COMP30027 Machine Learning Project Report

Anonymous

1. Introduction

The aim of this project was to build and analyse supervised machine learning models to predict the cooking time for recipes based on recipe features. The cooking time was categorized into three classes (quick, medium and slow). The dataset has been sourced from (Majumder, Li, Ni, & McAuley, 2019).

There has already been quite a lot of research done using recipe details as features, but most of these models relate to generating new recipes or ingredients. Bień, Gilski, Maciejewska, & Taisner (2020) propose a deep learning model that reads through recipes and creates a recipe given input ingredients. A more relevant example is detailed by Wang (2019) where the process of cooking time prediction at Uber Eats is based off a feedback loop that adds data from every completed order, to improve the initial ground truth.

2. Method

For this project, I created 4 models : Naïve Bayes model, Decision Tree, Support Vector Machine and a Logistic Regression model. For each model, I tested on the datasets with a range of hyperparameters as described below. After the best one was chosen, it was used to make predictions on the test dataset.

2.1 Naïve Bayes

For the Doc2Vec datasets (50 and 100 features), I trained and tested a Gaussian Naïve Bayes model. The data features a range of continous values (positive and negative), hence a normal/Gaussian distribution works best. For the Bag-of-Word representation dataset, I trained and tested a Multinomial and a Bernoulli model. This dataset is a word count consisting of 1s and 0s. This is why Multinomial and Bernoulli distribution would fit better than a Gaussian distribution. In both, Laplace smoothing was used with an alpha value of 1.

2.2 Decision Tree

Initally, I attempted to discretise the data using 3 methods : equal-width, equal-frequency and k-means. However, the discretised data did not yield good performance. Hence, I trained two Decision Trees on the continuous data using information gain and Gini impurity to check the quality of the tree split.

2.3 Support Vector Machine

I tested the methods for handling multiple classes: one-vs-all and one-vs-one. Then, I tested various kernel choices (linear, polynomial and rbf) on the Doc2Vec datasets. For the polynomial kernel, I tested a range of degree values to find the most accurate one.

2.4 Logistic Regression

The method for handling multiple classes was once again tested. I also compared different optimization algorithms available in the sklearn library such as newton-cg, lbfgs, liblinear, sag and saga. Aside from the given datasets, I also tested on a dataset of my own with the number of steps and number of ingredients as the features.

3. Evaluation

For each of these models and hyperparameters, I trained and tested on the training dataset. This was done using the random holdout approach (train_test_split) with a test set size of 0.33. I initially used cross validation, but my computer struggled with the time and processing power required, hence random holdout was used.

Once trained, the models were tested by making predictions and evaluating them. To evaluate, I calculated the accuracy and the F-score (globally). The F-score was chosen because it gives the harmonic mean of precision and recall, an ideal way of comparing both metrics together.

4. Analysis

This section analyses each model's results and provides an explanation for their behaviour.

4.1 Naïve Bayes (NB)

The Gaussian NB has lower scores than the other two (Figures 1, 2). This could be because a normal distribution isn't the best fit for the Doc2Vec datasets.

Doc2Vec Dataset (No. of features)	Average Accuracy	Average F1-score
Name (50)	0.576	0.576
Steps (50)	0.621	0.621
Ingredients (50)	0.541	0.541
Name (100)	0.569	0.569
Steps (100)	0.603	0.603
Ingredients (100)	0.540	0.540

Figure 1 – Table showing Gaussian NB evaluation scores for Doc2Vec datasets.

The Multinomial NB performed best because the bag-of-words dataset fitted the multinomial distribution well (Figure 2). This is surprising as I expected the Bernoulli to fit better, as its attributes are binary. The dataset which performed the best has a very high number of features, which is in line with the theory that NB models scale well to high-dimensional datasets.

The NB model performed well in comparison with the other models. However, the conditional independence assumption might not hold for this dataset. For example, given the class, I believe there is a correlation and dependency between number of steps and steps, and also between number of ingredients and ingredients.

Bag-of-words dataset (model)	Accuracy	F1-score
Name(MNB)	0.692	0.692
Name(BNB)	0.689	0.689
Steps(MNB)	0.737	0.737
Steps(BNB)	0.733	0.733
Ingredients(MNB)	0.657	0.657
Ingredients(BNB)	0.652	0.652

Figure 2 – Table showing Multinomial NB and Bernoulli NB evaluation scores for Bag-of-words datasets.

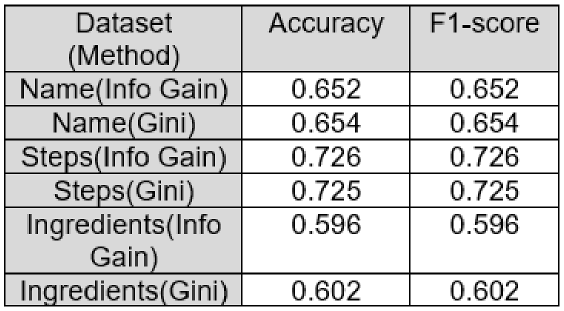
4.2 Decision Tree

The discretised data did not perform well (Figure 3) maybe due to the fact that the discretisation methods are sensitive to outliers. Another reason is because discretisation always leads to information loss. If the information is relevant to classification, it could lead to poor performance.

	Equal-width	Equal-frequency	K-means
Accuracy	0.484	0.484	0.483

Figure 3 – Table showing accuracy scores for discretisation methods on a decision tree model, on one dataset.

The performance of the two methods for selecting split (information gain and Gini impurity) were very closely matched (Figure 4). I chose Gini impurity, as it requires slightly less processing power.



Dataset (Method)	Accuracy	F1-score
Name(Info Gain)	0.652	0.652
Name(Gini)	0.654	0.654
Steps(Info Gain)	0.726	0.726
Steps(Gini)	0.725	0.725
Ingredients(Info Gain)	0.596	0.596
Ingredients(Gini)	0.602	0.602

Figure 4 – Table showing Decision Tree evaluation scores for Bag-of-words datasets.

The model performed acceptably. It was important to prevent the decision tree from overfitting. This was done by having a smaller training set (0.3) and larger test set (0.6). A smaller training set leads to a smaller tree, hence a lower chance of overfitting. Sebban, Nock, Chauchat, & Rakotomalala (2000) explained this in their statistical proof of the "*linear relationship between training set size and tree size*".

4.3 Support Vector Machine (SVM)

When comparing results of the methods for handling multiple classes (Figure 5), the one-vs-all approach is slightly worse as it will always see an unbalanced distribution (more of 'all' class than the 'one' class).

	Accuracy
One-vs-One	0.594
One-vs-All	0.593

Figure 5 – Table showing accuracy scores for 2 methods of handling multiple classes in SVM on one dataset.

For the SVM model with the polynomial kernel, I chose the degree 3 based off Figure 6. Despite the graph showing accuracy decreasing when the degree increases, I concluded that this is just a local descent and accuracy will eventually increase with higher degree values (due to overfitting).

Figure 7 shows that the kernel choices had very similar scores. The radial basis kernel function was chosen, as it performed slightly better. This might be because the rbf has similarity to the Gaussian distribution which fits this dataset well.

The dataset with the best performance has a high number of features (100) which proves the idea that SVM works well with high dimensions.

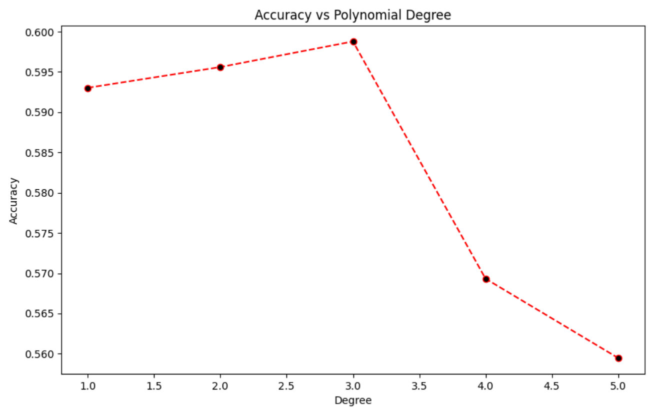


Figure 6 – Graph showing the SVM accuracy against the degree of the polynomial.

Doc2Vec Dataset (Kernel)	Accuracy	F1-score
Name (linear)	0.606	0.605
Name (rbf)	0.635	0.635
Name (poly)	0.610	0.610
Steps (linear)	0.664	0.664
Steps (rbf)	0.720	0.720
Steps (poly)	0.698	0.698
Ingredients (linear)	0.581	0.581
Ingredients (rbf)	0.627	0.627
Ingredients (poly)	0.599	0.599

Figure 7 – Table showing evaluation scores for the 3 SVM kernels for Doc2Vec datasets.

4.4 Logistic Regression

The choice of optimisation algorithm did not have a significant impact on the performance scores (Figure 8). The liblinear algorithm was chosen because it performed slightly better. This could be due to the fact that this algorithm which uses gradient descent, performs better when the chosen dataset has high dimensions.

Algorithm	Accuracy
Newton solver	0.5996
Lbfgs	0.5997
Liblinear	0.6000
Sag	0.5997
Saga	0.5996

Figure 8 – Table showing accuracy scores for optimisation algorithms in a logistic regression model, on one dataset.

The performance of this model on the features I chose (number of steps and number of ingredients) is comparatively low (accuracy of 0.636). This might be due to a poor correlation between the features and the class label. For example, recipes such as the 'chickpeas bruschetta' have a large number of ingredients/steps (14) but low cooking time (1).

This model performs the best out of the 4 created (Figure 9). It has an accuracy of 0.798 on the Kaggle test set. It does not over-fit as L2-norm regularisation has been used. It has high performance compared to the NB model as it doesn't need to estimate probabilities with Bayes' rule and does not need to assume conditional independence (Ling, 2021).

Model	Accuracy
Naïve Bayes (MNB)	0.737
Decision Tree (Gini)	0.725
Support Vector Machine (rbf)	0.720
Logistic Regression (liblinear)	0.785

Figure 9 – Table showing the accuracy scores for the 4 models with their best-performing hyperparameters.

5. Conclusion

This report presented my approach to the task of predicting cooking time. It explained my method and presented an analysis of results.

6. Bibliography

Bień, M., Gilski, M., Maciejewska, M., & Taisner, W. (2020). *Cooking recipes generator utilizing a deep learning-based language model.* Poznan University of Technology, Poznan. doi:10.13140/RG.2.2.23904.51200

Ling, L. (2021). Logistic Regression. *Lecture Notes*. Retrieved from https://canvas.lms.unimelb.edu.au/courses/102017/files/7197725?wrap=1

Majumder, B., Li, S., Ni, J., & McAuley, J. (2019). Generating Personalized Recipes from Historical User Preferences. *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing.* doi:10.18653/v1/D19-1613

Sebban, M., Nock, R., Chauchat, J., & Rakotomalala, R. (2000). Impact of Learning Set Quality and Size. *International Journal of Computer Science and Security, 1*(1), 85 - 105. doi:10.1.1.62.6365

Wang, Z. (2019, November 20). *Predicting time to cook, arrive and deliver at Uber Eats*. Retrieved from InfoQ: https://www.infoq.com/articles/uber-eats-time-predictions/