2022

Sentiment Extraction and Analysis of Tweets using various Models and Ensembles

SUPERVISED BY DR. MOBASHER

RAWAN HAMMAD, SAURIN PATEL, ANISHA PATEL

DEPAUL UNIVERSITY | Chicago, IL

Rawan Hammad
Saurin Patel
Anisha Patel

DEPAUL UNIVERSITY

1. Introduction

1.1 Context

This project has been done as a part of DSC Programming Machine Learning course at DePaul University, supervised by Dr. Bamshad Mobasher. This was a group project by Rawam Hammad, Saurin Patel, and Anisha Patel.

1.2 Motivation

We chose to investigate the sentiment analysis field of machine learning. It allows to approach natural language processing which is a very hot topic. Being happy is quite important factor of life, and it depends highly on our surroundings. We wanted to see how happy the citizens are.

1.3 Idea

The growth of social networks like Twitter, Facebook, LinkedIn, Google+, Pinterest and others, has generated a large amount of information about the preferences and behaviors of users. Text from users' posts could be analyzed to extract sentiment that could be used for marketing or advertising purposes, for example. One of the areas that have surged in this field is Natural Language Processing (NLP) which allows us to analyze text on the web or in documents to mine sentiment. Sentiment analysis, also called opinion mining, which was initially focused on electronic commerce (or e-commerce), is nowadays popular in a variety of fields like education, medicine, politics; etc. In this project, we will build and explore various models to predict the sentiment of each tweet based on its text.

1.4 Sources

We got our dataset from https://www.kaggle.com/datasets/kazanova/sentiment140

2. Project

Sentiment analysis, also refers as opinion mining, is a sub machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive or negative. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to working on. In this project we choose to try to classify tweets from Twitter into "positive" or "negative" sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending a tweet limited by 140 characters.

2.1 Dependencies

The Jupyter Notebook depends on the following libraries/ packages: Pandas, Numpy, Matplotlib, WordCloud, Sklearn, Seaborn, Re, Warnings, Nltk, Gensim, Operator, Pickle, and XGBoost. The Notebook also depends on a locally saved csv file of original data used within the code.

2.2 Exploratory Discussion

The dataset has 6 attributes and 1.6 million tweets. Briefly, the dataset contains the following 6 fields:

- target: the polarity of the tweet (0 = negative, 4 = positive)
- ids: The id of the tweet (2087)
- date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- flag: The query (lyx). If there is no query, then this value is NO_QUERY.
- user: the user that tweeted (robotickilldozr)
- text: the text of the tweet ("*Lyx is cool*")

Based on our preprocessing, we found that there is no null values in the dataset, and there is a total of 1,600,000 non-null entries within each column listed above. The target attribute is categorized into either 0 or 4, where 0 is negative and 4 is positive, and are split evenly into 800,000 of each sentiment. We noticed that there are no neutral values in the target values.
We have done some additional basic analysis to explore the timeline of those tweets. The dataset dates to 2009. The maximum number of tweets contained in this dataset dates to April of 2009, followed by May then June. Most tweets are posted on Mondays, with Thursdays being the least active day.
Since the goal is to extract the sentiment from the tweets, we decided to create a word cloud for each negative and positive set of tweets. We also investigated the common hashtags, with followfriday being the most common positive hashtag, and squarespace being the most common negative hashtag.

2.3 Preprocessing

The following tasks were completed in the preprocessing step:

- Converted all tweets to lowercase
- Replaced usernames with empty strings
- Replaced tags with empty strings
- Removed any URLs and links
- Removed emojis
- Removed special characters
- Removed some consecutive letters, such as additional letters in 'heyyy'
- Remove multiple spacing
- Tokenized tweets by removing stop words, lemmatizing the words, and stemming them

We also created a word to vector model to handle word embeddings.

3. Evaluation:

To evaluate a model, we used confusion matrix, classification matrix, accuracy, precision, recall, and cross validation. For the cross validation, we had 80:20 ratio for training and testing.

Rawan Hammad
Saurin Patel
Anisha Patel

4. Modeling

While the dataset holds 1.6 million tweets, only 500,000 tweets were utilized due to limited computing power and time constraints. The tweets were split into 90% to 10% for training and testing, with a total of 450,000 tweets used to build the models. TF-IDF vectorizing and transformation was also applied to the training set prior to the modeling step.
The models created and evaluated in this project include:

4.1 Logistic Regression

We started by running a baseline model of the logistic regression with max_iter = 1000. Next, we performed hyperparameter tuning using grid search, then reran the logistic regression using the best parameters, max_iter = 1000, c = 1.0, and penalty = l2.

4.2 Naïve Bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive)independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum - likelihood training can be done by evaluating a closed form expression. The Multivariate Bernoulli Model, also called binomial model, useful as our feature vectors are binary. We went with the baseline Naïve Bayes with alpha of 2.

4.3 SVM

While we did not apply SVM to our final results, it is important to note that we did test with it. However, due to its runtimes inefficiencies, we decided to exclude SVM from this assessment.

4.4 Decision Trees

We used the baseline decision tree model in sklearn

4.5 kNN

kNN was first explored at n =3 and distance weights, we then decided to compare this to Rocchio due to the low accuracy results we got from kNN alone.

4.6 Rocchio

The Nearest Centroid (Rocchio) method was created to test against kNN's weak performance.

4.7 XGBoost

The baseline and tuned XGBoost ensemble was ran. We used randomized search CV to find the best parameters and reran the ensemble using objective = binary:logistic, max_depth = 10, n_estimators = 103, and learning_rate = 0.1.

4.8 Random Forest

Baseline Random Forest and a tuned version were run. The best parameters selected were gini, max_depth = 8, max_features = auto, and n_estimators = 300.

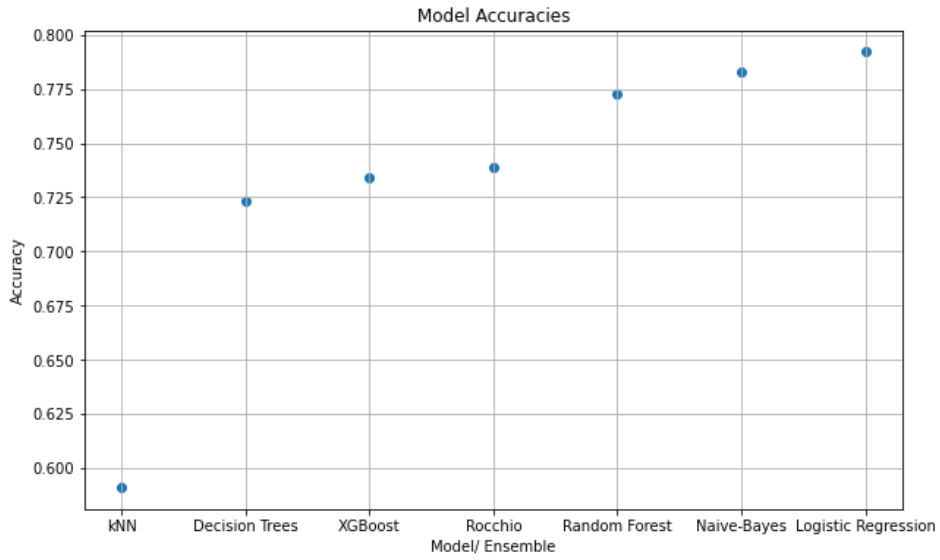
A few functions were also written to evaluate the models by predicting values, printing the evaluation metrics, plotting a confusion matrix, and printing accuracies.

Rawan Hammad
Saurin Patel
Anisha Patel

Model	Accuracy
Logistic Regression	79%
Bernoulli's Naïve Bayes	78%
Decision Tree	72%
KNN	60%
Rocchio Method	74%
XGBoost	73%
Random Forest	77%

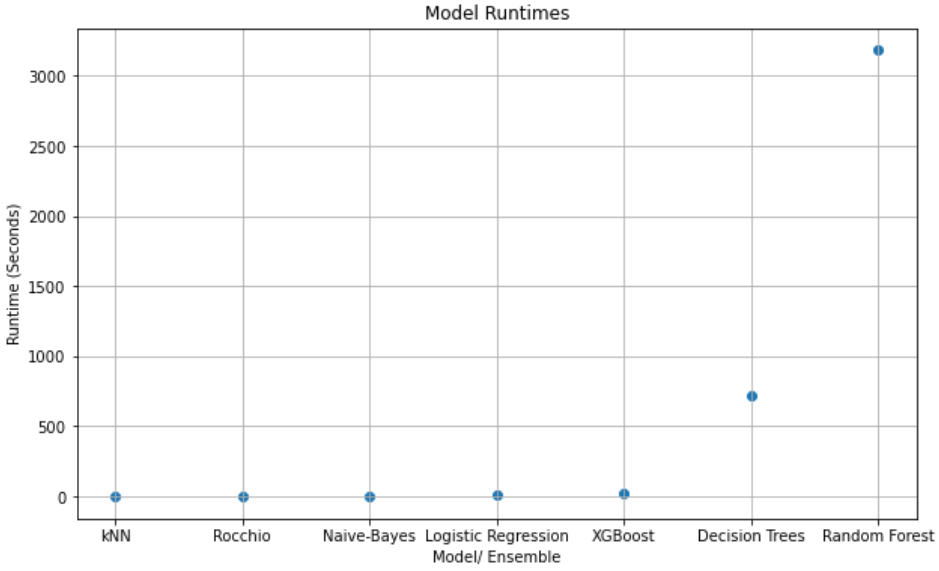
5. Results

The graph below describes the accuracies for each model ran (except SVM). The best performing model in regards to accuracy is logistic regression at 79.2%. The next best performing is Naïve Bayes at 78.3%, and the worst was kNN at 59.1%. However, the nearest centroid method (Rocchio) did perform dramatically better at 73.9% compared to kNN alone and exceeded the two ensembles, decision trees and XGBoost, accuracies. It is also interesting to note that hyperparameter tuning resulted in worse accuracy results in Random Forest and XGBoost, where the default parameters worked best.



The graph below shows the runtimes for each ensemble and model. SVM is excluded due to the fact that it exceeded a day at runtime, and was ruled out at too inefficient for testing purposes at this time. The fastest model was kNN at 0.0352 seconds, which happened to have the worst accuracy measures. The next fastest was Rocchio at 0.0932 seconds, which is impressive as it did perform fairly well for its speed of processing. Naïve Bayes and Logistic Regression were fairly speedy with, as expected, the ensembles taking the longest to run with Random Forest taking 3182 seconds to run.

Rawan Hammad
Saurin Patel
Anisha Patel



Looking at other measures of Logistic Regression continues to prove that it performed the best on average, with a precision of 82.5% for negative sentiment, and 76.5% for positive sentiment. Naïve Bayes was somewhat comparable too, so we tested both in our section of the code. The first 10 tweets sentiments were predicted using both, Logistic Regression and Naïve Bayes, which results in the same results. However, a few predictions were incorrect. However, this could be due to the fact that some tweets could have been neutral, yet neutrality was never accounted for in this analysis.

6. Conclusion and Recommendations

Various models and ensembles were trained to extract the sentiment from 500,000 tweets. The final accuracies, runtimes, and overall performance of each result in Logistic Regression scoring best, followed by Naïve Bayes. A few recommendations or testing that may be needed in the future to perfect those models is to use a slightly larger training set to train the models to improve the models' performance considering that 1.6 million tweets were given. It would also be crucial to include a neutral target in the future to improve the overall ensembles/ models performances. A huge caveat to the project was runtimes and the computing power of our devices which hindered us from testing on a larger dataset and eliminated our ability to try SVM.

7. References

https://www.kaggle.com/datasets/kazanova/sentiment140
https://towardsai.net/p/nlp/natural-language-processing-nlp-with-python-tutorial-for-beginners-1f54e610a1a0