Module 3

-By Suprateek Halsana

In [268]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm
from statsmodels import regression
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,accuracy_score,mean_squared_error
import warnings
warnings.filterwarnings('ignore')
sns.set(style='darkgrid')
```

Problem [3.1]

In [269]:
```
data=pd.read_csv(r"C:\Users\Suprateek Halsana\Documents\Python Scripts\Aspiring Mind Internship\Prereq
uisites\Gold\GOLD.csv")
```

In [270]:
```
data['Date']=pd.to_datetime(data['Date'])
data
```

Out[270]:

	Date	Price	Open	High	Low	Vol.	Change %	Pred	new
0	2017-05-04	28060	28400	28482	28025	0.08K	-1.79%	738.0	117.570740
1	2017-05-05	28184	28136	28382	28135	0.06K	0.44%	-146.0	295.430176
2	2017-05-08	28119	28145	28255	28097	7.85K	-0.23%	30.0	132.123714
3	2017-05-09	27981	28125	28192	27947	10.10K	-0.49%	357.0	101.298064
4	2017-05-10	28007	28060	28146	27981	9.28K	0.09%	124.0	112.153318
...	...	...	...	...	...	...	...	...	...
507	2019-04-26	31868	31851	31934	31705	9.67K	0.08%	NaN	247.177322
508	2019-04-30	31625	31800	31824	31597	6.44K	-0.76%	NaN	52.201158
509	2019-05-01	31563	31604	31657	31503	1.55K	-0.20%	NaN	113.293305
510	2019-05-02	31203	31420	31425	31160	0.48K	-1.14%	NaN	48.365693
511	2019-05-03	31341	31250	31500	31163	0.08K	0.44%	NaN	429.924911

512 rows × 9 columns

Splitted the data into the Data With Null & Data Without Null

In [271]:
```
d1=data.copy()
dnull=d1[d1['Pred'].isnull()==True]
dnotnull=d1[d1['Pred'].isnull()==False]
```

Fitted the Model with Data Without Null

In [272]:
```
m1=LinearRegression()
x=dnotnull[dnotnull.columns[1:5]]
y=dnotnull['Pred']
xtrain,xtest,ytrain,ytest1=train_test_split(x,y,test_size=0.25)
m1.fit(xtrain,ytrain)
ypred1=m1.predict(xtest)
```

In [273]:
```
# The Dataframe with Null values of Pred
dnull
```

Out[273]:

	Date	Price	Open	High	Low	Vol.	Change %	Pred	new
411	2018-12-11	31812	31850	31850	31618	10.53K	-0.26%	NaN	195.423493
412	2018-12-12	31626	31749	31749	31582	7.57K	-0.58%	NaN	44.232664
413	2018-12-13	31414	31550	31600	31337	8.43K	-0.67%	NaN	127.646233
414	2018-12-14	31437	31440	31514	31384	6.75K	0.07%	NaN	127.219539
415	2018-12-17	31501	31369	31530	31291	5.97K	0.20%	NaN	372.603976
...	...	...	...	...	...	...	...	...	...
507	2019-04-26	31868	31851	31934	31705	9.67K	0.08%	NaN	247.177322
508	2019-04-30	31625	31800	31824	31597	6.44K	-0.76%	NaN	52.201158
509	2019-05-01	31563	31604	31657	31503	1.55K	-0.20%	NaN	113.293305
510	2019-05-02	31203	31420	31425	31160	0.48K	-1.14%	NaN	48.365693
511	2019-05-03	31341	31250	31500	31163	0.08K	0.44%	NaN	429.924911

101 rows × 9 columns

Predicted and filled the Null Values in dnull

In [274]:
```
dnull['Pred']=m1.predict(dnull[dnull.columns[1:5]])
# ****************************************************
# The Dataframe with null values are filled with Predicted Values
# ****************************************************
dnull
```

Out[274]:

	Date	Price	Open	High	Low	Vol.	Change %	Pred	new
411	2018-12-11	31812	31850	31850	31618	10.53K	-0.26%	852.0	195.423493
412	2018-12-12	31626	31749	31749	31582	7.57K	-0.58%	422.0	44.232664
413	2018-12-13	31414	31550	31600	31337	8.43K	-0.67%	530.0	127.646233
414	2018-12-14	31437	31440	31514	31384	6.75K	0.07%	144.0	127.219539
415	2018-12-17	31501	31369	31530	31291	5.97K	0.20%	415.0	372.603976
...	...	...	...	...	...	...	...	...	...
507	2019-04-26	31868	31851	31934	31705	9.67K	0.08%	535.0	247.177322
508	2019-04-30	31625	31800	31824	31597	6.44K	-0.76%	438.0	52.201158
509	2019-05-01	31563	31604	31657	31503	1.55K	-0.20%	269.0	113.293305
510	2019-05-02	31203	31420	31425	31160	0.48K	-1.14%	601.0	48.365693
511	2019-05-03	31341	31250	31500	31163	0.08K	0.44%	280.0	429.924911

101 rows × 9 columns

In [275]:
```
dnull.isnull().sum()
```

Out[275]:
```
Date        0
Price       0
Open        0
High        0
Low         0
Vol.        0
Change %    0
Pred        0
new         0
dtype: int64
```

In [276]:
```
# Merged Data
```

In [277]:
```
data=pd.concat([dnotnull,dnull])
```

Fitting the model for 'new' Column

In [278]:
```
m2=LinearRegression()
x,y=data[data.columns[1:5]],data['new']
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25)
m2.fit(xtrain,ytrain)

print('Coeff    :',m2.coef_)
ypred=m2.predict(xtest)
print('Accuracy :',r2_score(ytest,ypred))
```

```
Coeff    : [ 1.01134237 -1.00207316  1.0059891  -1.01533831]
Accuracy : 0.9999687049476801
```

In [279]:
```
print('Mean Square Error of new  :',mean_squared_error(ytest,ypred))
print('Mean Square Error of Pred   :',mean_squared_error(ytest1,ypred1))
```

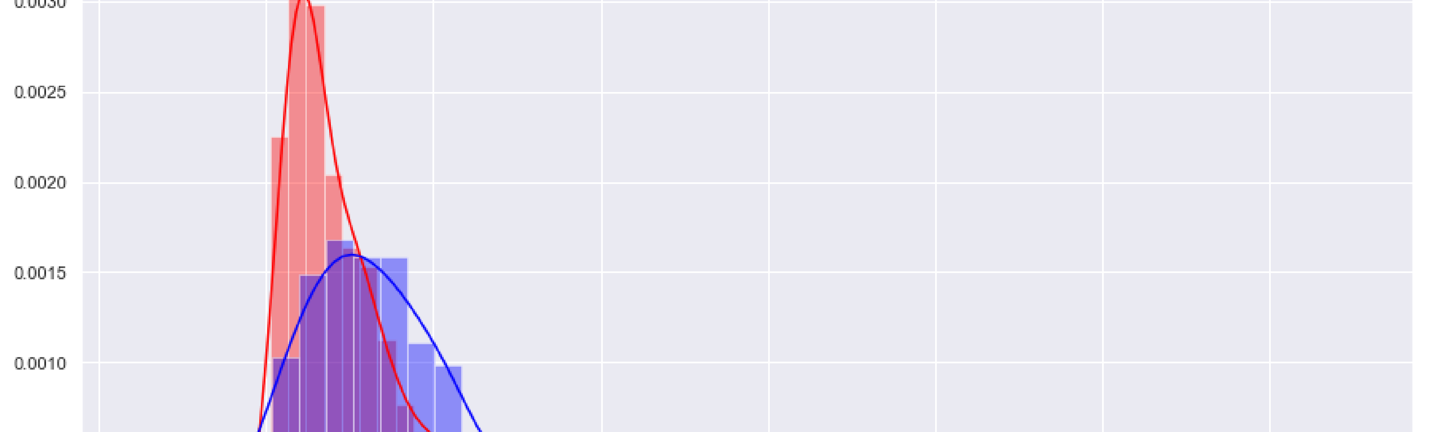
```
Mean Square Error of new  : 1.0504763248049744
Mean Square Error of Pred   : 1.1632227364026952e-23
```

In [280]:
```
lx=list(data.columns[1:5])
lnewy=m2.coef_
lpredy=m1.coef_
```

In [281]:
```
plt.figure(figsize=(18,5))
plt.plot(lx,lnewy,label='new',marker='o')
plt.plot(lx,lpredy,label='pred',marker='*')
plt.title('Slope Graph',fontsize=15)
plt.legend()
plt.show()
```

In [282]:
```
f,(ax1,ax2) = plt.subplots(1,2,figsize=(18,7))
sns.distplot(data['new'],label='New',ax=ax1,color='red')
sns.distplot(data['Pred'],label='Pred',ax=ax2,color='blue')
plt.show()
```

In [283]:
```
plt.figure(figsize=(15,8))
sns.distplot(data['new'],label='New',color='red')
sns.distplot(data['Pred'],label='Pred',color='blue')
plt.legend()
plt.xlabel('')
plt.show()
```



The Above Resultant Distplot of Pred shows a Normal Distributed Graph Depicting the Linearity , However the 'New' Shows the high elevation and it doesn't show a distributed graph therefore shows that :

'Pred' is Linear Function (As Described by the Normal Distributed Graph)

'New' is Polynomial Function (As Described by the Non Distributed Graph)

Problem [3.2]

CAPM : Capital Asset Pricing Model represents and explains the Relationship between the Expected Returns & the Risk

Beta : It Represents the Sensitivity of the Returns relative to the Benchmark Index . It Even Represents the Volatility of the Stock also as its very crucial for stock investors to know the Volatility. It may be -ve as well as +ve , generally its value doesn't goes more than 4 even in rare cases. It actually is the Ratio of the Stock Returns to the Market Returns.

Beta Calculation Using OLS Regression

In [284]:
```
# nifty index
lalpathlab=pd.read_csv(r'C:\Users\Suprateek Halsana\Documents\Python Scripts\Aspiring Mind Internship
\Week3.csv')

#Lalpathlab Stocks
nifty=pd.read_csv(r'C:\Users\Suprateek Halsana\Documents\Python Scripts\Aspiring Mind Internship\Prere
quisites\Nifty50\Nifty50.csv')
```

Beta Calculation for Past 3 Months

In [285]:
```
x_3months=lalpathlab[(lalpathlab.shape[0])-91:]['Close Price'].pct_change()[1:]
y_3months=nifty[(nifty.shape[0])-91:]['Close'].pct_change()[1:]
```

In [286]:
```
xnew = sm.add_constant(x_3months)
model = regression.linear_model.OLS(y_3months,xnew).fit()
```

In [287]:
```
model.summary()
```

Out[287]:

OLS Regression Results

Dep. Variable:	Close	**R-squared:**	0.040
Model:	OLS	**Adj. R-squared:**	0.029
Method:	Least Squares	**F-statistic:**	3.635
Date:	Wed, 23 Sep 2020	**Prob (F-statistic):**	0.0599
Time:	23:51:51	**Log-Likelihood:**	322.93
No. Observations:	90	**AIC:**	-641.9
Df Residuals:	88	**BIC:**	-636.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	0.001	0.343	0.733	-0.001	0.002
Close Price	0.0710	0.037	1.906	0.060	-0.003	0.145

Omnibus:	3.616	**Durbin-Watson:**	1.697
Prob(Omnibus):	0.164	**Jarque-Bera (JB):**	2.155
Skew:	0.133	**Prob(JB):**	0.340
Kurtosis:	2.290	**Cond. No.**	52.3

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

In [288]:
```
print('Beta  :',model.params[1])
print('Alpha :',model.params[0])
```

```
Beta  : 0.07104520931392258
Alpha : 0.00024461869893492333
```

In [289]:
```
x2=np.linspace(x.min(),x.max(),100)
yhat=x2*model.params[1]+model.params[0]
```

Monthly Beta calculation

In [290]:
```
d=pd.DataFrame()
d['Month']=lalpathlab['Month']
d['Close_Lal']=lalpathlab['Close Price']
d['Close_Nifty']=nifty['Close']
```

In [291]:
```
g=d.groupby('Month')
```

In [292]:
```
g.first()
```

Out[292]:

Month	Close_Lal	Close_Nifty
1	884.65	10435.55
2	910.45	11016.90
3	910.50	10458.35
4	871.65	10211.80
5	900.60	9445.40
6	900.60	9616.10
7	775.45	9615.00
8	824.10	10114.65
9	810.25	9974.40
10	790.15	9859.50
11	783.35	10440.50
12	889.25	10121.80

In [293]:
```
for n,group in g:
    x,y=group['Close_Lal'].pct_change()[1:],group['Close_Nifty'].pct_change()[1:]
    xnew = sm.add_constant(x)
    model = regression.linear_model.OLS(y,xnew).fit()
    print('Beta for Month',n,' is : %0.4f'%model.params[1])
```

```
Beta for Month 1  is : 0.0244
Beta for Month 2  is : 0.2675
Beta for Month 3  is : 0.4091
Beta for Month 4  is : 0.3359
Beta for Month 5  is : 0.1773
Beta for Month 6  is : 0.5230
Beta for Month 7  is : 0.2937
Beta for Month 8  is : 0.6456
Beta for Month 9  is : 0.5794
Beta for Month 10  is : 0.2736
Beta for Month 11  is : 0.0146
Beta for Month 12  is : 0.0752
```

In [294]:
```
# ****************************************** Brief Conclusion ******************************************
*********
```

The Above Beta Values actually represent the Sensitivity of the Stock Returns with Bench Mark Index or the Other Way Describes the Volatility of the Stocks and the Risk in Investment

The 3 months Beta value comes out to be 0.0710 ie.. >0 and <1 and shows a safe Beta Value and Also Less Volatility of the Stock. The Beta of 0.07 represents that on 10% market rise the Stock would be gaining only 0.7 % gain and on loss of 10% will have 0.7 % loss only. Hence, Stock Investors will be on a Safer Side

Even all the Month Beta Values Range the Same between 0 and 1 hence are Safe...and Less Risky.

If the Beta would have gone below 0 then it could have been Riskier as the Gain of market would result in a loss in the Stocks... Like Gold Investment is been considered in most cases of the category whose beta is negative . These Investment only gain Profit during the Loss of Market

If the Beta Would have been greator than 1 then it could have been More Volatile and Riskier as for a beta of 2 the 10% loss wouldresult in 20% loss of stock Which is really a great Loss for the investors . Hence the Greator the Beta value more is the Risk.

In [ ]: