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

William C. Webb

ABSTRACT

Sarcasm is an effective communication tool, but mostly in the presence of cues that signal its use. Written communication lacks the cues used to express sarcasm, resulting in potential misinterpretation. Although it is an important source of data, social media often contains sarcastic posts which represent a potentially large source of error in sentiment analyses if not properly identified. I used natural language processing and machine learning to construct and tune a model for predicting sarcastic tweets. The tuned model, constructed using random forest, achieved a weighted average f1 score of 0.8316 using the standard classifier threshold (0.5). Additionally, I demonstrated the sarcasm model's utility for addressing three practical business applications using various classification metrics. The business applications included: 1) identifying as many sarcastic tweets as possible by reducing false negatives; 2) capturing an accurate sentiment by maximizing accuracy and 3) balancing the risk of mislabeling both sarcastic and non-sarcastic tweets by emphasizing the f1 score. Finally, I explored the sarcasm model's ability to improve a naïve Bayes sentiment analysis conducted using the sentiment140 dataset, a well-known dataset containing tweets labeled as either positive or negative. After applying the sarcasm model to the sentiment140 dataset, I flipped either the positive, negative or both labels based on a range of threshold values for sarcasm probability and compared the respective f1 scores with the f1 score associated with the standard threshold. Changing labels based on sarcasm probability scores failed to improve the sentiment analysis for flipping negative labels and for flipping both negative and positive labels for the full range of threshold values. However, the sarcasm model contributed a very small improvement in the sentiment analysis when applied to positive labels. Poor performance by the sarcasm model on the sentiment140 data was probably due at least in part, from large differences between the datasets. I hypothesized that the ability of the sarcasm model to improve a sentiment analysis probably depends in large part on the degree of semantic and methodological similarities between datasets.

INTRODUCTION

Problem / Question

Irony and sarcasm are powerful ways to communicate derision or ridicule. Irony conveys a meaning which is the polar opposite of the literal interpretation of a message and sarcasm is the use of 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 indicate irony.

When using sarcasm, speakers normally communicate their sarcastic intent through nonverbal body language or through 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 sometimes presents a major challenge.

Users of social media occasionally 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 tool for users to systematically indicate sarcasm and not all users employ sarcastic hashtags. This sometimes creates uncertainty about the true intent of some tweets.

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 business 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. A business conducting sentiment analysis for gauging public attitude towards a product might produce inaccurate results based on a potentially-sarcastic tweet 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

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 ensuring 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.

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

Exploratory Data Analysis

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

Figure 1. Word cloud generated from the preprocessed tweets

A frequency distribution of the most-abundant words in the preprocessed text corpora revealed 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).

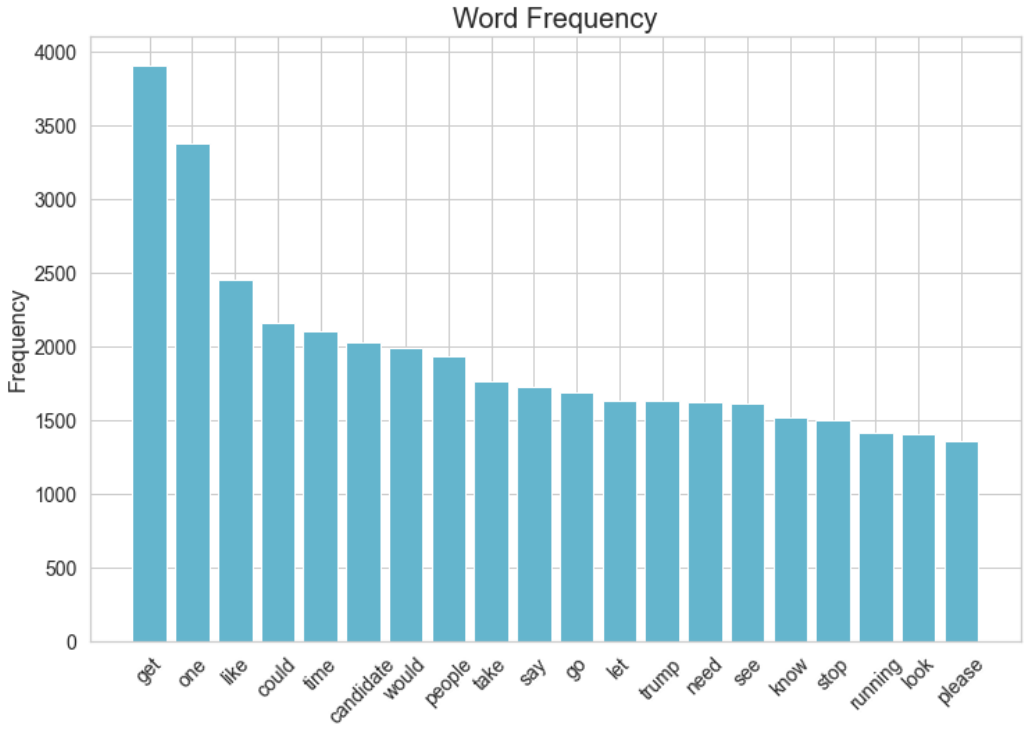


Figure 2. Frequency distribution of the 20 most-abundant words in the preprocessed tweets.

Preprocessed tweets averaged 6.0159 (SD 3.6876; range: 1-28) words per tweet. The most common tweet length after preprocessing consisted of five words (Fig. 3).

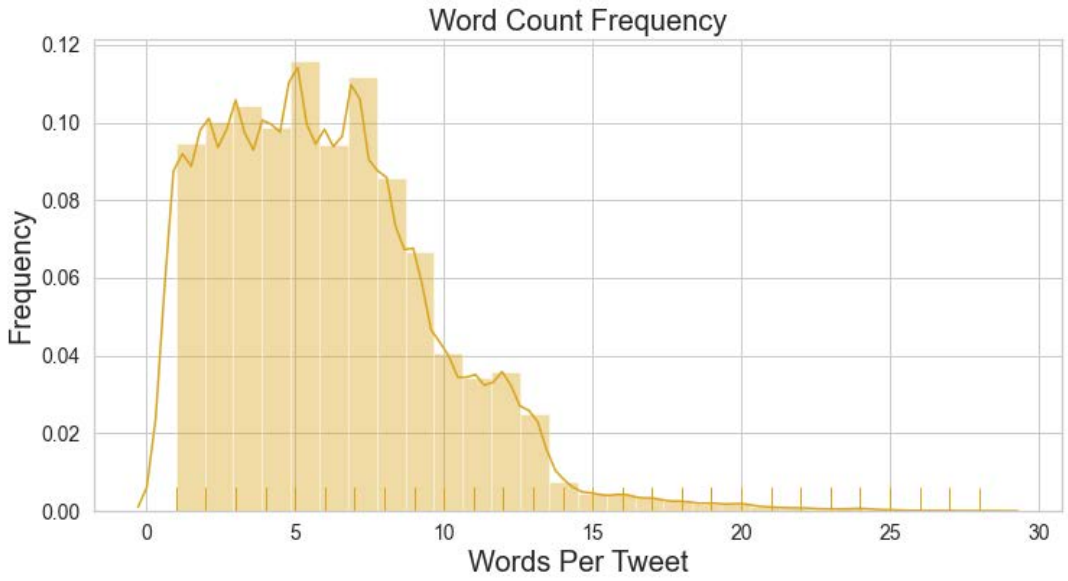


Figure 3. Frequency distribution for the number of words in preprocessed tweets.

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).

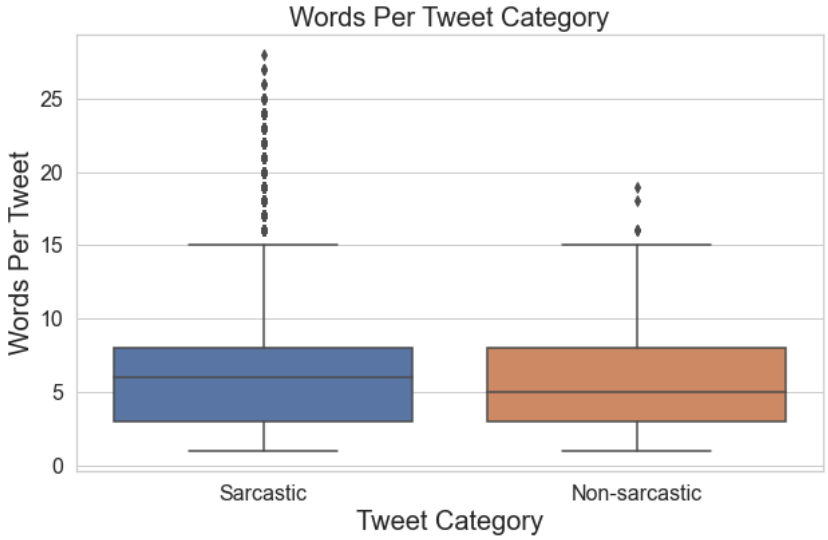


Figure 4. Comparison of words per tweet between pre-processed tweets labeled as either sarcastic or non-sarcastic.

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).

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

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

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.

RESULTS

Model Selection - Choosing a Classifier

I compared the performance of four classifier algorithms: multinomial naïve Bayes, logistic regression, linear support vector and random forest. Within each candidate algorithm, I searched the respective hyperparameter space for values that maximized algorithm performance (sklearn.model_selection.GridSearchCV) (Pedregosa et al. 2011) and characterized the relative performance of the algorithms by calculating receiver operating characteristic area under the curve (ROC AUC) scores. Random forest performed the best with a ROC AUC score of 0.9101 (Table 4, Fig. 5) and overall accuracy of 0.820 (Table 5).

Table 4. Classifier Performance

Classifier	ROC-AUC Score	Best Hyperparameter Values
Random forest classifier	0.9101	max_depth = 550, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 4500}
Logistic regression	0.8764	C = 1.0
Linear support vector classification	0.8752	C = 0.1
Multinomial naïve Bayes	0.8460	Alpha = 0.15

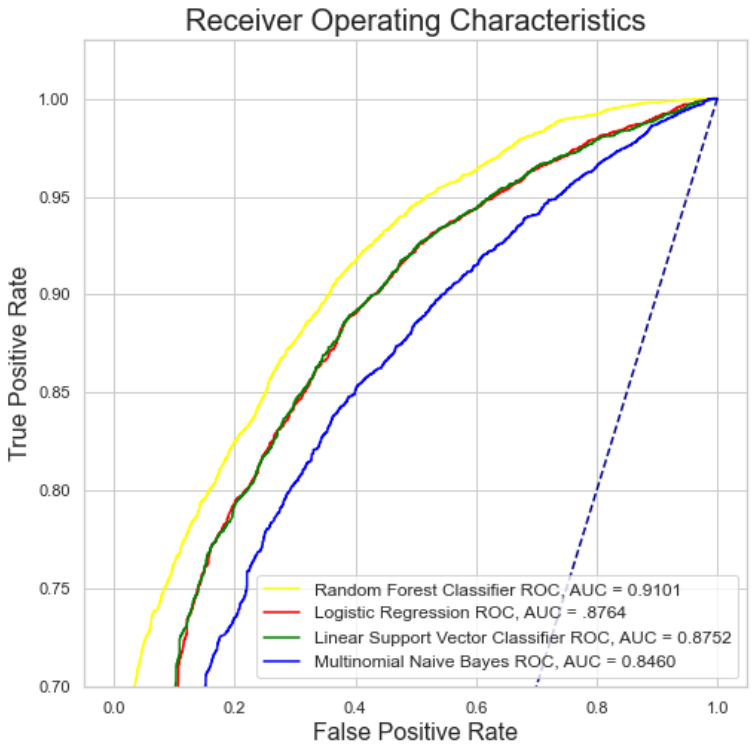


Figure 5. ROC AUC curves for the classification models compared for their relative performance on the sarcasm testing data.

Model Performance

The random forest model generated higher precision scores (0.906) compared to recall (0.768) and the weighted average f1 score (0.832). These results indicated the standard probability threshold of 0.5 generated fewer false positives compared to false negatives. Thus, the model mislabeled sarcastic tweets (false negatives) more frequently than the model mis-labeled non-sarcastic tweets (false positives; Fig. 6).

Table 5. Classification report for the random forest model for discriminating between non-sarcastic and sarcastic tweets.

Metric	Non-sarcastic	Sarcastic	Accuracy	Macro avg	Weighted avg
Precision	.7366	.9064	.8199	.8215	.8349
Recall	.8910	.7682	.8199	.8296	.8199
f1-score	.8065	.8316	.8199	.8190	.8210
Support	4100	5633	.8199	9733	9733

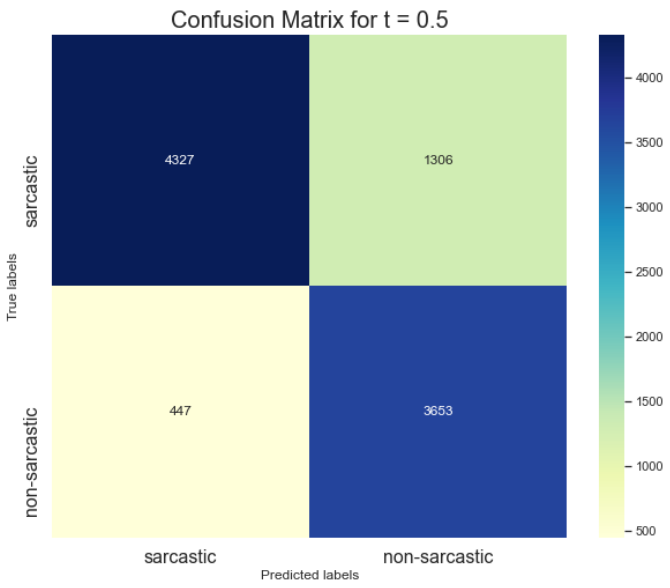


Figure 6. Confusion matrix for the random forest model using the standard threshold (0.5) for predicting tweets as either sarcastic or non-sarcastic.

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 recall and precision are emphasized equally. However, when set to values < 1, fbeta favors recall whereas when beta is set to values >1, fbeta favors precision.

Practical Business Applications

Modifying the Model

I explored the sarcasm model's relevance to real-world business applications associated with the field of sentiment analysis. Sentiment analyses glean 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 Model's Probability of Mislabeling Sarcastic Tweets

In order to avoid overlooking potentially sarcastic content, businesses looking to complement their sentiment analyses 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 (0.01 increments). The threshold value producing the largest fbeta score (0.8213) for this scenario was 0.4915 (Fig.7). The new threshold associated with maximizing fbeta resulted in 834 fewer false negatives compared to the standard threshold value of 0.5 (Figs. 6, 8). Adjusting the threshold

improved recall from 0.7682 (Table 5) at the default threshold (0.5) to 0.9024 at this maximum-recall threshold of 0.49 (Fig. 8).

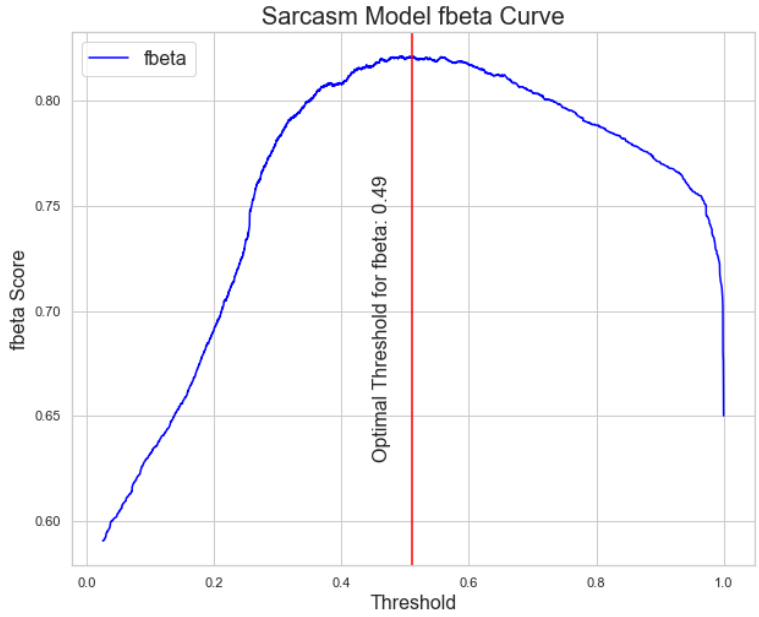


Figure 7. Relationship between the threshold value for categorical assignment of the random forest model of the sarcasm data and fbeta. The maximum fbeta score occurred at a threshold value of 0.4915.

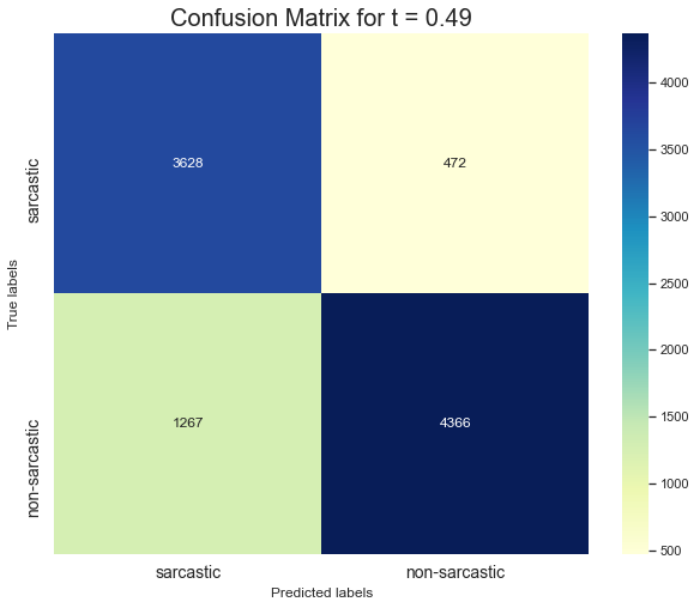


Figure 8. Confusion matrix using the threshold for categorizing tweets which maximized the fbeta score (0.4915). This threshold minimizes false negatives (upper right corner) and maximizes the recall score.

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

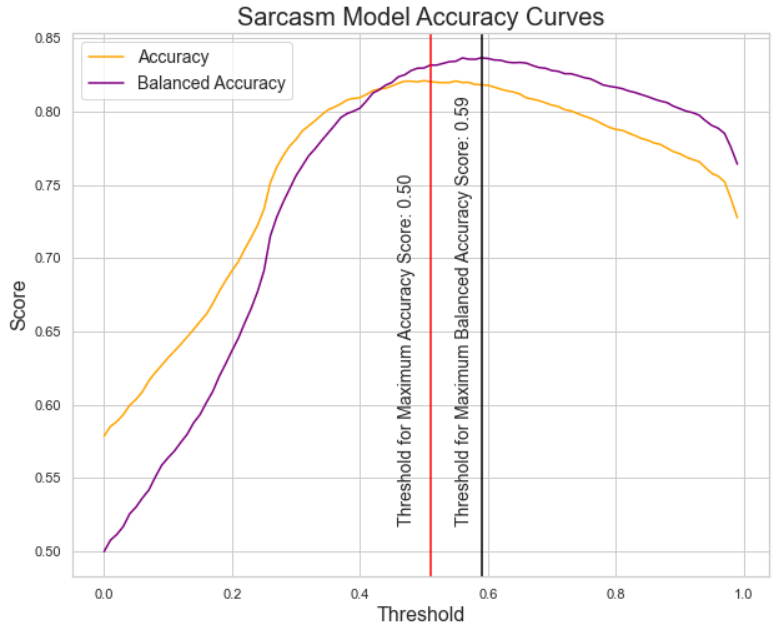
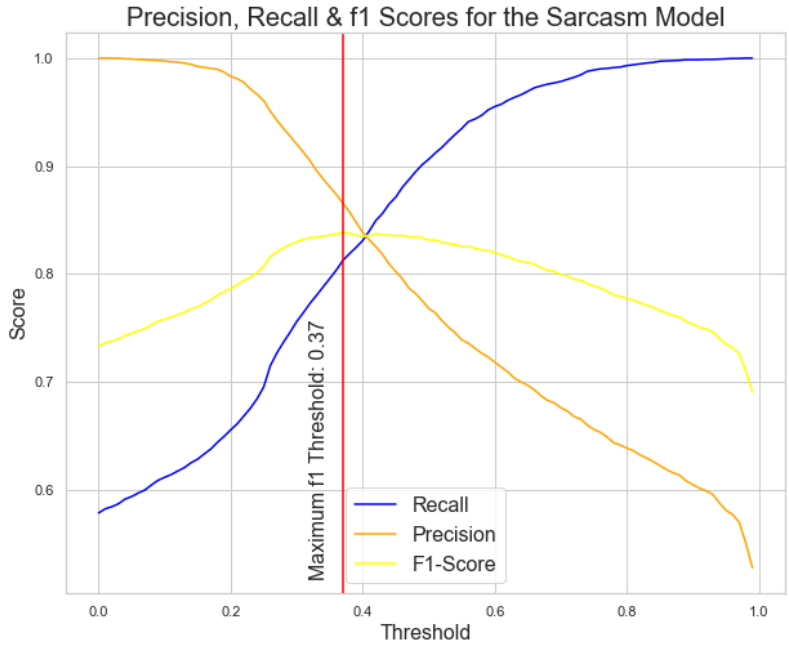


Figure 9. Accuracy and balanced accuracy scores for the sarcasm model across a range of threshold values.

while providing a classification performance metric. Using the standard threshold of 0.5, the sarcasm model yielded accuracy and balanced accuracy scores of 0.8209 and 0.8305, 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 an identical maximum accuracy score and threshold compared to the default threshold value. However, varying the threshold values produced a relatively larger gain in the maximum balanced accuracy score. The maximum balanced accuracy score of 0.8365 occurred at a threshold of 0.59 (Fig. 9).

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

Business Scenario Three: Balance the Risk of Mislabeling 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 (using 0.01 increments). The largest f1 score of 0.8383 occurred at a threshold value of 0.37 (Fig. 10). Implementing the maximum f1 threshold of 0.37 reduced the number of false negatives by 549 (757 compared to 1306 at the default threshold), but elevated the number of false positives (1124 compared to 447 at the default threshold) (Fig. 11.)

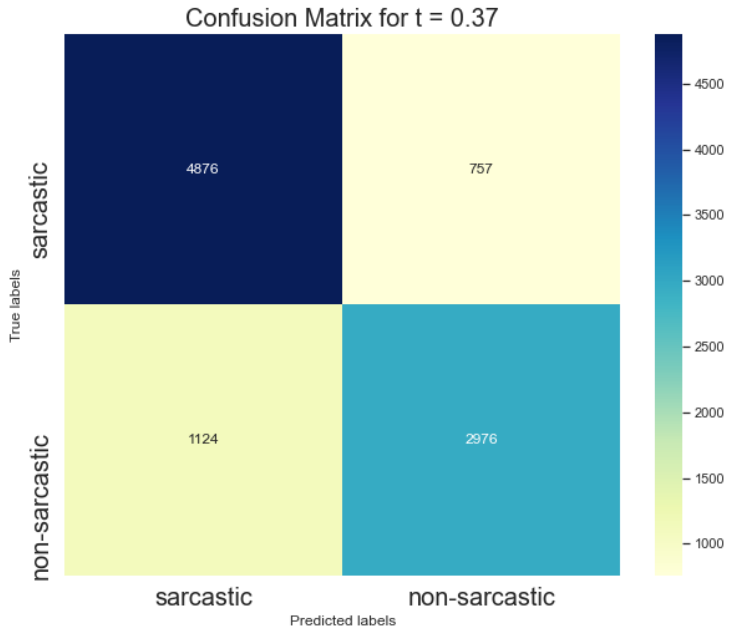


Figure 11. Confusion matrix for the sarcasm model at the threshold which maximizes the f1 score.

Business Scenario Four: Improving Sentiment Analysis Using Sarcasm Scores

I tested the model's ability to improve a sentiment analysis of novel data. As a source of novel data I obtained a copy of the sentiment140 dataset. 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 for the sentiment analysis, I subsampled the novel data to match the 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 across 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). The sentiment analysis of the novel data achieved a maximum f1 score of 0.7524 at a threshold value of 0.35 for correctly predicting the labels of individual tweets (Fig. 12). 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 (Tables 6 and 7). For a range of sarcasm probability scores between 0-1 (at 0.01 increments), I flipped the original sentiment140 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.

Flipping negative labels alone and flipping both together failed to increase f1 scores above the highest naïve Bayes f1 score (0.752446) for any sarcasm probability threshold value (Fig 13). However, flipping positive scores alone yielded a very small, yet measurable increase in the f1 score (0.752468) associated with a sarcasm probability threshold of 0.96 (Fig. 14).

Table 6. Ten most sarcastic tweets from the sentiment140 data

Original Tweet	Preprocessed Tweet	Sarcasm Probability
@lindasmith247 Will take a look	take look	0.9998
Is getting very sad!!!!	getting sad	0.9990
@ray_anthony damnn it. my secret is ruined..	secret ruined	0.9854
@ferlishious It is naked, a snail is homeless.	naked homeless	0.9826
@hyunjin4jc pouring, even!	pouring even	0.9722
@GlamorousSlim awww not even to me	even	0.9722
@drewflo22 Now you and @lexfritter are even	even	0.9722
didn't even do all my hw slackinggggg mimi	even	0.9722
even tweetdeck testen	even tweetdeck testen	0.9722
Not even me?	even	0.9722
@lindasmith247 Will take a look	take look	0.9998

Table 7. Ten least sarcastic tweets from the sentiment140 data.

Original Tweet	Preprocessed Tweet	Sarcasm Probability
@butterflysong You're welcome. I can imagine you are. That is very sad	you re welcome imagine sad	0.016436
thank for following me,	thank following	0.012239
THANK YOU PEOPLE..FOR FOLLOWING MEEE	thank following	0.012239
@doubm hahah youre welcome	hahah you re welcome	0.010098
@mdig1968 You're welcome Manos.	you re welcome manos	0.010098
@Zammie255 youre welcome!!! <3	you re welcome	0.010098
@dzschille You're welcome.	you re welcome	0.010098
@caramelflavored You're welcome.	you re welcome	0.010098
@stelabird You're very welcome... ?	you re welcome	0.010098
@team_Kaos You're welcome	you re welcome	0.010098
@butterflysong You're welcome. I can imagine you are. That is very sad	you re welcome imagine sad	0.016436

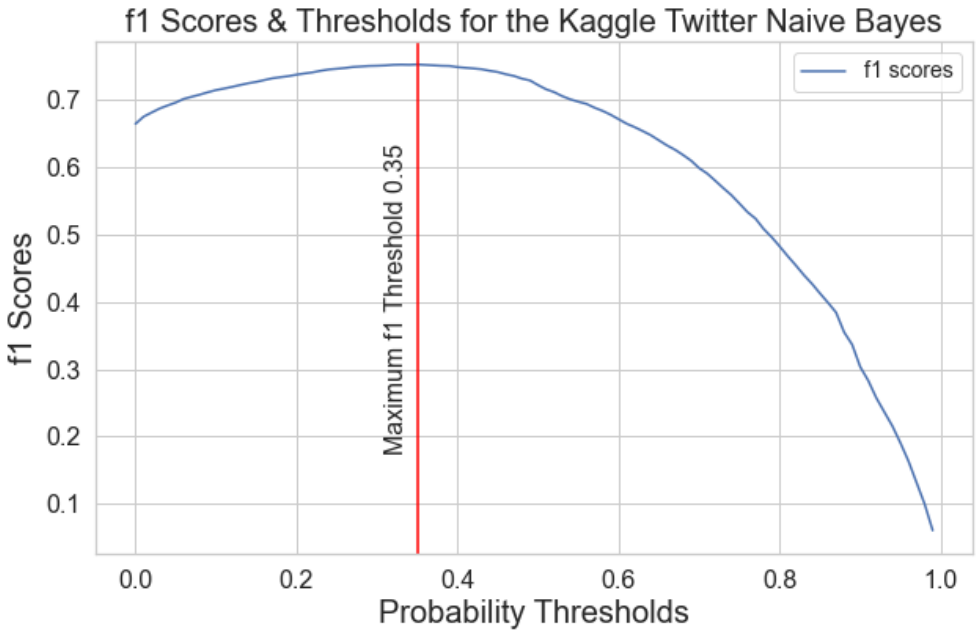


Figure 12. Variation in f1 scores in relation to probability thresholds for the sentiment analysis of the sentiment140 data using multinomial naïve Bayes.

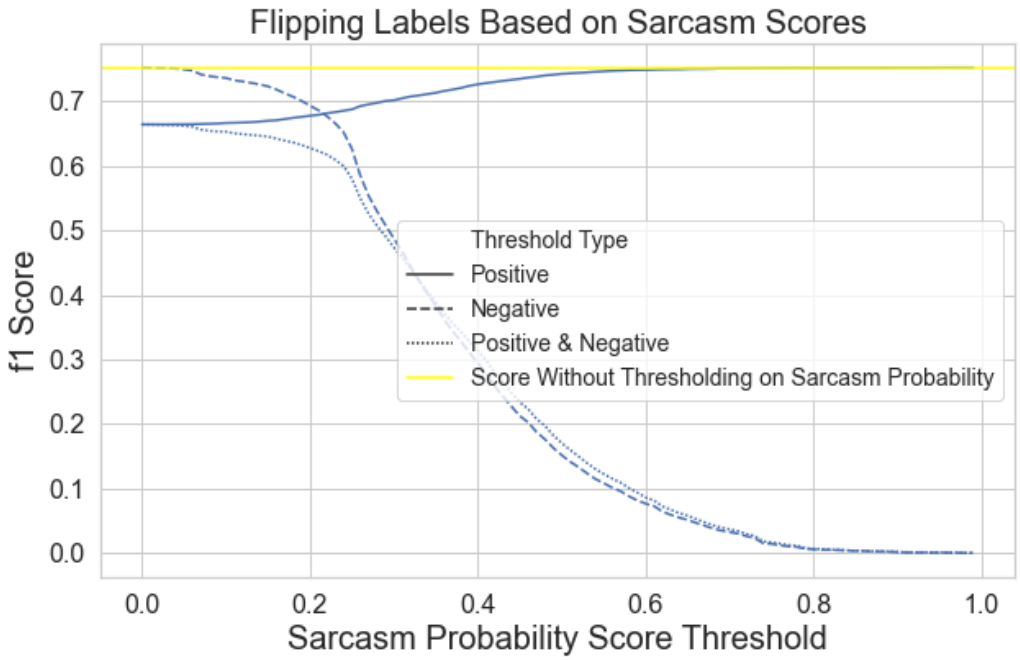


Figure 13. f1 scores associated with flipping sentiment140 labels based on sarcasm probability thresholds.

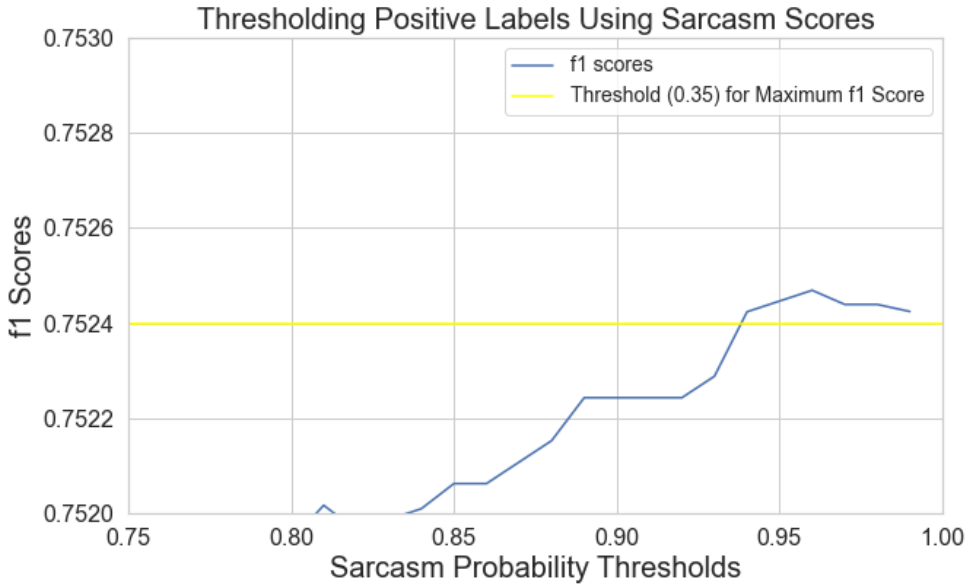


Figure 14. f1 scores associated with flipping positively-labelled sentiment140 data based on sarcasm probability thresholds.

WEB APP

I created a web app for users to test the sarcasm model for themselves. The landing page prompts users to enter a novel Twitter username (Fig. 15).

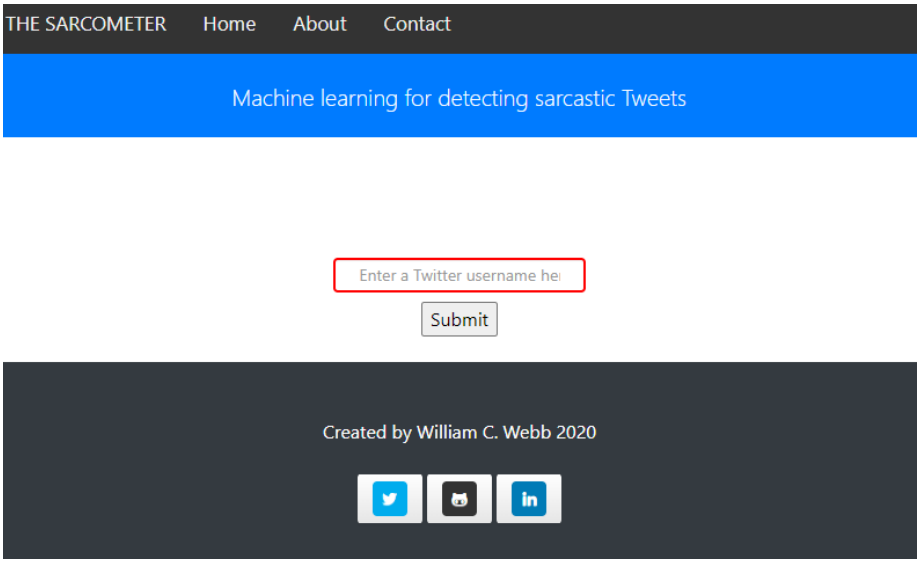


Figure 15. Landing page for the web app.

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. 16). The bottom portion of the results page contains a five-panel carousel which users navigate by operating the carousel's forward and backward

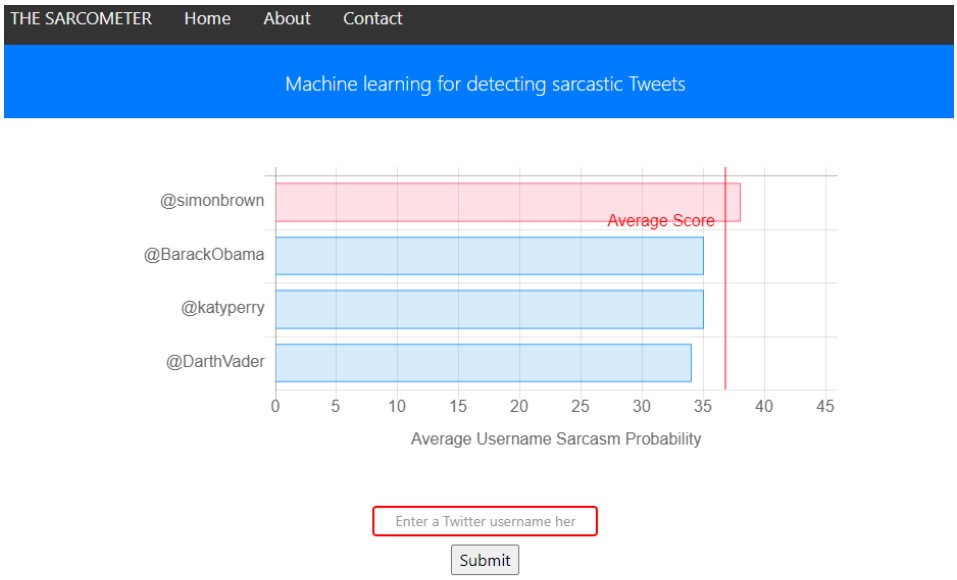


Figure 16. Top portion of the results page returned after a user enters a novel username and clicks the 'submit' button.

arrows. The carousel contains three sections. The top section of the carousel 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. 17). 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. 18). 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.

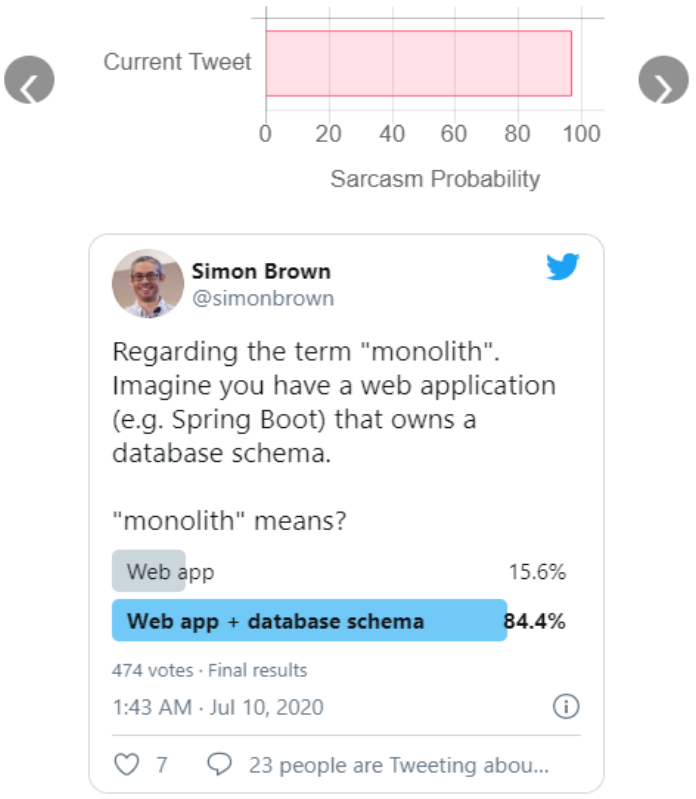


Figure 17. The top two sections of the carousel from the results page. The top section contains a horizontal bar chart with the score for the most-sarcastic tweet created by the novel username. Below the top section, the middle section displays the Twitter widget for the most-sarcastic tweet generated by the novel username.

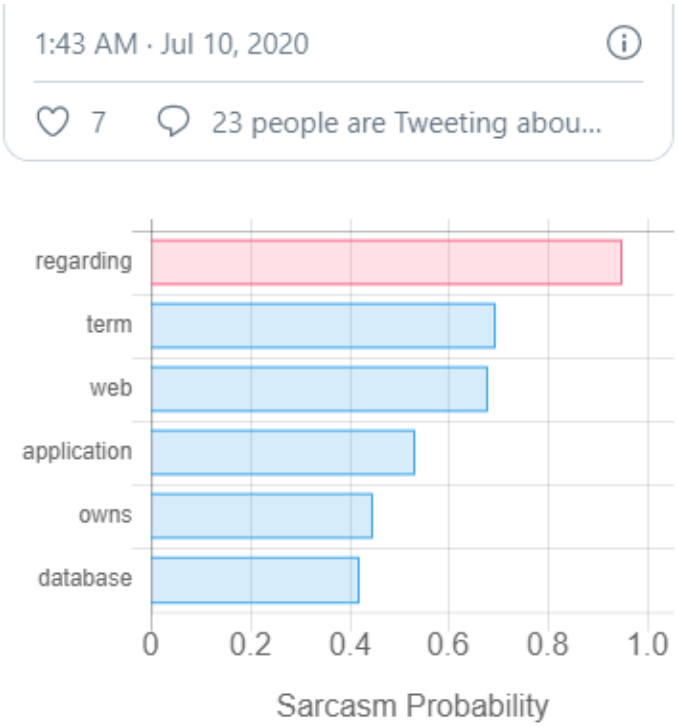


Figure 18. Bottom third of the results page carousel. The bar chart displays the sarcasm score for each word in the current tweet visible in the Twitter widget above (Fig. 17).

CONCLUSION

In this project, I explored the practical business applications of sentiment analysis using NLP and machine learning to detect sarcastic tweets. After choosing the best-performing classifier and tuning it, the model performed well for identifying sarcastic tweets. The accompanying web app demonstrates the model for interested users.

In addition, I tuned the sarcasm model to maximize its utility for three hypothetical, practical business applications. Tailoring the model for each scenario involved choosing the correct classification metric and adjusting the classification threshold to maximize the respective classification metric. In most cases. the new threshold differed from the default classification threshold (0.5).

Finally, I explored the model's ability to complement a sentiment analysis conducted using a novel dataset. To this end, I developed a simple naïve Bayes model for classifying tweets labeled as positive or negative from the sentiment140 dataset. I used the sarcasm model to calculate a sarcasm probability score for each tweet in the novel dataset. For a range of sarcasm threshold values, I flipped the scores for the positive labels, negative labels and both types before calculating a new f1 score for the naïve Bayes model. The sarcasm model failed to improve the f1 score when flipping negative labels, or when negative and positive labels together. However, the sarcasm model produced a very small improvement over the original f1 score when flipping positive labels alone.

A few potential explanations exist for the failure to significantly improve the sentiment analysis using the sarcasm model. Semantic differences between sarcasm and negativity used for labeling the sarcasm and sentiment140 data, respectively represents one potential explanation. For example, sarcastic tweets likely represent a subset of a much broader class of negative tweets. The limited semantic overlap between sarcasm and negativity therefore hampers the potential for the sarcasm model to

improve a sentiment analysis such as the sentiment140 data that employs a very broad classification approach (*i.e.* "positive" versus "negative").

Methodological differences used for collecting and filtering tweets likely resulted in very different data between the two datasets and these differences could also explain the poor ability of the sarcasm model to improve the sentiment analysis. The authors of the sentiment140 data labeled tweets using emoticons, whereas the sarcasm data labeled tweets using hashtags, while also removing emoticons during the preprocessing stage. Data collection for the sarcasm data also included a location filter which excluded tweets without geographic information specific to the United States. In contrast, the sentiment140 data collection process lacked a geographic filter, thus potentially introducing greater variation within the sentiment140 data compared to the sarcasm data.

Inherent differences between datasets combined with the very slight improvement in the sentiment analysis for flipping positive labels suggest the sarcasm model possesses some potential utility for improving sentiment analyses of novel data. The ability of the sarcasm model to improve a sentiment analysis probably depends in large part on the degree of semantic and methodological differences between datasets. To test this hypothesis, the sarcasm model could be applied to a novel dataset with fewer semantic differences compared to the sarcasm data and collected using methods used to generate the sarcasm data.

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.