A PROJECT REPORT

on

"FLIGHT FARE PREDICTION"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE & ENGINEERING

BY

PRATYUSH AANAND	1905189
ASHUTOSH MISHRA	1905600
SAMBHAV CHOUDHARY	1905634

UNDER THE GUIDANCE OF
SANKALP NAYAK

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

May 2022

A PROJECT REPORT

on

"FLIGHT FARE PREDICTION"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND
ENGINEERING

BY

PRATYUSH AANAND	1905189
ASHUTOSH MISHRA	1905600
SAMBHAV CHOUDHARY	1905634

UNDER THE GUIDANCE OF
SANKALP NAYAK

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

May 2022

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"FLIGHT FARE PREDICTION"

submitted by

PRATYUSH AANAND	1905189
ASHUTOSH MISHRA	1905600
SAMBHAV CHOUDHARY	1905634

is a record of bona fide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2022-2023, under our guidance.

Date: 5/12/2022

(Sankalp Nayak)
Project Guide

Acknowledgements

We are profoundly grateful to **SANKALP NAYAK** of **School of Computer Engineering, KIIT, BBSR** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. .....

PRATYUSH AANAND
ASHUTOSH MISHRA
SAMBHAV CHOUDHARY

ABSTRACT

Indian airline sector has been on a straight upward trend since the liberalization in the 90s, the rising incomes has lead to a peak load of over 300 million airline passengers in 2019 and they have plenty of options to choose from as more and more private players continue to join what can be described as one of the most cut throat business sectors in the country which has seen many well established players going bankrupt. And to carter to the demand millions of websites have sprung to book airline tickets.

But more options also imply more hassle in choosing the right and suitable options as the prices of airline tickets are very volatile and knowing the right time and price to book a particular flight ticket can be difficult.

We try to tackle this problem using Machine Learning. We have used different “Regression Models” and compared them and chose the best performing model which can yield high-enough accurate results in predicting the flight fare using the flight information and let the user know about the optimal time for booking the flight by giving thme accurate fare predictions,

Keywords: Machine Learning, Value Prediction using Regression Models, Python, Flight Fare, Data analysis.

Contents

List of Figures

Chapter 1

Introduction

The Indian aviation industry has been on the rise since the dawn of the 21st century with private players turning what was once a monopoly now the one of the most cut throat and competitive industries in the market. The ever increasing internet penetration in our country has assured that days of over the counter bookings are long gone and Indian are no longer dependent on the travel agents for their travel bookings.

According to statista Indian airport handled a whooping 344.7 million passengers in fiscal year 2019 which was the peak of air traffic in the country. Unfortunately the pandemic caused a major dent on the industry as air travel was on halt for over a quarter in 2020 and the restrictions were eased over a period time but experts believe that the traffic would restore to the same heights by the last quarter of 2023.

The boom in the air travel industry has led to creation of lots of startups and websites specializing in airline ticket booking and the market is more competitive than ever as unlike rail bookings when IRCTC a government owned vendor is responsible for 97% of the online tickets sold by the Indian railways, the airline booking market remains open for all as multiple private players are involved, and new players are joining the race each day. Currently via.com booking.com and make my trip are the biggest players while new start ups such as easemytrip are catching up. The official sites of the carriers also attract a major chunk of this booking numbers, Indigo the countries largest domestic carrier by fare sells more than 38% of its tickets via its official website. More often than not the deciding factor for the customers these sites is the variation in the price range for the same usually it becomes a very tedious task to decide which website to chose from the array of available website.

Coupled with problem of having a flurry of options. There is the the problem that unlike train tickets, the airline fares are never static, each day and every ticket booked causes the fare to either go up or down, the price changes every hour and that's why advance booking in flights is rarely a question of seat availability but price volatility.

Tackling this problem itself has incepted an entire market of its own with sites providing notification alerts about price drops and hikes as well as giving the user the option to compare the prices across multiple available sites. The most used and by far the most successful of these sites is the Google Flights which entered Indian markets in 2012, the flight not only allows user to assess the prices across every major airline booking site but also provides notification alerts when they are changes in the flight prices.

Aside from the above mentioned features the Google flight also uses one of the most sophisticated ML models to predict whether the price currently beign shown to the user is the most suitable fare or there are chances of the fare of further increasing or decreasing, on top of that google provide the optimal cost.

Google has access to user data as well as airline and flight details and their vast models can churn out numbers almost 100% accurate and thus ensuring that their users get the right result and lead to customer retention.

Our intent for this project was same as google flights or any other flight prediction sites/apps/models available that aim to provide the customer with accurate information regarding the price of their airplane ticket, and using different machine learning regression models we were able to come up with systems that could predict the fare very accurately, these models have great scope in the future because as stated earlier the airline travel industry in India is booming and for the majority the deciding factor behind choosing the airline or flight boils down to the price. Our models takes some flight information and we can predict its price thus enabling a user to make more informed decisions. We would like to further enhance the project in the future so that it can make real time assessments and predict prices more accurately, getting results and can be used to determine the ideal price as well as ideal airline carrier to purchase tickets for a particular route.

Chapter 2

Basic Concepts/ Literature Review

Flight fare prediction is aimed towards intimating the customer/user to make an informed decision regarding the optimal time to purchase the flight ticket, sites like GoogleFlights use this to convey to the customer regarding whether the time of booking is optimal or not.

But sites like Google have an abundance of real time data directly from carriers as well as the luxury of storing vast amounts of data spanning across years to verify and identify changes in flight fare trends and make precise predictions despite them.

We need a vast dataset that spanned across a long period of time and had enough features to properly train the models because some features played a key role in determining the price variation e.g., the price of airline ticket is heavily dependent on the carrier as different carrier charge different fare based on their brand value, another example can be the flight fare varies based on their source-destination airports as the frequency of flights varies and hence carries have flexibility to decide the fare based on the competition they face in that particular route.

To understand such trends the models, have to be trained on a vast set with enough variations so that they can pick the exact trend causing an exact hike or drop in the price.

For the project we researched datasets from numerous sources and we decided to got with this particular dataset from Kaggle, the reason for this particular dataset was 1) The dataset was vast as well as has good variations in terms of time period, route, layovers, etc. 2) It was the one of the very few datasets that had domestic(Indian) carrier and flight information as most of the datasets were based on international flight information.

The database itself has 10683 entries and 11 columns including the price.
To list feature they were:

1. Airline
2. Date of Journey
3. Source
4. Destination
5. Route
6. Departure time
7. Arrival time
8. Durations
9. Stops
10. Additional Information

And the 11th parameter was price itself which had to be predicted based on these features

The Additional Information attribute contains amenities that came included or were excluded in the price like in-flight meals, only hand baggage allowance, business class seat, etc.

Some parameters required to be split in order to gain and use all the desired features properly so we made some amends.

The date of journey was in dd/mm/yyyy format and had to be broken down into three separate attributes namely 1)dd 2)mm 3)yyyy.

Similarly the Duration, Arriva time and Departure time were in the hh:mm format and had to be broken down into 2 separate formats denoting the hh and mm separately.

The route attribute contained the entire route of the flight rom its source to destination including all the layovers the flights would make, we had flights having multiple stop before reaching the destination we again broke down all the stoppages as each represented a distinct city so we took a variable 'ri' and split the cities in route and start assigning the cities starting from r1 and all the subsequent stops and destination was represented using the same format by incrementing the value of I by this method the route feature was divided and substituted into 5 features represented by ri features namely r1, r2… so on.

The dataset was raw in nature and required analysis and refining to be able build regression models upon.

The vast number of logs and the diversity of the dataset helped us in model building and model comparison as there was ample amount of data to be trained upon which helped us in achieving our objective.

Chapter 3

Problem Statement / Requirement Specifications

Using the dataset, we are going analyze and understand the trend and deciding factor on which price of a ticket depend. Then using that knowledge and our understanding of machine learning, we are going to apply regression modelling to train certain model on the dataset so it can predict flight fare. And finally compare the models.

3.1 Project Planning

First of our approach would be to understand and grasp the given dataset so that further strategy could be developed. From a layman perspective it appears as the dataset is a little complex and contains variety of data, and appears to be requiring a lot of data engineering to be able to made compatible to the machine. And a lot of important information is required to be extracted from the existing features. Along with that phase we are also going to see what the relation and trends are their in the data. As some of the data is categorical. We have to either label encode or one hot encode them. Once done with this, next phase would be to see what are correlation between the newly formed dataset and how much they are influencing the end result, following with dropping of unnecessary data. Then the machine learning part would begin and we will split the data into test and training set and apply different models to them and analyze the best performing model. And choosing top 2 performing models in order to hyper tune them in search of best parameter for them in hopes of increasing the accuracy.

3.2 Project Analysis

Data set consists of 10,683 entry which is quite a lot it can prove better for fitting the data according to the model. It contains total of 10 dependent attributes. Multiple of which are strings, they are to be broken and made into machine understandable data. Feature engineering would take time. Also, many of the newly created attributes would be categorical data in need of encoding. Looking at the complexity we are anticipating that label encoding would be preferable choice compared to one hot encode, as it will create multiple rows that could trap the models in curse of dimensionality.

3.3 System Design

3.3.1 Design Constraints

The whole project is done in 'python environment' either using 'Jupyter Notebook' (python version 3.9.13) or 'Google Colab' (python version 3.7.15).
Important Libraries necessary to be imported

- NumPy :- use to tackle multidimensional array, processing and operation
- Pandas :- used to analyze, clean and manipulate data in our project we have used to import dataset
- Matplotlib :- these are visualizing tools for python
- Seaborn :- its based on matplotlib, its use is to visualize high level statistical data
- Scikit-learn :- library for ML processing

PC Requirement

- Intel® Core™ i5-8250U CPU @ 1.60 GHZ 1.80 GHz
- 16.0 GB Ram
- 64-bit operating system, x64-based processor

3.3.2 System Architecture **OR** Block Diagram

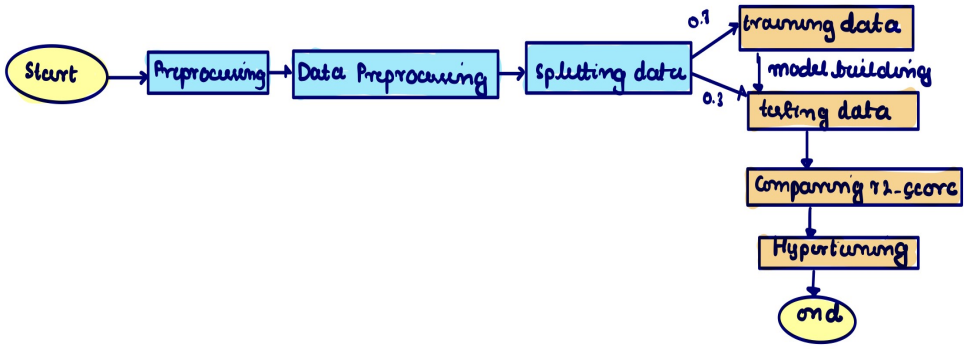

Fig 3.1 : Flow diagram of the project

We will start with importing important libraries and dataset. Next step will be to visualize and analyze the attributes and taking out useless data and simplifying the data. We will create some new columns containing data and see the collinearity and feature select those attributes that are necessary. Next step will be to split data into training and test in the ratio 7:3, (70% data approximately 7478 entries will be for training and the rest 30% 3204 will be for training purposes). And training set will be used to fit the model, after which training set will be used to check how close the predicted value is to the actual value using r2_score. Support Vector Regression's perquisite is feature scaling which will be done by us just for that. As implementing the models we will calculates each's r2_score, mean absolute error, mean square error and root mean square error and use represent them for comparison purposes. The two models with the most r2_score will hyper tuned in hopes of seeing better performance

Chapter 4

Implementation

Regression is a machine learning tools that helps in understanding correlation between a dependent variable and one or more independent variables and uncovering association which can be further used to do extract some useful/profitable. Usually denoted by y and x respectively. It essentially does so by finding a best fitting line and see how the data is dispersed around it. The better it fits the more likely the data are related. So we will be using Regression model in order to uncover the best price prediction for a foreseeable data. The model we are going to use include

- Multiple Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forrest Regression
- XGB Boost Regression

Afterward, we will implement hyper tunning of parameters of the best performing model to see which one performs better.

4.1 Methodology OR Proposal

4.1.1 Understanding the dataset

FEATURE	DESCRIPTION
Airline	Categorical Data 1. Jet Airways 2. Indigo 3. Air India 4. Multiple Carriers 5. Spicejet 6. Vistara 7. Air Asia 8. GoAIr 9. Multiple carriers Premium Economy 10. Vistara Premium Economy 11. Trujet
Date_of_Journey	String containing Date/Month/Year
Source	Where the flight originate/takeoff (categorical data) 1. Cochin 2. Banglore 3. Delhi 4. New Delhi 5. Hyderbad

	6. Kolkata
Desitination	Where the flight terminates/Land (categorical data 1. Cochin 2. Banglore 3. Delhi 4. New Delhi 5. Hyderabad 6. Kolkata
Route	String containing detail of the route the flight takes
Dep-Time	Departure Time of flight
Arrival_Time	Arrival Time of flight
Duration	Time taken by the flight to reach from source to destination
Total_Stop	Another string telling how many stop does the flight takes beform reaching its destination
Additional_Info	1. No info 2. In-flight meal not included 3. No check-in baggage included 4. 1 long layover 5. Change airports 6. Business Class 7. No Info 8. 1 short layover 9. Red-eye flight 10. 2 long layover

Some of our attributes are categorical like Airline, Source, Destination. Additional info. While the attributes like Date_of_Journey, Dep_time, Arrival_time, Route, Duration, Total_stop carry information in them that are usefull but are in a string form and hence require new attributes be created which can be easily processed. This part carries data extraction which will increase the no of attributes. Once this is done we need to understand how the features affect the data. And then split them in 7:3 ratio, we have use random_state=0 as we are comparing the models so its necessary to have same train and test set during different code runs. Most of the algorithm that we used doesn't require feature scaling except svr, so we have created separate variable for it. fit will help give us mean and standard deviation of the data while transform will use that info and scale the data. So we will apply fit_transform(does both) on our training data. And using the mean and variance of the training set transform our test set by transform function. If we apply fit_transform on training set it will create a new mean and variance, which could lead to data leakage.

4.1.1 Preprocesing

In this step we will apply basic function to out imported dataset like the

1. Head() :- shows first 5 entries in the dataset
2. Tail() :- shows the last 5 entries in the dataset
3. .shape() :- returns a tupple containing first element being no of rows and second being no of coloumns result = (10683,11)
4. Describe() :- return a dataframe consisting with each row representing the following : count, mean, standar deviation, minimum value, 25%

percentile, 50% percentile, 75% percentile, maximum value. of price attribute.

5. .info() :- prints a summary of the dataframe (attributes name, no. of non -null count, datatype description)

Next we will remove the missing data set with the function using .dropna() function

4.1.2 Feature Engineering

1. Date_of_Journey

```
In [145]: dataframe['Date']=dataframe['Date_of_Journey'].str.split('/').str[0]
          dataframe['Month']=dataframe['Date_of_Journey'].str.split('/').str[1]
          dataframe['Year']=dataframe['Date_of_Journey'].str.split('/').str[2]
```

Fig 4.1: Extraction of date, month, year

It will be divided it into 3 new attributes called date, month, year, we have used '/' as the condition for split. The next step would be type convert them into integer types using 'astype()' function. As the data we have are all of 2019 meaning in the 'Year' column all the rows have same making it useless for further processing, so we have will drop it.

2. Total_stops

   These values in the format "'integer' stops" except for 'non-stop'. We will convert these into integer same way we did date above. First of all we need to replace 'non-stop' with '0 stop' to bring a consistency, and making use of space to extract the integer part.

3. Arrival_Time and Dep_time

   Simply we will create arrival/departure hour and min using ':' condition for split

4. Routes

   Quite long string, they are routes that a flight take some being 2 part source and destination, while some having multiple stops so we can split base on the airport its stop. Split condition can be '→ '. Using this we get 5 new attributes as r1,r2,r3,r4,r5

5. Duration

   In the form "'int'h 'int'm", difficulty here is some entries have missing h or m making the spilt here difficult. The way we tackled it is running a loop in on the string whose word length didn't came to be 2. Then simply running a check for 'h', if found adding '0m' else adding '0h'. afterward this loop taking out the integers part.

6. Updated data with extra dummy variables : 'Airline', 'Source', 'Destination', 'Additional_Info', 'Date', 'Month', 'Layover', 'Arrival_Hour', 'Arrival_Min', 'Dep_Hour', 'Dep_Min', 'r1', 'r2', 'r3', 'r4', 'r5', 'Dur_h', 'Dur_m', 'Price'

4.1.3. Feature selection

1. Lasso (Least Absolute Shrinkage and selection Operator): In case of multiple regression the points in a 2d space are much distant from the best fit line while in a polynomial regression there is cases of overfitting such that the best fit line passes through all points. Due avoid that you can add a penalty term which is equal to the absolute

value equal to the magnitude of the coefficient. (This is called L1 regularization). It automatically perform feature selection.

2. SelectFromModel: It's an scikit api which uses a threshold condition to determine the weightage of attributes present in the dataset. In our case we are going to give the threshold as lasso model trained at alpha(penalty term) =0.05
3. ExtraTreeRegressor : Meta estimator uses random decision tree to improve prediction and reduce over-fitting.

4.1.4. Model Training

1. Multiple Linear Regression: Just like in a linear regression where some coefficient is used to see dependency between dependent and independent variable. In this we have multiple coefficient and multiple independent variable. It's assume that there is a linear relationship between dependent and independent variable not much correlations between the other independent variables

   $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \dots\dots\dots\dots\dots\dots + \beta_n x_n + e$

   where y: dependent variable

   β_0: y-intercept

   $\beta_1 x_1$: regression coefficient (β_1) of the variable (x_i)

   e: model error
2. Support Vector Regression: in support regression instead of fitting a line we consider a tube around a with a thickness of ε and this tube is called ε sensitive tube. We can say it as margin of error we are allowing for our models to have. Any point inside it is error free, while the ones whose outside their distance would be calculated from the tube not the lines. We call them slack variables they all are vectors and they support this tube.

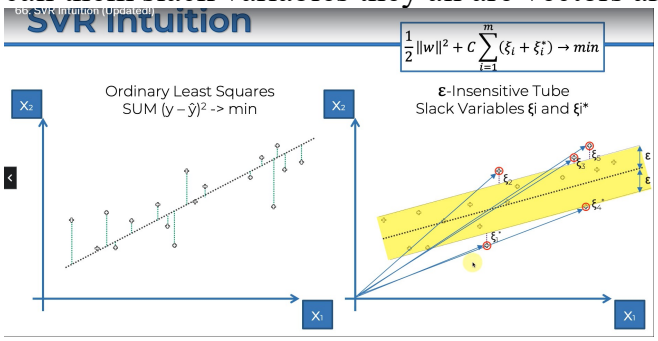

Fig 4.2: Linear v SVR intuition

3. Decision Tree Regression: In this model, an algorithm decides the condition of split and data is split based on that condition decided by the algorithm and different leaf are created containing some data. Whenever a new prediction has to be made, they are first run through the algorithm to find the best suitable leaf they belong to. In each of the leaf the average of all the data's dependent variable is calculated, and this the value that is given as the predicted value to the new data entered.

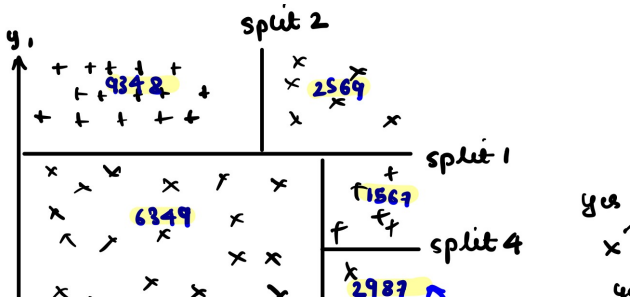

Fig 4.3 : a) decision tree split b) example of condition for split

4. Random Forrest Regression: Part of Ensemble Learning models (extra tree regressor is another example). Basically, in these types of model multiple models are clubbed together or same algorithm is applied multiple time to use everyone average to get as close as possible to the actual value. Following the algorithm, we can apply this
    - Pick at random K data points from the training set
    - Build the decision tree associated to these K data points
    - Choose the number Ntree of trees you want to build and repeat step 1&2
    - For a new data point, make each of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values
5. XGBoost : Extreme gradient boosting, another ensemble learning model.
6. RandomizedSearchCV: hyperparameter tunning, helps in finding the best parameter suited to the data. Unlike GridSearchCV only a fixed no. of parameter settings is used.

4.2 Testing OR Verification Plan

1. R^2 score: coefficient of determination used to see how well the model is fitted according to the given data. A value near or being 1 is considered to be well fitted while if the value goes near 0 or becomes 0 its very poorly fitted.

$$\text{Sum of Squares Regression (SSR)} = \Sigma(\hat{Y}_i - \bar{Y}_i)^2$$
$$\text{Sum of Squares Error (SSE)} = \Sigma(\hat{Y}_i - Y_i)^2$$
$$\text{Sum of Squares Total (SST)} = \Sigma(Y_i - \bar{Y})^2$$
$$\text{SST} = \text{SSR} + \text{SSE}$$
$$R^2 = 1 - \frac{\text{SSR}}{\text{SST}}$$

$\bar{Y}$ is the mean of the actual values of Y
$\hat{Y}_i$ is predicted values of Y_i.

Fig 4.4: R^2 formula and explanation

2. Mean absolute error: absolute error is the summation of difference between actual value of dependent variable and predicted value of the same. And its average is the mean absolute error.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Fig 4.5: MAE formula

Where n: total no of data, y_i: actual value, x_i: predicted value

3. Mean square error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Fig 4.6: MSE formula

Where n: total no of data, y_i: actual value, $\hat{y}_i$: predicted value

4. Root Mean Square error: is the square root of the above mean square error. RMSE=√MSE

4.3 Result Analysis OR Screenshots

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Fig 4.7: data.head()

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m	2 stops	No info	11753

Fig 4.8: data.tail()

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

Fig 4.9: data.describe()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Fig 4.10: data.info()

```
dataframe=dataframe.dropna()
```

```
dataframe.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

```
dataframe.shape
```

```
(10682, 11)
```

Fig 4.11: deleting empty cells

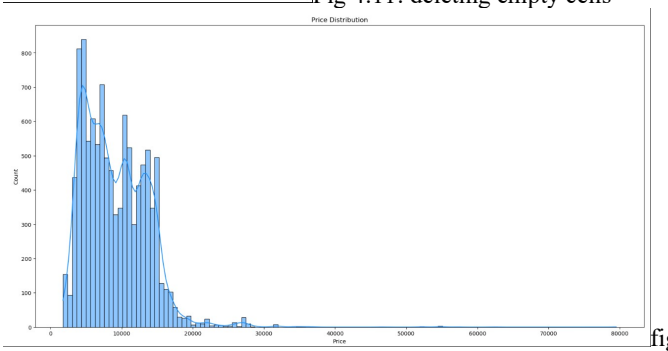

fig 4.12: Price distribution

```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

Fig 4.13: Airline counts

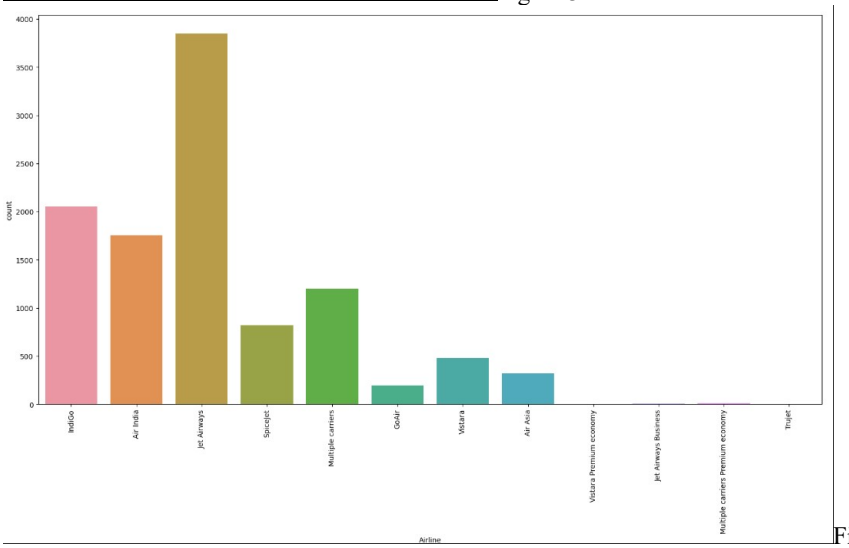

Fig 4.14: airline barplot

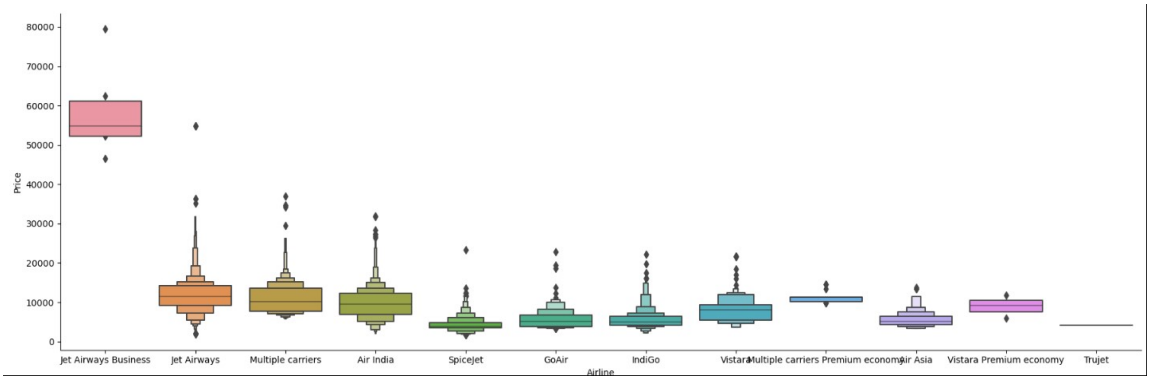

Fig 4.15: Price v airlines (catplot)

```
Delhi        4536
Kolkata      2871
Banglore     2197
Mumbai        697
Chennai       381
Name: Source, dtype: int64
```

fig 4.16: Source count

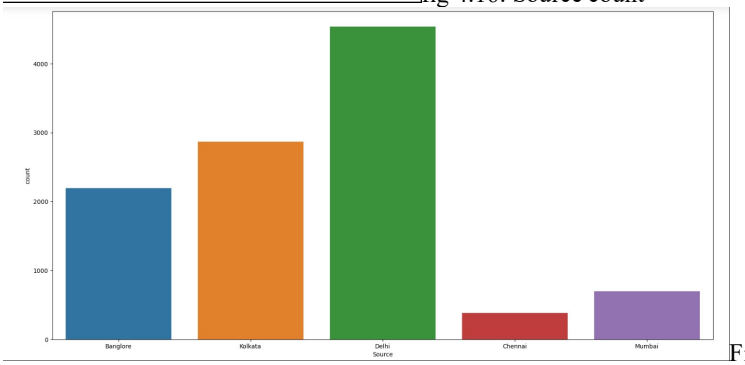

Fig 4.17: Source barplot count

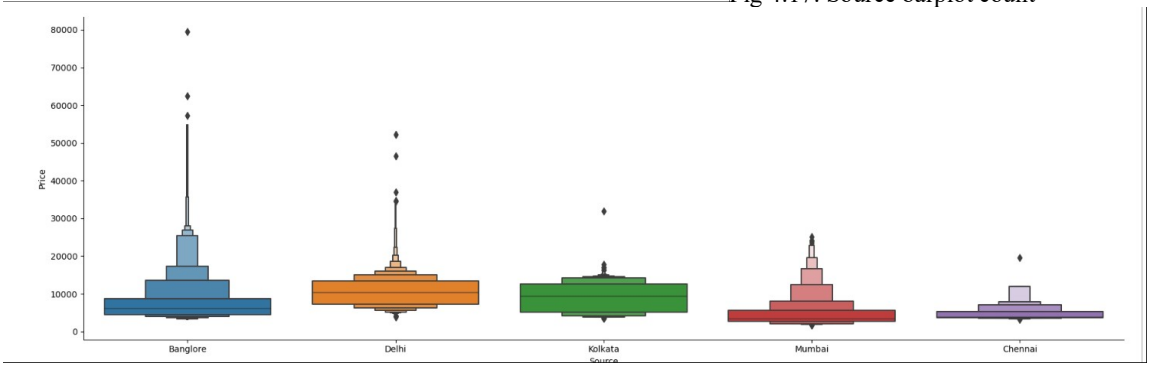

Fig 4.18: Price v Source

```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

Fig 4.19: Destination count

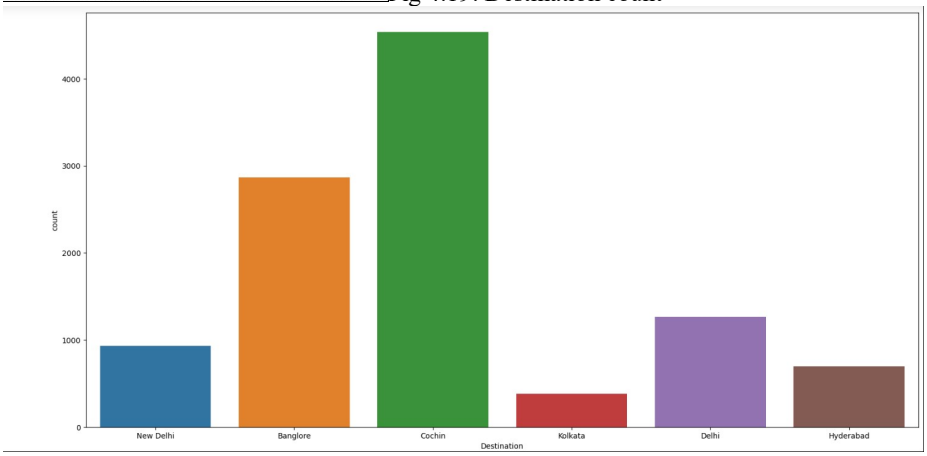

Fig 4.20: Destination barplot

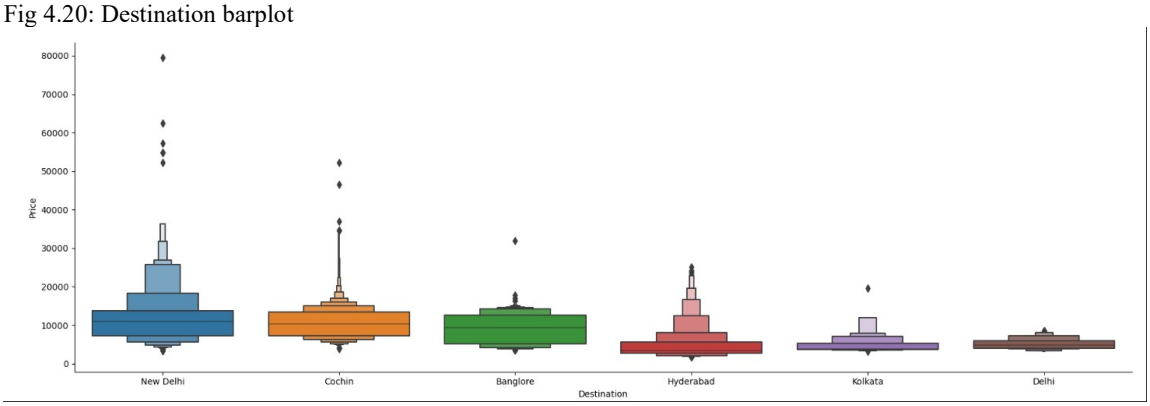

Fig 4.21: Destination v Price

```
dataframe['Date']=dataframe['Date_of_Journey'].str.split('/').str[0]
dataframe['Month']=dataframe['Date_of_Journey'].str.split('/').str[1]
dataframe['Year']=dataframe['Date_of_Journey'].str.split('/').str[2]
```

Fig 4.22: Extracting Date, Month, Year

```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
Red-eye flight                     1
2 Long layover                     1
Name: Additional_Info, dtype: int64
```

Fig 4.23: Additional Info

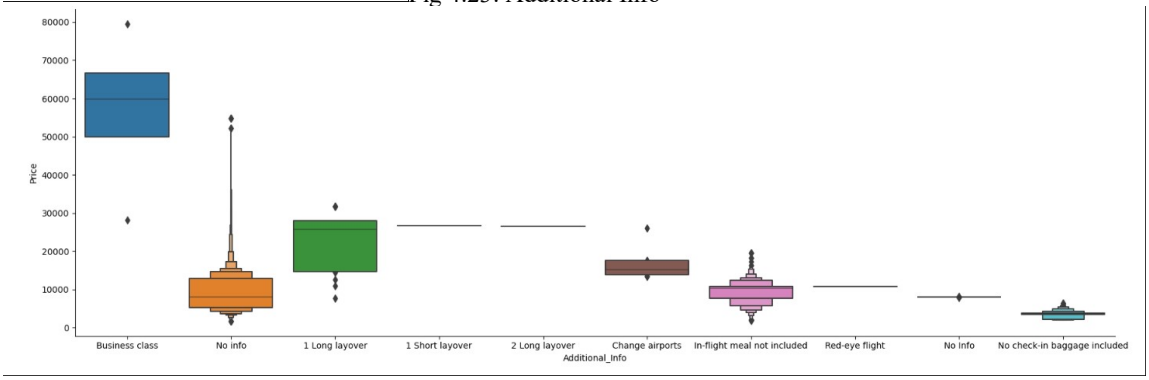

Fig 4.24: Additiona info v price

```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

Fig 4.25: Total_stops count

```
dataframe['Layover']=dataframe['Total_Stops'].str.split(" ").str[0]
```

Fig 4.26: Extracting numeric part

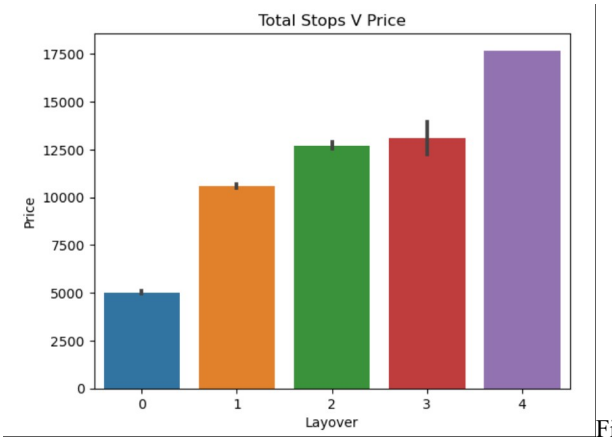

Fig 4.27: Price v layover

```
dataframe['Arrival_Hour']=dataframe['Arrival_Time'].str.split(':').str[0]
dataframe['Arrival_Min']=dataframe['Arrival_Time'].str.split(':').str[1]
```

```
dataframe['Dep_Hour']=dataframe['Dep_Time'].str.split(':').str[0]
dataframe['Dep_Min']=dataframe['Dep_Time'].str.split(':').str[1]
```

Fig 4.28: Arrival and Department time splitting

```
dataframe['r1']=dataframe['Route'].str.split('→ ').str[0]
dataframe['r2']=dataframe['Route'].str.split('→ ').str[1]
dataframe['r3']=dataframe['Route'].str.split('→ ').str[2]
dataframe['r4']=dataframe['Route'].str.split('→ ').str[3]
dataframe['r5']=dataframe['Route'].str.split('→ ').str[4]
```

```
dataframe['r1'].fillna("None",inplace=True)
dataframe['r2'].fillna("None",inplace=True)
dataframe['r3'].fillna("None",inplace=True)
dataframe['r4'].fillna("None",inplace=True)
dataframe['r5'].fillna("None",inplace=True)
```

Fig 4.29: splitting route into multple attributes

```
duration=list(dataframe['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]
dataframe['Duration']=duration
```

```
dataframe['Dur_h']=dataframe['Duration'].str.split('h').str[0]
dataframe['Dur_m']=dataframe['Duration'].str.split(' ').str[1]
dataframe['Dur_m']=dataframe['Dur_m'].str.split('m').str[0]
```

Fig 4.30: Splitting hour, minute from duration

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
dataframe['Airline']=le.fit_transform(dataframe['Airline'])
dataframe['Source']=le.fit_transform(dataframe['Source'])
dataframe['Destination']=le.fit_transform(dataframe['Destination'])
dataframe['Additional_Info']=le.fit_transform(dataframe['Additional_Info'])
dataframe['r1']=le.fit_transform(dataframe['r1'])
dataframe['r2']=le.fit_transform(dataframe['r2'])
dataframe['r3']=le.fit_transform(dataframe['r3'])
dataframe['r4']=le.fit_transform(dataframe['r4'])
dataframe['r5']=le.fit_transform(dataframe['r5'])
```

Fig 4.31: Label Encoding

	Airline	Source	Destinatio	Additional	Date	Month	Layover	Arrival_Ho	Arrival_Mi	Dep_Hour	Dep_Min	r1	r2	r3	r4	r5	Dur_h	Dur_m	Price
Airline	1	-0.0134	0.018446	-0.06075	0.026137	0.024674	-0.1994	-0.00757	-0.07109	-0.03527	-0.05992	0.035309	-0.06141	0.052975	0.261934	0.055025	-0.15814	-0.02801	-0.03956
Source	-0.0134	1	-0.59258	-0.02211	0.004902	0.183268	0.19284	0.025635	0.02104	0.059047	-0.057	0.437892	0.035847	-0.2795	-0.07735	-0.01125	0.166121	-0.14519	0.015999
Destinatio	0.018446	-0.59258	1	0.026821	-0.04102	-0.36468	-0.29548	-0.03973	0.017196	-0.073	0.127931	-0.48334	0.077887	0.423357	0.135107	0.014689	-0.25845	0.061235	-0.07112
Additional	-0.06075	-0.02211	0.026821	1	-0.0163	-0.05149	-0.08244	0.026204	0.04131	-0.05098	-0.01939	0.015319	0.077884	0.17001	-0.04293	-0.03278	-0.16882	0.04691	-0.06546
Date	0.026137	0.004902	-0.04102	-0.0163	1	-0.03836	-0.00945	-0.00324	-0.01751	0.00217	-0.00817	0.079724	-0.02474	0.021849	-0.01884	0.004578	-0.02206	-0.00894	-0.15377
Month	0.024674	0.183268	-0.36468	-0.05149	-0.03836	1	0.054383	-0.00393	-0.10063	0.039127	-0.05927	0.230657	0.048409	-0.09494	-0.07321	-0.00162	0.016141	-0.0409	-0.10364
Layover	-0.1994	0.19284	-0.29548	-0.08244	-0.00945	0.054383	1	0.03814	-0.10694	-0.06148	-0.00262	0.428918	0.162259	-0.77894	-0.72611	-0.20072	0.739916	-0.13671	0.603897
Arrival_Ho	-0.00757	0.025635	-0.03973	0.026204	-0.00324	-0.00393	0.03814	1	-0.15436	0.00518	0.043122	0.015895	-0.01222	-0.08017	0.049044	-0.02749	0.055276	-0.11831	0.024244
Arrival_Mi	-0.07109	0.02104	0.017196	0.04131	-0.01751	-0.10063	-0.10694	-0.15436	1	0.067911	-0.0176	-0.17299	0.045967	0.157028	-0.02284	0.019033	-0.07445	0.151628	-0.08616
Dep_Hour	-0.03527	0.059047	-0.073	-0.05098	0.00217	0.039127	-0.06148	0.00518	0.067911	1	-0.02475	-0.04441	-0.12621	0.006141	0.047736	0.045501	0.002869	-0.02371	0.006799
Dep_Min	-0.05992	-0.057	0.127931	-0.01939	-0.00817	-0.05927	-0.00262	0.043122	-0.0176	-0.02475	1	-0.07398	0.091579	0.051247	-0.0427	0.007104	-0.0221	0.092485	-0.02446
r1	0.035309	0.437892	-0.48334	0.015319	0.079724	0.230657	0.428918	0.015895	-0.17299	-0.04441	-0.07398	1	-0.08366	-0.4728	-0.19982	-0.01717	0.296813	-0.24399	0.182169
r2	-0.06141	0.035847	0.077887	0.077884	-0.02474	0.048409	0.162259	-0.01222	0.045967	-0.12621	0.091579	-0.08366	1	0.00101	-0.26267	-0.08679	0.035143	0.0345	-0.08724
r3	0.052975	-0.2795	0.423357	0.17001	0.021849	-0.09494	-0.77894	-0.08017	0.157028	0.006141	0.051247	-0.4728	0.00101	1	0.225283	-0.03478	-0.64563	0.160772	-0.57952
r4	0.261934	-0.07735	0.135107	-0.04293	-0.01884	-0.07321	-0.72611	0.049044	-0.02284	0.047736	-0.0427	-0.19982	-0.26267	0.225283	1	0.119488	-0.47862	0.016921	-0.31479
r5	0.055025	-0.01125	0.014689	-0.03278	0.004578	-0.00162	-0.20072	-0.02749	0.019033	0.045501	0.007104	-0.01717	-0.08679	-0.03478	0.119488	1	-0.11158	0.029886	-0.0531
Dur_h	-0.15814	0.166121	-0.25845	-0.16882	-0.02206	0.016141	0.739916	0.055276	-0.07445	0.002869	-0.0221	0.296813	0.035143	-0.64563	-0.47862	-0.11158	1	-0.12647	0.508778
Dur_m	-0.02801	-0.14519	0.061235	0.04691	-0.00894	-0.0409	-0.13671	-0.11831	0.151628	-0.02371	0.092485	-0.24399	0.0345	0.160772	0.016921	0.029886	-0.12647	1	-0.12485
Price	-0.03956	0.015999	-0.07112	-0.06546	-0.15377	-0.10364	0.603897	0.024244	-0.08616	0.006799	-0.02446	0.182169	-0.08724	-0.57952	-0.31479	-0.0531	0.508778	-0.12485	1

Fig 4.32: Collinearity

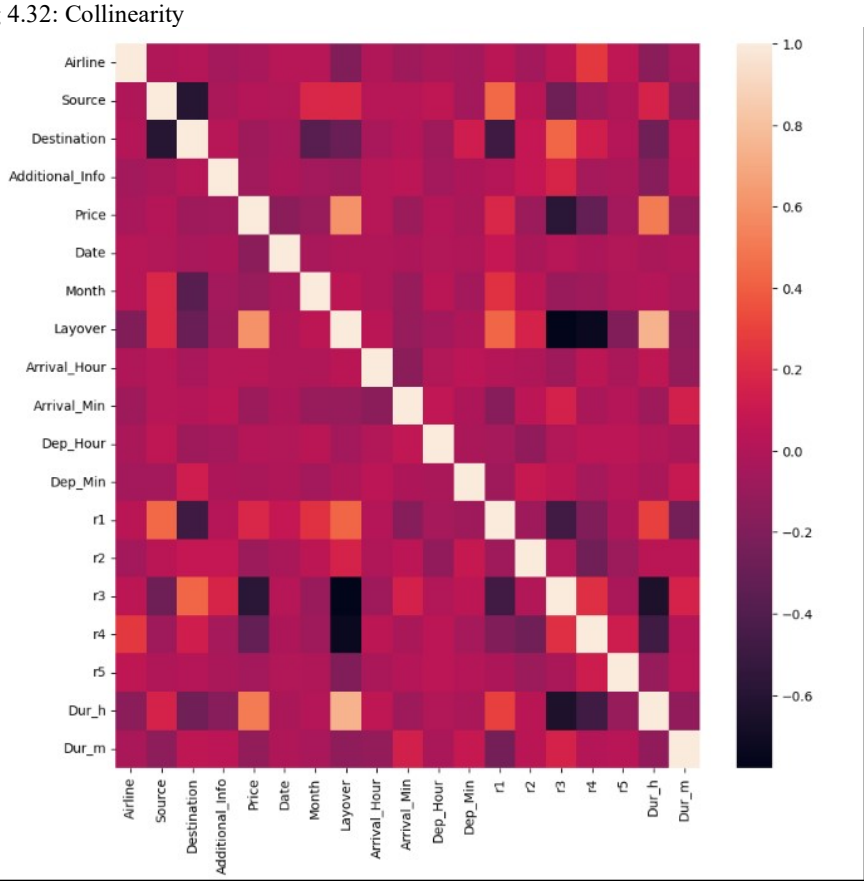

Fig 4.33 : Heatmap

```
In [77]: from sklearn.linear_model import Lasso
         from sklearn.feature_selection import SelectFromModel

In [78]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)

In [79]: model=SelectFromModel(Lasso(alpha=0.05,random_state=0))

In [80]: model.fit(x_train,y_train)

Out[80]: SelectFromModel(estimator=Lasso(alpha=0.05, random_state=0))

In [81]: model.get_support()

Out[81]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True])

In [82]: selected_features=x_train.columns[(model.get_support())]

In [83]: selected_features

Out[83]: Index(['Airline', 'Source', 'Destination', 'Additional_Info', 'Date', 'Month',
                'Layover', 'Arrival_Hour', 'Arrival_Min', 'Dep_Hour', 'Dep_Min', 'r1',
                'r2', 'r3', 'r4', 'r5', 'Dur_h', 'Dur_m'],
               dtype='object')
```

Fig 4.34: Feature Scalling

```
In [84]: from sklearn.ensemble import ExtraTreesRegressor

In [85]: ETR=ExtraTreesRegressor()
         ETR.fit(x_train,y_train)

Out[85]: ExtraTreesRegressor()

In [86]: plt.style.use('ggplot')
         plt.figure(figsize=(15,8))
         feature_importance=pd.Series(ETR.feature_importances_,index=x.columns)
         feature_importance.nlargest(35).plot(kind='bar')
         plt.show()
```

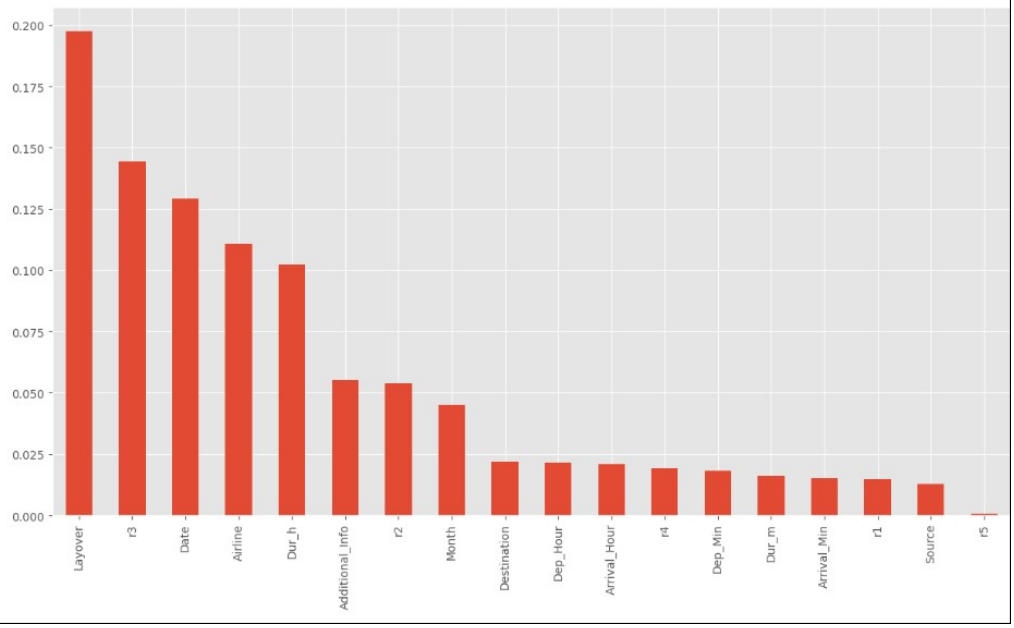

Fig 4.35 : Feature Importance

Multiple Linear Regression

```
In [92]: from sklearn.linear_model import LinearRegression
         MLR=LinearRegression()
         MLR.fit(x_train,y_train)

Out[92]: LinearRegression()

In [93]: y_pred=MLR.predict(x_test)

In [94]: from sklearn import metrics
         t=metrics.r2_score(y_test,y_pred)
         u=metrics.mean_absolute_error(y_test,y_pred)
         v=metrics.mean_squared_error(y_test,y_pred)
         w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
         print('Multiple Regression r2_score =',t)
         perf.append(['Multiple Regression',t,u,v,w])

         Multiple Regression r2_score = 0.489696475781607
```

Fig 4.36: Multiple Regression

Support Vector Regression

```
from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
sc_Y=StandardScaler()
y_train_2=y_train.reshape(len(y_train),1)
y_test_2=y_test.reshape(len(y_test),1)
x_train_2=sc_X.fit_transform(x_train)
y_train_2=sc_Y.fit_transform(y_train_2)
```

```
from sklearn.svm import SVR
regressor=SVR(kernel='rbf')
regressor.fit(x_train_2,y_train_2)
```

```
C:\Users\ashto\anaconda3\lib\site-packages\sklearn\utils
ssed when a 1d array was expected. Please change the sha
  y = column_or_1d(y, warn=True)
```

```
96]: SVR()
```

```
y_pred_2=regressor.predict(sc_X.transform(x_test))
y_pred_2=y_pred_2.reshape(len(y_pred_2),1)
y_pred_2=sc_Y.inverse_transform(y_pred_2)
```

```
t=metrics.r2_score(y_test_2,y_pred_2)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('Support Vector Regression r2_score=',t)
perf.append(['Support Vector Regression',t,u,v,w])
```

```
Support Vector Regression r2_score= 0.740900317560284
```

Fig 4.37: Support Vector Regression

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
DTR=DecisionTreeRegressor(random_state=0)
DTR.fit(x_train,y_train)
```

```
99]: DecisionTreeRegressor(random_state=0)
```

```
y_pred=DTR.predict(x_test)
```

```
t=metrics.r2_score(y_test,y_pred)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('Decision Tree r2_score =',t)
perf.append(['Decision Tree Regression',t,u,v,w])
```

```
Decision Tree r2_score = 0.7788913969921891
```

Fig 4.38: Decision Tree

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
RFR=RandomForestRegressor()
RFR.fit(x_train,y_train)
```

```
2]: RandomForestRegressor()
```

```
y_pred=RFR.predict(x_test)
```

```
t=metrics.r2_score(y_test,y_pred)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('Random Forest r2_score =',t)
perf.append(['Random Forest Regression',t,u,v,w])
```

```
Random Forest r2_score = 0.8624899551836451
```

Fig 4.39: Random Forest

XGBoost

```
from xgboost import XGBRegressor
xg_b=XGBRegressor()
```

```
xg_b.fit(x_train,y_train)
```

```
]: XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

```
y_pred=xg_b.predict(x_test)
```

```
t=metrics.r2_score(y_test,y_pred)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('XGBoost r2_score =',t)
perf.append(['XGBoost Regression',t,u,v,w])
```

```
XGBoost r2_score = 0.8593768414828512
```

Fig 4.40: XGBoost

XGBOOST Hypertuned

In [109]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [110]:
```
params={
    "learning_rate" : [0.05,0.10,0.15,0.20,0.25,0.30],
    "max_depth" : [3,4,5,6,8,10,12,15],
    "min_child_weight" : [1,3,5,7],
    "gamma" : [0.0,0.1,0.2,0.3,0.4],
    "colsmaple_bytree" : [0.3,0.4,0.5,0.7]
}
```

In [111]:
```
xgb_model_tuned=RandomizedSearchCV(estimator=xg_b,param_distributions=params,
                                   scoring='neg_mean_squared_error',n_iter=10,cv=5,verbose=2,
                                    random_state=42,n_jobs=1)
```

In [112]:
```
xgb_model_tuned.fit(x_train,y_train)
```

```
  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[CV] END colsmaple_bytree=0.7, gamma=0.1, learning_rate=0.2, max_depth=4, min_child_weight=5; to
[07:07:16] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_1.6.0/src/learner.cc:
Parameters: { "colsmaple_bytree" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[CV] END colsmaple_bytree=0.7, gamma=0.1, learning_rate=0.2, max_depth=4, min_child_weight=5; to
[07:07:16] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_1.6.0/src/learner.cc:
Parameters: { "colsmaple_bytree" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being used
```

In [113]:
```
xgb_model_tuned.best_params_
```

Out[113]:
```
{'min_child_weight': 5,
 'max_depth': 6,
 'learning_rate': 0.25,
 'gamma': 0.1,
 'colsmaple_bytree': 0.4}
```

In [114]:
```
y_pred=xgb_model_tuned.predict(x_test)
```

In [115]:
```
t=metrics.r2_score(y_test,y_pred)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('XBG Hypertuned r2_score =',t)
perf.append(['XGB Hypertuned',t,u,v,w])
```

```
XBG Hypertuned r2_score = 0.8691164517141872
```

Fig 4.41: XGBoost Hypterutned

Random Forest Hypertuned

```
param_grid={
    'n_estimators' : [int(x) for x in np.linspace(start=100,stop=1500,num=15)],
    'max_features' : ['auto','sqrt'],
    'max_depth' : [int(x) for x in np.linspace(5, 30, num=6)],
    'min_samples_split' : [2,3,5,7,10,15,100],
    'min_samples_leaf' : [1,2,3,5,8,10]
}
```

```
RFR_random=RandomizedSearchCV(estimator=RFR,param_distributions=param_grid
                              ,scoring='neg_mean_squared_error',n_iter=50,cv=5,verbose=2,
                               random_state=42,n_jobs=3
                              )
```

```
RFR_random.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
```

```
8]: RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=50, n_jobs=3,
                       param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                            'max_features': ['auto', 'sqrt'],
                                            'min_samples_leaf': [1, 2, 3, 5, 8, 10],
                                            'min_samples_split': [2, 3, 5, 7, 10,
                                                                  15, 100],
                                            'n_estimators': [100, 200, 300, 400,
                                                             500, 600, 700, 800,
                                                             900, 1000, 1100, 1200,
                                                             1300, 1400, 1500]},
                       random_state=42, scoring='neg_mean_squared_error',
                       verbose=2)
```

```
RFR_random.best_params_
```

```
9]: {'n_estimators': 500,
     'min_samples_split': 3,
     'min_samples_leaf': 1,
     'max_features': 'sqrt',
     'max_depth': 30}
```

```
y_pred=RFR_random.predict(x_test)
```

```
t=metrics.r2_score(y_test,y_pred)
u=metrics.mean_absolute_error(y_test,y_pred)
v=metrics.mean_squared_error(y_test,y_pred)
w=np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print('Random Forrest Hypertuned r2_score =',t)
perf.append(['Random Forrest Hypertuned',t,u,v,w])
```

```
Random Forrest Hypertuned r2_score = 0.8506011554318316
```

Fig 4.42: Random Forrest Hypertuned

	Names	r2_Score	MAE	MSE	RMSE
5	XGB Hypertuned	0.869116	790.031474	2.825771e+06	1681.003080
3	Random Forest Regression	0.862490	703.579753	2.968837e+06	1723.031395
4	XGBoost Regression	0.859377	795.692132	3.036049e+06	1742.426242
6	Random Forrest Hypertuned	0.850601	809.941614	3.225516e+06	1795.972152
2	Decision Tree Regression	0.778891	765.123193	4.773727e+06	2184.886088
1	Support Vector Regression	0.740900	2326.635635	1.101744e+07	3319.252285
0	Multiple Regression	0.489696	2326.635635	1.101744e+07	3319.252285

Fig 4.43: Comparision btw models

Chapter 5

Standards Adopted

5.1 Design Standards

Our project was aimed at predicting flight fares which involved building Machine Learning Regression models, training them and then testing them to measure the accuracy of their prediction, the accuracy was determined based on various parameters. The basic Schema of this project was:

- Project was build upon the interactive-python or ipython notebook(.ipynb) and first step involved importing the libraries which had to be use.
- Data collection :- we used a raw dataset imported via Kaggle which required some preparation before use.
- Exploratory Data Analysis: Involves getting familiar with the data that we would be working with and preprocessing and visualization.
- Feature Engineering: Getting familiar with the particular features, comparing them via visualization and other tools and if some features are incomparable in their actual form and thus require encoding. Some features had to be broken down as they were a amalgamation of multiple features are checked, we chose the features that were relevant for prediction and were used for model building (feature selection).
- Data splitting : Data is split into training and testing sets.
- Model building and checking performance
- Hyper tuning the best performing models
- Results

5.2 Coding Standards

The entire model was built using Python 3.10 and we used multiple python libraries such as numpy, pandas, seaborn and pyplot module of matplotlib, which are common libraries used in machine learning and data analysis. We chose Google Collaboratory as our development environment as it enabled us to collaborate on the project virtually as it enabled access to every group member via link and allowed each of use to work on the project as well as see the changes the other group member had made simultaneously.

Google colab is developed upon Jupyter notebooks which are basically simple json documents including the source script, the output(in all formats) along with the metadata, and which uses the interactive python format which can be described as an advanced command shell which enables the user to write code

and which compiles and executes it via a kernel which was python 3.10 in our case and simultaneously yields results including to rich media elements which can be the visualization of the data and results.

Coding in the .ipynb involves the scripting in python while the code in broken down in segments which are basically cell object which can be referenced by every other segment in the entire script when required or can be executed independently and the outputs can be simultaneously stored on the same .ipynb file.

There are few advantaged of choosing the jupyter notebook format over simple python scripting using functions. Coding on jupyter notebook format assists in finding problems and difficulties as each error or ill execution can be isolated to the particular cell. Another advantage is the fact that execution of each cell can be independently analyzed and understood which gave us more control and information as the project progressed and end results were achieved.

5.3 Testing Standards

We used the below listed parameters to evaluate the performance of our models. The parameter used :-

1. R-Squared(R^2) Score: also known as the coefficient of determination is the most common metric used to evaluate the fitting/performance of Regression models, it's a statistical evaluator and in simple terms measures the variation in the prediction against the actual observed value explaining the proportions of variance in the dependent variable that can be explained by the independent variable, thus it determines how good the model fits (how close the predictions were to actual value) explaining how accurate the model performed not focused on the loss it incurred, its optimal value is around 1 and closer the R^2 score is toward 1, better the model fits.

$$\text{Sum of Squares Regression (SSR)} = \Sigma(\hat{Y}_i - \bar{Y}_i)^2$$
$$\text{Sum of Squares Error (SSE)} = \Sigma(\hat{Y}_i - Y_i)^2$$
$$\text{Sum of Squares Total (SST)} = \Sigma(Y_i - \bar{Y})^2$$
$$SST = SSR + SSE$$
$$R^2 = 1 - \frac{SSR}{SST}$$

Fig 5.1: R2

2. Mean Absolute Error (MAE): Mean absolute error simply is the mean of the absolute difference between the actual simple value and predicted value, simply put taking the absolute difference between each actual and predicted value and then dividing them by the total number of samples observed. A lower MAE is desired as it is essentially a measure of the loss incurred by the model.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Fig 5.2: MAE

3. Mean Squared Error (MSE): Like the mean absolute error, mean squared error is mean the squared differences between actual and predicted values, the advantage of MSE over MAE is the fact that due to squaring of errors, the larg error are properly observed. Like the MAE it is also a loss metric hence lower MSE means better performance.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Fig 5.3: MSE

4. Root Mean Squared Error (RMSE): It's simply the square root of MSE, the advantage RMSE is that in case of MSE squaring the differences also squares the units hence the results is derived in the unit^2, so RMSE solves the problem, like the MAE and MSE lower RMSE means less loss hence better result, RMSE is generally preferred as any mathematical calculation in absolute terms is less desirable (MAE), while MSE doesn't give the results in the same unit.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The Project involved finding a suitable dataset which was vast and diverse as well as a dataset that had domestic samples as we wanted to get familiar with the Indian airline market. We took the raw data, processed it and made it useable, then trained and tested multiple regression models on the processed dataset. We also developed an understanding about how different parameters/attributes/features affected the flight fare and how important each of the feature were in determining the flight price. Our end goal was to develop a model that give accurate enough predictions so that the user can use the predictions to decide when is the optimal time to buy and airplane ticket by comparing the predicted price against the listed price or can simply get an estimate on flight fare for their travel.

Based on the results of our projects Random Forest and XGboost regressor outperformed all the other models, so we decided to apply the hyper parameter tunning on the 2 models in order to get the best result possible and XGboost hypertuned yielded the best result with the R2 score of about 0.866 with minimal loses (we used MSE, MAE and RMSE parameters) which gave accurate enough results. We were satisfied with the results as it performed well and also left room for improvement.

6.2 Future Scope

Our aim was to develop a model that have real life function but for real world deployment would require a steady and rich flow of data that is accessible to sites like google flight, so limitation of dataset was on thing that can be improved upon. Another factor that arose due to the limitation of the dataset on which we can improve upon was the feature selection as there remains multiple more factors that play a pat in the air fare estimation which were absent in our dataset so the accuracy could be improved by introducing more usable features. Once the dataset is efficient enough we can also introduce some unsupervised ML models and check their performance. Once our prediction accuracy reaches close to the industry standards we plan to scale the model and deploy web applications that cold provide access points to people to use our model and get an estimate of their flight fare and even introduce live fare tracking that can notify users about price drop, etc.

Our goal for the project was to test the regression models as well as develop an understanding of how the carriers determine the prices for their flights but the project can be scaled and deployed in the market for consumer usage

---

School of Computer Engineering, KIIT, BBSR

References

1. https://m.economictimes.com/industry/transportation/airlines-/-aviation/68-indians-book-flight-tickets-directly-with-the-airline-survey/articleshow/50350217.cms
2. https://www.statista.com/statistics/588028/passengers-boarded-by-type-by-indian-air-carriers/
3. https://factly.in/irctc-improvements-from-9-tickets-booked-in-a-day-to-13-lakh-tickets-irctc-has-come-a-long-way/
4. https://www.mordorintelligence.com/industry-reports/online-travel-market-in-india
5. https://www.goindigo.in/
6. https://en.wikipedia.org/wiki/Google_Flights
7. https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh
8. https://www.investopedia.com/terms/r/regression.asp#:~:text=A%20regression%20is%20a%20statistical,more%20of%20the%20explanatory%20variables.
9. https://docs.python.org/3/
10. https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe
11. https://www.statisticshowto.com/lasso-regression/
12. https://www.datatechnotes.com/2021/04/selectfrommodel-feature-selection.html
13. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html#:~:text=An%20extra%2Dtrees%20regressor.,accuracy%20and%20control%20over%2Dfitting.
14. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.nlargest.html
15. https://www.investopedia.com/terms/m/mlr.asp
16. https://www.udemy.com/course/machinelearning/learn/lecture/19505880#overview
17. https://www.udemy.com/course/machinelearning/learn/lecture/5732732#overview
18. https://www.python.org/downloads/release/python-3100/
19. https://numpy.org/doc/
20. https://pandas.pydata.org/docs/
21. https://seaborn.pydata.org/
22. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html
23. https://colab.research.google.com/
24.

Flight Fare Prediction

ASHUTOSH MISHRA
1905600

Abstract: Flight price ticket depends upon many factors which leads to a spontaneity. Machines learning helps us to make sense of spontaneity and reveals many things and factors associated with target value. Using its application, we tried many regression models on dataset imported from Kaggle to find the best fitting model to predict the price.

Individual contribution and findings: Task I took upon myself was feature engineering, features selection, feature scaling, hyper tunning. My first approach was to see what are types and how to make them useful. Many data were easy to extract with the mercy of split function. But some data were non coherent like the stop except for nonstop all the were numeric followed by useless data used replace to make it coherent, this was easy but the duration was a little difficult had to use a list to extract the whole feature and using a loop to identify data that were missing and taking the necessary measure. For the feature selection use lasso and model selection to see which of my edited attributes were useful. Turns out all of them. After that used extratressregressor to show importance of each feature. Next, I did label encoding after a group discussion. Next, the model my peer selected except for SVR were able to work without feature scaling, so I created an alternate variable to make the task easier for them. Finally, I did the hyper tunning of the two best performing models Random Forrest XGBoost. I used documentary and public forums to understand best parameter I could choose, and used that knowledge to my disposal. XGBoost ever so slightly did better but it's sad that random forest didn't. Due to constraint this was the last stretch.

Individual contribution to project report preparation: I wrote chapter 3,4 and any unquoted drawing used in this presentation was prepared by me.

Individual contribution for project presentation and demonstration: here also I created the slide of feature engineering, feature selection, hyper tunning.

Full Signature of Supervisor: ……

Full signature of the student: ……..

Flight Fare Prediction

SAMBHAV CHOUDHARY
1905634

Abstract: Flight price ticket depends upon many factors which leads to a spontaneity. Machines learning helps us to make sense of spontaneity and reveals many things and factors associated with target value. Using its dapplication, we tried many regression models on dataset imported from Kaggle to find the best fitting model to predict the price.

Individual contribution and findings: Task assigned to me was preprocessing, splitting the data and applying the model along with testing. I started my part with usual dataframe functions like info, head, tail, describe and removing empty dataset. For splitting as the data set is so large no matter what we threw at it had more than enough data to train upon. I tried 7:3, 3:1 ,4:1. They somewhat gave the same result so I settled with 7:3 I also used random state 0 here to keep the split constant along all the iteration. After a long and productive discussion with my friend I decided to use Multiple Regression, SVR, Decision Tree and Random Forrest regression along XGboost. This decision was taken with taking everyone understanding of the model. Decision was taken and I implemented all these models. Next step was to check them so I used R^2 score along with mean absolute error, mean square error and root mean square error. Found out XGBoost and Random Forest were at each other throats with little more 0.01 as difference so, we decided to hyper tune these two

Individual contribution to project report preparation: I wrote chapter 1, 2 and along with chapter 5. I used a lot of source to back my claims so the referencing part for the whole document fell under my shoulder.

Individual contribution for project presentation and demonstration: created slide for preprocessing, splitting the data and applying the model.

Full Signature of Supervisor: ……

Full signature of the student: ……..

Flight Fare Prediction

PRATYUSH AANAND
1905189

Abstract: Flight price ticket depends upon many factors which leads to a spontaneity. Machines learning helps us to make sense of spontaneity and reveals many things and factors associated with target value. Using its application, we tried many regression models on dataset imported from Kaggle to find the best fitting model to predict the price.

Individual contribution and findings: My peers trusted me with task to visualize the data and keeping track of the performances. I started with usually the price distribution using a histplot which showed the most demanded price segment were around 5000 indicating a huge middle-class traveler. Count plot was used to show the distribution of all the data of independent variable. Next used catplot to show bivariate comparison between all the independent variable and dependent variable. Kind was boxen as it reveals much more about data things such as median, 75%ile, 25%tile of the data etc. For tracking the data I used a list and stored R^2 score, MAE, MSE, RMSE. After data storage I used the list and converted it into a data frame sorted on the basis of R^2 score. And finally displayed it.

Individual contribution to project report preparation: I did a lot of back end part, I wrote the abstract and conclusion. And used the chapter provided by my friends and edited them to the prescribed format of report. Provided the images either from the project or internet

Individual contribution for project presentation and demonstration: Similarly I prepared the slides for visualization final result and merged all the slides provided by my peers.

Full Signature of Supervisor: …….

Full signature of the student: ……..

Sambhav Choudhary

ORIGINALITY REPORT

12%	9%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	4%
2	Ashutosh Shankhdhar, Pawan Kumar Verma, Prateek Agrawal, Vishu Madaan, Charu Gupta. "Quality analysis for reliable complex multiclass neuroscience signal classification via electroencephalography", International Journal of Quality & Reliability Management, 2022 Publication	1%
3	Submitted to University of Liverpool Student Paper	1%
4	Submitted to National University of Singapore Student Paper	1%
5	katalog.ticaret.edu.tr Internet Source	<1%
6	www.ijraset.com Internet Source	<1%
7	www.irjmets.com Internet Source	<1%

8	Submitted to University of Leicester Student Paper	<1 %
9	Submitted to University of Bradford Student Paper	<1 %
10	Submitted to University of Essex Student Paper	<1 %
11	analyticsindiamag.com Internet Source	<1 %
12	archive.org Internet Source	<1 %
13	eprints.utar.edu.my Internet Source	<1 %
14	www.freepatentsonline.com Internet Source	<1 %
15	Submitted to The University of Wolverhampton Student Paper	<1 %
16	www.aiktcdspace.org:8080 Internet Source	<1 %
17	www.knosof.co.uk Internet Source	<1 %
18	docslib.org Internet Source	<1 %
19	kitakyu.repo.nii.ac.jp Internet Source	

		<1 %
20	repository.upi.edu Internet Source	<1 %
21	theses.liacs.nl Internet Source	<1 %
22	www.aionlinecourse.com Internet Source	<1 %
23	www.osmania.ac.in Internet Source	<1 %
24	Zigner, Michael. "Machine Learning Models of Participation in Work-Related Training", The Pennsylvania State University, 2021 Publication	<1 %
25	erepository.uonbi.ac.ke:8080 Internet Source	<1 %
26	levelup.gitconnected.com Internet Source	<1 %
27	pdf.xuebalib.com:1262 Internet Source	<1 %
28	www.ibm.com Internet Source	<1 %
29	Qi, L.. "Neural network prediction of carbonate lithofacies from well logs, Big Bow	<1 %

and Sand Arroyo Creek fields, Southwest Kansas", Computers and Geosciences, 200608

Publication

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Sambhav Choudhary

GRADEMARK REPORT

FINAL GRADE

/11

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38