Gold_Price_data_Prediction

April 19, 2025

[1]:
```python
import pandas as pd
```

[2]:
```python
import numpy as np
```

[3]:
```python
import matplotlib.pyplot as plt
```

[4]:
```python
import seaborn as sns
```

[5]:
```python
from sklearn.model_selection import train_test_split
```

[6]:
```python
from sklearn.ensemble import RandomForestClassifier
```

[7]:
```python
from sklearn.metrics import accuracy_score
```

[8]:
```python
gold_dataset=pd.read_csv('gld_price_data.csv')
```

[9]:
```python
gold_dataset.head()
```

[9]:

	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

[10]:
```python
gold_dataset.tail()
```

[10]:

	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

[11]:
```python
gold_dataset.shape
```

[11]: (2290, 6)

[12]:
```
gold_dataset.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
```

[13]:
```
gold_dataset.isnull().sum()
```

[13]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

[14]:
```
gold_dataset.describe()
```

[14]:

	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

[15]:
```
gold_dataset=gold_dataset.drop(['Date'],axis=1)
```

[16]:
```
gold_dataset.head()
```

[16]:

	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

[17]:
```
correlation=gold_dataset.corr()
```

[18]:
```
plot=plt.figure(figsize=(5,5))
sns.heatmap(correlation,cbar=True,fmt='.2f',annot=True,annot_kws={'size':
 ↪8},cmap='Blues')
```

[18]:
```
<Axes: >
```

Correlation values of Gold

[19]:
```
print(correlation['GLD'])
```

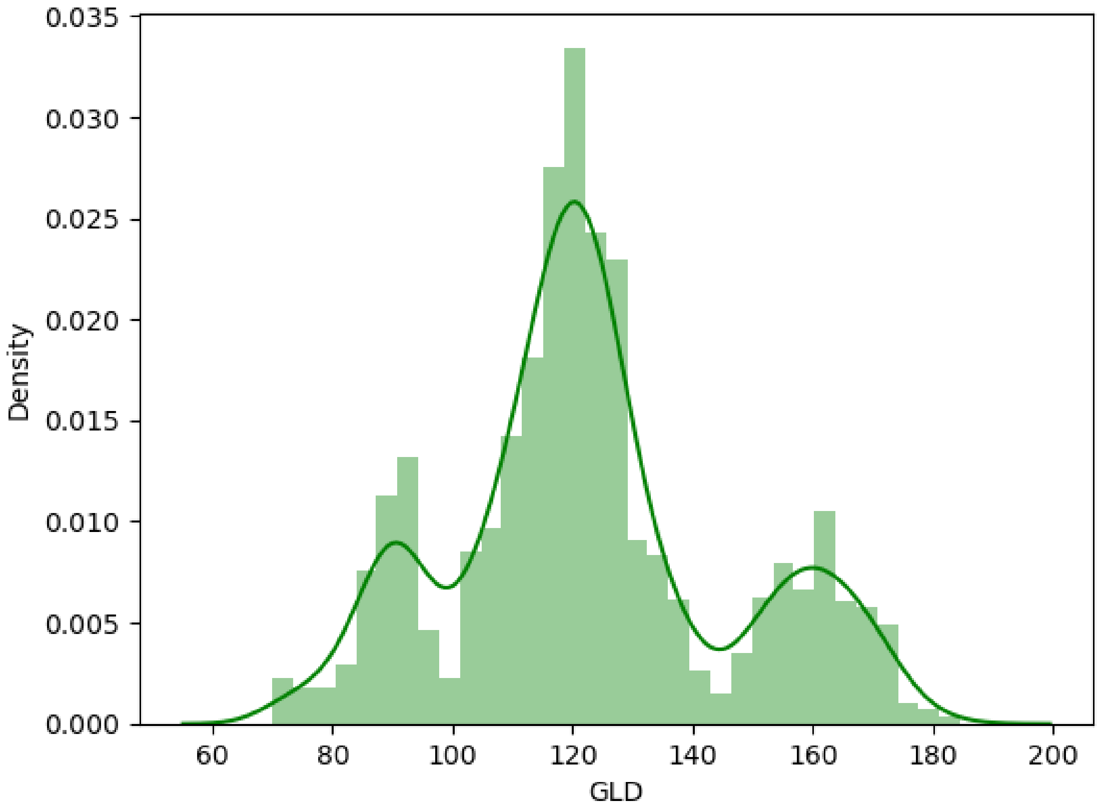
```
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

Checking the distribution of the goldprice

[20]:
```
sns.distplot(gold_dataset['GLD'],color='green')
```

```
C:\Users\indhu\AppData\Local\Temp\ipykernel_37420\1854168806.py:1: 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_dataset['GLD'],color='green')
```

[20]: <Axes: xlabel='GLD', ylabel='Density'>



Spliting the dataset into Features and Target

[21]:
```
X=gold_dataset.drop(['GLD'],axis=1)
```

[22]:
```
X.head()
```

[22]:
```
           SPX        USO     SLV   EUR/USD
0  1447.160034  78.470001  15.180  1.471692
1  1447.160034  78.370003  15.285  1.474491
2  1411.630005  77.309998  15.167  1.475492
3  1416.180054  75.500000  15.053  1.468299
4  1390.189941  76.059998  15.590  1.557099
```

[23]:
```python
Y=gold_dataset['GLD']
```

[24]:
```python
Y.head()
```

[24]:
```
0    84.860001
1    85.570000
2    85.129997
3    84.769997
4    86.779999
Name: GLD, dtype: float64
```

Spliting into training and test data

[25]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

Model Training

[26]:
```python
from sklearn.ensemble import RandomForestRegressor
```

[27]:
```python
regressor=RandomForestRegressor(n_estimators=100)
```

[28]:
```python
regressor.fit(X_train,Y_train)
```

[28]:
```
RandomForestRegressor()
```

Model Evaluation

[29]:
```python
test_data_prediction=regressor.predict(X_test)
```

[30]:
```python
print(test_data_prediction)
```

```
[168.78889959  81.98589997 116.17199986 127.57590098 120.62920123
 154.73569682 150.30019894 126.21659982 117.60039871 125.86740074
 116.64520071 171.50800049 141.83519857 167.88089844 115.04740008
 117.61050048 139.06080246 170.11520084 159.40120351 155.74739991
 155.15390022 125.06150017 176.63889947 156.8460039  125.15130044
  93.67139999  77.67870009 120.7698999  119.03929935 167.40169939
  88.39830072 125.10539983  91.20060059 117.75240028 121.10149884
 136.40780143 115.51290095 115.26090079 149.418      106.95170113
 104.37950246  87.22119808 126.53800025 117.94260026 153.08049903
 119.6506     108.38499999 108.27379887  93.33130072 127.07349819
  75.08680047 113.62749933 120.9847998  111.12499899 118.83879883
 120.54389975 159.53830056 169.4509009  147.16159679  85.93109879
```

94.24000021	86.80109885	90.58660019	118.98130063	126.46000082
127.57830001	169.98029934	122.23489945	117.45829869	98.72190077
167.93230091	142.88549858	131.54180237	121.19540219	121.02299947
119.73520061	114.54830186	118.2374007	106.88280127	127.88520079
113.99119999	106.78560013	116.83790076	119.45649936	88.92330051
88.31929864	146.24930251	127.1044996	113.63350033	110.19179813
108.16099891	77.7110991	168.86520137	113.98439902	121.61389943
127.81090176	154.89659731	91.71229936	135.78390152	158.87230297
125.69060069	125.45600068	130.2904006	114.80060155	119.78619974
92.11659988	110.362499	168.64839844	157.28379865	114.14539949
106.89340125	79.09819974	113.40570021	125.84810046	107.24879966
119.56910092	156.03360297	159.26289881	120.03390001	134.26280313
101.82459976	117.66179811	119.16519985	112.98580075	102.77309944
160.05549763	98.99660026	148.24439913	125.62010113	169.47929907
125.87429862	127.30489748	127.53070197	113.71499897	113.06880051
123.53319939	102.245299	89.26630002	124.39729981	102.57179934
106.93879941	113.28460076	117.3149008	99.09119983	121.84950053
163.21609938	87.25969847	106.9273997	116.98650043	127.73690117
124.18830078	80.79619917	120.29390036	158.20739886	87.79719973
110.32659922	118.89039926	172.44429877	103.03339897	106.02620078
122.44709993	158.93639799	87.62809854	93.25720025	112.73170014
177.91399952	114.81889946	119.19029987	94.74630144	125.68300009
166.21330119	114.74000091	116.64470135	88.41429869	148.86300108
120.1236995	89.58599955	112.58159985	116.96690044	118.8418012
88.2951995	94.28590041	117.1731002	118.54880191	120.26190032
126.9652976	121.8450997	152.34809979	165.3556006	118.5471998
120.3997014	150.71690035	118.56909942	172.18609884	105.28249922
104.99850113	149.33470138	113.89720087	124.85240129	147.45279968
119.60110099	115.35830069	112.71189985	113.41520221	140.31780128
117.85189764	102.99489999	115.78490082	103.93580175	98.71910059
117.37360087	90.7118	91.59520035	153.39759996	102.75359951
154.64530084	114.25120165	138.95090121	90.12999791	115.51629935
114.95100014	122.55590021	121.76120037	165.30270153	92.89689948
135.66810121	121.33539929	120.58080038	104.57850021	142.62330253
121.21339912	116.53490043	113.57450104	127.18079728	122.68629944
125.71519916	121.24510046	86.89429948	132.24120128	142.33040225
92.59969972	159.48399902	159.60620214	126.34489868	165.06089952
108.88949972	110.06500098	103.68419849	94.45430102	127.6929026
107.3545006	162.07829921	121.84790024	131.89690023	130.4129006
160.5441003	90.0889982	175.59980174	127.57859996	126.83289839
86.55979925	124.52789955	150.24299722	89.65829982	107.13779969
109.07750003	84.36079899	136.55279978	155.12960114	139.60150407
74.26440029	152.23880069	126.14880003	126.70539974	127.48929891
108.76559936	155.99770027	114.38610095	116.89410132	125.28779922
153.91470164	121.61109985	156.28689928	92.88980105	125.51990108
125.4141004	87.6844002	92.07269903	126.1262998	128.13010275
113.37550128	117.72919725	120.76209979	127.3116976	119.76040095
136.20570056	93.80529922	119.89700046	113.09210089	94.13129923

```
 108.93859963  86.77869909 109.20249986  89.60089988  92.51070007
 131.38420315 162.52060088  89.34309992 119.61060081 133.41940203
 123.6197998  128.40410129 101.98299828  89.10329887 131.77840026
 119.71770026 108.49769981 168.85830112 115.31980056  86.66209933
 118.86800073  90.87629956 161.69310036 116.43170047 121.67339976
 160.06249779 120.21539945 112.9368993  108.4056986  126.74480003
  76.40220012 103.03749988 127.42420253 121.91019921  92.6818
 132.01150022 118.16470122 115.94660005 154.71610239 159.46560092
 109.86189972 155.53739793 119.30400096 160.52590075 118.44750044
 158.32999993 115.05249926 116.79880034 148.54879872 114.6546009
 125.28139838 165.54600007 117.74490035 125.13889924 153.54040316
 153.46370234 132.37820041 114.6820004  121.28740207 124.51530057
  89.7207006  123.19300008 154.52670242 111.60880019 106.7487999
 162.00100166 118.75380006 165.63620037 133.83110095 115.00289964
 153.0963989  168.52749997 115.14600022 113.97450115 159.03339897
  85.53019869 127.06200078 127.9338007  128.85789988 124.28020094
 123.49860025  90.50100066 153.2712996   97.28249972 136.15970034
  89.15469889 106.73950005 115.08190061 112.56190066 124.27869915
  91.41329868 125.36610118 162.31349822 119.93949877 165.12750073
 126.98309744 112.29410021 127.6198991   95.04469934  91.12709968
 102.99569919 120.92010005  83.4274993  126.33469996 160.28000531
 117.30840067 118.13899991 120.11319965 122.42829944 120.02620129
 121.60369997 118.3802005  106.81640004 148.06489919 126.41549826
 115.83620106  73.99650036 127.82610102 154.97330013 122.86349994
 125.64990055  88.78680006 104.05669879 124.43880035 120.28930017
  73.20160118 151.31610104 120.95120006 104.54930008  86.17149778
 115.06419895 172.26969854 119.86640021 160.01419758 113.19199985
 121.42630011 118.65140115  95.99009987 118.68970003 125.76150008
 118.39359958  96.28190103 154.07040156 122.17030026 147.31380058
 159.38750154 113.34089985 122.41359941 150.74639794 126.94810027
 165.86650074 134.9767003  119.95029969 167.47289866 108.40979962
 121.87609826 138.89110053 107.31599874]
```

[31]:
```python
from sklearn import metrics
```

[32]:
```python
score1=metrics.r2_score(test_data_prediction,Y_test)
```

[33]:
```python
print(score1)
```

```
0.9878581983998764
```

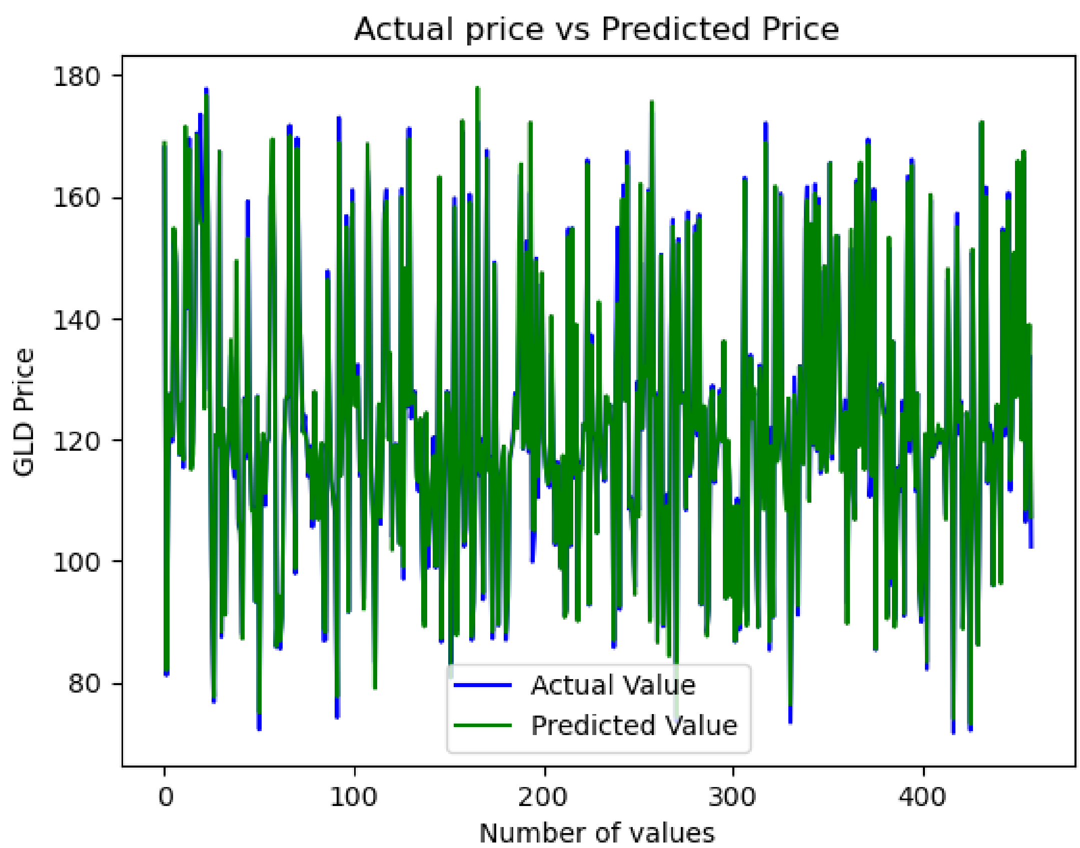
[34]:
```python
score2=metrics.mean_absolute_error(test_data_prediction,Y_test)
```

[35]:
```python
print(score2)
```

```
1.3448781177292586
```

[36]:
```python
Y_test=list(Y_test)
```

[37]:
```
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()
```



[38]:
```
import pickle
```

[40]:
```
filename='gold_model.sav'
pickle.dump(regressor,open(filename,'wb'))
```

[41]:
```
loaded_model=pickle.load(open('gold_model.sav','rb'))
```

[ ]:
```

```