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
```

```
(2290, 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
```

```
# 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()
```

```
<ipython-input-10-b9d572e5c3ef>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versior
  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')
```

```
<Axes: >
```


```
# 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')
```

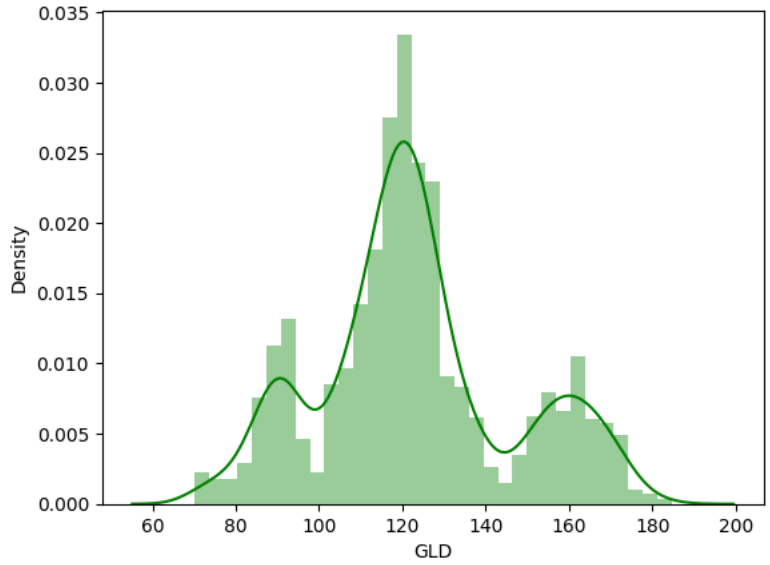
```
<ipython-input-13-6e6442c8d5d9>:2: UserWarning:

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



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
RandomForestRegressor()
```

Model Evaluation

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

```
print(test_data_prediction)
```

```
 108.83139953  84.20239902 136.17639981 155.05850282 139.29410373
  74.66550006 152.38750087 125.8759996  126.69819997 127.60389868
 108.56859958 156.03310006 114.49040094 116.84850127 125.32869934
 154.03300165 121.23109995 156.35209919  92.97170058 125.42810098
 125.71760052  88.06000078  92.34459909 126.24279992 128.49400368
 113.06810024 117.78029753 120.99020032 127.09039833 119.59210111
 136.28720094  93.84899942 119.82420056 113.1318011   94.22729929
 108.86819983  86.66179901 109.20319939  89.83349966  92.5119001
 131.46890242 162.34460077  89.27080028 119.81980087 133.43420185
 123.85079976 128.3547015  102.00779862  88.81759911 131.91890096
 119.69100041 108.34340011 168.60740146 115.39080029  86.71179905
 118.75670072  91.1229994  162.02000047 116.31490018 121.59690039
 160.3200979  120.13379924 112.70349926 108.36219863 126.77040047
  76.10630016 102.93989991 127.69820279 122.00789895  92.58670002
 132.14200078 118.09380067 116.13239969 154.23680306 159.66250111
 109.92579996 154.96099798 119.22570083 160.75630084 118.44550066
 158.83329948 115.12239929 116.6790007  149.12459947 114.70690084
 125.35599881 165.7843997  117.71740031 124.76839945 153.55000294
 153.57150234 132.23079995 114.77260036 121.22730197 124.58980069
  89.60060089 123.20300036 154.46260144 111.81250039 106.75779971
 161.52460121 118.51479965 165.69239987 134.08640111 114.93139968
 153.11669918 168.47850007 114.7213998  114.11440111 157.93839908
  85.44359857 127.14250052 127.80510041 128.9845998  124.33670106
 123.89330065  90.36760015 153.23059997  97.18039959 138.14599978
  88.9977991  107.06309994 115.15300061 112.85270099 123.85309981
  91.35789873 125.35320087 162.50779905 120.04369863 164.93900097
 126.77789786 112.3226002  127.40249885  94.82309926  90.92550011
 103.26259909 120.85310005  83.10419943 126.31409991 160.70640465
 117.18640087 118.27870026 119.7477999  122.84819934 120.02670125
 121.61620001 118.34850059 107.0499003  148.21960001 126.36429854
 115.6999008   73.77869996 127.8214013  153.94330085 121.94169993
 125.5845007   88.89320006 103.98429887 125.45050078 120.27600046
  73.24820091 151.69899964 121.0396004  104.61470025  86.43519764
 115.04529873 172.25859822 120.09310027 160.75739726 113.1919994
 121.2691003  118.81660091  95.98529992 118.74620019 125.80770022
 118.55319978  95.72820052 153.73280187 122.24589995 147.6278004
 159.4087021  113.49990026 122.50069935 148.64159858 127.27570025
 165.66700061 136.41890123 119.9745999  167.30199824 108.31269965
 121.59909882 139.54840107 106.89909912]
```

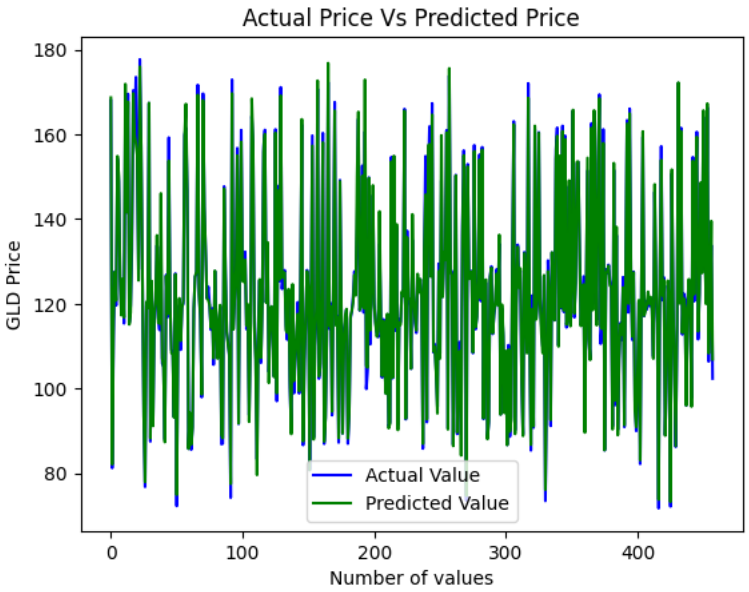
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print('R squared error is:-', error_score)
```

```
R squared error is:- 0.9895431823334062
```

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()
```



Colab paid products - Cancel contracts here

✓ 1s completed at 9:52 AM