```
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
```

Data collection and processing

```
# loading the csv data into panda data frame
gold_data = pd.read_csv('/content/gld_price_data.csv')
```

```
#printing first five rows
gold_data.head()
```

| | 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 |

```
#printing last five rows
```

```
gold_data.tail()
```

| | 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 |

```
gold_data.shape
```

```
(2290, 6)
```

```
#basic information of 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
```

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

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

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

| | 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 |

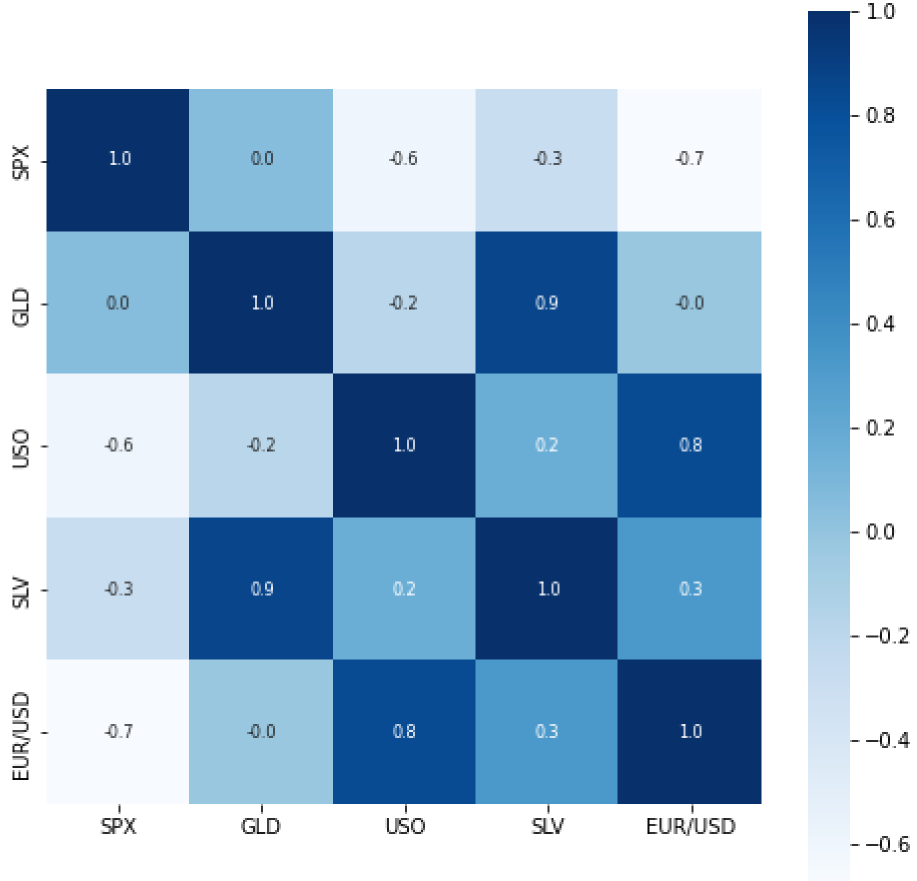
correlation:

1. Postive correlation
2. negative correlation

```
correlation = gold_data.corr()

#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='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3207cffd90>
```



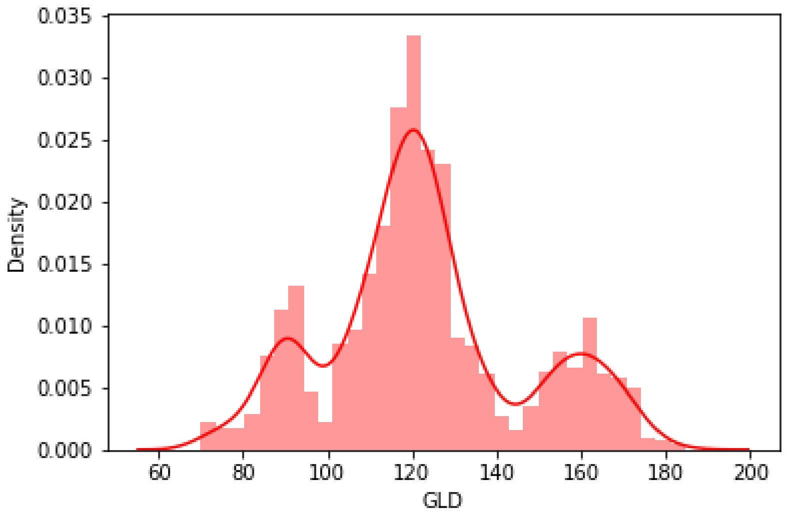

```
#correlation of GLD
print(correlation['GLD'])
```

```
SPX        0.049345
GLD        1.000000
```

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

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3204f3ea50>
```




splitting of features and target

1. features=other stack price
2. target=gold GLD

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

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]
```

```
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

```
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

```
regressor=RandomForestRegressor(n_estimators=100)
#training the model
regressor.fit(X_train,Y_train)
```

```
RandomForestRegressor()
```

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

```
print(test_data_prediction)
```

```
[168.62269904  81.9125999  115.96660022 127.66960065 120.5893014
 154.78599807 150.32499858 126.13990018 117.41229881 126.08970088
 116.71580116 171.4446008  142.0542983  167.88719843 115.03360002
 117.5215004  139.50500276 170.1713006  159.50110283 160.07119984
 155.17510066 125.05270049 174.81569956 156.87760243 125.26060062
  94.13259961  77.03210022 120.79439997 119.06269932 167.58390048
  88.07440077 125.03920006  91.20370091 117.60830045 121.13539929
 136.23020152 115.78950082 115.16670102 146.27469911 107.11280112
 103.8922024   87.02099787 126.62860048 117.95840022 152.98899856
 119.52870001 108.47869972 108.18379857  93.3222007  127.29279742
  75.51540028 113.79419909 121.23880001 111.33089942 118.73539865
 120.85519922 159.53580042 167.60720154 147.12849731  85.90489855
  94.36790037  86.9103989   90.66200018 118.85560066 126.45250051
 127.48710008 170.45850001 122.37769904 117.6218986   98.57869998
```

```
 168.81130162 143.2082993  131.64210248 121.12720194 121.59849925
 119.60340032 114.30150142 118.36020039 106.84240099 127.87980155
 113.91719941 107.48369999 117.01240053 119.6956985   88.63970023
  88.33809863 146.68360228 127.19379996 113.65190085 110.38619828
 108.24889894  77.45789933 169.95010249 113.93329918 121.56659897
 127.82840182 154.98789798  91.78659954 135.74460159 159.49730322
 125.70250082 125.22880048 130.52400197 114.82410123 119.81530006
  92.08549986 109.97079884 165.54819942 156.61330022 114.14079939
 106.40570138  79.32900026 113.25280024 125.80900049 106.8803994
 119.48300109 155.49940356 159.44429854 120.07059989 135.65000363
 101.13289997 117.38289788 119.2142004  112.98540093 102.81899903
 160.4090979   98.71190035 147.32349871 125.41460099 169.62529953
 126.01189825 127.48249695 127.43380165 113.63079924 112.93670077
 123.60569908 102.17419889  89.01129979 124.53779953 102.15869935
 106.90549885 113.85410098 117.361101    99.40999943 121.68030075
 163.471399    87.18879868 106.77910019 117.14280089 127.74200177
 124.05450063  80.89869925 120.55370037 157.69189801  87.69819957
 110.16759986 119.00259892 172.75529925 103.06339907 105.36570069
 122.51530013 157.91189751  87.86219828  93.0316003  113.26960018
 176.88269945 114.65929953 119.27470063  94.93950108 125.88480047
 166.19540113 114.76670108 116.62050107  88.4029988  148.69560071
 120.5268993   89.46119992 111.89319975 117.28320023 118.83580113
  88.17719967  94.08989999 117.22879977 118.46150187 120.58670076
 126.75959794 121.82430003 148.62059966 165.32930154 118.43109948
 120.22100123 149.60340022 118.63929902 172.46269915 105.79579926
 104.91950136 148.7360008  113.81950094 124.81720081 147.60589951
 119.60880114 115.31970037 112.49979989 113.42650205 142.70930171
 118.05939749 102.9163005  115.86840113 103.56380186  98.97490027
 117.32200101  90.65470011  91.58440041 153.21389871 102.7527
 154.56920082 114.45230157 138.52050145  90.17579807 115.47979946
 114.86769947 123.12860033 121.70000012 165.31810148  92.92749955
 135.14770112 121.31149956 120.67540056 104.68440026 142.39120275
 121.41269937 116.66950049 113.4656006  126.91839755 122.71259969
 125.71199951 121.18900076  86.87429917 132.76250166 145.29600176
  92.63969986 157.96359975 159.03020209 126.36359879 164.89619948
 108.91369956 109.5493009  103.71829838  94.3128008  127.77310282
 107.30080034 161.52269997 121.62750066 132.22590035 130.55860158
 160.77040009  90.18089835 174.21280189 127.55080049 126.85729814
  86.44959929 124.70679925 150.20099753  89.57900022 107.04159992
 108.99379979  84.17439931 137.04200035 155.02450285 139.95590325
  73.60750025 151.16190107 126.08869974 126.73799993 127.53699942
```

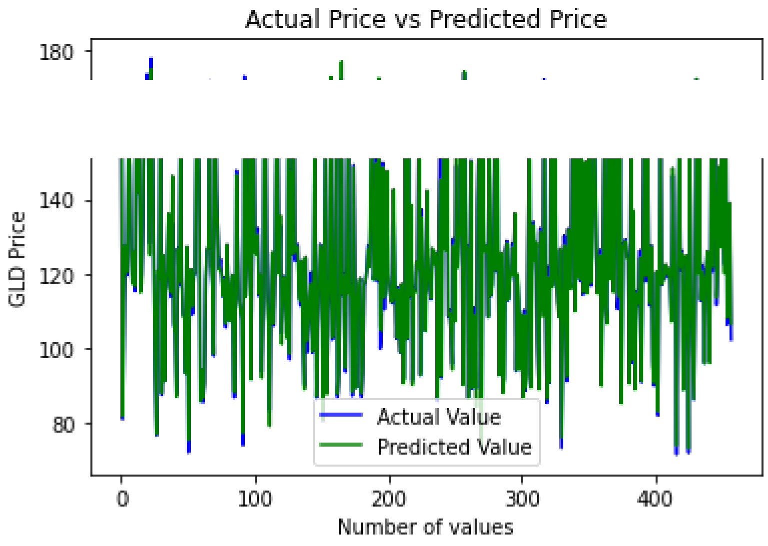
```
108.54929917 156.35749993 114.45900109 116.89720145 125.44229939
154.0067009  121.12620012 156.39549954  92.85940041 125.53540138
```

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

```
R squared error :  0.9896066050036644
```

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

```
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()
```


Actual Price vs Predicted Price
180
140
120
100
80
GLD Price
Actual Value
Predicted Value
0
100
200
300
400
Number of values