Gold Price Prediction

(1) Import Python Libraries

In [1]:
```
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
```

(2) Loading the Data Set

In [2]:
```
path = r'D:\IITG\portfolio_finance\gold_price\gold_price_data.csv'
```

In [3]:
```
gold_data = pd.read_csv(path)
gold_data.head()
```

Out[3]:

	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

In [4]:
```
gold_data.tail()
```

Out[4]:

	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

(3) Exploring the Data Set

In [5]:
```
gold_data.shape
```

Out[5]:
```
(2290, 6)
```

In [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
```

In [7]:
```
gold_data.isnull().sum()
```

Out[7]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

In [8]:
```
gold_data.describe()
```

Out[8]:

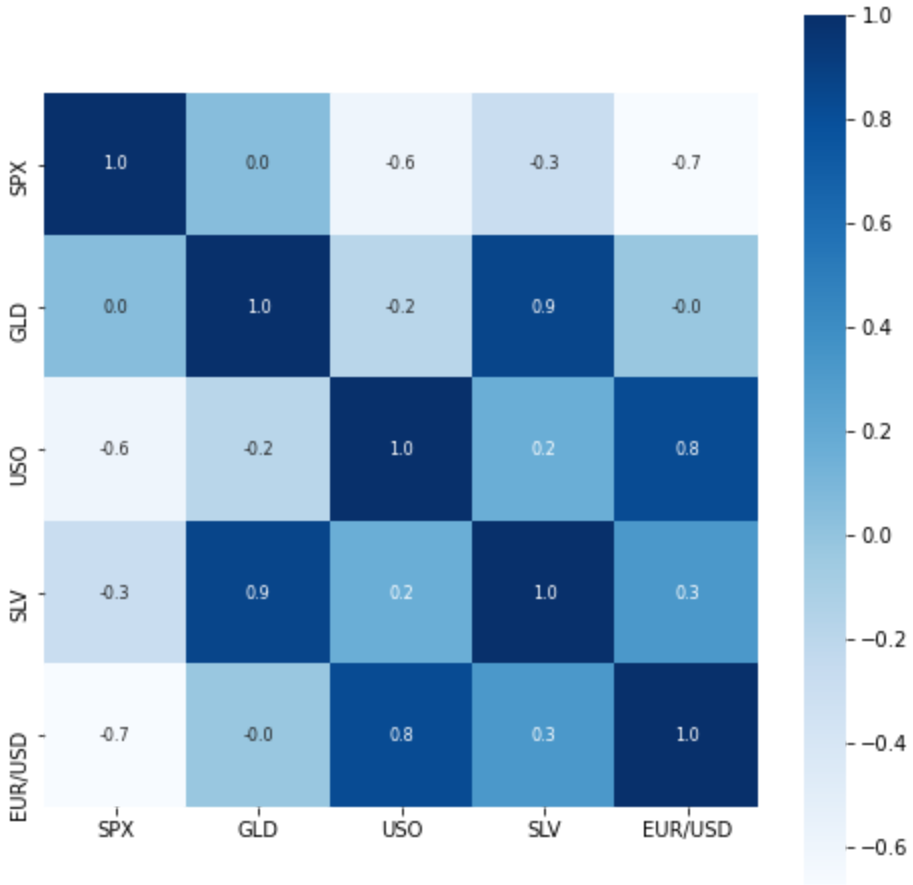
	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000
mean	1654.315776	122.732875	31.842221	20.084997	1.283653
std	519.111540	23.283346	19.523517	7.092566	0.131547
min	676.530029	70.000000	7.960000	8.850000	1.039047
25%	1239.874969	109.725000	14.380000	15.570000	1.171313
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

(4) Finding Correlations in the Data Set

In [9]:
```
correlation = gold_data.corr()
```

In [10]:
```
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot
```

Out[10]: <AxesSubplot:>



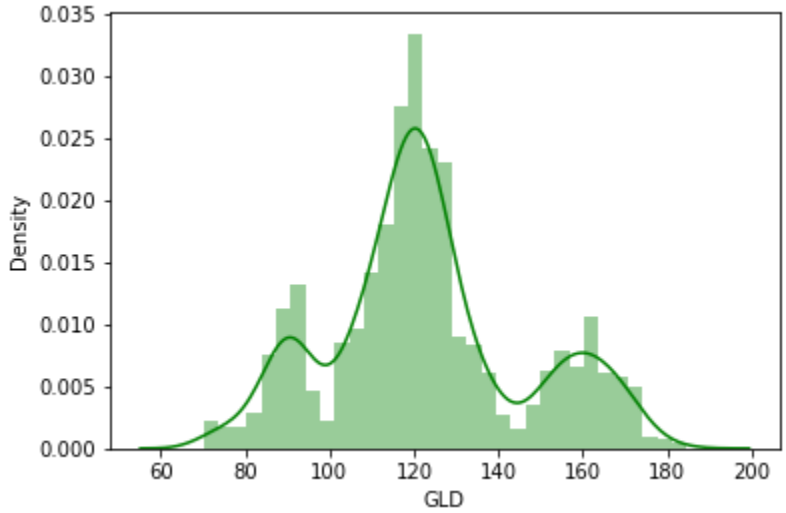
In [12]:
```
# correlation values of GLD
print(correlation['GLD'])
```

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

In [16]:
```
# checking the distribution of the GLD Price
sns.distplot(gold_data['GLD'],color='green')
```

```
C:\Users\SHBHAM\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-leve
l function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]: <AxesSubplot:xlabel='GLD', ylabel='Density'>



(5) Building the Model

Split the Data

In [17]:
```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

In [18]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, r
```

Create and Fit the Model

In [19]:
```
regressor = RandomForestRegressor(n_estimators=100)
regressor.fit(X_train,Y_train)
```

Out[19]: RandomForestRegressor()

(6) Evaluating the Model

In [21]:
```
test_data_prediction = regressor.predict(X_test)
test_data_prediction[:5]
```

Out[21]:
```
array([168.77899949,  82.20419984, 115.78319995, 127.67640081,
       120.47140151])
```

R-squared Error

In [22]:
```
### R-squared Errorerror_score = metrics.r2_score(Y_test, test_data_predicti
print("R squared error : ", error_score)
```

```
R squared error :  0.9895529764582724
```

In [23]:
```
Y_test = list(Y_test)
```