qhatjvld5

March 3, 2025

[1]:
```
# Import all the Dependices

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

[3]:
```
# Data Collection and processing
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving gold_price_data.csv to gold_price_data.csv
```

[4]:
```
# Load the csv file into a pandas dataframe
gold_data = pd.read_csv('gold_price_data.csv')
```

[6]:
```
# Display the first five rows of the dataframe
gold_data.head()
```

[6]:

	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

[7]:
```
# peint the last five rows of the dataframe
gold_data.tail()
```

[7]:

	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
```

[8]:
```
# number of rows and columns
gold_data.shape
```

[8]:
```
(2290, 6)
```

[9]:
```
# 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
```

[10]:
```
# Checking the number of missing values
gold_data.isnull().sum()
```

[10]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

[11]:
```
# Getting stastical measures about the data
gold_data.describe()
```

[11]:

	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
```

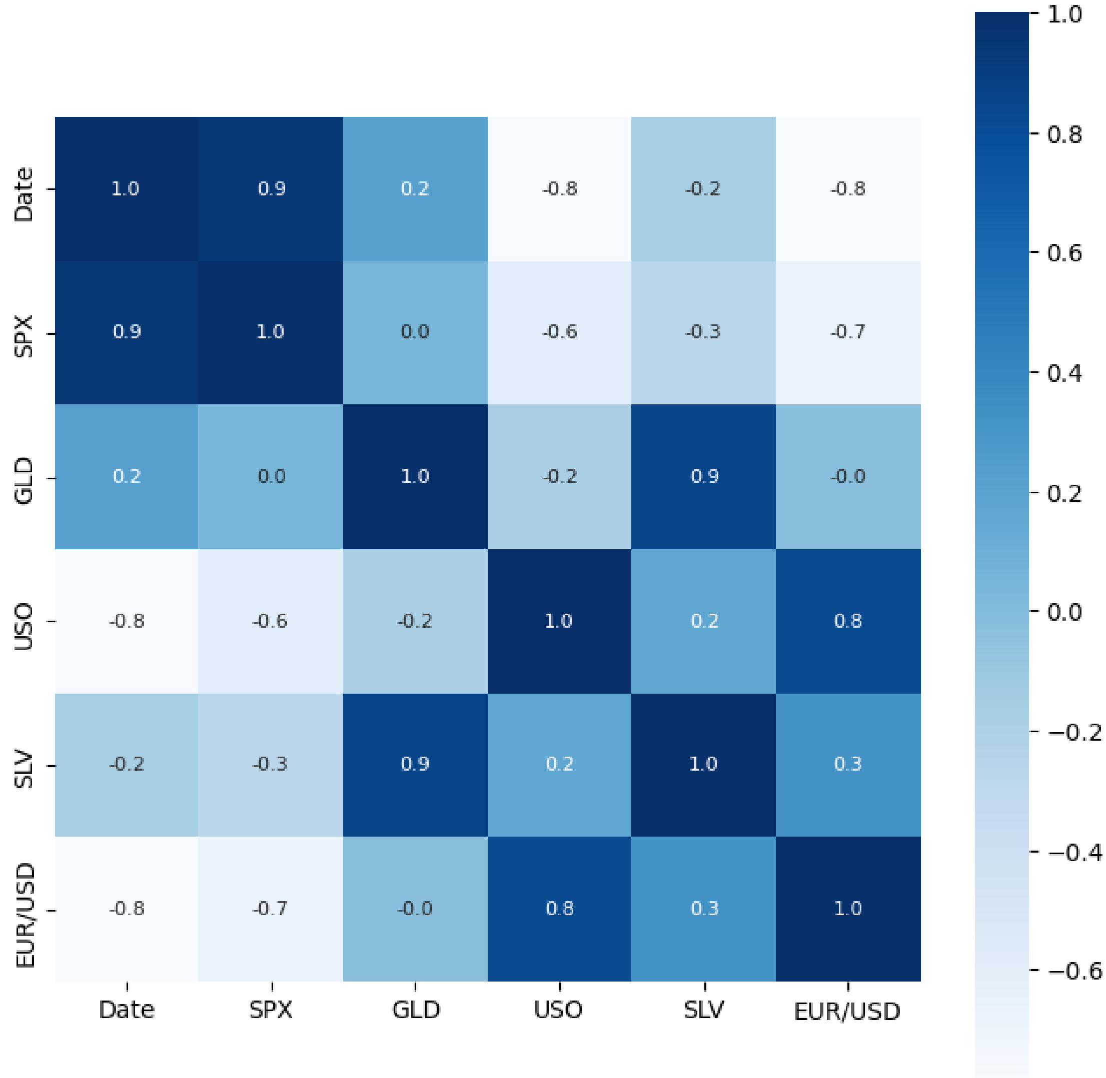
[14]:
```
# Conversion of Date column to datatime objects
gold_data['Date']= pd.to_datetime(gold_data['Date'])
```

[15]:
```
# Extract numerical features for correlation analysis
numerical_features = gold_data.select_dtypes(include=[np.number])
```

[16]:
```
# Find the correlationn
# Positive correlation - one value decrease the other one increase
# Negative correlation - one value increase the other value decrease
correlation = gold_data.corr()
```

[18]:
```
# Constructing heatmap 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')
```

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



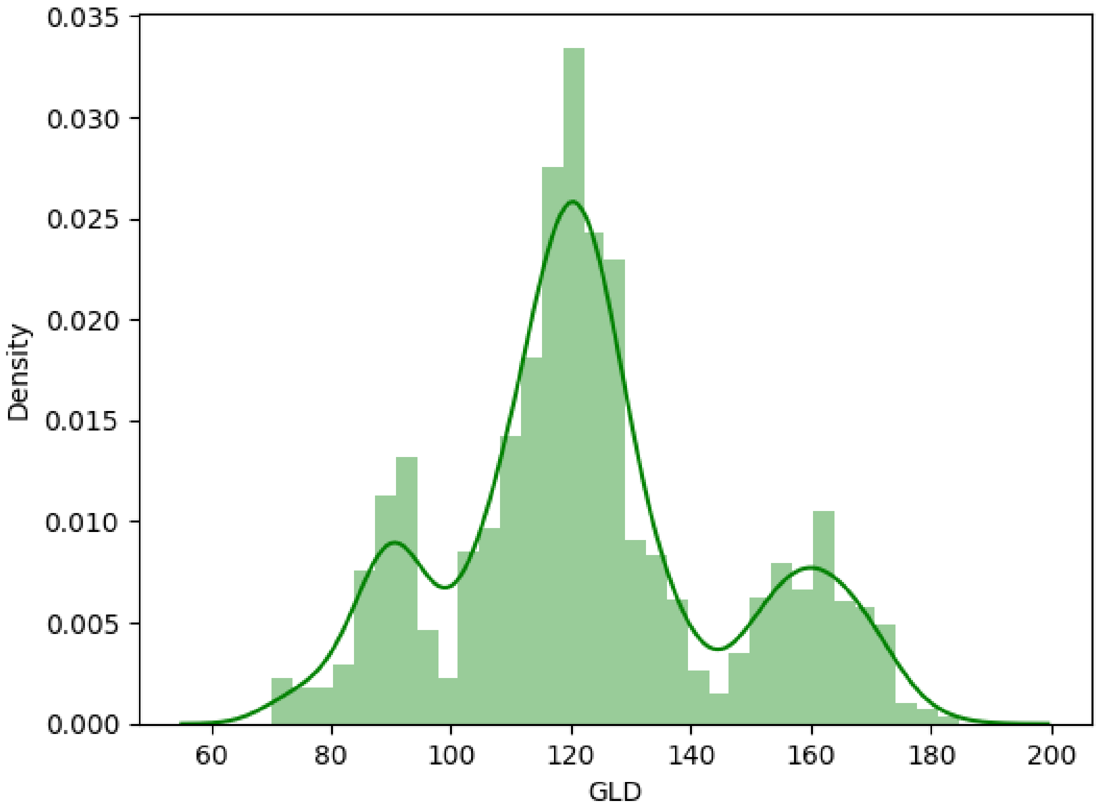
[20]:
```
# Correlation values of GMT
print(correlation['GLD'])
```

```
Date       0.209118
SPX        0.049345
GLD        1.000000
USO       -0.186360
SLV        0.866632
EUR/USD   -0.024375
Name: GLD, dtype: float64
```

[21]:
```
# Distribution of Gold prices
sns.distplot(gold_data['GLD'], color='green')
```

```
<ipython-input-21-8ccc92bf9399>:2: 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_data['GLD'], color='green')
```

[21]:
```
<Axes: xlabel='GLD', ylabel='Density'>
```



[23]:
```python
# Split the features and the target
X = gold_data.drop(['Date','GLD'], axis =1)
Y = gold_data['GLD']
```

[24]:
```python
print(X)
print(Y)
```

```
              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]
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
```

[25]:
```python
# Split into training and test data
X_train, X_test, Y_train,Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)
```

[27]:
```python
# Model training Using a Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100)
```

[28]:
```python
# Training the model
regressor.fit(X_train,Y_train)
```

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

[29]:
```python
# Model Evaluation
# Prediction on test data
test_data_prediction = regressor.predict(X_test)
```

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

```
[168.61639929  81.86009968 116.1287007  127.68350025 120.85330153
 154.66349777 150.47749906 126.03650063 117.583299   125.89600093
```

116.92000077	170.62320082	142.04319826	167.82899825	115.15559968
117.50150086	139.30750329	170.37550112	159.63260304	158.16210011
155.14059962	125.31780011	175.59569991	156.63090337	125.23580043
93.85529991	77.22370005	120.73620033	119.07189948	167.36200011
87.99900079	125.47400021	91.22460043	117.6568004	121.18759962
136.54840064	115.49970127	115.21100074	147.2259993	107.08300081
104.17390263	87.25719797	126.5021003	117.93879974	152.56229922
119.6317	108.29310024	108.20369801	93.22750048	127.00939804
75.17280015	113.76319933	121.37559969	111.30149925	118.82349909
120.74389944	158.76770042	168.20020096	147.19279722	85.94779854
94.43520026	86.90329875	90.57899982	119.06600049	126.3963008
127.47840005	170.4378004	122.23199951	117.33229903	98.30530032
168.51560079	143.27349768	132.09620232	121.3152024	120.98449911
119.62650041	114.61050178	118.33680061	106.9933009	127.75670111
113.89169976	107.47779992	116.71020066	119.68249856	89.1374009
88.30059896	147.12990277	127.30389968	113.53200029	110.28109846
108.24169914	77.53889896	169.48190198	114.14479913	121.63209955
127.77570175	154.99219852	91.73109911	135.32810125	158.92890314
124.99500047	125.34210034	130.41840181	114.80150143	119.73769978
92.06059992	110.20499864	168.49420003	156.30129963	114.20779962
106.72050131	79.08749997	113.38620003	125.79060034	107.25549913
119.31380139	156.0606026	159.27829981	120.21690001	135.48020242
101.51019968	117.39239805	119.28060028	113.0201008	102.87499958
160.26619828	99.4568005	147.48239889	125.8361011	169.82849949
125.86499898	127.24199799	127.5533016	113.81449951	113.14940073
123.76199866	102.26329892	88.99750018	124.50979947	101.62219928
107.38879902	113.73620059	117.25820114	99.15419965	121.86130043
163.36389901	87.42389894	106.88039959	117.42200058	127.60680107
124.24200051	80.69779927	120.14870041	158.57129804	87.88579982
110.51279922	119.05289911	171.37799843	103.02649892	105.62880067
122.33930046	159.09379756	87.53919807	93.29560028	112.7491002
176.50399877	113.98940016	119.23960016	94.69080136	125.88499999
166.31540141	114.87590096	116.8574014	88.3650988	149.09930112
120.36299995	89.41099966	112.30970034	117.13640008	118.81860101
87.79099911	94.19030044	116.62189983	118.57560176	120.31250056
126.72429863	121.94949955	150.71120052	164.86030049	118.62799964
120.27030154	150.33350031	118.42649915	172.2092986	105.4518994
104.91430097	149.64560134	113.92530104	124.75920095	147.5674997
119.4854013	115.41730047	112.61800025	113.53280202	141.44740154
117.9732976	102.9792005	115.74820122	103.689402	98.67740037
117.57500055	90.85759975	91.6796005	153.16269935	102.74269961
154.94210044	114.39010135	138.91920088	90.13299811	115.45049955
114.8995998	123.27160048	121.90039999	165.2798016	92.97039943
134.97740111	121.38509922	120.96310014	104.73140025	141.87760291
122.35349935	116.66630037	113.76610087	127.06839827	122.70859916
125.79189925	121.27240017	86.74559883	132.9932022	144.53120194
92.74429946	159.06009958	158.8652021	126.39799855	165.4867994
108.84889928	110.05330041	103.62129793	94.31680122	127.806603

```
 107.21740064 162.46349937 121.8415003  132.10840079 130.39840166
 160.37229974  90.10889826 173.64960182 127.99359987 126.81809866
  86.64579933 124.4436992  149.97379713  89.56709992 107.12539943
 109.02269995  84.69869887 136.34219921 155.00550322 138.88310364
  74.18820021 152.265      126.30060019 126.75490031 127.53469899
 108.64699941 156.50500041 114.7125014  116.83590134 125.05729976
 154.17680186 121.48179962 156.48309849  92.98580079 125.46600122
 125.66089985  87.95450051  92.21609885 126.25869934 128.53900385
 113.4164011  117.51499721 120.98639992 127.0411984  119.56660109
 136.78650089  94.04429945 119.80030039 113.27720078  94.42239933
 109.05309963  87.80199904 109.25219925  89.60659999  92.51740019
 131.71200243 162.45850011  89.44350013 119.51750097 133.20880162
 123.83620044 128.61410169 102.03079856  88.82779881 131.5662006
 119.99470029 108.86040033 169.05470096 115.30400049  86.56699881
 118.75750074  90.99219958 161.6946998  116.52670048 121.53619972
 160.14309781 119.97869953 112.98599914 108.43569874 126.73329979
  76.06850032 103.0255998  127.70890281 121.78539911  92.58539997
 132.14300106 118.08760082 116.0652     154.35530277 159.45440049
 110.13839973 155.7902979  119.1878008  160.49290137 118.34850057
 157.37369972 115.02739916 116.52550032 148.92919865 114.92930076
 125.3523989  165.8140999  117.85590014 124.68639903 153.0719037
 153.4836024  132.14530196 114.88810034 121.32480229 125.27200083
  89.76880008 123.06249975 155.02550188 111.74640046 106.78179946
 161.39440177 118.53099978 165.60350002 133.99560122 114.86609948
 153.0104991  168.80130051 115.02500049 113.95270137 157.00339813
  85.42169875 127.10640044 127.83220094 128.81750025 124.26740075
 123.99150091  90.69390077 153.2889995   96.96539997 137.58320037
  89.11899928 107.49889985 115.00050027 112.92990089 123.50129912
  91.33209844 125.44270124 162.45689906 120.1091987  165.10320135
 126.69229849 112.37680035 127.4009995   95.00349911  91.04999986
 103.2335991  120.8706999   83.38469952 126.46179998 160.5039048
 117.4724009  118.34199974 119.97990004 123.0729998  120.09550138
 121.62469996 118.23270052 107.11809992 148.38730064 126.25009841
 115.80470085  74.08369981 127.81810078 154.45380074 122.70380023
 125.64240021  88.81340022 103.23379849 124.0075003  120.1220002
  73.39900082 151.40139985 121.26949974 104.60770037  86.42559769
 115.21259947 172.15639867 119.98500046 159.27029772 113.2655994
 121.38610013 118.4325009   95.99159996 118.89390025 125.83450015
 118.54699962  96.01390076 154.32300195 121.95389984 147.21789973
 159.59810201 114.07099982 122.47649946 149.71099843 127.30320015
 165.78220037 134.93720006 119.99709947 167.71209891 108.3343993
 121.83119892 139.78530099 106.33399911]
```

[32]:
```
# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print('R squared:', error_score)
```

```
R squared: 0.9888148679656213
```

[39]:
```
# Compare actual values and predicted values

Y_test = list(Y_test)

# Compare actual values and predicted values
Y_test = list(Y_test)

Plot = pd.DataFrame(data={'Actual Values':Y_test, 'Predicted Values':
 ↪test_data_prediction})
plt.plot(Plot['Actual Values'], color='blue', label='Actual Values')
plt.plot(Plot['Predicted Values'], color='green', label='Predicted Values')
plt.title('Actual Values vs Predicted Values')
plt.xlabel('No of values')
plt.ylabel('GLD Price')
```

[39]:
```
Text(0, 0.5, 'GLD Price')
```