Machine Learning (COMP30027)
Ordinal Classification of Cooking Time of Recipes

1. Introduction

Food.com (or formerly known as Genius Kitchen), is a popular website where visitors can access a broad range of food recipes.

The aim of this task is to perform ordinal classification on the estimated preparation time of recipes, which are classified as 1, 2, or 3 (corresponding to fast, medium and slow preparation times) (Majumder et al. 2019).

2. Initial Model Selection

The classification problem extends to three classes, meaning any binary model chosen would need to be able to extend to a multiclass (and ordinal) problem.

The models of K-Nearest Neighbours Classification and Ordinal Logistic Regression were chosen for experimentation. K-NN has shown strong effectiveness in handling text features (Soucy and Mineau 2001).

Multinomial Logistic Regression should also perform well for this task (Madigan 2005), and Ordinal Logistic Regression would be an improvement over Multinomial.

3. Feature Engineering

3.1 Finding the optimal representation

The datasets provided were of multiple representations for the text features (name, steps and ingredients), and two numerical features of number of steps (n_steps) and number of ingredients (n_ingredients).

The representations were doc2vec with 50 and 100 features, and bag-of-words (raw frequency count vectors).

Several training sets were derived from these representations, and the models were evaluated on an 80-20 holdout strategy. This means 80% of the training set is used for training and then evaluated over the remaining 20%.

The tested text feature sets are:

1, Doc2vec50

2, Doc2vec100

3, Feature selection from raw frequency counts with chi-squared for 50 and 100 features

4, Feature selection from Term Frequency – Inverse Document Frequency (TF-IDF from bag-of-words) with chi-squared for 50 and 100 features

5, TruncatedSVD dimensionality reduction with 50 and 100 components

Each of these representations are for 1 text feature, so the training is generated by combining the 3 sets of text features and the 2 numerical values, giving a dimension of either (40000, 152) or (40000, 302).

All features were also scaled for better performance.

Feature Representation	K-NN	Logistic Regression
Doc2vec50	0.619	0.706
Doc2vec100	0.535	0.705
Chi2 Feature Selection, k = 50	0.715	0.775
Chi2 Feature Selection, k = 100	0.687	0.784
Chi2 TF-IDF Feature Selection, k = 50	0.728	0.773
Chi2 TF-IDF Feature Selection, k = 100	0.714	0.782
TruncatedSVD, n-components = 50	0.666	0.740
TruncatedSVD, n-components = 100	0.621	0.760

Table 1 – Accuracy of models across different represenations

From table 1, chi-squared feature selection outperforms both doc2vec representations and truncatedSVD.

Doc2vec is an embedding method that maps words in similar contexts near each other. However, given the large variability of recipes, some of this information would have been redundant and instead become noise in the data.

TruncatedSVD is a reduction method that aims to capture the maximum variability in the data; but as with doc2vec, a lot of features may have been redundant and add noise to the data.

For chi-squared feature selection, it is observed that TF-IDF performed better for K-NN, while raw frequency counts (bag-of-words) did better for Ordinal Logistic Regression.

For K-NN, TF-IDF may performing better is

likely due TF-IDF factoring the rarity of words which played a role in labelling.

For Ordinal Logistic Regression, the normalization involved in TF-IDF may have led to key frequency counts being undervalued.

Thus, chi-squared with TF-IDF was used for K-NN, and chi-squared with frequency counts was used for Ordinal Logistic Regression.

3.2 Finding the optimal number of features

In table 1, it was observed that for both models, accuracies changed with the number of features, creating the need to determine the ideal number of features selected.

This method involved selecting the k features with the highest chi-squared value over the whole training data and testing for accuracy for each model.

3.2.1 Feature filtering for K-NN

The method explained above produced the following results:

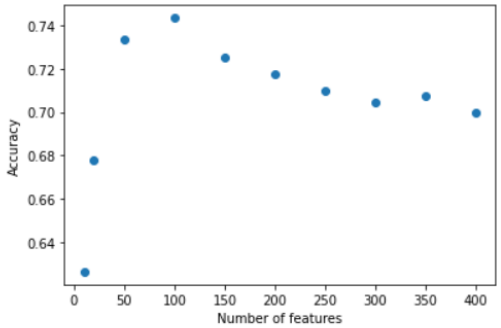

Figure 1 – Accuracy of K-NN across different feature numbers

The performance of the model declined as k > 100 (Figure 1), indicating that features beyond this range contains noise. Thus, k = 100 was chosen for K-NN.

3.2.2 Feature filtering for Logistic Regression

The method explained above produced the following results:

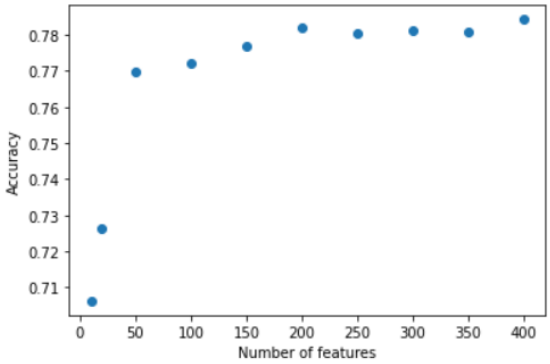

Figure 2 – Accuracy of Ordinal Logistic Regression across different feature numbers

In Figure 2, it appears that performance is fluctuating at high feature numbers, so higher feature numbers were also tested.

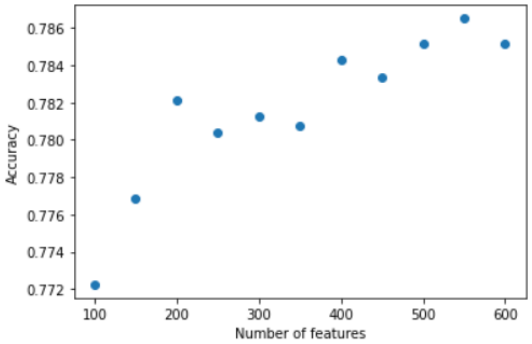

Figure 3 – Accuracy of Ordinal Logistic Regression with higher numbers of features

In Figure 3, the performance seems to rise with higher features but at a diminishing rate. Thus, k = 400 was chosen as it should keep have better accuracy but avoid severe overfitting.

4. Improving Base Models

4.1 Improving K-NN

4.1.1 Tuning n_neighbors

The base K-NN model using the dataset outlined in section 3.2.1 performed at a reasonable accuracy of 74.35%.

K-NN's reasonable performance it has captured a large amount of useful information from the dataset.

However, by tuning the number of neighbours (n_neighbors hyperparameter) considered, we can trade off training time with potentially improved performance.

Testing this gave the following result:

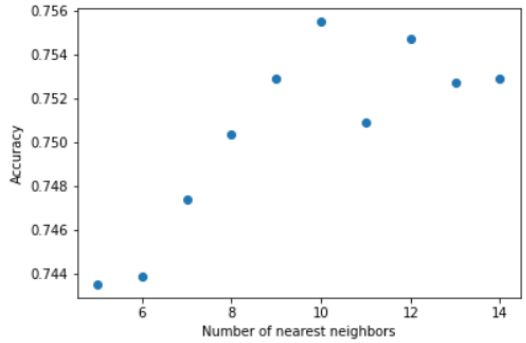

Figure 4 – Accuracy of K-NN against n_neighbors

In Figure 4, performance seems to increase linearly up to n_neighbors = 10, at which the score begins to fluctuate. The value of 10 is then chosen.

4.1.2 Implementing an ordinal version of K-NN

K-NN so far has performed at a good accuracy, reaching 75.29% after tuning n_neighbors and performing feature selection

However, K-NN is a categorical classifier, so the ordinal nature of the labels is unutilised.

An ordinal (label-weighted) K-NN model has shown to perform better than a normal K-NN model, by considering the weight of class labels (Hechenbichler & Schliep 2004).

With this motivation, a label-weighted K-NN is created with sklearn's KNeighborsRegressor and converting the output to ordinal labels.

4.2 Improving Logistic Regression

For Logistic Regression, the model trained using the data set described in 3.2.2 performed at an accuracy of 77.8%.

Logistic Regression's good performance indicates that the data roughly follows assumptions of no multicollinearity, and that there are features with good correlation with the labels.

However, by tuning the regularization hyperparameter, we can shrink less useful features to zero and reduce overfitting.

Testing this gave the following result:

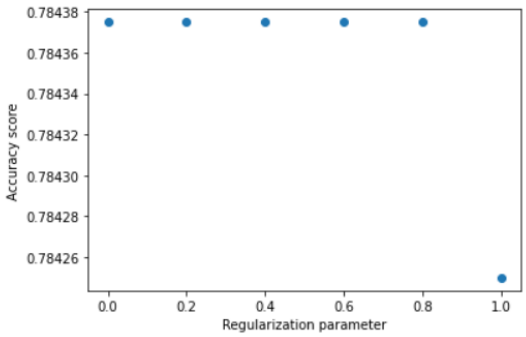

Figure 5 – Accuracy of Logistic Regression against regularization

In figure 5, there was almost no difference in the accuracy of the model while changing regularization. This means that less useful coefficients are already close to zero. A regularization hyperparameter of 0.8 is then chosen.

5 Final Implementation

The two finalised models are:

Ordinal K-NN with 100-feature TF-IDF, n_neighbors = 10

LogisticIT Ordinal Regression with 400-feature frequency counts, regularization = 0.8

Implementing these on the training set:

For Ordinal K-NN:

True Labels	1	2797	697	6
	2	1061	3050	9
	3	55	153	172
		1	2	3
		Predicted Labels		

Table 2 - Confusion matrix for K-NN

The accuracy, precision and recall (from micro-averaging) were all 75.24%. The Kaggle score was 74.47%.

For Ordinal Logistic Regression:

True Labels	1	2832	658	10
	2	808	3246	66
	3	31	152	197
		1	2	3
		Predicted Labels		

Table 3 - Confusion matrix for Logistic Regression

The accuracy, precision and recall (from micro-averaging) were all 78.44 %. The Kaggle score was 78.5%.

6 Evaluation

From the results, feature selection and hyperparameter tuning had minor benefits on both the classifiers.

The classifiers generalised well onto the Kaggle data, so there is little evidence of overfitting taking place.

Despite expectations, Ordinal K-NN failed to outperform K-NN over the holdout data (75.24% to 75.29%)

A possible explanation for this is:

From the confusion matrix, the data is mainly focused on labels 1 and 2. Thus, the classifier is effectively a binary classifier. With just two labels, the fitting a regression line adds provides no additional meaningful information to the model.

Ordinal Logistic Regression performed closer to the top percentiles of Kaggle (4% below the benchmark). This indicates that the model was fitted reasonably well with the data, and that most of the inaccuracy lies within the data's own variability. However, the difference in performance perhaps indicates that the features selected did not contain all the information needed – which is supported by small accuracy increases with higher feature numbers in figure 3.

7 Concluding Remarks

By performing feature engineering and selection and tuning model parameters, minor performance improvements were detected. The results showed Ordinal Logistic Regression performing better than K-NN and close to the benchmark, indicating that it was an appropriate choice for the data.

To handle the complexity of K-NN, a simple 80-20 holdout strategy was used while the model was being built. Performing k-fold cross validation in may yield more conclusive results in the future.

Further attempts to tackle this problem can also involve other models which have been successful for ordinal classification, such as SVMs, Decision Trees & Random Forests, all of which have shown better performance than Logistic Regression in other studies (Saad & Yang 2019). Other non-ordinal models can also be considered, as the results indicated normal (nominal) K-NN performing similarly to an ordinal variant.

8 References

Majumder, BP, Li, S, Ni, J & McAuley, J 2019, '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)*.

Soucy, P & Mineau, GW 2001, 'A simple KNN algorithm for text categorization', *Proceedings 2001 IEEE International Conference on Data Mining*.

Madigan, D 2005, 'Bayesian Multinomial Logistic Regression for Author Identification', *AIP Conference Proceedings*.

Hechenbichler, K & Schliep, K 2004, 'Hechenbichler, Schliep: Weighted k-Nearest-Neighbor Techniques and Ordinal Classification Projektpartner Weighted k-Nearest-Neighbor Techniques and Ordinal Classification', *Sonderforschungsbereich*, vol. 386,.

Saad, SE & Yang, J 2019, 'Twitter Sentiment Analysis Based on Ordinal Regression', *IEEE Access*, vol. 7, pp. 163677–163685.