In [1]:
```
import pandas as pd
import numpy as np
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
```

In [2]:
```
df = pd.read_csv("gld_price_data.csv")
```

In [5]:
```
df
```

Out[5]:

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.1800	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.2850	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.1670	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.0530	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.5900	1.557099
...	...	...	...	...	...	...
2285	5/8/2018	2671.919922	124.589996	14.060000	15.5100	1.186789
2286	5/9/2018	2697.790039	124.330002	14.370000	15.5300	1.184722
2287	5/10/2018	2723.070068	125.180000	14.410000	15.7400	1.191753
2288	5/14/2018	2730.129883	124.489998	14.380000	15.5600	1.193118
2289	5/16/2018	2725.780029	122.543800	14.405800	15.4542	1.182033

2290 rows × 6 columns

In [7]:
```
df.tail()
```

Out[7]:

	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

In [9]:
```
df.shape
```

Out[9]:
```
(2290, 6)
```

In [11]:
```
df.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 [13]:
```
df.isnull().sum()
```

Out[13]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

In [15]:
```
##getting some statisctical data
df.describe()
```

Out[15]:

	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

In [17]:
```
##co relation postive if one variable increase and the other one also increases otherwise it is negative corelation
```

In [32]:
```
z = df.drop(['Date'],axis = 1)
```

In [34]:
```
z
```

Out[34]:

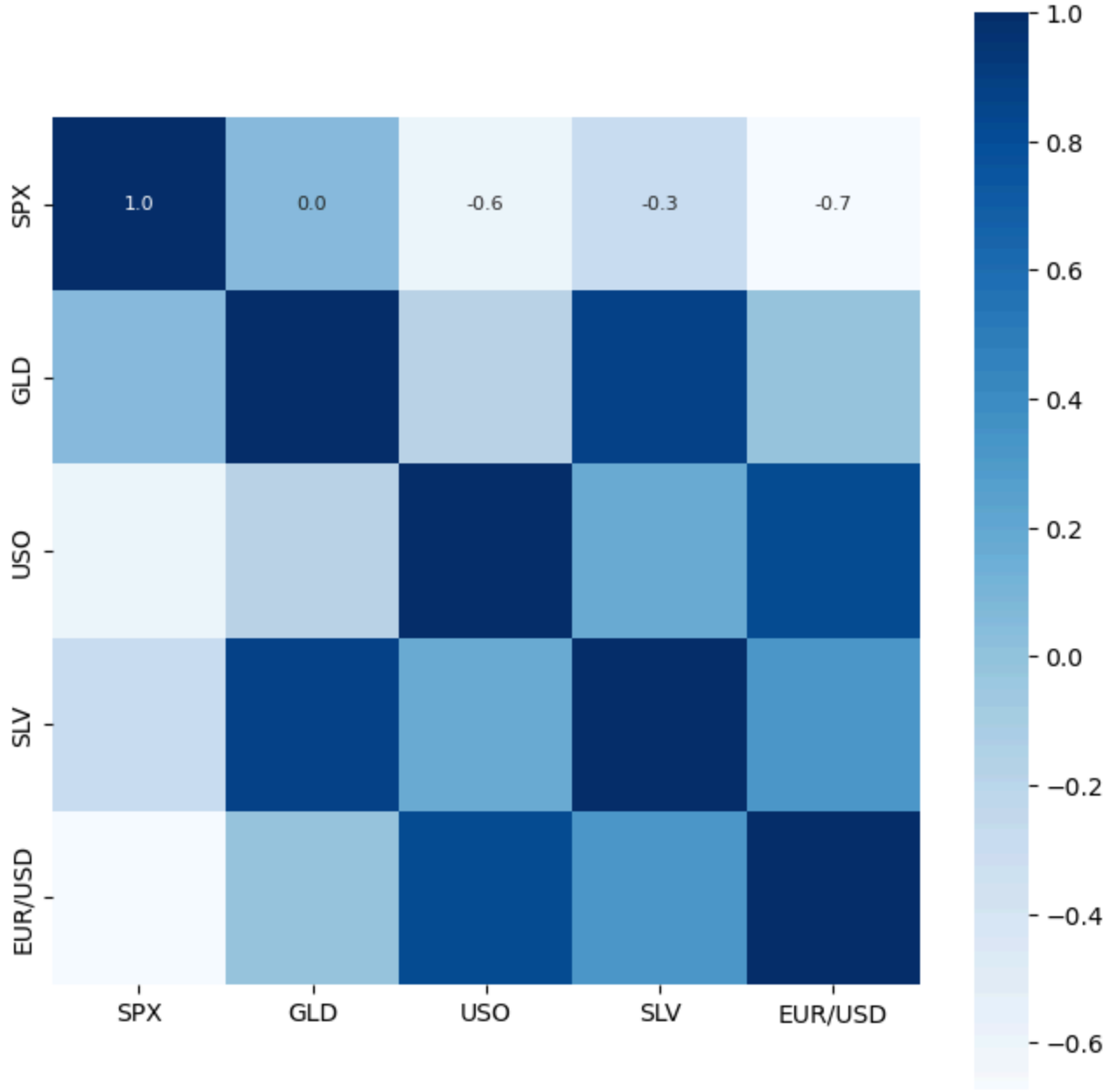
	SPX	GLD	USO	SLV	EUR/USD
0	1447.160034	84.860001	78.470001	15.1800	1.471692
1	1447.160034	85.570000	78.370003	15.2850	1.474491
2	1411.630005	85.129997	77.309998	15.1670	1.475492
3	1416.180054	84.769997	75.500000	15.0530	1.468299
4	1390.189941	86.779999	76.059998	15.5900	1.557099
...	...	...	...	...	...
2285	2671.919922	124.589996	14.060000	15.5100	1.186789
2286	2697.790039	124.330002	14.370000	15.5300	1.184722
2287	2723.070068	125.180000	14.410000	15.7400	1.191753
2288	2730.129883	124.489998	14.380000	15.5600	1.193118
2289	2725.780029	122.543800	14.405800	15.4542	1.182033

2290 rows × 5 columns

In [44]:
```
correlation = z.corr()
```

In [60]:
```
##create a heatmap
plt.figure(figsize = (8,8))
sns.heatmap(correlation,cbar = True , square = True ,fmt = '.1f',annot = True,annot_kws={'size':8},cmap ='Blues')
```

Out[60]: <Axes: >



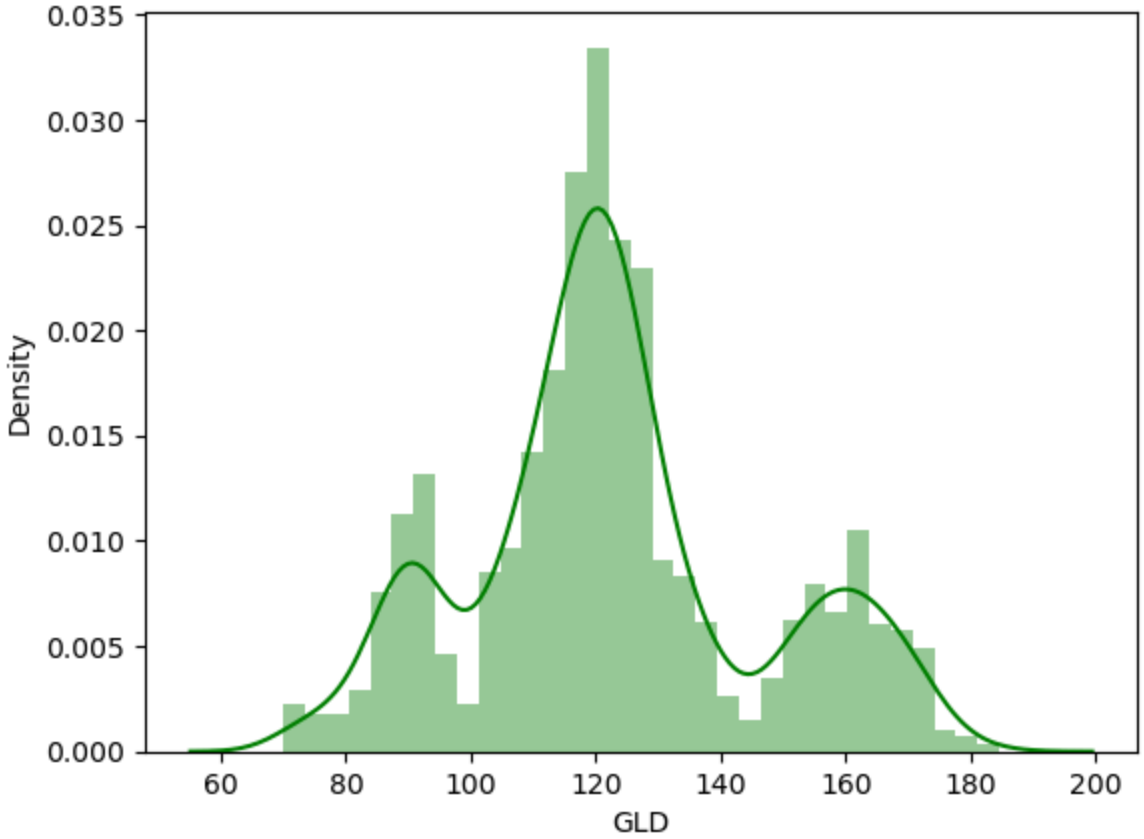
In [62]:
```
#cor values of gld
correlation['GLD']
```

Out[62]:
```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

In [70]:
```
sns.distplot(df['GLD'],color='green')
```

```
C:\Users\parth\AppData\Local\Temp\ipykernel_23648\3736964373.py:1: 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(df['GLD'],color='green')
C:\Users\parth\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecate
d and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[70]:
```
<Axes: xlabel='GLD', ylabel='Density'>
```


0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
60
80
100
120
140
160
180
200
GLD

In [72]:
```
z
```

Out[72]:

	SPX	GLD	USO	SLV	EUR/USD
0	1447.160034	84.860001	78.470001	15.1800	1.471692
1	1447.160034	85.570000	78.370003	15.2850	1.474491
2	1411.630005	85.129997	77.309998	15.1670	1.475492
3	1416.180054	84.769997	75.500000	15.0530	1.468299
4	1390.189941	86.779999	76.059998	15.5900	1.557099
...	...	...	...	...	...
2285	2671.919922	124.589996	14.060000	15.5100	1.186789
2286	2697.790039	124.330002	14.370000	15.5300	1.184722
2287	2723.070068	125.180000	14.410000	15.7400	1.191753
2288	2730.129883	124.489998	14.380000	15.5600	1.193118
2289	2725.780029	122.543800	14.405800	15.4542	1.182033

2290 rows × 5 columns

In [74]:
```
y = df['GLD']
```

In [76]:
```
y
```

Out[76]:
```
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
```

In [78]:
```
x = df.drop(['GLD','Date'],axis = 1)
```

In [80]:
```
x
```

Out[80]:

	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 × 4 columns

In [82]:
```
##splitting into training and test data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2 , random_state=2)
```

In [84]:
```
a = RandomForestRegressor()
a.fit(x_train,y_train)
```

Out[84]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

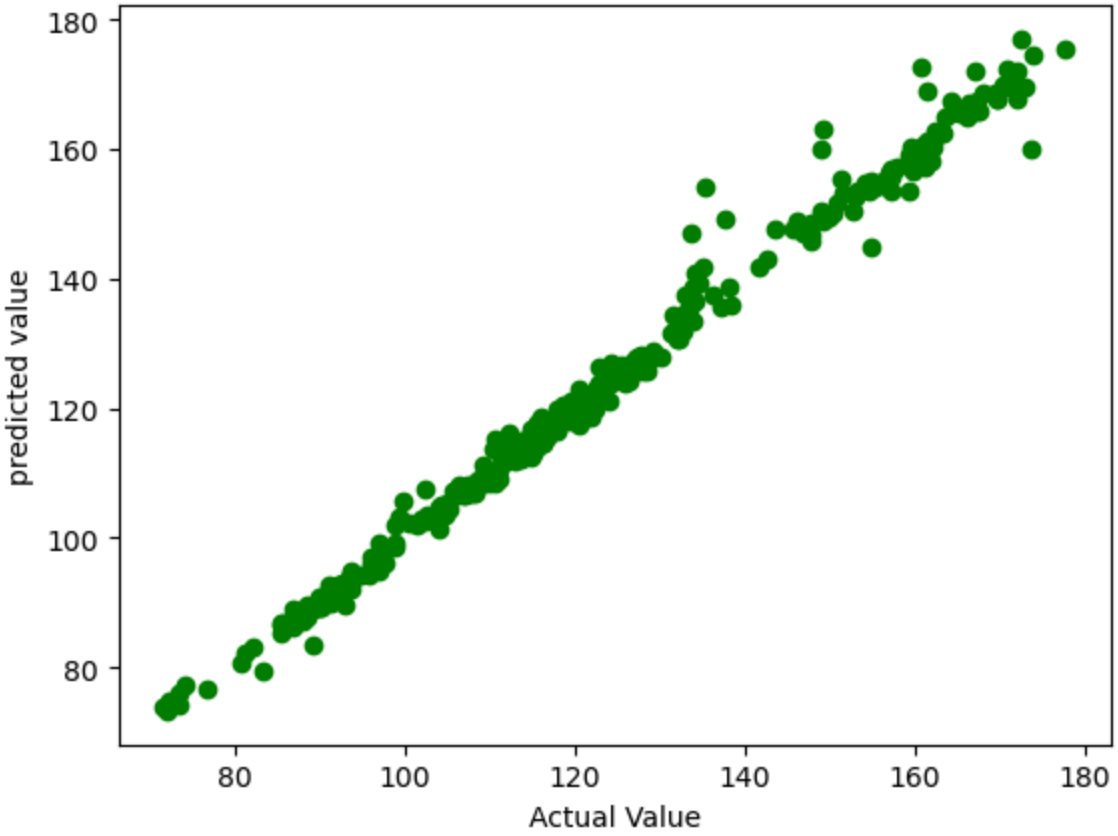
In [86]:
```
y_pred_test = a.predict(x_test)
```

In [88]:
```
error_test = metrics.r2_score(y_test,y_pred_test)
```

In [90]:
```
error_test
```

Out[90]: 0.9892769343291239

In [100…
```
#visualize
plt.scatter(y_test,y_pred_test,color='green')
plt.xlabel('Actual Value')
plt.ylabel('predicted value')
plt.show()
```



In [ ]: