



Flight Price Prediction Project

Submitted by:

Kshitij Chawla

ACKNOWLEDGMENT

In this project different libraries and methods are used that are available in python which helped in completion of the project:

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html>

<http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

http://scikit-learn.org/stable/modules/model_evaluation.html

http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

<https://seaborn.pydata.org/generated/seaborn.countplot.html>

<https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/>

<https://www.analyticsvidhya.com/blog/2020/10/how-to-choose-evaluation-metrics-for-classification-model/>

<https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>

INTRODUCTION

- Business Problem Framing

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on -

1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

STEPS

1. Data Collection

You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data better the model In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

2. Data Analysis

After cleaning the data, you have to do some analysis on the data.

Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time?

What is the best time to buy so that the consumer can save the

most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

Model Building

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

- **Conceptual Background of the Domain Problem**

The project will require knowledge and practice in building Graphs /plots and analysing them to get the relationship between dataset, Knowledge of Different Learning Models to build and predict the required output. Basic Data science concepts to increase the quality of the dataset and Python Knowledge (Coding Language) which will be used to solve the complete Flight Price Prediction project. Understanding of calculating R2 score, RMSE, accuracy, skewness and basic mathematics/statistical approaches will help to build an accurate model for this project.

- **Review of Literature**

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

- **Motivation for the Problem Undertaken**

I wanted to solve the real-life problem using the Technical skills gathered during the course of being a Data Analyst and improving the skill set.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem----

Regression Models->

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. For example, relationship between rash driving and number of road accidents by a driver is best studied through regression.

Decision Tree –

It is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs and utility.

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

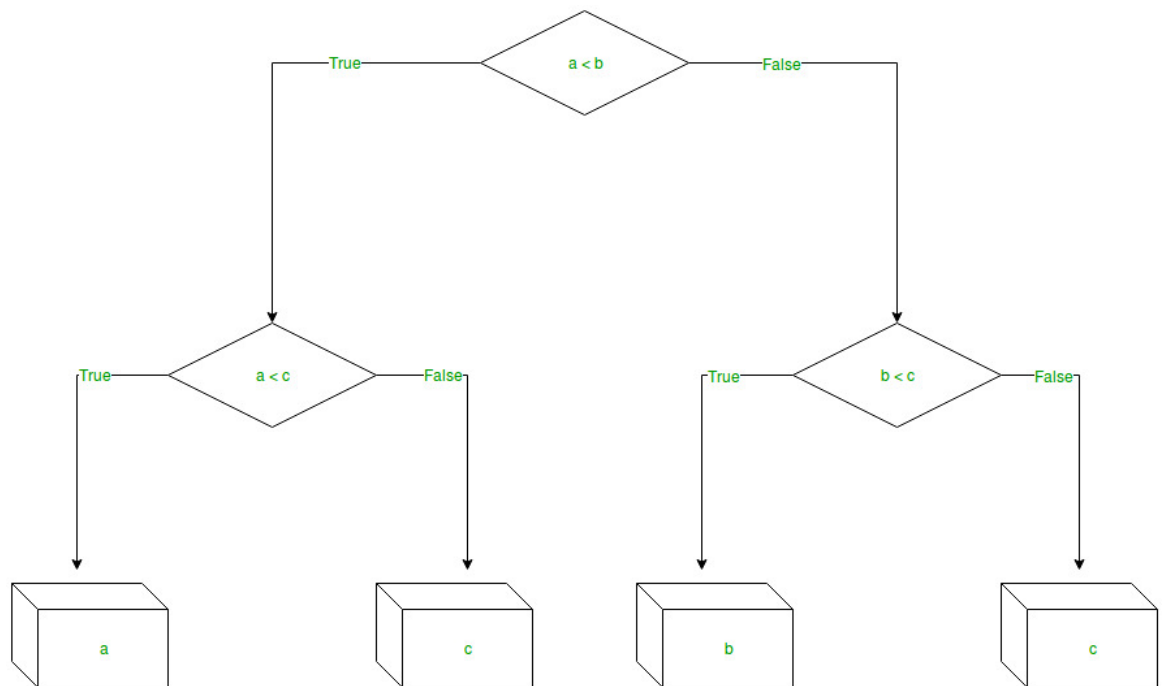
The branches/edges represent the result of the node and the nodes have either:

Conditions [Decision Nodes]

Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and takes makes a decision based on that in the example below which shows a decision tree that evaluates the smallest of three

numbers:



Random Forest –

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.

Naive Bayes –

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Linear Regression –

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

SVM –

Supervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

We used different Plots/ graphs to perform EDA on the dataset->

- 1) Box Plot: It is a type of chart that depicts a group of numerical data through their quartiles. It is a simple way to visualize the shape of our data. It makes comparing characteristics of data between categories very easy.
- 2) Count Plot: IT is kind of like a histogram or a bar graph for some categorical area. It simply shows the number of occurrences of an item based on a certain type of category
- 3) Heat Map: It contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

4) Scatter Plot: A scatter plot is a diagram where each value in the data set is represented by a dot. The Matplotlib module has a method for drawing scatter plots

- Data Sources and their formats

Below are the fields present in our dataset with the information what these fields describe

COLUMN	DATA_TYPE
name	object
departure	object
arrival	object
duration	object
stop	object
price	object
departure_loc	object
arrival_loc	object
doj	object

- Data Pre-processing Done

1) First we checked the data set dimensions (We have 1525 rows and 11 columns)

```
df.shape
```

```
(1525, 11)
```

2) Then we checked whether there is any repeating data available

```
duplicate = df.duplicated()
print(duplicate.sum())
df[duplicate]
```

```
0
```

3) We checked the outliers using the Box Plot and replaced the outliers with more appropriate values. Removal of outliers can also be done but taking the Data Loss percentage into

consideration It is better to replace the outlier

4) We have checked the null values in the dataset.

```
df.isnull().sum()
```

```
Unnamed: 0      0
Unnamed: 0.1    0
name            0
departure       0
arrival         0
duration        0
stop            0
price           0
departure_loc   0
arrival_loc     0
doj             0
.
```

5) We have separated the hours and minutes for arrival, departure and travel time and converted them to integer as shown below:

```
#splitting hours and minutes
df[['departure_hour', 'departure_minute']] = df.departure.str.split(":", expand=True)
```

```
#splitting hours and minutes
df[['arrival_hour', 'arrival_minute']] = df.arrival.str.split(":", expand=True)
```

```
#splitting hours and minutes
df[['duration_hour', 'duration_minute']] = df.duration.str.split(" ", expand=True)
```

```
#converting object type to int
df["departure_hour"] = df["departure_hour"].str.strip()
df["departure_hour"] = df["departure_hour"].astype(str).astype(int)
```

```
#converting object type to int
df["arrival_hour"] = df["arrival_hour"].str.strip()
df["arrival_hour"] = df["arrival_hour"].astype(str).astype(int)
```

```
#converting object type to int
df["duration_hour"] = df["duration_hour"].str.strip('h')
df["duration_hour"] = df["duration_hour"].astype(str).astype(int)
```

```
#converting object type to int
df["arrival_minute"] = df["arrival_minute"].str.strip()
df["arrival_minute"] = df["arrival_minute"].astype(str).astype(int)
```

```
#converting object type to int
df["duration_minute"] = df["duration_minute"].str.strip('m')
df["duration_minute"] = df["duration_minute"].astype(str).astype(int)
```

```
#converting object type to int
df["departure_minute"] = df["departure_minute"].str.strip()
df["departure_minute"] = df["departure_minute"].astype(str).astype(int)
```

6) We have separated the day, date and month from date of journey as shown below:

```
: df["date"] = df["date"].str.strip()  
df["date"] = df["date"].astype(str).astype(int)  
  
: df["day"] = df["day"].str.strip(',')
```

● Hardware and Software Requirements and Tools Used

- 1) Software: Jupyter Notebook - To code and build the project in python
- 2) Libraries:
 - a) numpy - To perform basic math operations
 - b) pandas - To perform basic File operations
 - c) Matplotlib - To plot Different Graphs/ Plots
 - d) Seaborn - Advance library to enhance the quality of graphs/plots
 - e) warnings - To ignore the unwanted warnings raised while interpreting the code
 - f) sklearn - To build the Prediction models
 - g) imblearn - To balance our dataset distribution

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We used different approaches from checking the dataset quality to building the model. We checked the null values and repeated rows in the dataset. For checking the Outliers, we used Box Plot and to remove the outliers we used IQR method. Then we moved to next step of checking data distribution and skewness. To scale the data, we used MinMax Scaler method and to remove the skewness we first checked the log and square root method but skewness of the dataset was not getting removed from it so we performed the Power transform to remove skewness.. We started building different models and checked their R2 score and selected the best suited model to perform Hyper tuning on. We got Linear Regression Algo with the best result and after performing Hyper tuning we finalized the model.

- Testing of Identified Approaches (Algorithms)

- 1) Linear Regression
- 2) Decision Tree
- 3) Elastic Net
- 4) Lasso
- 5) Random Forest
- 6) Ridge

- Run and Evaluate selected models

```
from sklearn import metrics
regr = LinearRegression()
regr.fit(x_train, y_train)
pred=regr.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

R2 score 0.7193417469577079
MAE: 0.40581145415946346
MSE: 0.275342407122605
RMSE: 0.524730794905926

```
rr = Ridge(alpha=0.01)
rr.fit(x_train, y_train)
pred=rr.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

R2 score 0.7193439530583787
MAE: 0.4058082766795796
MSE: 0.27534024280688474
RMSE: 0.5247287325913121

```
model_lasso = Lasso(alpha=0.01)
model_lasso.fit(x_train, y_train)
pred=model_lasso.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

R2 score 0.7221521087335158
MAE: 0.4018958450229042
MSE: 0.2725852753873066
RMSE: 0.5220969980638719

```

model_enet = ElasticNet(alpha = 0.01)
model_enet.fit(x_train, y_train)
pred=model_enet.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))

```

R2 score 0.7208857487896936
MAE: 0.40300046951269486
MSE: 0.2738276496678987
RMSE: 0.5232854380430423

```

from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
pred=dtr.predict(x_test)
print('R2 score', r2_score(y_test, pred))
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))

```

R2 score 0.8150690017212923
MAE: 0.21654402209030146
MSE: 0.18142828748375592
RMSE: 0.4259439957127649

```

from sklearn.ensemble import RandomForestRegressor
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
pred1=rdr.predict(x_test)
print('R2 score',r2_score(y_test, pred1))
print('MAE:', metrics.mean_absolute_error(y_test, pred1))
print('MSE:', metrics.mean_squared_error(y_test, pred1))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,pred1)))

```

R2 score 0.9078191225732629
MAE: 0.18444255639699758
MSE: 0.09043491294562749
RMSE: 0.3007239813277742

- Key Metrics for success in solving problem under consideration

1) Mean Absolute Error(MAE)

MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line.

Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.

so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.

The diagram shows the formula for Mean Absolute Error (MAE) with several annotations. The formula is $MAE = \frac{1}{N} \sum |Y - \hat{Y}|$. An arrow points from the text "Divide by total Number of Data Points" to the fraction $\frac{1}{N}$. Another arrow points from "Actual Output" to the Y term in the absolute value. A third arrow points from "Predicted Output" to the \hat{Y} term. A bracket under the absolute value term $|Y - \hat{Y}|$ is labeled "Absolute Value of residual". An arrow points from "Sum Of" to the summation symbol \sum .

Advantages of MAE

The MAE you get is in the same unit as the output variable.

It is most Robust to outliers.

Disadvantages of MAE

The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

```
from sklearn.metrics import mean_absolute_error  
print("MAE",mean_absolute_error(y_test,y_pred))
```

Now to overcome the disadvantage of MAE next metric came as MSE.

2) Mean Squared Error(MSE)

MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.

So, above we are finding the absolute difference and here we are finding the squared difference.

What actually the MSE represents? It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n} \sum \left(\underbrace{y - \hat{y}}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}} \right)^2$$

Advantages of MSE

The graph of MSE is differentiable, so you can easily use it as a loss function.

Disadvantages of MSE

The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.

If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

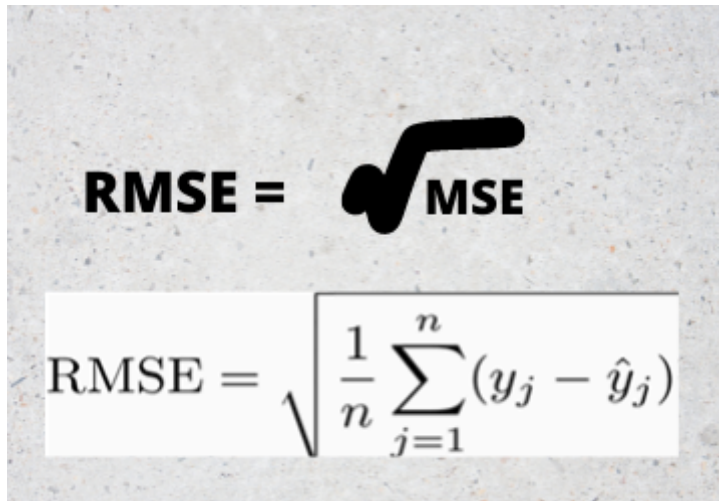
```
from sklearn.metrics import mean_squared_error
```



```
print("MSE",mean_squared_error(y_test,y_pred))
```

3) Root Mean Squared Error(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.



The image shows a hand-drawn diagram on a textured background. At the top, it says **RMSE =** followed by a large, bold, hand-drawn square root symbol, and then **MSE**. Below this, there is a white rectangular box containing the mathematical formula for RMSE:
$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

Advantages of RMSE

The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages of RMSE

It is not that robust to outliers as compared to MAE.

for performing RMSE we have to NumPy NumPy square root function over MSE.

```
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

4) Root Mean Squared Log Error(RMSLE)

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.

To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.

To perform RMSLE we have to use the NumPy log function over RMSE.

```
print("RMSE",np.log(np.sqrt(mean_squared_error(y_test,y_pred)))
)
```

It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions.

5) R Squared (R2)

R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.

So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how much regression line is better than a mean line.

Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$\text{R2 Squared} = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

R2 Squared

Now, how will you interpret the R2 score? suppose If the R2 score is zero then the above regression line by mean line is equal means 1 so 1-1 is zero. So, in this case, both lines are overlapping means model performance is worst, It is not capable to take advantage of the output column.

Now the second case is when the R2 score is 1, it means when the division term is zero and it will happen when the regression line does not make any mistake, it is perfect. In the real world, it is not possible.

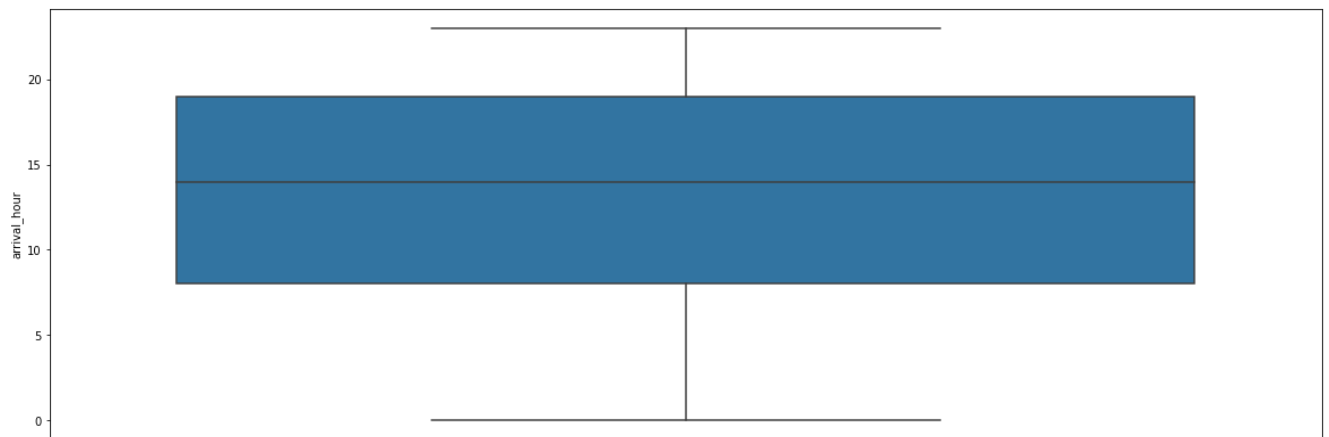
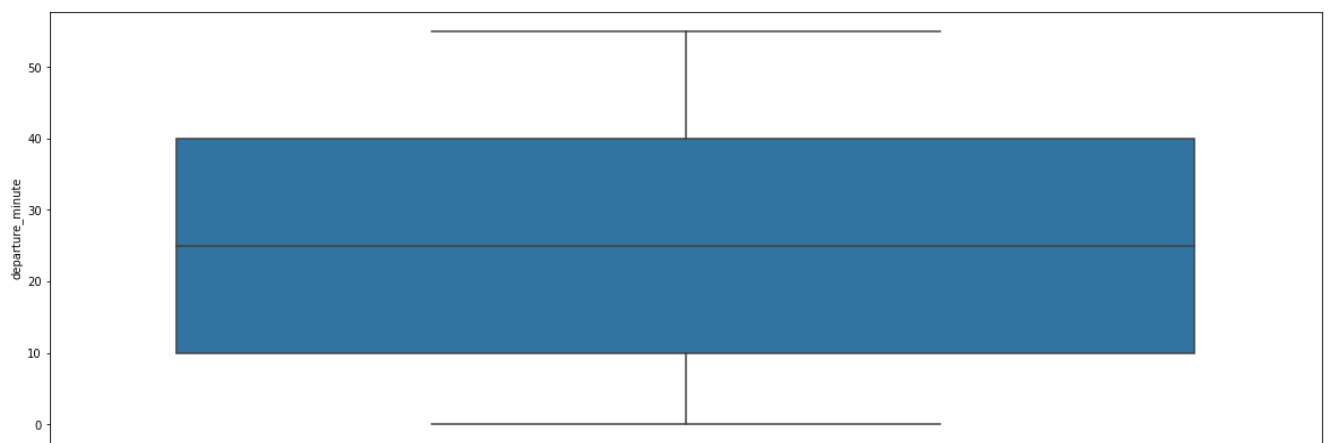
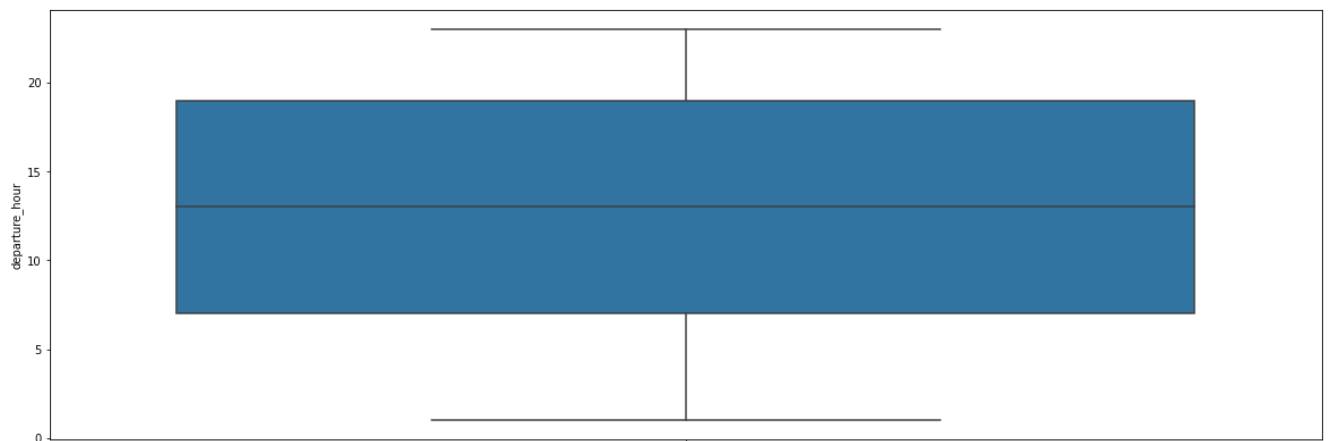
So we can conclude that as our regression line moves towards perfection, R2 score move towards one. And the model performance improves.

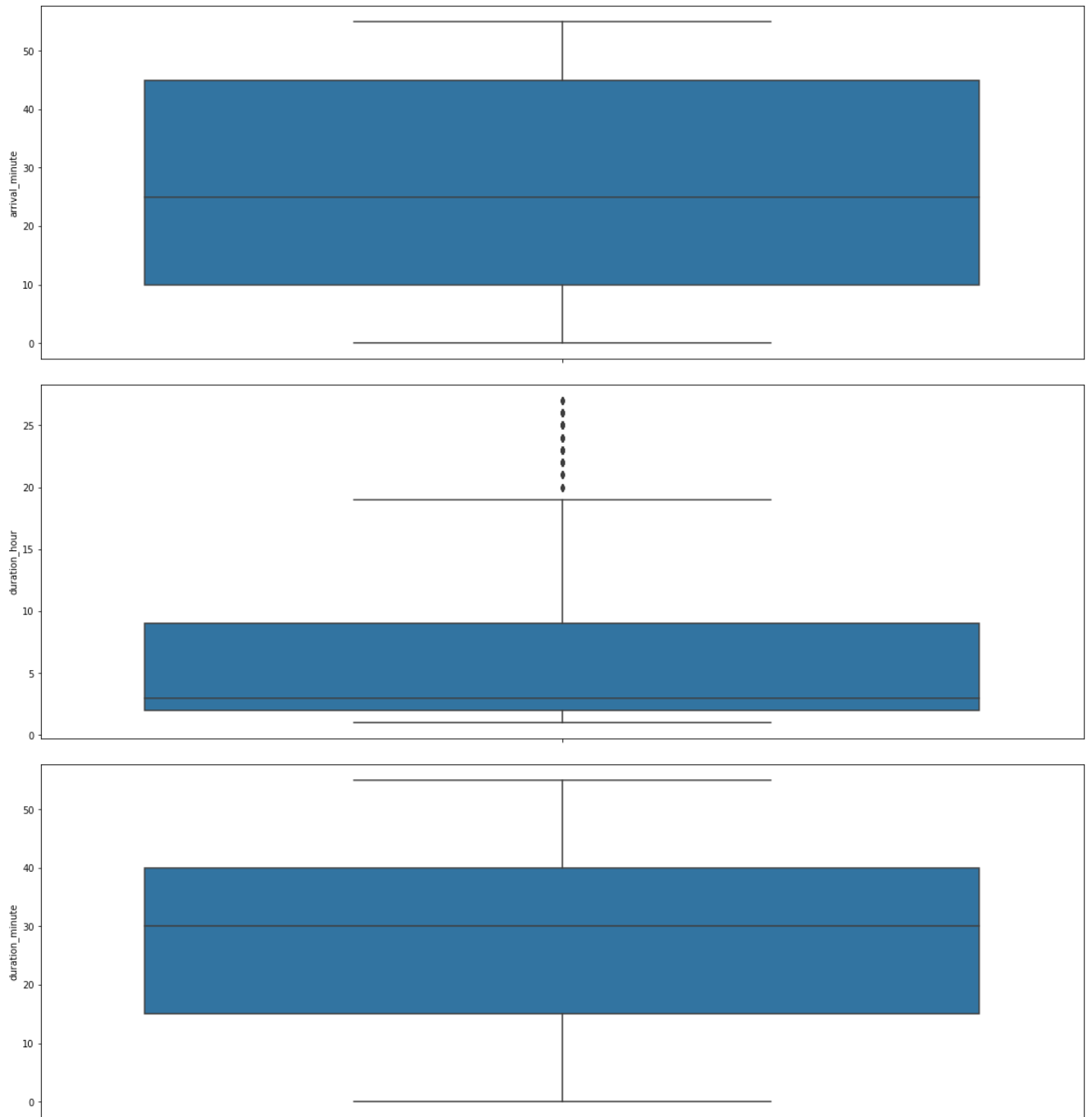
The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable to explain 80 per cent of the variance of data.

```
from sklearn.metrics import r2_score  
r2 = r2_score(y_test,y_pred)  
print(r2)
```

- Visualizations & Analysis

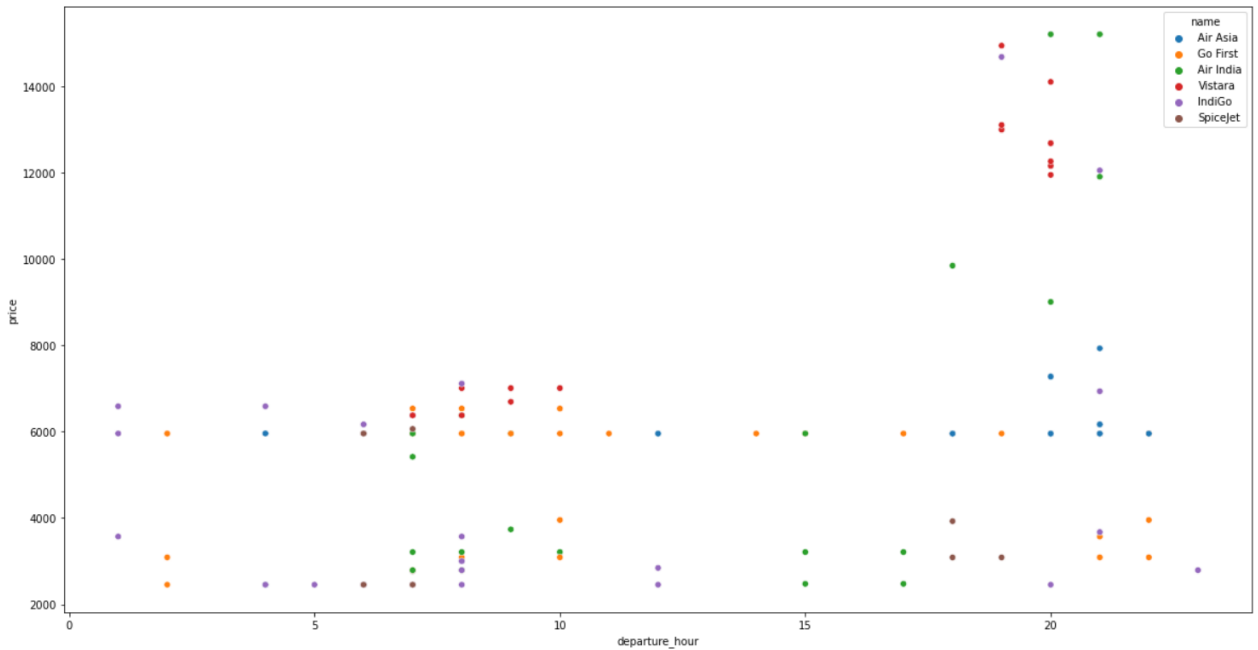
```
continuous=['departure_hour', 'departure_minute', 'arrival_hour', 'arrival_minute',  
            'duration_hour', 'duration_minute']  
counter=1;  
for i in range(0,len(continuous)):  
    plt.figure(figsize=(20,500))  
    plt.subplot(60,1,counter)  
    counter=counter+1  
    sns.boxplot(y=continuous[i],hue = continuous[i],data=df)  
    plt.show()
```



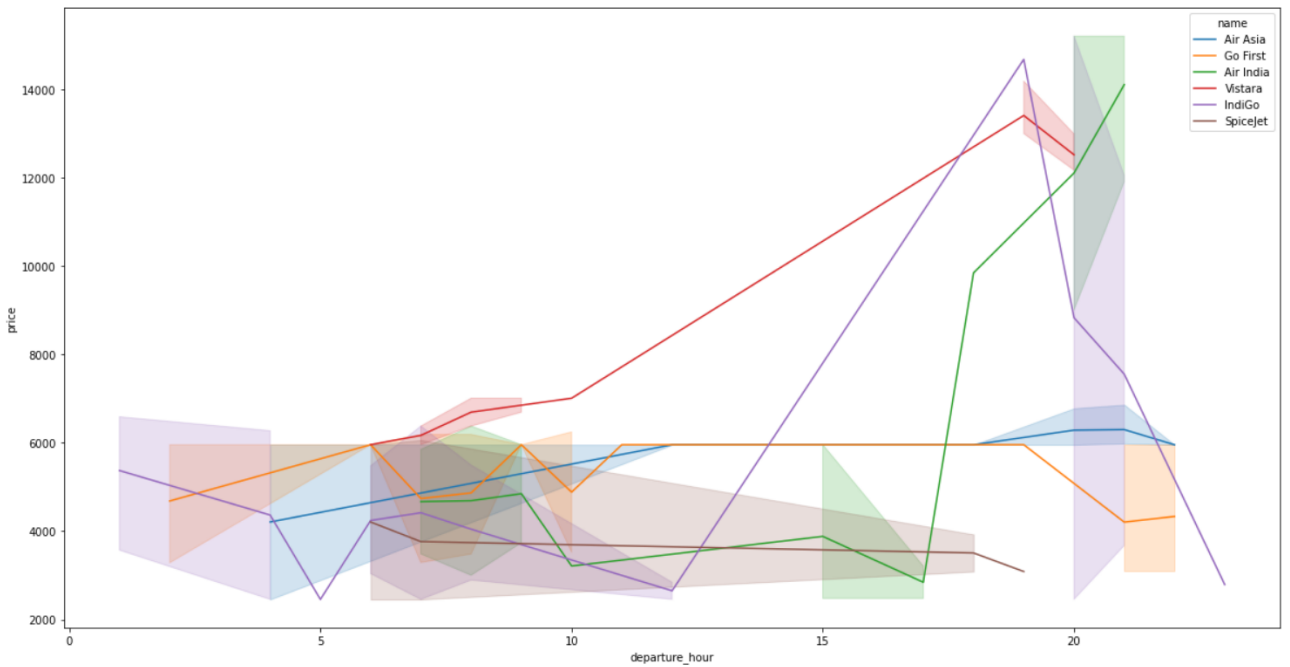


We have outliers in destination_hour column

```
plt.figure(figsize=(20,500))
plt.subplot(40,1,1)
sns.scatterplot(x='departure_hour',y='price',data=df_vis,hue='name')
plt.show()
```

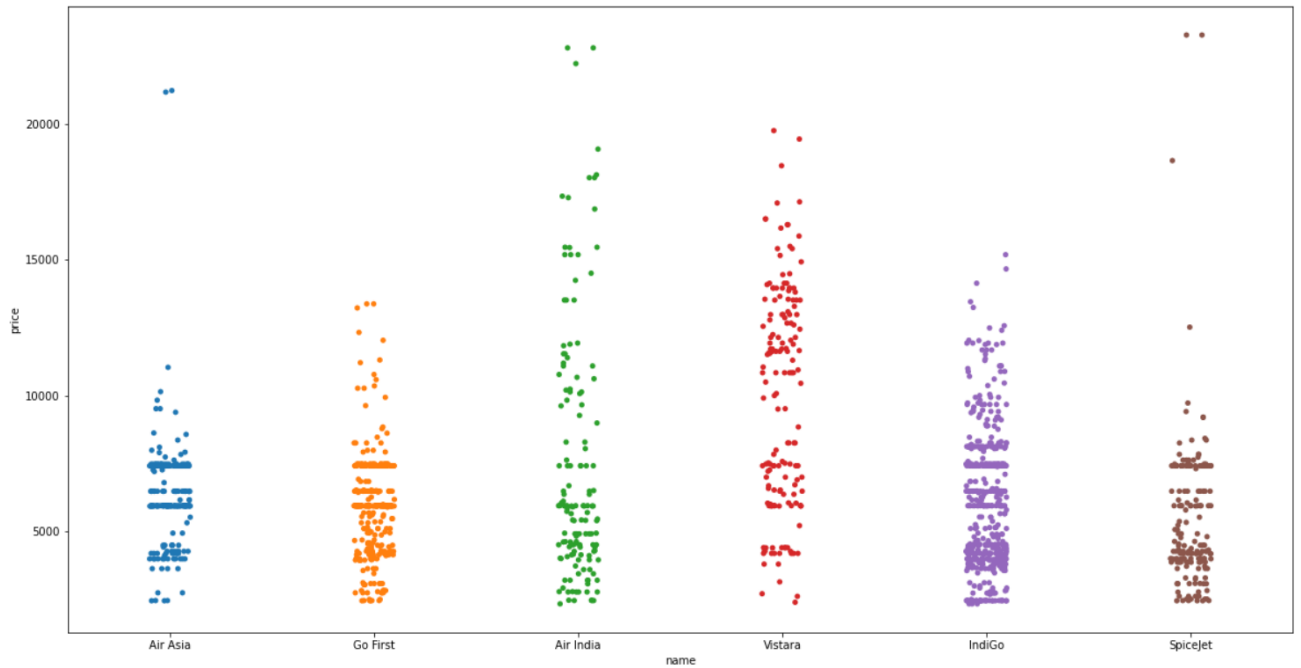


```
plt.figure(figsize=(20,500))
plt.subplot(40,1,1)
sns.lineplot(x='departure_hour',y='price',data=df_vis,hue='name')
plt.show()
```



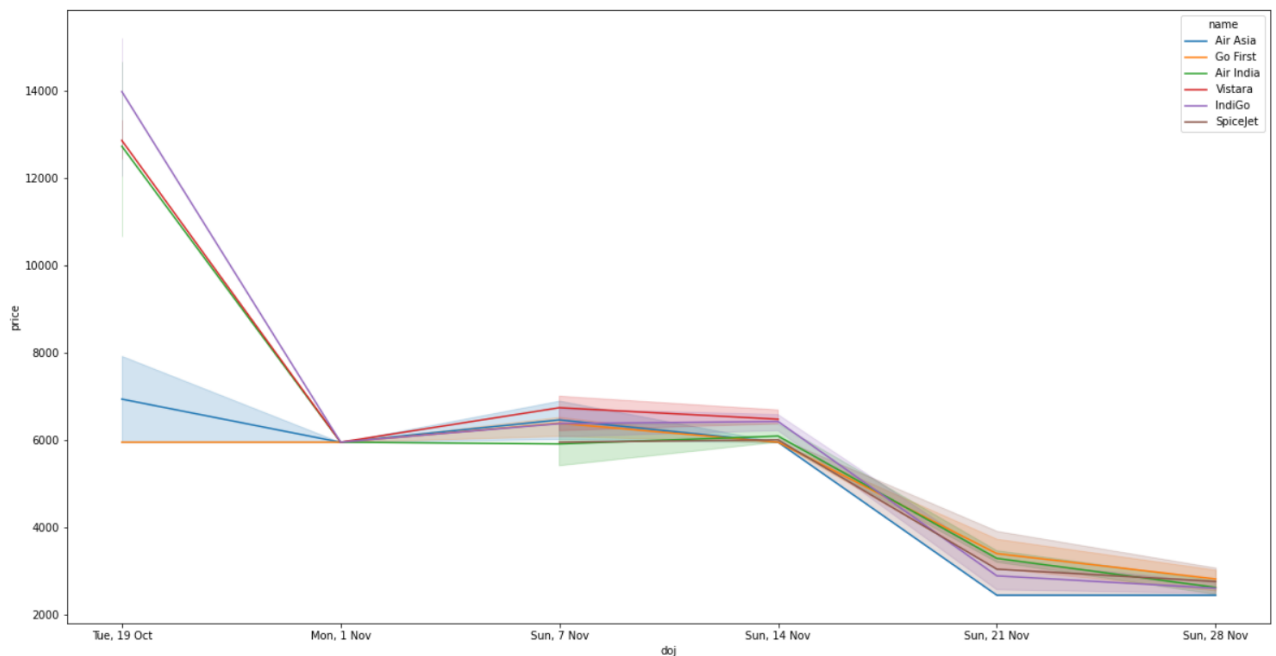
We can observe that early morning flights are cheaper than the late flights

```
plt.figure(figsize=(20,500))
plt.subplot(40,1,1)
sns.stripplot(x='name', y='price', data=df)
plt.show()
```



We can observe that Indigo flight price is comparatively low as compared to AirIndia, Vistara.

```
plt.figure(figsize=(20,500))
plt.subplot(40,1,1)
sns.lineplot(x='doj', y='price', data=df_vis, hue='name')
plt.show()
```



We can observe that the ticket prices are low for flights that are on later dates or next month and price is higher for the immediate flights. Therefore, Best time to book the flight is 1 month before the travel

Interpretation of the Results

Results:

- 1) No null values are present in the dataset
- 2) Dataset have outliers in the variables
- 3) Dataset is not normalized
- 4) Dataset is skewed
- 5) The ticket prices are low for flights that are on later dates or next month and price is higher for the immediate flights. Therefore, Best time to book the flight is 1 month before the travel
- 6) Indigo flight price is comparatively low as compared to AirIndia, Vistara.
- 7) Early morning flights are cheaper than the late flights
- 8) Linear Regression Algorithm is best suited for the current dataset

CONCLUSION

- Key Findings and Conclusions of the Study

We found that to predict the Flight price using Data Science the best way after performing Data Cleaning is to use the Linear Regression Algorithm it provides 70% accuracy which is better than other Regression algorithms.

- Learning Outcomes of the Study in respect of Data Science

In data science, there are various steps involved during Data analysis and cleaning. With the help of various Visualization tools like plots, Graphs we were able to perform the actions and observe different things. Like for finding the outliers we used Box Plot visualization, for finding the skewness and normalization we used Count Plot visualization, for finding skewness we visualized the skewness using Heat Map for the clear picture of how the variables are co-related to each other in the dataset. We used different metrics to check which model best fits the prediction for the dataset.

- Limitations of this work and Scope for Future Work

Data was unbalanced if data was balanced more accurate and clear picture of the output -> result is dependent on the data

Neural network classifier which are still unexplored & can be taken for future consideration