Machine Learning: Model Analysis

Predicting Cooking Time of Recipes

Dated: May, 2021

Anonymous

1. INTRODUCTION

The aim of this project is to predict the cooking time for recipes based on the steps, ingredients and other features. The recipes data (Majumder et al., 2019) has been collected from Food.com[1]. Further investigation reveals that refinement over the initial feature space including data pre-processing and redundant feature elimination yields enormous improvement of prediction accuracy, which suggests the feasibility of classification of cooking time given recipes.

The selection over classifiers will be limited to multi-class classifiers from intuitive inspection. Therefore, the primal classification algorithms need coupling with multi-class strategy (One-Vs-Rest, etc.). Time complexity should be one of the crucial factors of algorithm selection given the largely spanning feature space. Provided the reasoning above, the Multinomial Naive Bayes Classifier (MNB) could be considered as a plausible benchmark classifier given its universal robustness of underlying assumption, i.e. conditional independence and its computational light-weightedness.

The further exploration will be over the Support Vector Machine Classifier (SVM), the Random Forest classifier, the Logistical Regression classifier and a stacked ensemble classifier. The predominance over the benchmark classifier and underlying reasoning will be investigated.

2. PRE-PROCESSING

2.1 Reserved words and numbers

The textual features will be stripped of all the punctuations, non-English words, words with accents, emoticons and stopping words (provided from 'nltk' library). Direct conversion without cleaning would incorporate data noise and outliers.

2.2 Stemming and lemmatization

The cleaned data will be piped to the tokenizer. The tokenized words will be stemmed and lemmatized. This could minimize the number of redundant features to the maximum extent. The complexity of lexicon morphology would lead to the potential redundancy and bias.

2.3 CountVectorizer

The textual features need converting to the numerical features, which allows learning of classifiers. The CountVectorizer converts text documents into Bag of Words ignoring the relative position information of the words, which could possibly lead to the loss of information. The proximity of semantic relations could not be captured. From intuition, the semantic proximity might not be significant. The occurrences of words in steps, ingredients and names of recipes data would capture most of the information. Compromising the spanning of the feature space, the hyper-parameter of n-gram n is set to be ranged from 1 to 3 in the word level. This parameter setting allows to retain contextual information related to the cooking time of recipes across n-numbers of consecutive words. Words with rare occurrences will be excluded from the vectorization by the parameter minimum document frequency, which improves robustness and eliminates potential

[1] https://www.food.com/

classification bias on rare words. In this stage, a size of (40000, 93831) sparse matrix will be generated.

3. FEATURE SELECTION

Elimination of redundant features will largely improve the efficiency of the model training. In this stage, multiple criteria are adopted to minimise the redundancy and decrease the risk of over-fitting.

3.1 Variance threshold: 0.01%

The features with low variances lack the ability to predict. Eliminate the features with the low variance below a certain cutoff.

3.2 Univariate feature selection:

Mutual-Information quantifies the rankings of the importance of the features used to predict the class labels. 50% of features will be filtered out.

3.3 Select from model

LASSO Linear Regression (Tibshirani 1996) penalizes with L1 norm by shrinking the coefficients of less important features. SelectFromModel embedding LASSO Linear Regression will select the non-zero coefficient features, which prevents over-fitting and reduces redundancy.
The parameter C could control the sparsity: the smaller C the less features will be selected.

4. PREDICT MODEL

4.1 Generative Multinomial Naive Bayes

Generative Multinomial Naive Bayes classifier (GMNB) is based on the conditional independence assumption. The word vectors were generated from the textual feature by CountVectorizer using n-gram (1,3). The dimension of the feature space is notably large, and hence the conditional independence among features will more likely hold. Naive Bayes Classifier performs computationally faster than other complex classification models, e.g. LinearSVC. Due to its moderate scalability (growing linearly with the number of instances) and robustness of the simple underlying assumption, GMNB is considered as the benchmark classifier for this task. Intuitively, GMNB gives the best performance for the word occurrences in the documents among all the typical naive bayes models. Generative models are based on the heuristic estimation of joint probability distribution of observing a word vector given the presence of a class (Ng and Jordan, 2001). However, the results reveal that Bernoulli Naive Bayes Classifier outperforms GMNB classifier (See Figure 1).

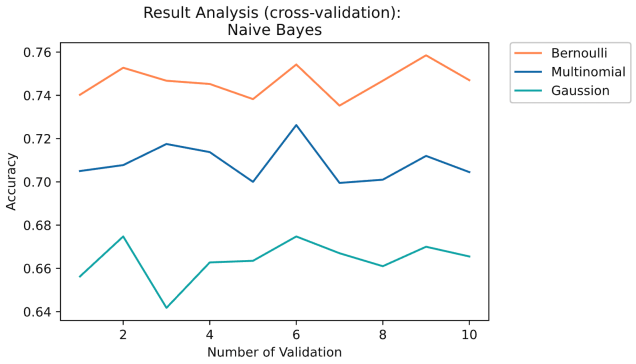

Figure1: Different Naive Bayes Classifiers Accuracy scores

From inspection, the vectorized data almost consists of the binary values 0s or 1s, which possibly violates the applicability of GMNB. Forced use of GMNB will result in over-fitting and compromising generalisation. The results showed that Bernoulli Naive Bayes Classifier gives the accuracy score of 74.65%. This benchmark classifier provides satisfactory prediction, which indicates the feasibility of this prediction task. However, the confusion matrix (See Figure 2) demonstrates the systematic errors. The errors mainly occur on the prediction of the class label 3 because of the unbalanced data distribution. The proportion of the training set data for class 3 is extremely low, which justifies the poor prediction performance.

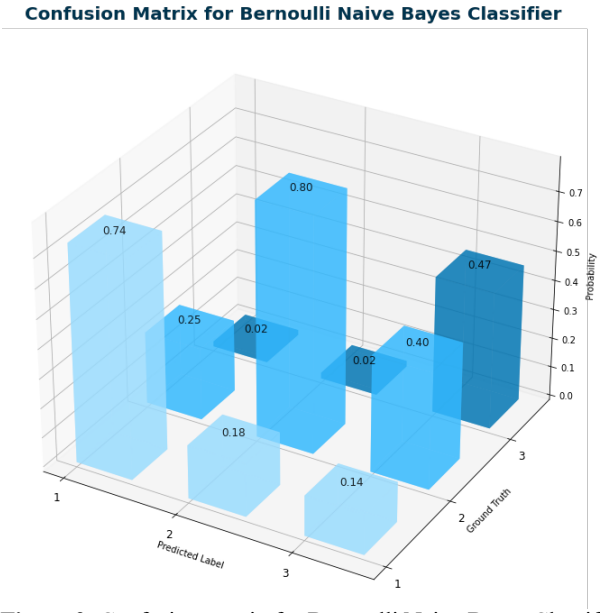

Figure 2: Confusion matrix for Bernoulli Naive Bayes Classifier

4.2 SVC and LinearSVC

Support Vector Machine classifier (SVC) maximizes the margin which separates the instances by the hyperplane. The hyperplane could be determined by solving the corresponding convex optimisation problem. The base SVC is not compatible with the binary classification problem without applying to One-Vs-Rest Approach. The textual data encoded with Bag of Words can produce highly sparse feature space, which consolidates linear separability. Therefore, the SVC intuitively works for the problem. Given the results from the GMNB model, the assumption that features might approximately follow with Bernoulli distribution could be made. The sparsity and possible orthogonality indicates the potential linearity between the features and duration label. Therefore, LinearSVC or Kernel SVM could be applied to validate this assumption (See Figure 3). The results showed that LinearSVC gives the accuracy of 81.27% on the unseen dataset and the model could converge fastly, which is an impressive improvement compared to GMNB.

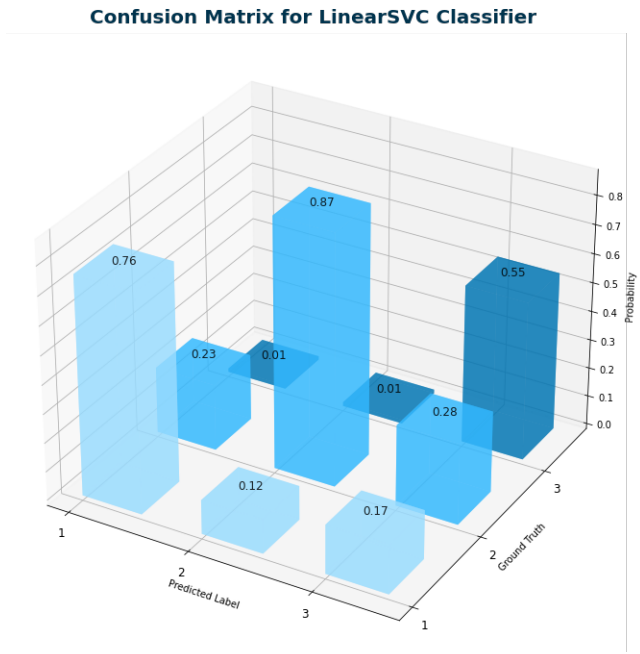

Figure 3: Confusion matrix for Linear Support Vector Classifier

The high degree of generalisation without hyper-parameter optimisation would even consolidate the linear separability of the feature space. Stacked on the stochastic gradient descent solver and GridSearchCV (Figure 4), the hyper-parameter will be fine tuned, which even escalates the degree of generalisation. The minimal regularisation term C tends to give the highest accuracy, which indicates that the model might be under-fit or over-generalised given a large C value.

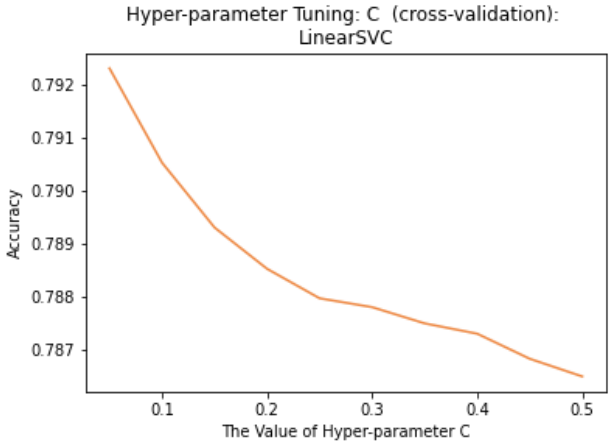

Figure 4: Hyper-parameter Optimisation: C (cross-validation)

Beyond this, the issue addressed above that the unbalanced data distribution would affect the prediction accuracy is further mitigated (Table 1).

Prediction/ Truth	1	2	3
1	1331	412	20
2	291	1724	23
3	22	49	128

Table 1:LinearSVC Confusion matrix With Hyper-parameter Tuning

4.3 Random Forest

Random forest classifier (RF) is an ensemble learning method based on aggregated decision trees and Bootstrap aggregating. By randomly constructing multiple decision trees at training time, the class label is given by the weighted majority voting of the classes from decision trees. Random forest classifiers mitigate the issue of over-fitting to the training set in comparison with decision tree classifiers. Random forests generally outperform decision trees with less variance but sacrificing the interpretability. RF produces relatively good prediction results even with the default hyper-parameter settings (Figure 5). Due to its randomness, the process and behaviour of the RF is hard to interpret. Although RF is extremely time consuming and unable to be interpreted, the high quality of prediction is worth an attempt, especially prediction on the minority class (Table 2).

Prediction/ Truth	1	2	3
1	1378	384	1
2	314	1719	5
3	33	71	95

Table 2: Random Forest matrix

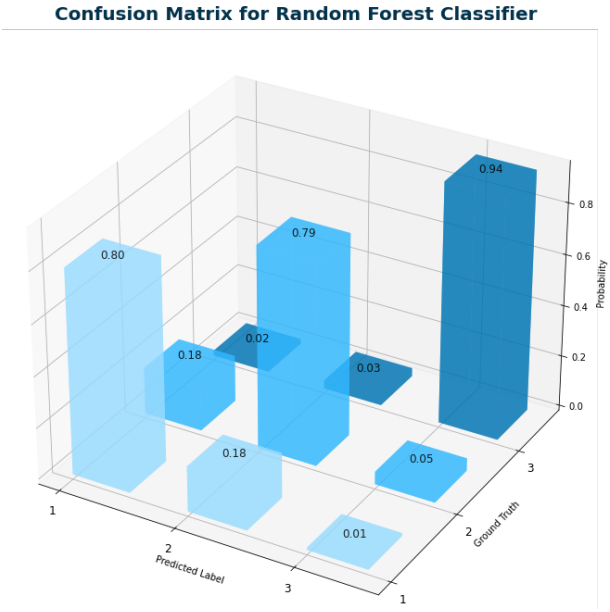

Figure 5: Confusion matrix for Random Forest Classifier

The results demonstrate a high degree of generalisation with low model bias. Uneven distribution could hardly increase the prediction bias. The over-fitting of minority classes could be further alleviated. However, the over-tuned hyper-parameter will reversely cause over-fitting. Hence, the advanced decision-tree based learner with gradient descent solver and regularization term could be one possible solution, e.g.XGBoost Classifier. The accuracy rate given by XGBoost Classifier dominates the other classifiers, which gives accuracy of 82.4%.

lead to poor prediction accuracy and hence this default implementation is not selected. The self-implemented Stacking learner gives notably high precision, and further decreases the model variance and model bias, which displays a high degree of generalisation (Table 3). Different base models would tend to different types of the prediction bias. For example, RF would give the highest accuracy of 95% on class three. Therefore, weighted majority voting could incorporate the advantages of each base classifier and reduce overall model variance (Table 4). The overall accuracy of this Stacking learner will be approximately 82.4%, albeit time consuming,

Different Calculation \ Results	Precision	Recall	F1 Score
MACRO	0.84	0.74	0.78
MICRO	0.81	0.81	0.81
WEIGHTED	0.82	0.81	0.81

Table 3: Multi-class Scoring Metrics for Stacking Learner

Labels	1	2	3
Precision of all class	0.78	0.84	0.89
Recall of all class	0.85	0.81	0.55

Table 4: Precision & Recall per class for Stacking Learner

5. ENSEMBLE MODEL

5.1 Stacking

Stacking uses a meta classifier to combine the predictions of the base classifiers. The base model encodes the features and renders the output to the meta classifier awaiting final prediction. The implementation of the Stacking method for this prediction task will use four base models: LinearSVC, Random Forest (using gini as criterion), XGBoost Classifier and SGD Classifier. The default implementation for meta classifier is overwritten. Weighted majority voting is applied on the predicted labels by base classifiers to determine the final results. Feature encoding (probabilistic prediction) by the base classifiers would

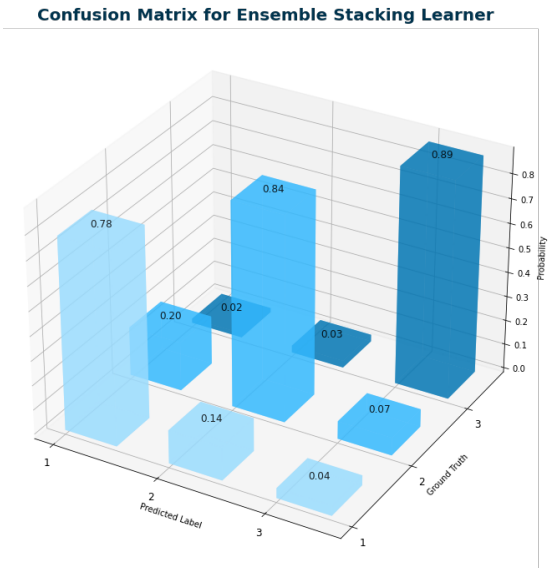

Figure 6: Confusion matrix for Stacking Learner

5.2 Boosting

Boosting is initiated with a baseline learner. Each run increases the weight of the instances that are incorrectly predicted so that the instances that are prone to be falsely labelled would be intensively re-learned. The main purpose of this method is to reduce bias for higher accuracy. The most common implementation for boosting is Ada-boosting. Boosting could improve the accuracy to some extent. However, this usually comes with a trade-off between the prediction bias and generalisation. The results showed Ada-boosting Classifier gives the accuracy rate of 72.2%, which is even lower than the benchmark Naive Bayes Classifier. This might be the indication of model over-fitting. Thus, further hyper-parameter optimization did not apply to this model. Conversely, the XGBoost is also an implementation of an ensemble model with Boosting algorithm. This classifier gives satisfactory performance of 82.4% on the unseen data and also gives a large degree of consideration to the bias on the prediction. Compared to RF, the model reduces model variance and bias through an iterative process though time-consuming.

6. ERROR ANALYSIS

As Table 5 indicated, the class is extremely unevenly distributed. The number of the instances for duration label 3 is apparently much less than the others. The instances could hardly be trained or fully learned on the training stage. The model would naturally bias towards the majority classes.

Duration Label	Total Number	Training	Testing
1	17705	15942	1763
2	20246	18208	2038
3	2046	1805	199

Table 5: Class distribution for training data

From inspection, the second class would be prone to incorrectly labelled as the other classes, which might impute to the unclear boundary between medium duration and other duration length. The exact formatted time often occurs in recipes with medium duration. However, this information would be eliminated after data pre-processing and thus causing loss of information. Even the keywords corresponding to the most important feature occurring in the recipes with medium duration after feature selection are still not quite seldom seen among the other classes. As the results, the feature selection would tend to retain keywords for the recipes or ingredients. The exact formatted time is forced to be retained using unigram, bi-gram or tri-gram in both word level and character level. These hypotheses are reckoned to be the main cause of model bias when predicting. After refinement on feature pre-processing and feature selection, the performance of the classifiers uniformly increases at approximately 1%. The feature number of ingredients and number of steps seem redundant and less predictive. For example, the number of steps in recipes does not necessarily positively correlate with the cooking time. However, removing these features will lead to the decrease of prediction accuracy on class three. If more training data of class three could be introduced, this might improve overall accuracy.

7. CONCLUSION

Model	Validation-Accuracy	Kaggle-Accuracy
Bernoulli Naive Bayes	0.7465	
LinearSVC (Hyper-parameter Tuning)	0.79675	0.81266
Random Forest	0.798	0.80700
Ensemble Stacking	0.8125	0.82333
Ensemble XGBoost	0.79675	0.82400

Table 6: Class distribution for training data

The additional pre-processing and feature selection using VarianceThreshold and mutual information scoring would increase the overall accuracy rate up to 4% observed on the classifiers selected. The best models among all the classifiers discussed above, will be ensemble models, including Stacking method and boosting algorithm (Table 6). The outstanding model will be Stacking learner, which gives prediction accuracy of 82.37% on the unseen data. This prediction task is deemed to be feasible and accessible. Combined with the model efficiency, the Stacking learner model tends to have the least model variance and model bias, and hence predominates the other classifiers. Even different models would tend to converge in prediction accuracy; However, the prediction over different classes would vary vastly.

Bibliography

[1] Majumder, B., Li, S., Ni, J. and McAuley, J., 2019. Generating Personalized Recipes from Historical User Preferences. 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)*.

[2] Robert, T., 1995. *Regression Shrinkage and Selection via the Lasso.*. Ph.D. University of Toronto, Canada.

[3] Ng, A. and Jordan, M., 2001. *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes*. Ph.D. University of California, Berkeley Berkeley, CA 94720.