企业非法集资风险预测总结综述

李*、车*、杨**、李敬、陈**

2020.12

北京 · 怀柔

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

Contents

赛题简介 & 小组分工

企业非法集资风险预测，是CCF主办的第八届BDCI大赛中的赛题，组委会给的数据集包含约25000家企业数据，其中约15000家企业带标注数据作为训练集，剩余数据作为测试集[1]。

数据由企业基本信息、企业年报、企业纳税情况等组成，数据包括数值型、字符型、日期型等众多数据类型（已脱敏），部分字段内容在部分企业中有缺失，其中第一列id为企业唯一标识。

[1] CCF企业非法集资风险预测竞赛 [OL]. https://www.datafountain.cn/competitions/469 (accessed Jan. 03, 2021).

数据初步分析

- 整理数据集可以知道，数据集一共可分成8张表,每一个表的内容都不一样，里面所含的特征、企业数量等，都是不同的。

缺失部分分析

- 每一行代表一个企业的基本数据，每一行有33列，其中id列为企业唯一标识

列名	含义	列名	含义	列名	含义	列名	含义
id	企业唯一标识	oplocdistrict	行政区划代码	industryphy	行业类别代码	industryco	行业细类代码
dom	经营地址	opscope	经营范围	enttype	企业类型	enttypeitem	企业类型小类
opfrom	经营期限起	opto	经营期限止	state	状态	orgid	机构标识
jobid	职位标识	adbusign	是否广告经营	townsign	是否城镇	regtype	主题登记类型
empnum	从业人数	compform	组织形式	parnum	合伙人数	exenum	执行人数
opform	经营方式	ptbusscope	兼营范围	venind	风险行业	enttypeminu	企业类型细类
midpreindcode	中西部优势产业代码	protype	项目类型	oploc	经营场所	regcap	注册资本（金）
reccap	实缴资本	forreccap	实缴资本（外方）	forregcap	注册资本（外方）	congro	投资总额

```
#读取数据
base_info = pd.read_csv(PATH + 'base_info.csv')
#输出数据shape和不重复企业id数
print(base_info.shape, base_info['id'].nunique())
#读取数据
base_info.head(1)
#查看缺失值，这里借助了missingno这个包，import missingno as msno。
msno.bar(base_info)#查看缺失值
```

剔除缺失并处理

```python
#用于剔除空值的函数
def filter_col_by_nan(df, ratio=0.05):
    cols = []
    for col in df.columns:
        if df[col].isna().mean() >= (1-ratio):
            cols.append(col)
    return cols
```

- 这里给个参数ratio用于控制缺失值比例
- 0.05，缺失值超过95%这个特征就剔除

特征工程

- 这是对上述主表的处理
- opscope ：
  - 方法一：按照分号，将每个经营范围分出来，然后每个经营范围是一个特征.
  - 方法二：经…标为1，未经金融…标为2，如果还有其他的标3 粗暴一点[1]
  - 方法三：自然语言处理

  关键词：词向量

orgid：机构标识，只要能区分不同就可以，但题目中给出的数值过大，所以处理成较小的1-n

jobid：职位标识，同上

opform：经营方式，处理成1-n

ptbusscope: 无任何值，缺失过于严重，删除该特征

enttypeminu：1-n

midpreindcodep，rotype：无任何值，删除该特征

oploc：经营场所1-n

enttypegb：1-n

```
#orgid  机构标识 oplocdistrict  行政区划代码    jobid 职位标识
base_info['district_FLAG1'] = (base_info['orgid'].fillna('').apply(lambda x: str(x)[:6]) == \
    base_info['oplocdistrict'].fillna('').apply(lambda x: str(x)[:6])).astype(int)
base_info['district_FLAG2'] = (base_info['orgid'].fillna('').apply(lambda x: str(x)[:6]) == \
    base_info['jobid'].fillna('').apply(lambda x: str(x)[:6])).astype(int)
base_info['district_FLAG3'] = (base_info['oplocdistrict'].fillna('').apply(lambda x: str(x)[:6]) == \
    base_info['jobid'].fillna('').apply(lambda x: str(x)[:6])).astype(int)

#parnum 合伙人数    exenum  执行人数  empnum    从业人数
base_info['person_SUM'] = base_info[['empnum', 'parnum', 'exenum']].sum(1)
base_info['person_NULL_SUM'] = base_info[['empnum', 'parnum', 'exenum']].isnull().astype(int).sum(1)

#regcap 注册资本（金） congro  投资总额
# base_info['regcap_DIVDE_empnum'] = base_info['regcap'] / base_info['empnum']
# base_info['regcap_DIVDE_exenum'] = base_info['regcap'] / base_info['exenum']

# base_info['reccap_DIVDE_empnum'] = base_info['reccap'] / base_info['empnum']
# base_info['regcap_DIVDE_exenum'] = base_info['regcap'] / base_info['exenum']

#base_info['congro_DIVDE_empnum'] = base_info['congro'] / base_info['empnum']
#base_info['regcap_DIVDE_exenum'] = base_info['regcap'] / base_info['exenum']

base_info['opfrom'] = pd.to_datetime(base_info['opfrom'])#opfrom    经营期限起
base_info['opto'] = pd.to_datetime(base_info['opto'])#opto  经营期限止
base_info['opfrom_TONOW'] = (datetime.now() - base_info['opfrom']).dt.days
base_info['opfrom_TIME'] = (base_info['opto'] - base_info['opfrom']).dt.days

#opscope    经营范围
base_info['opscope_COUNT'] = base_info['opscope'].apply(lambda x: len(x.replace("\t",
", ").replace("\n", ", ").split('、')))

#对类别特征做处理
cat_col = ['oplocdistrict', 'industryphy', 'industryco', 'enttype',
           'enttypeitem', 'enttypeminu', 'enttypegb',
          'dom', 'oploc', 'opform','townsign']
#如果类别特征出现的次数小于10转为-1
for col in cat_col:
    base_info[col + '_COUNT'] = base_info[col].map(base_info[col].value_counts())
    col_idx = base_info[col].value_counts()
    for idx in col_idx[col_idx < 10].index:
        base_info[col] = base_info[col].replace(idx, -1)

# base_info['opscope'] = base_info['opscope'].apply(lambda x: x.replace("\t", " ").replace("\n", "
").replace(", ", " "))
# clf_tfidf = TfidfVectorizer(max_features=200)
# tfidf=clf_tfidf.fit_transform(base_info['opscope'])
# tfidf = pd.DataFrame(tfidf.toarray())
# tfidf.columns = ['opscope_' + str(x) for x in range(200)]
# base_info = pd.concat([base_info, tfidf], axis=1)

base_info = base_info.drop(['opfrom', 'opto'], axis=1)#删除时间

for col in ['industryphy', 'dom', 'opform', 'oploc']:
    base_info[col] = pd.factorize(base_info[col])[0]
```

[1] 甘鹭. (2017). 基于机器学习算法的信用风险预测模型研究[D]. (Doctoral dissertation, 北京: 北京交通大学).

模型训练

五折交叉验证

- 单模lightbgm
- Catboost
- 融合
- 效果显著

```python
def eval_score(y_test,y_pre):
    _,_,f_class,_=precision_recall_fscore_support(y_true=y_test,y_pred=y_pre,labels=[0,1],average=None)
    fper_class={'合法':f_class[0],'违法':f_class[1],'f1':f1_score(y_test,y_pre)}
    return fper_class


def k_fold_serachParmaters(model,train_val_data,train_val_kind, test_kind):
    mean_f1=0
    mean_f1Train=0
    n_splits=5

    cat_features = ['oplocdistrict', 'industryphy', 'industryco', 'enttype',
           'enttypeitem', 'enttypeminu', 'enttypegb',
          'dom', 'oploc', 'opform']

    sk = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=2021)
    pred_Test = np.zeros(len(test_kind))
    for train, test in sk.split(train_val_data, train_val_kind):
        x_train = train_val_data.iloc[train]
        y_train = train_val_kind.iloc[train]
        x_test = train_val_data.iloc[test]
        y_test = train_val_kind.iloc[test]

        model.fit(x_train, y_train,
                  eval_set=[(x_test, y_test)],
                  categorical_feature = cat_features,
                 early_stopping_rounds=100,
                 verbose=False)

        pred = model.predict(x_test)
        fper_class = eval_score(y_test,pred)#验证集的准确率

        pred_Train = model.predict(x_train)
        pred_Test += model.predict_proba(test_kind)[:, 1]/n_splits
        fper_class_train = eval_score(y_train,pred_Train)

        mean_f1 += fper_class['f1']/n_splits
        mean_f1Train+=fper_class_train['f1']/n_splits
        # print(mean_f1, mean_f1Train)


    return mean_f1, pred_Test
```

循环执行（调优）

循环执行代码

- 参数加入随机性 sendrandom
- 然后训练20次
- 迭代

```
score_tta = None
score_list = []

tta_fold = 20
for _ in range(tta_fold):
    clf = lgb.LGBMClassifier(
        num_leaves=np.random.randint(6, 10), min_child_samples= np.random.randint(2,5),
        max_depth=5,learning_rate=0.03,
        n_estimators=150,n_jobs=-1,silent=False)

    score, test_pred = k_fold_serachParmaters(clf,
                           train_data.drop(['id', 'opscope','label'], axis=1),
                           train_data['label'],
                           test_data.drop(['id', 'opscope'], axis=1),
                          )

    if score_tta is None:
        score_tta = test_pred/tta_fold
    else:
        score_tta += test_pred/tta_fold
    # print(score)
    score_list.append(score)

print(np.array(score_list).mean(), np.array(score_list).std())
```

继续调优

自动寻最优参数

- 手动不上分
- 自寻
- 参数回代

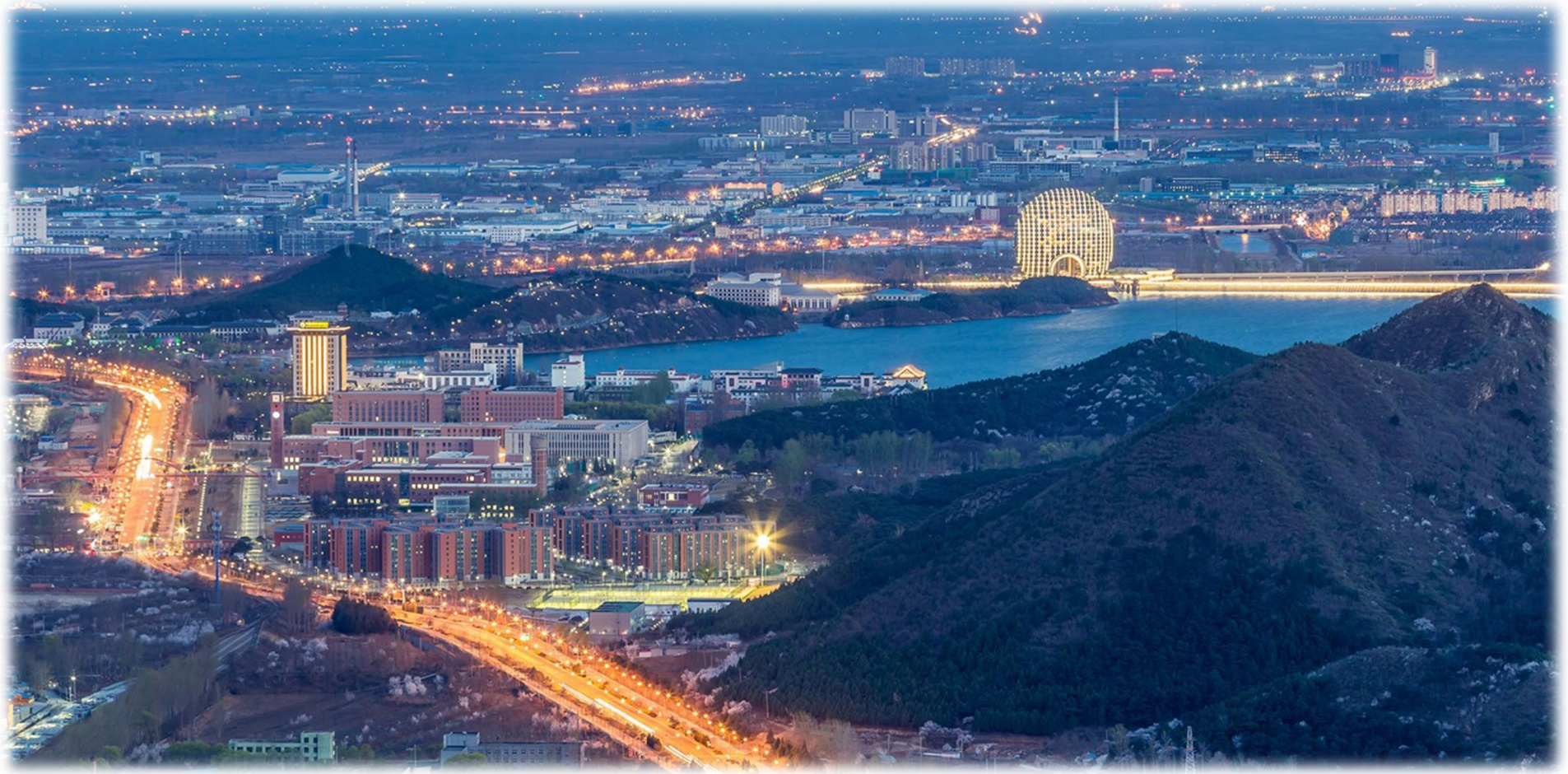
```
lg = lgb.LGBMClassifier(silent=False)
param_dist = {"max_depth": [4,5,6,7,8],
              "learning_rate" : [0.01,0.03,0.05,0.07,0.09],
              "num_leaves": [4, 5, 6, 7, 8],
              "n_estimators": [50, 100, 150,200]
             }

cat_features = ['oplocdistrict', 'industryphy', 'industryco', 'enttype',
           'enttypeitem', 'enttypeminu', 'enttypegb',
          'dom', 'oploc', 'opform']


grid_search = GridSearchCV(lg, n_jobs=-1, param_grid=param_dist, cv = 5, scoring='f1', verbose=5)
grid_search.fit(train_data.drop(['id', 'opscope','label'], axis=1),
                           train_data['label'], categorical_feature = cat_features,)
grid_search.best_estimator_, grid_search.best_score_
```

最终成绩

我们小组也是初次边学习参加这种结构化的比赛，再接再厉！



欢迎批评指正

THANKS

中国科学院大学
University of Chinese Academy of Sciences