In order to predict the cooking times of various recipes, different models of classifiers were utilized, and features were engineered to explore what the most optimal way of classification is. In this report, three different classifiers were explored: Naïve Bayes, Decision Tree and K-nearest neighbor classifiers.

In order to initialize the model building, preprocessing the data was the first approach. This applied to the feature's 'name', 'steps' and 'ingredients'. The preprocessing done was to countvectorize the unstructured data (text) into structured data as a sparse matrix of bag-of-words, which can be passed into classifiers. Since evaluation was needed with only the train data available, two approaches were considered: the holdout approach and the cross-validation approach. However, as the dataset is large, the cross-validation approach was thought to heavily burden the computational time and hence the holdout method seemed sufficient in evaluation. Also, the way the features were utilized together was simply by horizontally stacking them onto each other and processing them through classifiers.

Since the Naïve Bayes model is a simple classifier, it is known to work efficiently with large datasets such as recipe_train. The classifier used was the MultinomialNB, as there will be counts of numerous words, which is not fit for BernoulliNB. Also, since the data is not normally distributed, GaussianNB is also not ideal. Furthermore, since there are three classes (slow, medium, fast) that can classify the speed of any recipe, it may be assumed that the test dataset will not contain instances that require a new class. In this case, zero posterior probability will not arise and cause an error when processing the Naïve Bayes classifier. There could be a problem in the model however, if the features are dependent. In this dataset, some features may be slightly dependent in the way that the words in a feature may overlap in another feature. This may apply to steps and ingredients, where the steps would have to mention most ingredients, but the names of the recipe may not be as dependent to other features. Hence when using the Naïve Bayes classifier, the names was the key feature and the other features were added or discarded, in a sequential forward selection fashion.

On the other hand, the decision tree classifier is not an ideal classifier when dealing with large datasets. However, it is still able to classify instances and has an embedded feature selection method where only selected features are used in the tree growth process of each node. This was an extremely simple use as a benchmark to compare it to the Naïve Bayes classifier results.

In regard to feature selection, the sequential forward selection technique was used. In order to choose the most optimal combination of features, the name feature was initially used (as it had the highest accuracy) and features were added on, eventually resulting in 'name + steps' for Naïve Bayes and 'name + steps + ingredients + number of ingredients' for Decision Tree as the best performing combinations. This may be because steps and ingredients are suspected to be overlapping in many words and since steps additionally contain terminology that directly affects the target (duration time), the feature steps was chosen over ingredients. Hence name and steps seemed to be when the Naïve Bayes performed the best. However, for the decision tree, a numeric metadata column 'number of ingredients' seemed to aid the algorithm's ability to classify as this feature may correlate as a direct measure in the duration of a recipe.

```
 MNB
complete acc 0.7332575757575758
x2 acc 0.7251515151515151
mi acc 0.7252272727272727

 Decision Tree
complete acc 0.7387878787878788
x2 acc 0.7379545454545454
mi acc 0.7384090909090909
```

Passing the classifier with complete, chi-squared or mutual information based selected data

After the texts were countvectorized and before the features were stacked onto each other, a feature selection was performed on the words of each individual feature. The reason the feature selection was performed before stacking is because when stacked, the independence of each features will not be recognized as they will be in one whole bag-of-words. Hence, by feature selecting each bag-of-words, the important words in a bag-of-words

can be selected in disregard to other bag-of-words. Two different methods were utilized for this; chi-squared and mutual information-based selection. Both methods picked out the words that are most independent to each other up to a given threshold. However, choosing this threshold is extremely hard to determine, so arbitrary values were given, based on the amount of words there were in a bag. Although these feature selection methods were applied to the bags, they did not increase the accuracy of the predictions (based on the subset training set) and hence were not applied in the final model. Moreover, the accuracy increased as the threshold increased for Naïve Bayes, which could mean that the model performs better when the dataset is complete and there are not many words that should be dropped.

0.7412878787878788

	precision	recall	f1-score	support
1.0	0.73	0.72	0.73	5814
2.0	0.76	0.77	0.77	6712
3.0	0.66	0.60	0.63	674
accuracy			0.74	13200
macro avg	0.72	0.70	0.71	13200
weighted avg	0.74	0.74	0.74	13200

```
[[4184 1538   92]
 [1397 5195  120]
 [ 135  133  406]]
```

Naïve Bayes

0.7356818181818182

	precision	recall	f1-score	support
1.0	0.72	0.74	0.73	5814
2.0	0.76	0.74	0.75	6712
3.0	0.64	0.59	0.61	674
accuracy			0.74	13200
macro avg	0.70	0.69	0.70	13200
weighted avg	0.74	0.74	0.74	13200

```
[[4312 1413   89]
 [1572 5000  140]
 [ 128  147  399]]
```

Decision Tree

The above image includes the classification report and confusion matrix of the Naïve Bayes classifier (left) and the Decision Tree classifier (right) was used. The features that were stacked were name and steps, which were the optimal feature combination for Naïve Bayes. However, both classifiers display a highly similar report and from observation, the confusion matrix seems extremely similar as well. Another similarity that can be found from the report is that the classification accuracy for the class '3' (corresponds to fast) seems to be significantly lower than the other classes.

0.7332575757575758

	precision	recall	f1-score	support
1.0	0.73	0.71	0.72	5814
2.0	0.75	0.77	0.76	6712
3.0	0.62	0.62	0.62	674
accuracy			0.73	13200
macro avg	0.70	0.70	0.70	13200
weighted avg	0.73	0.73	0.73	13200

```
[[4124 1591   99]
 [1412 5138  162]
 [ 125  132  417]]
```

Naïve Bayes

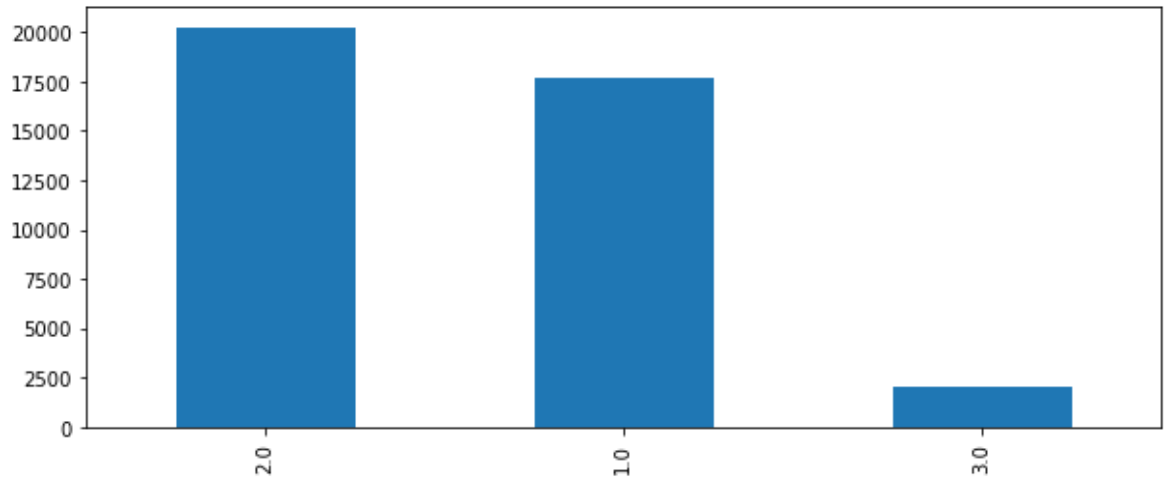
0.7406060606060606

	precision	recall	f1-score	support
1.0	0.72	0.75	0.73	5814
2.0	0.77	0.75	0.76	6712
3.0	0.64	0.57	0.60	674
accuracy			0.74	13200
macro avg	0.71	0.69	0.70	13200
weighted avg	0.74	0.74	0.74	13200

```
[[4332 1398   84]
 [1516 5059  137]
 [ 143  146  385]]
```

Decision Tree

In the case where optimal feature combination for the decision tree was in place, the decision tree indeed had a slightly higher accuracy, although very slight. Also, the results did not highly differ from the results before and the lower accuracy for the class '3' was still observed. Moreover, another observation is that there are very little instances of the class 3 and class 1 and 2 are misclassified often as one another.



Number of instances per class in the training dataset

From the bar graph above, it can be clearly seen that the reason for the relatively low accuracy when classifying the class '3' was due to the imbalance in the dataset. The bar graph above is displaying the amount of instances per class in the training dataset. Through this, it can be observed that there are significantly lower numbers of instances of class 3 compared to the other classes, and hence sufficient information could not be extracted in the training phase to use for the testing phase. This leads to misclassification and perhaps zero posterior probability for Naïve Bayes, in the case that the classifier encounters instances of class 3 but cannot identify the words given (only theoretically, since the classifier is smoothed in python).

For the Kaggle submission, the accuracy of the Naïve Bayes model was higher than the results above (slightly), which may mean that the test dataset is slightly more balanced. This however, cannot be evidence that the main reason of misclassification of the instances is due to imbalance. As the number of class 3 instances are very small, the classifier is mainly trained on class 1 and 2, but there is still misclassification between the two, hence the classifier used or the preprocessing method could be at fault.

From these observations, it seems that CountVectorizing the data and passing it through Naïve Bayes may not be the ideal classifier for this dataset. Although the Decision Tree algorithm was expected to act as a personal baseline, the Naive Bayes model did not perform significantly better than the Decision Tree. Perhaps merely stacking the features could be too simple and another way of combining the features may have been used. An example is classifying the instances using individual features and combining those predictions to pass them into a classifier for a final prediction.

Bibliography:

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 ssProcessing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019