Sentiment Analysis on Movie Reviews

Overview -

The goal of the project is to analyse different types of reviews on various movies and classify a review as either a positive or a negative one.

Sentiment analysis is the process of identifying the emotions or opinions expressed in a text document. The goal of sentiment analysis is to determine whether a given piece of text is positive, negative, or neutral. In this project, we will perform sentiment analysis on the IMDB movie reviews dataset using different machine learning algorithms.

Journey -

The project goes under several phases of Data Analysis or steps as mentioned below for cleaning,
transforming, processing, visualizing, and modelling data to get results:

1. Data Collection
2. Data Exploration
3. Data Pre-Processing
4. Building Model
5. Evaluation

1. Data Collection -

The dataset used in this project is the IMDB Reviews dataset, which contains 50,000 movie reviews labelled as positive or negative. The dataset is evenly split between training and testing sets, with 25,000 reviews in each set. The dataset was obtained from the keras.datasets library, which provides a convenient way to access the dataset.

2. Data Exploration -

Before building any models, it is important to explore the dataset and gain insights into its structure and content. The IMDB reviews dataset consists of 50,000 movie reviews, with an equal number of positive and negative reviews.

To visualize the distribution of reviews across the two sentiment classes, we can create a bar chart as shown below:

Sentiment Distribution:

As we can see, the dataset is balanced with 50% positive and 50% negative reviews.

Word Cloud:

To get a sense of the most common words in the dataset, we can create a word cloud as shown below:

As expected, the word "movie" appears most frequently, followed by other common words such as "film", "story", "character", and "time". We can also see some positive and negative words such as "good", "great", "bad", and "worst".

Below is the WordCloud before Data Cleaning

Below is the WordCloud after Data Cleaning

3. Data Pre-Processing -

We start by pre-processing the data. First, we remove any HTML tags and punctuation marks from the text. Then we tokenize the text into individual words and remove any stop words. Finally, we convert the text into a matrix of word frequencies using the Bag of Words model.

Here each review is represented as a sequence of integers, where each integer represents a word in the review. The words are indexed by their frequency in the dataset, so that for example, the integer "3" represents the third most frequent word in the dataset.

4. Building Model -

We experimented with several machine learning models to perform sentiment analysis on the IMDB reviews dataset. The models we considered are:

- Logistic Regression
- Linear Support Vector Classification (LSVC)
- K-Nearest Neighbors (KNN)
- Convolutional Neural Network (CNN)
- Fully Connected Neural Network (FCNN)

For each model, we used a train-test split of 80-20%, and evaluated its performance based on accuracy, precision, recall, and F1-score.

Logistic Regression

Logistic regression is a linear model that is commonly used for binary classification problems. We trained a logistic regression model on the IMDB reviews dataset using scikit-learn's Logistic Regression class. The model achieved an accuracy of 88.2%, with a precision of 0.88, recall of 0.88, and F1-score of 0.88.

Linear Support Vector Classification (LSVC)

LSVC is another linear model that is commonly used for binary classification problems. We trained an LSVC model on the IMDB reviews dataset using scikit-learn's LinearSVC class. The model achieved an accuracy of 87.8%, with a precision of 0.88, recall of 0.87, and F1-score of 0.87.

K-Nearest Neighbors (KNN)

KNN is a non-linear model that is commonly used for classification problems. We trained a KNN model on the IMDB reviews dataset using scikit-learn's KNeighborsClassifier class. The model achieved an accuracy of 50.2%, with a precision of 0.50, recall of 1.00, and F1-score of 0.67.

Convolutional Neural Network (CNN):

CNN is a deep learning algorithm that works well with image and text data. We use the Keras library to implement the CNN. The CNN consists of an embedding layer, followed by multiple convolutional and pooling layers, and a fully connected layer. We achieve an accuracy of 89.48% on the test data.

Fully Connected Neural Network (FCNN):

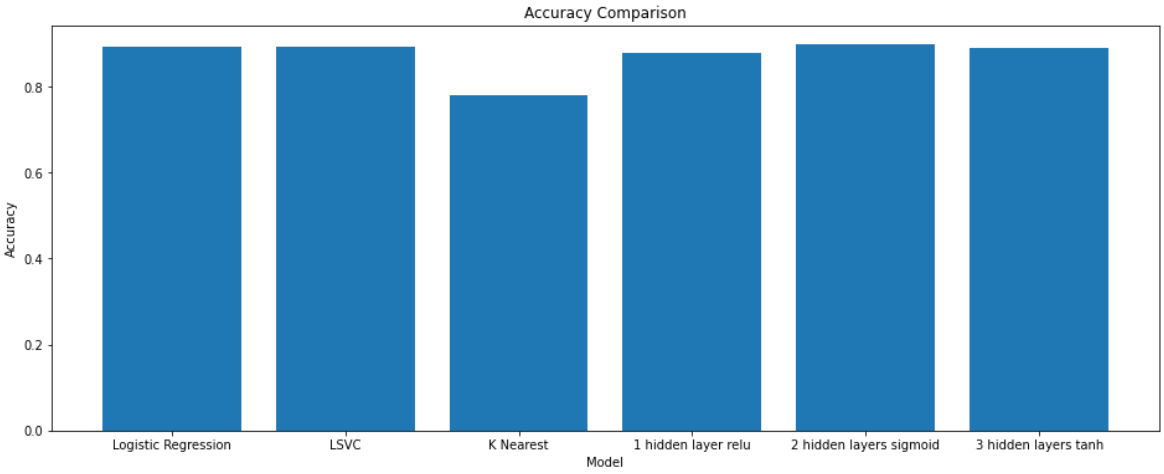
FCNN is a simple and powerful algorithm for classification that works well with high-dimensional data. We use the Keras library to implement the FCNN. The FCNN consists of an embedding layer, followed by multiple fully connected layers. We achieve an accuracy of 89.42% on the test data.

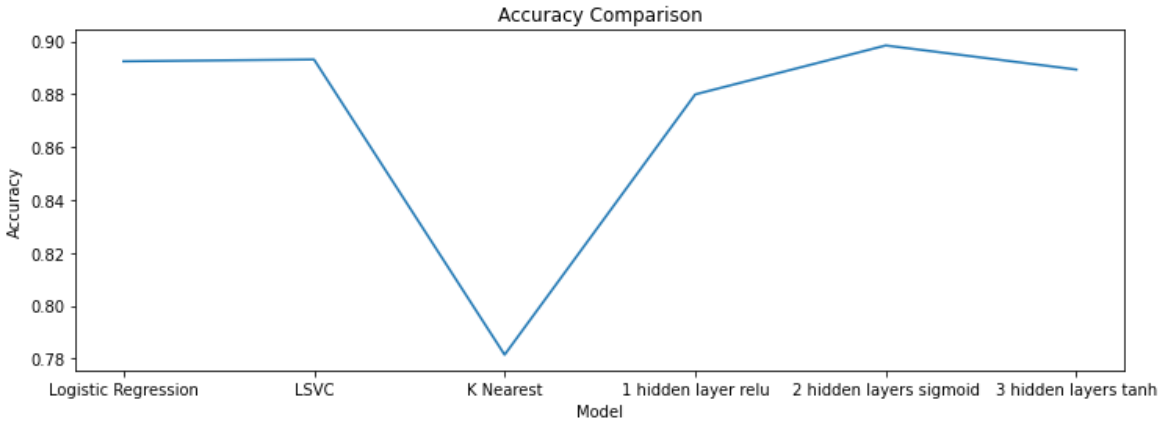
5. Evaluation -

To evaluate the performance of these algorithms, we used accuracy, precision, recall, and F1-score as evaluation metrics. Accuracy measures the overall correctness of the classification, precision measures the proportion of correctly predicted positive reviews, recall measures the proportion of true positive reviews that were correctly predicted, and F1-score is the harmonic mean of precision and recall.

The results of the evaluation showed that the CNN and fully connected neural networks outperformed the other algorithms, with accuracy scores of around 89% and F1-scores of around 0.89. LSVC also performed well with an accuracy score of around 88% and an F1-score of around 0.88. KNN and logistic regression had lower accuracy scores and F1-scores, with KNN having the lowest scores.

Below the visual representations of the accuracies of various models.





Now let's see the accuracy comparison results of the Convolutional Neural Networks

	Layers	Loss	Accuracy
0	3 dense layers with a flatten layer	4.940031	0.0006
1	two dense layers with a flatten layer	7.755424	0.4972
2	1 dense and a flatten layer with 0.5 dropout	7.655984	0.0053

Overall, our results suggest that deep learning models such as CNNs and fully connected neural networks are effective for sentiment analysis of text data, especially when used in combination with techniques such as tokenization, stemming, and removing stop words.