0048)

Dhruva Santosh(ENG17CS0070)

ACKNOWLEDGEMENT

I express my sincere thanks and gratitude to my university DAYANANDA SAGAR UNIVERSITY for providing me an opportunity to fulfill my most cherished desire of reaching my goal and thus helping me to make a bright career.

I am grateful to my project guide **Prof Suha Aimen, Associate Professor,** Department of Computer Science and Engineering, DSU, Bangalore for his valuable guidance, encouragement and for extending all possible help in timely completion of the project.

I would also like to express my sincere gratitude to my project coordinator **Dr Reeja S R, Associate Professor,** Department of Computer Science & Engineering, DSU for the support for my B.Tech Study and Research.

I am highly grateful **to Dr. M.K Banga, Chairman**, Department of Computer Science and Engineering, DSU, Bangalore for his kind support, guidance and encouragement throughout the course of this work.

I express my heartfelt thanks to **Dr. A Srinivas, Dean** at the esteemed institution DSU Bangalore for providing me an opportunity to reach my goal.

I would like to thank all the teaching and non-teaching staff of Department of Computer Science and Engineering for their kind co-operation during the course of the work. The support provided by the Departmental library is gratefully acknowledged.

Finally, I am thankful to my parents and friends, who helped me in one way or the other throughout my project work.

Devansh Awasthi(ENG17CS0065)

Deepak Purohit(ENG17CS0062)

Bvm Anirudh(ENG17CS0048)

Dhruva Santosh(ENG17CS0070)

DECLARATION

We **BVM Anirudh(ENG17CS0048) ,Deepak R Purohit(ENG17CS0062),Devansh Awasthi (ENG17CS0065) and Dhruva Santosh(ENG17CS0070)** students of 6th semester **B.Tech in Computer Science and Engineering, Dayananda Sagar University,** Bengaluru, hereby declare that titled **"AIRLINE PRICE PREDICTION"** submitted to the Dayananda Sagar University during the academic year 2019-2020, is a record of an original work done by us under the guidance of **Prof Suha Aimen, Associate professor,** Department of computer science engineering, Dayananda Sagar University, Bengaluru. This project work is submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Computer Science. The result embodied in this thesis not been submitted to any other university or institute for the award of any degree.

Devansh Awasthi(ENG17CS0065)

Deepak Purohit(ENG17CS0062)

Bvm Anirudh(ENG17CS0048)

Dhruva Santosh(ENG17CS0070)

TABLE OF CONTENTS

1.INTRODUCTION

Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination.

From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price.

The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Nowadays, ticket prices can vary dynamically and significantly for the same flight, even for nearby.

The ticket price of a specific flight can change up to 7 times a day .Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit.

However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company loosing revenue.

Customer side researches focus on saving money for the customer while airline side studies are aimed at increasing the revenue of the airlines.

Conducted researches employ a variety of techniques ranging from statistical techniques such as regression to different kinds of advanced data mining techniques.

dynamic pricing enables a more optimal forecasting of ticket prices based on vibrant factors such as changes in demand and price discrimination.

dynamic pricing is challenging as it is highly influenced by various factors including internal factors, external factors, competition among airlines and strategic customers.

Internal factors consist of features such as historical ticket price data, ticket purchase date and departure date, season, holidays, supply (number of available airlines and flights), fare class, availability of seats, recent market demand and flight distance.

External factors include features such as occurrence of some event at the origin or destination city like terrorist attacks, natural disaster (hurricane, earthquake, tsunami, etc.), political instability (protest, strike, coup, resignation), concerts, festivals, conferences, political gatherings and sports events, competitors' promotions, weather conditions and economic activities

2.PROBLEM STATEMENT

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable. As data scientists, we are gonna prove that given the right data anything can be predicted. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

Size of training set: 10683 records

Size of test set: 2671 records

3.LITERATURE SURVEY

One of the pioneers on optimal ticket purchase timing prediction is probably the work done by (Etzioni et al, 2015). The authors proposed a model that advise the user whether to buy a ticket or to wait at a particular point of time.

For each query day, the model generates a buy or wait signal based on historical price information. The model uses various data mining techniques such as Rule learning (Ripper), Reinforcement learning (Q-learning), time series methods, and combinations of these to achieve various accuracy levels.

Around 12,000 historical ticket price data representing 41 departure dates for two routes was used for the analysis. The dataset has limitations in which the collection was done only starting from twenty-one days before the departure. Moreover, a constant seven days round-trip is considered.

The features used by the model include flight number, number of hours until departure, current price, airline and route (origin and destination city). Simulation is used to measure the savings passengers gained due to each of these data mining methods.

The saving (or loss) performance of a model is calculated by computing the cost due to the price difference between the ticket price at an earlier purchase point and the ticket price at the time recommended by the algorithm.

The best accuracy (61.9% as compared to optimal saving) is achieved from the combination of all the techniques used in the study.

According to (William Groves and Maria Gini, 2013), the model proposed by (Etzioni et al, 2015) has been implemented in real time for a popular ticket search website known as Bing Travel as “Fare Predictor” tool.

A closely related work to that of (Etzioni et al, 2015) is also proposed by (William Groves and Maria Gini, 2015) which predicts optimal ticket purchase timing and is in fact inspired by (Etzioni et al., 2015).

unlike (Etzioni et al, 2015), (William Groves and Maria Gini, 2015) can forecast the optimal purchase time for all available flights across different airlines for a given departure date and route.

they use a dataset that is collected 60 days ahead of the departure date. The data was collected for a period of 3 months using daily price quotes from an OTA website from February 22, 2018 to June 23, 2018. Each query returned approximately 1,200 quotes for a single route from all airlines.

4.SYSTEM REQUIREMENT SPECIFICATION:

4.1 SOFTWARE REQUIREMENTS:

Python 3.8 version

Jupiter Notebook

Modules required: Numpy, Pandas, Matplotlib, Seaborn, sklearn

4.2 HARDWARE REQUIREMENTS:

Windows 7/8/10 , Linux ,MacOS operating systems

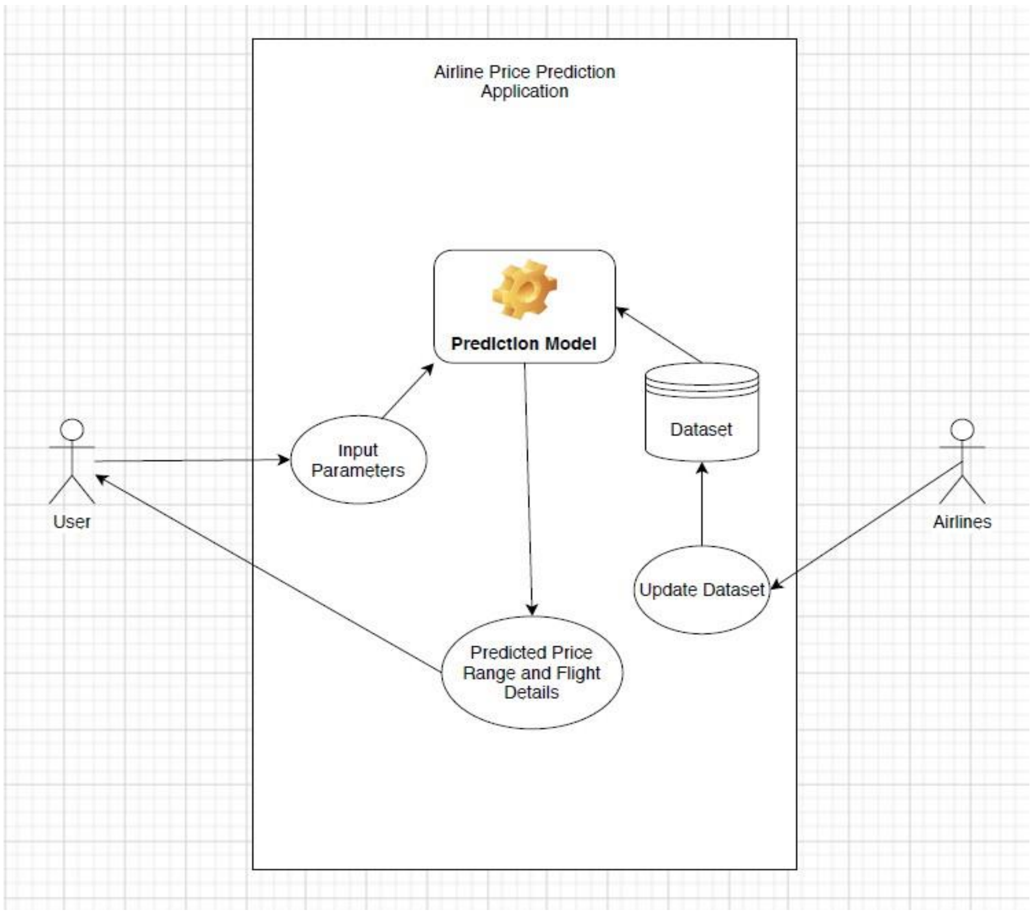
Processors: intel Atom or intel core 3 or more processor.

RAM:8GB or more

Disk space:1GB or more

5.SYSTEM DESIGN

5.1 USE-CASE DIAGRAM:


Airline Price Prediction Application
Prediction Model
Dataset
Input Parameters
User
Airlines
Update Dataset
Predicted Price Range and Flight Details

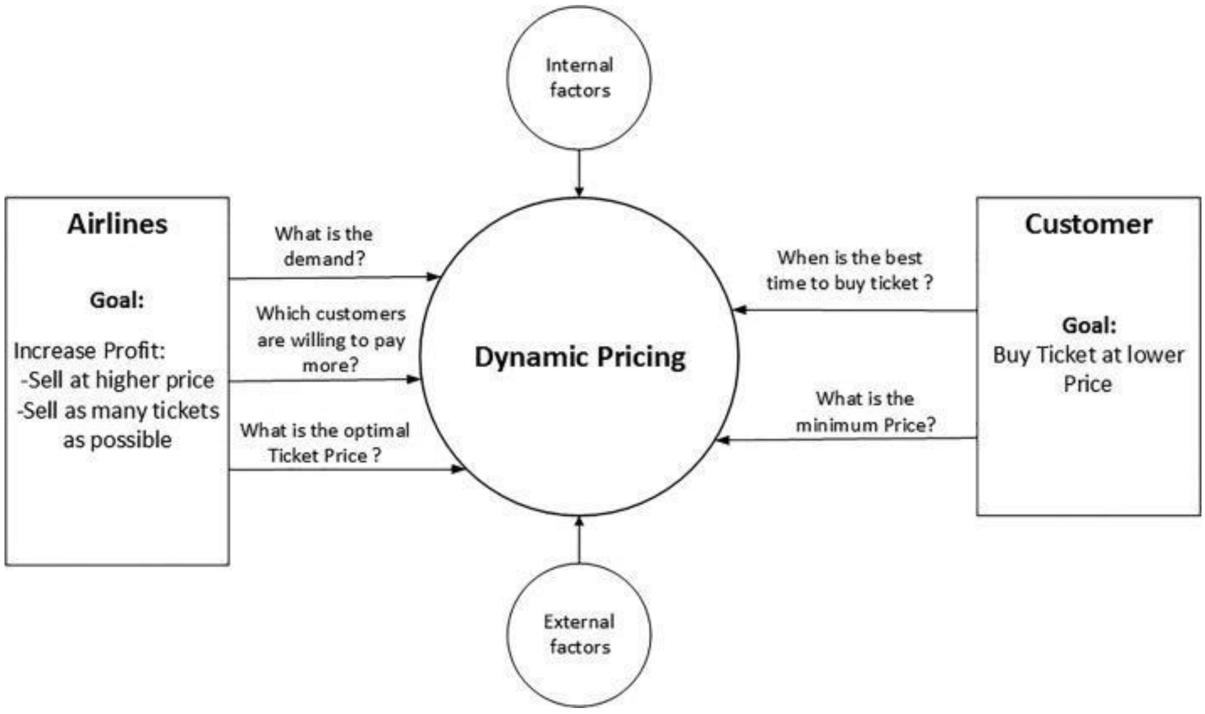
User: The user enters the parameters which are inserted in prediction model and yhe predicted prices for the given dates are shown to the User

Input Parameters: These include the date, previous dates, festive and non- festive seasons, availability of seats etc.

Dataset: These are the prices for airline seats according to its availability for given date for different destinations.

Airlines: The given airline will access the given details for the passengers and change the prices for each seat according to the demand.

6.SYSTEM ARCHITECTURE


Internal factors
Airlines
Goal:
Increase Profit:
-Sell at higher price
-Sell as many tickets as possible
What is the demand?
Which customers are willing to pay more?
What is the optimal Ticket Price ?
Dynamic Pricing
When is the best time to buy ticket ?
What is the minimum Price?
Customer
Goal:
Buy Ticket at lower Price
External factors

7.ALGORITHMS

1. Decision Tree Regressor:

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

2. Support Vector Regressor:

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

3. Random Forest Regressor:

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

4. Gradient Boost Regressor:

In boosting, each new tree is a fit on a modified version of the original data set. The gradient boosting algorithm (gbm) can be most easily explained by first introducing the AdaBoost Algorithm.The AdaBoost Algorithm begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. Here, the idea is to improve upon the predictions of the first tree. Our new model is therefore *Tree 1 + Tree 2*. We then compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

8.FEATURES OF DATA SETS

Airline: The name of the airline.

Date_of_Journey: The date of the journey

Source: The source from which the service begins.

Destination: The destination where the service ends.

Route: The route taken by the flight to reach the destination.

Dep_Time: The time when the journey starts from the source.

Arrival_Time: Time of arrival at the destination.

Duration: Total duration of the flight.

Total_Stops: Total stops between the source and destination.

Additional_Info: Additional information about the flight

Price: The price of the ticket A regression problem in which duration, source, destination, dates of flight and route are provided

9.IMPLEMENTATION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
plt.rcParams['figure.figsize'] = (12,6)
data = pd.read_excel('Data_Train.xlsx')
data.head(5)
data.info()
sns.distplot(data['Price'],kde = False,bins = 30)
sns.barplot(x = 'Price',y = 'Airline',data = data)
sns.countplot(y = 'Airline',data = data)
sns.barplot(x = 'Price',y = 'Source',data = data)
sns.barplot(x = 'Price',y = 'Destination',data = data)
sns.countplot(y = 'Source',data = data)
sns.countplot(y = 'Destination',data = data)
plt.figure(figsize = (14,8))
sns.boxplot(x = 'Price', y = 'Airline', data = data)
sns.countplot(y = 'Total_Stops',data = data)
data[data['Total_Stops'] == '4 stops']
sns.boxplot(x = 'Price',y = 'Total_Stops', data = data)
test = pd.read_excel('Test_set.xlsx')
test.head()
sum(data['Airline'].isnull())
sum(data['Total_Stops'].isnull())
data[data['Total_Stops'].isnull()]
data[(data['Source'] == 'Delhi') & (data['Destination'] == 'Cochin') & (data['Airline'] == 'Air
India') & (data['Duration'] == '23h 40m') & (data['Price'] == 7480)]
data = data.drop(data.index[9039],axis = 0)
```

```
data.iloc[9039]

sum(data['Route'].isnull())

sum(data['Arrival_Time'].isnull())

sum(data['Price'].isnull())

X = data.drop('Price',axis = 1)

y = data['Price'].astype('float32')


X['Additional_Info'] = X['Additional_Info'].replace({'No Info':'No info'})

test['Additional_Info'] = test['Additional_Info'].replace({'No Info':'No info'})

X['Total_Stops'] = X['Total_Stops'].apply(lambda x: int(x[0]) if x[0]!='n' else 0)

test['Total_Stops'] = test['Total_Stops'].apply(lambda x: int(x[0]) if x[0]!='n' else 0)

X['Date'] = X['Date_of_Journey'].apply(lambda x: int(x.split('/')[0]))

X['Month'] = X['Date_of_Journey'].apply(lambda x: int(x.split('/')[1]))

test['Date'] = test['Date_of_Journey'].apply(lambda x: int(x.split('/')[0]))

test['Month'] = test['Date_of_Journey'].apply(lambda x: int(x.split('/')[1]))

X['Overnight_flight'] = X['Arrival_Time'].apply(lambda x: 1 if len(x) > 5 else 0)

test['Overnight_flight'] = test['Arrival_Time'].apply(lambda x: 1 if len(x) > 5 else 0)

X['Date_of_Journey'] = X['Date_of_Journey'].apply(lambda x: x[-4:]+'-'+x[-7:-5]+'-'+x[-10:-
8])

X['Day_of_Week'] = pd.to_datetime(X['Date_of_Journey']).dt.dayofweek

test['Date_of_Journey'] = test['Date_of_Journey'].apply(lambda x: x[-4:]+'-'+x[-7:-5]+'-'+x[-
10:-8])

test['Day_of_Week'] = pd.to_datetime(test['Date_of_Journey']).dt.dayofweek

X['Airline'] = X['Airline'].replace({'Vistara Premium economy':'Trujet', 'Jet Airways
Business':'Trujet','Multiple carriers Premium economy':'Trujet'})

test['Airline'] = test['Airline'].replace({'Vistara Premium economy':'Trujet', 'Jet Airways
Business':'Trujet','Multiple carriers Premium economy':'Trujet'})

plt.scatter(y = X['Day_of_Week'],x = y)

# As number of counts for stops are very less we will merge 3 stops and 4 stops

X['Total_Stops'] = X['Total_Stops'].replace({4:3})

test['Total_Stops'] = test['Total_Stops'].replace({4:3})

X = X.drop(['Date_of_Journey','Route','Additional_Info'],axis = 1)
```

```
test = test.drop(['Date_of_Journey','Route','Additional_Info'],axis = 1)
X.head()
cat_col = ['Airline', 'Source', 'Destination']
from sklearn.preprocessing import LabelEncoder
for cat in cat_col:
    encoder = LabelEncoder()
    X[cat] = encoder.fit_transform(X[cat])
        test[cat] = encoder.transform(test[cat])
        def time_to_hour(time):
            if 'h' in time:
                index_h = time.index('h')
                hour = int(time[:index_h])
                minute = 0
            else:
                hour = 0
                minute = int(time[:-1])
            if len(time) > 4:
                minute = int(time[-3:-1])
                return hour + minute / 60
        X['Duration'] = X['Duration'].apply(time_to_hour)
        test['Duration'] = test['Duration'].apply(time_to_hour)
        X['Dep_Time'] = X['Dep_Time'].apply(lambda x: int(x[:2])+int(x[3:])/60)
        test['Dep_Time'] = test['Dep_Time'].apply(lambda x: int(x[:2])+int(x[3:])/60)
        test['Arrival_Time'] = test['Arrival_Time'].apply(lambda x: int(x[:2])+int(x[3:5])/60)
        X['Arrival_Time'] = X['Arrival_Time'].apply(lambda x: int(x[:2])+int(x[3:5])/60)
        X.head()
        from sklearn.model_selection import train_test_split
        X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3,random_state = 101)
        from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor(min_samples_split=10)
dtree.fit(X_train,y_train)
```

```
y_pred = dtree.predict(X_test)

from sklearn.metrics import mean_squared_log_error

print("Decision Tree Regressor 1-RMSLE:",1 -
mean_squared_log_error(y_test,y_pred)**0.5)

from sklearn.svm import SVR

svr.fit(X_train,y_train)

y_pred2 = svr.predict(X_test)

print("SVR 1-RMSLE:",1 - mean_squared_log_error(y_test,y_pred2)**0.5)

from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor(n_estimators=100,min_samples_split=12)

forest.fit(X_train,y_train)

y_pred3 = forest.predict(X_test)

y_pred3 = forest.predict(X_test)

print("Random Forest Regressor 1-RMSLE:",1 -
mean_squared_log_error(y_test,y_pred3)**0.5)

from sklearn.ensemble import GradientBoostingRegressor

Grad = GradientBoostingRegressor(learning_rate=0.1,n_estimators=100)

Grad.fit(X_train,y_train)

y_pred4 = Grad.predict(X_test)

print("Gradient Boosting Regressor 1-RMSLE:",1 -
mean_squared_log_error(y_test,y_pred4)**0.5)

forest.fit(X_test,y_test)

y_predict = forest.predict(test)

y_predict

df = pd.DataFrame(y_predict,columns = ['Price'])

df.to_excel('answers.xlsx',index = False)

df

y_pred4

y_predict

y_pred3

y_pred
```

10.ANALYSIS AND COMPARISON OF MACHINE LEARNING ALGORITHMS

- Machine learning algorithms are procedures that are implemented in code and are run on data.
- Machine learning models are output by algorithms and are comprised of model data and prediction algorithm.
- Machine learning algorithms provide a type of automatic programming where machine learning models represent the program.

So now we are familiar with a machine learning "*algorithm*" vs. a machine learning "*model*."

Specifically, an algorithm is run on data to create a model.

- Machine Learning => Machine Learning Model

We also understand that a model is comprised of both data and a procedure for how to use the data to make a prediction on new data.

- Machine Learning Model == Model Data + Prediction Algorithm

This division is very helpful in understanding a wide range of algorithms.

For example, most algorithms have all of their work in the "*algorithm*" and the "*prediction algorithm*" does very little.

Typically, the algorithm is some sort of optimization procedure that minimizes error of the model (data + prediction algorithm) on the training dataset. The linear regression algorithm is a good example. It performs an optimization process (or is solved analytically using linear algebra) to find a set of weights that minimize the sum squared error on the training dataset.

Linear Regression:

- **Algorithm**: Find set of coefficients that minimize error on training dataset
- **Model**:
- **Model Data**: Vector of coefficients
- **Prediction Algorithm**: Multiple and sum coefficients with input row

Some algorithms are trivial or even do nothing, and all of the work is in the model or prediction algorithm.

The k-nearest neighbor algorithm has no "algorithm" other than saving the entire training dataset. The model data, therefore, is the entire training dataset and all of the work is in the prediction algorithm, i.e. how a new row of data interacts with the saved training dataset to make a prediction.

k-Nearest Neighbors

- **Algorithm**: Save training data.
- **Model**:
- **Model Data**: Entire training data set.
- **Prediction Algorithm**: Find k most similar rows and average their target variable.

Decision tree

- **Algorithm**: creates a training model which can use to predict class or value of target
- **Model**:
- **Model Data**: Entire training data set.

Prediction Algorithm: The decision tree algorithm tries to solve the problem, by using **tree** representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

SVM

- **Algorithm**: used for both classification or regression challenges.
- **Model**:
- **Model Data**: Entire training data set.

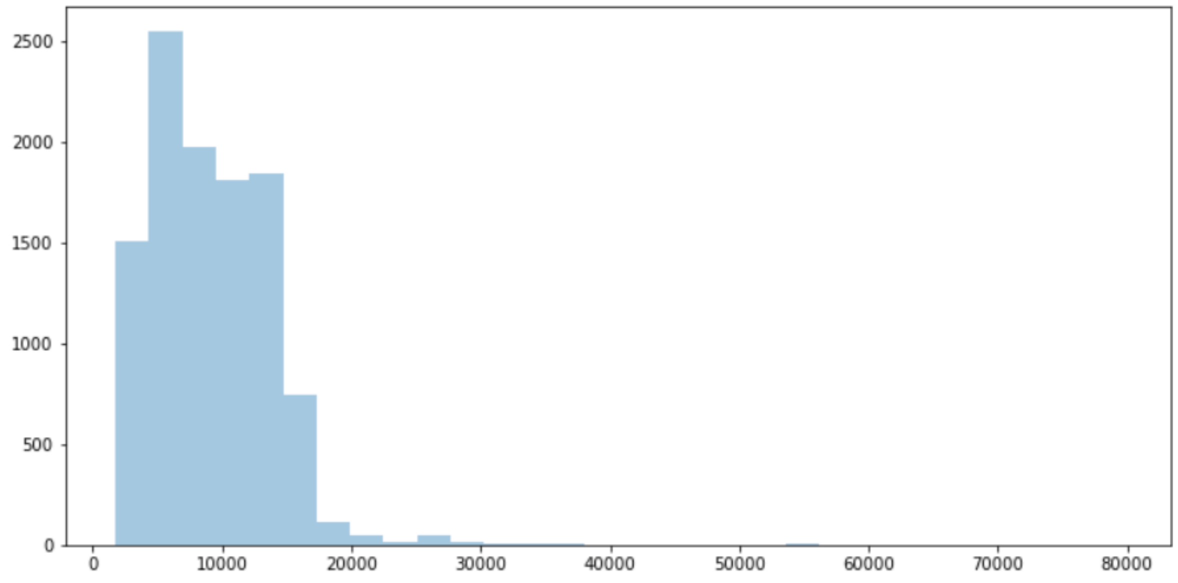
Prediction Algorithm: The SVM algorithm is implemented in practice using a kernel. A kernel transforms an **input** data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space.

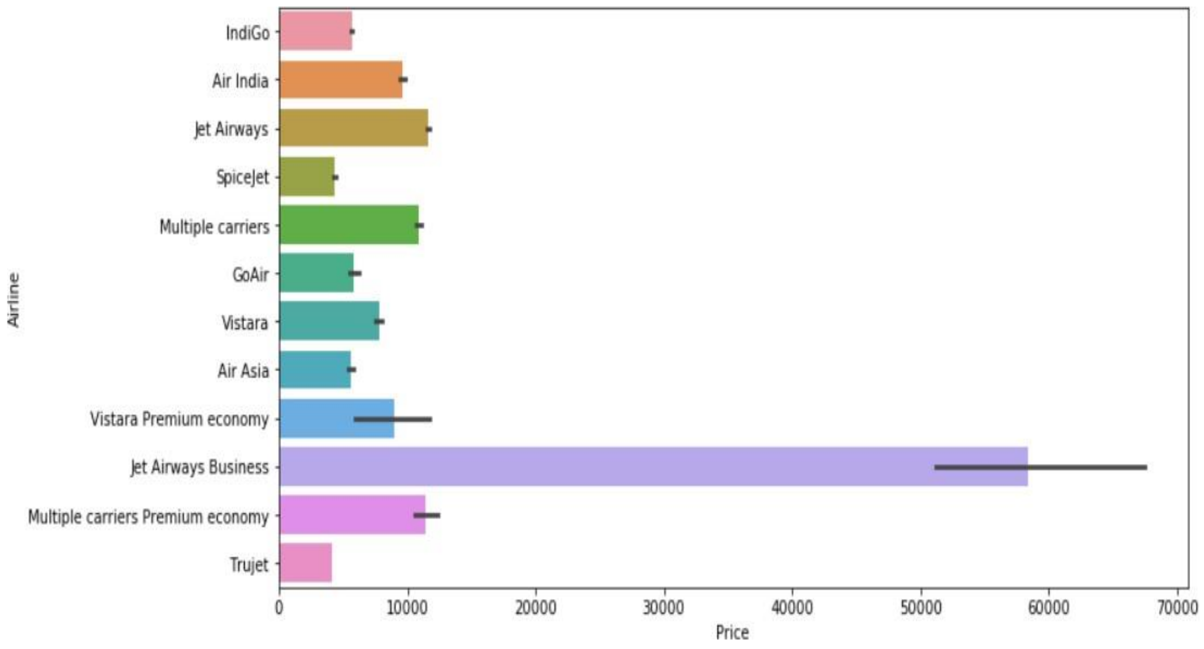
GRADIENT BOOSTING

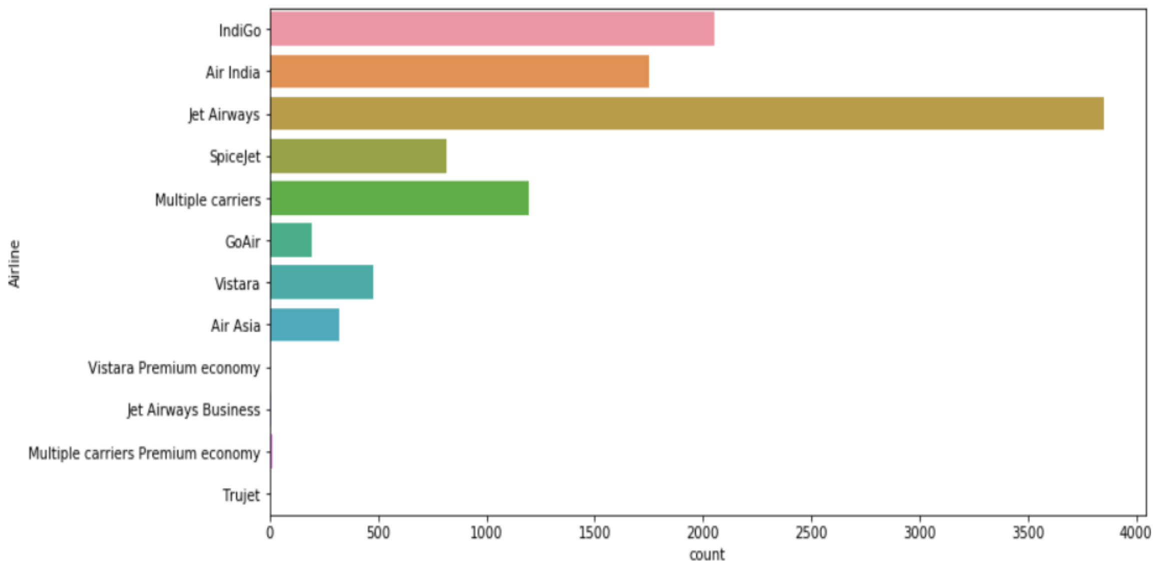
- **Algorithm**: produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.
- **Model**:
- **Model Data**: Entire training data set.

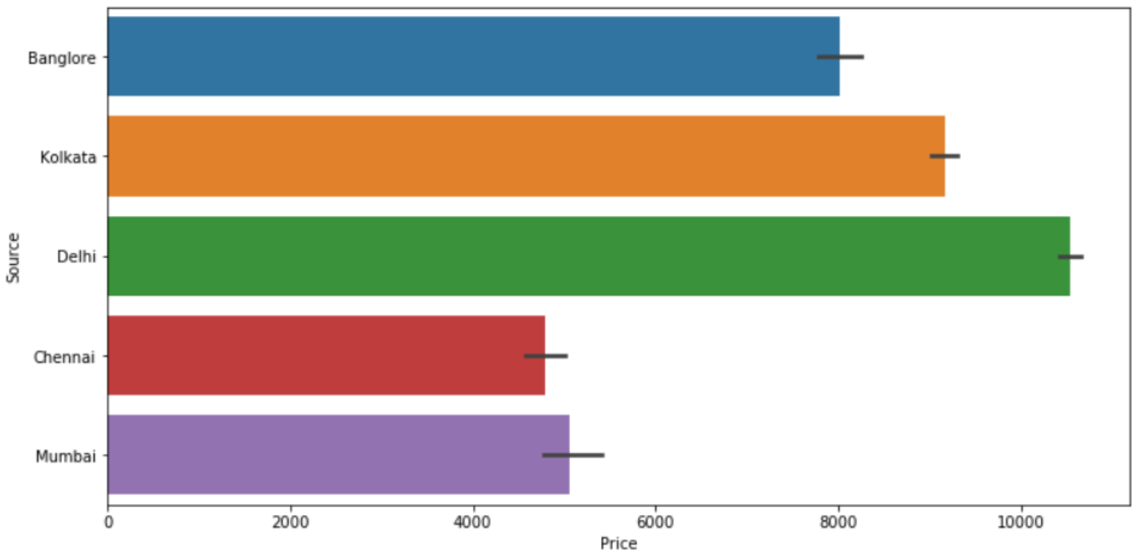
Prediction Algorithm: It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

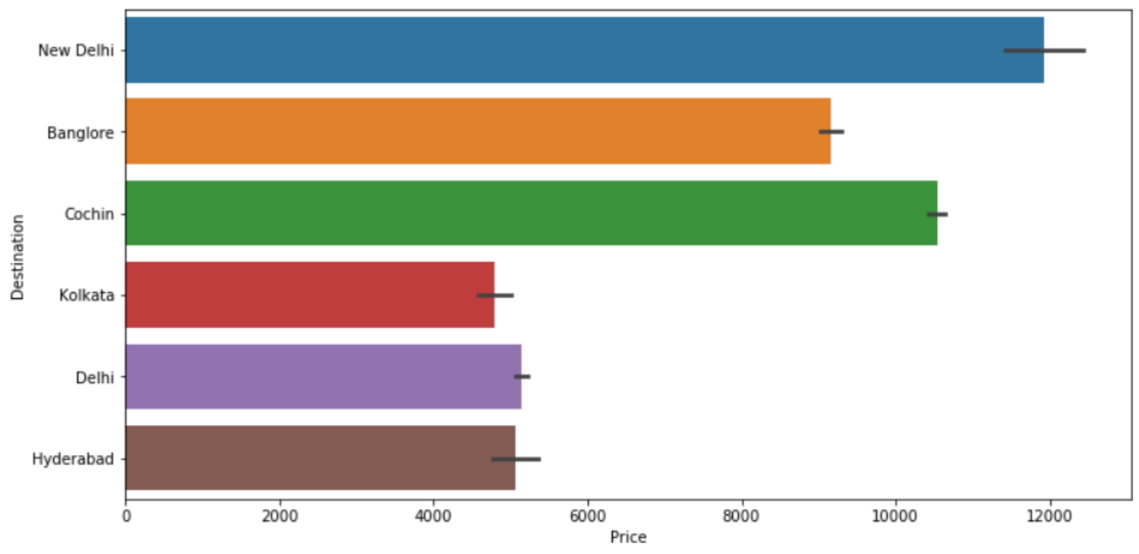
11.OUTPUT SCREEN SHOTS

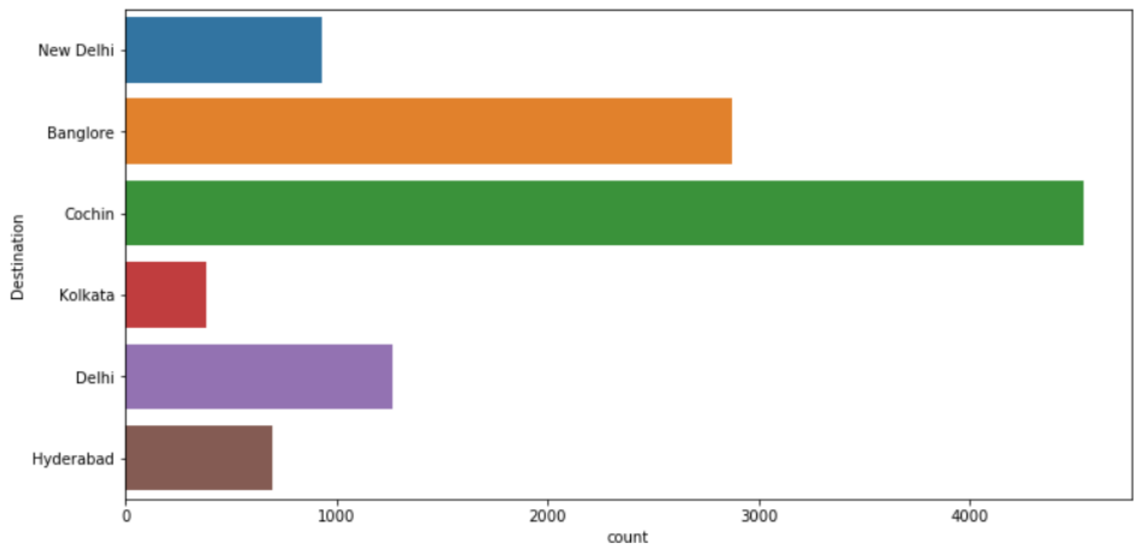


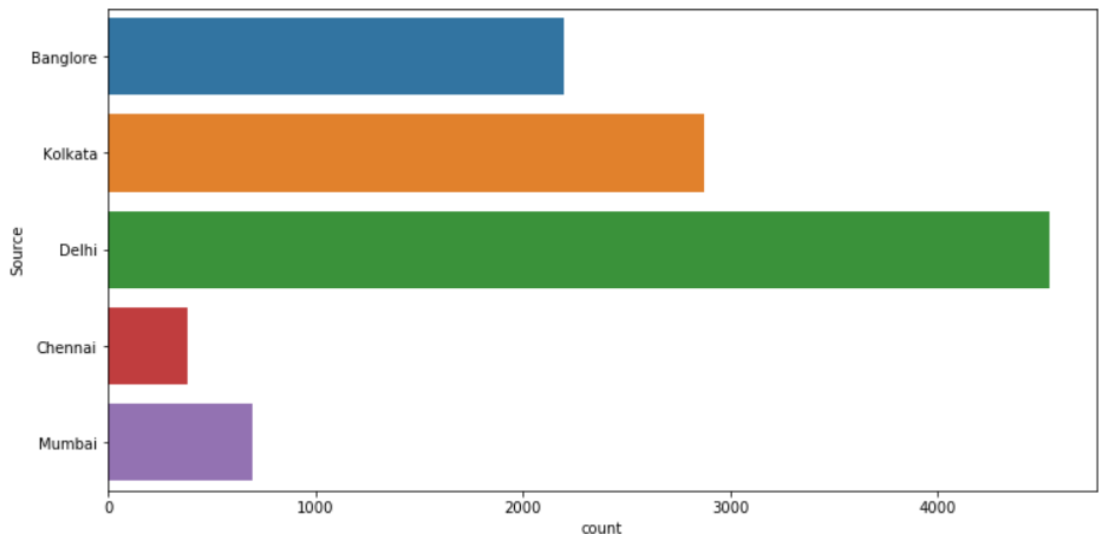


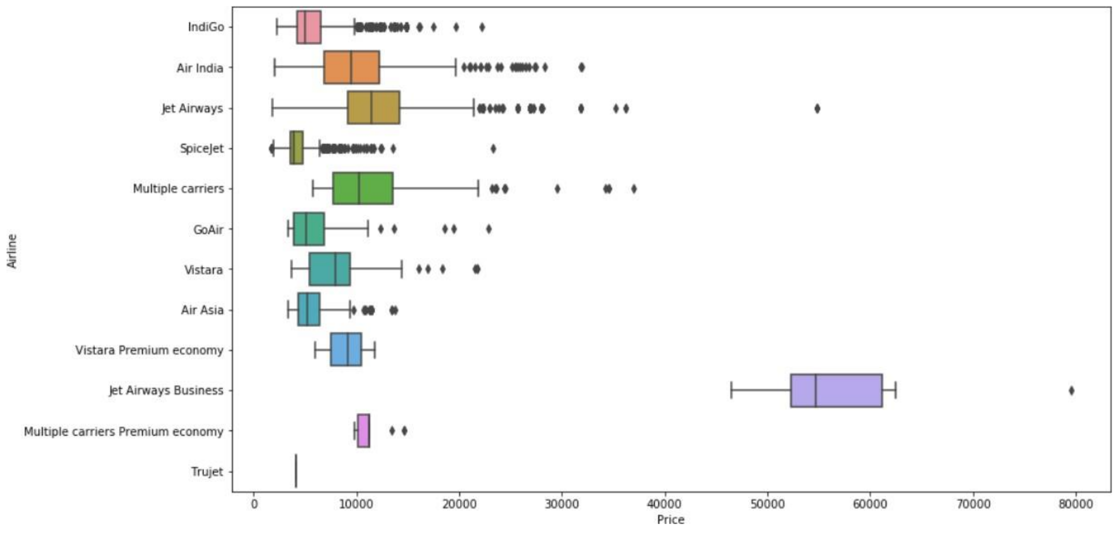

IndiGo
Air India
Jet Airways
SpiceJet
Multiple carriers
GoAir
Vistara
Air Asia
Vistara Premium economy
Jet Airways Business
Multiple carriers Premium economy
Trujet
Airline
0
500
1000
1500
2000
2500
3000
3500
4000
count

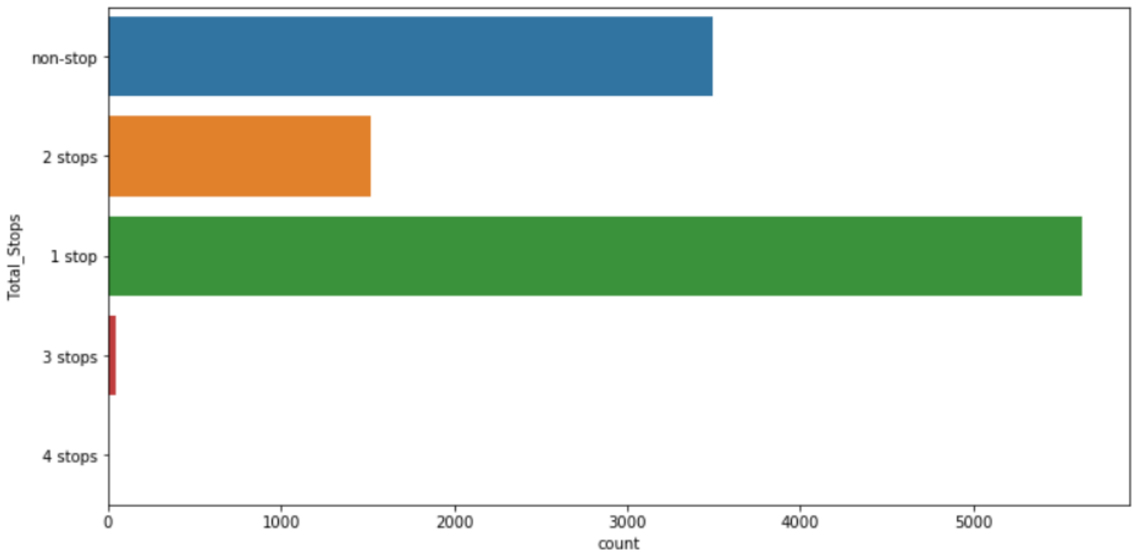

Banglore
Kolkata
Delhi
Chennai
Mumbai
Source
0
2000
4000
6000
8000
10000
Price

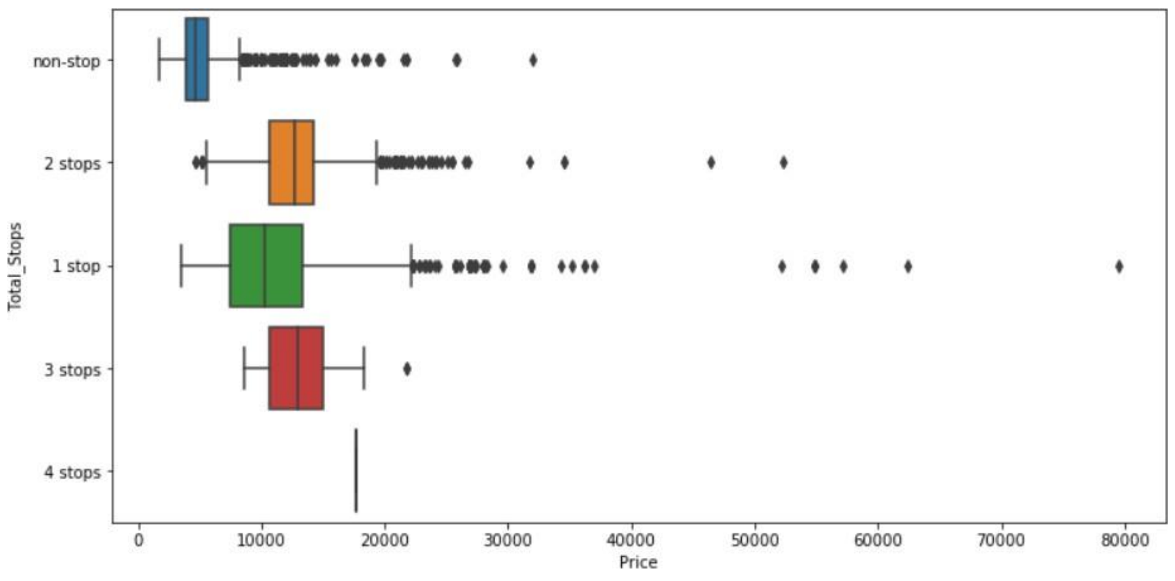

New Delhi
Banglore
Cochin
Kolkata
Delhi
Hyderabad
Destination
0
2000
4000
6000
8000
10000
12000
Price


New Delhi
Banglore
Cochin
Kolkata
Delhi
Hyderabad
Destination
0
1000
2000
3000
4000
count


Banglore
Kolkata
Delhi
Chennai
Mumbai
Source
0
1000
2000
3000
4000
count


IndiGo
Air India
Jet Airways
SpiceJet
Multiple carriers
GoAir
Vistara
Air Asia
Vistara Premium economy
Jet Airways Business
Multiple carriers Premium economy
Trujet
Airline
0
10000
20000
30000
40000
50000
60000
70000
80000
Price


non-stop
2 stops
1 stop
3 stops
4 stops
Total_Stops
0
1000
2000
3000
4000
5000
count


non-stop
2 stops
1 stop
3 stops
4 stops
Total_Stops
0
10000
20000
30000
40000
50000
60000
70000
80000
Price

12.CONCLUSION

The advantages of travelling by airplanes have increased overtime. The industry tries to make the ticket fare reasonable as well as to make profit out of it. Airline industry has lots of dynamic factors affecting them in a day today operation. It’s one of the highly sophisticated industry which aims at making revenue. The purpose of this study is to better analyze the features that affect airfare and develop and tune models to predict the airfare well in advance. We used XGBoost and Keras with a Tensorflow backend Neural Network, two state of the art prediction models for our study and used various machine learning tasks to achieve the best performance for our task.

13.FUTURE SCOPE

More routes can be added and the same analysis can be expanded to major airports and travel routes in India.

- 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.

- 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.

- Developing a more user friendly interface for various routes giving more flexibility to the users.

14.REFERENCES

a) https://docs.python.org/3/tutorial/

b) https://www.tutorialspoint.com/python/index.htm

c) https://www.coursera.org/learn/machine-learning

d) https://www.sas.com/en_in/insights/analytics/machine-learning.html

e) https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

f) https://www.dataquest.io/blog/top-10-machine-learning-algorithms-for-beginners/

g) https://www.edureka.co/blog/machine-learning-algorithms/

h) https://towardsdatascience.com/the-beginners-guide-to-selecting-machine-learning-predictive-models-in-python-f2eb594e4ddc

i) O. Etzioni, R. Tuchinda, C. A. Knoblock, and A. Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price.

j) Manolis Papadakis. Predicting Airfare Prices.

k) Groves and Gini, 2011. A Regression Model For Predicting Optimal Purchase Timing For Airline Tickets.

l) Modeling of United States Airline Fares – Using the Official Airline Guide (OAG) and Airline Origin and Destination Survey (DB1B), Krishna Rama-Murthy, 2006.

m) B. S. Everitt: The Cambridge Dictionary of Statistics, Cambridge University Press, Cambridge (3rd edition, 2006). ISBN 0-521-69027-7.

n) Bishop: Pattern Recognition and Machine Learning, Springer, ISBN 0-387-31073-8.