**About :**

Sentiment analysis is the technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

Here, we are using a supervised machine learning technique to perform text mining and predicting the tweets into good and bad ones based the labeled tweets in the training dataset.

**Problem statement:**

The goal of the program is to classify the tweets into a good tweet or a racist/sexist tweet. The training data already consists of the classified tweets into good and bad with the labels 0 and 1 respectively.

It is easier to predict rather than making use of human resource to spend their time identify which tweet belongs to what category. Thus, the problem is to predict the labels of the tweets in the training data set.

**Approach:**

Initially the input data is already split into a train dataset and test dataset. So, the data should be cleaned and a model should be constructed and trained to predict a good and bad tweet.

I have used an inbuilt library called as tweet-processor, which automatically removes certain symbols and words from the tweet which do not add any value to our model. For example, this library removes the user names of the person who has tweeted and also removes symbols such as ‘@’, ‘#’ etc.

Moving forward, I need to replace some characters in the data with a space and some without a space and thus I have created two variables, each containing a list of characters that need to be replaced. Then I have created a user defined function that removes the unnecessary characters, numbers from the data using the tweet -processor and Regular expression and replaces those characters with or without space. This user defined function also lowers the case of the text that is cleaned and returns it to a new column, ‘Clean’ to our original train data.

Now, the tweet data in each row is cleaned, however, they are in the form of a sentence and as words in different tenses. Thus, I create another user defined function to tokenize the sentence into words, using word\_tokenize function, which is imported from NLTK library. Once the sentence is converted to a combination of words called as tokens, I have lemmatized the words, which refers to the process of converting all the words to its original noun form or root word.

For example, if a tweet consists of a sentence that read:

“She has no knowledge on science, to be able to give a scientific proof”

In the above sentence, the word science and scientific are same words in different verb/ tense forms. Thus, by lemmatizing, these words get converted to one root word – Science.

The output csv file that we export required the id column containing the user id of each person who tweeted and thus, I am using a pop function to store those values in the column ‘ID’ of our train data set, to a new variable and removing it from the train dataset.

Now that the data is cleaned, we have to convert these text data into vectors and then fit those vectors on the model created, since machine doesn’t understand how to relate words those are good tweets and racist tweets. There are different models that convert words to vectors and also tag the words based on certain classes like entity, date, Personality etc. some of it are listed below:

1. Bag of words model
2. TfIdf vectorizer
3. Word embedding – Word2Vec model
4. Named entity recognition
5. BERT
6. Elmo
7. GPT-1
8. GPT-2

In this problem, I have used TfIdf vectorizer, which unlike the bag of words model, not only counts the frequency of each word, it also assigns a weightage to the words and converts them to vectors and this serves as a better model to convert to vectors.

Moreover, I am removing the stop words from the cleaned twitter data by passing an argument of English stop words along the TfIdf vectorizer function. Followed by that I fit my cleaned training data on the vectorizer model.

The evaluation metric that is asked is the F1 score to show the misclassification of our prediction to good and bad tweet. And for the computation of our F1 score, we require our original output and the predicted data. Since, we do not have the output labels, I have split our train data into train and test in a ration of 70% and 30% respectively.

Having converted the data into vectors and making a train, test split, all that is left is to create a model to classify the tweets. The models, I have used are:

1. SVM
2. Decision Tree Classifier
3. XGBoost Classifier
4. Logistic Regression
5. Random forest classifier
6. Word Embedding Model with Tensor flow and keras – Neural network
7. Auto - NLP

1) Support Vector Machines – SVM :

SVM is a highly advanced and stable algorithm that helps in classification of the datasets. This model ensures that it reaches a maximum marginal distance between the classified groups. Thus, the model will also be able to give lesser misclassification of the tweets into good and bad ones.

SVM works for both linearly and non-linearly separable data. The idea of SVM is to plot the data point in a n dimensional space with the value of each feature in being the value of particular coordinate. Then, it is followed by identifying a hyper plane that differentiates the classes ensuring that there is a maximum separation between each of the classes. There by becoming one of the efficient methods for a classification problem.

2) Decision Tree Classifier:

Decision tree algorithm belongs to supervised machine learning. Unlike other supervised machine learning algorithms, decision tree helps to solve both regression and classification problems.

Decision tree helps predict a class or value of the target variable by taking simple decisions that is inferred from the prior data. It starts from the root node of the tree. The root node is the feature with the maximum information gain. The root node keeps splitting into terminal or leaf node after passing through a. Decision node. If the branches can no longer be further split, then they an entropy of 0 and are referred as the pure branches. On the other hand, if they can be further split into branches, they are impure nodes with an entropy value greater than 0.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. Each decision node simply answers the question yes or no, and further splits into sub trees.

3) XGBoost Classifier:

This is one of the gradient boosted decision tree method in order to increase the speed and performance that is more competitive over other machine learning algorithms.

It tries to push the limit of computation resources for boosted tree algorithms, which means:

* **Parallelization** of tree construction using all of your CPU cores during training.
* **Distributed Computing** for training very large models using a cluster of machines.
* **Out-of-Core Computing** for very large datasets that don’t fit into memory.
* **Cache Optimization** of data structures and algorithm to make best use of hardware.

4) Logistic Regression:

Logistic regression is used when the target or the dependent variable which we want to predict is binary.

It describes the data and explains the relationship between one dependent binary variable or multinomial variable and among one or more nominal, ordinal, interval or ratio-level independent variables. Moreover, it is used when the dependent variable is categorical, similar to the one in the problem of predicting a racist/sexist tweet.

The output is in the form of a sigmoid curve, with the values between 0 and 1 unlike a linear regression model.

5) Random Forest Classifier :

Random forest is one of the ensemble technique method of bagging techniques, which is used to overcome the overfitting of the data in a decision tree. Random forests is nothing but construction of multiple decision trees that are constructed on top of randomly selected sample from the datasets. It is also referred as a parallel classifier. This is because, it parallelly, tests the results on different decision tree samples, thereby the stability of the model is preserved throughout both in the training and the testing dataset.

Finally, the output is taken from the majority of the predictions that is got form all the randomly selected decision trees in the random forest.

While this is the opposite in case of a boosting algorithm, they are sequential classifiers and thus the accuracy is increased, however, they are not as stable compared to that of a Random Forest classifier.

6) Word Embedding Model

The word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.

These vectors are based on cosine similarity and we know that the value of angle cos 0 is 1, meaning that if the vectors with lesser angle towards 0, show more similarity. Thus word2vector or the word embedding model, helps identify the words with closer similarity to each other based on the cosine similarity of these vectors that are projected on a n-dimensional space. Here, I have used to identify similar positive of negative words among the tweets.

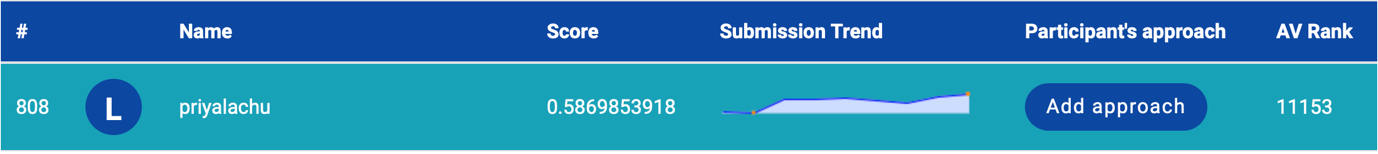
7) Auto NLP:

This is a process to automate the process of creating an end to end NLP model. Auto NLP is atoll developed by the Hugging face team in March 2021. It aims to automate all the life cycle of an NLP model, starting from training and optimizing the model to deploying it. It does very well for binary classification, multi-class classification, entity recognition and supports 8 different languages. Thus, it speeds up the process of creating an NLP model.

Some of the features of an AUTO NLP are as follows:

1. Data cleansing – The entire dataset can be directly sent to the model, without any vectorization. It also fills in the missing values and cleans the data automatically
2. Uses feature tool library for feature extraction: Feature tools is another library that helps in extracting the features in a easier way
3. Model performances and Visualising graphs are produced automatically, just by setting the verbose value
4. Feature reductions are automatic even for huge datasets.

**Leader Board – public leaderboard**

****

**Leader Board – private leaderboard**