1. INTRODUCTION

With ever-growing access to the internet, the human race has gained abilities to find information and articles to obtain knowledge that used to take much longer to find. The days of library books and endless searching through myriads to physical archives for information are now gone. With such a large amount of information that only continues to expand, classification is required to organise articles, blogs, etc. The AG News Topic Classification Dataset is a multitude of news articles that have been gathered from over 2000 news sources. This assembly was completed by the academic news search engine ComeToMyHead after over a year of activity. The original dataset consists of classes 1-4 where 1-World, 2-Sports, 3-Business, and 4-Sci/Tech [1]. In this study, we use a variety of different models to determine adept methods of accurate classification of news articles based on keywords. We will compare the use of deep learning models and unsupervised machine learning.

Though both subsets of machine learning, the key difference between the two lies in how they handle data to create their classification algorithms. Unsupervised machine learning learns by identifying patterns in the data it is introduced to [2]. Deep learning employs sophisticated neural networks to learn from the data with barely any human intervention [2]. In this study, we use the Keras Library in Python to employ the Long Term Short Memory and Convolutional Neural Network deep learning models to produce a classification algorithm for the dataset. In addition to that, we investigate the effectiveness of K-Means Clustering and Hierarchical Agglomerative Clustering. The two unsupervised machine learning models were implemented from the Sci-Kit Learn library.

2. METHODOLOGY

2.1. DATA PREPROCESSING (SEE APPENDIX 1)

The data was first loaded into Python using Pandas. The 'Title' and 'Description' columns were combined, forming a single text feature. Those two columns were subsequently dropped to streamline the analysis. In order to reduce computational expense, the size of the dataset was decreased in its size. The class values were extracted and stored in a separate data frame for both training and testing datasets. The combined text field was then converted to lowercase, and contracted words were expanded using regular expressions. The final results were exported to CSV files to prevent the need to preprocess the data again at a later state.

2.2. MODEL 1 – K MEANS CLUSTERING (SEE APPENDIX 2)

The pre-processed data was converted from tokenised text to a single string per document. The TF-IDF Vectorizer from the Sci Kit library was then used to transform the text into a high-dimensional feature space. The vectoriser was configured to process up to 10000 features, get rid of English stop words and consider unigrams and bigrams.

Clustering was performed across a range of 2 to 14, and for each k value, the inertia was computed to assist in ascertaining the being k value using the elbow method. The elbow method is used to determine the best number for k in K means clustering. It involves computing the inertia for each varied k value and then plotting the inertia against the k values. The best k value is selected at the point on the graph that looks similar to the apex of a bent arm [3]. The K value selected was 9. Using Principal Component Analysis, the final clustering was visualised, and the silhouette score was observed.

2.3. MODEL 2 – MULTI LEVEL PERCEPTRON (SEE APPENDIX 3)

In this part of the study, preprocessed training and testing datasets were loaded from CSV files and refined by removing the 'Class Index' column from the features. A custom function converted stored token lists into strings with uniform whitespace, preparing the text for analysis. The cleaned text was then tokenized with Keras' Tokenizer (limited to 10,000 words) and padded to a fixed length of 100 tokens. The conversion to string format was done using the ast.literal_eval() helper function [4].

A Multi-Layer Perceptron (MLP - Convolutional Neural Network) was constructed for five-class classification. The classification was done for five classes because the class labels were between 1 to 4. The model begins with a Flatten layer, followed by two Dense hidden layers with hyperparameters—number of neurons and activation functions—selected from [20, 40, 60, 80] and ['relu', 'sigmoid', 'tanh'], respectively. The output layer uses softmax activation, and the model is compiled with the Adam optimizer and sparse categorical cross-entropy loss.

Hyperparameter tuning was performed using Keras Tuner's Hyperband algorithm on an 80/20 training/validation split over 10 epochs. The best-performing model was identified and subsequently retrained for 10 additional epochs. The final evaluation of the test set confirmed the model's robust generalization performance. The Keras tuner was used as an alternative to GridSearchCV to parse over the selected values for the hyperparameters to determine the best ones.

2.4. MODEL 3 – LONG SHORT TERM MEMORY (SEE APPENDIX 4)

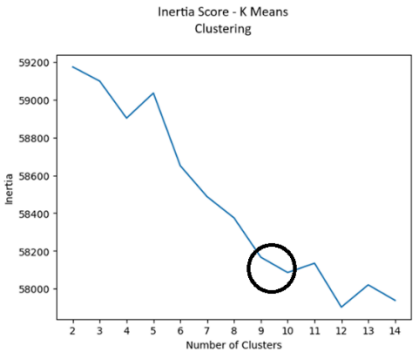
The model begins with an LSTM layer that uses the padded sequences that were reshaped to be made acceptable by the LSTM layer. It is proceeded by two dense layers whose hyperparameters were a number of neurons and activation functions. Hyperparameter tuning was achieved using the Keras Tuner's Hyperband algorithm. It explored a variety of activation functions and varied the number of neurons in the dense layers of the network. It parsed over the hyperparameters in each trial for up to 10 epochs with an 80/20 validation split. The set-up of this portion of the study was very similar to that of section 2.3.

2.5. MODEL 4 – K MODES CLUSTERING (SEE APPENDIX 5

A variation of K means clustering was employed using K modes clustering. This differs from K means where it uses the matching categories of data points instead of their distances.

3. RESULTS/DISCUSSION

3.1. MODEL 1 – K MEANS CLUSTERING



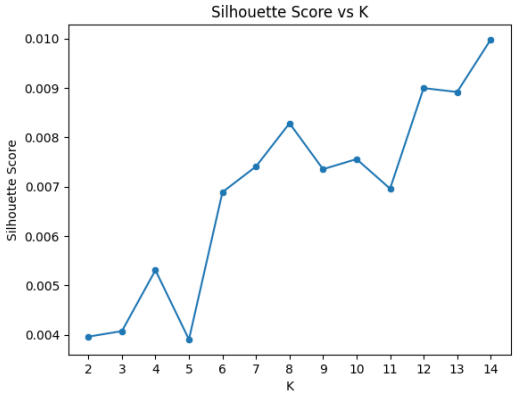


Figure 3.1.1. – The above graph on the left depicts the inertia score for each selected number of clusters. The optimal selection ranging between 9 and 10. The graph to the right shows the silhouette score for each k.

Figure 3.1.1. depicts the analysis of parsing through various k values for the K-Means Clustering unsupervised machine learning model. We trained the model for values ranging from 2 to 14 and measured their inertia and silhouette scores. On the left, we observed the general trend of the inertia decreasing as the number of clusters increases. Inertia is the sum of squared distances within each cluster. This measurement decreases because as more clusters are formed, the distance between data points and centroids decreases, allowing for points to be grouped more locally [5]. Through the elbow inspection method, we determined that the best range for k values was approximately 9 or 10. We observe a slight elbow-like bend around that point on the graph to the left in Figure 3.1.1. The silhouette scores depicted on the right of Figure 3.1.1 are considerably low from a general perspective. Low silhouette scores can be attributed to clusters not being well separated or the data inherently not being naturally able to form clusters. For categorical data, K means clustering is not the best approach, as categorical variables will not contribute much to the distance from the mean. An alternative approach could be the use of K modes clustering, which clusters based on the number of matching categories of data points [6].

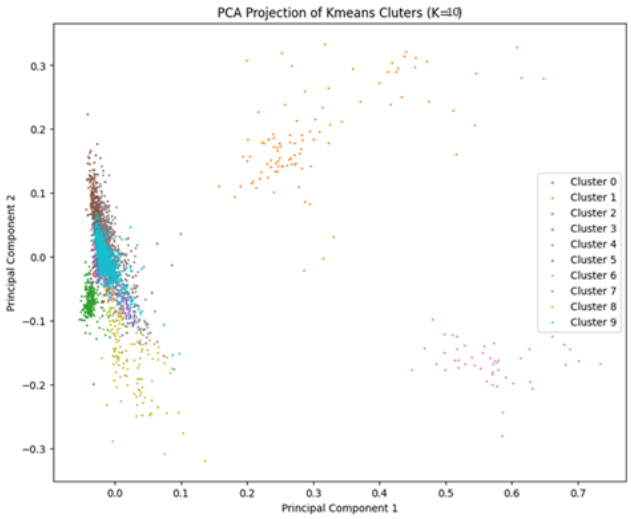


Figure 3.1.2. – The figure depicts a two-dimensional PCA projection of the data, where each point is colour-coded based on its allocated cluster from the K-Means Clustering.

After initial training, the model was tested on a new set of data where the number of clusters was 10, as ascertained by the elbow method. The Silhouette score calculated was extremely low at 0.0068, thus substantiating the unsuitability of K-Means Clustering for this particular categorical dataset. For reference, a silhouette score closer to one indicates that points are more effectively matched to their clusters and poorly matched to their neighbouring clusters. Figure 3.1.2. depicts a PCA visual of the predictions to their clusters in a 2-dimensional space. There is a lot of overlapping, and the clusters are very close to each other, indicating that many data points fail to group neatly in a higher-dimensional setting. In summary, the low silhouette score and overlapping data points indicate the data does not cluster in a well-separated manner.

3.2. MODEL 2 – MULTI LEVEL PERCEPTRON (CONVOLUTIONAL NEURAL NETWORK)

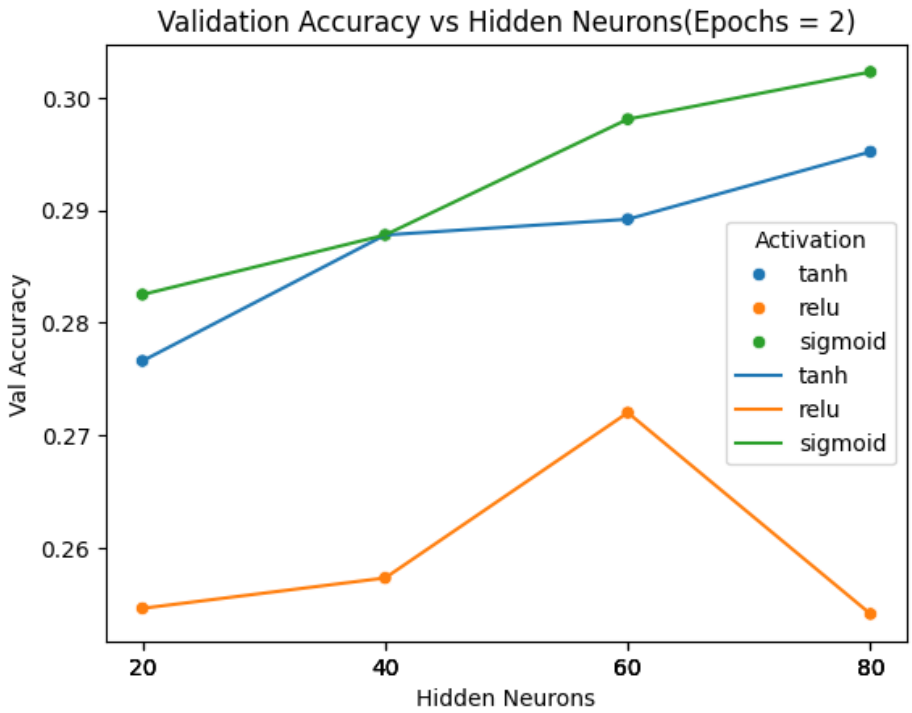


Figure 3.2.1. – The above chart shows the relationship between number of neurons in the hidden layer against the validation accuracy for each activation function for a number of two epochs.

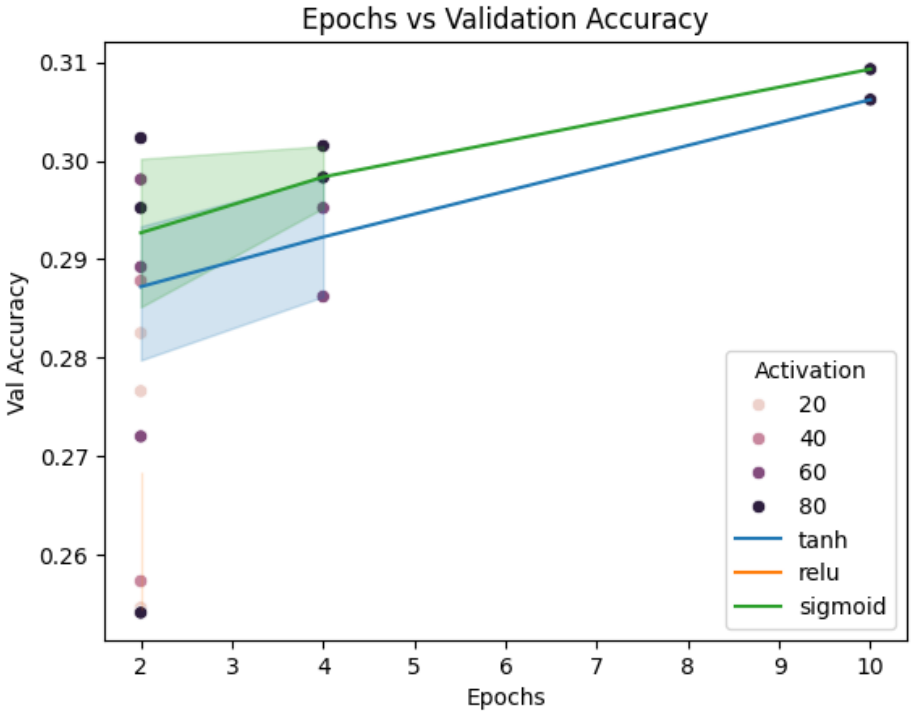


Figure 3.2.2. – The above graph shows the relationship between the validation accuracy and epochs for each activation function.

The Keras tuner was used to parse through various hyperparameters, which included the following activation functions: Relu, Tanh and Sigmoid. Initial results showed that the Relu activation function performed with the lowest sets of accuracies (please refer to Figure 3.2.1). The Keras tuner uses a

performance-based approach to parse through the hyperparameters it is given. As a result, its focus shifted away from the Relu activation function due to this pruning strategy. The

The sigmoid function has the highest validation accuracy and only increases with the increase of neurons. We can observe a similar trend with the tanh and Relu activation functions, except that the validation accuracy plummets for the Relu function at a higher number of neurons. Figure 3.2.1 depicts the relationship between the number of hidden neurons and the validation accuracy. The sigmoid function may be held to an advantage due to its use in a shallow network. This gives it a slight head start and allows it to perform better in fewer epochs. The Relu activation function tends to perform better in a deeper network, but in this instance, it may have been 'left behind' because of the short training window or limited hyperparameter combinations. The Tanh function is a zero-centred approach and tends to yield stable convergence in earlier epochs [7, 8].

Figure 3.2.2 shows the validation accuracy of the model against the number of epochs for each activation function. Generally, it shows an upward trend for all lines as the number of epochs increases. The lines representing the tanh and sigmoid activation functions show an increase in their accuracy as the number of epochs increases. Similarly, as the number of hidden neurons increases, the two models' performances improve. Given the Keras Tuner's property to parse through better-performing hyperparameters, it may be worthwhile to optimise the structure of the neural network and make it deeper [9].

The best-performing model had 80 hidden neurons in the hidden layers of the model and used the sigmoid activation function. It was trained over 10 epochs and obtained a validation accuracy of 0.3093. These parameters were then used to make predictions using the test data. An accuracy of 0.3161 was observed.

3.3. MODEL 3 – LONG TERM SHORT MEMORY

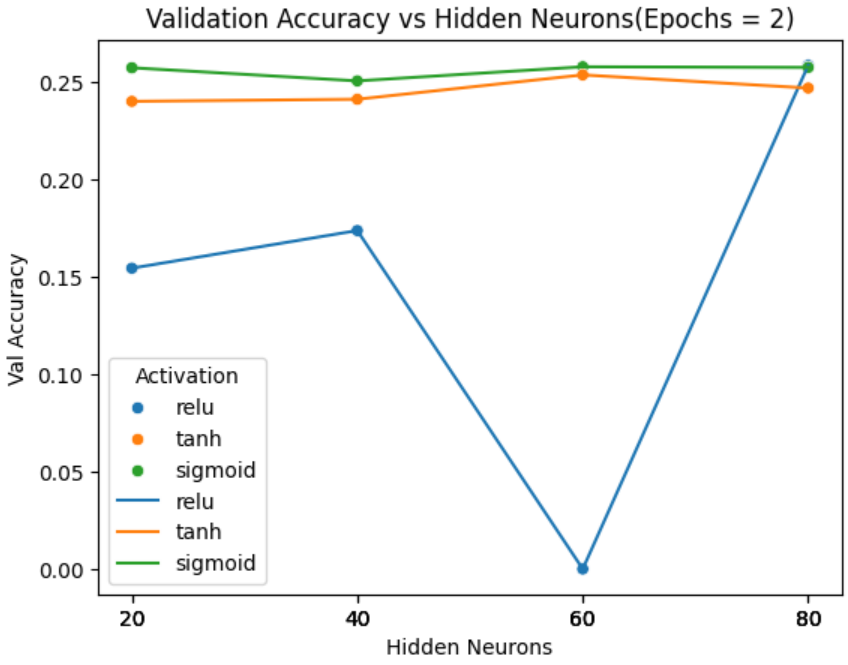


Figure 3.3.1 – The above plot details the relationship between the number of hidden neurons and the validation accuracy of the model.

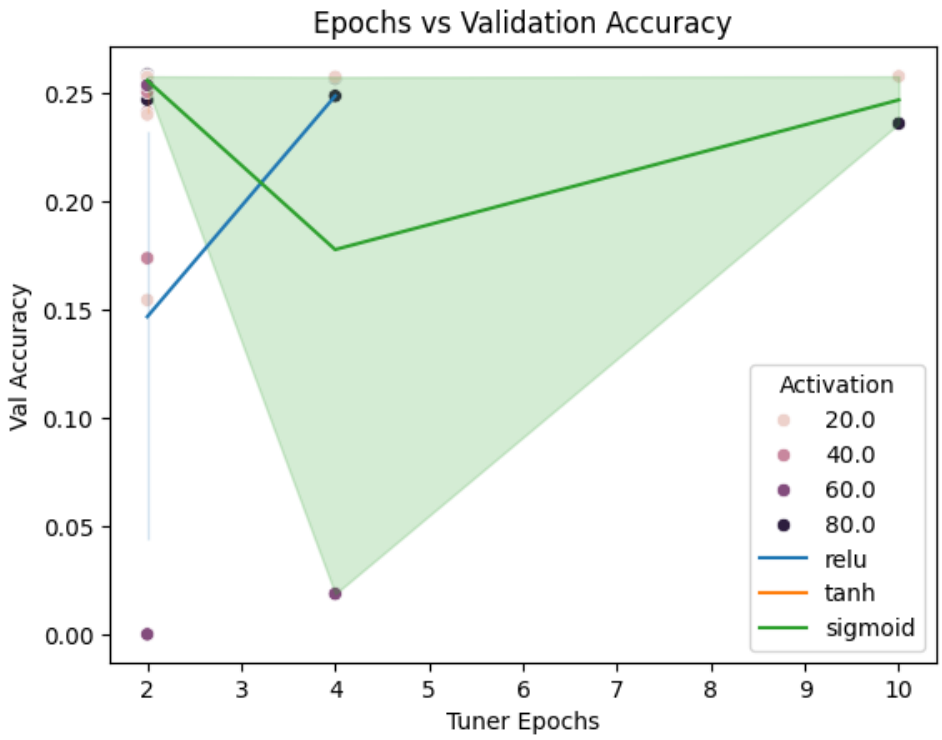


Figure 3.3.2. – The above plot depicts the relationship between the number of epochs and validation accuracy for each activation function and number of neurons in the hidden layer.

This model varies from the initial CNN in which a long-term short-memory layer is introduced at the beginning of the neural network. These experiments look at how to optimise the LTSM-based model for text classification tasks. Various hyperparameters were tuned, such as the LTSM units and a suite of activation functions (Relu, sigmoid and tanh). The LTSM layer comprises 'units', which refer to the specialised neurons with internal gating properties. These properties (input, output, forget gates) help the network handle sequential data and dependencies over time. These 'units' allow the LTSM layer to handle time-series data such as text.

The graphs in Figures 3.3.1 and 3.3.2 offer different insights into the model's performance. Figure 3.2.2 highlights the validation accuracy against the number of epochs, comparing the performance across different activation functions and a number of units. It depicts how the use of tanh and sigmoid activation functions achieves a higher validation accuracy with an increasing number of epochs. The real function appears less stable in earlier epochs due to its non-saturated nature. Figure 3.2.1 shows the relationship between validation accuracy and the number of LTSM units with the number of epochs fixed at 2. There is a slight improvement in validation accuracy for the tanh and sigmoid activation, but they are generally more stable than the real function. This may be because the Relu function requires a deeper neural network to reach convergence. The performance of an LTSM model is very clearly also linked to the activation function used; hence, it is very important to tune the model with the appropriate activation function [11].

The best-fitting model used the sigmoid function over 10 epochs with 80 units. The accuracy recorded when the model was used to make predictions against the test data was 0.1466. To improve this score, better preprocessing techniques, such as the removal of elements like commas and numbers, could be employed. This would lead to greater uniformity in the data, making it more feasible for the LTSM model [10].

3.4. MODEL 4 – K MODES CLUSTERING

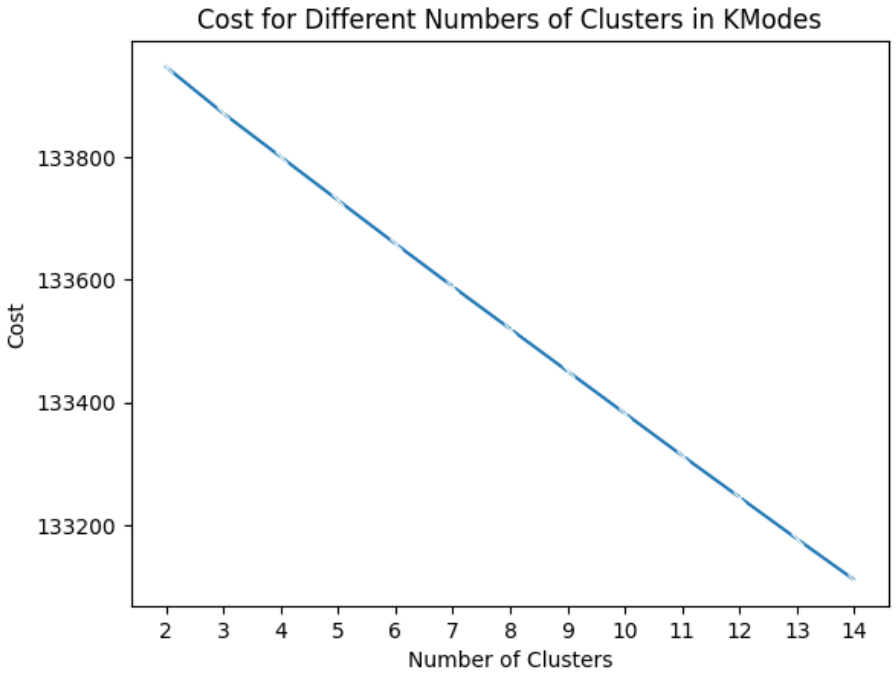


Figure 3.4.1 – The above graph shows the costs for the k modes clustering.

Number of Clusters (k)	Silhouette Score
2	-0.00593
3	-0.00939
4	-0.01183
5	-0.01222
6	-0.01373
7	-0.01455
8	-0.0153
9	-0.01567
10	-0.01656
11	-0.01751
12	-0.01809
13	-0.01826
14	-0.01958

Figure 3.4.2 – The above table shows the silhouette score for each k value.

The cost (within-cluster dissimilarity) steadily decreased with increasing k, but the silhouette scores were consistently negative (ranging from -0.00593 for k=2 to -0.01958 for k=14), indicating that the clustering structure may not be well defined, with articles potentially being closer to other clusters than to their own.

These results suggest that applying KModes—an algorithm designed for categorical data—to numeric TF-IDF vectors from the AG News Collection may not be optimal. The absence of a clear "elbow" in the cost curve coupled with negative silhouette scores implies that the current clustering configuration does not effectively capture the underlying structure of the dataset. As the scores were uniformly negative and the cost was linearly decreasing, an optimal k value was not selected. It was expected that the K modes method would be better equipped to handle the categorical nature of the dataset. This was contradicted by the results as a clear elbow was not found, as depicted in figure 3.4.1.

4. CONCLUSION

In conclusion, the analysis conducted between the unsupervised clustering models and deep learning models yields mixed results. For the k-means clustering, the inertia decreased steadily until a relatively distinct elbow was observed. In comparison to the K modes method, this was not observed as a clear elbow was not observed. This highlights how important the data type is of the categorical data despite it being specifically designed for that type of data. It did not provide a clear elbow for numeric TF-IDF data.

The neural networks that were studied yielded better performance than the unsupervised learning models. While the models achieved humble accuracy levels, their performance emphasises the value of effective preprocessing, hyperparameter tuning and model architecture selection. The results imply that while unsupervised approaches may provide insights into data structure, supervised neural networks may be better suited to handling the variability and intricacies of text-based data. Future work should explore refining the preprocessing pipeline and exploring alternative feature representations such as embedding or topic modelling.

5. BIBLIOGRAPHY

1. (2015). AG News Classification Dataset. X. Zhang.
2. Holdsworth, J. and M. Scapicchio (2024). "What is deep learning?". from https://www.ibm.com/think/topics/deep-learning#:~:text=Some%20form%20of%20deep%20learning,driving%20cars%20and%20generative%20AI.
3. Tomar, A. (2025). "Elbow Method in K-Means Clustering: Definition, Drawbacks, vs. Silhouette Score." from https://builtin.com/data-science/elbow-method#:~:text=The%20elbow%20method%20is%20a%20graphical%20method%20for%20finding%20the,on%20the%20x%2Daxis)
4. . "ast — Abstract Syntax Trees." from https://docs.python.org/3/library/ast.html.
5. Rykov, A., et al. (2024). "Inertia-Based Indices to Determine the Number of Clusters in K-Means: An Experimental Evaluation." IEEE Access **12**.
6. HUANG, Z. (1997). "CLUSTERING LARGE DATA SETS WITH MIXED NUMERIC AND CATEGORICAL VALUES."

7. Glorot, X. and Y. Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks." Proceedings of Machine Learning Research **9**.
8. Goodfellow, I., et al. (2016). Deep Learning, MIT Press},.
9. Nair, V. and G. E. Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning.
10. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780.
11. Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)

5.1. APPENDIX

Appendix 1 – Preprocessing code and pre-processed data (see accompanying files)

Appendix 2 – Code for K means Clustering

Appendix 3 – Code for Multi-Level Perceptron (CNN)

Appendix 4 – Code for Long Term Short Memory (LTSM)

Appendix 5 – Code for K modes clustering