In [1]:
```python
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
```

In [5]:
```python
# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('/Users/divyanshsahai/Desktop/gld_price_data.csv')
```

In [6]:
```python
# print first 5 rows in the dataframe
gold_data.head()
```

Out[6]:

	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 [7]:
```python
# print last 5 rows of the dataframe
gold_data.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 [29]:
```python
# number of rows and columns
gold_data.shape
```

Out[29]:
```
(2290, 6)
```

In [9]:
```python
# 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
```

In [10]:
```python
# checking the number of missing values
gold_data.isnull().sum()
```

Out[10]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

In [22]:
```python
# getting the statistical measures of the data
gold_data.describe()
```

Out[22]:

	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 [30]:
```
if 'Date' in gold_data.columns:
    gold_data['Date'] = pd.to_datetime(gold_data['Date'])
```

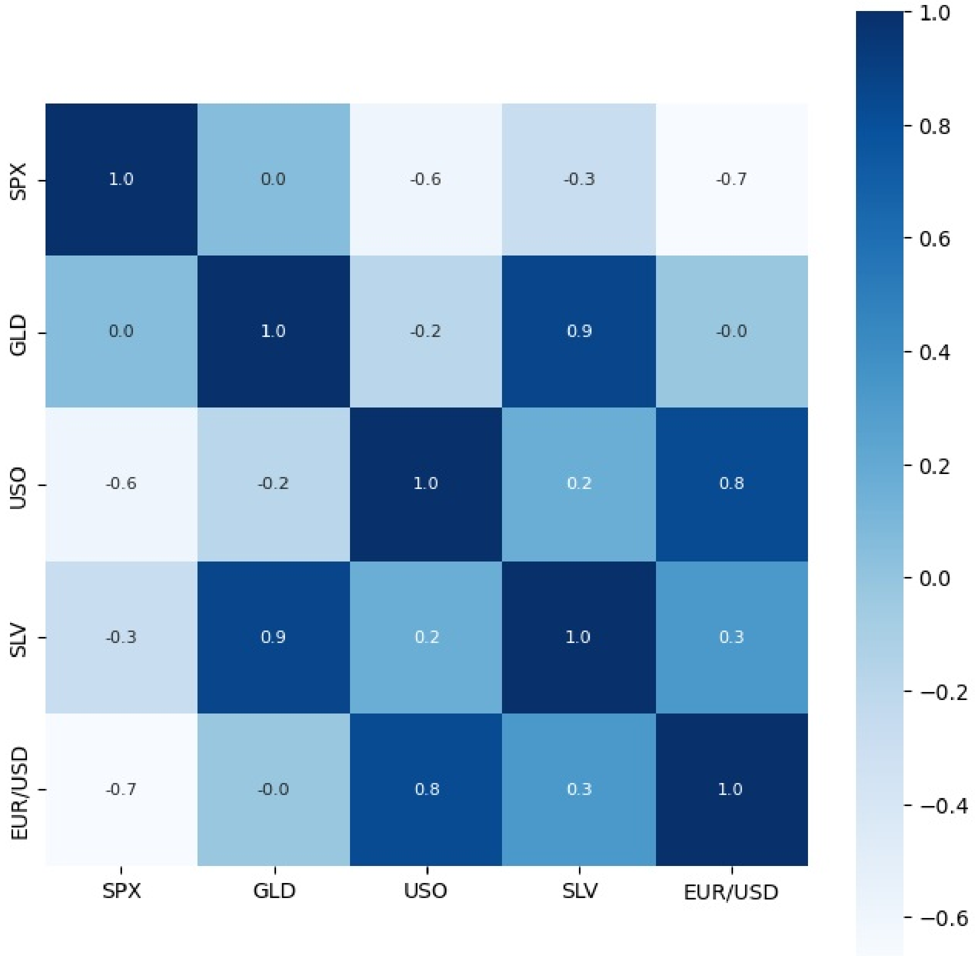
In [31]:
```
numeric_data = gold_data.select_dtypes(include=[np.number])
```

In [32]:
```
correlation = numeric_data.corr()
print(correlation)
```

```
              SPX       GLD       USO       SLV   EUR/USD
SPX      1.000000  0.049345 -0.591573 -0.274055 -0.672017
GLD      0.049345  1.000000 -0.186360  0.866632 -0.024375
USO     -0.591573 -0.186360  1.000000  0.167547  0.829317
SLV     -0.274055  0.866632  0.167547  1.000000  0.321631
EUR/USD -0.672017 -0.024375  0.829317  0.321631  1.000000
```

In [33]:
```
# 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')
```

Out[33]: <Axes: >



In [34]:
```
# 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
```

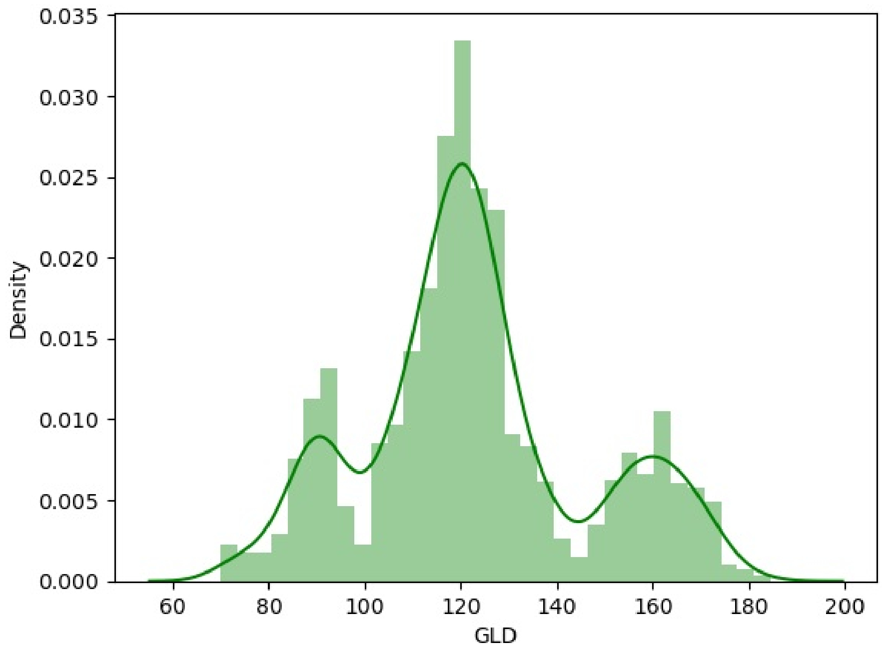
In [36]:
```
# checking the distribution of the GLD Price
```

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

```
/var/folders/6p/qjs3633d0dv716cpnqqw2tlw0000gn/T/ipykernel_72900/563101706.py: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')
```

Out[36]:

```
<Axes: xlabel='GLD', ylabel='Density'>
```



In [37]:

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

In [38]:

```
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]
```

In [39]:

```
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
```

In [40]:

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

In [41]:

```
regressor = RandomForestRegressor(n_estimators=100)
```

In [42]:

```
# training the model
regressor.fit(X_train,Y_train)
```

Out[42]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [43]:
```
# prediction on Test Data
test_data_prediction = regressor.predict(X_test)
```

In [44]:
```
print(test_data_prediction)
```

```
[168.76829971  82.18829959 116.23410027 127.5674008  120.62270108
 154.71719784 150.04719887 126.16890035 117.66989876 126.06290049
 116.73160075 172.27110078 141.83419834 167.76609874 115.22450001
 117.50600051 138.24400331 170.43690097 159.44380243 159.35360011
 155.00559987 125.07500044 175.29609979 157.42990403 125.15680053
  93.7665996   78.52829999 120.3315998  119.13719967 167.46479965
  88.14750083 125.56270024  91.13840088 117.60690032 121.03429896
 135.75240091 115.35360115 115.20890059 146.93200024 107.28520116
 104.21120231  87.21809814 126.55120089 117.77010013 152.16679857
 119.47440021 108.43369996 108.11349845  93.16320066 126.93079794
  74.99960049 113.66059923 121.55940049 111.4665992  118.94509913
 121.06329933 159.37650092 167.73280074 147.07139661  85.78759859
  94.37560042  86.77779895  90.62120029 119.01440081 126.31670088
 127.32839977 168.85600031 122.28969894 117.24229909  98.53520055
 167.38750157 142.82689856 132.07460331 121.07910221 121.11059944
 119.49270029 114.57210167 118.45810047 107.2110013  127.85710028
 114.03889951 107.64999997 116.59100088 119.73919877  89.02940102
  88.32229857 146.02410214 127.24129991 113.40510009 110.13299841
 108.06209908  77.77509894 169.22650192 114.03349909 121.72629889
 127.95050204 154.96979768  91.66889952 136.41830156 158.64060388
 125.19920075 125.53520052 130.86430227 114.80760136 119.93300008
  92.06619963 110.38969867 166.63459928 157.69169879 114.14229947
 106.57850129  79.56849964 113.03140032 125.89620076 107.29549897
 119.28880093 155.71030333 159.78199868 120.13549986 134.83220365
 100.9523     117.4757981  119.28400026 113.02250073 102.75959892
 159.99869706  99.22050022 146.93249929 125.49020097 170.16899939
 125.76389855 127.49479673 127.42570164 113.65619936 112.81140055
 123.68629896 102.15189912  89.03199994 124.48209956 102.19799948
 107.17229942 113.78620041 117.0614007   99.53309969 121.69260048
 163.39999899  87.36439868 106.86199979 117.23340072 127.66520092
 124.10780054  80.68509901 120.33230059 157.49299811  87.8069993
 110.33009942 118.93989903 172.42019888 102.90619899 105.32000062
 122.90260034 157.8503978   87.85599817  92.85140048 112.56010026
 176.59000005 114.31519999 119.31880008  94.64340093 125.76210012
 166.36260125 114.82360081 116.90950129  88.32869856 148.71810071
 120.40929917  89.68779989 112.30680003 117.29140024 118.73890125
  88.04859918  94.14340014 116.88480017 118.69490152 120.40890049
 126.78439809 122.03819964 150.37990052 165.07900039 118.54099964
 120.61500153 150.92920043 118.29679902 173.41409921 105.63489942
 104.89170174 149.15490065 113.77410081 124.78420134 147.00900042
 119.6824009  115.49290054 112.59730003 113.57670187 140.65130147
 117.89769785 102.94360028 115.8441012  103.20730149  99.00540049
 117.09020045  90.84119981  91.43770049 153.93619924 102.64239983
 155.08690073 114.46180143 139.45430088  90.14519827 115.52499953
 114.72109993 122.68900026 121.89180011 165.29420179  92.82759962
 135.60660098 121.35249932 120.5186009  104.71969999 141.19590304
 121.62279951 116.58960032 113.52520049 127.03829728 122.39659957
 125.79169926 121.25340035  86.90429909 132.7075016  144.04590242
  92.66679998 160.31379921 157.98170281 126.40809863 165.62089997
 108.72619927 110.34820106 103.61409825  94.36070065 127.6160028
 107.08530045 162.20589988 121.83370021 132.12089978 130.97760242
 160.63280025  90.04709833 174.73430216 128.10750013 126.77749803
  86.54569935 124.66639961 150.14199748  89.68800009 106.93329967
 109.02689969  84.56069911 136.06219932 154.8288023  139.088604
  73.79670045 152.13230116 126.30539976 126.83829983 127.49099877
 108.64109961 156.0903999  114.48390105 117.09570107 125.1347995
 154.0024018  121.30209997 156.44909909  92.9162005  125.46510144
 125.65800017  88.06620098  92.0583991  126.40279933 128.62030397
 113.18700058 117.6246973  120.68320033 127.1277975  119.67520108
 136.10650165  93.88539931 119.75480042 113.12400093  94.22299919
 108.97669968  87.68509918 109.09419943  89.59539957  92.43740036
 131.41210341 162.29870023  89.17760022 119.47550093 133.26340188
 123.971      128.71540204 101.91679841  88.95409901 131.60600069
 119.92569995 108.56150012 168.3552013  115.25880041  86.65519882
 118.97720059  91.03509982 161.84330064 116.55680044 121.60850011
 160.1049977  120.00539935 112.86239938 108.44539871 126.67209979
  75.78290035 103.03459987 127.65950237 121.89919927  92.61340025
 132.09020057 118.1096014  116.04949954 154.44130299 159.38760084
 110.21039933 157.69499813 119.27440082 160.2606009  118.50980065
 157.26660075 115.13729921 116.57600019 149.34569903 114.68170085
 125.78089879 167.29229996 117.98700029 124.8028993  153.29100377
 153.51920237 132.12510088 114.74210067 121.15840197 125.2400005
  89.89210059 123.18149958 154.89610151 111.82430038 106.88409962
 162.16680101 118.5193995  165.67049967 133.92430073 114.91729957
 153.02879861 168.52260006 114.61520012 113.98620136 158.34109893
  85.48089859 126.97430085 127.87730046 128.7544002  124.37360085
 123.84980061  90.5399006  153.12749948  97.06169988 137.61030046
  89.07619948 107.59559994 115.02510059 112.43670069 123.88599937
```

```
 91.51799891 125.47220149 162.37499818 119.79869924 165.08550179
 126.79689784 112.28110026 127.48189938  94.6366988   90.93560009
 103.85459921 120.7429998   82.67939952 126.38529987 160.59110439
 117.32700079 118.34679992 119.98109984 122.56849959 120.06590117
 121.54219995 118.06790094 107.28489998 147.89899909 126.18709791
 115.81800072  73.98550005 127.78900097 154.17810067 123.12140006
 125.56950055  88.95590011 103.16809869 124.25970047 120.29720021
  73.47450088 151.89300019 121.25960077 104.46180029  86.84789774
 115.18059908 172.27199897 119.89320035 159.63209812 113.28109951
 121.20990029 118.80010117  95.94119973 118.78900028 125.96950057
 118.69279937  95.65790051 153.91100209 122.06140012 147.83389998
 158.33530295 113.67220018 122.43029965 150.57819858 127.31410015
 165.67390064 136.41360017 120.08529946 166.70659934 108.52519929
 121.63299867 138.7994014  107.39499866]
```

In [45]:

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

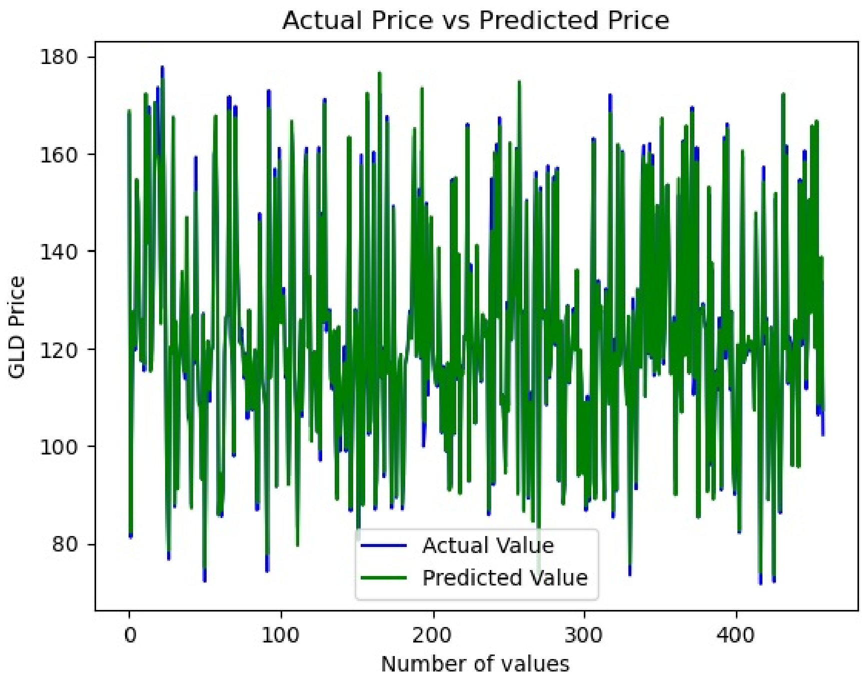
```
R squared error :  0.9884013301809286
```

In [46]:

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

In [47]:

```python
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()
```



In [ ]: