# GOLD PRICE PREDICTION

## Importing Dependencies

In [43]:
```
import numpy as np
import pandas as pd


import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

In [42]:
```
import matplotlib.pyplot as plt
```

## Data collection and processing

In [45]:
```
gold_data = pd.read_csv(r'C:\Users\HP\Downloads\archive (1)\gld_price_data.csv')
```

In [46]:
```
gold_data.head()
```

Out[46]:

| | 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 [47]:
```
gold_data.tail()
```

Out[47]:

| | 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 |

## Number of rows and columns in the dataframe

In [48]:
```
gold_data.shape
```

Out[48]:
```
(2290, 6)
```

## Checking null values (if any)

In [49]:
```
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 [50]:
```
gold_data.isnull().sum()
```

Out[50]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

It can be observed that the given in dataset there is no null values.

# Statistical insights of the dataframe

In [51]:
```
gold_data.describe()
```

Out[51]:

| | 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 |

# Correlation :

```
1. Positive correlation
2. Negative correlation
```

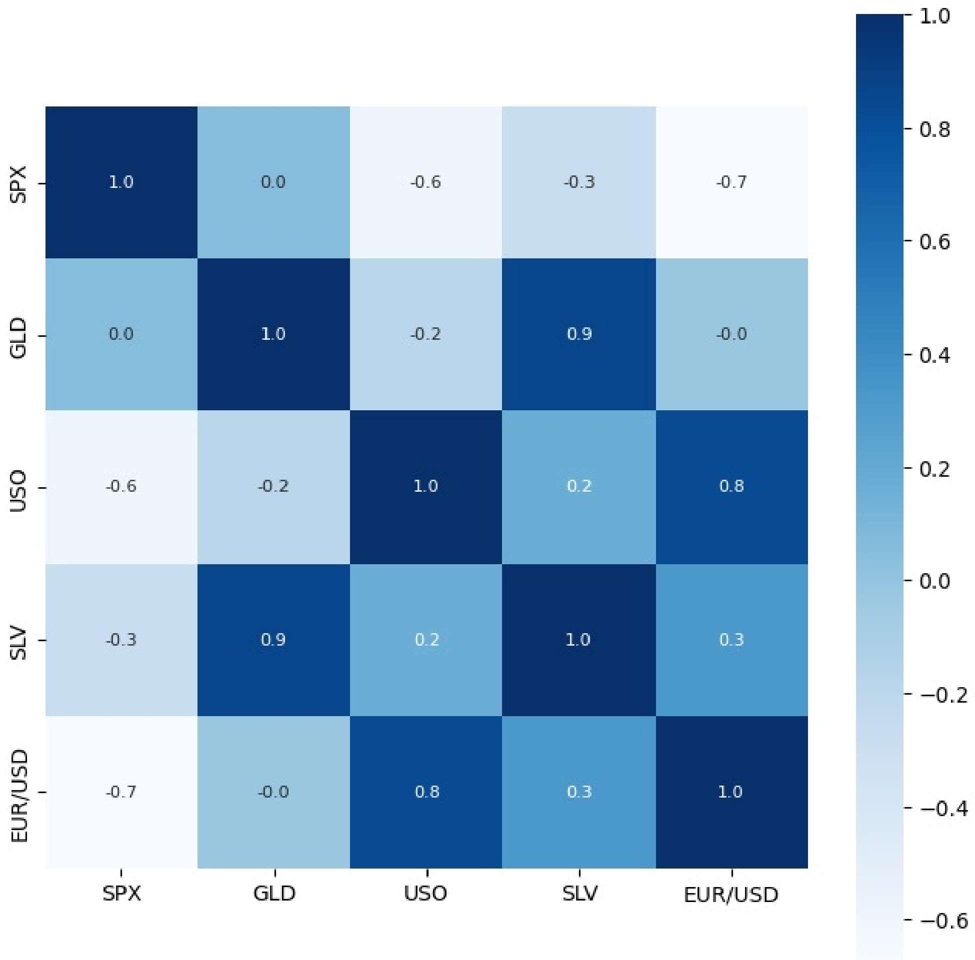
In [52]:
```
correlation = gold_data.corr()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17232\1828644926.py:1: FutureWarning: The default value of numeric_onl
y in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  correlation = gold_data.corr()
```

# Heatmap for understanding correlation

In [53]:
```
plt.figure(figsize=(8,8))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Blues')
```

Out[53]:
```
<Axes: >
```




## Correlation values of gold

In [54]:
```
print(correlation['GLD'])
```

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

## Checking the distribution of Gold price

# Checking the distribution of Gold price

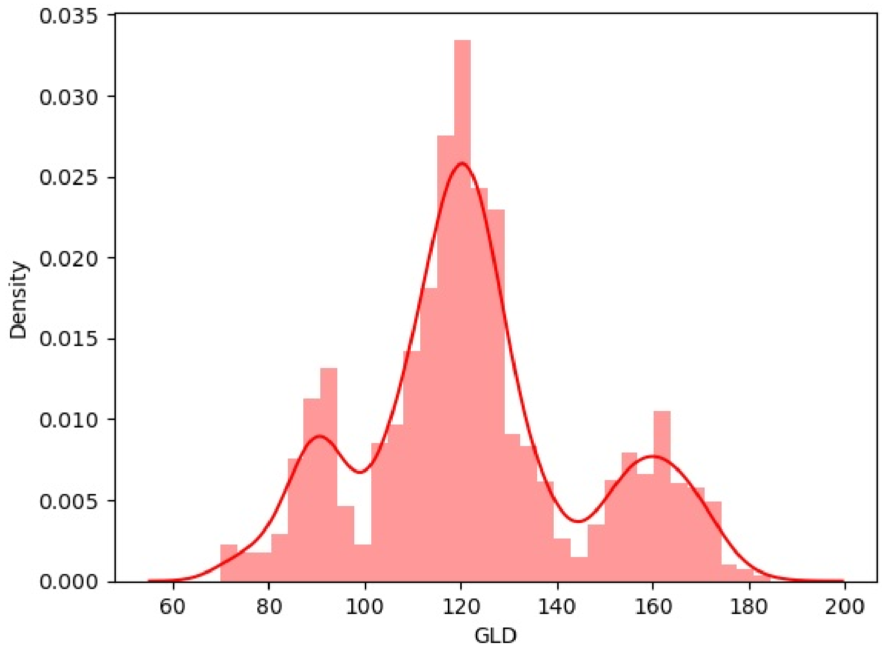
In [56]:

```
sns.distplot(gold_data['GLD'],color='Red')
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_17232\4230193789.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(gold_data['GLD'],color='Red')
```

Out[56]:

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




# Splitting the Features and Target

In [58]:

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

In [59]:

```
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 [60]:

```
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
```

# Splitting into Training Data and Test Data

In [61]:
```
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 [63]:
```
regressor = RandomForestRegressor(n_estimators=100)
```

In [64]:
```
regressor.fit(X_train,Y_train)
```

Out[64]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

# Model Evaluation

# Prediction on test data

In [65]:
```
test_data_prediction = regressor.predict(X_test)
```

In [66]:
```
print(test_data_prediction)
```

```
[168.51839882  81.99409991 116.06170004 127.52480036 120.81720113
 154.85419751 150.29649868 126.22640024 117.54439901 125.93430087
 116.81520105 171.71100068 141.2726981  167.7287982  115.28499996
 117.58570061 137.08320328 170.24530173 159.13090251 159.42879889
 155.07370031 125.37390051 176.45139946 157.61330415 125.14460037
  93.84869956  76.60970065 120.52130006 119.14329972 167.46979991
  88.20360063 125.11070023  91.15180076 117.73090035 121.11079909
 136.18900041 115.41520166 114.9108008  146.9875995  107.20650128
 103.67540222  87.2882979  126.46980061 118.09260047 151.84949843
 119.47610036 108.41929969 108.18759849  93.23350066 127.08449816
  75.51820007 113.55139922 121.22789979 111.28109921 118.97479896
 120.80989927 159.00000026 169.97130144 146.83599684  85.87679896
  94.28710026  86.85209908  90.56300027 119.02720074 126.40320051
 127.55750003 168.71140095 122.3481995  117.42949921  98.72470032
 167.62810081 143.15489841 132.10550267 121.20670265 120.92129933
 119.74430077 114.72700168 118.52920031 107.14060122 127.90960025
 113.87429965 107.30379969 116.81810039 119.5911988   89.00620046
  88.19539863 146.77610244 127.31299991 113.33159989 109.90729847
 108.28939907  77.04259916 169.25430217 114.17749933 121.82799879
 128.03930197 155.14489792  91.52049921 136.82430133 159.12420316
 125.92690061 125.26870071 130.84510167 114.98640137 119.86600012
  92.1363999  110.29929888 168.00420002 156.09859909 114.30019951
 106.61220113  79.37759994 113.23210053 125.81920096 107.17799919
 119.51540107 155.75870317 159.41629827 120.12889969 134.80040311
 101.64129976 117.35669819 119.25660033 113.06720068 102.69359922
 159.96679795  99.2158007  146.92429906 125.77250108 169.28649904
 125.74039887 127.33859763 127.50740187 113.92569955 112.73360062
 123.72549906 102.16899907  89.01810002 124.32639982 101.88009959
 107.24179919 113.62960052 117.33480065  99.6954995  121.86590042
 163.01759936  87.34899881 106.69289984 117.24590062 127.71260092
 124.07270076  80.71669885 120.18410068 157.67339796  87.83049987
 110.40599931 118.85149916 172.08139856 103.00299899 105.85100041
 122.45210024 158.08789757  87.42049821  93.39960027 112.6892005
 177.47649968 114.25219981 119.24200004  94.84360093 125.9182005
 166.11560087 114.86820076 116.86930146  88.35359884 149.62920163
 120.31089939  89.38270001 111.4067998  117.11050088 118.8129012
  88.14389973  94.46269997 117.27369992 118.43340192 120.38390078
 126.75959821 121.91730014 148.7444     164.94530104 118.69049971
 120.25770095 152.42300065 118.60849917 172.63229897 105.72119947
 104.94360162 149.67910184 113.70160054 124.80560093 147.60009936
 119.58860112 115.24690038 112.50349972 113.37800222 141.19610041
 117.87369772 102.92970053 115.87210099 103.33900173  99.12290042
 117.1114007   90.65740011  91.4814001  153.49119939 102.75379984
 154.39310066 114.37050171 138.82650039  90.39469815 115.51659953
 114.82059985 123.05380054 121.90130018 165.4036018   92.94839933
 135.40760111 121.31779929 120.77360093 104.53680019 140.98560305
 121.96949896 116.62860053 113.45570092 126.93459787 122.53959939
 125.69679936 121.16350074  86.87349925 132.2941008  145.26980196
  92.77659963 157.6608988  158.05520213 126.08639896 164.51819932
 108.73079987 110.21670086 103.68709852  94.27870052 128.08930338
 107.19800069 162.18799967 121.83620043 132.05750025 130.8967014
 160.64799951  90.12679844 175.65010112 128.20730093 126.75929862
  86.43469916 124.78930013 150.21389748  89.67240048 106.78279979
 109.01899997  84.19529887 135.84870001 155.22000289 138.59250351
  73.80920031 152.46830122 126.44279986 126.70130022 127.60029848
 108.64219921 156.44539913 114.6413012  116.95020146 125.61499947
 154.06410114 121.09080023 156.46329851  92.96410064 125.51260136
 125.4512003   87.90200045  92.08409924 126.36859911 128.67770375
 113.1476006  117.71279745 121.01320009 127.0646981  119.52540113
 136.10480012  93.89239903 119.79220041 113.40970107  94.41709937
```

```
 108.9283998   87.86599918 109.28549926  89.59249994  92.3203004
 131.91070344 162.31100031  89.53679982 119.69330078 133.26390192
 123.8549999  128.58050239 101.96459854  89.17199865 131.95850103
 120.04930022 108.56220005 169.02790089 115.2228007   86.62369864
 118.89790071  90.87659962 161.75240042 116.50530037 121.15350003
 160.37769777 119.85679908 112.62239954 108.49919886 126.77070056
  76.16440039 103.02229996 127.82220318 121.71309939  92.65510005
 132.07250049 118.02240096 115.96279983 154.38010246 159.62460084
 109.9914996  152.35779746 119.21180066 160.26480014 118.42040008
 157.2907003  115.13879927 116.87300015 148.65479964 114.88650061
 125.58779873 166.5907997  117.82110007 124.78339931 153.04870349
 153.53140272 132.01920016 114.67720038 121.2232024  125.07060055
  89.81860048 122.63920012 155.45220184 111.78560041 106.74900013
 161.84680136 118.40870002 165.74649976 133.98220141 115.0074999
 152.95949887 168.68499954 115.43639995 114.18460121 159.34809864
  85.29399894 127.05040015 127.88470077 128.72359975 124.25820083
 123.88710094  90.8321009  153.2217999   97.13299961 136.25149951
  89.02159922 107.30559977 115.05730035 112.55760121 124.1280992
  91.34049864 125.43840127 162.40639845 119.84839913 165.02860088
 126.79629808 112.57619994 127.54479934  94.94449919  91.03140013
 103.79349913 120.73569978  82.67629946 126.32839986 159.98570444
 117.20340096 118.37219969 119.83119977 122.77779995 120.07200138
 121.42290017 118.2989     107.26239978 148.3882003  126.16999874
 115.59750073  74.18370013 127.88030115 152.4793003  122.90910008
 125.61410047  89.0047004  104.06579837 124.73580043 120.17260068
  73.4263008  151.95550004 121.18140007 104.64750012  86.50979779
 115.23999924 172.25089783 119.57240027 159.19619768 113.17629935
 120.63620003 118.65450093  96.03759976 118.65380051 125.83160021
 118.56579963  96.05850057 153.99420189 121.92729977 147.54829949
 159.63370199 113.5526003  122.62729928 148.79059763 127.23860046
 165.8403006  135.23050018 120.21569983 166.83969872 108.57339957
 121.91829843 139.28720053 106.66049884]
```

# R squared error

In [67]:
```
error_score=metrics.r2_score(Y_test,test_data_prediction)
```

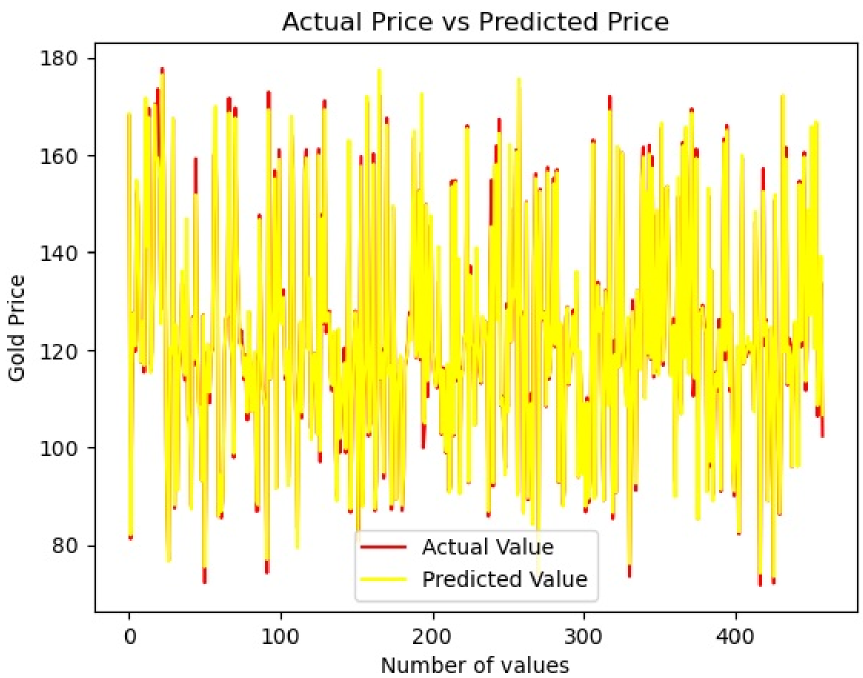
In [69]:
```
print('R squared error :',error_score)
```

```
R squared error : 0.9897759709088313
```

# Compare the Actual Values and Predicted Values in a plot

In [70]:
```
Y_test=list(Y_test)
```

In [73]:
```
plt.plot(Y_test,color='red',label='Actual Value')
plt.plot(test_data_prediction,color='yellow',label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('Gold Price')
plt.legend()
plt.show()
```




In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js