```
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
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# loading the csv data to a pandas dataframe
gold_data = pd.read_csv('/content/gold_price_data.csv')
```

```
# print first five rows in the dataframe
gold_data.head()
```

	Date	SPX	GLD	USO	SLV	EUR/USD
0	01/02/2008	1447.160034	84.860001	78.470001	15.180	1.471692
1	01/03/2008	1447.160034	85.570000	78.370003	15.285	1.474491
2	01/04/2008	1411.630005	85.129997	77.309998	15.167	1.475492
3	01/07/2008	1416.180054	84.769997	75.500000	15.053	1.468299
4	01/08/2008	1390.189941	86.779999	76.059998	15.590	1.557099

Next steps: Generate code with gold_data | View recommended plots | New interactive sheet

```
# print last five rows in the dataframe
gold_data.tail()
```

	Date	SPX	GLD	USO	SLV	EUR/USD
2285	05/08/2018	2671.919922	124.589996	14.0600	15.5100	1.186789
2286	05/09/2018	2697.790039	124.330002	14.3700	15.5300	1.184722
2287	05/10/2018	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

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

```
(2290, 6)
```

```
# getting some information about 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 number of missing values
gold_data.isnull().sum()
```

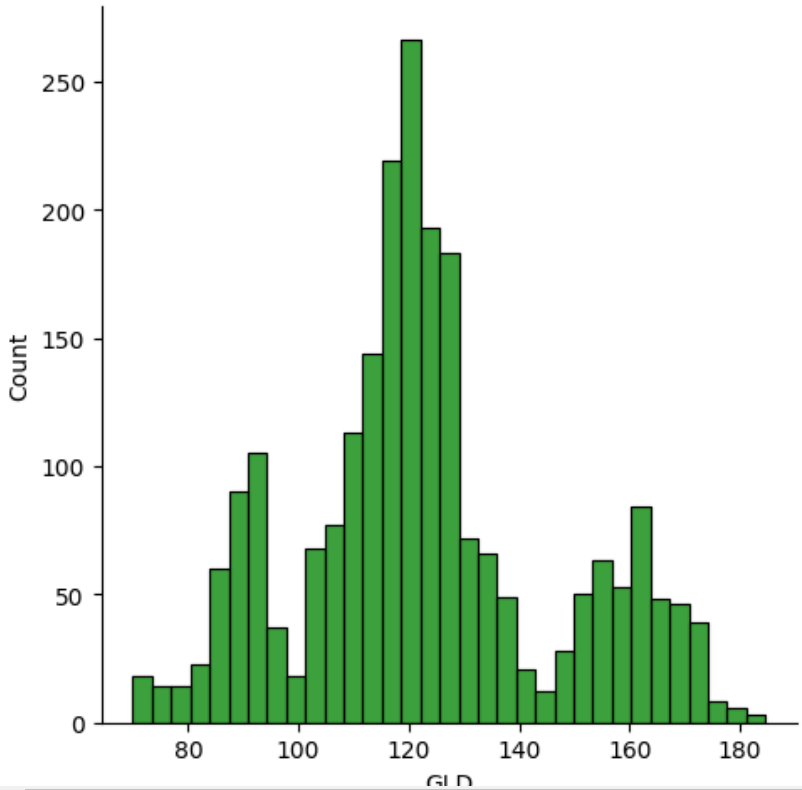
	0
Date	0
SPX	0
GLD	0
USO	0
SLV	0
EUR/USD	0

```
# statistical measures of 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

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

```
<seaborn.axisgrid.FacetGrid at 0x7ac788f4d360>
```




```
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
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)
regressor = RandomForestRegressor(n_estimators=100)
# training the model
regressor.fit(X_train, Y_train)
```

```
RandomForestRegressor
RandomForestRegressor()
```

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

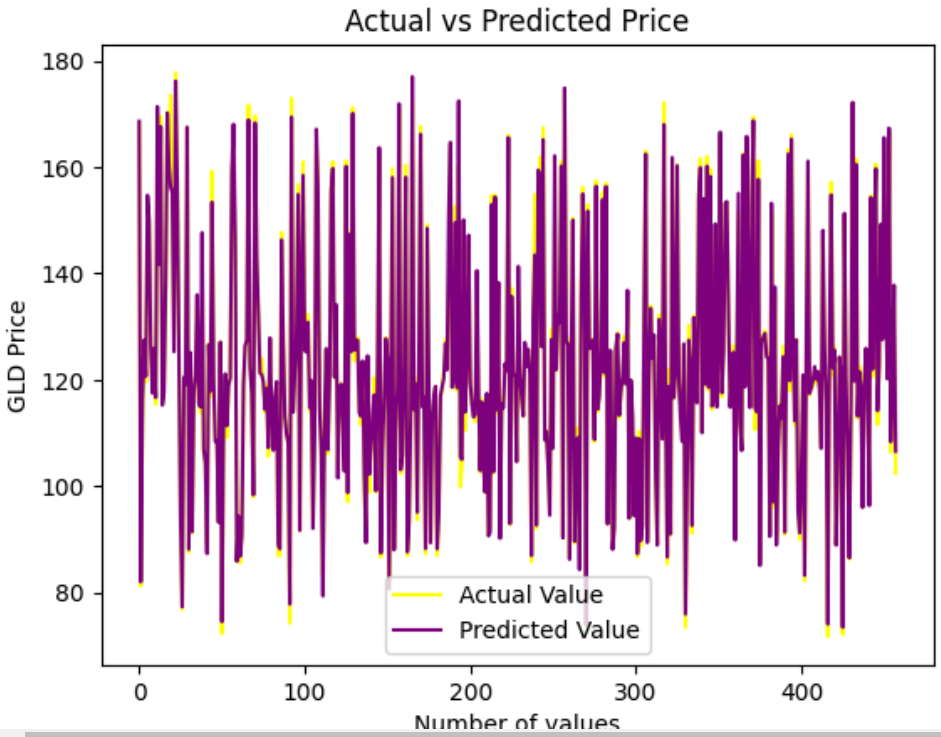
```
[166.20360046 114.91740061 116.65890134  88.24749851 148.50419991
 120.41529943  89.35619963 112.4113     116.8470003  118.76960125
  88.20089934  94.55200035 116.96880016 118.62690169 120.29650054
 126.81079824 121.8794003  151.47869962 164.67429992 118.61749954
 120.31600125 149.58460037 118.52459868 172.47639913 105.512499
 105.01770119 150.08060036 113.94490043 124.9724009  147.21469989
 119.45500145 115.35270017 113.01430002 113.46690209 140.55900123
 117.82129763 102.94690059 115.81330108 103.60200169  98.87810043
 117.44710057  90.64670023  91.31420054 152.99139879 102.71159984
 154.39940085 114.37470144 138.31530107  90.16529783 115.42309957
 114.71919942 123.02410044 121.70690014 165.55660135  92.94559955
 135.70470158 121.33349966 120.55540098 104.58810041 141.3211028
 121.60339941 116.62060037 113.19860078 127.03019719 122.51389964
 125.73779934 121.21740065  86.94029933 132.35240213 143.45590256
  92.64979966 159.45349951 158.64610374 126.25899887 165.17089936
 108.70299965 110.28630036 103.66049833  94.51960011 127.55630255
 107.12810038 162.14749991 121.91170034 132.09719998 130.85280172
 160.19090005  90.26469866 174.88160132 127.24780029 126.72969881
  86.22419914 124.66619962 150.03339715  89.59559993 107.10559971
 109.07960006  84.28669901 135.8139997  154.98550277 139.13360327
  73.93200016 151.74220045 125.87659928 126.70910021 127.49449921
 108.76629973 156.41050026 114.49330105 116.91100127 125.58799929]
```

```
       165.55839982 135.54300024 120.17019979 167.35329845 108.34079886
       121.45409882 137.79360186 106.57389863]
```

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

```
R squared error: 0.9888570165950995
```

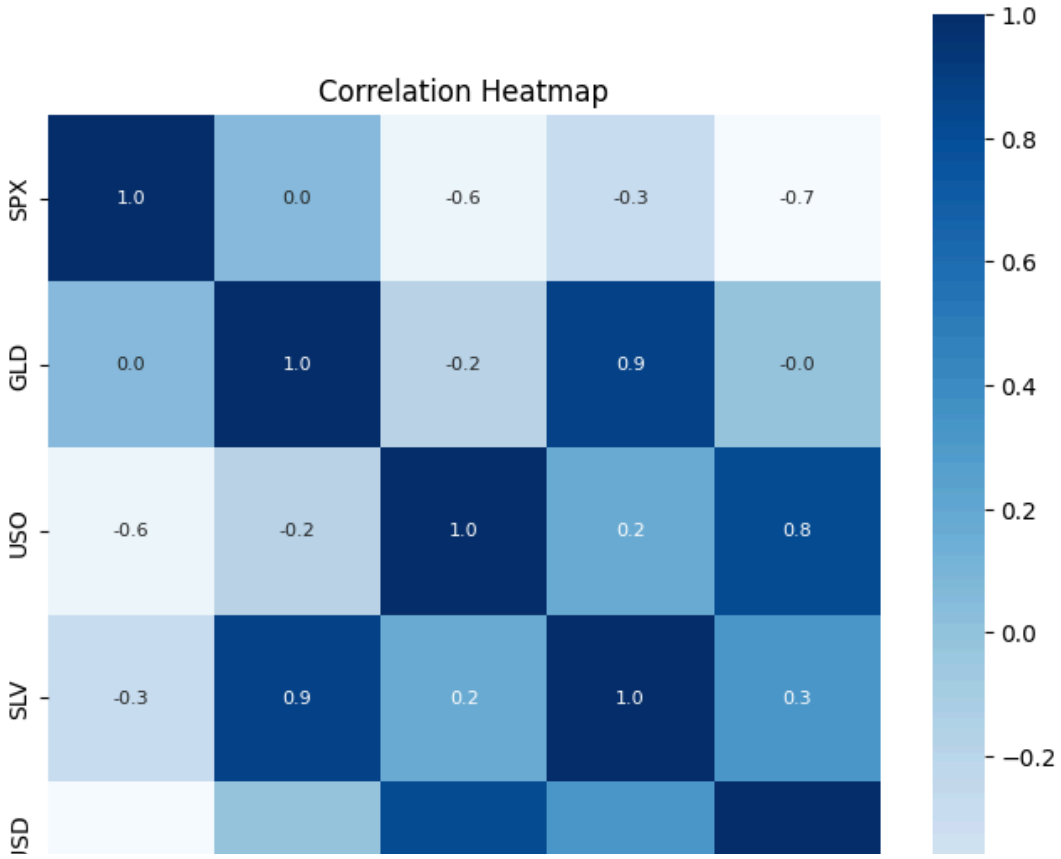
```
Y_test = list(Y_test)
plt.plot(Y_test, color='yellow', label = 'Actual Value')
plt.plot(test_data_prediction, color='purple', label='Predicted Value')
plt.title('Actual vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```




```
# Drop non-numerical columns like 'Date'
numeric_data = gold_data.drop(['Date'], axis=1)

# Compute correlation on numerical data
correlation = numeric_data.corr()

# Plot the heatmap
plt.figure(figsize=(8, 8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 8}, cmap='Blues')
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap
SPX
GLD
USO
SLV
USD
1.0
0.0
-0.6
-0.3
-0.7
0.0
1.0
-0.2
0.9
-0.0
-0.6
-0.2
1.0
0.2
0.8
-0.3
0.9
0.2
1.0
0.3
1.0
0.8
0.6
0.4
0.2
0.0
−0.2