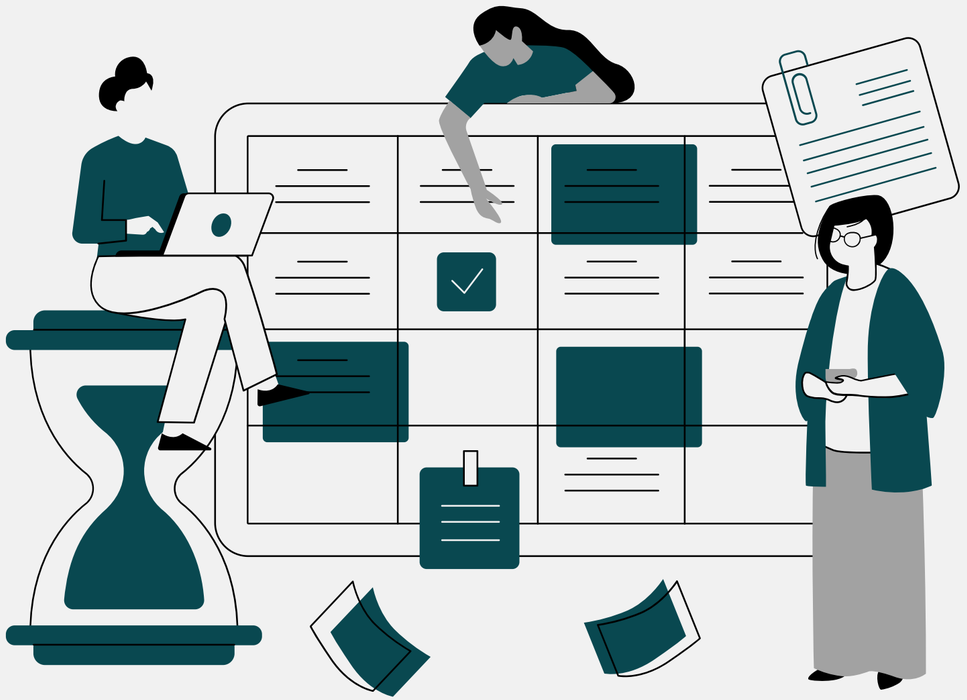
Portfolio Management

SDSC4107 Project

Members:
Leung Tsun Ming 56211500
Ng Tin Lai 56227570
Lam Ming Hon 56207509
Lau Hong Wo Peter 56202655
Wong King Hei 56227662

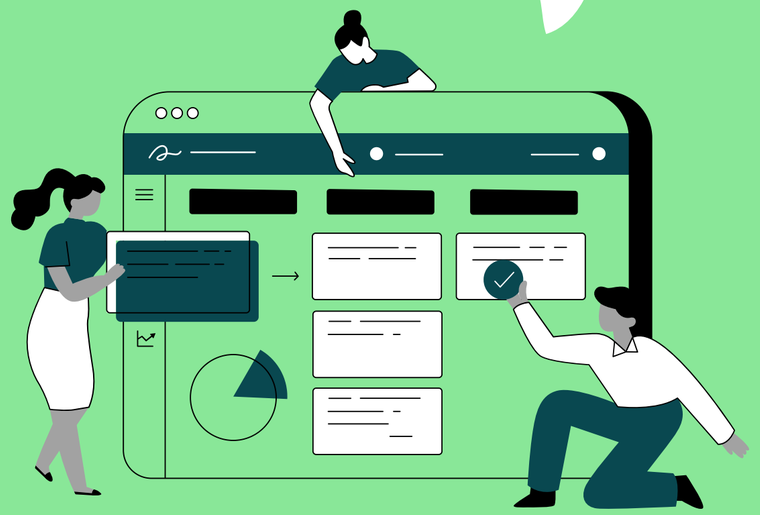
Take a look!

A brief look at what we will discuss on this report



Introduction

What is portfolio management?



1. Asset Allocation
2. Risk Minimization
3. Performance Tracking

Introduction

Why portfolio management is important for investors and asset managers?

Process of Portfolio Management

01

DATA COLLECTION

Determine asset allocation

02

PORTFOLIO SELECTION

Determine the optimal portfolio by using various portfolios

03

MODEL TESTING

Assess how well the model performs on the set of data in reality

Data Collection

Optimization of Portfolio Management in reality

Sources: Yahoo Finance & NASDAQ dataset

Financial Assets:

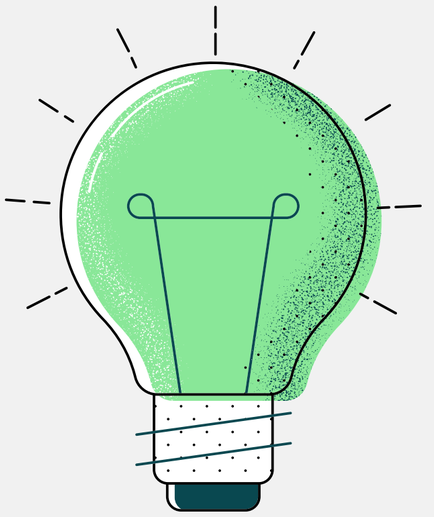
- Risky assets: AAPL, TSLA, META, NFLX, GOOG
- Risk-free asset: ^FVX (5-Years Treasury Bills)

Maturity:

- Portfolio construction: 2017-2021 (1258 trading days)
- Test: 2022 (251 trading days)

Data Exploration

Optimization of Portfolio Management in reality



Data selection: Closing price after adjustments
(Stock price after paying off the dividends)

Normalisation: Finding % of return rather than return value

- Alogorithm: Log-difference (✓symmetrical)

Code (Data Collection)

```
[ ]  #Data Collection
     rf_free = ['^FVX'] #Risk-free asset
     stocks = ['AAPL', 'TSLA','META','NFLX','GOOG'] #Five risky assets
     #Five risky assets and a risk-free asset
     rf_stocks = ['AAPL', 'TSLA','META','NFLX','GOOG','^FVX']
     years = 5 #Maturity = 5 years
     data_rf = yf.download(tickers=rf_stocks, start="2017-01-01", end="2021-12-31", interval="1d") #Download the data
     data_rf = data_rf['Adj Close'] #Data selections: closing price after adjustments
     print(data_rf.shape)
     data_rf.head() #Dataset for five risky assets and a risk-free asset
```

```
[**********************100%**********************]  5 of 5 completed
(1258, 5)
```

Date	AAPL	GOOG	META	NFLX	TSLA
2017-01-03	27.133331	39.306999	116.860001	127.489998	14.466000
2017-01-04	27.102953	39.345001	118.690002	129.410004	15.132667
2017-01-05	27.240784	39.701000	120.669998	131.809998	15.116667
2017-01-06	27.544472	40.307499	123.410004	131.070007	15.267333
2017-01-09	27.796768	40.332500	124.900002	130.949997	15.418667

Code (Data Exploration)

```
[ ]  #Data Exploration
     normalised = (data.pct_change().apply(lambda x: np.log(1+x)).dropna(axis = 0,inplace = False))
     # Normalisation using log difference (first row is deleted)
     # Example: 2017-01-04 to 2017-01-05 AAPL changes
     # x = (27.102953-27.133331)/(27.133331)
     # output : ln(1+x)
     print(np.log(1+((27.102953-27.133331)/(27.133331)))) #Demonstration
     normalised.head() # Normalised data
```

```
-0.0011202096064214535
```

Date	AAPL	GOOG	META	NFLX	TSLA
2017-01-04	-0.001120	0.000966	0.015538	0.014948	0.045055
2017-01-05	0.005073	0.009007	0.016544	0.018376	-0.001058
2017-01-06	0.011087	0.015161	0.022453	-0.005630	0.009918
2017-01-09	0.009118	0.000620	0.012001	-0.000916	0.009863
2017-01-10	0.001008	-0.002309	-0.004413	-0.008128	-0.006115

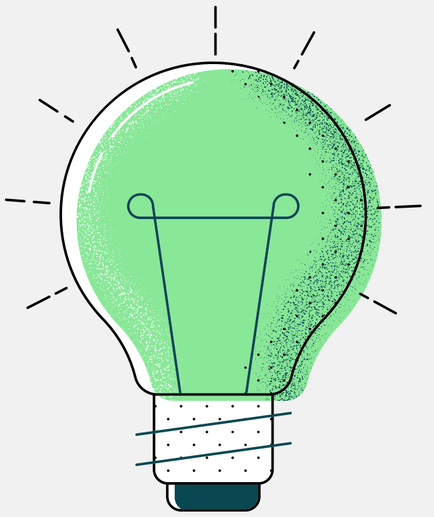
Portfolio Selection

Optimization of Portfolio Management in reality

- Minimize Variance Portfolio (Volatility-based)
- Maximize Sharpe Ratio Portfolio (Sharpe Ratio-based)
- Tangency Portfolio (Sharpe Ratio with Risk-free based)
- Risk Averse Portfolio (Utility-based)

Portfolio Parameters

Expected Return and Variance



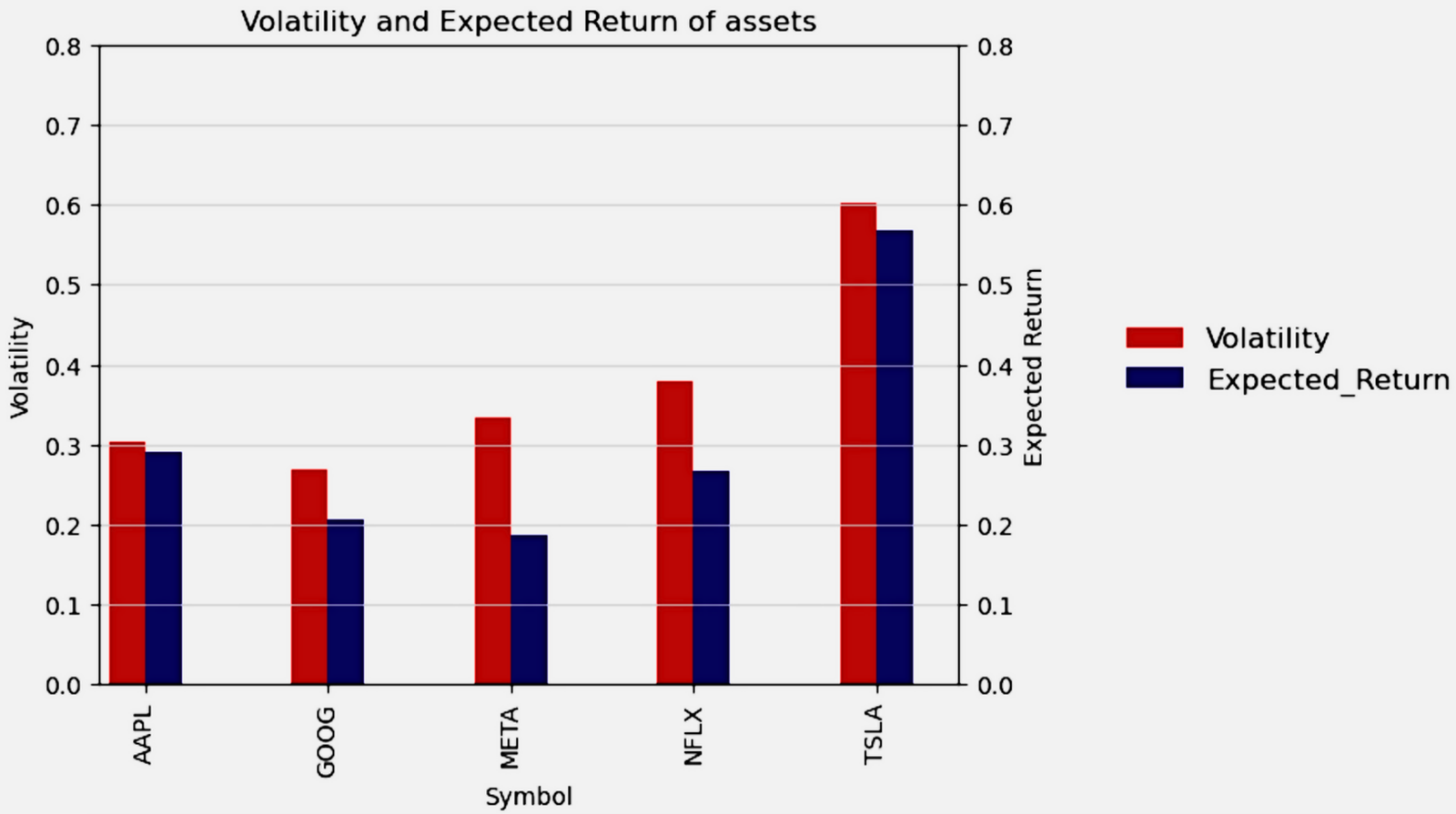
- Expected return:
  - $$E(r_p) = w_D E(r_D) + w_E E(r_E)$$
- Variance:
  - $$\sigma_p^{\,2} = w_D^{\,2}\sigma_D^{\,2} + w_E^{\,2}\sigma_E^{\,2} + 2w_D w_E \operatorname{cov}(r_D, r_E)$$

Code (Portfolio Parameters)

```
[ ]  cov_matrix = normalised.cov() #covarience martix
     #Volatility and Expected Return
     sd = normalised.std()*np.sqrt(round(len(data.index)/years)) #Volatility (sd * root(trading days))
     Expected_return = data.resample('D').last().pct_change().mean() * (len(data.index)/years) #Daily return * trading days = annual return
     combined_set = {'Volatility': sd,'Expected_Return': Expected_return}
     new_df = pd.DataFrame(combined_set)
     new_df.index.name = 'Symbol'
     new_df #Create Dataset for Volatility and Expected Return
```

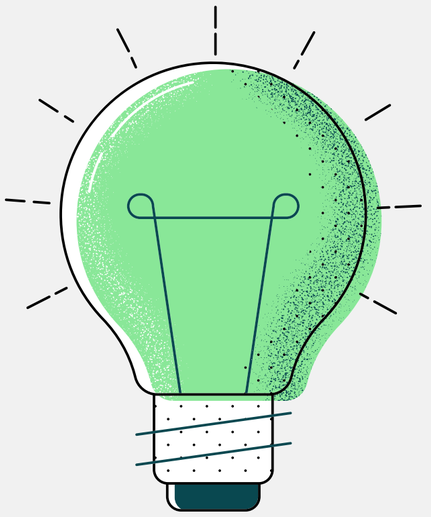
Symbol	Volatility	Expected_Return
AAPL	0.304878	0.291061
GOOG	0.270128	0.206402
META	0.334355	0.187545
NFLX	0.380226	0.266652
TSLA	0.601808	0.568051

Code (Data Collection)



Portfolio Parameters

Optimal Weight



- Equation:

$$1 - w^* = \frac{E(\tilde{r}_{mkt}) - r_f}{2\gamma\sigma^2_{mkt}}$$

- risk averse= γ

Code (Portfolio Parameters)

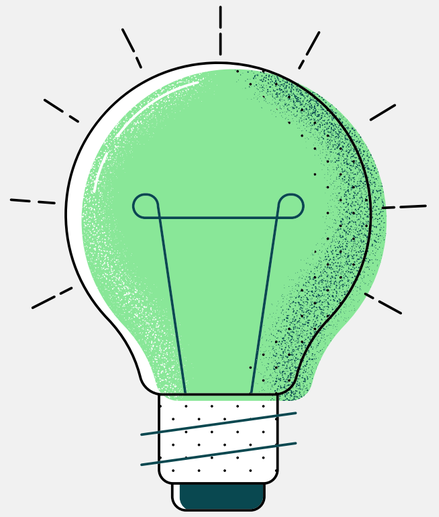
```
[ ] returning,sigma,weighting = [],[],[]
    trading = len(data.columns)
    #cov_matrix2 = cov_matrix.drop(columns = ['^FVX'],index = ['^FVX'])

    for portfolio in range(100000):

        weights = np.random.dirichlet(np.ones(5)) #random choose weighting (dirichlet since sum of weight = 1)
        weights = weights/np.sum(weights) #weight mean (np.sum(weights) suppose be 1)
        weighting.append(weights) #Empty append
        returns = np.dot(weights, Expected_return) #Plot dots
        returning.append(returns) #Empty append
        varience = cov_matrix.mul(weights, axis=0).mul(weights, axis=1).sum().sum() #Varience calculation
        standard = np.sqrt(varience)
        volatility = standard *np.sqrt(round(len(data.index)/years))
        sigma.append(volatility)

    adj_data = {'Expected_Return':returning, 'Volatility':sigma}

    for counter, symbol in enumerate(data.columns.tolist()):
        adj_data[symbol+' weight'] = [w[counter] for w in weighting]

    portfolios = pd.DataFrame(adj_data)
    print(portfolios.shape)
    print(np.sum(weights))
    portfolios.head()
```

```
(100000, 7)
1.0
```

	Expected_Return	Volatility	AAPL weight	GOOG weight	META weight	NFLX weight	TSLA weight
0	0.227688	0.282730	0.045595	0.277933	0.559154	0.047968	0.069350
1	0.342036	0.311602	0.134561	0.312809	0.037741	0.203238	0.311652
2	0.256503	0.263813	0.179683	0.387316	0.109481	0.265583	0.057937
3	0.486273	0.476154	0.142119	0.045325	0.009095	0.074850	0.728611
4	0.223119	0.276505	0.053919	0.393979	0.491098	0.002154	0.058851

Concept

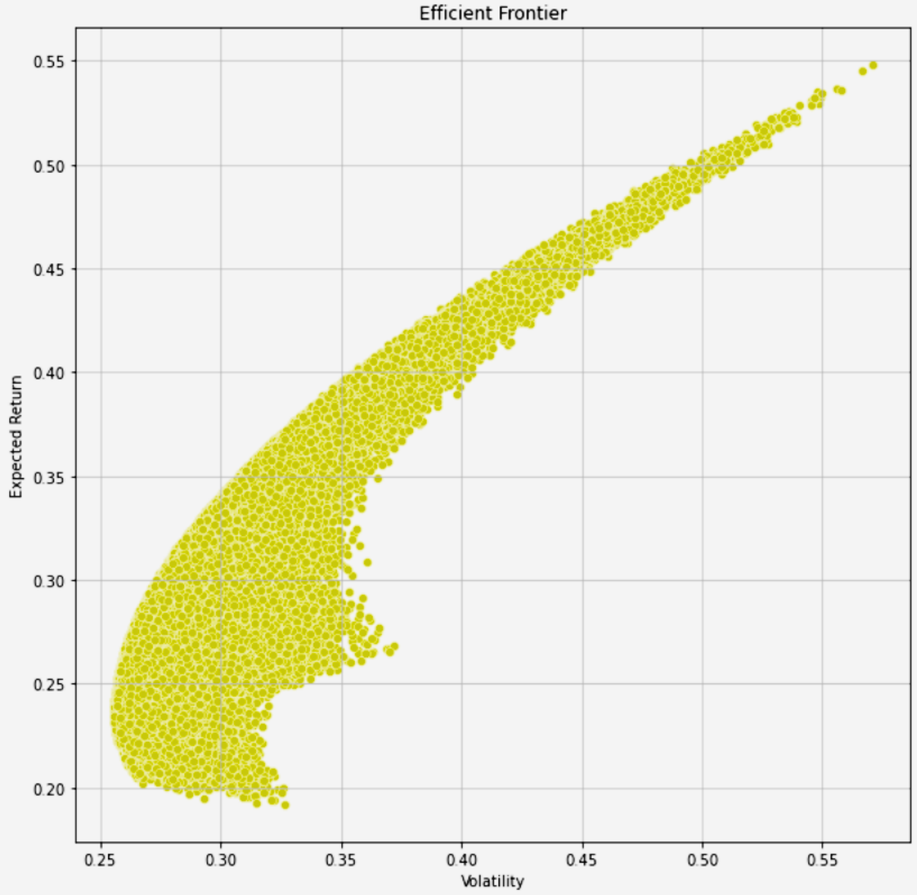
Efficient Frontier



- Set of optimal portfolios
- Offers the level of expected return
- Relative weights are considered
- Usually as a concave curve

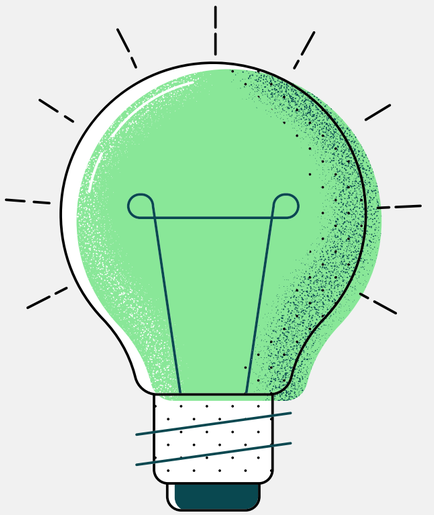
Code (Efficient Frontier)

```
[ ]  fig, ax = plt.subplots(figsize=(10, 10))
     sns.scatterplot(data=portfolios, x="Volatility", y="Expected_Return",markers = 'o', ax=ax, color='y')
     ax.set(xlabel='Volatility', ylabel='Expected Return')
     plt.title('Efficient Frontier');
     ax.grid(True)
     plt.show() #Efficient Frontier
```



Portfolio Selection

Minimum variance portfolio



- Portfolio that gives lowest variance (risk),
- while it still achieves considerable expected return

- Equation:

$$w^{MVP} = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}'\Sigma^{-1}\mathbf{1}}$$

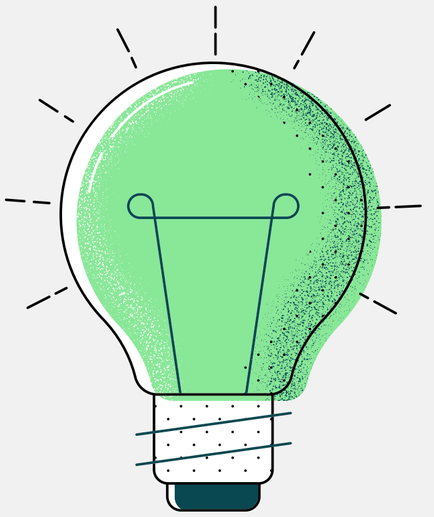
Code (Portfolio Selection)

```
[ ]  #Portfolio 1: Minimize Volatility (correct)
     min_vol_port = portfolios.iloc[portfolios['Volatility'].idxmin()] #Portfolio 1: Minimize the volatility
     min_vol_port
```

```
Expected_Return    0.235929
Volatility         0.255587
AAPL weight        0.261529
GOOG weight        0.538681
META weight        0.078901
NFLX weight        0.115612
TSLA weight        0.005276
Name: 7990, dtype: float64
```

Concept

Capital allocation line



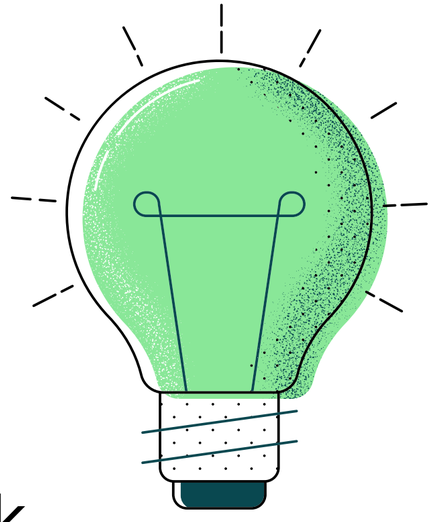
- The balance between expected return and risk
- Displays graphically (as linear relationship)
- Accounts for the investor's customized preferences.

  Slope: Sharpe Ratio (will be discussed)

  y- intercept: Risk free rate

Concept

Sharpe Ratio



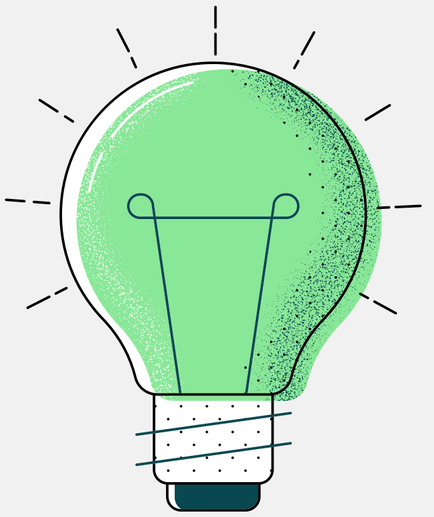
- The slope of CAL
- Represent the ratio of the risk premium to portfolio risk

- Equation:

$$\text{Sharpe Ratio} = \frac{E(\tilde{r}_p) - r_f}{\sigma_p}$$

Portfolio Selection

Maximum Sharpe Ratio Portfolio



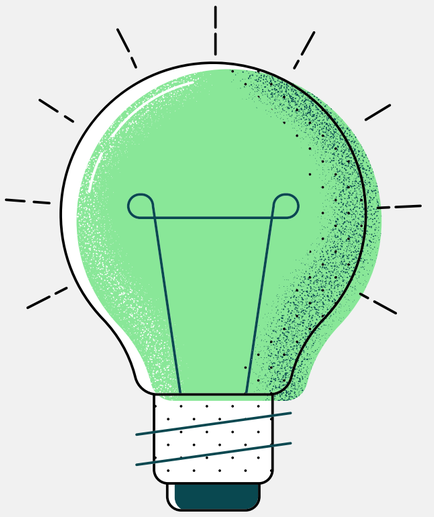
- Portfolio that has highest sharpe ratio
- Gives greatest risk-adjusted return

Code (Portfolio Selection)

```
[ ]  #Portfolio 2: Tangency Portfolio with five risky assets (Maximize Sharpe Ratio Portfolio)
     portfolios["Sharpe_Ratio"] = (portfolios['Expected_Return'])/portfolios['Volatility'] #rf=0 since not include risk-free assets
     optimal_risky_port = portfolios.loc[portfolios['Sharpe_Ratio'].idxmax()]
     optimal_risky_port

     Expected_Return    0.362150
     Volatility         0.317194
     AAPL weight        0.539027
     GOOG weight        0.067283
     META weight        0.000546
     NFLX weight        0.106355
     TSLA weight        0.286790
     Sharpe_Ratio       1.141733
     Name: 53721, dtype: float64
```

Portfolio Selection

Tangency Portfolio



- Maximizes the slope of the CAL
- Unlike Portfolio 2, risk-free asset also be considered
- To maximize it, we can use the formula below

$$w^T = \frac{\Sigma^{-1}\mu}{\mathbf{1}'\Sigma^{-1}\mu} \qquad \mu = E(\tilde{r}) - r_f$$

Code (Portfolio Parameter)

```
# Generate Portfolios with risk-free asset
returning_rf,sigma_rf,weighting_rf = [],[],[]
trading_rf = len(data_rf.columns)
for portfolio_rf in range(100000):
    weights_rf = np.random.dirichlet(np.ones(6)) #random choose weight (dirichlet)
    if (weights_rf[0]* new_df_rf["Expected_Return"][0] + weights_rf[1]* new_df_rf["Expected_Return"][1]
        +weights_rf[2]* new_df_rf["Expected_Return"][2]+weights_rf[3]* new_df_rf["Expected_Return"][3]
        +weights_rf[4]* new_df_rf["Expected_Return"][4]+weights_rf[5]* new_df_rf["Expected_Return"][5]
        == optimal_risky_port[0]):
# Condition 1: Expected return of all assets
# = Expected retuen of Maximize Sharpe Ratio Portfolio
      if(weights_rf[0]* new_df_rf["Volatility"][0]+weights_rf[1]* new_df_rf["Volatility"][1]
         +weights_rf[2]* new_df_rf["Volatility"][2]+weights_rf[3]* new_df_rf["Volatility"][3]
         +weights_rf[4]* new_df_rf["Volatility"][4] == optimal_risky_port[1]):
# Condition 2:  Volatility of risky assets (risk-free asset = 0)
# = Volatility of Maximize Sharpe Ratio Portfolio

        weights_rf = weights_rf/np.sum(weights_rf) #weight mean
        weighting_rf.append(weights_rf) #Empty append

    else:
        weights_rf = weights_rf/np.sum(weights_rf) #weight mean
        weighting_rf.append(weights_rf) #Empty append
        returns = np.dot(weights_rf, Expected_return_rf) #Plot dots with all assets
        returning_rf.append(returns) #Empty append
        #Volatility calculation
        varience = cov_matrix.mul(weights, axis=0).mul(weights, axis=1).sum().sum()
        standard = np.sqrt(varience)
        volatility = standard *np.sqrt(round(len(data.index)/years))
        sigma_rf.append(volatility)

adj_data_rf = {'Expected_Return':returning_rf, 'Volatility':sigma_rf}

for counter, symbol in enumerate(data_rf.columns.tolist()):
    adj_data_rf[symbol+' weight'] = [w[counter] for w in weighting_rf]

portfolios_rf = pd.DataFrame(adj_data_rf)
portfolios_rf = portfolios_rf.drop(columns =["Expected_Return","Volatility"])
risky_asset = ['AAPL weight','GOOG weight','META weight','NFLX weight','TSLA weight']
risk_free_asset = ['^FVX weight']
portfolios_rf['risky_weight'] = portfolios_rf[risky_asset].sum(axis=1)
portfolios_rf['risk_free_weight'] = portfolios_rf[risk_free_asset].sum(axis=1)
portfolios_rf.head() #Create 100000 portfolios with random weights
```

	AAPL weight	GOOG weight	META weight	NFLX weight	TSLA weight	^FVX weight	risky_weight	risk_free_weight
0	0.317805	0.124810	0.048050	0.167283	0.042523	0.299528	0.700472	0.299528
1	0.428072	0.113920	0.136062	0.226483	0.083436	0.012027	0.987973	0.012027
2	0.002269	0.081398	0.741907	0.081859	0.002537	0.090030	0.909970	0.090030
3	0.198923	0.342043	0.047799	0.089968	0.217994	0.103273	0.896727	0.103273
4	0.067635	0.184935	0.256558	0.003055	0.385308	0.102508	0.897492	0.102508

Generate weights of all assets

- Expected return: All assets
- Variance: Risky assets only
- Sharpe Ratio: Same

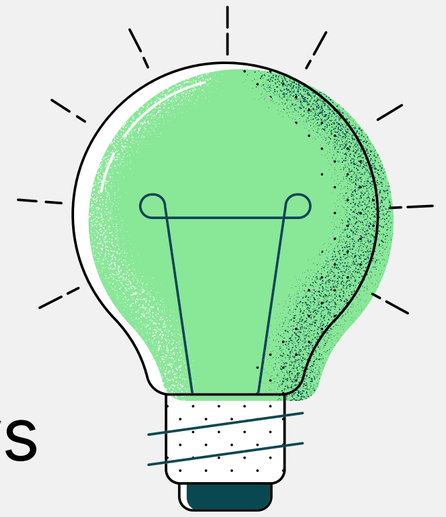
Code (Portfolio Selection)

```
#Portfolio 3: Tangency Portfolio
portfolios_rf['Expected_Return'] = optimal_risky_port[0]
* portfolios_rf['risky_weight'] + portfolios_rf['risk_free_weight'] * rf
portfolios_rf['Volatility'] = optimal_risky_port['Volatility']
* portfolios_rf['risky_weight']
portfolios_rf['Sharpe_Ratio'] = (portfolios_rf['Expected_Return'])
                                /portfolios_rf['Volatility']
optimal_risk_free = portfolios_rf.loc[portfolios['Sharpe_Ratio'].idxmax()]
print(optimal_risk_free)
```

```
AAPL weight          0.269948
GOOG weight          0.267350
META weight          0.035897
NFLX weight          0.063275
TSLA weight          0.118230
^FVX weight          0.245300
risky_weight         0.754700
risk_free_weight     0.245300
Expected_Return      0.269268
Volatility           0.233212
Sharpe_Ratio         1.154607
```

Concept

Utility Function U(E,σ)



- Model for the preferences and choices of the investors
- Represents the attitude about an investment
- Utility means expected return with risk averse

- Equation:

$$U(E,\sigma) = E - \gamma\sigma^2$$

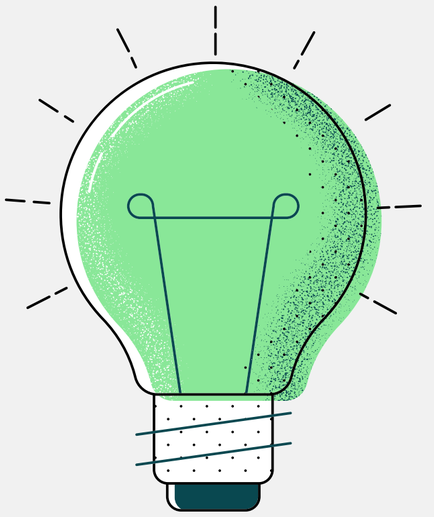
γ = the coefficient of risk aversion, where γ> 0
γ is in [1,5], where γ = 5 is the most risk averse

Code (Utility Function)

```
# Utility Function
X = []
Y = []
utility = []
a = 5 #risk averse coefficient
max_returns = portfolios.Expected_Return.max()

for optimal_returns in np.linspace(rf, max_returns, 1000):
    optimal_volatility = (optimal_returns - rf)/((optimal_risky_port[0]-rf)
                                                 /optimal_risky_port[1])
    initial = optimal_returns - 0.5*a*(optimal_volatility**2)
    X.append(optimal_volatility)
    Y.append(optimal_returns)
    utility.append(initial)

data2 = {'Utility':utility, 'Optimial volatility':X, 'Optimial returns':Y}
cal = pd.DataFrame(data2)

print(cal.shape)
cal.head()

investors_port = cal.iloc[cal['Utility'].idxmax()]
investors_port
```

```
(1000, 3)

Utility                0.134019
Optimial volatility    0.220546
Optimial returns       0.255620
Name: 455, dtype: float64
```

Portfolio Selection



Risk Averse Portfolio

- Minimizes the risk of loss
- Maximizes the utility function
- Target: Investors who are risk averse
- To get optimal expected return portfolio

- Equation: $max_w U(E,\sigma) = max_w\{wr_f + (1-w)E(\tilde{r}_{mkt}) - \gamma((1-w)^2\sigma^2_{mkt})\}$

Code (Portfolio Selection)

Risk Averse Portfolio

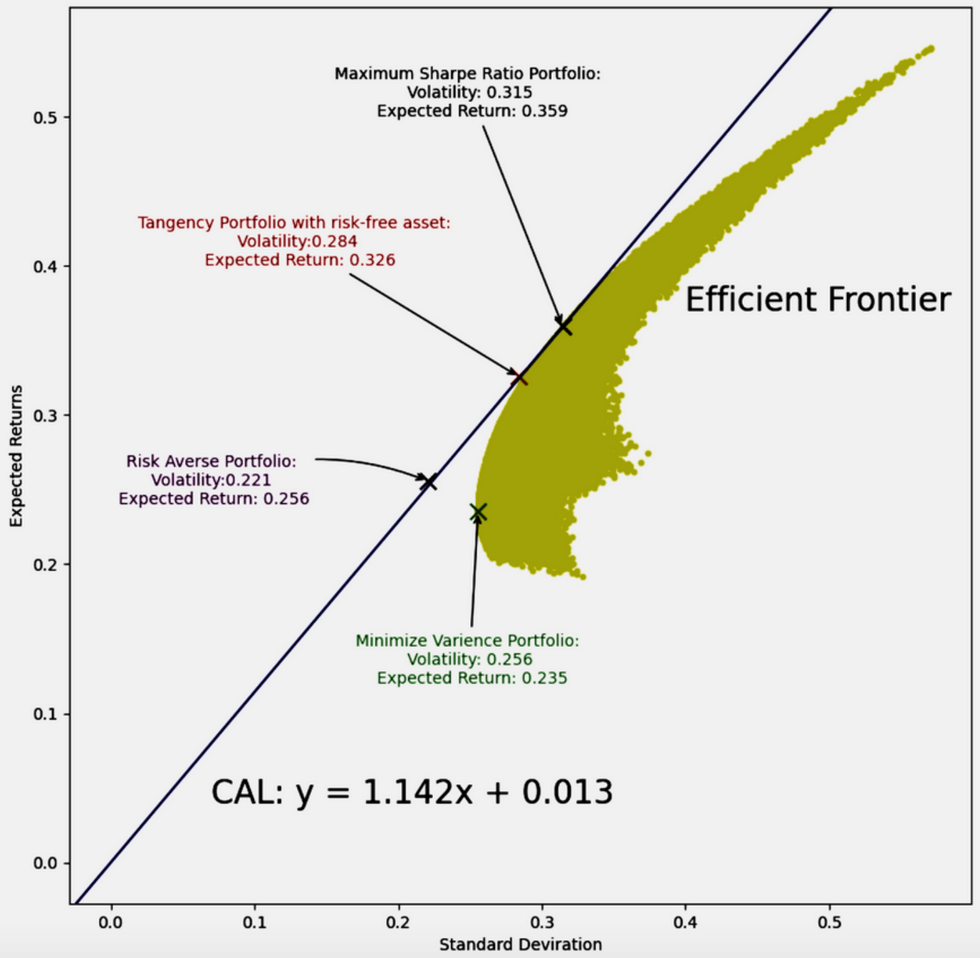
```
# Adjusted Risk Averse portfolio
average_risk = investors_port[1]/optimal_risky_port[1]

risk_rate = optimal_risky_port[2:]*average_risk

# find final returns, vol, weights of rf
risk_free_rate = pd.Series([(1-average_risk)], index=['Cash Flow'])

final_portfolio = pd.concat([investors_port,risk_rate,risk_free_rate], axis=0)
.rename({'Optimial volatility':'Volatility',
         'Optimial returns':'Expected_Return'})
final_portfolio
```

Out[94]:

```
Utility            0.133729
Volatility         0.220137
Expected_Return    0.254879
AAPL weight        0.346866
GOOG weight        0.088892
META weight        0.000435
NFLX weight        0.083053
TSLA weight        0.193143
Sharpe_Ratio       0.813073
Cash Flow          0.287611
dtype: float64
```

Graph (Portfolio Selection)

All Portfolios Graph



Code (Portfolio Selection)

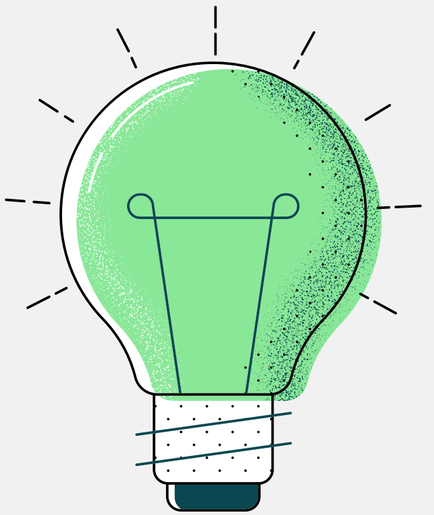
All portfolio weighting

```
table = pd.concat([min_vol_port.to_frame().T
                   ,optimal_risky_port.to_frame().T
                   ,optimal_risk_free.to_frame().T,
                   final_portfolio.to_frame().T])
table = table.reset_index()
table = table.rename({0:'Minimize Variance Portfolio',
                      1:'Maximum Sharpe Ratio Portfolio',
                      2:'Tangency Portfolio with risk-free asset',
                      3:'Risk Averse Portfolio'})
table = table.drop(columns = ['Sharpe_Ratio','Utility','Cash Flow'
                              ,'risky_weight','risk_free_weight'
                              ,'index'])
table['^FVX weight'] = table['^FVX weight'].fillna(0)
table
```

	Expected_Return	Volatility	AAPL weight	GOOG weight	META weight	NFLX weight	TSLA weight	^FVX weight
Minimize Variance Portfolio	0.235121	0.255582	0.262961	0.549082	0.064692	0.122435	0.000831	0.000000
Maximum Sharpe Ratio Portfolio	0.359306	0.314651	0.523709	0.057844	0.001656	0.139794	0.276997	0.000000
Tangency Portfolio with risk-free asset	0.325551	0.284015	0.065973	0.315833	0.025405	0.044097	0.451329	0.097364
Risk Averse Portfolio	0.255620	0.220546	0.367080	0.040544	0.001161	0.097985	0.194154	0.000000

Verification

Comparision using real data



```
test = yf.download(tickers=rf_stocks, start="2022-01-01", end="2022-12-31",
                   interval="1d") # Test data
test = test['Adj Close']
print(test.shape)
test.head()
```

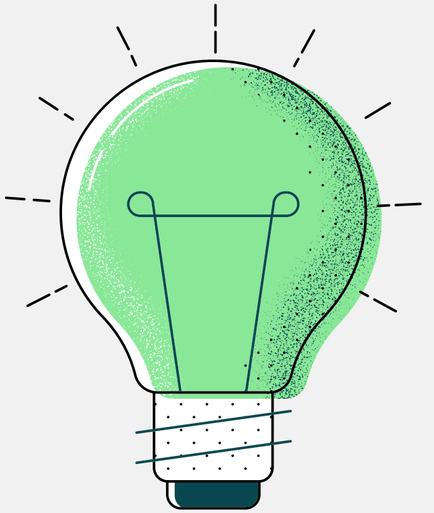
```
[**********100%**********]  6 of 6 completed
(251, 6)
```

Out[97]:

Date	AAPL	GOOG	META	NFLX	TSLA	^FVX
2022-01-03	180.683884	145.074493	338.540009	597.369995	399.926666	1.365
2022-01-04	178.390701	144.416504	336.529999	591.150024	383.196655	1.375
2022-01-05	173.645538	137.653503	324.170013	567.520020	362.706665	1.433
2022-01-06	170.746811	137.550995	332.459991	553.289978	354.899994	1.477
2022-01-07	170.915588	137.004501	331.790009	541.059998	342.320007	1.505

Verification

Comparision using real data



```
#Same for training data
normalised_t = (test.pct_change().apply(lambda x: np.log(1+x))
                .dropna(axis = 0,inplace = False))
normalised_t.head()
sd_t = normalised_t.std()*np.sqrt(round(len(test.index)/1))
Expected_return_t = test.resample('D').last().pct_change().mean()
* (len(test.index)/1)
combined_set_t = {'Volatility': sd_t,'Expected_Return': Expected_return_t}
new_df_t = pd.DataFrame(combined_set_t)
new_df_t.index.name = 'Symbol'
new_df_t
```

Out[98]:

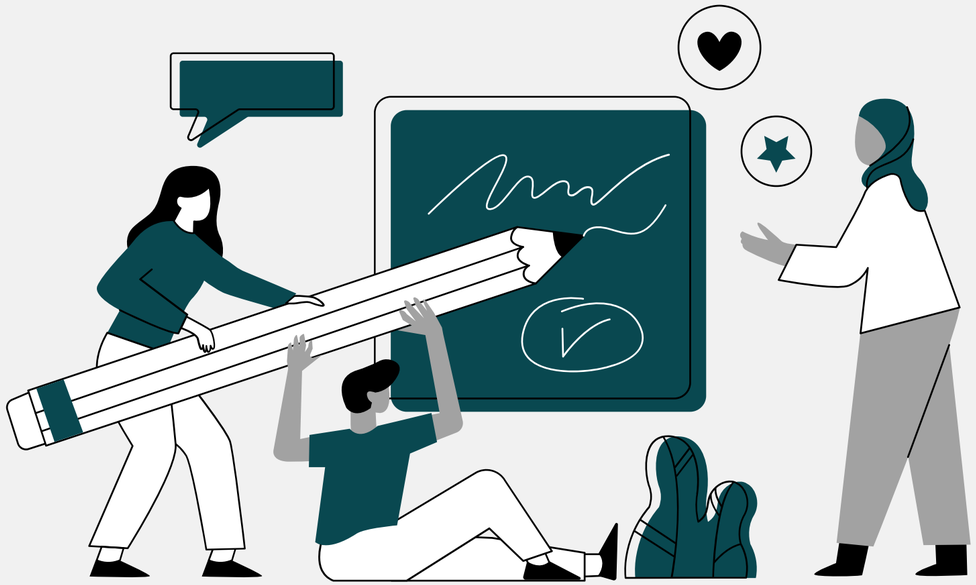
Symbol	Volatility	Expected_Return
AAPL	0.355398	-0.186599
GOOG	0.387983	-0.289661
META	0.672232	-0.567051
NFLX	0.749662	-0.305890
TSLA	0.657922	-0.668783
^FVX	0.487344	0.831511

Result

```
#Verification: Compute return using portfolios
#new_p1: Minimize Volatility Portfolio
#new_p2: Maxmize Sharpe Ratio Portfolio
#new_p3: Tangency Portfolio
#new_p4: Risk Averse Portfolio
new_p1 = table['AAPL weight'][0]*new_df_t['Expected_Return'][0]+table['GOOG weight'][0]*new_df_t['Expected_Return'][1]
+table['META weight'][0]*new_df_t['Expected_Return'][2]+table['NFLX weight'][0]*new_df_t['Expected_Return'][3]
+table['TSLA weight'][0]*new_df_t['Expected_Return'][4]
new_p2 = table['AAPL weight'][1]*new_df_t['Expected_Return'][0]+table['GOOG weight'][1]*new_df_t['Expected_Return'][1]
+table['META weight'][1]*new_df_t['Expected_Return'][2]+table['NFLX weight'][1]*new_df_t['Expected_Return'][3]
+table['TSLA weight'][1]*new_df_t['Expected_Return'][4]
new_p3 = table['AAPL weight'][2]*new_df_t['Expected_Return'][0]+table['GOOG weight'][2]*new_df_t['Expected_Return'][1]
+table['META weight'][2]*new_df_t['Expected_Return'][2]+table['NFLX weight'][2]*new_df_t['Expected_Return'][3]
+table['TSLA weight'][2]*new_df_t['Expected_Return'][4]+table['^FVX weight'][2]*new_df_t['Expected_Return'][5]
new_p4 = table['AAPL weight'][3]*new_df_t['Expected_Return'][0]+table['GOOG weight'][3]*new_df_t['Expected_Return'][1]
+table['META weight'][3]*new_df_t['Expected_Return'][2]+table['NFLX weight'][3]*new_df_t['Expected_Return'][3]
+table['TSLA weight'][3]*new_df_t['Expected_Return'][4]
print(new_p1,new_p2,new_p3,new_p4)
# Reason: Too less assets (diversiation too low)
# Suggestion: Increase no. of stocks!!
```

```
-0.20811590512000339 -0.11447850614910705 -0.10379506698813241 -0.08024069173813983
```

How can we do better?



Portfolio diversification

Buy more stock from different industry

Do the Short Side

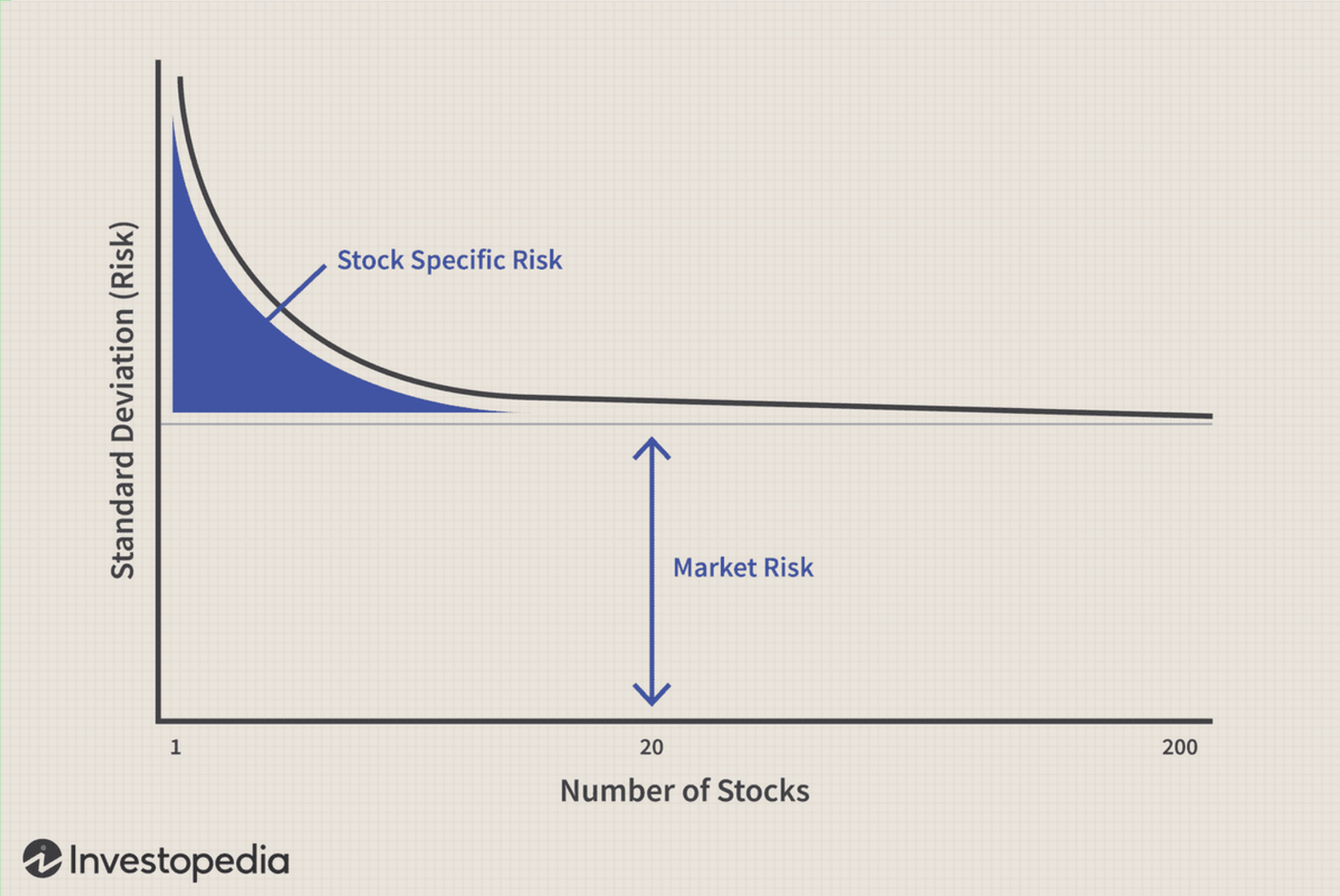
Use technical analysis to identified down trend

Update portfolio

Build a portfolio that can update everyday

Portfolio diversification

Diversification can help reduce the volatility. The simplest explain is to buy more stock from different industries. The theory implies that if a portfolio has more than 20 stocks with a negative correlation to each other, the risk of portfolio will have a significant drop.



Do the short side

Due to the covid-19, the stock price has a big drop in those period. Basic technical analysis tells us that there is a bear trend if the stock price keeps creating higher low and lower low. including technical analysis will provide us with a reference that when we should short the stock for keeping our portfolio more profitable.

Update Portfolio

Financial market are changing in every seconds. Updating portfolio would benefit on both risk and expected return. If a portfolio can instantly react to the news, it will become more stable and profitable.

Netflix closes down 35% wiping more than $50 billion off market cap

PUBLISHED WED, APR 20 2022•9:41 AM EDT | UPDATED WED, APR 20 2022•7:47 PM EDT

Jessica Bursztynsky @JBURSZ | Sarah Alessandrini @SARAHALESSAN

SHARE

KEY POINTS

- Shares of Netflix plunged Wednesday after the streamer reported it lost subscribers in its most recent quarter.
- The company shed more than $50 billion in market cap as a result.
- At least nine Wall Street firms downgraded the stock on the disappointing report.

When Elon Musk announced that Tesla would accept Bitcoin as payment for its cars, it proved to be a short-lived initiative. But the EV company's move opened a door for Adidas' plans to use blockchain, said the sportswear company's Web3 lead Erika Wykes-Sneyd.