Classification of Political Bias Presence in Social Media

Joseph Cruz
Computer and Mathematical Sciences Department
Lewis University
Romeoville, IL
josephbcruz@lewisu.edu

***Abstract*— The utilization of social media has increased in popularity exponentially over the last decade. Social media activity poses as a lucrative source for user data that can be used for the development of effective marketing campaigns for either commercial or political use. However, these large sources of data need to be sorted based on relevance before the data can be used for marketing purposes. The proposed binary classification models are intended for political bias detection in social media posts. To elaborate, the political bias detection is looking for posts that are related to politics rather than the political alignment of the post; furthermore, the intent of these models are to pose as the preliminary model required for sorting the data according to relevance to politics. The two binary classification models proposed are a multinomial naive Bayes classifier and support vector machine model using a cosine similarity kernel. The two models are examined using social media data gathered that is classified as either neutral or partisan. The models are then analyzed for effectiveness and discussed.**

Keywords—Multinomial Naive Bayes, Support Vector Machine, Cosine Similarity Kernel, Binary Classification, Machine Learning, Social Media

I. Introduction

Social media usage has increased exponentially over the course of the last decade and has created a new avenue for marketing to reach consumers. From personalized ads to relevant news targeting, marketing strategies have drastically changed due to the introduction of social media. Many of these marketing strategies are based on the identification of an individual's biases. In other words, understanding what an individual is interested in or what they are partial to is imperative to establishing an effective marketing strategy. Although these market strategies may seem incredibly complex, they can be simplified to basic classification models. Furthermore, they can be reduced to basic *binary classification* models, where the models need to properly identify consumers that are interested and those that are not interested. After determining whether a consumer is interested or not, the information can then be used to correlate a bias to the individual, which can then be used further upstream in the marketing strategy. This initial classification is important for the rest of the marketing model, as without determining the presence of interest in the item/idea, the marketing strategy cannot determine to whom they should market their product/idea.

The marketing of politically based news and ideas is one example of marketing in which an individual's political bias dictates the kind of political marketing that will be marketed to them (i.e. democratic, republican, libertarian, etc.). To properly identify the alignment of political marketing that should be marketed, the individual's political bias must be understood. However, to properly identify an individual's political bias, the marketing model needs to learn from the individual. For the model to learn from the individual, user data that is relevant to politics is required. The implementation of a binary classification model at this initial level of the overarching marketing model that can identify whether data is relevant to politics is pivotal in the development of a truly effective marketing model.

In this paper, two different binary classification models are examined and applied in the determination of political bias presence in social media posts. These binary classification models that will be explored are a *multinomial naive Bayes* (MNB) classifier and a *support vector machine* (SVM) classifier using a *cosine similarity kernel*. The models are tested with a series of social media data that is composed of both political and nonpolitical posts [1]. These models' effectiveness is then discussed and evaluated.

II. Background

Classification models are a cornerstone of supervised machine learning that have been utilized in a multitude of complex models and a myriad of different applications. The main purpose of these predictive classification models is to predict to what class a given data point belongs. Binary classification, one of the simpler classification models, are used to classify data between two classes. There are several different algorithms that can be used for binary classification models. Since this paper utilizes the MNB and SVM classifiers, only these algorithms will be discussed.

The MNB classifier is a classification algorithm that is commonly used in natural language processing and text classification applications. MNB is part of a larger class of predictive classification algorithms known as the *naive Bayes classifiers*. These naive Bayes classifiers rely *on Bayes' theorem*, an equation that is used for the calculation of conditional probabilities [2]. The equation for Bayes' theorem is given by:

$$P(C_i \mid x) = [p(x|C_i)\, P(C_i)] \,/ [\textstyle\sum^{k}_{k=1} p(x|C_k)\, (P(C_k)]. \quad (1)$$

The term k represents the number of classes, x is representative of the value being classified, and C_i is the class in question. Furthermore, the theorem calculates the posterior probabilities of a data point x being classified as C_i by using the *class likelihood* ($p(x|C_i)$) and the *prior probability* ($P(C_i)$) and the evidence (bottom term of the equation). The main deciding factor for a Bayes classifier comes down to which class has the max posterior probability. Since the naive Bayes classifiers output posterior probabilities, this is as simple as determining which has the highest posterior probability given x. Furthermore, naive Bayes classifiers work under the assumption that the

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

“attributes are conditionally independent” [3]. In other words, features within a given class are not related to the presence of any other feature. MNB classifiers can be thought of as an instance of a naive Bayes classifier. In application to text classifications, MNB classifiers utilize a similar conditional independence assumption, where “the probability of each word event in a document is independent of the word’s context and position in the document” [4, p. 43]. Furthermore, the distribution for an MNB classifier is a multinomial distribution, where the distribution is based off word counts and probabilities. In the case of only two classes, the Bernoulli distribution is utilized, but as the number of classes increases, the multinomial distribution is used.

SVMs are supervised learning algorithms that can be used in both classification models and regression models. SVMs have amassed large popularity in text classification applications. The primary function of SVMs are to establish a decision boundary in the form of a hyperplane that can distinctly classify data points such that the margin between the two classes is maximized. To maximize the margin, the SVM algorithm relies on the support vectors and the utilization of the loss function hinge loss [5]. In cases that the data is not linearly separable, the *kernel trick* is utilized, where kernel functions are used to transform the data into higher dimensions where a decision boundary can be established. These kernel functions “return the inner product between two points in a suitable feature space” [6]. In this paper, the cosine similarity kernel is used, where the cosine of the angle between two nonzero vectors is used to determine the inner product that is scaled to have a length of 1. The cosine similarity kernel is commonly used in text classification applications as the text input is characterized as a vector, where the value is representative of the frequency of a text within the document.

III. Methods

A. Analyzing and Cleaning the Dataset.

The dataset was composed of 5000 unique social media posts from Facebook and Twitter that had a total of 21 different columns with information regarding each of the unique social media posts. The data are split up into two different classes based upon the political association of the post. The first class is ‘partisan’, where there is a political party bias, but it does not matter what political party it is as long as there is one. The other class is ‘neutral’, where there is no political party bias present. A simple class analysis was performed to understand the spread of the two classifications types within the data. Of the 5000 data points, 3689 were labeled as ‘neutral’ and 1311 were labeled as ‘partisan’.

To model this data with the proposed binary classification models, the data needed to be cleaned. Cleaning of the dataset began by extracting the necessary columns from the dataset and storing them for use. After isolating the data, the text underwent processing to transform all capital letters into lower case letters and to remove special characters and emoticons. The cleaned text was then appended onto the stored data. At this point in time, the stored data had a total of 3 columns: ’text’, ’bias’, and ‘cleaned text’. After cleaning the text, the dataset was checked for any null values to ensure that there was no missing information. Luckily, the dataset did not have any null values. Then, the ‘text’ column was removed so that the stored data only held information that will be used in the models. The datapoints were also randomly reindexed so that they were in a random order. Then, a list of stop words was established so they could be removed from the text during tokenization. The stop words consisted of common words and individual letters that likely had no correlation with the classifications (i.e. the, an, a, b, c, etc.). Then, at this point, the text data had to be split up into bags of words to tokenize the words into individual elements that can be processed in the learning models.

B. MNB Learning Model

The first step of the MNB learning model was to generate a bag of words from the ‘cleaned text’ data for each text. The resulting bag of words was then vectorized for processing. The data was then split into the training and testing datasets. The training of the model was performed with 80% of the samples (4000 samples) and the testing was performed with 20% of the samples (1000 samples). Then, the model performed predictions on the testing dataset and the accuracy and confusion matrix for the predictions was determined. Precision, recall, and F1 scores were also determined. Furthermore, the model was tested with a single sample of text from each classification found in the dataset (controls) as well as 3 additional texts (tests).

- “Obamacare spawning a new concept: "medical homelessness."” (‘Partisan’ control sample)
- “Please join me today in remembering our fallen heroes and honoring the men and women currently in military service for their sacrifices.” (‘Neutral’ control sample)
- “who is going to see Antman today?” (‘Neutral’ test sample)
- “vote republican” (‘Partisan’ test sample)
- “vote for obama" (‘Partisan’ test sample)

C. Cosine Similarity SVM Model

The first step of the cosine similarity SVM model was to generate a bag of words from the ‘cleaned text’ data for each text. The bag of words was then vectorized for processing. The data was then split into training and testing sets, where 80% of samples (4000 samples) were used for training and 20% of samples (1000 samples) were used for testing. The model uses a support vector classifier with the cosine similarity kernel and a value of 0.98 for the cost value (c-value). The model then performs predictions upon the testing dataset and the accuracy, confusion matrix, precision, recall, and F1 scores were all determined. The model was also tested with the same sample texts as in the MNB learning model.

IV. Data and Results

A. MNB Learning Model

The MNB learning model data can be seen in Figure 1. The classification accuracy of the model was 76.2%, where the model was able to properly classify the data as either a ‘neutral’ or ‘partisan’ post 76.2% of the time. The confusion matrix shows that of the 1000 samples tested, 592 samples are classified as true ‘neutral’ samples, 133 samples are

```
Testing data assessment

Accuracy:  0.762
Confusion Matrix:
 [[592 133]
 [105 170]]
(row=expected, col=predicted)

Multinomial Naive Bayes Results:

              precision    recall  f1-score   support

     neutral       0.85      0.82      0.83       725
    partisan       0.56      0.62      0.59       275

    accuracy                           0.76      1000
   macro avg       0.71      0.72      0.71      1000
weighted avg       0.77      0.76      0.77      1000

Obamacare spawning a new concept: "medical homelessness."
MNB Prediction:  [[0.13292597 0.86707403]]

Please join me today in remembering our fallen heroes and honoring the men and women currently in military service for their sa
crifices.
MNB Prediction:  [[9.9998057e-01 1.9430000e-05]]

who is going to see Antman today?
MNB Prediction:  [[0.89648861 0.10351139]]

vote republican
MNB Prediction:  [[0.367427 0.632573]]

vote for obama
MNB Prediction:  [[0.34822884 0.65177116]]
```

Figure 1. Results of the MNB learning model using the testing data.

misclassified as ‘neutral’, 105 samples are misclassified as ‘partisan’, and 170 samples are truly classified as ‘partisan’. Classification of ‘neutral’ posts had a precision score of 0.85 while the classification of the ‘partisan’ posts had a precision of 0.56. The weighted average of the model’s precision was 0.77. The recall for the ‘neutral’ classification was 0.82 and 0.62 for the ‘partisan’ recall with a weighted average of 0.76 for the model. The F1 score of the classifications were 0.83 and 0.59 for the ‘neutral’ and ‘partisan’ classes, respectively, with a weighted average of 0.77 for the model. Since the MNB classifier produces posterior probabilities for a given data point for the possible classes, the values of the sample texts results are posterior probabilities that the data point is either ‘neutral’ or ‘partisan’. In other words, the output of the sample text predictions are posterior probabilities for the sample being either ‘neutral’ or ‘partisan’. The following values are ordered as ‘neutral’ posterior probability followed by the ‘partisan’ posterior probability. The ‘Partisan’ control sample yielded 0.1329 and 0.8671 and the ‘Neutral’ control sample yielded 0.9999 and 0.000001943. The ‘Neutral’ test sample yielded 0.8965 and 0.1035 while the first ‘Partisan’ test sample yielded 0.3674 and 0.6326 and the second ‘Partisan’ test sample yielded 0.3482 and 0.6518.

B. *Cosine Similarity SVM Model*

The SVM learning model data is shown in Figure 2. The classification accuracy of the model was 76.6%, where the model could accurately classify text as either ‘neutral’ or ‘partisan’ 76.6% of the time. The confusion matrix shows that of the 1000 samples tested, 679 samples are classified as true ‘neutral’ samples, 46 samples are misclassified as ‘neutral’, 188 samples are misclassified as ‘partisan’, and 87 samples are truly classified as ‘partisan’. The classification of ‘neutral’ posts had a precision score of 0.78 and the classification of the ‘partisan’ posts had precision score of 0.65. The model’s weighted average of precision was 0.75. The recall for ‘neutral’ classification was 0.94 and the recall for ‘partisan’ classification was 0.32. The weighted recall average of the model was 0.77. The F1 score of the ‘neutral’ classifications was 0.85, while the ‘partisan’ classification F1 score was 0.43. The weighted F1 score average for the model was 0.74. It is worth noting that the model predictions output the sample’s classification that the model predicts. The model predicted that the ‘Partisan’ control sample was ‘partisan’ and that the ‘Neutral’ control sample was ‘neutral’. The ‘Neutral’ test sample was predicted as ‘neutral’ and the two ‘Partisan’ test samples were predicted to be ‘partisan’.

```
Testing data assessment

Accuracy:  0.766
Confusion Matrix:
 [[679  46]
 [188  87]]
(row=expected, col=predicted)

SVM Results:

              precision    recall  f1-score   support

     neutral       0.78      0.94      0.85       725
    partisan       0.65      0.32      0.43       275

    accuracy                           0.77      1000
   macro avg       0.72      0.63      0.64      1000
weighted avg       0.75      0.77      0.74      1000

Obamacare spawning a new concept: "medical homelessness."
SVM Prediction:  partisan

Please join me today in remembering our fallen heroes and honoring the men and women currently in mil
itary service for their sacrifices.
SVM Prediction:  neutral

who is going to see Antman today?
SVM Prediction:  neutral

vote republican
SVM Prediction:  partisan

vote for obama
SVM Prediction:  partisan
```

Figure 2. Results of the SVM learning model using the testing data.

V. Discussion

Of the 2 models, the model with the highest accuracy was the SVM model (76.6%), but both models retained an accuracy within only 0.4% of each other. It is also important to note that these values are based off a single set of training and testing data and there may be slight variance in the accuracy between different iterations of training and test data. In other words, since the models use a random set of data for the testing and training data, it is possible that one iteration of any of these models could have a slightly lower or higher accuracy than what is stated in this paper. The accuracy of the two models was further tested using the sample text predictions. The MNB model was able to consistently provide higher posterior probabilities for the appropriate classes for each text sample. For example, using the ‘Neutral’ control sample, the model generated a posterior probability for the ‘neutral’ class of 0.9999 and a posterior probability for the ‘partisan’ class of 0.000001943. This shows that the model was able to identify that the sample most likely belonged to the ‘neutral’ class. The Cosine Similarity SVM model was also able to correctly predict all sample texts including the ‘neutral’ and ‘partisan’ control samples.

Another important set of features to analyze in a model are the precision and the recall of the model. Generally, it seemed that the ‘neutral’ classifications were more precise than the ‘partisan’ classifications throughout. ‘Neutral’ classifications had precision scores of 0.78 and above while the ‘partisan’ classifications had precision scores ranging from 0.56 to 0.65. However, at the same time, the recall for the ‘partisan’ class was relatively low throughout the 2 models, with the lowest being in the SVM model (0.32) and the highest being in the MNB model (0.62). The case is different for the ‘neutral’ classifications, as the recall values range from 0.82 to 0.94. For instance, the MNB model had a precision for ‘partisan’ classifications of 0.56 and the recall was 0.62. Although the model can correctly identify at least 56% of ‘partisan’ samples, it is still missing a lot of the partisan samples that are there and it is misclassifying other data as ‘partisan’ samples. On the other hand, the SVM model had a ‘partisan’ precision of 0.65 and a recall of 0.32. The SVM model is more accurately classifying samples correctly, but it is so specific that it is missing samples that are ‘partisan’.

A key factor that could be playing into the low recall and precision values for ‘partisan’ classification is the composition of the original dataset. As noted previously, the number of samples that were labeled ‘neutral’ and ‘partisan’ were 3689 and 1311, respectively. The ‘neutral’ labeled data points are almost 3 times the number of ‘partisan’ labeled data points. This uneven distribution of labeled points may be the reason that the precision values and the recall are so high for the ‘neutral’ classifications but may also be the reason that the ‘partisan’ classifications being skewed. Without enough data points labeled as ‘partisan’, it is possible that the model is not completely learning how to properly classify the ‘partisan’ class. This distribution may be leading to a bias in the model. Evidence of this can be seen in both confusion matrices. Note that the bottom row of the confusion matrix denotes samples incorrectly classified and samples correctly classified as ‘partisan’, respectively. The total ‘neutral’ classifications in the top row of the confusion matrices have significantly more data points than the total ‘partisan’ classifications (~450 more data points).

To further investigate these testing data observations, the training data was also run with both models to compare the metrics of the models (Figure 3 and Figure 4). Using the training data, the MNB model had an accuracy of 91.5% with precision scores above 0.85, recall scores above 0.82, and F1 scores above 0.83 for both classes. Using the training data, the SVM model had an accuracy of 86.9% with precision scores above 0.86, recall scores above 0.55, and F1 scores above 0.69 for both classes. For both models, the training data runs provide better metrics than the testing data runs, which may be a sign of overfitting of the data due to some bias. The bias, as mentioned previously, may be a consequence of the dataset being used, but it may also have something to do with the kind of models that are being used. Naive Bayes classifiers tend to overfit data due its simple linear hypothesis function for more complex situations. Furthermore, SVM models have difficulty with highly skewed or imbalanced data sets as creating an effective decision boundary is difficult with more skewed data. Consequentially, these associated issues with the models may be a potential root cause for such overfitting that is observed.

```
Training data assessment

Accuracy:  0.91475
Confusion Matrix:
 [[2810  154]
 [ 187  849]]
(row=expected, col=predicted)

Multinomial Naive Bayes Results:

              precision    recall  f1-score   support

     neutral       0.94      0.95      0.94      2964
    partisan       0.85      0.82      0.83      1036

    accuracy                           0.91      4000
   macro avg       0.89      0.88      0.89      4000
weighted avg       0.91      0.91      0.91      4000
```

Figure 3. Results of the MNB learning model using the training data.

Although these models in their current state display signs of overfitting, it is possible that the models can be altered to provide much better results. For instance, the MNB model could undergo *bagging*, where the data is split into a few subsets and run on different classification algorithms. The MNB model and an SVM model could be paired through a pipeline and the average modeling efficiency can be determined. Furthermore, the models could be subjected to *boosting*, where a model is developed that improves upon the errors of the previous model until a minimum error is reached. Although these models may show overfitting, this has established a good first step in the development of a well performing classification model.

Regardless of the overfitting, both models have their pros and cons, but based on the weighted averages, it seems that the MNB model is the more effective model, by a slim margin. It is worthy to note that the differences between these models weighted averages are also within 5% of each other. It is also good to note that in the training data run, the MNB model retained higher overall scores. For example, the MNB precision values for both classes are 0.94 and 0.85 (‘neutral’ and ‘partisan’) and their respective recall values are 0.95 and 0.82. The SVM training data run only yielded precision values of 0.86 and 0.91 with recall of 0.98 and 0.55. The SVM training run was not able to properly identify the ‘neutral’ class nor recall the ‘partisan’ class as well as the MNB. Further, the weighted averages of the precision, recall, and F1 scores are approximately 4-5% higher in the MNB model than the SVM.

```
Training data assessment

Accuracy:  0.869
Confusion Matrix:
 [[2905   59]
 [ 465  571]]
(row=expected, col=predicted)

SVM Results:

              precision    recall  f1-score   support

     neutral       0.86      0.98      0.92      2964
    partisan       0.91      0.55      0.69      1036

    accuracy                           0.87      4000
   macro avg       0.88      0.77      0.80      4000
weighted avg       0.87      0.87      0.86      4000
```

Figure 4. Results of the SVM learning model using training data.

VI. CONCLUSION

The MNB and SVM learning models examined in this paper are a good demonstration of machine learning algorithms performing binary classification. The utilization of the MNB and SVM learning models are a step in the right direction in the development of a model that can accurately distinguish between partisan and neutral social media posts. In terms of this application, the MNB seems to be a suitable candidate to pursue further for a binary classification model, but other options may also be considered in the future. There will need to be some improvements made in the model to further increase its efficiency and effectiveness, but the proposed model is a good first step in the creation of a multitier model that will contribute to an effective marketing model.

REFERENCES

[1] Crowdflower, *Classification of Pol Social*, Crowdflower: data.world, inc.[Dataset].Available: https://data.world/crowdflower/classification-of-pol-social. [Accessed: August 19, 2020].

[2] Joyce, James, "Bayes' Theorem", in The Stanford Encyclopedia of Philosophy (Spring 2019 Edition), Stanford: Metaphysics Research Lab,Stanford University (2019). [Online]. Available: https://plato.stanford.edu/archives/spr2019/entries/bayes-theorem/. [Accessed: August 24, 2020].

[3] S. Asiri, "Machine Learning Classifiers," Medium, June 11, 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623. [Accessed: August 24, 2020].

[4] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," AAAI/ICML, Work Learn Text Categ, pp. 41-48, 1998. [Online]. Available: http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf. [Accessed: August 24,2020].

[5] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," Medium, June 7, 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed: August 24,2020].

[6] DATAFLAIR TEAM, "Kernel Functions-Introduction to SVM Kernel & Examples," DataFlair, November 16, 2018. [Online]. Available: https://data-flair.training/blogs/svm-kernel-functions/. [Accessed: August 24, 2020].