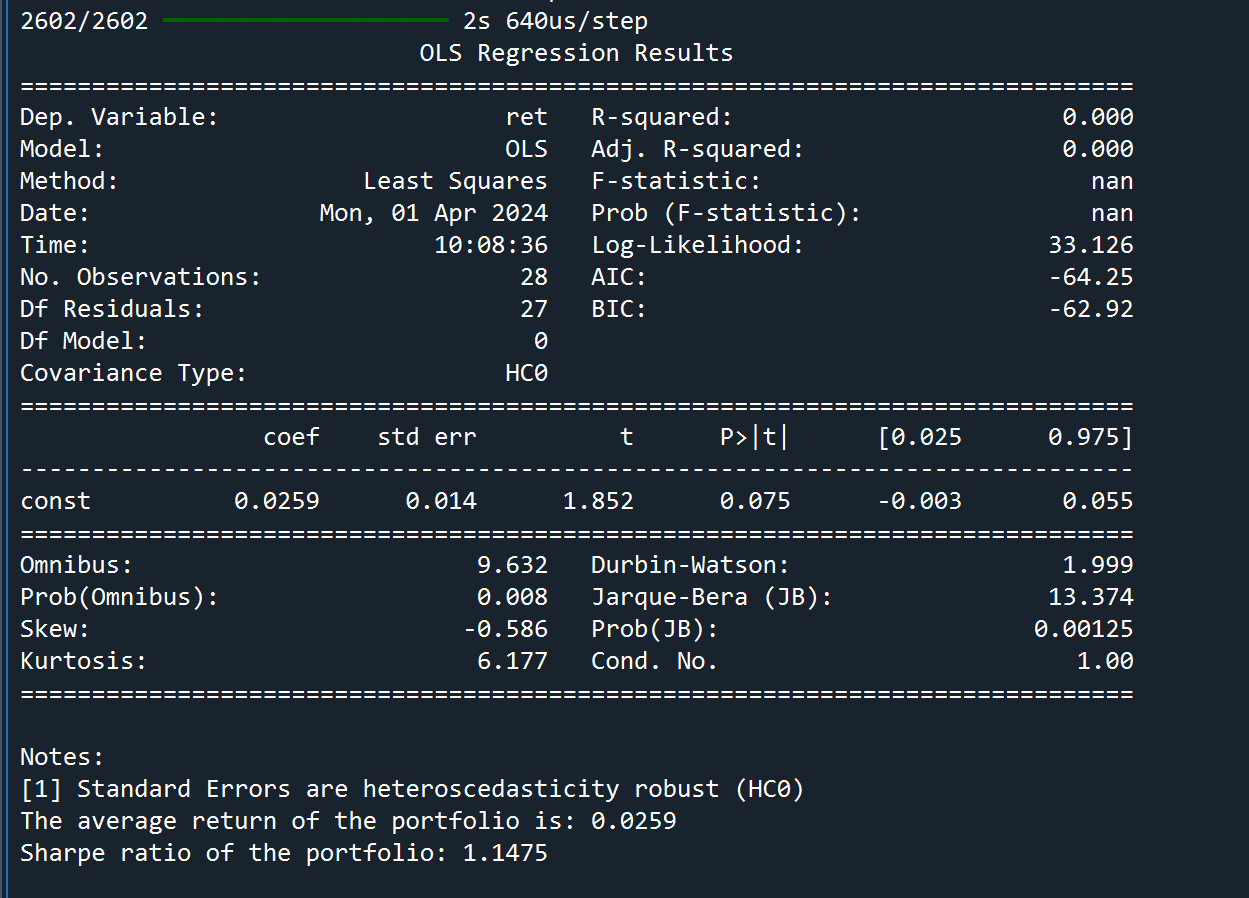
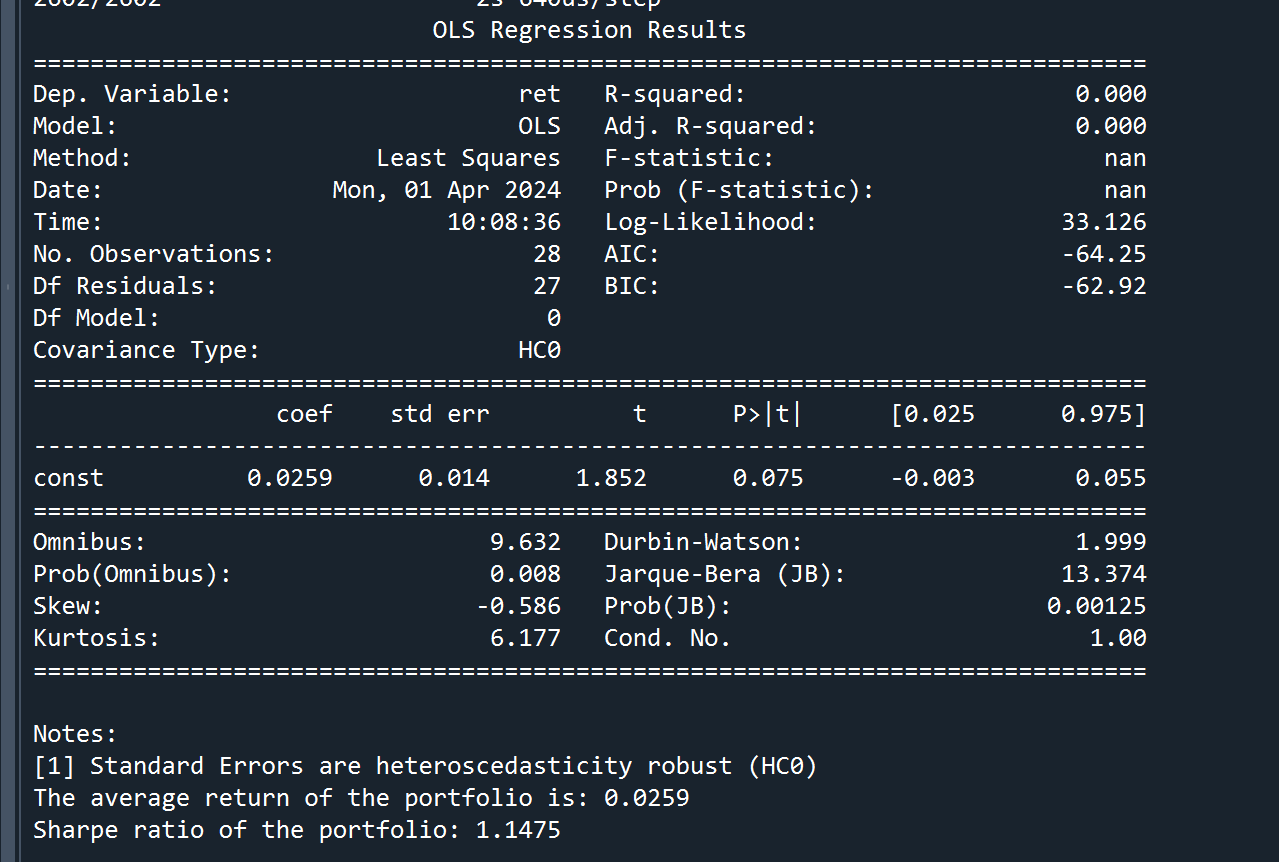
***Robert Gorman***

***Homework – Week 10***

Problem 3:



Problem 4:



Code below:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.dates as dates

import statsmodels.api as sm

plt.rcParams['figure.figsize'] = [20, 15]

sample1 = pd.read\_stata(r"C:\Users\rdg83\OneDrive - Rutgers University\Course Investment Portfolio Management\Week 9 Stuff\finalsample.dta")

sample1.sort\_values(by=['datadate'], inplace=True)

var\_remove = ['PE', 'BM']

sample2 = sample1.drop(var\_remove, axis=1)

sample2['Year']=sample2['datadate'].dt.year

sample2['Month']=sample2['datadate'].dt.month

sample2=sample1[sample1['lagPrice2']>=5]#remove penny stocks

sample2['Year']=sample2['datadate'].dt.year

sample2['Month']=sample2['datadate'].dt.month

#set gvkey and datadate as the index

sample2=sample2.set\_index(['gvkey','datadate'])

#split training and testing samples

Train1=sample2[sample2['Year']<2019]

Test1=sample2[sample2['Year']>=2019]

X\_train=Train1[['lagRet2','loglagVOL2','loglagPrice2', 'loglagMV2','lagShareturnover2','lagRet2\_sic',

'lagRet12','loglagVOL12','lagShareturnover12','lagRet12\_std','lagRet12\_min',

'lagRet12\_max','lagRet12\_sic','epspiq','dvpspq','sale','div\_p', 'cash',

'debt','logatq',

'sp500\_ret\_d','nasdaq\_ret\_d','r2000\_ret\_d','dollar\_ret\_d','VIX',

'yield\_3m','yield\_10y','gdp\_growth','Bull\_ave','Bull\_Bear']]

Y\_train=Train1[['ret']]

X\_test=Test1[['lagRet2','loglagVOL2','loglagPrice2', 'loglagMV2','lagShareturnover2','lagRet2\_sic',

'lagRet12','loglagVOL12','lagShareturnover12','lagRet12\_std','lagRet12\_min',

'lagRet12\_max','lagRet12\_sic','epspiq','dvpspq','sale','div\_p', 'cash',

'debt','logatq',

'sp500\_ret\_d','nasdaq\_ret\_d','r2000\_ret\_d','dollar\_ret\_d','VIX',

'yield\_3m','yield\_10y','gdp\_growth','Bull\_ave','Bull\_Bear']]

Y\_test=Test1[['ret']]

Factor = pd.read\_excel(r"C:\Users\rdg83\OneDrive - Rutgers University\Course Investment Portfolio Management\Week 5 Stuff\Factors-1.xlsx")

rf1 = pd.read\_excel(r"C:\Users\rdg83\OneDrive - Rutgers University\Course Investment Portfolio Management\Week 7 Stuff\Treasury bill.xlsx")

rf1['rf']=rf1['DGS3MO']/1200

rf2=rf1[['Date','rf']].dropna()

rf2['Year']=rf2['Date'].dt.year

rf2['Month']=rf2['Date'].dt.month

rf3=rf2[['Year','Month','rf']].groupby(['Year','Month'], as\_index=False).mean()

indexret1=pd.read\_stata(r"C:\Users\rdg83\OneDrive - Rutgers University\Course Investment Portfolio Management\Week 9 Stuff\Index return-1.dta")

"""PROBLEM #3"""

"""Build a neural network with one hidden layer and 20 neurons in the hidden layer. Set batch

size=10,000. Feel free to pick the values for other hyperparameters for this shallow network (e.g.,

epochs, kernel\_initializer, etc.). Train this neural network and use the trained neural network to

predict returns based on your new testing sample. Report the average return of the portfolio that

consists of the 100 stocks with the highest predicted returns in each year-month. Also, report the

Sharpe ratio of the portfolio """

#pip install tensorflow

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

import numpy as np

import pandas as pd

import statsmodels.api as sm

#build neural network using a function

def network1():

model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(20, input\_dim=X\_train.shape[1], kernel\_initializer='glorot\_uniform', activation='relu'))

model.add(tf.keras.layers.Dense(1))

model.compile(loss='mean\_squared\_error', optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001))

return model

#define the model after

network\_m = network1()

#set batch size to 10,000, pick any hyperprams

batch\_size = 10000

epochs = 50

network\_m.fit(X\_train, Y\_train, batch\_size=batch\_size, epochs=epochs, verbose=1)

#predict returns on the test set

Y\_predict = pd.DataFrame(network\_m.predict(X\_test), columns=['Y\_predict'])

Y\_test1 = pd.DataFrame(Y\_test).reset\_index()

Comb1 = pd.merge(Y\_test1, Y\_predict, left\_index=True, right\_index=True, how='inner')

Comb1['Year'] = Comb1['datadate'].dt.year

Comb1['Month'] = Comb1['datadate'].dt.month

#rank stocks by predicted returns in each year-month

rank1 = Comb1[['Y\_predict', 'Year', 'Month']].groupby(['Year', 'Month'], as\_index=False).rank(ascending=False)

rank1.rename(columns={'Y\_predict': 'Y\_predict\_rank'}, inplace=True)

stock\_long1 = pd.merge(Comb1, rank1, left\_index=True, right\_index=True)

stock\_long2 = stock\_long1[stock\_long1['Y\_predict\_rank'] <= 100]

#calculate the real returns on selected stocks

stock\_long3 = stock\_long2[['ret', 'Year', 'Month']].groupby(['Year', 'Month']).mean()

#merge with risk-free rate and index return

stock\_long4 = pd.merge(stock\_long3, rf3, left\_on=['Year', 'Month'], right\_on=['Year', 'Month'], how='left')

stock\_long5 = pd.merge(stock\_long4, indexret1, left\_on=['Year', 'Month'], right\_on=['Year', 'Month'], how='left')

#calculate excess returns

stock\_long5['ret\_rf'] = stock\_long5['ret'] - stock\_long5['rf']

stock\_long5['ret\_sp500'] = stock\_long5['ret'] - stock\_long5['sp500\_ret\_m']

#perform robust regression analysis

stock\_long5 = sm.add\_constant(stock\_long5)

model\_summary = sm.OLS(stock\_long5[['ret']], stock\_long5[['const']]).fit().get\_robustcov\_results(cov\_type='HC0').summary()

print(model\_summary)

#report the average return of the portfolio that consists of the 100 stocks with the highest predicted returns in each year-month

avg\_return = stock\_long5['ret'].mean()

print(f"The average return of the portfolio is: {avg\_return:.4f}")

#report the Sharpe ratio of the portfolio

Ret\_rf = stock\_long5[['ret\_rf']]

SR = (Ret\_rf.mean()[0] / Ret\_rf.std()[0]) \* np.sqrt(12)

print(f"Sharpe ratio of the portfolio: {SR:.4f}")

"""PROBLEM 4"""

"""Build a deep neural network with more than 2 hidden layers. Feel free to pick the number of

hidden layers and the number of neurons in each hidden layer. Also, pick your own values for

other hyperparameters for this deep neural network (e.g., epochs, batch size, kernel\_initializer,

etc.). Train this deep neural network and use the trained deep neural network to predict returns

based on your new testing sample. Report the average return of the portfolio that consists of the

100 stocks with the highest predicted returns in each year-month. Also, report the Sharpe ratio of

the portfolio."""

#build a deep neural network with more than 2 hidden layers

#mine has 3 hidden layers (64,32,16)

def deep\_network1():

model = Sequential()

model.add(Dense(64, input\_dim=X\_train.shape[1], kernel\_initializer='glorot\_uniform', activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(16, activation='relu'))

model.add(Dense(1))

model.compile(loss='mean\_squared\_error', optimizer=Adam(lr=0.001))

return model

#Train - kept batch size at 10,000 and epochs at 50

deep\_m = deep\_network1()

deep\_m.fit(X\_train, Y\_train, epochs=50, batch\_size=10000, verbose=1)

#predict returns based on the trained model

Y\_predict = pd.DataFrame(deep\_m.predict(X\_test), columns=['Y\_predict'])

Y\_test1 = pd.DataFrame(Y\_test).reset\_index()

Comb1 = pd.merge(Y\_test1, Y\_predict, left\_index=True, right\_index=True, how='inner')

Comb1['Year'] = Comb1['datadate'].dt.year

Comb1['Month'] = Comb1['datadate'].dt.month

#rank stocks by predicted returns in each year-month

rank1 = Comb1[['Y\_predict', 'Year', 'Month']].groupby(['Year', 'Month'], as\_index=False).rank(ascending=False)

rank1.rename(columns={'Y\_predict': 'Y\_predict\_rank'}, inplace=True)

stock\_long1 = pd.merge(Comb1, rank1, left\_index=True, right\_index=True)

stock\_long2 = stock\_long1[stock\_long1['Y\_predict\_rank'] <= 100]

#calculate the real returns on selected stocks

stock\_long3 = stock\_long2[['ret', 'Year', 'Month']].groupby(['Year', 'Month']).mean()

#merge with risk-free rate and index return

stock\_long4 = pd.merge(stock\_long3, rf3, left\_on=['Year', 'Month'], right\_on=['Year', 'Month'], how='left')

stock\_long5 = pd.merge(stock\_long4, indexret1, left\_on=['Year', 'Month'], right\_on=['Year', 'Month'], how='left')

#calculate excess returns

stock\_long5['ret\_rf'] = stock\_long5['ret'] - stock\_long5['rf']

stock\_long5['ret\_sp500'] = stock\_long5['ret'] - stock\_long5['sp500\_ret\_m']

#report the average return of the portfolio that consists of the100 stocks with the highest predicted returns in each year-mont

avg\_return = stock\_long5['ret'].mean()

print(f"The average return of the portfolio is: {avg\_return:.4f}")

#report the Sharpe ratio of the portfolio.

Ret\_rf = stock\_long5[['ret\_rf']]

SR = (Ret\_rf.mean()[0] / Ret\_rf.std()[0]) \* np.sqrt(12)

print(f"Sharpe ratio of the portfolio: {SR:.4f}")