DA 4210 - TEXT ANALYTICS

# IMDB MOVIE

## SENTIMENTAL REVIEW ANALYSIS


ASSIGNMENT 01

INDEX NUMBER: 206001F

# Table of Contents

## Introduction

In many situations, public opinion and comments are important components, especially when using sentiment analysis to assess how movies are viewed by the public. The utilization of large-scale text data, such as the IMDB 50k row dataset on Kaggle, which includes many movie reviews, greatly improves this analytical process. We aim to create a strong model that can independently determine whether these reviews are positive or negative. We want to make use of neural networks' capacity to identify complex patterns in textual data by using them for sentiment classification.

Neural networks are the best option for tasks involving natural language processing because they provide unmatched advantages in handling complex language structures. Their ability to identify subtle facial expressions and contextual cues enables them to precisely categorize feelings in film assessments. This method not only makes it easier for reviews to be automatically categorized, but it also makes it possible to collect insightful data on audience perceptions. Filmmakers, producers, and distributors can make better decisions by using large-scale sentiment analysis to identify patterns, preferences, and opportunities for development in the business.

Ultimately, my effort is to create a neural network-based sentiment analysis model.

## Problem Statement

The task at hand is to use the IMDB 50k row dataset from Kaggle to create a neural network-based sentiment categorization model. Accurately categorizing movie reviews as having positive or negative sentiments is the aim. This categorization exercise is crucial for comprehending how people react to films, assisting producers, directors, and moviegoers in determining the general attitude surrounding a given picture. I can effectively process many movie reviews and obtain insightful data about audience attitudes by automating the sentiment analysis process. Training a neural network model to efficiently identify the underlying patterns in the text data to produce accurate sentiment predictions is the difficult part.

The purpose of this Analysis is to show how well neural networks perform sentiment classification tasks and to offer important insights into the distribution of sentiments found in the IMDB movie review dataset.

# Data Preprocessing

## Exploratory Data Analysis

```
[ ] # Read the CSV file and take only stratified 20000 rows
    df = pd.read_csv("/content/drive/MyDrive/Text analytics/a1_IMDB_Dataset.csv", nrows=20000)

    # Shuffle the dataframe to ensure the rows are randomly selected
    df = df.sample(frac=1).reset_index(drop=True)
```

**Data Preprocessing**

```
[ ] df.shape

    (20000, 2)
```

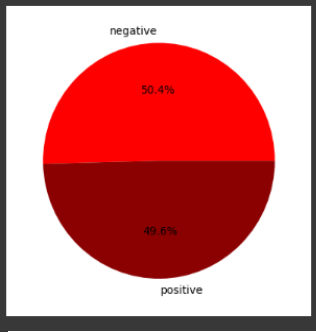
```
    df.head()
```

| | review | sentiment |
|---|---|---|
| 0 | "Hood Rat" is absolutely terrible. This is a u... | negative |
| 1 | Maybe I loved this movie so much in part becau... | positive |
| 2 | Ridiculous fluff, that compounds its error by ... | negative |
| 3 | I am new at this, so bear with me please. I am... | positive |
| 4 | I saw 'I Smell the Dead' -- sorry, 'I SELL the... | positive |

I started by using the pandas library to import the dataset into Python. After loading the dataset, I extracted 20,000 rows using stratified sampling, ensuring the relative proportions of the various categories were preserved. To ensure the integrity of the sampling process, I examined the extracted dataset's form after the process to make sure it included 20,000 rows. Lastly, I used the df. head() method to show the first few rows of the dataset so that the structure and contents could be examined to get a general overview of the data.

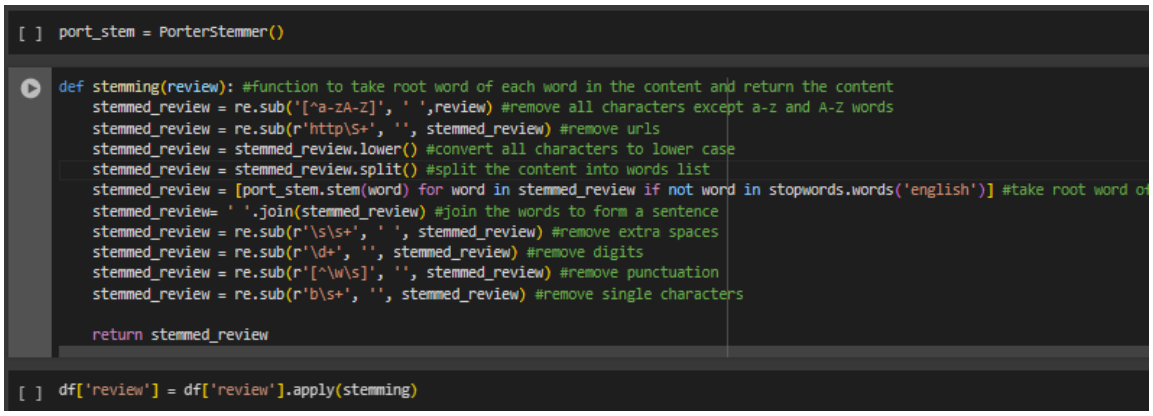
```
[ ] df.isnull().values.any()

    False

[ ] df['review'].isnull().sum()

    0

[ ] df['review'].duplicated().sum()

    74

[ ] df = df.drop_duplicates(subset=['review'])

[ ] df['review'].duplicated().sum()

    0
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create the pie chart
plt.pie(df["sentiment"].value_counts(), labels=df["sentiment"].unique(), autopct="%1.1f%%", colors=["red", "darkred"])

# Display both plots
plt.show()
```



Examining duplicate values came next, following confirmation that the dataset had no null values. 74 duplicate entries were discovered. These duplicate rows were eliminated from the dataset to protect data integrity and prevent biases in analysis. I next evaluated if there was an imbalance in the dataset. Examining the dataset revealed that it was balanced, therefore no additional correction was needed.

**Text preprocessing & analysis**



```
[ ] port_stem = PorterStemmer()

def stemming(review): #function to take root word of each word in the content and return the content
    stemmed_review = re.sub('[^a-zA-Z]', ' ',review) #remove all characters except a-z and A-Z words
    stemmed_review = re.sub(r'http\S+', '', stemmed_review) #remove urls
    stemmed_review = stemmed_review.lower() #convert all characters to lower case
    stemmed_review = stemmed_review.split() #split the content into words list
    stemmed_review = [port_stem.stem(word) for word in stemmed_review if not word in stopwords.words('english')] #take root word o
    stemmed_review= ' '.join(stemmed_review) #join the words to form a sentence
    stemmed_review = re.sub(r'\s\s+', ' ', stemmed_review) #remove extra spaces
    stemmed_review = re.sub(r'\d+', '', stemmed_review) #remove digits
    stemmed_review = re.sub(r'[^\w\s]', '', stemmed_review) #remove punctuation
    stemmed_review = re.sub(r'b\s+', '', stemmed_review) #remove single characters

    return stemmed_review

[ ] df['review'] = df['review'].apply(stemming)
```

In this function, I try to preprocess text data effectively to prepare it for subsequent analysis. The process involves several key steps to ensure the text is standardized and cleaned comprehensively.

Firstly, we remove any characters that are not alphabets (a-z and A-Z) from the content, ensuring that only valid words are retained. This step helps in eliminating unnecessary symbols and special characters that may not contribute to the analysis. Next, we address the removal of URLs from the text. This is crucial to eliminate any hyperlinks present in the content, as they are not relevant to the text analysis and might interfere with subsequent processing steps. For analysis and consistency, every character in the text has been changed to lowercase.

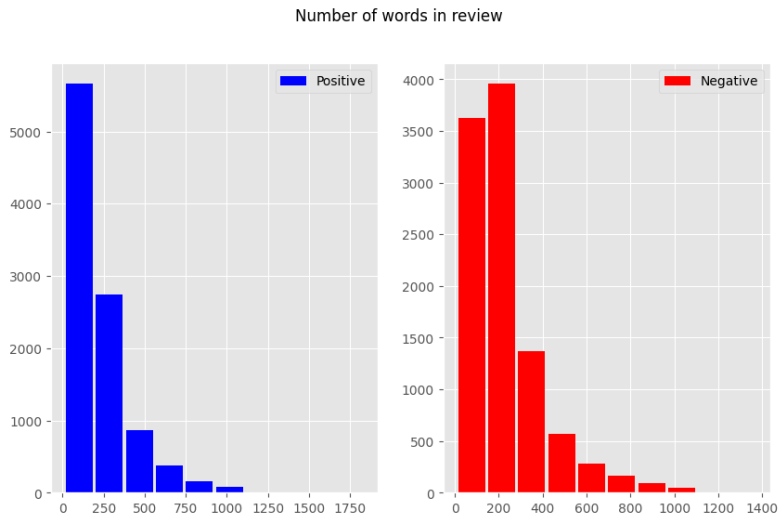
Then, the text is then divided into individual words to create a list. This stage makes the text ready for more in-depth examination by preparing it for word-by-word processing. Further, stemming is applied to each word to extract its root form. Stemming lowers words to their base or root form, enabling normalization, and lowering the dimensionality of the text input. This helps in improving the efficiency and effectiveness of subsequent analysis tasks. After stemming, the words are rejoined to form coherent sentences, with proper spacing between words. This stage legibly reconstructs the text for further analysis and interpretation.

To preserve cleanliness and readability, any unnecessary spaces in the text are deleted, maintaining a uniform structure throughout. Text that contains numbers, punctuation, and single characters is also eliminated. This guarantees that there are only meaningful words in the text, improving the quality of the data for further analysis.

```
[ ]  def remove_single(review):
         # Define a pattern to match single or two-character words
         pattern = r'\b\w{1,2}\b'  # \b indicates word boundaries, \w{1,2} matches one or two word characters

         # Use re.sub() to replace matched patterns with an empty string
         cleaned_review = re.sub(pattern, '', review)

         return cleaned_review
```

```
[ ]  df['review'] = df['review'].apply(remove_single)
```

My aim with this function is to find and eliminate terms with one or two characters from the text data.




All things considered, this thorough pretreatment guarantees that the text data is standardized, cleaned, and prepared for a variety of analytical applications.

Then we converted the sentiment column which had negative positive into numerical 0 and 1

```
[ ]  df.sentiment.replace("positive", 1, inplace = True)
     df.sentiment.replace("negative", 0, inplace = True)
```

```
df.head()
```

| | review | sentiment | word count |
|---|---|---|---|
| 0 | admit absolut love movi cours sure know malcol... | 1 | 278 |
| 1 | suppos novel much downer mobi dick would find ... | 0 | 301 |
| 2 | hulk alien beasti crash land earth soon wreck ... | 0 | 230 |
| 3 | farscap total rule opinion close babylon five ... | 1 | 102 |
| 4 | warn spoiler ahead even doubt anybodi seen yet... | 0 | 189 |

Then we checked good review frequent words and bad review frequent words

```
[31] from collections import Counter
     count = Counter()
     for text in pos_reviews['review'].values:
         for word in text.split():
             count[word] +=1
     count.most_common(10)
```

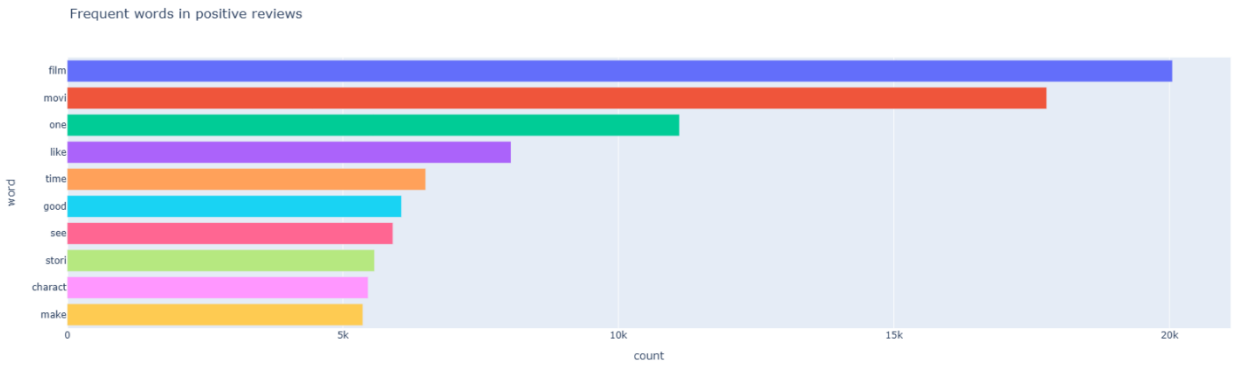
```
[('film', 20053),
 ('movi', 17770),
 ('one', 11108),
 ('like', 8051),
 ('time', 6502),
 ('good', 6063),
 ('see', 5907),
 ('stori', 5573),
 ('charact', 5458),
 ('make', 5362)]
```

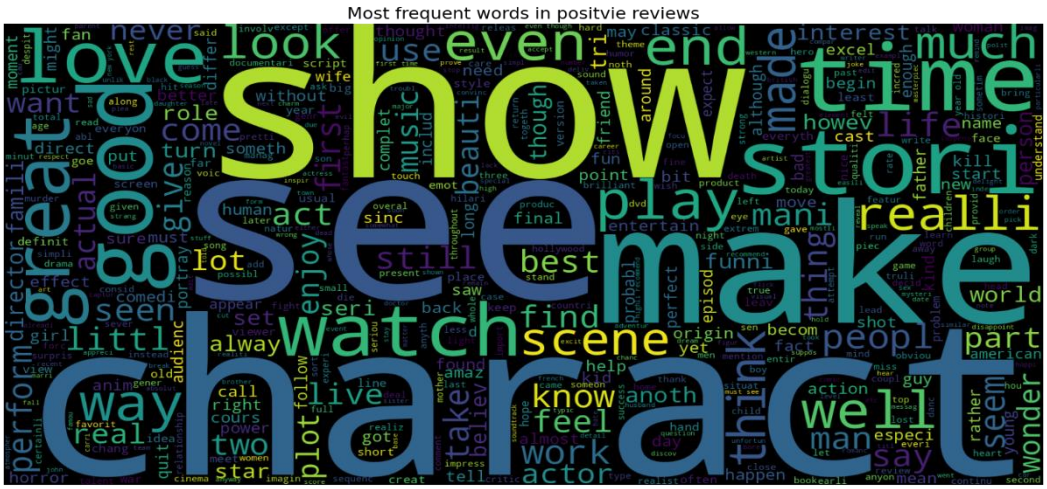
```
count = Counter()
for text in neg_reviews['review'].values:
    for word in text.split():
        count[word] += 1
count.most_common(10)
```

```
[('movi', 23420),
 ('film', 17861),
 ('one', 10807),
 ('like', 9848),
 ('make', 6623),
 ('get', 6310),
 ('even', 6207),
 ('time', 6129),
 ('good', 6057),
 ('bad', 5969)]
```

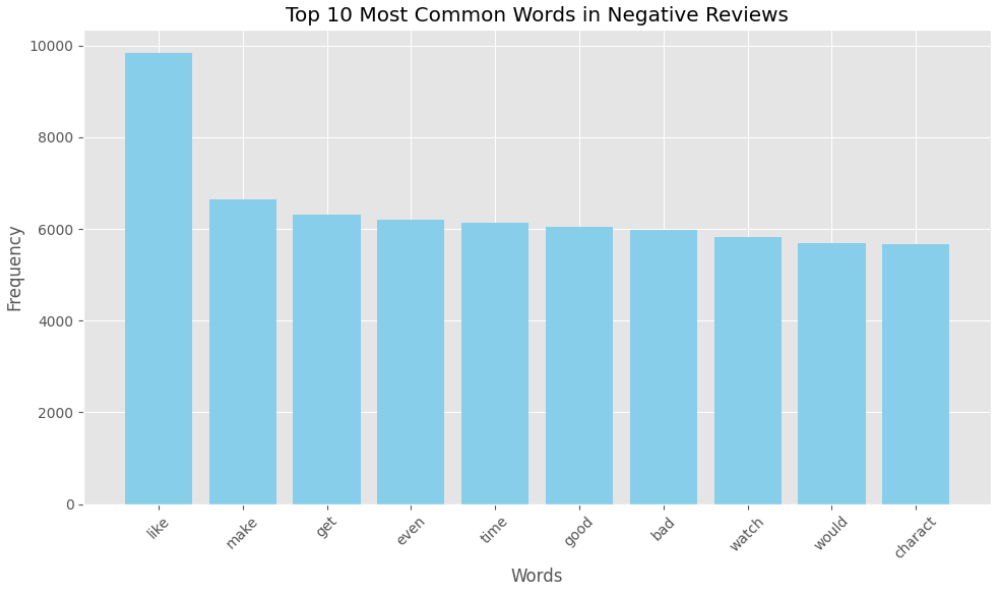
```
[10] # Words to remove
     words_to_remove = ['movi', 'film', 'one']

     # Removing the words from the review column
     for word in words_to_remove:
         df['review'] = df['review'].str.replace(word, '')
```

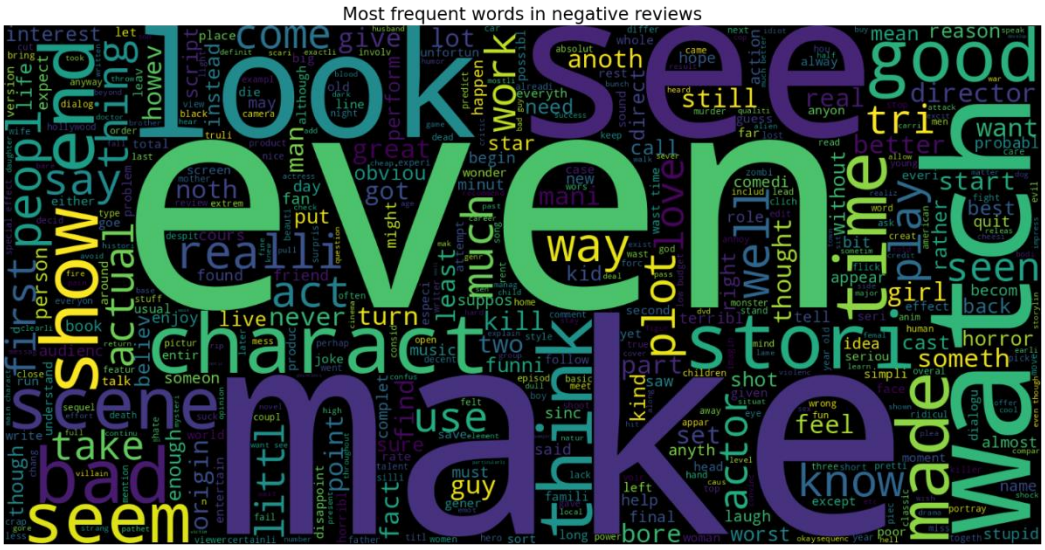
After identifying the top three most frequent words common to both the 'negative' and 'positive' categories, we have excluded these words from the dataset to enhance the quality of the analysis.



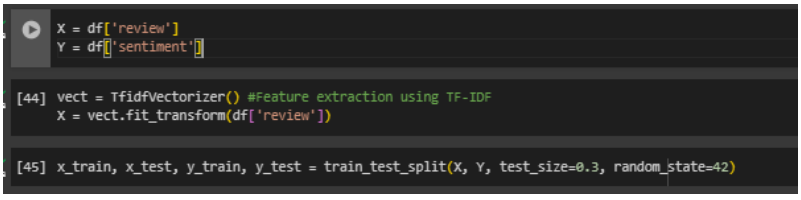

This visualization allows for a clear understanding of the most prevalent positive terms.




This visualization allows for a clear understanding of the most prevalent negative terms.

## Splitting the data



```
X = df['review']
Y = df['sentiment']

[44] vect = TfidfVectorizer() #Feature extraction using TF-IDF
     X = vect.fit_transform(df['review'])

[45] x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

First, we separate variable x as review column and y as sentiment column. Then, we use TfidfVectorizer for feature extraction in sentiment analysis tasks for the When compared to word counts. TF-IDF offers a more accurate representation of text data. Sentiment analysis results may be more precise because of its emphasis on word importance rather than just frequency in a document. Then, As is commonly done, we split the dataset into two subsets: a larger training subset and a smaller testing subset, with a 70:30 ratio maintained in both. This distribution finds a compromise between setting aside enough data for training to fully capture underlying patterns and keeping some for testing to evaluate the model's generalization capabilities.

```
[47] x_train = x_train.toarray()
     x_test = x_test.toarray()
```

Then We convert a sparse matrix x_train into a dense numpy array

## Model building.

The architecture of a sequential neural network was used to create a sentimental model. Layers can be stacked on top of one another in sequential models, making it easier to create a linear stack of layers.

The model architecture comprised three densely connected layers.

1. Input Layer - 32 units using the Rectified Linear Unit (ReLU) activation function made up the first layer. ReLU is frequently utilized to add non-linearity to models.
2. Hidden Layer - 16 units with the ReLU activation function were also present in the second layer. The network can identify intricate patterns in the data by using hidden layers.
3. Output Layer - A single unit with the Sigmoid activation function made up the last layer. For binary classification problems, sigmoid activation was selected because it provides probability for binary outcomes by squashing the output between 0 and 1.

With a learning rate of 0.001, the Adam optimizer was used to construct the model. Adam is a technique for adaptive learning rate optimization that works well for deep-learning neural networks. As is usual for binary classification issues, binary cross-entropy was selected as the loss function.

Furthermore, accuracy was chosen as the performance indicator for the model to be tracked throughout training.

## Model evaluation.

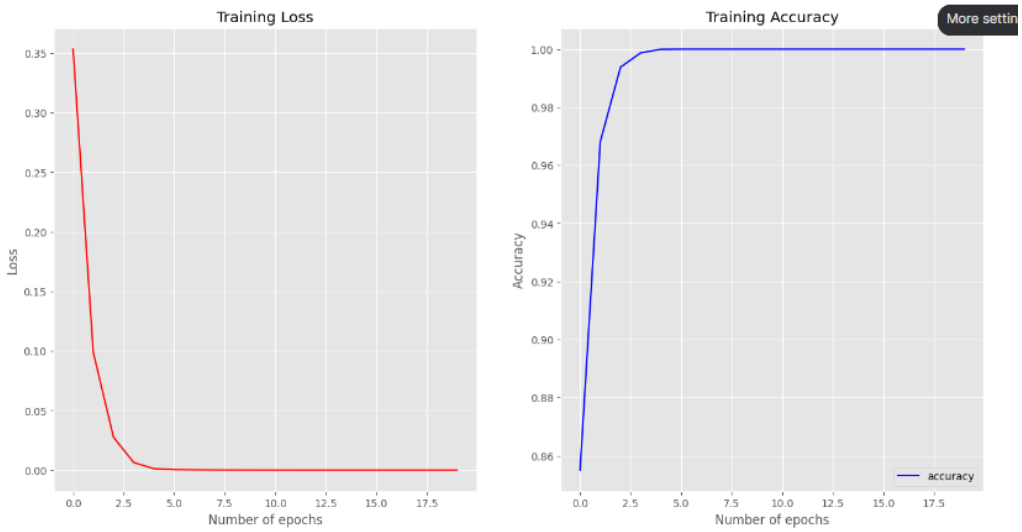
The training dataset, which included features (x_train) and matching binary labels (y_train), was used to train the model throughout 20 epochs with a batch size of 15. To optimize both computational efficiency and model performance, a batch size of 15 was used. The model was trained for 20 epochs, which allowed it to iterate over the full dataset numerous times and enhance its capacity to identify patterns in the data.

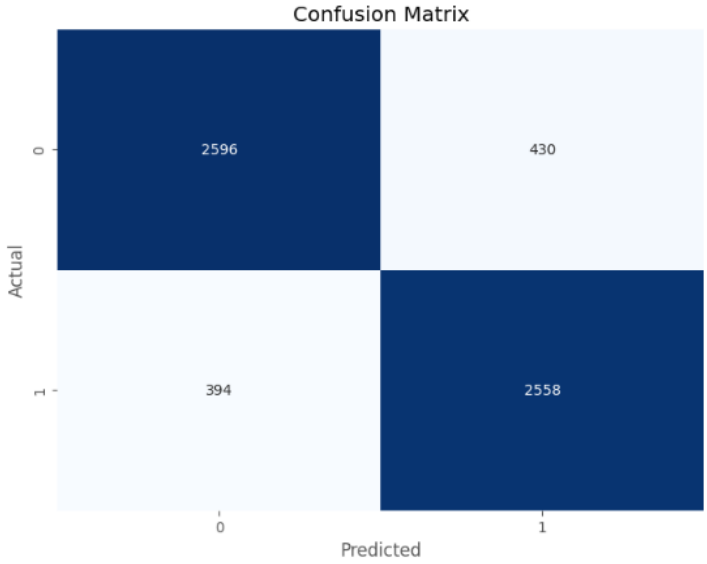
The test dataset was used to assess the model after it had finished training. The following are the evaluation's findings.

- Test Loss: 1.199
- Test Accuracy: 86.75%

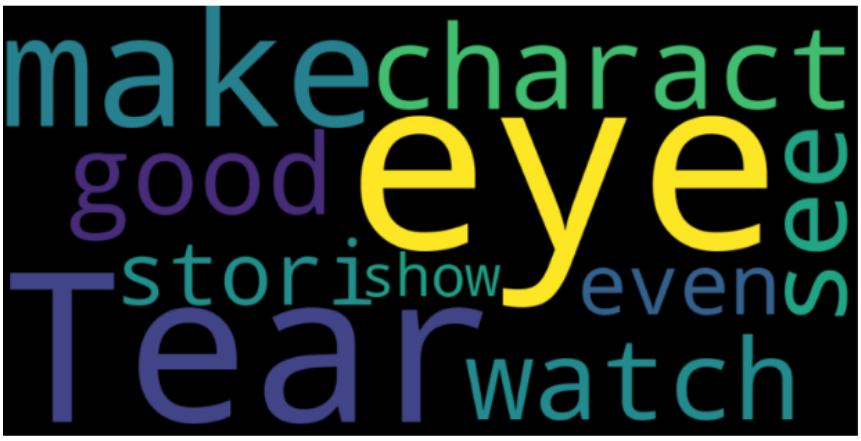
The test loss shows how well the model's predictions match the actual labels by expressing the value of the selected loss function (binary cross-entropy) on the test dataset. Better alignment is indicated by a reduced test loss.

The percentage of successfully categorized samples in the test dataset is represented by the test accuracy. It sheds light on how well the model performs overall in terms of producing precise predictions on unobserved data. The model's 86.75% accuracy in this instance shows that it can successfully categorize cases into the binary categories.






We use the confusion matrix to assess the effectiveness of our sentiment analysis model. Of the 5978 test data points, 2596 were correctly classified as true negatives, demonstrating that the algorithm could identify negative sentiment in certain cases. In a similar vein, 2558 test data points were appropriately categorized as true positives, presenting cases in which positive sentiment was appropriately detected. This suggests that our algorithm is capable of accurately identifying between sentiment that is positive and sentiment that is negative inside the movie review dataset.




Furthermore, looking at the word cloud created from the test data, we discover that terms like "good", “see” and "story" are common, indicating a positive attitude. The efficacy of our model is further supported by the agreement between the real sentiment in the testing data and the identified positive phrases.

After taking these things into account, we can state with confidence that our sentiment analysis model does a good job of categorizing movie reviews and telling positive from negative sentiment. The precision of sentiment categorization, exemplified by the confusion matrix and bolstered by the abundance of affirmative terms in the test data, highlights the dependability and effectiveness of our model. The favorable outcomes suggest that our model can be effectively utilized in sentiment analysis jobs.

## Conclusion

Ultimately, our effort to create a sentiment analysis model for movie reviews using neural networks has been successful and significant. Making use of the IMDB 50k row dataset, our goal was to reliably classify reviews as favorable or negative, an undertaking crucial to comprehending public opinion on films. After conducting thorough data pretreatment, exploratory data analysis, and model construction, we have developed a strong model that can identify complex patterns in textual data and produce correct sentiment predictions.

We surmounted difficulties including complicated data pretreatment, intricate visualization, and limited computational resources by working hard and creatively. We have proven the practicality and efficiency of our method in sentiment analysis tasks by successfully resolving these issues.

Our model's evaluation produced encouraging findings, showing that it can correctly classify attitudes in unseen data with a test accuracy of 86.75%. The model's performance is further validated by the confusion matrix, which shows that it can accurately recognize both positive and negative thoughts.

Furthermore, the testing data's word cloud analysis demonstrated the model's effectiveness visually, with the majority of positive phrases reflecting the actual opinions stated in the evaluations. The fact that the anticipated sentiment and actual observations agree highlights how trustworthy and useful our sentiment analysis methodology is.

In summary, the sentiment analysis model we have developed using neural networks offers producers, distributors, and filmmakers a useful tool for understanding audience opinions and making wise decisions in the ever-changing film industry. This model's successful development and validation represents a major advancement in the use of cutting-edge technologies for sentiment analysis applications, opening new research and practical application opportunities across a range of industries.