gold-price-prediction-rand-forest

August 9, 2024

[1]:
```
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
```

[2]:
```
# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('gold_price_data.csv')
```

[3]:
```
# print first 5 rows in the dataframe
gold_data.head()
```

[3]:

	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

[4]:
```
# print last 5 rows of the dataframe
gold_data.tail()
```

[4]:

	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

[5]:
```
# number of rows and columns
gold_data.shape
```

[5]:
```
(2290, 6)
```

[6]:
```
# getting some basic informations 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
```

[7]:
```
# checking the number of missing values
gold_data.isnull().sum()
```

[7]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

[8]:
```
# getting the statistical measures of the data
gold_data.describe()
```

[8]:

	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

Splitting the Features and Target

[10]:
```
X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']
```

[11]:
```
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]
```

[12]:
```
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

[13]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,␣
↪random_state=42)
```

Model Training: Random Forest Regressor

[14]:
```
regressor = RandomForestRegressor(n_estimators=100)
```

[15]:
```
# training the model
regressor.fit(X_train,Y_train)
```

[15]:
```
RandomForestRegressor()
```

Model Evaluation

[16]:
```
# prediction on Test Data
test_data_prediction = regressor.predict(X_test)
```

[17]:
```
print(test_data_prediction)
```

```
[122.59609927 130.76910265 127.74740007  96.54939748 118.63830065
 114.56339968 124.73380146 117.79479961 108.06130139  98.37709988
  95.45289985 167.65259831 148.40030153 116.3743003  170.48180149
  85.04159987 123.24519891 108.9123972  113.15950058 131.42750326
 124.25969907 113.46970069 116.03230059 108.89800018 108.55050206
 125.80029958 119.6059991  112.63329896 113.54800165 125.81719884
 145.98660103  89.49480013 167.90619997 113.91509924 108.65540126
 120.186501   142.04469867 161.14700178 174.02429739 153.19060139
 119.4375012  113.67280055 121.43699926 113.57219921 122.07963824
 108.0881009   88.19889902 114.44289915 129.91330279 118.04820117
 103.42709982 129.89220261 107.09839832 160.711603   131.6566999
 118.44849973 146.51970036 136.1940017   95.33660122 124.62670144
 114.84069867  86.16460044 104.2407993  113.90470052  84.28899928
 122.29083812 116.5423992  113.57070185 165.65860315  92.2007001
  80.38280087 161.20120031 158.15940336 106.70620025 148.69920153
 109.57299782 123.38490064 128.45680056 113.17109883 120.13970072
 135.59219734 107.42130073  93.80930106  92.54409851 111.59740064
 118.4965999  108.67989939 112.24459969 167.86409871 160.96699841
 107.77489862 125.2582005  108.28350037 115.04940179 126.77699863
 108.59959946 163.23980262  84.46999863 131.3165032  114.18310032
 155.23280067 110.47279865 113.61749981 107.64160034 139.26470061
  88.3746995   92.37949922 175.22320166 119.02790069 118.76870051
 121.33960015 171.30019832 131.76959987 119.83650049 157.36200245
 118.91599866 119.04089964 110.76969933 119.81019894 122.10239983
 129.09899878 114.82800015  89.56689962 114.08150106 131.82309904
 115.58390113 125.17509976  90.98350052 106.866201   117.30110123
 109.80709968 166.63700193  80.56010063 121.95327545  73.34390151
 111.1926995  100.34350078 124.12880022  76.07919992 125.22569907
 119.75260062 104.68529985  90.55049932 131.71820007 137.32000216
 176.6335997  126.57469928 126.84479882 124.31609989  91.92859891
 149.12020121 102.97289894 117.76230017 133.96199854 136.08239917
 117.94900077 116.95050163 102.24669778 124.10509885  89.63610009
 108.2110994  117.5602005  168.2553012  117.2380006  117.69579972
 155.86180083 111.34840024  87.20259912 116.61220139 124.05369957
 120.66070204 118.30300015  96.48979871 109.33080017 115.00699894
 127.48670059 156.06710102 108.03850141 123.97499886 139.37650287
  90.92580061 118.17280123 130.34120106 114.1342992  108.38319957
 119.09180014 128.1363003  125.61810068 145.50670108 112.41680092
  93.61949998 115.0207001  125.52500077 120.49890131 122.33520074
  92.67750087 121.22829882  93.27360053 118.92120047 124.46400025
 121.86950029 131.3620003  124.39639921 114.78260145 127.41260023
 113.35630079 165.24659937 122.149798   119.65180188 114.3188008
 120.3435002  120.12649967 105.65310153 116.49110065 125.75799927
 172.20249714  85.65569978 134.51289867 127.67989816  73.85360075
 119.29919968  88.97259976 163.41190211  92.16780001 158.35390179
```

```
102.19359863 102.97839921 102.48149888 118.43309969 165.4251011
120.21580113 135.33899825  96.61709847 113.21139955 132.46910043
145.4837997  125.83780021 101.59830011 125.33390072 160.22640032
119.54550092 126.48420079 127.5294011  115.44069922 156.92960218
128.57150021 114.11249942 176.89479917 119.89050183 119.4520013
102.73259888 161.07089951 114.75730067 118.47829945 125.55729963
116.80300121 114.65539967  90.58779953 101.30770019 132.13420062
118.95260231 165.27039909 107.9759012   86.42390056  91.82439951
156.17039935 158.4526     152.98839835  72.93099952 120.90089993
117.27330021 159.03709981 135.64669841 111.75029965 113.76219976
160.86510146 125.44999938 119.56710123 118.10800011 158.42980248
104.01369949  89.39529967  83.46119918  90.33999936 115.4694002
113.6585     119.41850133 119.55670079  79.11600009  90.62620088
153.74170377 119.59430094 131.90680019 127.11450096 113.65070124
 82.36610083 118.21309906  89.85280025 117.88019953 163.30970219
121.52140059 110.60220016 125.43059822 114.42460083 135.91540001
 80.34580108 164.0258993  132.60940205 164.55080088 127.68489927
 91.65469924 108.32929936 114.31319949 127.78270113 119.46070194
 92.41919925 132.33479941 162.44700064  72.54540173 112.19690022
108.67549963 113.77289845 120.14980088 111.93599994 120.92659975
118.54150197 126.278101   125.71190135 109.30989969 167.49810017
166.97089865 112.52699945 169.45639691 112.00460017 161.82000256
127.34299868 167.83759915 135.38120192 109.29839866 167.43120011
116.94600135  72.57540117 113.64810051  93.29979974  87.88430066
104.09849889 125.91950066 123.30289777 165.39419943 121.56530075
 87.08549827 131.69899833 121.78360044 107.8928995  167.95869983
126.10489772 127.28410108 113.43280075 135.04010093 125.21350123
143.91769893 123.30179942 118.89360019 120.98169997 166.87469996
 71.82130061 163.5993992  167.38889903 118.26340061 103.68359811
127.83459824 154.55640021 172.12109996 135.29409701 126.87029998
123.30270026 152.06409797  88.01600035 131.16830193 110.92350078
163.59260044 156.52449906 167.18450105 121.6708003   90.26750048
132.26030156  99.65860004 127.52759832 127.70229842 108.70429934
 90.93109856 153.17970134  94.92399903  87.78819921 124.93599942
 87.24399815  94.48160144 113.34109966 156.31220327 147.3876999
105.31950009 166.53069739 111.13650016 128.29070066  90.77369999
109.46929923  76.91590118 110.90639963 163.47609934 155.63739829
152.36910191 161.92509998  92.08619868 117.9982017   93.50770147
130.14330041 117.36750053 117.52090049 124.31610021 120.74263795
 98.00409948 168.48600118 146.45810203 124.57349879 169.93039882
 84.01370005 166.47409837 130.53550293 119.75640181  88.43410025
119.91719879  83.79159887 118.75550085 113.68539899 116.64439921
155.01359751 134.07720427 118.4900012  118.91309857 123.23799874
115.80550114 118.5675001  122.2373004  146.74460054 149.02580014
167.9577004   98.24619921 160.04419966  92.9981001  138.8795997
121.44310092  84.1084987  106.60740006 123.47679952 169.08479625
 93.61959913  96.46440088 153.10179956]
```

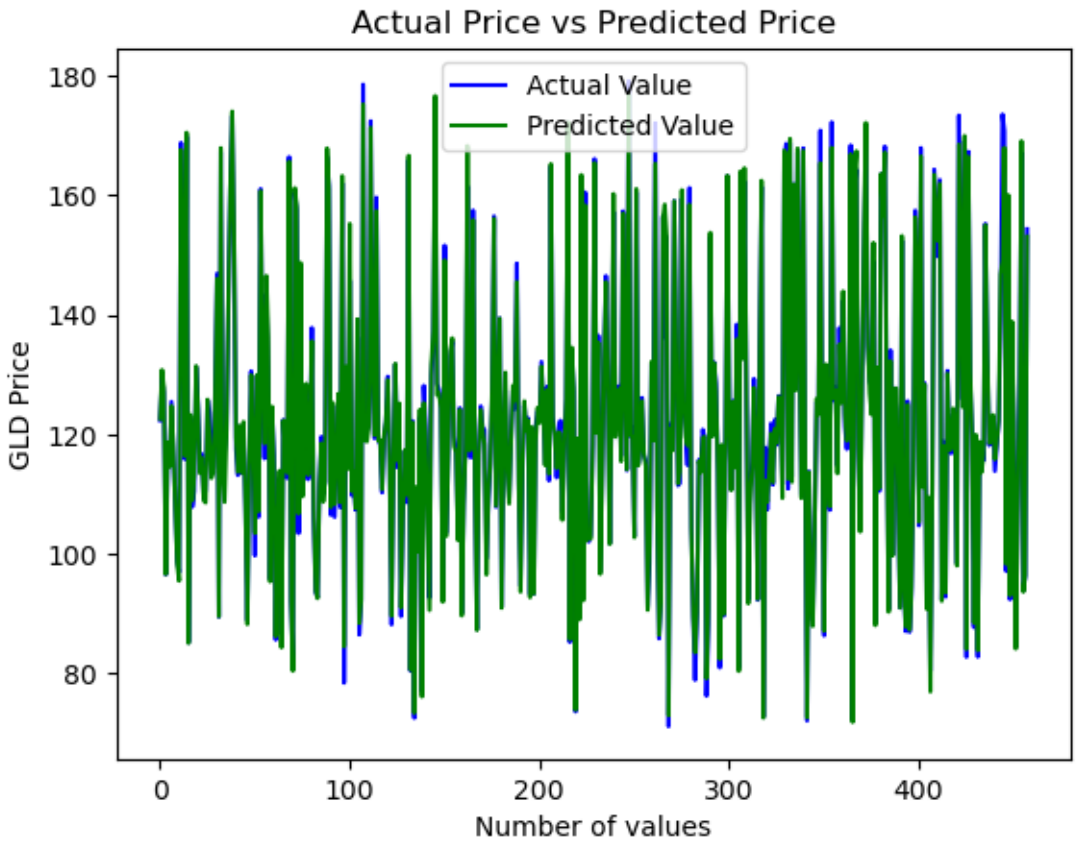
[18]:
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", error_score)
```

```
R squared error :  0.9896219003262587
```

Compare the Actual Values and Predicted Values in a Plot

[19]:
```
Y_test = list(Y_test)
```

[20]:
```
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()
```



[ ]: