CARNEGIE MELLON UNIVERSITY - AFRICA

DATA, INFERENCE & APPLIED MACHINE LEARNING

(COURSE 18-785)

Professor Patrick McSharry

Assignment 1

MUNYANEZA Kenny Roger

4th September 2023

Introduction

This is a final report of DIAML assignment 1 that comprises of solutions for the ten questions in assignment 1.

Tools used

- Python
- Jupyter Notebook

Question 1

Implementation process used

The program I developed is mainly composed of a function that takes two parameters, namely the given thickness of the paper, and the height of Everest Mountain. I created a variable that serves as a counter that increases by one thought out the iterations of a while loop that runs as long as the parameter that holds the thickness of the paper is less than the parameter that holds the height of Mountain Everest. On each iteration of the while loop the thickness of the paper is doubled, to simulate the physical increase in the thickness of a paper when it is folded, and the mentioned counter is incremented by one. The function returns the counter once the while loop stops iterating. This function needs to be called at a zero indentation to serve its purpose.

Results: The result of the program is 24 folds.

Findings: For a paper that is 1mm thick to exceed the height of Mountain Everest, it would be required to fold that paper 24 times.

Question 2

Third-party libraries used

- numpy

Implementation process used

This question literally requires calculating the Half-Life of water in the reservoir. The Half-Life of a decaying quantity depends on the decay rate when the decay function is equated to half the of the original quantity. This creates an equation whereby the decay rate and the time are the only unknowns, and one can be calculated provided the value of the other.

The resultant equation is:

$$\mathrm{V(t)}e^{-at} = \frac{1}{2}\,\mathrm{v(t)} \;\Leftrightarrow\; e^{-at} = \frac{1}{2} \Leftrightarrow t = \frac{-\ln(2)}{-\mathrm{a}}$$

The above function can be used to calculate the half-life of the water in a reservoir. [1]

The function I created takes the decay rate provided in the question as parameter and computes the time by dividing the negative of the natural logarithm of two by the decay rate. The function returns the time required thereafter.

To obtain the time elapsed for the volume of water to fall below its original value, I added a single time unit to the half life because the volume of will fall below half its original value in the next time unit after the half-life. I rounded of the result obtained to two decimal points.

Results: 6.94 Time Units

Findings:

Time required for the volume to drop below half of the original value = 6.94 Time Units.

Question 3

Implementation process used

To solve this problem, my program uses a function that expects three parameters that hold the principle deposited, the interest rate value, and the frequency of compounding the interest consecutively. The rate is divided by 100 to convert it to a ratio, and for a loop of five iterations, starting from one to five, an amount at the end of the year 1, 2, 3, 4, and 5 is calculated.

The formula below is used to calculate the amount of money made in a given time at a certain frequency of periodic compounding of interests. [2]

$$A = P(1 + \frac{r}{n})^{nt}$$

Where:

A = Amount obtained after compound interest

P = Principle invested

r = interest rate

n = frequency of compounding interests

Results:

- The amount of money obtained after year 1 = $ 105
- The amount of money obtained after year 2 = $ 111
- The amount of money obtained after year 3 = $ 116
- The amount of money obtained after year 4 = $ 122
- The amount of money obtained after year 5 = $ 128

Question 4

Implementation process used

To solve this problem, my program uses a function that expects two parameters, whereby the first parameter holds the value of the loan acquired by and the second parameter holds the value of the interest rate of the loan acquired. First, the interest rate is converted to a ratio by dividing the second parameter by one hundred. After that, for a loop running from one to three, a monthly payment required to pay if the loan is to be paid back in one, two, or three years respectively is calculated.

The formula below is used to calculate the monthly payment to be made for a loan that is to be paid back in a given period. [2]

$$A = P\left(\frac{r(1+r)^n}{(1+r)^n - 1}\right)$$

Where:

A: The amount to be paid

P: The acquired loan amount

r: interest rate

n: number of payments for the loan per month

Results:

- Payment necessary to pay back the loan in 1 year(s) = $ 1777
- Payment necessary to pay back the loan in 2 year(s) = $ 942
- Payment necessary to pay back the loan in 3 year(s) = $ 665

Question 5

Third-party libraries used

- matplotlib
- numpy

Implementation Process Used

To solve this question, I used two functions, whereby the first function computes the break-even day of the business investment, and the second function is called once the breakeven day is determined to visualize it on a graph plotting the days passed before the break-even day and the cumulative profits on each day.

Starting with the first function of the program, it expects four parameters, which are the principal investment, the growth rate of customers, the number of customers expected on day one and the profit made per customer consecutively.

The logic used in this function aims to determine the period in which the amount invested would fall to zero if each daily profit is deducted from the initial investment every day. Since the number of customers expected on day one was provided, I calculated the profit made on that day by multiplying the number of customers to the amount of profits per customer and I deducted the profit made on day one from the initial investment. Also, I counted that day as the first day elapsed before the break-even day hence, I created a counter that will increment on each day until the break-even day by initiating it with one and I created a counter that will sum up all the daily profits made daily until the break-even day by initiating it with the profit for day one. Both the days and cumulative frequency counter are respectively appended to a List of days elapsed and a List of cumulative profit made until the breakeven day.

After, I created a while loop that will end when the variable holding the initial investment falls to zero or below zero. During this loop, the processes mentioned in the previous paragraphs are repeated. The processes repeated are:

- Increase the profit of the previous day by the rate at which the customer number is expected to grow per day.
- Deduct the daily profit obtained from the initial investment.
- Increment the counter that counts the days elapsed by one
- Increment the cumulative profits made until the current day by the profit made on that day
- Append the counter of the current day to the List of days elapsed
- Append the counter of the cumulative profits to the List of profits made

At the end of this loop, the mentioned second function will be called to visualize the results of the first function which is mainly the days elapsed and the cumulative profit made until the break-even day.

The second function expects two parameters which are of List data structures whereby the first list contains the days elapsed and the second list contains the cumulative profits made until the break-even day. I used the “pyplot” submodule of “matplotlib” to draw a line graph with days on the x-axis and cumulative profits made on the y-axis.

Results:

```
It will take 70 days to recover the investment made as shown in the graph below:
```

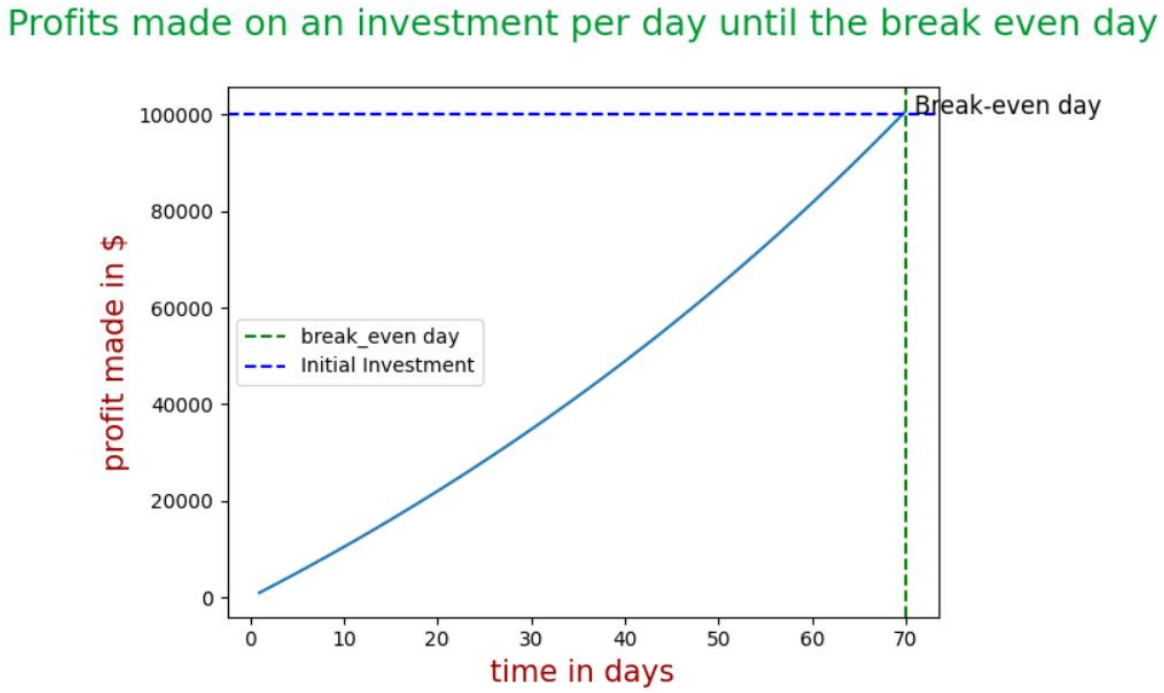


Figure 1: Graph showing days elapsed against cumulative profits made until the breakeven day

In the figure above, the days elapsed until the breakeven day are plotted on the x-axis (on a scale of 1:10) against the cumulative profits made of each day until the breakeven day plotted on the y-axis (on a scale of 1:20000).

The resultant graph is a smooth curve that is steadily increasing. The smooth curve is consistent with the growth rate of 1% provided in the question because the growth rate is exponential which result to the line graph being curved and the growth rate is low which results in line graph's low curvature.

Insights:

- Profits of a business grows at the same rate as the customer's growth rate

Question 6

Libraries Used

- **pandas**: It was used to manipulate the data-frame provided which involves cleaning data, merging data, and reshaping the dataset
- **numpy**: Used to find the maximum and minimum value in List data structure

- **matplotlib**: Used to plot the 2D graph of the results obtained
- **scipy**: Used the submodule “interpolate” to interpolate the missing values in the dataset
- **xlrd:** Used to read ‘.Xls’ files

Implementation Process Used

This problem, required to read the file provided using the panda’s module and analyze the cleanliness of the data. Soon, I realized that the data was not cleaned as many dates were missing between the start date and the end date mentioned. I started by generating all dates between the mentioned start date and end date and I assigned those dates to a Series which is one dimensional array in the panda’s module that indexes all the data within hence become able to serve as normal data frame with value and index columns. This allowed me to merge the data-frame provided with the array and this resulted into a merged data-frame with new instances that are blank corresponding to the new dates merged.

I used the ‘interp1d’ function that belongs to the interpolate module in SciPy to linearly interpolate all columns that had missing values, namely the Cases column, the Deaths column, and the number of days column. This gave me a cleaned dataset which a saved under a new name ‘cleaned_dataset.csv’ for further visualization and estimation purposes. Next, I read the new csv dataset using pandas and I extracted the columns: Date, Cases, and Death. Then, I plotted these three columns on a line graph to visualize the distribution of Deaths with time against the distribution of Cases with time.

To estimate the dates when the thresholds 100, 500, 1000, 2000, and 5000 were exceeded, I assigned these thresholds to a loop, and I run a nested loop that compares each threshold to the values in both the Cases list and Deaths list extracted earlier. In case a threshold is exceeded for the first time in the minor loop I re-plotted the coordinates of that value already plotted on the graph with a colored marker to highlight that date. [3] [4] [5] [6]

Results:

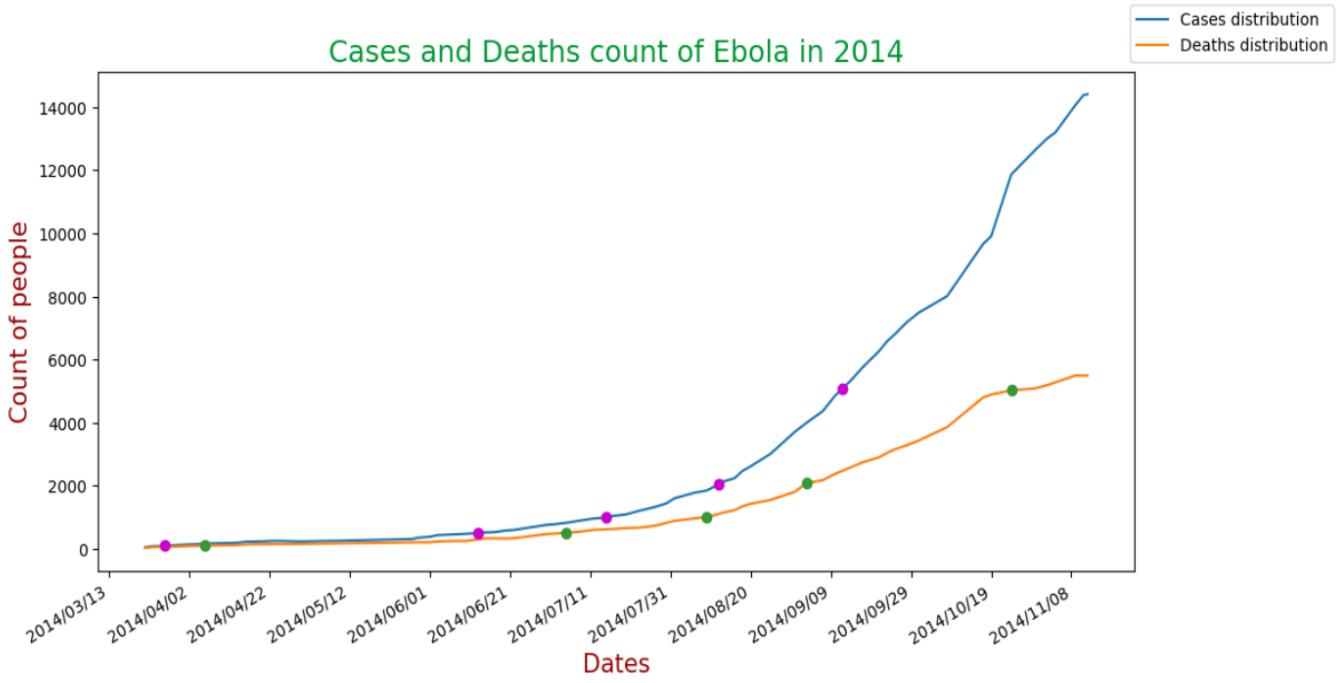


Figure 2: Figure showing the distribution of deaths against cases

The graph above shows the distribution of cases against deaths of Ebola in 2014 and it can be inferred from the graph that the cases distribution was higher than the distribution of Deaths that resulted from the Cases.

Insights:

The rate of growth of cases and deaths is not consistent with time. Rather, it increases with time as it can be inferred from the graph because the lines have higher slope as time goes by.

Question 7

Libraries Used

- **pandas**: It was used to manipulate the data-frame provided which involves reading the dataset in this question
- **numpy**: Used to find the maximum and minimum value in List data structure

To solve this problem, I have two functions whereby the first function reads the cleaned csv file used in question 6 and return that dataset. The second function reads the dataset and extracts the columns Cases and Deaths in two lists, respectively. After those two loops are run with one loop running for iterations equal to the length of the Cases list and the other running for the number of iterations equal to the length of the cases list. During the iteration A daily rate is computed by multiplying the difference between the cases for two consecutive days with one hundred and dividing by the present day's cases

and its is appended to a list the serves as a collection of all daily rates. This is repeated for the Deaths list. Once both daily growth rates are saved in a list, the average growth rate as a percentage for both Cases and deaths is computed by dividing the sum of percentage growths by the number of percentage growths. This is done for the Cases and Deaths columns of the dataset.[7]

Results:

The average daily growth-rate of Ebola cases in 2014: 2.506522236509118 %

The average daily growth-rate of deaths caused by Ebola in 2014: 2.330608629197429 %

Insights: Even though the cases and deaths caused by Ebola are not equally distributed with time, their average growth rates are almost equal.

Question 8

Libraries Used

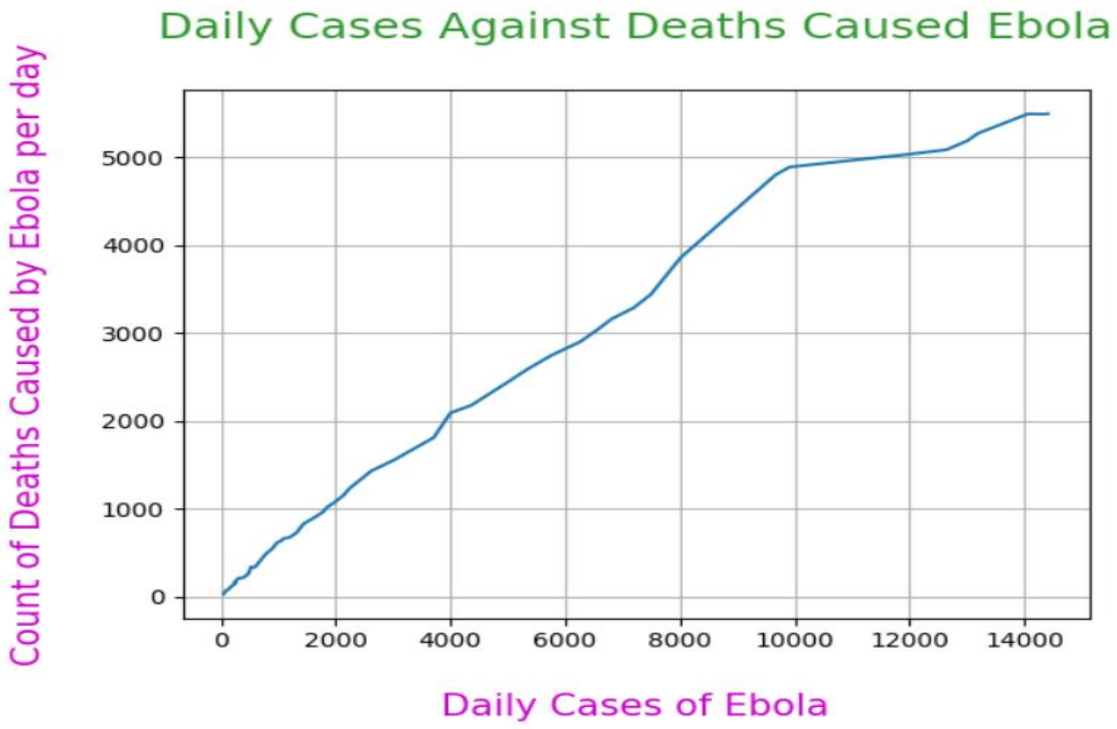
- **pandas**: It was used to manipulate the data-frame provided which involves reading the dataset in this question

Implementation Process Used

To solve this problem, I used two functions. The first one reads the dataset cleaned in the previous question and return a pandas data-frame. The second function leverages the first function to read the csv file and extracts the Cases and Deaths columns. After, the function plots the cases against deaths on a line graph, whereby the cases are plotted on the x-axis and the and the deaths are plotted on the y-axis. The graph is styled with some custom font qualities and a grid. At the end, the ratio of deaths to cases is calculated by dividing the sum of values in the Deaths list by sum of values in the cases list.

The function is then called to in the main execution of the program to return the average ratio of deaths to cases.

Results:




```
The ratio of deaths count to daily cases of Ebola is 0.47
```

Figure 3: Graph showing the daily cases against daily death caused by Ebola

It can be inferred from the graph that the daily cases and the resultant deaths caused by Ebola are directly proportional.

Question 9

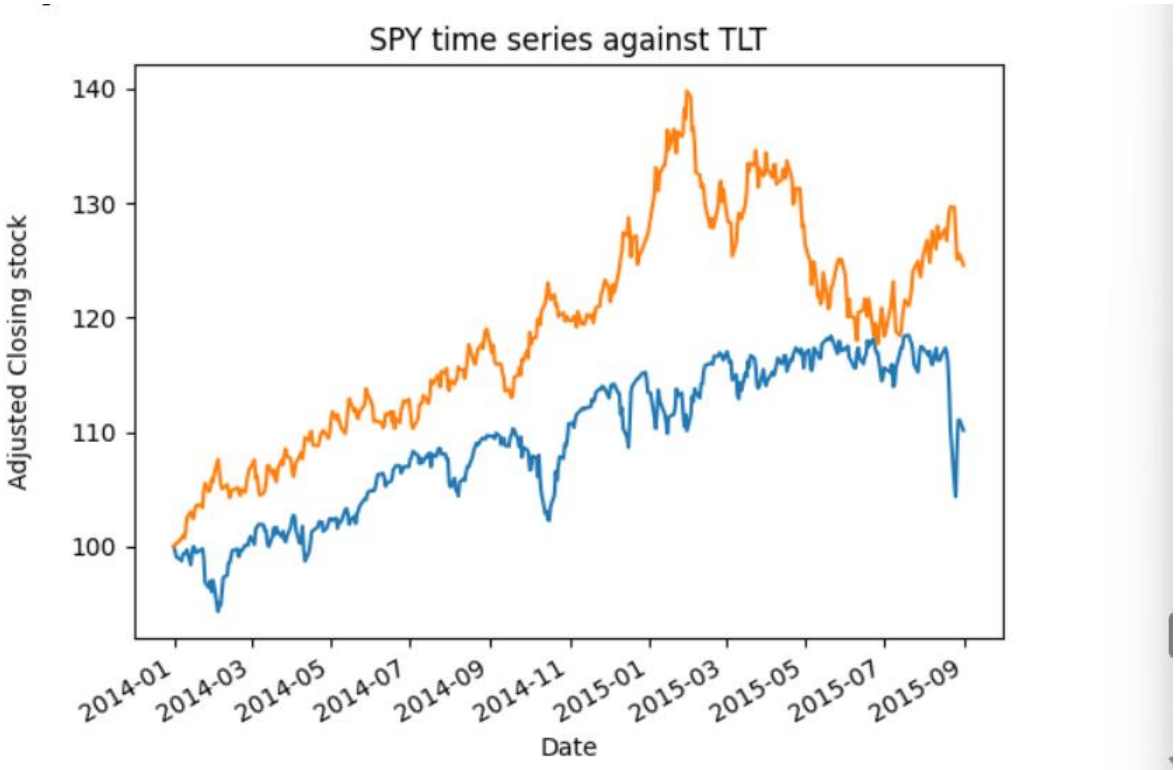
Libraries Used

- **matplotlib**: Used to plot the 2D graph of RSY and TLT time series
- **pandas**: It was used to manipulate the data-frame provided which involves reading the dataset in this question

Implementation Process Used

To solve this problem, I read the two timeseries using pandas by parsing the dates into timestamps given that they were in string format and setting the 'Date' column to be used for indexing the time series datasets. Next, I used a function to normalize the Adjusted Closing stock column for both RSY and TLT timeseries. The function simply multiplies each value in each column by one hundred divided by the first value of each Adjusted Closing stock column. After that I plot the time series using line graph.

Results:



Findings: The two-time series different distribution with time even when scale to a common starting point which is $100.

Question 10

Libraries Used

- **pandas**: It was used to manipulate the data-frame provided which involves reading the dataset in this question

Implementation Process:

To solve this problem, I read both the SPY and TLT timeseries using the panda framework and recorded the first value in the Adjusted Closing stock column of the SPY time series and I recorded the first value in the Adjusted Closing stock column of TLT. Also, I created lists that will record the daily returns for SPY and TLT timeseries respectively. Then I run a loop that runs from the second index of the Adjusted Closing stock column until the last index. And for each iteration it records the current daily price and the price of a previous date into variables and applies the formula

$$r(t)= p(t)/p(t-1)-1$$

- p(t) = current daily price
- p(t-1) = price of a previous date
- r(t) = daily return

The first executes this action for the SPY time series and the second loop does the same action for TLT timeseries

At the end of these two loops, I had two lists that have recorded daily returns percentages for both the SPY and TLT timeseries.

I used the first and second lists to calculate the average, minimum value and maximum value of percentage daily returns for RSY and TLT timeseries, respectively.

Result:

```
========SPY statistics============

The average daily return value in SPY is 0.4020361674323234

The minimum daily return value in SPY is -4.210697162689259%

The maximum daily return value in SPY is 157.853638%


========TLT statistics============

The average daily return value in TLT is 0.25539787121237784

The minimum daily return value in TLT is -2.4324931167024166%

The maximum daily return value in TLT is 83.814583%
```

Figure 4:Image showing the results of question 10

[1] ‘Exponential Growth and Decay | College Algebra’. https://courses.lumenlearning.com/waymakercollegealgebra/chapter/exponential-growth-and-decay/ (accessed Sep. 03, 2023).
[2] ‘Recitation_1_F2023.pdf: Data, Inference, and Applied Machine Learning - DIAML’. https://canvas.cmu.edu/courses/36029/files/9892520?module_item_id=5577714 (accessed Sep. 03, 2023).
[3] ‘Interpolation (scipy.interpolate) — SciPy v1.11.2 Manual’. https://docs.scipy.org/doc/scipy/tutorial/interpolate.html (accessed Sep. 04, 2023).
[4] ‘Formatting date ticks using ConciseDateFormatter — Matplotlib 3.7.2 documentation’. https://matplotlib.org/stable/gallery/ticks/date_concise_formatter.html (accessed Sep. 04, 2023).
[5] ‘numpy.where() in Python’, *GeeksforGeeks*, Oct. 31, 2018. https://www.geeksforgeeks.org/numpy-where-in-python/ (accessed Sep. 04, 2023).
[6] ‘pandas.DataFrame.merge — pandas 2.1.0 documentation’. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html (accessed Sep. 04, 2023).
[7] ‘Growth Rates: Formula, How to Calculate, and Definition’, *Investopedia*. https://www.investopedia.com/terms/g/growthrates.asp (accessed Sep. 04, 2023).