Flight Price Prediction

Name: Shrutayu Aggarwal

Roll no.: B20CS096

➢ Problem Statement

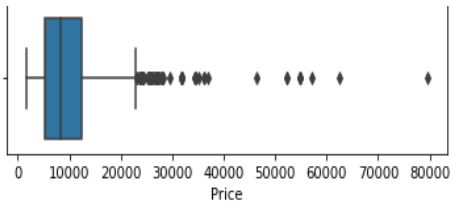
We need to predict The Flight Ticket Price based on this dataset. Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. In this project I have predicted the price of flight tickets.

➢ Dataset Description

This dataset has 11 columns and 10683 rows. The 11 columns contain journey date, time, airline, route, source station, destination station, departure time, arrival time, duration of flight, number of stops which occurs in between and the price column,

➢ Preprocessing of Dataset and constructing the proper dataset

- Dropping rows - There were some null values in rows so these rows were dropped. These rows were only 1.
- Correcting data types – The data types of all the columns were object which was contracting with date and time columns, so these data types were changed to date time.
- Extracting important things from date and time – From date month and date was extracted and similarly from the time, hour and min were extracted and date time columns were dropped.
- Tackling with 'route' column – Each entry in route column was a list of station the flight would be going, so entry with largest route was find out and each station were aligned in a different column named as 'Route$_i$' where i ranges from 1 to 6. This is followed by Label Encoding each route column.
- Getting Dummies- We get the dummies for source, destination, airline, additional_info and drop the current columns. Source column is quite obvious to drop since it will be similar to route1 column.
- Normalizing continuous columns- StandardScaler () is used to normalize the continuous columns. So, now we will be having 43 columns after processing all the above steps and 10682 rows.
- Removing outlier – Presence of any outlier is checked and the obtained distribution looks like



So, all the outliers having price >= 40000 were removed and these values were filled with the median

- Train Test Split- The above formed dataset is splitted into three parts i.e., train, validation, test in the ratio 70:10:20.

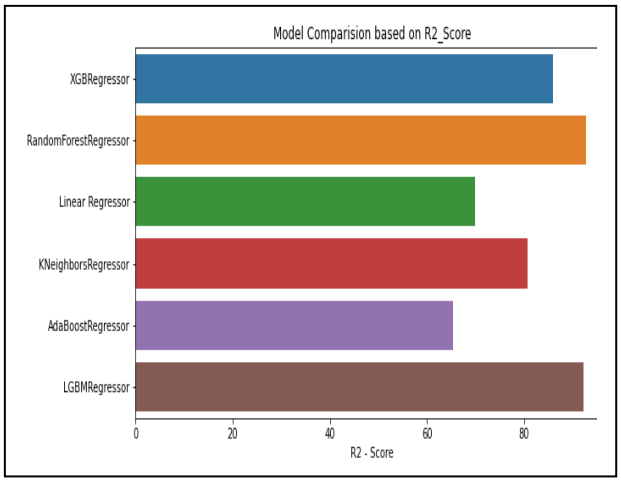
➢ Metric Selection

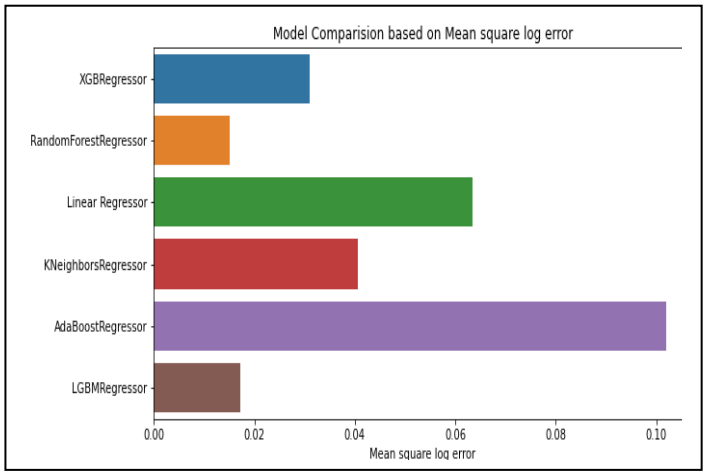
We will be using R2-Score and Mean squared log error (MSLE) are the evaluation metric as it fits the best here since it is a regressive problem statement where our X data is normalized but the Y needs to be predicted.
For the 5-fold cross validation same metric (R2-Score) is used to get the cross-validation score and the mean of all the cross-validation score is reported as the final cross validation score.

➢ Applying Regressive Models to check which model works how

A total of 6 models were applied to check which model works how much. These models consist of XGBRegressor, RandomForestRegressor, Linear Regressor, KNeighborsRegressor, AdaBoostRegressor, LGBMRegressor.
These models were compared based upon the training score, testing score, cross-validation score, R2 score and MSLE and the following result has been caught out.

Model	Training score	Testing Score	R2 - Score	Cross_val_scores	Mean square log error
XGBRegressor	0.850407	0.859584	85.958376	0.833664	0.031097
RandomForestRegressor	0.983705	0.927178	92.717821	0.900688	0.015056
Linear Regressor	0.687144	0.699428	69.942783	0.680370	0.063380
KNeighborsRegressor	0.859190	0.807498	80.749839	0.768938	0.040525
AdaBoostRegressor	0.636625	0.652969	65.296860	0.570653	0.101954
LGBMRegressor	0.935200	0.922776	92.277581	0.894059	0.017273





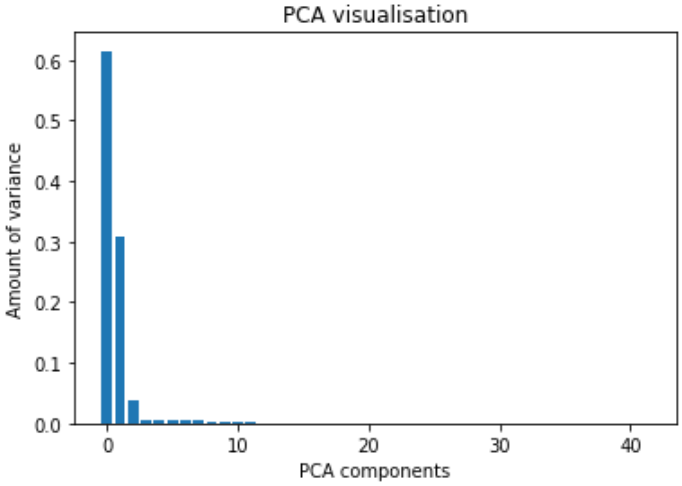
From the above three results that our XGBRegressor, LGBMRegressor, and RandomForestRegressor work finest. So now all these three models are tuned to get the maximum R2 score and least MSLE. For tuning these models RandomSearchCV is applied with cross validation = 5 for each model. So, the result after hyper tuning each of them the received are

Model	R2-score	MSLE
RandomForestRegressor	88.058049	0.022243
LightGBM	92.701052	0.015446
XGBRegressor	93.626330	0.013747

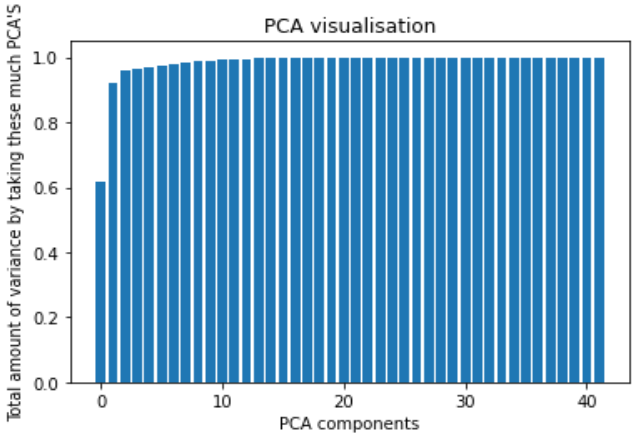
So, all these 3 models are good to get the prediction, but here we are analyzing 43 columns and many of them may be strongly correlated. So, I have tried to reduce the dimensions of the dataset and then tried to check on these 3 models. So, PCA is applied to reduce the dimension.

- ## Applying PCA to reduce the dimension and applying checking on these 3 models.

PCA is applied over dataset (10682 X 43) and corresponding PCA components are visualized. We got the below plot for the PCA components.



As we can see that most variance is captured by first two principal components but we need to find the exact value of variance which is captured, which on exploring I got that it was only 96% variance which was captured by first two principal components. So, I took starting 10 principal components which captures about 99% of variance.



From the left plot we can clearly see that after PCA = 10 the variance is approximately constant, so we prepared the dataset having these 10 PCA'S and the price column which will act as out new dataset. And now the we check our new dataset with the above 3 models.

These 3 models are tuned using RandomSearchCV and done cross validation using scoring as R2-score and we are able to receive the result as:

Model	R2-Score	MLSE
RandomForestRegressor	87.906624	0.022873
LightGBM	91.682792	0.016453
XGBRegressor	92.384028	0.013637

We can clearly see that our RandomForestRegressor didn't worked well with PCA but the other two models worked very well on the testing data. So, we will be rejecting RandomForestRegressor.

Although we are getting somewhat less R2-score when PCA is not applied but the difference is not very large and it's pretty easy and convenient to work with the small dataset.

- ## Justification why XGB and LightGBM performed well than Random Forest

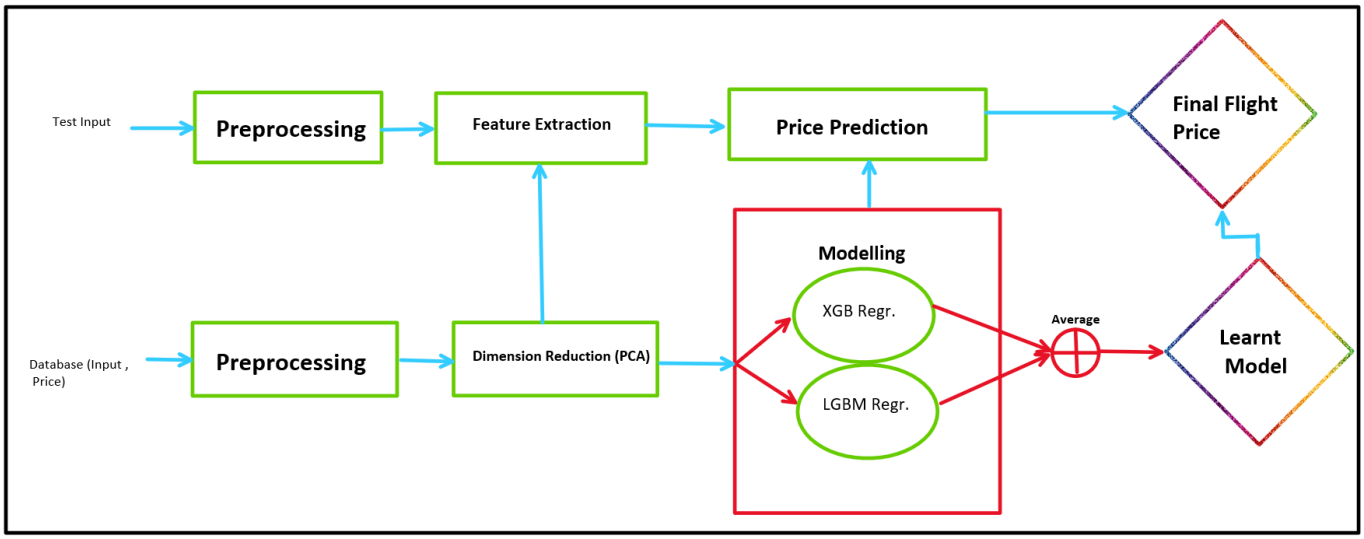
XGBoost always gives more importance to functional space when reducing the cost of a model while Random Forest tries to give more preferences to hyperparameters to optimize the model. A small change in the hyperparameter will affect almost all trees in the forest which can alter the prediction. Also, this is not a good approach when we

expect test data with so many variations in real-time with a pre-defined mindset of hyperparameters for the whole forest but XG boost hyperparameters are applied to only one tree at the beginning which is expected to adjust itself in an efficient manner when iterations progress. Also, the XGBoost needs only a very low number of initial hyperparameters (shrinkage parameter, depth of the tree, number of trees) when compared with the Random Forest. Also, XGBoost is a good option for unbalanced datasets but we cannot trust random forest in these types of cases.
In contrast to the level-wise (horizontal) growth in XGBoost, LightGBM carries out leaf-wise (vertical) growth that results in more loss reduction and in turn higher accuracy while being faster.

- Final estimation of price

  So, we predicting the final price by taking the average of the price of what predicted/ estimated by XGBRegressor and LGBMRegressor. And this would be out final price predicted for the given flight.

- END – TO - END PIPELINE FOR THE PROJECT WOULD BE



As mentioned in the above pipeline, we would be extracting the data, do the preprocessing over it as mentioned above. Apply PCA over it and would take first 10 principal components. Do modelling on the new dataset formed using XGBRegressor and LGBMRegressor and would report the final answer as the average of the two estimated price from the two corresponding models.

For the given dataset if we pass the dataset throw this pipeline the R2-score which we will be receiving is 92.2471 %.

GitHub Repository link for the project