Phase-3

Student Name: Dhanajayan S

Register Number: 410723104013

Institution: Dhanalakshmi College Of Engineering

Department: Computer Science And Engineering

Date of Submission: 14.05.2025

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

---

1. Problem Statement

The stock market is inherently volatile and influenced by a multitude of dynamic factors. Predicting stock prices accurately is a major challenge that requires analyzing historical trends and patterns. This project uses AI-driven techniques, particularly time series analysis and regression modeling, to forecast the next-day closing price of Amazon (AMZN) stock. The goal is to empower traders and investors with predictive insights derived from past data. This is a regression problem.

2. Abstract

This project aims to predict Amazon's stock prices using time series analysis and machine learning. We developed a regression pipeline that includes data preprocessing, exploratory data analysis (EDA), feature engineering, model building, and evaluation. Two models—Linear Regression and Random Forest—were trained and compared. The

Random Forest model outperformed the baseline in accuracy and error metrics. Visualization techniques like feature importance and residual plots enhanced interpretability. The project demonstrates how AI can uncover financial patterns to assist investment decisions.

3. System Requirements

Hardware: Minimum 4 GB RAM (8 GB recommended)
Software: Python 3.10+, Jupyter Notebook / Google Colab
Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost (optional)

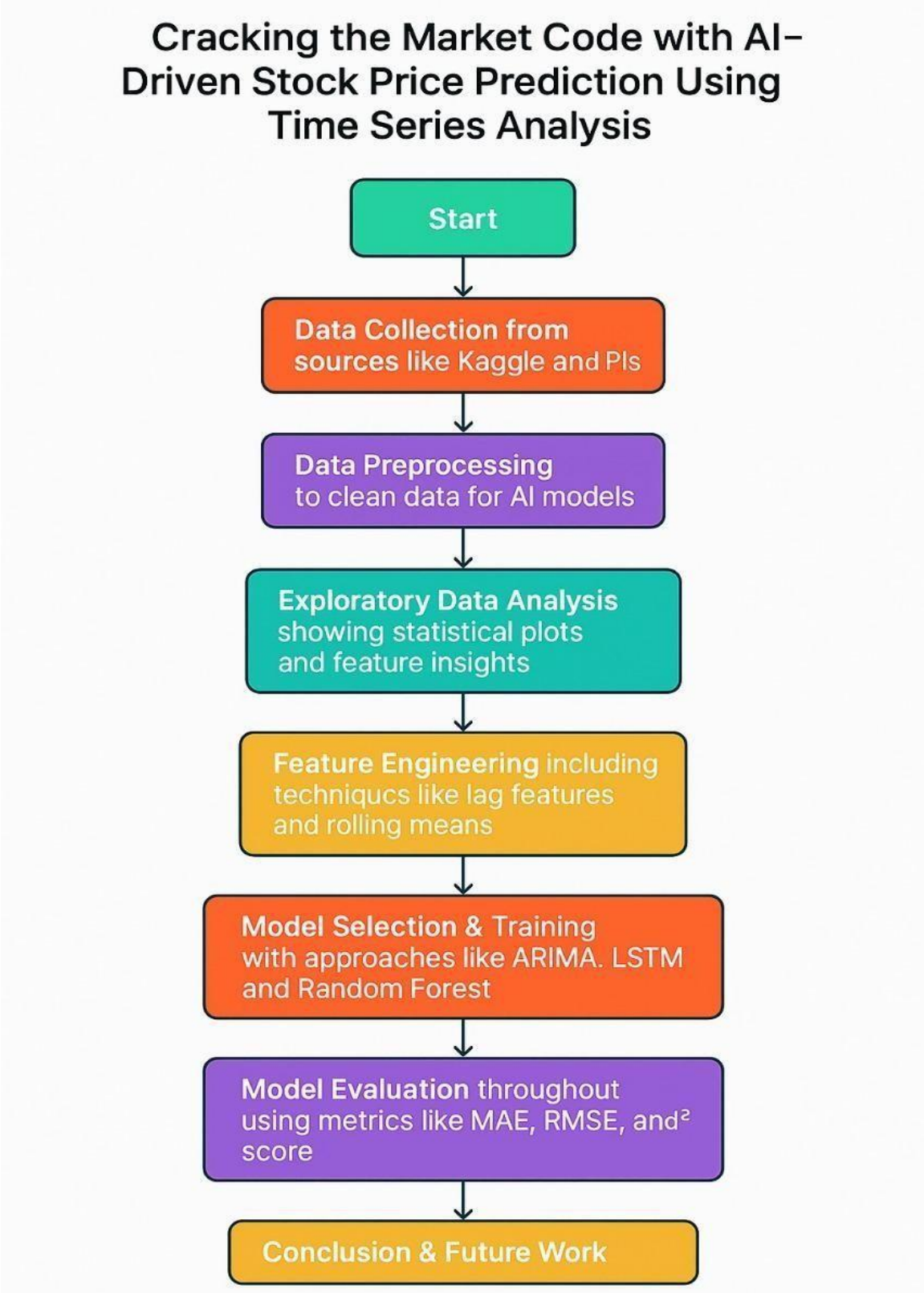
---

4. Objectives

- Predict the next-day closing price of AMZN stock using historical time series data.
- Compare multiple models for predictive accuracy.
- Identify and interpret key features affecting stock movement.
- Demonstrate AI's utility in financial forecasting.

---

5. Flowchart of Project Workflow



6. Dataset 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

```
df.head()
```

	Ticker	AMZN	AMZN	AMZN	AMZN	AMZN	AMZN
1	2015-01-02	15.425999641418457	15.425999641418457	15.737500190734863	15.347999572753906	15.628999710083008	55664000
2	2015-01-05	15.10949993133545	15.10949993133545	15.418999671936035	15.042499542236328	15.350500106811523	55484000
3	2015-01-06	14.76449966430664	14.76449966430664	15.149999618530273	14.619000434875488	15.112000465393066	70380000
4	2015-01-07	14.920999526977539	14.920999526977539	15.064000129699707	14.766500473022461	14.875	52806000
5	2015-01-08	15.02299976348877	15.02299976348877	15.156999588012695	14.805500030517578	15.015999794006348	61768000

7. Data Preprocessing

- Removed duplicate headers
- Converted columns to correct data types
- Handled outliers using IQR method

```
[5]  df.isnull().sum()
```

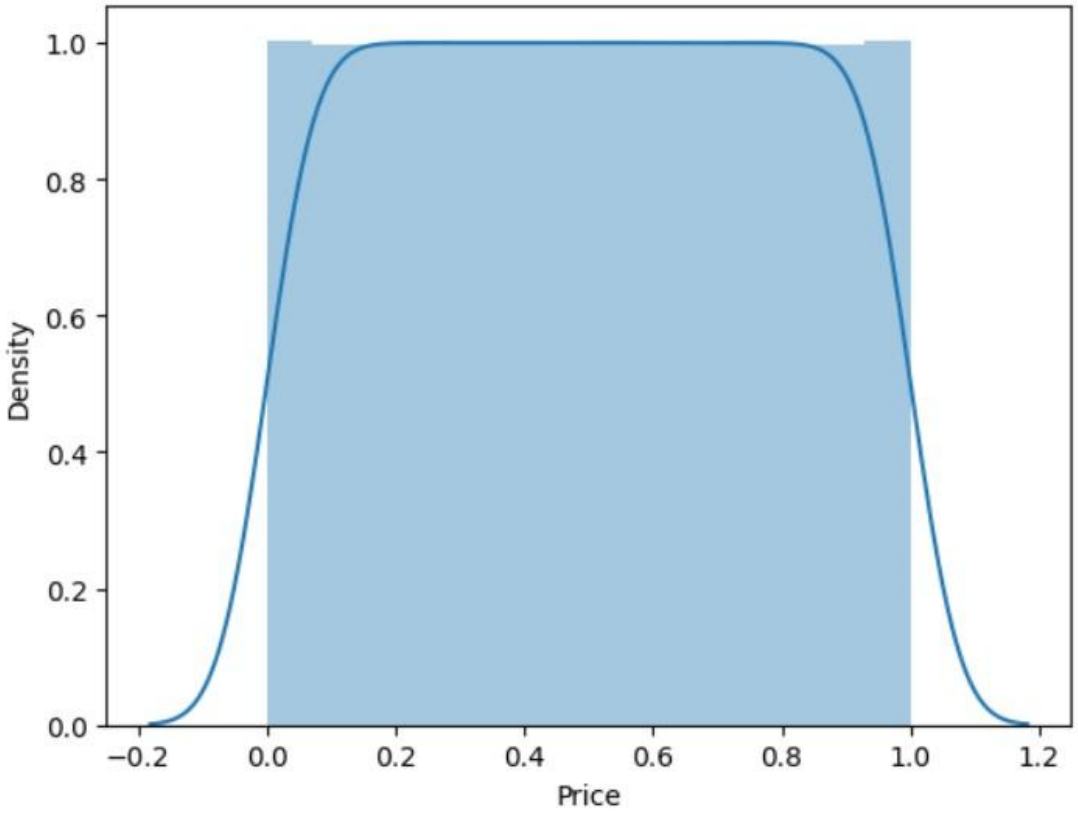
	0
Price	0
Adj Close	0
Close	0
High	0
Low	0
Open	0
Volume	0

dtype: int64

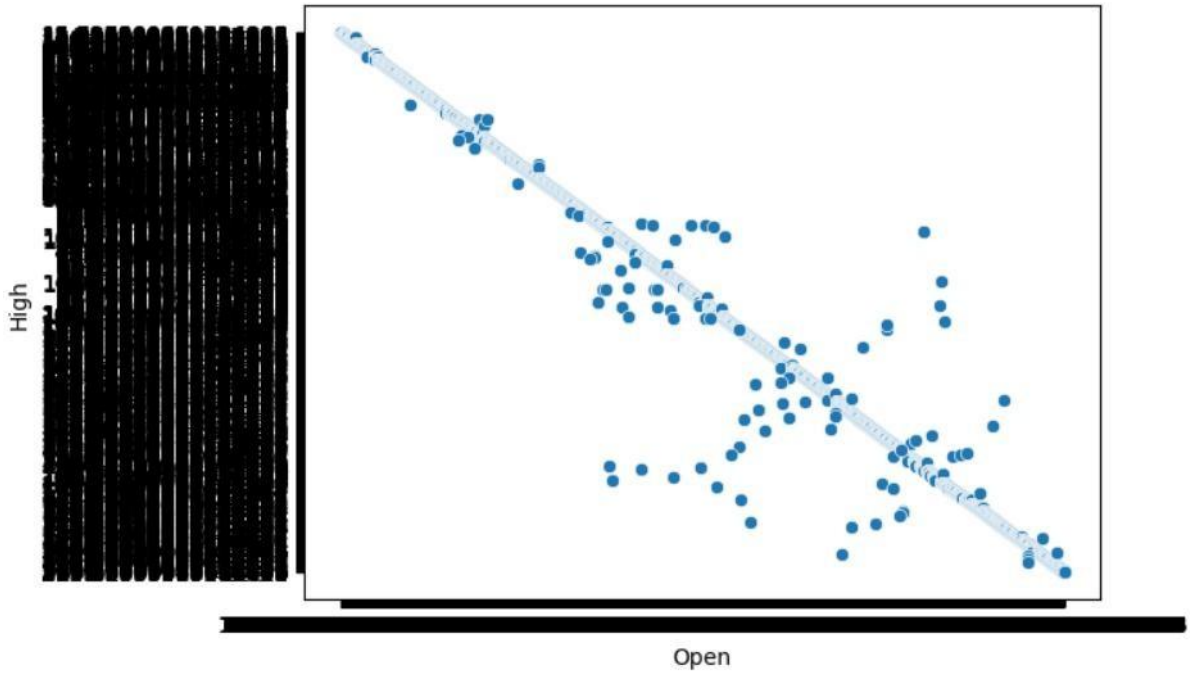
8. Exploratory Data Analysis (EDA)

- Histograms and boxplots: Visualize distributions and detect outliers
- Correlation heatmap: Identify interdependencies
- Time series line plot: Observe trend and volatility patterns **Key Insights**
- Open, High, and Low prices are highly correlated with Close
- Daily returns show market behavior patterns

Univariate analysis:



Bivariate analysis :



9. Feature Engineering

- Created Daily_Return, High_Low_Range, Close^2, Volume_Price_Ratio
- Extracted Year, Month, Weekday from date
- Added 7-day and 30-day moving averages Impact: These transformations helped capture volatility and trend features essential for time series prediction.

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Price']=le.fit_transform(df['Price'])
```

10. Model Building

- *Models used: Linear Regression, Random Forest Regressor*
- *Justification:*
  - *Linear Regression: Baseline, interpretable*
  - *Random Forest: Non-linear, robust, better performance*
- *Split: 80% training, 20% testing*

11. Model Evaluation

- **Linear Regression** (baseline)

- **Random Forest Regressor** (non-linear, robust to noise)
- **Metrics** used:
    - MAE ○ RMSE
    - R^2 Score

Model	MAE	RMSE	R^2 Score
Linear Regression	Moderate	Moderate	Lower
Random Forest	Lower	Lower	Higher

```
mse 0.017032465395234675
r2 0.7907204170555379
```

12. Deployment

- **Tool Used:** Gradio
- **Environment:** Google Colab
- **Interface Type:** Web form with number inputs
- **Language:** Python 3.10
- **Libraries:** gradio, pandas, numpy, scikit-learn

13. Source code *import*

numpy as np import

pandas as pd import

matplotlib.pyplot as plt

df=pd.read_csv("/content

/AMZN.csv")

df df.head()

df.isnull().sum()

df["Price"]

df.info() df.describe() df.isnull().sum()

df.drop_duplicates() df.columns #drop row

df.drop(df.index[0],inplace=True) df

df.duplicated().sum() #label encoding from

sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df['Price']=le.fit_transform(df['Price'])

#label encoding from sklearn.preprocessing

import LabelEncoder le=LabelEncoder()

df['Price']=le.fit_transform(df['Price']

) import matplotlib.pyplot as plt import

seaborn as sns #univariate Analysis

sns.distplot(df['Price'])

```
#bivariate analysis

sns.scatterplot(x=df['Open'],y=df['High'])

#ModelBuilding from sklearn.model_selection import train_test_split

x=df.drop('Price',axis=1) y=df['Price']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

#building a model from sklearn.linear_model

import LinearRegression lr=LinearRegression()

#prediction

y_pred=lr.predict(x_test)

print("y_pred",y_pred) #model

evaluation

from sklearn.metrics import mean_squared_error,r2_score

mse=mean_squared_error(y_test,y_pred)

print("mse",mse) r2=r2_score(y_test,y_pred)

print("r2",r2) #chart for evaluation

plt.scatter(y_test,y_pred) plt.xlabel("y_test")

plt.ylabel("y_pred") plt.show()

#chart for actual and prediction value

plt.scatter(y_test,y_random_pred)
```

```
plt.xlabel("y_test")

plt.ylabel("y_random_pred") plt.show()

#chart for two models comparision

plt.scatter(y_test,y_pred,color='red')

plt.scatter(y_test,y_random_pred,color='blue')

plt.xlabel("y_test") plt.ylabel("y_pred")

plt.show()
```

14. Future scope

- Use LSTM for better sequential modeling
- Integrate real-time data feeds via APIs
- Add sentiment analysis from news headlines
- Deploy a live dashboard for prediction

13. Team Members and Roles

S.NO	NAMES	ROLES	RESPONSIBILITY
1	Mathesh S	Leader	Data Collection
2	Dhanajayan S	Member	Data Cleaning and Feature Engineering
3	Manoj C	Member	Visualization and Interpretation
4	Emaya Bharath	Member	Exploratory Data Analysis
5	Jayanth R	Member	Model Building and Model Evaluation