In [1]:
```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

In [2]:
```python
df = pd.read_csv('E:\paython\gld_price_data.csv')
```

In [3]:
```python
#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
```

In [ ]:
```
 FULL FORMS : 1) SPX- STANDARD AND POOR INDEX
              2) GLD- GOLD
              3) USO- UNITED STATES OIL FUND
              4) SLV- SILVER
              5) USD- DOLLAR
```

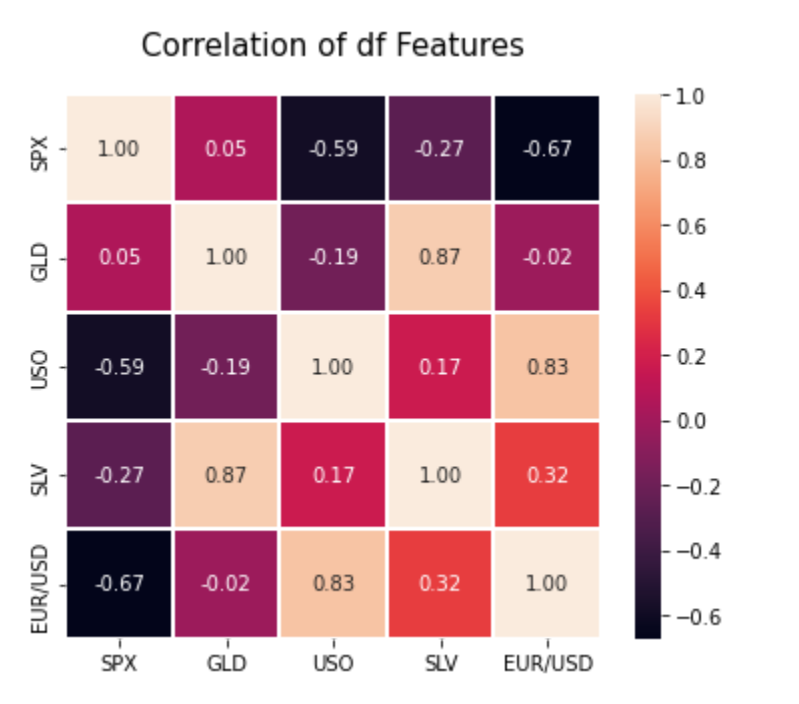
In [4]:
```python
#Clearly we see there is no null value in the dataset
#Lets study the Statistical Inferance of the dataset
df.describe()
```

Out[4]:

	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

In [7]:
```python
#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)
```

Out[7]:
```
Text(0.5, 1.05, 'Correlation of df Features')
```

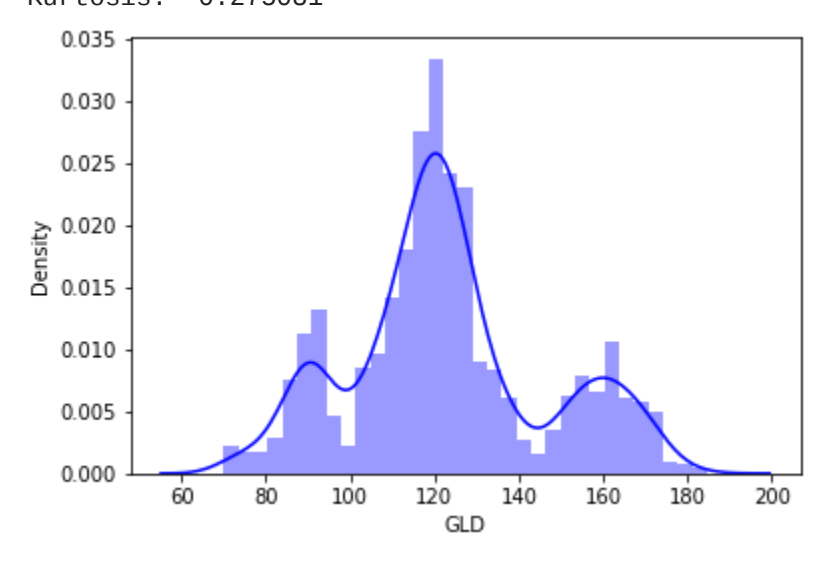


In [9]:
```python
#Lets look the correlation score
print (corr['GLD'].sort_values(ascending=False), '\n')
```

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

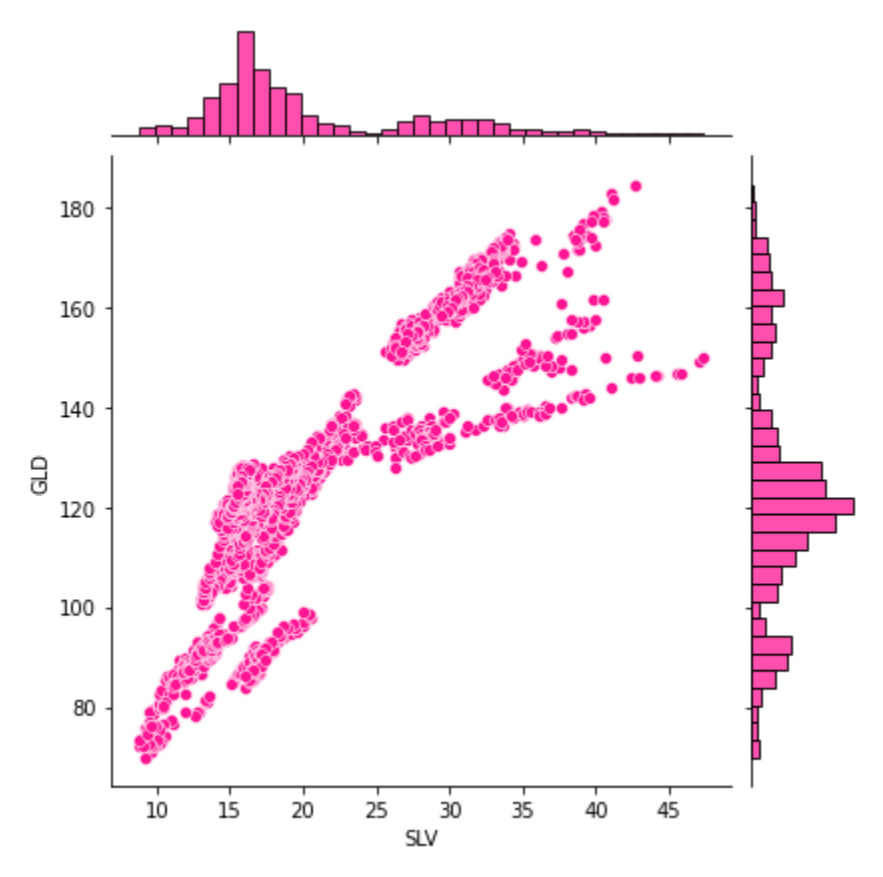
In [10]:
```python
#Lets Check our target variable
sns.distplot(df['GLD'], color = 'blue')
print('Skewness: %f', df['GLD'].skew())
print("Kurtosis: %f" % df['GLD'].kurt())
```

```
C:\Users\Raj\AppData\Local\Programs\Python\Python39\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
Skewness: %f 0.3341383472692508
Kurtosis: -0.275081
```



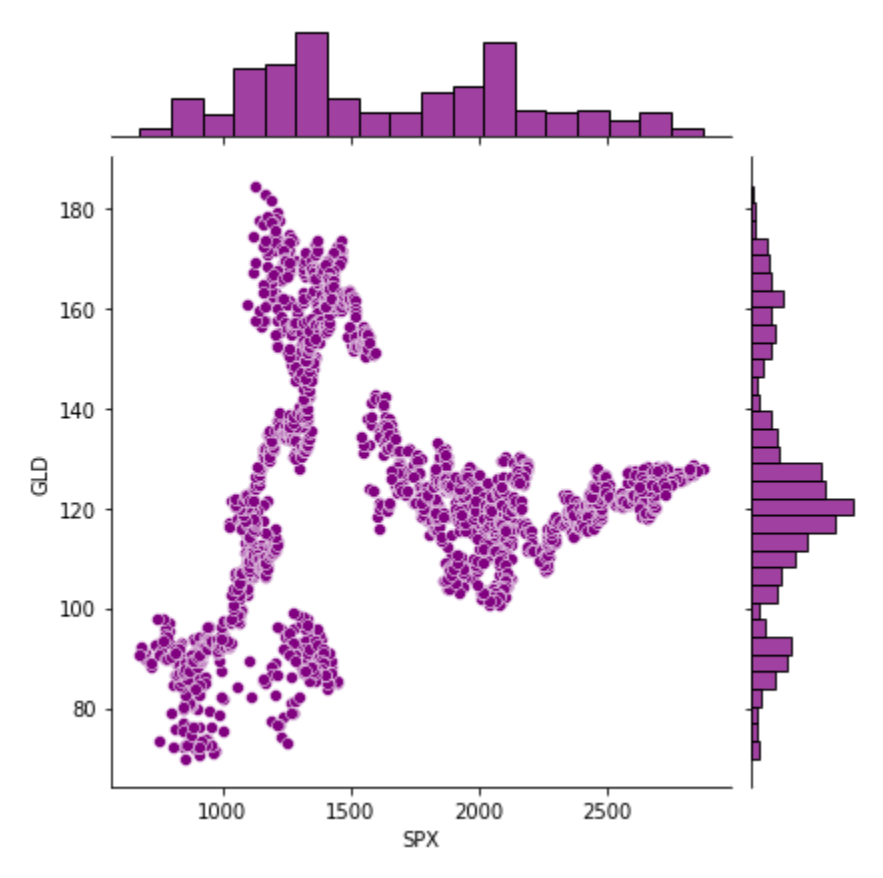
In [11]:
```python
#Now we check the relation with GLD variable
sns.jointplot(x =df['SLV'], y = df['GLD'], color = 'deeppink')
```

Out[11]:
```
<seaborn.axisgrid.JointGrid at 0x256f70ff1c0>
```



In [12]:
```python
#Now we check the relation with GLD variable
sns.jointplot(x =df['SPX'], y = df['GLD'], color = 'purple')
```

Out[12]:
```
<seaborn.axisgrid.JointGrid at 0x256f726e0d0>
```



In [13]:
```python
#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
```

In [14]:
```python
#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)
```

In [20]:
```python
#Now fitting the Random forest regression to the training set
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(x_train, y_train)
```

Out[20]:
```
RandomForestRegressor(random_state=0)
```

In [21]:
```python
#Now predicting the test set result
y_pred = regressor.predict(x_test)
```

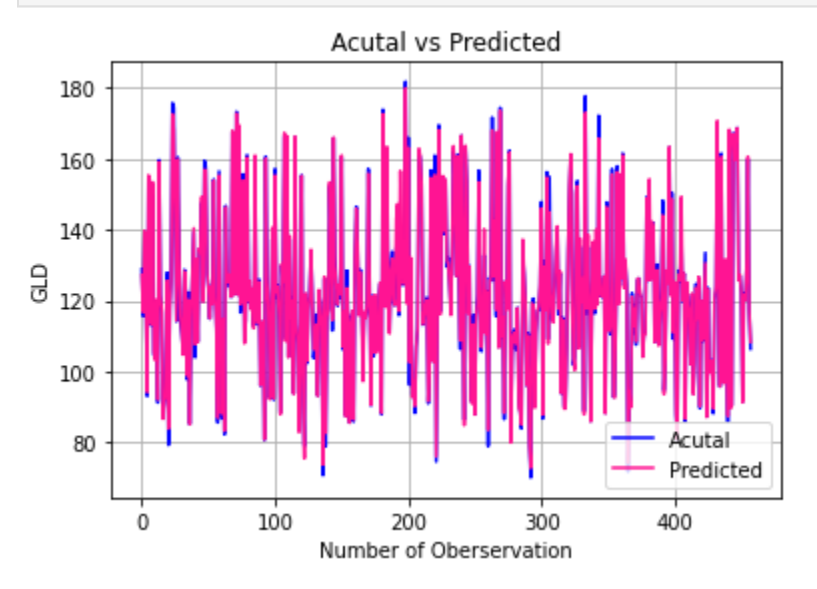
In [22]:
```python
#Now Check the error for regression
from sklearn import metrics
print('MAE :'," ", metrics.mean_absolute_error(y_test,y_pred))
print('MSE :'," ", metrics.mean_squared_error(y_test,y_pred))
print('RMAE :'," ", np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE :   1.297793151724892
MSE :   5.16257387057774
RMAE :   2.272129809358994
```

In [17]:
```python
#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
```

In [23]:
```python
#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()
```



In [34]:

In [52]: