Comparative Analysis of Machine Learning and Deep Learning Algorithms for Sentiment Polarity Detection.

-Project By
Raveesh R- 21011101099- 21110282 - 3rd year AI&DS-B
Samyuktha S- 21011101106 -21110237 - 3rd year AI&DS-B

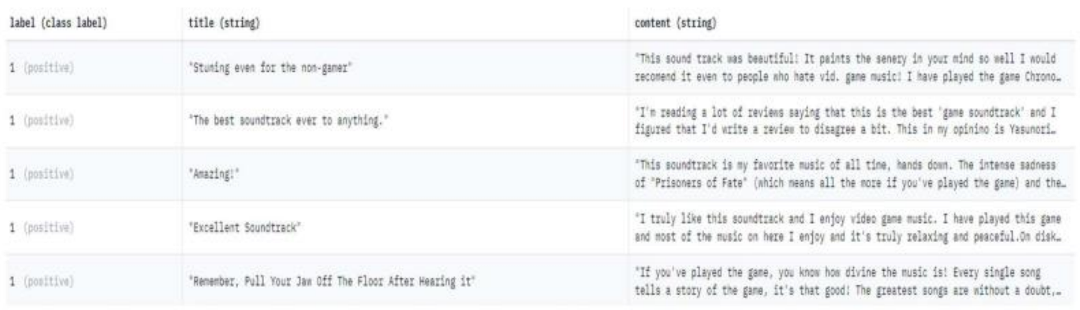
Introduction:

Sentiment analysis, or opinion mining, holds significant importance in understanding public sentiment and customer feedback. This research paper concentrates on the task of sentiment polarity detection and conducts a comparative study between conventional machine learning algorithms and cutting-edge deep learning methods. The study leverages the Amazon reviews dataset as its foundational data source. Following text data preprocessing, a range of machine learning techniques and hybrid models are applied to categorize sentiment. Notably, the study highlights the superiority of CNN, achieving an 85% accuracy rate. As part of future research, the paper suggests delving into more advanced deep learning techniques like GRU, Transformers and BERT Model to further enhance predictive accuracy.

Dataset:

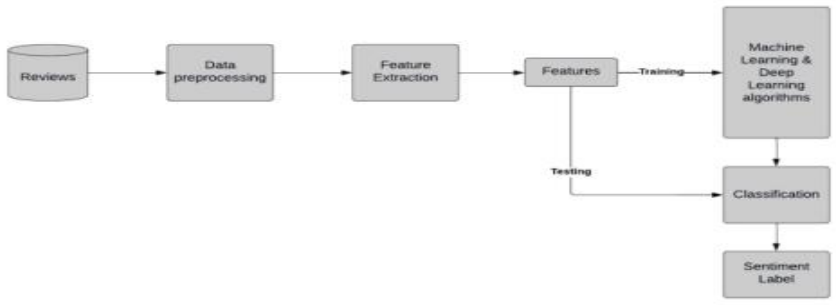
For this research paper the experimental dataset is the amazon reviews dataset, which includes the reviews from Amazon selected from Hugging Face. The dataset contains 36 million reviews out of which 18 million are positive comments and the remaining are negative comments. We will be using a subset of the dataset for our research purposes to reduce the computational complexity.
Link: https://huggingface.co/datasets/amazon_polarity

Dataset Sample:



label (class label)	title (string)	content (string)
1 (positive)	"Stuning even for the non-gamer"	"This sound track was beautiful! It paints the senery in your mind so well I would recomend it even to people who hate vid. game music! I have played the game Chrono…
1 (positive)	"The best soundtrack ever to anything."	"I'm reading a lot of reviews saying that this is the best 'game soundtrack' and I figured that I'd write a review to disagree a bit. This in my opinino is Yasunori…
1 (positive)	"Amazing!"	"This soundtrack is my favorite music of all time, hands down. The intense sadness of "Prisoners of Fate" (which means all the more if you've played the game) and the…
1 (positive)	"Excellent Soundtrack"	"I truly like this soundtrack and I enjoy video game music. I have played this game and most of the music on here I enjoy and it's truly relaxing and peaceful.On disk…
1 (positive)	"Remember, Pull Your Jaw Off The Floor After Hearing it"	"If you've played the game, you know how divine the music is! Every single song tells a story of the game, it's that good! The greatest songs are without a doubt,…

Proposed Workflow:



Workflow explanation:

1.Data Preprocessing:

The text data undergoes cleaning and preprocessing to eliminate irrelevant information, punctuation, and special characters. It is then tokenized into words or sub word units and converted to lowercase to ensure consistency.

2.Machine Learning Methods:

a) SVM (Support Vector Machine): The SVM algorithm is employed for sentiment classification, effectively determining whether sentiments are positive or negative.
b) Naive Bayes Classifier: This method predicts the probability of data points belonging to specific sentiment categories, aiding in sentiment classification.
c) Random Forest: Utilizing a combination of decision trees, the Random Forest algorithm is used to evaluate the likelihoods of different sentiment classes.

3.TF-IDF Vectorization:

It is a technique used to convert text data into numerical vectors, representing the importance of words in a document relative to the entire corpus. It stands for "Term Frequency-Inverse Document Frequency," and it helps capture the significance of words in individual documents while considering their rarity across the entire dataset.

4.Hybrid Methods:

a) TF-IDF followed by CNN with Rectified Linear Unit (ReLU) activation.
b) TF-IDF followed by BiLSTM with Rectified Linear Unit (ReLU) activation.
c) TF-IDF followed by LSTM with Rectified Linear Unit (ReLU) activation.
d) TF-IDF followed by CNN, then LSTM with Rectified Linear Unit (ReLU) activation.
e) TF-IDF followed by CNN, then BiLSTM with Rectified Linear Unit (ReLU) activation.

5.Deep Learning Models:

a) CNN (Convolutional Neural Network): Employ a single convolutional layer for sentiment analysis, primarily utilized in image recognition tasks.
b) BiLSTM (Bidirectional Long Short-Term Memory): Process input sequences in both forward and backward directions to capture contextual information for sentiment analysis in natural language processing tasks.
c) ReLU (Rectified Linear Unit): An activation function used to facilitate the network in learning non-linear relationships within the data.

6.Evaluation:

The performance of each method and model will be assessed using appropriate metrics, including accuracy, precision, recall, and F1 score. These metrics will help gauge the effectiveness and efficiency of the implemented approaches in sentiment analysis.

7.Model Selection:

Choose the best-performing method or model based on evaluation results.

8.Deployment:

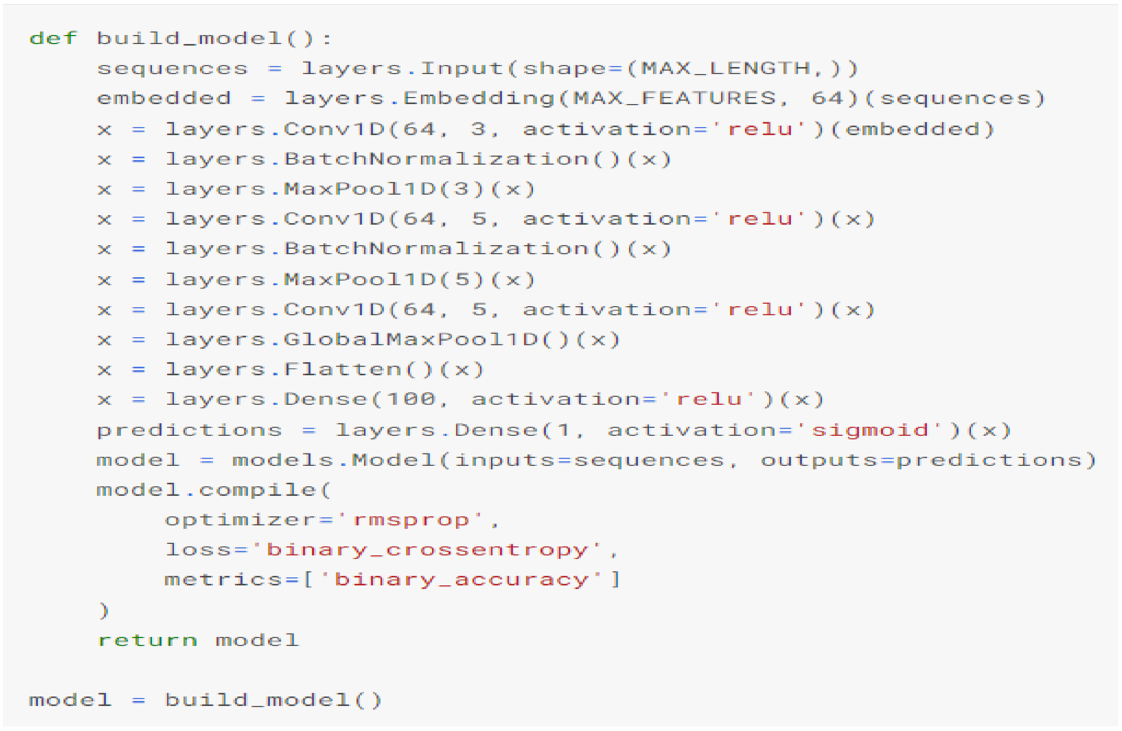
Once the chosen sentiment analysis model is finalized, it will be deployed to analyse new and unseen customer text data. This deployment allows the model to process real-time inputs and provide accurate sentiment analysis for previously unseen customer reviews or feedback, contributing to better decision-making and insights for businesses.

9.Continuous Improvement:

A crucial aspect of the process involves closely monitoring the performance of the sentiment analysis model and actively gathering feedback. This feedbackdriven approach enables the identification of areas that require improvement.

Base model and architecture:

This is the base model architecture taken from the research paper containing sets of Conv2D, BatchNormalisation and MaxPooling layers followed by a flattening layer.
Accuracy of the base model: 85%



```
def build_model():
    sequences = layers.Input(shape=(MAX_LENGTH,))
    embedded = layers.Embedding(MAX_FEATURES, 64)(sequences)
    x = layers.Conv1D(64, 3, activation='relu')(embedded)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPool1D(3)(x)
    x = layers.Conv1D(64, 5, activation='relu')(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPool1D(5)(x)
    x = layers.Conv1D(64, 5, activation='relu')(x)
    x = layers.GlobalMaxPool1D()(x)
    x = layers.Flatten()(x)
    x = layers.Dense(100, activation='relu')(x)
    predictions = layers.Dense(1, activation='sigmoid')(x)
    model = models.Model(inputs=sequences, outputs=predictions)
    model.compile(
        optimizer='rmsprop',
        loss='binary_crossentropy',
        metrics=['binary_accuracy']
    )
    return model

model = build_model()
```

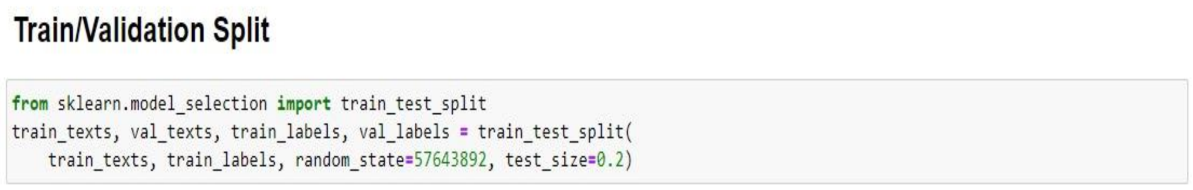
Results of the base model:

The study compared machine learning and deep learning models with SVM and CNN achieving the highest accuracy, respectively. Naive Bayes outperformed Random Forest, and hybrid models performed better than BiLSTM. Deep learning showed superior learning capacity, with CNN being the best-performing model overall, with an accuracy of 85%.

Proposed model:

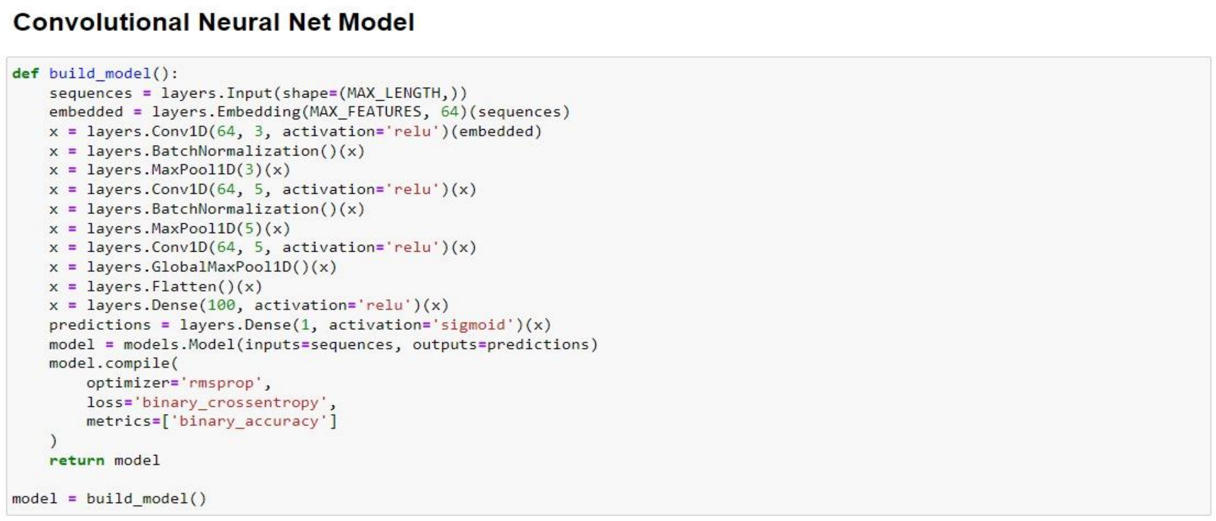
We have implemented an improved model using CNN and RNN that has yielded an accuracy of 95%.

CNN architecture used:



Train/Validation Split

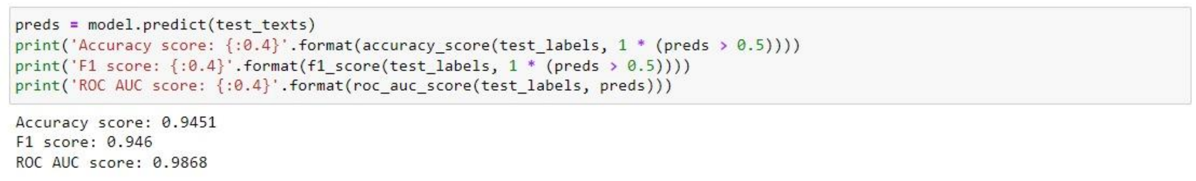
```
from sklearn.model_selection import train_test_split
train_texts, val_texts, train_labels, val_labels = train_test_split(
    train_texts, train_labels, random_state=57643892, test_size=0.2)
```



Convolutional Neural Net Model

```
def build_model():
    sequences = layers.Input(shape=(MAX_LENGTH,))
    embedded = layers.Embedding(MAX_FEATURES, 64)(sequences)
    x = layers.Conv1D(64, 3, activation='relu')(embedded)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPool1D(3)(x)
    x = layers.Conv1D(64, 5, activation='relu')(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPool1D(5)(x)
    x = layers.Conv1D(64, 5, activation='relu')(x)
    x = layers.GlobalMaxPool1D()(x)
    x = layers.Flatten()(x)
    x = layers.Dense(100, activation='relu')(x)
    predictions = layers.Dense(1, activation='sigmoid')(x)
    model = models.Model(inputs=sequences, outputs=predictions)
    model.compile(
        optimizer='rmsprop',
        loss='binary_crossentropy',
        metrics=['binary_accuracy']
    )
    return model

model = build_model()
```

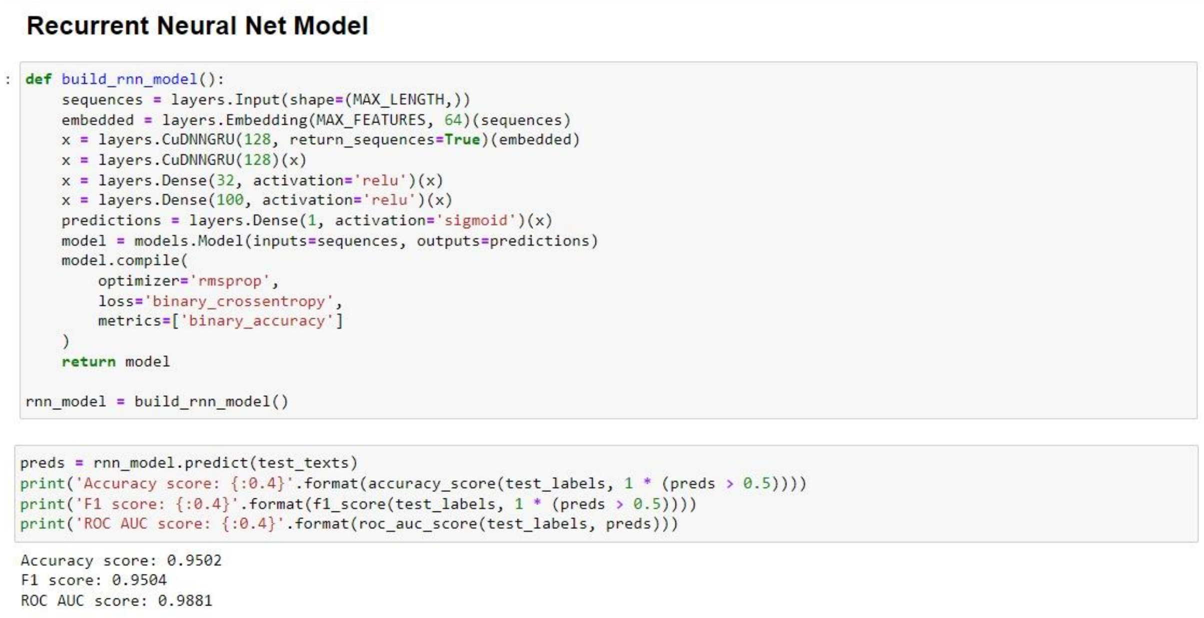


```
preds = model.predict(test_texts)
print('Accuracy score: {:0.4}'.format(accuracy_score(test_labels, 1 * (preds > 0.5))))
print('F1 score: {:0.4}'.format(f1_score(test_labels, 1 * (preds > 0.5))))
print('ROC AUC score: {:0.4}'.format(roc_auc_score(test_labels, preds)))
```

```
Accuracy score: 0.9451
F1 score: 0.946
ROC AUC score: 0.9868
```

The CNN model has yielded an accuracy of 94.5%.

RNN architecture used:



Recurrent Neural Net Model

```
def build_rnn_model():
    sequences = layers.Input(shape=(MAX_LENGTH,))
    embedded = layers.Embedding(MAX_FEATURES, 64)(sequences)
    x = layers.CuDNNGRU(128, return_sequences=True)(embedded)
    x = layers.CuDNNGRU(128)(x)
    x = layers.Dense(32, activation='relu')(x)
    x = layers.Dense(100, activation='relu')(x)
    predictions = layers.Dense(1, activation='sigmoid')(x)
    model = models.Model(inputs=sequences, outputs=predictions)
    model.compile(
        optimizer='rmsprop',
        loss='binary_crossentropy',
        metrics=['binary_accuracy']
    )
    return model

rnn_model = build_rnn_model()
```

```
preds = rnn_model.predict(test_texts)
print('Accuracy score: {:0.4}'.format(accuracy_score(test_labels, 1 * (preds > 0.5))))
print('F1 score: {:0.4}'.format(f1_score(test_labels, 1 * (preds > 0.5))))
print('ROC AUC score: {:0.4}'.format(roc_auc_score(test_labels, preds)))
```

```
Accuracy score: 0.9502
F1 score: 0.9504
ROC AUC score: 0.9881
```

The RNN model has yielded an accuracy of 95%

Possible Refinements in future:

Here are a few ways by which the accuracy of the model can be further improved:

Gated Recurrent Units(GRU):

GRU is a variation of recurrent neural networks (RNNs) that can effectively capture long-range dependencies in sequential data. Replacing LSTM with GRU in the model architecture might lead to better performance.

Transformers:

Transformers, particularly the Transformer architecture introduced in the "Attention is All You Need" paper, have shown remarkable success in various natural language processing tasks. Implementing a transformer-based model can potentially improve the model's understanding of context and relationships between words.

Bert Model:

Pre-trained BERT models have achieved state-of-the-art results in many NLP tasks. Fine-tuning a pre-trained BERT model on the sentiment analysis task can bring significant improvements to the accuracy.

Conclusion and remarks:

Our ongoing objective is to further enhance our model by leveraging advanced deep learning models. Nonetheless, it is essential to highlight our impressive achievement of 95% accuracy using CNN and RNN. We also recognize that there remains room for improvement and the need to explore additional facets while employing advanced techniques.