Fabian Gonzalez

Professor Ryan

CS-110

13 December 2020

Python Program for DJIA Analysis: Summary

This program centers around the Dow Jones Industrial Average and its volatility since the end of the Cold War. The USSR dissolved in December 1991. The loss of the Cold War-related geopolitical threats took away an element of uncertainty for market investors. However, greater globalization, more dependence on leverage, and increased financialization that resulted from the fall of the USSR introduced more factors that could make the markets more volatile. There have been many instances of significant market volatility since 1991, including the burst of the Tech Bubble, the Great Recession, and the COVID-19 reset. Annual business cycles, federal government policy, and the performance of the DJIA are all related to each other. My project seeks to measure which administrations faced the most volatility during each month.

The first function this program offers is a reader display of the CSV that the DJIA data is pulled from. The program opens the CSV from the directory location and prints the data row by row. This is a simple concept. Below is an example of one of the reader displays, and an excerpt of the output.

def Bush43\_CSV():  
 with open("C:/Users/12019/Desktop/Fabian Python/Python Final/Data/George W Bush.csv", "r") as Bush43\_Data:  
 reader = csv.reader(Bush43\_Data)  
 for row in reader:  
 print(row)  
 Bush43\_Data.close()

['ï»¿"Date"', 'Price', 'Open', 'High', 'Low', 'Vol.', 'Change %']

['Jan 01', '10,887.36', '10,790.92', '11,028.00', '10,468.04', '290.79M', '0.92%']

['Dec 00', '10,787.99', '10,416.76', '10,917.68', '10,299.21', '270.67M', '3.59%']

['Nov 00', '10,414.49', '10,966.21', '11,006.50', '10,292.39', '240.12M', '-5.07%']

['Oct 00', '10,971.14', '10,659.06', '10,995.41', '9,654.64', '297.12M', '3.01%']

['Sep 00', '10,650.92', '11,219.54', '11,401.19', '10,567.32', '243.81M', '-5.03%']

['Aug 00', '11,215.10', '10,523.81', '11,319.06', '10,516.76', '165.39M', '6.59%']

['Jul 00', '10,521.98', '10,450.36', '10,874.63', '10,393.09', '171.24M', '0.71%']

The second function this program offers is to determine the average volatility of the DJIA during a presidential administration. There are 5 post-Cold War administrations that my project has data for: George H.W. Bush, Bill Clinton, George W. Bush, Barack Obama, and Donald Trump. Functions were created for every monthly average for each administration. Here is an example:

def Dec\_HW\_Bush():  
 Dec\_HW\_Bush\_Data = pd.read\_csv("C:/Users/12019/Desktop/Fabian Python/Python Final/Data/George HW Bush.csv", header=None)  
 Dec\_HW\_Bush\_Data = pd.DataFrame(Dec\_HW\_Bush\_Data.values.reshape(26, 7))  
 Dec\_HW\_Bush\_column\_names = Dec\_HW\_Bush\_Data[0:1].values[0]  
 Dec\_HW\_Bush\_Data2 = Dec\_HW\_Bush\_Data[1:]  
 Dec\_HW\_Bush\_Data2.columns = Dec\_HW\_Bush\_Data[0:1].values[0]  
 Dec\_HW\_Bush\_Data2.head()  
 Dec\_HW\_Bush\_Data3 = Dec\_HW\_Bush\_Data2.iloc[[11, 23], [0, 6]]  
 Dec\_HW\_Bush\_Data3['Change %'] = list(map(lambda x: x[:-1], Dec\_HW\_Bush\_Data3['Change %'].values))  
 Dec\_HW\_Bush\_Data3['Change %'] = [float(x) for x in Dec\_HW\_Bush\_Data3['Change %'].values]  
 Dec\_HW\_Bush\_Data3['Change %'] = Dec\_HW\_Bush\_Data3['Change %'].abs()  
 Dec\_HW\_Bush\_Mean = (Dec\_HW\_Bush\_Data3['Change %'].mean())  
 print("Average Movement Under HW Bush in:", Dec\_HW\_Bush\_Mean)

A challenging aspect of creating this function was making the data within the “Change %” column—which measures the change in the DJIA, by percentage—numeric. Before making it numeric, it was necessary to remove the percentage sign from each cell. Once it is numeric, each cell’s value could be converted into its absolute value. This was necessary because a volatile market can go either up or down; a -7% change is just as volatile as a 7% change. Without making everything positive, these two 7% changes would have averaged out to volatility 0f 0% rather than the accurate 7%. Then the program finds the the mean of each of the monthly market changes during that administration and prints it for the user. The function for each month during each president was put into a larger function, containing the option to select a president and month and display the data. Below is a snapshot of the larger function.

def Single\_President():  
 print("This program will tell you the average volatility of the Dow Jones Industrial Average in each month, for every presidential administration, since the end of the cold war.")  
 print("")  
 print("1=H.W.Bush. 2=Clinton. 3=W.Bush. 4=Obama. 5=Trump")  
 print("")  
 president = input("Select a President by entering his corresponding number:")  
 print("")  
 if president == "4":  
 month = input("Select a month by typing the first 3 letters, with EVERY letter in lowercase:")  
 print("")  
 if month == "jan":  
 Jan\_Obama()  
 elif month == "feb":  
 Feb\_Obama()

The third function this program offers is to display to the user which presidential administration saw the greatest average volatility for a given month. This function was created manually, with pre-written statements and answers depending on the user input. The user was asked to put the first three letters of the month they are interested in, without any caps lock. Any error value would start the process over. Once the user selects their months, a message appears saying which administration saw the greatest volatility and another message saying what the average volatility was. Below is a snapshot of how the function is written.

def Ranking():  
 print("This program will allow you to select a certain month and see which presidential administration since the end"  
 "of the Cold War saw the most volatility on the DJIA.")  
 print("")  
 month = input("Select a month by typing the first 3 letters, with EVERY letter in lowercase:")  
 print("")  
 if month == "jan":  
 print("The administration of Barack Obama saw the greatest volatility on the DJIA during January.")  
 Jan\_Obama()  
 elif month == "feb":  
 print("The administration of Donald Trump saw the greatest volatility on the DJIA during February.")  
 Feb\_Trump()

The fourth function this program offers is a tool for the user to find indicators of the DJIA’s performance over the past half year. The user is asked to input the closing points of the DJIA over the past 20 days. The program then finds the 20-day, 10-day, 5-day, and 2-day moving averages of the DJIA, the 20-day, 10-day, 5-day, and 2-day percent changes of the DJIA, and its high-point, low-point, and range. Below is an excerpt of the function.

tday, yday, three\_days\_ago, four\_days\_ago, five\_days\_ago, six\_days\_ago, seven\_days\_ago, eight\_days\_ago, nine\_days\_ago, ten\_days\_ago, eleven\_days\_ago, twelve\_days\_ago, thirteen\_days\_ago, fourteen\_days\_ago, fifteen\_days\_ago, sixteen\_days\_ago, seventeen\_days\_ago, eighteen\_days\_ago, nineteen\_days\_ago, twenty\_days\_ago = eval(input("Enter the closing price today, yesterday, and the previous eighteen days, all separated by commas:"))  
print("")  
twenty\_day\_mavg = (tday + yday + three\_days\_ago + four\_days\_ago + five\_days\_ago +six\_days\_ago + seven\_days\_ago + eight\_days\_ago + nine\_days\_ago + ten\_days\_ago + eleven\_days\_ago + twelve\_days\_ago + thirteen\_days\_ago + fourteen\_days\_ago + fifteen\_days\_ago + sixteen\_days\_ago + seventeen\_days\_ago + eighteen\_days\_ago + nineteen\_days\_ago + twenty\_days\_ago)/20  
ten\_day\_mavg = (tday + yday + three\_days\_ago + four\_days\_ago + five\_days\_ago + six\_days\_ago + seven\_days\_ago + eight\_days\_ago + nine\_days\_ago + ten\_days\_ago)/10  
five\_day\_mavg = (tday + yday + three\_days\_ago + four\_days\_ago + five\_days\_ago) / 5  
two\_day\_mavg = (tday + yday) / 2  
highest\_closing\_price = (max(tday, yday, three\_days\_ago, four\_days\_ago, five\_days\_ago, six\_days\_ago, seven\_days\_ago, eight\_days\_ago, nine\_days\_ago, ten\_days\_ago, eleven\_days\_ago, twelve\_days\_ago, thirteen\_days\_ago, fourteen\_days\_ago, fifteen\_days\_ago, sixteen\_days\_ago, seventeen\_days\_ago, eighteen\_days\_ago, nineteen\_days\_ago, twenty\_days\_ago))  
lowest\_closing\_price = (min(tday, yday, three\_days\_ago, four\_days\_ago, five\_days\_ago, six\_days\_ago, seven\_days\_ago, eight\_days\_ago, nine\_days\_ago, ten\_days\_ago, eleven\_days\_ago, twelve\_days\_ago, thirteen\_days\_ago, fourteen\_days\_ago, fifteen\_days\_ago, sixteen\_days\_ago, seventeen\_days\_ago, eighteen\_days\_ago, nineteen\_days\_ago, twenty\_days\_ago))  
spread\_of\_closing\_prices = highest\_closing\_price - lowest\_closing\_price  
twenty\_day\_growth = (tday - twenty\_days\_ago)/(twenty\_days\_ago)  
ten\_day\_growth = (tday - ten\_days\_ago) / (ten\_days\_ago)  
five\_day\_growth = (tday - five\_days\_ago) / (five\_days\_ago)  
two\_day\_growth = (tday - yday) / (yday)  
print("")  
print("The 20-day moving average is:", twenty\_day\_mavg)  
print("The 10-day moving average is:", ten\_day\_mavg)  
print("The 5-day moving average is:", five\_day\_mavg)  
print("The 2-day moving average is:", two\_day\_mavg)

The fifth function this program offers is an “Exit” option, and it is the shortest function written. It asks the user to enter the “ENTER” button should they wish to Exit the program. The entire function is:

def Exit():  
 input("Press Enter to Exit.")

All 5 of these functions are put a larger function, called the “Execution” function. The Execution function has various *if*, *elif*, and *else* commands that directs the user to certain functions. If the user enters the number “4”, for instance, when asked to select a function, they will be directed toward the function that finds various indicators of the DJIA. Below is a part of the Execution Function.

def Execution():  
 print("")  
 selection = input("Please select 1, 2, 3, 4, or 5:")  
 print("")  
 if selection == "1":  
 Display\_Data()  
 Execution()  
 elif selection == "2":  
 Single\_President()  
 Execution()

The instructions are explained in a separate function, known as the “Instructions” function. The Instructions functions provides a basic framework of the Python program, and it provides a number key for the user to use to select a function. Below is an excerpt of the instructions.

def Instructions():  
 print("")  
 print("This program focuses on 5 presidential administrations since the end of the Cold War: George H.W. Bush, Bill Clinton, George W. Bush, Barack Obama, and Donald Trump.")  
 print("")  
 print("The data begins at December 1991 (in which the USSR dissolved).")  
 print("")  
 print("Additionally, this program inclues a tool to find the recent moving average of the DJIA.")  
 print("")  
 print("This program lets you select one of several options.")  
 print("")  
 print("1= Display the DJIA monthly data for various presidents.")  
 print("")

The Instructions function and the Execution function are separate, and have to be combined into the main function to bring all the directions and the functions together. This is the main function, and it is accordingly called “Main”.

def main():  
 Instructions()  
 Execution()  
  
main()

To reiterate, this program is primarily intended to provide insights into the performance of the Dow Jones Industrial Average since the end of the Cold War. The most important question being asked here, is which administrations saw the most volatility during given months. Investors can use this information to see which months may be the most volatile by comparing the policies of one presidential administration to another. The other important question being asked is what the average monthly volatility was during any given administration. Again, policies could be compared to the current administration to develop a sense of how volatile the DJIA is set to be.