Importing the Libraries

In [31]:

```
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
from sklearn.linear_model import LinearRegression
import warnings
warnings.simplefilter("ignore")
```

Data Collection and Processing

In [37]:

```
# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv(r'C:\Users\student\Downloads\datasets\gld_price_data.csv' )
```

In [5]:

```
# print first 5 rows in the dataframe
gold_data.head()
```

Out[5]:

	Date	SPX	GLD	USO	SLV	EUR/USD
4	1/8/2008	1390.189941	86.779999	76.059998	15.590	1.557099

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

	Date	SPX	GLD	USO	SLV	EUR/USD
2285	5/8/2018	2671.919922	124.589996	14.0600	15.5100	1.186789
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

In [6]: Out[6]:

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 [7]:

```
# number of rows and columns
gold_data.shape
```

Out[7]: (2290, 6)

In [8]:

```
# 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):
```

```
Date       2290 non-null object SPX
2290 non-null float64
GLD        2290 non-null float64
USO        2290 non-null float64
SLV        2290 non-null float64
EUR/USD    2290 non-null float64
dtypes: float64(5), object(1)
memory usage: 98.4+ KB
```

In [9]:

```
# checking the number of missing values
gold_data.isnull().sum()
```

```
dtype: int64
```

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

Out[9]:

```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
```

In [10]:

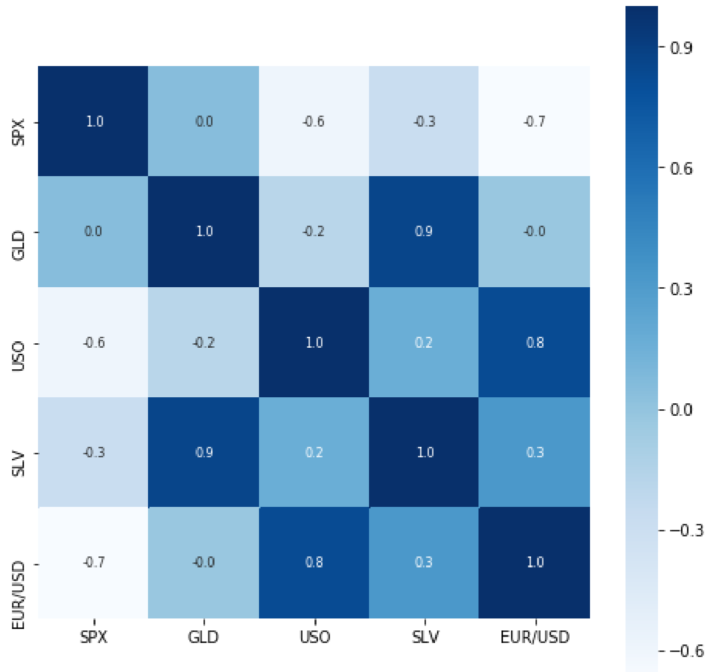
Out[10]:

	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	1.303296
75%	2073.010070	132.840004	37.827501	22.882499	1.369971
max	2872.870117	184.589996	117.480003	47.259998	1.598798

In [12]:
```
correlation = gold_data.corr()
```

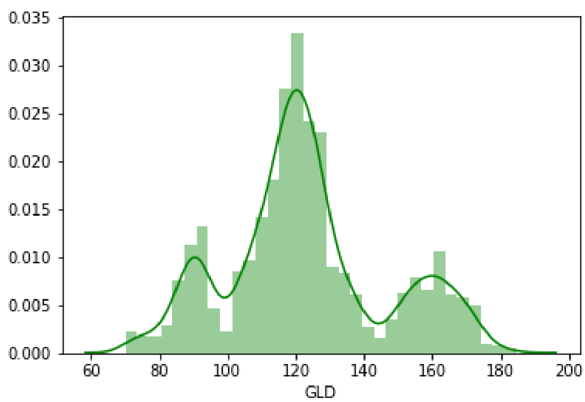
In [13]:
```
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0xd 0d050>



In [19]:
```
# checking the distribution of the GLD Price
sns.distplot(gold_data['GLD'],color='green')
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xc 36f90>



Splitting the Features and Target

In [15]:
```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

In [21]:
```
print(X.head())
```

```
           SPX        USO     SLV   EUR/USD
0  1447.160034  78.470001  15.180  1.471692
1  1447.160034  78.370003  15.285  1.474491
2  1411.630005  77.309998  15.167  1.475492
3  1416.180054  75.500000  15.053  1.468299
4  1390.189941  76.059998  15.590  1.557099
```

In [22]:
```
print(Y.head())
```

```
0    84.860001
1    85.570000
2    85.129997
3    84.769997
4    86.779999
Name: GLD, dtype: float64
```

Splitting into Training data and Test Data

In [23]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)
```

Model Training: Random Forest Regressor

In [24]:
```
regressor = RandomForestRegressor(n_estimators=100)
```

In [25]:
```
# training the model
regressor.fit(X_train,Y_train)
```

Out[25]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
 max_features='auto', max_leaf_nodes=None,
 min_impurity_decrease=0.0, min_impurity_split=None,
 min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0,          n_estimators=100,          n_jobs=1,
 oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Model Evaluation

In [40]:
```
# prediction on Test Data
test_data_prediction = regressor.predict(X_test)
print(test_data_prediction)
```

```
[168.13929893  82.14339969 115.66279962 127.64870078 120.7304014
154.67709822 150.33269897 126.17520013 117.57609889 126.05390106
 116.72890108 171.53750074 141.78859837 168.0252986  115.09040028
 117.49100029 139.86340162 170.32390114 159.76010294 157.86399921
 155.24900034 125.34480028 175.34269984 157.282103   125.30450053
  93.66129965  77.13250036 120.39639976 119.13429947 167.47699921
  88.23619996 125.23730027  91.26890089 117.75410019 121.09049914
 137.09230124 115.42470115 115.52030049 149.48109938 107.22660108
 104.54110232  87.07879785 126.52320012 118.36309998 153.64219945
 119.53159991 108.33759993 107.93369829  93.22950029 127.16529778
  74.60520062 113.73939975 121.15400018 111.23839928 118.87099897
 120.98429926 160.02199942 168.98030136 146.93619671  85.82709856
  94.33140051  86.92179856  90.5422003  118.88340063 126.4502005
 127.40329955 170.35210031 122.32669923 117.550799    98.68900041
 168.47000198 143.03319783 132.19210278 121.21680197 120.28409959
 119.58720073 114.44810174 118.36800039 107.37610113 127.86010017
 113.98689943 107.82429981 116.7754003  119.51399905  89.15630054
  88.30449871 146.41290236 127.17359977 113.11980048 110.33749847
 108.244699    77.34339924 169.51040189 114.16259918 121.55329913
 127.87640171 154.74449802  91.85799911 136.15530127 158.92590312
 125.60310065 125.24910079 130.69990219 114.76920131 119.84109965
  92.10529986 110.3799987  166.77209989 157.5053987  114.44589981
 106.76940118  79.59899988 113.2464004  125.79700058 107.31869931
 119.24990104 155.93810319 159.98219959 120.13359988 135.31110317
 101.61379973 117.46249799 119.4356004  113.07320087 102.80299951
 160.1394983   98.94250032 149.34909842 125.76090098 169.82779885
 125.7306985  127.32639744 127.39700168 113.70269947 112.89060044
 123.65749911 102.18289891  89.30659985 124.35509944 100.85899941
 107.19739912 113.30300053 117.20360074  99.12359941 121.54060065
 163.0720993   87.25619894 106.57519978 116.99210126 127.68670093
 123.94400047  80.77269935 120.29480051 158.49229844  87.85979926
 110.32869935 119.02899896 172.07239846 102.99169885 106.20550036
 122.4297003  158.56939768  87.64249821  93.42490071 112.5256005
 176.75690007 114.29789979 119.3760002   94.73720086 125.68110014
 166.18940137 114.93590085 116.91490126  88.36559869 148.64740079
 120.40999954  89.47449987 112.68920004 117.41510052 118.77500125
  88.16269958  94.08049986 117.06889978 118.75800191 120.39280044
 126.75709814 121.94529957 150.35119991 165.11700099 118.61629967
 120.20550104 150.96400072 118.17669914 172.38249838 105.52499936
 104.99050141 149.1596011  113.65250099 124.80490116 148.38479986
 119.81500105 115.59930052 112.85290004 113.55240201 141.61290141
 117.74819782 102.94209994 115.82280094 104.15090195  98.75490047
 117.22750074  90.82320007  91.68110019 153.49489902 102.64129988
 154.99050056 114.24990166 138.3433014   90.17229795 115.50699912
 114.50289964 123.05870035 121.83040014 165.34700136  92.88199936
 134.97600143 121.29729977 120.62250098 104.38540031 141.56550298
 121.403499   116.63670031 113.66530115 127.21469713 122.70029922
 125.77729965 121.18070089  86.94869901 133.09710205 143.47170198
  92.61339987 158.61059937 158.63410176 126.513999   164.3133996
 108.95289925 109.84730102 103.56839854  94.28940113 127.88510297
 107.12620041 162.56929961 121.59910047 131.95910017 130.82030185
 161.06689911  90.16159816 174.87680154 127.86909993 126.89289824
  86.44359969 124.50619883 150.04929769  89.57350012 106.63499982
 109.10579986  84.30639886 136.47990001 154.81750211 138.89710317
  74.33630007 151.93500151 126.0211     126.75060013 127.53049888
 108.48509938 156.24560021 114.48740131 116.9303012  125.07469936
 153.99020184 121.35769978 156.44509902  92.98120068 125.51170135
 125.55060046  87.95060067  92.18079912 126.25999946 128.47250416
 113.30180103 117.41379698 120.76570057 127.14979806 119.27500095
 136.7266007   93.84339912 119.77250037 112.94220125  94.2506993
 108.90659963  87.77009902 108.85629971  89.61199969  92.50570011
 131.7913034  162.50900034  89.35690004 119.49420053 133.20650223
 123.95690016 128.3693026  101.8224983   89.2389984  132.05150029
 119.50850007 108.5994003  169.40940009 115.13430006  86.6144988
```

```
 118.9640007   91.04839961 161.88140048 116.65940062 121.56099985
 160.41359828 120.03599945 112.70289961 108.41939857 126.90470031
 76.32500026 102.9279998  127.79030266 121.88729921  92.59609977
 132.44160097 118.08390083 115.83569964 154.94000243 159.45110078
 109.97399981 157.59019758 119.31580099 160.7691013  118.44090013
 157.97769888 115.13439918 116.51460047 149.88429934 114.8159008
 125.84159852 166.0188994  117.69930019 125.11149927 153.32710345
 153.33270277 132.15230036 114.84930032 121.30950156 125.28290102
 89.66200073 122.81389988 154.37540148 111.69100035 106.49309977
 161.71280111 118.54159988 165.66220029 134.11370121 114.98919952
 153.0406988  168.7329007  114.34880038 114.03560122 160.64219901
 85.39129882 127.05570069 127.9787007  128.94159948 124.45510054
 123.93080087  90.89910094 152.7377002   97.14319967 138.34050037
  89.07239939 107.82519987 115.08230035 112.88540087 124.01349919
  91.50589838 125.27810097 162.26829902 119.81559881 165.10360076
 126.78999806 112.60329998 127.54839908  94.75809916  90.80089998
 103.28349907 120.85910018  82.99889941 126.40019962 160.68580441
 117.21840087 118.31700002 120.014      122.92549977 120.03980137
 121.51980006 118.28570077 106.96209998 147.83189978 126.29889801
 115.93860093  74.17890003 127.8012006  154.22100041 122.5528003
 125.62100084  88.85620031 104.63079896 124.39340056 120.32110018
 73.42960074 151.47310037 120.90980074 104.6653999   86.24359782
 115.00239857 172.17849806 120.05290039 159.92139771 113.18340017
 120.90870022 118.95400108  96.04529999 119.09280007 125.95280016
 118.55119965  95.80710058 153.9803019  122.23360016 147.48499951
 159.43570346 113.97320046 122.37879956 151.04549795 126.81130076
 165.86210041 136.20350039 119.95319973 167.03509851 108.38189929
 121.91009804 140.85780127 106.46729904]
```

In [28]:

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

```
R squared error :  0.9881058280225586
```

Compare the Actual Values and Predicted Values in a Plot

In [32]:

```
model1 = LinearRegression()
model1
```

Out[32]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [33]:

```
model1.fit(X_train, Y_train)
```

Out[33]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [34]:

```
y_pred = model1.predict(X_test)
y_pred
```

Out[34]:

```
array([159.45290968,   81.50858067, 113.4868037 , 128.55153817,
 126.20403783, 141.31818338, 165.49416492, 124.44197659,
       113.95389904, 122.9965895 , 113.95492464, 174.89529849,
       132.66890174, 159.24793304, 118.8791679 , 122.14315717,
       150.37153054, 161.27175405, 152.2792036 , 162.19946709,
       145.88372183, 118.07669395, 180.21729122, 178.13331554,
       123.51701986,  98.82914322,  76.63486248, 119.38435511,
       113.85027665, 159.25706609,  93.76037617, 120.05334649,
 88.61086583, 114.28801659, 112.77954274, 144.19518102,       118.78252489,
 115.87334354, 144.06409553, 113.81864246,
       100.49676727,  89.42842238, 120.36720699, 110.47635305,
```

```
      146.4894628 , 119.30321261, 110.99311348, 111.3386287 ,
96.48139076, 123.40842816,  79.51413425, 115.36111603,       121.78896764,
114.55367722, 120.9074622 , 117.10612464,
      152.31198525, 182.7735091 , 196.73570503,  91.06670951,
      101.92097594,  87.48443506,  94.33158785, 119.17357764,
      121.8935122 , 125.85678089, 163.2699701 , 122.91100556,
      113.55098736, 104.06905043, 155.16775229, 176.87972077,
      125.62460858, 113.85896766, 119.69577146, 120.03159609,
      117 85194226 118 35897733 113 72954121 129 97412373
```

In [35]:

```
comp = pd.DataFrame({'actual':Y_test,'pridict':y_pred})
comp['error']=comp['actual']-comp['pridict']
comp
```

Out[35]:

	actual	pridict	error
0	168.020004	159.452910	8.567094
1	81.230003	81.508581	-0.278578
2	112.320000	113.486804	-1.166804
3	127.589996	128.551538	-0.961542
4	119.620003	126.204038	-6.584035
5	154.210007	141.318183	12.891824
6	148.910004	165.494165	-16.584161
7	126.190002	124.441977	1.748025
8	117.470001	113.953899	3.516102
9	125.739998	122.996589	2.743409
10	115.379997	113.954925	1.425072
11	167.119995	174.895298	-7.775303
12	141.630005	132.668902	8.961103
13	169.559998	159.247933	10.312065
14	115.599998	118.879168	-3.279170
15	119.669998	122.143157	-2.473159
16	132.949997	150.371531	-17.421534
17	170.399994	161.271754	9.128240
18	159.369995	152.279204	7.090791
19	173.529999	162.199467	11.330532
20	154.720001	145.883722	8.836279
21	128.119995	118.076694	10.043301
22	177.720001	180.217291	-2.497290
23	157.190002	178.133316	-20.943314
24	125.309998	123.517020	1.792978
25	93.400002	98.829143	-5.429141
26	76.790001	76.634862	0.155139
27	119.690002	119.384355	0.305647
28	118.989998	113.850277	5.139721
29	167.389999	159.257066	8.132933
...	...	...	...
428	104.099998	113.075384	-8.975386
429	86.230003	88.391806	-2.161803
430	113.580002	114.033422	-0.453420
431	172.100006	163.405110	8.694896
432	121.480003	125.205805	-3.725802
433	161.539993	152.114434	9.425559

	actual	pridict	error
434	112.769997	113.253199	-0.483202
435	122.419998	118.624320	3.795678
436	120.940002	117.286420	3.653582
437	95.989998	104.680788	-8.690790
438	120.760002	118.419216	2.340786
439	125.320000	126.050353	-0.730353
440	118.080002	120.412925	-2.332923
441	97.730003	100.695334	-2.965331
442	154.649994	143.832804	10.817190
443	120.589996	121.012654	-0.422658
444	143.470001	160.522755	-17.052754
445	160.589996	149.574142	11.015854
446	111.629997	108.422241	3.207756
447	122.120003	120.971090	1.148913
448	146.240005	158.885207	-12.645202
449	127.959999	118.214578	9.745421
450	164.119995	158.116072	6.003923
451	133.429993	143.837791	-10.407798
452	122.379997	120.478000	1.901997
453	166.380005	159.866392	6.513613
454	106.379997	111.240450	-4.860453
455	122.239998	126.454208	-4.214210
456	133.830002	144.935055	-11.105053
457	102.360001	106.948384	-4.588383

458 rows × 3 columns

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()
```