Group Number: **Group #20**

Group Members: **Bernardo Machado** **20221001@novaims.unl.pt**

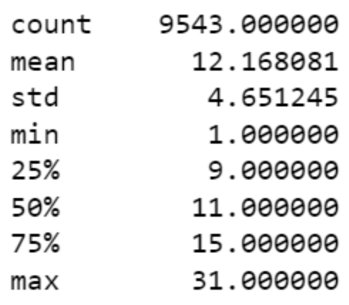
Nelson Lima **20221539@novaims.unl.pt**

1. Introduction

This study is for the Text Mining course, a development of a Natural Language Processing (NLP) model. We had to create a project based on stock tweets. We created four models to predict the sentiment of each tweet **- Bearish (0), Bullish (1), or Neutral (2)**, using python 3 with NLTK, Scikit Learn and TensorFlow as main libraries. The purpose of this work is to retrieve signals from tweets in a fast and direct way to help investors in the decision-making process. This new information extracts value from a wide range of data in a simple and summed up form. We used two data sets, train.csv and test.csv. We trained our data in train.csv and tested on test.csv. As Feature Engineering we implemented TF-IDF and Word Embeddings. The models used were GNB, KNN, MLP and LSTM. In general, we managed to have good results in our models.

2. Data Exploration

The data exploration of this dataset is performed by using Pandas library in Python. The code begins by importing the dataset from the CSV files using the pd.read_csv method. The exploratory analysis begins with the visualizations of the train data and the test data, by printing the columns of both datasets using the columns attribute. The following command was to print the describe method for the text column of the train_df dataset to get a statistical summary of the column. Before initiating the word count, we had to run a verification to check if there were any NULL values in the text column by using the isna method and sum function. Since it returned 0 (zero) we had the confirmation that there were not null values in the text column. For the word count, a new column was added to the train_df dataset: inicial_word_count which contained the number of words in the text column. We used the lambda function to apply the len function on each row of the text column so it could count the number of words. In the next step by using the describe method the code prints a statistical summary (**SEE Figure 1**) of the inicial_word_count column of the train_df dataset. It shows the count= 9543, mean=12.168081, standard deviation=4.651245, minimum=1, maximum=31 and quartile values: 25%=9, 50%=11, 75%=15 of the inicial_word_count column. And after this the code prints each analysis: the maximum, minimum, mean, median=11, mode=10, square root=4.6512245064 and unique values of the inicial_word_column by using the respective function. Regarding the visualization of the data, we created a bar chart (**SEE Figure 2**) with the distribution of the word count concerning the frequency. As demonstrated it is possible to verify that the maximum frequency was around 2500 and the word count maximum around 30. An important notice is that <u>we did not remove any outlier</u> because for this study we establish that every tweet is independent. Also, in **Figure 3** we performed the boxplot to check the dispersion of the initial word count. Moreover, to realize the frequency we needed to split the string by each word and then we created a dataframe (series) for all the words and their frequencies. Then we allocated the word frequency values to a list in a descending direction, with the first 15 values. The word “to” was the most recurring word with a total of 2332 times. Before the creation of the bar chart (**SEE Figure 4**) with the most common words, we had to transform the list of frequency words to an index.



```
count    9543.000000
mean       12.168081
std         4.651245
min         1.000000
25%         9.000000
50%        11.000000
75%        15.000000
max        31.000000
```

Figure 1: Statistical Summary

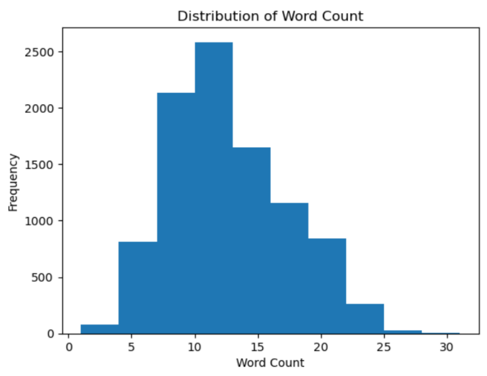


Figure 2: Distribution of Word Count

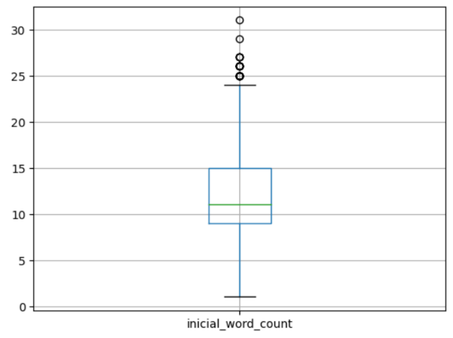


Figure 3: Initial Word Count

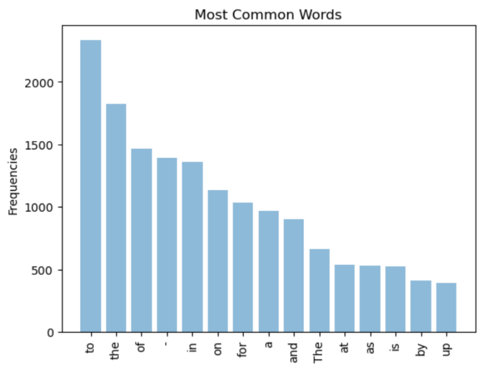


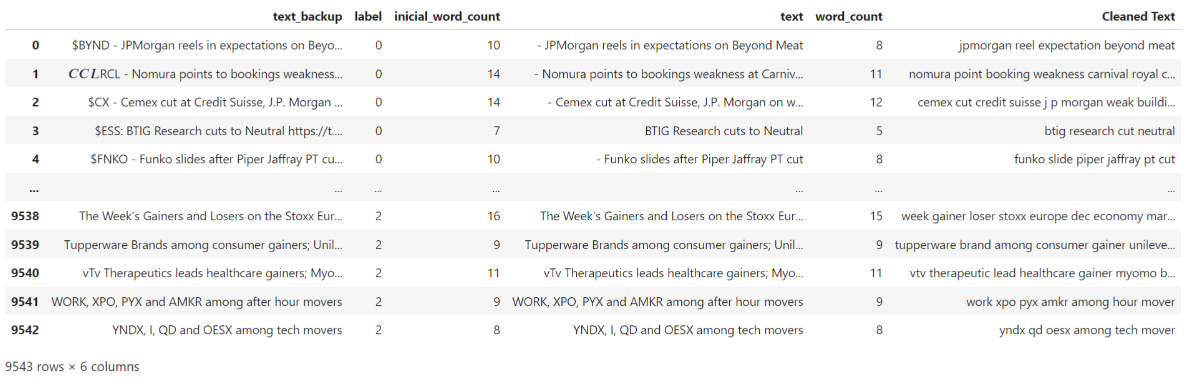
Figure 4: Most Common Words

This data exploration process was very important, first the statistical summary and its analysis and also the primary data cleaning which consists in a fundamental step towards the preparation of the dataset for further analysis and model construction.

3. Data Preprocessing

Stop Words: is a process to reduce dimensionality so we started by removing the punctuation and rearranged all the words with no punctuation in a lowercase text. While using the NLTK library we removed stop words and then created a new variable all_words_clean in order to have the text with no punctuation and no stop words. After cleaning we checked again which were the most frequent words and the word stock was the most frequent with 544 values. The text appeared with many website links and stock tickers, so we decided to create a print before and after. For this process we removed the words with "http" to remove the links and the Stock Ticker by removing the words with "$" followed by letters. Our code was working but we noticed that some sentences had more than one stock ticker, so we looped again to remove. The next step was to join the string text back again, we renamed the columns and added a new column to train-df with the number of words.

Stemming and Lemmatization: Stemming consists in a process where it is possible to remove the last few characters of a given word to get a shorter form and this form can be without meaning while the Lemmatization process turns the words to their root word, so we applied first the preprocessing steps, lowercase text, removing numerical data and punctuation and stop words. After the text cleaning we compared the text before and after for validation. Furthermore, we created a function to update the data frame and established a new column with the cleaned text (**SEE Figure 5**).



	text_backup	label	inicial_word_count	text	word_count	Cleaned Text
0	$BYND - JPMorgan reels in expectations on Beyo...	0	10	- JPMorgan reels in expectations on Beyond Meat	8	jpmorgan reel expectation beyond meat
1	*CCL*RCL - Nomura points to bookings weakness...	0	14	- Nomura points to bookings weakness at Carniv...	11	nomura point booking weakness carnival royal c...
2	$CX - Cemex cut at Credit Suisse, J.P. Morgan ...	0	14	- Cemex cut at Credit Suisse, J.P. Morgan on w...	12	cemex cut credit suisse j p morgan weak buildi...
3	$ESS: BTIG Research cuts to Neutral https://t....	0	7	BTIG Research cuts to Neutral	5	btig research cut neutral
4	$FNKO - Funko slides after Piper Jaffray PT cu...	0	10	- Funko slides after Piper Jaffray PT cut	8	funko slide piper jaffray pt cut
...	...	...	...	...	...	...
9538	The Week's Gainers and Losers on the Stoxx Eur...	2	16	The Week's Gainers and Losers on the Stoxx Eur...	15	week gainer loser stoxx europe dec economy mar...
9539	Tupperware Brands among consumer gainers; Unil...	2	9	Tupperware Brands among consumer gainers; Unil...	9	tupperware brand among consumer gainer unileve...
9540	vTv Therapeutics leads healthcare gainers; Myo...	2	11	vTv Therapeutics leads healthcare gainers; Myo...	11	vtv therapeutic lead healthcare gainer myomo b...
9541	WORK, XPO, PYX and AMKR among after hour movers	2	9	WORK, XPO, PYX and AMKR among after hour movers	9	work xpo pyx amkr among hour mover
9542	YNDX, I, QD and OESX among tech movers	2	8	YNDX, I, QD and OESX among tech movers	8	yndx qd oesx among tech mover

9543 rows × 6 columns

Figure 5: Cleaned Text Column

FEATURE ENGINEERING

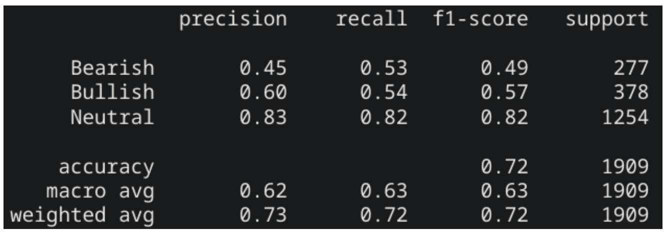
TF-IDF (Term Frequency and Inverse Document Frequency)**:** it is a sparse weighting which weights each cell, which means that the meaning of the word is given through a function that counts the nearby words. In this technique we tried with several ngram_range and the best result was with (1,2).
Word Embeddings: is a process where it is possible to extract a semantic relationship between words from their distribution in texts. We used this function to get embeddings for each word based on word_to_vec_list while establishing the LSTM Model.

4. Classification Models

Gaussian Naive Bayes (GNB) is an algorithm used for classification. The GNB calculates the probability of each class given the feature values and selects the class with the highest probability as the prediction. We decided to use it because it is fast for large datasets and is commonly used for text classification (for example detecting spam emails). Because of GNB characteristics we decided to apply it in our work.
The **KNN Model** is a ML algorithm that makes the predictions by finding the k-nearest neighbors in a class of data in a training set and rearranging them within the class label to a new data point. We decided to use it because it is simple to understand (and interpret), it doesn't make any assumptions about the underlying data distribution and has a high accuracy, so it is a reliable model to use.
The **MLP** (Multilayer Perceptron) is an algorithm type of neural network typically used for NLP (Natural Language Processing), such as text classification, language modeling and machine translation and is well known for performing well when combined with other techniques like word embeddings and regularization. An MLP algorithm works in a multilayer of interconnected nodes or neurons, so each layer processes and transforms the input data in a nonlinear way. This model is known for being good for pattern recognition, and since it is used for text we decided to apply to see how it performs compared to others.
The **LSTM** (Long Short-Term Memory Neural Networks) is a type of recurrent neural network and is also commonly used in NLP because it is well-suited for sentiment analysis and language modeling and it is able to handle and remember sequential data and avoid the long-term dependency problem. This model uses a memory cell and three gates to forget, store and output information. It is well known because it can handle sequential data and has the ability to remember past inputs. We decided to use it to compare with other models.

5. Evaluation and Results

Before evaluating a Model's Performance, it is fundamental to understand the meaning of each indicator and for that reason we decided to describe each measure in a brief summary: **Accuracy** gives the proportion of correct predictions made by the model on a given data set and usually it is the first metric to measure performance. **Precision** consists in a proportion of true positive predictions over all positive predictions realized by the model, this metric is a good indicator to use when the cost of the false positive is high (example the medical diagnosis). **Recall** also gives the proportion of true positive predictions over all actual positive instances in the data set and concerns the opposite when the cost of a false negative is high (example fraud detection). **F1 Score** represents the harmonic mean of precision and recall, and it is very important to look at when we need to balance precision and recall. Our evaluation priority was to check first the Precision, then Recall, Accuracy and F1-Score. In this project we did four classification models and after getting the results we started the comparison between models. First the **Gaussian Naive Bayes** (GNB) had some interesting results (**SEE Figure 6**) because it got the lowest precision in Bearish=0.45, but the Recall=0.53 on the other hand was the highest value for Bearish between the four models results. While looking at the Bullish and Neutral that were more in tune with the other results. The accuracy was the lowest=0.72. The confusion matrix (**SEE Figure 7**) was important for the visualization of the predictions and the target values distribution, in the GNB the values showed some signaling towards the Neutral-Neutral=1023, Bullish-Bullish=206 and Bearish-Bearish=147.



```
              precision    recall  f1-score   support

     Bearish       0.45      0.53      0.49       277
     Bullish       0.60      0.54      0.57       378
     Neutral       0.83      0.82      0.82      1254

    accuracy                           0.72      1909
   macro avg       0.62      0.63      0.63      1909
weighted avg       0.73      0.72      0.72      1909
```

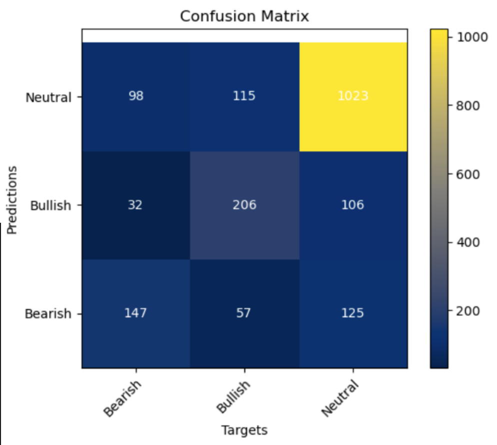
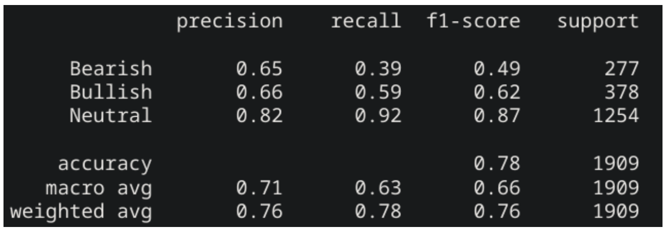


Figure 6: GNB Classification Report

Figure 7: GNB Confusion Matrix



```
              precision    recall  f1-score   support

     Bearish       0.65      0.39      0.49       277
     Bullish       0.66      0.59      0.62       378
     Neutral       0.82      0.92      0.87      1254

    accuracy                           0.78      1909
   macro avg       0.71      0.63      0.66      1909
weighted avg       0.76      0.78      0.76      1909
```

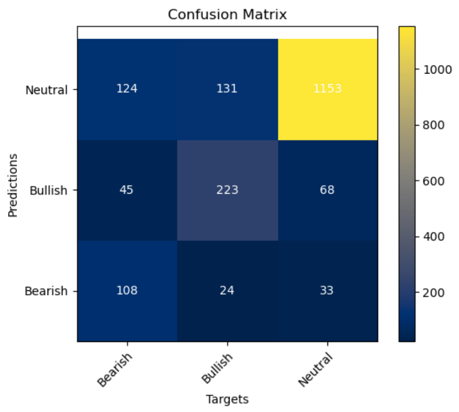
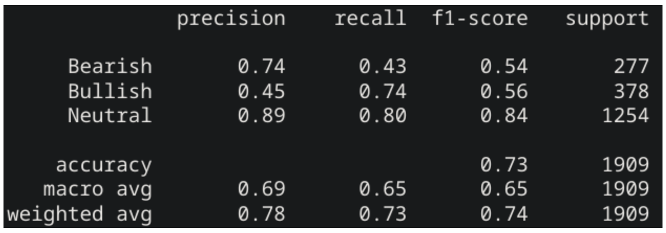


Figure 8: KNN Classification Report

Figure 9: KNN Confusion Matrix

The **KNN (SEE Figure 8)** was the Model that had the best overall precision and accuracy=0.78. The confusion matrix distribution (**SEE Figure 9**) was Neutral-Neutral=1153, Bullish-Bullish=223 and Bullish-Neutral=131. The **MLP** model (**SEE Figure 10**) had the lowest precision=0.45 in Bullish but the highest value in recall=0.74. Although the accuracy was the second lowest when comparing with the other models. The confusion matrix (**SEE Figure 11**) the distribution was Neutral-Neutral=1003, Bullish-Bullish=279 and Bullish-Neutral=233. For the last model the **LSTM** (**SEE Figure 12**) the model in the precision showed some evidence that in the overall was good but the recall was low for Bearish=0.36 and Bullish=0.50 when comparing with the other models. Even though for Neutral values, the recall had the highest=0.90 value. The confusion matrix (**SEE Figure 13**) distribution was Neutral-Neutral=1128, Bullish-Bullish=188 and Bullish-Neutral=165.



```
              precision    recall  f1-score   support

     Bearish       0.74      0.43      0.54       277
     Bullish       0.45      0.74      0.56       378
     Neutral       0.89      0.80      0.84      1254

    accuracy                           0.73      1909
   macro avg       0.69      0.65      0.65      1909
weighted avg       0.78      0.73      0.74      1909
```

Figure 10: MLP Classification Report

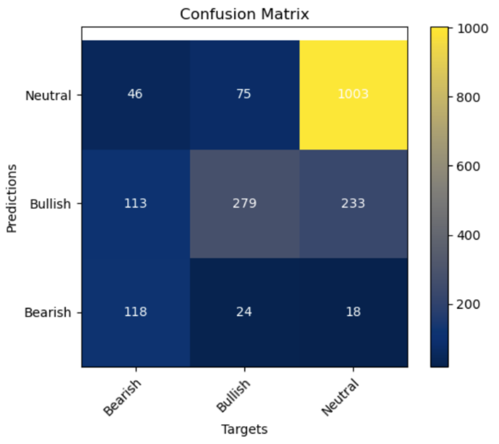
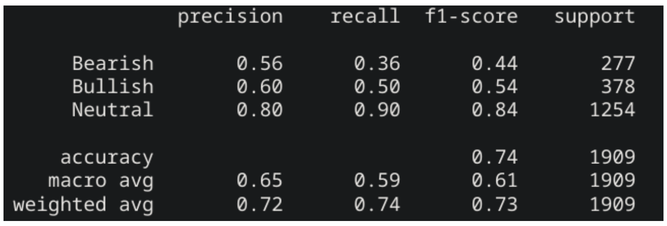


Figure 11: MLP Confusion Matrix



```
              precision    recall  f1-score   support

     Bearish       0.56      0.36      0.44       277
     Bullish       0.60      0.50      0.54       378
     Neutral       0.80      0.90      0.84      1254

    accuracy                           0.74      1909
   macro avg       0.65      0.59      0.61      1909
weighted avg       0.72      0.74      0.73      1909
```

Figure 12: LSTM Classification Report

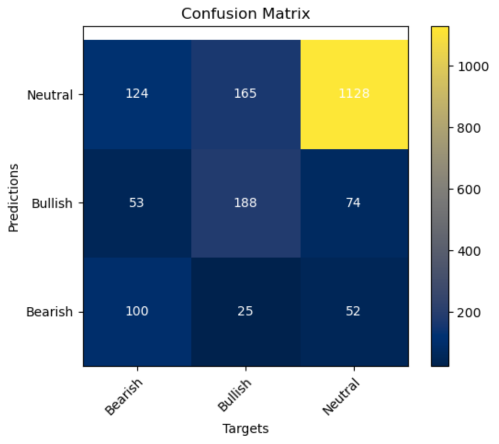


Figure 13: LSTM Confusion Matrix

CONCLUSION

The overall results showed some evidence that all four models were in general good because their accuracy was between 0,72 and 0,78. After analyzing and comparing the results between the four models our choice was the KNN Model because of the overall precision, recall and accuracy=0.78.