Sentiment Analysis Assignment Report

1 - Executive Summary

The objective of this assignment was to perform sentiment analysis on a dataset using both conventional machine learning (ML) and deep learning models. The dataset consists of reviews that needed to be preprocessed using NLP techniques and represented using N-gram models.

2. Data Understanding and Preparation

2.1 - Data Description

The Cornell Sentence Polarity Dataset consists of snippets of movie reviews collected from Rotten Tomatoes webpages. The dataset contains a total of 10,662 snippets, evenly distributed between positive and negative classes, with 5,331 snippets labeled as positive and 5,331 labeled as negative. The purpose of this dataset is to facilitate sentiment analysis tasks, where the goal is to classify text data into positive or negative sentiment categories based on the content of the reviews.

2.2 - Data Preparation using NLP techniques

Tokenization and Stopword Removal:

- *Tokenization was implemented using the ToktokTokenizer from the nltk.tokenize module.*
- *Stopwords were removed from the text using a predefined list of English stopwords from the nltk.corpus.stopwords corpus.*

Stemming & Lemmatization:

- *Stemming was carried out using the PorterStemmer from the nltk.stem.porter module*
- *Lemmatization was performed using the WordNetLemmatizer from the nltk.stem.wordnet module.*

Tf-Idf Conversion:

- *This conversion process transformed the text data into a matrix of TF-IDF features, where each row represents a document (review) and each column represents a unique term or n-gram.*

3 – Conventional ML Model

3.1 – Model Building

The conventional ML model building process involved training several classifiers on the preprocessed text data. Initially, four models were trained: Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and Random Forest. Later, an advanced model, LightGBM, was also trained. The following summarizes the performance of each model:

	Training Acc	Validation Acc
Logistic	0.742760	0.661978
Tree	0.758471	0.581810
KNN	0.857779	0.573840

Random Forest: *The confusion matrix for the training dataset showed 2520 true negatives, 1742 false positives, 881 false negatives, and 3386 true positives. The AUC of ROC on the training dataset was 0.6924, and on the validation, dataset was 0.6336.*

LightGBM: *The confusion matrix for the training dataset showed 3428 true negatives, 834 false positives, 1241 false negatives, and 3026 true positives. The AUC of ROC on the training dataset was 0.7567, and on the validation, dataset was 0.6619.*

3.2 – Model Evaluation

The performance metrics, including confusion matrices, ROC AUC scores, and accuracies, were reported for each model on both the training and validation datasets. After hyperparameter tuning, the LightGBM model showed improved performance, with a training accuracy of 75.43%, a validation accuracy of 66.43%, a ROC AUC of 0.8185 on the training dataset, and a ROC AUC of 0.6549 on the validation dataset.

4 – Deep Learning Model (FCFNN)

4.1 - Data Preparation

Conversion to Sparse Tensor:

The TF-IDF matrix obtained from vectorization was converted into a sparse tensor.

This conversion was achieved using the tf.SparseTensor() method along with the .tocoo() method to convert the TF-IDF matrix to a coordinate format.

Sparse Reordering:

After obtaining the sparse tensor, sparse reordering was performed to ensure that the input is in row-major format, which is required by TensorFlow. This was done using the tf.sparse.reorder method.

4.2 - Basic Neural Network Model

Model Architecture:

- *Input Layer: 500 features*
- *Hidden Layer 1: 256 neurons*
- *Hidden Layer 2: 64 neurons*
- *Output Layer: 1 neuron*
- *Total trainable parameters: 144,769*

Performance Metrics:

- *Training Accuracy: Gradually increases from 66.55% to 97.17% over epochs*
- *Training Loss: Decreases from 0.61 to 0.07 over epochs*
- *Validation Accuracy: Fluctuates around 64.32% over epochs*

- *Validation Loss: Increases from 0.62 to 2.78 over epochs*

Observations:

- *The model shows significant overfitting, as evidenced by the large gap between training and validation performance.*
- *While the training accuracy improves steadily with each epoch, the validation accuracy remains relatively stable, indicating that the model fails to generalize well to unseen data.*
- *The validation loss increases sharply after a certain number of epochs, suggesting that the model starts to overfit the training data.*

4.3 - Hyperparameter Tuning

We tuned the model by experimenting with different values of hyperparameters such as learning rate, batch size, and number of epochs.

Model Architecture	**Number of Trainable Parameters**	**Mean Training Accuracy**	**Mean Validation Accuracy**
Sigmoid	144,769	0.7375	0.6258
Relu	72,449	0.9785	0.6367

5 - Final Model

The optimal model had the following structure:

- *Activation Function Relu and Sigmoid.*
- *tfidf_maxfeatures* as a input Layer.
- *Dense Layer with 64 units and ReLU activation function.*
- *Hidden neurons as [128 ,256]*

The model was trained with a learning rate of 0.001 and achieved a testing accuracy of approximately **63.99**%.

6 - Way Forward

The final model showed improvement but still had limitations in accuracy on the testing data. Future work could involve:

- Further hyperparameter tuning
- Exploring deeper and more complex architectures

The assignment provided valuable insights into the process of building and optimizing sentiment analysis on the dataset and done preprocessed using NLP techniques and represented using N-gram models. And overall conventional model performed better than deep learning model in this case.