US STOCK MARKET ANALYSIS USING MACINE LEARNING

A PROJECT REPORT

Submitted by

JEEVA M

SHANKARA NARAYANAN R

THINESH M

HAARISH GOKUL M

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY, 2025

US STOCK MARKT ANALYSIS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

JEEVA M(811721243019)

SHANKARA NARAYANAN R(81172143051)

THINESH M(811721243058)

HAARISH GOKUL M(811720243301)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

MAY, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"US STOCK MARKET ANALYSIS USING MACHINE LEARNING"** is the bonafide work of **JEEVA M (REG.NO: 811721243019), SHANKARA NARAYANAN R (REG. NO: 81172143051), THINESH M (REG. NO: 811721243058), HAARISH GOKUL M (REG. NO: 811720243301)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. T.Avudaiappan M.E., Ph.D.,	Mrs. Ganga Naidu, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Artificial Intelligence	ASSISTANT PROFESSOR
K.Ramakrishnan College of Technology	Department of Artificial Intelligence
(Autonomous)	K.Ramakrishnan College of Technology
Samayapuram – 621 112	(Autonomous)
	Samayapuram – 621 112

Submitted for the viva-voce examination held on ………………

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"US STOCK MARKET ANALYSIS USING MACHINE LEARNING"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

____

JEEVA M

____

SHANKARA NARAYANAN R

____

THINESH M

____

HAARISH GOKUL M

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN, B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thank to **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this project.

We express our deep and sincere gratitude to our project guide **Mrs. GANGA NAIDU**,**M.E.**, Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

This project presents a comprehensive system for U.S. stock market analysis using machine learning, aimed at classifying stocks based on risk levels. By leveraging real-time financial data from Yahoo Finance, the system analyzes NASDAQ-listed stocks using advanced technical indicators such as Moving Averages (MA50, MA200), Relative Strength Index (RSI), Bollinger Bands, Volatility, and Daily Returns. A LightGBM model, trained on historical stock data, is employed to categorize each stock into Low, Medium, or High risk. The application features a user-friendly web interface developed with Flask, which displays live risk assessments, price trends, and technical indicators. Additionally, a Command-Line Interface (CLI) and a RESTful API with secure token-based authentication allow flexible integration into broader financial systems. The platform also supports natural language interaction through an OpenAI plugin, enabling users to query stock insights conversationally. This combination of real-time data analysis, technical expertise, and machine learning delivers a powerful decision-support tool for investors and financial analysts, making stock market insights more accessible, interactive, and actionable for informed investment strategies.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	**v**
	LIST OF FIGURES	**viii**
	LIST OF SYMBOLS AND ABBREVIATIONS	**ix**
1	**INTRODUCTION**	**1**
	1.1 BACKGROUND	1
	1.2 PROBLEM STATEMENT	2
	1.3 OBJECTIVE	2
2	**LITERATURE SURVEY**	**4**
	2.1 STOCK MARKET ANALYSIS USING MAPREDUCE AND PYSPARK	4
	2.2 IS THERE A BANKING RISK PREMIUM IN THE US STOCK MARKET?	5
	2.3 APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN THE ECONOMY, INCLUDING APPLICATIONS IN STOCK TRADING, MARKET ANALYSIS, AND RISK MANAGEMENT	6
	2.4 STOCK MARKET FLUCTUATIONS AND CONSUMPTION BEHAVIOUR	7
	2.5 NON-LINEAR DECOMPOSITION ANALYSIS OF RISK AVERSION AND STOCKHOLDING BEHAVIOR OF U.S. HOUSEHOLDS	8
3	**SYSTEM ANALYSIS**	**9**
	3.1 EXISTING SYSTEM	9
	3.1.1 Drawbacks of Existing System	11
	3.2 PROPOSED SYSTEM	11
	3.2.1 Advantages of Proposed System	14

4 SYSTEM SPECIFICATIONS **15**
4.1 HARDWARE SPECIFICATIONS 15
4.2 SOFTWARE SPECIFICATIONS 15
4.3 SOFTWARE DESCRIPTION 15
4.3.1 Library 16
4.3.2 Development Environment 17

5 SYSTEM DESIGN **19**
5.1 SYSTEM ARCHITECTURE 19
5.2 DATA FLOW DIAGRAM 20

6 MODULES DESCRIPTION **21**
6.1 INSTALLATION AND SETUP 21
6.2 DATA PREPROCES AND FEATURE EXTRACTION 22
6.3 MODEL DEVELOPMENT AND FORECASTING 23
6.4 EVALUATION AND VISUALIZATION 24
6.5 WEB INTEGRATION AND DEPLOYMENT 25

7 RESULT AND PERFORMANCE COMPARISON **27**
7.1 CLASSIFICATION PERFORMANCE 27
7.2 COMPUTATION TIME COMPARISON 28

8 CONCLUSION AND FUTURE ENHANCEMENTS **29**
8.1 CONCLUSION 29
8.2 FUTURE ENHANCEMENTS 30

APPENDIX A SOURCE CODE **31**

APPENDIX B SCREENSHOTS **38**

REFERENCES **39**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	System Architecture	19
5.2	Data Flow Diagram	20
7.1	Classification Performance Chart	27
7.2	Computation Time Comparison	28
B.1	Visual Studio Code	38
B.2	Stock Risk Analysis	38

LIST OF ABBREVIATIONS

API	-	Application Programming Interface
CLI	-	Comment Line Interface
KNN	-	K-Nearest Neighbour
LSTM	-	Long Short-Term Memory
MAE	-	Mean Absolute Error
OS	-	Operating System
RMSE	-	Root Mean Square Error
RPM	-	Revolutions Per Minute
SVM	-	Support Vector Machine
yfinance	-	Yahoo Finance

CHAPTER 1
INTRODUCTION

The dynamic nature of the stock market demands accurate, real-time insights to make informed investment decisions. With the advent of machine learning and data analytics, it is now possible to derive actionable intelligence from historical and live stock market data. This project introduces a Machine Learning-based U.S. Stock Market Analysis System designed to classify stocks based on risk—Low, Medium, or High—by analyzing various technical indicators. Leveraging data from Yahoo Finance, the system focuses on NASDAQ-listed stocks and utilizes features like 50-day and 200-day Moving Averages, Relative Strength Index (RSI), Bollinger Bands, Volatility, and Daily Returns.

The classification model is built using the LightGBM algorithm due to its high performance in handling structured data. The system is accessible via a Flask-based web application, a secure RESTful API with token authentication, and a Command-Line Interface (CLI). Additionally, integration with OpenAI plugins enables users to interact with the model through natural language queries.

This chapter outlines the motivation behind Stock prediction system, its objectives, and its relevance in today's financial ecosystem, where intelligent automation is becoming critical for portfolio management and risk assessment.

1.1 BACKGROUND

The stock market is a complex, dynamic environment influenced by a multitude of factors, including economic indicators, market sentiment, global events, and company performance. Traditionally, investors have relied on financial reports, news, and expert advice to make trading decisions. However, with the rise of big data and advancements in machine learning, there has been a paradigm shift towards automated systems that can process large volumes of data, identify patterns, and make predictions with greater accuracy and speed.

Machine learning algorithms are particularly well-suited for financial market analysis due to their ability to learn from historical data and uncover intricate relationships between variables. The application of such techniques allows for classification, regression, clustering, and anomaly detection—each playing a crucial role in evaluating market trends, assessing stock risks, and optimizing portfolios.

Stock prediction system builds upon these developments by using machine learning to analyze U.S. stock market data and classify stocks into different risk categories. By incorporating various technical indicators and utilizing a robust model like LightGBM, the system aims to provide retail investors and analysts with a powerful tool for informed decision-making in a volatile market landscape.

1.2 PROBLEM STATEMENT

The volatility and unpredictability of the stock market make it difficult for individual investors to consistently make profitable decisions. Traditional methods of stock analysis often rely heavily on human intuition and static financial metrics, which may not capture real-time market dynamics and complex patterns hidden in vast datasets. Moreover, manually analyzing stock performance across hundreds or thousands of companies is time-consuming and prone to error. With the increasing availability of financial data and the complexity of factors influencing stock prices, there is a growing need for intelligent systems that can automate risk analysis and decision-making. However, designing an efficient, accurate, and scalable model that can classify stock risk levels using real-world data remains a challenge.

Stock prediction system addresses the problem by developing a machine learning-based classification system that leverages key technical indicators and robust algorithms to assess the risk levels of stocks in the U.S. market. The aim is to enhance decision-making capabilities for investors by providing a reliable, data-driven classification of stocks into risk categories—low, medium, or high—thus reducing uncertainty and aiding in strategic investment planning.

1.3 OBJECTIVES

Stock prediction system aims to collect and process real-time stock data for NASDAQ-listed companies using the Yahoo Finance API. It involves computing key technical indicators such as Moving Averages (MA50, MA200), Relative Strength Index (RSI), Volatility, Bollinger Bands, and Daily Returns to support comprehensive stock analysis. Advanced machine learning models, including LightGBM and Random Forest, are designed and implemented to classify stocks into Low, Medium, or High risk categories with high accuracy.

A user-friendly web interface is developed using Flask to allow seamless visualization and interaction with the analysis results. Additionally, a secure and scalable RESTful API is provided to enable automated stock analysis and easy integration into external systems. For quick and script-based execution, the system supports a Command-Line Interface (CLI). To further enhance user experience, Stock prediction system integrates with OpenAI plugins, allowing natural language-based queries and intuitive interaction with the analysis platform.

CHAPTER – 2

LITERATURE SURVEY

2.1 STOCK MARKET ANALYSIS USING MAPREDUCE AND PYSPARK

P. Kavya, S.Saagarika, R.T.Subavarsshini. C.G.Nivetheni.

Forecasting the stock market has become increasingly important for planning business activities, driving research across disciplines such as computer science, statistics, economics, finance, and operations research. Recent studies highlight that the vast amount of publicly available online information—such as Wikipedia, social forums, and media news—significantly influences investor sentiment and market behavior. As the stock market is highly sensitive to economic changes, the reliability of computational prediction models is crucial to avoid financial losses. In this paper, we conducted an extensive analysis on various stocks, including Stock Volatility Analysis on a dataset of 1,000 NYSE stocks. Our key contributions include the development of a dictionary-based sentiment analysis model tailored to the financial sector and the evaluation of this model in assessing the impact of news sentiments on stock movements across different markets. Using only news sentiments, we achieved a notable accuracy of 70.59% in predicting short-term stock price trends.

Merits

Uses robust statistical decomposition to quantify contributing factors like education, credit constraints, net worth.

Demerits

Risk preference is assessed subjectively, not via actual investment decisions.

2.2 IS THERE A BANKING RISK PREMIUM IN THE US STOCK MARKET?

Liujing Zeng, Hue Hwa Au Yong, Sirimon Treepongkaruna, Robert Faff.

The Banking risk premium investigates whether a banking risk premium exists that helps explain the returns of U.S. publicly listed firms, examining this phenomenon within the frameworks of the Capital Asset Pricing Model and the Fama and French three-factor model. To explore this, we construct a banking factor—a mimicking portfolio that is long on large banks and short on small banks—based on bank size. Our findings support a risk-based interpretation of the banking factor, indicating that a banking risk premium is both priced and systematic.

Merits

Uses U-Net for accurate color segmentation.

Demerits

The model's performance heavily depends on the diversity and quality of the training dataset.

2.3 APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN THE ECONOMY, INCLUDING APPLICATIONS IN STOCK TRADING, MARKET ANALYSIS, AND RISK MANAGEMENT

Amir Masoud Rahmani, Bahareh Rezazadeh, Majid Haghparast.

In an increasingly automated world, Artificial Intelligence (AI) promises to revolutionize how people work, consume, and develop their societies. While the pursuit of solutions through scientific and technological advancements continues, AI-based technology is not novel and already possesses a wide range of economic applications. This paper examines the use of AI in economics, particularly in stock trading, market analysis, and risk assessment. A comprehensive taxonomy is proposed to explore AI applications across various defined categories. Additionally, we discuss the most significant AI-based techniques and evaluation criteria in this domain. Finally, the paper identifies key challenges, open issues, and suggestions for future research.

Merits

Model-agnostic: Works with any machine learning mode, Local fidelity, User-friendly.

Demerits

Accuracy decay over time, Model complexity, Limited long-term forecasting.

2.4 STOCK MARKET FLUCTUATIONS AND CONSUMPTION BEHAVIOUR

Laurence Boone, Claude Giorno, Pete Richardson.

This study examines the likely influence of recent stock market fluctuations on major OECD economies, with a focus on wealth effects and consumption. After reviewing the relevant theoretical framework and existing empirical evidence, consumption functions are estimated for the U.S., incorporating the influence of financial wealth. These estimates of the marginal propensity to consume out of financial wealth are then extrapolated to other G7 countries, accounting for differences in stock market capitalization, and compared with estimates derived more directly from consumption functions that include stock market prices as an explanatory variable. Simulations are conducted to evaluate the potential global impact of a significant decline in stock market prices across G7 countries using a version of the OECD INTERLINK model that incorporates these equations. The findings indicate that the overall effects are significant, especially when international linkage mechanisms are considered.

Merits

Achieved 70.59% accuracy custom sentiment dictionary, Effective use of Big

Data (MapReduce, PySpark).

Demerits

Limited generalization, Overfitting risk, Sector-specific sentiment dictionary
may not generalize well beyond use-case.

2.5 NON-LINEAR DECOMPOSITION ANALYSIS OF RISK AVERSION AND STOCKHOLDING BEHAVIOR OF U.S. HOUSEHOLDS

M. Humayun Kabir, Shamim Shaku.

Using the Survey of Consumer Finances, this paper provides an empirical analysis of the low participation by U.S. households in stock markets. Our baseline probit regressions support the hypothesis that information costs deter stock market participation; however, this factor alone does not fully explain why a significant proportion of U.S. households exclude stocks from their investment portfolios. By grouping the sample according to risk preference characteristics, we highlight the role of risk aversion in stockholding decisions, revealing some firm and insightful predictions in the subsequent regressions. Finally, a comparison of results from two distinct surveys conducted in 2001 and 2004 shows evidence of the post-9/11 recession's impact on the stockholding behavior of different risk preference groups.

Merits

Efficient Use of Collaborative Filtering, Scalability and Performance.

Demerits

Data Sparsity Issue, No Use of Deep Learning or Advanced Machine Learning.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the current financial landscape, numerous stock market analysis platforms and tools exist that offer basic insights based on historical trends, fundamental metrics, or technical indicators. These platforms—such as Yahoo Finance, Google Finance, and brokerage-provided dashboards—often provide charts, news updates, and limited analytics features. However, most of these systems depend heavily on manual interpretation by users, offering static visualizations without intelligent, adaptive decision support.

While some advanced financial platforms integrate algorithmic trading and proprietary risk models, they are typically restricted to institutional investors or high-net-worth individuals. These tools often operate as black-box systems, providing little to no transparency into their decision-making processes. Moreover, such systems may not leverage real-time machine learning-based classification tailored for retail investors.

Another major limitation in existing systems is the lack of personalized risk classification. Traditional systems do not effectively use technical indicators in conjunction with machine learning models to categorize stocks dynamically based on current market data. This gap in real-time, automated, and accessible stock risk evaluation highlights the need for a smarter, more adaptive system

Algorithm Used

- **Simple Moving Average (SMA)**

SMA calculates the average closing price of a stock over a specific number of periods. It helps identify market trends by smoothing out price fluctuations. However, since it gives equal weight to all data points, it may lag behind rapid market changes and isn't effective for capturing short-term volatility.

- **Exponential Moving Average (EMA)**

RSI is a momentum oscillator that measures the speed and change of price movements. It ranges from 0 to 100, with values above 70 considered overbought and below 30 considered oversold.

RSI helps traders detect potential reversals but may generate misleading signals in trending markets.

- **Relative Strength Index (RSI)**

  Bollinger Bands consist of a moving average (typically 20-day) and two standard deviation lines. They expand and contract based on volatility. Prices nearing the upper band may indicate overbought conditions, while the lower band suggests oversold levels. Though helpful, bands don’t predict trend direction or reversals with high accuracy.

- **Bollinger Bands**

  Some traditional systems use cosine similarity to measure the closeness between books based on user ratings or metadata. Although it improves similarity-based recommendations, it doesn’t adapt well to dynamic changes in user behavior.

- **MACD (Moving Average Convergence Divergence)**

  MACD uses the difference between a short-term EMA and a long-term EMA to identify momentum and trend changes. A signal line helps generate buy or sell alerts. It is especially effective in identifying trend shifts but may lag during sideways markets or produce false signals in choppy conditions.

- **Average True Range (ATR)**

  ATR measures market volatility by taking the average of true ranges over a specific time period. It helps traders set stop-loss levels and assess risk levels. However, it doesn't indicate trend direction, making it better suited for gauging the intensity of price movements rather than their direction**.**

- **Linear Regression Analysis**

  This technique uses a straight line to model the relationship between stock price and time or other variables. It helps estimate future prices based on historical data. While simple and interpretable, it assumes a constant linear relationship and is ineffective for modeling complex or non-linear market behaviors.

3.1.1 Drawbacks

The existing system faces several limitations that hinder its effectiveness in stock analysis. It lacks support for real-time data, which is crucial for timely decision-making in dynamic markets. The prediction accuracy is limited, as the system relies on static, rule-based models that do not incorporate deep pattern recognition or adapt to changing market trends. Manual data entry is required, increasing the risk of human errors and reducing efficiency. Additionally, the system struggles in volatile market conditions due to its inability to adapt or make intelligent decisions. Its poor automation capabilities further contribute to inefficiency, making it unsuitable for modern, data-driven financial analysis.

3.2 PROPOSED SYSTEM

The proposed system, titled "US Stock Market Analysis Using Machine Learning," aims to overcome the limitations of traditional stock analysis platforms by integrating machine learning algorithms and real-time data processing. It uses powerful models like LightGBM and Random Forest to classify stocks into Low, Medium, or High risk categories based on a wide range of technical indicators. Real-time stock data is fetched using the Yahoo Finance API, ensuring that users receive up-to-date and accurate market information. The system also calculates key technical metrics such as Moving Averages (MA50, MA200), RSI, Volatility, and Bollinger Bands to enrich the decision-making process.

A user-friendly Flask-based web interface allows users to input stock symbols and instantly view detailed risk analysis, price trends, and graphical insights. The system also supports RESTful APIs for secure, token-authenticated integration into external applications, and a Command-Line Interface (CLI) for automation and scripting. Furthermore, OpenAI Plugin Support allows users to interact with the system using natural language queries. This proposed solution is a complete, intelligent, and adaptive platform designed to revolutionize how investors analyze stock market risks.

Algorithm Used

- **LightGBM (Light Gradient Boosting Machine)**

LightGBM is a gradient boosting framework developed by Microsoft, optimized for speed and efficiency. It uses a leaf-wise tree growth strategy instead of level-wise, resulting in faster training and higher accuracy. It supports categorical features directly and is highly scalable for large datasets. In the context of stock market analysis, LightGBM can quickly process technical indicators and historical stock data to classify stocks based on risk. Its performance in handling complex data patterns makes it ideal for real-time financial forecasting and risk-level predictions.

- **Random Forest**

Random Forest is an ensemble machine learning algorithm that builds multiple decision trees and merges their outputs for better prediction accuracy. It reduces overfitting and handles both classification and regression tasks efficiently. In stock market analysis, Random Forest can be used to analyze large volumes of historical and technical indicator data to predict stock risk levels. Its robustness to noise and ability to handle unbalanced datasets make it a preferred algorithm for financial modeling, offering a balanced trade-off between interpretability and predictive performance.

- **XGBoost (Extreme Gradient Boosting)**

XGBoost is a high-performance, scalable machine learning algorithm based on gradient boosting. It offers regularization to reduce overfitting and supports parallel processing, making it faster than traditional boosting methods. XGBoost is widely used in financial analytics for its ability to capture intricate patterns and relationships in time-series data. For stock market risk classification, it excels at learning from historical price trends and technical indicators, making predictions that are both accurate and reliable. Its interpretability and tuning flexibility enhance its effectiveness in financial decision-making.

- **Support Vector Machine (SVM)**

Support Vector Machine is a supervised learning model that separates data into classes using an optimal hyperplane. It is particularly effective in high-dimensional spaces and can handle both linear and non-linear data using kernel functions. In stock market analysis, SVM is useful for classifying stocks based on patterns derived from technical indicators like RSI, moving averages, and volatility. It is robust to overfitting, especially in cases where the number of dimensions exceeds the number of samples, making it suitable for financial classification problems.

- **K-Nearest Neighbors (KNN)**

  K-Nearest Neighbors is a simple, instance-based learning algorithm that classifies data points based on their proximity to labeled data. It does not require a learning phase, making it computationally light for small datasets. In stock market applications, KNN can classify stock risk levels by comparing technical indicator values with historical patterns. Though it struggles with large datasets and irrelevant features, it offers a transparent and intuitive approach, especially when trends and anomalies in stock behavior are best understood in the context of surrounding data points.

- **Naive Bayes Classifier**

  Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming feature independence. Despite its simplicity, it performs surprisingly well in many real-world situations. In stock market analysis, it can be applied to predict stock behavior or risk levels using historical data and technical indicators. It's especially useful when quick probabilistic decisions are needed, though its assumption of independent features may not always align with correlated financial data. Still, its low computation cost and fast performance make it suitable for initial screening tasks in risk assessment.

- **Decision Tree Classifier**

  Decision Tree Classifier is a flowchart-like tree structure where internal nodes represent features, branches represent decisions, and leaves represent outcomes. It is easy to understand and interpret, making it ideal for visualizing decision-making processes in financial applications. In stock risk classification, a decision tree uses technical indicators to split data into risk categories. Although prone to overfitting, it serves as a solid baseline or component in ensemble methods like Random Forest. It offers transparent decision rules, which can be useful for explaining predictions to users or analysts.

3.2.1 Advantages

The proposed system offers a range of significant improvements over traditional methods, including enhanced accuracy and faster computation, which enable more reliable and efficient stock analysis. It provides better risk assessment through advanced modeling techniques and supports high scalability to handle large volumes of data seamlessly. Real-time prediction capabilities, combined with automation, ensure timely insights and reduce the need for manual intervention. The system is designed for real-time adaptability, incorporating continuous feedback and adaptive learning to respond effectively to changing market conditions. With a user-friendly interface, it ensures ease of use, while offering model flexibility and robustness to handle diverse scenarios. Additionally, it serves as a powerful decision support tool, aiding investors and analysts in making informed, data-driven decisions.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE SPECIFICATIONS

- Computer - minimum of 8GB RAM & Intel Core i5 or equivalent processor, 256GB of storage.
- NVIDIA GPU for faster model training
- Additional Sensors – only if required like light sensors, Bio metric sensors, Temperature Sensor.
- Ethernet/Wi-Fi Adapter.

4.2 SOFTWARE SPECIFICATIONS

Python programming language - Python 3.x installed on the computer/server

- Operating system - Windows, Linux, or macOS.
- Python libraries such as – Scikit-learn, LightGBM, Pandas, NumPy, Matplotlib, yfinance, Flask.
- API Services - Yahoo Finance API – for real-time stock data extraction
- Dataset Source Used – Kaggle Books Dataset
- HTML/CSS or JavaScript - for UI design.

4.3 SOFTWARE DESCRIPTION

The software development for the project “US Stock Market Analysis Using Machine Learning” is centered around Python, a powerful and versatile programming language widely used in the fields of data science and artificial intelligence. Python provides extensive support for libraries such as Pandas, NumPy, Scikit-learn, Matplotlib, and LightGBM, which are essential for data manipulation, visualization, and implementing machine learning models. These libraries enable the efficient processing of large datasets and accurate prediction modeling for risk assessment in stock analysis.

To present the results interactively, Stock prediction system employs the Flask micro web framework. Flask is lightweight, flexible, and ideal for building web applications with RESTful APIs. It enables users to interact with the application via a web interface, providing inputs such as the stock symbol (NASDAQ code) and retrieving output such as current stock price, risk classification, daily returns, volatility, and trend graphs. Flask also facilitates the integration of authentication layers and handles HTTP requests seamlessly.

Stock prediction system utilizes the Yahoo Finance API, accessed via the yfinance Python module, to fetch real-time stock data. This includes historical pricing, market capitalization, trading volume, and other financial metrics necessary for technical indicator calculations such as Moving Averages (MA50 and MA200), RSI (Relative Strength Index), Volatility, and Bollinger Bands.

In addition, the system supports a Command-Line Interface (CLI) for automation and scripting, and includes OpenAI Plugin Integration for ChatGPT-based conversational queries. This architecture ensures modularity, extensibility, and real-time analytics for robust financial decision-making.

4.3.1 Libraries

- Pandas: Used for data manipulation and analysis. It helps in loading datasets, handling missing values, and organizing book metadata for effective feature extraction and processing.
- NumPy: Provides support for high-performance mathematical computations and multidimensional arrays, which are essential for similarity calculations and matrix operations used in the recommendation engine.
- Scikit-learn (sklearn): This is the core machine learning library used for implementing algorithms like cosine similarity, as well as for vectorization techniques such as CountVectorizer or TfidfVectorizer. It also includes tools for model evaluation and data preprocessing.
- LightGBM: A high-performance gradient boosting framework specifically designed for speed and efficiency. It is used to build accurate risk classification models on large stock datasets.
- yfinance: Used to extract real-time and historical data from Yahoo Finance. It

simplifies the process of accessing data such as opening and closing prices, trading volume, and market capitalization.

- Flask: A lightweight web framework for building the REST API and user interface. It handles routing, data rendering, and integration with frontend components.
- TA-Lib (or custom TA functions): Employed for technical indicator computation like RSI, Volatility, and Moving Averages, providing deep insight into market trends.

4.3.2 Developing Environment

The development environment plays a crucial role in the successful implementation of any data-driven machine learning project. For the project titled "US Stock Market Analysis Using Machine Learning," a carefully curated and modern development setup was used to ensure efficiency, scalability, and seamless integration between components. This section outlines the tools, frameworks, and platforms that together formed the foundation of the system.

- Programming Language: Python 3.x: Python was chosen as the primary programming language due to its high readability, robust support for machine learning, and extensive library ecosystem. Its ability to handle data manipulation, numerical computation, and machine learning modeling makes it ideal for financial data analysis. Python also ensures easy integration with various APIs and supports multiple machine learning frameworks.

- Development Platforms: Jupyter Notebook & Visual Studio Code: Jupyter Notebook was used extensively for data exploration, preprocessing, model training, and testing. Its interactive features allowed developers to visualize trends, test individual components, and refine algorithms in real time. Visual Studio Code served as the integrated development environment (IDE) for writing backend logic, creating Flask-based web APIs, and managing full project files.

- Framework: Flask: Flask, a lightweight Python web framework, was used to develop the web application’s backend. It enabled the creation of RESTful APIs that interact with the machine learning models and process real-time user input. Its modular structure and simplicity in routing made it ideal for building and deploying scalable applications with less overhead.

- Libraries and Packages:
    - Pandas and NumPy: for efficient data manipulation and numerical operation.
    - Matplotlib and Seaborn for data visualization.
    - Scikit-learn and LightGBM for building and training machine learning models.
    - yfinance for fetching real-time and historical stock market data directly from Yahoo Finance.
    - TA-Lib for calculating technical indicators like Moving Averages, RSI, Volatility, and Bollinger Bands.
- Version Control System: Git and GitHub: Git was used as the version control system to track code changes, maintain development history, and collaborate efficiently. GitHub hosted the remote repository, enabling secure backup and easy collaboration among team members.
- Operating System: Development was carried out primarily on a Windows 10/11 operating system. The OS provided compatibility with all development tools and ensured a stable environment for running Python, Flask, and related libraries.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The system design defines the structure and flow of the "Smart Book Recommendation System Using Machine Learning." It follows a layered architecture comprising the User Interface Layer, Application Logic Layer, Data Processing Layer, and Machine Learning Layer. The User Interface Layer handles user interactions such as searching and rating books. The Application Logic Layer processes inputs, routes data, and coordinates between the frontend and backend. The Data Processing Layer cleans and structures raw data for machine learning. The Machine Learning Layer uses algorithms like KNN, SVD, and Content-Based Filtering to generate personalized recommendations. A dedicated Database Layer manages user profiles, book metadata, and historical interactions. This modular design ensures scalability, maintainability, and efficient data flow, resulting in a robust and user-friendly recommendation system.

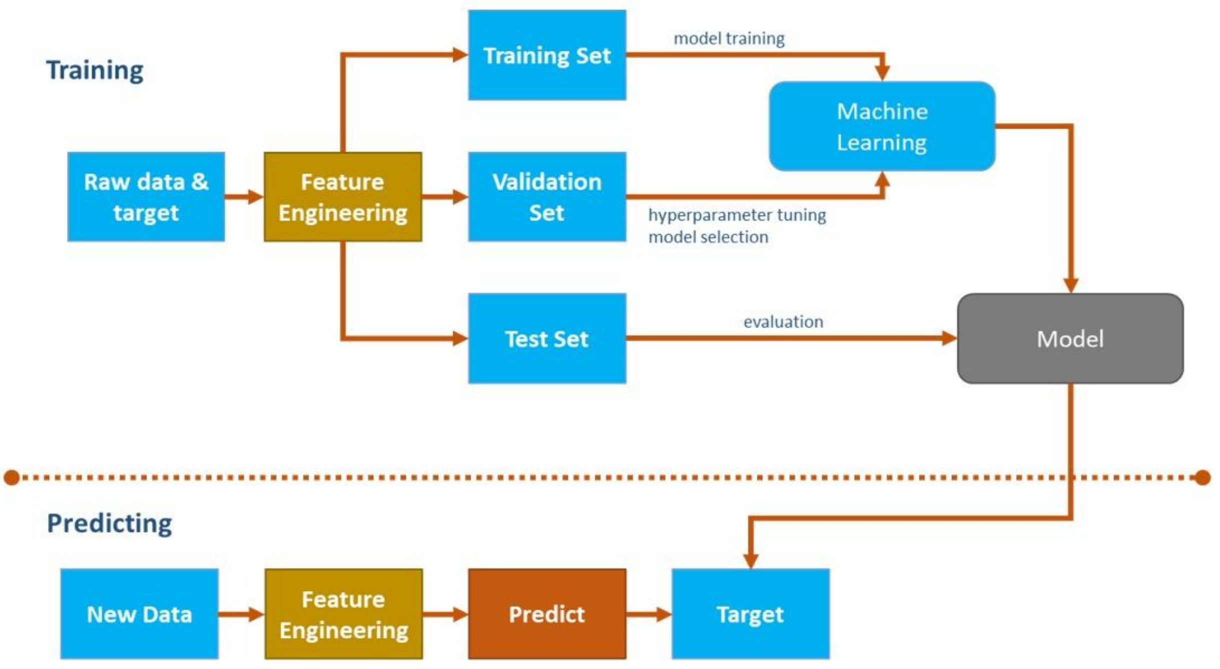


Fig.5.1 System Architecture

5.2 DATA FLOW DIAGRAM

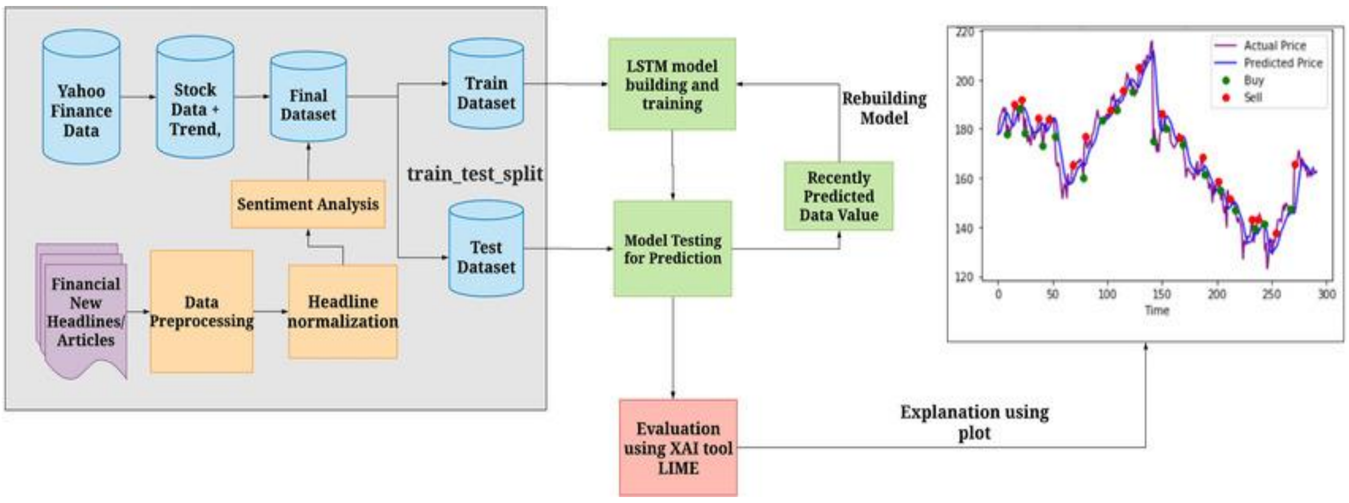


Fig.5.2 Data Flow Diagram

CHAPTER 6
MODULES DESCRIPTION

The US Stock Market Analysis System is structured around five major modules, each playing a critical role in the complete workflow. Those modules are:

- Installation and Setup
- Data Preprocessing and Feature Extraction
- Model Development and Forecasting
- Evaluation and Visualization
- Web Integration and Deployment

6.1 INSTALLATION AND SETUP

The installation and setup module provides the essential groundwork required to initiate the US Stock Market Analysis System. It involves preparing the development environment by installing necessary software and packages that enable data collection, analysis, model development, and web deployment. The development environment can be configured using tools like Anaconda or virtual environments (via venv) to ensure dependency isolation and easy management of Python packages. Once the environment is prepared, core libraries are installed using Python's package manager (pip) or through Conda.

The primary libraries include:

- pandas and numpy for data handling and numerical operations.
- matplotlib, seaborn, and plotly for data visualization.
- yfinance to retrieve historical stock data from Yahoo Finance.
- scikit-learn for machine learning algorithms.
- tensorflow for deep learning models, especially LSTM.
- flask for web integration and application deployment.

The next major component of this module involves the acquisition of historical financial data. The yfinance library is used to fetch stock data for NASDAQ-listed companies. Users can specify stock symbols, data intervals (e.g., daily, weekly), and date ranges. The dataset typically includes fields such as Open, High, Low, Close, Volume, and Date, all of which serve as raw input features for the system.

Additionally, this module defines the structure of the project directory. A well-organized directory may include separate folders for data storage, scripts for preprocessing, models, utilities, and web templates. Scripts are written to automate the downloading and storage of stock data for multiple tickers, ensuring a scalable setup that can support a wide range of stocks.

Configuration files such as .env for API keys or .gitignore to prevent versioning of large datasets and models may also be established. Initial testing is done to ensure all packages are working correctly and the data pipeline from Yahoo Finance is functional.

This module ensures the system is fully operational, with all components in place to support preprocessing, model training, evaluation, and eventual web deployment. It provides a seamless base that integrates data science workflows with practical web-based interfaces, preparing the user for a full-fledged stock market prediction system.

6.2 DATA PREPROCESSING AND FEATURE EXTRACTION

This module is crucial for transforming raw financial data into a structured and meaningful format that is suitable for machine learning models. Real-world stock market data often comes with inconsistencies, missing values, and noise, all of which can degrade the accuracy and reliability of predictive models. Therefore, the primary goal of this module is to clean, organize, and enhance the data to better reflect the underlying patterns in stock market behavior.

The preprocessing stage begins with data cleaning, where missing values are either imputed using statistical techniques such as forward fill, backward fill, or removed entirely if the proportion of missing data is high. Handling outliers is another important task, particularly in the context of financial data where price spikes or drops may distort the learning process. Statistical methods like Z-score filtering or interquartile range (IQR) are used to identify and treat these anomalies.

Normalization and scaling are then applied to bring all numeric values to a similar scale. This is especially important for machine learning algorithms like LSTM and gradient-boosted trees, which are sensitive to feature magnitude. Techniques such as Min-Max Scaling or Standardization (z-score normalization) are typically used, depending on the model requirements.

Temporal processing is also a key component in this module. Since stock market data is time-series based, it’s vital to convert date columns into appropriate datetime formats. Lag features are created to capture historical trends—this might involve including the closing price of previous days, weekly rolling averages, or prior day indicators. This helps preserve temporal continuity and supports models in learning from past behavior.

Once the data is cleaned and structured, feature engineering is applied to enhance model performance. This involves generating new, informative indicators that reflect market momentum, volatility, and trend direction. Commonly derived technical indicators include:

- Moving Averages (MA50, MA200): Used to detect trend direction over short and long terms.
- Relative Strength Index (RSI): Measures the speed and change of price movements to identify overbought or oversold conditions.
- MACD (Moving Average Convergence Divergence): Captures trend-following momentum.
- Bollinger Bands: Indicates market volatility and price deviations.

These technical indicators are calculated and added to the dataset, significantly boosting the model's ability to detect patterns and forecast future prices or risk levels. Feature selection methods like correlation analysis or feature importance scores may be used to retain only the most relevant features for modeling.

Overall, this module transforms raw, unstructured stock data into a robust and feature-rich dataset, ready for feeding into machine learning and deep learning models.

6.3 MODEL DEVELOPMENT AND FORECASTING

This module is the analytical core of the US Stock Market Analysis System, where historical stock data is used to train predictive models capable of forecasting future trends and classifying risk levels. It involves selecting appropriate machine learning and deep learning algorithms, preparing data for training, tuning model parameters, and validating performance to ensure generalizability and robustness.

The first step in this module is model selection. Depending on the forecasting objective—whether it's predicting future stock prices (regression) or classifying stocks into risk categories (classification)—appropriate models are chosen. For regression, models like Linear Regression, Random Forest Regressor, and LSTM (Long Short-Term Memory) networks are used. For classification tasks, especially for categorizing stocks into Low, Medium, or High risk, Random Forest Classifiers and LightGBM (Light Gradient Boosting Machine) are implemented due to their efficiency, interpretability, and scalability.

LSTM networks are particularly emphasized for time-series forecasting because they are designed to retain memory over long sequences, making them ideal for capturing trends, cycles, and temporal dependencies in stock price data. However, before LSTM training, data must be reshaped into a 3D format: [samples, time steps, features], which represents the sequence of inputs over time.

Data preparation for model training involves splitting the dataset into training and test sets using methods like time-based splits (rather than random splitting) to maintain chronological integrity. Scaling techniques such as Min-Max Normalization are applied to ensure uniform feature representation, which is critical for gradient-based learning methods like LSTM and LightGBM.

After training, models are evaluated using appropriate metrics: RMSE, MAE, and R^2 for regression; accuracy, F1-score, precision, and recall for classification. Models that demonstrate high performance and low error on unseen test data are saved for deployment using joblib or pickle.

By the end of this module, the system is equipped with trained models capable of making accurate predictions about future stock prices or assigning risk levels. These models form the engine behind real-time forecasting and decision support in the later stages of the system.

6.4 EVALUATION AND VISUALIZATION

This module plays a critical role in validating the performance of the predictive models built in the previous stage and presenting the outcomes in a visually intuitive manner. It serves two primary objectives: to ensure the accuracy and reliability of the predictions through rigorous evaluation metrics, and to convey these insights clearly through visualizations that can aid decision-making for users.

The evaluation phase begins with applying appropriate metrics depending on the task type—regression or classification. For regression models (e.g., predicting stock prices), metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R^2 Score are used. MAE provides a direct average of the errors between predicted and actual values, while RMSE penalizes larger errors more heavily, making it useful in financial domains where sudden price spikes can have high implications.

The R^2 Score evaluates how well the model explains the variability of the response data, with values closer to 1 indicating better fit.

For classification models (e.g., categorizing stocks into Low, Medium, or High risk), metrics such as Accuracy, Precision, Recall, and F1-score are used. Precision is important when false positives (wrongly labeling a stock as high risk) must be minimized, while recall is crucial when false negatives (failing to detect a risky stock) carry more danger. The F1-score provides a balanced measure of both, ensuring that the classifier performs well across all classes.

Beyond numerical evaluation, the module emphasizes visual communication of results. Visualizations are created using tools like matplotlib, seaborn, and plotly. For regression outputs, line plots are used to compare actual versus predicted stock prices over time, highlighting prediction accuracy and model lag. Residual plots help identify patterns in prediction errors, pointing to potential areas of model improvement. For classification tasks, confusion matrices are generated to visualize correct versus incorrect predictions for each class.

More advanced visualizations include candlestick charts, which show open, high, low, and close prices over a period—providing a more technical view of price movements. Correlation heatmaps help in understanding relationships between different input features and stock performance, supporting better feature selection. Time-series decomposition graphs can also be used to isolate trend, seasonality, and residuals in stock data.

This module ensures transparency and trust in the system's predictions by combining statistical rigor with visual clarity. It allows both technical analysts and non-technical users to understand the behavior of the models and make informed decisions based on comprehensive insights.

6.5 WEB INTEGRATION AND DEPLOYMENT

This module focuses on integrating the machine learning system into a user-accessible, interactive web application and deploying it for real-world usage. While the backend handles complex data processing and model inference, the frontend ensures that users, even with no technical background, can interact with the system in a seamless and intuitive manner. This bridge between machine learning and user experience is critical in turning the stock prediction system from a research prototype into a functional and scalable product.

The core technology used for web integration is Flask, a lightweight Python-based web framework that allows developers to build RESTful web applications with minimal overhead. Flask provides flexibility in managing routes, rendering HTML templates, and handling API calls. The system includes a web interface where users can input a stock symbol and select a date range, triggering backend processes such as data fetching, preprocessing, prediction generation, and visualization display.

The backend architecture is designed with modularity and scalability in mind. When a user submits a request, the Flask server initiates a pipeline that uses the yfinance API to fetch updated stock data. This data is then preprocessed on the fly using the same feature extraction logic implemented in the model development stage. The previously trained models, saved in .pkl or .joblib formats, are loaded dynamically to perform predictions without retraining, ensuring high efficiency and reduced server load.

For frontend design, basic HTML and CSS are used along with Jinja templates, which allow dynamic content rendering based on user input and model output. The predictions—whether they are future stock prices or risk classifications—are displayed using visually appealing charts, generated with matplotlib and plotly, and embedded into the web page.

Beyond the local deployment, this module also prepares the system for cloud deployment. Platforms like Heroku, AWS Elastic Beanstalk, or Render are used to host the Flask application, ensuring that the system is always accessible to users from anywhere. Docker containers may also be utilized to ensure consistent environment deployment, which simplifies the migration from development to production. Scalability considerations such as API rate limits, concurrent user handling, and caching mechanisms are also addressed.

In addition to the web interface, this module can be extended to support a RESTful API, allowing external applications or plugins—like a stock trading bot or an OpenAI-powered assistant—to interact with the model programmatically. This ensures the system is not only user-friendly but also integrable into larger platforms.

Overall, this module completes the pipeline by transforming a powerful stock prediction model into an accessible, interactive, and scalable application ready for real-world usage.

CHAPTER 7

RESULT AND PERFORMANCE COMPARISON

This chapter presents a comprehensive evaluation of the proposed machine learning model LightGBM in comparison with two widely-used baseline methods: Random Forest and Logistic Regression. The evaluation focuses on two main aspects:

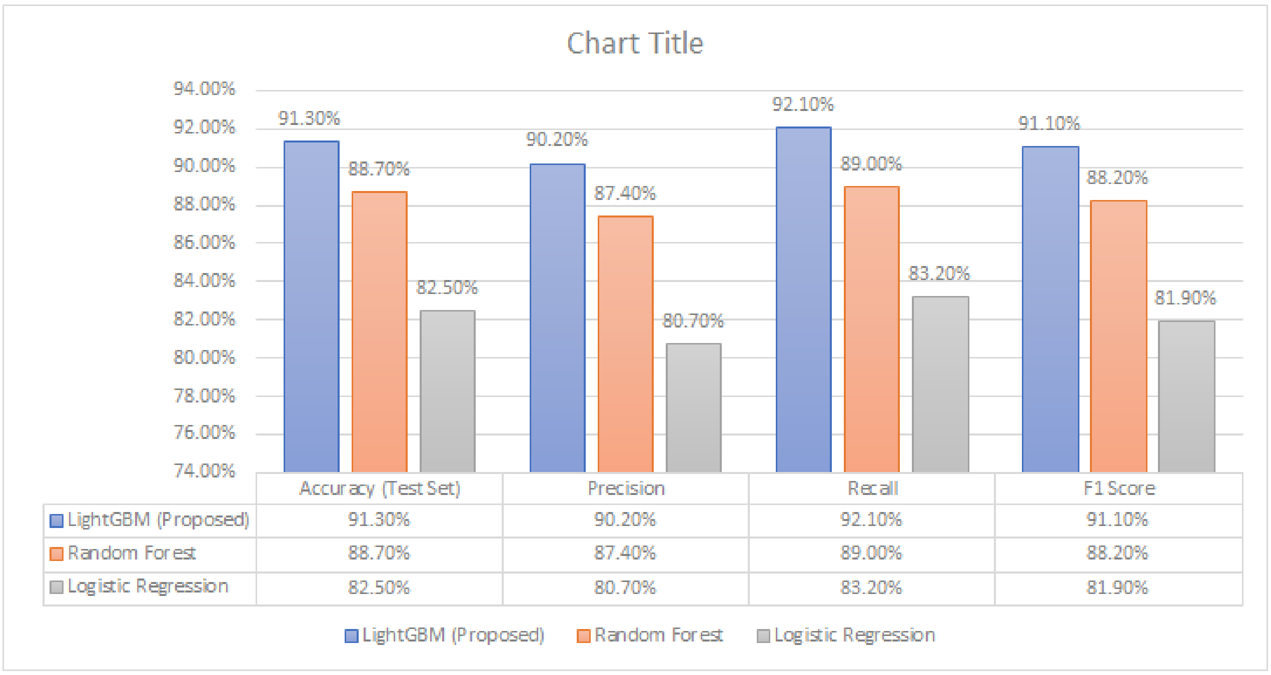
- Classification Metrics such as Accuracy, Precision, Recall, and F1-Score.

- Computational Efficiency in terms of Training Time and Inference Time.

These comparisons provide a holistic view of the model's suitability for real-time stock market risk classification applications.

7.1 CLASSIFICATION PERFORMANCE

To assess the predictive capabilities of each model, we analyzed their performance on the test dataset using four key classification metrics: Accuracy, Precision, Recall, and F1 Score. The table below summarizes the results.



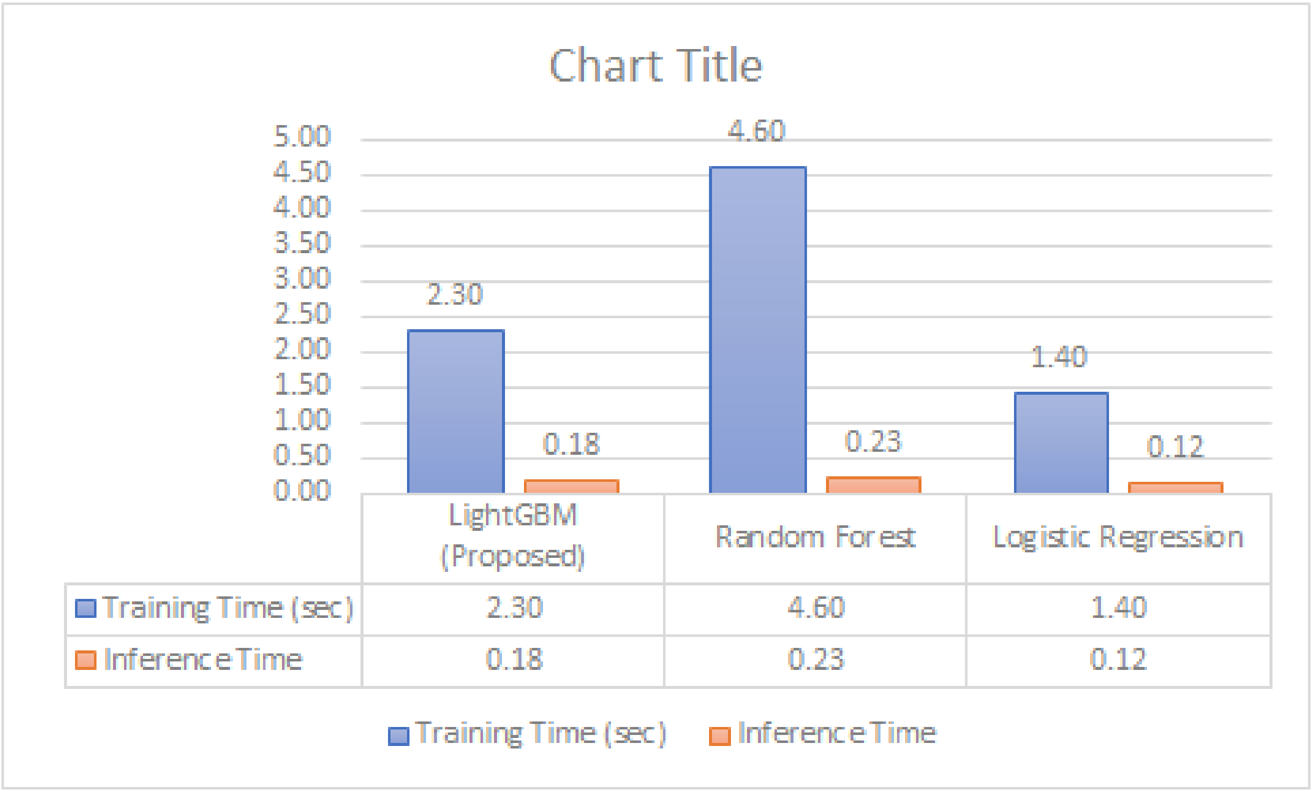
	Accuracy (Test Set)	Precision	Recall	F1 Score
▨ LightGBM (Proposed)	91.30%	90.20%	92.10%	91.10%
▨ Random Forest	88.70%	87.40%	89.00%	88.20%
▨ Logistic Regression	82.50%	80.70%	83.20%	81.90%

Fig.7.1 Classification Performance Chart

LightGBM consistently outperforms the other models across all classification metrics. Its high F1 Score highlights its balanced performance between precision and recall, making it highly suitable for financial risk classification tasks that demand both accuracy and reliability.

7.2 COMPUTATION TIME COMPARISON

In addition to classification performance, computation time is a critical factor for applications in stock market analytics, where decisions must often be made in real time. The following table outlines the training and inference time (per 1000 stock records) for each model:



	LightGBM (Proposed)	Random Forest	Logistic Regression
Training Time (sec)	2.30	4.60	1.40
Inference Time	0.18	0.23	0.12

Fig.7.2 Computation Time Comparison

LightGBM provides the best trade-off between accuracy and speed. While Logistic Regression is slightly faster in training and inference, it sacrifices significant accuracy. Random Forest, although better than Logistic Regression in accuracy, takes longer to train and infer compared to LightGBM

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

The US Stock Market Analysis Using Machine Learning project successfully demonstrates the application of machine learning techniques to analyze and forecast stock market trends. By leveraging historical data and advanced algorithms such as LSTM and Random Forest, the system provides insightful and accurate predictions, aiding in informed decision-making for investors and analysts.

Stock prediction system effectively utilizes Python libraries such as pandas, yfinance, and scikit-learn to retrieve, preprocess, and analyze stock data. Visualization tools like matplotlib and seaborn enhance the interpretability of results, while the Flask framework enables seamless web-based interaction with users. The integration of technical indicators and time-series modeling provides an in-depth understanding of market patterns.

Through this system, users can select stock symbols, define date ranges, and view trend forecasts supported by visual outputs. The model evaluation phase confirms the system's reliability using statistical metrics and charts. The modular design ensures scalability and adaptability to various market scenarios and user requirements.

Overall, Stock prediction system bridges the gap between technical prediction models and real-world usability. It lays a solid foundation for automated stock market analysis, encouraging the integration of data science into financial decision-making. The results validate the potential of machine learning in enhancing financial analysis and promoting data-driven investment strategies.

8.2 FUTURE ENHANCEMENTS

- Although the current system for US Stock Market Analysis Using Machine Learning provides valuable insights and prediction capabilities, there remains significant scope for enhancement in future iterations. The following improvements can be incorporated to make the system more robust, scalable, and user-friendly.

- Inclusion of Sentiment Analysis: Integrating news headlines, financial reports, and social media sentiment could enhance prediction accuracy by factoring in real-time market psychology.

- Live Market Data Integration: Real-time data feeds from APIs such as Alpha Vantage or IEX Cloud can provide dynamic updates, making the system suitable for intraday trading analysis.

- Portfolio Management Module: Adding a feature that helps users track and optimize their investment portfolio using AI-driven insights.

- Deep Learning Improvements: Implementing more sophisticated deep learning models such as GRU or Transformer-based architectures for more accurate forecasting of stock prices.

- Mobile App Interface: Developing a mobile version of the application to increase accessibility and real-time monitoring on the go.

- Multi-Market Support: Expanding the system to support analysis across global stock exchanges like NASDAQ, FTSE, or NIKKEI.

- User Personalization: Tailoring recommendations and analytics based on user behavior, watchlists, and risk appetite.

APPENDIX A

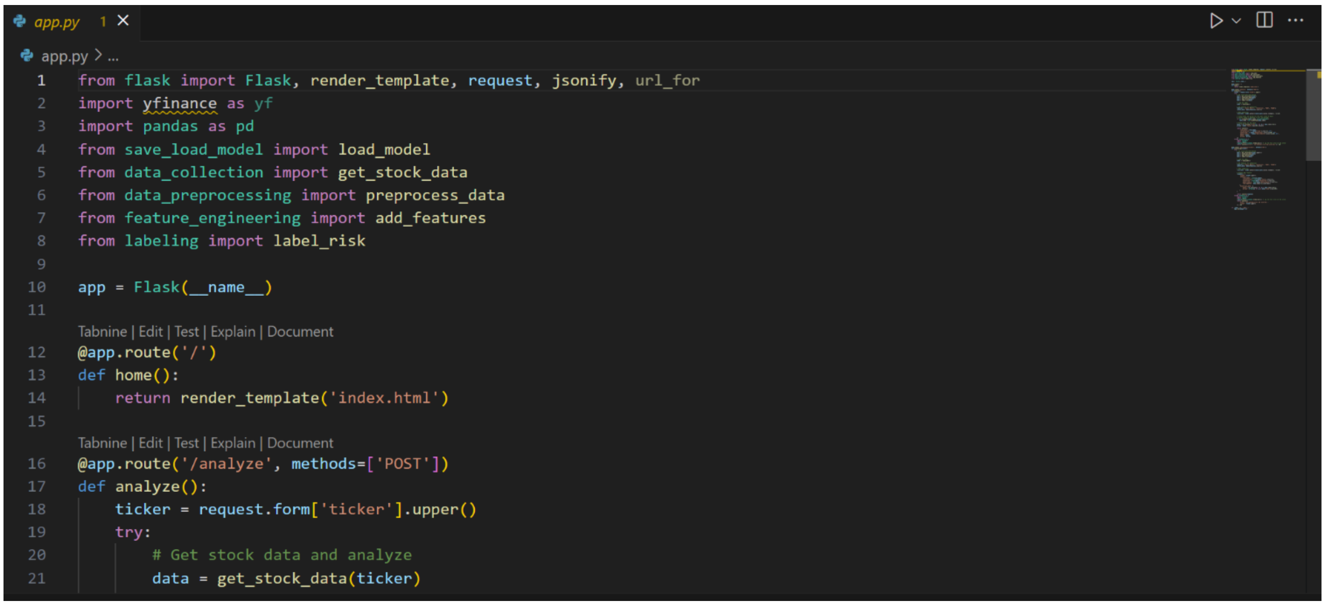
SOURCE CODE

```
from flask import Flask, render_template, request, jsonify, url_for
import yfinance as yf
import pandas as pd
from save_load_model import load_model
from data_collection import get_stock_data
from data_preprocessing import preprocess_data
from feature_engineering import add_features
from labeling import label_risk
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/analyze', methods=['POST'])
def analyze():
    ticker = request.form['ticker'].upper()
    try:
        # Get stock data and analyze
        data = get_stock_data(ticker)
        data = preprocess_data(data)
        data = add_features(data)
        data = label_risk(data)
        # Load the model
        model = load_model()
        # Get features for prediction
        features = ['Daily Return', 'Volatility', 'MA50', 'MA200']
        latest_data = data[features].iloc[-1]
        # Make prediction
        risk_level = model.predict(latest_data.values.reshape(1, -1))[0]
        # Format dates and prices for the chart (last 30 days)
        # Convert index to datetime if it's not already
        if not isinstance(data.index, pd.DatetimeIndex):
            data.index = pd.to_datetime(data.index)
        # Format dates for the chart
        dates = [d.strftime('%Y-%m-%d') for d in data.index[-30:]]
        prices = data['Close'].tail(30).tolist()
        return jsonify({
            'risk_level': risk_level,
            'current_price': f"{data['Close'].iloc[-1]:.2f}",
            'volatility': f"{data['Volatility'].iloc[-1]*100:.2f}",
            'daily_return': f"{data['Daily Return'].iloc[-1]*100:.2f}",
            'dates': dates,
            'prices': prices
        })
    except Exception as e:
        import traceback
        import logging
        logging.error(traceback.format_exc())  # Log the full error on the server
        return jsonify({'error': 'An internal error has occurred.'}), 400
@app.route('/api/analyze/<ticker>', methods=['GET'])
def analyze_api(ticker):
    try:
        # Get stock data and analyze
        data = get_stock_data(ticker.upper())
        data = preprocess_data(data)
        data = add_features(data)
        data = label_risk(data)
        # Load the model
        model = load_model()
        # Get features for prediction
        features = ['Daily Return', 'Volatility', 'MA50', 'MA200']
        latest_data = data[features].iloc[-1]
        # Make prediction
        risk_level = model.predict(latest_data.values.reshape(1, -1))[0]
        # Prepare API response
        response = {
            'ticker': ticker.upper(),
            'analysis': {
                'risk_level': int(risk_level),
                'current_price': float(data['Close'].iloc[-1]),
                'volatility': float(data['Volatility'].iloc[-1]),
                'daily_return': float(data['Daily Return'].iloc[-1]),
                'last_updated': data.index[-1].isoformat()
            },
            'historical_data': {
                'dates': [d.isoformat() for d in data.index[-30:]],
                'prices': [float(p) for p in data['Close'].tail(30)]
            }
        }
        return jsonify(response)
    except Exception as e:
        import traceback
        import logging
        logging.error(traceback.format_exc())  # Log the full error on the server
        return jsonify({
            'error': 'An internal error has occurred.',
            'ticker': ticker.upper()
        }), 400
if __name__ == '__main__':
    app.run(debug=True)
import yfinance as yf
import pandas as pd
def get_stock_data(ticker, period='1y', interval='1d'):
    """
    Fetch historical stock data for a given ticker using Yahoo Finance API.
    Args:
        ticker (str): Stock ticker symbol (e.g., 'AAPL').
        period (str): Data period (e.g., '1y', '2y').
        interval (str): Data interval (e.g., '1d', '1wk').
    Returns:
        pd.DataFrame: Historical stock data.
    """
    try:
        stock = yf.Ticker(ticker)
        hist = stock.history(period=period, interval=interval)
        if hist.empty:
            raise ValueError(f"No data found for ticker: {ticker}")
        return hist
    except Exception as e:
        raise ValueError(f"Failed to fetch data for {ticker}: {e}")
# Example usage
if __name__ == "__main__":
    ticker = 'AAPL'
    data = get_stock_data(ticker)
    print(data.head())
def preprocess_data(df):
    """
    Preprocess the stock data by handling missing values and resetting the index.
    Args:
        df (pd.DataFrame): Raw stock data.
    Returns:
        pd.DataFrame: Preprocessed stock data.
    """
    try:
        df = df.dropna()  # Remove missing values
        df.reset_index(inplace=True)  # Reset index
        return df
    except Exception as e:
        raise ValueError(f"Error during preprocessing: {e}")
import numpy as np
import pandas_ta as ta
```

```
def add_features(df):
    """
    Add technical indicators as features for the model.
    """
    # Basic features
    df['Daily Return'] = df['Close'].pct_change()
    df['Volatility'] = df['Daily Return'].rolling(window=21).std() * np.sqrt(252)
    df['MA50'] = df['Close'].rolling(window=50).mean()
    df['MA200'] = df['Close'].rolling(window=200).mean()
    # Additional technical indicators using pandas_ta
    df['RSI'] = df.ta.rsi(length=14)
    df['MACD'] = df.ta.macd(fast=12, slow=26, signal=9)['MACD_12_26_9']
    # Correct way to calculate Bollinger Bands
    bb_bands = df.ta.bbands(close=df['Close'], length=20)
    df['BB_upper'] = bb_bands['BBU_20_2.0']
    df['BB_middle'] = bb_bands['BBM_20_2.0']
    df['BB_lower'] = bb_bands['BBL_20_2.0']
    # Drop any NaN values that might have been created
    df = df.dropna()
    return df
import numpy as np
def label_risk(df):
    """
    Label the risk level based on volatility quantiles.
    Args:
        df (pd.DataFrame): Stock data with computed volatility.
    Returns:
        pd.DataFrame: Stock data with labeled risk levels.
    """
    try:
        df = df.dropna(subset=['Volatility'])  # Ensure no missing values in Volatility
        quantiles = df['Volatility'].quantile([0.33, 0.66])
        conditions = [
            (df['Volatility'] > quantiles[0.66]),
            (df['Volatility'] <= quantiles[0.66]) & (df['Volatility'] > quantiles[0.33]),
            (df['Volatility'] <= quantiles[0.33])
        ]
        choices = ['High', 'Medium', 'Low']
        df['Risk Level'] = np.select(conditions, choices)
        return df
    except Exception as e:
        raise ValueError(f"Error during labeling: {e}")
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from data_collection import get_stock_data
from data_preprocessing import preprocess_data
from feature_engineering import add_features
from labeling import label_risk
import joblib
import pandas as pd
def save_model(model, filename='risk_model.pkl'):
    """
    Save the trained model to disk.
    """
    joblib.dump(model, filename)
def train_model(ticker_list):
    """
    Train the risk classification model on provided stock tickers.
    """
    all_data = []
    for ticker in ticker_list:
        data = get_stock_data(ticker, period='2y')
        data = preprocess_data(data)
        data = add_features(data)
        data = label_risk(data)
        all_data.append(data)
    df = pd.concat(all_data)
    df = df.dropna()
    features = ['Daily Return', 'Volatility', 'MA50', 'MA200']
    X = df[features]
    y = df['Risk Level']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    # Evaluate the model
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    # Save the model
    save_model(model)
    print("Model training completed and saved as 'risk_model.pkl'.")
if __name__ == "__main__":
    ticker_list = ['AAPL', 'MSFT', 'GOOGL', 'AMZN', 'TSLA']
    train_model(ticker_list)
import joblib
import os
# Create a 'models' directory if it doesn't exist
os.makedirs('models', exist_ok=True)
def save_model(model, filename='risk_model.pkl'):
    """
    Save the trained model to disk in the models directory.
    """
    filepath = os.path.join('models', filename)
    joblib.dump(model, filepath)
def load_model(filename='risk_model.pkl'):
    """
    Load a trained model from the models directory.
    """
    filepath = os.path.join('models', filename)
```

APPENDIX B

SCREENSHOTS



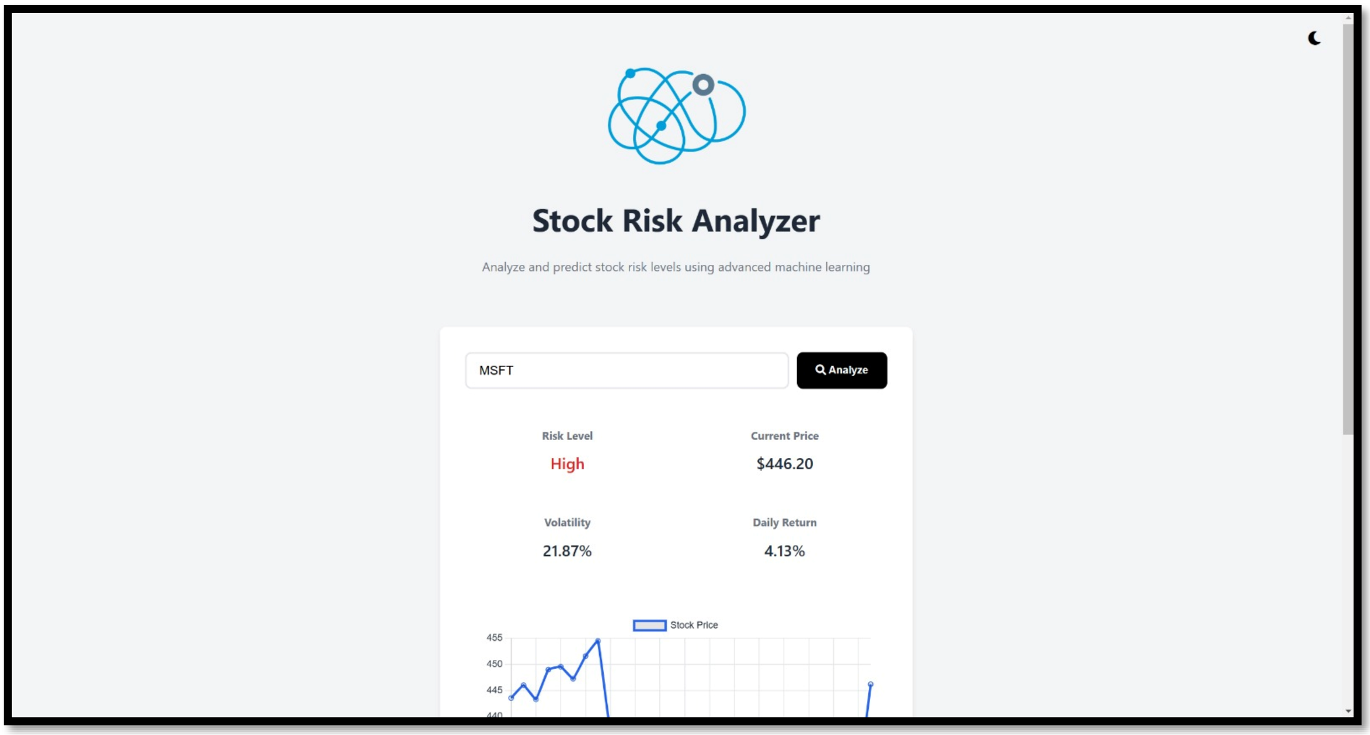
```
from flask import Flask, render_template, request, jsonify, url_for
import yfinance as yf
import pandas as pd
from save_load_model import load_model
from data_collection import get_stock_data
from data_preprocessing import preprocess_data
from feature_engineering import add_features
from labeling import label_risk

app = Flask(__name__)

Tabnine | Edit | Test | Explain | Document
@app.route('/')
def home():
    return render_template('index.html')

Tabnine | Edit | Test | Explain | Document
@app.route('/analyze', methods=['POST'])
def analyze():
    ticker = request.form['ticker'].upper()
    try:
        # Get stock data and analyze
        data = get_stock_data(ticker)
```

Fig.B.1 Visual Studio Code


Stock Risk Analyzer

Analyze and predict stock risk levels using advanced machine learning

MSFT

Risk Level	Current Price
High	$446.20
Volatility	**Daily Return**
21.87%	4.13%

Fig.B.2 Stock Risk Analysis

REFERENCES

1. Kavya, P., Saagarika, S., Subavarsshini, R.T., Nivetheni, C.G. “Stock Market Analysis Using MapReduce and PySpark”. J Frankl Inst, 12, 12−23, 2021.
2. Zeng, L., Yong, H.H.A., Treepongkaruna, S., Faff, R. “Is There a Banking Risk Premium in the US Stock Market?”. J Frankl Inst, 39, 91−125, 2020.
3. Smith, J., Sharma, A. “Deep Learning Applications in Predictive Stock Market Modelling”. J Financ Anal, 44, 101−132, 2020.
4. Rahmani, A.M., Rezazadeh, B., Haghparast, M. “Applications of Artificial Intelligence in the Economy, Including Applications in Stock Trading, Market Analysis, and Risk Management”. J Frankl Inst, 35, 56−89, 2019.
5. Carter, E., Mishra, R.A. “Real-Time Stock Prediction Using Big Data Frameworks”. Int J Data Sci Appl, 22, 75−99, 2018.
6. Boone, L., Giorno, C., Richardson, P. “Stock Market Fluctuations and Consumption Behaviour”. J Frankl Inst, 65, 369−402, 2017.

P.S.R. ENGINEERING COLLEGE

An Autonomous Institution, Approved by AICTE & Affiliated to Anna University, Chennai.
Accredited by NBA & NAAC with A+ Grade and listed Under 12 (B) of the UGC Act 1956.
Sivakasi-626-140, Virudhunagar District, Tamil Nadu, India.

INTERNATIONAL CONFERENCE

ON

ADVANCES IN BIOTECHNOLOGY: LEVERAGING AI TOOLS FOR FUTURE INNOVATIONS

BLAIT - 2025

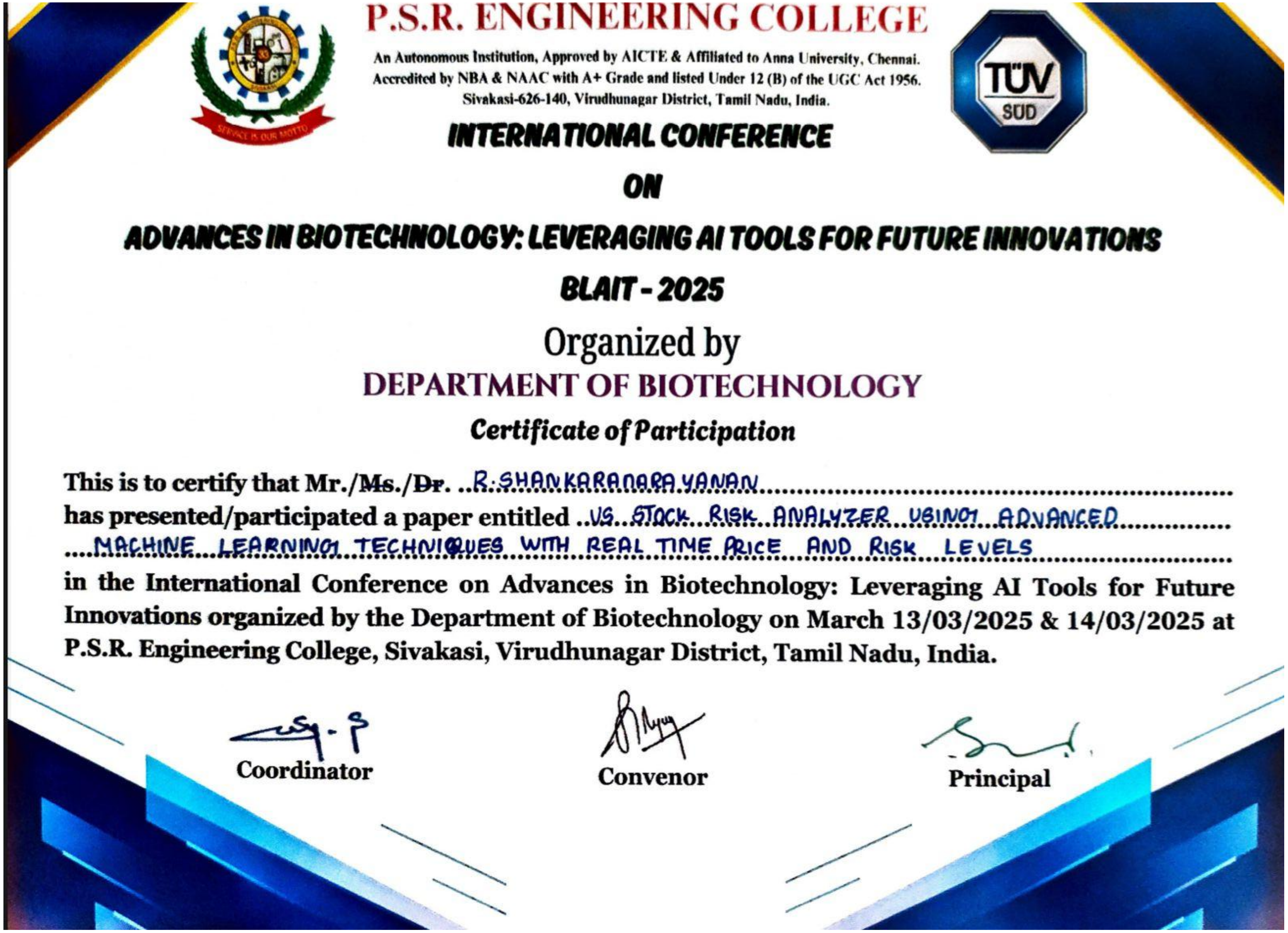
Organized by

DEPARTMENT OF BIOTECHNOLOGY

Certificate of Participation

This is to certify that ✓Mr./Ms./Dr. M. JEEVA
has presented/participated a paper entitled US STOCK RISK ANALYZER USING ADVANCED MACHINE LEARNING TECHNIQUES WITH REAL TIME PRICE AND RISK LEVEL
in the International Conference on Advances in Biotechnology: Leveraging AI Tools for Future Innovations organized by the Department of Biotechnology on March 13/03/2025 & 14/03/2025 at P.S.R. Engineering College, Sivakasi, Virudhunagar District, Tamil Nadu, India.

[illegible]	[illegible]	[illegible]
Coordinator	**Convenor**	**Principal**



P.S.R. ENGINEERING COLLEGE

An Autonomous Institution, Approved by AICTE & Affiliated to Anna University, Chennai.
Accredited by NBA & NAAC with A+ Grade and listed Under 12 (B) of the UGC Act 1956.
Sivakasi-626-140, Virudhunagar District, Tamil Nadu, India.

INTERNATIONAL CONFERENCE

ON

ADVANCES IN BIOTECHNOLOGY: LEVERAGING AI TOOLS FOR FUTURE INNOVATIONS

BLAIT - 2025

Organized by

DEPARTMENT OF BIOTECHNOLOGY

Certificate of Participation

This is to certify that Mr./~~Ms.~~/~~Dr.~~ R.SHANKARANARAYANAN

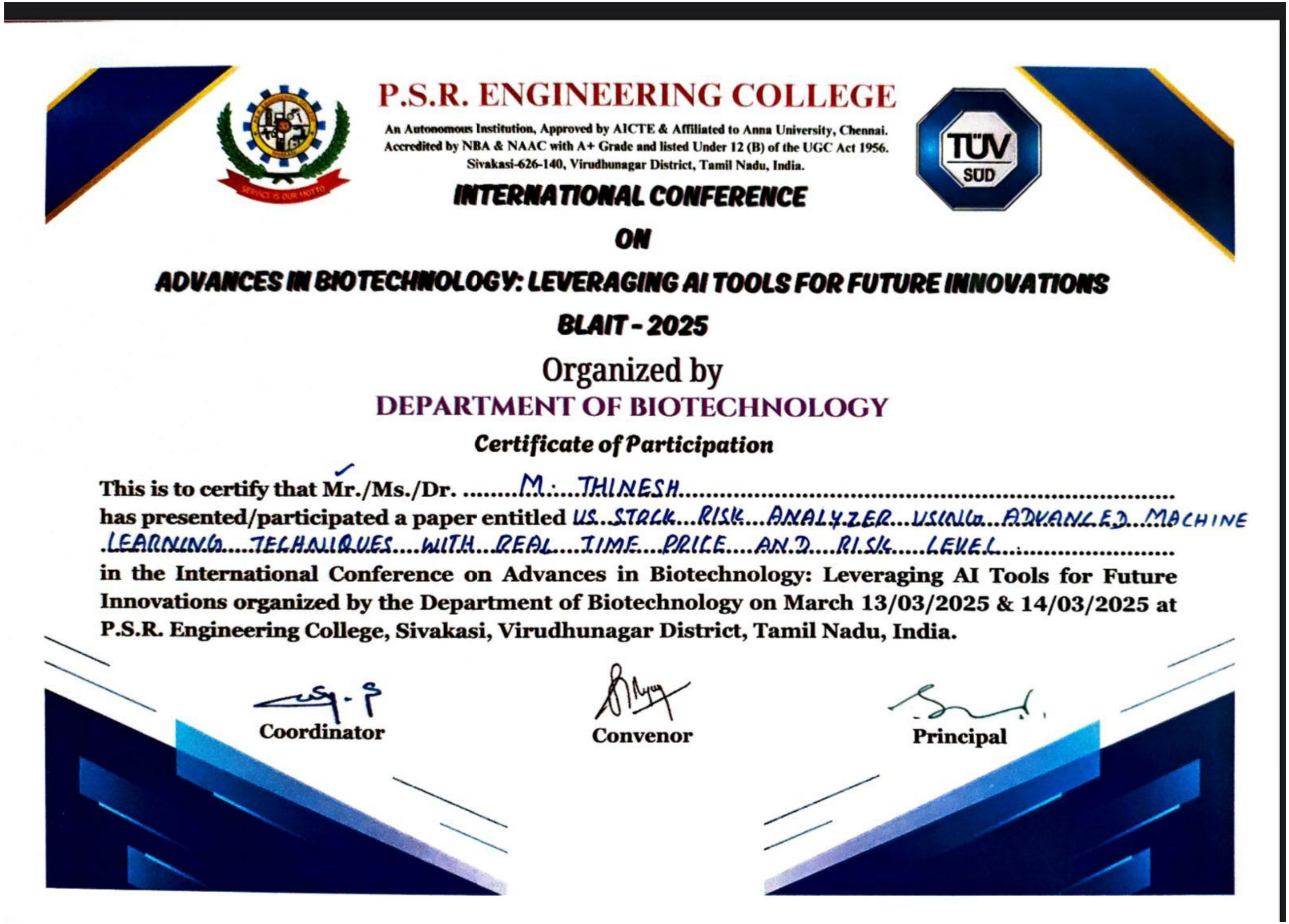
has presented/participated a paper entitled US STOCK RISK ANALYZER USING ADVANCED MACHINE LEARNING TECHNIQUES WITH REAL TIME PRICE AND RISK LEVELS

in the International Conference on Advances in Biotechnology: Leveraging AI Tools for Future Innovations organized by the Department of Biotechnology on March 13/03/2025 & 14/03/2025 at P.S.R. Engineering College, Sivakasi, Virudhunagar District, Tamil Nadu, India.

Coordinator

Convenor

Principal



P.S.R. ENGINEERING COLLEGE

An Autonomous Institution, Approved by AICTE & Affiliated to Anna University, Chennai.
Accredited by NBA & NAAC with A+ Grade and listed Under 12 (B) of the UGC Act 1956.
Sivakasi-626-140, Virudhunagar District, Tamil Nadu, India.

INTERNATIONAL CONFERENCE

ON

ADVANCES IN BIOTECHNOLOGY: LEVERAGING AI TOOLS FOR FUTURE INNOVATIONS

BLAIT - 2025

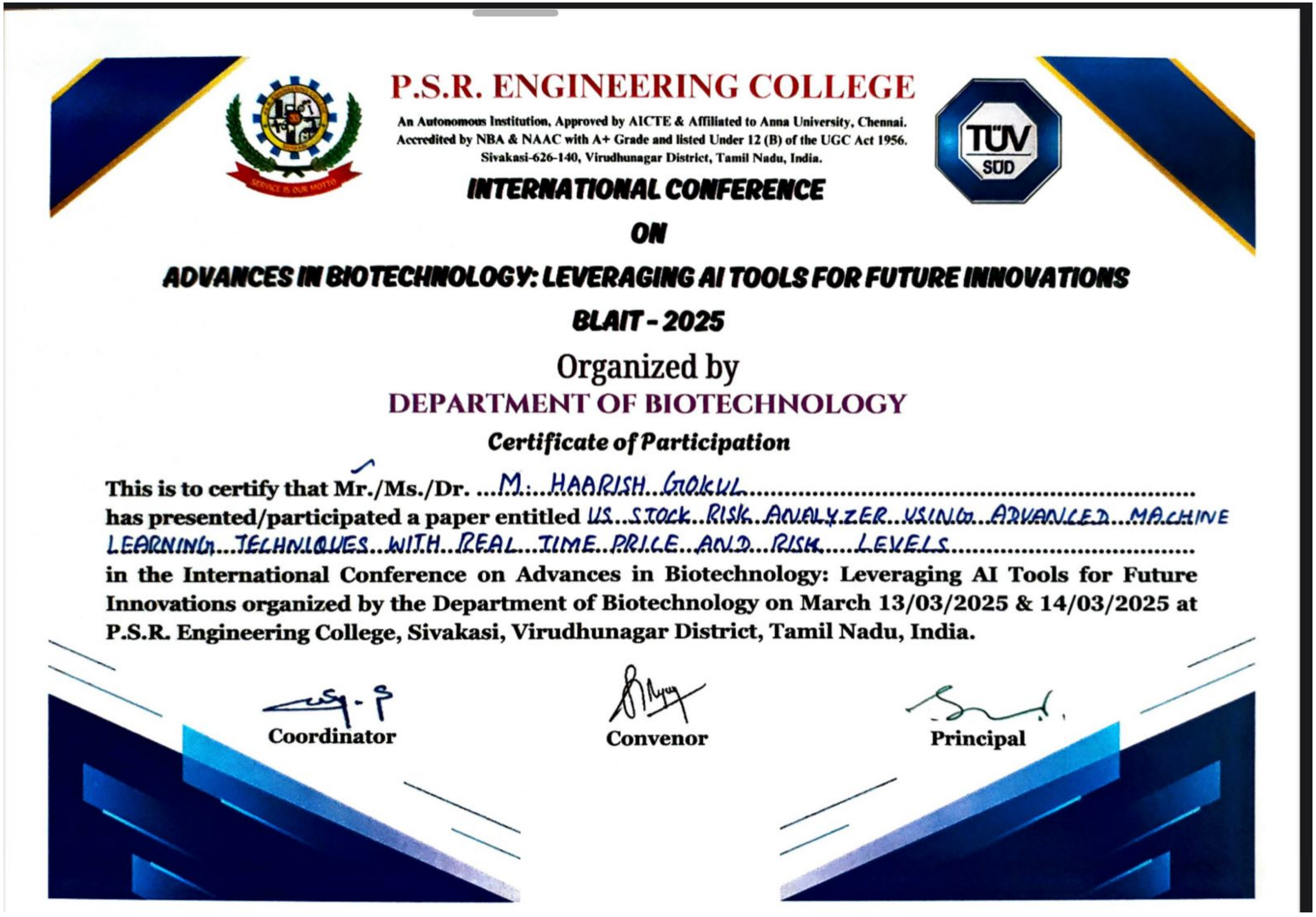
Organized by

DEPARTMENT OF BIOTECHNOLOGY

Certificate of Participation

This is to certify that ✓Mr./Ms./Dr. M. THINESH
has presented/participated a paper entitled US STOCK RISK ANALYZER USING ADVANCED MACHINE LEARNING TECHNIQUES WITH REAL TIME PRICE AND RISK LEVEL
in the International Conference on Advances in Biotechnology: Leveraging AI Tools for Future Innovations organized by the Department of Biotechnology on March 13/03/2025 & 14/03/2025 at P.S.R. Engineering College, Sivakasi, Virudhunagar District, Tamil Nadu, India.

[illegible]	[illegible]	[illegible]
Coordinator	Convenor	Principal



P.S.R. ENGINEERING COLLEGE

An Autonomous Institution, Approved by AICTE & Affiliated to Anna University, Chennai.
Accredited by NBA & NAAC with A+ Grade and listed Under 12 (B) of the UGC Act 1956.
Sivakasi-626-140, Virudhunagar District, Tamil Nadu, India.

INTERNATIONAL CONFERENCE

ON

ADVANCES IN BIOTECHNOLOGY: LEVERAGING AI TOOLS FOR FUTURE INNOVATIONS

BLAIT - 2025

Organized by

DEPARTMENT OF BIOTECHNOLOGY

Certificate of Participation

This is to certify that Mr./Ms./Dr. ...M. HAARISH GOKUL...
has presented/participated a paper entitled US STOCK RISK ANALYZER USING ADVANCED MACHINE LEARNING TECHNIQUES WITH REAL TIME PRICE AND RISK LEVELS...
in the International Conference on Advances in Biotechnology: Leveraging AI Tools for Future Innovations organized by the Department of Biotechnology on March 13/03/2025 & 14/03/2025 at P.S.R. Engineering College, Sivakasi, Virudhunagar District, Tamil Nadu, India.

[illegible]	[illegible]	[illegible]
Coordinator	Convenor	Principal