Exploratory Data Analysis and Linear Regression of Financial Variables

A term project report submitted for

MA 541 - Statistical Methods

by

Ujwala Kamineni
Lakshmi Manaswini Guntaka
Rasagna Kusuma
Dharani Aerla

Under the guidance of

Prof. HONG DO
Stevens Institute of Technology

SECTION 1

ABSTRACT

This project report delves into exploratory data analysis (EDA) and linear regression analysis of financial variables including daily ETF returns, daily relative changes in crude oil and gold prices, and the daily returns of JPMorgan Chase & Co stock. Utilizing a dataset of 1000 observations, the study aims to uncover linear relationships between these variables and predict ETF returns based on changes in crude oil and gold prices

INTRODUCTION:

This project analyzes four variables in a dataset. These are daily ETF returns, crude oil price changes, gold price changes, and JPMorgan Chase stock returns. The sample size is 1000 observations. We aim to explore if these variables have linear relationships. We'll use linear regression modeling. A dependent variable will link to one or more independent variables. We'll assess our model's accuracy via statistical tools. Python will be used for analysis.

OBJECTIVE

Linear regression is a statistical method used to model the relationship between a scalar response (dependent variable) and one or more explanatory variables (independent variables). This approach is employed to predict the value of the dependent variable based on the values of the independent variables, establishing a linear relationship between them.

SECTION 2

PART 1- Meet Your Data

The dataset for this project includes four variables: daily returns on an ETF, daily relative changes in crude oil prices, daily relative changes in gold prices, and daily returns of JPMorgan Chase & Co stock. The sample size is 1000 observations. The primary goal of this project is to conduct exploratory data analysis (EDA) on these data points to identify any linear relationships among the variables. To achieve this, linear regression analysis will be carried out. After establishing the regression model, its accuracy will be evaluated using statistical tools. Python has been chosen as the programming language to perform all necessary statistical analyses for this project. The analysis will also include examining the correlations among the variables.

	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

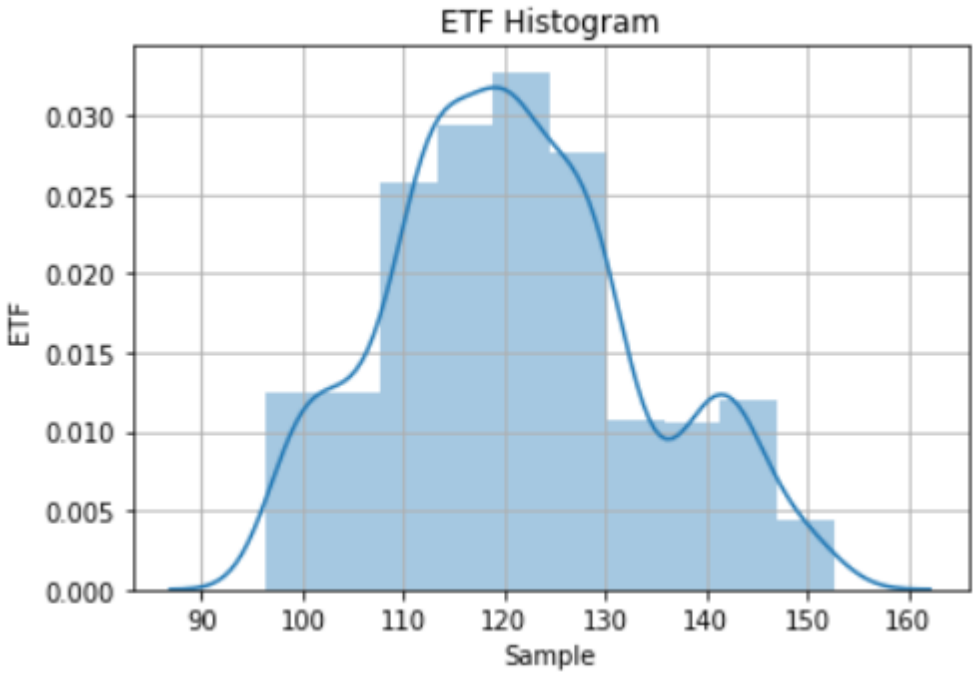
		oil	gold	jpm
Close_ETF		-0.009	0.023	0.037
		gold	jpm	
oil		0.236	-0.121	
		jpm		
gold		0.1		

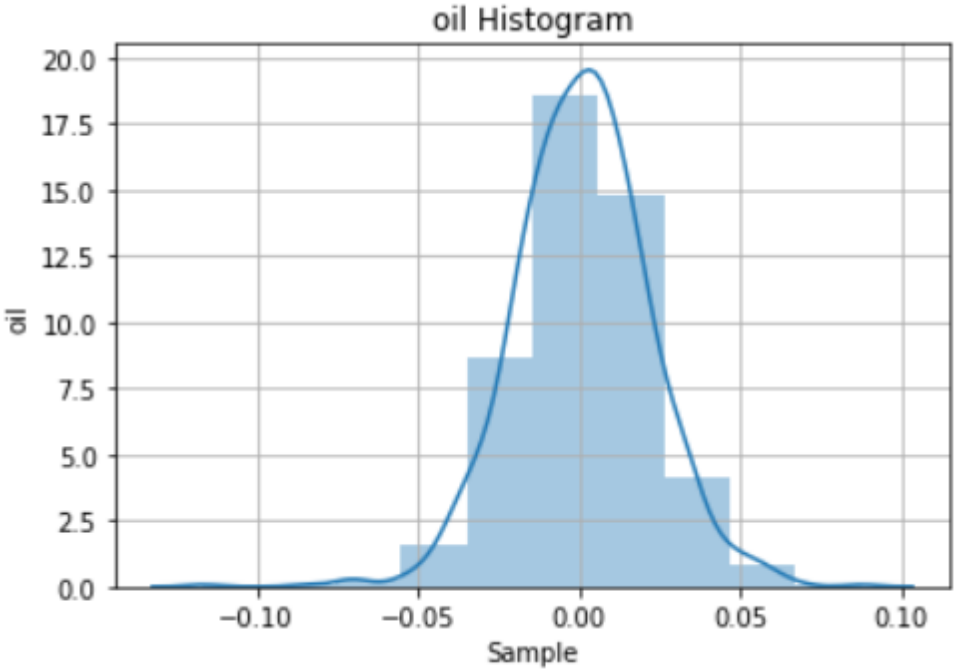
The data analysis reveals a weak negative correlation between the ETF closing prices and oil prices, as well as between oil prices and JPMorgan Chase & Co stock returns. Conversely, a weak positive correlation is observed between the ETF closing prices and gold prices, the ETF closing prices and JPMorgan Chase & Co stock returns, between oil prices and gold prices, and between gold prices and JPMorgan Chase & Co stock returns.

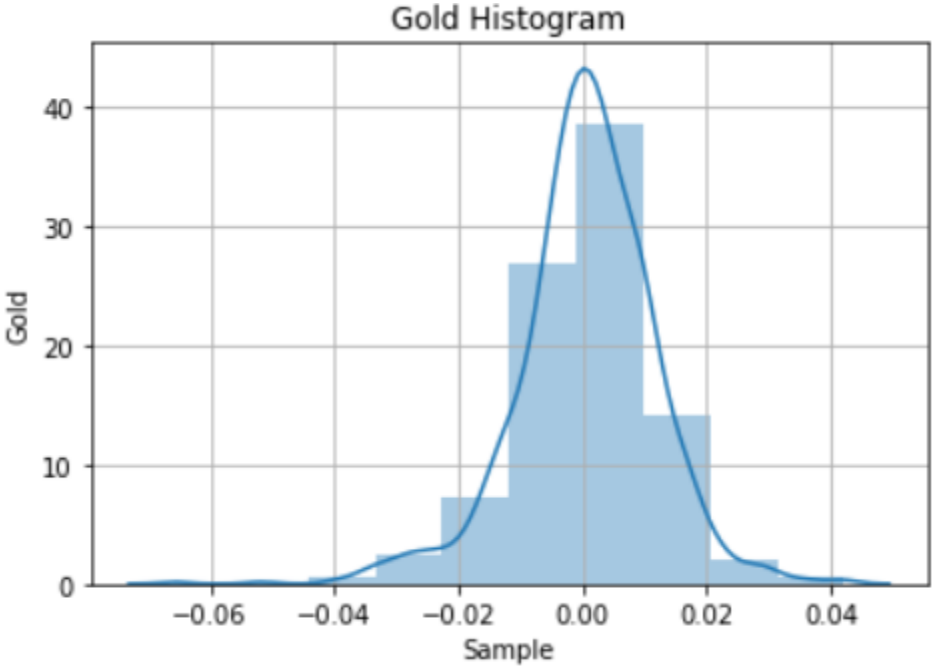
PART 2- Describe Your Data

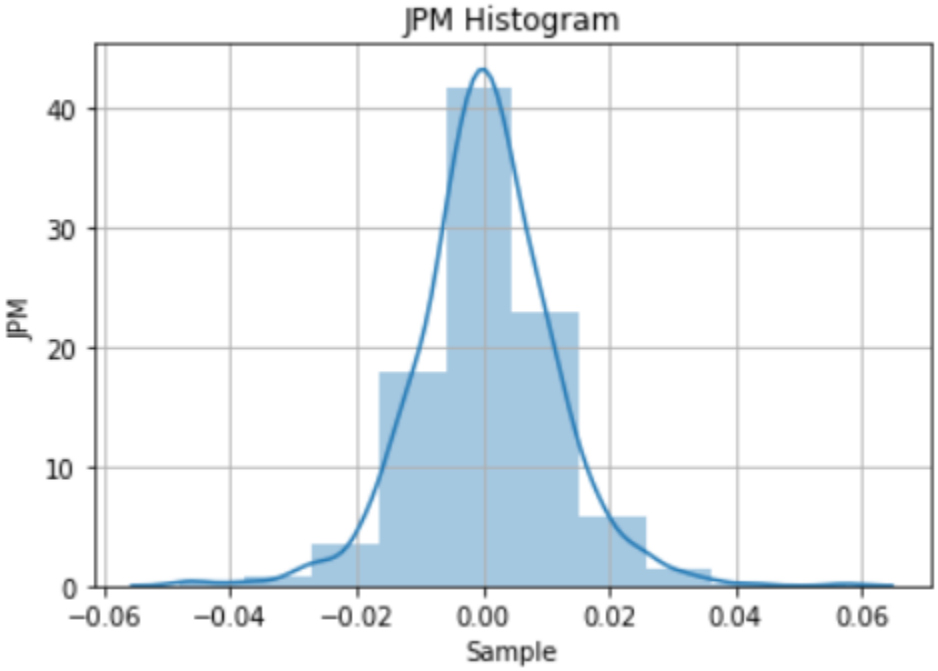
1. A Histogram for each Column:

To analyze the distribution of the data, we will create a histogram for each variable using the Python libraries seaborn and matplotlib. Histograms provide a visual approximation of the distribution of numerical data, helping to identify patterns such as skewness, modality, and outliers within each variable.




oil Histogram
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
oil
−0.10
−0.05
0.00
0.05
0.10
Sample


Gold Histogram
40
30
20
10
0
Gold
−0.06
−0.04
−0.02
0.00
0.02
0.04
Sample



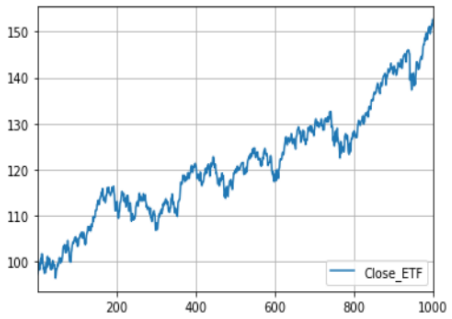
2. Time series plot for each column:

A time-series plot is a univariate graphical representation, displaying only one variable. It is organized as a two-dimensional chart where one axis, typically the horizontal one, represents time marked at suitable intervals (such as seconds, minutes, weeks, quarters, or years). The other axis, usually vertical, displays the numerical values of the variable being observed.

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')
```

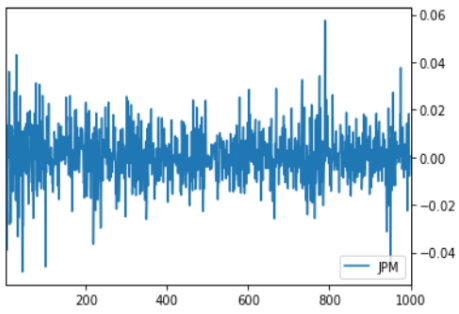
Out[33]: <matplotlib.legend.Legend at 0x14b5517c640>



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>

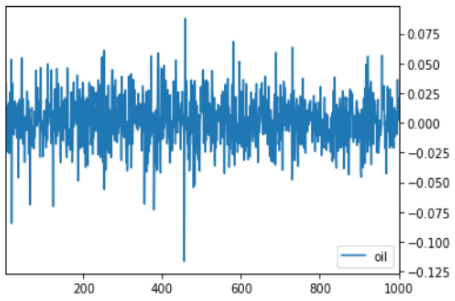


---

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')
```

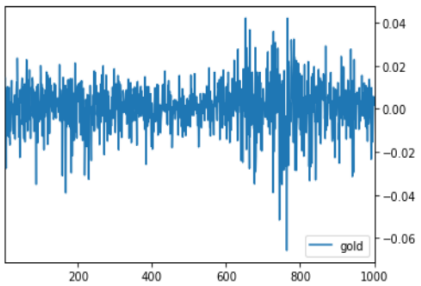
Out[35]: <matplotlib.legend.Legend at 0x14b553de280>



4)Gold

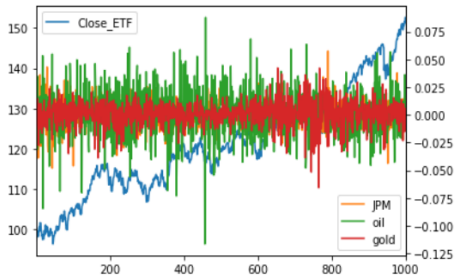
In [36]:
```
import pandas as pd
import matplotlib.pyplot as plt

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>
```

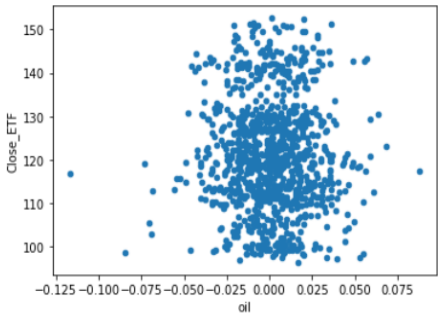



```
In [ ]:
```

2.d) Scatter Plots

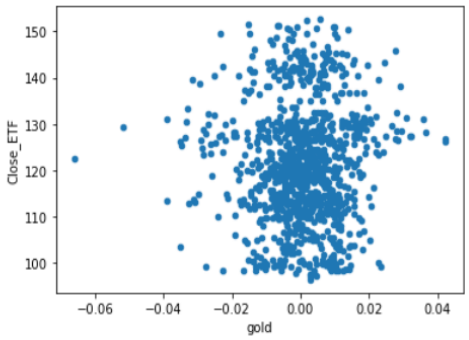
A scatter plot is a type of graph that uses dots to visualize the values for two distinct numerical variables. Each dot on the plot represents an individual data point, with its position along the horizontal and vertical axes indicating the values for these variables. Scatter plots are commonly used to explore and illustrate the relationships between variables. For example, analyzing the relationship between oil prices and ETF closing prices would involve plotting oil price values on one axis and ETF values on the other.

```
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'>
```



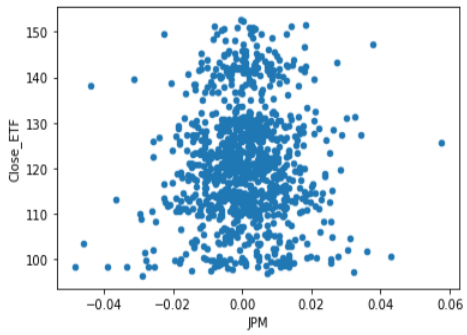
Relationship between gold and Close_ETF

```
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

The scatter plots presented above each demonstrate a positive correlation between the ETF returns and the other independent variables. In these plots, the data points are densely clustered, indicating a consistent relationship across all variables observed.

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 confirm the hypotheses stated, various normality tests were employed, including the Shapiro-Wilk Test, Anderson-Darling Test, Kolmogorov-Smirnov Test, and the QQ Plot. Each of these tests was applied to all variables to evaluate the normality of the dataset.

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 from each normality test are 0, which is below the significance level of 0.05, leading us to reject the null hypothesis that the data distributions are normal for each variable. Contrarily, the histograms created earlier depict a bell-shaped curve, suggesting a normal distribution.

It is important to note that a small p-value suggests inconsistencies with the null hypothesis. In this scenario, the low p-values are attributed to the large sample size, which comprises the entire population for each variable. With such a large sample size, the tests may detect even minor deviations from normality as statistically significant, resulting in very small p-values. Therefore, while the tests suggest non-normality, the actual distribution of the data may still be essentially normal when considering the influence of the large sample size.

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.

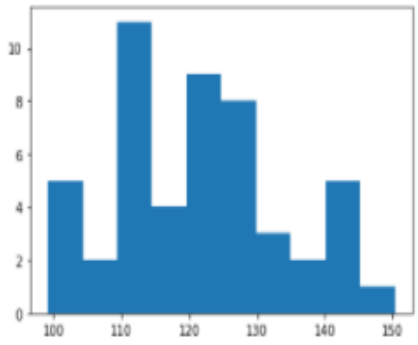
3) Calculated mean of each group

```
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, 106.75099974999998, 111.6580009, 114.49950014999997, 114.40050045000001, 112.77649960000001, 112.28599980000001, 111.80899929999998, 113.27149915, 109.9474991, 110.14300039999998, 112.53550034999998, 112.0754997, 117.78150055, 120.0504997, 118.20800089999997, 119.98099934999998, 119.76750025000001, 116.80299985000003, 117.24199984999998, 120.55450105, 121.09150044999998, 123.40999985, 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 is a statistical measure that quantifies the dispersion of data points in a distribution. A larger standard deviation indicates a wider spread of data around the mean, while a smaller standard deviation suggests that the data points are clustered closely around the

mean. The mean of the entire "Close_ETF" dataset and the means of individual samples from this column are typically similar. However, the standard deviation of these sample means is often quite different from the mean of the samples, indicating a spread-out data distribution.

To visually assess the distribution of data, histograms are commonly used. In this case, the histogram of the "Close_ETF" data shows a long tail extending to the right, signifying a right-skewed distribution. This skewness indicates that while most data points are grouped towards the left of the mean, a few values stretch far to the right.

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 the initial analysis, the population mean is identified as 121.152960. The confidence interval for a subset of 10 simple random samples is calculated to be between 117.801627 and 122.738972, and for a subset of 50 simple random samples, it ranges from 118.105150 to 129.823849. Both these intervals include the true population mean of 121.152960, indicating that the mean value is encompassed within both confidence intervals.

The second scenario, involving 50 simple random samples, provides a broader sample base compared to the first scenario with just 10 samples, making it a more representative and thus potentially more accurate reflection of the entire population. Given that the true population mean falls within both intervals, we can conclude that each set of samples effectively captures the actual mean, affirming the accuracy of the estimates in both cases.

PART 6

a) Using the same sample to test $H0:\mu=100$ vs. $Ha:\mu\neq100$ 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 previously discussed, we have selected random samples with each sample containing 100 values and a total of 10 samples considered. The null hypothesis posited is that the mean of these samples should be 100, whereas the alternative hypothesis suggests that the mean will not be 100. From the analysis presented in the figure above, it is evident that the mean of the samples is approximately 121, which necessitates the rejection of the null hypothesis. This decision is supported by both the 'p' value and the critical value derived from the analysis, confirming the rejection of 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.

```
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$\boldsymbol{H0}$:$\boldsymbol{\sigma}$=$\mathbf{15}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\sigma}$≠$\mathbf{15}$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$\boldsymbol{H0}$:$\boldsymbol{\sigma}$=$\mathbf{15}$vs.$\boldsymbol{Ha}$:$\boldsymbol{\sigma}$<$\mathbf{15}$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-

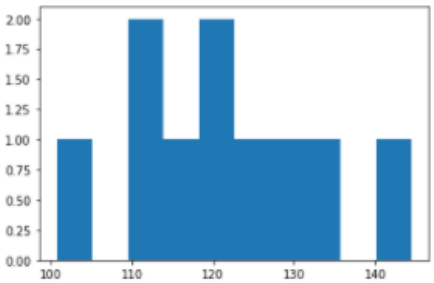
We consider the entire column of Gold prices as a random sample from one population and the entire column of Oil prices as a random sample from another population. Assuming these samples are drawn independently, we set up a hypothesis to test whether the means of Gold and Oil prices are equal.

The null hypothesis (H0) is that the mean of the Gold prices is equal to the mean of the Oil prices. The alternative hypothesis (H1) is that the means are not equal. This hypothesis will be tested at a significance level of 0.05.

Using statistical tests such as the t-test for independent samples, we can assess whether to accept or reject the null hypothesis based on the p-value compared to the 0.05 significance level. If the p-value is less than 0.05, we reject the null hypothesis, suggesting a significant difference in the means of the Gold and Oil prices.

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 [ ]:

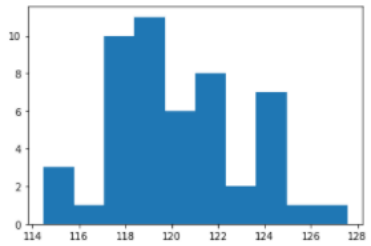
In the analysis, the most frequent occurrence is observed at a value of 120, which is slightly to the left of the midpoint. Despite this, the distribution does

not conform perfectly to a normal distribution. Outliers are evident in the histogram, particularly for values greater than 140 and below 100 on the x-axis, confirming that the distribution remains right-skewed.

The means of the samples and the population are observed to be similar, indicating no significant differences between them. However, there is a slight variation in the standard deviations compared to those from previous sample sizes. Despite these findings, the observed data do not fully align with the Central Limit Theorem, which suggests discrepancies in the theoretical expectations of sample distributions as sample sizes increase.

```
In [78]:
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
```




```
In [ ]:
```

3. In this analysis, the sample means differ from the population mean, with the sample means being slightly lower than the population mean. It's possible

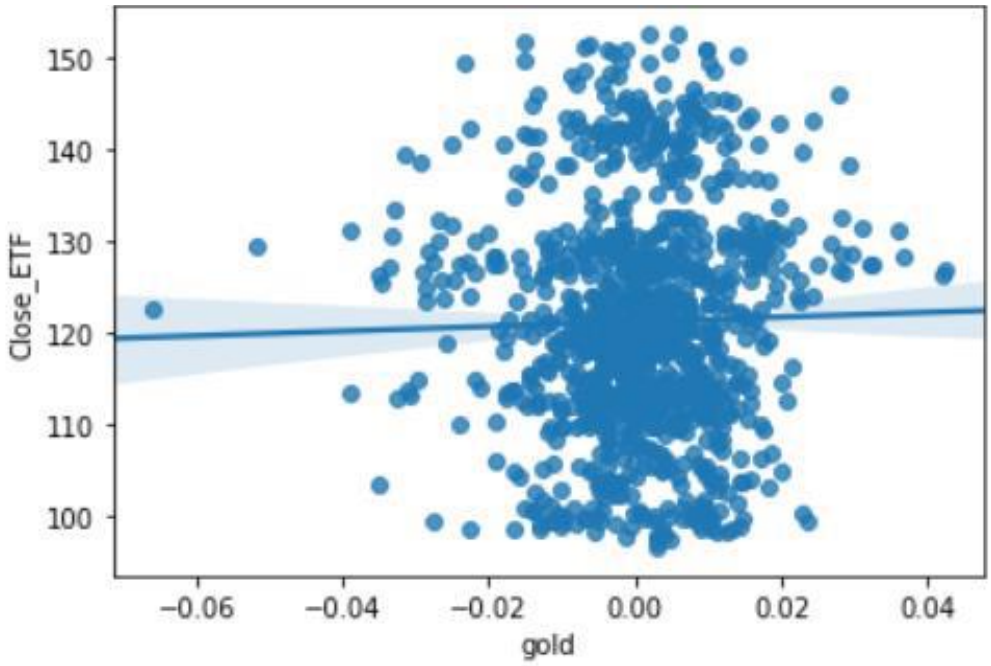
that with larger sample sizes, the sample means could converge to the population mean. The standard deviation is close to zero, indicating a tighter clustering of data points around the mean, and there is a noticeable difference between the mean and standard deviation, suggesting a well-distributed dataset.

From the histogram, it is clear that the sampling technique used does not yield a normal distribution; instead, the data continues to exhibit a right-skewed distribution. This skewness persists despite the sampling method, indicating that even in this scenario, the Central Limit Theorem does not apply, as the theorem typically predicts a normal distribution for large enough sample sizes.

PART 8- Fitting The Line To The Data

1 . In the analysis, the Gold prices are designated as the independent variable (X variable), and the ETF returns are considered the dependent variable (Y variable). A scatter plot is created using these variables, utilizing Cartesian coordinates to visualize the data for the two variables.

The resulting scatter plot, as depicted below, demonstrates a dense scattering of points. However, it fails to indicate the presence of a linear relationship between Gold and ETF returns. The lack of alignment along a straight line or any discernible linear pattern among the data points suggests that no linear relationship exists between these variables based on the scatter plot.



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, 0.467611780618296 67)
```

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: $\beta 1\neq 0$. 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 statistical method where the critical areas of a distribution are split on both sides, allowing for the testing of whether a sample's values are significantly greater than or less than a certain set of values. This method is commonly used in hypothesis testing to assess statistical significance. After conducting a two-tailed T-test, we find that the p-value is 0, which falls below the significance level (alpha) of 0.01. Therefore, we reject the null hypothesis and accept the alternative hypothesis that $\beta 1 \neq 0$, indicating that it significantly differs from 0. This result implies that there is a significant linear relationship between the X and Y variables.

Part 8 Question 4

Null Hypothesis- $H0{:}\beta1=0$

Alternate Hypothesis- $H1{:}\beta1\neq0$

```
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 $\beta1\neq0$ 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.        ]])
```

1. 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
```

2. 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.
3. 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), often referred to as multiple regression, is a statistical method that employs multiple explanatory variables to predict the outcome of a response variable. The primary objective of MLR is to establish a linear relationship between the independent (explanatory) variables and the dependent (response) variable. In this project, the ETF returns are designated as the dependent variable, while the Gold and Oil price changes are treated as the independent variables. We have applied machine learning techniques to divide the dataset into training and testing subsets. This allows us to effectively fit a multiple linear regression model to the data and analyze the relationships within.

[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
```

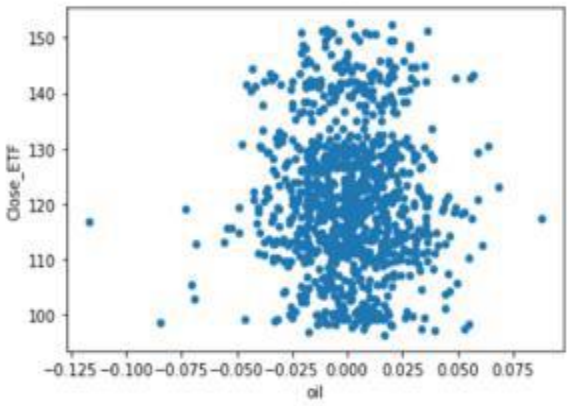
Interpretation-

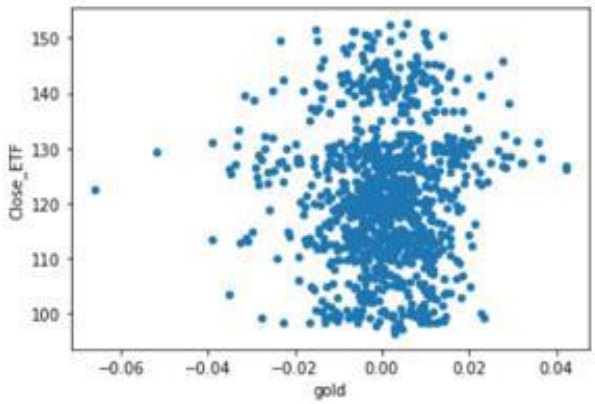
The accuracy score of the linear regression model, as indicated above, is very close to 0, and the adjusted R-squared value is approximately -0.00158, which falls below 0. The adjusted R-squared is a metric used to estimate the proportion of variance in the dependent variable explained by the independent variables, adjusted for the number of predictors. A negative adjusted R-squared indicates that the variation in the data around the model's predictions (Sum of Squares of Errors, SSE) exceeds the total variance in the dependent variable (Total Sum of Squares, SSTO, which is the variation around the mean). This suggests that the model performs worse than a simple model that uses the mean of the dependent variable as a prediction. This indicates the need for reevaluation or enhancement of the model to better capture the underlying relationships in the data.

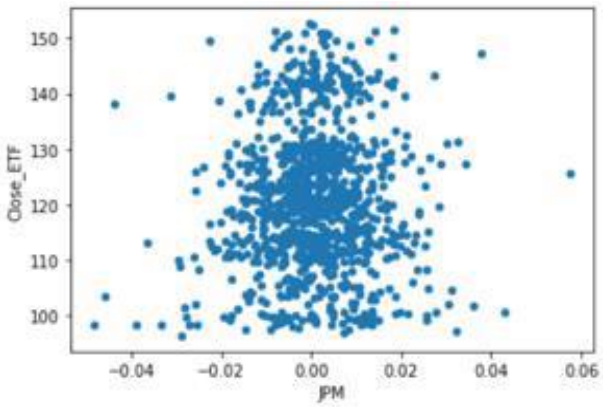
PART 10

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 scatter plots derived from the data indicate the presence of linear relationships between the dependent and independent variables. However, these relationships are weak as they do not consider the magnitude of correlation. Specifically, a weak negative linear correlation exists between the Oil prices and ETF returns. Additionally, the Variance Inflation Factor (VIF) values for Oil and Gold are both calculated at 1.059952, suggesting minimal multicollinearity.

The histogram of the residuals shows no skewness, which is a concern as it challenges the linearity assumption necessary for proper regression analysis. Regarding multi collinearity in the multiple linear regression model, the VIF values are indicative; values below 5 generally suggest acceptable levels of multi collinearity, which is confirmed here with both Oil and Gold having VIF values of 1.059952.

To refine the model selection, features were chosen based on a detailed understanding of the dataset. The best model likely requires a combination of features to be tested to see whether the adjusted R-squared improves with each configuration. The goal is to identify the model with the highest R-squared and adjusted R-squared values. High values in both might still signal an underlying issue with multi collinearity, necessitating further investigation.

SECTION – 3

DISCUSSIONS –

The objective of this project was to conduct exploratory data analysis on the provided dataset to explore potential linear relationships among the variables by performing linear regression and assessing the model's accuracy with statistical tools. This endeavor enhanced our understanding of several fundamental statistical concepts and their practical applications, including:

1. Understanding and applying correlation and regression analyses.
2. Utilizing T-Tests and ANOVA for hypothesis testing.
3. Conducting normality tests to evaluate the distribution of data.
4. Employing various sampling techniques and constructing confidence intervals.
5. Analyzing and interpreting the relationships between different variables.
6. Implementing linear regression models and assessing their effectiveness through the adjusted R-squared value.
7. These activities helped deepen our grasp of statistical methodologies and their relevance in analyzing real-world data.

IMPROVEMENTS-

Overall, the project was quite informative, but there are a few areas where improvements could enhance our learning experience and prompt further exploration of statistical concepts:

- Redundancy in Plotting: We noted redundancy in plotting histograms for the same variables multiple times. Streamlining these tasks to avoid repetition could save time and focus our efforts on more varied analyses.
- Sampling Redundancy: The requirement to draw samples of different sizes for variables like Gold, Oil, and ETF and perform repeated tests to compare their means and validate hypotheses led to redundancy. Consolidating these tasks or varying the analysis could provide deeper insights.
- Outlier Detection and Handling: The project could benefit from including exercises on detecting and handling outliers, as managing outliers is crucial for accurate data analysis. This addition would better prepare us to deal with real-world data complexities.
- Clarity of Instructions: Some parts of the project, such as the sampling questions in part 4, lacked clarity. More detailed instructions or examples could help in understanding the requirements and objectives more effectively.

While the project covered all basic key concepts well, incorporating these improvements could make it more progressive and analytically challenging, enhancing our understanding and application of statistical methods in data analysis.

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/