QUERY 1-

3.1 Import the file 'gold.csv' (you will find this in the intro section to download or in '/Data/gold.csv' if you are using the jupyter notebook), which contains the data of the last 2 years price action of Indian (MCX) gold standard. Explore the dataframe. You'd see 2 unique columns - 'Pred' and 'new'. One of the 2 columns is a linear combination of the OHLC prices with varying coefficients while the other is a polynomial function of the same inputs. Also, one of the 2 columns is partially filled. Using linear regression, find the coefficients of the inputs and using the same trained model, complete the entire column. Also, try to fit the other column as well using a new linear regression model. Check if the predictions are accurate. Mention which column is a linear function and which is polynomial. (Hint: Plotting a histogram & distplot helps in recognizing the discrepencies in prediction, if any.) CAPM CAPM Analysis and Beta Calculation using regression - CAPM(Capital Asset Pricing Model) attempts to price securities by examining the relationship that exists between expected returns and risk. Read more about CAPM. (Investopedia CAPM reference) The Beta of an asset is a measure of the sensitivity of its returns relative to a market benchmark (usually a market index). How sensitive/insensitive is the returns of an asset to the overall market returns (usually a market index like S&P 500 index). What happens when the market jumps, does the returns of the asset jump accordingly or jump somehow? Read more about Beta (Investopedia Beta reference)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('GOLD.csv')
```

In [3]:
```
data.shape
```

Out[3]: (512, 9)

In [4]:
```
data.head()
```

Out[4]:

	Date	Price	Open	High	Low	Vol.	Change %	Pred	new
0	May 04, 2017	28060	28400	28482	28025	0.08K	-1.79%	738.0	117.570740
1	May 05, 2017	28184	28136	28382	28135	0.06K	0.44%	-146.0	295.430176
2	May 08, 2017	28119	28145	28255	28097	7.85K	-0.23%	30.0	132.123714
3	May 09, 2017	27981	28125	28192	27947	10.10K	-0.49%	357.0	101.298064
4	May 10, 2017	28007	28060	28146	27981	9.28K	0.09%	124.0	112.153318

In [5]:
```
data.isnull().sum()
```
Out[5]:
```
Date           0
Price          0
Open           0
High           0
Low            0
Vol.           0
Change %       0
Pred         101
new            0
dtype: int64
```

In [6]:
```
data_lr = data.dropna()
```

In [7]:
```
data_lr.shape
```
Out[7]:
```
(411, 9)
```

In [8]:
```
x = data_lr[['Price', 'Open', 'High', 'Low']]
y = data_lr['Pred']
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33,random_state = 42)
```

In [10]:
```
x_train.head()
```
Out[10]:

	Price	Open	High	Low
382	31631	31650	31780	31590
181	29761	29876	29884	29705
119	29541	29600	29622	29524
361	30614	30399	30660	30376
209	30306	30518	30597	30270

In [11]:
```
from sklearn import linear_model
```

In [12]:
```
lm = linear_model.LinearRegression()
```

In [13]:
```python
lm.fit(x_train,y_train)
```
Out[13]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [14]:
```python
lm.coef_
```
Out[14]:
```
array([ 2.,  3., -1., -4.])
```

In [15]:
```python
coeffs = pd.DataFrame(lm.coef_, x.columns, columns=['Coefficient'])
coeffs
```
Out[15]:

	Coefficient
Price	2.0
Open	3.0
High	-1.0
Low	-4.0

In [16]:
```python
predictions = lm.predict(x_test)
```

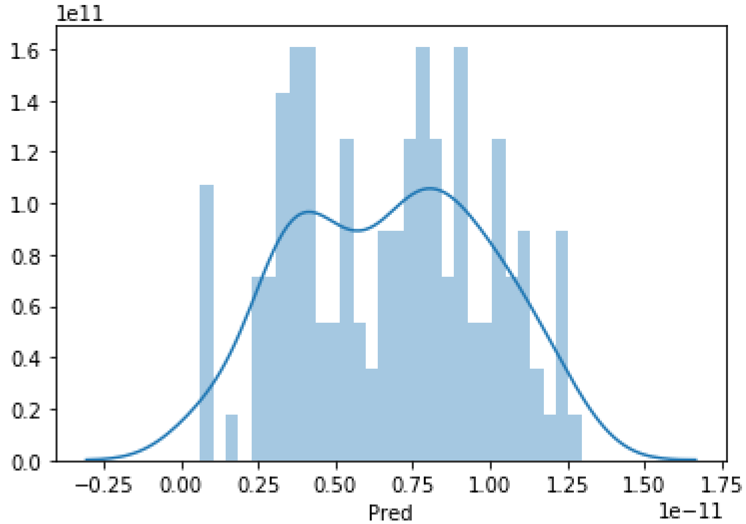
In [17]:
```python
data['Pred']=lm.predict(data[['Price','Open','High','Low']])
```

In [18]:
```python
data.shape
```
Out[18]:
```
(512, 9)
```

In [19]:
```python
data.isnull().sum()
```
Out[19]:
```
Date        0
Price       0
Open        0
High        0
Low         0
Vol.        0
Change %    0
Pred        0
new         0
dtype: int64
```

In [20]:
```
sns.distplot(y_test-predictions,bins=30)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2667b813940>



In [21]:

```python
import statsmodels.api as sm
X = data_lr[['Price','Open','High','Low']]
y1 = data_lr['Pred']
X1 = sm.add_constant(X)
model = sm.OLS(y1,X1)
results1=model.fit()
print(results1.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                   Pred   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 2.178e+28
Date:                Tue, 02 Jun 2020   Prob (F-statistic):               0.00
Time:                        15:41:08   Log-Likelihood:                 9574.4
No. Observations:                 411   AIC:                        -1.914e+04
Df Residuals:                     406   BIC:                        -1.912e+04
Df Model:                           4                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      1.886e-10   2.76e-11      6.831      0.000    1.34e-10    2.43e-10
Price         2.0000    1.5e-14   1.33e+14      0.000       2.000       2.000
Open          3.0000   1.28e-14   2.35e+14      0.000       3.000       3.000
High         -1.0000    1.5e-14  -6.68e+13      0.000      -1.000      -1.000
Low          -4.0000   1.52e-14  -2.63e+14      0.000      -4.000      -4.000
==============================================================================
Omnibus:                       24.254   Durbin-Watson:                   1.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               29.246
Skew:                           0.525   Prob(JB):                     4.46e-07
Kurtosis:                       3.779   Cond. No.                     1.80e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.8e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [22]:
```
import statsmodels.api as sm
X = data[['Price','Open','High','Low']]
y = data['new']
X1 = sm.add_constant(X)
model = sm.OLS(y,X1)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    new   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.118e+07
Date:                Tue, 02 Jun 2020   Prob (F-statistic):               0.00
Time:                        15:41:08   Log-Likelihood:                -538.12
No. Observations:                 512   AIC:                             1086.
Df Residuals:                     507   BIC:                             1107.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3282      0.738      0.445      0.657      -1.122       1.778
Price          1.0129      0.000   2210.033      0.000       1.012       1.014
Open          -1.0004      0.000  -2593.068      0.000      -1.001      -1.000
High           1.0050      0.000   2286.401      0.000       1.004       1.006
Low           -1.0177      0.000  -2233.145      0.000      -1.019      -1.017
==============================================================================
Omnibus:                      543.754   Durbin-Watson:                   1.932
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            76044.049
Skew:                           4.382   Prob(JB):                         0.00
Kurtosis:                      62.057   Cond. No.                     1.46e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.46e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

QUERY 2-

3.2 Import the stock of your choosing AND the Nifty index. Using linear regression (OLS), calculate - The daily Beta value for the past 3 months. (Daily= Daily returns) The monthly Beta value. (Monthly= Monthly returns) Refrain from using the (covariance(x,y)/variance(x)) formula. Attempt the question using regression.(Regression Reference) Were the Beta values more or less than 1 ? What if it was negative ? Discuss. Include a brief writeup in the bottom of your jupyter notebook with your inferences from the Beta values and regression results

In [23]:
```
maruti = pd.read_csv('MARUTI.csv')
maruti= maruti[maruti.Series=='EQ']
maruti.index=maruti.Date
nifty=pd.read_csv('Nifty50.csv')
nifty.index=nifty.Date
prices = pd.concat([maruti['Close Price'],nifty['Close']], axis=1)
prices.columns=['maruti','nifty']
print(prices.head())
returns=prices.pct_change()
returns=returns.dropna(axis=0)
returns.head()
```

```
              maruti    nifty
Date
15-May-2017  6823.90  9445.40
16-May-2017  6953.95  9512.25
17-May-2017  6958.20  9525.75
18-May-2017  6831.05  9429.45
19-May-2017  6790.55  9427.90
```

Out[23]:

	maruti	nifty
Date		
16-May-2017	0.019058	0.007078
17-May-2017	0.000611	0.001419
18-May-2017	-0.018273	-0.010109
19-May-2017	-0.005929	-0.000164
22-May-2017	-0.013084	0.001098

In [24]:
```
returns=returns.iloc[-60:,:]
```

In [25]:
```
returns
```

Out[25]:

Date	maruti	nifty
11-Feb-2019	0.006118	-0.005007
12-Feb-2019	-0.002145	-0.005271
13-Feb-2019	-0.018743	-0.003485
14-Feb-2019	-0.001530	-0.004410
15-Feb-2019	-0.012462	-0.002015
18-Feb-2019	-0.012605	-0.007781
19-Feb-2019	-0.005575	-0.003440
20-Feb-2019	0.006665	0.012363
21-Feb-2019	-0.007058	0.005067
22-Feb-2019	0.016298	0.000167
25-Feb-2019	0.001808	0.008196
26-Feb-2019	-0.002975	-0.004118
27-Feb-2019	0.009299	-0.002644
28-Feb-2019	-0.019918	-0.001309
01-Mar-2019	0.015440	0.006579
05-Mar-2019	0.026330	0.011410
06-Mar-2019	-0.008549	0.005966
07-Mar-2019	-0.003946	0.000470
08-Mar-2019	-0.008728	-0.002062
11-Mar-2019	0.016390	0.012020
12-Mar-2019	0.010633	0.011922
13-Mar-2019	-0.009242	0.003584
14-Mar-2019	-0.000670	0.000137
15-Mar-2019	-0.000325	0.007370
18-Mar-2019	-0.024689	0.003094
19-Mar-2019	-0.012143	0.006124
20-Mar-2019	-0.022219	-0.000984

	maruti	nifty
Date		
22-Mar-2019	-0.018109	-0.005568
25-Mar-2019	-0.004594	-0.008960
26-Mar-2019	0.009674	0.011361
27-Mar-2019	-0.010280	-0.003327
28-Mar-2019	0.012005	0.010917
29-Mar-2019	0.011567	0.004659
01-Apr-2019	0.025200	0.003893
02-Apr-2019	0.007163	0.003775
03-Apr-2019	0.026590	-0.005912
04-Apr-2019	0.005684	-0.003946
05-Apr-2019	-0.000759	0.005859
08-Apr-2019	0.003060	-0.005267
09-Apr-2019	0.012217	0.005812
10-Apr-2019	-0.004185	-0.007509
11-Apr-2019	0.000209	0.001070
12-Apr-2019	0.021564	0.004031
15-Apr-2019	0.001314	0.004028
16-Apr-2019	0.014424	0.008280
18-Apr-2019	-0.001488	-0.002914
22-Apr-2019	-0.016945	-0.013473
23-Apr-2019	-0.037200	-0.001596
24-Apr-2019	-0.004568	0.012975
25-Apr-2019	-0.015884	-0.007193
26-Apr-2019	-0.009037	0.009694
30-Apr-2019	-0.025786	-0.000553
02-May-2019	0.002528	-0.001992
03-May-2019	0.004003	-0.001066
06-May-2019	-0.000052	-0.009733
07-May-2019	-0.001140	-0.008652

	maruti	nifty
Date		
08-May-2019	-0.007736	-0.012041
09-May-2019	-0.003789	-0.005075
10-May-2019	0.001004	-0.002026
13-May-2019	-0.013247	-0.011588

In [26]:
```
X = returns['nifty']
y = returns['maruti']
X1 = sm.add_constant(X)
model = sm.OLS(y, X1)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 maruti   R-squared:                       0.151
Model:                            OLS   Adj. R-squared:                  0.136
Method:                 Least Squares   F-statistic:                     10.28
Date:                Tue, 02 Jun 2020   Prob (F-statistic):            0.00219
Time:                        15:41:09   Log-Likelihood:                 179.65
No. Observations:                  60   AIC:                            -355.3
Df Residuals:                      58   BIC:                            -351.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0016      0.002     -1.006      0.319      -0.005       0.002
nifty          0.7523      0.235      3.206      0.002       0.283       1.222
==============================================================================
Omnibus:                        1.985   Durbin-Watson:                   1.401
Prob(Omnibus):                  0.371   Jarque-Bera (JB):                1.208
Skew:                          -0.196   Prob(JB):                        0.547
Kurtosis:                       3.574   Cond. No.                         147.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [27]:
```
prices['month']=prices.index.str.slice(3)
```

In [28]:
```
month=np.zeros((25, 2), dtype=float)
for i,j in enumerate (prices.month.unique()):
    temp=prices[prices.month==j]
    month[i]=temp.iloc[-1,0:2]
month
```

Out[28]:
```
array([[ 7211.  ,  9621.25],
       [ 7217.6 ,  9520.9 ],
       [ 7750.05, 10077.1 ],
       [ 7700.3 ,  9917.9 ],
       [ 7978.2 ,  9788.6 ],
       [ 8211.25, 10335.3 ],
       [ 8599.1 , 10226.55],
       [ 9729.55, 10530.7 ],
       [ 9509.7 , 11027.7 ],
       [ 8850.95, 10492.85],
       [ 8861.1 , 10113.7 ],
       [ 8814.95, 10739.35],
       [ 8537.2 , 10736.15],
       [ 8825.6 , 10714.3 ],
       [ 9520.55, 11356.5 ],
       [ 9096.4 , 11680.5 ],
       [ 7347.95, 10930.45],
       [ 6616.4 , 10386.6 ],
       [ 7661.6 , 10876.75],
       [ 7465.5 , 10862.55],
       [ 6641.15, 10830.95],
       [ 6829.7 , 10792.5 ],
       [ 6672.55, 11623.9 ],
       [ 6666.4 , 11748.15],
       [ 6543.75, 11148.2 ]])
```

In [29]:
```
month=pd.DataFrame(month)
```

In [30]:
```
month.coulmns=['maruti','nifty']
month.index=prices['month'].unique()
```

In [31]:
```
month
```

Out[31]:

	0	1
May-2017	7211.00	9621.25
Jun-2017	7217.60	9520.90
Jul-2017	7750.05	10077.10
Aug-2017	7700.30	9917.90
Sep-2017	7978.20	9788.60
Oct-2017	8211.25	10335.30
Nov-2017	8599.10	10226.55
Dec-2017	9729.55	10530.70
Jan-2018	9509.70	11027.70
Feb-2018	8850.95	10492.85
Mar-2018	8861.10	10113.70
Apr-2018	8814.95	10739.35
May-2018	8537.20	10736.15
Jun-2018	8825.60	10714.30
Jul-2018	9520.55	11356.50
Aug-2018	9096.40	11680.50
Sep-2018	7347.95	10930.45
Oct-2018	6616.40	10386.60
Nov-2018	7661.60	10876.75
Dec-2018	7465.50	10862.55
Jan-2019	6641.15	10830.95
Feb-2019	6829.70	10792.50
Mar-2019	6672.55	11623.90
Apr-2019	6666.40	11748.15
May-2019	6543.75	11148.20

In [32]:
```
returnsm = month.pct_change()
returnsm = returnsm.dropna(axis=0)
Xm = returns['nifty']
ym = returns['maruti']
X1m = sm.add_constant(Xm)
model = sm.OLS(ym, X1m)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 maruti   R-squared:                       0.151
Model:                            OLS   Adj. R-squared:                  0.136
Method:                 Least Squares   F-statistic:                     10.28
Date:                Tue, 02 Jun 2020   Prob (F-statistic):            0.00219
Time:                        15:41:09   Log-Likelihood:                 179.65
No. Observations:                  60   AIC:                            -355.3
Df Residuals:                      58   BIC:                            -351.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0016      0.002     -1.006      0.319      -0.005       0.002
nifty          0.7523      0.235      3.206      0.002       0.283       1.222
==============================================================================
Omnibus:                        1.985   Durbin-Watson:                   1.401
Prob(Omnibus):                  0.371   Jarque-Bera (JB):                1.208
```