HW1

February 5, 2022

1 1. Setup and Data Retrieval

[1]:
```python
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_ta as ta
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

[2]:
```python
msci = yf.download("XWD.TO", start="2010-01-01")
msci
```

```
[*********************100%***********************]  1 of 1 completed
```

[2]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2010-01-04	22.665331	22.945892	22.665331	22.935871	18.619492	4491
2010-01-05	22.905811	22.945892	22.865730	22.945892	18.627623	6587
2010-01-06	22.865730	22.905811	22.775551	22.845692	18.546282	7086
2010-01-07	22.775551	22.905811	22.755510	22.905811	18.595087	39521
2010-01-08	22.875751	22.945892	22.835670	22.945892	18.627623	45209
…	…	…	…	…	…	…
2022-01-31	69.410004	70.080002	69.349998	70.080002	70.080002	6000
2022-02-01	70.510002	70.750000	70.160004	70.750000	70.750000	6000
2022-02-02	71.160004	71.320000	71.010002	71.290001	71.290001	4000
2022-02-03	70.959999	70.959999	69.879997	69.910004	69.910004	11700
2022-02-04	70.430000	71.010002	70.139999	70.559998	70.559998	7300

[3035 rows x 6 columns]

2 2. Data Analysis and Exploration in Pandas

2.1 2.1. Plot the Adj Close column of the MSCI ETF

[3]:
```
msci["Adj Close"].plot()
plt.ylabel("Adj Close")
plt.show()
```

2.2 2.2. Gather information about the data frame by looking at key statistics, i.e. the moments, data types, statistics...

The first moment (mean), the second central moment (Variance), standard deviation, median, data types are collected in the data frame shown below.

[4]:
```
msci_info = pd.concat([msci.describe(),
                       pd.DataFrame(msci.dtypes, columns=["data types"]).T,
                       pd.DataFrame(np.var(msci), columns=["variance"]).T,
                       pd.DataFrame(msci.median(), columns=["median"]).T], axis␣
 ↪= 0)

msci_info
```

[4]:

	Open	High	Low	Close	Adj Close	\
count	3035.0	3035.0	3035.0	3035.0	3035.0	
mean	40.506974	40.630312	40.3604	40.501604	37.462726	
std	14.237339	14.289796	14.175661	14.236313	15.293977	

```
min          20.440882   20.440882   19.539078   20.340681  16.512701
25%          24.974951   25.020041    24.92485    24.98497  21.478388
50%          40.480961   40.581161   40.330662   40.450901    36.3647
75%          50.901804   51.102203   50.761524   50.951904  48.560871
max          75.320641   75.430862   75.240479   75.430862  74.924858
data types     float64     float64     float64     float64    float64
variance    202.635034  204.130999  200.883146  202.605831  233.82865
median       40.480961   40.581161   40.330662   40.450901    36.3647

                       Volume
count                  3035.0
mean             22681.648105
std              52084.230134
min                       0.0
25%                    4092.0
50%                    8583.0
75%                   20459.0
max                  983529.0
data types              int64
variance    2711873200.975342
median                 8583.0
```

2.3 2.3. Get all rows where the "High" value is > 40

[5]:
```python
msci = msci[msci["High"] > 40]
msci
```

[5]:
```
                 Open       High        Low      Close  Adj Close  Volume
Date
2015-02-13  39.939880  40.020039  39.859718  40.010021  35.659203   30539
2015-02-18  40.060120  40.110222  39.979961  40.030060  35.677052   10479
2015-02-19  40.340679  40.420841  40.260521  40.260521  35.882462   23253
2015-02-20  40.270542  40.751503  40.210423  40.721443  36.293255   27545
2015-02-23  40.731464  40.781563  40.661324  40.741482  36.311115   37325
…                 …          …          …         …          …       …
2022-01-31  69.410004  70.080002  69.349998  70.080002  70.080002    6000
2022-02-01  70.510002  70.750000  70.160004  70.750000  70.750000    6000
2022-02-02  71.160004  71.320000  71.010002  71.290001  71.290001    4000
2022-02-03  70.959999  70.959999  69.879997  69.910004  69.910004   11700
2022-02-04  70.430000  71.010002  70.139999  70.559998  70.559998    7300

[1588 rows x 6 columns]
```

2.4 2.4. Drop the volume column

[6]:
```
del msci["Volume"]
msci
```

[6]:
```
                 Open       High        Low      Close  Adj Close
Date
2015-02-13  39.939880  40.020039  39.859718  40.010021  35.659203
2015-02-18  40.060120  40.110222  39.979961  40.030060  35.677052
2015-02-19  40.340679  40.420841  40.260521  40.260521  35.882462
2015-02-20  40.270542  40.751503  40.210423  40.721443  36.293255
2015-02-23  40.731464  40.781563  40.661324  40.741482  36.311115
…                 …          …          …          …          …
2022-01-31  69.410004  70.080002  69.349998  70.080002  70.080002
2022-02-01  70.510002  70.750000  70.160004  70.750000  70.750000
2022-02-02  71.160004  71.320000  71.010002  71.290001  71.290001
2022-02-03  70.959999  70.959999  69.879997  69.910004  69.910004
2022-02-04  70.430000  71.010002  70.139999  70.559998  70.559998

[1588 rows x 5 columns]
```

2.5 2.5. Get the adjusted close price of the 01/17/2022

[7]:
```
msci.loc[pd.to_datetime("2022-01-17"), "Adj Close"]
```

[7]:
```
72.19999694824219
```

2.6 2.6. Create a new column that stores the daily return

[8]:
```
msci = msci.copy()
msci["daily return"] = (msci["Adj Close"] - msci["Open"]) / msci["Open"]
msci
```

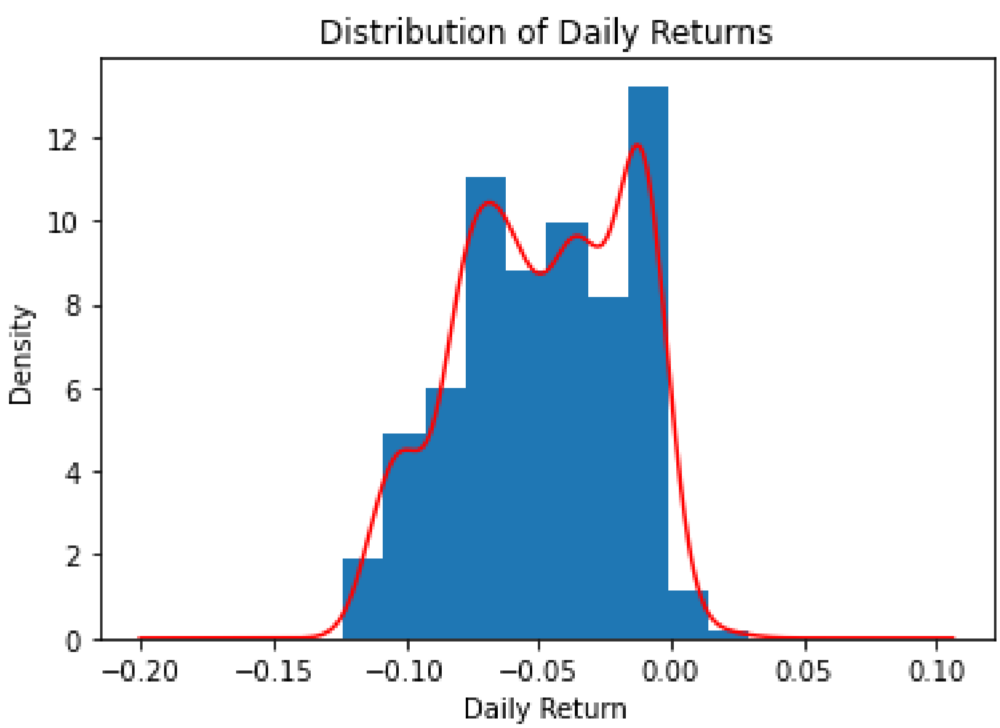
[8]:
```
                 Open       High        Low      Close  Adj Close  \
Date
2015-02-13  39.939880  40.020039  39.859718  40.010021  35.659203
2015-02-18  40.060120  40.110222  39.979961  40.030060  35.677052
2015-02-19  40.340679  40.420841  40.260521  40.260521  35.882462
2015-02-20  40.270542  40.751503  40.210423  40.721443  36.293255
2015-02-23  40.731464  40.781563  40.661324  40.741482  36.311115
…                 …          …          …          …          …
2022-01-31  69.410004  70.080002  69.349998  70.080002  70.080002
2022-02-01  70.510002  70.750000  70.160004  70.750000  70.750000
2022-02-02  71.160004  71.320000  71.010002  71.290001  71.290001
2022-02-03  70.959999  70.959999  69.879997  69.910004  69.910004
2022-02-04  70.430000  71.010002  70.139999  70.559998  70.559998
```

```
              daily return
Date
2015-02-13       -0.107178
2015-02-18       -0.109412
2015-02-19       -0.110514
2015-02-20       -0.098764
2015-02-23       -0.108524
…                        …
2022-01-31        0.009653
2022-02-01        0.003404
2022-02-02        0.001827
2022-02-03       -0.014797
2022-02-04        0.001846

[1588 rows x 6 columns]
```

2.7 2.7. How are the returns distributed?

From the histogram below, we can observe an approximate bell-shaped distribution whose mean is round -0.05.

[9]:
```python
plt.hist(x = msci["daily return"], density = True)
plt.xlabel("Daily Return")
plt.title("Distribution of Daily Returns")
msci["daily return"].plot.density(color = "red")
plt.show()
```



[10]:
```
# Check if the mean is around -0.05.
np.nanmean(msci["daily return"])
```

[10]:
```
-0.04826754326367532
```

2.8 2.8. Calculate the cumulative return (Hint: cumsum())

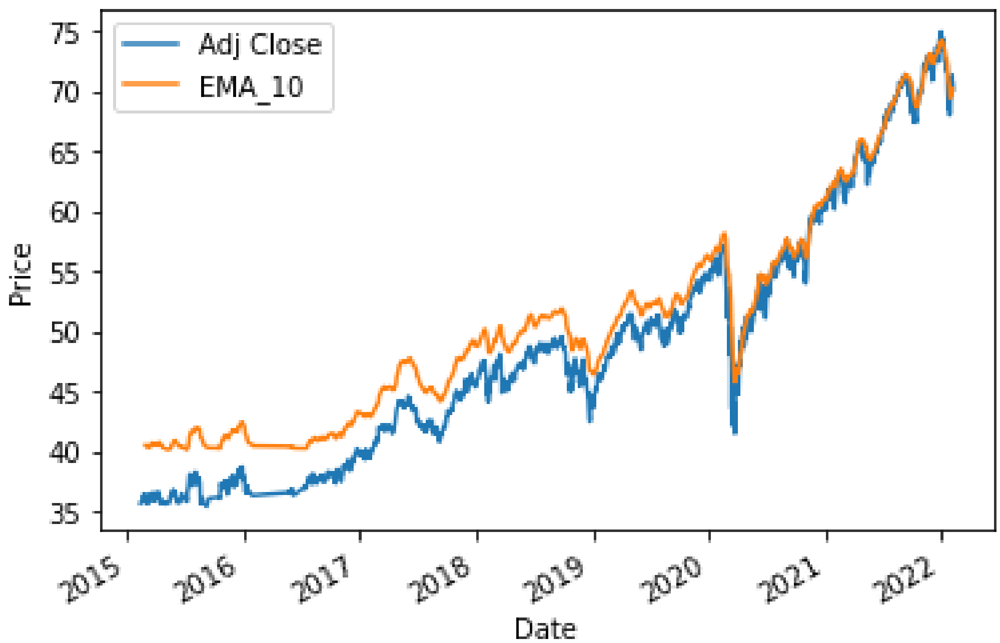
[11]:
```
msci["daily return"].cumsum()
```

[11]:
```
Date
2015-02-13    -0.107178
2015-02-18    -0.216590
2015-02-19    -0.327104
2015-02-20    -0.425869
2015-02-23    -0.534393
                 …
2022-01-31   -76.641138
2022-02-01   -76.637734
2022-02-02   -76.635907
2022-02-03   -76.650704
2022-02-04   -76.648859
Name: daily return, Length: 1588, dtype: float64
```

2.9 2.9. Add a 10-day exponential moving-average "EMA 10" column to the MSCI data frame based on the "Adj Close" column (Hint: you can use pandas_ta using pip install pandas_ta)

[12]:
```
msci["EMA_10"] = msci.ta.ema(signal = 10)
```

[13]:
```
msci["Adj Close"].plot(), msci.ta.ema(window = 10).plot()
plt.ylabel("Price")
plt.legend()
plt.show()
```



3 3. Linear Regression using Scikit-Learn

3.1 3.1 Handle Missing Values

Adding the EMA has created missing values for the first n indexes, where n is your window size (9, in our case).

3.1.1 3.1.1 Find and print out the number of missing values grouped by column.

[14]:
```
msci.isnull().sum()
```

[14]:
```
Open            0
High            0
Low             0
Close           0
Adj Close       0
daily return    0
EMA_10          9
dtype: int64
```

There are 9 missing values in the column "EMA_10" and no missing value exists in other columns.

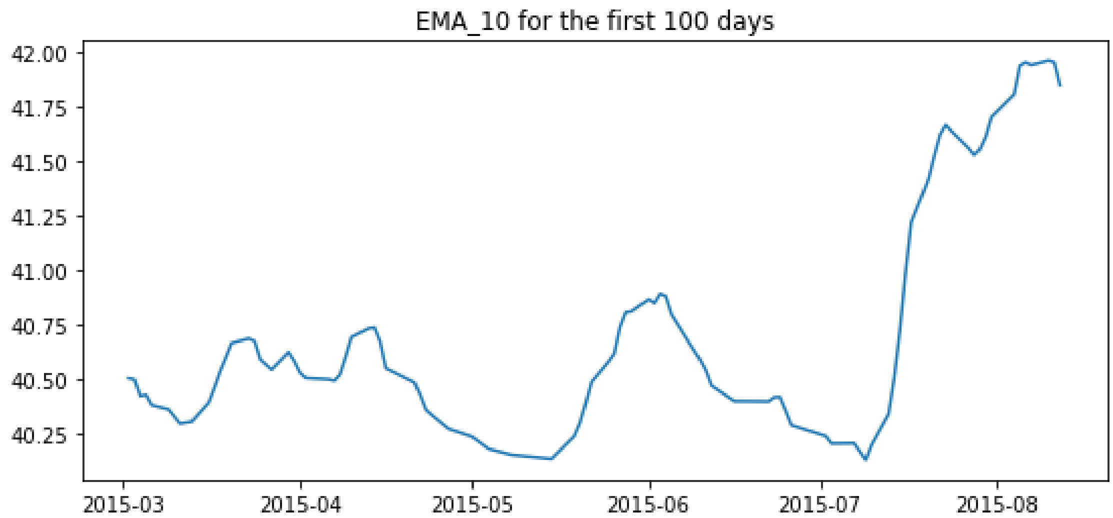
3.1.2 3.1.2 Replace missing values

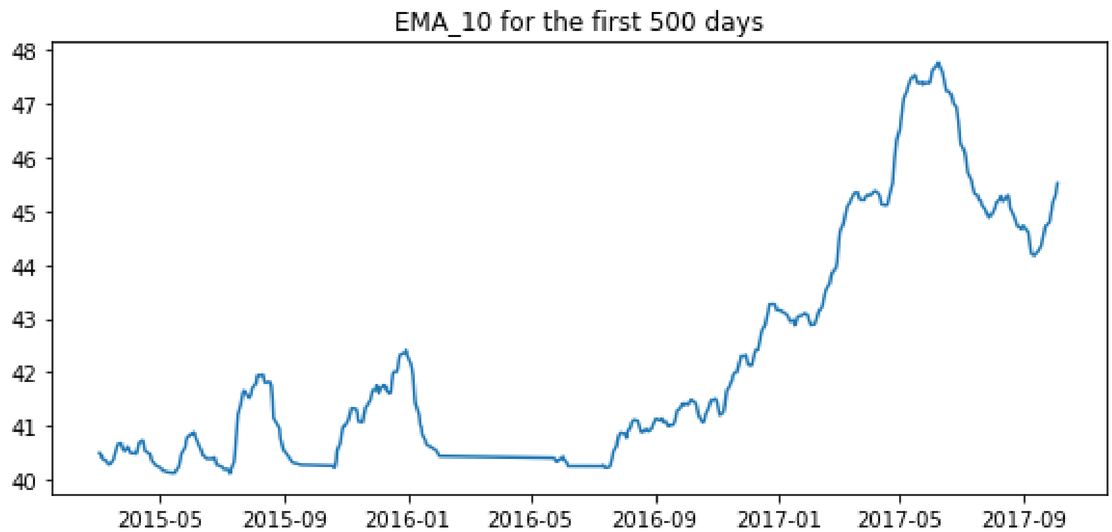
Often times, if statistically reasonable (esp. in time series), we can replace missing values with their mean or mode. This can especially be done if the missing values make up only a little percentage of the data set. Another approach would be to drop the missing entries, which will however often times decrease our performance. Try filling the missing rows of the explanatory variable with "reasonable" values (Hint: Look at the distribution of the beginning of the EMA 10 time series). Make sure there are no missing values left.

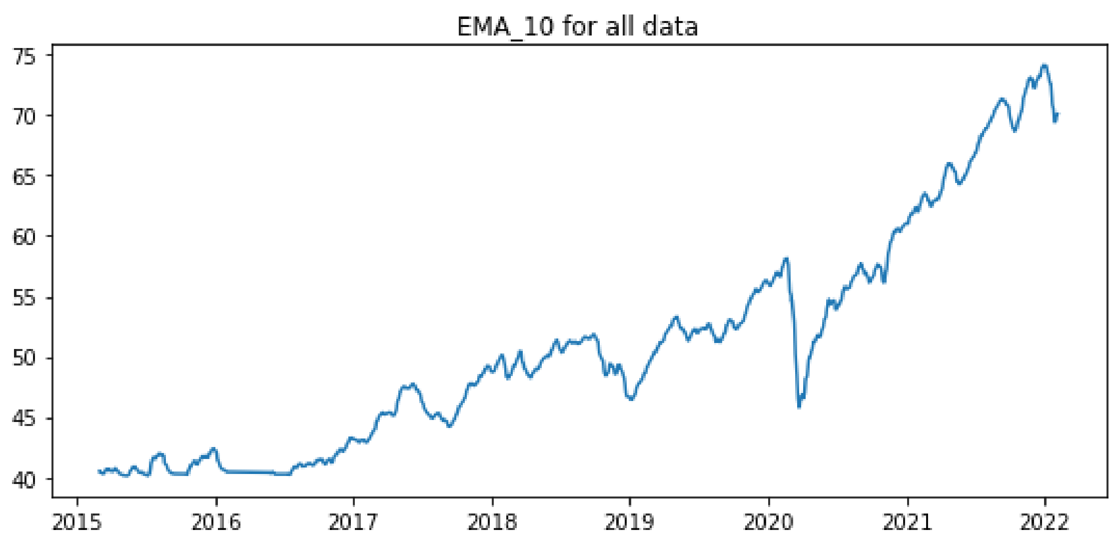
[15]:
```python
# Get the number of rows in our data frame.
msci.shape[0]
```

[15]: 1588

[16]:
```python
plt.figure()
plt.plot(msci["EMA_10"][0:100])
plt.subplots_adjust(right=1.3)
plt.title("EMA_10 for the first 100 days")

plt.figure()
plt.plot(msci["EMA_10"][0:500])
plt.subplots_adjust(right=1.3)
plt.title("EMA_10 for the first 500 days")

plt.figure()
plt.plot(msci["EMA_10"])
plt.subplots_adjust(right=1.3)
plt.title("EMA_10 for all data")

plt.show()
```







From the above we can see that, EMA_10 just slightly changed from 2015-03 to 2015-04, compared to both the first 500-days variation of EMA_10 and certainly, the total variation of EMA_10 over the time scale (from 2015 to 2022). Besides, EMA_10 is a weighted average statistic for the past 10 days, hence, the short-term data reflects more information than long-term data, so it is reasonable to replace the first 9 missing values by the average EMA_10 from 2015-03 to 2015-04.

[17]:
```
msci["EMA_10"].fillna(np.nanmean(msci[msci.index < pd.
 ↪to_datetime("2015-04-01")]["EMA_10"]), inplace = True)
msci["EMA_10"]
```

[17]:
```
Date
2015-02-13    40.491352
2015-02-18    40.491352
2015-02-19    40.491352
2015-02-20    40.491352
2015-02-23    40.491352
                 …
2022-01-31    69.548786
2022-02-01    69.767189
2022-02-02    70.044064
2022-02-03    70.019689
2022-02-04    70.117927
Name: EMA_10, Length: 1588, dtype: float64
```

[18]:
```
# Check if there is any missign value.
msci.isnull().sum()
```

[18]:
```
Open            0
High            0
Low             0
Close           0
Adj Close       0
daily return    0
EMA_10          0
dtype: int64
```

3.2 3.2 Write the function split_data

The function split_data takes explanatory and response time series as parameter to and splits it into X_train, X_test, y_train and y_test. In the context of linear regression, y denotes the response variable (=label) and X (=features) the explanatory variable. Other than the series, it takes the size of your train split as parameter.

Also, be sure to add docstring documentation as learnt in the Python Refresher. Furthermore, print out useful information in the function, like the shape of the initial time series as well as the subsets created.

For example: the first 80% of the time series as train, the last 20% as test.

[19]:
```
def split_data(X: pd.Series, y: pd.Series, size: float) -> tuple:
    """
    Input: X: pd.Series, y: pd.Series, size: float
    X: The explanatory variable.
    y: The response variable.
    size: The proportion of training data in the time series,

    Output: A tuple with length of 4.
    tuple[0]: X_train
```

```
    tuple[1]: X_test
    tuple[2]: y_train
    tuple[3]: y_test

    Notice: X and y should have identical length and time scale.
    """
    if len(X) != len(y):
        raise Exception("The inputs X and y should have identical length!")
    elif sum(~(X.index == y.index)) > 0:
        raise Exception("The inputs X and y should have identical time scale!")

    time_quantile = np.quantile(X.index, q = size)
    X_train, X_test = X[X.index < time_quantile].values, X[X.index >=␣
↪time_quantile].values
    y_train, y_test = y[y.index < time_quantile].values, y[y.index >=␣
↪time_quantile].values

    print(f"The length of the initial time series is {len(X)}.")
    print(f"The first {size*100}% of the time series as train, the last␣
↪{round((1-size)*100)}% as test.")
    print(f"""
    X_train: {X_train}
    X_test: {X_test}
    y_train: {y_train}
    y_test: {y_test}
    """)
    return X_train, X_test, y_train, y_test
```

3.3 3.3 Run a linear regression model on the EMA and the Adj Close

3.3.1 3.3.1. Split the data into train and test using your custom function using 20% test size.

[20]:
```
print(split_data.__doc__)
```

```
    Input: X: pd.Series, y: pd.Series, size: float
    X: The explanatory variable.
    y: The response variable.
    size: The proportion of training data in the time series,

    Output: A tuple with length of 4.
    tuple[0]: X_train
    tuple[1]: X_test
    tuple[2]: y_train
    tuple[3]: y_test
```

Notice: X and y should have identical length and time scale.

```
[21]: X_train, X_test, y_train, y_test = split_data(X = msci["EMA_10"], y = msci["Adj␣
      ↪Close"], size = 0.8)
      print(split_data(X = msci["EMA_10"], y = msci["Adj Close"], size = 0.8))
```

```
The length of the initial time series is 1588.
The first 80.0% of the time series as train, the last 20% as test.

    X_train: [40.49135186 40.49135186 40.49135186 … 57.1533377  56.78368343
 56.60876714]
    X_test: [56.28529285 56.12629848 56.13649247 56.32519397 56.6198664
56.85913999
 57.22069533 57.56570283 57.95182607 58.19304984 58.55073538 58.90532954
 59.18269915 59.30943762 59.4386378  59.53706027 59.65038028 59.83783153
 59.98026989 60.10591986 60.25244776 60.23205392 60.33560801 60.42033408
 60.48054661 60.50977107 60.50817692 60.57610175 60.58795254 60.5138453
 60.45867752 60.36070738 60.36435335 60.42927877 60.53705447 60.63616544
 60.68992912 60.76488807 60.82439609 60.86397566 60.99291587 61.03282667
 61.06001553 61.03125014 61.00953616 61.07921761 61.25100493 61.48993645
 61.65081132 61.74600031 61.83481304 61.84189227 61.82582199 61.87097186
 61.94252791 62.04844122 62.13691891 62.24210297 62.35731179 62.40420611
 62.24946035 62.26130912 62.0177701  62.02437693 62.12269536 62.20860314
 62.38455774 62.5376294  62.73574305 62.8777957  62.97944637 63.09176425
 63.25653389 63.3949888  63.49916128 63.52245096 63.47774244 63.38104242
 63.30192422 63.23901319 63.04179412 62.88772053 62.9292684  62.88492349
 62.74661895 62.50775462 62.4908591  62.46610465 62.54969539 62.65270238
 62.75519912 62.80626688 62.86262451 62.91237867 62.96766155 62.97099097
 62.971893   63.03639473 63.03269263 62.98047416 63.02337603 63.14956838
 63.30747156 63.43302169 63.48473301 63.62359919 63.90118099 64.12282795
 64.37340458 64.59117541 64.78574779 64.96498372 65.14077975 65.249999
 65.46324353 65.67233159 65.81243245 65.88151434 65.9161742  65.89352074
 65.96972098 65.97376835 65.97343714 65.90757972 65.85369637 65.72762869
 65.67549253 65.5344581  65.45732424 65.36688761 65.39491566 65.26481453
 65.04177158 64.6734555  64.52149508 64.53562277 64.46884317 64.38141279
 64.28801716 64.28083168 64.30227975 64.38541461 64.51537502 64.58527104
 64.68253799 64.70017768 64.77655251 64.82993151 64.89364558 65.01318305
 65.12556136 65.25940927 65.33430594 65.40651587 65.54393535 65.68187523
 65.82935038 65.95001187 66.11431996 66.16677254 66.26252115 66.36636529
 66.39667448 66.44697866 66.49906747 66.57812218 66.6919921  66.77422756
 66.89616671 66.98500426 67.15242442 67.37138586 67.46491074 67.63434464
 67.82398251 67.98642891 68.15213213 68.27313258 68.30108247 68.25289871
 68.33189493 68.38377463 68.446262   68.61216325 68.7460779  68.82467464
 68.8762275  68.90747626 68.88021057 69.01093544 69.1014959  69.16465997
 69.28739014 69.40966881 69.47145565 69.54751429 69.65529018 69.76533213
 69.89544714 69.97457638 70.00470461 70.15323832 70.34035234 70.44061276
 70.49167284 70.55166687 70.61897189 70.71229758 70.79412038 70.88110665
```

```
 71.01057552 71.06549307 71.1140697  71.24672774 71.32429497 71.34585704
 71.29244687 71.27243206 71.22872921 71.22576485 71.25248791 71.20330177
 70.98998505 70.91018807 70.85400914 70.84448115 70.81482385 70.73043776
 70.45917264 70.34107145 70.01489414 69.84639937 69.4808123  69.29647083
 69.16750767 69.14033098 69.01242978 68.7966522  68.66565162 68.66049236
 68.77651179 68.89329847 69.04897083 69.20002354 69.34911788 69.482035
 69.68187618 69.85995752 69.92367774 70.068726   70.21837303 70.38635761
 70.58027509 70.78994636 71.11999418 71.39185465 71.62886085 71.780873
 71.8378388  72.00104513 72.18194503 72.31902319 72.49129769 72.64864581
 72.79378144 72.90888608 73.0067053  73.07216429 73.11296946 73.13178059
 72.96316716 72.93452052 72.74529701 72.48845473 72.4440978  72.28027696
 72.27741321 72.42446063 72.58120876 72.69852638 72.86009858 72.96861098
 73.00638234 73.21035969 73.26247502 73.26139147 73.19674123 73.33149261
 73.47089358 73.67603973 73.85857801 74.14444791 74.1563663  74.0352084
 74.09971586 73.99613093 73.84047065 73.59674872 73.39006696 73.21914514
 73.09748295 72.8415774  72.68492757 72.59675837 72.21552957 71.81088745
 71.37436312 70.86629743 70.49060754 70.03776981 69.67999342 69.41090316
 69.43073834 69.54878624 69.76718875 70.04406369 70.01968914 70.11792703]
    y_train: [35.65920258 35.67705154 35.88246155 … 55.52363586 54.26105881
 54.95153046]
    y_test: [53.9750061  54.54711151 55.30662918 56.28315353 57.04266739
57.03280258
 57.93041229 58.1967392  58.75897598 58.35455704 59.22257614 59.55794907
 59.48890305 58.94638824 59.0844841  59.04502869 59.22257614 59.73549652
 59.67631531 59.72563553 59.96236801 59.20285034 59.85386276 59.85386276
 59.80454254 59.69604111 59.55794907 59.93277359 59.69604111 59.24230957
 59.27190018 58.98584747 59.43957901 59.77495193 60.07087326 60.13005447
 59.98209381 60.14977646 60.13991165 60.09059525 60.61338043 60.45109558
 60.42140961 60.1443367  60.15423203 60.62921143 61.25262451 61.78697205
 61.59896088 61.40105438 61.46043015 61.10419464 60.98544312 61.30210114
 61.49011612 61.74739456 61.75728607 61.93540573 62.09373474 61.83645248
 60.78753662 61.53958893 60.16412735 61.28231049 61.78697205 61.81665802
 62.39059448 62.44007111 62.83588791 62.72703934 62.64787674 62.80620193
 63.20202255 63.22180939 63.17233276 62.83588791 62.48955154 62.16300201
 62.16300201 62.17289734 61.38126373 61.42084503 62.33122635 61.90572357
 61.35157776 60.66879272 61.63854599 61.57917023 62.14321136 62.33122635
 62.4301796  62.25205994 62.33122635 62.35101318 62.4301796  62.20258331
 62.19268799 62.53902435 62.23226929 61.96509171 62.4301796  62.92494965
 63.22180939 63.20202255 62.92494965 63.44940567 64.33999634 64.31030273
 64.68632507 64.7556076  64.8446579  64.95350647 65.11183167 64.92382812
 65.59671021 65.78472137 65.61650085 65.36911011 65.25036621 64.97329712
 65.48786163 65.17121124 65.15142059 64.79518127 64.79518127 64.34989166
 64.62696075 64.09261322 64.30041504 64.15198517 64.70612335 63.87491226
 63.24160385 62.23226929 63.04368973 63.79574203 63.37023926 63.19212341
 63.07338333 63.44940567 63.59783554 63.95407104 64.29051208 64.09261322
 64.31030273 63.97386169 64.31030273 64.26082611 64.36967468 64.73580933
 64.81497955 65.04257202 64.85454559 64.91392517 65.33943176 65.47795868
 65.66597748 65.66597748 66.0222168  65.57691956 65.8638916  66.00242615
```

```
 65.70555878 66.22608948 66.28581238 66.48486328 66.75359344 66.69387054
 66.99246216 66.93274689 67.45029449 67.89816284 67.43038177 67.93798065
 68.21665192 68.25646973 68.43562317 68.35599518 67.9678421  67.57967377
 68.22661591 68.15693665 68.26641846 68.89344788 68.88349152 68.71430206
 68.64463043 68.58490753 68.29627991 69.13231659 69.04273987 68.9830246
 69.37117767 69.49062347 69.28160858 69.42094421 69.66976929 69.78919983
 70.00817108 69.85887146 69.66976929 70.34655762 70.7048645  70.41622925
 70.24703217 70.34655762 70.4460907  70.65509796 70.68495941 70.79444122
 71.1129303  70.83424377 70.85415649 71.36174774 71.19255066 70.96363831
 70.57546997 70.7048645  70.55557251 70.73471832 70.89396667 70.50580597
 69.56028748 70.07783508 70.12760162 70.32666016 70.20722198 69.87877655
 68.77401733 69.34132385 68.08727264 68.62471771 67.38061523 68.00764465
 68.12708282 68.55505371 67.97779083 67.3706665  67.61948395 68.17684937
 68.8337326  68.95316315 69.28160858 69.41099548 69.55033875 69.61005402
 70.10769653 70.18731689 69.73943329 70.24703217 70.41622925 70.66505432
 70.97358704 71.25226593 72.11816406 72.12811279 72.20774078 71.9788208
 71.61056519 72.24755096 72.50632477 72.4466095  72.7750473  72.86462402
 72.95420074 72.93429565 72.95420074 72.87457275 72.80490875 72.72528076
 71.720047   72.3172226  71.41151428 70.85415649 71.75986481 71.06316376
 71.77976227 72.59590149 72.79496002 72.73524475 73.09353638 72.9641571
 72.68547821 73.6309967  73.00396729 72.76509857 72.41675568 73.4418869
 73.60113525 74.09877777 74.179039   74.92485809 74.20999908 73.48999786
 74.38999939 73.52999878 73.13999939 72.5        72.45999908 72.44999695
 72.55000305 71.69000244 71.98000336 72.19999695 70.5        69.98999786
 69.41000366 68.58000183 68.80000305 68.         68.06999969 68.19999695
 69.51999664 70.08000183 70.75       71.29000092 69.91000366 70.55999756]

The length of the initial time series is 1588.
The first 80.0% of the time series as train, the last 20% as test.

    X_train: [40.49135186 40.49135186 40.49135186 … 57.1533377  56.78368343
 56.60876714]
    X_test: [56.28529285 56.12629848 56.13649247 56.32519397 56.6198664
56.85913999
 57.22069533 57.56570283 57.95182607 58.19304984 58.55073538 58.90532954
 59.18269915 59.30943762 59.4386378  59.53706027 59.65038028 59.83783153
 59.98026989 60.10591986 60.25244776 60.23205392 60.33560801 60.42033408
 60.48054661 60.50977107 60.50817692 60.57610175 60.58795254 60.5138453
 60.45867752 60.36070738 60.36435335 60.42927877 60.53705447 60.63616544
 60.68992912 60.76488807 60.82439609 60.86397566 60.99291587 61.03282667
 61.06001553 61.03125014 61.00953616 61.07921761 61.25100493 61.48993645
 61.65081132 61.74600031 61.83481304 61.84189227 61.82582199 61.87097186
 61.94252791 62.04844122 62.13691891 62.24210297 62.35731179 62.40420611
 62.24946035 62.26130912 62.0177701  62.02437693 62.12269536 62.20860314
 62.38455774 62.5376294  62.73574305 62.8777957  62.97944637 63.09176425
 63.25653389 63.3949888  63.49916128 63.52245096 63.47774244 63.38104242
 63.30192422 63.23901319 63.04179412 62.88772053 62.9292684  62.88492349
 62.74661895 62.50775462 62.4908591  62.46610465 62.54969539 62.65270238
```

```
 62.75519912 62.80626688 62.86262451 62.91237867 62.96766155 62.97099097
 62.971893   63.03639473 63.03269263 62.98047416 63.02337603 63.14956838
 63.30747156 63.43302169 63.48473301 63.62359919 63.90118099 64.12282795
 64.37340458 64.59117541 64.78574779 64.96498372 65.14077975 65.249999
 65.46324353 65.67233159 65.81243245 65.88151434 65.9161742  65.89352074
 65.96972098 65.97376835 65.97343714 65.90757972 65.85369637 65.72762869
 65.67549253 65.5344581  65.45732424 65.36688761 65.39491566 65.26481453
 65.04177158 64.6734555  64.52149508 64.53562277 64.46884317 64.38141279
 64.28801716 64.28083168 64.30227975 64.38541461 64.51537502 64.58527104
 64.68253799 64.70017768 64.77655251 64.82993151 64.89364558 65.01318305
 65.12556136 65.25940927 65.33430594 65.40651587 65.54393535 65.68187523
 65.82935038 65.95001187 66.11431996 66.16677254 66.26252115 66.36636529
 66.39667448 66.44697866 66.49906747 66.57812218 66.6919921  66.77422756
 66.89616671 66.98500426 67.15242442 67.37138586 67.46491074 67.63434464
 67.82398251 67.98642891 68.15213213 68.27313258 68.30108247 68.25289871
 68.33189493 68.38377463 68.446262   68.61216325 68.7460779  68.82467464
 68.8762275  68.90747626 68.88021057 69.01093544 69.1014959  69.16465997
 69.28739014 69.40966881 69.47145565 69.54751429 69.65529018 69.76533213
 69.89544714 69.97457638 70.00470461 70.15323832 70.34035234 70.44061276
 70.49167284 70.55166687 70.61897189 70.71229758 70.79412038 70.88110665
 71.01057552 71.06549307 71.1140697  71.24672774 71.32429497 71.34585704
 71.29244687 71.27243206 71.22872921 71.22576485 71.25248791 71.20330177
 70.98998505 70.91018807 70.85400914 70.84448115 70.81482385 70.73043776
 70.45917264 70.34107145 70.01489414 69.84639937 69.4808123  69.29647083
 69.16750767 69.14033098 69.01242978 68.7966522  68.66565162 68.66049236
 68.77651179 68.89329847 69.04897083 69.20002354 69.34911788 69.482035
 69.68187618 69.85995752 69.92367774 70.068726   70.21837303 70.38635761
 70.58027509 70.78994636 71.11999418 71.39185465 71.62886085 71.780873
 71.8378388  72.00104513 72.18194503 72.31902319 72.49129769 72.64864581
 72.79378144 72.90888608 73.0067053  73.07216429 73.11296946 73.13178059
 72.96316716 72.93452052 72.74529701 72.48845473 72.4440978  72.28027696
 72.27741321 72.42446063 72.58120876 72.69852638 72.86009858 72.96861098
 73.00638234 73.21035969 73.26247502 73.26139147 73.19674123 73.33149261
 73.47089358 73.67603973 73.85857801 74.14444791 74.1563663  74.0352084
 74.09971586 73.99613093 73.84047065 73.59674872 73.39006696 73.21914514
 73.09748295 72.8415774  72.68492757 72.59675837 72.21552957 71.81088745
 71.37436312 70.86629743 70.49060754 70.03776981 69.67999342 69.41090316
 69.43073834 69.54878624 69.76718875 70.04406369 70.01968914 70.11792703]
    y_train: [35.65920258 35.67705154 35.88246155 … 55.52363586 54.26105881
 54.95153046]
    y_test: [53.9750061  54.54711151 55.30662918 56.28315353 57.04266739
57.03280258
 57.93041229 58.1967392  58.75897598 58.35455704 59.22257614 59.55794907
 59.48890305 58.94638824 59.0844841  59.04502869 59.22257614 59.73549652
 59.67631531 59.72563553 59.96236801 59.20285034 59.85386276 59.85386276
 59.80454254 59.69604111 59.55794907 59.93277359 59.69604111 59.24230957
 59.27190018 58.98584747 59.43957901 59.77495193 60.07087326 60.13005447
 59.98209381 60.14977646 60.13991165 60.09059525 60.61338043 60.45109558
```

```
 60.42140961 60.1443367  60.15423203 60.62921143 61.25262451 61.78697205
 61.59896088 61.40105438 61.46043015 61.10419464 60.98544312 61.30210114
 61.49011612 61.74739456 61.75728607 61.93540573 62.09373474 61.83645248
 60.78753662 61.53958893 60.16412735 61.28231049 61.78697205 61.81665802
 62.39059448 62.44007111 62.83588791 62.72703934 62.64787674 62.80620193
 63.20202255 63.22180939 63.17233276 62.83588791 62.48955154 62.16300201
 62.16300201 62.17289734 61.38126373 61.42084503 62.33122635 61.90572357
 61.35157776 60.66879272 61.63854599 61.57917023 62.14321136 62.33122635
 62.4301796  62.25205994 62.33122635 62.35101318 62.4301796  62.20258331
 62.19268799 62.53902435 62.23226929 61.96509171 62.4301796  62.92494965
 63.22180939 63.20202255 62.92494965 63.44940567 64.33999634 64.31030273
 64.68632507 64.7556076  64.8446579  64.95350647 65.11183167 64.92382812
 65.59671021 65.78472137 65.61650085 65.36911011 65.25036621 64.97329712
 65.48786163 65.17121124 65.15142059 64.79518127 64.79518127 64.34989166
 64.62696075 64.09261322 64.30041504 64.15198517 64.70612335 63.87491226
 63.24160385 62.23226929 63.04368973 63.79574203 63.37023926 63.19212341
 63.07338333 63.44940567 63.59783554 63.95407104 64.29051208 64.09261322
 64.31030273 63.97386169 64.31030273 64.26082611 64.36967468 64.73580933
 64.81497955 65.04257202 64.85454559 64.91392517 65.33943176 65.47795868
 65.66597748 65.66597748 66.0222168  65.57691956 65.8638916  66.00242615
 65.70555878 66.22608948 66.28581238 66.48486328 66.75359344 66.69387054
 66.99246216 66.93274689 67.45029449 67.89816284 67.43038177 67.93798065
 68.21665192 68.25646973 68.43562317 68.35599518 67.9678421  67.57967377
 68.22661591 68.15693665 68.26641846 68.89344788 68.88349152 68.71430206
 68.64463043 68.58490753 68.29627991 69.13231659 69.04273987 68.9830246
 69.37117767 69.49062347 69.28160858 69.42094421 69.66976929 69.78919983
 70.00817108 69.85887146 69.66976929 70.34655762 70.7048645  70.41622925
 70.24703217 70.34655762 70.4460907  70.65509796 70.68495941 70.79444122
 71.1129303  70.83424377 70.85415649 71.36174774 71.19255066 70.96363831
 70.57546997 70.7048645  70.55557251 70.73471832 70.89396667 70.50580597
 69.56028748 70.07783508 70.12760162 70.32666016 70.20722198 69.87877655
 68.77401733 69.34132385 68.08727264 68.62471771 67.38061523 68.00764465
 68.12708282 68.55505371 67.97779083 67.3706665  67.61948395 68.17684937
 68.8337326  68.95316315 69.28160858 69.41099548 69.55033875 69.61005402
 70.10769653 70.18731689 69.73943329 70.24703217 70.41622925 70.66505432
 70.97358704 71.25226593 72.11816406 72.12811279 72.20774078 71.9788208
 71.61056519 72.24755096 72.50632477 72.4466095  72.7750473  72.86462402
 72.95420074 72.93429565 72.95420074 72.87457275 72.80490875 72.72528076
 71.720047   72.3172226  71.41151428 70.85415649 71.75986481 71.06316376
 71.77976227 72.59590149 72.79496002 72.73524475 73.09353638 72.9641571
 72.68547821 73.6309967  73.00396729 72.76509857 72.41675568 73.4418869
 73.60113525 74.09877777 74.179039   74.92485809 74.20999908 73.48999786
 74.38999939 73.52999878 73.13999939 72.5        72.45999908 72.44999695
 72.55000305 71.69000244 71.98000336 72.19999695 70.5        69.98999786
 69.41000366 68.58000183 68.80000305 68.         68.06999969 68.19999695
 69.51999664 70.08000183 70.75        71.29000092 69.91000366 70.55999756]

(array([40.49135186, 40.49135186, 40.49135186, …, 57.1533377 ,
```

```
        56.78368343, 56.60876714]), array([56.28529285, 56.12629848, 56.13649247,
56.32519397, 56.6198664 ,
        56.85913999, 57.22069533, 57.56570283, 57.95182607, 58.19304984,
        58.55073538, 58.90532954, 59.18269915, 59.30943762, 59.4386378 ,
        59.53706027, 59.65038028, 59.83783153, 59.98026989, 60.10591986,
        60.25244776, 60.23205392, 60.33560801, 60.42033408, 60.48054661,
        60.50977107, 60.50817692, 60.57610175, 60.58795254, 60.5138453 ,
        60.45867752, 60.36070738, 60.36435335, 60.42927877, 60.53705447,
        60.63616544, 60.68992912, 60.76488807, 60.82439609, 60.86397566,
        60.99291587, 61.03282667, 61.06001553, 61.03125014, 61.00953616,
        61.07921761, 61.25100493, 61.48993645, 61.65081132, 61.74600031,
        61.83481304, 61.84189227, 61.82582199, 61.87097186, 61.94252791,
        62.04844122, 62.13691891, 62.24210297, 62.35731179, 62.40420611,
        62.24946035, 62.26130912, 62.0177701 , 62.02437693, 62.12269536,
        62.20860314, 62.38455774, 62.5376294 , 62.73574305, 62.8777957 ,
        62.97944637, 63.09176425, 63.25653389, 63.3949888 , 63.49916128,
        63.52245096, 63.47774244, 63.38104242, 63.30192422, 63.23901319,
        63.04179412, 62.88772053, 62.9292684 , 62.88492349, 62.74661895,
        62.50775462, 62.4908591 , 62.46610465, 62.54969539, 62.65270238,
        62.75519912, 62.80626688, 62.86262451, 62.91237867, 62.96766155,
        62.97099097, 62.971893  , 63.03639473, 63.03269263, 62.98047416,
        63.02337603, 63.14956838, 63.30747156, 63.43302169, 63.48473301,
        63.62359919, 63.90118099, 64.12282795, 64.37340458, 64.59117541,
        64.78574779, 64.96498372, 65.14077975, 65.249999  , 65.46324353,
        65.67233159, 65.81243245, 65.88151434, 65.9161742 , 65.89352074,
        65.96972098, 65.97376835, 65.97343714, 65.90757972, 65.85369637,
        65.72762869, 65.67549253, 65.5344581 , 65.45732424, 65.36688761,
        65.39491566, 65.26481453, 65.04177158, 64.6734555 , 64.52149508,
        64.53562277, 64.46884317, 64.38141279, 64.28801716, 64.28083168,
        64.30227975, 64.38541461, 64.51537502, 64.58527104, 64.68253799,
        64.70017768, 64.77655251, 64.82993151, 64.89364558, 65.01318305,
        65.12556136, 65.25940927, 65.33430594, 65.40651587, 65.54393535,
        65.68187523, 65.82935038, 65.95001187, 66.11431996, 66.16677254,
        66.26252115, 66.36636529, 66.39667448, 66.44697866, 66.49906747,
        66.57812218, 66.6919921 , 66.77422756, 66.89616671, 66.98500426,
        67.15242442, 67.37138586, 67.46491074, 67.63434464, 67.82398251,
        67.98642891, 68.15213213, 68.27313258, 68.30108247, 68.25289871,
        68.33189493, 68.38377463, 68.446262  , 68.61216325, 68.7460779 ,
        68.82467464, 68.8762275 , 68.90747626, 68.88021057, 69.01093544,
        69.1014959 , 69.16465997, 69.28739014, 69.40966881, 69.47145565,
        69.54751429, 69.65529018, 69.76533213, 69.89544714, 69.97457638,
        70.00470461, 70.15323832, 70.34035234, 70.44061276, 70.49167284,
        70.55166687, 70.61897189, 70.71229758, 70.79412038, 70.88110665,
        71.01057552, 71.06549307, 71.1140697 , 71.24672774, 71.32429497,
        71.34585704, 71.29244687, 71.27243206, 71.22872921, 71.22576485,
        71.25248791, 71.20330177, 70.98998505, 70.91018807, 70.85400914,
        70.84448115, 70.81482385, 70.73043776, 70.45917264, 70.34107145,
        70.01489414, 69.84639937, 69.4808123 , 69.29647083, 69.16750767,
```

```
        69.14033098, 69.01242978, 68.7966522 , 68.66565162, 68.66049236,
        68.77651179, 68.89329847, 69.04897083, 69.20002354, 69.34911788,
        69.482035  , 69.68187618, 69.85995752, 69.92367774, 70.068726  ,
        70.21837303, 70.38635761, 70.58027509, 70.78994636, 71.11999418,
        71.39185465, 71.62886085, 71.780873  , 71.8378388 , 72.00104513,
        72.18194503, 72.31902319, 72.49129769, 72.64864581, 72.79378144,
        72.90888608, 73.0067053 , 73.07216429, 73.11296946, 73.13178059,
        72.96316716, 72.93452052, 72.74529701, 72.48845473, 72.4440978 ,
        72.28027696, 72.27741321, 72.42446063, 72.58120876, 72.69852638,
        72.86009858, 72.96861098, 73.00638234, 73.21035969, 73.26247502,
        73.26139147, 73.19674123, 73.33149261, 73.47089358, 73.67603973,
        73.85857801, 74.14444791, 74.1563663 , 74.0352084 , 74.09971586,
        73.99613093, 73.84047065, 73.59674872, 73.39006696, 73.21914514,
        73.09748295, 72.8415774 , 72.68492757, 72.59675837, 72.21552957,
        71.81088745, 71.37436312, 70.86629743, 70.49060754, 70.03776981,
        69.67999342, 69.41090316, 69.43073834, 69.54878624, 69.76718875,
        70.04406369, 70.01968914, 70.11792703]), array([35.65920258, 35.67705154,
35.88246155, …, 55.52363586,
        54.26105881, 54.95153046]), array([53.9750061 , 54.54711151, 55.30662918,
56.28315353, 57.04266739,
        57.03280258, 57.93041229, 58.1967392 , 58.75897598, 58.35455704,
        59.22257614, 59.55794907, 59.48890305, 58.94638824, 59.0844841 ,
        59.04502869, 59.22257614, 59.73549652, 59.67631531, 59.72563553,
        59.96236801, 59.20285034, 59.85386276, 59.85386276, 59.80454254,
        59.69604111, 59.55794907, 59.93277359, 59.69604111, 59.24230957,
        59.27190018, 58.98584747, 59.43957901, 59.77495193, 60.07087326,
        60.13005447, 59.98209381, 60.14977646, 60.13991165, 60.09059525,
        60.61338043, 60.45109558, 60.42140961, 60.1443367 , 60.15423203,
        60.62921143, 61.25262451, 61.78697205, 61.59896088, 61.40105438,
        61.46043015, 61.10419464, 60.98544312, 61.30210114, 61.49011612,
        61.74739456, 61.75728607, 61.93540573, 62.09373474, 61.83645248,
        60.78753662, 61.53958893, 60.16412735, 61.28231049, 61.78697205,
        61.81665802, 62.39059448, 62.44007111, 62.83588791, 62.72703934,
        62.64787674, 62.80620193, 63.20202255, 63.22180939, 63.17233276,
        62.83588791, 62.48955154, 62.16300201, 62.16300201, 62.17289734,
        61.38126373, 61.42084503, 62.33122635, 61.90572357, 61.35157776,
        60.66879272, 61.63854599, 61.57917023, 62.14321136, 62.33122635,
        62.4301796 , 62.25205994, 62.33122635, 62.35101318, 62.4301796 ,
        62.20258331, 62.19268799, 62.53902435, 62.23226929, 61.96509171,
        62.4301796 , 62.92494965, 63.22180939, 63.20202255, 62.92494965,
        63.44940567, 64.33999634, 64.31030273, 64.68632507, 64.7556076 ,
        64.8446579 , 64.95350647, 65.11183167, 64.92382812, 65.59671021,
        65.78472137, 65.61650085, 65.36911011, 65.25036621, 64.97329712,
        65.48786163, 65.17121124, 65.15142059, 64.79518127, 64.79518127,
        64.34989166, 64.62696075, 64.09261322, 64.30041504, 64.15198517,
        64.70612335, 63.87491226, 63.24160385, 62.23226929, 63.04368973,
        63.79574203, 63.37023926, 63.19212341, 63.07338333, 63.44940567,
        63.59783554, 63.95407104, 64.29051208, 64.09261322, 64.31030273,
```

```
       63.97386169, 64.31030273, 64.26082611, 64.36967468, 64.73580933,
       64.81497955, 65.04257202, 64.85454559, 64.91392517, 65.33943176,
       65.47795868, 65.66597748, 65.66597748, 66.0222168 , 65.57691956,
       65.8638916 , 66.00242615, 65.70555878, 66.22608948, 66.28581238,
       66.48486328, 66.75359344, 66.69387054, 66.99246216, 66.93274689,
       67.45029449, 67.89816284, 67.43038177, 67.93798065, 68.21665192,
       68.25646973, 68.43562317, 68.35599518, 67.9678421 , 67.57967377,
       68.22661591, 68.15693665, 68.26641846, 68.89344788, 68.88349152,
       68.71430206, 68.64463043, 68.58490753, 68.29627991, 69.13231659,
       69.04273987, 68.9830246 , 69.37117767, 69.49062347, 69.28160858,
       69.42094421, 69.66976929, 69.78919983, 70.00817108, 69.85887146,
       69.66976929, 70.34655762, 70.7048645 , 70.41622925, 70.24703217,
       70.34655762, 70.4460907 , 70.65509796, 70.68495941, 70.79444122,
       71.1129303 , 70.83424377, 70.85415649, 71.36174774, 71.19255066,
       70.96363831, 70.57546997, 70.7048645 , 70.55557251, 70.73471832,
       70.89396667, 70.50580597, 69.56028748, 70.07783508, 70.12760162,
       70.32666016, 70.20722198, 69.87877655, 68.77401733, 69.34132385,
       68.08727264, 68.62471771, 67.38061523, 68.00764465, 68.12708282,
       68.55505371, 67.97779083, 67.3706665 , 67.61948395, 68.17684937,
       68.8337326 , 68.95316315, 69.28160858, 69.41099548, 69.55033875,
       69.61005402, 70.10769653, 70.18731689, 69.73943329, 70.24703217,
       70.41622925, 70.66505432, 70.97358704, 71.25226593, 72.11816406,
       72.12811279, 72.20774078, 71.9788208 , 71.61056519, 72.24755096,
       72.50632477, 72.4466095 , 72.7750473 , 72.86462402, 72.95420074,
       72.93429565, 72.95420074, 72.87457275, 72.80490875, 72.72528076,
       71.720047  , 72.3172226 , 71.41151428, 70.85415649, 71.75986481,
       71.06316376, 71.77976227, 72.59590149, 72.79496002, 72.73524475,
       73.09353638, 72.9641571 , 72.68547821, 73.6309967 , 73.00396729,
       72.76509857, 72.41675568, 73.4418869 , 73.60113525, 74.09877777,
       74.179039  , 74.92485809, 74.20999908, 73.48999786, 74.38999939,
       73.52999878, 73.13999939, 72.5       , 72.45999908, 72.44999695,
       72.55000305, 71.69000244, 71.98000336, 72.19999695, 70.5       ,
       69.98999786, 69.41000366, 68.58000183, 68.80000305, 68.        ,
       68.06999969, 68.19999695, 69.51999664, 70.08000183, 70.75      ,
       71.29000092, 69.91000366, 70.55999756]))
```

3.3.2 3.3.2. Train the model using scikit-learn

[22]:
```
lr = LinearRegression(fit_intercept=True)
lr.fit(X_train.reshape(-1, 1), y_train)
```

[22]:
```
LinearRegression()
```

3.3.3 3.3.3. Retrieve and present statistical measures and key numbers i.e. R^2, intercept and coefficient.

[23]:
```
lr.coef_
```

[23]:
```
array([1.17302995])
```

The coefficient(estimated slope) of our simple linear regression model is 1.17302994.

[24]:
```
lr.intercept_
```

[24]:
```
-10.923473833260871
```

The estimated intercept of our simple linear regression model is -10.923472656011356.

Hence, our fitted model is:

$$\hat{Y} = 1.17302994X - 10.923472656011356$$

[25]:
```
r2_score(y_train, lr.predict(X_train.reshape(-1, 1)))
```

[25]:
```
0.984578187294296
```

The R^2 of our simple linear regression model is 0.9845781871086455.

[26]:
```
mean_squared_error(y_train, lr.predict(X_train.reshape(-1, 1)))
```

[26]:
```
0.5708205565316868
```

The $MSE = \frac{SSE}{n-2} = \frac{\|Y-\hat{Y}\|^2}{n-2}$ (Mean Squared Error) of our simple linear regression model is 0.5708204284208702.

3.4 3.4. Visualize the results in one plot using Matplotlib.Pyplot

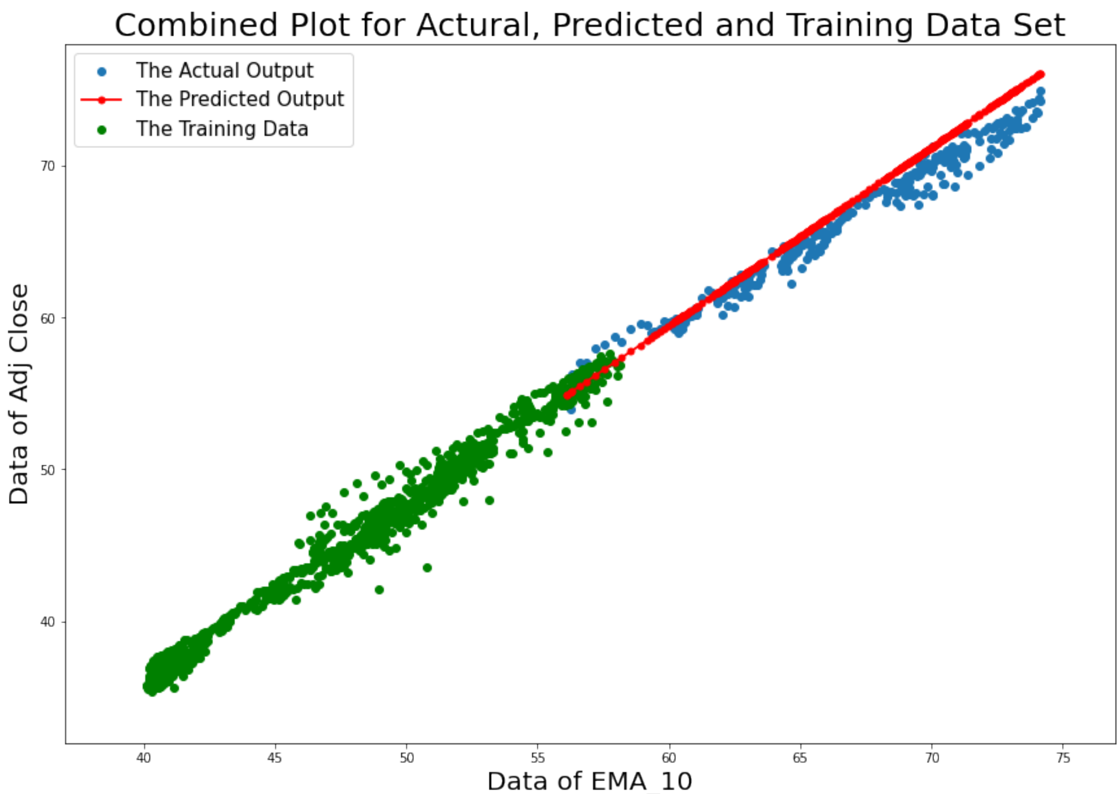
Create a combined plot that shows 1. the actual output 2. the predicted output 3. the training data

Make sure to add a legend and labels to each line and that every series is at the "correct" x coordinates.

[27]:
```
y_predict = lr.predict(X_test.reshape(-1, 1))
```

[28]:
```
plt.figure()

plt.subplots_adjust(top = 2, right = 2)
plt.xlim(round(min(msci["EMA_10"]) - 3), round(max(msci["EMA_10"]) + 3))
plt.ylim(round(min(msci["Adj Close"]) - 3), round(max(msci["Adj Close"]) + 3))

plt.scatter(X_test, y_test, label = "The Actual Output")
plt.plot(X_test, y_predict, label = "The Predicted Output", color = "red",
```

```
        marker=".", linewidth=1.8, markersize=10)
plt.scatter(X_train, y_train, label = "The Training Data", color = "green")
plt.xlabel("Data of EMA_10", fontsize = 20)
plt.ylabel("Data of Adj Close", fontsize = 20)
plt.title("Combined Plot for Actural, Predicted and Training Data Set",␣
 ↪fontsize=25)

plt.legend(fontsize = 15)
plt.show()
```

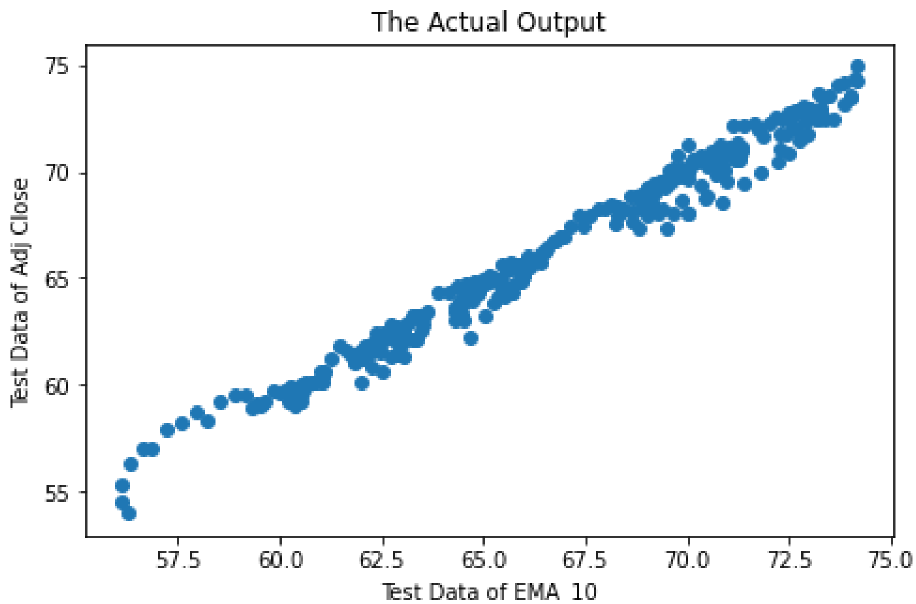


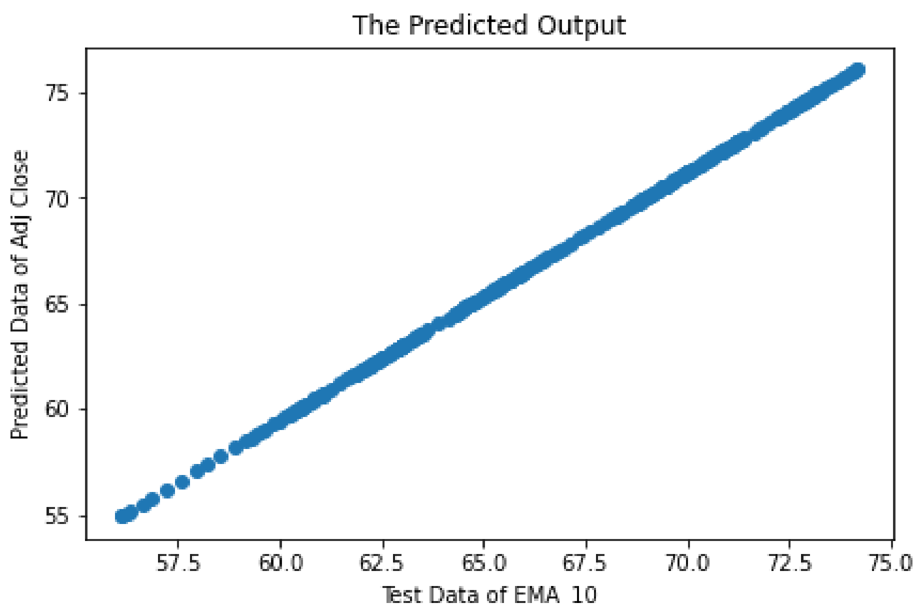
Also create subplots for each data set.

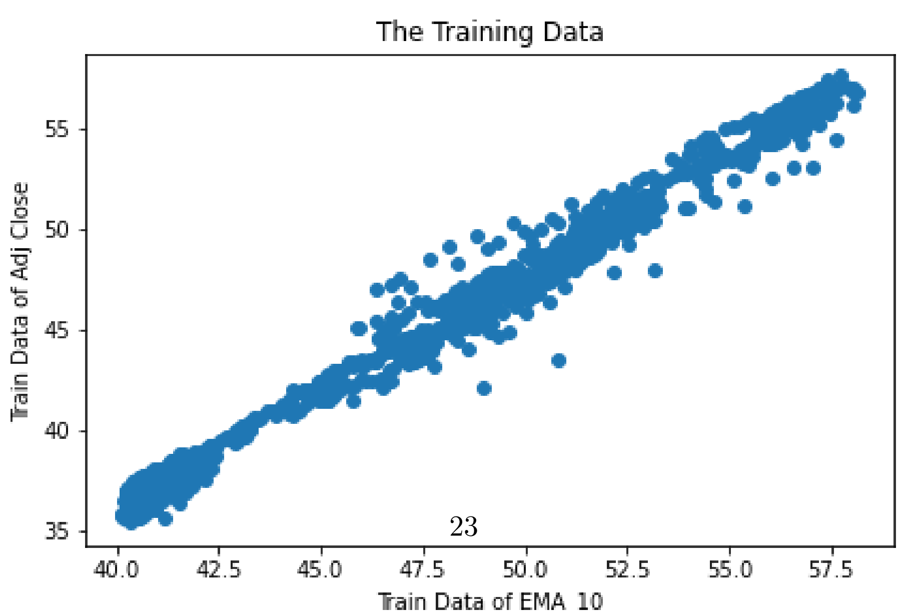
[29]:
```
fig, axs = plt.subplots(3)
plt.subplots_adjust(top = 3, right=1, hspace = 0.3)

axs[0].scatter(X_test, y_test)
axs[0].set_title("The Actual Output")
axs.flat[0].set(xlabel = "Test Data of EMA_10",
                ylabel = "Test Data of Adj Close")

axs[1].scatter(X_test, y_predict)
axs[1].set_title("The Predicted Output")
```

```
axs.flat[1].set(xlabel = "Test Data of EMA_10",
                ylabel = "Predicted Data of Adj Close")


axs[2].scatter(X_train, y_train)
axs[2].set_title("The Training Data")
axs.flat[2].set(xlabel = "Train Data of EMA_10",
                ylabel = "Train Data of Adj Close")
```

[29]:
```
[Text(0.5, 0, 'Train Data of EMA_10'), Text(0, 0.5, 'Train Data of Adj Close')]
```


The Actual Output
Test Data of Adj Close
Test Data of EMA_10
55
60
65
70
75
57.5
60.0
62.5
65.0
67.5
70.0
72.5
75.0


The Predicted Output
Predicted Data of Adj Close
Test Data of EMA_10
55
60
65
70
75
57.5
60.0
62.5
65.0
67.5
70.0
72.5
75.0


The Training Data
Train Data of Adj Close
Train Data of EMA_10
35
40
45
50
55
40.0
42.5
45.0
47.5
50.0
52.5
55.0
57.5

4 4 Optional: Basic Data Structures in Python

4.1 4.1. Create an empty dictionary

[30]:
```
dic = {}
```

4.2 4.2. Update the new dictionary with any key values

[31]:
```
dic["x"] = 1
dic
```

[31]:
```
{'x': 1}
```

4.3 4.3. List comprehension: iterate through the following dictionaries values and return all values as string

new_dict = {0: 1, 1 : "a", 2: 0.34, 3 : True}

[32]:
```
new_dict = {0: 1, 1 : "a", 2: 0.34, 3 : True}
for value in new_dict.values():
    print(str(value), type(str(value)))
```

```
1 <class 'str'>
a <class 'str'>
0.34 <class 'str'>
True <class 'str'>
```

4.4 4.4. Convert the two lists to sets and then return the union of the two

L1 = ["A", "B", "C", "D"]

L2 = ["B", "D", "E", "F"]

[33]:
```
L1, L2 = ["A", "B", "C", "D"], ["B", "D", "E", "F"]
set(L1) | set(L2)
```

[33]:
```
{'A', 'B', 'C', 'D', 'E', 'F'}
```

Or alternatively:

[34]:
```
set(L1 + L2)
```

[34]:
```
{'A', 'B', 'C', 'D', 'E', 'F'}
```

4.5 4.5. Iterate through the first list and check if it is in the second by element-wise comparison

[35]:
```
for i in L1:
    for j in L2:
        if i == j:
            print(f"Element {i} is ALSO in L2!")
            break
    else: print(f"Element {i} is NOT in L2!")
```

```
Element A is NOT in L2!
Element B is ALSO in L2!
Element C is NOT in L2!
Element D is ALSO in L2!
```