Natural Language Processing

Classifying Blog Posts Using the Title

By Zakaria Zerhouni

Problem Statement

We will use natural language processing to classify Reddit titles form the "happy" and "sad" subreddits. The goal will be to classify the testing data as coming from either the "happy" or "sad" subreddit as accurately as possible.

Description of the Data

The data was pulled using the Pushshift API (https://github.com/pushshift/api) from the following websites:

- https://www.reddit.com/r/happy/
- https://www.reddit.com/r/sad/

The data used to model was a 20,000 row by 2 column dataframe. The critical column of data here are the titles from posts made to each subreddit. A target column was created by assigning the value 1 to all the rows of the "happy" subreddit and the value 0 to all the rows of the "sad" subreddit.

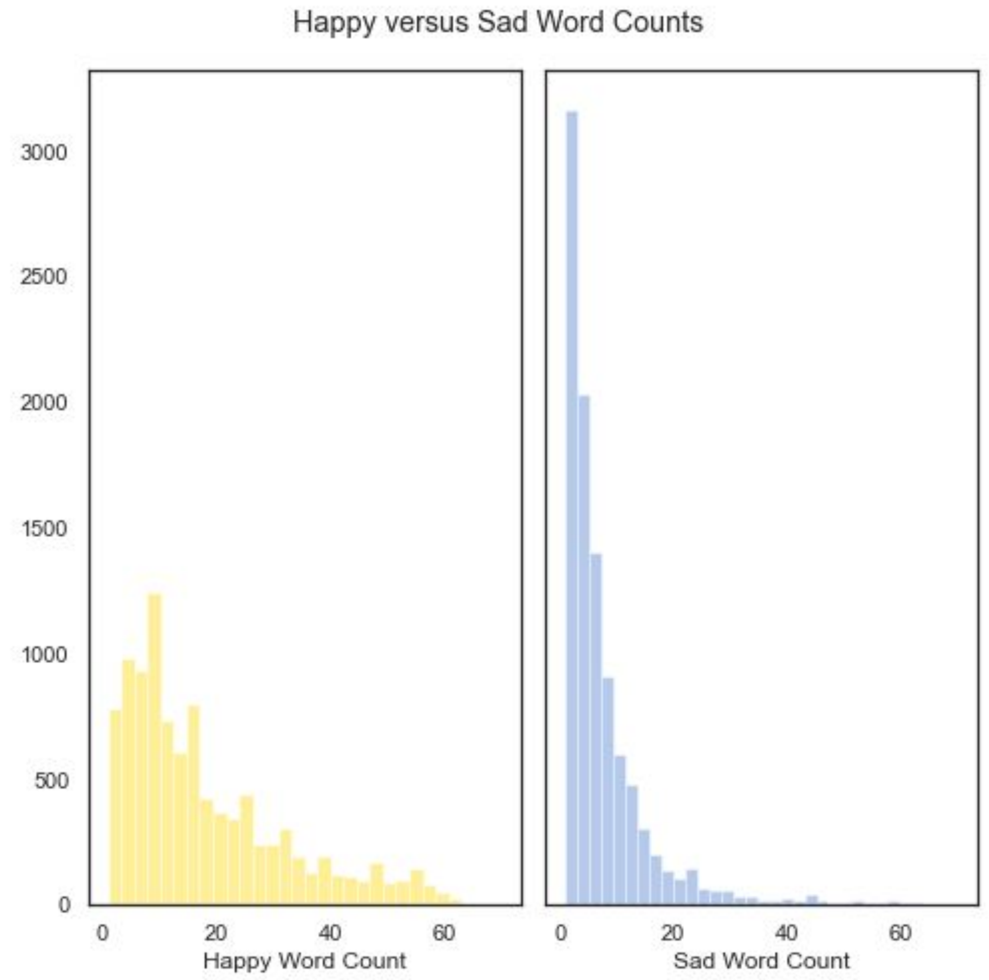
Collecting and Cleaning the Data

- From each subreddit 10,000 titles were stored into a Pandas dataframe by requesting 100 titles at a time, and then requesting the next 100 titles from posts before the "created_utc" timestamp of the last title in the previous API request.
- Cleaning the data was relatively simple as only a simple row had a null value. This row of data was dropped.

Exploratory Data Analysis

The word count of each title was calculated and showed that "happy" posts are on average longer than "sad" posts.

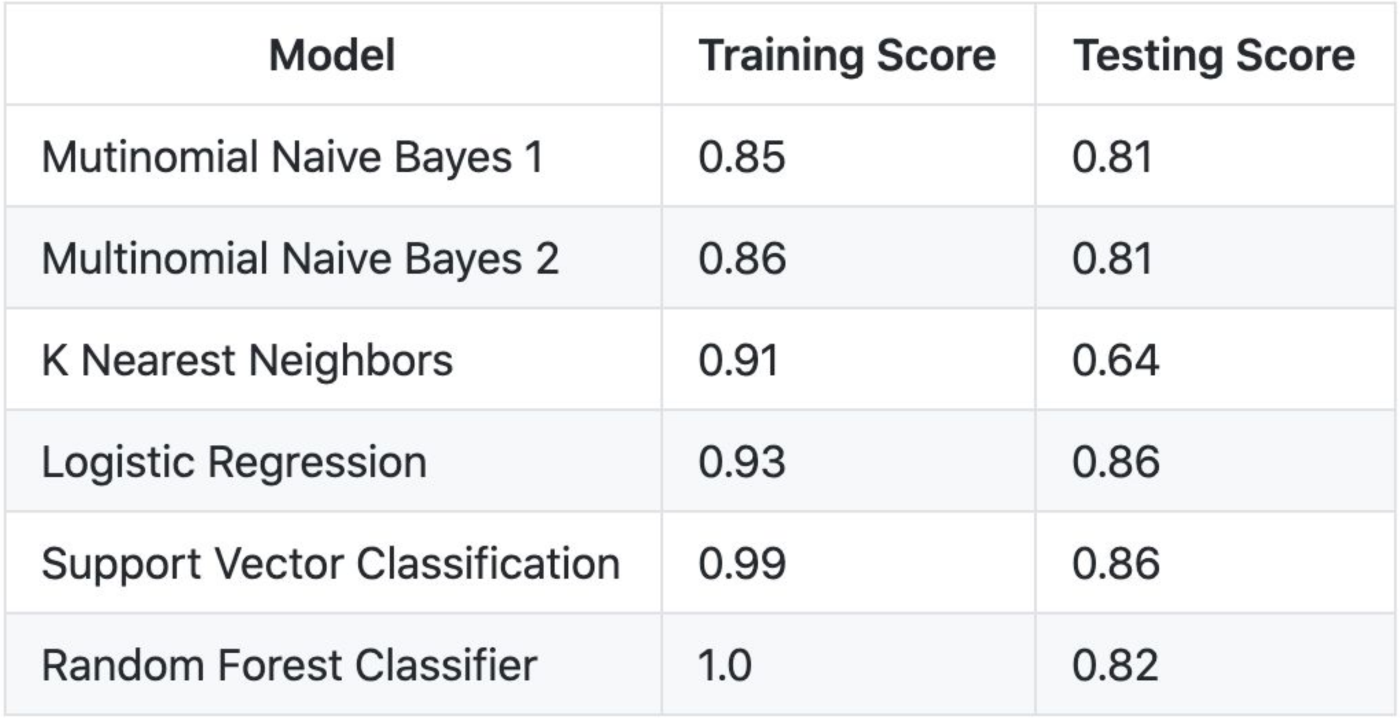
- "Happy" posts were approximately 94 words long
- "Sad" posts were approximately 40 words long


Happy versus Sad Word Counts
3000
2500
2000
1500
1000
500
0
0
20
40
60
Happy Word Count
0
20
40
60
Sad Word Count

Model Performance: Baseline Accuracy

The first thing determined to benchmark our model performance was the baseline accuracy. As half of the data is from "happy" post and hald the data is from "sad" posts the baseline accuracy is almost exactly 0.5. It is slightly in the favor of "happy" posts as a single null entry of data was removed and this was a "sad" entry.

Model Performance: Training vs. Test Scores

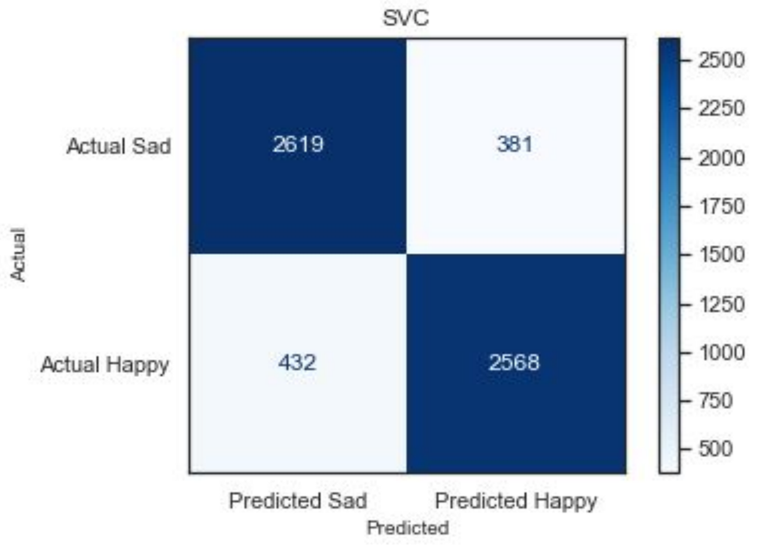


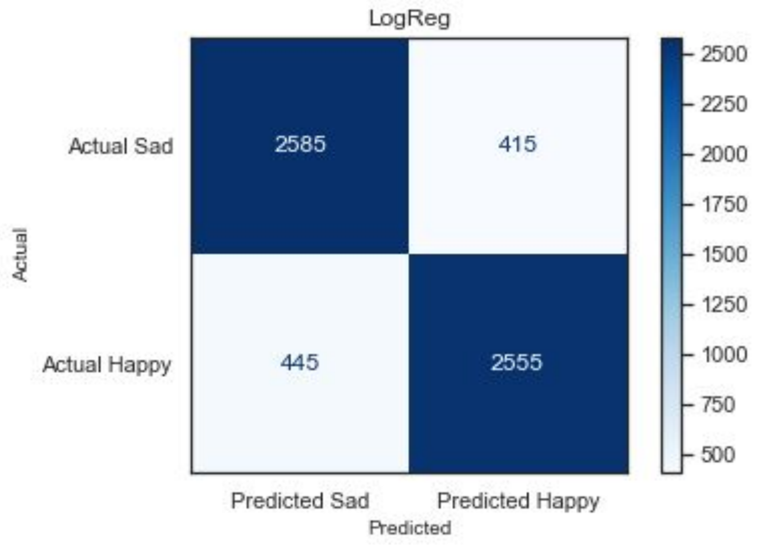
Model	Training Score	Testing Score
Mutinomial Naive Bayes 1	0.85	0.81
Multinomial Naive Bayes 2	0.86	0.81
K Nearest Neighbors	0.91	0.64
Logistic Regression	0.93	0.86
Support Vector Classification	0.99	0.86
Random Forest Classifier	1.0	0.82

Note: Multinomial Naive Bayes 1 uses the Count Vectorizer and Multinomial Naive Bayes 2 uses the TF-IDF Vectorizer.

Primary Findings

In the end it appears that Support Vector Classification and Logistic Regression performed the best with **only** the vectorized data from the post titles. The confusion matrices of the SVC and Logistic Regression models are shown below.





Conclusion: The Final Decision

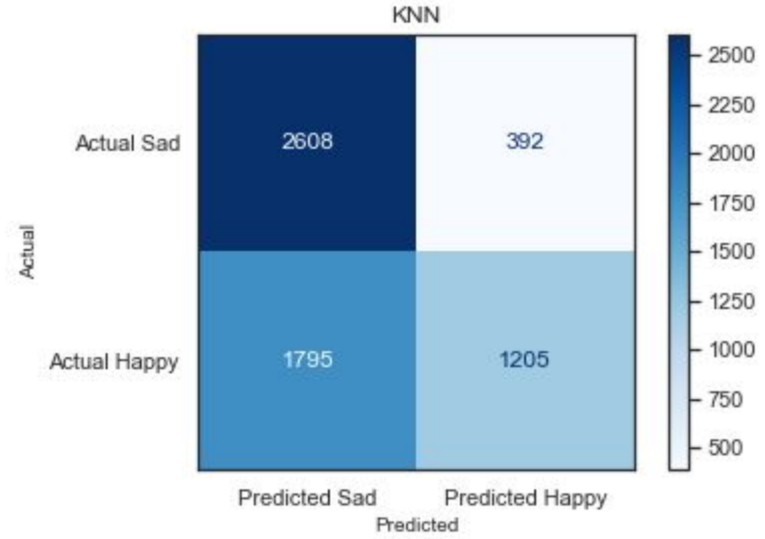
Support Vector Classification

It is of note that to find these settings for the performance and speed of the single Logistic Regression model used a three hour grid search was implemented. The solver used for the model, the "saga" solver, slowed down the modeling substantially for most of the grid search.

For this reason the final choice in model is the Support Vector Classifier. It has relatively easy hyper parameters to tune for good performance. We were able to substantially surpass our baseline accuracy of 0.5 with a score of 0.86 and that is with only the text data.

Sidenote on KNN: What happened?

KNN significantly underperformed compared to the other models tested. If we look at the confusion matrix we see that the model has an issue classifying “sad” posts.



- It was highly overfit with a training accuracy of 0.91 and a testing accuracy of 0.64.
- If this deficiency could be addressed is there the possibility that KNN could be improved to meet the performance of the other models?

What can we do next?

Exploring adding additional features could assist in improving our accuracy. Just from our initial exploratory data analysis it was shown that happy posts often are longer than sad posts. Some ideas for exploring new features are:

- Are sad posts more often flagged as 18 or over?
- Do happy posts more often have an image attached?
- Do happy or sad posts have a difference in how many comments each post has?

New data would also require some exploration into the performance of the other models with additional data. Are support vector models just as good if additional numerical or categorical features are included in the data?

Questions and New Problems

There is also an interesting question to look into following this. Can these models be used to classify subreddits that are similar in theme to "happy" or "sad"?

Can we classify posts from https://www.reddit.com/r/MadeMeSmile/ or https://www.reddit.com/r/awfuleverything/ assuming that "MadeMeSmile" would lean more often towards what is classified as "happy" and "awfuleverything" would lean more often towards what is classified as "sad"? This is an excellent question for the next investigation using Natural Language Processing.

Finally, a bigger and much more difficult question is how far can classification methods like this go? Could a model be built that would accurately assign any title to its proper category? This would be a substantially more involved process, but could assist in recommending posts to those who would ready them based on the title.