Importing the necessary 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.metrics import r2_score,mean_squared_error,
mean_absolute_error
import warnings
warnings.filterwarnings("ignore")
```

Data Collection and Analysis

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

Displaying the first 5 rows of 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

Displaying the last 5 rows of 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

No. of rows and columns

```
shape = gold_data.shape
print("Rows",shape[0])
print("Columns",shape[1])
```

```
Rows 2290
Columns 6
```

```
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
```

```
gold_data['Date'] = pd.to_datetime(gold_data['Date'])
```

```
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   datetime64[ns]
 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: datetime64[ns](1), float64(5)
memory usage: 107.5 KB
```

```
gold_data['Day'] = gold_data['Date'].dt.day
gold_data['Month'] = gold_data['Date'].dt.month
gold_data['Year'] = gold_data['Date'].dt.year
```

```
gold_data.head()
```

```
        Date          SPX        GLD        USO     SLV   EUR/USD  Day
Month  \
0 2008-01-02  1447.160034  84.860001  78.470001  15.180  1.471692    2
1
1 2008-01-03  1447.160034  85.570000  78.370003  15.285  1.474491    3
1
2 2008-01-04  1411.630005  85.129997  77.309998  15.167  1.475492    4
1
3 2008-01-07  1416.180054  84.769997  75.500000  15.053  1.468299    7
1
4 2008-01-08  1390.189941  86.779999  76.059998  15.590  1.557099    8
```

```
1

   Year
0  2008
1  2008
2  2008
3  2008
4  2008
```

```python
gold_data.drop(labels=['Date'],axis=1,inplace=True)
```

```python
gold_data.head()
```

	SPX	GLD	USO	SLV	EUR/USD	Day	Month
Year							
0	1447.160034	84.860001	78.470001	15.180	1.471692	2	1
2008							
1	1447.160034	85.570000	78.370003	15.285	1.474491	3	1
2008							
2	1411.630005	85.129997	77.309998	15.167	1.475492	4	1
2008							
3	1416.180054	84.769997	75.500000	15.053	1.468299	7	1
2008							
4	1390.189941	86.779999	76.059998	15.590	1.557099	8	1
2008							

```python
gold_data.tail()
```

	SPX	GLD	USO	SLV	EUR/USD	Day	Month
Year							
2285	2671.919922	124.589996	14.0600	15.5100	1.186789	8	5
2018							
2286	2697.790039	124.330002	14.3700	15.5300	1.184722	9	5
2018							
2287	2723.070068	125.180000	14.4100	15.7400	1.191753	10	5
2018							
2288	2730.129883	124.489998	14.3800	15.5600	1.193118	14	5
2018							
2289	2725.780029	122.543800	14.4058	15.4542	1.182033	16	5
2018							

Check for missing values

```python
gold_data.isnull().sum()
```

```
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
```

```
Day        0
Month      0
Year       0
dtype: int64
```

Check for duplicate values

```
gold_data.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...  
2285    False
2286    False
2287    False
2288    False
2289    False
Length: 2290, dtype: bool
```

```
gold_data.duplicated().sum()
```

```
0
```

Statistical measures of 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   


               Day        Month         Year
```

```
count   2290.000000  2290.000000  2290.000000
mean      15.644541     6.329258  2012.724891
std        8.746132     3.591149     2.993271
min        1.000000     1.000000  2008.000000
25%        8.000000     3.000000  2010.000000
50%       15.500000     6.000000  2013.000000
75%       23.000000    10.000000  2015.000000
max       31.000000    12.000000  2018.000000
```

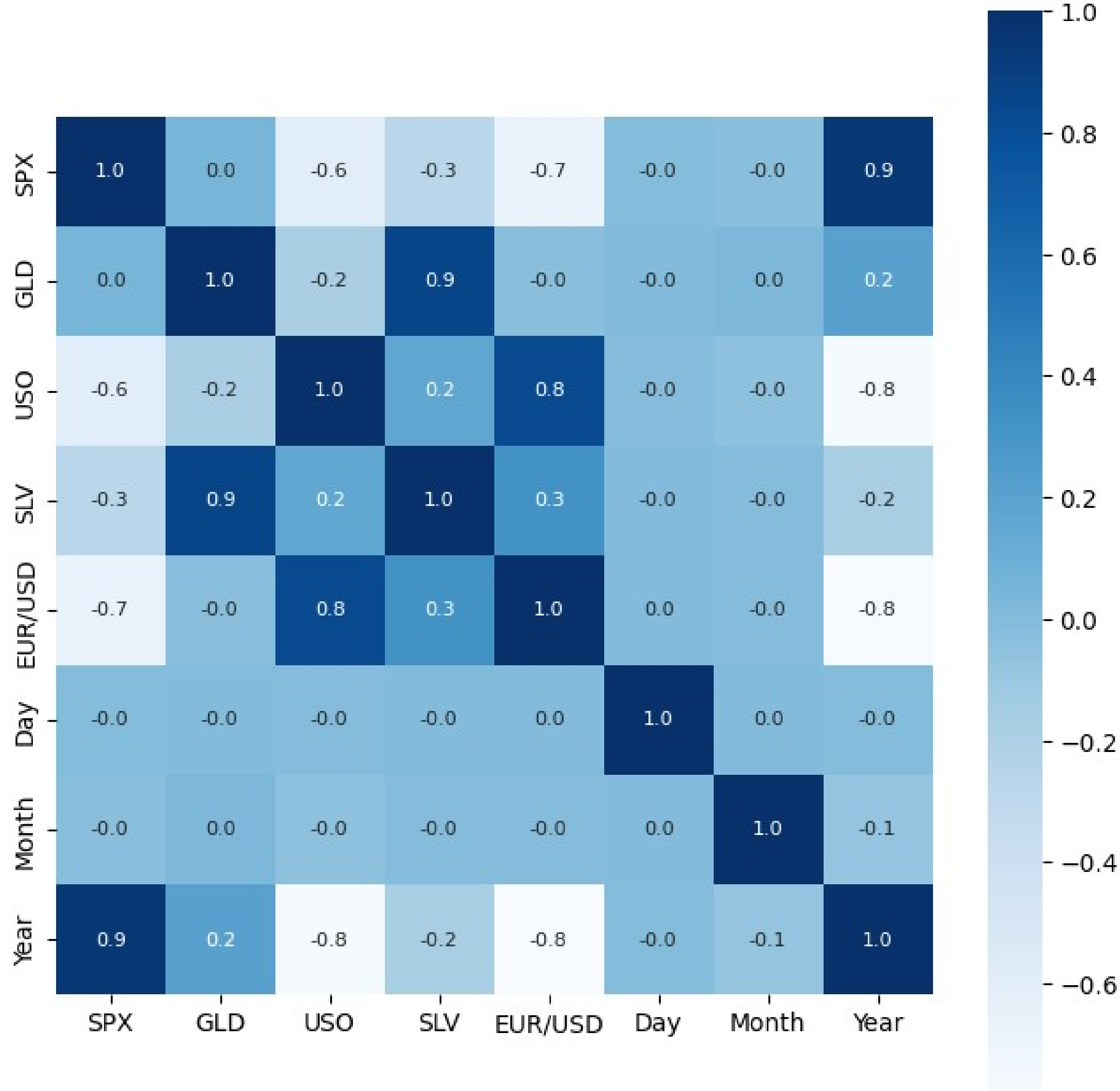
Check for correlation

1. Positive Correlation -> if 2 variables are directly proportional
2. Negative Correlation -> if 2 variables are inversly proportional

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

Constructing a heatmap for understanding correlation

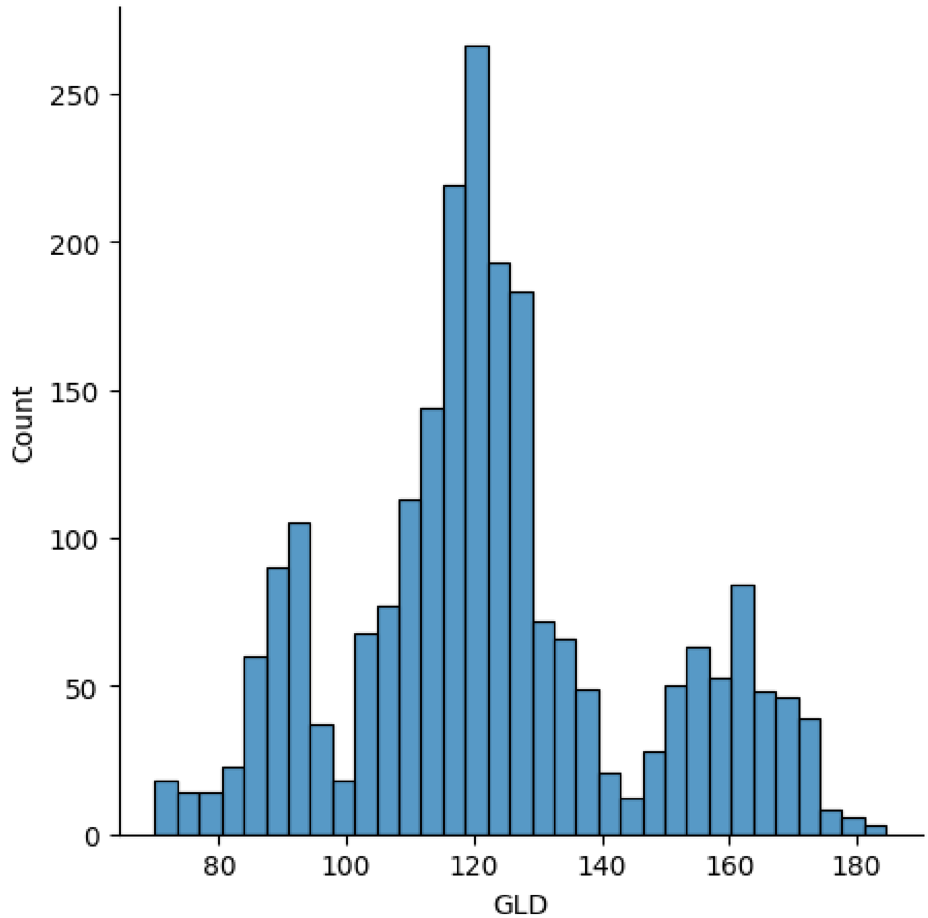
```
plt.figure(figsize=(8,8))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,ann
ot_kws={'size':8},cmap='Blues')
plt.show()
```




```
correlation['GLD']
```

```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Day       -0.000198
Month      0.020494
Year       0.206654
Name: GLD, dtype: float64
```

```
sns.displot(gold_data['GLD'])
plt.show()
```



Splitting the dataframe into independent and dependent features

```
X = gold_data.drop(labels=['GLD'],axis=1)
y = gold_data['GLD']

print(X)
```

	SPX	USO	SLV	EUR/USD	Day	Month	Year
0	1447.160034	78.470001	15.1800	1.471692	2	1	2008
1	1447.160034	78.370003	15.2850	1.474491	3	1	2008
2	1411.630005	77.309998	15.1670	1.475492	4	1	2008
3	1416.180054	75.500000	15.0530	1.468299	7	1	2008
4	1390.189941	76.059998	15.5900	1.557099	8	1	2008
...	...	...	...	...	...	...	...
2285	2671.919922	14.060000	15.5100	1.186789	8	5	2018
2286	2697.790039	14.370000	15.5300	1.184722	9	5	2018
2287	2723.070068	14.410000	15.7400	1.191753	10	5	2018
2288	2730.129883	14.380000	15.5600	1.193118	14	5	2018

```
2289  2725.780029  14.405800  15.4542  1.182033   16      5  2018

[2290 rows x 7 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 the data into train 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

```
regressor = RandomForestRegressor(n_estimators=100)

regressor.fit(X_train,y_train)

RandomForestRegressor()
```

Model Evaluation ->Prediction on test data

```
test_data_pred = regressor.predict(X_test)

print(test_data_pred)

[168.12319861  83.45989975 116.16550065 127.642401   120.16740078
 154.57959826 150.49939935 126.31580002 117.759099   126.08060079
 115.57280124 170.86170078 141.49929978 167.85419812 115.20920065
 117.36080062 134.4451009  171.2555025  159.84410352 172.44989994
 155.13320062 124.48800034 175.57850005 157.00770295 125.21680053
  93.62789898  77.02670013 119.20120027 118.97919892 167.35789855
  88.10680065 125.4247003   91.90680032 117.68329984 121.10680007
 135.61430156 115.82580053 114.53190095 142.02929945 107.40270074
 105.84140242  87.01399754 126.43380092 117.68040036 154.30019928
 120.06739946 108.3386998  108.1112973   92.852       127.25619697
  75.68699986 114.17250042 121.08280013 111.34229954 118.78679872
 121.04939902 160.24450128 174.40640005 146.32059691  87.22459973
```

93.39520016	86.76839868	89.66600039	119.13990095	126.42070046
127.7928996	172.0920012	122.1513999	117.7065984	97.56519994
168.21490228	142.33399865	133.09690301	120.57090064	123.54279889
119.53450122	114.27720153	117.86780054	107.53460044	128.19780014
114.71169944	106.39650013	117.58040139	119.49109894	88.507999
88.18679869	149.75970394	127.56360111	113.95399999	110.09849819
108.24749948	77.01249892	170.50030226	114.02649896	121.69619895
128.01020043	154.90619838	91.62919941	136.28790083	159.53510267
126.41630038	125.93999993	131.48990117	114.76860104	119.43299977
92.13579956	110.9416988	171.30630094	157.90919893	114.55580073
107.7263007	79.28040006	113.09430039	125.83540024	107.42410009
118.97400122	156.00400299	160.1646983	119.51479993	133.11810297
105.99489923	117.39209876	119.00860024	112.95920038	102.74089893
159.94139779	97.6394005	146.22429922	125.71890101	170.88519944
125.20060006	127.40699689	127.25500132	113.63279941	111.38190048
123.05929922	102.10999927	89.30639993	125.16529954	98.61249948
106.80429807	111.15420143	117.41430016	97.60280008	121.81220033
165.29690089	87.19149783	106.32939982	117.34030065	128.12070088
124.03630105	80.3778992	119.29550094	158.15059879	88.10079865
110.37959919	117.20709967	172.12169928	103.0472989	105.71090084
122.60569956	158.27649858	87.21049857	92.76230055	112.36240035
176.21499945	115.07229952	119.21610039	94.28970063	125.87179981
166.82170126	114.83140135	116.62040151	88.15209859	146.48069669
120.00729856	89.54369949	112.68110024	116.92400079	118.71300131
88.1466992	94.01009962	116.88690025	118.48770125	120.03430085
127.01849784	121.85669966	139.3554006	166.0816008	118.51499971
120.49590171	150.99030047	118.74249935	172.31339911	99.35359884
105.25770041	146.39319669	111.15540152	125.04730068	146.37469957
119.42470093	115.04740011	112.67270025	113.84420142	139.87540108
118.29439754	103.01640092	116.04890113	105.33910204	97.9246002
117.97300067	90.93019948	91.56689974	152.98349785	102.90809928
154.79790089	114.47890096	137.47190174	91.20469954	115.50629912
114.86230025	122.13960023	121.8384003	165.23370135	92.69169982
136.24360089	121.52249862	121.02440061	104.94970025	138.54820313
122.21069912	116.51280024	113.84360042	126.62419979	122.89879918
125.75259926	121.48419896	86.94599868	132.34240119	152.60559964
92.63639997	148.88959801	159.8493014	126.48009931	167.42449944
108.99699983	109.91520109	103.7019985	94.42130003	129.09520115
109.3948	149.68459916	121.81200001	132.10250012	131.63900134
160.69849799	90.16749868	173.4135015	127.19550109	126.93819848
86.23789918	124.83179919	150.23419691	89.58839962	106.96779897
109.77839975	86.58539904	136.50680033	154.72670257	137.37510361
73.96260039	153.04200058	126.47199893	126.7781999	127.55309869
108.87979885	156.71840181	114.67969973	117.08130158	123.9717
154.75990189	121.17699995	156.27549869	92.86800038	125.49620078
125.21840038	87.83540066	92.02309913	126.24069967	128.55230401
112.99579974	117.98869754	121.05619981	127.2253979	120.55380156
135.8109012	95.64370079	119.87350071	113.15860116	94.44929953
109.18229914	88.07739924	110.98119941	89.11380029	92.37400022

```
 131.8691039  162.35489937  89.12909973 119.25570089 133.6080016
 123.77219965 128.44280123 101.79309844  88.82569813 131.75680105
 121.06270118 108.39569989 170.35140039 115.61320099  86.87519919
 120.1440009   90.74319969 161.10030116 116.81440097 121.78499987
 160.36989797 120.06329947 111.62359916 108.68659967 126.63589978
  77.07579917 102.74010035 128.98240153 121.91939966  92.29879966
 132.70889982 117.49700089 116.3626997  154.61540262 160.51530048
 110.42669896 135.97619807 118.97190124 160.38329988 118.01489936
 160.020201   115.28749937 116.3003006  146.6282974  114.15190067
 125.49529886 166.1347979  117.52590045 124.97129958 152.82380359
 153.35500255 132.15050067 114.82339996 120.8617009  122.91480001
  90.13980054 123.21619972 152.93210052 111.58210021 106.43680061
 161.98480105 118.71689991 165.54150036 133.72670146 115.65619989
 152.73829751 168.96590044 115.0682001  114.14630137 161.26539899
  86.17199941 126.94260129 127.70920069 128.36220191 123.79310104
 123.95140095  90.49170092 152.33130141  96.82169995 137.08669991
  89.53999971 106.54700026 114.97350019 111.14980059 125.35469915
  91.27839911 125.45300136 162.18779754 118.34540175 165.27030128
 127.27679719 112.21370006 127.82580028  95.44509906  91.38909959
  98.91589941 120.9366999   83.47259912 126.15369992 160.58870314
 117.24100036 118.25349991 119.32949977 120.53390046 119.53740087
 121.07619976 117.92360034 108.25330025 146.41729704 125.30960017
 115.71940067  73.97450032 127.88070079 155.33790071 120.54830005
 125.651301    89.31630077 102.69569935 125.34109968 119.89759982
  73.33100113 151.15789975 121.10530014 104.6690995   86.33659798
 115.1119     171.27519854 120.39879992 161.78199656 112.92789896
 121.29480085 117.68910115  95.28459975 117.47740035 125.61970015
 118.36959968  96.24980119 153.87270151 122.1491     146.23009813
 159.91490318 113.55230051 121.77179972 146.33899699 127.77830066
 165.59779971 135.54550111 119.97730028 166.50229796 108.21569865
 122.01559876 137.89999969 102.88779892]
```

Compare Y_test and test_data_pred

```
score = r2_score(y_test,test_data_pred)
print(score)
```

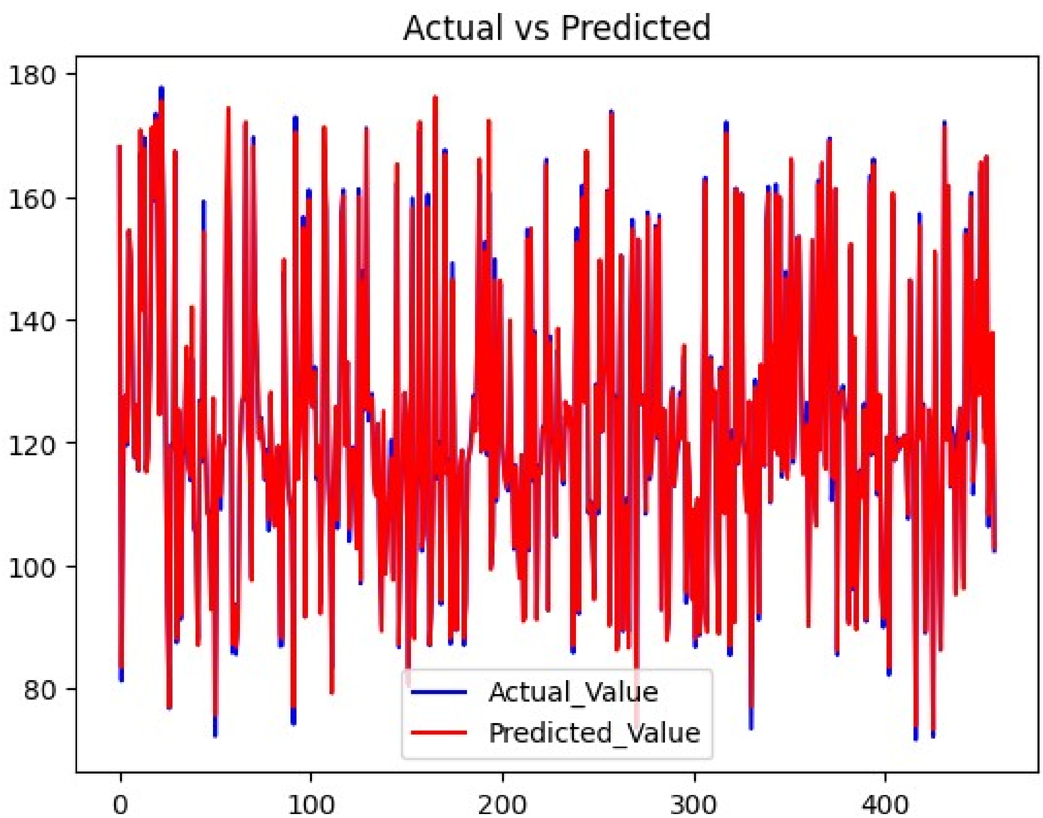
```
0.9951537829986654
```

```
mse = mean_squared_error(y_test,test_data_pred)
print("Mean Squared Error:", mse)

mae = mean_absolute_error(y_test,test_data_pred)
print("Mean Absolute Error:", mae)
```

```
Mean Squared Error: 2.556105660110906
Mean Absolute Error: 1.009208545436679
```

Compare the actual values and predicted values in plot

```
y_test = list(y_test)

plt.plot(y_test,color='blue',label='Actual_Value')
plt.plot(test_data_pred,color='red',label='Predicted_Value')
plt.title('Actual vs Predicted')
plt.legend()
plt.show()
```



Creating the predictive system

```
input_data = [1252.540039,101.459999,17.26,1.5673,2008,7,24] # y = 
91.330002

# Convert the list to a numpy array for easy manipulation
input_data = np.array(input_data)

# reshape array as we are predicting for one instance
input_data_reshaped = input_data.reshape(1,-1)

prediction = regressor.predict(input_data_reshaped)
print("Gold price for given input is:",prediction)

Gold price for given input is: [90.95869901]
```