GOLD PRICE PREDICTION

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
```

In [10]:

```
gold_df=pd.read_csv("C:\\Users\\srujani\\OneDrive\\Documents\\gld_price_data.csv")
print(gold_df)
```

```
           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 x 6 columns]
```

In [11]:

```
gold_df.shape
```

Out[11]:

```
(2290, 6)
```

In [12]:

```
gold_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]:

```
gold_df.isnull().sum()
```

Out[13]:

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

In [14]:

```
gold_df.describe()
```

Out[14]:

	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 [16]:

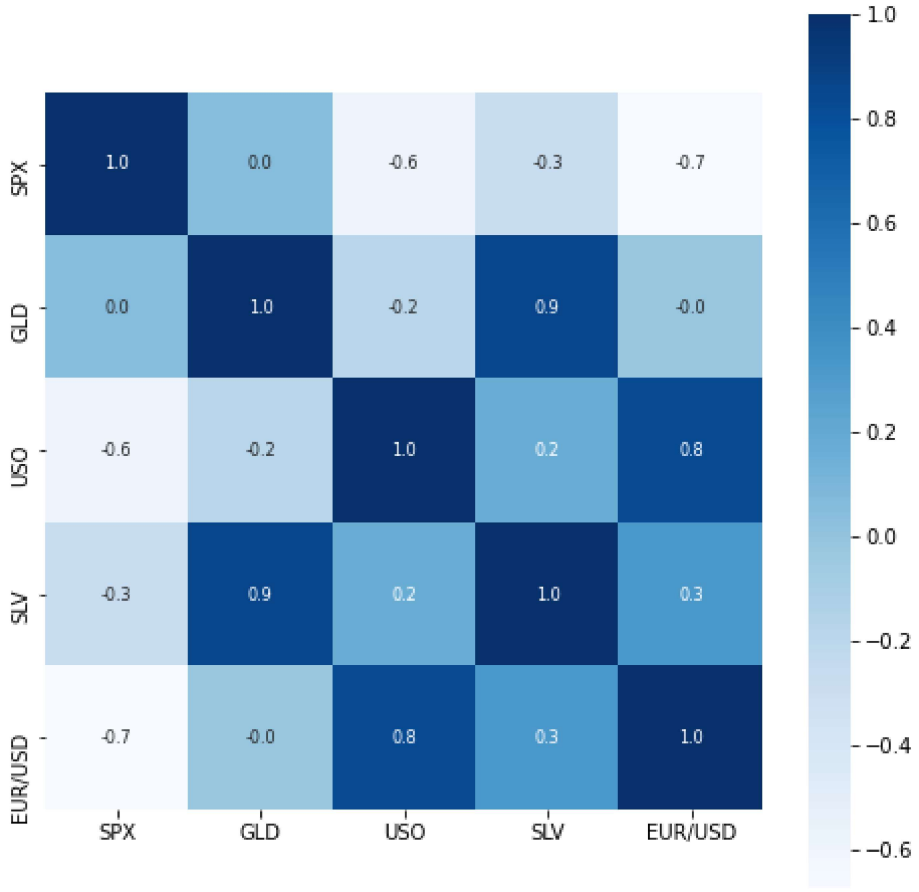
```
correlation =gold_df.corr()
```

In [17]:

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

Out[17]:

<AxesSubplot:>



In [18]:

```
print(correlation['GLD'])
```

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

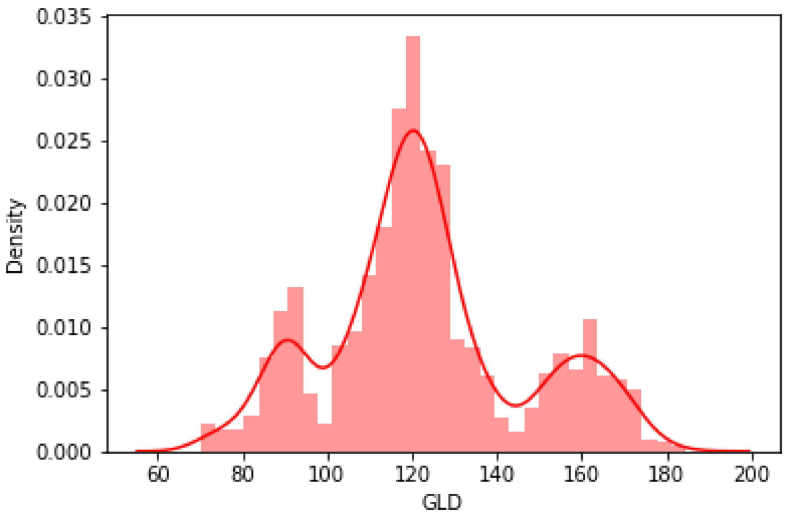
In [19]:

```
sns.distplot(gold_df['GLD'],color="red")
```

```
C:\Users\srujani\anakonda\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[19]:

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



In [20]:

```
x = gold_df.drop(['Date','GLD'], axis=1)
y = gold_df['GLD']
```

In [21]:

```
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 [22]:

```
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 [23]:

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=2)
```

In [24]:

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

In [25]:

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

Out[25]:

```
RandomForestRegressor()
```

In [26]:

```
test_data_prediction = regressor.predict(X_test)
```

In [27]:

```
print(test_data_prediction)
```

```
[168.50499927  81.98709993 116.1593001  127.57920074 120.84680122
 154.65399746 150.41439857 126.08380046 117.58049866 125.92950093
 116.75760124 171.82120102 141.17899816 167.95409919 114.80310026
 117.4666004  139.41580341 170.0273002  159.55000285 158.52929942
 155.12980034 125.0294     176.06029944 157.4578035  125.1864002
  93.85949932  76.75850045 120.46250006 119.15389971 167.54350032
  88.19130042 125.32050024  91.32610081 117.6437003  121.10049924
 136.21810017 115.3838013  115.34360066 145.91399981 107.3107009
 103.91520211  87.23179789 126.43490074 117.72679999 153.56339923
 119.58189998 108.45819918 108.12069859  93.18590041 127.04259792
  74.80910029 113.73849983 121.37629971 111.44259946 118.88179877
 120.58779934 159.18250024 168.30290132 146.98889663  85.67459845
  94.42060036  86.86259917  90.35480062 118.95990091 126.45980039
 127.51100015 170.38040018 122.17099919 117.19299918  98.46620033
 167.95490174 142.51819684 131.29970243 121.28240187 121.0395994
 119.67130093 114.72290139 118.36430021 107.24790117 127.92620084
 114.05789972 107.75050015 116.81400047 119.54969884  89.15020083
  88.20269867 147.20230226 127.32990056 113.43230024 110.02109845
 108.25459906  77.37039931 170.10770186 114.13769911 121.61859923
 127.87570202 154.97639788  91.87849999 136.15170143 158.70040348
 125.01690054 125.43550042 130.75940145 115.00480086 119.82810002
  92.14109986 110.21609878 168.52829923 157.67779922 114.23569936
 106.64070133  79.40189985 113.26690015 125.77990057 106.91399966
 119.18770056 156.19280326 159.59749979 120.23550015 134.60980358
 101.43639989 117.7110978  119.25520021 113.1149006  102.83049903
 160.41779803  98.88730063 148.13769922 125.75350124 170.12519888
 125.52099912 127.4748969  127.20010156 113.83359927 113.22760065
 123.710299   102.07529894  88.84260023 124.1989997  101.80469952
 107.18549909 113.55800077 117.3704006   99.19499969 121.87820047
 162.75919821  87.38479846 106.96130007 117.22960077 127.7265015
 124.15560044  80.79409936 120.2288007  157.94919755  87.9899994
 110.4596994  118.7790989  172.58849879 103.04609886 105.03730061
 122.49919999 158.40579707  87.56719821  93.46640041 112.71110042
 176.90349917 114.3546998  119.33350027  94.84850115 125.83419976
 166.18790067 115.1921003  116.63300143  88.2310986  149.1842004
 120.33299976  89.57000021 112.80789985 117.53560074 118.82170118
  88.21079924  94.11800016 116.89059985 118.49560179 120.3795009
 126.78209852 121.89599987 151.05890015 166.12730012 118.51199976
 120.21420144 151.23880053 118.22319921 172.89299876 105.25989936
 104.85960179 149.37330058 113.65280072 124.807501   147.28180012
 119.71180166 115.33490043 112.99709988 113.43490187 142.36240094
 117.80689773 102.85190017 115.84060084 103.26810163  98.7491003
 117.42600116  90.59430023  91.79320037 153.53419898 102.75249949
 154.78920069 114.33870176 138.23450103  90.10669822 115.57509944
 115.04229947 122.68610029 121.82730051 165.31130127  92.72439922
 135.31320115 121.32399944 120.79930028 104.96949989 141.66930242
 122.1743993  116.81570054 113.89150072 127.11099731 122.72079949
 125.80719889 121.22470068  86.92849928 132.3096014  143.59580214
  92.8249995  159.37459895 158.72230275 126.3437993  165.81409929
 108.9368997  109.69500106 103.74179795  94.39120069 127.6762025
 106.92600051 162.03709912 121.74650042 131.93039912 130.70280143
 160.59649975  90.15089856 175.36060218 128.06800047 126.7646987
  86.37189909 124.54439966 150.07409693  89.60020026 107.05910022
 108.95009962  84.28279918 135.76289957 154.48680173 139.54780308
  73.91960054 153.06280106 126.2419006  126.77239999 127.48529864
```

```
 108.47359937 156.54370065 114.66030106 116.90920134 125.4091997
 154.16470184 121.32799972 156.42329883  92.79080033 125.56120131
 125.68370036  87.8470005   92.11709932 126.10059962 128.35760363
 113.06260024 117.49609735 121.01519994 127.1376988  119.76240085
 136.35440108  93.82529905 119.55830067 113.14630121  94.21559925
 108.82399995  87.62859927 109.12549906  89.67009968  92.33899992
 131.49910308 162.27060118  89.32959994 119.44070081 133.51410156
 123.85999999 128.26180213 101.91299844  89.22189865 131.44900165
 119.71100043 108.55969921 170.49240028 115.2816003   86.6778988
 118.94180052  91.09889964 161.89650066 116.6406005  121.65789973
 160.14519758 120.0338994  112.6680997  108.5763988  126.8374009
  76.30190008 103.03989976 127.61740254 121.60149902  92.45509969
 132.18590066 118.06210108 115.99060012 154.43070312 159.75480073
 110.31399906 154.84569785 119.39550083 160.56230061 118.35150061
 156.82800041 115.16529951 116.2814002  149.80329929 115.04840054
 126.00349887 165.91849917 117.58589999 125.36799926 153.09270363
 153.50890225 132.04139984 114.78700056 121.27810173 125.05650083
  89.71010042 123.28320007 154.77440185 111.64220044 106.78969993
 162.00350119 118.44179989 165.75089977 134.101301   114.91319977
 153.1558     168.57630066 115.10609999 113.97400137 159.29109913
  85.32939913 127.09560051 127.9588003  128.89179944 124.43660068
 123.94610122  90.44510041 153.16730084  96.89939985 137.76950001
  89.26419937 107.64920024 115.09700046 112.96890096 124.57709908
  91.4375986  125.34290109 162.41549911 119.93149885 165.00190103
 126.78629835 112.51879996 127.58019941  94.79419881  90.75529976
 102.66849917 120.84450005  83.0106995  126.27300001 160.53240486
 117.06730107 118.26259965 120.05530015 122.56749985 120.10070118
 121.50029981 118.15500065 106.96769979 148.61540001 126.16199844
 115.72830078  73.98529992 127.85890084 154.78060066 122.347
 125.56610059  88.89489982 102.87019826 124.33470025 120.2731002
  73.40040068 151.5844     121.19099986 104.57970002  86.70609787
 115.13769959 172.12689901 120.16690046 159.0330983  113.18289941
 121.20900001 118.29430093  95.99759987 118.69629981 126.05400046
 118.64839958  95.71270048 153.94970176 122.38670018 147.60240018
 159.14740296 114.06540035 122.48869924 150.53269845 126.99480036
 166.00460041 135.97710029 120.06879946 167.74079891 108.35749934
 121.81719833 138.17610035 106.67389894]
```

In [28]:

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

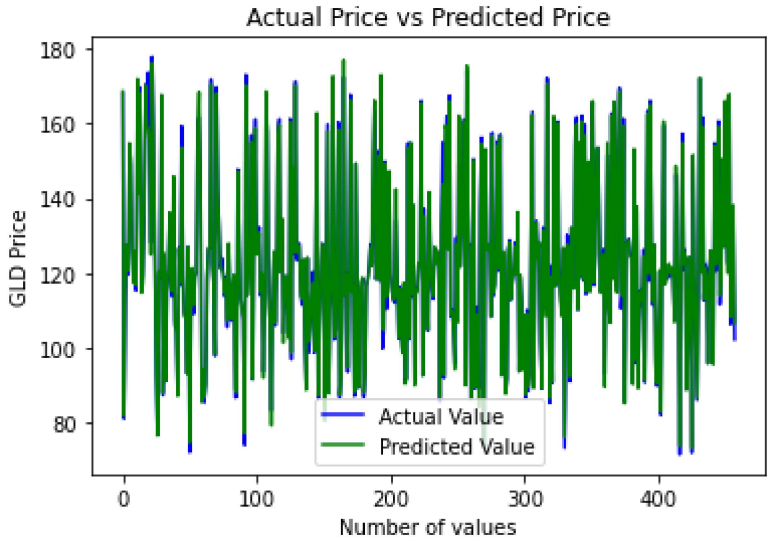
```
R squared error :  0.9890422376683814
```

In [29]:

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

In [30]:

```
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 [ ]:

```
```