COMP30027 Assignment 2 Report

1. Introduction

Analysing twitter sentiments can lead to predicting trends and can provide marketers with insights on what kind of products and services customers would be interested in buying.

In this paper we aim to classify tweets by the sentiment in them (one of *positive, negative,* or *neutral*), and find the most optimal classifier to carry out this task (the one that provides the highest accuracy values).

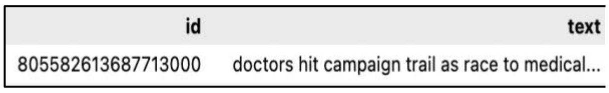
1.1 Classifiers

The models used are one baseline model (0R), Multinomial Naive Bayes, Support Vector Machine, Random Forest, and an Ensemble model that stacks all the individual models into a single model. Through the paper, we will evaluate different aspects of each model, and whether the features and results that they provide make them desirable to use for the sentiment analysis classification.

1.2 Dataset

1.2.1 Training Data

The dataset contains two main sections. The first section is a labelled training set, containing three columns: the tweet ID, the tweet text, and the associated class label (the sentiment associated with the tweet).



id	text
805582613687713000	doctors hit campaign trail as race to medical...

Figure 1: Example of an instance from the training data

The training set contains 21802 such instances, with 3715 tweets labelled "negative", 12659 tweets labelled "neutral", and 5428 tweets labelled "positive".

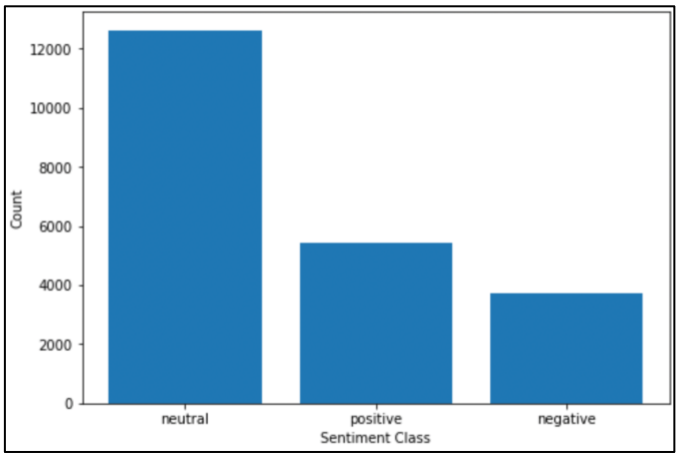
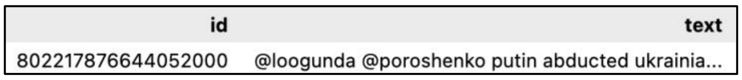


Figure 2: Instance counts of the class labels

There is an imbalance in the number of instances for each of the three class labels, and the implications of this will be discussed further in the report.

1.2.2 Testing Data

The second section of the dataset contains the set of testing instances, with a similar format to the ones in the training set, except that they are unlabelled (and testing for these is carried out using an external testing system). This set contains 6099 instances.



id	text
802217876644052000	@loogunda @poroshenko putin abducted ukrainia...

Figure 3: Example of an instance from the testing data

As the instances in the test set are unlabelled, we will not be using this for the initial part of the model evaluation. Instead, we will be splitting the training set further into training and validation sets to evaluate the performance of our chosen models.

2. Methodology

2.1 Feature Engineering

Prior to passing the data into the machine learning models, we carried out various steps to perform feature engineering on the raw text data.

2.1.1 Twitter Handles

First, we removed Twitter handles from the tweet text. Twitter handles start with an '@' symbol and are used to tag and identify the original poster of the tweet. They do not contribute to the sentiment of the tweet and hence are worth removing.

2.1.2 Numeric values

Next, we removed all numbers, and letters attached to numbers in the tweets. These also do not contribute to the sentiment of the tweet and are typically present in tweets only to provide quantitative information and facts- which are irrelevant to this study.

2.1.3 URL links

Any URL links (beginning with 'http' or 'www.)' present were also removed, as they do not contribute to sentiment of a tweet and removing them makes the tweet considerably shorter.

2.1.4 Special characters

Special characters and numbers were also removed. Although these sometimes form emoticons which convey sentiment, these characters cannot be converted into features during the vectorization process.

2.1.5 Tokenizing

Tokenizing involves breaking raw text into words called tokens. Tokens aid in interpreting the meaning of text by analysing the sequence of words, help in understanding the context of the text, and present the text in a format that is more efficiently processed by Natural Language Processing models.

2.1.6 Duplicate tweets (re-tweets)

Next, we removed all duplicate tweets from the dataset. As 're-tweeting' tweets is very common, there are multiple tweets that contain the same text. These only inflate the dataset and may introduce bias into the models.

2.1.7 Stop words

Stop words were removed from the tweet text as they are common across every tweet, and do not contribute to the sentiment or context of individual tweets. This allows the model to focus on the more important words in the tweet.

2.1.8 Vectorization

The final section of the feature engineering stage includes vectorisation; as machine learning algorithms are only able to take numerical inputs, vectorisation enables the text tokens to be encoded into numerical values that can be fed into models. We have explored two methods of vectorisation: Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

2.1.8.1 Bag of Words (BoW)

Using the BoW vectorizer, we created feature vectors for every training instance which can be later fed into any machine learning model. BoW creates feature vectors from text data containing the count of the occurrences of every word.

2.1.8.2 TF-IDF

Using the TF-IDF vectorizer, we created feature vectors for every training instance to be fed into any machine learning model. The TF-IDF vectorizer give a composite weight for each word/term by combining term frequency and inverse document frequency (Christopher D, et al., 2010).

2.2 Classifiers

The final testing dataset that we have does not have the target column. Hence to evaluate our model performance and tune our hyperparameter, we split the training data into training and validation sets using the 70-30 split. At this stage, we have two sets of vectorized features - i) Features generated using BoW and ii) Features generated using TF-IDF. We evaluated each of the following machine learning models (except baseline model) using both the vectorized features individually.

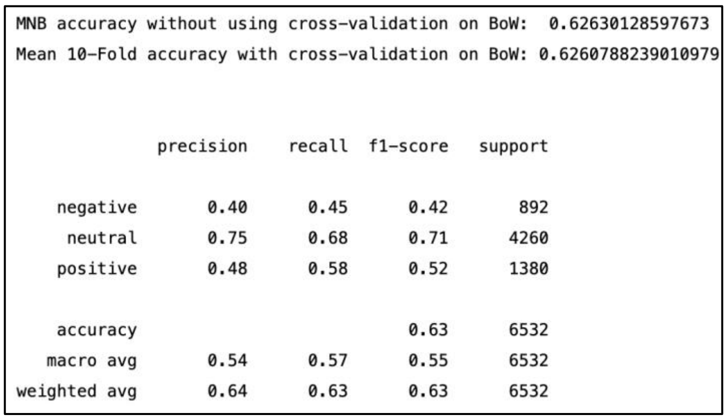
2.2.1 0R (baseline model)

We chose the baseline model to serve as a benchmark for model performance and contextualizes the results of the trained models.

2.2.2 Multinomial Naïve Bayes

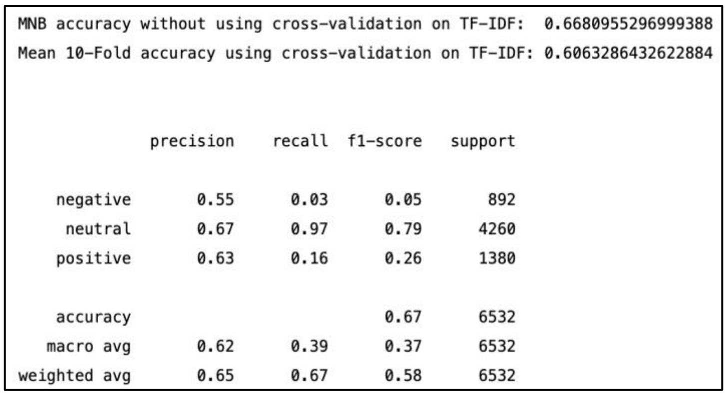
We implemented the Multinomial Naive Bayes classifier on both the dataset (produced by two different vectorizations) individually. We cross validated the data by using the 10-Fold cross validation and calculated the model accuracy with and without cross validation on the validation set to observe the difference. For the hyperparameter tuning, we performed the classification by using the Laplace smoothing.

We chose this model because according to (Kibriya et al. 2004) it works best with discrete features; while raw text data is continuous, the tokenization and vectorization process has discretised it.



```
MNB accuracy without using cross-validation on BoW:  0.62630128597673
Mean 10-Fold accuracy with cross-validation on BoW: 0.6260788239010979


              precision    recall  f1-score   support

    negative       0.40      0.45      0.42       892
     neutral       0.75      0.68      0.71      4260
    positive       0.48      0.58      0.52      1380

    accuracy                           0.63      6532
   macro avg       0.54      0.57      0.55      6532
weighted avg       0.64      0.63      0.63      6532
```

Figure 4: Multinomial Naive Bayes evaluation metrics using BoW

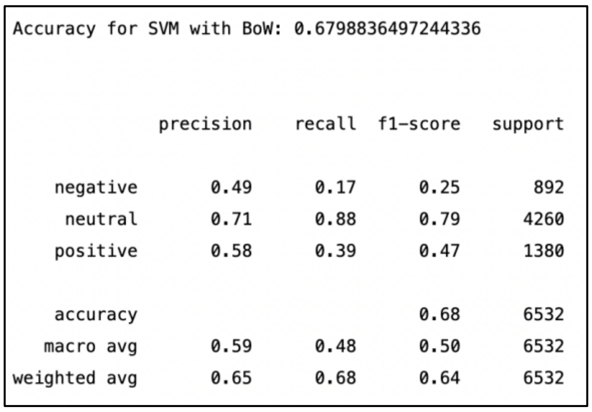


```
MNB accuracy without using cross-validation on TF-IDF:  0.6680955296999388
Mean 10-Fold accuracy using cross-validation on TF-IDF: 0.6063286432622884


              precision    recall  f1-score   support

    negative       0.55      0.03      0.05       892
     neutral       0.67      0.97      0.79      4260
    positive       0.63      0.16      0.26      1380

    accuracy                           0.67      6532
   macro avg       0.62      0.39      0.37      6532
weighted avg       0.65      0.67      0.58      6532
```

Figure 5: Multinomial Naive Bayes evaluation metrics using TF-IDF

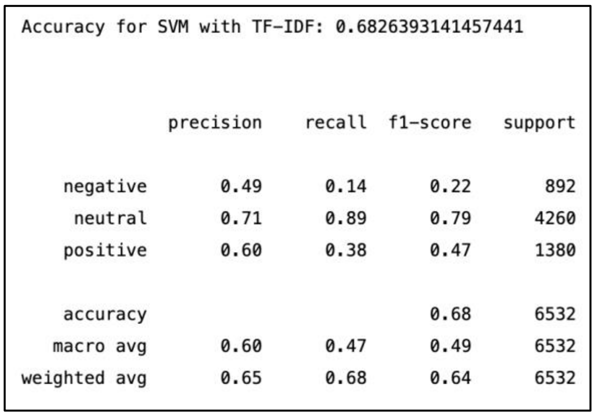
2.2.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) - Support Vector Machines are hyperplanes that separate classes. Although SVM is used mainly for binary classification, we have used SVM for our multiclass classification by using the one-versus-one approach. We did not apply hyperparameter tuning to this section to do the computational cost attached to the tuning of the parameters.



```
Accuracy for SVM with BoW: 0.6798836497244336

              precision    recall  f1-score   support

    negative       0.49      0.17      0.25       892
     neutral       0.71      0.88      0.79      4260
    positive       0.58      0.39      0.47      1380

    accuracy                           0.68      6532
   macro avg       0.59      0.48      0.50      6532
weighted avg       0.65      0.68      0.64      6532
```

Figure 6: SVM evaluation metrics with BoW



```
Accuracy for SVM with TF-IDF: 0.6826393141457441

              precision    recall  f1-score   support

    negative       0.49      0.14      0.22       892
     neutral       0.71      0.89      0.79      4260
    positive       0.60      0.38      0.47      1380

    accuracy                           0.68      6532
   macro avg       0.60      0.47      0.49      6532
weighted avg       0.65      0.68      0.64      6532
```

Figure 7: SVM evaluation metrics with TF-IDF

2.2.4 Random Forests

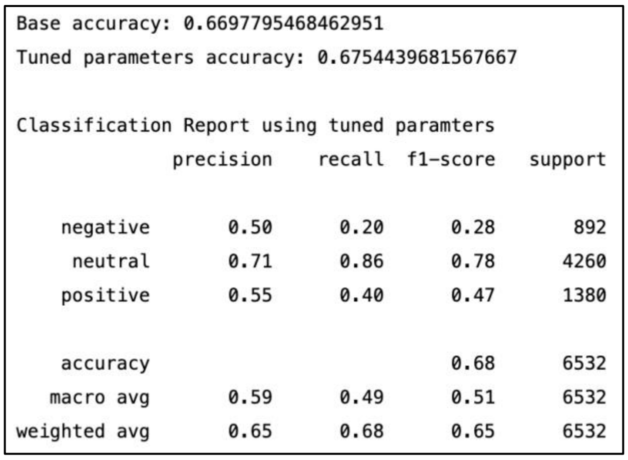
Random Forests - Random forests is an ensemble model which consists of various random trees. We chose this model because it is known to be suitable in dealing with noisy, high dimensionality data (such as in Tweets) in text classification. For tuning hyperparameters, we found that the model performs best with the following hyperparameter settings:

Hyperparameter (for BoW)	Setting/value
n_estimators	466
min_samples_split	5
min_samples_leaf	2
max_features	‘auto’
max_depth	None
bootstrap	True

Table 1: Optimal hyperparameters found for SVM with BoW

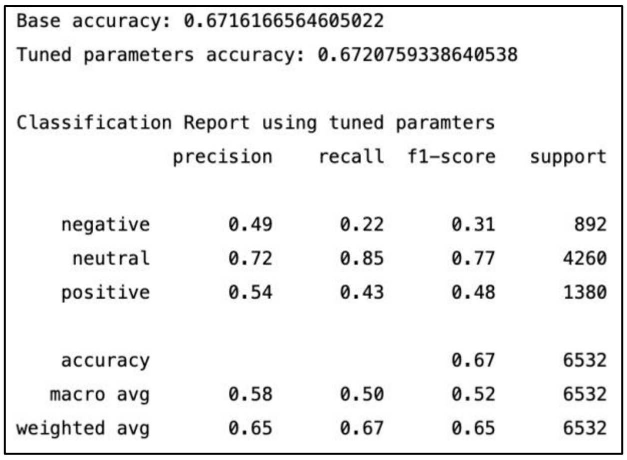
Hyperparameter (for TF-IDF)	Setting/value
n_estimators	1800
min_samples_split	2
min_samples_leaf	2
max_features	‘auto’
max_depth	None
bootstrap	True

Table 2: Optimal hyperparameters found for SVM with TF-IDF



```
Base accuracy: 0.6697795468462951
Tuned parameters accuracy: 0.6754439681567667

Classification Report using tuned paramters
              precision    recall  f1-score   support

    negative       0.50      0.20      0.28       892
     neutral       0.71      0.86      0.78      4260
    positive       0.55      0.40      0.47      1380

    accuracy                           0.68      6532
   macro avg       0.59      0.49      0.51      6532
weighted avg       0.65      0.68      0.65      6532
```

Figure 8: Random Forest evaluation metrics with BoW



```
Base accuracy: 0.6716166564605022
Tuned parameters accuracy: 0.6720759338640538

Classification Report using tuned paramters
              precision    recall  f1-score   support

    negative       0.49      0.22      0.31       892
     neutral       0.72      0.85      0.77      4260
    positive       0.54      0.43      0.48      1380

    accuracy                           0.67      6532
   macro avg       0.58      0.50      0.52      6532
weighted avg       0.65      0.67      0.65      6532
```

Figure 9: Figure 8: Random Forest evaluation metrics with TF-IDF

2.2.5 Ensemble Model

Ensemble model we stacked all three individual models that we explored in the report so far. We ran three iterations, and in each iteration, a different model was the final estimator. We implemented stacking, as according to studies by (Dietterich, 1997) ensemble classifiers have been shown to perform with increased accuracy than standalone ones.

3. Results

We ran all five models with the BoW and TF-IDF vectorizers on the training set (by splitting it further into training and validation sets). We started with the 0R (Baseline) model, to gauge the benchmark accuracy and performance for the rest of our models, and the results are summarized in Table 3:

Vectorizer	Accuracy
BoW	65.2%
TF-IDF	65.2%

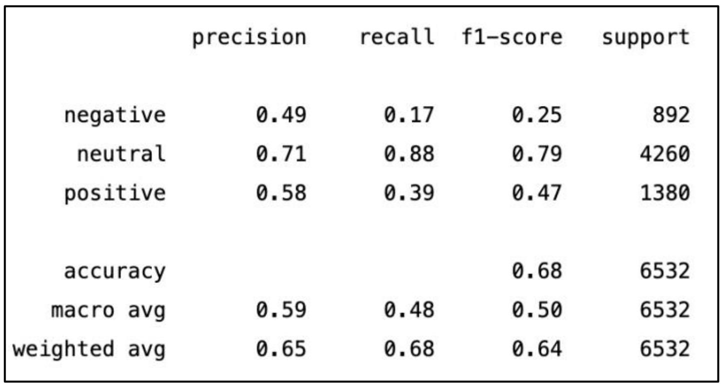
Table 3: Accuracies for the 0R model

For all the other classifiers that we have implemented, we will be measuring their performances using accuracy, as this is the metric that measures the overall performance of the model rather than just specific categories. Table 4 provides a summary of the model performances for the individual models.

Model	Accuracy		Comparison to Baseline
	Bag of Words	TF-IDF	
Multinomial Naive Bayes	62.63% 62.61%, with cross validation (10 folds)	66.81% 60.63%, with cross validation, (10 folds)	Accuracy higher than baseline only with TF-IDF, and no cross validation
Support Vector Machine	67.99%	68.26%	Both accuracies higher than baseline
Random Forest	66.98%, default hyperparameters 67.54%, tuned hyperparameters	67.16%, default hyperparameters, 67.21%, tuned hyperparameters	All accuracies higher than baseline

Table 4: Summary of accuracies for all the individual models

It can be observed that the model with the highest accuracy with the individual models is the Support Vector Machine, with TF-IDF as the vectorizer. We then conducted further analysis of this model's performance through other metrics, which can be seen in *Figure 10* and *Figure 11*.



```
              precision    recall  f1-score   support

    negative       0.49      0.17      0.25       892
     neutral       0.71      0.88      0.79      4260
    positive       0.58      0.39      0.47      1380

    accuracy                           0.68      6532
   macro avg       0.59      0.48      0.50      6532
weighted avg       0.65      0.68      0.64      6532
```

Figure 10: SVM evaluation metrics with TF-IDF

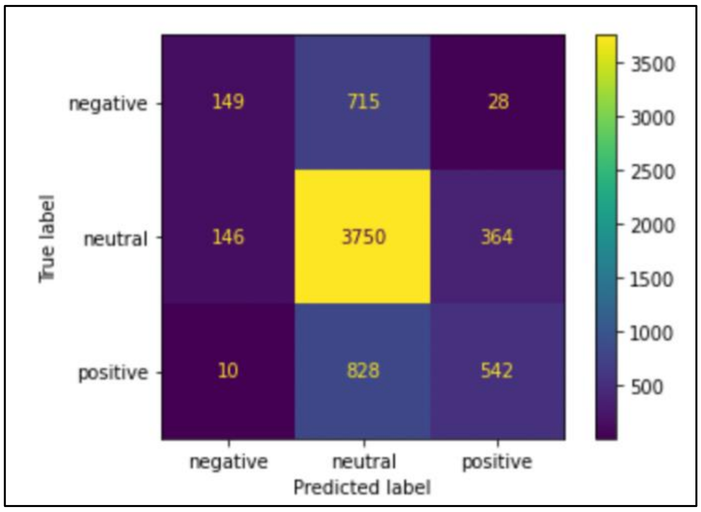


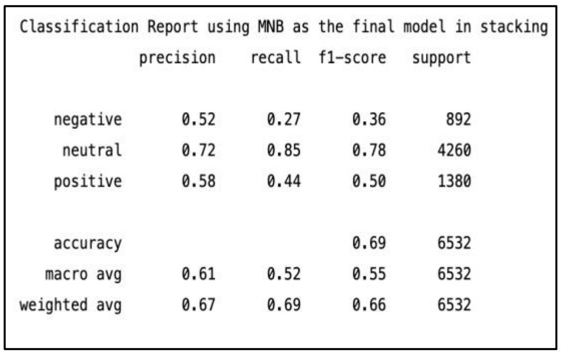
Figure 11: Heatmap for the confusion matrix of SVM with TF-IDF

From both these figures, it is evident that the Support Vector Machine performs best at classifying neutral Tweets, but not as much with the positive and negative Tweets.

After examining performances of the individual models, the performances of various ensemble models (different combinations of the individual models above) we observed. A summary is provided in Table 5.

Final Estimator in Stack	Accuracy		Comparison to Baseline
	Bag of Words	TF-IDF	
Multinomial Naive Bayes	66.88%	68.45%	Both accuracies higher than baseline
Support Vector Machine	65.29%	65.55%	Both accuracies higher than baseline
Random Forest	65.16%	66.23%	Lower than baseline for BoW, higher than baseline for TF-IDF

Table 5: Summary of the accuracies of ensemble models with different classifiers as the final estimator



```
Classification Report using MNB as the final model in stacking
              precision    recall  f1-score   support

    negative       0.52      0.27      0.36       892
     neutral       0.72      0.85      0.78      4260
    positive       0.58      0.44      0.50      1380

    accuracy                           0.69      6532
   macro avg       0.61      0.52      0.55      6532
weighted avg       0.67      0.69      0.66      6532
```

Figure 12: MNB as the final estimator in the ensemble model with TF-IDF

It can be observed that the ensemble model with the Multinomial Naive Bayes classifier, with TF-IDF as the vectorizer is the best performing model, with an accuracy of 68.45%; better performance than the best performing individual model. Carrying out further analysis of the model performances with additional metrics from *Figure n* indicates that there are also higher F1 scores for the ensemble model (except for with the *Neutral* class). As the F1 score is a weighted average of precision and recall, this implies better model performance.

Model	F1 Scores for each class		
	Negative	Neutral	Positive
Individual Support Vector Machine	0.25	0.79	0.47
Ensemble Model, Multinomial Naive Bayes as final estimator	0.36	0.78	0.50

Table 6: Summary of the F1 scores for best individual model and best ensemble model

4. Discussion/Critical Analysis

Based on the results obtained above, and our theoretical results, we have formed two hypotheses that we will test with further experiments.

4.1 **Hypothesis:** Does the accuracy of the machine learning model increase (i.e., it performs better) if the dataset is transformed to a more balanced dataset?

In the dataset section of this paper, it was noted that there is an uneven distribution in the number of instances assigned to each label; the dataset is unbalanced. The instance counts of label - 'neutral' was found to be very high (12659) compared to the other labels (5428 - "positive" and 3715 - "negative").

To test this hypothesis, we used the Multinomial Naive Bayes classifier as it has the lowest computational cost amongst the other implemented classifiers. We implemented two ways to balance the dataset -

1. Random Under sampling - Randomly deleting samples from the majority class
2. Random Oversampling - Randomly duplicating samples from the minority class

This yielded the following results –

Vectorization approach	Balancing method	
	Random Under sampling (accuracy)	Random Oversampling (accuracy)
BoW	48%	54%
TF-IDF	47%	47%

Table 7: Summary of the accuracies for different final estimators in the ensemble models

Our baseline 0R model yielded an accuracy of 65.22%. Comparing the above models against the baseline model, we concluded that for our current dataset, balancing the dataset worsened the model performance. According to (Tan et al., 2020) in under sampling, some of the useful negative examples might not be chosen leading to an inferior classification model. For oversampling, even though the training instances are increased in number, there are not enough unique training instances in the minority classes for the model to train. These implications justify the poor performance of our Multinomial NB classifier on the balanced dataset.

4.2 **Hypothesis** - Does the ensemble model perform better than the individual models? How does the behaviour of the ensemble model change with the change in different final estimator models?

We have used 4 individual models - 0R, Multinomial NB, SVM and Random Forest to perform the sentiment analysis. We then implemented a final ensemble model by stacking the above-mentioned classifiers and got the following results –

Vectorizer	Type of classifier as the final estimator in the ensemble model		
	Multinomial NB(MNB)	SVM	Random Forests
BoW	66.84%	65.16%	65.29%
TF-IDF	68.45%	66.23%	65.55%

Table 8: Summary of the accuracies for different final estimators in the ensemble models

From table no. 8 and table no. 4 we found that our ensemble model with MNB as the final estimator performed better than all the other models. According to (Dietterich, 1997) ensemble models formed by stacking individual models are often found to be more accurate than the individual classifiers. The better performance of such ensemble models can be explained from their ability to learn from the errors of the base classifiers and hence result in a much better accuracy for the final classifier (Arya & Choudhary, 2017). Our findings align with these analyses and hence we can say that for the current dataset for the twitter sentiment analysis, our ensemble model with TF-IDF as the feature vectorizer and Multinomial NB as the final classifier, performs better than all the other models implemented throughout this analysis.

5. Conclusion

The best performing model was the ensemble model with Multinomial Naive Bayes as the final estimator, with TF-IDF as the vectorizer, since stacked models can learn from the errors of base classifiers, which results in an improved accuracy for the ensemble model. Although the dataset was imbalanced, both under sampling and oversampling caused a significant decrease in accuracy, because random under sampling may cause a loss in the number of important instances from the majority class, and oversampling may not be useful as there are not enough unique training instances in the minority class.

Limitations in our research include that we could not implement hyperparameter tuning for the Support Vector Machine classifier, since the code to implement this took too long to run. This is because the training time for SVM is O(n2), where *n* is the number of training instances; grid-searching over all the hyperparameters is not practical. Another limitation was that we could not analyse the effect of emoticons on sentiment as our pre-processing stage eliminated all special characters. A remedy for this would be to convert any emojis to their Unicode representations right before the tokenization stage. Lastly, as our pre-processing stage involved a basic form of word encoding, we could not detect nuances of natural language, such as sarcasm and humour, which play a significant role in conveyed sentiments. A remedy for this would be to train additional supervised models in detecting sarcasm and apply this learning to the sentiment analysis classifiers. These would be questions for future research.

Overall, this investigation has provided practical insights on techniques of sentiment analysis for text data, which can be applied to more advanced natural language processing, such as generative chatbots.

6. References

1. Arya, M., & Choudhary, C. (2017). A Survey on Classification Algorithm for Real Time Data Streams using Ensembled Approach. About Conference
2. Dietterich, T. G. (1997). Machine-Learning Research. In *2371-96210738-4602*
3. Mogotsi, I. C. (2010). Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze: Introduction to information retrieval. *Information Retrieval, 13*(2), 192-195
4. Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2005, 2004-12-04 to 2004-12-06). *Multinomial naive Bayes for text categorization revisited* [Conference Contribution]. AI 2004, Conference held at Cairns, Australia
5. Tan, P.-N. (2020). Introduction to data mining / Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar. In (Second Edition. ed.): Pearson Education.
6. Rosenthal, Sara, Noura Farra, and Preslav Nakov (2017). SemEval-2017 Task 4: sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on semantic evaluation (SemEval '17). Vancouver, Canada.