Blog is generally on ‘Flight Price Prediction’ an evaluation project as part of the PG Data Science curriculum in on **Data trained** academy  walks you through each and every step in detail and helps us to understand the whole ML model building process . So, let’s get started.

Problem Statement



Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, and it will be a different story. We might have often heard travellers saying that flight ticket prices are so unpredictable.

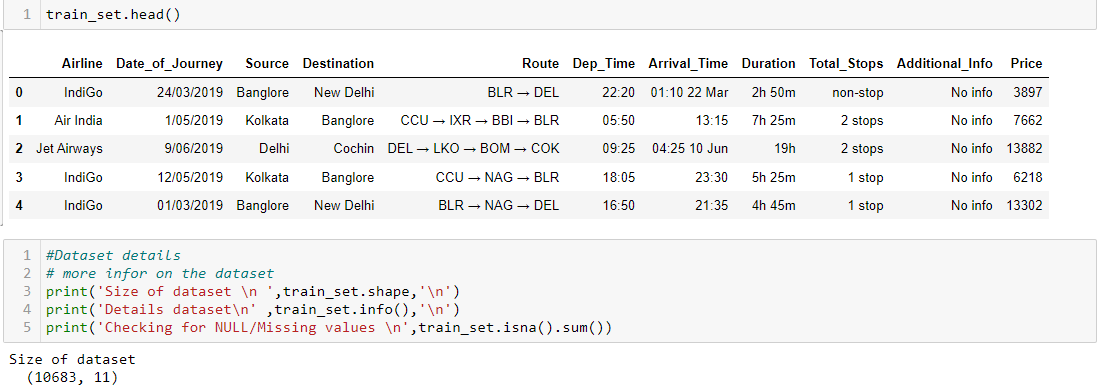
As data scientists, we will be responsible to build a model to predict the flight ticker fare correctly .The dataset is provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

**Datasets**

We will be using two datasets — Train data and Test data, which can be downloaded from kaggle.com or below

<https://github.com/diptasarathi/DataTrained/blob/main/Eval_Week3/Flight_Ticket_Participant_Datasets-20190305T100527Z-001.zip>

**The dataset sample:**



Screenshot of the **Train dataset with** (10683 rows and 11 columns): Training datarefers to that portion of data used to fit a model.

Training data is consisting of both numerical & categorical also we can observed some special character are also used the data transformation on the data is required before applying it to our model

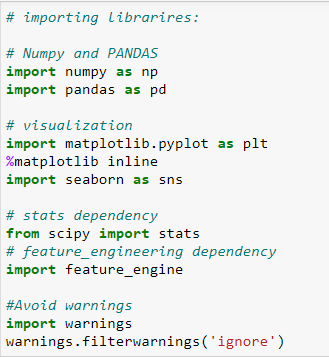


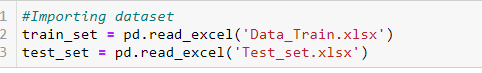
Screenshot of the Test data (2671 rows and 11 columns): Test data refers to that portion of data used to test the efficiency a model.

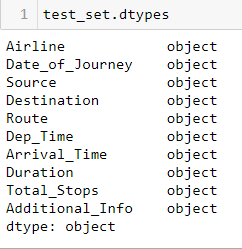
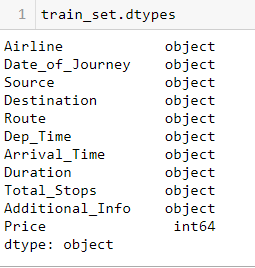
The test data is similar to the training data set, except the ‘Price’ column which needs to be predicted using the model we will be building

**Python Coding**

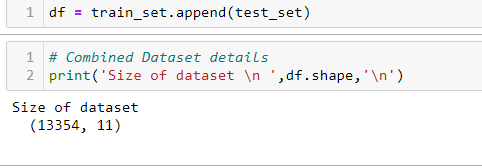
**Step 1: Import the relevant libraries in Python and the dataset for excel.**







## Step 2: we will append these sets for feature engineering the data



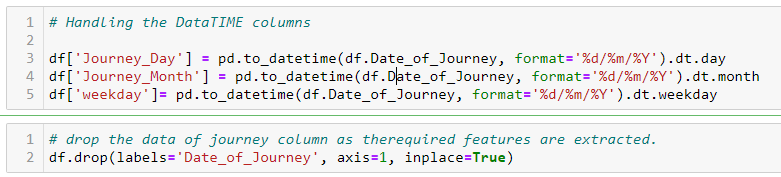
Appending of the data set is done to work together with both train and test at a same time in order to avoid doing the same steps separately .Once the Data engineering / transformation is completed then we can separate them again into test and train for model training and testing

**Step 3: Feature Generation [Univariate/Bivariate]**

In this step the main objective is to work on the data set and perform some transformation such as creating different bins of particular columns ,clean the dirty data so that it can be used in our ML model . This step is very important as to build a model with high prediction score with a higher accuracy

**Date\_of\_Journey:**

In the column ‘**Date\_of\_Journey**’, we can see the date format is given as **dd/mm/yyyy** and as you can see the datatype is object. Now there are two ways to tackle this column, either convert the column into Timestamp or divide the column into date, Month ,day. Let’s take the approach to divide them into multiple columns

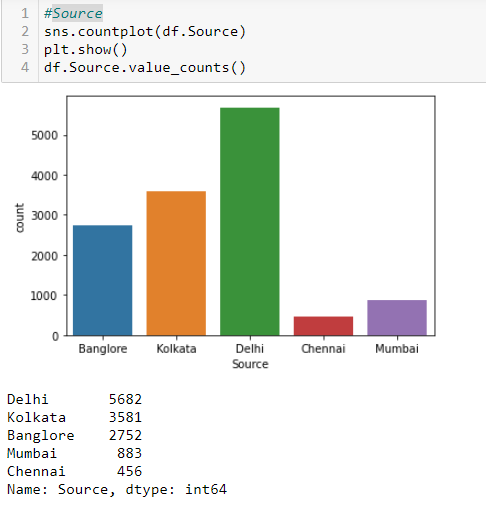


Date\_of\_Journey will be split into 3 features (Date, Month, weekday ). Once done we will drop the columns ‘Date\_of\_Journey’ and Arrival\_Time.

**Source & Destinations Stops:**

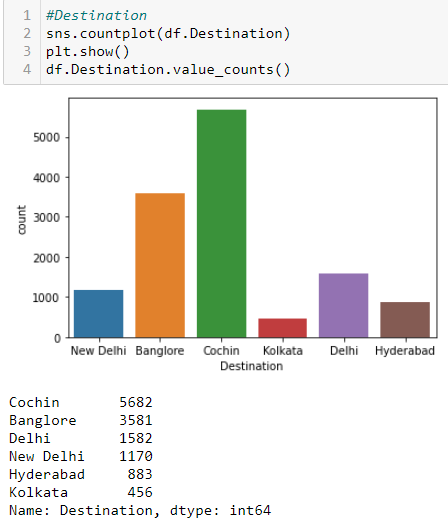
These columns are basically 2 columns which show the source and destination, Count wise

For Source



Delhi has high on boarding followed by Kolkata .

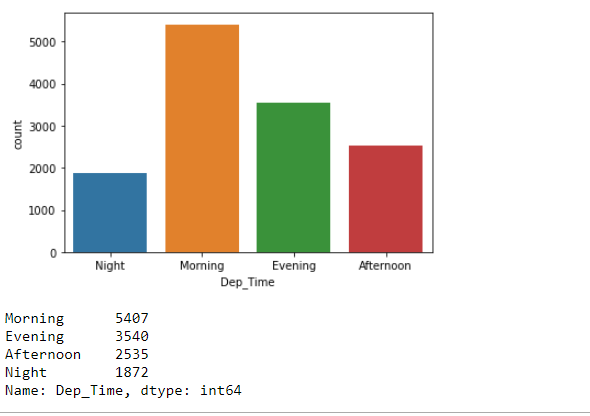
For Destination



Thus if we observe correctly we can observer most of the flights landed in COCHIN or Bangalore.

**Dep\_Time:**

We divided the departure time into 4 slots to see when the flight frequency will be higher



Most Busy airport is Bangalore as the source and Destination count is higher as compare to others

Delhi has high on boarding and Cochin has a higher arrival

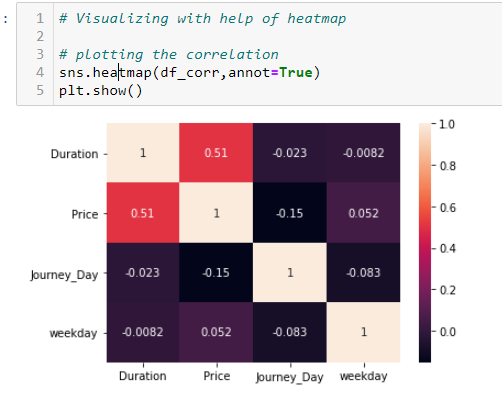
Most flights are from 5-11 am or 4-9 PM

**Route:**

The ‘Route’ columns mainly tell us that how many cities they have taken to reach from source to destination .This information can be leveraged from the columns **Total\_Stops**

**Heat Map for correlation**

**Even when we plotted the Coreeation Matrix , we couldnot find any heavily related feature**

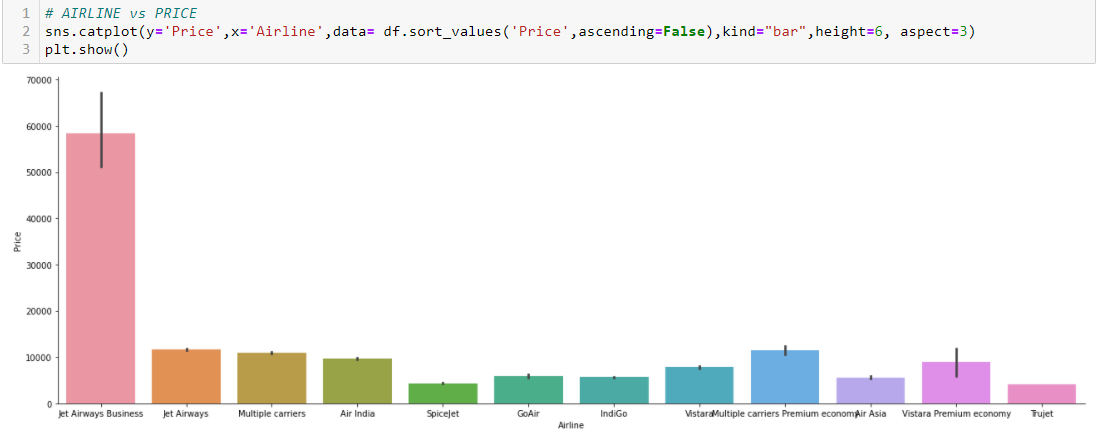
****

**Bivariate Analysis:**

Bivariate analysis is a kind of statistical analysis when two variables are observed against each other. One of the variables will be dependent and the other and vice versa

**FLIGHT vs Airfare:**

When we plot a graph between the Airlines and Ticket Price , we find that for Business class the price are high as compared to the economy class



**Departure time and Air fare:**

Flights with better convenient timings like Evening has higher price, Morning and night have relative lower than Evening. Night has Cheapest fare of all

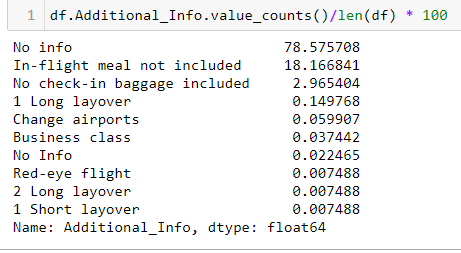


**Total Stops and Air fare:**

We observe that the Route plays a vital role in setting the price for the



**Additional \_INFO**

This Column shows add on facilitates availed while purchasing the ticket 

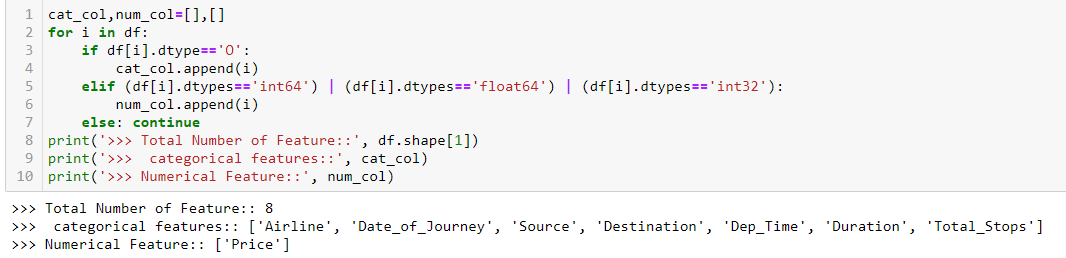
Since 78 percent data is missing from the column it will be safe to remove the column for Datasets



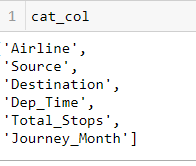
**Step 4: Prepare categorical variables for model using label encoder**

To convert categorical text data into model-understandable numerical data, we use the Label Encoder class. So all we have to do, to label encode a column is import the Label Encoder class from the sklearn library, fit and transform the column of the data, and then replace the existing text data with the new encoded data. For that we must segregate the Numerical and Categorical data

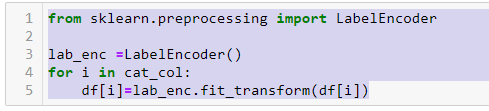
We can do this by running a loop where based on data type it can easily devide then into Numerical or Categorical



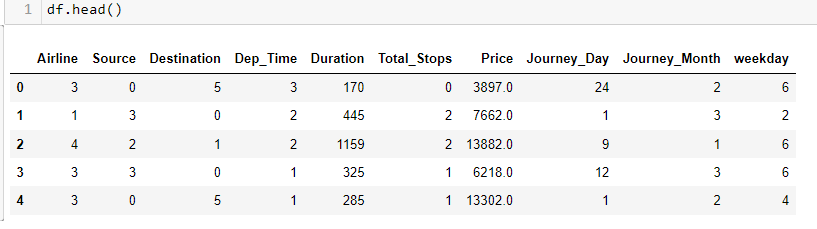
Label encoding of Categorical variables



Label Encoding :

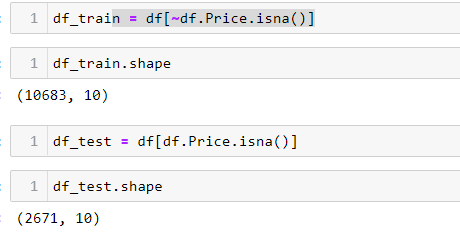


Data set post Encoding

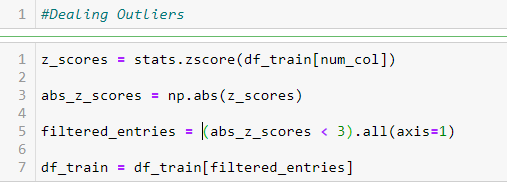


**Step 5 :** **Divide the data set into test and train**

Now that all our data is numerical after label encoding so we split the data into test and train and drop the price column from the test set because we have to predict the price with our test data set

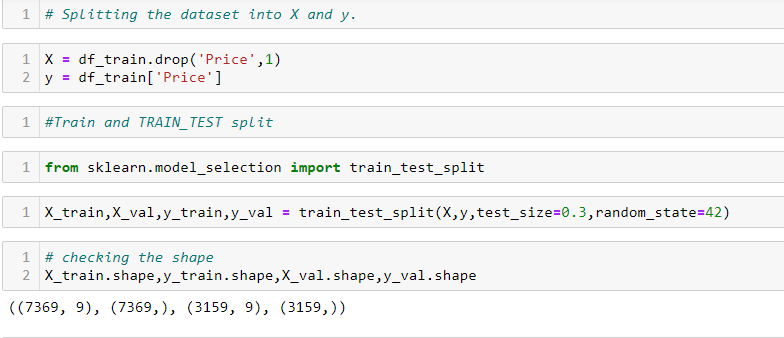


**Step 6: Eliminate the outliers for Numerical variables for model using ZScore method for Test data**

****

**Step 8: Split the Train model to X & Y for Training**

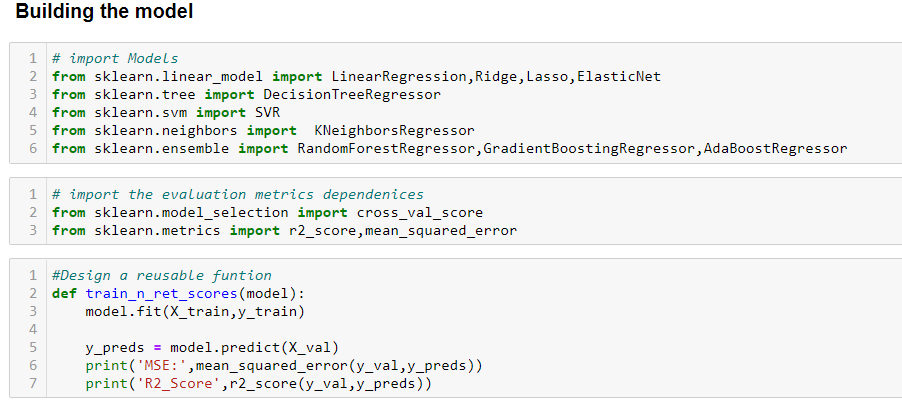
To train any model the data needs to be divided into feature data and label data. First it needs to be used to train the data and then the same model is used to predict for test data .It must be ensured that the Shape of these data split is in sync

****

**Step 9: Building Model and Train**

The main objective in this step is to develop a benchmark model that serves us as a baseline, upon which we will measure the performance of a better and more tuned algorithm. We will be using different Regression Models and compare t them to see which algorithm is giving better performance other and At the end we will combine all of them using Stacking and see how our model is predicting

FOR THAT WE WILL USE A Reusable function :



1. **Linear Regression :**

**MSE( Mean Square Error):** 8148120.587022088

**R2\_Score**: 0.5100815518785173

1. **Ridge Regression:**

**MSE( Mean Square Error):** 8148191.78661949

**R2\_Score**: 0.5100772708917678

**3. Lasso Regression:**

**MSE( Mean Square Error):** 8148120.589969974

**R2\_Score**: 0.510081551701271

**4. KNeighborsRegressor:**

**MSE( Mean Square Error):** 6837890.721709402

**R2\_Score :** 0.5888611643598058

**5.** **RandomForestRegressor**

**MSE( Mean Square Error):** 3938115.1513332403

**R2\_Score :** 0.7632146894661993

**6.** **DecisionTreeRegressor:**

**MSE( Mean Square Error):** 5687174.818501988

**R2\_Score :** 0.658049751286957

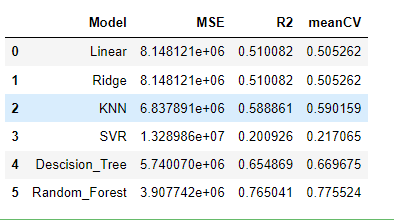
**RMSE( Root Mean Square Error):**1747.2331238078746

**7. Cross Validation:**

Cross-validation is a re-sampling procedure used to evaluate machine learning models on a limited data sample.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.



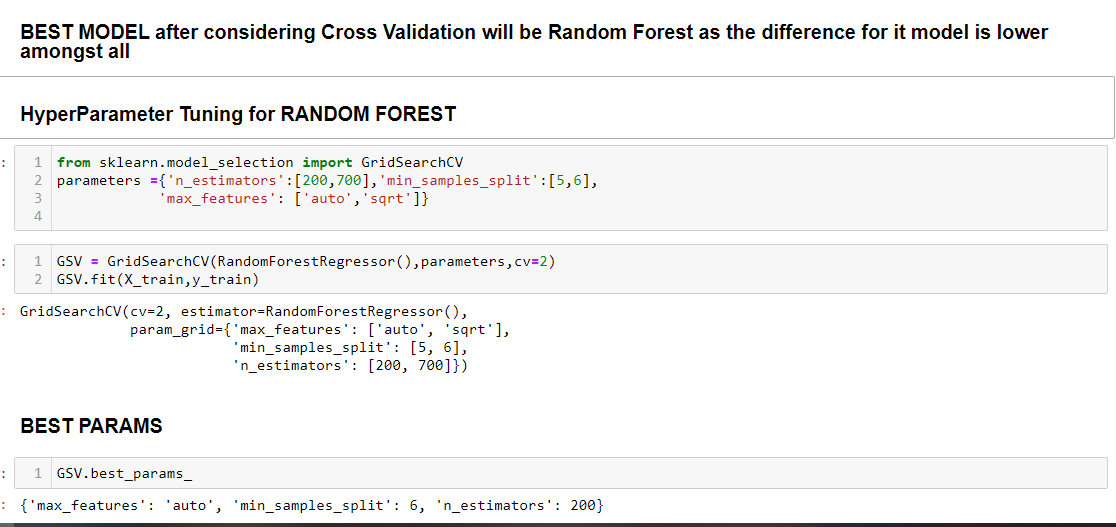


Thus we found the Random Forest Model to be apt . TO make it more suitable we will use Hyper parameterisation

**Step 10: Hyper parameter Tunning:**

Parameters which define the model architecture are referred to as **hyperparameters** and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning.

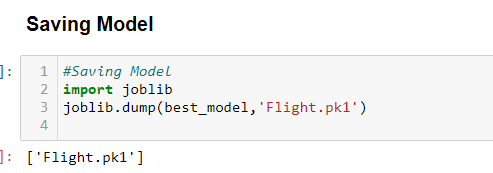
Implementing Hyperparamters we get the Best Parameters, we will rerun the training model with it.



Once done the Test data will be run on the model and fine R2 score will be calculated

**Step 10: Saving Model for deployment**

Once done we will save the model for deployment using joblib or pickle libraries

****

**Final Conclusion**

In this type of problems , the Feature Engineering is the most important part. We need to handle the categorical and numerical data so that we do not lose any influential data and also how we build different ML models on the training dataset. We also check the R2 score of each model so that we can understand how it may perform in our test dataset . At last the Model is made more efficient by implementing Hyperparameter Tuning..

Please let me know your thoughts about this article and do comment if you face any issues while implementing it **Sources :**

1. [Cross-Validation. Validating your Machine Learning Models… | by Kurtis Pykes | Towards Data Science](https://towardsdatascience.com/cross-validation-c4fae714f1c5)
2. [Hyperparameter tuning for machine learning models. (jeremyjordan.me)](https://www.jeremyjordan.me/hyperparameter-tuning/)
3. <https://www.analyticsvidhya.com/>