

# Stock Price Prediction System

(Option 3)

## Weekly Report (Tasks C.2 -Data Processing 1)

COS30018 Intelligent Systems

Class: Thursday 2:30pm to 4:30pm (2 hours)

Name: Suyogya Raj Joshi

Student ID: 105098233

## Contents

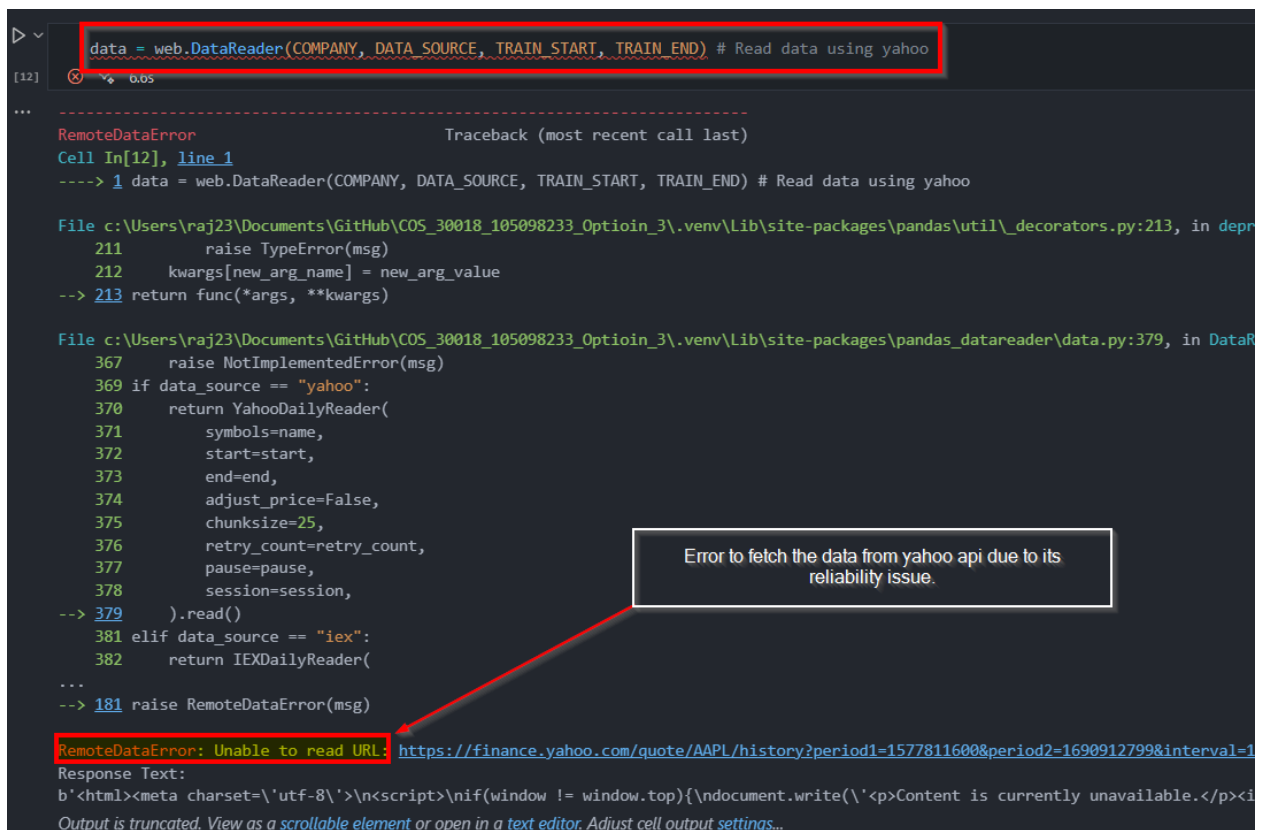
Weekly Report (Tasks C.2 -Data Processing 1).....	1
Check List of given weekly tasks: .....	3
Initial Testing with the given code base.....	3
Defining a function for loading and processing the data. ....	4
Summary of the function along with output. ....	6
Observed output .....	8

## Check List of given weekly tasks:

No.	Weekly Tasks	Progress
1.	Writing a function that load and processes a dataset with multiple features	Complete
2.	Summary of Implemented function and features selected for processing	Complete

## Initial Testing with the given code base.

In the given starter code base, there were a lot of comments with instructions on various ways of implementing the techniques required for the second assessment. The first instruction was to fetch the data for the stock using yahoo with **web.Datareader** . However, this method of fetching the data had some reliability issue with the yahoo API. Hence, decision to fetch the data using yahoo finance was made.



```
data = web.DataReader(COMPANY, DATA_SOURCE, TRAIN_START, TRAIN_END) # Read data using yahoo

...
RemoteDataError                                Traceback (most recent call last)
Cell In[12], line 1
----> 1 data = web.DataReader(COMPANY, DATA_SOURCE, TRAIN_START, TRAIN_END) # Read data using yahoo

File c:\Users\raj23\Documents\GitHub\COS_30018_105098233_Option_3\.venv\Lib\site-packages\pandas\util\decorators.py:213, in deprecate
    211         raise TypeError(msg)
    212     kwargs[new_arg_name] = new_arg_value
--> 213 return func(*args, **kwargs)

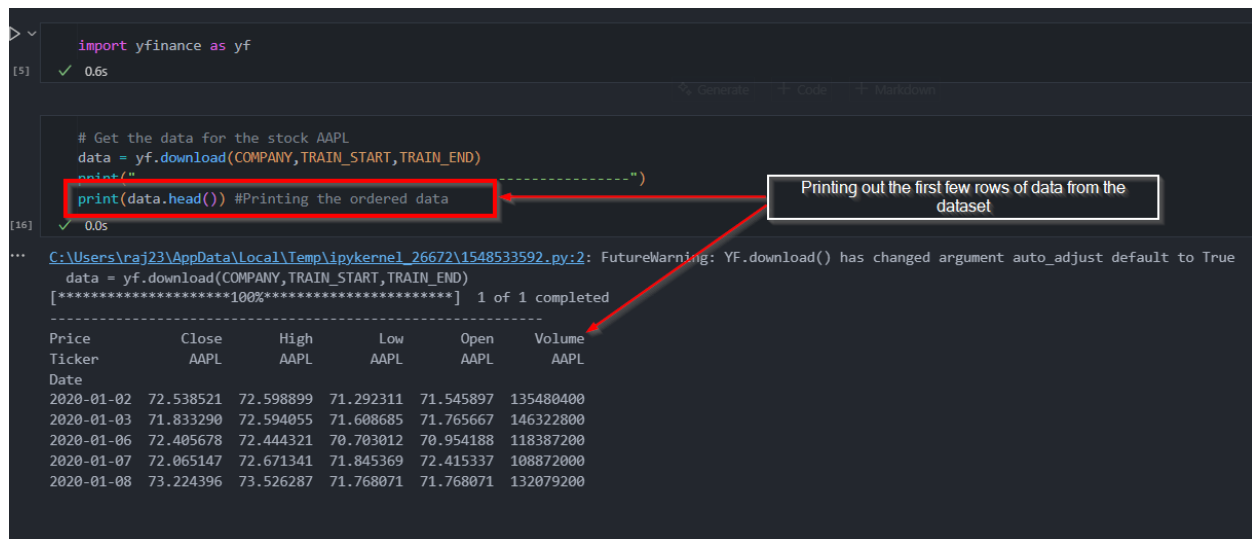
File c:\Users\raj23\Documents\GitHub\COS_30018_105098233_Option_3\.venv\Lib\site-packages\pandas_datareader\data.py:379, in DataReader
    367     raise NotImplementedError(msg)
    369 if data_source == "yahoo":
    370     return YahooDailyReader(
    371         symbols=name,
    372         start=start,
    373         end=end,
    374         adjust_price=False,
    375         chunksize=25,
    376         retry_count=retry_count,
    377         pause=pause,
    378         session=session,
--> 379     ).read()
    381 elif data_source == "iex":
    382     return IEXDailyReader(
...
--> 181 raise RemoteDataError(msg)

RemoteDataError: Unable to read URL: https://finance.yahoo.com/quote/AAPL/history?period1=1577811600&period2=1690912799&interval=1
Response Text:
b'<html><meta charset=\'utf-8\'>\n<script>\nif(window != window.top){\ndocument.write(\'<p>Content is currently unavailable.</p></i>
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Fig: Reliability issue with the yahoo API.

## Defining a function for loading and processing the data.

Before defining the function, Dataset was loaded from yahoo finance to see the different attributes of the dataset which will help to define function and how to process the dataset.



```
> <
import yfinance as yf
[5] ✓ 0.6s

# Get the data for the stock AAPL
data = yf.download(COMPANY, TRAIN_START, TRAIN_END)
print("")
print(data.head()) #Printing the ordered data
[16] ✓ 0.05s

C:\Users\raj23\AppData\Local\Temp\ipykernel_26672\1548533592.py:2: FutureWarning: YF.download() has changed argument auto_adjust default to True
data = yf.download(COMPANY, TRAIN_START, TRAIN_END)
[*****100%*****] 1 of 1 completed

Price      Close      High      Low      Open      Volume
Ticker      AAPL      AAPL      AAPL      AAPL      AAPL
Date
2020-01-02  72.538521  72.598899  71.292311  71.545897  135480400
2020-01-03  71.833290  72.594055  71.608685  71.765667  146322800
2020-01-06  72.405678  72.444321  70.703012  70.954188  118387200
2020-01-07  72.065147  72.671341  71.845369  72.415337  108872000
2020-01-08  73.224396  73.526287  71.768071  71.768071  132079200
```

Fig: Printing out the data using head () function from pandas' library.

Here dataset had 6 attributes:

- Close
- High
- Low
- Open
- Volume

Now using this information, function was defined which loads and processes the dataset:

```

# Defining a function that loads the data
def load_data(COMpany: str, TRAIN_START: str, TRAIN_END: str, test_size: float = 0.2, data_dir: str = "data"):
    # Ensuring data directory exists
    os.makedirs(data_dir, exist_ok=True)
    DATA_PATH = os.path.join(data_dir, f"{COMpany}_{TRAIN_START}_{TRAIN_END}.csv")

    # Loading data if present, else downloading and saving
    if os.path.exists(DATA_PATH):
        data = pd.read_csv(DATA_PATH, index_col=0, parse_dates=True)
    else:
        data = yf.download(COMpany, start=TRAIN_START, end=TRAIN_END)
        data.to_csv(DATA_PATH)

    print("-----")
    print("Initial output with the NaN")
    print(data.head(10)) # First 10 rows
    print("-----")
    print(data.isna().sum()) # Checking NaN Or missing values.
    print("=====")

    # Removing NaN values
    data = data.dropna()

    # Quick check
    print("-----")
    print("Final output without the NaN")
    print(data.isna().sum()) # Checking the NaN after removing it

    # Ensure columns are numeric before calculation
    for col in ["Open", "High", "Low", "Close"]:
        data[col] = pd.to_numeric(data[col], errors='coerce')

    # Add OHLC average column and set PRICE_VALUE
    data["OHLC_Avg"] = (data["Open"] + data["High"] + data["Low"] + data["Close"]) / 4
    PRICE_VALUE = "OHLC_Avg"

    # Scaling the data
    # Choosing the price column (i.e OHLC_Avg) and reshape it into 2D
    values = data[PRICE_VALUE].values.reshape(-1, 1)
    # Creating scaler
    scaler = MinMaxScaler(feature_range=(0, 1))
    # Fitting and transforming the scaler.
    scaled_data = scaler.fit_transform(values)

    # Splitting the dataset into training and testing set.
    train, test = train_test_split(scaled_data, test_size=test_size, shuffle=False)

    return data, train, test, scaler, PRICE_VALUE, scaled_data

```

✓ 0.0s

Fig: Defining the function that loads and processes data.

The function allows for the reusability of code as one can easily tune the data processing and loading by changing few variables.

For the above function, the below given input as shown in the image:

```

TRAIN_START = '2020-01-01' # Start date to read
TRAIN_END = '2023-08-01' # End date to read
COMpany = "AAPL" #Ticker for apple

# Calling the function to load the dataset and save the return value to specific variable
data, train, test, scaler, PRICE_VALUE, scaled_data = load_data(
    COMpany, TRAIN_START, TRAIN_END, test_size=0.2
)

print("Dataset shape:", data.shape)
print("Train shape:", train.shape)
print("Test shape:", test.shape)
print("Using PRICE_VALUE:", PRICE_VALUE)

```

✓ 0.0s

## Summary of the function along with output.

The function accepts five parameters: ticker, start\_date, end\_date, test\_size, and folder name. Its purpose is to collect (or load) historical prices, clean and convert them.

```
# Defining a function that loads the data
def load_data(COMPANY: str, TRAIN_START: str, TRAIN_END: str, test_size: float = 0.2, data_dir: str = "data"):
```

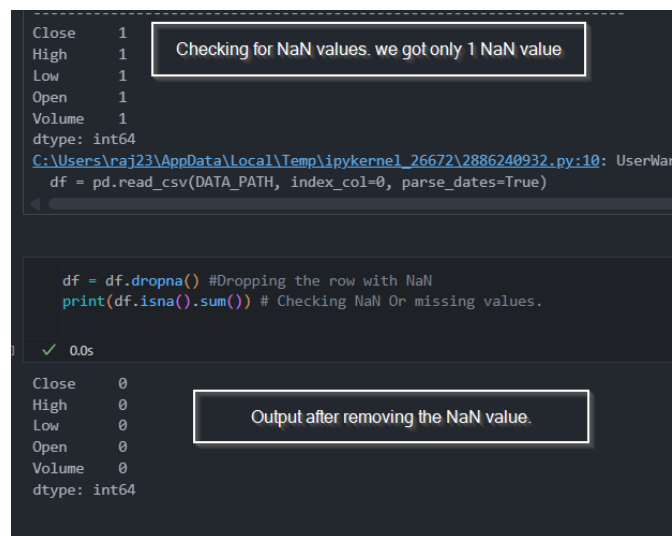
The OS module ensures the setup of a data directory. A DATA\_PATH variable stores this location, ensuring that datasets are properly stored and reused in future.

```
# Ensuring data directory exists
os.makedirs(data_dir, exist_ok=True)
DATA_PATH = os.path.join(data_dir, f"{COMPANY}_{TRAIN_START}_{TRAIN_END}.csv")
```

After this process, the function had an if else loop which checked If the CSV already exists, it is loaded from disc for faster results. Otherwise, historical price data is collected from Yahoo Finance, saved to CSV at DATA\_PATH, and then imported into a DataFrame named data.

```
# Loading data if present, else downloading and saving
if os.path.exists(DATA_PATH):
    data = pd.read_csv(DATA_PATH, index_col=0, parse_dates=True)
else:
    data = yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END)
    data.to_csv(DATA_PATH)
```

Missing values are determined using **data.isna().sum()**. Because only one row was missing in this experiment, the technique row dropping (**data = data.dropna()**) was used. Dropping avoids adding false patterns into the target series.



```
Close    1
High     1
Low      1
Open     1
Volume   1
dtype: int64
C:\Users\raj23\AppData\Local\Temp\ipykernel_26672\2886240932.py:10: UserWarning
df = pd.read_csv(DATA_PATH, index_col=0, parse_dates=True)

df = df.dropna() #Dropping the row with NaN
print(df.isna().sum()) # Checking NaN Or missing values.
✓ 0.0s
Close    0
High     0
Low      0
Open     0
Volume   0
dtype: int64
```

Checking for NaN values. we got only 1 NaN value

Output after removing the NaN value.

Instead of capturing a single attribute like the Close, the function calculates the day's entire movement:

$$OHLC \text{ avg} = \frac{Open + High + Low + Close}{4}$$

The target selection variable is set, and this new column is added to the data: "OHLC\_Avg" is PRICE\_VALUE.

```
# Ensure columns are numeric before calculation
for col in ["Open", "High", "Low", "Close"]:
    data[col] = pd.to_numeric(data[col], errors='coerce')

# Add OHLC average column and set PRICE_VALUE
data["OHLC_Avg"] = (data["Open"] + data["High"] + data["Low"] + data["Close"]) / 4
PRICE_VALUE = "OHLC_Avg"
```

The selected series is extracted as **data[PRICE\_VALUE]** and reshaped into a 2D array using **reshape(-1, 1)**. This shape is required by scikit-learn transformers and numerous deep neural network layers.

A **MinMaxScaler** is applied to the target series and transformed into the [0, 1] range, resulting in scaled\_data. Scaling maintains training and usually improves consistency in LSTM-style models.

```
# Scaling the data
# Choosing the price column (i.e OHLC_Avg) and reshape it into 2D
values = data[PRICE_VALUE].values.reshape(-1, 1)
# Creating scaler
scaler = MinMaxScaler(feature_range=(0, 1))
# Fitting and transforming the scaler.
scaled_data = scaler.fit_transform(values)
```

Using given test\_size (0.2), the series is divided sequentially into train (first 80%) and test (last 20%). This ensures time order while preventing forecasting bias.

```
# Splitting the dataset into training and testing set.
train, test = train_test_split(scaled_data, test_size=test_size, shuffle=False)

return data, train, test, scaler, PRICE_VALUE, scaled_data
```

Here the final output looks like this:

```

Initial output with the NaN
Price
Close
High
Low
\
Ticker
AAPL
AAPL
AAPL
Date
NaN
NaN
NaN
2020-01-02 72.53852081298828 72.59889913722518 71.29231138597861
2020-01-03 71.833299388889766 72.59485542623857 71.68868452797844
2020-01-06 72.48567779541816 72.44432880433776 70.7838121336898
2020-01-07 72.96518739998234 72.67138688922641 71.84336384211234
2020-01-08 73.1248973595312 73.526872893597 73.7688712964865
2020-01-09 74.77975463867188 74.97296238565743 73.9513658951312
2020-01-10 74.34879913338078 75.51394659834878 74.44645484641248
2020-01-13 76.35883356933594 76.5766838580251 75.14684225138329

Open
Volume
Price
AAPL
AAPL
Date
NaN
NaN
2020-01-02 71.5458973217052 135480480
2020-01-03 71.76566667931918 146322800
2020-01-06 70.95418888531902 118387200
2020-01-07 72.41513721548589 189872000
2020-01-08 71.76887129858055 132879200
2020-01-09 74.28253438813242 170188400
...
Dataset shape: (901, 6)
Train shape: (720, 1)
Test shape: (181, 1)
Using PRICE_VALUE: OHLC_Avg
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
C:\Users\raj23\AppData\Local\Temp\ipykernel_26672\608134679.py:9: UserWarning: Could not infer format, so each element will be parsed individually, falling back to 'datetutil'. To ensure parsing is consistent and as-expected, please specify a format.
data = pd.read_csv(DATA_PATH, index_col=0, parse_dates=True)

```

## Observed output

- Dataset shape: (901, 6)
- Train shape: (720, 1)
- Test shape: (181, 1)
- Using PRICE\_VALUE: OHLC\_Avg

```

***
Dataset shape: (901, 6)
Train shape: (720, 1)
Test shape: (181, 1)
Using PRICE_VALUE: OHLC_Avg

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
C:\Users\raj23\AppData\Local\Temp\ipykernel_26672\608134679.py:9: UserWarning: Co
data = pd.read_csv(DATA_PATH, index_col=0, parse_dates=True)

```