Mini Project

ON

STOCK MARKET PIPELINE AND PREDICTION

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

in

Computer Science and Business Systems

by

K. Meghana	**22241A3223**
K. Shreshta	**22241A3226**
N. Navaneetha	**22241A3238**

Under the Esteemed guidance of

Mr. P. Aditya Sharma

Assistant Professor

Department of Data Science

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Autonomous under JNTUH, Hyderabad)

Bachupally, Kukatpally, Hyderabad-500090

2024-2025

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

Hyderabad-500090

CERTIFICATE

This is to certify that the major project entitled "**STOCK MARKET PIPELINE AND PREDICTION**" is submitted by **K. Meghana (22241A3223), K. Shreshta (22241A3226) and N.Navaneetha (22241A3238)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Business Systems during Academic year 2024-2025.

Internal Guide

Mr. P. Aditya Sharma

Head of the Department

Dr. S. Govinda Rao

External Examiner

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **Mr.P. Aditya Sharma, Assistant Professor,** Department of Computer Science and Business System, for his support in the completion of our dissertation. We are thankful to our major project coordinators **Mr. P. Aditya Sharma,** Assistant Professor and **Ms. Vemulapalli. Manasa**, Assistant Professor, for their valuable suggestions and comments during this project period.

We wish to express our sincere thanks to **Dr. S. Govinda Rao,** Head of the Department**,** and to our principal **Dr. J. PRAVEEN,** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

K. Meghana (22241A3223)

K. Shreshta (22241A3226)

N. Navaneetha (22241A3238)

DECLARATION

We hereby declare that the major project titled **"STOCK MARKET PIPELINE AND PREDICTION"** is the work done during the period from **20 August 2024 to 18 December 2024** and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Business Systems from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad). The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

K. Meghana (22241A3223)

K. Shreshta (22241A3226)

N. Navaneetha (22241A3238)

ABSTRACT

The Stock Market Pipeline and Prediction harness the power of Azure Machine Learning and base it on cloud-based services when creating an end-to-end fully predictive pipeline for stock prices. It is an innovative solution which gathers multiple datasets of small-cap, mid-cap, and large-cap stock data from cloud-based services through the Kaggle platform. The project offers efficient handling and preprocessing of financial data in large volumes by using Azure Data Factory for data ingestion and centralizing data management through Azure Blob Storage.The predictive model, trained using Azure Machine Learning and based upon an algorithm of Linear Regression, gave very accurate predictions of the actual stock prices. The model uses a fully automated pipeline of train model, test model, deploy model, all through the service of Kubernetes and the endpoint of Batch. These all can scale and make requests in real-time. performance metrics including Mean Squared Error and Mean Absolute Percentage Error, the later with precision up to 96.43%.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AKS Azure Kubernetes Service

ADF Azure data factory

ETL Extract, Transform, Load

AzureML: Azure Machine Learning

API: Application Programming Interface

MA: Moving Average

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Introduction to the Project

The stock market is a dynamic and complex system that needs perfect tools and strategies for analyses and predictions. The best way to empower investors as well as financial analysts in better decision-making, reducing risks, and increasing profits is accurate forecasting of the stock price. In recent years, the integration of cloud computing with machine learning revolutionized this field of accurate prediction by allowing the proper processing, storage, and use of data in building better predictive models.

This project uses combined cloud infrastructure with the abilities of machine learning in the use of Azure to forecast a whole pipeline for the determination of the prices of stocks on the stock market. Practical applications of these solutions, which happen to be cloud-based, utilize datasets from the Kaggle on small-cap, mid-cap, and large-cap stocks. This process involves the ingestion of data and will include the use of Azure Blob Storage, development of a model, its implementation through Azure Machine Learning, followed by the application of linear regression to accomplish predictive analytics.

The project aims at simplifying the process of predicting stock prices using insightful metrics, visualizations, and next-day price forecasts. This document describes the methodologies, tools, and outcomes of the project and highlights its potential to simplify and enhance decision-making in the financial domain.

1.2 Objective of the project

The project will work on designing a pipeline that develops a comprehensive, efficient, and scalable system to forecast the prices of stocks in the market. Well-defined objectives have been developed with an understanding of the challenges presented in predicting the stock market, such as large and diverse datasets to ensure scalability, and finally to present actionable insights to investors and analysts.

Scalable Design and Implementation of a Data Pipeline

Data Heterogeneity: Develop an integrated data pipeline that can handle various kinds of stock data small-cap, mid-cap, and large-cap so that it captures all aspects of the market.

Cloud Integration: Implement Azure Data Factory to streamline data ingestions and preprocessions easily so that handling raw data becomes easy.

Centralized Storage: Azure Blob Storage as a safe and massively scalable repository for storing the preprocessed datasets will ensure efficient data management and accessibility.

Building the Machine Learning Model

Predictive Modeling: Azure Machine Learning: Building appropriate models on Azure should be able to predict stock price on the basis of historical data.

Model Selection: Emphasize Linear Regression- it is one of the traditional, interpretable models which may ensure very reliable prediction with little chance of overfitting, which often arises with more complex deep learning techniques.

Feature Engineering: Include enough feature engineering with a look-back period

Automation and Deployment

End-to-End Automation: Develop an automated pipeline in Azure Machine Learning to automate data pre-processing, model training, model evaluation, and result generation.

Scalable Deployment: The models are now ready for deployment using Kubernetes, and the system is ready for live or batch predictions for efficiency. Batch Endpoints ensure consistent output for users.

Analysis and Visualization.

Performance Metrics :Use Key Performance Metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error to predict the performance evaluation of the models.

Create the plot of Actual vs Predicted of all models to show visually exactly how a model behaves by generating such plots. Concentrate on next day prediction of stock prices and thus allow near future investor decision-making and hence will increase practicality of a system

Scalable Data Pipeline Design and Implementation

Data Diversity: Develop a good data pipeline that processes and deals well with data diversity in stock datasets of small-cap, mid-cap, and large-cap stocks and ensures adequate coverage of any market segment.

Cloud Integration: Use Azure Data Factory to ingest and preprocess raw data by hand.

Centralized Storage: Use Azure Blob storage as a safe, scalable source of preprocessed datasets ensuring effective data management, accessibility, and security.

Machine Learning Model Development

Predictive Modeling: Develop machine learning models using Azure Machine Learning with historical data to predict stock prices.

Model Selection: Highlight Linear Regression as the model under consideration; it is an older and more interpretable model which promises to give good predictions with less chances of overfitting, common in deeper models.

Feature Engineering: Apply good feature engineering practices like a look-back period that helps capture all relevant trends and patterns in the stock data.

Automation and Deployment

Scalable Deployment: The trained models should be deployed on Kubernetes. In this way, the system will be able to take real-time or batch predictions as efficiently as possible. Use of Batch Endpoints will ensure safe accessibility for all those who need constant output.

Analysis and Visualization

1.3 Methodology adopted to satisfy the objective

For achieving the objectives of the Stock Market Pipeline and Prediction project, innovative cloud-based technologies, machine learning techniques, and systematic data handling practices were adapted. The elaboration of methodologies applied to every objective is described below in detail:

1.3.1 Data Pipeline Design and Implementation

Data Sourced

The source of historical stock data is Kaggle, covering a mix of small-cap, mid-cap, and large-cap stocks. This diversity ensures that the analysis done here would be pretty diverse.

All datasets included essential features: Open, Close, High, Low, Adjusted Close, and Volume, to ensure comprehensive data analysis.

Data Ingestion

Data ingestion is automated by Azure Data Factory, thereby enabling smooth integration of raw datasets into the pipeline.

Pre-processing checks: Data was clean, consistent, and in the right format.

Data Storage

Central repository used, which is Azure Blob Storage. It allows scalability and security for preprocessed datasets.

Folder structure organizes datasets to separate them by stock-small-cap, mid-cap, large-cap categories

1.3.2. Development of Machine Learning Models

Model Selection

Linear Regression was opted since it is easy to interpret and also excellent for time-series prediction tasks like predicting the price of stocks.

Feature Engineering

The look-back period would be taken into account while incorporating it to determine the effect of the last days' closing prices on predicting it, assuming the number of days before closing

Moving averages and the daily price differences were computed for better accuracy

Training and Evaluation of Model

Azure Machine Learning Notebooks were used to train the regression model.

Metrics like Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error were acquired in order to determine the performance of the model.

1.3.3. Automation and Deployment

The pipeline could do all that it is expected to, with virtually no manual interference at all. This made the whole process effective.

The trained model was deployed in AKS hence scalable predictions.

Configuration of Batch Endpoints, ensuring that it runs in high performance when it processes big datasets in batches.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing approaches

In his research paper, Antreas and G. Samakovitis come up with a modern alternative through an event-driven, serverless ETL pipeline on Amazon Web Services as a way of presenting the limitations of the traditional ETL approach. Traditional ETL pipelines have architectures that are monolithic or batch-oriented, which makes them lack scalability and unable to satisfy real-time data processing requirements. While cloud-based ETL solutions like AWS Glue and Azure Data Factory are scalable, they typically rely on manual scheduling or triggers and therefore are not suitable for completely dynamic workflows. Furthermore, message-queue-driven architectures, implemented in Apache Kafka or RabbitMQ, enable event-driven processing but introduce a significant amount of complexity to manage queue systems and downstream data transformations. The modularity and scalability of Docker and Kubernetes still require infrastructure management and hence cannot eliminate operational overhead entirely. This study bridges the gaps by using the serverless technologies of AWS, such as AWS Lambda, Amazon S3, and DynamoDB, to implement an event-driven ETL pipeline that scales automatically while reducing latency without the necessity of infrastructure management. With that, this approach brings in greater scalability, cost-effectiveness, as well as streamlines real-time data workflow integration. With serverless computing, it reveals a great number of practical advantages of serverless computing for modern ETL processes. Of course, among the benefits are those in organization interest in making the leap from the old architectures into much more agile, event-driven, and serverless data engineering solutions.

It describes the progress of predictive modeling in this paper, titled "Stock Price Prediction using Machine Learning and Deep Learning," published on IEEE Xplore in the year 2021, by Pratheeth S. and Vishnu Prasad R.[2], on advancing computational techniques towards developing new predictive models to predict changes in stock prices in stock markets, considering the unpredictability of change in nature and the underlying influence due to multiple reasons. This study underlines the strength of models like linear regression, SVM, and decision trees in the ML framework and establishes superiority over deep learning architectures like RNN and LSTM networks in terms of the capture of temporal dependencies and patterns in financial time series data. The techniques proposed may be employed for better prediction, leading to a more reliable projection. Using historic stock values may enable them to do so and hence assist trade and investors in decision. It is one of the promising ones; the restriction lies in handling variables with respect to macros and how these impact the economy for several economies and data dependence on computations, making this deep learning model less practical in real application. Overall, the study presents some valuable contributions to the use of ML and DL in financial markets.This is one of the most promising advancements in applying machine learning (ML) and deep learning (DL) techniques, especially in financial markets. The most significant drawback that has emerged is in variable handling related to macros, taking into consideration the interaction between macro factors and their broader effect on various economies. The main challenge in practical applications in the real world relates to its computational dependency on large datasets, which make its real-world application hard to apply. However, this paper brings some considerable contributions regarding its potential as ML and DL to revitalize the whole financial markets' analytical world, pointing out to important perspectives that are of

immense interest for new researches and development on such aspects.

The research paper,"cloud -Native ETL:Integration DATAbricks and Azure Data factory for scalable Data Processing in Enterprise Environment" by M.Thoutom[3] .This might leave readers without a sound grasp of how the integration works in practice.Another weakness might be in the lines of technical complexity; it may not be one of the more accessible papers because of the assumption that needs to be made to consider a specific high level of expertise and tools, that of Databricks or even Azure Data Factory for the cloud-native ETL solutions. This would, probably make its utility more shallow for new minds not fully exposed to them, or at least unexposed to the state of art architectures within the clouds.It may also miss doing a deep cost analysis of the paper. The cost of the tools like Databricks and Azure Data Factory are quite high and, in more important ways, escalating exponentially for large enterprise environments. It will also leave short of details regarding control of cost in case these services are implemented: cost models, possible overruns etc.It most likely will discuss scalability, but it won't cover performance concerns in scaling the cloud-native ETL process. For instance, big data processing and storage require a huge amount of memory, leading to data latency, processing delays, and storage bottlenecks, which may affect enterprises significantly. Further elaboration of these challenges and solutions could be beneficial for the audience.If these services are to be implemented, the discussion is likely to fall short of dealing with the critical aspects related to cost control, such as development of comprehensive cost models, the risk of budget overruns, and strategies for controlling the unexpected expenses. If these services are to be implemented, the discussion is likely to fall short of dealing with the critical aspects related to cost control, such as development of comprehensive cost models, the risk of budget overruns, and strategies for controlling the unexpected expenses.

The paper by A. Thota, M. Ganapuram, S. R. Maddineni, and S. Cheemalamarri, titled "A Fast and Scalable ETL Pipeline for Serverless Architectures,"[4] presents an innovative approach to building ETL pipelines using serverless computing. However, the proposed solution has certain limitations.This significant downside is cold start latency within a serverless architecture; initializing functions under low usage has longer periods, hence impeding real-time data processing. Relying on the use of serverless platforms causes more vendor lock-in than what flexibility it brings into, increasing reliance on cloud provider specifics. Serverless systems are excellent for event-driven workloads but can be challenging for resource-intensive tasks because of the constraints on execution time and memory, and this requires complex workarounds such as splitting workflows into smaller functions. The pay-per-use model is cost-effective for small workloads but can lead to unexpectedly high costs for high-frequency or high-volume data pipelines. These limitations make it challenging to achieve careful architectural planning along with optimization in order to address scalability challenges while minimizing cost and maintaining performance uniformity.Serverless systems are very efficient at processing event-driven workloads because of the dynamic allocation of resources as they become available. However, this can be a significant hindrance when applied to resource-intensive workloads. These limitations come from constraints on execution time, memory, and other computational resources, which typically involve complex workarounds. For such large-scale operations, the cumulative cost of repeatedly invoking serverless functions becomes a cause of surprise expense. Besides, managing the complexities of scaling serverless architectures for such scenarios is demanding in terms of careful planning and optimization to balance performance and cost. Addressing these limitations and exploring alternative strategies to enhance the

feasibility of serverless systems for resource-intensive workloads would give very valuable insights for organizations considering adopting at scale.

The paper by A. M. B. Dawood, A. M. Hussein, and M. K. Ibraheem, titled "Design and Implementation of Efficient ETL Pipelines for Incremental Data Loading,"[5] is on the improvement of ETL processes by incorporating efficient techniques for incremental data loading. Although the approach overcomes the problems of processing large datasets by avoiding full data reloads, it has some limitations. This involves complex maintenance in accurate metadata and tracking of changes that becomes challenging when high volatility in data is prevalent or from unstructured sources. Incremental loading can create problems of synchronization with respect to distributed data systems that could lead to possible inconsistency or delay in time. It also necessitates an advanced mechanism of monitoring and error handling in terms of integrity, which again contributes to system overhead in operations. In addition, incremental loading reduces data transfer volumes but may not fully optimize performance for highly dynamic datasets with frequent updates because the overhead of change detection and indexing can offset the benefits. These limitations point to the need for further refinement to handle diverse and complex enterprise data environments effectively.

T.Villica,R.Oliveria and H.Lemecka,"Optimising Microsevies Integrated in ETL pipelines using canonical Data models"[6]One of the major disadvantages is that canonical data models are hard to design and maintain, especially in dynamic environments where the data schemas often change. The rigidity of this model reduces the adaptability of the system to accommodate new data sources or changes in business requirements.Furthermore, this reliance on a centralized data model may introduce performance bottlenecks because all microservices must conform to the standardized schema, which may slow down data processing in distributed architectures. The first deployment of canonical models also requires considerable development effort and coordination across teams, extending project timelines and costs. Canonical data models may also result in over-engineering when simpler, decentralized integration patterns would suffice, thus creating unnecessary complexity. These limitations indicate that while canonical data models can optimize microservices integration, applicability must be carefully evaluated against the specific needs and scalability requirements of the ETL pipeline.This might bring unwanted complexity into the system and may be difficult to manage and maintain. Such challenges indicate that canonical data models, despite some benefits in optimizing the microservices integration process, call for careful assessment regarding the applicability. Organisations should evaluate whether using the canonical data models matches with the specific requirements and objectives of the ETL pipeline in terms of operation goals and scalability demands.

P.Kavya,S.Saagarika,R.T.Subhadarshini,"STOCK MARKET ANALYSIS" [7]This research paper they developed about stock market analysis thier drawbacks are explained This might be a major limitation-the paper might over-simplify the dynamics of the markets, because it relies heavily on historical data and classical statistical models, which can be inadequate to capture non-linear trends or real-time fluctuations in the stock prices. The study may also not be that robust in handling external shocks, such as geopolitical events, changes in economic policies, or market sentiment, which really affect the stock performance. Advanced machine learning techniques or hybrid models might not be included in the study, which may limit the predictive accuracy in highly volatile markets. In

addition, the dataset applied to the analysis may not reflect a variety of stocks or market conditions, thereby limiting the generalizability of the results. Finally, the paper may not evaluate some limitations of the methods developed as it opens its grounds to future developments and applicability for the analysis in financial data that is wider in extent. These are a consequence that requires advanced methodologies coupled with broader datasets so the output can be taken further.Finally, the paper may not evaluate some limitations of the methods developed as it opens its grounds to future developments and applicability for the analysis in financial data that is wider in extent.

A paper by Pratheeth S. and Vishnu Prasad R.,[8] "Stock Price Prediction using Machine Learning and Deep Learning," issued in IEEE Xplore back in 2021, discuss the use of machine learning and deep learning as techniques for stock price forecasts.The study will also face challenges in dealing with noise and volatility in financial data, which may affect accuracy in predictions. Furthermore, deep learning models, although powerful, might need extensive computational resources, along with large datasets, which prove to be a barrier for even smaller organizations or real-time applications. Overfitting becomes a common problem in a deep learning model, especially while dealing with a limited or imbalance dataset, which leads to poor generalization across unseen data. The paper could also lack an overall assessment of the interpretability of the models, which will make it hard for the end-users or investors to understand the reason behind the prediction.The study will also have the problem of noise and volatility, which could affect accuracy in forecasting. Moreover, although powerful, deep learning models require vast computational resources and huge datasets, which act as a hindrance even to the smallest organizations or in real-time applications. A deep learning model suffers from overfitting issues while dealing with a limited or unbalanced dataset, which leads to poor generalization across unseen data.

In this paper titled "Development of an ETL-Pipeline for Automatic Sustainability Data Analysis," [9]authored by Akintayo, A., Ahmed, S., Gana, I., Okwuosa, I., Ugbebor, C., and Ojo, O. and presented in the International Journal of Sustainable Development & World Ecology, in 2024, presents an automated ETL pipeline in order to process sustainability-related data. Although research works show enormous contributions to the field of sustainability analytics, it does have some obvious limitations and drawbacks. It mainly limits the scalability of unstructured and heterogeneous sustainability data handling, potentially leading to data inconsistencies and inaccuracies in extraction and transformation. Moreover, this pipeline will suffer from issues of scalability in real-time applications in dealing with exploding datasets or heterogeneous data formats. High automation reduces human oversight, sometimes resulting in the failure to consider critical nuances pertinent to specific contexts of sustainability metrics.Last but not least, there is no proper mechanism to handle errors and ensure the quality of data, which might affect the reliability of analytical results, making the pipeline less effective in practical use. This drawback suggests that further research is needed to make sustainability-focused ETL pipelines scalable, adaptable, and of better quality in terms of data.Real-time applications, especially in terms of dealing with growing datasets or diverse and heterogeneous data formats, also exacerbate the scalability challenges in processing and integrating sustainability data.Another important issue comes from the very high degree of automation integrated into the pipeline. Though automation does improve efficiency, it diminishes human oversight dramatically, leading sometimes to a failure to consider the most crucial nuances and contextual factors that are necessary in the interpretation and processing of sustainability metrics.

It is lacking in a strong error-handling mechanism to ensure quality at various levels of the process, such that the reliability and accuracy of analytical results are severely impacted and are not reliable for informed decisions based on the data that went through the pipeline. Hence, lack of proper error-handling protocols and quality assurance mechanisms reduce the overall efficiency and practicability of the pipeline in real-world applications

A paper by D. Doreswamy and S. Krishna Bhat titled "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications,"[10] published in the International Journal of Financial Studies in 2023, is a comprehensive review of techniques in forecasting stock market prices. While the paper contributes to knowledge consolidation in machine learning and deep learning models, there exist some limitations and disadvantages with the study. Notable among these is a reliance on previous studies such that the current study fails to take into account the current updates or alternative approaches within stock market prediction. The review may also mostly be focused on technical metrics of performance, such as accuracy and precision, thereby failing to address the difficulties of applying these models on a practical level, such as interpretability and resource use. Discussion may also miss to touch upon the relevance of extraneous factors of political nature, economic policies or any shock in the marketplace which influences stock prices to some large extent but cannot really be modeled. Another limitation is that there may be an underestimation of hybrid models integrating traditional and deep learning methods, which are increasingly coming to the fore for being superior. The implications as discussed may also be somewhat too general and not very practically actionable for practitioners who look forward to implementing these models in real-life applications. These limitations emphasize the importance of more balanced, contemporary, and practically relevant evaluation in future systematic reviews on stock market forecasting methods.The conclusions developed within the study also seem likely to be too broad and do not reach the specificity or the practical actionable insights that are necessary for the practitioner who is trying to use these models in the context of real-world application. The difference between theoretical insight and practical usability represents a crucial weakness, since practitioners require clear, detailed, and context-specific guidelines in order to effectively apply these models in live environments. The challenges identified highlight the need for future systematic reviews to take a more balanced approach to ensure that evaluations of stock market forecasting methods are not only comprehensive and contemporary but also relevant in practice. This way, future research work will fill the gap and make a more concrete link of theoretical advances to practical reality by addressing real-world constraints that actual practitioners face, thus possibly increasing the adoption and, therefore, effectiveness of those forecasting models.

Table 1: Summary of Existing Approaches

Paper Title	Objective	Methodology	Significance	Drawbacks
Event-Driven Serverless ETL Pipeline on AWS (2021)	To design an event-driven ETL pipeline using AWS services for efficient data processing.	Utilized AWS Lambda, S3, and other AWS services to automate data extraction, transformation, and loading	The paper provides an efficient serverless solution for automating ETL processes, reducing costs	Complex configuration of AWS services can be difficult for teams with limited cloud expertise. Scalability may

		(ETL) in a serverless architecture.	and improving scalability.	be limited when handling highly dynamic datasets
Stock Market Analysis (2021)	To analyze stock market data and predict future stock price trends.	Uses statistical models and basic machine learning techniques for market analysis and forecasting.	Provides a foundational analysis of stock market trends and the potential of machine learning for stock predictions.	Limited in scope and may oversimplify complex market dynamics. Lack of incorporation of macroeconomic or geopolitical factors affecting stock prices.
Cloud-Native ETL: Integrating Databricks and Azure Data Factory (2024)	To explore the integration of Databricks and Azure Data Factory for scalable ETL processing in cloud-native environments.	Combination of Databricks for data processing and Azure Data Factory for orchestration of ETL tasks.	Presents a robust solution for large-scale data integration and processing in cloud environments, ideal for enterprise applications.	High cost and resource requirements for setting up the integrated environment. Limited scalability for certain data workloads.
A Fast and Scalable ETL Pipeline for Serverless Architectures (2020)	To design a scalable and fast ETL pipeline suited for serverless architectures.	The approach leverages serverless technologies such as AWS Lambda and AWS S3, with emphasis on reducing processing time.	Offers scalability and speed in data processing while leveraging serverless architecture to minimize infrastructure management.	Serverless architectures may face cold start issues and can struggle with larger datasets. Complex configurations can be challenging to set up.
Design and Implementation of Efficient ETL Pipelines for Incremental Data Loading (2023)	To design an ETL pipeline that supports efficient incremental data loading for improved processing performance.	Focus on incremental data loading techniques, using efficient querying and loading strategies to minimize data	Improves efficiency in processing incremental data, reducing load times and improving system performance.	The method may be less effective in handling highly variable or unstructured data. Data inconsistency issues can arise

		duplication.		if incremental loading is not managed carefully.
Optimizing Microservices Integration in ETL Pipelines Using Canonical Data Models (2020)	To optimize the integration of microservices in ETL pipelines by using canonical data models.	The paper proposes a canonical data model to standardize data representation and improve integration across microservices.	Helps to standardize data integration across various microservices, leading to improved consistency and interoperability.	Canonical models may introduce performance bottlenecks and reduce flexibility in adapting to new data sources. Complexity in maintaining a centralized model.
Stock Price Prediction using Machine Learning and Deep Learning (2021)	To apply machine learning and deep learning models to predict stock prices.	Various machine learning and deep learning models such as LSTM, CNN, and decision trees were employed for stock price forecasting.	Provides an overview of machine learning techniques for stock prediction, highlighting their potential in improving prediction accuracy.	Overfitting is common with deep learning models. Models may not account for sudden market events or external factors, reducing prediction accuracy.
Development of an ETL-Pipeline for Automatic Sustainability Data Analysis (2024)	To develop an ETL pipeline that automates sustainability data analysis.	Utilizes cloud-based ETL tools and automation to process large datasets related to sustainability.	Provides an automated, scalable approach for analyzing sustainability metrics, improving the efficiency of data processing.	Complexity in handling unstructured and diverse sustainability data. Scalability and adaptability issues with rapidly growing datasets.

Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review (2023)	To systematically review the application of machine learning and deep learning models for stock market price forecasting	The paper reviews a wide range of studies using machine learning and deep learning for stock market prediction.	Offers valuable insights into the performance of various forecasting models and provides a comprehensive overview of existing research.	May over-rely on past studies and fail to include the latest advancements. Lack of attention to model interpretability and external factors affecting market prices.
Design and Implementation of Efficient ETL Pipelines for Incremental Data Loading (2023)	To design an efficient ETL pipeline that supports incremental data loading for improved data processing	The pipeline design is based on incremental loading techniques, aiming to reduce the time and resources required for large data sets.	Improves the efficiency of ETL processes by reducing redundancy and accelerating data transfer in real-time systems.	Potential issues with scalability when handling very large datasets or unpredictable data flow. Data consistency and synchronization challenges during incremental loading.

2.2 Summary: Drawbacks of Existing Approaches

Deep learning models are powerful but very computationally intensive as well as large in dataset size, often making them unusable in real-time applications and even in resource-constrained organizations. This is true, especially when models need continuous updating or new data sources added to the ETL pipeline as mentioned in ETL pipeline papers. Some methods are not scalable, for example, when dealing with growing, dynamic datasets, or when scaling up from controlled, smaller environments into more complex real-world systems.

The research papers reviewed above, although providing valuable insights in their respective fields, share several common drawbacks and limitations. One key limitation across many of these papers is the reliance on historical data for prediction or analysis. In particular, this is a problem that afflicts stock market prediction since past performance does not always predict future market conditions, especially when unexpected geopolitical events, market crashes, or economic policy changes occur. It results in overfitting in machine learning and deep learning models, where the models work well on the training data but fail to generalize to unseen data.

Another common drawback is the complexity and resource requirements of the models discussed. Some of the approaches also do not scale well, especially when facing growing, dynamic datasets, or when scaling up from more controlled, smaller environments to large, complex real-world systems.

Most of the surveyed studies also failed to consider including external factors that can seriously impact results.Another important aspect is that, though automation is highlighted in the paper, it sometimes neglects error handling, data quality assurance, or real-time monitoring issues, which are of paramount importance to ensure reliability and effectiveness in the proposed systems.

Another important limitation is the potential lack of interpretability of the models.Especially when they are employed in the use cases related to stock market prediction or even ETL pipelines, such methods can function as a "black box," limiting the understanding of end-users on how the predictions are made or the analysis is actually conducted. The lack of transparency becomes an obstacle, especially in areas like finance and sustainability where decision making has to be explainable and justifiable.

Last but not least, most of the papers propose promising methodologies, yet they neither provide real-world validation or case studies, so it is to be shown as to how far their proposed solution could practically be put into work in a working scenario. Sometimes, this obstructs the practitioners from truly finding the actual effectiveness of methods within various operational scenarios. Such common limitations imply further refinements in the models, better external incorporation, and also extensive validation to be provided in real scenarios. These techniques then become far more actionable and dependable.

CHAPTER 3
PROPOSED METHOD

3.1 problem statement

The nature of the stock methods usually fail to process and analyze the huge and diverse datasets in a market is highly dynamic and complex. It is, therefore, hard to predict the actual stock prices. Traditional effective way to forecast stock prices. Additionally, the volatility and behaviors of small-cap, mid-cap, and large-cap stocks make the prediction process very difficult. Such systems today are not scalable, automated, and integrated with modern cloud environments and thus cannot handle large-scale datasets or make real-time predictions. These systems also frequently depend on overly complex models, risking overfitting failure to generalize to unseen data.

3.1.1 objective of the project

The objective of this project will cover all the challenges because the proposed stock market prediction system will be scalable, automated, and efficient with Azure cloud services. The core concept for this project is to combine several data streams sourced from different categories of stocks into robust preprocessing and feature engineering approaches and utilize interpretable machine learning models that allow precise forecasts of stock prices so actionable insights are delivered for the investors and analysts in consideration.

3.2.1 Explanation of Architecture diagram

The architecture of the Stock Market Pipeline and Prediction project integrates cloud-based tools and machine learning into a streamlined, scalable system. Using Azure Data Factory, historical stock data from Kaggle was ingested and stored in Azure Blob Storage for a centralized, secure repository. Data preprocessing, including feature engineering and normalization, is automated within Azure Machine Learning. A Linear Regression model was trained and tuned to predict the stock prices of companies based on metrics such as MAE and MSE. The trained model is deployed on Azure Kubernetes Service (AKS) with Batch Endpoints so that batch processing can scale up for efficient prediction of next day's price. Architecture provides a smooth flow of data, coupled with actionable insights from performance visualizations like the "Actual vs Predicted" plots, offering a strong solution for forecasting in the stock market.

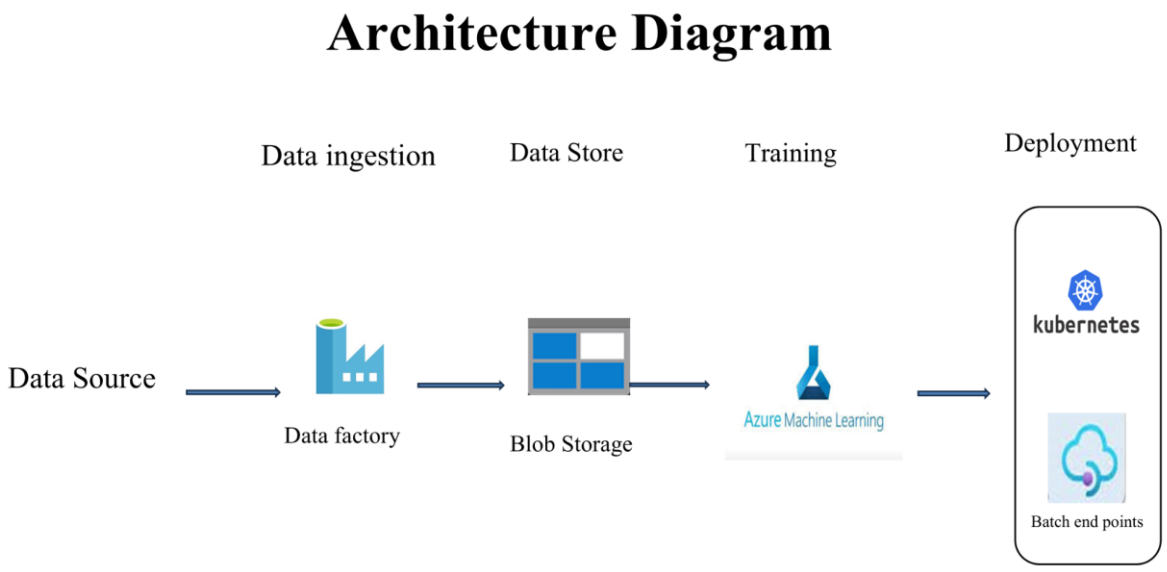


Figure 1: Architecture diagram

3.2.2 Module connectivity diagram

The module connectivity diagram shows how several components of the Stock Market Pipeline and Prediction system interact seamlessly. The Data Ingestion block is the starting point: it fetches historical stock data, uploads it into Azure Blob Storage through Azure Data Factory. The processed data is then held within the Preprocessing and Transformation block, where normalization and feature engineering occur before it can be used by machine learning tasks. This transformed data is fed into the Model Training module within Azure Machine Learning so that it can train a model using the Linear Regression model in order to predict the stock prices. This trained model is then deployed within AKS, allowing for batch predictions via Batch Endpoints. Last, but not least, the module Results Visualization, which does the job through performance metrics and "Actual vs Predicted" plots that give a holistic view of the accuracy and predictive capability of the system. Such connectivity assures a robust, scalable, and automated workflow for the forecast in the stock market.

Module Connectivity Diagram

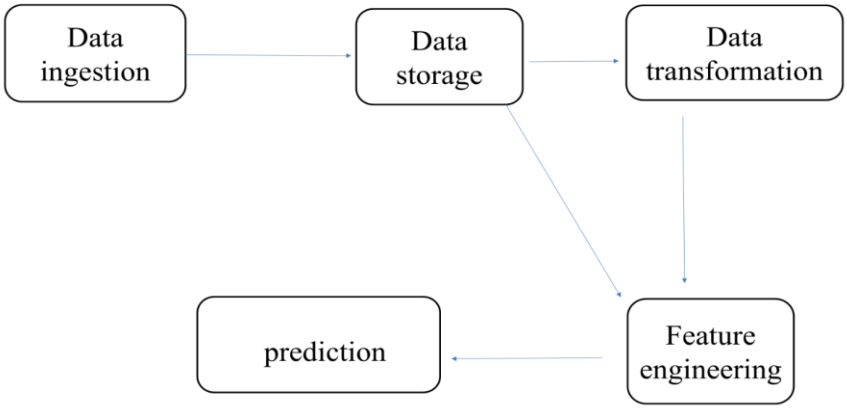


Figure 2 : Module Connectivity Diagram

3.2.3 Software and hardware requirements

Table 2 :software requirements:

software requirement	**platform**
Integrated Development Environment (IDE)	Azure Machine Learning Studio
Libraries	pandas, numpy, matplotlib, scikit-learn, azureml-core, azureml-dataset-runtime, azureml-pipeline, azureml-sdk
Cloud Services	Azure Data Factory, Azure Blob Storage, Azure Kubernetes Service (AKS)
Programming Language	Python

Table 3 : Hardware requirements:

Hardware requirement	**Platform**
Operating System	Windows,macOS or Linux
Processor	Intel Core i5 or equivalent
Compute Instance	Standard_DS3_v2 (4 vCPUs, 14GB RAM)
RAM	Minimum of 8GB

3.3 Analysis and Design through UML

3.3.1 Class diagram

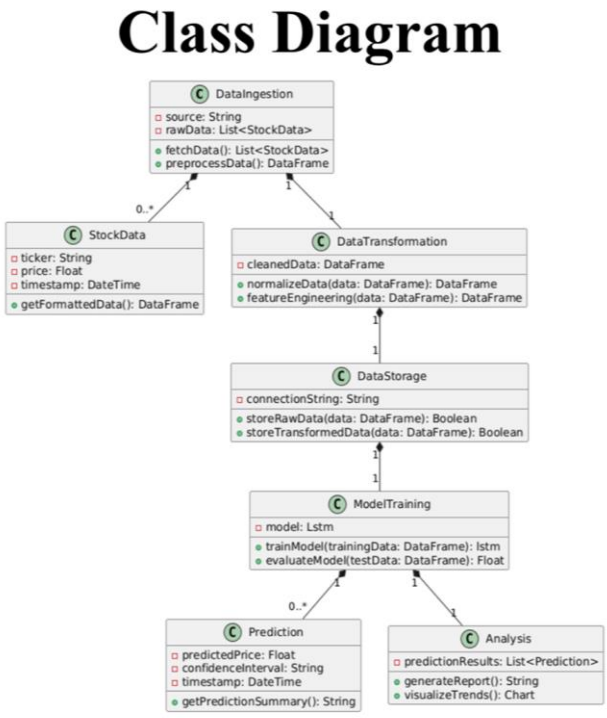


Figure 3: Class diagram

DataIngestion Class is designed to connect with Azure Blob Storage to ingest raw stock market datasets. DataPreprocessing Class will ensure that the data gets cleaned and feature engineered (e.g. lag features) and normalized data preparation so that the model is trained. ModelTraining Class will ensure training of the Linear Regression model, its evaluation through the metrics MAPE and RMSE, and saving of the trained model for further usage.Prediction Class takes the trained model and makes predictions on test data, then computes evaluation metrics based on accuracy. Visualization Class is responsible for generating visualizations, such as "Actual vs Predicted" plots, as well as model evaluation metrics for users to consume. Pipeline Class manages the pipeline-from ingestion, through preprocessing, training, prediction, and visualization steps-to smoothly run the workflow.

Composition: The Pipeline class has other instances of classes such as DataIngestion, DataPreprocessing, ModelTraining, etc., because the workflow encompasses all its stages.

Dependency: Classes such as Prediction depend on ModelTraining to retrieve the trained model and Visualization depends on Prediction to retrieve prediction results.

3.3.2 UseCase diagram

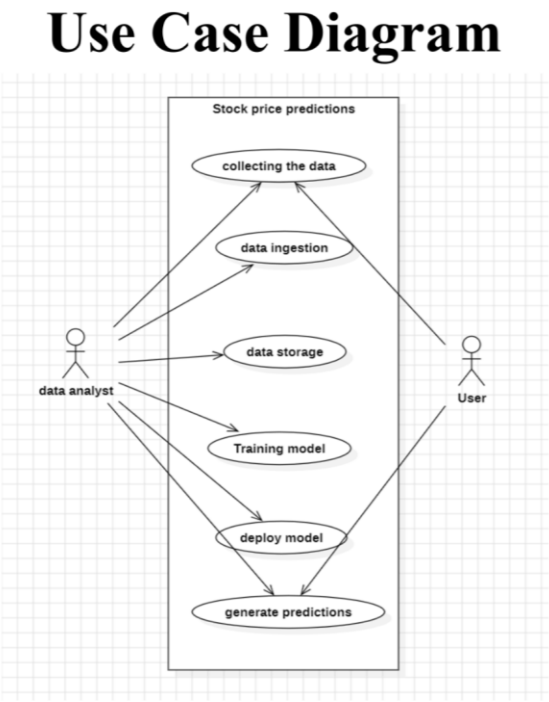


Figure 4 : Use Case diagram

Use Cases:

Data Ingestion: The system ingests historical stock data from external sources (e.g., Kaggle) and stores it securely in Azure Blob Storage for further processing.
Data Preprocessing: The Data Scientist preprocesses the raw data by handling missing values, normalizing features, and performing feature engineering (e.g., creating lag features for stock price prediction).
Model Training and Evaluation: The Data Scientist trains the Linear Regression model on the preprocessed data, evaluates it using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and generates performance metrics to assess accuracy.
Prediction Generation: The system generates stock price predictions for test data and makes future predictions based on the trained model. Investors or analysts use these predictions to make investment decisions.
Model Deployment: The Azure Administrator deploys the trained model on Azure Kubernetes Service (AKS) for scalable predictions, ensuring that the system can handle batch or real-time prediction requests efficiently.

Relationships:

Data Scientist interacts with most of the system components (e.g., Data Ingestion, Preprocessing, Model Training, Prediction, Visualization, and Pipeline Orchestration).
Azure Administrator handles cloud infrastructure-related tasks, such as Model Deployment and Data Ingestion. Investors/Analysts mainly use the Prediction Generation and Result Visualization functionalities to obtain predictions and view the results.

3.3.3 Data flow diagram

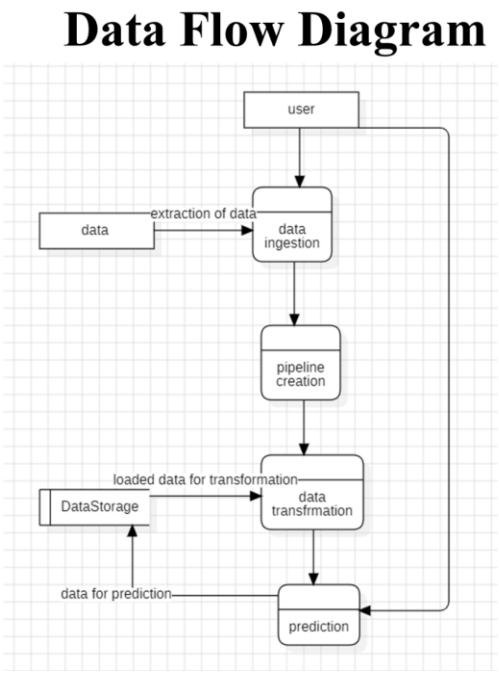


Figure 5 : Dataflow diagram

Data flow diagram represents data flow through the system-from ingesting data to generating predictions. It is assumed that raw stock market data is ingested from external sources such as Kaggle into Azure Blob Storage using Azure Data Factory. Then, the data undergoes preprocessing, which may include cleaning, feature engineering, and normalization in the Azure Machine Learning. The performance of the model is measured in terms of MSE, MAE, and MAPE using preprocessed data, where a Linear Regression model is trained. Once the model is obtained, scalable prediction follows on AKS (Azure Kubernetes Service). The predicted stock price values are plotted as "Actual vs Predicted" for easy interpretation by the users. It offers real-time or batch prediction with the smooth automation of the process while providing actionable insights to the decision-maker.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Description of Dataset

The dataset used for the Stock Market Pipeline and Prediction project is historical stock data that depicts small-cap, mid-cap, and large-cap stocks. This, therefore, provides diversity and comprehensiveness in analysis. The data came from Kaggle and entails the following key features:

Date: The date of trading of each record.

Open: The opening price of the stock for the date.

High: The highest price during the session.

Low: The lowest price recorded in the trading session.

Close: The closing price of the stock at the end of the trading session

Adj Close: The adjusted closing price that includes the effect of stock splits and dividends.

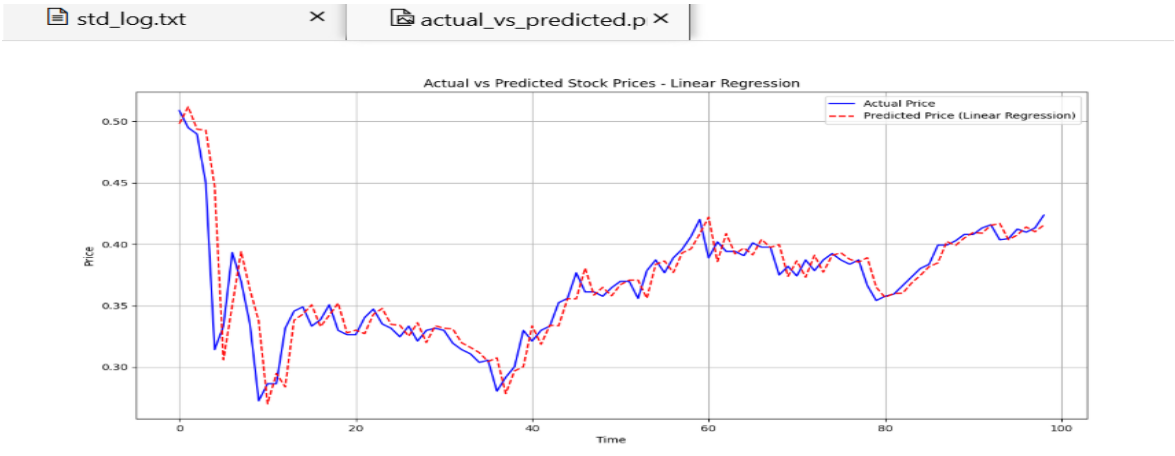
Volume: The total number of shares traded during the session.

The selected stocks consist of small-cap stocks such as Vuzix Corporation, Turtle Beach Corporation; mid-cap stocks, for instance, SSR Mining Inc., Alliance Resource Partners; and large-cap stocks, such as Microsoft Corporation, PepsiCo, Inc. This will help conduct an in-depth study on market behavior at various categories of stocks.

The dataset is in CSV and was ingested into Azure Blob Storage using Azure Data Factory. Basic preprocessing includes handling of missing values, normalization features, and creation of derived metrics for data preparation to utilize with machine learning. Then, the dataset will then be clean, consistent, and ready for predictive analysis.

4.2 Detailed Explanation of the experimental results

Large cap(MSFT)



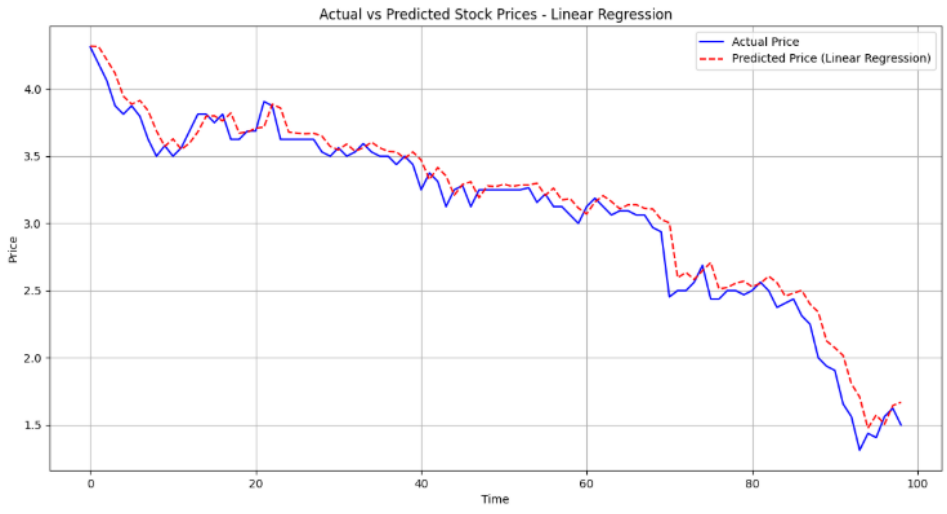
std_log.txt

```
Date of pipeline ML code run is :  2024-11-26
setting up the ML workspace
MeghanaWorkspace
stockmarket
westus2
6bca6fe6-9cc6-48db-a6ab-0f2f286cc348
setting up the blobstore for reading input files
3000
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.02
Mean Absolute Error (MAE): 0.01
Mean Absolute Percentage Error (MAPE): 3.52%
Model Accuracy: 96.48%
Predicted next day's price: 0.40
Cleaning up all outstanding Run operations, waiting 300.0 seconds
1 items cleaning up...
Cleanup took 0.062308549880981445 seconds
```

Figure 6 : Results of Largecap

Mid cap(SSRM)

std_log.txt | actual_vs_predicted.p

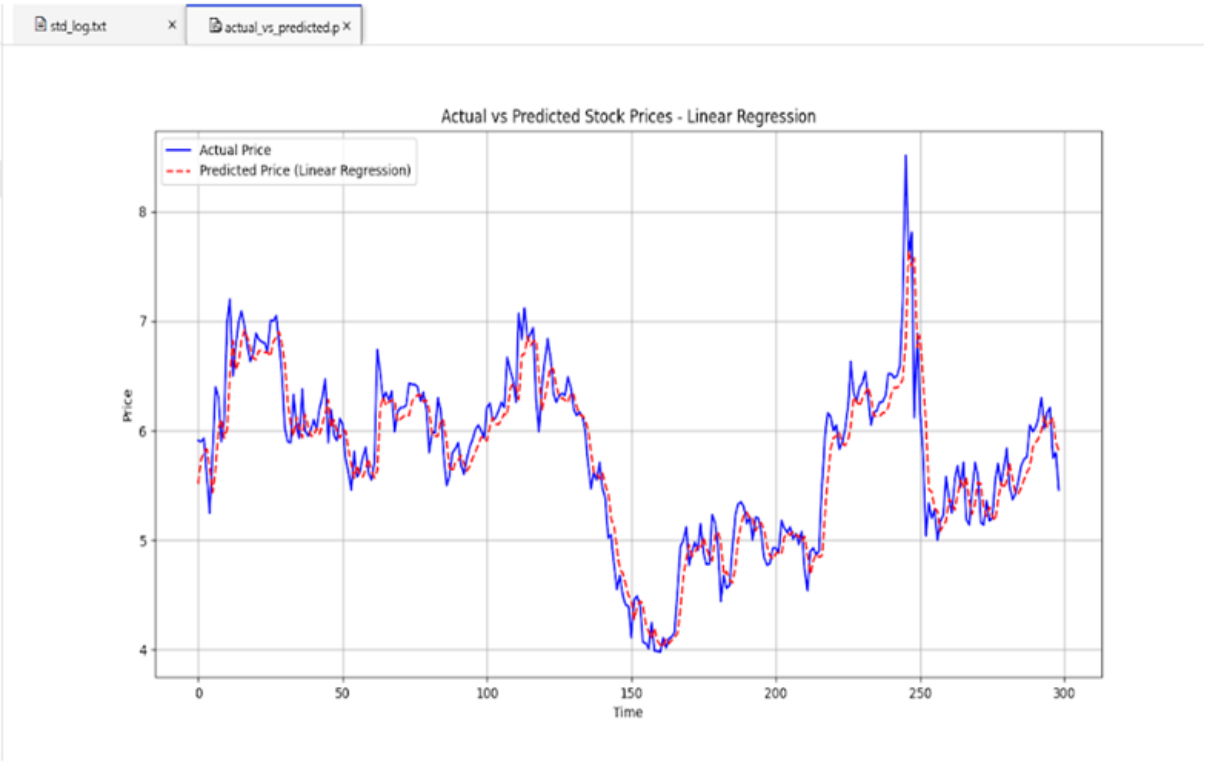


std_log.txt

```
Date of pipeline ML code run is :  2024-11-26
setting up the ML workspace
MeghanaWorkspace
stockmarket
westus2
6bca6fe6-9cc6-48db-a6ab-0f2f286cc348
setting up the blobstore for reading input files
3000
Mean Squared Error (MSE): 0.02
Root Mean Squared Error (RMSE): 0.14
Mean Absolute Error (MAE): 0.10
Mean Absolute Percentage Error (MAPE): 3.89%
Model Accuracy: 96.11%
Predicted next day's price: 1.54
Cleaning up all outstanding Run operations, waiting 300.0 seconds
1 items cleaning up...
Cleanup took 0.07543396949768066 seconds
```

Figure 7 : Results of Midcap

Smallcap



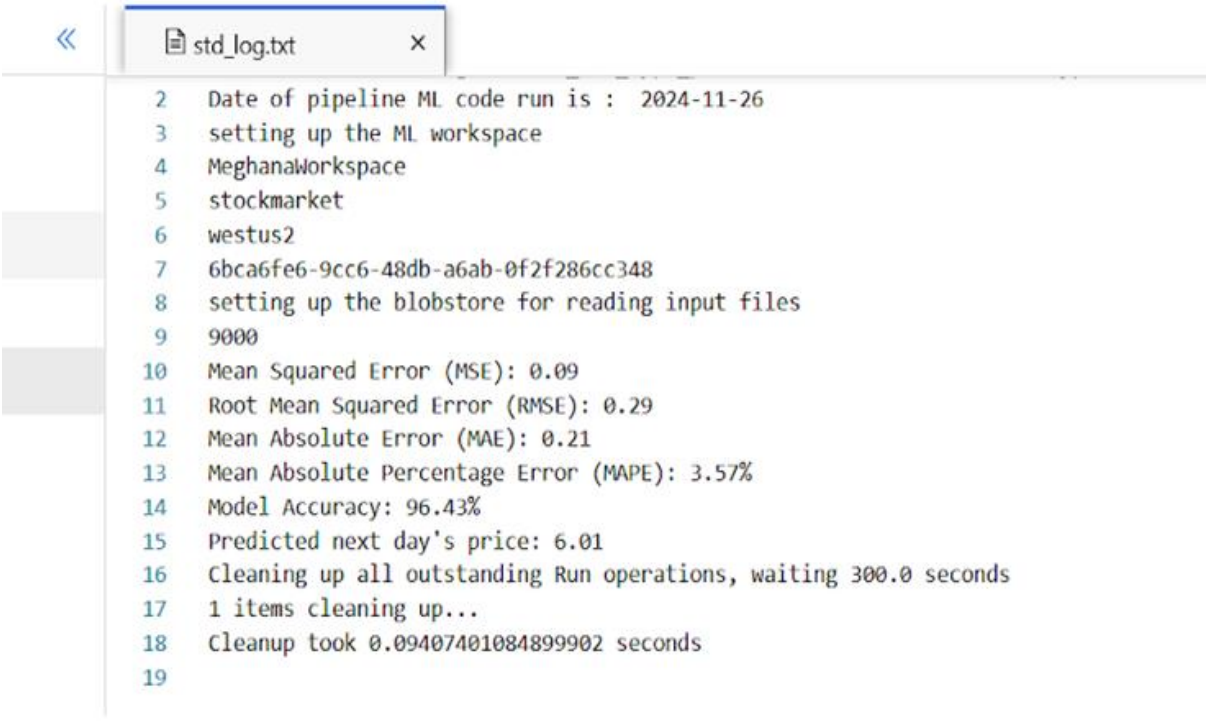


Figure 8 : Result of smallcap

4.3 Test Case

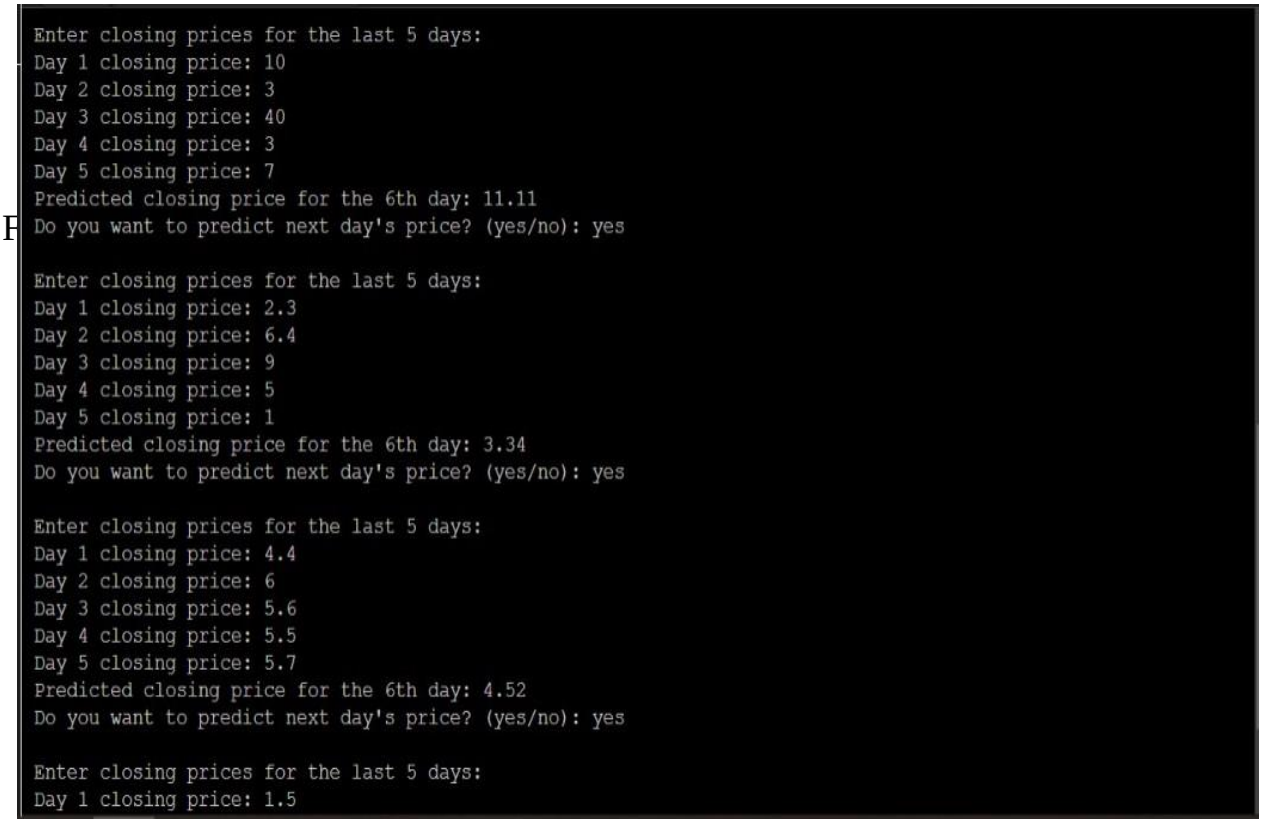


Figure : Test case1

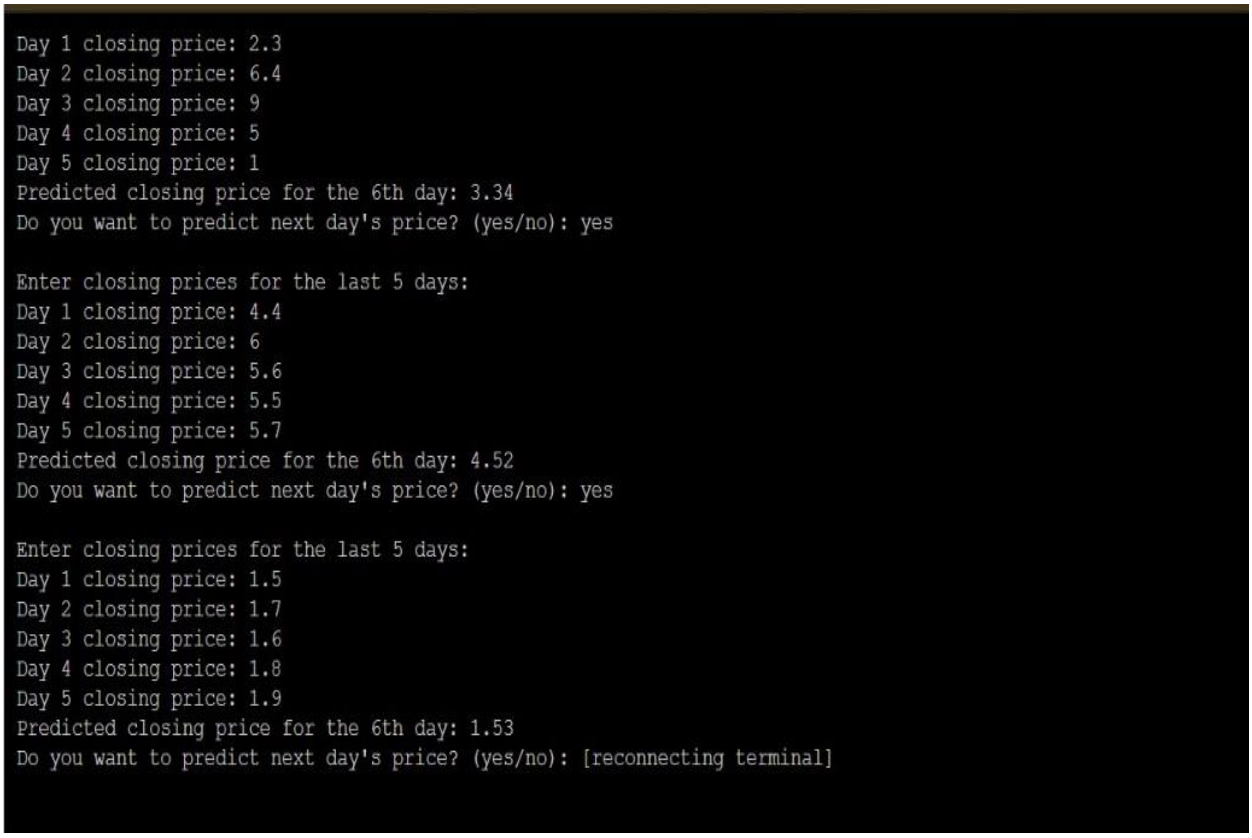


Figure : Test Case1

CHAPTER 5
CONCLUSION AND FUTURE ENHANCEMENTS

The Stock Market Pipeline and Prediction project successfully integrates cloud-based infrastructure and machine learning techniques to build a scalable, automated system for forecasting stock prices. It leverages Azure Machine Learning and Azure Blob Storage to deal with huge datasets quite efficiently and thus ensures smooth ingestion, preprocessing, and training of models. A Linear Regression model was built to have adequate foresight of predicting stocks at an accurate price and with some results visualized as per direction for investors and analysts in sight. Azure Kubernetes Service inbuilt cloud resources empower scailability in model deployment besides carrying the capability of predicting data instantly.

Automated pipeline for a process from data ingestion through generation of predictions - therefore, a pretty robust, efficient system open to future extension.

Further improvement ideas may be suggested to make the system more efficient and perform better. It will be integrated with sentiment analysis of financial news or social media, in which it will consider the market's mood for predicting the stock price and may allow the model to take into account external factors that can drive markets' actions. Moreover, real-time data intake pipelines can be developed so that the system can actually give actual live stock predictions. Thus, it will make it more dynamic and response-oriented towards the changes in the market. Multi-stock portfolio prediction and optimization functionalities can also be included to make investors better at making investment decisions. The expansion of this system's deployment through REST APIs will enable a seamless interaction with external applications, while enhanced visualization dashboards will provide users with more interactive insights into the predictions of the model. All these advances will make the system grow to handle the growing demands of the financial world.

APPENDICES

Pythonscript.py

```
from datetime import datetime, date, timedelta
import numpy as np
import os
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error,mean_absolute_error, r2_score

from azureml.core import Workspace, Dataset, Datastore
print("Date of pipeline ML code run is : ",date.today())

from azureml.core import Run
run = Run.get_context()
print('setting up the ML workspace')
ws=Workspace(subscription_id='6bca6fe6-9cc6-48db-a6ab-
0f2f286cc348',resource_group='stockmarket',workspace_name='MeghanaWorkspace')
print(ws.name,ws.resource_group,ws.location,ws._subscription_id,sep='\n')

blob_datastore_name='blobconnection'
container_name=os.getenv("BLOB_CONTAINER","smdata")
account_name=os.getenv("BLOB_ACCOUNTNAME","stockmarket22241")
account_key=os.getenv("BLOB_ACCOUNT_KEY","ig/ouFUuVK+PCHqTsdfOT6BfX9qT
gnBIktlhddJxuIYJWfwi8sG2KTVWcBqXOixmi0gdGdH48aHE+AStbWDmyA==")

datastore= Datastore.get(ws, 'blobconnection')
print('setting up the blobstore for reading input files')

from azureml.core import Run
run=Run.get_context()
```

```
df=Dataset.Tabular.from_delimited_files(path=[(datastore,"ARLP.csv")]).to_pandas_datafra
me()

df = df.iloc[:500]
df = df.copy()
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

df.head()

print(df.size)
df.index = pd.to_datetime(df.index)

plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Closing Price')
plt.title('Stock Closing Price History')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()



look_back = 5  # Number of previous days to consider
for i in range(1, look_back + 1):
    df[f'Close_Lag_{i}'] = df['Close'].shift(i)

df.dropna(inplace=True)



X = df[[f'Close_Lag_{i}' for i in range(1, look_back + 1)]]
y = df['Close']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
shuffle=False)

model = LinearRegression()
```

```
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Error metrics
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
accuracy = 100 - mape

# Print metrics
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
print(f"Model Accuracy: {accuracy:.2f}%")

# Plot actual vs predicted prices
plt.figure(figsize=(14, 7))
plt.plot(y_test.values, label="Actual Price", color='blue')
plt.plot(y_pred, label="Predicted Price (Linear Regression)", linestyle='--', color='red')
plt.title("Actual vs Predicted Stock Prices - Linear Regression")
plt.xlabel("Time")
plt.ylabel("Price")
plt.legend()
plt.show()

# Get the last 5 closing prices (same as the number of features used for training)
last_5_days = df['Close'].iloc[-5:].values  # Get the last 5 closing prices

# Convert to a DataFrame with appropriate column names (matching training features)
last_5_days_df = pd.DataFrame([last_5_days], columns=[f'Close_Lag_{i}' for i in range(1,
6)])

# Predict the next day's price using the trained model
```

```
future_pred = model.predict(last_5_days_df)
print(f"Predicted next day's price: {future_pred[0]:.2f}")
```

pipeline.py

```
import azureml.core
from azureml.core import Workspace, Datastore
from azureml.core.compute import ComputeTarget, AmlCompute
from azureml.core.runconfig import RunConfiguration
from azureml.core.conda_dependencies import CondaDependencies
from azureml.core import Environment
from azureml.pipeline.steps import PythonScriptStep
from azureml.pipeline.core import Pipeline
from azureml.core import Experiment

# Load the AzureML Workspace
ws = Workspace.from_config()
datastore=Datastore.get(ws,'blobconnection')
# Specify the compute target
compute_name = "meghanakammam1"
vm_size = "Standard_E4ds_v4"
compute_target = ws.compute_targets[compute_name]

# Configure the run environment
aml_run_config = RunConfiguration()
aml_run_config.target = compute_target

USE_CURATED_ENV = True
if USE_CURATED_ENV:
    # Use a curated AzureML environment
    curated_environment = Environment.get(workspace=ws, name="AzureML-sklearn-0.24-ubuntu18.04-py37-cpu")
    aml_run_config.environment = curated_environment
else:
```

```
    # Create a custom environment
    aml_run_config.environment.python.user_managed_dependencies = False
    aml_run_config.environment.python.conda_dependencies = CondaDependencies.create(
        conda_packages=['pandas', 'scikit-learn'],
        pip_packages=['azure-sdk', 'azureml-dataset-runtime[fuse,pandas]','packaging<=23.0'],
        pin_sdk_version=False
    )
# Define the Python script step
entry_point = "pythonscript.py"

from azureml.pipeline.core import PipelineData

# Define outputs
output_dir = PipelineData("output_dir", is_directory=True)
py_script_run_step = PythonScriptStep(
    script_name="pythonscript.py",
    compute_target=compute_target,
    runconfig=aml_run_config,
    outputs=[output_dir],  # Register outputs
    allow_reuse=False
)
# Create the pipeline
pipeline_steps = [py_script_run_step]
pipeline1 = Pipeline(workspace=ws, steps=pipeline_steps)
# Submit the pipeline run
pipeline_run1 = Experiment(ws, 'ARLP').submit(pipeline1)
pipeline_run1.wait_for_completion(show_output=True)
```

CHAPTER 6

REFERENCES

[1] **Antreas and G. Samakovitis,** "An Event-Driven Serverless ETL Pipeline on AWS, " *Applied Sciences*,vol. 11, no. 1, pp. 191, 2021.

[2] **M. Thoutam,** "Cloud-Native ETL: Integrating Databricks and Azure Data Factory for Scalable Data Processing in Enterprise Environments, " *International Journal for Multidisciplinary Research*, vol. 6,no. 6, pp. 1–15, 2024.

[3] **D. Doreswamy and S. Krishna Bhat,** "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications,"*International Journal of Financial Studies*, vol. 11, no. 3, pp. 94, 2023.

[4] **R. Jain and R. Vanzara,** "Emerging Trends in AI-Based Stock Market Prediction: A Comprehensive and Systematic Review, " *Engineering Proceedings*, vol. 56, no. 1, pp. 254, 2023.

[5] **A. Thota, M. Ganapuram**, S. R. Maddineni, and S. Cheemalamarri, "A Fast and Scalable ETL Pipeline for Serverless Architectures, " in *Procedia Computer Science*, vol. 171, pp. 2565-2573, 2020.

[6] **A. M. B. Dawood, A. M. Hussein, and M. K. Ibraheem**, "Design and Implementation of Efficient ETL Pipelines for Incremental Data Loading, " in *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)*, vol. 11, no. 3, pp. 274-280, 2023.

[7] **T. Villaça, R. Oliveira, and H. Lemck**e, "Optimizing Microservices Integration ETL Pipelines Using Canonical Data Models, " in *Informatik 2020: Proceedings of the 50th Annual Conference of the Gesellschaft für Informatik*, 2020.

[8] **Akintayo, A., Ahmed, S., Gana, I., Okwuosa, I., Ugbebor, C., and Ojo, O.,** "Development of an ETL-Pipeline for Automatic Sustainability Data Analysis, " in *International Journal of Sustainable Development & World Ecology*, vol. 30, pp. 455-467, 2024.

[9] **Pratheeth S, Vishnu Prasad R**, "Stock Price Prediction using Machine Learning and Deep Learning" , in IEEExplore. 2021.

[10] **P. Kavya, S. Saagarika, R T Subhavarshin**i, "STOCK MARKET ANALYSIS" in Research gate in 2021.