Group Number: 09

Group Members: Gonçalo Nuno Matos Gomes (m20211007@novaims.unl.pt)

1. Introduction

The rapid rise of social media platforms like Twitter has made them a vital source for market sentiment analysis, offering valuable insights for financial decision-making. This project focuses on developing a robust NLP model to classify market sentiment into three categories: Bearish (0), Bullish (1), and Neutral (2). A comprehensive Machine Learning framework was established, including Exploratory Data Analysis, advanced Preprocessing and Feature Engineering to ensure the quality and relevance of the data. Several ML models, ranging from classical algorithms to cutting-edge Transformer architectures like RoBERTa, were employed to capture the different sentiments. These models were evaluated and compared using multiple performance metrics, culminating in an optimized solution capable of accurately predicting sentiment on unseen data.

2. Data Exploration

For this project, I began with two datasets: a training set (train.csv) and a test set (test.csv). The original test set, consisting of 2,388 records, was reserved for final predictions with the fully trained model and remained untouched throughout the framework development. To enable model training and validation, the original training set was split into two subsets: a training set with 7,157 rows and a validation set with 2, 2386 rows, maintaining the original columns, "text" and "label". This split ensured an unbiased evaluation of the models during development.

The Exploratory Data Analysis (EDA) phase was conducted exclusively on the new training set. This phase aims to understand the data distribution, assess class balance, and identify patterns within the textual data.

A word cloud was generated to visualize the most frequent keywords in the raw text data, offering insights into the dominant topics and terms. As shown, terms like "https," "stock," "market," "price," and "China" were highly prevalent, highlighting the financial and market-oriented nature of the dataset. This visualization reinforced the importance of cleaning the data by addressing redundant terms such as "https" and URLs during preprocessing to improve model performance.

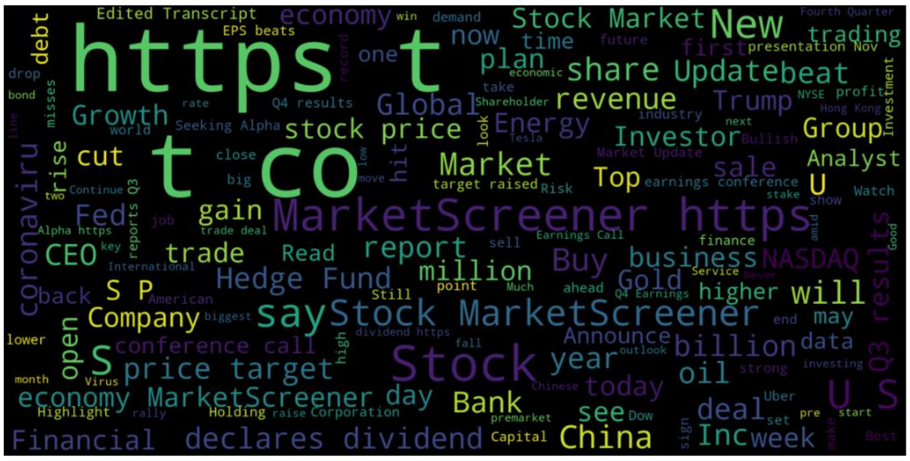


Figure 1: Word Cloud in Raw Data

The training dataset shows an imbalanced class distribution: Neutral dominates with 64.7%, followed by Bullish (20.1%) and Bearish (15.1%). This imbalance may lead to biased predictions favoring the majority class. To address this, techniques like class weighting or resampling are going to be considered, and metrics like the F1-score should be used for fair evaluation across all classes.

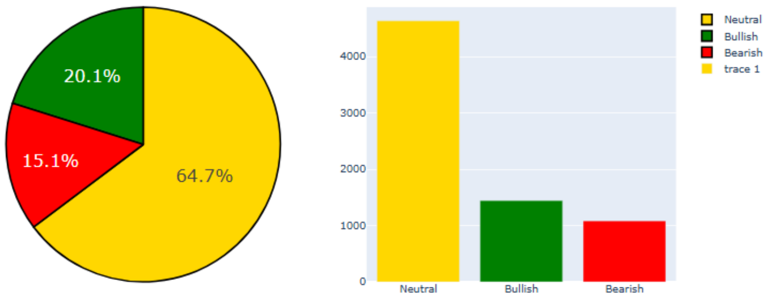


Figure 2: Target Distribution

As part of the data exploration process, a BERT Tokenizer was applied to the raw text to evaluate token length and character count distributions for each sentiment label.

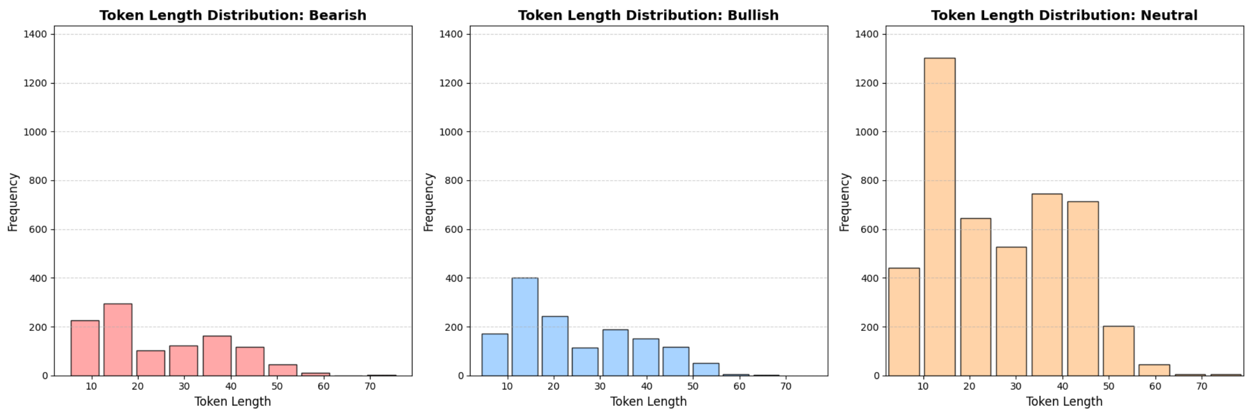


Figure 3: Token Length Distribution across Labels

- Most texts for all classes fall within 10–40 tokens, with Neutral tweets tending to be slightly longer on average compared to Bullish and Bearish tweets.
- Outliers with higher token counts exist but are minimal, indicating most tweets are concise and fit well within the typical transformer tokenization limit.

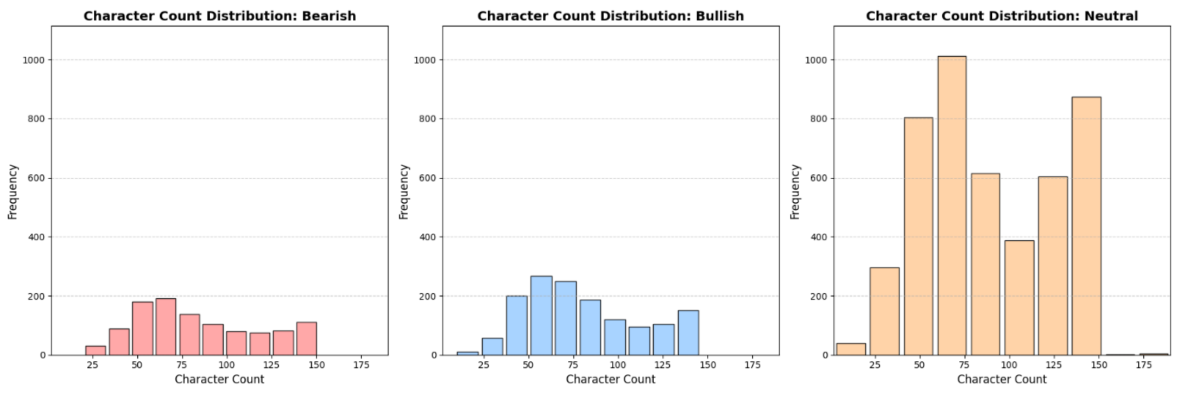


Figure 4: Character Count Distribution across Labels

- The character count distributions are consistent with token distributions, with Neutral tweets showing longer character counts on average.
- Most tweets for all classes are between 50–100 characters, reflecting the brevity of tweets as a social media format.

Following the tokenization process, an analysis of the most common words, unigrams, bigrams, and trigrams was conducted to better understand the textual structure of the dataset.

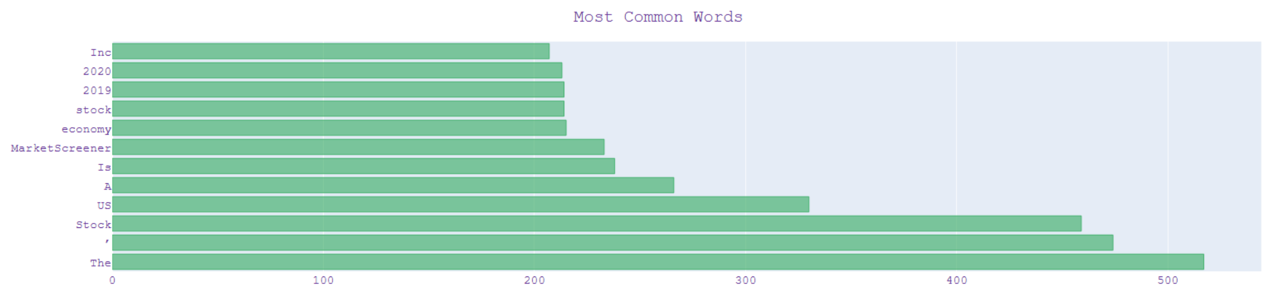


Figure 5: Most Common Words

- The most frequent words include generic terms like "The," "US," and "stock," indicating the prevalence of general market-related tweets. Terms like "MarketScreener" and "economy" also reflect the focus on financial topics.

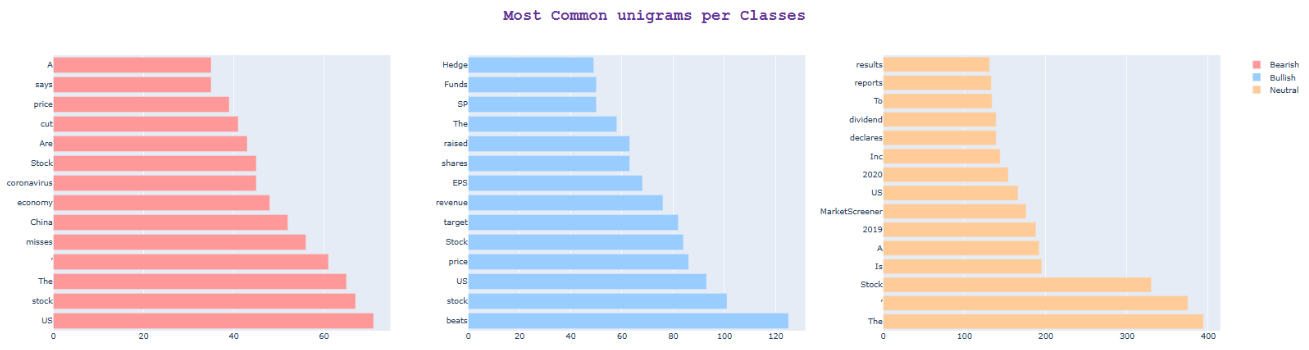
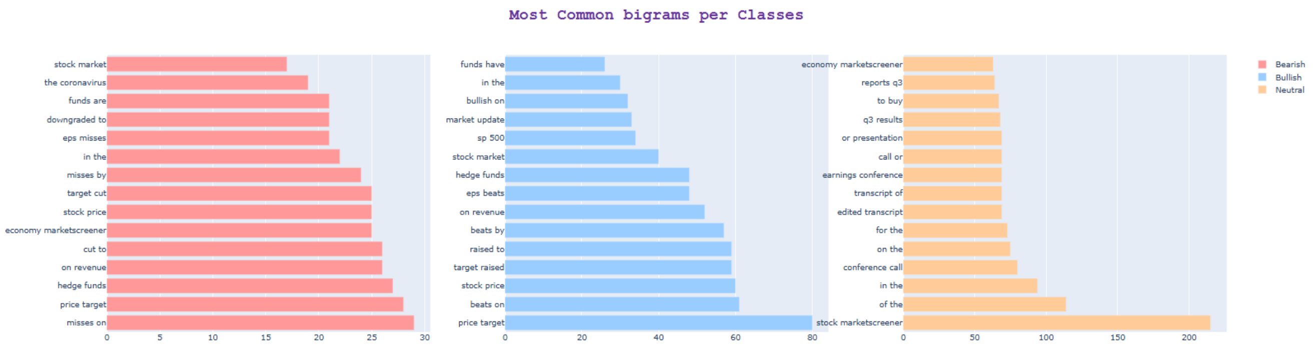


Figure 6: Most Common Unigrams per Label

The unigram distribution shows class-specific keywords. For example:

- Bearish: Words like "cut" and "misses" suggest negative sentiment.
- Bullish: Words like "raised" and "beats" highlight positive financial news.
- Neutral: Words such as "reports" and "results" align with neutral market updates.



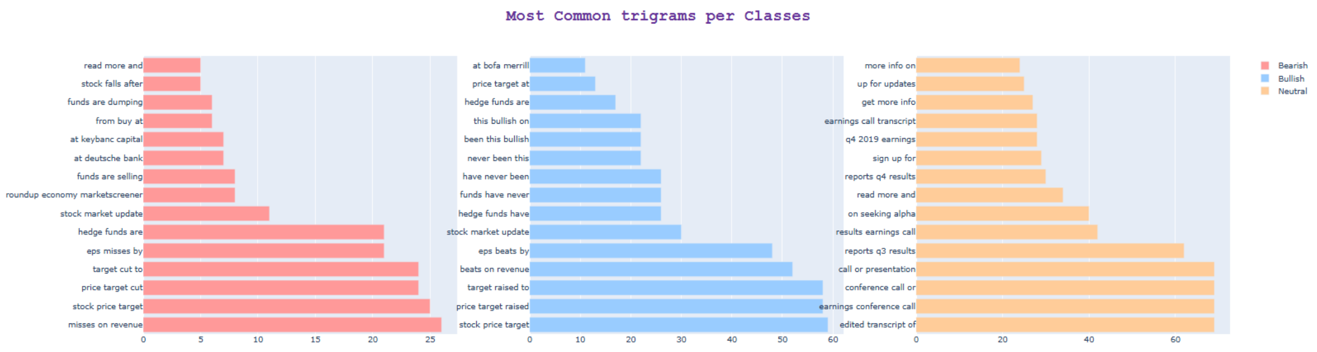


Figure 7 & 8: Most Common Bigrams and Trigrams

The bigrams and trigrams reveal common phrases across the classes:

- Bearish: Phrases like "misses on revenue" and "cut to target."
- Bullish: "EPS beats by" and "price target raised" reflect optimism.
- Neutral: Phrases such as "conference call transcript" and "earnings results" denote factual updates.

To further analyze the data, the presence of URLs in tweets and the diversity of languages in the dataset were examined. A new feature, **has_url**, was engineered as a binary flag to indicate whether a tweet contains a URL.

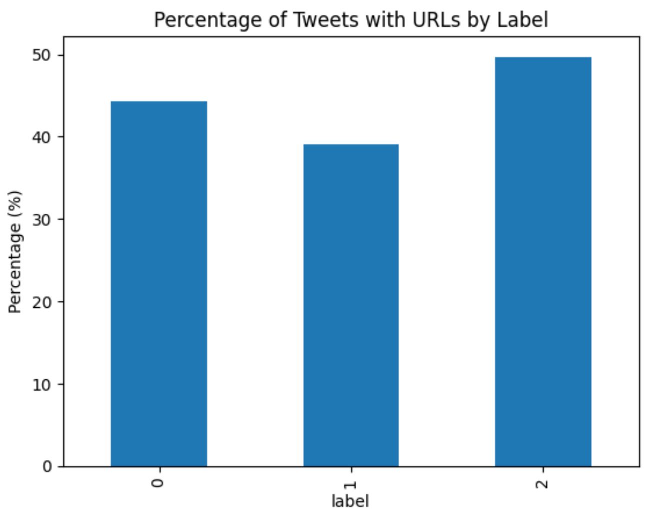
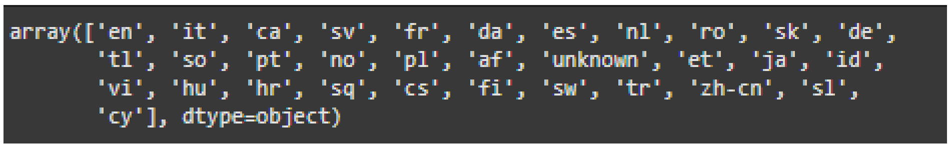


Figure 9: Percentage of Tweets with URLs by Label

This analysis shows that tweets labeled as Neutral have the highest percentage of URLs, while Bullish tweets have the lowest.

Additionally, I applied a language detection pipeline to identify the languages present in the dataset. This analysis revealed a diverse set of languages, as shown in the second attachment, including English, Italian, Spanish, and others.



```
array(['en', 'it', 'ca', 'sv', 'fr', 'da', 'es', 'nl', 'ro', 'sk', 'de',
       'tl', 'so', 'pt', 'no', 'pl', 'af', 'unknown', 'et', 'ja', 'id',
       'vi', 'hu', 'hr', 'sq', 'cs', 'fi', 'sw', 'tr', 'zh-cn', 'sl',
       'cy'], dtype=object)
```

Figure 10: Set of Languages detected on the Training Data

3. Data Preprocessing

In this project, two distinct preprocessing pipelines were designed to align with the requirements of different modeling approaches - classical Machine Learning models and Transformer models. Each pipeline ensures the data is prepared appropriately for its corresponding method.

Preprocessing for Classical Models

This preprocessing function is more comprehensive and aims to standardize and clean the data thoroughly for use in traditional algorithms such as Logistic Regression and KNN. The steps include:

- Removing URLs: Extracts and removes any URLs to focus on the text content.
- Translation to English: Standardizes text to English, as it is the dominant language, ensuring uniformity.
- Stop Words Removal: Eliminates commonly used stop words that do not contribute significant meaning (e.g., "the", "is").
- Possessives Removal: Cleans possessive words (e.g., "Here's" becomes "Here") for better token standardization.
- Lemmatization: Converts words to their base form (e.g., "running" → "run"), ensuring the semantic integrity of the text while reducing redundancy.
- Cleaning and Lowercasing: Removes punctuation, emojis, strange symbols, and converts text to lowercase to standardize the input.

This pipeline prepares the data to extract meaningful features and ensures compatibility with feature engineering techniques like BOW, TF-IDF, or Word2Vec, which will be applied later.

Preprocessing for Transformer Models

This preprocessing pipeline is tailored for Transformer-based models, which are generally more capable of handling raw text variations. Therefore, it applies lighter cleaning steps to retain nuanced information. The steps include:

- Removing URLs: Removes irrelevant links to focus on textual content.
- Translation to English: Ensures uniformity by translating all text to English.
- Selective Stop Words Removal: Retains certain stop words that contribute to sentence meaning or contrast, such as "not" and "but," as Transformers can benefit from these nuances.
- Lemmatization: Simplifies words to their root form without stripping context, which complements the tokenization process.
- Cleaning and Lowercasing: Removes emojis and lowercases the text while retaining most punctuation and symbols that might carry relevant information.

This lighter cleaning process preserves the semantic richness of the text, leveraging the inherent capabilities of Transformers to extract contextual insights.

The preprocessing phase employed two distinct pipelines to cater to the requirements of classical machine learning models and Transformers. To ensure flexibility and facilitate comparison, the original text column in the dataset was preserved, and a new *clean_text* column was created. All preprocessing steps were applied to the *clean_text* column, leaving the original text column unaltered for use in the Transformers-based approach.

4. Feature Engineering

In this phase, two distinct methodologies were applied for feature engineering, aligned with the respective preprocessing pipelines for classical Machine Learning models and Transformer models. Each method ensures that the data is prepared optimally for its modeling framework.

Feature Engineering for Classical Models

Bag of Words (BOW):

- Converts text into a matrix of token counts, where each column represents a unique word in the dataset, and each row represents the frequency of those words in a text sample.
- This technique is straightforward but can lead to sparse matrices, especially for large vocabularies. It captures word occurrences but lacks contextual information.

TF-IDF (Term Frequency-Inverse Document Frequency):

- Builds on BOW by assigning weights to words based on their importance in a document relative to the entire dataset.
- Words that occur frequently in a document but rarely across others are given higher weights, capturing relevance more effectively than simple word counts.
- This technique helps to reduce the impact of common but less informative words.

Word2Vec using GloVe:

- Pre-trained word embeddings from the glove-twitter-25 model were used to represent each word as a 25-dimensional vector. These embeddings capture semantic relationships between words.
- For each text sample, word vectors were averaged to produce a single 25-dimensional vector. An additional binary feature has_url was appended, resulting in a final 26-dimensional feature vector for each text sample.
- This approach incorporates word semantics and context.

Feature Engineering for Transformers

For Transformer-based models like BERT and Roberta, tokenization was employed to convert text into a format suitable for Transformer architectures:

BERT Tokenizer:

- Breaks down text into tokens using a WordPiece algorithm, which segments words into subwords to handle rare and unknown words effectively.
- Adds special tokens such as [CLS] (classification token) at the beginning and [SEP] (separator token) at the end of each text sample to guide the model in understanding sentence boundaries.
- Ensures uniform sequence lengths by padding or truncating sequences to the maximum length allowed by the model.

Roberta Tokenizer:

- Based on Byte Pair Encoding (BPE), it tokenizes text into subwords, similar to BERT, but with modifications tailored for Roberta's architecture.
- Retains rich context through its tokenization and avoids adding unnecessary sentence-specific tokens like [CLS] and [SEP].
- Also performs padding and truncation to fit model input size while leveraging Roberta's pre-trained embeddings.

5. Classic Classification Models

The classification phase employed three classical machine learning models—K-Nearest Neighbors (KNN), Logistic Regression, and Multi-Layer Perceptron (MLP)—to predict tweet sentiment based on three feature engineering approaches: Bag of Words (BOW), TF-IDF, and Word2Vec (W2V). The general methodology involved:

✓ **Feature Representation:**
- For BOW and TF-IDF, text data was converted into numerical matrices.
- For W2V, embeddings were generated using the glove-twitter-25 model, with the has_url binary feature appended to create 26-dimensional vectors.

✓ **Grid Search and Cross-Validation:**
- Hyperparameters for each model were optimized using grid search with 5-fold cross-validation, focusing on maximizing the F1-Score.
- This ensured robust parameter selection and minimized overfitting.

✓ **Evaluation:**
- Models were assessed using performance metrics, including classification reports and confusion matrices, on both training (via cross-validation) and test datasets.

K-Nearest Neighbors (KNN):

KNN relies on similarity-based predictions. Hyperparameters tuned included:
- n_neighbors: Number of nearest neighbors to consider [3, 5, 7].
- metric: Distance metrics ['euclidean', 'manhattan'].
- weights: Weighting schemes ['uniform', 'distance'].

The optimal parameters were selected based on the F1-Score, and the final model's performance was evaluated on unseen test data.

Logistic Regression:

Logistic Regression was applied for its interpretability and efficiency. The grid search optimized:
- C: Regularization strength [0.1, 1, 10].
- penalty: Regularization type ['l2', 'none'].
- solver: Optimization algorithms ['lbfgs', 'saga'].

Cross-validation ensured the best configuration, and performance was assessed on both training and test sets.

Multi-Layer Perceptron (MLP):

MLP, a simple neural network, captured nonlinear relationships in the data. The following hyperparameters were optimized:
- hidden_layer_sizes: Number and size of layers [(5,), (5, 5)].
- activation: Activation functions ['relu', 'tanh'].
- solver: Optimization algorithms ['adam', 'sgd'].
- alpha: L2 regularization [0.0001, 0.001].

The model was trained with a maximum of 500 iterations, with F1-Score guiding the tuning process. Evaluation metrics were reported for both training and test datasets.

6. Transformers

The integration of Transformer models into the framework followed a structured pipeline:

1. **Preprocessing**: The text column (text) was preserved in its raw form, while a cleaned version (clean_text) was created for classical models. Both columns were used for Transformer evaluation to compare performance between raw and preprocessed text.

2. **Tokenization**: Each Transformer utilized its respective tokenizer—BertTokenizer for BERT and RobertaTokenizer for RoBERTa. This step ensured the text data was tokenized and padded to match the input requirements of the models.

3. **Model Initialization**: Pretrained Transformer models (BERT and RoBERTa) were loaded with their sequence classification heads, fine-tuned to the specific task of predicting sentiment (Bearish, Neutral, Bullish).

4. **Data Loaders**: Tokenized datasets were wrapped into PyTorch DataLoader objects to facilitate batch processing during training and evaluation.

5. **Training**: The models were fine-tuned using AdamW optimizer and a learning rate scheduler with warm-up steps. The training loop implemented gradient clipping to prevent exploding gradients.

6. **Evaluation**: Performance was evaluated using metrics like F1-score and accuracy on the validation set, along with a confusion matrix for deeper insights. Each model's performance was assessed on both the raw text and the preprocessed *clean_text* columns for comparative analysis.

BERT Model:

- Tokenizer: The BertTokenizer was employed to tokenize and preprocess the input text, handling subtokens and special tokens (e.g., [CLS] and [SEP]).

- Model Architecture: A BertForSequenceClassification model was initialized from the bert-base-uncased checkpoint. This architecture was adapted to classify inputs into three labels.

- Key Features: BERT's bidirectional attention mechanism enabled it to understand contextual nuances, capturing dependencies between words in both directions.

- Training and Results: The model was fine-tuned for multiple epochs with an emphasis on minimizing validation loss and maximizing F1-score. Evaluation on both raw and preprocessed text highlighted the impact of preprocessing on model performance.

RoBerta Model:

- Tokenizer: The RobertaTokenizer handled tokenization, ensuring compatibility with the roberta-base model architecture.
- Model Architecture: A RobertaForSequenceClassification model was initialized from the roberta-base checkpoint. This model shared the same classification head structure as BERT but benefited from enhanced pretraining strategies and a larger dataset.
- Key Features: RoBERTa's dynamic masking during pretraining and optimizations like removal of Next Sentence Prediction resulted in a more robust understanding of textual relationships.
- Training and Results: Fine-tuning was conducted using the same methodology as BERT. Evaluation on both raw and preprocessed text demonstrated that RoBERTa achieved superior performance metrics, particularly in capturing sentiment nuances in financial text data.

7. Evaluation and Results

The evaluation phase revealed valuable insights into the performance of both classical models (KNN, Logistic Regression, and MLP) and Transformer-based models (Bert and RoBERTa) in the sentiment classification task, as detailed in the provided tables.

Classical Models:

Feature Engineering: Bag of Words (BOW)

Metric	KNN	LR	MLP
F1-Score (label 0)	0.25	0.53	0.55
F1-Score (label 1)	0.39	0.63	0.62
F1-Score (label 2)	0.82	0.86	0.86
F1-Score (label 3)	0.65	0.76	0.76

Feature Engineering: TF-IDF

Metric	KNN	LR	MLP
F1-Score (label 0)	0.40	0.54	0.51
F1-Score (label 1)	0.50	0.64	0.63
F1-Score (label 2)	0.84	0.86	0.83
F1-Score (label 3)	0.70	0.76	0.74

Feature Engineering: Word2Vec

Metric	KNN	LR	MLP
F1-Score (label 0)	0.31	0.07	0.10
F1-Score (label 1)	0.39	0.25	0.35
F1-Score (label 2)	0.77	0.79	0.80
F1-Score (label 3)	0.63	0.58	0.61

Transformers:

Metric	Bert	RoBERTa
F1-Score (label 0)	0.63	0.82
F1-Score (label 1)	0.72	0.85
F1-Score (label 2)	0.88	0.92
F1-Score (label 3)	0.82	0.89

Key Observations:

1. Classical Models:

   - Among classical models, Logistic Regression (LR) and Multi-Layer Perceptron (MLP) consistently outperformed KNN across all feature engineering techniques.
   - TF-IDF proved to be the most effective text representation method for classical models, yielding the highest f1-scores for all sentiment labels.
   - Word2Vec delivered the poorest results, indicating its limitations in capturing sentiment-related nuances when used with simple averaging techniques.

2. Transformers:

   - RoBERTa emerged as the best-performing model, achieving an f1-score weighted average of **0.89**, outperforming Bert's **0.82**. This result highlights the robustness of RoBERTa’s pretraining and its superior ability to capture contextual information.
   - A key decision in using Transformers was applying them on the **raw text** instead of the preprocessed *clean_text* column. This choice was guided by empirical testing, where the raw text consistently delivered better results, as the Transformers' tokenization and pretraining are optimized to handle raw, unaltered language data.

Metric Explanation:

- The f1-score was selected as the primary evaluation metric due to its balanced consideration of precision and recall, making it highly suitable for imbalanced datasets where certain labels (e.g., bearish sentiment) are less frequent. Unlike accuracy, the f1-score avoids bias from over-represented labels, ensuring a more reliable evaluation.

Champion Model:

- The RoBERTa model was identified as the champion model, demonstrating superior performance across all sentiment labels. Its advanced architecture and tokenization strategy allowed it to excel in capturing the subtleties of market sentiment from tweets, making it the optimal choice for this task.

Final Step:

- To maximize its predictive potential, the RoBERTa model was retrained on the full original training dataset. This approach ensured that all available data was utilized before predicting sentiments on the original test set.