**Report**

**1.Dataset Description:**

|  |  |
| --- | --- |
| **Dependent columns** | **Independent columns** |
| Impact | Sentiment score - Ranges from -20 to +20 (0 - neutral)  Post Length - The length of the tweet  Hashtag Count - The number of hashtags used in the tweet  Content URL Count - The number of URLs mentioned in the tweet  Tweet Count - The total number of tweets posted by the author of the tweet  Followers Count - The number of followers of the author of the post  Listed Count - the number of lists the post author is a part of  Media Type - The media type of the post (Text, image, video)  Published Datetime - The published time of the tweet  Mentions Count - The number of user mentions in the tweet  Post Author Verified - 1 if author is a verified user  Likes - Likes received for the tweet  Shares - Retweets received for the tweet  Comments - Number of comments for the tweet |

**Table1: Dataset Independent and Dependent Columns**

**1.2 Exploratory Data Analysis of Dataset:**

**Result of df.info():**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 50000 entries, 0 to 49999

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 50000 non-null int64

1 Id 50000 non-null object

2 Post Contet 50000 non-null object

3 Sentiment score 50000 non-null float64

4 Post Length 50000 non-null float64

5 Hashtag count 50000 non-null float64

6 Content URL count 50000 non-null float64

7 Tweet count 50000 non-null float64

8 Followers count 50000 non-null float64

9 Listed Count 50000 non-null int64

10 Media Type 50000 non-null object

11 Published DateTime 50000 non-null object

12 Mentions Count 50000 non-null float64

13 Post author verified 50000 non-null float64

14 Likes 50000 non-null float64

15 Shares 50000 non-null float64

16 Comments 50000 non-null float64

17 Impact 50000 non-null float64

dtypes: float64(12), int64(2), object(4)

memory usage: 6.9+ MB

The dataset is loaded in pandas dataframe(df) by using df.info() the non-null values count , datatype of each column and count value in each column can be analysed . **“Id”,”Post** **Contet”,”Media Type”, “Published DataTime”** are object datatype and others are numeric as float and integer.

|  |  |
| --- | --- |
| Figure1(a) | Figure1(b) |
| Figure1(c) | Figure1(d) |

**Table2: Line plot using seaborn on various columns to establish relation between independent columns with “Impact” column.**

From table2, graphs of figure1(a)-1(d), it is hard to establish relationship between “Impact” and independent column.

**2.Dataset Pre-processing:**

Step1: **Dimension Reduction:** In dataset pre-processing, the columns “Id”, “Post Contet” and “Published DateTime” are removed from data frame. The column “Id” doesn’t hold any relation with “Impact”, “Post Contet” is multilingual holds meaningless in feature value, and “Published DateTime” is doesn’t relate to Impact. Moreover, the three columns are “object” datatype. By using syntax to remove three columns as:

**df.drop(columns=[“Id”,”Post Contet”,”Published DataTime”])**

Step2: In second stage, “Media Type” is object and multi-classed can hold feature value in scoring “Impact”. “Media Type” column is labelled using LabelEncoder() function of sklearn.

Step3: From exploratory data analysis through df.describe() functionality and table2, a great divergence mean, min, max of each column of dataset is observed. Standard scalar is used to standardised the value of columns.

Step4: x and y are initialized. “x” contains all dependent columns and “y” independent column(“Impact”)

Step5: Dataset is splitted into train and test split with 75:25 ratio using train\_test\_split().

**3.Model Architecture Deployed:**

**3.1Linear Regression:**

Linear regression is a basic and commonly used type of predictive analysis.  The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?  (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?  These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.  The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b\*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

**3.2Decision Tree:**

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.

Hyper-parameter tuning in decision tree:

**(a)max\_depth *int, default=None***

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

**(b)min\_samples\_leaf *int or float, default=1***

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_sample\_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

**3.2.1Pruning in decision Tree:**

Pruning is a **data compression technique in machine learning and search algorithms** that reduces the size of decision trees by removing sections of the tree that are non-critical and redundant to classify instances. In the pruning step of decision tree, column “Post Length” column is removed (less significant in contribution in feature). Hyper-parameter is tuned same as Decision tree above mentioned.

**3.3** **Multilayer Perceptron:**

**Multi-layer Perceptron (MLP)** is a supervised learning algorithm that learns a function f(⋅):Rm→Ro by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,...,xm and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

**Hyper-parameter tuning in MLP:**

(a) **learning\_rate\_init *double, default=0.001***

The initial learning rate used. It controls the step-size in updating the weights. Only used when solver=’sgd’ or ‘adam’.

(b) **max\_iter *int, default=200***

Maximum number of iterations. The solver iterates until convergence or this number of iterations. For stochastic solvers (‘sgd’, ‘adam’), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

**Model Evaluation:**

The models are evaluated on the basis of root\_mean\_sqaured\_error(RMSE/rmse) calculated on predicted and actual value on both training and testing datasets. Hyper-parameter tuning done on basis of “random search” approach to reach optimal values.

**4.Result:**

The following code is implemented Python3.7.6 on Jupyter Notebook. Skicit-learn library functions are used in importing models and in data-pre-processing techniques along with Pandas and NumPy. The figures are plotted using matplotlib and seaborn library functions. For time calculation of model running time, the code is run on google colab.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **RMSE on Training set** | **RMSE on Test set** | **Time in seconds** |
| **Linear Regression** | **2.2593725600683368e-09** | **2.470665661863774e-09** | **0** |
| **Decision Tree with max\_depth=4,min\_samples\_leaf=0.1** | **0.199430619** | **0.21083972206** | **0** |
| **Decision Tree with max\_depth=4,min\_samples\_leaf=0.01** | **0.0331314385** | **0.03149226655** | **0** |
| **Decision Tree with max\_depth=4 , min\_samples\_leaf=2** | **0.016556581** | **0.163872484** | **0** |
| **Pruned Decision Tree with max\_depth=4 and min\_samples\_leaf=2** | **0.016556581** | **0.0163872484** | **0** |
| **MLP with max\_iter=100 and learning\_rate\_init=0.001** | **0.00017698433915** | **0.00017698433915** | **4** |
| **MLP with max\_iter=20 and learning\_rate\_init=0.1** | **0.002911266097** | **0.0029112660979** | **4** |
| **MLP with max\_iter=200 and learning\_rate\_init=0.01** | **9.873544674868278e-05** | **9.873544674868278e-05** | **4** |

**Table3: Result on training and testing set based root squared mean error.**

(a)From table3, three different models are trained and tested for determination of “Impact” variable using regression methods of each model.”rmse” significantly approaching 0 will have the best model performance.

(b)In linear regression, rmse is less in training set as compared to testing test, but difference is negligible.

(c)In decision tree, max\_depth is set to 4 as it significantly doesn’t affect the output as compared to min\_samples\_leaf. At min\_samples\_leaf=2 , the rmse value is significantly low on both training set = 0.016556581 and testing set=0.0163872484 in comparison to other decision tree models hyper tuned.

(d)Pruning Decision Tree doesn’t add much change in value , performed same as decision tree with max\_depth=4 and min\_samples\_leaf=2.

(e)In MLP with max\_iter=200 and learning\_rate\_init=0.01 performed best in all MLP variants with rmse 9.873544674868278e-05 on both train and test dataset.

(f)In overall model evaluation, Linear Regression performed best followed by MLP with max\_iter=200 and learning\_rate=0.01 and then decision tree with max\_depth=4 and min\_samples\_leaf=2.But in terms of timeframe is worst performer with 4 seconds in running the model in comparison 0 seconds of other models.

**5.Future Work:**

The detailing “data visualisation” between “Impact” column and independent columns would significantly help “Principal Component Analysis”. Support Vector Regression and Random Forest can be applied in order to fetch more accurate prediction.