Title: Task B-1 Report

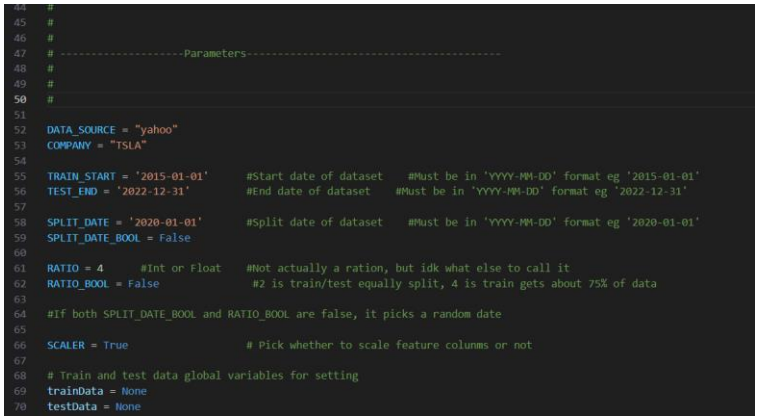
Author: Robin Findlay-Marks, s103603871

Setup Attempt:

Option B - Task 2: Data processing 1 took a long time to complete and had a lot of trouble understanding what I was meant to do.
The task seems to suggest that it is recommended for me to copy the code from P1 and then explain how it works. However P1 seems to do thing so differently from v0.1 that it is impossible without overhauling v0.1 to the point where it is basically P1 itself. I instead decided to figure out how to get the loading and processing done myself. I struggled for a while to get it to work properly but I'm fairly happy with what I did.
I made the program download a single dataframe, rather than two, which is defined with two dates. This dataframe is then split using one of three methods, with a third date, a ratio or simply randomly. I did this by having several smaller functions that first of all handle which splitting method to use, by checking which parameters are set, and second of all having functions dealing with the splitting by ratio or date.
The NaN issue was a annoying. I didn't understand what I meant by the "NaN issue in the data" as I couldn't find it referenced anywhere. I did see that something about NaNs was referenced in the P1 code, and after I realised that I researched what NaNs are and used the line from P1 to remove them.
Task 2 – 1.e was very confusing, and I'm not sure if it's implemented or not. It seems to want me to scale the feature columns and store them as data structure, but it already scales the feature colunms and stores them as an array, which is a data structure. So as far as I can tell, this is already done, but that doesn't seem right.

The rest of the explanations of how my code works can be found as comments in the code.
I have included screenshots for easy access.

Parameters:



```
#
#
#
# --------------------Parameters-----------------------------------------
#
#
#

DATA_SOURCE = "yahoo"
COMPANY = "TSLA"

TRAIN_START = '2015-01-01'      #Start date of dataset    #Must be in 'YYYY-MM-DD' format eg '2015-01-01'
TEST_END = '2022-12-31'         #End date of dataset    #Must be in 'YYYY-MM-DD' format eg '2022-12-31'

SPLIT_DATE = '2020-01-01'       #Split date of dataset    #Must be in 'YYYY-MM-DD' format eg '2020-01-01'
SPLIT_DATE_BOOL = False

RATIO = 4      #Int or Float    #Not actually a ration, but idk what else to call it
RATIO_BOOL = False               #2 is train/test equally split, 4 is train gets about 75% of data

#If both SPLIT_DATE_BOOL and RATIO_BOOL are false, it picks a random date

SCALER = True                   # Pick whether to scale feature columms or not

# Train and test data global variables for setting
trainData = None
testData = None
```

Downloading and storing of data as file:

```python
# Function for checking if the data is already downloaded (needs internet connection to download)
# If the data is NOT in a file, it downloads the data and makes a csv file and returns the data
# If the data IS in a file, it reads the data from the file and returns the data
def checkFiles(filename):
    if (os.path.exists(filename)):
        #Read csv file and return the data inside
        data = pd.read_csv(filename)
        # NaN values from pandas are values that are not present. For example in stocks if the stock data for a
        # specific day was not recorded, i believe it would still have a record for that day, only the values
        # would be NaN or 'Not a Number'

        #what dropna() does is simply remove the missing values from the dataset
        data.dropna(inplace=True)
        return data
    else:
        #Download data from online
        data = yf.download(COMPANY, start=TRAIN_START, end=TEST_END, progress=False)

        # Save data to csv file
        data.to_csv(filename)
        # For some reason it needs to read it from the file otherwise it won't work
        data = pd.read_csv(filename)
        # remove NaN values from the dataset
        data.dropna(inplace=True)
        return data
```

Splitting data by date

```python
# This function gets the datafile name as well as the split date
# it then runs the file checker to get the dataset, then splits the dataset at the split date
def getDataSplitDate(filename, splitDate):
    df = checkFiles(filename)

    # Make it know that the date colunm is indeed a date
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.set_index(df['Date'])
    df = df.sort_index()

    #Convert input to datetime, add 1 day, then convert back to string+
    date = datetime.strptime(splitDate, '%Y-%m-%d')
    testStartDate = date + timedelta(days=1)
    testStartDate = testStartDate.strftime('%Y-%m-%d')

    # create train/test partition
    global trainData
    trainData = df[TRAIN_START:splitDate]
    global testData
    testData = df[testStartDate:TEST_END]
    print('Train Dataset:',trainData.shape)
    print('Test Dataset:',testData.shape)

    #trainData.to_csv("trainfilename.csv")
    #testData.to_csv("testfilename.csv")        #test that the data is split correctly
```

Splitting data by ratio

```python
# This function gets the datafile name as well as the "ratio" number
# it then runs the file checker to get the dataset, then splits the dataset at the split date
def getDataRatio(filename, ratio):
    df = checkFiles(filename)

    # Make it know that the date colunm is indeed a date
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.set_index(df['Date'])
    df = df.sort_index()

    # Convert strings to dates
    date1 = datetime.strptime(TRAIN_START, '%Y-%m-%d')
    date2 = datetime.strptime(TEST_END, '%Y-%m-%d')

    # do math to get the date we want
    trainEndDate = date2 + (date1 - date2) / ratio

    #Convert input to datetime, add 1 day, then convert back to string+
    print("Middle : " + trainEndDate.strftime('%Y-%m-%d'))
    testStartDate = trainEndDate + timedelta(days=1)
    #testStartDate = testStartDate.strftime('%Y-%m-%d') # i don't remember why i commented this, but it works

    # create train/test partition
    global trainData
    trainData = df[TRAIN_START:trainEndDate]
    global testData
    testData = df[testStartDate:TEST_END]

    #trainData.to_csv("trainfilename.csv")
    #testData.to_csv("testfilename.csv")        #test that the data is split correctly
```

Processing of parameters

```python
def getData(): #Main function for deciding which split method was chosen
    #Make filename for the saved data file
    ticker_data_filename = os.path.join("data", f"{COMPANY}_{TRAIN_START}_{TEST_END}.csv")
    if (SPLIT_DATE_BOOL):
        getDataSplitDate(ticker_data_filename, SPLIT_DATE)
    elif (RATIO_BOOL):
        getDataRatio(ticker_data_filename, 3.5)
    else: #Random Date
        #Convert dates to datetime
        dateStart = datetime.strptime(TRAIN_START, '%Y-%m-%d')
        dateEnd = datetime.strptime(TEST_END, '%Y-%m-%d')
        #Get random date inbetween the start and end
        random_date = dateStart + (dateEnd - dateStart) * random.random()
        #convert back to string
        random_date = random_date.strftime('%Y-%m-%d')
        #Use random date as split
        getDataSplitDate(ticker_data_filename, random_date)

getData()
```