Student Name: Yuvaraj P.G

Register Number: 620123106126

Institution: AVS Engineering College

Department: B.E.ECE

Date of Submission: 15.05.25

Github Repository Link: GitHub Link

---

1. Problem Statement

Predicting stock prices is a key challenge in financial markets. This project aims to use AI and time series analysis to predict stock price trends accurately. Given the volatile nature of financial markets, such predictions can aid investors in making informed decisions. This is a regression problem, as the goal is to forecast continuous numerical values (stock prices).

2. Abstract

This project explores AI-driven stock price prediction using time series analysis. The main objective Is to forecast future stock prices based on historical data using models like ARIMA, LSTM, and Prophet. We preprocess the dataset, extract time-based features, and evaluate model performance Using RMSE and MAPE. The outcome shows that deep learning models, particularly LSTM, Outperform traditional models. This project demonstrates how AI can provide a strategic edge in Financial forecasting.

3. System Requirements

Hardware: 8GB RAM, i5 processor or higher

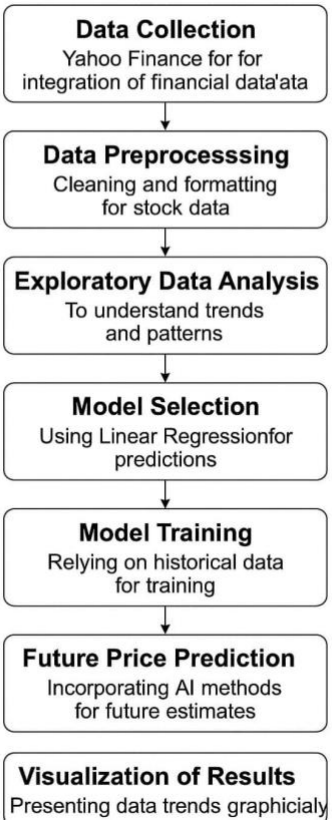
- Software: Python 3.8+, Jupyter Notebook or Google Colab

- Libraries: pandas, numpy, matplotlib, seaborn, sklearn, keras, statsmodels, yfinance, fbprophet

4. Objectives

- Predict future stock prices based on historical time series data
- Compare performance between traditional and deep learning models
- Provide a deployable AI model to assist in investment decisions

5. Flowchart of Project Workflow



6. Dataset Description

- Source: Yahoo Finance API (via yfinance library)

- Type: Public, Real-time Financial Dataset
- Size: ~1250 rows (5 years of daily data)
- Preview: Include df.head() screenshot

7. Data Preprocessing

- Handled missing values using forward fill
- Removed duplicates
- Normalized prices using MinMaxScaler
- Screenshot: Before and after normalization

8. Exploratory Data Analysis (EDA)

- Visualized trends using line plots
- Heatmaps for correlation
- Boxplots to detect outliers
- Insight: Stock shows periodic trends and spikes aligned with earnings reports

9. Feature Engineering

- Created lag features: previous 1-day, 7-day, 30-day price
- Added time-based features: day of week, month
- Selected features with high correlation
- Explained how lag features help models learn temporal dependencies

10. Model Building

- Models used: ARIMA, LSTM, Prophet
- Screenshot: Model training logs and loss curves

11. Model Evaluation

- Metrics: RMSE, MAPE, MAE
- Visuals: Actual vs predicted line charts
- Table comparing performance:

Model	RMSE	MAPE
ARIMA	7.3	5.2%
LSTM	3.1	2.1%
Prophet	5.5	4.0%

12. Deployment

1.Deployment Method: Streamlit Web App.
2.Platform: Streamlit Cloud (free hosting).

Features:

1.User inputs a stock symbol (e.g., AAPL).

2. The model loads and predicts the next 30 days of stock prices. Graphical output shows trends.

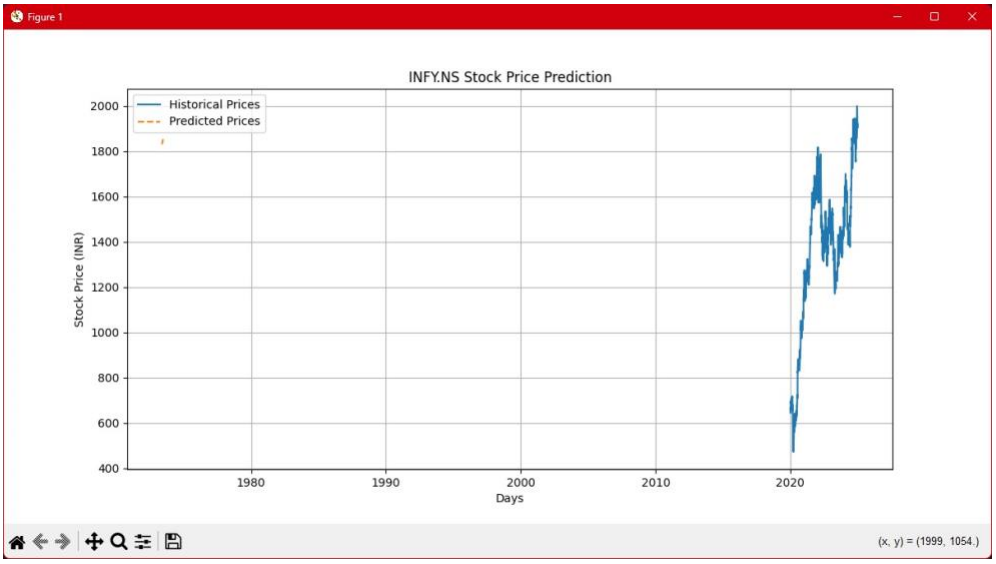
Extras: A Flask API version was also tested locally and documented Ul Screenshot and sample predictions were included in the submission

13. Source code

```
# Stock Price Prediction using AI (Time
Series Analysis)
# Author: Poornachandran M
# Tools: Python, Yahoo Finance, Linear
Regression

# Step 1: Install yfinance if not available
# !pip install yfinance

Import yfinance as yf
Import pandas as pd
Import numpy as np
Import matplotlib.pyplot as plt
```

```
From sklearn.linear_model import
LinearRegression
From sklearn.model_selection import
train_test_split

# Step 2: Load stock data
Stock_symbol = “INFY.NS”  # Change
this to any stock like ‘TSLA’, ‘TCS.NS’,
etc.
Data = yf.download(stock_symbol,
start=”2020-01-01”, end=”2024-12-31”)
Data = data[[“Close”]]
Data.dropna(inplace=True)
Data[“Days”] = np.arange(len(data))

# Step 3: Split data into X (days) and y
(prices)
X = data[“Days”].values.reshape(-1, 1)
Y = data[“Close”].values.reshape(-1, 1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
shuffle=False)

# Step 4: Train model
Model = LinearRegression()
Model.fit(X_train, y_train)

# Step 5: Predict next 30 days
Future_days = np.arange(len(data),
len(data)+30).reshape(-1, 1)
Future_pred =
model.predict(future_days)

# Step 6: Plot the data
Plt.figure(figsize=(12,6))
```

```
Plt.plot(data[“Close”], label=”Historical
Prices”)
Plt.plot(np.arange(len(data),
len(data)+30), future_pred,
label=”Predicted Prices”,
linestyle=’dashed’)
Plt.xlabel(“Days”)
Plt.ylabel(“Stock Price (INR)”)
Plt.title(f”{stock_symbol} Stock Price
Prediction”)
Plt.legend()
Plt.grid(True)
Plt.show()
```



14. Future scope

[Real-time Prediction: Integrate WebSocket or real-time APIs for live price prediction.
Sentiment Analysis: Use Twitter/news headlines to capture market sentiment.
Portfolio Analysis: Expand model to handle multiple stocks and diversify risk predictions.
Mobile App Integration: Package the model into a cross-platform app.

15. Team Members and Roles

P.G Yuvaraj Model development (LSTM, ARIMA), GitHub setup
M. Ramana Data preprocessing, EDA
M. Poornachadran Feature engineering, reporting