In [11]:

```
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 [12]:

```
%pwd C:\Users\POULAMI\Desktop
```

Out[12]:

```
'C:\\Users\\POULAMI\\Desktop'
```

In [13]:

```
%cd C:\Users\POULAMI\Desktop
```

```
C:\Users\POULAMI\Desktop
```

In [14]:

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

In [15]:

```
gold_data.head(24)
```

Out[15]:

	Date	SPX	GLD	USO	SLV	EUR/USD
0	01-02-2008	1447.160034	84.860001	78.470001	15.180000	1.471692
1	01-03-2008	1447.160034	85.570000	78.370003	15.285000	1.474491
2	01-04-2008	1411.630005	85.129997	77.309998	15.167000	1.475492
3	01-07-2008	1416.180054	84.769997	75.500000	15.053000	1.468299
4	01-08-2008	1390.189941	86.779999	76.059998	15.590000	1.557099
5	01-09-2008	1409.130005	86.550003	75.250000	15.520000	1.466405
6	01-10-2008	1420.329956	88.250000	74.019997	16.061001	1.480100
7	01-11-2008	1401.020020	88.580002	73.089996	16.077000	1.479006
8	1/14/2008	1416.250000	89.540001	74.250000	16.280001	1.486900
9	1/15/2008	1380.949951	87.989998	72.779999	15.834000	1.480210
10	1/16/2008	1373.199951	86.699997	71.849998	15.654000	1.466405
11	1/17/2008	1333.250000	86.500000	71.029999	15.717000	1.464000
12	1/18/2008	1325.189941	87.419998	71.540001	16.030001	1.461796
13	1/22/2008	1310.500000	88.169998	70.550003	15.902000	1.464794
14	1/23/2008	1338.599976	87.889999	69.500000	15.900000	1.463208
15	1/24/2008	1352.069946	90.080002	70.930000	16.299999	1.477410
16	1/25/2008	1330.609985	90.300003	71.910004	16.298000	1.467502
17	1/28/2008	1353.959961	91.750000	72.349998	16.549999	1.478809
18	1/29/2008	1362.300049	91.150002	72.980003	16.534000	1.477192
19	1/30/2008	1355.810059	92.059998	73.080002	16.674999	1.483107
20	1/31/2008	1378.550049	91.400002	72.349998	16.818001	1.486503
21	02-01-2008	1395.420044	89.349998	70.470001	16.618999	1.479991
22	02-04-2008	1380.819946	89.099998	71.370003	16.514999	1.482800
23	02-05-2008	1336.640015	87.680000	70.150002	16.167000	1.463807

In [16]:

```
# print last 5 rows of the dataframe
gold_data.tail()
```

Out[16]:

	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

In [17]:

```
gold_data.shape
```

Out[17]:

```
(2290, 6)
```

In [18]:

```
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: 98.5+ KB
```

In [19]:

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

Out[19]:

```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

In [20]:

```
# getting the statistical measures of the data
gold_data.describe()
```

Out[20]:

	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	22.882499	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

In [21]:

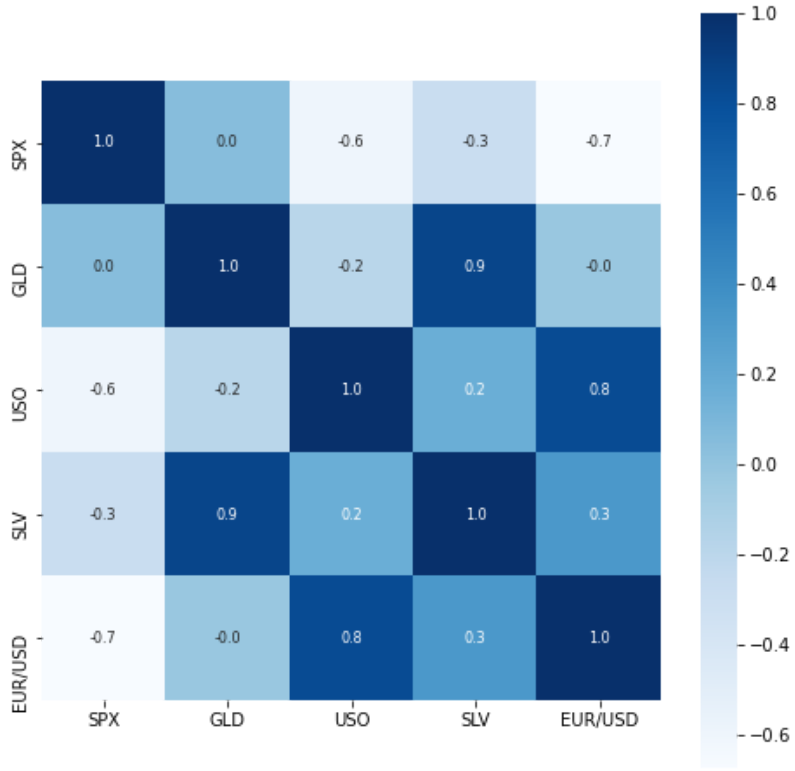
```
correlation = gold_data.corr()
```

In [22]:

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

Out[22]:

```
<AxesSubplot:>
```



In [23]:

```
# 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
```

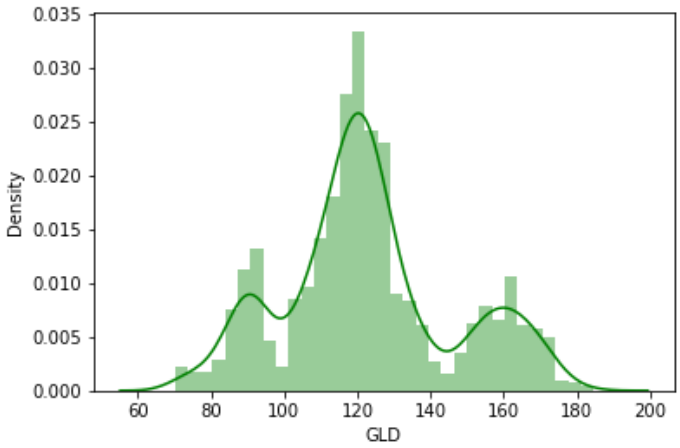
In [27]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `histp
lot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[27]:

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



In [ ]:

```
#Splitting the Features and Target
```

In [41]:

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

In [42]:

```
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 [43]:

```
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 [51]:

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

In [52]:

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

In [53]:

```
regressor.fit(X_train,Y_train)
```

Out[53]:

```
RandomForestRegressor()
```

In [54]:

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

In [55]:

```
print(test_data_prediction)
```

```
[168.63089983  82.19659985 116.32170004 127.71670087 120.84210165
 154.6019986  150.53579889 126.18410003 117.41369875 125.79600125
 116.50860127 171.13290053 141.78359881 167.79709896 115.3353
 117.78630025 140.07560256 170.24530126 158.88330327 159.70569997
 155.10340033 125.09919988 176.81229943 156.85820298 125.23680024
  93.70399956  77.72220007 120.74200027 119.05409936 167.43349989
  87.9992007  125.18130026  91.13400113 117.80240009 121.29189886
 136.26310035 115.57920143 115.17560069 147.72999955 107.59180079
 104.43670237  87.15419807 126.59980045 118.08549982 152.96169925
 119.70240015 108.4669999  108.33229843  93.18690049 127.01549837
  75.06820037 113.75069933 121.16460033 111.23529878 118.8784988
 120.49109917 158.4020997  167.21680129 146.93419694  85.82059894
  94.3832002   86.81169882  90.54050001 119.07130062 126.37990072
 127.61370005 168.91349941 122.28379913 117.21369879  98.50440038
 168.01830084 142.65479821 132.35600181 121.16060252 120.10769963
 119.46860079 114.73900139 118.27200061 107.27920086 127.88170056
 113.91699949 108.08079992 116.6702004  119.75339837  88.8109007
  88.27799863 145.7831021  127.3888003  113.51240028 110.39829862
 108.36589888  77.97959891 170.01490215 113.89069898 121.56929948
 128.04580203 154.81879886  91.63689871 135.24810138 158.43140328
 125.43260058 125.23640055 130.5402013  114.90030103 119.84909988
  92.05339968 110.34089905 167.59919889 156.66349897 114.02799922
 106.58250131  79.38479986 113.15720019 125.72810086 107.24369894
 119.21380101 155.53490327 159.77709863 120.15649999 133.5224032
 101.40240009 117.56479805 119.32820012 112.88450066 102.87609903
 160.4834982   99.30030074 147.10969932 125.83480133 169.79489878
 125.76099887 127.35849727 127.45480214 113.82569906 112.58610078
 123.5209988  102.24419917  89.45449988 124.22719965 101.86509932
 107.24299886 113.58490044 116.95780038  99.20099942 121.88140039
 163.31220004  87.23279873 106.78090037 117.28230066 127.70380094
 124.03950058  80.73579956 120.19940067 157.82059786  87.98339948
 110.26649965 118.70559925 172.24949867 103.08549872 105.44540042
 122.68870021 158.00299788  87.42939856  93.64460069 112.93640008
 177.32139927 114.29069982 119.37970029  94.7109008  125.88470027
 165.39320099 114.92770027 116.76980137  88.27519855 148.9794011
 120.3285995   89.42759984 111.57180028 117.39340017 118.54390072
  88.27879952  94.01719982 116.99820034 118.58140172 120.14020048
 126.86279807 121.87560008 150.1215002  165.27260146 118.59099968
 120.27130136 151.58220074 118.43599911 172.48949924 105.42059922
 105.05660133 149.94140107 113.64350069 124.92850112 147.04090072
 119.74360116 115.24390059 112.7720001  113.40360205 141.43260092
 117.6773979  102.9248006  115.91400111 103.7879018   98.71850055
 117.32300047  90.73899991  91.56510042 153.47139898 102.76869973
 155.02770103 114.29220175 138.14420112  90.20779855 115.4392992
 114.62799999 122.97100045 121.95160028 165.31340175  92.92519956
 135.1883014  121.32739948 120.75840097 104.74709999 140.4467032
 121.5121992  116.5610006  113.51630058 127.26199712 122.57479927
 125.77949974 121.22470032  86.87219903 132.28410133 143.6636017
  92.67649969 155.85269994 158.72030295 126.38519908 165.20199964
 108.69369989 109.92140088 103.64019805  94.1436007  127.5966028
 107.20470072 158.81210041 121.89190022 131.92819948 130.64080113
 160.3521992   90.17369888 175.43180196 128.23090034 126.82629851
  86.28829961 124.3500995  150.1361973   89.67820026 106.95480049
 109.09069986  83.90679897 135.42169912 155.35910232 139.33210315
  74.07730022 152.41950134 125.94089987 126.68450031 127.55479885
 108.62639929 156.44660026 114.52630124 116.96300137 124.92439916
 153.95810116 121.27720005 156.42179874  92.88340035 125.44580111
 125.94160076  88.0644004   92.21039923 126.0435999  128.31190367
 113.1533006  117.64299733 120.93010022 127.35629764 119.08530076
 136.24160035  93.8279992  119.81020026 113.25060093  94.20499964
 108.79630016  87.34179908 109.00469963  89.65529986  92.39390051
 131.32750337 162.38150067  89.27580015 119.6561006  133.26710224
 123.81609998 128.53820211 101.99529864  89.03309861 131.31960012
 119.0807002  108.61449989 168.10970123 115.36040062  86.6820988
 118.91730055  91.02539967 161.96700049 116.56260064 121.59439999
 160.39629757 119.93759933 112.84449937 108.48759856 126.71240059
  75.84640052 102.98749969 127.70490293 121.86149911  92.66640003
 131.62650049 118.05950105 115.88729982 154.47680262 160.23050065
 110.16189963 155.27319777 119.23220089 160.53210066 118.58499996
 156.9290989  115.2254993  116.63900054 148.23249941 114.94580025
```

```
 125.93069849 166.53739898 117.55850013 125.31959913 153.17710375
 153.3363026  132.07699984 114.86870022 121.26010218 124.99760081
  89.67030046 123.0098     155.15290232 111.76240029 106.72680028
 161.68760136 118.50329967 165.68779965 134.09560125 114.8947994
 153.11289866 168.68030011 114.08550051 114.01030123 159.39909959
  85.11889885 127.10150057 127.90950027 128.57320036 124.32860054
 123.78010044  90.56320031 153.05770046  96.92799985 137.28570009
  88.88639868 108.06760008 115.08950068 112.53140149 124.01159931
  91.44219867 125.30240109 162.39059893 119.77689885 165.09820168
 126.88579777 112.39920024 127.61939949  94.93709913  91.03159951
 103.32609908 120.89560027  83.11109934 126.47939999 159.92740474
 117.16940097 118.15849971 120.26159968 122.7995998  120.02240128
 121.58169996 118.31550028 107.1192002  148.40910016 126.15959864
 115.74180065  74.26570014 127.79630089 154.37360053 122.47759994
 125.62870065  88.94010014 103.37169864 124.83000066 120.22050054
  73.48850066 151.98539989 121.20340039 104.67950007  86.44839798
 115.10639925 172.0925991  119.66100022 159.26189775 113.0821993
 121.39109984 118.42810113  95.95119981 118.62340039 125.77370051
 118.6024993   95.83040054 153.85350195 122.04660027 147.0390995
 159.24880254 113.75820005 122.49359951 150.34679868 127.53840047
 165.78610034 135.95340047 120.02389948 167.18729807 108.28649948
 121.76039884 137.89460145 107.23709904]
```

In [56]:

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

```
R squared error :  0.9895917869505301
```

In [ ]:

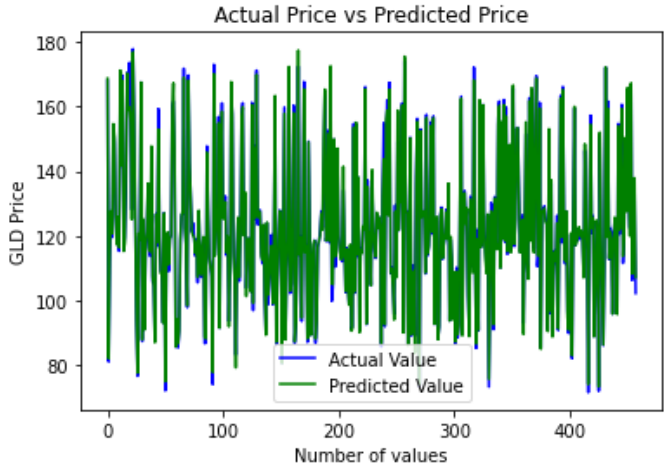
```
#Compare the Actual Values and Predicted Values in a Plot
```

In [58]:

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

In [59]:

```
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 [ ]: