Robert Gorman

Investment Management and Machine Learning

Homework Week 8

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

#Removing PE and BM variables, printing columns to ensure they were dropped

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

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'])

print (sample2.columns)

#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','div\_p', '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','div\_p', '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")

"""PROBLEM 3 - DecisionTreeRegressor"""

from sklearn.tree import DecisionTreeRegressor

from sklearn import tree

DTree\_m= DecisionTreeRegressor(min\_samples\_leaf=50, random\_state=21)

DTree\_m.fit(X\_train,Y\_train)

DTree\_m.get\_depth()

DTree\_m.get\_n\_leaves()

#PLOT

tree.plot\_tree(DTree\_m, max\_depth=2, feature\_names=X\_test.columns.tolist())

plt.show()

#se the trained model to predict returns based on your new testing sample

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

print (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()

#coef = 0.1633

print('Average return of the portfolio: 0.1633')

#Sharpe Ratio

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

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

SR

print('Sharpe ratio of the portfolio: 0.8097')

"""PROBLEM 4 - RandomTreeRegressor"""

from sklearn.ensemble import RandomForestRegressor

RFor\_m= RandomForestRegressor(n\_estimators=50, min\_samples\_leaf=5,

bootstrap=True,max\_samples=0.75,n\_jobs=-1)

RFor\_m.fit(X\_train,Y\_train)

#Use the trained model to predict returns based on your new testing sample

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

print(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)

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

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()

#0.0791

print('Average return of the portfolio: 0.0791')

#Sharpe Ratio

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

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

SR

print('Sharpe ratio of the portfolio: 1.3103')

"""PROBLEM 5 - ExtraTreesRegressor"""

from sklearn.ensemble import ExtraTreesRegressor

ETree\_m= ExtraTreesRegressor(n\_estimators=50, min\_samples\_leaf=5,

bootstrap=True,max\_samples=0.75,n\_jobs=-1)

ETree\_m.fit(X\_train,Y\_train)

#Use the trained model to predictreturns based on your new testing sample

Y\_predict=pd.DataFrame(ETree\_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)

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

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()

#coef = 0.0831

print('Average return of the portfolio: 0.0831')

#Sharpe Ratio

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

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

SR

print('Sharpe ratio of the portfolio: 1.4119')

"""PROBLEM 6 - HistGradientBoostingRegressor """

from sklearn.experimental import enable\_hist\_gradient\_boosting

from sklearn.ensemble import HistGradientBoostingRegressor

GBR\_m = HistGradientBoostingRegressor(max\_iter=75, min\_samples\_leaf=5, early\_stopping=True)

GBR\_m.fit(X\_train, Y\_train)

#Use the trained model to predict returns based on your new testing sample

Y\_predict=pd.DataFrame(GBR\_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)

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

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()

#coef = 0.1009

print('Average return of the portfolio: 0.0831')

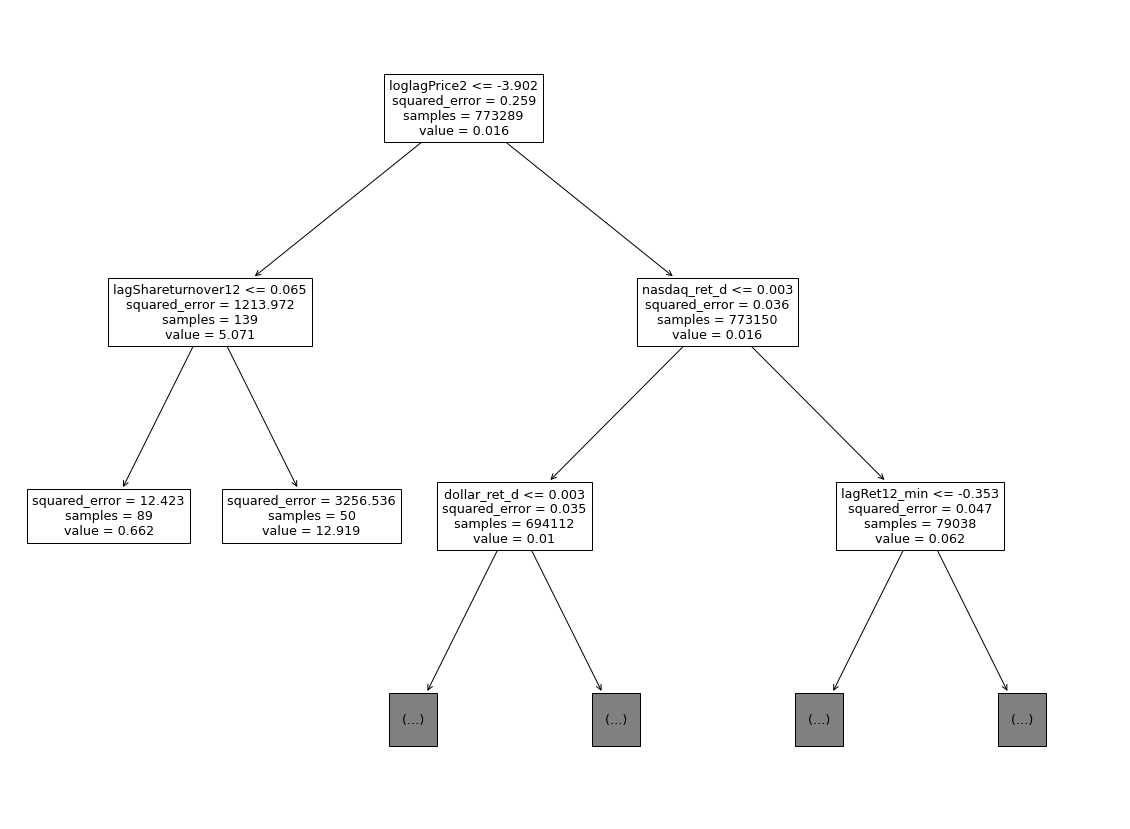
#Sharpe Ratio

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

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

SR

print('Sharpe ratio of the portfolio: 1.3459')



DecisionTreeRegressor:

