债券违约研究

胡蝶

风险管理

目录

综述

本课题第一部分从违约债券概况开始引出债券违约始末整理简表，从而得到基于违约事件分析、违约债券最终兑付与否以及宏观流动性这三个角度的归因，并汇总成违约债券重要特征参考表。第二部分在数据的收集与处理后，通过对数据集进行描述性统计了解数据集属性后，选定基于机器学习的复杂分类器作为债券违约预测模型，并利用 Lasso 回归进行特征提取，以增加模型的泛化能力。第三部分先利用 contour 图可视化线性分类器（Logistic Regression、Regularized logistic regression、Support Verctor Machine（linear kernel））与非线性分类器（Support Verctor Machine（RBF kernel）、Neural Network 以及 Random Forest）的差异（未调参），然后对非线性分类器的代价函数和优化算法进行简介，最后给出基于 5 层随机分层抽样交叉验证集的实验结果，得出 Gradient Boosting Desition Tree 分类效果最优以及特征提取有利于提高线性分类器效果的课题结论。其中实验模型超参数调优采用网格搜索方法。第四部分为本阶段课题的展望。本课题使用 Python 进行编程，因为 Python 相关的扩展库 Pandas 和 Sklearn 是进行数据挖掘和机器学习任务强有力的工具。

一、债券违约事件简介与归因

本部分先对违约债券的所属区域、发行人公司属性、行业分布以及债券类型进行统计，形成对债券违约概况的认知。在此基础上，深入整理所有违约债券的违约始末，得出对应债券的一系列违约征兆并汇总成表格。最后，在前述研究的基础上，基于违约事件分析、违约债券最终兑付与否以及宏观流动性这三个角度对债券违约进行归因，得到输入模型的重要特征参考表

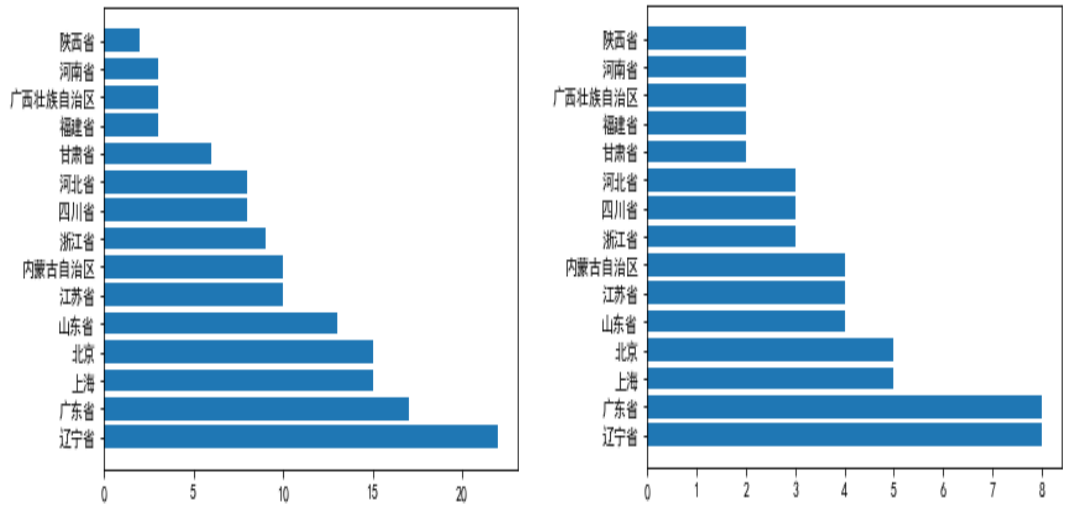
（一）债券市场违约概况

近年来我国债券市场快速发展，发债主体性质、发债主体所属行业及债券类型等日益多样化，债券市场规模迅速壮大。然而，由于宏观经济下行，需求不振以及供给侧去产能等因素，近年来债券违约进入频发期。尤其是 2016 年，我国违约债券数量大幅增长，共有 78 只债券违约，约为 2015 年和 2014 年违约债券总和的 2.7 倍。

由于违约债券中存在同一发行人多只债券违约的情况，为了更为精确的描述债券违约概况。下面将**分别基于违约债券以及违约发行人**，对债券违约概况从地域分布、行业分布、发行人公司属性和债券类型进行统计分析。

1. 地域分布

(1)基于违约数目绝对值统计



数据来源：wind，截止至 2017 年 7 月 17 日

基于违约债券---数量前 15　　**基于违约发行人---数量前 15**

图 1

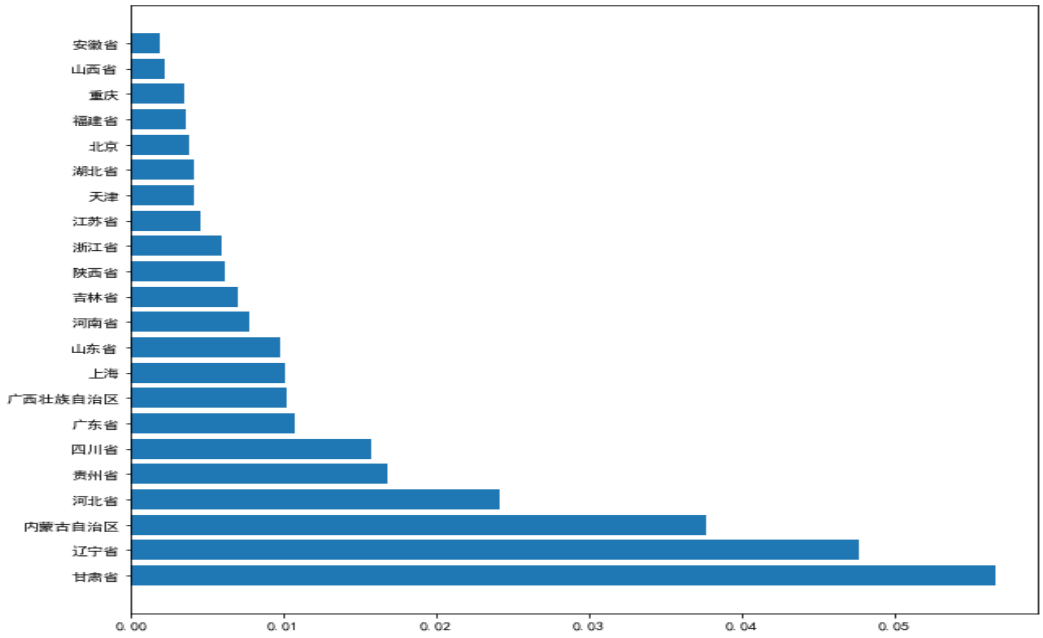
从违约的绝对数目来看，涉及违约的区域（省份）多大 22 处。基于违约债券的统计与基于违约发行人的统计基本一致。其中，辽宁省违约数目最多，这与东北三省近年来经济严重衰退的实情相符。不过可以看到，广东省、上海市与北京市等经济发展良好的区域也有不少债券违约，可能的原因一方面是经济发展较好的区域本身发行债券的基数较大，其中不可避免会有一些资质较差的发行主体。因此，对债券进行评判时，还应多关注发行主体自身的情况。另一方面是违约债券所属区域统计的口径是公司注册地，而公司注册地与主营业务所在区域之间可能存在差异。为了剔除地区发行总量对违约数目分析的影响，设计以下基于

$\frac{\text{违约债券中省份 i 数目}}{\text{到期债券中省份 i 的总数}}$以及$\frac{\text{违约发行人中省份 i 数目}}{\text{到期债券发行人中省份 i 的总数}}$，其中到期债券总数与到期发行人总数样本选取 2017 年 7 月 17 日前所有到期企业债、到期公司债、到期中期票据、到期短期融资券、到期定向工具、到期资产支持证券。

（2）基于违约数目相对于到期债券数目统计

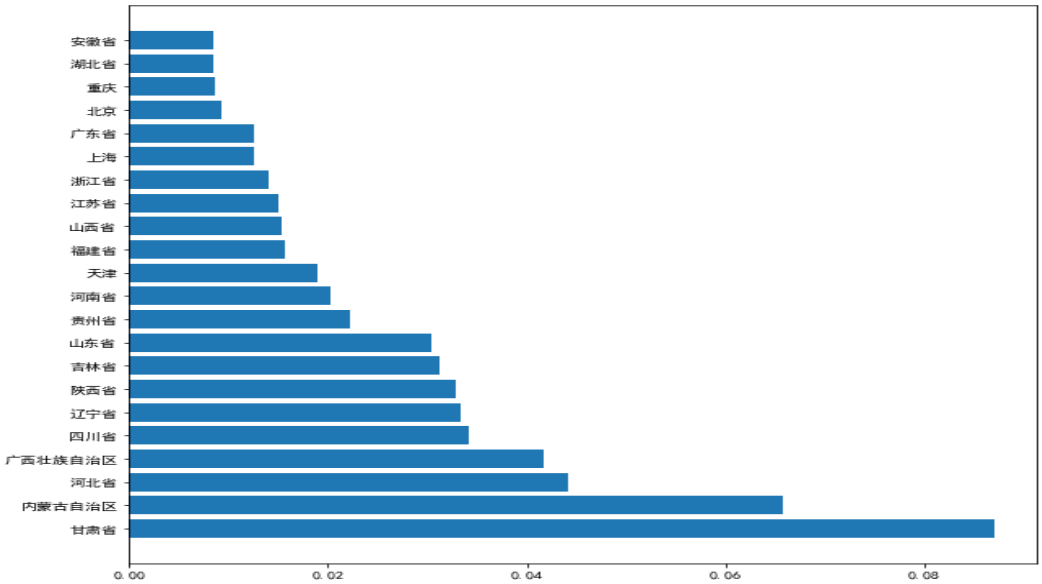
定义：基于违约债券比例的区域分布=$\frac{\text{违约债券中省份i数目}}{\text{到期债券中省份i的总数}}$

基于违约发行人比例的区域分布=$\frac{\text{违约发行人中省份i数目}}{\text{到期债券发行人中省份i的总数}}$



数据来源：wind，截止至 2017 年 7 月 17 日

图 2 基于违约债券比例



数据来源：wind，截止至 2017 年 7 月 17 日

图 3 基于违约发行人比例

地域分布对比表

排序（降序）	基于违约债券比例	基于违约发行人比例	基于违约债券数目	基于违约发行人数目
1	甘肃省	甘肃省	辽宁省	江苏省
2	辽宁省	内蒙古自治区	广东省	山东省
3	内蒙古自治区	河北省	北京	浙江省
4	河北省	广西壮族自治区	上海	内蒙古自治区
5	贵州省	四川省	山东省	四川省
6	四川省	辽宁省	江苏省	广东省
7	广东省	陕西省	内蒙古自治区	北京
8	广西壮族自治区	吉林省	浙江省	上海
9	上海	山东省	河北省	河北省
10	山东省	贵州省	四川省	辽宁省
11	河南省	河南省	甘肃省	甘肃省
12	吉林省	天津	广西壮族自治区	天津
13	陕西省	福建省	河南省	广西壮族自治区
14	浙江省	山西省	福建省	河南省
15	江苏省	江苏省	天津	陕西省
16	天津	浙江省	陕西省	福建省
17	湖北省	上海	重庆	吉林省
18	北京	广东省	贵州省	山西省
19	福建省	北京	湖北省	贵州省
20	重庆	重庆	山西省	湖北省
21	山西省	湖北省	吉林省	安徽省
22	安徽省	安徽省	安徽省	重庆

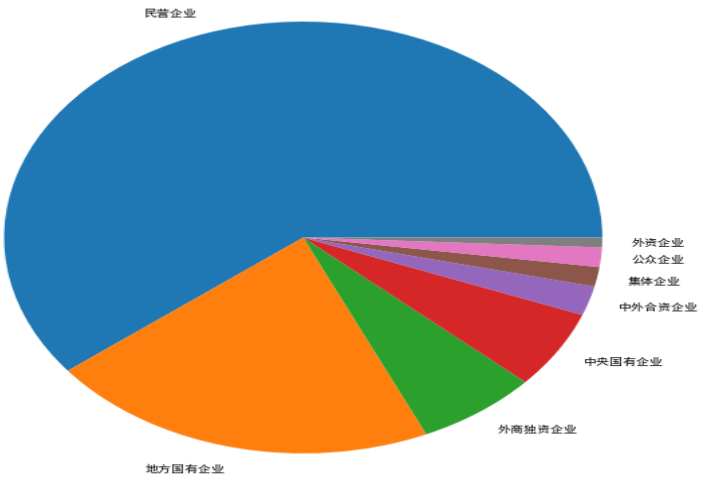
数据来源：wind，截止至 2017 年 7 月 17 日

从对比表中可以看到，基于$\frac{\text{违约债券中省份 i 数目}}{\text{到期债券中省份 i 的总数}}$以及$\frac{\text{违约发行人中省份 i 数目}}{\text{到期债券发行人中省份 i 的总数}}$发现，甘肃省发行的债券中违约比例最高，而辽宁省、河北省、内蒙古自治区以及四川省的违约比例也排名较高。特别的，从各省基于违约发行人的违约比例来看，违约主要分布在经济发展较缓的区域，并且具有一定的**聚集性**。将基于违约发行人违约比例较高的省份，在中国地图上圈出来，可以更清楚的看到区域分布情况。

图 4 基于违约发行人违约比例的主要分布图

2. 发行人公司属性

(1)基于违约数目绝对值统计



数据来源：wind，截止至 2017 年 7 月 17 日

图 5 基于违约债券

债券发行人公司属性中，民营企业占债券违约发行主体的大部分，不过值得注意的是，地方国有企业违约占比也不低。因此，地方国有企业虽然表面上有地方政府为其信用背书，然而在经济下行，政府财政收入下降，刚性兑付逐渐被打破等因素的叠加之下，地方国有企业在债券偿还方面并不一定受到地方政府支持。特别的，通过后面事件归因可以发现，相对于地方产业债，地方政府对地方城投债兑付的支持力度仍较大。

（2）违约数目相对于到期债券数目统计

定义：基于违约债券比例的公司属性分布=$\frac{\text{违约债券中公司属性i数目}}{\text{到期债券中公司属性i的总数}}$

基于违约发行人比例的公司属性分布=$\frac{\text{违约发行人中省份i数目}}{\text{到期债券发行人中省份i的总数}}$

公司属性分布对比表

排序（降序）	基于违约债券比例	基于违约发行人比例	基于违约债券数目	基于违约发行人数目
1	民营企业 **历史违约率：2.44%**	外资企业 **历史违约率：5.56%**	民营企业	民营企业
2	外商独资企业 历史违约率：2.34%	集体企业 历史违约率：5.56%	地方国有企业	地方国有企业
3	集体企业 历史违约率：2.27%	外商独资企业 历史违约率：4.82%	外商独资企业	中央国有企业
4	外资企业 历史违约率：1.28%	民营企业 历史违约率：4.21%	中央国有企业	外商独资企业
5	中外合资企业 历史违约率：0.737%	中外合资企业 历史违约率：2.04%	中外合资企业	中外合资企业
6	地方国有企业 历史违约率：0.303%	公众企业 历史违约率：1.08%	集体企业	集体企业
7	公众企业 历史违约率：0.272%	中央国有企业 历史违约率：0.739%	公众企业	外资企业
8	中央国有企业 历史违约率：0.169%	地方国有企业 历史违约率：0.418%	外资企业	公众企业

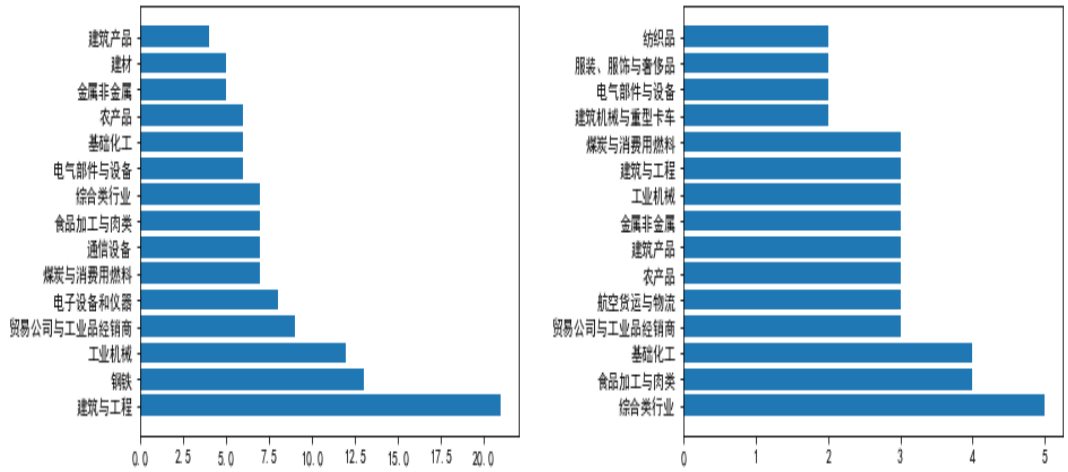
数据来源：wind，截止至 2017 年 7 月 17 日

通过分别计算$\frac{\text{违约债券中公司属性 i 数目}}{\text{到期债券中公司属性 i 的总数}}$以及$\frac{\text{违约发行人中省份 i 数目}}{\text{到期债券发行人中省份 i 的总数}}$的公司属性分布，得出了与数目分布不同的结论。其中，基于$\frac{\text{违约债券中公司属性 i 数目}}{\text{到期债券中公司属性 i 的总数}}$计算出的违约债券比例排名前三依次为民营企业（历史违约率：2.44%）、外商独资企业（历史违约率：2.34%）和集体企业（历史违约率：2.27%）。而基于绝对数量中第二大的地方国有企业

违约率较低（0.303%），排到了第六。因此，从基于违约债券比例排名来看，具有国有背景（地方国有企业、中央国有企业）的违约率较低。从基于违约发行人比例来看，同样可以得到国有背景企业违约率较低的结论。此外，可以看到，具有外资背景的企业违约率排名相对靠前，特别是外资企业，历史违约率达到了 5.56%，外商独资企业和中外合资企业的历史违约率也分别达 4.82%和 2.04%。因此，对外资企业所发行的债券应格外谨慎。

3. 行业分布

(1)基于违约数目绝对值统计



数据来源：wind，截止至 2017 年 7 月 17 日

基于违约债券---数量前 15　　**基于违约发行人---数量前 15**

图 6

从违约行业分布来看，所涉及的 **wind 四级行业**种类多达 32 种，并且基于违约债券和违约发行人统计的分布有一定的不同。(1) **基于违约债券**的统计来看：违约行业分布主要集中在建筑与工程、钢铁、工业机械、电子设备和仪器、煤炭与消费用燃料等产能过剩和周期性较强的传统行业；同时受全球经济不景气以及逆全球化的影响，贸易公司和工业品经销商违约规模也较大；另外，同样由于经济下行等因素的影响，食品加工与肉类、农产品等行业也出现了较大规模的违约。(2) **基于违约发行人**的统计来看：违约发行人行业排名前三依次为综合类行业、食品加工与肉类、基础化工，而贸易公司与工业经销商、航空货运与物流、农产品、建筑产品、金属与非金属、工业机械、建筑与工程以及煤炭与消费用染料的违约发行人数目相同。

（2）基于违约数目相对于到期债券数目统计

定义：基于违约债券比例的行业分布=$\frac{\text{违约债券中行业i数目}}{\text{到期债券中行业i的总数}}$

基于违约发行人比例的行业分布=$\frac{\text{违约发行人中行业i数目}}{\text{到期债券发行人中行业i的总数}}$

排序（降序）	基于违约债券比例	基于违约发行人比例	基于违约债券数目	基于违约发行人数目
1	餐馆 历史违约率 50%	餐馆 历史违约率 50%	建筑与工程	综合类行业
2	鞋类 历史违约率 20%	鞋类 历史违约率 25%	钢铁	食品加工与肉类
3	电子设备和仪器 历史违约率 17.8%	工业气体 历史违约率 25%	工业机械	基础化工
4	工业气体 历史违约率 16.7%	航空货运与物流 历史违约率 16.7%	贸易公司与工业品经销商	贸易公司与工业品经销商
5	石油天然气设备与服务 历史违约率 11.1%	商业印刷 历史违约率 16.7%	电子设备和仪器	航空货运与物流
6	商业印刷 历史违约率 11.1%	建筑产品 历史违约率 14.3%	通信设备	农产品
7	建筑产品 历史违约率 7.84%	金属与玻璃容器 历史违约率 14.3%	食品加工与肉类	建筑产品
8	金属与玻璃容器 历史违约率 7.41%	石油天然气设备与服务 历史违约率 12.5%	综合类行业	金属非金属
9	航空货运与物流 历史违约率 6.15%	电子设备和仪器 历史违约率 9.52%	煤炭与消费用燃料	工业机械
10	服装、服饰与奢侈品 历史违约率 4.55%	服装、服饰与奢侈品 历史违约率 9.09%	基础化工	建筑与工程
11	半导体产品 历史违约率 4.55%	半导体产品 历史违约率 8.33%	农产品	煤炭与消费用燃料
12	工业机械 历史违约率 4.51%	食品加工与肉类 历史违约率 6.25%	电气部件与设备	航空货运与物流
13	农产品 历史违约率 4.41%	农产品 历史违约率 6%	金属非金属	电气部件与设备
14	通信设备 历史违约率 4.29%	纺织品 历史违约率 5.88%	建材	服装、服饰与奢侈品
15	电气部件与设备 历史违约率 3.21%	通信设备 历史违约率 5.56%	建筑机械与重型卡车	纺织品
16	燃气 历史违约率 2.82%	化纤 历史违约率 5.26%	航空货运与物流	电子设备和仪器
17	食品加工与肉类 历史违约率 2.31%	燃气 历史违约率 4.55%	建筑产品	半导体产品

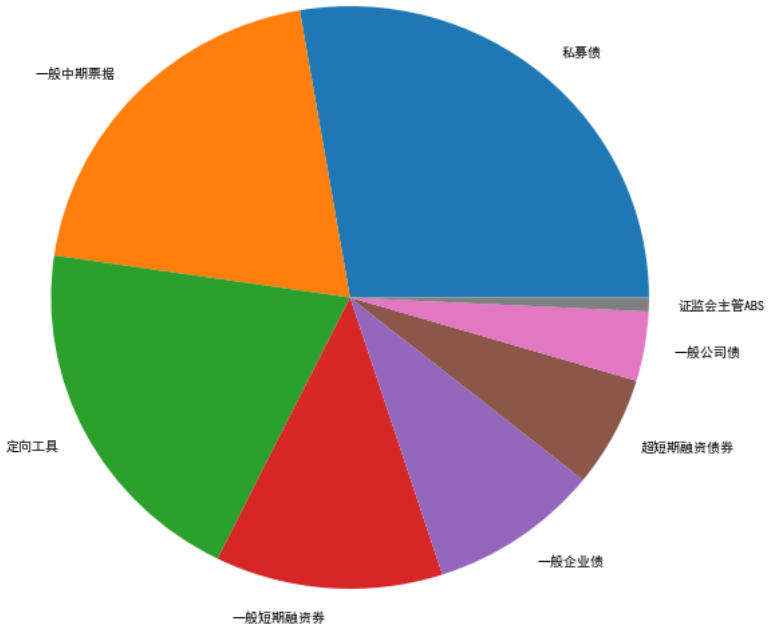
18	钢铁 历史违约率 2.27%	建筑机械与重型卡车 历史违约率 4.35%	半导体产品	建筑机械与重型卡车
19	金属非金属 历史违约率 1.68%	贸易公司与工业品经销商 历史违约率 4.29%	服装、服饰与奢侈品	钢铁
20	建筑机械与重型卡车 历史违约率 1.59%	基础化工 历史违约率 4.26%	燃气	石油天然气设备与服务
21	贸易公司与工业品经销商 历史违约率 1.58%	电气部件与设备 历史违约率 3.85%	纺织品	鞋类
22	纺织品 历史违约率 1.57%	工业机械 历史违约率 3.61%	西药	燃气
23	建筑与工程 历史违约率 1.48%	机动车零配件与设备 历史违约率 3,57%	金属与玻璃容器	公路与铁路
24	基础化工 历史违约率 1.32%	金属非金属 历史违约率 3.53%	鞋类	金属与玻璃容器
25	化纤 历史违约率 1.12%	钢铁 历史违约率 2.82%	化纤	餐馆
26	西药 历史违约率 1.12%	煤炭与消费用燃料 历史违约率 2.68%	商业印刷	海港与服务
27	建材 历史违约率 1.09%	西药 历史违约率 2.22%	工业气体	机动车零配件与设备
28	煤炭与消费用燃料 历史违约率 0.848%	海港与服务 历史违约率 1.89%	机动车零配件与设备	建材
29	机动车零配件与设备 历史违约率 0.813%	建材 历史违约率 1.52%	海港与服务	商业印刷
30	综合类行业 历史违约率 0.349%	综合类行业 历史违约率 1.17%	餐馆	化纤
31	海港与服务 历史违约率 0.236%	公路与铁路 历史违约率 0.855%	公路与铁路	通信设备
32	公路与铁路 历史违约率 0.106%	建筑与工程 历史违约率 0.743%	石油天然气设备与服务	工业气体

数据来源：wind，截止至 2017 年 7 月 17 日

从$\frac{\text{违约债券中行业 i 数目}}{\text{到期债券中行业 i 的总数}}$以及$\frac{\text{违约发行人中行业 i 数目}}{\text{到期债券发行人中行业 i 的总数}}$的分布来看，受到分母的影响，分布变化较大。在违约数目中排名较靠前的行业受到分母的影响，在违约比例中的排名降到了较后的位置，具体违约率参见上表。

4. 债券类型

(1)基于违约数目绝对值统计



数据来源：wind，截止至 2017 年 7 月 17 日

图 7

我国违约债券的债券类型中，私募债违约数目最大，紧跟其后的分别是定向工具。可见非公开发行类别的债券由于发行时信息披露范围减少，发行注条件被简化等原因，存在较大风险隐患，对投资者的风险管理能力要求较大。并且风险更多地发生于由银行间市场交易商协会主管的产品，而国家发改委主管的企业债被感染的风险相对较少。一方面，这是国家发改委对于违约的零容忍态度所致。另一方面，外部支持风险通常与资本运作所关联，发改委对此要求更为严格

（2）违约数目相对于到期债券数目统计

定义：基于违约债券比例的债券类型分布=$\frac{\text{违约债券中债券类型i数目}}{\text{到期债券中债券类型i的总数}}$

排序 （降序）	基于违约债券 比例	基于违约债券 数目
1	私募债 历史违约率 4.64%	私募债

2	一般公司债 历史违约率 1.91%	一般中期票据
3	一般中期票据 历史违约率 1.5%	定向工具
4	一般企业债 历史违约率 1.42%	一般短期融资券
5	定向工具 历史违约率 1.17%	一般企业债
6	一般短期融资券 历史违约率 0.238%	超短期融资债券
7	超短期融资债券 历史违约率 0.171%	一般公司债
8	证监会主管 ABS 历史违约率 0.0746%	证监会主管 ABS

数据来源：wind，截止至 2017 年 7 月 17 日

因为同一发行人可能会发行不同类型的债券，因此这里不对违约债券类型基于发行人进行统计。基于违约债券比例得出的分布于基于违约债券数目得出的分布基本一致，除了在一般公司债的分布上有一定出入。基于比利的一般公司债违约率排到了第二位，为 1.91%，而基于数目的一般公司债排到了第六位。私募债的历史违约率高达 4.64%，因此在投资私募债时应谨慎为之。

（二）债券违约事件整理

本研究对 2017 年 7 月 17 日前违约的全部债券进行了详尽的考察，并总结其违约始末，最后从中得出一系列特征用于债券违约事件的预测。下表将展示违约债券违约征兆简括，具体违约始末参见债券违约事件整理 Word 文档。

表 1 债券违约始末整理简表

违约时间	违约债券	违约征兆	最终是否兑付
2014.03.04	11 超日债	1.连续亏损 2.信息披露不规范 3.总经理辞职 4.生产线停产 5.借款逾期 6.供应商诉讼	是（国家队救）
2014.03.28	13 中森债	1.经济持续下行 2.企业经营业绩不佳 3.流动性管理存在问题	是（担保人救）
2014.07.10	12 金泰债	1. 联保互保 2.地方政府将发行作为政治任务给予承销商发行 3.主承销商浙商证券存在重大未披露事项	否
2014.07.16	13 华通路 CP001	1.公司实际流动资金有限 2.实际控制人风险 3.关联企业风险	是（当地政府救）

2014.7.28	12 津天联	1.母公司深陷债务危机 2.对母公司连带担保责任 3.面临大量诉讼	否
2014.08.25 **2016.08.22**	13 华珠债	担保人未履约	否
2014.12.27	11 华锐 01	1.盈利能力呈持续下行态势 2.经营现金持续净流出 3.现金比率下降	是（资本公积金转增股本）
2015.01.27	12 东飞 01	1.市县政府融资平台管理不规范 2.保人东台交投以公告的形式撇清偿付责任，声明只担保信用评级，不担保债券偿还责任	否
2015.02.04	12 蓝博 01	1.债务逾期 2.现金流紧张 3.资产受限及涉诉 4.担保人失信	否
2015.02.05	12 致富债	1.因环保标准停产改造只有投入没有盈利导致现金流紧张 2.担保人违约	否
2015.04.07	12 湘鄂债	1. 限制“三公消费”导致中高端酒楼市场低迷 2.盲目转型导致资金压力增大	是(代偿，上市公司壳价值)
2015.04.12	14 波鸿 CP001	1. 在资金划转过程中，因资金调度和操作时间原因	是
2015.04.21	11 天威 MTN2	1. 行业层面：(1) 市场供需极度不平衡，产能严重过剩； (2) 光伏过剩+出口依赖、欧盟双反政策； (3) 风电过剩+美国对我国风塔反倾销、反补贴调查 2. 公司层面：(1) 自身经营不佳，兵装弃军保帅；(2) 企业财务数据远逊于行业平均水平 (3) 盲目扩张规模导致资金链紧张； (4) 评级异常下调	否（破产重组）
2015.12.30	12 天威 PPN001		
2016.03.28	13 天威 PPN001		
2016.04.21	11 天威 MTN2		
2016.05.12	11 威利 MTN1		
2015.05.25	12 中富 01	1.下游饮料行业增速放缓 2.主要三大客户订单量减少以及销售价格明显下降	是
2017.03.28	12 珠中富 MTN1	3.股东变更等因素与贷款协议不符，资金请求遭到银团拒绝，从而最终触发公司资金链断裂	是
2015.07.15	13 大宏债	担保方的射阳县城投提出的有条件进行代偿，挑战担保的契约精神	是(地方政府救)
2015.10.13	10 英利 MTN1	1.光伏组件行业景气度下降 2.关联企业占用运营资金，导致公司开工率不足盈利下降 3.流动资产大规模被关联方占用	否
2015.10.14	08 二重债	1.传统行业，产能过剩和竞争过度 2.连续四年亏损 3.产能投放节奏与行业景气度大多不匹配，导致大量资金投资沉淀，公司资产负债率急剧上升 4.融资能力丧失，资金枯竭资不抵债	是（实际控制人央企救）

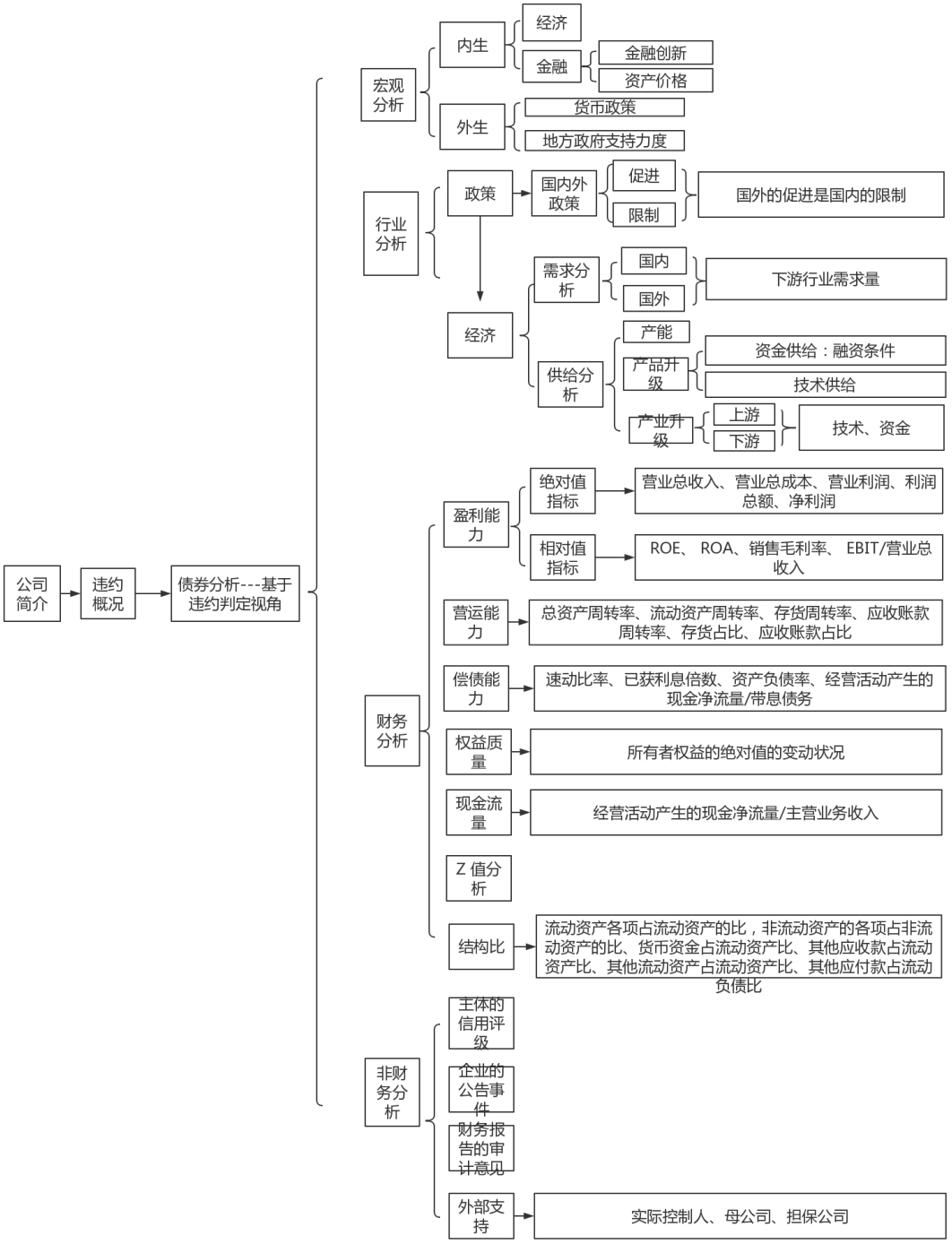
2015.10.14	12 二重债 MTN1	.严重依赖子公司，受子公司拖累：二重重装占二重集团经营的几乎全部，若二重重装进入重整程序，公司对二重重装的长期股权投资、债权、担保等事项可能会受到影响，集团进入破产重整的概率较大	是（实际控制人央企救）
2015.10.19	10 中钢债	1.宏观经济环境下行 2.行业景气度下滑 3.公司连年亏损 4.债务压力繁重 5.大股东自身难保	否
2015.10.28	12 蒙农科	三大业务农牧、化肥房地产分别受到流动资金紧张、价格倒挂及安全事故影响、出现停产、可售面积减少、后续投入资金相对缺乏等问题	否
2016.05.04	11 蒙奈伦债、11 蒙奈伦		
2017.05.04	11 蒙奈伦债、11 蒙奈伦		
2015.11.12	15 山水 SCP001	1. 水泥行业整体盈利能力降低 2. 再融资受阻：内部股权纠纷以及外部股权之争导致公司迟迟不能复牌，亦无法从股票市场获得资金支持	否
2016.01.21	13 山水 MTN1		
2016.02.14	15 山水 SCP002		
2017.02.27	14 山水 MTN001		
2017.05.12	14 山水 MTN002		
2015.12.07	12 圣达债	1.宏观经济下行导致下游需求减少 2.钢铁价格下跌导致利润大幅下滑 3.公司对子公司生产线投资过大 4.受限资产比重大 5.增信资产多次质押	否
2016.02.09	15 亚邦 CP001	1. 传统产能过剩行业，经营可持续性难度大 2. 负债率高 3.资产上市控股公司股权全部质押 4. 股价下跌加大抵押补充需求 5.董事长被要求协助调查导致银行抽贷、压贷	是
2016.09.29	15 亚邦 CP004		
2016.2.29	14 云峰 PN001、14 云峰 PN002、14 云峰 PN003、15 云峰 PN001、15 云峰 PN003、15 云峰 PPN005	1、煤炭业务恶化 2、前期激进投资带来长短期周转压力影响偿债能力 3、原实际控制人绿地控股接触股权托管后导致再融资渠道收紧	否
2016.08.01	15 云峰 PPN004		
2016.03.09	15 宏达 CP001	1.秘鲁邦沟铁矿项目沉重的投资压力 2.行业不景气投资项目转让无果 3.股票市场下跌导致股权质押借款需要补缺	是
2016.03.09	13 桂有 PPN001	1.有色行业萎靡 2.资不抵债 3.旗下优质资产被转移	否（破产清算）
2016.04.22	13 桂有 PPN002		
2016.03.10	12 中成债	1.国际石油价格下跌，公司石油相关业务出现了较大下滑 2.公司应收账款占流动资产占比高	否

2016.03.17	15 雨润 CP001	1.营业收入明显下滑，经营状态远差于行业平均水平。 2.前期产能扩张太快 3.实际控制人祝义才先生被检查机关执行指定居所监视居住至今，导致外部融资受影响	是（母公司救）
2016.05.13	13 雨润 MTN1		
2016.03.28 **2016.04.05** **2016.04.12** **2016.05.05** **2016.06.06** **2016.07.11** **2016.07.18** **2016.09.06** **2016.09.26** **2017.01.16** **2017.04.12** **2017.07.17**	15 东特钢 CP001 15 东特钢 SCP001 13 东特钢 MTN2 15 东特钢 CP002 14 东特钢 PPN001 13 东特钢 PPN001 15 东特钢 PPN002 13 东特钢 PPN002 15 东特钢 CP003 13 东特钢 MTN1 13 东特钢 MTN2 15 东特钢 PPN002	1.公司盈利在钢铁行业尚可，但是资产负债率长期偏高 2.短期债务占总债务在 70%以上，公司债务负担过重。 3.东特钢自 2013 年以来已有 12 条失信信息，其中 2014 年、2015 年分别有 7 条和 3 条。 4.固定资产和在建工程规模庞大，同时已大量抵押	否
2016.04.06	15 华昱 CP001	1.煤炭行业景气度下降影响 2.公司近两年盈利恶化非常迅速，亏损侵蚀净资产 3.债务规模上升，推动财务杠杆和偿债指标迅速恶化	是（实际控制人央企救）
2016.05.16	15 春和 CP001	1.船舶行业周期性较强，景气波动剧烈，行业信用品质较差 2.国际原油价格的大幅下跌使海洋工程装备制造和营运市场持续低迷 3.2015 年以来，公司出现核心高管变更、出售核心公司股权换取现金流等系列问题 4.涉足钾肥业务失败。	否
2017.04.24	12 春和债		
2016.05.29	14 益优 02	煤炭行业不景气	否
2016.06.15	15 川煤炭 CP001	1.下游多重需求下滑，主营成本收入倒挂 2. 逆周期扩张产能，导致资产负债率达 88.48%，且非流动资产占比大 3.区域风险，四川省企业违约事件频发	否
2016.12.26	13 川煤炭 PPN001		
2017.05.12	12 川煤炭 MTN1		
2017.05.19	14 川煤炭 PPN001		
2016.08.08	15 国裕物流 CP001	1. 散货船市场极度低迷使中小船厂风险暴露，公司经营压力大。近三年产能利用率水平较低，仅 20%左右 2. 航运业务资产收益很低，国际航运业实体已接近资不抵债，重资产的经营模式进一步拖累经营。	否
2016.10.28	15 国裕物流 CP002		
2016.11.17	15 冀物流 CP002	1.行业运营特征使得公司负债集中于流动负债公司即期债务偿付压力很大 2.受宏观经济下行影响，公司近年盈利能力一直较弱，利息保障倍数低于警戒线	是
2016.11.17	15 冀物流 CP002		
2016.11.21	15 机床 CP003、16	1.债务负担重	否

	大机床 SCP001	2.货币资金几乎全部为受限资金，债务严重依赖再融资周转。 3.公司大量资产已用于抵质押，银行授信基本使用完毕 4.交叉违约条款	
2016.12.19	15 机床 CP004		
2017.01.16	16 大机床 MTN001		
2017.02.06	15 机床 PPN001		
2017.02.07	16 大机床 SCP003		
2017.02.13	16 大机床 SCP002		
2017.05.24	14 机床 PPN002		
2016.11.28	14 中城建 PPN003 14 中城建 PPN004 12 中城建 MTN1	1、控股股权纠纷，公司融资渠道受到限制，导致资金链紧张 2、公司面临多起债务诉讼，部分银行账户及资产已被查封或保全。	否
2016.12.09	11 中城建 MTN1		
2016.12.19	12 中城建 MTN2		
2017.03.01	16 中城建 MTN001		
2017.06.12	14 中城建 PPN002		
2017.07.14	15 中城建 MTN001		
2016.12.05	16 博源 SCP001	1. 公司规模优势显著但行业景气向下，盈利能力表现羸弱。 2. 公司受限资产规模大，占净资产比重很高，再融资的腾挪空间有限。 3.公司负面事件频发，短期内对公司的资金筹措产生直接负面影响。	是
2017.02.03	16 博源 SCP002		
2017.03.13	12 江泉债	1.钢铁行业产能过剩，江泉集团利润大幅亏损，盈利能力下滑 2.不符合临沂市专项环保行动的要求被停产整治 3.多笔银行贷款逾期，再融资渠道枯竭。 4.对外担保和互保规模庞大	否
2017.06.19	13 弘燃气、13 弘昌燃气债	1.大规模投资，在建工程投资激进，经营和非经营性占款均严重 2.债务规模攀升，致使公司资金周转困难，违约状况不断	否

（三）债券违约归因

本部分基于违约事件分析角度、违约债券最终兑付与否角度以及宏观流动性角度三方面对债券违约进行归因，得到包含宏观层面、企业层面——财务指标以及企业层面——经营管理指标三大类的重要特征表，构成后期模型输入的依据。



资料来源：各债券违约事件

图 8 违约分析框架总结

1. 债券违约归因---违约事件分析角度

本部分对 2017 年 7 月 17 日前违约的全部 130 只债券进行了详尽的考察，通过了解债券违约事件的始末并进行总结，得到债券违约的原因，下面从微观原因和宏观原因两方面出发来具体阐述。

(1) 微观原因

1) **公司经营状况恶化，现金流紧张**

几乎所有违约债券违约的必要非充分原因，但是压死骆驼的最后一根稻草各有不同，也就是下面所述的原因。

2) **担保方未履行担保代偿责任**

由中海信达担保的'13 中森债'等一系列债券违约，因此担保人资质对于私募债的投资者至关重要。对于担保公司来说，公司业务不会集中于私募债，还有大量包括银行贷款在内的担保责任，相对于私募债，银行贷款责任更为重要，因此私募债可能不在优先偿还考虑范围内。在业务执行过程中，担保公司多数实施**杠杆化**操作，将公司大部分资金以保证金的形式存放于银行账户，通过保证金账户，担保公司能够获得更大规模的银行授信，从而扩大担保额度。当项目发生违约后，担保公司即使不履行连带责任，法院也很难进行强制执行，只能在银行配合下查封账户，并不具备转出资金的权利。只有担保公司保证金对应债务到期的情况下，法院才能强制执行担保，但目前大多数担保公司都实行循环担保，因此法院基本上没有机会强制执行担保。

3) **企业重要资产剥离、投资激进等导致偿债能力和再融资能力下降**

重要资产剥离，一方面，企业将缺少优质抵押品，**再融资条件恶化**，导致资金了断裂；另一方面，重要资产一般被母公司剥离，而相对于投资者，**母公司更了解企业的经营情况**，当出现重要资产剥离时，极有可能是公司经营恶化，母公司的自保行为。例如，造成'13 天威 PPN001'违约的一个非常重要的原因是，母公司兵装集团将天威集团和上市公司保变电气进行了资产置换（图 9），将传统优势业务输变电剥离给上市公司并增持上市公司股份，取代天威集团成为第一大股东；同样在'13 桂有色 PPN001'违约事件中，广西有色是广西国资委企，在其违约前，广西有色旗下的优质资产华锡集团的股份被受让给同为广西国资委旗下的广西北部湾国际港务集团。**投资激进，**带来资产负债率快速上升也会导致资金链断裂。例如在"08 二重债、12 二重债 MTN1"违约事件中，二重重装的产能投放节奏与行业景气度不

匹配，导致大量资金投资沉淀，资产负债率急剧上升，最终违约。

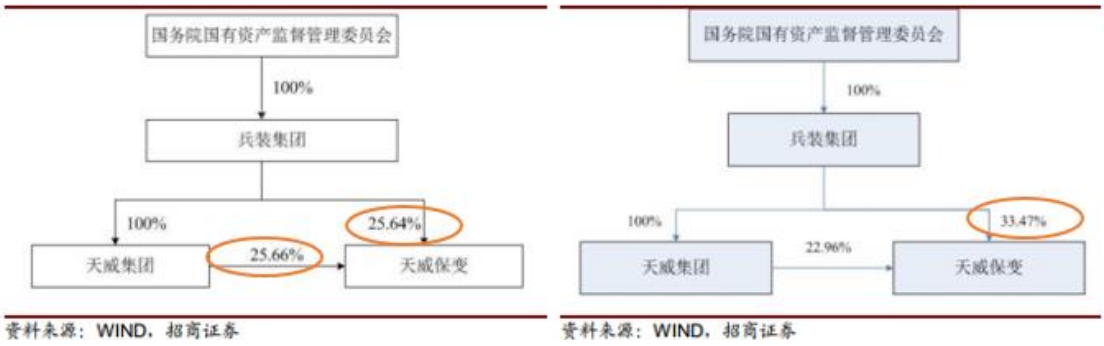


图 9

4) **对外担保过大、代偿风险加剧、母公司经营不善等导致经营陷入困境**

对外担保过大，例如在'12 金泰债'中，实际控制人为关联及非关联企业担保，在被担保企业出现无力还款的情形下，相关债务被转移到实际控制人名下造成违约。（图 10）。**母公司经营不善，**例如'12 津天联'中，天联复材的母公司天联集团深陷债务危机，且大部分由天联复材担保，由于对母公司的连带担保责任，面临大量诉讼，以致停产。

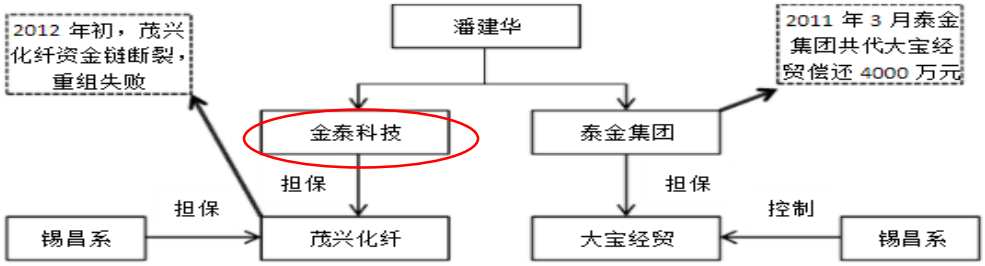


图 10

5) **股权结构历史沿革复杂、公司治理不善，发生内部冲突、实际控制人负面消息以及关联企业风险等原因，导致银行抽贷、停贷等再融资方式受限，而出现流动性危机**

股权结构历史沿革复杂，例如'14 云峰 PPN001'违约事件中，由于股权变更导致第一大股东由绿地集团变为职工持股（图 11），**而银行包括 PPN 投资者等债权人愿意对其融资主要考虑了绿地集团对云峰的支持和增信，**因此股权变更导致银行收缩信贷，加快流动性枯竭，所以要特别注意有股权托管协议的发行主体。同样，在"12 中富 01"中，发行人珠海中富实业是国内屈指可数的 PET 瓶生产企业，由于**股东变更等因素与银团贷款协议不符**，向银团的资金请求遭到拒绝，触发公司资金链断裂。**内部冲突，**例如在"15 山水 SCP001"的违约事件中，山水水泥集团内部股权纠纷，外部股权之争、造成业绩下滑以及公司停牌无法从股票市场获得融资，最终违约。**实际控制人负面消息，**在"15 亚邦 CP001"违约事件中，山水水泥公司董事长徐小初被要求协助调查，对公司造成

较大负面影响，部分银行由于风控原因抽贷压贷，导致公司资金紧张，‘15 雨润 CP001’等也是类似情况。

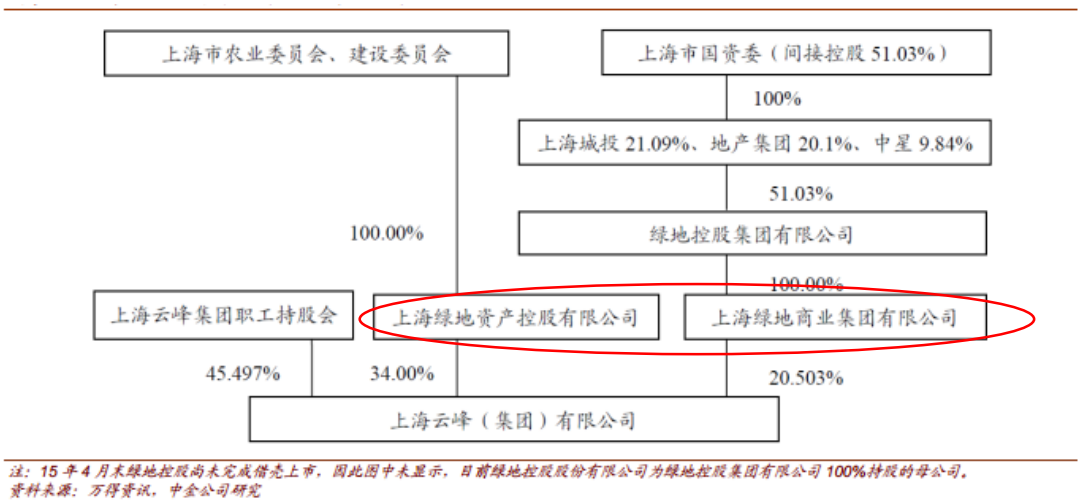


图 11

(2) 宏观原因

1) 行业产能过剩

例如在违约样本中**“两高一剩”**[1]行业占比较大，例如在“11 天威 MTN2”的违约事件中，天威集团投资光伏和风电等产能过剩行业，投资支出大导致资产负债率高，且经营这些行业不但没有带来收入，还加重企业的亏损情况。**下游行业需求大幅下降**也是触发违约的一个关键原因，在“12 圣达债”违约事件中，四川圣达集团主要收入来源是生铁和制动鼓业务，而 2014 年重型货车需求减少，因此下游对制动鼓需求减少，公司收入大幅下降，导致 2015 年 12 圣达债违约。

2) 国内外相关政策

例如国家对**三公消费**的限制导致中高端酒楼代表企业湘鄂情违约**（图 12）**。国外针对我国的**“双反”**政策，导致严重依赖光伏产品出口的保定天威集团违约。

[1] 根据国发【2010】7 号、国家发展和改革委员会令第 21 号，发改产业【2010】1635 号、国发【2013】41 号等文件，“两高一剩”等国家限制性行业主要包括钢铁、水泥、电力、电石、焦炭、电解铝、平板玻璃、船舶、煤炭、有色金属、造纸、制革、印染、纺织、石油加工、煤化工等行业。

图 5：餐饮行业销售利润率呈稳步上升趋势（全行业，单位：%）

资料来源：WIND，招商证券

图 12

3) **地方政府支持力度，例如政府财政支持，政府对发行人抵押资产处置的方式等**

政府对发行人抵押资产处置，例如‘08 蒙奈伦债’能兑付的主要原因是为促成债券足额按期偿付，呼和浩特市政府成立专项小组，同意在债券存续期间将所抵押的土地列入市土地收储计划，将用地性质由种植用地变更为商住开发用地，市政府土地收储中心依公允价格收购。

2. 债券违约归因---违约债券最终兑付与否角度

通过对债券最终是否偿还进行归因也可以得到一些重要的特征，主要反映的是债券的外部支持因素，因此可以用于事前预测债券违约的概率。在违约样本中，最终得以成功偿还的债券具备以下特征的一项或多项：

(1) 私募/公募

中小企业私募债信息披露有限。评级机构、承销商、会计师事务所、律所等获取的基础资料均来自于发行人，中介机构只是在职责范围内和作为从业人员一般的知识能力水平下对资料进行鉴别。

(2) 城投/非城投

因为城投债具有整体属性，为了维持地方城投的担保信用等原因，城投债或者城投以不可撤销连带责任担保的债券较少违约。2015 年 7 月 18 日，财政部副部长表示，政府允许部分企业破产，但必须避免对地方政府产生任何负面影响，引发系统性风险。

例如，“13 大宏债”违约后，射阳城投本不愿承担担保责任，在射阳政府得知情况后，责成新的领导班子偿付。

(3) 上市/非上市

由于上市企业具有的壳价值，特别是在 IPO 审核制的情况下，因此能获得更多的外部救助。例如 12 湘鄂债最终兑付成功。

(4) 实际控制人背景

实际控制人为央企的发行人，在违约后的兑付中表现更积极。例如二重重装违约后，其实际控制人中国机械工业集团有限公司受让债务。

(5) 是否具有品牌或规模优势

当发行人具有品牌和规模优势的时候，获得母公司等救援的可能性较大。例如'15 雨润 CP001'中，**"雨润"品牌为驰名商标**；'16 博源 SCP001'中，发行人博源控股集团是国内五大纯碱生产企业之一，**控制了国内 90%**的天然碱资源，同时是小苏打行业龙头企业、天然气制甲醇产能规模国内排名第二。

(6) 母公司持股比例、财务状况及行业地位

当母公司持股比例较大以及财政状况和行业地位高时，对发行人救助的可能性较大。例如'15 华昱 CP001'中，实际控制人中煤能**源持股规模高达 49.46%，财务状况好并且是煤炭行业仅有的两家央企之一**。反之，在'10 中钢债'违约事件中，中国中钢股份有限公司的母公司中钢集团由于**自身财务情况不好，难以给公司提供有力的财务支持。**

3. 债券违约归因---宏观流动性角度

通过对债券违约事件违约原因进行归因，可以得到一系列特征来衡量企业经营、管理能力，以及衡量行业、区域、外部支持等外在因素的特征。由于债券违约归根结底的原因是流动性丧失，因此，还应从**宏观经济总体流动性**的角度出发来寻找被**遗漏的特征**。

(1) 内生原因

1) 经济

(A) GDP 增速：一方面，若 GDP 增速下降，经济下行，需求不振等内生因素导致的**风溢价上升**，私人部门持有现金意愿增加，从而削减市场流动性，**加大发行人违约的概率**；另一方面，由于发行人融资成本一部分取决于信贷资金成本，而信贷资金通过调节国民经济各部门间的资金余缺，不可避免的参与了国民财富的创造过程，当对国民财富进行分配时，信贷资金作为资本的一部分，享受经济增值的成果，因此发行人融资成本又与 GDP 增速成正相关关系，**若 GDP 增速上升，融资成本上升最终会助推经营不善的企业违约。**在著名的**泰勒规则**中，以美联储名义利率政策的制定为例，联邦基金利率指标=通货膨胀率+均衡实际联邦基金利率+1/2（通货膨胀缺口）+1/2（产出缺口），**即名义利率与产出缺口成正比。**

(B) CPI 增速：CPI 增速加大将**降低实际利率**，即降低企业再融资成本。

2) 金融

(A) 金融创新：随着近几年金融创新迅速发展，主要以影子银行形式存在，一方面金融创新的增速**反映了实体经济融资缺口的增速**，代表了实体经济流动性的松紧程度；同时，金融创新拉长了部分资金在金融体系内的运动过程，**推升资金最终到达实体经济的成本**，加大企业债务负担；最后，由于**逆向选择**的存在，愿意以高成本融资的企业本身资质较差。因此，金融创新增速的加大，**企业违约概率增加**。另一方面，作为**银行信贷缺口的补充**，可以弥补银行信贷向大企业倾斜的缺陷，缓解小微企业的流动性压力，**减少违约的概率。**因此，将反应金融创新程度的委托贷款增速、信托贷款增速以及未贴现银行承兑汇票增速纳入到模型特征中。

(B) 资产价格：由于信息不对称导致金融市场存在**金融摩擦**，所以一般融资活动需要抵质押品，而资产价格直接决定了抵质押品的价值，从而决定了企业融资成本和融资额度。并且，在**金融加速器效应下**，资产价格冲击对企业投资经营活动的影响会通过企业资产负债表放大。因此，资产价格对企业是否违约影响重大。资产价格不仅对单个企业融资影响巨大，更是直接影响到市场流动性。**特别是在危机时期，**资产价格大幅下跌引发踩踏，会带来整个市场流动性暂停。应将资产价格纳入到模型中，本研究选取房价指数增速和上证指数增速。

(2) 外生因素

1) 货币政策

货币当局在稳健中性货币政策总基调下，积极运用各种货币政策工具的维持市场流动性适中。因此，应将央行货币政策紧密相关的变量例如 M2 增长率、社会融资规模增长率以及流动性调节工具的利率纳入到模型特征中。

(A) M2 增长率：货币供应量是我国货币政策的**中介目标**。

(B)社会融资规模：随着直接融资占比不断升高，非银金融机构作用不断加强以及银行表外业务的持续增加，以货币供应量 M2 来衡量市场流动性的有效性逐步下降。社会融资规模不仅包括银行贷款，还包括金融机构表外业务、直接融资、保险公司与小额贷款公司业务等，能够比 M2 **更加全面的反映金融与实体经济的关系**。自 2010 年 12 月中央经济工作会议上被提出以来，其度量作用的重要性不断增强。在 2016 年政府工作报告中，已将社融融资规模余额增速与 M2 增速并列。

(C) 流动性调节工具利率：随着利率市场化的完成，我国货币政策正在尝试**从数量型**

向价格型转变。借鉴国际经验，央行正则积极构建和完善政策利率体系，央行以此引导和调控包括市场基准利率和收益率曲线在内的整个市场利率，以实现货币政策目标。**对短期利率**的调控我国主要参考欧央行的**利率走廊（图 13）**模式，其公开市场利率是利率操作目标，常备借贷便利（SLF）和超额准备金利率则分别为上下限。**而对中长期利率的调控**，我国央行则主要通过中期借贷便（MLF）、抵押补充贷款（PSL）等工具来实施。因此，本课题将 7 天逆回购利率、常备借贷便利利率以及超额存款准备金利率以及中期借贷便利利率纳入到宏观层面的特征中。

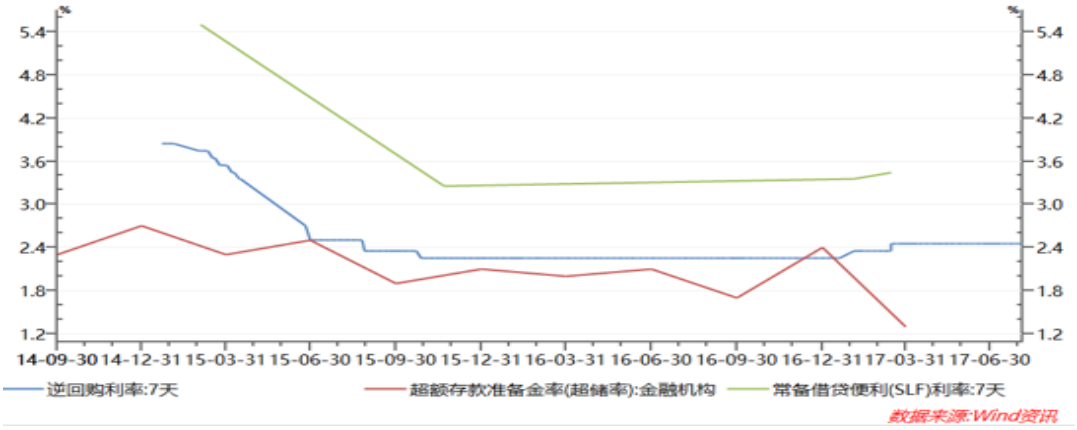


图 13 利率走廊

4. 债券违约归因---重要特征表

综上，本研究初步梳理出一系列特征以用于对债券违约事件的预测。其中，鉴于数据的可得性以及时间的限制，有些特征并没有被纳入数据收集过程，然而有些未列入当前数据收集过程的特征十分重要，例如实际控制人是否存在负面事件、下游行业景气程度等，因此进一步的数据收集也是本课题之后应该进行的方向。并且考虑到后面会有特征工程进一步筛选特征，因此在样本数据集的特征中添加了许多在本表中未出现的特征。**因此，本表给出的特征来自于主观对事件分析得出的，与债券违约高度相关的因素。而样本特征则是在可得的范围内尽量囊括多的特征，一方面包括了不在主观分析中的特征，一方面损失了主观分析中难以获得样本数据的特征。**

表 2 违约债券重要特征参考表

	特征	类型		特征	类型
宏观层面	CPI 增长率	numeric	宏观层面	行业	catogary
	M2 增长率	numeric		行业景气程度	numeric
	社会融资规模余额增速	numeric		行业是否存在限制性政策	boolean
	上证指数增速	numeric		shibor	numeric

	房价指数增速	numeric		国际石油价格增速	numeric
	7 天逆回购利率	numeric		省份	catogary
	超额存款准备金利率	numeric		地方 GDP 增速	numeric
	常备借贷便利：7 天	numeric		地方财政可支配收入	numeric
	中期借贷便利利率	numeric		下游行业景气程度	numeric
企业层面——财务指标	净利润增速	numeric	企业层面——财务指标	现金满足投资比率	numeric
	对外担保额度/净资产	numeric		现金营运指数	numeric
	预付款/净资产	numeric		长期资产适合率	numeric
	应收账款/净资产	numeric		权益乘数	numeric
	存货/净资产	numeric		资本固定化率	numeric
	速动比率	numeric		产权比率	numeric
	经营性现金流	numeric		有形净值债务率	numeric
	资产负债率	numeric		资本项目维持率	numeric
	现金比率	numeric		应收账款/流动资产	numeric
	ROA	numeric		流动负债/负债合计	numeric
	流动比率	numeric		产能利用率	numeric
	应收账款周转率	numeric		利息保障倍数	numeric
	存货周转率	numeric		质押资产/总资产	numeric
	受限资产/总资产	numeric			
	债务增速	numeric			
	银行贷款/总债务	numeric			
企业层面——经营管理指标	主承销商类别（银行、证券、地方股权交易所）	catogary	企业层面——经营管理指标	已发债数量	numeric
	是否存在停产整治等重大事件	boolean		最新评级	catogary
	是否存在借款逾期	boolean		最新评级下调跨度	numeric
	涉诉件数	numeric		控股股东是否变更	numeric
	增信机构性质（民企、央企、地方国企城投、地方国企非城投）	catogary		是否为上市公司	boolean
	债券利率	numeric		是否已发生债券违约	boolean
	实际控制人是否存在负面事件	boolean		控股股东企业性质	catogary
	实际控制人股权占比	numeric		是否存在高管职务变动	boolean
	控股母公司财务情况	numeric		是否存在股权纠纷或旗下资产股权转让	boolean
	担保机构涉讼情况	numeric		资产质押比例	numeric
	担保公司杠杆率	numeric		债券类型	catogary
	失信次数（国家失信被执行人）	numeric		上市子公司股价增长率	numeric

	拥有上市子公司股权占比	numeric		对母公司收益贡献百分比	numeric
	是否存在品牌、规模优势	boolean		对母公司净资产贡献百分比	numeric
	评级机构	catogary		是否为城投公司	boolean
	行业平均资产负债率	numeric		母公司持股比例	numeric

二、数据收集与特征工程

本部分首先收集 2017 年 7 月 31 日前到期的所有公司债，并通过 Wind 量化接口获取企业层面指标，删去空值占比较大的特征和样本后，通过匹配剩余每个样本特定的到期/违约时间和特征来填充宏观层面指标，形成数据集。在此基础上，对数据集进行描述性统计分析以初探数据集属性，**一方面为后续预测模型的选择以及模型效果的评判提供依据，另一方面对预测模型的透明性和可信性提供参考。**最后，采取 Lasso 回归从客观上提取部分特征，防止预测模型出现过拟合的问题。

（一）数据收集与处理

鉴于本阶段研究时间限制，拟打算从全部违约债券中选取公司债作为研究对象，初始样本为 2017 年 7 月 31 日（含）前到期的**所有公司债共 873 只，其中包括所有已违约的公司债共 30 只**。数据获取与匹配步骤如下：

1. 从 Wind 数据库中导出全部已到期公司债表格，包含选中的指标类别有债券基本资料、发行兑付资料、信用分析指标。
2. 将全部到期公司债与违约公司债进行匹配，若到期债务中存在违约债券，则将其到期时间替换为违约时间（后面简称为时间）；**同时对全部债券进行编码，未违约编码为 0（负样本，SVM 中为-1），违约编码为 1（正样本）。**
3. 为了得到债券违约或到期时点之前最新的财务分析指标，采用 Wind 量化接口，通过循环公司债时间来获得一年半以内最新的财务指标。
4. 通过观察取出的财务指标数据发现，有许多财务指标控制较多，由于存在大量空值特征会降低分类器的预测效果，因此查看财务指标中空值占比排序，删掉空值占比大于 15%的财务指标。并且删掉财务指标全部为空的到期（不包含违约）公司债，剩下样本量为 533 只到期或违约债券。

简单的删掉空值大于 15%的财务指标存在的问题是，有些财务指标对债券违约的判可能很重要，例如图 6 中的 ocftonetdebt(经营活动产生的现金流量净额/净债务),其占比为 15.0094%，但仍被排除在预测所使用的特征之外。因此**对于阈值的选取需要更多的试验。**

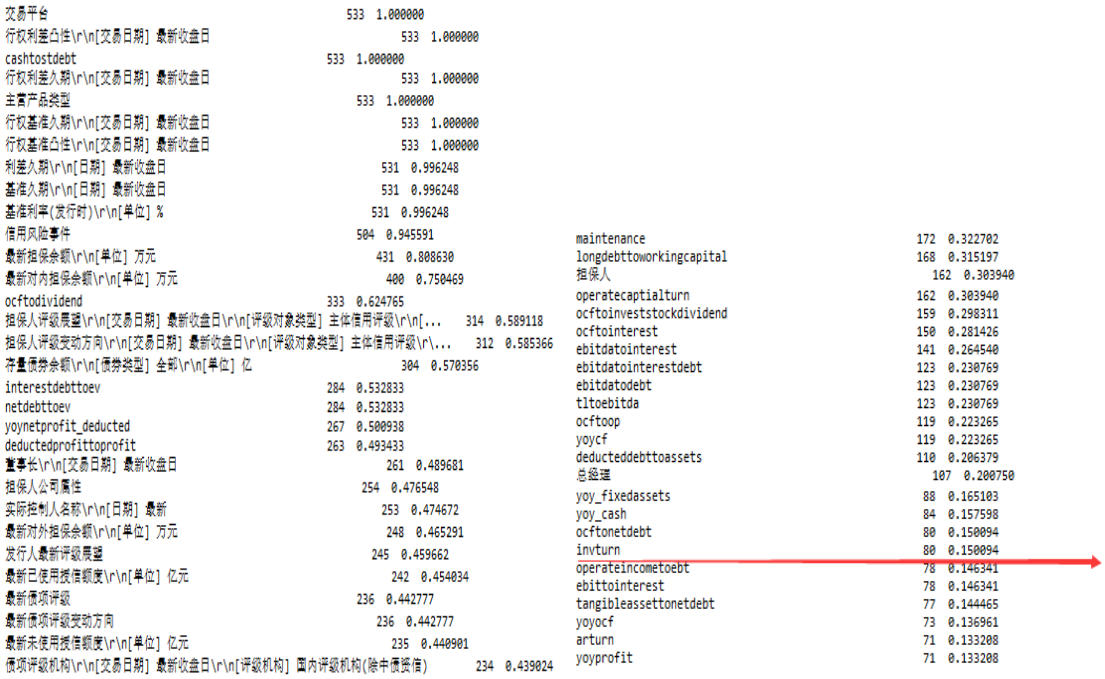


图 14

5. 将财务数据指标与原指标合并之后开始填充宏观层面和地区层面的指标。对于时间序列结构的指标：GDP 增速、CPI 增速、房价指数增速、社会融资规模余额增速、委托贷款增速、信托贷款增速、未贴现银行承兑汇票增速、M2 增速、国际石油价格增速、逆回购利率：7 天、超额存款准备金率、常备借贷便利利率以及上证指数增速，根据债券到期或违约的时间，选择在此时间前半年至 1 年的相应指标数据填充样本。对于面板数据结构的指标：地方 GDP 增速、地方财政净收入、以及相应行业增速，先根据样本指标中的省份或行业进行匹配，再根据时间，选择在此时间前半年到 1 年的相应的数据填充样本。
6. 观察样本发现，总共 30 只违约公司债中有 17 只私募债在违约前 1 年半内无任何财务数据，为此，将数据区间推前 10 年，又得以取出 9 只违约债券在前 1.5 年到 10 年内最新的财务数据。因此，可以认为预测模型考虑了一定程度的财务粉饰。

（二）特征工程

在机器学习的具体实践任务中，选择一组具有代表性的特征用于构建模型是非常重要的问题。“Coming up with features in difficult, time-consuming, requires expert knowledge. Apply machine learning is basically feature engineering”(Andrew Ng)。本部分先对数据集进行描述性统计分析，删去类别特征中变量较多的特征，并只保留相关性较高的数量特征中的一项。然后利用 Lasso 回归客观提取特征，以减少模型的过拟合问题。最终形成预测模型的

输入。

1. 了解数据集

（1）数据集有 533 个样本，99 个特征

1) 样本的个数大于特征的个数，可以选择复杂的分类器（机器学习模型）

2) 533 个样本中有 30 个正样本，503 个负样本，因此为非平衡样本。对于**非平衡样本**，在后面划分训练集个交叉验证集的时候应该选择分层抽样的方式；在后面对分类器预测效果的判断时，不应该关注 ROC-AUC，应选择 PRC-AUC 作为分类器预测效果的度量指标，并且，由于在实际应用场景中，我们对将非违约样本预测为违约样本要比把违约样本预测为非违约样本的容忍度高，因此还应该重点关注 TPR。没有进行上下采样使样本均衡的原因是，如果应用场景是 1：100，那训练集合最好也是 1：100，所以这里**保持训练集的样本比例**。

（2）数据集有数量型特征 90 个，类别型特征 9 个

1) 类别型特征：

类别型特征包括省份 (30 个省)、公司属性 (9 个)、发行时债项评级 (7 个)、Wind 债券二级分类（2 个)、所属 Wind 行业名称（89 个)、是否上市公司（2 个)、是否含权债券 (2 个)、是否城投债 (2 个)、大股东类型 (10 个）。可以看到省份一级所属 wind 行业名称包含的类别较多，如果一些特征包含的类别较多，一方面不能满足一些分类器的要求，例如由 Breiman 和 Cutler 写的随机森林算法中，一个特征包含的类别不能超过 32 个；另一方面，类别型变量在作为模型输入前，要进行 0-1 编码转换为数值型特征。若一个特征包含 89 个类别，则相应回增加 88 个特征，加大模型的自由度，减少模型样本外预测的能力，并且在特种中包括了对应省份财政净收入、GDP 增速一级对应行业的景气指数，因此删去这两个类别特征。

2) 数量型特征：

采用机器学习算法得到的响应函数是非线性的，因此无法使用一个单一的系数来描述自变量与因变量间的关系，**模型缺少透明度**。虽然我们可以使用交叉验证、错误率来评估模型。但是为了增加对模型的信任和理解。**在建模之前一般要对数据集进行一定了解。主要是对数据进行可视化。**一个准确的机器学习模型给出的预测，应当能够反映出数据集所体现的结构和相关性。要明确一个模

型给出的预测是否可信，应对数据集中呈现出的结构和相关性进行观察（不过大部分机器学习算法自动提取变量之间的高阶交互作用）。**同时也可以帮助我们将重要变量引入机器学习模型中。**

在对数量型特征进行**归一化处理**后，用 **Parallel Coordinates Plots 可视化**数量型征。下图中，**红线代表违约债券样本，蓝线代表未违约债券样本。**

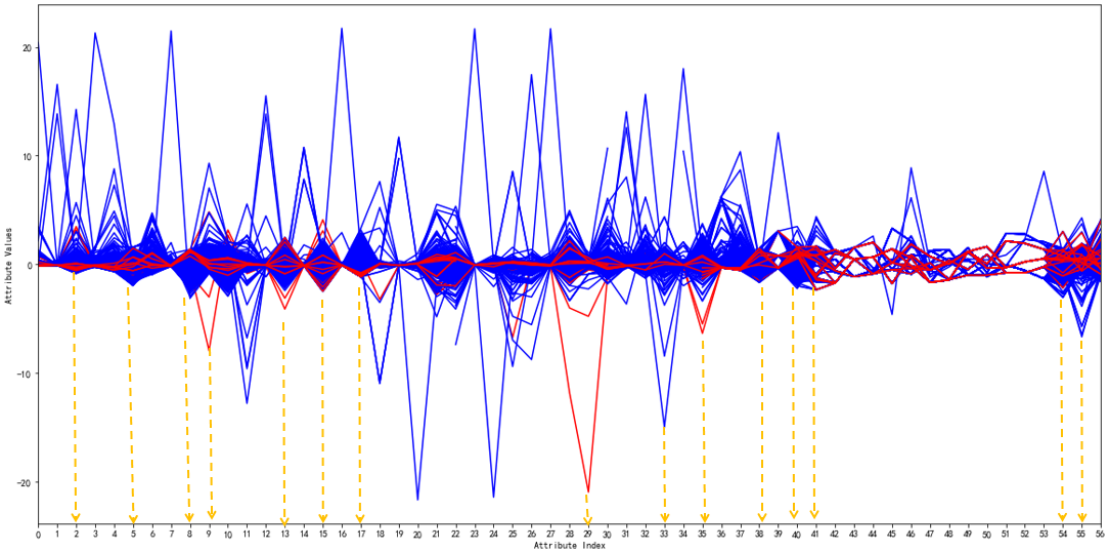


图 15 平行坐标轴图

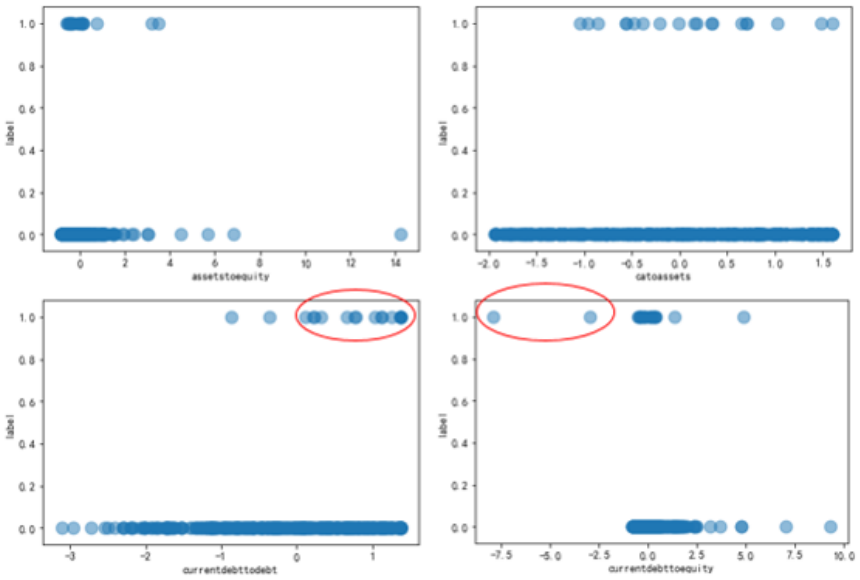
平行坐标轴图

由图 15 可以看到，在特征 2（权益乘数）、特征 5（流动资产/总资产）、特征 8（流动负债/负债合计）、特征 9（流动负债权益比）、特征 13（归属母公司股东的权益/全部投资资本）、特征 15 （带息债务/全部投入资本）、特征 17（长期负债占比）、特征 29 （净资产收益率）、特征 33（经营活动产生的现金流净额（同比增长率））、特征 35（z_score）、特征 38（大股东持股比例）、特征 40（票面利率(发行时)）、特征 41（CPI 增速）、特征 54（对应省份的财政净收入）、特征 55（对应省份的 gdp），共 15 个特征处，正样本与负样本有明显的不重合。

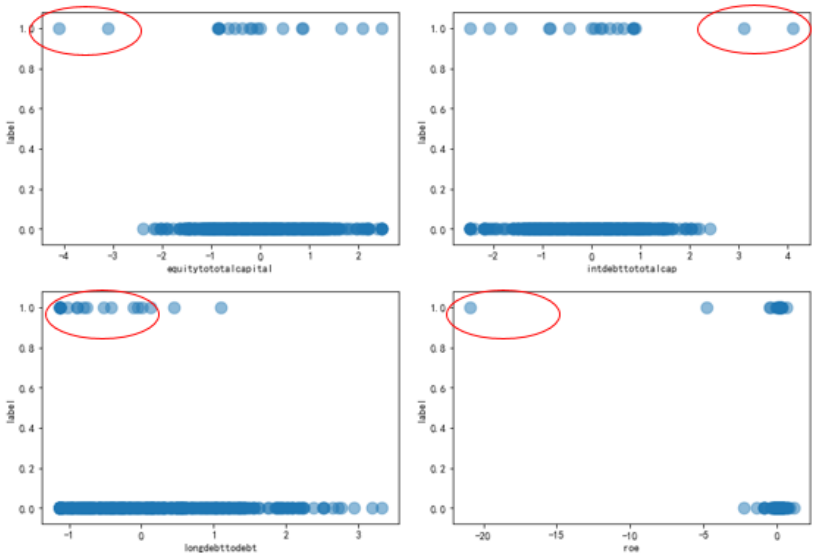
因此，**这里猜想这 15 个变量在预测中起到了关键作用**。并且，从不重合的处可以对特征与标签（1：违约；0 不违约，当采用 SVM 作为分类器时，-1 不违约）的关系进行猜测。例如，在特征 41（CPI 增速）处，可以发现，违约样本的 CPI 增速多集中在 CPI 增速均值 0 之下，因此可以猜测当 CPI 增速下降时，债券违约的概率加大。同理可以对其他特征与标签的关系进行初步猜测。

散点图

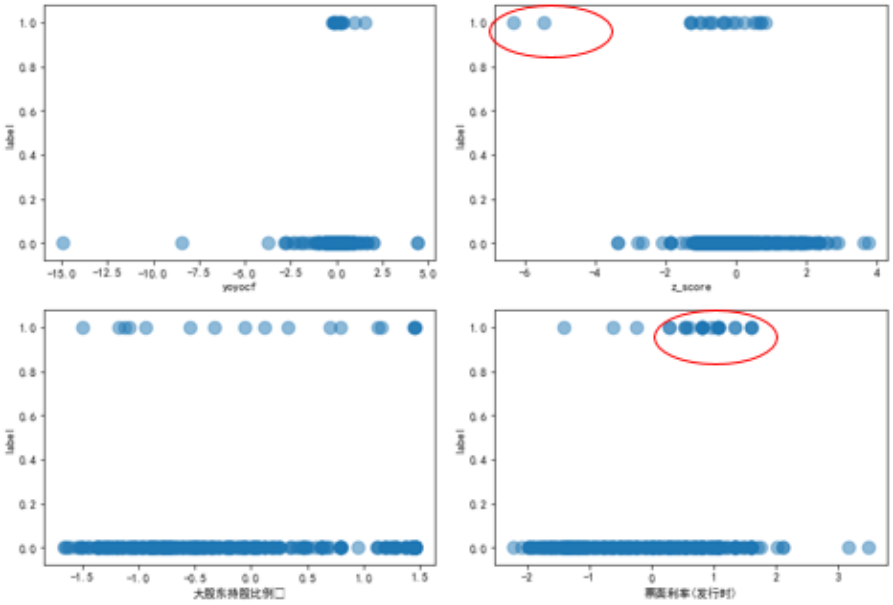
为了进一步探究上述 15 个特征与标签之间的关系，下面分别画出每个特征与标签之间的散点图，通过观察每个特征违约样本（1）与未违约样本（0）的分布，可以对特征与标签间的关系进行猜测。



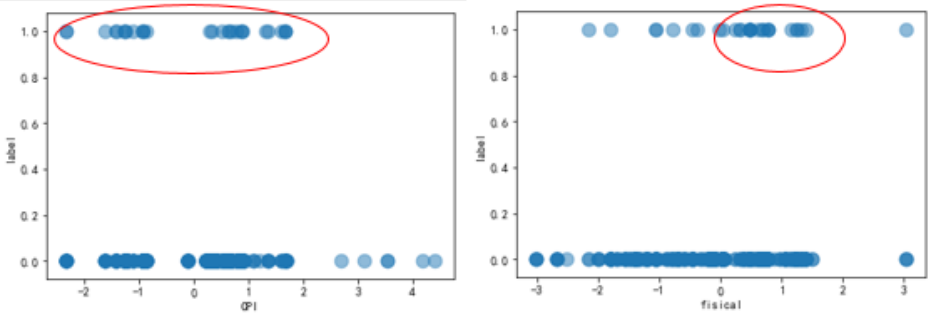
我们可以看到在特征 2（权益乘数）中，违约样本点和未违约样本点的分布并没有显著差异；而在特征 5（流动资产/总资产）中，违约样本点的分布竟相对于未违约样本点明显偏右，与常规分析思路不一致，因此，特征 5 在模型预测中不靠谱；在特征 8（流动负债/负债合计）中，违约样本点分布相对于未违约样本来说明显偏右，与常规分析思路一致，因此特征 8 应保留在模型中。特征 9（流动负债权益比）中，虽然违约债券样本数较少，但是样本分布偏左，与常规分析思路不一致，因此此特征在模型预测中不可靠。

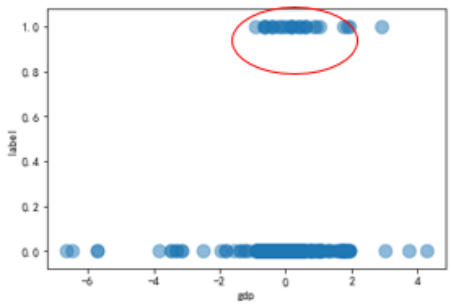


在特征 13（归属母公司股东的权益/全部投资资本）在左边出现了极端值，因此判断在模型预测中有作用；在特征 15（带息债务/全部投入资本）在右边出现了极端值，与常规分析思路一致，因此认为在特征等 15 模型预测中可靠；在特征 17（长期负债占比）中，在左边出现极端值，与常规分析思路一致，因此认为特征 17 在模型预测中可靠；特征 29 （净资产收益率）中，违约样本在左端出现极端值，与常规分析思路一致，因此认为特征 29 在模型预测中可靠。



特征 33（经营活动产生的现金流净额（同比增长率））中，违约样本的分布偏右，与常规分析的思路不一致，因此认为特征 33 在模型预测中不可靠；特征 35（z_score）中，违约样本的分布偏左，且在左边出现了极端值，与常规分析思路一致，因此认为特征 35 在模型预测中可靠；在特征 38（大股东持股比例）中，违约样本的分布于未违约样本的分布没有显著差别，因此认为特征 38 在模型预测中作用不大；在特征 40（票面利率(发行时)）时，违约样本密集分布在右边，因此特征 40 在模型预测中十分重要。





在特征 41（CPI 增速）中，违约样本分布偏左，因此认为在模型预测中可靠；而在特征 54（对应省份的财政净收入）与特征 55（对应省份的 gdp）中，违约样本分布偏右，因为样本在区域中分布不均匀，因此并不好判断这两个特征的预测效果。

Pearson 相关系数

除了通过观察平行坐标轴图和散点图来发现特征与标签之间的相关性，本研究进一步计算特征间以及特征与标签间的 Pearson 相关系数,并绘制出热图(图 16)，一方面删除一些相关性高的特征以避免多重共线性，另一方面选择与标签相关性高的特征用于模型的预测。在热图中，颜色越暖意味着相关性越高。图 8 是与标签相关性的前 23 个特征。从图 19 可以发现，与标签相关性最高的前 23 个特征中，除了 roe、票面利率、z_score、流动负债/负债合计、年化 roa、长期负债占比、资产负债率、现金比率，流动负债权益比这九个微观特征外，其他均为宏观特征。**因此，在对债券违约作出判断时，应充分考虑宏观因素的影响。**不过，本阶段可以的缺陷之一是，由于第一只债券违约时间为 2014 年，因此，宏观指标并没有包含一个完整的经济周期。

```
label                 1.000000
roe                  -0.254139
票面利率(发行时)           0.180508
z_score              -0.171912
reverseRepo           0.163716
currentdebttodebt     0.155714
roa_yearly           -0.142288
entrustLoan           0.128361
longdebttodebt       -0.127469
houseprice           -0.124498
undiscounted          0.121160
gdpCountry            0.118934
SLF                   0.116525
industry              0.114595
发行总额\r               -0.094343
stock                 0.092295
debttoassets          0.087351
gdp                   0.081666
oilPrice             -0.081061
fisical               0.076719
cashtocurrentdebt    -0.075775
tsf                   0.075318
已发债总数                -0.069683
currentdebttoequity  -0.060864
```

图 16

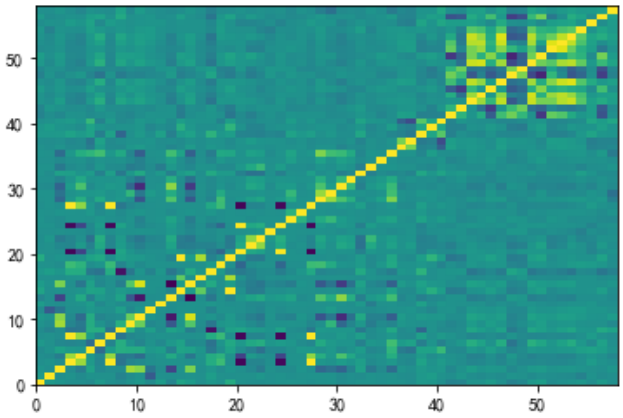


图 17

从图 17 可以发现，前面平行坐标轴中违约与不违约样本有明显差距的特征，基本上均出现在与标签（label）相关系数较高的特征变量里：首先，逆回购利（reverseRepo）、常备借贷便利利率（SLF）与债券违约呈正相关关系，即**货币市场短期利率越高，违约概率越大**；其次，委托贷款增速（entrustLoan）、未贴现银行承兑汇票增速（undiscounted）与社会融资规模增速（tsf）与债券违约呈正相关，也就是**金融创新并没有缓解流动性缺口，而是反映了融资缺口或推高资金价格和逆向选择问题**；然后，房价（houseprice）以及石油价格（oliPrice）与债券违约呈负相关关系，股票价格（stock）与债券违约呈正相关关系。因此，**为防范资产价格下跌带来的风险，资管资产组合配置中不适合同时配置大规模的房地产资产和债券资产**，因为一旦房价下跌，债券违约概率也将增大。相反，可以多开发债券资产与股票资产组合的资产池；最后，全国 GDP 增速（gdpCountry）、地方 GDP 增速（gdp）、以及地方财政净收入增速（fisical）均与债券违约呈正相关关系，**一方面可能是原因是 GDP 增加导致信贷资金成本增加大于 GDP 增加减少的风险溢价；另一方面可能的原因是样本时间跨度较短，**2014~2017 仅有 4 年数据，并未跨度一个经济周期；另外，区域发债数目不一致，经济发展较好的区域本身发行债券的基数较大，其中不可避免的会有一些资质较差的发行主体，因此样本不均衡可能影响到地方 GDP 增速和地方财政净收入与债券违约的关系。最后一个可能性则是，发行人所在区域的统计口径为注册地，可能与发行人主营业务所在地存在差异。

从图 17 可以发现，如果将相关系数绝对值超过 0.75 判定为具有高度相关性，通过相关系数矩阵发现流动负债/负债合计与长期负债占比相关系数为-0.911，资产负债率与带息债务/全部投入资本、有形资产/负债合计、归属母公司股东的权益/全部投资资本的相关系数分别为 0.878、-0.761、-0.878，固定资产周转率与非流动资产周转率相关系数为 0.996、非筹资性现金净流动/流动负债与现金比率、速动比率、流动比率、经营活动产生的现金流量净额/流动负债相关系数分别为-0.969、-0.984、-0.976、0.99、SLF 与未贴现银行承兑汇票、7 天逆回购利率、GDP 增速以及社会融资规模余额增速的相关系数分别为 0.835、0.828、0.804、0.779、委托贷款增速与 7 天逆回购利率以及社会融资规模余额增速的相关系数分别为 0.861、0.794、上证指数与百房价格指数的相关系数为-0.804、信托贷款与社会融资规模余额增速的相关系数为 0.903。

综上，将 Parallel Coordinates Plots、特征与标签的相关性以及特征间的相关系数结合，保留与标签相关性高的特征，删掉特征间相关性高的一部分特征，构成我们初步筛选

出的特征集。

2. 特征选择---Lasso Regression

当我们进行特征的初步筛选后，特征数仍有 90 个。**特征太多，可能会使噪声得到了过分的关注，造成过拟合问题，导致模型泛化能力下降。**对于最优预测模型来说，既不能出现过拟合，也不能出现拟合不足（特征较少），拟合不足或过拟合均会造成样本外预测精度下降，因此特征选择十分重要。如图 18 示例，黄线对样本的拟合不足，而蓝线对噪声过分的关注出现过拟合问题。

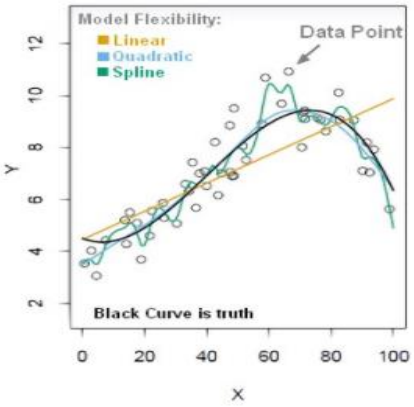


图 18

特征选择是指选择获得相应模型和算法最好性能的特征集。对于特征的选择，可以采取传统的主观上挑选。然而当特征数量很多时，例如，在本研究中有 90 个特征，变量间的高阶交互很难考虑全面。**特别是作为风控，而不是行业研究员，不可能对所有行业都去研究后主观提取特征。而基于客观特征提取**，即利用模型来选择特征，如果提取的特征集中已经包括“行业”这个变量，那么模型选出来的一系列特征已经考虑到了与行业的交互作用，**既提高了后续预测模型的精度，同时可以增强对特征与特征间关系的理解。**

特征选择模型的关键是可以产生**稀疏解**，稀疏解可以找到有用的特征并且减少冗余，提高回归预测的准确性和鲁棒性（减少过拟合问题）。带惩罚项的回归通过最小化带约束的目标函数，来找到给定数据集中的一组最优回归系数。**L1/LASSO 惩罚把无足轻重的回归系数拉回到 0**，能够选取一小部分有代表性的变量进行回归系数估计，供线性模型使用，从而规避了那些靠前向、后向、逐步变量选择法（forward, backward, stepwise variable selection）所带来的多模型比较困难的问题。如图 19 所示，在 L1 约束下，参数集中分布在各坐标轴附近。

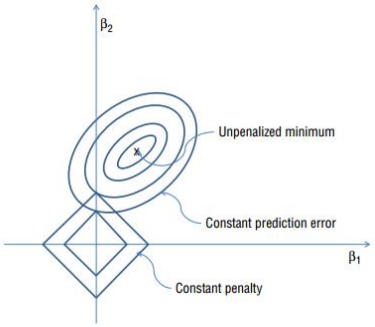


Figure 4-2: Optimum solutions with sum absolute value coefficient penalty.

图 19 Cost Function: $\frac{1}{2m}\|\boldsymbol{y}-\boldsymbol{X\beta}\|_2^2+\boldsymbol{\lambda}\|\boldsymbol{\beta}\|_1$

Lasso 回归的主要解法有两种，最小角回归（LARS）算法和 Glmnet 算法（$\alpha=1$）

最小角回归算法：

1. 初始化参数β，nsteps，stepSize

2. 对 m=1,2……nsteps:

(a)计算每个特征与残差间的相关性

(b)将相关性最大的特征的系数做相应的增加或减少 stepSize，取决于相关系数的正负。

Glmnet 算法用于解决 ElasticNet 问题：

$$\min_{\boldsymbol{\beta}}\frac{1}{2m}\|\boldsymbol{y}-\boldsymbol{X\beta}\|_2^2+\alpha\boldsymbol{\lambda}\|\boldsymbol{\beta}\|_1+\frac{\boldsymbol{\lambda}(1-\alpha)}{2}\|\boldsymbol{\beta}\|_2^2$$

Lasso 回归与 Ridge 回归同属于一个被称为 ElasticNet 的广义线性模型家族。这一家族的模型除了相同作用的参数λ之外，还有另一个参数α来控制应对高相关性(highly correlated)数据时模型的性状。**Lasso 回归 α=1**，Ridge 回归 α=0，一般 ElasticNet 模型 0<α<1。Glmnet 算法是一种坐标轴下降法，在每次迭代中可以在当前点处沿一个坐标方向进行一维搜索。在整个过程中循环使用不同的坐标方向，一个周期的一维搜索迭代过程相当于一个梯度迭代。梯度下降（Gradient Descent）利用目标函数的导致来确定搜索方向，而该梯度方向可能不与任何坐标轴平行。而坐标轴下降方法利用当前坐标轴进行搜索，不需要求目标函数的导数，只按照某一坐标方向进行搜索最小值。因此，坐标下降法在稀疏矩阵上的计算速度非常快，同时也是 Lasso 回归的最快的解法。

Glmnet 算法：

1.初始化参数β,nsteps,$\alpha\lambda_0$=max($\frac{1}{m}\sum_{i=1}^{m}x_{ij}r_i+\beta_j^-$),lamMult=0.93(算法提出者 Friedman 推荐)

2.对 m=1,2……nsteps

(a)更新$\alpha\boldsymbol{\lambda}=\alpha\lambda_{m-1}*\mathrm{lamMult}$以及参数$\beta_j \leftarrow \frac{S(\frac{1}{m}\sum_{i=1}^{m} x_{ij}r_i+\beta_j^{-},\alpha\boldsymbol{\lambda})}{1+\boldsymbol{\lambda}(\mathbf{1}-\alpha)}$,其中 S()如图 20 所示

(b)当β_j^{-}数值不在变化后，算法得到与α和λ对应的一个解β_j。

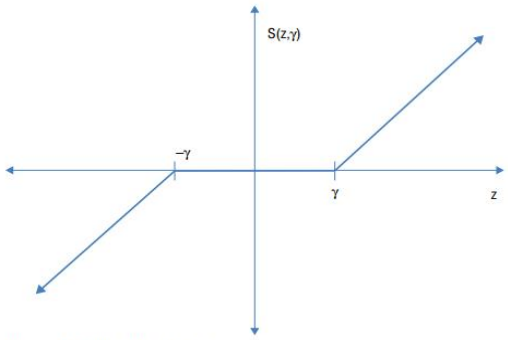


Figure 4-5: Plot of S() function

图 20

运用 Glmnet 算法求解债券违约的 Lasso 回归问题，得到如下参数路径（图 21），随着惩罚因子λ变小，越来越多的参数进入模型。**其中，越早进入模型的特征，在预测中的重要性越大**：

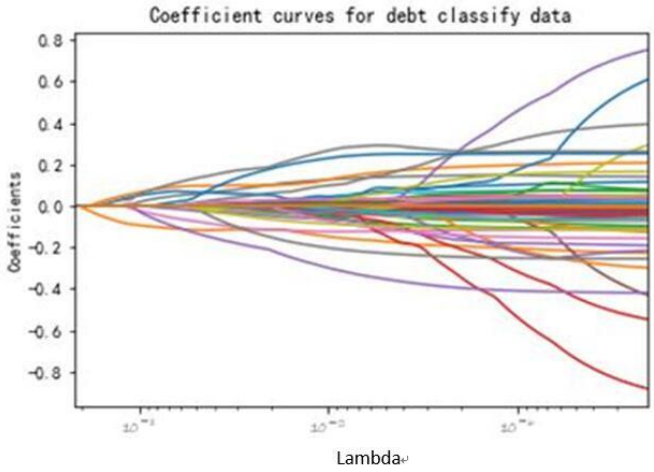


图 21

具体来看，**特征进入模型的顺序为：**

'roe', '票面利率(发行时)', 'reverseRepo', '公司属性\r_民营企业', '大股东类型\r_个人', 'z_score', 'currentdebttodebt', '大股东类型\r_民营企业', '公司属性\r_地方国有企业', '公司属性\r_中外合资企业', 'gdpCountry', 'houseprice', 'gdp', '是否含权债_否', 'industry', '大股东类型\r_中外合资企业', '是否含权债_是', 'currentdebttoequity', 'faturn', '公司属性\r_外商独资企业', '大股东类型\r_外商独资企业', 'yoydebt', 'oilPrice', '发行时债项评级_AA+', '发行时债项评级_AA-', 'netprofitmargin', 'operateincometoebt', '大股东持股比例\r', **'公司属性\r_公众企业',** 'catoassets', 'M2', '发行时债项评级_A', 'yoyprofit', 'entrustLoan', 'optoebt', 'roa_yearly',

'tangibleassettonetdebt', **'fisical',** 'assetstoequity', 'longcapitaltoinvestment', 'trustLoan', 'ocficftodebt', 'yoyocf', '发行总额\r', '发行时债项评级_A+', '大股东类型\r_公众企业', 'caturn', 'Wind 债券二级分类_一般公司债', 'Wind 债券二级分类_私募债', '每年付息次数', 'ocftoassets', 'arturn', 'debttoassets', 'tsf', '公司属性\r_其他企业', '大股东类型\r_中央国有企业', 'CPI', 'SLF', 'undiscounted', '公司属性\r_中央国有企业', '大股东类型\r_外资企业', 'apturn', 'debttotangibleequity', '是否上市公司_否', '是否上市公司_是', '发行时债项评级_-', '公司属性\r_集体企业', '大股东类型\r_集体企业', 'excessReserve', '大股东类型\r_其他企业', '是否城投债_否', 'stock', '已发债总数', '发行时债项评级_AAA', '是否城投债_是', '公司属性\r_外资企业', 'cashratio', 'ebittointerest', 'cashtocurrentdebt', '大股东类型\r_地方国有企业', 'ocftoquickdebt'

其中**'公司属性\r_公众企业'**和**'fisical'**是进入模型的第 30 和第 40 个特征，在选择最终进入模型特征数目时，尽量多包括在图 18（特征与标签相关性排名前 23 个）中的变量。**为了寻找最优预测，本研究分别用上面前 30 个特征和前 40 个特征进行预测实验，以期望得到最优的样本外预测模型和特征数量。**

三、模型简介与实验结果

学界与业界已运用许多量化模型来进行违约预测。Ohlson（1980）第一个提出用逻辑回归（Logistic Regression）去预测违约概率。由于信用风险研究类似于模式识别问题，近年来人工智能和机器学习等计算机技术被应用于预测信用风险（Kruppa,Schwarz,Arminger, & Ziegler, 2013; Pal, Kupka, Aneja, & Militky, 2016）。相对于逻辑回归，**机器学习方法能够更精确的拟合非线性、非单调的响应函数，从而提高预测效果（图 22）**。而在**金融经济中，非线性现象是广泛存在的。例如，金融加速器效应**，资产负债表对企业投资的影响在经济下降时期比在繁荣时期大；对小公司的作用比对大公司的作用大。另外一个例子是，**在极端情况下（金融危机），资产价格的相关性会增加**。还有就是**债券价格与利率的非线性负向关系。**

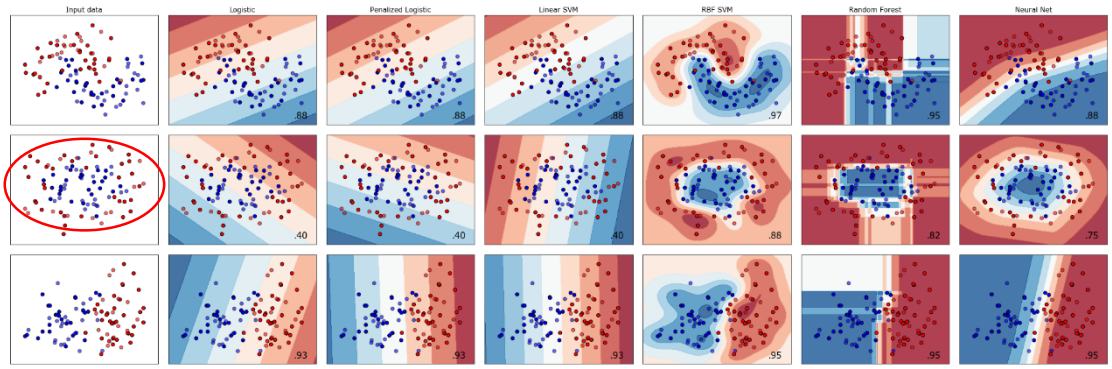


图 22

从图 14 可以看到，当正负样本非线性可分时（红圈），带高斯核的 SVM、随机森林以及神经网络比线性分类器逻辑回归等的分类效果更好。下面将对这些模型做逐一简介。

（一）模型简介

1. Regularized logistic regression

前面探讨过一个好的预测模型要克服过拟合或拟合不足的问题，因为本课题数据集特征较多，容易使模型过拟合，这里重点来说一下如何解决过拟合问题。一般从两个方面入手，一是减少特征个数，一是增加训练集个数。由于训练集个数受制于客观条件约束，因此从减少特征个数着手，正则化（regularize）就是减少特征的一个方法。通过在逻辑回归的代价函数后面加上参数惩罚项，使得一些冗余特征的回归系数非常小，削弱不相关特征对预测的影响，如图 23 所示。

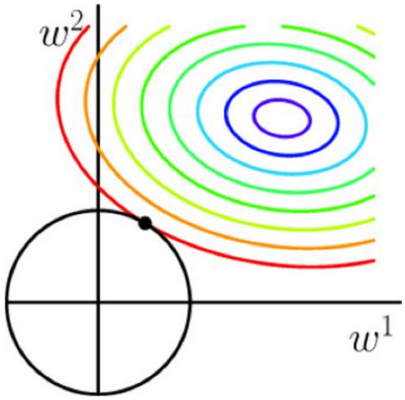


图 23

$$\min_{\beta} \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta(x^i)\right)-(1-y^{(i)})\log\left(1-h_\theta(x^i)\right)]+\frac{\lambda}{2m}\|\theta\|_2^2$$

加上参数惩罚项$\frac{\lambda}{2m}\|\theta\|_2^2$后，最小化的代价函数是由前半部分预测残差大小和参数惩罚项大小最小化共同决定，其中超参数λ决定了上面代价函数的最小值更依赖于预测残差还是参数平方和。例如，当λ很小时（$\lambda \to 0$），即代价函数中对参数平方和的惩罚较小，最小化代价函数取决于预测残差，等价于未正则化的逻辑回归，容易过拟合。而当λ很大时（$\lambda \to \infty$），代价函数对参数平方和的惩罚非常大，最小化代价函数即是让参数都非常小，容易拟合不足。因此，通过寻找一个合适大小的λ，即优化超参数λ，可以尽量避免过拟合或拟合不足的问题。

2. Support Verctor Machine

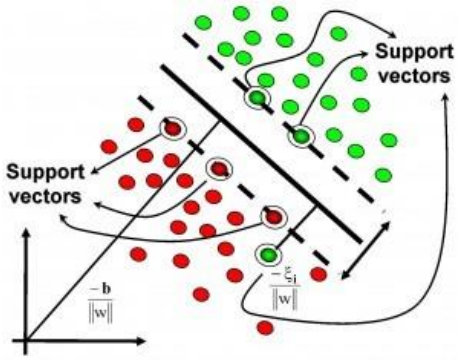


图 24

支持向量机的基本思想是求解能够正确划分训练数据集并且使几何间隔[2]最大的分离超平面（w,b），其中超平面关于训练数据集的几何间隔（γ）为超平面关于训练集中所有样本点的几何间隔最小值（$\gamma = \min_{i=1\ldots m} \gamma_i$）。训练数据集样本点中与分离超平面距离最近的样本点称为支持向量 (support vector)，由于支持向量在确定分离超平面中起着决定性作用，所以将这种分类模式成为支持向量机。支持向量的个数一般很少，所以支持向量机由很少的“重要的”训练样本确定。求解最大间隔分离超平面可表示为下面的约束最优化问题：

$$\max_{w,b} \ \gamma$$

$$\text{s.t.} \ \ y_i\left(\frac{w}{\|w\|}\cdot x_i + \frac{b}{\|w\|}\right) \geq \ \gamma \ \ i = 1,2,\ldots N$$

转化为下面的形式，即得到线性可分持向量机学习的优化问题为：

$$\min_{w,b} \ \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad y_i(w\cdot x_i + b) - 1 \geq 0 \ \ i = 1,2,\ldots N$$

与逻辑回归不同的是，支持向量机不仅要将正负样本区分开来，而且要使最难分开的点都能以足够大的确信度将它们分开。需要不仅仅是一个超平面，还要让这两个类在超平面两边尽可能的远离开来，就像两个类之间有一条沟一样，这样对未知数据也会有更好的分类预测能力。

核技巧

对解线性分类问题，线性分类支持向量机是一种非常有效的方法。但是有时问题是非线性的，非线性问题往往不好求解，所以希望能用解线性分类问题的方法解决问题，采取的方法是进行一个非线性变换（$\phi(x): \chi \to \mathcal{H}$），将非线性问题变换为线性问题，通过解变换后的

[2] 几何间隔是使得分离超平面的法向量 w 的模$\|w\| = 1$的函数间隔。

线性问题的方法求解原来的非线性问题。例如图 25 所示，通过变换，将左边椭圆决策边界变成右边的平面决策边界。

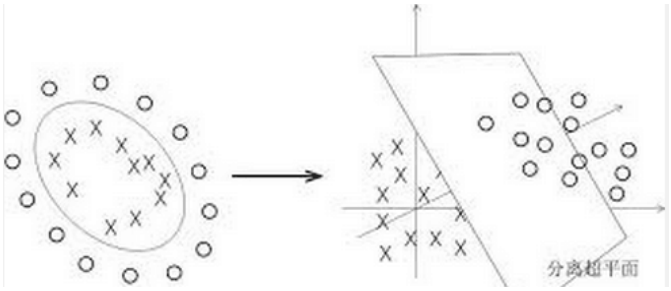


图 25

核技巧的想法是，在学习与预测中只定义核函数$\mathrm{K}(\mathrm{x},\mathrm{z}) = \phi(x)\cdot\phi(z)$，而不显示定义映射函数$\phi$，因为通常计算$\mathrm{K}(\mathrm{x},\mathrm{z})$比较容易。

3. Neural Network

理论证明，两层（带一个隐藏层）神经网络可以无限逼近任意连续函数，也就是说，面对复杂的非线性分类任务，两层神经网络可以分类的很好。因为，数据从输入层到隐藏层后，隐藏层对原始数据进行了空间变换，使其线性可分。

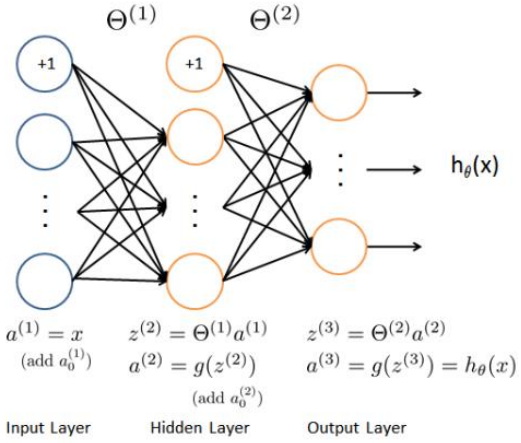


图 26

以带一个隐藏层的神经网络为例（图 26），不妨设训练集有 m 个样本，输入层有 x 个神经元（即一个样本中的变量），隐藏层有 y 个神经元，输出层有 K 个神经元。如果激励函数为$g(\theta^T x) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$，则代价函数

$$\mathrm{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log(h_\theta(x^i)_k - \left(1-y_k^{(i)}\right)\log(1-h_\theta(x^i)_k\right] + \frac{\lambda}{2m}[\sum_{j=1}^{y}\sum_{i=1}^{x}(\Theta_{j,i}^{(1)})^2$$

$$+\sum_{j=1}^{K}\sum_{i=1}^{y}(\Theta_{j,i}^{(2)})^2]$$

神经网络算法：

1.初始化参数Θ

2.对 step=1,2……M

（a）向前传导（forward propagation）计算残差δ

（b）利用δ向后传导(back propagation)计算梯度

（c）梯度下降更新Θ

4. Ensemble Methods

集成学习(Ensemble Methods)将多种在一定程度独立的模型融合在一起，这样往往可以取得比独立模型更好的预测效果，特别是解决单个模型过拟合（overfitting）问题。集成学习由上下两层算法组成下层算法叫做 base learner，指的是被应用在所有将被集成的模型上的一个机器学习算法；上层算法则控制输入到每个 base learner 的数据，这样也能使模型的结果有差异。Base learner 包括决策树、支持向量机等；上层算法包括 bagging、boosting 以及随机森林。例如图 27 所示，通过上层算法将决策树模型集成在一起，得到强分类器。

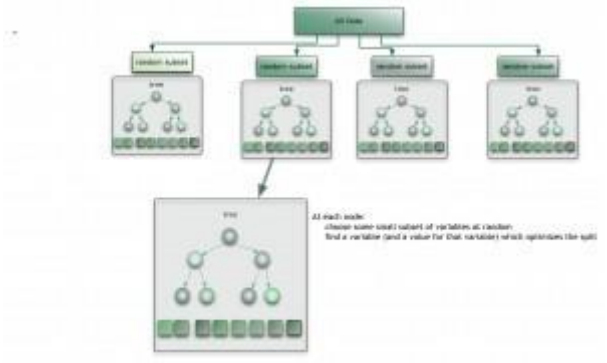


图 27

(1) Gradient Boosting Decision Tree

提升（boosting）方法通过改变训练样本概率分布，针对不同的训练数据分布调用弱学习算法。反复学习，得到一系列弱分类器，最后通过组合这些弱分类器，构成一个强分类器。提升方法实际采用加法模型（即基函数的线性组合）与前向分布算法。以决策树为基函数的提升方法称为提升树（boosting tree）。提升树模型可以表示为决策树的加法模型：

$$f_M(x) = \sum_{m=1}^{M} eps * T(x;\Theta_m)$$

其中，$T(x;\Theta_m)$表示决策树，Θ_m为决策树参数；M为树的个数。

提升树算法：

1. 初始化$f_0(x) = 0,$ learningRate = eps
2. 对 m = 1,2……M

   (a) 计算残差

   $r_{mi} = y_i - f_{m-1}(x_i),\ i = 1,2,\ldots,N$

   (b) 拟合残差r_{mi} 学习一个树，得到$T(x;\Theta_m)$

   (c) 更新$f_m(x) = f_{m-1}(x) + eps * T(x;\Theta_m)$

3. 得到提升树$f_M(x) = \sum_{m=1}^{M} eps * T(x;\Theta_m)$

提升树的损失函数为平方损失函数，因此可以用真实值减去预测值计算残差，梯度提升树将提升树残差的计算方法泛化成损失函数的负梯度，可以拟合一般损失函数的残差，即(a)中的$r_{mi} = -[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$

(2) Random Forest

随机森林在运算量没有显著提高的前提下提高了预测精度。随机森林对多重共线性不敏感，结果对缺失数据和非平衡数据比较稳健，可以很好的预测多达几千个解释变量的作用（Breiman, 2001b）,被誉为最好的算法之一（Iverson et al., 2008）。随机森林通过在训练集的随机子集上进行学习得到一系列弱分类器，一方面，这个随机子集与 bagging 一样，随机抽取训练集的样本，另一方面，与 bagging 不同的是，它还随机抽取特征子集而不是用所有的特征进行训练。

随机森林算法：

1. 初始化$f_0(x) = 0$
2. 对 m = 1,2……M

   (a) 在原样本应用有放回随机抽样抽取 bootstrap 样本 m

   (b)在原特征（N 个）中采用不放回随机抽样抽取 n 个特征

   (c)重构子样本 m,只保留(b)中抽取的特征

   (d)拟合标签y_m学习一个树，得到$T(x;\phi m)$

   (e)更新$f_m(x) = f_{m-1}(x) + T(x;\phi m)$

3.得到随机森林 $f_M(x) = \frac{1}{M}\sum_{m=1}^{M} T(x;\phi m)$

（二）实验结果---样本外

5层随机分层抽样交叉验证集结果

classifier	PRC-AUC 30 features	PRC-AUC 40 features
Logistic	0.3071	0.2429
L2 Logistic	0.4596	0.3429
SVM (Linear)	0.3709	0.2824
SVM (RBF)	0.5281	0.5281
Neural Network	0.5295	0.5386
Gradient Boosting	0.5596	0.5491
Random Forest	0.5386	0.5491

从上面的实验结果可以看到，首先复杂分类器的预测精确度更高，其中在将 Lasso 回归选出的前 30 个特征作为模型输入时，GBDT 分类效果最好；在将输入特征数量扩展 40 个时 GBDT 和 RF 分类效果相当；其次，对于逻辑回归，较少的特征可以帮助模型克服过拟合问题，因此如果实践中应用逻辑回归进行违约概率测算，特征工程是一件很重要的事情，即要在不过拟合的情况下提升样本外预测的精度；最后，其实 GBDT 的 PRC-AUC 值仍较低，原因可能是 1.样本数太小，只有 533 只公司债，应拓展到全债；2.正负样本差距太大，约为 1:17，因此模型还需进一步改进来提高预测的精度，既能识别出违约风险大的债券，又不会损失太多正常状态的债券。

最优模型

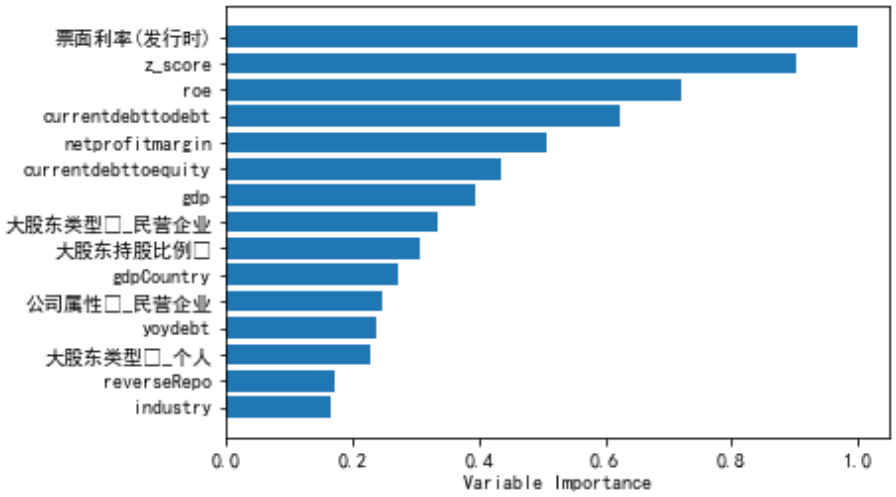


图 28 GBDT feature importance ------top 15 of 30 features

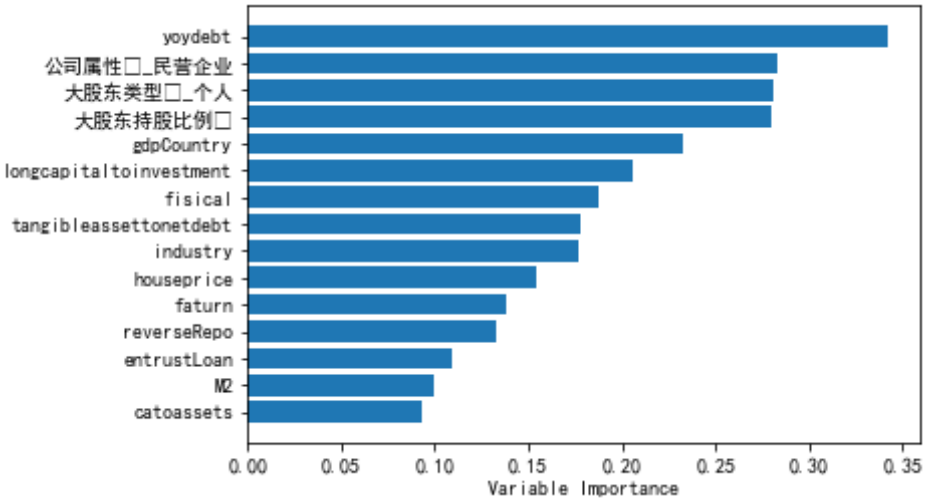


图 29 GBDT feature importance ------top 15 of 40 features

有时因为监管或者模型透明度需求，不得不使用线性模型，此时在传统分析流程中引入机器学习模型可以更有效、更准确的使用线性模型。**因为机器学习模型可以帮助我们思考究竟是什么驱动力导致了数据表现出非线性、时间趋势、模式迁移等。这样我们可以在线性模型中引入非线性项，或使用门限模型等使结果更精确。**

对比前 30 个特征与前 40 个特征学习出的梯度增强决策树可以发现，变量的重要性有了较大的变化，其中在 Lasso 回归中排名靠后的长期资产合适率、有形资产/净债务，委托贷款增速以及 M2 进入了前 40 个特征学习出的模型中，**因此特征之间的相互作用对模型预测效果影响显著。**而总负债（同比增长率）、公司属性_民营企业、大股东类型_个人、大股东持股比例、我国 GDP 增速、行业景气程度增速以及逆回购利率均出现在了两个模型中，因此这几个特征对违约预测尤为关键。

四、展望

本阶段课题通过对 2017 年 7 月 17 日（含）之前市场上所有违约债券进行归因分析，总结出一系列主观上判定对债券是否违约影响较大的因子。然后取 2017 年 7 月 31 日（含）前到期的所有公司债作为样本，在数据收集与清洗后，保留 503 只未违约公司债与 30 只违约公司债作为模型输入。并在前面总结的因子中，选取一系列数据易得的因子作为模型的特征。在进行特征工程后分别取 Lasso 回归选出的前 30 及前 40 个特征，带入不同的分类器中进行训练，并测算样本外得分。最终得出结论，复杂分类器，例如 GBDT 的分类效果要优于传统的逻辑回归；如果使用逻辑回归，需做好特征工程。

不过，模型仍有许多需要继续改进的地方，首先样本数量太小，应该加纳入全债进行分析；其次，正负样本数量差距太大，需要使用一定的实验技巧进行调整完善；可以利用集成学习的思想，尝试将多个分类器进行一定集成来增加预测精度；同样，可以扩展集成学习的思路，将机器学习与传统模型（例如 KMV 模型）结合，例如加权打分，来预测债券违约的可能性。

KMV 模型

选取 2011 年 1 月到 2017 年 8 月间，相应债券发行人股价波动率、所有者权益、无风险债务（短期债务+0.5*长期债务）以及无风险利率（1 年期国债利率）作为 KMV 模型的输入，得到一系列时点上的预期违约概率（EDF）。

图 30

对比违约和未违约企业预期违约概率的变化，大概可以总结出如下经验,当预期违约概率陡然上升**并远超前值**时，是将要违约的预兆。

KMV 模型的缺陷：

在将违约距离映射到预期违约概率的过程中使用的是正态分布，与实际违约概率分布有偏差。

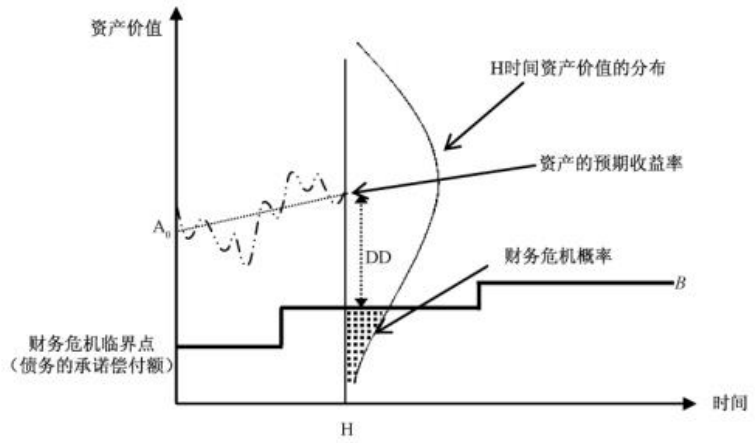


图 1　资产价值和财务危机概率

图 31

代码附录

代码一：违约样本描述性统计

```
'''违约样本描述性统计'''
default = pd.read_excel('F:\\GFsummer2017\\债券违约报表依据发行人删
减.xlsx',sheet_name = 'Sheet1')
List2014 = []
List2015 = []
List2016 = []
List2017 = []
for i,time in enumerate(default['发生日期']):
    intTime = int(str(time)[:4])
    if intTime < 2015:
        List2014.append(default.iloc[i,:])
    elif 2015<=intTime<2016:
        List2015.append(default.iloc[i,:])
    elif 2016<=intTime<2017:
        List2016.append(default.iloc[i,:])
    else:
        List2017.append(default.iloc[i,:])
frame2014 = DataFrame(List2014)
frame2015 = DataFrame(List2015)
frame2016 = DataFrame(List2016)
frame2017 = DataFrame(List2017)

plot.figure(figsize=[10,10])
plot.pie(default['所属 wind 行业'].value_counts(),labels=default['所属 wind 行业
'].value_counts().keys())
plot.barh(range(15),default['所属 wind 行业'].value_counts()[:15])
plot.yticks(range(15),np.array(default['所属 wind 行业
'].value_counts().index)[:15])

plot.figure(figsize=[10,10])
plot.pie(default['省份'].value_counts(),labels=default['省份
'].value_counts().keys())
plot.barh(range(15),default['省份'].value_counts()[:15])
plot.yticks(range(15),np.array(default['省份'].value_counts().index[:15]))
```

代码二：数据收集与处理

数据处理包

```
# -*- coding: utf-8 -*-
"""
Created on Wed Aug  2 09:03:13 2017
timeSeriesMatch 按照样本的时间 在数据集中匹配相应的数据
panelMatch 以按照样本某个特征 在数据集中匹配相应时间和特征的数据
readWindExcel
frequencyDeal
getFeaturesWind
@author: hudie
"""
import pandas as pd
from pandas import DataFrame,Series
import sys
import numpy as np
import re
from datetime import *
from imp import reload

from WindPy import *
from datetime import *
w.start()

#frequency = [3, 1, 3, 12],timeList 是月度:1，季度:3，半年:6，还是年:12

#将时间序列数据按照 allcororatebonds 中债券到期的时间选出来，并按照其排列顺序排列
#timeList, dataList,nameList 必须一一对应，长度一样
def timeSeriesMatch(timeList, dataList, nameList, maturityDate):
    addDict = {}
    for i, timeset in enumerate(timeList):
        frequency = timeset[1]
        time = timeset[0]
        data = dataList[i]
        name = nameList[i]
        addDict[name] = []
        if data.shape[0] == len(time):
            for j, maturity in enumerate(maturityDate):
                tempdate = [] #*****记得设为空 不然得到的 CPI 都一样 因为 list 接着
append
                maturity = str(maturity)[:10]
                if str(min(time))[:10] > maturity:
```

```
                tempdata = data.iloc[time.index(min(time))].values[0]
                addDict[name].append(tempdata)
            else:
                for k, date in enumerate(time):
                    date = str(date)[:10]
                    if maturity > date:
                        tempdate.append(date)
                tempdate.sort() #date 一定要是一个上升的趋势 才能对上 time 和 data
的 index
                if frequency == 1:
                    tempdata = data.iloc[max(tempdate.index(max(tempdate)) -
6,0)].values[0]
                elif frequency == 3:
                    tempdata = data.iloc[max(tempdate.index(max(tempdate)) -
2,0)].values[0]
                elif frequency == 6:
                    tempdata = data.iloc[max(tempdate.index(max(tempdate)) -
1,0)].values[0]
                else:
                    tempdata =
data.iloc[max(tempdate.index(max(tempdate)),0)].values[0]
                addDict[name].append(tempdata)
        else:
            print('错误：数据时间与时间列表长度不符合！')
    return addDict

#读取面板数据 timepanellist datapanellist namepanellist searchbench 必须一一对应
长度一致 matchnumber 是指从数据集字段名称的哪里开始匹配
def panelMatch(timePanelList, dataPanelList, namePanelList, searchBench,
maturityDate, matchNumberMinList, matchNumberMaxList):
    addDictPanel = {}
    for i in range(len(searchBench)):
        provinces = searchBench[i]
        fisicalPd = dataPanelList[i]
        fisicalTime = timePanelList[i][0]
        frequency = timePanelList[i][1]
        namePanel = namePanelList[i]
        matchNumberMin = matchNumberMinList[i]
        matchNumberMax = matchNumberMaxList[i]
        addDictPanel[namePanel] = []
        for j, province in enumerate(provinces):
            tempdate = []
            maturityPanel = str(maturityDate[j])[:10]
            for key in fisicalPd.keys():
```

```
                matchWord = key[matchNumberMin:matchNumberMax]
                pattern = re.compile(matchWord)
                match = pattern.match(str(province))
                if match:
                    dataPanel = fisicalPd[key]
                    if str(min(fisicalTime))[:10] > maturityPanel:
                        tempdata =
dataPanel.iloc[fisicalTime.index(min(fisicalTime))]
                        addDictPanel[namePanel].append(tempdata)
                    else:
                        for fisicaldate in fisicalTime:
                            if maturityPanel > str(fisicaldate)[:10]:
                                tempdate.append(fisicaldate)
                        tempdate.sort()
                        if frequency == 1:
                            tempdata =
dataPanel.iloc[max(tempdate.index(max(tempdate))-6,0)]
                        elif frequency == 3:
                            tempdata =
dataPanel.iloc[max(tempdate.index(max(tempdate))-2,0)]
                        elif frequency == 6:
                            tempdata =
dataPanel.iloc[max(tempdate.index(max(tempdate))-1,0)]
                        else:
                            tempdata =
dataPanel.iloc[max(tempdate.index(max(tempdate)),0)]
                        addDictPanel[namePanel].append(tempdata)
                #前提是第一次的时候 match 对了，如果与 pattern 的开头一样的话
                if len(tempdate) != 0:
                    break
            if len(tempdate) == 0:
                addDictPanel[namePanel].append(None)
    return addDictPanel

#从 WIND 上提取数据的格式较为统一
def readWindExcel(path):
    data = pd.read_excel(path, header=0, sheet_name = 'sheet1')
    data = data[1:-2]
    dataTime = list(data['指标名称'])
    del data['指标名称']
    columns = data.keys()
    datanp = np.array(data)
    data = DataFrame(datanp, columns = columns, index = dataTime)
    return data, dataTime
```

```
def frequencyDeal(price,period,days):
    priceData = []
    priceIndex = []
    for i in range(period):
        priceData.append((price.iloc[(i+1)*days] -
price.iloc[i*days])/price.iloc[i*days])
        priceIndex.append(price.index[(i+1)*days])
    priceData = DataFrame(priceData, index = priceIndex)
    priceTime = [i.date() for i in priceIndex]
    return priceData,priceTime

def getFeaturesWind(features, forwardYear, maturityDate, allCorporateBonds):
    featureDict = {}
    for feature in features:
        index = 0
        for md in maturityDate:
            tempMonth = 0
            mdstr = str(md)[:10]
            mddate = md.date()
            month = mddate.month
            tempYear = mddate.year
            if month <= 6:
                tempMonth = 6 + month
                tempYear = tempYear - (forwardYear + 1)
            else:
                tempMonth = month - 6
                tempYear = tempYear - forwardYear
            beginDateStr = str(tempYear) + '-' + str(tempMonth) + '-'+'01'
            debtCode = allCorporateBonds['证券代码'][index]
            data = w.wsd(debtCode, feature, beginDateStr, mdstr , "Period=Q")
            featureSeries = Series(data.Data[0])
            featureSeriesNotNull = featureSeries.dropna()
            if len(featureSeriesNotNull) == 0:
                featureNum = None
            else:
                featureNum = featureSeries[max(featureSeriesNotNull.index)]
            if not (feature in featureDict):
                featureDict[feature] = []
            featureDict[feature].append(featureNum)
            index += 1
            print(index)
    return DataFrame(featureDict)
```

```
def financialfeatures(forwardYear,maturityDate,allCorporateBonds):
    features =
["netprofitmargin","optoebt","deductedprofittoprofit","operateincometoebt",\
          "roa_yearly","ocftoinveststockdividend","ocftoop","ocftoassets",\

"ocftodividend","debttoassets","deducteddebttoassets","longcapitaltoinvestme
nt",\

"assetstoequity","catoassets","currentdebttoequity","intdebttototalcap",\
          "equitytototalcapital","currentdebttodebt","current","quick",\
          "cashratio","cashtocurrentdebt","ocftointerest","ocftoquickdebt",\

"tangibleassettodebt","tangibleassettonetdebt","debttotangibleequity",\

"ebitdatodebt","ocftoshortdebt","ocftonetdebt","ocficftocurrentdebt",\

"ocficftodebt","ebittointerest","longdebttoworkingcapital","longdebttodebt",
\

"netdebttoev","interestdebttoev","ebitdatointerestdebt","ebitdatointerest",\

"tltoebitda","cashtostdebt","invturn","arturn","caturn","operatecaptialturn"
,\

"faturn","non_currentassetsturn","apturn","yoyprofit","yoynetprofit_deducted
",\

"yoyocf","maintenance","yoy_cash","yoy_fixedassets","yoycf","yoydebt","roe",
"z_score"]
    searchDict = {"netprofitmargin":'销售净利率',"optoebt":'主营业务比率
',"deductedprofittoprofit":'扣除非经常损益后的净利润/净利润
',"operateincometoebt":'经营活动净收益/利润总额',\
          "roa_yearly":'年化 ROA',"ocftoinveststockdividend":'现金满足投资比率
',"ocftoop":'现金营运指数',"ocftoassets":'全部资金回收率',\
          "ocftodividend":'现金股利保障倍数',"debttoassets":'资产负债率
',"deducteddebttoassets":'剔除预收款项后的资产负债率',"longcapitaltoinvestment":'
长期资产合适率',\
          "assetstoequity":'权益乘数',"catoassets":'流动资产/总资产
',"currentdebttoequity":'流动负债权益比',"intdebttototalcap":'带息债务/全部投入资
本',\
          "equitytototalcapital":'归属母公司股东的权益/全部投资资本
',"currentdebttodebt":'流动负债/负债合计',"current":'流动比率',"quick":'速动比率
',\
```

```
            "cashratio":'保守速动比率',"cashtocurrentdebt":'现金比率
','ocftointerest":'现金流量利息保障倍数',"ocftoquickdebt":'现金到期债务比',\
            "tangibleassettodebt":'有形资产/负债合计',"tangibleassettonetdebt":'
有形资产/净债务',"debttotangibleequity":'有形净值债务率',\
            "ebitdatodebt":'息税折旧摊销前利润/负债合计',"ocftoshortdebt":'经营活动
产生的现金流量净额/流动负债',"ocftonetdebt":'经营活动产生的现金流量净额/净债务
',"ocficftocurrentdebt":'非筹资性现金净流动/流动负债',\
            "ocficftodebt":'非筹资性现金净流动/负债总额',"ebittointerest":'已获利息
倍数（EBIT/利息费用）',"longdebttoworkingcapital":'长期债务/营运资金
',"longdebttodebt":'长期负债占比',\
            "netdebttoev":'净债务/股权价值',"interestdebttoev":'带息债务/股权价值
',"ebitdatointerestdebt":'EBITDA/带息债务',"ebitdatointerest":'EBITDA/利息费用
',\
            "tltoebitda":'全部债务/EBITDA',"cashtostdebt":'货币资金/短期债务
',"invturn":'存货周转率',"arturn":'应收账款周转率',"caturn":'流动资产周转率
',"operatecaptialturn":'营运资产周转率',\
            "faturn":'固定资产周转率',"non_currentassetsturn":'非流动资产周转率
',"apturn":'应付账款周转率',"yoyprofit":'净利润（同比增长率）
',"yoynetprofit_deducted":'归属母公司股东的净利润-扣除非经营损失（同比增长率）',\
            "yoyocf":'经营活动产生的现金流净额（同比增长率)',"maintenance":'资本项目规
模维持率',"yoy_cash":'货币资金增长率',"yoy_fixedassets":'固定资产投资扩张率
',"yoycf":'现金净流量（同比增长率）',"yoydebt":'总负债（同比增长率）',"roe":'净资产收
益率',"z_score":'z 值'}
#    forwardYear = 1
#    maturityDate = list(allCorporateBonds['到期日期'])
    res =  getFeaturesWind(features, forwardYear,maturityDate,
allCorporateBonds)
    return res
```

数据处理流程

```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug  7 09:57:33 2017

@author: hudie
"""

import pandas as pd
from pandas import DataFrame,Series
import sys
sys.path.append('F:\\GFsummer2017')
import numpy as np
import re
import processingData
from imp import reload
reload(processingData)

from WindPy import *
from datetime import *
w.start()

'''待处理样本数据集 '''
allCorporateBonds = pd.read_excel('F:\\GFsummer2017\\到期公司
债.xlsx',sheet_name = 'Sheet1')
allCorporateBonds = allCorporateBonds[:-2]
allCorporateBonds.rename(columns = {'到期日期↓' : '到期日期'}, inplace=True)
'''违约样本数据集'''
defaultFrame = pd.read_excel('F:\\GFsummer2017\\违约债券报表.xlsx',sheet_name
= 'Sheet1')
defaultName = defaultFrame['名称'].dropna()
debtRemainIndex = []
matchWord = [r'[^S]CP',r'MTN',r'PPN',r'SCP']
for index in range(len(defaultName)):
    text = defaultName[index]
    count = 0
    for subPattern in matchWord:
        pattern = re.compile(subPattern)
        match = pattern.search(text)
        if match:
            continue
        else:
            count += 1
```

```
            if count == len(matchWord):
                debtRemainIndex.append(index)
defaultDebtRemain = defaultFrame.iloc[debtRemainIndex]
defaultdebtCorporateBonds = defaultDebtRemain['Wind 债券二级分类'] == '一般企业债
'
defaultdebtABS = defaultDebtRemain['Wind 债券二级分类'] == '证监会主管 ABS'
defaultdebtNAN = defaultDebtRemain['Wind 债券二级分类'].isnull()
defaultdebtCBABSNAN = (defaultdebtCorporateBonds | defaultdebtABS) |
defaultdebtNAN
defaultDebt = defaultDebtRemain[~defaultdebtCBABSNAN]
'''将待处理样本数据集中的到期日期变为违约样本数据集中的违约日期，并对违约样本编码 1 对未违约
样本编码 0'''
label = {}
label['label'] = []
allCorDefault = []
for i,maturitycode in enumerate(allCorporateBonds['证券代码']):
    count = 0
    for j,defaultcode in enumerate(defaultDebt['代码']):
        if defaultcode == maturitycode:
            allCorporateBonds['到期日期'].iloc[i] =  defaultDebt['发生日期
'].iloc[j]
            label['label'].append(1)
            allCorDefault.append(allCorporateBonds.iloc[i])
            count += 1
            break #对于某只债券二次违约的只匹配第一违约
    if count == 0:
         label['label'].append(0)
allCorDefault = DataFrame(allCorDefault)
allCorporateBonds = pd.concat([allCorporateBonds,DataFrame(label)], axis=1)
'''计算已发行债券总数'''
corSum = allCorporateBonds['公司发行证券一览']
corSumNum = []
for bonds in corSum:
    bondsList = re.split(';',bonds)
    corSumNum.append(len(bondsList))
corSumNum = DataFrame(corSumNum, columns = ['已发债总数'])
allCorporateBonds = pd.concat([allCorporateBonds,corSumNum], axis=1)
'''获取财务数据,注释掉的语句即可，但是数据量太大，获取时间很长，这里用已经获取储存的文件读
入'''
financialfeatures =
pd.read_excel('F:\\GFsummer2017\\financialfeatures1.xlsx')
allCorporateBonds = pd.concat([allCorporateBonds,financialfeatures], axis=1)
financialfeatures.dropna(how = 'all',inplace = True)
allCorporateBonds = allCorporateBonds.iloc[list(financialfeatures.index)]
```

```
allCorporateBonds = pd.concat([allCorporateBonds,allCorDefault], axis=0)
allCorporateBonds.drop_duplicates(subset=['证券代码'], keep='first',
inplace=True)
allCorporateBonds['label'].fillna(1, inplace = True)

'''处理缺失值'''
nrows, ncols = allCorporateBonds.shape
totalMissingData =
allCorporateBonds.isnull().sum().sort_values(ascending=False)
percentMissingData =
(allCorporateBonds.isnull().sum()/allCorporateBonds.isnull().count()).sort_v
alues(ascending=False)
missingData = pd.concat([totalMissingData,percentMissingData], axis=1,
keys=['Total', 'Percent'])
sys.stdout.write('head information \n')
print(missingData.head(int(0.5*ncols)))
delColumns = percentMissingData > 0.15
for columns in list(delColumns.keys()):
    try:
        if delColumns[columns]:
            del allCorporateBonds[columns]
    except KeyError:
        continue

'''
加入对应时间的宏观变量
'''
'''get timeseries data'''
#1
gdpCountry = w.edb('M0000541',"2010-08-03", "2017-08-03","Fill=Previous")
gdpCountryData = np.array(gdpCountry.Data)
gdpCountryTime = list(gdpCountry.Times)
gdpCountryData = DataFrame(gdpCountryData.T, columns =
list(gdpCountry.Codes), index = gdpCountryTime)

#2
CPI = w.edb('M0000729',"2010-08-03", "2017-08-03","Fill=Previous")
CPIData = np.array(CPI.Data)
CPITime = list(CPI.Times)
CPIData = DataFrame(CPIData.T, columns = list(CPI.Codes), index = CPITime)

#3
housePrice,housePriceTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\房价.xls')
```

```
#housePrice 为月度数据，换算成半年度涨跌幅数据 共有 80 个月，13.3 个半个年
housePriceData,housePriceTime =
processingData.frequencyDeal(housePrice,13,6)

#4 社会融资规模(TSF)
tsf, tsfTime = processingData.readWindExcel(u'F:\GFsummer2017\data\社会融资规
模增速.xls')

#5 委托贷款增速
entrustLoan, entrustLoanTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\委托贷款.xls')

#6 信托贷款增速
trustLoan, trustLoanTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\信托贷款.xls')

#7 未贴现银行承兑汇票增速
undiscounted, undiscountedTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\未贴现银行承兑汇票.xls')


#8 M2
M2,M2Time = processingData.readWindExcel(u'F:\GFsummer2017\data\货币供应
量.xls')
M2Data,M2Time = processingData.frequencyDeal(M2,11,6)

#9 oil price
oilPrice, oilPriceTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\油价数据.xls')
oilPrice = DataFrame(oilPrice['OPEC:一揽子原油价格'])
#oilprice 为日度数据，换算成半年度涨跌幅数据 共有 1683 天 13.4 个半个年 因此每个区间大概为
126 天
oilPriceData,oilPriceTime = processingData.frequencyDeal(oilPrice,13,126)

#10 11 12 nterest rate corridor (IRC)利率走廊数据
IRC,IRCTime = processingData.readWindExcel(u'F:\GFsummer2017\data\利率走廊数
据.xls')
reverseRepo = DataFrame(IRC['逆回购利率:7 天'])
excessReserve = DataFrame(IRC['超额存款准备金率(超储率):金融机构'])
SLF =  DataFrame(IRC['常备借贷便利(SLF)利率:7 天'])

#上证指数
stock,stockTime = processingData.readWindExcel(u'F:\GFsummer2017\data\上证指
数.xlsx')
```

```
stock = DataFrame(stock['收盘价(元)'])
stock = np.log(stock/stock.shift(1))
stock = stock[1:]
stockTime = stockTime[1:]

'''get panel data'''
#1
gdp =
w.edb("M0048668,M0049108,M0048912,M0049080,M0049047,M0049034,M0049006,M00489
72,M0048698,M0049014,\

M0049148,M0048736,M0048752,M0049024,M0049071,M0048988,M0048997,M0048824,M004
8862,M0048903,\

M0048720,M0049098,M0048884,M0049130,M0049113,M0049088,M0048773,M0049062,M004
9057,M0049119,\
     M0048949", "2010-08-03", "2017-08-03","Fill=Previous")
gdpData = np.array(gdp.Data)
gdpTime = list(gdp.Times)
gdpData = DataFrame(gdpData.T, columns = list(gdp.Codes), index = gdpTime)
gdpData.rename(columns = {"M0048668":'北京','M0049108':'天津','M0048912':'河北
','M0049080':'山西',\
                      'M0049047':'内蒙古','M0049034':'辽宁','M0049006':'吉林
','M0048972':'黑龙江',\
                      'M0048698':'上海','M0049014':'江苏','M0049148':'浙江
','M0048736':'安徽',\
                      'M0048752':'福建','M0049024':'江西','M0049071':'山东
','M0048988':'湖北',\
                      'M0048997':'湖南','M0048824':'广东','M0048862':'广西
','M0048903':'海南',\
     'M0048720':'重庆','M0049098':'四川','M0048884':'贵州','M0049130':'云南
','M0049113':'西藏',\
     'M0049088':'陕西','M0048773':'甘肃','M0049062':'青海','M0049057':'宁夏
','M0049119':'新疆',\
     'M0048949':'河南'}, inplace = True)


#2
industry, industryTime =
processingData.readWindExcel(u'F:\GFsummer2017\data\wind四级行业指数.xls')
#industry为日度数据，换算成半年度涨跌幅数据 共有4260天 35.4个半个年 因此每个区间大概为
121天
industryData,industryTime = processingData.frequencyDeal(industry,35,121)
```

```
#3 地方财政收入 - 支出
fisical = w.edb("M0025044,M0025868,M0025070,M0025900,M0025096,M0025932,\

M0025122,M0025964,M0025148,M0025996,M0025174,M0026028,M0025200,M0026060,M002
5226,M0026092,\

M0025252,M0026124,M0025278,M0026156,M0025304,M0026188,M0025330,M0026220,M002
5356,M0026252,\

M0025382,M0026284,M0025408,M0026316,M0025434,M0026348,M0025460,M0026380,M002
5486,M0026412,\

M0025512,M0026444,M0025538,M0026476,M0025564,M0026508,M0025590,M0026540,M002
5616,M0026572,\

M0025642,M0026604,M0025668,M0026636,M0025694,M0026668,M0025720,M0026700,M002
5746,M0026732,\
     M0025772,M0026764,M0025798,M0026796,M0025824,M0026828","2010-08-03",
"2017-08-03","Fill=Previous")
fisicalTime = list(fisical.Times)
fisicalData  = np.array(fisical.Data).T
nrowsFisicalData, ncolsFisicalData = fisicalData.shape
fisicalNet = {}
fisicalCode = list(fisical.Codes)
for i in range(0,ncolsFisicalData,2):
    tempCode = fisicalCode[i]
    fisicalNet[tempCode] = []
    tempData = list(fisicalData[:,i] - fisicalData[:,(i+1)])
    fisicalNet[tempCode] = tempData
fisicalPd = DataFrame(fisicalNet, index = fisicalTime)
fisicalPd.rename(columns = {"M0025044":'北京','M0025070':'天津','M0025096':'河
北',\
     'M0025122':'山西','M0025148':'内蒙古','M0025174':'辽宁','M0025200':'吉林
','M0025226':'黑龙江',\
     'M0025252':'上海','M0025278':'江苏','M0025304':'浙江','M0025330':'安徽
','M0025356':'福建',\
     'M0025382':'江西','M0025408':'山东','M0025434':'河南','M0025460':'湖北
','M0025486':'湖南',\
     'M0025512':'广东','M0025538':'广西','M0025564':'海南','M0025590':'重庆
','M0025616':'四川',\
     'M0025642':'贵州','M0025668':'云南','M0025694':'西藏','M0025720':'陕西
','M0025746':'甘肃',\
     'M0025772':'青海','M0025798':'宁夏','M0025824':'新疆'}, inplace = True)
```

```
'''seachBench'''
maturityDate = list(allCorporateBonds['到期日期'])
provinces = allCorporateBonds['省份']
industries = allCorporateBonds[list(allCorporateBonds.keys())[48]]
#frequency = [3, 1, 3, 12],timeList 是月度:1，季度:3，半年:6，还是年:12
timeList =
[(CPITime,1),(gdpCountryTime,3),(housePriceTime,6),(tsfTime,12),(M2Time,6),(
oilPriceTime,6),\

(IRCTime,6),(IRCTime,6),(IRCTime,6),(stockTime,6),(entrustLoanTime,12),(trus
tLoanTime,12),(undiscountedTime,12)]
dataList =
[CPIData,gdpCountryData,housePriceData,tsf,M2Data,oilPriceData,reverseRepo,e
xcessReserve,SLF,stock,entrustLoan,trustLoan,undiscounted]
nameList =
['CPI','gdpCountry','houseprice','tsf','M2','oilPrice','reverseRepo','excess
Reserve','SLF','stock','entrustLoan','trustLoan','undiscounted']

timePanelList = [(fisicalTime,12),(gdpTime,3),(industryTime,6)]
dataPanelList = [fisicalPd,gdpData,industryData]
namePanelList = ['fisical','gdp','industry']
searchBench = [provinces,provinces,industries]
matchNumberMinList = [0,0,9]
matchNumberMaxList = [2,2,None] #正则化选择 pattern.match

addDict = processingData.timeSeriesMatch(timeList, dataList, nameList,
maturityDate)
addDictPanel = processingData.panelMatch(timePanelList, dataPanelList,
namePanelList, searchBench,
maturityDate,matchNumberMinList,matchNumberMaxList)

allCorporateBondsnp = np.array(allCorporateBonds)
nrow,ncol = allCorporateBondsnp.shape
columns = list(allCorporateBonds.keys())
index = list(range(nrow))
allCorporateBonds = DataFrame(allCorporateBondsnp, columns = columns, index
= index)
allCorporateBonds =
pd.concat([allCorporateBonds,DataFrame(addDict),DataFrame(addDictPanel)],
axis=1)
allCorporateBonds.to_excel('sample.xlsx',sheet_name = 'Sheet1')
```

代码三：数据集描述性统计

```
rawData = pd.read_excel('F:\\GFsummer2017\\sample.xlsx', sheet_name =
'sheet1')
data = rawData.iloc[:,2:]
y = data['label']
del data['label']
X = data
data = pd.concat([X,y],axis=1)
nrows, ncols = X.shape
X.get_dtype_counts()
findCategory = X.dtypes == object
categoricalFeatures = X.loc[:,findCategory]
numericFeatures = X.loc[:,~findCategory]
#numericFeatures = StandardScaler().fit_transform(numericFeatures)
mean_vals = numericFeatures.mean()
sd_vals = numericFeatures.std()
numericFeatures = (numericFeatures - mean_vals)/sd_vals
categoryCount = {}
for feature in categoricalFeatures:
    categoryCount[feature] = []
    categoryCount[feature] = categoricalFeatures[feature].value_counts()
#
#numeric qq plot
for feature in numericFeatures:
    stats.probplot(numericFeatures[feature],dist='norm',plot=plot)
    feature = u"%s" %(feature) #加上 u unicode 进行编码 为了读取中文
    plot.title(feature)
    plot.show()

#parallel coordinate plot
plot.figure(figsize=[20,10])
for i in range(nrows):
    if y[i] == 0:
        pcolor = 'blue'
    else:
        pcolor= 'red'
    dataRow = numericFeatures.iloc[i,:]
    dataRow.plot(color=pcolor)
plot.xticks(np.arange(numericFeatures.shape[1]),range(numericFeatures.shape[
1]))
plot.xlabel('Attribute Index')
plot.ylabel('Attribute Values')
plot.show()
```

```
attributeList = [2,5,8,9,13,15,17,29,33,35,38,40,41,54,55]
for i in attributeList:
    attribute = numericFeatures.iloc[:,i]
    plot.scatter(attribute,y,alpha=0.5,s=120)
    plot.xlabel(numericFeatures.keys()[i])
    plot.ylabel('label')
    plot.show()

corMat = DataFrame(data.corr())
plot.pcolor(corMat)
plot.show()
labelAbs = DataFrame(abs(corMat['label']))
labelAbsTemp = labelAbs.sort_values(by=['label'],ascending=False)
print(labelAbsTemp)
corMat['label'][labelAbsTemp[:20].index]
highcorr = corMat[np.abs(corMat)>0.75]
highcorr.sort_values(by=list(highcorr.keys()),ascending=False,inplace=True)

#将省份和行业去掉 原因 1.算法对类别数目的要求 2 不重要：相关性等
del X['省份']
del X['所属 Wind 行业名称\r']
highcorrDel =
['quick','current','ocftoshortdebt','ocficftocurrentdebt','longdebttodebt',\

'intdebttototalcap','tangibleassettodebt','equitytototalcapital',\
            'non_currentassetsturn']
for delvar in highcorrDel:
    del X[delvar]
```

代码四：Lasso Regression

```
def enetModel(Xfillna,yNor,l1_ratio,names):
    alphas, coefs, _ = linear_model.enet_path(Xfillna,yNor,l1_ratio = 1.0, \
                                        fit_intercept=False,
return_models=False)
    plot.plot(alphas,coefs.T)
    plot.xlabel('alpha')
    plot.ylabel('Coefficients')
    plot.title('Coefficient curves for debt classify data')
    plot.axis('tight')
    plot.semilogx()
    ax = plot.gca()
    ax.invert_xaxis()
    plot.show()

    nattr, nalpha = coefs.shape
    #find coefficient ordering
    nzList = []
    for iAlpha in range(1,nalpha):
        coefList = list(coefs[:,iAlpha])
        nzCoef = [index for index in range(nattr) if coefList[index] != 0.0]
        for q in nzCoef:
              if not(q in nzList):
                  nzList.append(q)
    nameList = [names[nzList[i]] for i in range(len(nzList))]
    print('Attribute Ordered by How Early They Enter the Model')
    print(nameList)
    return nameList
```

代码五：分类器比较---改编自 sklearn 的例子

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
Created on Thu Aug 29 09:06:58 2017
@author:hudie change from sklearn example
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier,GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression

h = .02  # step size in the mesh

names = ['Logistic','Penalized Logistic', "Linear SVM", "RBF SVM",
         "Random Forest", "Neural Net"]

classifiers = [
    LogisticRegression(C=1000000,penalty='l2',tol=0.001),
    LogisticRegression(C=0.025,penalty='l2',tol=0.001),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    #RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    GradientBoostingClassifier(max_depth=5,learning_rate=0.003),
    MLPClassifier(alpha=1)]

X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
                           random_state=1, n_clusters_per_class=1)
```

```
rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_separable = (X, y)

datasets = [make_moons(noise=0.3, random_state=0),
            make_circles(noise=0.2, factor=0.5, random_state=1),
            linearly_separable
            ]

figure = plt.figure(figsize=(27, 9))
i = 1
# iterate over datasets
for ds_cnt, ds in enumerate(datasets):
    # preprocess dataset, split into training and test part
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=.4, random_state=42)

    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))

    # just plot the dataset first
    cm = plt.cm.RdBu
    cm_bright = ListedColormap(['#FF0000', '#0000FF'])
    ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
    if ds_cnt == 0:
        ax.set_title("Input data")
    # Plot the training points
    ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,
               edgecolors='k')
    # and testing points
    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,
alpha=0.6,
               edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    i += 1

    # iterate over classifiers
```

```
    for name, clf in zip(names, classifiers):
        ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
        clf.fit(X_train, y_train)
        score = clf.score(X_test, y_test)

        # Plot the decision boundary. For that, we will assign a color to each
        # point in the mesh [x_min, x_max]x[y_min, y_max].
        if hasattr(clf, "decision_function"):
            Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
        else:
            Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

        # Put the result into a color plot
        Z = Z.reshape(xx.shape)
        ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)

        # Plot also the training points
        ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,
                   edgecolors='k')
        # and testing points
        ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright,
                   edgecolors='k', alpha=0.6)

        ax.set_xlim(xx.min(), xx.max())
        ax.set_ylim(yy.min(), yy.max())
        ax.set_xticks(())
        ax.set_yticks(())
        if ds_cnt == 0:
            ax.set_title(name)
        ax.text(xx.max() - .3, yy.min() + .3, ('%.2f' % score).lstrip('0'),
                size=15, horizontalalignment='right')
        i += 1

plt.tight_layout()
plt.show()
```

代码六：实验

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 17 09:06:58 2017

@author: hudie
"""

import sys
sys.path.append('F:\\GFsummer2017')
import pandas as pd
from pandas import DataFrame,Series
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets, linear_model, svm, ensemble, neural_network
from sklearn.metrics import precision_recall_curve, auc,f1_score
from sklearn.model_selection import StratifiedShuffleSplit
import matplotlib.pyplot as plot
import classifierStat
from imp import reload
reload(classifierStat)
plot.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plot.rcParams['axes.unicode_minus']=False #用来正常显示负号

rawData = pd.read_excel('F:\\GFsummer2017\\sample.xlsx', sheet_name =
'sheet1')
data = rawData.iloc[:,2:]
y = data['label']
del data['label']
X = data
data = pd.concat([X,y],axis=1)
ySVM = []
for yi in y:
    if yi == 1:
        ySVM.append(yi)
    else:
        ySVM.append(-1)
ySVM = np.array(ySVM)
#将省份和行业去掉 原因 1.算法对类别数目的要求 2 不重要：相关性等
del X['省份']
del X['所属 Wind 行业名称\r']
highcorrDel =
['quick','current','ocftoshortdebt','ocficftocurrentdebt','longdebttodebt',\
```

```
'intdebttototalcap','tangibleassettodebt','equitytototalcapital',\
            'non_currentassetsturn']
for delvar in highcorrDel:
    del X[delvar]
X = pd.get_dummies(X)
Xfillna = X.fillna(X.mean())
XfillnaMean = Xfillna.mean()
XfillnaStd = Xfillna.std()
yMean = y.mean()
yStd = y.std()
Xfillna = (Xfillna - XfillnaMean)/XfillnaStd
yNor = (y - yMean)/yStd
names = list(Xfillna.keys())
Xfillna = np.array(Xfillna)
yNor = np.array(yNor)

#lasso feeture engineering
nameList = classifierStat.enetModel(Xfillna,yNor,0.9,names)
nameListChoose = nameList[:30]

for iName in names:
    if not(iName in nameListChoose):
        del X[iName]
Xfillna = X.fillna(X.mean())
XfillnaMean = Xfillna.mean()
XfillnaStd = Xfillna.std()
Xfillna = (Xfillna - XfillnaMean)/XfillnaStd
namesAfter = list(Xfillna.keys())
Xfillna = np.array(Xfillna)
y = np.array(y)
X = np.array(X)
#k fold cv and 分层抽样
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.3,random_state=0)
#数据、训练、选参数、样本外 PRC

penalChooseList = []
Clist = []
f1ChooseList = []
for i,C in enumerate((1000,100,10,1,0.1)):
#for C in [0.003*(3**i) for i in range(20)]:
    print(C)
    prcList = []
    f1Score = []
```

```
    Clist.append(C)
    for train, test in cv.split(Xfillna,ySVM):
        #l2 penalized logistic regression C 要加 1000000 表示没 penalized
        #classifier =
linear_model.LogisticRegression(C=C,penalty='l2',tol=0.001)
        #svm
        classifier = svm.SVC(C=C,kernel='linear',tol=0.00001)
        #nn
        #classifier =
neural_network.MLPClassifier(hidden_layer_sizes=(1,),activation='logistic',a
lpha=C)
        classifier.fit(Xfillna[train],ySVM[train])
        predictions = classifier.predict(Xfillna[test])
        precision, recall, thresholds =
precision_recall_curve(ySVM[test],predictions)
        prc_auc = auc(recall, precision)
        prcList.append(prc_auc)

    penalChooseList.append(np.mean(prcList))
withoutPenalized = penalChooseList[0]
maxPRC =  max(penalChooseList)
CBest = Clist[penalChooseList.index(maxPRC)]
print(maxPRC,CBest,withoutPenalized)

#gradient boosting
def findBestGBM(Xfillna,y):
    cv = StratifiedShuffleSplit(n_splits=5, test_size=0.3,random_state=0)
    saveDict = {}
    saveDict['nEst'] = []
    saveDict['depth'] = []
    saveDict['learnRate'] = []
    saveDict['meanprc'] = []
    for nEst in list(range(100,2001,100)):
        for depth in list(range(1,6)):
            for learnRate in [0.003*(3**i) for i in range(9)]:
                maxFeatures = int(np.sqrt(Xfillna.shape[1]))

                prcList = []
                for train, test in cv.split(Xfillna,y):
                    classifier = ensemble.GradientBoostingClassifier(
                            n_estimators=nEst, max_depth=depth,
                            learning_rate=learnRate,max_features=maxFeatures)
                    classifier.fit(Xfillna[train],y[train])
                    predictions = classifier.predict(Xfillna[test])
```

```
                precision, recall, thresholds =
precision_recall_curve(y[test],predictions)
                prc_auc = auc(recall, precision)
                prcList.append(prc_auc)
            saveDict['meanprc'].append(np.mean(prcList))
            saveDict['nEst'].append(nEst)
            saveDict['depth'].append(depth)
            saveDict['learnRate'].append(learnRate)
            print(np.mean(prcList),nEst,depth,learnRate)
    maxPRC =  max(saveDict['meanprc'])
    maxPRCIndex = saveDict['meanprc'].index(maxPRC)
    nEstBest = saveDict['nEst'][maxPRCIndex]
    depthBest = saveDict['depth'][maxPRCIndex]
    learnRateBest = saveDict['learnRate'][maxPRCIndex]
    print(maxPRC,nEstBest,depthBest,learnRateBest)
    return maxPRC,nEstBest,depthBest,learnRateBest
maxPRC,nEstBest,depthBest,learnRateBest = findBestGBM(Xfillna,y)

cv = StratifiedShuffleSplit(n_splits=1, test_size=0.01,random_state=0)

#最优模型 GDBT 特征重要性求解
for train, test in cv.split(Xfillna,y):
    bestClassifier = ensemble.GradientBoostingClassifier(
                        n_estimators=1100, max_depth=2,

learning_rate=0.003,max_features=int(np.sqrt(Xfillna.shape[1])))
    bestClassifier.fit(Xfillna[train],y[train])
    predictions = bestClassifier.predict(Xfillna[test])
    precision, recall, thresholds =
precision_recall_curve(y[test],predictions)
    prc_auc = auc(recall, precision)
    print(prc_auc)
param = bestClassifier.get_params()
featureImportance = bestClassifier.feature_importances_
estimations = bestClassifier.estimators_
# normalize by max importance
featureImportance = featureImportance / featureImportance.max()
#plot importance of top 15
idxSorted = np.argsort(featureImportance)[15:30]
idxTemp = np.argsort(featureImportance)[::-1]
print(idxTemp)
barPos = np.arange(idxSorted.shape[0]) + .5
plot.barh(barPos, featureImportance[idxSorted], align='center')
plot.yticks(barPos, np.array(namesAfter)[idxSorted])
```

```
plot.xlabel('Variable Importance')
plot.show()

#RF
saveDict = {}
saveDict['nEst'] = []
saveDict['depth'] = []
saveDict['meanprc'] = []
for nEst in list(range(100,2001,100)):
    for depth in list(range(1,6)):
            maxFeatures = int(np.sqrt(Xfillna.shape[1]))

            prcList = []
            for train, test in cv.split(Xfillna,y):
                classifier = ensemble.RandomForestClassifier(
                        n_estimators=nEst, max_depth=depth,
                        max_features=maxFeatures)
                classifier.fit(Xfillna[train],y[train])
                predictions = classifier.predict(Xfillna[test])
                precision, recall, thresholds =
precision_recall_curve(y[test],predictions)
                prc_auc = auc(recall, precision)
                prcList.append(prc_auc)
            saveDict['meanprc'].append(np.mean(prcList))
            saveDict['nEst'].append(nEst)
            saveDict['depth'].append(depth)
            print(prcList,np.mean(prcList),nEst,depth)
maxPRC =  max(saveDict['meanprc'])
maxPRCIndex = saveDict['meanprc'].index(maxPRC)
nEstBest = saveDict['nEst'][maxPRCIndex]
depthBest = saveDict['depth'][maxPRCIndex]
print(maxPRC,nEstBest,depthBest)
```

代码七：KMV 模型

- **获取 KMV 模型所需数据**

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 25 14:54:22 2017

@author: Administrator
"""
import pandas as pd
from pandas import DataFrame,Series
import sys
import numpy as np
import re
from datetime import *
from imp import reload
import scipy.io as sio

from WindPy import *
from datetime import *
w.start()

#国债 1 年期收益率
data5 = w.wsd("CGB1Y.WI",
"close,wgsd_liabs_curr,wgsd_liabs_lt,wgsd_stkhldrs_eq", "2011-01-01", "2017-
08-24", "unit=1;rptType=1;currencyType=;PriceAdj=F")
riskFreeRate = DataFrame(data5.Data).T[0]

codesList =
["002506.SZ","002306.SZ","000659.SZ","600188.SH","600795.SH","002594.SZ","60
0783.SH"]
nameList =
['chaori','xiange','zhongfu','chongzhou','guoli','biyadi','luxinzhai']
beginTime = "2011-01-01"
endTime = "2017-08-24"
def KMVDataWind(codesList,nameList,beginTime,endTime,riskFreeRate):
    inputDict = {}
    for i,code in  enumerate(codesList):
        dataWind = w.wsd(code,
"close,wgsd_liabs_curr,wgsd_liabs_lt,wgsd_stkhldrs_eq", \
                         beginTime, endTime,
"unit=1;rptType=1;currencyType=;PriceAdj=F")
        data = DataFrame(dataWind.Data).T
```

```
        dataTime = DataFrame(dataWind.Times)
        inputDict[nameList[i]] = {}
        inputDict[nameList[i]]['debt'] = []
        inputDict[nameList[i]]['equity'] = []
        inputDict[nameList[i]]['sigma'] = []
        inputDict[nameList[i]]['riskFreeRate'] = []
        inputDict[nameList[i]]['time'] = []
        inputDict[nameList[i]]['tau'] = []
        dataIndex = list(data.iloc[:,1:4].dropna().index)
        dataIndex.insert(0,0)
        price = data[0]
        currentDebt = data[1][dataIndex[1:]]
        longDebt = data[2][dataIndex[1:]]
        equity = data[3][dataIndex[1:]]
        time = dataTime[0][dataIndex[1:]]
        debt = currentDebt + 0.5*longDebt
        debt = debt.shift(-1)
        ndataIndex = len(dataIndex)
        stdList = []
        rateList = []
        tauList = []
        for j in range(ndataIndex-1):
            std = np.std(price[dataIndex[j]:dataIndex[j+1]])*np.sqrt(252)
            stdList.append(std)
            rate = np.mean(riskFreeRate[dataIndex[j]:dataIndex[j+1]])
            rateList.append(rate)
            tau = (dataIndex[j+1]-dataIndex[j])/252
            tauList.append(tau)
        tauSeries = Series(tauList).shift(-1)
        inputDict[nameList[i]]['tau'].append(tauSeries)
        inputDict[nameList[i]]['debt'].append(debt)
        inputDict[nameList[i]]['equity'].append(equity)
        inputDict[nameList[i]]['sigma'].append(stdList)
        inputDict[nameList[i]]['riskFreeRate'].append(rateList)
        inputDict[nameList[i]]['time'].append(time)
    return inputDict
inputDict =  KMVDataWind(codesList,nameList,beginTime,endTime,riskFreeRate)
sio.savemat('F:\\GFsummer2017\\inputDict.mat', inputDict)
```

KMV 计算

```
clc;clear all
load  inputDict
Tau = chaori.tau
TD = chaori.debt
TE = chaori.equity
sigmaP = chaori.sigma
rf = chaori.riskFreeRate
n = length(T)
for i=1:n
    r=rf(i);%无风险利率
    T=Tau(i);%债务到期时间
    D=TD(i);  %高等索取权：无风险债务
    Equity=sigmaP(i); %低等索取权波动率
    E=TE(i) ;  %低等索取权
    [Va,AssetTheta]=KMVOptSearch(E,D,r,T,Equity);
    DD=(Va-D)/(Va*AssetTheta)
    EDF(i)=normcdf(-DD)
end

function [Va,AssetTheta]=KMVOptSearch(E,D,r,T,Equity)
EtoD=E/D;
x0=[1,1];
VaThetaX=fsolve(@(x)KMVfun(EtoD,r,T,Equity,x),x0);
Va=VaThetaX(1)*E;  %隐含资产价值
AssetTheta=VaThetaX(2);%银行资产价值波动率

function F=KMVfun(EtoD,r,T,Equity,x)
d1=(log(x(1)*EtoD)+(r+0.5*x(2)^2)*T)/(x(2)*sqrt(T));
d2=d1-x(2)*sqrt(T);
F=[x(1)*normcdf(d1)-exp(-r*T)*normcdf(d2)/EtoD-1;normcdf(d1)*x(1)*x(2)-
Equity];
```

代码八：CoVaR

分位数回归

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: hudie
"""
from scipy.optimize import minimize
import numpy as np

def minimizer_func(beta, fit_func, x_vals, q, observations):
    return q*np.sum(np.abs(np.where(observations>fit_func(beta,
x_vals),observations - fit_func(beta, x_vals),0))) + \
          (1-q)*np.sum(np.abs(np.where(observations<fit_func(beta,
x_vals),observations - fit_func(beta, x_vals),0)))

def quantile_regression(fit_func, x_vals, observations, beta_init,
bounds=None, q_value = 0.5):
    return minimize(minimizer_func, beta_init, args=(fit_func, x_vals,
q_value, observations), bounds=bounds)
```

CoVaR

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 29 22:37:06 2017
@author: hudie
"""

import sys
sys.path.append('../')
import qreg
import numpy as np
import pandas as pd
from pandas import DataFrame,Series
import os
import re
import matplotlib.pyplot as plot
plot.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plot.rcParams['axes.unicode_minus']=False #用来正常显示负号
```

```
from WindPy import *
from datetime import *
w.start()

codesList = ["000001.SZ","600000.SH","600015.SH","600016.SH","600036.SH",
            "601166.SH","601288.SH","601328.SH","601398.SH","601818.SH",
            "601939.SH","601988.SH","601998.SH","002142.SZ","002807.SZ",
            "002839.SZ","600908.SH","600919.SH","600926.SH","601009.SH",
            "601128.SH","601169.SH","601229.SH","601997.SH","603323.SH",
            "601318.SH","601336.SH","601601.SH","601628.SH","000166.SZ",
            "000416.SZ","000686.SZ","000728.SZ","000750.SZ","000776.SZ",
            "000783.SZ","000987.SZ","002500.SZ","002670.SZ","002673.SZ",
            "002736.SZ","002797.SZ","600030.SH","600061.SH","600109.SH",
            "600369.SH","600621.SH","600837.SH","600909.SH","600958.SH",
            "600999.SH","601099.SH","601198.SH","601211.SH","601375.SH",
            "601377.SH","601555.SH","601688.SH","601788.SH","601878.SH",
            "601881.SH","601901.SH","000001.SZ","600000.SH","600015.SH",
            "600036.SH","600016.SH","601166.SH","601288.SH","601328.SH",
            "601398.SH","601818.SH","601939.SH","601988.SH","601998.SH",
            "002142.SZ","002807.SZ","002839.SZ","600908.SH","600919.SH",
            "600926.SH","601009.SH","601128.SH","601169.SH","601229.SH",
            "601997.SH","603323.SH","601318.SH","601336.SH","601601.SH",
            "601628.SH","000166.SZ","000416.SZ","000686.SZ","000728.SZ",
            "000750.SZ","000776.SZ","000783.SZ","000987.SZ","002500.SZ",
            "002670.SZ","002673.SZ","002736.SZ","002797.SZ","600030.SH",
            "600061.SH","600109.SH","600369.SH","600621.SH","600837.SH",
            "600909.SH","600958.SH","600999.SH","601099.SH","601198.SH",
            "601211.SH","601375.SH","601377.SH","601555.SH","601688.SH",
            "601788.SH","601878.SH","601881.SH","601901.SH"]

MESingle = {}
LEVSingle = {}
for code in codesList:
    res = w.wsd(code,
"close,sec_name,trade_code,total_shares,tot_assets,tot_equity",\
              "2007-01-01", "2017-08-29",
"unit=1;rptType=1;currencyType=;Period=W;PriceAdj=F")
    tempData = res.Data
    time = res.Times
    MESingle[tempData[1][0]] = []
    LEVSingle[tempData[1][0]] = []
    price = Series(tempData[0])
    stock = Series(tempData[3])
    asset = Series(tempData[4])
```

```
    equity = Series(tempData[5])
    ME = price*stock
    LEV = asset/equity
#    ret = np.log(price/price.shift(1))
    MESingle[tempData[1][0]] = ME
    LEVSingle[tempData[1][0]] = LEV


MESingle = DataFrame(MESingle)
LEVSingle = DataFrame(LEVSingle).fillna(method='backfill')
retSingle = MESingle*LEVSingle
'''处理缺失值'''
nrows, ncols = retSingle.shape
totalMissingData = retSingle.isnull().sum().sort_values(ascending=False)
percentMissingData =
(retSingle.isnull().sum()/retSingle.isnull().count()).sort_values(ascending=
False)
missingData = pd.concat([totalMissingData,percentMissingData], axis=1,
keys=['Total', 'Percent'])
sys.stdout.write('head information \n')
#print(missingData.head(int(ncols)))
delColumns = percentMissingData > 0.85 #从国泰君安保留起
for columns in list(delColumns.keys()):
    try:
        if delColumns[columns]:
            del retSingle[columns]
    except KeyError:
        continue
names = retSingle.keys()
retSingle.dropna(axis=0, how='all',inplace=True)
startIndex = np.max(np.where(retSingle['国泰君安'].isnull()==True))
retSingle = retSingle.iloc[startIndex:,:]

weight = np.array(1/np.sum(retSingle.shift(1),axis=1))
noNorMVC = np.array(retSingle-retSingle.shift(1))
MVCsingle = np.zeros(list(noNorMVC.shape))
for i in range(2,noNorMVC.shape[0]):
    MVCsingle[i,:] = noNorMVC[i,:]*weight[i]
MVCsingle = 100*MVCsingle[2:,:]
MVCTotal = np.sum(MVCsingle,axis=1)

def getListFiles(path):
    rootList = []
    dirList = []
```

```
    fileList = []
    for root, dirs, files in os.walk(path):
        rootList.append(root)
        dirList.append(dirs)
        fileList.append(files)
    return rootList,dirList,fileList

rootList,dirList,fileList = getListFiles('./')
matchWord = '.xlsx'
pattern = re.compile(matchWord)
rawXDict = {}
for file in fileList[0]:
    match = pattern.search(file)
    if match:
        rawXDict[file] = pd.read_excel(file,sheet_name = 'Sheet1').iloc[:,1]
xVariable = {}
xVariable['counterPartyLiquiditySpread'] = rawXDict['银行质押 3m.xlsx'] -
rawXDict['国债收益 3m.xlsx']
xVariable['futureVolatility'] = rawXDict['沪深 300 期权波动率.xlsx']
xVariable['3mBond'] = rawXDict['国债收益 3m.xlsx']
xVariable['yieldCurve'] = rawXDict['国债收益 10y.xlsx'] - rawXDict['国债收益
3m.xlsx']
xVariable['creditSpread'] = rawXDict['铁道债 3y.xlsx'] - rawXDict['国债收益
3y.xlsx']
xVariable['hs300'] =  100*np.log(rawXDict['沪深 300.xlsx']/rawXDict['沪深
300.xlsx'].shift(1)).dropna()

xVariable = DataFrame(xVariable)
xVariable = np.array(xVariable)

m,n = MVCsingle.shape
VaRq = np.zeros([m,n])
VaRm = np.zeros([m,n])
CoVaR = np.zeros([m,n])
VaRsysM = np.zeros([m,n])
deltaCoVaR = np.zeros([m,n])
xVariable = np.column_stack((np.ones([m,1]), xVariable[:m,:]))
mx,nx = xVariable.shape


def funcSingle(beta, x):
    return
beta[0]*x[:,0]+beta[1]*x[:,1]+beta[2]*x[:,2]+beta[3]*x[:,3]+beta[4]*x[:,4]+b
eta[5]*x[:,5]+beta[6]*x[:,6]
```

```
def funcTotal(beta, x):
    return
beta[0]*x[:,0]+beta[1]*x[:,1]+beta[2]*x[:,2]+beta[3]*x[:,3]+beta[4]*x[:,4]+b
eta[5]*x[:,5]+beta[6]*x[:,6]+beta[7]*x[:,7]

Beta_VaRsysM =
qreg.quantile_regression(funcSingle,xVariable,MVCTotal,[1,1,1,1,1,1,1],q_val
ue = 0.5).x
VaRsysM =
Beta_VaRsysM[0]*xVariable[:,0]+Beta_VaRsysM[1]*xVariable[:,1]+Beta_VaRsysM[2
]*xVariable[:,2]+Beta_VaRsysM[3]*xVariable[:,3]+\

Beta_VaRsysM[4]*xVariable[:,4]+Beta_VaRsysM[5]*xVariable[:,5]+Beta_VaRsysM[6
]*xVariable[:,6]
for i in range(n):
    Beta_VaRq =
qreg.quantile_regression(funcSingle,xVariable,MVCsingle[:,i],[1,1,1,1,1,1,1]
,q_value = 0.05).x
    VaRq[:,i] =
Beta_VaRq[0]*xVariable[:,0]+Beta_VaRq[1]*xVariable[:,1]+Beta_VaRq[2]*xVariab
le[:,2]+Beta_VaRq[3]*xVariable[:,3]+\

Beta_VaRq[4]*xVariable[:,4]+Beta_VaRq[5]*xVariable[:,5]+Beta_VaRq[6]*xVariab
le[:,6]
    XVariable = np.column_stack((xVariable, MVCsingle[:,i]))
    Beta_CoVaR =
qreg.quantile_regression(funcTotal,XVariable,MVCTotal,[1,1,1,1,1,1,1,1],q_va
lue = 0.05).x
    CoVaR[:,i] =
Beta_CoVaR[0]*XVariable[:,0]+Beta_CoVaR[1]*XVariable[:,1]+Beta_CoVaR[2]*XVar
iable[:,2]+Beta_CoVaR[3]*XVariable[:,3]+\

Beta_CoVaR[4]*XVariable[:,4]+Beta_CoVaR[5]*XVariable[:,5]+Beta_CoVaR[6]*XVar
iable[:,6]+Beta_CoVaR[7]*VaRq[:,i]
    deltaCoVaR[:,i] = CoVaR[:,i] - VaRsysM
deltaCoVaR = -deltaCoVaR
deltaCoVaR = DataFrame(deltaCoVaR.T, index = list(names)).T
deltaCoVaR.to_excel('deltaCoVaR.xlsx',sheet_name = 'Sheet1')
print(np.mean(deltaCoVaR.iloc[1:,:],axis=0).sort_values())
```