UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Market Colors - Financial tweets sentiment analysis

Ruonan GONG
Charaf ZGIOUAR
Hugo MICHEL

Professor: Bertrand HASSANI
-
M2 Finance Technology Data

Université Paris 1 Panthéon-Sorbonne
Paris, France, Winter 2024

Contents

1 Introduction

In the intricate tapestry of financial markets, understanding market price movements is akin to deciphering a complex puzzle influenced by myriad factors, both tangible and intangible. Among these, sentiments often manifested and disseminated through platforms like Twitter play a pivotal role in shaping investor perceptions, behaviors, and, by extension, market dynamics. The ability to discern these sentiments not only offers a window into the collective behaviour of market participants but also provides invaluable insights that can inform strategic investment decisions.

Financial analysis, enriched by sentiment insights, serves as a compass for investors operate in the volatile stock markets. When investors can gauge the prevailing sentiments surrounding specific stocks or sectors, they gain a nuanced understanding of market sentiment trends, potential shifts, and underlying sentiments driving price movements. For instance, positive sentiments surrounding a particular stock—indicative of optimism, confidence, or favorable market conditions—often correlate with increased buying activities, potentially driving up the stock price. Conversely, negative sentiments, characterized by skepticism, apprehension, or adverse news, can trigger selling pressures, leading to price declines.

Moreover, sentiment analysis transcends mere price movements, offering a holistic perspective that incorporates qualitative factors. By analyzing sentiments expressed across financial tweets, investors can identify emerging trends, sentiment-driven catalysts, or sentiment anomalies that may deviate from fundamental valuation metrics. Such insights empower investors to make informed decisions, whether it involves capitalizing on market momentum, hedging against potential downturns, or identifying undervalued assets poised for growth.

Therefore, our study's significance lies in its exploration of sentiment analysis methodologies, aiming to refine the tools investors rely upon to distill actionable insights from the vast reservoir of financial tweets. By comparing traditional sentiment models based on lexicons like SentiWordNet to rules-based models like VADER and advanced deep learning models like BERT, we seek to elucidate which methodologies offer the most accurate, timely, and actionable sentiment insights. In doing so, our research endeavors to bridge the gap between sentiment analysis and investment decision-making, highlighting the pivotal role sentiments play in shaping market perceptions, behaviors, and outcomes.

2 Problematic

In light of the last NLP advancements, our study focus around a pivotal problematic: ***How does the accuracy of traditional sentiment models based on lexicons compare to rules-based and advanced deep learning models in analyzing sentiment from financial tweets?***. Furthermore, what ramifications do these comparative insights hold for stakeholders navigating the volatile terrains of financial markets? Through this lens, our research aims to elucidate the evolving landscape of sentiment analysis, juxtaposing foundational methodologies against cutting-edge innovations to distill actionable insights for the financial ecosystem.

3 Data

3.1 Data Description

The data we use comes from HuggingFace link The dataset holds 11 932 documents annotated with 3 labels:

- Bearish = 0
- Bullish = 1
- Neutral = 2

The data was collected using the Twitter API. The current dataset supports the multi-class classification task.

3.2 Data Analysis

The dataset is split on two set:

- **Train set:** 9 543 tweets
- **Test set:** 2 388 tweets

We check for NaN values in the dataframe and we dont find any NaN values.

Then, we check the distribution of the labels for the Train set and Test set and check if out dataset is well balanced. Indeed, in the realm of machine learning, the distribution of labels within both the training set and the test set plays a critical role in determining the model's performance, generalizability, and reliability. Specifically, when dealing with imbalanced datasets—where one class significantly outnumbers the other(s)—a host of challenges and implications arise that necessitate careful consideration and remedial actions.

	text	label
0	$BYND - JPMorgan reels in expectations on Beyo...	0
1	$CCL $RCL - Nomura points to bookings weakness...	0
2	$CX - Cemex cut at Credit Suisse, J.P. Morgan ...	0
3	$ESS: BTIG Research cuts to Neutral https://t....	0
4	$FNKO - Funko slides after Piper Jaffray PT cu...	0
...	...	...
9538	The Week's Gainers and Losers on the Stoxx Eur...	2
9539	Tupperware Brands among consumer gainers; Unil...	2
9540	vTv Therapeutics leads healthcare gainers; Myo...	2
9541	WORK, XPO, PYX and AMKR among after hour movers	2
9542	YNDX, I, QD and OESX among tech movers	2

9543 rows × 2 columns

(a) Sample of Train set

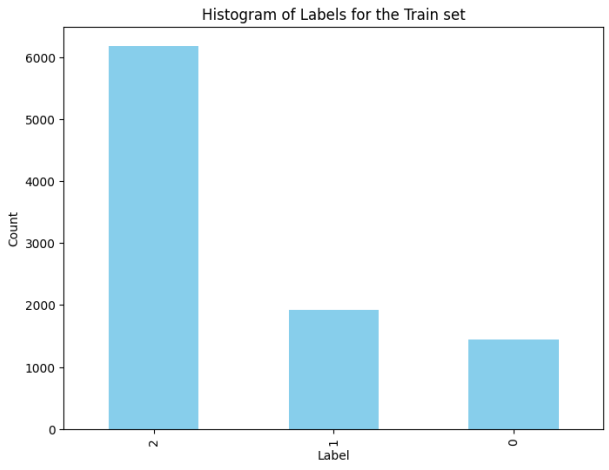
	text	label
0	$ALLY - Ally Financial pulls outlook https://t...	0
1	$DELL $HPE - Dell, HPE targets trimmed on comp...	0
2	$PRTY - Moody's turns negative on Party City h...	0
3	$SAN: Deutsche Bank cuts to Hold	0
4	$SITC: Compass Point cuts to Sell	0
...	...	...
2383	Stocks making the biggest moves midday: TD Ame...	2
2384	Stocks making the biggest moves premarket: Fit...	2
2385	Stocks making the biggest moves premarket: Hom...	2
2386	Stocks making the biggest moves premarket: TD ...	2
2387	TCO, NNVC, GPOR and JE among midday movers	2

2388 rows × 2 columns

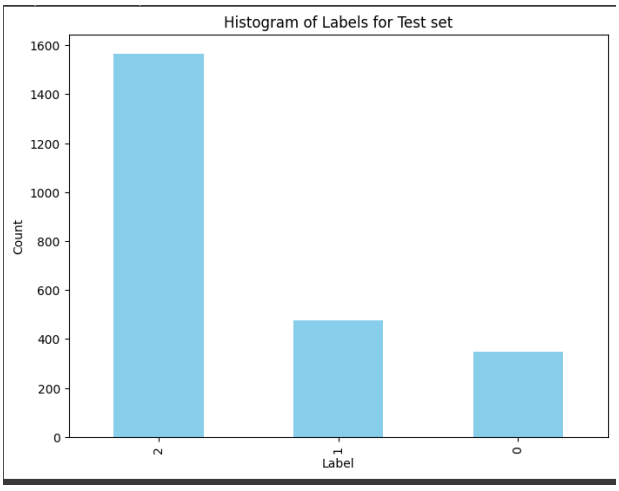
(b) Sample of Test set

Figure 3.1: Sample of Train and Test set

- **Performance Bias:** In an imbalanced dataset, models tend to favor the majority class due to the skewed distribution of data. Consequently, the model may achieve high accuracy simply by predicting the majority class, thereby overlooking the minority class. While accuracy might appear satisfactory, the model's real-world applicability and effectiveness in capturing minority class instances diminish significantly.
- **Misleading Evaluation Metrics:** Traditional performance metrics like accuracy become misleading in imbalanced settings. For instance, a model that predicts the majority class for all instances might still achieve high accuracy but fail miserably in capturing the minority class. Hence, relying solely on accuracy without considering other metrics like precision, recall, F1-score, or the area under the ROC curve can yield a distorted view of the model's true performance.
- **Costly Errors:** In scenarios where the minority class represents critical or rare events (e.g., fraudulent transactions, rare diseases), misclassifying these instances can have severe consequences. An imbalanced dataset exacerbates this risk, as the model's propensity to overlook the minority class can result in costly false negatives.
- **Generalizability Concerns:** Imbalanced training datasets can lead to models that generalize poorly to unseen data, as they have not been adequately trained to recognize and differentiate minority class instances. This lack of generalizability undermines the model's efficacy in real-world applications, where capturing all classes' nuances is important.



(a) Distribution of labels of the Train set

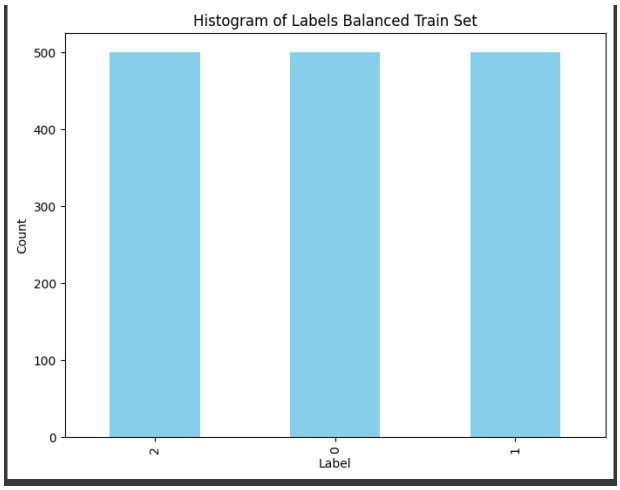


(b) Distribution of labels of the Test set

Figure 3.2: Distribution of labels dataset

Since we don't need a lot of data to train our dataset, we haven't opted for oversampling or undersampling techniques such as the SMOTE approach to balance our training dataset. on the other hand. However, to balanced the dataset we simple select randomly 500 tweets for each labels and shuffle then balanced dataset. The results is as follow :

Figure 3.3: Distribution of labels of the balanced Train set



4 Challenger Model

4.1 Rule-Based Appoach

4.1.1 Naive Model {SentiWordNet}

The foundation of our exploration rests upon the **SentiWordNet** dictionary—a lexical resource meticulously crafted to assign sentiment scores to individual words. While such dictionary-based methods have historically provided a rudimentary yet effective means of sentiment classification, their efficacy in capturing the nuanced and evolving language of financial tweets remains a subject of debate.

We propose to apply a naive model to predict the sentiment score of set of tweets. To do so, we use a dictionary **SentiWordNet**.

We then improve our model by taking into account of potential emoji and negation and booster words in the tweets.

Furthermore, given that, in our case the Naive model is rules-based algorithm which use a dictionary to compute the sentiment score, we don't have to train the model. Hence, we directly evaluated the model on the Test set.

In terms of data preprocessing, we have to clean the tweet by removing some useless piece of texts which don't give any information about the sentiment of a given tweet such as urls and hashtags. To do so, we operate as follow:

- Segment tweet into set of tokens thanks to **TweetTokenizer** from NLTK library
- Remove urls
- Clean up characters inherent in the structure of a tweet (hashtags!)
- Correct abbreviations and language specificities in tweets using the DicoSlang dictionary (SlangLookupTable.txt file)

Figure 4.1: Sample of the Slang Dictionary

	original	new
0	121	one to one
1	a/s/l	age, sex, location
2	adn	any day now
3	afaik	as far as I know
4	afk	away from keyboard
...	...	...
85	wibni	wouldn't it be nice if
86	wtf	what the fuck
87	wtg	way to go
88	wtgp	want to go private
89	ym	young man

90 rows × 2 columns

Example of tweet before and after the cleansing function

Original tweet test

121 @stellargirl #looser adn [link] [1 comment] @author #looser This is ! a text with! a URL https://www.java2blog.com/ to remove + :) ok =) =’(/ [link] #looser [1 comment] @author #looser.

Cleaned tweet

[’one’, ’to’, ’one’, ’looser’, ’any’, ’day’, ’now’, ’[’, ’link’, ’]’, ’[’, ’1’, ’comment’, ’]’, ’looser’, ’This’, ’is’, ’!’, ’a’, ’text’, ’with’, ’!’, ’a’, ’URL’, ’to’, ’remove’, ’+’, ’:)’, ’ok’, ’=)’, "=’(", ’/’, ’looser’, ’[’, ’link’, ’]’, ’[’, ’1’, ’comment’, ’]’, ’looser’, ’.’]

We can see that url and "#" symbols are removed.

The last step of the preprocessing is the POS-Tagging process. Indeed, knowing the grammatical category (or in this case the ’part of speech’ or ’POS’) of words will help us to carry out the sentiment analysis. We implement a function capable of determining the part of speech of each word in the tweet using the following command from the nltk library

Example of POS-Tagging on a tweet

Original cleaned tweet: *$ALLY - Ally Financial pulls outlook*

POS-Tagging

[(’$’, ’.’), (’ALLY’, ’NOUN’), (’-’, ’.’), (’Ally’, ’NOUN’), (’Financial’, ’NOUN’), (’pulls’, ’NOUN’), (’outlook’, ’NOUN’)]

Then we compute for each tweet of the sum of the positive and negative scores of the tweet’s synsets (1st synset) and compare the sum of positive and negative scores for each tweet to decide which class to associate with the tweet.

Based on the POS-Tagging, the goal of this first approach is retrieve only words corresponding to adjectives (**ADJ**), nouns (*NOUN*), adverbs (*ADV*) and verbs (*VERB*), and access the scores (positive and negative) of the first synset.

This model is our baseline model.

Note: The calculation of the positive and negative scores of the tweet synsets are automatically based on the 1st synset of the word in question.

Example of the use SentiWordNet dictionary

Figure 4.2: SentiWordNet on "dog" word

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('dog')
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'),
Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'),
Synset('chase.v.01')]
```

To implement the naive model I apply the following step:

1. Browse the tag list and focus on words with the tag of interest (ADJ, NOUN,...)
2. Search for the given word in the dictionary (we take the first synset of the dictionary)
3. Check that the given word contains an accepted tag and we check that the word is contained in the dictionary
4. Search for the first synset that matches the given word
5. If the word matches the first synset, we calculate the scores
6. Sum the scores of the different words in the given tweet
7. Define the class by majority vote

The output of the model is:

Figure 4.3: Naive Model {SentiWordNet}

	true_polarity	our_polarity
0	bearish	neutral
1	bearish	bullish
2	bearish	bearish
3	bearish	bearish
4	bearish	neutral
...	...	...
2383	neutral	neutral
2384	neutral	neutral
2385	neutral	neutral
2386	neutral	neutral
2387	neutral	neutral

2388 rows × 2 columns

4.1.2 Naive Model {SentiWordNet + Negation Words + Booster Words}

We would like to improve the first naive model by taking into account negation and booster words. We think that handling negative and booster words can refine the accuracy of the sentiment score.

Indeed, in the context of sentiment analysis of financial tweets, considering negative words and booster words offers several advantages:

- **Contextual Understanding:** Financial tweets often contain specific jargon, idioms, or context-specific language. Incorporating negative words helps in capturing the nuances of sentiment, especially in situations where a statement might appear positive at first glance but contains subtle negative implications.
- **Enhanced Accuracy:** Booster words, such as "highly," "very," "extremely," etc., can intensify the sentiment expressed in a sentence. By considering these words, sentiment analysis models can more accurately gauge the strength or intensity of the sentiment, rather than just classifying it as positive, negative, or neutral.
- **Improved Decision Making:** For traders, investors, and financial institutions, understanding the sentiment accurately is crucial for decision-making. By considering negative words, analysts can identify potential risks, concerns, or negative perceptions that might impact investment decisions or market strategies.
- **Risk Management:** In the financial world, risk management is important. Analyzing sentiment with a focus on negative words allows for early detection of negative sentiments or trends that could indicate potential risks, market downturns, or unfavorable events related to specific financial assets or markets.
- **Holistic Sentiment Analysis:** By incorporating both negative and booster words, sentiment analysis becomes more holistic. It not only identifies the overall sentiment (positive, negative, neutral) but also provides insights into the strength, intensity, and nuances of the sentiment expressed in financial tweets.
- **Competitive Advantage:** For financial institutions, hedge funds, or individual traders using sentiment analysis, incorporating negative words and booster words can provide a competitive advantage. It enables them to extract deeper insights from financial tweets, anticipate market movements, or identify emerging trends before they become widely recognized.

In addition to calculating the score for each word, based on the *SentiWordNet* dictionary, this second model takes into account negative words and booster words to calculate the overall sentiment score for the tweet by referring respectively to two dictionaries *NegatingWordList* and *BoosterWordList*.

Example: I don't like cat don't is a negation and the score of the following word likes are: {scorePos : 0.125, scoreNeg : 0.0} then the global positive score of the sentence is : 0.0 and the global negative score of the sentence becomes : 0.125

Secondly, the idea is to take the rules we implemented previously for the first version of calculating the sentiment score (Detection algorithm: use of the SentiWordNet dictionary) of a tweet and make them more complex by adding the rules to take into account of negative and positive words.

Since we're keeping the rules we implemented previously, we're concentrating here only on words whose tag is ["NOUN", "ADV", "VERB", "ADJ"].

Basically, for each word, the algorithm perform the following additional operations:

- multiply the negative and positive scores associated with the word by 2 if the preceding word is a booster word
- use only the word's negative score for the tweet's overall positive score and the word's positive score for the tweet's overall negative score if the preceding word is a negation.

	0
0	aren't
1	arent
2	can't
3	cannot
4	cant
5	don't
6	dont
7	isn't
8	isnt
9	never
10	not
11	won't
12	wont
13	wouldn't
14	wouldnt

(a) NegatingWordList dictionary sample

	0	1
0	absolutely	1
1	definitely	1
2	extremely	2
3	fuckin	2
4	fucking	2
5	hugely	2
6	incredibly	2
7	just	-1
8	overwhelmingly	2
9	so	0
10	some	-1
11	sum	-1
12	very	1

(b) BoosterWordList dictionary sample

Figure 4.4: Booster and Negation words dictionnaries

The output is as follow:

Figure 4.5: Prediction sample of Naive Model {SentiWordNet + Negation Words + Booster Words}

	true_polarity	our_polarity_2
0	bearish	neutral
1	bearish	bullish
2	bearish	bearish
3	bearish	bearish
4	bearish	neutral
...	...	...
2383	neutral	neutral
2384	neutral	neutral
2385	neutral	neutral
2386	neutral	neutral
2387	neutral	neutral

2388 rows × 2 columns

4.1.3 Naive Model {SentiWordNet + Negation Words + Booster Words + Emoji}

We would like to further improve the performance of ou Naive Model by taking into account emoji in a tweet. Using emojis in sentiment analysis, especially in the context of financial tweets, can provide additional layers of context and meaning to the analysis. Emojis help capture the nuances of sentiment that might be challenging to discern from text alone. For example, a tweet might contain positive words, but a negative emoji accompanying it could indicate sarcasm or a contrasting sentiment. By analyzing both text and emojis, sentiment analysis models can provide a more nuanced understanding of the sentiment expressed.

In other words, given that in the financial world, accuracy is crucial, understanding the sentiment correctly is vital. Emojis can clarify whether a statement is meant to be taken literally, sarcastically, optimistically, or pessimistically. Furthermore, considering emojis, sentiment analysis models can improve their accuracy in classifying sentiments. Emojis serve as additional data points that offer insights into the emotional tone, intensity, and context of the sentiment expressed in a tweet. Emojis provide contextual clarity, helping to disambiguate the sentiment of a tweet.

Sample of Emoji dictionary

Figure 4.6: Emoji dictionary sample

	emoticone	score
0	%-(	-1
1	%-)	1
2	(-:	1
3	(:	1
4	(^ ^)	1
...	...	...
110	xP	-1
111	|8C	-1
112	|8c	-1
113	|D	1
114	}:)	1

115 rows × 2 columns

We're keeping the same rules as the previous version of our model, but in addition to these rules we're adding a rule specific to handle emoji contained in a tweet.

To do so, positive emoticons encountered increase the value of the tweet's positive score by +1, while negative emoticons increase the value of the tweet's negative score by −1.

The output is as follow:

Figure 4.7: Prediction sample of Naive Model {SentiWordNet + Negation Words + Booster Words}

	true_polarity	our_polarity_4
0	bearish	neutral
1	bearish	bullish
2	bearish	bearish
3	bearish	bearish
4	bearish	neutral
...	...	...
2383	neutral	neutral
2384	neutral	neutral
2385	neutral	neutral
2386	neutral	neutral
2387	neutral	neutral

2388 rows × 2 columns

4.1.4 VADER Algorithm

In addition to the naive model, as a comparison we applied the commonly used VADER model. VADER (Valence Aware Dictionary and sEntiment Reasoner) is a pre-built, lexicon and rule-based sentiment analysis tool designed for analyzing text data in natural language. It is specifically crafted to handle sentiments expressed in social media texts, as it incorporates features like handling of emoticons, capitalization, and context-based sentiment scoring.

Here are some key aspects of the VADER library:

- **Lexicon and Rule-Based Approach:** VADER uses a combination of a sentiment lexicon (a predefined list of words and their associated sentiment scores) and a set of grammatical and syntactical rules to determine the sentiment of a piece of text.
- **Valence Scores:** The lexicon assigns polarity scores to words, indicating the positive or negative sentiment conveyed by each word. These scores range from -1 to 1, where -1 represents extreme negativity, 1 represents extreme positivity, and 0 represents neutrality.
- **Emoticon Handling:** VADER is designed to handle sentiments expressed through emoticons, making it suitable for analyzing text data from social media platforms where emoticons are commonly used to convey emotions.
- **Capitalization and Punctuation:** VADER takes into account the intensity of sentiment by considering the impact of capitalization and punctuation in the text.
- **Contextual Valence Shifting:** VADER can recognize and handle some degree of valence shifting, where the sentiment of a word changes based on the context in which it is used.

- **Sentiment Intensity:** VADER provides a compound score that represents the overall sentiment intensity of a piece of text. This score considers both the individual word scores and their arrangement in the text.

In summary VADER is a lexicon and rule-based sentiment analysis model designed for social media texts and short texts. Its vocabulary contains internet informal expressions, short forms and emoticons, while it considers punctuation, capitalisation and degree modifiers to understand the direction of the sentiment and its intensity. This means that it does not only judge sentiment based on the frequency of occurrence of the text, but also incorporates the context and the specific usage of the words, making it very effective when dealing with informal, abbreviated and graphic symbolised texts.

Like the naive model, VADER is a rule-based algorithm that uses a dictionary to compute sentiment scores, so there is still no need to train the model, and we will evaluate the model directly on a test set.

The output is as follow:

Figure 4.8: Prediction sample of VADER Model

	predicted_label	actual_label
0	Neutral	Negative
1	Neutral	Negative
2	Positive	Negative
3	Negative	Negative
4	Negative	Negative
...	...	...
2383	Neutral	Neutral
2384	Neutral	Neutral
2385	Neutral	Neutral
2386	Neutral	Neutral
2387	Neutral	Neutral

[2388 rows x 2 columns]

4.2 Machine Learning Approach

Our hypothesis is that a machine learning approach can improve prediction performance because it can learn from a specific vocabulary, unlike a rule-based model.

4.2.1 {CountVectorizer, TF-IDF, MultinomialNB}

For this study we seek to evaluate a purely supervised machine learning approach for the sentiment analysis task. To do this, we use a series of statistical tools adapted to the NLP domain in order to calculate the sentiment score. We chose to apply the following pipeline CountVectorizer, TF-IDF, MultinomialNB. We use the Multinomial Naive Bayes model for sentiment analysis, by leveraging the frequency distribution of words to classify texts providing by the CountVectorizer and TF-IDF algorithm into different sentiment categories.

To do so, we implement the following pipeline:

1/ Pre-processing: Stemmatization step

In order to simplify the task of our model, we aim to reduce the size of our vocabulary (e.g. Bag Of Words). To do this, we apply a preprocessing step consisting in normalizing the words. This NLP technique is called stemmatization, and reduces words to their base forms, stemming helps in focusing on the core meaning of the text without getting distracted by grammatical variations. By converting different word forms to their common root, stemming can help reduce the dimensionality of the text data. This simplification can be particularly useful in tasks like text classification, where reducing the vocabulary size without losing significant meaning can lead to more efficient and effective models.

To achieve this, we use ***SnowballStemmer*** from the NLTK library. Allows you to go back to the root of a word: in this way, we group different words around the same root, facilitating generalization.

2/ CountVectorizer

CountVectorizer is used to convert a collection of tweets into a matrix of token counts. In simpler terms, it converts a text document into a vector where each element of the vector corresponds to the count of a particular word in the document. This process is also known as "bag-of-words" representation.

Once tokenized, CountVectorizer builds a vocabulary from all the words in the documents. It considers each unique word as a feature. For each document, the vectorizer counts the occurrence of each word (from the vocabulary) and places that count in the corresponding position of the vector. If a word doesn't appear in a particular document, its count will be zero.

CountVectorizer is used in various NLP tasks like text classification, clustering, sentiment analysis, and more. By converting text data into a numerical format (vector representation), it becomes easier to apply machine learning algorithms that require numerical input.

To sum up CountVectorizer is a powerful tool for converting text data into a format that can be fed into machine learning models. It facilitates the transformation of textual information into a structured numerical representation suitable for machine learning tasks.

3/ TF-IDF

The TF-IDF algorithm is seen as an advance on the word probability method. Clearly, the algorithm assigns low weights to words that appear very frequently in most documents because they are considered to be words with little informative value. Conversely, it gives a higher weight to terms that appear infrequently in the corpus. Below is the word weighting formula:

This is the product of the frequency of the term (TF) and its inverse frequency in the documents (IDF). This method is usually used to extract the importance of a term i in a document j relative to the rest of the corpus, from a matrix of occurrences $words \times documents$. Thus, for a matrix $\mathbf{T}$ of $|V|$ terms and D documents:

$$\mathrm{TF}(T, w, d) = \frac{T_{w,d}}{\sum_{w'=1}^{|V|} T_{w',d}}$$

$$\mathrm{IDF}(T, w) = \log\left(\frac{D}{|\{d : T_{w,d} > 0\}|}\right)$$

$$\text{TF-IDF}(T, w, d) = \mathrm{TF}(X, w, d) \cdot \mathrm{IDF}(T, w)$$

It can be adapted to our use case by considering that the context of the second word is the document. However, TF-IDF is generally better suited to sparse matrices, since this measure will penalise terms that appear in a large proportion of documents.

4/ MultinomialNB

For the classification task, we use Multinomial Naive Bayes (MNB) which is a probabilistic learning method primarily used for text classification tasks in Natural Language Processing (NLP), including sentiment analysis. Multinomial Naive Bayes is based on Bayes' theorem, which calculates the probability of a particular class given some features. In sentiment analysis, the goal is to determine the sentiment (e.g., positive, negative, neutral) of a text based on its content. The "Naive" in Naive Bayes comes from the assumption that features (in this case, words) are conditionally independent given the class label. This assumption simplifies the model and makes computations more tractable, although it might not hold true in reality for all datasets.

In NLP, text is often represented as a bag-of-words, where each document (or sentence) is treated as an unordered set of words, disregarding grammar and word order but retaining frequency information. The Multinomial Naive Bayes model assumes that the frequency of words (or features) follows a multinomial distribution.

For sentiment analysis, each word or term becomes a feature. The MNB model calculates the probability of each word given a particular sentiment class (e.g., $P(word|positive)$, $P(word|negative)$). This is done using a training dataset where each document is labeled with a sentiment class.

Once the model calculates the probabilities for each word given a sentiment class, it uses these probabilities to predict the sentiment of new, unseen texts. Specifically, for a given text, the model calculates the likelihood of it belonging to each sentiment class based on the words it contains and then chooses the sentiment class with the highest likelihood as its prediction.

We use scikit-learn's 'MultinomialNB', an implementation of the naive Bayesian model. Here, the naive hypothesis is that the different variables (words) in a review are independent of each other.

4.3 Models Evaluation

To discuss the results from a data science point of view we need to compare the prediction performance of the models against the ground-truth (i.e label of the dataset).

To do so, as we are in a multi-class classification task ($0 = bearish$, $1 = bullish$, $0 = neutral$) we will use the confusion matrix and classification report (Accuracy, Recall, F1-Score) to assess the performance of our models and select the best model.

4.3.1 Naive Models

Overall, the performance of the 3 models is similar (see Appendix A1). Taking into account negative words, booster words and emoji does not seem to improve the prediction performance of the naive model. As far as emoji are concerned, this can be explained by the fact that financial tweets are often professional and very rarely contain emoji. On the other hand, when it comes to negative and positive words, the result is more surprising, as we expected the model to be much more accurate, but this turns out not to be the case. This may be due to the quality of the dictionaries of negative words and booster words that we use, which may not be complete and exhaustive enough and, above all, are not specific to financial vocabulary. Finally, the 1st iteration (i.e. baseline model) has an accuracy of 0.42. This is not a very good score because the model works less well than the random model.

4.3.2 VADER Algorithm

In the classification report of the VADER model (see Appendix A2), we can see that the sentiment prediction for financial news performs slightly better in the "Neutral" category compared to the other categories, but the F1 score is only 0.59, indicating that the model performs poorly in general. Specifically, the "Neutral" category has a recall and precision of 0.50, while the "Bullish" and "Bearish" categories do not perform as well. In particular, the precision of the "Bullish" category is only 0.27, indicating that only a small proportion of outcomes predicted to be "Bullish" are actually "Bullish", which may indicate that the model is not as good as it should be. This may indicate that the model has some difficulty in distinguishing positive financial news.

The results of the confusion matrix show that the VADER model does not perform well in distinguishing between the Neutral and Bullish sentiment categories, as 545 of the results are actually Neutral and 545 are actually Bullish. Neutral" was incorrectly predicted to be "Bullish", indicating that the VADER model is not clear enough to distinguish the boundary between these two categories, resulting in a high misclassification rate. This finding is important for further optimisation of the model and suggests that we need to further improve the model to increase overall classification accuracy.

4.3.3 ML Model: {CountVectorizer, TF-IDF, MultinomialNB}

We can see that in the classification report (see Appendix A3), the machine learning approach produces much better results than the rule-based Naive model. Indeed, the accuracy of the machine learning model is almost 2 times higher than the accuracy of the Naive model. We can affirm the same thing for the F1-Score. In conclusion, we can validate our hypothesis and assert that Machine Learning approaches seem better suited to evaluating the sentiment of financial tweets.

4.4 Models advantages and limitations

Regarding the Naive model, the main advantages of rule-based algorithms using dictionaries provide clear rules and criteria for sentiment scoring. This transparency allows users to understand how sentiments are determined, making it easier to interpret the results and validate the sentiment analysis process. Furthermore, unlike machine learning models that require labeled training data for training and validation, rule-based algorithms are ready to use algorithm and do not necessitate this data-intensive process. This absence of training data minimizes the effort and resources needed to develop and deploy sentiment analysis systems, making it more accessible and cost-effective for certain applications. Moreover, rule-based algorithms typically have lower computational requirements compared to complex machine learning models. This lower computational overhead makes rule-based sentiment analysis systems more lightweight, faster, and less resource-intensive, making them suitable for deployment in environments with limited computational resources.

Otherwise, even if, dictionaries in rule-based algorithms can be customized and updated based on specific domains, industries, or contexts, in the most of case the accuracy of these models are subject to some limitations, particularly concerning their accuracy and contextual understanding compared to advanced deep learning models. Indeed, rule-based models often struggle to capture the nuanced and contextual meanings of sentences. Unlike machine learning and deep learning models that can learn from vast amounts of data to understand the intricate relationships between words, phrases, and context, rule-based models rely on predefined rules and dictionaries, limiting their ability to comprehend complex sentences, sarcasm, figurative language, or subtle nuances. For instance, rule-based models may struggle to handle ambiguous or polysemous words, phrases, or expressions that have multiple meanings or interpretations. In addition, rule-based models are typically designed based on specific rules, criteria, or dictionaries, making them less adaptable to new or unseen data outside their predefined scope.

In contrast, machine learning models can generalize from training data to make predictions on unfamiliar or evolving contexts, enhancing their adaptability and performance across diverse datasets Financial tweets can be highly nuanced and context-dependent and Machine learning models can adapt to diverse and evolving data patterns, learning from the variability and complexity inherent in financial language. In contrast, rule-based models may struggle to capture the intricacies of sentiment due to their static nature. Furthermore, Machine learning models have the potential to

generalize better across different financial contexts, market conditions, and user sentiments. By learning from a large corpus of data, they can identify underlying patterns and features that might not be explicitly defined in rule-based systems, leading to more robust sentiment analysis outcomes. One of the main advantages of Machine learning approach is that they can handle large datasets efficiently, adapt to changing market dynamics, and incorporate new information without manual rule adjustments, making them more agile and responsive to real-time sentiment analysis needs.

Overall, the performance of rule-based models heavily depends on the quality, coverage, and accuracy of the rules, dictionaries, or lexicons used. Inaccurate, outdated, or incomplete rules and dictionaries can lead to suboptimal sentiment analysis results, requiring frequent updates and maintenance to ensure relevance and effectiveness. In other words, the inability to capture context, semantics, and relationships between words can hinder their accuracy and reliability in analyzing sentiments accurately, especially in challenging or diverse datasets.

In summary, while rule-based models have their merits, especially in scenarios with well-defined rules, structured and unlabeled data, machine learning approaches generally offer greater flexibility, adaptability, and accuracy for evaluating the sentiment of financial tweets. By leveraging advanced algorithms, feature learning capabilities, and continuous learning mechanisms, machine learning models can provide more nuanced, reliable, and actionable insights for financial market participants and analysts.

5 State of the Art Model

5.1 Models Description

After a quick review of the scientific literature, we can learn that the state-of-the-art models for sentiment analysis are those based on a transform architecture such as the BERT model. BERT is built upon the Transformer architecture, which was introduced in the paper ***Attention Is All You Need*** by Vaswani et al. (2017). BERT is a significant model in NLP based on the Transformer architecture.

Therefore, we propose to use a pre-trained then fine-tuned BERT model on a corpus of financial news to improve the computation of the sentiment score associated with each ECB statement. To do so we use the 'mrm8488/distilroberta-finetuned-financial-news-sentiment-analysis' from 'HuggingFace' library

This model is based BERT (Bidirectional Encoder Representations from Transformers) is a pivotal model in the realm of natural language processing (NLP), and its innovation can be attributed to several key components, with the attention mechanism being a crucial aspect. Let's delve into its significance and the innovations behind it:

1. Pre-training and Fine-tuning: Innovation: BERT introduced a two-step training process that leverages vast amounts of text data. First, the model is pre-trained on a large corpus of text using two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). For example for the first task the model is trained to predict a word in the sentence given the context and meaning of the sentence .After pre-training, the model can be fine-tuned on specific downstream tasks, such as sentiment analysis or question-answering, with smaller, task-specific datasets. In our case the model we use is fine-tuned for the sentiement analysis task on a financial new dataset.

2. Bidirectional Context: Innovation: Unlike previous models that processed text in a left-to-right or right-to-left manner, BERT utilizes a bidirectional approach. This means it considers the context from both directions (before and after a word) when encoding a word's representation. This bidirectional context helps in capturing a deeper understanding of words in their specific contexts.

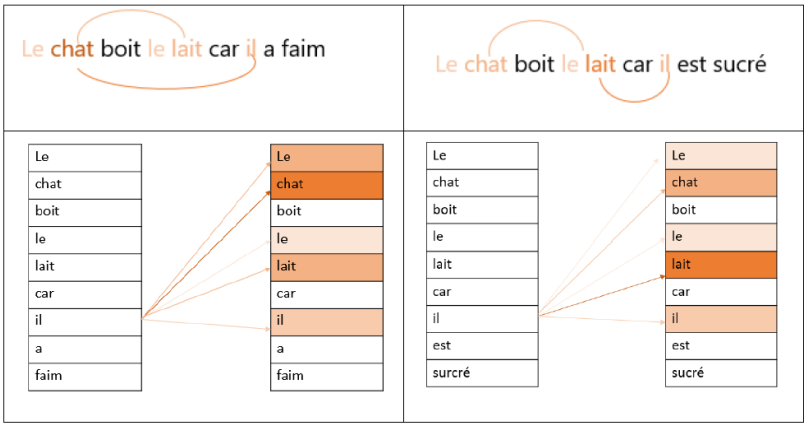
3. Transformer Architecture: Innovation: BERT is built upon the Transformer architecture, which was introduced in the paper "Attention Is All You Need" by Vaswani et al. (2017). The Transformer's core is the attention mechanism, which allows it to weigh the significance of different words in a sentence when processing each word. This attention mechanism replaces the recurrent layers used in previous models and offers more parallelizable computations, making it more efficient and effective for capturing long-range dependencies in text.

Attention Mechanism in BERT

When we watch a football match on television, we instinctively focus our attention on the ball, because our experience has taught us that it is close to the ball that most of the information about the game in progress is to be found. Translation systems and image descriptions can now simulate this kind of mechanism to improve their performance. This is known as the attention mechanism, which tends to resemble the mechanism inspired by the functioning of the cerebral cortex.

Let's illustrate the attention mechanism with an example:

Figure 5.1: Attention Mechanism



Les couleurs plus prononcées représentent une attention plus élevée

The key to the Transformer's performance lies in the use of the attention mechanism. To explain this, let's start with an example:

- The cat drank the milk because it was hungry
- The cat drank the milk because it was sweet

In the first sentence, the word "it" refers to "cat". Conversely, in the second sentence, the word "it" refers to "milk". The role of attention is therefore to examine the sentence selectively, finding the right word associations.

In fact, to explain graphically, attention constructs a kind of heat map that indicates which word in the input sentence the model should focus on when generating each word in the output. By training a Transformer, it learns from the data which word it should pay particular attention to. For example, the encoder communicates information to the decoder such as: "Words 8, 11 and 23 are very important to give the exact meaning of this sentence.

- **Self-Attention:** The attention mechanism in BERT allows the model to weigh the importance of different words (tokens) in a sequence when processing a particular word. This is termed as "self-attention" because it determines how much focus (attention weight) each word should give to every other word in the sequence.
- **Capturing Dependencies:** By employing self-attention, BERT can capture dependencies between words that are far apart in a sentence. For instance, in the sentence "The cat sat on the mat," understanding the relationship between "sat" and "mat" requires considering all the words in between, which self-attention can efficiently capture.
- **Multiple Layers and Heads:** BERT uses multiple attention layers (stacked on top of each other) and multiple attention heads within each layer. This design enables the model to capture various types of relationships in the words in the sentence, enhancing its ability to understand and represent complex linguistic structures and nuances.

In summary, the innovation behind BERT, especially its attention mechanism, revolutionized the field of NLP by enabling models to capture deeper contextual information from text data. By employing bidirectional context and the Transformer architecture's efficiency, BERT set new benchmarks in various NLP tasks, leading to significant advancements and applications in areas like machine translation, question-answering, sentiment analysis, and more.

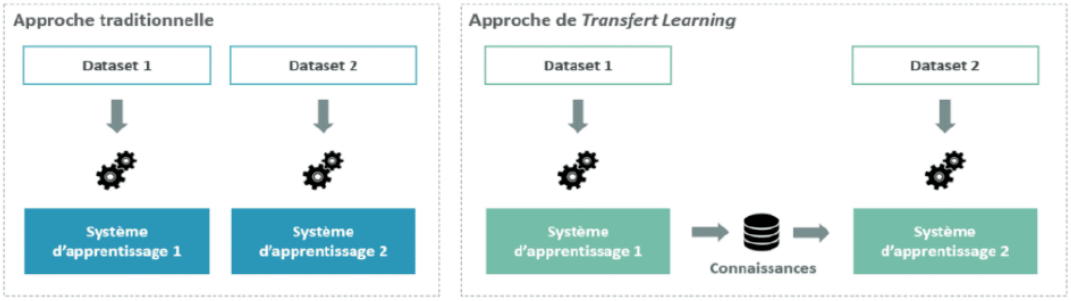
5.1.1 BERT fine-tuned

In order to specialize the BERT model on classification tasks for sentiment analysis of financial tweets, we fine-tuned it to make it perform well on this task. Before explaining the approach we used to train the model, it's worth explaining what lies behind fine-tuning.

Transfer learning occurs when a model developed for one task is reused to work on a second task. Fine-tuning is actually an approach to transfer learning.

In Transfer Learning, we train the model with one dataset and then train the same model with another dataset that has a different class distribution than in the training dataset. This technique makes it possible to transfer the knowledge acquired on a "source" dataset to better process a new "target" dataset. Transfer Learning can be intuitively explained by a simple example: let's imagine a person who wants to learn to play the piano, but will find it easier to do so if he or she already knows how to play the guitar. The person will be able to capitalize on his or her musical knowledge already acquired on the piano to learn to play a new instrument.

Figure 5.2: Transfer learning Approach



In Fine-tuning (see figure below), which is a Transfer Learning approach, we have a data set with which we perform 90% of the model training. The result is a pre-trained model. We then finish training the model, i.e. the remaining 10%, with another dataset specific to the task on which the model is to perform. The purpose of fine-tuning is to fine-tune the neural network's weights so that the model can be fine-tuned to a specific NLP task. In fact, the goal of fine-tuning is refine the weight of the last outputs layers on the specific NLP task. In most of case, the learning rate chosen for fine-tuning is lower than for conventional training, so as to minimize the impact on the weights of the neural network layers already adjusted.

The methodology of fine-tuning a BERT (Bidirectional Encoder Representations from Transformers) model for sentiment analysis in the financial domain encompasses several critical technical aspects.

The initial step involves selecting the BERT model, which is distinguished for its proficiency in various natural language processing tasks. This model, pre-trained on extensive general text, possesses a foundational linguistic understanding. Such pre-training enables the model to have a comprehensive base of language, context, and grammar, crucial for the subsequent fine-tuning process.

Data preparation for fine-tuning is a pivotal stage where datasets pertinent to the financial sector, containing texts like financial news or statements, are curated. This process includes the structuring and cleaning of data to ensure its compatibility with the BERT model. Key tasks in this phase involve selecting appropriate text fields and their corresponding sentiment labels.

Tokenization is an integral component of preparing the data for processing by BERT. This step involves converting text strings into a format comprehensible to the model, specifically by splitting texts into tokens or subwords and converting them into numerical IDs. Additionally, sequences are truncated or padded to maintain a uniform length, essential for batch processing.

The heart of the matter of fine-tuning lies in adapting the pre-trained model to the specific task of sentiment classification within the financial context. The model is trained on the financial text dataset with the aim to aptly adjust the model weights for precise sentiment classification. This is achieved using a labeled dataset where each text piece is tagged with a sentiment label, such as positive, negative, or neutral.

Configuring the training parameters is a critical aspect of fine-tuning. Parameters such as the learning rate, batch size, and the number of epochs directly influence the model's learning progression. The learning rate, for example, governs the magnitude of model weight adjustments in response to observed errors during batch processing.

Post-fine-tuning, the model undergoes evaluation using a validation dataset. This dataset, distinct from the training set, serves as an unbiased metric to assess the model's performance. Standard measures such as accuracy, precision, recall, and F1 score are employed to evaluate the efficacy of the model.

Lastly, the deployment and sharing of the fine-tuned model represent the culmination of the process. The model can be deployed for practical applications or shared within the community for further use or collaborative enhancements. Platforms like the Hugging Face model hub are commonly utilized for uploading and disseminating the fine-tuned model.

6 Model Descriptions

6.1 FINBERTX

FINBERTX is a sentiment analysis model fine-tuned on a balanced dataset. It employs an uncased BERT model, adapted using the *zeroshot/twitter-financial-news-sentiment* dataset from Hugging Face. This dataset, specifically designed for financial domain tweets, comprises approximately 11.9k tweets, divided into a training set of about 9.54k tweets and a validation set of around 2.39k tweets. The dataset was balanced using resampling techniques.

6.2 FINBERTA

FINBERTA, like FINBERTX, uses the same uncased BERT model. However, it is fine-tuned on the original *zeroshot/twitter-financial-news-sentiment* dataset without any resampling, maintaining the dataset's initial unbalanced state. This approach provides insights into how the model performs on naturally distributed data.

6.3 FinTweetBERT

FinTweetBERT is a model pre-trained on the *yiyanghkust/finbert-tone* on Twitter Financial News dataset. This model focuses on leveraging pre-existing financial sentiment analysis frameworks and applying them to Twitter data, specifically curated for financial news.

6.4 DistilRoBERTa

As the use of large pre-trained models for Transfer Learning in Natural Language Processing (NLP) grows, there are challenges in deploying these expansive models on edge devices or within computational constraints. In this study, we wanted to assess the performance of this kind of distilled model for our use-case. This model, must be subsequently be fine-tuned to achieve good results across various NLP tasks, similar to its more extensive counterparts. DistilRoBERTa is a distilled (smallel, lighter) version of the RoBERTa-base model, fine-tuned on the *financial_phrasebank* dataset. It combines the efficiency of the RoBERTa model with the specialized context of financial language, making it suitable for tasks like financial sentiment analysis, trend prediction, and market insight extraction.

6.5 FinBERT

FinBERT is a pre-trained NLP model developed for analyzing the sentiment of financial texts. It extends the BERT language model, trained on a large financial corpus, and is fine-tuned specifically for financial sentiment classification using the Financial *PhraseBank* dataset.

7 Models Evaluation

Please refer to the Appendix A4 to see the confusion matrix of each BERT-based models.

7.1 FinBERTx - finetuned on a balanced datase

Table 7.1: Classification Report: FINBERTX

	Precision	Recall	F1-score	Support
Bearish	0.56	0.83	0.67	347
Bullish	0.63	0.80	0.71	475
Neutral	0.93	0.76	0.84	1566
accuracy				
0.78				
macro avg	0.71	0.80	0.74	2388
weighted avg	0.82	0.78	0.79	2388

The classification report for FINBERTX, which was fine-tuned on a balanced dataset, reveals several key insights into its performance:

- **High Recall in Bearish Category:** The model exhibits a high recall of 0.83 in the Bearish category, indicating its effectiveness in identifying most of the Bearish instances. However, its precision in this category is comparatively lower at 0.56, suggesting a higher rate of false positives.
- **Balanced Performance in Bullish Category:** With a precision of 0.63 and a recall of 0.80, FINBERTX shows a balanced performance in the Bullish category, indicating a good mix of accuracy and coverage.

- **Strong Precision in Neutral Category:** The model demonstrates strong precision in the Neutral category (0.93), though the recall is slightly lower (0.76). This suggests that while it is highly accurate in identifying Neutral instances, it may miss some of them.
- **Overall Accuracy:** An overall accuracy of 0.78 is indicative of the model's robustness across all categories.
- **Weighted and Macro Averages:** The weighted average of the F1-score is 0.82, higher than the macro average of 0.71, indicating the model's better performance in classes with more instances.

7.1.1 FinBERTa - fine-tuned on an unbalanced dataset

Table 7.2: Updated Classification Report: FINBERTA

	Precision	Recall	F1-score	Support
Bearish	0.77	0.83	0.80	347
Bullish	0.83	0.81	0.82	475
Neutral	0.92	0.91	0.92	1566
accuracy				
0.88				
macro avg	0.84	0.85	0.85	2388
weighted avg	0.88	0.88	0.88	2388

FINBERTA exhibits commendable performance, particularly in Bullish and Neutral categories, with a high accuracy of 0.88. This indicates effective handling of the unbalanced dataset, supported by a significant weighted average score.

- **Enhanced Precision and Recall in Bearish Category:** The model now shows a higher precision (0.77) and recall (0.83) in the Bearish category, indicating its improved accuracy and ability to identify most Bearish instances.
- **Balanced Performance in Bullish Category:** The precision and recall in the Bullish category are closely matched (0.83 and 0.81, respectively), signifying a balanced identification of Bullish sentiments.
- **Exceptional Performance in Neutral Category:** FINBERTA achieves high precision (0.92) and recall (0.91) in the Neutral category, highlighting its exceptional capability in accurately identifying and not missing Neutral sentiments.
- **High Overall Accuracy:** An overall accuracy of 0.88, along with a weighted average of 0.88, indicates the model's high effectiveness in sentiment classification across all categories.
- **Strong Macro Average:** The macro average of 0.84 reflects the model's uniform strength across all categories, further emphasizing its robust performance on an unbalanced dataset.

7.1.2 FinTweetBERT

Table 7.3: Classification Report: FinTweetBERT

	Precision	Recall	F1-score	Support
Bearish	0.00	0.00	0.00	347
Bullish	0.87	0.99	0.93	475
Neutral	0.12	0.03	0.05	1566
accuracy				
0.22				
macro avg	0.33	0.34	0.32	2388
weighted avg	0.25	0.22	0.22	2388

FinTweetBERT faces challenges, especially in Bearish and Neutral categories, as reflected by the low precision and recall values. The overall accuracy of 0.22 suggests difficulties in capturing sentiment nuances in financial tweets.

- **Zero Performance in Bearish Category:** The model shows zero precision, recall, and F1-score for the Bearish category. This indicates a complete inability to correctly identify any Bearish instances, which is a significant concern.
- **High Performance in Bullish Category:** The model performs exceptionally well in the Bullish category with high precision (0.87), recall (0.99), and F1-score (0.93). This suggests a strong bias towards Bullish predictions.
- **Poor Performance in Neutral Category:** The performance in the Neutral category is notably weak, with low precision (0.12) and recall (0.03), resulting in a very low F1-score (0.05).
- **Overall Low Accuracy:** The overall accuracy of the model is just 0.22, indicating that it is correct only about 22% of the time across all categories.
- **Low Macro and Weighted Averages:** Both the macro average (0.33) and weighted average (0.25) F1-scores are low, further confirming the model's poor performance.

7.1.3 DistilRoBERTa

Table 7.4: Classification Report: DistilRoBERTa

	Precision	Recall	F1-score	Support
Bearish	0.57	0.78	0.65	347
Bullish	0.07	0.19	0.10	475
Neutral	0.38	0.14	0.21	1566
accuracy				
0.24				
macro avg	0.34	0.37	0.32	2388
weighted avg	0.34	0.24	0.25	2388

DistilRoBERTa shows a relatively balanced performance in the Bearish category but underperforms in the Bullish and Neutral categories. With an overall accuracy of 0.24, the model requires improvement.

- **Bearish Category:** The model shows a decent precision of 0.57 and a high recall of 0.78 in the Bearish category, suggesting it can reliably identify most Bearish instances, though it may also include some false positives.
- **Bullish Category:** The performance in the Bullish category is notably poor, with a very low precision of 0.07 and a recall of 0.19. This indicates that while the model identifies only a small fraction of Bullish instances, most of its Bullish predictions are incorrect.
- **Neutral Category:** The Neutral category also shows suboptimal results, with a precision of 0.38 and a recall of 0.14, indicating a struggle in correctly identifying Neutral instances and a tendency to miss many of them.
- **Overall Accuracy and Averages:** The overall accuracy of the model is low at 0.24, with macro and weighted averages of 0.34. This suggests that the model performs poorly across all categories.

7.1.4 FinBERT

Table 7.5: Classification Report: FinBERT

	Precision	Recall	F1-score	Support
Bearish	0.03	0.03	0.03	347
Bullish	0.11	0.12	0.11	475
Neutral	0.85	0.76	0.80	1566
accuracy				
0.53				
macro avg	0.33	0.39	0.32	2388
weighted avg	0.58	0.78	0.53	2388

FinBERT excels in identifying Neutral sentiments but struggles in the Bearish and Bullish categories. The moderate overall accuracy of 0.53 and lower macro and weighted averages suggest limited reliability for comprehensive sentiment analysis.

7.1.4.1 Potential Use Case:

This model is well-suited for niche applications focused primarily on bullish sentiment detection.

8 Discussion from a data science point of view

Table 8.1: Performance Metrics of Different Sentiment Analysis Models

Model	Accuracy	Precision	Recall	F1-Score
Naive Model {SentiWordNet}	0.42	0.37	0.38	0.36
Naive Model {SentiWordNet + Negation Words + Booster Words}	0.43	0.37	0.38	0.36
Naive Model {SentiWordNet + Negation & Booster Words + Emoji}	0.42	0.37	0.38	0.36
Vader Algorithm	0.49	0.45	0.48	0.44
ML Model {CountVectorizer, TF-IDF, MultinomialNB}	0.64	0.55	0.60	0.57
FinBERTx fine-tuned	0.78	0.71	0.80	0.74
FinBERTa	0.88*	0.84*	0.85*	0.85*
FinTweetBERT	0.22	0.33	0.34	0.32
DistilRoBERTa	0.24	0.34	0.37	0.32
FinBERT	0.53	0.33	0.39	0.32

The graph above shows the performance metrics of all the sentiment analysis models used in this report. We can see that the naive model based on SentiWordNet scores around 40% on all metrics, suggesting that it performs moderately well. In the financial domain, a sentiment analysis model that relies solely on vocabulary may not be sufficient to capture the complex emotions in the text. The performance of the VADER model improves, but not significantly, which may indicate that the semantics of emoticons may not be as important in financial text analysis as they are in everyday communication.

In contrast, models using machine learning approaches (e.g. simple polynomial Bayesian models based on CountVectorizer and TF-IDF) show significant improvements in all metrics, highlighting the capabilities of machine learning algorithms for feature learning and pattern recognition. Regarding Deep Learning model, BERT model fine-tuned on our dataset (e.g. FinBERTa, FinBERTx,...) outperform on all performance metrics, and in particular the FinBERTa model exceeds 80% of our evaluation metrics especially in terms of accuracy, precision, recall and F1 score.

For the purposes of our study, we're most interested in the prediction accuracy associated with bearish and bullish financial tweets, because these tweets provide very useful information for the trader. We note that the even if FinTweetBERT is not model which have the higher overall F1-Score this model is the most accurate model at detecting bullish tweets (F1-Score at 0.93 for bullish tweets).

Overall we observe that, BERT models that are not fine-tuned, (i.e. models for which we have not applied a transfer learning approach), perform less well than naive models and even less well than machine learning models. In our case study, we show the importance of the transfer learning method to refine the prediction of the model.

In conclusion, from a quantitative point of view, deep learning models, especially those that have been targeted and improved for a financial domain thanks to fine-tuning approach such as FinBERTa, perform better in sentiment analysis tasks.

9 Discussion from a business point of view

The results we were able to obtain are consistent with the scientific literature, which states that the fine-tuned model based on a transform architecture outperforms traditional approaches based on sentiment dictionaries or statistical learning approach.

This is due to innovations in the BERT model architecture. Indeed the main innovation behind BERT, especially its attention mechanism, revolutionized the field of NLP by enabling models to capture deeper contextual information from text data. By employing bidirectional context and the Transformer architecture's efficiency, BERT set new benchmarks in various NLP tasks, leading to significant advancements and applications in areas like machine translation, question-answering, sentiment analysis, and even more.

Even if, VADER model may not perform as well as more sophisticated deep learning models on certain types of data or in specific domains, it's important to note that VADER is a useful tool when we do not have access to labels because the model does not require labeled data.

Machine Learning model provide significant improvement than rule-based model thanks to the fact that machine learning model are able to fit their vocabulary (e.g. build their own dictionary with the bag of words technique) on the dataset. But the drawback of these statistical learning approaches is that they are supervised and require labeled data. Indeed, in NLP projects it is rare to have datasets that have been labeled by a human.

However, we show that new approaches based on Deep Learning (i.e. Transformer architecture) outperform traditional NLP approaches based on the use of sentimental word dictionaries. Deep learning approach are more suitable on sentiment task analysis in case of long dependencies between words and very specific vocabulary (in our case financial terms) and rule-based models are typically designed based on specific rules, criteria, or dictionaries, making them less adaptable to new or unseen data outside their predefined scope but are useful when we don't have access to labeled data.

Overall, in this study, we compare different approaches to analyze financial tweets. Our benchmark shows that BERT-based model (particularly, FinBERTa model) are useful to assess the sentiment of financial tweets. Thanks to this kind of model, traders can get a sense of overall market sentiment, whether it's bullish (positive) or bearish (negative). This macro-level insight can help the trader to improve strategies and take better informed decisions.

From a business point of view, our goal is to maximize the accuracy in predicting bullish or bearing financial tweets because these two types of tweets give interesting insights for the trade to take informed investment decisions. Indeed, if the predicted sentiment of a financial tweet is bullish, the trader might opt to buy or hold, anticipating an upward movement in price. A neutral sentiment does not provide such actionable insights, making it less valuable for immediate trading decisions. In other words, knowing whether the sentiment is bearish or bullish helps traders manage their risks effectively. If the sentiment is bearish, a trader might employ strategies like short-selling or using options to hedge against potential losses. On the other hand, a bullish sentiment might encourage traders to take on more risk or leverage their positions, expecting higher returns. Therefore, maximizing accuracy in predicting bearish or bullish sentiments enables traders to refine their trading strategies continually. By analyzing patterns in sentiment data, traders can identify trends, correlations, and anomalies that inform their decision-making processes. Over time, this iterative refinement can lead to more robust, adaptive, and profitable trading strategies.

As a matter of fact, analyzing sentiment in financial tweets for trading activities can provide several advantages and insights for traders and investors. Financial tweets can give early market signals, financial tweets often reflect the immediate sentiments and reactions of traders, investors, and the general public. Analyzing these sentiments can provide early signals about market trends or potential shifts in market sentiment.

Sentiment analysis helps in extracting actionable insights from this massive volume of data, which might be challenging to process manually. Analyze financial tweets can give relevant insights in terms of risk management by understanding sentiment and helping traders and investors gauge the level of risk associated with specific stocks, commodities, or financial instruments. Positive sentiment (e.g. bullish) may indicate potential growth or stability, while negative (e.g. bearish) sentiment could signal risks or downturns.

In addition, financial tweets often react to significant events like earnings reports, geopolitical events, regulatory changes, or breaking news. Sentiment analysis can help traders quickly assess the impact of such events on specific assets or the market as a whole.

Furthermore, incorporating sentiment analysis into trading strategies can provide a competitive edge by enabling traders to react rapidly to market sentiment shifts. This agility can result in better trade execution and potentially higher returns.

Some model such as FinBERTa model can be can be integrated into algorithmic trading systems or trading bots. By automatically analyzing tweets and adjusting trading strategies based on sentiment scores, these systems can execute trades more efficiently and capitalize on market opportunities.

In summary, financial tweet sentiment analysis offers a data-driven approach to understanding market sentiment, identifying trading opportunities, managing risks, and enhancing trading strategies. By leveraging advanced analytics and deep learning techniques, traders and investors can make more informed decisions and potentially achieve better financial outcomes.

Please refer to the following link to see the full code of this study: Github

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*. Revised 2023.

Appendix

A1 Naive Models: Classification Report and Confusion Matrix

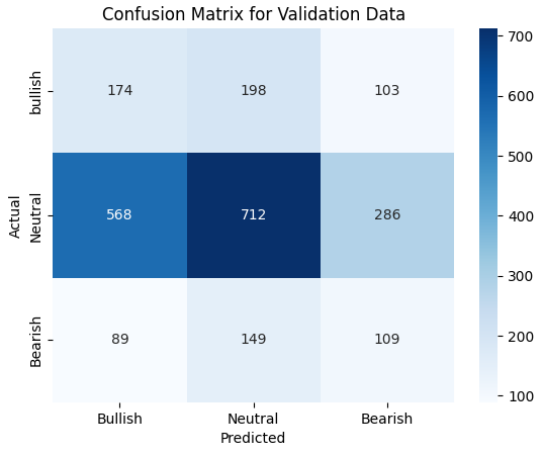


Figure A1.1: Confusion Matrix: Naive Model - {SentiWordNet}

Classification Report:	precision	recall	f1-score	support
bearish	0.22	0.31	0.26	347
bullish	0.21	0.37	0.27	475
neutral	0.67	0.45	0.54	1566
accuracy			0.42	2388
macro avg	0.37	0.38	0.36	2388
weighted avg	0.51	0.42	0.45	2388

Figure A1.2: Classification Report: Naive Model - {SentiWordNet}

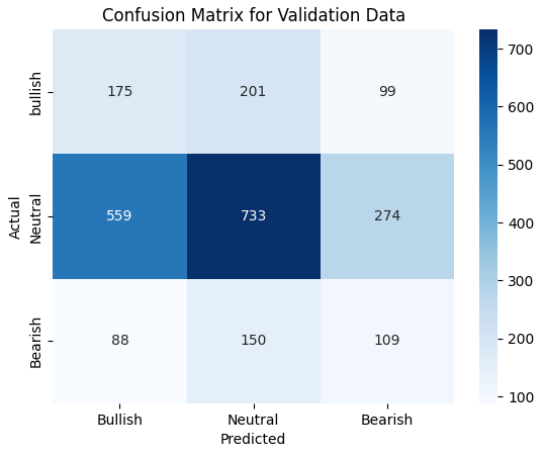


Figure A1.3: Confusion Matrix: Naive Model - {SentiWordNet + Negation Words + Booster Words}

Classification Report:	precision	recall	f1-score	support
bearish	0.23	0.31	0.26	347
bullish	0.21	0.37	0.27	475
neutral	0.68	0.47	0.55	1566
accuracy			0.43	2388
macro avg	0.37	0.38	0.36	2388
weighted avg	0.52	0.43	0.45	2388

Figure A1.4: Classification Report: Naive Model - {SentiWordNet + Negation Words + Booster Words}

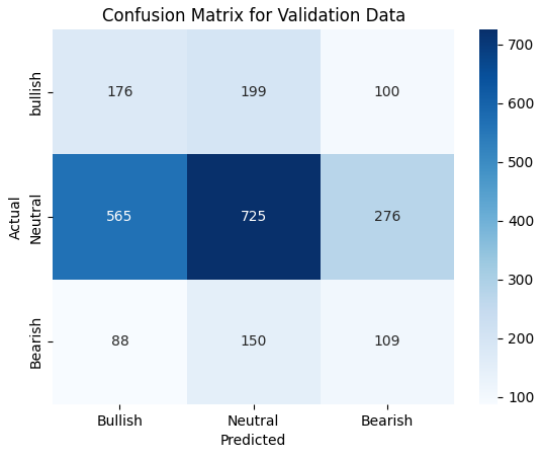


Figure A1.5: Confusion Matrix: Naive Model - {SentiWordNet + Negation Words + Booster Words + Emoji}

Classification Report:	precision	recall	f1-score	support
bearish	0.22	0.31	0.26	347
bullish	0.21	0.37	0.27	475
neutral	0.68	0.46	0.55	1566
accuracy			0.42	2388
macro avg	0.37	0.38	0.36	2388
weighted avg	0.52	0.42	0.45	2388

Figure A1.6: Classification Report: Naive Model - {SentiWordNet + Negation Words + Booster Words + Emoji}

A2 Vader Algorithm: Classification Report and Confusion Matrix

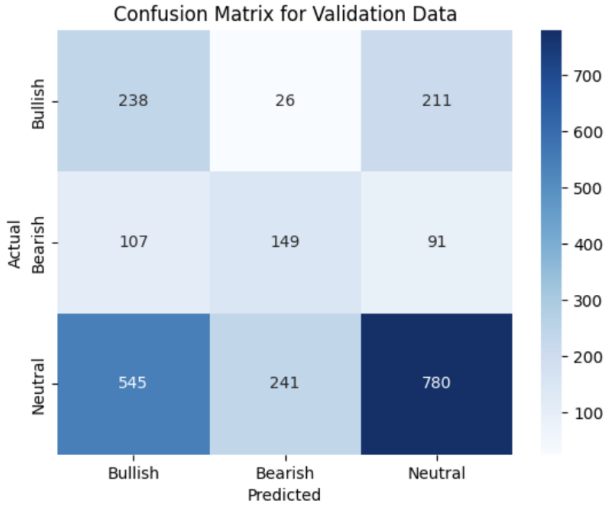


Figure A2.1: Confusion Matrix: Vader Model

Classification Report:	precision	recall	f1-score	support
Bearish	0.36	0.43	0.39	347
Bullish	0.27	0.50	0.35	475
Neutral	0.72	0.50	0.59	1566
accuracy			0.49	2388
macro avg	0.45	0.48	0.44	2388
weighted avg	0.58	0.49	0.51	2388

Figure A2.2: Classification Report: Vader Model

A3 ML Model: Classification Report and Confusion Matrix

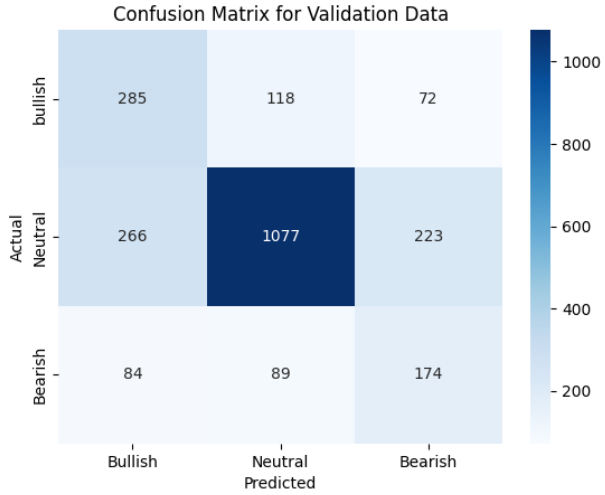


Figure A3.1: Confusion Matrix: ML Model - {CountVectorizer, TF-IDF, MultinomialNB}

```
---- Classification report ----
              precision    recall  f1-score   support

           0       0.37      0.50      0.43       347
           1       0.45      0.60      0.51       475
           2       0.84      0.69      0.76      1566

    accuracy                           0.64      2388
   macro avg       0.55      0.60      0.57      2388
weighted avg       0.69      0.64      0.66      2388
```

Figure A3.2: Classification Report: ML Model - {CountVectorizer, TF-IDF, MultinomialNB}

A4 Deep Learning model - BERT models: Classification Report and Confusion Matrix

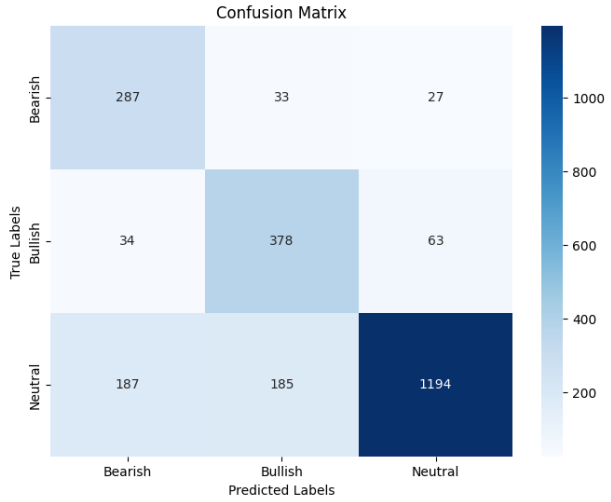


Figure A4.1: Confusion Matrix: ML Model - FINBERX - Uncased BERT Fine-tuned on a balanced dataset

	precision	recall	f1-score	support
Bearish	0.56	0.83	0.67	347
Bullish	0.63	0.80	0.71	475
Neutral	0.93	0.76	0.84	1566
accuracy			0.78	2388
macro avg	0.71	0.80	0.74	2388
weighted avg	0.82	0.78	0.79	2388

Figure A4.2: Classification Report: ML Model - FINBERX - Uncased BERT Fine-tuned on a balanced dataset

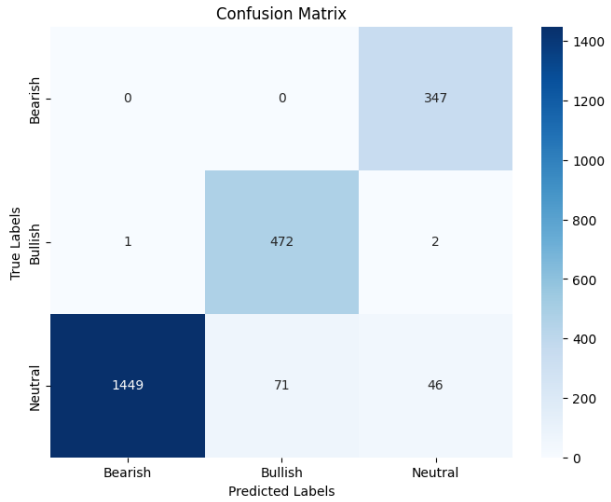


Figure A4.3: Confusion Matrix: ML Model - FinTweetBERT

	precision	recall	f1-score	support
Bearish	0.00	0.00	0.00	347
Bullish	0.87	0.99	0.93	475
Neutral	0.12	0.03	0.05	1566
accuracy			0.22	2388
macro avg	0.33	0.34	0.32	2388
weighted avg	0.25	0.22	0.22	2388

Figure A4.4: Classification Report: ML Model - FinTweetBERT

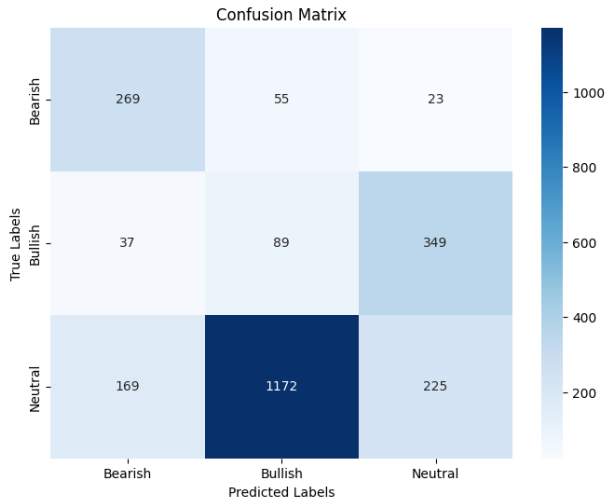


Figure A4.5: Confusion Matrix: ML Model - DistilRoBERTa

	precision	recall	f1-score	support
Bearish	0.57	0.78	0.65	347
Bullish	0.07	0.19	0.10	475
Neutral	0.38	0.14	0.21	1566
accuracy			0.24	2388
macro avg	0.34	0.37	0.32	2388
weighted avg	0.34	0.24	0.25	2388

Figure A4.6: Classification Report: ML Model - DistilRoBERTa

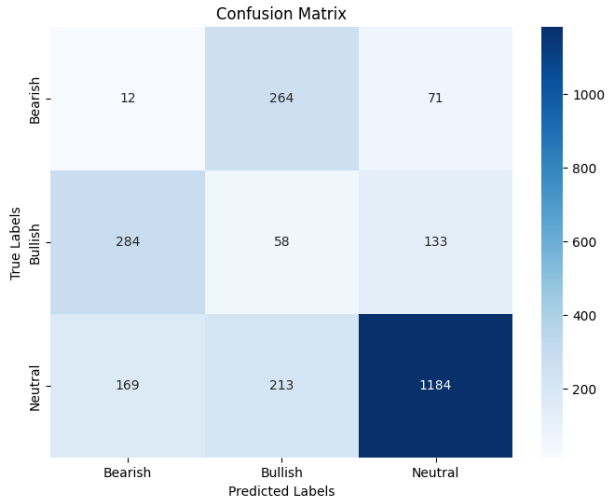


Figure A4.7: Confusion Matrix: ML Model - FinBERT

	precision	recall	f1-score	support
Bearish	0.03	0.03	0.03	347
Bullish	0.11	0.12	0.11	475
Neutral	0.85	0.76	0.80	1566
accuracy			0.53	2388
macro avg	0.33	0.30	0.32	2388
weighted avg	0.58	0.53	0.55	2388

Figure A4.8: Classification Report: ML Model - FinBERT

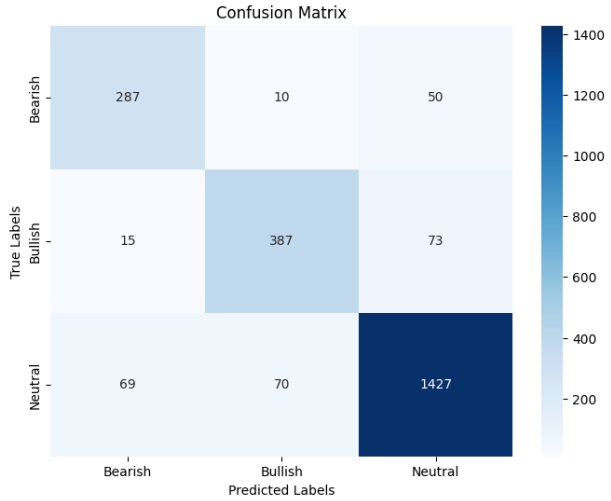


Figure A4.9: Confusion Matrix: ML Model - FinBERTA - Uncased BERT Fine-tuned on an unbalanced dataset

	precision	recall	f1-score	support
Bearish	0.77	0.83	0.80	347
Bullish	0.83	0.81	0.82	475
Neutral	0.92	0.91	0.92	1566
accuracy			0.88	2388
macro avg	0.84	0.85	0.85	2388
weighted avg	0.88	0.88	0.88	2388

Figure A4.10: Classification Report: ML Model - FinBERTA - Uncased BERT Fine-tuned on an unbalanced dataset