Stock_Analysis_Returns

September 29, 2021

1 Stock Analysis Returns

[1]:
```
# Library
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

import fix_yahoo_finance as yf
yf.pdr_override()
```

[2]:
```
start = '2016-01-01'
end = '2019-01-01'

market = 'SPY'
symbol1 = 'AAPL'
symbol2 = 'MSFT'
symbol3 = 'AMD'
symbol4 = 'INTC'
bench = yf.download(market, start=start, end=end)
stock1 = yf.download(symbol1, start=start, end=end)
stock2 = yf.download(symbol2, start=start, end=end)
stock3 = yf.download(symbol3, start=start, end=end)
stock4 = yf.download(symbol4, start=start, end=end)
```

```
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
```

1.1 Calculate Daily Gains

[3]:
```
#Daily gain for the stock
stock1["Gain"]=(stock1["Adj Close"].pct_change())*100
stock2["Gain"]=(stock2["Adj Close"].pct_change())*100
stock3["Gain"]=(stock3["Adj Close"].pct_change())*100
stock4["Gain"]=(stock4["Adj Close"].pct_change())*100
```

1.2 Calculate the Mean and Variances of Daily Gains

[4]:
```
print('Stock '+ symbol1 + ' Mean:', stock1["Gain"].mean())
print('Stock '+ symbol1 + ' Variances:', stock1["Gain"].var())
```

```
Stock AAPL Mean: 0.07176158262397078
Stock AAPL Variances: 2.2250639945491892
```

[5]:
```
print('Stock '+ symbol2 + ' Mean:', stock2["Gain"].mean())
print('Stock '+ symbol2 + ' Variances:', stock2["Gain"].var())
```

```
Stock MSFT Mean: 0.10088954069328435
Stock MSFT Variances: 2.0235968045099493
```

[6]:
```
print('Stock '+ symbol3 + ' Mean:', stock3["Gain"].mean())
print('Stock '+ symbol3 + ' Variances:', stock3["Gain"].var())
```

```
Stock AMD Mean: 0.34459086934253985
Stock AMD Variances: 19.3330778674231
```

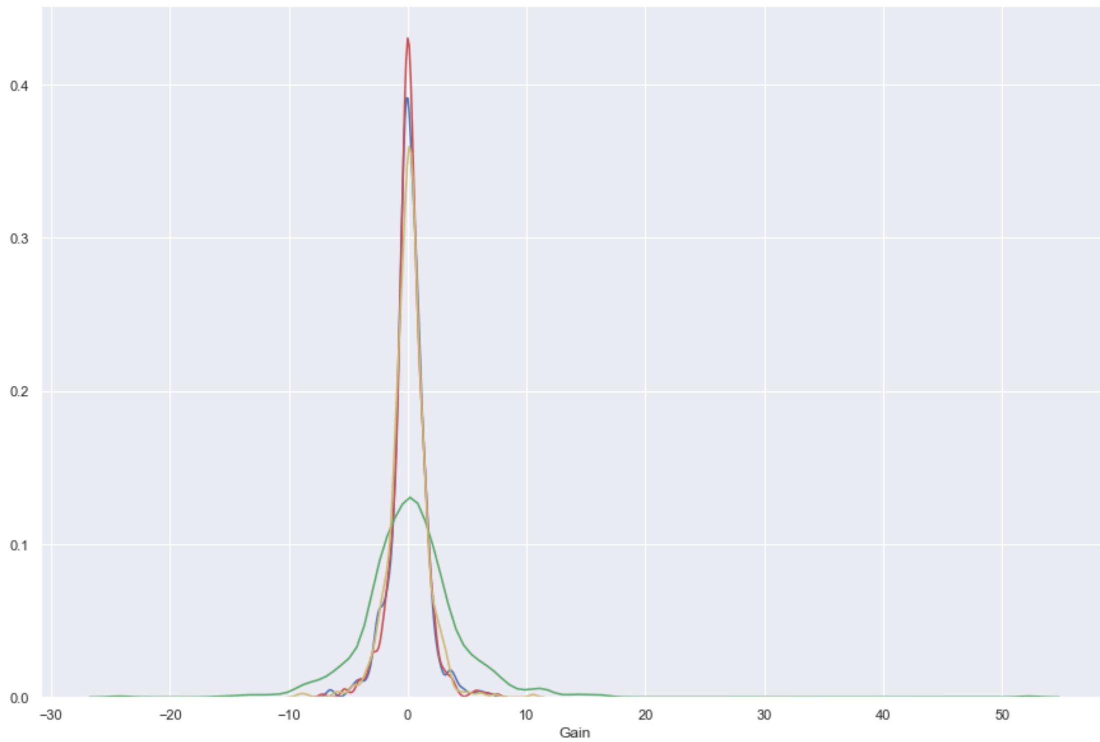
[7]:
```
print('Stock '+ symbol4 + ' Mean:', stock4["Gain"].mean())
print('Stock '+ symbol4 + ' Variances:', stock4["Gain"].var())
```

```
Stock INTC Mean: 0.06706428724452075
Stock INTC Variances: 2.5783495558348006
```

1.3 Highest volatality and draw the histogram distribution of daily returns for all the stock

[8]:
```
sns.set(rc={"figure.figsize": (15, 10)});
sns.distplot(stock1['Gain'], hist = False, color = 'b' )
sns.distplot(stock2['Gain'], hist = False, color = 'r' )
sns.distplot(stock3['Gain'], hist = False, color = 'g' )
sns.distplot(stock4['Gain'], hist = False, color = 'y' )
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1eaa2db6860>
```



1.4 Correlation

[9]:
```
All_Stocks = pd.
↪concat([stock1['Gain'],stock2['Gain'],stock3['Gain'],stock4['Gain']], axis=1)
```

[10]:
```
names = ['AAPL', 'MSFT', 'AMD', 'INTC']
All_Stocks.columns = names
```

[11]:
```
All_Stocks = All_Stocks.dropna()
All_Stocks
```

[11]:
```
                AAPL      MSFT       AMD      INTC
Date
2016-01-05 -2.505943  0.456213 -0.722022 -0.470730
2016-01-06 -1.956978 -1.816536 -8.727273 -2.216973
2016-01-07 -4.220433 -3.478261 -9.163347 -3.748490
2016-01-08  0.528771  0.306692 -6.140351 -1.036431
2016-01-11  1.619202 -0.057342  9.345794  1.745483
2016-01-12  1.451340  0.917797  2.136752  1.933870
2016-01-13 -2.571009 -2.159906 -5.857741 -2.356170
2016-01-14  2.187060  2.846633 -1.777778  2.601050
2016-01-15 -2.401509 -3.991714 -8.144796 -9.102010
2016-01-19 -0.483894 -0.843303 -3.940887  0.134406
```

2016-01-20	0.134499	0.454896	-7.692308	-0.704701
2016-01-21	-0.506262	-0.610359	16.111111	0.236560
2016-01-22	5.316732	3.585598	-3.349282	0.910327
2016-01-25	-1.952270	-0.956216	4.950495	-1.102578
2016-01-26	0.553084	0.733724	-2.358491	1.148663
2016-01-27	-6.570673	-1.820953	2.898551	-0.434224
2016-01-28	0.717201	1.639971	-2.347418	0.536744
2016-01-29	3.454119	5.820207	5.769231	3.503505
2016-02-01	-0.934834	-0.689786	-2.727273	-0.644743
2016-02-02	-2.022214	-3.125569	-7.009346	-3.309541
2016-02-03	1.979255	-1.584907	4.020101	-0.677044
2016-02-04	0.803508	-0.306745	0.966184	1.465569
2016-02-05	-2.670808	-3.538454	-5.263158	-2.452125
2016-02-08	1.052986	-1.495229	-2.525253	-0.757601
2016-02-09	-0.021062	-0.263084	-1.554404	-0.034677
2016-02-10	-0.757979	0.872556	-3.157895	-2.013203
2016-02-11	-0.604641	-0.040237	1.086957	-0.035417
2016-02-12	0.309496	1.630105	-1.612903	1.488297
2016-02-16	2.819466	1.894691	0.000000	0.488853
2016-02-17	1.531444	2.603262	3.825137	2.397486
…	…	…	…	…
2018-11-15	2.467874	2.200622	3.267665	2.166070
2018-11-16	1.107568	0.941465	-3.862262	1.496568
2018-11-19	-3.963206	-3.389040	-7.502415	-1.699773
2018-11-20	-4.777803	-2.781504	0.523276	-1.270839
2018-11-21	-0.112997	1.376466	-2.498694	-0.759656
2018-11-23	-2.539880	-0.038792	3.470363	-1.041884
2018-11-26	1.352367	3.298728	3.611976	1.955310
2018-11-27	-0.217613	0.629283	4.830672	1.306635
2018-11-28	3.845272	3.714769	1.377677	1.643440
2018-11-29	-0.768210	-0.836933	0.421743	-2.374128
2018-11-30	-0.540243	0.635264	-0.606631	3.375267
2018-12-03	3.494240	1.082147	11.314555	1.662948
2018-12-04	-4.398882	-3.184935	-10.923653	-4.747661
2018-12-06	-1.114935	0.617403	0.852263	1.298424
2018-12-07	-3.565713	-4.002209	-8.638498	-4.403543
2018-12-10	0.658798	2.642624	2.723541	2.097742
2018-12-11	-0.571937	0.929460	-0.050025	0.360091
2018-12-12	0.278719	0.451243	2.502503	0.949766
2018-12-13	1.094021	0.339198	-3.027339	0.961743
2018-12-14	-3.199760	-3.124715	0.201405	-0.890452
2018-12-17	-0.930627	-2.961430	-5.376884	-1.629747
2018-12-18	1.299252	1.049672	3.558152	1.401861
2018-12-19	-3.119167	-0.269313	-6.871795	-4.545456
2018-12-20	-2.523467	-2.102417	-1.211448	-0.065823
2018-12-21	-3.889559	-3.231207	-5.629883	-1.537126
2018-12-24	-2.587412	-4.173882	-1.653869	-2.787690

```
2018-12-26  7.042159  6.830979   7.507508  5.964674
2018-12-27 -0.648980  0.616550  -2.290503  0.368052
2018-12-28  0.051228 -0.780784   1.886792  0.841243
2018-12-31  0.966536  1.175414   3.591465  0.385032

[753 rows x 4 columns]
```

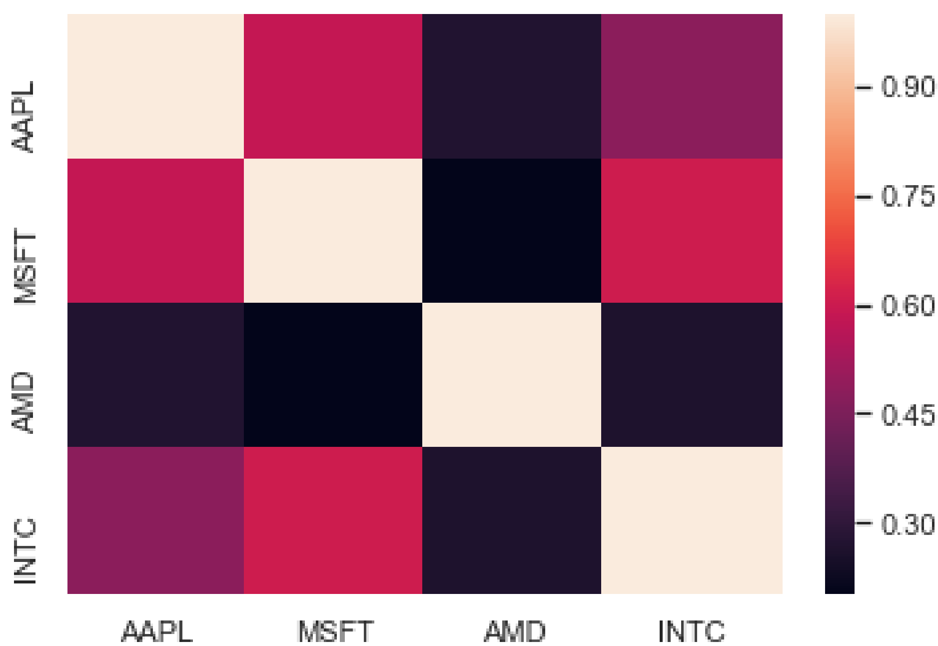
[12]:
```python
All_Stocks.corr()
```

[12]:

	AAPL	MSFT	AMD	INTC
AAPL	1.000000	0.587342	0.270291	0.479345
MSFT	0.587342	1.000000	0.200654	0.603708
AMD	0.270291	0.200654	1.000000	0.264619
INTC	0.479345	0.603708	0.264619	1.000000

[13]:
```python
#Heat map
sns.set(rc={"figure.figsize": (6, 4)});
sns.heatmap( All_Stocks.corr())
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1eaa2ec3b70>



1.4.1 Monthly Returns

[14]:
```
Stock1_Monthly = stock1.asfreq('M').ffill()
Stock2_Monthly = stock2.asfreq('M').ffill()
Stock3_Monthly = stock3.asfreq('M').ffill()
Stock4_Monthly = stock4.asfreq('M').ffill()
```

[15]:
```
print('Monthly Returns')
print('Stock '+ symbol1 + ' Mean:', Stock1_Monthly["Gain"].mean())
print('Stock '+ symbol1 + ' Variances:', Stock1_Monthly["Gain"].var())
```

```
Monthly Returns
Stock AAPL Mean: 0.2862637568937698
Stock AAPL Variances: 0.7227331190053209
```

[16]:
```
print('Monthly Returns')
print('Stock '+ symbol2 + ' Mean:', Stock2_Monthly["Gain"].mean())
print('Stock '+ symbol2 + ' Variances:', Stock2_Monthly["Gain"].var())
```

```
Monthly Returns
Stock MSFT Mean: 0.21547252764849448
Stock MSFT Variances: 1.1411972390607814
```

[17]:
```
print('Monthly Returns')
print('Stock '+ symbol3 + ' Mean:', Stock3_Monthly["Gain"].mean())
print('Stock '+ symbol3 + ' Variances:', Stock3_Monthly["Gain"].var())
```

```
Monthly Returns
Stock AMD Mean: 0.32643003321886305
Stock AMD Variances: 7.488704333660259
```

[18]:
```
print('Monthly Returns')
print('Stock '+ symbol4 + ' Mean:', Stock4_Monthly["Gain"].mean())
print('Stock '+ symbol4 + ' Variances:', Stock4_Monthly["Gain"].var())
```

```
Monthly Returns
Stock INTC Mean: 0.1408411967334154
Stock INTC Variances: 2.1223156784242296
```

1.5 Monthly Returns with Box Plot

[19]:
```
Stock1=np.array(Stock1_Monthly["Gain"])
Stock1= Stock1[~np.isnan(Stock1_Monthly["Gain"])]

Stock2 = np.array(Stock2_Monthly["Gain"])
Stock2=Stock2[~np.isnan(Stock2_Monthly["Gain"])]

Stock3 = np.array(Stock3_Monthly["Gain"])
```

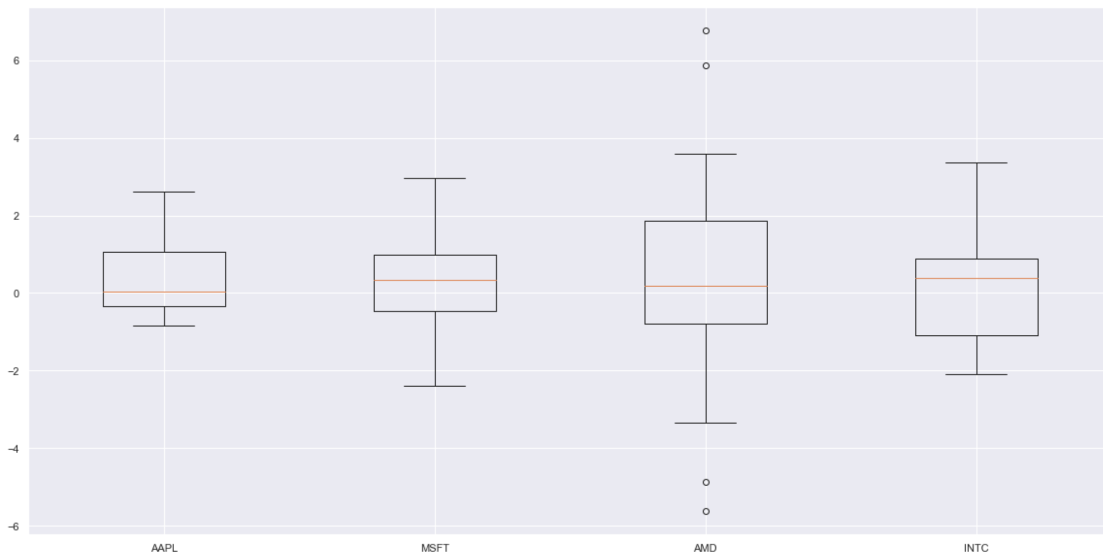
```
Stock3=Stock3[~np.isnan(Stock3_Monthly["Gain"])]

Stock4 = np.array(Stock4_Monthly["Gain"])
Stock4=Stock4[~np.isnan(Stock4_Monthly["Gain"])]

AllStocks =[Stock1,Stock2,Stock3,Stock4]
```

[20]:
```
fig = plt.figure(1, figsize=(20, 10))
ax = fig.add_subplot(111)
bp = ax.boxplot(AllStocks)
ax.set_xticklabels([symbol1, symbol2, symbol3, symbol4])
```

[20]: `[Text(0,0,'AAPL'), Text(0,0,'MSFT'), Text(0,0,'AMD'), Text(0,0,'INTC')]`



1.6 Stock with highest probability gains with 2% or more

[21]:
```
#Probability of Stock1
stock1_p = 1-stats.norm.cdf( 0.02,
             loc=Stock1_Monthly["Gain"].mean(),
             scale=Stock1_Monthly["Gain"].std())

print(symbol1 + " probability of gains:", round(stock1_p, 2))
```

```
AAPL probability of gains: 0.62
```

[22]:
```
stock2_p = 1-stats.norm.cdf( 0.02,
             loc=Stock2_Monthly["Gain"].mean(),
             scale=Stock2_Monthly["Gain"].std())
```

```python
print(symbol2 + " probability of gains:", round(stock2_p, 2))
```

```
MSFT probability of gains: 0.57
```

[23]:
```python
stock3_p = 1-stats.norm.cdf( 0.02,
             loc=Stock3_Monthly["Gain"].mean(),
             scale=Stock3_Monthly["Gain"].std())

print(symbol3 + " probability of gains:", round(stock3_p, 2))
```

```
AMD probability of gains: 0.54
```

[24]:
```python
stock4_p = 1-stats.norm.cdf( 0.02,
             loc=Stock4_Monthly["Gain"].mean(),
             scale=Stock4_Monthly["Gain"].std())

print(symbol4 + " probability of gains:", round(stock4_p, 2))
```

```
INTC probability of gains: 0.53
```

1.7 Stock with highest probability of loss with 2% or more

[25]:
```python
#Probability of Stock1
stock1_l = stats.norm.cdf(-0.02,
             loc=Stock1_Monthly["Gain"].mean(),
             scale=Stock1_Monthly["Gain"].std())

print(symbol1 + " probability of loss:", round(stock1_l, 2))
```

```
AAPL probability of loss: 0.36
```

[26]:
```python
stock2_l = stats.norm.cdf(-0.02,
             loc=Stock2_Monthly["Gain"].mean(),
             scale=Stock2_Monthly["Gain"].std())

print(symbol2 + " probability of loss:", round(stock2_l, 2))
```

```
MSFT probability of loss: 0.41
```

[27]:
```python
stock3_l = stats.norm.cdf(-0.02,
             loc=Stock3_Monthly["Gain"].mean(),
             scale=Stock3_Monthly["Gain"].std())

print(symbol3 + " probability of loss:", round(stock3_l, 2))
```

```
AMD probability of loss: 0.45
```

```
[28]: stock4_l = stats.norm.cdf(-0.02,
              loc=Stock4_Monthly["Gain"].mean(),
              scale=Stock4_Monthly["Gain"].std())

print(symbol4 + " probability of loss:", round(stock4_l, 2))
```

```
INTC probability of loss: 0.46
```

1.8 Portfolio Analysis

```
[29]: x=np.array([Stock1_Monthly["Gain"].mean(),Stock2_Monthly["Gain"].
 ↪mean(),Stock3_Monthly["Gain"].mean(),Stock4_Monthly["Gain"].mean()])
print(x)
```

```
[ 0.28626376  0.21547253  0.32643003  0.1408412 ]
```

```
[30]: #Weights of the stocks is 0.25 which is added up to 1
weights = np.array([0.25,0.25,0.25,0.25])
exp_val=np.sum(x*weights)

print("Expected Value is ",round(exp_val,4))
print("\n")
#Calculate Covariance matrix
y = np.vstack([Stock1,Stock2,Stock3,Stock4])

cov = np.cov(y)
print("Below is covariance matrix")
print("\n")
print(cov)
```

```
Expected Value is  0.2423


Below is covariance matrix


[[ 0.72273312  0.35454055  0.72956139  0.3469801 ]
 [ 0.35454055  1.14119724  1.49464282  0.49112976]
 [ 0.72956139  1.49464282  7.48870433  0.32942418]
 [ 0.3469801   0.49112976  0.32942418  2.12231568]]
```

```
[31]: #Calcualte the variance of monthly return of portfolio
covar=np.dot(weights.T,np.dot(cov,weights))
print("Variance of portfolio is ",round(covar,4))
```

```
Variance of portfolio is  1.1855
```

[32]:
```
#Calculate the probability
1-stats.norm.cdf(0.005,
                 loc=exp_val,
                 scale=covar)
```

[32]: 0.57931183676482079

[33]:
```
# Create 25 Iteration of weights
# Generate a random number

number=range(1,26)
```

[34]:
```
# Function to calculate expected value of portfolio and variance
def calculate(weights, meanReturns, covMatrix):

     portReturn = np.sum(weights*meanReturns)
     portVar = (np.dot(weights.T, np.dot(covMatrix, weights)))
     return portReturn, portVar
```

[35]:
```
# Generate weights in random that sum to 1
import random
random.seed(4)
d=[]
for i in number:
    weights = np.random.random(4)
    weights /= weights.sum()
    print("Set of random weight for Iterartion-->",i,"is", weights)
    pret, pvar = calculate(weights, x, cov)

    d.append((weights[0],weights[1],weights[2],weights[3],pret,pvar))
    df=pd.
 ↪DataFrame(d,columns=('Stock1_weight','Stock2_weight','Stock3_weight','Stock4_weight','mean_
    print("Mean monthly return for iteration-->",i,"is",pret)
    print("Variance of monthly return for iteration-->",i,"is",pvar)
    print("\n")
```

```
Set of random weight for Iterartion--> 1 is [ 0.3848186   0.36151152  0.16429325
0.08937663]
Mean monthly return for iteration--> 1 is 0.25427358196
Variance of monthly return for iteration--> 1 is 0.908980794775


Set of random weight for Iterartion--> 2 is [ 0.00454593  0.41422227  0.29738145
0.28385034]
Mean monthly return for iteration--> 2 is 0.227606915146
Variance of monthly return for iteration--> 2 is 1.57262314596
```

```
Set of random weight for Iterartion--> 3 is [ 0.24472972  0.41455793  0.30923541
0.03147695]
Mean monthly return for iteration--> 3 is 0.264760068295
Variance of monthly return for iteration--> 3 is 1.54778775312


Set of random weight for Iterartion--> 4 is [ 0.37906084  0.01872867  0.28758687
0.31462362]
Mean monthly return for iteration--> 4 is 0.250735852198
Variance of monthly return for iteration--> 4 is 1.26205610736


Set of random weight for Iterartion--> 5 is [ 0.2940364   0.39046648  0.09553035
0.21996676]
Mean monthly return for iteration--> 5 is 0.230471123061
Variance of monthly return for iteration--> 5 is 0.784504283061


Set of random weight for Iterartion--> 6 is [ 0.38135555  0.15178525  0.26866442
0.19819477]
Mean monthly return for iteration--> 6 is 0.257487950671
Variance of monthly return for iteration--> 6 is 1.18483172598


Set of random weight for Iterartion--> 7 is [ 0.11378785  0.07041875  0.45331977
0.36247363]
Mean monthly return for iteration--> 7 is 0.246775051304
Variance of monthly return for iteration--> 7 is 2.17110695256


Set of random weight for Iterartion--> 8 is [ 0.05527481  0.35307643  0.4519602
0.13968856]
Mean monthly return for iteration--> 8 is 0.259108732959
Variance of monthly return for iteration--> 8 is 2.33829820835


Set of random weight for Iterartion--> 9 is [ 0.47245706  0.13779025  0.31125011
0.07850258]
Mean monthly return for iteration--> 9 is 0.277595127595
Variance of monthly return for iteration--> 9 is 1.36294418122


Set of random weight for Iterartion--> 10 is [ 0.0574168   0.33496984
0.36052021  0.24709314]
Mean monthly return for iteration--> 10 is 0.241098666888
Variance of monthly return for iteration--> 10 is 1.78802658816
```

```
Set of random weight for Iterartion--> 11 is [ 0.27403096  0.23236197
0.35606134  0.13754573]
Mean monthly return for iteration--> 11 is 0.264113973566
Variance of monthly return for iteration--> 11 is 1.63011043003


Set of random weight for Iterartion--> 12 is [ 0.13314844  0.34550485
0.39105207  0.13029464]
Mean monthly return for iteration--> 12 is 0.25856436944
Variance of monthly return for iteration--> 12 is 1.93256216528


Set of random weight for Iterartion--> 13 is [ 0.35722622  0.23661181
0.28861182  0.11755015]
Mean monthly return for iteration--> 13 is 0.264011734259
Variance of monthly return for iteration--> 13 is 1.30254780369


Set of random weight for Iterartion--> 14 is [ 0.1982255   0.36107538
0.41913181  0.02156731]
Mean monthly return for iteration--> 14 is 0.274401377137
Variance of monthly return for iteration--> 14 is 2.13466617511


Set of random weight for Iterartion--> 15 is [ 0.4986417   0.04223101
0.45238715  0.00674014]
Mean monthly return for iteration--> 15 is 0.300464710381
Variance of monthly return for iteration--> 15 is 2.12023889388


Set of random weight for Iterartion--> 16 is [ 0.09142834  0.46559868
0.13824328  0.3047297 ]
Mean monthly return for iteration--> 16 is 0.214541598557
Variance of monthly return for iteration--> 16 is 1.02111885965


Set of random weight for Iterartion--> 17 is [ 0.15133269  0.05777029
0.32080811  0.47008891]
Mean monthly return for iteration--> 17 is 0.22669826147
Variance of monthly return for iteration--> 17 is 1.567920736


Set of random weight for Iterartion--> 18 is [ 0.43962751  0.20354291
0.15370873  0.20312084]
Mean monthly return for iteration--> 18 is 0.248490258657
Variance of monthly return for iteration--> 18 is 0.830181000318
```

```
Set of random weight for Iterartion--> 19 is [ 0.1797379   0.43788708
0.37616091  0.00621411]
Mean monthly return for iteration--> 19 is 0.269470502891
Variance of monthly return for iteration--> 19 is 1.95370987587


Set of random weight for Iterartion--> 20 is [ 0.2134741   0.72133744
0.01021475  0.05497372]
Mean monthly return for iteration--> 20 is 0.227615262863
Variance of monthly return for iteration--> 20 is 0.815789418793


Set of random weight for Iterartion--> 21 is [ 0.31605233  0.1171546
0.33414109  0.23265198]
Mean monthly return for iteration--> 21 is 0.2575585952
Variance of monthly return for iteration--> 21 is 1.4652307057


Set of random weight for Iterartion--> 22 is [ 0.21948847  0.23254742
0.52582491  0.0221392 ]
Mean monthly return for iteration--> 22 is 0.28770232827
Variance of monthly return for iteration--> 22 is 2.75435787196


Set of random weight for Iterartion--> 23 is [ 0.12615652  0.20807894
0.40970194  0.2560626 ]
Mean monthly return for iteration--> 23 is 0.250752515384
Variance of monthly return for iteration--> 23 is 1.94983198254


Set of random weight for Iterartion--> 24 is [ 0.19271225  0.29133889
0.17887772  0.33707115]
Mean monthly return for iteration--> 24 is 0.223806621777
Variance of monthly return for iteration--> 24 is 1.03160908235


Set of random weight for Iterartion--> 25 is [ 0.30070032  0.08778169
0.32451712  0.28700088]
Mean monthly return for iteration--> 25 is 0.25134782545
Variance of monthly return for iteration--> 25 is 1.42985881999
```

[36]:
```
# Dataframe containing stock weights,mean and variances of all possible␣
↪portfolios
print(df)
```

	Stock1_weight	Stock2_weight	Stock3_weight	Stock4_weight	mean_return	\
0	0.384819	0.361512	0.164293	0.089377	0.254274	
1	0.004546	0.414222	0.297381	0.283850	0.227607	
2	0.244730	0.414558	0.309235	0.031477	0.264760	
3	0.379061	0.018729	0.287587	0.314624	0.250736	
4	0.294036	0.390466	0.095530	0.219967	0.230471	
5	0.381356	0.151785	0.268664	0.198195	0.257488	
6	0.113788	0.070419	0.453320	0.362474	0.246775	
7	0.055275	0.353076	0.451960	0.139689	0.259109	
8	0.472457	0.137790	0.311250	0.078503	0.277595	
9	0.057417	0.334970	0.360520	0.247093	0.241099	
10	0.274031	0.232362	0.356061	0.137546	0.264114	
11	0.133148	0.345505	0.391052	0.130295	0.258564	
12	0.357226	0.236612	0.288612	0.117550	0.264012	
13	0.198225	0.361075	0.419132	0.021567	0.274401	
14	0.498642	0.042231	0.452387	0.006740	0.300465	
15	0.091428	0.465599	0.138243	0.304730	0.214542	
16	0.151333	0.057770	0.320808	0.470089	0.226698	
17	0.439628	0.203543	0.153709	0.203121	0.248490	
18	0.179738	0.437887	0.376161	0.006214	0.269471	
19	0.213474	0.721337	0.010215	0.054974	0.227615	
20	0.316052	0.117155	0.334141	0.232652	0.257559	
21	0.219488	0.232547	0.525825	0.022139	0.287702	
22	0.126157	0.208079	0.409702	0.256063	0.250753	
23	0.192712	0.291339	0.178878	0.337071	0.223807	
24	0.300700	0.087782	0.324517	0.287001	0.251348	

	var_return
0	0.908981
1	1.572623
2	1.547788
3	1.262056
4	0.784504
5	1.184832
6	2.171107
7	2.338298
8	1.362944
9	1.788027
10	1.630110
11	1.932562
12	1.302548
13	2.134666
14	2.120239
15	1.021119
16	1.567921
17	0.830181
18	1.953710
19	0.815789

```
20    1.465231
21    2.754358
22    1.949832
23    1.031609
24    1.429859
```

[37]:
```
fig = plt.figure(1, figsize=(20, 10))
plt.scatter(df.mean_return,df.var_return, c=df.var_return)
plt.colorbar()
fig.suptitle('Mean Return VS Volatility', fontsize=20)
plt.xlabel('Volatility', fontsize=18)
plt.ylabel('Mean Return', fontsize=16)
plt.show()
```