Machine Learning Assignment 2.

Introduction.

This Project required a machine learning model to be developed to predict the recipe length based on its name, steps and ingredients.

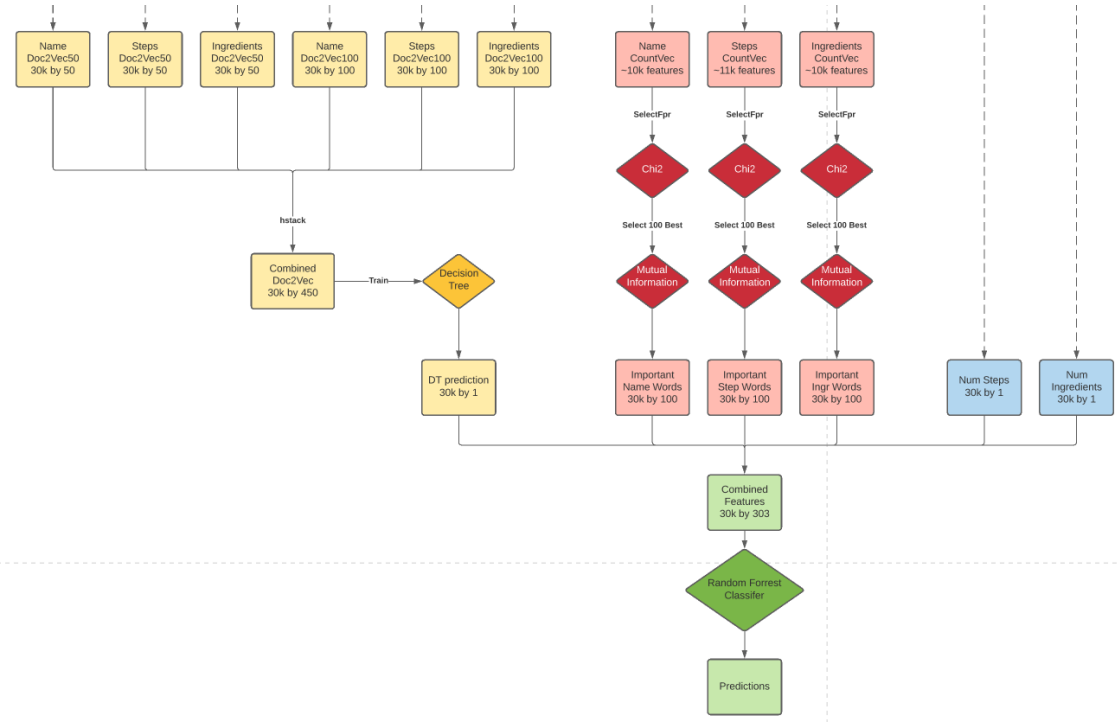
The pre processed data.

The text data features were provided pre processed. This included count vectors for the names, steps and ingredients as well as doc 2 vectors. One downside to word vectors is that it does not store information about the relationships between words. Doc2Vec takes into account the group of words in a feature and engineers vectors based on this. It aims to make text with the same idea have a small distance between them while different idea texts are placed with large distances away from each other. Word stemming was also applied, to the count vector with the french stemming method, which reduces the number of words present. This turned out to perform worse and was scrapped. A possible reason is that the word endings sometimes contains important information for classification. For example, the word “minute” indicates a quick recipe but when it is stemmed from the words “60 minutes” the stemming caused information to be lost.

Overview of Feature Engineering and Feature selection.

Feature engineering and selection was performed throughout the 2 models and include reducing the doc2vec matrices into a single feature, reducing the count vector to a small sub set with high mutual information, and later on omitting the doc2vec features and classifying purely with the subset of count vector. These decisions will be explored and discussed when going though each individual model.

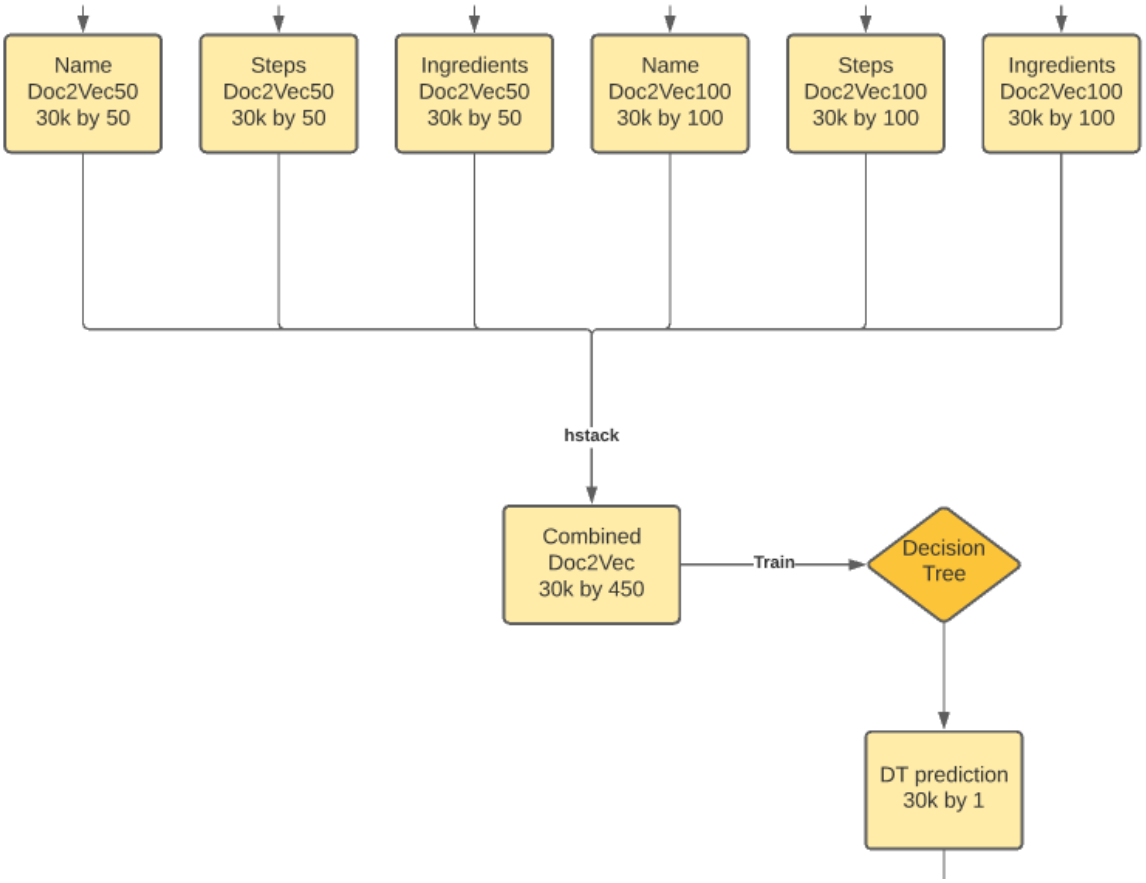
The First Model



Graphical representation of the first model.

This model combines all the features together through feature engineering and feature selection. This combined matrix is then trained on and used in classification. This discussion will be broken down into three sections.

doc2Vec feature engineering.



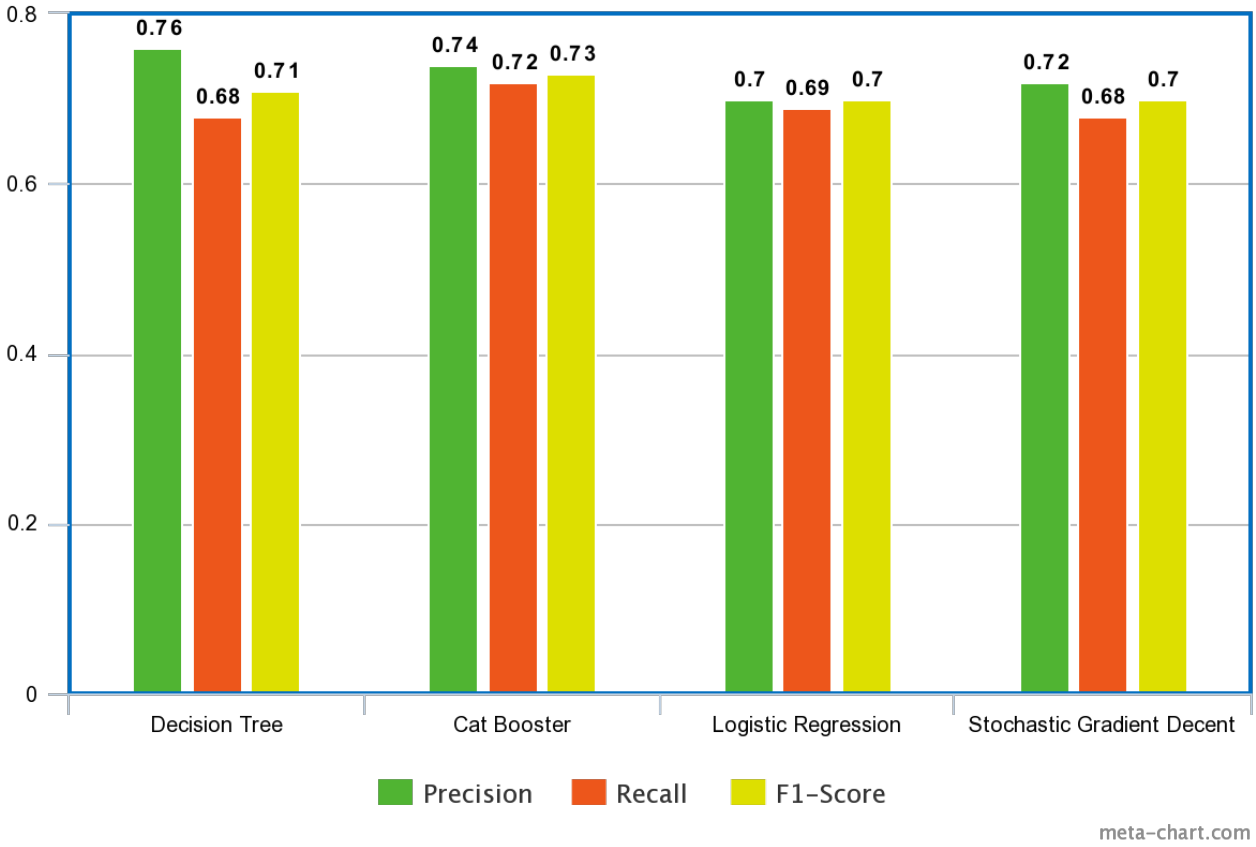
Method

The 6 matrices were stacked next to each other to create a combined feature consisting of 450 features. These features were then passed through a chosen classifier for classification. Finally, the classifier made predictions on both the test and training data and those predictions are used as the engineered feature in the final classification. To find the best classifier for the feature engineering tree based models and logistic regression were trailed and compared based on performance metrics. These include Decision Tree, Cat Boost Classifier, Logistic Regression and Stochastic Gradient Descent. These methods were chosen as they have low computation cost and easy to interpret.

Notes of the classifiers used.

Logistic regression is used in classifying categorical data. It runs a regression using the logistic function to achieve this. Logistic regression requires uncorrelated features which is not true for the data set which is discussed later. **Stochastic Gradient Decent** is a method of optimising the objective function bringing randomness to improve the speed of Gradient Decent and minimising the chance of being stuck at a local maximum. I used **Logistic Regression** as the objective function. **Random forest** is an assemble method which is the weighted average of multiple **decision trees.** This is particular good in reducing the over-fitting ,which may be present in an optimised **decision tree**. The **Cat Boost** Method is a boosting model is also tree based and optimises multi classification with the LogLoss metric. It performs extremely well. (References at end)

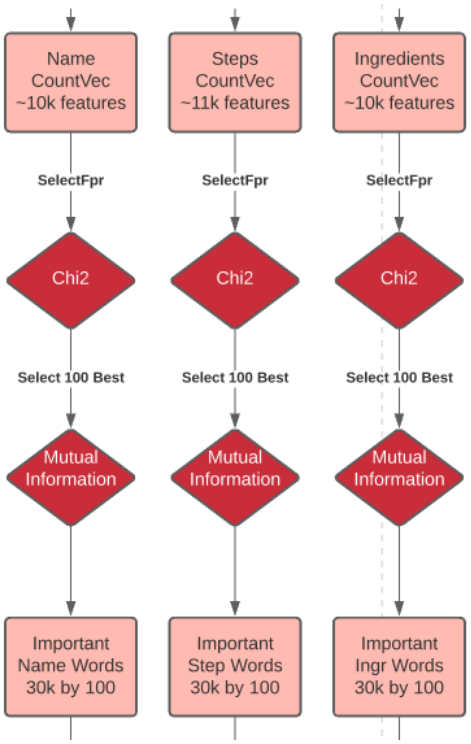
Doc2Vec model metrics



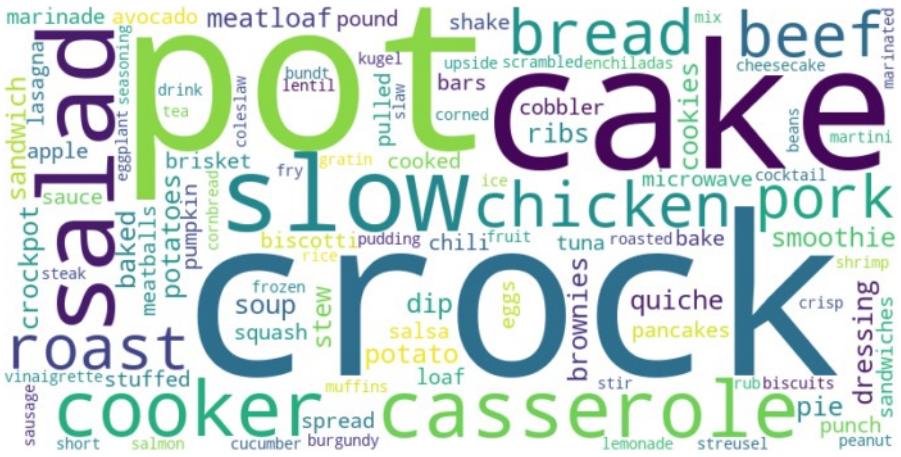
The Decision Tree and the Booster method performed better than the Logistic regression based methods, this may be because a logistic regression struggles to fit non linear decision boundaries which is the tree based models strength. This may suggest a complex relationship between the classes and the doc2vec.

Feature selection of the count factorisation data.

On each of the text features, the top K number of words was selected. This was done by using chi squared and SelectFpr to reduce the number of words to a smaller amount based on which features had a significance level below 0.05. Then select K words with highest mutual information to further reduce the word vectors until they contain only the most important words. This method reduced the computational cost as chi2 works much faster than mutual information. Different values for K were tested and in the final model (parameter turning), top 75 name words , top 175 steps words and the top 100 ingredients words worked the best. The below word clouds visually presents the most important words according to their mutual information score.



Name Highest MI Words



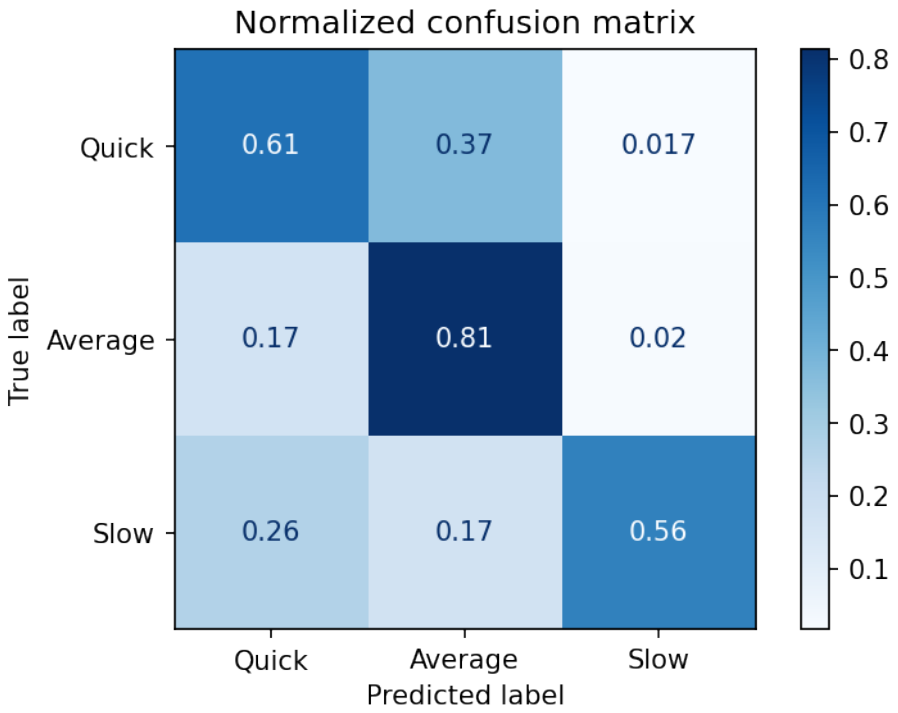
Steps Highest MI Words

Ingredients Highest MI Words

Although these words have high mutual information, it is highly likely that they are also highly correlated with each other and therefore may double up the same information. This can be seen when looking at the selected ingredient words. Ingredients like “flour”, “eggs”, “butter”, “salt”, “baking”, “powder” are all likely to be in the recipes together. This is also seen in ‘name’ feature, with the two highest words “Crook” and “Pot” which are certainly the same recipe. The selection overall performed well which is seen by the steps words “minutes” and “hours” containing the highest mutual information which is to be expected. The method fails to find the most optimal selection of words for classification, but it provides a good approximation. A better less correlated set of words will lead to better classifier predictions.

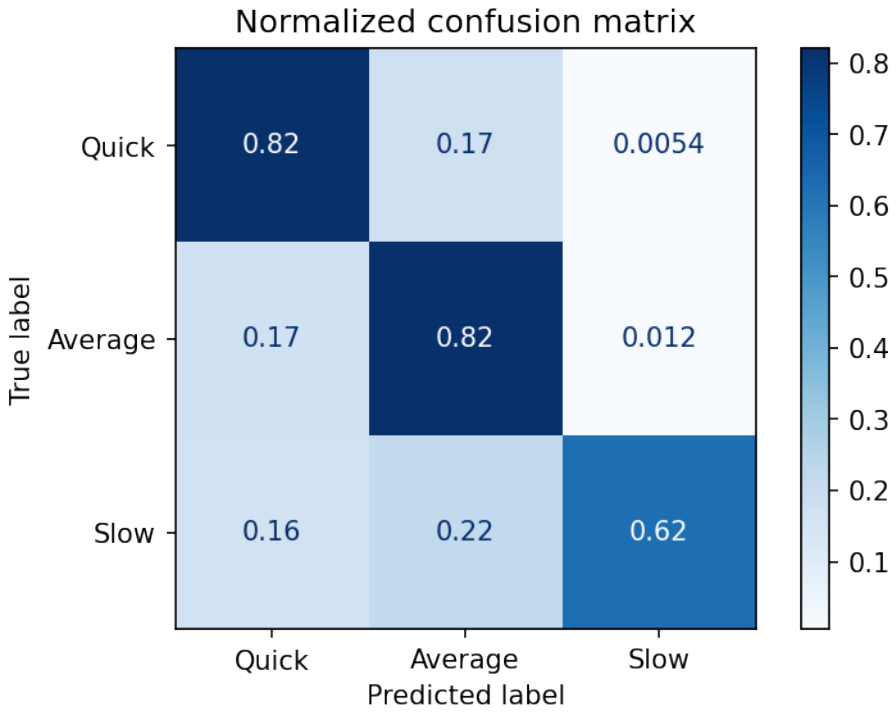
The Model

The doc2vec feature was added to the reduced word count vec with the number of steps and number of ingredients. This was a matrix with approx 353 features. During testing tree based models, which were random Forrest and Cat boost, and logistic regression based models, which were stochastic gradient decent logistic regression and standard logistic regression, were tried. Initial testing showed extremely poor performance with this model which can be seen by the normalised confusion matrices for Stochastic Gradient Decent Logistic Regression. (similar confusion matrix for other classifiers)



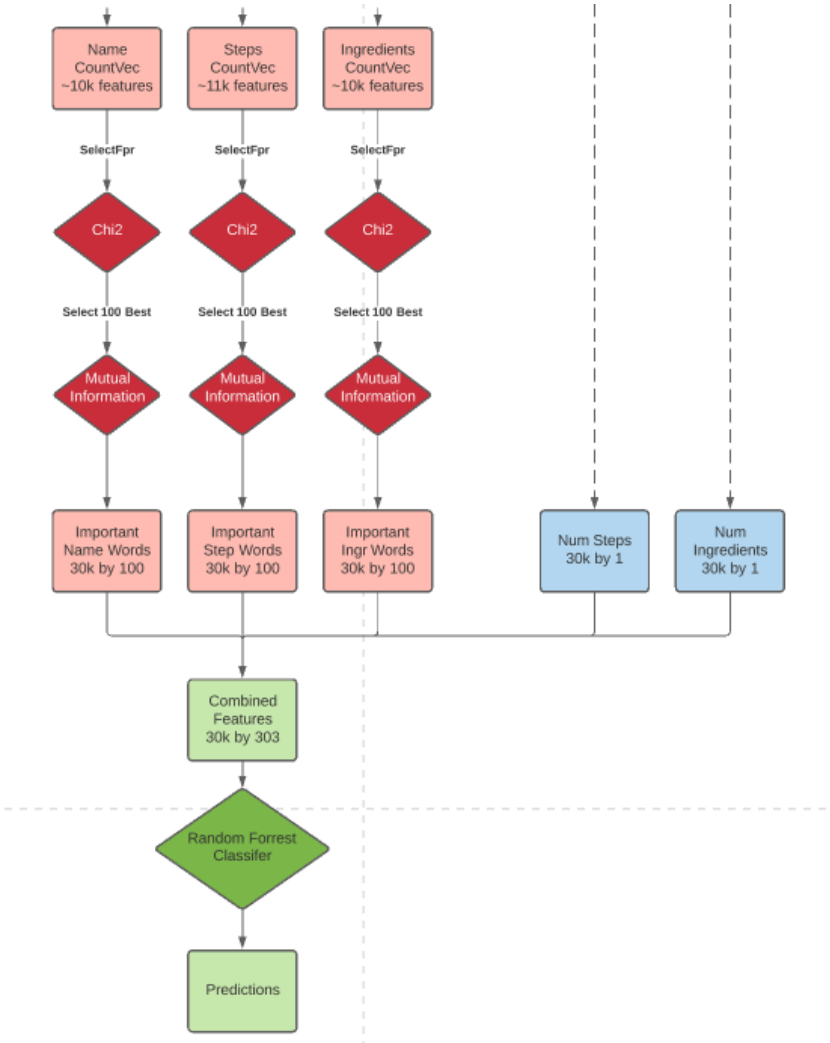
While the average time was predicted to a high level, quick recipes were mis-classified as average and slow recipes often classified as quick or average. This may be caused by the over fitting of the doc2vec into the model. The doc2vec picks up on similarities between text and does not necessarily learn patterns that help classify the duration. The engineered feature may also prevent the models form learning successfully from the count vec as 1 single feature already contains 70% accuracy.

Dropped the Doc2Vec

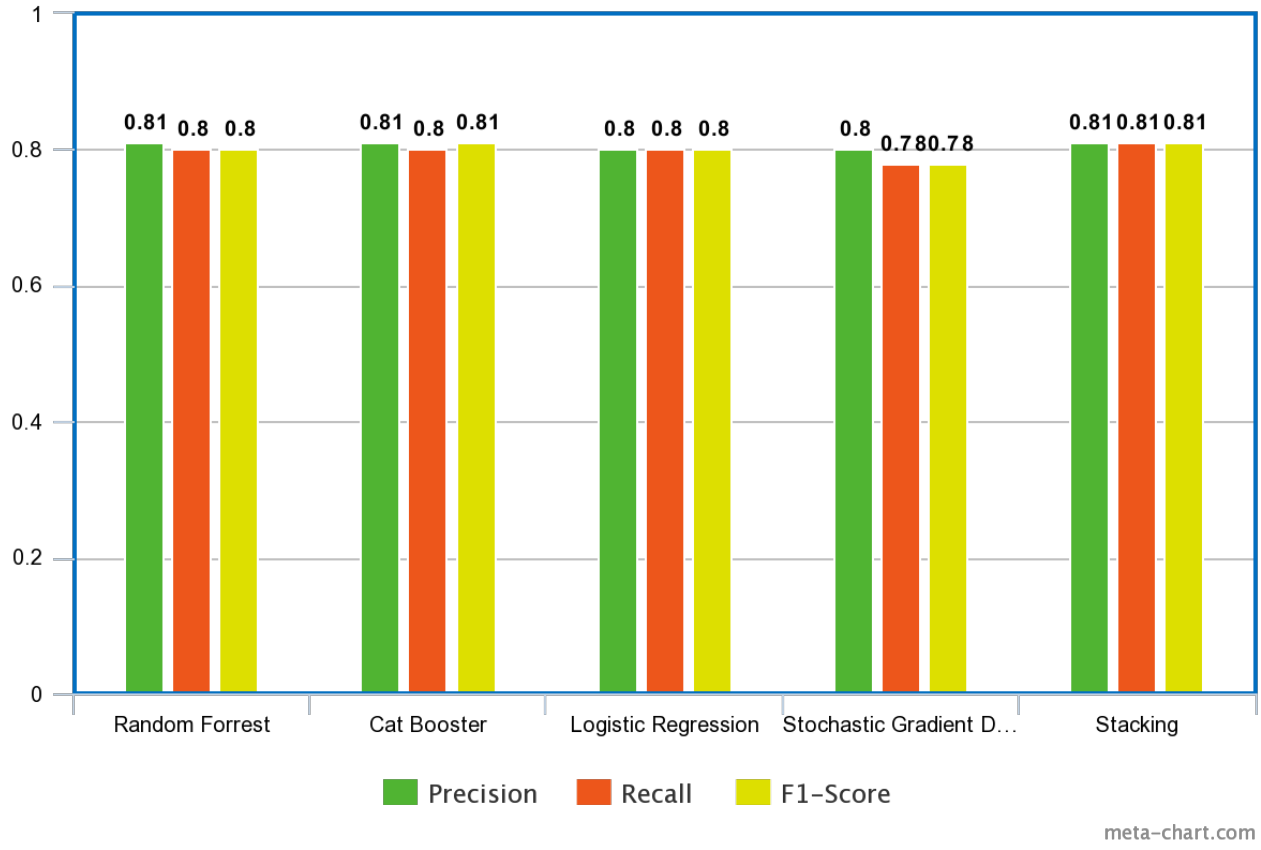


When dropping the doc2vec engineered feature, the quick and slow class saw improved precision and recall with similar improvements across all classifiers used.

The new pipeline.

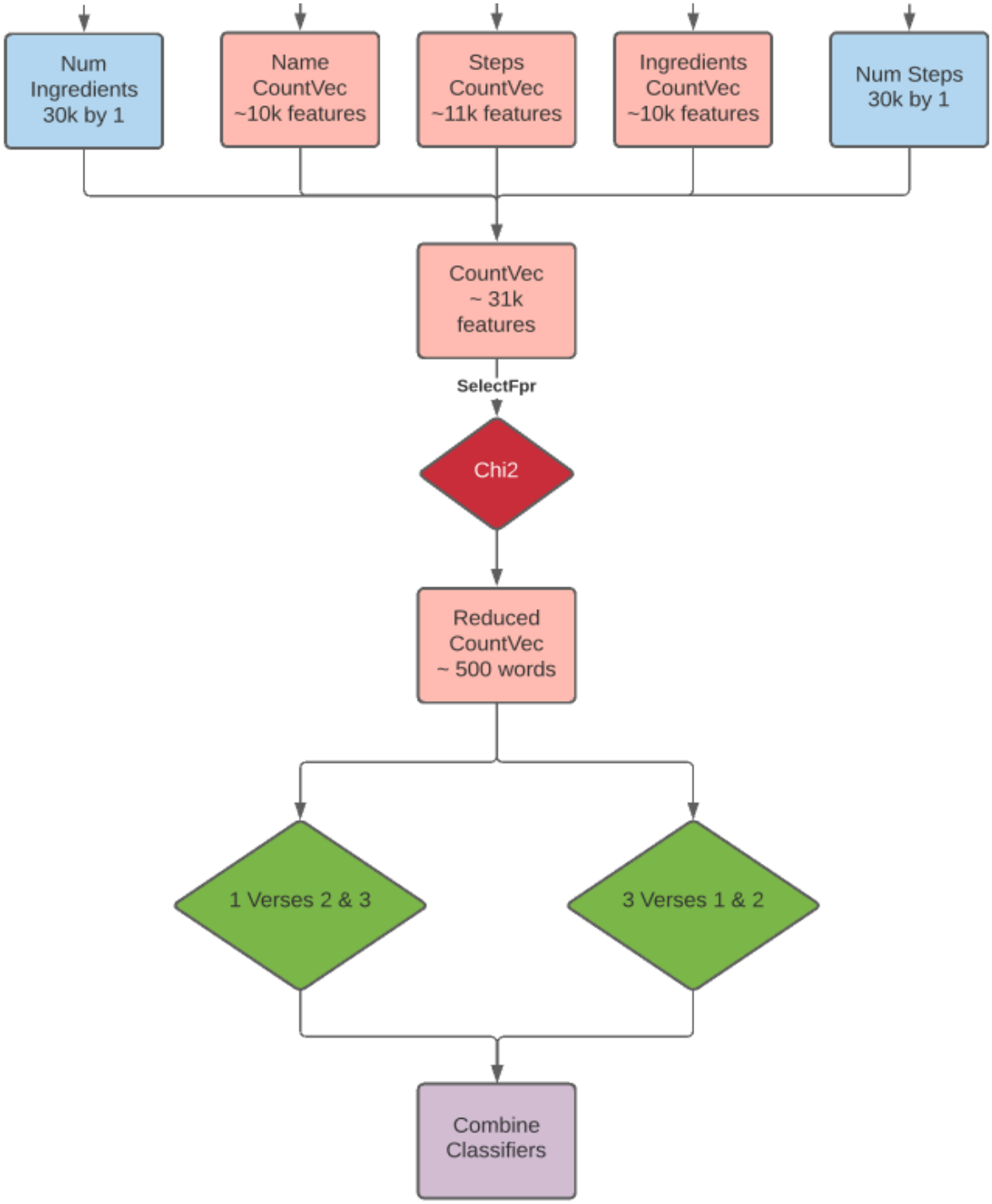


The tree based learning classifiers, performed better but all models performed well which may be due to them having high correlation to each other (95 % of predictions are made in the same way). A stacking model was engineered to stack these 4 models in order to reduce the bias of any particular method. The stacked classifier had very similar performances to the random forest tree and the cat booster models, indicating that these tree models may have learnt all the relevant information that was contained in the word selection.



The Second Model

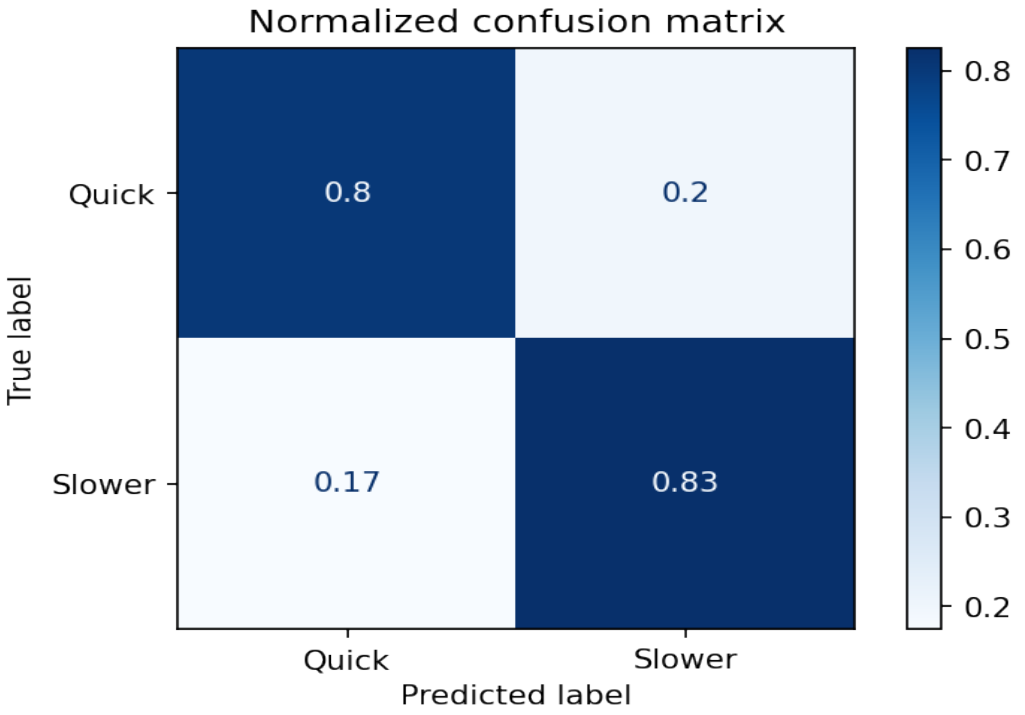
The previous model applied multi classification models which were not aware and did not take into account the ordinal nature of the three classes. Quick, average and slow. The third model incorporates this information.



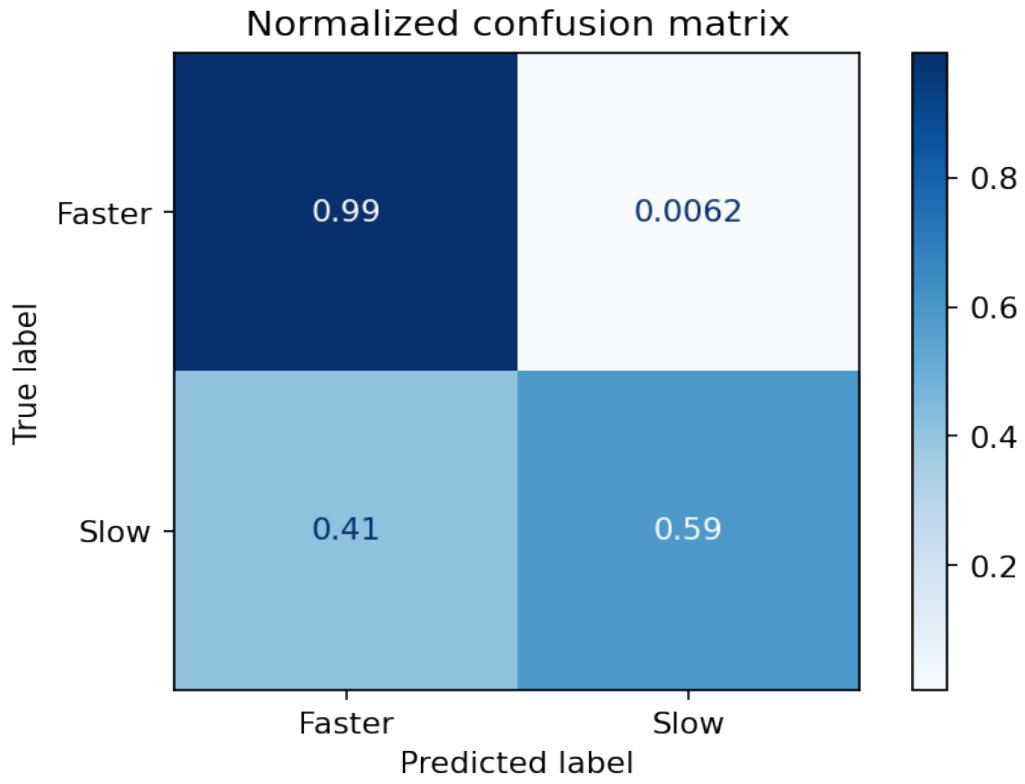
This model consisted of two classifications and a combination. The first model, trains class 1 against classes 2 & 3 performing binary classification. The model creates a boundary between the quick recipes and the average and slow recipes. Likewise, the second model creates a boundary between the slow recipes and the quick and average recipes. The features were also stacked prior to

feature selection to increase model generality. The model performed binary classification, using Cat Boost and Random Forrest, and returned the following metrics.

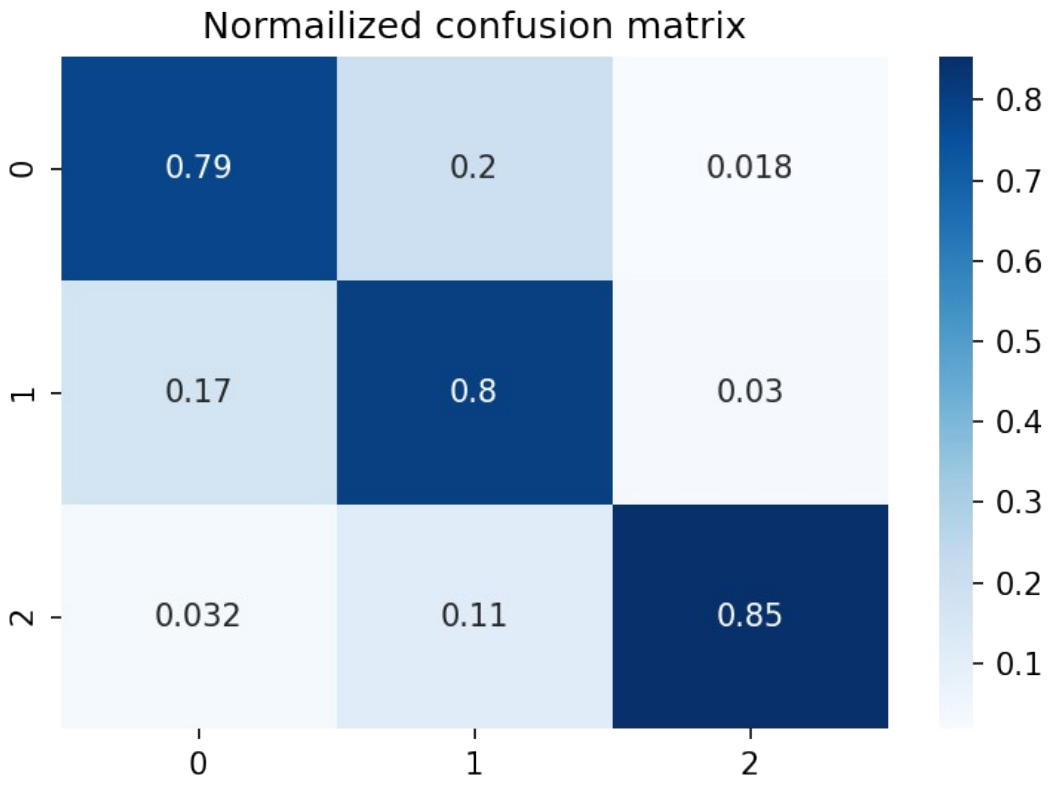
Cat Boost Quick Vs Slower



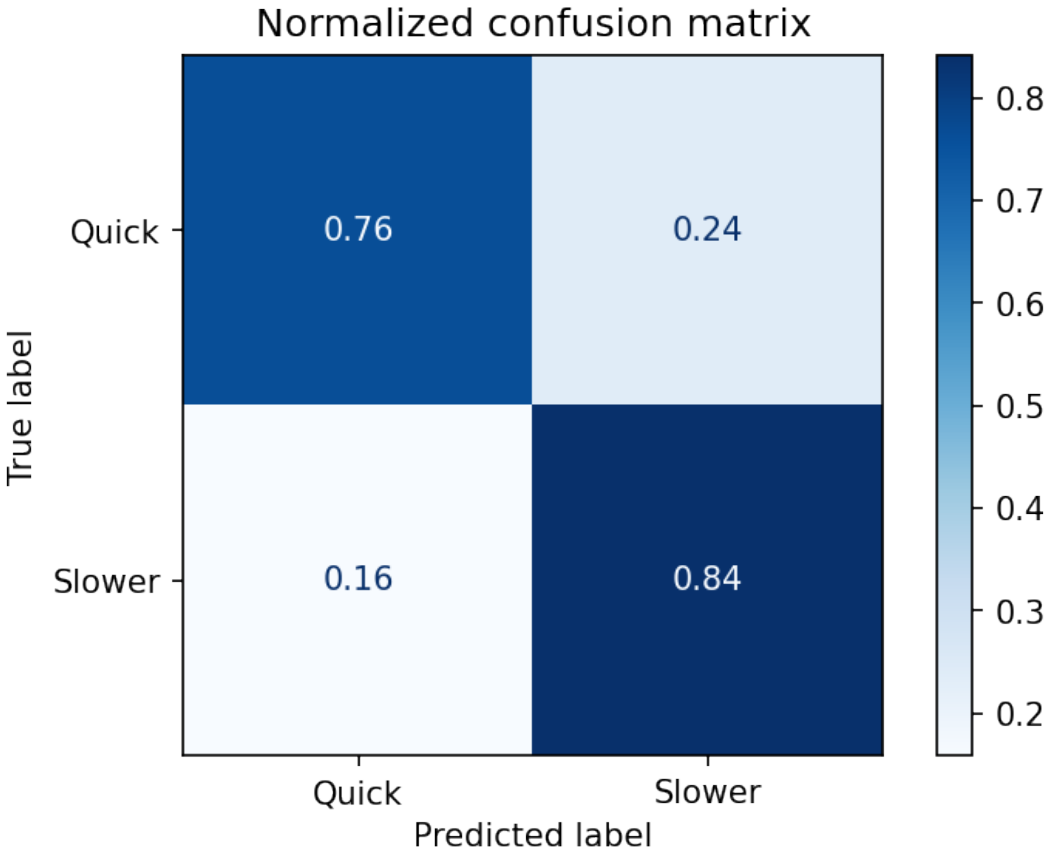
Cat Boost Faster Vs Slow.



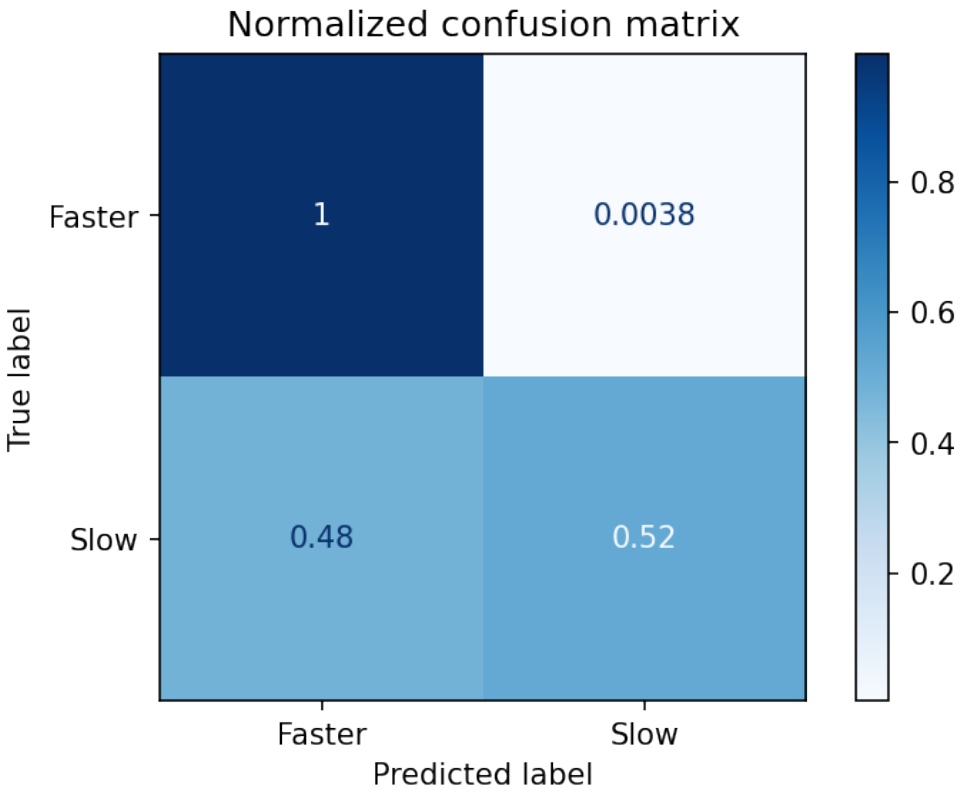
Cat Boost Combined



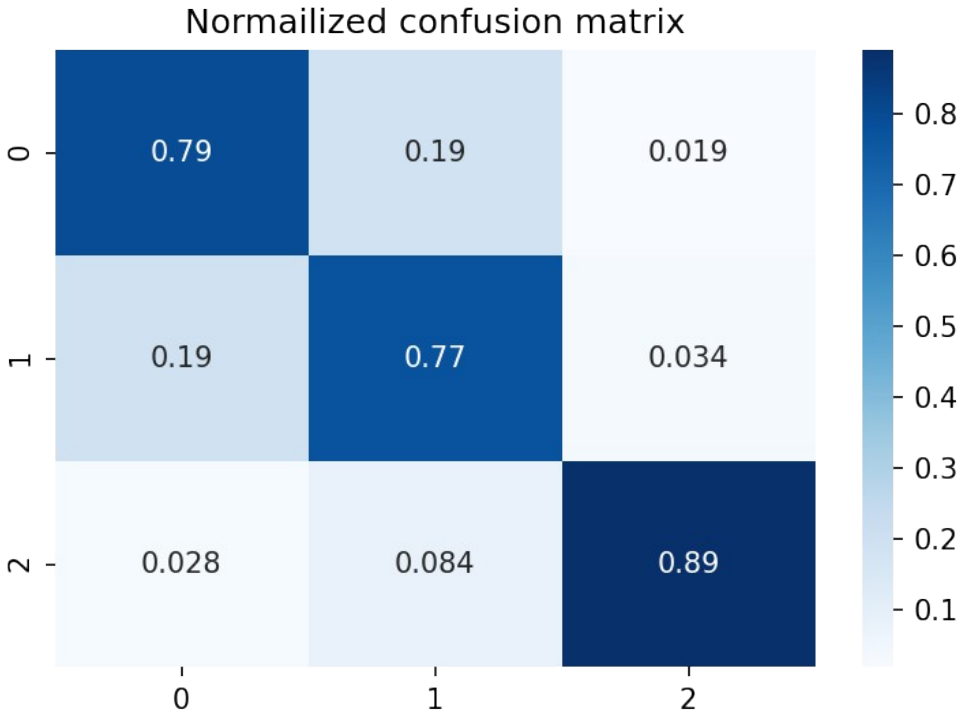
Random Forest Quick Vs Slower



Random Forest faster Vs slow



Random Forest Combined



Combining the two classifiers.

For future improvements, models can be developed which are optimised to find quick recipes and others which are optimised to find slow recipes and combined. The model 2 approach, performed 81.5% on the kaggle test, but fell short of the first models best result of 81.966%.

Hyper parameter tuning.

When using Logistic regression, the parameters were turned using grid searching. The optimal penalty was elasticnet which combines the ‘l2’ and ‘l1’ penalty. (Lasso and SSE) The forest classifiers did not require parameter turning. They n_estimators for the forests, reached a sufficient level at 200 where more n_estimators increased, computation without extra performance. The Cat Boost performed equally well over different tree depths, and set to 8 for computational efficiency.

References:

Generating Personalized Recipes from Historical User Preferences. Bodhisattwa Prasad Majumder,Shuyang Li, Jianmo Ni, Julian McAuley, in Proceedings of the 2019 Conference on EmpiricalMethods in Natural Language Processing and the 9th International Joint Conference on NaturalLanguage Processing (EMNLP-IJCNLP), 2019.

CatBoost - state-of-the-art open-source gradient boosting library with categorical features support, https://catboost.ai/

Stochastic gradient descent – Wikipedia, https://en.wikipedia.org/wiki/Stochastic_gradient_descent

Random forest – Wikipedia, https://en.wikipedia.org/wiki/Random_forest

Logistic regression – Wikipedia, https://en.wikipedia.org/wiki/Logistic_regression