COMP30027 Report

Project 2: Book Rating Prediction

1. Introduction

This project aims to build and analyse supervised Machine Learning methods in a multi-class classification problem. The goal is to predict the rating of a book based on various numerical and text features such as the book's title, authors, publishing date, publisher, language, page numbers and description. This project explores the use of Neural Network (NN), k-Nearest Neighbours (kNN), and Logistic Regression (LR) classifiers. We then evaluated and compared the three classifiers to find the best performing model in terms of accuracy, precision, and overall efficiency.

2. Methodology

2.1 Experimental set up

The training dataset consists of 23063 instances with 10 features. Before developing the classifier models, the data needs to be pre-processed.

2.1.1 Pre-processing

First, we need to handle missing values. In this case, we had 17000 missing values in the feature 'Language'. In addition to its high number of missing values, the feature itself shows an imbalance. The class 'eng' has about 5400 instances, while the next most common languages 'fre' and 'spa' have about 150 each. Since the dataset is biased towards one language, it doesn't provide any useful information. Thus, we remove the feature 'Language' from our dataset.

As mentioned above, the dataset contains several numerical, categorical and text features. We standardised the numerical features using StandardScaler() from sklearn (Pedregosa et al., 2011). The categorical feature ('Publisher') was encoded using OneHotEncoder in sklearn. The feature 'Publisher' had over 4000 unique values in the training dataset, but about 2000 of them had a frequency of 1. To avoid unnecessary computation, we only encoded publishers that appeared more than two in our training instances. The reasoning is that the conditional probability of a book having a particular rating doesn't change if the publisher has only one book in the dataset. However, if the publisher has many books rated 5.0, we can say that the next book is likely to be rated 5.0 as well.

For encoding text features, such as name, authors, and description, we used the Doc2Vec technique. Doc2Vec generates a fixed-length vector representation of documents. Unlike other techniques such as Bag-Of-Words, Doc2Vec captures the context and relationship between words and sentences in a way that sentences in a similar context would be closer to each other in the Doc2Vec vector representation.

2.2 Choice of models

Since the underlying model of the dataset is unknown, several classifier models have been implemented: k-Nearest Neighbours, Multi-Layer Perceptron Neural Network, Logistic Regression.

K-Nearest Neighbours works well in understanding the underlying structure (clusters) of the data and tries to find the relationship between close data. It is very fast to train and good at capturing nonlinear relationships if the underlying structure of the data (clusters) are distinguishable.

Neural network can learn complex patterns and relationships that are hard to learn in other models. Although it offers a very high performance, computational resources can be an issue if the dataset is very large. In addition, NN models require very careful tuning of parameters to perform optimally.

On the other hand, Logistic regression assumes linear relationship between the log-odds of target variable and input variables. Logistic regression is effective when the data has a clear linear separation between classes. Although it is fast to train, logistic regression has a problem of learning complex relationships between data.

The above three models have distinct characteristics and may help us understand more about the underlying structure of the data based on the model's performance. In the implementation, we use Pipeline() from sklearn to avoid data leakage.

2.2.1 Optimization of parameters

In the implementation of the models, KNeighborsClassifier, MLPClassifier, LogisticRegression in sklearn have been used for kNN, NN and LR classifiers, respectively. To maximize the performance by optimizing the hyperparameters, GridSearchCV in sklearn has been implemented. This module allows us to optimize the parameters using cross-validated exhaustive grid-search for different values of parameters. After performing the GridSearchCV, following parameters were obtained:

kNN:

- n_neighbors = 30

NN:

- hidden_layer_sizes = (100,)
- activation = 'logistic'
- solver = 'sgd'
- alpha = 10
- learning_rate = 'adaptive'

LR:

- no parameter was set exclusively except for max_iter=1000.

2.2.2 Comparison set up

In our method, the cross-validation method was employed to compare the models without sampling bias over the training data. The k=5 value was used for the cross-validation (80% training set, 20% validation set).

2.3 Performance measurements

Average accuracy, precision, and total runtime during cross validation were used to compare the three models in the training data over cross-validation of k=5.

3. Results

The three models were trained and evaluated using cross-validation and yielded following results. (see Table 1.)

Model	Accuracy	Precision	Time (s)
kNN	0.70487	0.65485	17.2
NN	0.70465	0.64369	936.9
LR	0.69531	0.49413	112.9

Table 1- Training data evaluation results, including the accuracy, precision, and runtime for all models in cross-validation with k=5.

Results in Table 1 shows us that the kNN and NN had very similar accuracy and precision scores. Both models outperformed the LR model. kNN was the fastest model, followed by LR and NN models.

The three models were then trained on 67% of the training data and evaluated on the 33% of the data. The confusion matrices of the resulting predictions have been plotted. (see Figure 1, Figure 2, Figure 3).

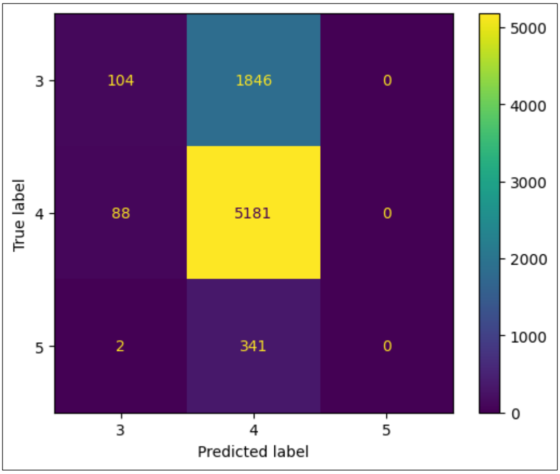

Figure 1- Confusion matrix for kNN classifier on 33% test size split on the training data.

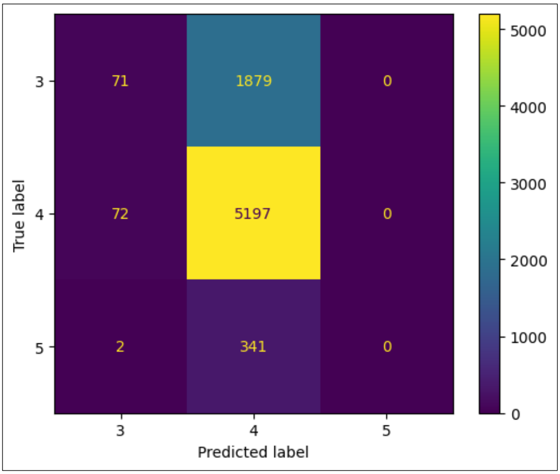

Figure 2- Confusion matrix for NN classifier on 33% test size split on the training data.

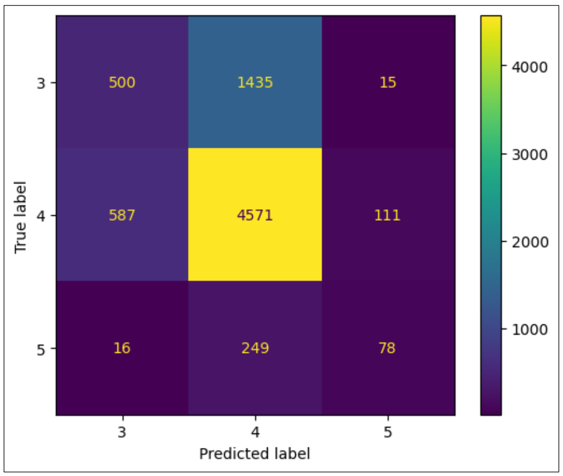

Figure 3- Confusion matrix for LR classifier on 33% test size split on the training data.

From the confusion matrices in Figure 1-3, kNN and NN had very similar results. The predictions from LR model are more evenly distributed. The other two model was somehow effective in predicting the ratings 3.0 and 4.0.

4. Discussion and Critical Analysis

Results shown in Table 1 demonstrate the performance of the three models: kNN, NN and LR. As can be seen, the three models present very similar results in terms of accuracy. The kNN model had the highest accuracy of 0.70487, indicating that around 70.49% of the predictions made by the model were correct. The NN model had similar accuracy score of 0.70465. Meanwhile, the logistic regression had slightly lower accuracy score than the other two, with around 69.5%.

In terms of precision, the kNN and NN model had a precision score of around 65.5% and 64.4%, respectively. This indicates that when the two models predicted a particular rating, around 65.5% and 64.4% of them were correct. On the other hand, the LR model yielded a lower precision score of 49.4%. This suggests the LR model had problems predicting a particular rating correctly. As mentioned above, the LR classifiers are effective when the data points are linearly separable. The low precision score of LR suggests that the true underlying model of the dataset were too complex and could not be represented as linear.

Higher accuracy of kNN and NN models support this claim as well. Both kNN and NN models are good at capturing non-linear, complex relationships and underlying structure of the data. The LR model assumes linearity, hence is not good at capturing non-linear, complex relationships. This explains the differing performance between the models.

Another important factor to note about in this project is the class imbalance in target variable in the training dataset. From the dataset, we have found that about 16100 instances were labelled 4.0, and about 5800 instances were labelled 3.0, and 1000 instances were labelled 5.0. This means that even very simple classifier like 0R, which classifies every test instance as the majority class, would give an accuracy score of around 70.29%. Since the LR model had lower accuracy score than 0R model, we remove the LR model from our considerations of the final model.

The confusion matrices of predictions made by the kNN and NN models show us that both models had a hard time predicting the rating 5.0. Both models had predicted 341 instances with a rating of 4.0, while the true label was 5.0. The kNN's inability to predict the label 5.0 might be due to the possibility that the underlying structure of the data doesn't have a clear distinct cluster for the classes 4.0 and 5.0. The clusters were probably merged, with one class (4.0) dominating the other class (5.0). Thus, the kNN model had a hard time predicting the label 5.0.

Fig. 3 shows us the confusion matrix made by the LR model. Although the predictions made by LR model seem to be more evenly distributed, the accuracy and precision scores are low. Again, this suggests the classes are not linearly separable.

On the other hand, the NN model was assumed to be effective at learning complex relationships between classes. However, that was not the case in this dataset. The NN's inability to distinguish between the classes 4.0 and 5.0 might come from the following two reasons: First, the model wasn't trained properly or wasn't suitable for this dataset. Although the parameters were optimised through exhaustive cross-validated search, there is still possibility that other parameters might yield better results. Second, the features in the dataset itself were not distinct enough for the model to learn its characteristics. In addition, the NN model requires very large datasets to perform effectively, thus, increasing the size of the training data can help overcome this issue.

Aside from the models itself, better feature selection process can help us yield better results. Both kNN and NN are very sensitive to outliers or noise in the data. Therefore, feature selection or dimensionality reduction techniques must be considered to improve the models.

Although the performance metrics are very similar between the kNN and NN models, the runtime for the training and evaluation of the models are very different. One disadvantage of NN is that it requires high computational resource to perform effectively. Meanwhile, the kNN is significantly faster and slightly more effective than NN in this case. Therefore, we consider kNN as our final model with the parameters specified above.

5. Conclusion

This works presents the results of the performance of k-Nearest Neighbours, Multi-Layer Perceptron Neural Network, and Logistic Regression models in a multi-class classification problem. They were compared based on accuracy, precision, and runtime during cross-validation to find the best performing model among the three.

From the results obtained, all models were similar in terms of accuracy. However, the precision score of LR model was lower than that of the others. Meanwhile, the runtime of the NN model was the slowest, and runtime of kNN model was the fastest. Based on its similar performance to NN model, and very fast runtime, kNN model seems to be the best performing model among the three.

6. References

Pedregosa et al., (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12,* 2825–2830.