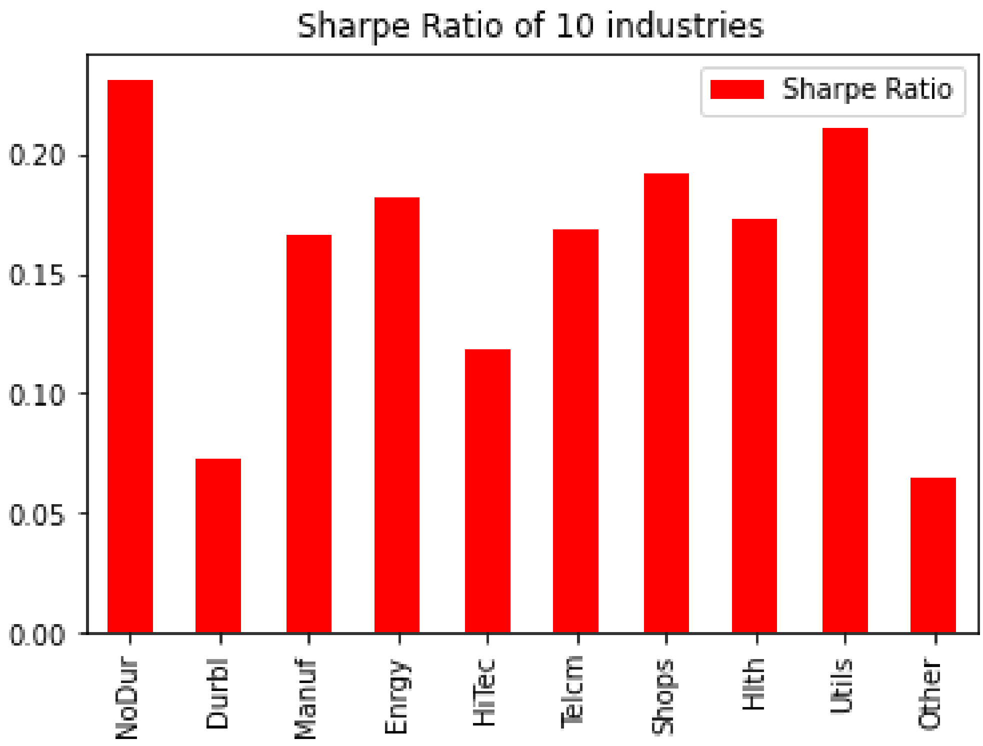
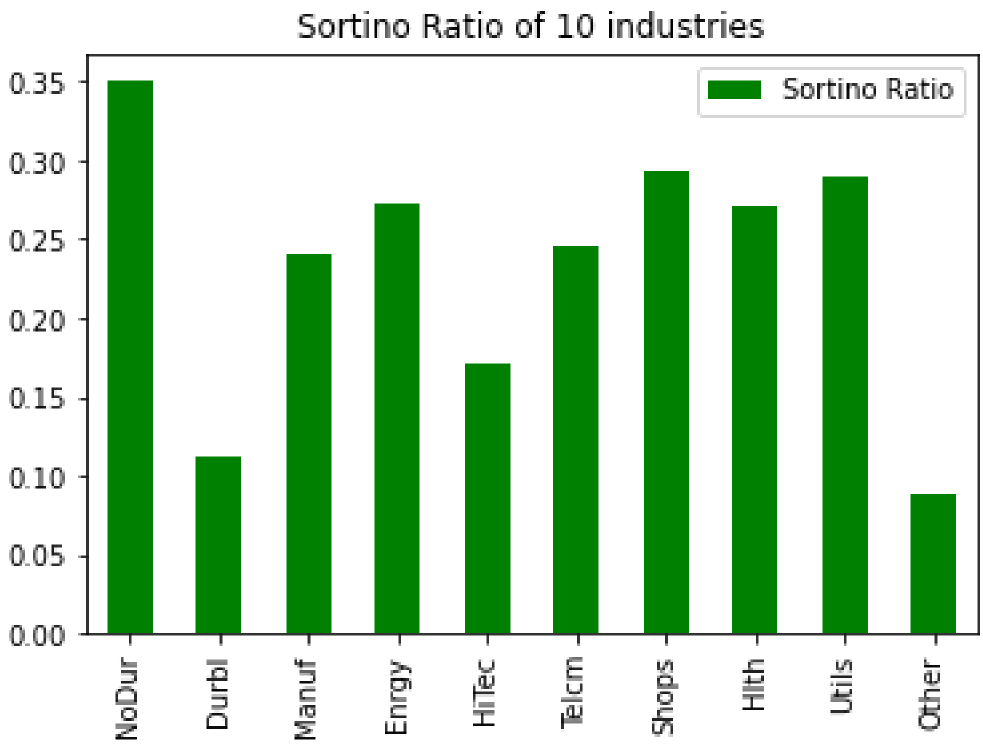
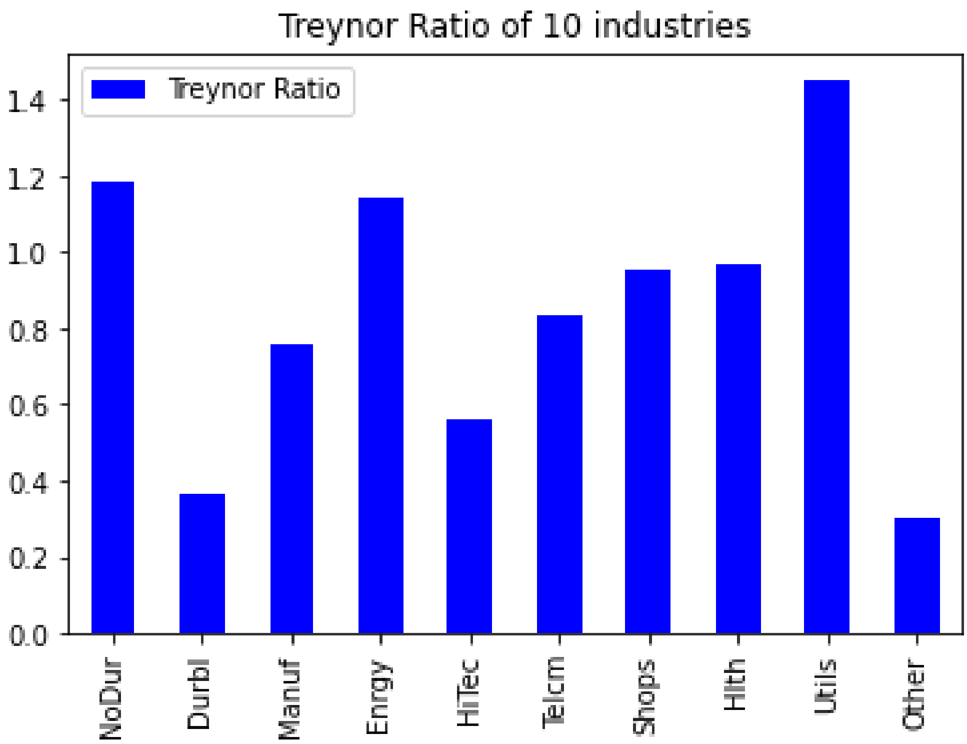
1) Create a table showing the performance metrics for the ten industry portfolios.

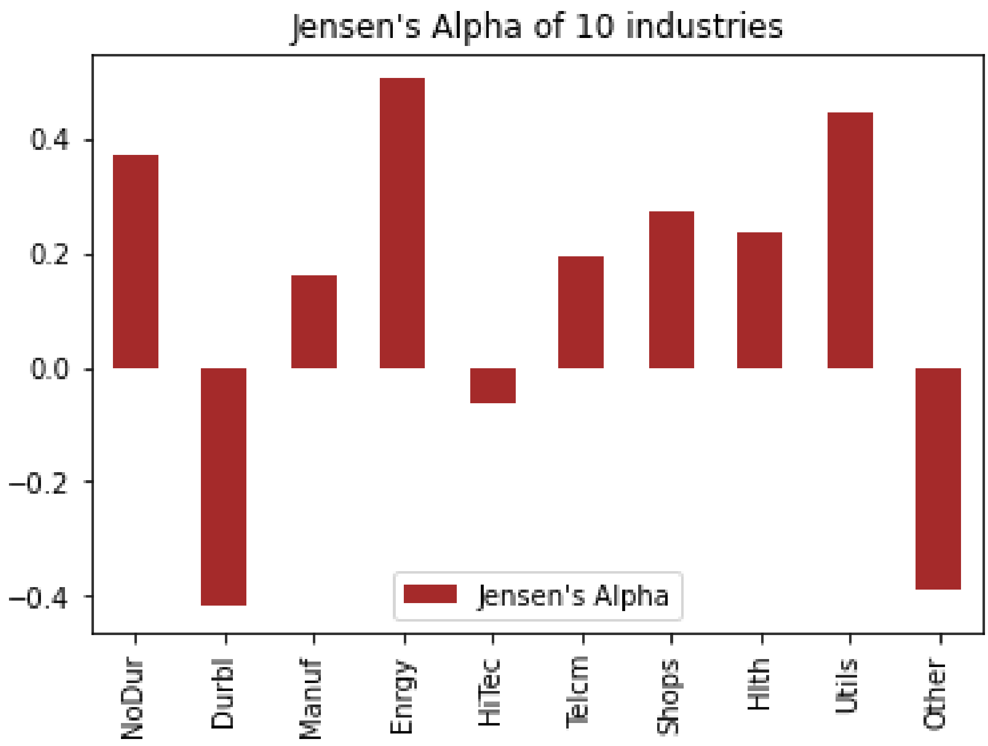
	Sharpe Ratio	Sortino Ratio	Treynor Ratio	Jensen's Alpha	Three-Factor Alpha
NoDur	0.231099	0.350804	1.186372	0.369717	0.386704
Durbl	0.072356	0.111967	0.367463	-0.417903	-0.474342
Manuf	0.166616	0.241260	0.758251	0.160494	0.153285
Enrgy	0.181708	0.273612	1.143330	0.504485	0.523007
HiTec	0.118552	0.170620	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

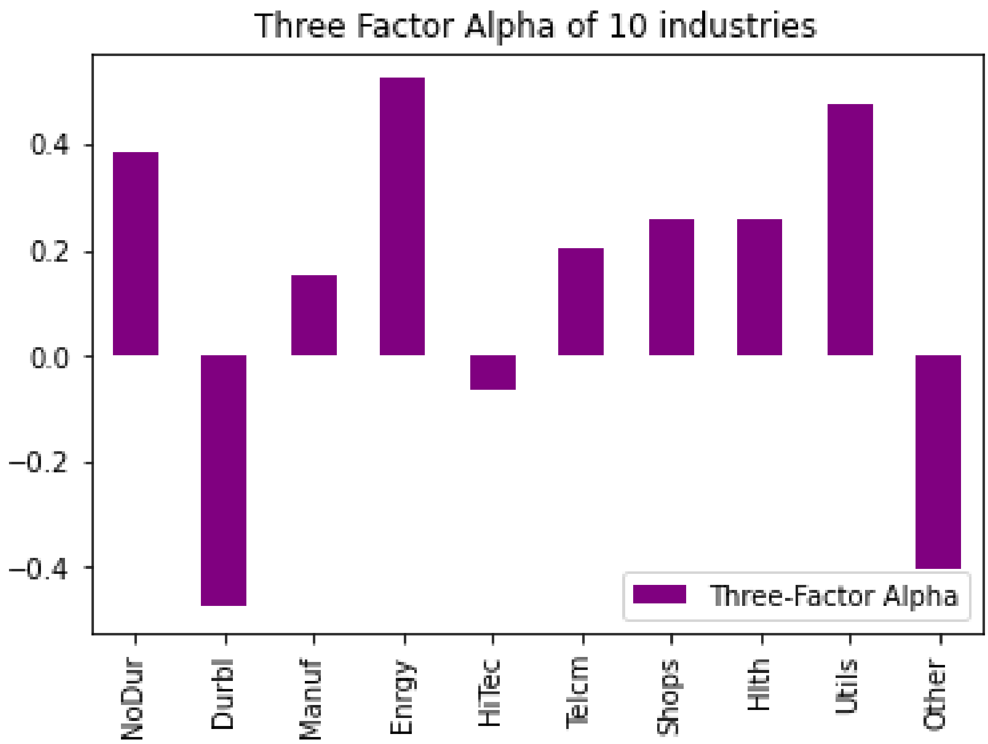
2) Plot your results as a bar chart for each performance metric.




Sortino Ratio of 10 industries
Sortino Ratio
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other


Treynor Ratio of 10 industries
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


Jensen's Alpha of 10 industries
0.4
0.2
0.0
−0.2
−0.4
Jensen's Alpha
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other


Three Factor Alpha of 10 industries
0.4
0.2
0.0
−0.2
−0.4
Three-Factor Alpha
NoDur
Durbl
Manuf
Enrgy
HiTec
Telcm
Shops
Hlth
Utils
Other

3) Briefly explain the economic significance of each of the three performance ratios (but not α's).

economic significance of Sharpe Ratio

The Sharpe Ratio compares the return of the investment with its risk. The Sharpe ratio divide its excess returns by volatility to measure risk adjusted performance. Excess returns are those more than risk-free rate or industry benchmark. A higher Sharpe Ratio is considered good when comparing against similar portfolios. The metric is less effective when comparing individual portfolio with diversified portfolios with idiosyncratic risk well hedged, or when companies whose return distributions are not well modelled by normal distribution.

economic significance of Sortino Ratio

The Sortino ratio difference from the Sharpe ratio by only taking in downside or negative volatility from total volatility by dividing excess return by the downside deviation instead of using the total standard deviation of a portfolio. The Sortino Ratio is useful in a way for investors and analysts to assess a portfolio return for a given level of bad risk. As it only focuses on the negative deviation of a portfolio return from the mean, it gives a better representation of portfolio’s risk adjusted performance since positive volatility is a benefit.

economic significance of Treynor ratio

The Treynor Ratio measures the portfolio’s excess return for investors in taking per additional unit of systematic risk.

The Treynor Ratio is like the Sharpe ratio in a way that Sharpe ratio uses a portfolio’s standard deviation to adjust the portfolio returns while Treynor Ratio uses Beta. A portfolio is a better investment if it has a higher Treynor Ratio. The Treynor ratio can be used to compare performance of individual portfolio with well-diversified portfolio, as only the systematic risk would be taken into consideration as they will not be reduced through diversification.

Appendix:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 15 17:27:24 2022
@author: XuebinLi
"""
import math
from sklearn.linear_model import LinearRegression
from numpy.linalg import inv
from tabulate import tabulate
import pandas as pd
from datetime import timedelta
from datetime import date
import datetime
from matplotlib.dates import DateFormatter, MinuteLocator
import matplotlib.pyplot as plt
import numpy as np
import glob
import warnings
import CAPM_latest as CAPM
warnings.simplefilter("ignore", UserWarning)
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
df_industries = pd.read_excel(
'C:\\Users\\lixue\\OneDrive\\Desktop\\smu\\MQF\\Asset Pricing\\lesson4\\Industry_Portfolios.xlsx')
df_riskfactors = pd.read_excel(
'C:\\Users\\lixue\\OneDrive\\Desktop\\smu\\MQF\\Asset Pricing\\lesson4\\Risk_Factors.xlsx')
#CAPM BETA from lesson 3
capm_beta = CAPM.capm_beta(CAPM.df_industry,CAPM.df_market,CAPM.rf_rate)
def std_mean_industries():
# excess returns for 10 portfolios
df_industries_excess_returns = df_industries.sub(
df_riskfactors['Rf'].values, axis=0)
# Remove date in columns
if('Date' in df_industries_excess_returns.columns or 'date' in df_industries_excess_returns):
df_industries_excess_returns = df_industries_excess_returns.drop('Date', axis=1)
# mean of excess returns
excess_returns = df_industries_excess_returns
mean_returns_industries_excess_returns = df_industries_excess_returns.mean()
# convert mean returns of industries to 2d arrays
mean_returns_industries_excess_returns = np.array(
[mean_returns_industries_excess_returns.tolist()])
# std of industries
std_returns_industries_excess_returns = df_industries_excess_returns.std()
# convert std to 2d arrays
std_returns_industries_excess_returns = np.array(
[std_returns_industries_excess_returns.tolist()])
return mean_returns_industries_excess_returns, std_returns_industries_excess_returns, excess_returns
def excess_market_return(df_riskfactors):
excess_market_returns = df_riskfactors['Rm-Rf'].mean()
excess_market_returns_without_mean = df_riskfactors['Rm-Rf']
excess_market_returns_without_rf_rate = df_riskfactors['Rm-Rf'] + df_riskfactors['Rf']
#convert to 2d array
excess_market_returns = np.array([[excess_market_returns]])
SMB = df_riskfactors['SMB']
HML = df_riskfactors['HML']
return excess_market_returns, excess_market_returns_without_mean, SMB, HML,
excess_market_returns_without_rf_rate
def sharpe_ratio(df_industries, df_riskfactors, mean_returns_industries_excess_returns,
std_returns_industries_excess_returns):
```

```
sharpe_ratio_industries = mean_returns_industries_excess_returns /
std_returns_industries_excess_returns
sharpe_ratio_industries = sharpe_ratio_industries.reshape(10,1)
return sharpe_ratio_industries
def downside_risk(mean_returns_industries_excess_returns,excess_market_return):
min_ri_minus_rt = mean_returns_industries_excess_returns - excess_market_return
min_ri_minus_rt[min_ri_minus_rt>=0] = 0
downside_risk = np.minimum(0,min_ri_minus_rt)**2
return downside_risk
#wrong answers
def sortino_ratio(rp_minus_rf,rm_minus_rf):
down_risk = np.mean(np.minimum(rp_minus_rf,0)**2)
sortino = np.mean(rp_minus_rf)/np.sqrt(down_risk)
sortino = np.array(sortino)
return sortino
def jenson_alpha(mean_returns_industries_excess_returns,excess_market_return):
mean_returns_industries_excess_returns = pd.DataFrame(mean_returns_industries_excess_returns)
excess_market_return = pd.DataFrame(excess_market_return)
reg = LinearRegression().fit(excess_market_return, mean_returns_industries_excess_returns)
beta = reg.coef_
alpha = reg.intercept_
# print(alpha.ndim)
alpha = alpha.reshape(10,1)
return alpha
def three_factor_alpha(market_risk,SMB,HML,portfolio_excess_returns):
market_risk = pd.DataFrame(market_risk)
SMB = pd.DataFrame(SMB)
HML = pd.DataFrame(HML)
portfolio_excess_returns = pd.DataFrame(portfolio_excess_returns)
df_all = pd.DataFrame()
df_all['SMB'] = SMB
df_all['HML'] = HML
df_all['market_risk'] = market_risk
reg = LinearRegression().fit(df_all,portfolio_excess_returns)
alpha = reg.intercept_
alpha = alpha.reshape(10,1)
return alpha
def treynor_ratio(excess_market_return,excess_returns):
all_port_excess_returns = pd.DataFrame(excess_returns)
excess_market_return = pd.DataFrame(excess_market_return)
reg = LinearRegression().fit(excess_market_return,all_port_excess_returns)
beta = reg.coef_
excess_return_mean = np.array(np.mean(excess_returns))
print(excess_return_mean)
print(beta)
treynor_ratio = np.divide(excess_return_mean.reshape(10,1),beta.reshape(10,1))
return treynor_ratio
def plot_chart(capm_industry):
treynor_plot = treynor_ratio(excess_market_return(df_riskfactors)[1],std_mean_industries()[2])
jenson_alpha_plot = jenson_alpha(std_mean_industries()[2],excess_market_return(df_riskfactors)[1])
three_factor_plot =
three_factor_alpha(excess_market_return(df_riskfactors)[1],excess_market_return(df_riskfactors)
[2],excess_market_return(df_riskfactors)[3],std_mean_industries()[2])
sharpe_ratio_plot = sharpe_ratio(df_industries, df_riskfactors,
std_mean_industries()[0], std_mean_industries()[1])
sortino_ratio_plot = sortino_ratio(std_mean_industries()[2],excess_market_return(df_riskfactors)[1])
plot_all_ratios =
pd.DataFrame(np.concatenate((sharpe_ratio_plot.reshape(10,1),sortino_ratio_plot.reshape(10,1), \
treynor_plot.reshape(10,1), jenson_alpha_plot.reshape(10,1),
three_factor_plot.reshape(10,1)),axis=1), \
index = capm_industry.columns,
columns = ['Sharpe Ratio','Sortino Ratio','Treynor Ratio','Jensen\'s Alpha','Three-Factor Alpha'] )
print(plot_all_ratios)
```

```
plot_all_ratios.plot(y= ["Sharpe Ratio"], kind = "bar", color = 'red', title='Sharpe Ratio of 10 industries')
plot_all_ratios.plot(y=["Sortino Ratio"], kind = "bar", color = 'green', title='Sortino Ratio of 10 industries')
plot_all_ratios.plot(y=['Treynor Ratio'], kind = 'bar', color = 'blue', title = 'Treynor Ratio of 10 industries')
plot_all_ratios.plot(y=["Jensen's Alpha"], kind = "bar", color = 'brown', title = "Jensen's Alpha of 10
industries")
plot_all_ratios.plot(y=["Three-Factor Alpha"], kind = "bar", color = 'purple', title = 'Three Factor Alpha of 10
industries')
#call functions
plot_chart(CAPM.df_industry)
```