Project 2 Documentation

Class: CSE472

Team: Group 2

Team Members: Isaac Walker, Myles Hobbs

Set-Up Instructions:

1. Ensure that Python 3.7 is installed on your local machine. You can do this by:
    a. Windows/Mac
        i. Visit Python Website & download the latest version
        ii. Run the installer and check the box "Add Python to PATH" during installation, then follow the prompts to install
    b. Linux (In Command Prompt)
        i. python3 --version
        ii. sudo apt update
        iii. sudo apt install python3
2. Download & Unzip Project Zip
    a. Should include
        i. stockDataGathering.py
        ii. combined.json
        iii. requirements.txt
3. Install necessary libraries
    a. pip install -r requirements.txt
4. Configure Directory & Change any existing pathways in stockDataGathering.py necessary
5. Sign up and obtain an API Key from AlphaVantage
    a. Replace the api_key in the file with the new one you have obtained
6. Verify the Environment with the line
    a. python -c "import requests, bs4, transformers, torch, numpy, matplotlib, sklearn, bs4, collections, json, datetime, pandas"
7. Ensure file addresses are correct
    a. File addresses such as for combined.json in format_tweets() need to be adjusted.
8. Run Python file with the command: python3 stockDataGathering.py

Usage Details:

- Dependencies
    - Follow the Set Up Instructions to obtain all dependencies and files necessary for your environment
- Files Outputted
    - buy_hold_sell.json
        - This is a JSON file that holds a log of the trade decision made by the Step 3 decision model and the other parameters that affected the decision.
        - It is an array of dictionaries with multiple keys:
            - Date: The date of the day in question.
            - Sentiment Score: The current day's daily sentiment score.
            - Action Taken: The Trading action such as "BUY", "SELL" or "HOLD".
            - Buy Threshold: The threshold at which it is okay to buy stock.
            - Sell Threshold: The threshold at which it is okay to sell stock.
            - Slope: The trend in which daily sentiment has historically shown.
    - portfolio.json
        - This JSON file logs the backtracking system's entries as it uses sentiment scores to make stock trade decisions
        - It is an array of dictionaries with multiple keys:
            - Date: The specific date for which the portfolio data is recorded.
            - Current_Portfolio_Value: The total value of the portfolio on that date, including cash and the value of held stocks.
            - Current_Stock_Price: The average stock price (calculated as the mean of open and close prices) for that date.
            - Stock_Held: The number of stocks held in the portfolio on that date.
            - Cash_Held: The amount of cash available in the portfolio on that date.

        - Slope: The trend of the daily sentiment score from the beginning till the current date.
        - Buy_Threshold: The sentiment score threshold above which the model decides to buy stocks.
        - Sell_Threshold: The sentiment score threshold below which the model decides to sell stocks.
        - Sentiment_Score: The current day'ssentiment score for the date.
        - Action_Taken: The action taken by the model on that date, which can be "BUY", "SELL", or "HOLD".
        - Buy_Count: The cumulative number of buy actions performed up to that date.
        - Sell_Count: The cumulative number of sell actions performed up to that date.
        - Profit_Today: The net profit or loss realized on that particular day.
        - Net_Profits: The total net profit or loss since the start of the strategy up to that date.
    - stock_price_portfolio_value.jpg
        - This shows the overlapping line graphs of stock price over time and portfolio value over time. The left y-axis and right y-axis are for different lines.
    - sentiment_score.jpg
        - This shows a scatter plot of the daily sentiment scores over time. The scores range from -100 to 100.
    - trade_actions_stock_price.jpg
        - This shows the stock price over time but is split into 3 separate line graphs for if the trade action was BUY, SELL, or HOLD during that stock price.
- Limitations of Files
    - While daily_open_close's API call can get you accurate up-to-date information up to the current day, combined.json is static and was a dataset created by APIFY. There are no API calls due to Twitter's limitations and persistence to not let users web scrape their platform. Given this, the predictions will not account for

real-time market factors & purely generate sentiment scores based on data from November 2024.

    - Sentiment analysis models are still very young. The most popular one, Vader, is not particularly good at understanding the context of text before populating it with a sentiment score. Here I used the HuggingFace bertweet-base-sentiment-analysis model instead so it could better understand the context of a given tweet around the tickers $META & $FB. However, it is hard to rely on an algorithm to understand things like sarcasm & ambiguity so some scores may be misinterpreted
    - This is also not to be used for a real model predictor, this is purely for academia & should not be used by any financial institution or be used for financial advice
- Ethical Considerations
    - Do not use the file for financial fraud or market manipulation.
    - Respect API rate limits and terms of service.

<u>Explanations of Components:</u>

- Step 1
    - get_daily_open_close_prices()
        - This function fetches the daily open and close price of the $FB/$META stock. It uses an API call to Alpha Vantage to get the complete dataset of the daily time-series data. From the data from the API's response, it creates daily_open_close which is a list where the key is the day's date and the values are the open and close stock prices.
    - format_tweets()
        - This function loads the JSON file combined.json and then formats the tweets within the JSON file to return tweet_list. step pertains to data collection. Combined.json comes from the use of APIFY, a third party, to gather Twitter posts from the past decade about the stock tickers $FB and $META, and compile all the results into a JSON file. This is not done within the Python file as it was from a third-party site. It was extremely

difficult to find sites that both allowed access to web scraping and also had content in a format that contained “posts” from a decade's worth of data. The function format_tweets() is simply formatting the dates of the tweets to match daily_open_close and grabbing their tweet content. The returned tweet_list is a list of elements in [formatted_date, full_text] format.

- Step 2
    - analyze_tweet_sentiment(tweet_list)
        - This uses HuggingFace finiteautomata/bertweet-base-sentiment-analysis model. This is because tweets can be highly complex and BERTweet is a derivative of BERT that specializes in tweets that can give sentiment analysis of the stock tickers in the context of the tweet. If the tweet is overall negative but says something positive about $FB/$META, it wants that to receive a positive score. Then it tokenizes the text to put it into the model. The output then produces logits which represent the model's confidence about each sentiment class, however, they are not probabilities. So it then converts them into probabilities using softmax & finally calculates the overall sentiment score very similarly to how they do in VADER. It returns analysis which is a list where each entry has the tweet’s date, the stock ticker, and the sentiment score of the tweet.
    - calculate_daily_sentiment_scores(analysis)
        - This function returns a list of daily sentiment scores created from the analysis of the tweet list. It creates sentiment_dict to contain the total amount of positive and negative sentiment scores from the sentiment analysis dataset we created. Scores > 0 were deemed positive and those < 0 were deemed negative. Afterward, it obtains a total number of positive posts and negative posts for each date and calculates a final score within the range of [-100,100] using the equation ((N_pos - N_neg) / (N_pos + N_neg)) * 100. The daily sentiment scores and their dates are appended to a list called daily_sentiment_scores.
- Step 3
    - decision_model(scores, sell_threshold, buy_threshold):

        - The function uses a linear regression model to find the trend in daily sentiment and then decides to BUY, SELL, or HOLD the stock based on whether the trend is positive or negative and if the daily sentiment score for the day is at the correct threshold. The linear regression model is trained off of an enumeration of the days up till the current day and their sentiment scores. It finds the trend in sentiment by finding the model's slope (coefficient). This slope is constantly updated as time passes. Finally, the function makes its decision based on whether to BUY if the trend is positive and the daily sentiment is equal to or greater than the buy threshold, to SELL if the trend is negative and the daily sentiment is equal to or less than the sell threshold, or HOLD if not selling buying stock. It creates and returns buy_hold_sell which is a list of dictionaries where the keys are the date, thresholds, sentiment score, action taken, and the slope. It also prints the last entry of buy_hold_sell before it returns.
    - print_latest_buy_hold_sell(buy_hold_sell):
        - The function prints the last entry of buy_hold_sell. It finds the last entry and prints out a formatted list of the date, sentiment score, action taken, buy threshold, sell threshold, and slope of the particular day that the decision model was working with.
- Step 4
    - calculate_return_on_investment(portfolio, starting_capital)
        - The function calculates the return on investment (ROI) of the backtracking system's portfolio. It calculates the ROI with the equation ROI = Net Profits / starting capital. This returns the number of times the initial investment into the portfolio is returned at the end.
    - calculate_win_loss_ratio(portfolio)
        - The function calculates the win/loss ratio of the backtracking system's portfolio. It calculates the win/loss ratio by counting how many days the profit was positive as wins and counting how many days the profit was negative as losses. Then it uses the equation win-loss ratio = wins/losses.

        It checks for 0 to not divide by 0. This returns the ratio of the portfolio money made versus money lost.

    - calculate_sharpe_ratio(portfolio)
        - The function calculates the Sharpe ratio of the backtracking system's portfolio. It calculates the Sharpe ratio by first finding the daily profit of the portfolio as daily returns. Then it calculates the mean and standard deviation of the daily returns. Finally, it uses the equation Sharpe ratio = average daily returns / standard deviation of daily returns. It checks for 0 so it doesn't divide by 0. This returns the risk assessment and performance of the investment.
    - calculate_max_drawdown(portfolio)
        - The function calculates the max drawdown of the backtracking system's portfolio. It finds the largest peak-to-trough decline which shows the worst-case scenario. It does this by going through every day of the portfolio and using the equation drawdown = (peak - portfolio value) / peak while adjusting the max drawdown as appropriate.
    - backtracking_system(starting_capital, buy_threshold, sell_threshold, daily_open_close, daily_sentiment_scores)
        - The function is the center point of the backtracking system. It acts the same as the decision model system, but it simulates actually using the decision to buy a stock with cash that was gained from starting capital. It counts how many times a stock is bought and sold and also constantly updates the portfolio's value by calculating the sum of the cash on hand plus the number of stocks times the current stock price. It uses the same linear regression model as the decision model system by training on enumerated dates and daily sentiment scores to find a trend in the daily sentiments (slope). The only additional parameter it has to choose BUY is that there must be enough cash to buy the stock and to choose SELL is that there must be a stock being held to sell. The function ends by printing the last portfolio entry and returns portfolio which is a list of dictionaries.

The description of the portfolio can be found in the Files Outputted section where it describes the the dictionaries' keys and values.

  - print_latest_portfolio(portfolio)
    - The function is used to output the last portfolio entry. It first grabs the last entry and then it outputs a formatted list of the entry. The description of the portfolio can be found in the Files Outputted section where it describes the the dictionaries' keys and values.
- Step 5
  - visualize_data(portfolio)
    - This function creates visualizations of the last backtracking system's attempt's portfolio for its plotting. It first creates a DataFram from portfolio and then it will create three plots that are output and then saved as jpegs. The first is the stock price and portfolio value over time. This plot needs to use twinning to overlay the stock price and portfolio value with different y-axes as they have vastly different ranges. The second is the daily sentiment scores over time as a scatter plot. This plot is a scatter plot as a line graph of the scores over time was unreadable. The third is the stock price split into three with each line corresponding to whether the trace action was to BUY, SELL, or HOLD that day.
  - get_threshold(prompt)
    - The function is called when user input is needed for the buy/sell thresholds of the decision-making or backtracking systems. It won't return until an acceptable value is input and will raise an error otherwise. It then returns the input value.
  - get_capital(prompt)
    - The function is called when user input is needed for the starting capital of the backtracking system. It won't return until an acceptable value is input and will raise an error otherwise. It then returns the input value.
  - main_prep()
    - This function declares daily_open_close, tweet_list, analysis, and daily_sentiment_scores as global variables so they can be called in the

main function. It then calls the functions that return the variables while printing what step the program is on. This is separated from main as it contains the analysis of the tweets' sentiment and also the API call of daily_open_close. The analysis can take over 10 minutes to finish and there is a rate limit on the API so it is suggested that this function is commented out at the beginning of main if the variables are already stored such as when running this function and main on a modular python file like an (.ipynb).

- main()
    - This is the main function. It calls main_prep so that daily_open_close, tweet_list, analysis, and daily_sentiment_scores are accounted for, although that can be commented out if the Python environment is set up that the main function can be called multiple times without calling the rest of the already run code. The main function has the user input the thresholds for buying and selling stocks and will print the last decision made. It will then prompt the user if they want to run it again. This comes with error handling. It will then export the last decision model attempt as a JSON file (see buy_sell_hold.json in Files Outputted). Next, the main function has the user input the thresholds for buying and selling stocks as well as the starting capital and will print the last portfolio day as well as its performance metrics. It will then prompt the user if they want to run it again. This comes with error handling. It will then export the last backtracking system attempt as a JSON file (see portfolio.json in Files Outputted). Finally, the function will call visualize_data to plot and show three graphs of the backtracking system's last attempt and end itself.

Program Usage Instructions:

- Preface:
    - The main program is divided into two parts: mainPrep() and main(). If you are running the program in such a way that you already have Steps 1 and 2 done and want to run the program without having to run analysis again, you can use the user input prompt to not call mainPrep() at the beginning of main() and skip 10 minutes of sentiment analysis. This is most relevant if you are running the files within an environment like Jupyter or Google Colab.
    - Main is set up to call mainPrep to get the daily sentiment scores, and then it will call the decision-making system, then the backtracking system, and finally, it will generate visualizations.
- Main Function Parts:
    - The decision-making system is set up to take user input. It wants the selling threshold and the buying threshold as user input. These must be numbers within -100 to 100 and it will detect false inputs. It will then print the last decision made. It can be run as many times as needed with new thresholds as it will prompt for user input to try again. The last decision-making system attempt is exported as a JSON.
    - The backtracking system is set up to take user input. It wants the selling threshold, buying threshold, and the starting capital as user input. These thresholds must be numbers within -100 to 100 and the starting capital must be a number between $0 and $1,000,000,000. It will detect false inputs. It will then print the last day's entry of the portfolio. It can be run as many times as needed with new thresholds or starting capitals as it will prompt for user input to try again. It will also print the return on investment, win/loss ratio, max drawdown, and Sharpe ratio. The last backtracking system attempt is exported as a JSON.
    - The visualizations are created as soon as the backtracking system's last attempt is over. It will use the last backtracking system's attempt's portfolio for its plotting. It will create 3 jpegs. The first is the stock price and portfolio value over time. The second is the daily sentiment scores over time as a scatter plot. The third is

the stock price split into three with each line corresponding to whether the trace action was to BUY, SELL, or HOLD that day.

- As soon as the visualizations are output, the program has ended.