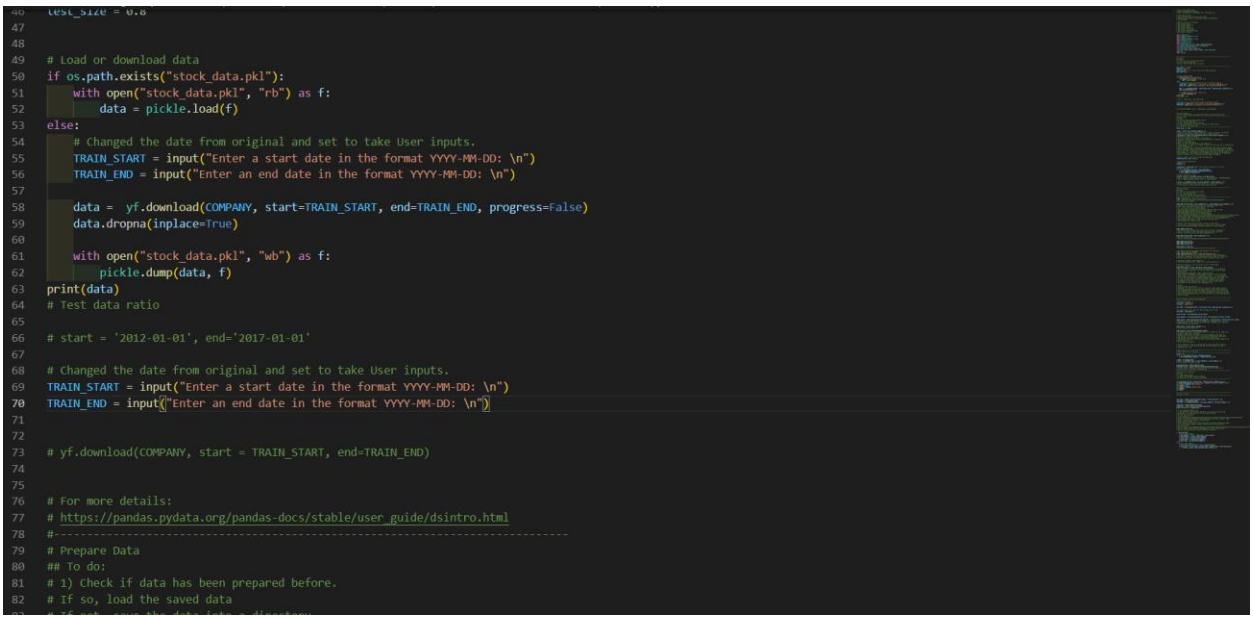
COS30018 - Option B - Task 2: Data processing 1

This tasks for this week are as follows :

a. This function will allow you to specify the start date and the end date for the whole dataset as inputs



```
test_size = 0.8


# Load or download data
if os.path.exists("stock_data.pkl"):
    with open("stock_data.pkl", "rb") as f:
        data = pickle.load(f)
else:
    # Changed the date from original and set to take User inputs.
    TRAIN_START = input("Enter a start date in the format YYYY-MM-DD: \n")
    TRAIN_END = input("Enter an end date in the format YYYY-MM-DD: \n")

    data =  yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END, progress=False)
    data.dropna(inplace=True)

    with open("stock_data.pkl", "wb") as f:
        pickle.dump(data, f)
print(data)
# Test data ratio

# start = '2012-01-01', end='2017-01-01'

# Changed the date from original and set to take User inputs.
TRAIN_START = input("Enter a start date in the format YYYY-MM-DD: \n")
TRAIN_END = input("Enter an end date in the format YYYY-MM-DD: \n")


# yf.download(COMPANY, start = TRAIN_START, end=TRAIN_END)


# For more details:
# https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html
#------------------------------------------------------------------------------
# Prepare Data
## To do:
# 1) Check if data has been prepared before.
# If so, load the saved data
```

In this function I have changed the values of the “Train_Start” and “Train_End” variables to take user inputs instead of hard coding the dates and specified the format as YYYY-MM-DD. This function checks the date entered and then downloads the data based on the dates specidied and then the training and testing is done.

b. This function will allow you to deal with the NaN issue in the data

This was present in the example code that was given (P1) I just had to add a single line that is data.dropna(inplace = true), which deals wit the issues for any null values or NaN.

c. This function will also allow you to use different methods to split the data into train/test data; e.g. you can split it according to some specified ratio of train/test and you can specify to split it by date or randomly.

```
# Can you find the reason?
# 2. The code base at
# https://github.com/x4nth055/pythoncode-tutorials/tree/master/machine-learning/stock-prediction
# gives a much better prediction. Even though on the surface, it didn't seem
# to be a big difference (both use Stacked LSTM)
# Again, can you explain it?
# A more advanced and quite different technique use CNN to analyse the images
# of the stock price changes to detect some patterns with the trend of
# the stock price:
# https://github.com/jason887/Using-Deep-Learning-Neural-Networks-and-Candlestick-Chart-Representation-to-Predict-Stock-Market
# Can you combine these different techniques for a better prediction??
# Add the following section for data splitting flexibility

if split_by_date:
    # Split data by date
    train_samples = int((1 - test_size) * len(x_train))
    x_train_final = x_train[:train_samples]
    y_train_final = y_train[:train_samples]
    x_test_final = x_train[train_samples:]
    y_test_final = y_train[train_samples:]
else:
    # Split data randomly
    from sklearn.model_selection import train_test_split
    x_train_final, x_test_final, y_train_final, y_test_final = train_test_split(
        x_train, y_train, test_size=test_size, shuffle=True
    )
```

This was a tricky function to do at first, however in this function , I imported a library train_test_split from sklearn. This function by default is set to true which will split the data by Time. If it is set to false then it will split the data randomly as the if else condition is made by me. The “ratio” or the “test_size” is also set to 0.8 by default so that it takes only 80 percent of data for training and the rest is used for testing purposes.

```
#------------------------------------------------------------------------------
DATA_SOURCE = "yahoo"
COMPANY = "TSLA"
split_by_date = True  # Set to False for random splitting
test_size = 0.8


# Load or download data
```

This was also present in the example code that was given to us, with some simple modifications it runs as required .

d. This function will have the option to allow you to store the downloaded data on your local machine for future uses and to load the data locally to save time.

```
47
48
49  # Load or download data
50  if os.path.exists("stock_data.pkl"):
51      with open("stock_data.pkl", "rb") as f:
52          data = pickle.load(f)
53  else:
```

This function was made by me importing another library called pickle and import os. I have made this function so that it saves the last configuration and then loads it automatically the next time we want to use it. In the above example we can see that the data is saved in a file name “stock_data.pkl” and there is an if condition applied with it . If the path with tha file name exists it will load the last saved file and start from there to save time.

```
52          data = pickle.load(f)
53  else:
54      # Changed the date from original and set to take User inputs.
55      TRAIN_START = input("Enter a start date in the format YYYY-MM-DD: \n")
56      TRAIN_END = input("Enter an end date in the format YYYY-MM-DD: \n")
57
58      data =  yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END, progress=False)
59      data.dropna(inplace=True)
60
61      with open("stock_data.pkl", "wb") as f:
62          pickle.dump(data, f)
63  print(data)
64  # Test data ratio
```

If there is no such file in the directory it will simply ask for the user input for dates and run the program from the beginning.

This idea was taken from the internet and the explanation is provided in: https://www.askpython.com/python/examples/save-data-in-python .

e. This function will also allow you to have an option to scale your feature columns and store the scalers in a data structure to allow future access to these scalers

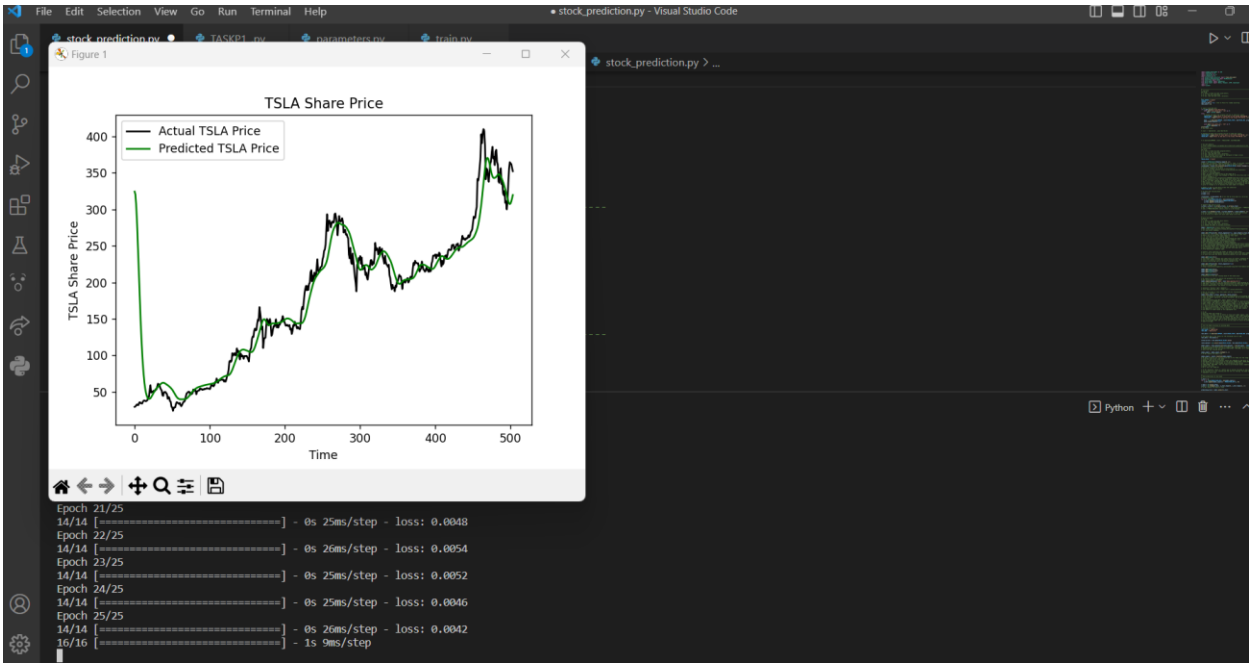
At last , in this function , I have imported another library called as joblib .

```
# feature_range (min,max) then you'll need to specify it here
scaled_data = scaler.fit_transform(data[PRICE_VALUE].values.reshape(-1, 1))
scaler_filename = "scaler.save"
joblib.dump = (scaler, scaler_filename)
# To Load it: scaler = joblib.load(scaler_filename)
```

This provides just two lines of code that saves our scaler function into a file named “scaler.save”

To load this, I have also added the comment which loads the saved data.

Additional Observations:



This is the final output with the current version of code. I noticed that the curve is not as accurate as previous version because I set the user input for only two years of data, where as in the previous task it was about five years. Therefore, using the functionality of the “ratio”, only 80% data was used for the training and 20% was used for testing. Hope this assignment demonstrates my clear understanding of the projects and it also fulfills the requirements that were given in this task. :)