EPL448 - Data Mining on the Web

Semester Project – Spring Semester 2023

Predicting the Flight Ticket Price

Panagiotis Christodoulou - 1013138
Emmanouil Theofilou – AN830433
Demetris Vereis - 1022053

Competition Link

Initial Description

The purpose of our project is to predict the price of a flight given a large dataset that contains information about various flights in India for the year 2019. The initial dataset consists of 11 columns and 10683 records. It has information about :

- the name of the airline
- the date of the trip
- the departure
- arrival town
- the route of the trip
- time of departure
- arrival time
- the total time of the flights, including the waiting time in the between of the flights.
- Additional Information
- Price, which we want to predict.

The features can be seen in ***Figure 1*** .

	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
...	...	...	...	...	...	...	...	...	...	...	...
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

Figure 1 – Dataset

To get some statistical information about the price, the pandas *describe()* function was used, and is displayed in ***Figure 2*** .

	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

Figure 2 – Price Fluctuation

Exploratory Data Analysis

The dataset in its original form contains multiple columns with categorical values (e.g., Airline name, Source city, Destination city). Since these values are not numerical, we could not investigate the continuous distribution that they follow or distinguish their separability easily through plots.

For this reason, we opted to examine the distribution of these categorical values and their impact on the final price of the plane ticket using count plots and box plots.

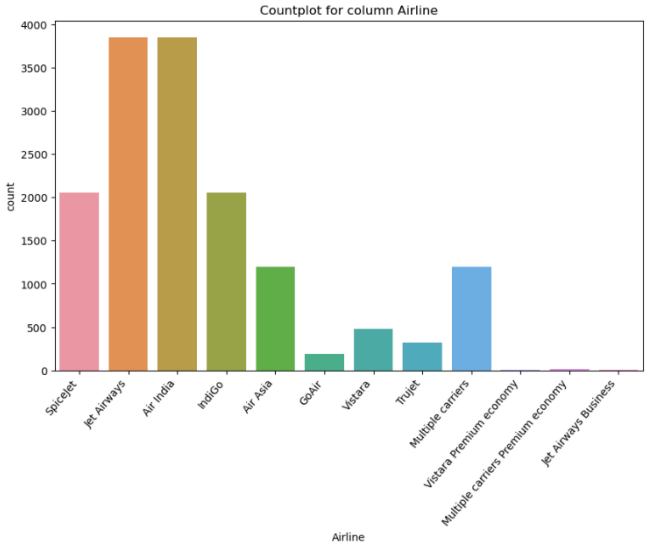

Figure 3 - Count plot for column 'Airline'

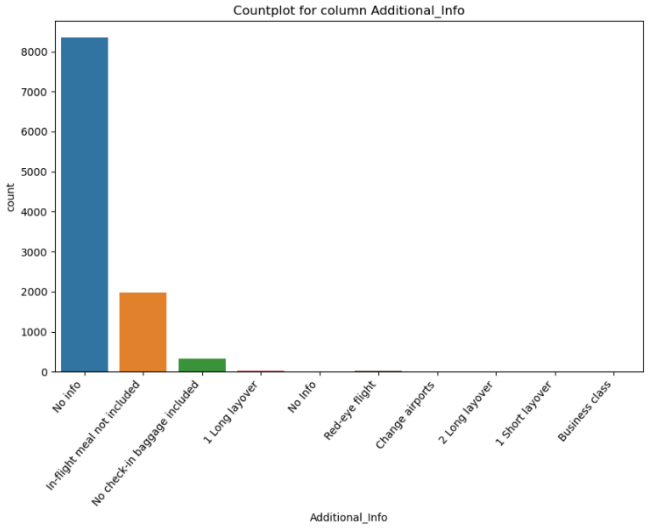

Figure 4 - Count plot for column 'Additional_Info'

The most notable count plots are shown in ***Figure 3*** and ***Figure 4*** . From these two count plots we can see that the most popular airlines by far are Jet Airways and Air India, however, there are enough occurrences of other airways as to not pose a problem for the regression models further down the line.

On the contrary, the count plot for the column *Additional_Info* displays some imbalances that are present in the dataset since most columns contain the categorical value 'No info'. This hints that this column might **not** prove too **useful** for the regression models, as there is not enough variety in its values.

Other categorical columns that were plotted include *Source* and *Destination*. The count plots for these columns show that while there are popular and less popular source and destination cities, there are enough occurrences of all cities, and the column can be used (after pre-processing and transformation to numerical values) for the prediction of the plane ticket price.

As described, we also wanted to examine the impact of these categorical values on the plane ticket

price themselves. This was done with the use of box plots.

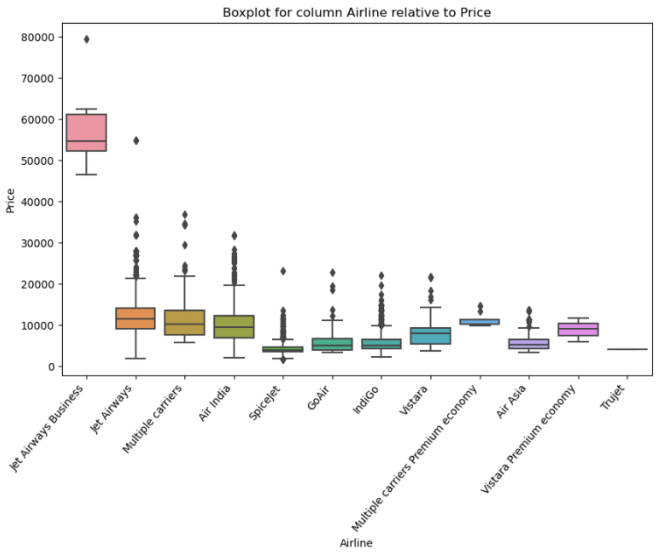

Figure 5 - Box plot for column Airline relative to Price

Airline	count	mean	std	min	25%	50%	75%	
Jet Airways Business	6.0	58358.666667	11667.596748	46490.0	52243.0	54747.0	61122.50	7
Jet Airways	3849.0	11643.923357	4258.940578	1840.0	9134.0	11467.0	14151.00	5
Multiple carriers Premium economy	13.0	11418.846154	1717.153936	9845.0	10161.0	11269.0	11269.00	1
Multiple carriers	1196.0	10902.678094	3721.234997	5797.0	7723.0	10197.0	13587.00	3
Air India	1752.0	9611.210616	3900.952942	2050.0	6896.0	9443.0	12219.00	3
Vistara Premium economy	3.0	8962.333333	2915.405518	5969.0	7547.0	9125.0	10459.00	1
Vistara	479.0	7796.348643	2914.298578	3687.0	5403.0	7980.0	9345.00	2
GoAir	194.0	5861.056701	2703.585767	3398.0	3898.0	5135.0	6811.25	2
IndiGo	2053.0	5673.682903	2264.142168	2227.0	4226.0	5000.0	6494.00	2
Air Asia	319.0	5590.260188	2027.362290	3383.0	4282.0	5162.0	6451.00	1
SpiceJet	818.0	4338.284841	1849.922514	1759.0	3574.5	3873.0	4760.00	2
Trujet	1.0	4140.000000	NaN	4140.0	4140.0	4140.0	4140.00	

Figure 6 – Properties of prices per Airline (pandas.describe())

The above figures, ***Figure 5*** and ***Figure 6*** , show that the *Airline* column plays a significant role in the resulting price of the ticket. Jet Airways Business, which also happens to be one of the least popular airlines (in count), has the highest mean price of a ticket, but even besides that, there are significant differences between tickets of various airlines. This is an indicator that this column is necessary for the performance of our regression model.

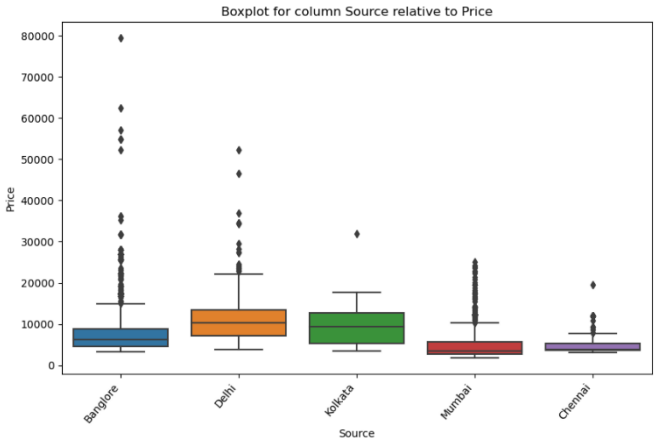

Figure 7 - Box plot for column Source Relative to Price

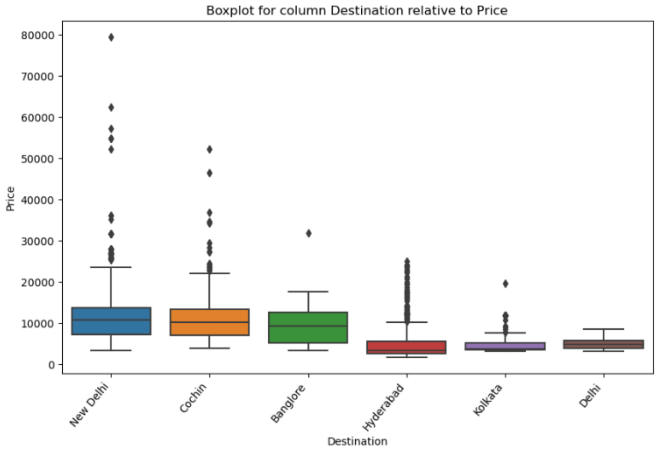

Figure 8 - Box plot for column Destination relative to Price

Since the count plots, ***Figure 7*** and ***Figure 8*** showed that the distribution of values for source and destination cities are quite even, we decided to explore the resulting prices of tickets based on source and destination cities. The above figures show that there are no significant differences between ticket prices where the source and/or destination cities differ, and this shows that those two columns **might not prove as critical** to our prediction later.

Preprocessing

First, the *Date_of_Journey* and *Dep_Time* columns were merged to create the *dep_date_time* column which was converted to the **datetime** object for easier processing. Then, the column *Duration* was split into hours and minutes, *duration_hours* and *duration_minutes* respectively. A column *total_duration_minutes* was also created that represents the total flight duration in minutes. 'None' was added to the columns of the records that did not write the duration. These two features were converted into **timedelta** objects for easier comparisons.

After these, using data from statisticstimes.com, the population of the cities that were present as source and destination were added as we believed that it could improve our predictive model's performance. (Auxiliary_Data/indian_cities_population.xlsx)

One Hot Encoding was performed on the *Airline* column. Each different airline was encoded as a column where 1 means that this airline was used in the route and 0 that it did not. It should be noted that there are categorical values that belong to the same airline such as 'JetAirways' and 'JetAirwaysBusiness', however they were treated as different airlines because during visualization we noticed that the plane tickets prices varied across these airlines of the same family. The total number of airlines was 12.

For the column *Total_Stops*, its content was coded into the integer number of the stops, e.g., "1 stop" was converted into 1.

Following that, the *Additional_Info* column was encoded with **One Hot Encoding** like the *Airline* column. We noticed some values that were different, but with the same meaning as "No info" and "No Info". These columns were adjusted to contain the same values.

The combination of day and month was encoded into an integer number from 0-365 to represent the exact day of the year and maintain their cyclical properties using **Cyclical Transformation**.

The exact day (Monday, Tuesday, etc.) was computed from the calendar date of the departure and arrival day. The reason behind this is that some specific days like Friday could have great importance in the price of a ticket. Days were numbered from 0 (Monday), to 6 (Sunday), and later were encoded using **Cyclical Transformation**.

We added columns that represented the source and destination airport classes (small, medium, large) and encoded these into integers, 0,1 and 2 respectively.

To get the intermediate flights the Python method *split()* was used on the column *Route*. The *Route* column originally contained the airport codes that each flight visited. We then found the geographic coordinates of each airport present in the route column (Auxiliary_Data/indian_airports_locations.xlsx).
Using these, we found out the total distance covered for each route.

Also, using the *Route* column we performed **One Hot Encoding** on the different airport codes that all flights in the dataset passed through, creating separate columns for each airport code present in the dataset.

A lot of intermediate, columns were dropped as they were only used for further preprocessing purposes and are not intended for final predictive use.

Feature Importance and Correlation Matrix

Following preprocessing, where a final form of the dataset was reached, the importance of all resulting features (columns) was examined. This was done with the use of an estimator for feature importances. The model used for estimating feature importances was an *ExtraTreesRegressor(n_estimators = 100, max_features = 13, random_state = 0)* from Python's **sklearn** library.

Feature importance was calculated for both standardized and non-standardized datasets (as standardization was used for models which work with distances in the vector space e.g., Support Vector Machine Regression, Linear Regression). Due to the large number of columns in the dataset as a result of preprocessing, the plots for feature importance are not very readable, but they are included in the submission of the project and can be found in the directory **vis/**.

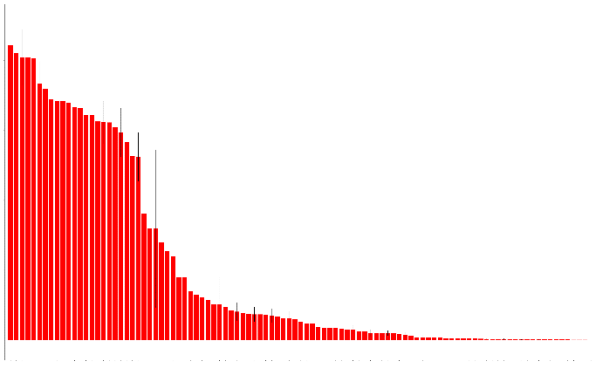

Figure 9 - Feature Importance plot for non-standardized dataset

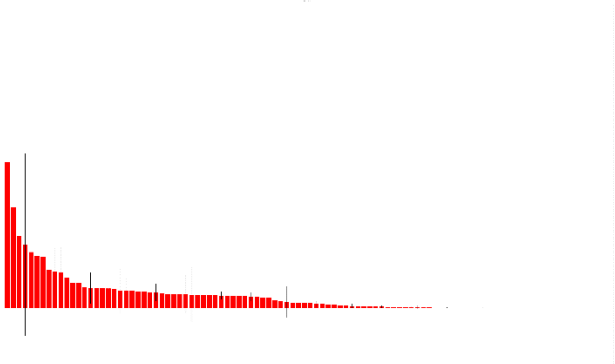

Figure 10 - Feature Importance plot for standardized data

As there are many features for each record in this dataset, the overall values for importance for each feature are understandably low as displayed in ***Figure 9*** and ***Figure 10*** , with the highest importance value for standardized data being around 0.11. Amongst features that rank for the highest importance are *Airline_Jet_Airways* (Whether or not the flight is with Jet Airways – one of the most expensive airways as can be seen in our visualization), *num_stops*, *flight_duration_minutes and distance covered_km*.

We proceeded to discover correlations between columns. High correlations (positive or negative) indicate that columns behave in similar ways, and possibly, by keeping a subset of such columns to train our model with, a satisfactory performance can still be achieved. The correlation matrix is very large due to the number of features and can be found in the source code for our project.

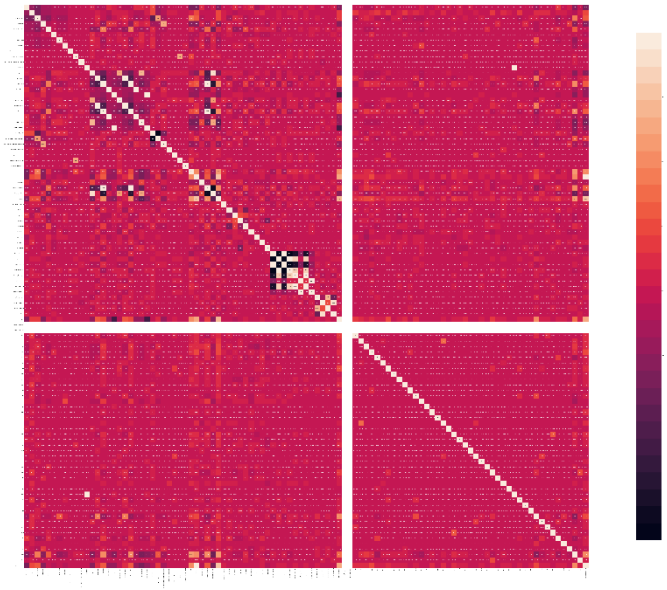

Figure 11- Correlation matrix for dataset features post preprocessing

To investigate the correlation matrix, ***Figure 11***, more comprehensively, we printed a list of features with an absolute correlation value of over an arbitrary threshold of 0.40. From this process, we discovered that 'Additional_Info' columns in general did not have a significant correlation with the price of the plane ticket, and in many cases were correlated with specific airlines (e.g., *Additional_Info_Business* class positively correlated with *Jet Airways Business*). This, coupled with the fact that the distribution of categories in the Additional Info columns in the original dataset was dominated by one category, led to the **removal** of the *Additional_Info* columns from the final version of the dataset.

Other findings from the correlation matrix include a positive correlation between distance covered and flight duration, as well as number of stops. This makes logical sense, as layover flights tend to be longer, and longer flights cover more distance.

The examination of feature importance and the correlation matrix was not used to remove a lot of features, but only remove features that we considered unnecessary, and features that we believed would not prove useful to our model and only increase the time spent to train it. Feature selection was performed at a later stage, where the performance of the model with different features was evaluated with multiple regression algorithms.

Machine Learning Algorithm Selection

As we know there are a lot of different machine learning algorithms to use for regression problems. The most usual machine learning algorithms are Linear Regression and Polynomial Regression, none of which could be used for our problem since our dataset contained mostly categorical data. To use Polynomial Regression the target value must follow the normal distribution so a curve can fit it but as can be seen in ***Figure 12*** , the Price column does not follow the normal distribution and its p-value is zero.

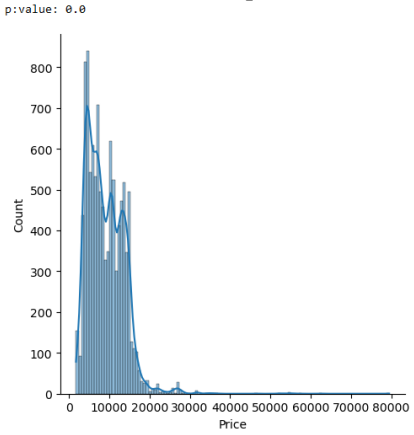

Figure 12 – Price Distribution

Also, while the features do not have any correlation between them, as showed in ***Figure 13*** , there is no linear relation between the features and the target value. Firstly, most of the features are zeros and ones, so there is no linear relation between them and the price. The other features that are not categorical, such as city's population and distance covered, do not have any linear relation to the price as shown in ***Figure 13*** .

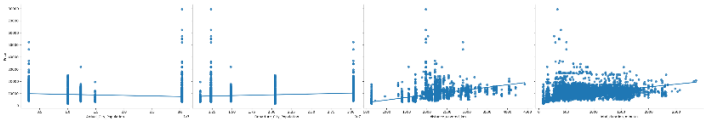

Figure 13 – Features and Price Relation

Furthermore, as shown in ***Figure 14*** , the features do not follow normal distribution which is another reason that Linear and Polynomial Regressors cannot be used.

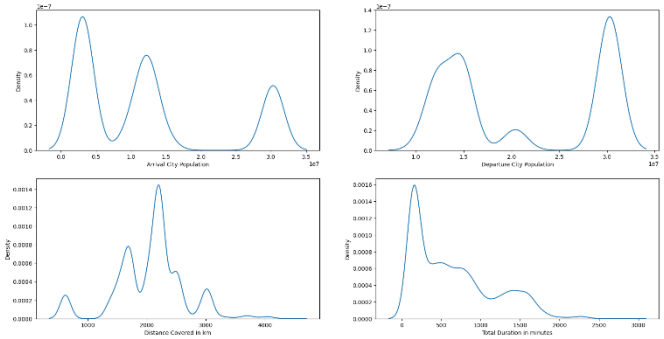

Figure 14 – Features Distribution

In summary, it was decided to run more algorithms from the **Ensemble** family like Decision Tree, Random Forest, AdaBoost, Gradient Boosting and some **other** types of machine learning algorithms like KNN and SVR. SVR was selected despite being similar in a way to Polynomial Regression, as it creates hyperplanes to fit the data, and we predicted that it will not perform well.

Executing the Algorithms

In this part, the research was separated in three different parts. Sequential Forward Selection was run, then we used a Grid for hyper-parameter tuning and finally the best algorithm with the best parameters was run again to verify the results.

The **r2-score** scoring metric was used as it is suitable for regression problems.

Sequential Forward Selection

Sequential Forward Selection was the chosen method to pick the best combination of features, as there are so many features and, as Figure 9 shows, less than half of them are important, so it is easier to add a feature instead of deleting one with the Backward elimination. The machine learning algorithms used in sequential Forward selection were KNN, SVR, Decision Tree, Random Forest, AdaBoost and Gradient Boosting. A lot of different executions were made for each model as can be seen in the submission of the project. Cross-validation was used in this method to be sure that the results are trustworthy. SVR achieved the worst performance at about 0.30 peak r2-score, which is in line with our hypothesis. KNN produced moderate results as r2-score is around 0.80, as shown in ***Figure 15*** **.**

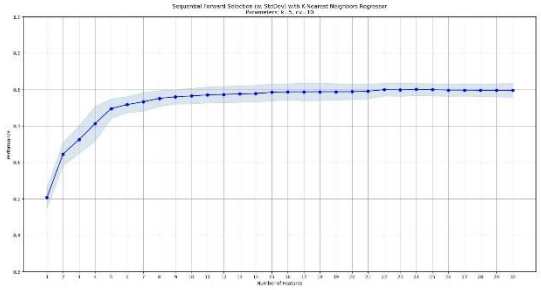

Figure 15 – KNN Sequential Forward Selection

Decision Tree and Random Forest produced very similar results at a little more than 0.80 of r2-score with only 15 features as shown in ***Figure 16*** .

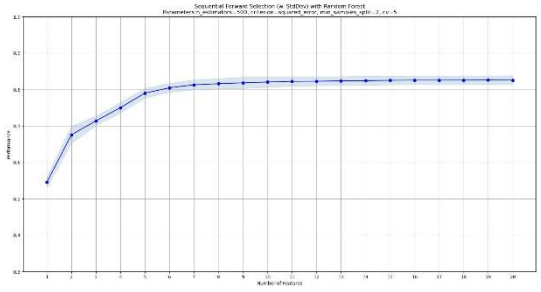

Figure 16 – Random Forest Sequential Forward Selection

AdaBoost regression was worse than the previous algorithms, with an r2-score of under 0.60. Finally, the Gradient Boosting algorithm was the **best**, with 30 features achieving an r2-score of 0.85 as displayed in ***Figure 17*** .

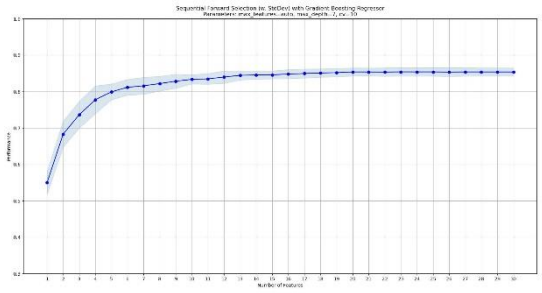

Figure 17 – Gradient Boosting Sequential Forward Selection

The selected final features were the 30 best features of the Sequential Forward Selection using Gradient Boosting as shown below, ***Figure 18*** . The preprocessing was successful as the features we added from others dataset like cities population and distance covered are included in the best features. Also, the Cyclical Transformations were important as many features that went through that are included.

```
('0' Airline_IndiGo,
 '1' Airline_Air India,
 '2' Airline_Jet Airways,
 '4' Airline_Multiple carriers,
 '6' Airline_Vistara,
 '9' Airline_Jet Airways Business,
 '13' Source_Kolkata ,
 '16' Source_Mumbai,
 '17' Destination_New Delhi,
 '23' Departure_City_Population,
 '24' Arrival_City_Population,
 '25' num_stops,
 '26' dep_hour_sin,
 '31' dep_minute_cos,
 '33' arr_minute_cos,
 '38' dep_day_of_month_sin,
 '39' dep_day_of_month_cos,
 '40' arr_day_of_month_sin,
 '42' dep_day_of_week_sin,
 '43' dep_day_of_week_cos,
 '44' arr_day_of_week_sin,
 '45' arr_day_of_week_cos,
 '46' distance_covered_km,
 '53' DED,
 '55' DEL,
 '56' ISK,
 '57' NDC,
 '76' RPR,
 '90' STV,
 '91' total_duration_minutes
 )}}
```

Figure 18 – Best Features Scaled Data

For the rest of the study, the performance metric used in the Machine Hack challenge was defined and used along with the r2-score. The used metric is **RMSLE** (*Root Mean Square Logarithmic Error*) and is equal to:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1) - \log(y_i+1))^2}$$

Due to the lacklustre performance of SVR and AdaBoost, the decision to replace them with Extra Trees and Extreme Gradient Boosting was made.

Exhaustive Hyperparameter Tuning

As mentioned before, the machine learning algorithms used are Gradient Boosting, Extreme Gradient Boosting, KNN, Decision Tree, Random Forest, Extra Trees. In this section, many different

combinations of parameters are used to discover the best algorithm and its parameters that has the highest r2 and RMSLE score. SKLearn's *GridSearchCV* was used for this purpose. The r2 and RMSLE score were calculated on the testing data to be sure that there is no overfitting phenomenon.

The best algorithm is **Gradient Boosting** which **produced 0.923** in the RMSLE metric as it can be seen in ***Figure 18*** . The **best** parameters are a learning rate of 0.1, max depth of seven, max features set to auto, n_estimators to 100 and subsample to 1.0.

```
R2-Score:0.8045531099699903
RMSLE (Machine-Hack metric):0.9234161702009416
Gradient Boosting Regression score: 0.8391842024510539 - best params: {'learning_rate': 0.1, 'max_depth': 7, 'max_featu
uto', 'n_estimators': 100, 'subsample': 1.0}
```

Figure 19 – Gradient Boosting Results.

In this part, should be noted that the 0.923 RMSLE score with the training set, ranks in the *first 300* participants in the Machine Hack contest and at only **0.030 difference** from the **first 20**.

After this, the Gradient Boosting algorithm was run again with a **K-fold Cross Validation** of 10 to ensure that the performance metrics are reliable, and do not depend on the randomness of the dataset split, to training and testing set. The results were the same.

Experiments with Unscaled Data

For the last part of the study, some experiments with unscaled data took place to figure out if even better results can be yielded. The motivation behind attempting to evaluating the performance of the regression algorithms using unscaled data is because the chosen algorithms predict using trees, and not distances, therefore we hypothesized that unscaled data would not cause a significant difference in the performance of the algorithms. The methodology follows the one used with the scaled dataset. First the Sequential Forward selection was made using the best algorithm so far, Gradient Boosting. Then, the best features were selected as shown in ***Figure 20*** .

```
'Airline_IndiGo',
   'Airline_Air India',
   'Airline_Jet Airways',
   'Airline_Multiple carriers',
   'Airline_Vistara',
   'Airline_Jet Airways Business',
   'Airline_Multiple carriers Premium economy',
   'Source_Banglore',
   'Destination_Cochin',
   'Destination_Kolkata',
   'Departure_City_Population',
   'Arrival_City_Population',
   'dep_hour_cos',
   'dep_minute_sin',
   'dep_minute_cos',
   'arr_minute_cos',
   'dep_day_of_month_sin',
   'arr_day_of_month_sin',
   'dep_day_of_week_sin',
   'dep_day_of_week_cos',
   'arr_day_of_week_sin',
   'arr_day_of_week_cos',
   'distance_covered_km',
   'departure_airport_class',
   'arrival_airport_class',
   'HYD',
   'JLR',
   'VGA',
   'PNQ',
   'COK',
   'STV',
   'RPR',
   'IXB',
   'NDC',
   'BLR',
   'total_duration_minutes')}
"""
```

Figure 20 – Best features unscaled data

The hyperparameter tuning was performed for Gradient Boosting, Extreme Gradient Boosting, and Random Forest models as these had the best results so far.

The best model for the unscaled data is Extreme Gradient Boosting, but the RMSLE and r2-score is a little worse than the scaled data, by around 0.002, at 0.921. So, there is not an improvement in the use of unscaled data, and the difference in performance is marginal, therefore, our hypothesis is correct.

Competition Results

Our best performing model achieved a score of 0.887 with the test set which ranks us in the best 460 of the competition with only 0.06 difference from the first 20. A reason this might have happened is that some categorical values that existed in the training set did not exist in the test dataset.

Conclusion

The conclusions made from the overall project is that with datasets that contain mostly categorical features (strings encoded into 1 and 0), the algorithms that use some form of distance metric such as SVR and algorithms that fit curves and lines such as Polynomial and Linear Regression cannot achieve satisfactory performance and the best algorithms to use in these cases are tree, ensemble algorithms like Random Forest, Gradient Boosting etc.

Another conclusion that we drew from this project is that whether the data was scaled or unscaled **did not** play a crucial role in the **performance** of the ensemble machine learning algorithms as in both cases the achieved performance was similar.