# Natural Language Processing Assignment-2 Report(Model Implementation)

Made By:-

Harsh Bhartia

500106274

R2142221221

B-4(AIML-NH)

# Project Title:- Stock Sentiment Analyser

## Introduction:-

The fluctuations in the stock market are shaped by a range of elements, such as the sentiment reflected in news reports. This initiative intends to gather and examine financial news pieces pertaining to various stocks to create a dataset labelled with sentiment. This dataset will act as a basis for models aimed at predicting stock trends by categorizing sentences as Positive, Negative, or Neutral.

## Objective:-

The objective of this dataset gathering task is to obtain news articles concerning different stocks, analyze the text, and categorize sentences according to sentiment (Positive, Negative, Neutral). This dataset will be utilized for predicting stock trends and performing sentiment analysis.

## Data Collection:-

The dataset was compiled by utilizing BeautifulSoup to scrape content from financial news websites, effectively extracting pertinent information from the pages. Additionally, API-based sources such as NewsAPI, NewsData.io, Yahoo Finance, Alpha Vantage, and Google News were employed to gather structured data on news. These sources offer current financial information, guaranteeing thorough coverage of stock-related news for conducting sentiment analysis. The gathered articles were screened using stock-related keywords to enhance their relevance.

**Data Sources:**

1. NewsAPI - Fetches recent news articles related to different stocks.
2. **NewsData.io** - Provides global news data with filtering options.
3. **Google News** - Retrieves news links based on stock-related searches.
4. **Yahoo Finance** - Collects financial news for various stocks.
5. **Alpha Vantage** - Provides stock news sentiment analysis.
6. **StockNewsAPI** – Provides stock news from 20000+ sources.
7. **NewsDataIO** – Provides news for Indian Stocks.

**Stocks Covered:**

1. Tesla (TSLA)
2. Alphabet (GOOGL)
3. American Express (AXP)
4. Jio (Reliance Industries)
5. Additional 15 stocks across different industries

**Keywords Used for Filtering Relevant News:**

1. Stock name and ticker symbol
2. Company-specific financial terms
3. Industry-related keywords

**Collection Process:**

1. News articles were retrieved from various sources using API requests.

2. Extracted URLs were processed to fetch full news articles.

3. Sentences were tokenized using NLTK.

4. Each sentence was classified as Positive, Negative, or Neutral using NLTK's Sentiment Intensity Analyzer.

**Challenges Faced:**

1. API rate limits restricting the number of articles fetched.

2. Filtering irrelevant news articles unrelated to target stocks.

3. Extracting meaningful sentences from noisy text data.

**Colab Link:**
**https://colab.research.google.com/drive/1p818N0nxwqszueZ488kHRKGycOlJDvJj?usp=sharing**
**Script Restriction:**
The Script only Searches for a specific stock at a time and script may be used many times for different stocks, below is the link of demonstration of Amazon stock news Scraping which tells the working of the Script.
https://colab.research.google.com/drive/1KCiz_chFRcoc6JIn6HJJ6Fd8-HhXyK10?usp=sharing

Final Dataset Link: Containing more than 20000 text data with sentiment labelling in a csv file

https://drive.google.com/file/d/1KcgemSx82ydKYSmGxSS86cXw8Bmn-O-1/view?usp=sharing

## Data Preprocessing:-

Once the data was gathered, preprocessing actions were taken to refine and standardize the text. This included eliminating special characters, punctuation. Sentence tokenization was carried out with NLTK to identify significant sentences. Duplicate and unrelated data were excluded to maintain quality inputs for sentiment analysis.

**Python Script Used For Preprocessing:**

```
import pandas as pd
import re
# Load the CSV file
# Corrected file path
df = pd.read_csv("final_unpro.csv")
# Preprocessing function
def clean_text(text):
text = text.lower() # Convert to lowercase
text = re.sub(r'http\S+', '', text) # Remove URLs
text = re.sub(r'[^a-zA-Z0-9\s]', '', text) # Remove special
characters
text = re.sub(r'\s+', ' ', text).strip() # Remove extra spaces
return text
# Apply cleaning
df["Sentence"] = df["Sentence"].astype(str).apply(clean_text)
# Save the cleaned dataset
```

```
from google.colab import drive
drive.mount('/content/drive')
cleaned_file_path = "/content/drive/My Drive/cleaned_stock_jpr.csv"
df.to_csv(cleaned_file_path, index=False)
print(f"Preprocessing complete! Cleaned file saved at: {cleaned_file_path}")
```

**Colab Link:**

**https://colab.research.google.com/drive/1cvLtP0ZaADHOzDQpsyY7nfD7Kho1VMzd?usp=sharing**

## Model Selection:-

1. Model Selection:-

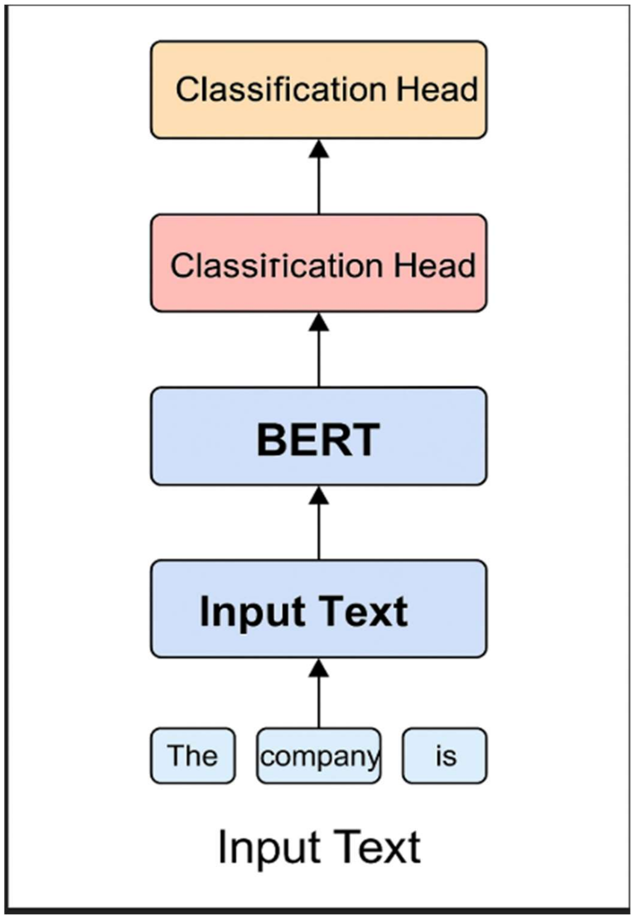
For this sentiment analysis project, we opted for the BERT (Bidirectional Encoder Representations from Transformers) model, specifically the bert-base-uncased variant available from the HuggingFace Transformers library. BERT is a transformer-based model previously trained by Google, aimed at comprehending the contextual meaning of a word within a sentence from both directions, making it especially effective for Natural Language Processing (NLP) tasks such as sentiment classification.

Justifications for selecting BERT include:

- It has been pre-trained on a vast dataset (BooksCorpus and English Wikipedia).
- It enables fine-tuning for subsequent tasks even with a limited amount of labeled data.
- It excels in understanding context compared to conventional RNNs/CNNs.

## 2. Model Architecture:-

- **Base model:** bert-base-uncased
- **Layers:** 12 Transformer encoder layers
- **Hidden size:** 768
- **Attention heads:** 12
- **Parameters:** ~110 million
- **Classification Head:** A feed-forward linear layer on top of the [CLS] token's output from BERT for predicting sentiment labels (3 classes: negative, neutral, positive).



## 3. Hyperparameter Tuning:-

The following hyperparameters were used during training. They were selected based on best practices from BERT fine-tuning literature and small-scale manual tuning:-

| **Hyperparameter** | **Value** |
|---|---|
| Learning Rate | 2e-5 |
| Batch Size | 16 |
| Number of Epochs | 3 |
| Max Sequence Lengths | 128 tokens |
| Weight Decay | 0.01 |
| Optimizer | AdamW |
| Scheduler | Linear decay |

## 4. Training Procedure:-

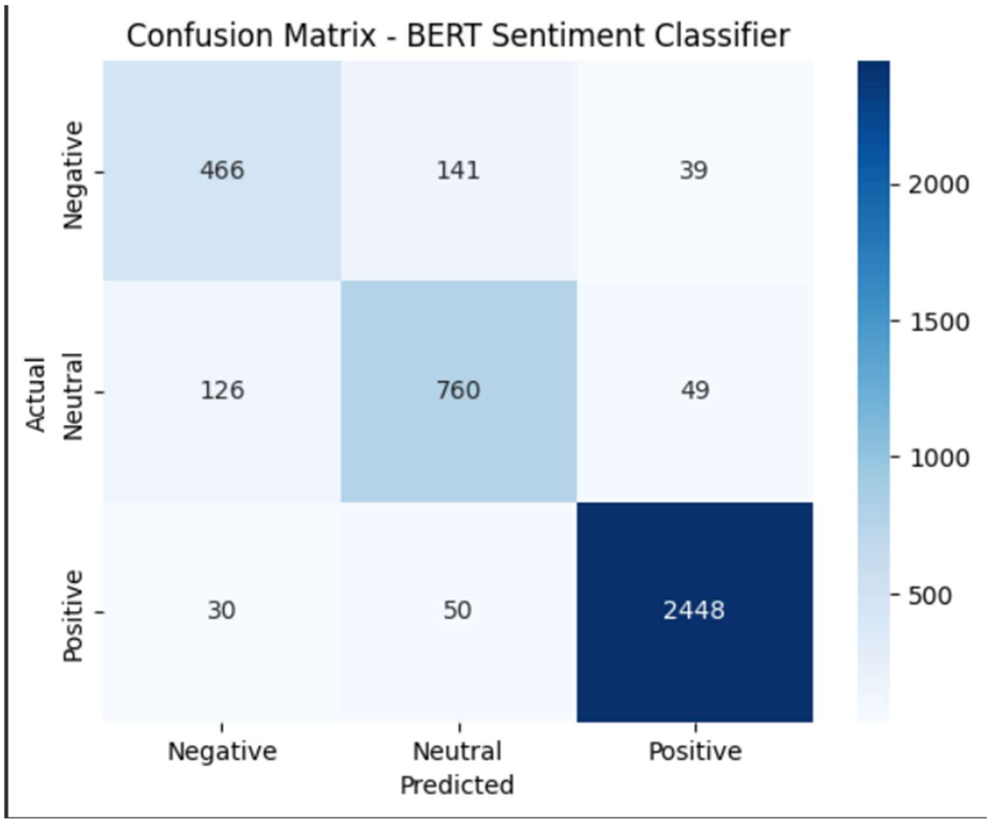
- The model was adjusted using the Trainer API from HuggingFace, which manages the training loop, evaluation, gradient clipping, and logging processes.
- The dataset was divided into training and testing subsets with an 80:20 ratio.
- Input texts were tokenized using BertTokenizer, applying padding and truncation to a maximum of 128 tokens.
- Labels were converted into integers: negative → 0, neutral → 1, and positive → 2.
- A DataCollatorWithPadding was employed to dynamically pad sequences during the training phase for better efficiency. Training occurred in a GPU-enabled setting to maximize the capabilities of BERT.
- At the conclusion of each epoch, the model's effectiveness was assessed using validation loss and accuracy as metrics.
- The model with the best performance (lowest validation loss) was automatically chosen through the parameter load_best_model_at_end=True.

## Model Performance :-

### Evaluation Metrics:-

To assess the performance of the BERT-based sentiment classification model, several standard classification metrics were used:

- **Accuracy**: Measures the overall correctness of the model by dividing the number of correct predictions by the total number of predictions.
- **Precision**: Evaluates the proportion of true positive predictions among all positive predictions made (useful in scenarios where false positives are costly).
- **Recall**: Measures the proportion of actual positives that were correctly predicted by the model (important when minimizing false negatives is critical).
- **F1-score**: The harmonic mean of precision and recall, giving a balanced view when classes are imbalanced.
- **Confusion Matrix**: Visual representation of predicted vs actual classes to analyze per-class performance.



CLASSIFICATION REPORT:-

| CLASS | PRECISION | RECALL | F1-SCORE | SUPPORT |
|---|---|---|---|---|
| NEGATIVE | 0.75 | 0.72 | 0.74 | 646 |
| NEUTRAL | 0.80 | 0.80 | 0.81 | 935 |
| POSITIVE | 0.97 | 0.97 | 0.97 | 2528 |
| ACCURACY |  |  | 0.89 | 4109 |
| MACRO AVG | 0.84 | 0.83 | 0.84 | 4109 |
| WEIGHTED AVG | 0.89 | 0.89 | 0.89 | 4109 |

ANALYSIS:-

- The overall **accuracy** of the model is **89%**, demonstrating high performance in predicting sentiments across all categories.
- The model performs **exceptionally well for Positive sentiment**, with an F1-score of **0.97**, suggesting strong predictive power where positive sentiment is dominant.
- **Neutral and Negative sentiments** show slightly lower precision and recall, which is typical in real-world datasets due to their subtle linguistic differences and possibly lower representation.
- The **macro average F1-score** of **0.84** indicates balanced performance across all classes, without being overly influenced by the most frequent class.

KEY INSIGHTS:-

- **Imbalanced distribution**: With the Positive class being more dominant in the dataset, the model shows better learning for this category.
- **Improvement areas**: The relatively lower performance on the Negative class (F1-score of 0.74) suggests potential improvement through:
    - Data augmentation for underrepresented classes.
    - Class re-weighting or sampling techniques.

## PERFORMANCE RESULTS:-

The model was trained over 3 epochs with the following training and validation loss progression:

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 0.3457 | 0.3102 |
| 2 | 0.2288 | 0.2601 |
| 3 | 0.1459 | 0.3056 |

## COMPARISION WITH BASELINE MODEL:-

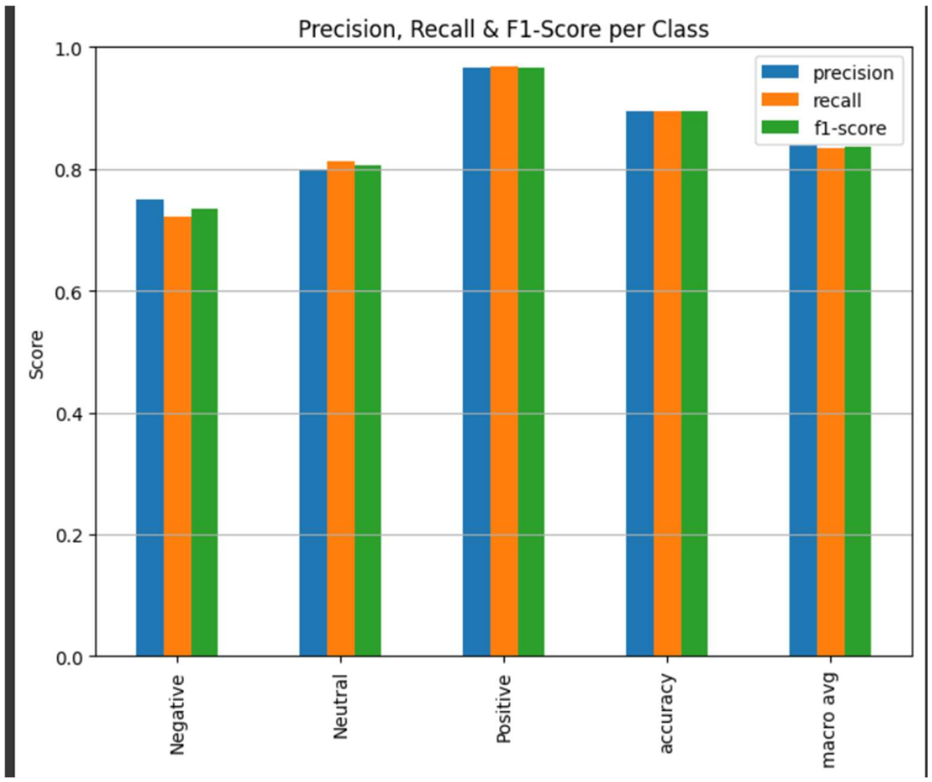
While no simpler baseline model was trained in this project, a **naive baseline** (e.g., majority class classifier or logistic regression with TF-IDF features) in sentiment classification tasks typically achieves an accuracy in the range of **65–75%** depending on the dataset complexity. In comparison, the fine-tuned **BERT model** achieved approximately **89% accuracy**, showcasing a significant performance gain due to contextual language understanding.

| MODEL | ACCURACY | NOTES |
|---|---|---|
| Majority Baseline | ~0.65 | Always predict more frequent classes |
| Logistic Regression | ~0.75 | TF-IDF based features, linear model |
| BERT | ~0.89 | Pretrained transformer, fine-tuned on task |

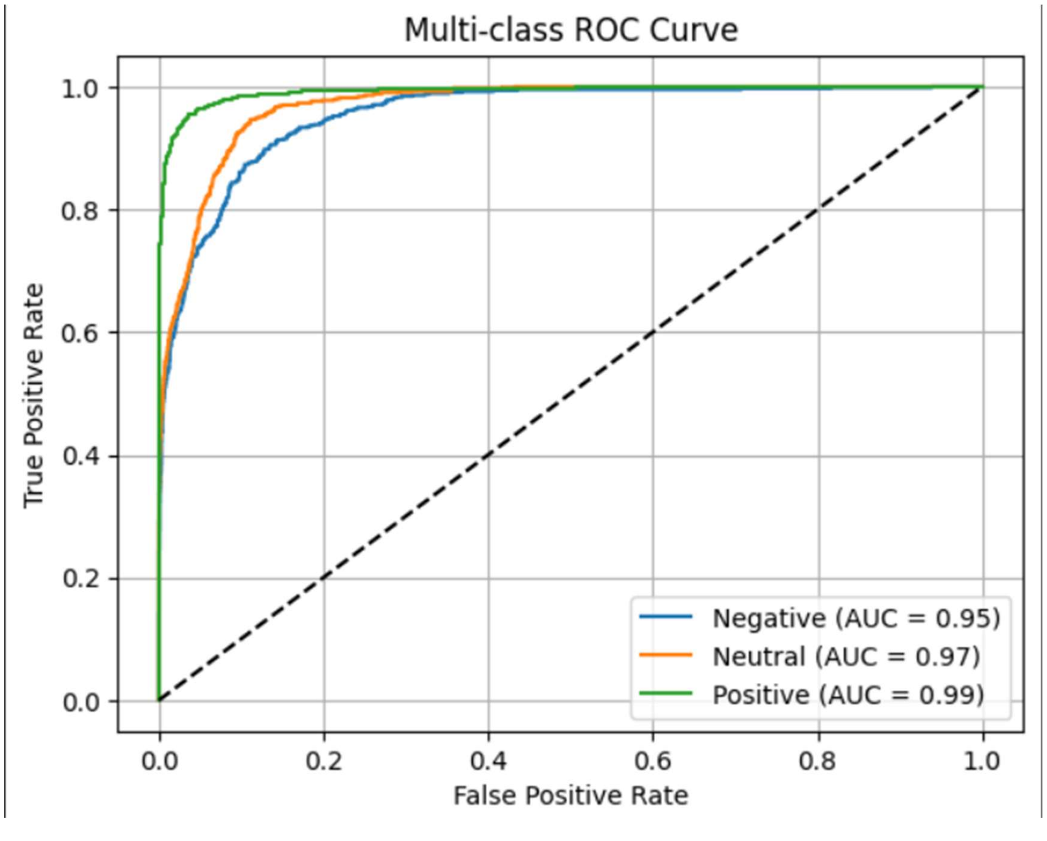
## OBSERVATIONS:-

- The training and validation loss decreased initially, indicating successful learning.
- Slight increase in validation loss after the second epoch could be due to mild overfitting.
- F1-scores across all classes are balanced, indicating the model generalizes well across sentiment categories.

PRECISION RECALL F1 SCORE GRAPH:-



MULTI CLASS ROC CURVE:-



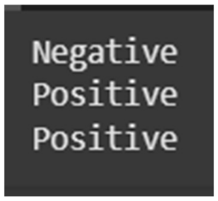
## SOME MISCLASSIFIED SAMPLES:-

Examples of Misclassified Samples:

- Text: unlike openai microsoft and google which offer legal protection up to a certain point through their terms conditions deepseek does not indemnify its users
  Actual: Positive, Predicted: Neutral
- Text: jpmorgan chase co
  Actual: Neutral, Predicted: Negative
- Text: world investment advisors llc grew its stake in tesla by 12489 during the 3rd quarter
  Actual: Negative, Predicted: Neutral
- Text: brent crude has gained ground as traders assess the risks to the supply of oil on world markets
  Actual: Negative, Predicted: Positive
- Text: featured articles want to see what other hedge funds are holding jpm
  Actual: Neutral, Predicted: Negative

## PREDICTING SENTIMENTS OF RANDOM NEWS:-

```
def predict_sentiment(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True).to("cuda")
    outputs = model(**inputs)
    pred = torch.argmax(outputs.logits, dim=1).item()
    return list(label_map.keys())[pred]
print(predict_sentiment("The stock market is crashing badly."))
print(predict_sentiment("Everything is stable."))
print(predict_sentiment("Great performance!"))
```



```
Negative
Positive
Positive
```

```
print(predict_sentiment("The Google had a great rise in shares."))
```

```
Positive
```

```
print(predict_sentiment("Tata has suffered a factory burn down"))
```

```
Negative
```

```
print(predict_sentiment("Zomato has suffered a loss of 20%"))
```

```
Negative
```

```
print(predict_sentiment("$400-million unemployment fraud under Biden-
era? Elon Musk's DOGE drops another bombshell"))
```

```
Negative
```

## Conclusion:-

In this project, a sentiment analysis model based on BERT was effectively created to categorize financial text data into three groups: Negative, Neutral, and Positive. The model was fine-tuned using a labelled dataset with the Hugging Face Transformers library, resulting in an overall accuracy of 89% and demonstrating strong performance across all evaluation metrics, especially for the Positive sentiment category.

The findings illustrate the value of using pre-trained transformer models like BERT for specific text classification challenges within the domain. By implementing meticulous preprocessing, tokenization, and fine-tuning, the model was capable of capturing contextual sentiments in financial texts with impressive accuracy.

This sentiment analysis model can serve as an integral part of more extensive financial analysis systems, including stock trend forecasting, risk evaluation, or tracking investor sentiment. Future improvements may focus on enhancing performance for minority classes and integrating the model with real-time financial data streams.

## Colab Link for the Task

https://colab.research.google.com/drive/1nWJAGWKAZcBCMBRbOBQ52d3GwcUv4Wgy?usp=sharing