Project 2: How long will it take to cook this?

1 Introduction

In this project, we aimed to predict the cooking time for recipes based on their name, steps, ingredients and other features, by using various supervised Machine Learning methods. We will utilise some machine learning software and existing libraries (like sklearn) to build the classifiers such as Naïve Bayes, logistic regression, linear SVC and then implement ensemble modelling based on these models.

The rest of this report is organised as follows. Section 2 describes our approach to processing the datasets as well as building the classifiers. In section 3 we discussed about the outcome of different methods we used and their behaviours. Later, in Section 4 we recap the results highlight a few possibilities for future work. Finally, Section 5 states the references.

2 Approach

2.1 The Datasets

There are training and testing datasets provided (from Food.com) in this project. The datasets contain 6 feature columns, where the training data have 40000 instances, and the test data have 10000 instances. There are 3 labels that represents the cooking time: 1.0, 2.0 and 3.0 with corresponding to quick, medium and slow.
The features are name, n steps, n ingredients, steps, ingredients, duration label.

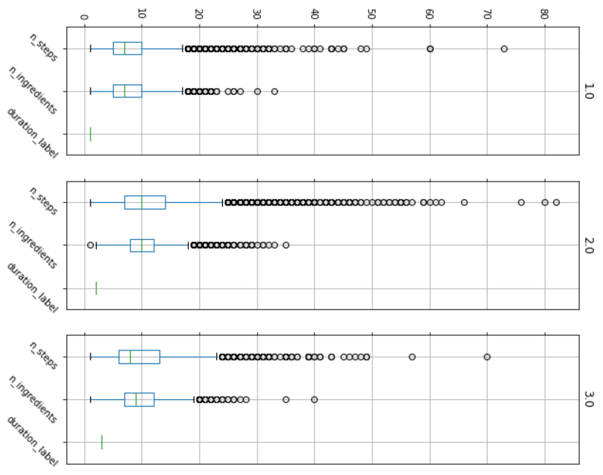
2.1.1 Instance Construction

For the feature with string values such as name, steps and ingredients, in order to get the effective information, some pre-processing will be applied to these data. The common English stop words will be removed as they are not useful in here.

We decided to split the train.csv in to training and developing set (30000 instances as training set and 10000 instances as development set), in this way, we will be able to test the accuracy of the development set easily since the duration label are known here. Additionally, we are able to use some evaluation metrics later since the actual labels are known.
As many scikit-learn classifiers cannot directly process string data, we will convert the texts into numerical data for further computation. Therefore, we used counter vectorizer to convert the texts into a sparse matrix. Counter Vectorizer is a method that simply count the number of times a word appears. Also, we choose tf-idf to process the data to compare with the counter vectorizer. Tf-idf vectorizer calculates the overall weight of each word, which penalizes the common word in the classes. We will use these two methods to process the text for each classification algorithms to obtain a better result.

2.1.2 Feature Selection

We are then trying to implement feature selection to get the useful features.



Through the visualisation of the n steps and n ingredients by using boxplot, we can see that the instances with duration label 2 clearly have higher n steps / n ingredients

Figure 1 – Boxplots for 'n steps' and 'n ingredients" for different class labels

than those with duration label 1. However, it seems like the instances with duration label 2 has similar n steps / n ingredients range compare the duration label 3, and the mean is even lower than the instances with duration label 3. That is reasonable to some extend since some recipe might not have so many numbers of steps, but they can have many things to do in single step. So we assume that the number of steps is not a really appropriate feature in classifying the duration label 2 and 3. Hence we choose to exclude the n steps in the features for predict the labels.

Furthermore, we then look at the remaining three features "name", "ingredients" and "steps". We plan to examine that use which feature combination to train the classifiers would get the best performance. Firstly, we concatenate the processed features of each instance in one array and train the classifiers to test the accuracy of the following combinations:

- "name" + "steps" + "ingredient"
- "name" + "steps"
- "name" + "ingredient"
- "steps" + "name"
- Each single feature

Finally, find out that "name" and "steps" seem to have the highest accuracy among the classifiers. As a result, we decided to just use "name" and "steps" to train the classifiers.

2.2 Machine Learning Algorithms

We employ the following 5 classifiers to predict the class labels for the test dataset using some scikit learn packages, include one ensemble model:

– Multinomial Naïve Bayes
– Logistic Regression
– Support Vector Classifier (Linear SVC)
– Random Forest
– Ensemble Model

We train the classifiers by using the 30000 instances for train.csv and test on the 10000 development set. The features "steps" and "name" are used to train these classifiers. We apply counter vectorizer and tf-idf to process the data in order to find which works better for each classifier on this dataset. Below are the details of the classifiers.

2.2.1 Multinomial Naïve Bayes

After the pre-processing of the datasets, we build a baseline model using Multinomial Naïve Bayes classifier (MNB). MNB is a very popular algorithm based on the principle of conditional probability of Bayes' theorem, where every pair of features being classified is independent of each other. This algorithm is suitable for classification with discrete features, and also works for fractional counts such as tf-idf.

2.2.2 Logistic Regression

After building the MNB as a baseline model, we try to use logistic regression classifier. The Logistic Regression measures the probabilities of the classification result by estimating probabilities using a logistic function, which is forms a linear decision boundary.

2.2.3 Support Vector Classifier

The SVM classifier aimed to find a find hyperplane that maximumly separates two classes.

2.2.4 Random Forest

Random forest classifier is an ensemble of Random Trees, many trees and each tree is built using a different Bagged training dataset. It then aggregates classification result from different decision trees by voting to decide the final class of each test instance.

2.2.5 Ensemble Model

Max voting is used to evaluate on the Linear SVC, Logistic Regression and the SGD classifier. It finds out the mode label that is been classified by these three classifiers for each instance. By using multiple classifiers to find the labels, we improve the reliability of our results.

2.3 Improve Model

Whilst MNB and Random Forest have relatively reasonable classification accuracy on training and developing set. Logistic regression and SVM have much higher accuracy on training set than developing set. We reckon that there has been "overfitting" for these classifiers. That might because of the decision boundary distorted by noise, which implies the variation might be high. A simpler decision boundary might generalise better for this data, so we choose to tune the parameters to find the best parameter that generalises the dataset well.
Also, we decide to perform a 5-fold cross-validation procedure for them to reduce some bias and get a more reasonable result.

2.3 Grid Search

We chose to use the SDG classifier to perform grid search since it simulates LinearSVC and Logistic regression through different loss functions. From the grid search, it suggests that the models with top performances all have an alpha of 0.0001, while using a hinge loss function and 5000-all features. The score function does not have an obvious impact on the results. The better performance when using hinge loss function indicates that LinearSVC might be used to build a better model. Moreover, from the alpha we get from the SDG classifier, the regularisation term C that should be used in LinearSVC is approximately 1/3 whereas the use of different score functions does not have a clear impact on the accuracy of the model.

Hence, I further conduct another grid search only use the LinearSVC. This time I set the range of k from 4000 to all features since in the previous grid search, I found out that the best performing k is around 5000 to all. To visualise the result, we graph a heatmap of the accuracy score against different hyperparameter k and C, we can see that the highest accuracy score is obtained when around 8000 features are used with a C value of 0.3.

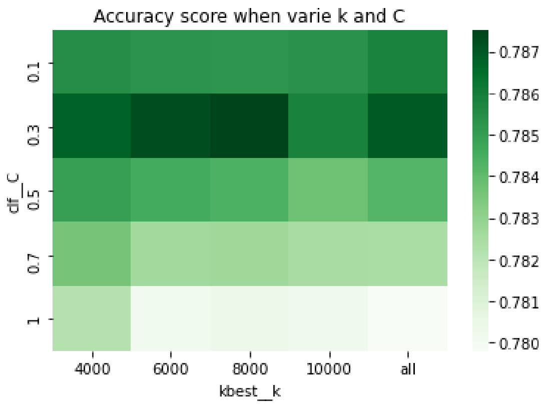


Figure 2 – Heatmap of accuracy when changing the parameter k and C.

3 Results

3. 1 Evaluation Metrics

The accuracy score on the development set is as follows:

model	Method to transform text	Feature used	Accuracy score (dev set)
Multinomial Naive Bayes	Count Vectoriser	Name, steps, ingredients	0.7213
Multinomial Naive Bayes	Count Vectoriser	steps	0. 7234
Multinomial Naive Bayes	Count Vectoriser	Name, steps	0.7247
Multinomial Naive Bayes	Tfidf	Name, steps	0.6994
Linear SVC	Count Vectoriser	Name, steps	0.7531
Linear SVC	Tfidf	Name, steps	0.7828
Logistic Regression	Count Vectoriser	Name, steps	0.7725
Logistic Regression	Tfidf	Name, steps	0.7876
Ensemble Model (max voting: svc+lr+sgd)	Tfidf	Name, steps	0.7882
Random Forest	Tfidf	Name, steps	0.7778

Table 1. Accuracy score on the development set when using different models

Form the results above, we can see that all of SVM, logistic regression, random forest and ensemble model has a better performance than the MNB baseline model. Out of all the models that we build, MNB that use tfidf to process the words have the worst performance, which is reasonable since tfidf assign each word a continuous score instead of counting the number of times that they appear. The accuracy will be lower since we are using continuous feature to fit a discrete model. We also find out that the Ensemble Model using max voting, the model of Logistic Regression using tfidf and Linear SVC using tfidf have a relatively high performance on the development set. Hence, once we use the model to test on the actual test set, we obtain an accuracy of 0.80166, result = 0.80033 and 0.79666 respectively. As a result, we can see that the Ensemble Model that select name and steps as its feature, with tfidf to process the words will give us the highest accuracy.

We further use the precision and recall as the evaluation metric. Under the baseline model using Multinomial Naive Bayes, we obtain a confusion matrix below.

	Predicted		
Actual	1	2	3
1	3169	1204	104
2	1053	3864	109
3	86	89	322

Table 2. Confusion matrix for Multinomial Naïve Bayes

From the matrix above we can get that the recall for each of the class is (0.7356, 0.7493, 0.6019) and the recall (0.7078, 0.7688, 0.6479). This indicates that for duration label 3, there is approximately 40% of the positive identifications was actually wrong. The relatively low precision and recall might due to the small sample in training set (2049/40000 training instances), hence the model might not efficiently learn the data well and tend to make systematically wrong predictions. The small sample set for the duration label 3 itself might lead to higher variability or bias.

The confusion matrix for Linear SVC and Logistic Regression in Tables 3, 4, 5 below.

Table 3. Confusion matrix for Logistic Regression

	Predicted		
Actual	1	2	3
1	3528	902	47
2	937	4054	35
3	94	109	294

Table 4. Confusion matrix for Linear SVC

	Predicted		
Actual	1	2	3
1	3548	896	33
2	963	4035	28
3	93	108	296

	Predicted		
Actual	1	2	3
1	3538	905	34
2	938	4056	32
3	95	114	288

Table 4. Confusion matrix for ensemble model (max voting)

By analysing the confusion matrix for these three models, we find that they all have similar precision and recall. Their precisions are all around [0.77, 0.80, 0.80] for each label and recall around [0.79, 0.80, 0.59]. A clear problem is that the recall for duration label 3 are quite low for these models, which means that there are too many false negatives, which means lots of the label 3 recipes has been incorrectly classified to other labels.

4 Conclusion & Future Work

In Summary, we predict the cooking time as some class labels for recipes by using some attribute such as dish name and cooking steps. Varies machine learning algorithms has been used to conduct the classification. Among these methods, ensemble model that use max voting has the highest accuracy for predicting the cooking duration labels on the test dataset, which achieves 0.80166 accuracy score. The feature “name” and “steps” are used in training the model, and the tf-idf vectorizer is the tool we employed for convert the text data into numerical value. During the project, we have utilized tools such as grid search and confusion matrix to help us obtain a better result.

Overall, we get an appropriate outcome although it still have some room for improvement as it did not reach the benchmark. Future work will also include exploring other techniques to further improve the model performance and stability, try to increase the accuracy while reducing the bias and variance. One thing I might improve next time is that in the feature selection, we mostly depend on our intuition and use graph or “trial and error” to find the features needed. However, I would like to try to use methods like sequential forward selection or filtering to engineer feature in a smarter way. Moreover, although accuracy is very important in the classification task, but considering the computation complexity is also essential for us to think about in machine learning. In future task, we would also consider the computation time and space complexity which aim to build more efficient models.

5 Reference

Generating Personalized Recipes from Historical User Preferences. Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, Julian McAuley, in 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.

Laura M. Stevens, Bobak J. Mortazavi, Rahul C. Deo, Lesley Curtis, David P. Kao, 2020, ‘*Recommendations for Reporting Machine Learning Analyses in Clinical Research*’, viewed 16 May 2021, <https://www.ahajournals.org/doi/full/10.1161/CIRCOUTCOMES.120.006556 >

Dan Nelson, 2021, ‘*Overview of Classification Methods in Python with Scikit-Learn*’, viewed 16 May 2021, <https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/ >

Junjie Wang1 , Qiang Cui1 , Qing Wang1,2 , Song Wang3, (n.d.), *‘Towards Effectively Test Report Classification to Assist Crowdsourced Testing’*, viewed 17 May 2021, <https://ece.uwaterloo.ca/~s446wang/paper/esem-16.pdf >

Antonio Rodriguez, Frederic Bartumeus, and Ricard Gavaldà, (n.d.), *‘Machine Learning Assists the Classification of Reports by Citizens on Disease-Carrying Mosquitoes’*, viewed 17 May 2021, <https://core.ac.uk/download/pdf/87651724.pdf >

Sudharsan Asaithambi, 2018, *‘Why, How and When to apply Feature Selection’*, Towards Data Science, viewed 18 May 2021, <https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2 >

Paritosh Kumar, 2019, *‘Computational Complexity of ML Models’*, Analytic Vidhya, viewed 20 May 2021, <https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770 >