Student Name: P.JANANI

Register Number: 510923104028

Institution:Global Institute of Engineering and Technology

Department: Computer Science and Engineering

Date of Submission: 08-05-25

GitHub Repository Link: https://github.com/Janani0117/phase-2.git

Project Title: Cracking the market code with AI-driven stock price prediction using time series analysis

1. Problem Statement

The stock market is inherently volatile and influenced by numerous dynamic and often unpredictable factors, making accurate price prediction a significant challenge for investors, analysts, and financial institutions. Traditional methods of stock price forecasting, which often rely on linear models or fundamental analysis, are limited in their ability to capture the complex, non-linear patterns present in financial time series data.

With the rise of artificial intelligence and machine learning, particularly time series modeling techniques such as Long Short-Term Memory (LSTM) networks, ARIMA, and Prophet, there is a growing opportunity to improve the precision of stock price predictions by leveraging historical data patterns, market trends, and sentiment analysis.

This project aims to develop an AI-driven model that applies advanced time series analysis techniques to predict future stock prices accurately. The objective is to enhance decision-making capabilities in financial markets by providing more reliable forecasts, reducing investment risks, and increasing returns.

2. Project Objectives

1.To collect and preprocess historical stock market data:

Gather time series data such as daily closing prices, trading volumes, and other relevant financial indicators from reliable sources (e.g., Yahoo Finance, Alpha Vantage).

2.To explore and analyze the underlying trends, patterns, and seasonality in stock price data:

Use statistical and visual methods to understand the temporal dynamics and characteristics of stock prices.

3.To build and compare AI-based time series forecasting models:

Implement and evaluate models like ARIMA, LSTM (Long Short-Term Memory), and Facebook Prophet to determine which provides the most accurate stock price forecasts.

4.To integrate additional features that may impact stock prices:

Incorporate technical indicators (e.g., moving averages, RSI), and optionally external data such as news sentiment or macroeconomic variables

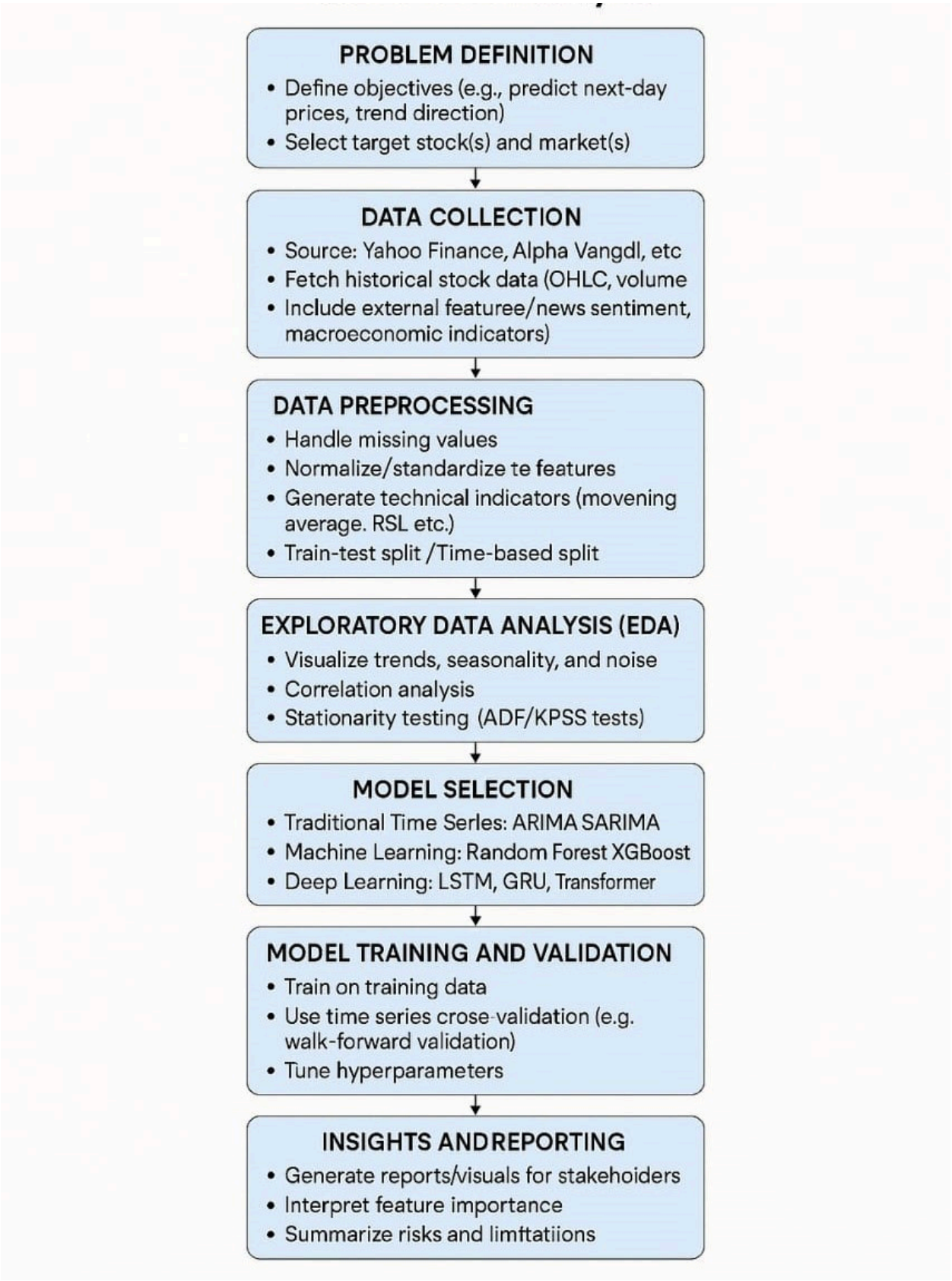
5. To optimize and validate model performance:

Use appropriate evaluation metrics (e.g., RMSE, MAE, MAPE) and cross-validation techniques to assess and fine-tune model predictions.

6. To visualize predictions and provide actionable insights:

Develop visualizations and dashboards to present actual vs. predicted prices and help users understand forecast trends and reliability.

3. Flowchart of the Project Workflow



4. Data Description

- **Dataset Name**: Historical Stock Market Dataset(e.g., AAPL Stock Prices)

- **Source**: Yahoo Finance via finance Python library
- **Type of Data**: Time-series financial data
- **Records**: ~5,000 daily records(approx.20 years)
- **Features:**

  1.date
  2.open price
  3.High price
  4.low price
  5.close price
  6.Adjusted price
  7.Volume

- **Target Variable**: Next day's Close or price trend direction(up/down)
- **Static or Dynamic**: Dynamic(updated regularly with new market data)
- **Attributes Covered**: Technical indicators(e.g.SMA,EMA,RSI,MACD)
- **Dataset Link**: Yahoo Finance AAPL Data

5. Data Preprocessing:

Data preprocessing is a critical step in time series forecasting, especially for stock price prediction, where the quality of input data directly impacts the accuracy of model predictions. The preprocessing steps for this project include the following:

Data Collection

- Source historical stock data from platforms such as Yahoo Finance, Alpha Vantage, or Quandl.

- Key fields: Date, Open, High, Low, Close, Volume, and Adjusted Close.

Data Cleaning

- **Handle missing values:** Fill or interpolate missing timestamps or price values using forward fill, backward fill, or linear interpolation.

- **Remove outliers:** Detect and treat price spikes or anomalies using z-score, IQR, or domain knowledge.

- **Adjust stock splits/dividends**: Ensure consistency in price series using the "Adjusted Close" where necessary.

Time Indexing

- Ensure the Date column is converted to a `datetime` format and set as the index.
- Confirm the data is sorted chronologically.

Normalization/Scaling

- Apply Min-Max Scaling or Standard Scaling to features, especially important for neural network models like LSTM.
- Ensure scaling is only done on training data and applied consistently to validation/test sets.

Train-Test Split

- Split the dataset chronologically into training and testing sets (e.g., 80/20).
- Optionally, use a validation set or time series cross-validation methods like walk-forward validation.

Stationarity Check (for ARIMA models)

- Perform ADF (Augmented Dickey-Fuller) test.
- Apply transformations (e.g., differencing, log transformation) if the data is non-stationary.

6. Exploratory Data Analysis (EDA)

- **Univariate Analysis**:

Univariate analysis focuses on examining a single variable at a time. When analyzing stock prices, this can involve:

Descriptive Statistics: Examining mean, median, standard deviation, and variance to understand stock price fluctuations.

Visualization Techniques: Using histograms, box plots, or density plots to explore distribution patterns.

Time Series Decomposition: Identifying seasonal trends, cyclic movements, and overall tendencies in historical stock prices.

Stationarity Testing: Applying tests like the Augmented Dickey-Fuller (ADF) test to determine if the stock price time series is stationary, which is crucial for many predictive models.

- **Bivariate & Multivariate Analysis:**

When forecasting stock prices using AI, **Bivariate and Multivariate analysis** help uncover deeper relationships between multiple financial indicators. Here's how they come into play:

Bivariate Analysis

Bivariate analysis examines the relationship between two variables. In stock price prediction, this could involve:

- **Correlation Analysis:** Measuring how closely stock price movements relate to another factor like trading volume or moving averages.
- **Scatter Plots**: Visualizing trends between stock prices and indicators such as interest rates or macroeconomic factors.
- **Regression Models**: Identifying linear dependencies between stock prices and secondary variables like market sentiment or commodity prices.

Multivariate Analysis

Multivariate analysis expands on bivariate techniques by exploring more than two variables at once. Key approaches include:

- **Multiple Regression Models:** Capturing relationships between stock price and multiple independent factors like interest rates, inflation, and historical trends.
- **Principal Component Analysis (PCA)**: Reducing dimensionality by selecting key features that influence stock movement.
- **Time Series Models (VAR, LSTM, ARIMA)**: Using multiple variables together to build AI-driven forecasting models with enhanced accuracy.

- **Key Insights**:

Model Selection: Choose the most suitable AI model for stock price prediction, considering factors like data quality,complexity and interpretability.

Hyperparameter tuning: Optimize model hyperparameters to improve prediction accuracy and robustness.

Backtesting: Evaluate the models performance on historical data to assess its reliability and potential for future predictions.

Risk management: Consider incorporating risk management strategies to mitigate potential losses and maximize gains.

7. Feature Engineering

- Moving Average: Calculate the short-term and long-term moving averages to capture trends.
- Relative Strength Index(RSI): measure the magnitude of recent price changes to identify overbought or oversold conditions.
- Moving average Convergence Divergence(MACD): Identify trends and potential buy/sell signals.
- Normalization: Scale features to a common range to improve model performance.
- Market data: Use data like trading volume,open interest or order book data.

8. Model Building

- **Algorithms Used**:
  - Long Short-Term Memory (LSTM) Neural Networks
  - LSTM is a type of Recurrent Neural Network (RNN) particularly well-suited for time series forecasting due to its ability to learn long-term dependencies and temporal patterns in sequential data like stock prices

- **Model Selection Rationale**:
  - Stock prices are sequential with temporal dependencies.
  - LSTM handles vanishing gradient problems better than standard RNNs.
  - It can capture seasonality, trends, and sudden shifts in market behavior better than traditional statistical models.
- **Train-Test Split**:
  - Typical Split: 80% Train, 20% Test
  - Time Series Approach:
  - Data is not shuffled; split chronologically.

- **Evaluation Metrics**:
  - **Mean Squared Error (MSE) – Measures average squared difference between predicted and actual values.**
  - **Root Mean Squared Error (RMSE) – Easier to interpret in original units.**
  - **Mean Absolute Percentage Error (MAPE) – Shows percentage accuracy; useful for stock prices.**
  - **R-squared (R^2) – Measures how well the model explains the variance in the data.**

9. Visualization of Results & Model Insights

1. Visualization of Results

a) Predicted vs Actual Prices

- **Line Plot**

X-axis: Dates

Y-axis: Stock Price

Plot both actual and predicted prices

```
import matplotlib.pyplot as plt

plt.figure(figsize=(14,6))
plt.plot(actual_prices, label='Actual Price')
plt.plot(predicted_prices, label='Predicted
Price')
plt.title('Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

b) Error Distribution

- **Histogram of residuals** (actual - predicted)
- **Insights**: Check for bias or systematic error

```
errors = actual_prices - predicted_prices
plt.hist(errors, bins=50)
plt.title('Prediction Error Distribution')
plt.xlabel('Error')
plt.ylabel('Frequency')
plt.show()
```

c) Rolling Window Prediction Accuracy

- Plot MAPE or RMSE over time using a rolling window (e.g., 30 days)
- Insight: Detect periods where the model underperforms (volatility, news, etc.)

2. Model Insights

a) Feature Importance (if using hybrid models like LSTM + Attention or feature-augmented inputs)

- Use SHAP or attention weights to visualize which features (volume, moving averages, sentiment scores) influenced the prediction most.

b) Temporal Patterns

- Detect how well the model captures trends, cycles, and volatility.
- Use seasonal decomposition (e.g., STL) on actual vs predicted series for comparison.

c) Prediction Confidence (Uncertainty)

Plot prediction intervals if your model outputs them (e.g., using Bayesian LSTM or Monte Carlo Dropout).

10. Tools and Technologies Used

- **Programming Language**: Python – Most popular due to vast ecosystem of libraries for machine learning and time series analysis.
- **Notebook Environment**: Jupyter Notebook – For interactive development
  Google Colab – Free GPU-powered environment
  VS Code / PyCharm – Popular Python IDEs
- **Data Manipulation & Analysis**
  Pandas – For data loading, manipulation, and preprocessing
  NumPy – For numerical computations
  Scikit-learn – For preprocessing, metrics, and traditional model
- **Deep Learning Framework**
  TensorFlow / Keras – For building LSTM, GRU, and Transformer-based models
  PyTorch – Alternative to TensorFlow, preferred for research-grade models
- **Visualization Tools:**
  Matplotlib & Seaborn – For basic and advanced plotting
  Plotly – For interactive time series charts
  TensorBoard – For model training visualization

11. Team Members and Contribution

NAME	CONTRIBUTION
P.JANANI	Data preprocessing
H.ANGEL	Exploratory Data Analysis (EDA)
K.HABITHA	Featuring engineering & Model building
R.LATHIKA	Visualization of results and tools and technology used