**Tweet Impact Prediction**

**Problem Statement:** Prediction of the impact of a tweet. The impact is measured in numerical value which could give insight if a tweet could go viral.

**Dataset information:**

The dataset contained 15 independent features and a target variable “Impact”. The shape of the data was (50000, 18), meaning it has 18 columns and 50000 rows.

**Data Pre-processing:**

* The dataset contained no null values.
* It had redundant columns like id and unnamed column which had unique values like id. Hence these both columns were removed from the dataset as they would not help in further analysis.

The feature named “**Post Contet**” is a text which is basically the tweet that was posted. It was multilingual in nature and would be difficult to process. However, the language of the tweet could play an important role in predicting the impact. Hence, to extract the language from the text, NLP was performed by using the help of Language Identification Tool – Langid. The tool predicted 92 unique languages from the feature. These languages were added into the data set as ‘**Text\_Language’** and since the sentiment from the tweet was already given in the dataset, it was found there will be no further use of the feature “Post Contet”. Hence the feature was dropped from the dataset.

* The feature “Published Date Time” is a date-time data type, with additional information about the UTC coordinates. The unnecessary information was removed and the date-time was converted to ordinal values.

**Feature preparation:**

Since it is a Regression problem, all the independent and dependent features should be of numerical data type.

The two categorical variables in the dataset were encoded into numerical values

* “Text\_Language” was Label encoded with 92 unique languages were assigned numbers from 1-92 in alphabetical order
* “Media Type” which has 3 unique classes was One-Hot encoded

The final shape of the dataset was (50000, 17) with one target variable in it

1. X was used to represent the independent features and Y represents Impact
2. X, Y were then split into train ad test data. The split was done in the ratio of 70:30 where train data contains 70% (35000 values) of the data where the modelling and training will be performed and the results will then be evaluated on the 30% (15000 values) of the test data.
3. The train and test data was scaled using StandardScaler function from scikit learn

**Modelling:**

**Statistical Regression Model**

The base model used was a statistical Ordinary Least Square Regression model from scipy package. This model establishes a linear relation between independent and dependent variables of form y = mx+c. The F-statistic probability value is 0 which implies that the model is valid. Although the model is valid, the R-square value of the model is 1 which tells the model was over-fitting.

The mean square error and mean absolute errors from this model were the highest compared to other models.

The wall time for the OLS model is 43ms

The errors in the made during the training were plotted with the predicted values to check homoscedasticity. The errors made had constant variance.

**Linear Regression (Scikit-learn model)**

Using scikit learn linear model, Linear Regression model was used to train the training data and the results were significantly better. Upon evaluating on the test data, the errors were less compared to training data. The R-square value is 99.99 both on test and train data, which tells the model is able to explain 99.99% of variation in the data.

On train data, the mean square error is 6.48e-9 and mean absolute error is 4.69e-6

On test data, the mean square error is 6.62e-10 and mean absolute error is 3.80e-6

The wall time for Linear Regression model is 14ms

The errors made in the model can be due to over-fitting of the model

**Decision Tree (Base model)**

A base Decision Tree model with default hyper parameters was trained on the training data and as expected the tree over fitted as there is no limit in the levels the tree can grow, which is explained by the R-square value that came out to be 1. The model had almost zero error in the training data, but when used on training data the percentage of errors made were significantly high.

On train data, the mean square error is almost 0 and mean absolute error is 0

On test data, the mean square error is 0.02 and mean absolute error is 0.008

The wall time for Decision Tree Base model is 409ms

The errors made in the model are due to over-fitting of the model

**Decision Tree (Pruned)**

As the base model of Decision Tree was over-fitting, hyper parameter tuning will help in fitting the data correctly. The parameter like max\_depth, min\_samples\_split, min\_samples\_leaf were used with various values to predict the impact with best results using GridSearchCV. The model after tuning was able to control over-fitting and was giving better results on train data, however there were errors in the test data and the evaluation metrics showed that the tuned Decision Tree was not performing as good as Liner Regression model which is explained by the R-square value which is 99.98 on train data and 99.78 on test data.

On train data, the mean square error is 0.00014 and mean absolute error is 0.0052

On test data, the mean square error is 0.0023 and mean absolute error is 0.011

The wall time for Pruned Decision Tree model is 252ms

**Random Forest Regressor**

To further improve the results of Decision Tree, Random Forest Regressor with the same hyper parameters and 100 trees was used to train the model. The model performed well and gave better results compared to pruned Decision Tree. The R-square for train is 99.97 and test is 99.87 which showed improved results

On train data, the mean square error is 0.00022 and mean absolute error is 0.0031

On test data, the mean square error is 0.0013 and mean absolute error is 0.0056

The wall time for Random Forest Regressor model is 15sec

**Neural Network model**

To compare the above models, a sequential neural network model with 2 dense layers is selected. Using Tensorflow and Keras packages, two hidden layers were given to the neural network with density of 128 on first layer and 64 on the second layer with ‘Relu’ activation. For the output, the activation used was ‘Linear’ as this is a Linear Regression problem. The loss was calculated on mean square error using ‘Adam’ optimizer.

The neural network performed significantly better compared to Decision Tree or Random Forest with R-square of 99.99 on both train and test data.

On train data, the mean square error is 7.00e-6 and mean absolute error is 0.0011

On test data, the mean square error is 8.24e-6 and mean absolute error is 0.0011

The wall time for Sequential Neural Network model is 2min

**Analysis**

**Performance of all the models on train vs test data**

Graphs were plotted for the models taking actual values on y-axis and predicted values on x-axis to study visually how the models were performing

Below graphs in Fig.1, 2, 3, 4 show the performance and errors in each model for train (left) and test (right)

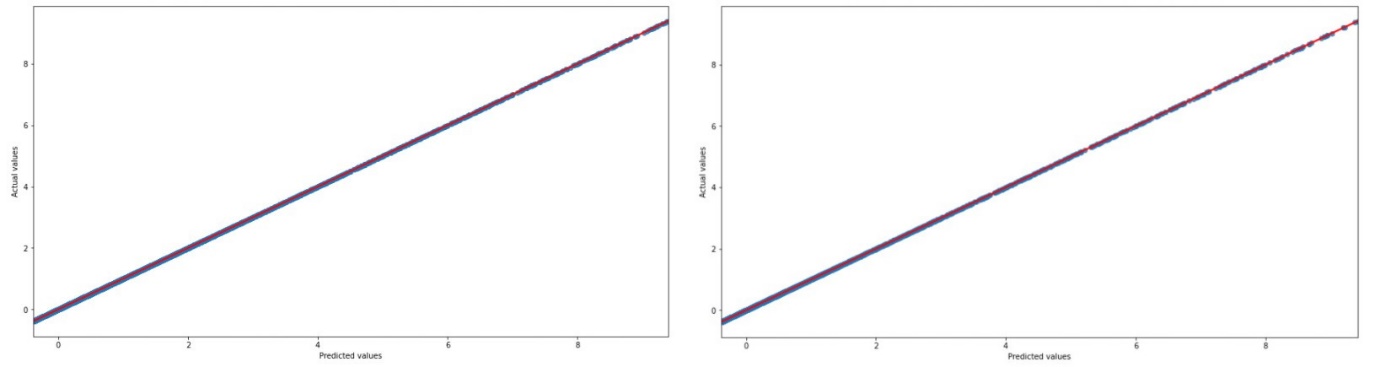
****

Fig. 1 Linear Regression train vs test

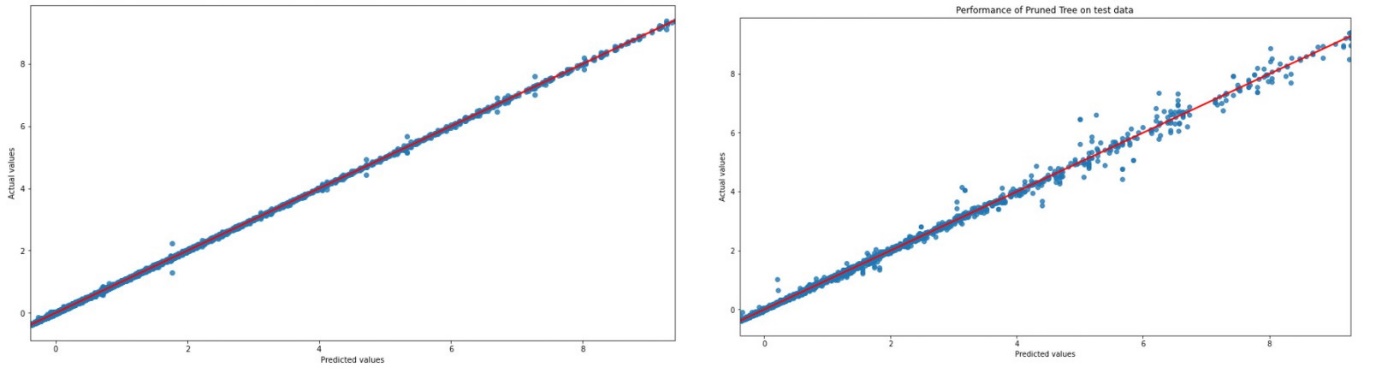


Fig.2 Pruned Decision Tree train vs test

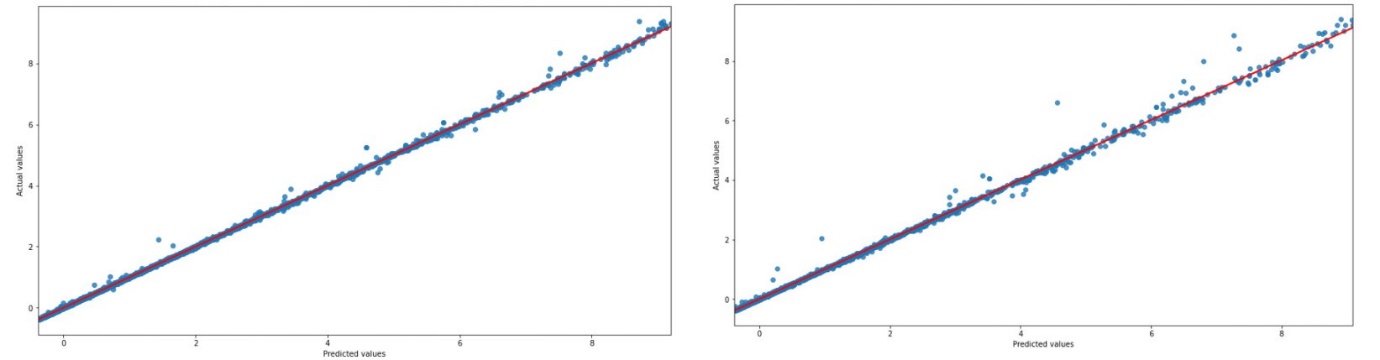


Fig.3 Random Forest Regressor train vs test

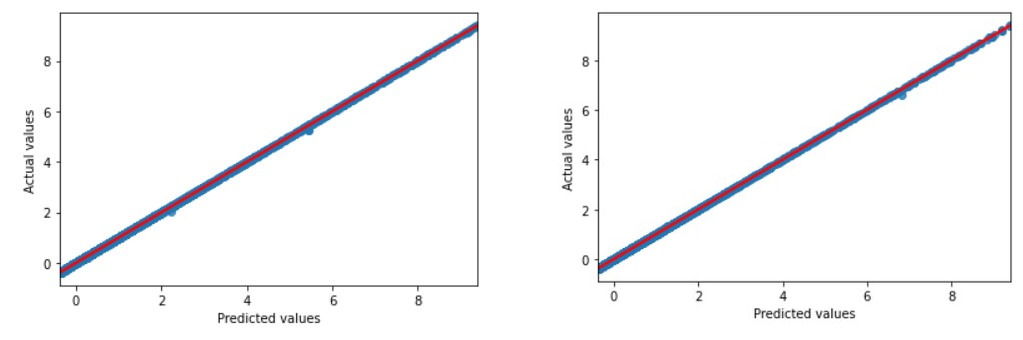


Fig. 4 Neural Network train vs test

**How were the hyper-parameters for Decision Tree selected?**

The base Decision Tree was over-fitting which is explained by the R-square value of 1. This means that the tree needs a restriction on depth in order to avoid over-fitting. The best depth can be found by using GridSearchCV. The depth values initially were given from 2 to 14 with a step count of 2. Along with depth, min samples split and min samples leaf values of (1, 2, 4, 5) were also passed to the algorithm as trial and error method. The algorithm suggested depth of 12 with min samples leaf of 1 and sample split as 4. Using these tuned parameters, the Decision Tree performed only slightly well on test data compared to base Tree as shown below.

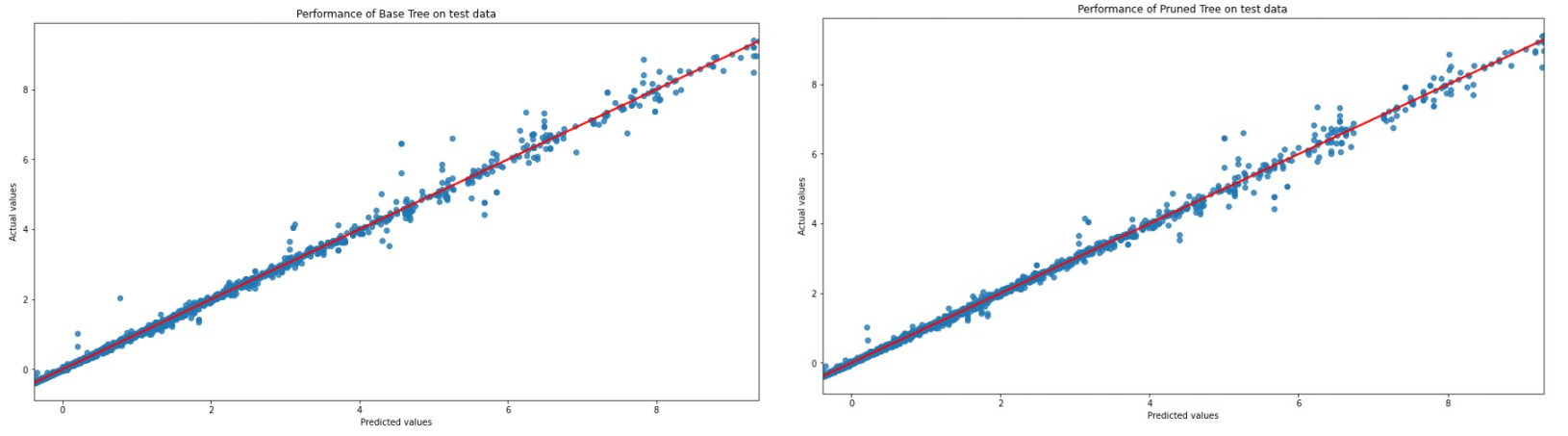


Fig.1 Base vs Pruned Tree

**What happens if you chose different values while tuning?**

Since the model was over-fitting, if max depth values were taken beyond the suggested value, the model will likely over-fit and if the values were taken which are less than the suggested depth and sample, the model will not perform good even on the train data (under-fit) leading to high bias error.

**What is the best model?**

Since base Decision Tree is an over-fitting model, it is excluded from the evaluation metrics analysis. Upon merging all the results, Linear Regression performs the best in predicting the Impact of a tweet followed by Sequential Neural Network model as shown in Fig.2 when the evaluation metrics mean square error and mean absolute error were considered.

But it is also important to consider that Linear Regression will not be the best model to predict the Impact every time as different datasets may establish different relations. Thus, for future predictions, it is reasonable to consider a more generic model that can perform well on any data which is the neural network model.

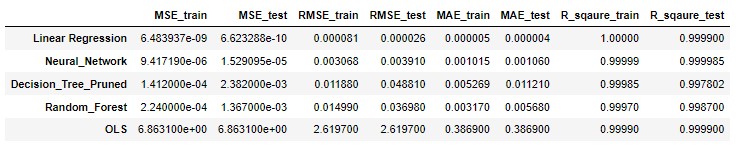


Fig.2 Model performance in descending order

**Why were the models able to explain more than 99% of variation?**

This can be answered by considering the feature importance that each model was considering. From the below Fig.3, it is observed that, in the linear relation between independent variables and Impact, more than 93% of the Impact is just explained by ‘Likes’ alone followed by ‘Shares’ and ‘Comments’. This is a logical explanation as this is the case practically. A tweet is more likely to go viral depending on the amount of interaction it gets. The interaction on Twitter is through liking, retweeting and commenting.

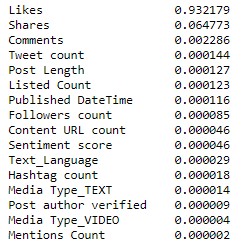


Fig. 3 Feature Importances

Hence, all the models were able to perform and explain more than 99% of the variation by just considering first three features from the figure.