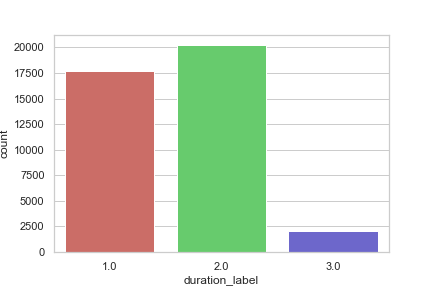
**COMP 30027 Report**

**Anonymous**

**1.** **Introduction**

With the major and vast development of the World Wide Web and big data, there are various machine learning models that could be applied to different kinds of datasets available online. In this project, we are going to classify how long it takes for a dish to cook using the datasets provided by Foods.com (Majumder et al., 2019). To accomplish this, 5 machine learning models are implemented, namely Naive Bayes, Decision Tree, Logistic Regression, Ensemble Stacking and Neural Networks. In this report, these models are compared and contrasted as well as the underlying reasons behind why these machine learning models work are discussed.

**1.1. Data Exploration**



**Figure 1-** number of instances for specific class labels (1,2,3)

In the training set provided to predict cooking duration, it can be seen from the data exploration graph that there are approximately 20,000 instances that have medium cooking time, around 17,500 data that takes little time to cook, and only 2,400 instances that take the longest time to cook. Here, there is a data imbalance with ratio of 44:50:5.

**2.** **Feature Engineering**

To classify class labels given the datasets (Majumder et al., 2019), features are important in the learning process. In this project, names, ingredients, steps, number of ingredients and number of steps are the given features. The first 3 features are text fields which could be considered fairly sparse to learn from. Therefore, preprocessing and data selection was done before doing the model evaluation.

**2.1 Data Preprocessing**

Before building our model, these text features should be cleaned and preprocessed in order to achieve a better performance. Data preprocessing means doing data manipulations in a way that makes it more effective while building a model, and this is done by removing less important features.

Although in the doc2vec50.csv file, the name, steps and ingredients of the recipe are provided, these only represent 50 dimensions extracted from the training and testing, which could exclude useful information in the modelling process. Therefore, we decided to use the raw data set and process them on our own to obtain better results.

**2.1.1 Punctuation Removal**

Punctuation may not have direct importance when it comes to analyzing the data.

**2.1.2 Stop words Removal**

Some words in the tokenized text do not necessarily give a significant result. Thus, it is better to remove them beforehand.

**2.1.3. Lower Casing**

Data is converted to lowercase for consistency.

**2.2 Feature Selection**

Initially, the recipe\_train.csv dataset is split into a train and test set with ratio 80:20. This is because we don't want our model to overfit by seeing the test set before actually testing it as it may bias the model evaluation. Moreover, it is set to be 80:20 because it is preferable for the model to learn generally enough data balanced with enough testing set to avoid overfitting.

Secondly, raw data text in each feature in the train set is vectorized separately using Count vectorizer such that a set of strings is converted into a frequency representation using the sk-learn library. A large distinct number of term document matrices are obtained for name, ingredient and step in the train set. Since the number of features are extremely large, the experiment in the training often takes longer time to compute and is computationally heavy. Therefore, feature selection such as select K best technique is used to transform the high dimensionality of the dataset to aid in choosing the features that contribute the most to the target variable. Using this method, we are able to choose the best predictors for the target variable. There are two scoring functions that we have experimented with, this includes Chi-Square and Mutual Information. However, it takes longer for MI to compute as it tries to measure the amount of information that one random variable contains about another random variable, so we used Chi-Square. 1000 best attributes are obtained using Chi-Square. Then, these are fit and transformed into corresponding data representation. The same feature selection is implemented in the test set as well. Once the feature selection is done, our columns for training and test set are concatenated, creating a new data frame with features of names (tokenized), ingredients (tokenized), steps (tokenized), number of ingredients and number of steps to do the prediction.

**3. Class Evaluation**

In this section, several classifiers are used over the recipe data to evaluate our performance along with tuned hyperparameters in order to choose the best performance for our data (Majumder et al., 2019)

**3.1 Hyperparameter Tuning**

Hyperparameter optimization or tuning are methods used to optimize the prediction results. Our grid search algorithm is measured by cross validation on the train set or evaluation on a held-out validation set.

**3.1.1 Naive Bayes**

Naive Bayes is useful especially in large data sets such as text classification. It is reasonably fast to predict the class of the test data set. There are three different types of Naïve Bayes Classifier, such as Gaussian Naive bayes (used in classification and assumes that the feature has a normal distribution), Multinomial Naïve Bayes (used in discrete counts) and Bernoulli Naïve Bayes (used mostly in text classification with “bag of words” model where 0 means “word does not occur in the document” and 1 means the “word occurs in the document” (Jaya, 2019). On the other hand, using Grid Search with various combinations of parameters, the best parameters for use are presented below.

|  |  |  |
| --- | --- | --- |
| Naive Bayes Classifier | Best hyperparameter | Accuracy |
| Bernoulli | c = 10 | 75.3% |
| Multinomial | c = 10 | 73.4% |
| Gaussian | var\_smoothing = 1e-09 | .64.75% |

**Table 1-** Naïve Bayes Classifier’s Accuracy when compared between Bernoulli, Multinomial and Gaussian Table 1.

Out of all Naive Bayes Classifiers, Bernoulli NB algorithm is shown to be the best approach for text classification tasks. Multinomial NB seems to have almost the same accuracy as Bernoulli. However, since multinomial works with occurrence count whereas Bernoulli works with Binary/Boolean features. Thus, a slightly different accuracy is generated. On the other hand, Gaussian NB only gives 64% of accuracy which means it is not really suitable to predict text classification problems. This algorithm is mainly used when all features are normally distributed, and all features have continuous value. Thus, only Bernoulli and Multinomial Naïve Bayes perform well on text-processing data. Bernoulli Naïve Bayes works best here because our assumption is that the features are binary.

**3.1.2 Decision Tree**

Decision Tree is known for its robustness to noise, handling of irrelevant, low computational cost and also fast run time. This classifier uses impurity to split the nodes with the internal nodes representing the conditions and the leaf nodes representing the decision based on the conditions (Abhijit Roy, 2020). The deeper and the more splits it has, the better accuracy we obtain as it collects more information.

When fitting the decision tree with max-depth ranging from 0 to 10, it was found that the max depth=10 results in a higher accuracy compared to other parameters for our model.

We hypothesized that Naive Bayes and decision will generate similar accuracy (Huang et al, 2006). We expect Bernoulli Naive Bayes to have higher performance because it is considered as one of the most common techniques to handle text classification.

After conducting the experiment using Decision Tree with tuned hyperparameters, it results in 75.23% accuracy. The Decision Tree and Naive Bayes models show similar results in terms of accuracy, with Decision Tree’s performance to be slightly lower than Naïve Bayes. Decision Trees gives more meaningful insight in how the model works. However, their power to learn complex data sets is limited and also to scale on a large dataset (Code mentor blog, 2020). Moreover, Decision Trees are prone to overfit if the hyperparameters are not tuned well.

**3.1.3 Logistic Regression**

Compared to Naïve Bayes and Decision Tree models, Logistic Regression is the most well-performed classifier. Logistic regression is also suitable for frequency features. Using the Logistic Regression in predicting our data with Chi-square Select K-Best feature selection, we obtained a prediction accuracy of 79.1%. To explain the underlying reason behind this accuracy, the ordinal logistic regression model requires the dependent variable to be ordinal. Second, logistic regression requires the observations to be independent of each other. In this case, our data are assumed to be independent since the Count Vectorizer data is widely, leading to the features being able to be assumed independent. In other words, the observations do not come from repeated measurements or matched data. Another assumption is that the Logistic Regression model requires fairly large sample sizes, and since the dataset contains 40,000 instances, it becomes more reliable for prediction analysis.

**3.1.4 Multilayer Perceptron**

Before fitting the train set to our model, standard scaler is applied. Although the accuracy decreased slightly by less than 1%, the prediction results are more reliable and have higher interpretability since Multilayer Perceptron is sensitive to the scale of input attribute values. Because our dataset is not linearly separable, we could not reach the absolute 100% accuracy. However, we set the threshold to be 78% which is reasonable.

In our Neural Network model, there are several hyperparameters that can be tuned. One of the most important hyperparameters is the learning rate (Gandhi, 2018). It is best practice to use exhaustive Grid Search or other automated tuning methods. Here, we decided to use learning rate (alpha) = 0.01 using trial and error method because of computing power limitations. We also chose the ReLU function because it is one of the most widely used activation functions, as it solves the problem of vanishing gradients (Gandhi, 2018).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| 1 | 0.75 | 0.77 | 0.76 | 4046 |
| 2 | 0.79 | 0.78 | 0.78 | 4695 |
| 3 | 0.73 | 0.63 | 0.68 | 459 |
| Accuracy | - | - | 0.77 | 9200 |
| Macro average | 0.76 | 0.73 | 0.74 | 9200 |
| Weighted average | 0.77 | 0.77 | 0.77 | 9200 |

**Table 2-** Evaluation matrix (Precision, Accuracy, Recall, F-Score) using Neural Networks model.

From the evaluation matrix above, it can be seen that the macro averaging and weighted averaging differ only slightly, which is a good indication. However, the individual F-1 score and recall scores for class 3 is significantly lower than the other classes, this is due to the data imbalances we found in data exploration. Further methods, such as *stratified sampling* can be used to verify the class labels when train-test splitting is perfectly balanced.

**3.1.5 Stacking**

Ensemble method or stacking is a model which combines multiple machine learning models to predict our dataset model and designed to bring an improvement to our prediction. It involves a base model which is the models that fit on the training data known as level 0 models and a meta model that combines the prediction of the base models known as level 1 model (Jason, 2020). After multiple trials and errors in finding the base model, we have chosen Logistic Regression, Decision Tree and Binomial Naive Bayes as our base model because they have a diverse range of models with different assumptions on the learning phase. On the other hand, we use a simple meta model which is Logistic Regression for the classification task to reduce computational cost and easier interpretation. Our approach to prepare the training data set is done using 3-Fold Cross Validation. After running our experiment, we obtain an accuracy of 79.97% which results in the highest performance out of all classifiers.

|  |  |  |
| --- | --- | --- |
| Level 0 | Level 1 | Accuracy |
| Logistic Regression Decision Tree  Multinomial NB (alpha = 10) | Logistic Regression | 0.802 |
| Logistic Regression Decision Tree  Bernoulli NB (alpha = 10) | Logistic Regression | 0.80 |
| Logistic Regression  Decision Tree (max depth=10) | Logistic Regression | 0.77 |

**Table 3-** Ensemble’s method using different base models in obtaining the best performance.

**3.1.6 Chosen Model for Validation Test**

For validation test submission in Kaggle, we decided to use the entire recipe\_train.csv to train our model instead of train test splitting because train test splitting may reduce the information learned by the model. Our previous method in train-test splitting the data is done to figure out which model gives the best performance in terms of the evaluation scores, and we find that ensemble stacking is the most accurate model to predict our dataset. Thus, with applying this ensemble method without train test splitting the data, our submission showed 81.266% of accuracy. Through this, it can be concluded that the validation test results are consistent with the evaluation test mentioned in 3.1.5.

**4. Conclusion**

In conclusion, we have tried various models such as Logistic Regression, Decision Tree, Naive Bayes Classifiers, Neural Networks and Ensemble Stacking to predict and analyze our dataset. The best accuracy we found was found by the Stacking method, although the results are almost similar to Logistic Regression.

In contrast, we have tried several other models, such as K-NN and Mutual Information for the feature selection. However, we failed to reach an exact conclusion in the training phase due to slow computation and memory limitations. We also tried Linear SVC, but we encountered a max iteration problem, this may be due to our data not being linearly separable.

Moreover, we noticed that the Logistic Regression model gives similar results to the Neural Network, with NN model having slightly lower accuracy. This is due to the similarities in between these two models, which is they both use Gradient Descent, meanwhile the difference is in the Learning Rule. Here, we used ReLU activation for Neural Networks, but if we used Logistic / Sigmoid function instead, we would get the same results as Logistic Regression. Although Neural Networks is more complex than Logistic Regression due additional hidden layers, it may not necessarily lead to better results. This is because Neural Networks are more difficult to train and more prone to overfitting than Logistic Regression.

Naive Bayes is generally suitable for smaller training sizes. However, as the dataset grows bigger, the performance will decrease. Since most real data sets are not perfectly independent, Logistic Regression performs better as it learns the probability of a sample belonging to a certain class of real data sets.

Furthermore, our hypothesis is that Stacking Ensembles would perform a lot better than any other base model. However, our result shows that it only differs a little bit with the Logistic Regression. We analyzed that one of the reasons is caused by strongly correlated base models. Because of this, the benefits of using Ensemble’s method may be lower. Another possible reason is that there is a large amount of data which has made the pattern easier to capture, so classifiers selection might not impact much on the prediction. Although ensemble is a powerful method because it devises sophisticated algorithms to produce high accuracy, often it is not preferred by industries because interpretability is generally more important (Demir, 2016).

**References**

Aiyappan, Jaya (2019, Aug 26th), *Naive Bayes Classifier for text Classification. How it works. https://medium.com/analytics-vidhya/naive-bayes-classifier-for-text-classification-556fabaf252b.*

Brownlee, J. (2020, April 10th). *Stacking Ensemble Machine Learning with Python. Ensemble Learning*. *https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/#:~:text=Stacked%20Generalization%20or%20%E2%80%9CStacking%E2%80%9D%20for,dataset%2C%20like%20bagging%20and%20boosting.*

Codemento Blog (2020, May 5th). *Comparing Support Vector Machines and Decision Trees for Text Classification*. *https://www.codementor.io/blog/text-classification-6mmol0q8oj.*

Demir, N. (2016). *Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results. https://www.toptal.com/machine-learning/ensemble-methods-machine-learning.*

Gandhi, R. (2018, May 18th). *Improving the Performance of a Neural Network. https://towardsdatascience.com/how-to-increase-the-accuracy-of-a-neural-network-9f5d1c6f407d.*

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.

Roy, Abhijit. (Nov 7th). *A dive into decision trees. What is a decision tree?* *https://towardsdatascience.com/a-dive-into-decision-trees-a128923c9298#:~:text=A%20decision%20tree%20is%20a,dataset%20to%20the%20fullest%20purity.*