Importing the necessary 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
```

```
gold_data = pd.read_csv('gold_price_data.csv')
```

```
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
```

```
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
```

```
# No. of rows and columns
gold_data.shape
```

```
(2290, 6)
```

```
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
```

```python
gold_data['Date'] = pd.to_datetime(gold_data['Date'])
```

```python
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   datetime64[ns]
 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: datetime64[ns](1), float64(5)
memory usage: 107.5 KB
```

```python
gold_data['Year'] = gold_data['Date'].dt.year
gold_data['Month'] = gold_data['Date'].dt.month
gold_data['Day'] = gold_data['Date'].dt.day
```

```python
gold_data.head()
```

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

   Month  Day
0      1    2
1      1    3
2      1    4
3      1    7
4      1    8
```

```python
gold_data.drop(labels=['Date'],axis=1,inplace=True)
```

```python
gold_data.head()
```

```
           SPX        GLD        USO     SLV   EUR/USD  Year  Month
Day
0  1447.160034  84.860001  78.470001  15.180  1.471692  2008      1
```

```
2
1  1447.160034  85.570000  78.370003  15.285  1.474491  2008       1
3
2  1411.630005  85.129997  77.309998  15.167  1.475492  2008       1
4
3  1416.180054  84.769997  75.500000  15.053  1.468299  2008       1
7
4  1390.189941  86.779999  76.059998  15.590  1.557099  2008       1
8
```

```
gold_data.tail()
```

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

Check for missing values

```
gold_data.isnull().sum()
```

```
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
Year       0
Month      0
Day        0
dtype: int64
```

Check for duplicate values

```
gold_data.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2285    False
```

```
2286    False
2287    False
2288    False
2289    False
Length: 2290, dtype: bool
```

```
gold_data.duplicated().sum()
```

```
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


              Year        Month          Day
count  2290.000000  2290.000000  2290.000000
mean   2012.724891     6.329258    15.644541
std       2.993271     3.591149     8.746132
min    2008.000000     1.000000     1.000000
25%    2010.000000     3.000000     8.000000
50%    2013.000000     6.000000    15.500000
75%    2015.000000    10.000000    23.000000
max    2018.000000    12.000000    31.000000
```

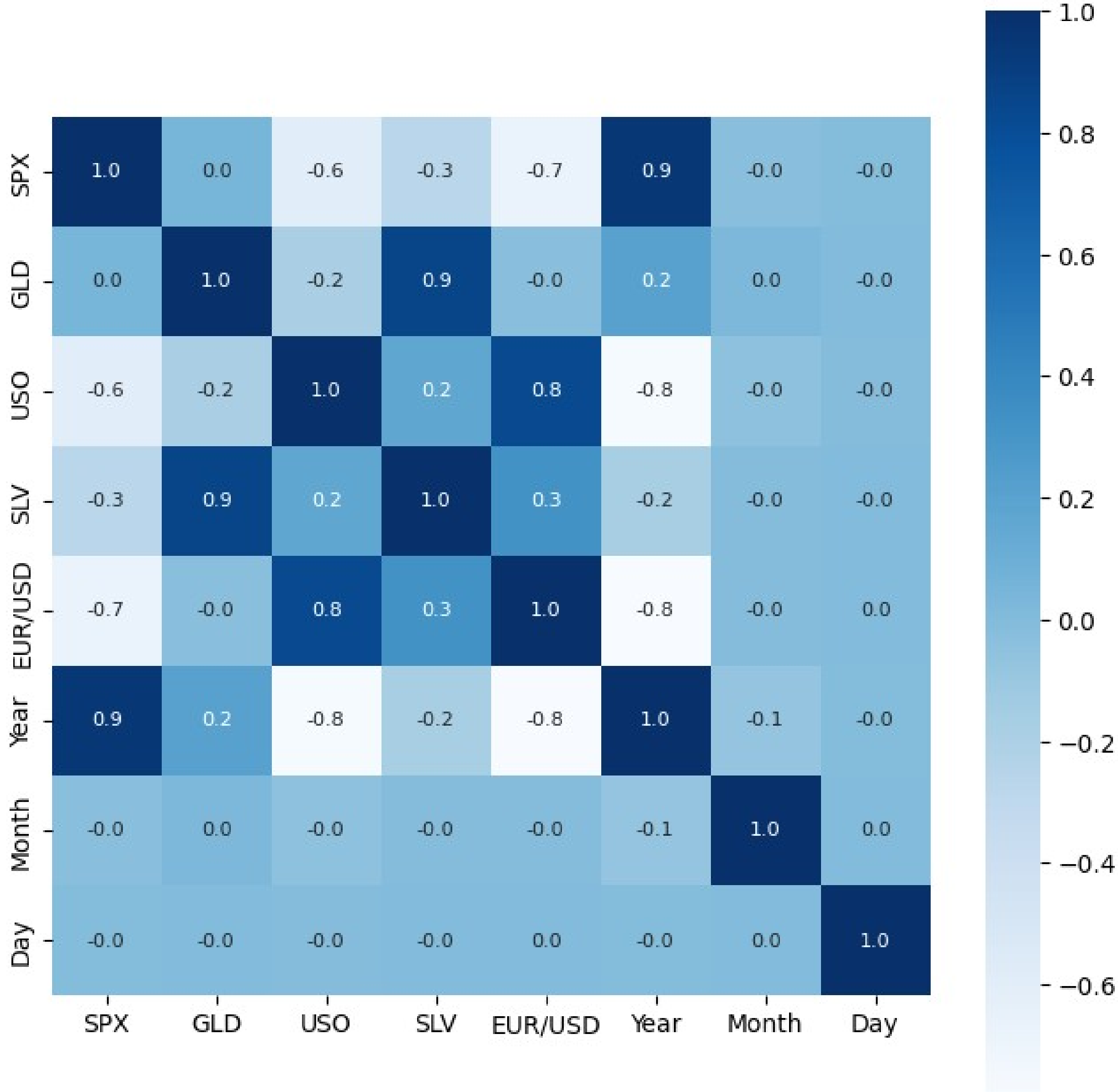
Check for correlation

1. Positive Correlation -> if 2 variables are directly proportional
2. Negative Correlation -> if 2 variables are inversly proportional

```
correlation = gold_data.corr()
```

Constructing a heatmap for understanding correlation

```
plt.figure(figsize=(8,8))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,ann
ot_kws={'size':8},cmap='Blues')
plt.show()
```



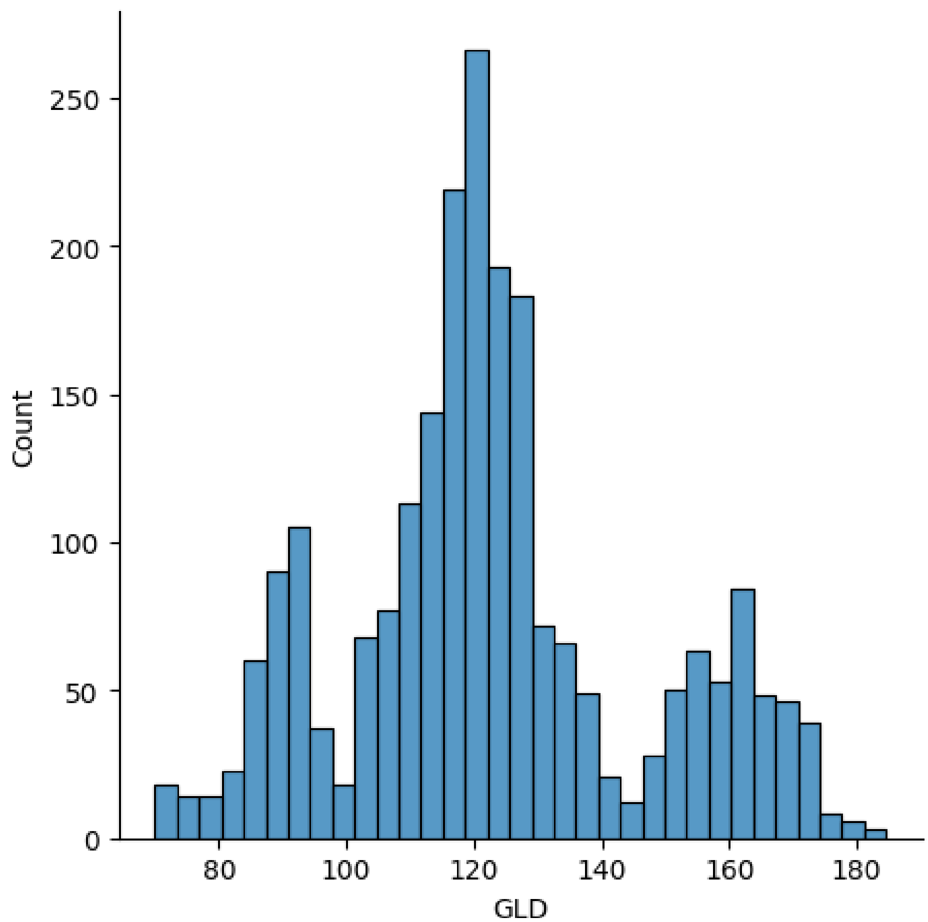

```
correlation['GLD']

SPX         0.049345
GLD         1.000000
USO        -0.186360
SLV         0.866632
EUR/USD    -0.024375
```

```
Year       0.206654
Month      0.020494
Day       -0.000198
Name: GLD, dtype: float64
```

```
sns.displot(gold_data['GLD'])
plt.show()
```

```
C:\Users\Sanke\AppData\Roaming\Python\Python311\site-packages\seaborn\
axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Splitting the dataframe into independent and dependent features

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

print(X)
```

```
              SPX        USO      SLV   EUR/USD  Year  Month  Day
0     1447.160034  78.470001  15.1800  1.471692  2008      1    2
```

```
1      1447.160034  78.370003  15.2850  1.474491  2008      1    3
2      1411.630005  77.309998  15.1670  1.475492  2008      1    4
3      1416.180054  75.500000  15.0530  1.468299  2008      1    7
4      1390.189941  76.059998  15.5900  1.557099  2008      1    8
...            ...        ...      ...       ...   ...    ...  ...
2285   2671.919922  14.060000  15.5100  1.186789  2018      5    8
2286   2697.790039  14.370000  15.5300  1.184722  2018      5    9
2287   2723.070068  14.410000  15.7400  1.191753  2018      5   10
2288   2730.129883  14.380000  15.5600  1.193118  2018      5   14
2289   2725.780029  14.405800  15.4542  1.182033  2018      5   16

[2290 rows x 7 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 the data into train 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

```
regressor = RandomForestRegressor(n_estimators=100)

regressor.fit(X_train,Y_train)
```

```
RandomForestRegressor()
```

Model Evaluation ->Prediction on test data

```
test_data_pred = regressor.predict(X_test)

print(test_data_pred)
```

```
[168.13399862  82.72289986 115.87160047 127.36800095 120.02710135
 154.7106986  150.32789945 126.27690027 117.72529869 126.03590111
 115.60930196 170.90740083 140.93229971 167.74749765 115.03330003
```

118.1354012	134.27690152	171.40640272	159.79000265	172.57749929
155.06740045	124.19100044	174.90370003	156.85800357	125.39850081
93.37679926	77.07370012	119.54720037	119.04349885	167.44109894
88.0095006	125.40290079	91.85030008	117.65850019	121.13900004
135.55230051	115.74720058	114.43460054	140.69029843	107.51260069
105.58760253	86.95809742	126.51060088	117.55140065	155.3545992
120.25649928	108.58079993	107.9188977	92.7201998	127.16809737
75.41410024	114.07180005	120.77629977	111.27559932	118.80909896
120.89829879	160.20100197	174.87800038	146.52599639	86.97279995
93.76640049	86.86719875	89.58230053	119.2814007	126.35610074
127.82859938	171.85730146	122.27519932	117.51159864	97.5438999
168.37830079	142.17379921	132.58900146	120.74870104	123.6035988
119.77020102	114.29000176	118.04080049	107.3159005	128.05310064
114.76959952	105.72460035	117.40800097	119.5867989	87.96109907
88.1291986	149.94420327	127.49110132	114.17439971	110.1886978
108.2905993	77.35709916	170.48130276	114.10779898	121.65319914
127.9672003	154.81089839	91.84209968	136.43450122	159.53200162
125.99600018	125.9186999	131.59500064	114.56710123	119.26680016
92.15649952	110.8977987	170.51430124	157.62009969	114.36850011
107.88640111	79.02829971	113.07890007	125.80990031	107.37629946
119.0971013	156.00690235	159.54719924	119.68850001	133.25930199
105.6488993	117.44319852	119.10720048	112.75910058	102.80599918
159.91139751	97.63730036	146.33550039	125.68940117	171.16659978
125.35799915	127.2207972	127.61430207	114.14029906	111.3817007
122.66519971	102.136799	89.12800026	125.09099945	98.45219951
106.06929835	110.79450152	117.95370028	97.74169958	121.65720018
165.4747012	87.20999784	106.42579965	117.25320088	127.92790111
123.72030093	80.59679903	119.52500112	158.36249829	87.98259823
110.24819927	116.9704	171.89460025	103.03979905	105.50880088
122.52970001	158.90649789	86.9017987	92.69190089	112.3234005
176.55970018	114.60829975	119.41930079	94.15580046	125.63770037
166.85300089	114.50250096	116.7467013	88.18889868	146.84749614
119.65979927	89.08149967	112.74639999	116.86690089	118.62070147
87.87579902	93.93709965	116.74280016	118.30680127	120.21929975
126.85569856	121.80599968	138.9887005	166.3822991	118.52099967
120.45870225	151.85590046	118.61949928	172.60779974	99.124099
105.13300057	146.84149623	110.89250142	125.09410084	146.29290102
119.2851011	114.74159994	112.76820003	113.7877013	137.80320093
117.76029803	103.04350058	115.9909014	105.64370213	97.92110074
117.85320072	90.88929919	91.42419972	152.55529791	102.86679946
155.03510122	114.43240157	137.45460119	91.54459983	115.51449883
114.47300006	122.16710066	121.79730037	165.30900071	92.67360027
135.64090066	121.50169839	120.6839008	104.58060008	137.7830035
122.04999906	116.64670012	114.18470073	126.92829913	122.56709895
125.89739902	121.41199902	86.87569863	132.15450152	152.00630019
92.84309981	149.08449817	159.45580085	126.65029882	167.08709939
109.1175003	109.07250113	103.66929837	94.4306001	129.20980275
109.36870052	149.99819899	121.74109991	132.15940029	131.5789008
160.78269781	90.05769933	172.40860174	127.03840111	126.88719866

```
  86.49089914 124.69919893 150.17739728  89.1651998  107.05389906
 109.51529949  87.72179924 135.94260004 154.7799026  137.28170392
  73.64000064 152.91620055 126.39209976 126.78470005 127.53839878
 108.61469878 156.67120164 114.53319972 117.12490163 123.8069002
 154.81420191 121.39269955 156.26489943  92.84730008 125.5814008
 125.00180031  87.98820092  91.95989902 126.28180073 128.40070412
 113.08430029 118.09719758 120.89959994 127.22749826 120.19260201
 135.55000079  95.49820073 119.78230042 113.23730116  94.37229972
 109.24599971  88.01489947 111.14019934  89.17880017  92.36350006
 131.96080356 162.48399935  89.01139938 119.69400107 133.46920169
 123.66389958 128.0562012  101.87009823  88.59299785 131.74930147
 120.59280113 108.30389987 171.36400081 115.5803009   86.74409895
 120.10750044  90.82859991 161.14460122 117.05180082 121.82389968
 160.33119848 119.90289952 111.70249929 108.84999965 126.77259999
  76.4275995  102.75139978 129.10700203 121.70910032  92.27029919
 132.61270044 117.9147006  116.2979     154.69160242 160.4259009
 110.04929954 136.80289807 119.03780106 159.91360005 118.00929944
 159.4061019  115.24849908 117.04960074 146.69699704 114.40030095
 125.60059848 166.94759804 117.70970059 125.13849961 152.99790424
 153.42990215 132.17880023 114.94060005 120.97020127 123.26600033
  90.30370073 123.54149938 153.32819983 111.68550009 106.4114007
 162.19730151 118.54989954 165.50879988 133.56600242 115.43980029
 152.66689724 169.04720167 114.29889942 114.11260124 161.26989871
  85.65989908 127.07710087 127.62620034 128.25810041 124.29080126
 124.06070135  90.42910074 152.35940102  96.89230002 136.53210043
  89.46839978 105.89740022 114.78700014 111.15030091 125.29119874
  91.36289922 125.40810083 162.23289768 118.36610123 165.48290169
 127.08819821 112.23959989 127.54380007  94.95049847  90.84669969
  98.79869936 120.81229979  83.42309957 126.19609996 160.58010374
 117.2648004  118.11410016 119.44859969 121.0882998  119.54560092
 121.2326996  117.95100056 107.11749977 146.77129652 125.98289866
 115.84700098  74.32270022 127.84220087 154.48880088 120.54870021
 125.68390108  89.26640049 102.96829904 125.30359988 120.1317998
  73.46810087 151.76890043 120.75180022 104.49829959  86.15269829
 115.18589931 171.09399868 120.43660023 161.30089725 113.0736994
 121.8574009  118.01890074  95.52619978 117.68150075 125.44679996
 118.57799982  95.95670063 154.58380105 122.59089973 146.50889803
 159.47180356 113.6832003  122.20759968 146.00259701 127.45080074
 165.53320035 134.91080077 119.43130001 167.01949871 108.16839875
 121.97449916 137.25069929 102.87869937]
```

Compare Y_test and test_data_pred

```
score = metrics.r2_score(Y_test,test_data_pred)
print(score)
```

```
0.9953783140409124
```

Compare the actual values and predicted values in plot

```
Y_test = list(Y_test)

plt.plot(Y_test,color='blue',label='Actual_Value')
plt.plot(test_data_pred,color='red',label='Predicted_Value')
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```