Assignment 2

A. Student

17/05/2021

The Task

In this project we will seek to understand the ability of machine learning models to classify recipe duration. Natural language processing has become possible since computing power expanded dramatically in recent years. It is a topic of active research. Neural network based approaches are one of the most promising routes of achieving human-level semantic understanding. Google's BERT technique its the current state-of-the-art performer on natural language understanding tasks. It relies on neural networks and is pre-trained on over 3 billion words collected on the internet.

Exploratory Data Analysis

Before embarking on any machine learning project it is important to understand the data that one is working with.

The data provided Bodhisattwa Prasad Majumder et. al contains 40,000 recipes, each containing;

- a title
- the number of steps and ingredients
- a list of steps and ingredients
- a duration label - quick, medium, or slow

The purpose of this project is to classify the recipe duration via inference of the other data.

A sample-proportion bar graph was generated to understand the distribution of recipe duration (Figure 1). It is clear that a classifier that always predicts "medium" duration will have an accuracy of close to 50%, this serves as an important benchmark for our classifiers.

Relative frequency histograms were generated for the two numerical attributes, the number of steps (Figure 2) and ingredients (Figure 3).

Quick recipes seem to be more likely to have fewer ingredients and steps than medium and slow recipes, however the distinction with slow recipes is weaker and not in the ordering intuition might expect.

Natural Language Analysis

The `sklearn` word count vectorizer (a Word2Vec implementation) was parsed the raw 'ingredients' data and told to count the frequency of individual words. This generates a large sparse matrix with too high in dimensionality to be feasibly used on a machine learning algorithm. The purpose of this process is to see which words *may* be relevant in modeling the problem. Reduced χ^2 statistics were computed for each word. A large χ^2 statistic indicates the presence of that word had a significant effect on the recipe duration. Figure 4 illustrates this concept, the probability of "beef" being in the instructions depends on the recipe duration label - "beef" has a large χ^2 statistic.

We have established that our features must be statistically significant to be useful, however there is another condition - they must be sufficiently common. If features are infrequent they reduce the density of our feature space which decreases model performance. Thus good features should be both statistically significant

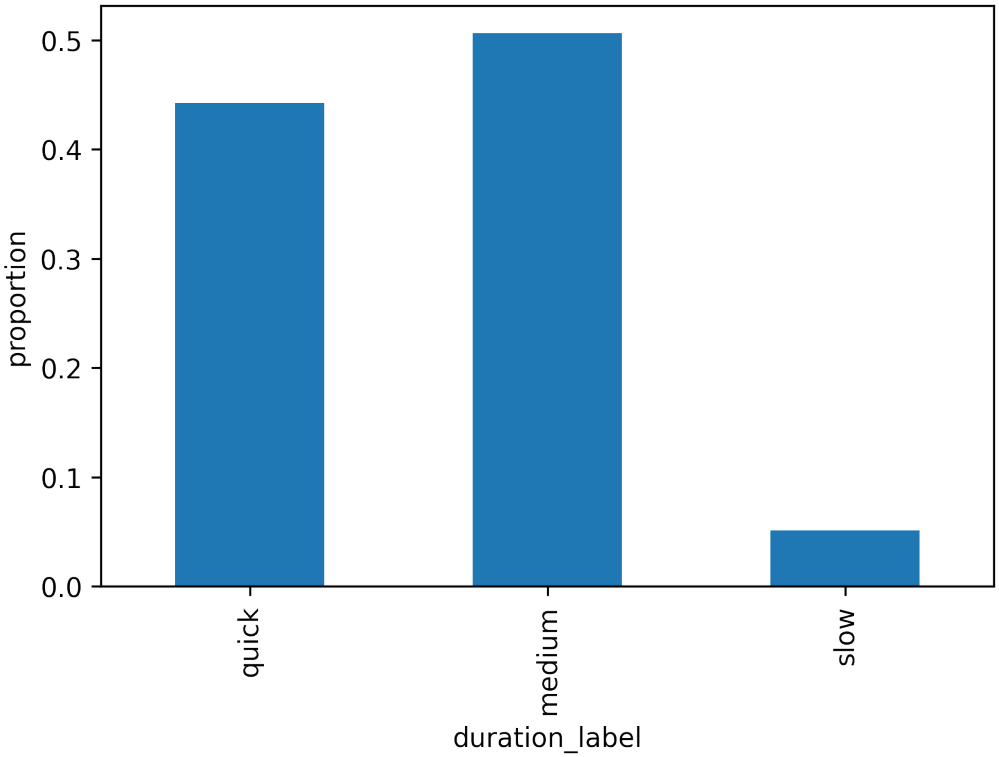

Figure 1: Distribution of Classes

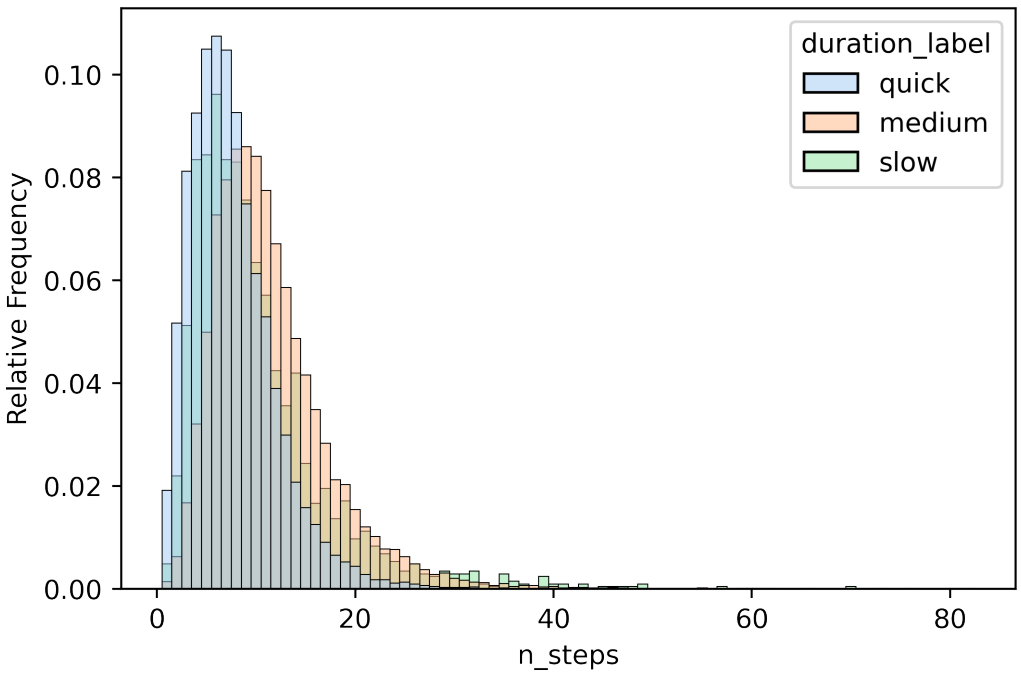

Figure 2: Relative Frequency of Number of Steps

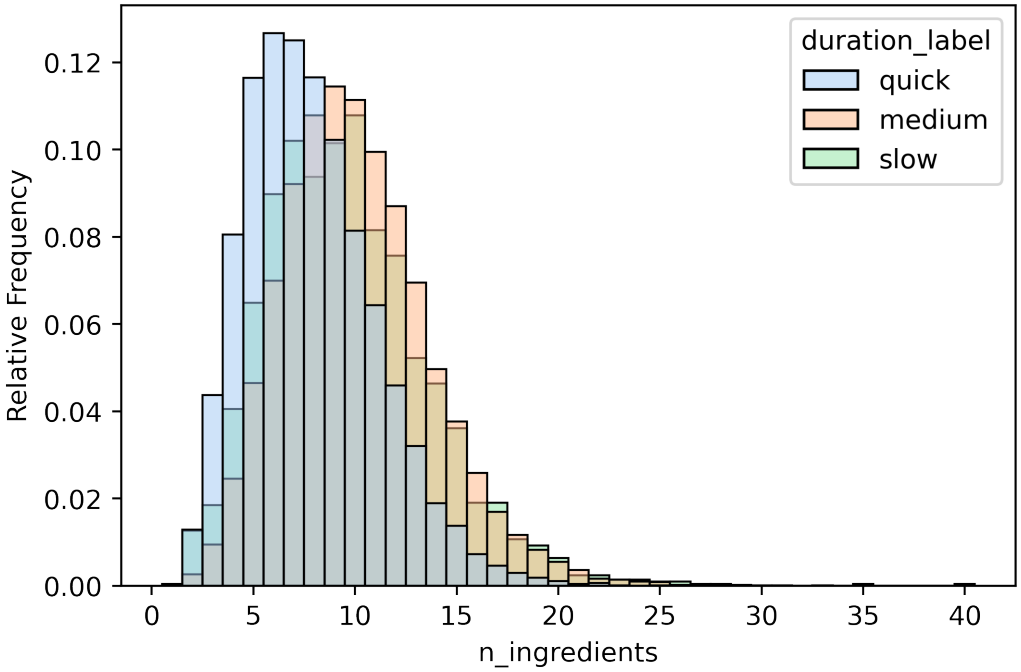

Figure 3: Relative Frequency of Number of Ingredients

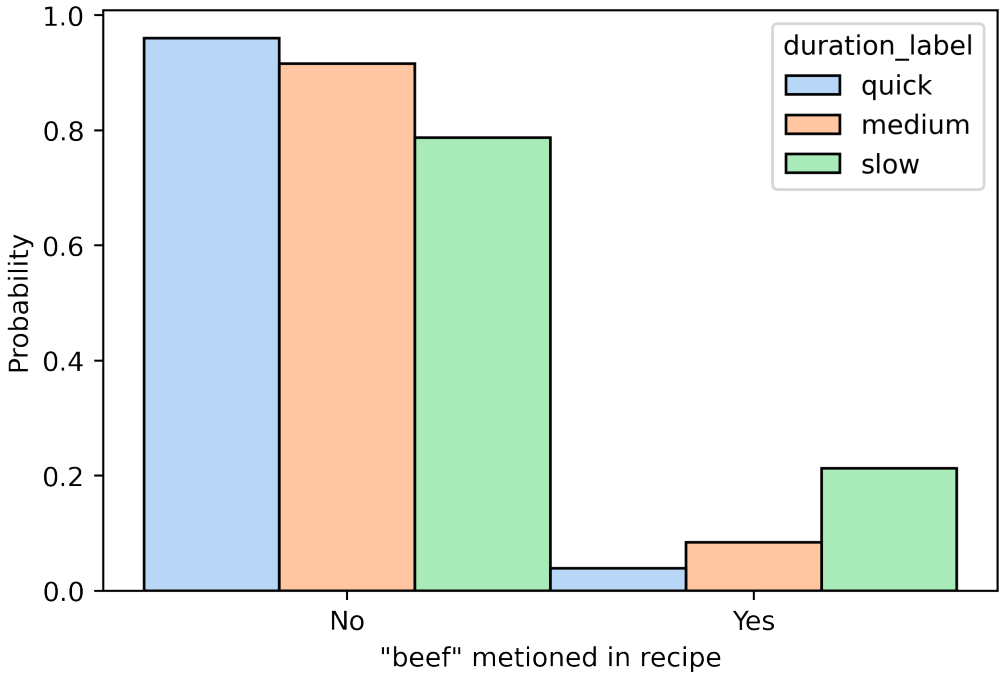

Figure 4: Probability of "beef" found in ingredients

and frequent. Figure 6 contains each feature's (logarithmized) reduced χ^2 statistic and frequency. Figure 7 is identical with features generated on the recipe instructions data. Noting the difference in distributions (density in the upper right) it is apparent that *recipe instructions are more informative than ingredients in predicting duration label.*

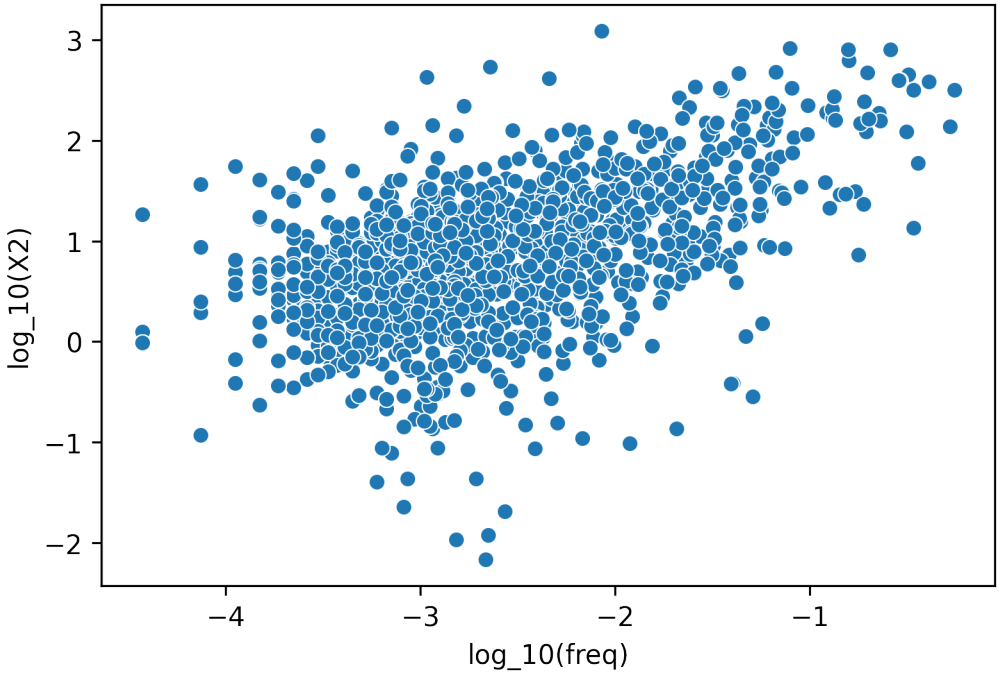

Figure 5: 'words' identified in ingredients Significance - Frequency Plot

n>1-grams In the analysis above we neglected the ordering of words and context around them, for example "preheat oven" is more informative than "oven". Given the scarcity of n>1-grams feature engineering needs to be employed to combine those highly correlated. "preheat oven" was one of the most significant n>1-grams; its effect on class is illustrated in Figure 7. Given the issues of scarcity, n>1-grams were not included in subsequent analysis.

Initial Moddelling

Initially, support vector machine (SVM) classifiers were trained on the instruction and ingredients data (n=1-grams only); optimal performance was found by taking approximately 300 significant and common features. SVM classifiers trained on instruction data had greater accuracy (80% vs 69%). SVM classifiers were used at this stage due to their reputation in handling high dimensional text data. It was decided to merge all the data from the title, instructions and ingredients before passing to the CountVectoriser, this has the advantage of increasing the information the model sees for each recipe while at the cost of some contextual information. This process resulted in 15,300 unique 'word' features. Models trained on this combined data had poorer performance than those trained on just the instruction data - this was likely a result of the different features selected. In subsequent analysis just the instruction data was used for modeling.

Feature Engineering, Reduction & Word Embeddings

Features were selected on a sequential ordering approach, filtering by frequency and subsequently χ^2 statistic. In the final models, 300 of the most significant features of the 1200 most frequent features. This approach has two obvious issues,

- omits significant but uncommon features

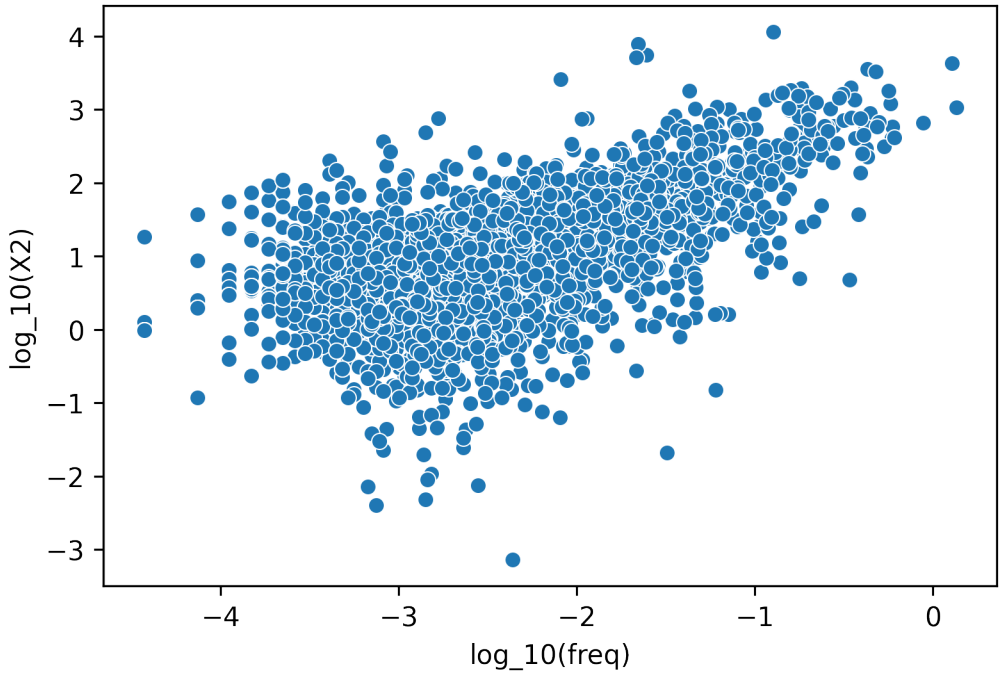

Figure 6: 'words' identified in steps Significance - Frequency Plot

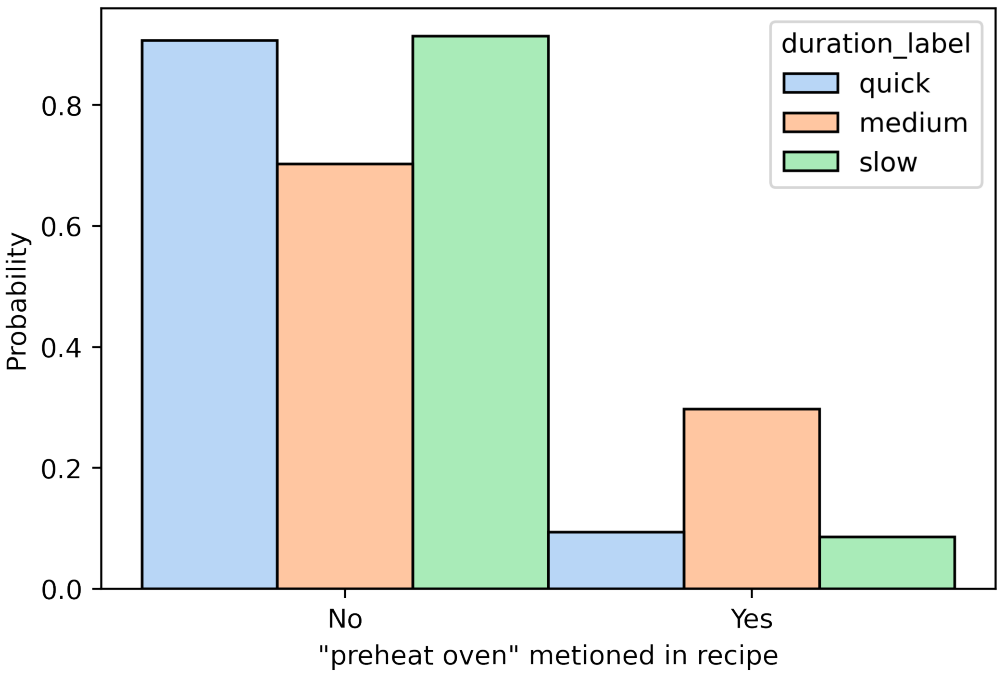

Figure 7: Probability of "preheat oven" found in instructions

- results in a high dimensionality

To ensure these parameters did not remove the highly significant features, the features were again scatter-plotted post-selection. To address the high dimensionality features engineering techniques were investigated. Applying principal component analysis reduced the training time but also reduced the performance. Given the ability of SVMs to handle high dimensional data and best performance, feature engineering was not employed in the final models. s This area could be considered the weakest part of this project. There is scope for better feature selection and dimensionality reduction. I would be hesitant in using word embeddings that were not trained on food data.

t-distributed stochastic neighbor embedding visualisation

t-distributed stochastic neighbor embedding (t-SNE) is an embedding technique useful for visualising high dimensional data in two or three dimensions. A random selection of 1200 recipes were selected for this algorithm. There does appear to be some clustering of the 'slow' recipes, however there is less clustering of the 'quick' and 'medium' classes.

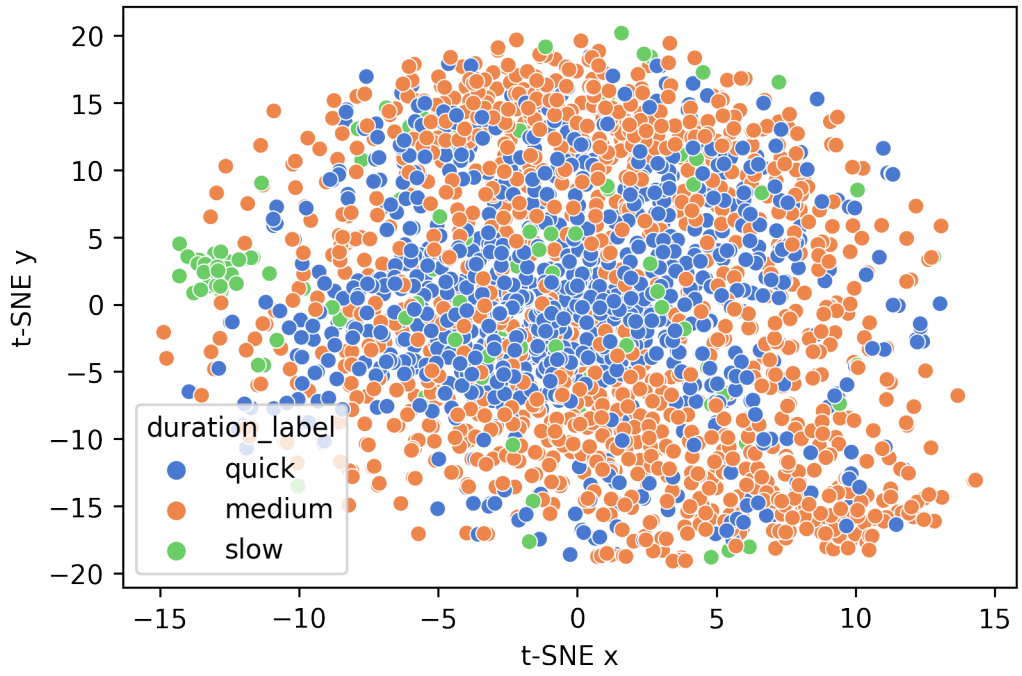

Figure 8: t-SNE

Classifier Evaluation

The following table contains important benchmarks and classifier metrics. Precision, recall and f1-score were calculated using the class-weighted averaging technique.

```
performance_data
```

```
##      Unnamed: 0  precision    recall  f1-score  support  accuracy
## 0        zero-r   0.257710  0.507652  0.341870    13200  0.507652
## 1         one-r   0.640999  0.642803  0.623335    13200  0.642803
## 2  Bernoulli NB   0.738851  0.734394  0.735204    13200  0.734394
## 3    Linear SVC   0.795003  0.792500  0.791809    13200  0.792500
## 4       rbf SVC   0.804598  0.801515  0.800599    13200  0.801515
```

Naive Bayes (NB) Classifier

The naive bayes classifier relies on the assumption of independent features. Given features were not selected for independence, this is an unlikely assumption. Despite this, remarkable performance was found with Bernoulli Naive Bayes trained on binarized data (turning the word frequency vectors into word presence vectors). Here, the presence of each word in recipe steps is modeled as a Bernoulli trial, conditional on the duration label. Gaussian distribution assumption is not appropriate for the (discrete) word frequency data. Given this classifier's simplicity, its performance is impressive.

Linear and Non-linear SVM Classifiers

The solving algorithm failed to converge given the default parameters. It is likely that the data is not *linearly seperable*. Increasing the normalization parameter and adjusting the tolerance allowed the solver to converge. Applying the *kernel trick* can make data linearly separable in the kernel image space. A radial basis function (rbf) was applied to the data before training the model, this resulted in better overall performance. Binarised data was found to improve performance on the SVM classifiers, and used in the evaluation here.

Comparison

Confusion Matrices are attached for the three models we looked at, Figure 10,11 & 12. All models struggle to identify 'slow' recipes. Figure 13 shows the relative frequency of the different model predicted classifiers on test data. Since the Bernoulli Naive Bayes model distribution is closer to the true distribution, it appears to suffer from *less bias* than the SVC. Since the Naive Bayes model calculates probabilities directly, its distribution better matching the true distribution is unsurprising. Linear SVC was omitted as it was similar to the rbf SVC.

The best performing models on Kaggle achieve accuracy around 82%. Our best performing model, rbf-kernel SVM, achieves accuracy of 80% on 10-fold-cross-validated data. These numbers are comparable, which is good. They do however suggest there is more information in the data to infer the class labels. This agrees with our analysis, as we did not train the model on the numeric features (number of ingredients & steps) which appear to have non-identical distributions. Furthermore, we did not analyse the title data or include the ingredients data in our final models.

Conclusion

In this project we were able to derive interesting insights within the data. Figures 4 & 7 highlight the probabilistic effect of a few key words commonly found in recipes. We found thousands of statistically significant effects that recipes can contain, see Figures 5 & 6. A key part of machine learning is distilling those effects down to the most important ones, arguably the most challenging part of this project. We investigated feature reduction techniques such as PCA, but did not find them useful. The t-SNE algorithm showed some degree of clustering (at least with the "slow" recipes.), this suggests utility in more advanced neural network embedding techniques such BERT. Our final models showed cross-validated accuracy above 80%, and scores within the top 50 models on Kaggle of 81.5%, and a significant improvement upon the 50% of zero-R.

References

Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing, Jacob Devlin, Ming-Wei Chang, Google AI Blog

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.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

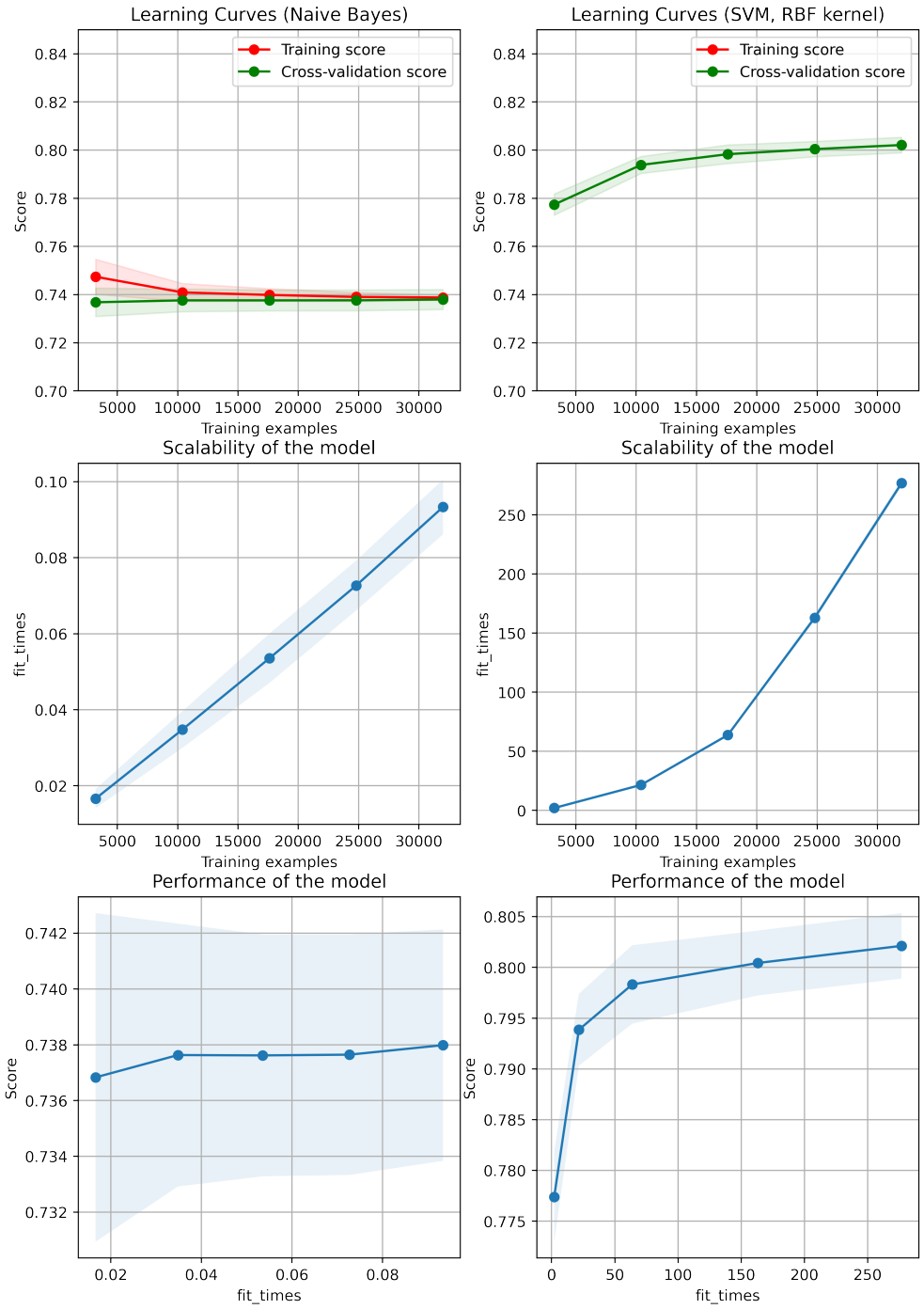

Figure 9: sklearn Learning Curves

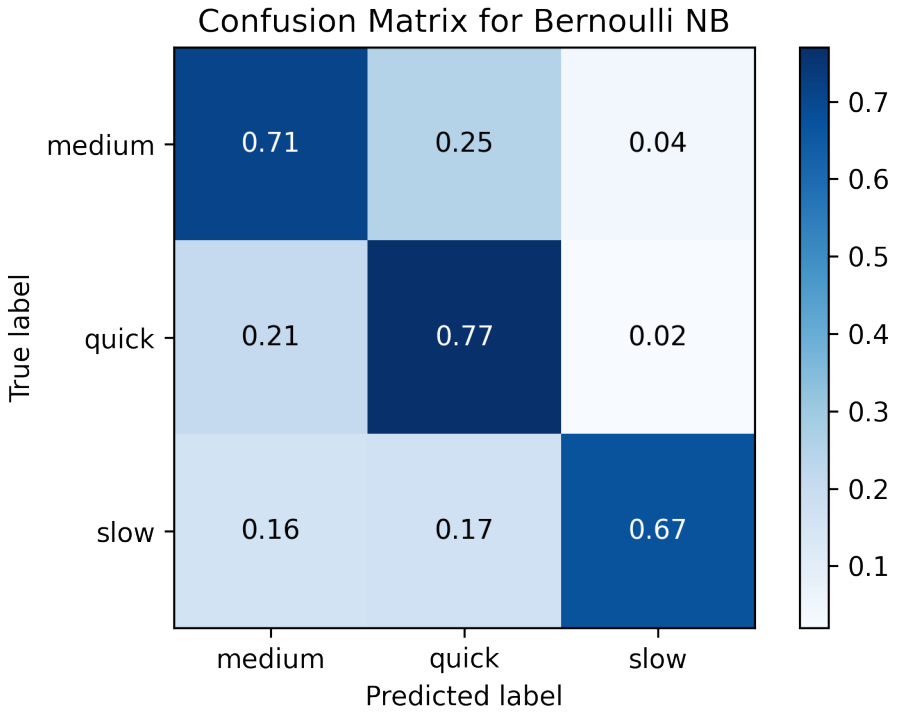

Figure 10: Bernoulli NB Confusion Matrix

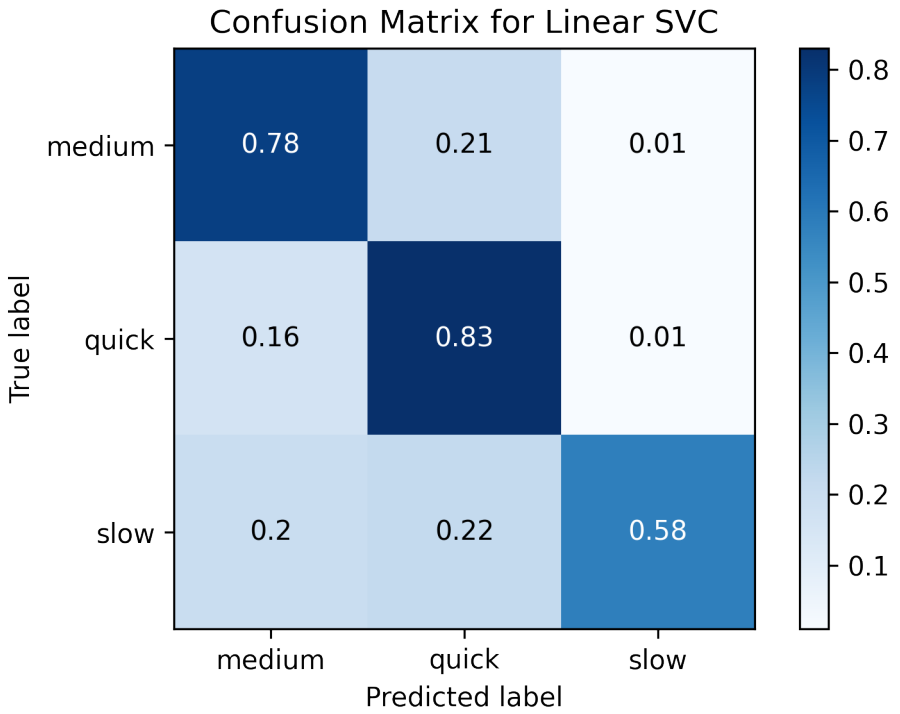

Figure 11: Linear SVM Confusion Matrix

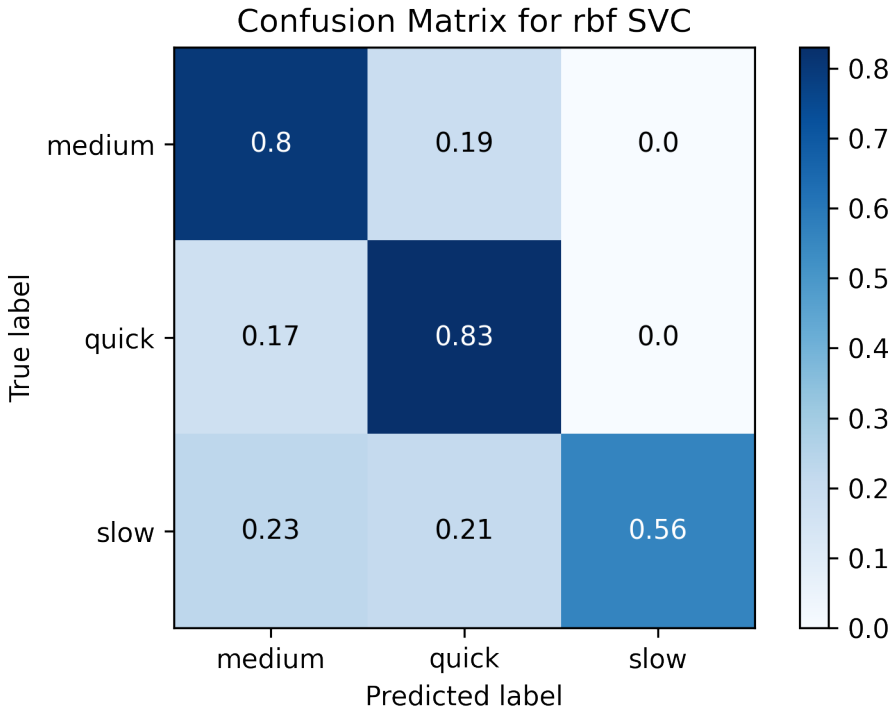

Figure 12: rbf SVM Confusion Matrix

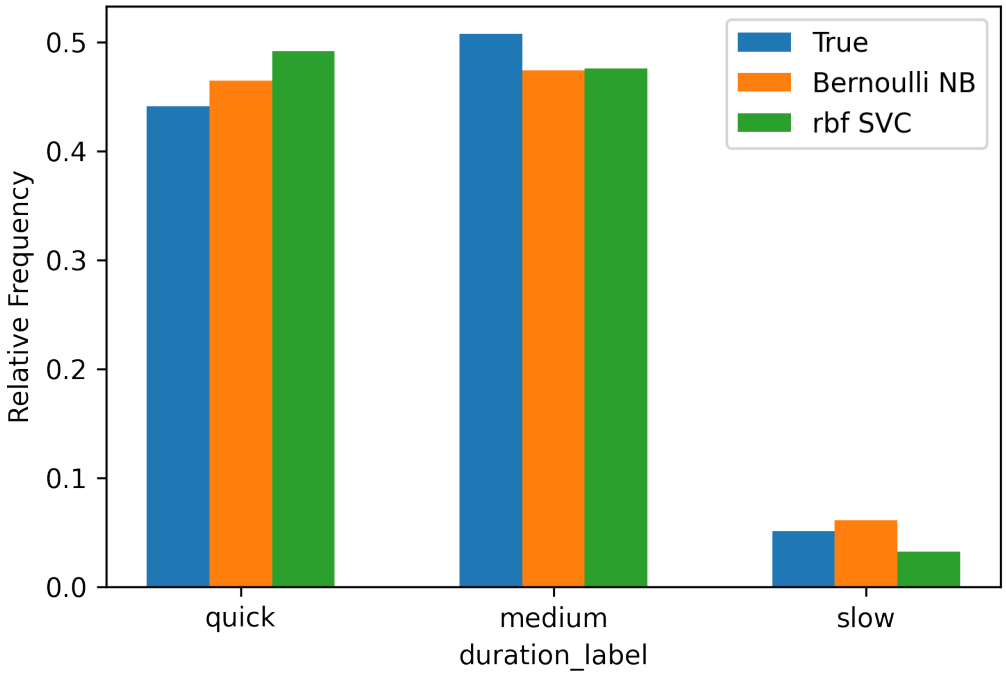

Figure 13: Model Bias Comparison

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2. (Publisher link)

J. D. Hunter, “Matplotlib: A 2D Graphics Environment”, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.