Introduction

This work focuses on using machine learning methods and algorithms in order to predict the flight price of the given dataset

Regression analysis is used when you want to predict a continuous dependent variable from a number of independent variables. If the dependent variable is dichotomous, then logistic regression should be used. (If the split between the two levels of the dependent variable is close to 50-50, then both logistic and linear regression will end up giving you similar results). The independent variables used in regression can be either continuous or dichotomous. Independent variables with more than two levels can also be used in regression analyses, but they first must be converted into variables that have only two levels. This is called dummy coding and will be discussed later. Usually, regression analysis is used with naturally-occurring variables, as opposed to experimentally manipulated variables, although you can use regression with experimentally manipulated variables. One point to keep in mind with regression analysis is that causal relationships among the variables cannot be determined. While the terminology is such that we say that X "predicts" Y, we cannot say that X "causes" Y.

It is a good idea to check the accuracy of the data entry. If you don't want to re-check each data point, you should at least check the minimum and maximum value for each variable to ensure that all values for each variable are "valid." For example, a variable that is measured using a 1 to 5 scale should not have a value of 8.

Regression analysis also has an assumption of linearity. Linearity means that there is a straight-line relationship between the IVs and the DV. This assumption is important because regression analysis only tests for a linear relationship between the IVs and the DV. Any nonlinear relationship between the IV and DV is ignored. You can test for linearity between an IV and the DV by looking at a bivariate scatterplot (i.e., a graph with the IV on one axis and the DV on the other). If the two variables are linearly related, the scatterplot will be oval.

The assumption of homoscedasticity is that the residuals are approximately equal for all predicted DV scores. Another way of thinking of this is that the variability in scores for your IVs is the same at all values of the DV. You can check homoscedasticity by looking at the same residuals plot talked about in the linearity and normality sections. Data are homoscedastic if the residuals plot is the same width for all values of the predicted DV.

Problem Definition:

**Flight Price Prediction -A Regression Analysis using Lazy Prediction**

## **1. Objective**

The objective of this article is to predict flight prices given the various parameters. This will be a regression problem since the target or dependent variable is the price (continuous numeric value).

## **2. Introduction**

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices.

Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That’s why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly

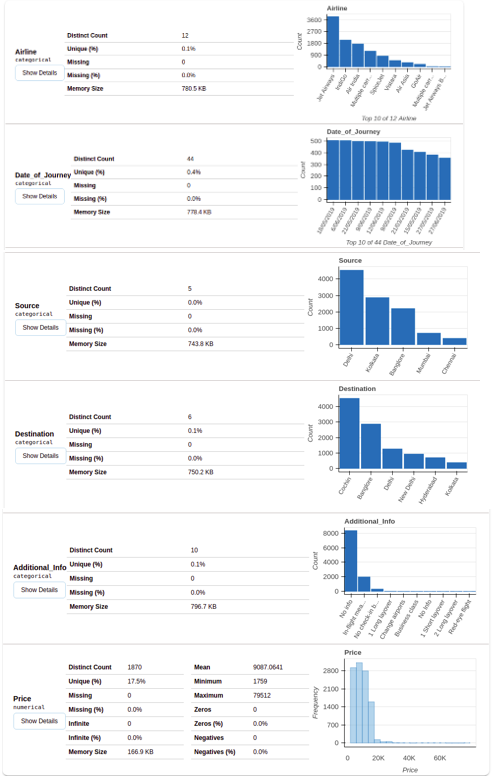
## **3. Data Used**

Data was used from “http://projects.datatrained.com”

We are using Jupyter-notebook to run Flight Price Prediction task.

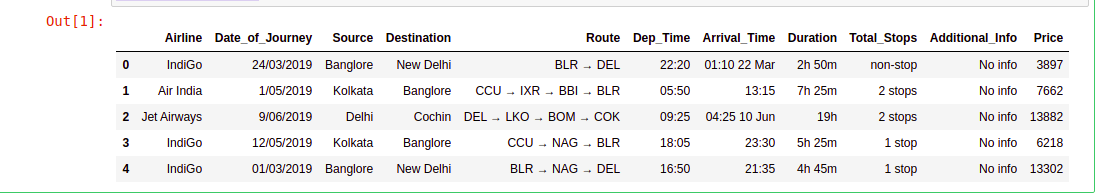
### **4. Variables**

After you select the variable section, you will get information as shown in the below.



## **5. Data Preparation**

Before starting data preparation let’s have a glimpse of data first.



As we saw in Data Analysis there are 11 variables in the given data. Below is the description of each variable.

**Airline**: Name of the airline used for traveling

**Date\_of\_Journey**: Date at which a person travelled

**Source**: Starting location of flight

**Destination**: Ending location of flight

**Route**: This contains information on starting and ending location of the journey in the standard format used by airlines.

**Dep\_Time**: Departure time of flight from starting location

**Arrival\_Time**: Arrival time of flight at destination

**Duration**: Duration of flight in hours/minutes

**Total\_Stops**: Number of total stops flight took before landing at the destination.

**Additional\_Info**: Shown any additional information about a flight

**Price**: Price of the flight

Few observations about some of the variables:

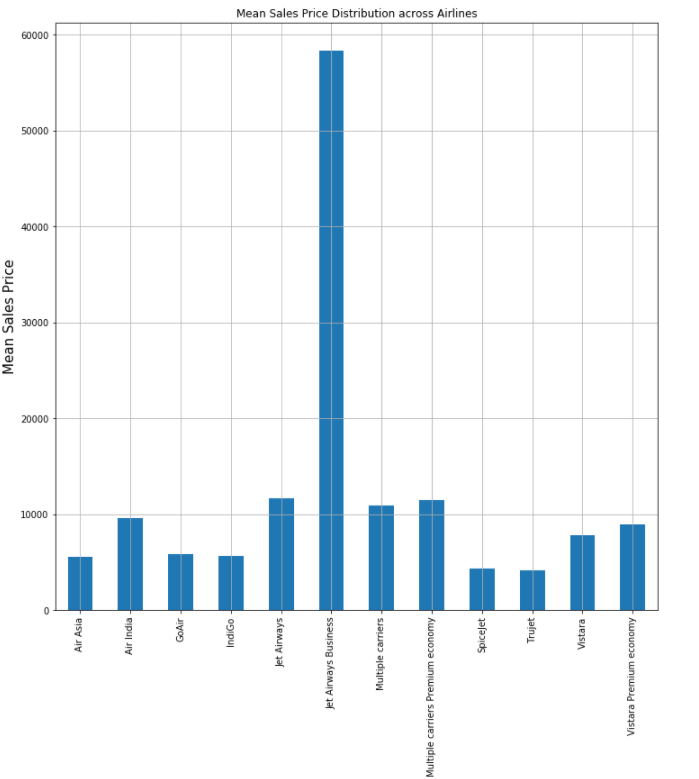
1. ‘**Price**’ will be our dependent variable and all remaining variables can be used as independent variables.

2. ‘**Total\_Stops**‘ can be used to determine if the flight was direct or connecting.

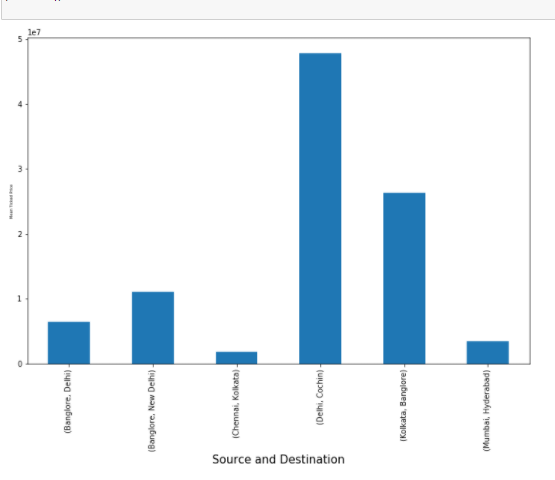
EDA:

**Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.**

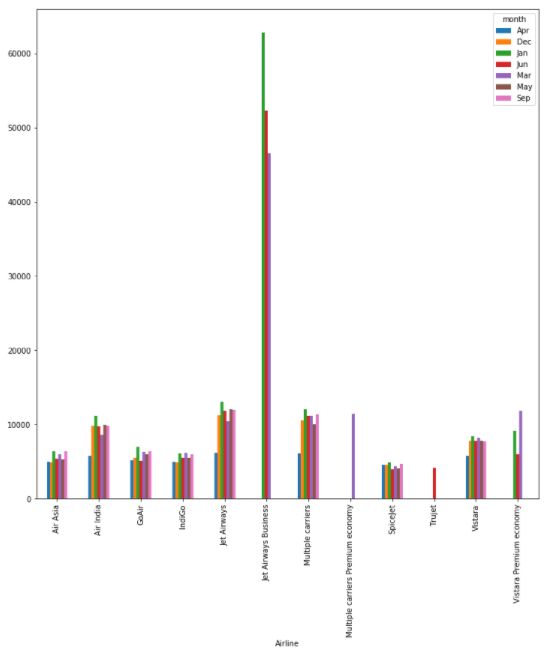
# Plot showing distribution of Price across Airlines



# Distribution of price between source and destination

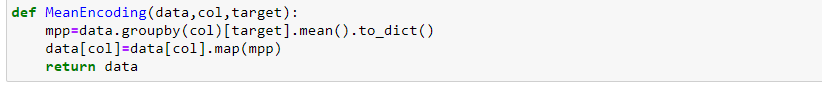


# Plot showing Distribution of price across months and airlines



### **5.1 Handling Categorical Data**

Airline, Source, Destination, Route, Total\_Stops, Additional\_info are the categorical variables we have in our data. Let’s handle each one by one

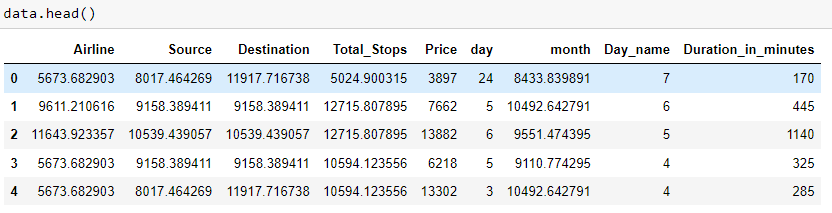


As we can see the name of the airline matters. ‘Jet Airways Business’ has the highest price range. Other airlines price also varies.

Since the **Airline** variable is **Nominal Categorical Data** (There is no order of any kind in airline names) we will use **Mean Encoding** to handle this variable.

### **5.2 Final Data frame**

Now we will create the final data frame by concatenating all the One-hot and Label-encoded features to the original data frame. We will also remove original variables using which we have prepared new encoded variables.



## **6. Model Building**

Various Machine Learning Models are used in this step.

One of the problems of the model-building exercise is ‘How to decide which machine learning algorithm to apply’?

Since we are working on a Regression task, we will use Regressor models

Model Used

1. LinearRegression
2. RandomForestRegressor
3. GradientBoostingRegressor
4. Adaboost
5. XGBOOST

**Simple Linear Regression**: A linear regression model with one independent and one dependent variable.

**Multiple Linear Regression**: A linear regression model with more than one independent variable and one dependent variable

**A random forest** is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

**GradientBoostRegressor:** GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

**AdaBoost regressor** is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

**XGBOOST:**

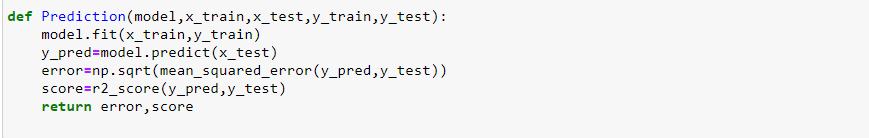
Extreme Gradient Boosting, or XGBoost for short, is an efficient open-source implementation of the gradient boosting algorithm. As such, XGBoost is an algorithm, an open-source project, and a Python library.

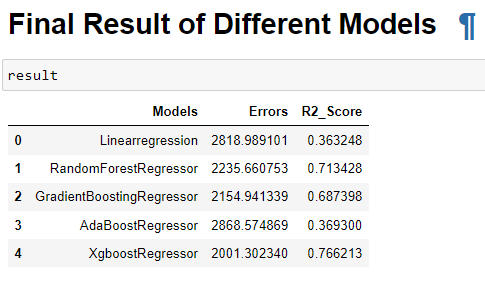
It was initially developed by [Tianqi Chen](https://www.linkedin.com/in/tianqi-chen-679a9856/) and was described by Chen and [Carlos Guestrin](https://homes.cs.washington.edu/~guestrin/index.html) in their 2016 paper titled “[XGBoost: A Scalable Tree Boosting System](https://arxiv.org/abs/1603.02754).”

It is designed to be both computationally efficient (e.g. fast to execute) and highly effective, perhaps more effective than other open-source implementations.

The two main reasons to use XGBoost are execution speed and model performance.

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems. The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform





## **7. Conclusion**

Out of all the Machine Learning Models used XGBOOST gave the highest r2\_score