Coronavirus Tweet Sentiment Analysis

Vinayak Potdar
Data science trainee,
AlmaBetter, Bangalore

Abstract:

Covid-19 hit the whole world in the year 2020. This year was a disastrous year when the world stopped functioning. People were alert and were using social media platforms to the fullest.

Twitter saw its users asking for help, criticizing the governments, providing help and much more. This sea of emotions and sentiments could be seen through their tweets.

In this experiment we will try and build a classification model which will classify the sentiments of the tweets.

1. Problem Statement

This challenge asks you to build a classification model to predict the sentiment of COVID-19 tweets. These tweets have been pulled from Twitter and manual tagging has been done then.

Data Description

The dataset consists of web scrapped data, from Twitter. The dataset consists of tweets made by users regarding Covid-19. We can see records from year 2020.

The names and usernames have been given codes to avoid any privacy concerns.

The provided data does not consist of many features. The features important for sentiment analysis are just two. Although we will be needing the rest of the features for the Exploratory Data Analysis.

The features present in the data are --

1- Username (Unique_id)

2- ScreenName

3- Location (Location of the tweet)

4- TweetAt (Date of tweet)

5- OriginalTweet (The string of the tweet)

6- Sentiment (manually labelled sentiments)

2. Introduction

Twitter is a short blogging platform where people can share their ideas and thoughts in maximum of 280 characters.

Twitter started seeing a whole lot of tweets in the first half of 2020 regarding Coronavirus. This surge in tweets is being analyzed by us in this experiment.

The data that we worked on here has been scrapped from twitter itself and manually labelled. These tweets were manually analyzed and classified into five different sentiments viz. **‘Extremely Positive’, ‘Extremely Negative’, ‘Positive’, ‘Negative’ and ‘Neutral’**. We will be building a multiclass classification model here which can classify the sentiments of the tweets.

3. Challenges Faced

The provided dataset was fairly well built and not much challenges occurred in this experiment.

The basic challenge that we faced was with the ‘location’, feature. The feature consisted of redundant data, but represented different locations for e.g. “London” and “London, England”. The analysis of location vs no. of tweets was affected by this anomaly.

Second challenge was the number of classes. There were 5 different classes present in the sentiment i.e. target variable. This made the algorithms work harder and took too much computational time.

4. Steps involved:

- **Library Imports and Data Loading**
  Here we started by importing the necessary libraries. Additional libraries were later added as per the requirement in the same section. This was followed by data loading and data description. The data consists of 41157 records in the whole dataset.

- **Null values Treatment**
  The whole dataset was fairly scrapped with only one feature consisting of missing values. About 20% of records from the feature ‘Location’ were missing. This did not affect the learning models as it was not the desired feature for model training.
  This definitely affected the Exploratory Data Analysis as there was a major piece of data missing. This feature also had some redundant values as stated in challenges.

- **Exploratory Data Analysis**
  After completing the null value analysis and basic data inspection, we proceeded with EDA.

  Here we figured out—
  The Major Tweet Locations.
  Tweet count and trend of tweets.
  Balance in the dataset distribution.
  Major keywords of each sentiment.
  Unique hashtags of each sentiment.

  This process helped us figuring out various aspects of the target and the independent variables.

- **Data Preprocessing**
  Here we wanted the data, especially the tweets to be properly processed before we proceeded to vectorization.
  The processing of the data was done as follows-
  We started with by removing the web urls from the tweets. This was followed by removing the tagged usernames in the tweets because they would not be the contributors of the sentiment analysis.
  Removal of punctuations and special characters was the next step,

followed by stop words removal and stemming.

- **Vectorization**
  Here we used CountVectorizer() as a vectorization function. This was sufficient enough, as we had a smaller dataset.
  A dataset with more classes and features could have made use of TF-IDF.

- **Fitting Models**
  Here, we experimented with four different classification algorithms

  a) Random Forest Classifier
  b) Support Vector Machine
  c) Naïve Bayes Classifier
  d) CatBoost

- **Tuning the hyperparameters for better accuracy**
  There were not many hyperparameters which could have resulted in a good score. Here we decided to combine and reduce the number of classes for the classification. This helped in increasing the accuracy score by just one unit.
  Further reduction into just two classes could have helped a lot, but this might have lost the overall goal of classification.

5. Algorithms:

1. Random Forest Classifier

Random Forest Classifier is an ensemble of trees. Here, we grow multiple trees. These trees are constructed from the subsets of the main dataset. In this process each constructed tree votes for a classification of a new object. The forest chooses the classification with the most votes.
In this model, the chances of overfitting are higher, as opposed to the other models we worked with.

.

2. Support Vector Machine Classifier:

SVM is one of the most powerful classifying and multi-classifying algorithm. The idea behind the working of this algorithm is very simple yet powerful. Here, we try and create a separator line or curve based on the number of classes. Instead of using a zero-width line as a separator, we give a margin to the line. The line with the maximum margin width is considered as the best separator.

3. Naive Bayes Classifier:

Naïve Bayes Classifier is one of the crude model that we can use for classification. They are faster and clumsier. This does not provide us with the best fitting model but gives us a baseline to work with.

The "naive" in "naive Bayes" means, if we make very naive assumptions about the generative model for each label, we can find a rough approximation of the generative model for each class, and then proceed with the Bayesian classification. Different types of naive Bayes classifiers rest on different naive assumptions about the data.

4. CatBoost-

CatBoost is an algorithm which is based on gradient boosted decision trees. Here, a set of decision trees are built consecutively. Every tree built next has lesser loss than the previous one.
Catboost is one of the top performers in terms of results and performances. It handles categorical features automatically.
The only thing that holds back catboost is the training time. It takes fairly longer to train, which can be tolerated for the better results.

7.2. Model performance:

Just as we choose a certain learning model for a particular dataset, we need to choose the evaluator for type a dataset as well.
We choose the evaluator based on various particulars of the data such as balance, criticality of the results etc. Here, we focused on the balance in the dataset. The dataset had a fairly balanced distribution of classes; hence we went ahead with Accuracy as our main evaluation metric. We calculated precision and recall as well just to keep an eye on those scores as well.

- **Confusion Matrix**-

Confusion Matrix is used to know the performance of a Machine learning classification. It is represented in a matrix form. Confusion Matrix has 4 terms to understand True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

- **Precision and Recall**-

'Precision' is the ratio of correct positive predictions to the overall number of positive predictions: TP/TP+FP and 'Recall' is the ratio of correct positive predictions to the overall number of positive examples in the set: TP/FN+TP.

- **Accuracy**-

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples. In terms of the confusion matrix, it is given by: TP+TN/TP+TN+FP+FN.

7.3. Hyper parameter tuning:

In hyperparameter tuning, we worked with Support Vector Machine only, as it did not show enough success or accuracy as expected. The dataset we are working with is fairly simple and small to handle. The low accuracy score of this model was a clear indication for hyperparametric tuning.

We worked with GridSearchCV in this model.
GridSearchCV-
GridSearchCV is a library function of sklearn's model_selection package. It loops through various parameter values provided by the user to return the best fitting model parameters for a more optimized model creation.

The next experimentation that we did was with the classes of the dataset. Here in order to increase the accuracy of the models, we broadened the classes of the dataset i.e. we combined and reduced some classes. We combined extremely positive and extremely negative with positive and negative class respectively. This gave us some improvements in the scores of each model.

8. Conclusion:

Here, we conducted two experiments with 5 classes and 3 classes. The first was conducted with hyperparameter tuning and Catboost turned out to be the best performing model with 0.8069 testing accuracy. In the second scenario we got Cat boost as the best perfroming model again. Here SVM improved its performance drastic ally as compared to the first scenario.

References-

1. GeeksforGeeks
2. Analytics Vidhya
3. Almabetter