**Project Report – Cab fare Prediction**

* **Gaurav Khurana**

**Problem Statement –**

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

**Data Set :**

1) train\_cab.zip 2) test.zip

**Number of attributes:**

· pickup\_datetime - timestamp value indicating when the cab ride started.

· pickup\_longitude - float for longitude coordinate of where the cab ride started.

· pickup\_latitude - float for latitude coordinate of where the cab ride started.

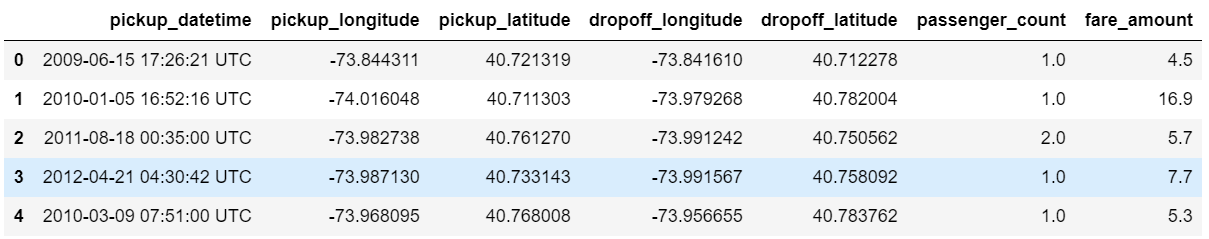
· dropoff\_longitude - float for longitude coordinate of where the cab ride ended.

· dropoff\_latitude - float for latitude coordinate of where the cab ride ended.

· passenger\_count - an integer indicating the number of passengers in the cab ride.

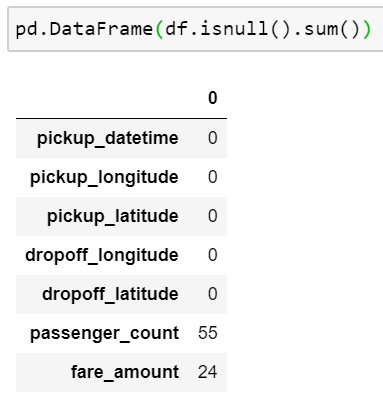
**EDA (Exploratory Data Analysis) –**

The given data was arranged in the following manner:

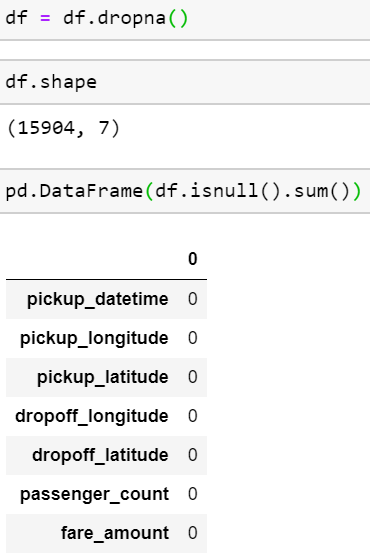


Shape of the data set – (16067,7)

1. **The data set was then checked for missing values**

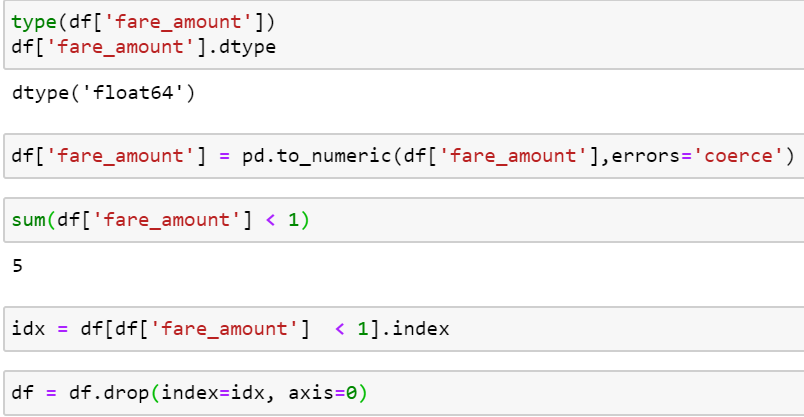


As it can be observed that there are 55 and 24 missing entries for passenger\_count and fare\_amount variables respectively which are fairly insignificant as compared to the data set almost as low as 0.003% and thus, can be dropped.

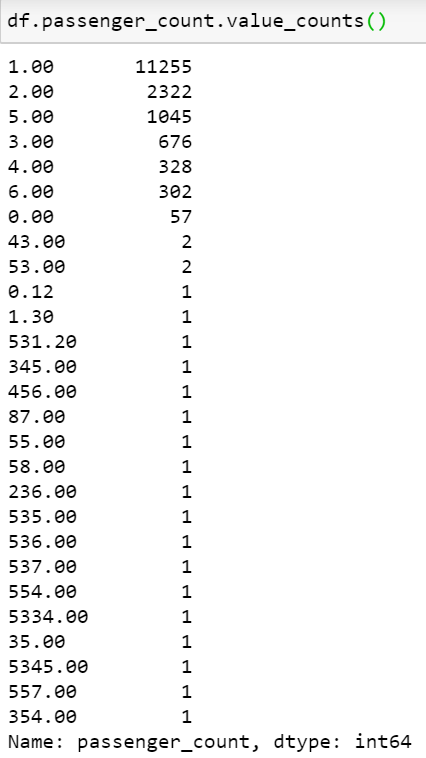


1. **Checking to see whether all the independent variables were in the correct range of values.**

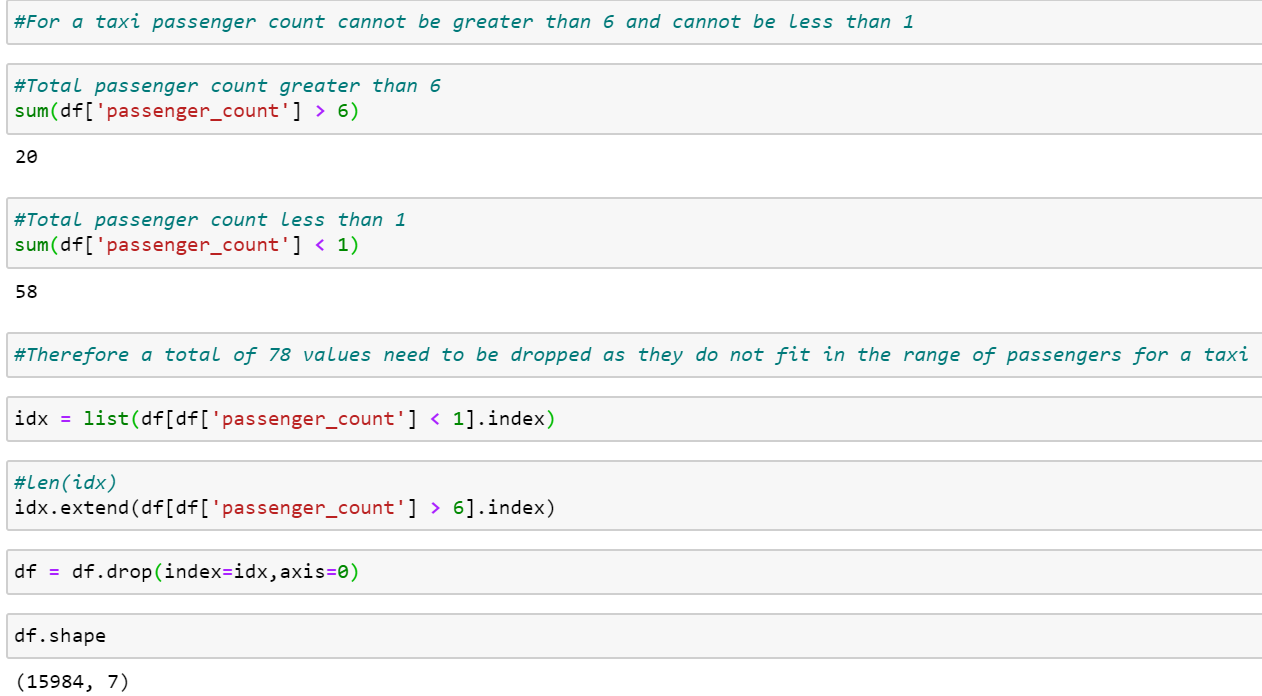
* Considering the variable fare\_amount, logically no fare amount can be less than 1 as every cab ride already starts with a certain base amount. In this case, I have decided that amount be minimum and set it to 1 unit. Thus, any fare amount less than 1 had to be treated. Since, the number of such examples were very nominal hence, I decided to drop all such rows.



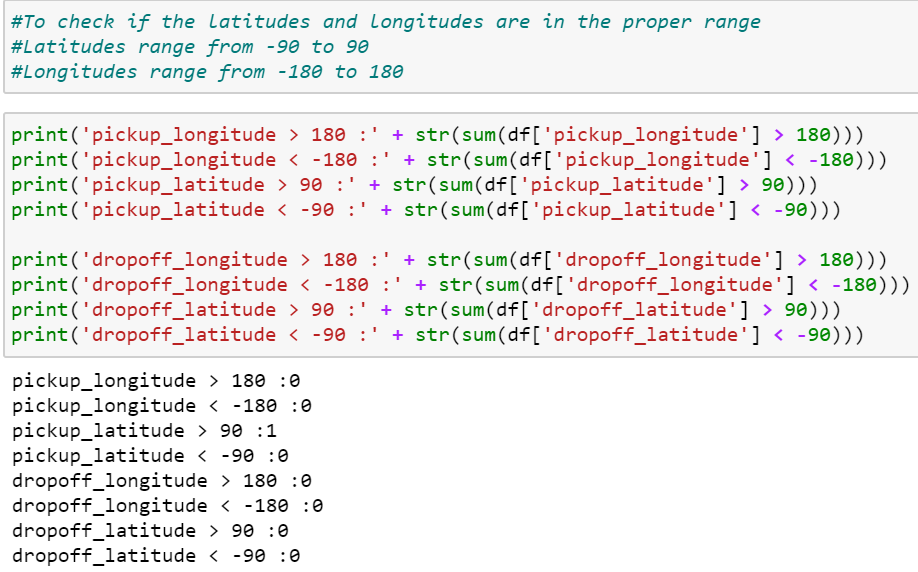
* Considering the variable passenger\_count, this variable indicates the number of passengers that travelled together in the cab at the same time. There were some fairly absurd values as can be seen below:



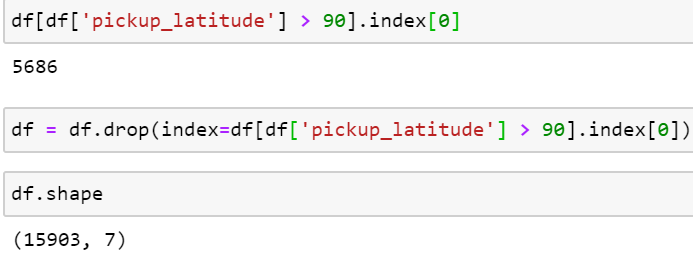
It is fair to say that it is almost impossible for a cab to hail that high number of passengers. That is why a limit of maximum 6 passengers were decided. Thus, all those values that were either below 1 or greater than 6 had to be treated and also the value that was a fraction (since fractional passengers are not possible) has to be treated. Since, the sum of all these values combined was < 0.05% of the total values, these values could be dropped without them influencing much on the data set.



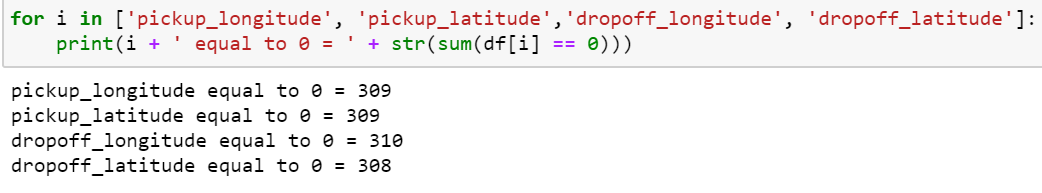
* There are 2 latitude variables i.e. pickup\_latitude, dropoff\_latitude and 2 longitude variables i.e. pickup\_longitude, dropoff\_longitude. When dealing with latitude and longitude it needs to be kept in mind that latitudes only exist between a range of -90 to 90 and longitudes between -180 to 180 and also that neither latitudes nor longitudes can have a value of 0.

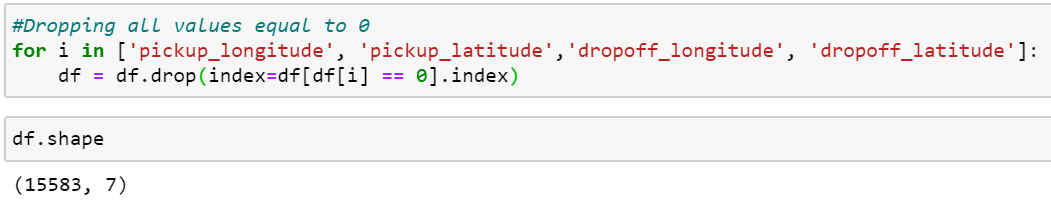


It is observed that only 1 such value exists that is out of range and therefore that value needs to be dropped.

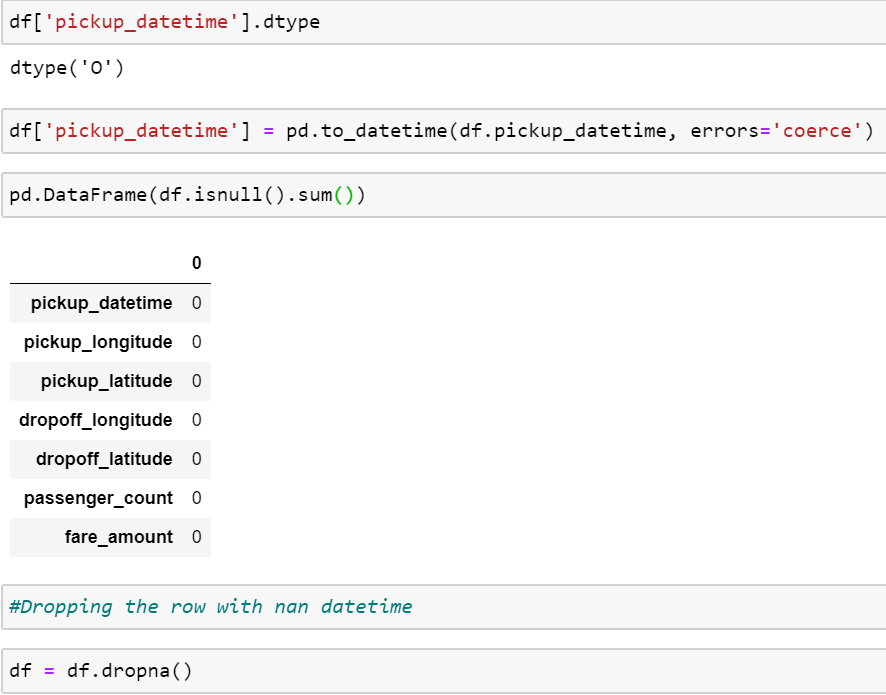


Same process was followed to check if any values were equal to 0, and then those values were dropped as well.



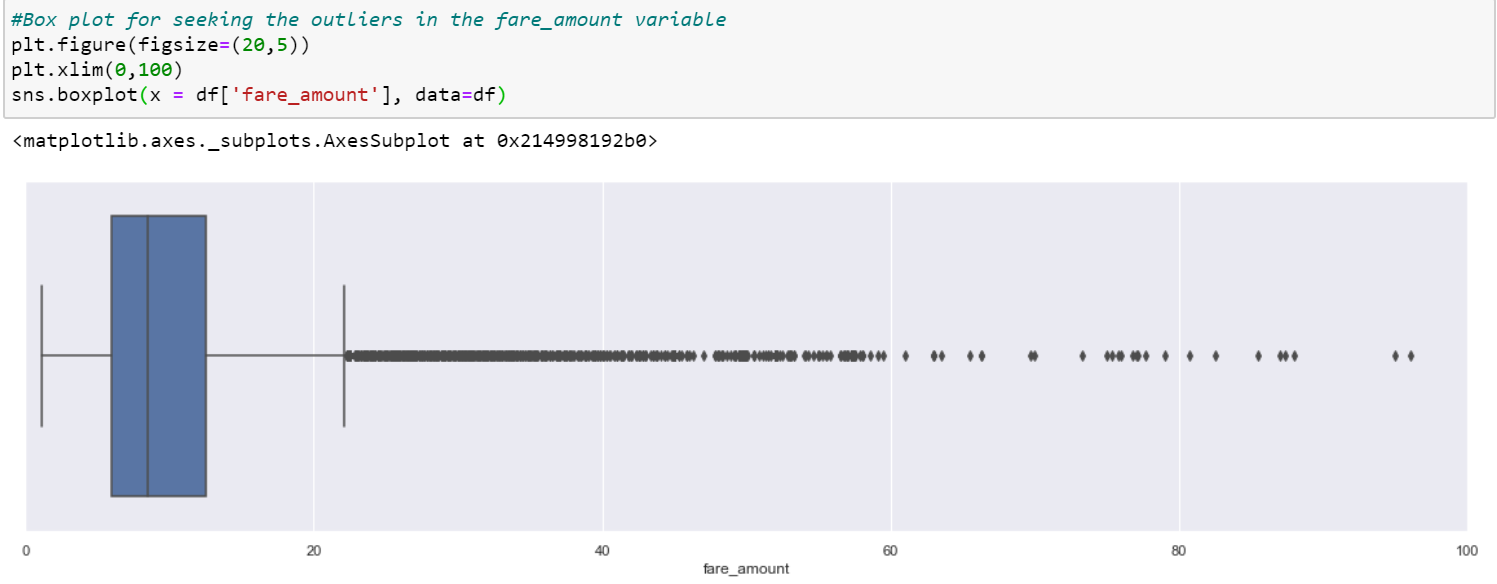


* The next variable to be examined was pickup\_datetime, the type of the variable was first converted into datetime using pandas to\_datetime function. On doing it was checked to see if there were any new missing values that had been created from the conversion of data type. It was observed that there was only one such value generated and thus could be easily dropped.

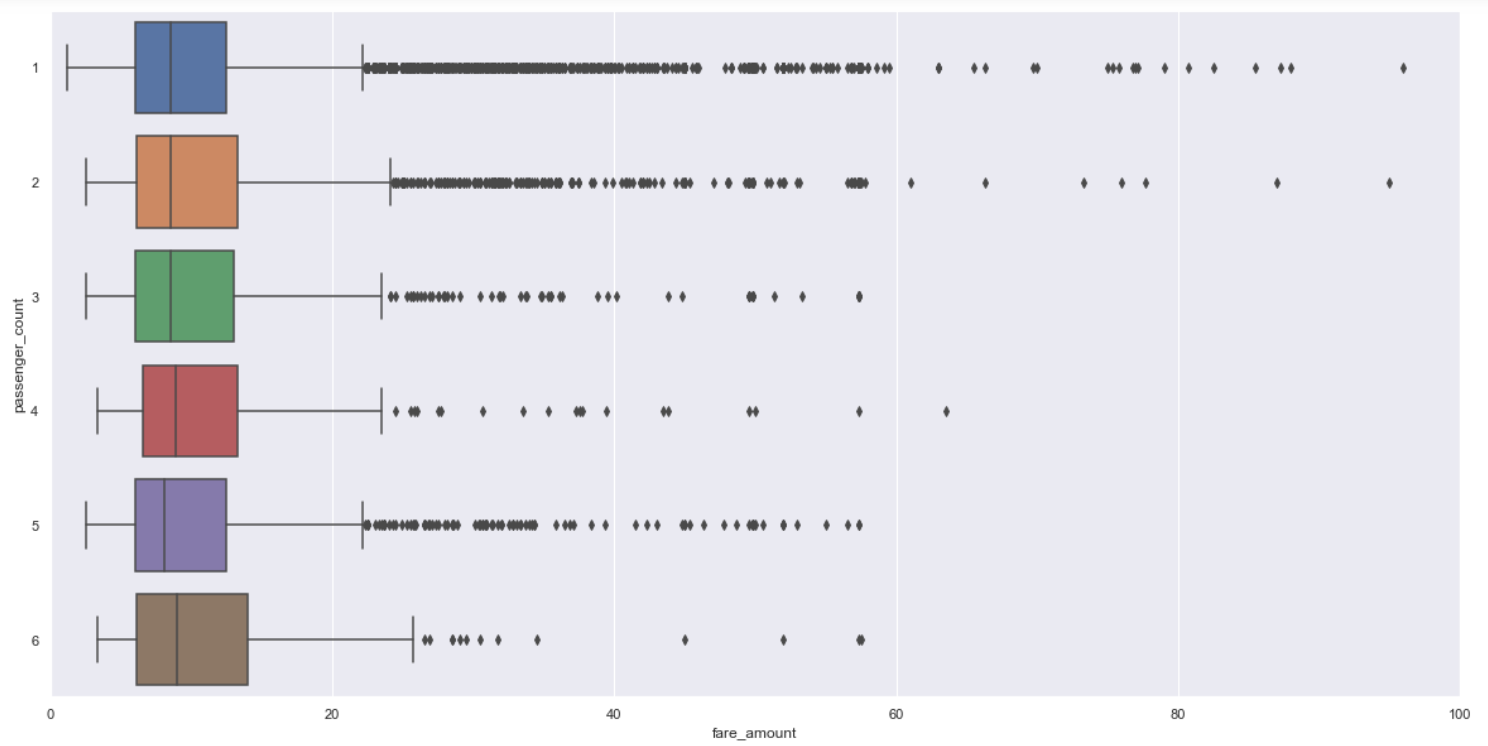


**Outlier Analysis –**

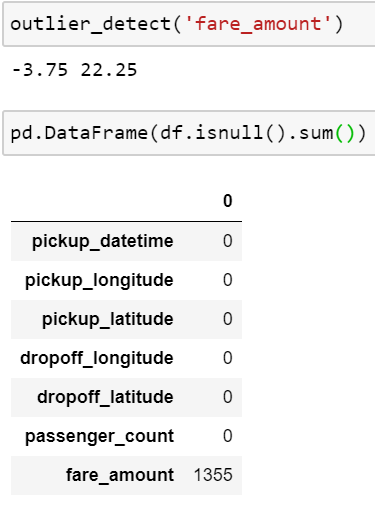
An outlier analysis was carried out using the Box-Plot graph on the continuous variable fare\_amount.

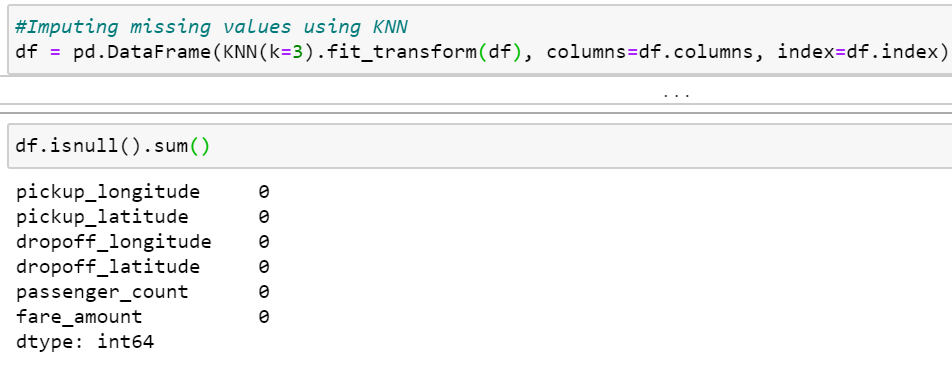


Another box-plot was plotted for fare\_amount vs passenger\_count in order to visualise the distribution of outliers for each passenger\_count category.



The outliers which were found using the outlier\_detect function were labelled as NaN and then imputed again using K-nearest neighbours (KNN).

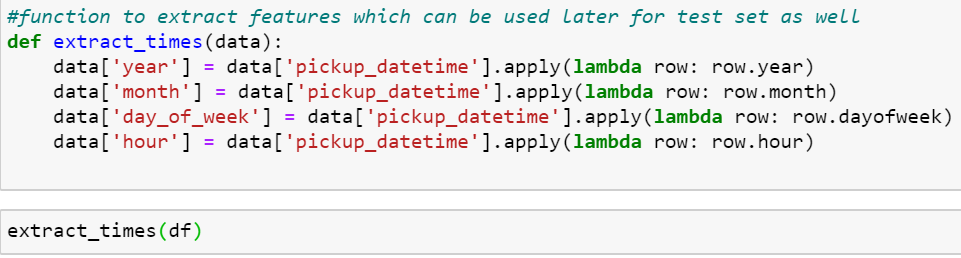




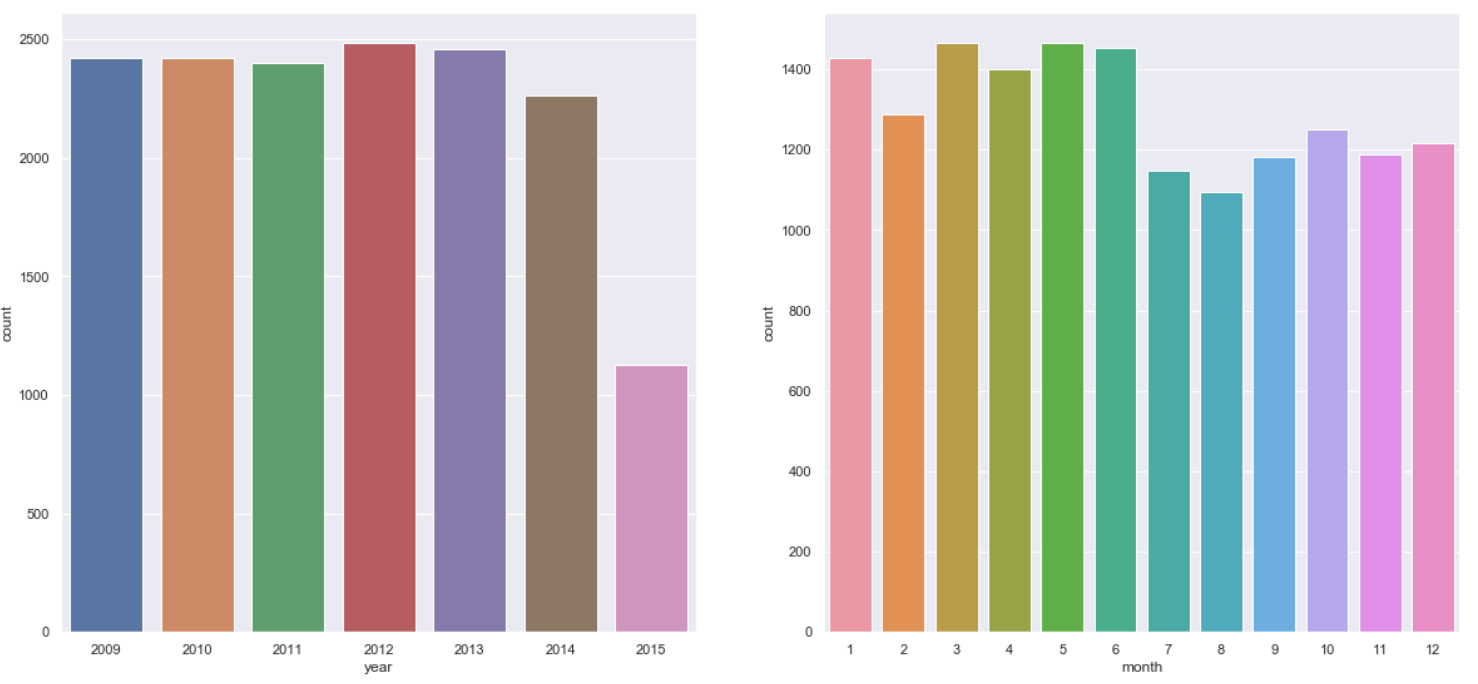
**Feature Engineering –**

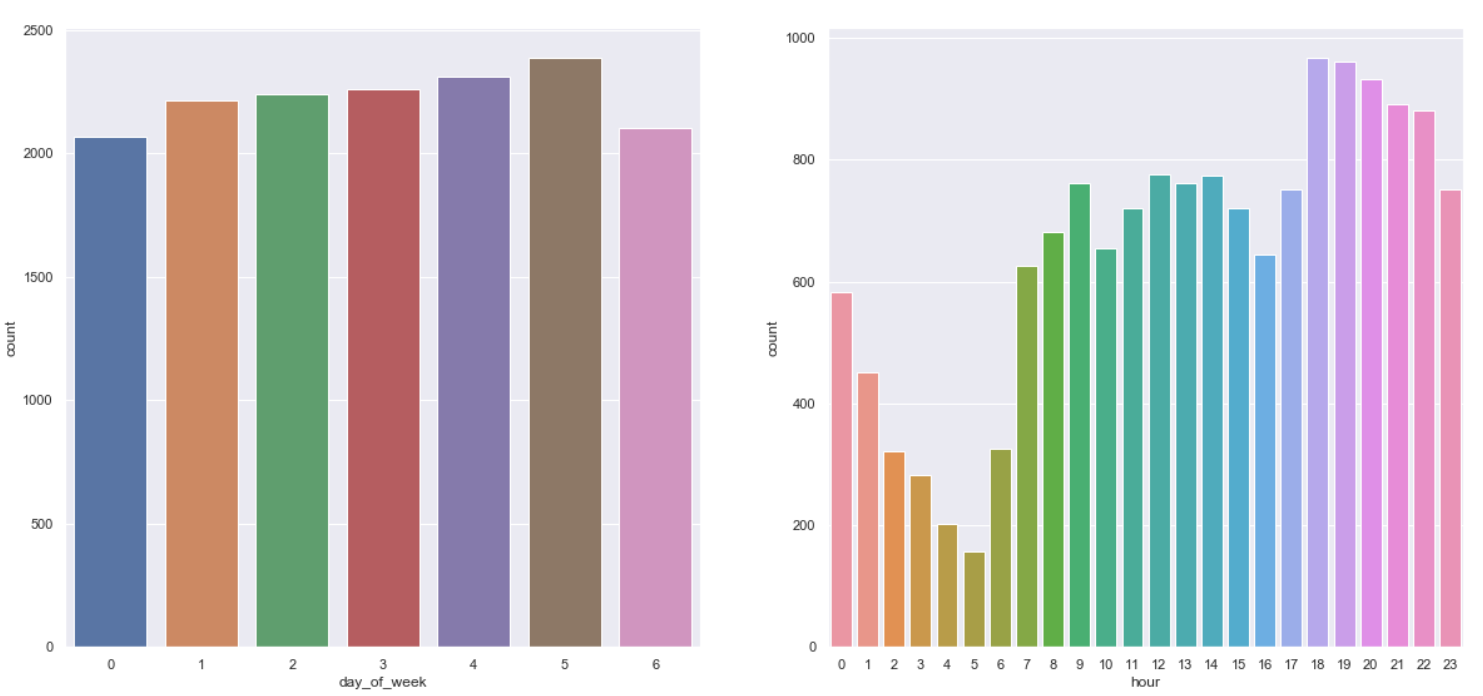
1. **Extracting relevant time information from the pickup\_datetime variable and dividing the data into year, month, day of the week, and hour.**

This procedure was carried out using the extract\_times function which extracts the year, month, day and hour of every row.



A count-plot visualisation was then used in order to get a better estimate of the distribution of the number of cab rides in a particular year, month, day of the week and hour respectively so that these categories could be further broken down into more simplistic data.



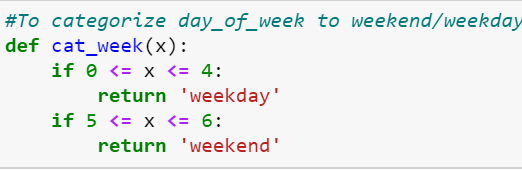


Thus, upon closely examining the result of the plot, it was decided that each of the three variables – month, day\_of\_week and hour were to be further subdivided as follows:

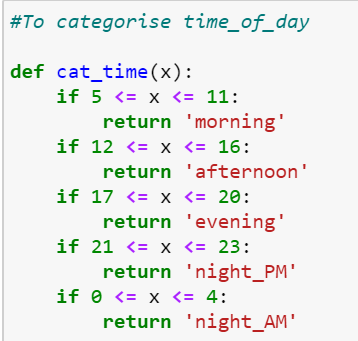
1. Month – To be divided into the new categorical variable “seasons”(summer, winter and monsoon)



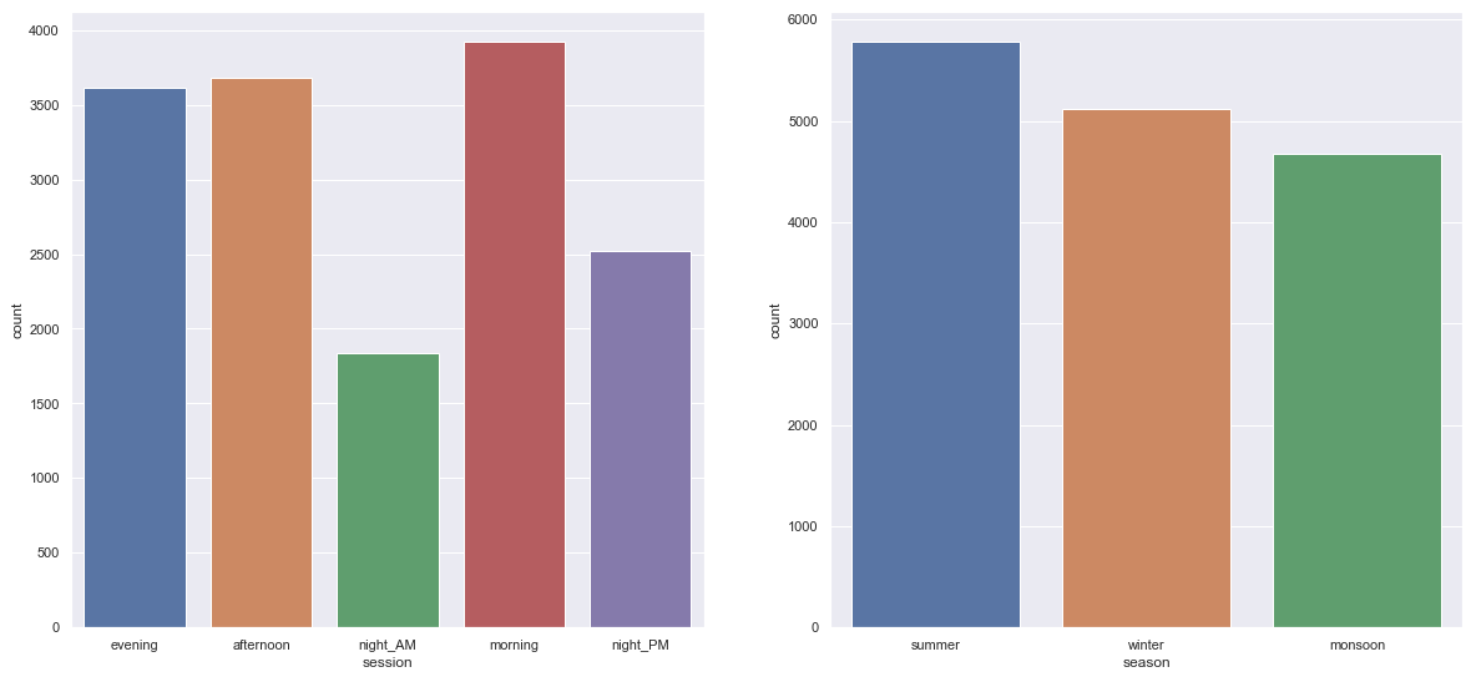
1. Day\_of\_week - To be divided into the new categorical variable “week”(weekdays and weekend)

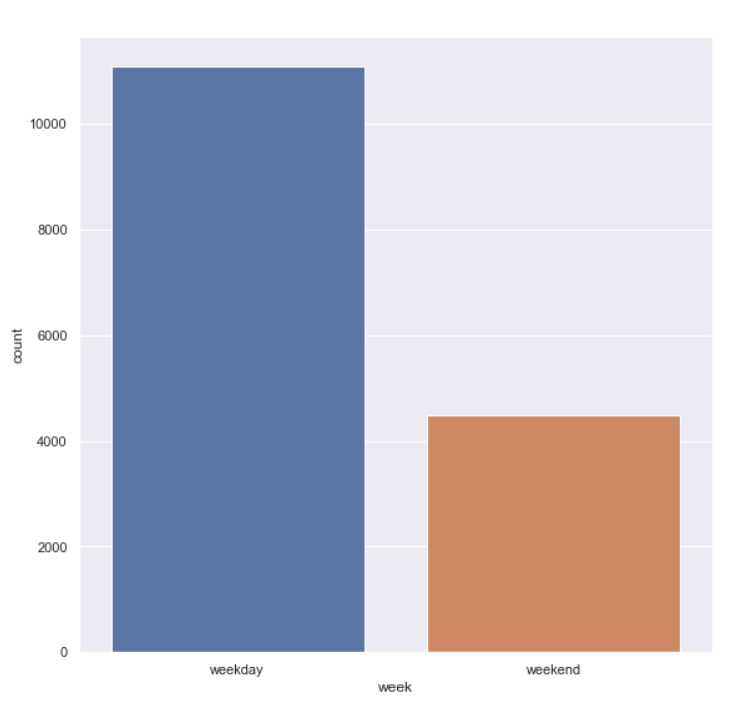


1. Hour – To be divided into the new categorical variable “session”(mornig, afternoon, evening, night\_am and night\_pm)



After the creation of the above mentioned new categorical variables, a count-plot was agaon used in order to see the distribution of the number of cab rides in each of the new categories.





As expected, the trend is that most of the public tends to hail a cab in the morning to reach their places of work hence, it was quite obvious to see that morning time to have the maximum number of cab rides.

The expected trend seems to be followed in the “week” variable as well as majority of the population tends to take a cab to get on weekdays, as the number of cab rides on weekends are considerably less which means travel through cabs on weekends are less as people prefer to stay at home and relax from a hectic week.

Whereas if we consider the seasons variable, the difference between the number of cab rides isn’t a lot apart from the reason that in summers more people tend to take cabs as its hot outside and they’d rather travel in an air-conditioned cabs rather than some other forms of public transport. But when we checkout the monsoon and winter seasons due to weather being close to chill or pleasant travelling via public transport is easier and would be preferred as its cheaper.

1. **Adding Dummy variables as a part of one-hot encoding:**

One -hot Encoding: For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.

In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

Thus, all the categorical variables I.e., passenger\_count, year, session, season, week had to be one-hot encoded.

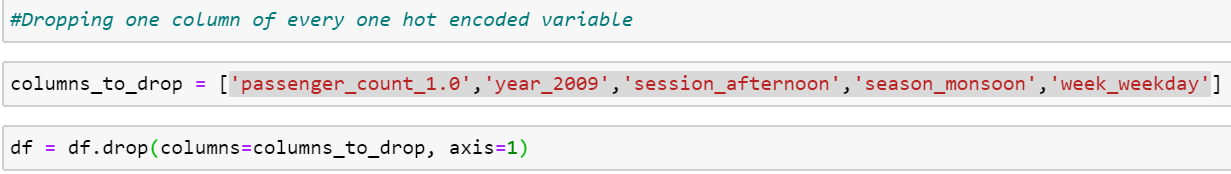




* Dropping one column of every one hot encoded category.

Reason: A common convention after one-hot encoding is to remove one of the one-hot encoded columns from each categorical feature. For example, the feature sex containing values of male and female are transformed into the columns sex\_male and sex\_female, each containing binary values. Because using either of these columns provides sufficient information to determine a person's sex, we can drop one of them.

Thus, we drop one column from every one hot encoded variable as follows:



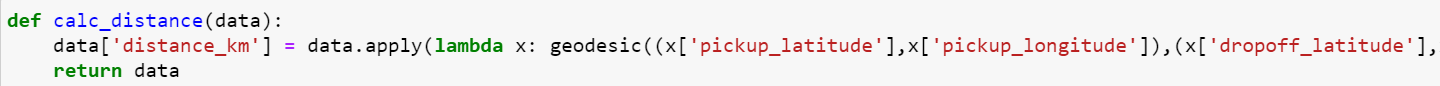
1. **Calculating distance travelled from the latitudes and longitudes**

We were given 2 sets of co-ordinates (pickup\_latitude,pickup\_longitude) and (dropoff\_latitude, dropoff\_longitude).

Thus, we can actually calculate the actual distance travelled in the cab from these co-ordinates. In this case, I’ve chosen the unit of measurement as Kilometers.

For calculation of the distance from the co-ordinates I have used the geodesic function from the geopy.distance library because:

Geopy package has many different methods for calculating distances, but it uses the [Vincenty’s formulae](https://en.wikipedia.org/wiki/Vincenty's_formulae) as default, which is a more exact way to calculate distances on earth since it takes into account that the earth is, as previously mentioned, an oblate spheroid.

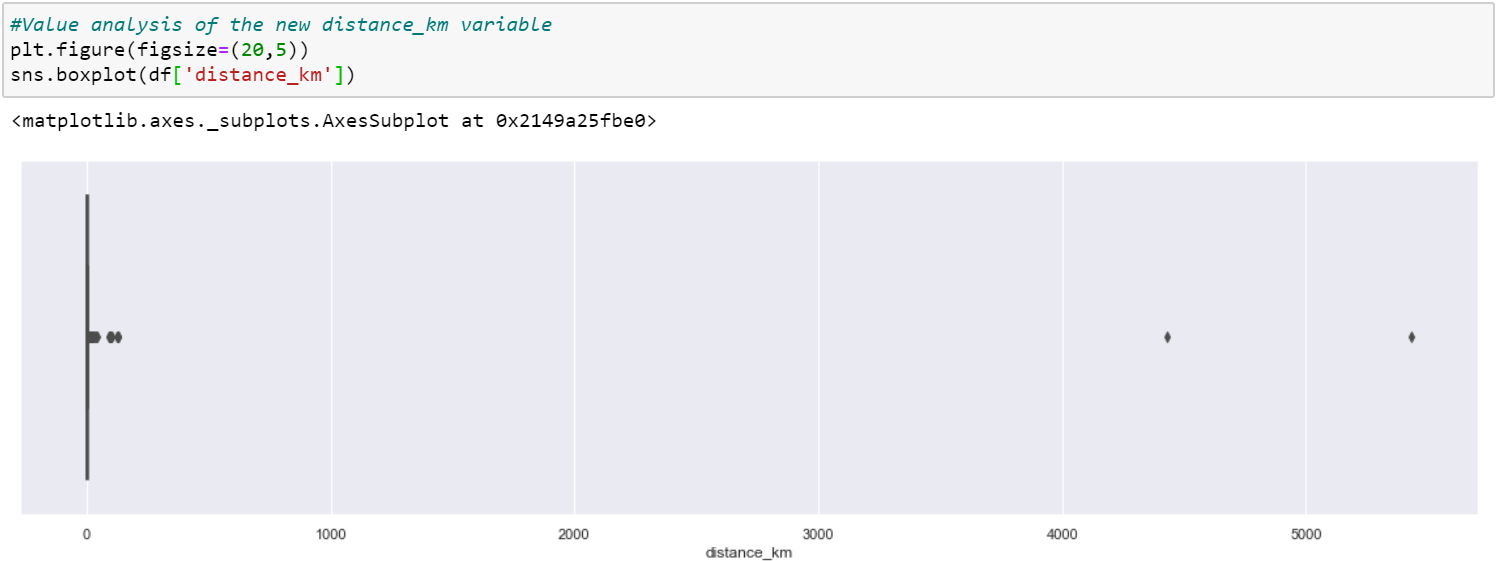






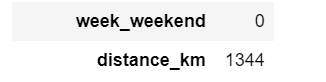
1. **The new calculated distance needs to be examined for outliers and need to be removed or re-imputed so that this doesn’t cause a bias in the model.**

* To analyse the values of the outliers we plot a box-plot of the new ‘distance\_km’



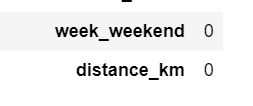
From the plot we can observe that there aren’t a lot of outliers, but a few of them have extremely large values and hence need to be treated.

Therefore, next in the next step I call the outlier\_detect function to detect the outliers and convert them into NaN

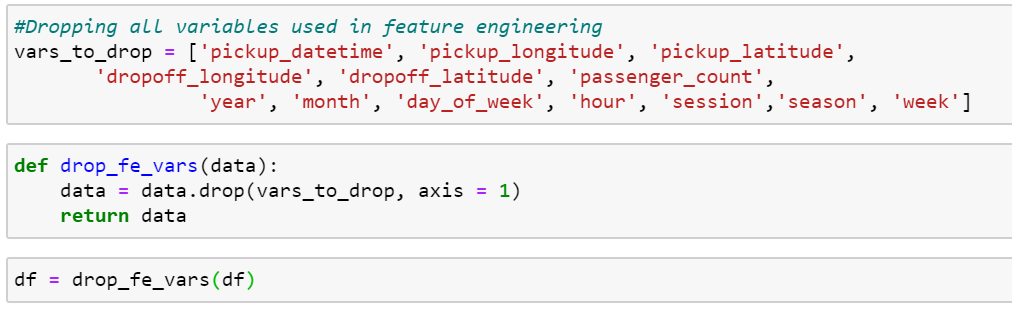


This indicates that there were a total of 1334 values that were outliers. These values were imputed again using the KNN method.





1. **After the relevant information was extracted from the given primary variables, those variables were no longer required and could be dropped so as to reduce the dimensionality of the data set.**



**Feature Selection –**

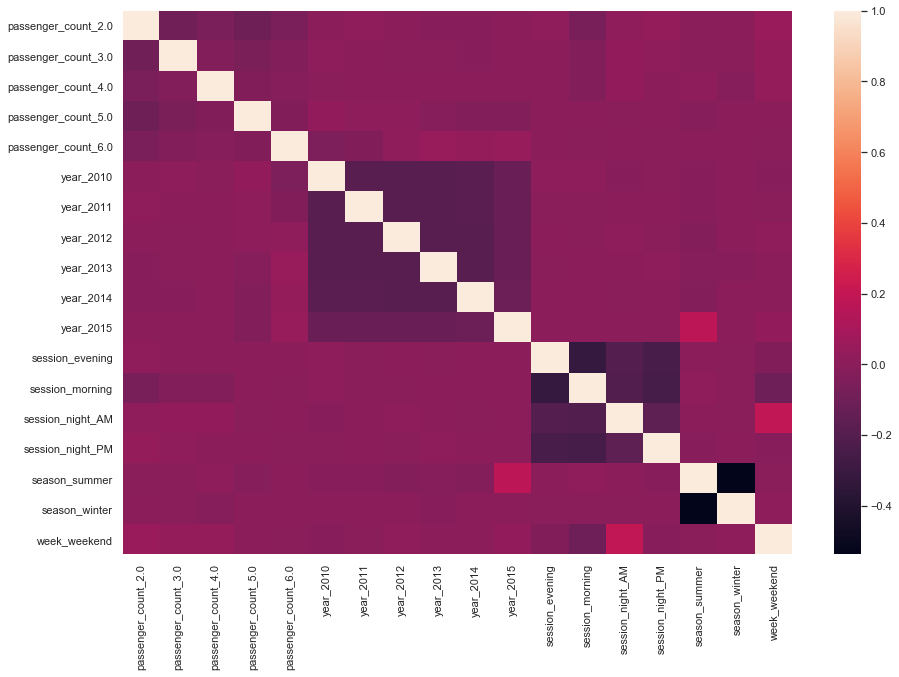
1. **Correlation Matrix:**

**Correlation** is a term that is a measure of the strength of a linear relationship between two quantitative variables.

Positive correlation is a relationship between two variables in which both variables move in the same direction. This is when one variable increases while the other increases and vice versa.

Whilst negative correlation is a relationship where one variable increases as the other decreases, and vice versa.

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.



From the above correlation matrix we observe that none of the variables are neither highly positively correlated nor highly negatively correlated.

1. **Variance Inflation Factor**

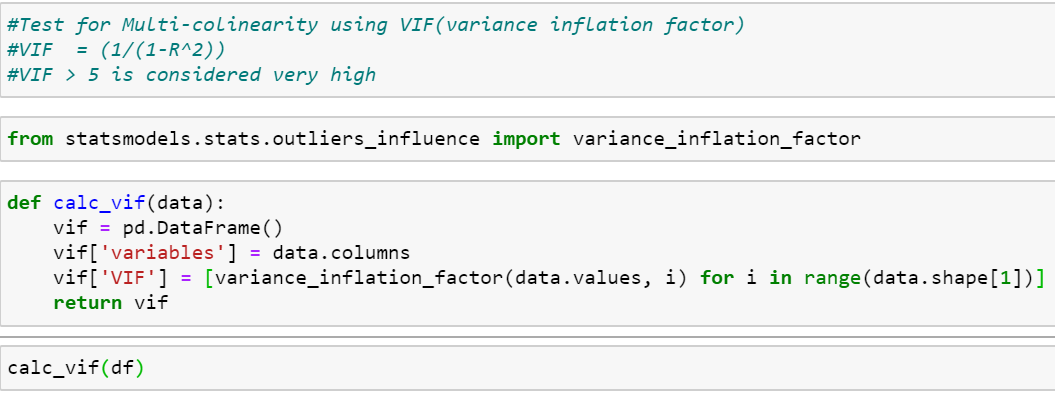
Multicollinearity: **Multicollinearity** refers to a situation in which two or more explanatory variables in a multiple regression model are highly linearly related.

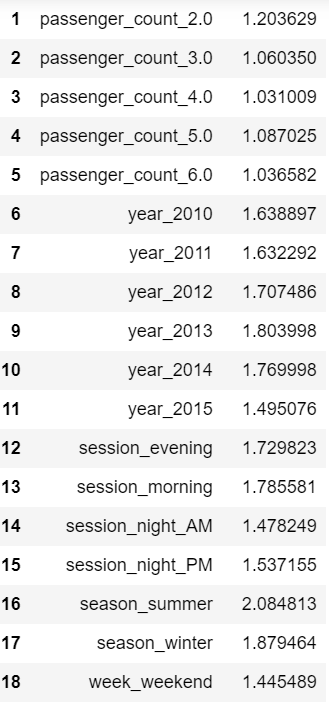
Problems of multicollinearity:

* The [coefficient](https://statisticsbyjim.com/glossary/regression-coefficient/) [estimates](https://statisticsbyjim.com/glossary/estimator/) can swing wildly based on which other independent variables are in the model. The [coefficients](https://statisticsbyjim.com/glossary/regression-coefficient/) become very sensitive to small changes in the model.
* Multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical [power](https://statisticsbyjim.com/glossary/power/) of your regression model. You might not be able to trust the p-values to identify independent variables that are statistically significant.

Variance inflation factor (VIF) is a measure of the amount of [multicollinearity](https://www.investopedia.com/terms/m/multicollinearity.asp) in a set of multiple [regression](https://www.investopedia.com/terms/r/regression.asp) variables. Mathematically, the VIF for a regression model variable is equal to the ratio of the overall model [variance](https://www.investopedia.com/terms/v/variance.asp) to the variance of a model that includes only that single independent variable. This ratio is calculated for each independent variable.

In VIF variables with a vif below 5 is acceptable, above 5 is considered very high and would have to be dropped.

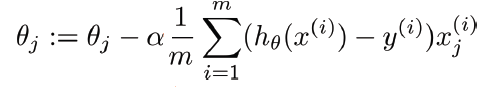




Since, none of the variables have extremely high VIF, therefore they need not be dropped.

**Feature Scaling –**

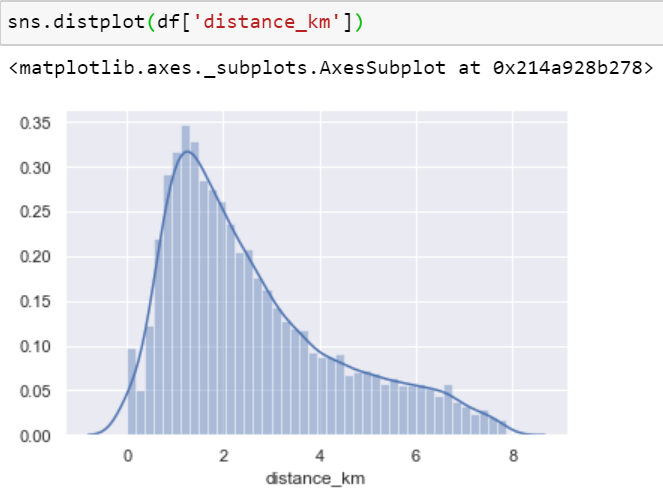
**Machine learning algorithms like**[**linear regression**](https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization)**,**[**logistic regression**](https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization)**,**[**neural network**](https://www.analyticsvidhya.com/blog/2017/05/neural-network-from-scratch-in-python-and-r/?utm_source=blog&utm_medium=feature-scaling-machine-learning-normalization-standardization)**, etc. that use gradient descent as an optimization technique require data to be scaled.** Take a look at the formula for gradient descent below:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/gradient-descent.png)

The presence of feature value X in the formula will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature. To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model.

Feature scaling has been applied on the continuous variable ‘distance\_km’ in order to scale the values.

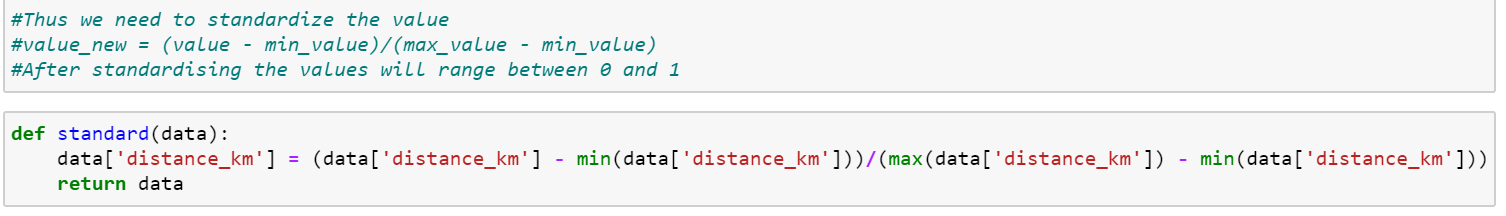
The ‘distance\_km’ variable is checked if it is distributed normally.



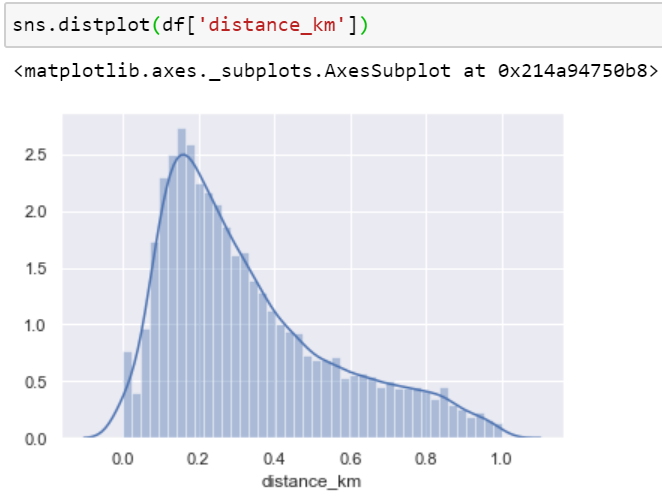
It is observed that the datapoints are not normally distributed and hence, we cannot use normalization.

The formula for standardization: X’ = (X – X\_min)/(X\_max – X\_min)

Therefore, we use the standardize method in order to scale the values between 0 and 1.



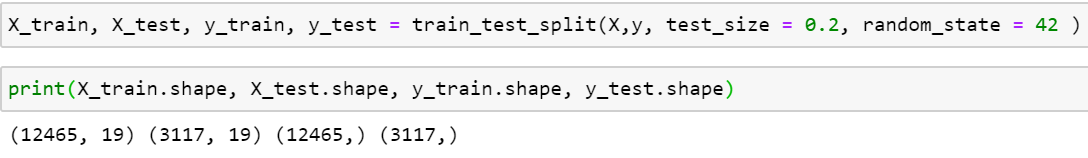
After standardizing the values, we get the following distribution of values.



**Model Selection –**

1. **Splitting the pre-processed ‘train.csv’ data set into train and test data set for evaluating different regression model before applying one of those models to the ‘test.csv’.**

The test size was taken as 0.2 or 20% of the train.csv dataset, with a random\_state = 42.



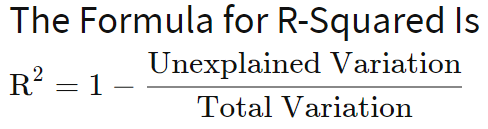
1. **Model evaluation metrics:**

The following metrics are the one that were used to evaluate various models.

**R-Squared –**

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a [regression](https://www.investopedia.com/terms/r/regression.asp)model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the R2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

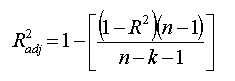
* R-Squared is a statistical measure of fit that indicates how much variation of a dependent variable is explained by the independent variable(s) in a regression model.
* In investing, R-squared is generally interpreted as the percentage of a fund or security's movements that can be explained by movements in a benchmark index.



**Adjusted R-Squared –**

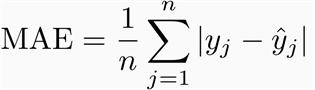
Adding more independent variables or predictors to a regression model tends to increase the R-squared value, which tempts makers of the model to add even more. This is called [overfitting](https://www.investopedia.com/terms/o/overfitting.asp) and can return an unwarranted high R-squared value. Adjusted R-squared is used to determine how reliable the correlation is and how much is determined by the addition of independent variables.

In a portfolio model that has more independent variables, adjusted R-squared will help determine how much of the correlation with the index is due to the addition of those variables. The adjusted R-squared compensates for the addition of variables and only increases if the new predictor enhances the model above what would be obtained by probability. Conversely, it will decrease when a predictor improves the model less than what is predicted by chance.



**Mean Absolute Error –**

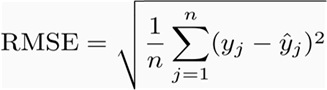
**Mean Absolute Error (MAE):**MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.



*If the absolute value is not taken (the signs of the errors are not removed), the average error becomes the Mean Bias Error (MBE) and is usually intended to measure average model bias. MBE can convey useful information, but should be interpreted cautiously because positive and negative errors will cancel out.*

**Root mean squared error (RMSE)**

RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and actual observation.

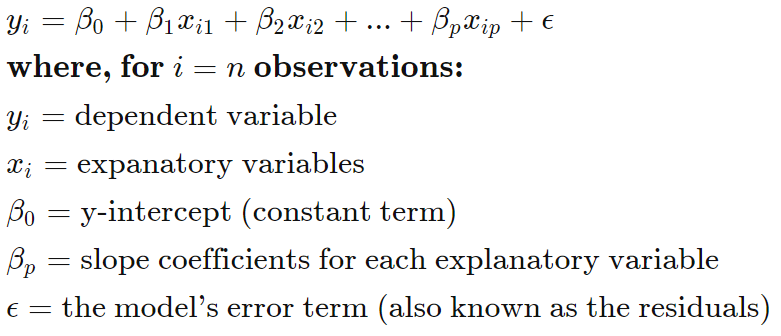


1. **Multiple Linear Regression –**

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the [linear relationship](https://www.investopedia.com/terms/l/linearrelationship.asp) between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) [regression](https://www.investopedia.com/terms/r/regression.asp) that involves more than one explanatory variable.

The Formula for Multiple Linear Regression Is



1. **Ridge Regression:**

ridge regression performs ‘**L2 regularization**‘, i.e. it adds a factor of sum of squares of coefficients in the optimization objective. Thus, ridge regression optimizes the following:

#### Objective = RSS + α \* (sum of square of coefficients)

Here, α (alpha) is the parameter which balances the amount of emphasis given to minimizing RSS vs minimizing sum of square of coefficients. α can take various values:

1. **α = 0:**
   * The objective becomes same as simple linear regression.
   * We’ll get the same coefficients as simple linear regression.
2. **α = ∞:**
   * The coefficients will be zero. Why? Because of infinite weightage on square of coefficients, anything less than zero will make the objective infinite.
3. **0 < α < ∞:**
   * The magnitude of α will decide the weightage given to different parts of objective.
   * The coefficients will be somewhere between 0 and ones for simple linear regression.

**3.Lasso Regression –**

LASSO stands for Least Absolute Shrinkage and Selection Operator.

Lasso regression performs **L1 regularization**, i.e. it adds a factor of sum of absolute value of coefficients in the optimization objective. Thus, lasso regression optimizes the following:

#### Objective = RSS + α \* (sum of absolute value of coefficients)

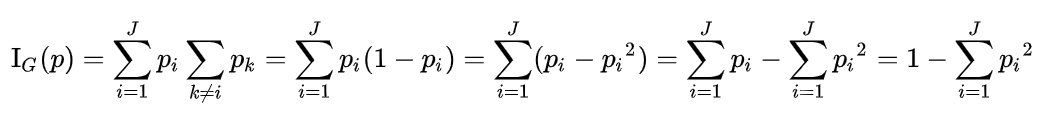
Here, α (alpha) works similar to that of ridge and provides a trade-off between balancing RSS and magnitude of coefficients. Like that of ridge, α can take various values. Lets iterate it here briefly:

1. α = 0: Same coefficients as simple linear regression
2. α = ∞: All coefficients zero (same logic as before)
3. 0 < α < ∞: coefficients between 0 and that of simple linear regression
4. **Decision Tree Regression –**

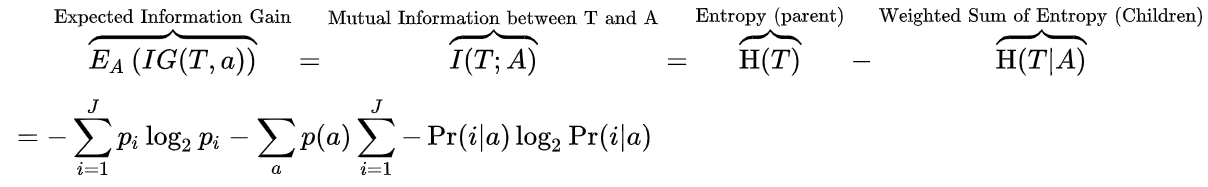
Decision tree regression is used for the continuous output problem. Continuous output means the output of the result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values. Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

Gini Impurity:{\displaystyle \operatorname {I} \_{G}(p)=\sum \_{i=1}^{J}p\_{i}\sum \_{k\neq i}p\_{k}=\sum \_{i=1}^{J}p\_{i}(1-p\_{i})=\sum \_{i=1}^{J}(p\_{i}-{p\_{i}}^{2})=\sum \_{i=1}^{J}p\_{i}-\sum \_{i=1}^{J}{p\_{i}}^{2}=1-\sum \_{i=1}^{J}{p\_{i}}^{2}}

Used by the CART (classification and regression tree) algorithm for classification trees, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity can be computed by summing the probability {\displaystyle p\_{i}} of an item with label {\displaystyle i} being chosen times the probability {\displaystyle \sum \_{k\neq i}p\_{k}=1-p\_{i}}pi of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.



Information Gain:



Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information which is measured in [bits](https://en.wikipedia.org/wiki/Bit). For each node of the tree, the information value "represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node".

1. **Random Forest Regressor –**

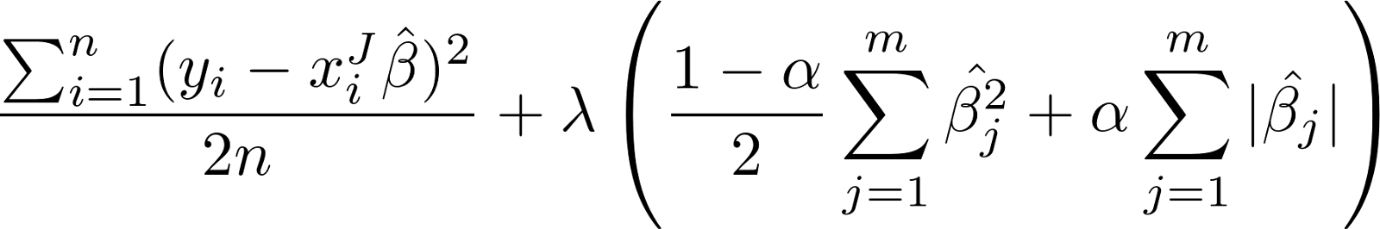
**Random forest** is a **bagging** technique and **not a boosting** technique. The trees in **random forests** are run in parallel. There is no interaction between these trees while building the trees.

It operates by constructing a multitude of decision trees at training time and outputting the class that is the **mode** of the **classes (classification)** or **mean prediction (regression)** of the individual trees.

A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which **aggregates many decision trees**, with some helpful modifications:

1. The number of features that can be split on at each node is limited to some percentage of the total (which is known as the **hyperparameter**). This ensures that the ensemble model **does not rely too heavily on any individual feature**, and makes **fair use of all potentially predictive features**.
2. Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents **overfitting**.
3. **Elastic-Net Regression –**

Elastic Net which incorporates penalties from both L1 and L2 regularization:



In addition to setting and choosing a lambda value elastic net also allows us to tune the alpha parameter where 𝞪 = 0 corresponds to ridge and 𝞪 = 1 to lasso. Simply put, if you plug in 0 for alpha, the penalty function reduces to the L1 (ridge) term and if we set alpha to 1 we get the L2 (lasso) term. Therefore we can choose an alpha value between 0 and 1 to optimize the elastic net. Effectively this will shrink some coefficients and set some to 0 for sparse selection.

**Hyperparameter Tuning using RandomSearchCV and GridSearchCV –**

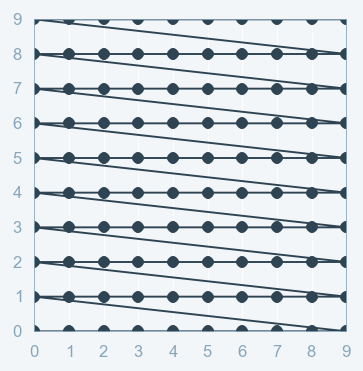
**Hyperparameter Tuning**is nothing but searching for the right set of hyperparameter to achieve high precision and accuracy. Optimising hyperparameters constitute one of the most trickiest part in building the machine learning models. The primary aim of hyperparameter tuning is to find the sweet spot for the model’s parameters so that a better performance is obtained.

There are several parameter tuning techniques, but in this article we shall focus on two of the most widely-used parameter optimising techniques:-

* Grid Search
* Random Search

# **Grid Search**

In Grid Search, we try every combination of a preset list of values of the hyper-parameters and evaluate the model for each combination. The pattern followed here is similar to the grid, where all the values are placed in the form of a matrix. Each set of parameters is taken into consideration and the accuracy is noted. Once all the combinations are evaluated, the model with the set of parameters which give the top accuracy is considered to be the best.

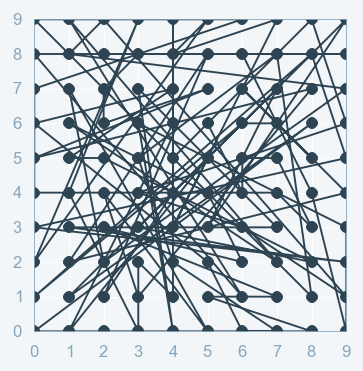


**Visual Representation of grid search**

One of the major drawbacks of grid search is that when it comes to dimensionality, it suffers when the number of hyperparameters grows exponentially. With as few as four parameters this problem can become impractical, because the number of evaluations required for this strategy increases exponentially with each additional parameter, due to the curse of dimensionality.

# **Random Search**

Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. It tries random combinations of a range of values. To optimise with random search, the function is evaluated at some number of random configurations in the parameter space.



**Visual Representation of Random search**

The chances of finding the optimal parameter are comparatively higher in random search because of the random search pattern where the model might end up being trained on the optimised parameters without any aliasing. Random search works best for lower dimensional data since the time taken to find the right set is less with less number of iterations. Random search is the best parameter search technique when there are less number of dimensions.

**Model metrics comparison –**

|  |
| --- |
| **Training Data** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **R-squared** | **Adjusted R-Squared** | **Absolute Mean Error** | **Root Mean Squared Error** |
| **Linear Regression** | 0.7303591840703934 | 0.7299475187025619 | 1.645240724370096 | 2.3112582991201482 |
| **Ridge Regression** | 0.7303589847053527 | 0.7299473190331471 | 1.6454323787008445 | 2.3112591535605604 |
| **Lasso Regression** | 0.729097417526163 | 0.7286838257971953 | 1.6549910388439353 | 2.316659678588732 |
| **Decision Tree Regression** | 0.7419685921122552 | 0.7415746510315105 | 1.6226724706188556 | 2.260955180491699 |
| **Random Forest Regression** | 0.7856980754983974 | 0.785370896987708 | 1.4766137098135521 | 2.0604814207639035 |
| **Elastic-Net Regression** | 0.6801533142540983 | 0.6796649987033412 | 1.8005340605586004 | 2.5172514213178867 |

|  |
| --- |
| **Testing Data** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **R-squared** | **Adjusted R-Squared** | **Absolute Mean Error** | **Root Mean Squared Error** |
| **Linear Regression** | 0.7388385859788489 | 0.7372363687149155 | 1.678211850892539 | 2.3445156624603376 |
| **Ridge Regression** | 0.7388200544551927 | 0.7372177235009301 | 1.6784706772657927 | 2.3445988421997987 |
| **Lasso Regression** | 0.7374672613041283 | 0.7358566310053807 | 1.6877074326989459 | 2.350662976999091 |
| **Decision Tree Regression** | 0.7335859670610554 | 0.7319515251411846 | 1.692802978866932 | 2.367975371527145 |
| **Random Forest Regression** | 0.7474434544966244 | 0.7458940278370945 | 1.647020473124045 | 2.3055680596948984 |
| **Elastic-Net Regression** | 0.6899074325321388 | 0.6880050241427654 | 1.8405915210050163 | 2.5547259011626773 |

From the metrics compared, it can be observed that the Random Forest Regression model fits this particular dataset the best as compared to the other regression models.

Therefore, the random forest regression model was used to predict the data of the test.csv dataset and was saved as ‘predicted\_test\_fare.csv’.

How to Run the code:

For Python:

* 1. Ensure that Jupyter notebooks have been installed on the system along with anaconda distribution.
  2. Download and save the file in any location on the system.
  3. Open the command prompt and change the current directory location to where the notebook has been saved on the system.
  4. After changing the location, type “jupyter notebook” in the command prompt.
  5. After a while, the jupter notebook should open up.
  6. Click on “Project2.ipynb” file.
  7. When the file opens, on the menu bar at the top, click on “Cell” and the “Run All”.
  8. This should run all the cell, the file includes pre-processing of the data with the help of charts and statistics, then runs the chosen models and gives out the accuracy and false negative rates along with the Gain and Lift charts.

For R:

1. Ensure R-studios has been downloaded and updated on the system.
2. Download and save the entire folder to any location on the system.
3. Open R-Studios, “open file”, select the destination of where the file has been saved, find the file and click on open.

(In the second line of code where setwd() is mentioned, enter the path at which the entire file was downloaded and kept for eg “setwd("C:/Users/Gaurav/Desktop/Project 2")” )

1. To run the entire code, select “Code”, then select “Run Region” and then “Run All”.
2. All the variables will then be run and stored in the R environment along with all the model results such as accuracy and false negative rate.