Explore Feature Selection and Different Performance of Machine Learning Classification Models on Cooking Time Estimation

****

Melbourne School of Computer Sciences, University of Melbourne
COMP30027: Machine Learning

Anonymous

Due Date: 21/05/2021

Word Count:2500

1. Introduction

In this report, general data manipulation techniques and a lot of commonly used classification models will be discussed by looking at the results after applying them to a classification task where models need to predict the label of the preparation time of a recipe (either quick medium or slow). Supervised models like Neural Network, Decision Tree, Random Forest, Naïve Bayes and Stacking model are used. The strength as well as weakness of each of these models will be analysed and illustrated through the process of experiment.

2. Task description

The classification task in this experiment is to classify the preparation time label of a recipe. The label can fall into three categories, quick, medium and slow. Features like name of the recipe(String), number of steps(Discrete), number of ingredients(Discrete), and detailed information about each step and every ingredient in text format(String) are provided to build the classification model. The work to be shown can be split into

1. Data Manipulation: where sample data are drawn to help draw a graph to give us a hunch of data distribution and partial features are used to build some very simple models and compare with 0R model.
2. Feature selection
3. Classifiers construction
4. Evaluation of each Classifier

2.1 Data Manipulation

Firstly, we explored original data features and learned a very important basic rule of machine learning: always hands on the data first rather than JUST working on algorithms.

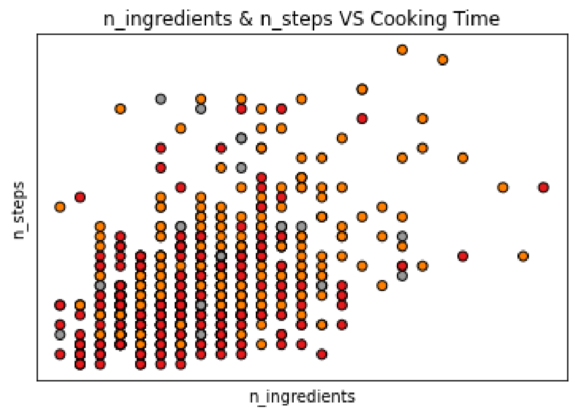
2.1.1 Logic and Pre-Processing

Numeric features could be discrete or continuous data and different methods shall be applied to two kinds. Since the target dataset includes numeric features "n_steps"(number of steps needed), "n_ingr"(number of ingredients used). We firstly use numeric data only to do the prediction.

2.1.2 Explore classification models based on Numeric Features

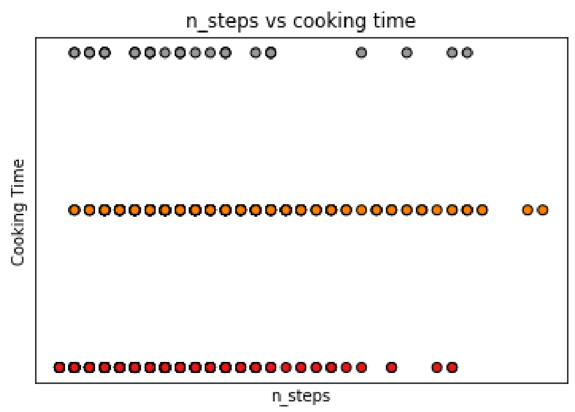
The relationships between numeric features has been tested, the test using 1000 random samples from the dataset, the relationship plots has been generated

Plot 2.1.2.1 Relationship between n_ingredients, n_steps and Cooking Time



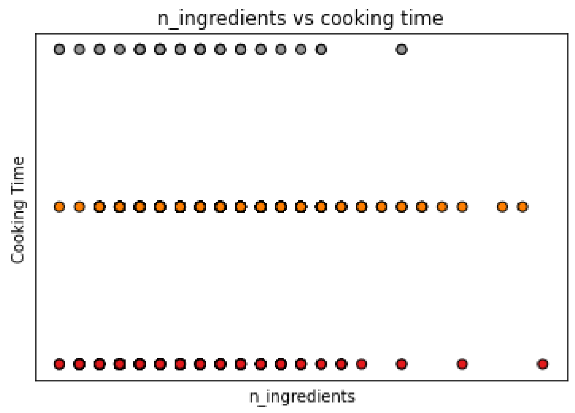
This Plot indicates n_ingredients and n_steps value according to different Labels

Plot 2.1.2.2 Relationship between n_ingredients and Cooking Time



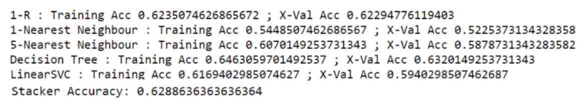
This Plot indicates the distribution of n_steps value according to different Labels

Plot 2.1.2.3 Relationship between n_steps and Cooking Time



This Plot indicates the distribution of n_ingredients svalue according to different Labels

Plot 2.1.2.4 Modelling based on numeric features



```
1-R : Training Acc 0.6235074626865672 ; X-Val Acc 0.62294776119403
1-Nearest Neighbour : Training Acc 0.5448507462686567 ; X-Val Acc 0.5225373134328358
5-Nearest Neighbour : Training Acc 0.6070149253731343 ; X-Val Acc 0.5878731343283582
Decision Tree : Training Acc 0.6463059701492537 ; X-Val Acc 0.6320149253731343
LinearSVC : Training Acc 0.6169402985074627 ; X-Val Acc 0.5940298507462687
 Stacker Accuracy: 0.6288636363636364
```

This Plot indicates the performance of numeric features prediction

By doing classification implementing several classifiers and as the plots shown above, there is a minor linear relationship between numeric features in this dataset. It is shown that only using numeric features may not proper describe the dataset and therefore resulting in bad performance.

2.1.3 Conclusion

In conclusion, a general issue in machine learning study has been raised. That no matter what algorithms we use, if the data itself is not well formed or informational, the performance of prediction will probably be always awful. Therefore, **Hands on the data first rather than algorithms by doing Feature Engineering Select Best Features.**

2.2 Hands on the Data

As we adapt our research direction after the lesson we learned in Initial Attempt, the next thing we do is to get our data selected and sorted using string features.

The string features can be expressed mainly in two ways.

1. A dictionary of counting for every word(CountVec). By doing word based counting, it is useful for us to examine the most frequently appeared word and can further calculate TF-IDF to check if a word is significant among other documents.
2. a vector standing for context similarity among words(Dov2Vec). By implementing this data structure for string features, it might be easier to analyse which words are most likely to

appear next to each other and it's good for text prediction.

2.2.2 - Feature Selection

2.2.2.1 Logic and Methods

After examining the numeric data in the dataset "Count Vec", this time we are specifically looking at string features "name", "ingredients" and "steps". Ingredients and Name has more than 10,000 words count features while steps has around 2400 words features.

The optimal feature selection steps are:

1. combining all relevant features into a big dataset.

2. doing use models like Mutual information or Chi-Square to do the feature selection:

Mutual Information Calculates the independence between feature x and class label y, the independence score also indicates whether this feature can be used to distinguish some class label than others, in plain words it calculates the relatedness of data and class label:

$$MI = \sum\sum P(x,y) * log\frac{P(x,y)}{P(x) * P(y)}$$

2.2.2.2 Divide and Conquer

However, as the dataset is too big with too many features, it is not practical to do all features combining then selection. Therefore, the feature selection has been performed before the combinations of data. So the final step we took for feature selection is:

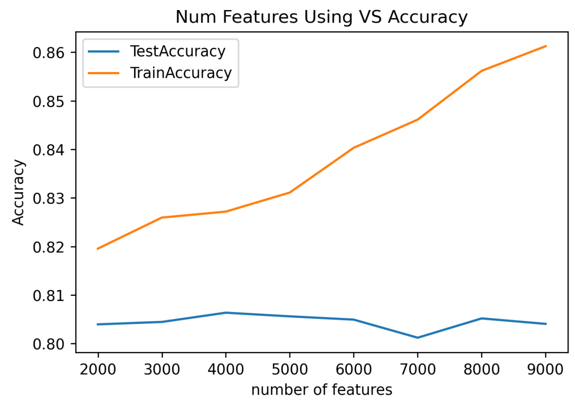
1. Select a small number of features from "name", "ingredient", "step" separately to get a smaller matrix for our data. (Ki is number of features selected from i-th features)
2. combine three dataset together and perform second-round feature selection to get an even smaller combined dataset. (Ktotal is the final combined number of features)

However, as the process introduced above, though it is reasonable to do so, how can we make sure that one of the specific features may not be good at all? (For example, selecting 1000 features from "name" and 1000 features from "step" then combining them together will get worse performance than solely using all 2000 features from steps if name is overall a worse predictor than steps). Therefore, to maximise our accuracy, we choose features from different data("steps", "ingredients") as many as possible (10906 features).

2.2.2.3 Overfitting Trap

It is generally agreed that choosing too many parameters (features) may lead to overfitting. Therefore, by selecting different number of features we test our model's train and test performance using cross validation our Multilayer Perceptron Model, we get following results:

Plot 2.2.2.3 - Number of Features Used and MLP Model Performance



The Testing scores are cross-validation score on original data.

As the plot above suggests, too many features used in fitting MLP classifiers may result in overfitting as our train accuracy is always improving but testing accuracy even falls around 6000 features in use. Therefore, in the following classification steps, we choose 5000 features for use.

3. Model 1: Decision Tree

Simple models like Decision tree and K-nearest neighbors are implemented and compared with 1-R. We do not specify the tree depth for the decision tree model. So it is very likely that it overfits the training data as it can have infinite large tree depth to perfect training accuracy. The gap between training accuracy and testing accuracy could be the evidence of overfitting. But the result of testing accuracy is still better than 1-R, meaning that the overfitting may not be that severe.

The reason why KNN has a worse accuracy than 1-R may be because KNN is an instance-based classification and it is very prone to the curse of high dimensionality as in this case, we select 5,000 features to train the model. The training space could be very sparse, which leads to the bad performance of KNN.

Document	Training Accuracy	Testing Accuracy
Decision Tree	1	0.734
KNN(n=5)	0.607	0.588
1-R	0.624	0.623

Table 1- results between DT, KNN(N=5), 1-R

4. Model 3&4: Multinomial Naïve Bayes & Gaussian Naïve Bayes Classifier

Both classifiers are based on bayes' rule which is that, based on our previous experience, We can fit a distribution to each feature given a certain class. The difference between MNB and GNB obviously can be seen from its name. MNB fits a multinomial distribution while GNB fits a normal distribution or gaussian distribution. In practice, to classifiers, training data are previous experience to them. For MNB, it stores the average frequency of each feature case when a certain class is happening while GNB calculates the mean value and standard deviation of each feature when a certain class takes place. After training, both classifiers can build a distribution for each feature. When it comes to prediction, they find the class label that has the highest probability and set it as the predicted result.

When applying them to the task we mentioned above, they achieved a quite distinguishable result. MNB could reach around 0.7 accuracy in both training and testing phase while GNB could only obtain less than 0.35 accuracy in both phases, which are only half of those of MNB.

Classifie r	Training Accuracy	Test Accuracy
GNB	0.3401	0.3260
MNB	0.7281	0.7182

Table 1- Performance of GNB&MNB

4.1 Reasons of performance

In this section, we will talk about the reasons why two similar models could lead to such different results. In addition, we will talk about what could impede MNB from achieving better results.

4.1.1 Weakness of GNB

The reason why GNB gets such a low accuracy is that features like number of steps, number of ingredients etc are in the nature discrete values. By fitting a gaussian distribution to these features, it gives probability value to feature values that would never exist. And therefore, the probability of most likely value for each feature is lowered, making it hard to distinguish the most likely class from the others. On the other hand, MNB does the right thing. It takes the feature values as discrete. Also, it works very well for features in string format. Because they can be turned into word counts, which is also a discrete feature rather than continuous feature.

4.1.2 Stumbling block of MNB

Although MNB reaches far better results than GNB, they are not in the high eighties. One stumbling block that may prevent MNB from obtaining high accuracy scores like neural networks does is its strong independence assumptions. MNB assumes that each feature value is independent of any of the other feature values. In this task, it may not be quite the case. For example, large number of ingredients usually takes a lot of steps to process them and thus the preparation time of such recipes tend to be slow.

4.2 Weakness and strength of Naïve Bayes model

Naive Bayes has many advantages compared to other models. First, they are rather simple models compared to models like neural networks, random forests. And thus, it is easy to build the models and tell how they are making decisions and predictions. Second, the computation scales well compared to other models. But we need to be careful not to forget the smoothing method and which smoothing method to use.

Weakness or limitation of GNB and MNB classifiers is their strong independence assumption. It limits them from extracting valuable relationships between features and takes advantage of them to make more accurate and faster predictions

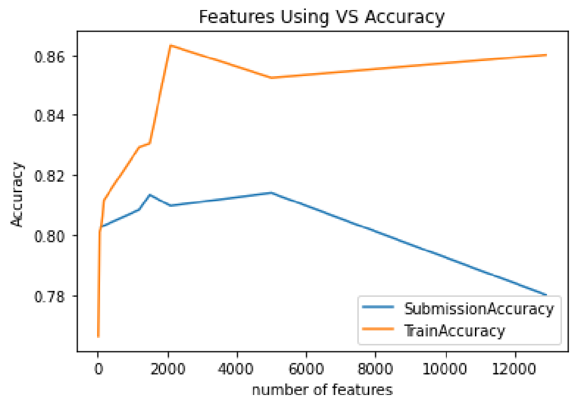
5. Model 5: Multi-Layer Neural Network

M By implementing MLP method to our dataset, we used a basic Neural Network classifier. The Neural Network MultiLayer Perceptron classifier basically uses the gradient descent method to calculate parameters. It takes input and assigns each feature a weight parameter, by iterating through the whole data set with different instances, those weight parameters get updated each round. The model is well trained after maximum iteration updates or it converges (the weight parameters satisfied for all instances). However, MLP's parameters can be tuned as it has tons of options.

5.1.1 GridSearch For Tuning

Grid Search Method has been used. Grid Search tries to find the best parameters for a single model (MLP for our task). Parameters can be tuned are: number of hidden layers. Number of layers MLP classifier may use to do the predicting. As this is an arbitrary number, it requires tons of computational power to find the best one. Another one is Alpha value which is a L2 Normalized parameter. The higher Alpha is, the larger error gap is allowed(higher alpha may result a single instance's effect can be zero on our weight parameters)

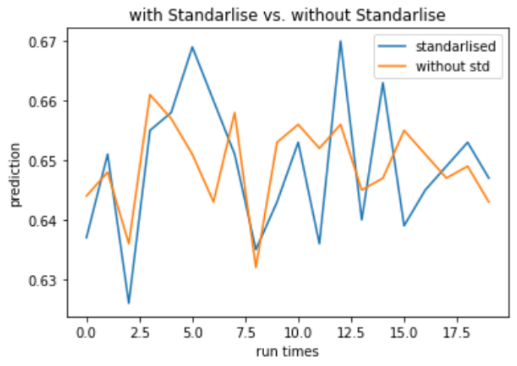
Plot 5.1.1.1 - Number of Features using vs MLP accuracy



Above plot indicates fewer features may results in not proper trained and large features may cause overfitting.

5.1.2 Standardized data matter?

Plot 2.2.1.1 -- Standardised vs Original Data on MLP Performance



```
Accuracy with Standarlised model: 0.649
Accuracy without Standarlised model: 0.6492000000000001
```

This plot shows the Accuracy Score of standardised and unstandardised data with model

As the above plot suggests, when implementing MLP classifiers, whether the data is standardised or not doesn't make a huge difference. The intuition behind this is that MLP does the weight changing and iterative calculation can adapt to the input data unlike a classifier using Gaussian Distribution that needs standardised data, which takes fixed mean and variance.

6. Model 6&7: Random Forest & Stacking

In this section, we will compare two classifier-combined models. One is Random Forest which combines multiple decision trees together and uses majority voting to make final predictions. The other is using the Stacking method, where we use Decision tree, Neural networks and Random forest as base classifiers and Logistic Regression as the meta classifier. The performance is shown below.

Classifier	Training Accuracy	Test Accuracy
Random Forest	0.999975	0.791650
Stacking(10892 features, RF,MNB, NN)	0.966875	0.80533
Stacking(5000 features, MNB, NN)	0.966875	0.81300

Table 1- Performance of Random Forest & Stacking model

6.1 Analysis of Random Forest

Random Forest is generally a good model. It easily outperforms simple models like decision tree and MNB. But it took a lot of time to find the optimal hyperparameter like number of decision trees to be used in the random forest, tree depth, and split of sample. It took us 1300mins to find them and finally we aborted this task as it is so computational intensive

6.2 Analysis of Stacking method

Unfortunately, the stacking model does not show better performance than the neural network contained inside it.

Also it is also computational expensive in terms of calculating the accuracy, as there is a nested cross validation.

7. Limitations

Although we tried different combinations of classifiers and feature engineering method, there are still limitations about this research.

Our feature selection and model fitting process has been limited due to lack of computation power. The dataset has a huge amount of data and a number of features. To find optimal combination/selection from such a complex dataset with features, it requires a lot of computation power. For example, it would be much better to combine all features together then do feature selections as we have more information to do feature selections. To store and compute the score for each feature, it requires lots of memory storage and computation power. Additionally, by implementing a Multilayer Classifier, it needs a lot of parallel computation and it would be much more efficient if the GPU can be made available for computing tasks. GridSearch can also take more options to find even better solutions as well.

7. Conclusions

Decision tree: difficult to determine the best depth of tree to get the optimal performance without trial and error.

KNN: it is a uncompetent classification model in natural language processing, as the dimensionality scales up to large numbers.

GNB: Simple and fast model. It is a more general model than MNB but it has a very disappointing performance on this classification task. Mainly because it fails to recognise the very nature of those features it has been provided(treating discrete data as continuous data is costly). And thus, it has an even worse performance than 0-R.

MNB: Better performance than GNB when features are discrete. Its pre-defined assumption has a limitation on its performance.

MLP: A modern Classification Technique that does heavy parallel calculations to get the best fitted model, though overfitting is an issue to be tackled and tuning takes a huge amount of time to get further improved.

Random Forest: could achieve better performance than the base model(decision tree) but more computationally difficult to find the optimal hyperparameter.

Stacking: Could maintain the best performance that base model obtains. But unknown how to achieve determinate improvement.

8. References

Majumder, B. P., Li, S., Ni, J. & McAuley, J. Generating personalized recipes from historical user preferences. 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.