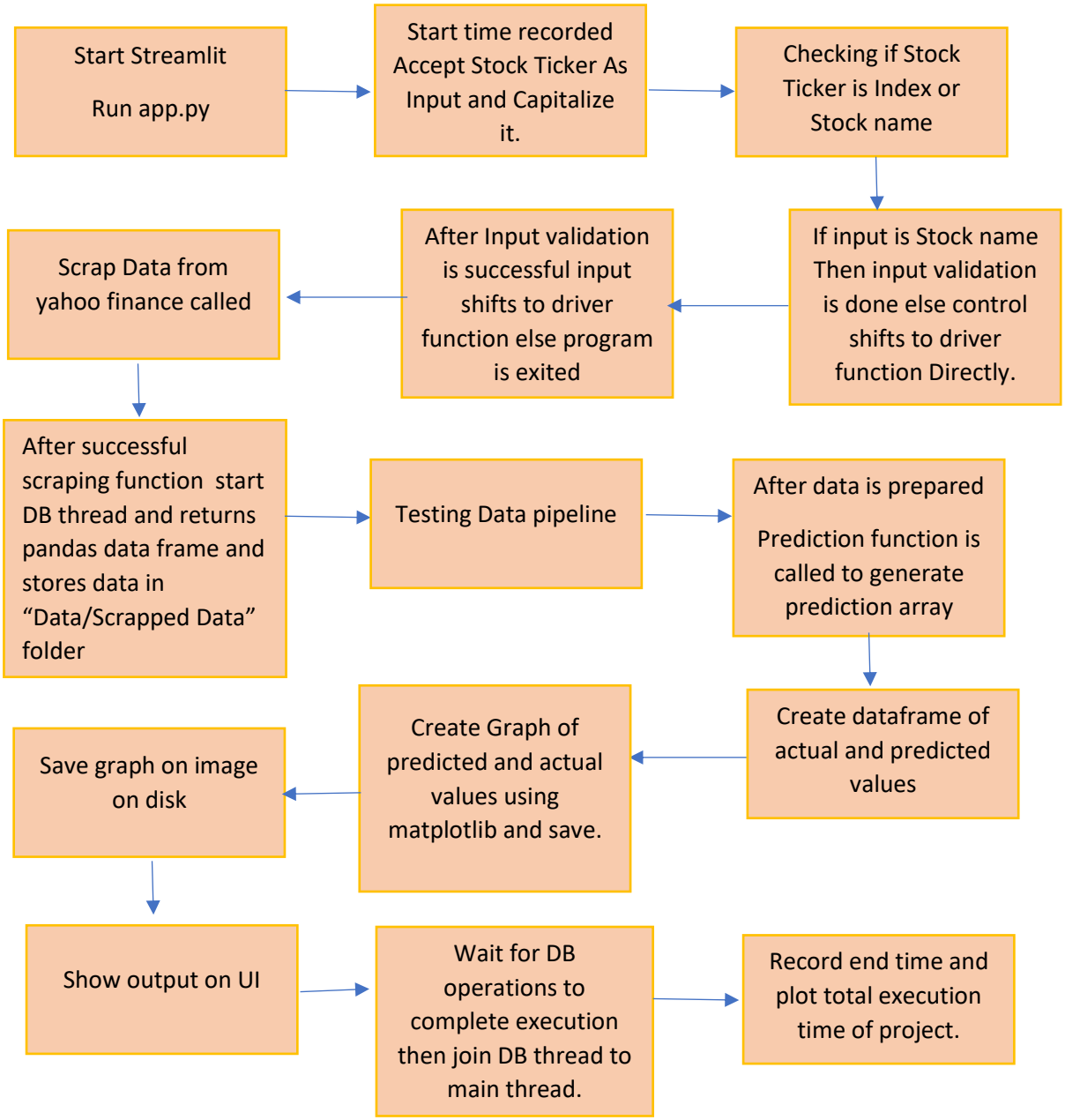
Income Prediction (LLD)

Rohit Chitte

Contents

Architecture



Architecture Description

UI

Streamlit Python library is used to accept user input and show output in simple and easy to understand design. Streamlit is used because it's easy to use and based on python syntax provide seamless UI design.

Input Mechanism

Basic Idea of project is based on assumption that user must input valid stock ticker symbols to know the trend. If user enters invalid symbol then running further function is useless as data will not be scaped, no dataframe will be generated and prediction will not be possible. Checking input at start will save us from unwanted and useless exection.User can enter data of any stock listed on NSE or two index names i.e. NIFTY and BANKNIFTY. To ensure stock symbols are correct, a dataset from NSE website is used. Basic mechanism is checking presence of input in EQUITY.CSV dataset. EQUITY.CSV dataset contains symbols of all registered companies on NSE. If entered input is not in EQUITY.CSV then exit the program.

Driver Function

Driver function is simple python function to implement different functionalities in the project. Functions implemented are "scrap_data_from_yahoo_finance" to scrap data, "testing" function to transform data into suitable input format for prediction. "prediction" function to predict future prices based on past prices.

Scrap_data_from_yahoo_finance (function)

Basic utility is to scrap data of any stock ticker symbol. Accepts one argument that is name of symbol to scrape data from. This function is defined in Data_Scrapping module in the project. Website named YAHOO FINANCE which is Authentic source of data is used. In this process Selenium Webdriver has been used to serve the purpose. Based on input different URLs are chosen. Symbols can be "NIFTY", "BANKNIFTY", or any stock name, based on those symbols URL is selected and browser window is opened. By default, Webdriver of Edge browser is used. Next step is checking for add banner on website for which we have defined mechanism. Then Navigate to calendar element click on that, next click on 5Year button to specify to get data of stock for 5 years, then click on apply button then scroll down to the bottom of page this scrolling is necessary because table is loaded on webpage dynamically if scrolling is not done then only 100 rows are scrapped by default.

Scrolling mechanism is developed after trial and error. Then data frame is scrapped from webpage completely, index is reset then data is stored in folder path Data/Scrapped Data file that is stored earlier there is moved to archive files. And data frame is returned at the end of function.

Testing Data (function)

This function is meant to transform the data scraped. "testing_data" function changes the form of data suitable to train time series model. Data transformation is as follows.

Original Data

Price
110
109
112
113
111
118
108

Transformed Data

X	Y
110, 109	112
109, 112	113
112, 113	111
113,111	108
111,108	108

In this example Y depends on past to prices, but in actual training data and testing data. Y is predicted based on past 100 prices. Next point is reading data using pandas from given path and using data and price column. Load 'scaler.pkl' file to load the scaler used during training phase.

Passing this data to scaler to normalize it followed by transforming data to format analogus to above example. Then final data for prediction is prepared and then fed to next function for prediction.

Prediction

Prediction function is defined in testing_pipeline.py module. Basic utility is to predict the future prices to know the trend using past data. This function accepts data output from testing_data function of same module. Model file in .h5 format is loaded for predictions. Followed by predictions and these prices are multiplied by the scale factor which is used to rescale the data to original scale and finally these predictions are returned in an array.

Plotting Final Graph

Based on Prediction and actual values. Final output graph is written on streamlit webapp in order to have clear comparison of actual values and predicted value to know the accuracy of model graphically. This graph is saved in .png format on the disk under folder section of archive files/ Prediction Figures.

Miscellaneous

DB operations are also implemented. Specialized mechanism is developed to upload table data on cloud Cassandrea DB. After data scraping is finished and .csv file is saved in folder system. Immediately thread for DB operation is called to upload this data in parallel as this is time consuming task because it involves connecting to remote server and uploading data to this remote servers. Before uploading data separate table is created to store data of specific stock symbol. Hence checking is done whether table for that stock already exist or not then data is uploaded to cloud DB.

Project is coded in modular fashion following all the coding standards. Try and except blocks are used as necessary. And logging is implemented in logging.py file under Logger folder. Logs are stored in log.txt file.

Archieve_Files (folder)

Stores scraped data in .csv format.

Archieve_Files/Prediction Figures(Folder)

Stores or graph for stocks in .png format to know the comparison between predicted and actual prices.

Data(Folder)

Stores .csv Files which are scraped from yahoo finance website.

Data/Static Data(Folder)

Stores data of EQUITY.CSV which contains names and symbols of companies registered on NSE.

DB_Files(Folder)

Contains module to implement DB operations and file named keyspace_table_names.txt file which stores the name of tables created on remote Database to save data of specific stocks. This file is useful when one wants to fetch data from cloud Database.to fetch data from cloud use command:

“SELECT * FROM schema.stock_name;” , refer keyspace_table_names.txt file to fetch data.

Geckodriver.log & msedgedrive.exe (file)

These files are useful to run web driver function for scraping

Main.py

Is same as app.py file but without streamlit.

New_model.h5

Model files used for prediction.

Requirements.txt

Contains all modules and respective versions to run project.

Secure-connect-test1.zip

Contains data about client id and other authentication details necessary to connect DB.