-:Report:-

1. Name - Vishal Singh
2. Year - 3rd
3. Semester - 5th
4. Branch - C.S.E.
5. College - Rajarshi Rananjay SInh. Institute of Management & Technology
   (Affiliated to A.K.T.U.)
6. Github account link - https://github.com/Vishusing/RINEX

Screenshots of Major_Project1:-

1.

Importing the Libraries

```
[38] 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

```
[39] # 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

2.

```
[40] # 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

```
[41] # 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

3.

```
gold_data.shape
```

```
(2290, 6)
```

```
[42] # 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
```

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

```
Date       0
SPX        0
GLD        0
```

4.

```
# 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

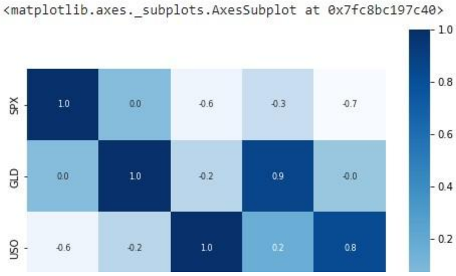
1. Positive Correlation
2. Negative Correlation

5.

```
[47] correlation = gold_data.corr()
```

```
[45] # 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>
```



6.

```
# 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
```

[49]
```
# 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 removed in a futur
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc8bc540880>
```

7.

[50]
```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

[51]
```
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]
```

[52]
```
print(Y)
```

```
0       84.860001
1       85.570000
2       85.129997
3       84.769997
```

8.

Splitting into Training data and Test Data

[53]
```
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

[54]
```
regressor = RandomForestRegressor(n_estimators=100)
```

[55]
```
#Training the model
regressor.fit(X_train,Y_train)
```

```
RandomForestRegressor()
```

Model Evaluation

[56]
```
# Prediction on test data
test_data_prediction = regressor.predict(X_test)
```

9.

```
[58] # 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

```
[63] Y_test = list(Y_test)
```

```
[64] 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')
```

10.

```
[64] 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()
```