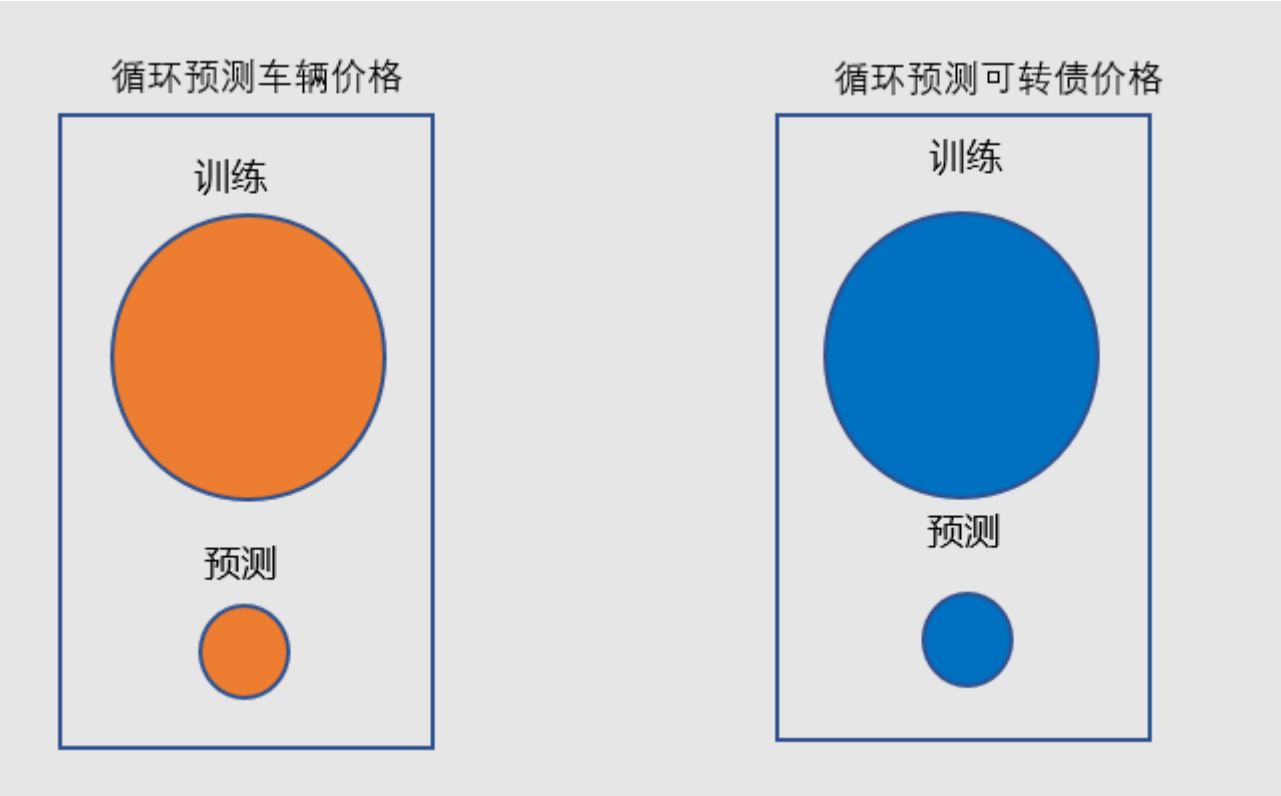
怎么使用机器学习预测可转债价格

原创 面壁木偶 大音如霜 2020-01-12

免责声明：本文不能作为投资建议，以此做出决策，风险自担。

无论是股票还是可转债，我们希望以便宜的价格买入，以较贵的价格卖出。那就需要选择一种方法对标的估值，比如对股票可以用未来自由现金流折现法，对转债可以使用债券价值加期权价值，这是从商业逻辑上估值，也叫绝对估值法。但我们今天一起学习下相对估值法，所谓相对估值就是基于相似的股票的属性来给某些股票定价，与我们常见的一种机器挖掘预测类似，比如我们根据人群的行为表现来预测某个人是男性还是女性，我们根据汽车配置预测某个车系的售价，这都是相对估计的类型。比如下图，我们每次从所有车系筛出一个作为被预测对象，其他车系用于训练一个基于品牌、类型、配置的价格预测模型，这样循环一下我们就会得到每一台车的预测价格，拿预测价格和厂家定价对比就可以基本看出车系定价是否合理。而预测转债价格也同样遵循相似的逻辑，模型从所有转债中学习转债定价的规律，来预测转债价格合理性，就可以找出哪些转债被错误定价了。



这种方法最大的缺陷就是如果所有车系的定价都是不合理的，机器就很难学到有用的知识；如果所有转债的定价都是不合理的，都是高估的，找到的那些定价失误的转债也只是相对便宜而已，比如牛市时所有转债市场都给予很高的定价，这时你再用类似逻辑就会犯下大错，所以如果使用相对估值法，还需要仓位管理的工具配合，即需要一个工具告诉你

现在是不是整体高估，这类工具我们后面再介绍，今天我们先一起学习下基于机器学习的相对估值法。

要预测转债价格，我们首先需要构建与转债价格相关的特征，即特征构建或成为特征工程。转债的价格主要包括三个方面：债券价值，转股价值、期权价值；那我们需要找到债券相关的属性，转股相关的属性和正股相关的属性，比如波动率，因为波动越大，期权价值才越大。

整理完成相关特征，我们只需要调试个循环预测的模型就可以了。

加载包

```
import pandas as pd
import numpy as np
import tushare as ts
from datetime import datetime
```

加载的包都是我们介绍过的，`datetime`包用来处理时间数据获取系统时间。

设置几个参数

```
updatetime = int(datetime.now().strftime('%Y%m%d'))
inputfile = 'D:/dsod/bond/networkgrasp/'+datetime.now().strftime('%Y-%m-%
timeToMarket = 20191201
financial_year = 2019
financial_quarter = 3
```

1句获取系统日期，并将日期通过`strftime`函数将其转化为类似2020012的格式，然后通过`int`将其转换为数值；2句为了获得存放可转债日交易数据的完整路径，python里

的字符粘贴只需要将其加在一起即可；后面的几句用来设置上市的时间，财报年份和财报季度。

决定转债价格除了纯债价值，还有期权价值，理论上说股票的价格波动越大，其期权价值越大，我们这里暂时不计算期权价值，仅以股票价格的变异系数作为代表期权价值的变量。我们算出最近60日的股价标准差，然后除以均值即得到变异系数。

计算个股股价变异系数

```
histdata = pd.read_csv("D:/dsod/basicdata/hist_data.csv", sep=',', header
histdata = histdata.sort_values(by = ['stockcode', '日期'], ascending = Fa
histdata  = histdata.groupby('stockcode').head(60).reset_index(drop=True)
histdata = histdata[['stockcode', '收盘价']]
stock_standard_var = histdata.groupby('stockcode').agg(np.std, ddof=0)
stock_standard_var  = stock_standard_var.reset_index()
stock_standard_var.columns = ['stockcode', '收盘价_stvar']
temp = histdata.groupby('stockcode').agg(np.mean)
temp = temp.reset_index()
temp.columns = ['stockcode', '收盘价_mean']
stock_standard_var = pd.merge(stock_standard_var, temp, how = 'left', on
stock_standard_var['收盘价_cv'] = stock_standard_var.收盘价_stvar/stock_sta
```

1句读取股票的日交易历史数据；2句使用`sort_values`函数对历史数据根据股票代码、日期进行降序排列；3句分组取每支股票的top60行作为所需数据，并重设每一行的索引；4句仅取出我们需要的两列数据；5句按股票代码分组使用`agg`函数将np包里的`std`标准差函数应用于分组数据，就可以计算出每支股票的股价标准差；6句重设行索引，不要设置drop参数；7句对结果重新命名；8句同样的方法获取股票的60日价格均值；10句将两个数据框使用`merge`函数关联起来;11句计算变异系数。

下面需要将从各大交易平台上采集的可转债日交易数据进行读取，具体每个字段的意思可以参照下表，我们后面也将会逐渐介绍，最后用到的变量如下图。

变量英文	中文名称	变量英文	中文名称
convert_value	转股价值	days_to_next_put_dt	距离回售天数
curr_iss_amt	剩余规模	year_left	剩余年限
pb	市净率	roe	roe
平均股东持有流通市值	平均股东持有流通市值	adj_scnt	成功下调次数
orig_iss_amt	初始规模	pe	市盈率
guarantor	是否担保	days_to_convert_dt	距离转股剩余天数
pe倒数	pe倒数	adj_cnt	建议下调次数
convert_price	转股价	days_to_maturity_dt	距离到期天数
rank_score	正股综合得分	issuer_评级	发行评级
convert_amt_ratio	转债余额与总市值比率	评级	评级
收入同比	收入同比	redeem_price_ratio	强制赎回限价
收盘价_cv	收盘价_cv	qflag	是否可交换债
现金比率	现金比率		
volume	最近交易量		

读取可转债当日数据

```
bond_daily = pd.read_csv(inputfile, sep=',', header = 0)
bond_daily.create_date = updatetime
temp = ['bond_id', 'bond_nm', 'stock_id', 'convert_dt', 'maturity_dt', 'n
        'redeem_price_ratio', 'orig_iss_amt', 'curr_iss_amt', 'rating_cd'
'issuer_rating_cd', 'guarantor', 'qflag', 'margin_flg',
'convert_value', 'premium_rt', 'year_left', 'ytm_rt', 'ytm_rt_tax',
'price', 'volume', 'adj_scnt', 'adj_cnt', 'force_redeem_price', 'convert_
bond_x = bond_daily[temp]
```

1句读取转债数据；2句将当天的日期赋值给数据创建日期；3句声明需要的变量；4句根据声明提取需要的变量列。

时序数据处理日期变量是最常见的工作，而转债有很多重要的日期变量，这些日期变量又和转债的债券价值、期权价值息息相关。如果一个长期向上的股票，当然期权的时间越长，价值越高，越短波动的不确定性越小，其价值也就越低。

处理日期相关数据

```
bond_x.convert_dt = pd.to_datetime(bond_x.convert_dt, format='%Y-%m-%d').
bond_x.maturity_dt = pd.to_datetime(bond_x.maturity_dt, format='%Y-%m-%d'
bond_x.next_put_dt = pd.to_datetime(bond_x.next_put_dt, format='%Y-%m-%d'
bond_x.create_date = pd.to_datetime(bond_x.create_date, format='%Y%m%d').
bond_x['days_to_convert_dt'] = (bond_x['convert_dt'] - bond_x['create_dat
del bond_x['convert_dt']
bond_x['days_to_maturity_dt'] = (bond_x['maturity_dt'] - bond_x['create_d
del bond_x['maturity_dt']
bond_x['days_to_next_put_dt'] = (bond_x['next_put_dt'] - bond_x['create_d
del bond_x['next_put_dt']
```

1句将转股日期处理成标准的日期类型，转债进入转股日期标致这转债可以随时兑换为股票，真正可以实现期权价值的开始；2句是到期日期，即转债到这个日期就要还本付息退市了；3句处理回售日期，回售是债主最重要的权利，即转债跌的太多，可以以设定的价格卖给公司；4句为创建日期；后面几句都是为了计算三大日期与当前日期的差值；这里说下python里的数据框删除变量可以采用 `del` 函数直接删除，也可以使用 `drop` 函数删除，计算完差值后将这些日期变量删除。

债券信用评级是以企业或经济主体发行的有价债券为对象进行的信用评级。等级标准分为A级债券、B级债券、C级和D级债券，其中A级债券是最高级别的债券，本金和收益的安全性最大。我们需要处理主体评级及债券的评级，将他们转化为定距的数值。

处理评级变量

```
bond_x.rating_cd = bond_x.rating_cd.replace(" ", "", regex=True)
bond_x.issuer_rating_cd = bond_x.issuer_rating_cd.replace(" ", "", regex=
rating = pd.read_csv("D:/dsod/bond/dict/rating.csv", sep=',', header = 0)
bond_x = pd.merge(bond_x, rating, how = 'left', on = 'rating_cd')
del bond_x['rating_cd']
rating.columns = ['issuer_rating_cd', 'issuer_评级']
bond_x = pd.merge(bond_x, rating, how = 'left', on = 'issuer_rating_cd')
del bond_x['issuer_rating_cd']
bond_x.loc[bond_x.issuer_评级.isnull(), 'issuer_评级'] = bond_x.loc[bond_x.
bond_x.loc[bond_x.评级.isnull(), '评级'] = bond_x.loc[bond_x.评级.isnull(),
```

1、2句将债券评级、主体评级字符里可能存在的空格全部移除；3读读取评级与定距数据的对照基础表；4句通过’rating_cd’字段关联，将债券评级的字符关联出一个’rating’字段；5句删除债券评级字段；6句对基础表重新命名；7句关联主体评级字段；8句删除主体评级字符字段，这样我们就将评级的字符字段替换成了大多数算法可以读取的数值字段；因为有些债券的评级可能缺失，9句找出主体评级缺失的字段使用相应的债券评级填充；10句找出债券评级缺失的使用主体评级填充。

处理回售价格

```
del bond_x['put_price']
bond_x.loc[bond_x.redeem_price_ratio.isnull(), 'redeem_price_ratio'] = 12
```

因为回售价格大部分都是100元，且有些没有回售条款，1句删除回售价格；2句使用120填充那些强制赎回价缺失值。

处理担保字段

```
bond_x.guarantor = bond_x.guarantor.replace(" ", "", regex=True)
bond_x['guarantor'] = np.where(bond_x.guarantor == '无担保', 0, 1)
```

有担保的债券往往更加安全，说明有物或人发生了连带责任。1句将担保字段中的空格移除；2句使用numpy包里的 `where` 函数将无担保的替换为0，有担保的替换为1，后期我们还会对担保类型进行分类，这里仅简单的分为有担保或者无担保。

处理是否是可交换债

```
bond_x['qflag'] = np.where(bond_x.qflag == 'Q', 1, 0)
```

可交换债和可转债虽然很多属性相同，但也有很多不同，比如股东的转股价下调意愿，参与门槛，与正股关联程度，可交换债都有一定的限定，所以这里要标注出哪些是可交换债，哪些是可转债。

处理是否是融资融券标的

```
bond_x['margin_flg'] = np.where(bond_x.margin_flg == 'R', 1, 0)
```

处理是否是融资标的，因为融资标的的转债价格可能更加稳定，所以我们这里对是否可融资融券进行标注，作为一个特征。

注意在预测价格时一定要除去与溢价率相关的字段

```
del bond_x['margin_flg']
del bond_x['ytm_rt']
del bond_x['ytm_rt_tax']
del bond_x['force_redeem_price']
del bond_x['premium_rt']
```

我们建模最怕的是使用到未来数据，即给模型开了天眼，让模型从某些字段能够非常准确的计算出可转债价格，这里删除一些和可转债价格具有直接兑换关系的变量。

无回售的全部填最大值,无回售威胁下调转股价动力比较小

```
bond_x.loc[bond_x.days_to_next_put_dt.isnull(), 'days_to_next_put_dt'] =
```

回售条款就是对转债持有者最厉害的保护，你觉得买的不满意，你可以以一定价格回售给公司，但有些公司自持禀赋优秀，就没设置回售条款，所以要处理下回售日期，回售日距今天数缺失的转债总是付给它最大值。

无回售的全部填最大值,无回售威胁下调转股价动力比较小

```
bond_x['convert_amt_ratio'] = bond_x['convert_amt_ratio'].str.rstrip('%')
bond_x.bond_id = bond_x.bond_id.astype(str)
```

转债在交易的过程中，会有一部分人选择转股，这样转债的剩余规模会逐渐变小，我们这里处理下转债剩余规模字段，`str.rstrip`函数移除剩余比率的百分号，然后将其转化为浮点数值，除以100即将百分数型转化为了小数型；2句将转债id字符化。

关联变异系数

```
stock_standard_var = stock_standard_var[['stockcode', '收盘价_cv']]
stock_standard_var.columns = ['stock_id', '收盘价_cv']
bond_x = pd.merge(bond_x, stock_standard_var, how = 'left', on = 'stock_i
```

1句提取计算好的变异系数字段；2句将其列重新命名；3句使用股票id列将其关联到转债数据。

我们前面一篇文章介绍了怎么通过多变量排名快速筛选股票的方法，这里要用到排名的结果作为特征输入。

关联正股排名数据

```
inputfile = 'D:/dsod/bond/result/stock_rank'+datetime.now().strftime('%Y%
stock_rank = pd.read_csv(inputfile, sep=',', header = 0)
stock_rank = stock_rank.drop(['name', 'industry', 'area', 'rev_scale', 'e
stock_rank = stock_rank.rename(columns = {"code":"stock_id"})
stock_rank = stock_rank.drop_duplicates()
bond_x = pd.merge(bond_x, stock_rank, how = 'left', on = 'stock_id')
bond_x = bond_x[~bond_x.rank_score.isnull()]
```

1句获取股票综合排名数据的地址；2句读取综合排名数据；3句筛除我们不用的变量，这里移除变量使用`drop`函数；4句使用`rename`函数对code列进行重新命名，方便关联；5句对数据进行一次简单去重；6句关联到可转债主表；7句删除掉综合评分缺失的转债。

通过以上方法我们构建了不少特征，涵盖了转债的方方面面，但也有其他方面的特征没有加入，比如正股的表现、比如行业分类，大家可以根据自己需要添加，后面我们会介绍对行业进行再编码的方法。

下面就进入建模测试阶段。

加载建模相关的模块

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
```

1句因为数据的量纲不同需要归一化模块；2、3句需要随机森林算法和xgb算法，它们是树状模型中两种不同的算法，避免选择同一个类型的算法造成模型的不稳定型；4句筛选分离训练集和测试机的函数`train_test_split`。

归一化及分离数据

```
bond_x =  bond_x.reset_index(drop = True)
idcolumns = ['bond_id','bond_nm','create_date','stock_id', 'price']
X = bond_x.drop(idcolumns, axis = 1).values
y = bond_x['price'].values
sc =  MinMaxScaler()
X_train = sc.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05)
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

1句重新设定行索引；2句声明id列和目标变量‘price’；3句获得所有的自变量赋值给x，并将其转化为array；4句获取目标变量，赋值给y；5句初始化一个归一化模块，这个模块会记住所有列的最大值和最小值，方便分组归一化；6句对所有的自变量归一化，其目的是让归一化模块获得最大值和最小值；7句将数据按比例分为相应的训练集和测试集；7、8句分别对训练集和测试集进行归一化。

构建随机森林模型和xgb模型

```
regressor1 = RandomForestRegressor(n_estimators=30,
                                   min_samples_split=2, oob_score=True)
regressor1.fit(X_train, y_train)
regressor2 = XGBRegressor(learning_rate=0.1, n_estimators = 600, subsampl
regressor2.fit(X_train, y_train)
```

1句声明一个随机森林模型，并设定参数，这里主要设定下树数和最小分裂节点样本数；2句使用训练集训练模型，即模型通过这些学习到什么样的x应该对应什么样的y值；3句声明xgb模型及其参数，我们设置了学习率为0.1，迭代次数为600，每次使用80%的数据；4句训练xgb模型。

预测训练集

```
y_pred = (regressor1.predict(X_test) + regressor2.predict(X_test))/2
```

1句使用随机森林和xgb分别对测试集进行预测，然后取他们的平均数作为预测的最终值。

预测训练集

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred)
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_te
```

1句调用评估模块；3-5分别计算mae、mse、rmse作为评估指标，并打印出来。mae就是真实值与预测值平均相差多少。

我们在建模时往往想看看哪些变量与目标变量相关，这里直接提取随机森林算法评估的特征重要性作为参考，需要说明的是特征重要性不代表简单相关，更不代表因果关系，它只是表示这个变量在做预测时很重要，剔除掉它会影响准确率。

输出变量重要性

```
importances = list(regressor1.feature_importances_)
columnsname = list(bond_x.drop(idcolumns, axis = 1).columns)
importances = pd.DataFrame(importances)
importances.columns = ['importances']
importances['columnsname'] = columnsname
outfile = 'D:/dsod/bond/result/importances.csv'
importances.to_csv(outfile,index =None, encoding = 'gbk')
```

1句提取随机森林算法中计算的特征重要性；2句获取参与建模的特征名称；3句将重要性转化为数据框；4句重新命名重要性列；5句赋值相应的特征名称；8、9写出结果。

这样我们就完成了可转债基于相对估值的机器学习全流程建模，后面用一半数据建模，然后对另外一半的可转债价格预测，这样反复一下，就完成了对所有转债的价格预测。可以选择那些现价与预测价差异较大的转债作为备选标的，但需要提醒大家的是这里有两个很重要的假设：

1.机器比人能够掌握更多变量，机器通过学习获得转债价格更合理

3.目前所有转债的价格整体比较低估

如果脱离这两个假设，一味盲目相信模型，其后果应该就是亏钱。