Assignment 2 AP

September 29, 2024

ASSIGNMENT 2

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

from sklearn.linear_model import LinearRegression
```

[5]:
```
data_ind_raw = pd.read_excel('Industry_Portfolios.xlsx')
data_ind = data_ind_raw.drop("Date", axis = 1)
data_mkt_raw = pd.read_excel('Market_Portfolio.xlsx')
data_mkt = data_mkt_raw.drop("Date", axis = 1)
```

Market Model

[11]:
```
# risk-free rate
Rf = 0.13
```

[13]:
```
# industry excess returns (y variable)
ind_excess = np.array(data_ind) - Rf
```

[15]:
```
# market excess returns (x variable)
mkt_excess = np.array(data_mkt) - Rf
```

[17]:
```
# market model regression
MM = LinearRegression().fit(mkt_excess ,ind_excess)
MM_alpha = MM.intercept_
MM_beta = MM.coef_
```

[64]:
```
# market model coefficients
MM_coefficient = pd.DataFrame(np.concatenate((MM_alpha.reshape(1,10),MM_beta.
 ↪reshape(1,10))),
                              index = ['Intercept ( )','Slope ( )'],
                              columns = data_ind.columns)
MM_coefficient
```

[64]:
```
                   NoDur     Durbl     Manuf     Enrgy     HiTec     Telcm  \
Intercept ( )  0.369443 -0.415599  0.159771  0.501719 -0.064020  0.194691
Slope ( )      0.652647  1.648536  1.169846  0.969850  1.132969  0.900729

                   Shops      Hlth     Utils     Other
Intercept ( )  0.275492  0.237841  0.444585 -0.387135
Slope ( )      0.826492  0.673036  0.538086  1.207309
```

Briefly explain (in words, without mathematical equations or formulas) the economic significance and pricing implications of the intercept and slope coefficients.

Alpha ( ): Measures abnormal returns. Positive suggests outperformance, while negative indicates underperformance. In efficient markets, should be close to zero. Deviations suggest mispricing.

Beta ( ): Captures portfolio sensitivity to the market. A of 1 means the portfolio moves with the market; higher implies greater risk and potential return. Higher portfolios should earn more to compensate for the risk.

Capital Asset Pricing Model (CAPM)

[39]:
```python
data_merge = data_ind_raw.merge(data_mkt_raw)
data_merge = data_merge.drop("Date", axis = 1)

# consolidated mean returns (y variable)
consolidated_return = data_merge.mean()
consolidated_return
```

[39]:
```
NoDur     0.902833
Durbl     0.733333
Manuf     1.012833
Enrgy     1.231167
HiTec     0.766250
Telcm     0.881417
Shops     0.916333
Hlth      0.783833
Utils     0.907167
Other     0.489083
Market    0.748083
dtype: float64
```

[41]:
```python
# market covariance matrix
consolidated_cov = data_merge.cov()
consolidated_cov["Market"]
```

[41]:
```
NoDur     12.300096
Durbl     31.069071
Manuf     22.047469
Enrgy     18.278244
```

```
HiTec     21.352470
Telcm     16.975563
Shops     15.576461
Hlth      12.684344
Utils     10.141021
Other     22.753517
Market    18.846466
Name: Market, dtype: float64
```

[45]:
```python
# market variance
market_var = consolidated_cov.iloc[10,10]
market_var
```

[45]:
```
18.84646604341737
```

[27]:
```python
# consolidated beta (x variable)
consolidated_beta = consolidated_cov["Market"]/market_var
consolidated_beta = pd.DataFrame(np.array(consolidated_beta),
                                 columns=['Beta ( )'],
                                 index = data_merge.columns)
consolidated_beta
```

[27]:

	Beta ( )
NoDur	0.652647
Durbl	1.648536
Manuf	1.169846
Enrgy	0.969850
HiTec	1.132969
Telcm	0.900729
Shops	0.826492
Hlth	0.673036
Utils	0.538086
Other	1.207309
Market	1.000000

[29]:
```python
# capital asset pricing model regression
CAPM = LinearRegression().fit(consolidated_beta, consolidated_return)
CAPM_alpha = CAPM.intercept_
CAPM_beta = CAPM.coef_[0]
```

[68]:
```python
# capital asset pricing model coefficients
CAPM_coefficient = pd.DataFrame((CAPM_alpha, CAPM_beta),
                                columns=["CAPM"],
                                index=["Intercept ( )", "Slope ( )"])
CAPM_coefficient
```

[68]:
```
                         CAPM
Intercept ( )  1.032768
Slope ( )     -0.185467
```

Security Market Line (SML)

[57]:
```python
# security market line returns
SML_return = np.arange(0, 2.01 , 1)
SML_return
```

[57]:
```
array([0., 1., 2.])
```

[59]:
```python
#security market line beta
SML_beta = CAPM_alpha + CAPM_beta*SML_return
SML_beta
```

[59]:
```
array([1.03276837, 0.84730091, 0.66183345])
```

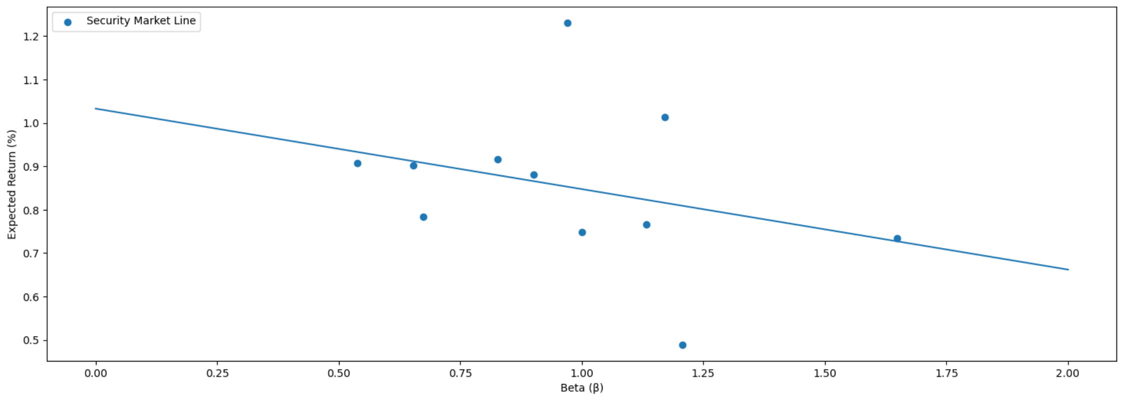
Plot the Security Market Line (SML)

[37]:
```python
plt.figure(figsize = (18, 6))

plt.scatter(consolidated_beta, consolidated_return)
plt.plot(SML_return, SML_beta, label ='SML')

plt.ylabel('Expected Return (%)')
plt.xlabel('Beta ( )')
plt.legend(["Security Market Line"], loc=2)
```

[37]:
```
<matplotlib.legend.Legend at 0x164cca870>
```



[ ]:
```
Briefly explain the economic significance and pricing implications of the SML.
```

The SML shows the relationship between risk (beta) and expected return under CAPM. Portfolios on the SML are fairly priced. If a portfolio is above the SML, it's undervalued, offering more return for its risk. Portfolios below the SML are overvalued.

Pricing implications: Investors use the SML to identify mispriced securities, presenting opportunities for profit if undervalued or potential loss if overvalued.