COMP24112 – NEWS ARTICLE CLASSIFICATION

Ayesha Tabrez

EXPERIMENT 1

In this experiment I performed k-NN classification using the Euclidean and Cosine distance setting k = 3. I observed that the accuracy of the k-NN classifier using the Euclidean distance measure was lower than the accuracy using Cosine distance while STD using Euclidean was higher than Cosine.

The higher accuracy using the Cosine distance measure indicates that it was better at capturing the similarity between the documents and classifying them into the correct categories while the low standard deviation indicates a more consistent and stable performance of the k-NN classifier.

This may be due to the fact that even if two similar documents are far apart by the Euclidean distance because of the size they could still have a smaller angle between them. The smaller the angle, higher the similarity. Thus, **Cosine distance measure is more effective** in capturing the semantic similarity between the documents based on their word frequencies, even if they have different magnitudes (lengths).

Experiment 2

The **training and testing accuracies differ** because they work on **different datasets**. The training accuracies are generated by testing the data on the same samples it is trained on. Hence it is not as relevant as training data on a different data set. I observed that when k = 1, training accuracy is 100%. This is because when the training data is tested against itself, the closest point to each value of training data is itself but testing accuracy differs because input is not in training set now. I observed that on average training accuracy is higher than testing accuracy because the **average training error rate is slightly lower than the average testing error rate**. One reason would be that **testing the data on the same set cannot capture overfitting** of the data, since it will learn and predict the patterns correctly when analyzed on itself again. Moreover, testing errors are fluctuating more, which is a good indicator of relevance.

When **k gets bigger**, error increases, hence **average training and testing accuracy decreases** whereas standard deviation increases. The effect of k on the model's performance can be analyzed by observing the model's bias and variance. When **k is small,** the bias for training data is low. As k increases, bias and variance for training data increases. For testing data, variance remains constant no matter what the value of k is.

A suitable value of k should balance the trade-off between bias and variance. For this dataset, the best values of k tend to be very close to 1 and **k = 3 appears to be a reasonable choice** for achieving the highest accuracy. This is because when k is set to 1, although accuracy is highest, the algorithm only considers the closest neighbor to the data point being classified. This may result in overfitting, as the classification could be heavily influenced by noise or outliers in the training data. On the other hand, setting k to a higher value, such as 3, would mean that the algorithm considers more neighbors when classifying a data point. This can lead to more stable and accurate predictions, as the classification is less likely to be affected by individual data points.

Experiment 3

Based on my understanding, all 5 articles belong to the class - **'sports'**. The classifier's predictions seem to be reasonable since there is no 'sports' label added yet, hence no training data for 'sports'. Since the classifier is not trained to recognize and classify articles belonging to 'sports' it is likely that it will classify them into one of the existing classes in the training data. Thus, the accuracy of the classifier is high, around 96%, which suggests it is performing well.

The experiment used 100 training samples for the first 4 classes and only 3 for the 'sports' class. The classifier performed well for the first four classes, achieving an average testing accuracy of around 96%, but only around 60% for the 'sports' class. This result indicates that the classifier is not able to generalize well for this class and that **additional training and testing data may be needed to improve its performance.**

Having only 2 test labels for the sports class means that the accuracy for this class may not be as reliable as the accuracy for the other classes. With such a small sample size, the accuracy could be either overestimated or underestimated.

The consequences of having no training data and limited training data for the 'sports' class are that the classifier has **not learned enough** about the features that distinguish between different types of sports news. As a result, the classifier may **not be able to accurately classify** new sports articles that it has **not seen before**. This problem could be addressed by collecting more training and testing data for the ‘sports’ class.

Zero-shot learning and few-shot learning deal with classification when there is limited labeled data available for training. Zero-shot learning involves classifying objects that have never been seen before by relying on prior knowledge or learning a mapping. Few-shot learning trains models to classify new classes with limited labeled examples. In the experiment, my k-NN classifier was trained using a limited number of examples from the sports class, making it a few-shot learning problem. The model was trained with only three examples and tested on only two examples, significantly less than what is required for traditional machine learning models. Thus, our model is learning to classify the sports class with very limited examples, making it a **few-shot learning problem.**

ANALYSIS 1

I have used the equation $alpha = z_p \cdot \sqrt{\frac{error_s \cdot (1-error_s)}{n}}$ to compute the confidence interval.

First, I set sample size to 480 (n) because I used 480 testing samples in experiment 2 and determine the value (z_p) for a 90% confidence interval using a z-score table. The mean error for the chosen trial ($error_s$) when k = 1 is retrieved using $test\_err\_mean[0]$ from experiment 2. On replacing all these values in the formula, I obtain the margin of error ($alpha$) for the confidence interval. Finally, the lower and upper bounds of the confidence interval for the mean error, based on the chosen trial and the estimated margin of error are given by $error_s - alpha$ and $error_s + alpha$ respectively.

Therefore, the estimated 90% confidence interval for the true error of the 1-NN classifier on the chosen fold is [0.017, 0.0466]. This means that we can be 90% confident that the true error of the 1-NN classifier on the chosen fold lies between these two values.

ANALYSIS 2

First, I retrieve the testing error for the 45-NN trial using $test\_err\_mean[44]$ from Experiment 2. I compared this with the testing error of the 1-NN to see which value is higher. I observed that 45-NN has a higher testing error. To calculate the probability that It also has a higher true error I used the z-test.

Step 1: I computed a quantity z_p using the formula: $\frac{d}{\sigma}$, where $d = |error_{s1} - error_{s45}|$ and

$$\sigma = \sqrt{\frac{error_{s1}\,(1 - error_{s1})}{n_1} + \frac{error_{s45}(1 - error_{s45})}{n_{45}}}$$

Step 2: I used the function $Get\_p\_value(\quad)$ to obtain p according to z_p.

Step 3: The final probability is then computed using the formula $C = 1 - \frac{(1-p)}{2}$

Based on my code, the estimated probability that 45-NN has a higher true error is 0.97, which is a high probability. This means that it is highly likely that 45-NN has a higher true error than 1-NN. This is because when k =45, the number of neighbors increases, false positive increases and therefore the prediction gets polluted.

<u>Hyperparameter Selection</u>

In this project, the data was split into training and testing sets using a random subsampling approach where I chose 40 splitting segments with size 12. A subset of the data was randomly sampled for testing, and the remaining data was used for training. Thus, resulting in 480 testing samples and 320 training samples. In random subsampling, there is no fixed size for the testing samples of each class. Therefore, the number of samples for each class in the testing and training set may vary, and there is no guarantee that the testing and training set will have an equal representation of all classes. To obtain a more reliable estimate of the classifier's performance, I repeated the process 20 times.

The hyperparameter selection method involved varying the value of k (ranging from 1-50), which is the number of nearest neighbors to consider in the k-NN algorithm. The process was repeated 20 times for each value of k, and the value that resulted in the lowest mean testing error rate was selected as the optimal hyperparameter value.

The best value of k I obtain is k=1. The results suggest that my random subsampling approach performs well in this particular classification task. The high overall accuracy of around 96% indicates that the classifier is able to accurately predict the class labels of the testing data. The confusion matrix also shows that the classifier is able to differentiate between the four classes, as evidenced by the relatively high number of true positives and low number of false positives and false negatives for each class.

While k-fold cross-validation is a more commonly used approach for hyperparameter selection, I used random subsampling due to its simplicity and ease of implementation. However, k-fold cross-validation may be preferred in situations where the data is highly imbalanced or where the size of the dataset is small.

Splitting the data into train, test, and validation sets is crucial in machine learning experiments to **evaluate model performance, prevent overfitting, and tune hyperparameters**. The test set is used to evaluate the model's performance on new, unseen data, and the validation set is used to choose the best hyperparameters for the model. Overall, splitting the data ensures that the model is performing well on new data and is not **overfitting** to the training data.