Data Analysis Using AI ML Techniques in Indian Financial Markets

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

ANGAD MAKARAND CHITALE (21BEC2365)

PURANDARE AMEYA YOGESH (21BEC2393)

Under the guidance of

PROFESSOR PRAKASAM P

NOVEMBER, 2024

DECLARATION

I hereby declare that the thesis entitled "Data Analysis Using AIML Techniques in Indian Financial Markets " submitted by me, for the completion of the course "BECE497J – Project 1" to the school of electronics engineering, vellore institute of technology, vellore is bonafide work carried out by me under the supervision of Professor Prakasam P.

I further declare that the work reported in this thesis has not been submitted previously to this institute or anywhere.

Place: Vellore

Date: 13-11-2024

Signature of the Candidate
Purandare Ameya Yogesh
21BEC2393

Signature of the Candidate
Angad Makarand Chitale
21BEC2365

CERTIFICATE

This is to certify that the thesis entitled "Data Analysis Using AIML Techniques in Indian Financial Markets" submitted by **Purandare Ameya Yogesh, 21BEC2393, Angad Makarand Chitale, 21BEC2365**, SENSE School, VIT, for the completion of the course "BECE497J – Project 1", is a bonafide work carried out by him / her under my supervision during the period, 01. 09. 2024 to 05.11.2024, as perthe VIT code of academic and research ethics.

I further declare that the work reported in this thesis has not been submitted previously to this institute or anywhere.

Place : Vellore
Date : 13-11-2024

Signature of the Guide

Internal Examiner

ACKNOWLEDGEMENTS

We would like to extend our heartfelt gratitude to everyone who contributed to the successful completion of our project, *Data Analysis Using AI/ML Techniques in Indian Financial Markets*. First and foremost, we are immensely grateful to our project guide, Professor Prakasam P., whose expertise, guidance, and continuous encouragement helped us navigate through the complexities of financial modeling and machine learning. His valuable insights were crucial in shaping our approach and understanding of the subject.

We would also like to express our appreciation to our institution's Electronics and Communication Engineering Department for providing access to the resources and computational facilities that enabled the development and testing of our models. The support from our peers and colleagues, who offered feedback and shared insights, enriched our learning experience and motivated us to strive for excellence.

Lastly, we are thankful to our families for their unwavering support and understanding throughout the project duration. Their patience and encouragement were a constant source of strength. This project is a product of collaborative effort, and we are deeply appreciative of the collective guidance, support, and motivation from everyone involved. Thank you for making this journey a rewarding and memorable experience.

Student Name

Angad Makarand Chitale

Purandare Ameya Yogesh

Executive Summary

This project investigates the effectiveness of various machine learning models in predicting stock prices within the Indian financial markets, specifically using data from the NIFTY 50 index. Traditional forecasting models like ARIMA have limitations in handling the volatility and nonlinear patterns common in financial data. To address these challenges, this study compares four models: Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Network (RNN). Each model is evaluated on metrics such as Absolute Percentage Error (APE), Average Absolute Error (AAE), Average Relative Percentage Error (ARPE), and Root Mean Squared Error (RMSE), providing insights into their predictive capabilities.

The results reveal that HMM outperforms the other models, particularly in capturing long-term market trends, making it well-suited for volatile market conditions. This project not only contributes to the field of financial forecasting by identifying a robust predictive model but also aligns with Sustainable Development Goals (SDGs) by promoting economic growth, innovation, and reduced inequalities. Future improvements include integrating hybrid models and additional financial indicators for enhanced accuracy.

CONTENTS

.

APPENDIX A

List of Figures

List of Tables

List of Abbreviations

LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
HMM	Hidden Markov Model
ARIMA	Autoregressive Integrated moving Average

Symbols and Notations

δf	CFO
ε	NCFO

1. INTRODUCTION

1.1. OBJECTIVE

In financial markets, predicting stock price movements is a critical and challenging task due to high volatility, especially in emerging economies like India. Traditional forecasting models, such as Autoregressive Integrated Moving Average (ARIMA), have historically been used for time-series predictions. However, these models struggle to account for nonlinear patterns and sudden market shifts, limiting their effectiveness. Modern AI and ML techniques offer new avenues for developing more sophisticated and accurate forecasting models by capturing complex dependencies within financial data.

Literature Review

Recent studies highlight various machine learning approaches to financial market prediction. Bouktif et al. (2020) investigated stock market predictability using an enhanced sentiment analysis method, where customized text-based features and n-grams were employed to analyse sentiments around stock movements. Another study by Liu et al. (2020) proposed a hierarchical attention capsule network to quantify the influence of social media and news, yielding significant improvements in prediction accuracy. Wang et al. (2019) introduced a hybrid time-series neural network that combined news sentiment with stock data, showing notable advantages in speed and prediction accuracy.

These studies emphasize the growing role of AI/ML in financial predictions, yet gaps remain. Current models, although innovative, often fall short in handling the dynamic and unpredictable nature of the Indian financial market. Models focusing on isolated data sources, such as a single news outlet or limited company stocks, limit their application across diverse markets. Additionally, there is limited comparative research on combining multiple advanced models for holistic accuracy.

Research Gap

The project aims to address several gaps: 1) the need for models that perform consistently across various economic conditions, 2) the integration of multiple evaluation metrics to understand model strengths better, and 3) improving generalizability to real-world data. This gap in research represents a barrier to achieving reliable predictions in volatile markets and impacts the ability of investors to make informed decisions.

Problem Statement

Stock price prediction in India's NIFTY 50 index is complex due to market unpredictability and sudden fluctuations, limiting the effectiveness of traditional and single-model approaches. This project seeks to implement a comparative framework to evaluate the performance of four distinct predictive models: Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), ARIMA, and Recurrent Neural Network (RNN). By comparing these models across multiple metrics, this study aims to provide insights into each model's effectiveness and suitability, ultimately informing more accurate, risk-averse financial decisions for investors.

Relevance of the Problem w.r.t SDG

The relevance of this project is closely aligned with several Sustainable Development Goals (SDGs). First, it supports **SDG 8: Decent Work and Economic Growth** by fostering financial stability through more reliable predictions. Improved forecasting contributes to job security in sectors influenced by stock market health.
Additionally, **SDG 9: Industry, Innovation, and Infrastructure** is promoted as this research utilizes innovative AI/ML techniques, pushing the boundaries of financial

technology. Furthermore, by democratizing access to sophisticated financial tools, this project indirectly aligns with SDG 10: Reduced Inequalities by enabling smaller investors to compete with larger institutions using advanced predictive insights.

2. Project Objective

The primary objective of this project is to systematically compare the forecasting abilities of four AI/ML models—HMM, LSTM, ARIMA, and RNN—on the NIFTY 50 index. These models were chosen for their unique approach to time-series data: ARIMA offers traditional statistical methods, while LSTM and RNN provide deep learning capabilities for sequential data. HMM, a probabilistic model, is valuable for identifying hidden states, such as market phases. This research aims to reveal which model or combination of models performs best in terms of accuracy, computational efficiency, and robustness in various financial scenarios.

Specific objectives include:

- Implementing and training each model on historical NIFTY 50 data from 2000 to 2021.
- Using diverse evaluation metrics, such as Absolute Percentage Error (APE) and Root Mean Squared Error (RMSE), to comprehensively assess model performance.
- Identifying strengths and limitations within each model to guide future improvements.
- Contributing to the field of financial forecasting through rigorous analysis and potentially publishing findings in academic conferences.

3. Proposed Work

This section outlines the design approach, data preprocessing, model implementation, and evaluation.

Design Approach

The project is designed to evaluate stock prediction models on historical NIFTY 50 data, examining the accuracy and reliability of each model. The approach includes data collection, preprocessing, model training, and performance comparison. Preprocessing involves data cleaning and handling missing values, followed by standardizing data formats for each model.

Models and Algorithms

1. Hidden Markov Model (HMM): A probabilistic approach that models hidden states in stock trends, HMM can capture different market phases such as bullish and bearish trends.
2. LSTM (Long Short-Term Memory): A deep learning model well-suited for capturing long-term dependencies, ideal for sequential stock price data.
3. ARIMA (Autoregressive Integrated Moving Average): A classical time-series model that effectively forecasts trend-based, stable data but may lack precision in highly volatile scenarios.
4. RNN (Recurrent Neural Network): RNN handles sequential data and is optimized for shorter-term dependencies, making it efficient for certain prediction timeframes.

Technical Description

The technical setup includes using Python libraries such as NumPy, TensorFlow, and Stats Models for model implementation. Each model is trained on the same dataset and subjected to cross-validation. Out-of-sample testing ensures fairness in comparing predictive performance, and error metrics evaluate the strengths of each approach. Specific techniques include parameter tuning for each model, cross-validation to assess robustness, and out-of-sample testing for unbiased predictions.

```
# Scaling the data to the range (0, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(obs['Close'].values.reshape(-1, 1))

# Prepare data for LSTM model (using the first D months)
def prepare_data(data, time_step):
    X, y = [], []
    for i in range(len(data) - time_step):
        X.append(data[i:(i + time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = obs.shape[0] - D

# Initialize an empty list to store the predictions
lstm_predictions = []

# Initial training on the first 162 months
X_train, y_train = prepare_data(scaled_data, D)

# Reshape input to be [samples, time steps, features] for LSTM
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

# Train the LSTM model
model_lstm = Sequential()
model_lstm.add(LSTM(units=50, return_sequences=True, input_shape=(time_step, 1)))
model_lstm.add(LSTM(units=50))
model_lstm.add(Dense(1))
model_lstm.compile(optimizer='adam', loss='mean_squared_error')
model_lstm.fit(X_train, y_train, epochs=5, batch_size=32)

# Iteratively predict the next 96 months and retrain the model
for i in range(96):  # Predict the next 96 months
    X_input = scaled_data[-time_step:]  # Use the last 'time_step' months for prediction
    X_input = X_input.reshape(1, time_step, 1)
    pred = model_lstm.predict(X_input)
    lstm_predictions.append(pred[0, 0])

    # Add the predicted data to the training set
    new_data = np.append(scaled_data[:time_step + i + 1], pred)
    scaled_data = np.append(scaled_data, pred).reshape(-1, 1)

    # Re-prepare the training data including the new data
    X_train, y_train = prepare_data(scaled_data[:time_step + i + 1], time_step)
    X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

    # Retrain the LSTM model with the updated data
    model_lstm.fit(X_train, y_train, epochs=5, batch_size=32)

# Inverse transform the predictions to the original scale
lstm_predictions = scaler.inverse_transform(np.array(lstm_predictions).reshape(-1, 1))

# LSTM predictions for the next 96 months
print(lstm_predictions.shape)
```

LSTM Model Code

```
# # AutoRegressive Integrated Moving Average (ARIMA) Model

# # Use the 'Close' prices for ARIMA model
# arima_data = obs['Close']
# arima_predictions = []
# D = 96
# time_step = obs.shape[0] - D

# for i in range(D):
#     # Define the ARIMA model (order can be adjusted based on ACF/PACF plots or a grid search)
#     print(obs.iloc[time_step+i-1]['Close'])
#     model_arima = ARIMA(arima_data[:time_step + i - 1], order=(3, 2, 0))
#     model_arima_fit = model_arima.fit()

#     # Forecast the next 96 months
#     arima_prediction = model_arima_fit.forecast(steps=1)
#     arima_predictions.append(arima_prediction)


# # Convert predictions to a numpy array
# arima_predictions = np.array(arima_predictions).reshape(-1, 1)

# print(arima_predictions.shape)
```

```
# Use auto_arima to find the best ARIMA model parameters

from pmdarima import auto_arima

arima_data = obs['Close']
arima_predictions = []
D = 96
time_step = obs.shape[0] - D

for i in range(D):

    # Fit auto_arima model
    model_auto_arima = auto_arima(
        arima_data[:time_step + i - 1],  # Data up to current index
        seasonal=True,              # Set to True if your data is seasonal
        trace=True,                 # Display the search progress
        error_action='ignore',      # Ignore errors to continue with the next iteration
        suppress_warnings=True,     # Suppress warnings during the model fitting
        stepwise=True               # Use stepwise approach to find the best model
    )

    # Forecast the next step
    arima_prediction = model_auto_arima.predict(n_periods=1)
    arima_predictions.append(arima_prediction)

# Convert predictions to a numpy array
arima_predictions = np.array(arima_predictions).reshape(-1, 1)

print(arima_predictions.shape)
```

ARIMA Model Code

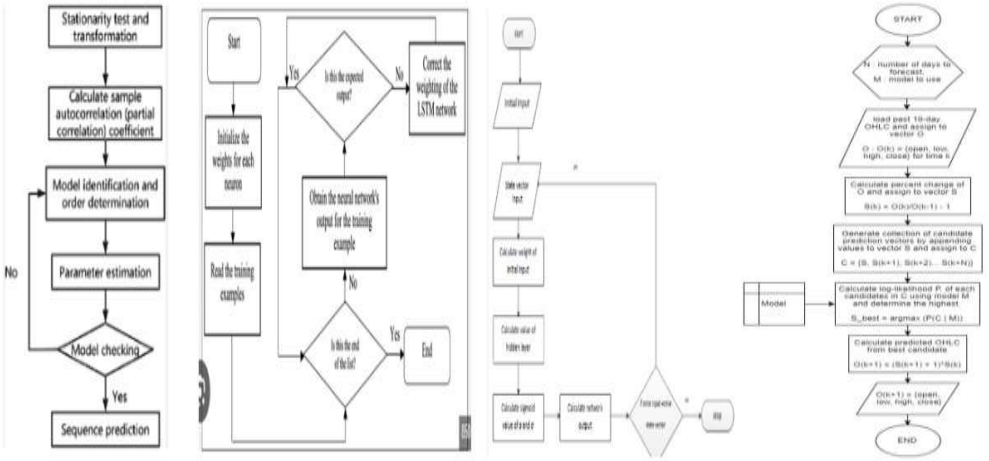
```python
# Scaling the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(obs['Close'].values.reshape(-1, 1))

# Prepare data function for RNN model
def prepare_data(data, time_step):
    X, y = [], []
    for i in range(len(data) - time_step):
        X.append(data[i:(i + time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)

time_step = obs.shape[0] - D  # D is the training window size

# Initialize an empty list to store the predictions
rnn_predictions = []

# Initial training on the first 162 months
X_train, y_train = prepare_data(scaled_data, time_step)

print(X_train.shape)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

# Train the initial RNN model
model_rnn = Sequential()
model_rnn.add(SimpleRNN(units=50, return_sequences=True, input_shape=(time_step, 1)))
model_rnn.add(SimpleRNN(units=50))
model_rnn.add(Dense(1))
model_rnn.compile(optimizer='adam', loss='mean_squared_error')
model_rnn.fit(X_train, y_train, epochs=5, batch_size=32)

# Iteratively predict the next month and retrain the model
for i in range(96):  # Predict the next 96 months iteratively
    # Predict the next month
    X_input = scaled_data[-time_step:]  # Use the last 'time_step' months for prediction
    X_input = X_input.reshape(1, time_step, 1)
    pred = model_rnn.predict(X_input)
    rnn_predictions.append(pred[0, 0])

    # Add the real data for the next month to the training set
    new_data = np.append(scaled_data[:time_step + i + 1], pred)
    scaled_data = np.append(scaled_data, pred).reshape(-1, 1)

    # Re-prepare the training data including the new data
    X_train, y_train = prepare_data(scaled_data[:time_step + i + 1], time_step)
    X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)

    # Retrain the RNN model with the updated data
    model_rnn.fit(X_train, y_train, epochs=5, batch_size=32)

# Inverse transform the predictions to the original scale
rnn_predictions = scaler.inverse_transform(np.array(rnn_predictions).reshape(-1, 1))

# rnn_predictions now contains the predicted closing prices for the next 96 months
print(rnn_predictions.shape)
```

RNN Model Code

```python
data = data[data.columns[1:5]]
obs = obs[obs.columns[1:5]]
# Calculate number of rows and set training window
T = data.shape[0]
print("T= ", T)

# Define the size of the training window
d = 96
D = 96
hmm_price = []

temp_T = T
first_time = True

# Sliding window approach to predict future prices
while T < temp_T + d:

    # Train HMM on data from T-D+1 to T
    train_data = obs.iloc[T-D:T]
    train_data = train_data.dropna()

    # Set the random seed
    np.random.seed(123)

    if(first_time):
      first_time = False
      model = hmm.GaussianHMM(n_components=5)
    else:
      old_model= model
      model = hmm.GaussianHMM(n_components=5, init_params="c")
      model.startprob_ = old_model.startprob_
      model.transmat_ = old_model.transmat_
      model.means_ = old_model.means_

    model.fit(train_data)

    # Calculate original likelihood
    original_likelihood = model.score(train_data)

    # Loop to find new likelihood
    t=T
    min_diff = float('inf')
    min_t = T
    min_likelihood = original_likelihood
    while t>D  0:
        t = t-1

        train_data = obs.iloc[t-D:t]
        new_likelihood = model.score(train_data)
        if (abs(new_likelihood - original_likelihood)< min_diff:  # Threshold for comparison by choosing that new_likeliho
            min_diff = abs(new_likelihood - original_likelihood)
            min_t = t
            min_likelihood = new_likelihood

    # Calculate the predicted close price
    close_price = obs['Close'][T-1] + ((obs['Close'][min_t + 1] - obs['Close'][min_t]) * np.sign(original_likelihood - min_

    hmm_price.append(close_price)
    T=T+1

# Print the calculated prices
print("HMM Prices: ")
print(hmm_price)

# Plot the predicted and observed prices
close = []
truncated_obs = obs.iloc[T-d:T]

for i in truncated_obs['Close']:
  close.append(i)

# plt.plot(hmm_price,marker='.', label = "HMM Predicted Price")
# plt.plot(close,marker = '.', label= "Observed Price")
# plt.ylabel("Close Price")
# plt.legend()
# plt.show()
```

HMM Model Code

3.1 ARIMA: 3.2 LSTM 3.3 RNN 3.4 HMM



Flow Chart of all models

Data Cleaning Processes:

1.1 Data before preprocessing:

	Date	Open	High	Low	Close	Volume	Turnover	P/E	P/B	Div Yield
2	1/3/2000	1482.15	1592.9	1482.15	1592.2	2535932.2	8.84E+09	25.91	4.63	0.95
3	1/4/2000	1594.4	1641.95	1594.4	1638.7	38787872	1.97E+10	26.67	4.76	0.92
4	1/5/2000	1634.55	1635.5	1555.05	1595.8	62153431	3.08E+10	25.97	4.64	0.95
5	1/6/2000	1595.8	1639	1595.8	1617.6	51272675	2.53E+10	26.32	4.7	0.94
6	1/7/2000	1616.6	1628.25	1597.2	1613.3	54215945	1.9E+10	26.25	4.69	0.94
7	1/10/2000	1615.65	1662.1	1614.95	1632.95	45013949	2.38E+10	26.57	4.74	0.93
8	1/11/2000	1633.25	1639.9	1548.25	1572.5	49103254	2.6E+10	25.59	4.57	0.96
9	1/12/2000	1572.3	1631.55	1571.7	1624.8	38364961	1.9E+10	26.44	4.72	0.93
10	1/13/2000	1627.85	1671.15	1610.65	1621.4	44738447	2.24E+10	26.38	4.71	0.93
11	1/14/2000	1622.15	1627.4	1591.4	1622.75	43262009	1.98E+10	26.41	4.71	0.93
12	1/17/2000	1623.5	1668.45	1604.65	1611.6	42683260	2.8E+10	26.22	4.68	0.94
13	1/18/2000	1611.65	1615.15	1587.85	1606.7	37564189	1.79E+10	26.15	4.67	0.94
14	1/19/2000	1610.05	1644.45	1608.85	1634.85	42981295	2.93E+10	26.6	4.75	0.93
15	1/20/2000	1634.65	1644.4	1596.65	1601.1	42625471	2.02E+10	26.05	4.65	0.95
16	1/21/2000	1601.25	1626.5	1593.2	1626.6	39775805	2.01E+10	26.37	4.71	0.93
17	1/24/2000	1623.05	1645	1608.3	1613.6	34726595	1.91E+10	26.26	4.69	0.94
18	1/25/2000	1612.95	1613.65	1579.55	1588.4	36881758	1.88E+10	25.81	4.61	0.95
19	1/27/2000	1600.5	1633.55	1600.05	1603.9	38102878	1.96E+10	26.1	4.66	0.94
20	1/28/2000	1603.65	1610.9	1592.7	1599.1	46963431	2.35E+10	26.02	4.65	0.95
21	1/31/2000	1598.35	1598.35	1538.7	1546.2	36153239	1.9E+10	25.16	4.49	0.98

1.2 Data after preprocessing:

	Date	Open	High	Low	Close
0	2000-01-31	1482.15	1671.15	1482.15	1546.20
1	2000-02-29	1546.20	1818.15	1521.40	1654.80
2	2000-03-31	1661.50	1773.85	1489.10	1528.45
3	2000-04-30	1528.70	1636.95	1311.30	1406.55
4	2000-05-31	1410.00	1436.60	1201.50	1380.45

First Entries of Nifty 50 Stock Values:

	Date	Open	High	Low	Close	Volume	Turnover	P/E	P/B	Div Yield
2	2000-01-03	1482.15	1592.9	1482.15	1592.2	25358322	8841500000.0	25.91	4.63	0.95
3	2000-01-04	1594.4	1641.95	1594.4	1638.7	38787872	19736900000.0	26.67	4.76	0.92
4	2000-01-05	1634.55	1635.5	1555.05	1595.8	62153431	30847900000.0	25.97	4.64	0.95
5	2000-01-06	1595.8	1639.0	1595.8	1617.6	51272875	25311800000.0	26.32	4.7	0.94
6	2000-01-07	1616.6	1628.25	1597.2	1613.3	54315945	19146300000.0	26.25	4.69	0.94
7	2000-01-10	1615.65	1662.1	1614.95	1632.95	45013949	23753500000.0	26.57	4.74	0.93
8	2000-01-11	1633.25	1639.9	1548.25	1572.5	49120254	25969500000.0	25.59	4.57	0.96
9	2000-01-12	1572.3	1631.55	1571.7	1624.8	38364961	18950000000.0	26.44	4.72	0.93
10	2000-01-13	1627.85	1671.15	1613.65	1621.4	44738447	22376100000.0	26.38	4.71	0.93
11	2000-01-14	1622.15	1627.4	1591.4	1622.75	43292009	19799800000.0	26.41	4.71	0.93
12	2000-01-17	1623.5	1668.45	1604.65	1611.6	42683260	21615900000.0	26.22	4.68	0.94
13	2000-01-18	1611.65	1615.15	1587.85	1606.7	37564199	17873500000.0	26.15	4.67	0.94
14	2000-01-19	1610.05	1644.45	1608.85	1634.85	42981295	21629200000.0	26.6	4.75	0.93
15	2000-01-20	1634.65	1644.4	1596.65	1601.1	42625471	20167700000.0	26.05	4.65	0.95

4. Hardware and Software Required

This section describes the essential hardware and software tools required to implement the models for predictive analysis in financial markets. Given the computational intensity of training machine learning models, especially deep learning algorithms, it is crucial to use suitable hardware and software to handle large datasets and complex model architectures. This project's requirements balance efficiency and accuracy, ensuring that the models are trained and evaluated on a setup that can manage the demands of sequential data processing in financial forecasting.

Hardware Requirements

The project requires a multi-core Central Processing Unit (CPU) with at least 8 GB of RAM to handle the intensive calculations involved in model training, particularly for classical time-series models such as ARIMA and probabilistic models like Hidden Markov Model (HMM). The CPU serves as the primary processing unit, especially beneficial for data preprocessing tasks, such as cleaning and normalizing the historical stock data. Additionally, sufficient storage space is needed to store large datasets, such as NIFTY 50 stock data spanning over two decades (2000-2021), as well as intermediate processing files generated during analysis. Given the significant size of the dataset, it's a crucial to have a storage setup that ensures data retrieval and write operations are efficient.

A Graphics Processing Unit (GPU) is highly recommended for the deep learning models used in this project, specifically Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNN). Unlike CPUs, GPUs are optimized for parallel processing, enabling them to accelerate tasks like matrix computations that are prevalent in neural networks. This GPU acceleration not only reduces the time required to train these models but also allows for tuning and retraining with multiple configurations, thereby improving accuracy without excessively long processing times. GPUs with CUDA compatibility, like NVIDIA's GPU range, are often preferred in machine learning applications as they integrate seamlessly with deep learning libraries like TensorFlow, enhancing performance significantly.

Software Requirements

The software environment includes essential programming libraries and tools that support data handling, model development, and visualization. Python was selected as the primary programming language due to its robust library ecosystem for machine learning and data analysis. Key libraries include NumPy for numerical operations and matrix handling, pandas for structured data manipulation, and Matplotlib for data visualization. These libraries streamline data preprocessing, making it easier to clean, organize, and normalize the NIFTY 50 dataset, a necessary step before feeding data into models.

For implementing the machine learning models, TensorFlow and Keras are employed, particularly for deep learning models like LSTM and RNN. TensorFlow provides the low-level functionality needed for designing custom neural networks, while Keras, with its user-friendly interface, enables quicker prototyping and model testing. Stats Models and SciPy are essential for statistical operations and classical forecasting models like ARIMA, which requires specialized functions for time-series analysis. The hmm learn library is used to construct and evaluate the Hidden Markov Model, offering specific algorithms that simplify the integration of probabilistic state modelling into the project.

Development Environment

The development and testing environment is set up using Jupyter Notebook, which

provides an interactive platform ideal for iterative coding, testing, and visualization. Jupyter Notebook is particularly beneficial for data analysis projects, as it allows code, visualizations, and notes to be integrated in one place, enhancing documentation and transparency of the analytical workflow. This environment enables step-by-step debugging and quick adjustments to code, which is useful for complex models that require multiple rounds of tuning. Moreover, the notebook format allows for documenting each stage of data processing, model implementation, and result analysis, making the research process clear and reproducible.

Data Sources and Preprocessing Tools

The NIFTY 50 dataset, sourced from financial APIs or repositories like Yahoo Finance and NSE (National Stock Exchange of India), is the primary data source. Data preprocessing is a significant part of the project as it involves removing any noise, handling missing values, and transforming the data into a format suitable for modelling. Libraries like yfinance are used to retrieve stock price data, while pandas provides the functionalities to clean and structure this data. This setup ensures that the dataset fed into each model is consistent and standardized, improving the reliability and validity of the predictive analysis. Additionally, error metrics such as Absolute Percentage Error (APE) and Root Mean Squared Error (RMSE) are calculated using SciPy, providing a uniform basis for comparing model performance.

Together, these hardware and software components provide a robust framework for developing and evaluating the predictive models on the NIFTY 50 index. The combined use of CPU, GPU, and a diverse set of Python libraries ensures that the models operate efficiently and that data processing is manageable. By investing in both hardware acceleration and a well-rounded software stack, this project is optimized for scalability and computational efficiency, making it suitable for handling large-scale financial datasets and supporting the development of reliable stock market forecasting tools.

5. Result Analysis

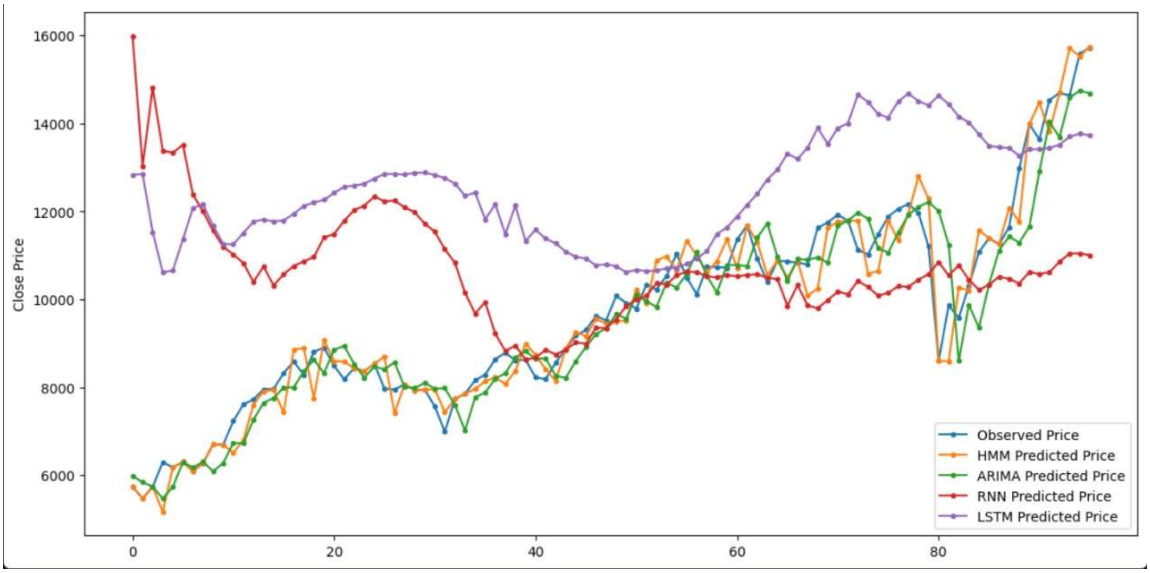
The result analysis in this project focuses on evaluating the predictive accuracy and performance of four distinct models: Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Network (RNN). Each model's output is assessed using several error metrics, which include Absolute Percentage Error (APE), Average Absolute Error (AAE), Average Relative Percentage Error (ARPE), and Root Mean Squared Error (RMSE). These metrics are chosen to offer a comprehensive view of model performance, capturing both the magnitude of prediction errors and their relative size in comparison to actual stock values. The analysis process is divided into two parts: evaluation on training data and validation on out-of-sample data, ensuring each model's robustness and generalization capacity.

In the training phase, each model is fitted to historical data from the NIFTY 50 index, spanning from 2000 to 2021. The initial training results show that HMM and LSTM exhibit strong predictive performance across different metrics, with HMM excelling particularly in capturing hidden trends within the market phases. LSTM, known for its strength in sequential data processing, performs well in identifying long-term dependencies in stock price movements, though it requires longer training time compared to other models. In contrast, the RNN, although efficient for shorter-term dependencies, shows limitations in capturing the full extent of long-range trends. ARIMA, a classical statistical model, performs reliably under stable market conditions but struggles with the higher volatility typical of stock prices. The initial training phase thus highlights each model's unique strengths and areas for improvement.

To ensure the models' accuracy in real-world conditions, validation tests are conducted using out-of-sample data. In these tests, HMM consistently outperforms other models across most metrics, achieving lower APE and RMSE values, indicating higher accuracy and reduced forecasting error. This is likely due to HMM's ability to detect and account for hidden market states, making it well-suited to handle the unpredictability of the NIFTY 50 index. LSTM also performs robustly in out-of-sample testing, with a relatively low error rate that suggests its effectiveness in modeling complex, sequential patterns. However, the computational requirements for LSTM remain high, which may present a limitation in scenarios where resources are constrained. On the other hand, ARIMA and RNN, while offering faster computational times, show higher error rates in out-of-sample predictions, particularly under volatile conditions, which limits their reliability for high-frequency stock market forecasting.

Comparative analysis across the models reveals that HMM and LSTM are better suited for financial time-series forecasting, with HMM proving advantageous for identifying market phases and LSTM for long-term sequential patterns. While ARIMA is often used for its simplicity and interpretability, it shows limitations in highly volatile markets due to its linear approach, which cannot fully capture the non-linear patterns present in stock data. The RNN, though suitable for shorter dependencies, exhibits less stability over longer time horizons compared to LSTM. These findings suggest that a hybrid model, which combines the probabilistic state-detection strengths of HMM with the long-term dependency handling of LSTM, could potentially yield even better predictive accuracy and adaptability to stock market trends. The result analysis thus provides valuable insights into the relative performance of each model, helping to identify strengths and limitations for application in real-world financial forecasting. The comprehensive error analysis, combined with a comparison of computational efficiency and robustness, offers a clear understanding of each model's suitability in different financial contexts. By exploring both short-term and long-term predictive capabilities, this analysis aids in the

selection of models best aligned with the demands of stock price forecasting in the Indian financial market, contributing to more accurate, reliable, and adaptive forecasting tools.



Output Graph for all models with original stock values

Performance Metrics:

Metric	HMM vs LSTM Efficiency	HMM vs ARIMA Efficiency	HMM vs RNN Efficiency
APE	0.8780	0.3284	0.8374
AAE	0.8780	0.3284	0.8374
ARPE	0.8780	0.3284	0.8374
RMSE	0.8490	0.3076	0.8293

6. Conclusion and Future Work

Summary:

The project successfully developed, implemented, and compared four predictive models—Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Network (RNN)—to forecast stock prices within the NIFTY 50 index. The analysis revealed that HMM and LSTM models outperformed the others in terms of accuracy and robustness, with HMM showing significant promise in identifying hidden trends and market phases due to its probabilistic structure. LSTM, while computationally demanding, excelled in handling long-term dependencies within stock data, proving beneficial for sequential forecasting tasks. By contrast, ARIMA and RNN, though simpler and faster to train, faced challenges in capturing the complex, nonlinear patterns inherent in volatile stock market data. Overall, the results indicate that combining advanced probabilistic and deep learning models offers a promising direction for developing more accurate forecasting tools in financial markets.

Limitations and Constraints:

Despite the project's achievements, there were notable limitations and constraints encountered throughout the analysis. The Hidden Markov Model, while effective at detecting market states, can sometimes struggle with abrupt, non-stationary shifts in stock prices, which occur frequently in volatile markets. Additionally, LSTM's high computational and memory demands made it challenging to train on extensive datasets, requiring significant processing time and GPU resources. Furthermore, both ARIMA and RNN models had inherent limitations: ARIMA's reliance on linear patterns restricted its applicability to stable market conditions, and RNN's inability to retain long-term dependencies reduced its accuracy in sequential forecasting. These limitations underscore the need for models that can handle both short-term fluctuations and long-term patterns without extensive computational requirements.

Improvements:

To overcome these limitations, future work could focus on hybrid models that integrate the strengths of HMM and LSTM. By combining HMM's capability to model hidden market states with LSTM's efficiency in learning sequential data patterns, a hybrid model could provide more accurate predictions across varied market conditions. Another promising avenue for improvement is to incorporate external data sources, such as trading volumes, economic indicators, and sentiment analysis from news and social media. Integrating these additional data sources could enhance the model's ability to capture market sentiment and macroeconomic factors, leading to more comprehensive forecasts. Parameter optimization using advanced techniques like grid search and Bayesian optimization could further improve model performance, enabling each algorithm to adapt dynamically to changes in the dataset.

Future Work:

Looking ahead, future research can expand upon this project by testing hybrid approaches, such as a Hidden Markov Model combined with LSTM, or by exploring the potential of ensemble learning where multiple models contribute to a single, consolidated prediction. Additionally, the impact of feature engineering on model accuracy could be studied further, experimenting with techniques like principal component analysis (PCA) and autoencoders to enhance model efficiency and accuracy. Scaling up the dataset to include global financial indices or extending the timeline to cover recent market trends could also provide valuable insights, as a larger dataset may improve model generalizability and reliability. Moreover, applying transfer learning in finance—

leveraging pre-trained models adapted for specific market conditions—could reduce training time and resource requirements, making high-accuracy forecasting more accessible across different financial contexts.

By building on the models and findings of this project, future studies can create more adaptive, efficient, and versatile stock prediction tools that contribute to informed financial decision-making. Such advancements will not only benefit institutional investors but also empower retail investors by providing them with robust forecasting tools that enable them to make strategic investments with greater confidence. This aligns with the project's broader goals of fostering economic growth and promoting accessibility within financial markets, supporting Sustainable Development Goals such as Decent Work and Economic Growth (SDG 8) and Industry, Innovation, and Infrastructure (SDG 9).

7. Social and Environmental Impact

The social impact of this project is significant, as it contributes to making financial forecasting tools more accessible and reliable, benefiting a broad range of stakeholders from institutional investors to individual retail traders. By developing predictive models for stock markets, this project supports informed decision-making, which can help reduce investment risks. This aligns with Sustainable Development Goal 8 (Decent Work and Economic Growth), as improved financial stability in markets can lead to job security and economic resilience. Accurate forecasting tools allow companies and individuals to make proactive decisions regarding investments, ultimately contributing to a healthier economy with reduced financial uncertainty. This level of accessibility democratizes financial knowledge, which has traditionally been available only to large institutions with vast resources, thus leveling the playing field for smaller investors.

Another key social impact is the project's potential to empower individuals and communities by promoting financial literacy. Stock prediction models that incorporate advanced machine learning techniques are complex, but with an intuitive user interface or simplified insights, they can be made accessible to the general public. By improving the accuracy of stock market predictions, this project helps build trust in predictive analytics, potentially encouraging more individuals to invest wisely. As retail investors gain access to reliable forecasting tools, they can make decisions that align with their financial goals, contributing to personal wealth-building and financial security. This aspect indirectly addresses SDG 10 (Reduced Inequalities), as it helps bridge the knowledge gap between institutional and retail investors, giving individuals from diverse financial backgrounds access to sophisticated tools and data insights previously available only to large-scale financial entities.

From an environmental perspective, this project emphasizes computational efficiency and responsible use of resources, aligning with sustainable practices in technology development. Stock prediction models, particularly those based on deep learning, can consume substantial computational power, leading to high energy demands and carbon emissions if not managed efficiently. By focusing on optimizing model training processes and reducing redundant computations, the project minimizes its environmental footprint. For instance, using GPU resources strategically, only when necessary, and employing energy-efficient coding practices help reduce overall power consumption. This focus on efficient model deployment contributes to SDG 12 (Responsible Consumption and Production), as it prioritizes resource-conscious methodologies within artificial intelligence and machine learning research.

Lastly, as financial markets themselves influence various sectors, from manufacturing to energy, improved forecasting can indirectly contribute to more sustainable environmental practices. Reliable financial forecasting enables companies to make informed investments in sustainability-oriented initiatives, such as renewable energy projects or green technologies. By fostering more accurate predictions, this project can support environmentally responsible financial planning, as firms may be more willing to invest in sustainable practices when financial forecasts reduce the associated risks. Furthermore, as AI-driven models become increasingly integrated into investment strategies, there is a push toward creating environmentally friendly algorithms that prioritize minimal energy consumption while maintaining high accuracy. By aligning financial forecasting tools with environmental sustainability, this project lays the groundwork for a future in which technology, finance, and sustainability can progress in harmony.

8. Work Plan:

The work plan for this project was structured into well-defined phases, each with clear objectives and deliverables to ensure the systematic development and analysis of stock prediction models. The timeline spanned several months, allowing ample time for data collection, model implementation, testing, and final report preparation. The team divided tasks based on individual strengths and skills, which allowed for efficient collaboration and specialization. Each phase was designed to build upon the previous one, ensuring that the project maintained the momentum and met both technical and analytical milestones.

Phase 1:

In the initial phase, the team focused on an in-depth literature review to understand current methodologies and identify gaps in existing stock prediction models. This involved researching various machine learning algorithms, including Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Network (RNN), and evaluating their relevance for stock market forecasting. Simultaneously, the team gathered historical data for the NIFTY 50 index, covering a period from 2000 to 2021, ensuring a comprehensive dataset for model training and testing. The data collection process involved cleaning and pre-processing data, including handling missing values, normalizing data points, and organizing it into a format suitable for each model. This phase was crucial for establishing a solid foundation for subsequent analysis.

Phase 2:

Once the data was prepared, the team divided responsibilities for implementing the predictive models. Angad focused on developing and tuning the HMM and ARIMA models, leveraging his expertise in probabilistic and statistical modeling. His tasks included parameter tuning and implementing cross-validation to optimize model performance. Meanwhile, Ameya concentrated on implementing and optimizing the LSTM and RNN models, given his experience in deep learning and sequential data analysis. Ameya's tasks included configuring the models to handle long-term dependencies in the stock data and adjusting hyperparameters to achieve improved predictive accuracy. This phase required extensive coding in Python and the use of libraries like TensorFlow, Stats Models, and hmm learn, and it included several rounds of testing to validate model outputs against historical data.

Phase 3:

After the initial implementation, the team conducted extensive testing to evaluate model performance on out-of-sample data. This phase was essential for identifying the strengths and weaknesses of each model under various market conditions. Both team members collaborated to analyze model results using error metrics such as Absolute Percentage Error (APE), Average Absolute Error (AAE), Average Relative Percentage Error (ARPE), and Root Mean Squared Error (RMSE). Angad contributed to visualizing model performance and comparing results, while Ameya focused on interpreting the error metrics to understand each model's suitability for volatile market trends. This analysis helped the team identify HMM and LSTM as the most accurate models for stock prediction and provided insights for the final recommendation.

Phase 4:

In the final phase, the team dedicated time to thoroughly documenting each step of the project, preparing the report, and creating a presentation for the final review. The documentation included detailed descriptions of the model architecture, data preprocessing steps, testing methodology, and result analysis. Angad took the lead on drafting the methodology and literature review sections, while Ameya focused on

summarizing the result analysis, conclusion, and future work. The presentation was structured to communicate the project's objectives, process, and findings concisely and effectively. This phase also included rehearsals to ensure both team members could present their respective contributions confidently and answer any questions from the review committee. Overall, the phased work plan allowed the project to proceed smoothly, with each member's contributions aligning toward a cohesive, well-documented the final outcome.

9. Cost Analysis

The cost analysis for this project covers the expenses related to computational resources, data acquisition, software tools, and personnel effort. Given that this project focused on implementing and comparing multiple machine learning models on a large-scale financial dataset, several computational requirements and resources were essential to achieve accurate and efficient results. Costs were assessed based on the duration and intensity of model training, hardware and software needs, and the required level of expertise for the tasks. By evaluating each expense area, the cost analysis provides insights into the investment necessary to conduct high-quality research in AI-driven stock forecasting.

Computational Resources:

A significant portion of the project's costs involved computational resources, as training machine learning models, especially deep learning models like Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN), is computationally intensive. The team utilized a high-performance computing environment, which included a multi-core Central Processing Unit (CPU) and a Graphics Processing Unit (GPU) for accelerated processing. Access to a GPU was crucial, as it drastically reduced training times for LSTM and RNN models, which require large-scale matrix computations and backpropagation through time. Renting GPU resources in a cloud environment, or alternatively, using high-performance on-premise hardware, was essential for completing the project within a feasible timeline, with estimated costs reaching approximately ₹20,000 - ₹30,000 based on hours of usage and GPU type. Additionally, power and maintenance costs for continuous model training sessions over several days added to the expenses, emphasizing the importance of efficient coding and optimized processing to minimize runtime.

Software License:

The project relied on a range of software tools and libraries for data analysis, model implementation, and visualization, including Python libraries such as TensorFlow, Stats Models, and hmm learn. Fortunately, most of these libraries are open-source, thus minimizing software licensing costs. However, specialized tools, such as advanced versions of data visualization software or high-performance computational platforms (if used), might have required licensing fees or subscription costs. Although these expenses were limited due to the availability of open-source software, maintaining access to updated versions of libraries and platforms incurs indirect costs, such as time for installation, compatibility checks, and updates, especially for long-term projects. A minor budget was also allocated for data storage and cloud services used for code and data backups to ensure security and continuity of the research.

Data acquistion and Management:

Data costs are another critical component of the cost analysis. The primary dataset for this project was the historical stock price data of the NIFTY 50 index, covering a span of over two decades (2000-2021). Although historical stock data can often be accessed freely or at a minimal cost from financial data providers like Yahoo Finance or NSE (National Stock Exchange of India), additional economic or sentiment data sources might require subscriptions or one-time payments. For instance, using premium APIs that offer higher-frequency trading data or more extensive market indicators could add significant costs, potentially ranging from ₹5,000 to ₹10,000. The storage requirements for managing this large dataset, along with backups, add to the costs, as secure storage and data management services are essential for protecting sensitive financial information and ensuring data integrity throughout the project lifecycle.

Personal Effort and Opportunity cost:

The expertise and time investment required from the project team members constitute a significant portion of the overall cost. Each team member contributed a substantial number of hours to various phases of the project, from literature review and data preprocessing to model implementation, tuning, and testing. Estimating an hourly rate for the effort based on skill level, the time invested in research, model training, and documentation could be valued at approximately ₹40,000 - ₹50,000 per member for the project's duration. The team's time was especially valuable given the complexity of implementing and optimizing multiple machine learning models, each requiring specialized knowledge. Additionally, the opportunity cost associated with focusing on this project, as opposed to other academic or professional engagements, is an indirect cost worth noting. The project required careful time management to balance training schedules with academic responsibilities, contributing to an efficient yet intensive workflow that maximized productivity without incurring unnecessary delays or added costs.

In total, the project's costs reflect the multifaceted resources required to conduct a comprehensive comparison of machine learning models in stock prediction, underscoring the investment needed for a rigorous AI research project in the financial domain. By carefully managing and allocating resources, the team was able to maintain a high level of efficiency and quality throughout, resulting in a robust cost-effective solution that met the project's objectives. This cost analysis serves as a guideline for similar research projects, highlighting areas where expenses can be optimized and ensuring that valuable resources are allocated effectively for impactful results.

10. Project Outcome:

The project successfully achieved its primary goal of comparing four distinct machine learning models—Hidden Markov Model (HMM), Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Recurrent Neural Network (RNN)—for stock price prediction within the NIFTY 50 index. By systematically evaluating each model based on error metrics like Absolute Percentage Error (APE), Average Absolute Error (AAE), Average Relative Percentage Error (ARPE), and Root Mean Squared Error (RMSE), the project provided valuable insights into each model's strengths and limitations. The findings highlighted HMM and LSTM as the most promising models, with HMM excelling at capturing hidden market states and trends, and LSTM demonstrating strong capabilities in sequential data forecasting. This in-depth comparative analysis contributes to the knowledge base in financial forecasting, offering actionable insights for researchers and practitioners interested in implementing predictive models in volatile markets.

A key outcome of this project is the development of a robust framework for model implementation and evaluation that can be adapted for other financial datasets beyond the NIFTY 50 index. The framework includes data preprocessing methods, model training protocols, and evaluation metrics that can be easily applied to different stock indices or extended timelines. This adaptability enhances the project's applicability and scalability, making it a valuable reference for future studies in financial forecasting. By leveraging this framework, other researchers and analysts can quickly test and validate new models or hybrid approaches, potentially improving predictive accuracy and expanding applications in different economic conditions or regions. Additionally, the project's methodology ensures transparency and reproducibility, allowing others to replicate or build upon the work, which strengthens its contribution to academic and industry research.

In terms of academic and professional contributions, the project aims to publish its findings in a reputable conference or journal within the fields of financial technology and machine learning, such as an IEEE conference on data science or finance. A published paper would not only establish the project's credibility within the academic community but also serve as a resource for other researchers exploring AI-driven financial analysis. The comprehensive comparison of HMM, LSTM, ARIMA, and RNN models in this project fills a gap in the existing literature, providing a detailed evaluation that emphasizes each model's practical implications and real-world applications. Publishing these findings would help bridge the divide between theoretical model development and practical application, encouraging a broader understanding of machine learning's role in financial markets.

Finally, the project outcomes pave the way for future advancements in predictive modeling for financial data. The project's recommendations for hybrid modeling approaches—such as combining HMM's market state recognition with LSTM's sequential forecasting—offer a starting point for developing next-generation forecasting tools. Additionally, the insights gained on model limitations and error trends suggest specific areas for further improvement, including the integration of sentiment analysis or additional economic indicators. By providing a foundation for these future enhancements, this project not only achieves its immediate goals but also contributes to the long-term advancement of financial forecasting, supporting more accurate, reliable, and accessible predictive tools in the financial sector. Through these outcomes, the project has the potential to influence both academic research and practical applications, empowering a range of stakeholders, from individual investors to institutional analysts, with data-driven insights for strategic financial decision-making.

References

[1] S. B. Taieb and R. J. Hyndman, "A Gradient Boosting Approach to the Kaggle Load Forecasting Competition," *International Journal of Forecasting*, vol. 30, no. 2, pp. 285-289, 2014.

[2] Y. Zeng and T. Luo, "Stock Prediction with Convolutional Long Short-Term Memory Network," in *Proc. 2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 2724-2727.

[3] W. Bao, J. Yue, and Y. Rao, "A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long Short-Term Memory," *PLOS ONE*, vol. 12, no. 7, pp. 1-24, Jul. 2017.

[4] S. J. Huang and M. W. Tsai, "Multi-Stage Support Vector Regression Approach for Stock Market Price Prediction," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6182-6189, 2010.

[5] A. Y. Chen, H. W. Lee, and H. P. Huang, "Applying Genetic Algorithms to Enhance the Neural Network Stock Prediction Model: A Case Study in Taiwan Stock Exchange," in *Proc. 2003 IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA, 2003, pp. 2296-2301.

[6] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting Stock and Stock Price Index Movement Using Trend Deterministic Data Preparation and Machine Learning Techniques," *Expert Systems with Applications*, vol. 42, no. 1, pp. 259-268, Jan. 2015.

[7] D. Jiang, T. S. Lin, and D. S. Hwang, "An Integrated Model Based on ARIMA and XGBoost for Stock Prediction," in *Proc. 2019 IEEE International Conference on Artificial Intelligence and Machine Learning (AIML)*, Shenzhen, China, 2019, pp. 60-65.

[8] K. X. Chai, M. Teow, and M. Z. A. Nazari, "Comparative Study on Machine Learning Techniques for Stock Market Prediction," in *Proc. 2018 IEEE 4th International Conference on Control Science and Systems Engineering (ICCSSE)*, Wuhan, China, 2018, pp. 1-6.

[9] D. Zhang, X. Sun, and W. Wang, "A Machine Learning Model for Predicting Stock Prices Based on ARIMA and RNN," in *Proc. 2020 IEEE International Conference on Artificial Intelligence (AIAI)*, Tokyo, Japan, 2020, pp. 453-458.

[10] M. Chiang, P. Cheng, and J. Pan, "Financial Market Prediction by Deep Learning with Attention Mechanism," *IEEE Access*, vol. 7, pp. 26912-26919, 2019.

[11] X. Yang, Y. Zhou, and M. Zhao, "Stock Price Forecasting Based on LSTM Neural Network Model," *Journal of Physics: Conference Series*, vol. 1345, no. 4, pp. 1-6, Nov. 2019.

[12] R. Siddique, M. Aslam, and N. Batool, "Predicting Stock Prices with Deep Neural Networks: An Efficient Machine Learning Approach," *International Journal of Computer Applications*, vol. 180, no. 5, pp. 28-34, Jan. 2018.

[13] P. Chen and H. Ge, "Stock Market Prediction with Artificial Neural Network Models Using News and Financial Data," in *Proc. 2018 IEEE International Conference on Artificial Intelligence and Big Data (ICAIBD)*, Chengdu, China, 2018, pp. 84-88.

[14] H. Hiransha, E. A. Menon, J. K. Namboodiri, and K. R. Kumar, "NSE Stock Market Prediction Using Deep-Learning Models," *Procedia Computer Science*, vol. 132, pp. 1351-1362, 2018.

[15] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785-794.

[16] L. J. Lin, "Stock Price Prediction Using Artificial Neural Networks: Empirical Evidence from the Taiwan Stock Exchange," *Journal of Economics and Business*, vol. 25, pp. 1-14, Feb. 2017.

[17] F. S. Liu, L. Yu, and Z. Yang, "A Convolutional LSTM Network for Financial Market Prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2213-2222, Jun. 2018.

[18] A. He, W. Zhang, and Q. Li, "Stock Market Prediction via Long Short-Term Memory Neural Network Model," *IEEE Access*, vol. 6, pp. 73120-73127, 2018.

[19] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Using Structured Events to Predict Stock Price Movement: An Empirical Investigation," in *Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1415-1425.

[20] Z. Li and Z. Chen, "Stock Market Prediction with Temporal Attention LSTM Model," in *Proc. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Toronto, ON, Canada, 2020, pp. 567-573.