CS361 A3 Group Project

Team members: Weidi Yan(wyan170), Joanne Chen (jehc820), Lin Lin(llin829), Cheryl Qiu(cqiu404)

Introduction

In this assignment, our task is to build BBC News article classifiers using Naive Bayes (NB), k-Nearest Neighbors (kNN), Support Vector Machines (SVM), and Neural Networks (NNs) algorithms. We are given two datasets: one for training and one for testing.

Task 1: Exploratory Data Analytics

We first loaded the 'train.csv' file into a DataFrame called df and extracted the 'Text' column. We then vectorized this column using CountVectorizer, which counts the frequency of each unique word across all the documents.

There are 428 articles in our training dataset with 13,518 unique words appearing in these articles.

Finally, we transformed the resulting count matrix into a DataFrame called count_feature_df, with the corresponding 'ArticleId' and 'Text' columns inserted. We printed out the first 5 examples in this DataFrame, from which we can clearly see the data structure: the first column is 'ArticleId', the second column is the initial 'Text', the following 13,518 columns are these unique words, and the last column is the corresponding category, either 'entertainment' or 'tech'. We recorded the frequency of each unique word appearing in the text.

We were interested to see the distribution of these unique words in all the texts in the training dataset. We plotted the top 50 most frequent words across the entire dataset and also plotted how these 50 words are distributed within the two categories. The results show that in the entire dataset, the most frequent words are a combination of common words, such as ‘said’, ‘people’, ‘new’, and ‘also’, as well as some words that may reflect the text category, such as ‘film’, ‘music’, ‘technology’, and ‘computer’.

In the articles labelled as ‘tech’, the most popular words are ‘said’ and ‘people’, but we see more occurrences of IT-related words, such as ‘mobile’, ‘technology’, and ‘software’. Words like ‘movie’ and ‘actor’ do not appear in the plot. Similarly, in the ‘entertainment’ articles, ‘film’ is the second most frequent word (‘said’ is the first here as well), and we see words like ‘music’, ‘actor’, ‘album’, and ‘movie’, yet words like ‘technology’ and ‘digital’ do not appear. Given the fact that we didn’t remove common words, the results meet our expectations based on common sense.

We also noted that in the training dataset, the distributions of the two classes are almost the same: 50.467% of the articles are labelled as ‘tech’, while 49.533% are labelled as ‘entertainment’.

Task 2: Classification Models Learning

(1) Naive Bayes classifier

For the NB classifier, we use two different methods to obtain the top 20 terms that are most likely to appear in articles across the two classes, according to our Naive Bayes classifier. According to the final result, we found that the second list describes the two classes better. Here are the reasons:

1. The first list is derived by ranking words based on the conditional probability given the class. It mainly shows which words are most common within the articles of each class, based on their frequency of occurrence in the text.

2. The second list is calculated based on maximising the ratio, which identifies the words that are not only common in one class but also not common in the other class. Making them highly distinctive for the class.

Therefore, the second list focusing on words that maximise the ratio of in-class to out-of-class probability is generally more beneficial.

(2) kNN classifier

We use TF-IDF Vectorizer to vectorize the text data from the 'Text' column of the training data. Then we reduce dimensions of the data into 2-dimensions using PCA for plotting. According to the six plots, for each distance metric, we have 3 plots using k = 1, 5, 10 respectively. When k is 10, the decision boundary is smoother, and if k is 1, the decision boundary is jagged. The three plots on the left side show that the decision boundaries tend to be spherical in higher dimensions when we use Euclidean Distance, because it measures the direct line distance between points. Meanwhile, the three plots on the right hand side demonstrate that using Manhattan Distance, the decision boundaries seem more grid-like, because it measures distance along axes-aligned paths.

(3) SVM

Similar to the classifiers built in the previous two questions, in this SVM classifier task, we extracted the 'text' column and performed the vectorization process. The target label 'category' was converted to numeric values, where 1 refers to the 'entertainment' category and 0 refers to the 'tech' category. We applied PCA transformation to reduce the original high-dimensional space to a 2-dimensional space, which allows for more explicit and clear visualisation of the data.

(3.1) Soft-margin linear SVM

We started with a soft-margin linear SVM. The hyperparameter in this case is the misclassification penalty parameter, C. The values we used were 0.1 and 10. Two surface plots were drawn accordingly to visualise the decision boundaries and the impact of the above two C values on the classification performance.

From the plots, we can see that when C =0.1, the margins are larger than when C=10. This met our expectation, as we knew that a larger C has lower tolerance for misclassification errors compared to a smaller C. A smaller value of C can make the model more generalised while a larger C may lead to overfitting.

However, in our case, we expected the two selected values to show a very straightforward difference in classification performance since there is a 100 times difference between the two values. In reality, the difference in margin widths between the two plots is small, and it is difficult to determine which one performs better as there seems to be a similar group of misclassified scatters in both plots. Therefore, we moved to the next SVM, the hard-margin RBF Kernel SVM.

(3.2) Hard-margin RBF Kernel SVM

Now we move to the hard-margin RBF Kernel SVM. The hyperparameter in this approach is the kernel width (radius) σ (sigma) of a single training data. We selected 0.1 and 1.0 to compare their different impacts. After defining σ, we calculated γ(gamma). When σ=0.1, γ=50. When σ=1.0, γ=0.5.

Two plots were drawn accordingly. From the plots, we can see that when σ=0.1, the decision boundary is highly complex, and many individual points are circled. This indicates that the model may be prone to overfitting. On the other hand, when σ=1.0, the decision boundary is smoother and less sensitive to individual training points.

(4) Neural Network
The dataset contains text data which is processed and vectorized, resulting in numerical feature vectors. As the target labels are categorical so we have encoded them to numerical values.

By the requirement of the task, below is the model configurations:
Single hidden layer unit test: 5, 20, 40
Learning rate : 0.01
Optimizer: Adam (refers to a stochastic gradient-based optimizer)
Activation function: Logistic
Loss Function: Cross-Entropy Loss (implicity used by 'MLPClassifier' function)

The plot shows that as the number of hidden units increases from 5 to 20, the average training cross-entropy loss decreases significantly. However, when the number of hidden units increases from 20 to 40, the decrease in average training loss is much smaller.

This observation shows that the hidden units from 5 to 20 allow the model to better capture the patterns in the data, however, beyond 20 units does not provide a significant improvement. This might be due to the neural network already having enough capacity to model the data, and adding more hidden units may lead to overfitting.

Task Three: Classification Quality Evaluation

A) Impact of the size of the training data set on accuracy

(1) Naive Bayes
For NB, the F1-score of the training set increases with the increase of the training set. This means that when the training set increases and more data is used for modelling, the performance of the model will be improved, so the F1-score of the validation set will be improved. For the validation set, the highest F1-score is 1.0 when 90% of training data is used to train the model. In the test set, F1-score also showed an increasing trend with the increase of the training set, which proved that the more data the model used in training, the model performance could indeed be improved. The highest F1 score is when 99.17% 90% of training data is used to train the model. It also shows that the model may not be overfitted.

(2) KNN
For KNN, we observed an increasing trend as well, which might indicate the training model with a larger data set will perform better on the testing data. Overfitting might not be a concern in this case. When we use 90% of training data to train the classifier, we get the highest F1-score with 0.8846 on the validation dataset and we get the highest F1-score with 0.9104 on the unseen test dataset.

(3) SVM

(3.1）soft-margin SVM
We set c=0.1 as the fixed hyperparameters in this question. The f1-score of the validation set increases with the size of the training set. When we train the model using 90% of the training set, the f1-score of the validation set can reach 1. The f1-score of the test set has reached 1 at 80% of the training size.

(3.2）Hard-margin RBF Kernel SVM
We set sigma=1 as the fixed hyperparameters for Hard-margin RBF Kernel SVM. The overall trend for both training and test accuracy (f1-score) increases with the size of the training set. When the training set size is 90%, the validation set f1-score reaches the highest 0.74, and the test set reaches 0.76. In addition, the f1-score of

the validation set and test set increases significantly when the training set size is higher than 70%.

(4) Neural Network

The validation F1 scores plot a clear increase trend as the fraction of the training data increases. At m = 0.1, the performance is lowest (F1 score = 0.9637), suggesting that the classifier is limited to learning from a small portion of data. As more data is included (m=0.3 to m=0.9), there is a consistent improvement in performance. When m = 0.9, the highest F1 score is 1.0.

In the test set, the plot also shows an increasing trend, but introduces a plateau starting around m = 0.5. Similar to the validation scores, the test score improves as more training data is used. The F1 score from 0.9621 to 1.0 corresponds to m = 0.1 to m = 0.9.

The general trend in both plots is an increase in F1 score with more training data, showing improved classification accuracy as the classifier has access to more examples during training. The plateau observed in test F1 scores could suggest that for this model and dataset, increasing the training data beyond a certain threshold might not benefit performance.

B) Impact of key hyperparameters

(1) Naive Bayes

For Naive Bayes Model, Alpha is the key Hyperparameters and we investigate the impact of Alpha. We do the 5-fold cross validation for NB with Laplace Smoothing (Alpha = 1) and without Laplace Smoothing (Alpha = 1e-10, a very small number) respectively. The result of validation shows that NB with Laplace Smoothing performs better than NB without Smoothing. However, when we use the test data to evaluate, the NB classifier without Laplace Smoothing leads to a higher test F1-score with 0.9917, while the NB classifier with Laplace Smoothing with test F1-score with 0.9748.

(2) KNN

In this part, we compare 10 different values of k, ranging from 1 to 10., using the 5-fold validation method because in kNN classification, k is the hyperparameter that can be tuned. For each value of k , we apply 5 groups of training folds to train the kNN classifier and then calculate the validation accuracy using the corresponding validation fold respectively. The plot shows the trend is decreasing as k increases. We conclude that the best k is 1 with the best mean validation F1-score 0.82. Therefore, we conclude that the best k is 1. Finally, we use the unseen test set to test the kNN classifier with k=1 and we get the result with 0.9057 test accuracy in F1-score.

(3) SVM

(3.1）Soft-margin SVM

To compare the impacts of different C values on the Soft-margin SVM, we selected the values 0.001, 0.01, 0.1, and 1. We performed each classifier with one of the above values on the training data using 5-fold cross-validation. From the results, we can see that there is no major difference between these Soft-margin SVMs with different C values. When C=0.001, the F1-score is 0.97. For the rest of the C values, the F1-scores are all 0.98. We decided to use C=0.01 and applied this to the test data. The resulting F1-score is 1.00. The F1-score seems unusually high, however, considering there are only 106 records in our test dataset, it is possible to achieve such a high F1-score.

(3.2）Hard-margin RBF Kernel SVM

We selected a range of different σ values for this task: 0.1, 1, 10, 100, and 10,000. We first performed the hard-margin RBF Kernel SVM with these σ values on our

training dataset using 5-fold cross-validation. We found that the F1-score increased as the value of σ increased. The best result was when σ=10,000. Hence, we applied σ=10,000 when performing the model on the test data. The F1-score was 1.00.

(4) Neural Network

We choose the hidden unit as the key hyperparameter to investigate the impact of the NN classifier, while other hyperparameters remain the same. Five different values of the hidden unit, including [5, 10, 20, 40, 60] are chosen to test on the validation dataset. The line on the graph shows some fluctuations, with the lowest point occurring when the hidden unit is set to 10 and the highest point occurring when the hidden unit is set to 40. When we have 40 hidden units, the F1-score is the highest, which is 0.9813 on average for the 5-fold cross validation set. When we have 40 hidden units, the F1-score is the highest, which is 0.9673 on average for the 5-fold cross validation set. Therefore, we conclude that the best hidden_unit is 40 and the F1-score for the test dataset using NN with 40 hidden units is 1.0.

C) Compare NB, kNN, SVM and NN classifiers with the best hyperparameter

We evaluate the performance of four classifiers: Naive Bayes (NB), k-Nearest Neighbors (kNN), Support Vector Machine (SVM), and Neural Network (NN) on the given test dataset.

For **NB**, we choose Alpha as the key hyperparameter to evaluate. The best Alpha is extremely small, equal to 1e-10. When Alpha is 1e-10, the accuracy in F1-score measure on the test dataset is 0.9917.

For **kNN**, we select k as the key hyperparameter to evaluate. The best k is 1, and when k is 1, the accuracy in F1-score measure on the test dataset is 0.9057.

For **Soft-margin SVM**, we choose C values as the key hyperparameter to evaluate. The best C value is 0.01. When C value is 0.01, the accuracy in F1-score measure on the test dataset is 1.00.

For **Hard-margin RBF Kernel SVM**, we choose σ values as the key hyperparameter to evaluate. When sigma is 10000, the model performing best, the accuracy in F1-score measure on the test dataset is 1.00.

For the Neural **Network**, we choose hidden_unit as the key hyperparameter to evaluate. The best sigma is 40. When a NN classifier with a single hidden layer of 40 units and logistic activation function provided optimal results the accuracy in F1-score measure on the test dataset is 1.00.

To conclude, the Neural Network (NN) classifier , Hard-margin RBF Kernel SVM and Soft-margin SVM demonstrated the best performance on the testing dataset with an F1 measure of 1.0, followed closely by the Naive Bayes (NB) classifier. These results suggest that for this particular classification task, more complex models like NN and SVM, which can better capture the patterns of the data, are more effective than simpler models like NB and kNN.