1. Abstract

A lot of data is generated every day, and an increasing proportion of data is in text form. Therefore, the rise of Natural Language Processing (NLP) has become more popular recently. It is important for computers to be able to detect whether a certain string of text is to be taken seriously, hence the idea of a sarcasm classifier. Our goal for this project is to make use of NLP and Data Science (DS) techniques to find the best method to predict whether a given text is sarcastic. Four supervised classification models were selected to conduct the classification task. Our results indicate that there is a marginal difference in the accuracy of the four models. The results did not fall into our expectations as we expected a larger difference in the accuracies of the models. We believe that to improve the accuracy and viability of the models, we should narrow the scope of the data set, i.e., only includes text regarding a specific topic. Aside from the data set, we could use other methods such as pre-trained language models i.e., BERT, which makes use of newer techniques such as transformers, which were trained on millions of words.

2. Introduction

There are many ways to approach NLP classification tasks. One of these approaches is to combine traditional DS algorithms with NLP techniques. In this paper, we made use of four supervised classification models (Linear Support Vector Machine, Multinomial Naïve Bayes, Logistic Regression, Random Forest) to conduct sarcasm classification. We then compared the effectiveness of these classification models by using cross-validation and confusion matrices, which are commonly used model evaluation techniques.

To use these models, we will need to make use of some data preprocessing and NLP techniques such as Tokenization, Lemmatization, Term Frequency-Inverse Document Frequency (TF-IDF), as we are working with text data.

The data set we selected was selected from Kaggle, an ML/DS community. It can be found at the following link: https://www.kaggle.com/datasets/danofer/sarcasm. The data was generated by scraping comments on Reddit containing the /s tag. The tag is often used by Reddit users to indicate that their comment is sarcastic (Ofer).

Our hypothesis in this paper was that the Random Forest technique would provide the best results, as it had the least assumptions out of the models we selected.

3. Methods

Data Set:

The data set selected has 1.3 million sarcastic comments from the internet community Reddit and there were balanced, and imbalanced (true distribution) versions of the data set provided, with the label 0 and 1 indicating the comment to be not sarcastic and sarcastic respectively. The balanced data set was selected due to the true distribution of comments being about 1:100 (Ofer), which would lead to extra difficulties in training the classification models.

Data Preprocessing & NLP Techniques:

To prevent any errors, we had to remove any rows that were missing data. We then proceeded to randomly sample from the 1.3 million lines of data due to our limited computing power. Normally, other projects would require us to remove any special symbols and make all letters in the text lowercase, but in this case, we wanted to also include the use of punctuations and uppercase letters when considering sarcastic comments, hence why we decided against this step.

The first NLP technique, tokenization, splits the string of text into individual words (tokens) and outputs it as a list of words, in which each word in the list is a “token”. Each “token” then goes through the process of lemmatization. Lemmatization takes into the context of the sentence, then reduces inflectional forms of words to a root form. Afterward, TF-IDF is applied

to the whole dataset. TF-IDF is an unsupervised technique that proportions each word in text string data and includes a penalty term for frequency if it appears too often in the text document/dataset. The benefit of using an unsupervised technique is that it uses machine learning algorithms to analyze and associate unlabeled datasets and discovers hidden patterns in data without the need for human intervention (IBM). TF-IDF basically treats each word as a random variable, and as mentioned above, proportionally increases the value of the word if it appears often in one text data but then proportionally decreases the value if it appears across multiple lines of text data. An example of this would be the word "the". The word "the" might appear in one comment multiple times, and TF-IDF would increase its value of it. But, as the word "the" appears across a lot of comments, TF-IDF would then decrease its value of it. TF-IDF was chosen over the bag-of-words (BOW) method as BOW only creates a set of vectors documenting the count of word occurrences in text and does not consider the context.

Train-Test Split:

To effectively train and test the model, we separated the randomly sampled data into a train-test split of 0.8 and 0.2 respectively. This split was chosen due to the usual convention of selecting around 75% - 80% of data to train, with the remaining to test.

Model Selection:

The four models that were selected by our group were Linear Support Vector Machine (LSVM), Multinomial Naive Bayes (MNB), Logistic Regression (LR), and Random Forest (RF). These four models all fall into the category of supervised classification models. Supervised classification models are designed to train algorithms into classifying data or predicting outcomes accurately (IBM). For example, they predict discrete values such as Male or Female, Yes or No, and in this case, sarcastic or non-sarcastic.

LSVM

MNB

LR

RF

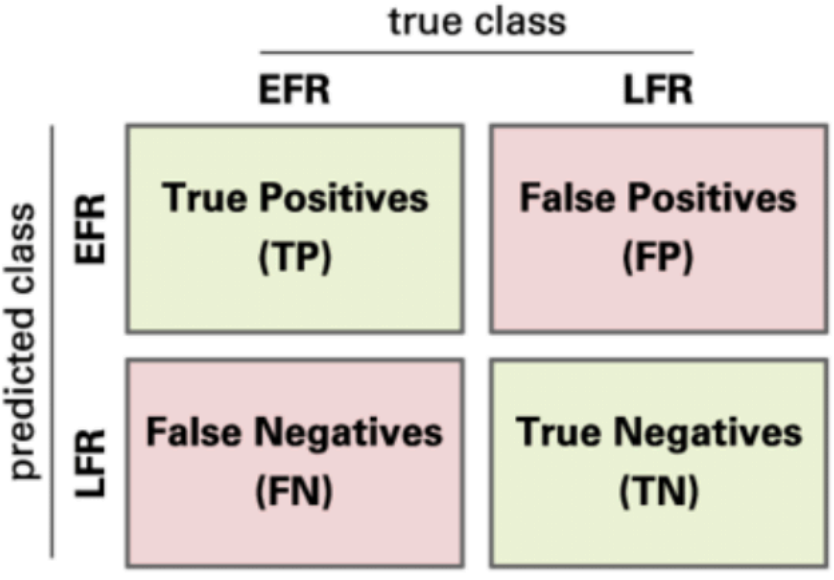
4. Results and Discussion

To effectively evaluate the results of our classifiers, we first introduce the concepts of Confusion Matrices (CM), Receiver Operating Characteristic (ROC) Curves & Area Under Curve (AUC) Value, and k-fold Cross-Validation (kCV). CM and kCV are common techniques that evaluate the accuracy of classification models.

Evaluation Methods:

Confusion Matrices:

CM is a matrix that shows the results of a model's prediction in a matrix. In this case, as the model is attempting to predict binary classes, the CM will be of size 2x2. In this matrix, we will be able to obtain the true-negative ($CM_{0,0}$), true-positive ($CM_{1,1}$), false-negative ($CM_{1,0}$), and false-positive ($CM_{0,1}$) values of the prediction from the model. The following image shows an illustration of a CM:



(Bittrich)

The CM for each model is shown in the following table:

Model	CM
LSVM	[[595 319] [393 492]]
MNB	[[495 237] [317 362]]
LR	[[640 274] [376 509]]
RF	[[651 263] [436 449]]

From the CMs, we can observe that LR performs the best in the sense that it has the highest number of accurate predictions, while MNB has the lowest.

The benefit of using a CM is that it provides information on what type of error is being made by the classifier, and it eases the calculation of the accuracy rate, plotting ROC Curves, and getting AUC values.

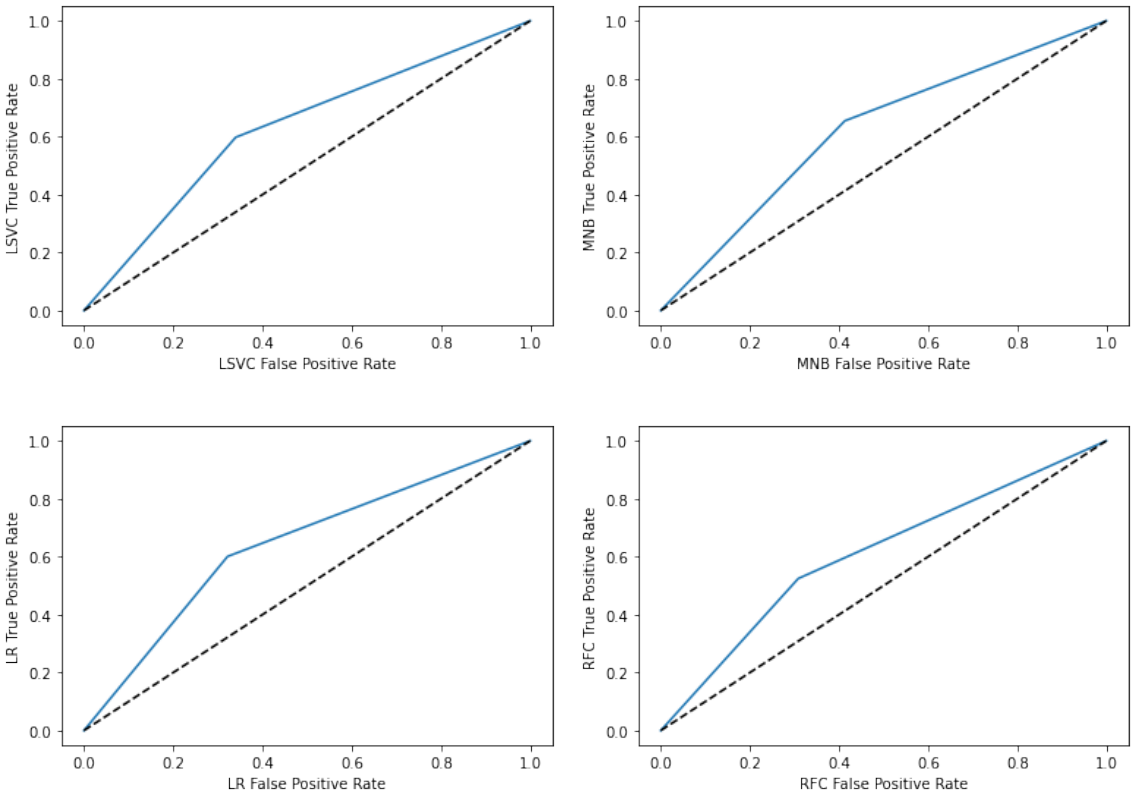
ROC Curves & AUC:

ROC curves are plots that show the ability of a binary classifier as the classifier boundary threshold is varied. Values of the threshold are the minimum probability for a data point to be classified into the positive class (in this case, sarcastic). It plots the True Positive Rate (tpr) against the False Positive Rate (fpr). The following equations calculate the tpr and fpr respectively:

$$TPR = TP / (TP + FN)$$

$$FPR = FP / (FP + TN)$$ (Accuracy)

The following four graphs are the ROC Curve of each model:



The dotted diagonal line represents the instance where the tpr equals the fpr. A point on the diagonal in our context represents the proportion of correctly classified sarcastic examples is the same as the proportion of incorrectly classified samples that are not sarcastic. Hence, at the two endpoints of the curves, they are to be equal as the threshold. On the left endpoint of the curve, the threshold is at 0 (all data points are to be classified as sarcastic), while at the right endpoint, the threshold is set to be at 1 (requiring 100% probability to be classified as sarcastic). Now, as the threshold varies between 0 and 1, the tpr and fpr values change as well, due to the values in the CM being adjusted accordingly. The ROC curve is then plotted along these values.

For a classifier to be considered good, we aim to have it as close as the top left corner, as the top right corner indicates that tpr is at 1, while the fpr is at 0. This is the theoretically best classifier as it correctly classifies all data into their actual classes. From our graphs, we can see that for all the models, the optimal threshold of the LSVM and RF models is at about

fpr ≈ 0.35 and tpr ≈ 0.60, while the optimal threshold of the MNB and LR models is at about fpr ≈ 0.40 and tpr ≈ 0.63.

The AUC value of the ROC curves allows for easy comparison of ROC curves with each other. The AUC values for each model are summarized into the following table:

Model	AUC Value
LSVC	0.6289
MNB	0.6208
LR	0.6394
RFC	0.6089

As we can see, the best model is LR, while the worst model is RF. But all the AUC values close to each other and are near the AUC value of the diagonal (0.5). This implies that through the evaluation of ROC curves and AUC values, the models are slightly effective at classifying sarcasm.

K-Fold Cross-Validation:

kCV is a method used to evaluate and compare different machine learning models. The parameter, k, decides the number of groups the data is split into, after shuffling the dataset randomly. For each group, it takes it out as a test group first, and assigns the remaining groups as training groups. Then, it fits the specified model on the training set and evaluates it on the test set. It then keeps the evaluation score, discards the model, and retrains it with the next group until all groups have been selected as a test set. The final score is the mean of the evaluation score of each group as a test group.

An illustration of kCV is shown in the following graph:

(Karl Rosaen)

The results of our 10-fold Cross-Validation is summarized into the following table:

Model	CV Score (10-fold)
LSVM	0.62
MNB	0.61
LR	0.64
RF	0.61

From the above table, we can observe that LR has the highest CV Score at 0.64, with MNB and RF tied at 0.61 for the worst performing model. But all these values are close to each other, so it is reasonable to suggest that they all have similar ability in classifying sarcasm. And as these values are all close to 0.5 (which suggests that the classification of sarcasm could be decided in a coinflip), they are not too effective in predicting sarcasm.

Discussion of Results:

The results of our investigation did not fall in line with our hypothesis and expectations. As mentioned earlier, we expected RF to be the best performing model due to it having the least assumptions out of the four models. But, according to our ROC Curves, AUC Values, and

10-fold Cross-Validation evaluation methods, RF is the worst performing classifier. However, we can confidently say that out of the four models, LR is the best at classifying sarcasm as it appears to be the most effective across all our model evaluation methods. But in the grand scheme of things, all these models are like each other in terms of accuracy, and all are only slightly effective at classifying sarcasm.

Possible Improvements:

After evaluating our investigation process and results, we believe that the reason why our models all low accuracies in classifying sarcasm is mainly due to three reasons, and we propose an alternate model aside from the four we mentioned above.

The first reason why our models were not too effective was due to the data set that we selected. Although the Reddit comments in the data set was definitively classified into sarcastic/not sarcastic, the comments are not of best quality. On Reddit, many users make use of informal language and abbreviations when commenting due to the website itself being a mostly casual community. Aside from this, from a qualitative sampling standpoint, many users made mistakes in typing their comments, which would affect the classification accuracy of our models. To improve this problem, we could either select another data set that has labelled data from other forms of text (i.e., Articles, Interviews etc.), or we could preprocess the data by adding a spell check correction feature (though it does not guarantee 100% success so that is a factor in which we must consider.)

The second reason is that our scope of sarcasm when initially proposing the idea was too large. There could be many types of sarcasm in different contexts, which might cause confusion for the models. For example, if someone commented “I had a great time here” regarding Disneyland, it would not be sarcastic. But, if someone commented the same phrase about prison, they might mean it sarcastically. Therefore, to improve this, we should have a

narrower defined scope, such as “Sarcasm in a Finance Context” or “Sarcasm in a Football Context” instead.

The last reason why our models did not perform to our expectations would be the feature selection process, more specifically, the number of features we had when training our models. In our code, we specified the maximum number of features to be 5000, as our group decided that 5000 different words would be a reasonable number of words to detect sarcasm. But this number was an arbitrary value. To improve this, we would need to undergo further investigation through our code to determine which number of features would be most appropriate.

Another suggestion to improve the performance of classifying sarcasm would be to consider models based on neural networks. There are two methods to apply neural networks. We can first train a neural network by ourselves, by specifying parameters such as the number of epochs (number of iterations over the training set), the learning rate (the change in weights after each optimizing step), number of hidden layers (layers of neurons between input and output layers) and many more. Or we could use pretrained neural network models which use transformers (i.e., BERT, GPT-3) and conduct finetuning to our sarcasm classifying task. This is the task of adjusting the parameters (in this case the weights/coefficients of an activation function of a neural network.) Both these methods require a deeper dive into the mathematics behind the theory of neural networks, and large amounts of computing power, but might increase ability of computers in understanding sarcasm.

5. Conclusion

In conclusion, after this investigation, we learned that we need to have a more specific scope when defining our topic. The process of collecting data/selecting a data set very

important and cannot be emphasized enough. Selecting which model to use is then a secondary problem if we have an ambiguous scope and low-quality data.

To improve on our paper, we suggest finding a higher-quality data set in a smaller scope, as well as attempting to apply a wider variety of models such as the neural networks mentioned above. Aside from this, an exploration in how to determine the optimal number of features when training the models we used in this paper would also be beneficial.

6. References

Accuracy, TPR, TNR, FPR and FNR Equations. https://researchgate.net/figure/Accuracy-TPR-TNR-FPR-and-FNR-Equations_fig4_318325614.

Bittrich, Sebastian, et al. “Fig. 2 Confusion Matrix. Exemplified CM with the Formulas of Precision...” ResearchGate, 26 June 2021, https://www.researchgate.net/figure/Confusion-matrix-Exemplified-CM-with-the-formulas-of-precision-PR-recall-RE_fig1_330174519.

Diagram of k-Fold Cross-Validation with K = 10. Image from Karl Rosaen ... https://www.researchgate.net/figure/Diagram-of-k-fold-cross-validation-with-k-10-Image-from-Karl-Rosaen-Log_fig1_332370436.

Ofer, Dan. “Sarcasm on Reddit.” Kaggle, 27 May 2018, https://www.kaggle.com/datasets/danofer/sarcasm.

“Supervised vs. Unsupervised Learning: What's the Difference?” IBM, https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.