Twitter Sentiment Analysis Using Natural Language Processing (NLP) with Python

The Ultimate Guide for Identifying Sentiments and Performing Text Analysis on Twitter Data



**Blog By: Chaithanya Vamshi Sai**

**Student ID: 21152797**

**Blog By: Chaithanya Vamshi Sai**

**Student ID: 21152797**

# **Introduction**

Natural Language Processing (NLP) is an emerging field and a subset of machine learning which aims to train computers to understand human languages. The most common application of NLP is Sentiment Analysis.

In the process of NLP, we aim to prepare a textual dataset to build a vocabulary for text classification.

In this blog, I will walk you through the entire process of how to do Twitter Sentiment Analysis using Python.

# **What is Sentiment Analysis?**

Sentiment Analysis is also known as opinion mining which is one of the applications of NLP. It is a set of methods and techniques used to extract information from text or speech. In simpler terms, it involves classifying a piece of text as positive, negative or neutral.

Twitter is one of those social media platforms where people are free to share their opinions. We mostly see negative opinions on Twitter. So, we should continue to analyse the sentiments to find the type of people who are spreading hate and negativity.

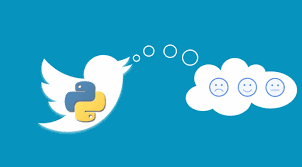
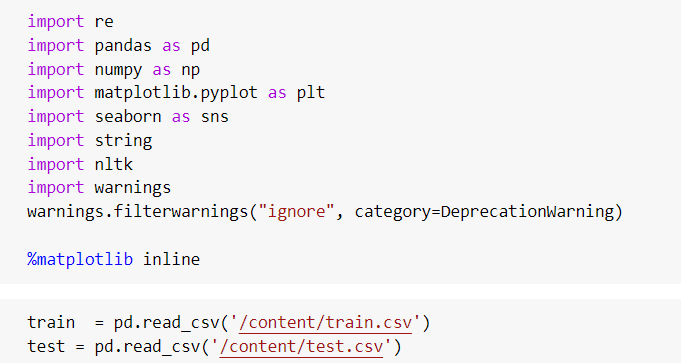


Image Source: https://giters.com/

# **Problem Statement**

The objective of this task is we are given a data set of tweets that are labelled as positive and negative. Using these labelled data, we need to train a Machine learning model using Python to predict the sentiment (positive or negative) of new tweets.

# **Importing Libraries and Reading Data**



# **Tweets Pre-processing and Cleaning**

The pre-processing of the text data is an essential step as it makes it easier to extract information from the text and apply machine learning algorithms to it.

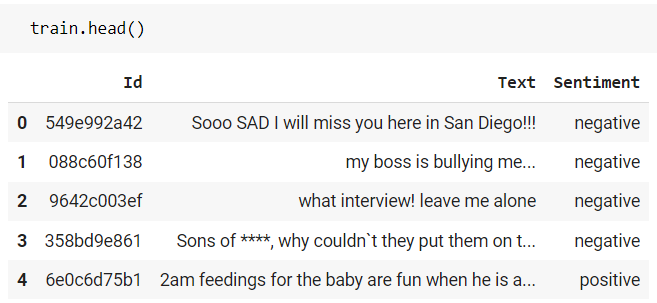
The objective of this step is to Inspect data and clean noise that is irrelevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms that don’t carry much weightage in context to the text.

Data Pre-processing is divided into two parts:

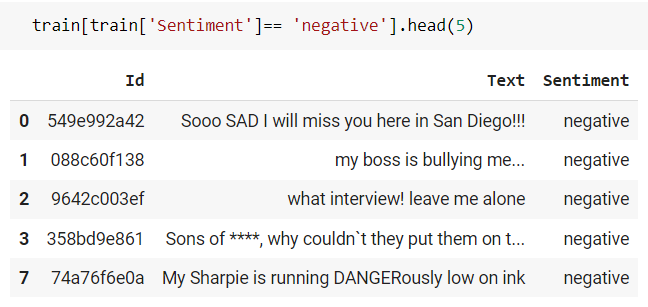
1. Data Inspection
2. Data Cleaning

## **Data Inspection**

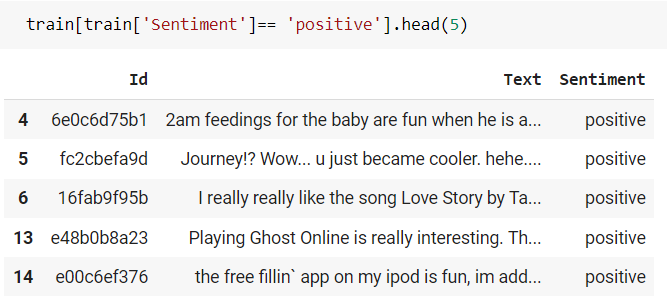
* Checking the first few rows of the training dataset.



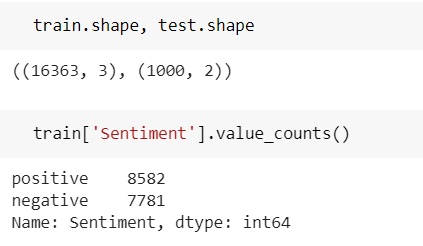
* Checking the first 5 rows of the Negative tweets in the training dataset



* Checking the first 5 rows of the Positive tweets in the training dataset

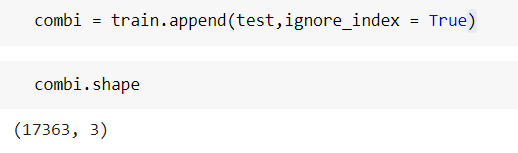


* Checking the Shape and Distribution of Tweets in the training dataset

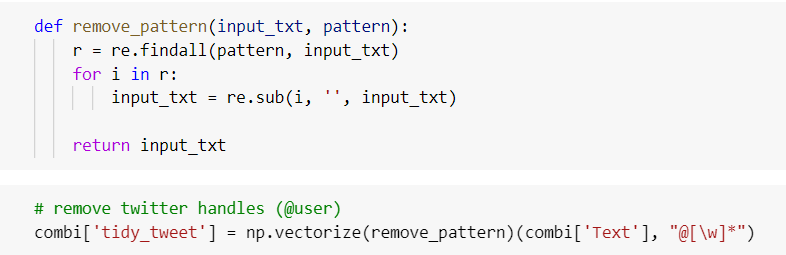


## **Data Cleaning**

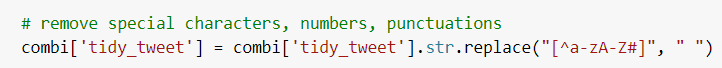
For our convenience, let’s first combine train and test sets. This saves the trouble of performing the same steps twice on test and train.



### **Removing Twitter Handles (@user)**



### **Removing Punctuations, Numbers, and Special Characters**



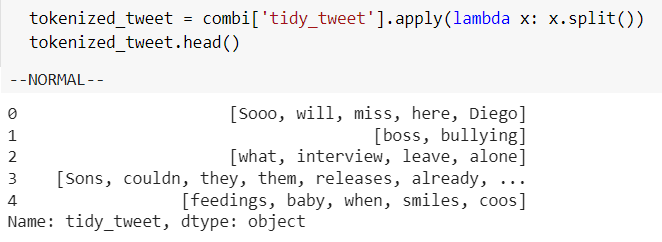
### **Removing Short Words**



We can see the difference between the actual tweets and the cleaned tweets (tidy\_tweet) quite clearly. Only the important words in the tweets have been retained and the noise (numbers, punctuations, and special characters) has been removed.

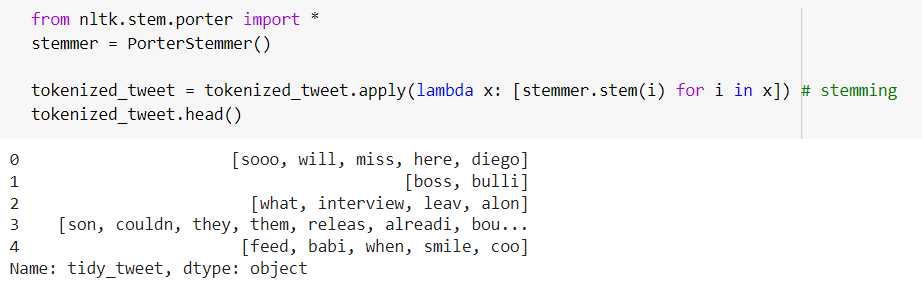
### **Tokenization**

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.



### **Text Normalization (Stemming)**

Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.



# **Data Exploration and Visualization from Tweets**

Exploring and visualizing data, no matter whether it's text or any other data is an essential step in gaining insights. we must think and ask questions related to the data in hand.

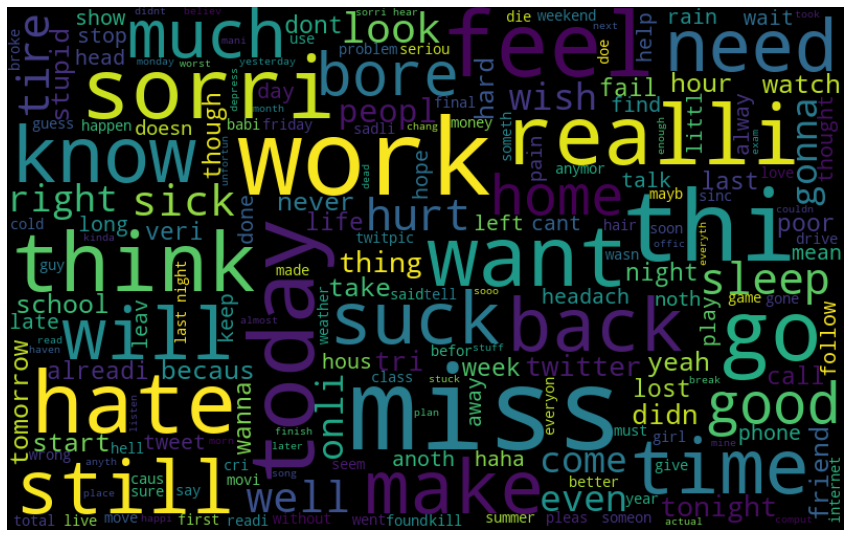
## **The common words used in the tweets: Word Cloud**



## **Words in Positive Tweets: Word Cloud**



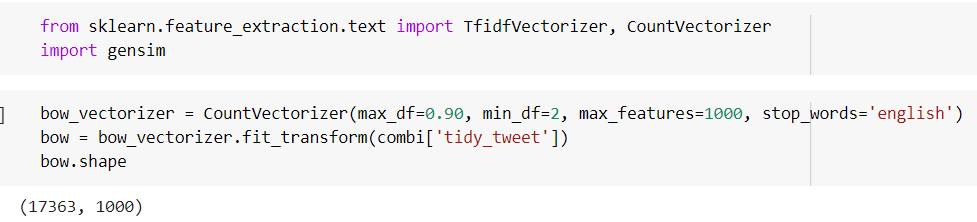
## **Words in Negative Tweets: Word Cloud**



# **Extracting Features from Cleaned Tweets**

To analyse pre-processed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted. In this blog, we will be covering Bag-of-Words and TF-IDF.

## **Bag-of-Words Features**



## **TF-IDF Features**



# **Model Building: Twitter Sentiment Analysis**

Now we will build predictive models on the dataset using the two-feature set — Bag-of-Words and TF-IDF.

## **Model Building using Bag-of-Words Features**

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. SVM Classifier
5. XGBoost Classifier

## **Model Building using TF-IDF Features**

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. SVM Classifier
5. XGBoost Classifier

## **Summary of Accuracy Scores on Training Data Set**

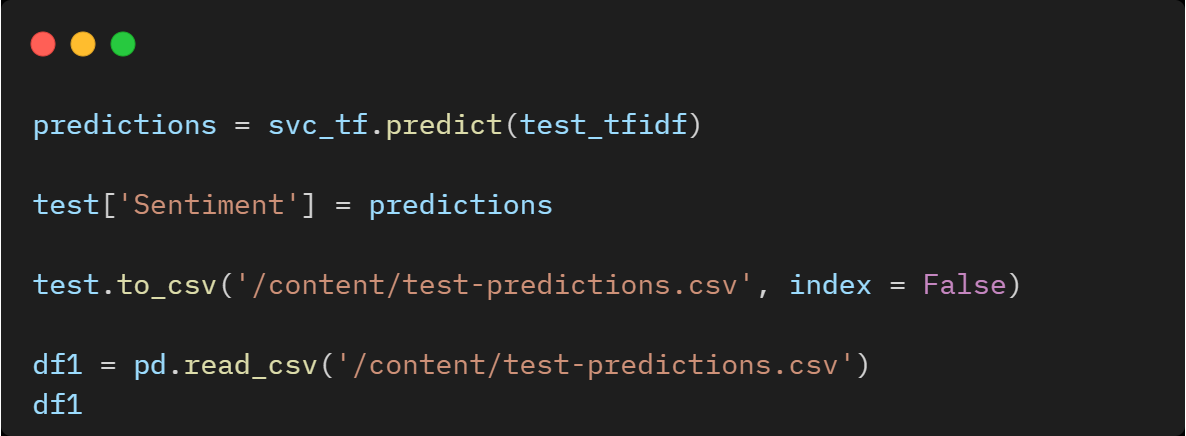
Below is the summary table showing accuracy scores on Training Data set for different Machine learning models and feature extraction techniques.

From all the models, the SVM classifier with TFI-IDF features achieved the best Accuracy of **94.648%** on Training data and 84.029% accuracy on validation data.

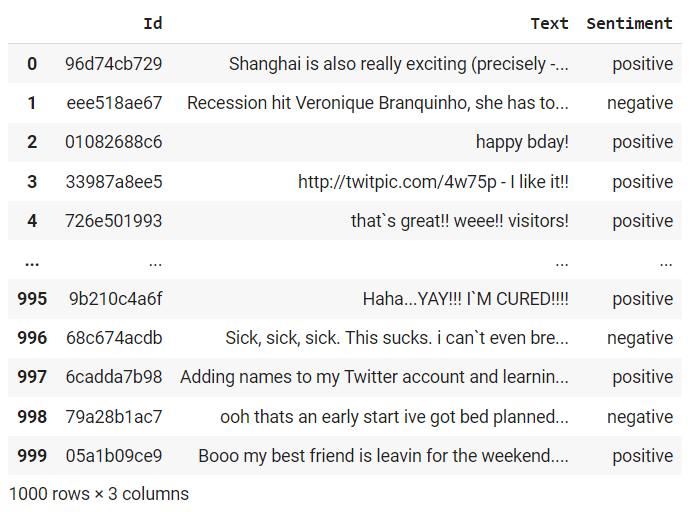
|  |  |  |  |
| --- | --- | --- | --- |
| Feature Extraction | Model | Training Accuracy | Validation Accuracy |
| Bag of Words | **Logistic Regression** | **87.515** | **83.662** |
|  | **Decision Tree Classifier** | **98.044** | **79.364** |
|  | **Random Forest Classifier** | **98.044** | **82.134** |
|  | **SVM Classifier** | **93.059** | **83.581** |
|  | **XGBoost Classifier** | **79.841** | **79.079** |
|  |  |  |  |
| TF-IDF | **Logistic Regression** | **87.401** | **83.907** |
|  | **Decision Tree Classifier** | **98.018** | **79.119** |
|  | **Random Forest Classifier** | **98.018** | **82.74** |
|  | **SVM Classifier** | **94.648** | **84.029** |
|  | **XGBoost Classifier** | **80.094** | **79.079** |

# **Predictions on Test Data set**

Using the best model SVM classifier, we will make predictions on the test data set and save the predictions to a .csv file named test-predictions.csv.



Test Data Predictions Data frame saved in a .csv file with columns Id, Text & Sentiment.



# **Conclusion**

In this blog, we have learnt how to approach a Sentiment Analysis problem. We started with pre-processing and exploration of data. Then we extracted features from the cleaned text using Bag-of-Words and TF-IDF.

Finally, we were able to apply different Machine Learning models using both the feature sets to classify the tweets and make predictions on the test data set using the best Machine learning model which outperformed the other models.

Source Code of the Project Link: [**GitHub**](https://github.com/ChaithanyaVamshi/Twitter_Sentiment_Analysis/blob/main/NLP_Project.ipynb)

# **References**

<https://www.analyticsvidhya.com/blog/2021/07/understanding-natural-language-processing-a-beginners-guide/>

<https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/>

<https://thecleverprogrammer.com/2021/09/13/twitter-sentiment-analysis-using-python/>