Practical Business Applications of Sarcastic Tweet Detection Using Natural Language Processing and Machine Learning

William C. Webb

ABSTRACT

TBD

INTRODUCTION

Problem / Question

Irony and sarcasm are powerful ways to communicate derision or ridicule. The use of irony entails meaning the polar opposite of a literal interpretation and sarcasm employs caustic language to mock or convey contempt, often using irony (Merriam-Webster 1958) . For the sake of simplicity, I hereafter use the term sarcasm to also include irony.

When using sarcasm, speakers communicate their sarcastic intent through nonverbal body language or by their tone of voice. Altering voice patterns and other methods allow for effective ways to communicate sarcastic intent. In this way, sarcastic speakers demonstrate the difference between their sarcastic intent and the literal interpretation of their messages. In contrast, written communication lacks clear, effective cues for communicating sarcastic intent. Unless the writer provides some indication such as the use of quotation marks or other punctuation, readers might miss the writer's sarcastic intent. Therefore, detecting sarcasm by the audience in written communication presents a major challenge.

Users of social media sometimes employ indicators of sarcastic intent in their written communications. For example, some Twitter users sometimes include special hashtags to indicate sarcasm, such as #Sarcastic or #sarcasm. However, Twitter lacks a platform tool for users to systematically indicate sarcasm and not all users employ sarcastic hashtags. This sometimes creates uncertainty about the true intent of some Twitter posts.

Detecting sarcasm from text in the absence of written cues – the "sarcasm problem" - can be a challenge for computers as well as humans. Indeed, natural-language understanding (NLU), or conferring the ability to understand meaning from text, represents one of the significant challenges in Natural language processing (NLP), the field of study concerning interactions between computers and human language (Bates 1995). At the same time, opinions, reviews and recommendations expressed on social media platforms including Twitter form valuable resources for understanding public attitudes, with significant implications for both economics and politics (Raschka and Mirjalili 2017).

Sentiment analysis is the subfield of NLP which employs NLU to quantify public attitudes as expressed in text including social media posts. Sarcastic text lacking written cues (such as a #Sarcasm hashtag on Twitter) in public media presents a significant NLU challenge for sentiment analysis. The inability of sentiment analysis to detect sarcasm could result in large errors since in many cases the writers intend the polar opposite of their text's literal meaning. For example, a business conducting sentiment analysis

for gauging public attitude towards a product might produce inaccurate results based on a potentially-sarcastic Twitter post such as the following: "I love (product X) since it works so well".

In this project, I addressed the sarcasm problem using Twitter. I used NLP and machine learning to develop a model to distinguish between sarcastic and non-sarcastic Tweets. I explored practical business applications of the model for complementing sentiment analysis based on different business objectives. Finally, I developed a sentiment analysis using a novel dataset of Tweets and explored the sarcasm model's ability to improve the sentiment analysis for this new data.

METHODS

Preprocessing

In this project, I addressed the sarcasm problem in sentiment analysis by training a machine-learning model to detect sarcastic tweets. I collected data using Twitter's API which allowed filtering to reduce potential sources of variation, such as filtering out retweets, filtering for English-only tweets and filtering for United States-based profiles. Data collection resulted in 104,940 tweets identified by their authors as sarcastic by hashtags such as #Sarcasm. I then collected 104,937 tweets from the same Twitter usernames making sure the matching tweets lacked sarcastic identifiers.

I pre-processed the tweets prior to constructing a bag-of-words model for the text corpora. In general, preprocessing involved two general goals: distilling the text corpora into its basic components and removing potential sources of bias. Preprocessing steps included removing the original search criteria words used to wrangle sarcastic tweets (e.g. "sarcasm"). Other components removed from the tweet corpora included capitalization, numbers, URLs, usernames, punctuation, special symbols and proper nouns using spaCy's 'en_core_web_sm' model (Honnibal and Montani 2017). I also removed stop-words using NLTK (Bird et al. 2009) and employed lemmatization using NLTK (Bird et al. 2009) for word-stemming with the goal of attaining the canonical (grammatically-correct) forms of the original words (Toman et al. 2006).

In the final step of preprocessing, I used semantic similarity scoring to remove tweets related to politics. To accomplish this, I scored tweets for their relationship to a custom set of political-related terms and removed all tweets with scores > 0.5 (Table 1). After preprocessing the tweet corpora, the dataframe contained 48,662 tweets including 28,220 sarcastic tweets and 20,442 non-sarcastic tweets.

Exploratory Data Analysis

After preprocessing the raw tweet text, the combined preprocessed tweet text consisted of 292,744 total processed words including 14,970 unique words (Fig. 1).

Table 1. Sample political-related scores for tweets

Political Score	Processed Tweet Text
0.49	state continue attack machine anything stop
0.75	republican manifesto look cut budget republican making claim

Figure 1. Word cloud generated from the preprocessed sarcasm data tweet corpora

A frequency distribution of the most-abundant words in the preprocessed text corpora reveals a positive skew. The two most-abundant words - 'get' and 'one' appeared to contribute most to the positive skew. The third-most abundant word was 'like'(Fig 2).

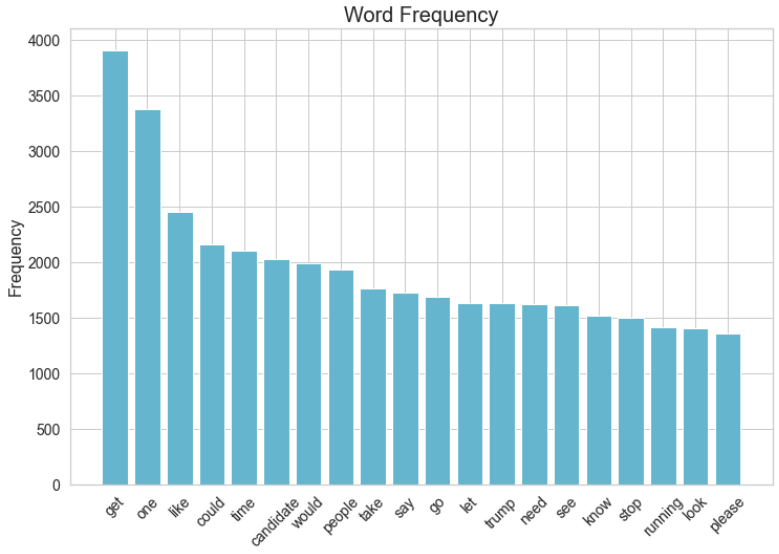


Figure 2. Frequency distribution of the 20 most-abundant words in the preprocessed text corpora.

The processed tweets averaged 6.0159 (SD 3.6876, range 1-28) words per tweet. The most common tweet length after preprocessing was five words (Fig. 3).

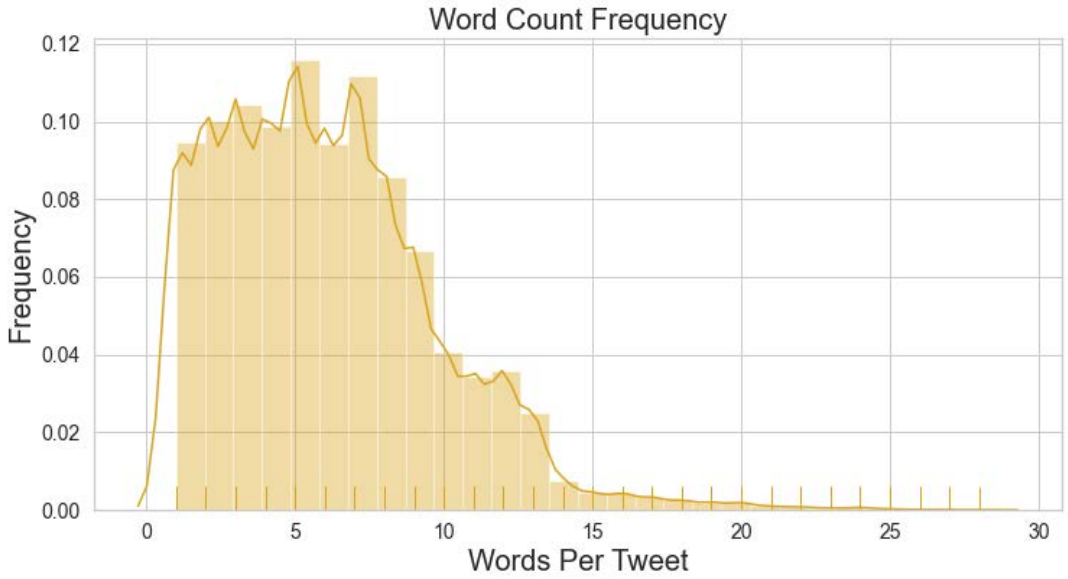


Figure 3. Frequency distribution for the number of words per tweet after preprocessing.

Tweets labeled as sarcastic contained significantly more words ($\bar{x}$ = 6.4734 ± 3.9833 SD) after preprocessing compared to tweets labeled as non-sarcastic ($\bar{x}$ = 5.3842 ± 3.1272; t = 32.5071, P < 0.01; Fig. 4).

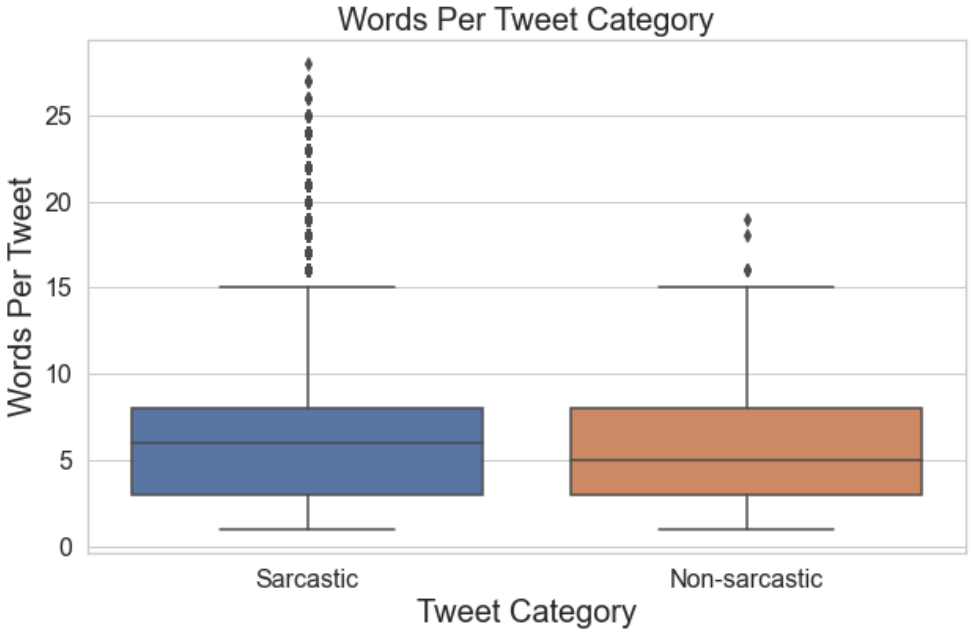


Figure 4. Comparison of words per tweet between pre-processed tweets labeled as either sarcastic or non-sarcastic.

Table 2. Top-twenty most sarcastic words from the preprocessed tweet text corpora identified by the model.

Word	Sarcasm Probability
resist	0.9517
belt	0.945
rose	0.933
relax	0.9301
king	0.9293
attempted	0.9265
ramping	0.9071
sentencing	0.9045
journalism	0.8942
anticipated	0.8861
summary	0.8829
ruined	0.8824
enraged	0.8808
covert	0.8805
woke	0.8805
furious	0.8761
homeless	0.8731
legal	0.8691
inner	0.8541
snapchat	0.8469

The most sarcastic words, based on the model's results, included "resist", "belt" and "rose" (Table 2). The least sarcastic words included "biden", "prepared" and "coronavirus" (Table 3).

Machine Learning

Bag-of -words approach

I employed a bag-of-words approach to convert the categorical-format text corpora into a numerical format suitable for passing along to machine-learning algorithms. I first split the text data into training and testing splits (sklearn.model_selection.train_test_split) (Pedregosa et al. 2011), followed by creation of a document-term matrix based solely on the training text. I then transformed the text from the testing split into a separate document-term matrix based on the terms that occurred in the training split (sklearn.feature_extraction.text.CountVectorizer) (Pedregosa et al. 2011). This approach yields more accurate model performance metrics since the document-term matrix created from the testing split remains independent of the training data, similar to how real-world text used to implement the finished model would also be independent from text in the training split.

Table 3. Top-twenty least sarcastic words from the preprocessed tweet text corpora identified by the model.

Word	Sarcasm Probability
biden	0.0219
prepared	0.0312
coronavirus	0.0346
pandemic	0.0363
timing	0.0477
bernies	0.0595
whirlwind	0.0636
bidens	0.0659
switched	0.071
birthday	0.071
schumer	0.0717
phony	0.0722
reported	0.0779
brazile	0.078
walked	0.079
testing	0.0806
math	0.0825
conducted	0.0833
maga	0.0838
aside	0.0852

RESULTS

Model Selection - Choosing a Classifier

I compared the performance of four classifier models: multinomial naïve Bayes, logistic regression, linear support vector and random forest. Within each candidate model, I searched the respective hyperparameter space for values that maximized model performance (sklearn.model_selection.GridSearchCV) (Pedregosa et al. 2011) and characterized the relative performance of the models by calculating receiver operating characteristic area under the curve (ROC AUC) scores. Logistic regression performed the best with a ROC AUC score of 0.8764 (Table 4, Fig. 5) and overall accuracy of 0.7908 (Table 5).

Table 4. Classifier Performance

Classifier	ROC-AUC Score	Best Hyperparameters
Logistic regression	0.8764	C = 1.0

Linear support vector classification	0.8752	C = 0.1
Random forest classifier	0.8485	max_depth = 30, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 1200}
Multinomial naïve Bayes	0.8460	Alpha = 0.15

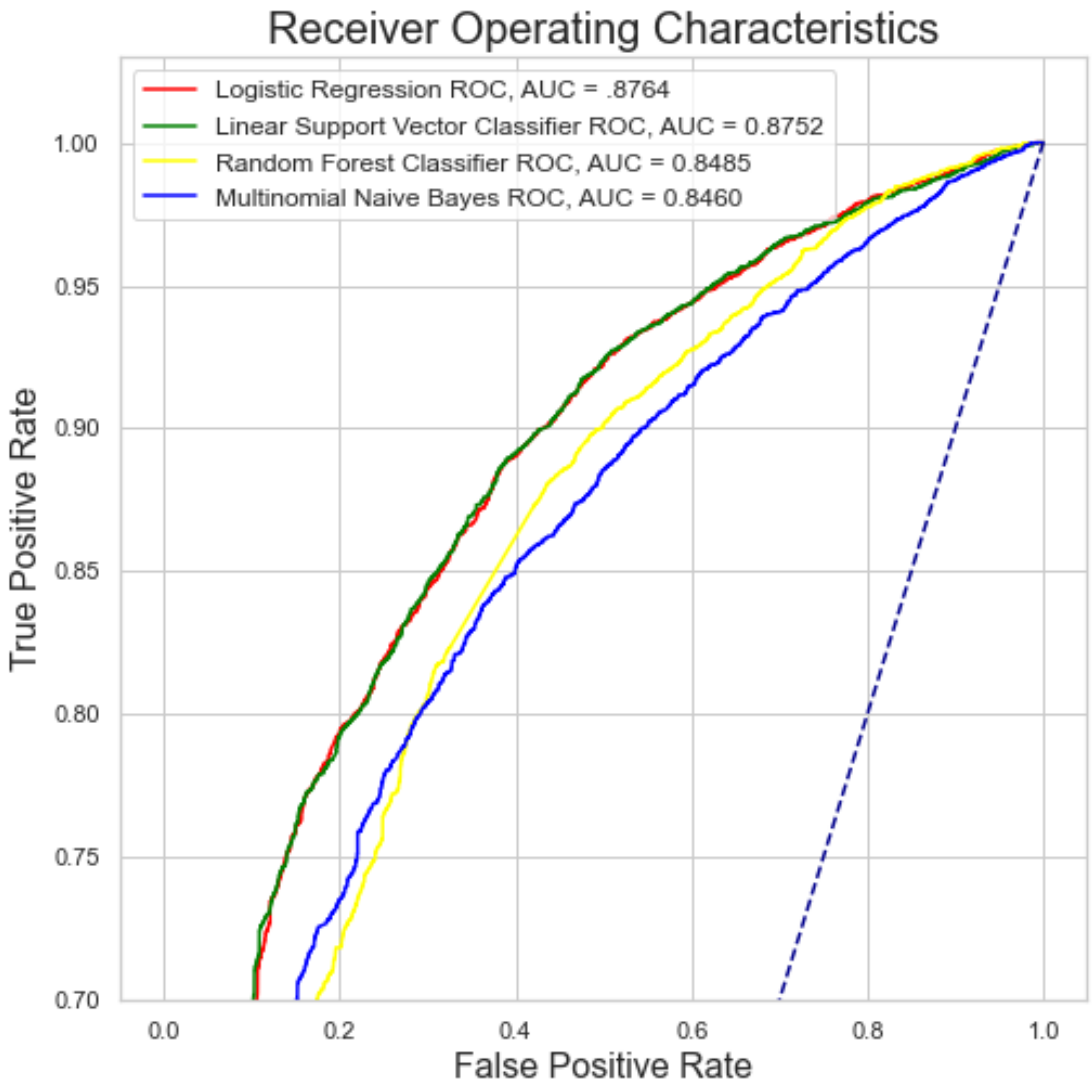


Figure 5. ROC AUC curves for the classification models compared for their relative performance on the sarcasm testing data.

Model Performance

The best-performing model, logistic regression, generated higher precision scores (0.8284) compared to recall (0.8054) and the weighted average f1 score (0.8167). These results indicated the standard model that employed a probability threshold of 0.5, generated fewer false positives compared to false negatives. Thus, the model failed to detect sarcastic tweets (false negatives) more frequently than the model mis-labeled non-sarcastic tweets as sarcastic (false positives; Fig. 6).

Table 5. Classification report for the logistic regression model for discriminating between non-sarcastic and sarcastic tweets.

Metric	Non-sarcastic	Sarcastic	Accuracy	Macro avg	Weighted avg
Precision	0.7425	0.8284	0.7908	0.7854	0.7922
Recall	0.7707	0.8054	0.7908	0.7881	0.7908
f1-score	0.7563	0.8167	0.7908	0.7865	0.7913
Support	4100	5633	0.7908	9733	9733

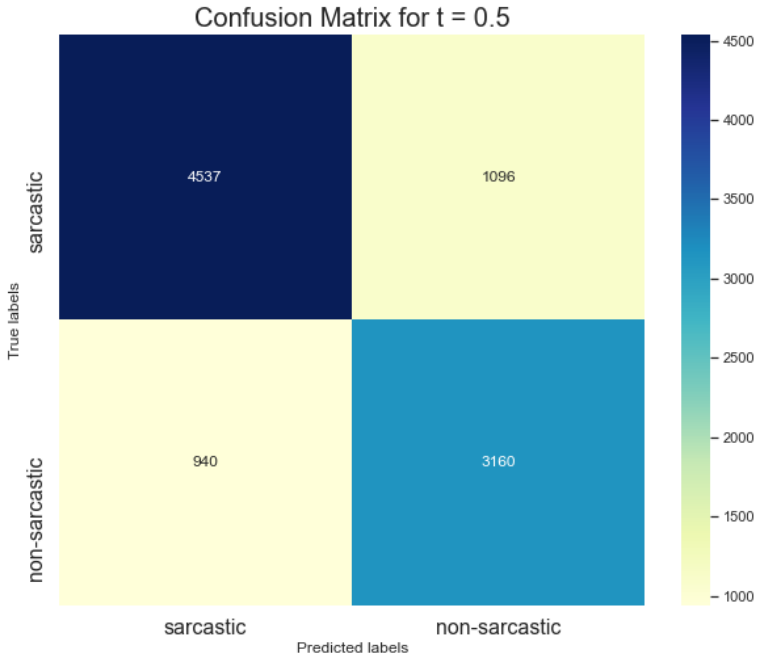


Figure 6. Confusion matrix for the best performing model, logistic regression using the standard threshold (0.5) for categorizing tweets as sarcastic or non-sarcastic.

Model Optimization – Adjusting the Threshold

The default cutoff probability of 0.5 for assignment (threshold) of individual cases lends equal weight to both precision and recall. However, different business scenarios potentially favor either recall or precision. Thresholding involves modeling a range of potential binary classification cutoff values and observing the effects on associated classification metrics. The fbeta metric incorporates a beta parameter which lends emphasis to either precision or recall (sklearn.metrics.fbeta_score) (Pedregosa et al. 2011). When the beta parameter is set to one, fbeta is equivalent to the f1 score and both metrics are emphasized equally. However, when set to values < 1, fbeta favors recall whereas when beta is set to values >1, fbeta favors precision.

Business Applications

Modifying the Logistic Regression Model

The best logistic regression model of the sarcasm data possesses real-world business applications related to the field of sentiment analysis. Sentiment analysis involves gleaning customer attitudes towards products and services, mostly through indirect means such as analyzing social media posts. However, the inability of sentiment analysis to identify sarcasm in customer feedback potentially introduces a significant source of error. Lacking the ability to detect sarcasm, sentiment analysis risks completely mislabeling unsatisfied customers when they create sarcastic content.

Business Scenario One: Minimize the Sarcasm Model's Probability of Overlooking Sarcastic Tweets

In order to avoid overlooking potentially sarcastic content, businesses looking to complement their sentiment analysis using the logistic regression model of the sarcasm data should seek to minimize false negatives and therefore emphasize higher recall compared to precision. In order to simulate such a scenario, I adjusted the beta parameter of the fbeta scoring metric to emphasize recall (beta = 10) and then quantified fbeta for threshold values ranging between 0-1 at 0.01 increments. The threshold value producing the largest fbeta score (0.7999) for this scenario was 0.58 (Fig.7). The new threshold associated with maximizing fbeta resulted in 432 fewer false negatives compared to the standard threshold value of 0.5 (Figs. 6,8). Adjusting the threshold improved recall from 0.8054 at the default threshold (0.5) to 0.8679 at this maximum-recall threshold of 0.58 (Fig. 8).

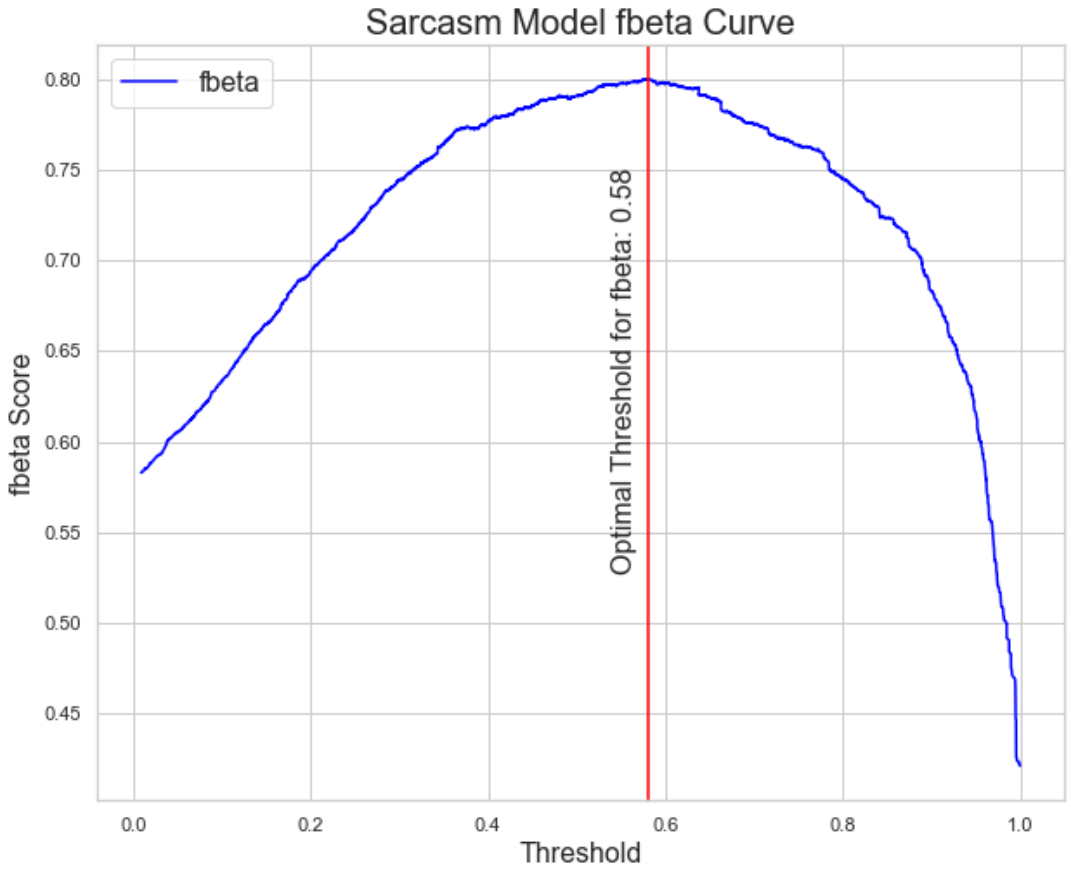


Figure 7. Relationship between the threshold value for categorical assignment of the logistic regression model of the sarcasm data and fbeta. The maximum fbeta score occurred at a threshold value of 0.58.

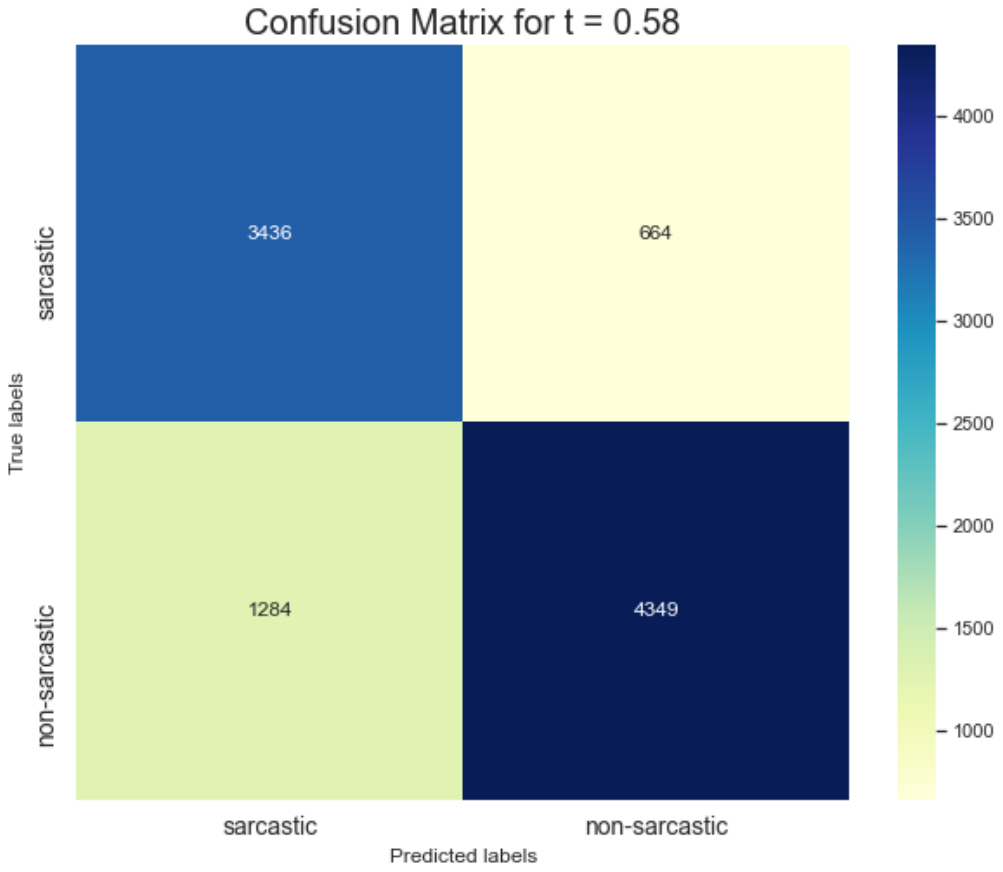
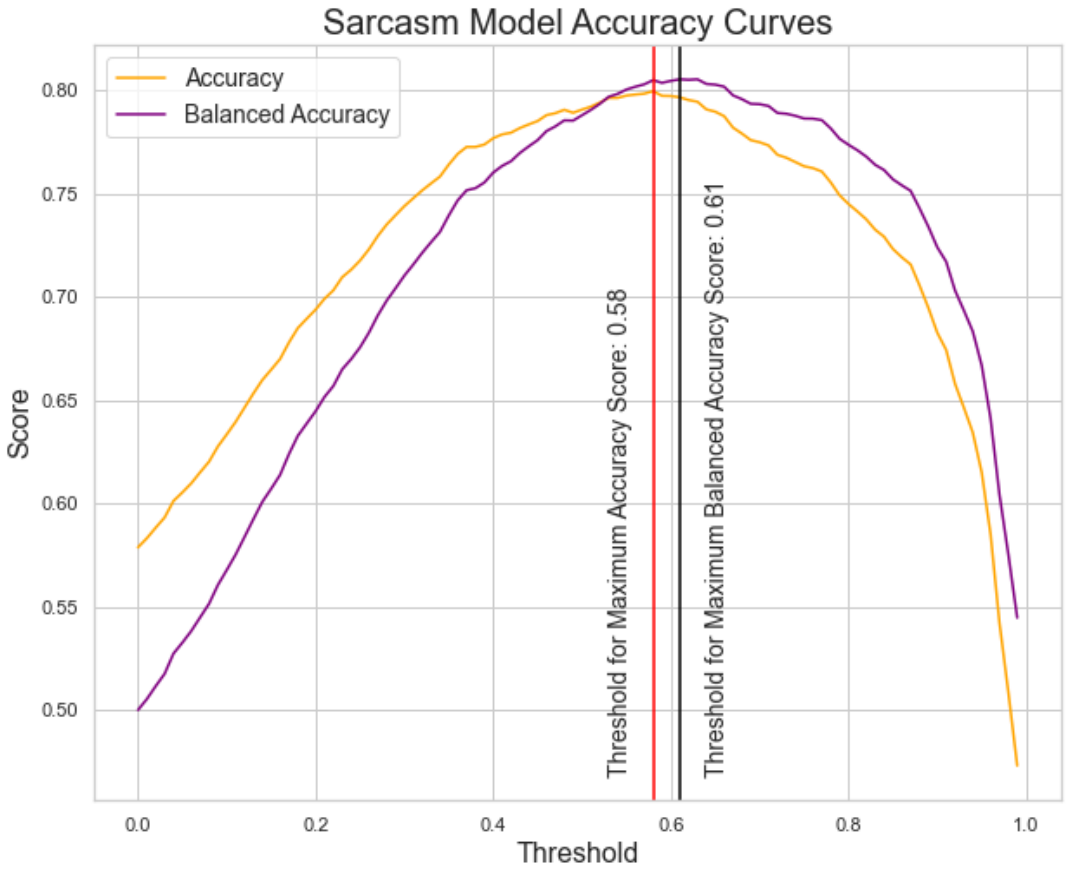


Figure 8. Confusion matrix for the best performing model, logistic regression using the threshold (0.61) for categorizing tweets which maximized the fbeta score. This threshold minimizes false negatives (upper right corner) and maximizes the recall score.

Business Scenario Two: Maximize the Sarcasm Model's Ability to Capture an Accurate Product Sentiment

Rather than focusing on sarcastic, potentially dissatisfied customers, a business might desire capturing an accurate product sentiment. While emphasizing the fbeta metric helps reduce false negatives, accuracy and balanced accuracy scores lend equal weight to both false positives and false negatives while providing a classification performance metric. Using the standard threshold of 0.5, the sarcasm model yielded accuracy and balanced accuracy scores of 0.7908 and 0.7419, respectively.

I investigated whether varying threshold values could produce greater accuracy and/or balanced accuracy scores compared to the standard threshold value of 0.5. I calculated accuracy scores and balanced accuracy scores for threshold values ranging between 0-1 at 0.01 increments to compare with the accuracy score and balanced accuracy scores for the default threshold. Varying the threshold values yielded a slightly larger maximum accuracy score. The maximum accuracy score occurred at a threshold value of 0.58, with an accuracy score of 0.7998. Varying the threshold values produced a relatively larger



gain in the maximum balanced accuracy score. The maximum balanced accuracy score of 0.8055 occurred at a threshold of 0.61 (Fig. 9).

Figure 9. Accuracy and balanced accuracy scores for the sarcasm model across a range of threshold values.

Business Scenario Three: Balance the Risk of the Sarcasm Model's Mis-labeling of Non-sarcastic as Well as Sarcastic Tweets

In some situations, a business may be equally concerned with identifying both satisfied and unsatisfied customers. Therefore, some businesses may seek to strike a balance between the model's mislabeling of tweet categories. The f1 score represents an appropriate metric for striking such a balance since the f1 score incorporates both recall (which minimizes false negatives) and precision (which minimizes false positives). To find this balance, I quantified the f1 score for a range of threshold values between 0-1 at 0.01 increments. The largest f1 score of 0.8192 occurred at a threshold value of 0.46 (Fig. 10). Implementing the maximum f1 threshold of 0.46 reduced the number of false negatives by 136 (960 compared to 1096 at the default threshold), but slightly elevated the number of false positives (1102 compared to 940 at the default threshold) (Fig. 11.)

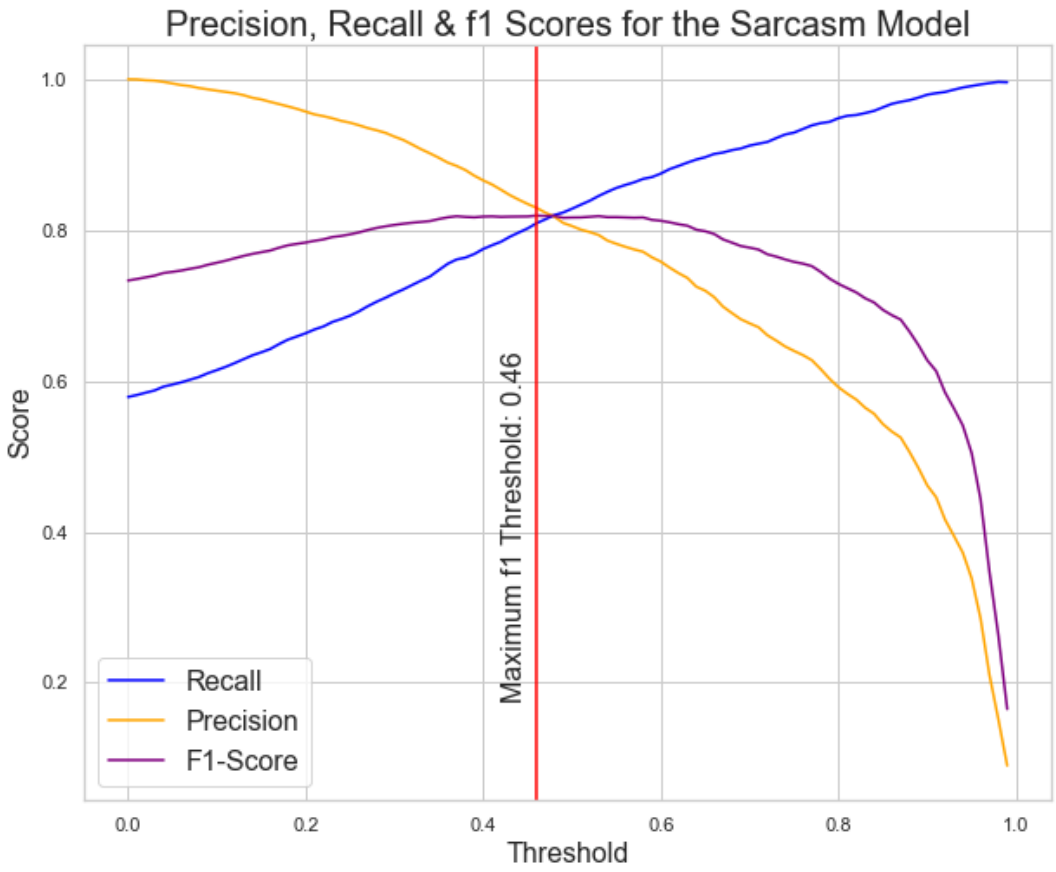


Fig. 10. Variation in precision, recall and f1 scores for a range of threshold values for the sarcasm model.

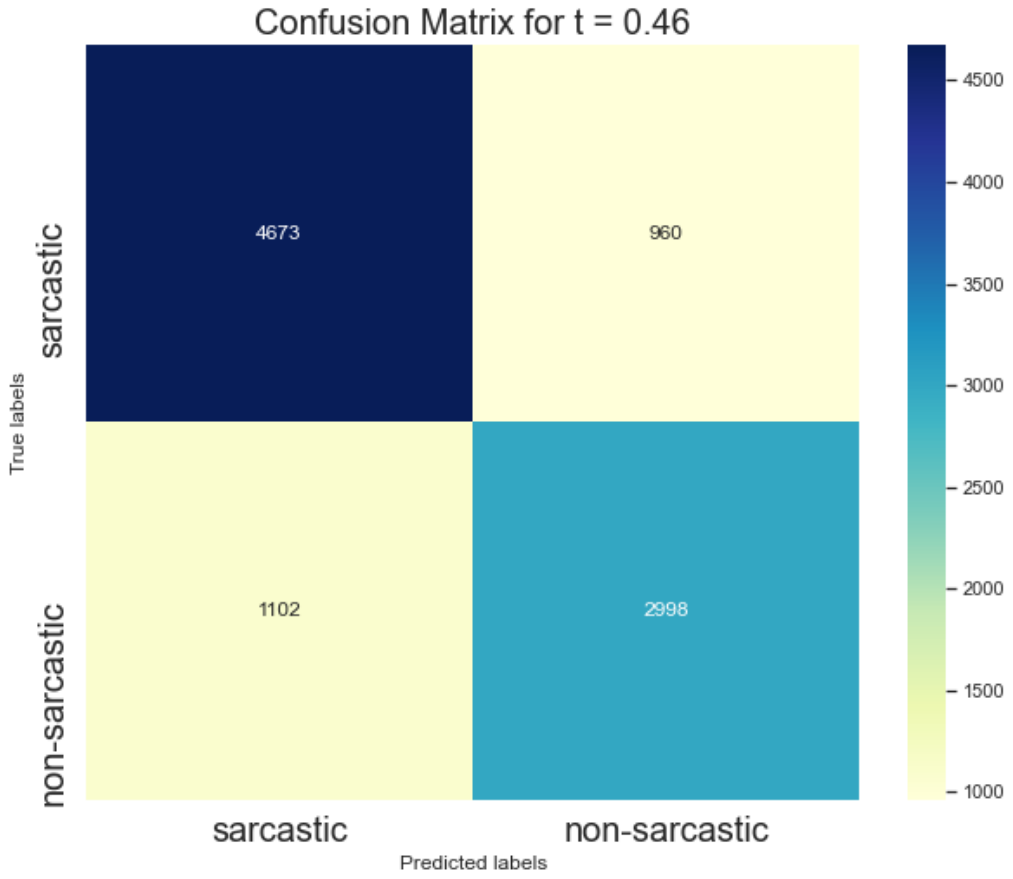


Figure 11. Confusion matrix for the sarcasm model at the threshold which maximizes the f1 score.

Business Scenario Four: Improving Sentiment Analysis Using Sarcasm Scores

I tested the best-performing sarcasm model's utility to improve a sentiment analysis of novel data. As a source of novel data suited for a sentiment analysis, I obtained a copy of the sentiment140 dataset from Kaggle. The sentiment140 dataset contains 1,600,000 annotated tweets extracted using the twitter API. This Kaggle dataset contains data collected by students at Stanford University as part of a class project (Go et al. 2009). Each tweet possesses either a positive or negative label.

To prepare the novel data for the sentiment analysis, I subsampled the novel data to match original size of the sarcasm data (209,877 rows), resulting in 105,079 rows labeled positive and 104,798 rows labeled negative for sentiment. This reduced preprocessing time and avoided any potential biases introduced by unequal sample sizes between datasets.

For the sentiment analysis, I used multinomial naïve Bayes (sklearn.naive_bayes.MultinomialNB) (Pedregosa et al. 2011) and tuned the model (sklearn.model_selection.GridSearchCV) (Pedregosa et al. 2011)using the novel data. The sentiment analysis of the novel data achieved a f1 score of 0.7518 for

correctly predicting the labels of individual tweets. This score represented a baseline to compare with scores generated after applying the sarcasm model to the novel data.

I explored the ability of the sarcasm model to enhance the baseline sentiment model by first applying the sarcasm model to the novel data. Applying the sarcasm model to the novel data resulted in a sarcasm probability score for each tweet in the novel data. For a range of threshold values between 0-1, I flipped the original labels to their polar opposite and recalculated the f1 score for the sentiment. model. In this manner, I explored the effects of the sarcasm mode for changing positive labels alone, negative labels alone and both together. However, none of the three approaches to flipping labels improved the f1 score at any threshold for the sentiment model (Fig. 12).

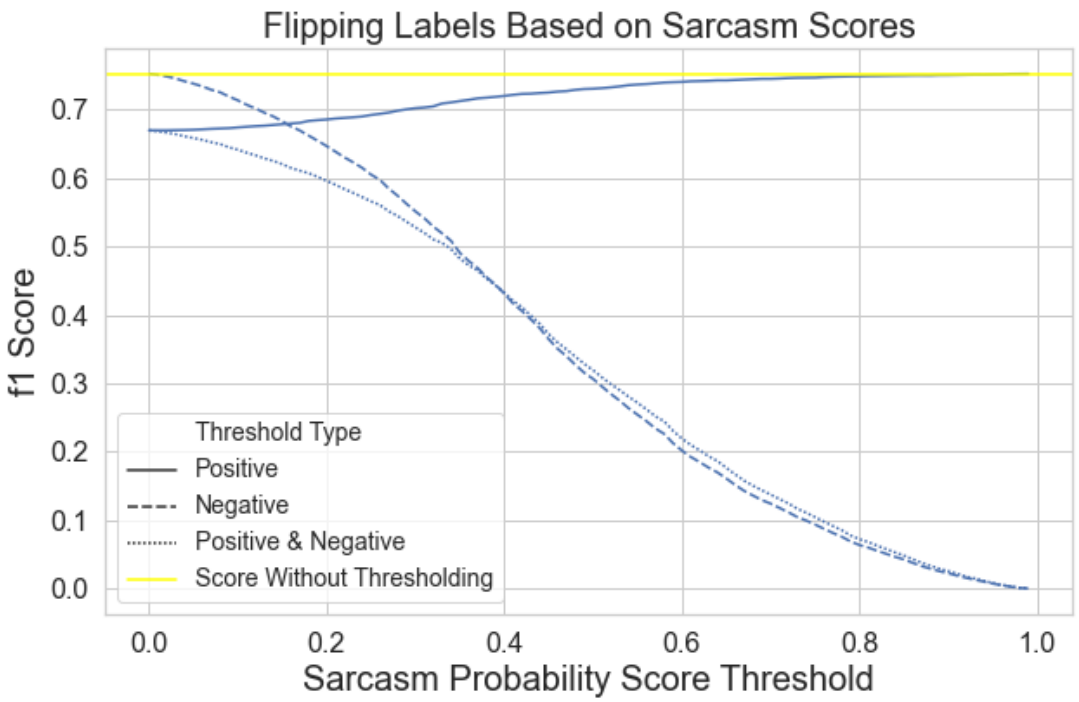


Figure 12. f1 scores associated with sarcasm probability thresholds for changing labels to their polar opposites in the sentiment140 dataset prior to classification using multinomial naïve Bayes.

Web App

I created a web app for users to test the sarcasm model for themselves for individual Twitter usernames. The landing page prompts visitors to enter a novel Twitter username (Fig. 13).

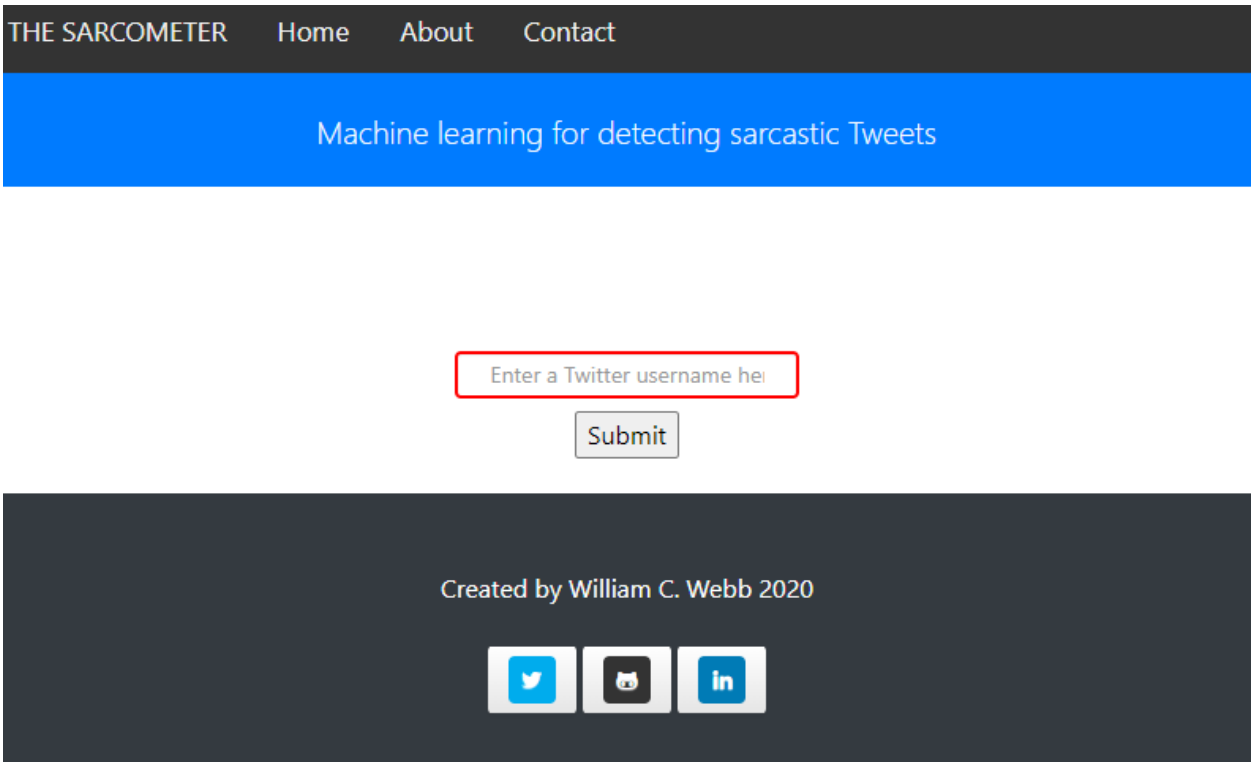


Figure 13. Landing page for the web app.

After entering a Twitter username and clicking on the 'submit' button, the web app returns a results page. The results page contains two primary sections. The top portion displays a horizontal bar chart comparing the average sarcasm scores of the novel username with three celebrities' average sarcasm score (Fig. 14).

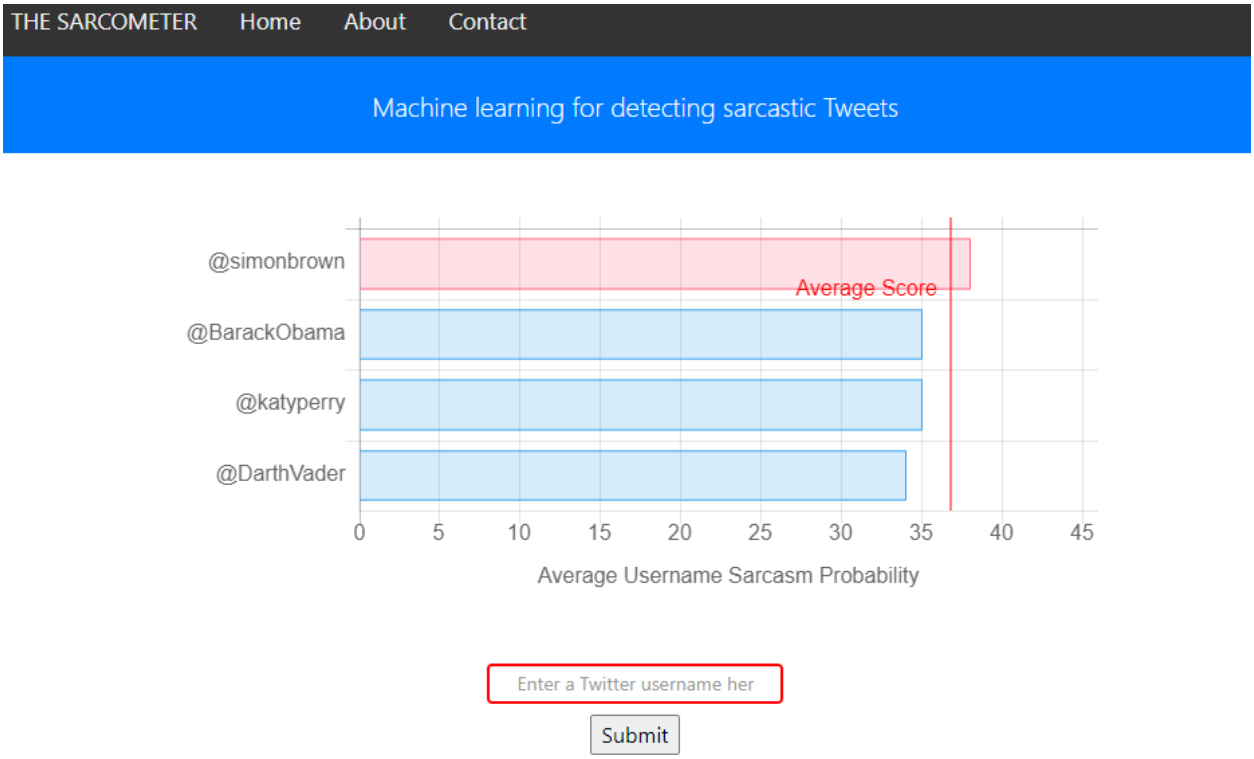


Figure 14. Top portion of the results page returned after a user enters a novel username and clicks the 'submit' button.

The bottom portion of the results page contains a five-panel carousel which users navigate by operating the carousel's forward and backward arrows. The carousel contains three sections. The top section consists of a horizontal bar chart that shows the sarcasm score of the most-sarcastic tweet from the novel username. Below the top section of the carousel, the middle section displays the Twitter widget for the most-sarcastic tweet from the novel username (Fig. 15).

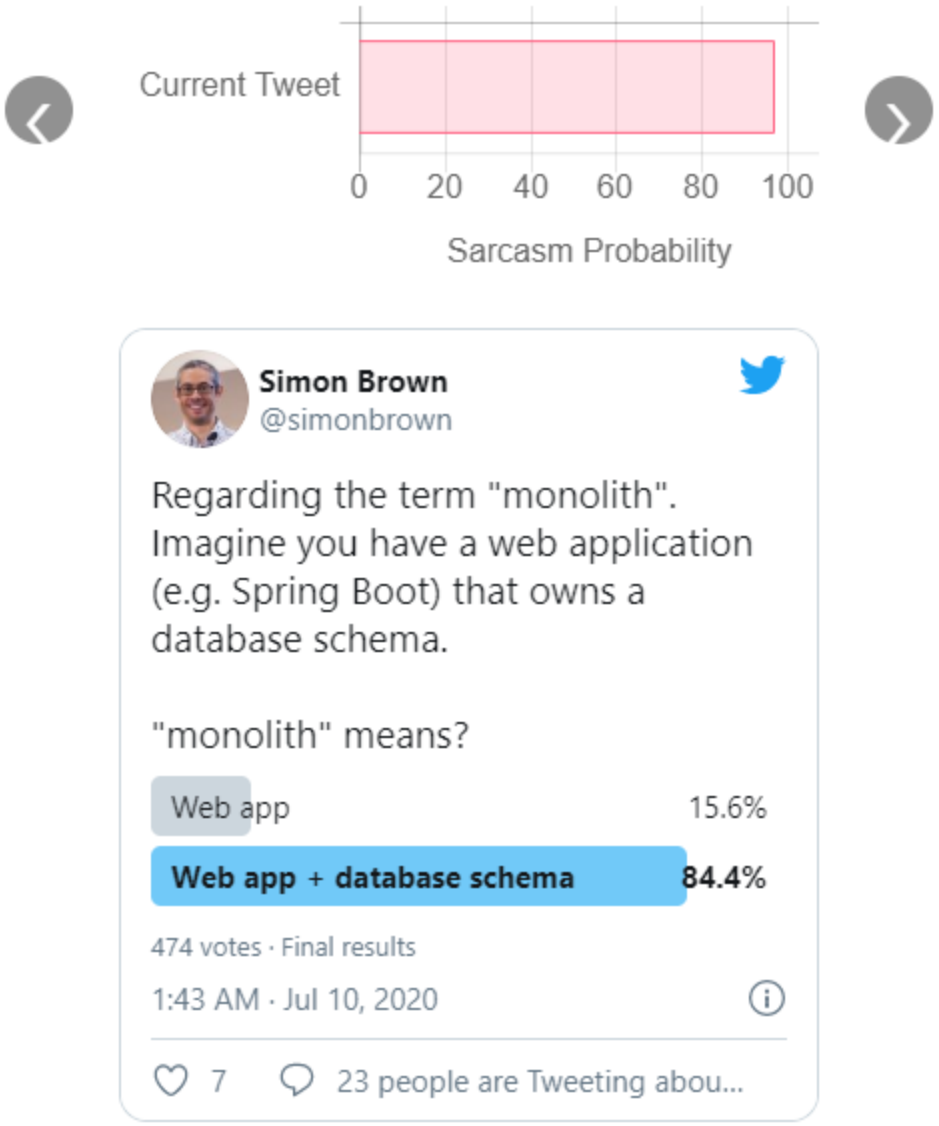


Figure 15. The top two sections of the carousel from the results page. The top section contains a horizontal bar chart with the score for the most-sarcastic tweet created by the novel username. Below the top section, the middle section displays the Twitter widget for the most-sarcastic tweet generated by the novel username.

The bottom third of the carousel displays a horizontal bar chart. Each bar corresponds to one word in the current tweet and represents the respective sarcasm score for each word in the tweet (Fig. 16). Users navigate the carousel in either forward and backward directions through the top five most-sarcastic tweets from the current username while the carousel displays the overall score for the tweet, the tweet's widget and the sarcasm scores for each word in the tweet.

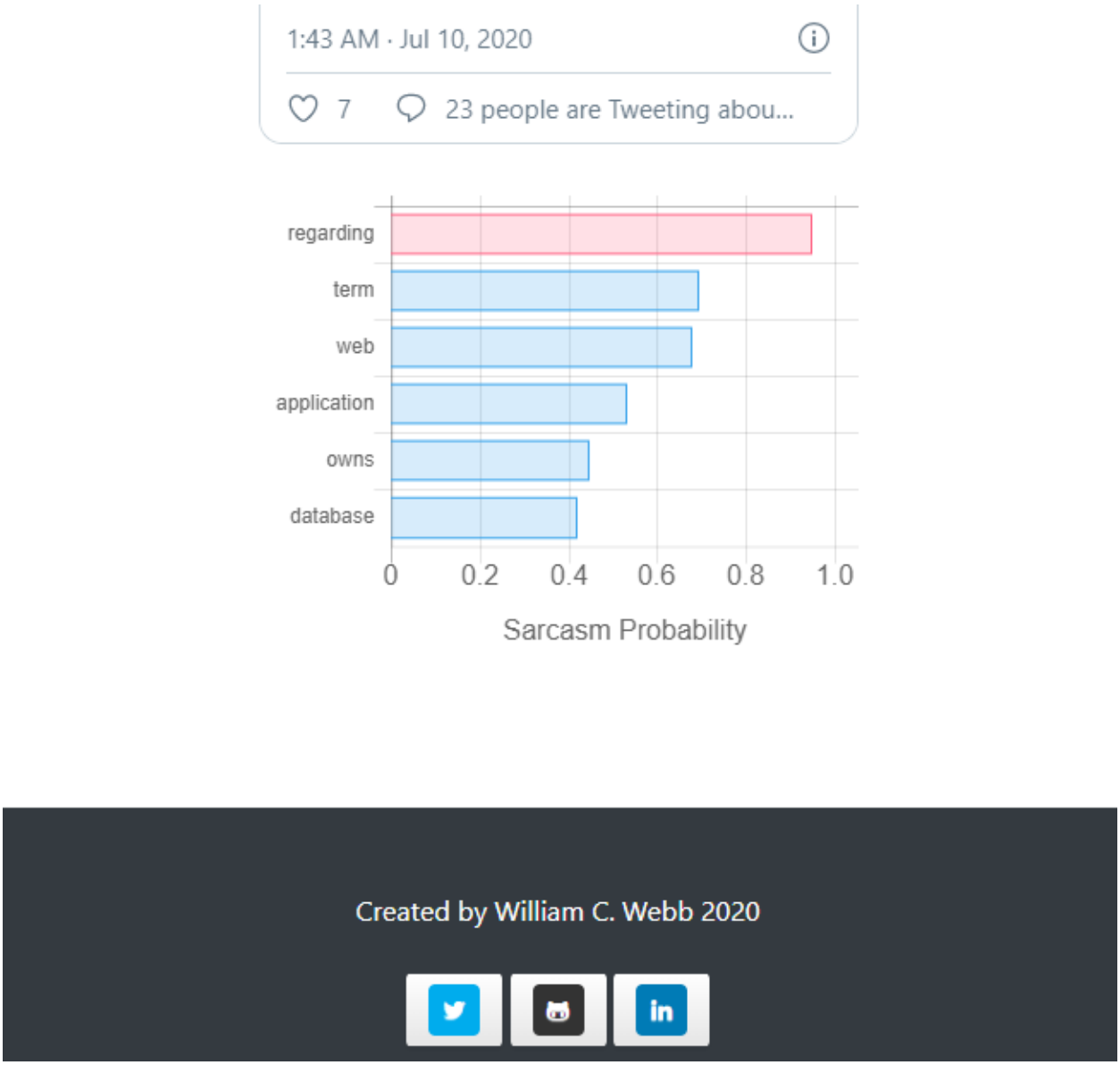


Figure 16. Bottom third of the results page carousel. The bar chart displays the sarcasm score for each word in the current tweet visible in the Twitter widget above (Fig. 10).

CONCLUSION

TBD

REFERENCES

Bates, M. 1995. Models of natural language understanding. Proceeding of the National Academy of Sciences **92**:9977-9982.

Bird, S., E. Klein, and E. Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.

Go, A., R. Bhayani, and L. Huang. 2009. Twitter sentiment classification using distant supervision. . Stanford.

Honnibal, M., and I. Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Merriam-Webster. 1958. Webster's new collegiate dictionary. G. & C. Merriam Co., Cambridge, Mass., U.S.A.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**:2825-2830.

Raschka, S., and V. Mirjalili. 2017. Python machine learning - seond edition: machine learning and deep learning with python, scikit-learn, and tensorflow. Packt Publishing, Birmingham, UK.

Toman, M., R. Tesar, and K. Jezek. 2006. Influence of word normalization on text classification. Pages 354-358 *in* Multidisciplinary information sciences and technologies; Current research in information sciences and technologies; INSCIT 2006.