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]

#LOAD DATA

sample1 = pd.read\_stata(r"C:\Users\rdg83\OneDrive - Rutgers University\Course Investment Portfolio Management\Week 7 Stuff\1finalsample.dta")

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

#PROBLEM 1 - removing two independent variables from finalsample.dta dataset

#Removed 'Bull\_ave' and 'Bull\_Bear'

var\_remove = ['Bull\_ave', 'Bull\_Bear']

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

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'])

#PROBLEM 2 - Split new dataset into training and testing samples

#Changed years to 2018 to help lighten the data load

Train1=sample2[sample2['Year']<2018] #feel free to use another year to split the sample.

Test1=sample2[sample2['Year']>=2018]

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

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

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

'debt','logatq',

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

'yield\_3m','yield\_10y','gdp\_growth']]

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','BM','div\_p','PE', 'cash',

'debt','logatq',

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

'yield\_3m','yield\_10y','gdp\_growth']]

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

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 7 Stuff\Index return-2.dta")

"""Linear regression"""

from sklearn.linear\_model import LinearRegression

LR\_m=LinearRegression() #define the model

LR\_m.fit(X\_train,Y\_train) #train the model and get coefficients on training sample

LR\_m.coef\_

coefficients\_LR=pd.DataFrame(LR\_m.coef\_).T #save all the coefficients in a dataframe

var\_lasso=X\_test.columns.tolist() #get the independent variable names

coefficients\_LR.index=var\_lasso

print (coefficients\_LR)

#predict returns based on the trained model

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

#merge the predicted returns with corresponding actual returns

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 stock based on 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)

#select the N stocks with top predicted returns in each year-month

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

#count the number of stocks selected in each month

stock\_long2['datadate'].value\_counts()

#calculate the real returns on selected stocks. equal weight

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

#merge with RF 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')

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

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

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

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

#0.0986 per month

#Sharpe Ratio

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

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

SR

#1.648

#PROBLEM 3 - Use LassoCV to run lasso regression to predict stock monthly returns

from sklearn.linear\_model import LassoCV

from sklearn.linear\_model import Lasso

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import TimeSeriesSplit

import numpy as np

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

tsplit = TimeSeriesSplit(n\_splits=5,test\_size=10000, gap=5000)

alphas = np.logspace(-4, 4, 100)

lasso\_cv = LassoCV(alphas=alphas, cv=tsplit)

lasso\_cv.fit(X\_train\_scaled, Y\_train.values.ravel())

Lasso\_m = 0.001

lasso\_final = Lasso(alpha=Lasso\_m)

lasso\_final.fit(X\_train\_scaled, Y\_train)

# Get coefficients

coefficients\_Lasso = pd.DataFrame(lasso\_final.coef\_, index=var\_lasso, columns=["Coefficient"])

print(coefficients\_Lasso)

coef\_select=coefficients\_Lasso.query("Coefficient != 0")

print (coef\_select)

#PROBLEM 4 - Use RidgeCV to run ridge regression to predict stock monthly returns.

from sklearn.linear\_model import RidgeCV

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import TimeSeriesSplit

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

tsplit=TimeSeriesSplit(n\_splits=5,test\_size=10000, gap=5000)

alpha\_candidate=np.linspace(0.01,0.1,20)

Ridge\_m = RidgeCV(alphas=alpha\_candidate, cv=tsplit)

Ridge\_m.fit(X\_train,Y\_train)#train the model

Ridge\_m.alpha\_

print('Optimal Alpha Chosen: ',Ridge\_m.alpha\_)

coefficients\_Ridge=pd.DataFrame(Ridge\_m.coef\_).T

coefficients\_Ridge.index=var\_lasso

coef\_select\_Ridge=coefficients\_Lasso.query("Coefficient != 0")

print (coefficients\_Ridge)

#PROBLEM 5 - Use ElasticNetCV to run elasticnet regression to predict stock monthly returns

from sklearn.linear\_model import ElasticNetCV

from sklearn.linear\_model import ElasticNet

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import TimeSeriesSplit

import numpy as np

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

tsplit = TimeSeriesSplit(n\_splits=5, test\_size=10000, gap=5000)

l1\_ratios = np.linspace(0.05, 0.95, 10)

elasticnet\_cv = ElasticNetCV(l1\_ratio=l1\_ratios, alphas=alphas, cv=tsplit, max\_iter=10000)

elasticnet\_cv.fit(X\_train\_scaled, Y\_train.values.ravel())

selected\_l1\_ratio = elasticnet\_cv.l1\_ratio\_

print("Optimal l1\_ratio Chosen: " , selected\_l1\_ratio)

elasticnet\_final = ElasticNet(alpha=elasticnet\_cv.alpha\_, l1\_ratio=selected\_l1\_ratio, max\_iter=10000)

elasticnet\_final.fit(X\_train\_scaled, Y\_train)

coef\_select\_Elas=coefficients\_Lasso.query("Coefficient != 0")

coefficients\_elasticnet = pd.DataFrame(elasticnet\_final.coef\_, index=var\_lasso, columns=["Coefficient"])

print(coefficients\_elasticnet)

#compare

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

#merge the predicted returns with corresponding actual returns

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 stock based on 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)

#select the N stocks with top predicted returns in each year-month

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

#count the number of stocks selected in each month

stock\_long2['datadate'].value\_counts()

#calculate the real returns on selected stocks

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

#merge with RF 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')

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

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

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

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

#0.0998

#Sharpe Ratio

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

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

SR

#1.507