Title: Text Classification on imdb movies using Supervised and Deep Learning

Note: The dataset used for the video presentation is different from the dataset used for this study. This is because I found out that the earlier dataset was not fit for this study. I engaged you in the lab on this, and you advised that I could go ahead with the new dataset. However, the methodology and background remain the same while the objective changed a little. The first proposed topic used in the video presentation was Analyzing IMDB Horror Movies: Impact of Ratings and Box Office Performance with Deep Learning Sentiment Analysis while the main topic for this work is Text Classification on imdb movies using Supervised and Deep Learning.

Introduction:

For filmmakers and producers, a thorough understanding of audience preferences is of utmost importance in the dynamic world of the entertainment industry. In this setting, IMDB stands out as a leading website that appeals to movie buffs and provides a wealth of information in the form of movie reviews. It is possible to realise a profound understanding of audience preferences by unlocking the latent potential contained in these reviews and achieving it via the lens of text classification. The primary motivation behind this research project is to build predictive algorithms that are adept at understanding the nuanced details of moviegoers' preferences.

The goal is to create models that exhibit extraordinary accuracy in predicting the sentiments entwined within the fabric of IMDB movie reviews through a careful fusion of both supervised and deep learning techniques.

Aims and Objectives:

The primary objectives of this work are:

- To build a predictive model that will be used to assess overall sentiment polarity (positive and negative) towards imdb movies.
- To thoroughly examine several text categorization methods, such as BOW, CBOW, TF-IDF, Word2Vec, RNN, and LSTM. It rates their effectiveness using measures such as confusion matrices and evaluation metrics to determine the best technique for interpreting viewer attitudes in movie reviews.

Background:

Online platforms and user-generated material have had a significant impact on the landscape of the entertainment industry, notably the film sector, in the digital age. A well-known online resource called IMDB (Internet Movie Database) offers a sizable collection of movie ratings and reviews, providing important insights into viewer sentiment and cinematic preferences (IMDB, nd.).

Text categorization has become a key method for sentiment analysis and text categorization in the field of Natural Language Processing (NLP). Text classification utilizes machine learning algorithms and pre-trained models to automatically assign raw text data to predefined categories, enabling the prediction of categories for new, unlabeled text (ProjectPro, 2023).

On websites like IMDB, text classification has been widely used to classify and rate the emotion of movie reviews. Banerjee et al, (2022) performed comparative analysis using different algorithms on imdb dataset and discovered that Logistic regression algorithm performed better than other algorithms like Random Forest, Decision Tree, KNN and Support Vector Machine. They however stated their findings based on the pros and cons of these models. For an example, they discovered that KNN is very easy to implement but experiences performance degradation if the dataset size is increased (Banerjee et al, 2022).

Deep learning models have gained popularity recently, and Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are particularly good at comprehending complex patterns in text. LSTM has shown the ability to analyse sequences and capture lengthy dependencies, which enables it to perform well in text classification using python ((Shekhar, 2021).

In previous studies, preprocessing, feature engineering, and model selection have been emphasized in sentiment analysis (Raj, 2021). A thorough comparison of conventional and deep learning techniques is missing, nevertheless. By using measurements like accuracy, precision, recall, F1-score, and confusion matrices to evaluate the supervised learning techniques, and loss, accuracy, validation loss and validation accuracy for deep learning techniques, this study closes the knowledge gap.

Methodology:

This project will employ a mixed methods approach integrating both computational text mining algorithms and qualitative human analysis to enable a full and robust investigation of user-generated content about self-driving automobiles on the twitter platform. Python libraries like Sklearn, seaborn, matplotlib, wordcloud, Kerras and others are used to carry out this work. These are the actions I took:

- **Text cleaning Techniques**: cleaning techniques like tokenization, lowercasing, lemmatization, and special character normalization increase the quality of the input data. By reducing text data noise, this frequently enhances the performance of NLP models (Kim, 2022).

- **Word Embeddings for Feature Engineering:** Word embeddings like Word2Vec, store the semantic connections between words, providing the model a more thorough understanding of word context and meaning. It is employed to turn words into distributed representations of numerical vectors (MathWorks, n.d).

- **Supervised learning with Gradient Boosting and Random Forest Models**: When compared to any individual member of the ensemble, the ensemble model improves overall predictive performance by reducing variation in a predictive model's average competence (Brownlee, 2020). By applying ensemble techniques, the models can handle non-linear patterns and complex correlations in the data, which aids in accurate sentiment prediction.

- **Hyperparameter Tuning:** By fine-tuning the models' hyperparameters, the performance of the models is maximized. This stage improves generalization and accuracy by ensuring that the models are operating to their full capacity**.**

- **Deep Learning with Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM):** LSTM model with embedding and output layer will be used in this study. Due to LSTM's ability to capture lengthy dependencies and analyze sequences, LSTM models performs well in text classification (Shekhar, 2021). RNN which are usually applied when dealing with sequential data will be used in this study.

- **Multiple Model Building and Training:** By creating a variety of models with multiple word embeddings and hyperparameter tunings, the problem may be fully investigated. The likelihood of identifying the best-performing model increases since this strategy considers potential variations in data distribution and properties.

This approach shows a comprehensive methodology that combines a variety of tools and models, making it a strong and effective strategy for sentiment analysis in the context of imdb movies. This distinguishes my methodology from other methodologies now in use.

Experiment and Discussion:

To build this predictive model, an imdb dataset[1] from Kaggle was used. Series of steps used are outlined and explained below. The dataset is made up of 50000 rows and two columns.

Data Cleaning: The data was explored using different data exploration techniques to check inconsistencies, null values, and duplicated values. Duplicate values were removed. Table 1.0 shows the samples of text data in the dataset used.

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns

Table 1.0 – samples of text data

[1] https://drive.google.com/file/d/1-5a6SDbL4dxWmUt7UPWHgYHd5CPgPhRI/view?usp=sharing

Exploratory Data Analysis: WordCloud was used to explore both positive and negative words in my dataset which are **review** and **sentiment.** This is to enable us to understand the most occurring words before performing our data preprocessing.

For the first column which is the review, words are as shown in figure 1.0 below were discovered.

Figure 1.0 - words in the review column

In figure 1.0, boring, least, kill, nothing, enough, bad, terrible, little, stupid, instead, used, wrong, old are some of the important recurring that appears as negative reviews while the positive reviews are shown to be words like good, better, beautiful, enjoy, give, love, funny, laugh, life, believe, family, real, great, feel, much, and best.

Data balance for the sentiment column was also checked to see if the classes are balanced. This showed an insignificant difference of 0.38% which I believe does not require further data balancing using algorithm like SMOTE or oversampling. Figure 2.0 below showed the 0 class has 49.81% while the 1 class has 50.19% of the data.

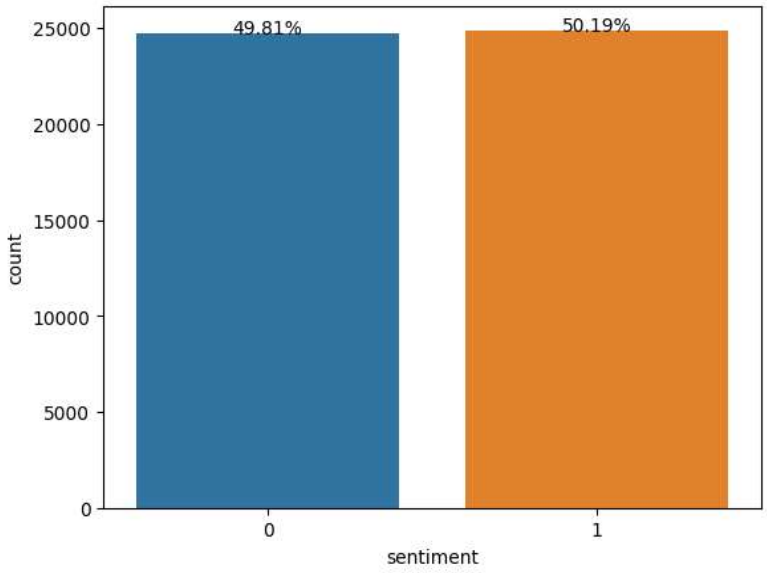

Figure 2.0 percentage of data make up of the sentiment classes.

Data preprocessing: This step involves the followings:

- **HTML Tag removal:** This entails removing the html code from the text data. This was implemented and a sample result is as shown in the table 2.0 below.

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming tec...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

Table 2.0 output of html tag removal

- **Fix Contraction:** This involved expanding shortened words for more accurate analysis during data cleaning and sample output is as shown in the table 3.0 below.

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming tec...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there is a family where a little boy...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

Table 3.0 – output of Fix Contraction

- **Tokenization:** This involves dividing words into smaller units called tokens which are usually sub-words or words. The output is as shown in table 4.0.

	review	sentiment
0	[One, of, the, other, reviewers, has, mentione...	1
1	[A, wonderful, little, production, ., The, fil...	1

Table 4.0 – Output of Tokenization.

- **Removal of Digits:** This involves numerical digits removal. This was done to my text data and the output is as shown in the table 5.0 below.

	review	sentiment
0	[One, of, the, other, reviewers, has, mentione...	1
1	[A, wonderful, little, production, ., The, fil...	1
2	[I, thought, this, was, a, wonderful, way, to,...	1
3	[Basically, there, is, a, family, where, a, li...	0
4	[Petter, Mattei, 's, ``, Love, in, the, Time, ...	1

Table 5.0 – Output of Digits removal.

- **Removal of Non-ASCII character:** This involved discarding of characters that are outside the ASCII range of encoding from text data. Table 6.0 is the output.

	review	sentiment
0	[One, of, the, other, reviewers, has, mentione...	1
1	[A, wonderful, little, production, ., The, fil...	1
2	[I, thought, this, was, a, wonderful, way, to,...	1
3	[Basically, there, is, a, family, where, a, li...	0
4	[Petter, Mattei, 's, ``, Love, in, the, Time, ...	1

Table 6.0 – Output of Non-ASCII character removal

- **Remove Punctuation:** This involved removal of punctuation from my text data. However, after this action, I notice that Some characters like `` were still present in text data. I will have to use regular expression to eliminate them before building my model. The output is as shown in the table 7.0 below.

	review	sentiment
0	[One, of, the, other, reviewers, has, mentione...	1
1	[A, wonderful, little, production, The, filmin...	1
2	[I, thought, this, was, a, wonderful, way, to,...	1
3	[Basically, there, is, a, family, where, a, li...	0
4	[Petter, Mattei, 's, ``, Love, in, the, Time, ...	1

Table 7.0 – Output of punctuation removal from my text data

- **Lower casing:** This involved converting all the words in my dataset to lower case. The output is as shown in table 8.0 below. You can se that all letters are in small letters.

	review	sentiment
0	[one, of, the, other, reviewers, has, mentione...	1
1	[a, wonderful, little, production, the, filmin...	1
2	[i, thought, this, was, a, wonderful, way, to,...	1
3	[basically, there, is, a, family, where, a, li...	0
4	[petter, mattei, 's, ``, love, in, the, time, ...	1

Table 8.0 – Output of the lowercasing

- **Stop-words removal:** This involves removing of insignificant and frequent occurring words like ‘and’, ‘is’ and ‘the’ which conveys no meaning on its own. The output is as seen in the table 9.0 below.

	review	sentiment
0	[one, reviewers, mentioned, watching, oz, epis...	1
1	[wonderful, little, production, filming, techn...	1
2	[thought, wonderful, way, spend, time, hot, su...	1
3	[basically, family, little, boy, jake, thinks,...	0
4	[petter, mattei, 's, ``, love, time, money, "...	1

Table 9.0 – Output of the stop words removal

- **Lemmatization:** This involves reducing words to their root or base form (e.g., "running" becomes "run") to normalize variants. Table 10.0 is the output of this action.

	review	sentiment
0	[one, reviewer, mentioned, watching, oz, episo...	1
1	[wonderful, little, production, filming, techn...	1

Table 10.0 – Output of lemmatization

- **Regular Expression:** This involves removing of the remaining special characters in my text data. The output is as shown in the table 11.0 below.

	review	sentiment	sentence_count
0	one reviewer mentioned watching oz episode hoo...	1	166
1	wonderful little production filming technique ...	1	88
2	thought wonderful way spend time hot summer we...	1	88
3	basically family little boy jake think zombie ...	0	65
4	petter mattei s love time money visuall...	1	131

Table 11.0 – regular-expression

Model Building - Supervised Learning: Different models were trained with their outputs evaluated. The models are discussed as shown below.

- **Random Forest with Bag of Words (BOW):**

  The Random Forest with Bag of Words model serves as the baseline model with feature selection and data splitting. The data is split into training and test sets with 70% in training and 30% in test. The split is randomized but stratified to preserve class balance between sets.

  Figure 3.0 shows balanced precision and recall around 0.80 for both classes, indicating no favoring. The 0.80 macro F1-score suggests good overall performance. The equal false positives and negatives are indicated by the 0.80 accuracy. The model was able to predict 6083 as True positive, 5885 as True Negative, 1525 as False positive and 1382 as False negative. Hyperparameter tuning or different text features could further enhance performance. Overall, a promising start without bias.

	precision	recall	f1-score	support
0	0.81	0.79	0.80	7410
1	0.80	0.81	0.81	7465
accuracy			0.80	14875
macro avg	0.80	0.80	0.80	14875
weighted avg	0.80	0.80	0.80	14875

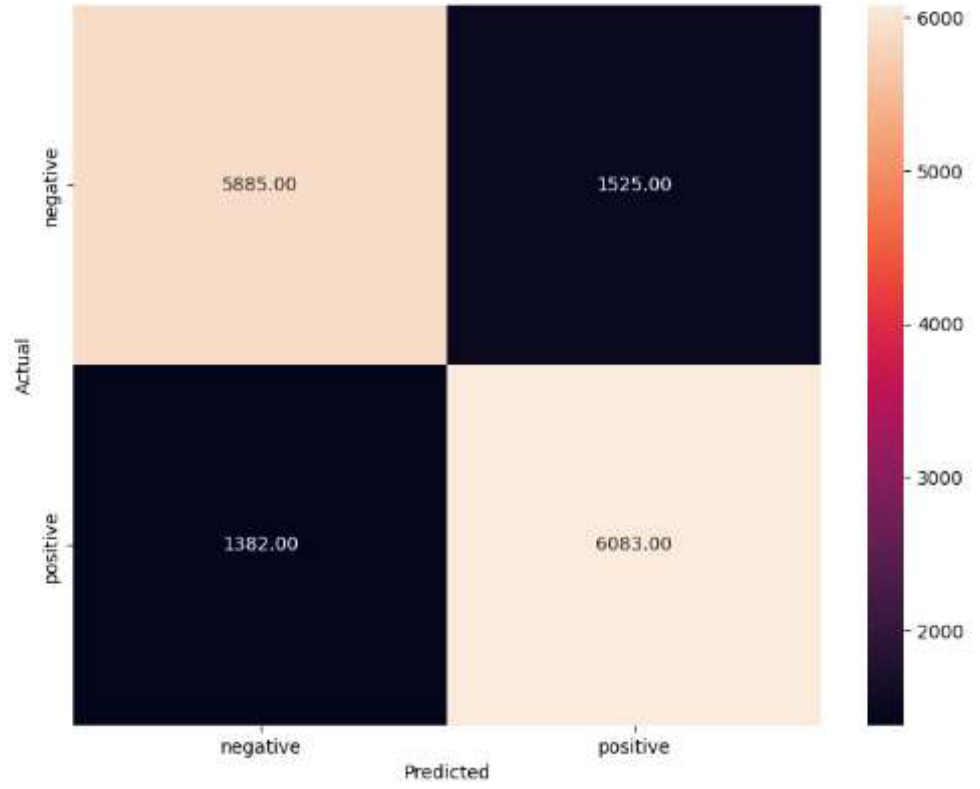

Figure 3.0 – Classification report of random-forest-bag-of-words

Figure 4.0 and figure 5.0 below shows the top 50 words and feature of importance of the words to my base model.

Figure 4.0 Top 50 featured wordcloud

Some of the words observed after building the COW model.

- **Positive words** observed are great, best, better, excellent, good, amazing, love, perfect, loved, brilliant, wonderful, beautiful, life,
- **Negative words** observed are worse, waste, bad, awful, horrible, even, plot, boring, poor, crap.

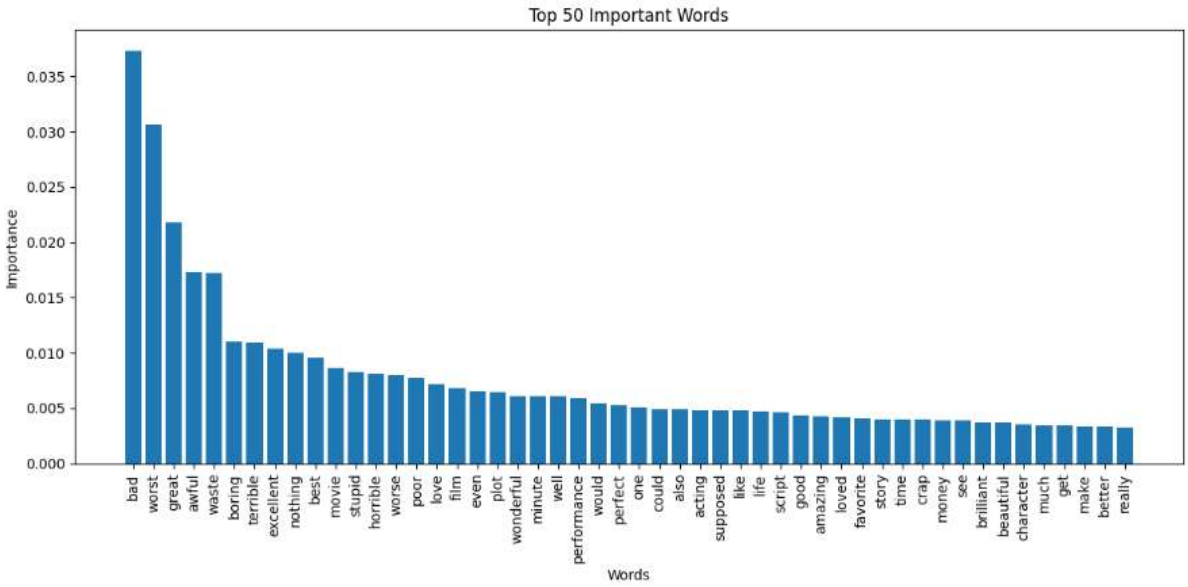

Figure 5.0 – Top 50 important words

- **Random Forest with TF-IDF:** TF-IDF which stands for "Term Frequency-Inverse Document Frequency access words importance based on rarity across corpus and frequency in a document was used with random forest to train the second model on the same dataset. The data split is 80% to training and 20% to test with stratify applied to preserve class balance. Result is as shown in the figure 6.0 below.

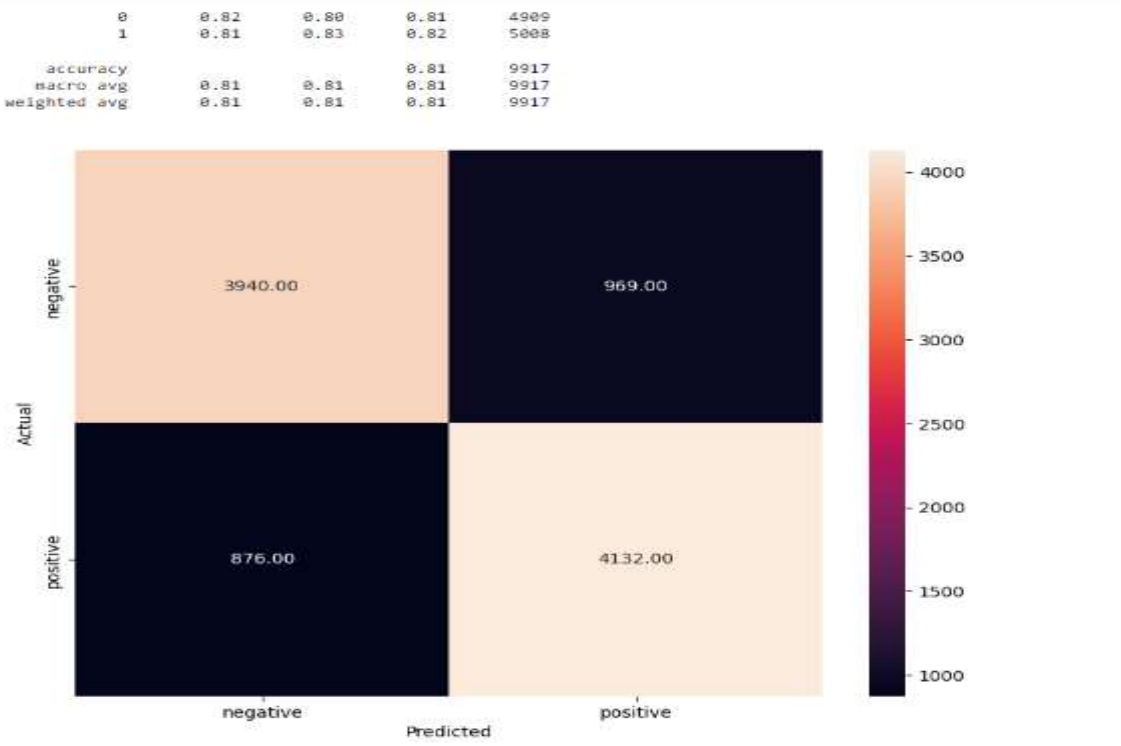

Figure 6.0 -Output random-forest-TF-IDF

The evaluation score for TF-IDF is 0.82, 0.80 and 0.81 for precision, recall and f1-score respectively for the 0 class while 1 class are 0.81,0.83 and 0.82. the accuracy score is 0.81. The confusion metrics for TF-IDF are TN of 3940, FN of 876, FP of 969 and TP of 4132.

- **Random Forest with Word2Vec – Skip-gram:** The 3rd model is random forest with word2vec-skip-gram. The data split is 80% to training and 20% to test with stratify applied to preserve class balance. This model uses the target word as a starting point, to predict the context words (Anand, 2023). The output and the analysis are as shown in figure 7.0 below.

0	0.85	0.82	0.84	4909
1	0.83	0.86	0.84	5008
accuracy			0.84	9917
macro avg	0.84	0.84	0.84	9917
weighted avg	0.84	0.84	0.84	9917

Table 7.0 Performance report of word2vec – skip-gram

The 0 class here has metrics of precision of 0.85, recall of 0.82 and f1-score of 0.84 while the 1s class has precision of 0.83, recall of 0.86 and f1-score of 0.84. The accuracy for random forest with word2vec-skip-gram is 84%. The model was able to predict 4296 as True positive, 4040 as True Negative, 869 as False positive and 712 as False negative.

- **Random Forest with Word2Vec – Continuous Bag of words (CBOW):** Tuning was performed on word2vec to see if there will be more improvement with the result from the earlier trained model. The data split is 80% to training and 20% to test with stratify applied to preserve class balance. Figure 8.0 below shows the out of the performance metrics for CBOW with random forest.

	precision	recall	f1-score	support
0	0.84	0.82	0.83	4909
1	0.83	0.85	0.84	5008
accuracy			0.84	9917
macro avg	0.84	0.84	0.84	9917
weighted avg	0.84	0.84	0.84	9917

Figure 8.0 – CBOW

The performance is almost the same thing with that of Skip gram. The evaluation metrics for the 0 class are precision 0.84, recall 0.82, f1-score 0.83. The 1s class has the following scores, precision = 0.83, recall 0.85 and f1-score 0.84. The model was able to predict 4264 as True positive, 4022 as True Negative, 887 as False positive and 744 as False negative.

Model Building – Deep Learning: Different models were trained with their outputs evaluated. The models are discussed as shown below.

- **Recurrent Neural Networks (RNN):** This algorithm was utilized to train a model that will predict sentiment on imdb movie**.** The RNN architecture consists of a 10,000-word vocabulary embedding layer, followed by a SimpleRNN layer with 100

units. With two units and sigmoid activation, the output layer is a dense layer. The data split is 80% to training and 20% test. It is also designed to Save best RNN model during training to specified file path, with early stopping to prevent overfitting. Below is the model summary report in table 12.0 and epoch output of the trained model on figure 9.0.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 150, 150)          1500000

 simple_rnn (SimpleRNN)      (None, 100)               25100

 dense (Dense)               (None, 2)                 202

=================================================================
Total params: 1,525,302
Trainable params: 1,525,302
Non-trainable params: 0
_________________________________________________________________
```

Table 12.0 – Model Summary - RNN

```
Epoch 1/10
992/992 [==============================] - ETA: 0s - loss: 0.5043 - accuracy: 0.7515
Epoch 1: val_loss improved from inf to 0.40713, saving model to C:\Users\User\Desktop\APPLY-AI\Datasets\Sentiment_analysis\RNN_
Models\rnn_model.h5
992/992 [==============================] - 79s 77ms/step - loss: 0.5043 - accuracy: 0.7515 - val_loss: 0.4071 - val_accuracy:
0.8279
Epoch 2/10
992/992 [==============================] - ETA: 0s - loss: 0.3520 - accuracy: 0.8502
Epoch 2: val_loss did not improve from 0.40713
992/992 [==============================] - 70s 71ms/step - loss: 0.3520 - accuracy: 0.8502 - val_loss: 0.5002 - val_accuracy:
0.7993
Epoch 3/10
992/992 [==============================] - ETA: 0s - loss: 0.2659 - accuracy: 0.8940
Epoch 3: val_loss did not improve from 0.40713
992/992 [==============================] - 73s 74ms/step - loss: 0.2659 - accuracy: 0.8940 - val_loss: 0.4585 - val_accuracy:
0.8202
Epoch 4/10
992/992 [==============================] - ETA: 0s - loss: 0.1580 - accuracy: 0.9420
Epoch 4: val_loss did not improve from 0.40713
992/992 [==============================] - 75s 75ms/step - loss: 0.1580 - accuracy: 0.9420 - val_loss: 0.5336 - val_accuracy:
0.8369
Epoch 4: early stopping
```

Figure 9.0 – Epoch output – RNN

The model's performance produced a range of outcomes. The model showed promise in the early epochs, especially epoch 1, with a validation loss of 0.4071 and validation accuracy of 0.7944. This shows that the model was initially picking up on the underlying trends. However, throughout succeeding epochs, the validation loss rose, and accuracy declined, indicating overfitting or convergence to unfavorable outcomes. Because the validation measures didn't improve during epoch 4, it was decided to use

early stopping, which shows that the model's performance peaked and perhaps wasn't at its best. This is the base model for Deep learning methodology.

- **Long Short-Term Memory (LSTM):** Another algorithm used to train predictive model for the imdb movie is the LSTM. To achieve this the architecture was built using Keras with dense layer for binary classification, LSTM and dropout layers for regularization, embedding layer for word input, and model checkpointing/early stopping to prevent overfitting. It is also designed to Save best LSTM-skip-gram model during training to specified file path. Below is the model summary report in table 13.0 and epoch output of the trained model on figure 10.0.

.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 150, 150)          1500000

 spatial_dropout1d (SpatialD  (None, 150, 150)         0
 ropout1D)

 lstm (LSTM)                  (None, 100)               100400

 dropout (Dropout)            (None, 100)               0

 dense_1 (Dense)              (None, 2)                 202

=================================================================
Total params: 1,600,602
Trainable params: 1,600,602
Non-trainable params: 0
_________________________________________________________________
```

Table 13.0 Model Summary - LSTM

```
Epoch 1/10
992/992 [==============================] - ETA: 0s - loss: 0.3714 - accuracy: 0.8404
Epoch 1: val_loss improved from 0.40713 to 0.30586, saving model to C:\Users\User\Desktop\APPLY-AI\Datasets\Sentiment_analysis
\RNN_Models\rnn_model.h5
992/992 [==============================] - 172s 171ms/step - loss: 0.3714 - accuracy: 0.8404 - val_loss: 0.3059 - val_accuracy:
0.8756
Epoch 2/10
992/992 [==============================] - ETA: 0s - loss: 0.2359 - accuracy: 0.9111
Epoch 2: val_loss improved from 0.30586 to 0.28809, saving model to C:\Users\User\Desktop\APPLY-AI\Datasets\Sentiment_analysis
\RNN_Models\rnn_model.h5
992/992 [==============================] - 168s 170ms/step - loss: 0.2359 - accuracy: 0.9111 - val_loss: 0.2881 - val_accuracy:
0.8795
Epoch 3/10
992/992 [==============================] - ETA: 0s - loss: 0.1805 - accuracy: 0.9324
Epoch 3: val_loss did not improve from 0.28809
992/992 [==============================] - 178s 179ms/step - loss: 0.1805 - accuracy: 0.9324 - val_loss: 0.3111 - val_accuracy:
0.8794
Epoch 4/10
992/992 [==============================] - ETA: 0s - loss: 0.1471 - accuracy: 0.9470
Epoch 4: val_loss did not improve from 0.28809
992/992 [==============================] - 173s 174ms/step - loss: 0.1471 - accuracy: 0.9470 - val_loss: 0.3924 - val_accuracy:
0.8669
Epoch 5/10
992/992 [==============================] - ETA: 0s - loss: 0.1073 - accuracy: 0.9629
Epoch 5: val_loss did not improve from 0.28809
992/992 [==============================] - 176s 178ms/step - loss: 0.1073 - accuracy: 0.9629 - val_loss: 0.3992 - val_accuracy:
0.8752
Epoch 5: early stopping
```

Figure 10.0 – Epoch output LSTM

The model shows signs of overfitting early in training. Training loss and accuracy improve from 0.3714 loss and 84.04% accuracy at epoch 1 to 0.1073 loss and 96.29% accuracy by epoch 5. However, validation loss stops decreasing after epoch 2 where it reaches 0.28809. Validation accuracy peaks at 87.95% in epoch 2 then declines slightly. The growing gap between training and validation performance indicates the model is overfitting after the initial gains, triggering early stopping at epoch 5.

- **Model Improvement - Long Short-Term Memory (LSTM)**: Two LSTM layers (one bidirectional), spatial dropout, embedding input layer, batch normalization, and L2 regularization were implemented to improve the model's sentiment analysis performance. Below is the epoch output of the trained model on figure 11.0.

```
496/496 [==============================] - 273s 543ms/step - loss: 4.5682 - accuracy: 0.5080 - val_loss: 3.7240 - val_accuracy:
0.4921 - lr: 0.0010
Epoch 2/10
496/496 [==============================] - 266s 536ms/step - loss: 2.5257 - accuracy: 0.4829 - val_loss: 1.6328 - val_accuracy:
0.6449 - lr: 0.0010
Epoch 3/10
496/496 [==============================] - 271s 546ms/step - loss: 1.4854 - accuracy: 0.4888 - val_loss: 0.7252 - val_accuracy:
0.6008 - lr: 0.0010
Epoch 4/10
496/496 [==============================] - 261s 526ms/step - loss: 1.6698 - accuracy: 0.4897 - val_loss: 1.0139 - val_accuracy:
0.4591 - lr: 0.0010
Epoch 5/10
496/496 [==============================] - 267s 539ms/step - loss: 1.2558 - accuracy: 0.4802 - val_loss: 4.6397 - val_accuracy:
0.5278 - lr: 0.0010
Epoch 6/10
496/496 [==============================] - 266s 536ms/step - loss: 3.6622 - accuracy: 0.4407 - val_loss: 7.2436 - val_accuracy:
0.4812 - lr: 5.0000e-04
Epoch 7/10
496/496 [==============================] - 260s 524ms/step - loss: 4.4112 - accuracy: 0.5064 - val_loss: 7.0393 - val_accuracy:
0.4973 - lr: 5.0000e-04
Epoch 8/10
496/496 [==============================] - 299s 603ms/step - loss: 5.4531 - accuracy: 0.5015 - val_loss: 7.3429 - val_accuracy:
0.4746 - lr: 2.5000e-04
Epoch 9/10
496/496 [==============================] - 266s 537ms/step - loss: 5.3278 - accuracy: 0.5102 - val_loss: 7.3503 - val_accuracy:
0.5249 - lr: 2.5000e-04
Epoch 10/10
496/496 [==============================] - 268s 540ms/step - loss: 5.0962 - accuracy: 0.4980 - val_loss: 5.5458 - val_accuracy:
0.4759 - lr: 1.2500e-04
```

Figure 11.0 – Epoch – Model Improvement – LSTM

Over ten epochs, training and validation accuracy stagnated around 50%, suggesting the model struggled to learn, with accuracy remaining near random guess levels despite tuning, indicating suboptimal performance.

Result and Evaluation:

- **Supervised Learning Models**

  For clearer understanding, the model's performance for supervised learning is tabulated and evaluated in the table 14.0 below.

S/N	Models Perfomance on Class 0	Precision	Recall	F1-Score	Accuracy
	RF with BOW (Base Model)	0.81	0.79	0.8	0.8
	RF withTF-IDF	0.82	0.8	0.81	0.81
	RF with Word2Vec - Skip-gram	0.85	0.82	0.84	0.84
	RF with Word2Vec - CBOW	0.84	0.82	0.83	0.84
S/N	Models Perfomance on Class 1	Precision	Recall	F1-Score	Accuracy
	RF with BOW	0.8	0.81	0.81	0.8
	RF withTF-IDF	0.81	0.83	0.82	0.81
	RF with Word2Vec - Skip-gram	0.83	0.86	0.84	0.84
	RF with Word2Vec - CBOW	0.83	0.84	0.84	0.84
RF	Random Forest				
BOW	Bag of Words				
TF-IDF	Term Frequency-Inverse Document Frequency				
CBOW	Continuos Bag of Words				

Table 14.0 Summary performance report of the Supervised Learning Models

The RF with Word2Vec skip-gram model showed the biggest improvements over the base model, outperforming on all metrics for both classes. It improved class 0 precision, recall, F1, and accuracy by 0.04, 0.03, 0.04, and 0.04 respectively over base model. For class 1 it increased precision, recall, F1, and accuracy by 0.03, 0.05, 0.03, and 0.04. RF with TF-IDF performed second best, with small gains over base model on most metrics. RF with Word2Vec CBOW performed like TF-IDF. Overall, RF with Word2Vec skip-gram was the best model, likely because of its ability to capture semantic meaning in the word embeddings.

Skip-gram also had the best confusion evaluation metrics with True Positive as 4296, True Negative 4040, False Positive 869, and False negative as 712.

- **Deep Learning Models (RNN & LSTM):** The summary of the top epoch performances of RNN and LSTM are shown in the table `15.0 below. Model improved model using LSTM was not included because the performance was too poor.

S/N	Models Perfomance on Class 1	Loss	Accuracy	Val_loss	Val_acc
1	LSTM	0.2359	0.9111	0.2881	0.8795
2	RNN	0.352	0.8502	0.5002	0.7993
LSTM	Lomg Short Term Memory				
RNN	Recurrent Neural network				
Val_Loss	Validation Loss				
Val_acc	Validation Accuracy				

Table 15.0 Summary performance report of the deep Learning Models -RNN & LSTM

The LSTM model performed better with lower loss of 0.2359 and higher accuracy of 91.11% on the training data compared to the RNN's 0.352 loss and 85.02% accuracy. The LSTM also generalized better with lower validation loss of 0.2881 and higher validation accuracy of 87.95% versus the RNN's 0.5002 validation loss and 79.93% accuracy. Overall, the LSTM model clearly outperformed the RNN, likely due to its ability to capture longer term dependencies.

Outcomes:

This work successfully built an LSTM model to analyze sentiment polarity in IMDB reviews, achieving the objective of building a predictive model.

Comparative analysis which is the second objective showed LSTM outperformed methods like BOW, Word2Vec and RNN for interpreting viewer attitudes.

Another observation is Unlike Banerjee et al. (2022) who found poor Random Forest performance with accuracy score of 51%, this work demonstrated tuned Random Forest models can exceed 80% accuracy on imbd dataset.

The evaluation evidenced LSTM as the preferred technique, with the trained model enabling sentiment prediction to support data-driven decisions in the movie industry. Overall, this

research contributes to effectively applying deep learning for text analysis and strategic insight generation.

Recommendations:

Though Word2Vec skip-gram performed well, I recommend deploying the LSTM model for sentiment prediction on IMDB reviews. LSTM is a more advanced deep learning technique better suited for textual data, as its recurrent connections can capture contextual information and long-range dependencies that Skip-gram lacks by treating words independently. With further tuning, LSTM can improve upon its high validation accuracy of 0.88, while Skip-gram is more limited in feature learning from sequences. LSTM's sequence processing strengths make it optimal for deploying on nuanced movie review text.

Conclusion:

This research demonstrated sentiment analysis of IMDB reviews using deep learning, with an LSTM model achieving 88% validation accuracy. The high-accuracy model provides valuable insights into audience attitudes to enable data-driven decisions in the movie industry. Future work could expand the dataset, add modalities, explore transfer learning applications, and deploy the model for real-time analysis. Overall, this study makes a useful contribution applying NLP and a top-performing LSTM model to extract strategic insights from user-generated content.

References:

Anand, A. (2023) *Word Embeddings: Techniques, Types and Applications in NLP | Analytics Steps*. www.analyticssteps.com. Available online: https://www.analyticssteps.com/blogs/word-embeddings-techniques-types-and-applications-nlp.

Banerjee, D., Mazumder, S. and Datta, S. (2022) Comparative Study on Sentiment Analysis on IMDB Dataset. *IJARCCE*, 11(3).

Brownlee, J. (2020) *Why Use Ensemble Learning?* Machine Learning Mastery. Available online: https://machinelearningmastery.com/why-use-ensemble-learning/.

IMDb (2008) *IMDb - Movies, TV and Celebrities*. IMDb. Available online: https://www.imdb.com/.

Kim, S. (Josh) (2022) *Primer on Cleaning Text Data*. Medium. Available online: https://towardsdatascience.com/primer-to-cleaning-text-data-7e856d6e5791#:~:text=Text%20cleaning%20here%20refers%20to.

MathWorks (n.d.) *What Is Word2vec?* www.mathworks.com. Available online: https://www.mathworks.com/discovery/word2vec.html#:~:text=Word2vec%20is%20one%20of%20the [Accessed 19 Aug. 2023].

ProjectPro (2023) *Machine Learning NLP Text Classification Algorithms and Models*. ProjectPro. Available online: https://www.projectpro.io/article/machine-learning-nlp-text-classification-algorithms-and-models/523.

Raj, N. (2021) *Sentiment Analysis | Sentiment Analysis in Natural Language Processing*. Analytics Vidhya. Available online: https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/.

Shekhar, S. (2021) *What is LSTM for Text Classification?* Analytics Vidhya. Available online: https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/#:~:text=In%20conclusion%2C%20LSTM%20(Long%20Short.