NAME: VANDHITHA R S

EMAIL: vandithachinnu77777@gmail.com (mailto:vandithachinnu77777@gmail.com)

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 [2]:

```
# Loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('gld_price_data.csv')
```

In [3]:

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

Out[3]:

	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 [4]:

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

Out[4]:

	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 [5]:

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

Out[5]:

```
(2290, 6)
```

In [6]:

```
# getting some basic information 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 [7]:

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

Out[7]:

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

In [8]:

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

Out[8]:

	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 [9]:

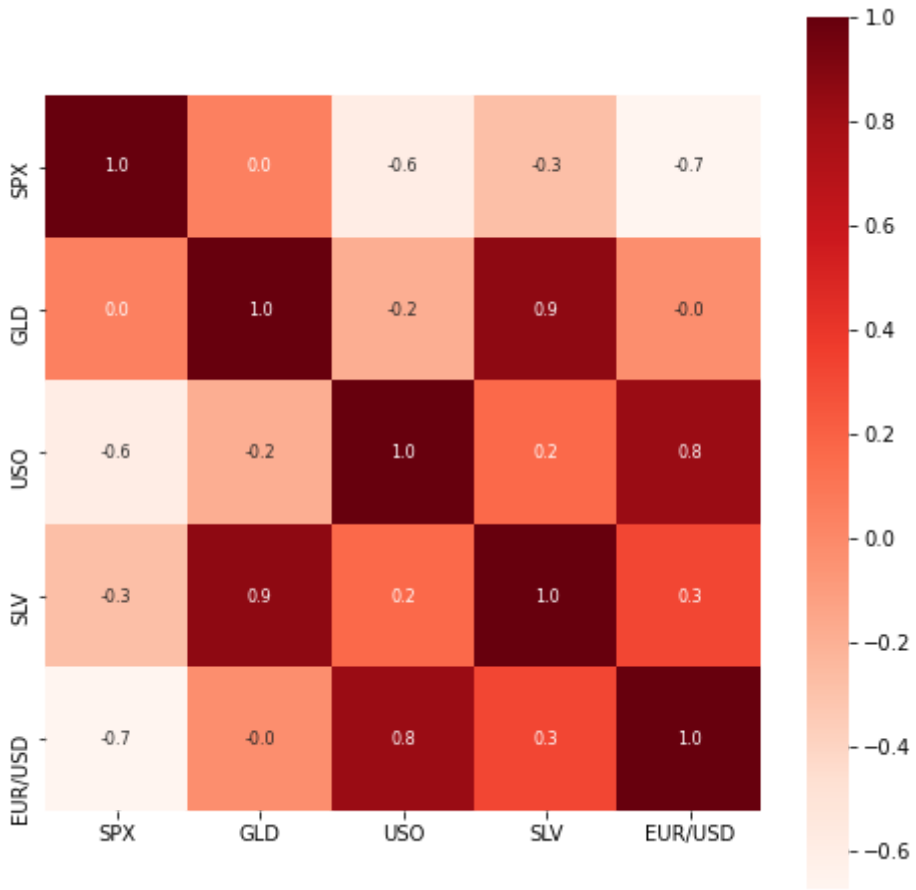
```
correlation = gold_data.corr()
```

In [10]:

```
# 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}
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
# 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
```

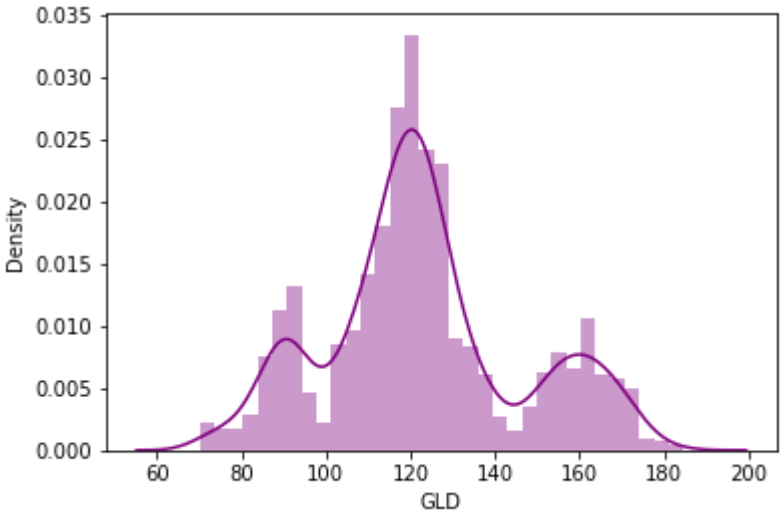
In [12]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

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



In [13]:

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

In [14]:

```
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 [15]:

```
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 [16]:

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

In [17]:

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

In [19]:

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

Out[19]:

```
RandomForestRegressor()
```

In [20]:

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

In [21]:

```
print(test_data_prediction)
```

```
[168.71029929  81.9443995  115.99720009 127.60040098 120.8770012
 154.66419797 150.51509864 126.28050014 117.51039874 125.94610104
 116.70170122 172.44470096 141.76259908 168.00979828 115.04429988
 117.78730064 138.98390306 170.17600078 159.63380286 156.96000001
 155.13490041 125.72960033 176.12389993 157.44430374 125.19450054
  93.96019976  77.67710001 120.42520002 119.15309944 167.48469925
  88.1043008  125.05129975  91.17530091 117.63980031 121.13029941
 135.73510059 115.60630105 114.80620069 149.07699977 107.20250098
 104.02420246  87.13589826 126.54370085 117.96360012 152.73189928
 119.67559985 108.56120002 107.98679785  93.12520079 127.23459755
  74.49590061 113.76969935 121.29830027 111.21959909 118.86319866
 120.79659953 159.61280007 168.68140107 146.88129666  85.87589885
  94.29870036  86.87299895  90.86599997 119.00160061 126.44130048
 127.53929988 169.25270105 122.41279946 117.34759913  98.54230049
 167.66420062 143.23869849 131.47010294 121.20610231 121.0377991
 119.65980062 114.64570155 118.4969004  106.93860107 127.825401
 114.06999995 107.48659937 116.68800064 119.50729877  88.72290098
  88.28249879 146.48210254 127.11989949 113.48160054 110.16339844
 108.007599    77.09869898 168.40630172 113.94579896 121.6070991
 127.99810177 155.05699892  91.81329945 134.78620057 158.96930309
 125.81730028 125.49500038 130.59070195 115.00530089 119.76109986
  92.13759975 110.66829897 167.98659998 156.78049914 114.05049926
 106.41000135  79.79089991 113.35020001 125.80460056 107.29479937
 119.48910085 155.80060345 159.84719919 119.87149968 134.8877032
 101.40039966 117.4744982  119.33860036 112.89320095 102.78529929
 160.36829789  98.90570033 147.05159916 125.64390101 169.85229882
 125.88809904 127.44729728 127.3868015  113.75089952 112.74470074
 123.87639893 102.29959891  89.17400018 124.97609925 101.62349946
 106.92939915 113.46680056 117.2512009   99.36889947 121.90570007
 163.43159902  87.24849864 106.84449933 117.45940069 127.71420107
 124.19540054  80.70009965 120.26740062 157.77749746  87.83909966
 110.30239927 118.89859933 173.1955987  103.02179896 105.73060042
 122.78070022 158.41709687  87.67019846  93.08340036 112.88420054
 177.25189929 114.20709939 119.32920027  94.58490076 125.56760016
 166.03240127 115.16510074 116.96110134  88.31759877 148.88210081
 120.27939933  89.48919941 112.36699988 117.35790077 118.70580135
  87.93189925  94.15360008 116.89200026 118.43820171 120.43150066
 127.04749871 121.93219994 152.62340014 165.24740154 118.53699941
 120.55760177 150.78060016 118.72649892 173.05469932 105.59069932
 105.01450161 149.84020078 113.9853007  124.88360113 147.30169987
 119.60890125 115.22390046 112.6913998  113.34070225 140.72630165
 117.85749781 103.03350027 115.83380104 103.63810186  99.02750054
 117.38400061  90.60699979  91.55330048 153.68089912 102.655
 154.95990122 114.26870158 138.0153018   90.15039822 115.50359933
 114.67659987 122.97600042 121.84380022 165.30720135  92.84129937
 135.76390155 121.40799904 120.79190063 104.67920017 140.83400319
 121.41999894 116.63850039 113.5112006  127.12439877 122.64919964
 125.85859923 121.30150003  86.95839921 132.52830194 144.41220212
  92.64659929 159.23130028 158.71620186 126.41349896 165.21469956
 108.97009998 110.0816006  103.5558982   94.25780048 127.72510278
 107.12480028 162.43210076 121.81709987 131.92960004 130.59560201
 160.75979975  90.10909834 174.95730183 127.77190056 126.93489931
  86.58289922 124.64650002 150.22209711  89.65849963 107.0711995
 108.93769955  84.51739884 136.03030007 155.24940232 137.24900345
  74.42580028 153.23590086 126.1340998  126.80850043 127.57539927
 108.68379939 156.49770027 114.48520102 117.07080161 125.10479954
```

```
 154.0709012  121.34170027 156.37289875  92.88000069 125.5985014
 125.59149994  87.69940025  92.0675992  126.23129928 128.33390335
 113.17800018 118.07099763 120.83500022 127.23289871 119.79320129
 135.72020002  93.88879929 119.8342004  113.31200086  94.33759949
 108.73139999  86.89899878 109.81999911  89.61340026  92.43169997
 131.80030317 162.19480019  89.11870034 119.44970102 133.4665015
 123.91830017 128.49520246 102.0560984   88.88879886 131.50610037
 119.95750038 108.76779996 168.91160129 115.23980054  86.68669924
 118.79920075  91.09299955 161.80600053 116.65670049 121.67540026
 160.29549756 120.19999932 112.80209949 108.41849871 126.80789991
  76.08210008 103.04109992 127.39310238 121.68859933  92.49789979
 132.28200016 118.16360109 115.62819956 154.8344024  159.28480055
 109.83910009 152.73389796 119.32450092 160.40950078 118.52640053
 158.20489969 115.06499925 116.4578001  148.58739913 114.72590054
 125.93539872 166.5831     117.64060002 125.28359928 153.49360336
 153.59700237 132.40870084 114.83320033 121.2230019  125.4438008
  89.72530077 123.08449996 154.91530162 111.77690035 106.87609968
 161.75570104 118.48919992 165.74410015 134.00530136 114.53869973
 152.95889891 168.51219922 115.28830042 114.21170134 158.24419865
  85.14369873 127.20850062 127.90770078 129.05030023 124.18560041
 123.8812005   90.59680073 153.27889993  97.03369984 137.85199954
  88.97769909 107.51719958 114.95010048 112.53480094 124.17639908
  91.31619902 125.35880113 162.34229951 120.07269892 165.15450087
 127.16319806 112.42300014 127.54439956  94.94439899  91.34759991
 103.62859906 120.79700013  83.06259938 126.32210005 160.08520448
 117.37590089 118.3359001  119.76629987 122.84159976 120.02080145
 121.41400004 118.08670041 107.02249962 148.19630014 126.1209989
 115.67140073  73.8650997  127.84550089 153.73450115 122.58520021
 125.60450069  88.70030027 103.13459889 124.93000039 120.29360001
  73.53440086 151.58190018 121.16760049 104.81840025  86.21229791
 115.16299904 172.28749851 119.70130041 159.61679794 113.20249935
 121.26080009 118.53480105  95.97859987 118.53099985 126.05970039
 118.44169965  95.9861003  153.9624015  122.00170036 147.13750052
 158.85230247 113.76850012 122.53759942 149.69089731 127.44390011
 165.91209978 135.86970021 120.0392989  166.84489787 108.46409926
 121.83739839 140.04290093 107.11909875]
```

In [22]:

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

```
R squared error :  0.9887188692408645
```

In [23]:

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

In [24]:

```
plt.plot(Y_test, color='blue', label = 'Actual Value')
plt.plot(test_data_prediction, color='pink', label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```