Stock Price Prediction Project Documentation

# 1. Project Overview

This project focuses on predicting the next closing prices of stocks using historical financial data. It utilizes technical indicators like moving averages and RSI, and employs a machine learning model (SVR) for regression analysis. The project includes a prediction pipeline, an automation system, and visualization support.

# 2. Project Structure

- predict.py: Contains feature engineering, model loading, and prediction logic.  
- automation.py: Automates the fetching of recent data, prediction generation, visualization, and saving results.  
- models/: Directory to store trained SVR models and scalers.  
- plots/: Directory to save prediction plots.  
- predictions/: Directory to save CSV files containing future price predictions.

# 3. predict.py

This script includes all logic related to feature extraction and stock price prediction.

## Functions:

• compute\_RSI(series, window): Calculates the Relative Strength Index (RSI) for a series.

• add\_features(df): Adds moving averages, RSI, lag features, and time-based features.

• prepare\_features(df): Selects feature columns and converts them to NumPy arrays.

• load\_models\_scalars(ticker): Loads the trained SVR model and associated scalers.

• predict(ticker, recent\_data\_df): Runs the model prediction on new data.

# 4. automation.py

This script automates the end-to-end workflow using the logic defined in predict.py.

## Functions:

• save\_predictions(future\_dates, predicted, ticker): Saves predictions to a CSV file.

• plot\_predictions(...): Generates a plot comparing historical and predicted prices.

• run\_automation(ticker): Core automation logic to fetch data, predict prices, save and plot results.

# 5. Streamlit Integration (Optional)

A Streamlit app can be built on top of automation.py to provide an interactive UI for selecting tickers, triggering predictions, and visualizing outputs.

# 6. Usage Instructions

1. Ensure the SVR model and scalers for a given ticker are saved in the `models/` directory.

2. Run automation.py or integrate with Streamlit to generate predictions.

3. Visualizations and CSVs will be saved in their respective folders.

# 7. Conclusion

This modular architecture ensures a clean separation of responsibilities, making the system maintainable, scalable, and suitable for both command-line and web-based applications.