TEXT MINING PROJECT

Master in Data Science and Advanced Analytics

NOVA Information Management School

Universidade Nova de Lisboa

Stock Sentiment
Predicting market behavior from tweets

Group 9

Martim Tavares, 20240508

Santiago Taylor, 20240542

Rita Palma, 20240661

Tomás Silva 20230982

Index

1. Introduction

Natural Language Processing (NLP) is the process of converting unstructured text into structured data, its implementations enabling great information gain that wasn't previously possible. Social media presents a rich source of data from which several insights can be gained, one such insight being **sentiment analysis**. On platform X (formerly known as Twitter), feelings of optimism and pessimism can be understood through NLP and used to identify potential trends in the stock market, these being bullish (optimistic) or bearish (pessimistic).

To achieve an understanding of people's sentiments, data will be analyzed to see how to categorize a person's feelings towards the market. An exploration of various text cleaning and normalization techniques gives way for the creation of predictive models - such as Bag-of-Words + K-NN, TF-IDF + Gaussian NB, TF-IDF + Logistic Regression, Word2Vec + Bi-LSTM, T5-embeddings + Logistic Regression, DistilBERT and RoBERTa - to interpret and classify the text's attitude toward the market. This allows for the application of the theoretical knowledge gained during the course whilst simulating a real-world application of NLP in the financial domain.

Further work was done to explore the preprocessing of emojis, like whether it was best to remove them, tokenize them or leave them as is, as well as the use of Optuna to automatically explore a search space of parameters with which to test models and find the best configurations.

2. Data Exploration

The corpus contains 9 543 training tweets that fall into three sentiment classes: bearish (0), neutral (1) and bullish (2). The distribution is moderately imbalanced; bullish messages outnumber bearish ones by roughly four to one (Figure 1). Tweets are short, averaging just over twelve words, and they often include ticker symbols, broker verbs and finance jargon. A large percentage of the posts contain embedded URLs that contribute no sentimental value and are therefore removed during preprocessing. Emojis appear in about 0.5% of documents and may provide potential insights for sentiment analysis. Preliminary word-frequency plots confirm that stop-words dominate raw counts, which justifies their elimination before feature extraction (Figure 2).

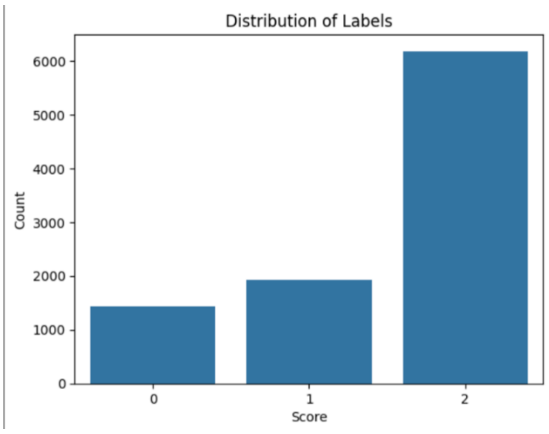

Figure 1. Label distribution.

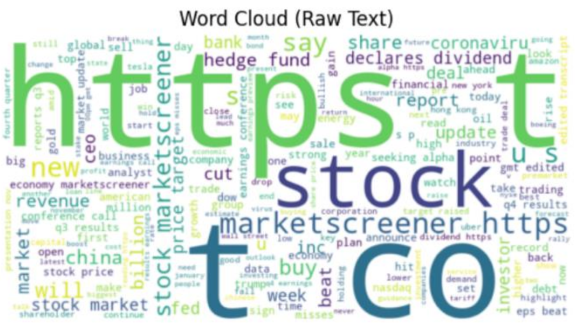

Figure 2. Word cloud for raw test.

3. Data Preprocessing

Before further exploration, data was divided into train and validation sets, such that models could learn from training data and then be compared to "unseen" data to understand their generalization capabilities. We implemented several text-cleaning methods on both the training and validation datasets and experimented with different combinations to optimize model performance. The applied steps were:

Lowercasing: All text was converted to lowercase to ensure uniformity.
URL removal: Links were stripped from the text using regular expressions, as they added no sentiment value.
Emoji and emoticon handling: Depending on the configuration, emojis were either removed entirely or converted into descriptive tokens using the emoji library (e.g., 😄 → smiling_face). Similarly, common ASCII emoticons like :), :(, or :/ were mapped to keywords such as *happy*, *sad*, or *uncertain*.
Punctuation removal: All punctuation marks and newline characters were removed.
Tokenization: The text was split into individual tokens using NLTK's tokenizer.
Stopword removal: Common English stopwords were filtered out using the NLTK stopword list.
Lemmatization: Words were reduced to their base forms (e.g., *running* → *run*) using WordNet's lemmatizer.
Stemming: Snowball stemming was also tested to reduce words to their root forms; however, this often produced unnatural or truncated words, which negatively impacted model performance. As a result, stemming was ultimately excluded from the final pipeline.

4. Feature Engineering

Several steps were taken regarding the feature engineering stage of the project. The results of the models depend greatly on this step, seeing as these features are what the models learn from. It is important to approach several techniques carefully, ensuring that each step correctly represents the original documents.

The first algorithm applied was Bag-of-Words (BoW), a widely used technique in NLP used to transform textual data into machine-readable format, more specifically converting words into numeric values. Due to the nature of BoW, each tweet becomes represented as a sparse vector, recording the presence of the constituent words in the document. Sparse vectors store a lot of “meaningless” information and aren’t the most preferable method of feature engineering, but BoW still presents a robust framework for extracting information from text. Additionally, all words are given the same importance and word order/semantics aren’t accounted for, which can be detrimental for model training.

Term Frequency-Inverse Document Frequency (TF-IDF) works similarly to BoW, but instead of only accounting for the presence of words, each word in the document is attributed to an importance based on its relative frequency. Words that appear in several documents are given less relevance for being common, whereas rare words that appear several times in a single document likely indicate a higher level of importance, which is reflected in that word's representation.

Word2Vec works differently to the previous algorithms and is more geared towards applications of Neural Network-based models that learn with word embeddings. It allows for the extraction of semantic relations between words, being able to link similar words by mapping them onto a dense vector. Word2Vec isn’t free of weaknesses, as it is unable to handle polysemy -- the coexistence of many possible meanings for a word or phrase.

In addition to the algorithms explored in class, one other feature engineering algorithm was implemented to enhance the predictive quality of the models. The T5 Encoder model [1] developed by Google implements the use of contextual embeddings, allowing for the understanding of more nuanced speech that Word2Vec cannot capture. Tokens pass through an encoder model, outputting hidden states for each. Token embeddings are averaged to generate single fixed-size vectors for each sentence, thus capturing the semantic context and being able to detect sarcasm, tone, financial jargon, and other subtleties. Documents go through a different preprocessing pipeline for a more raw and natural output, allowing the encoder to capture more information than with the other preprocessing steps used.

5. Classification Models

Several different models were tested to see which performed best at correctly predicting the label associated with each document. Simple models are unlikely to outperform more complex ones, but may give more robust results and avoid overfitting, so models of varying complexity were explored. Preprocessing can affect performance, and when a grid search to determine the best method was not possible, an educated decision was made based on the way the model worked.

K-Nearest Neighbour (KNN) is an instance-based learning algorithm that classifies new data points based on how similar they are to points in the training set, using the cosine distance to measure similarity. Its simplicity allows for an easy interpretation of the results and allows for baseline model evaluation. BoW was the chosen preprocessing algorithm, but due to its creation of sparse matrices, the models struggle to fit well to the data.

The Gaussian Naive Bayes is a probabilistic classifier based on the Bayes' theorem and assumes that the features are independent and normally distributed. Despite this, it performs relatively fast and reasonably well on certain types of data. TF-IDF was more appropriate than BoW and provided an efficient and lightweight classification model.

Logistic regression is a linear model used for classifying multiple classes by estimating the probability of each class label using the softmax function. It learns the feature weights that best separate the classes. It is particularly effective when working with high-dimensional and sparse data, which makes it a good fit for BoW and TD-IDF representations. This sparsing leverages the strengths of both components: TF-IDF for capturing term importance and logistic regression for handling the resulting feature vectors, providing a robust and interpretable baseline for text classification

Bi-LSTM (Bidirectional Long Short-Term Memory) is a recurrent neural network. It's an extension of a traditional LSTM network that processes sequences in both forward and backward directions to capture context from both past and future tokens. This bidirectional structure makes it especially useful for tasks when understanding the full context of a word within a sentence is important. Seeing as tweets are composed of very few words, it makes sense that this would be useful for understanding word context. It was paired with word2vec, which supplies meaningful input vectors, while the Bi-LSTM learns the dependencies and structure within the sequence, making this a powerful approach for capturing both word-level semantics and sentence-level context

In addition to these models, we used two pre-trained, DestilBERT and RoBERTa. Both models have their own methods of preprocessing, so no prior changes were necessary since they perform most of the work. DestilBERT is a smaller and faster version of BERT, created using knowledge distillation. Like Bert, it retains the contextual understanding while being computationally more efficient, offering a strong balance between performance and speed. RoBERTa, on the other hand, is an optimized version of BERT. It does not use the next-sentence prediction and is trained with much larger batches and longer times. These differences allow it to better capture hidden patterns at the cost of higher computational requirements

6. Evaluation and Results

Table 1 summarizes the performance of the different classification pipelines, evaluated on both validation and training datasets. Overall, transformer-based methods significantly outperform classical models. Among classical methods, TF-IDF combined with Logistic Regression delivers slightly superior performance compared to the other classical approaches, highlighting the value of weighting word importance. However, classical pipelines, especially BoW combined with KNN and TF-IDF combined with Gaussian Naïve Bayes, exhibit very high accuracy on the training data but substantially lower scores on validation, indicating overfitting.

The introduction of richer semantic context significantly improves performance. Specifically, using contextual embeddings from the T5 encoder with Logistic Regression improves macro F1 scores notably compared to traditional text-representation methods.

Deep-learning models further improve the results. DistilBERT achieves strong validation performance while maintaining a good balance between accuracy and computational efficiency. RoBERTa, however, achieves the best overall validation performance across all metrics, confirming its effectiveness for accurately capturing nuanced sentiment in financial text. The small gap between RoBERTa's training and validation scores demonstrates its robustness and strong generalization capabilities compared to classical alternatives.

Table 1. Train and validation scores for the different pipelines tested.

	Validation				Train			
Pipeline	Accuracy	Macro F1	Precision	Recall	Accuracy	Macro F1	Precision	Recall
BoW + KNN	0.782	0.663	0.743	0.628	0.999	0.999	0.999	0.999
TF-IDF + GNB	0.730	0.623	0.640	0.611	0.997	0.996	0.994	0.998
TF-IDF + LR	0.786	0.664	0.792	0.617	0.883	0.833	0.939	0.775
T5 + LR	0.797	0.714	0.740	0.695	0.851	0.795	0.830	0.770
W2V + Bi-LSTM	0.764	0.655	0.676	0.639	0.820	0.741	0.775	0.717
DistilBERT	0.855	0.806	0.813	0.799	0.958	0.943	0.945	0.941
RoBERTa	0.897	0.872	0.862	0.883	0.973	0.965	0.960	0.970

7. Conclusion

As can be seen from the results, RoBERTa produced the best results across all evaluation metrics. Using the Macro F1 score, with a value of 0.872, it can be said that RoBERTa is able to predict very well on unseen data with a good balance amongst all possible classes. RoBERTa's preprocessing can capture deep contextual relationships between words, which simpler models and algorithms (like KNN with BoW) fail to understand. Being a pretrained model, much like DistilBERT, its understanding of language patterns is much greater than that of models trained purely on the training documents provided. For these reasons, it makes sense that RoBERTa outperforms the other model and preprocessing combinations.

Despite its positive results, the best model suffers from quite severe overfitting, much like all other models tested. The final model was tested with very few epochs but still returned training accuracy scores that were extremely high and are unlikely to reflect actual predictive capacities when faced with unseen data. Efforts were made to reduce this overfitting, but little could be done to properly address this issue. Despite this, the validation scores were still relatively good, and seeing as the cost of an incorrect prediction is not very severe, the overfitting was left as it was. In future studies, more ways to prevent overfitting could be studied to provide the model with better generalization capabilities.

8. References

[1] T5 – Hugging Faces. Accessed on June 14 2025
https://huggingface.co/docs/transformers/model_doc/t5

[2] Bidirectional LSTM in NLP – Geeks for Geeks. Accessed on June 14 2025
https://www.geeksforgeeks.org/bidirectional-lstm-in-nlp/

[3] DistilBERT – Hugging Face. Accessed on June 14 2025
https://huggingface.co/docs/transformers/model_doc/distilbert

[4] RoBERTa – Hugging Face. Accessed on June 14 2025
https://huggingface.co/docs/transformers/model_doc/roberta