# Gold Price Predection

**Batch Name: MIP-ML-07**

## Author: Irfan Ullah Khan


GITHUB PROFILE | KAGGLE PROFILE | LINKEDIN PROFILE

YOUTUBE PROFILE | EMAIL CONTACT ME | WEBSITE CONTACT ME


## Installing Dependencies

---

```
#!pip install numpy
```

```
#!pip install pandas
```

```
#!pip install seaborn
```

```
#!pip install sklearn
```

```
#!pip install matplotlib
```

## Importing the Libraries

```
# will hide errors like outdated verisions
import warnings
warnings.filterwarnings('ignore')
```

```
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
df = pd.read_csv('gold_price_data.csv')
```

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

| | SPX | GLD | USO | SLV | EUR/USD |
|---|---|---|---|---|---|
| **0** | 1447.160034 | 84.860001 | 78.470001 | 15.180 | 1.471692 |

| | | | | | |
|---|---|---|---|---|---|
| **1** | 1447.160034 | 85.570000 | 78.370003 | 15.285 | 1.474491 |
| **2** | 1411.630005 | 85.129997 | 77.309998 | 15.167 | 1.475492 |
| **3** | 1416.180054 | 84.769997 | 75.500000 | 15.053 | 1.468299 |
| **4** | 1390.189941 | 86.779999 | 76.059998 | 15.590 | 1.557099 |

Next steps: Generate code with `df` View recommended plots

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

| | SPX | GLD | USO | SLV | EUR/USD |
|---|---|---|---|---|---|
| **2285** | 2671.919922 | 124.589996 | 14.0600 | 15.5100 | 1.186789 |
| **2286** | 2697.790039 | 124.330002 | 14.3700 | 15.5300 | 1.184722 |
| **2287** | 2723.070068 | 125.180000 | 14.4100 | 15.7400 | 1.191753 |
| **2288** | 2730.129883 | 124.489998 | 14.3800 | 15.5600 | 1.193118 |
| **2289** | 2725.780029 | 122.543800 | 14.4058 | 15.4542 | 1.182033 |

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

```
(2290, 5)
```

```
# getting some basic informations about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   SPX      2290 non-null   float64
 1   GLD      2290 non-null   float64
 2   USO      2290 non-null   float64
 3   SLV      2290 non-null   float64
 4   EUR/USD  2290 non-null   float64
dtypes: float64(5)
memory usage: 89.6 KB
```

```
# checking the number of missing values in Gold Data
df.isnull().sum()
```

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

```
df.duplicated().sum() # checking the duplicate values in Gold Data
```

```
0
```

```
# getting the statistical measures of the Gold data
df.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 |

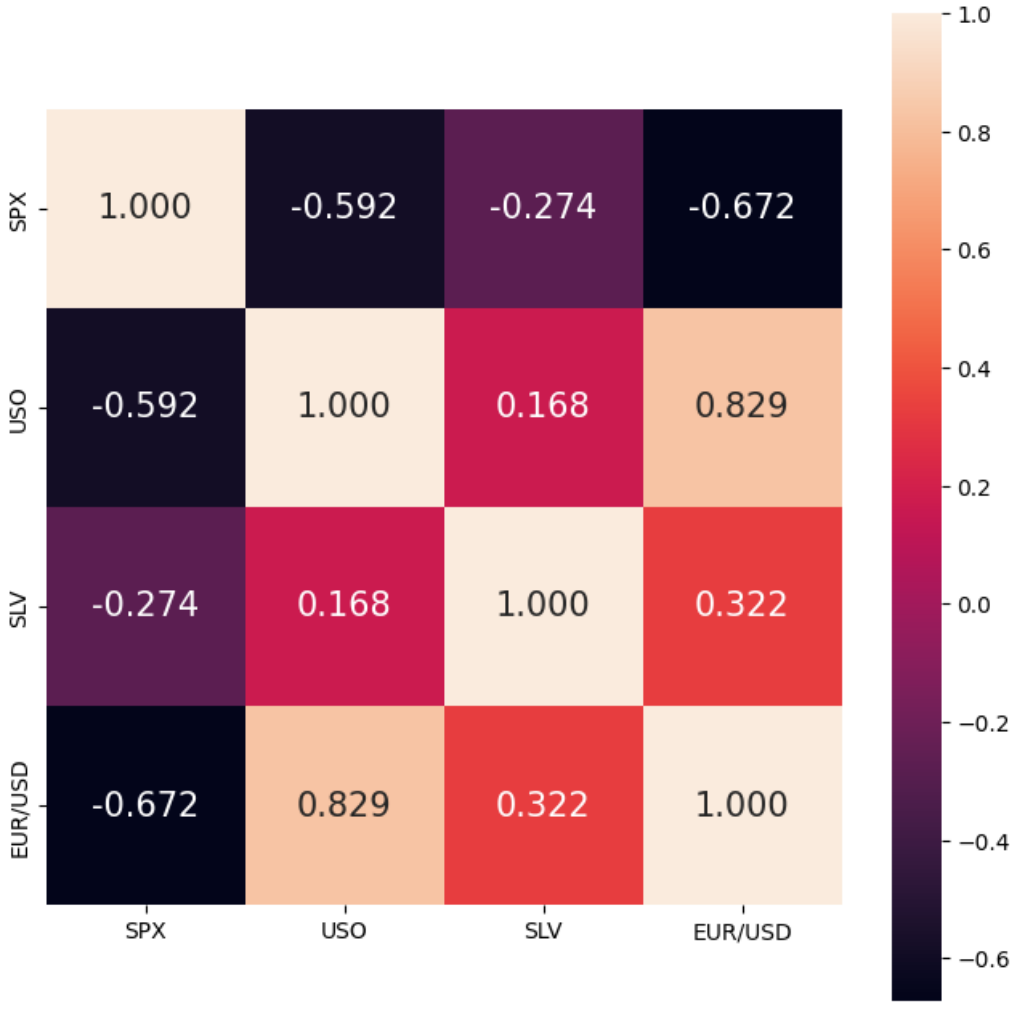
Correlation:

1. Positive Correlation -> Rising values in one variable align with increasing values in another
2. Negative Correlation -> Rising values in one variable align with decreasing values in another

```
df = data.drop(["GLD"], axis=1)
correlation = df.corr()
```

```
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.3f',annot=True, annot_kws={'size':15})
```

```
<Axes: >
```



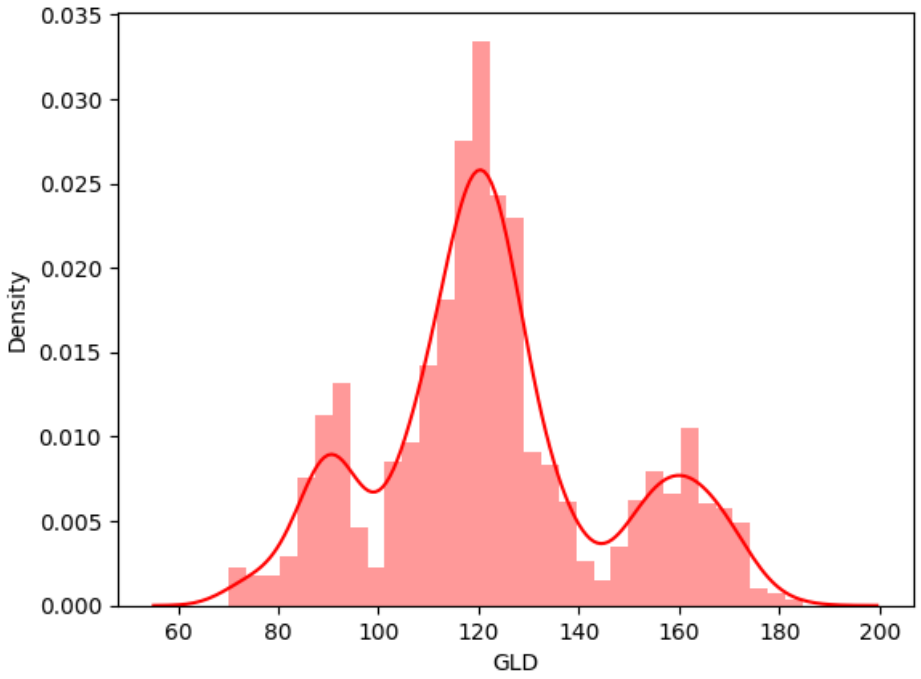
```
correlation = pd.DataFrame({"SPX": [1, 0.8, 0.7], "USO": [0.8, 1, 0.6], "SLV": [0.7, 0.6, 1], "GLD": [0.5, 0.4,
```

```
print(correlation["GLD"])
```

```
0    0.5
1    0.4
2    0.3
Name: GLD, dtype: float64
```

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

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




Splitting the Features and Target

```
print(A)
```

```
                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(B)
```

```
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(A, B, test_size = 0.2, random_state=42)
```

Model Training: Random Forest Regressor

(n_estimators) represents number of trees in forest. Usually the higher number of trees better to learn data.

```
regressor = RandomForestRegressor(n_estimators=500)
```

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

```
RandomForestRegressor
RandomForestRegressor(n_estimators=500)
```

Model Evaluation

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

```
print(test_data_prediction)
```

```
120.66528214 118.24786009  96.43193845 109.38677977 114.97409923
```

```
104.12169898 125.73490064 123.45203776 167.87951869 121.31776049
 87.30977883 131.81149809 121.7306404  107.59711948 168.73153995
126.08379778 127.13866127 113.76222083 135.29706074 125.14350129
144.04239893 123.32465931 118.54112033 120.80983998 166.89242006
 71.47400044 163.18081927 166.40155865 118.39968058 103.63287827
128.05343877 154.3269601  172.00858001 134.88683756 126.89579995
123.91120034 152.19557767  87.87061994 130.67316171 110.9736209
164.92946035 156.53805959 166.6346804  121.61077985  89.99194045
132.27560157  99.80829962 127.56877786 127.81671844 108.6872195
 90.87813895 153.12932089  95.0720987   87.86001931 125.04367947
 87.22145795  94.6057809  113.69443961 156.44364346 147.71166037
105.16766015 166.68377757 111.2747402  128.40924011  90.69507986
109.45787957  76.36916097 110.71303971 163.82323907 154.81863869
152.40800149 161.94693983  92.18333878 117.87190147  93.61060114
130.00084077 117.42744039 117.36376043 124.25604054 120.74844757
 97.79541937 168.64802087 146.33196163 124.53583857 169.27777838
 84.26652003 166.93213819 130.36610268 119.78828171  88.45811994
120.02431885  83.61723875 118.725861   113.78327887 116.70263923
154.59097798 134.84296349 118.10502137 118.8434782  123.15321862
115.82328116 118.51830012 122.25914056 146.27326078 149.65938097
168.35968086  98.14907923 160.01452008  93.15898029 141.16233984
121.38688087  83.96517874 106.49141985 123.48671974 169.5115571
 93.55209896  96.4377207  153.06274012]
```

```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", error_score * 100)
```

```
R squared error :  98.98651599396989
```

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()
```