**CAB FARE PREDICTION**

**Haythem Gaja**

**2 Feb 2020**

**Contents**

**1 Introduction** **3**

1.1 Problem Statement……………………………………………………………………………..3

1.2 Data…………………………………………………………………………………………….3

**2 Methodology** **5**

2.1 Pre-Processing………………………………………………………………………………… 5

2.1.1 Missing Value Analysis……………………………………………………………………5

2.1.2 Outlier Analysis……………………………………………………………………………6

2.1.3 Feature Selection………………………………………………………………………….10

2.1.4 Feature Scaling……………………………………………………………………………12

2.2 Modelling………………………………………………………………………………………14

2.2.1 Model Selection…………………………………………………………………………...14

2.2.2 Model Evaluation………………………………………………………………………….15

2.2.3 Linear Regression...…………………………………………………………………….…17

2.2.3.1 Implementation...……………………………………………………………………....17

2.2.4 Decision Tree……………………………………………………………………………....19

2.2.4.1 Implementation………………………………………………………………………...20

2.2.5 Random Forest …………………………………………………………………………….21

2.2.5.1 Implementation….………………………………………………………………….…22

**3 Conclusion** **24**

3.1 Model Selection……………………………………………………………………………….24

3.2 Output………………………………………………………………………………………….25

**References**  **25**

**Chapter 1**

**Introduction**

* 1. **Problem Statement**

A cab rental start-up company has successfully run the pilot project and now they want to launch a cab service across the country. A historical data has been collected from a pilot project and data analyses required for fare prediction. A model needs to be design that predicts the fare amount for a cab ride in the city.

* 1. **Data**

In this project, the task is to predict the fare amount for a cab ride in the city based on the historical data. The data given below is a sample from the whole population which is used to predict the fare amount for a cab ride:

**Table 1.1 Cab Fare Prediction Sample Training Data (Columns1-7)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **fare\_amount** | **pickup\_datetime** | **pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** | **dropoff\_latitude** | **passenger\_count** |
| 4.5 | 2009-06-15 17:26:21 UTC | -73.844311 | 40.721319 | -73.84161 | 40.712278 | 1 |
| 16.9 | 2010-01-05 16:52:16 UTC | -74.016048 | 40.711303 | -73.979268 | 40.782004 | 1 |
| 5.7 | 2011-08-18 00:35:00 UTC | -73.982738 | 40.76127 | -73.991242 | 40.750562 | 2 |
| 7.7 | 2012-04-21 04:30:42 UTC | -73.98713 | 40.733143 | -73.991567 | 40.758092 | 1 |
| 5.3 | 2010-03-09 07:51:00 UTC | -73.968095 | 40.768008 | -73.956655 | 40.783762 | 1 |

From the given data (**Training dataset)**, the fare amount will be predicted’, in the **test dataset**, using the pickup timestamp, latitude and longitude details and the data on total number of passengers travelling. The complete structure of the data is as follows:

**Table 1.2 Cab Fare Prediction Dataset Structure**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Predictor/Response** |
| **pickup\_datetime** | Timestamp value indicating when the cab ride started in GMT | Predictor |
| **pickup\_longitude** | Float value of longitude coordinate of where the cab ride started. | Predictor |
| **pickup\_latitude** | Float value of latitude coordinate of where the cab ride started. | Predictor |
| **dropoff\_longitude** | Float value of longitude coordinate of where the cab ride ended. | Predictor |
| **dropoff\_latitude** | Float value of latitude coordinate of where the cab ride ended. | Predictor |
| **passenger\_count** | An integer indicating the number of passengers in the cab ride. | Predictor |
| **fare\_amount** | The total cost of cab fare of the trip | Response |

Instead of using these variables directly to predict the fare amount., few other information from these variables will be driven to predict the fare amount effectively. Generally, the fare amount varies depending on various factors, it depends on the distance covered and the number of passengers.

Using the latitude and longitude details, the **distance covered** can be calculated and it can be used for effectively predicting the fare amount. Additionally, the time and date can be separated from the timestamp date. These can be seen in the next data pre-processing section.

**Chapter 2**

**Methodology**

* 1. **Pre-Processing**

Before entering the modelling phase, initially the data must be analysed. The data need to be cleaned and select the required variables for modelling and analyse the data to impute the missing values; detect and handle the outliers; normalise the data under the same scale for giving equal weightage for all variables. For performing these tasks, the data will be visualized for easy processing using plots and graphs. This process is known as **Exploratory Data Analysis (EDA).** In this project, response (dependent) variable is a numerical variable and the regressing modelling can be used to predict it. For regression, the data must usually be normally distributed. In the following pages, more details will be given about the various pre-processing techniques carried out in this project.

* + 1. **Missing Value Analysis**

Missing values in data is a common phenomenon in real world problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models. Durng analysing the data, a few missing values were found in all the variables except pickup\_datetime. It has been represented in a tabulated format as follows:

**Table 2.1 Before Missing Value Analysis**

|  |  |  |
| --- | --- | --- |
|  | **Variables** | **Missing\_percentage** |
| **0** | **pickup\_longitude** | **1.960540** |
| **1** | **pickup\_latitude** | **1.960540** |
| **2** | **dropoff\_longitude** | **1.954316** |
| **3** | **dropoff\_latitude** | **1.941868** |
| **4** | **passenger\_count** | **0.697081** |
| **5** | **fare\_amount** | **0.161822** |
| **6** | **pickup\_datetime** | **0.000000** |

Generally, the missing value analysis is performed and the variables which are having missing values greater than 30% are dropped as per industry standards. Here as can be seen, only less than 2% of data are missing in all columns. these rows will be dropped.

* + 1. **Outlier Analysis**

The next pre-processing technique that usually being carry out is the **Outlier Analysis**. Since the data is free of missing values, the outlier can be defined as a data point that is unusually different when compared to the remaining data of the variable. These Outliers need to be handled before the modelling phase start. The outliers may make the model biased to a particular variable and alter the output.

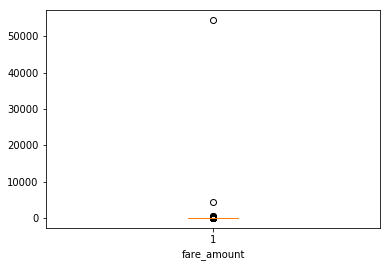
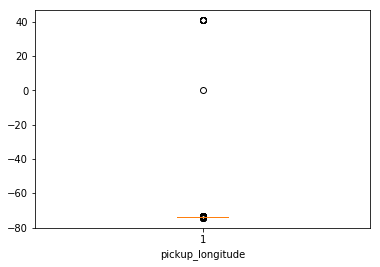
Here, the outliers will be handled based on the following steps:

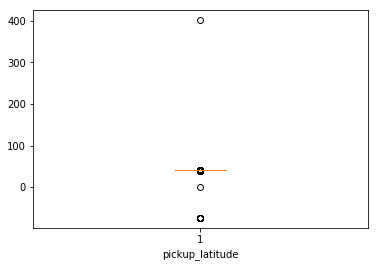
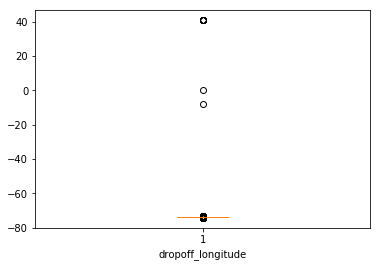
* create boxplots to check for outliers.
* check the correlation between the independent and dependent variables before and after the removal of outliers and handle them accordingly.

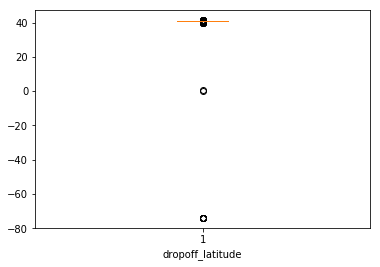
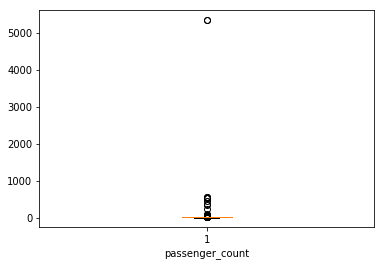
**Note**: Outlier Analysis need to be carried out carefully and it must be made sure that no useful information gets removed due to this.

Now, check the **box and whisker plot** for these variables.

**Fig 2.1 Outlier Analysis - Box and Whisker**

Figures 2.1 is the Python plots using matplotlib library.

**Outliers**

fare\_amount 1397

pickup\_longitude 831

pickup\_latitude 532

dropoff\_longitude 945

dropoff\_latitude 778

passenger\_count 1703

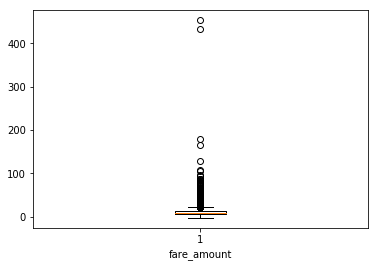
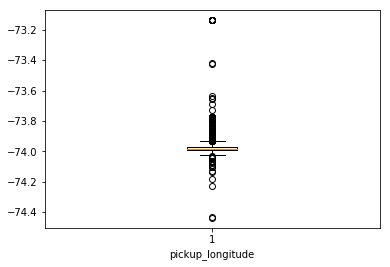
From the box and whisker plots, the majority of data points are very close to the whisker range and there are only very few points that are very far from the range. these outliers should be handled with care and make sure that none of the useful information is removed.

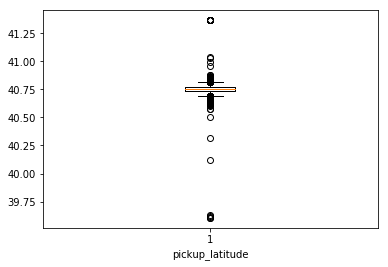
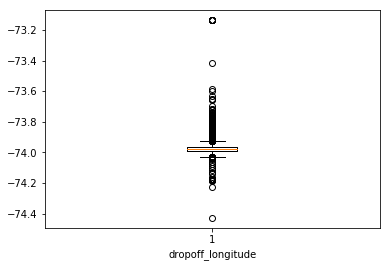
In the variables, **dropoff\_latitude, pickup\_longitude, pickup\_latitude, dropoff\_longitude, fare\_amount,** a few data points can be omited fwhich are far from the complete range based on the box and whisker plots.

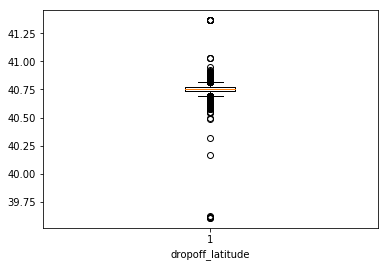
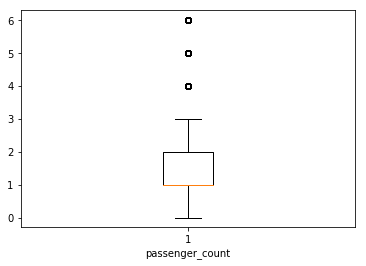
Generally, in a cab, maximum 7 passengers can be boarded at a time. So those values beyond 7 can be omitted in **'passenger\_count'** variable. Also any non-integer passenger number is omitted.

After imputation of outlier points, the remaining data can be used and analyized.

**Fig 2.2 Box and Whisker Plot(After Outlier analysis)**

At this point, instead of handling outliers in the latitude and longitude points, the latitude and longitude points were converted into distance travelled.

For calculating the distance travelled, **‘Haversine formula’** will be used, It gives approximate distances between the latitude and longitude points (**distance travelled)**.

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

even the time, day, month and year data are fetched from the **pickup datetime** variable and the latitude and longitude and timestamp variables are dropped from the dataset.

Now, the final dataset looks like this:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **fare\_amount** | **distance\_travelled** | **passenger\_count** | **Year** | **Month** | **Day** | **Hour** | **Time** |
| **0** | 4.5 | 1.030764 | 1 | 2009 | 6 | 15 | 17.0 | 17:26:21 |
| **1** | 16.9 | 4.827250 | 1 | 2010 | 1 | 5 | 16.0 | 16:52:16 |
| **2** | 5.7 | 1.389525 | 2 | 2011 | 8 | 18 | 0.0 | 00:35:00 |
| **3** | 7.7 | 2.799270 | 1 | 2012 | 4 | 21 | 4.0 | 04:30:42 |
| **4** | 5.3 | 1.999157 | 1 | 2010 | 3 | 9 | 7.0 | 07:51:00 |

Now again outlier analysis is carried out in **fare\_amount** and **distance\_travelled** variables and the final box and whisker plot after outlier analysis is as follows:

**Fig 2.3 Final Box and Whisker Plot(After Outlier analysis)**

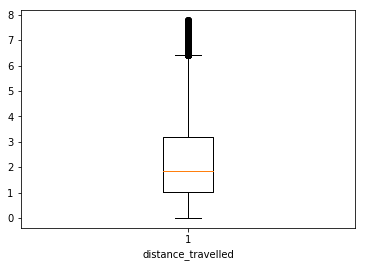
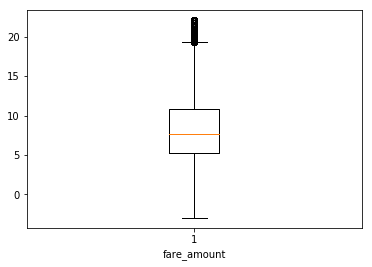


Figure 2.3 is the Python plot using matplotlib library.

After dropping negative values from fare amount, the outliers in 'fare\_amount' and 'distance\_travelled' have been considerably reduced and brought close to the whiskers of the plot. Hence these outliers are not again treated, and they are considered as part of data as they wont bias the model.

* + 1. **Feature Selection**

The noise in the data is important when the number of variables is very large. only the necessary number of variables for creating an algorithm should be used, or else, it might lead to **‘overfitting’**. Overfitting is a modelling error which occurs when a function is too closely fit to a limited set of data points.

the following criteria should be considered when selecting the variables for model:

* The correlation between two independent variables should be less and
* The correlation between independent and target variables should be high.

Initially, the correlation matrix will be formed, then checking the correlation between variables using the heatmap as follows.

**Fig 2.4 Heatmap**

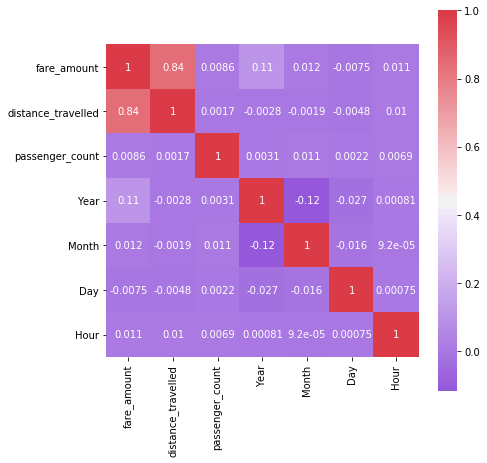
****

Figure 2.4 is the Python plot using **seaborn** library.

From the heatmap, the following can be suggested:

* the correlation between Independent and Target variables should be high (i.e., away from zero). keeping a cut-off region of (-0.25 to 0.25). Under this criterion, only **‘distance\_travelled’** variable gets selected as it has high correlation with **‘fare\_amount’.**
* Additionally, the correlation between two Independent variables should be low (i.e., close to zero). keeping a cut-off region (>0.5 and <-0.5). The independent variables having a correlation in this region should be removed while going into modelling phase, as one among the two variables is enough to predict the target variable. This will reduce **overfitting.** Under thiscriteria, none fall under this category.

But, as the variables, ‘passenger\_count’,’Year’,’Month’,’Day’, ‘Hour’ ave limited number of values, they can be considered categorical and when independent variables are categorical and dependent variable is numeric, **ANOVA** testing for correlation is preferred.

**ANOVA** stands for **Analysis of variance**. It is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

A null hypothesis is described as if no dependency exists between two variables. Based on ANOVA test, F-statistic value and p-value are determined to be 0.05, then if **p-value < 0.05, then null hypothesis is rejected and so dependency exists between the two variables.** p-value must be close to zero and F-statistic value must be higher for dependency to occur between two variables.

**p-value**

|  |  |
| --- | --- |
| ‘Year’ | 0.00000000e+00 |
| ‘Month’ | 1.31281529e-05 |

Based on ANOVA test**, ‘Year’ and ‘Month’** also prove to be having dependency with the **‘fare\_amount’** and so they are also considered going forward.

As part of **Dimensionality Reduction**, the dataset is proceeded with the variables, **‘distance\_travelled’, ‘Year’, ‘Month’, ‘fare\_amount’** to the next pre-processing section.

* + 1. **Feature Scaling**

**Feature Scaling** typically means to bring all the variable data under a common scale so that none of the variables overweigh during modelling phase. **For example:** If a column value ranges between 0 to 5 and another column value ranges between 1 million to 10 million, then the second variable will overweigh the model during modelling phase. to reduce this effect, **Feature Scaling** is carried out. Generally, Normalization and Standardization are the two types of Feature Scaling methods.

**Normalization** typically means that the range of values are **normalized** to be from 0.0 to 1.0.

**Standardization** typically means that the range of values are **standardized** to measure how many standard deviations the value is from its mean.

**Normalization:**  Xchanged=X−Xmin / (Xmax−Xmin)

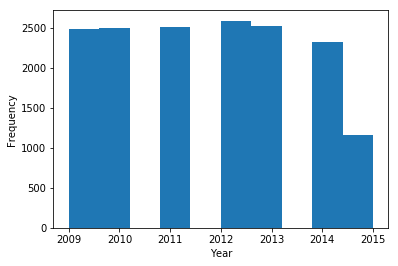
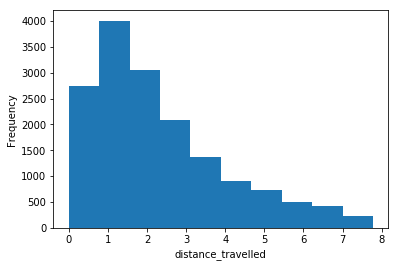
**Standardization:** Xchanged=X−μ / σ

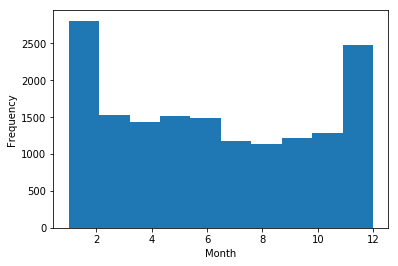
Normally, when the given **data set is uniformly distributed** (i.e., it forms a bell-shaped curve when plotted), **standardization** is used, as it will be easier for us to measure, how many standard deviations away, the particular value is located.

But, when the data set is left skewed or right skewed or **ununiformly distributed** when plotted, we use **Normalization** method, as standardization won’t be so efficient as Normalization here due to accumulation of huge data at a particular area.

after Feature Selection; the variables, **‘distance\_travelled’, ‘Year’** and **‘Month’** are to be Normalized as it will be easier for evaluation of models during Modelling phase as all the metrics will be brought under common scale. As the data is not distributed perfectly in uniform, we choose Normalization. The corresponding histograms are again given for reference.

**Fig 2.5(A) Histograms – Feature Scaling**





**Fig 2.5(B) Data after pre-processing**

After normalization, the whole of the data will be scaled according to their corresponding ranges. A sample of it is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **distance\_travelled** | **Year** | **Month** | **fare\_amount** |
| **0** | 0.132538 | 0.000000 | 0.454545 | 4.5 |
| **1** | 0.620698 | 0.166667 | 0.000000 | 16.9 |
| **2** | 0.178668 | 0.333333 | 0.636364 | 5.7 |
| **3** | 0.359936 | 0.500000 | 0.272727 | 7.7 |
| **4** | 0.257056 | 0.166667 | 0.181818 | 5.3 |

At this moment, the data have undergone various pre-processing techniques such as **Missing Value Analysis, Outlier Analysis, Feature Selection** and **Feature Scaling.** These pre-processing techniques have made it possible to increase the accuracy and reduce the inefficiency of model due to noise (errors) in the data. Now, from the raw data, it has come to a point where the data is ready to be fed into a machine learning model and train it to make predictions in the future using the same model.

* 1. **Modelling**

The next phase in this project is the Modelling phase. Till now, the data is pre-processed, and it is made ready to train the model and make predictions. which model to be used based on the dataset.

* + 1. **Model Selection**

For model development, dependent variable may fall under one of the below categories

* Nominal
* Ordinal
* Interval
* Ratio

In this project, the dependent variable falls under **interval** category. the predictive analysis that can be performed is **Regression Analysis**. There are various algorithms and statistical models for such regression problems. Here the following models will be used for predicting cab fare for a journey.

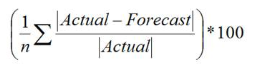
Here are the Regression Algorithms in the order from simple to complex models as follows:

* Linear regression
* Decision Tree
* Random Forest
  + 1. **Model Evaluation**

Generally, for classification problems, the dependent variable has only two outcomes as ‘Yes’ or ‘No’. In such cases, the outcomes can be classified, and the model evaluated by forming the confusion matrix.

But, in regression problems, the dependent variable is continuous and confusion matrix can’t be used in such cases. some other error metrics should be utilized to evaluate the model performance. Some of the commonly used error metrics in Regression Problems are **Mean absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Square Error (MSE), Root Mean Square Error (RMSE), R-Squared and Adjusted R\_ Squared.** Here, two of these Error Metrics, **MAPE** and **RMSE will be used** for evaluating our models.

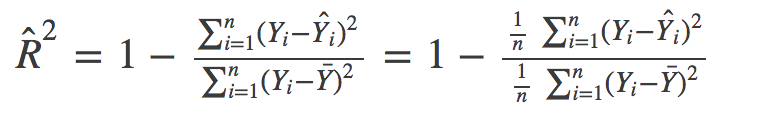
**Mean Absolute Percentage Error (MAPE)** is commonly used as a loss function for regression problems and in model evaluation, because of its very intuitive interpretation in terms of relative error. The MAPE measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error, as shown in the example below:



**Root Mean Square Error (RMSE)** is the standard deviation of the [residuals](https://www.statisticshowto.datasciencecentral.com/residual/) (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the [line of best fit](https://www.statisticshowto.datasciencecentral.com/line-of-best-fit/).

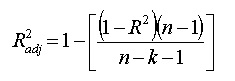


**R Squared & Adjusted R Squared** are often used for explanatory purposes and explains how well our selected independent variable(s) explain the variability in our dependent variable(s). Mathematically, R-Squared is given by:



The numerator is MSE (average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, **smaller the R\_squared and poorer is the model.**

Just like R², adjusted R² also shows how well terms fit a curve or line but adjusts for the number of terms in a model. It is given by below formula:



where n is the total number of observations and k is the number of predictors. Adjusted R² will always be less than or equal to R²

An adjusted R² will consider the marginal improvement added by an additional term in our model. So, it will increase if we add the useful terms and it will decrease if we add less useful predictors. However, R² increases with increasing terms even though the model is not actually improving.

Basically, the **MAE and MAPE won’t be affected by outliers**, but **MSE and RMSE will be affected by outliers**. Here in this dataset there are some outliers, but they are **useful data** points. using RMSE, it will be better when compared to MAPE or MAE as, RMSE will give more weightage to the outlier points as it is squared.

On the other hand, **R-Squared and Adjusted R-Squared** values describe how well the independent variables are capable of predicting the dependent variables. It alone gives the power of predictability of model. It ranges from **-infinity to 1**; where 1 is the best and close to 0 and negative values are worse. They can be used as **goodness of fit** metrics of models.

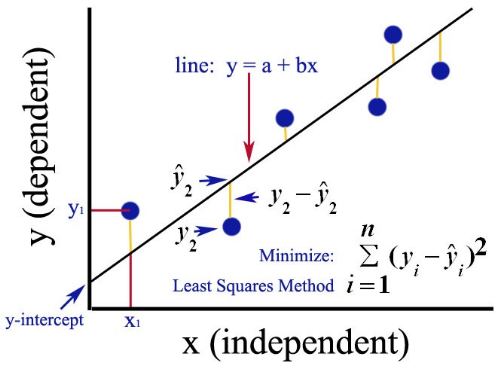
in this project, **RMSE, R-Squared and Adjusted R-Squared** will be used as metrics for evaluating and deciding the best model for this dataset based on **goodness of fit.**

* + 1. **Linear regression**

Linear regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression- Simple and Multiple. Simple Linear Regression means there will be one predictor and one response variable. Multiple Linear Regression is the current scenario which we face where there will be many predictors and a single target variable.

Linear Regression employs various methodologies to formulate the linear relationship and one among them is **the Least Squares Method**, where, there will be a linear line formed, which has the least squared error from the actual values, when compared to any other line formed for the data. A linear regression looks something similar to the below:

**Fig 2.6 Structure of Linear Regression based on Least Squares**



* + - 1. **Implementation**

In this project **OLS (Optimum Least Square)** method was used in Python to develop the model. This method formulates the line with least squared error.

Using test\_train\_split method, the data divided into 80% for training and 20% for testing in Python. There are various sampling methods such as Simple Random Sampling, Stratified Sampling which can also be used to generate sampled data. The test\_train\_split method will randomly shuffle the data and pick the data accordingly.

below is the summery of the regression model:

**Fig 2.7 (A) reg\_model Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | fare\_amount | **R-squared:** | 0.927 |
| **Model:** | OLS | **Adj. R-squared:** | 0.927 |
| **Method:** | Least Squares | **F-statistic:** | 5.421e+04 |
| **Date:** | Thu, 27 Jun 2019 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 20:43:05 | **Log-Likelihood:** | -30463. |
| **No. Observations:** | 12852 | **AIC:** | 6.093e+04 |
| **Df Residuals:** | 12849 | **BIC:** | 6.095e+04 |
| **Df Model:** | 3 |  |  |
| **Covariance Type:** | nonrobust |  |  |

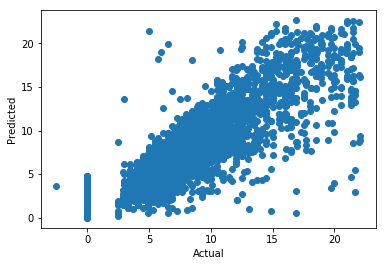
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **distance\_travelled** | 19.1451 | 0.090 | 212.015 | 0.000 | 18.968 | 19.322 |
| **Year** | 3.2769 | 0.060 | 55.031 | 0.000 | 3.160 | 3.394 |
| **Month** | 2.0063 | 0.058 | 34.708 | 0.000 | 1.893 | 2.120 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 2074.540 | **Durbin-Watson:** | 1.970 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 9156.902 |
| **Skew:** | 0.731 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.868 | **Cond. No.** | 3.30 |

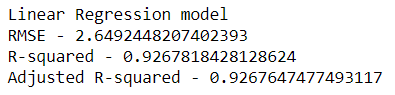
From the model summary, the **R-squared and Adjusted R-squared** values are close to 1. This shows that the model developed can determine 92-93% of the predicted values. this model has a high **goodness of fit** index.

Similarly, the column **‘coef’**explains 1 unit of a variable brings about how many units of change in the predicted variable. Based on this, the highest change (i.e., more correlation) is brought about by the **‘distance\_travelled’** column. The **p-value** column is also negligible for all variables and so none of the variables are unnecessary in predicting the target variable. The scatter plot between ‘actual’ and ‘predicted’ values of cnt\_model is shown below:

**Fig 2.7 (B) Actual vs Predicted (reg\_model)**



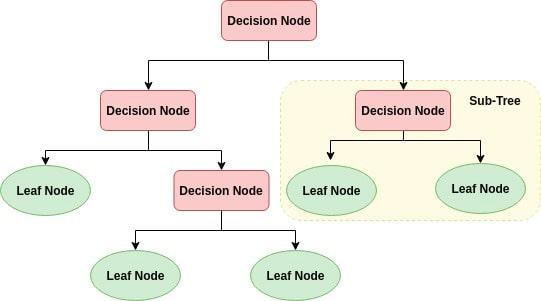
The linear relationship in the scatter plot shows that the predicted values are close to the actual values. If more data is fed in, the model performance can be improved. The evaluation of the model based on the predictions made as follows:



* + 1. **Decision Tree**

The first ML algorithm that will be used here is the **Decision Tree Algorithm.** Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The result is a tree with decision nodes and leaf nodes. A **decision node** has two or more branches. **Leaf node** represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node.** Decision trees can handle both categorical and numerical data.

**Fig 2.8 Structure of a Decision Tree**

**Root Node**

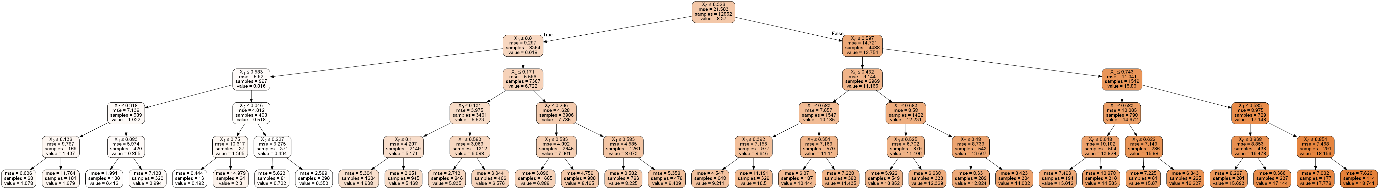
* + - 1. **Implementation**

Here the **Decision Tree Regressor** will be used for building the model in Python utilizing **CART Algorithm for formation of trees**.

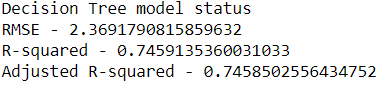
After the model is built, using the independent variables of test data, dependent variables will be predicted and then evaluate the error metrics for this model.

The depth of the tree can be limited to keep constraints in developing the model, otherwise the tree will develop until it reaches exact leaf node for all branches. This is called ‘**Pruning**’. When different depths were tried for the decision tree, a **tree depth of 5** was optimum for this dataset as after that if the tree depth is increased, the output change was not remarkable. Below is the **sample decision tree generated for the tree model.**

**Fig 2.9 Decision Tree For Tree model**

****

the model was evaluated based on the predictions made. The results are as follows:



The RMSE is slightly lesser when compared to the regression model, but the goodness of fit metrics of R-squared and Adjusted R-squared values have been considerably less when compared to Linear Regression Model.

This shows that the **decision tree model is not as effective as the Linear regression model** for this dataset.

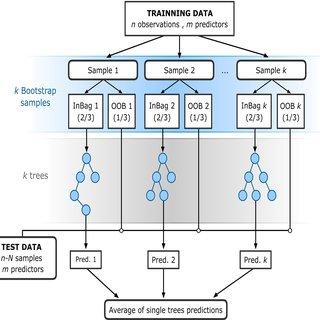
* + 1. **Random Forest**

The next regression model that we are going to use to predict the target variables is the Random Forest model. Random Forest is a supervised learning algorithm. From its name, we can sense that it creates a forest and makes it somehow random. The forest it builds, is an **ensemble** **of Decision Trees**, most of the time trained with the “**bagging**” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random Forest basically is a collection of many decision trees and it combines the result of all the decision trees to get more accurate prediction of dependent variable.

Random Forest algorithm can be used for both Classification and Regression problems and it is a great advantage for this algorithm. Here, we can use this algorithm to predict the dependent variable by regression.

**Fig 2.10 Structure of a Random Forest**

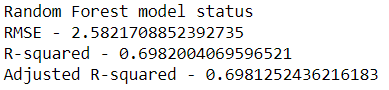


Here we can use the Random Forest for Regression and predict the cab fare.

* + - 1. **Implementation**

The **Random Forest Regressor was used** for building the model in Python. Random Forest is an ensemble of Decision Trees. We can specify and decide on the number of trees to be used in the algorithm. On trying out a different number of trees for this dataset, **10 trees** are decided for building this model as above 10, the result of the predictions didn’t change much.

Usually, ensemble models perform better when compared to the normal ML algorithms. But it isn’t guaranteed that the ensemble models are always the better ones. The results are as follows:



The RMSE and goodness of fit metrics have increased, R-squared and Adjusted R-squared have decreased when compared to the Decision Tree model. This shows that the **random forest model is not as effective as the Linear regression model and the decision tree model** for this dataset.

**Chapter 3**

**Conclusion**

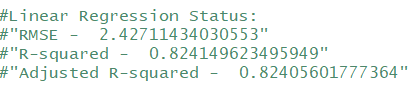
**3.1 Model Selection**

various pre-processing techniques were performed on the data as part of data cleaning to be ready for the Modelling phase. Then models were built using three different algorithms to decide which model is better in prediction. finally, the models were developed and evaluated.

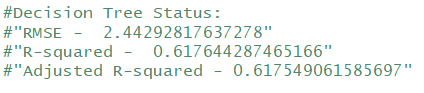
Based on the error metrics, the Decision Tree and the Random Forest algorithms can predict the target variable, but they are not better when compared to the **Linear Regression model** in predicting the target variable for this dataset. This is because the **goodness of fit metrics R-squared and Adjusted R-squared are high for the Linear Regression model** when compared to the other two models.

Even for the R-code, the error metrics are as follows when the models are run:

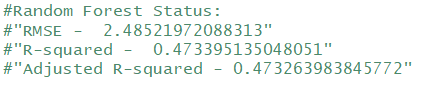
**Linear Regression:**



**Decision Tree:**



**Random Forest:**



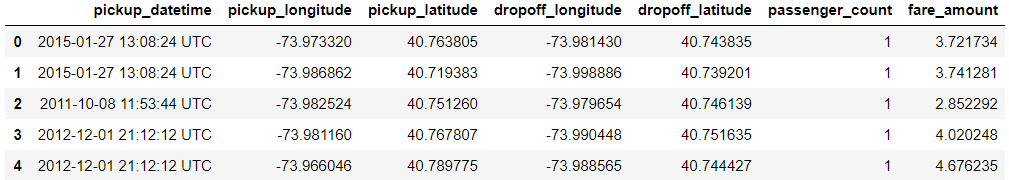
So, for this dataset, we use **Linear Regression model** to predict the cab fare for our launch across the country

**3.2 Output**

Using Linear Regression Model in Python, the cab fare was predicted based on the test data given in **test.csv.**

The output in Python is stored in **Python Test Output.csv** file and it is as follows:

**Fig 3.1 (A) Python Output**



Thus, the model is generated based on the data and the output is predicted by the model and stored in the CSV files.

**References**

Matplotlib library attributes – referred from [www.matplotlib.org](http://www.matplotlib.org)

Pydotplot library and graphviz for visualising decision tree – referred from blogs in [www.medium.com](http://www.medium.com)

Small functionalities in Python – referred from [www.stackoverflow.com](http://www.stackoverflow.com)

Modelling and evaluation metrics – referred from [www.edwisor.com](http://www.edwisor.com)

Error metrics choosing techniques – referred from [www.medium.com](http://www.medium.com)

Importance of error metrics – referred from [www.analyticsvidhya.com](http://www.analyticsvidhya.com)