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

GITHUB LINK : https://github.com/Sandhiya17-gce/Sandhiya-muthu.git

STUDENT NAME : M Sandhiya

REGISTER NUMBER : 620623121028

INSTITUTION : Ganesh College of Engineering

DEPARTMENT : BME

DATE OF SUBMISSION : 20-05-2025

1. Problem Statement

Stock price prediction is a complex and highly dynamic task due to the volatility of the financial market. Accurately forecasting stock prices helps investors and financial institutions make informed decisions, reduce risks, and maximize returns. This project aims to predict future stock prices using AI-driven time series analysis, framing the problem as a regression task.

2. Abstract

This project focuses on predicting stock prices using AI and time series techniques. We collected historical stock data from Yahoo Finance, performed preprocessing, explored trends through EDA, engineered relevant features, and built various predictive models including LSTM and ARIMA. The model was evaluated using RMSE and MAE, then deployed via Streamlit for real-time forecasting. The final result is an interactive tool that aids in predicting next-day stock prices with reasonable accuracy.

3. System Requirements

Hardware: 8 GB RAM, Intel i5 or better

Software:

- Python 3.10+

- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, keras, statsmodels, yfinance, streamlit

- IDE: Google Colab / Jupyter Notebook

4. Objectives

- Forecast future closing prices of a selected stock.

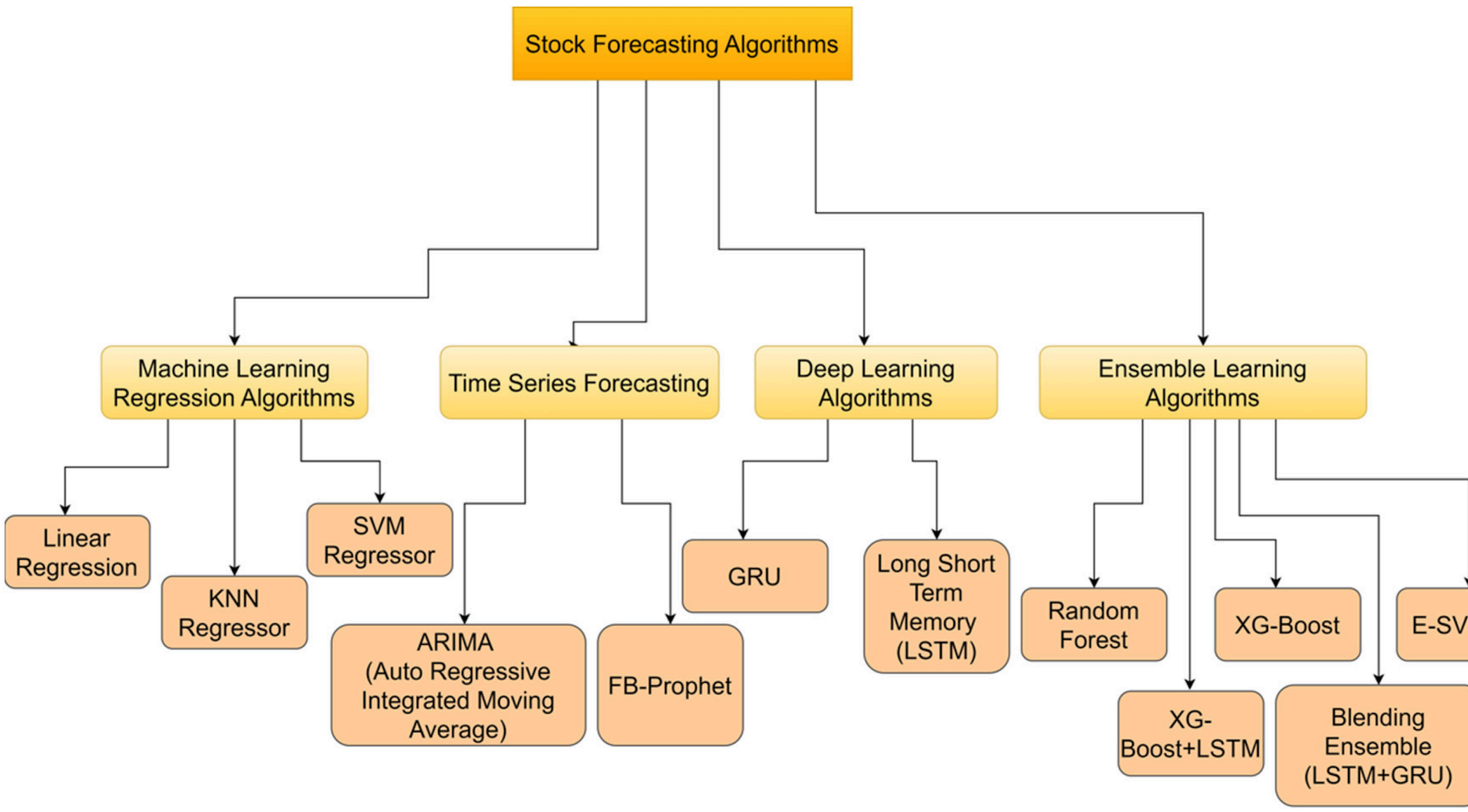
- Identify patterns and trends through data analysis

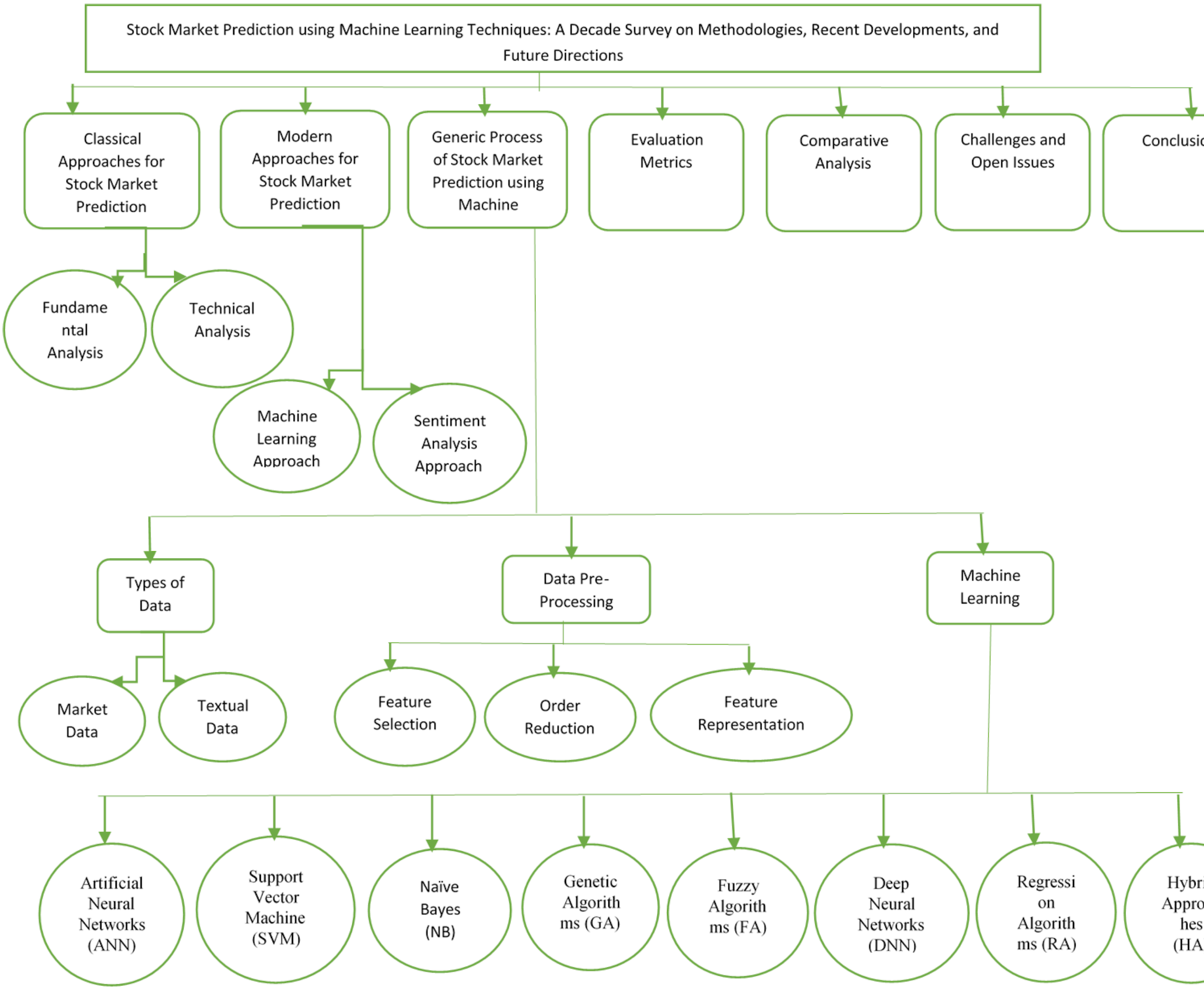
- Compare AI models (LSTM, ARIMA, Linear Regression) for accuracy.

- Deploy a user-friendly prediction app.

5. Flowchart of Project Workflow

Data Collection -> Preprocessing -> EDA -> Feature Engineering -> Modeling -> Evaluation -> Deployment





6. Dataset Description

- Source: Yahoo Finance via yfinance package

- Type: Public

- Structure: ~2,500 rows, 7 columns (Date, Open, High, Low, Close, Adj Close, Volume)

Example:

Date | Open | High | Low | Close | Adj Close | Volume

2020-01-01 | 296.24 | 299.96 | 295.19 | 297.43 | 297.43 | 33,870,100

7. Data Preprocessing

- Removed null values and duplicates

- Normalized price features

- Converted 'Date' to datetime and set it as index

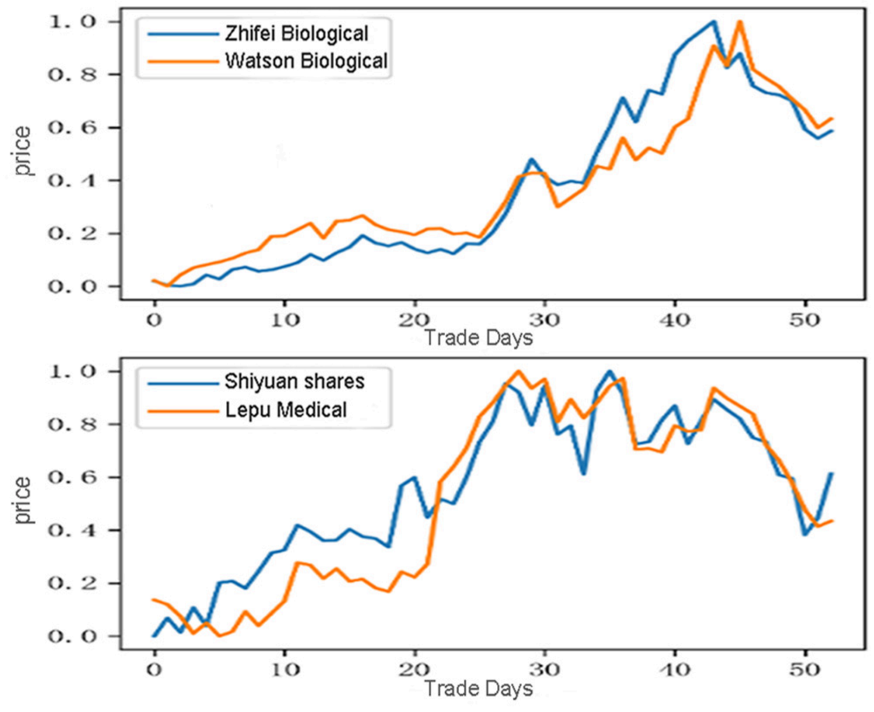
8. Exploratory Data Analysis (EDA)

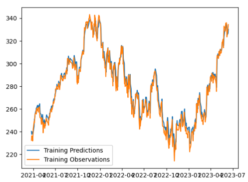
- Line plots showed increasing trends and volatility

- Heatmaps revealed strong correlation between 'Close' and 'Adjust’

- Volume spikes observed during major events

- Boxplots revealed outliers





9. Feature Engineering

- Created 7-day moving average and returns

- Added lag features (Close_t-1, Close_t-2)

- Performed feature selection using correlation matrix

10. Model Building

- Models used: Linear Regression (baseline), ARIMA, LSTM

- LSTM performed best with lowest RMSE

	date	open	high	low	close	volume	Name
449309	2014-10-30	86.16	87.000	85.670	86.940	5560308	PG
382759	2013-07-25	100.39	101.920	100.000	100.710	3285061	MON
29309	2014-07-03	61.95	62.200	61.830	62.010	773696	AKAM
303701	2016-12-09	53.54	54.115	53.390	53.830	2826688	IP
137693	2016-01-04	132.46	132.460	130.935	131.860	411964	COO
53947	2013-02-19	76.84	77.360	76.270	77.300	5854522	APA
349007	2014-07-10	29.33	29.835	29.255	29.775	916948	LNT

11. Model Evaluation

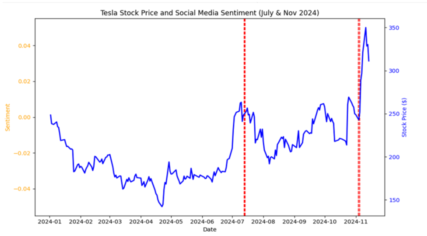
- Metrics:

 - Linear Regression RMSE: 5.12

 - ARIMA RMSE: 4.78

 - LSTM RMSE: 3.65

- Visuals: Actual vs Predicted plot, RMSE bar chart

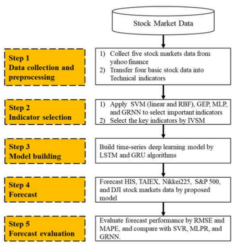


12. Deployment

- Platform: Streamlit Cloud

- Link: (placeholder)

- UI: Stock symbol input, output next price
-Example: AAPL -> $172.35



13. Source Code

```
import yfinance as yf

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense


# 1. Download stock data

def get_stock_data(ticker, start, end):

    data = yf.download(ticker, start=start, end=end)

    return data[['Close']], data


# 2. Preprocess the data

def preprocess_data(data, time_step=60):

    scaler = MinMaxScaler(feature_range=(0, 1))

    data_scaled = scaler.fit_transform(data)


    X, y = [], []

    for i in range(time_step, len(data_scaled)):

        X.append(data_scaled[i - time_step:i])

        y.append(data_scaled[i])
```

```
    X = np.array(X)

    y = np.array(y)



    return X, y, scaler



# 3. Build the LSTM model

def build_lstm_model(input_shape):

    model = Sequential()

    model.add(LSTM(units=50, return_sequences=True, input_shape=input_shape))

    model.add(LSTM(units=50))

    model.add(Dense(units=1))

    model.compile(optimizer='adam', loss='mean_squared_error')

    return model



# MAIN BLOCK

if __name__ == "__main__":

    ticker = 'AAPL'  # You can change this to any stock symbol

    start_date = '2015-01-01'

    end_date = '2023-12-31'



    data, raw_df = get_stock_data(ticker, start_date, end_date)

    X, y, scaler = preprocess_data(data)
```

```
# Reshape input to be 3D for LSTM [samples, time steps, features]

X = X.reshape((X.shape[0], X.shape[1], 1))



# Split into train and test

split = int(len(X) * 0.8)

X_train, X_test = X[:split], X[split:]

y_train, y_test = y[:split], y[split:]



# Build and train model

model = build_lstm_model((X.shape[1], 1))

model.fit(X_train, y_train, epochs=20, batch_size=32, verbose=1)



# Predict

predicted = model.predict(X_test)

predicted_prices = scaler.inverse_transform(predicted)

real_prices = scaler.inverse_transform(y_test.reshape(-1, 1))



# Plotting

plt.figure(figsize=(10, 6))

plt.plot(real_prices, color='blue', label='Actual Price')

plt.plot(predicted_prices, color='red', label='Predicted Price')

plt.title(f'{ticker} Stock Price Prediction')

plt.xlabel('Days')
```

```
plt.ylabel('Stock Price')

plt.legend()

plt.show()
```

14. Future Scope

- Integrate sentiment analysis
- Extend to long-term forecasts
- Use macroeconomic indicators

15. Team Members and Roles

M. Sandhiya - Responsible for data collection, data cleaning, and preprocessing tasks.

M. Neha - Handled exploratory data analysis (EDA) and feature engineering.

H. Ramya - Focused on model selection, training, and evaluation of prediction models.

R. Santhiya - Worked on deployment using Streamlit and final documentation.