JP Morgan & Co. Stock Regression Analysis

- By

1.) SHIVANI MOGILI (10473465)
2.) VARNIKA TOSHNIWAL (10473454)
3.) DEEPSHIKA REDDY AG (10473464)
4.) MANAV SHARMA (10466575)

SECTION 1

INTRODUCTION

The data given for this project comprises of four columns/random variables-the daily ETF return, the daily relative change in the crude oil price, the daily relative change in the gold price, and the daily return of the JPMorgan Chase & Co stock. The given sample size is 1000. The objective of this course project is to perform an exploratory data analysis on the given data, and determine the existence of linear relationship between the variables under study, by performing a linear regression on the data and then evaluating the accuracy of the model, by means of using any statistical tool. We have used Python for our project to perform the necessary statistical tests.

OBJECTIVE

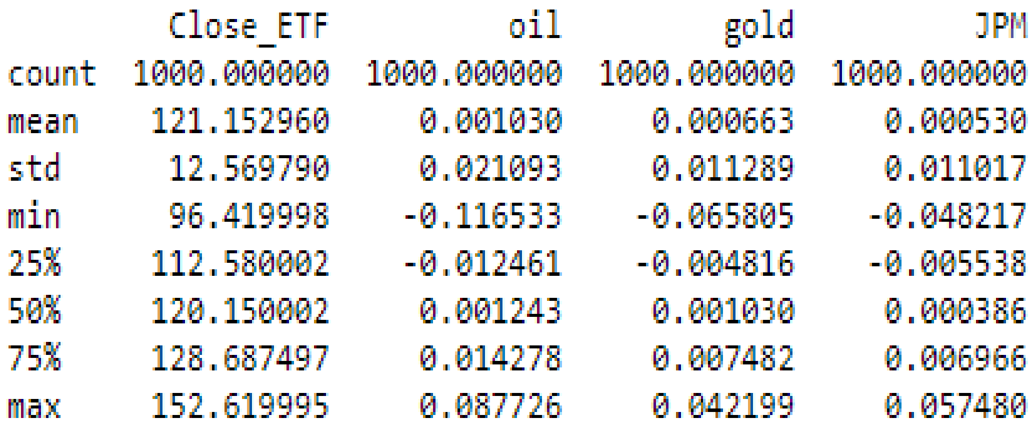
To find the regression analysis of given data, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables)

SECTION 2

PART 1- Meet Your Data

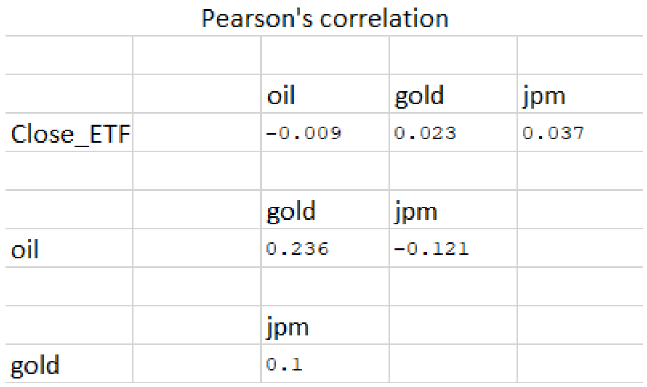
The data given for this project comprises of four columns/random variables-the daily ETF return, the daily relative change in the crude oil price, the daily relative change in the gold price, and the daily return of the JPMorgan Chase & Co stock. The given sample size is 1000. The objective of this course project is to perform an exploratory data analysis on the given data, and determine the existence of linear relationship between the variables under study, by performing a linear regression on the data and then evaluating the

accuracy of the model, by means of using any statistical tool. We have used Python for our project to perform the necessary statistical tests.



```
         Close_ETF          oil         gold          JPM
count  1000.000000  1000.000000  1000.000000  1000.000000
mean    121.152960     0.001030     0.000663     0.000530
std      12.569790     0.021093     0.011289     0.011017
min      96.419998    -0.116533    -0.065805    -0.048217
25%     112.580002    -0.012461    -0.004816    -0.005538
50%     120.150002     0.001243     0.001030     0.000386
75%     128.687497     0.014278     0.007482     0.006966
max     152.619995     0.087726     0.042199     0.057480
```

The correlation between the variables are as followed:-



Pearson's correlation

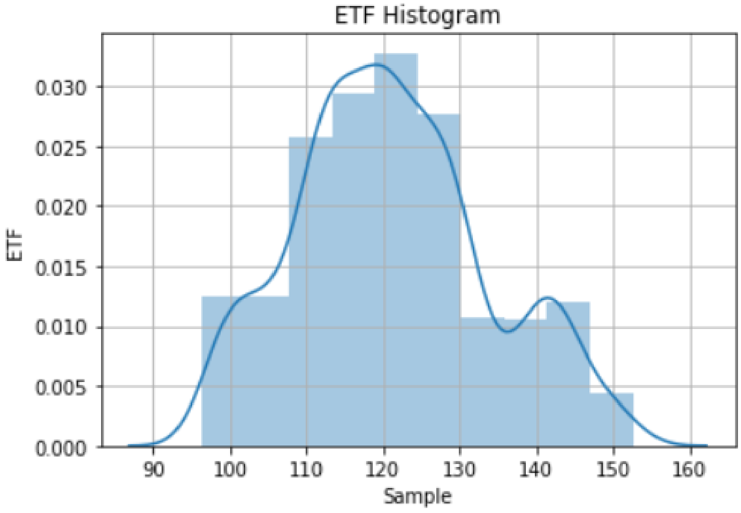
		oil	gold	jpm
Close_ETF		-0.009	0.023	0.037
		gold	jpm	
oil		0.236	-0.121	
		jpm		
gold		0.1		

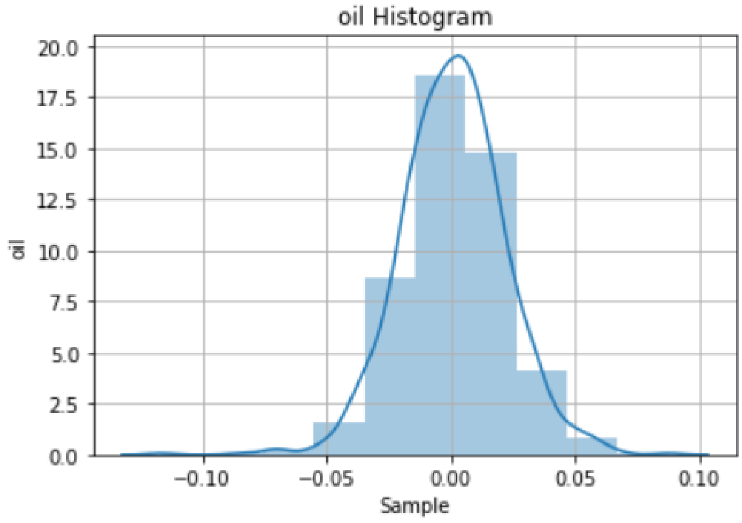
There is a weak negative correlation between Close_ETF and oil, oil and JPM.
There is a weak positive correlation between Close_ETF and gold, Close_ETF and JPM, Oil and gold, gold and JPM

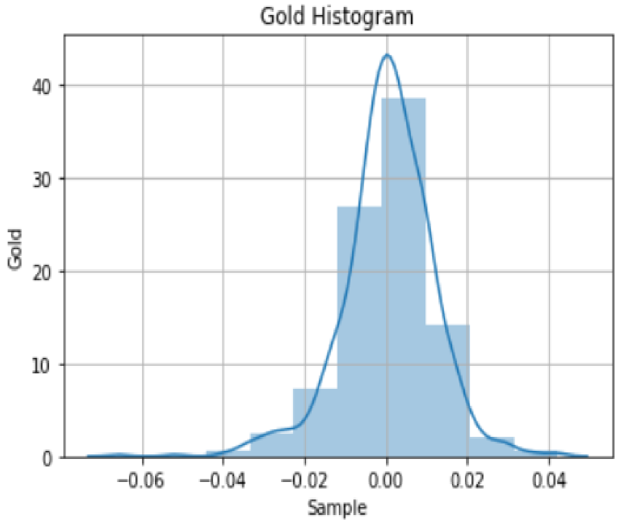
PART 2- Describe Your Data

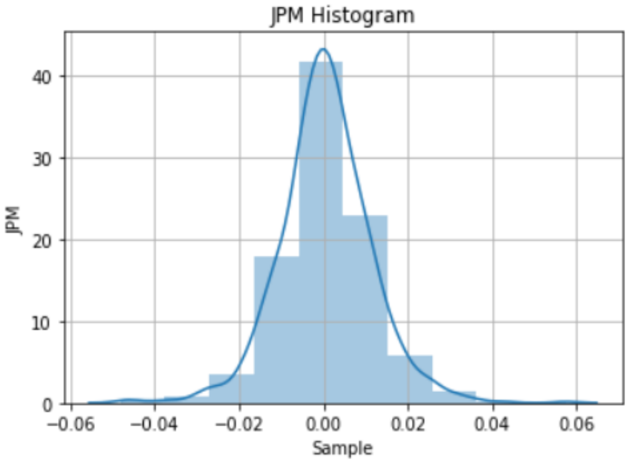
1. **A Histogram for each Column:**

We plot one histogram for each of the variables to determine the type of distribution of the data. A histogram is an approximate representation of the distribution of numerical data. This is done using the seaborn and matplotlib libraries of Python.





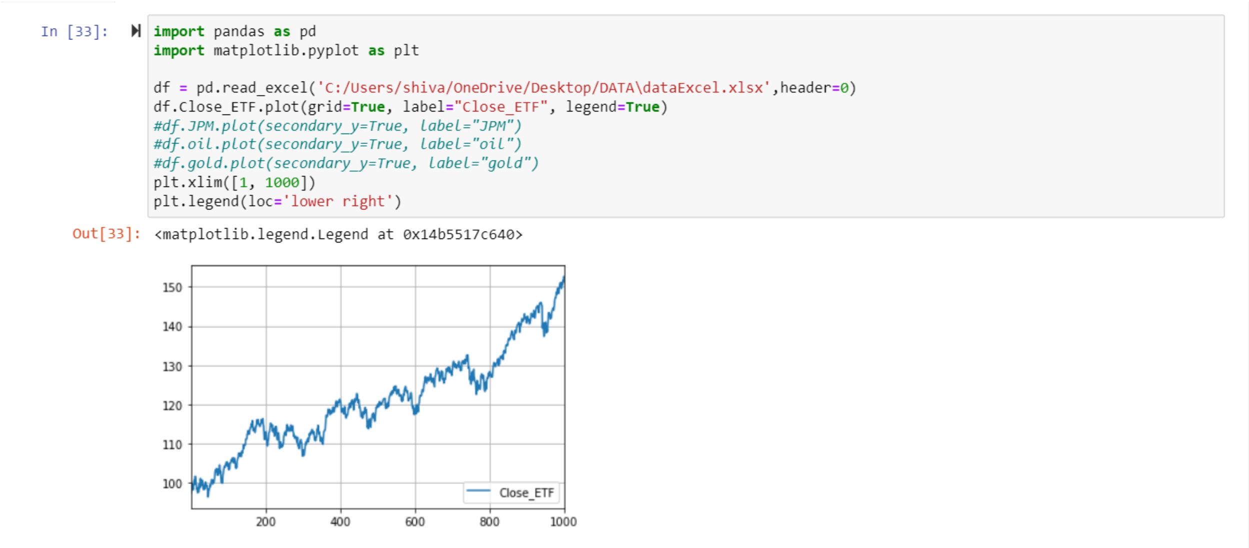




2. Time series plot for each column:

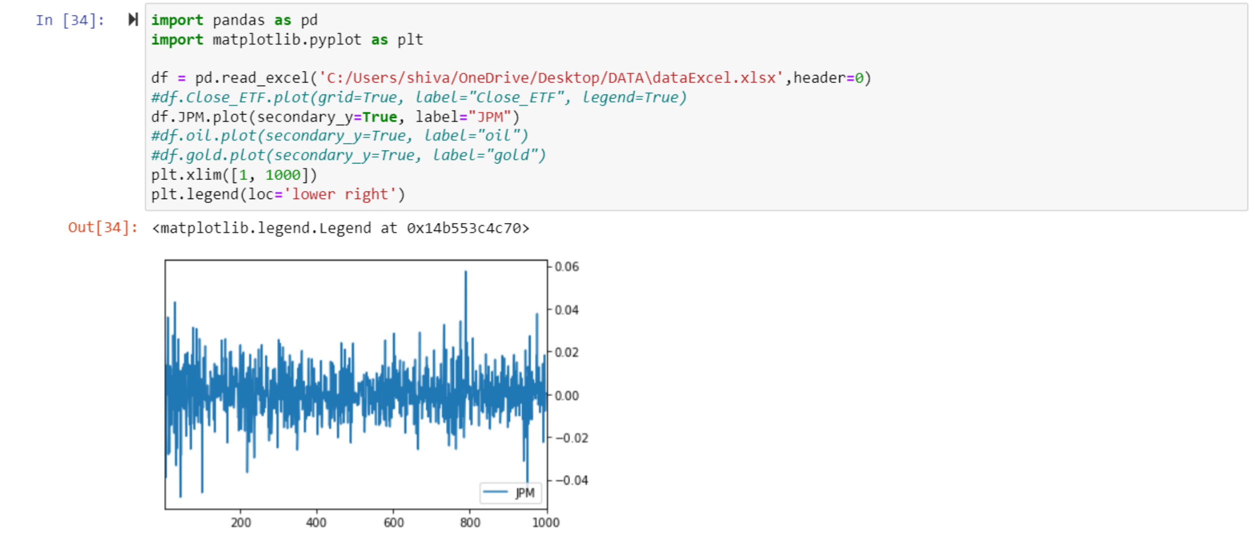
The time-series plot is a univariate plot: it shows only one variable. It is a 2-dimensional plot in which one axis, the time-axis, shows graduations at an appropriate scale (seconds, minutes, weeks, quarters, years), while the other axis shows the numeric values.

1)Close_ETF



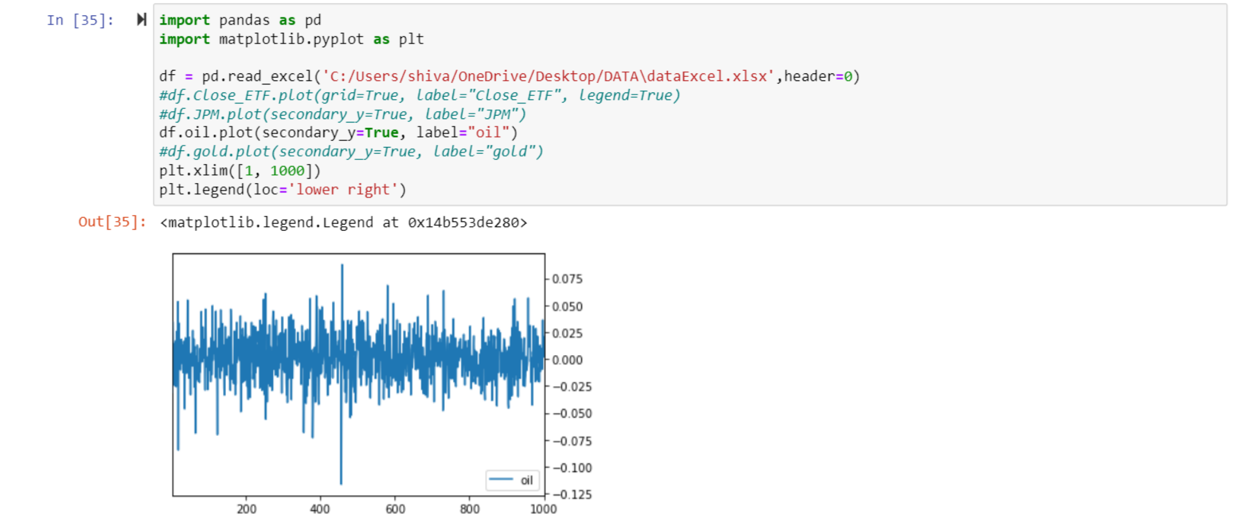
```
In [33]:  import pandas as pd
          import matplotlib.pyplot as plt

          df = pd.read_excel('C:/Users/shiva/OneDrive/Desktop/DATA\dataExcel.xlsx',header=0)
          df.Close_ETF.plot(grid=True, label="Close_ETF", legend=True)
          #df.JPM.plot(secondary_y=True, label="JPM")
          #df.oil.plot(secondary_y=True, label="oil")
          #df.gold.plot(secondary_y=True, label="gold")
          plt.xlim([1, 1000])
          plt.legend(loc='lower right')

Out[33]:  <matplotlib.legend.Legend at 0x14b5517c640>
```

2)JPM



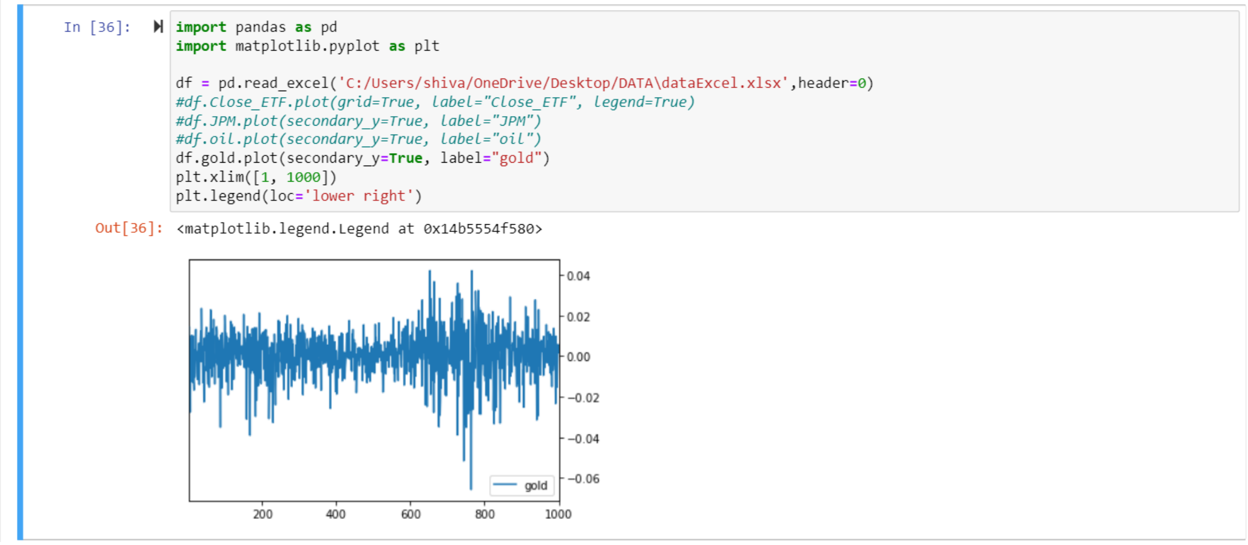
```
In [34]:  import pandas as pd
          import matplotlib.pyplot as plt

          df = pd.read_excel('C:/Users/shiva/OneDrive/Desktop/DATA\dataExcel.xlsx',header=0)
          #df.Close_ETF.plot(grid=True, label="Close_ETF", legend=True)
          df.JPM.plot(secondary_y=True, label="JPM")
          #df.oil.plot(secondary_y=True, label="oil")
          #df.gold.plot(secondary_y=True, label="gold")
          plt.xlim([1, 1000])
          plt.legend(loc='lower right')

Out[34]:  <matplotlib.legend.Legend at 0x14b553c4c70>
```

3)Oil



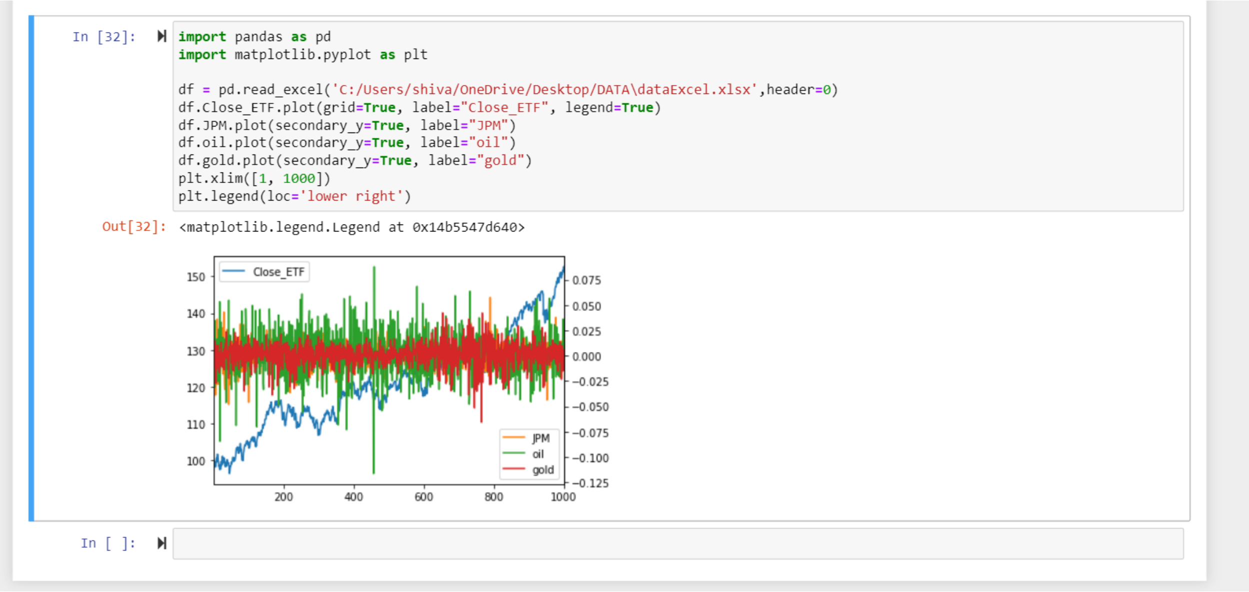
```
In [35]:  import pandas as pd
          import matplotlib.pyplot as plt

          df = pd.read_excel('C:/Users/shiva/OneDrive/Desktop/DATA\dataExcel.xlsx',header=0)
          #df.Close_ETF.plot(grid=True, label="Close_ETF", legend=True)
          #df.JPM.plot(secondary_y=True, label="JPM")
          df.oil.plot(secondary_y=True, label="oil")
          #df.gold.plot(secondary_y=True, label="gold")
          plt.xlim([1, 1000])
          plt.legend(loc='lower right')

Out[35]:  <matplotlib.legend.Legend at 0x14b553de280>
```

4)Gold



```
In [36]:  import pandas as pd
          import matplotlib.pyplot as plt

          df = pd.read_excel('C:/Users/shiva/OneDrive/Desktop/DATA\dataExcel.xlsx',header=0)
          #df.Close_ETF.plot(grid=True, label="Close_ETF", legend=True)
          #df.JPM.plot(secondary_y=True, label="JPM")
          #df.oil.plot(secondary_y=True, label="oil")
          df.gold.plot(secondary_y=True, label="gold")
          plt.xlim([1, 1000])
          plt.legend(loc='lower right')

Out[36]: <matplotlib.legend.Legend at 0x14b5554f580>
```

2.c) Time series (All four columns)

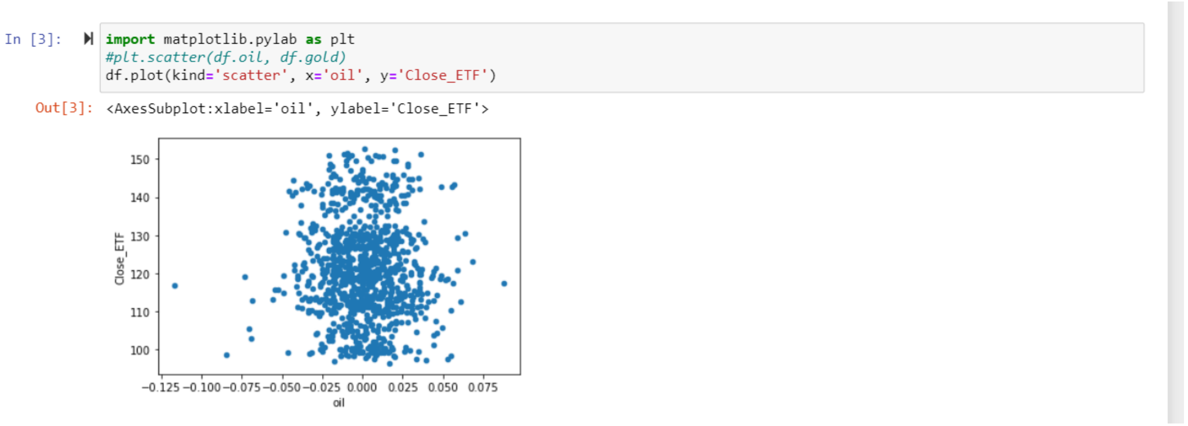


```
In [32]:  import pandas as pd
          import matplotlib.pyplot as plt

          df = pd.read_excel('C:/Users/shiva/OneDrive/Desktop/DATA\dataExcel.xlsx',header=0)
          df.Close_ETF.plot(grid=True, label="Close_ETF", legend=True)
          df.JPM.plot(secondary_y=True, label="JPM")
          df.oil.plot(secondary_y=True, label="oil")
          df.gold.plot(secondary_y=True, label="gold")
          plt.xlim([1, 1000])
          plt.legend(loc='lower right')

Out[32]: <matplotlib.legend.Legend at 0x14b5547d640>

In [ ]:
```

2.d) Scatter Plots

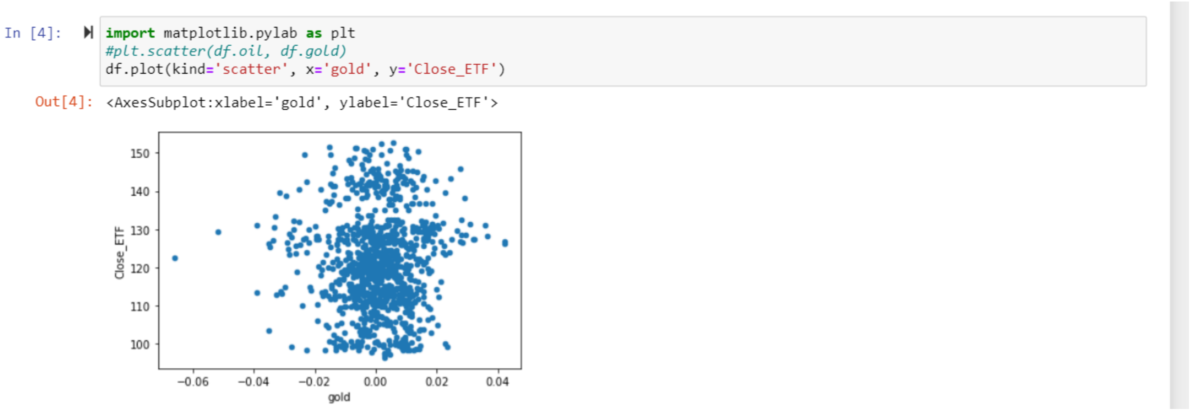
A scatter plot uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

Relationship between oil and Close_ETF



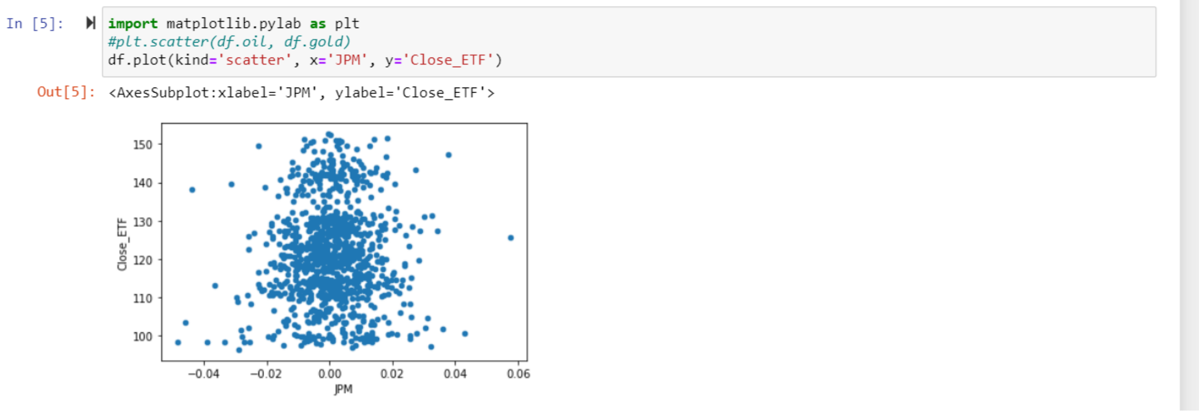
```
In [3]:  import matplotlib.pylab as plt
         #plt.scatter(df.oil, df.gold)
         df.plot(kind='scatter', x='oil', y='Close_ETF')

Out[3]:  <AxesSubplot:xlabel='oil', ylabel='Close_ETF'>
```

Relationship between gold and Close_ETF



```
In [4]:  import matplotlib.pylab as plt
         #plt.scatter(df.oil, df.gold)
         df.plot(kind='scatter', x='gold', y='Close_ETF')

Out[4]:  <AxesSubplot:xlabel='gold', ylabel='Close_ETF'>
```

Relationship between JPM and Close_ETF



```
In [5]:  import matplotlib.pylab as plt
         #plt.scatter(df.oil, df.gold)
         df.plot(kind='scatter', x='JPM', y='Close_ETF')

Out[5]:  <AxesSubplot:xlabel='JPM', ylabel='Close_ETF'>
```

Interpretation-

Each scatter plot shown above, depicts a positive correlation between the variables ETF and the remaining independent variables. The points in the plot are densely scattered, as seen in all the plots.

PART 3- What Distribution Does Your Data Follow

Hypotheses: -

ETF

Null Hypothesis- H0- The distribution of the data is normal

Alternate Hypothesis- H1- The distribution of the data is not normal

OIL

Null Hypothesis- H0- The distribution of the data is normal

Alternate Hypothesis- H1- The distribution of the data is not normal

Gold

Null Hypothesis- H0- The distribution of the data is normal

Alternate Hypothesis- H1- The distribution of the data is not normal

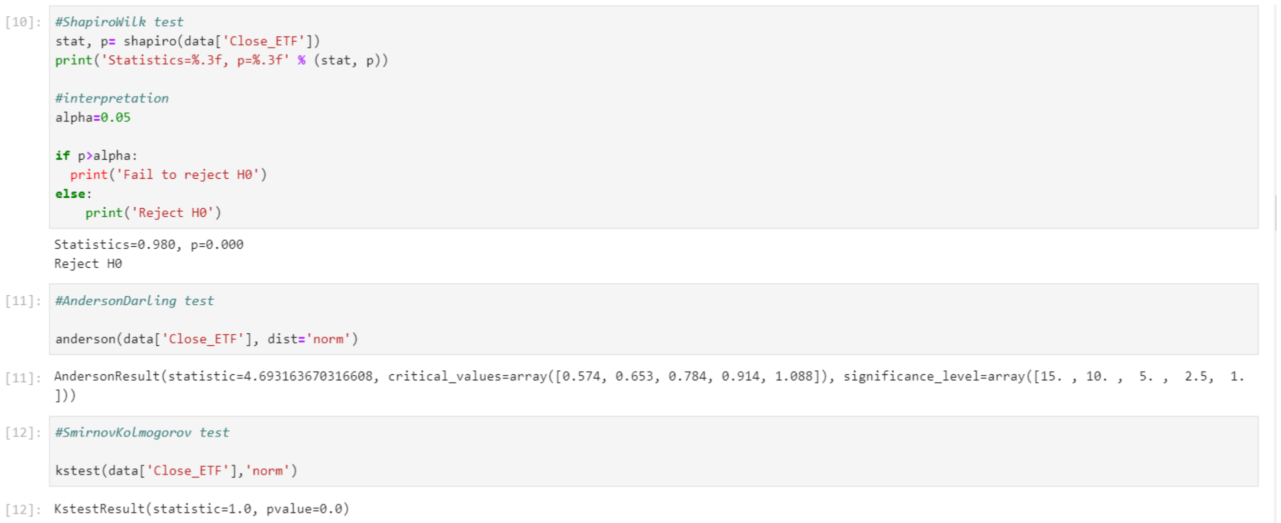
JPM

Null Hypothesis- H0- The distribution of the data is normal

Alternate Hypothesis- H1- The distribution of the data is not normal

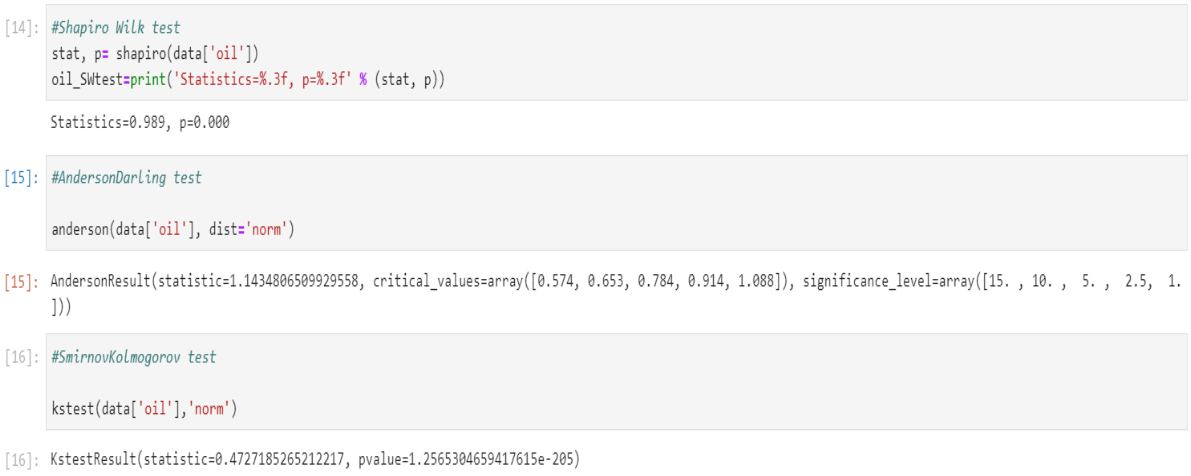
To verify the above hypotheses, we have used normality tests like the Shapiro-Wilk Test, Anderson-Darling Test, Kolmogorov Smirnov Test and QQ Plot on each variable to assess the normality of the data.

ETF:-



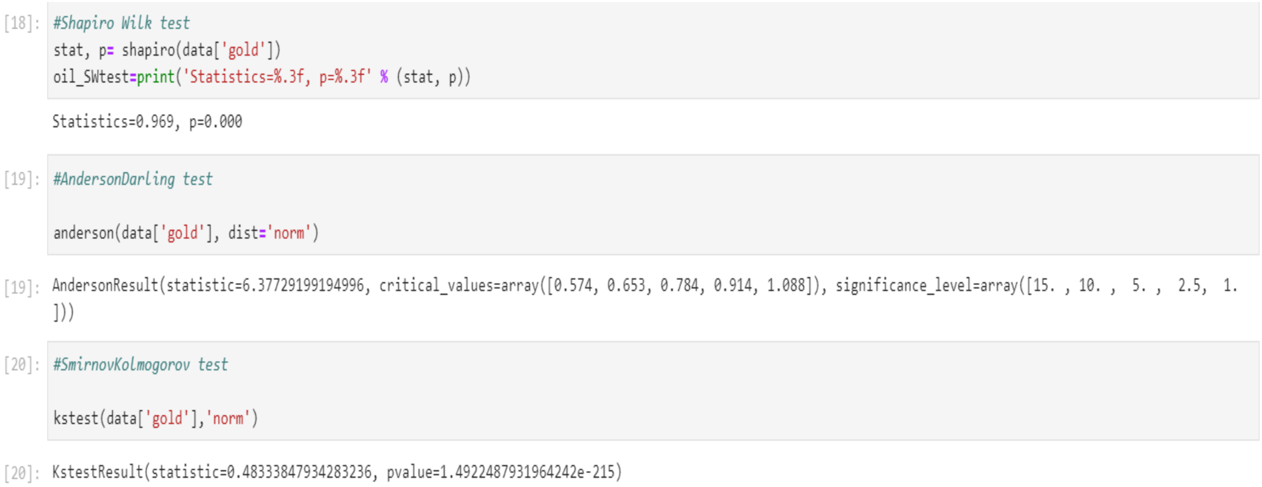
```
[10]: #ShapiroWilk test
      stat, p= shapiro(data['Close_ETF'])
      print('Statistics=%.3f, p=%.3f' % (stat, p))

      #interpretation
      alpha=0.05

      if p>alpha:
        print('Fail to reject H0')
      else:
          print('Reject H0')

      Statistics=0.980, p=0.000
      Reject H0

[11]: #AndersonDarling test

      anderson(data['Close_ETF'], dist='norm')

[11]: AndersonResult(statistic=4.693163670316608, critical_values=array([0.574, 0.653, 0.784, 0.914, 1.088]), significance_level=array([15. , 10. ,  5. ,  2.5,  1.
      ]))

[12]: #SmirnovKolmogorov test

      kstest(data['Close_ETF'],'norm')

[12]: KstestResult(statistic=1.0, pvalue=0.0)
```

OIL:-



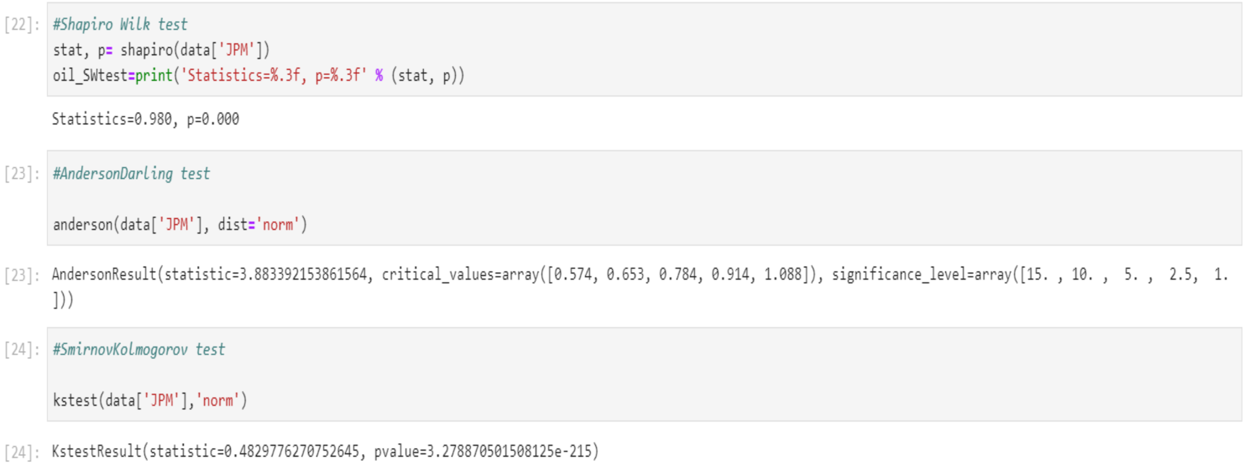
```
[14]: #Shapiro Wilk test
      stat, p= shapiro(data['oil'])
      oil_SWtest=print('Statistics=%.3f, p=%.3f' % (stat, p))

      Statistics=0.989, p=0.000

[15]: #AndersonDarling test

      anderson(data['oil'], dist='norm')

[15]: AndersonResult(statistic=1.1434806509929558, critical_values=array([0.574, 0.653, 0.784, 0.914, 1.088]), significance_level=array([15. , 10. ,  5. ,  2.5,  1.
      ]))

[16]: #SmirnovKolmogorov test

      kstest(data['oil'],'norm')

[16]: KstestResult(statistic=0.4727185265212217, pvalue=1.2565304659417615e-205)
```

GOLD:-



```
[18]: #Shapiro Wilk test
      stat, p= shapiro(data['gold'])
      oil_SWtest=print('Statistics=%.3f, p=%.3f' % (stat, p))

      Statistics=0.969, p=0.000

[19]: #AndersonDarling test

      anderson(data['gold'], dist='norm')

[19]: AndersonResult(statistic=6.37729199194996, critical_values=array([0.574, 0.653, 0.784, 0.914, 1.088]), significance_level=array([15. , 10. ,  5. ,  2.5,  1.
      ]))

[20]: #SmirnovKolmogorov test

      kstest(data['gold'],'norm')

[20]: KstestResult(statistic=0.48333847934283236, pvalue=1.4922487931964242e-215)
```

JPM:-l



```
[22]: #Shapiro Wilk test
      stat, p= shapiro(data['JPM'])
      oil_SWtest=print('Statistics=%.3f, p=%.3f' % (stat, p))

      Statistics=0.980, p=0.000

[23]: #AndersonDarling test

      anderson(data['JPM'], dist='norm')

[23]: AndersonResult(statistic=3.883392153861564, critical_values=array([0.574, 0.653, 0.784, 0.914, 1.088]), significance_level=array([15. , 10. ,  5. ,  2.5,  1.
      ]))

[24]: #SmirnovKolmogorov test

      kstest(data['JPM'],'norm')

[24]: KstestResult(statistic=0.4829776270752645, pvalue=3.278870501508125e-215)
```

Interpretation:-

The p-values obtained from each of the normality tests, as shown above, are 0 which is less than 0.05. Hence, we reject the null hypothesis for each variable, that the distribution of the data is normal.

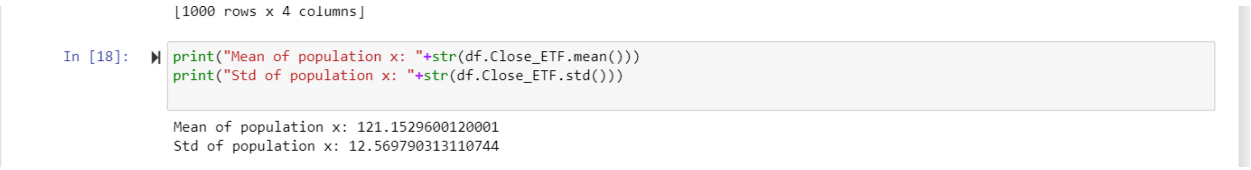
However, the histograms plotted in Part 2, show a bell curved shape, suggesting that the distribution of the data is normal.

It should also be noted that a small p-value indicates that the results are inconsistent with the null hypothesis. In this case, the p-value is low because of the relatively large sample size, which is the entire population in this case. When the sample size is large, a test is likely to report significant differences from the normal distribution even for small deviations from normal.

Hence, in actuality, the distribution of the data is normal but taking the entire population of a variable as its sample results in an extremely low p-value.

PART 4- Sampling And Central Limit Theorem

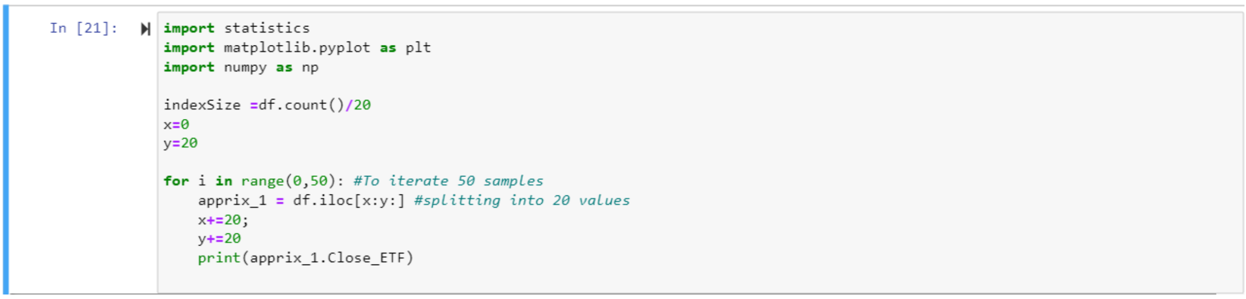
1) Mean and Standard Deviation Calculation



```
[1000 rows x 4 columns]

In [18]:  print("Mean of population x: "+str(df.Close_ETF.mean()))
           print("Std of population x: "+str(df.Close_ETF.std()))

           Mean of population x: 121.1529600120001
           Std of population x: 12.569790313110744
```

2) Break the population into 50 groups sequentially and each group includes 20 values.

We are considering small sample size and breaking each group into 20 values.



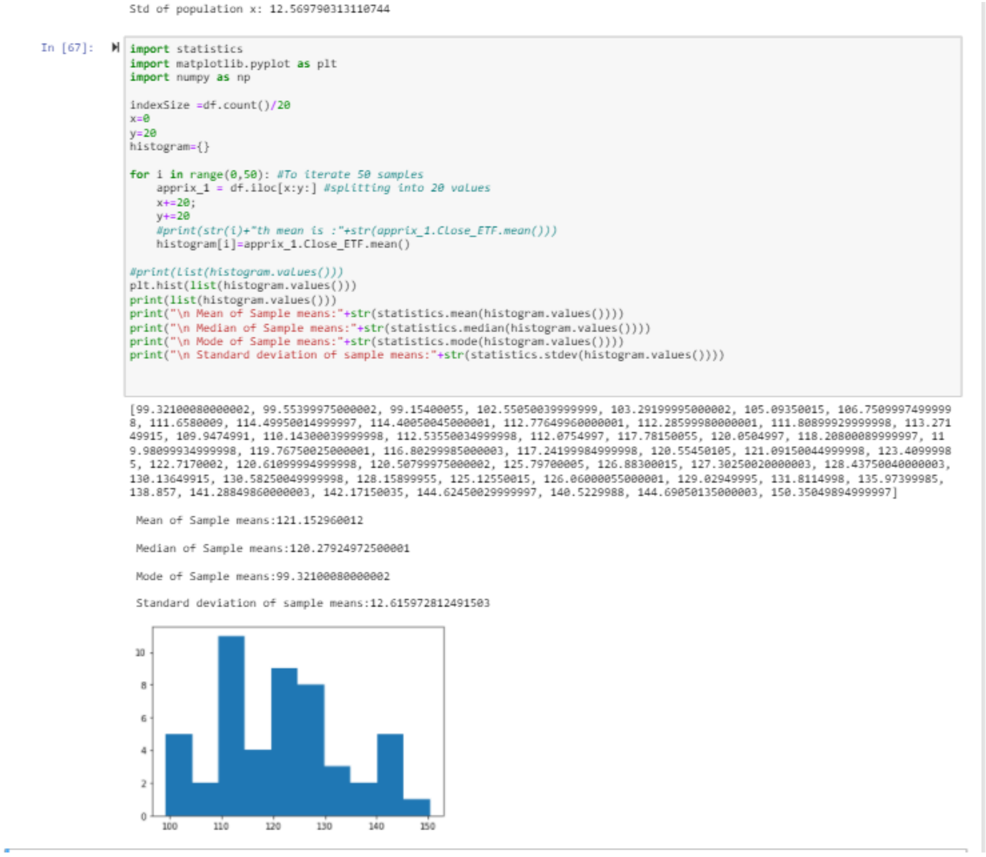
```
In [21]:  import statistics
           import matplotlib.pyplot as plt
           import numpy as np

           indexSize =df.count()/20
           x=0
           y=20

           for i in range(0,50): #To iterate 50 samples
               apprix_1 = df.iloc[x:y:] #splitting into 20 values
               x+=20;
               y+=20
               print(apprix_1.Close_ETF)
```

```
0      97.349998
1      97.750000
2      99.160004
3      99.650002
4      99.260002
5      98.250000
6      99.250000
7     100.300003
8     100.610001
9      99.559998
10    101.660004
11    101.660004
12    101.570000
13    100.019997
14     99.440002
15     98.419998
16     98.519997
17     97.529999
18     98.800003
19     97.660004
Name: Close_ETF, dtype: float64
20     97.629997
```

```
21     98.529999
22     99.769997
23     98.739998
24    100.699997
25    101.150002
26    100.580002
27     99.300003
28    100.239998
29    100.730003
30    100.510002
31     99.919998
32     98.500000
33     99.510002
34     98.279999
35     99.169998
36     99.239998
37     98.489998
38    100.230003
39     99.860001
Name: Close_ETF, dtype: float64
40     99.400002
41     99.160004
42     99.389999
43     98.510002
44     98.510002
45     96.419998
46     96.980003
47     98.000000
48     98.279999
49     98.650002
50     99.550003
51     99.040001
52     99.309998
53     99.620003
54    100.480003
55    100.860001
56    100.449997
57    100.769997
58     99.769997
59     99.930000
```

Interpretation- The above screenshot , is a sample of population having 20 values and 50 samples. Rest of the code can be seen in the appendix.

3) Calculated mean of each group



```
Std of population x: 12.569790313110744
```

```
In [67]:
import statistics
import matplotlib.pyplot as plt
import numpy as np

indexSize =df.count()/20
x=0
y=20
histogram={}

for i in range(0,50): #To iterate 50 samples
    apprix_1 = df.iloc[x:y:] #splitting into 20 values
    x+=20;
    y+=20
    #print(str(i)+"th mean is :"+str(apprix_1.Close_ETF.mean()))
    histogram[i]=apprix_1.Close_ETF.mean()

#print(list(histogram.values()))
plt.hist(list(histogram.values()))
print(list(histogram.values()))
print("\n Mean of Sample means:"+str(statistics.mean(histogram.values())))
print("\n Median of Sample means:"+str(statistics.median(histogram.values())))
print("\n Mode of Sample means:"+str(statistics.mode(histogram.values())))
print("\n Standard deviation of sample means:"+str(statistics.stdev(histogram.values())))
```

```
[99.32100080000002, 99.55399975000002, 99.15400055, 102.55050039999999, 103.29199995000002, 105.09350015, 106.7509997499999
8, 111.6580009, 114.49950014999997, 114.40050045000001, 112.77649960000001, 112.28599980000001, 111.80899929999998, 113.271
49915, 109.9474991, 110.14300039999998, 112.53550034999998, 112.0754997, 117.78150055, 120.0504997, 118.20800089999997, 11
9.98099934999998, 119.76750025000001, 116.80299985000003, 117.24199984999998, 120.55450105, 121.09150044999998, 123.4099998
5, 122.7170002, 120.61099994999998, 120.50799975000002, 125.79700005, 126.88300015, 127.30250020000003, 128.43750040000003,
130.13649915, 130.58250049999998, 128.15899955, 125.12550015, 126.06000055000001, 129.02949995, 131.8114998, 135.97399985,
138.857, 141.28849860000003, 142.17150035, 144.62450029999997, 140.5229988, 144.69050135000003, 150.35049894999997]

 Mean of Sample means:121.152960012

 Median of Sample means:120.27924972500001

 Mode of Sample means:99.32100080000002

 Standard deviation of sample means:12.615972812491503
```

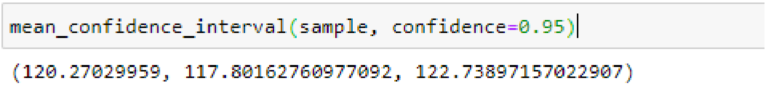
Interpretation

Standard deviation measures the spread of a data distribution. The more spread out a data distribution is, the greater its standard deviation. A standard deviation close to 0 indicates that the data points tend to be close to the mean. The further the data points are from the mean, the greater the standard deviation. The mean of the entire Close_ETF column and the sample means of the same column tend to be the same value. Also, the standard deviation of the sample's mean tends be far from the mean of the sample. As we know that the more spread-out data, the greater the SD we can say that the distribution of data is spread out.

To identify the shape of the distribution histograms are used. According to the output the long tail extends to the right so it indicates that it is right skewed distribution.

PART 5- Construct a confidence interval with your data

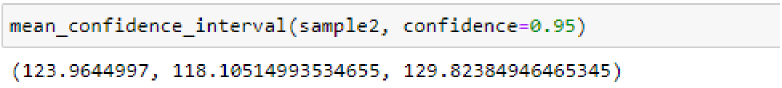
1. 95% confidence interval for one of the 10 simple random samples



```
mean_confidence_interval(sample, confidence=0.95)
```

```
(120.27029959, 117.80162760977092, 122.73897157022907)
```

2. 95% confidence interval for one of the 50 simple random samples



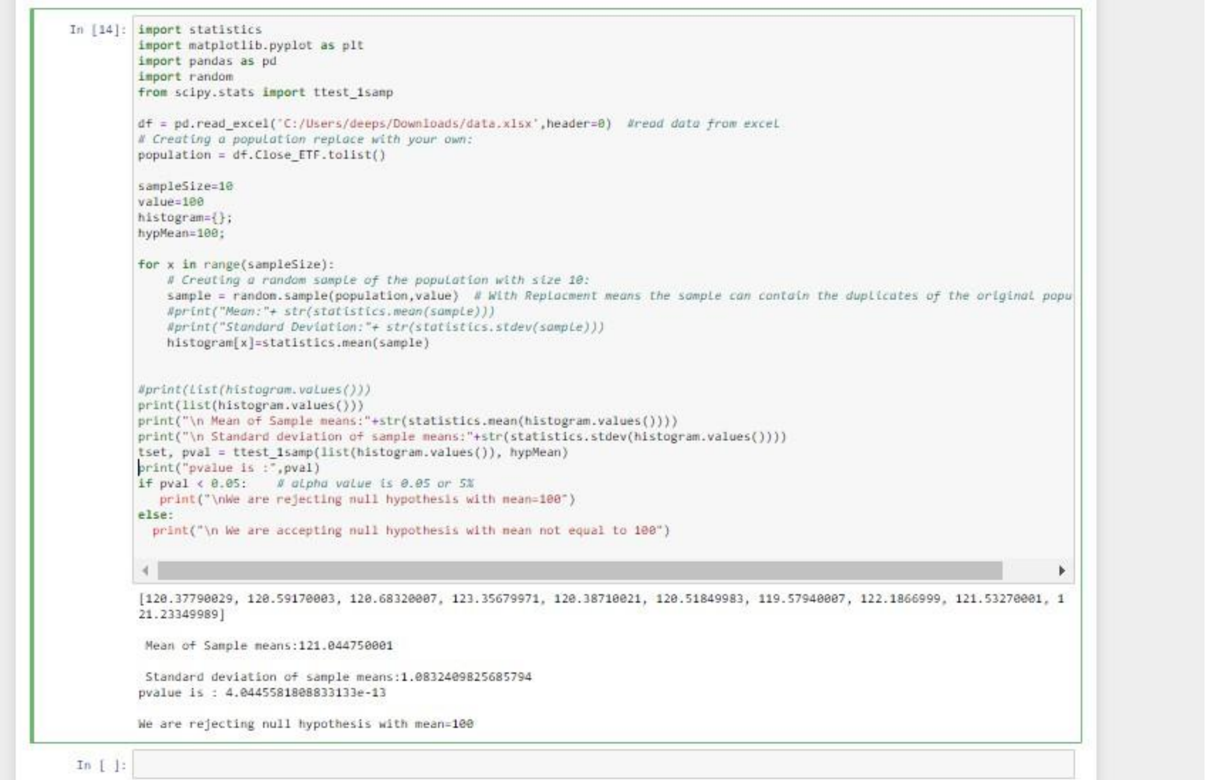
```
mean_confidence_interval(sample2, confidence=0.95)
```

```
(123.9644997, 118.10514993534655, 129.82384946465345)
```

From part-1 we can see that the population mean is 121.152960. The confidence interval for one of the 10 simple random samples (117.80162760977092, 122.73897157022907) and for one of the 50 simple random samples (118.10514993534655, 129.82384946465345) . Hence, the two intervals from (1) and (2) include the true value of the population mean $\boldsymbol{\mu}$ which is 121.152960 as it lies within the confidence interval. The 2nd case includes 50 simple random samples of the population, which is more than 10 simple random samples, and hence, a better representation of the population. So, it will be more accurate. Since the true value lies within the interval for both the samples, we can say that both are accurate in this case.

PART 6

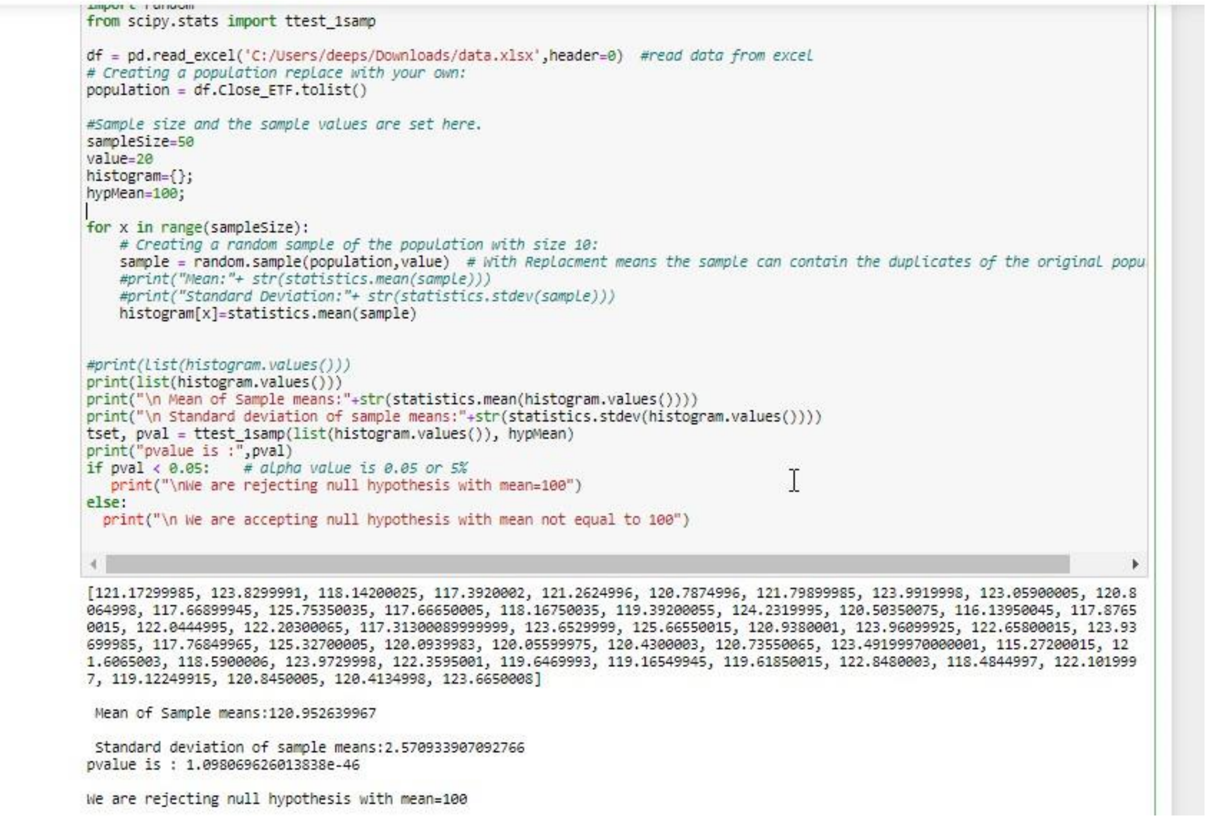
a) Using the same sample to test$\boldsymbol{H0}$:$\boldsymbol{\mu}$=$\mathbf{100}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\mu}$≠$\mathbf{100}$at the significance level 0.05.



```
In [14]: import statistics
         import matplotlib.pyplot as plt
         import pandas as pd
         import random
         from scipy.stats import ttest_1samp

         df = pd.read_excel('C:/Users/deeps/Downloads/data.xlsx',header=0)  #read data from excel
         # Creating a population replace with your own:
         population = df.Close_ETF.tolist()

         sampleSize=10
         value=100
         histogram={};
         hypMean=100;

         for x in range(sampleSize):
             # Creating a random sample of the population with size 10:
             sample = random.sample(population,value)  # With Replacment means the sample can contain the duplicates of the original popu
             #print("Mean:"+ str(statistics.mean(sample)))
             #print("Standard Deviation:"+ str(statistics.stdev(sample)))
             histogram[x]=statistics.mean(sample)


         #print(List(histogram.values()))
         print(list(histogram.values()))
         print("\n Mean of Sample means:"+str(statistics.mean(histogram.values())))
         print("\n Standard deviation of sample means:"+str(statistics.stdev(histogram.values())))
         tset, pval = ttest_1samp(list(histogram.values()), hypMean)
         print("pvalue is :",pval)
         if pval < 0.05:    # alpha value is 0.05 or 5%
            print("\nWe are rejecting null hypothesis with mean=100")
         else:
           print("\n We are accepting null hypothesis with mean not equal to 100")

         [120.37790029, 120.59170003, 120.68320007, 123.35679971, 120.38710021, 120.51849983, 119.57940007, 122.1866999, 121.53270001, 1
         21.23349989]

          Mean of Sample means:121.044750001

          Standard deviation of sample means:1.0832409825685794
         pvalue is : 4.0445581808833133e-13

         We are rejecting null hypothesis with mean=100

In [ ]:
```

Interpretation : As Discussed we have taken the mentioned samples that are random. It has a sample size of 10 and each sample has 100 values . Given that null hypothesis is mean of the samples must be 100 and alternate hypothesis that the mean wont be 100 . From the figure above we can clearly see that the mean of the samples is coming up to 121 , so we need to reject the null hypothesis . So based on the ‘p’ value and the critical value we can see that the code is rejecting the null hypothesis.

b) Using the same sample to test$\boldsymbol{H0}$:$\boldsymbol{\mu}$=$\mathbf{100}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\mu}$≠$\mathbf{100}$at the significance level 0.05.



```
import random
from scipy.stats import ttest_1samp

df = pd.read_excel('C:/Users/deeps/Downloads/data.xlsx',header=0)  #read data from excel
# Creating a population replace with your own:
population = df.Close_ETF.tolist()

#Sample size and the sample values are set here.
sampleSize=50
value=20
histogram={};
hypMean=100;

for x in range(sampleSize):
    # Creating a random sample of the population with size 10:
    sample = random.sample(population,value)  # With Replacment means the sample can contain the duplicates of the original popu
    #print("Mean:"+ str(statistics.mean(sample)))
    #print("Standard Deviation:"+ str(statistics.stdev(sample)))
    histogram[x]=statistics.mean(sample)


#print(list(histogram.values()))
print(list(histogram.values()))
print("\n Mean of Sample means:"+str(statistics.mean(histogram.values())))
print("\n Standard deviation of sample means:"+str(statistics.stdev(histogram.values())))
tset, pval = ttest_1samp(list(histogram.values()), hypMean)
print("pvalue is :",pval)
if pval < 0.05:    # alpha value is 0.05 or 5%
   print("\nWe are rejecting null hypothesis with mean=100")
else:
  print("\n We are accepting null hypothesis with mean not equal to 100")
```

```
[121.17299985, 123.8299991, 118.14200025, 117.3920002, 121.2624996, 120.7874996, 121.79899985, 123.9919998, 123.05900005, 120.8
064998, 117.66899945, 125.75350035, 117.66650005, 118.16750035, 119.39200055, 124.2319995, 120.50350075, 116.13950045, 117.8765
0015, 122.0444995, 122.20300065, 117.31300089999999, 123.6529999, 125.66550015, 120.9380001, 123.96099925, 122.65800015, 123.93
699985, 117.76849965, 125.32700005, 120.0939983, 120.05599975, 120.4300003, 120.73550065, 123.49199970000001, 115.27200015, 12
1.6065003, 118.5900006, 123.9729998, 122.3595001, 119.6469993, 119.16549945, 119.61850015, 122.8480003, 118.4844997, 122.101999
7, 119.12249915, 120.8450005, 120.4134998, 123.6650008]

 Mean of Sample means:120.952639967

 Standard deviation of sample means:2.570933907092766
pvalue is : 1.098069626013838e-46

We are rejecting null hypothesis with mean=100
```

Interpretation: As discussed, we have taken the mentioned samples that are random. It has a sample size of 50 and each sample has 100 values. Given that null hypothesis is mean of the samples must be 20 and alternate hypothesis that the mean will not be 100. From the figure above we can clearly see that the mean of the samples is coming up to 120, so we need to reject the null hypothesis. So based on the ‘p’ value and the critical value we can see that the code is rejecting the null hypothesis.

c) Using the same sample to test$\boldsymbol{H0}$:$\boldsymbol{\sigma}$=$\boldsymbol{15}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\sigma}$≠$\boldsymbol{15}$at the significance level 0.05.

We have used the hypothesis test called Chi-Square test for single variance

The test statistics $\frac{(n-1)s^2}{\sigma^2}$
Here s is the sample standard deviation and sigma is the hypothesized standard deviation

Now if we have n=100

test statistic = 99*11.59832/152= 59.1890

From the chi- square table , value is corresponds to 0.05 and 99 degrees of freedom and should be between 74.222 and 129.561

As the test static is not in that range Hence we ‘’reject the null hypothesis ‘’.

d) Using the same sample to test$\boldsymbol{H0}$:$\boldsymbol{\sigma}$=$\boldsymbol{15}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\sigma}$<$\boldsymbol{15}$at the significance level 0.05.

We have used the hypothesis test called Chi-Square test for single variance

The test statistics $\frac{(n-1)s^2}{\sigma^2}$
Here s is the sample standard deviation and sigma is the hypothesized standard deviation

Now if we have n=20

test statistic = 19*13.9092/152= 16.33.66

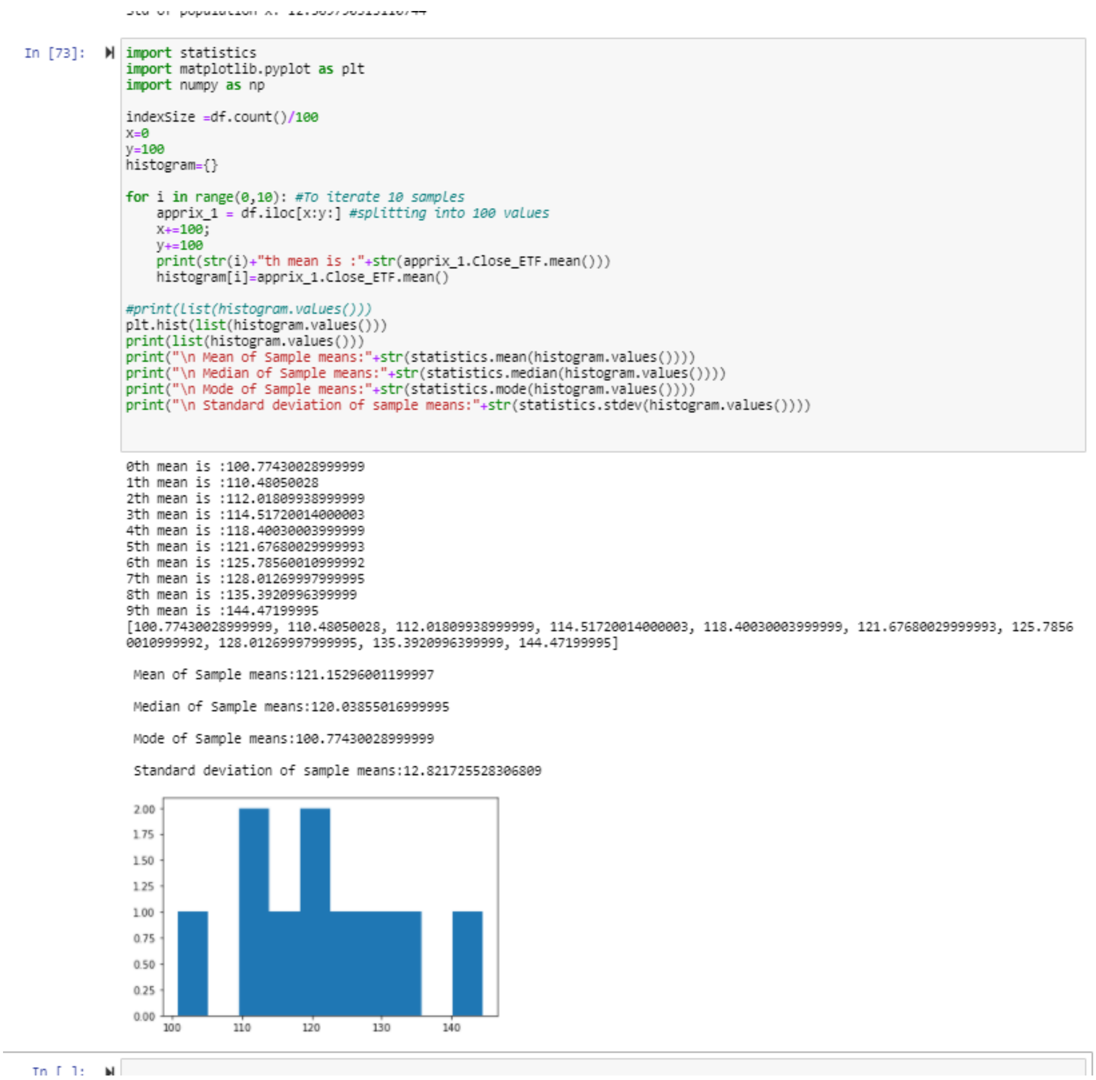
From the chi- square table , value is corresponds to 0.05 and 19 degrees of freedom and should be between 8.907 and 32.852

As the test static is not in that range Hence we ‘’reject the null hypothesis ‘’.

PART 7- Comparing Data To The Different Set

Requirements-

The entire Gold column as a random sample from the first population, and the entire Oil column as a random sample from the second population. Assuming these two samples be drawn independently, form a hypothesis and test it to see if the Gold and Oil have equal means in the significance level 0.05



```
Std of population x: 12.569756515116744
```

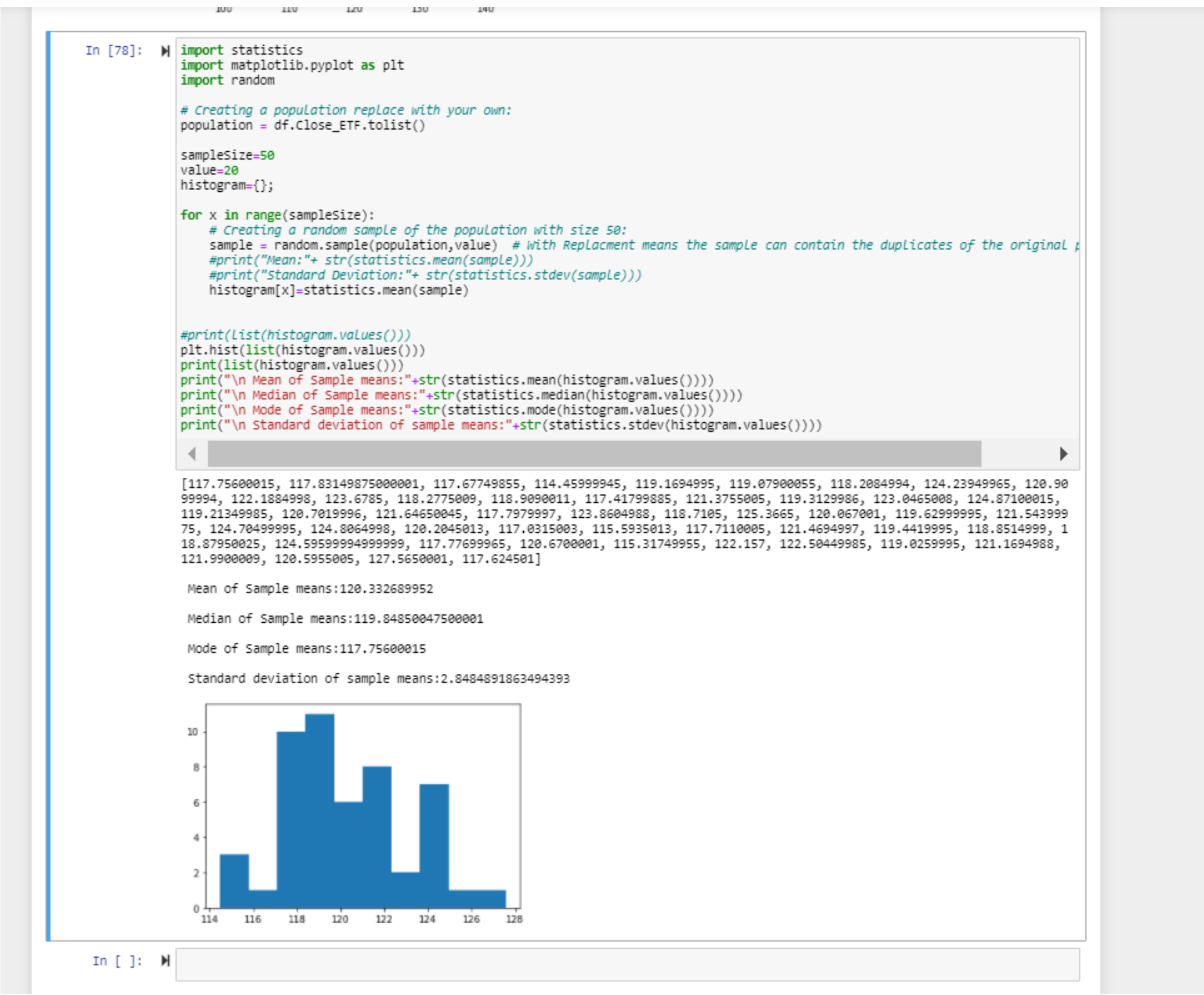
```
In [73]:  import statistics
          import matplotlib.pyplot as plt
          import numpy as np

          indexSize =df.count()/100
          x=0
          y=100
          histogram={}

          for i in range(0,10): #To iterate 10 samples
              apprix_1 = df.iloc[x:y:] #splitting into 100 values
              x+=100;
              y+=100
              print(str(i)+"th mean is :"+str(apprix_1.Close_ETF.mean()))
              histogram[i]=apprix_1.Close_ETF.mean()

          #print(list(histogram.values()))
          plt.hist(list(histogram.values()))
          print(list(histogram.values()))
          print("\n Mean of Sample means:"+str(statistics.mean(histogram.values())))
          print("\n Median of Sample means:"+str(statistics.median(histogram.values())))
          print("\n Mode of Sample means:"+str(statistics.mode(histogram.values())))
          print("\n Standard deviation of sample means:"+str(statistics.stdev(histogram.values())))
```

```
0th mean is :100.77430028999999
1th mean is :110.48050028
2th mean is :112.01809938999999
3th mean is :114.51720014000003
4th mean is :118.40030003999999
5th mean is :121.67680029999993
6th mean is :125.78560010999992
7th mean is :128.01269997999995
8th mean is :135.3920996399999
9th mean is :144.47199995
[100.77430028999999, 110.48050028, 112.01809938999999, 114.51720014000003, 118.40030003999999, 121.67680029999993, 125.7856
0010999992, 128.01269997999995, 135.3920996399999, 144.47199995]

 Mean of Sample means:121.15296001199997

 Median of Sample means:120.03855016999995

 Mode of Sample means:100.77430028999999

 Standard deviation of sample means:12.821725528306809
```

```
In [ ]:
```

We can see that there is maximum occurring at 120 as well has left side of the mid . It is still not totally normal distribution. We can also observe outliers in the histogram occurring at >140 and > 100 values of x axis . We can conclude that the distribution is still right skewed. The sample means and the population means are same and no significant difference is found in their values . But the standard deviations is varied slightly when compared to the previous Sampling size. Yet these are not consistent with the Central Limit theorem as well

-



```
import statistics
import matplotlib.pyplot as plt
import random

# Creating a population replace with your own:
population = df.Close_ETF.tolist()

sampleSize=50
value=20
histogram={};

for x in range(sampleSize):
    # Creating a random sample of the population with size 50:
    sample = random.sample(population,value)  # With Replacment means the sample can contain the duplicates of the original p
    #print("Mean:"+ str(statistics.mean(sample)))
    #print("Standard Deviation:"+ str(statistics.stdev(sample)))
    histogram[x]=statistics.mean(sample)


#print(List(histogram.values()))
plt.hist(list(histogram.values()))
print(list(histogram.values()))
print("\n Mean of Sample means:"+str(statistics.mean(histogram.values())))
print("\n Median of Sample means:"+str(statistics.median(histogram.values())))
print("\n Mode of Sample means:"+str(statistics.mode(histogram.values())))
print("\n Standard deviation of sample means:"+str(statistics.stdev(histogram.values())))
```

```
[117.75600015, 117.83149875000001, 117.67749855, 114.45999945, 119.1694995, 119.07900055, 118.2084994, 124.23949965, 120.90
99994, 122.1884998, 123.6785, 118.2775009, 118.9090011, 117.41799885, 121.3755005, 119.3129986, 123.0465008, 124.87100015,
119.21349985, 120.7019996, 121.64650045, 117.7979997, 123.8604988, 118.7105, 125.3665, 120.067001, 119.62999995, 121.543999
75, 124.70499995, 124.8064998, 120.2045013, 117.0315003, 115.5935013, 117.7110005, 121.4694997, 119.4419995, 118.8514999, 1
18.87950025, 124.59599994999999, 117.77699965, 120.6700001, 115.31749955, 122.157, 122.50449985, 119.0259995, 121.1694988,
121.9900009, 120.5955005, 127.5650001, 117.624501]

 Mean of Sample means:120.332689952

 Median of Sample means:119.84850047500001

 Mode of Sample means:117.75600015

 Standard deviation of sample means:2.8484891863494393
```

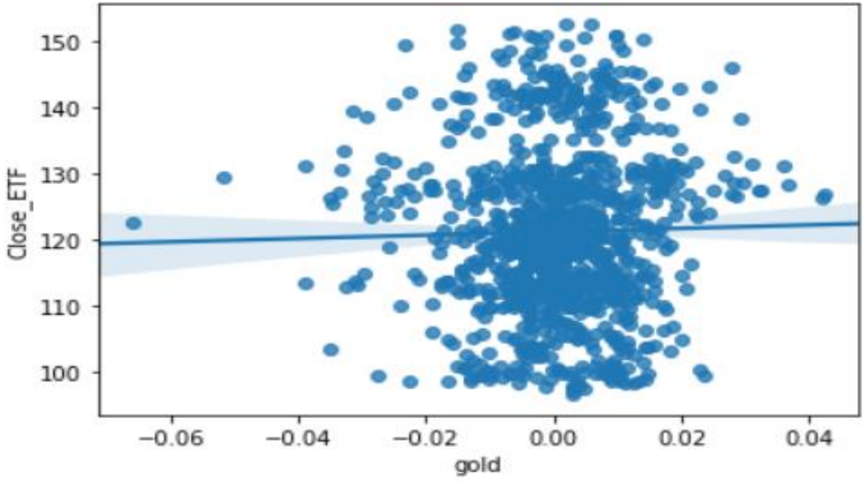
3. Here the sample mean and population means are different, which means sample means are slightly lesser than the population means. For large sample sizes they may be equal. Here the standard deviation is closer to zero and there is quite a difference between the means and standard deviation so the data is well distributed as well. From the histogram it is evident that the above sampling technique doesn't provide a normal distribution still and we still get right skewed distribution. Even in this case central limit theorem isn't applicable.

PART 8- Fitting The Line To The Data

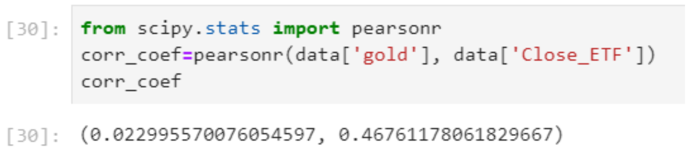
1. The Gold column is taken as the X variable or the independent variable, while the ETF column is taken as the Y variable or the dependent variable.

A scatter plot is drawn taking the above-mentioned variables. A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

The below image shows the scatter plot but it does not show the presence of any linear relationship between the variables because although the points are densely scattered, they are not present on the line. There is no linear relationship that can be observed from the above plot because the points do not form a straight line or a linear pattern.

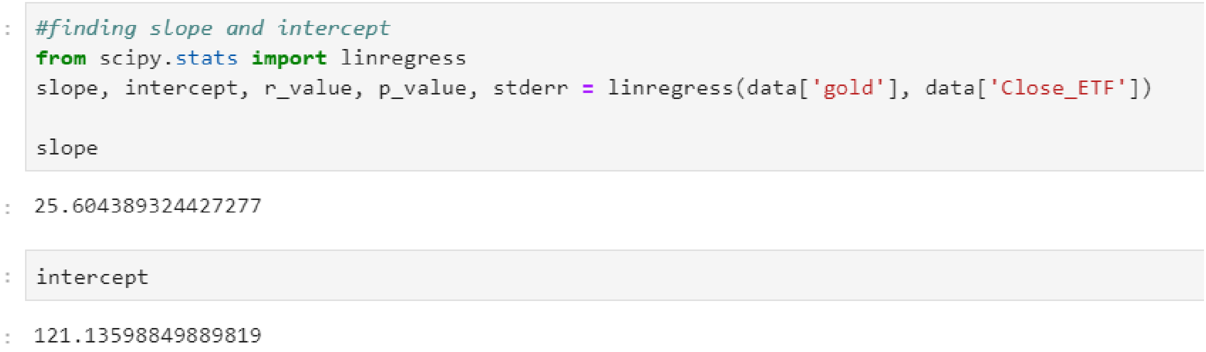


2. The coefficient of correlation, denoted as r, was calculated to be 0.022995570076054597. The range for ‘r’ is -1 to 1. This value obtained explains a weak positive correlation between the variables.



```
[30]: from scipy.stats import pearsonr
      corr_coef=pearsonr(data['gold'], data['Close_ETF'])
      corr_coef

[30]: (0.022995570076054597, 0.46761178061829667)
```

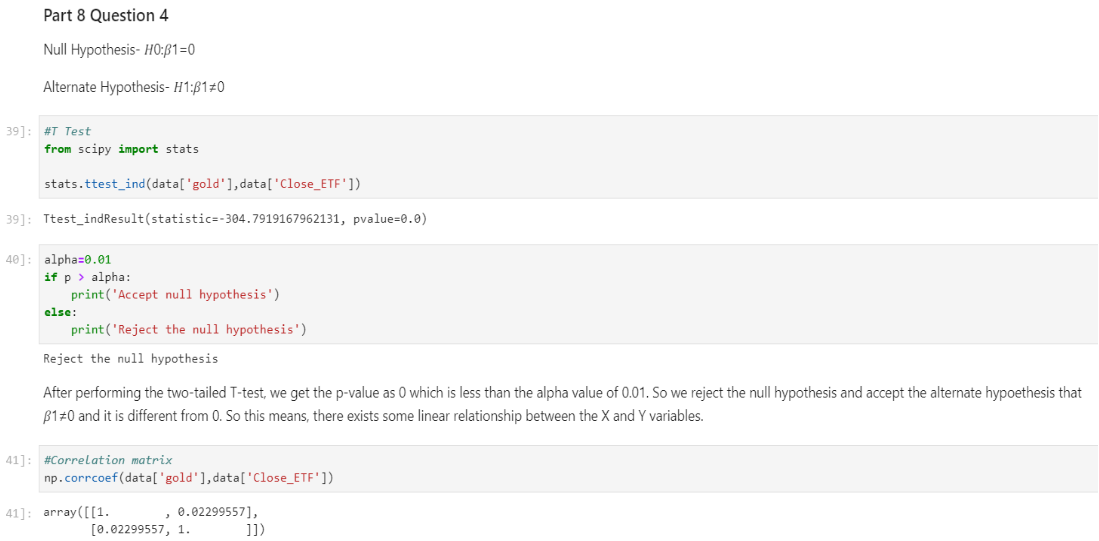
3. We have fit a regression line to the scatter plot, as shown in the image above as a blue line. It’s slope and intercept are 25.604389324427277 and 121.13598849889819 respectively. If you move to the right o the x-axis by one unit, the change in y takes place by 121.13598849889819 units.



```
: #finding slope and intercept
  from scipy.stats import linregress
  slope, intercept, r_value, p_value, stderr = linregress(data['gold'], data['Close_ETF'])

  slope

: 25.604389324427277

: intercept

: 121.13598849889819
```

4. H0: $\beta_1=0$ and H1: $\beta_1\neq0$.
   Here, β1 is the slope of the regression line.
   In order to test the above hypothesis, we use a two-tailed T-Test.

A two-tailed test is a method in which the critical area of a distribution is two-sided and tests whether a sample is greater than or less than a certain range of values. It is used in null-hypothesis testing and testing for statistical significance.
After performing the two-tailed T-test, we get the p-value as 0 which is less than the alpha value of 0.01. So, we reject the null hypothesis and accept the alternate hypothesis that $\beta 1 \neq 0$ and it is different from 0. This means, the linear relationship is significant between the X and Y variables.



Part 8 Question 4

Null Hypothesis- $H0{:}\beta 1=0$

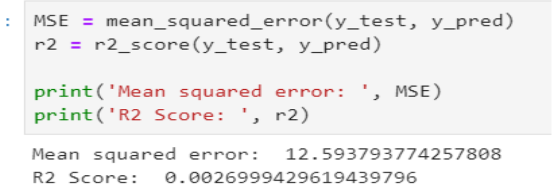
Alternate Hypothesis- $H1{:}\beta 1\neq 0$

```
39]: #T Test
     from scipy import stats

     stats.ttest_ind(data['gold'],data['Close_ETF'])

39]: Ttest_indResult(statistic=-304.7919167962131, pvalue=0.0)

40]: alpha=0.01
     if p > alpha:
         print('Accept null hypothesis')
     else:
         print('Reject the null hypothesis')

     Reject the null hypothesis
```

After performing the two-tailed T-test, we get the p-value as 0 which is less than the alpha value of 0.01. So we reject the null hypothesis and accept the alternate hypoethesis that $\beta 1\neq 0$ and it is different from 0. So this means, there exists some linear relationship between the X and Y variables.

```
41]: #Correlation matrix
     np.corrcoef(data['gold'],data['Close_ETF'])

41]: array([[1.        , 0.02299557],
            [0.02299557, 1.        ]])
```

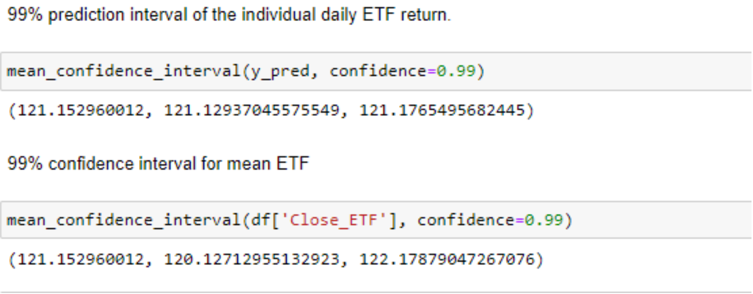
5. On calculating the coefficient of determination, which is R-squared, we get a value of 0.0026999429619439796 which lies between 0 and 1. Since this value is closer to 0, it indicates that the model is not a good fit for the data.



```
: MSE = mean_squared_error(y_test, y_pred)
  r2 = r2_score(y_test, y_pred)

  print('Mean squared error: ', MSE)
  print('R2 Score: ', r2)

  Mean squared error:  12.593793774257808
  R2 Score:  0.0026999429619439796
```

6. The assumption made in order to fit the model was that there existed a linear relationship between the variables, i.e. fitting a linear model is the underlying assumption here.

7. The 99% confidence interval of the mean daily ETF return, and the 99% prediction interval of the individual daily ETF return.



99% prediction interval of the individual daily ETF return.

```
mean_confidence_interval(y_pred, confidence=0.99)

(121.152960012, 121.12937045575549, 121.1765495682445)
```

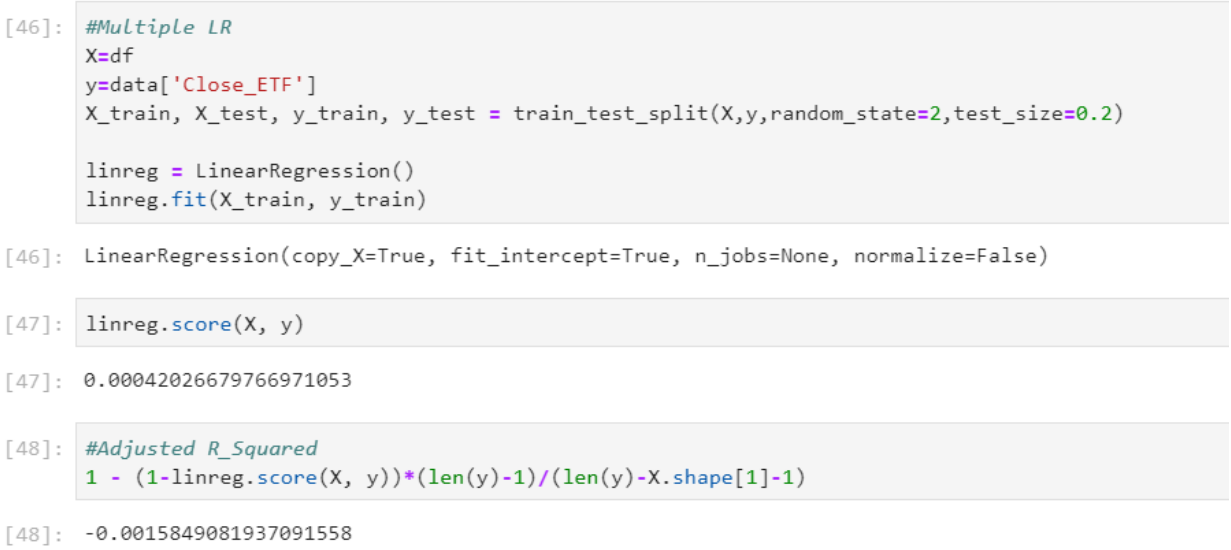
99% confidence interval for mean ETF

```
mean_confidence_interval(df['Close_ETF'], confidence=0.99)

(121.152960012, 120.12712955132923, 122.17879047267076)
```

PART 9- Predicting With The Model

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In this project, we have taken the ETF column as the response variable, and Gold and Oil columns as the independent variables.
We have used machine learning in order to split the data into train and test, in order to fit a multiple linear regression model to the data.



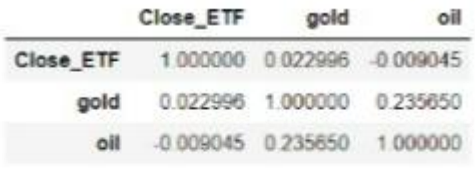
```
[46]: #Multiple LR
      X=df
      y=data['Close_ETF']
      X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=2,test_size=0.2)

      linreg = LinearRegression()
      linreg.fit(X_train, y_train)

[46]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[47]: linreg.score(X, y)

[47]: 0.00042026679766971053

[48]: #Adjusted R_Squared
      1 - (1-linreg.score(X, y))*(len(y)-1)/(len(y)-X.shape[1]-1)

[48]: -0.0015849081937091558
```

Interpretation-

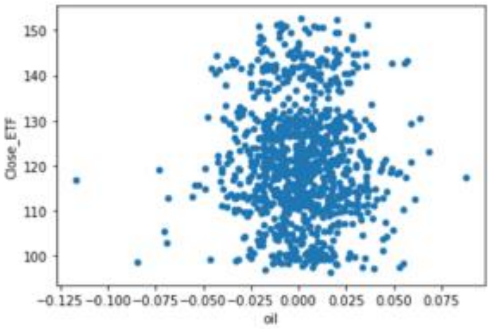
The linear regression accuracy score (shown above) is very close to 0 and the adjusted R_squared value is around -0.00158 which is less than 0. Adjusted r-squared is intended to approximate the actual percentage variance explained. Negative value means that variation in the values around model predictions (SSE) is greater than the total variance (SSTO, which is variation around the mean value). This means that the model is worse than assumption that the mean value is a good prediction.

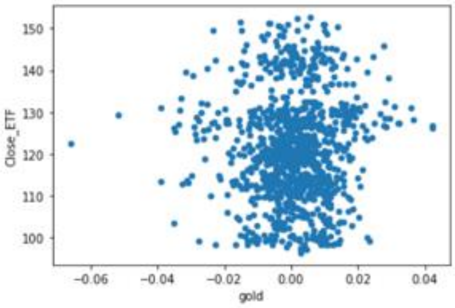
PART 10

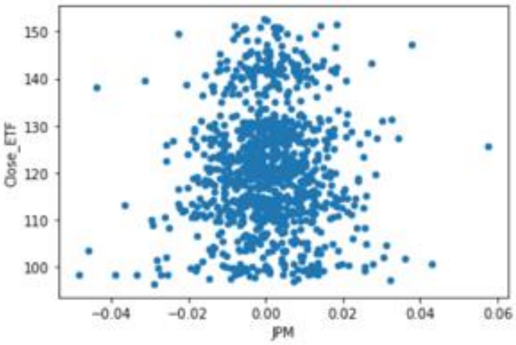
Check the four assumptions made for the error terms of the multiple regression model using these residuals.
Consider Close_ETF ,Oil and Gold column.



	Close_ETF	gold	oil
Close_ETF	1.000000	0.022996	-0.009045
gold	0.022996	1.000000	0.235650
oil	-0.009045	0.235650	1.000000







The above are the scatter plots that we have got from the data. From the above residual table we see that there is linear relationship which exists between both dependent and the independent variables. But we can say that these are weak linear relationships as they do not include the terms of magnitude of co-relation .

We can say that the Oil and ETF are having weak negative linear co-relation.
We also can estimate by the VIF values for the Oil- 1.059952 , Gold- 1.059952

The Histogram of the residuals is not skewed .As it is not skewed , it does not satisfy the linearity assumption.
Multi-collinearity- The VIF values for the multi-linear model if the values are less than 5 only then it satisfies Multi-collinearity
the VIF values for OIL- 1.059952
VIF values for Gold- 1.059952

We have selected features, based on our understanding of the data. To pick the best model, we might needed to pick combination of the features and see if the adjusted R^2 is improving or not for each combination and select the equation which has maximum R^2 value and even the maximum adjusted R^2 value. There is a possibility that both the values are high , there might be an issue with multicollinearity .

SECTION – 3

DISCUSSIONS -

The aim of this project was to perform an exploratory data analysis on the given data, and determine the existence of linear relationship between the variables under study, by performing a linear regression on the data and then evaluating the accuracy of the model, by means of using any statistical tool.

It helped us understand certain key statistical concepts and how to practically apply them: -

1. Like correlation, regression
2. Testing a hypothesis using T-Tests and F-tests/ANOVA
3. Performing normality tests in order to interpret the data distributions
4. Perform sampling techniques, and create confidence intervals
5. Interpreting the relationship between variables
6. Running a linear regression model and then evaluating its goodness of fit on the basis of the *adjusted R-squared* value.

IMPROVEMENTS-

Although the project was very helpful on the whole, we would like to highlight a few areas of improvements that could have been more helpful for us and would have us brainstorm on other concepts.

1. We were asked to plot histograms for the same variables more than once which results in a redundancy

2. We were required to draw samples of different sizes, for variables- Gold, Oil and ETF, and perform tests to compare their means and validate hypotheses, under parts 4 and 6, resulting in redundancy.
3. The project could have included questions where we were required to implement ways to detect outliers and remove them, because all datasets would contain outliers and it is essential to remove them.
4. In certain parts, the questions weren't very clear and explanatory, like under part 4, the questions regarding sampling weren't very clear.

On the whole, the project was basic and included all the basic key concepts, however, it could have been more progressive and analytical.

REFERENCES:

(1)https://machinelearningmastery.com/time-series-data-visualization-with-python/

(2)https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

(3)https://datagy.io/histogram-python/

(4)https://realpython.com/python-histograms/

(5)https://seaborn.pydata.org/

(6)https://pythonspot.com/matplotlib-scatterplot/

(7)https://towardsdatascience.com/data-normalization-with-python-scikit-learn-e9c5640fed58

(8)https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

(9)https://towardsdatascience.com/normality-tests-in-python-31e04aa4f411

(10)https://www.statology.org/left-skewed-vs-right-skewed/

APPENDIX

CODE :

Final (1).ipynb

Final(2).ipynb

Final3.txt