Analyzing the Correlation Between Retail Traders' Sentiments and Equity Market Movements

Haozhe Zeng | Cornell University | hz657@cornell.edu

Zixiao Wang | Cornell University | zw699@cornell.edu

Abstract

This research seeks to explore the impact of retail traders' sentiments, primarily from forums like Twitter and Reddit, on equity market movements. The investigation will discern the duration of this correlation, whether it's short-term or extends to mid-long term. It will also ascertain if the correlation is more pronounced in specific stock categories like penny stocks or tech giants or if such a correlation might be absent altogether. Upson determining the relationship, the project aim to develop a machine learning model that can detect potential trading signal.

Introduction

The financial arena has experienced a profound metamorphosis in the past few years, predominantly propelled by the digitization of trading platforms. Such innovations have democratized financial market access, resulting in an influx of retail traders actively engaging in stock trading. Characterized by their agility and swift mobilization capacity, these traders have ascended as a potent force in the equity market, contesting the dominance of traditional institutional entities.

One of the most prominent platforms that has come to symbolize this new wave of retail trading is the WallStreetBets forum on Reddit. Serving as a discussion hub, WallStreetBets has become a focal point for retail traders to share insights, strategies, and sentiments about various stocks. The power of such collective sentiment became glaringly evident during events like the GameStop short squeeze, where concerted buying actions driven by discussions and emotion on the forum led to unprecedented stock price surges, catching many institutional investors off guard.

However, while events like the GameStop incident have made headlines, a holistic analysis scrutinizing such forums' overarching influence on the equity market is yet to be undertaken.

This initiative intends to meticulously examine the sentiments reverberating within these platforms and ascertain their potential linkage with equity market oscillations. The endeavor is not just about identifying superficial correlations; it seeks to fathom the extent and intensity of such influences. Inevitably, questions emerge: Is there consistency in these correlations across varied stock sectors? Is the impact of these sentiments more pronounced for specific stock types, be it penny stocks or industry giants? And crucially, can the sentiments serve as a predictive tool for forecasting market trajectories?

This study's ambition is to offer an exhaustive insight into the dynamic interplay between retail trader sentiments and the intricacies of equity market behavior in this digital era. The ultimate aspiration is to harness the insights garnered from sentiment analysis to gauge midterm market fluctuations. Adding an intriguing dimension, the project also explores the potential of crafting a machine learning model aimed at identifying trading signals.

Why this approach?

Previous research has studied the relationship between retail sentiment and equity market movements, with many findings showing a positive correlation. There's also been interest in using machine learning to predict stock prices. However, the integration of sentiment analysis with machine learning to predict stock movements is less common in the literature.

Most prevailing research models tend to fall into one of two categories: they either employ machine learning for long-term stock predictions, sidelining sentiment analysis, or they entirely overlook the sentiment component. This presents a significant oversight. The stock market is inherently dynamic, continuously shaped by a myriad of factors. To solely rely on a monolithic prediction model, as many current studies opt to, poses limitations.

There emerges an undeniable imperative for more adaptive techniques, such as the rolling window method. By ensuring periodic model training and consistent recalibrations in line with fresh data, this approach promises a model that evolves in tandem with market changes, ensuring a more robust prediction mechanism.

Problem Statement:

Given a stock market with a set of stocks $S = \{s_1, s_2, \ldots, s_n\}$, where each stock s_i has a daily closing price $p_i(t)$ at time t, and a corresponding set of daily retail sentiment scores $R = \{r_1(t), r_2(t) \ldots, r_n(t)\}$, the problem is to construct a predictive model M that forecasts the future price $\hat{p}_i(t + \Delta t)$ of stock s_i at a future time $t + \Delta t$.

Sentiment Analysis

Challenges:

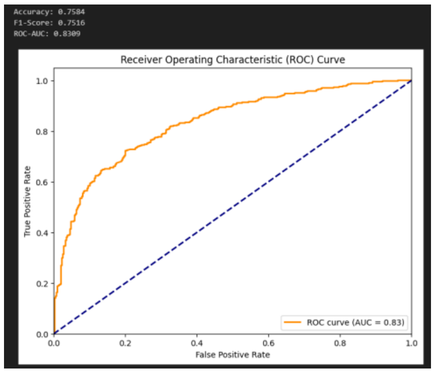
Data

Data Acquisition: Gathering relevant and high-quality data presented significant obstacles. We initially aimed to source real-time data from platforms like Reddit and Twitter. However, our efforts were hampered by API rate limits, reducing our collection efficiency. The premium versions of these APIs come at a steep price, while the free versions have many limitations. We had to explore alternative methods to gather sufficient stock-related posts; otherwise, the scarcity of data could severely hinder our model's performance.

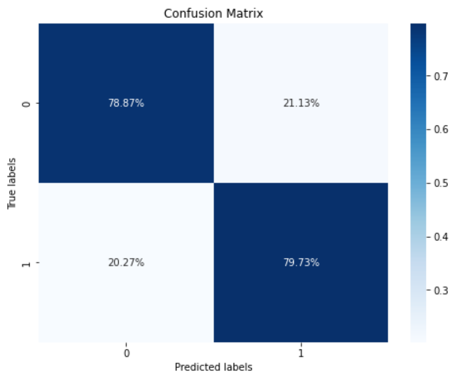
NLP Model

Noise Filtering: We sourced a dataset containing over 1.6 million Twitter posts. However, this dataset wasn't exclusively about the equity market; instead, it was a broader collection of general tweets. Other datasets we identified that were specific to the equity market were either unlabeled or contained a limited number of posts, typically around 8,000 entries. Currently, we are using the 1.6 million post dataset as our training set and the smaller, equity-specific datasets as validation sets. Given the non-specific nature of the larger dataset (with many irrelevant posts), our model's training set accuracy stands at 80%. In contrast, its accuracy on the test set drops to approximately 60%. This is visualized in the confusion matrices shown below:

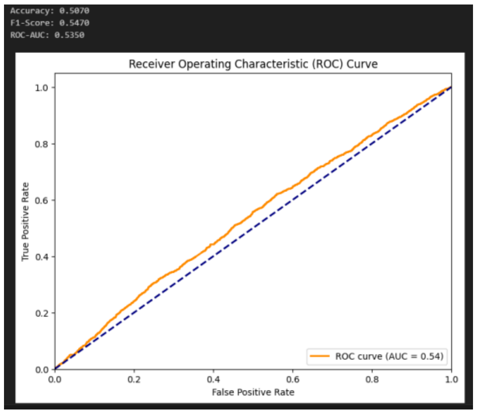
In the context of utilizing FastText embeddings with a training process involving three epochs on the initial dataset, the ROC curve exhibits a certain behavior. Specifically, during training on the original data, the model yields an ROC-AUC score of 0.83, which can be considered relatively satisfactory in terms of its performance. This suggests that the model effectively distinguishes between positive and negative cases within the training data. FastText, known for its subword information, is proving to be a valuable asset here. It excels in capturing not just whole words but also subword information, which is especially beneficial for languages with complex morphology and out-of-vocabulary words. This property often enables it to provide richer and more context-aware representations compared to Word2Vec, which we previously experimented with.



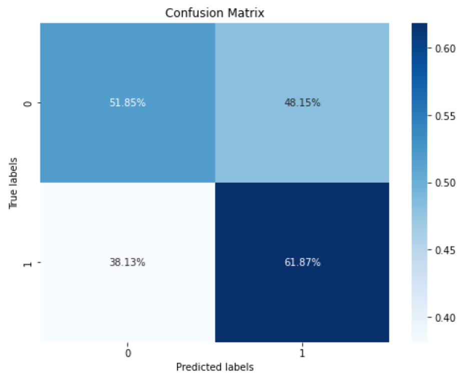
```
Accuracy: 0.7584
F1-Score: 0.7516
ROC-AUC: 0.8309
```



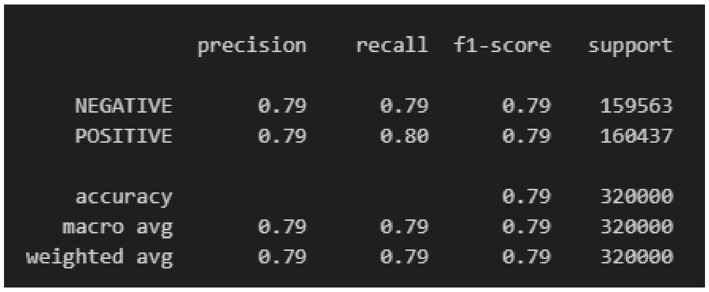
However, when the same model is subjected to testing using an entirely unseen dataset, a disparity emerges. The model performs suboptimally on this new data, as indicated by an ROC-AUC score of only 0.535, which is significantly lower than the training performance. This discrepancy, where the model's performance regresses when applied to unseen data and its ROC-AUC score falls closer to the baseline value of 0.5, is far from ideal. It suggests that the model might not generalize well to new, unseen instances and may need further refinement or adjustments to enhance its predictive capabilities on diverse datasets.



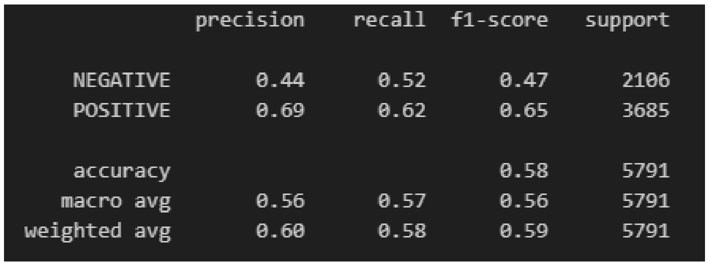
```
Accuracy: 0.5070
F1-Score: 0.5470
ROC-AUC: 0.5350
```



The initial training and testing accuracy using the RNN LSTM neural network displayed a noticeable disparity, with a commendable 0.79 on the validation set but a less satisfactory 0.58 on the testing set. This discrepancy raised concerns, as the model appeared to perform exceptionally well within the known confines of the training data but struggled when presented with new, unseen data. This disparity prompted the comprehensive evaluation of the data and model, necessitating an exploration into potential improvements in the training dataset composition and, perhaps, model architecture, to achieve a more balanced and consistent performance.



```
              precision    recall  f1-score   support

    NEGATIVE       0.79      0.79      0.79    159563
    POSITIVE       0.79      0.80      0.79    160437

    accuracy                           0.79    320000
   macro avg       0.79      0.79      0.79    320000
weighted avg       0.79      0.79      0.79    320000
```

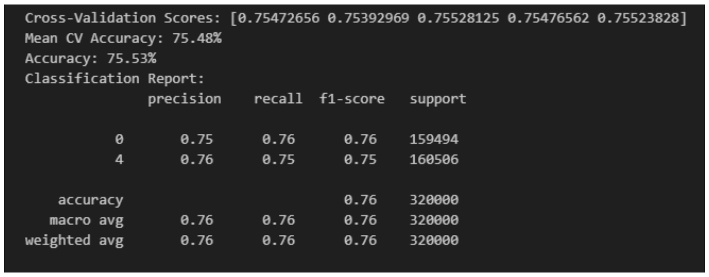


```
              precision    recall  f1-score   support

    NEGATIVE       0.44      0.52      0.47      2106
    POSITIVE       0.69      0.62      0.65      3685

    accuracy                           0.58      5791
   macro avg       0.56      0.57      0.56      5791
weighted avg       0.60      0.58      0.59      5791
```

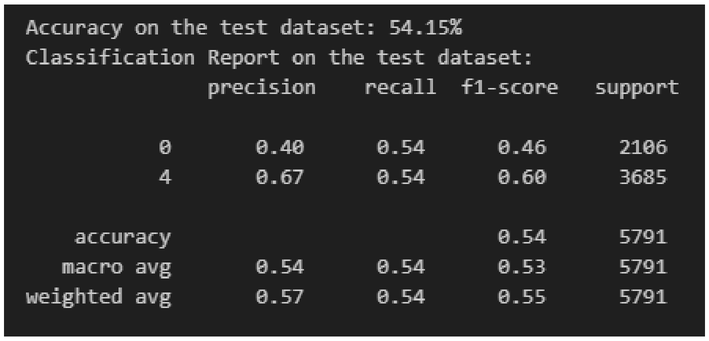
When confronted with the drop in testing accuracy, I initiated a comparative study involving different machine learning models. The objective was to discern if the discrepancy in performance was a result of the training data's quality or if it stemmed from the chosen model's limitations.

In pursuit of a thorough comparative analysis, I carefully chose a diverse array of machine learning models, each renowned for its specific strengths in handling sentiment analysis tasks and its adaptability across various data types. The ensemble of models included the Naive Bayes classifier, well-regarded for its simplicity and robustness, the Random Forest, which excels in capturing complex relationships within data, and XGBoost, a highly versatile model known for its efficiency and performance across a broad spectrum of tasks. This selection was not arbitrary; it aimed to illuminate whether the observed dip in testing accuracy could be attributed to the intricacies of the model itself or if it was intricately tied to the composition of the training data.

The experiment employing the Naive Bayes classifier yielded a training validation accuracy of 0.76, accompanied by a testing accuracy of 0.54. Surprisingly, these results did not exhibit a substantial divergence from the initial neural network approach. This suggests that the issue may not solely lie within the choice of machine learning model but may also be influenced by inherent challenges posed by the dataset itself. Consequently, these findings highlight the importance of addressing data quality and diversity to enhance overall model performance.

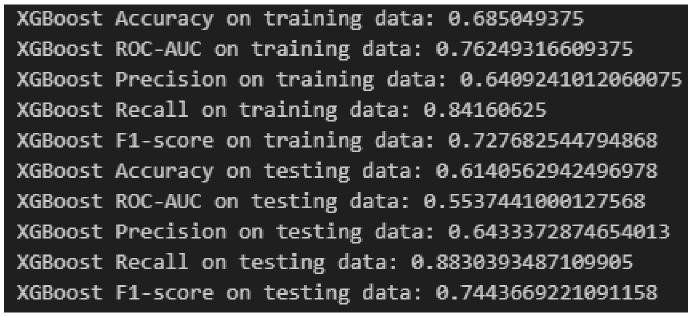


```
Cross-Validation Scores: [0.75472656 0.75392969 0.75528125 0.75476562 0.75523828]
Mean CV Accuracy: 75.48%
Accuracy: 75.53%
Classification Report:
               precision    recall  f1-score   support

           0       0.75      0.76      0.76    159494
           4       0.76      0.75      0.75    160506

    accuracy                           0.76    320000
   macro avg       0.76      0.76      0.76    320000
weighted avg       0.76      0.76      0.76    320000
```

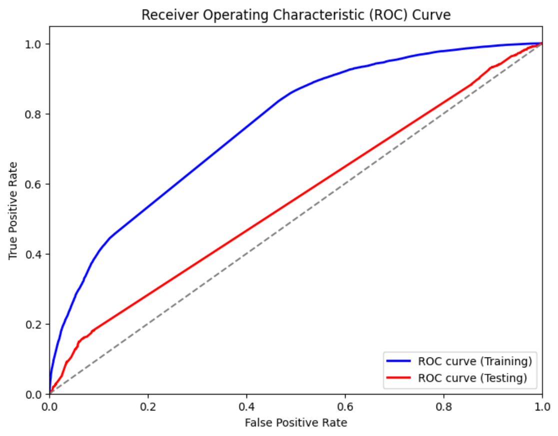


```
Accuracy on the test dataset: 54.15%
Classification Report on the test dataset:
              precision    recall  f1-score   support

           0       0.40      0.54      0.46      2106
           4       0.67      0.54      0.60      3685

    accuracy                           0.54      5791
   macro avg       0.54      0.54      0.53      5791
weighted avg       0.57      0.54      0.55      5791
```

Likewise, the experiment incorporating the XGBoost model resulted in a training validation accuracy of 0.68 and a testing accuracy of 0.61, mirroring the outcomes achieved with the initial neural network approach. These consistent results across different machine learning methods emphasize the persistent challenges presented by the dataset's composition. It underscores the necessity for further data preprocessing, feature engineering, or the exploration of alternative data sources to enhance the model's capability to discern equity market-related sentiments effectively.



```
XGBoost Accuracy on training data: 0.685049375
XGBoost ROC-AUC on training data: 0.76249316609375
XGBoost Precision on training data: 0.6409241012060075
XGBoost Recall on training data: 0.84160625
XGBoost F1-score on training data: 0.727682544794868
XGBoost Accuracy on testing data: 0.6140562942496978
XGBoost ROC-AUC on testing data: 0.5537441000127568
XGBoost Precision on testing data: 0.6433372874654013
XGBoost Recall on testing data: 0.8830393487109905
XGBoost F1-score on testing data: 0.7443669221091158
```



Overall, the intriguing discovery was that the testing accuracy, obtained from these alternative models, exhibited no substantial improvement over the initial results achieved with the more complex RNN LSTM neural network. This outcome suggested that the model's complexity wasn't the primary bottleneck in this scenario.

This observation led me to a pivotal realization: the primary challenge lay in the composition and quality of the training dataset. Our training dataset, sourced from a broader collection of Twitter posts, encompassed numerous unrelated and irrelevant posts that weren't directly related to the equity market. As a result, the model's performance was hampered by the noise present in this extensive dataset.

To address this, it became apparent that our focus should shift towards enhancing the training data. A two-fold strategy was identified. First, we needed to curate and filter the training dataset to include a more concentrated subset of posts that specifically related to the equity market. Secondly, we should explore incorporating additional labeled data from the smaller, equity-specific datasets to further fine-tune the model.

By refining and enriching our training data with these strategies, we can expect to mitigate the impact of noise and boost the model's accuracy on the test dataset. This comprehensive approach is instrumental in tackling the challenges posed by the non-specific nature of the larger dataset and ultimately achieving the desired accuracy and reliability in equity market sentiment prediction.

Current State:

Data

We conducted an extensive search across platforms like Kaggle and sought out open-source datasets employed in research publications. This concerted effort yielded a collection of textual data encompassing financial domain posts. However, our data curation extended beyond the strictly formal financial discourse, as we also incorporated Twitter posts reflecting the informal language used by individuals when discussing market matters. Furthermore, we incorporated a selection of news headlines featuring labeled sentiments, aiming to replicate the influence of up-to-date news citations within typical stock-related conversations.

The decision to explore and incorporate these alternative data sources stems from a recognition of their unique advantages. First and foremost, financial data extracted from Kaggle and other research papers offers a specialized and well-curated collection of content. These datasets are inherently attuned to the intricacies of stock market discussions, making them highly pertinent for sentiment analysis in the financial realm.

On the other hand, the inclusion of Twitter posts introduces an entirely different dimension to the dataset. Twitter has become a prominent platform for investors, traders, and financial experts to express their views on the market in real-time. By integrating these real-world, colloquial conversations, we aim to capture the informal, yet valuable insights shared by individuals participating in the market. This diversified data source enables us to confront the inherent noise and unpredictability of social media discussions, which is integral to sentiment analysis.

Additionally, the incorporation of news headlines with labeled sentiments presents a vital facet. Recent financial news plays a pivotal role in influencing market sentiments. The ability to integrate these data points into the training dataset allows our model to respond dynamically to real-time information. It mirrors the actual scenario where market participants react to breaking news, encapsulating the rapid ebb and flow of sentiment that characterizes the equity market.

Furthermore, recognizing the need for a well-rounded training dataset, we harnessed the capabilities of ChatGPT to bolster our content repository. Through the ChatGPT API and user interface, we generated a wealth of additional data. ChatGPT's natural language generation capabilities allowed us to produce an array of text, closely resembling the conversational style and diversity present in stock market discussions. These generated contents were then meticulously labeled with sentiments to ensure their compatibility with the sentiment analysis task.

To bolster the quality of our training data, a variety of data augmentation techniques were employed.

1. Synonym Replacement. The technique involves replacing words in a sentence with synonyms to introduce variety. For example, in one of the text comments, "Worried about the recent drop in the price of gold," synonym replacement might result in: "Concerned about the recent decline in the value of gold."

2. Back Translation. It involves translating text into another language and then back into the original language, which can introduce subtle phrasing changes. For instance, in one of the text comments, "AAPL's product launch was underwhelming, considering selling my shares," back translation might yield: "AAPL's product launch was disappointing; thinking about divesting my shares."

3. Paraphrasing. It offers alternative sentence structures and expressions. For example, "Just sold my Amazon shares; they've become too expensive," paraphrasing might produce: "I've recently disposed of my Amazon holdings as they've become unaffordable."

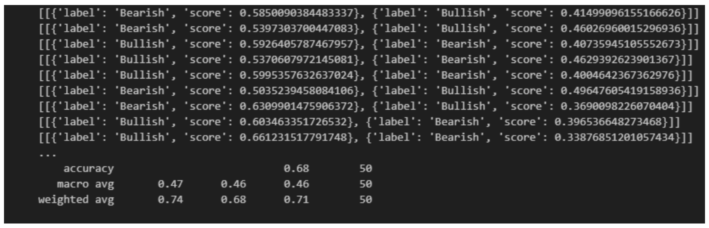
4. Oversampling/Undersampling. The techniques help address class imbalances, ensuring that sentiment categories are equally represented. If there's an imbalance between positive and negative sentiment comments, oversampling can duplicate examples from the minority class, while undersampling can reduce examples from the majority class to balance the dataset. Here I oversampled the negative data to have a balanced training set.

Overall, these methods are designed to introduce diversity and flexibility into our dataset, facilitating improved model generalization. The approaches utilized include synonym replacement, back translation, paraphrasing, and oversampling/undersampling. Synonym replacement broadens the dataset by substituting words with synonyms, allowing the model to encounter varied language expressions. Back translation generates paraphrased text by translating it into another language and back, thereby enhancing linguistic diversity. Paraphrasing offers alternative sentence structures and expressions, further enriching the dataset. Oversampling and undersampling address class imbalances, ensuring equitable representation of sentiment categories. These augmentation techniques collectively empower the model to better comprehend linguistic nuances, leading to enhanced accuracy and adaptability in sentiment analysis.

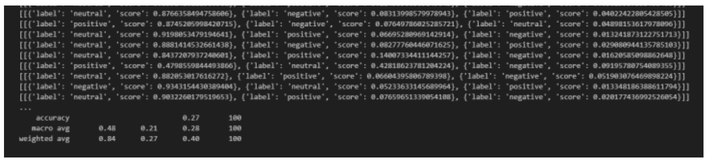
NLP Model

To enhance the capabilities of our initial model, we made the strategic decision to leverage the power of BERT, a state-of-the-art natural language processing model. In particular, we adopted a variant of BERT known as FinBERT, as detailed in the paper titled "FinBERT: A Large Language Model for Extracting Information from Financial Text" by Huang, Wang, and Yang (2022) [Huang, Wang, & Yang, 2022]. This model offers a plethora of advantages, such as its specialization in understanding financial text and sentiment. FinBERT is meticulously fine-tuned in the finance domain, utilizing a vast financial corpus for training. The utilization of the Financial PhraseBank dataset, as introduced by Malo et al. in 2014 [Malo et al., 2014], plays a crucial role in the fine-tuning process, enabling precise sentiment classification within financial contexts. For more comprehensive insights, please refer to the paper "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models" and our related Medium blog post.

To employ the model, I initiated the deployment process via the Hugging Face Query API, utilizing the repository "tarnformnet/Stock-Sentiment-Bert." The performance of this model exceeded our expectations, achieving an accuracy rate of 0.68 on the test dataset. Moreover, I explored an alternative variant known as the ProsusAI/finbert model, which provides softmax outputs for three sentiment labels: positive, negative, and neutral. However, given the binary nature of our testing dataset, I endeavored to further fine-tune the model to align it with the specific requirements of our dataset.

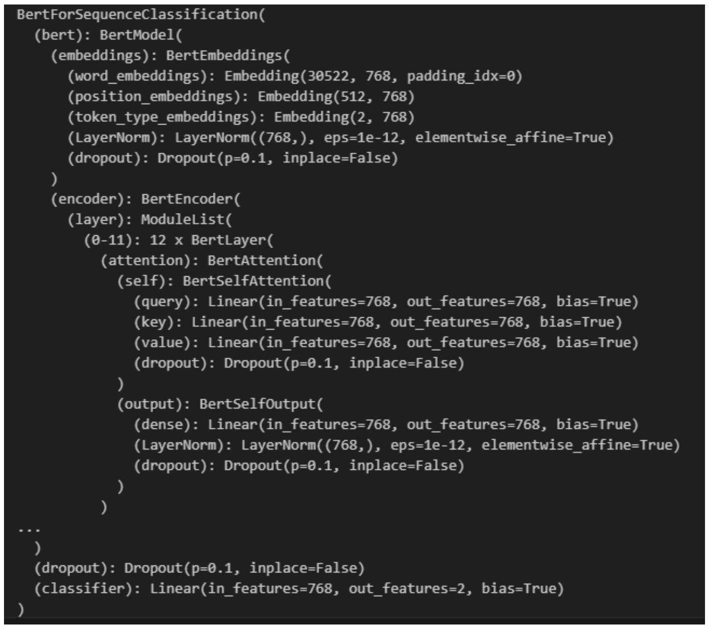


```
[[{'label': 'Bearish', 'score': 0.5850090384483337}, {'label': 'Bullish', 'score': 0.41499096155166626}]]
[[{'label': 'Bearish', 'score': 0.5397303700447083}, {'label': 'Bullish', 'score': 0.46026960015296936}]]
[[{'label': 'Bullish', 'score': 0.5926405787467957}, {'label': 'Bearish', 'score': 0.40735945105552673}]]
[[{'label': 'Bullish', 'score': 0.5370607972145081}, {'label': 'Bearish', 'score': 0.4629392623901367}]]
[[{'label': 'Bullish', 'score': 0.5995357632637024}, {'label': 'Bearish', 'score': 0.4004642367362976}]]
[[{'label': 'Bearish', 'score': 0.5035239458084106}, {'label': 'Bullish', 'score': 0.49647605419158936}]]
[[{'label': 'Bearish', 'score': 0.6309901475906372}, {'label': 'Bullish', 'score': 0.3690098226070404}]]
[[{'label': 'Bullish', 'score': 0.603463351726532}, {'label': 'Bearish', 'score': 0.396536648273468}]]
[[{'label': 'Bullish', 'score': 0.661231517791748}, {'label': 'Bearish', 'score': 0.33876851201057434}]]
...
    accuracy                           0.68        50
   macro avg       0.47      0.46      0.46        50
weighted avg       0.74      0.68      0.71        50
```



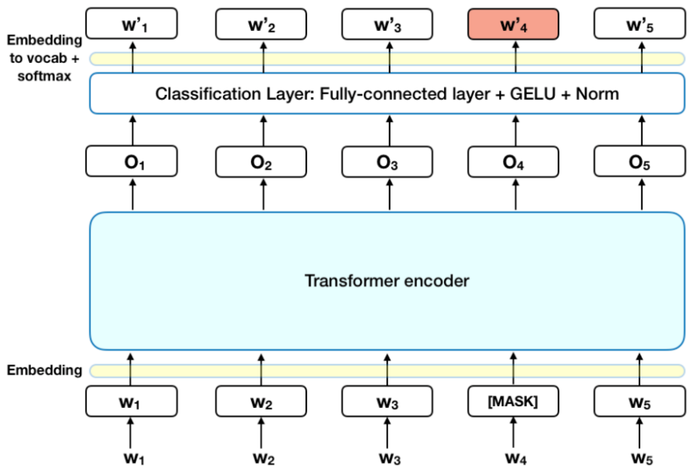
```
[[{'label': 'neutral', 'score': 0.8766358494758606}, {'label': 'negative', 'score': 0.08313998579978943}, {'label': 'positive', 'score': 0.04022422805428505}]]
[[{'label': 'positive', 'score': 0.8745205998420715}, {'label': 'negative', 'score': 0.07649786025285721}, {'label': 'neutral', 'score': 0.04898153617978096}]]
[[{'label': 'neutral', 'score': 0.9198053479194641}, {'label': 'positive', 'score': 0.06695280969142914}, {'label': 'negative', 'score': 0.013241873122751713}]]
[[{'label': 'neutral', 'score': 0.8881414532661438}, {'label': 'negative', 'score': 0.08277760446071625}, {'label': 'positive', 'score': 0.029080944135785103}]]
[[{'label': 'neutral', 'score': 0.8437207937240601}, {'label': 'positive', 'score': 0.14007334411144257}, {'label': 'negative', 'score': 0.01620585098862648}]]
[[{'label': 'positive', 'score': 0.4798559844493866}, {'label': 'neutral', 'score': 0.42818623781204224}, {'label': 'negative', 'score': 0.09195780754089355}]]
[[{'label': 'neutral', 'score': 0.882053017616272}, {'label': 'positive', 'score': 0.06604395806789398}, {'label': 'negative', 'score': 0.05190307646989824}]]
[[{'label': 'negative', 'score': 0.9343154430389404}, {'label': 'neutral', 'score': 0.05233633145689964}, {'label': 'positive', 'score': 0.013348186388611794}]]
[[{'label': 'neutral', 'score': 0.9032260179519653}, {'label': 'positive', 'score': 0.07659651339054108}, {'label': 'negative', 'score': 0.020177436992526054}]]
...
    accuracy                           0.27       100
   macro avg       0.48      0.21      0.28       100
weighted avg       0.84      0.27      0.40       100
```

In our pursuit of further refining the model, I embarked on a journey to fine-tune it using the 'yiyanghkust/finbert-tone' model, closely following the comprehensive guidelines they provided. Unfortunately, during this process, I encountered certain challenges stemming from compatibility issues with the environment and libraries, leading to an unsuccessful attempt.



```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
...
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

In response, I decided to explore an alternative approach by training the BERT model from scratch using the 'bert-base-uncased,' the original uncased base model, in combination with the newly acquired financial data. This method offered the advantage of full control and customization over the training process, enabling us to align the model precisely with our specific requirements.


Embedding to vocab + softmax
W'1
W'2
W'3
W'4
W'5
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

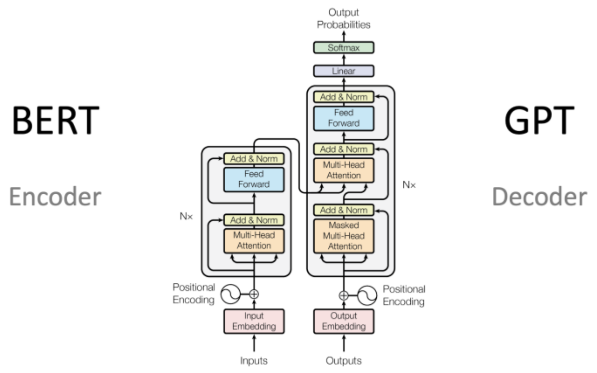
The provided model structure is a variant of the BERT model called BertForSequenceClassification. It is initialized with weights from the "bert-base-uncased" checkpoint and some weights have been newly initialized for the specific downstream classification task.

The model architecture consists of a pre-trained BERT model with additional layers for sequence classification. The BERT model itself has multiple components, such as embeddings, an encoder, and a pooler. The embeddings layer includes word embeddings, position embeddings, and token type embeddings, which handles token, position, and token type embeddings, allowing the model to understand the sequential and structural aspects of the input text. The encoder is composed of multiple layers (12 in this case), with each layer containing a self-attention mechanism, intermediate feed-forward layers, and output layers. These components help the model capture contextual information and relationships within the input sequences. The pooler is responsible for providing a fixed-size representation of the input sequence. It is typically used for classification tasks.

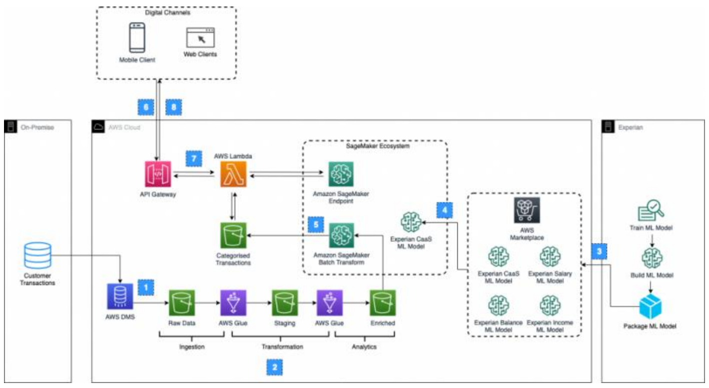
A dropout layer is used to prevent overfitting, which is a regularization technique. It is applied at various stages within the model. It helps prevent overfitting by randomly setting a fraction of input units to zero during training. The p parameter (0.1 in this case) specifies the probability that each element is dropped out.

The "classifier" is a linear layer that maps the BERT model's output to the specific classification task. It has 768 input features (matching the output size of the BERT model) and 2 output features (indicating a binary classification task). We can adjust the number of output features to match specific classification task. In this case, it has two output features, indicating a binary classification task.

Overall, BERT is considered a significant advancement in the field of Natural Language Processing and has several special characteristics that set it apart from previous NLP models. First of all, BERT is a bidirectional model, meaning it can consider both left and right context when encoding a word in a sentence. Traditional models like LSTMs and traditional Transformers were unidirectional, considering only the previous context [Devlin et al., 2018]. This bidirectionality allows BERT to capture more comprehensive context and dependencies in language. Secondly, BERT is pre-trained on a massive corpus of text, which enables it to learn rich and generalizable language representations. During pre-training, BERT learns to predict missing words (masked language model) and understand sentence relationships (next sentence prediction) [Devlin et al., 2018]. This pre-training helps BERT acquire world knowledge and linguistic understanding. Thirdly, BERT models are typically large-scale, with hundreds of millions to billions of parameters. The large architecture allows them to capture intricate language patterns and nuances. However, it also demands substantial computational resources for training and inference.



While it's worth noting that I have the option to fine-tune the model using Amazon SageMaker in conjunction with Hugging Face, which can potentially expedite the process and offer additional benefits, I have yet to explore this avenue. The decision not to do so at this stage is primarily motivated by a desire to manage time effectively and minimize any potential additional financial costs that might be associated with this approach.



In our pursuit of refining our model further, I ventured into the "yiyanghkust/finbert-tone" repository, a valuable resource that promised to enhance our model's sentiment analysis capabilities. With great enthusiasm, I followed their comprehensive fine-tuning guidelines to make the most of this tool. Unfortunately, my endeavors hit a roadblock due to compatibility issues with certain libraries in my environment, making it impossible to proceed with this method.

Undeterred by this setback, I decided to take a different approach. I embarked on the task of training a BERT model from scratch, using the original 'bert-base-uncased' model as my foundation. This choice was based on the model's established reputation and its adaptability to a wide range of tasks. To bolster its performance, I integrated the additional dataset mentioned earlier, ensuring it was well-equipped to tackle the intricacies of financial sentiment analysis.

Although Amazon SageMaker offered a promising platform for fine-tuning models in conjunction with Hugging Face, I opted not to explore this avenue fully at the moment. My decision was motivated by the desire to save time and avoid any potential additional financial costs associated with the process.

Next Step

In the next phase of our project, we will proceed with training the new BERT-based model using the augmented dataset we discussed earlier. Our primary objective is to achieve an accuracy rate of at least 80%, which has been our standard benchmark. If, during the training process, we find that the model's accuracy falls short of this threshold, we will resort to the troubleshooting techniques that have proven effective in the past. This entails investigating whether the issue lies within the dataset or if adjustments are needed in the model architecture. Our dedication to achieving the 80% accuracy mark is driven by the importance of a robust foundation for our stock price prediction model.

Once we attain the desired accuracy level, we will integrate the prediction model with its numeric predictions into our final stock price prediction model. This fusion is pivotal as it enables us to assign precise numeric values to labels, which are derived from the probabilistic predictions made by our model. These numeric values, which played a crucial role in the label selection process, will serve as a cornerstone in our stock price prediction model. This integration represents a significant step toward rigorously testing our initial hypothesis. It allows us to delve into the intricate relationship between market sentiment and stock prices with a high degree of precision and data-rich insights.

The decision to incorporate numeric values instead of categorical values into our final model is motivated by our pursuit of greater information richness and flexibility. Numeric values inherently offer a wealth of data that can be harnessed to gain deeper insights into the complex world of financial sentiment and stock market behavior. Unlike categorical values, which categorize data into discrete groups, numeric values are versatile and adaptable. They empower us to apply a range of statistical techniques, including calculating means, maximums, minimums, variances, and more. These analytical tools provide us with a wealth of statistical information, which offers a comprehensive understanding of the data's distribution and characteristics.

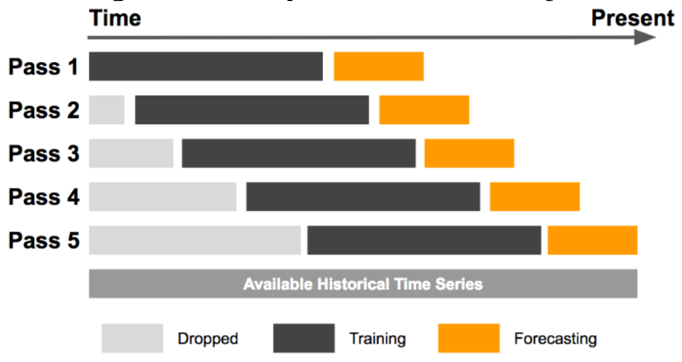
In essence, using numeric values equips us with the precision and adaptability needed to explore the complex interplay between market sentiment and stock prices comprehensively. It enables us to uncover subtleties that can have a profound impact on stock price movements. This approach is pivotal in our endeavor to gain a nuanced understanding of the dynamics of financial markets. By leveraging the power of numeric values, we aim to extract valuable insights that will inform our future strategies and decisions, ultimately enhancing our ability to navigate the intricacies of the financial landscape.

Stock

Challenges: Objective

Several studies propose using a singular model to forecast stock returns for an extended period, sometimes spanning up to a hundred days. We find this approach potentially limiting. Given the dynamic nature of the market, relying on one model to predict returns over multiple days seems unrealistic.

In contrast, we advocate for a model that is recalibrated daily, leveraging fresh data for each day's prediction. After forecasting the next day's or even the next week's return, the model can then assimilate the actual return data for that day. This iterative approach allows the model to continually refine its predictions based on the latest market conditions. Termed the "rolling window" method, this strategy emphasizes daily predictions while updating the dataset after each forecast. Such an approach is more attuned to the market's dynamic, enhancing the accuracy and relevance of predictions.



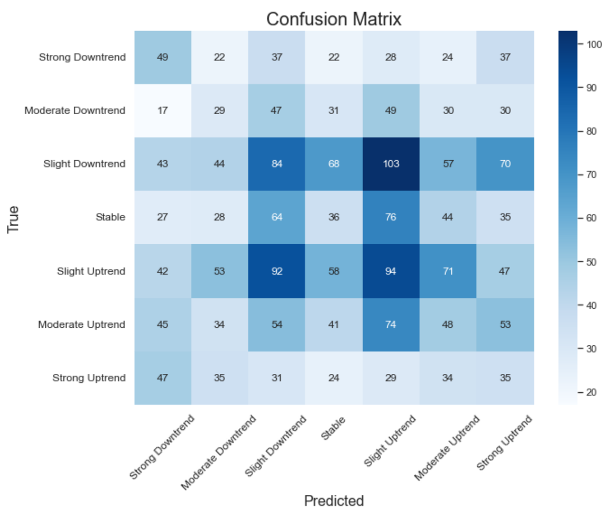
Two critical components define a rolling window model: the window size and the duration of the return you're predicting. While this model excels in capturing market dynamics, it can be computationally demanding due to its iterative training nature. Determining the optimal window size poses a challenge, as it can range from a short span of 5 days to several thousand days. Naturally, larger window sizes intensify the computational burden. When working with intricate deep learning models like Long Short-Term Memory (LSTM), it might be more reasonable to set a threshold for deciding when to update the model, rather than retraining it at every iteration. This can balance the need for updated information with the practicalities of computational efficiency.

Moreover, determining the precise aspect to predict brings its own set of challenges. The main objective of this research is to identify the correlation between retail sentiment and stock movement. Given this aim, it initially seemed fitting to treat it as a classification challenge, aiming to predict if the stock movement for the next day would be positive or negative.

To achieve this, we used an XGBoost classifier as the baseline method. The methodology applied to categorize the next day's return was as follows:

1. If the return value fluctuated between -0.002 and 0.002, it was categorized as 'Stable'.
2. A slight increase between 0.002 and 0.01 was labeled as 'Slight Uptrend', while a slight decrease between -0.01 and -0.002 was termed as 'Slight Downtrend'.
3. If the rise was between 0.01 and 0.02, it indicated a 'Moderate Uptrend', and a fall between -0.02 and -0.01 indicated a 'Moderate Downtrend'.
4. Any return value above 0.02 was classified as a 'Strong Uptrend', while any value below -0.02 was termed as a 'Strong Downtrend'.

The initial approach to understanding stock movements involved a detailed categorization ranging from stable periods to pronounced uptrends and downtrends. The confusion matrix presented showcased the performance of this classification method. The methodology took approximately 20 minutes for execution, which may be deemed lengthy for real-time analysis. When using only the stock price as a predictor, the accuracy was marginally better than a random guess. This suggests a need for a more comprehensive and efficient methodology.



Recognizing the limitations of the classification approach, a shift towards regression was considered. The rationale was that predicting a continuous outcome (the stock's future price or return) might be more effective. Once the future price is estimated, it can then be discretized into categories. The revised strategy employed an XGBoost Regressor, aiming to predict the next day's return. The predicted return was then converted into a categorical representation of stock movement. The XGBoost Regressor was notably more efficient, completing its run in about 4 minutes. This model achieved an accuracy of approximately 20%. While this is an improvement over the classifier approach, there remains room for enhancement.

Current State: Stock

In the face of consistent challenges, it became evident that our model's focus on predicting next-day returns might not be the optimal approach. A deeper dive into the methodology and its implications illuminated several key insights.

a) Temporal Dynamics of Sentiment: Leveraging sentiment analysis in our model highlighted that the effects of retail sentiment on stock prices aren't instantaneous. Rather, there's a lagged impact, reflecting a more gradual influence on stock movements.

b) Uncertainty of Short-Term Predictions: The attempt to predict single-day returns proved fraught with uncertainties. Factors like daily news events, global market movements, and institutional trading decisions can cause significant price fluctuations in the short term.

c) Broadening the Timeframe: Our objective isn't about chasing daily fluctuations but understanding broader market dynamics. Adopting a swing trading perspective, which focuses on capturing gains in a stock (or any financial instrument) over a period of several days to weeks, aligns more closely with our goals.

Considering these insights, it is clear that a shift in strategy towards predicting mid to long-term stock movements, taking into account the more subtle and prolonged impacts of retail sentiment, could provide a more accurate and actionable framework for our endeavors.

Consequently, we shifted the focus of my model to forecast the returns for the upcoming week. My primary interest transitioned from pinpointing stock movements to uncovering viable trading strategies, which I deem to be more pragmatic. As it stands, I employ a rolling window time series model. Each day, the model predicts the stock price for five days ahead and undergoes daily retraining to assimilate the latest information.

Time Series Model

The objective now is to predict the stock price of APPLE for a given time frame. Various features and methodologies were experimented with, to improve the model's performance.

a) Dataset: The dataset encompasses a comprehensive range of equities, including prominent stocks such as Apple, Amazon, QQQ, and SPY. It provides daily trading data, including the opening price, closing price, daily highs, daily lows, adjusted closing price, and trading volume. This dataset spans an extensive period from January 1, 2010, to January 1, 2023, offering a rich historical perspective for analysis.

b) Initial Metric: The model started with a Mean Absolute Percentage Error (MAPE) of approximately 4%.

a) Model Optimization:

    a) The model was adjusted to run on only 10% of the original time series data for feature selection, which optimized processing and ensured a more streamlined approach.

b) Feature Selection:

    a) Several features were experimented with, including volume data, open price and its lags, highs and lows of a day, and economic indicators.

    b) Inclusion of moving averages (Mas) and SPY brought a significant increase in the model's performance.

    c) Several other features were added and tested such as RSI, WVAD, MACD, CCI, BOLL, and others. However, not all added significant value to the model's predictive capability.

c) Model Performance:

   a) After multiple iterations, feature additions, and adjustments, the model achieved a MAPE of 1.87%. This is a notable improvement from the initial 4%.

d) Processing Time:

   a) The model takes approximately 2.46 minutes to run on the entire dataset, demonstrating efficiency in processing.

e) Window Size:

   a) A window size of 200 was used for the model, typically representing 1 year of stock history.

The stock price prediction model for APPLE has undergone extensive fine-tuning and experimentation. The emphasis on feature engineering and model adjustments has led to a significant improvement in prediction accuracy, as evidenced by the reduction in Mean absolute percentage error (MAPE) from 4% to 1.74% for the training set. The inclusion of moving averages (MAs) and SPY as features was especially beneficial, highlighting the importance of these variables in predicting APPLE's stock price.

The current model is a rich compilation of various columns, each presenting a unique facet of stock market information. The depth and variety of these columns allow for in-depth analysis and the crafting of sophisticated trading strategies. Here's a succinct breakdown of each column:

1. **Date**: Represents the specific day for the data point, giving chronological context to the observations.

Price & Volume Columns:

2. **Close**: The price at which the stock settled at the day's end.
3. **Close_lag_i**: A historic reference, this reflects the closing price from 'i' days ago, aiding in drawing comparisons over time. The current data set includes a 10-day lag.
4. **Volume**: Represents the sheer volume of shares that exchanged hands on that day, indicating the day's trading intensity.

Moving Averages: These offer a smoothed version of the price data, revealing underlying trends by averaging out short-term fluctuations:

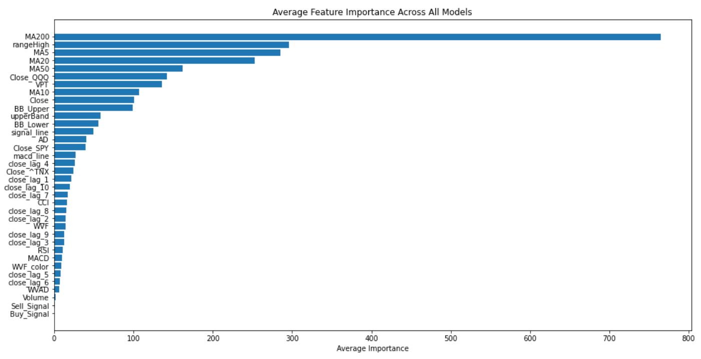
5. **MA5**: Reflects short-term trends using a 5-day period.
6. **MA10 & MA20**: Capture medium-term movements.
7. **MA50 & MA200**: Provide insights into longer-term trends and are particularly watched by traders.

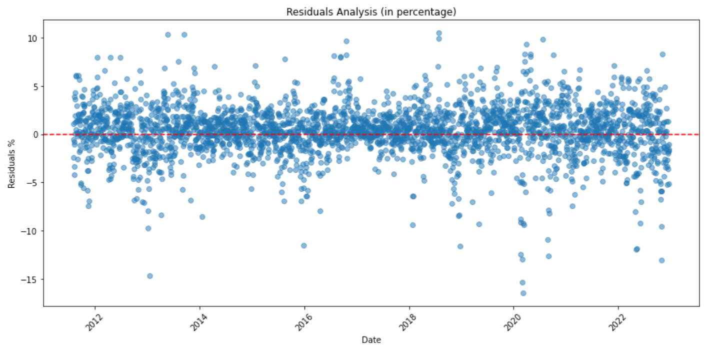
Indicators: These are a mix of momentum, volume, and volatility metrics that traders often utilize to decipher market sentiments:

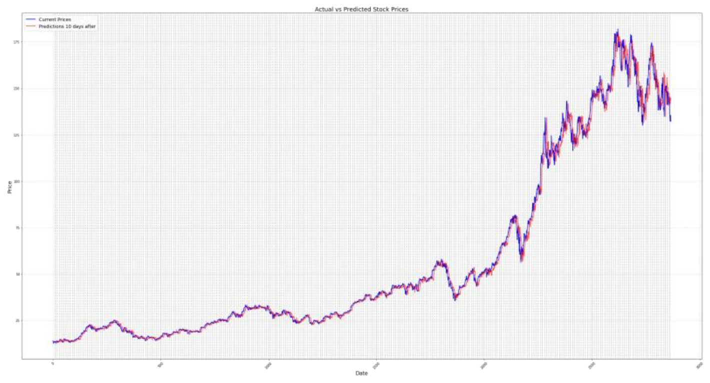
8. **WVAD**: This indicates the flow of money, revealing the balance between buying and selling pressure.
9. **MACD**: Illustrates the relationship between two moving averages of a stock's price. It's accompanied by:
10. **macd_line**: The main line indicating the trend.
11. **signal_line**: The trigger for buy and sell signals.
12. **RSI**: Measures the speed and change of price movements, often used to identify overbought or oversold conditions.
13. **CCI**: Helps in determining cyclical trends.
14. **BB_Upper, BB_Lower, Buy_Signal & Sell_Signal**: These boundaries of the Bollinger + RSI, Double Strategy serve as volatility indicators.
15. **WVF, WVF_color, upperBand & rangeHigh**: Relates to the Williams Vix Fix, identifying bottoms in stock advancements.
16. **VPT**: Combines volume and price to spotlight changes in trend direction.
17. **AD**: Shows the flow of money, offering insights into the accumulation or distribution state of the stock.

The current model performance:

Mean Squared Error (MSE):	4.568
Mean Absolute Percentage Error (MAPE)	1.87%
Root Mean Squared Error (RMSE)	2.137







The displayed residual plot showcases the differences between observed and predicted values for a given model. The data points, represented as blue dots, seem to be scattered randomly around the horizontal red-dotted line, which signifies zero residual or perfect prediction. The random dispersion indicates that the model has a good fit for the data, as there's no discernible pattern or trend in the residuals. This suggests that the model's assumptions, particularly those regarding linearity, independence, and homoscedasticity, are likely met. However, there are a few notable outliers, which might require further investigation to understand if they result from specific external factors or data anomalies. Overall, the residual plot suggests a well-performing model, but attention should be given to the few outliers present.

Current State: Trading Strategy

In the realm of financial forecasting, possessing merely a model that predicts weekly outcomes falls short of the comprehensive approach needed. What truly matters is the development of a sturdy methodology that seamlessly translates these projections into concrete, actionable measures, ultimately leading to a sophisticated trading strategy. To this end, I have architected a straightforward yet effective strategy that seamlessly integrates predictive return analytics with in-depth historical stock price information.

Forecast Generation: Predictions are meticulously crafted for each stock data entry following the stipulated window size. Concurrently, both the predicted and actual returns spanning a 5-day period are discerned.

Portfolio Initialization: The strategy springs to life with a pre-allotted capital and without any initial stock engagements. As it unfolds, the cumulative value of the portfolio, synthesizing both available capital and the value of held stocks, is consistently monitored and documented.

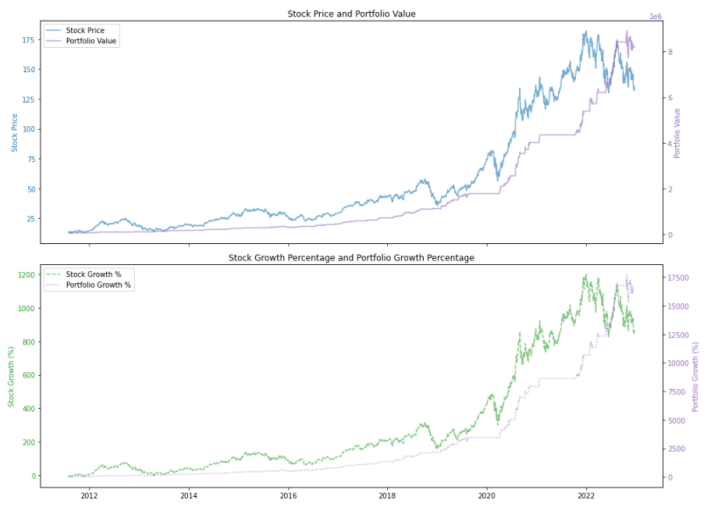
Trading Logic—The Pivotal Mechanism: At the heart of our strategic architecture lies the adaptive trading logic. This mechanism sets buy and sell benchmarks anchored on prior window_size real returns, employing the 75th and 25th percentiles as guiding metrics. For every predictive interval, signals that either breach the buying criteria or fall below the selling criteria are registered. Decisions flow organically from these cues:

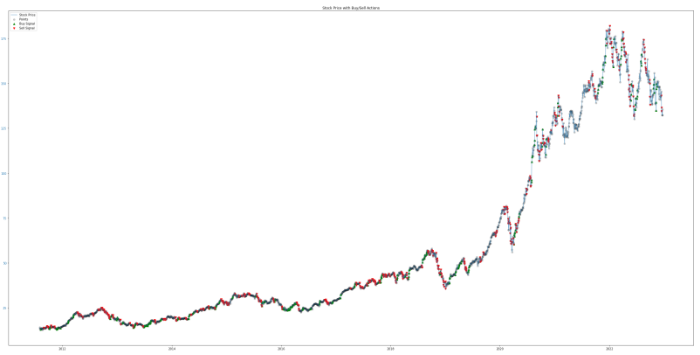
a) Acquisition Strategy: Should there be at least three robust buy prompts within the forecasted range and provided there's ample capital on hand, a stock purchase is greenlit.

b) Divestment Strategy: On the flip side, if three or more sell prompts surface and stocks are held, a divestiture move is undertaken.

Strategy Visualization: The tangible outcomes of the trading strategy are vividly portrayed through two primary graphical illustrations: The premier chart contrasts the stock's market performance with the trajectory of the investor's portfolio. In tandem, the following chart illuminates the parallel growth dynamics of the stock and the portfolio.

Complementing these, an exhaustive visual narrative plots out each buy/sell maneuver against the backdrop of the stock's price timeline.





Currently, our model and trading strategy outperform the stock's return by 10x, demonstrating the ability to yield profits even amidst bearish market trends. While our present framework adeptly identifies selling cues, it requires further refinement in effectively discerning buying signals.

Robustness Test

The robustness of the stock prediction model is assessed by examining its performance across various stocks, considering their average weekly returns and the volatility of these returns. The following table encapsulates the statistical data derived from the model's application to different stocks from 2010 to 2023:

The model demonstrates a consistent predictive accuracy across a range of stocks with varying levels of volatility. This consistency is indicative of the model's robustness and its potential utility as a predictive tool in diverse market conditions.

Stock	Model Performance (MAPE)	Volatility (Std. Dev. of Weekly Returns)
TSLA	3.16%	7.71%
AAPL	1.64%	3.89%
MSFT	1.46%	3.23%
AMZN	1.87%	4.31%
GOOG	1.51%	3.67%

The data reveals that the model maintains a high level of accuracy across various stocks, even when faced with differing degrees of return volatility. Notably, the model shows exceptional precision with AAPL and GOOG, where the Mean Absolute Percentage Error (MAPE) is relatively low, suggesting a stronger predictive capability for these stocks. This is particularly impressive given that these stocks also have lower volatility in their weekly returns, which may contribute to the model's effectiveness.

Rolling Window Analysis

In our pursuit to refine the stock prediction model, we conducted an in-depth analysis of the impact of varying rolling window sizes on the model's accuracy. This analysis was performed using Apple's stock data, ranging from 2010 to 2023. The rolling window approach is critical as it simulates a dynamic environment where the model is periodically updated to reflect the most recent data, thereby potentially enhancing its predictive accuracy.

The table below presents the outcomes of this analysis, where we maintained all other variables constant while systematically altering the window size. The performance metrics used to evaluate the model's effectiveness are the Mean Absolute Percentage Error (MAPE) and the Root Mean Squared Error (RMSE), which provide insights into the model's precision and reliability.

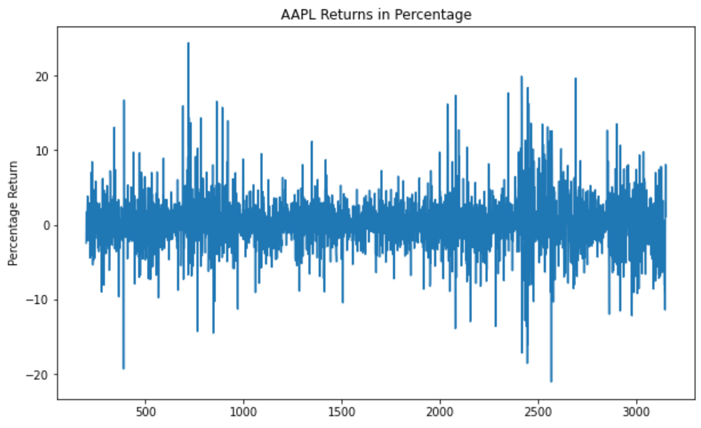
Window Size (Days)	Mean Absolute Percentage Error (MAPE)	Root Mean Squared Error (RMSE)

5	1.64%	1.859
10	1.77%	2.034
20	1.76%	1.965
40	1.72%	1.985
80	1.73%	1.957
160	1.88%	2.122
320	1.98%	2.254
640	1.99%	2.375
1280	2.04%	2.683
2560	2.23%	4.278

The unexpected findings from our rolling window analysis on Apple's stock data suggest that smaller windows are more effective for predictive accuracy. A 5-day window achieved the lowest MAPE and RMSE, while larger windows resulted in higher error rates. This could be attributed to the non-stationary nature of stock prices. In traditional time series analysis, stationarity is a key assumption, meaning the statistical properties of the series do not change over time. Stock prices, however, are typically non-stationary—they do not hover around a constant mean or variance.

The paper by Bao, Yue, and Rao (2017) indicates that deep learning approaches, such as those involving stacked autoencoders and LSTM networks, can handle non-stationarity in financial time series data more effectively than traditional methods, which we are currently using. This could explain why smaller windows, which may capture more recent, relevant trends, outperform larger ones that include more historical data, potentially diluting the model's focus.

Given this context, it might be advantageous to predict daily returns as a percentage rather than absolute stock prices. Returns are often considered stationary because they represent relative changes, which are more likely to fluctuate around a constant mean. The provided ADF (Augmented Dickey-Fuller) statistics and stock return of APPLE support this, with a highly negative ADF statistic and a p-value of 0.0, indicating strong evidence against the null hypothesis of a unit root (non-stationarity).



ADF Statistic	54.82
p-value	0.0
Critical Value (1%)	-3.432
Critical Value (5%)	-2.862
Critical Value (10%)	-2.567

The critical values further reinforce this, as the ADF statistic is well below the threshold for the 1%, 5%, and 10% levels, suggesting that the series is stationary. In layman's terms, the statistical test confirms that the percentage returns of Apple's stock do not follow a random walk and have a consistent pattern over the 10-year period, which is favorable for predictive modeling using traditional time series analysis techniques.

By focusing on returns instead of prices, we may improve the model's predictive performance, as stationary data aligns better with the assumptions of many time series forecasting methods. This shift could potentially address the limitations observed with non-stationary price data and lead to more accurate and reliable predictions.

To investigate whether the non-stationary nature of stock prices negatively impacts the performance of our predictive model, we developed an alternative model using identical parameters and features. However, instead of predicting stock prices, this new model forecasts stock returns. Here are the comparative results:

Model	Mean Square Error	Mean Absolute Error
Pricing Model	7.95	2.03
Return Model	4.79	1.56

The Return Model exhibits a lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) compared to the Pricing Model. This suggests that forecasting returns, which are typically stationary, results in more accurate predictions than forecasting non-stationary prices, when apply XGBoost model (non-neural network model).

Potential issues with return model

The transition to a return-based model, while beneficial for its predictive accuracy, presents a new challenge: traditional market indicators like Bollinger Bands, which are price-based and inherently non-stationary, lose their direct applicability. These indicators are designed to provide insights based on the price movements and volatility of stocks, which are not immediately translatable to a model that focuses on returns.

To address this, we must consider alternative approaches to incorporate the valuable information these indicators provide. One potential solution is to transform these indicators into a form that reflects the relative change in price, which could then be aligned with the return-based perspective of the model. For instance, instead of using Bollinger Bands based on absolute price, we could calculate them based on a percentage change from a moving average.

Another approach could involve the development of new, return-oriented indicators that capture similar aspects of market behavior as Bollinger Bands do for price. These indicators would need to be designed to reflect the volatility and trends in stock returns, rather than stock prices.

In summary, while the shift to a return-based model necessitates a reevaluation of traditional price-based indicators, it also opens up avenues for innovation in the development of new analytical tools tailored for return-based analysis. This adaptation will be crucial for maintaining the integrity and usefulness of technical indicators within the context of a return-focused predictive model.

Next Step

Moving forward, there are several avenues to explore to enhance the robustness and efficacy of our stock prediction model:

a) Feature Enrichment: Dive deeper into the inclusion of potential predictors that encapsulate global economic shifts and overarching market dynamics. Such features can be pivotal in capturing exogenous shocks and external factors influencing stock prices.

b) Addressing Multicollinearity: A systematic evaluation of the features is essential to ascertain any collinearity present. Multicollinearity can undermine the model's interpretability and diminish its predictive prowess. Utilizing techniques like Variance Inflation Factor (VIF) can assist in detecting and mitigating these issues.

c) Advanced Trading Strategies: Expand the trading strategy's scope to encompass more sophisticated tactics such as short-selling. This would allow capitalization on both upward and downward market movements, offering a more holistic trading approach.

d) Refining Buy-Signal Identification: Given the current model's shortcoming in accurately pinpointing buying signals, targeted efforts should be made to optimize this aspect. This might involve recalibrating threshold values or integrating alternative algorithms.

e) Incorporating Sentiment Analysis: A key dimension that's often overlooked is the sentiment prevailing among retail investors. Once a reliable sentiment analysis model is in place, merging it with the current framework could provide a more rounded perspective on market movements. Analyzing chatter on social media platforms, financial forums, or news outlets can be instrumental in this regard.

f) Model Evaluation and Continuous Feedback: It would be prudent to establish a feedback loop where the model's predictions are constantly compared with actual outcomes. Such a mechanism would be invaluable for ongoing model refinement. Also, other models other than XGBoost are left to be experimented.

a) Stress Testing: Given the unpredictable nature of financial markets, stress-testing the model under various hypothetical adverse scenarios can provide insights into its resilience and areas of potential vulnerability.

By adopting these strategies and continually iterating based on real-world outcomes, we can aspire to achieve a state-of-the-art stock prediction model that's both adaptive and predictive in an ever-evolving market landscape.

Citations:

Stock

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE, 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944

Relation: Bao, Yue, and Rao present a deep learning framework utilizing stacked autoencoders and long-short term memory for analyzing financial time series. Notably, they introduce concepts of buy and sell signals based on predicted prices, resonating with our exploration into machine learning-driven financial predictions.

Differentiation: While they lay the groundwork in understanding financial time series through deep learning, our research extends this by incorporating contemporary machine learning methodologies to forecast stock returns over shorter durations. Additionally, we delve into portfolio management through our trading strategy, a topic not explored in their paper.

Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. The Journal of Finance and Data Science, 2(1), 42-57. https://doi.org/10.1016/j.jfds.2016.03.002

Relation: This paper delves into trading signals and the intricacies of implementing a comprehensive trading strategy. Its content is rich in explaining how trading decisions can be informed and executed.

Differentiation: Unlike the paper's emphasis on broader sectors like SPY, our approach zeroes in on individual stocks. Our research also capitalizes on a myriad of indicators, dedicating significant effort to feature selection and engineering, aspects that weren't as extensively addressed in the referenced paper.

Pezim, B. (2018). How To Swing Trade. Preface by A. Aziz. ISBN: 9781726631754.

Relation: The book provides an extensive overview of swing trading strategies and market dynamics, setting the stage for our exploration of stock market behaviors.

Differentiation: Our project enhances these basic principles with state-of-the-art machine learning techniques to forecast stock market returns, delivering a modern, technology-enhanced viewpoint.

Online Social Media and Stock Market:Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), 1-8

Siganos, A., Vagenas-Nanos, E., & Verwijmeren, P. (2014). Facebook's daily sentiment and international stock markets. Journal of Economic Behavior & Organization, 107, 730-743.

Chen, H., De, P., Hu, Y. J., & Hwang, B. H. (2014). Wisdom of crowds: The value of stock opinions transmitted through social media. Review of Financial Studies, 27(5), 1367-1403.

Sentiment Analysis and Opinion Mining:

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

Kumar, A., & Lee, C. M. (2016). Retail investor sentiment and return comovements. The Journal of Finance, 61(5), 2451-2486.

Huang, A. H., Wang, H., & Yang, Y. (2022). FinBERT: A Large Language Model for Extracting Information from Financial Text. Contemporary Accounting Research.

Yang, Y., Uy, M. C. S., & Huang, A. (2020). Finbert: A pretrained language model for financial communications. arXiv preprint arXiv:2006.08097.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). FastText: Enriching Word Vectors with Subword Information. arXiv preprint arXiv:1607.04606.

Malo, P., Sinha, D., Nandi, S., Lytinen, S., & Li, S. (2014). Financial PhraseBank.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Bidirectional Encoder Representations from Transformers. arXiv preprint arXiv:1810.04805.

NLP Techniques for Financial Markets:

Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. The Journal of Finance, 66(1), 35-6

GitHub

https://github.com/howie-zeng/Analyzing-the-Correlation-Between-Retail-Traders--Sentiments-and-Equity-Market-Movements