Course Number	CS 5593
Course Name	Data Mining
Semester, Year	Fall 2022

Title

Sentiment Analysis on Twitter Data

Group 3

Bhargav Durga Prasad Vummadi	bhargav.vummadi@ou.edu
Sai Prakash Davuluri	saiprakashdavuluri@ou.edu
Sai Sharath Kumar Sabbarapu	s.sharath.kumar@ou.edu

Sentiment Analysis of Twitter Data

Bhargav Durga Prasad Vummadi
University of Oklahoma
Norman, Oklahoma, USA
bhargav.vummadi@ou.edu

Sai Prakash Davuluri
University of Oklahoma
Norman, Oklahoma, USA
saiprakashdavuluri@ou.edu

Sai Sharath Kumar Sabbarapu
University of Oklahoma
Norman, Oklahoma, USA
s.sharath.kumar@ou.edu

Abstract

Twitter is a well-known social media platform where users can share information via tweets. People are free to express their ideas and opinions on any topic, including products, specific cases, etc. With the growth and evolution of web technologies, internet users now have access to a lot of data in the form of the web. Twitter users frequently share information, feelings, and sentiments about various aspects of life.

Sentiment analysis is the area of managing decisions and reactions, like emotions created through messages, widely used in areas such as social media analysis, web and data mining analysis Emotion is the most fundamental quality by which human behavior is judged. Sentiment analysis is based on and aims at discovering conclusions, describing the attitudes they convey, and finally classifying them. Surveys are first gathered When its mood is perceived, the highlights are selected, the mood is sequenced and finally the mood, the polarization is decided or determined.

One of the most significant benefits of the timely discovery of such sentimental or opinionated web content is monetization. The improvement of contextual advertising, recommendation engines, and market trend analysis can be enhanced by an understanding of how people feel about various entities and products.

We used a data set of 100,000 tweets to test different models. The models we used were Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes. Each model was tested on 20,000 new tweets comprised of equal distribution of positive and negative tweets. The best-performing model was then used to create a web application that can analyze real-time tweets to see if they are positive or negative.

Keywords: Support Vector Machine, K-Nearest Neighbours, Naive Bayes, Frequency Dictionary, Classification

ACM Reference Format:
Bhargav Durga Prasad Vummadi, Sai Prakash Davuluri, and Sai Sharath Kumar Sabbarapu. 2022. Sentiment Analysis of Twitter Data. In *Proceedings of December 05, 2022.* ACM, New York, NY, USA, 10 pages.

December 05, 2022, Norman, OK,
© 2022

1 Introduction

These days, social networking websites like Twitter, Facebook, and YouTube are extremely popular. Opinion mining, which includes sentimentality analysis, is a sub field of computational linguistics and data mining. Its primary objective is to infer from written documents a person's mood, conduct, and opinion.

When making a decision, the views of others can be crucial. People consider the prior experiences of their peers when their decisions depend on important resources. Political parties, for instance, could be curious to discover whether people agree with their educational philosophy. The most crucial point of view in the current situation has been the need to gather opinions from social networking sites and make judgments about what people like or detest.

The web application is inspired by the micro-blogging twitter concept [7] and provides the user with two input fields, one to enter the hashtag, which is some kind of Twitter slang[7], or the search key (any word, for example: "Elon Musk") and the other is to enter the number of tweets to retrieve. Here, we are using the social network scrape module from Python. Using this module, users can scrape data from almost every social network. After retrieving the tweets, the user can predict the sentiment of the tweets and can see the visual representation.
Note: Using the social network scrape module, users can get any number of tweets. But, the only constraint is that it takes a very long time to process the tweets and perform the prediction. Due to this, we are restricting the maximum number of tweets that can be entered should be less than 1000. Below are the functionalities that are available for the user:

1. Users can retrieve the latest tweets using hashtag or search key and view reports (polarity score, percentage distribution, etc) on the data.
2. Displaying the classified tweets, whether it is positive or negative.
3. Users can visualize the results in both tabular and graphical formats.

In our project we are using snscrape module and not the twitter API[7] because we have to authenticate and also there is a limit to the number of tweets retrieved monthly; this gives an advantage to our web application.

The main comparison from the previous design (literature review) [7] is that we are developing the model prior with the labeled data before testing it with the tweets from API. In our case, it is a snscrape module. We can achieve that speed by classifying the tweets.

2 Related Work

In paper [1], they compared Naïve Bayes‘ and K-NN algorithms for sentiment classification of movie and hotel reviews. The experimental results are measured using Accuracy, Recall, and Precision metrics. The Naive Bayes' approach has the following scores on each metric on movie reviews and hotel reviews; accuracy of 82.43 and 55.09; recall score of 80.12 and 51.84; precision of 81.01 and 61.11 on movie reviews and hotel reviews, respectively. On the other hand, the K-NN approach has the following scores; accuracy of 69.81 and 52.14; recall score of 61.81 and 46.31; precision of 66.73 and 56.77 on movie reviews and hotel reviews respectively. These results show that the classifiers yielded better results for the movie reviews, with the Naïve Bayes‘ approach giving above 80 percent accuracy and outperforming than the k-NN approach.

In paper [4], they performed sentiment analysis using Support Vector Machines (SVM). For preprocessing they used TF-IDF vectorizer, stemmer, stopwords handler an tokenizer. The results are measured in terms of precision, recall, and f-measure. According to the results, for the first dataset (consisting of tweets regarding self-driving cars), the average precision, recall, and f-measure are 55.8, 59.9, and 57.2, respectively. For the second dataset (consisting of tweets about apple products), the average Precision, Recall, and F-Measure are 70.2, 71.2, and 69.9, respectively. The results clearly show the dependency of SVM performance upon the input dataset.

3 Dataset and Pre-processing

3.1 Dataset

The dataset used is from Kaggle[13], which consists of 1.6 Million tweets. Due to the system limitations, we are only using 100,000 tweets comprising of equal distribution of positive and negative tweets. The pre-processing of data took 6 minutes for the complete dataset. This pre-processed data is then used to train the models. For validation, we are using 20,000 tweets of equal distribution. Based on performance metrics, one of the models is selected. This model is exported as a pickle file, to be used by the web application to perform prediction on real-time data obtained by the snscrape module.

3.2 Data Pre-processing

Tweets cannot be directly used for data mining since they contain noise, making it difficult to detect patterns or insights. To analyze tweets, data must be pre-processed, which removes unnecessary information and noise. This is accomplished in a series of phases, as seen in the flow chart below.

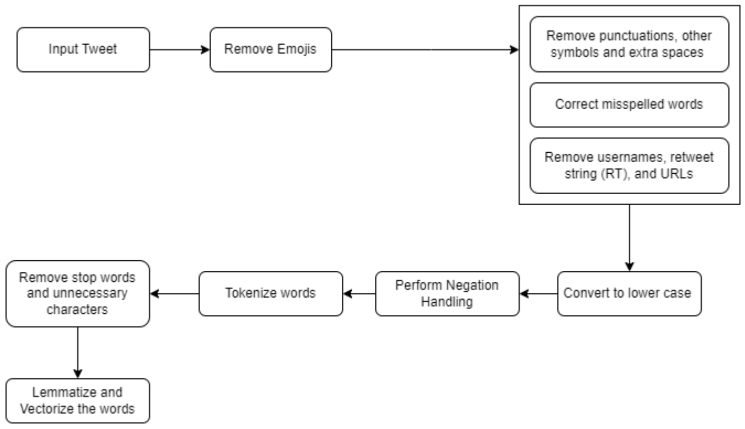

Input Tweet
Remove Emojis
Remove punctuations, other symbols and extra spaces
Correct misspelled words
Remove usernames, retweet string (RT), and URLs
Convert to lower case
Perform Negation Handling
Tokenize words
Remove stop words and unnecessary characters
Lemmatize and Vectorize the words

Figure 1. Pre-processing Flow

3.2.1 Data cleaning and Noise reduction.

The following pre-processing activities are done to remove noise in the data and converted into features:

1. Using regex the following have been removed:
   - Emojis
   - Punctuation
   - Duplicated characters based on repetitions
   - Usernames
   - RT (Re-tweet string)
   - Hyperlinks and URL
   - Extra spaces
   - Multiple periods i.e, (.......)
2. Contraction words such as "can't" have been replaced by "can not". This is done by defining a dictionary of such contraction words and replaced with its corresponding string in the text.
3. Using the existing set of stop words from the NLTK package, stop words such as a, an, the, have been removed.
4. The cleaned data is then lemmatized i.e., each word in the text is converted to a root word. For example, eat, eats, eating, ate, all have the same root 'eat'.

3.2.2 Feature Conversion.

For converting the cleaned data into features, we have used the TF-IDF Vectorization method. TF-IDF which stands for Term Frequency – Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document. Below is the formula to calculate Term Frequency

$$tf(w,d) = log(1 + f(w,d)) \tag{1}$$

where w is a word in the document, d is a document in the dataset, f(w,d) is the frequency of the word w in document d. Inverse Document Frequency (IDF) is a weight that represents how often a word is used. The more frequently it is used in a document, the lower its score. The lower the score, the less important the word becomes.

Below is the formula to calculate Inverse Document Frequency

$$idf(w, D) = log(\frac{N}{f(w, D)}) \tag{2}$$

Where N is the number of documents in the dataset and f(w,D) is the frequency of the word w in the complete dataset.

The final step would be to multiply both these values to get the TF-IDF score for a particular word.

This method is one of the most commonly used weighting metrics for measuring the relationship of words to documents and is widely used for word feature extraction.

The following steps were taken to convert the data into features:

1. Collected the unique set of words from each tweet in the dataset
2. Created the word dictionary containing the word and it's occurrence in dataset
3. Calculated the term frequency and inverse document frequency using the above formulae

Performing this task manually on 20,000 tweets took over 3 hours. Given that the selected dataset is large, we have used scikit-learn package's implementation of the TF-IDF Vectorizer which took about 20 minutes to complete.

3.3 Classification Algorithms

To classify the tweets, we have implemented the below algorithms

3.3.1 Support Vector Machine[14]. SVM is a machine learning tool that analyzes data and identifies patterns or decision boundaries in datasets, primarily for classification and regression analysis. SVM constructs a hyperplane in a multidimensional space that separates the boundaries of different classes, and the dimensionality is called the feature vector of the data set. SVM has the ability to handle multiple continuous and categorical variables.

In the Support Vector Machine classification model, the main idea is to use a linear model, which is best suited for our case to classify between positive and negative classes. Using the linear model, we then try to find a linear decision boundary hyperplane that best separates the data points. The best hyperplane is the one that yields the largest separation or margin between both the positive and negative classes. Finally, we will choose the hyperplane in such a way that the distance from it to the nearest data point on each side is maximized.

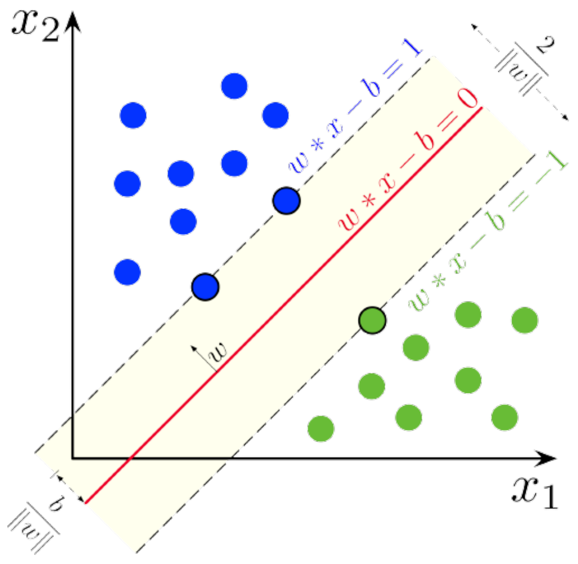


Figure 2. SVM distribution https://en.wikipedia.org/wiki/Support_vector_machine

From the above figure we can see the decision boundary and positive tweets are labeled as 1 and negative tweets as -1. Let's look at an example to make this more clear, so here we have our feature vectors in 2D with two classes blue and green We want to find a linear decision boundary so, in this case this idea is simply a line that has the largest margin between both classes so the distance from this decision boundary to the nearest point on each side is maximized and these nearest points are also called the support vectors so if we describe this in a mathematical way then here we have this line equation w * x - b = 0 at the point of the decision boundary. Here is the weight and b is the bias.
On the top side of decision boundary where the class is plus one, we have the w*x-b =1 and then minus one on the other side where the class is -1, we have w*x-b = -1 and then this margin here this distance here can be calculated with:

$$\frac{2}{||w||}$$

So, we have to learn this weights and bias and corresponding categories or labels while training and try to maximize the margin from the hyperplane this can be called as hard margin and mathematically represented as:

$$w.x_i - b >= 1 -> if y_i = 1 - For positive Tweets$$

$$w.x_i - b <= -1 -> if y_i = -1 - For negative Tweets$$

Generalizing the formula we can get following:

$$y_i(w.x_i - b) >= 1$$

where y ϵ -1,1 For all the operations we will be performing dot operations. For initial weight assignment we will be using a technique called Hinge loss. Where we assign weight as zero if we are on correct side of hyperplane otherwise we use following formula:

$$l = max(0, 1 - y_i(w \cdot x_i - b))$$

$$l = \begin{cases} 0 & \text{if } y \cdot f(x) \geq 1 \\ 1 - y \cdot f(x) & \text{otherwise} \end{cases}$$

If a data point is further from the decision boundary higher will be the loss. We will be using regularization technique by using lambda parameter as follows: Finally we will be using

$$J = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i - b))$$

if $y_i \cdot f(x) \geq 1$:

$$J_i = \lambda \|w\|^2$$

else:

$$J_i = \lambda \|w\|^2 + 1 - y_i(w \cdot x_i - b)$$

gradients which is finding derivatives accordingly and we will be updating the bias and weights using the following formulas: The algorithm uses all the formulas mentioned

if $y_i \cdot f(x) \geq 1$:

$$w = w - \alpha \cdot dw = w - \alpha \cdot 2\lambda w$$

$$b = b - \alpha \cdot db = b$$

else:

$$w = w - \alpha \cdot dw = w - \alpha \cdot (2\lambda w - y_i \cdot x_i)$$

$$b = b - \alpha \cdot db = b - \alpha \cdot y_i$$

above and here is the flow of the classifier we implemented:

1. Initialize number of iterations in which classifier runs and updates the weights, learning rate, lambda parameter and making initial weight and bias as None.
2. Maps the class labels to -1 and 1.
3. Now every data point is sent for constraint checking as mentioned above.
4. Respective gradients are calculated.
5. Finally weights are updated again using the above formula with the help of numpy module dot operation.
6. Finally we can easily predict the class label using the following formula:

$$weight.x + bias$$

We get the sign of value obtained from above value and return corresponding prediction. We have tried different learning rates and lambda parameter to increase the accuracy. We have used equally distributed positive and negative tweets i.e, 50000 positive and 50000 negative. The train data consists of 80000 tweets and test data of 20000. Finally we got the best at learning rate of 1e-4 and lambda of 1e-2. The confusion matrix after getting predictions on 20 percent test data is:

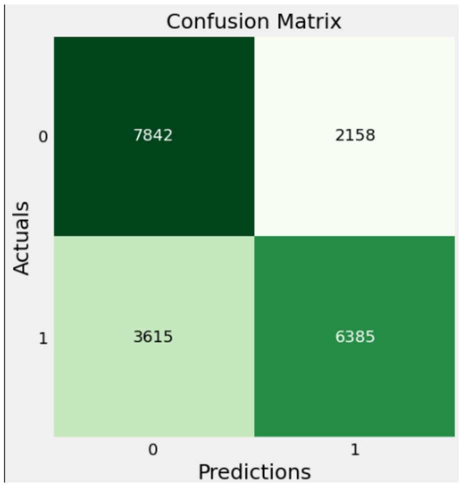


Figure 3. SVM confusion matrix

Highest accuracy we got is: 0.71135.

3.3.2 K Nearest Neighbors[10].

The k-Nearest-Neighbours (kNN) is a supervised machine learning algorithm. For a data record X to be classified, its k nearest neighbours are to be found.

There are multiple ways to calculate the distance between data points. Below are few methods available:

- Minkowski Distance
- Manhattan Distance
- Euclidean Distance
- Cosine Distance

Among these, Euclidean distance is the most widely used method and has been used for this experiment as well. The Euclidean distance between points A and B is calculated as shown below

$$distance = \sqrt{(a_0 - b_0)^2 + (a_1 - b_1)^2 + (a_2 - b_2)^2 + ...(a_n - b_n)^2} \quad (3)$$

Based on this euclidean distance, we sort then in ascending order to find the data points nearest to X. The most recurring labels of these k neighbors would be applied to X. For a classification example, in a group of 10 neighbors 4 data points have a positive label and the remaining 6 data points have a negative label. Since most of the neighbors have negative labels, the algorithm predicts that the data point X will have a negative label. However, k must first be determined in order to apply kNN, and the classification's outcome is greatly influenced by this number. There are many ways to find the value of k, but a straightforward way is to run the algorithm repeatedly with different values of k and choose the one that gives the best results. Another approach is to use cross-validation for hyper parameter tuning, where a range of k values is provided and the one that gives the smallest error is chosen. As kNN is a lazy learning learning method, the model does not learn when training data is provided and only stores it. The training data is only used when the model performs a prediction. This prohibits it in many applications such as dynamic web mining for a large repositories.

Below are the steps taken to implement this classifier:

1. Create a list to hold the nearest neighbors.
2. Use the euclidean distance method to calculate the distance between the data point and the training data.
3. Sort the distances in ascending order along with their corresponding label.
4. Pick the top k neighbors and store them in the list created in step 1.
5. Among these neighbors, the most recurring label is selected as the prediction for the data point

We have tried different k values on the dataset (20,000 tweets comprising equally distributed positive and negative tweets) with training data consisting of 16,000 tweets and testing data of 4,000 tweets. As the algorithm is compute intensive, the algorithm ran over 8 hours for higher k values (greater than 10) and failed at a point where the system was unable to allocate enough memory to perform the calculations. Due to this restriction, we had to use lower k values with a fraction of the dataset. which led to lower accuracy scores.

Below is the confusion matrix for this model:

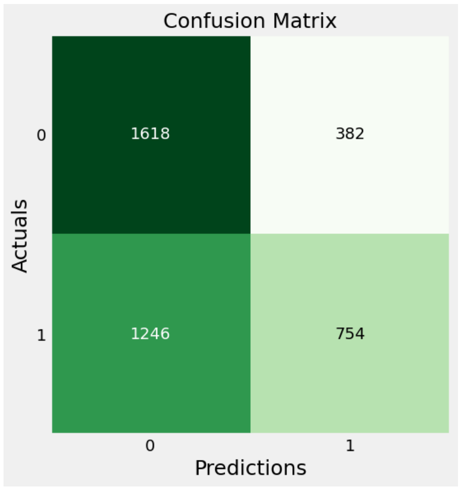


Figure 4. KNN confusion matrix

The highest accuracy achieved is 0.593 with k = 2 which took around 82 minutes to run.

3.3.3 Naive Bayes. This algorithm is based on the Bayes Theorem, it is used as a probabilistic machine learning algorithm and it is used for various classification tasks and also be used for clustering. Bayes' Theorem is a simple mathematical formula used for calculating conditional probabilities. Conditional Probability is a probability measure of an event based on the criteria that another event has already been occurred previously.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \tag{4}$$

Where P(A) is the probability of A occurring and P(B) is the probability of B. Similarly, P(A|B) is the probability of A occurring given evidence that B has already occurred and P(B|A) is the probability of B occurring given evidence that A has already occurred

Naïve Bayes algorithms can be used in sentiment analysis, spam filtering, recommendation systems, etc. advantages of Naïve Bayes algorithm: they are quick and easy to implement.

Disadvantage They can be the necessity of predictors for them to be independent.

The name of this algorithm came because for every hypothesis, to make the calculations easy they probabilities were simplified. Each value in an attribute are assumed to be independent. Independent value can be represented as (P(d1, d2, d3|h). Probability is calculated as P(d1|h) * P(d2|H)

Representation Used By Naive Bayes Models
We use probabilities for the Naïve Bayes representation. Below are the probabilities that are stored for a Naïve Bayes classifier :

- Class Probabilities: The probabilities of each class in the training dataset
- Conditional Probabilities: The conditional probabilities of each input value given each class value

1. Multinomial Naïve Bayes Classifier
2. Bernoulli Naïve Bayes Classifier
3. Gaussian Naïve Bayes Classifier

Below are the steps taken to implement this classifier:

1. First we are instantiating the class
2. Class separation to know the prior probability of the class. Here we separate the classes and use a dictionary to save the values. This way, we are assigning the feature values to the specified class.
3. Probability is calculated based on the mean and standard deviation
4. We fit the model with the training data and label the data as input. We create a dictionary containing input data separated by class label as in the second step.

5. We predict the class based on the posterior probability (joint probability / marginal probability). The class with maximum posterior probability value is selected and predicted
6. We calculate the accuracy (correct predictions / total predictions) to measure the model's performance.

Below is the confusion matrix for this model:

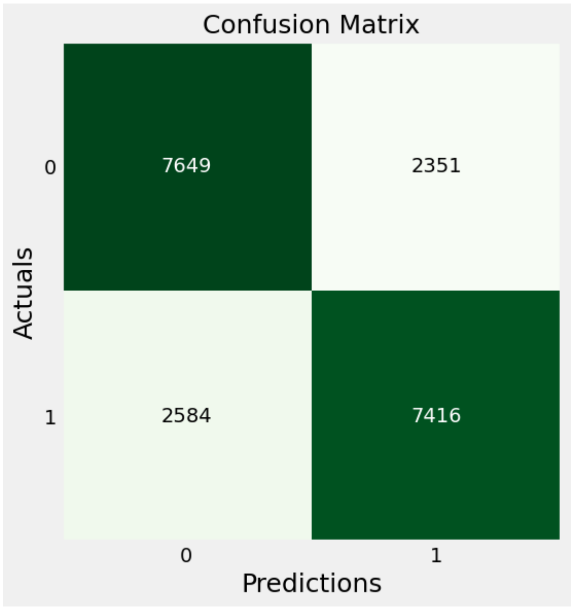


Figure 5. Naive Bayes confusion matrix

3.3.4 Observations and Performance metrics.

We have considered the following performance metrics for all the three algorithms:

- Accuracy
- Precision
- Recall
- F1 Score

For performance method we have **K Fold cross validation**. Where k = 3 , we have kept the k value low because of the longer run times and performance of our systems are not supported for such longer runs.
SVM For SVM ,the regularization gave higher advantage over all other models. As we are using linear classifier which gave an edge as the data has binary classifier values which are 1 and -1. The running time increase exponentially when we increased number of iterations more than 1000, So we kept the iterations to 1000 , which took an average running time of 32 minutes. The performance metrics are as follows:

- Accuracy: 0.71135
- Precision : 0.74739
- Recall : 0.6385
- F1 Score : 0.6886

For K-Fold cross validation we have used StratifiedKFold(n-splits=3) to make the required splits for different folds. For the different folds we achieved the following results:
Fold: 1, Accuracy: 0.684
Cross-Validation accuracy: 0.684 +/- 0.000
Fold: 2, Accuracy: 0.621
Cross-Validation accuracy: 0.652 +/- 0.031
Fold: 3, Accuracy: 0.717
Cross-Validation accuracy: 0.674 +/- 0.040

KNN
For KNN, we have experimented with various k values. With the increase of k value, the algorithm took longer to run. Due to hardware restrictions, we were unable to test the model with higher k values. Below is the plot of performance comparison for different k values:

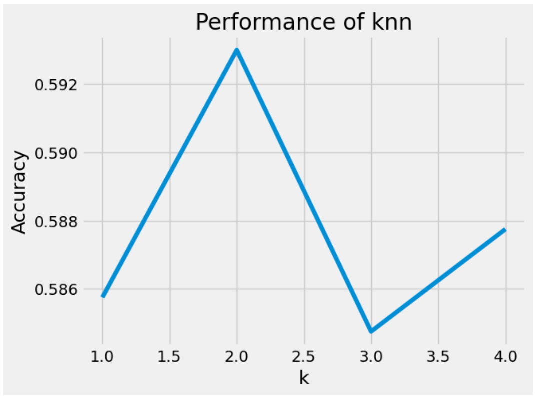


Figure 6. KNN Performance Comparison

It can be seen that for a k value of 2, the model achieved the highest accuracy and as k value increased, the accuracy decreased but the model could have performed better with higher k values and by using the complete dataset. The performance metrics for KNN are as follows:

- Accuracy: 0.593
- Precision : 0.663
- Recall : 0.377
- F1 Score : 0.480

The accuracy of the KNN model with 2 neighbors with n-splits = 3 are as follows: Fold: 1, Accuracy: 0.575
Cross-Validation accuracy: 0.575 +/- 0.000
Fold: 2, Accuracy: 0.576
Cross-Validation accuracy: 0.575 +/- 0.001
Fold: 3, Accuracy: 0.574
Cross-Validation accuracy: 0.575 +/- 0.001

Naive Bayes
The performance metrics for Naive Bayes are as follows:

- Accuracy: 0.500
- Precision : 0.747

- Recall : 0.764
- F1 Score : 0.531

Performing K fold cross validation using Naive Bayes resulted in the following accuracy values : Fold: 1, Accuracy: 0.487
Cross-Validation accuracy: 0.487 +/- 0.000
Fold: 2, Accuracy: 0.489
Cross-Validation accuracy: 0.487 +/- 0.002
Fold: 3, Accuracy: 0.0.484
Cross-Validation accuracy: 0.487 +/- 0.002

3.4 Application Architecture

As there are many web frameworks and different tech stacks available, we decided to use HTML5, CSS3, ES6 and bootstrap 5 for maintaining good user experience and responsiveness. For backend , flask turned out to be best for simple framework for deploying small scale machine learning model based application. We have used jinja-2 syntax for representing data that is coming from backend.

We have leveraged the MVC architecture which is widely used in different large scale web applications. Where M is model which is flask .py files in our case and V is view which is interactive web pages and C is Controller where js is used to interact between the backend and frontend through HTTP.

Here is our entire architecture in a snapshot:

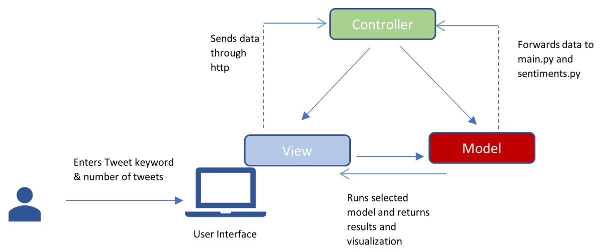

Controller
Sends data through http
Forwards data to main.py and sentiments.py
View
Model
Enters Tweet keyword & number of tweets
User Interface
Runs selected model and returns results and visualization

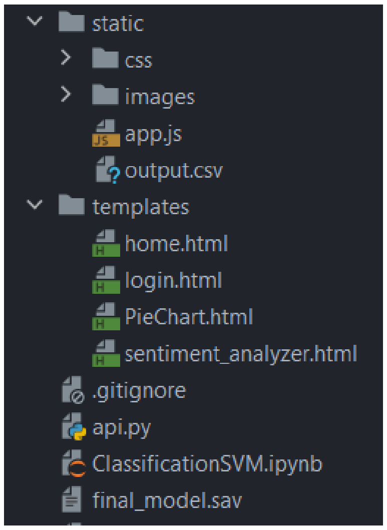
Figure 7. MVC Architecture

For entire web application development we have used pycharm which is most preferable when using python web frameworks for backend. It provides proper debugging and logging tools for seamless development.

Finally after considering the performance metrics and evaluation method we have utilized svm model for our final application. We pickeled both vectorizer and model and used in the web application.

3.5 Application Setup

For setting up the web application, we used the flask folder structure and it should be maintained while setting up.



```
static
    css
    images
    app.js
    output.csv
templates
    home.html
    login.html
    PieChart.html
    sentiment_analyzer.html
.gitignore
api.py
ClassificationSVM.ipynb
final_model.sav
```

Figure 8. Flask folder structer

3.5.1 Installation.

Install pycharm from jetbrains and maintain the folder structure as shown in the above image, which is all the images, .css, .js files should be inside the static folder, .html files should be in templates folder and all .py files , .sav pickle files should be under the root folder (we can get pickle files from this links Model.sav: https://tinyurl.com/ys425ebk and
vectorizer.sav: https://tinyurl.com/33yuax4m). Python should be installed and path is set . Configure the python interpreter inside the pycharm by clicking the setting from the options panel. Make the main.py as default runner by configuring it under run configurations on the right top corner of the pycharm.

All the required CDN links are configured inside the HTML files, so there is no need for advance installations. There we are left installation of only required python packages as:

- snscrape
- pandas
- pickle
- NLTK
- Flask
- wordcloud
- requests

These are some of the important packages and all other can be located inside the requirements.txt file. We can install all the packages using the following pip command:
pip install -r requirements.txt
After installing all the required packages and configuring main.py click on run otherwise we can directly right click on main.py and run which opens localhost at port 5000 which is default running port for flask application.

3.5.2 User Manual. After following above instructions users can see the landing page of our web application. We have included an inbuilt 3d spline model for beautiful user experience. We can rotate it in 360 degree. Here is the snapshot:

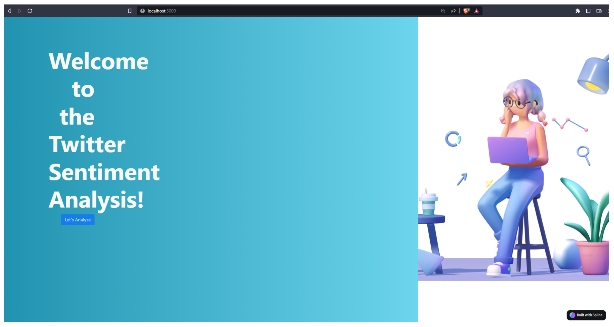


Figure 9. Landing page

Users have to click on Let's Analyze button to go to the home page of our web application. In the homepage, user is provided with navigation menu between tweet analyze section and visualize section. Home page consists of carousel displaying all the necessary information with. When we click on Tweet Analyze or button on the carousel it will redirect to the input form. Here is the snapshot of the home page:

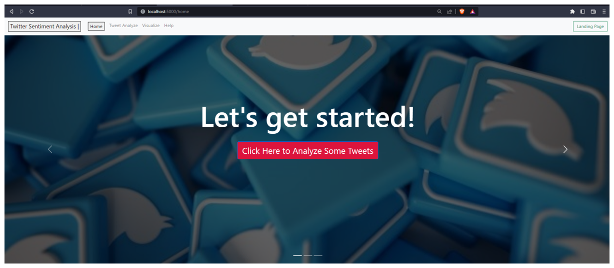
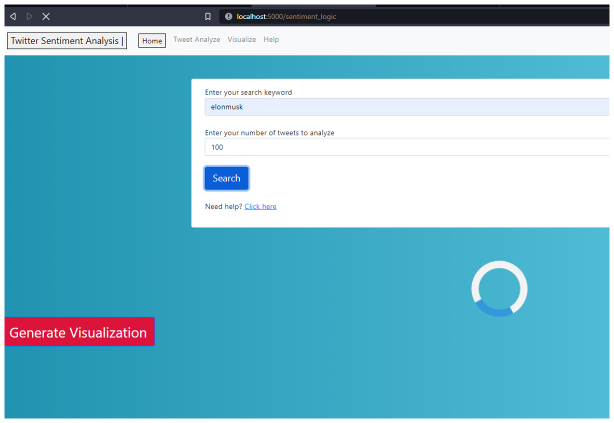


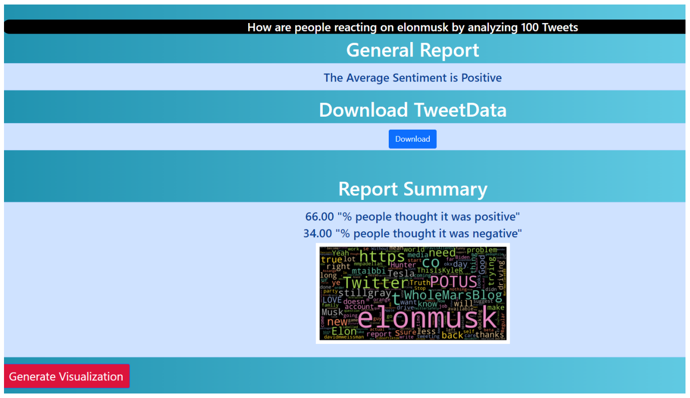
Figure 10. Home page

In the tweet analyze page user will be given an option to enter the search keyword and number of tweets to be scrapped over the internet which is done from the scrapesn python module. For example enter elonmusk and number of tweets as 100 and clicking on the search will run the svm model in the background and returns the relevant report.
Here is the screenshot of analyzing tweets [Figure 11]:
Finally after completion of running model here is the relevant analysis report. Tweets are categorized as positive and negative, and percentages are calculated also a word cloud is generated with all the tweets. Users also provided with download option to csv report of all the tweets fetched.[Figure 12]



Enter your search keyword
elonmusk
Enter your number of tweets to analyze
100
Need help? Click here

Figure 11. Tweet Analysis



How are people reacting on elonmusk by analyzing 100 Tweets
The Average Sentiment is Positive
66.00 "% people thought it was positive"
34.00 "% people thought it was negative"

Figure 12. Tweet Analysis Report

User can also get beautiful visualizations which are plotted using plotly library. We can click on generate visualization button after displaying of the results. It takes some time to load all the visualizations again loader is displayed while loading the visualizations.

After the loading the visualizations page user can have three sub sections or tabs. Initially overall tweet information is visualized in dashboard format. Here is the snapshot of visualization overview tabs.[Figure 13] .

In this page user can view, the keyword,number of positive tweets, number of negative tweets. Bar graph and relevant pie chart distribution is displayed. Also a scrollable table is provided to show all the scrapped tweets information. On the last card item the positive word is the most used positive word which is calculated by grouping positive tweets and same with the negative word.

After that when we click on positive tweets tab we will get the following visualizations:[Figure 14]

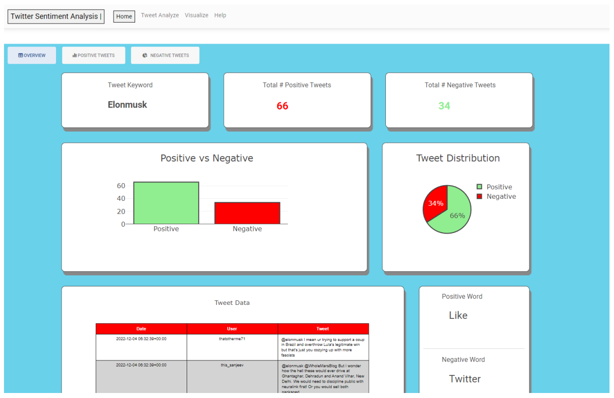
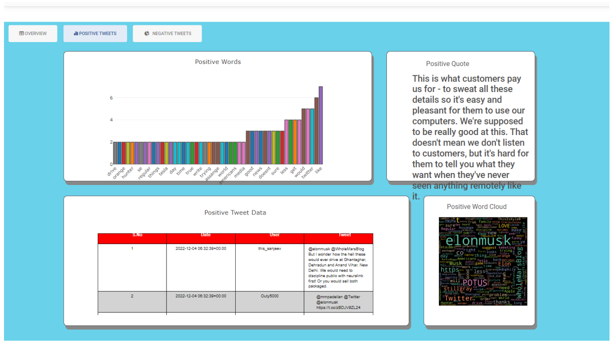


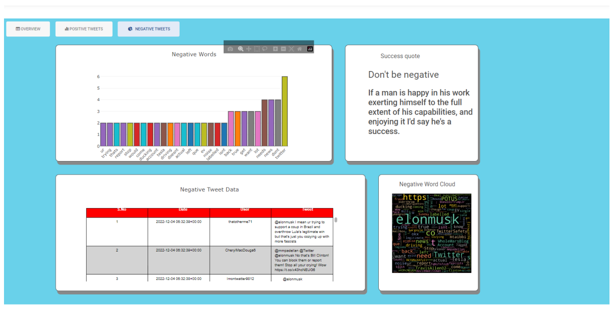
Figure 13. Visualizations Overview page



This is what customers pay us for - to sweat all these details so it's easy and pleasant for them to use our computers. We're supposed to be really good at this. That doesn't mean we don't listen to customers, but it's hard for them to tell you what they want when they've never seen anything remotely like it.

Figure 14. Positive Tweet Visualizations page

This page[Figure 14] shows the positive word count distributions using bar plots and ninja api is used to get positive quote every time we get visualization. We can also get positive tweet data in table and a positive word cloud. Similar information is visualized for negative tweets. Here is the snapshot:



Don't be negative

If a man is happy in his work exerting himself to the full extent of his capabilities, and enjoying it I'd say he's a success.

Figure 15. Negative Tweet Visualizations page

Note: Files submitted to canvas have Group3 as a prefix, please remove this prefix to run the project properly.

4 Conclusion and Future Work

For this project, we have implemented several algorithms to perform sentiment analysis. We have used several techniques for pre-processing the data such as regular expressions to remove noise, negation handling to remove contraction words and then preformed lemmatization on words. For feature conversion, we specifically selected TF-IDF Vectorizer as it takes into account the recurrence of words and its importance in the dataset. We implemented and tested the accuracy of the selected algorithms. The results show that SVM performed relatively better than the remaining algorithms. To compare the algorithms, we also used K-Fold Cross validation, Confusion Matrix, Accuracy, Precision, Recollect and F1 Score metrics. From our experiment, SVM has greater accuracy of 0.71, an average run time of 32 minutes and KNN had an accuracy of 0.59 and an average run time of 82 minutes. On the other hand, Naive Bayes had the least accuracy of 0.53 and an average run time of 46 minutes.

For future work, better pre-processing techniques can be added on top of the selected techniques and other ways to perform feature conversions can be used such as CountVectorizer and Bag Of Words. Also, KNN used only a fraction of the data selected due to hardware limitations. If we have access to better hardware, we can improve the KNN model by performing grid search with different distance metrics and different approaches to compute the nearest neighbors. Likewise, SVMs can be improved using grid search with different learning rates, maximum number of iterations, and values of lambda. These improvements would also be dependent on having better hardware. Naive Bayes algorithm can also be improved by providing more data of better quality which would include in more preprocessing steps such as stemming, other approaches to correcting spellings. Also, experimenting with MultinomialNB could lead to better results.

References

1. Lopamudra Dey, Sanjay Chakraborty, Anuraag Biswas, Beepa Bose, Sweta Tiwari. Sentiment Analysis of Review Datasets Using Naïve Bayes‘ and K-NN Classifier,International Journal of Information Engineering and Electronic Business,Volume: 8, Number: 4, July 2016 .
2. Ajay Rao, Varun Kanade, Chinmay Motarwar, Prof. Shital Girme. Election Result Prediction using Twitter Analysis, International Research Journal of Engineering and Technology (IRJET), Volume: 09, Issue:05, May 2022.
3. Tahreem Zahra, Dr Hamid Ghous, Iqra Hussain. SENTIMENT ANALYSIS OF TWITTER DATASET USING LLE AND CLASSIFICATION METHODS, International Research Journal of Modernization in Engineering

Technology and Science, Volume:03/Issue:01/January-2021.
4. Ahmad Munir, Aftab Shabib and Ali Iftikhar, Sentiment Analysis of Tweets using SVM, International Journal of Computer Applications (0975 – 8887), Nov, 2017.
5. Atharva Mashalkar. Sentiment Analysis using Logistic Regression and Naive Bayes. Nov, 2020 https://towardsdatascience.com/sentiment-analysis-using-logistic-regression-and-naive-bayes-16b806eb4c4b 9/17/2022
6. Saket Garodia. Twitter Sentiment Analysis. Dec 2019 https://medium.com/analytics-vidhya/twitter-sentiment-analysis-134553698978 9/17/2022
7. Mustafa Bashir B El-Khunni. Visualizing in Twitter Sentiment to Measure Consumer Insight,publication 273447919. September 2013.
8. Yanwei Bao, Changqin Quan, Lijuan Wang1 and Fuji Ren. The Role of Pre-processing in Twitter Sentiment Analysis, Researchgate publication 339982951. August 2014.
9. Yan Zhang , Yue Zhou , and JingTao Yao. Feature Extraction with TF-IDF and Game-Theoretic Shadowed Sets,Communications in Computer and Information Science volume 1237. June 2020.
10. Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer, KNN Model-Based Approach in Classification, August 2004
11. Suman Rani, Jaswinder Singh.SENTIMENT ANALYSIS OF TWEETS USING SUPPORT VECTOR MACHINE.International Journal of Computer Science and Mobile Applications,Vol.5 Issue. 10, October- 2017.
12. Rasika Wagh, Payal Punde, Dr. BAMU, Survey on Sentiment Analysis using Twitter Dataset, Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018)
13. https://www.kaggle.com/datasets/kazanova/sentiment140
14. Sourish Ghosh, Anasuya Dasgupta, Aleena Swetapadma. A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. International Conference on Intelligent Sustainable Systems IEEE Xplore Part Number: CFP19M19-ART; ISBN: 978-1-5386-7799-5. (ICISS 2019).
15. İhsan Rıza Kara, Asaf Varol. Detection of Network Anomalies with Machine Learning Methods 978-1-6654-9796-1/22 ©2022 IEEE

Received 05 December 2022