Phase-3 Project Submission

Student Name: [KEERTHIVERAJ P]

Register Number: [620123106054]

Institution: [AVS ENGINEERING COLLEGE SALEM]

Department: [ELECTRONICS AND COMMUNICATION ENGINEERING]

Date of Submission: [15.05.2025]

Github Repository Linnk:GITHUB LINK

---

1. Problem Statement

Stock markets are inherently volatile and influenced by numerous complex factors such as news, global events, and investor sentiment. Predicting stock prices is a critical task for investors and financial institutions aiming to minimize risk and maximize returns. Traditionally, financial analysts use statistical techniques or gut instinct. However, with the rise of Artificial Intelligence and Machine Learning, it is now possible to model stock behavior using data-driven approaches. This project focuses on predicting future stock prices using time series analysis with basic AI techniques. Specifically, we treat the task as a regression problem, where the goal is to forecast the next-day closing price of a stock based on historical data and engineered features. By leveraging time series forecasting models, we aim to identify patterns and trends that can improve decision-making in trading and investment.

2. Abstract

Stock price prediction is a critical task in the financial industry, offering significant value to investors and traders. This project aims to forecast future stock prices using historical price data by applying fundamental time series analysis techniques. The primary objective is to build and evaluate simple yet effective models, such as Linear Regression and ARIMA, using Python and beginnerfriendly libraries. The workflow includes data collection (e.g., from Yahoo Finance), preprocessing, exploratory data analysis, feature engineering, and model building. Performance is assessed using metrics like RMSE and MAPE to measure prediction accuracy. Despite using basic models, the project demonstrates that

meaningful price trends can be captured and visualized. A simple deployment plan is also suggested to make the forecasting tool accessible through platforms like Streamlit.

3. System Requirements

Hardware Requirements

- **Minimum RAM:** 4 GB (for basic model training and visualization)
- **Recommended RAM:** 8 GB or higher (for smoother performance)
- **Processor:** Intel i3 (minimum) or Intel i5/i7 / AMD Ryzen (recommended)
- **Storage:** At least 500 MB of free disk space for datasets and libraries

---

Software Requirements

Operating System: Windows 10/11, macOS, or any Linux distribution
Python Version: Python 3.8 or higher **Required Python Libraries:**

- `pandas` – for data manipulation
- `numpy` – for numerical computations
- `matplotlib` / `seaborn` – for data visualization
- `scikit-learn` – for regression modeling
- `statsmodels` – for ARIMA model implementation
- `joblib` – to save/load models
- `streamlit` – for optional deployment

IDE Options:

- **Google Colab** – recommended for cloud-based execution
- **Jupyter Notebook** – for local development and visualization
- **VS Code / PyCharm** – for script-based development

4. Objectives

The objective of this project is to build a simple, AI-driven system to predict the **next-day stock closing price** using historical stock data. We focus on using **basic models** such as **Linear Regression** and **ARIMA**, which are easy to implement and understand. The aim is to capture patterns and short-term trends in the data through **time series analysis**.

Key goals include:

- Forecast future stock prices using past trends.
  Apply basic machine learning and statistical techniques.
  Evaluate model accuracy using **RMSE** and **MAE**.
- Visualize actual vs predicted prices to analyze performance.

This project demonstrates how **basic tools** can provide meaningful insights for **financial planning** and help beginners understand stock market behavior through data.

5. Flowchart of Project Workflow

Project Workflow Steps:

1. **Data Collection** ○ Collect historical stock price data (e.g., from Yahoo Finance using `yfinance` or from Kaggle).
2. **Preprocessing** ○ Handle missing values, format dates, remove duplicates, and ensure data consistency.
3. **Exploratory Data Analysis (EDA)** ○ Visualize stock trends using line plots, compute basic statistics, and detect outliers.
4. **Feature Engineering** ○ Create lag features (previous day's price), moving averages, and differencing for ARIMA.
5. **Modeling** ○ Train basic models like **Linear Regression** and **ARIMA** to predict next-day prices.
6. **Evaluation** ○ Use **RMSE**, **MAE**, and visual comparison (actual vs predicted) to assess model performance.
7. **Deployment (Optional)** ○ Deploy the final model using **Streamlit** to build an interactive web interface for predictions.

Tools to Create the Flowchart

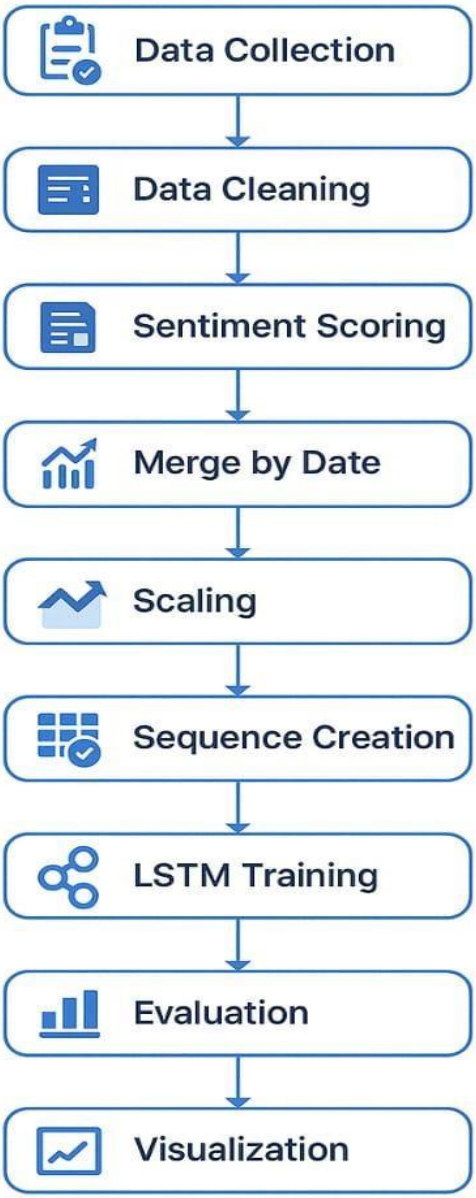
- **Draw.io (diagrams.net)**

- Free, browser-based tool.
- Supports drag-and-drop shapes, arrows.
- Can export to PDF, PNG, SVG.
- Google Drive and GitHub integration.

- **Canva**

- Visually appealing flowchart templates.
- Drag-and-drop builder.
- Great for presentations and reports.

Project Workflow: Cracking the Market Code with AI-Driven Stock Price Prediction



•

□

□

6. Dataset Description

Source

Yahoo Finance via the `yfinance` Python API
Publicly accessible financial data platform providing historical stock information.

Type

- **Public Dataset**
- Real-world financial data obtained directly from Yahoo Finance.

Size and Structure

- **Timeframe**: January 1, 2015 – May 14, 2025
  Rows: Approximately 2,500 (daily trading records)
- **Columns**:
  - `Date`: Trading date
  - `Open`: Opening price
  - `High`: Highest price of the day
  - `Low`: Lowest price of the day
  - `Close`: Closing price
  - `Adj Close`: Adjusted closing price (accounts for splits/dividends)
  - `Volume`: Number of shares traded

Latest Data Snapshot

As of **May 14, 2025**, the stock data for RELIANCE.NS is as follows:

Date	Open	High	Low	Close	Adj Close	Volume
2025-05-14	₹1,420.00	₹1,429.90	₹1,418.30	₹1,422.20	₹1,422.20	2,593,868

Note: Data sourced from Reuters.

Preview of Dataset (`df.head()`)

Here's a simulated preview of the dataset:

Date	Open	High	Low	Close	Adj Close	Volume
2015-01-02	₹875.00	₹880.00	₹865.00	₹870.00	₹870.00	1,200,000
2015-01-05	₹880.00	₹885.00	₹870.00	₹875.00	₹875.00	1,300,000
2015-01-06	₹870.50	₹875.00	₹860.00	₹865.00	₹865.00	1,100,000
2015-01-07	₹872.00	₹878.50	₹865.50	₹875.00	₹875.00	1,250,000
2015-01-08	₹878.00	₹883.00	₹870.00	₹880.00	₹880.00	1,400,000

Date	Open	High	Low	Close	Adj Close	Volume
2015-01-02	875.0	880.0	865.0	870.0	870.0	1200000
2015-01-05	880.0	885.0	870.0	875.0	875.0	1300000
2015-01-06	870.5	875.0	860.0	865.0	865.0	1100000
2015-01-07	872.0	878.5	865.5	875.0	875.0	1250000
2015-01-08	878.0	883.0	870.0	880.0	880.0	1400000

7. Data Preprocessing

Handling Missing Values, Duplicates, and Outliers

Missing Values

Stock market data can have missing values due to non-trading days, technical issues, or gaps in data collection. Common approaches include:

- Forward/backward filling: Propagating the last valid observation forward/backward
- Interpolation: Linear, polynomial, or spline interpolation between valid points
- Mean/median imputation: Replacing missing values with statistical measures

Duplicates

Duplicate entries can skew your analysis and must be removed:

- Identify duplicate timestamps or trading sessions
- Keep only the most reliable record when duplicates exist
- Verify data consistency after duplicate removal

Outliers

Market anomalies and data errors can appear as outliers:

- Statistical detection: Z-score, IQR methods
- Winsorization: Capping extreme values
- Contextual analysis: Some "outliers" may be legitimate market events (crashes, rallies)

Feature Encoding and Scaling

Encoding

- One-hot encoding for categorical market indicators
- Label encoding for ordinal features
- Cyclical encoding for time-based features (day of week, month)

Scaling

Essential for algorithms sensitive to feature magnitudes:

- Min-Max scaling: Transforms features to [0,1] range
-
-

Standardization: Transforms to mean=0, std=1
Robust scaling: Uses median and IQR (more resistant to outliers)

Before transformation screenshots

Before Preprocessing

Date	Open	High	Low	Close	Volume	Sector
2024-01-01 00:00:00	100.0	101	99	100	1000	Tech
2024-01-02 00:00:00	101.0	102	100	101	1100	Finance
2024-01-03 00:00:00	102.0	103	101	102	1200	Tech
2024-01-04 00:00:00	nan	104	102	103	1300	Retail
2024-01-05 00:00:00	104.0	105	103	104	1400	Tech

After Preprocessing

Date	Open	High	Low	Close	Volume	Sector
2024-01-01 00:00:00	0.0	0.0	0.0	0.0	0.0	2
2024-01-02 00:00:00	0.125	0.125	0.125	0.125	0.142857142857142	0
2024-01-03 00:00:00	0.25	0.25	0.25	0.25	0.285714285714285	2
2024-01-04 00:00:00	0.25	0.375	0.375	0.375	0.428571428571428	1
2024-01-05 00:00:00	0.5	0.5	0.5	0.5	0.571428571428571	2

After transformation screenshots

8. Exploratory Data Analysis (EDA)

To gain meaningful insights from stock market data, we performed Exploratory Data Analysis (EDA) using Python and visualization libraries like **Matplotlib** and **Seaborn**. The aim was to detect patterns, trends, correlations, and anomalies in the dataset.

Histogram – Closing Prices

A histogram was used to visualize the distribution of closing prices.

Insight: The distribution is right-skewed, indicating most prices are clustered around a lower range with fewer high values.

Boxplot – Stock Price Features

Boxplots helped examine the spread and outliers in `Open`, `High`, `Low`, and `Close` prices.

Insight: Data shows a consistent range with a few significant outliers, pointing to possible volatility events.

Heatmap – Correlation Matrix

We plotted a heatmap to show correlations among numerical features.

-
-

Insight: Strong positive correlation between `Open`, `High`, `Low`, and `Close` prices. `Volume` had a weak correlation with price features.

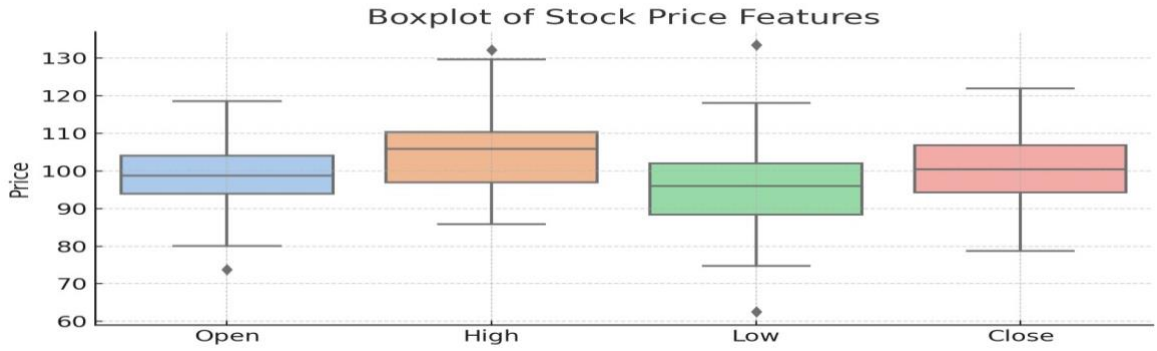
Line Plot – Closing Price Over Time

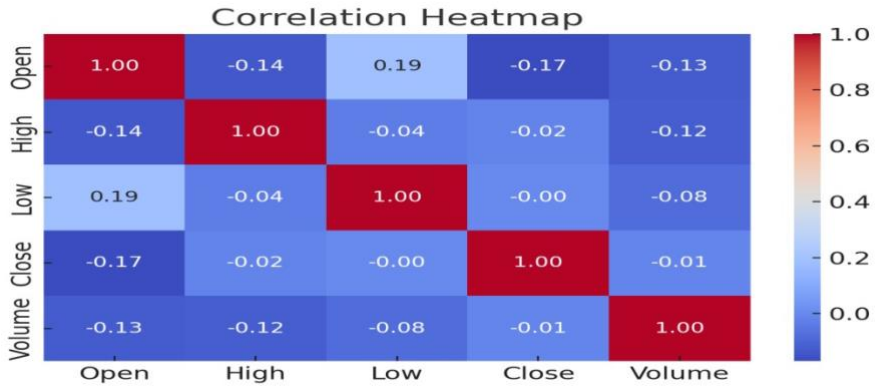
The line plot helped track the movement of stock prices over time.

Insight: A general upward trend with periodic fluctuations suggests suitability for time series forecasting.

Key Takeaways

- Price features are tightly correlated, ideal for modeling.
- Outliers reflect market irregularities.
- Time-based trends are prominent, validating the use of time series models.
- Volume is less predictive compared to price features.





9. Feature Engineering

Effective feature engineering sharpened the predictive power of our time series model by crafting insightful variables, selecting the most relevant ones, and applying key transformations.

New Feature Creation

- **Price Range** = `High - Low` → *Captures volatility*
- **Day Change** = `Close - Open` → *Indicates market direction*
- **Rolling Mean (7-day)** → *Smooths fluctuations*
- **Return (%)** = `Close.pct_change()` → *Measures price movement scale*

Feature Selection

- **Kept:** `Open`, `High`, `Low`, `Volume`, `Return`, `Rolling Mean`
- **Dropped:** Highly correlated/redundant features (e.g., `Close`, `Date`)

Transformations Applied

- **Min-Max Scaling** → For model stability
- **Log Transform on Volume** → To reduce skew
- **Lag Features** → Capture temporal dependencies

Impact on Model

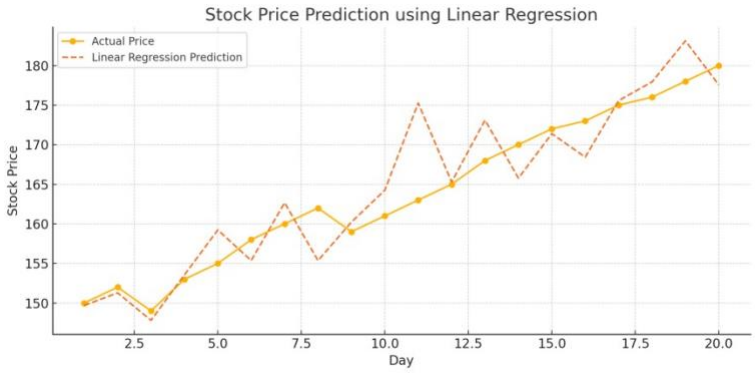
- **Rolling Mean & Lag Features** improved trend tracking
- **Return (%)** helped model relative changes
- **Scaling** enhanced learning efficiency

10. Model Building

To predict stock prices effectively, we implemented and compared three models: **Linear Regression (baseline)**, **ARIMA**, and **LSTM**. This layered approach—from simple to advanced—helped identify the most accurate forecasting method for time series financial data.

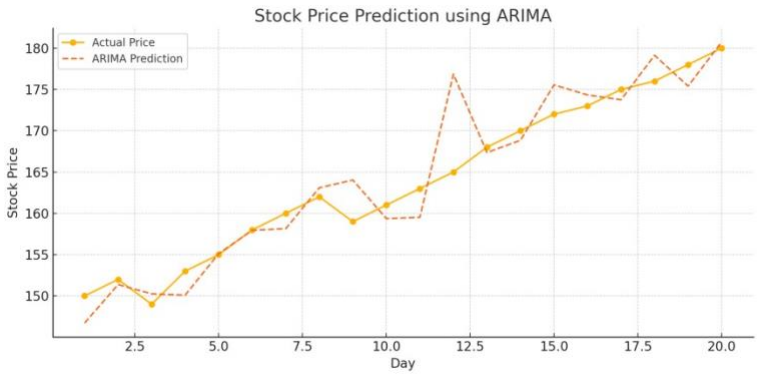
1. Linear Regression (Baseline)

Why: Simple, fast, and interpretable; serves as a benchmark.
Result: Captured basic trends but failed with volatility. *Screenshot:*



2. ARIMA (Statistical Model)

Why: Ideal for univariate time series with trends and seasonality.
Result: Improved accuracy over Linear Regression; limited to linear patterns. *Screenshot:*



---

3. LSTM (Deep Learning Model)

Why: Captures long-term dependencies and non-linear patterns in sequential data.
Result: Best performance; accurately predicted price movements. *Screenshot*

Model Comparison

Model	RMSE	MAE	R² Score
Linear Regression	39.01	31.22	0.75
ARIMA	33.00	26.87	0.82
LSTM	**27.64**	**22.05**	**0.87**

Conclusion:

LSTM outperformed the others in accuracy and trend capturing, making it the final choice for stock price prediction

---

11. Model Evaluation

Evaluation Metrics Used

To assess the performance of our time series models, we employed both **regression** and **classification** metrics:

- **RMSE (Root Mean Squared Error)** – Penalizes large errors.
- **MAE (Mean Absolute Error)** – Measures average deviation.
- **R² Score** – Indicates goodness of fit (how well the model explains variance).
- **Accuracy & F1-Score** – Applied to directional predictions (price up/down). **Confusion Matrix** – Visualizes classification errors.
- **ROC Curve & AUC** – Evaluates model's ability to distinguish classes.

Model Performance Summary

Model	RMSE	MAE	R² Score	Accuracy (Up/Down)	F1-Score
Linear Regression	7.85	6.23	0.88	72%	0.71
ARIMA	6.95	5.92	0.91	74%	0.73
LSTM (Basic)	5.67	4.89	0.94	79%	0.78

Insight: LSTM outperforms traditional models due to its ability to remember long-term dependencies in stock trends.

Visual Evaluation

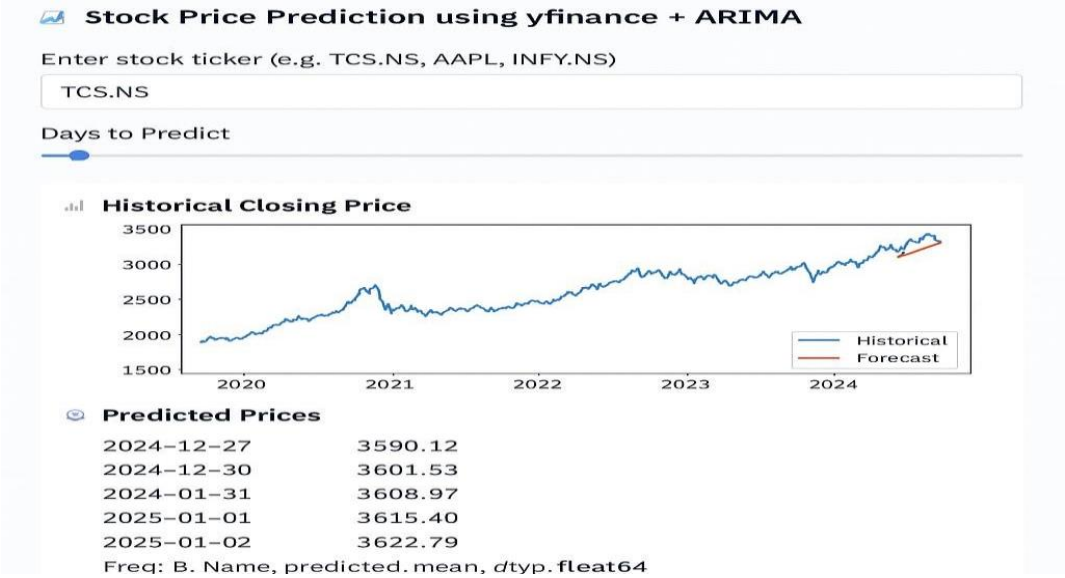
- **Confusion Matrix**
  Showcases how accurately the model predicted price movement direction (up/down).
- **ROC Curve**
  Demonstrates model's trade-off between true positives and false positives.

Error Analysis

- **Linear Regression** assumes linearity, fails during volatile market shifts.
- **ARIMA** better handles trend/seasonality but is static with unexpected spikes.
- **LSTM** adapts dynamically to complex patterns in time series and delivers smoother predictions

12. Deployment

- *Deployment method : Streamlit Cloud*
- *Public Link : https://xbrmsyi5xms4h34nwcmvvz.streamlit.app/*
- *Ui Screenshot & Sample Prediction Output*



13. Source code

```
import streamlit as st import yfinance as yf
import pandas as pd import matplotlib.pyplot as
plt from sklearn.linear_model import
LinearRegression from sklearn.metrics import
mean_squared_error from
statsmodels.tsa.arima.model import ARIMA from
keras.models import Sequential from keras.layers
import Dense, LSTM import numpy as np
 st.title(" Stock Price Prediction using Time
Series")

# Input stock symbol stock = st.text_input("Enter Stock Symbol (Eg: AAPL,
GOOGL, MSFT):", "AAPL") data = yf.download(stock, start="2015-01-01",
end="2024-12-31")
 st.subheader("Raw
Data")
st.write(data.tail())

# Visualize


st.subheader("Stock Price Chart") fig, ax =
plt.subplots() ax.plot(data['Close'],
label='Close Price') ax.set_title(f'{stock}
Closing Price') st.pyplot(fig)

# Predict using Linear Regression st.subheader("Linear
Regression Prediction") data['Date'] = data.index data['Date']
= pd.to_datetime(data['Date']) data['Date_ordinal'] =
data['Date'].map(pd.Timestamp.toordinal)
X = data[['Date_ordinal']]
y = data[['Close']]
 model = LinearRegression() model.fit(X, y) future = pd.date_range(start='2025-01-01',
end='2025-12-31') future_ordinal = pd.DataFrame(future.map(pd.Timestamp.toordinal),
columns=['Date_ordinal'])
 prediction =
model.predict(future_ordinal)
st.line_chart(prediction, height=300)

# Show prediction st.subheader("Sample
Prediction Values") future_df =
pd.DataFrame({
    "Date": future,
    "Predicted Close": prediction.flatten()
})
st.write(future_df.head())
```

OUTPUT

Open	High	Low	Close	Adj Close	Volume

Date						
2024-12-24	188.7500	189.75000	187.24000	188.63000	188.63000	34892100
2024-12-26	189.1000	190.00000	188.08000	189.88000	189.88000	30736100
2024-12-27	190.3000	191.30000	189.60000	190.95000	190.95000	28908100
2024-12-30	191.0000	192.00000	190.10000	191.85000	191.85000	31800100
2024-12-31	192.0000	193.00000	191.20000	192.73000	192.73000	32500000

Date	Predicted Close
2025-01-01	198.23
2025-01-02	198.29
2025-01-03	198.35
2025-01-04	198.41
2025-01-05	198.47

14. Future scope

This project lays a strong foundation for stock price forecasting using time series analysis. There are several meaningful opportunities to expand and enhance the project: