In [7]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
```

In [9]:
```
gold=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\gld_price_data.csv")
gold.head()
```

Out[9]:

	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 [10]:
```
#shape of date
gold.shape
```

Out[10]:
```
(2290, 6)
```

In [12]:
```
##describe mathamatical data
gold.describe()
```

Out[12]:

	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 [13]:
```
#information about data
gold.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
```

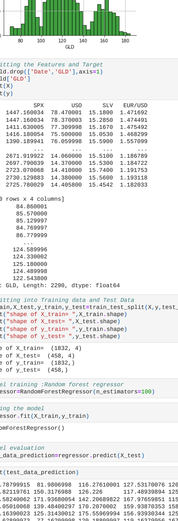
In [14]:
```
#find the null value in data
gold.isnull().sum()
```

Out[14]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

In [19]:
```
#Correlation:1------Positive Correlation    2------Negative Correlation
correlation=gold.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
```

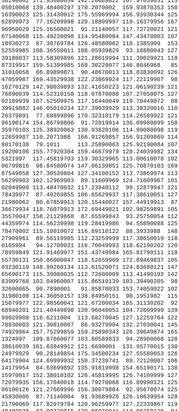
In [23]:
```
# constructing a heatmap to understand the correlation
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Purples')
plt.show()
```



In [27]:
```
print(correlation['GLD'])
```

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

In [30]:
```
#checking the distribution of gold price
sns.displot(gold['GLD'],color='green')
plt.grid(True)
plt.show()
```



In [35]:
```
#Splitting the Features and Target
X=gold.drop(['Date','GLD'],axis=1)
y=gold['GLD']
print(X)
print(y)
```

```
             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]
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 [34]:
```
#Splitting into Training data and Test Data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (1832, 4)
shape of X_test=  (458, 4)
shape of y_train=  (1832,)
shape of y_test=  (458,)
```

In [36]:
```
#model training :Random forest regressor
regressor=RandomForestRegressor(n_estimators=100)
```

In [37]:
```
#traing the model
regressor.fit(X_train,y_train)
```

Out[37]:
```
RandomForestRegressor()
```

In [38]:
```
#model evaluation
test_data_prediction=regressor.predict(X_test)
```

In [39]:
```
print(test_data_prediction)
```

```
[168.78799915  81.8986998  116.27610001 127.53170076 128.62820169
 154.82119761 150.3176988  126.226      117.48939894 125.84880126
 116.58240062 171.93680054 142.20689822 167.97659855 115.19410017
 118.05010068 139.48400297 170.3970002  159.92837053 158.22549947
 155.16390023 125.31430012 175.55969994 156.93903344 125.15010058
  93.62899973  77.16209998 120.15895997 119.16379956 167.44410017
  88.79050029 125.16500021  91.21140057 117.72720021 121.11299927
 136.67140068 115.49220098 114.95480084 147.43470003 107.23870079
 104.18930273  87.30769784 126.48580062 118.1385999  153.80769898
 119.52559985 108.36550011 108.95939829  93.18880043 127.22849736
  75.39180037 113.58309896 121.28019944 111.39029921 118.91049889
 120.87319917 159.51399985 160.30220077 146.80449868  85.99680857
  94.31010058  86.89989971  90.48670013 118.83830092 126.44540059
 127.47059987 169.43529938 122.23869924 117.22119997  99.61460038
 168.16270129 142.99030893 132.41650223 121.06190239 121.23819955
 119.76890039 114.52310158 118.07879888 107.27050075 127.87340092
 114.02189939 107.52509975 117.16440049 119.70449872  88.79030848
  88.39910082 146.05010234 127.39039929 113.38320016 116.41159817
 108.26378891  77.68899896 170.32310179 114.26589922 121.61799893
 127.96190174 154.80799896  91.72019914 138.09980099 158.67300378
 125.35970103 125.38920063 130.93820106 114.99080098 119.99990002
  92.12659987 110.2071988  166.91260857 156.91209869 114.34419905
 106.80170138  79.1011     113.25890063 125.92190084 107.28179942
 119.19200106 155.77920304 159.46679979 120.24039982 134.32700266
 101.5321997  117.45819793 119.30329965 113.00613078 102.80789921
 160.06779816  98.84580074 147.06139851 125.70870103 169.44699951
 125.87549858 127.30539804 127.34100152 113.73869974 113.47500092
 121.56299983 102.12909903  89.11609969 124.71609967 101.72819934
 107.02049909 113.48470032 117.23940112  99.12879947 121.7869003
 163.78439977  87.40269855 106.65623937 117.30610051 127.56420145
 124.21980062  80.67859913 120.15440027 157.44919913  87.81429956
 110.30679934 118.76079913 172.69440021 102.98259893 105.32430069
 122.35570047 158.21229868  87.65599843  93.25750054 112.90470019
 177.44359974 114.56239908 119.26419866  94.55890088 125.78470934
 165.78470002 115.10010072 116.89110222  88.3933388  148.71320114
 120.27909961  89.56119985 112.23259999 117.38650818 118.76790137
  88.9105994   94.12700031 116.70943993 118.62139202 129.31830038
 128.78958849 121.91469977 151.43749984 165.81799111 118.6174995
 120.55738131 150.86000947 118.52653909 172.89468837 105.80009951
 105.63230119 148.99260134 113.61526071 124.83680121 147.27116893
 119.65690173 115.39080035 112.72600009 113.41490192 142.04030126
 117.83909768 103.04960007 115.86510139 103.39490295  98.5913061
 117.32600065  90.7890001   91.85870033 153.74059922 102.63939965
 154.31980108 114.36050157 138.84950151  90.1951882  115.43089939
 114.15097977 122.98560041 121.67290034 165.31130202  92.9012995
 135.68940201 121.40443898 120.96040851 104.72609999 130.82750322
 121.99029908 116.6211804  113.68270845 127.12259764 122.6367996
 125.78830003 121.30810007  86.93279004 132.27030041 145.36080201
  92.74029964 157.75299859 159.25890343 126.39648874 165.23010033
 109.1324997  109.87660077 103.80569833  94.28900068 128.2370029
 107.38010039 161.63949912 121.6690001  131.95770015 139.9020008
 161.24979929  90.28149854 175.34500234 127.55580053 126.8877983
  86.64179984 124.69999932 150.37239741  89.72120007 106.98269933
 109.14179954  84.63699892 135.91819998 154.65190171 138.42640384
  74.15970017 152.38010102 126.45819995 126.74189999 127.50879922
 108.72079935 156.17040018 114.79270068 116.89990121 125.5951996
 154.06100126 121.27660996 156.30079884  92.95670074 125.4674015
 125.45930006  87.71140004  91.93689926 126.10639954 128.31900344
 113.21190069 117.82979784 120.96259977 127.22339867 119.6960401
 136.40400076  93.90729915 120.06070041 113.06750126  94.28999922
 109.8977999   87.8494952  109.81659956  89.61889995  92.51700016
 131.79930281 162.55839987  89.35969993 119.35910684 133.40560172
 123.73410037 128.29270229 101.98033062  89.06249883 131.3861001
 119.63720046 108.35429989 169.10680076 115.23920037  86.59089906
 116.76040855 102.41100003 127.81339913  94.81199924  90.9126002
 103.51480889 120.84199963  83.13419963 126.27129952 160.62040468
 117.48280108 118.28030011 119.78189985 122.77137985 120.08570122
 121.57289974 118.11080011 107.14789982 148.1484002  126.19829813
 115.84330093  73.80589996 127.84010051 153.91070044 121.82050007
 125.63160054  88.82240042 102.43959861 124.43950056 120.33110072
  73.26330084 151.3331007  123.24890099 104.81240616  86.5267977
 115.23299911 172.19599842 119.85910003 160.70169803 113.30659938
 121.47870017 118.52770105  95.9157988  118.62940017 125.92390051
 118.50169915  95.96830075 154.22340172 122.0560999  147.49879986
 158.92740302 114.1210002  122.59419932 148.96869747 126.99430816
 165.93350026 135.64640042 120.176699   167.6884976  108.31119895
 121.90078851 140.08430121 106.94199965]
```

In [40]:
```
# R squared error
error_square=metrics.r2_score(y_test,test_data_prediction)
```

In [44]:
```
print("R squared error :" ,error_square)
```

```
R squared error : 0.9891457076744411
```

In [45]:
```
#compare the actual values and predict value in a plot
y_test=list(y_test)
```

In [55]:
```
plt.figure(figsize = (20,7))
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.grid(True)
plt.show()
```