FLIGHT FARE PRICE PREDICTION REPORT

BUSINESS PROBLEM

To create a model that will predict the flight fare prices based on some features such as type of airline, arrival time, departure time, duration of the flight, source, destination and more. This is a regression task since our goal is to predict a continuous variable which in this case is flight price.

THE DATASET

This is a public dataset containing passenger details of domestic flights in India with 10683 observations. There are 11 columns including the target variable “Price”. Most of the predictor variables are categorical with some of them being date/time in nature.

DATA PREPARATION

Before moving onto data analysis and data preprocessing, this particular dataset needed some additional steps of data cleaning and creation of new features. The most prominent of them all being the Date of Journey, departure and arrival time columns. These columns were unique string values that needed to be converted into a python datetime object so that more useful and meaningful features could be extracted from it. For example, from the Date of Journey column, the weekday, month and day of Journey has been extracted and 3 new individual columns have been created. Similarly, from the departure and arrival time the hour has been extracted. Also, the Duration column, initially a string type data, has been cleaned and converted to total duration in hours of float data type. The Airline column as well contained some values that need to be cleaned. Eg: 'Vistara Premium economy' was not another airline and would actually come under 'Vistara' while 'Premium economy' part of it is just additional information about the seat type. By doing this we will eliminate creation of unnecessary extra dimensions during the categorical data encoding step. Few changes were

also made to the Destination column where ‘New Delhi’ and ‘Delhi’ flights were put under the same label since departure would be from the same and only commercial airport in Delhi region- Indira Gandhi International Airport. Finally, the additional Info column needed some data cleaning and grouping of similar categories under one branch which significantly helped in reducing dimensions.

EDA OBSERVATIONS

Basic Domain Analysis was conducted where each variable was described and a general relationship was established with the target variable. For instance, after some online research, it was found that ‘Date of Journey’ influenced flight prices such that flights that take off and land on weekends, or Mondays and Fridays generally cost more as compared to midweek flights such as on a Tuesday or Wednesday. Similarly, ‘Departure Time’ as well affects flight prices. Flights scheduled during peak hours between 11am and 8pm are more expensive than unpopular times-of-day such as at dawn, during meal times or overnight “red-eye” flights.

A detailed univariate analysis was conducted and visualized using pie charts, count plots and histograms using the seaborn library. Some of the most insightful plots were the arrival and Departure time which showed distinct peaks during particular hours of the day. The ‘Month of Journey’ and ‘Day of Journey’ features did not contain all the months and days which would prove to be problematic in prediction of a fare of a flight that was supposed to depart on a month or day not included in the dataset.

Bivariate analysis was also performed using violin plots and bar charts that revealed interesting insights in the relationship of each variable with the target variable. For instance a violin plot of Airlines vs Price revealed how some airlines like Jet Airways had a higher median flight price as compared to more budget airlines like Indigo and SpiceJet which was inline with the domain analysis. The insights from the bivariate analysis corroborated the findings during the domain analysis.

FEATURE ENGINEERING

MISSING VALUES

Since the dataset had only two missing values (in ‘Route’ and ‘Total stops’), it was handled by removing that observation.

OUTLIERS

Random Forest Regressor was used as the final model which does not require handling outliers. Tree based algorithms are robust to outliers because every split slices the dataset and is condition based. Once the dataset is split, we only check the sample proportion in each split region and check how much impurity exists thus not considering the absolute value of any observation for any feature. Consequently, the outliers would end up in a particular branch of the tree and wouldn’t impact model performance.

CATEGORICAL ENCODING

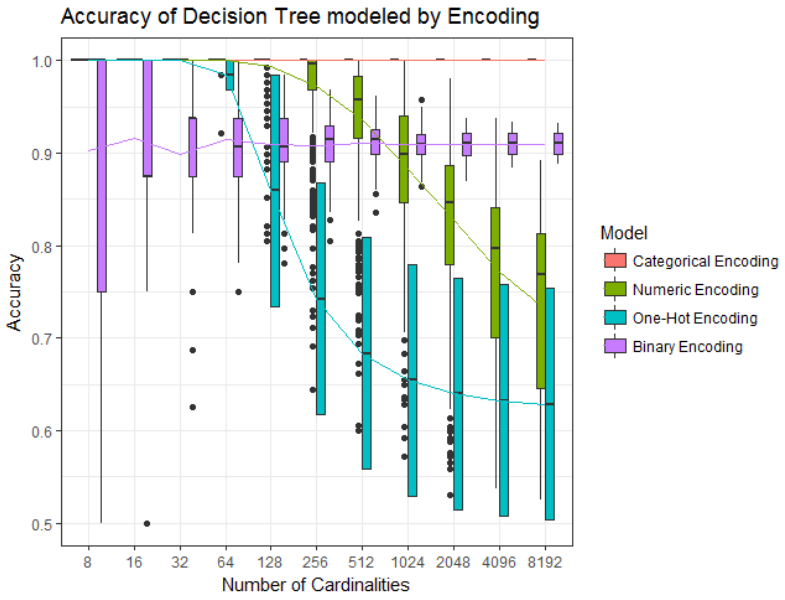
For this dataset several encoding strategies were used to convert the categorical data into numerical data while trying to minimize the number of new dimensions created. After encoding total number of features increased to 19. Few considerations that were made while choosing encoding strategies were as follows-

1. **One hot encoding does not perform well on tree-based models.**

   Tree based models particularly suffer from such a large number of one-hot features because the tree only grows in the direction of zeroes of one hot feature (which are more as levels of feature increase). Also, a single level of a categorical variable must meet a very high bar in order to be selected for splitting early in the tree building which can degrade predictive performance.

   Since our model contains many categorical features, applying one-hot encoding to all is unfavourable since it will result in creation of extra dimensions (curse of dimensionality) and also take up more storage space.

A solution to this was found by minimizing the usage of one hot encoding to only two features which are ‘Source’ and ‘Destination’. Moreover, some research in this area revealed that the accuracy of the model when using this encoding strategy is not much affected when number of cardinalities (levels) are lower.



https://towardsdatascience.com/getting-deeper-into-categorical-encodings-for-machine-learning-2312acd347c8

2. **Frequency Encoding to prevent creation of new dimensions.**

   To avoid the dimensionality curse, we used frequency encoding where every label in a feature is assigned the respective percentage if that label amongst all other labels. This prevents creations of an order as in label encoding since the label is assigned a value based on the number of times it appears. The major drawback is that the value assigned may not be in the same proportion as how that label influences the target variable. So, even though a particular label may have a higher value it may not truly have an influencing factor on the target and would be a misrepresentation. Frequency encoding is a widely used technique in Kaggle competitions, and many times proves to be a very reasonable way of dealing with categorical features with high cardinality.

3. **Manual Encoding was applied for total number of stops.**

FEATURE SCALING

Tree based models don't require feature scaling.

MODEL SELECTION

After pre-processing, the data was fit to 3 models- KNN Regressor, Linear Regression and Random Forest Regressor. Additional pre-processing steps such as feature scaling and outlier handling was performed for KNN and Linear Regressor. For all the models, the data was split into training and testing set using sklearn's train test split method.

EVALUATION/ MODEL PERFORMANCE

Every model was evaluated on the following metrics –

1. R2 score- a statistical value that tells us the proportion of variance in the target (dependent) variable that is explained by the predictor(independent) variables. A higher R2 score indicates that the model is more accurate and fit well to the data with lower residuals.
2. Adjusted R2 score – is a modified version of R2 that includes a penalization for addition of predictors (independent) variables that don't significantly contribute to the model performance.
3. Root Mean Squared Error (RMSE)- it is the standard deviation of the residuals (prediction errors). The lower the value the better.
4. Mean Absolute Error- it is the mean of the residuals.

The R2 score was used as the primary evaluation metric.

10-fold cross validation score was also calculated to compare model performance and for choosing the final model.

The model evaluations scores were as follows-

Model	R2 Score	Adjusted R2 score	RMSE	MAE	Average Cross Val Score	Std of cross Val score
Random Forest Regressor	0.89	0.89	1503.22	670.35	0.87	0.047
KNN Regressor	0.772	0.771	2167.14	1190.21	0.57	0.037
Linear Regression	0.532	0.531	3106.44	2150.77	0.51	0.034

From the above table we can clearly see that Random Forest performs the best and has the overall best scores.

KNN has an average performance on the R2 score however cross validation reveals an extremely poor performance.

Linear Regression as well performs extremely poorly which was expected since the model assumes a linear relationship between variables which is not the case for our dataset.

HYPER PARAMETER TUNING

Random Forest model having the best cross validation score was selected as the final model and hyperparameter tuning was performed. First a Randomized Search CV was performed to get a general range of best performing parameters and then a Grid Search CV as well was executed for the model.

The following hyper parameters of the model were tuned- max_depth, min_samples_leaf, min_samples_split, n_estimators.

	Best Estimator Parameters	Avg Cross Val Score
Randomized Search CV (Model 2)	max_depth=20, min_samples_leaf=3, min_samples_split=18, n_estimators=300	0.86
Grid Search CV (Model 3)	max_depth=11, min_samples_leaf=6, min_samples_split=8, n_estimators=400	0.84

No hyper parameter tuning (Model 1)	Default parameters	0.89

Despite the fact that the RF model with default parameters has the best cross validation score, it was not selected as the final model because it had a very high training r2 score of 0.97 which is indicative of an overfitted model.

Thus **Model 2** with parameters of the Random Forest Regressor Model being - max_depth=20, min_samples_leaf=3, min_samples_split=18, n_estimators=300 was selected as the **final model**.

Final Model 2 Evaluation Metrics-

- Root Mean Squared error: 1606.77
- Mean Absolute Error: 789.61
- Testing r2: 0.874
- r2_train: 0.89
- 10-fold Cross validation Score: 0.86 with std 0.04