Importing the Libraries

```
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 to a Pandas DataFrame
gold_data = pd.read_csv('/content/gld_price_data.csv')
gold_data
```

	Date	SPX	GLD	USO	SLV	EUR/USD
0	1/2/2008	1447.160034	84.860001	78.470001	15.1800	1.471692
1	1/3/2008	1447.160034	85.570000	78.370003	15.2850	1.474491
2	1/4/2008	1411.630005	85.129997	77.309998	15.1670	1.475492
3	1/7/2008	1416.180054	84.769997	75.500000	15.0530	1.468299
4	1/8/2008	1390.189941	86.779999	76.059998	15.5900	1.557099
...	...	...	...	...	...	...
2285	5/8/2018	2671.919922	124.589996	14.060000	15.5100	1.186789
2286	5/9/2018	2697.790039	124.330002	14.370000	15.5300	1.184722
2287	5/10/2018	2723.070068	125.180000	14.410000	15.7400	1.191753
2288	5/14/2018	2730.129883	124.489998	14.380000	15.5600	1.193118
2289	5/16/2018	2725.780029	122.543800	14.405800	15.4542	1.182033

2290 rows × 6 columns

```
# print first 5 rows of the dataframe
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

```
# 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
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
gold_data.shape
```

```
# 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
```

```
# checking the no. of missing values
gold_data.isnull().sum()
```

```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
```

```
EUR/USD    0
dtype: int64
```

```
# getting the statistical measure 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

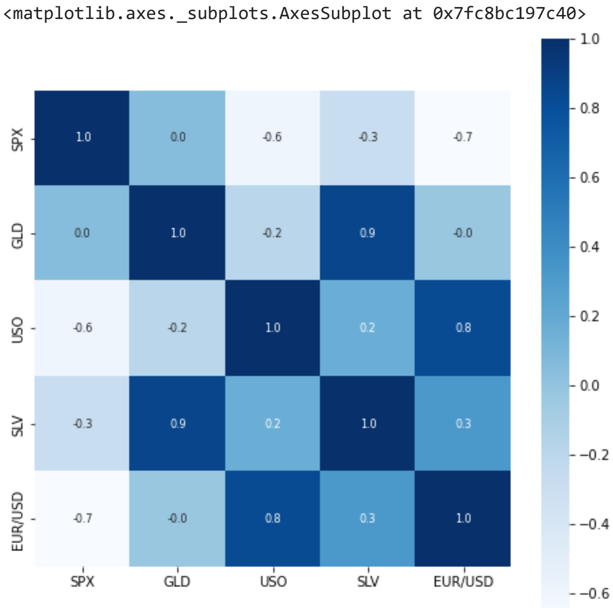
Correlation

1. Positive Correlation
2. Negative Correlation

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

```
# Constructing a heat map 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 0x7fc8bc197c40>
```

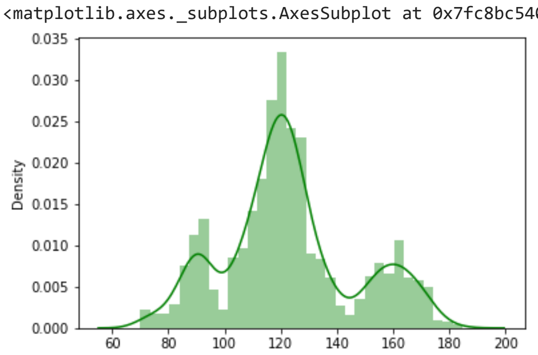



```
# correlation values at gld
print(correlation['GLD'])
```

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

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be re
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc8bc540880>
```



Splitting the Features and Target

```
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()
```

Model Evaluation

```
# Prediction on test data
test_data_prediction = regressor.predict(X_test)
```

```
print(test_data_prediction)
```

```
[168.78900006  82.02609982 116.32910014 127.63060057 120.55880098
 154.46919753 150.20919856 125.94890064 117.4766988  126.23949988
 116.66930089 171.55470045 141.94239827 167.88669858 115.1351999
 117.82390042 141.54160309 170.23150138 159.26600288 159.64409892
 155.16370031 125.54609991 174.98350002 157.67230271 125.17790034
  93.92660003  77.75700038 120.53750015 119.10059946 167.58379962
  88.3324006  125.36020034  91.33640107 117.57590021 121.12449913
 136.19100119 115.51070095 115.30770084 148.7129997  107.03750092
 104.53850247  87.25529821 126.56440055 118.17729987 154.06699891
 119.50499989 108.40119994 108.13749856  93.09420042 127.15249746
  74.21380064 113.61479923 121.14630027 111.3331991  118.91239891
 121.3221991  159.45789959 167.86400115 147.24089674  85.87409898
  94.44280032  86.74389909  90.49340019 119.07590065 126.42980069
 127.55240007 171.04270027 122.29629945 117.45869884  98.45810041
 167.92660147 142.77769768 132.02510281 121.27330224 121.36869943
 119.57050072 114.39160166 118.11130069 106.94400114 127.98880026
 114.04159957 107.15289991 117.03120036 119.77159868  88.86240055
  88.14559857 146.77060238 126.87839984 113.81380028 110.75449825
 108.19149903  76.99729904 169.75630228 114.04599898 121.66849939
 127.94310192 154.80529819  91.73019952 136.16460146 159.50220302
 125.50500067 125.23120074 130.52500202 114.62550116 119.77870004
  92.05730008 110.11959884 169.30589976 156.80299914 114.31299981
 106.66530144  79.59419944 113.3009001  125.80240053 106.94709962
 119.20760095 155.78390364 159.5443995  120.56449988 136.12400372
 101.61309987 117.36899803 119.35640005 113.03610092 102.7689993
 160.27009786  99.13760036 147.73659993 125.37910087 170.14589938
 125.79309855 127.39829703 127.23520114 113.88299899 112.80840068
 123.55209903 102.08909903  89.30939972 124.97189916 101.56209922
 107.18899922 113.50080068 117.17130077  99.07469945 121.71130057
 162.74729891  87.42589883 106.65579971 117.41020079 127.71220122
 124.212601    80.81859902 120.2890008  158.7031975   87.9793999
 110.48289917 119.07069918 171.66699833 102.96289933 105.81100057
 122.93130026 158.74639722  87.76529841  93.1554005  112.60320026
 177.1432001  114.55089962 119.36030012  94.77710125 125.78660016
 165.88890118 114.86510069 117.09300149  88.26489852 148.64440045
 120.35549972  89.4272999  111.73879979 117.25840012 118.71490131
  88.20119939  94.21330014 117.03409994 118.670602   120.26620085
 126.90909821 121.94359971 149.74260037 165.65700087 118.56599964
 120.17650149 149.81800028 118.50839912 172.21759874 105.24409911
 104.95870109 148.67480054 113.851401   124.8387007  147.32649982
 119.57390122 115.26790023 112.52800005 113.45560189 141.40530129
 117.77009772 102.91210068 115.85010124 103.91190165  98.80060036
 117.24770083  90.63420016  91.43840088 153.626899   102.60020025
```

```
154.91000099 114.39360164 139.19640155  89.96789824 115.45889956
114.08289968 122.91750031 121.70610062 165.29730159  92.85729935
135.18940104 121.39599938 120.67720057 104.71950004 142.27720299
121.683199   116.6146005  113.56150045 127.04709767 122.45089944
125.83689947 121.28170028  86.87679909 132.69070194 145.30230211
 92.83089915 156.89829932 158.90580267 126.44629878 165.25839966
108.90560006 109.41850088 103.62199807  94.33480086 127.92040273
106.86560088 160.15799918 121.68590039 131.98520017 130.60940233
160.81029923  90.0154984  174.95790151 127.59680031 126.92579848
 86.38969951 124.40029946 149.97779732  89.69710031 106.84749962
109.02669976  84.72729902 136.4390002  154.87190152 139.35360342
 74.66500018 151.50210131 126.68019993 126.72430002 127.43449874
108.84869983 156.10229916 114.49990101 116.84980155 124.97699937
153.97630182 121.27819991 156.4140984   92.96210044 125.50750138
```

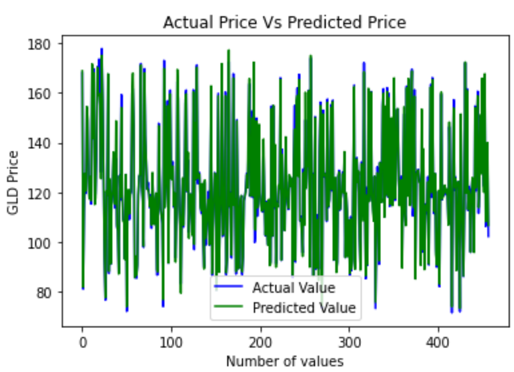
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print('R squared error is:-', error_score)
```

```
R squared error is:- 0.9894027113422046
```

Compare the Actual Values and Predicted Values in a Plot

```
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()
```



✓ 0s completed at 11:18 AM