COMP30027 Assignment 2 Report

Anonymous

1. Introduction

Nowadays, with the widespread popularity of the internet and smartphones. People don't have to be a trained chef to make delicious food. Food.com is one of those websites that allow people to find and share their delightful cooking recipes and mouthwatering cooking ideas.

This project aims to implement supervised text classification models to predict the cooking time of a set of recipes listed on food.com. There are three possible levels of cooking duration, 1, 2, or 3, of a given recipe. Such predictions can be helpful, from helping people make informed choices when looking for cooking recipes to assisting websites in predicting the cooking time of unlabelled recipes through an automated process.

2. Feature Engineering

Giving the CSV file provided contains raw text as features; therefore we will need to convert raw text into helpful information so that we can feed the data to train our model. Although the processed data are already provided, I decides to implement some other practical text cleansing techniques, such as lemmatization, to ensure there is no distortion introduced to the model.

2.1 Text Cleansing and Preprocessing

1. Punctuations are removed from each text because they provide no support during training and prediction.
2. Then use Lemmatizer from WordNet library converts each word to its lemma. For example, *writes*, *writing* and *wrote* are all forms of the word *write;* therefore, lemmatization will convert all these words into *write*, which can reduce size of the dataset.

2.2 Train Test Split

Since we have a large training set containing 40,000 instances, I have chosen to randomly split our data into 80% instances composing the training set and 20% instances for the testing set, and use the test set to evaluate the quality of model when predicting unseen data.

2.3 Text Representation

I decided to use *steps* as the training feature, given that *name* will probably not provide much information about the cooking duration, and most of the *ingredients* are already included in *steps*.

Since machine learning model operates on a numeric feature space. Therefore, I choose to apply the CountVectorizer and TfidfVectorizer technique, which count the frequency of each word in the recipe and store the frequency as a sparse matrix, where each word becomes a feature. Whilst TF-IDF also downscales words that are too common.

These techniques also remove stop words like *the*, *and*, *is*, *in*, etc., that does not help the model to study the underlying relationships between attributes and class labels, and might increase the complexity of calculation.

3. Predictive Models

Here, I choose to implement the Multinomial Naive Bayes as our baseline classifier. And implement some other classifier such as Multi-class Logistic Regression, Support Vector Machine, and Stacking. See if these models could increase the prediction accuracy.

3.1 Multinomial Naive Bayes

A probabilistic classifier, which is suitable for classification with discrete feature since our dataset is an sparse matrix. It assume that every feature is independent from the others. Then, calculate the probability of each class using Bayes Theorem for each recipe, and the outcome will be the category with the highest probability.

3.2 Logistic Regression

Given that Logistic Regression classifier is used to solve binary class classification, Here, I choose to implemented Multi-class Logistic Regression, a generalised version that solves multi-class classification. Also use grid search to tune the hyperparameter *C (the inverse of regularization strength)*, *solver*, and *penalty*.

3.3 Support Vector Machine

SVM finds an optimal solution that maximises the margin between the hyperplane and the difficult points close to the threshold. Given that we have a large dataset, I decided to implement *Linear SVC*, which converges faster on a large dataset than *SVC*. Multi-class classification is achieved by using the one-versus-all scheme. And use validation curve to find an optimal value for penalty parameter *C*.

3.4 Stacking

Finally, I will implement a stacking classifier and use Logistic Regression as the meta-classifier. Use the output of the above three model and the estimation of Random Forest as the input of the meta-classifier. I hope this could derive a variety of variances and biases from the base classifiers, and improve the prediction performance.

3.5 Model Performance with Different Vectorization Method

Before feeding the data to fit the model, I performed feature selection using mutual information to reduce the dimension of the dataset to 1000 attributes, which removes redundant features to avoid overfitting and improve training time.

Multinomial Naive Bayes performed better with frequency count than TF-IDF, which is probably due to the fact that attribute values in the frequency count sparse matrix follows the multinomial distribution. On the other hand, Logistic Regression and SVM performs better with TF-IDF.

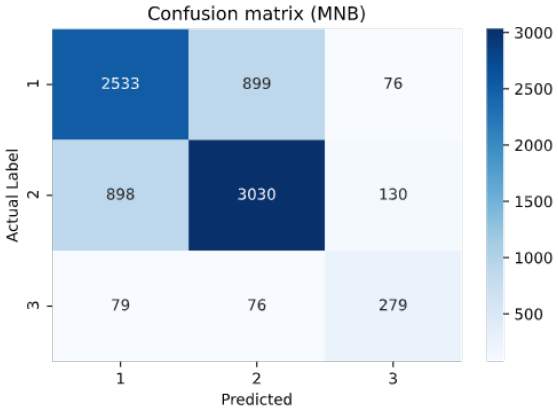
4. Error Analysis

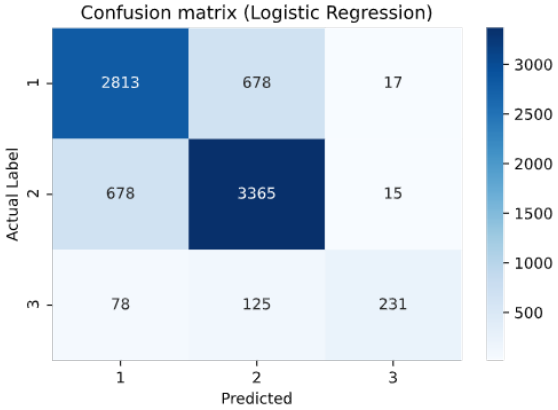
The distribution of each class are shown in the table below, We can see that the dataset are imbalanced, number of class 3 samples are significantly less than the other two classes, which only account for approximately 5% of the overall population.

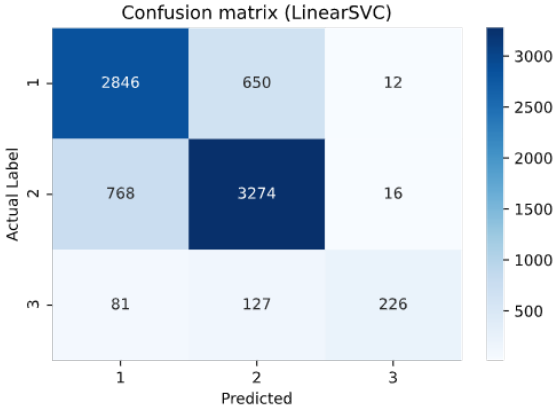
	Class 1	Class 2	Class 3
Number of instances	17705	20246	2049

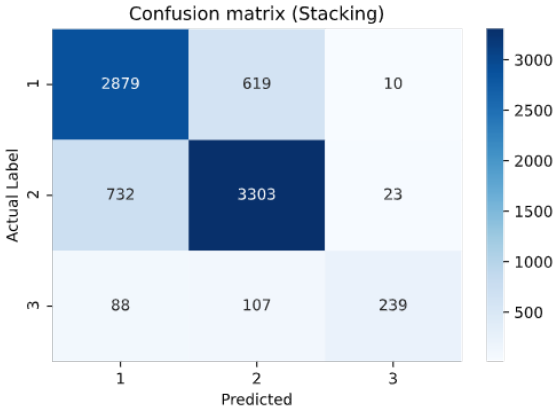
Table 1- Instances Distribution

And confusion matrix are computed for each model to further interpret their behaviour and pattern of errors.









As we can see from these confusion matrices, the vast majority of predictions ended up on the diagonal. All four models have done a decent job when classifying class 1 and 2, but have a low recall value when classifying class 3. A significant portion of class 3 instances are biased towards the majority class, which might be because we do not have sufficient training samples for class 3 due to class imbalance. Logistic Regression, Linear SVC and Stacking manage to obtain high precision and low recall while classifying class 3, which indicates these models are very picky; it requires much evidence to predict class 3 instances correctly. On the other hand, Multinomial Naive Bayes obtained very low precision and recall when classifying class 3.

5. Final Result and Evaluation

	Training Set Accuracy	Test Set Accuracy	Stratified 5-fold CV Mean
Multinomial Naive Bayes	0.7508	0.7303	0.7258
Logistic Regression	0.8105	0.7949	0.7925
Linear SVM	0.8117	0.7932	0.7900
Stacking	0.8238	0.8028	0.7988

Table 2- Model Performance Table

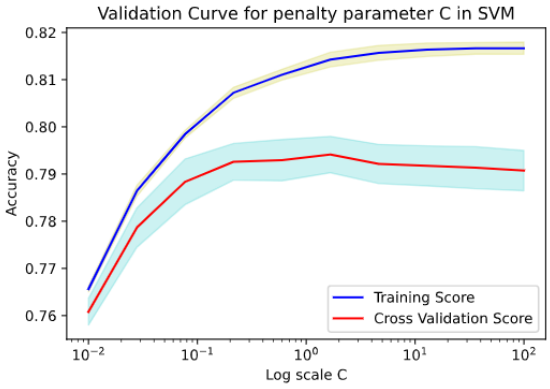
As we can see from the table above, Logistic Regression and Support Vector Machine did a better job than Multinomial Naive Bayes, whilst Stacking ensemble classifier manage to achieve the highest accuracy. Since we have an imbalanced dataset, therefore using random sampling K-fold cross-validation, the training data may have different distributions with the overall population. Here, I choose to use Stratified K-fold cross-validation to evaluate model variance, which preserves the class distribution in the train and test sets for each fold. All four models have low variance as Stratified 5-fold cross-validation produce a similar prediction score with different training folds.

Fold	1	2	3	4	5
MNB	0.7175	0.7350	0.7267	0.7228	0.7270
LR	0.7873	0.7931	0.7905	0.7947	0.7967
Linear SVC	0.7848	0.7919	0.7845	0.7907	0.7981
Stacking	0.7921	0.7998	0.7975	0.7990	0.8059

Table 3- Stratified 5-Fold CV Score

Using grid search, I was able to tune the hyperparameter for the Logistic Regression, which returns an optimal value for C = 2, penalty for *ridge regression*, *newton-cg* for solver. However, this only improves the performance slightly.

For linear SVC, I computed a validation curve over the penalty parameter *C*. It is a clear sign that with a higher *C* value, the model starts to lose its ability to generalise well to predict the correct label for unseen instances, which allows less error and causes overfitting. Hence we choose an optimal value of *C* somewhere the curves do not diverge too much, and it finalised at around 1.668, which returns the best model.



6. Conclusions

In this assignment, several classic machine learning models, such as Naive Bayes, Logistic Regression, Support Vector Machine, and Stacking, are implemented from the sci-kit learn library to perform text classification. The built model could predict the cooking duration of a given recipe. Additionally, the models are evaluated using several evaluation metrics and shown the performance of the models in terms of their prediction capability.

The class distribution in the given dataset is quite imbalanced. But due to time limitations, I was unable to apply some oversampling techniques, which involves increasing the size of the training dataset with multiple copies of the minority class, to resolve class imbalance.

7. 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.