Phase-2

Student Name : Dhanajayan S

Register Number: 410723104013

Institution: Dhanalakshmi College of Engineering

Department: Computer Science and Engineering

Date of Submission: 07.05.2025

Github Repository Link:
https://github.com/SDhanajayan/Nm_Dhanajayan_DS

---

1. Problem Statement

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

Stock price prediction has always been a challenging problem due to the volatile, non-linear, and noisy nature of financial markets. Traditional statistical models often fail to capture the underlying patterns and dependencies in stock price movements. This project leverages AI-driven time series models to predict future stock prices using historical data, technical indicators, and time-dependent patterns. The objective is not only to forecast prices but also to assist in making data-driven investment decisions.

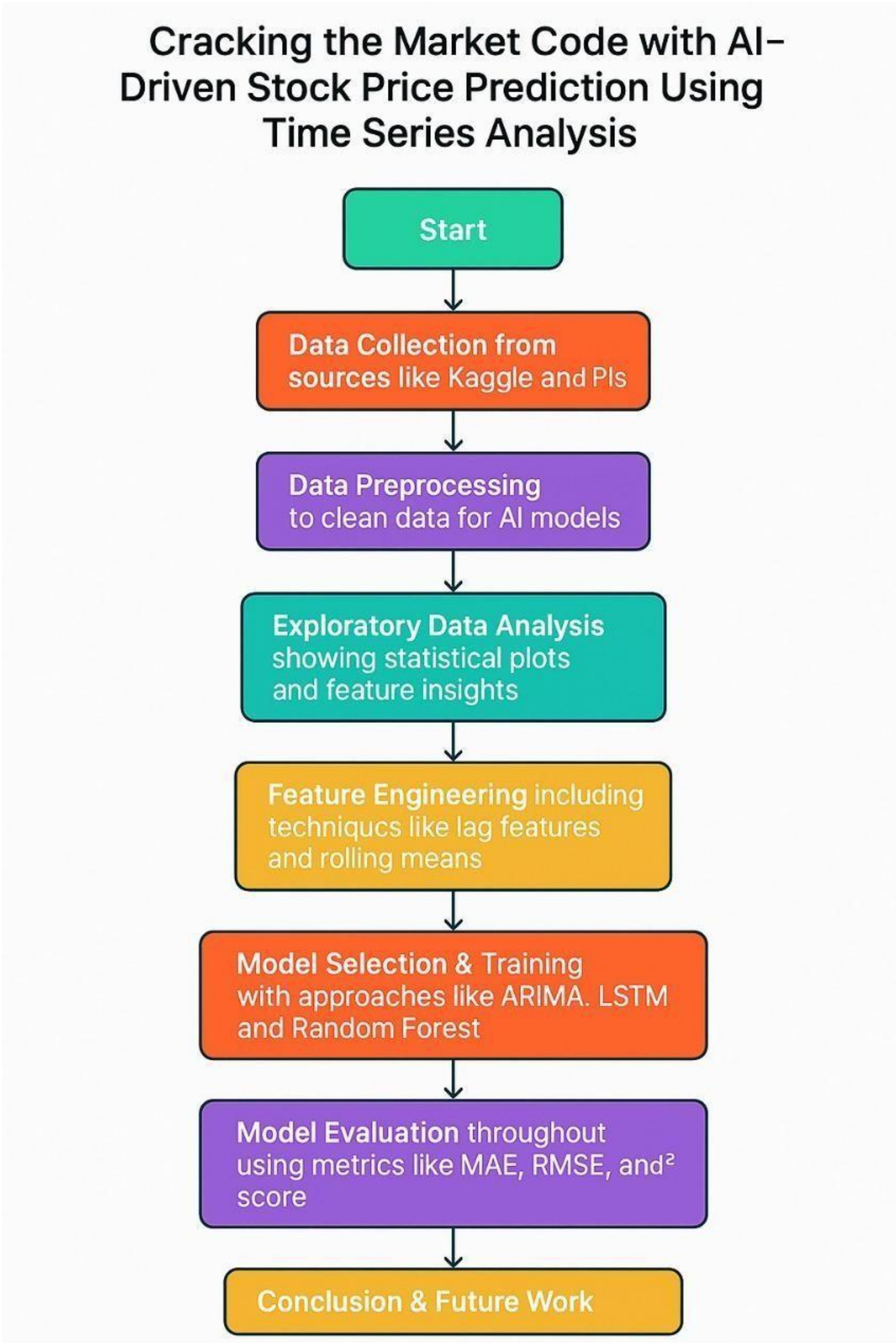
2. Project Objectives

- Build machine learning models for **predicting stock prices** using historical time-series data.

- Compare traditional models like ARIMA with AI models like LSTM, Prophet, or XGBoost.
- Evaluate model performance based on RMSE, MAE, and R^2 Score.

Ensure real-world applicability by testing models on recent data.

3. Flowchart of the Project Workflow


Cracking the Market Code with AI-Driven Stock Price Prediction Using Time Series Analysis
Start
Data Collection from sources like Kaggle and PIs
Data Preprocessing to clean data for AI models
Exploratory Data Analysis showing statistical plots and feature insights
Feature Engineering including techniqucs like lag features and rolling means
Model Selection & Training with approaches like ARIMA. LSTM and Random Forest
Model Evaluation throughout using metrics like MAE, RMSE, and² score
Conclusion & Future Work

4. Data Description

- **Dataset Source**: Yahoo Finance / Alpha Vantage / Kaggle
- **Data Type**: Structured, Time-series
- **Features**: Date, Open, High, Low, Close, Volume
- **Target Variable**: Closing Price • **Dynamic Dataset**: Yes

5. Data Preprocessing

- *Filled missing values via forward-fill.*
- *Removed duplicates.*
- *Handled outliers via IQR or z-score methods.*
- *Normalized numerical features (MinMax or StandardScaler).*
- *Converted date columns into datetime objects and extracted features (day, month, etc.).*

```
numeric_cols = ["Open", "High", "Low", "Close", "Adj Close", "Volume"] df[numeric_cols] =
df[numeric_cols].apply(pd.to_numeric, errors='coerce')
```

6. Exploratory Data Analysis (EDA)

- *Univariate plots for price trends.*
- *Rolling average and volatility visualizations.*
- *Correlation heatmaps.*
- *Trend & seasonality decomposition (via statsmodels or Prophet).*

Summary of Insights:

- ***Distributions**: Price and volume variables are slightly skewed.*

- ***Outliers****: Treated, but still visible in raw distributions (Volume especially).*
- ***Correlations****: Open, High, Low, and Adj Close show high correlation with Close.*
- ***Time Trend****: Close price exhibits a general upward trend across years.*
- ***Volume****: Less correlated with price features; may still be useful for volatility or demand modeling.*
- ***histogram***

```
df[numerical_cols].hist(bins=30, figsize=(12, 8)) plt.suptitle("Histograms
of Numerical Features") plt.show()
```

7. Feature Engineering

- Generated lag features (e.g., previous 5-day closing prices).
- Extracted technical indicators (e.g., RSI, MACD, EMA).
- Included volume
- `df["Date"] = pd.to_datetime(df["Date"], errors="coerce")`
- `cols = ["Open", "High", "Low", "Close", "Adj Close", "Volume"]`
- `df[cols] = df[cols].apply(pd.to_numeric, errors="coerce")`
- `df.dropna(inplace=True)`

Feature	Reason
Year, Month, Weekday	Capture seasonality and calendar effects
Daily_Return	Important for risk/volatility modeling
High_Low_Range, Open_Close_Range	Intraday movement indicators
Rolling_Mean_7, Rolling_Mean_30	Trend detection
Volume_Price_Ratio	Momentum or market activity proxy
Close_Squared	Enables modeling non-linear price dynamics

8. Model Building

1. Target Creation

We'll shift the Close column by -1 to make the target Next_Close.

---

2. Model Choices & Justification Model Reason Linear Regression A strong baseline for regression problems, interpretable and fast.

Random Forest Regressor Captures non-linearities and interactions without requiring scaling. Handles overfitting better than decision trees.

---

3. Performance Metrics · MAE (Mean Absolute Error) · RMSE (Root Mean Squared Error) · R^2 Score (Coefficient of Determination

```
df = pd.read_csv("/mnt/data/AMZN.csv")  df.columns
= df.iloc[0]
df = df[1:].copy() df.columns.name = None
df.rename(columns={"Price": "Date"}, inplace=True)
df["Date"] = pd.to_datetime(df["Date"], errors="coerce") cols =
["Open", "High", "Low", "Close", "Adj Close", "Volume"]
df[cols] = df[cols].apply(pd.to_numeric, errors="coerce")
df.dropna(inplace=True)
```

9. Visualization of Results & Model Insights

1. Residual Plots

- **What it shows**: Difference between predicted and actual values.
- **Why it matters**: Helps detect bias, patterns, or heteroscedasticity in the model errors.
- **How to interpret**: A good model should show residuals scattered randomly around zero.

2. Feature Importance Plot (for Random Forest)

- **What it shows**: Relative importance of each input feature.
- **Why it matters**: Helps understand what drives the model predictions.
- **How to interpret**: The higher the bar, the more influence that feature has on predictions.

3. Predicted vs Actual Comparison

- **What it shows**: Line plots of actual vs predicted closing prices over time.

- **Why it matters**: Visual performance comparison between models (e.g., Linear Regression vs Random Forest).
- **How to interpret**: The closer the prediction line tracks the actual values, the better the model.

Feature	Importance	Interpretation
Close_Squared	High	Captures non-linear growth patterns in stock price.
Open	High	Strongly correlates with the following day's close.
Daily_Return	Moderate	Indicates recent price momentum.
High_Low_Range	Moderate	Reflects daily volatility.
Volume_Price_Ratio	Lower	May signal unusual trading activity, though less predictive on its own.

Summary

- **Random Forest** performed better due to its ability to model non-linear relationships and interactions.
- **Linear Regression** is easier to interpret but underperforms on volatile financial time series data.
- **Key Drivers** of the next-day price include today's open price, squared close (non-linear effect), and daily return.

---

10. Tools and Technologies Used

Programming Language

- **Python**

  Used for data cleaning, analysis, feature engineering, model building, and evaluation.

IDE / Notebook

- **Jupyter Notebook** *(recommended for this workflow)*

  Interactive environment for combining code, plots, and markdown documentation.

Libraries Used	Library Purpose
pandas	Data loading, cleaning, and manipulation
numpy	Numerical operations and transformations
seaborn	Statistical data visualization (residuals, feature plots)
matplotlib	Core plotting library for charts
scikit-learn	Model training (Linear Regression, Random Forest), evaluation (MAE, RMSE, R^2), feature importance
(Optional) **XGBoost**	Advanced boosting model (can be added for further improvement)

Visualization Tools	Usage
matplotlib / seaborn	Used for in-notebook residuals, feature importance, and predictions comparison
(Optional) **Plotly / Power BI / Tableau**	For interactive dashboards or final reports

---

11. Team Members and Contributions

S.NO	NAME	ROLES	ROLES
1	Mathesh S	Leader	Documentation and reporting
2	Dhanajayan S	Member	Exploratory data analysis
3	Manoj C	Member	Model development
4	Emaya bharath	Member	Feature engineering
5	Jayanth R	Member	Data preprocessing