COMP30027 Report

Anonymous

1. Introduction

Nowadays, people are getting busier in their life. We all want to use our time as much efficiently as possible. The same thing goes with cooking. Most people want to cook delicious food in the shortest time. Therefore, the problem to categorize the duration of cooking time based on the recipe is critical. This project aims to develop classifiers that can indicate the cooking time base on the recipe feature: name, ingredients, and steps and evaluate these models to determine the best model for this problem.

The data used for this project is collected from Food.com (Majumder, 2019). They are raw text data so the focused methodology will be natural language processing and text classification.

2. Feature Engineering

2.1 Data pre-processing and Feature Extraction

In this project, I will make use of 2 data pre-processing techniques which are already given by the staff of the COMP30027 teaching team and I will also combine these techniques as a new one. Then, I will experiment with different classifiers to find the one that gives the best performance.

2.1.1 Count Vectorizer

The count vectorizer method also knows as the one-hot encoding is a technique to forms a corpus that contains a range of identified tokens from the data. It converts text documents into a "Bag of Words" that uses these tokens as the attributes and records their occurrences while ignoring any useful information about the position of the words. This method is considered a good technique when dealing with a text classification problem because it keeps most of the data features and makes each instance identical to the other.

2.1.2 Doc2vec

The main idea of this technique is to convert words to a high-dimensional vector space so that words which has similar meaning will be close together in that vector space. To do it, the technique "concatenate the paragraph vector with several word vectors from a paragraph and predict the following word in the given context" (Le, Mikolov 2014). Then, the paragraph vectors and word vectors will be trained by stochastic gradient descent and backpropagation. This technique is considered superior to count vectorizer as it keeps the ordering of the words and it does not ignore the meaning of the words.

2.1.3 Combine Count Vectorizer and Doc2Vec.

I will merge the data from the count vectorizer and the data from doc2vec into a sparse matrix using the stacking technique. There is no guarantee that this method will make the data better for prediction so I will use it cooperative with other datasets to find the best method.

2.2 Feature Selection

The mutation information (MI) will be used as the score function for feature selection. The MI calculates how much knowledge of an instance reduces the uncertainty on the other. Despite other methods like the Pearson correlation, the MI is good to deal with non–linear dependences.

For this project, I will use the MI with the top 50% features. However, I cannot guarantee it can eliminate the irrelevant attributes. Therefore, I will compare the prediction with and without feature selection.

3. Single Model

3.1 Linear SVC

The Support Vector Machine (SVM) is a non-probabilistic for classifying text data into suitable categories. It is a rational choice if we don't have much knowledge of the data. Especially, with the kernel trick, SVC can process complex data. It also avoids over-fitting by tunning appropriate hyperparameter. However, it is also difficult to find suitable hyperparameters because the

SVM is hard to visualize and interpret the final model. Table 1 below determines that Linear SVM provides fair accuracy with the default hyperparameter. The Linear SVN with feature selection has a little higher accuracy than the one without feature selection. While in all cases, the steps feature to provide the highest accuracy which is 77% in the combined dataset. If the input data is Doc2Vec, Linear SVC produces lower accuracy than the other two may be since SVC works better under high dimensional space. When we look at the confusion matrix in Figure 1, we can see that the prediction for the class label '3' has the smallest accuracy. It is aligned with the fact that SVM will underperform if the number of features bigger than the number of training data points Overall, the SVM can give good performance if an appropriate hyper-parameter is chosen.

Input	Feature	Accuracy	
		With FS	W/o FS
CountVec	Name	0.69	0.69
	Steps	0.77	0.76
	Ingr	0.68	0.68
Doc2Vec	Name	0.57	0.59
	Steps	0.63	0.65
	Ingr	0.57	0.58
Combined	Name	0.70	0.69
	Steps	0.77	0.76
	Ingr	0.69	0.69

Table 1: Accuracy Table for Linear SVM Classifier

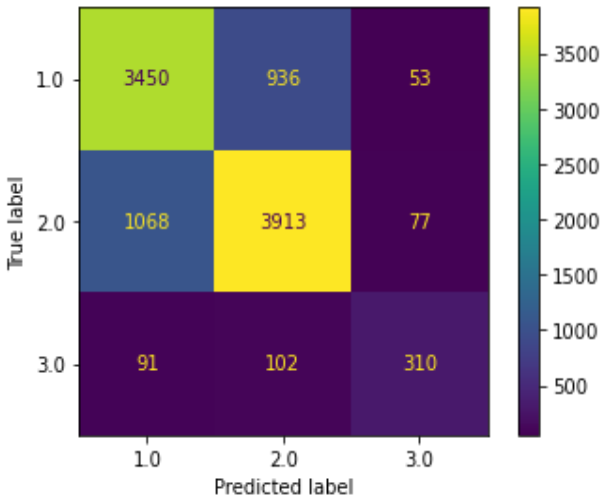

Figure 1: Confusion Matrix for Linear SVM Classifier

3.2 Logistic Regression

Logistic regression is a regression method to classify and handle categorical data. This method predicts the class label based on the observations of the training dataset. It is very efficient if the features are linearly separable. This is one of the simplest models in machine learning and final results are also easy to interpret. As Table 2 shows, logistic regression performs well, slightly better than the Linear SVM model. The errors from the confusion matrix in Figure 2 indicate that it may affect by the non-linear attribute that can not be resolved. In conclusion, logistic regression still gives acceptable accuracy.

Input	Feature	Accuracy	
		With FS	W/o FS
CountVec	Name	0.70	0.70
	Steps	0.78	0.78
	Ingr	0.69	0.68
Doc2Vec	Name	0.57	0.59
	Steps	0.63	0.65
	Ingr	0.57	0.58
Combined	Name	0.71	0.71
	Steps	0.78	0.78
	Ingr	0.69	0.69

Table 2: Accuracy Table for Logistic Regression Classifier

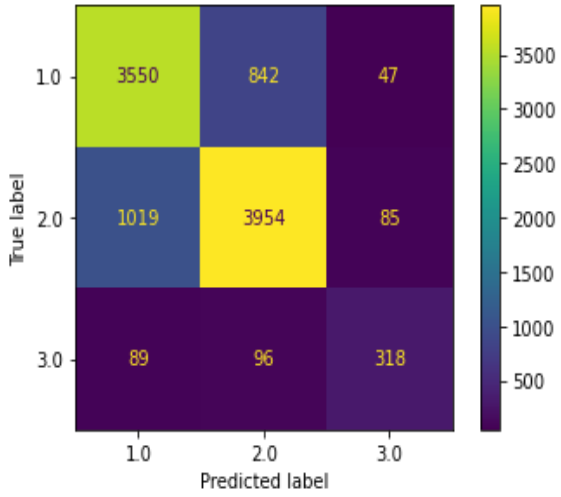

Figure 2: Confusion Matrix for Logistic Regression Classifier

4 Ensemble Model

Ensemble learning is a process to use multiple machine learning models together to reduce errors.

4.1 Extreme Gradient Boosting (XGBoost)

Boosting is a sequential ensemble learning technique that aims to reduce bias and increase the accuracy of the model.

Gradient boosting is also boosting but makes use of a gradient descent algorithm to minimize the error.

However, boosting try to fit with the lowest bias so it can cause the over-fitting problem.

XGboost can solve these weaknesses while implementing "decision trees with boosted gradient, enhanced performance, and speed"(Kumar 2019).

One of the techniques XGBoost use is tree pruning. By replacing the nodes that do not improve the accuracy, it can reduce the size of regression trees. Therefore, it can go deeper depth to obtain more data.

With all of this enhancement, XGBoost produces the best accuracy compared to other classifications in this project. The accuracy for combined data of steps is 81% which is an impressive result.

The confusion matrix in Figure 3 also shows low error rates across all class labels.

Input	Feature	Accuracy	
		With FS	W/o FS
CountVec	Name	0.70	0.69
	Steps	0.80	0.80
	Ingr	0.70	0.70
Doc2Vec	Name	0.57	0.6
	Steps	0.67	0.68
	Ingr	0.59	0.61
Combined	Name	0.69	0.68
	Steps	0.81	0.81
	Ingr	0.69	0.69

Table 3: Accuracy Table for XGBoost Classifier

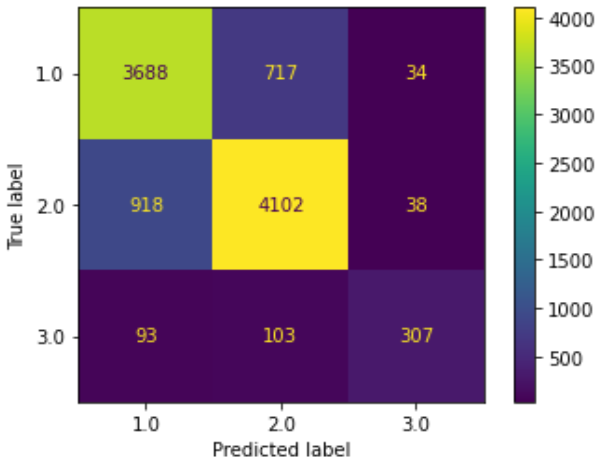

Figure 3: Confusion Matrix for XGBoost Classifier

5. Conclusions

In conclusion, among all classifiers experimented above, an Extreme Gradient Boosting gives the best performance. XGBoost is considered one of the most winning algorithms in Kaggle due to its system optimization and algorithmic enhancements. It also works really with the raw text training data provided in this project. The accuracy is 81.066% when I submit it to Kaggle indicates that it can work well under unseen data.

Another problem that can be is that the feature selection just has a small impact on the accuracy. Thus, feature selection seem not to drop many irrelevant attributes.

Moreover, the doc2vec50 dataset always gives the lowest accuracy while the Count Vectorize and combined dataset give similar results. It may be because the doc2vec50 only record 50 feature while the Count Vectorize record all feature in the training data. Therefore, when we merge the doc2vec data to Count Vectorize data it does not contribute much useful information toward the final prediction.

6. References

Majumder, B. P., Li, S., Ni, J. & McAuley, J. Generating personalized recipes from historical user preferences. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019.

Statinfer. 2016. 204.6.8 SVM: Advantages disadvantages and applications.

Harode, R. 2020. XGBoost: A Deep Dive into Boosting, Medium, Viewed 21st May 2021. <https://medium.com/sfu-cspmp/xgboost-a-deep-dive-into-boosting-f06c9c41349>

Quoc Le & Tomas Mikolov.2014. Distributed Representations of Sentences and Documents.

Beraha, M. Metelli, A. M. Papini, M. Tirinzoni, A. Restelli, M. 2019. Feature Selection via Mutual Information: New Theoretical Insights

Khanday, A. M. U. D, Rabani, S. T., Khan Q. R., Rouf, N., Di, M. M. U. 2020. Machine learning-based approaches for detecting COVID-19 using clinical text data

BANSAL SHIVAM. 2018. A comprehensive guide to understand and implement text classification in python.

Dhiraj, K. 2019. Top 4 advantages and disadvantages of Support Vector Machine or SVM, Medium. Viewed 21st May 2021, <https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107>

Kumar, N. 2019. Advantages and Disadvantages of Logistic Regression in Machine Learning, The Professional Point. Viewed 21st May 2020. <http://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of.html >