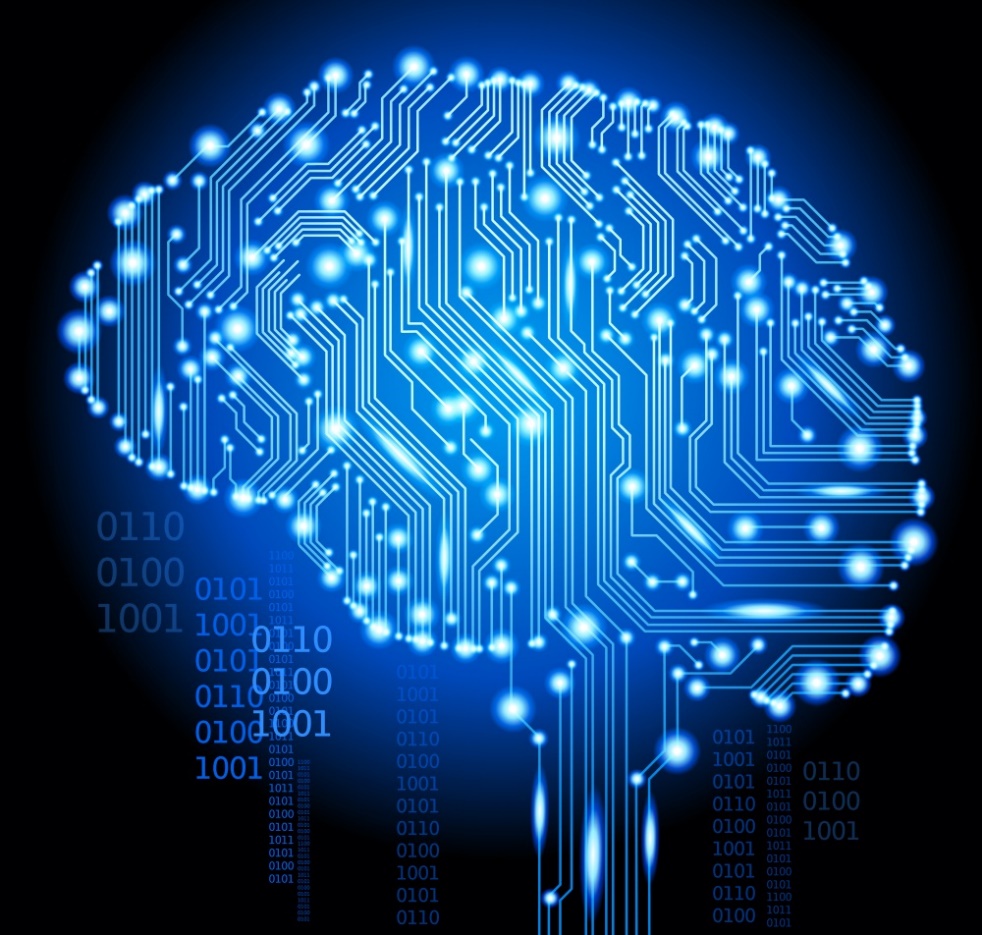


COS30018 – Intelligent Systems

**REPORT**

TASK B2 – DATA PROCESSING 1



**TABLE OF CONTENTS**

*Table of contents…………………………………………………………………………………………………………………2*

*I. Processing data solution…………………………………………………………………………………………………..3*

*1. Load data………………………………………………………………………………………………………………3*

*2. Split data……………………………………………………………………………………………………………….4*

*3. Save data……………………………………………………………………………………………………………….4*

*4. Scale data………………………………………………………………………………………………………………5*

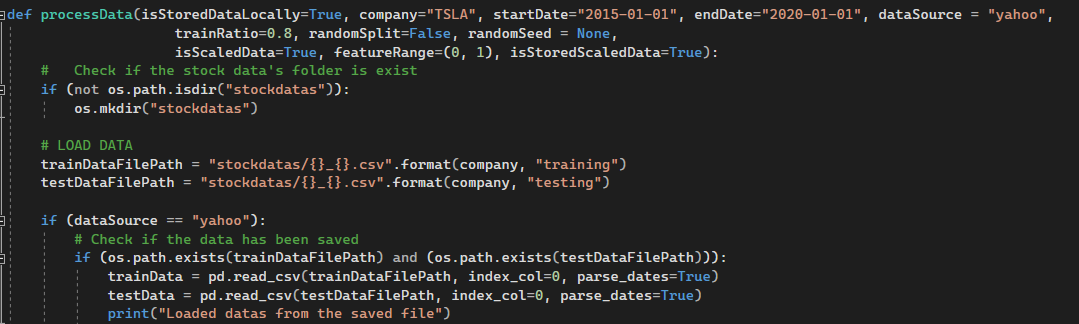
*II. Testing…………………………………………………………………………………………………………………………….7*

**PROCESSING DATA SOLUTION**

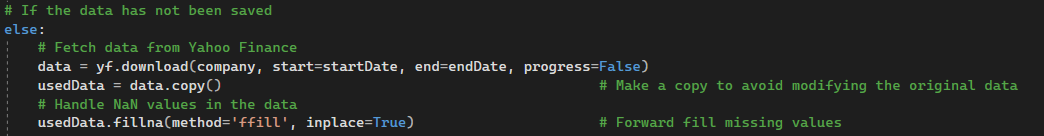
For this processing task, I have created a new file of “dataProcessing.py”, with the method “processData”, returning the processed training and testing data.

**1. Load data**

Firstly, “stockdatas” is the folder in which I intended to put all the saved data files, so I have checked if this folder and the .csv files for the testing and training data existed using “os”:



* If so, simply load the saved data by the “read\_csv” method of “pandas” library.
* If not:
  + The whole data is downloaded from Yahoo Finance through the “yfinance” library
  + To avoid unexpected changes, the data is copied to “usedData”
  + Then, the “usedData” goes to the “Handling NaN” process, using the “fillna” method of “pandas” library. In my code, it uses “ffill” meaning forward fill: filling missing values with the last known non-missing value in the column. Also, “inplace=True” specifies that the operation should be performed in-place on the “usedData” dataframe, modifying it directly.

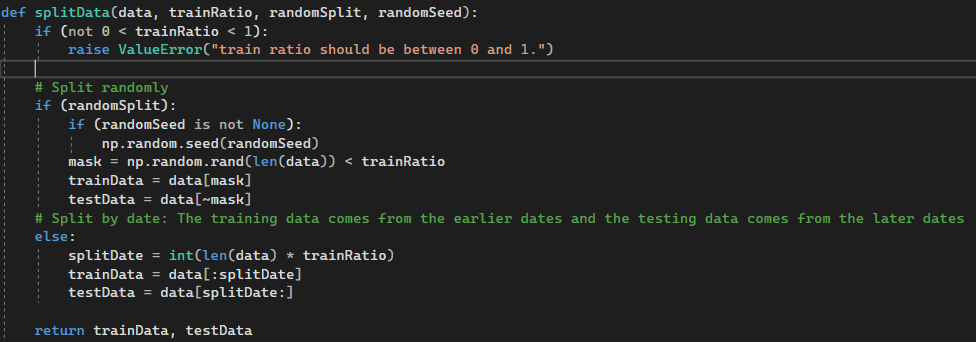


**2. Split data**

If the data has been downloaded by “yfinance”, after finishing the above stages, the “usedData” will be split to training and testing data:



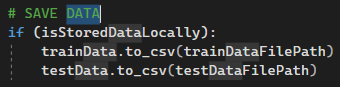
A method of “splitData” has been created for this, it takes 4 parameters, including “data” as the variable for initial data, “trainRatio” as the splitting ratio between training and testing data (0.8 by default), “randomSpilt” as the boolean variable indicating whether the splitting process is implemented by date or randomly, “randomSeed” as the seed index for the random process:



After splitting data, the method returns both training and testing data.

**3. Save data**

If the parameter variable of “isStoredDataLocally” is “True”, the training and testing data from the splitting process is stored in the “stockdatas” folder by the format “{Company’s symbol}\_{training/testing}.csv”:

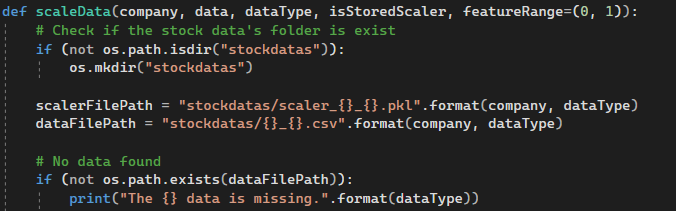


**4. Scale data**

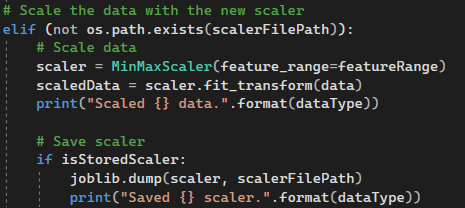
The final method created in this “dataProcessing.py” file for this task is “scaleData”, used for the scaling process.

The method takes 5 parameters, including the “company” for the company’s symbol, “data”, “dataType” (training or testing), “isStoredScaler” decides whether the scaler is saved in “stockdatas” or not, “featureRange” used in “MinMaxScaler” of “scikit-learn” library to specify the desired range of the scaled features after modifying (scaling).

Initially, it will check if the folder “stockdatas” and the .csv files exist or not by “os”.



If all of them exist, but the scaler file has not been saved before, then it createx a new instance of “MinMaxScaler”, and uses its “fit\_transfrom” method:



This method uses the following formula: Given:

* x is the original feature value;
* min(x) is the minimum value of the original feature in the dataset;
* max(x) is the maximum value of the original feature in the dataset;
* minValue is the minimum value of the desired feature range (e.g., 0);
* maxValue is the maximum value of the desired feature range (e.g., 1),

Now the scale feature for the original one is: xScale = (x – min(x)) / (max(x) – min(x)) \* (maxValue – minValue) + minValue.

For example, as the default, “featureRange” has the value of (0, 1); Given the original data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Close | Adj Close | Volume |
| 12/31/2021 | 357.813324 | 360.6666565 | 351.5299988 | 352.2600098 | 352.2600098 | 40733700 |
| 1/3/2022 | 382.5833435 | 400.3566589 | 378.6799927 | 399.9266663 | 399.9266663 | 103931400 |
| 1/4/2022 | 396.5166626 | 402.6666565 | 374.3500061 | 383.1966553 | 383.1966553 | 100248300 |
| 1/5/2022 | 382.2166748 | 390.1133423 | 360.3366699 | 362.706665 | 362.706665 | 80119800 |
| 1/6/2022 | 359 | 362.6666565 | 340.1666565 | 354.8999939 | 354.8999939 | 90336600 |
| 1/7/2022 | 360.1233215 | 360.3099976 | 336.6666565 | 342.3200073 | 342.3200073 | 84164700 |

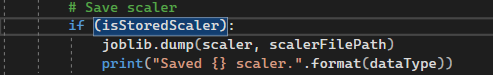
Let’s take the values of the Open column:

* The original Open values are: 357.813324, 382.5833435, 396.5166626, ...
* The minimum value of Open is 357.813324, and the maximum value is 396.5166626.
* The feature\_range is (0, 1).

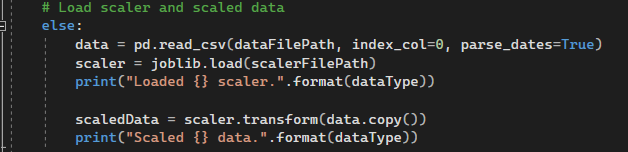
Applying the above formula:

* For the first value (357.813324): xScaled = (357.813324 - 357.813324) / (396.5166626 - 357.813324) \* (1 - 0) + 0 = 0
* For the second value (382.5833435): xScaled = (382.5833435 - 357.813324) / (396.5166626 - 357.813324) \* (1 - 0) + 0 = 0.366050923
* … (and so on)

Following that, if the boolean variable “isStoredScaler” is “True”, the method uses the “dump” method of the library “joblib” to save the scaler:



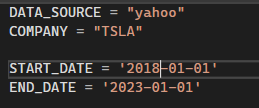
For the third case, if the folder “stockdatas”, the data files, and the scaler file have already been in the repository, the data and the scaler is loaded by “pandas.read\_csv” and “joblib.load” respectively, then, a copy of data will be scaled using “transform”:



The method finally returns the value of the scaled data and the scaler.

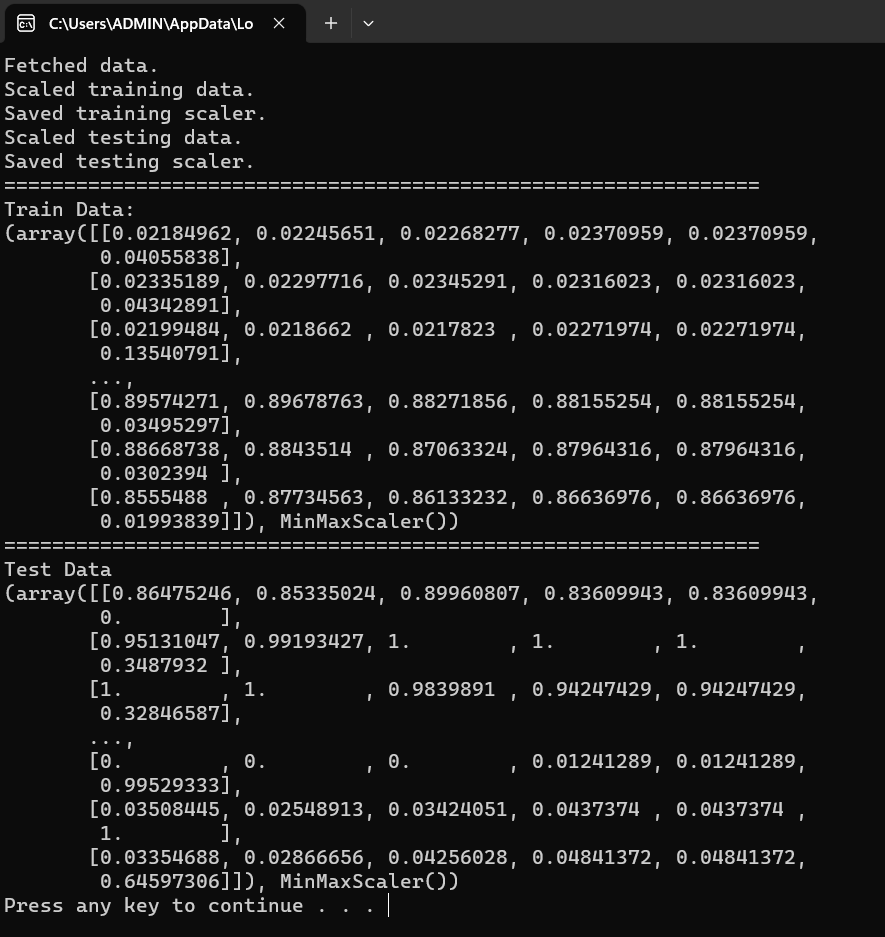
**TESTING**

For testing, I have created another file of “parameters.py”, including data source (yahoo), company’s symbol, start and end date:



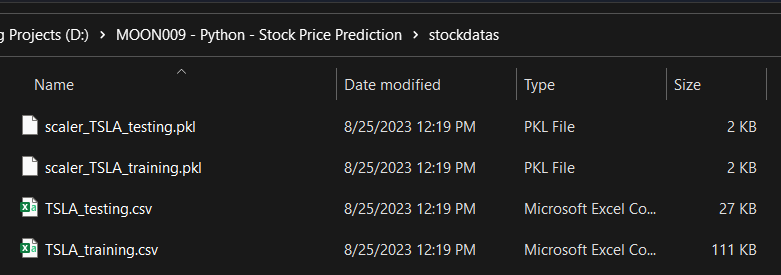
Firstly, testing the “processData” function with:

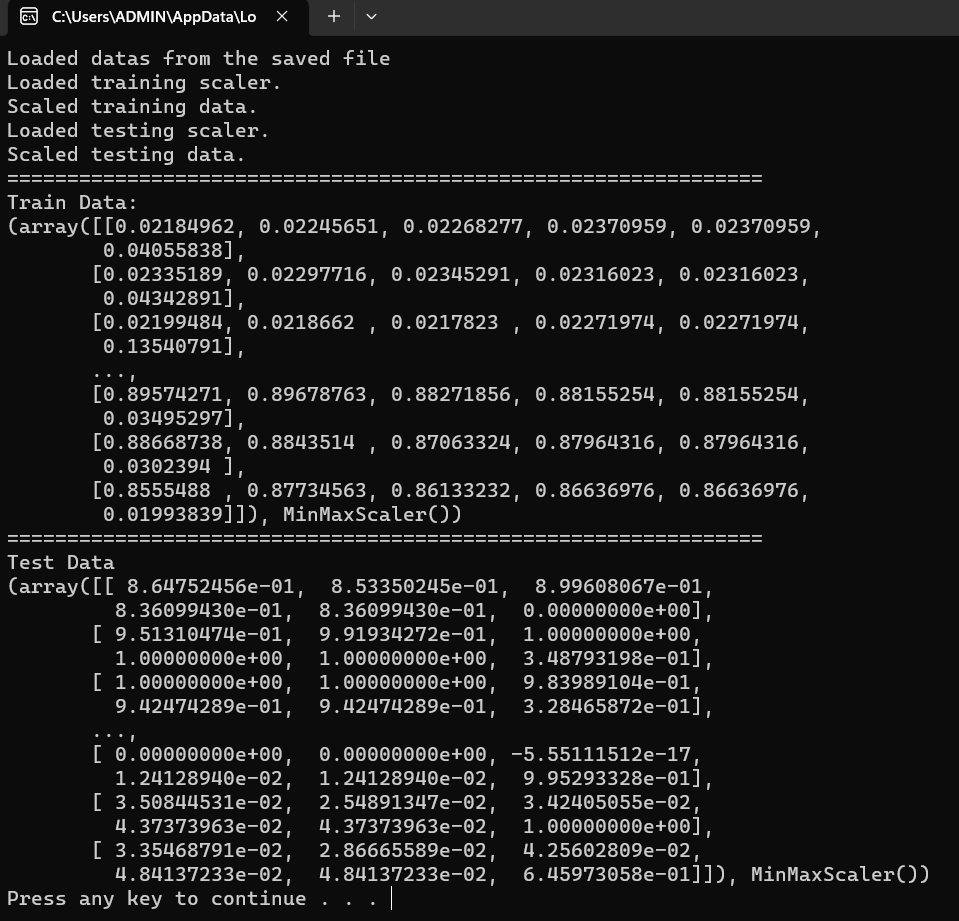
* isStoreDataLocally = True, company = “TSLA”, startDate = “2018-01-01”, endDate = “2023-01-01”, dataSource = “yahoo”;
* trainRatio = 0.8, randomSplit = False, randomSeed = None;
* isScaledData = True, featureRange = (0, 1), isStoredScaler = True;
* No “stockdatas” folder:



Secondly, testing with:

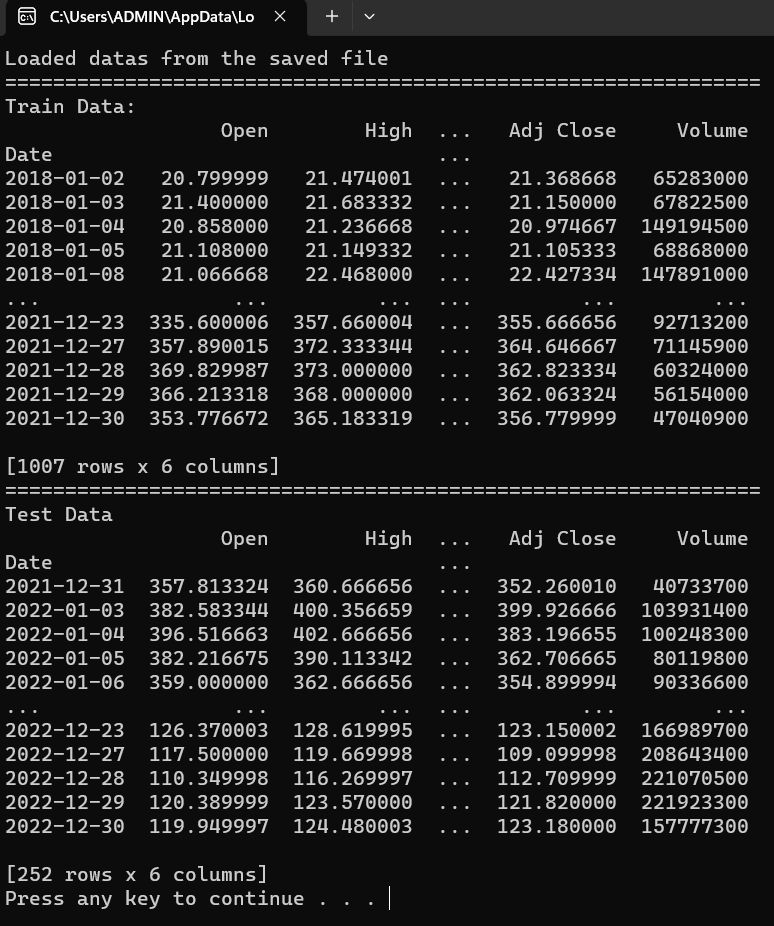
* isStoreDataLocally = True, company = “TSLA”, startDate = “2018-01-01”, endDate = “2023-01-01”, dataSource = “yahoo”;
* trainRatio = 0.8, randomSplit = False, randomSeed = None;
* isScaledData = True, featureRange = (0, 1), isStoredScaler = True;
* Having “stockdatas” folder with training, testing data and scaler:





Then, I tested with:

* isStoreDataLocally = True, company = “TSLA”, startDate = “2018-01-01”, endDate = “2023-01-01”, dataSource = “yahoo”;
* trainRatio = 0.8, randomSplit = False, randomSeed = None;
* isScaledData = False, featureRange = (0, 1), isStoredScaler = False; (No scaling process)
* Having “stockdatas” folder with training, testing data:

****