# COMP30027 Report

## 1. Introduction

.

This project aims to predict the cooking time for recipes based on their steps, ingredients and other features using Machine Learning methods using datasets from Majumder (Majumder et al., 2019). This task is a three-class classification problem with three classes, corresponding to quick, medium and slow. The experiment involves the use of supervised classification models, Logistic regression, Random Forest and decision trees, and one unsupervised classification model, neural networks.

The issue of class imbalance and class overlap restricts the ability of conventional classifiers to predict. Traditional ways to deal with class imbalance and overlap include resampling the dataset such as oversampling the minority class and under sampling the majority class. This project uses the Synthetic Minority Oversampling Technique (SMOTE) method (Nitesh Chawla, et al., 2002)

## 2. Methodology

### 2.1 Data Exploration

For this experiment, we use a training dataset, recipe (Majumder et al., 2019), with 40000 instances and 3 classes, and a test dataset with 10000 instances. There are two observations. Firstly, distribution of class labels is highly imbalanced, with class 3 constituting only 5.1% of the entire dataset (Fig.1). Secondly, there is significant class overlap in which data samples appear valid instances of more than one class (Fig. 2).

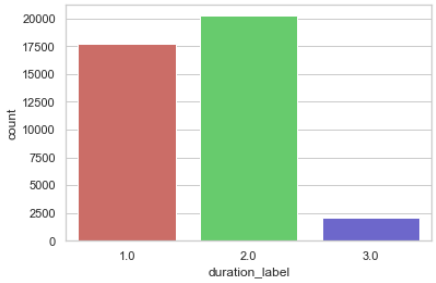


**Figure 1**. Instance counts of class labels.

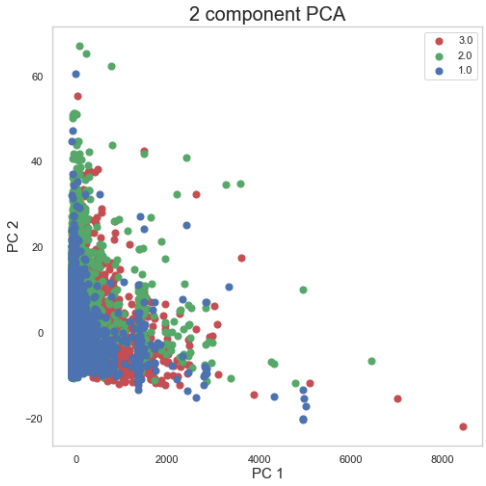


**Figure 2**. Visualization of class clusters. 2D Principal Component Analysis was performed for a 2D visualization of the class instances.

### 2.2 Feature Engineering and Selection

| Duration label | n_steps | n_ingredients | times |
|---|---|---|---|
| 1.0 | 7.635922 | 7.851341 | 34.159910 |
| 2.0 | 11.472044 | 10.135879 | 71.137885 |
| 3.0 | 10.123963 | 9.648121 | 558.978429 |

**Figure 3**. Median values of numerical features based on class labels.

Originally, there are two numerical features, n_steps and n_ingredients, and three text features, ingredients, steps and name.

Manual inspection of 'steps' indicates that numbers could inform cooking time. One feature 'Times' is engineered by extracting the sum of the numbers in each 'step' using regular expression. 'Times' is a good predictor of duration label since there are large differences between classes (Fig. 3), and the magnitude of 'times' well correlates with the length of cooking time.

The text features were converted to vectors of size 50, which can then be fed into estimators for prediction. The size 50 was chosen instead of 100 to give more weight to the numerical features once horizontally stacked.

### 2.3 Train-Validation

Feature selection was implemented during the train-evaluation phase. A 10-fold repeated stratified cross validation was conducted to evaluate the classification accuracy of each model using a combination of features. The set of features that give the highest accuracy were chosen. The final features selected were horizontally stacked (Fig. 4).

| Feature | Description |
|---|---|
| n_steps | Number (integer) of steps of cooking the meal |
| n_ingredients | Number (integer) of ingredients |
| Steps_vec50 | Vector of size 50 representing the text feature 'steps' |
| Times | Time of cooking, extracted from 'steps' |

**Figure 4**. Features generated.

### 2.4 Bordline SMOTE for Imbalanced Classification

To deal with the class imbalance, new synthetic samples from minority class were synthesized. The decision boundary was approximated by the support vectors "*obtained after training a standard SVMs classifier on the original training set. New instances will be randomly created along the lines joining each minority class support vector with a number of its nearest neighbors using the interpolation"* (Hien Nguyen, et al., 2009).

This method involves selecting misclassified instances of the minority class and oversampling the difficult instances, thus improving accuracy. Moreover, this method focuses on synthesizing samples near the bord line region, which increases the distinction between two classes.

The method was implemented via the SVMSMOTE class from the imbalanced-learn library (Hien Nguyen, et al., 2009).

The use of SMOTE method significantly increased the accuracy on validation set but led to little change in accuracy on test dataset (Fig. 6). Adjustments to SMOTE or other methods are required to address class imbalance.

## 3. Discussion

| Model | Class | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Decision tree | 1 | 0.68 | 0.68 | 0.68 | 0.690 |
| | 2 | 0.71 | 0.71 | 0.71 | |
| | 3 | 0.52 | 0.53 | 0.53 | |
| AdaBoost | 1 | 0.75 | 0.78 | 0.76 | 0.765 |
| | 2 | 0.79 | 0.77 | 0.78 | |
| | 3 | 0.64 | 0.65 | 0.64 | |
| MLPC | 1 | 0.76 | 0.76 | 0.76 | 0.770 |
| | 2 | 0.78 | 0.79 | 0.79 | |
| | 3 | 0.70 | 0.61 | 0.65 | |
| Logistic Regression | 1 | 0.7 | 0.73 | 0.71 | 0.713 |
| | 2 | 0.73 | 0.74 | 0.73 | |
| | 3 | 0.64 | 0.30 | 0.41 | |
| Random forest | 1 | 0.78 | 0.72 | 0.75 | 0.741 |
| | 2 | 0.72 | 0.83 | 0.77 | |
| | 3 | 0 | 0 | 0 | |
| Random forest with class weighting | 1 | 0.72 | 0.76 | 0.74 | 0.742 |
| | 2 | 0.76 | 0.75 | 0.76 | |
| | 3 | 0.76 | 0.49 | 0.60 | |

**Figure 5a**. Accuracy scores of classifiers after a 10-fold cross validation without the use of SMOTE.

| Model | Class | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|
| Decision tree | 1 | 0.70 | 0.79 | 0.74 | 0.793 |
| | 2 | 0.78 | 0.71 | 0.74 | |
| | 3 | 0.91 | 0.88 | 0.89 | |
| AdaBoost | 1 | 0.73 | 0.76 | 0.75 | 0.799 |
| | 2 | 0.75 | 0.74 | 0.74 | |
| | 3 | 0.91 | 0.90 | 0.91 | |
| MLPC | 1 | 0.77 | 0.77 | 0.77 | 0.821 |
| | 2 | 0.78 | 0.73 | 0.75 | |
| | 3 | 0.90 | 0.97 | 0.93 | |
| Logistic Regression | 1 | 0.67 | 0.72 | 0.69 | 0.735 |
| | 2 | 0.68 | 0.65 | 0.66 | |
| | 3 | 0.86 | 0.84 | 0.85 | |
| Random forest | 1 | 0.65 | 0.65 | 0.65 | 0.731 |
| | 2 | 0.65 | 0.67 | 0.66 | |
| | 3 | 0.91 | 0.88 | 0.89 | |
| Random forest with class weighting | 1 | 0.74 | 0.79 | 0.77 | 0.823 |
| | 2 | 0.78 | 0.70 | 0.74 | |
| | 3 | 0.95 | 0.97 | 0.96 | |

**Figure 5b**. Accuracy scores of classifiers after a 10-fold cross validation with the use of SMOTE.

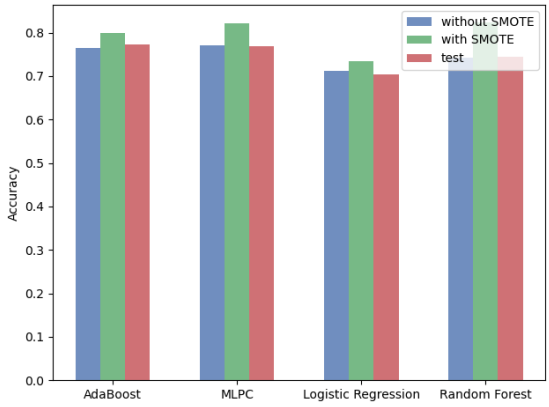


**Figure 6**. Graph of accuracy scores of classifiers after a 10-fold cross validation with and without the use of SMOTE, and on the test dataset

## 4. Error Analysis and Interpretation

### 4.1 Decision Trees

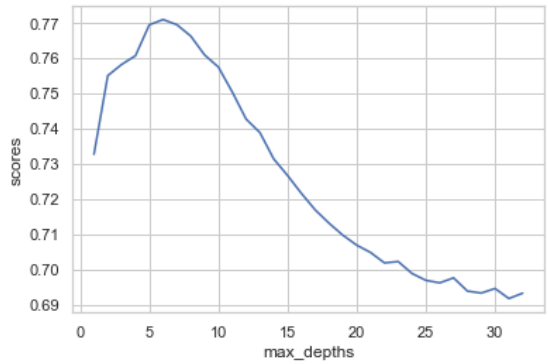


**Figure 7**. Accuracy scores of decision tree classifier with max_depth 1 – 32. Max_depth = 6 was chosen.

Decision tree achieved the lowest accuracy among all classifiers since it is susceptible to the effects of irrelevant features.

The maximum depth is the stopping condition of a single decision tree. To prevent overfitting and optimize accuracy, max_depth is tuned and 6 was chosen. The tuning and SMOTE significantly improved the accuracy (from 0.69 to 0.79, Fig 5a and Fig 5b).

Decision trees performed better on the minority class compared to the standard random forest (Fig 8 & Fig 10) since the splitting criteria used in the creation of a single tree can force the minority class to

be included.

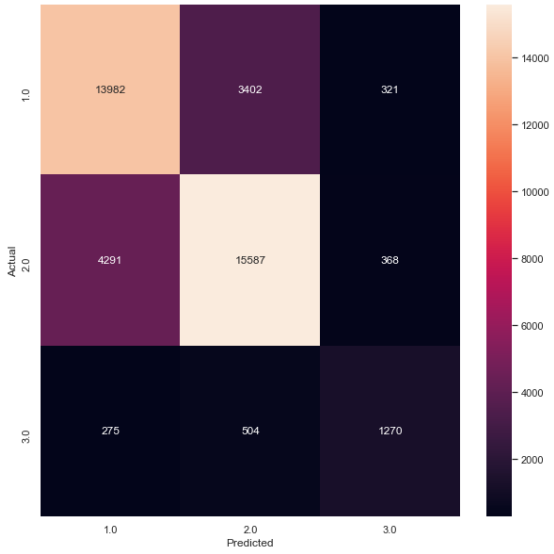


**Figure 8**. Confusion matrix of decision trees.

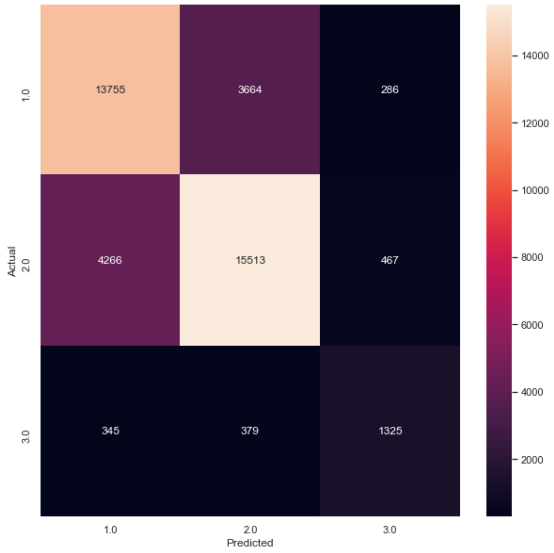


**Figure 9**. Confusion matrix of AdaBoost.

To improve the low precision and recall score for minority class, an ensemble method 'AdaBoost' was used. The AdaBoost classifier is a meta-estimator that begins by fitting the decision tree on the original dataset and then focuses more on difficult classes by adjusting the weights of incorrectly classified instances. AdaBoost improved the accuracy score significantly from 0.690 to 0.765. The recall and precision score for class 3 instances also improved significantly (from 0.53 to 0.64, Fig 5a).

## 4.2 Random Forest

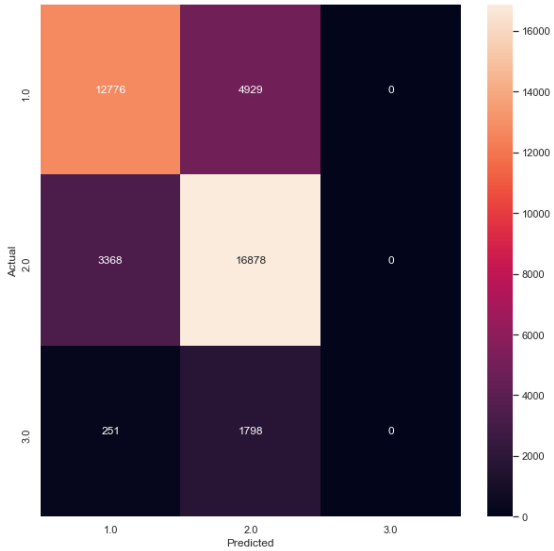


**Figure 10**. Confusion matrix of standard Random Forest.

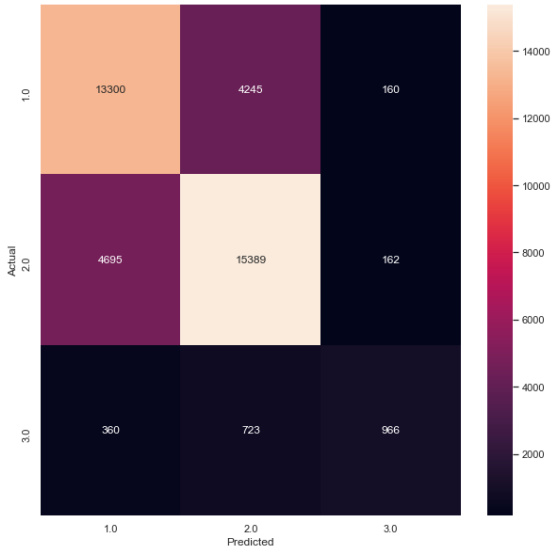


**Figure 11.** Confusion matrix of Random Forest with class balance.

Random Forest was chosen because of the high dimensionality of the input data and the ease of having little hyperparameters. The random sampling of training instances and feature lessens the effects of irrelevant features, which was seen in decision trees.

Gini impurity was used as the impurity measure. Two values of the hyperparameter 'class_weight' were ran and contrasted. The standard random forest achieved an accuracy score of 0.743,

significantly higher than that of a single decision tree.

Analysis of the confusion matrix (Fig 10) shows that no instances were classified as class 3. This indicates that due to extreme data sparsity no class 3 instances were sampled, which would lead to poor results when exposed to test datasets if class 3 is majority class.

Hence, a modified random forest with {class_weight = 'balanced'} was performed. The incorporation of class weighting improved the accuracy slightly from 0.741 to 0.744 (Fig. 5a), yet it significantly improved the precision and recall of class 3 instances (Fig 10 & 11).

### 4.3 Logistic Regression

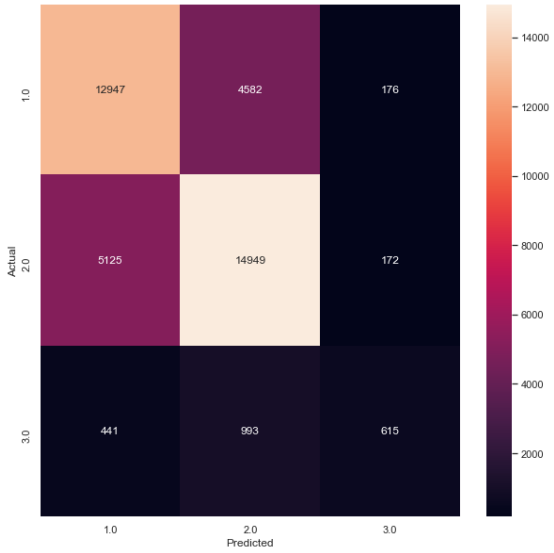


**Figure 12**. Confusion matrix of logistic regression.

A multinomial logistic regression model was used by splitting the multi-class classification problem into multiple binary classification problems and fitting a standard model on each subset. Class labels were assigned based on probabilities.

Logistic regression achieved a low accuracy score of 0.713. There are three possible reasons. Firstly, logistic regression is by default powerful for binary classification problems. As shown in Fig 2, the distinction between class 2 and 3 is minor, which adds difficulty to the model. Secondly, logistic regression assumes a linearity between the dependent variable and the independent variables, and it makes linear decision boundaries. However, this assumption may not be well satisfied since the data is not linearly separable, as shown in Fig 2. Thirdly, logistic regression adopts a probabilistic function, thus it is not powerful and flexible to capture complex relationships. This is validated from the low R squared value of 0.438, which the inefficiency of the function for accurate predictions.

The confusion matrix shows that the model tends to misclassify class 1 instances as class 2. This indicates class overlap and the non-linearly separable property of the data. To improve, more predictive features need to be engineered to increase margin between classes.

The data sparsity issue of class 3 instances was lessoned using SMOTE, as seen by the increased f1 score (Fig 5b).

### 4.4. Neural Network MPLC

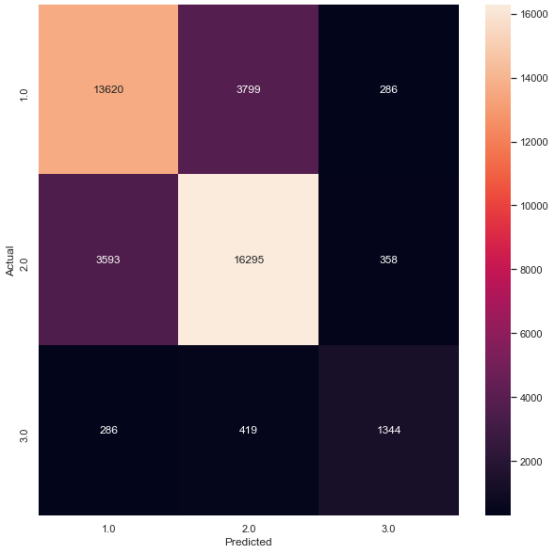


**Figure 13**. Confusion matrix of neural network.

Compared to random forest and logistic regression, the neural network has more hyperparameters, is slower to train, and is more prone to overfitting. However, neural networks achieved the highest accuracy score (0.77, Fig 5a) because the model is

flexible to model arbitrary basis functions. The idea of representation learning allows the model to learn a feature hierarchy from simple combinations of the input to more complex features, which is suitable for the text classification task.

To avoid overfitting, parameters was tuned using grid search. The learning rate was chosen to be 0.05, and the number of layer was (10, 30, 10) to reduce time complexity.

Parameter tuning was found to improve accuracy from 77.6% to 78.6%.

Neural networks correctly predicted the greatest number of the minority class instances, which indicates the flexibility of the model to handle class imbalance.

However, the high false positive rates (Fig. 4) indicates that the perceptron algorithm is not guaranteed to converge over non-linearly separable data.

## 5. Conclusions

The Neural Network MPLC classifier achieved the highest accuracy score 0.782 among other classifiers. The use of bordline SMOTE increased classification accuracy by solving the class imbalance issue. Further methods are needed to solve the class overlap issue.

## 6. References


1. 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.
2. Nguyen Hien M., Cooper Eric W., Kamei Katsuari. Borderline Over-sampling for Imbalanced Data Classification, 2009.
3. K. W. Bowyer, SMOTE: Synthetic Minority Over-sampling Technique, 2002.