## 机器学习因子手册

《机器学习与因子投资：从基础到实践》，附录三python代码，下载数据data\_ml\_org.csv

目录：/Users/huangyinfei/ebook/量化策略/机器学习与因子投资：从基础到实践

目录：/Users/huangyinfei/pwork/mlfactor

Div\_Yld股息率

Mkt\_Cap\_3M\_Usd

Mkt\_Cap\_12M\_Usd市值

R1M\_Usd未来一个月收益

R3M\_Usd未来三个月收益

R6M\_Usd未来六个月收益

R12M\_Usd未来十二个月收益

MKT\_RF市场收益减无风险利率

SMB规模因子（小公司减大公司）

HML价值因子（高价值减低价值）

WML动量因子（赢家减输家）

RMW盈利因子（高利润减低利润）

CMA投资因子（保守减激进）

BAB低风险因子

Eps每股收益

Mom\_5M\_Usd

Mom\_11M\_Usd动量线指标

Ocf现金流

Pb市净率

Pe

Vol1Y\_Usd波动率

Vol3Y\_Usd

Fcf\_Bv

Interest\_Expense

Net\_Margin

Recurring\_Earning\_Total\_Assets

Total\_Liabilities\_Total\_Assets

Advt\_3M\_Usd

1. 数据清洗

日期字段要补0，格式转为YYYY/MM/DD，创建：date\_converter.py

定义convert转换函数

data = pd.read\_csv('data\_ml\_org.csv')

data.date=data.date.apply(convert)

data.to\_csv('data\_ml.csv')

1. 基础数据

代码：nb\_assets.py

1206支产品，按季度统计，97个因子

首先筛选2000～2018年区间的数据，绘制每个日期的资产总数，看到基本是1180左右

其次取股息率Div\_Yld因子，选择2000-02-29，绘制股息率的分布，看到基本是平均在10左右

最后计算收益，先清洗数据，保留793支股票，然后看228个时点的收益R1M\_Usd

根据筛选条件提取索引列表，用于后续直接访问

data\_raw.index[筛选条件].tolist()

分组后做统计

data\_ml.groupby('date' ).size()

data\_ml.groupby(['stock\_id' ]).count()

直接截取矩阵

data\_ml.iloc[:,3:95]

添加列：

pd.merge(data\_ml,df\_median,how='left' , on=['date' ])

分类标签：

data\_ml['R12M\_Usd\_C' ] = np.where(df['R12M\_Usd' ] > df['R12M\_Usd\_median' ], 1.0, 0.0)

把索引转为新增一列：

reset\_index()

把某一列转为索引：

set\_index('date')

把列转为索引，进行折叠：

data\_ml.pivot(index='date' ,columns='stock\_id' ,values='R1M\_Usd' )

1. 因子分析

代码：factors.py

按年份、按大/小市值做分组，取R1M\_Usd因子的均值

折叠后，按年份，大市值的平均收益、小市值的平均收益，绘制直方图

代码：FF\_factors.py

从1963年7月，到2020年3月，取6个因子，绘制月度收益率的变化曲线

代码：ols.py

对每个产品，取nb\_assets返回的收益率作为Y，取FF\_factors五因子作为X，以每月数据代入进行线性回归；注意是以N月的五因子去回归N+1月的收益率

对每个月，取收益率作为Y，取五因子作为X，以所有产品带入进行线性回归。绘制按照时间顺序，五因子系数的取值变化曲线

代码：compete.py

取FF\_factors五因子，用其中四因子去回归单一因子，得到协方差值，即所谓因子竞争

代码：motion.py

用plot\_acf绘制每个因子的自相关数据，即所谓因子动量

数据向上平移一格：

xx.shift(-1)

线性回归：

import statsmodels.api as sm

results=sm.OLS(endog=Y,exog=sm.add\_constant(data\_FM.iloc[0:227,2:7])).fit()

results.params

列顺序调整：

reg\_result = df\_res\_full\_mat.reindex(columns=column\_names\_inverted)

自相关数据：

sm.graphics.tsa.plot\_acf(FF\_factors.RMW, lags=10)

1. 数据预处理

代码：prepare.py

选择8个因子，对每一月的所有产品，计算因子之间的相关性，重点关注未来收益因子与其它因子间的相关性，最后绘制分簇散点图

绘制2个因子与未来收益因子间的阴影折线图，表现为一个平滑函数

根据产品、因子进行分组，对每个分组计算自回归系数，然后绘制直方图

代码：scale.py

首先是一个exp(sin(x))的原始序列

然后使用三种归一化方法进行数据平滑，绘制柱状图、直方图

代码：toy.py

对市值因子作归一化，并用markdown展示

用市值因子对收益率作线性回归，看上去ECDF归一化比简单归一化效果要好

协方差：

data\_corr.groupby('date' ).corr()

绘制小提琴图与分簇散点图：

sns.violinplot(x="Factors" , y="R1M\_Usd" , data=data\_corr);

sns.swarmplot(x="Factors" , y="R1M\_Usd" , data=data\_corr);

把列转为索引，进行折叠：

unpivoted\_data\_ml = pd.melt(data\_ml[['R1M\_Usd' ,'Mkt\_Cap\_12M\_Usd' ,'Vol1Y\_Usd' ]], id\_vars='R1M\_Usd' )

绘制带阴影的折线图：

sns.lineplot(data = unpivoted\_data\_ml, y='R1M\_Usd' , x='value' , hue='variable' );

自相关数据：

Xxx.apply(lambda x: x['value' ].autocorr(lag=1))

使用函数批量生成列：

data\_norm=pd.DataFrame.from\_dict(dict(

index=np.linspace(1,length,length),

norm\_0\_1=norm\_0\_1(x),

norm\_standard=norm\_standard(x),

norm\_unif=norm\_unif(x)))

从字典生成DataFrame：

pd.DataFrame.from\_dict( xxx )

1. LASSO回归

代码：lasso.py

设定一组alpha系数（小于1），作Lasso回归得到coef值，从92个因子中选出有效度最高的12个因子，绘制折线图

设定一组alpha系数（大于1），作岭回归得到coef值，对有效度最高的12个因子，绘制折线图

备注：实际上是因子筛选

对每个月，用历史月份计算3种权重，然后对所有产品进行组合计算收益，最后看收益的标准差，也就是波动率；协方差矩阵求逆、弹性网络回归两种方法得到的权重更优

备注：实际上是股票筛选

根据训练集作弹性网络回归，然后对测试集计算MSE均方差、命中率

Lasso回归：

lasso = Lasso(alpha=alpha)

lasso.fit(X\_penalized,y\_penalized)

lasso\_res[alpha] = lasso.coef\_

岭回归：

ridge = Ridge(alpha=alpha)

ridge.fit(X\_penalized,y\_penalized)

ridge\_res[alpha] = ridge.coef\_

协方差矩阵求逆：

sigma=np.cov(returns.T)+1e-2\*np.identity(N)

w = np.matmul(np.linalg.inv(sigma),np.repeat(1,N))

return w / np.sum(w)

弹性网络回归：

lr=ElasticNet(alpha=alpha,l1\_ratio=Lambda)

lr.fit(X,y)

err=y-lr.predict(X)

w=(1-np.sum(lr.coef\_))/np.var(err)

1. 决策树

代码：tree.py

以93个因子为条件，收益率为目标，训练决策树

拿训练集数据作验证

绘制3个因子与未来收益因子间的阴影折线图

用训练集训练决策树，用测试集检验，看MSE均方差、命中率

代码：forest.py

构造包含40个决策树的随机森林

用训练集训练随机森林回归，用测试集检验，看MSE均方差、命中率

用训练集训练随机森林分类，用测试集检验，看命中率

用训练集训练提升树，用测试集检验，看命中率

代码：xgb.py，跳过

决策树：

fit\_tree2 = tree.DecisionTreeRegressor(min\_samples\_split = 4000, max\_depth = 5, ccp\_alpha=0.0001, min\_samples\_leaf =1500)

fit\_tree2 = fit\_tree2.fit(X\_train, y\_train)

mse = np.mean((fit\_tree2.predict(X\_test) - y\_test)\*\*2)

hitratio = np.mean(fit\_tree2.predict(X\_test) \* y\_test > 0)

随机森林回归：

fit\_RF = RandomForestRegressor(n\_estimators = 40, criterion ='mse' , min\_samples\_split= 250, bootstrap=False,max\_features=30,max\_samples=10000)

随机森林分类：

fit\_RF\_C = RandomForestClassifier( n\_estimators = 40,criterion ='gini' ,min\_samples\_split= 250,bootstrap=False,max\_features=30,max\_samples=20000)

提升树：

fit\_adaboost\_C = AdaBoostClassifier(DecisionTreeClassifier( max\_depth=3), n\_estimators=3)

xgb提升树：跳过

1. 神经网络

代码：nn1.py

使用Tensorflow和Keras，构造双层神经网络93-16-8-1，用训练集训练网络，用测试集检验，看命中率

代码：nn2.py

使用onehot作二分类，构造神经网络93-16-8-2，训练过程中设置早停回调函数

代码：nn3.py

自定义损失函数，构造神经网络93-16-8-1

代码：rnn.py

构造循环神经网络16-1，训练输入793\*168\*92（产品-日期-因子），训练标签793\*168\*1（产品-日期-收益率）；复制模型参数

双层神经网络：

model = keras.Sequential()

model.add(layers.Dense(16, activation="relu" , input\_shape=(len(features),)))

model.add(layers.Dense(8, activation="tanh" ))

model.add(layers.Dense(1))

model.compile(optimizer='RMSprop' , loss='mse' ,metrics=['MeanAbsoluteError' ])

fit\_NN = model.fit(NN\_train\_features,NN\_train\_labels,batch\_size=256,epochs = 10, validation\_data=(NN\_test\_features,NN\_test\_labels), verbose = True)

hitratio = np.mean(model.predict(NN\_test\_features) \* NN\_test\_labels > 0)

查看模型拓扑：

model.summary()

训练过程可视化：

show\_history(fit\_NN)

onehot编码：

NN\_train\_labels\_C = to\_categorical(training\_sample['R1M\_Usd\_C' ].values)

早停回调函数：

callback=tf.keras.callbacks.EarlyStopping(monitor="val\_loss", min\_delta = 0.001, patience = 4, verbose = 0 )

自定义损失函数：

model\_custom.compile( optimizer='RMSprop' , loss=custom\_loss, metrics=['MeanAbsoluteError' ])

循环神经网络：

model\_RNN = keras.Sequential()

model\_RNN.add(layers.GRU(16, batch\_input\_shape = (nb\_stocks, nb\_dates\_train, nb\_feats), activation='tanh', return\_sequences=True))

model\_RNN.add(layers.Dense(1))

model\_RNN.compile(optimizer='RMSprop' , loss='mse' , metrics=['MeanAbsoluteError' ])

fit\_RNN = model\_RNN.fit(train\_features\_rnn, train\_labels\_rnn,epochs = 10,batch\_size = nb\_stocks, verbose=False)

复制模型参数：

new\_model.set\_weights(model\_RNN.get\_weights())

1. 支持向量机

代码：svm.py

跳过

支持向量机：

model\_svm=svm.SVR(kernel='rbf' ,C=0.1,epsilon=0.1,gamma=0.5)

fit\_svm=model\_svm.fit(x, y)

mse = np.mean((fit\_svm.predict(test\_feat\_short) - y\_test)\*\*2)