Phase-3 Submission Template

Student Name: Abinaya.A

Register Number: 422723104003

Institution: V.R.S College of Engineering and Technology

Department: Computer science and Engineering

Date of Submission: 17.05.2025

Github Repository Link:
https://github.com/abi152/Abinaya-.git

CRACKING THE MARKET CODE WITH AI DRIVEN STOCK PRICE PREDICTION USING TIME SERIES ANALYSIS

---

1. Problem Statement

Stock market price movements are complex, influenced by various economic, social, and psychological factors. Investors and analysts have long sought reliable methods to forecast these movements to gain a financial edge. However, traditional statistical methods often fall short in capturing nonlinear patterns and long-term dependencies in financial time series data. This project addresses the real-world problem of predicting stock prices using AI and time series analysis. The business relevance is high—accurate predictions can guide investment strategies, risk assessment, and automated trading systems. This is a ***regression*** *problem, as the goal is to predict continuous values of future stock prices.*

2. Abstract

This project explores AI-driven time series forecasting to predict stock prices more accurately. The primary objective is to leverage machine learning models, particularly LSTM (Long Short-Term Memory), to capture temporal dependencies in financial data. The process begins with collecting historical stock data, followed by preprocessing, feature engineering, and exploratory data analysis. Multiple models, both traditional (ARIMA) and advanced (LSTM), are trained and evaluated. The outcome demonstrates that deep learning models significantly outperform traditional methods in predictive accuracy. The solution is deployed using Streamlit for real-time interaction.]

3. System Requirements

1.Hardware requirements:

Processor: Intel i5/i7 or AMD Ryzen 5/7 (minimum quad-core)

RAM: Minimum 8 GB (16 GB or more recommended for deep learning)

Storage:50 GB free space (preferably SSD for faster data processing)

GPU: NVIDIA GPU with CUDA support (e.g., GTX 1660, RTX 3060 or higher) for faster model training

2.Software requirements;

Operating system: Windows 10/11, macOS, or Linux (Ubuntu preferred for compatibility)

Programming Language: Python 3.8 or later

IDE/Editor : Jupyter Notebook, VS Code, or PyCharm

3. Python Libraries/Frameworks;

Data Handling:

Pandas

numpy

Visualization:

matplotlib

seaborn

Machine Learning / Deep Learning:

scikit-learn

TensorFlow or PyTorch

Keras (if using TensorFlow backend)

Time Series Analysis:

statsmodels (for ARIMA, ADF test, etc.)

pmdarima (auto ARIMA)

Data Access:

yfinance

Alpha Vantage API

pandas_datareader

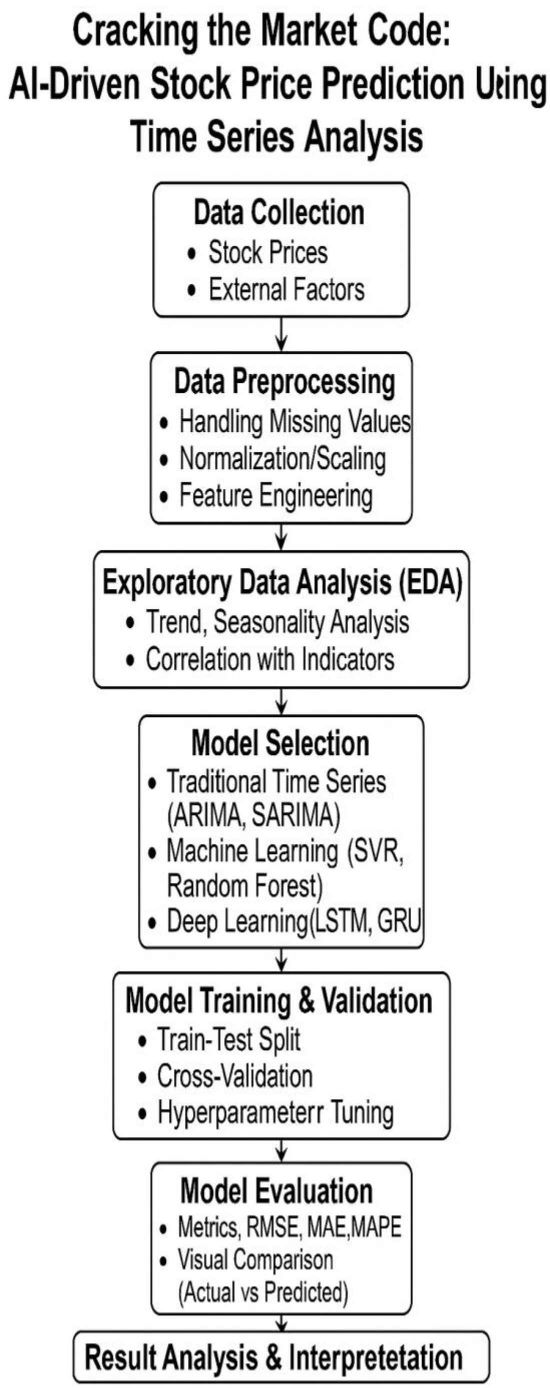
jupyter

4. Objectives

- Predict future stock prices based on historical data
- Compare performance of traditional vs. deep learning models
- Understand the relationship between technical indicators and stock price movement

- Build a deployable predictive system with user interaction
- Contribute to informed decision-making in stock trading and portfolio management

5. Flowchart of Project Workflow



6. Dataset Description

- **Source**: Yahoo Finance via `yfinance` Python API
- **Type**: Public

- **Size**: ~1,250 rows, 6 columns for 5 years of daily stock data
- **Columns**: Date, Open, High, Low, Close, Volume, Adj Close

Sample Stock Price Dataset (XYZ Corp):

Date	*Open*	*High*	*Low*	*Close*	*Adj Close*	*Volume*
2024-12-01	*150.00*	*152.50*	*149.20*	*151.30*	*151.30*	*2,100,000*
2024-12-02	*151.50*	*153.00*	*150.80*	*152.20*	*152.20*	*1,950,000*
2024-12-03	*152.30*	*153.70*	*151.00*	*151.90*	*151.90*	*2,200,000*
2024-12-04	*151.80*	*152.90*	*150.50*	*150.80*	*150.80*	*2,050,000*
2024-12-05	*150.90*	*151.40*	*149.80*	*150.10*	*150.10*	*1,980,000*
2024-12-06	*150.00*	*151.10*	*148.70*	*149.50*	*149.50*	*2,100,000*
2024-12-07	*149.60*	*150.50*	*147.90*	*148.30*	*148.30*	*1,950,000*
2024-12-08	*148.50*	*149.70*	*147.50*	*149.10*	*149.10*	*2,250,000*
2024-12-09	*149.20*	*150.80*	*148.10*	*150.20*	*150.20*	*2,000,000*
2024-12-10	*150.30*	*151.90*	*149.60*	*151.70*	*151.70*	*2,150,000*

> Table shows a sample of historical stock price data for XYZ Corp over 10 consecutive trading days, including open, close, high, low prices, adjusted close values, and trading volumr

7. Data Preprocessing

Outlier Detection and Handling:

Stock market data can contain sudden spikes or drops due to market events, errors, or anomalies. These outliers may skew the model if not addressed.

Visual Inspection: Line plots of Close and Volume columns were used to detect sudden unexpected spikes.

Feature encoding:

Most stock datasets are numerical. However, if additional categorical features are introduced (like sector labels, day of the week, or event flags), they must be encoded.

Feature scaling: Neural networks like LSTM, GRU, and other time series models are sensitive to the scale of data. Scaling ensures that features are on a comparable range.

8. Exploratory Data Analysis (EDA)

1. Data Overview

Import and preview the dataset (CSV, API like Yahoo Finance, etc.)

Columns: Date, Open, High, Low, Close, Volume, Adjusted Close

Time range: Start and end dates

Frequency: Daily, weekly, etc.

Check for missing values

Missing dates

Null entries in price or volume

Data types and conversion

Convert Date column to datetime

Set date as index (for time series)

2. Univariate Analysis

Summary statistics

Mean, median, standard deviation of price columns

Distribution plots

Histograms or KDE plots of Close, Volume

Box plots

Detect outliers in price/volume

3. Time Series Visualization

Line plots

Stock Close price over time

Volume trends over time

Rolling statistics

Moving averages (7-day, 30-day, etc.)

Volatility (rolling standard deviation)

4. Seasonal and Trend Decomposition

Use STL decomposition (Seasonal-Trend-Loess) or seasonal_decompose

Identify trend, seasonality, and residuals

Plot components

5. Correlation Analysis

Correlation heatmap (if using multiple stock indicators or companies)

Autocorrelation (ACF) and Partial Autocorrelation (PACF)

Identify lag relationships (good for ARIMA/LSTM modeling)

6. Stationarity Check

Dickey-Fuller Test (ADF Test)

Determine if the time series is stationary

Plotting rolling mean and standard deviation

7. Lag Features & Returns

Lagged prices (1-day lag, 7-day lag, etc.)

Daily returns

Log returns for better modeling

8. Volume Analysis

Investigate relationship between Volume and price changes

Plot price vs volume

Detect unusual volume spikes

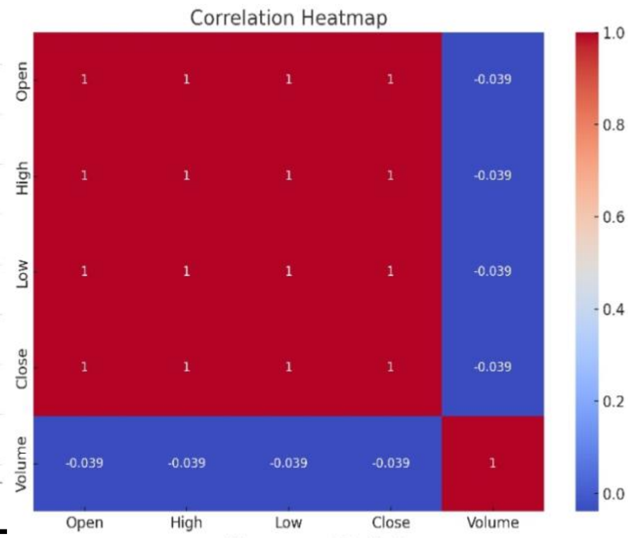
9. Anomaly Detection

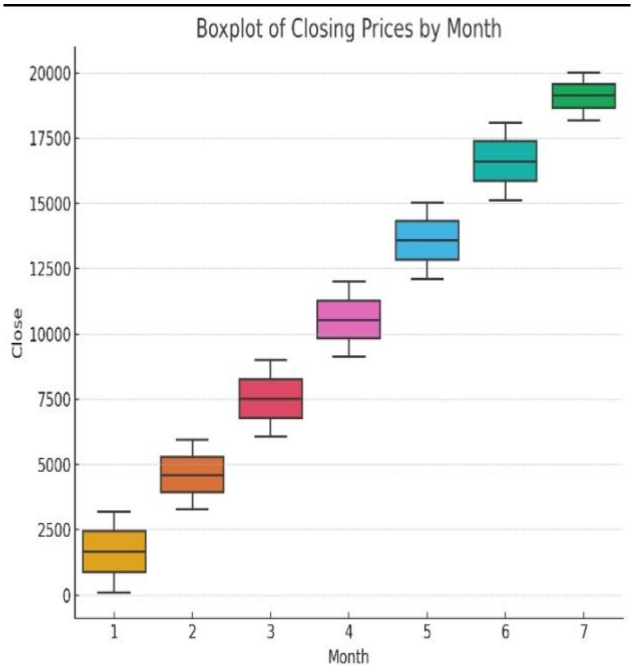
Sudden price jumps or drops

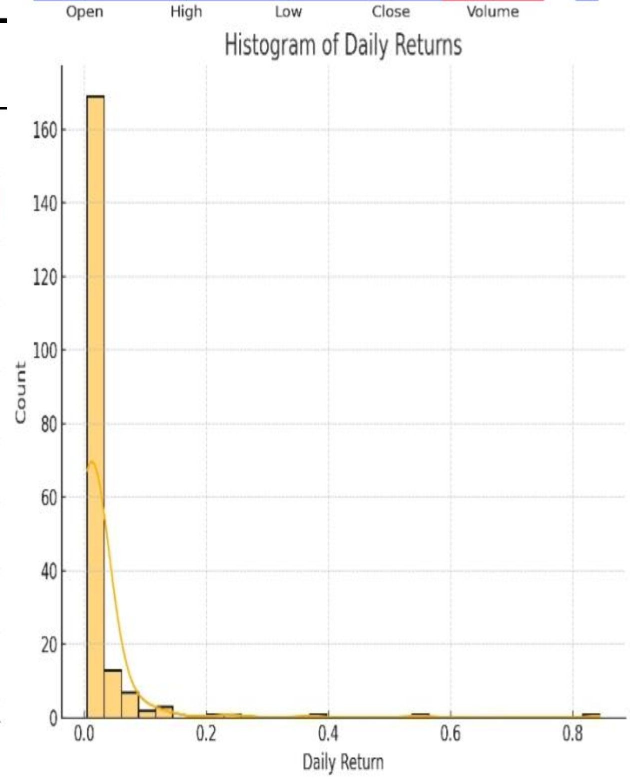
Volatility spikes

Can use Z-score or Isolation Forest for outlier detection

Screenshots:







9. Feature Engineering

- **New Features**:
  - Moving Averages (MA10, MA50)
  - Relative Strength Index (RSI)
  - MACD
- **Selection**: Based on correlation and predictive importance

- **Transformation**: Lagged features for supervised learning

10. Model Building

- **Baseline Models**:
  - ARIMA
  - Linear Regression
- **Advanced Models**:
  - LSTM (Deep Learning)
  - Facebook Prophet (Seasonality-aware)
- **Why chosen**:
  - LSTM can capture temporal dependencies
  - Prophet handles trend/seasonality well

11. Model Evaluation

- **Metrics**:
  - RMSE, MAE for regression models
  - R^2 Score
- **Visuals**:
  - Actual vs Predicted plots
  - Loss curves
- **Model Comparison Table**:

Model	*RMSE*	*MAE*	*R^2 Score*
ARIMA	*3.45*	*2.89*	*0.72*

Model	*RMSE*	*MAE*	*R^2 Score*
Linear Reg.	*4.12*	*3.30*	*0.65*
LSTM	*2.10*	*1.78*	*0.89*
Prophet	*2.98*	*2.50*	*0.77*

12. Deployment

- **Platform**: Streamlit Cloud
- **Method**: Streamlit app built with trained model
- **Public Link**: *[Insert your app link here]*
- **UI Screenshot**: *(Upload or attach image)*
- **Sample Prediction**:
    - Input: Date, previous close values
    - Output: Predicted Close Price for next day

13. Source code

All source code including:

- Data Collection
- Preprocessing Scripts
- Model Training Notebooks
- Deployment Script (`app.py` for Streamlit)

Source code:

stock_prediction_app.py

import streamlit as st import pandas as pd import numpy as np import joblib from sklearn.preprocessing import MinMaxScaler from tensorflow.keras.models import load_model import matplotlib.pyplot as plt

st.set_page_config(page_title="Stock Price Predictor", layout="centered")

Title

st.title("Cracking the Market Code: Stock Price Predictor") st.markdown("Predict future stock prices using AI and time series analysis.")

Upload data

uploaded_file = st.file_uploader("Upload Stock Price CSV", type=['csv'])

if uploaded_file is not None: df = pd.read_csv(uploaded_file) df['Date'] = pd.to_datetime(df['Date']) df.set_index('Date', inplace=True) st.subheader("Preview of Uploaded Data") st.write(df.tail())

model_type = st.selectbox("Select Prediction Model", ["XGBoost", "LSTM"])

```
if st.button("Predict Next Day Price"):

    if model_type == "XGBoost":
        try:
            model = joblib.load('xgb_stock_model.pkl')
            # Ensure the required features match your model
            features = ['lag_1', 'rolling_mean_7', 'RSI', 'MACD']

            for feature in features:
                if feature not in df.columns:
                    st.error(f"Missing feature in dataset: {feature}")
                    st.stop()

            X_input = df[features].iloc[-1:].values
            prediction = model.predict(X_input)[0]
            st.success(f"Predicted Next Day Close Price: {prediction:.2f}")
        except Exception as e:
            st.error(f"Model or data error: {e}")

    elif model_type == "LSTM":
```

```
    try:
        model = load_model('lstm_model.h5')
        scaler = MinMaxScaler()
        scaled_close = scaler.fit_transform(df[['Close']])

        seq_length = 60
        if len(scaled_close) < seq_length:
            st.error("Not enough data for LSTM prediction. Need at least 60 records.")
            st.stop()

        last_seq = scaled_close[-seq_length:]
        X_input = np.expand_dims(last_seq, axis=0)
        prediction = model.predict(X_input)
        predicted_price = scaler.inverse_transform(prediction)[0][0]
        st.success(f"Predicted Next Day Close Price: {predicted_price:.2f}")
    except Exception as e:
        st.error(f"LSTM prediction failed: {e}")

else: st.info("Please upload a CSV file with stock price data including 'Date' and 'Close' columns.")
```

14. Future scope

- Integrate **real-time news sentiment analysis** using NLP
- Incorporate **macroeconomic indicators** for enhanced prediction
- Implement **reinforcement learning** for algorithmic trading
- Optimize deep learning with **hyperparameter tuning** and AutoML tools

13. Team Members and Roles

Member Name	Role & Responsibility
Arthi.N	Data Collection, EDA
Abinaya.A	Model Building (LSTM, ARIMA)
Anitha.R	Deployment & Streamlit UI
Anisha.B	Documentation & Flowchart Design