Importing the Libraries

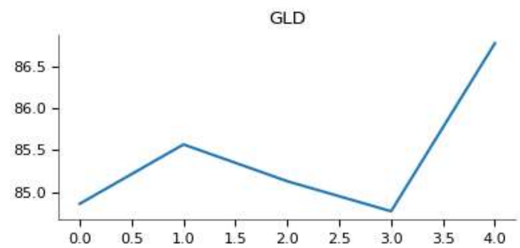
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

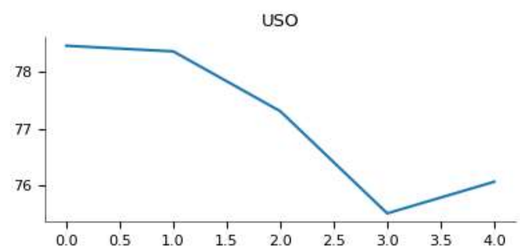
Data Collection and Processing

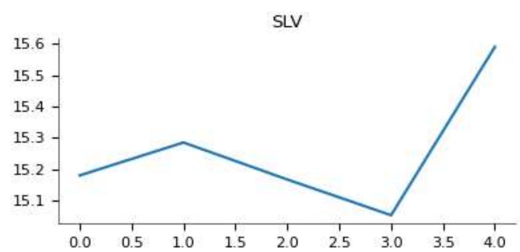
```
# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('/content/gld_price_data.csv')
# print first 5 rows in the dataframe
gold_data.head()
```

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

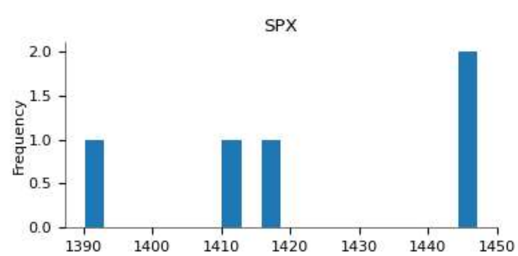
Values

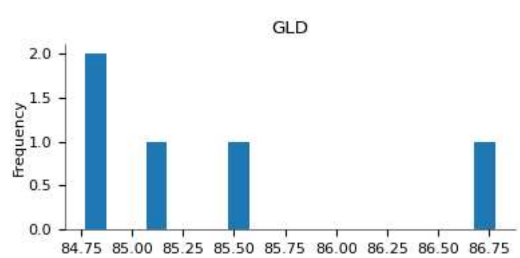


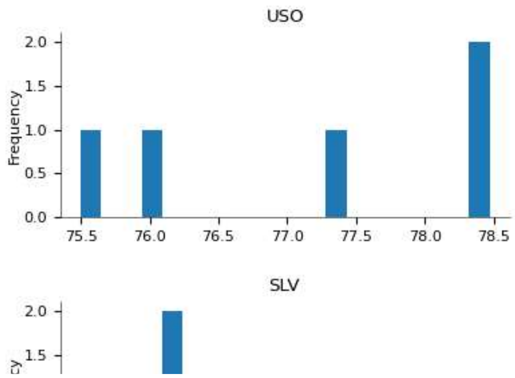




Distributions



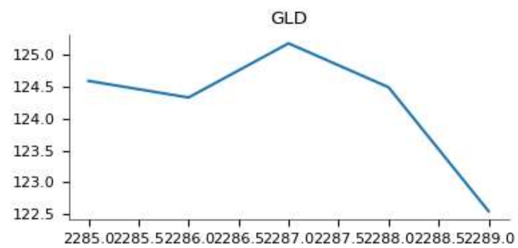


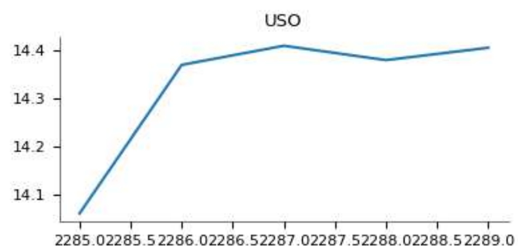


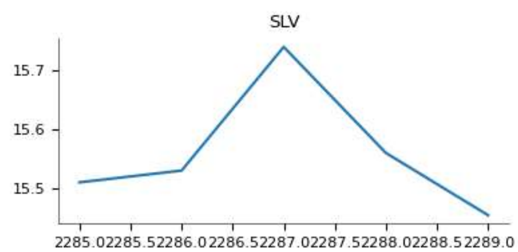

```
# print last 5 rows of the dataframe
gold_data.tail()
```

	Date	SPX	GLD	USO	SLV	EUR/USD
2285	5/8/2018	2671.919922	124.589996	14.0600	15.5100	1.186789
2286	5/9/2018	2697.790039	124.330002	14.3700	15.5300	1.184722
2287	5/10/2018	2723.070068	125.180000	14.4100	15.7400	1.191753
2288	5/14/2018	2730.129883	124.489998	14.3800	15.5600	1.193118
2289	5/16/2018	2725.780029	122.543800	14.4058	15.4542	1.182033

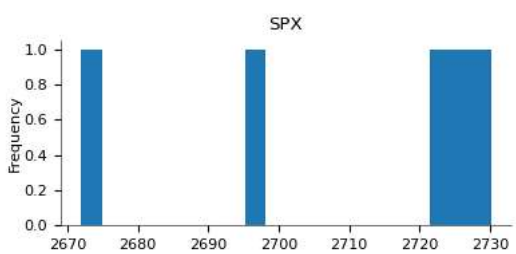
Values

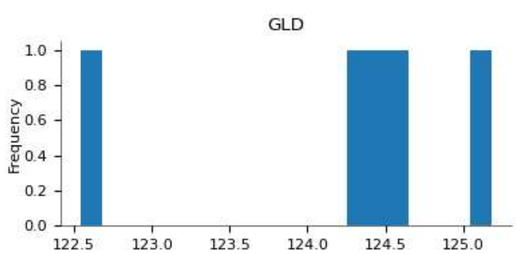


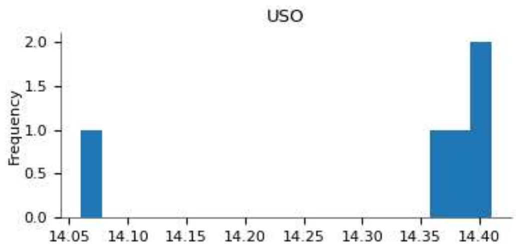




Distributions








```
# number of rows and columns
gold_data.shape
```

```
(2290, 6)
```

```
# getting some basic informations about the data
gold_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     2290 non-null   object
 1   SPX      2290 non-null   float64
 2   GLD      2290 non-null   float64
 3   USO      2290 non-null   float64
 4   SLV      2290 non-null   float64
 5   EUR/USD  2290 non-null   float64
dtypes: float64(5), object(1)
memory usage: 107.5+ KB
```

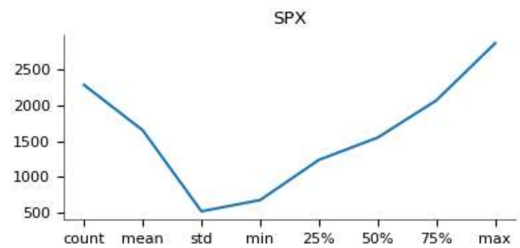
```
# checking the number of missing values
gold_data.isnull().sum()
```

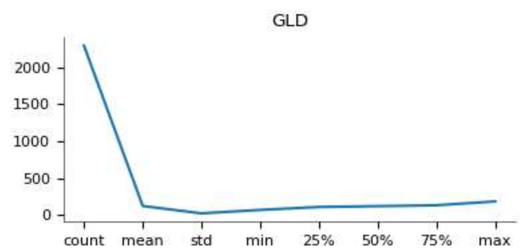
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

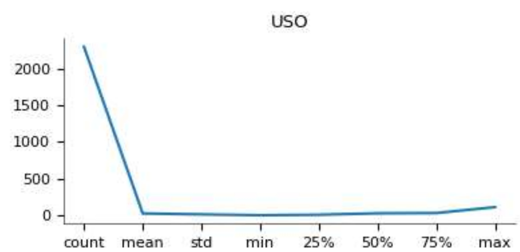
```
# getting the statistical measures of the data
gold_data.describe()
```

	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000
mean	1654.315776	122.732875	31.842221	20.084997	1.283653
std	519.111540	23.283346	19.523517	7.092566	0.131547
min	676.530029	70.000000	7.960000	8.850000	1.039047
25%	1239.874969	109.725000	14.380000	15.570000	1.171313
50%	1551.434998	120.580002	33.869999	17.268500	1.303297
75%	2073.010070	132.840004	37.827501	22.882500	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

Values







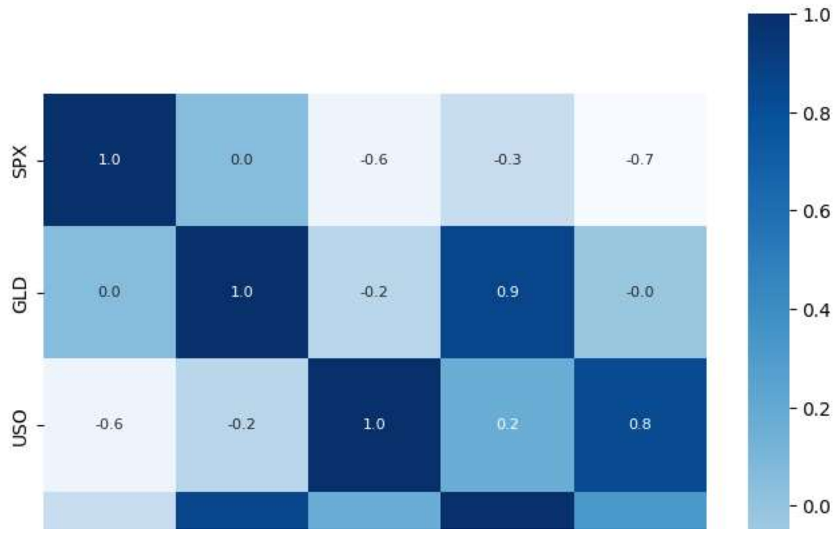
SLV

Correlation:

1. Positive Correlation
2. Negative Correlation

```
correlation = gold_data.corr()
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')
```

```
<ipython-input-9-ac468a117088>:1: FutureWarning: The default value of numeric_only in Da
  correlation = gold_data.corr()
<Axes: >
```




```
# correlation values of GLD
print(correlation['GLD'])
```

```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

SPX GLD USO SLV EUR/USD

```
# checking the distribution of the GLD Price
sns.distplot(gold_data['GLD'],color='green')
```

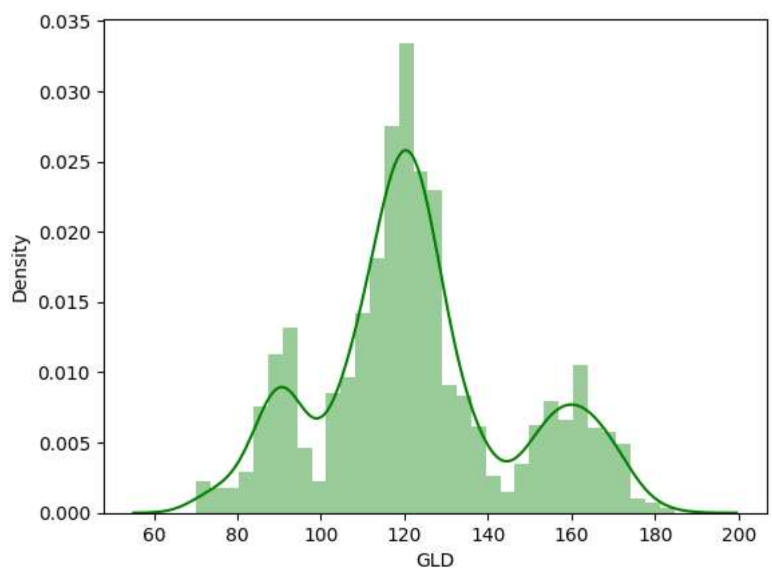
```
<ipython-input-11-b94eac2e88dd>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(gold_data['GLD'],color='green')
<Axes: xlabel='GLD', ylabel='Density'>
```



Splitting the Features and Target

```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
print(X)
```

```
              SPX        USO      SLV   EUR/USD
0     1447.160034  78.470001  15.1800  1.471692
1     1447.160034  78.370003  15.2850  1.474491
2     1411.630005  77.309998  15.1670  1.475492
3     1416.180054  75.500000  15.0530  1.468299
4     1390.189941  76.059998  15.5900  1.557099
...           ...        ...      ...       ...
2285  2671.919922  14.060000  15.5100  1.186789
2286  2697.790039  14.370000  15.5300  1.184722
2287  2723.070068  14.410000  15.7400  1.191753
2288  2730.129883  14.380000  15.5600  1.193118
2289  2725.780029  14.405800  15.4542  1.182033

[2290 rows x 4 columns]
```

```
print(Y)
```

```
0        84.860001
1        85.570000
2        85.129997
3        84.769997
4        86.779999
           ...
2285    124.589996
2286    124.330002
2287    125.180000
2288    124.489998
2289    122.543800
Name: GLD, Length: 2290, dtype: float64
```

Splitting into Training data and Test Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)
```

Model Training: Random Forest Regressor

```
regressor = RandomForestRegressor(n_estimators=100)
# training the model
regressor.fit(X_train,Y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

Model Evaluation

```
# prediction on Test Data
test_data_prediction = regressor.predict(X_test)
print(test_data_prediction)
```

```
 153.95100109 121.20859994 156.34899862  93.04930062 125.52470186
 126.14160044  87.84690022  92.31969906 126.26369918 127.96430337
 113.03239982 117.69269717 120.73140022 127.10859796 119.52300098
 137.32900041  93.98649926 119.81700042 113.36440105  94.29009927
 109.00550013  87.88449913 109.23489923  89.63409991  92.36150031
 131.91610263 162.32540003  89.36600005 119.40130098 133.5950017
 124.04670044 128.46710132 101.91449848  88.85149896 131.3031007
 119.7300001  108.61770002 168.15360125 115.25100064  86.58299949
 118.74960041  91.08919968 161.67280069 116.58740033 121.69999998
 160.36689843 120.17259969 112.69949964 108.48119877 126.73780022
  75.95280043 102.9959997  127.7304026  121.77789922  92.70440028
 132.02360078 118.18340097 115.80879991 154.95820267 158.66030046
 110.04349938 155.99279751 119.14970092 160.48320117 118.29040001
 157.87150015 115.1127992  116.37920034 150.20789946 114.81610075
 125.80409858 165.53399997 117.4625     125.08299912 153.41130373
 153.46530238 131.8451002  114.80450061 121.26490215 124.78130062
  89.79290036 123.08979994 155.79400194 111.8201004  106.53159985
 161.82320134 118.38409992 165.54479947 134.22430075 115.06679969
 152.92089855 168.50970032 114.77780061 114.08420118 158.88149889
  85.2990989  127.1159006  127.81610082 129.01939999 124.33550064
 123.90940024  90.68310081 153.19219982  97.20069941 136.16799987
  88.84029931 107.53389998 115.16850045 112.78140112 124.5349993
  91.40079858 125.34960113 162.35019892 119.90629852 165.00090146
 126.87759788 112.34890017 127.55689917  94.77529915  90.79379975
 102.66239909 120.73169996  83.16679944 126.21380064 160.38650476
 117.16730083 118.25129994 119.91660003 122.74529944 120.1069014
 121.55900001 118.35560038 107.10579998 148.64770046 126.22659819
 115.72320088  73.94539988 127.82380128 154.66100061 122.12090012
 125.64720078  88.92320028 103.99009904 124.60220052 120.1938004
  73.37120093 151.81469969 121.07280041 104.70940007  86.29039784
 115.3047991  172.29169735 119.93780022 160.02799833 113.12859959
 121.01950013 118.39540099  95.89329981 118.7290003  125.86480057
 118.52849973  95.80380041 154.42520148 122.04080019 148.11149987
 159.90360216 114.01310009 122.55969933 149.95659826 127.26730018
 165.91240086 135.4577006  120.19619938 167.06889851 108.41759861
 121.77889852 138.92960088 106.82129904]
```

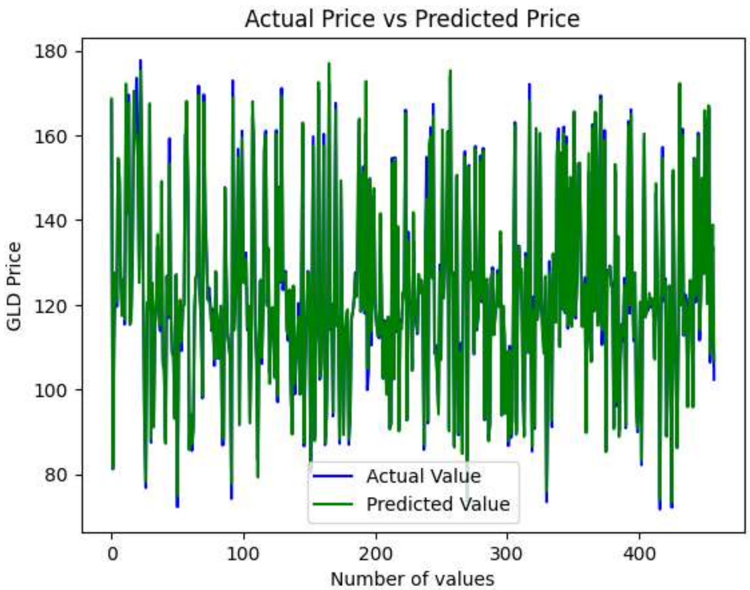
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", error_score)
```

```
R squared error :  0.9890515443068826
```

Compare the Actual Values and Predicted Values in a Plot

```
Y_test = list(Y_test)
```

```
plt.plot(Y_test, color='blue', label = 'Actual Value')
plt.plot(test_data_prediction, color='green', label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```



✓ 0s completed at 4:32 PM