Tweet Text Sentiment Classification
Group Project

1. Introduction
Over the past decade, the rise of social media has enabled millions of people to share their ideas and respond to current events in real-time. Twitter is an open social networking platform where users can access media through a type of message denoted as tweets.

Each tweet reflects the user's mood at that moment, and it is meaningful to recognize the user's emotion automatically. The usage of such predictions can range from public opinion guidance to precision marketing. The aim of this project is to predict the sentiment presented in the tweet as one of the three sentiments (positive, negative, and neutral). Generally, it is a text classification problem based on Machine Learning. In this research, this topic will be further investigated.

The training data set has 21802 samples, and each sample contains a tweet id, a tweet text, and the corresponding sentiment. Furthermore, the test data set has 6099 samples with a similar format without label sentiments.

2. Method

2.1 Data Pre-processing

2.1.1 Data Splitting

This is a supervised Machine Learning classification question, and the test dataset does not contain labels, so the given training dataset needs to be divided into a training set and a validation set by applying random hold-out, with the validation set ratio of 0.28, which is the percentage of given training data and testing data (6099/21802).

2.1.2 Data Cleaning

Since the raw tweet text contains a lot of unnecessary information, the following four steps are used to clean the data.

a. Remove all numbers, punctuations, special characters, whitespaces, HTTP links, HTML tags, and English stop words (exclude 'not' and 'no' from the stop word lists as they may change the meaning of the sentence), convert all the characters to lowercase

b. Convert accented characters to ASCII characters and expand the contractions (change don't to do not) to standardize text

c. Tokenize the cleaned text

d. Lemmatize and stem the tokens

2.1.3 Text Vectorization

Machine Learning algorithms operate on numeric feature space, and therefore, vectorization needs to be performed on the preprocessed data. The text vectorization algorithms considered are Bag of Words (BoW) and TF-IDF. Bag of Words converts a given text into a feature vector based on the frequency of each word in the text. Compared with BoW, TF-IDF considers the frequency of each word and includes the weight of each word in the dataset.

2.1.4 Feature Selection

After text vectorization, the resulting feature space has an extremely high dimension. BoW and TF-IDF have 127764 dimensions, so irrelevant features are needed to be removed so that it will have a better performance of the model training, as a lot of irrelevant features could increase the training time and the risk of overfitting. In feature selection, the SelectKBest algorithm under the scikit-learn package is applied. Furthermore, this algorithm is tuned with different k values, and the score function used is Chi-Squared.

2.2 Model selection

2.2.1 Zero-R

The baseline model is a simple naïve method that any classifier is expected to outperform. Zero-R was chosen to be the baseline model in this project since it is simple, it classifies all instances according to the most common class in the training data.

2.2.2 Logistic Regression (LR)

Logistic Regression uses the logistic function to estimate the probability of the dependent variable and given features. In this research, Multinomial logistic regression is used to predict multiclass outcomes. Logistic Regression is easy to interpret and can tell some information about features, such as whether the correlation between features and classes is positive or negative, and it is very efficient for handling large datasets, making Logistic Regression suitable for this classification task.

2.2.3 Support Vector Machine (SVM)

Support Vector Machine is a commonly used classifier that maximizes the margin separating the examples from the hyperplane. In this research, multi-class SVM will be utilized. Since SVM has better robustness and the ability to analyze many features, it is suitable for this problem.

2.2.4 Random Forest

The Random Forest classifier is an ensemble of random trees, and each tree is constructed using a different bagged dataset, in which each node uses only some attributes. Combining the results of multiple decision trees helps reduce the tendency of decision trees to overfit the training set. It also inherits the advantage of Bootstrap that it is parallelizable.

2.2.5 Stacking

Stacking combines the output of several base classifiers as input to a further supervised learning model. The implementation of the Stacked Ensemble learner in this research combines Logistic Regression and Random Forest as the base classifiers, then uses the Logistic Regression as the meta classifier. The stacking can usually make better predictions than any single contributing model and can decrease the dispersion of predictions and model performance.

2.3 Evaluation Metrics

2.3.1 Confusion Matrix

The confusion matrix helps visualize the performance of prediction results on the classification problem. It clearly shows the errors in a multiclass classification task, and the number of TP, TF, FP, and FN can be easily interpreted.

2.3.2 Weighted Average

Take the weighted average of the precision/ recall / F1 scores of each class, where the weight is the portion of instances of each class in the total instances. It is more reliable for uneven data distribution.

2.3.3 Time complexity

The time spent on model training reflects the model's efficiency, which is a crucial factor for comparing model performance.

2.3.4 Score function

This research presents prediction and validation accuracy separately by the score () and cross_val_score () function.

3. Results

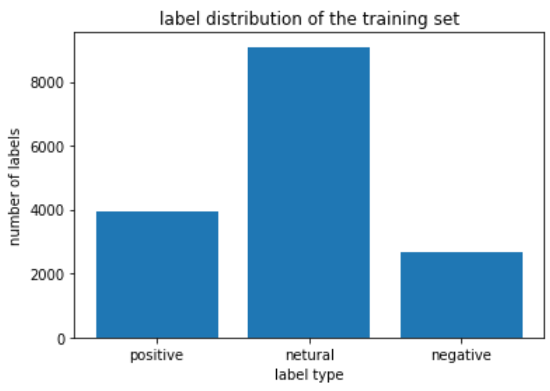


Figure 1 – Label Distribution

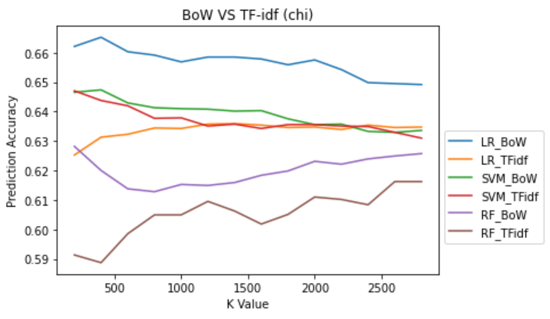


Figure 2 - Accuracy rate for BoW and TF-IDF text vectorizer on different models

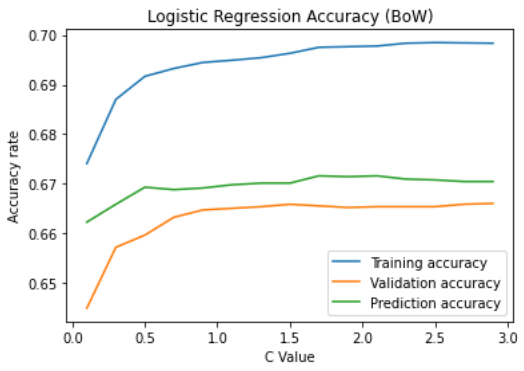


Figure 3 - Accuracy rate for Logistic Regression on different C values

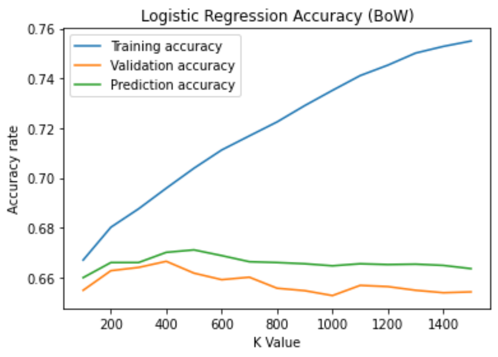


Figure 4 - Accuracy rate for Logistic Regression on different k value

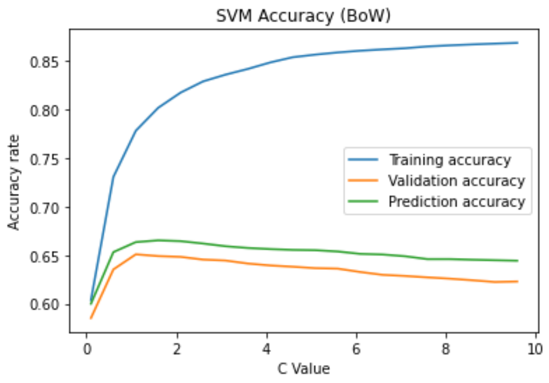


Figure 5 - Accuracy rate for SVM on different C values

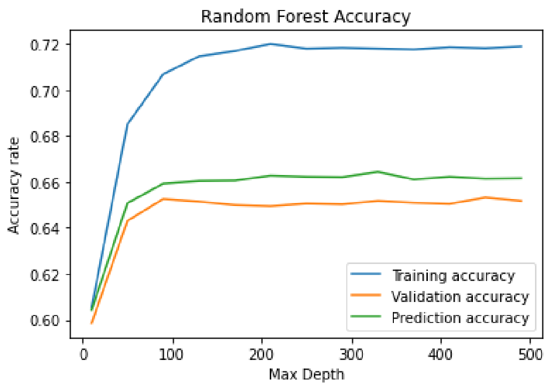


Figure 6- Accuracy rate for Random Forest on different max_depth

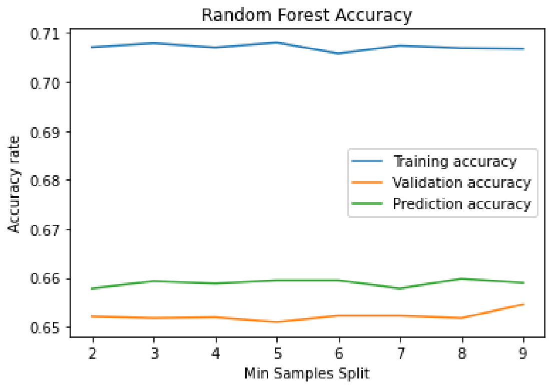


Figure 7- Accuracy rate for Random Forest on different min_sample_split

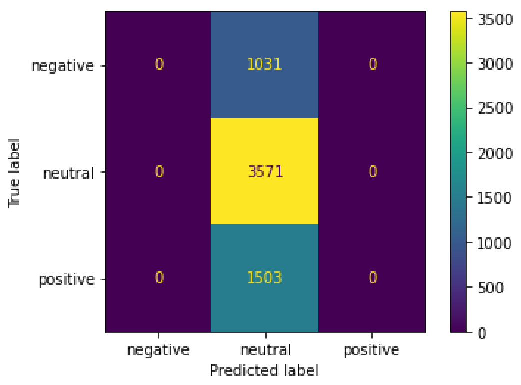


Figure 8 – Confusion matrix for Zero R

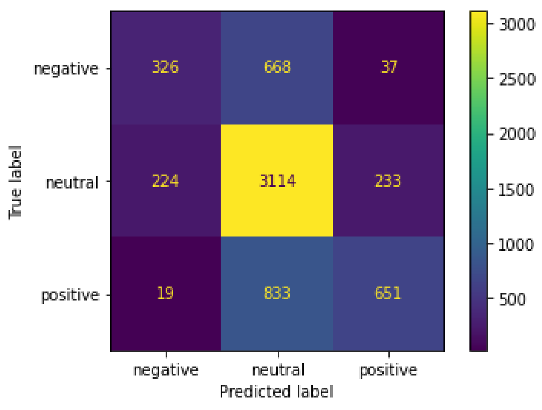


Figure 9 – Confusion matrix for Logistic Regression

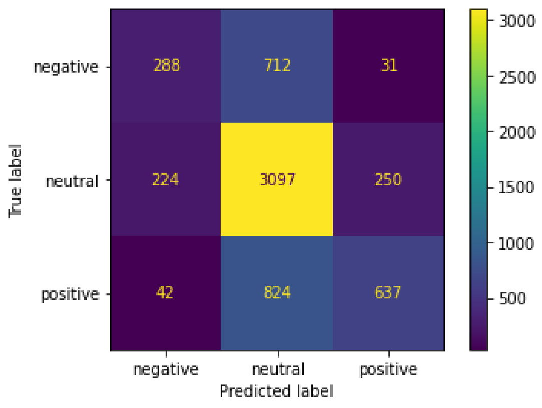


Figure 10 – Confusion matrix for SVM

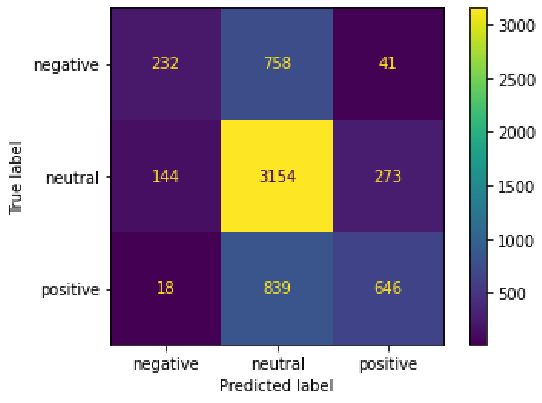


Figure 11 – Confusion matrix for Random Forest

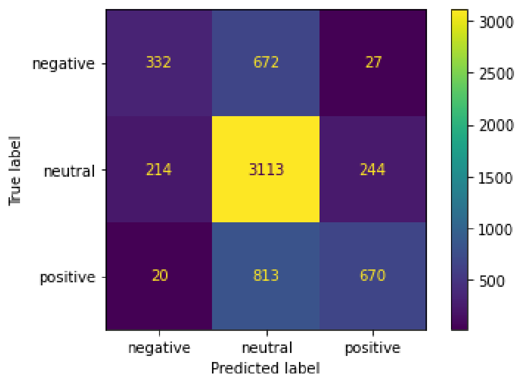


Figure 12 – Confusion matrix for Stacking

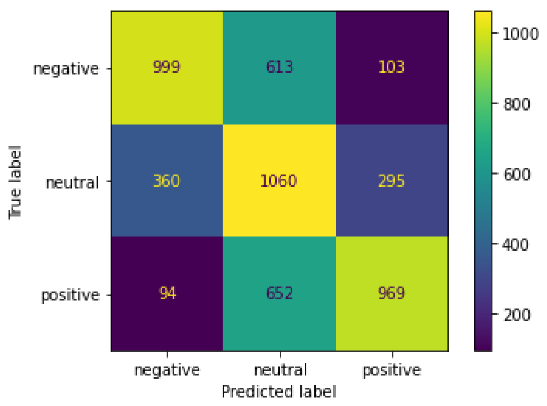
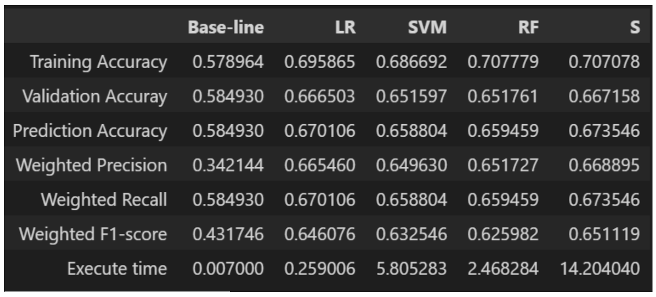


Figure 13– Confusion matrix for SVM with even dataset



	Base-line	LR	SVM	RF	S
Training Accuracy	0.578964	0.695865	0.686692	0.707779	0.707078
Validation Accuray	0.584930	0.666503	0.651597	0.651761	0.667158
Prediction Accuracy	0.584930	0.670106	0.658804	0.659459	0.673546
Weighted Precision	0.342144	0.665460	0.649630	0.651727	0.668895
Weighted Recall	0.584930	0.670106	0.658804	0.659459	0.673546
Weighted F1-score	0.431746	0.646076	0.632546	0.625982	0.651119
Execute time	0.007000	0.259006	5.805283	2.468284	14.204040

Table 1 – Final Result

4. Discussion/Critical Analysis

4.1 Feature Selection

During the data cleaning process, it is found that removing the stop words improves the performance of the models. In this research, the validation accuracy of Logistic Regression and SVM grows from 0.659 and 0.648 to 0.667 and 0.652, respectively. After reducing the noise by removing stop words, it is easier for the model to learn correct and generalized patterns.

By experimenting with different ngram_range values in text vectorization ((1,1), (1,2), (2,2)), the model has the highest validation accuracy when ngram_range is equal to (1,2). Therefore, both bigrams and unigrams should be included as features since only taking each word as a feature might not be very generalizable. For instance, if "love" represents positive and "hate" represents negative, the model may not be able to correctly identify the sentiment if "not love" or "not hate" appears in a tweet. By adjusting ngram_range, more combinations of words can be obtained, which contains more information, improving the generalization ability and performance of the model.

Figure 2 shows the performance of each classifier when using different text vectorizers. Logistic Regression and Random Forest perform significantly better on BoW than TF-IDF. However, the performance of BoW and TF-IDF vectorizer on SVM is similar. TF-IDF has the advantage of quantifying the importance of words in the dataset. With the informal content of Twitter, the advantages of TF-IDF may not be brought into play. In addition, when k increases, the feature dimension will increase. However, the validation accuracy for most models will decrease, which is generally caused by model overfitting, which can be seen in Figure 4. However, for Random Forest, the k value may not be a sensitive hyperparameter since the validation accuracy is not affected. Moreover, the higher the feature dimension, the longer the model training time, which shows that feature dimension reduction is meaningful.

Therefore, BoW is chosen as the text vectorizer because it has better performance for most models. By observing Figure 2, the k value will be 400 as most models have a better performance at 400.

4.2 Logistic Regression

The grid search method is first applied to find the optimal values of the Logistic Regression hyperparameters. The optimal parameter combination is found to be solver = ‘saga’, C = 2.1, multi_class = “auto”, with validation accuracy of 0.6654. Figure 3 shows how the accuracy of the logistic regression model varies with different values of C when solver = "saga". When C increases, training accuracy, prediction accuracy, and validation accuracy all increase, proving that tuning C can indeed improve the performance of the logistic regression model. When the value of C is greater than 1, the validation accuracy tends to remain unchanged. However, with the increase of C, the risk of model overfitting will also increase because it can be seen from the figure that the gap between training accuracy and validation accuracy will gradually increase. By tuning the C value many times, when C = 1.4, the validation accuracy of the model is improved to 0.6665. Although the model's prediction accuracy is higher at C = 2.1, C is ultimately chosen to be 1.4 because the validation accuracy is more reliable.

4.3 Support Vector Machine

Support Vector Machines are highly effective in text classification, whose nature is a method of placing text into meaningful groups (Khairnar & Kinikar, 2013). However, in high-dimensional cases, it is difficult to see the decision boundary intuitively.

The hyperparameter C is the regularization parameter. Figure 5 shows the change in model accuracy for different values of C, while other hyperparameters are set to default values. All three accuracies increase with the increase of C, but when C exceeds 1, the difference between training accuracy and validation accuracy becomes significant. This might be because fewer errors are permitted when the penalty for misclassification increases, so training accuracy increases with C. However, when C is too large, the model tends to have “perfect performance” on the training dataset, which means an increased risk of overfitting. After applying the grid search function and hyperparameter tuning for SVM, the best combination of hyperparameters is kernel = ‘linear’, C = 0.4, decision_function_shape = ‘ovo’.

SVM uses a kernel trick technique, where the kernel maps low-dimensional data into higher-dimensional data to make it linearly separable. By running the model with different kernels under the same input, it is found that SVM with the linear kernel is faster than those with non-linear kernels. The execution time of the linear kernel is 5.772s, while the “rbf” and “poly” kernels both take about 8.2s, which is slightly longer than that of the linear kernel because the algorithm of the linear kernel is simpler than the others.

4.4 Random Forest

According to Shah et al. (2020), Random Forest is an ensemble method whose objective is predictive models for classification and regression problems. It has been proved as an effective classifier in many cases. The advantage of random forest is that it does not depend on feature importance given by a single decision tree, but on the decision tree to vote for the most popular class and then output the predicted value. However, the decision logic of the random forest is not as intuitive as the decision tree.

After running the grid search, the hyperparameter combination is found to be max_depth = 70, min_samples_leaf = 2, n_estimators = 250, and min_samples_split = 3. By observing Figure 6, the validation accuracy stops increasing after max_depth reaches 90, so it will be set as the optimal value for max_depth since deeper trees require longer execution time and a higher risk of overfitting. Also, if the maximum depth of the tree is too small, the model is at risk of underfitting. The hyperparameters min_samples_split and min_samples_leaf also control the depth of the tree. The larger these two hyperparameters are, the harder it is to divide the tree downward, the shallower the depth of the tree, and the model is more prone to underfitting. Conversely, the larger these two hyperparameters are, the model is more prone to overfitting. From Figure 7, the changes in min_samples_split do not significantly improve the validation accuracy, so the default value is used. Therefore, the final settings for the hyperparameter of Random Forest are max_depth = 90, min_samples_leaf = 2, n_estimators = 250.

4.5 Stacking

To build an effective ensemble model, the diversity of base models plays a significant role. According to Güneş et al. (2017), an ensemble model achieves higher accuracy and robustness than a single model with fine-tuning by combining information from different modeling approaches. By experimenting with different base classifiers, the combination of logistic regression and random forest provides the highest validation accuracy.

As shown in Table 1, the stacking classifier has the longest training time and the highest validation accuracy compared with other classifiers. However, logistic regression provides similar validation accuracy with less training time in the study, which means that stacking may have relatively poor efficiency. Furthermore, increasing the number of base classifiers is not strictly related to the performance of stacking. This study tests many different combinations of base classifiers and finds that stacking with three base classifiers sometimes has worse validation accuracy than stacking with two base classifiers.

4.6 Error Analysis

Confusion matrices are used as an error analysis method. Looking at the confusion matrices in Figures 8 to 12, the FN for negatively labeled and positively labeled instances is abnormally high. After examining the data distribution in the training set (Figure 1), the number of neutral labels is almost twice the number of positive and negative labels in the training set. This phenomenon suggests that the non-uniform distribution of samples may affect the model training process and text sentiment prediction. The model is more likely to predict that the sentiment of a tweet is neutral. Figure 13 shows the confusion matrix of SVM trained with balanced label training data. The number of FN for negatively labeled data and positive labeled data becomes more balanced, suggesting that the distribution of the predicted labels is related to the distribution of the samples. However, the validation accuracy of the balanced labeled training data set drops, which will be explored further in the future.

In the evaluation process, cross-validation is used to evaluate the model performance on new data to reduce overfitting to a certain extent. It can get more effective information from limited data, reducing variance and making the measured accuracy more reliable. Furthermore, all models except Zero-R have similar weighted precision and weighted recall, indicating the reliability of these models.

5. Conclusion

In this report, Sci-kit learn library is used to perform sentiment classification on tweets from a large dataset. It is found that after pre-processing, Bag of Words vectorization, and feature selection using chi-square scoring, all the built models outperformed Zero-R. The validation accuracy indicates that the stacking classifier outperforms all the other classifiers. Furthermore, SVM provides relatively worse validation accuracy, and this might be because SVM is less effective with a large dataset. Therefore, SVM may not be suitable for this problem.

Since the training/validation set is an example of a 72:28 holdout strategy, future considerations include performing k-fold cross-validation on the training set to produce more convincing results. In the future, more classifiers such as deep learning models should be implemented to explore their performance.

Reference

Güneş, F., Wolfinger, R., & Tan, P.-Y. (2017, April). Stacked Ensemble Models for Improved Prediction Accuracy. In Proc. Static Anal. Symp. (pp. 1-19).

Khairnar, J., & Kinikar, M. (2013). Machine learning algorithms for opinion mining and sentiment classification. International Journal of Scientific and Research Publications, 3(6), 1-6.

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.

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. Augmented Human Research, 5(1), 1-16.