Stock Analysis application

By SMU fintech bootcamp team ‘22

Augmenting code with CII infrastructure Beyond simple services of code.

Summary:

. Stock analysis Research

. Simple software solution for non-traders, and non-coders

. The Software Architecture of the solution

. Development process and roadmap

Stock analysis Research

- In general finance research involves statistical analysis of individual financial instruments.
- For this particular occasion: thorough statistical analysis of the combined decrease, and increase of stock securities between the dates September 12th - September 20th
- Over 10 securities such as the top traded stocks between this date like AAPL, AMZN, GME, GOOGL, JPM, LYFT, NVDA
- In practice there is one CSV file containing each day of data for those securities.

The Stock Analysis research

To analyse one security, two steps are necessary

- Step 1. Produce proper data for the selected dates.

  1A. Output production of data with latest trade volume, price, sell etc. all data on events that occurred on the specified day. We use CSV , and import path from pathlib and import a get_data function.

  2A. Process to visualization so we can obtain output in form of tables and graphs. For this we import pandas as Pd, from qualifier.utils.SQL import add_new_table, import matplotlib.pyplot as plt

The Stock Analysis research and simple software

———

To analyze one security two steps are required

- Step 2 the security is ideally suited to a command line interface. Since it is embarrassingly a data explosion of numbers and words in that csv file.

  2A. The analyzer we are modeling has a little over simple to interact with CLI, a solution in 3 distinct parts, this is context within which the software solution is looked for. This illustrated on a menu as options A, B ,and C

  2B. These three solutions use code that encapsulates the finance/statistics knowledge, and the main menu interfaceable function accepts analysis requests and exports too local environment.

Simple Software

Yet, overall goal of Application is: reliable automatic processing of securities data, too provide visualization and tables just as well as prediction for that data in simple interface usable by anyone who can read.

Software architecture

Given, the problem in developing constantly trading financial securities brings, and the software for such objective is a little more then just storage + computation and information systems. The stock prediction analyzer solves and will fit well at least part of the problems. In which we use an architecture of functions that in conception that would fit well to solve many classes of problems, usable beyond finance.

Software architecture that does so is more likely to be usable: it can cooperate in more elaborate solutions. Our manin menu utilizes functions with iterative development processes in place for quick initial solutions for stock analysis.

Development process of CLI

144 lines (100 sloc) | 3.58 KB

```python
import csv
import symbol
from symtable import Symbol
from webbrowser import get
import sys
from pathlib import Path
import pandas as pd
from Modules.CleanData import get_data



def load_stocks():
    """Writes stock information from CSV to list."""

    stocks = get_data('09-14-2022')
    with open(csvpath, newline='') as csvfile:
        rows = csv.reader(csvfile)
        header = next(rows)
        for row in rows:
            symbol_i =  (row[0])
            price = float(row[1])
            stocks = {
                "Symbol": symbol_i,
                "Price": price,
                "Exp_date": exp_date
            }
            stocks.append(stocks)
        return stocks


def validate_symbol(symbol_i):
    """Verifies that symbol exist."""

    # Verifies symbol prints validations message and return True. Else returns False.
    if len(symbol_i) == stocks["Symbol"]:
        print(f"Your stock is valid")
        return True
    else:
```

```python
stock_a = (stocks.get("Price"))


def choose_date():

        # Use input to determine the date
    # Re-type date from a string to a floating point number.

    date = input("For what date?\n")
    date = float(Exp_date)

    # Validates date. If less than or equal to 0 system exits with error message.
    # Else system exits with error messages indicating that the date is not in system.
    if date <= stocks["Exp_date"]:

        print(f"The value of the stock for this date is {stocks['price']")
        return stocks
    else:
        sys.exit(
            "You do not have valid entry."
        )

#should get visualization and display
def choose_visualize():

    # Use input to determine the date
    # Re-type date from a string to a floating point number.
    date = input("For what date?\n")
    date = float(Exp_date)

    # Validates date. If less than or equal to 0 system exits with error message.
    # Else system exits with error messages indicating that the date is not in system.
    if date <= stocks["Exp_date"]:
```

Development process of CLI pt 2

```
67
68      # Use input to determine the date
69      # Re-type date from a string to a floating point number.
70      date = input("For what date?\n")
71      date = float(Exp_date)
72
73      # Validates date. If less than or equal to 0 system exits with error message.
74      # Else system exits with error messages indicating that the date is not in system.
75      if date <= stocks["Exp_date"]:
76
77
78          print(f"The value of the stock for this date is{stock_a}")
79          return stocks
80      else:
81          sys.exit(
82              "You do not have valid entry."
83          )
84
85
86  """Save the results.
87
88  Output the lists of stocks to a csv file
89     Use the csvwriter to write the `stock.values()` to columns in the CSV file.
90
91  """
92
93
94
95  header = ["Symbol","Price","Type","Strike","Exp Date","DTE","Bid","Midpoint","Ask","Last","Volume","Open Int","OI Chg","IV
96  output_path = Path("all_stocks.csv")
97
98  def Save_data(stocks)
99        output_path = Path("all_stocks.csv")
100       with open(output_path, 'w', ) as csvfile:
101       csvwriter = csv.writer(csvfile, delimiter=",")
102       csvwriter.writerow(header)
103     for stock in stocks:
104         csvwriter.writerow(stock.values())
105
106
```

```
106
107
108
109 def main_menu():
110     """Dialog for the analyzer Main Menu."""
111
112     # Determines action taken by application.
113     action = input("Would you like to check a stock price (b), Do you want to visualize data (c) or save data (d)? Enter b
114     return action
115
116
117 def run():
118
119
120     Stocks = validate_symbol()
121
122     # Initiates action: check stock, visualize or get data.
123     action = main_menu()
124
125     # Processes the chosen action
126     if action == "b":
127         choose_date()
128
129     elif action == "c":
130         choose_visualize()
131
132     elif action == "d":
133         Save_data(stocks)
134
135     # Prints the adjusted balance.
136     print(
137         "Thank you for using stock analayzer."
138     )
139
140     anyelse = input("Anything else to do? (y/n)")
141     if anyelse == "y":
142         run()
143     else:
144         print("Thank you for using stock analyzer. Have a nice day!")
```

Development process and road map of project