Master thesis
Econometrics and Mathematical economics

Sentiment Analysis and Market Volatility: Predicting VIX Direction with Hybrid Models

by

Giorgia Ceccarelli (SNR: 2121726)

Date: November 27, 2024

Contents

Chapter 1

Management summary

1.1 Introduction

This thesis investigates the relationship between sentiment analysis and market volatility, specifically focusing on the VIX, a volatility index that reflects market expectations. Given the increasing relevance of social media sentiment in shaping financial markets, this research aims to explore how sentiment expressed in tweets impacts changes in the VIX direction. By integrating sentiment data with GARCH-based features, this study offers a novel approach to predicting volatility, moving beyond traditional stock price analysis.

1.2 Research Questions

Sentiment Analysis and Volatility: My research aims to explore the relationship between sentiment analysis (neutral, positive, and negative sentiments) and market volatility, specifically focusing on the VIX, an index that reflects market expectations of future volatility. While there is already substantial research linking sentiment to stock prices, fewer studies have examined its connection to the VIX. I want to directly investigate how sentiment influences changes in the direction of volatility rather than absolute values, an area less explored in the literature. Additionally, I aim to integrate GARCH-based features with sentiment data to determine whether combining traditional financial models with sentiment analysis improves predictions. Hybrid modeling in volatility prediction is still relatively novel, and I hope to contribute to this growing field by testing whether these two approaches together can provide better insights than using either one alone.

1.3 Approach

I used a dataset downloaded from the IEEEDataPort that contains 943673 tweets that were collected between April 9 and July 16, 2020. The tweets included references to the S&P 500 index and its top 25 companies, identified using relevant hashtags and stock tags. Out of this dataset, 1,300 tweets were manually annotated into positive, neutral, and negative categories. These annotations served as a training set for machine learning models, which were then applied to classify the remaining tweets. To diversity the source of data, I also tried to scrape journal articles from the New York Times with API, however I was able to gather only two or three articles per day, which was not enough for the analysis I intend to do.

1.4 Key Findings

This research indicates that both positive and negative sentiments have a significant influence on market volatility direction, showing clear connections between social media sentiment and VIX fluctuations. Positive sentiment is linked to decreased volatility, while negative sentiment corresponds to increased volatility. By combining GARCH-based features with sentiment data, the hybrid model enhances the precision of predicting volatility trends compared to using each method independently. Incorporating machine learning models such as Random Forest and XGBoost has led to high prediction accuracy, with Logistic Regression achieving a rate of 92.86%.

1.5 Innovations

A key innovation of this study is the development of a hybrid model that combines sentiment analysis with GARCH-based volatility features to predict the direction of market volatility. This approach represents a novel contribution to the field of financial forecasting, as it integrates emotional market drivers with traditional financial indicators. The use of machine learning techniques such as Random Forest and XGBoost for the sentiment analysis further enhanced the predictive accuracy of the models.

1.6 Relevance

This research has significant implications for professionals in finance, risk management, and marketing. In finance, sentiment analysis is crucial for predicting market volatility and evaluating operational risk. By incorporating sentiment analysis in volatility forecasting, professionals gain a more dynamic way to grasp market sentiment, enhancing portfolio management, derivative pricing, and volatility-based trading strategies.

When it comes to operational risk, which faces unforeseen disruptions and market reactions, sentiment analysis acts as an early alert system. Monitoring social media sentiment lets finance experts identify shifts in public sentiment and foresee potential impacts on market volatility, especially during economic uncertainty or crises. Introducing real-time sentiment data into existing risk models enables a more proactive risk management approach, empowering firms to adjust strategies promptly and mitigate potential losses.

Outside of finance, marketing and customer experience teams widely apply sentiment analysis to comprehend consumer sentiment and brand perception. Examining social media and review platforms helps companies measure public reactions to products or services, allowing them to adapt swiftly to consumer needs. By analyzing customer reviews and feedback, companies can refine offerings, improve brand image, and optimize marketing strategies based on real-time sentiment data.

In conclusion, this studys use of sentiment analysis in financial volatility prediction introduces an advanced toolset that benefits various business functions. By revealing sentiment-driven trends, this research provides a versatile approach relevant to finance, risk management, and marketing sectors, emphasizing the importance of understanding and addressing sentiment for competitive advantage.

1.7 Outline of following chapters

Chapter 2: this chapter introduces the key concepts of sentiment analysis and its relevance to financial markets, focusing on how sentiment from social media, particularly Twitter, can be used to predict

market volatility. It provides an overview of lexicon-based and machine learning approaches.

Chapter 3: in this chapter, I review the relevant literature on sentiment analysis and its application to financial markets, particularly in predicting volatility. This chapter discusses various models and identifies gaps in the research.

Chapter 4: explains in details the methodology for sentiment analysis, including data collection, cleaning, and the application models.

Chapter 5: here, I present the results of the VIX prediction models, comparing different machine learning methods and the hybrid approach. The chapter highlights how combining sentiment analysis with GARCH-based features improves the accuracy of volatility direction predictions.

Chapter 6: the final chapter summarizes the key findings, practical implications, and the improvement of integrating sentiment analysis with traditional financial models. It also provides recommendations for future research.

Chapter 2

Introduction to sentiment analysis

2.1 What is sentiment analysis

Sentiment analysis is an application of Natural Language Processing (NLP), it is used to determine the emotions and tone behind some text, allowing businesses, researchers, and analysts to understand public opinion, gauge market trends, and make informed decisions. With the growing influence of social media and digital communication, sentiment analysis has become increasingly valuable in various domains, including finance, marketing, politics, and customer service. The task involves classifying text into categories such as positive, negative, or neutral sentiment, often with the added complexity of detecting sarcasm, irony, or subtle emotional undertones.

The effectiveness of sentiment analysis largely depends on the quality of data preprocessing and the choice of models used. Data cleaning and preprocessing are the most important steps in text data processing, ensuring that the input data is standardized, noise-free, and ready to be inserted into a model. Properly cleaned data enhances the performance of sentiment analysis models by allowing them to learn from relevant and meaningful information. All the steps I did to clean my dataset are explained in 2.2.

The models used in sentiment analysis can be broadly categorized into lexicon-based approaches, machine learning models, deep learning models, and transfer learning models.

Lexicon-based models use predefined dictionaries of sentiment-laden words, offering a straightforward but often limited approach due to their static nature and inability to handle context.

Machine learning models, such as Naive Bayes and Support Vector Machines (SVM), traditionally rely on feature extraction techniques to convert text data into numerical formats that these models can process. Two common techniques used for this purpose are Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

In the Bag-of-Words (BoW) approach, a text is represented as an unordered collection of words, without considering grammar or word order. Each unique word is treated as a feature, and the model creates a vector that records how frequently each word appears in the document. This method is simple and may struggle to capture the importance of certain words, also common but uninformative terms (like "the" and "and") can dominate the representation.

To address this, TF-IDF goes a step further by not only counting word frequencies but also taking into account how important a word is in relation to the entire text. Words that appear frequently in one document but rarely in others are given higher importance. This helps highlight more informative terms

while reducing the importance of common words that do not contribute much to the understanding of a documents content.

Both BoW and TF-IDF transform textual data into numerical vectors, enabling machine learning models like Naive Bayes and SVM to process them as input for classification tasks. However, these techniques are limited in their ability to capture the contextual meaning of words since they treat words as independent units. This limitation has led to the development of more advanced methods, such as word embeddings and deep learning models, which can better capture the nuances and relationships between words.

Deep learning models like CNNs, RNNs, and Transformers have transformed sentiment analysis by automatically learning complex patterns and features from raw text data, often outperforming traditional methods.

Transfer learning models, using pre-trained models such as BERT, GPT, and RoBERTa, boost performance by adapting general language representations to sentiment-specific tasks with minimal fine-tuning. However, these models are highly complex, require large datasets for training, and demand significant computational resources.

Lastly, evaluation metrics play a crucial role in assessing the performance of sentiment analysis models, helping to measure their accuracy, precision, recall, and other relevant aspects. The selection of appropriate metrics ensures that models are evaluated in a complete way.

In this chapter, I will explain the components of data cleaning and explore various sentiment analysis models, discussing their methodologies, strengths, limitations, and applications.

2.2 Data Cleaning in Text Data Processing

Data cleaning is a crucial step in preparing text data for sentiment analysis or any other natural language processing (NLP) task. Clean data ensures that the models learn from relevant and accurate information, which can significantly improve performance and accuracy. The process of cleaning text data involves several steps, each aimed at removing noise and standardizing the text.

- **Removing Punctuation and Special Characters:** Punctuation marks, special characters, and numbers do not carry significant meaning in text analysis tasks and can be removed to reduce noise. Example: Converting "Hello, world! It's a beautiful day." to "Hello world It's a beautiful day".
- **Lowercasing:** Converting all characters in the text to lowercase ensures uniformity. This step prevents the same words in different cases from being treated as separate entities. For example, converting "Hello World" to "hello world".
- **Removing Stop Words:** Stop words are common words (such as "and", "the", "is") that usually do not contribute significant meaning to the text and can be removed to reduce dimensionality. For example, converting "This is a sample sentence" to "sample sentence".
- **Tokenization:** Tokenization is the process of splitting text into individual words or tokens. This is a fundamental step for further text-processing tasks. For example converting "Hello world" to ["Hello", "world"].
- **Stemming and Lemmatization: Stemming** consists in reducing words to their base or root form by removing suffixes, this can help in reducing inflectional forms of a word, while **lemmatization** goes a step further by considering the morphological analysis of words, reducing them to their meaningful base form or lemma.

- **Removing HTML Tags:** In cases where text is extracted from web pages, HTML tags need to be removed as they do not contribute to the textual analysis.
- **Removing Whitespace:** Extra whitespace, including tabs and newline characters, should be removed or normalized to ensure consistency in text formatting.
- **Handling Contractions:** Contractions are shortened forms of words or groups of words (e.g., "don't" for "do not"). Expanding contractions can help in standardizing text.
- **Dealing with Numbers:** Numbers may need to be removed or transformed based on the specific task. In some cases, converting numbers to their textual representation can be helpful.
- **Handling Emojis and Emoticons:** mojis and emoticons can carry sentiment information and should be either removed or converted to their textual descriptions. Libraries like emoji in Python can convert emojis to their textual descriptions.

Data cleaning is a multi-step process that prepares text data for analysis by removing noise and standardizing the text. Each step, from removing punctuation and special characters to handling emojis and emoticons, is essential for ensuring that the data is in a consistent and usable format. Properly cleaned data enables more accurate and efficient sentiment analysis, as the models can focus on the meaningful content of the text.

2.3 Lexicon Based Model

Lexicon-based models rely on a predefined list of words that is associated with some sentiment (negative, positive, or neutral for example). This model is very simple and not as computationally expensive as other models, this is one of the simplest models for Sentiment Analysis and there is no need to train the data.

These models determine the sentiment of a text by analyzing the sentiment of individual words or phrases within the text.

How lexicon-based models work is not complex:

- **Sentiment Lexicon:** There is a dictionary where words are associated with scores (positive, neutral, and negative), the dictionary can also be automatically created, instead of using one already created manually.

  Among the dictionaries commonly used the most popular among researchers is:

  - **SentiWordNet**, it assign sentiment based on its internal system, and it is often used for detailed sentiment analysis Esuli and Sebastiani [2006].
  - **AFINN** is a list of words, in English, to which an integer ranging from -5 to 5 is assigned. It very simple to use and it is often updated to include slang as well Nielsen [2011].
  - **VADER** is a dictionary that is focused on social media language because it accounts for the intensity of sentiment expression also making use of emoticons, slang, and abbreviations. It widely used to process social media text. Hutto and Gilbert [2014].
  - **Loughran-McDonald** Sentiment Word Lists is a dictionary based on financial text, and therefore designed for financial applications. Loughran and McDonald [2011].

  There are two types of lexicon-based approaches: dictionary-based approaches and corpus-based approaches.

2.3.1 Dictionary-Based Approaches

Dictionary-based approaches utilize predefined lists of words (dictionaries or lexicons) that are associated with sentiment values. These dictionaries are manually created or curated by experts and contain words that are classified as positive, negative, or neutral. Each word in the input text is matched against the entries in the sentiment dictionary. The sentiment scores (positive, negative, or neutral) of the matched words are then aggregated to determine the overall sentiment of the text. The main advantage is that it is easy to implement and work without the need for labeled training data. The main disadvantage is that cannot handle context-specific meanings or the impact of word combinations.

2.3.2 Corpus-Based Approaches

In sentiment analysis, the corpus-based approach extends and refines sentiment dictionaries by taking advantage of large text corpora. These methods use big datasets to deduce, from the context, the sentiment associations of words and phrases through statistical and computational techniques. In a way, corpus-based approaches can look at how words flow within varied contexts and, therefore, can tell a better sentiment score than dictionary-based model. One of the most used is SenticNet, which combines common-sense knowledge with contextual information to include all multi-word expressions and concepts in its lexicon. A notable drawbacks lie in the complexity of the implementation and substantial computational resources and large corpora required. This is why I did not use in my analysis.

While lexicon-based models like VADER are widely used for sentiment analysis due to their simplicity and efficiency, they have significant limitations, particularly when applied to domain-specific texts. These models often struggle with context sensitivity, failing to recognize meanings of words in specific domains, such as financial terms that may shift sentiment depending on the context. Additionally, they may misinterpret complex sentence structures and cannot adequately handle sarcasm, irony, or polarity shifts within the text.

However, these limitations can be addressed by integrating lexicon-based models with more sophisticated approaches, and this is exactly what I did in my analysis, such as machine learning models and advanced feature extraction techniques like TF-IDF (Term Frequency-Inverse Document Frequency). When lexicon-based sentiment scores are used as features in machine learning models, the models can leverage additional context from the data, overcoming the static nature of lexicons. Combining lexicon-based features with TF-IDF and using machine learning algorithms like Random Forest or Support Vector Machines (SVM) allows for better performance and more nuanced sentiment predictions, especially in context-heavy domains like financial markets.

2.4 Feature extraction

Machine learning models are widely used in the field of Sentiment Analysis because of their ability to learn patterns from labeled data and make accurate predictions. In the rest of this section, I'm going to outline all the models that I considered for my analysis.

Before being able to apply any kind of sentiment classification model to the text data, feature extraction has to be applied. it consists in transforming text into some kind of numerical representation that can serve as input in the model. There are multiple techniques for feature extraction, and among the most common ones to be explained later are a bag-of-words, TF-IDF, and word embeddings.

2.4.1 Bag-of-Words (BoW)

The bag-of-words model is an easy and quick representation used in natural language processing and information retrieval. In this model, a text is represented as an unordered collection of words, disregarding grammar and word order but keeping multiplicity. Mathematically, this can be expressed as a vector

$$\mathbf{v}_d = [f(w_1, d), f(w_2, d), \ldots, f(w_N, d)]$$

where $f(w_i, d)$ is the frequency count of word w_i in document d, and N is the size of the vocabulary.

This method is simple to implement and provides a straightforward way to convert text data into numerical form. However, it ignores the order of words and can result in high-dimensional vectors if the vocabulary is large, and lastly, it treats all words equally, not recognizing that they often have different weights Harris [1954].

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

TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It is commonly used in information retrieval and text mining to identify words that are important to a specific document while reducing the weight of commonly used words.

1. **Term Frequency (TF)**:
   - Term Frequency measures how frequently a term appears in a document. It reflects the importance of a term within the context of that document. It helps in understanding the importance of a term within the specific document context.
   - Formula:
   $$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$
2. **Inverse Document Frequency (IDF)**:
   - Document Frequency measures how important a term is across the corpus. It aims to reduce the weight of common terms that appear in many documents and increase the weight of terms that appear in fewer documents. IDF adjusts the term frequency by accounting for how common or rare a term is across the entire corpus. Common terms that appear in many documents will have lower IDF values, while rare terms that appear in few documents will have higher IDF values. It helps in reducing the impact of commonly used words (like "the", "is", etc.) and highlighting unique terms that might be more informative.
   - Formula:
   $$\text{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right)$$
3. **TF-IDF Score**:
   - The TF-IDF score combines TF and IDF to give a score that represents the importance of a term in a document within the corpus. It reflects both the term's frequency in the document and its rarity across the corpus. It is a balance measure that emphasizes terms that are important in the specific context of the document while downplaying terms that are common across the corpus. It is particularly useful for tasks like text classification, information retrieval, and document clustering.

- Formula:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

TF-IDF is a powerful tool for text analysis, helping to identify terms that are both significant in specific documents and unique across the corpus. By combining local and global importance, TF-IDF provides a more nuanced measure of term relevance, making it essential for various natural language processing tasks.

2.4.3 Word embeddings

Word embeddings are a type of word representation that allows words to be represented as vectors in a continuous vector space. They are designed to capture the semantic meaning of words in such a way that words with similar meanings are located close to each other in this space. This concept is crucial for various natural language processing (NLP) tasks.

Word embeddings can be created using several methods, with Word2Vec and GloVe being two prominent approaches. Word2Vec, developed by Mikolov et al. at Google in 2013 Mikolov et al. [2013a], includes two main models: Continuous Bag of Words (CBOW) and Skip-Gram.

The CBOW model predicts a target word w_t based on its surrounding context words, maximizing the probability:

$$\max \prod_{t=1}^{T} \Pr(w_t | w_{t-k}, \ldots, w_{t+k})$$

where k is the window size around the target word.

The Skip-Gram model, on the other hand, does the reverse by predicting context words given a target word, maximizing:

$$\max \prod_{t=1}^{T} \prod_{-k \leq j \leq k, j \neq 0} \Pr(w_{t+j} | w_t)$$

The training objective for both models is to maximize the probability of context words given a target word (or vice versa), effectively capturing word relationships based on proximity in a text corpus.

On the other hand, GloVe (Global Vectors for Word Representation), developed by Pennington et al. at Stanford in 2014 Pennington et al. [2014], combines global matrix factorization with local context window methods. GloVe utilizes word co-occurrence statistics across the corpus, aiming to create word vectors $\mathbf{w}$ and $\mathbf{c}$ such that their dot product approximates the logarithm of the probability of their co-occurrence:

$$\mathbf{w}_i \cdot \mathbf{c}_j = \log(X_{ij})$$

where X_{ij} is the frequency of co-occurrence between words i and j. This approach integrates both global and local context information but it is more suitable for longer text, as it may not perform as well with short text such as tweets.

Word embeddings suffer from the out-of-vocabulary (OOV) problem, as unseen words during training are ignored. Additionally, they have high dimensionality, which increases memory and computational demands, and they can perpetuate biases present in the training data. Furthermore, word embeddings

struggle with capturing complex semantics, such as negation or sarcasm, and require large corpora for effective pre-training, which can limit their performance in specialized domains like finance.

2.5 Machine learning models

Machine learning classifiers come in various forms, each with its own strengths and weaknesses. **Naive Bayes**, a probabilistic classifier based on Bayes' theorem, assumes independence between features. This classifier is simple, easy to implement, efficient, and works well with small datasets, but its assumption of feature independence is often unrealistic, which can limit its performance on complex datasets, Rish [2001]. **Support Vector Machines (SVM)** are robust classifiers that excel in high-dimensional spaces and are effective in finding a clear margin of separation between classes. However, they are memory-intensive and can be slow to train on large datasets, with their performance heavily dependent on the choice of kernel and regularization parameters Cortes and Vapnik [1995]. **Logistic regression** models the probability of class membership using the logistic function and is valued for its simplicity and interpretability. It handles both binary and multi-class problems well, especially in large datasets, though it assumes a linear relationship between features and the log-odds, which may not capture complex relationships Hosmer et al. [2013]. **Decision trees** split data based on feature values, creating a tree-like structure that is easy to understand and interpret. They handle both numerical and categorical data with minimal preprocessing but are prone to overfitting and can be unstable with small changes in data Breiman et al. [1984]. **Random forests**, an ensemble method of decision trees, improve generalization by reducing overfitting and maintaining robustness with large datasets. Despite their computational expense and reduced interpretability compared to individual trees, they are highly effective in various applications Breiman [2001]. Each classifier offers unique advantages and trade-offs, making the choice of algorithm crucial depending on the specific requirements and characteristics of the dataset at hand.

2.6 Deep Learning Models

Deep learning models have revolutionized sentiment analysis by automatically learning feature representations from raw text data, often leading to better performance on complex tasks. These models can capture intricate patterns in data, which is essential for understanding the nuanced sentiments expressed in text. The key deep learning models used in sentiment analysis include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers.

- **Convolutional Neural Networks (CNN)**

  Convolutional Neural Networks (CNNs), though initially developed for image processing, have been adapted for text analysis by treating text as a sequence of one-dimensional grids. In sentiment analysis, CNNs use filters to capture local patterns like phrases or n-grams that indicate sentiment. A convolutional layer applies a filter W to an input sentence represented as a matrix X, producing feature maps:

  $$h_i = f(W \cdot X_{i:i+k-1} + b)$$

  where f is an activation function (such as ReLU), $X_{i:i+k-1}$ is a window of k words, and b is a bias term. Pooling layers then reduce the dimensionality of these feature maps, retaining only the most significant features. Their simple architecture and ability to handle varying text lengths through pooling layers make them computationally efficient and fast to train. However, CNNs often require

large labeled datasets to perform well, which can be a challenge when data is limited. Additionally, while they are good at capturing local context, CNNs may struggle with understanding long-term dependencies that are important for capturing the overall sentiment of a text. Kim [2014] helped popularize CNNs for text classification tasks, including sentiment analysis.

- **Recurrent Neural Networks (RNN)**

  Recurrent Neural Networks (RNNs) are designed to process sequential data by maintaining a hidden state that carries information from previous steps, making them ideal for sentiment analysis, where word order affects meaning. Basic RNNs, known as Vanilla RNNs, calculate the hidden state h_t at time step t as follows:

$$h_t = f(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h)$$

  where f is an activation function (such as tanh), W_h and W_x are weight matrices, x_t is the input at time t, and b_h is a bias term. However, Vanilla RNNs struggle with capturing long-term dependencies due to issues like vanishing or exploding gradients. To address this, Long Short-Term Memory (LSTM) networks were developed, using gating mechanisms to manage information flow and effectively capture long-range dependencies. The LSTM cell is governed by three gates: the input gate i_t, the forget gate f_t, and the output gate o_t:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

  where σ is the sigmoid function, and W terms are weight matrices for each gate. The cell state c_t is updated as:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

  and the hidden state h_t is given by:

$$h_t = o_t \cdot \tanh(c_t)$$

  Gated Recurrent Units (GRUs), a simpler variant of LSTMs, offer better computational efficiency while still handling long-term dependencies well.

  RNNs, especially LSTMs and GRUs, are highly effective for tracking the flow of sentiment across sentences or documents, making them useful for variable-length text sequences. However, they can be challenging to train due to the gradient problems mentioned earlier, and they also require significant computational resources, particularly for processing long sequences or large datasets. Despite these challenges, LSTMs have become widely used in sequential data tasks, including sentiment analysis, since their introduction by Hochreiter and Schmidhuber in 1997.Hochreiter and Schmidhuber [1997]

I tried to use deep learning models in my thesis but due to the complexity of the models and the limited size of my training set, which consists of 1,300 manually annotated tweets, I did not achieve good

results, but I reported them anyway for completeness in chapter 4. Deep learning models, such as LSTMs and CNNs, typically require large datasets to effectively learn feature representations. With a smaller dataset, there is a high risk of overfitting, where the model becomes too specialized to the training data and struggles to generalize to new, unseen data. Additionally, deep learning models demand substantial computational resources, which would be impractical given the size of my dataset and the scope of my research and the computational power I have available. Therefore, to avoid these challenges, I focused on integrating lexicon-based approaches with machine learning techniques that are more suitable for smaller datasets, ensuring better generalization and computational efficiency.

2.7 Evaluation Metrics

Evaluation metrics are essential for assessing the performance of sentiment analysis models, providing quantitative measures that help in understanding how well a model performs, identifying areas for improvement, and comparing different models. The choice of evaluation metric depends on the specific goals and characteristics of the sentiment analysis task. Key evaluation metrics used in sentiment analysis include accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC.

- **Accuracy** is a straightforward metric that measures the proportion of correctly predicted instances out of the total instances. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. While simple and intuitive, accuracy can be misleading for imbalanced datasets where one class is much more frequent than others. In the context of sentiment analysis, accuracy indicates the overall correctness of the model in predicting sentiments (positive, negative, or neutral).

- **Precision** and **recall** are more nuanced metrics that provide deeper insights into a models performance.

  - Precision is the proportion of correctly predicted positive instances out of the total predicted positive instances, which is particularly useful when the cost of false positives is high. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  - Recall, or sensitivity, measures the proportion of correctly predicted positive instances out of the total actual positive instances, which is important when the cost of false negatives is high. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  - The F1-score is the harmonic mean of precision and recall, balancing the trade-off between these two metrics and providing a single measure of a models performance. It is particularly useful for imbalanced datasets in sentiment analysis and is calculated as:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- The **confusion matrix** is a comprehensive tool that provides a detailed breakdown of a models performance by showing the actual versus predicted classifications. It includes True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN), allowing the calculation of various metrics like accuracy, precision, recall, and F1-score. In sentiment analysis, the confusion matrix helps in understanding the distribution of correctly and incorrectly classified sentiments.
- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)** is a performance measurement for classification problems at various threshold settings. The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold levels, where:

  $$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

  $$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

  The AUC (Area Under Curve) represents the degree or measure of separability and provides an aggregate measure of performance across all classification thresholds, making it useful for comparing different models. However, ROC-AUC can be misleading for highly imbalanced datasets.

Evaluation metrics are critical for comprehensively and accurately assessing the performance of sentiment analysis models. Accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC each offer unique insights into different aspects of model performance.

Chapter 3

Literature Review

3.1 Literature review of Sentiment Analysis

3.1.1 Lexicon based model

VADER and Lexicon-based model

The origins of sentiment analysis can be traced back to the early 2000s when the growing accessibility of large-scale textual data, such as online reviews and social media postings, created interest in the creation of automated methods for opinion extraction. In its early stages, sentiment analysis primarily relied on lexicon-based models. These models made use of dictionaries containing words that were labeled with sentiment scores of positive, negative, or neutral. The process involved counting the occurrences of words within a text that are associated with some sentiment and summing their associated scores to determine the overall sentiment.

One of the most well-known lexicon-based models is the Valence Aware Dictionary and sEntiment Reasoner (VADER), introduced by Hutto and Gilbert [2014]. VADER was specifically designed for short, informal text, making it particularly effective for analyzing social media content like tweets. Unlike traditional lexicon-based models, VADER not only assigns polarity (positive, negative, or neutral) but also captures the intensity of sentiment, providing a compound score that reflects the overall sentiment strength.

VADER assigns sentiment scores to each word in a given text based on its dictionary of predefined valence scores. For example, words such as "good" or "excellent" have positive valence, while "bad" or "terrible" have negative valence. Additionally, VADER accounts for degree modifiers (also known as intensifiers) that amplify or dampen the sentiment intensity. For instance, "very good" is scored more positively than "good" alone, while "slightly bad" has a less negative score than "bad".

The overall sentiment score S for a sentence in VADER is calculated using the following steps:

- **Identify Sentiment-Intensive Words:** Each word w in the sentence is assigned a valence score $V(w)$ from VADER's lexicon.
- **Apply Rules for Intensifiers and Punctuation:** - Intensifiers such as "very" or "extremely" modify the valence score of adjacent words. For example, if a word w has an intensifier, its score $V(w)$ is multiplied by a factor (e.g., 1.5 for "very" or 0.5 for "slightly"). - Exclamation marks (e.g., "!!!") amplify the sentiment intensity, increasing the valence score of a sentence. Similarly, words

in all uppercase letters are given extra weight.

- **Calculate the Compound Score:** - The compound score C represents the normalized sum of the valence scores of each word in the sentence. It is calculated as:

$$C = \frac{\sum_{i=1}^{n} V(w_i)}{\sqrt{\sum_{i=1}^{n} V(w_i)^2 + \alpha}}$$

  where $V(w_i)$ is the valence score of word w_i, n is the number of words in the sentence, and α is a smoothing parameter to prevent division by zero. The compound score C ranges from -1 (most negative) to +1 (most positive), providing an overall measure of the sentence's sentiment.

- **Classify Sentiment Based on Compound Score:** - Based on the compound score C, VADER classifies the sentiment as positive, neutral, or negative: - Positive: $C > 0.05$ - Neutral: $-0.05 \leq C \leq 0.05$ - Negative: $C < -0.05$

I chose VADER for my sentiment analysis because it is specifically designed for short, informal texts, making it ideal for analyzing Twitter data. Given that my dataset consists of tweets, VADERs ability to handle abbreviations, slang, emoticons, and informal language makes it a strong choice for accurately capturing the sentiment expressed in these posts. Additionally, VADER not only detects positive or negative sentiment but also captures the intensity of that sentiment, which is particularly important when analyzing financial markets, where extreme emotions can have a more pronounced impact on market volatility.

While VADER offers clear advantages, particularly for the nature of my data, there are also hybrid approaches that combine lexicon-based models like VADER with more sophisticated machine learning techniques to enhance performance. One such example is the hybrid model proposed by the paper LSTM, VADER, and TF-IDF based Hybrid Sentiment Analysis Model Chiny et al. [2021]. This hybrid approach integrates VADER with machine learning models to improve accuracy while addressing the challenge of needing large labeled datasets.

The authors of the paper combined three distinct models , VADER, and TF-IDFalong with various classification algorithms such as Logistic Regression, Random Forest, and Support Vector Machine (SVM) to calculate sentiment scores and classify them. This method achieved an accuracy of around 75% with a relatively small training dataset, demonstrating how hybrid models can enhance performance by leveraging the strengths of both lexicon-based and machine learning approaches.

Class imbalance is a common issue in sentiment analysis, as well as in classification problems in general, and it is something I also encounter in my work, as I will explain later. Addressing this imbalance is crucial for improving the accuracy of classification models. One paper that tackles this issue effectively is The Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks **?**. The author employs the Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for the minority class before applying Convolutional Neural Networks (CNNs) for feature extraction.

The paper highlights how addressing class imbalance with SMOTE led to superior performance compared to models that ignored this issue. The combination of SMOTE and CNNs allowed the author to achieve more balanced results, enhancing the models ability to correctly classify instances from both the majority and minority classes. This approach is particularly relevant for sentiment analysis, where class imbalance can significantly affect the models performance.

SentiWordNet

SentiWordNet, developed by Esuli and Sebastiani [2006], is a widely used lexicon-based tool for sentiment analysis, built upon the WordNet lexical database. It assigns sentiment scores positive, negative, and neutral (objective) to each synset, or group of similar words, allowing for a more balanced sentiment evaluation compared to simple polarity-based lexicons. Each synset s in WordNet is associated with three sentiment scores:

$$\text{Pos}(s), \quad \text{Neg}(s), \quad \text{Obj}(s)$$

where $\text{Pos}(s)$, $\text{Neg}(s)$, and $\text{Obj}(s)$ represent the positive, negative, and objective (neutral) sentiment intensities, respectively, for synset s. These scores range from 0 to 1 and satisfy the constraint:

$$\text{Pos}(s) + \text{Neg}(s) + \text{Obj}(s) = 1$$

This approach enables SentiWordNet to handle semantic ambiguity by assigning different sentiment scores to each synset of a word. For example, the word "close" in "close to victory" (indicating proximity and possibly a positive sentiment) would have a different synset and corresponding sentiment score than "close the door" (indicating an action with neutral sentiment).

For a given text or sentence, the overall sentiment score can be computed by averaging the sentiment scores of the individual synsets associated with each word. For a sentence T containing words $w_1, w_2, \ldots, w_n$, the overall positive, negative, and objective scores can be calculated as:

$$\text{Pos}(T) = \frac{1}{n} \sum_{i=1}^{n} \text{Pos}(s_i)$$

$$\text{Neg}(T) = \frac{1}{n} \sum_{i=1}^{n} \text{Neg}(s_i)$$

$$\text{Obj}(T) = \frac{1}{n} \sum_{i=1}^{n} \text{Obj}(s_i)$$

where s_i represents the synset of word w_i in the context of sentence T.

One of SentiWordNets key strengths is its ability to differentiate between various meanings of a word. This feature is particularly valuable in applications like opinion mining, where the goal is to detect nuanced sentiments in text. By assigning separate sentiment scores to different synsets, SentiWordNet can handle ambiguous words more accurately, improving the precision of sentiment analysis in contexts like product reviews, social media, and other user-generated content. It has also been widely applied in financial sentiment analysis, where its ability to analyze sentiment in financial news and reports provides insights into market trends and investor sentiment.

Despite its advantages, SentiWordNet faces limitations in dealing with highly context-dependent language, such as sarcasm, irony, or negation, which are difficult to capture through lexical scoring alone. Moreover, while it performs well in general-purpose sentiment analysis, its effectiveness diminishes in domain-specific applications without further customization.

AFINN

AFINN, developed by Nielsen [2011], is a lexicon-based sentiment analysis tool specifically designed for analyzing social media content. It assigns integer sentiment scores $S(w)$ ranging from -5 to +5 to words, where negative values indicate negative sentiment and positive values indicate positive sentiment. This simple scoring system makes AFINN highly efficient for large-scale text analysis, especially on platforms like Twitter and Reddit. Its design focuses on handling informal language, such as slang and misspellings, which are common in social media, making it well-suited for real-time sentiment analysis applications.

For a given sentence or text T containing words $w_1, w_2, \ldots, w_n$, the overall sentiment score $S(T)$ is calculated by summing the individual sentiment scores of the words:

$$S(T) = \sum_{i=1}^{n} S(w_i)$$

where $S(w_i)$ is the sentiment score assigned by AFINN to word w_i. The resulting score $S(T)$ provides a quick overall sentiment measure for the text, with positive values indicating positive sentiment, negative values indicating negative sentiment, and a score of zero suggesting a neutral sentiment.

AFINNs key strengths include its simplicity and speed, allowing it to efficiently process large amounts of data, such as in social media monitoring or consumer product analysis. Its ability to track shifts in public sentiment, such as during the COVID-19 pandemic, highlights its adaptability to different domains. AFINN is continually updated to include modern slang and new words, ensuring its relevance in evolving online conversations. However, it is limited by its inability to capture complex linguistic constructs like sarcasm or irony, and its word list, while updated, may not fully cover domain-specific language.

In comparison to other lexicon-based models like SentiWordNet and VADER, AFINN stands out for its simplicity. While SentiWordNet assigns multiple sentiment scores across different dimensions and VADER handles emoticons and emojis, AFINN offers a more lightweight and faster alternative for large-scale sentiment analysis. Although it may lack depth in handling nuanced sentiment, AFINN remains an effective tool for real-time analysis in informal textual domains, making it valuable for both researchers and businesses.

Loughran-McDonald Sentiment Word Lists

The Loughran-McDonald Sentiment Word Lists, introduced by Tim Loughran and McDonald [2011], are specifically made for sentiment analysis in financial texts. Unlike general-purpose lexicons such as VADER or AFINN, which are designed for everyday language, the Loughran-McDonald lexicon focuses on the unique language used in financial documents like 10-K filings, earnings calls, and financial news. In these texts, words that may be neutral in a general context often carry significant sentiment in the financial domain. For example, terms like "liability" or "cost" are associated with negative sentiment in financial reporting, making this lexicon more accurate for analyzing corporate disclosures and investor communications.

A key feature of the Loughran-McDonald lexicon is its classification of words into sentiment categories such as negative, positive, uncertainty, litigious, and constraining. For a given document D, the frequency of each category of words can be computed as follows:

$$\text{Count}_{\text{category}}(D) = \sum_{w \in \text{category}} f(w, D)$$

where $f(w, D)$ represents the frequency of word w in document D, and category can be any one of the sentiment categories (e.g., negative, positive). This allows analysts to compute sentiment scores for each category, giving a detailed breakdown of sentiment in financial texts.

To derive an overall sentiment measure for the document, a net sentiment score can be calculated by taking the difference between the counts of positive and negative words, normalized by the total number of words in the document:

$$\text{Net Sentiment}(D) = \frac{\text{Count}_{\text{positive}}(D) - \text{Count}_{\text{negative}}(D)}{\text{Total Words}(D)}$$

This granularity is particularly useful in financial analysis, where the tone of corporate filings, such as earnings reports, can provide critical insights into a company's performance, risks, and financial health. For example, the frequent use of negative or uncertainty-related terms in financial disclosures can signal potential financial distress, affecting stock price movements and influencing investor behavior.

However, despite its effectiveness in formal financial documents, I chose not to use the Loughran-McDonald lexicon in my thesis for several reasons. Firstly, the lexicon's classification into multiple sentiment categories may add unnecessary complexity, especially for the analysis of tweets and how my dataset is constructed. Secondly, its focus on corporate communications limits its applicability to informal text like tweets. Since social media platforms like Twitter involve slang, abbreviations, and other informal expressions, this lexicon lacks the flexibility required for analyzing sentiment in such informal contexts. Thus, more appropriate sentiment analysis tools, like VADER, better capture the nuances of social media sentiment in my opinion.

3.1.2 Machine learning

When Machine Learning models achieved popularity, they began to dominate the sentiment analysis field as well, providing more robust solutions. In these methods, algorithms like Naive Bayes, Support Vector Machines (SVMs), and Random Forests are trained on large datasets labeled with sentiment categories. These models learn to identify patterns in the data, enabling them to classify unseen text into sentiment categories, I explained them in greater detail in section 2.5. For example, Pang et al. [2002] were among the first to apply machine learning algorithms to sentiment analysis in movie reviews, achieving significant improvements over lexicon-based methods. However, these classical machine learning methods still required substantial preprocessing and feature engineering, such as converting text into numerical representations using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or Bag of Words (BoW).

Key Algorithms in Machine Learning Sentiment Analysis

Naive Bayes has long been one of the go-to algorithms, mainly because its straightforward and efficiency. Its built around Bayes' theorem and assumes that every word or phrase in a text independently contributes to whether a piece of writing is classified as positive or negative. This simplicity makes Naive Bayes great for quick, effective training, especially with small datasets. Its major downside is that it assumes words do not affect each other, which isn't always true in real-world language. Words often

work together in ways that Naive Bayes can't quite capture, which can make it less accurate for more complex text analysis Pang et al. [2002].

Support Vector Machines (SVMs), introduced by Joachims [1998], take a more sophisticated approach. SVMs are designed to handle large sets of features, which is what you get when you turn text into numbers through techniques like TF-IDF or word embeddings. Research shows that SVMs often outperform simpler models like Naive Bayes, especially when dealing with datasets that are large and have many dimensions. The trade-off is that they need more computational power and fine-tuning.

Random Forest is another popular choice in sentiment analysis, particularly for handling large datasets. Its an ensemble method that creates multiple decision trees during training and then averages the results, which helps improve both accuracy and consistency. In sentiment analysis, Random Forest are very used because they balance bias and variance, which helps avoid the overfitting that can occur with single decision trees or simpler models. Studies, including those by Breiman [2001] and Pang and Lee [2008], have shown that Random Forest is highly effective when working with complex text data, where relationships between words aren't always straightforward.

Challenges with Classical Machine Learning Models

Classical machine learning models have provided foundational approaches to sentiment analysis, but they come with notable challenges. One of the primary limitations is their reliance on manual feature engineering, particularly using techniques like Bag of Words (BoW). These methods require significant preprocessing, often resulting in sparse matrices that fail to capture the semantic relationships between words. For instance, similar words like "happy" and "joyful" may be treated as independent features, losing valuable context. This issue is solved by more advanced methods, such as word embeddings like Word2Vec, GloVe, and TD-IDF, which address this issue by generating dense vectors that capture the relationships between words, making them more contextually aware and efficient for handling sentiment analysis tasks (Mikolov et al. [2013b] and **?**).

Another major challenge of classical models is their inability to capture context. Algorithms like Naive Bayes and SVMs struggle with handling context-dependent sentiment, such as sarcasm, negation, or sentiment shifts within a sentence. These models treat words as independent entities, which limits their capacity to understand complex expressions. In contrast, more advanced deep learning models like LSTMs and Transformers are designed to handle sequential data, allowing them to capture sentiment shifts based on word order and dependencies Vaswani et al. [2017]. Lastly, classical models are prone to overfitting, especially when trained on small or imbalanced datasets.

Overall, while classical machine learning models have been useful in sentiment analysis, their limitations, especially in feature engineering and context understanding, have left room for the development of more sophisticated models.

3.1.3 Deep learning

Deep learning models brought sophisticated way to handle textual data. The main models are Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and more advanced architectures like Transformer models and BERT. These models possess the ability to capture long-term dependencies and context in sentences, enabling them to outperform traditional methods in various NLP tasks, including sentiment analysis.

According to **?**, LSTMs have proven to be a powerful tool for text-based tasks that involve complex dependencies. In the context of sentiment analysis, LSTM models have been shown to perform better

than traditional machine learning algorithms, such as logistic regression and support vector machines (SVMs), especially on large-scale datasets, as demonstrated by Zhang et al. [2018].

Building on the LSTM architecture, Bidirectional LSTMs (BiLSTMs) process sequences in both forward and backward directions, thereby capturing context from both sides of a word or phrase. This bidirectional approach improves performance in tasks where the context surrounding a word is important.

Gated Recurrent Units (GRUs) are another variant of RNNs that simplify the LSTM architecture by using fewer parameters while still maintaining the ability to capture long-term dependencies. GRUs are often preferred for real-time sentiment analysis tasks due to their computational efficiency and faster training times. While Chung et al. [2014] found that GRUs perform comparably to LSTMs in many sequential tasks, their reduced complexity can make them more suitable in cases where computational resources are limited.

More recently, Transformer-based models, particularly BERT (Bidirectional Encoder Representations from Transformers), have set new benchmarks in sentiment analysis. Unlike RNNs, which process sequences step-by-step, Transformer models use an attention mechanism to capture the relationships between words in a sentence regardless of their distance from each other. This makes Transformers particularly effective for understanding complex language phenomena, such as negation, sarcasm, and multi-layered sentiments, often encountered in social media data or financial texts. Transformers have not only outperformed traditional machine learning models but also offered a scalable solution for handling large unstructured datasets, making them invaluable for modern sentiment analysis tasks.

These models have significantly enhanced the extraction of insights from textual data by capturing long-term dependencies and understanding nuanced language. Their application in financial markets, as shown by Bollen et al. [2011] and others, demonstrates the potential of using public sentiment, particularly from platforms like Twitter, to predict market movements. However, despite their advantages, these models require substantial computational resources.

In my case, I opted not to use models like LSTM or BERT due to their high computational demands, which would have been impractical. Moreover, the sentiment classes in my study (negative, positive, neutral) are relatively straightforward, not necessitating the complexity of deep learning models. Additionally, simpler models like Random Forest and logistic regression, combined with VADER, are more efficient for my use case and provide great performance without requiring the extensive computational power of deep learning models.

3.2 Literature review of Sentiment Analysis in Financial Markets

In this section, I will outline key literature regarding the application of sentiment analysis in the financial field.

The paper "Approaches, Tools, and Applications for Sentiment Analysis Implementation" by D'Andrea et al. [2015] provides a comprehensive overview of all applications of sentiment analysis, explaining in detail the various approaches, and tools. The authors categorize sentiment classification methods into three main approaches: machine learning, lexicon-based, and hybrid approaches. Machine learning models rely on training data to predict sentiment, while lexicon-based methods use predefined word lists, and hybrid models combine the two for enhanced performance.

The authors conclude by stating the challenges of ambiguity, such as detecting sarcasm and irony, and

emphasize the importance of hybrid approaches in improving sentiment classification performance across different domains.

As stated above, most of the papers focus on the relationship between sentiment and stock market movements, and the most important one is by Bollen et al. [2011] which tried to establish a link between public sentiment and stock market fluctuations, specifically examining whether public mood could predict stock market performance. The study used a dataset consisting of 9.85 million tweets posted by approximately 2.7 million users over the period from February 28, 2008, to December 19, 2008. Rather than developing a new classification model to analyze the sentiment of the tweets, the authors used pre-existing tools. They used OpinionFinder (OF), which classifies tweets into two categories, positive and negative, and the Google-Profile of Mood States (GPOMS), which identifies six mood dimensions: Calm, Alert, Sure, Vital, Kind, and Happy.

Regarding the relationship between public sentiment and stock market movements, Bollen et al. relied on a Granger causality test to determine whether public mood could predict changes in the Dow Jones Industrial Average (DJIA). The results showed that the Calm dimension of public mood was significantly correlated with DJIA changes, with a time lag of 2-6 days. This led to the conclusion that a calm public mood could be predictive of positive stock performance in the following days. Unexpectedly, the OpinionFinder classification did not yield significant results.

Additionally, the study used a Self-Organizing Fuzzy Neural Network (SOFNN) to model the relationship between public mood and the stock market. By incorporating the Calm dimension, the authors achieved an accuracy of 86.7% in predicting the direction of stock market movement.

Another widely used source of text in financial sentiment analysis is financial news articles, as demonstrated in the influential study by Tetlock [2007]. This research investigates how negative news impacts stock market behavior. Tetlock used a Vector Autoregression (VAR) model to estimate the relationship between negative news sentiment and stock returns.

The study collected a large dataset of news articles spanning 16 years, from 1984 to 1999. For each article, a daily pessimism score was calculated as the proportion of negative words relative to the total number of words. This sentiment score served as the basis for analyzing the impact of negative news on stock market performance.

Tetlock's findings indicate that negative news sentiment leads to a temporary decline in stock market returns. However, the effect was found to be short-lived, suggesting that the market tends to recover after an initial negative reaction. In addition, the study revealed that extreme pessimism in news articles caused a noticeable decrease in trading volume, indicating that highly pessimistic news not only impacts stock prices but also reduces market activity.

Li et al. [2014] further explored the role of sentiment analysis in predicting stock price movements by leveraging a news archive from FINET, a major financial news provider in Hong Kong. The authors employed two dictionary-based models for sentiment analysis: the Harvard IV-4 psychological dictionary and the LoughranMcDonald financial sentiment dictionary. They implemented six different stock price prediction models, comparing the effectiveness of the sentiment analysis against bag-of-words (BoW) approaches.

The aim of this study, conducted on a five-year dataset containing stocks from multiple sectors, was to determine whether models incorporating sentiment analysis would outperform traditional BoW models. These traditional models, referred to in the paper as the baseline, predict stock price returns based purely on word frequency and statistical patterns in news articles, without accounting for the emotional

or sentiment content.

The authors' findings suggest that sentiment-based models offered notable improvements in prediction accuracy at the individual stock, sector, and index levels. Moreover, the study concluded that sentiment polarity alone, simply distinguishing between positive and negative sentiments, did not provide sufficient predictive power, highlighting the importance of a more nuanced sentiment analysis.

The studies by Bollen et al. (2011), Tetlock (2007), and Li et al. (2014), while foundational, have several limitations that reduce their broader applicability and predictive power in more complex financial contexts. Bollen et al. (2011) rely heavily on a single mood dimension, Calm, to predict stock market movements, which raises concerns about the robustness of the findings. The overemphasis on one dimension overlooks the potential impact of other mood states on financial markets, limiting the model's capacity to capture a more holistic view of public sentiment. Furthermore, their reliance on a Granger causality test only establishes correlation, not causation, leaving open the question of whether public sentiment drives market performance or vice versa. The use of pre-existing tools like OpinionFinder and GPOMS, while practical, may not be fully optimized for financial contexts, leading to potentially inaccurate sentiment classifications.

Similarly, Tetlocks (2007) focus on negative sentiment alone is a major limitation. While negative news certainly influences market behavior, ignoring the effects of positive or neutral sentiment leaves out critical components of market dynamics. Additionally, Tetlock's sentiment analysis method is quite basic, relying on simple word-counting techniques to measure pessimism. This approach fails to account for more complex linguistic features like negations, context, and intensity, which are important for accurately assessing the sentiment of news articles. Furthermore, the reliance on Vector Autoregression (VAR) models assumes linear relationships between sentiment and stock market movements, potentially missing non-linear dynamics that are often present in financial markets.

Li et al. (2014) also present some limitations, particularly as their focus on the Hong Kong market limits the generalizability of their findings. The sentiment dynamics and market behavior in Hong Kong may differ significantly from those in other global financial markets. Moreover, their study suggests that sentiment polarity alone (positive vs. negative) is insufficient for accurate prediction, highlighting the need for more granular sentiment analysis. However, the authors fail to propose or implement more advanced methodologies to address this issue.

3.3 Literature review of Sentiment Analysis and market Volatility

This section extensively reviews key studies on the relationship between sentiment analysis and market volatility, focusing specifically on sentiment's predictive power for volatility measures like the VIX. The analysis presented here demonstrates that both positive and negative sentiment from various sources, such as news articles or social media, can influence market volatility and can serve as early indicators of market shifts. These research findings highlight the close connection between sentiment fluctuations and changes in investor attitudes, revealing that negative sentiment often foreshadows increased volatility. Additionally, the literature underscores the evolution of sentiment analysis techniques, evolving from basic sentiment classification to more advanced machine learning models, to enhance the accuracy of volatility predictions.

Smales [2016] expanded the application of sentiment analysis by directly examining its effects on the VIX, particularly in times of high uncertainty like financial crises. The study analyzed sentiments extracted

from financial news articles to understand how positive and negative sentiments impacted changes in the VIX in terms of both direction and magnitude. The sentiment data was extracted from RavenPack, a tool that uses natural language processing (NLP) to categorize news headlines as positive, neutral, or negative. Each headline received a sentiment score through RavenPacks Multi-Classifier for Equities (MCQ), with positive news scored as +1, neutral as 0, and negative as -1. These sentiment scores were aggregated daily and weighted by the size of firms in the S&P 500 to create an overall news sentiment measure for each trading day.

Smales employed an ordinary least squares (OLS) regression model to investigate the contemporaneous relationship between changes in the VIX and news sentiment, testing both aggregate news sentiment and disaggregated positive and negative sentiment separately. The results showed that negative news sentiment often preceded spikes in the VIX, indicating heightened investor fear and uncertainty, while positive sentiment tended to dampen expectations of future volatility. This research was significant for establishing a direct connection between sentiment analysis and the VIX, diverging from previous studies that mainly focused on stock prices. By concentrating on periods of increased market uncertainty, such as the 2008 financial crisis, the study offered valuable insights into how sentiment functions in unstable markets, highlighting negative sentiment as an early indicator of sharp volatility spikes.

The main limitation of this study is that it relied on financial news sentiment, which, though informative, might not completely capture the real-time sentiment of market participants since news articles often respond to past events. This could lead to delays in predicting VIX changes, especially in fast-paced markets. Additionally, Smales used relatively basic sentiment analysis techniques, focusing on a simple positive or negative classification. This simplistic approach may not fully capture investor emotions. Integrating more advanced models, like those based on deep learning, or considering other aspects of sentiment, such as uncertainty or fear, could provide deeper insights into forecasting volatility.

In the research paper "Predicting Financial Markets: Comparing Survey, News, Twitter, and Search Engine Data" Mao et al. [2011], the study investigates various online sentiment sources to predict financial markets, including the Dow Jones Industrial Average (DJIA) and market volatility (VIX). The authors compared traditional sentiment indicators like investor surveys with modern data sources such as Twitter sentiment, news headlines, and Google search volumes. Their main goal was to evaluate how well these different sentiment indicators could predict market returns and volatility. Using simple methods like correlation analysis and regression models, the study found that social media sentiment, especially from Twitter, and search engine data was highly effective in predicting market outcomes, often outperforming traditional sentiment measures.

Despite the paper's significant progress in incorporating new data sources, there are several limitations in the methodology and models used. The sentiment analysis mainly relied on a basic lexicon-based method focusing on keywords like "bullish" or "bearish." This approach oversimplifies the complexity of financial sentiment, potentially missing nuances such as sarcasm or changing sentiment trends over time. Then, the research employed simple regression models, which, while effective in finding correlations, may not fully capture the nonlinear dynamics in financial markets. Implementing more advanced machine learning techniques could improve the accuracy of their predictive models. Moreover, the study heavily relied on historical data and correlations, which might not reflect real-time market conditions, reducing its relevance in fast-paced markets.

In my thesis, I use more advanced sentiment analysis techniques, including VADER, along with machine learning models like Random Forest and XGBoost. These advanced models are better at interpreting sentiment in financial discussions, which leads to more reliable and precise predictions. An important improvement in my thesis is the full integration of GARCH-based features into the predictive model.

Unlike Mao et al., who ignored volatility clustering, a key feature of financial markets, my model includes GARCH features. This helps capture sentiment-driven insights and traditional financial volatility patterns, resulting in a more comprehensive forecast of market volatility. Additionally, my approach addresses the challenge of predicting both upward and downward market movements by also taking into account the problem of class imbalances. This has boosted prediction accuracy for both bullish and bearish trends, an area where the original study was limited.

In the paper "Stacking Hybrid GARCH Models for Forecasting Bitcoin Volatility" by Aras [2021], the authors introduce a new methodology to forecast Bitcoin price volatility by combining GARCH models with various machine learning techniques. They utilize GARCH(1,1) models to capture the changing volatility over time, a common feature in financial markets where high volatility periods are frequently followed by more volatility. This forms the basis of the model, centering on Bitcoin returns' conditional variance using historical data.

The authors extend the predictive capability of their framework by integrating various machine learning models alongside GARCH. These models encompass Artificial Neural Networks (ANN), which adeptly capture intricate non-linear relationships through interconnected neuron layers; Support Vector Machines (SVM), known for their efficacy in identifying data relationships in regression tasks; Random Forest (RF), a collective of decision trees that detect non-linear patterns and mitigate overfitting via prediction averaging; and k-Nearest Neighbors (KNN), which forecast new values based on proximity to historical data points. The hybrid models produced by those techniques are then merged utilizing a stacking ensemble technique, where a meta-learner consolidates predictions from each model. This meta-learner is trained through LASSO regression to aid in feature selection by reducing irrelevant coefficients. Additionally, Principal Component Analysis (PCA) is implemented to diminish dimensionality and retain only the most informative variables.

The data used in this study is based on Bitcoin price data over an extended period. The authors calculate daily returns from the Bitcoin closing prices and use these as inputs for the GARCH model and machine learning techniques. The primary focus is on predicting future volatility, given Bitcoins reputation for extreme price fluctuations. Volatility forecasts are generated from both historical volatility and the predictions of the machine learning models, which are fed into the stacking ensemble.

The combination of econometric models like GARCH and machine learning techniques in this paper is innovative because it takes advantage of both approaches, GARCH captures the inherent volatility clustering, while machine learning models handle the complex, non-linear relationships in the data. However, this innovation also brings certain limitations. The computational complexity of the stacking method introduces challenges, especially when it comes to real-time forecasting. Moreover, the studys exclusive focus on Bitcoin, a particularly speculative asset, may limit the generalizability of its findings to more stable financial markets.

The methodology of this study served as the inspiration for my thesis, where I created a similar hybrid GARCH method to forecast VIX volatility by utilizing Twitter sentiment analysis in place of traditional price data. While Aras concentrated on the use of past Bitcoin information for prediction purposes, I integrated real-time sentiment data sourced from Twitter to gauge market sentiment as an initial predictor of volatility. This enabled me to investigate the impact of public sentiment on market dynamics, introducing a new perspective to the hybrid GARCH framework.

One key difference in my research is combining GARCH-based features with sentiment analysis to create a comprehensive framework that considers both market volatility trends and sentiment-driven perspectives. This combined approach helped me predict market volatility direction as indicated by the VIX, offering

valuable insights for decision-makers. In contrast, Aras' model focused only on predicting Bitcoin's price volatility, limiting the model's usefulness in different financial markets.

Chapter 4

Sentiment Analysis

In this chapter, I will explain in greater detail my textual dataset and how I performed the sentiment analysis in order to classify each tweet into positive, negative, and neutral.

4.1 Explanatory data analysis of the labeled tweet

4.1.1 Twitter Data

Tweets were obtained from a dataset available on IEEE DataPort, which included posts mentioning the S&P 500 index and individual companies within the top 25 constituents. Specific hashtags and stock symbols (e.g., AAPL for Apple, MSFT for Microsoft) were used to scrape the relevant tweets. The data spans from April 9 to July 16, 2020, capturing a period marked by significant market volatility due to ongoing economic uncertainties and the global pandemic. A total of 943,672 tweets were included in the dataset. These tweets were further refined by filtering out non-English tweets and retweets to ensure the analysis focused on original sentiment expressions, and 1300 of those are already labeled making it my training set.

- **Sentiment Distribution:**

  The **positive sentiments** are in the majority within the labeled tweets, about 40.6% of the tweets. The amount of positive tweets is more than the other two classes, therefore I will consider this class imbalance when training models, as it might cause the model to be biased toward the positive class.

  It was followed by **neutral** sentiment, found in approximately 32.6% of the tweets. This significant portion indicates that discussions on social media are often balanced, neither too much positive or negative. It reflects the exchange of information, observations, or discussions without strong sentiment.

  Negative sentiments constituted 26.8% of the tweets, the least represented but still significant enough to capture the voice of users expressing pessimism or negative feelings toward the market.

  Despite positive sentiment being dominant, the presence of neutral and negative tweets shows that caution and critical perspectives were also significant in social media conversations.

  The graph in Figure 4.1 provides a clear view of how sentiments were divided among the labeled tweets.

- **Distribution of Tweet Lengths:**

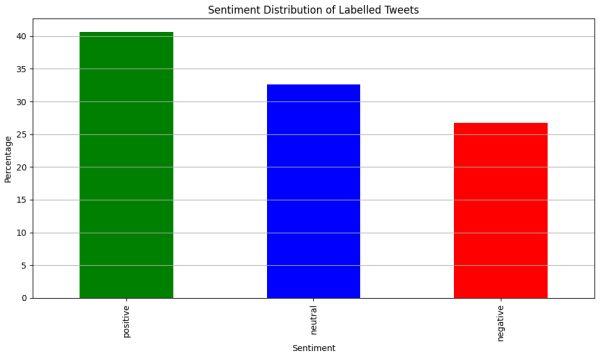

Figure 4.1: Sentiment Distribution of Labelled Tweets

The histogram in figure 4.2 shows the distribution of tweet lengths in characters. There is a significant peak around the 140-character mark, likely because Twitter originally limited tweets to 140 characters. This peak indicates that many tweets are exactly at this limit. The majority of tweets seem to be around 100 to 150 characters in length, with some longer tweets present as well, especially following Twitters increase in character limit.

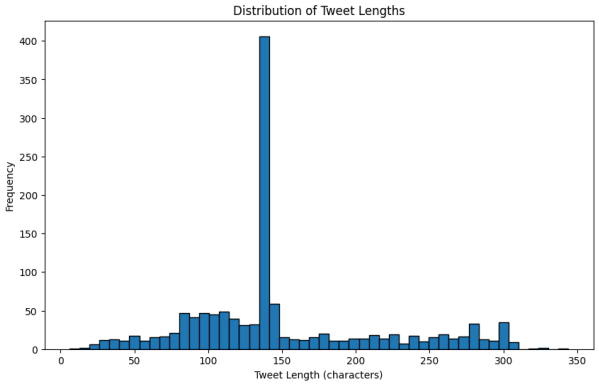

Figure 4.2: Distribution of tweet length

- **Tweet Length by Sentiment:**

  The box plot in figure 4.3 shows the distribution of tweet lengths across different sentiment classes (positive, neutral, and negative). It can be noticed that tweets with positive sentiment tend to be slightly longer on average than neutral and negative tweets. The variability in tweet length is higher for positive tweets, with several outliers present. Negative tweets also show a range of lengths but are generally shorter compared to positive tweets. Neutral tweets have a more concentrated range of lengths, with fewer outliers, indicating they tend to be more consistent in length.

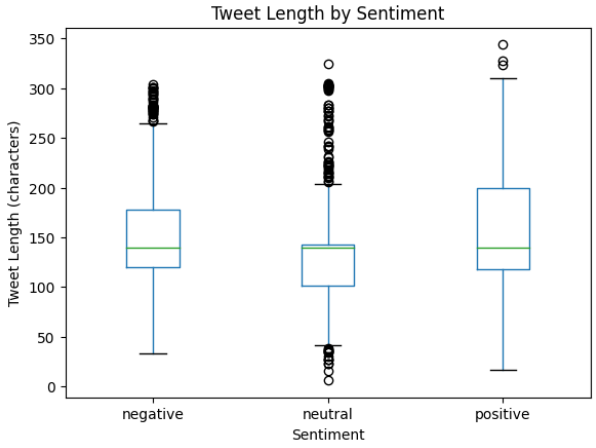

Figure 4.3: Tweet Length by Sentiment

4.2 Lexicon based approach + Machine Learning

In my analysis, I began by applying the VADER lexicon-based model, which served as my baseline. Although I initially explored deep learning models and Transfer Learning approaches, both considered state-of-the-art for sentiment analysis, I found that they were not the best fit for my dataset, as I will elaborate in a later section. As a result, I shifted my focus toward lexicon-based models and machine learning models, with the aim of combining them to optimize performance while accounting for the specific characteristics of my dataset.

Upon reviewing several studies such as D'Andrea et al. [2015] that suggest combining lexicon-based models and machine learning models for sentiment analysis, I adapted this idea to my problem by combining VADER sentiment score as a feature in my machine learning models. For my analysis, I selected TF-IDF, as I believed it was the most suitable approach for my dataset. After normalizing all features of my dataset, I fed it into a machine-learning model for classification.

To ensure completeness of my analysis, I also tested and reported results from a machine learning model that used only TF-IDF for feature extraction, alongside a deep learning model. While I considered using Transfer Learning models, they proved too computationally intensive to run on my computer.

4.2.1 VADER

I started my analysis by cleaning the dataset as explained in section 2.2 and then I first applied the VADER lexicon-based model.

The VADER sentiment analysis model was evaluated on the labeled dataset, achieving an overall accuracy of 69%, indicating that it correctly predicted the sentiment of 69% of the tweets. Detailed performance metrics, including precision, recall, and F1-score for each sentiment class (negative, neutral, and positive), are summarized in Table 4.1.

The classification report reveals that the model has the highest precision for the negative class (83%), indicating a strong ability to correctly identify them. However, the recall for the negative class is lower (54%), showing that many negative tweets were misclassified as other sentiments. For the neutral class, the recall is 71%, demonstrating that the model is fairly effective at correctly identifying neutral tweets, although the precision is somewhat lower at 64%. The positive class has the highest recall at 77%, suggesting that the model is particularly effective at recognizing positive tweets, but with a precision of 67%, it occasionally misclassifies other sentiments as positive.

Table 4.1: Classification Report for VADER Sentiment Analysis

Class	Precision	Recall	F1-Score	Support
Negative	0.83	0.54	0.65	348
Neutral	0.64	0.71	0.67	424
Positive	0.67	0.77	0.72	528
Accuracy		0.69		1300
Macro Avg	0.71	0.67	0.68	1300
Weighted Avg	0.70	0.69	0.69	1300

The confusion matrix, presented in Table 4.2 , provides a detailed breakdown of the models performance by showing the number of correct and incorrect predictions for each sentiment class.

Table 4.2: Confusion Matrix for VADER Sentiment Analysis

	Predicted Negative	Predicted Neutral	Predicted Positive
Actual Negative	187	68	93
Actual Neutral	20	300	104
Actual Positive	18	102	408

From the confusion matrix, it is evident that the model often misclassifies negative tweets as neutral (68 instances) and sometimes mistakes positive tweets as neutral (102 instances). Given that the VADER is one of the simplest model techniques to do sentiment analysis and does not even need to train the model I can conclude that is a good baseline model.

4.2.2 VADER with Random Forest

To improve the performance of the baseline VADER model, I incorporated VADER sentiment scores directly into the dataset. VADER provides four key sentiment metrics: positive (`pos`), neutral (`neu`), negative (`neg`), and a composite score (`compound`). The `compound` score, ranging from -1 (indicating highly negative sentiment) to +1 (indicating highly positive sentiment), serves as a summary metric for the overall sentiment of each tweet.

In addition to the VADER scores, I applied TF-IDF (Term Frequency-Inverse Document Frequency) to the text data, converting the tweets into numerical representations while accounting for the relative importance of each word in the corpus. This allowed the model to capture both the sentiment and specific language features within tweets, which is particularly relevant in financial contexts.

After preparing the features, I split the dataset into training (80%) and test (20%) sets. For classification, I employed a Random Forest model, an ensemble technique that combines multiple decision trees. Each tree is trained on a random subset of the data, and their predictions are aggregated by majority voting:

$$\hat{y} = \text{mode}(y_1, y_2, \ldots, y_N)$$

where $\hat{y}$ is the final prediction, and y_i represents the prediction from the i-th tree. This ensemble approach reduces variance and increases the model's robustness against overfitting, making it suitable for complex, high-dimensional datasets.

The Random Forest model achieved an accuracy of approximately 72.69%, representing a substantial improvement over the baseline. This enhanced performance highlights the models ability to learn from the hybrid feature set, distinguishing sentiment classes more accurately.

The classification report in Table **??** shows marked improvements over the baseline VADER model,

Table 4.3: Classification Report for Random Forest Sentiment Analysis

Class	Precision	Recall	F1-Score	Support
Negative	0.88	0.63	0.73	78
Neutral	0.73	0.75	0.74	88
Positive	0.65	0.79	0.71	94
Accuracy		**0.73**		
Macro Avg	0.75	0.72	0.73	260
Weighted Avg	0.74	0.73	0.73	260

particularly in negative and neutral sentiments. For example, precision for the negative class increased from 0.83 to 0.88, and recall improved from 0.54 to 0.63. Similarly, the F1-score for neutral sentiment rose from 0.67 to 0.74. These enhancements indicate that combining VADER sentiment scores with TF-IDF features enables the Random Forest model to better discern subtle variations in tweet sentiment, providing a balanced and robust tool for sentiment-based market analysis.

4.2.3 VADER with Random Forest and Grid Search

To attempt further improvement over the previous Random Forest model, I fine-tuned the hyperparameters using Grid Search. Specifically, I optimized the number of estimators (`n_estimators`), maximum depth (`max_depth`), minimum samples split (`min_samples_split`), and minimum samples per leaf (`min_samples_leaf`). The goal was to find the combination of hyperparameters that maximizes accuracy by exhaustively searching over a specified parameter grid.

Grid Search is a hyperparameter optimization technique that iterates over all possible combinations of a set of hyperparameters. For this Random Forest model, I defined the parameter grid as follows:

$$\begin{aligned}&\text{n\_estimators} : [100, 200, 300],\\ &\text{max\_depth} : [\text{None}, 10, 20, 30],\\ &\text{min\_samples\_split} : [2, 5, 10],\\ &\text{min\_samples\_leaf} : [1, 2, 4].\end{aligned}$$

This resulted in $3 \times 4 \times 3 \times 3 = 108$ combinations, which were evaluated using 5-fold cross-validation, totaling 540 model fits.

The optimized model achieved an accuracy of 72%, a slight decrease compared to the initial Random Forest model (73% accuracy). Despite this minor drop in overall accuracy, the model's performance was more balanced across individual sentiment classes, suggesting that the hyperparameter tuning led to improvements in class-specific metrics.

As shown in Table 4.4, precision for the negative sentiment class remains high at 0.88, though recall decreased slightly from 0.63 to 0.59, leading to a marginally lower F1-score of 0.71. For neutral sentiment, precision saw a slight dip to 0.71, but recall improved to 0.76, resulting in an F1-score of 0.74, similar to the previous model. The positive sentiment class showed consistency with the earlier model, maintaining a precision of 0.65 and a recall of 0.79, leading to a stable F1-score of 0.71.

Although the overall accuracy dropped slightly, the Grid Search optimization yielded a more balanced performance across sentiment classes. This balance is crucial for sentiment-based analysis, as it helps ensure that predictions are not biased toward any particular sentiment category.

Table 4.4: Classification Report of the Random Forest Model with Grid Search

Sentiment	Precision	Recall	F1-Score	Support
Negative	0.88	0.59	0.71	78
Neutral	0.71	0.76	0.74	88
Positive	0.65	0.79	0.71	94
Accuracy		0.72		260
Macro Avg	0.75	0.71	0.72	260
Weighted Avg	0.74	0.72	0.72	260

The Grid Search process, with 5-fold cross-validation for each of the 108 parameter combinations, allowed for comprehensive exploration of potential improvements to the model.

4.2.4 Random forest with Synthetic Minority Over-sampling Technique (SMOTE)

To try to address the issue of the class imbalances outlined in the explanatory analysis, I tried to use a Random Forest model trained on a dataset enhanced using Synthetic Minority Over-sampling Technique (SMOTE). SMOTE is a widely used technique in machine learning to address the issue of class imbalance, particularly in classification problems where some classes have significantly fewer instances than others. Imbalanced datasets can cause models to perform poorly on the minority classes because the algorithm tends to be biased toward the majority class, thus reducing the model's ability to generalize well on underrepresented data.

SMOTE was introduced by Chawla et al. [2002] and it is an advanced method used to deal with imbalanced dataset in classification problem. This method instead of oversampling by repeating samples, like bootstrap it created new synthetic samples. For each sample in the minority class, SMOTE identifies its k-nearest neighbors (typically k=5) within the minority class. It then generates new synthetic data points by interpolating between the original sample and one or more of its nearest neighbors. The synthetic instance is created by selecting a point along the line segment that joins the two original data points.

Then SMOTE randomly selects one of the k-nearest neighbors and generates a synthetic instance by adding a randomly weighted difference between the feature vectors of the original sample and its neighbor. Mathematically, this can be represented as:

$$\text{New sample} = \text{original sample} + \lambda \times (\text{neighbor sample} - \text{original sample})$$

where λ is a random number between 0 and 1.

I choose to use SMOTE because it avoids overfitting by increasing the diversity of the minority class creating new samples that are similar but not identical to the existing sample.

The SMOTE-enhanced Random Forest model outperformed the earlier Random Forest model with Grid Search, especially in terms of handling class imbalances. Overall, accuracy has increased slightly from 0.72 to 0.73, but the important improvements were in the recalls and F1-scores for the minority sentiment classes. For example, negative sentiment recall went up from 0.59 to 0.64, while precision slightly went down, leading to a better balance in F1-scores. The positive sentiment class saw a slight increase in precision, too, from 0.65 to 0.67, whereas recall remained the same and the F1-score increased from 0.71 to 0.72. In the case of the neutral class, both models were consistent.

Additionally, both the macro and weighted average F1-scores increased from 0.72 to 0.73, indicating further improved performance of the SMOTE-enhanced model in all classes of sentiment. Overall, SMOTE

improved the model's performance for the minority class, without losing some other critical accuracy in the majority class.

Although SMOTE improved the models balance across sentiment classes, it also introduced limitations due to its reliance on the K-Nearest Neighbors (KNN) approach. SMOTE generates synthetic samples by interpolating between a sample and its nearest neighbors, which works well in continuous data. However, for binary data like mine, this interpolation can produce samples that dont align with natural binary patterns, leading to synthetic data points that lack interpretability. This process can introduce noise, as synthetic samples may not truly represent real-world sentiment patterns in binary form. Given these limitations with binary data, I experimented with SMOTE but ultimately chose not to select it as a final approach for this analysis.

Table 4.5: Classification Report of the SMOTE Model

Sentiment	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.83	0.64	0.72	78
Neutral	0.74	0.76	0.75	88
Positive	0.67	0.78	0.72	94
Accuracy		0.73		260
Macro Avg	0.75	0.73	0.73	260
Weighted Avg	0.74	0.73	0.73	260

The SMOTE model improved performance for all sentiment classes with an average F1-score of 0.73. However, due to limitations associated with the binary target variable, it was excluded from the final model selection.

4.2.5 XGBoost Model

The next model I tried was the XGBoost (Extreme Gradient Boosting), an advanced ensemble learning algorithm that is particularly effective for handling structured data and is known for its high performance in various machine learning tasks. XGBoost is a powerful and scalable implementation of gradient boosting, a technique that builds a series of decision trees sequentially. In each iteration, the model aims to correct the errors made by the previous tree, gradually refining the predictions to minimize the overall loss function.

XGBoost works by minimizing a loss function $L(y, \hat{y})$, with each new tree in the ensemble aiming to reduce the residuals from previous trees. The general form of an additive model in gradient boosting is:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F}$$

where $\hat{y}_i$ is the prediction for data point x_i, T is the total number of trees, and each f_t is a decision tree that belongs to the function space $\mathcal{F}$. XGBoost builds each tree sequentially, with each new tree correcting the residuals of the previous predictions.

To optimize the model, XGBoost includes a regularized objective function that balances the loss function and model complexity. The objective function at each iteration t is:

$$\text{Obj} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) + \sum_{t=1}^{T} \Omega(f_t)$$

where $L(y_i, \hat{y}_i^{(t)})$ is the loss function measuring the difference between the true label y_i and the predicted label $\hat{y}_i^{(t)}$, and $\Omega(f_t)$ is a regularization term that penalizes the complexity of the trees to prevent overfitting. The regularization term includes both L1 (Lasso) and L2 (Ridge) components:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where T is the number of leaves, γ is the parameter controlling model complexity, w_j are the leaf weights, and λ is the regularization parameter. This regularization helps control overfitting, making XGBoost highly effective for complex tasks like sentiment analysis.

XGBoost further improves on traditional gradient boosting by employing efficient techniques like tree pruning, which halts the growth of branches that fail to improve model performance, and early stopping, which stops the training once there is no improvement in validation loss after a specified number of rounds. It also includes parallelization, which accelerates training by allowing trees to be built simultaneously.

In the context of this analysis, the XGBoost model was trained on a labeled dataset where the sentiment labels (negative, neutral, and positive) were first transformed into numerical values using label encoding. The model then learned to classify the tweets by iteratively improving its predictions, reducing errors made by earlier trees in the ensemble. After training, the XGBoost model achieved an accuracy of 0.74.

As shown in Table 4.6, the XGBoost model performed well across all sentiment classes. For the negative sentiment class, it achieved a high precision of 0.81, indicating that most of the tweets predicted as negative were indeed negative. However, the recall for negative sentiments was 0.67, meaning the model missed some true negative instances. The neutral class had balanced precision and recall values, with an F1-score of 0.75. The positive sentiment class achieved the highest recall at 0.82, demonstrating the model's ability to capture most positive tweets, but with a slightly lower precision of 0.68, resulting in an F1-score of 0.74. This performance highlights XGBoost's ability to balance precision and recall effectively, making it a robust choice for sentiment classification tasks.

Table 4.6: Classification Report of the XGBoost Model

Sentiment	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.81	0.67	0.73	78
Neutral	0.77	0.73	0.75	88
Positive	0.68	0.82	0.74	94
Accuracy		0.74		260
Macro Avg	0.76	0.74	0.74	260
Weighted Avg	0.75	0.74	0.74	260

This model demonstrated robust performance across all sentiment classes, with a balanced macro-average F1-score of 0.74. This performance suggests that the model is well-suited to capturing the complexities of sentiment in text data. XGBoost is also a good choice for this classification problem because it effectively handles imbalanced datasets, allows for feature importance analysis, and is highly versatile across various use cases.

4.2.6 Voting classifier combines Random Forest and XGBoost

I also tried the Voting Classifier that combines the predictive strengths of multiple models, in this case, a Random Forest (RF) and an XGBoost (XGB) classifier by aggregating their predictions to improve

overall performance, I chose to combine those two because they were the one that gave me the most promising results.

For this ensemble technique i choose to use a soft voting strategy, where the models output the predicted probabilities for each class rather than the final class label. The classifier then averages these probabilities, and the class with the highest average probability is selected as the final output. I chose soft voting over hard voting because the former takes into account the confidence level of each model, leading to more balanced results.

In this case, both the Random Forest and XGBoost models calculate probabilities for each class (positive, negative, and neutral). These probabilities are then averaged class by class, and the class with the highest average probability is selected for the final prediction.

The benefit of this technique is its robustness, as it reduces sensitivity to small variations in the data. It combines the strengths of both Random Forest and XGBoost into a single model, and it reduces overfitting since the final prediction is based on the combined output of two models rather than relying on a single one.

The Voting Classifier achieved an overall accuracy of 0.75, indicating that 75% of the instances were correctly classified. As shown in Table 4.8, the classifier exhibits strong precision for the negative class at 0.86, though its recall is somewhat lower at 0.65, resulting in an F1-score of 0.74. This suggests that while the model is effective at identifying negative instances, it misses some cases, as indicated by the recall. For the neutral class, the precision is 0.76 with a recall of 0.74, resulting in a balanced F1-score of 0.75. The positive class shows a precision of 0.68 and a high recall of 0.83, leading to an F1-score of 0.75, indicating that the model is good at identifying positive sentiments, albeit with slightly lower precision.

Table 4.7: Classification Report of the Voting Classifier

Sentiment	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.86	0.65	0.74	78
Neutral	0.76	0.74	0.75	88
Positive	0.68	0.83	0.75	94
Accuracy		0.75		260
Macro Avg	0.77	0.74	0.75	260
Weighted Avg	0.76	0.75	0.75	260

4.2.7 Voting Classifier Combining Random Forest and XGBoost

I also tried the Voting Classifier, which combines the predictive strengths of multiple models, in this case, a Random Forest (RF) and an XGBoost (XGB) classifier, by aggregating their predictions to improve overall performance. I chose to combine these two models because they provided the most promising individual results.

For this ensemble technique, I chose to use a soft voting strategy. In soft voting, the models output predicted probabilities for each class rather than final class labels. The Voting Classifier then averages these probabilities across all models, and the class with the highest average probability is selected as the final prediction. The formula for soft voting can be expressed as:

$$\mathrm{P}_{\text{final}}(C_k) = \frac{1}{M} \sum_{m=1}^{M} \mathrm{P}_m(C_k)$$

where $\mathrm{P}_{\text{final}}(C_k)$ is the final probability of class C_k, M is the number of models in the ensemble (in this

case, 2), and $P_m(C_k)$ is the probability assigned to class C_k by the m-th model. The class with the highest P_{final} is chosen as the final output.

Soft voting was preferred over hard voting because it considers each models confidence level in its predictions, leading to more balanced and nuanced results. In hard voting, each model simply votes for a class, and the class with the majority vote is selected as the final prediction, disregarding the confidence in each models predictions.

In this setup, both the Random Forest and XGBoost models compute probabilities for each class (positive, negative, and neutral). These probabilities are averaged class by class, and the class with the highest average probability is selected as the final prediction. This approach allows the Voting Classifier to leverage the individual strengths of both models, as Random Forest is robust to noise due to its ensemble of decision trees, while XGBoost captures subtle patterns through gradient boosting.

The benefit of this ensemble technique lies in its robustness, as it reduces sensitivity to small variations in the data by combining the strengths of both Random Forest and XGBoost into a single model. Additionally, this approach helps reduce overfitting because the final prediction is based on the consensus of two models rather than relying solely on one.

The Voting Classifier achieved an overall accuracy of 0.75, indicating that 75% of the instances were correctly classified. As shown in Table 4.8, the classifier exhibits strong precision for the negative class at 0.86, though its recall is somewhat lower at 0.65, resulting in an F1-score of 0.74. This suggests that while the model is effective at identifying negative instances, it misses some cases, as indicated by the recall. For the neutral class, the precision is 0.76 with a recall of 0.74, resulting in a balanced F1-score of 0.75. The positive class shows a precision of 0.68 and a high recall of 0.83, leading to an F1-score of 0.75, indicating that the model is good at identifying positive sentiments, albeit with slightly lower precision.

Table 4.8: Classification Report of the Voting Classifier

Sentiment	**Precision**	**Recall**	**F1-Score**	**Support**
Negative	0.86	0.65	0.74	78
Neutral	0.76	0.74	0.75	88
Positive	0.68	0.83	0.75	94
Accuracy		0.75		260
Macro Avg	0.77	0.74	0.75	260
Weighted Avg	0.76	0.75	0.75	260

4.3 Machine Learning Models

In this section, I discuss how I used machine learning models for the sentiment analysis classification problem. As explained earlier, before applying these models, the text data needs to be converted into a numerical format. To do this I used the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. TF-IDF was used to represent each tweet in a high-dimensional feature space by capturing the importance of words relative to their occurrence within the entire corpus. After transforming the text data into numerical features, each machine learning model was trained using five-fold cross-validation to ensure robust performance evaluation and minimize the risk of overfitting. The following subsections describe the specific models and their corresponding performance.

I choose TF-IDF to transform text into numbers because, given the nature of my dataset, which is made of short tweets that can also be noisy, this technique helps in finding the most important words and gives less weight to less important words that are not so relevant. Also, it works well with machine learning

models.

4.3.1 Naive Bayes Classifier

The first model I implemented was the Naive Bayes classifier, specifically the Multinomial Naive Bayes (MNB) variant, which is widely used for text classification tasks due to its simplicity and effectiveness with high-dimensional data. Naive Bayes is a probabilistic classifier that assumes conditional independence between features, which is a reasonable assumption in certain applications such as document classification, where word occurrences are treated as independent events.

The Multinomial Naive Bayes algorithm is particularly suited for classification problems with discrete features, such as word counts or TF-IDF scores, which makes it a natural choice for sentiment classification tasks. However, in this specific experiment, the models performance was poor. After cross-validation, the average accuracy of the Naive Bayes classifier was 53.15%.

4.3.2 Logistic Regression

Next, I experimented with Logistic Regression, a well-known linear model used for classification tasks. It is particularly effective for binary classification tasks, but can also be adapted for multiclass classification as was the case here.

I tested Logistic Regression with two different types of regularization: L1 regularization, which encourages sparsity in the model by forcing some feature weights to zero, and L2 regularization, which discourages large coefficients by penalizing extreme weights. The purpose of regularization is to prevent overfitting by simplifying the model and making it more generalizable to unseen data. In this case, the L1-regularized Logistic Regression model achieved an average cross-validation accuracy of 49.38%, while the L2-regularized Logistic Regression performed slightly better, with an accuracy of 56.92%.

Although the L2-regularized model outperformed its L1 counterpart, both models yielded relatively low accuracy scores compared to expectations. Logistic Regression typically performs well on linearly separable data; however, the performance gap in this experiment suggests that the text data might not be linearly separable in the high-dimensional TF-IDF feature space.

4.3.3 Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a robust classification algorithm that excels in high-dimensional spaces, such as those generated by TF-IDF features. SVMs aim to find the optimal hyperplane that maximizes the margin between classes, making it a particularly powerful tool for classification tasks.

Despite SVM's widespread success in text classification, the results from this experiment were less promising. The model achieved an average cross-validation accuracy of 56.69%, with individual accuracy scores ranging from 0.53 to 0.63, which is considerably lower than expected. There are several potential explanations for this performance. First, the class imbalance in the dataset might have skewed the models ability to correctly classify tweets in minority classes (e.g., negative sentiments). SVMs can be sensitive to imbalanced data, as the optimal hyperplane is influenced by the distribution of classes, and in cases where one class dominates, the model may struggle to distinguish between minority classes.

Additionally, the relatively small training set used in this experiment may have contributed to the models underperformance. SVMs, especially with linear kernels, often perform better when trained on large datasets with well-distributed class representations. In cases where the training data is limited, the model may not have enough information to accurately separate the classes.

4.3.4 Conclusion on Machine Learning Models

The performance of traditional machine learning models, including Naive Bayes, Logistic Regression, and SVM, was relatively disappointing. The class imbalances in the dataset and the small size of the training set likely contributed to the poor results. While each model has its strengthssuch as Naive Bayes' simplicity and speed, Logistic Regressions interpretability, and SVMs robustnesstheir ability to generalize effectively in this case was limited.

4.4 Deep learning models

In addition to traditional machine learning models, I tried to use deep learning techniques, which, according to the literature review, have shown promising results in text classification tasks, particularly in sentiment analysis, however, due to the reasons already mentioned above I did not achieve good results. I reported them anyway for the completeness of my analysis.

The models that I selected for this analysis after studying the literature include Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNN), Bidirectional LSTM (BiLSTM), and Gated Recurrent Units (GRU) that I already explained in Section 2.6. Each of these models is designed to handle sequential data, such as text, and provides distinct advantages in terms of capturing complex relationships between words in sentences, thus enhancing sentiment prediction performance.

4.4.1 LSTM Model

LSTM networks are a type of Recurrent Neural Network (RNN) capable of learning long-term dependencies in sequential data, making them particularly effective for text classification tasks.

The LSTM model achieved a training accuracy of 90%, indicating strong performance during the training phase. However, the validation accuracy peaked at 54.62% in the early epochs and dropped slightly afterward, suggesting that the model may be overfitting to the training data. Overfitting occurs when a model learns patterns specific to the training data that do not generalize well to new, unseen data. To mitigate this issue, I applied early stopping, a regularization technique but the validation accuracy remained relatively lower than expected.

4.4.2 Bidirectional LSTM (BiLSTM) Model

Bidirectional LSTMs extend the basic LSTM architecture by incorporating two LSTM layers, one processing the input sequence from left to right and the other from right to left. This allows the model to learn dependencies in both directions, potentially improving performance in tasks where the order of words matters. The BiLSTM model in this analysis included 100 units and a dropout rate of 0.2, similar to the standard LSTM model.

The validation accuracy of the BiLSTM model improved over the training epochs, peaking at 50.38%. It did not outperform the standard LSTM. BiLSTMs tend to work better in tasks where the context of words on both sides significantly influences meaning (e.g., Named Entity Recognition), but in this sentiment classification task, the BiLSTM did not demonstrate a clear advantage over the unidirectional LSTM.

4.4.3 Gated Recurrent Units (GRU)

GRUs are a variant of RNNs similar to LSTMs but with a simplified architecture that reduces computational complexity. GRUs have fewer parameters than LSTMs, making them faster to train while still maintaining the ability to capture long-term dependencies. I trained a GRU model with 100 units and the same dropout configuration as the LSTM model.

The GRU model achieved a validation accuracy of 46.92%, which was lower than both the LSTM and BiLSTM models. While GRUs generally perform well in tasks involving sequential data, in this case, the model underperformed relative to the LSTM. This may be due to the reduced complexity of GRUs compared to LSTMs, which might have affected the model's ability to capture nuanced patterns in the text data.

4.4.4 Comparison and Conclusion

Among the deep learning models tested, the LSTM model demonstrated the best performance with a training accuracy of 90%. However, the significant drop in validation accuracy to 54.62% suggests that the model may have overfitted to the training data, limiting its ability to generalize to new data. The BiLSTM models showed comparable validation accuracy peaking at 50.38%, neither model outperformed the LSTM.

As already mentioned deep learning models, particularly LSTMs and GRUs, require large datasets to generalize effectively. In this case the dataset is not suitable for those kinds of models, and this explain the poor performance compared to what it should be according to the literature. Also models like LSTM and BiLSTM makes them prone to overfitting, especially when the training set is small.

4.5 Model selection

After carefully considering all the options in this chapter, I decided to go with the Voting Classifier, which combines Random Forest and XGBoost, as the best model for my analysis. It delivered the best performance and balanced results across different sentiment classes. With an accuracy of 74.62%, the Voting Classifier outperformed individual models like XGBoost, Random Forest, Logistic Regression, SVM, and Naive Bayes. This ensemble approach allows me to take advantage of Random Forests robustness and XGBoosts ability to capture complex patterns and improve recall. By balancing the strengths of each model, I was able to mitigate their individual weaknesses, improving overall reliability.

Given the relatively small training set of 1,300 manually labeled tweets, combining these two models through a soft voting strategy seems the best choice. Random Forest brought stability by averaging over many decision trees, while XGBoost fine-tuned predictions through boosting and regularization, helping to prevent overfitting, which can be a major issue with smaller datasets. The integration of TF-IDF for text representation also played a key role in enhancing performance, as it captured the relative importance of words in the tweets and complemented the sentiment features provided by VADER.

When compared to other models, the Voting Classifier consistently delivered strong precision, recall, and F1-scores for negative, neutral, and positive sentiments. This made it particularly well-suited for the nuanced nature of social media text. By combining the strengths of both models, the Voting Classifier offered a well-rounded performance, reducing miss classifications and delivering the most robust results among all the models I tested. Its soft voting method allowed me to average class probabilities and leverage the confidence of each model for a more balanced final prediction, especially in sentiment-heavy contexts like financial data analysis.

I proceeded in my analysis by applying the model to the rest of the dataset. The next section will present a brief explanatory analysis of the complete dataset after having classified it.

4.6 Classified dataset

After applying the model to my whole dataset I had a dataset that spans a total of 77 days. I have an average of 12,012 tweets per day, ranging from a minimum of 4,442 tweets to a maximum of 19,101 tweets.

Regarding the distribution of sentiment classes, the majority of tweets, 415,703 or 44%, were classified as positive. A significant portion of tweets, 343,718 or 36%, were identified as neutral, and the remaining 165,552 tweets, representing 17% of the total, were classified as negative. This distribution highlights a predominance of positive sentiment, with a notable presence of neutral and negative sentiments.

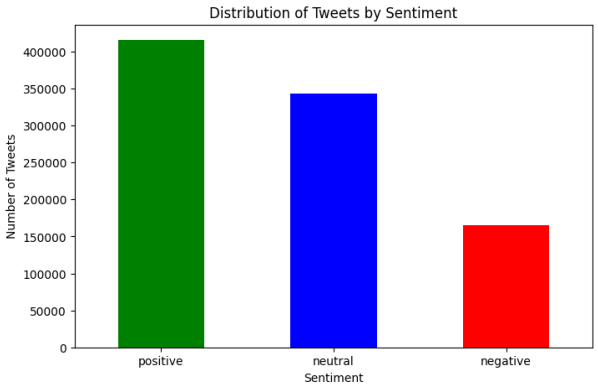

Figure 4.4: Distribution among classes

Chapter 5

VIX prediction

5.1 Introduction

This chapter will report the findings of the analysis on my main research question: do differences in sentiment relate to market volatility, as measured by VIX. The detective work begins with a review of S&P 500 and VIX financial data to see how public sentiment has impacted market performance (how the stock markets have fared) and volatility.

5.1.1 Financial Data Collection

I gathered the financial information for the S&P 500 from the Yahoo Finance API, a widely used data source. I specifically retrieved the adjusted closing prices of the index, as well as other details like opening, high, and low prices for each trading day, and the closing price. The data spans from April 9, 2020, to July 16, 2020, which coincides with the time frame of my Twitter sentiment data.

The S&P 500, made up of the top 500 U.S. companies, is a key stock market index and a main measure of U.S. market performance. In the period examined, the market was fiercely competitive and unstable, mainly due to global uncertainty from the COVID-19 pandemic. This timeframe provides insight into how market sentiment can impact markets in times of crisis.

In the provided graph 5.1, I plotted the Adjusted Close price of the S&P 500 during the specified period. The index demonstrates significant variations, displaying an upward trajectory in the latter part of the period. Despite this positive trend, distinct declines occur, signaling periods of increased uncertainty. The market fluctuations observed are in line with larger economic and political occurrences, particularly the economic impacts arising from COVID-19 and subsequent governmental actions.

The Adjusted Close price is chosen as the main metric because it includes corporate actions such as dividends, stock splits, and other factors that affect a company's stock value. This selection guarantees a more accurate depiction of the stock's true value over time, exceeding the reliability of the regular closing price.

5.1.2 VIX Data Collection

In addition to the S&P 500 data, I collected data for the VIX, often referred to as the "fear index," which reflects the market's expectations of volatility based on S&P 500 index options. The VIX data was sourced from Yahoo Finance, covering the same date range as the S&P 500 data to ensure consistency.

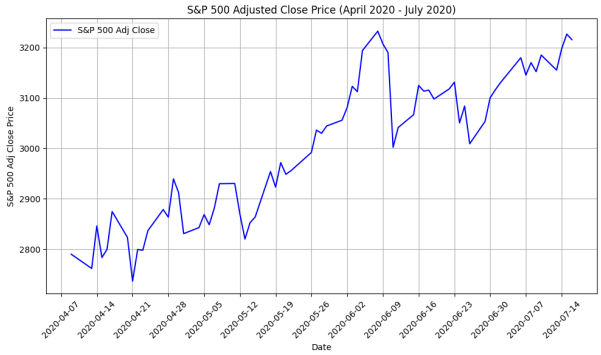

Figure 5.1: S&P 500 adjusted close price

For this analysis, the Adjusted Close price of the VIX was selected as the primary metric to represent the level of market volatility perceived by investors during this period.

The VIX, formally known as the CBOE Volatility Index, measures expected volatility over the next 30 days. Unlike most financial indices that track asset prices, the VIX derives its values from the implied volatility of options on the S&P 500. Specifically, it is calculated using the prices of out-of-the-money call and put options across various strikes and maturities. The VIX represents a weighted blend of these implied volatilities, providing a gauge of market sentiment.

The calculation of the VIX relies on the formula:

$$\text{VIX} = 100 \times \sqrt{\frac{2}{T}\sum_{i}\frac{\Delta K_i}{K_i^2}e^{RT}Q(K_i) - \frac{1}{T}\left(\frac{F}{K_0} - 1\right)^2}$$

where:

- T is the time to expiration,
- K_i represents the strike prices of the options,
- ΔK_i denotes the interval between strike prices,
- R is the risk-free interest rate,
- F is the forward price of the S&P 500,
- K_0 is the first strike price below the forward price,
- $Q(K_i)$ refers to the mid-point of bid-ask spreads for each option.

This formula captures a measure of expected volatility by weighting option prices, which contain market consensus on future volatility. Higher VIX values indicate elevated expected volatility and uncertainty, while lower values suggest confidence and stability.

The VIX receives special attention due to its tendency to rise sharply during times of market instability or uncertainty, functioning as a contrary gauge of market sentiment. When investors anticipate potential market downturns, demand for protective options increases, thereby driving up implied volatility and, consequently, the VIX. An increasing VIX usually signals growing apprehension or negativity in the market, whereas a decreasing VIX points to greater stability or optimism.

In Figure 5.2, I plot the Adjusted Close price of the VIX over the analysis period, which includes the early stages of the COVID-19 pandemic. This data reveals significant fluctuations, notably a sharp spike early

on, followed by a gradual decrease. The initial spike aligns with the early market disruption caused by the pandemic, characterized by heightened uncertainty and risk aversion. As governments implemented economic stimulus measures and public health responses, the market began to stabilize, reflected in the gradual decline of the VIX.

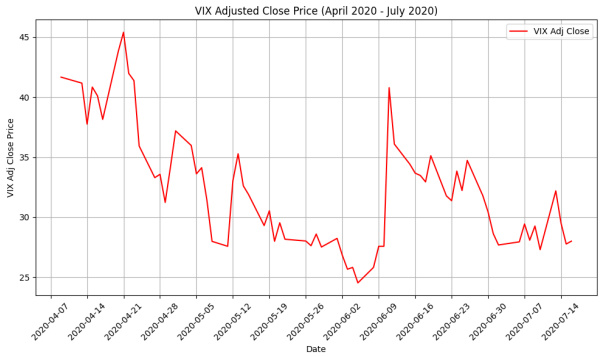

Figure 5.2: VIX Adjusted Close Price

By focusing on the VIX alongside the S&P 500 data, this analysis captures a holistic view of market sentiment during a period of significant volatility. The VIX serves as a valuable indicator for understanding investor sentiment, especially in times of crisis, and provides insight into the markets response to external economic shocks.

5.1.3 Sentiment Data Processing

The next step in the data preparation process was to integrate the financial data with the sentiment data derived from Twitter as explained in the previous section.

To prepare the sentiment data for analysis, I first grouped the tweets by date and sentiment type. For each day, I calculated the total number of positive, neutral, and negative tweets. This aggregation allowed me to derive daily percentages for each sentiment type, providing a clearer understanding of the prevailing public sentiment on each trading day.

The sentiment percentages were calculated as follows:

$$\text{Positive Percentage} = \frac{\text{Number of Positive Tweets}}{\text{Total Tweets}} \times 100$$

$$\text{Neutral Percentage} = \frac{\text{Number of Neutral Tweets}}{\text{Total Tweets}} \times 100$$

$$\text{Negative Percentage} = \frac{\text{Number of Negative Tweets}}{\text{Total Tweets}} \times 100$$

By normalizing the sentiment data in this way, I ensured that the analysis would not be skewed by the varying number of tweets each day. These percentages were then merged with the financial data for the S&P 500 and VIX, aligning by the date to create a single comprehensive dataset for analysis. This dataset included both the daily sentiment indicators and the market variables.

5.1.4 Data Alignment and Feature Creation

The integration of the sentiment and financial data required careful alignment of the time series. The sentiment data, which was based on Twitter activity, was recorded on a daily basis. The financial data, however, only includes trading days, meaning there are days in the sentiment data where no corresponding financial data exists (weekends and holidays). To handle this discrepancy, I used forward-filling techniques for the financial data, ensuring that sentiment data on non-trading days did not distort the analysis but was carried forward to the next available trading day.

In addition, I created several new features based on the sentiment data and market performance. I introduced lagged variables for both sentiment and market volatility. These lagged features were created to account for the delayed impact that sentiment might have on market behavior. These lagged variables help capture the temporal relationships between sentiment and market movements, which is essential for predictive modeling.

Another critical step in the feature engineering process was the handling of missing values. Given that the sentiment data was collected from public Twitter activity, there were certain days with lower tweet volumes, leading to missing values for certain sentiment types. I used the K-Nearest Neighbors (KNN) imputation method to fill in these missing values, leveraging the similarity between neighboring data points to estimate the missing sentiment percentages.

All feature engineering steps, including the creation of lagged variables and the imputation of missing values, will be explained in detail in the next section.

5.2 Feature engeneering

In this section, I detail the construction of key features in my analysis, particularly focusing on the creation of lagged variables, the imputation of missing values, and the reasons behind my feature choices based on insights from the literature.

5.2.1 Lagged Variables: Capturing Temporal Effects

In time-series forecasting, it is often essential to account for temporal dependencies between variables so I decided to introduce lagged variables for both sentiment and market data. Specifically, I created 1-day lagged variables for each sentiment category (positive, neutral, and negative) and for the VIX and S&P 500 Adjusted Close prices. These lagged variables capture the delayed effect that sentiment and market volatility may have on future VIX movements.

5.2.2 Why Lagged Variables Are Important

Lagged variables are a widely used method in time-series modeling to account for autocorrelation, where past values of a variable can influence its future values. According to Box and Jenkins [1976], lagging independent variables can help improve predictive accuracy by incorporating the temporal dependencies inherent in financial time-series data. For example, investor sentiment on a given day may not immediately impact market volatility but could have a delayed effect as market participants digest and react to the information. By incorporating lagged sentiment and price data, I aimed to capture this delayed response.

From the literature, Bollen et al. [2011] found that sentiment extracted from Twitter could be used to predict stock market movements several days in advance. This finding supports my decision to include

lagged sentiment variables in the analysis, as they allow the model to capture the delayed effects of market sentiment on future volatility.

Choosing a 1-Day Lag

The choice of a 1-day lag is particularly relevant to the nature of my dataset and the problem at hand. Given that both the S&P 500 and VIX data are highly volatile and influenced by daily market events, a 1-day lag ensures that my model captures short-term dependencies between sentiment and volatility. This choice aligns with studies such as Tetlock [2007], which demonstrates that media sentiment can have a near-immediate effect on market prices, with most of the impact occurring within a day. Thus, a 1-day lag seemed appropriate for my analysis to capture these immediate, but slightly delayed, effects.

In the case of the VIX, a 1-day lag in both sentiment and price data allows the model to account for the time it takes for investors to process new information and adjust their expectations of future volatility. Since the VIX is forward-looking, it is particularly sensitive to new information that reflects market sentiment, making the inclusion of lagged sentiment variables critical for accurate prediction.

5.2.3 Imputation of Missing Values

Another critical step in the feature engineering process was handling missing values in the sentiment data. Given that the sentiment data was collected from Twitter, there were certain days with lower tweet volumes, which resulted in missing values for some sentiment categories. For instance, on days with fewer than expected tweets, it was necessary to fill in the gaps to avoid bias in the model.

To address this, I employed the K-Nearest Neighbors (KNN) Imputation technique. KNN imputation is a non-parametric method that fills missing values by averaging the values of the nearest neighbors in the dataset. This technique is widely used in machine learning because it preserves the relationships between features and is robust to various data types.

Algorithm for KNN Imputation

The algorithm starts by selecting only the numerical columns that require imputation. In this case, I focused on columns related to the sentiment percentages (`neutral_percentage`, `positive_percentage`, `negative_percentage`, and `total_tweets`) as they are quantitative and therefore suited to be used with KNN.

To determine the number of nearest neighbors to consider when imputing missing values I used a value of k=5, a common choice in the literature Troyanskaya et al. [2001], where the authors demonstrated that KNN imputation is particularly effective when k is small, as it captures the local structure of the data.

KNN uses a distance metric (typically Euclidean distance) to calculate the similarity between observations. For each missing value, the algorithm identifies the *k-nearest* rows in the dataset based on their distance to the observation with missing data.

Once the nearest neighbors are identified, the algorithm imputes the missing value by averaging the values of the neighboring observations. This ensures that the imputed value is consistent with the underlying structure of the data.

The use of KNN imputation in my analysis allowed me to keep the full dataset without discarding days with missing sentiment data, preserving valuable information for model training.

$$\text{Imputed Value} = \frac{1}{k} \sum_{i=1}^{k} X_i$$

Where X_i represents the values of the k-nearest neighbors for the missing value.

The choice of KNN for imputation was driven by its ability to handle missing data in a way that preserves the relationships between variables. Unlike mean imputation, which simply replaces missing values with the mean of the column, KNN accounts for the similarity between observations, ensuring that the imputed values are contextually relevant. This was particularly important for the sentiment data, as simply replacing missing values with an average could have led to skewed results, especially in cases where sentiment on a particular day was extreme.

Incorporating KNN imputation also allowed me to avoid introducing bias into the dataset. According to Batista and Monard [2003], KNN imputation tends to outperform simpler methods like mean or median imputation, particularly in datasets where variables are highly correlated, as is the case with sentiment and market data.

5.2.4 Feature Scaling

After constructing the lagged variables and imputing missing values, I applied feature scaling to standardize the data. Feature scaling was necessary because the variables in the dataset were measured on different scales. For example, the VIX Adjusted Close price ranged from single digits to over 80, while sentiment percentages were bounded between 0 and 100. To ensure that all variables contributed equally to the model, I used StandardScaler, which transforms each feature by subtracting the mean and dividing by the standard deviation:

$$X' = \frac{X - \mu}{\sigma}$$

Where μ is the mean of the feature, and σ is the standard deviation.

Standardization is particularly important in models such as Support Vector Machines (SVM) and logistic regression, where the magnitude of the input features can significantly affect the model's performance. By scaling the data, I ensured that all features were on an equal footing, which improved the training stability and convergence speed of the machine learning algorithms.

5.3 Target Variable: VIX Direction

The main goal of my analysis is to predict the direction of the VIX, which serves as a proxy for market volatility. Rather than predicting the exact value of the VIX, I transformed the problem into a classification task by predicting the direction of the VIX, whether it increases or decreases on a given day. This approach simplifies the prediction task and aligns with real-world applications, where investors and market participants are often more interested in whether volatility will rise or fall rather than the precise level of the VIX.

5.3.1 Defining the Target Variable

The target variable, `VIX_direction`, was constructed by comparing the adjusted close price of the VIX on consecutive days. For each day t, if the adjusted close price on day t was higher than on day $t-1$,

the target was labeled as 1, indicating an increase in the VIX (higher volatility). If the adjusted close price on day t was lower than on day $t-1$, the target was labeled as 0, indicating a decrease in the VIX (lower volatility).

The formula for constructing the `VIX_direction` is as follows:

$$\texttt{VIX\_direction}_t = \begin{cases} 1 & \text{if Adj Close}_t > \text{Adj Close}_{t-1} \\ 0 & \text{if Adj Close}_t \leq \text{Adj Close}_{t-1} \end{cases}$$

This binary target variable transforms the problem into a classification task, allowing me to apply a variety of machine learning algorithms optimized for classification problems. The distribution of the `VIX_direction` variable shows that there are 42 instances of a decrease (labeled as 0) and 25 instances of an increase (labeled as 1). This indicates that downward movements in the VIX were more common than upward movements during the study period, providing a slight class imbalance in the target variable.

Table 5.1: Distribution of VIX Direction Target Variable

VIX Direction	**Count**
0 (Decrease)	42
1 (Increase)	25

5.3.2 Rationale for Predicting Direction

The decision to predict the direction of the VIX, rather than its exact value, stems from the nature of volatility in financial markets and the needs of market participants. In practice, predicting whether volatility will increase or decrease can be more actionable for traders, portfolio managers, and risk management professionals. For example, a hedge fund may implement strategies based on whether volatility is expected to rise (buying volatility) or fall (selling volatility), rather than attempting to predict the exact numerical value of the VIX.

Moreover, focusing on directional movement simplifies the classification task. Since my objective is to predict whether the VIX will go up or down, splitting the dataset into training and evaluation sets, rather than employing a forecast window approach, allows for a more straightforward model-building process. This approach maximizes the amount of data available for training, which is particularly important given that the period over which I have data is relatively short. By maintaining overall data consistency, I can better learn the underlying patterns without overcomplicating the model with temporal dependencies that may not be fully captured due to the limited data span.

Additionally, using a simple split between training and evaluation sets enhances computational efficiency. Forecast window methods require retraining the model at each step, which can be computationally intensive. With a limited dataset and the need for efficiency, especially when handling time sensitive financial data, the chosen approach provides a faster and more practical evaluation without significantly sacrificing the ability to capture important patterns.

Several studies in the literature have demonstrated the effectiveness of volatility direction prediction. For instance, Bollerslev [1986] found that predicting the direction of volatility is more robust to model inaccuracies than predicting exact levels, as the latter requires higher precision in capturing the non-linear dynamics of volatility movements. Furthermore, Engle and Patton [2001] highlight that volatility forecasting models, particularly those used in finance, are often more effective when simplified to directional movements, as they provide more practical utility for decision-makers.

5.4 Models and Results

In this section, I first examine the results of implementing various machine learning models without additional volatility measures. Then I present a significant improvement achieved by incorporating the GARCH volatility feature. These two phases of model development demonstrate the effectiveness of hybrid approaches in financial forecasting.

5.4.1 Machine learning models for VIX prediction

The first phase of the analysis involved applying standard machine learning algorithms to predict the direction of the VIX using only the sentiment data (positive, neutral, and negative percentages), S&P 500 adjusted close prices and lagged variables. These models were trained and tested using the following features:

- **Sentiment features:** Positive, neutral, and negative tweet percentages
- **Market data:** S&P 500 adjusted close prices
- **Lagged variables:** 1-day lagged versions of the sentiment and market variables

The dataset was split into training and test sets (80% training, 20% test), and the features were standardized using StandardScaler to ensure uniform scaling. I evaluated six machine learning models: Random Forest, Gradient Boosting, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression, and XGBoost.

The performance of each model was evaluated based on accuracy, with results summarized below:

Model	**Accuracy (%)**
Random Forest	64.29
Gradient Boosting	64.29
SVM	64.29
KNN	64.29
Logistic Regression	**85.71**
XGBoost	78.57

From the results, Logistic Regression performed the best with an accuracy of 85.71%, followed by XGBoost with 78.57%. Other models such as Random Forest, Gradient Boosting, SVM, and KNN exhibited similar accuracy, all around 64.29%. This suggests that while basic classification methods can capture some patterns in the data, the logistic regression model, in particular, excelled due to the linear separability of the data in this phase.

5.4.2 Integrating GARCH Volatility Features: A Hybrid Approach

After evaluating the initial model's performance, I introduced a more advanced measure of volatility to enhance the predictive capabilities of the model. Specifically, I integrated the GARCH(1,1) model to capture the conditional volatility in the VIX. The GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is particularly suitable for analyzing financial time series that exhibit volatility clustering, where periods of high volatility are often followed by high volatility and periods of low volatility are typically followed by low volatility.

Before fitting the GARCH model, it was necessary to analyze the stationarity of the VIX series, as GARCH models are generally more effective with stationary time series. To assess this, I conducted

Test Statistic	p-value	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)
-2.7255	0.0697	-3.5320	-2.9060	-2.5904

Table 5.2: ADF Test on Original VIX Series

Augmented Dickey-Fuller (ADF) tests on the original and differenced VIX series. The results are summarized in Tables 5.2 and 5.3.

Table 5.2 shows that the ADF test statistic of -2.7255 has a p-value of 0.0697, which is above the common significance levels. Therefore, we fail to reject the null hypothesis of non-stationarity, suggesting that the VIX series is not stationary in its original form.

Test Statistic	p-value	Critical Value (1%)	Critical Value (5%)	Critical Value (10%)
-9.5315	2.8833e-16	-3.5336	-2.9064	-2.5907

Table 5.3: ADF Test on Differenced VIX Series

Table 5.3 shows that, after differencing, the VIX series becomes stationary. The ADF test statistic of -9.5315 has a very low p-value (2.8833e-16), well below the critical values at the 1%, 5%, and 10% levels, confirming that we can reject the null hypothesis of non-stationarity.

With the VIX series confirmed to be stationary after differencing, I proceeded to apply the GARCH(1,1) model to capture the conditional volatility. I decided to use the GARCH(1,1) model with the VIX instead of the S&P 500 for several reasons that align with the study's goal of understanding and predicting volatility. The VIX, known as the "fear index," is tailored to represent market expectations of short-term volatility, reflecting market sentiment and expected risk. It directly gauges implied volatility from S&P 500 options, making it more suitable for volatility modeling.

To determine the optimal order of the GARCH model, I conducted a series of diagnostics by testing different combinations of the autoregressive (p) and moving average (q) parameters. These combinations were evaluated based on the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), which are standard criteria for model selection in time-series analysis. The results are presented in Table 5.4.

p	q	AIC	BIC
1	1	337.513507	348.536970
2	1	337.513642	348.537105
3	1	339.416371	352.644526
4	2	339.513481	352.741637
5	2	341.416371	356.849219
7	3	341.513487	356.946336
8	3	343.416371	361.053912

Table 5.4: GARCH Model Order Diagnostics

From Table 5.4, it is evident that the GARCH(1,1) model offers the lowest AIC and BIC values among the tested configurations, indicating that this model best captures the volatility clustering in the VIX series. The simplicity and effectiveness of the GARCH(1,1) model make it particularly suitable, as it balances model complexity with explanatory power. Therefore, I selected the GARCH(1,1) specification to represent the volatility of the VIX in this analysis.

The model is mathematically described as follows:

$$r_t = \mu + \epsilon_t$$

where r_t is the return at time t, μ is the mean return, and ϵ_t is the error term.

The conditional variance equation is given by:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

where:

- σ_t^2 is the conditional variance at time t (volatility).
- ϵ_{t-1}^2 is the squared residual from the previous period (the ARCH term).
- σ_{t-1}^2 is the conditional variance from the previous period (the GARCH term).
- α_0, α_1, and β_1 are parameters to be estimated.

The GARCH model in this setup considers both the long-term (by using β_1) and short-term (by using α_1) volatility persistence. This is essential for modeling the VIX, which is impacted by recent and past market events.

The GARCH(1,1) model was applied to the closing prices of VIX adjusted to calculate the conditional volatility. This volatility was then included as an extra feature in the dataset. Following the creation of the GARCH volatility feature, I proceeded to retrain the machine-learning models with the new feature set, now including:

- Sentiment features (positive, neutral, and negative percentages),
- Market data (S&P 500 adjusted close prices and volume),
- Lagged variables (e.g., lagged values of the VIX and S&P 500),
- **GARCH volatility**: the newly added feature representing conditional volatility in the VIX.

The models were re-trained using the same procedure as before but with the additional GARCH feature. The results are summarized below:

Model	**Accuracy with GARCH**
Random Forest	64.29
Gradient Boosting	57.14
SVM	64.29
KNN	71.43
Logistic Regression	**92.86**
XGBoost	71.43

Table 5.5: Model Accuracy with GARCH (1,1)

The introduction of the GARCH volatility feature significantly improved the performance of several models:

- Logistic Regression improved from 85.71% to 92.86%, solidifying its position as the best-performing model.
- KNN and XGBoost both showed improvements, achieving accuracies of 71.43% with the GARCH feature, compared to their earlier performance of 64.29%.

The mix of sentiment analysis and GARCH volatility modeling improved performance in various models, showing the benefit of using advanced volatility measures in market direction predictions. GARCH enhances the understanding of market fluctuations, and when integrated with sentiment-based forecasts, enhances the model's accuracy.

Benefits of the GARCH-Sentiment Hybrid Model:

1. **Capturing Volatility Clustering**: GARCH(1,1) is specifically designed to capture periods of increased or decreased volatility, a pattern that is common in financial markets.
2. **Enhanced Predictive Power**: Incorporating GARCH-based volatility features alongside sentiment data provides a more holistic view of the market. This hybrid approach allows the model to account for both *short-term sentiment* (via Twitter data) and *long-term volatility trends* (via GARCH).
3. **Robustness Across Models**: The improved performance across multiple machine learning models suggests that the GARCH volatility feature is generalizable and can enhance the predictive power of various algorithms.

5.5 Innovations and Contributions

This thesis presents several important improvements that extend the current understanding of predicting financial market behavior and analyzing sentiment. By combining machine learning techniques with traditional economic models, I address gaps in current research and suggest a new method for predicting market volatility, focusing on the VIX. This section explains the unique contributions of my research, showing how each part builds on existing knowledge.

5.5.1 Integration of Sentiment Analysis with Volatility Prediction

This thesis introduces a key innovation by combining sentiment analysis with volatility prediction models. While sentiment analysis is commonly used to forecast stock prices, few studies have delved into its relationship with market volatility, especially concerning the VIX, also known as the "fear index."

Utilizing Twitter data to gauge public sentiment and merging it with market data, my research presents a new method for anticipating market volatility. Unlike prior research like Bollen et al. [2011], which focused on using sentiment to predict stock market movements, my approach goes further by predicting the VIX's direction, a forward-looking indicator of expected volatility. This represents a notable advancement in the field, underscoring the significance of public sentiment in influencing market participants' views on future volatility.

5.5.2 Hybrid Approach: Combining GARCH Volatility with Machine Learning Models

One important contribution of this thesis is combining traditional econometric models, particularly GARCH, with machine learning techniques to improve prediction accuracy. While research has extensively explored the use of GARCH models for predicting volatility, limited studies have joined these models with machine learning methods, especially in sentiment-driven volatility prediction.

Using the GARCH(1,1) model to account for volatility clustering and conditional heteroscedasticity, I computed the conditional volatility of VIX. By incorporating this volatility metric as an additional feature in machine learning models, I showed that this combined approach significantly improved the

predictive capabilities of various classifiers. This hybrid method harnesses the strengths of econometric modeling in capturing volatility trends and the flexibility of machine learning models in handling complex relationships, offering a robust solution for forecasting market volatility.

Improvements with GARCH

One key outcome of this study is the boost in model accuracy seen with the addition of the GARCH volatility feature. By incorporating the VIX's conditional volatility as a factor, the Logistic Regression model achieved an impressive accuracy of 92.86%, a notable enhancement from its original 85.71% accuracy. This discovery shows how combining volatility analysis with sentiment data can be very effective, supporting the use of hybrid models for financial forecasting.

The contribution here is twofold:

- **Introducing volatility measures to sentiment-based models**: Adding quantitative volatility measures to sentiment models bridges the gap between financial econometrics and machine learning, which usually focus on qualitative features only.
- **Highlighting the power of hybrid models**: This work provides empirical evidence that hybrid models outperform standalone machine learning or econometric models, particularly in capturing the intricate relationship between sentiment and volatility.

Another unique aspect of this thesis is its focus on the COVID-19 pandemic period, a time characterized by extreme market volatility and uncertainty. Many previous studies on sentiment analysis focus on normal market conditions, but few address the role of sentiment during periods of crisis. By analyzing sentiment data during the pandemic, I contribute new insights into how public sentiment drives volatility in times of uncertainty.

Chapter 6

Conclusion

6.1 Summary of Results

In this research, I explored how sentiment analysis relates to market volatility, focusing on predicting the VIX's direction, a key index indicating market expectations of future volatility. The study involved analyzing a dataset of tweets about the S&P 500 and its top 25 companies during the unsettled period from April 9 to July 16, 2020, marked by increased uncertainty during the COVID-19 pandemic. The aim was to investigate if social media sentiment could be a predictive indicator for market volatility.

I used different sentiment analysis methods like the VADER lexicon-based model and machine learning models like Random Forest, XGBoost, and a Voting Classifier that merged these techniques. Furthermore, I created a hybrid model that combined GARCH-based features with sentiment analysis to improve predicting the VIX direction.

Integrating GARCH volatility features with sentiment analysis boosted prediction accuracy, especially in identifying the VIX trend. By prioritizing the trend over the exact VIX value, this study proposed a new method capturing the link between market sentiment and volatility more effectively. The combined model surpassed conventional models, underscoring the value of merging sentiment data with established financial indicators for a comprehensive market overview.

6.2 Key Conclusions

The research findings underscore the importance of sentiment analysis in forecasting market volatility, especially during uncertain economic periods. The research study supported that social media sentiment, analyzed using tools such as VADER, can impact investor decisions and, in turn, market volatility. Notably, predicting the VIX movement through sentiment analysis marks a significant advancement in the field, given previous studies mainly concentrated on stock prices rather than volatility indexes like the VIX.

The integration of GARCH characteristics with sentiment data in a hybrid approach demonstrated significant effectiveness. This indicates that incorporating sentiment analysis can greatly enhance traditional volatility models, especially in forecasting market conditions during periods of heightened uncertainty. Moreover, leveraging machine learning models like Random Forest and XGBoost alongside GARCH-based features led to a further enhancement in prediction accuracy.

6.3 Practical Implications

The outcomes of this research bear practical significance for investors and financial analysts. Integrating sentiment analysis into current predictive models for market volatility can yield a more comprehensive grasp of market dynamics. Specifically, by monitoring neutral sentiment in conjunction with positive and negative sentiment, one can discern early signals of market shifts, especially in times of heightened volatility, such as those observed amid the COVID-19 crisis.

For professionals, employing hybrid models that blend sentiment analysis with GARCH-based features offers a pragmatic tool for forecasting market volatility trends. This method proves especially beneficial for enhancing risk management and investment strategies, enabling analysts to better foresee market fluctuations and adapt their approaches accordingly.

6.4 Recommendations for Future Research

Although this research has made notable progress in comprehending the link between sentiment analysis and market volatility, there are various aspects where additional research could build upon these findings.

Advanced Sentiment Models: Future research could explore more advanced sentiment analysis models, such as transformer-based models (e.g., BERT or GPT), which have the potential to capture more complex emotional tones, such as sarcasm or irony. These models could further improve the accuracy of sentiment-based market predictions, especially when applied to more nuanced or ambiguous language commonly found in financial discussions.

Generalizing the Hybrid Approach: This study mainly examined the SP 500 and the VIX. Subsequent studies could utilize the hybrid model approach with various financial indices, asset classes, or markets to assess how broadly the findings apply. Testing these techniques across different sectors or regions could confirm if the hybrid model's effectiveness in forecasting volatility remains consistent in other financial settings.

Incorporating Real-Time Sentiment Data: Future research could focus on creating real-time sentiment analysis models. By integrating live data feeds from social media platforms or financial news sources, these models could become more useful in rapidly changing financial markets. Tracking sentiments in real-time would help analysts grasp market feelings as they change, allowing for more timely market volatility predictions.

Exploring Alternative Data Sources: Future studies could investigate incorporating various sentiment sources beyond Twitter data, like news articles, blogs, and financial reports. By merging different sentiment sources, researchers can develop stronger models that encompass a wider array of investor sentiments and viewpoints, resulting in more precise forecasts.

6.5 Limitations of the Study

While the findings of this study are promising, there are several limitations that should be acknowledged. First, the dataset used in this study was limited to tweets related to the S&P 500 and its top 25 companies. This narrow focus may limit the generalizability of the findings to other financial markets or asset classes. Additionally, the reliance on a relatively small set of manually labeled tweets for training the sentiment models may have constrained the performance of the machine learning algorithms.

Another limitation is the simplicity of the sentiment analysis model used. While VADER proved effective

for this study, more advanced sentiment analysis models, such as Transformer, could potentially yield more nuanced and accurate sentiment predictions. Future studies could explore the use of such models to further enhance the accuracy of market volatility predictions.

6.6 Potential effect of COVID-19 on this analysis

The time period of this analysis coincides with the COVID-19 pandemic, a period marked by global uncertainty and widespread disruptions. Non-essential activities came to a halt, and financial markets experienced significant volatility as market participants reacted to rapid economic changes, government interventions, and fluctuating investor sentiment.

Despite these extraordinary circumstances, I believe the validity of this analysis remains intact for the following reasons:

- **Robust Data Processing:** To ensure the reliability of the sentiment data, I calculated the percentage of tweets in each category (positive, negative, and neutral) to prevent days with higher social media activity from disproportionately influencing the model. Additionally, I standardized all other variables in the dataset to maintain consistency and comparability. I also differentiated the VIX time series, applied GARCH models, and conducted all necessary tests to confirm that the differentiated series was stationary.
- **Classification problem:** Instead of attempting to predict the exact value of the VIXwhich would require a comprehensive analysis of all external factors influencing it,I turn it into a classification problem. This approach focuses on predicting the direction of change, which is driven by broader, more stable factors that remain relatively constant over time.
- **Model Robustness:** By integrating GARCH features with sentiment analysis, this study reduces the reliance on any single factor, ensuring the models remain robust to external shocks such as the pandemic.

COVID-19 undoubtedly introduced unique conditions in the financial markets and strongly influenced volatility. However, I am confident that the steps taken to address these challengesthrough rigorous data processing, a focus on classification, and a robust model designensure the validity of my analysis. I believe that similar results would be obtained if the same analysis were conducted during a different period, further highlighting the adaptability and reliability of this approach.

6.7 Conclusion

In summary, this thesis has shown the significance of using sentiment analysis to forecast market volatility. By combining GARCH-based features with sentiment data in an innovative way, this study underscores the importance of including neutral sentiment when making volatility predictions. Additionally, it suggests that hybrid models have the potential to enhance prediction accuracy. These findings contribute to the existing research on sentiment analysis and financial forecasting, providing fresh perspectives on how public sentiment impacts market behavior. Further research could expand on these results by delving into more sophisticated models and applying them across a wider array of financial markets.

Appendix A

A.1 Detailed Results of Deep Learning Models

A.1.1 LSTM Model

Table A.1: LSTM Training and Validation Results

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.3508	1.0955	0.4462	1.0766
2	0.4201	1.0589	0.4346	1.0610
3	0.4927	0.9900	0.5231	1.0331
4	0.6754	0.8049	0.5000	1.0228
5	0.8113	0.5648	0.5269	1.0154

A.1.2 Bidirectional LSTM (BiLSTM) Model

Table A.2: BiLSTM Training and Validation Results

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.4106	1.0874	0.4192	1.0756
2	0.4512	1.0587	0.4385	1.0442
3	0.5160	0.9494	0.4769	1.0166
4	0.7523	0.6401	0.5308	0.9699
5	0.8884	0.4248	0.5231	1.0675

A.1.3 CNN Model

Table A.3: CNN Training and Validation Results

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.3690	1.0951	0.4308	1.0721
2	0.5612	1.0276	0.4692	1.0597
3	0.6421	0.9532	0.4808	1.0412
4	0.6874	0.8367	0.4692	1.0204
5	0.7066	0.6875	0.4885	0.9989

A.1.4 GRU Model

Table A.4: GRU Training and Validation Results

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	0.3876	1.0919	0.4192	1.0793
2	0.4313	1.0589	0.3615	1.0760
3	0.5791	11.7827	0.3846	1.0770
4	0.6742	0.9450	0.4115	1.0662
5	0.7374	0.8291	0.4038	1.0614

A.2 Summary of Model Results

Table A.5: Summary of Best Model Results

Model	Best Training Accuracy	Best Validation Accuracy	Training Loss	Validation Loss
LSTM	92.12%	52.69%	0.3402	1.0154
BiLSTM	88.84%	53.08%	0.4248	1.0675
CNN	70.66%	48.85%	0.6875	0.9989
GRU	73.74%	41.92%	0.8291	1.0614

Bibliography

S. Aras. Stacking hybrid garch models for forecasting bitcoin volatility. *Expert Systems with Applications*, 174:114747, 2021. doi: 10.1016/j.eswa.2021.114747.

G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003.

J. Bollen, H. Mao, and X.-J. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3): 307–327, 1986.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 1976.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

M. Chiny, M. Chihab, O. Bencharef, and Y. Chihab. Lstm, vader and tf-idf based hybrid sentiment analysis model. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12(7):265–275, 2021. URL `https://thesai.org/Downloads/Volume12No7/Paper_30-LSTM_VADER_and_TF_IDF_based_Hybrid_Sentiment.pdf`.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

A. D'Andrea, F. Ferri, P. Grifoni, and T. Guzzo. Approaches, tools and applications for sentiment analysis implementation. *International Journal of Computer Applications*, 125(3):26–33, 2015.

R. F. Engle and A. J. Patton. What good is a volatility model? *Quantitative Finance*, 1(2):237–245, 2001.

A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 417–422, 2006.

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*. Wiley, 3rd edition, 2013.

C. Hutto and E. Gilbert. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, 2014.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, 1998.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

X. Li, H. Xie, L. Chen, J. Wang, and X. Deng. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14–23, 2014.

T. Loughran and B. McDonald. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65, 2011.

H. Mao, S. Counts, and J. Bollen. Predicting financial markets: Comparing survey, news, twitter and search engine data. *arXiv preprint arXiv:1112.1051*, 2011.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119, 2013b. URL `https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

F. A. Nielsen. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on Making Sense of Microposts: Big Things Come in Small Packages (MSM2011)*, 2011.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing (EMNLP)*, pages 79–86. Association for Computational Linguistics, 2002.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

I. Rish. An empirical study of the naive bayes classifier. *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

L. A. Smales. News sentiment and the investor fear gauge. *Finance Research Letters*, 17:167–182, 2016. doi: 10.1016/j.frl.2016.03.011.

P. C. Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168, 2007.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

Y. Zhang, X. Guo, and Y. Yin. The effective class-imbalance learning based on smote and convolutional neural networks. *Mathematical Problems in Engineering*, 2018:1–10, 2018.