Project Overview:

The application is designed to cater to the needs of stock traders, offering three key features. Firstly, users can access real-time stock data, including the opening price, closing price, highest and lowest prices of the day, and the stock's daily increase. Users input the stock symbol (e.g., AAPL) and the desired date to retrieve the relevant information. The system also provides an auto-complete feature in the stock selection dropdown for user convenience. It performs error handling for non-trading days or future dates.

Secondly, the "prediction" feature allows users to input a stock name on the homepage. The program fetches historical opening and closing prices for the past year, computes the stock's three-day rise and fall, and calculates the Bollinger Band difference. Subsequently, it generates a scatter plot based on these values and utilizes a multiple linear regression model to predict the stock's future movement, displaying two scatter plots and a prediction result on the page. Navigation through the entire program is facilitated via a sidebar, and the display is structured using the render function in page_app, with the main object instantiated on the home page in app.py.

The program leverages the Polygon API (https://api.polygon.io/v2/) to retrieve stock parameters. The API allows specification of stock name, start date, end date, time unit, and other relevant details, returning key indicators such as opening price, closing price, trading volume, highest and lowest prices, and weighted average price for a specified time period.

Overall, the application provides a comprehensive platform for stock traders to access real-time data, historical trends, and predictive analytics, enhancing their decision-making process.

REST API：polygon.io

The program relies on the versatile https://api.polygon.io/v2/ API, tailored for stocks and futures. It serves two primary purposes in our application.

Firstly, users can query specific stock data by inputting parameters like stock name, start time, end time, and time unit. This allows them to access crucial information such as opening and closing prices, trading volume, highest and lowest prices, and weighted average price.

Secondly, by entering "us market," users can retrieve a list of all stock names currently in the US stock market. This feature streamlines the process, providing a quick overview of available stocks, enhancing usability, and offering a snapshot of the current US stock market landscape.

These features leverage the Polygon API, empowering users with efficient tools to access targeted stock data and explore the broader US market.

Base URL: https://api.polygon.io/v2/

Endpoint for check stock information:
aggs/ticker/{stock_name}/range/1/{time_unit}/{from_time}/{to_time}?adjusted=true&sort=asc&limit=280&apiKey={api_key}
Used in the function get_info(stock_name, from_time, to_time, time_unit="day") in function_help file. Also used in class predict to get one_year_data and used in class stock to get the data for one day.

Endpoint for check all the stock name in market:
snapshot/locale/us/markets/stocks/tickers?apiKey={api_key}
Used in function get_stock_list() in function_help file. Show the match list in app.py for stock name user input.

List of Feature:

#Feature 1: **Search**

On the "Stock Check" page, users input the stock name and query date, creating a Stock class object in the Final_Stock_class. The stock name and query date are used to construct a URL through the get_info function in function_help, fetching corresponding data stored as a dictionary. This data is assigned to the object's 'data' attribute. During initialization, specific parameters filter and convert the data into float values, which are then stored within the class for further use.

Data Class: Stock
Function: get_info(stock_name, from_time, to_time, time_unit="day")
Api end point: aggs/ticker/{stock_name}/range/1/{time_unit}/{from_time}/{to_time}?
Pages: Stock Check
Page View:
First a box ask user input name of stock, enter, then a select will show all stock match the name input. Select one, then select the date you want to check. Final, press the button, the information will show.

#Feature 2: **Data Visualization**

Users begin by entering a stock name, creating a Predict class object in Final_predict_class. The get_one_year_data(self) method retrieves past year stock data, stored in the class. Subsequently, get_stock_shake(self) calculates the three-day total increase store in class with list type, get_bbands(self) computes Bollinger Bands values, and get_bbands_rate(self) determines the Bands' oscillation amplitude, all stored as lists in the class.

The get_scatter_plot(self) method classifies Bollinger Band fluctuation and stock's three-day increase as profitable or loss (green or red) for the next day. Streamlit is then used to generate scatter plots on the homepage, illustrating the data relationships.

Data Class: Predict
Function:
get_scatter_plot(self)
get_info(stock_name, from_time, to_time, time_unit="day")
get_one_year_data(stock_name)
get_stock_shake(self)
get_bbands(self)
get_bbands_rate(self)

Api end point: aggs/ticker/{stock_name}/range/1/{time_unit}/{from_time}/{to_time}?
Pages: Stock Predict

Page View:
User will input stock name, then select box show, choose the match stock and press button. The two scatter plot will show.

#Feature 3: **Statistics**
After we use The get_one_year_data(self), get_stock_shake(self), get_bbands(self) and get_bbands_rate(self), we continue to use get_increase(self) to get the daily increase list. Next, we convert the three-day increase list and the Bollinger Band difference list into arry as X1, X2, and the daily increase list as Y and enter the multivariate linear return model for analysis. And use the data of the most recent day to predict the next day. If the result is greater than 0, return string "The stock will rise." If the result is less than 0, return string "The stock will fall"Data Class: Predict. Finally, I use streamlit show it in page, Stock Predict.

Function:
get_info(stock_name, from_time, to_time, time_unit="day")
get_one_year_data(stock_name)
get_stock_shake(self)
get_bbands(self)

get_bbands_rate(self)
get_increase(self)
muti_regression(self)

Api end point: aggs/ticker/{stock_name}/range/1/{time_unit}/{from_time}/{to_time}?
Pages: Stock Predict
Page View:
A string will show the result after user input stock name.

Other supplements:
Function: Display stock prices, display scatter plots, and predict stocks.

Data Type:
Data imported from API: dic
Stock opening price: list (by data filtering)
Stock closing price: list (by data filtering)
Bollinger Bands: list (calculated by)
Bollinger band shock amplitude: list (calculated by)
Three-day average stock price increase: list (calculated by)
Daily rise and fall of stocks: list (calculated by)
X-axis of prediction model (three-day average stock price increase, Bollinger Band fluctuation range): array
Forecast model Y-axis (daily rise and fall of stocks): array

Data Frame:

final_project
├── app.py
├── models
│ ├── __init__.py
│ ├── Final_predict_class.py
│ ├── Final_stock_class.py
│ └── function_help.py
├── pages_app

```
│   ├── __init__.py
│   ├── Stock_Check.py
│   └── Stock_Predict.py
└── test
    ├── __init__.py
    ├── test_Function_uni.py
    ├── test_Predict_class_uni.py
    └── test_Stock_class_uni.py
```