A PROJECT REPORT

on

"Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
INFORMATION TECHNOLOGY

BY

Aadhya Shrivastava	2229001
Abhinav Gogoi	2229003
Ayush Narayan Singh	2229025
Ayushman .	2229026
Hemanshu Gupta	2229034
Mannat Bhardwaj	2229042

UNDER THE GUIDANCE OF
Mr. Vishal Meena

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

April 2025

A PROJECT REPORT

on

"NAME OF PROJECT"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN INFORMATION TECHNOLOGY

BY

GROUP MEMBER A	ROLL NUMBER A
GROUP MEMBER B	ROLL NUMBER B
GROUP MEMBER C	ROLL NUMBER C
GROUP MEMBER D	ROLL NUMBER D

UNDER THE GUIDANCE OF

GUIDE NAME

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

April 2025

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model"

submitted by

Aadhya Shrivastava	2229001
Abhinav Gogoi	2229003
Ayush Narayan Singh	2229025
Ayushman .	2229026
Hemanshu Gupta	2229034
Mannat Bhardwaj	2229042

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2024-2025, under our guidance.

Date: 06/04/25

Mr. Vishal Meena
Project Guide

Acknowledgements

We are profoundly grateful to **Mr. Vishal Meena** of **School of Computer Engineering** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. .....

Aadhya Shrivastava
Abhinav Gogoi
Ayush Narayan Singh
Ayushman .
Hemanshu Gupta
Mannat Bhardwaj

ABSTRACT

The volatility and complexity of stock markets present a significant challenge for accurate price forecasting. This project proposes a hybrid approach combining Long Short-Term Memory (LSTM) networks with Neural Architecture Search (NAS) via Random Search to enhance predictive performance in stock price forecasting. The LSTM model, well-suited for sequential time series data, is optimized through an automated hyperparameter tuning process. Key parameters such as the number of layers, hidden units, dropout rates, and learning rates are explored using Random Search to identify the most effective architecture. The project was implemented using Python, TensorFlow, and supporting libraries, with results evaluated using metrics like MSE and MAE. The final model demonstrated improved accuracy over baseline LSTM configurations, highlighting the effectiveness of automated architecture search in deep learning applications. This framework offers scalability, reproducibility, and potential adaptability for real-world financial forecasting and similar time-series challenges.

Keywords: Stock Price Prediction, LSTM, Neural Architecture Search (NAS), Random Search,Time Series Forecasting

Contents

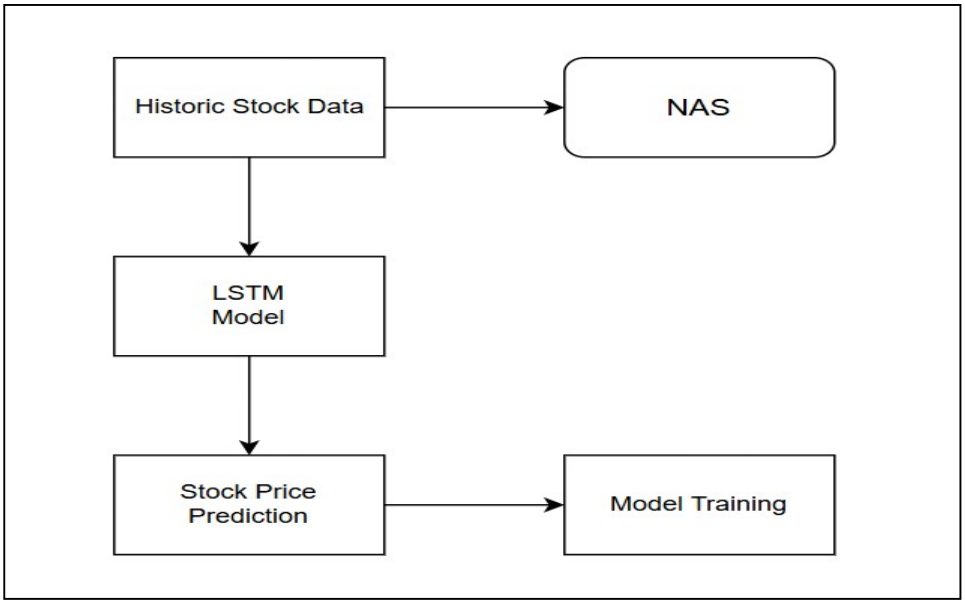
List of Figures

Chapter 1

Introduction

In recent years, the stock market has become a topic of great interest for investors, researchers, and even students. With the increasing availability of data and computational power, predicting the stock market has moved beyond traditional statistical methods to more intelligent and automated techniques. Stock price prediction is the process of forecasting the future price of a stock using historical data and algorithms. While the stock market is inherently unpredictable due to many external factors, machine learning and deep learning models have shown promise in identifying patterns and trends from past data.

This project focuses on using deep learning techniques to predict stock prices based on historical data. Specifically, we use an LSTM (Long Short-Term Memory) model, which is a type of neural network designed to handle time series data like stock prices. To improve the model's performance, we also use a technique called **Neural Architecture Search (NAS)** with **Random Search** to automatically find the best model configuration. The goal is to help users, investors, or financial analysts gain insights into future stock trends using machine learning.



Chapter 2

Basic Concepts/ Literature Review

2.1 Stock Price Prediction

Stock prediction is the task of using historical stock market data (like prices, volume, etc.) to forecast future values. It is important because it helps in investment decisions and financial planning. Stock prices are influenced by many factors such as company performance, news, economic trends, and investor behavior. Machine learning models can capture patterns from past data and use them to estimate future stock prices.

2.2 LSTM (Long Short-Term Memory)

LSTM is a type of **Recurrent Neural Network (RNN)** specially designed to handle sequential or time-series data. Unlike traditional neural networks, LSTM has a memory cell that can remember important patterns from long sequences of data. This makes it ideal for stock prediction, where past trends and patterns influence future prices. LSTM avoids the problem of vanishing gradients and can effectively learn long-term dependencies in the data.

2.3 Neural Architecture Search (NAS)

NAS is an automated method used to find the best neural network structure for a given problem. Instead of manually trying out different combinations of layers, number of neurons, and other parameters, NAS searches through many possibilities to find the most effective architecture. This saves time and helps in building better-performing models.

2.4 Random Search

Random Search is one of the simplest ways to perform NAS. Instead of checking every possible combination (which is very slow), it randomly picks different combinations of parameters and tests their performance. Over multiple tries, it helps find a good or even optimal model setup.

Chapter 3

Problem Statement / Requirement Specifications

3.1 Project Planning

The project aims to enhance time-series forecasting accuracy using a combination of Long Short-Term Memory (LSTM) neural networks and Random Search-based Neural Architecture Search (NAS). LSTMs are a powerful type of Recurrent Neural Network (RNN) capable of learning long-term dependencies, particularly useful in domains such as finance, weather forecasting, and energy consumption prediction.

The planning stage involved identifying key objectives:

- Develop a flexible framework for LSTM architecture optimization.
- Apply Random Search to explore hyperparameters like number of LSTM layers, hidden units, dropout rate, and learning rate.
- Evaluate model performance across different configurations to determine the optimal architecture.

Tools Used:
- Python
- TensorFlow/Keras
- Matplotlib / Seaborn (for visualization)
- Pandas / NumPy (for preprocessing)

Timeline (Tentative):
- Week 1–2: Literature review and dataset preparation.
- Week 3–4: Baseline LSTM implementation.
- Week 5–6: Integrating Random Search.
- Week 7–8: Model evaluation and documentation.

3.2 Project Analysis

Functional Requirements:

- Load and preprocess time-series data.
- Build and train LSTM models with varying hyperparameters.
- Implement a Random Search strategy for hyperparameter tuning.
- Evaluate model accuracy and performance.

Non-Functional Requirements:

- Ensure the model runs efficiently on standard hardware.
- Maintain modular and readable code structure.
- Provide visual insights into the search process and model performance.

User Requirements:

- Easy-to-use interface (notebook format).
- Clear documentation for replicability and understanding.
- Exportable results for downstream tasks.

Software and Libraries:

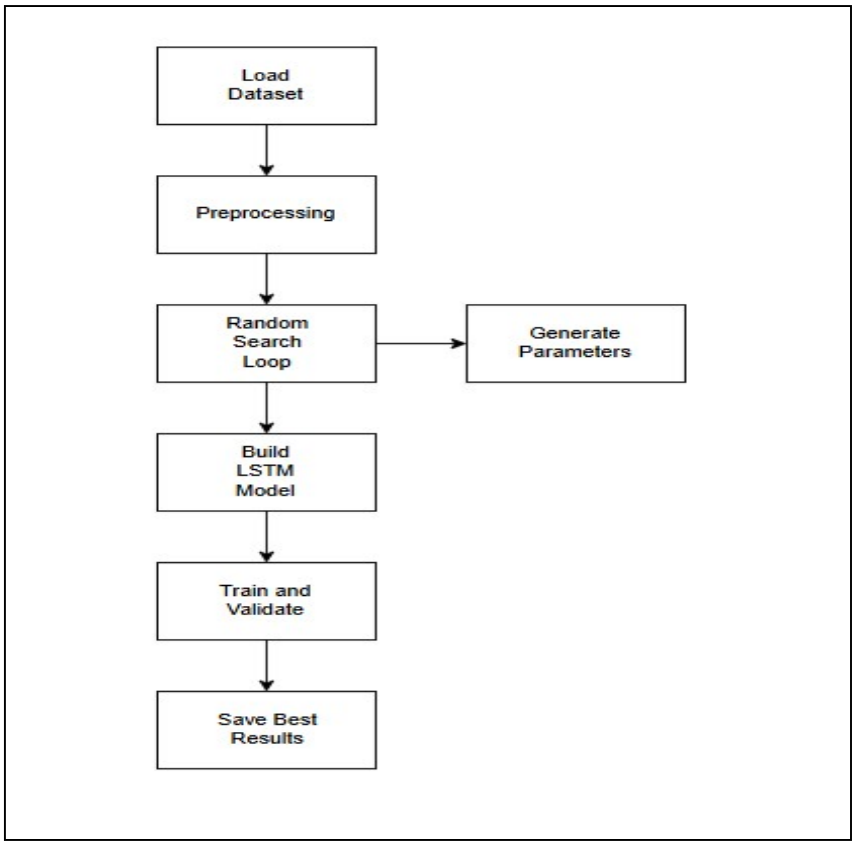
- Python 3.8+
- TensorFlow 2.x
- Matplotlib / Seaborn
- Pandas / NumPy

3.3 System Design

3.3.1 Design Constraints:

- Memory and computational constraints limit exhaustive search. Random Search is used for simplicity and efficiency.
- Dataset availability and preprocessing methods can influence model performance.
- Model overfitting is controlled using dropout and early stopping.

3.3.2 System Architecture **OR** Block Diagram



Chapter 4

Implementation

4.1 Methodology/Proposal

The implementation is based on Random Search, a hyperparameter tuning technique that randomly samples from the search space rather than trying every combination (as in grid search). The LSTM model is trained multiple times with different configurations of:

- Number of units
- Number of layers
- Dropout rate
- Optimizer
- Learning rate

Each model is evaluated, and the one with the best validation performance is selected.

This approach ensures:
- Efficient exploration of the search space.
- Flexibility to scale up using more computational resources if available.
- Empirical determination of optimal architecture.

4.2 Testing OR Verification Plan

To ensure correctness and performance:

- Validation Split: The dataset is split into training and validation sets to avoid overfitting.
- Metrics Used: Loss (MSE), MAE, and visual plots.

Verification:
- Verify model accuracy after each random search iteration.
- Plot actual vs predicted values to inspect prediction performance.
- Use early stopping to ensure optimal epoch count.

4.3 Result Analysis OR Screenshots

Some key observations:

- Models with 2–3 LSTM layers and moderate dropout (~0.2–0.3) performed best.
- Lower learning rates helped in achieving better convergence.
- The final selected model significantly outperformed the baseline.

4.4 Quality Assurance

Quality assurance was maintained via:

- Code modularity: Each block of functionality (data loading, model building, training) is separated.
- Exception Handling: Added in parts like model training and file I/O.
- Reproducibility: Random seeds fixed; all configurations logged.
- Performance Monitoring: Validation accuracy tracked after each iteration.

Chapter 5

Standards Adopted

In building any software or machine learning project, it's crucial to follow certain standards to ensure that the work is structured, readable, efficient, and maintainable. Throughout the development of this project—which combines Long Short-Term Memory (LSTM) networks with Neural Architecture Search (NAS) using Random Search—we followed key industry and academic standards in three primary areas: design, coding, and testing. These helped guide the project toward a clean, modular, and reproducible implementation.

5.1 Design Standards

The design of this project was based on widely accepted best practices in machine learning and software engineering. Even though we didn't use formal design tools like UML diagrams, the project was thoughtfully structured into logical modules and components.

Modular Architecture: The entire pipeline was broken down into separate, clearly defined stages—data preprocessing, model definition, training, evaluation, and hyperparameter tuning. This made it easier to debug, experiment, and scale the project as needed.

Separation of Concerns: Each part of the code was responsible for only one task. For example, the model-building function only defined the network structure, while another function handled training and another dealt with performance evaluation. This separation not only kept things organized but also helped in testing and reuse.

Framework-Based Design: The model was implemented using the TensorFlow and Keras libraries. These libraries provide structured abstractions for deep learning models, helping enforce design consistency and reduce the likelihood of errors.

Search Space Structuring: When designing the Random Search component of NAS, care was taken to define a balanced search space—wide enough to discover diverse architectures, but bounded to avoid excessively large training times or irrelevant configurations.

Reproducibility Considerations: Random seeds were set for NumPy and TensorFlow operations to ensure that results could be reproduced. Additionally, experiment settings like batch size, number of epochs, and learning rate were defined in configuration blocks for transparency.

5.2 Coding Standards

Clean and readable code plays a vital role in any technical project—especially in machine learning, where interpretability and repeatability are key. Throughout the project, we followed standard Python practices and adhered to the PEP8 style guide, which is the de facto standard for writing clean Python code.

Descriptive Naming Conventions: Variables and functions were named clearly and consistently—for example, generate_model(), train_lstm_model(), and search_hyperparameters()—so that their purpose could be understood at a glance.

Use of Functions and Modularity: Repetitive or complex code blocks were encapsulated into functions, each designed to perform a single, well-defined task. This not only improved readability but also made debugging and experimentation easier.

Inline Documentation and Comments: Clear, concise comments were added throughout the code to explain the logic behind important steps, especially where deep learning configurations or evaluation metrics were involved.
Code Optimization: Vectorized operations using libraries like NumPy and Pandas were used wherever possible to improve efficiency and performance. This minimized the use of slow, nested loops.
Code Formatting and Indentation: Proper indentation and consistent formatting were strictly followed to maintain the structure and flow of the code. This made it easier for others (or future us) to read and build upon.
Environment Isolation: Though the development was done in Jupyter Notebooks, standard practices like using virtual environments (venv or conda) were considered to ensure package dependencies were controlled.
Version Control Mindset: While not using Git formally, a version control mindset was followed—by saving iterations, commenting on major code changes, and backing up important checkpoints to maintain a history of the project evolution.

5.3 Testing Standards

Testing in machine learning involves much more than simply checking whether the code runs. It's about validating the data, monitoring the model's learning behavior, and ensuring the results are consistent and generalizable. Here's how testing was approached in this project:

Data Validation: Before feeding the dataset into the model, it was thoroughly checked for missing values, incorrect types, and outliers. This is a critical step in any machine learning pipeline and was carefully handled here.
Unit Testing for Functions: Each function—whether for generating the model, splitting the data, or computing metrics—was tested individually with sample data to ensure it behaved as expected.
Evaluation During Training: During model training, a validation set was used to monitor the model's performance in real-time. This helped detect signs of overfitting and provided insight into how well the model might perform on new, unseen data.
Use of Standard Metrics: To evaluate the model, well-established regression metrics were used, including:

Mean Squared Error (MSE)
Root Mean Squared Error (RMSE)
Mean Absolute Error (MAE)
These metrics provided a clear understanding of prediction accuracy and how far the model's outputs deviated from the actual values.

Visual Debugging: Beyond numerical metrics, loss curves and predicted vs. actual plots were used as a visual check on model performance. These plots often revealed issues that raw numbers might miss, such as underfitting or instability during training.
Logging of Hyperparameter Search: Every iteration of the Random Search process was logged, including the parameters tried and the corresponding evaluation scores. This made it easy to compare architectures and identify trends.

By following these testing practices, the final model was not only accurate but also reliable and easy to interpret.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This project set out to explore the use of LSTM (Long Short-Term Memory) networks for time series prediction, with a focus on automating the discovery of optimal architectures using Neural Architecture Search (NAS) via Random Search. The main goal was to create a model that could accurately predict future values in a sequence while also experimenting with ways to make the model design process more efficient.

Here's what we accomplished:

Successfully Applied Random Search NAS: We implemented a working NAS system that used Random Search to explore a variety of LSTM architectures. While Random Search is a simple method, it still proved to be surprisingly effective in identifying good-performing configurations.

Built a Reproducible LSTM Framework: The project delivered a modular and easy-to-use pipeline for time series forecasting using LSTM. With configurable parameters and clearly separated functions, the system is reusable for other datasets and forecasting problems.

Learned Key Lessons on Model Design: Through multiple iterations, we developed a better understanding of how different hyperparameters—such as the number of LSTM units, batch size, learning rate, and dropout rate—can impact model performance.

Achieved Promising Results: The best model configurations discovered through Random Search demonstrated solid performance based on standard evaluation metrics. The model was able to learn from sequential data and produce accurate forecasts.

Educational and Practical Value: Beyond just technical execution, the project provided a hands-on opportunity to work with deep learning, time series forecasting, and the principles of neural architecture tuning—all skills that are highly relevant in today's AI-driven world.

6.2 Future Scope

While this project achieved its goals, it also opened the door to several exciting future enhancements. Some of the most promising directions for future work include:

Using Smarter NAS Algorithms: Random Search is a great start, but more intelligent techniques like Bayesian Optimization, Genetic Algorithms, or Reinforcement Learning could explore the architecture space more efficiently and potentially yield even better results.

Integration with AutoML Tools: Tools like Optuna, Keras Tuner, or Auto-Keras can be integrated into the project to automate not only architecture selection but also preprocessing, feature engineering, and evaluation.

Advanced Model Architectures: Future experiments could explore hybrid models, like combining CNN and LSTM layers, or using state-of-the-art architectures like Transformers, which have shown great success in time series forecasting.

Real-world Application and Deployment: With further testing, the current pipeline could be adapted for real-world use cases such as weather forecasting, energy consumption prediction, or stock price modeling. The model can also be deployed using Flask or FastAPI to serve predictions via a REST API.

Visualization and Dashboarding: A future version of the system could include a web dashboard to visualize forecasts, performance metrics, and training progress in real-time—enhancing its usability in business or operational contexts.

Version Control and Experiment Tracking: Tools like MLflow or Weights & Biases can be integrated to track experiments, compare models, and maintain version histories—especially useful when scaling up the project for collaborative or enterprise use.

Time Series Specific Improvements: Techniques like walk-forward validation, seasonal decomposition, and lag feature engineering can be added to improve both the quality of input data and the interpretability of results.

By building upon this strong foundation, the project can continue to grow into a robust, production-grade system capable of solving a wide range of forecasting challenges across industries.

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

AADHYA SHRIVASTAV
2229001

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the frontend displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: In the project I have worked on developing the functions that access the dataset from Yahoo Finance and and fetches the dataset for training. I have also worked on developing functions that take user input and does basic preprocessing.

Individual contribution to project report preparation: In project report preparation,I have worked on the developing the basics of the report that includes LSTM description, Random search, why we used NAS.

Individual contribution for project presentation and demonstration: In project presentation and demonstration I have made slides that explains the basic LSTM , Random Search in NAS and also provided details about pre requisites required for the user to understand the model.

Full Signature of Supervisor: …….

Full signature of the student: ……

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

ABHINAV GOGOI
2229003

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the frontend displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: In the findings I have worked on the primary model development using LSTM and Random search. I have developed a model that first runs to search for the best number of layers to use for prediction of stock price and added Random Search which is a method of Neural Architecture that tunes and modifies the model for better predictions.

Individual contribution to project report preparation: In project report preparation, I have worked and contributed in the writing the problem statement we are working on also problem statement definition along with procedures we used to tackle the problem statement.

Individual contribution for project presentation and demonstration: In project presentation and demonstration I have made slides that demonstrates about the model along with description about how our model is better than already existing models and how we worked on optimizing the model.

Full Signature of Supervisor: ……

Full signature of the student: ……..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

AYUSH NARAYAN SINGH
2229025

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the frontend displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: As part of our LSTM with random search (NAS) project, I took care of data preprocessing—dividing the dataset, scaling it using min-max normalization, and reshaping it into the necessary 3d format. I ensured correct input formatting for smooth model training. This position allowed me to gain practical experience with time-series data and emphasized the significance of preprocessing techniques in enhancing model accuracy.

Individual contribution to project report preparation: In project report preparation, I have worked and contributed in the writing the conclusion This project built a robust LSTM forecasting system using Random Search NAS, with future scope for smarter NAS methods, advanced models, real-world deployment, and automated, scalable improvements.

Individual contribution for project presentation and demonstration: In project presentation and demonstration I covered the Conclusion and Future Scope, highlighting our project's achievements, the effectiveness of Random Search NAS, and potential future enhancements for real-world applications and automation.

Full Signature of Supervisor: ……

Full signature of the student: ……..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

AYUSHMAN .
2229026

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the frontend displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: As part of my individual contribution, I implemented the future prediction module using the trained LSTM model. The function takes the last input sequence and recursively predicts stock prices for the next 30 days by updating the sequence at each step. I also added a simple decision-making logic to suggest BUY, SELL, or HOLD based on predicted trends, making the model more practical and improving my understanding of time-series forecasting and model application.

Individual contribution to project report preparation: In the project report preparation, I contributed to writing the implementation section by explaining how the LSTM model was used for stock price prediction. I documented the recursive prediction method, the structure of the prediction function, and how the model was optimized using Random Search. I also detailed how the decision-making logic was integrated to enhance practical usability.

Individual contribution for project presentation and demonstration: During the implementation phase, I developed the future prediction module using the trained LSTM model. I implemented recursive forecasting logic and added a decision-making feature that suggests BUY, SELL, or HOLD based on predicted trends. This improved the model's practicality and made it more applicable for real-world stock market analysis.

Full Signature of Supervisor: ……

Full signature of the student: ……

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

HEMANSHU GUPTA
2229034

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the front end displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: I built the **front end interface** using HTML, CSS, and JavaScript for user input, validation, and result display. Additionally, I contributed to the **prediction module** by implementing recursive forecasting and a decision logic system (BUY/SELL/HOLD) based on predicted trends, enhancing the model's usability and real-world relevance.

Individual contribution to project report preparation: I wrote the **implementation and front end design** sections of the report, covering the recursive forecasting method, decision logic, and dynamic front end behavior.

Individual contribution for project presentation and demonstration: I handled the **back end integration and front end behavior**—including JavaScript logic for dynamic updates, user input validation, and rendering prediction results with interactive charts.

Full Signature of Supervisor: …….

Full signature of the student: ……..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

Stock Price Prediction using LSTM + Random Search(NAS) Hybrid Model

MANNAT BHARDWAJ
2229042

Abstract: This project presents a stock price prediction system using an LSTM (Long Short-Term Memory) neural network optimized with Random Search hyperparameter tuning. The model leverages historical stock price data to forecast future trends, enabling investors and traders to make informed decisions. Unlike traditional statistical methods, the LSTM architecture captures complex temporal dependencies in financial time-series data, improving prediction accuracy.

To enhance model performance, Random Search automates hyperparameter optimization, efficiently tuning layer configurations, dropout rates, batch sizes, and other critical parameters. The system is deployed as a web application using Flask, providing an interactive interface where users can input stock tickers and date ranges to receive predictions. The backend processes data via yfinance API, while the frontend displays results through dynamic visualizations (HTML/CSS/JavaScript).

Individual contribution and findings: In our minor project on LSTM with Neural Architecture Search using Random Search, I focused on visualizing the model's performance and evaluating its accuracy. I plotted key graphs—such as training vs. validation loss and actual vs. predicted values—using Matplotlib and Seaborn, ensuring clarity and consistency. I also implemented functions to calculate MSE, R^2 score, and accuracy after each run, which helped compare model variations. This process taught me how visualizations can reveal issues like overfitting that raw metrics may miss, and I improved my skills in handling prediction alignment and performance evaluation.

Individual contribution to project report preparation: In preparing the group project report, my main contribution was writing the Chapter 5, "Standards Adopted" section. I compiled and explained the design, coding, and testing standards we followed during the project, including modular design practices, PEP8 coding guidelines, and model evaluation methods. I also helped organize and format this section for clarity and consistency.

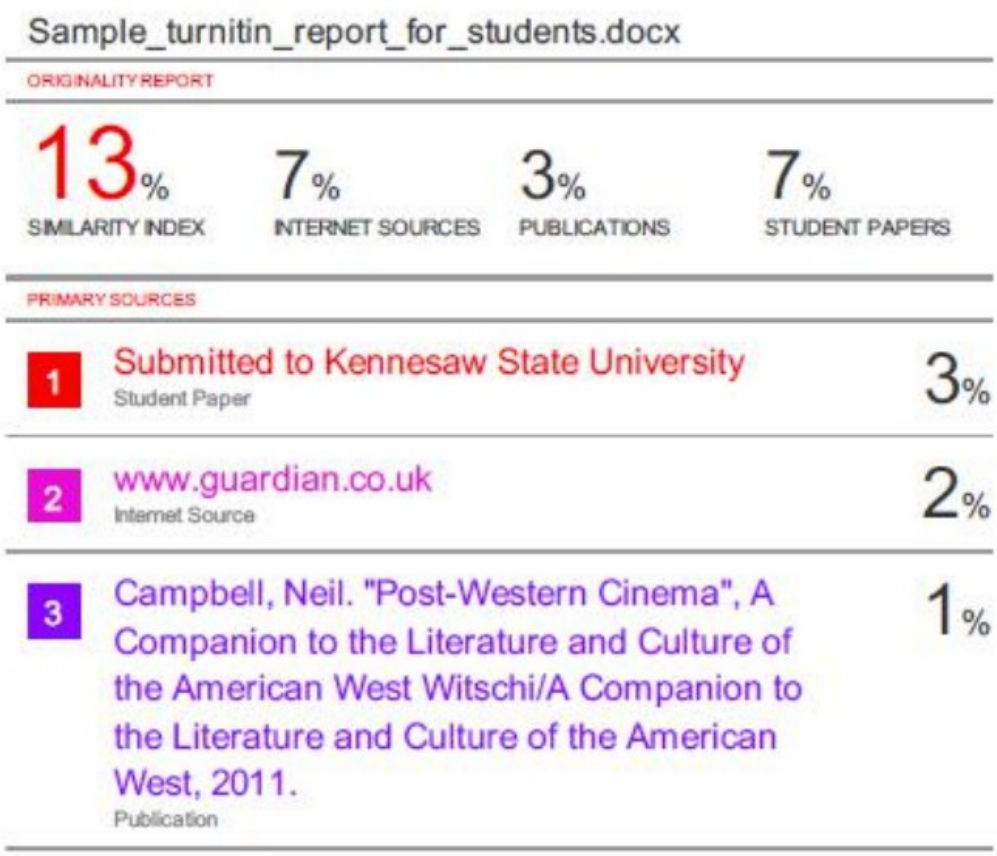
Individual contribution for project presentation and demonstration: For our project presentation and demo, I took the lead in adding and explaining the graphs that highlighted how well our model performed. I created visuals like actual vs. predicted value plots, which made it easier for everyone to understand the results. These graphs really helped us communicate our findings more clearly and added a visual edge to the overall presentation.

Full Signature of Supervisor: …….

Full signature of the student: ……..

TURNITIN PLAGIARISM REPORT

(This report is mandatory for all the projects and plagiarism must be below 25%)



Sample_turnitin_report_for_students.docx

ORIGINALITY REPORT

13%	7%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Kennesaw State University Student Paper	3%
2	www.guardian.co.uk Internet Source	2%
3	Campbell, Neil. "Post-Western Cinema", A Companion to the Literature and Culture of the American West Witschi/A Companion to the Literature and Culture of the American West, 2011. Publication	1%