import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

df = pd.read\_csv('gld\_price\_data.csv')

#Lets have a quick look of dataset  
df.info()

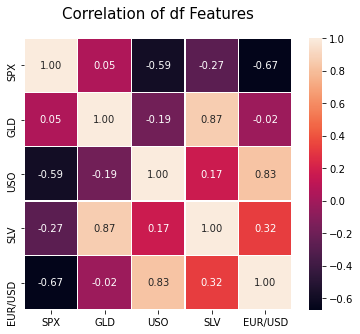
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2290 entries, 0 to 2289  
Data columns (total 6 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Date 2290 non-null object   
 1 SPX 2290 non-null float64  
 2 GLD 2290 non-null float64  
 3 USO 2290 non-null float64  
 4 SLV 2290 non-null float64  
 5 EUR/USD 2290 non-null float64  
dtypes: float64(5), object(1)  
memory usage: 107.5+ KB

#Clearly we see there is no null value in the dataset  
#Lets study the Statistical Inferance of the dataset  
df.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

#Now see the correlation matrix and heatmap  
import matplotlib.pyplot as plt  
import seaborn as sns  
corr = df.corr()  
plt.figure(figsize = (6,5))  
sns.heatmap(corr,  
 xticklabels=corr.columns.values,  
 yticklabels=corr.columns.values,  
 annot=True,fmt='.2f',linewidths=0.30)  
plt.title('Correlation of df Features', y = 1.05, size=15)

Text(0.5, 1.05, 'Correlation of df Features')



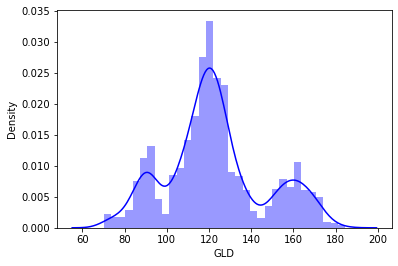
#Lets look the correlation score  
print (corr['GLD'].sort\_values(ascending=False), '\n')

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

#Lets Check our target variable  
sns.distplot(df['GLD'], color = 'blue')  
print('Skewness: %f', df['GLD'].skew())  
print("Kurtosis: %f" % df['GLD'].kurt())

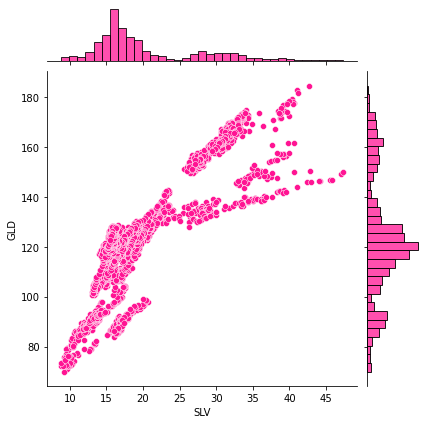
D:\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

Skewness: %f 0.3341383472692508  
Kurtosis: -0.275081



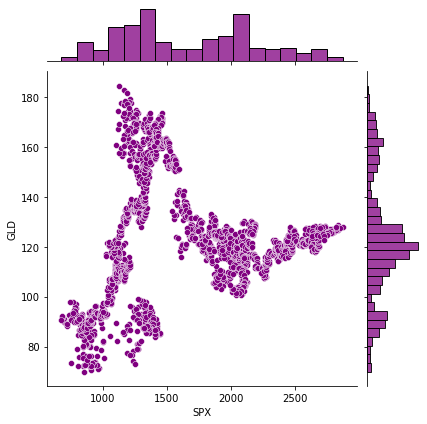
#Now we check the relation with GLD variable  
sns.jointplot(x =df['SLV'], y = df['GLD'], color = 'deeppink')

<seaborn.axisgrid.JointGrid at 0x1e43e2ecca0>



#Now we check the relation with GLD variable  
sns.jointplot(x =df['SPX'], y = df['GLD'], color = 'purple')

<seaborn.axisgrid.JointGrid at 0x1e43e4a5f70>



#Now Lets create a ml model  
# Now lets take our matrix of feature and target  
x\_trail = df[['SPX','USO','SLV','EUR/USD']]  
x = x\_trail.iloc[:, :].values  
y = df.iloc[:, 2].values

#Spliting the dataset into training and test set  
from sklearn.model\_selection import train\_test\_split  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 0)

#Now fitting the Random forest regression to the traning set  
from sklearn.ensemble import RandomForestRegressor  
regressor = RandomForestRegressor(n\_estimators = 100, random\_state = 0)  
regressor.fit(x\_train, y\_train)

RandomForestRegressor(random\_state=0)

#Now predicting the test set result  
y\_pred = regressor.predict(x\_test)

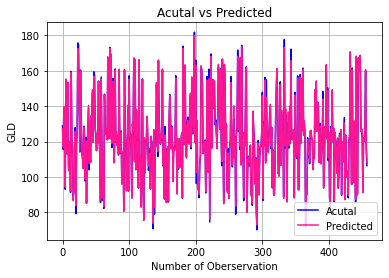
#Now Check the error for regression  
from sklearn import metrics  
print('MAE :'," ", metrics.mean\_absolute\_error(y\_test,y\_pred))  
print('MSE :'," ", metrics.mean\_squared\_error(y\_test,y\_pred))  
print('RMAE :'," ", np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

MAE : 1.297793151724892  
MSE : 5.16257387057774  
RMAE : 2.272129809358994

#Now Lets Check the Training and Test set Accuracy  
accuracy\_train = regressor.score(x\_train, y\_train)  
accuracy\_test = regressor.score(x\_test, y\_test)  
print(accuracy\_train)  
print(accuracy\_test)

0.9984324726699736  
0.9899648553789232

#Visualising the Accuracy of Predicted result  
plt.plot(y\_test, color = 'blue', label = 'Acutal')  
plt.plot(y\_pred, color = 'deeppink', label = 'Predicted')  
plt.grid(0.3)  
plt.title('Acutal vs Predicted')  
plt.xlabel('Number of Oberservation')  
plt.ylabel('GLD')  
plt.legend()  
plt.show()



# Displaying the predicted values  
  
y\_pred

array([127.11519861, 116.75389919, 139.64640069, 121.84449996,  
 94.16690037, 155.39120046, 117.54780113, 113.35050113,  
 153.44089984, 105.56870046, 103.22629882, 120.27290084,  
 92.02489943, 159.13370062, 121.44209864, 117.43930097,  
 86.60659837, 91.83999934, 92.9420998 , 125.58319999,  
 83.81939914, 117.61189833, 126.15909843, 172.60959729,  
 168.24839714, 139.56230262, 114.2682995 , 159.86910228,  
 133.47620133, 115.68370012, 110.98790099, 104.68350184,  
 128.26529987, 122.0387 , 98.8036998 , 120.18919953,  
 85.07499988, 114.99030011, 127.32199864, 140.42899853,  
 107.65770049, 131.61690135, 108.26699972, 134.65220058,  
 133.53169916, 149.28329945, 119.62220064, 156.90000052,  
 156.80590136, 127.58380013, 123.8733001 , 114.95029846,  
 123.92690041, 119.08869989, 153.83670055, 121.4558995 ,  
 112.25060008, 86.88879954, 155.34909997, 117.88850108,  
 88.12159929, 114.65729994, 83.17699948, 146.34709705,  
 125.31440012, 127.35959831, 125.1063993 , 121.05779968,  
 167.91400151, 121.59189905, 125.63030163, 172.81839774,  
 121.57550046, 169.49820115, 118.97590046, 127.33329946,  
 153.87929957, 107.88879795, 114.50539929, 160.39669954,  
 119.79550044, 125.54419809, 117.46819927, 123.41820011,  
 112.1592005 , 160.96839966, 114.59469967, 113.73909987,  
 125.34149829, 96.0939 , 113.63059975, 114.11040139,  
 80.74449876, 160.06179915, 125.5253 , 118.92580093,  
 92.10329986, 103.81730108, 140.61410201, 92.4138985 ,  
 155.3658037 , 122.28850017, 123.8988995 , 115.14869955,  
 87.91840109, 130.24520031, 125.58209957, 167.31780175,  
 108.51320044, 166.3898993 , 103.9408996 , 138.23369919,  
 115.48730129, 119.91000077, 93.62689977, 166.28150184,  
 113.39050003, 108.86569959, 82.83719984, 110.24659933,  
 155.2131013 , 90.6704 , 75.42170047, 122.02369964,  
 102.9510003 , 117.142299 , 121.47240005, 134.37920056,  
 119.57490017, 106.22049966, 106.57999894, 113.23989931,  
 93.06939967, 123.04079863, 118.73599965, 112.64710066,  
 73.68550073, 126.72900033, 82.6034994 , 115.30139987,  
 117.16769963, 153.36490299, 139.4103994 , 114.46260045,  
 166.04340222, 129.86270042, 124.67930016, 118.8200011 ,  
 124.48799888, 122.01590037, 160.99299908, 106.95910086,  
 116.88419939, 87.31789891, 123.58160044, 112.10430031,  
 85.48169925, 113.21610004, 117.06090011, 86.37909942,  
 132.28399901, 145.88859923, 135.30760411, 115.78540051,  
 128.00170051, 118.68920051, 97.21570068, 116.76050093,  
 114.97840122, 118.83619881, 155.8369014 , 122.26759915,  
 90.53709992, 120.95540072, 105.38809979, 121.59510016,  
 105.24699896, 107.17760119, 120.8903988 , 125.06480043,  
 88.24369888, 172.65259928, 120.65610152, 118.17550175,  
 163.38989932, 119.74410152, 110.67249906, 128.48339994,  
 131.763601 , 120.31219899, 118.62900059, 142.96290276,  
 147.24120123, 116.97549983, 156.57520273, 136.26310084,  
 124.70029954, 125.02859877, 179.94749669, 119.15010187,  
 163.13910392, 102.55480158, 132.0316019 , 92.69039981,  
 97.35859834, 90.20180076, 106.40050048, 112.49910025,  
 162.86050006, 160.90719936, 153.23970304, 113.34950131,  
 116.56120177, 118.93220161, 120.20360037, 91.57440175,  
 102.44670008, 154.70899973, 108.78239822, 102.70169902,  
 155.46489977, 76.01269948, 120.50530115, 168.13459811,  
 116.09269929, 144.11190171, 155.16970052, 139.61890002,  
 154.21519867, 131.26710404, 130.37209939, 124.23040043,  
 115.86830144, 135.49359845, 163.61760072, 154.32390161,  
 126.9983011 , 160.75879857, 130.08330012, 108.71439853,  
 166.59049958, 117.92009805, 84.77509943, 163.73590249,  
 153.35510038, 116.33820156, 129.49170033, 92.21179903,  
 90.85510094, 113.24069916, 87.80300045, 115.69409947,  
 119.41970057, 153.41379974, 117.12559861, 106.73850111,  
 118.07339938, 140.15699987, 110.77489932, 122.42059931,  
 83.4550989 , 125.6439995 , 125.71810069, 168.07400009,  
 126.97199916, 142.60169826, 117.40380166, 167.63970073,  
 106.68079894, 173.58450064, 109.86309862, 115.19040168,  
 88.24629885, 125.3796 , 117.4198999 , 153.14930284,  
 161.73059932, 79.91479946, 105.40769962, 110.35419968,  
 110.96919934, 107.92850025, 117.87360066, 89.7069998 ,  
 88.5851991 , 84.97550039, 137.22459899, 127.3365004 ,  
 119.54510085, 96.02360046, 109.99699775, 78.91229938,  
 72.72190197, 119.2031008 , 118.38450014, 89.71059971,  
 111.56310109, 117.22299937, 118.38180098, 124.65860057,  
 146.04579878, 87.7776994 , 129.39149821, 117.44520068,  
 154.78010199, 107.81169871, 144.91720231, 124.56199901,  
 116.50930012, 113.82469898, 125.17289942, 135.49890016,  
 140.96869907, 117.64730017, 128.25339833, 93.8205989 ,  
 114.56959982, 89.4910998 , 89.49360119, 106.3343006 ,  
 132.40470194, 152.88609861, 161.34780203, 118.72560048,  
 103.55099967, 100.35480122, 150.20039819, 152.34499794,  
 106.48070087, 137.54290164, 119.49549927, 126.27999819,  
 87.84500088, 173.02739956, 117.84919831, 138.58690157,  
 114.91459991, 85.84150001, 121.13139974, 136.61190118,  
 117.61380029, 140.00759923, 118.80570106, 165.72810085,  
 121.37540052, 123.06429858, 121.04839912, 126.80630114,  
 88.09760068, 146.38180203, 114.20530021, 111.28930013,  
 113.81520085, 155.62090224, 92.95930156, 92.56129995,  
 119.92720138, 156.32270211, 118.94230044, 155.64100361,  
 126.41470044, 161.03749807, 123.51419974, 131.62590116,  
 123.45600001, 74.13480004, 94.4866998 , 89.85819972,  
 121.06030148, 112.77130088, 117.76959969, 108.43280031,  
 126.05399986, 126.29900183, 122.45590006, 120.66129977,  
 104.02139909, 109.90559851, 110.46120126, 148.82150411,  
 125.84029842, 154.27260114, 125.85000087, 142.2277999 ,  
 140.90150086, 108.24039913, 109.80299824, 128.61390236,  
 107.26229943, 116.35199941, 120.73200093, 144.08180023,  
 93.550899 , 103.94560039, 128.50280081, 124.94209916,  
 163.38620021, 121.43980007, 148.57980401, 111.04219948,  
 125.23110073, 88.86760013, 80.84609893, 127.17169883,  
 125.97680033, 149.26670258, 120.05669894, 127.58039773,  
 86.63599905, 125.20979919, 118.65529996, 115.59139843,  
 102.01419885, 122.00560078, 93.5437987 , 102.80769896,  
 123.57809961, 114.88529985, 126.66970017, 90.19689993,  
 119.80340049, 108.89739928, 120.67329993, 130.51520122,  
 87.11199915, 123.17299964, 102.01429985, 99.55920157,  
 102.97119977, 89.5662995 , 120.746401 , 124.10140037,  
 170.73859771, 125.9796992 , 95.68860112, 160.56900452,  
 108.34709976, 96.69559938, 131.66059908, 113.49440068,  
 87.39639862, 168.25489901, 89.58750008, 90.34669903,  
 167.10009777, 159.7592026 , 165.99160146, 168.66539595,  
 125.69809824, 127.25499941, 122.3269996 , 91.25669912,  
 121.34859976, 120.1562008 , 129.37229838, 160.58889946,  
 116.88620006, 108.34129963])

# Predicting for a random train values by using rf regressor

regressor.predict([[1247,73.8,16.90,1.3]])

array([89.89109993])

# So the price of gold for the above given inputs is predicted to be 89

# The End