Aviation prediction

FEB 2022 - APR 2022

Master of Technology Project Report

Members:

Pradeep Janakiraman
Prashant Nair
Vikram Sankireddypally
Institute of System Science
May 1, 2022

Contents

1 Executive Summary

The aviation industry has been growing rapidly over the last few decades and the demand for air transport is forecasted to increase by an average of 4.3% over the next 20 years. This is not surprising, given that an increasing number of people are living in urban city areas. The UN predicts that by 2050, two-thirds of the world's population will be living in cities. Given this rising airline passenger footprint, the aviation industry has become competitive, with every airline trying to offer the best packages. As a passenger, one of the key decisions we make is with regards to price and comfort. We want the best possible flight for the cheapest possible price. However, we are often spoilt for choice.

Normally, the process of booking a flight involves going online to one of several booking engines and finding a flight to our destination that seems reasonable in terms of price. However, we realised that often times we don't have any benchmark regarding how cheap should flights be. If there is a tool that informs us what the prices should be like when travelling between two places, we could then use it to judge whether the prices shown in the booking engines are inflated. This is the kind of tool that our team has attempted to build.

Using a South-East Asia data set on airline operating data and airline ratings data containing passenger ratings of the airline, our team has built a flight price and ratings predictor using an explainable machine reasoning technique and have designed a simple UI for a user to interact with the system. Furthermore, we also recognise that price is something not the only factor when it comes to deciding between different airlines as we are also concerned about the quality. As such, we have also included a recommendation engine that gives a predicted rating for an airline given the user's past rating history using techniques from collaborative filtering.

2 Business use case

The current atmosphere in the commercial aviation industry is one of friendly competition, which has lead to airlines striving to price its itineraries at the best possible rates. However, passenger are not able to reap the real benefits of the best price due to the dynamics of ticket pricing used in the aviation industry by key players. The few key players playing the role of booking engines are shown in the figure below.

Figure 1: Major booking sites and fare aggregators

Our solution in order to allow passengers to be more aware of price trends and make better judgements is a simple yet smart platform called "Aviation Prediction" which would enlighten aviation passengers on the best possible price for a given route and a given airline. Furthermore, our platform will also give a predicted rating for the airline chosen by the passenger which gives the passenger an idea of how good their flight would be. An example of the UI with our platform implemented is shown below.

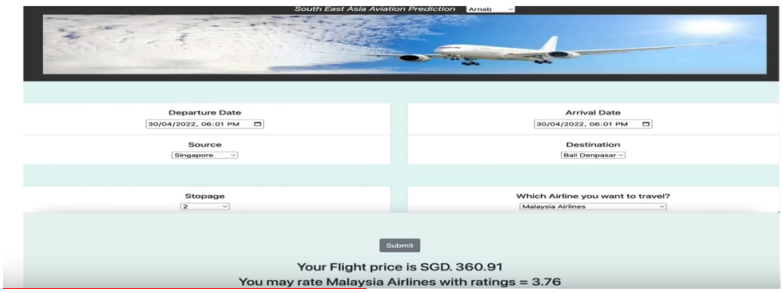

Figure 2: UI of the aviation prediction platform

The future of our implementation would be to integrate with the booking sites above and to use their real time data to adjust and train our airline price and ratings predictor.

3 Structure of dataset

3.1 Airline price data

The raw dataset that was used for our flight prediction model contained a total of 11 columns with the following labels and example data.

- Airline: Scoot
- Date of journey: 24/03/2019
- Source: Jakarta
- Destination: Singapore
- Route: CGK → SIN
- Departure time: 22:20
- Arrival time: 1:10
- Duration: 2h 50m
- Total stops: non-stop
- Additional info: No info
- Price: 260

Before this data set could be used for training an appropriate model, some of the fields had to be modified and new columns needed to be added. The first step was to remove data entries with null values in them. Secondly the column "Date of Journey" is not convenient for computation and had to be split up into two columns of day and month. Similar procedures were done on the columns containing information on the departure time and arrival time since the raw time cannot be used for proper computation. In addition, the duration field also have to be processed as entries like "2h 50m" are not computable. The duration hours column was therefore split into two further columns one for number of hours and another for number of minutes.

Next, from the dataset it can be seen that there are few fields that contain categorical data. In order to train our model, we apply one-hot encoding on the columns with Airline, Source and Destination. The one-hot encoding procedure converts the categorical variables into many columns with 1 indicating the presence of a variable and 0 indiating its absence, as illustrated below:

$$\begin{bmatrix} \text{Airline} \\ \hline \text{Scoot} \\ \text{Jetstar Asia} \\ \text{Singapore Airlines} \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} \text{Scoot} & \text{Jetstar Asia} & \text{Singapore Airlines} & \cdots \\ \hline 0 & 0 & 1 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

We see above, that one column of categorical data is converted to many columns of ones and zeros. The same one-hot encoding procedure is repeated with the source and destination

columns. Finally, some of the columns in the raw data set are not required, such as "Additional info" and "Route", hence they are removed from the final data set. We are then ready to train a model with the resulting dataframe, which now has 29 columns as opposed to the 11 we started with.

3.2 Airline ratings data

Other than the price of the airline, our system also aims to provide a predicted rating for the user given their past rating history. The data frame needed to accomplish such a task needs to have information on the users and their ratings for different airlines. The raw data is in the following format:

User	Airline	Rating
Slawosz	ScootBiz	2.5
Slawosz	Singapore Airlines	3.5
Arnab	AirAsia	3
⋮	⋮	⋮

Table 1: Raw ratings table

In order to apply recommendation techniques, we have to convert the above table into a ratings matrix for easy computation. This would lead to the creation of a sparse matrix, since we expect there to be many users relative to the total number of airlines and most users would not have rated every airline. The ratings matrix will then have the following structure:

Users	ScootBiz	Singapore Airlines	AirAsia	⋯
Slawosz	2.5	3.5	–	⋯
Arnab	–	–	3	⋯
Alex	–	3	3	⋯
⋮	⋮	⋮	⋮	

With this format, the similarity between users can be computed using various methods which will be covered later. The result of the similarity computation would then lead to a similarity matrix which contains information on how similar the users are to each other. Finally, with this table we can recommend items based on predicted ratings for unrated items derived from similarity information. The similarity matrix will then have the following structure.

–	Slawosz	Arnab	Alex	⋯
Slawosz	1	0.5	0.3	⋯
Arnab	0.5	1	0.2	⋯
Alex	0.3	0.2	1	⋯
⋮	⋮	⋮	⋮	⋮

For the purposes of this project, the user ratings data was created just to demonstrate the concept. But it can be applied to a real case where large ratings data are created. The price data set that was obtained online did not have any ratings which we could use.

4 Data exploration

Before, training the data we performed some data exploration to see if there was any insight to be gained. Below are some plots where we compare the the price of flights based on different factors.

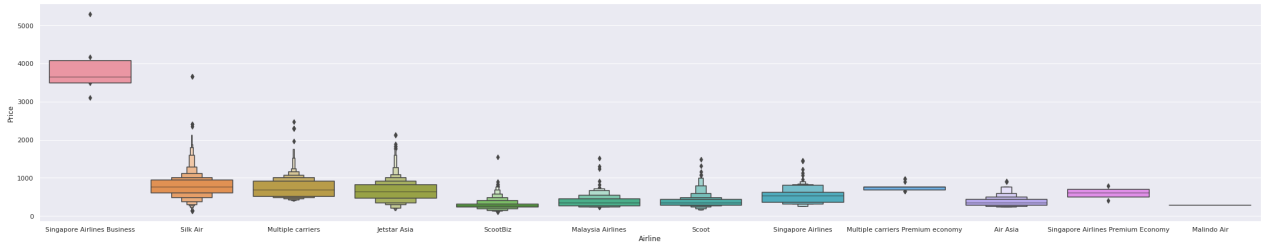

Figure 3: Price vs Airline. Singapore Airlines Business is the clear outlier here.

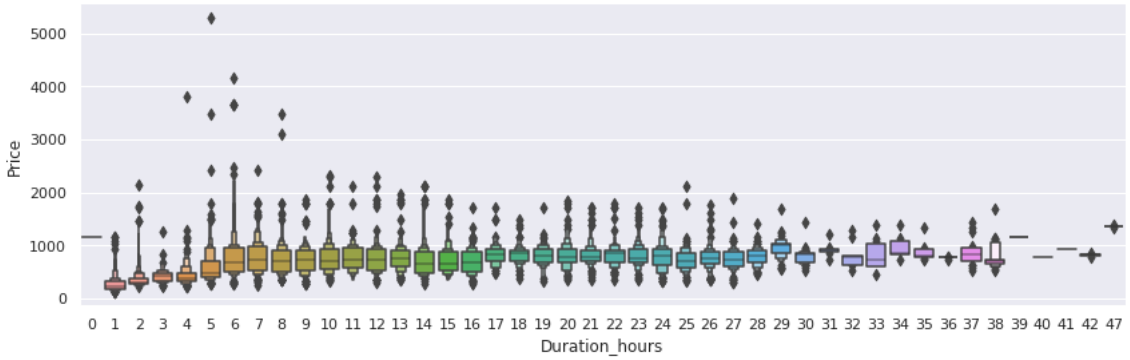

Figure 4: Price vs Duration hours. The flights price don't vary much beyond a certain number of duration hours.

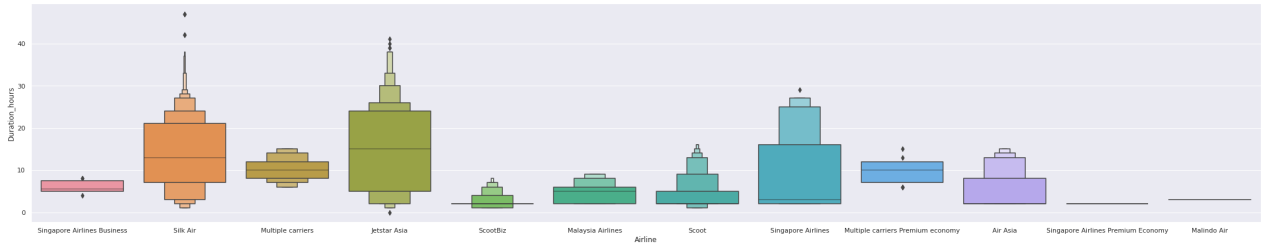

Figure 5: Duration of flight vs Airline. This shows that some of the more expensive airlines do not fly for long hours (at least the ones in the dataset) which might explain the previous figure.

Another possible interesting visualisation would be to see which variables were related to each other, and in particular since we know that price is the independent we can see which factors is the price of a flight most strongly related to using a correlation heat-map as shown below. We can see that the price has the strongest correlation with the number of stops and the duration of the flight and of course there is a strong correlation between the number of stops and the total duration of a flight. The importance of each feature can also be estimated and quantified using the `feature_importances_` function as part of sklearn's library. The relative importance of each feature in predicting the price is shown below.

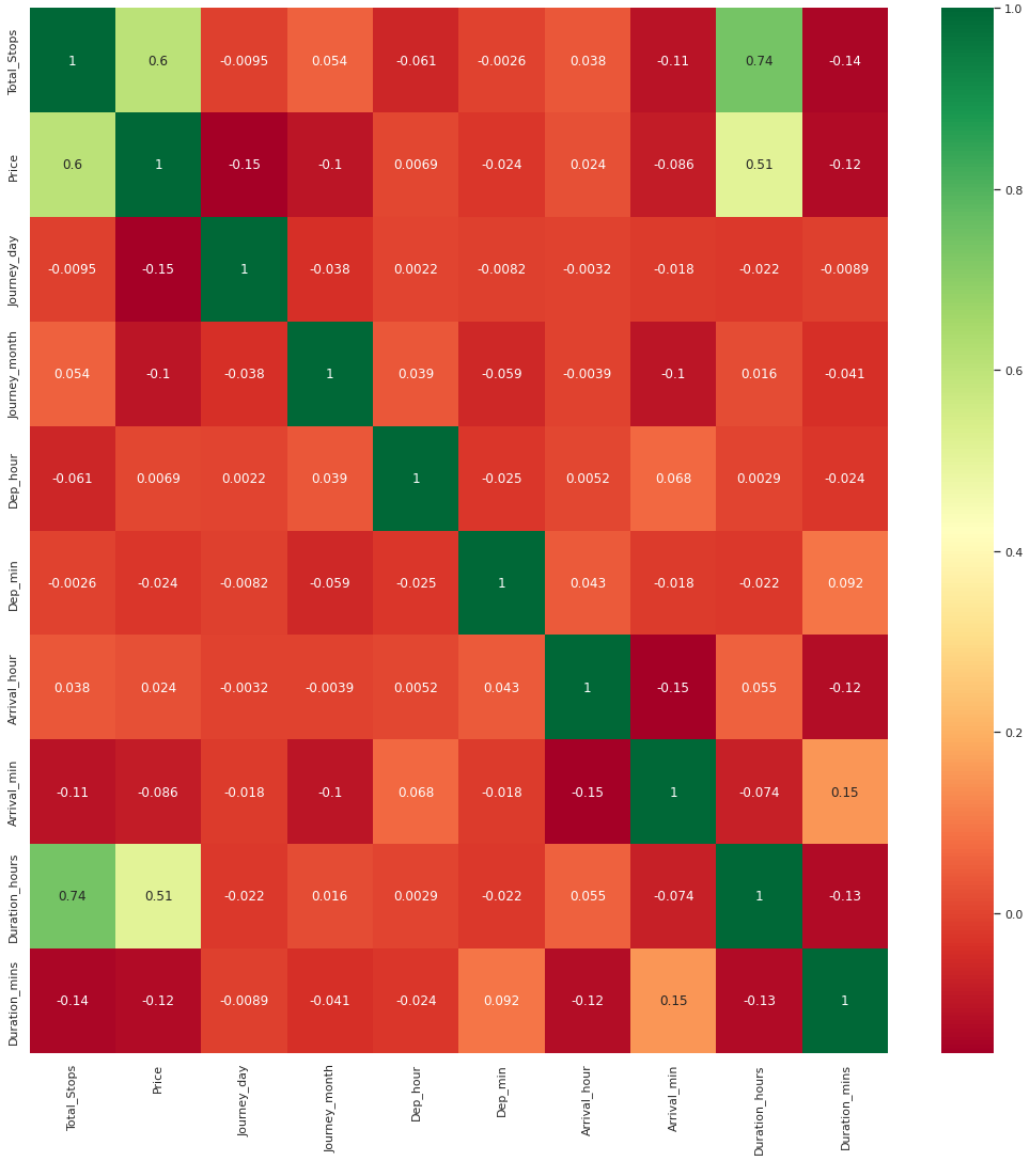

Figure 6: Correlation heat map of the different variables

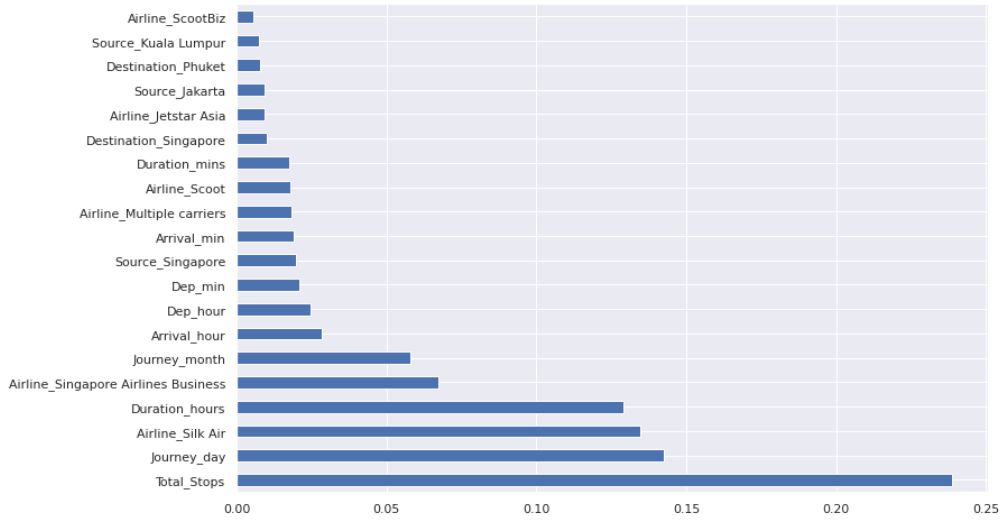

Figure 7: Relative importance of the different features in relation to price prediction

5 Building a predictor

Based on the exploratory data analysis, we can conclude that there is no clear relationship between any of the factors other than some correlation that was seen between the duration of the flight and the price. However, beyond a certain duration the correlation was not present. As a result, since we have a relatively large number of features and no obvious correlation we can exploit, we have chosen to go for a decision tree stye regression method in order to predict the price of a flight.

5.1 Regression Tree methodology

Regression trees are useful tools for data that has many independent variables and with no obvious functional relationships between the dependent and independent variables, so it can handle non-linearities farely well. The basic concpet is that the regression tree contains nodes where a condition is checked to be true or false based on the input data different paths of the tree can called until we arrive at an estimate. The estimate is based on the average value of data points that have the same responses to the conditional statements. The general structure of a regression tree is as follows:

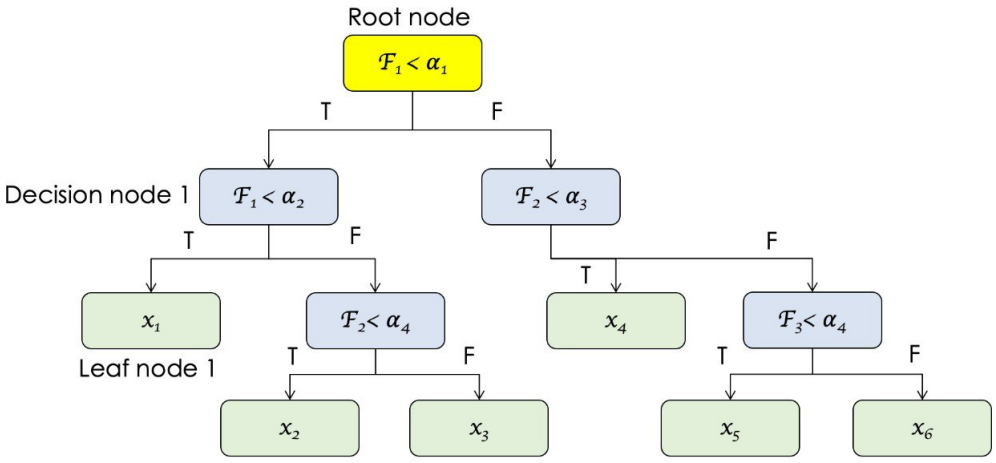

Figure 8: Example of a regression tree

The blue nodes seen in the above figure are examples of conditional statements where F_n refers to the feature in question, for example in our case F_1 could refer to "Duration in hours". α_n are the numeric values that the conditional checks with the feature. Finally the green nodes are estimates given by the regression tree, therefore all the x_n would be price estimates in our case.

The conditionals (blue nodes) are decided recursively based on finding the smallest sum of squared residuals for each data set. For the root node, the entire data set is considered for an appropriate conditional, while for the remaining blue nodes the data set is split into smaller groups.

The disadvantage of a single trained decision tree is that it may lead to over fitting. Th=o counter this, we employ the random forest method, which uses an assortment of decision trees to provide a regression result.

5.2 Random forest regression

The random forest regression technique is based on the principle explained above for regression analysis, however it creates and ensemble of decision trees and provides an estimate based on the results from all the created decision trees. A key feature of the random forest is that the data set used to train each of the trees is different. From the original data sets, multiple data sets are created using a "bootstrap" method. The bootstrap method creates a data set of the same size by randomly selecting samples from the original dataset as shown below.

$$\begin{bmatrix} \text{Samples} & \text{Feature 1} & \text{Feature 2} & \cdots \\ \text{Sample 1} & \cdots & \cdots & \cdots \\ \text{Sample 2} & \cdots & \cdots & \cdots \\ \vdots & & & \\ \text{Sample n} & \cdots & \cdots & \cdots \end{bmatrix} \rightarrow \begin{bmatrix} \text{Samples} & \text{Feature 1} & \text{Feature 2} & \cdots \\ \text{Sample 2} & \cdots & \cdots & \cdots \\ \text{Sample 23} & \cdots & \cdots & \cdots \\ \text{Sample 7} & \cdots & \cdots & \cdots \\ \text{Sample 7} & \cdots & \cdots & \cdots \\ \vdots & & & \\ \text{Sample 2} & \cdots & \cdots & \cdots \end{bmatrix}$$

Both the original dataset on the left and the bootstrap dataset on the right are the same size. With the bootstrap method it is possible to create a dataset with repeated samples since they are randomly selected from the original dataset. This bootstrap data set is then used to train one decision tree. During the training process, not all the features are used to decide the conditional nodes at each step. By step, we refer to the current depth of the tree. The deeper the tree goes, the more conditional nodes and leaf nodes are created. The features are randomly selected at each step, for example the features that might be used for one decision tree created as part of the random forest for our case might be:

- Step 1: Duration Hours, Arrival Hours, Total Hours
- Step 2: Airline, Source_Singapore, Departure Hour
- ⋮

At the end of this process we create a regression tree. This processes is repeated again until enough trees are created as desired. For the regression task, a sample with the n number of features is taken as input and run through all the created decision trees. The numerical result from all the decision trees is then averaged out to give a final value. This process of creating bootstrapped data, randomly selecting features at each step to create a certain number of regression trees and finally aggregating and averaging out the results obtained from a given new sample is called "bagging".

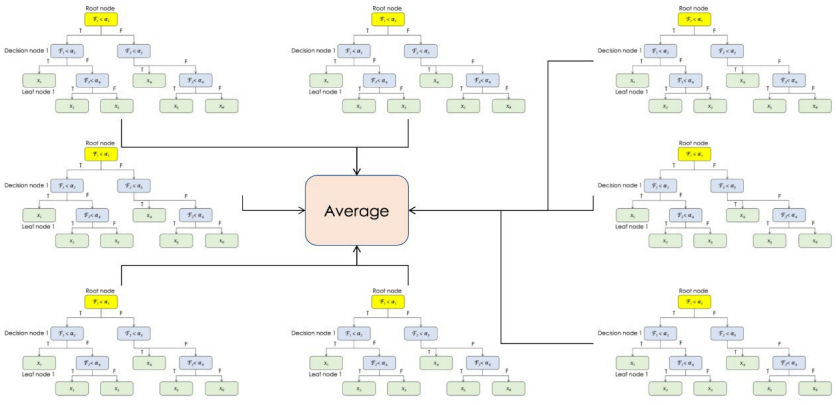

Figure 9: Bagging process for random forest

5.3 Collaborative filtering

For the prediction of user rating as mentioned in subsection 3.2, once we have the ratings matrix we can begin to find the similarity between users. This can be done using the pearson similarity, which has the advantage of eliminating ratings bias between users as some users might generally give higher or lower ratings with the same level of satisfaction for a given product. The pearson similarity is then given by:

$$\mathbf{Sim}(u, v) = \frac{(\vec{u} - \vec{\bar{u}})^\top \cdot (\vec{v} - \vec{\bar{v}})}{|(\vec{u} - \vec{\bar{u}})||\vec{v} - \vec{\bar{v}})|}$$

The above equation of course assumes, that the vectors $\vec{u}, \vec{v}$ represents the ratings for all the possible flights given by user u and v respectively. And the vectors $\vec{\bar{u}}, \vec{\bar{v}}$ represent the vector containing a single value which is the average rating given by user u, v across all flights. However, it is very likely that some flights will not be rated by a user in which case these null entries are neglected in the computation of the dot product. Once the similarity is found betwween all users, we can predict a rating of user u for item i as follows:

$$\tilde{u}_i = \frac{u_s^\top \cdot i}{\sum_{j \in v} u_s^j}$$

In the above formula, $\tilde{u}_i$ is the predicted rating of user u for item i, u_s is the vector containing the similarities of user u with all other users v in the data base, where u_s^j represents an element in this vector.

6 Results

We created the random forest regressor using `sklearn`'s built-in library. Without any tuning, and using the default setting for the random forest function, we trained the model using our pre-processed data set and with a test-train split of 40-60%. We then got predictions from the created model by feeding the model inputs from the test data to obtain y_p which represents the predictions from the trained model on the test data. We could then compare to the actual prices in the test data y_t to judge the performance of the model.

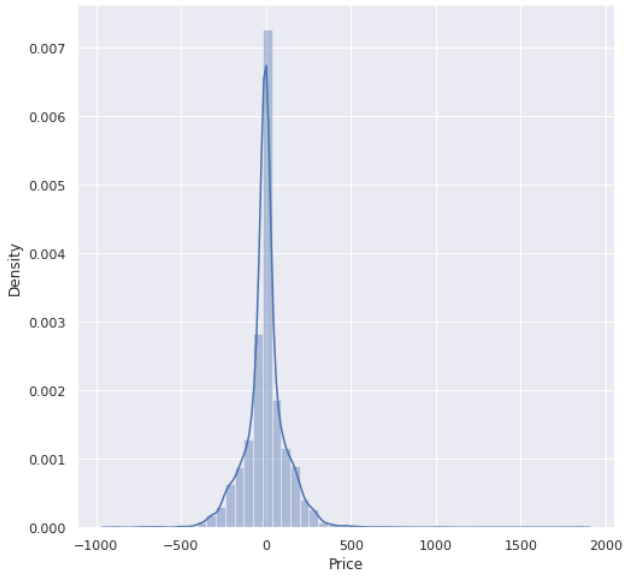

Figure 10: Distribution of $y_t - y_p$

Besides the above histogram, we can also use other methods such as the root-mean-squared error, which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ti} - y_{pi})}$$

The smaller the value of the RMSE, the closer the relationship between y_t and y_p would have to a linear function with gradient 1, which can be visualised in the below graph.

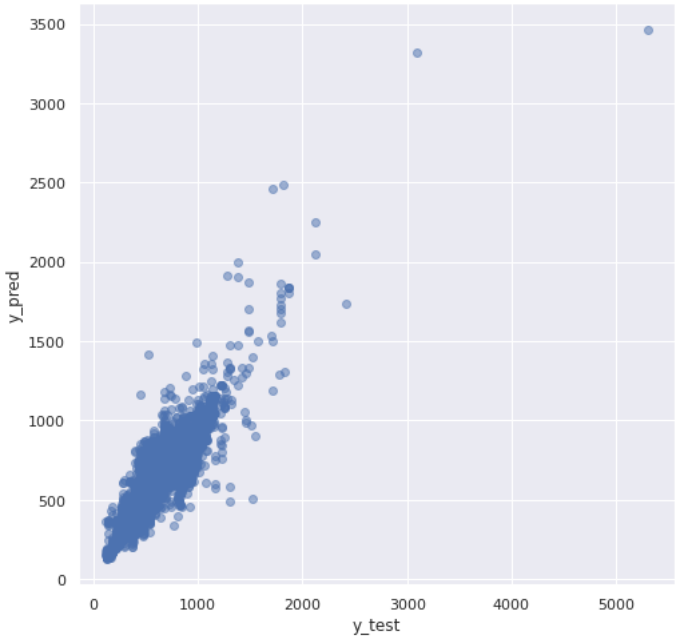

Figure 11: y_t vs y_p. RMSE = 123.9, suggesting that the model is off by an average of 123 dollars on average.

In order to improve the mode, we attempted to tune some of the hyperparameters that are involving in the building and training of the model. Specifically, we iterated across the following parameters and values.

- Number of decision trees in the forest → iterate 12 steps between 100 - 1200.
- Depth of the decision trees in the forest → iterate 6 times between 5 - 30.
- The minimum number of observations required to split an inter node in a decision tree: $[2, 5, 10, 15, 100]$
- The minimum number of observation required to have a valid leaf node: $[1, 2, 5, 10]$

We then iterated through all the possible combinations of the parameters listed above and evaluated them with each other using the negative mean-squared error function. We then arrived at the following results for the best parameters.
We saw a minor improvement in the RMSE values in the end from 123 to 118. The corresponding distribution and scatter plot for the improved model obtained from the grid search are given below.

Parameter	Value
Number of trees	700
Maximum depth of trees	20
Minimum #observations to split node	15
Minimum #observation for valid lead node	1

Table 2: Final parameter values chosen

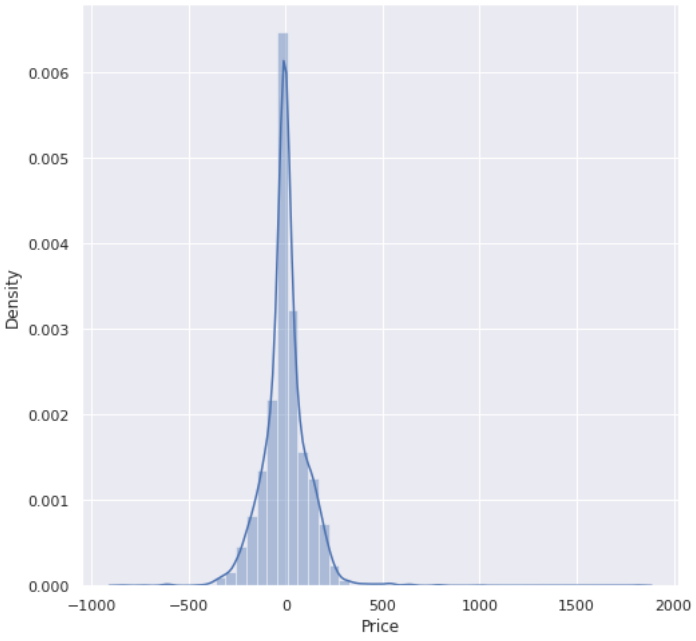

Figure 12: Distribution of $y_t - y_p$ for the improved model.

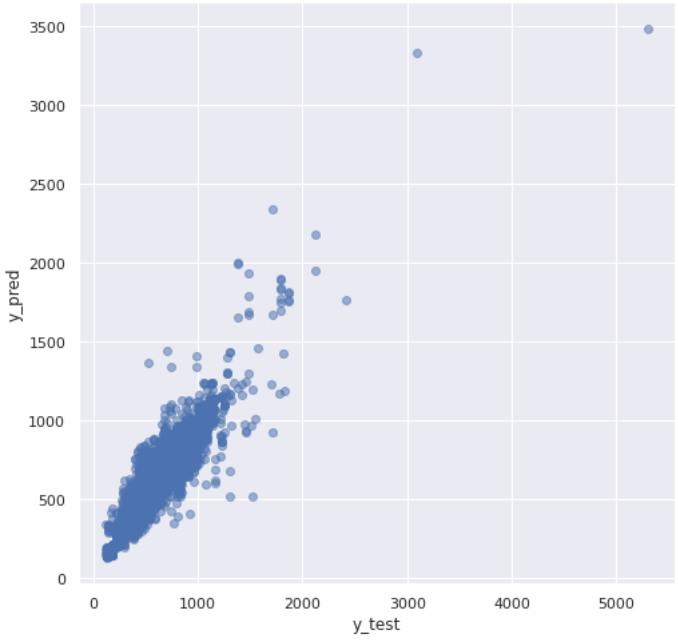

Figure 13: y_t vs y_p. RMSE = 118, suggesting that the model is off by an average of 118 dollars on average.

7 Future plans for implementation

So far, the discussion has been on the technical aspects of our predictive model and in this section we propose how our system can be implemented in a real scenario. For example, if we wanted to integrate our solution with existing popular booking sites like Trip.com, we would first need an API gateway like the Azure API gateway. We would then need an ingestion layer like kafka or azure data factory to get data from the gateway to a data lake. The Azure Data Lake Storage will store data from all sources. We then need to build a data pipeline to suit our recommendation systems and machine learning models to handle our ratings and airline data along with Azure Synapse Analytics. Furthermore, we can use the Azure machine learning platform to build, train and deploy our random forest model and collaborative filtering recommendation system, which would then be integrated with the front end using the Flask framework. The figure below summarises the our proposal for the full scale architecture in a real scenario.

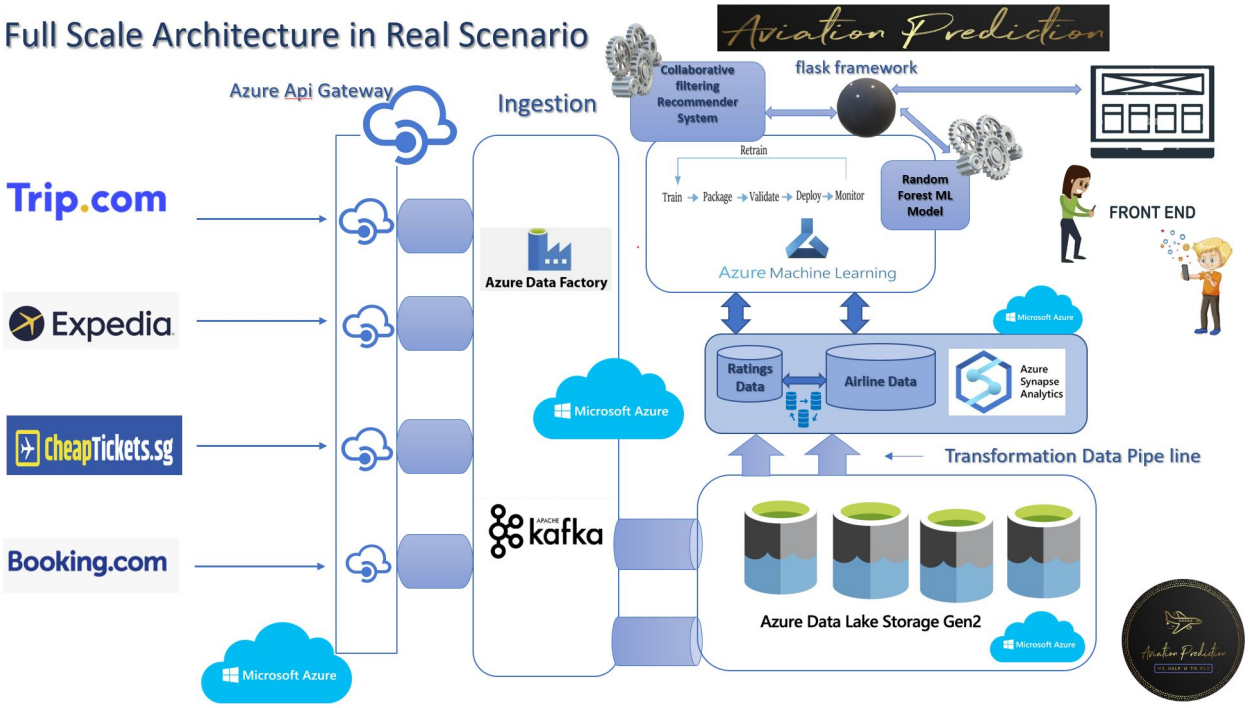

Figure 14: Full Scale Architecture in a real scenario