Optimal Stock Allocation2

December 16, 2020

In [209]:
```
##imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pandas_datareader as web
from matplotlib.ticker import FuncFormatter

from pypfopt.efficient_frontier import EfficientFrontier
from pypfopt import risk_models
from pypfopt import expected_returns
from pypfopt.cla import CLA
from pypfopt import plotting
from matplotlib.ticker import FuncFormatter
from pypfopt import discrete_allocation
```

In [210]:
```
##Scraping data for Adj close price
tickers = ['INDIGO.NS', 'JETAIRWAYS.NS', 'ICICIBANK.NS', 'SBIN.NS', 'TATACHEM.NS', 'T

thelen = len(tickers)

price_data = []

#Getting the open/close prices
for ticker in range(thelen):
    prices = web.DataReader(tickers[ticker], start='2019-01-01', end='2019-12-31', da
    price_data.append(prices.assign(ticker=ticker)[['Adj Close']])
df_stocks = pd.concat(price_data, axis=1)
df_stocks.columns=tickers
df_stocks.head()
```

Out[210]:

Date	INDIGO.NS	JETAIRWAYS.NS	ICICIBANK.NS	SBIN.NS	TATACHEM.NS	\
2019-01-01	1161.959595	280.799988	361.981659	299.600006	667.345398	
2019-01-02	1157.373901	263.549988	362.827515	293.899994	663.134888	
2019-01-03	1152.937866	247.199997	361.484100	291.100006	657.079285	
2019-01-04	1139.629517	245.199997	363.424622	297.649994	660.864075	
2019-01-07	1110.271484	246.199997	365.912445	296.299988	662.236084	

```
                TATACOMM.NS
Date
2019-01-01   514.936157
2019-01-02   514.048828
2019-01-03   528.493652
2019-01-04   524.007385
2019-01-07   521.443726
```

In [225]:
```
##Check for Nan
```

In [226]:
```
nullin_df = pd.DataFrame(df_stocks, columns=tickers)
print(nullin_df.isnull().sum())
##No Nan values (Conclusion)
```

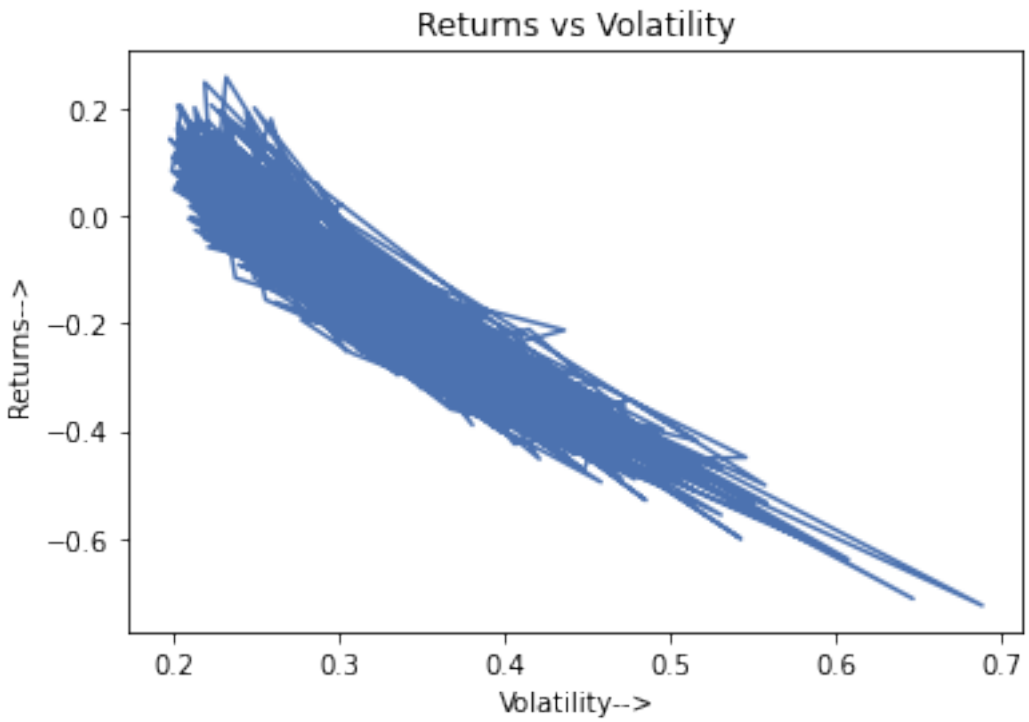
```
INDIGO.NS        0
JETAIRWAYS.NS    0
ICICIBANK.NS     0
SBIN.NS          0
TATACHEM.NS      0
TATACOMM.NS      0
dtype: int64
```

In [224]:
```
#Calculate returns and covariance matrix
returns = df_stocks.pct_change()
cov_matrix_annual = returns.cov() * 242
```

In [222]:
```
##Calculate variance and returns for 1000 weights and store values in list
portfolioSimpleAnnualReturn_list = []
port_volatility_list = []
for i in range(1000):
    weights = np.random.rand(6,1)
    weights /= weights.sum()
    weights = weights.reshape(6)
    port_variance = np.dot(weights.T, np.dot(cov_matrix_annual, weights))
    port_volatility = np.sqrt(port_variance)
    port_volatility_list.append(port_volatility)
    portfolioSimpleAnnualReturn = sum(returns.mean()*weights) * 242
    portfolioSimpleAnnualReturn_list.append(portfolioSimpleAnnualReturn)
```

In [223]:
```
#Plot the graph
fig, ax = plt.subplots()
ax.plot(port_volatility_list, portfolioSimpleAnnualReturn_list)
ax.set(xlabel='Volatility-->', ylabel='Returns-->')
ax.set_title('Returns vs Volatility')
```

Out[223]:
```
Text(0.5, 1.0, 'Returns vs Volatility')
```


Returns vs Volatility
0.2
0.0
−0.2
−0.4
−0.6
Returns-->
0.2
0.3
0.4
0.5
0.6
0.7
Volatility-->