

**Time-Series Forecasting Using REST API**

**Hackathon '25**



NACHAMMAI S N 23PD21

RITUJAA B G 23PD32

**1. Data Retrieval (dataRetrieval.py)**

* **Objective**: Fetch historical stock data from an external API (e.g., Alpha Vantage).
* **Methodology**:
  + Use the requests library to send HTTP requests to the API.
  + Parse the JSON response and convert it into a Pandas DataFrame.
  + Save the retrieved data to a CSV file (stock\_data.csv) for further processing.
* **Key Features**:
  + Dynamic API URL and parameters for flexibility.
  + Error handling for API responses (e.g., invalid status codes or missing data).
  + Data is saved in a structured format with columns for open, high, low, close, and volume.

**2. Data Processing (dataProcessing.py)**

* **Objective**: Clean and preprocess the raw stock data for analysis and modelling.
* **Methodology**:
  + Load the raw data from stock\_data.csv.
  + Handle missing values using forward and backward filling (ffill and bfill).
  + Perform feature engineering to create additional features:
    - Daily returns (daily\_return).
    - Rolling average (rolling\_avg).
    - Day of the week (day\_of\_week).
    - Month (month).
  + Ensure no missing values remain in the dataset.
  + Save the cleaned and processed data to processed\_stock\_data.csv.
* **Key Features**:
  + Robust handling of missing values.
  + Creation of time-based and statistical features for improved modeling.

**3. Exploratory Data Analysis (eda.py)**

* **Objective**: Analyse the processed data to understand trends, seasonality, and stationarity.
* **Methodology**:
  + Load the cleaned data from processed\_stock\_data.csv.
  + Perform summary statistics and data type checks.
  + Visualize key metrics:
    - Closing price over time.
    - Daily returns over time.
    - Rolling mean and standard deviation.
  + Decompose the time series into trend, seasonality, and residual components.
  + Conduct autocorrelation analysis and stationarity tests (ADF test).
  + Generate a correlation heatmap and boxplot for daily returns.
* **Key Features**:
  + Comprehensive visualization of time series data.
  + Statistical tests to assess stationarity and autocorrelation.
  + Insights into data patterns and relationships.

**4. Model Training (modelTraining.py)**

* **Objective**: Train an ARIMA model for time series forecasting.
* **Methodology**:
  + Load the processed data from processed\_stock\_data.csv.
  + Split the data into training and testing sets (80-20 split).
  + Check for stationarity using the Augmented Dickey-Fuller (ADF) test.
  + Apply differencing if the data is non-stationary.
  + Fit an ARIMA model with exogenous variables (e.g., open, high, low, volume).
  + Evaluate the model using metrics such as MAE, RMSE, and MAPE.
  + Visualize the actual vs. forecasted values.
* **Key Features**:
  + Support for exogenous variables to improve model accuracy.
  + Robust error handling and debugging for model training.
  + Detailed evaluation metrics and visualizations.

**5. Forecasting (forecasting.py)**

* **Objective**: Generate future forecasts using the trained ARIMA model.
* **Methodology**:
  + Simulate stock data for demonstration purposes.
  + Train an ARIMA model on the simulated data.
  + Generate forecasts for the next 5 business days.
  + Plot the forecasted values along with confidence intervals.
  + Provide additional insights into forecasted values and confidence intervals.
* **Key Features**:
  + Visualization of forecasted values with confidence intervals.
  + Simulated data for quick testing and demonstration.

**6. Streamlit App (app.py)**

* **Objective**: Provide an interactive web interface for time series forecasting.
* **Methodology**:
  + Use Streamlit to create a user-friendly interface.
  + Allow users to:
    - Fetch data from an API.
    - Upload and preprocess custom datasets.
    - Perform EDA and visualize data.
    - Train ARIMA models with customizable parameters.
    - Generate and visualize forecasts.
  + Display evaluation metrics and visualizations dynamically.
* **Key Features**:
  + Interactive widgets for user inputs (e.g., API keys, ARIMA parameters).
  + Real-time feedback and error handling.
  + Comprehensive visualizations and insights.

**Summary of Workflow**

1. **Data Retrieval**: Fetch raw data from an API.
2. **Data Processing**: Clean and preprocess the data.
3. **EDA**: Analyse and visualize the data to understand patterns.
4. **Model Training**: Train an ARIMA model on the processed data.
5. **Forecasting**: Generate and visualize future forecasts.
6. **Streamlit App**: Provide an interactive interface for end-to-end workflow.

This modular approach ensures flexibility, scalability, and ease of use for time series forecasting tasks. Each script can be run independently or as part of the integrated Streamlit app.