Title: Sentiment Classification of IMDB Movie Reviews Using Traditional and Deep Learning Models

1. Definition of Problems

Natural Language Processing includes sentiment analysis which focuses on discovering emotional tones present in textual inputs. The project separates movie reviews into positive and negative opinions to determine their sentiment. Identifying elements in text is essential for computers to understand human language. The analysis of movie reviews becomes difficult due to their use of figurative expressions and contextual meanings. This addresses why sentiment analysis systems must progress past basic positive-negative categorization to incorporate a deeper understanding of grammatical structures.

2. Scope of the Project

The focus of this project is on sentiment analysis in the context of movie reviews. The aim is to identify whether a certain movie review expresses positive or negative sentiment. Movie reviews are typically complex expressions and figurative phrases. This makes them need more advanced text processing methods, as they are especially difficult to classify automatically. This challenge facilitates the assessment and comparison of models' effectiveness, particularly to capture verbal variations. Additionally, it shows real user feedback that is greatly valued in media analytics and recommendation systems, it offers a useful context. This project makes an impactful contribution to the NLP domain by creating more context-aware models and analysing subjective, human-generated information in long-form reviews.

3. Importance

Understanding sentiment in text data can benefit many industries. Saif M. Mohammad's Sentiment analysis (Mohammad, 2021) proposes its uses in commerce, public health, government policy, social science, and art. It helps to measure audience happiness, promotes marketing strategies, and influences production decisions. (Bollen, Mao and Zeng, 2011) research demonstrated that collective mood states obtained from Twitter feeds could predict the Dow Jones Industrial Average (DJIA) with an accuracy of 87.6%. Millions of reviews are available on IMDB, yet it is not possible to manually analyse them. Automatic sentiment classification enables timely insights. Sentiment analysis is a stepping stone towards developing more complex applications like chatbots, social monitoring systems, etc., in a world where decisions are made mostly based on data. Sentiment analysis is still challenging despite its wide adoption, particularly in lengthy texts like movie reviews, where idiomatic language, sentence structure varies widely, highlighting the need for improved understanding and broader models' applicability.

4. Background Review of Related Work

Various studies have laid the foundation for sentiment classification. Semi-supervised sentiment analysis was proposed by (Maas *et al.*, 2011), using IMDB dataset of 50,000 reviews. Traditional baseline models were employed using vector space model features with Naïve Bayes and Support Vector Machines, these methods ignored the sequence in which words appear in a sentence. The authors learned word vectors using a probabilistic model. Meaning and sentiment captured by these vectors were combined into global features using logistic regression for classification.
Recursive Neural Tensor Networks (RNTNs), created by (Socher *et al.*, 2013). This approach broke sentences into smaller phrases using syntactic parse trees, computing sentiment for each phrase. By integrating word vectors through grammatical rules, the model detected and interpreted advanced contextual sentiment. The method showed excellent accuracy during the first use, but its recursive architecture and tensor-based computations led to high operational costs that became more severe with extended sentences and larger datasets.

(Zhang, Zhao and LeCun, 2015) Introduced a character-level Convolutional Neural Network (CNN) for text classification. This model converts raw text without the need for tokenization or word embeddings. By learning structured figures directly from sequences of characters, the model captures morphological and sub-word features, making it rich in all languages and domains. The project delivered competitive results across multiple large-scale text classification tasks. This approach necessitated large volumes of labelled data along with intensive training sessions and major computational resources because it involved deep learning exclusively from character data.
(Devlin *et al.*, 2019) introduced BERT (Bidirectional Encoder Representations from Transformers), a transformer framework model that preliminarily trains deep bidirectional encoding by mutual inference on both left and right contexts. (e.g., The ball is over the fence, left context would be “The ball is”, and the right context would be “the fence”). Unlike traditional unidirectional models, BERT captures contextual dependencies more effectively, facilitating the highest performance across diverse NLP tasks. Through refinement, BERT achieved high performance on GLUE, SQuAD, and MNLI benchmarks. Whilst it is not specifically designed for sentiment analysis, BERT has since been successfully applied to sentiment tasks, consistently outperforming traditional RNN and CNN-based models.
The Universal Language Model Fine-tuning method, known as ULMFiT was developed by (Howard and Ruder, 2018). This approach to text classification operates by implementing transfer learning techniques. Fine-tuning a pre-trained model enables it to perform new tasks while requiring less task-specific data. The approach outperforms the state-of-the-art on various text labelling tasks across multiple benchmark datasets by reducing misclassification rates more than previously attempted methods.

5. Smart Objectives

The objective is to develop a sentiment classifier to categorise IMDB movie reviews as either positive or negative. The dataset selected is the IMDB Large Movie Review Dataset. The project will involve preprocessing the text data using NLP techniques and training two traditional (Logistic Regression and Naïve Bayes) and two deep learning models (MLP and CNN) using Python-based libraries such as Scikit-learn and Keras. The objective of this project is to achieve performance results that match or surpass the existing

benchmarks set for this dataset. For example, Maas et al. (2011) reached 88% accuracy through traditional machine learning methods. Therefore, the model must reach at least 88% accuracy and F1-score as the performance benchmark.
Studies like those by (Pang, Lee and Vaithyanathan, 2002; Devlin *et al.*, 2019) have affirmed the success of these approaches on similar NLP datasets, confirming that this goal is realistic with standard computational resources. Achieving high efficiency in sentiment classification of movie reviews has shown real-time utility in content recommendation, automated moderation, and user feedback analysis. This project contributes meaningful advancements by implementing and refining multiple models on a complex real-world dataset.
This project was concluded over a 4-week duration. Week 1 involved data exploration and cleaning, Week 2 focused on feature extraction and traditional model training, Week 3 focused on deep learning model implementation and refinement, and Week 4 on evaluation, visualisation, and documentation.

6. Dataset Description

This project employs the IMDB Movie Dataset, which (Maas *et al.*, 2011) presented. It is a publicly available standard dataset typical for sentiment classification tasks in NLP. The collection consists of 50,000 labelled movie reviews, which are split equally between positive and negative sentiment categories. These reviews are stored in a tabular CSV format with two main columns: Each review features textual content and sentiment labels, either positive or negative. The reviews exhibit diverse lengths, as well as varied vocabulary and structural elements, which enables effective training for both traditional machine learning and deep learning approaches. All entries are in English and largely consist of user-driven reviews.
The dataset structure is textual and semi-structured. Each row contains a review in natural language text and its equal sentiment label. The balance between positive and negative labels makes it very suitable for a binary classification task, eliminating the need for artificial class rebalancing methods during preprocessing. The dataset's suitability lies in its wide usage, balanced class distribution, and realistic textual variability.

7. Exploratory Data Analysis (EDA) and Preprocessing

7.1. Initial Dataset Exploration

Important libraries like pandas, NumPy, etc., were imported. The dataset consisted of 50,000 entries with the two main columns: review (textual movie review) and sentiment (labelled either ‘positive or ‘negative’). A simple class count revealed a perfectly even dataset, with 25,000 positive and 25,000 negative reviews. This favours binary classification as it avoids the bias commonly introduced by class imbalance.

7.2. Text Cleaning and Preprocessing

Initial preprocessing steps included: Lowercasing and removing special characters.
Lowercasing involves changing all text to ensure uniformity and reduce sparseness. Special characters like punctuation and HTML tags are unnecessary for sentiment classification. Regular expressions (re) were used as tools to clean text. These steps reduced noise and ensured that tokens retained for modelling were meaningful.

Tokenisation

The Gensim library was used for tokenisation. Each review was split into a list of tokens (unigrams), which formed the foundation of preprocessing pipeline. Bigrams and trigrams were generated using Gensim’s Phrases model. This step obtained multi-word expressions such as “darker_side”, “looking_forward”, and “devil_wears_prada”, which hold more sentiment information together than separately.

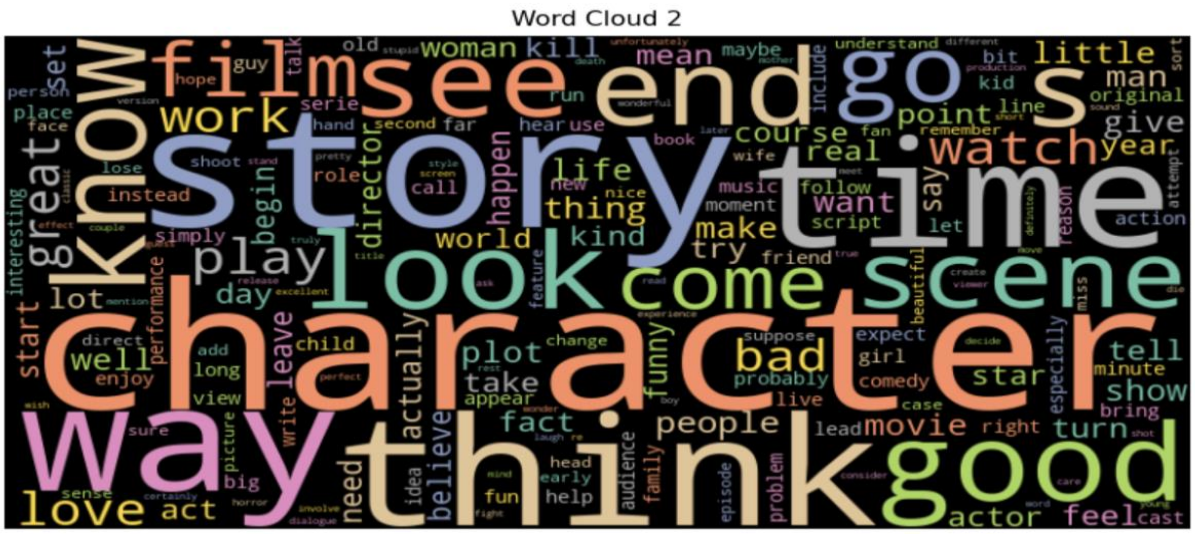
Stopword Removal

Stopwords are typically existing words like “the, is, and at”, which were removed using Genism’s built-in stopword list, enlarged to include domain-specific words such as “br”, “film”, and “movie”.

Lemmatisation

The SpaCy library performed the text cleaning process of lemmatisation. The process transforms every token into its base form while maintaining only nouns, verbs, adjectives, and adverbs. The process reduced word redundancy like changing calling to call and giving to give, while preserving essential sentiment-related tokens.

Frequency Analysis
Understanding the most dominant word in the dataset after completing the cleaning process, the word frequency distribution was computed. It showed domain-specific words, e.g., “story”, “character”, “think”, “way”, etc, among the most frequent. This confirms the effectiveness of the stopword removal.



Frequency and word-cloud visualisation, showing the dominant specific word.

As a baseline, Maas et al (2011)., research achieved 88% accuracy. This served as an adequate baseline to evaluate whether the models captured meaningful patterns. The machine learning models used (e.g., Logistic Regression had 88% accuracy) were approximately equal to the baseline. Demonstrating that they successfully learned sentiment-related features rather than random guessing.

7.3. Traditional machine learning method

- **Naïve Bayes:** It is a probabilistic predictive model that assumes feature independence, which makes it effective for text classification using term-frequency or TF-IDF representations.

Strengths:
It is computationally effective and fast.

Performs well with complex sparse data, like text.
Easy to implement and interpret.

Weakness:
It assumes feature independence, which may not hold in actual language.
Less suitable when word dependencies are necessary for classification.

- **Logistic Regression:** It is a linear classifier used for binary classification. It evaluates the probability that input data fits into a particular class, making it widely used for sentiment analysis.

Weakness:
Limited to linear decision boundaries.
May underperform when feature interactions or non-linear patterns are present.

Strengths:
Interpretable and simple to implement.
Perform well with TF-IDF features
Can be regularised to prevent overfitting.

- **Support Vector Machine (SVM)**

SVMs are strong classifiers that seek to find the optimal hyperplane differentiating classes in high-dimensional space.

Strength:
Effective in high-dimensional feature spaces.
Resilient to overfitting with proper regularisation.

Weakness:
Computationally intensive for large datasets
Difficulty in scaling with increasing sample size.

- **Decision Tree**

A non-parametric supervised method used for categorisation. It builds a model by learning simple decision rules collected from the data features.

Strength:
Easy to visualise and interpret
Handles both numerical and categorical data.

Weakness:
Prone to overfitting on training data.
It is less stable due to high variance.

- **Random Forest**

An ensemble learning approach that builds multiple decision trees and aggregates their results to boost classification performance.

Strength:
Reduces overfitting.
It achieves high accuracy on many problems.
It works well with large datasets.

Weakness:
The model makes predictions at a slower pace while using more memory compared to simpler models.
The ensemble nature of the model makes it less interpretable.

Chosen Models and Justification
The two models selected for implementation in this project are.

1. Logistic Regression: It was chosen for its solid performance on hyperplanes, simplicity, interpretability, effectiveness for binary classification, and it works well with TF-IDF features and provides a strong baseline.
2. Multinomial Naïve Bayes: It was chosen for its strength and compatibility with text data represented as word frequencies or TF-IDF vectors, is fast, efficient, and performs well with high-dimensional data.

8. Deep Learning Methods

- **Feedforward Neural Network (Multilayer Perception (MLP))**

An MLP is a simple deep learning model composed of fully connected layers. It can be trained effectively on numerical representations of text, like TF-IDF, but it does not inherently model sequence information.

Strengths:
Straightforward to implement.
Can approximate any function given enough hidden units
Performs well on sparse feature vectors like TF-IDF.

Weakness:
Ignores word order and sequence context.
Can overdraft without proper regularisation.

- **Convolutional Neural Network (CNN)**

Originally designed for visual recognition, CNNs have shown to be effective at classifying text sequences by capturing local patterns through sliding filters.

Strength:
Efficient at detecting local patterns like n-grams
More parameters efficient.
Faster to train and parallelise

Weakness:
Limited in capturing long-range dependencies in text.
Less intuitive when interacting with temporal relationships.

- **Recurrent Neural Network (RNN)**

RNNS are built to handle sequential data, making them inherently suited for language tasks. They have a form of memory that store context to inform current predictions.

Strengths:
Good at modelling sequence and time-dependent data

Naturally captures the order of words.

Weakness:
Prone to vanishing gradients
Training can be slow for a long sequence.

- **Long Short-Term Memory (LSTM)**

LSTMs, an extension of RNNs built to handle long-term components. They include memory gates that control the flow of information.

Strengths:
Solves vanishing gradient issues seen in RNNs
Effective at capturing long-context relationships.

Weakness:
Computationally expensive
Requires significant tuning and training time.

- **BERT (Bidirectional Encoder Representations from Transformers)**

BERT is a transformer-based model that preliminarily trains deep bidirectional encoding by mutual inference on both left and right contexts.

Strengths:
Excellent results in many NLP tasks.
Captures complex dependencies and contextual relationships.
Highly transferable via fine-tuning

Weakness:
Requires significant computational resources.
Model size can be prohibitive for small-scale training.

Chosen Deep Learning Models and Justification
The following two were selected for implementation.

1. **Feedforward Neural Network (MLP)**

Justification: Chosen due to its simplicity and compatibility with TF-IDF features already generated for the traditional models. It allowed the reuse of existing preprocessing pipelines and provided a useful benchmark for comparing more advanced models.

2. **Convolutional Neural Network (CNN)**

Justification: It captures local feature patterns in padded sequences derived from tokenised text. CNNs are less resource-intensive than LSTMs or BERT while still providing competitive performance in the sentiment analysis task.

9. Implementation and Refinement

The practical implementation of the sentiment classification pipelines using both the traditional and deep learning methods. All strategies and refinements used are discussed.

Libraries and Tools Used

- Pandas and NumPy for data handling and numerical operations
- Genism for tokenisation, bigram/ trigrams construction
- SpaCy for lemmatisation with part-of-speech filtering
- Matplotlib and WordCloud for visualising token frequency and other visualisations.
- Scikit-learn for traditional models,TF-IDF vectorisation, performance metrics, and train-test splitting.
- TensorFlow/Keras for building, training, and evaluating deep learning models.

Traditional Model 1: Logistic Regression

Implementation:
Input features were generated using TF-IDF from pre-processed lemmatised text. The model was trained using LogisticRegression() from sklearn.linear_model.

Refinement:
Hyperparameter Tuning:

Tested for the regularisation strength (C), using the original and adjusted values.

Traditional Model 2: Multinomial Naïve Bayes

Implementation
TF-IDF feature vectors were used as inputs
The model was trained using MultinomialNB() from sklearn.naive_bayes.

Refinement:

Hyperparameter Tuning: The smoothing parameter alpha was tested with values like 0.5 and 1.0
Pipeline Simplification: Grid search was used to compare alpha values and select the best-performing version.

Deep Learning Model 1: Feedforward Neural Network (MLP)

Implementation:
The MLP was built using Keras' Sequential () API with Dense layers.
TF-IDF feature vectors were used as input.
Binary cross-entropy was applied as a loss function and Adam optimizer.

Refinement:
Hyperparameter Tuning: Tuned the number of hidden layers (1 to 5), hidden units decreasing from 256 to 16, each followed by a dropout layer to reduce overfitting.

Deep Learning Model 2: Convolutional Neural Network (CNN)

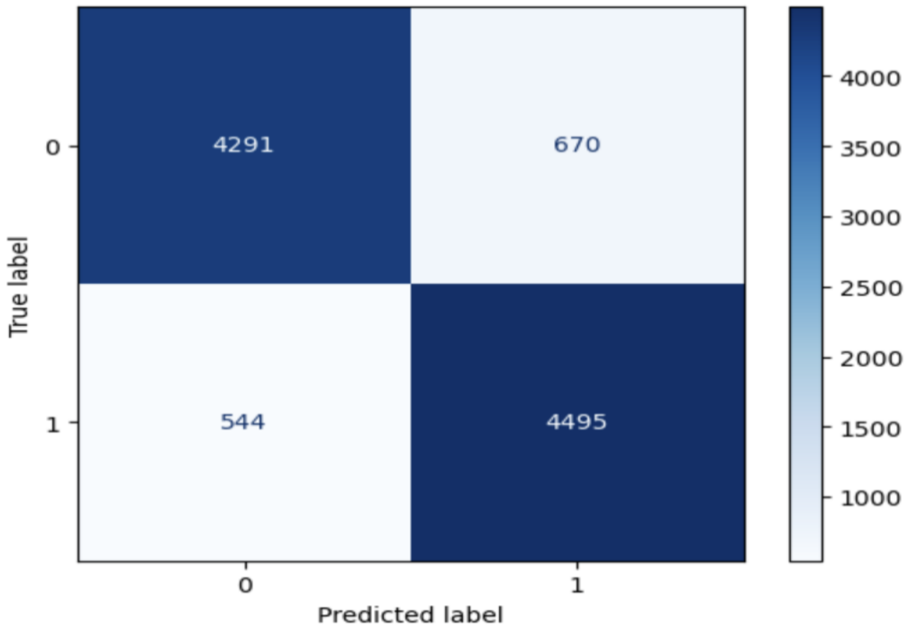
Implementation
Text was tokenised and padded using keras' Tokenizer and pad_sequences.
Input passed through an embedding layer, next by a convolutional layer, then a pooling layer
Final classification passed through a dense output layer with binary activation.

Refinement:
Hyperparameter Tuning: Adjusted the kernel size (5 to 3), filter count (from 128 to 64), and Dense layer (64 to 32)
Dropout and L2 Regularisation:
Dropout layers (0.5 to 0.2) and L2 kernel regularisation were added to reduce overfitting.

10. Evaluation and Metrics

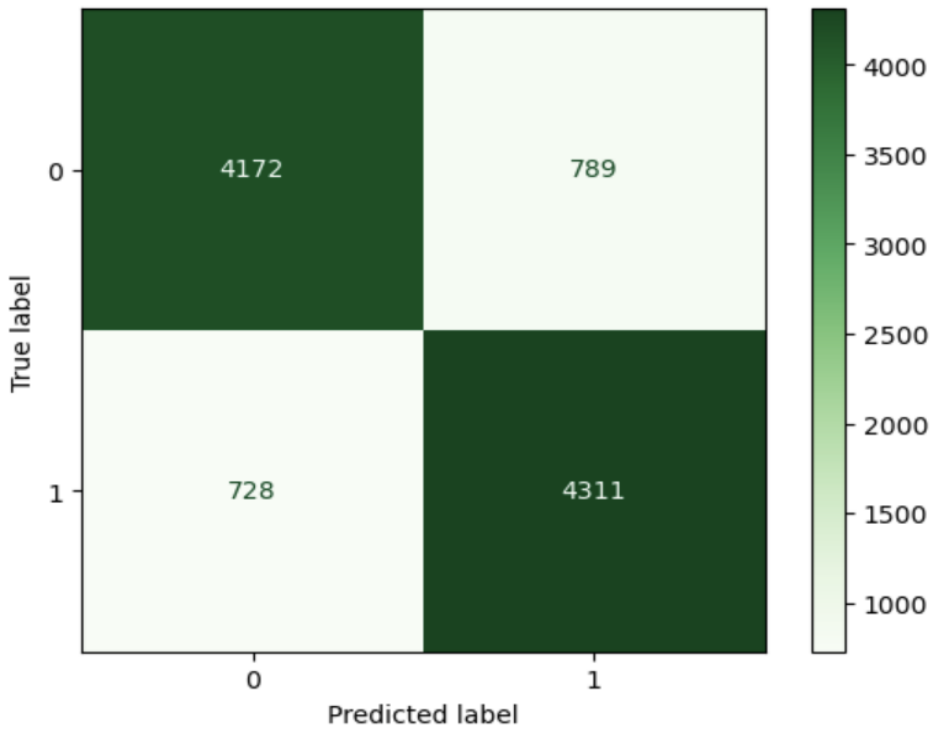
Each model was evaluated on multiple metrics.

- **Simple Regression Classifier Evaluation:** After training was completed, the classification report gave an accuracy score of 0.88 and an F1-score of 0.88. The confusion matrix displayed how well the model has learnt to differentiate between positive and negative reviews by showing the counts of true positives/ true negatives, false positives/false negatives.



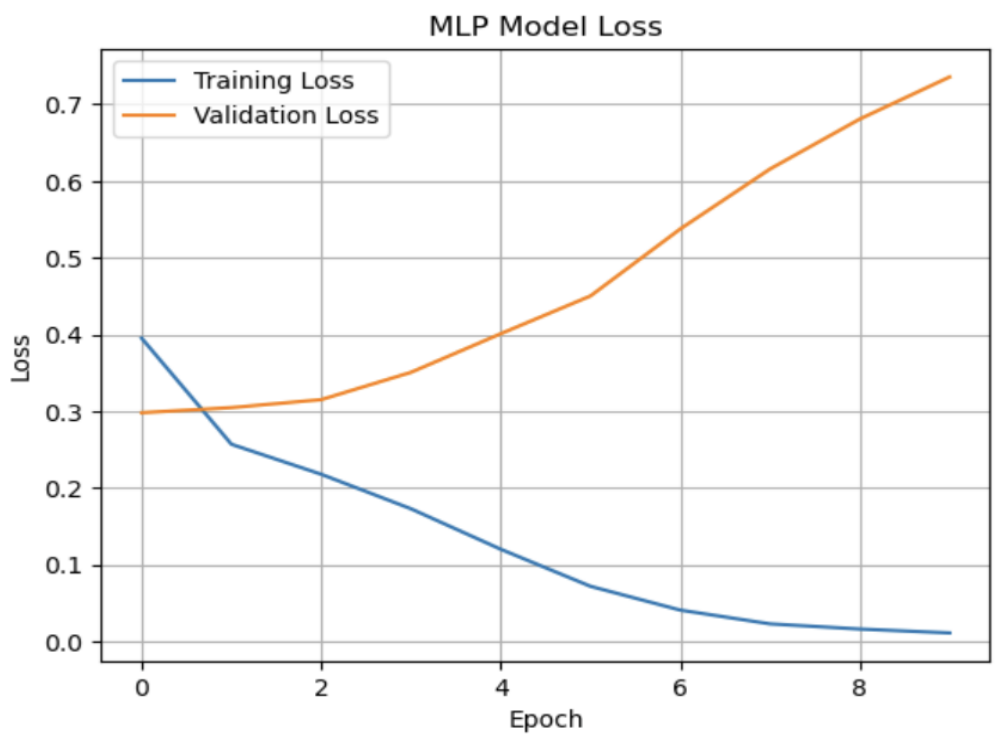
0 represents negative, and 1 represents positive.

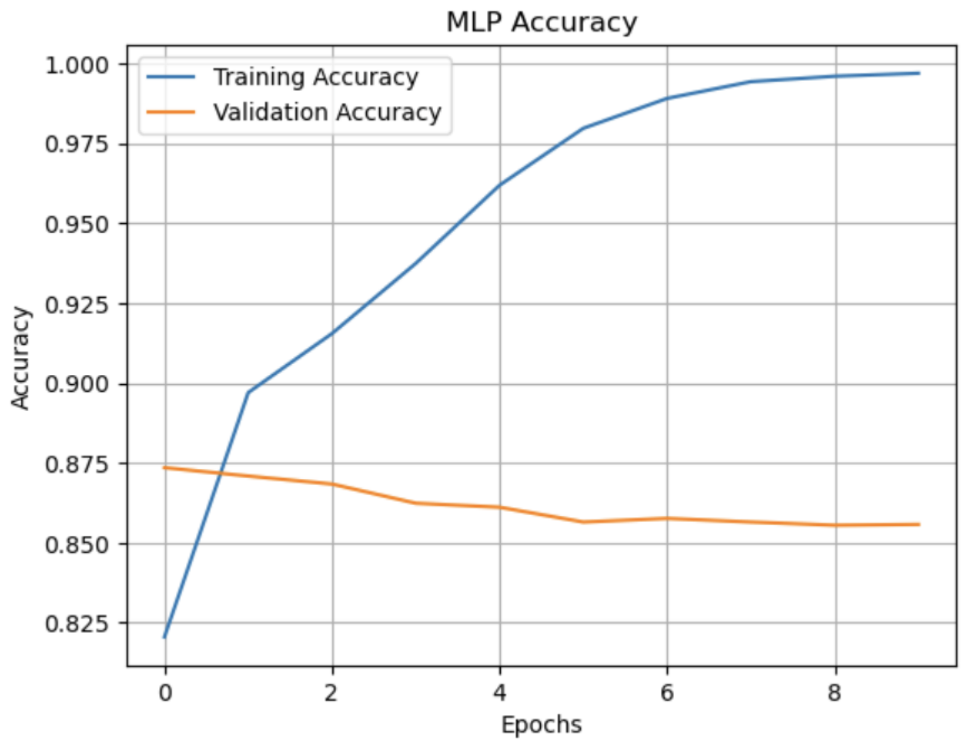
- **Multinomial Naïve Bayes Evaluation:** Naïve Bayes had an accuracy score of 0.85 and an F1-score of 0.85 after complete training. The confusion matrix shows how the model finds distinctions between the positive and negative reviews.



The negative (0) had true negative of 4172 and false positive 789. Comparison between these two models shows that Simple Regression performed better with a 0.88 accuracy, compared to Naïve Bayes.

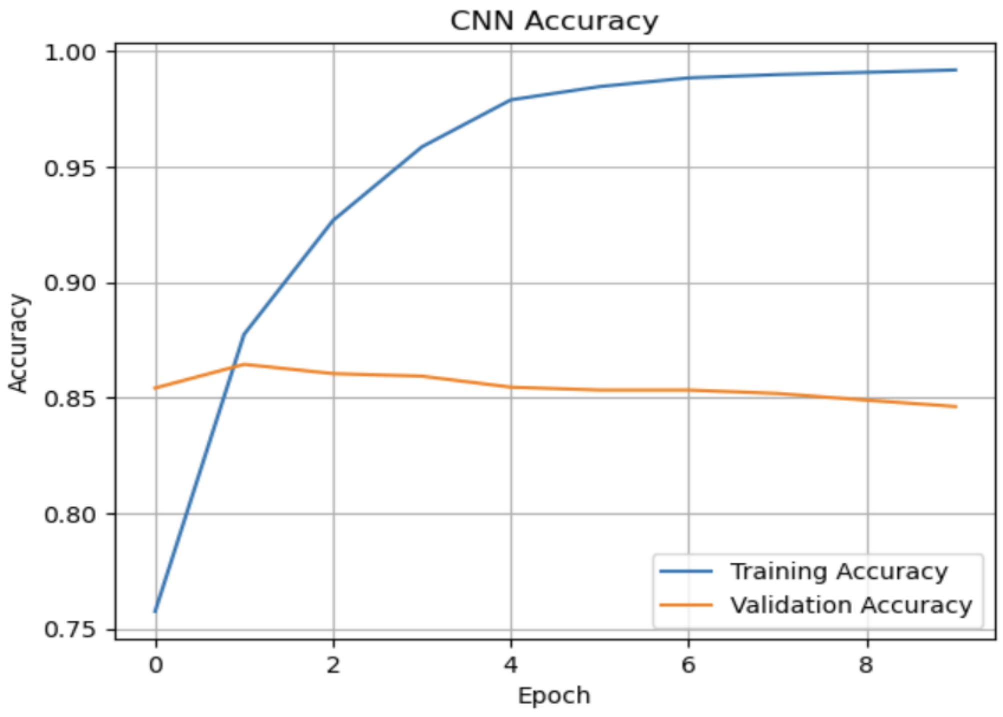
Evaluation for deep learning models.
Feedforward Neural Network (MLP) using keras: After successful training, the model achieved an accuracy of 0.86 and an F1-score of 0.86 as well. The model was trained for 10 epochs, and the training/validation loss plot and training accuracy/validation accuracy plot were visualised.

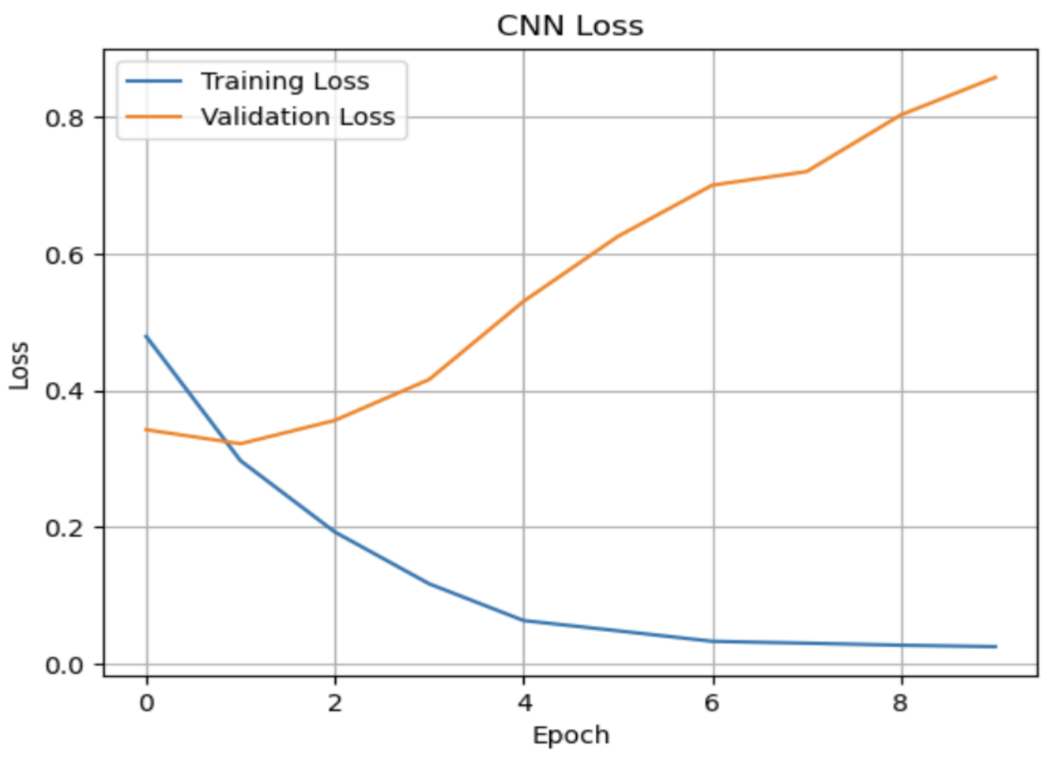




The MLP training and validation show the model is learning very, there is no sign of overfitting from the two plots visualised. The confusion matrix records that the negative review had 4299 true negatives and 662 false positives. While the positive had a false negative of 724 and a true positive of 4315.

Evaluation of Convolutional Neural Network (CNN): The model was trained for 10 epochs using the Adam optimizer. It had an accuracy of 0.85, which shows the model performed well. The training accuracy and validation loss plots show that the model trains well without any sign of overfitting.



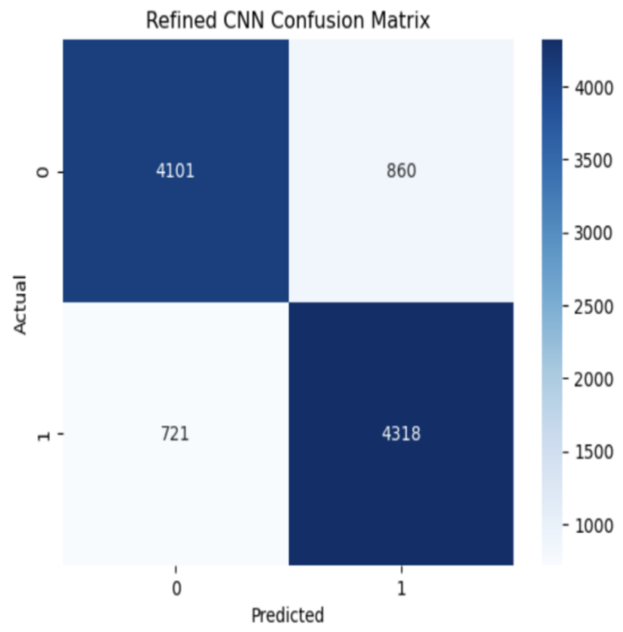


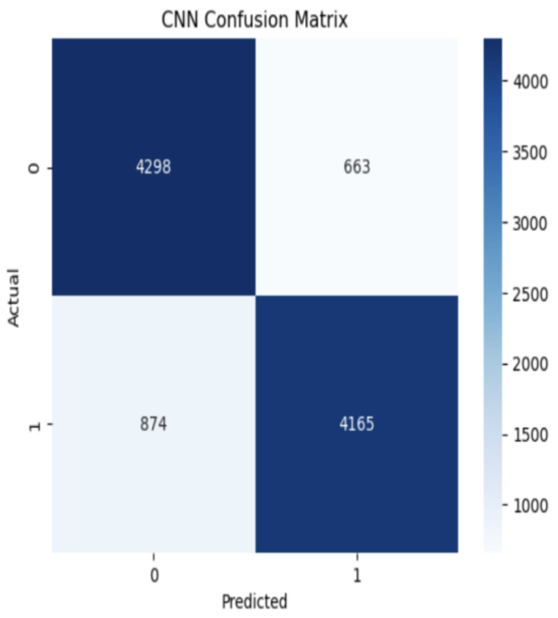
Evaluation of model refinement

The CNN model was refined through hyperparameter tuning and L2 regularisation. Dropout layer was added, filters were reduced to 64, and the kernel size was reduced to 3 compared to the older one of 128 filters, kernel size of 5. Training accuracy and loss plot showed the model trained well with no sign of overfitting. The older CNN model performed better in terms of overall accuracy compared to the refined model.

Classification report for CNN refined and unrefined (older CNN).

Metric	Older-CNN	Refined-CNN
F1-Score 0 (Negative)	0.85	0.84
F1-Score 1 (Positive)	0.84	0.85
Macro Average	0.85	0.84
Weighted Average	0.85	0.84
Accuracy	0.85	0.84





The refined had a lower recognition compared to the unrefined. I suggest the reason this happened is that there's no problem to solve; the unrefined CNN model was stable in training and wasn't underperforming. Adding dropout/L2 just made the model more cautious, slightly affecting its performance.
The refined MLP slightly improved the true positives from 4315 to 4336, adding more dense layers and dropout layers. But true negatives underperformed compared to the non-refined MLP. Although they both still had the same accuracy score of 0.86.

Traditional model Refinement.

Multinomial Naïve Bayes was refined by tuning the alpha parameter to find the best-performing version. The best alpha was 2.0. The accuracy score was 0.85, just like the older Naïve Bayes accuracy score. The result recorded from the confusion matrix shows that there isn't much difference between the two.
The Simple Regression model was tuned by testing for the regularisation strength (C), using the original and adjusted values. The best strength (C) was 1 (one). Accuracy was 0.88, just like the non-refined.
Tuning did not change results according to the report from this analysis, which may be because the hyperparameter space was already optimal, or data preprocessing is already good.

11. Conclusion

This project achieved its aim by successfully building and evaluating traditional and deep learning sentiment classifiers, achieving a minimum 88% accuracy and F1-score.

Reference

Bollen, J., Mao, H. and Zeng, X. (2011) 'Twitter mood predicts the stock market', *Journal of Computational Science,* 2(1), pp. 1-8.

CWAUTHORS, 2023. Choosing the Right Citation Style. [online] Available at: https://www.cwauthors.com/article/Choosing-the-Right-Citation-Style

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, June: Association for Computational Linguistics, 4171-4186.

EliteDataScience, n.d. Modern Machine Learning Algorithms: Strengths and Weaknesses. Available at: https://elitedatascience.com/machine-learning-algorithms

Howard, J. and Ruder, S. (2018) 'Universal language model fine-tuning for text classification', *arXiv preprint arXiv:1801.06146*.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. and Brown, D., 2019. Text classification algorithms: A survey. *Information*, *10*(4), p.150.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. and Potts, C. 'Learning word vectors for sentiment analysis'. *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142-150.

Mohammad, S. M. (2021) 'Sentiment analysis: Automatically detecting valence, emotions, and other affectual states from text', *Emotion measurement*: Elsevier, pp. 323-379.

Pang, B., Lee, L. and Vaithyanathan, S. (2002) 'Thumbs up? Sentiment classification using machine learning techniques', *arXiv preprint cs/0205070*.

Shiri, F.M., Perumal, T., Mustapha, N. and Mohamed, R., 2023. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. *arXiv preprint arXiv:2305.17473*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y. and Potts, C. 'Recursive deep models for semantic compositionality over a sentiment treebank'. *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631-1642.

Wordvice, 2023. Differences in citation styles: APA, MLA, Vancouver, Chicago. [online] Available at: https://blog.wordvice.com/differences-in-citation-styles-apa-mla-vancouver-chicago/

Zhang, X., Zhao, J. and LeCun, Y. (2015) 'Character-level convolutional networks for text classification', *Advances in neural information processing systems,* 28.