**“Sentiment Analysis using Machine Learning and Deep Learning Algorithms”**

**Project Report**

By

Sathvick Reddy Narahari

Student ID: 016003520

To

Professor Shilpa Gupta

On

May 15, 2023

**Problem definition**

Sentiment analysis, also known as opinion mining, is a field of natural language processing (NLP) that involves determining and categorizing the sentiment expressed in a piece of text, such as reviews, social media posts, or customer feedback. The goal of sentiment analysis is to automatically identify and classify the sentiment or emotion conveyed in the text, whether it is positive, negative, or neutral. By analyzing and understanding the sentiment behind textual data, sentiment analysis enables businesses and organizations to gain valuable insights into public opinions, customer satisfaction, brand reputation, and market trends.

One of the key applications of sentiment analysis is in the field of customer feedback and reviews. With the increasing popularity of online platforms and social media, customers are sharing their opinions and experiences more openly than ever before. Sentiment analysis provides a powerful tool for businesses to monitor and analyze these customer sentiments at scale. By automatically processing and categorizing large volumes of customer feedback, sentiment analysis allows companies to gauge customer satisfaction levels, identify areas for improvement, and respond to customer concerns in a timely manner. This valuable feedback can drive product enhancements, customer service improvements, and overall business strategies to align with customer preferences and needs.

The paper titled "Sentiment Analysis on IMDB Movie Reviews using Machine Learning and Deep Learning Algorithms" [1]addresses the task of sentiment analysis in the context of movie reviews. It explores the application of various machine learning and deep learning algorithms to accurately classify the sentiments expressed in movie reviews. The authors compare the performance of algorithms such as logistic regression, support vector machines (SVM), and long short-term memory (LSTM) networks.

By conducting experiments and evaluating the performance of the algorithms on the IMDB movie reviews dataset, the authors demonstrate the effectiveness and potential of machine learning and deep learning approaches for sentiment analysis in the movie industry. The results of the study can provide valuable insights for content creators, distributors, and decision-makers in the movie industry, enabling them to improve their products, tailor marketing campaigns, and understand customer sentiments better.

Overall, the paper contributes to the field of sentiment analysis by presenting a comprehensive comparison of machine learning and deep learning algorithms specifically applied to the task of sentiment analysis on movie reviews. It highlights the importance of understanding customer sentiments in the movie industry and showcases the potential of machine learning and deep learning techniques for extracting valuable insights from textual data.The following papers worked on similar aspects:

1)Classification of sentiment reviews using n-gram machine learning approach[2]: In this paper, the authors propose a machine learning approach for sentiment classification of reviews. They specifically use an n-gram-based approach, which involves representing the text using sequences of n words. N-grams are a commonly used technique in NLP to capture the local context and dependencies in the text.

2)Thumbs up? sentiment classification using machine learning techniques[3]: In this paper they classified the paper based on sentiment rather than topic. They found out machine learning techniques outsmart human baselines

3)Sentiment Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis[4]:

This paper contributes to the sentiment analysis for customers’ review classification which is helpful to analyze the information in the form of the number of tweets where opinions are highly unstructured and are either positive or negative, or somewhere in between of these two.

4)Linguistically regularized LSTMs for sentiment classification[5]:

Sentiment understanding has been a long-term goal of AI in the past decades. This paper  
deals with sentence-level sentiment classification

# 5)Sentiment Analysis Based on Attention Mechanisms and Bi-Directional LSTM Fusion Model[6]

 Based on the researches on sentiment analysis and deep learning, they propose a hybrid framework AM-Bi-LSTM that combines Attention Mechanism and Bi-directional Long-Short-Term Memory (Bi-LSTM) neural networks for sentence classification.

# 6) Sentiment Analysis By Using Modified RNN And A Tree LSTM[7]

In this work, they provide a brief overview of deep learning classification algorithms and also conclude that Tree-LSTM gives state-of-the-art accuracy for fine-grained sentiment analysis.

## [**Combining Recurrent and Convolutional Neural Networks for Relation Classification**](https://aclanthology.org/N16-1065.pdf)

**Project Objectives**

Here are some of the objectives of image captioning project:

• To use neural network models to perform sentiment analaysis

• To train the system using a sizable collection of texts and the sentiments that are assigned to them.

• To make the model better at understanding the connection between the text and sentiment.

• To compare different models using suitable metrics (such as precision, accuracy, f1score, recall)

• To build a web application which can input text and show the sentiment of the text and run it in local computer

**Analysis**

**Deep Learning Models**

I have used 3 deep learning models in this project

**1)Convolutional Neural Network(CNN)**

CNN, or Convolutional Neural Network, is a deep learning architecture specifically designed for processing grid-like data, such as images or time-series data. CNNs are particularly effective in capturing local patterns and spatial relationships within the input data.

The key component of a CNN is the convolutional layer. It applies convolutional filters to the input data, which are small windows that scan across the data and perform element-wise multiplication and summation operations. This allows the network to extract local features and patterns from the input.CNNs often include pooling layers, such as max pooling or average pooling, which downsample the feature maps generated by the convolutional layers. Pooling reduces the spatial dimensionality while preserving the most important information, aiding in translation invariance and reducing computational complexity.

CNNs also typically incorporate activation functions, such as ReLU (Rectified Linear Unit), to introduce non-linearity into the model, allowing it to learn complex relationships between the input data and target outputs.The convolutional layers of a CNN are followed by fully connected layers, which connect every neuron from the previous layer to the subsequent layer. These fully connected layers aggregate the learned features and make predictions based on the extracted information.

CNNs have been widely successful in various computer vision tasks, including image classification, object detection, and image segmentation. They have revolutionized the field by achieving state-of-the-art performance on benchmark datasets. CNNs can be helpful for sentiment analysis by extracting meaningful features from text data that can effectively capture sentiment-related patterns and dependencies. Although CNNs are widely used in computer vision tasks, they can also be adapted for text classification tasks like sentiment analysis.

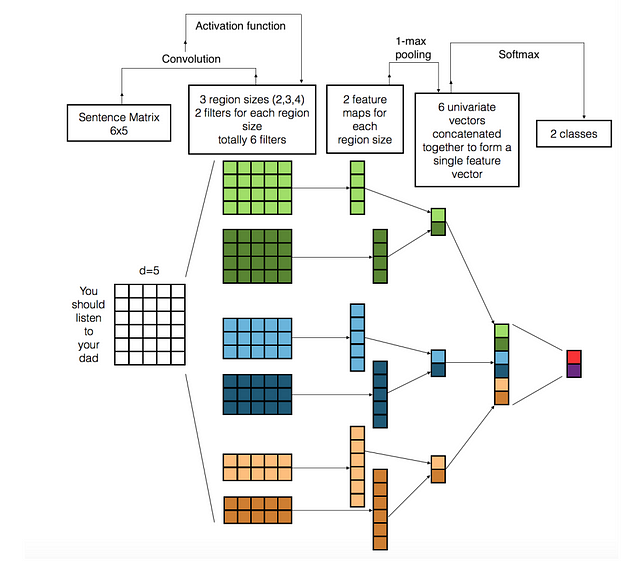


Fig:How CNN works[8]

**2)LSTM**

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is well-suited for processing sequential data, such as text, speech, or time-series data. LSTMs are designed to overcome the limitations of traditional RNNs in capturing long-term dependencies by incorporating memory cells and gating mechanisms.

The main innovation of LSTMs lies in their ability to selectively retain and forget information over extended sequences. LSTMs achieve this by utilizing three gating mechanisms: the input gate, forget gate, and output gate. These gates regulate the flow of information through the network, allowing it to selectively remember or forget specific information at each time step.

The input gate determines how much new information should be incorporated into the memory cell. It takes into account the current input and the previous hidden state to compute the update to the memory cell. The forget gate controls the extent to which the previous memory cell content should be retained or discarded. It considers the current input and the previous hidden state to compute the forget gate activation. Finally, the output gate determines how much of the memory cell content should be exposed as the output of the LSTM cell. It combines the input and the previous hidden state to compute the output gate activation.

By incorporating these gating mechanisms, LSTMs can effectively capture long-range dependencies and mitigate the vanishing or exploding gradient problem that can occur in traditional RNNs. This makes LSTMs particularly suitable for tasks that require understanding and generating sequences of data.

In the context of sentiment analysis, LSTMs can be applied to capture the contextual information and dependencies within a sentence or text. By modeling the sequential nature of the text data, LSTMs can learn to extract relevant features and representations that capture sentiment-related patterns over variable-length sequences.

LSTMs have demonstrated impressive performance in various natural language processing tasks, including sentiment analysis, machine translation, and text generation. They have the ability to capture the nuanced semantic relationships in text data, allowing them to effectively model sentiment and generate coherent and contextually appropriate responses.

**3)DistilBERT**

DistilBERT is a state-of-the-art transformer-based model for natural language processing tasks, including text classification. It is a distilled version of the original BERT (Bidirectional Encoder Representations from Transformers) model, designed to be smaller and faster while still maintaining high performance.

DistilBERT leverages the transformer architecture, which is a self-attention mechanism that allows the model to capture the relationships between words in a sentence. The model consists of multiple transformer layers, each composed of self-attention and feed-forward neural networks. The self-attention mechanism enables the model to attend to different words and their contextual information, allowing it to capture the dependencies and semantic relationships within the text.

DistilBERT specifically focuses on the task of sequence classification, where the goal is to classify a given sequence of text into predefined categories or labels. It utilizes a pretraining and fine-tuning approach. During pretraining, the model is trained on a large corpus of text using unsupervised learning, which allows it to learn general language representations. Fine-tuning involves training the model on task-specific labeled data, such as sentiment analysis datasets, to adapt it to the specific classification task.

One of the key advantages of DistilBERT is its smaller size and faster inference speed compared to the original BERT model. This makes it more practical for deployment in resource-constrained environments. Despite its smaller size, DistilBERT still achieves competitive performance on various natural language processing tasks, including sentiment analysis.

For sentiment analysis, DistilBERT can effectively learn contextual representations of text, capturing sentiment-related patterns and semantic dependencies. It can understand the sentiment expressed in a sentence by considering the surrounding words and their interactions. By fine-tuning DistilBERT on sentiment analysis datasets, it can learn to make accurate predictions and classify text into positive, negative, or neutral sentiment categories.

DistilBERT has been widely adopted and has demonstrated impressive results on various text classification benchmarks. Its ability to distill the knowledge from the original BERT model into a more compact and efficient form makes it a powerful tool for sentiment analysis and other text classification tasks.

**Machine Learning Models**

The Machine Learning Models which are used in the project are

1. Logistic Regression: Logistic regression is a popular supervised learning algorithm used for binary classification tasks. It models the relationship between the dependent variable and independent variables by estimating the probabilities using a logistic function. With its simplicity and interpretability, logistic regression is often used when the goal is to understand the influence of input features on the outcome and make probabilistic predictions.
2. Support Vector Machine (SVM): Support Vector Machine is a powerful machine learning algorithm used for both classification and regression tasks. It constructs a hyperplane or a set of hyperplanes to separate different classes by maximizing the margin between them. SVM is effective in handling high-dimensional data and can handle both linearly separable and non-linearly separable datasets by using various kernel functions.
3. Naive Bayes: Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence between features. Despite its simplicity, Naive Bayes is widely used for text classification and spam filtering tasks. It works well with high-dimensional data and provides fast training and prediction speed, making it a popular choice for large-scale applications.
4. XGBoost: XGBoost stands for Extreme Gradient Boosting and is an ensemble learning algorithm known for its exceptional performance and versatility. It uses a combination of decision trees as base learners and gradient boosting techniques to iteratively improve the model's predictions. XGBoost excels in handling complex datasets, capturing non-linear relationships, and dealing with missing values, making it a go-to algorithm for a wide range of machine learning problems.
5. Random Forest: Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. It builds a forest of trees, where each tree is trained on a different subset of the data and features. Random Forest mitigates overfitting, handles high-dimensional data, and provides estimates of feature importance. It is robust to outliers and noisy data and can handle both classification and regression tasks effectively.
6. Decision Tree: Decision trees are versatile machine learning models that make predictions by partitioning the feature space into segments based on a set of rules. Each internal node of the tree represents a decision based on a feature, and each leaf node represents a class or a regression value. Decision trees are interpretable, handle both categorical and numerical features, and are robust to outliers and missing data. They are widely used for classification, regression, and feature selection tasks.

**Text Vectorization Techniques**

**1)TF-IDF**

TF-IDF is a numerical statistic that reflects the importance of a word in a document within a collection or corpus of documents. It calculates the product of two factors: Term Frequency (TF), which measures how frequently a term appears in a document, and Inverse Document Frequency (IDF), which measures the rarity of a term across the entire document collection. TF-IDF assigns higher weights to terms that are more specific to a particular document, making it useful for tasks such as text classification, information retrieval, and keyword extraction.

**2)CountVectorizer**

CountVectorizer is a simple and commonly used technique to convert text documents into a numeric representation called a "bag of words." It counts the occurrence of each word or token in a document and creates a sparse matrix where each row corresponds to a document and each column represents a unique word in the entire corpus. CountVectorizer ignores the order of the words but retains information about their frequency. It is a popular choice for tasks such as text classification, clustering, and building language models.

Both TF-IDF and CountVectorizer are widely employed in NLP to transform raw text data into numerical representations that machine learning algorithms can process. These techniques play a crucial role in extracting meaningful features from text, enabling efficient analysis and modeling of textual data.

**Datasets**

I have used two datasets for this project. One of the dataset is imdb movie review dataset and another one is rotten tomatoes dataset.

**1)IMDB Movie Review Dataset:**

The IMDb movie review dataset is a well-known dataset widely used in sentiment analysis research. It contains a large collection of movie reviews from the IMDb website. The dataset is labeled with binary sentiment labels, indicating whether a review is positive or negative. The reviews cover a wide range of movie genres, providing a diverse set of opinions and sentiments. This dataset is often used to train and evaluate machine learning models for sentiment analysis and text classification tasks. Researchers and practitioners use the IMDb dataset to develop and test algorithms that can accurately classify movie reviews based on their sentiment.

**2)Rotten Tomatoes Dataset:**

The Rotten Tomatoes dataset is another popular dataset used for sentiment analysis and movie review classification. It consists of movie reviews and their associated binary sentiment labels (positive or negative). The reviews in this dataset are sourced from the Rotten Tomatoes website, which aggregates movie reviews from critics and audiences. The Rotten Tomatoes dataset provides a different perspective compared to IMDb, as it includes reviews from professional critics rather than solely user-generated reviews. This dataset is valuable for sentiment analysis tasks focused on understanding critical opinions and evaluating the reception of movies from professional reviewers. In this dataset we are only considering review\_content which contains the original review and review\_type which tells us if it is fresh or rotten.

**Project Approach**

**1)Machine Learning Models**

First we loaded the dataset in the form of pandas dataframe. I performed certain pre processing steps on the text data

**Removal of Punctuation:**

Punctuation refers to the set of symbols and characters used to enhance the structure and clarity of written text. However, in many NLP tasks, punctuation marks do not contribute significantly to the meaning of the text and can introduce noise. Removing punctuation involves eliminating characters such as periods, commas, question marks, and exclamation marks. By doing so, we can reduce the dimensionality of the data and ensure that the focus is on the actual content of the text.

**Removal of Stopwords:**

Stopwords are common words that often occur frequently in a language and carry little to no semantic meaning. Examples of stopwords include "the," "is," "and," and "to." In many NLP applications, stopwords can be safely removed to eliminate noise and reduce computational overhead. Removing stopwords helps to focus on the more informative content words and improve the efficiency of subsequent analyses.

**Lemmatization:**

Lemmatization is the process of reducing words to their base or dictionary form, known as a lemma. It involves transforming words to their canonical or root form, such as converting "running" to "run" or "better" to "good." Lemmatization helps to normalize the text, ensuring that variations of words with the same meaning are treated as a single term. By reducing words to their base forms, lemmatization improves the accuracy and consistency of downstream NLP tasks, such as text classification or information retrieval.

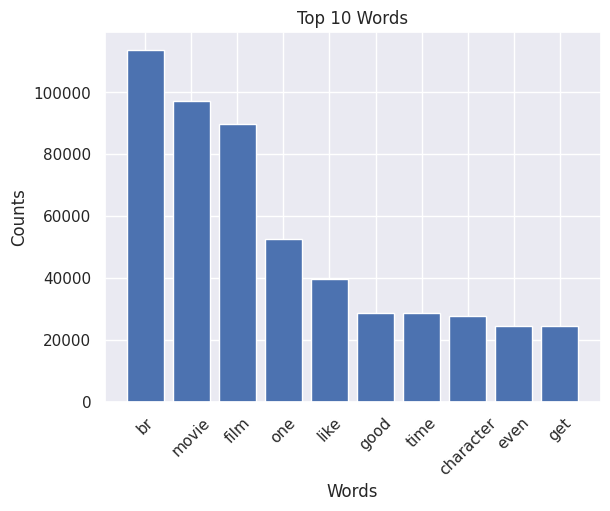
**Stemming:**

Stemming is a simpler technique compared to lemmatization. It involves reducing words to their root form by removing prefixes or suffixes. For example, stemming converts "running" to "run" and "playing" to "play." The goal of stemming is to reduce words to a common base form to capture their essential meaning. While stemming is less sophisticated than lemmatization, it can still be effective in certain cases where linguistic precision is not critical or computational efficiency is a concern.

**Exploratory Data Analaysis**

After the cleaning of data, I have performed exploratory data analysis. I have generated wordcloud of all the reviews, wordcloud for reviews which belonged to positive sentiment and wordcloud for reviews which belonged to negative sentiment. I have also found the top 10 words in similar fashion





**Vectorizing text data**

Using tf-idf and count vectorizer we have vectorized the text data. This vectorized data was used as input for machine learning models (Logistic Regression, Naïve Bayes, SVM, XGBoost, decision trees and Random Forest). The accuracy , f1score, recall and precision of all these models are compared in a graph.

From the above graphs I have observed that svm model which was trained on tfidf vector data works well when compared with all other models. I saved that svm model in a pkl format.

**2)Deep Learning Models**

I have used CNN, LSTM and DistilBERT for performing sentiment analysis on tweets.

**1)CNN**

First, the necessary libraries and modules are imported, including pandas for data manipulation, numpy for numerical computations, and various components from the TensorFlow and scikit-learn libraries for building and evaluating the model.

The IMDB dataset is then loaded from a CSV file using pd.read\_csv(). The sentiment labels are mapped to 0 (negative) and 1 (positive) to create a binary classification task.

The dataset is split into training and testing sets using train\_test\_split() from scikit-learn, with a test size of 20% and a random state of 42.

Next, the text data is tokenized using the Tokenizer class from TensorFlow. The tokenizer is fit on the training reviews to learn the vocabulary and convert the text into sequences of integers.

The sequences are then padded to ensure they have the same length using pad\_sequences() from TensorFlow. The maximum length of the sequences is determined by the longest review in the training set.

The labels are converted to numpy arrays, and the CNN model architecture is defined using the Sequential API from Keras. It consists of an embedding layer, a 1D convolutional layer, a global max pooling layer, and two dense layers with dropout for regularization.

The model is compiled with binary cross-entropy loss and the Adam optimizer.

The model is trained on the training set using model.fit() with a specified batch size and number of epochs. The validation data is used to monitor the performance during training.

Finally, the trained model is evaluated on the testing set using model.evaluate(), and the accuracy and other metrics are displayed.

**2)LSTM**

This code demonstrates a text classification task using a Long Short-Term Memory (LSTM) model on the IMDB movie review dataset.

The code begins by importing the required libraries and modules, including pandas for data manipulation, numpy for numerical computations, scikit-learn for model evaluation, and various components from TensorFlow for building and training the LSTM model.

The IMDB dataset is then loaded from a CSV file using pd.read\_csv(). The sentiment labels are mapped to 0 (negative) and 1 (positive) to create a binary classification task.The dataset is split into training and testing sets using train\_test\_split() from scikit-learn, with a test size of 20% and a random state of 42.Next, the text data is tokenized using the Tokenizer class from TensorFlow. The tokenizer is fit on the training reviews to learn the vocabulary and convert the text into sequences of integers.

The sequences are then padded to ensure they have the same length using pad\_sequences() from TensorFlow. The maximum length of the sequences is determined by the longest review in the training set.

The labels are converted to numpy arrays, and the LSTM model architecture is defined using the Sequential API from Keras. It consists of an embedding layer, an LSTM layer for sequential data processing, two dense layers with dropout for regularization, and an output layer with sigmoid activation for binary classification.

The model is compiled with binary cross-entropy loss and the Adam optimizer.The model is trained on the training set using model.fit() with a specified batch size and number of epochs. The validation data is used to monitor the performance during training.After training, the model predicts the sentiment of the testing set using model.predict(). The predicted probabilities are converted to binary labels using a threshold of 0.5.

The accuracy of the model is calculated using accuracy\_score() from scikit-learn, and the classification report is generated using classification\_report().Finally, the accuracy and classification report are printed to evaluate the performance of the LSTM model on the testing set.

**3)DistilBERT**

The IMDB reviews dataset is loaded into a pandas DataFrame. The sentiment column is converted into a binary classification problem, with positive sentiment labeled as 1 and negative sentiment as 0. The data is split into training and testing sets using train\_test\_split().

The DistilBERT tokenizer is loaded using DistilBertTokenizer.from\_pretrained(). The text data is then converted into BERT input format by tokenizing the text, truncating and padding the sequences, using the tokenizer on the training and testing sets.

The DistilBERT model for sequence classification is loaded using DistilBertForSequenceClassification.from\_pretrained(). The model is set to the appropriate device, either CUDA if available or CPU.

The data is converted into tensors and moved to the device. The batch size is set, and data loaders are created using TensorDataset and DataLoader from torch.utils.data.

The optimizer and learning rate are set using torch.optim.AdamW. The training loop begins with the specified number of epochs. For each epoch, the loss is computed and the model parameters are updated using backpropagation and the AdamW optimizer. The average loss is calculated for each epoch.

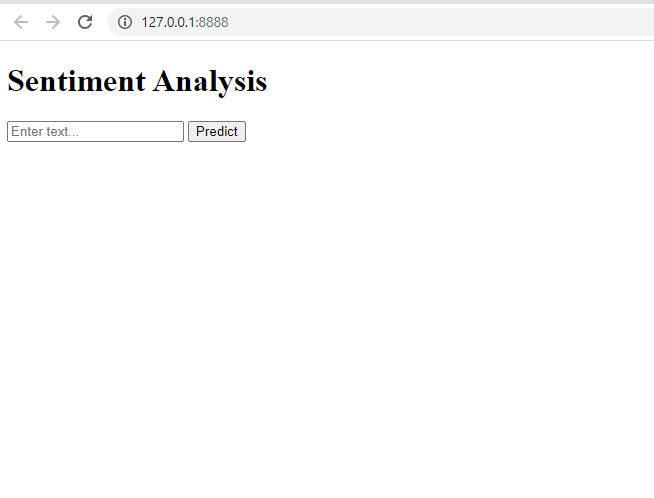
After training, the model is switched to evaluation mode. Predictions are made on the test set using the trained model, and the probabilities are converted to predicted labels. The metrics, including accuracy, precision, recall, and F1 score, are computed using scikit-learn's accuracy\_score() and precision\_recall\_fscore\_support() functions.

Finally, the metrics are printed to evaluate the performance of the DistilBERT model on the sentiment classification task.

The same steps and approaches which are used on imdb dataset are again used on Rotten tomatoes datatset.

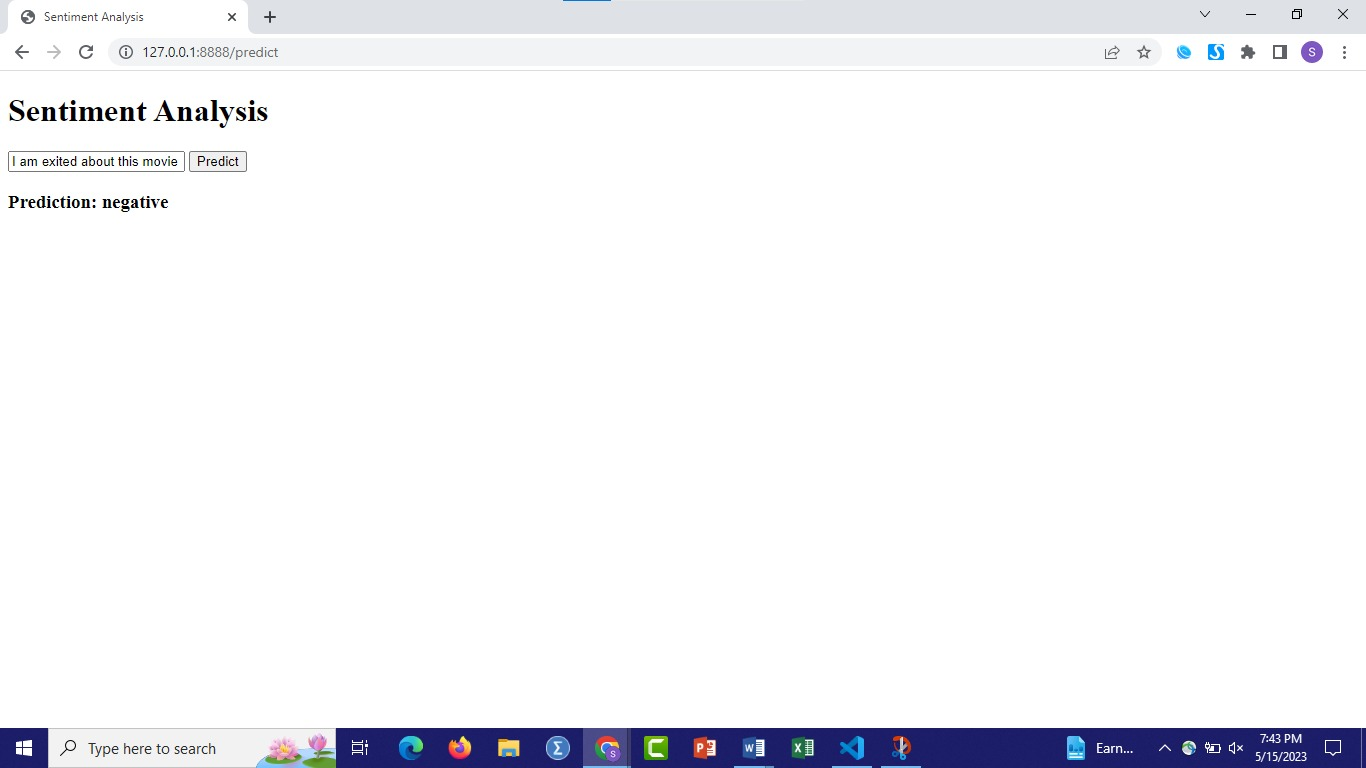
**Flask app**

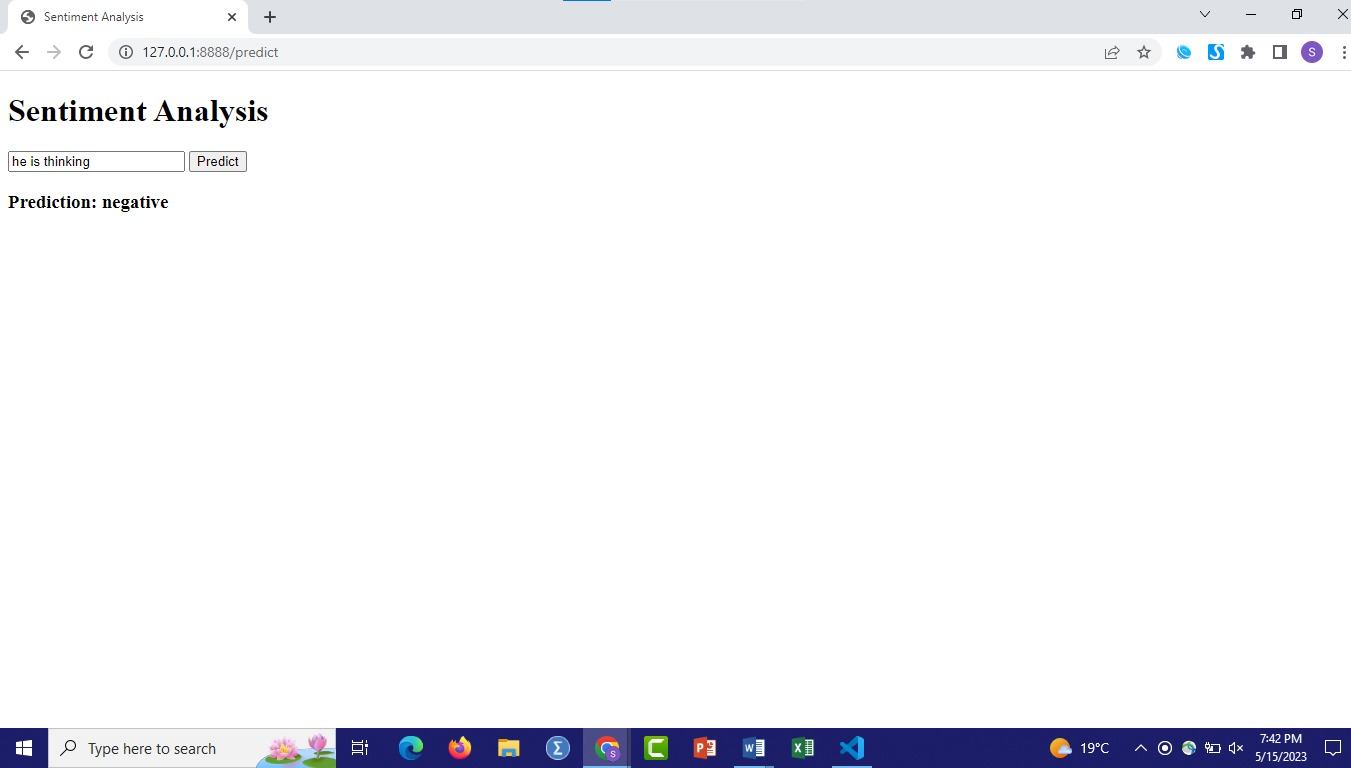
I have created a flask web application which uses the saved svm model to predict the sentiment of the text which is entered in the textbox when the predict button is pressed. I have also used the tfidf pkl file in the flask app to vectorize the text .

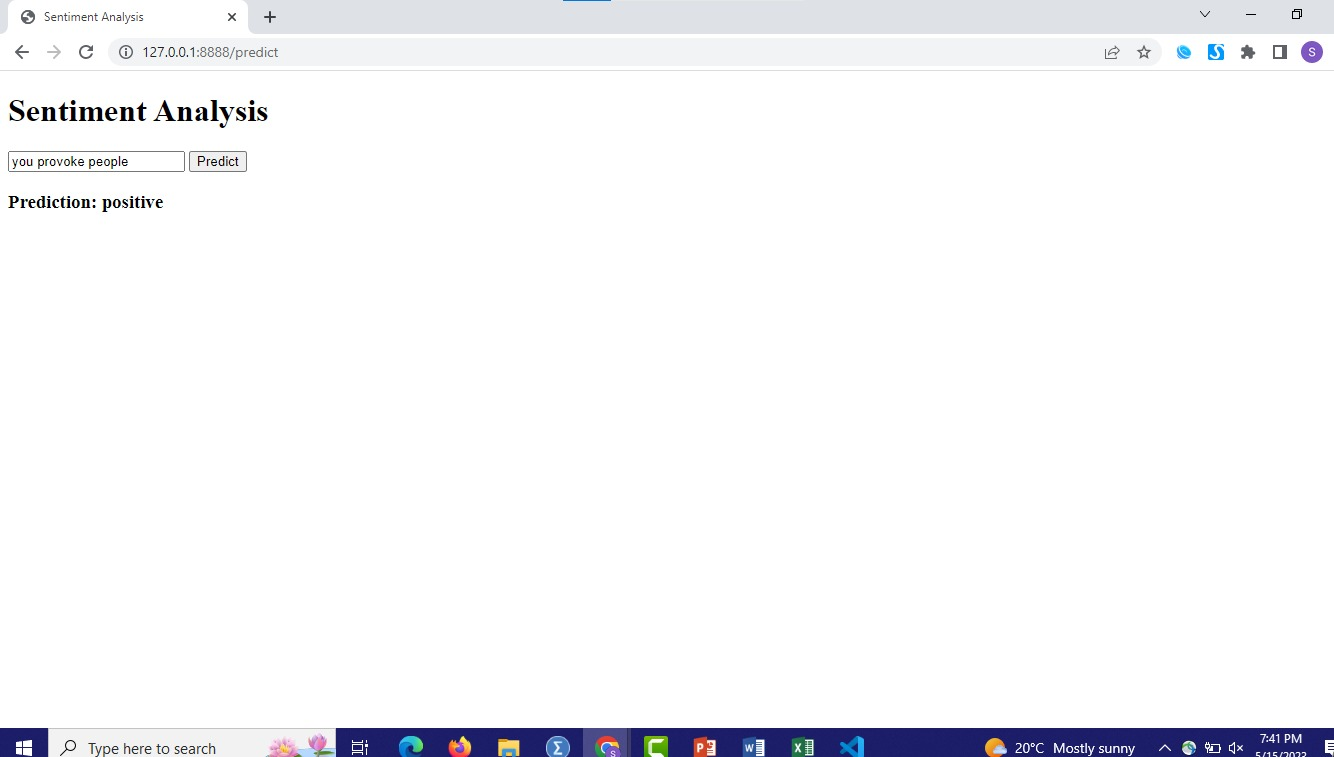
****

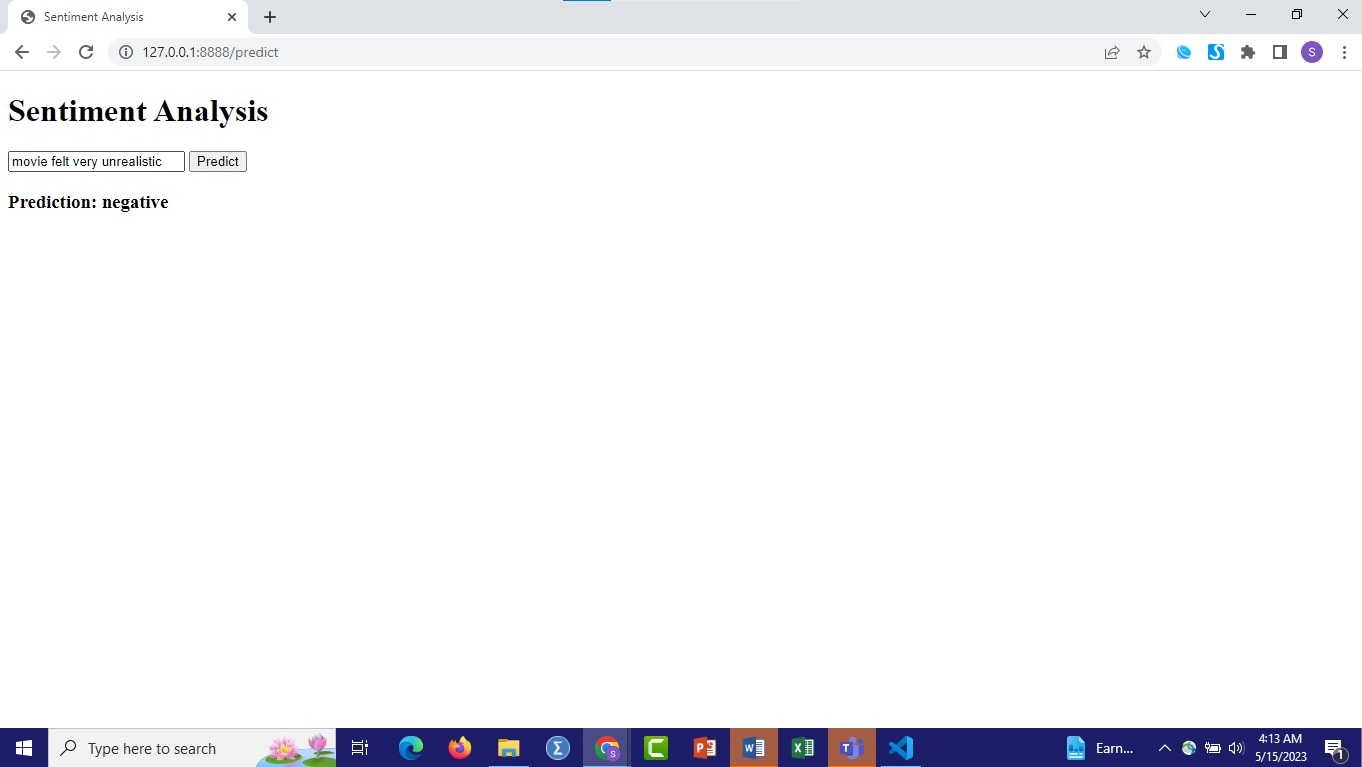
**Results**

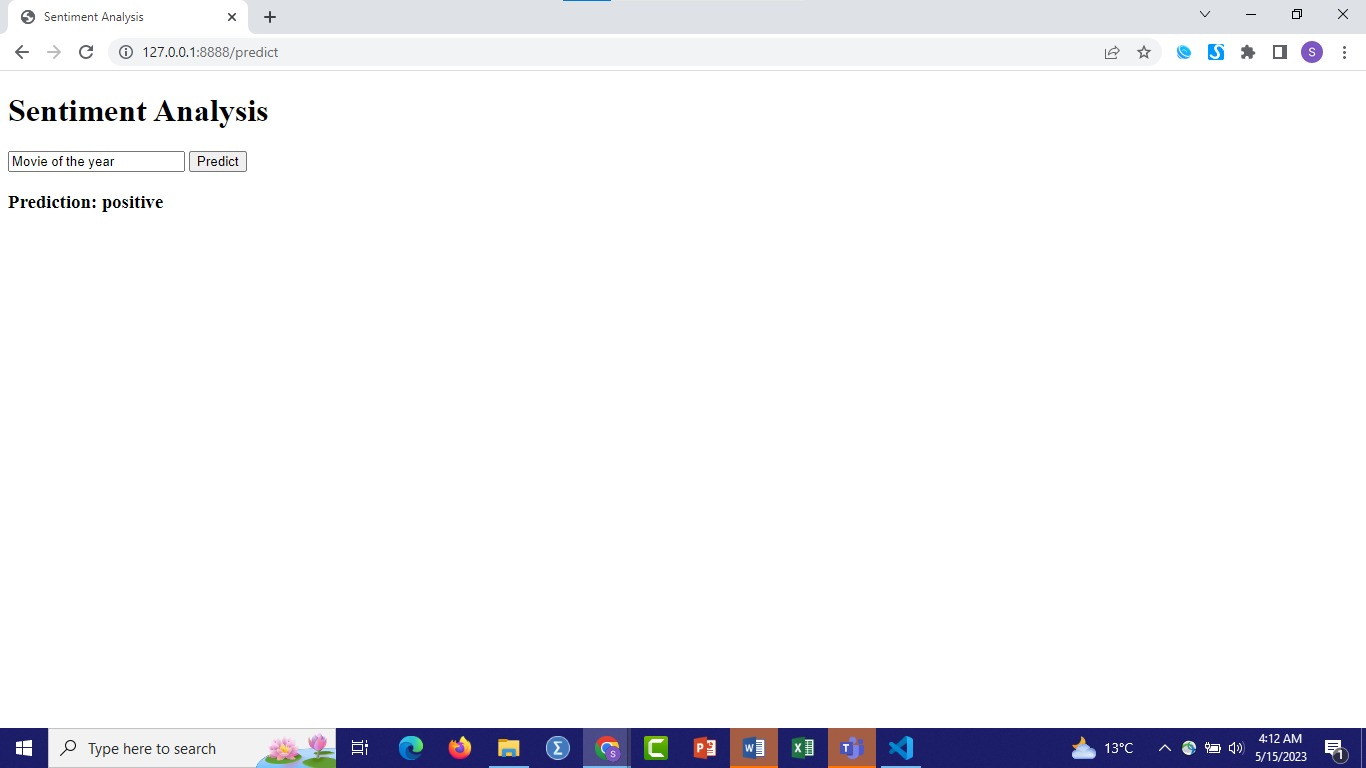
The following images have predictions from the flask web application which is using an svm and tfidf pkl to predict the sentiment of the text entered in the textbox.





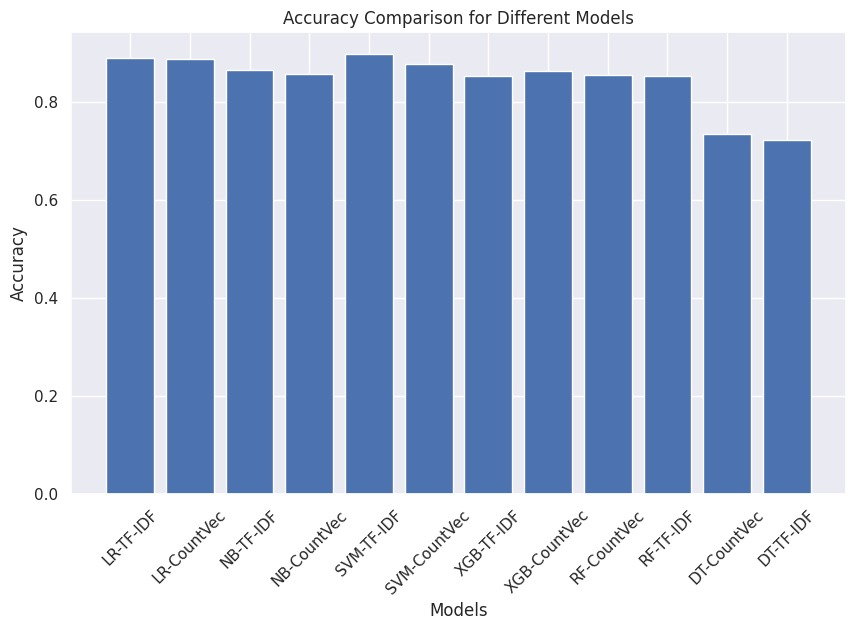




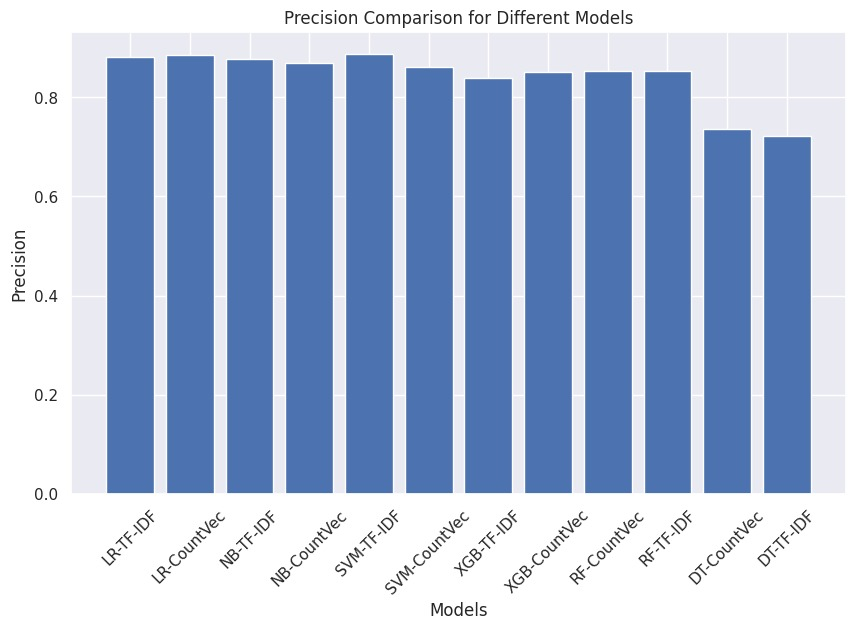


**Comparison of Machine Learning Models(IMDB Dataset\_)**

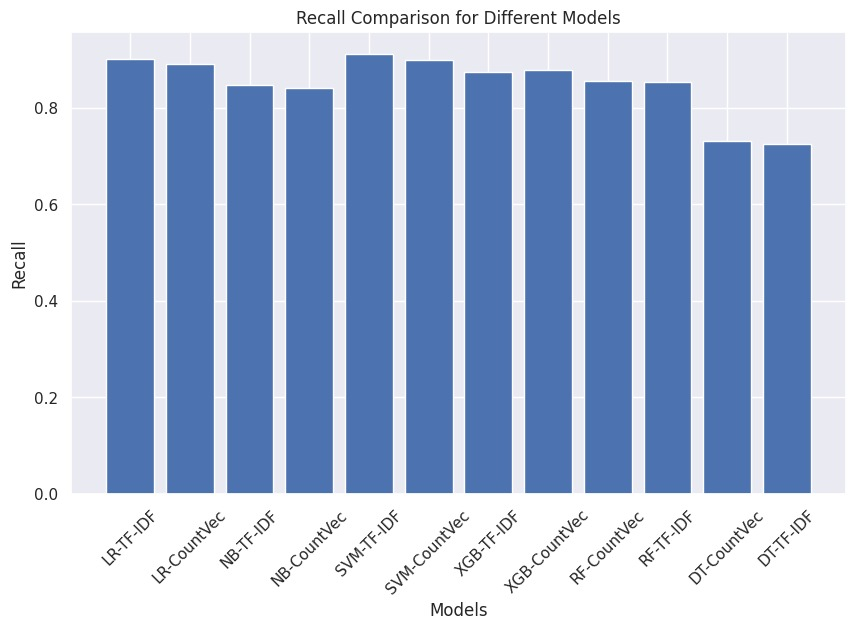
**1)Accuracy**

****

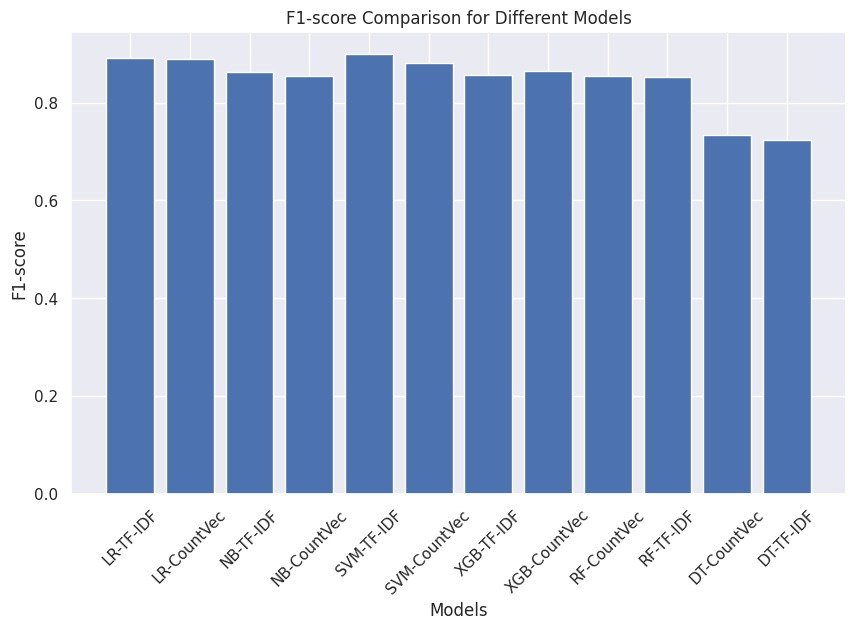
**2)Precision**



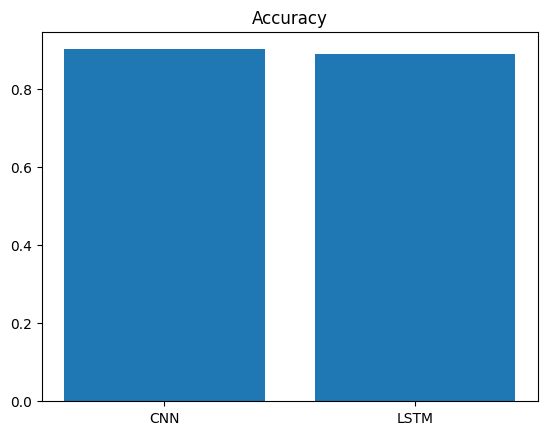
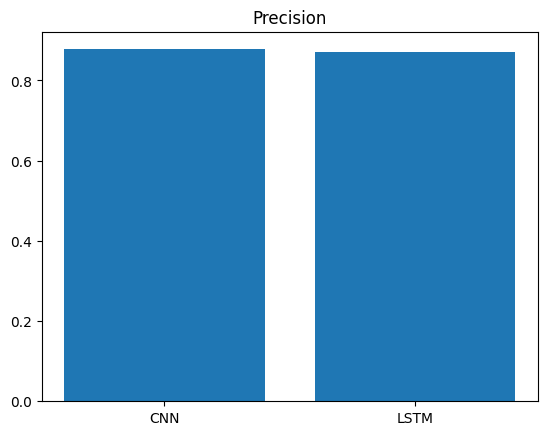
**3)Recall**

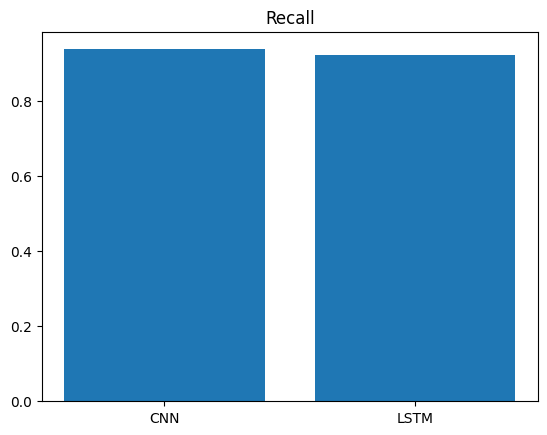
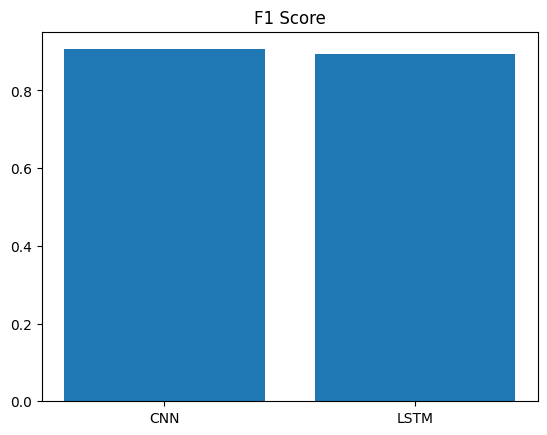


**4)F1 score**

****

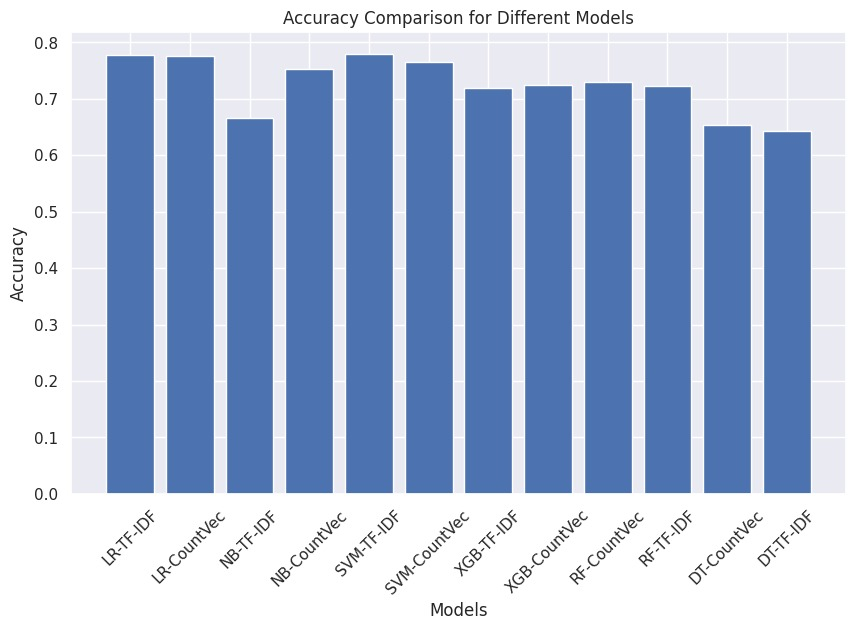
**Comparision of CNN vs LSTM (IMDB Dataset)**

** **

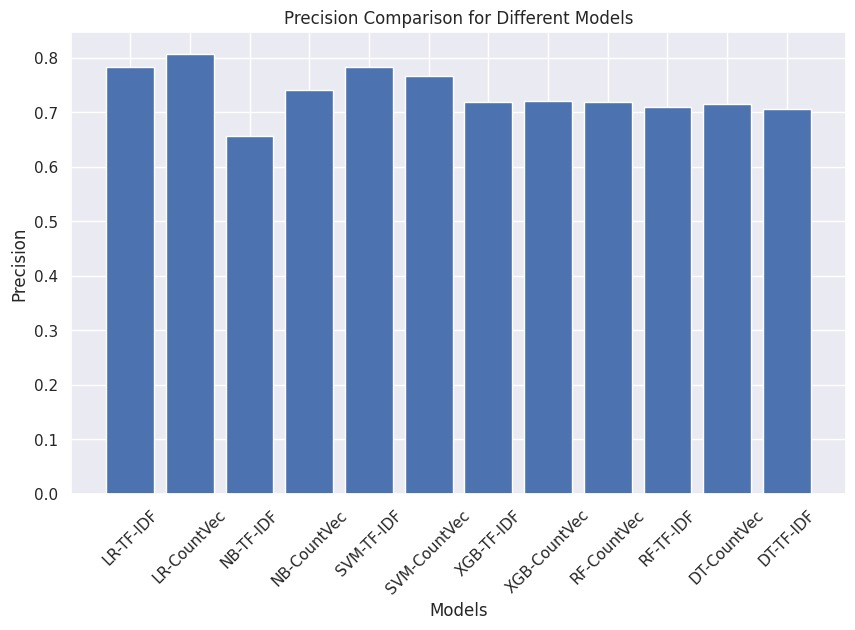
** **

**Comparison of Machine Learning (Rotten Tomatoes)**

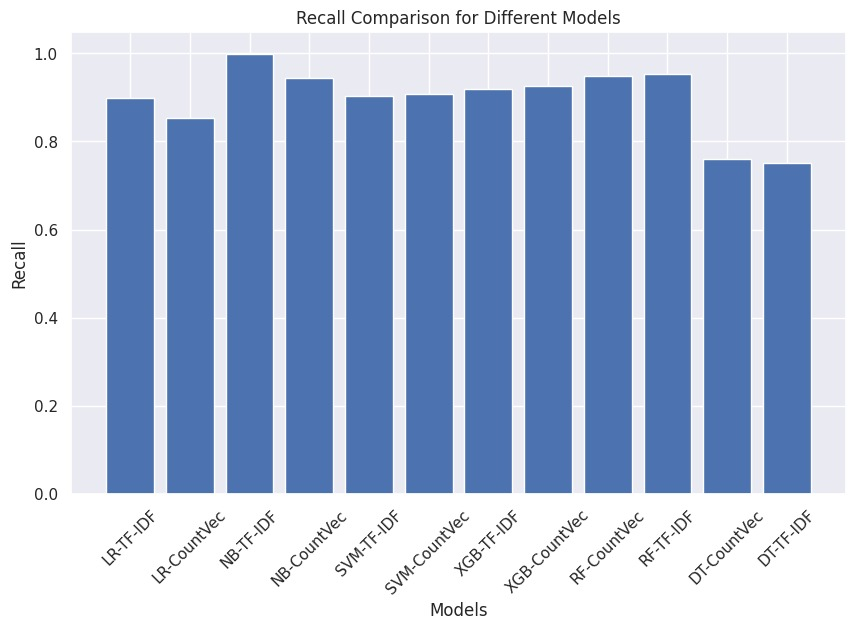
**1)Accuracy**

****

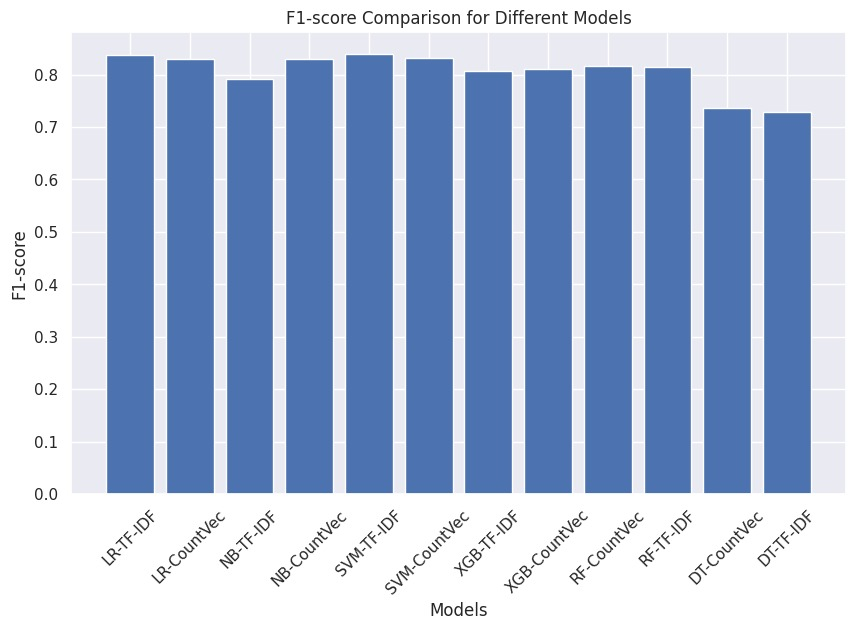
**2)Precision**

****

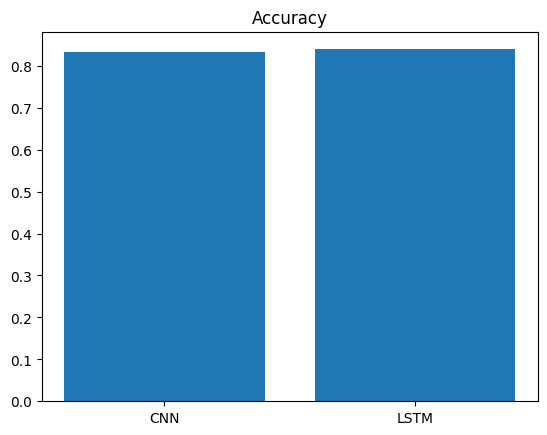
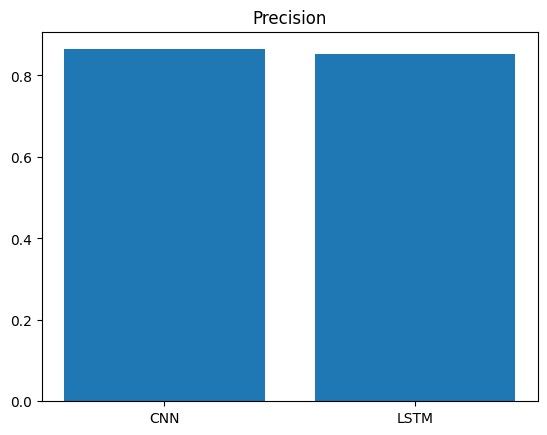
**3)Recall**

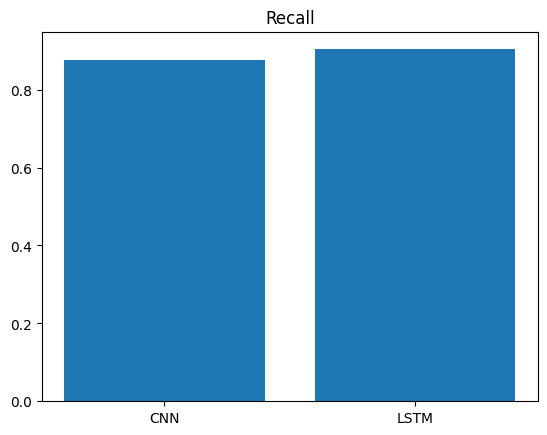
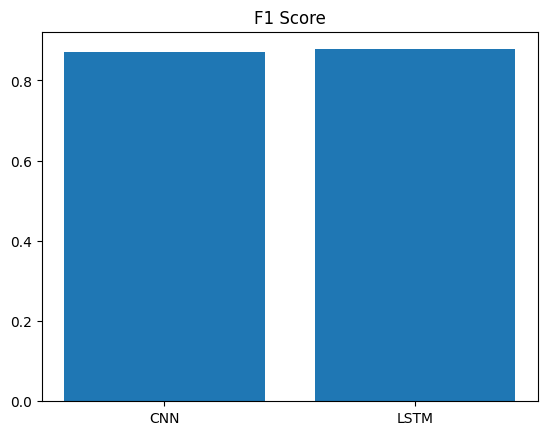
****

**4)f1 Score**

****

**Comparision of CNN vs LSTM (rotten tomatoes Dataset)**

** **

** **

**Discussion**

One thing which I have observed is the same models and vectorization techniques yield better results on imdb dataset when compared with rotten tomatoes dataset. My analysis on this is that the reviews in rotten tomatoes dataset is a bit complex in language when compared with that of the reviews of imdb.

SVM model with tfidf works a lot better when compared with other models in terms of accuracy. I think most of the models were close to each other in their evaluation metrics. Most of the simple models worked well on text data and they were able to make predictions sooner than the deep learning models.

In deep learning models CNN performed better than LSTM in both the datasets. But the problem with deep learning models is they consumed a lot of time and they were also requiring a large computing resources.

As it is seen that certain sentences were labeled incorrectly , I am excited about the movie was labelled as negative where as the movie was exciting was positive. This shows us that the models need to be trained on a very large dataset and even though the accuracies are shown as higher they aren’t that good as they are trained on a relatively small data.

The sentence he is thinking is also wrongly labeled as negative sentiment, this could be very dangerous because if a model like this comes out as a product then decisions taken based on its predictions are going to be very harmful.

**Evaluation**

The following are the evaluation metrics which are used in this project

Accuracy is a commonly used evaluation metric that measures the overall correctness of a model's predictions. It calculates the ratio of the number of correct predictions to the total number of predictions. Accuracy is a straightforward metric but can be misleading when dealing with imbalanced datasets, where the number of instances in different classes is significantly different.

Precision is a metric that focuses on the positive predictions made by a model. It measures the proportion of true positive predictions out of all positive predictions made by the model. Precision is useful when the cost of false positives is high, and we want to minimize the occurrence of false alarms or incorrect positive predictions.

Recall, also known as sensitivity or true positive rate, is a metric that measures the proportion of true positives that were correctly identified by the model out of all actual positive instances. Recall is valuable when the cost of false negatives is high, and we want to minimize the occurrence of missed positive predictions.

F1 score is a balanced metric that combines precision and recall into a single value. It is the harmonic mean of precision and recall and provides a comprehensive evaluation of a model's performance. F1 score is particularly useful when there is an uneven distribution between the classes or when we want to find a balance between precision and recall.

In summary, accuracy provides an overall measure of correctness, precision focuses on true positive predictions, recall emphasizes the detection of positive instances, and F1 score combines precision and recall to provide a balanced evaluation of a model's performance. Each of these metrics serves a specific purpose and helps assess different aspects of a model's effectiveness in various scenarios.

**Reflections**

In conclusion, comparing machine learning models to deep learning models reveals some trade-offs in terms of performance, time, and resource requirements. Machine learning models, such as logistic regression or random forest, often exhibit competitive performance in various tasks while requiring less computational resources and training time compared to deep learning models.

Proper care must be taken while collecting the data as the data shouldn’t be biased and we should make sure that it is anonymous data. Any model which is going to be launched should be properly checked to see if it is well trained on a large dataset and has very high values of accuracy as these models affect businesses in real time.

**References**

[1] K. Amulya, S. B. Swathi, P. Kamakshi and Y. Bhavani, "Sentiment Analysis on IMDB Movie Reviews using Machine Learning and Deep Learning Algorithms," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 814-819, doi: 10.1109/ICSSIT53264.2022.9716550.

[2]A. Tripathy, A. Agrawal, and S.K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," Expert Systems with Applications, vol. 57, pp. 117-126, 2016.

[3]B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," Proceedings of the ACL-02 conference on Empirical methods in natural language processing, vol. 10, pp. 79-86, 2002.

[4]M. Gupta, A. Kumar, and P. Kumar, "Sentiment Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis," International Journal of Computer Science and Mobile Computing, vol. 3, no. 2, pp. 476-481, 2014.

[5]Y. Tang, F. Qi, X. Dong, X. Huang, and X. Zhu, "Linguistically regularized LSTMs for sentiment classification," in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol. 2, pp. 165-169, 2016.

[6]Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Sentiment Analysis Based on Attention Mechanisms and Bi-Directional LSTM Fusion Model," in Proceedings of the 31st AAAI Conference on Artificial Intelligence, pp. 3358-3364, 2017.

[7]R. M. Al-Hajj, Z. J. Raad, and G. M. Kamel, "Sentiment Analysis By Using Modified RNN And A Tree LSTM," International Journal of Computer Science and Information Security, vol. 16, no. 2, pp. 66-72, 2018.

[8] Moreno Lopez, M., and Kalita, J. (2017, June). Deep learning applied to natural language processing. arXiv preprint arXiv:1706.03762.