



Supervised ML(classification) - Corona virus tweet sentiment analysis

Technical documentation

Mohammed Arifuddin Atif
arifuddinatif63@gmail.com

Introduction

The world encountered its first corona virus case on **December 31, 2019** in china. People all around the world in a state of confusion and fear started sharing their concerns on twitter. We are given a dataset with the information of tweets all around the world.

Problem statement

We are tasked with predicting the sentiment of the tweets data which has been pulled from twitter. This is a supervised ml classification problem.

Our model helps understand the sentiment behind a particular tweet and build a sentiment prediction algorithm around it.

Overview of data

We are given the following columns in our data:

1. **Location** - name of the location from which the tweet was shared.
2. **TweetAt** - time of the tweet at which the tweet was shared.
3. **OriginalTweet** - the original tweet shared by the user.
4. **UserName** - identification given by twitter to the user.
5. **ScreenName** - name projected on the screen of the user.
6. **Sentiment** - defined sentiment or label from the shared tweet.

The **Sentiment** column is the dependent variable which consists of 5 different labels which are positive , negative , neutral , extremely positive , extremely negative.

The **OriginalTweet** column is the independent variable and needs some preprocessing to fit our classification model.

Steps involved

I. Performing EDA (exploratory data analysis)

- A. Exploring head and tail of the data to get insights on the given data.
- B. Looking for null values and removing them if it affects the performance of the model.
- C. Removing the unimportant text from the **OriginalTweet** which the model does not understand by using a function.
- D. Removing stop words and punctuations such as is , are , the , / , < etc which will affect the performance of our predictions.
- E. Classifying the 5 labels of **Sentiment** column into 3 labels i.e positive , negative and neutral.
- F. Applying a Stemming function on the given text. Stemming is an operation which reduces the word to its base word or stem in such a way that the words of similar kind lie under a common stem. Example : 'cared' and 'bumped' will be stemmed as 'care' and 'bump'.
- G. Changing the string format of **TweetAt** column into date-time format using a function.

II. Drawing conclusions from the data

Plotting necessary graphs which provides relevant information on our data like :

- A. Getting a plot on the number of tweets in different months.
- B. Plotting the number of tweets on a particular day of the month.
- C. Plotting the number of positive , negative , neutral tweets.

III. Training the model


- A. Vectorizing the text in **OriginalTweet** using `CountVecorizer`. `CountVecorizer` is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.
- B. Assigning the dependent and independent variables
- C. Splitting the model into train and test sets.
- D. Fitting logistic regression on train set.
- E. Getting the predicted variables from the model.

IV. Evaluating metrics of our model

- A. Getting confusion matrix of train and test sets.
- B. Evaluating different scores such as accuracy score , precision score , recall score , f1 score.
- C. Getting the classification report of our model.

Models used

Model training:



Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts $P(Y=1)$ as a function of X .

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

Logistic regression assumptions:

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.
- The independent variables are linearly related to the log odds.
- Logistic regression requires quite large sample sizes.

We have to train our model considering the above assumptions.

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

Types of Logistic Regression:

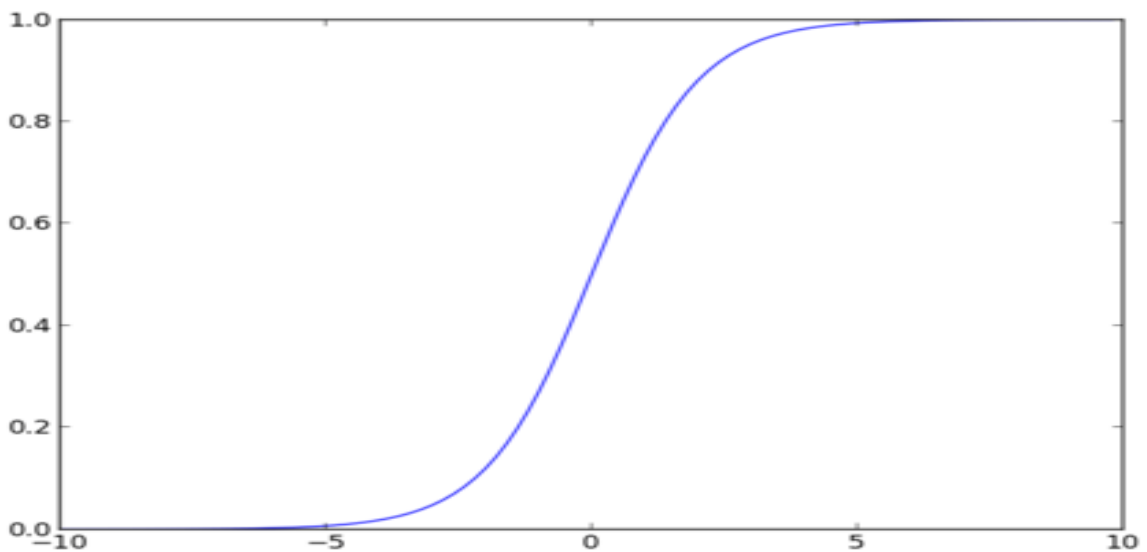
- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

Our model comes under multinomial logistic regression as there are 3 categories positive , negative , neutral.

Sigmoid Function:

The sigmoid function, also called logistic function gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The output cannot For example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that patient will suffer from cancer.

$$f(x) = \frac{1}{1 + e^{-(x)}}$$




Decision trees classifier:

A decision tree classifier is a systematic approach for multiclass classification. It poses a set of questions to the dataset (related to its attributes/features). The decision tree classification algorithm can be visualized on a binary tree. On the root and each of the internal nodes, a question is posed and the data on that node is further split into separate records that have different characteristics. The leaves of the tree refer to the classes in which the dataset is split.

We experimented with decision trees for our model and achieved good results.

Multinomial Naive bayes classifier:



Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.


Approach used

Word Counts with CountVectorizer:

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

You can use it as follows:

1. Create an instance of the *CountVectorizer* class.
2. Call the *fit()* function in order to learn a vocabulary from one or more documents.
3. Call the *transform()* function on one or more documents as needed to encode each as a vector.



An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

Tf-IDF method:

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

It has many uses, most importantly in automated text analysis, and is very useful for scoring words in machine learning algorithms for Natural Language Processing (NLP).

TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don't mean much to that document in particular.

However, if the word *Bug* appears many times in a document, while not appearing many times in others, it probably means that it's very relevant. For example, if what

we're doing is trying to find out which topics some NPS responses belong to, the word *Bug* would probably end up being tied to the topic Reliability, since most responses containing that word would be about that topic.

Challenges faced

1. Pre-processing the data was one of the challenges we faced which includes removing unimportant text from the data so as to not hinder the performance of our classification model.
2. Exploring the **Location** column was a challenging task as there were a large number of unique location values in the given dataset. So we had to settle for the top 10 locations with the highest number of tweets.
3. Selecting the appropriate vectorizer to maximize the performance of our model was one of the challenges.

Conclusion

We are finally at the conclusion of our project!

Coming from the beginning we did EDA on the dataset and also cleaned the data according to our needs. After that we were able to draw relevant conclusions from the given data and then we trained our model on logistic regression.

For our model we were able to get the accuracy of 88% for the train set and accuracy of 80% for the test set using logistic regression.



Given the size of data and the amount of irrelevance in the data , the above score is good.