AP HW3

Linear Factor Models

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [2]:
```
industry_portfolios = \
    pd\
    .read_csv("/Users/lu/Desktop/Industry_Portfolios.csv")

risk_factor =\
    pd\
    .read_csv("/Users/lu/Desktop/Risk_Factors.csv")
```

In [3]:
```
#risk_factor
# print(industry_portfolios.columns)
# print(risk_factor.columns)
```

In [4]:
```
#risk_factor["Rf"].to_frame()
```

In [5]:
```
data = pd.merge(industry_portfolios, risk_factor, on='Date')
# data
```

Fama-French 3-Factor Model:

$$\tilde{R}_i - R_f = \alpha_i + \beta_i(\tilde{R}_m - R_f) + \gamma_i(\tilde{R}_s - \tilde{R}_b) + \delta_i(\tilde{R}_h - \tilde{R}_l) + \tilde{\epsilon}_i$$

In [6]:
```
industry_columns = ['NoDur', 'Durbl', 'Manuf', 'Enrgy', 'HiTec', 'Telcm',

for industry in industry_columns:
    data[industry] = data[industry] - data['Rf']

# data# excess return
```

Create A Table

In [7]:
```
X = sm.add_constant(data[["Rm-Rf",'SMB', 'HML']])
factor = []

for industry in industry_columns:
    y = data[industry]
    model = sm.OLS(y, X).fit()
    factor.append({
        "Industry": industry,
        "Fama-French 3-factor alpha": model.params["const"],
        "Market risk": model.params["Rm-Rf"],
        "SMB Loading": model.params['SMB'],
        'HML Loading': model.params['HML'],
```

```
    })

df = pd.DataFrame(factor)
df
```

Out[7]:

	Industry	Fama-French 3-factor alpha	Market risk	SMB Loading	HML Loading
0	NoDur	0.386704	0.712134	-0.229102	-0.023342
1	Durbl	-0.474342	1.447452	0.670878	0.240949
2	Manuf	0.153285	1.142282	0.087388	0.027727
3	Enrgy	0.523007	1.028354	-0.259360	-0.008158
4	HiTec	-0.065979	1.152803	0.335674	-0.556947
5	Telcm	0.200724	0.924137	-0.080299	-0.019063
6	Shops	0.255941	0.770227	0.280191	-0.039080
7	Hlth	0.257472	0.751976	-0.212655	-0.143765
8	Utils	0.474411	0.631827	-0.387961	-0.016881
9	Other	-0.404412	1.123473	-0.061676	0.547325

Calculate:

> Sharpe ratio
> Sortino ratio (using risk-free rate as target)
> Treynor ratio (using CAPM β)
> Jensen's α
> Fama–French three-factor α

In [8]:

```
X = sm.add_constant(data[["Rm-Rf"]])
factor = []

for industry in industry_columns:
    y = data[industry]
    model = sm.OLS(y, X).fit()  # OLS
    factor.append({
        "Industry": industry,
        "CAPM": model.params["const"],
        "Market risk": model.params["Rm-Rf"], # beta_i
    })

df1 = pd.DataFrame(factor)
df1
```

Out[8]:

	Industry	CAPM	Market risk
0	NoDur	0.369717	0.653744
1	Durbl	-0.417903	1.649374
2	Manuf	0.160494	1.167929
3	Enrgy	0.504485	0.965527
4	HiTec	-0.064024	1.132387
5	Telcm	0.194348	0.901721
6	Shops	0.274093	0.829515
7	Hlth	0.236968	0.675890
8	Utils	0.446523	0.537009
9	Other	-0.387508	1.206992

Sharpe Ratio: $S_i = \frac{E(\tilde{R}_i - R_f)}{\sqrt{Var(\tilde{R}_i - R_f)}}$

In [9]:
```
# SHarpe ratio
res = pd.DataFrame()
S_i = data[industry_columns].mean() / data[industry_columns].std()
res["Sharpe ratio"] = S_i.values
```

Sortino Ratio: $St_i = \frac{E(\tilde{R}_i - \tilde{R}_t)}{\sqrt{SV(\tilde{R}_i;\tilde{R}_t)}}$

In [10]:
```
# Sortino ratio
St_i = data[industry_columns].mean() / np.sqrt((data[industry_columns].ap
res["Sortino ratio"] = St_i.values
```

Treynor Ratio: $T_i = \frac{E(\tilde{R}_i - R_f)}{\beta_i}$

In [11]:
```
# Treynor ratio
T_i = data[industry_columns].mean() / df1["Market risk"].values
res["Treynor ratio"] = T_i.values
```

Jensen's α : $\alpha_i = E(\tilde{R}_i - R_f) - \beta_i E(\tilde{R}_m - R_f)$

In [12]:
```
# Jensen's α
J_alpha = data[industry_columns].mean() - df1["Market risk"].values * dat
res["Jensen's α"] = J_alpha.values
# res
```

In [13]:
```
# Fama-French 3-factor
res["Fama-French 3-factor alpha"] = df["Fama-French 3-factor alpha"]
res.index = industry_columns
res
```

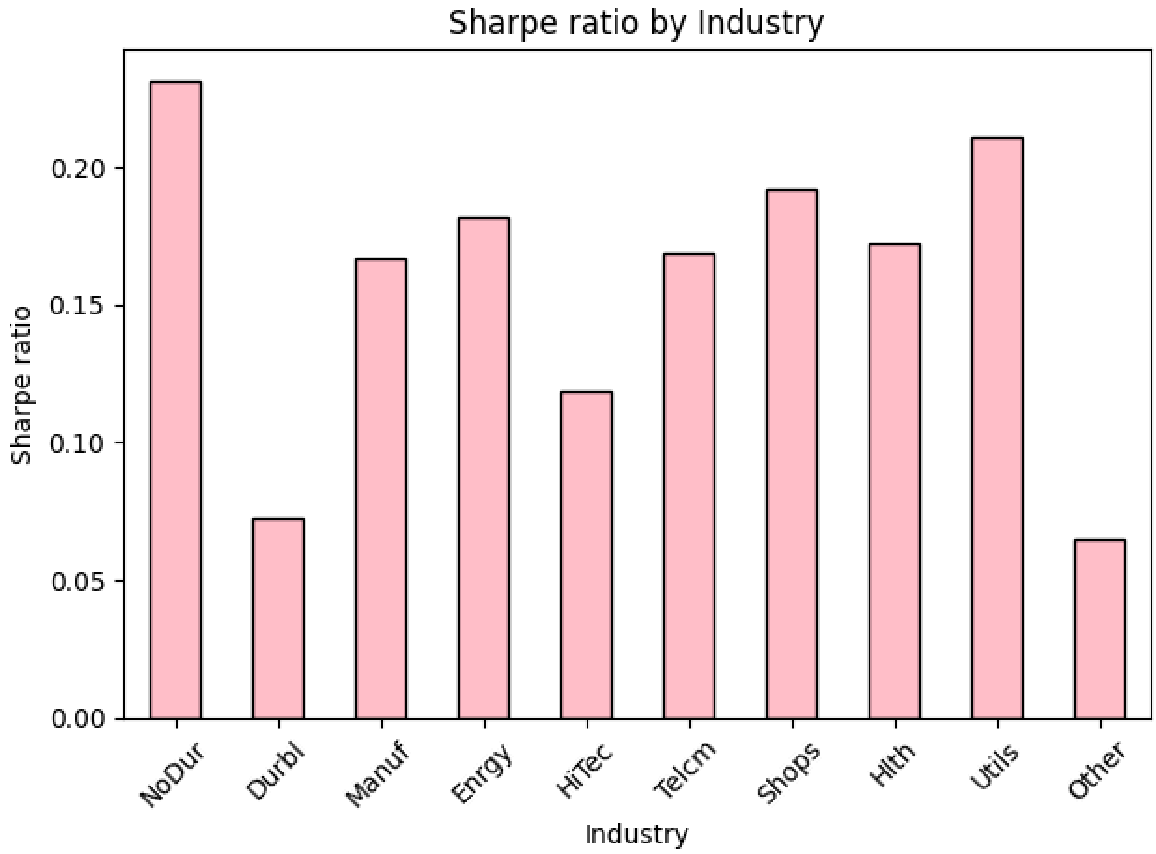
Out[13]:

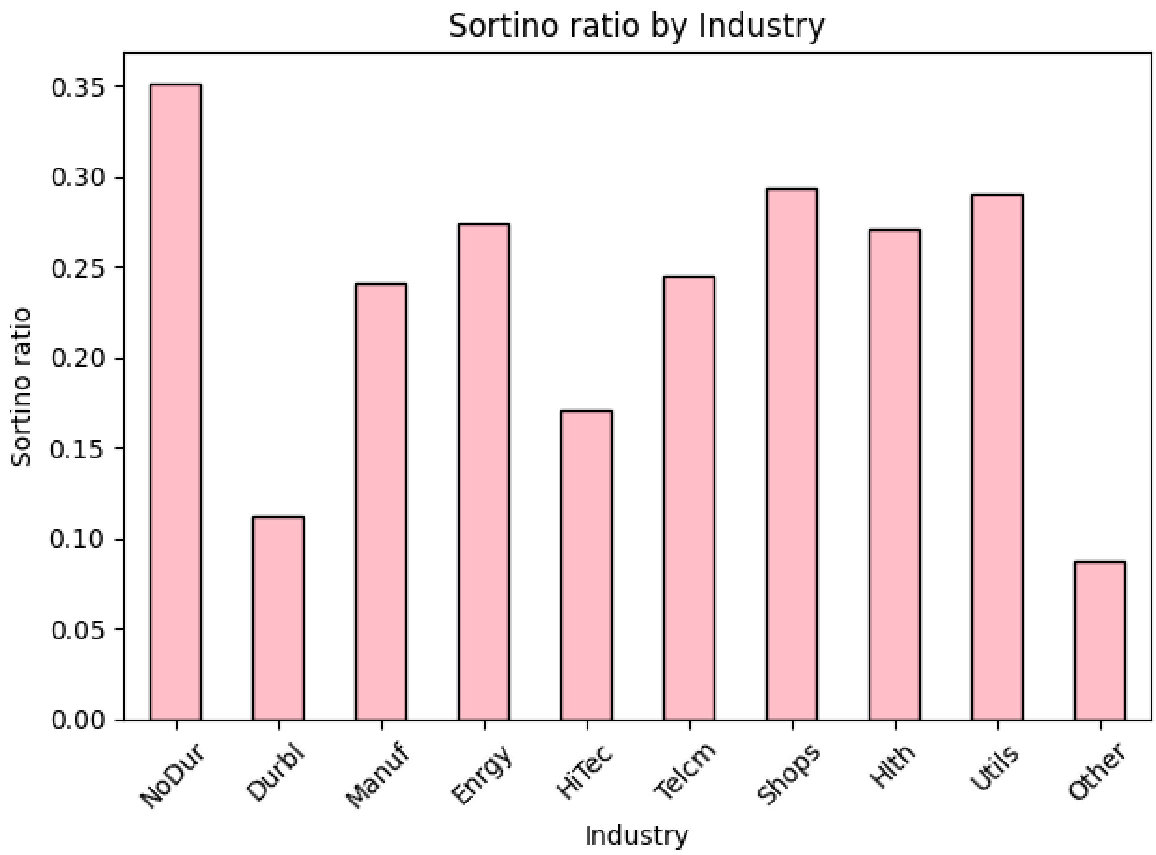
	Sharpe ratio	Sortino ratio	Treynor ratio	Jensen's α	Fama-French 3-factor alpha
NoDur	0.231099	0.350804	1.186372	0.369717	0.386704
Durbl	0.072356	0.111967	0.367463	-0.417903	-0.474342
Manuf	0.166616	0.241260	0.758251	0.160494	0.153285
Enrgy	0.181708	0.273612	1.143330	0.504485	0.523007
HiTec	0.118552	0.170620	0.564295	-0.064024	-0.065979
Telcm	0.169064	0.244940	0.836363	0.194348	0.200724
Shops	0.191753	0.293032	0.951258	0.274093	0.255941
Hlth	0.172529	0.270294	0.971435	0.236968	0.257472
Utils	0.210948	0.290044	1.452334	0.446523	0.474411
Other	0.064693	0.087351	0.299781	-0.387508	-0.404412

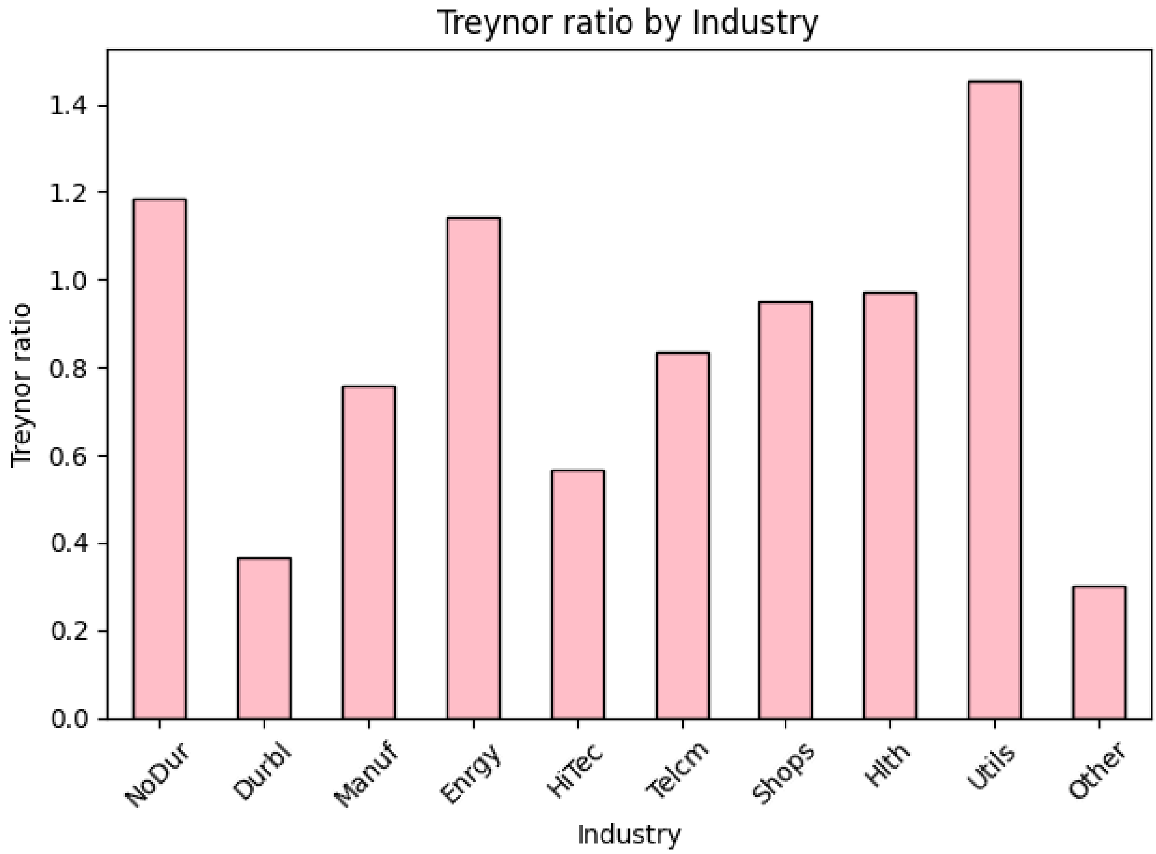
In [14]:

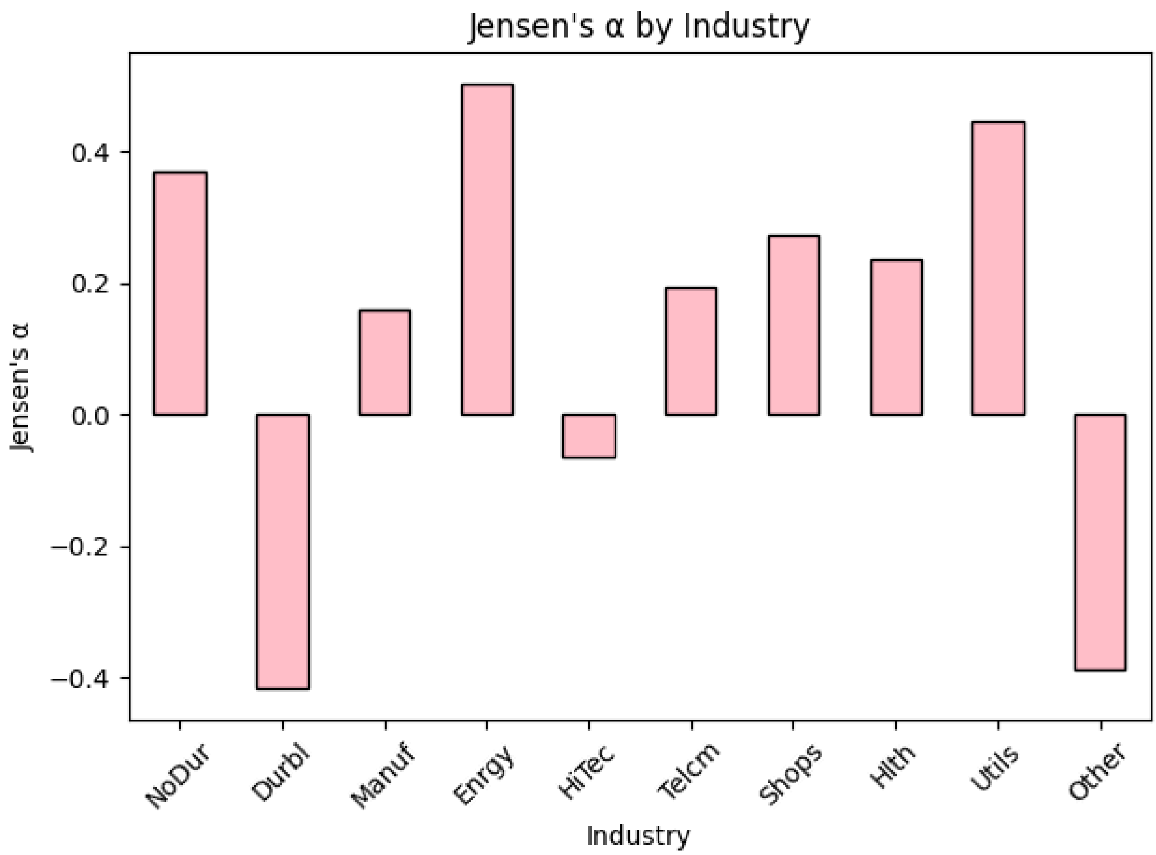
```python
performance_metrics = ["Sharpe ratio", "Sortino ratio", "Treynor ratio",


plt.figure(figsize=(10, 6))

for metric in performance_metrics:
    plt.figure()
    res[metric].plot(kind='bar', color='pink', edgecolor='black')
    plt.title(f'{metric} by Industry')
    plt.xlabel('Industry')
    plt.ylabel(metric)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

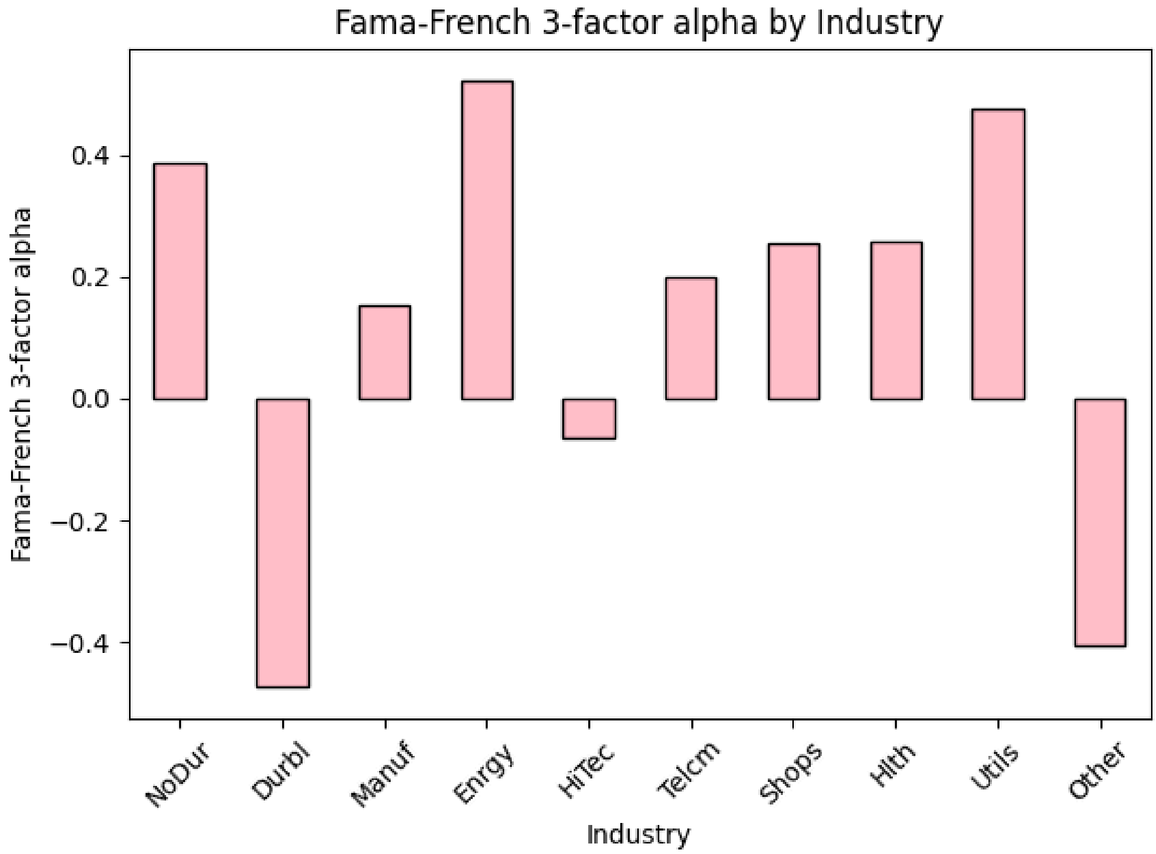
```
<Figure size 1000x600 with 0 Axes>
```


Sharpe ratio by Industry
Sharpe ratio
0.00
0.05
0.10
0.15
0.20
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other
Industry


Sortino ratio by Industry
Sortino ratio
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other
Industry


Treynor ratio by Industry
Treynor ratio
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other
Industry


Jensen's α by Industry
Jensen's α
0.4
0.2
0.0
−0.2
−0.4
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other
Industry



Explanation

The Sharpe ratio helps investors assess the return per unit of total risk. The pricing implication is that returns in proportion to risk may be appropriately priced.

The Sortino ratio focuses on downside risk, measuring performance in negative risk situations. The pricing implication is that assets with well-controlled downside risk may be appropriately priced.

The Treynor ratio evaluates the risk-return for each unit of market risk. The pricing implication is that assets providing sufficient compensation for market risk will likely be priced appropriately.