MDSAA

Master Degree Program in

Data Science and Advanced Analytics

Text Mining

Stock Sentiment - Predicting market behavior from tweets

Paulo Martins, r2015469

Beatriz Fonseca, r20201599

Group 20

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Project available on GitHub

INDEX

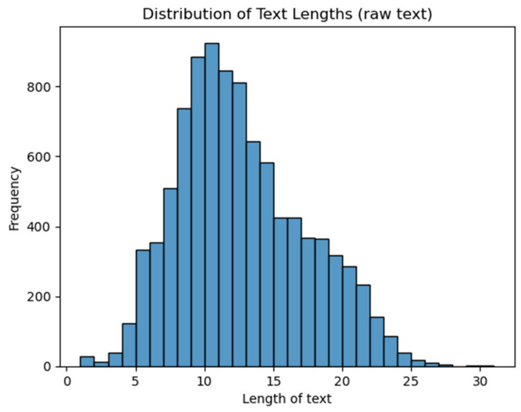
1. INTRODUCTION

Sentiment analysis plays a key role in identifying the emotional tone conveyed in textual data. In the context of the stock market, this technique has gained significant traction for its potential to predict market trends and understand investor sentiment. This project focuses on applying Natural Language Processing (NLP) models and techniques to classify the sentiment of tweets related to the market. Each tweet is categorized as *Bearish* (0), *Bullish* (1), or *Neutral* (2), reflecting its overall sentiment toward market movement.

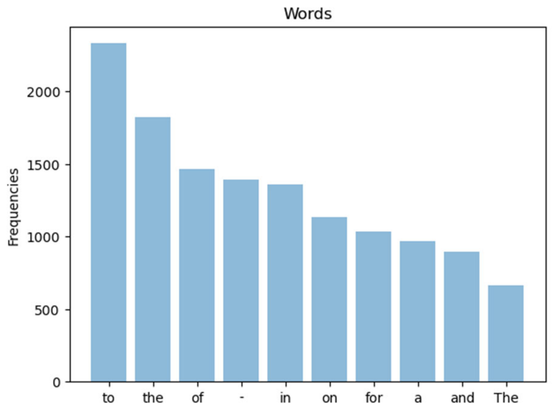
2. DATA EXPLORATION

2.1. Text Feature

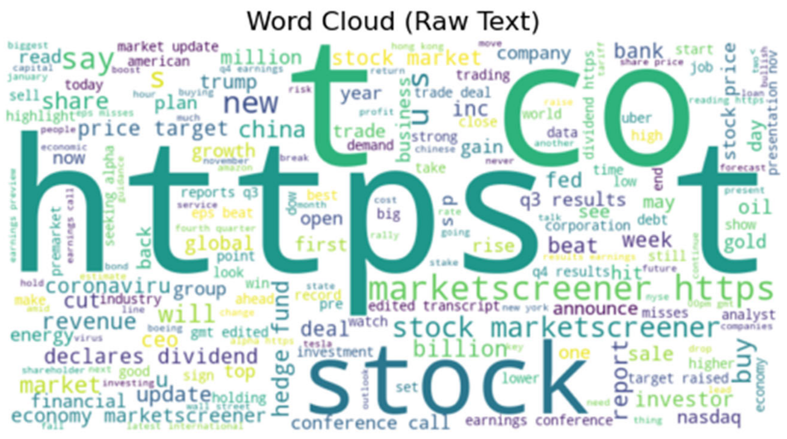
As with any predictive modeling process, the first step involves exploring and understanding the available data. The dataset contained no missing values, and the average tweet length in the training set was approximately 12 words. Upon analyzing the distribution of text lengths, a small number of extremely short entries - specifically those with three words or fewer - were identified. Those belonging to the majority class (*Neutral*) were removed to reduce noise and potential bias.



An examination of word frequency revealed that the top 10 most common words were stopwords, appearing in both lowercase and uppercase forms, as illustrated in the plot below. These were addressed during preprocessing, and subsequently replaced with stock market–related terms, which will be discussed in the following section.

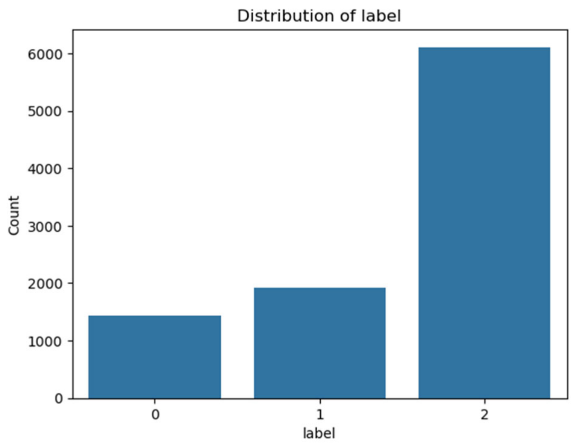


Additionally, the initial word cloud highlighted irrelevant tokens such as single letters and web-related text like ‘https’ as the most prominent, as shown in the image below.



2.2. Target Feature

An analysis of the target feature's distribution reveals a significant class imbalance. The *Neutral* class makes up approximately half of the dataset, with a representation nearly twice as large as the combined total of the *Bearish* and *Bullish* classes. In contrast, these two latter classes appear to be relatively balanced with respect to each other.

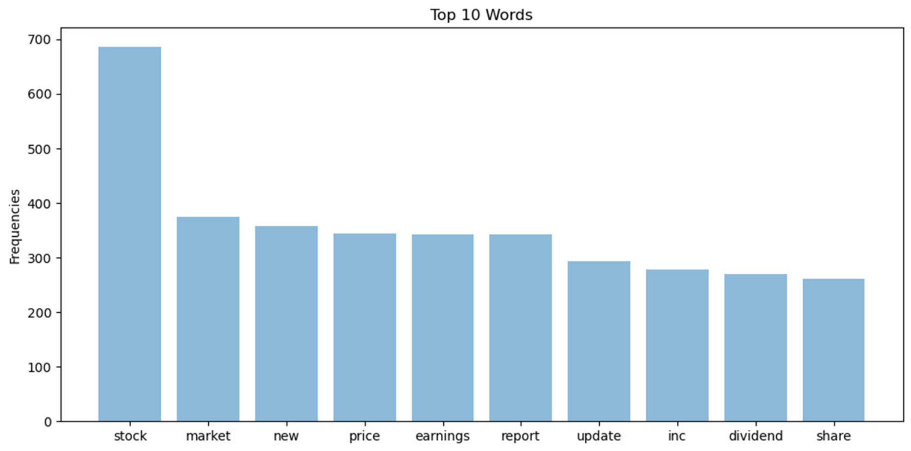


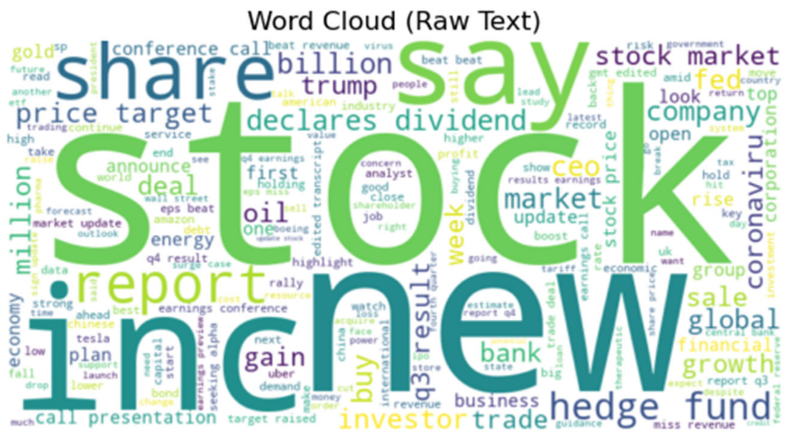
3. DATA PREPROCESSING

To preprocess the data, a custom function was developed. This function takes as input the raw tweets, a configuration dictionary specifying which transformations to apply, a set of stopwords, a lemmatizer object, and a stemmer object.

The preprocessing function performs a series of text cleaning and normalization steps, including: expanding acronyms, removing extra spaces, emojis, stock tickers, URLs, and Twitter handles, as well as filtering out prices, punctuation, special characters, stopwords, contractions, possessives, geolocations, integers, and dates. It also supports lemmatization, stemming, and converting all text to lowercase. Each of these transformations can be toggled via the configuration dictionary.

The specific preprocessing steps applied in this project were based on a heuristic basis and are detailed in the accompanying notebooks. Below are the updated plots showing the new top 10 most frequent words and the revised word cloud.





4. FEATURE ENGINEERING

The **feature engineering** techniques that were implemented in the project were **Bag of Words**, **Word2Vec** and **DistilBERT** as the mandatory techniques, and **RoBERTa** and **FinBERT** as part of the extra work that was developed by the team.

The **Bag of Words** implementation converts text into fixed-length vectors, where each dimension represents the presence or absence of a specific word.

The **Word2Vec** implementation captures semantic similarity from learning dense, low-dimensional vectors of words based on context. To facilitate integration with the *sklearn* pipeline, a custom class **W2VVectorizer** was developed. This class generates fixed-length vectors for entire corpus by averaging the individual word vectors.

DistilBERT is a transformer-based language model that produces context-aware embeddings. This implementation uses the pretrained *distilbert-base-uncased-finetuned-sst-2-english* model to extract classification (CLS) token embeddings for sentiment classification.

Similarly, the **RoBERTa** and **FinBERT** implementations differ only in the pretrained models used. The **RoBERTa** implementation uses the pretrained *cardiffnlp/twitter-roberta-base-sentiment* model, which specializes in sentiment analysis of Twitter data; while the **FinBERT** implementation uses the pretrained *yiyanghkust/finbert-tone* model, which is fine-tuned for financial sentiment analysis.

5. CLASSIFICATION MODELS

The **classification models** tested include **K-Nearest Neighbors** (KNN), **Long-Short Term Memory** (LSTM) and **RoBERTa** as the mandatory classification methods, and **FinBERT** and a **Language Model** as part of the extra work that was developed by the team.

The **KNN** implementation classifies each input by identifying its *k* nearest neighbors based on cosine similarity and predicting the class through a weighted majority vote - where closer neighbors exert a stronger influence. This model was evaluated using all five feature engineering techniques.

The **LSTM** implementation processes sequences of word embeddings using a Bidirectional LSTM layer, which captures contextual information from both past and future tokens. The final prediction is made through a softmax output layer. Due to format incompatibility, this model was tested with all feature engineering techniques except for Bag of Words.

Both the **RoBERTa** and **FinBERT** models leverage HuggingFace pipelines for sentiment classification, using the pretrained *cardiffnlp/twitter-roberta-base-sentiment* and *yiyanghkust/finbert-tone* models, respectively. Each model extracts a CLS token embedding to encode the input into a single, fixed-length vector, serving both as a feature representation and as the basis for sentiment prediction.

The **Langauage Model** implementation performs sentiment classification using a text generation HuggingFace pipeline and the pretrained *tiiuae/falcon-7b-instruct* model. Instead of relying on traditional feature engineering, the model is prompted with a system message that includes a detailed task description along with a few illustrative examples. Since Large Language Models are capable of understanding and reasoning over raw text, no feature engineering techniques were applied in this approach.

6. EVALUATION AND RESULTS

The models mentioned above were all tested using the same preprocessing conditions and with the mentioned feature engineering techniques. The evaluation was performed using Stratified K-Fold with 5 folds. The results are presented in the table below and the confusion matrices can be checked in the Appendix section.

Classifier	**Feature Engineering**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
KNN	Bag of Words	0.7396	0.7499	0.7373	0.7499
	Word2Vec	0.5930	0.6407	0.6053	0.6407
	DistilBERT	0.7119	0.7179	0.7137	0.5222
	RoBERTa	0.7593	0.7659	0.7612	0.7659
	FinBERT	0.7534	0.7643	0.7540	0.7643
LSTM	Word2Vec	0.6189	0.5468	0.5739	0.5468
	DistilBERT	0.7217	0.6708	0.6857	0.6708
	RoBERTa	0.7453	0.6990	0.7107	0.6990
	FinBERT	0.7561	0.7202	0.7311	0.7202
RoBERTa	RoBERTa	0.6076	0.6353	0.6058	0.6353
FinBERT	FinBERT	0.6710	0.6778	0.6739	0.6778
Language Model	-	0.4685	0.2302	0.2132	0.2302

According to the results discussed earlier, the encoder-based models overall demonstrated inferior performance when compared to the more traditional machine learning methods. Notably, the standalone Language Model emerged as the weakest performer among all models evaluated, indicating its limited effectiveness in this specific context and the limited available resources.

Given these findings, the K-Nearest Neighbors (KNN) model, when combined with RoBERTa as the feature engineering technique, consistently outperformed all other models across every evaluation metric. This superior and consistent performance established it as the most reliable and effective approach. Consequently, this model was selected for making predictions on the test dataset, as it demonstrated the highest potential for accurate and robust classification.

7. CONCLUSION

Based on the results obtained, it is evident that utilizing Word2Vec for feature engineering yielded comparatively lower performance than the other approaches explored in the study. This suggests that, despite its popularity in earlier natural language processing applications, Word2Vec may not capture the nuanced semantic and contextual information required for more complex financial text classification tasks. Furthermore, the performance of the encoder-based models did not surpass that of the more traditional machine learning techniques, indicating that the added complexity of these models does not necessarily guarantee superior outcomes in all scenarios.

The Language Model was clearly the poorest performing model among all those tested, which may be attributed to potential model misspecification or incompatibility with the applied preprocessing pipeline. Due to limited team resources, it was not feasible to experiment with more advanced or fine-tuned versions of the model that could potentially yield better results. Additionally, the team strongly suspects that the preprocessing steps—specifically the removal of stopwords and the application of lemmatization—may have negatively impacted the model's performance. Since language models are typically pre-trained on large corpora of natural, unaltered text, stripping away linguistic elements that contribute to the flow and meaning of natural language may have impaired the model's ability to accurately interpret and contextualize the input data. As a result, the model not only struggled to make reliable predictions but also significantly underrepresented the majority class, further highlighting its limitations under the current experimental setup. When passing the input to ChatGPT, it was still able to make accurate predictions when given the output constraints and information regarding the preprocessing treatment, further reinforcing the team's belief that a larger model would have been able to yield better results.

However, among the encoder models evaluated, FinBERT - an architecture specifically pre-trained on financial and market-related language - demonstrated a clear advantage over RoBERTa, which is primarily trained on data derived from Twitter and social media platforms. This discrepancy in performance highlights an important insight: the contextual relevance of the training data appears to have a more significant impact on model effectiveness than the structural format of the text itself. In other words, aligning the model's training corpus with the target domain - in this case, financial language - yields more meaningful and accurate results than focusing solely on the stylistic or structural characteristics of the input text.

This further supports the idea that using a uniform preprocessing configuration for all models may have negatively impacted the results, by biasing them towards simpler models like KNN, and that tailoring the preprocessing for each model could have led to better overall performance.

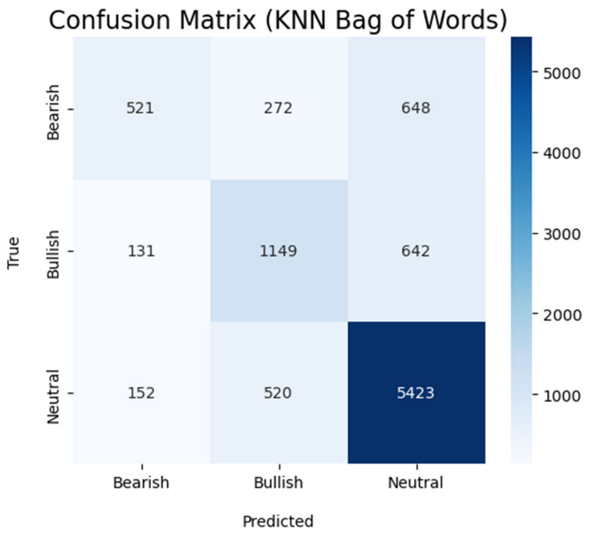
8. REFERENCES

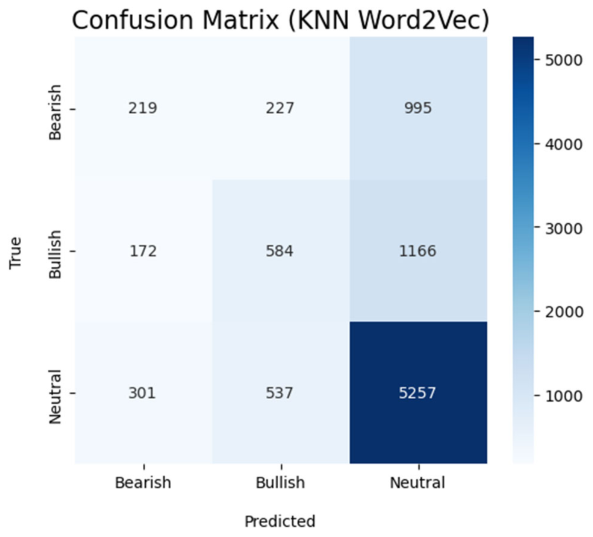
Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach*. arXiv preprint arXiv:1907.11692. https://arxiv.org/abs/1907.11692

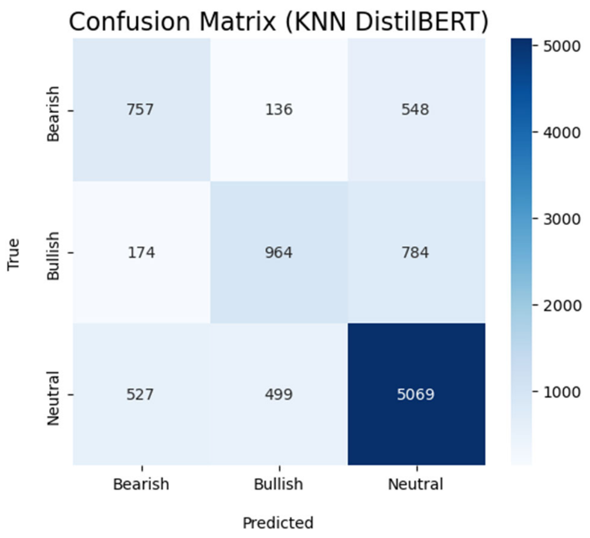
Araci, D. (2019). *FinBERT: Financial sentiment analysis with pre-trained language models*. arXiv preprint arXiv:1908.10063. https://arxiv.org/abs/1908.10063

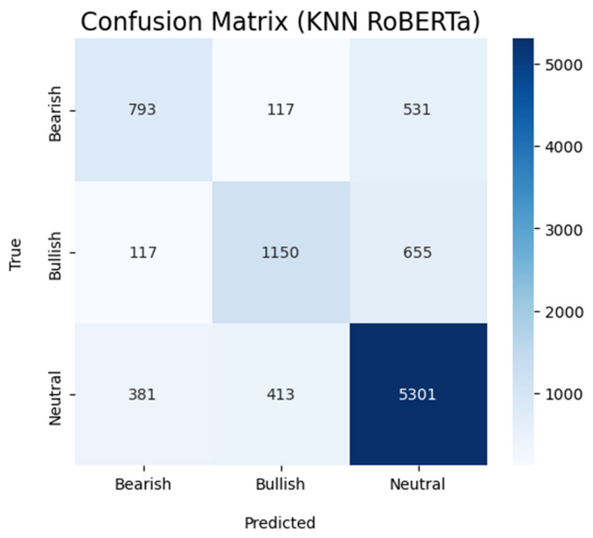
Falcon-LLM Team. (2025, May). Falcon-H1: A family of hybrid-head language models redefining efficiency and performance. Retrieved from https://falcon-lm.github.io/blog/falcon-h1 [Visited on June 10th, 2025]

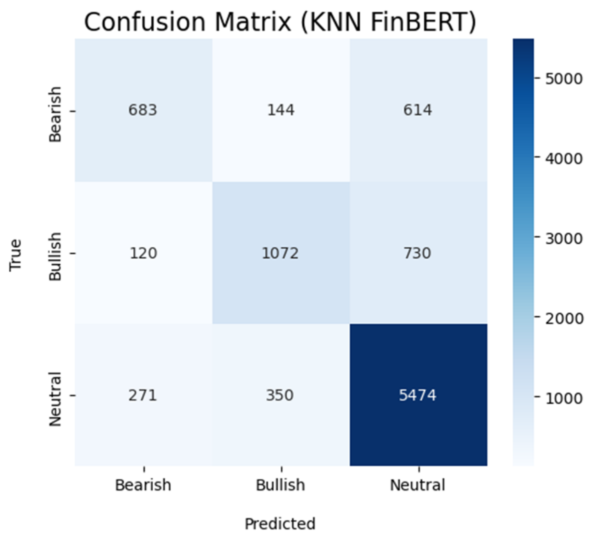
9. APPENDIX

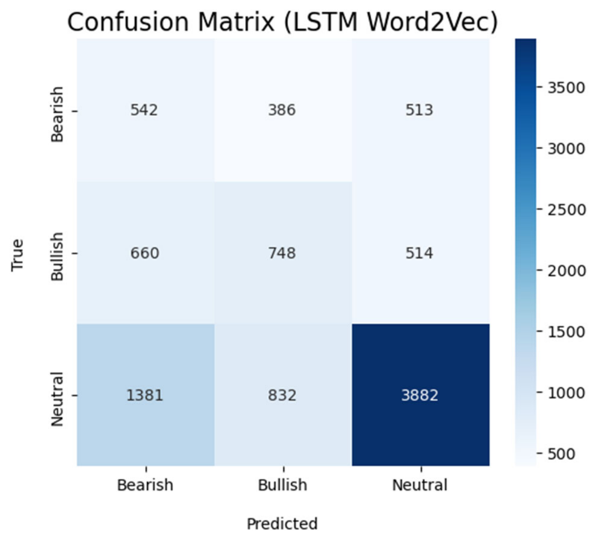


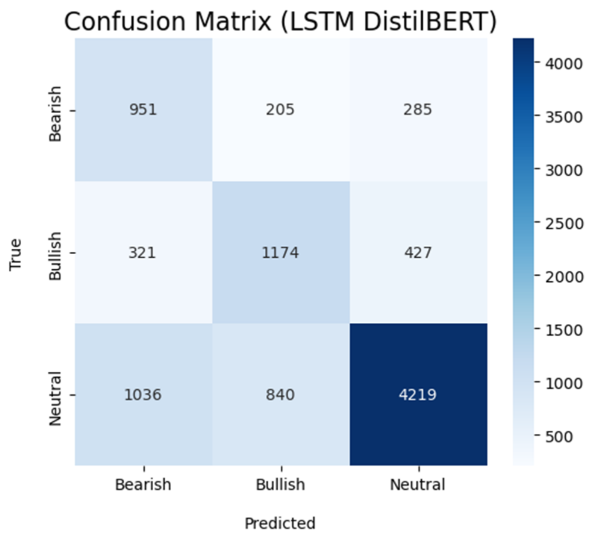


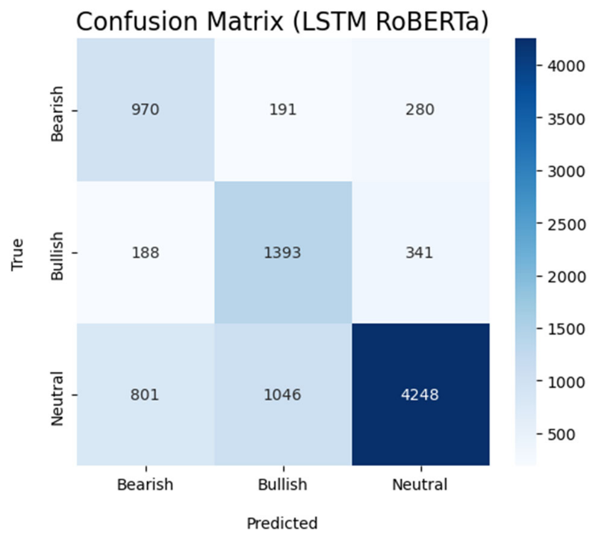


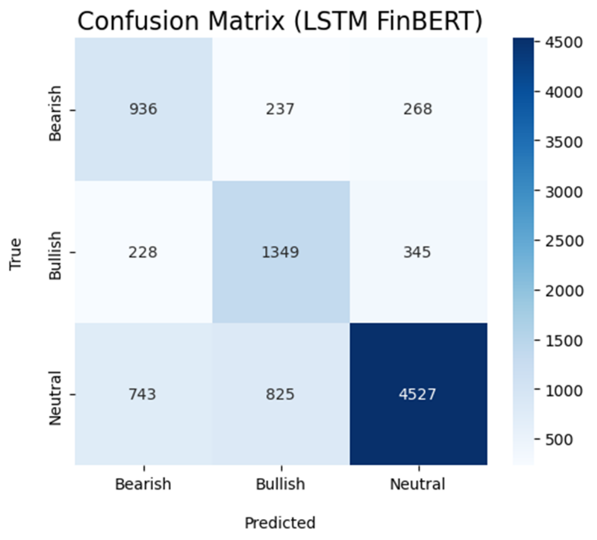


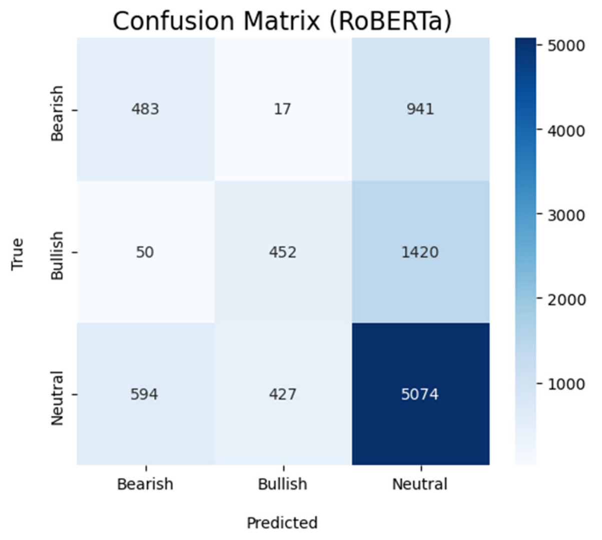

Confusion Matrix (KNN FinBERT)
True
Predicted
Bearish
Bullish
Neutral
683 144 614
120 1072 730
271 350 5474
5000
4000
3000
2000
1000

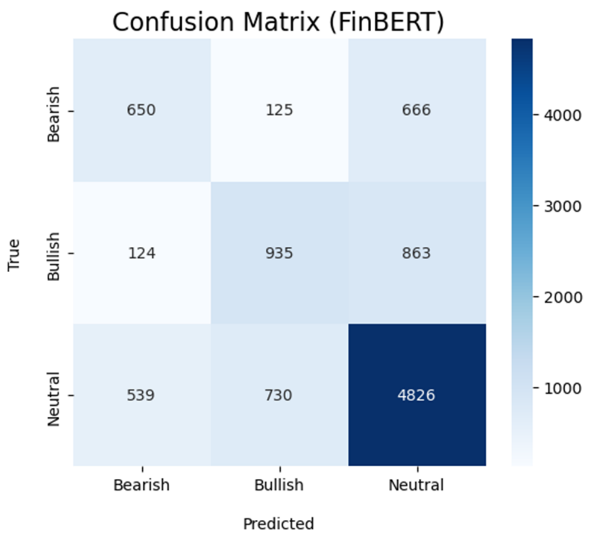

Confusion Matrix (LSTM Word2Vec)
True
Predicted
Bearish
Bullish
Neutral
542 386 513
660 748 514
1381 832 3882
3500
3000
2500
2000
1500
1000
500

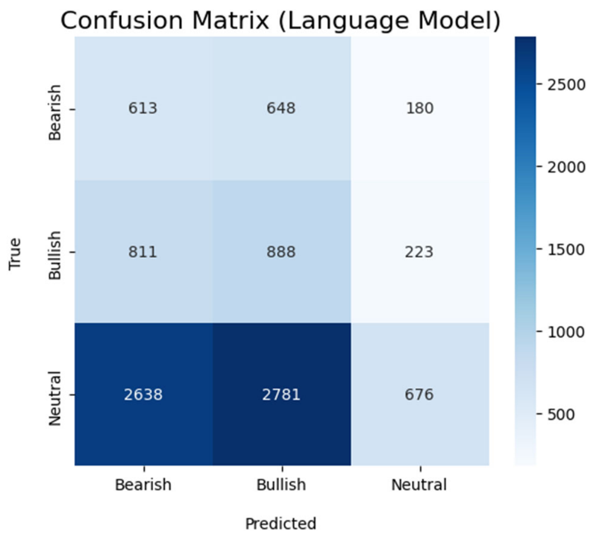

Confusion Matrix (LSTM DistilBERT)
True
Predicted
Bearish
Bullish
Neutral
951 205 285
321 1174 427
1036 840 4219
4000
3500
3000
2500
2000
1500
1000
500


Confusion Matrix (LSTM RoBERTa)
True
Predicted
Bearish
Bullish
Neutral
970 191 280
188 1393 341
801 1046 4248
4000
3500
3000
2500
2000
1500
1000
500


Confusion Matrix (LSTM FinBERT)
True
Predicted
Bearish
Bullish
Neutral
936 237 268
228 1349 345
743 825 4527
4500
4000
3500
3000
2500
2000
1500
1000
500


Confusion Matrix (RoBERTa)
True
Predicted
Bearish
Bullish
Neutral
483 17 941
50 452 1420
594 427 5074
5000
4000
3000
2000
1000


Confusion Matrix (FinBERT)
True
Predicted
Bearish
Bullish
Neutral
650
125
666
124
935
863
539
730
4826
1000
2000
3000
4000


Confusion Matrix (Language Model)
True
Predicted
Bearish
Bullish
Neutral
613
648
180
811
888
223
2638
2781
676
500
1000
1500
2000
2500