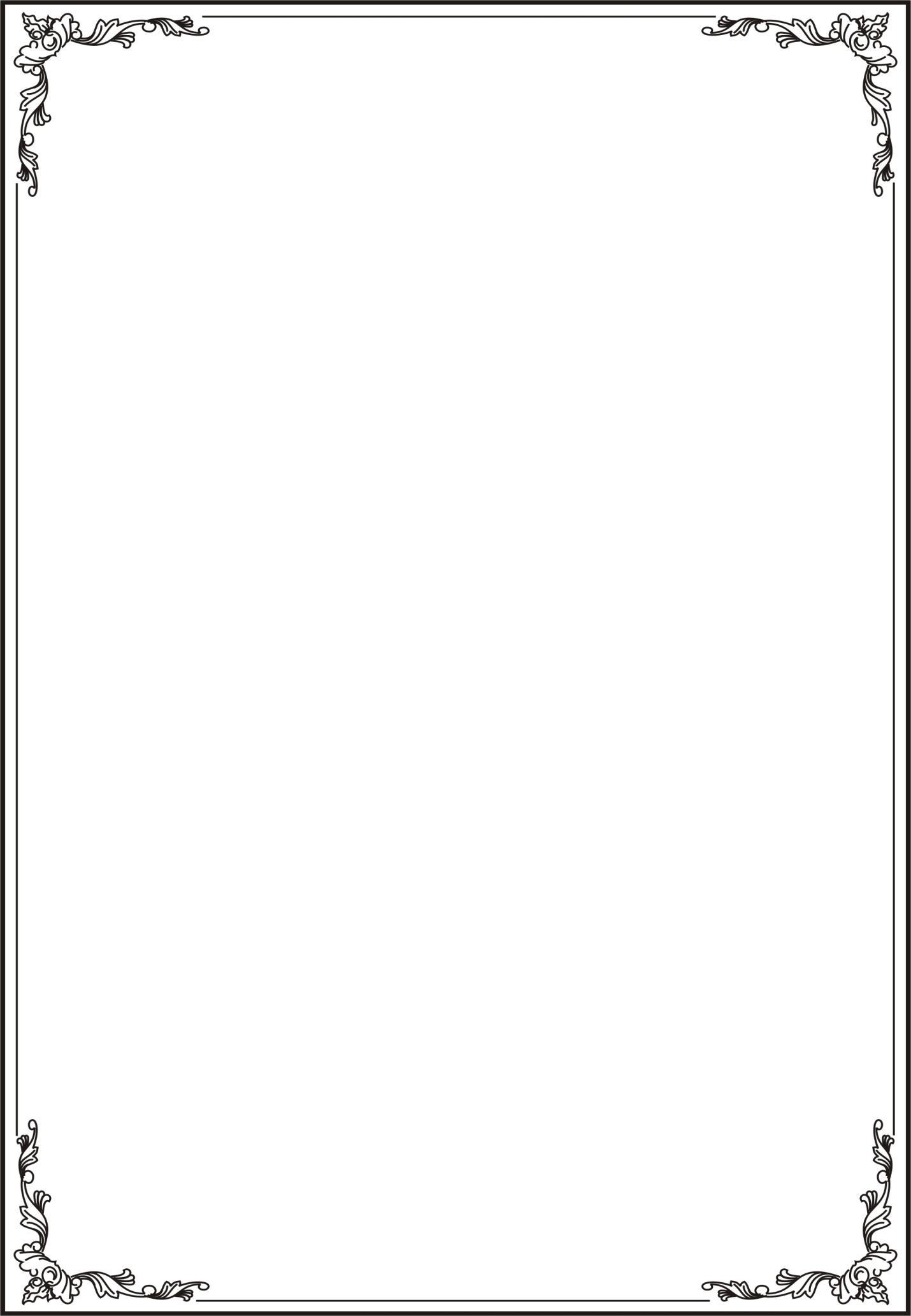
****

**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH-QUALITY TRAINING**

**----------------------------**



**Capstones Project**

**Topic: Building Remider Mobile Application**

**using Machine Learning**

**Instructor: Tran Nhat Quang, MSc**

|  |  |
| --- | --- |
| Nguyen Hoang Anh Khoa | 19110514 |
| Nguyen Tan Dat | 19110116 |
| Thai Thi Thu Thao | 19110515 |

**Members:**

*Ho Chi Minh City, Dec 1st 2023*

**SOCIALIST REPUBLIC OF VIETNAM**

**Independence - Freedom – Happiness**

**\*\*\*\*\*\*\***

**GRADUATION THESIS TASK**

**Student’s Name:**

|  |  |  |
| --- | --- | --- |
| 1.Nguyen Hoang Anh Khoa | ID: 19110514 | Class: 1911CLA5 |
| 2.Nguyen Tan Dat | ID: 19110116 | Class: 1911CLA4 |
| 3. Thai Thi Thu Thao | ID:19110515 | Class: 1911CLA4 |

Major: **Information Technology**

Lecturer: **Tran Nhat Quang, MSc**  Phone: ………………

The date of receiving the topic: 02/09/2023 Thesis submission date: 01/12/2023

Project name: **Building Remider Mobile Application using Machine Learning**

Content implementation of the topic:

- Research and analysis stock market behavior by reading papers.

- Study models that have performing well on forecasting time series dataset.

- Training and Evaluation models have well performance.

- Build a mobile application that can help user get notification with time specify, using NLP and voice regconization

LECTURER

**Tran Nhat Quang**

**SOCIALIST REPUBLIC OF VIETNAM**

**Independence – Freedom – Happiness**

\*\*\*\*\*\*\*

**INSTRUCTOR’S COMMENTARY**

**Students’ names:**

|  |  |  |
| --- | --- | --- |
| 1.Nguyen Hoang Anh Khoa | ID: 19110514 | Class: 1911CLA5 |
| 2.Nguyen Dat | ID: | Class: |
| 3. Thai Thi Thu Thao | ID: | Class: |

Major: **Information Technology**

Project’s name: **Building Remider Mobile Application using Machine Learning**

Reviewer’s full name: **Tran Nhat Quang, MSc**

**COMMENTARY**

Project’s contents and workload:

**Theory:**

- Research and learn Javascript, Python, Machine Learning, Flask, Flutter, Dart.

**Implementation:**

- Build a mobile application that can help user get notification with time specify, using NLP and voice regconization.

**Results:**

- The contents and workload are appropriate for an undergraduate capstone project. Students can apply the technologies learned to successfully build a system with basic features.

**Link:**

https://github.com/anhkhoanghg/gpt.git

**Advantages**:

* The technologies learned and applied are contemporary and powerful.
* The features of the system can somehow fulfill the business process.
* The models performing well on entire datasets.

**Disadvantages**:

* The features are quite simple. The group should have built more advanced features for better support to the users.
* Text Analysis has to be improve.
* Website should update.

Lack Approving for dissertation defense or not? Yes.

**Achievement level:** Good.

*Ho Chi Minh, Thu December 25th 2023*

*Supervisor*

*(Sign, write full name)*

**Tran Nhat Quang**

**SOCIALIST REPUBLIC OF VIETNAM**

**Independence – Freedom – Happiness**

\*\*\*\*\*\*\*

**REVIEWER’S COMMENTARY**

**Students’ names:**

|  |  |  |
| --- | --- | --- |
| 1.Nguyen Hoang Anh Khoa | ID: 19110514 | Class: 1911CLA5 |
| 2.Nguyen Dat | ID: | Class: |
| 3. Thai Thi Thu Thao | ID: | Class: |

Major: **Information Technology**

Project’s name: **Building Remider Mobile Application using Machine Learning** Reviewer’s full name:

**COMMENTARY:**

1. Regarding the content of the topic and the volume of implementation:

1. Strength:

1. Drawback:

1. Recommend for defending or not?

5. Grade:

6. Mark:

*Ho Chi Minh, December 25st 2023*

Reviewer

*(Sign, write full name)*

***Contents***

[**Chapter 01: Related work** 8](#_Toc137516930)

[**1.1. Stock Price Prediction Using LSTM on Indian Share Market** [1]**:** 8](#_Toc137516931)

[**1.2. AEI-DNET: A Novel DenseNet Model with an Autoencoder for the Stock Market Predictions Using Stock Technical Indicators** [2]**:** 8](#_Toc137516932)

[**1.3. Predict stock prices with ARIMA and LSTM** [3]**:** 8](#_Toc137516933)

[**1.4. Convolutional neural networks for time series classification** [4]**:** 8](#_Toc137516934)

[**1.5. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling** [5]**:** 8](#_Toc137516935)

[**Chapter 02: Introduction** 10](#_Toc137516936)

[**Chapter 03: Theory Fundamental** 12](#_Toc137516937)

[**3.1.** **System Architecture** 12](#_Toc137516938)

[**3.2.** **Library** 12](#_Toc137516939)

[**3.3.** **Technologies** 14](#_Toc137516940)

[**3.3.1.** **Flask** 14](#_Toc137516941)

[**3.3.2.** **TensorFlow** 14](#_Toc137516942)

[**3.3.3.** **Torch** 14](#_Toc137516943)

[**Chapter 04: System Analysis and Design** 14](#_Toc137516944)

[**4.1. Use-case diagram** 15](#_Toc137516945)

[**4.2. Use-case specification** 15](#_Toc137516946)

[**5.1. Introduction** 18](#_Toc137516947)

[**5.2. Visual Design and Layout** 18](#_Toc137516948)

[**5.3. Key Sections and Features:** 18](#_Toc137516949)

[**5.4. User Experience and Responsiveness:** 20](#_Toc137516950)

[**5.5. Conclusion** 21](#_Toc137516951)

[**Chapter 06: Data preparation** 22](#_Toc137516952)

[**6.1 Data Collection** 22](#_Toc137516953)

[**6.2. Data Preprocessing** 23](#_Toc137516954)

[**6.3. Time Series Data Preparation** 26](#_Toc137516955)

[**6.4. Indicators:** 27](#_Toc137516956)

[6.4.1. Momentum 27](#_Toc137516957)

[6.4.2. William Percentage R 28](#_Toc137516958)

[6.4.3. RSI 28](#_Toc137516959)

[6.4.4. Commodity Channel Index 29](#_Toc137516960)

[6.4.5. Moving Average Convergence/Divergence 29](#_Toc137516961)

[6.4.6. Directional Movement 30](#_Toc137516962)

[6.4.7. Forecast Oscillator 31](#_Toc137516963)

[6.4.8. Chande Momentum Oscillator 31](#_Toc137516964)

[6.4.9. Efficiency Ratio 32](#_Toc137516965)

[6.4.10. Rate of Change 33](#_Toc137516966)

[6.4.11. The Schaff Trend Cycle 34](#_Toc137516967)

[6.4.12. Elder's Bulls Ray Index 34](#_Toc137516968)

[6.4.14. Simple Moving Average 35](#_Toc137516969)

[6.4.15. Exponential Moving Average 36](#_Toc137516970)

[6.4.16. Volume Weighted Average Price 37](#_Toc137516971)

[6.4.17. Hull Moving Average 37](#_Toc137516972)

[6.4.18. Chaikin Money Flow 38](#_Toc137516973)

[6.4.19. Ultimate Oscilators 39](#_Toc137516974)

[**6.4.20. Slope 39**](#_Toc137516975)

[**Chapter 07: Methodology** 41](#_Toc137516976)

[**7.1. LSTM** 41](#_Toc137516977)

[**7.2. Random Forest** 41](#_Toc137516978)

[**7.3. Support Vector Machine** 42](#_Toc137516979)

[**7.4. Natural Language Processing** 42](#_Toc137516980)

[**7.5. XGBoost** 48](#_Toc137516981)

[**Chapter 08: Evaluation** 50](#_Toc137516982)

[**8.1. Sectors Predicted** 50](#_Toc137516983)

[**8.2. Model Evaluation** 50](#_Toc137516984)

[**Chapter 09: Conclusion** 66](#_Toc137516985)

[**References** 68](#_Toc137516986)

# **Chapter 01: Related work**

## **1.1. Discord Reminder Bot with the NVIDIA Jetson Nano**

Achyut Ghosh et al. conducted a study using LSTM to identify the ideal window for predicting future share prices. at various banks and sectors over various time periods. The study concludes that businesses in the same industry have comparable growth rates and dependencies. A larger dataset can be used to train the model, which will increase prediction accuracy. The findings indicate that the prediction error will gradually decrease over time, and that the longer the prediction period, the less error there will be.

## **1.2. AEON: A Method for Automatic Evaluation of NLP Test Cases**

These test cases require extensive manual checking effort, and instead of improving NLP software, they can even degrade NLP software when utilized in model training. To address this problem, we propose AEON for Automatic Evaluation Of NLP test cases. For each generated test case, it outputs scores based on semantic similarity and language naturalness. We employ AEON to evaluate test cases generated by four popular testing techniques on five datasets across three typical NLP tasks. The results show that AEON aligns the best with human judgment. In particular, AEON achieves the best average precision in detecting semantic inconsistent test cases, outperforming the best baseline metric by 10%. In addition, AEON also has the highest average precision of finding unnatural test cases, surpassing the baselines by more than 15%. Moreover, model training with test cases prioritized by AEON leads to models that are more accurate and robust, demonstrating AEON's potential in improving NLP software.

## **1.3. Increasing Students' Engagement to Reminder Emails Through Multi-Armed Bandits**

Using Multi-Armed Bandits (MAB) algorithms like Thompson Sampling (TS) in adaptive experiments can increase students' chances of obtaining better outcomes by increasing the probability of assignment to the most optimal condition (arm), even before an intervention completes. This is an advantage over traditional A/B testing, which may allocate an equal number of students to both optimal and non-optimal conditions. The problem is the exploration-exploitation trade-off. Even though adaptive policies aim to collect enough information to allocate more students to better arms reliably, past work shows that this may not be enough exploration to draw reliable conclusions about whether arms differ. Hence, it is of interest to provide additional uniform random (UR) exploration throughout the experiment.

# **Chapter 02: Introduction**

* 1. **Reason for chosen topic**
  2. **Purpose of project**
  3. **Object and Scope**
  4. **Expected Result**

# **Chapter 03: Theory Fundamental**

* 1. **System Architecture**
  2. **Library**
     1. **Front-end**

|  |  |  |
| --- | --- | --- |
| Library | Version | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* + 1. **Back-end**

|  |  |  |
| --- | --- | --- |
| Library | Version | Description |
| scikit-learn | 1.1 | data mining and data analysis |
| transformer | 4.29.2 | save you the time and resources required to train a model |
| torch | 2.0 | load data, build deep neural networks, train and save your models |
| pandas | 2.0.2 | repetitive tasks associated with working with data |
| numpy | 1.17.3 | perform a wide variety of mathematical operations on arrays |
| request | 2.31.0 | send HTTP requests using Python |
| tqdm | 4.65.0 | create progress bars, training machine learning models, multi-loop Python function, and downloading data |
| flask | 2.3.2 | developing web applications |
| subprocess | 0.0.8 | run new codes and applications by creating new processes |
| pathlib | 1.0 | provides a modern and Pythonic way of working with file paths, making code more readable and maintainable |
| math | 3.11.3 | use the built-in mathematical operators |
| beautifulsoup | 4.12.2 | used for web scraping purposes to pull the data out of HTML and XML files |
| parsedatetime | 2.6 | Parse human-readable date/time strings |
| python\_dateutil | 2.8.2 |  |
| pandas\_ta | 0.3.14b | Easier to use library |
| spacy | 3.5.3 | spaCy is designed for tasks such as part-of-speech tagging, named entity recognition, and dependency parsing. |
| gunicorn | 19.7.1 | Gunicorn provides a simple and efficient way to serve web applications with multiple worker processes. |

* 1. **Technologies**
     1. **Flask**

Flask is a popular and lightweight web framework for building web applications using the Python programming language. It is known for its simplicity, flexibility, and ease of use, making it a popular choice among developers, especially for small to medium-sized projects.

Flask supports template engines like Jinja2, which allows developers to separate the presentation logic from the application's code. Templating enables the creation of dynamic HTML pages by rendering data into predefined templates.

Flask uses a routing mechanism to map URLs to corresponding Python functions or views. Developers can define routes and associated functions to handle specific URLs and HTTP methods. This allows for the creation of clean and intuitive URL structures.

Flask-WTF is an extension that integrates with Flask to provide support for handling web forms. It simplifies form validation, CSRF protection, and other form-related tasks.

* + 1. **TensorFlow**

TensorFlow is an open-source machine learning framework developed by Google. It is widely used for building and training machine learning models, particularly deep neural networks. TensorFlow provides a comprehensive ecosystem of tools, libraries, and resources that enable developers to create and deploy machine learning applications efficiently.

* + 1. **Torch**

Like TensorFlow, Torch, or PyTorch, is an open-source machine learning framework widely used for developing and training deep learning models. It provides a flexible and efficient platform for building neural networks and conducting various machine learning tasks. Torch is known for its dynamic computational graph, extensive support for GPU acceleration, and easy-to-use APIs.

# **Chapter 04: System Analysis and Design**

## **4.1. Use-case diagram**

## **4.2. Use-case specification**

1. **View analysis chart**

|  |  |
| --- | --- |
|  | |
| **Description** |  |
| **Activation agent** |  |
| **Pre-conditions** |  |
| **Step to be taken** |  |

1. **Customize chart specificities**

|  |  |
| --- | --- |
|  | |
| **Description** |  |
| **Activation agent** |  |
| **Pre-conditions** |  |
| **Step to be taken** |  |

1. **View stock prediction**

|  |  |
| --- | --- |
|  | |
| **Description** |  |
| **Activation agent** |  |
| **Pre-conditions** |  |
| **Step to be taken** |  |

1. **View technical indicators**

|  |  |
| --- | --- |
|  | |
| **Description** |  |
| **Activation agent** |  |
| **Pre-conditions** |  |
| **Step to be taken** |  |

1. **Download analysis chart**

|  |  |
| --- | --- |
|  | |
| **Description** |  |
| **Activation agent** |  |
| **Pre-conditions** |  |
| **Step to be taken** |  |

**Chapter 05: Interface evaluation**

## **5.1. Introduction**

## **5.2. Visual Design and Layout**

## **5.3. Key Sections and Features:**

# **Chapter 06: Data preparation**

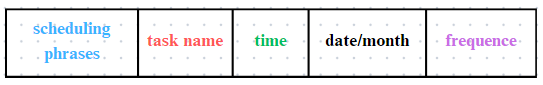
## **6.1 Data Preparation**

The data collection phase was meticulously undertaken by our team to compile a comprehensive dataset for our project. Within this dataset, we systematically cataloged various tasks that users typically engage in for their daily and routine activities. The evaluation of these tasks was performed through our own assessment, considering their regularity and significance in users' scheduling patterns.In total, our dataset encompasses around 2000 distinct tasks, each accompanied by detailed information. For each task, we not only recorded its inherent importance but also estimated the approximate time required for its completion. This dual categorization allows for a nuanced understanding of the dataset, providing insights into both the priority and temporal aspects of the tasks included.

Our team conducted a thorough analysis to ensure the relevance and representation of diverse tasks within the dataset. We aimed to capture a spectrum of activities that individuals commonly encounter in their daily lives. The tasks span various domains, reflecting the dynamic nature of users' scheduling needs.The inclusion of importance ratings provides a qualitative dimension to the dataset, enabling a nuanced exploration of task priorities. Additionally, the time estimates associated with each task offer valuable insights into the anticipated effort required for task completion. These attributes collectively contribute to the richness of our dataset, making it a robust resource for understanding user behavior and preferences in task management.

Subsequently, we leveraged the curated set of tasks to generate user-friendly sentences that individuals can employ for scheduling their activities. These sentences are carefully crafted to encompass various components, including scheduling terms, task names, time specifications, dates, and the recurrence frequency of tasks. The integration of these elements results in the creation of coherent and complete sentences that users can readily adopt for efficient task scheduling.The scheduling terms embedded in the sentences serve as cues for organizing and planning activities. These terms are strategically selected to resonate with users' scheduling preferences and habits. Additionally, task names are seamlessly integrated, ensuring clarity and specificity in the scheduling process. Users can easily identify and relate to their intended tasks through the inclusion of these names.Time specifications and date references add a temporal dimension to the sentences, enabling users to precisely allocate tasks within their schedules. Whether it's a specific time of day, a duration, or a particular date, the sentences provide flexibility to cater to diverse scheduling needs. Furthermore, the recurrence frequency of tasks is incorporated to accommodate repetitive activities, allowing users to efficiently plan and manage recurring commitments.

The culmination of these components results in well-formed sentences that encapsulate the essential details needed for effective scheduling. By providing users with ready-made sentences, our system aims to streamline the task management process, offering a convenient and user-centric approach to creating comprehensive and meaningful schedules. This approach not only enhances the user experience but also contributes to the overall efficiency and effectiveness of task scheduling within the scope of our project.



The components within the sentence can be repositioned based on the context of the utterance.

## **6.2. Time Series Data Preparation**

## **6.3. Indicators:**

# **Chapter 07:** **Methodology**

## **7.1. LSTM**

LSTM represents a powerful and widely used approach for processing sequential data. This architecture has unique mechanisms such as the forget gate, input gate, and output gate, which enable it to selectively store and discard information while maintaining important global information in the cell state. Compared to traditional recurrent neural networks (RNNs), which suffer from the vanishing gradient problem, LSTM's incorporation of memory cells and gating mechanisms have demonstrated superior performance on a range of sequential tasks. In particular, the deep structure of non-linear functions in LSTM makes it well-suited for handling time series data efficiently while using fewer computational resources.

Increasing the number of layers and hidden units in an LSTM can improve its ability to model structured data through linear combinations. However, stock data consists of multiple dimensions, such as high, low, close, and various indicators, which are not always related. As a result, an LSTM may not be able to capture the complex patterns and relationships within the data. Additionally, the predictive nature of an LSTM is based on past events, which may not accurately reflect the dynamic and ever-changing nature of the stock market. To address these issues, researchers have proposed using 1D convolutions to extract features in both temporal and spatial dimensions, which can be fed into an LSTM for improved performance in modeling stock market data.

## **7.2. Random Forest**

In the context of stock market analysis, Random Forest utilizes historical stock data and relevant features such as price and technical indicators. By constructing an ensemble of decision trees, the algorithm collectively leverages the diversity and independence of individual trees to generate more accurate predictions.

Random Forest overcomes limitations of single decision trees by employing two key techniques: random feature selection and bootstrap aggregating (or bagging). Random feature selection involves randomly selecting a subset of features at each node of a decision tree, which reduces the potential bias of the model and promotes robustness. Bagging, on the other hand, involves randomly sampling the training data with replacement, enabling each decision tree in the forest to learn from slightly different subsets of the data.

The final prediction from the Random Forest algorithm is obtained through aggregating the predictions of individual decision trees. This ensemble approach ensures improved generalization, increased stability, and reduced overfitting compared to single decision tree models.

## **7.3. Support Vector Machine**

Support Vector Machines (SVM) is a machine learning algorithm widely used in predicting stock trends. SVM operates on the principle of finding an optimal hyperplane that best separates different classes of data points. In the context of stock trend prediction, SVM seeks to classify stock prices into either an upward or downward trend. By utilizing historical stock data and relevant features such as price, volume, and technical indicators, SVM analyzes patterns and establishes decision boundaries to distinguish between bullish and bearish trends. The algorithm aims to maximize the margin between the hyperplane and the closest data points, thereby improving its ability to generalize and make accurate predictions for unseen data. SVM's robustness, ability to handle high-dimensional data, and flexibility in incorporating various features make it a valuable tool for traders and investors seeking insights into stock market trends.

One of the key advantages of SVM is its ability to handle high-dimensional data, making it suitable for incorporating numerous features and indicators that influence stock price movements. Moreover, SVM aims to maximize the margin between the hyperplane and the nearest data points, enhancing its ability to generalize and make accurate predictions for unseen data.

## **7.4. Natural Language Processing**

**7.4.1. Word Embedding**

Word embedding is a powerful technique in natural language processing (NLP) that transforms words or phrases into dense vector representations in a continuous vector space. It overcomes the limitations of traditional sparse and high-dimensional representations by capturing semantic and contextual relationships between words. This report provides an overview of word embedding, its significance in NLP, and its applications in various tasks.

Word embedding techniques, such as Word2Vec, have revolutionized NLP by enabling machines to understand and process textual data more effectively. Unlike traditional methods that represent words as sparse and high-dimensional vectors, word embeddings map words to dense vectors, where similar words are represented by vectors that are closer in the vector space. This dense representation captures semantic relationships, allowing algorithms to understand the meaning of words and infer relationships. For example, words like "king" and "queen" or "man" and "woman" have similar vector representations, enabling algorithms to perform word analogy tasks. Furthermore, word embeddings capture contextual similarities by assigning similar vector representations to words that appear in similar contexts. This contextual understanding enhances the performance of algorithms in various NLP tasks.

Word embeddings have found extensive applications in NLP tasks such as sentiment analysis, machine translation, text classification, and information retrieval. By utilizing word embeddings, algorithms can leverage the semantic and contextual relationships between words to improve accuracy and performance. Pre-trained word embeddings like GloVe and FastText are available and provide a solid starting point for NLP tasks. These embeddings are trained on large corpora and capture general language semantics. However, it is also possible to train domain-specific word embeddings using specific datasets to capture domain-specific semantics and contextual information. This flexibility allows NLP practitioners to tailor word embeddings to the specific requirements of their tasks and achieve better results.

In summary, word embedding is a fundamental technique in NLP that captures semantic and contextual relationships between words by representing them as dense vectors in a continuous vector space. The ability to encode semantic and contextual information within these vector representations has transformed the field of NLP, enabling algorithms to understand and process textual data more effectively. By capturing word relationships and context, word embeddings have proven invaluable in a wide range of NLP applications, contributing to improved accuracy and performance. As the field of NLP continues to advance, word embedding techniques will play a crucial role in further enhancing the capabilities of natural language understanding and processing systems.

A picture containing text, screenshot, businesscard, font

Description automatically generated

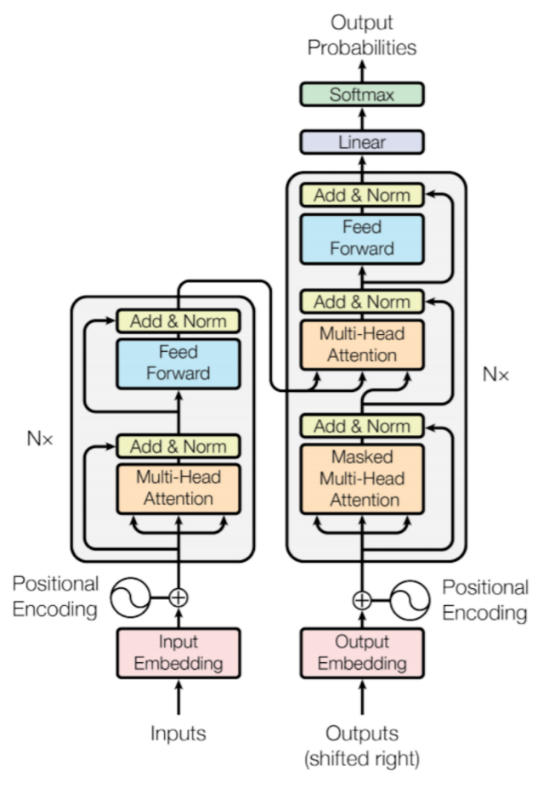
**7.4.2. Transformer Architecture**

The Transformer architecture has emerged as a significant breakthrough in natural language processing (NLP), revolutionizing the field by introducing a self-attention mechanism that captures word dependencies without traditional recurrent or convolutional structures. This groundbreaking approach allows the model to attend to all positions in the input sequence simultaneously, enabling efficient parallelization and effective handling of long-range dependencies. As a result, the Transformer has achieved remarkable success in various NLP tasks, including machine translation, text generation, and language understanding, surpassing previous state-of-the-art results.

At the core of the Transformer architecture is the self-attention mechanism, which fundamentally changes the way models process sequential data. By employing an encoder-decoder structure with multiple layers, each comprising a self-attention module and a position-wise feed-forward neural network, the Transformer enables the model to understand both global and local dependencies within the input sequence. This comprehensive understanding, combined with the ability to perform non-linear transformations between positions, empowers the Transformer to capture intricate linguistic patterns and relationships.

The parallelization-friendly design of the Transformer architecture has further contributed to its success. Unlike traditional sequential models, such as recurrent neural networks (RNNs), the Transformer can process the entire input sequence in parallel. This characteristic leverages the computational power of modern hardware, such as GPUs, leading to faster training and inference times, particularly for longer sequences. Moreover, the Transformer's capacity to learn from vast amounts of data has made it a preferred choice for NLP tasks, where large-scale datasets are often available.

In summary, the Transformer architecture has reshaped the NLP landscape by offering a powerful and efficient alternative to traditional sequence models. Its ability to capture word dependencies through self-attention, along with its parallelization-friendly design, has propelled it to achieve state-of-the-art results in various NLP tasks. With its exceptional performance, the Transformer continues to drive advancements in machine translation, text generation, and language understanding, and its impact on the field is likely to endure. [24]



**7.4.3. BERT (Bidirectional Encoder Representations from Transformers)**

BERT is a new approach to pre-training language representations that achieves state-of-the-art results on eleven natural language processing tasks. BERT is based on the Transformer architecture, which is a neural network architecture that has been shown to be effective for sequence-to-sequence tasks. BERT is pre-trained on a massive dataset of text and code. The pre-training process involves masking some of the tokens in the input and then predicting the missing tokens. BERT can be fine-tuned on a variety of natural language processing tasks. Fine-tuning involves training a BERT model on a specific task, using the pre-trained BERT representations as a starting point.

BERT is a powerful language representation model that can be used for a variety of natural language processing tasks. BERT has been shown to be effective for tasks such as question answering, natural language inference, and sentiment analysis. BERT is a valuable tool for researchers and developers who are working on natural language processing tasks. [25]

A diagram of embedding and encoder

Description automatically generated with low confidence

1. **BERT LARGE**

BERT LARGE, an expanded version of the BERT model, takes the power of BERT to new heights. With 24 transformer layers, 16 attention heads, and 340 million parameters, BERT LARGE offers an even more comprehensive and nuanced understanding of natural language. This increased model size allows BERT LARGE to capture intricate language patterns, semantic relationships, and context at a greater depth.

Similar to its base model, BERT LARGE utilizes the pre-training and fine-tuning paradigm. During pre-training, BERT LARGE is trained on a vast corpus of text and code, learning to predict missing tokens by leveraging the masked language model objective. This process helps BERT LARGE develop a rich representation of language, enabling it to grasp the nuances and complexities of various linguistic tasks.

After pre-training, BERT LARGE can be fine-tuned on specific natural language processing tasks. Fine-tuning involves training BERT LARGE on a task-specific dataset, utilizing the pre-trained representations as a starting point. This approach allows BERT LARGE to adapt its knowledge to the specifics of the target task, resulting in highly accurate and effective models.

BERT LARGE has achieved state-of-the-art results on a wide range of natural language processing tasks. It has demonstrated exceptional performance in tasks such as question answering, natural language inference, sentiment analysis, and more. With its extensive capacity for understanding and representing language, BERT LARGE serves as a valuable tool for researchers and developers working in the field of natural language processing.

1. **Fin-BERT**

Fin-BERT is a pre-trained NLP model to analyze sentiment of financial text. It is built by further training the BERT language model in the finance domain, using a large financial corpus and thereby fine-tuning it for financial sentiment classification. Financial PhraseBank by Malo et al. (2014) is used for fine-tuning.

Fin-BERT is an opensource pre trained Natural Language Processing (NLP) model, that has been specifically trained on Financial data, and outperforms almost all other NLP techniques for financial sentiment analysis.

The main advantage of Fin-BERT is its ability to understand financial jargon, context, and nuances, which are often unique to the finance industry. It can comprehend financial terms, identify sentiment, classify financial news articles, extract financial entities, and perform various other financial text analysis tasks. This makes Fin-BERT a valuable tool for sentiment analysis, stock market prediction, risk assessment, financial news summarization, and other applications in the financial domain.

Fin-BERT has gained popularity and widespread adoption in the finance industry due to its ability to handle the complexities and challenges specific to financial text analysis. By leveraging the pre-trained BERT model and fine-tuning it on financial data, Fin-BERT offers a reliable and efficient solution for extracting insights and understanding the financial landscape from textual information. [26]

**7.4.4. Apply NLP to project**

Our innovative idea revolves around leveraging the power of Natural Language Processing (NLP) for the purpose of summarizing articles specifically focused on the stock market. By utilizing NLP techniques, we aim to develop a robust system that takes the content of an article as input and generates a concise, comprehensive sentence that captures the essence of the stock price movement discussed within the article.

The core objective of our system is to distill complex and often lengthy articles into succinct summaries that highlight the key insights related to stock market fluctuations. By automating this process, we aim to save time for users who need to stay updated with the latest developments in the stock market but may not have the bandwidth to thoroughly read and analyze numerous articles.

Furthermore, we recognize the potential value of providing users with not only article summaries but also the output of our NLP model's analysis for forecasting future stock prices. By combining the power of NLP with predictive modeling, we can empower users with insights and information that can aid them in making informed decisions regarding their investments.

In summary, our ambitious endeavor involves harnessing NLP techniques to generate comprehensive summaries of stock market articles, offering users a concise understanding of the stock price movement discussed within. Additionally, we aim to provide users with the output of our NLP model to assist in forecasting stock prices, empowering them with valuable insights for their investment decisions.

A screenshot of a computer

Description automatically generated with low confidence

## **7.5. XGBoost**

XGBoost (Extreme Gradient Boosting) is a powerful and widely-used machine learning algorithm known for its exceptional performance and scalability. It belongs to the gradient boosting family, where weak prediction models, such as decision trees, are sequentially built to correct the errors made by previous models. XGBoost incorporates regularization techniques, including L1 and L2 regularization, to prevent overfitting and improve model generalization. It also offers a feature importance measure, allowing users to identify the most influential variables in their datasets. Additionally, XGBoost handles missing values effectively without the need for imputation techniques. With its parallel processing capabilities, XGBoost efficiently utilizes multiple CPU cores during training, enabling faster model building. The algorithm employs tree pruning to control model complexity by removing unnecessary branches, and it supports cross-validation for assessing performance and hyperparameter tuning. Being an open-source library, XGBoost is available in various programming languages, making it accessible and widely adopted by the data science community.

In summary, XGBoost stands out as a robust machine learning algorithm due to its exceptional performance, scalability, and advanced features. Its regularization techniques, feature importance measure, and handling of missing values contribute to model accuracy and interpretability. With parallel processing and tree pruning, XGBoost offers efficient and controlled model building. The algorithm's support for cross-validation aids in assessing performance and selecting optimal hyperparameters. Moreover, its open-source nature and availability in multiple programming languages make XGBoost accessible and widely used in both academic research and practical applications.

# **Chapter 08: Evaluation**

## **8.1. Sectors Predicted**

Apple Inc. (AAPL)

Microsoft Corporation (MSFT)

Alphabet Inc. (GOOGL)

Amazon.com, Inc. (AMZN)

Tesla Inc. (TSLA)

## **8.2. Model Evaluation**

* + 1. **Evaluate model**

|  |  |  |
| --- | --- | --- |
| Symbol | Mean Square Error | |
| Train | Test |
| **AAPL** | 0.000147 | 0.000103 |
| **GOOGL** | 0.000171 | 0.000096 |
| **MSFT** | 0.000140 | 0.000105 |
| **AMZN** | 0.000223 | 0.000104 |
| **TSLA** | 0.000176 | 0.000125 |

**Visualize Prediction:**

A picture containing line, plot, diagram, slope

Description automatically generated

* 1. **Trend prediction on test set:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | Output Size | Window Size | Type Data | Balance Test Accuracy | | |
| Random Forest | SVM | XGBoost |
| **AAPL** | 3 | 3 | 0 | 0.73 | 0.50 | 0.67 |
| 1 | 0.63 | 0.50 | 0.60 |
| 2 | 0.69 | 0.54 | 0.69 |
| 7 | 0 | 0.69 | 0.50 | 0.67 |
| 1 | 0.63 | 0.67 | 0.62 |
| 2 | 0.65 | 0.67 | 0.65 |
| 14 | 0 | 0.73 | 0.50 | 0.67 |
| 1 | 0.75 | 0.62 | 0.65 |
| 2 | 0.71 | **0.71** | 0.69 |
| 7 | 3 | 0 | 0.72 | 0.69 | 0.72 |
| 1 | 0.67 | 0.53 | 0.75 |
| 2 | 0.75 | 0.53 | 0.64 |
| 7 | 0 | 0.75 | 0.69 | 0.72 |
| 1 | 0.78 | 0.61 | 0.81 |
| 2 | **0.75** | 0.64 | 0.81 |
| 14 | 0 | 0.75 | 0.69 | 0.72 |
| 1 | 0.81 | 0.61 | 0.58 |
| 2 | 0.81 | 0.61 | 0.64 |
| 14 | 3 | 0 | 0.71 | 0.32 | **0.86** |
| 1 | 0.89 | 0.57 | 0.86 |
| 2 | 0.79 | 0.57 | 0.86 |
| 7 | 0 | 0.86 | 0.32 | **0.75** |
| 1 | 0.65 | 0.54 | 0.71 |
| 2 | 0.71 | 0.54 | 0.68 |
| 14 | 0 | 0.82 | 0.32 | **0.75** |
| 1 | 0.75 | 0.54 | 0.50 |
| 2 | 0.75 | 0.54 | 0.64 |
| Symbol | Output Size | Window Size | Type Data | Full Test Set Accuracy | | |
| Random Forest | SVM | XGBoost |
| **AAPL** | 3 | 3 | 0 | **0.67** | 0.50 | **0.62** |
| 1 | 0.57 | 0.43 | 0.55 |
| 2 | **0.67** | 0.48 | 0.63 |
| 7 | 0 | 0.63 | 0.50 | **0.62** |
| 1 | 0.55 | 0.58 | 0.53 |
| 2 | **0.67** | 0.60 | 0.62 |
| 14 | 0 | 0.65 | 0.50 | **0.62** |
| 1 | 0.63 | 0.55 | 0.57 |
| 2 | **0.67** | 0.60 | 0.63 |
| 7 | 3 | 0 | 0.63 | 0.45 | 0.57 |
| 1 | 0.63 | 0.36 | 0.48 |
| 2 | 0.68 | 0.36 | 0.64 |
| 7 | 0 | 0.63 | 0.45 | 0.57 |
| 1 | **0.70** | **0.66** | 0.61 |
| 2 | **0.70** | **0.66** | 0.66 |
| 14 | 0 | 0.64 | 0.45 | 0.57 |
| 1 | 0.59 | **0.66** | 0.50 |
| 2 | **0.70** | **0.66** | 0.57 |
| 14 | 3 | 0 | 0.65 | 0.22 | 0.63 |
| 1 | 0.61 | 0.49 | 0.57 |
| 2 | **0.71** | 0.49 | 0.61 |
| 7 | 0 | 0.69 | 0.22 | 0.65 |
| 1 | 0.68 | 0.49 | 0.51 |
| 2 | 0.67 | 0.49 | 0.57 |
| 14 | 0 | 0.69 | 0.22 | 0.65 |
| 1 | 0.53 | 0.59 | 0.37 |
| 2 | 0.65 | 0.57 | 0.51 |

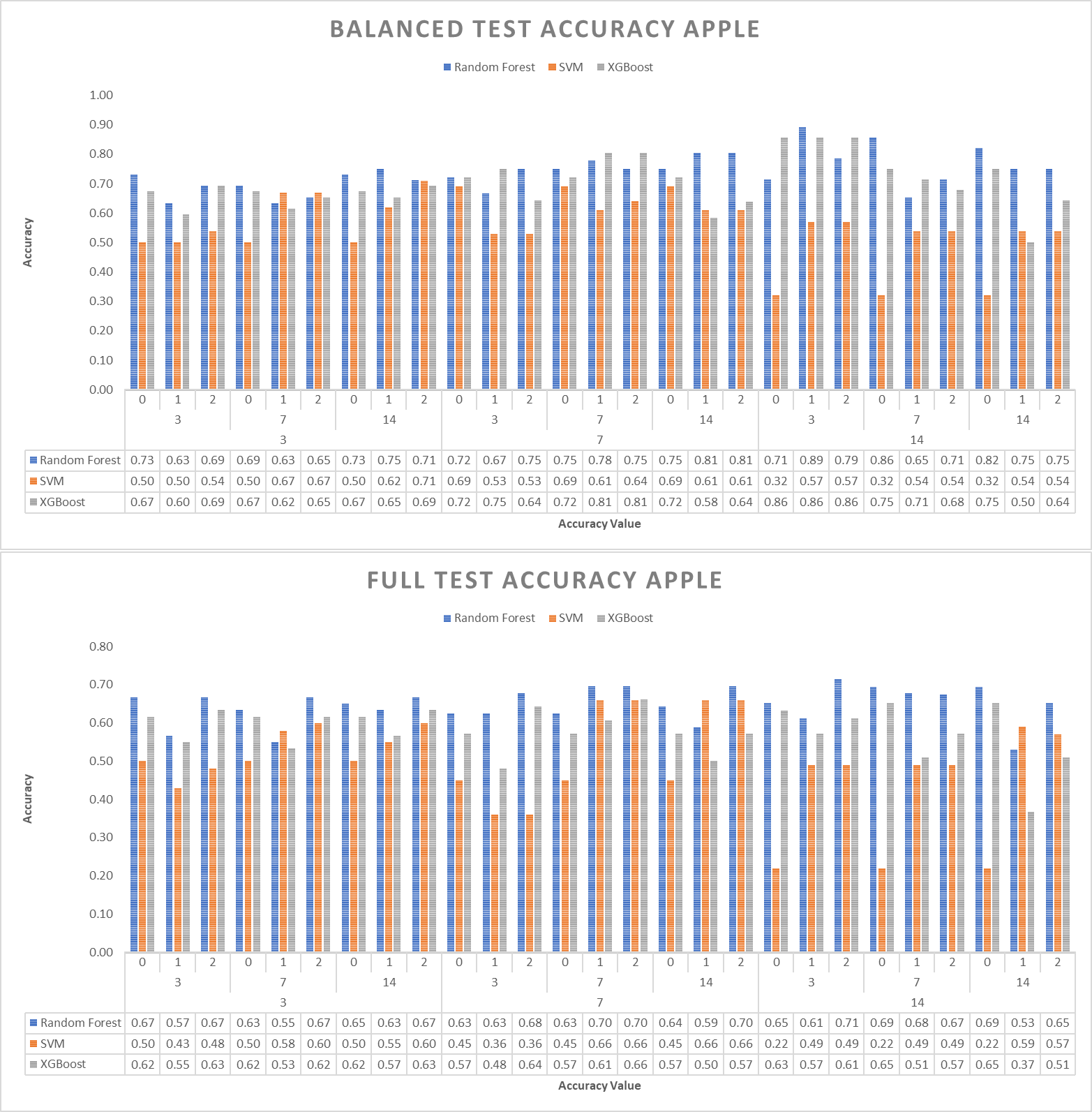
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | Output Size | Window Size | Type Data | Balanced Class Test Accuracy | | |
| Random Forest | SVM | XGBoost |
| GOOGL | 3 | 3 | 0 | 0.60 | 0.50 | 0.56 |
| 1 | 0.69 | 0.46 | 0.58 |
| 2 | 0.67 | 0.50 | 0.69 |
| 7 | 0 | 0.67 | 0.50 | 0.56 |
| 1 | 0.69 | 0.50 | 0.71 |
| 2 | 0.75 | 0.58 | 0.71 |
| 14 | 0 | 0.62 | 0.50 | 0.56 |
| 1 | 0.73 | 0.54 | 0.63 |
| 2 | 0.56 | 0.50 | 0.71 |
| 7 | 3 | 0 | 0.54 | 0.54 | 0.52 |
| 1 | 0.68 | 0.68 | 0.68 |
| 2 | 0.70 | 0.70 | 0.66 |
| 7 | 0 | 0.52 | 0.54 | 0.52 |
| 1 | 0.66 | 0.66 | 0.72 |
| 2 | 0.76 | 0.72 | 0.62 |
| 14 | 0 | 0.68 | 0.54 | 0.52 |
| 1 | 0.74 | 0.58 | 0.56 |
| 2 | 0.66 | 0.60 | 0.62 |
| 14 | 3 | 0 | 0.54 | 0.53 | 0.56 |
| 1 | 0.65 | **0.76** | **0.82** |
| 2 | 0.76 | 0.79 | 0.79 |
| 7 | 0 | 0.56 | 0.53 | 0.53 |
| 1 | **0.97** | 0.50 | 0.85 |
| 2 | 0.79 | 0.50 | **0.85** |
| 14 | 0 | 0.56 | 0.53 | 0.56 |
| 1 | 0.74 | 0.44 | 0.76 |
| 2 | 0.79 | 0.47 | 0.71 |
| Symbol | Output Size | Window Size | Type Data | Full Test Set Accuracy | | |
| Random Forest | SVM | XGBoost |
| GOOGL | 3 | 3 | 0 | 0.65 | 0.57 | 0.48 |
| 1 | 0.60 | 0.48 | 0.58 |
| 2 | 0.65 | 0.55 | 0.65 |
| 7 | 0 | 0.58 | 0.52 | 0.48 |
| 1 | 0.63 | 0.63 | 0.62 |
| 2 | 0.67 | 0.67 | 0.67 |
| 14 | 0 | 0.53 | 0.57 | 0.48 |
| 1 | 0.63 | 0.47 | 0.63 |
| 2 | 0.65 | 0.55 | 0.65 |
| 7 | 3 | 0 | 0.48 | 0.48 | 0.46 |
| 1 | 0.66 | 0.63 | 0.59 |
| 2 | 0.68 | 0.62 | 0.59 |
| 7 | 0 | 0.59 | 0.48 | 0.46 |
| 1 | 0.70 | 0.70 | 0.64 |
| 2 | 0.68 | **0.73** | 0.57 |
| 14 | 0 | 0.61 | 0.48 | 0.46 |
| 1 | 0.64 | 0.54 | 0.59 |
| 2 | 0.66 | 0.54 | 0.63 |
| 14 | 3 | 0 | 0.39 | 0.37 | 0.39 |
| 1 | 0.63 | 0.61 | 0.59 |
| 2 | 0.61 | 0.55 | 0.57 |
| 7 | 0 | 0.39 | 0.37 | 0.37 |
| 1 | **0.73** | 0.65 | **0.61** |
| 2 | 0.61 | 0.65 | 0.61 |
| 14 | 0 | 0.39 | 0.37 | 0.39 |
| 1 | 0.67 | 0.47 | 0.53 |
| 2 | 0.65 | 0.45 | 0.59 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | Output Size | Window Size | Type Data | Balanced Class Test Accuracy | | |
| Random Forest | SVM | XGBoost |
| AMZN | 3 | 3 | 0 | 0.67 | 0.51 | 0.63 |
| 1 | 0.59 | 0.56 | 0.54 |
| 2 | 0.63 | 0.54 | 0.69 |
| 7 | 0 | 0.61 | 0.52 | 0.63 |
| 1 | 0.65 | 0.48 | 0.67 |
| 2 | 0.65 | 0.54 | 0.69 |
| 14 | 0 | 0.65 | 0.52 | 0.63 |
| 1 | 0.63 | 0.59 | 0.67 |
| 2 | 0.63 | 0.57 | 0.72 |
| 7 | 3 | 0 | 0.65 | 0.63 | 0.69 |
| 1 | 0.65 | 0.56 | 0.56 |
| 2 | 0.67 | 0.54 | 0.50 |
| 7 | 0 | 0.57 | 0.63 | 0.52 |
| 1 | 0.67 | 0.50 | 0.59 |
| 2 | 0.72 | 0.52 | 0.61 |
| 14 | 0 | 0.63 | 0.63 | 0.52 |
| 1 | 0.52 | 0.74 | 0.67 |
| 2 | 0.70 | 0.72 | 0.67 |
| 14 | 3 | 0 | 0.82 | 0.50 | 0.82 |
| 1 | 0.82 | 0.53 | 0.79 |
| 2 | 0.76 | 0.53 | 0.68 |
| 7 | 0 | 0.76 | 0.50 | 0.76 |
| 1 | 0.74 | **0.82** | 0.66 |
| 2 | 0.79 | 0.79 | 0.84 |
| 14 | 0 | 0.84 | 0.50 | 0.82 |
| 1 | 0.82 | 0.71 | 0.87 |
| 2 | **0.89** | 0.68 | **0.89** |
| Symbol | Output Size | Window Size | Type Data | Full Test Set Accuracy | | |
| Random Forest | SVM | XGBoost |
| AMZN | 3 | 3 | 0 | 0.67 | 0.56 | 0.65 |
| 1 | 0.63 | 0.60 | 0.58 |
| 2 | 0.65 | 0.58 | 0.70 |
| 7 | 0 | 0.62 | 0.57 | 0.65 |
| 1 | 0.63 | 0.52 | 0.67 |
| 2 | 0.52 | 0.55 | 0.70 |
| 14 | 0 | 0.63 | 0.57 | 0.65 |
| 1 | 0.67 | 0.53 | 0.70 |
| 2 | 0.67 | 0.53 | 0.75 |
| 7 | 3 | 0 | 0.63 | 0.63 | 0.68 |
| 1 | 0.64 | 0.55 | 0.57 |
| 2 | 0.66 | 0.54 | 0.64 |
| 7 | 0 | 0.59 | 0.63 | 0.50 |
| 1 | 0.66 | 0.48 | 0.59 |
| 2 | 0.73 | 0.50 | 0.63 |
| 14 | 0 | 0.61 | 0.63 | 0.50 |
| 1 | 0.68 | 0.75 | 0.64 |
| 2 | 0.68 | 0.73 | 0.66 |
| 14 | 3 | 0 | 0.63 | 0.39 | 0.63 |
| 1 | 0.84 | 0.63 | 0.82 |
| 2 | 0.82 | 0.63 | 0.73 |
| 7 | 0 | 0.82 | 0.39 | 0.68 |
| 1 | 0.80 | **0.86** | 0.67 |
| 2 | **0.84** | **0.86** | 0.78 |
| 14 | 0 | 0.67 | 0.39 | 0.63 |
| 1 | 0.82 | 0.82 | **0.76** |
| 2 | **0.92** | 0.82 | **0.71** |

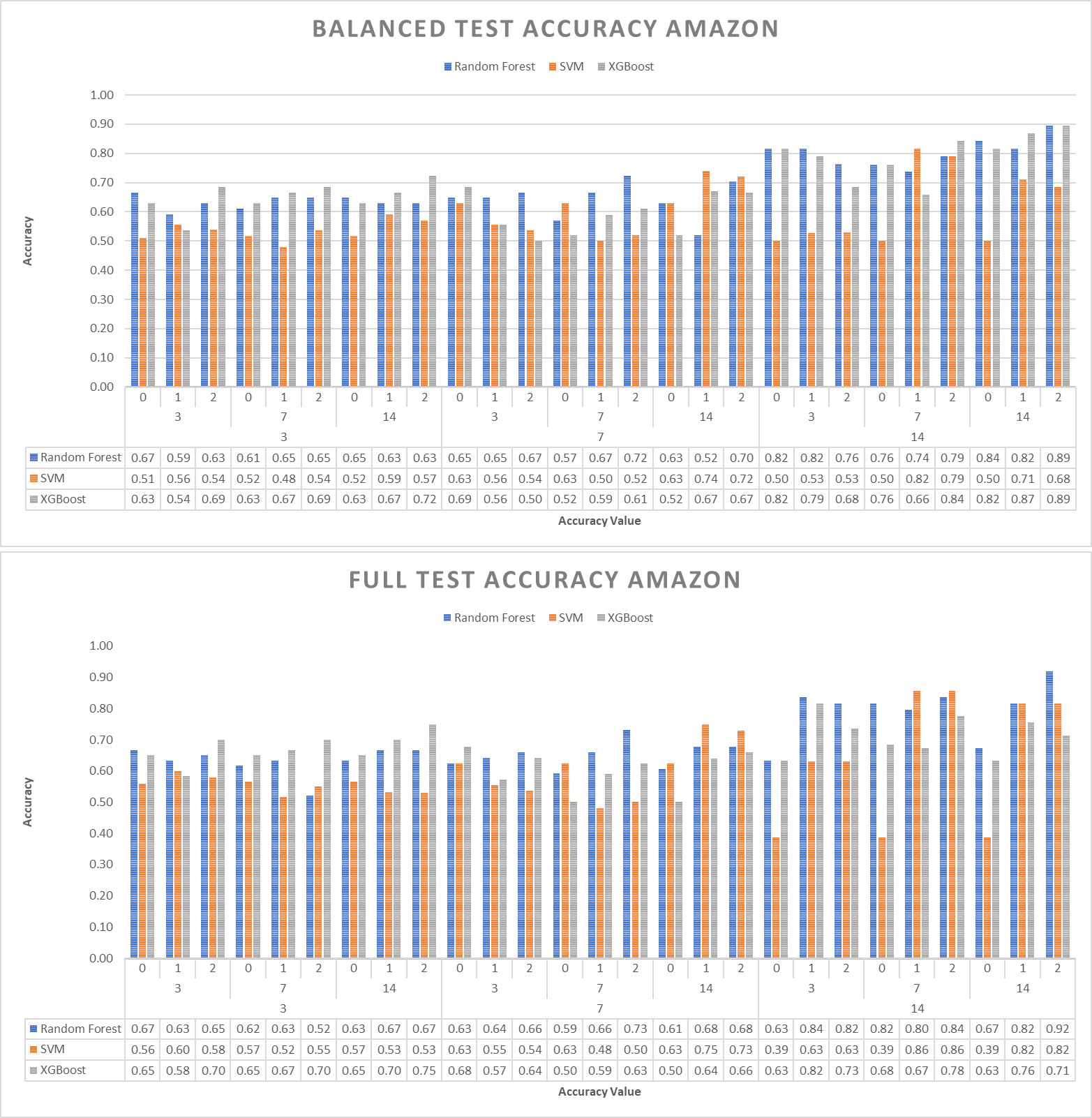
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | Output Size | Window Size | Type Data | Balanced Class Test Accuracy | | |
| Random Forest | SVM | XGBoost |
| MSFT | 3 | 3 | 0 | 0.48 | 0.46 | 0.60 |
| 1 | 0.52 | 0.60 | 0.56 |
| 2 | 0.52 | 0.62 | 0.44 |
| 7 | 0 | 0.48 | 0.46 | 0.60 |
| 1 | 0.66 | 0.50 | 0.68 |
| 2 | 0.52 | 0.46 | 0.46 |
| 14 | 0 | 0.48 | 0.46 | 0.60 |
| 1 | 0.66 | 0.57 | 0.66 |
| 2 | 0.44 | 0.50 | 0.52 |
| 7 | 3 | 0 | **0.86** | 0.61 | 0.78 |
| 1 | 0.78 | 0.47 | 0.72 |
| 2 | 0.64 | 0.47 | 0.50 |
| 7 | 0 | **0.86** | 0.56 | **0.78** |
| 1 | 0.75 | 0.61 | **0.78** |
| 2 | 0.64 | 0.75 | 0.67 |
| 14 | 0 | **0.86** | 0.61 | **0.78** |
| 1 | 0.78 | 0.72 | 0.61 |
| 2 | 0.64 | 0.72 | 0.64 |
| 14 | 3 | 0 | 0.75 | **0.84** | 0.66 |
| 1 | 0.66 | 0.16 | **0.78** |
| 2 | 0.69 | 0.16 | 0.59 |
| 7 | 0 | 0.75 | **0.84** | 0.66 |
| 1 | 0.69 | 0.66 | 0.47 |
| 2 | 0.53 | 0.62 | 0.44 |
| 14 | 0 | 0.75 | **0.84** | 0.66 |
| 1 | 0.41 | 0.50 | 0.44 |
| 2 | 0.47 | 0.53 | 0.31 |
| Symbol | Output Size | Window Size | Type Data | Full Test Set Accuracy | | |
| Random Forest | SVM | XGBoost |
| MSFT | 3 | 3 | 0 | 0.47 | 0.43 | 0.47 |
| 1 | 0.52 | 0.62 | 0.57 |
| 2 | 0.58 | 0.57 | 0.52 |
| 7 | 0 | 0.47 | 0.43 | 0.47 |
| 1 | 0.67 | 0.55 | 0.63 |
| 2 | 0.60 | 0.57 | 0.50 |
| 14 | 0 | 0.47 | 0.43 | 0.47 |
| 1 | 0.53 | 0.46 | 0.55 |
| 2 | 0.47 | 0.57 | 0.53 |
| 7 | 3 | 0 | 0.54 | 0.42 | 0.57 |
| 1 | 0.55 | 0.38 | 0.55 |
| 2 | 0.64 | 0.36 | 0.59 |
| 7 | 0 | 0.54 | 0.61 | 0.57 |
| 1 | 0.61 | 0.48 | 0.61 |
| 2 | 0.59 | 0.57 | **0.68** |
| 14 | 0 | 0.54 | 0.61 | 0.57 |
| 1 | 0.52 | 0.46 | 0.43 |
| 2 | 0.59 | 0.48 | 0.59 |
| 14 | 3 | 0 | **0.71** | **0.73** | 0.65 |
| 1 | 0.57 | 0.24 | 0.65 |
| 2 | 0.49 | 0.22 | 0.43 |
| 7 | 0 | **0.71** | **0.61** | 0.65 |
| 1 | 0.59 | 0.45 | 0.37 |
| 2 | 0.39 | 0.47 | 0.33 |
| 14 | 0 | **0.71** | **0.71** | 0.65 |
| 1 | 0.39 | 0.43 | 0.41 |
| 2 | 0.37 | 0.39 | 0.27 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | Output Size | Window Size | Type Data | Balanced Class Test Accuracy | | |
| Random Forest | SVM | XGBoost |
| TSLA | 3 | 3 | 0 | 0.53 | 0.50 | 0.53 |
| 1 | 0.53 | 0.59 | 0.60 |
| 2 | 0.53 | 0.53 | 0.52 |
| 7 | 0 | 0.53 | 0.50 | 0.53 |
| 1 | 0.62 | 0.53 | 0.55 |
| 2 | 0.55 | 0.66 | 0.52 |
| 14 | 0 | 0.53 | 0.50 | 0.53 |
| 1 | 0.53 | 0.60 | 0.50 |
| 2 | 0.59 | 0.53 | 0.48 |
| 7 | 3 | 0 | 0.36 | 0.70 | 0.43 |
| 1 | 0.55 | 0.50 | 0.59 |
| 2 | 0.43 | 0.52 | 0.36 |
| 7 | 0 | 0.36 | 0.64 | 0.43 |
| 1 | 0.55 | 0.64 | **0.70** |
| 2 | 0.45 | 0.64 | 0.45 |
| 14 | 0 | 0.36 | 0.70 | 0.43 |
| 1 | 0.55 | 0.70 | 0.57 |
| 2 | 0.57 | **0.75** | 0.55 |
| 14 | 3 | 0 | 0.58 | 0.45 | 0.50 |
| 1 | 0.53 | 0.47 | 0.55 |
| 2 | 0.58 | 0.47 | 0.61 |
| 7 | 0 | 0.58 | 0.45 | 0.50 |
| 1 | 0.58 | 0.58 | 0.61 |
| 2 | **0.63** | 0.63 | 0.66 |
| 14 | 0 | 0.58 | 0.45 | 0.50 |
| 1 | 0.61 | 0.63 | 0.42 |
| 2 | 0.61 | 0.61 | 0.55 |
| Symbol | Output Size | Window Size | Type Data | Full Test Set Accuracy | | |
| Random Forest | SVM | XGBoost |
| TSLA | 3 | 3 | 0 | 0.55 | 0.52 | 0.53 |
| 1 | 0.55 | 0.57 | 0.60 |
| 2 | 0.53 | 0.53 | 0.53 |
| 7 | 0 | 0.55 | 0.50 | 0.53 |
| 1 | 0.63 | 0.52 | 0.57 |
| 2 | 0.57 | 0.63 | 0.52 |
| 14 | 0 | 0.55 | 0.52 | 0.53 |
| 1 | 0.55 | 0.62 | 0.52 |
| 2 | 0.58 | 0.58 | 0.50 |
| 7 | 3 | 0 | 0.36 | 0.61 | 0.39 |
| 1 | 0.46 | 0.46 | 0.48 |
| 2 | 0.38 | 0.41 | 0.32 |
| 7 | 0 | 0.36 | 0.57 | 0.39 |
| 1 | 0.46 | 0.50 | 0.61 |
| 2 | 0.43 | 0.52 | 0.43 |
| 14 | 0 | 0.36 | 0.61 | 0.39 |
| 1 | 0.70 | 0.63 | 0.64 |
| 2 | 0.54 | **0.68** | 0.50 |
| 14 | 3 | 0 | 0.45 | 0.43 | 0.39 |
| 1 | 0.39 | 0.37 | 0.43 |
| 2 | 0.67 | 0.35 | **0.69** |
| 7 | 0 | 0.45 | 0.37 | 0.39 |
| 1 | 0.43 | 0.45 | 0.47 |
| 2 | **0.71** | 0.47 | 0.69 |
| 14 | 0 | 0.45 | 0.43 | 0.39 |
| 1 | 0.49 | 0.53 | 0.43 |
| 2 | 0.69 | 0.51 | 0.65 |

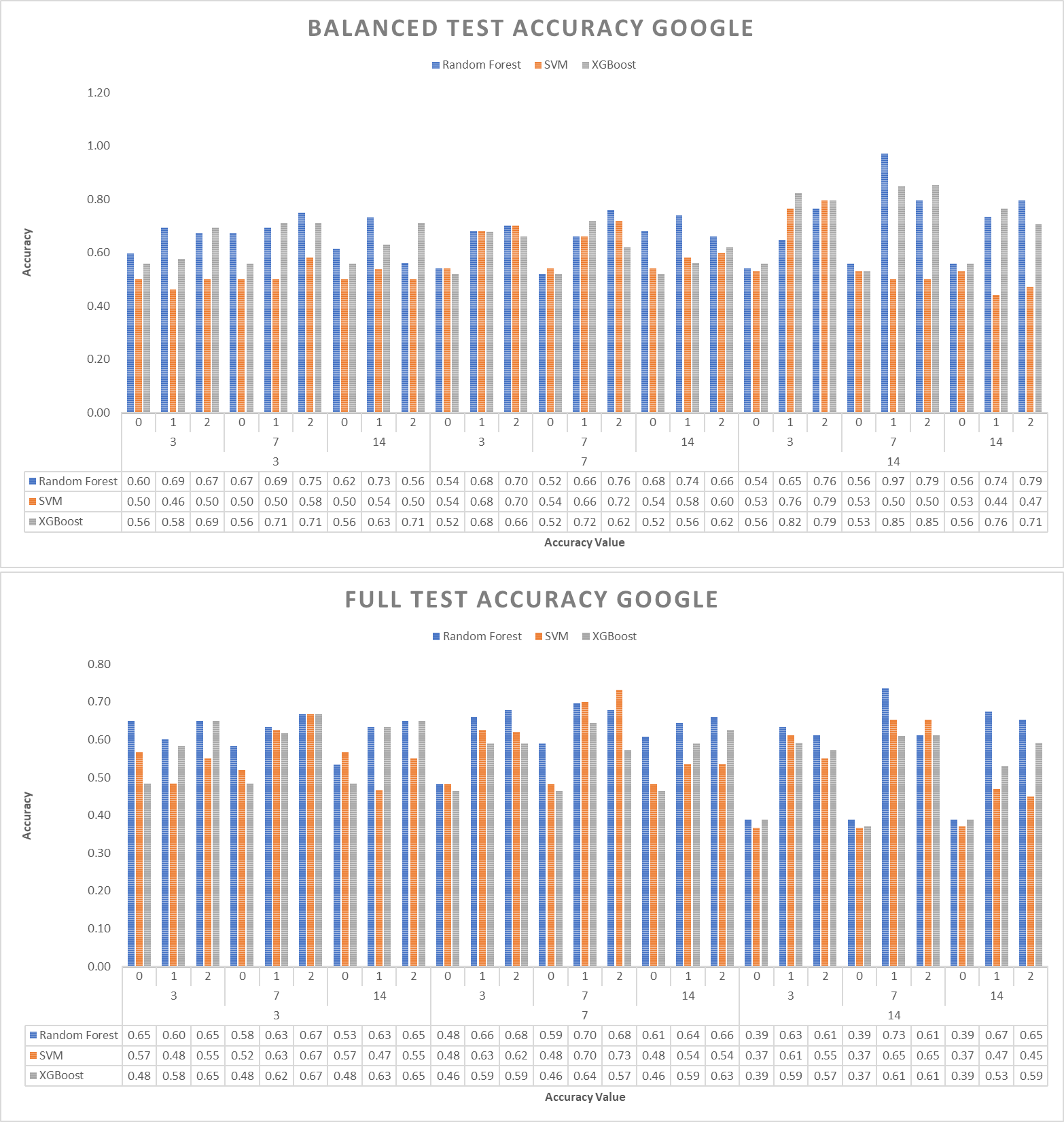
* + 1. **Visualize Prediction Accuracy**
       - 1. **Apple**

****

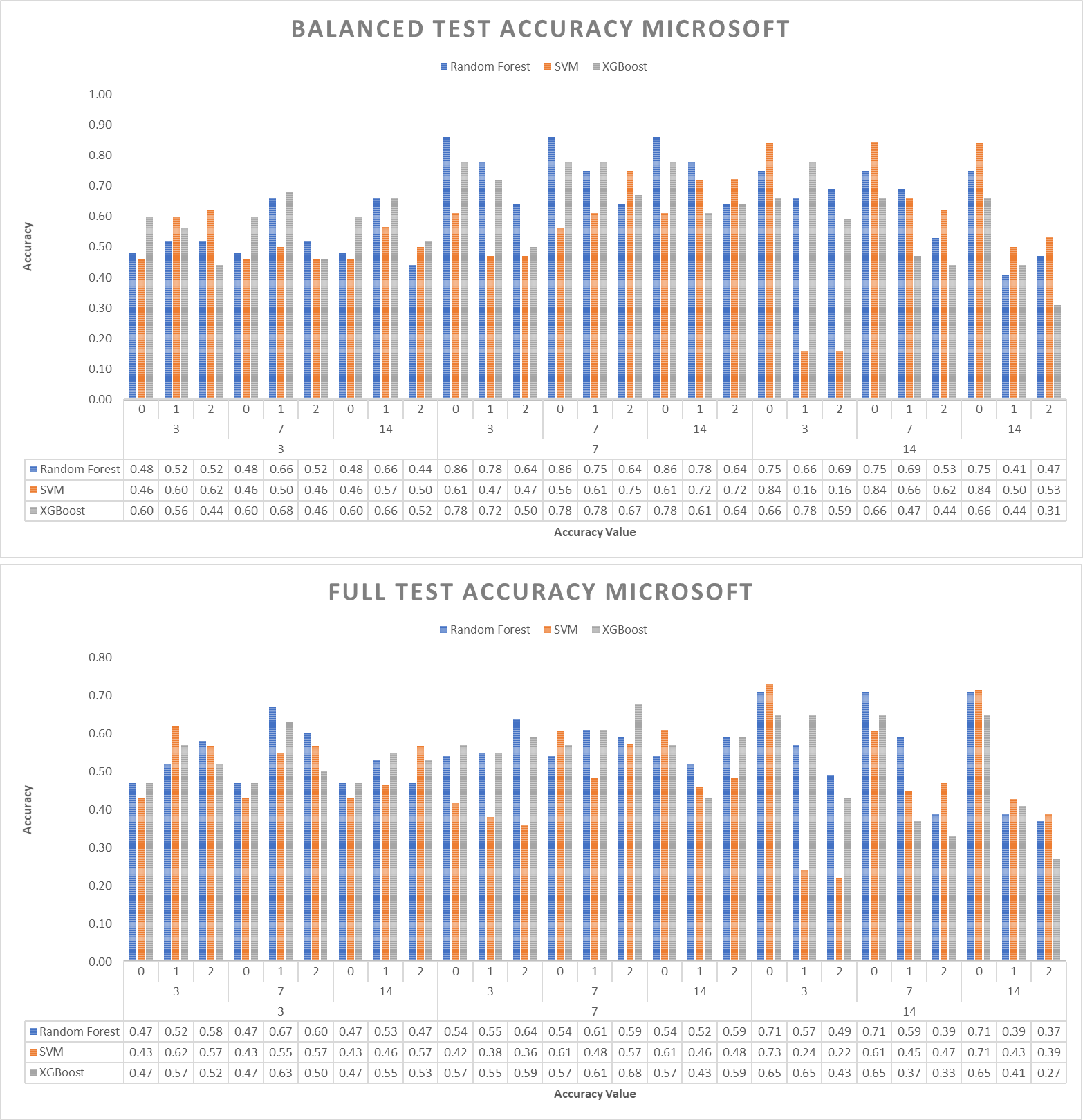
* + - * 1. **Amazon**



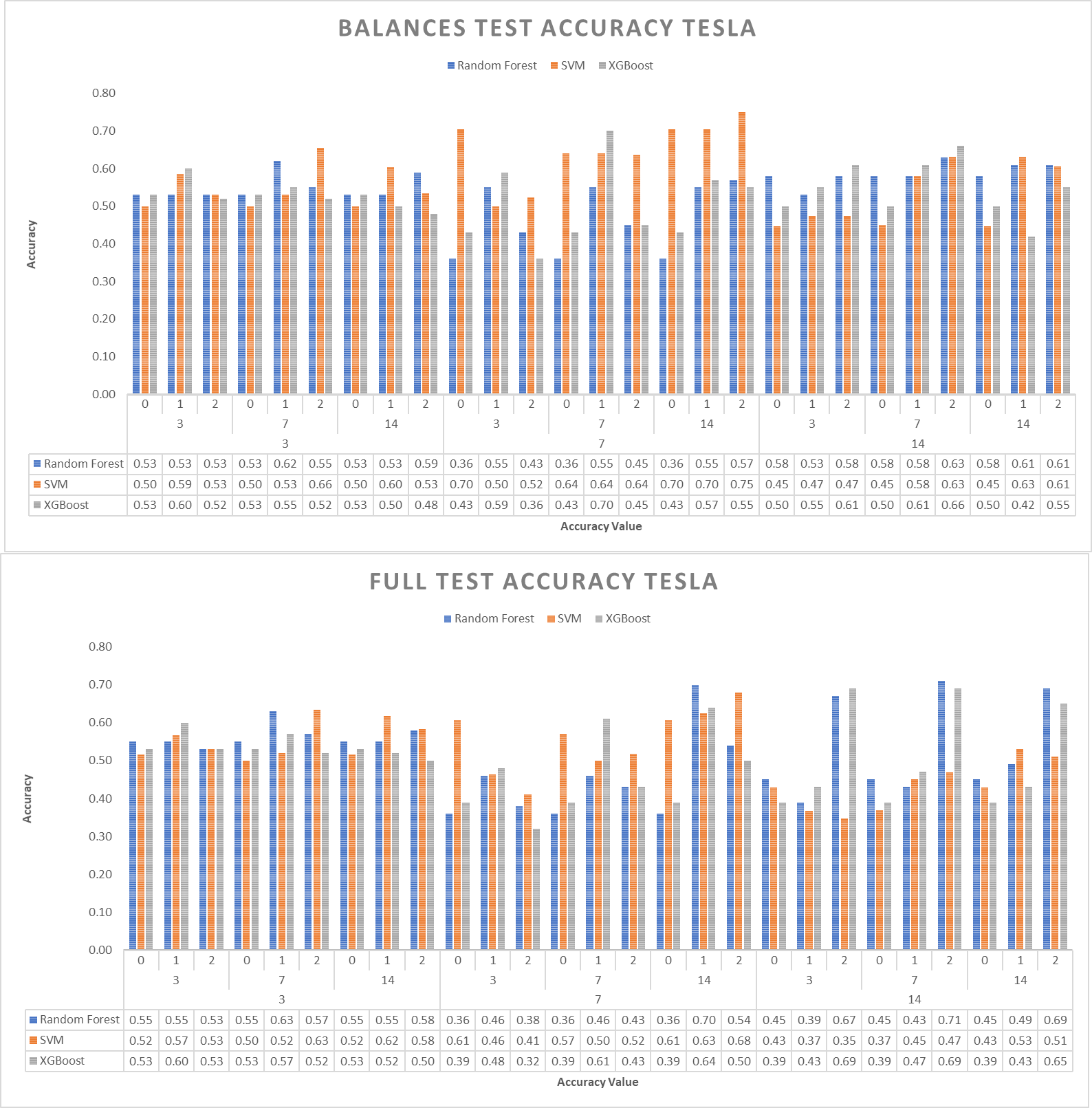
* + - * 1. **Google**

****

* + - * 1. **Microsoft**

****

* + - * 1. **Tesla**

****

# **Chapter 09: Conclusion**

Our analysis demonstrates that utilizing technical indicators enables us to make predictions about the trending of the stock market. The results indicate a 61% accuracy rate on the entire dataset for predicting the market trends over a 7-day period. The achieved accuracy rate of 61% on the entire dataset for the next 7 days suggests that the selected technical indicators are indeed informative and useful for predicting stock market trends. While it's important to note that prediction accuracy may vary across different time periods and market conditions, this result provides a promising foundation for further research and application.

Using technical indicators as a predictive tool offers advantages in terms of objectivity and data-driven analysis. By relying on quantitative metrics derived from historical market data, we can reduce reliance on subjective judgments and potentially improve decision-making processes.

In conclusion, our report has presented an innovative approach to stock market analysis by combining the power of language modeling with advanced predictive modeling techniques. By utilizing 4 different models: Long-Short Term Memory, Random Forrest, Support Vector Machine and an assembled model that combines BERT for article summarization and Transformer models for trend prediction, we have aimed to enhance the accuracy and reliability of stock market predictions.

LSTM, a type of recurrent neural network, has demonstrated its capability to capture temporal dependencies and patterns in time series data. By utilizing LSTM, we aimed to leverage its sequential modeling abilities to predict stock market prices. The LSTM model has shown promise in capturing the complex dynamics of the stock market, making it a valuable tool for forecasting future prices.

SVM, on the other hand, is a supervised learning algorithm that excels in classification tasks. We employed SVM to classify stock market trends based on input features such as close price, technical indicators and text summarize articles. The SVM model has proven to be effective in separating different classes of stock market trends, allowing us to make accurate predictions and decisions based on the classified trends.

Random Forest, a popular ensemble learning technique, combines multiple decision trees to improve prediction accuracy. By utilizing Random Forest, we aimed to leverage its ability to handle high-dimensional data and capture complex interactions between features. The Random Forest model has shown robustness and resilience to noise in the data, making it a reliable choice for stock market trend prediction.

Throughout our analysis, we observed that each model had its strengths and limitations. LSTM excelled in capturing temporal patterns but required extensive training and tuning. SVM did not perform well in classifying stock market trends, but its performance depended on the selection of appropriate input features. Random Forest exhibited good generalization capabilities but could be sensitive to overfitting in certain scenarios.

Through the integration of BERT, we have effectively summarized news articles from a Web API, providing valuable insights into the market sentiment and factors influencing stock movements. The BERT model's ability to distill information from large volumes of text has enabled us to obtain concise summaries that are relevant to the stock market domain.

Additionally, by incorporating Transformer models and leveraging time series data, we have developed robust predictive models capable of forecasting stock market trends for the next N days. The inclusion of close price data and technical indicators in our dataset has enriched the models' predictive capabilities, allowing for more accurate trend predictions.

# **References**

|  |  |
| --- | --- |
| [1] | A. Ghosh , S. Bose , G. Maji and N. C. Debnath , "researchgate.net," September 2019. [Online]. Available: https://www.researchgate.net/publication/346450872\_Stock\_Price\_Prediction\_Using\_LSTM\_on\_Indian\_Share\_Market. |
| [2] | S. Albahli, T. Nazir, A. Mehmood and A. Irtaza, "researchgate.net," February 2022. [Online]. Available: https://www.researchgate.net/publication/358669067\_AEI-DNET\_A\_Novel\_DenseNet\_Model\_with\_an\_Autoencoder\_for\_the\_Stock\_Market\_Predictions\_Using\_Stock\_Technical\_Indicators. |
| [3] | R. Xiao, Y. Feng, L. Yan and Y. Ma, "arxiv.org/," 31 August 2022. [Online]. Available: https://arxiv.org/abs/2209.02407. |
| [4] | B. Zhao, H. Lu, . S. Chen, J. Liu and D. Wu, "ieeexplore.ieee.org," February 2017. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7870510. |
| [5] | S. Bai, J. Z. Kolter and V. Koltun , "arxiv.org," 19 April 2018. [Online]. Available: https://arxiv.org/abs/1803.01271. |
| [6] | C. MITCHELL, "investopedia.com," 31 August 2021. [Online]. Available: https://www.investopedia.com/terms/w/williamsr.asp. |
| [7] | C. MITCHELL, "investopedia.com," 25 May 2022. [Online]. Available: https://www.investopedia.com/terms/c/commoditychannelindex.asp. |
| [8] | B. Dolan, "investopedia.com," 15 March 2023. [Online]. Available: https://www.investopedia.com/terms/m/macd.asp. |
| [9] | C. Mitchell, "investopedia.com," 19 September 2021. [Online]. Available: https://www.investopedia.com/terms/d/dmi.asp. |
| [10] | tradingtechnologies, "tradingtechnologies.com," [Online]. Available: https://library.tradingtechnologies.com/trade/chrt-ti-chande-forecast-oscillator.html. |
| [11] | A. Hayes, "investopedia.com," 30 June 2022. [Online]. Available: https://www.investopedia.com/terms/c/chandemomentumoscillator.asp. |
| [12] | W. Kenton, "investopidia.com," 7 May 2021. [Online]. Available: https://www.investopedia.com/terms/e/efficiencyratio.asp. |
| [13] | C. Mitchell, "investopedia.com," 1 January 2021. [Online]. Available: https://www.investopedia.com/terms/p/pricerateofchange.asp#:~:text=The%20Price%20Rate%20of%20Change%20(ROC)%20oscillator%20is%20an%20unbounded,ROC%20will%20hover%20near%20zero.. |
| [14] | B. Twomey, "investopedia.com," 15 March 2022. [Online]. Available: https://www.investopedia.com/articles/forex/10/schaff-trend-cycle-indicator.asp. |
| [15] | C. Mitchell, "investopedia.com," 25 October 2021. [Online]. Available: https://www.investopedia.com/terms/e/elderray.asp#:~:text=The%20Elder%2DRay%20Index%20is,exponential%20moving%20average%20(EMA).. |
| [16] | fidelity, "fidelity," January 2023. [Online]. Available: https://www.fidelity.com/viewpoints/active-investor/understanding-bollinger-bands#:~:text=Bollinger%20Bands%20are%20a%20short,lower%20band%20around%20the%20price.. |
| [17] | A. Hayes, "investopedia.com," 1 February 2022. [Online]. Available: https://www.investopedia.com/terms/s/sma.asp#:~:text=A%20simple%20moving%20average%20is,weighted%20on%20recent%20price%20action.. |
| [18] | fidelity, "fidelity," [Online]. Available: https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/ema. |
| [19] | J. Fernando, "investopedia.com," 28 March 2023. [Online]. Available: https://www.investopedia.com/terms/v/vwap.asp#:~:text=Understanding%20the%20Volume%2DWeighted%20Average,by%20the%20total%20shares%20traded.. |
| [20] | fidelity, "fidelity.com," [Online]. Available: https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/hull-moving-average#:~:text=The%20Hull%20Moving%20Average%20(HMA,be%20used%20to%20identify%20trend.. |
| [21] | fidelity, "fidelity.com," [Online]. Available: https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/cmf. |
| [22] | C. Mitchell, "investopedia.com," 29 May 2022. [Online]. Available: https://www.investopedia.com/terms/u/ultimateoscillator.asp. |
| [23] | I. Mukherjee, "stockmaniacs.net," 5 May 2023. [Online]. Available: https://www.stockmaniacs.net/linear-regression-slope-indicator/#:~:text=How%20to%20Calculate%20Linear%20Regression,a'%20is%20the%20intercept%20point.. |
| [24] | A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez and Ł. Kaiser, "arxiv.org," 6 December 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf. |
| [25] | J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "arxiv.org," 24 May 2019. [Online]. Available: https://arxiv.org/abs/1810.04805. |
| [26] | D. T. Araci, "arxiv.org," 25 June 2019. [Online]. Available: https://arxiv.org/pdf/1908.10063.pdf. |