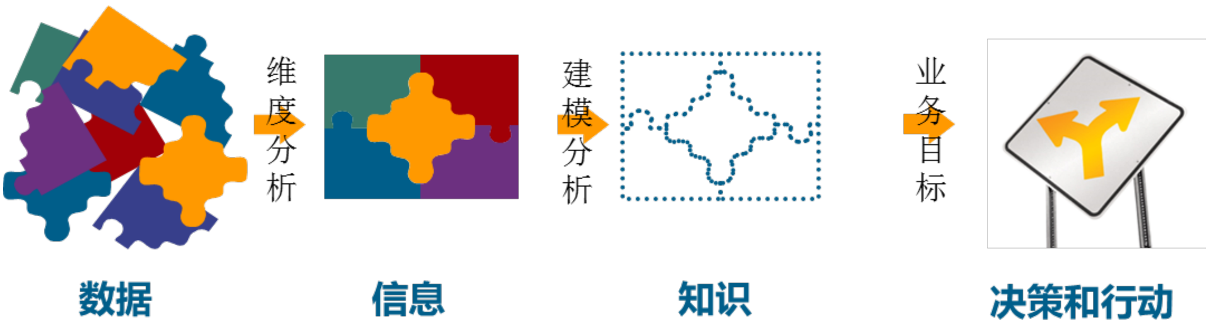
违约客户的影响因素分析

关键字：逻辑回归，银行，贷款，违约，预测，Logistics Regression

运行环境：Python3.5, VS Code

违约客户的影响因素分析

一.数据准备

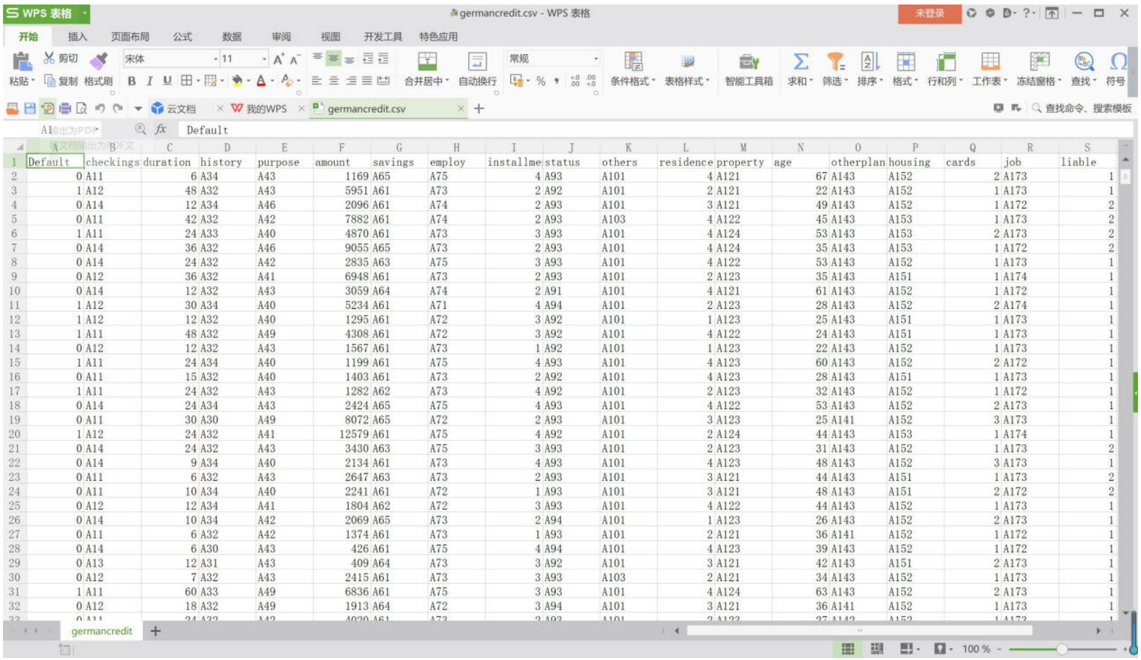


1．数据介绍

现已有某德国银行个人贷款数据，通过这个模型对新增的贷款人“是否具有偿还能力，是否具有偿债意愿”进行分析，预测贷款申请人是否会发生违约贷款。这是一个监督学习的场景，因为已知了特征以及贷款状态是否违约（目标列），我们判定贷款申请人是否违约是一个**二元分类问题**，可以通过一个**分类算法**来处理，这里选用逻辑斯蒂回归（Logistic Regression）。**构建贷款违约预测模型**，对新增贷款申请人进行预测是否会违约，从而决定是否放款。

2．数据输入

```
# 导入需要使用的库
import pandas as pd
import re
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# 导入并处理数据
bank_data = pd.read_csv("./data/germancredit.csv")
```



Default	checkings	duration	history	purpose	amount	savings	employ	installme	status	others	residence	property	age	otherplan	housing	cards	job	liable
0	A11	6	A34	A43	1169	A65	A75	4	A93	A101	4	A121	67	A143	A152	2	A173	1
1	A12	48	A32	A43	5951	A61	A73	2	A92	A101	2	A121	22	A143	A152	1	A173	1
0	A14	12	A34	A46	2096	A61	A74	2	A93	A101	3	A121	49	A143	A152	1	A172	2
0	A11	42	A32	A42	7882	A61	A74	2	A93	A103	4	A122	45	A143	A153	1	A173	2
1	A11	24	A33	A40	4870	A61	A73	3	A93	A101	4	A124	53	A143	A153	2	A173	2
0	A14	36	A32	A46	9055	A65	A73	2	A93	A101	4	A124	35	A143	A153	1	A172	2
0	A14	24	A32	A42	2835	A63	A75	3	A93	A101	4	A122	53	A143	A152	1	A173	1
0	A12	36	A32	A41	6948	A61	A73	2	A93	A101	2	A123	35	A143	A151	1	A174	1
0	A14	12	A32	A43	3059	A64	A74	2	A91	A101	4	A121	61	A143	A152	1	A172	1
1	A12	30	A34	A40	5234	A61	A71	4	A94	A101	2	A123	28	A143	A152	2	A174	1
1	A12	12	A32	A40	1295	A61	A72	3	A92	A101	1	A123	25	A143	A151	1	A173	1
1	A11	48	A32	A49	4308	A61	A72	3	A92	A101	4	A122	24	A143	A151	1	A173	1
0	A12	12	A32	A43	1567	A61	A73	1	A92	A101	1	A123	22	A143	A152	1	A173	1
1	A11	24	A34	A40	1199	A61	A75	4	A93	A101	4	A123	60	A143	A152	2	A172	1
0	A11	15	A32	A40	1403	A61	A73	2	A92	A101	4	A123	28	A143	A151	1	A173	1
1	A11	24	A32	A43	1282	A62	A73	4	A92	A101	2	A123	32	A143	A152	1	A172	1
0	A14	24	A34	A43	2424	A65	A75	4	A93	A101	4	A122	53	A143	A152	2	A173	1
0	A11	30	A30	A49	8072	A65	A72	2	A93	A101	3	A123	25	A141	A152	3	A173	1
1	A12	24	A32	A41	12579	A61	A75	4	A92	A101	2	A124	44	A143	A153	1	A174	1
0	A14	24	A32	A43	3430	A63	A75	3	A93	A101	2	A123	31	A143	A152	1	A173	2
0	A14	9	A34	A40	2134	A61	A73	4	A93	A101	4	A123	48	A143	A152	3	A173	1
0	A11	6	A32	A43	2647	A63	A73	2	A93	A101	3	A121	44	A143	A151	1	A173	2
0	A11	10	A34	A40	2241	A61	A72	1	A93	A101	3	A121	48	A143	A151	2	A172	2
0	A12	12	A34	A41	1804	A62	A72	3	A93	A101	4	A122	44	A143	A152	1	A173	1
0	A14	10	A34	A42	2069	A65	A73	2	A94	A101	1	A123	26	A143	A152	2	A173	1
0	A11	6	A32	A42	1374	A61	A73	1	A93	A101	2	A121	36	A141	A152	1	A172	1
0	A14	6	A30	A43	426	A61	A75	4	A94	A101	4	A123	39	A143	A152	1	A172	1
0	A13	12	A31	A43	409	A64	A73	3	A92	A101	3	A121	42	A143	A151	2	A173	1
0	A12	7	A32	A43	2415	A61	A73	3	A93	A103	2	A121	34	A143	A152	1	A173	1
1	A11	60	A33	A49	6836	A61	A75	3	A93	A101	4	A124	63	A143	A152	2	A173	1
0	A12	18	A32	A49	1913	A64	A72	3	A94	A101	3	A121	36	A141	A152	1	A173	1

3. 描述数据

Default：是否违约

Checkingstatus1：现有账号情况

Duration：持续时间

History：信用历史

Purpose：贷款目的

Amount：信贷数额

Savings：储蓄金额

Employ：受雇时间

Installment：分期付款占可支配收入的百分比

Status：婚姻状况

Others：其它担保人

Residence：现有住房

Property：财产

Age：年龄

Otherplans：其它分期付款

Housing：住房状况

Cards：这家银行的信用卡数量

Job：工作类型

Liable：劳动力数量

Tele：电话

Foreign：是否是外国人

4. 查看描述统计数据

```
print(bank_data.describe())
```

```
  This module will be removed in 0.20.", DeprecationWarning)
          Default  checkingstatus1     duration      history      purpose  \
count  1000.000000      1000.000000  1000.000000  1000.00000  1000.000000
mean      0.300000        12.577000    20.903000    32.54500    47.148000
std       0.458487         1.257638    12.058814     1.08312    40.095333
min       0.000000        11.000000     4.000000    30.00000    40.000000
25%       0.000000        11.000000    12.000000    32.00000    41.000000
50%       0.000000        12.000000    18.000000    32.00000    42.000000
75%       1.000000        14.000000    24.000000    34.00000    43.000000
max       1.000000        14.000000    72.000000    34.00000   410.000000

             amount      savings       employ  installment       status  \
count   1000.000000  1000.000000  1000.000000  1000.000000   1000.00000
mean    3271.258000    62.105000    73.384000     2.973000     92.68200
std     2822.736876     1.580023     1.208306     1.118715      0.70808
min      250.000000    61.000000    71.000000     1.000000     91.00000
25%     1365.500000    61.000000    73.000000     2.000000     92.00000
50%     2319.500000    61.000000    73.000000     3.000000     93.00000
75%     3972.250000    63.000000    75.000000     4.000000     93.00000
max    18424.000000    65.000000    75.000000     4.000000     94.00000

           ...     residence     property          age   otherplans  \
count      ...   1000.000000  1000.000000  1000.000000  1000.000000
mean       ...      2.845000   122.358000    35.546000   142.675000
std        ...      1.103718     1.050209    11.375469     0.705601
min        ...      1.000000   121.000000    19.000000   141.000000
25%        ...      2.000000   121.000000    27.000000   143.000000
50%        ...      3.000000   122.000000    33.000000   143.000000
75%        ...      4.000000   123.000000    42.000000   143.000000
max        ...      4.000000   124.000000    75.000000   143.000000

           housing        cards          job       liable         tele  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean    151.929000     1.407000   172.904000     1.155000   191.404000
std       0.531264     0.577654     0.653614     0.362086     0.490943
min     151.000000     1.000000   171.000000     1.000000   191.000000
25%     152.000000     1.000000   173.000000     1.000000   191.000000
50%     152.000000     1.000000   173.000000     1.000000   191.000000
75%     152.000000     2.000000   173.000000     1.000000   192.000000
max     153.000000     4.000000   174.000000     2.000000   192.000000

           foreign
count  1000.000000
mean    201.037000
std       0.188856
min     201.000000
25%     201.000000
50%     201.000000
75%     201.000000
max     202.000000

[8 rows x 21 columns]
```

二. 数据预处理

1. 已知现有数据源可靠，不存在缺失值。

2. 数据标准化，去除数据中非数字字符‘A’。

```
bankData['checkingstatus1'] = bankData['checkingstatus1'].map(lambda x: int(re.sub("\D", "", x)))
bankData['history'] = bankData['history'].map(lambda x: int(re.sub("\D", "", x)))
bankData['purpose'] = bankData['purpose'].map(lambda x: int(re.sub("\D", "", x)))
bankData['savings'] = bankData['savings'].map(lambda x: int(re.sub("\D", "", x)))
bankData['employ'] = bankData['employ'].map(lambda x: int(re.sub("\D", "", x)))
bankData['status'] = bankData['status'].map(lambda x: int(re.sub("\D", "", x)))
bankData['others'] = bankData['others'].map(lambda x: int(re.sub("\D", "", x)))
bankData['property'] = bankData['property'].map(lambda x: int(re.sub("\D", "", x)))
bankData['otherplans'] = bankData['otherplans'].map(lambda x: int(re.sub("\D", "", x)))
bankData['housing'] = bankData['housing'].map(lambda x: int(re.sub("\D", "", x)))
bankData['job'] = bankData['job'].map(lambda x: int(re.sub("\D", "", x)))
```

```
bankData['tele'] = bankData['tele'].map(lambda x: int(re.sub("\D", "", x)))
bankData['foreign'] = bankData['foreign'].map(lambda x: int(re.sub("\D", "", x)))
```

三. 特征工程（特征的计算和选取）

特征选择是模型成功的基础性重要工作。一般特征筛选方法有

1.看模型系数显著性（F值大、P值小）

2. 递归特征消除：反复构建模型，根据变量系数选择最好特征，然后再递归在剩余变量上重复该过程，直到遍历所有特征。特征被挑选出顺序就是特征重要性排序顺序。

3. 稳定性选择:在不同特征子集、数据子集上运行算法，不断重复，最终汇总特征选择结果。统计，各个特征被认为是重要性特征的频率作为其重要性得分(被选为重要特征次数除以它所在子集被测试次数)。

4.所选择的特征列如下：

Checkingstatus1：现有账号情况

Duration：持续时间

History：信用历史

Purpose：贷款目的

Amount：信贷数额

Savings：储蓄金额

Employ：受雇时间

Installment：分期付款占可支配收入的百分比

Status：婚姻状况

Others：其它担保人

Residence：现有住房

Property：财产

Age：年龄

Otherplans：其它分期付款

Housing：住房状况

Cards：这家银行的信用卡数量

Job：工作类型

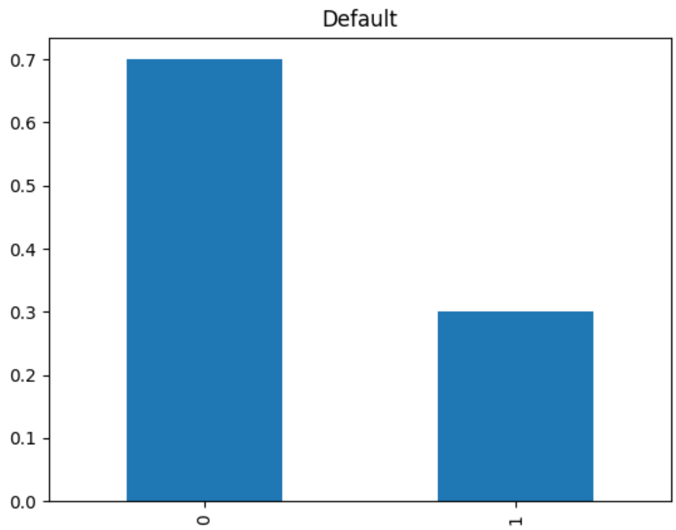
Liable：劳动力数量

Foreign：是否是外国人

四.单变量分析

1.违约情况统计：

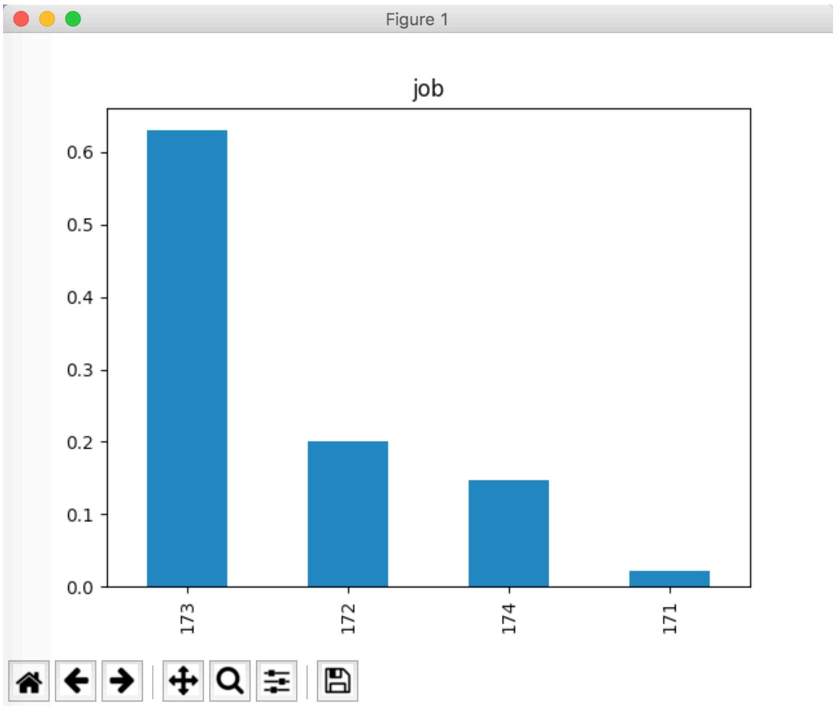
```
违约情况统计：0    0.7
1    0.3
Name: Default, dtype: float64
```



违约情况约占30%

2.工作情况分析

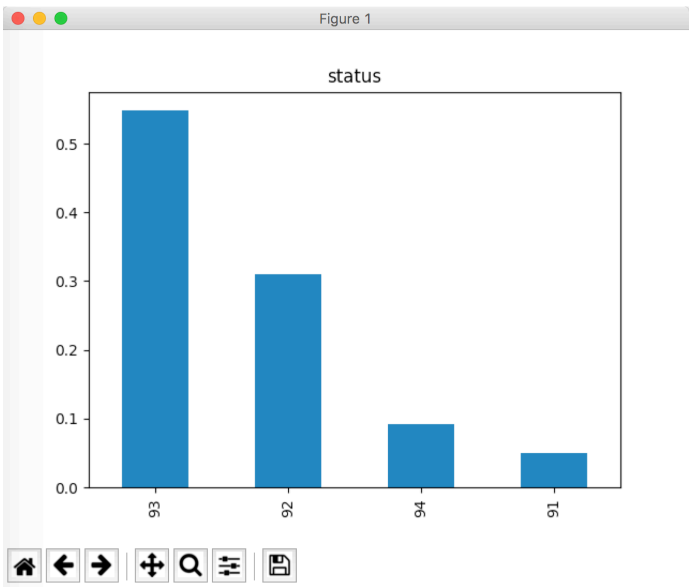
```
  "This module will be removed in 0
雇佣情况统计：173    0.630
172    0.200
174    0.148
171    0.022
Name: job, dtype: float64
```



173， skilled employee占比最高

3.婚姻情况统计：

```
ule into which all the refactored classes and
  the new CV iterators are different from that
   "This module will be removed in 0.20.", Dep
婚姻情况统计: 93    0.548
92    0.310
94    0.092
91    0.050
Name: status, dtype: float64
```



93，单身人士贷款比例最高

五.构建模型

```
#提取建模用数据
train_data = bank_data[:900]
# 提取需要进行预测的数据
predict_data = bank_data[900:]

# 去除无关变量
train_data = train_data.drop(['tele'], axis=1)
predict_data = predict_data.drop(['tele'], axis=1)
```

```
# 定义一个函数对因变量进行重新编码，编程成数值型，即0和1
def coding(col, code_dict):
colCoded = pd.Series(col, copy=True)
for key, value in code_dict.items():
colCoded.replace(key, value, inplace=True)
return colCoded
```

```
# 是=1, 否=0:
train_data["Default"] = coding(train_data['Default'], {'否':
0,'是':1})
# 将自变量与因变量分开
X,y =
train_data.drop(['Default'],axis=1),train_data[['Default']]
```

```
# 随机抽取训练集与测试集
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size = 0.3,random_state = 10)
# 开始构建一个逻辑回归模型
model = LogisticRegression()
```

六.训练模型

```
# 模型以X_train,y_train为输入数据进行训练
model.fit(X_train,y_train)
```

七.评估模型

```
# 打印针对测试集而言的准确率
print("预测测试集的准确率:",
accuracy_score(y_test,model.predict(X_test)))
# 使用训练得到模型对这些新申请贷款的人的违约风险进行预测
print("测试集中贷款申请人违约风险预测情况:",
model.predict(predict_data.drop(['Default'],axis=1)))
```

```
(py3) ➜  credit_predict python predict.py
/Users/dengyongqing/anaconda/envs/py3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This mo
.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the
s are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
/Users/dengyongqing/anaconda/envs/py3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataConversionWarning: A c
 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
预测测试集的准确率: 0.7518518518518519
测试集中贷款申请人违约风险预测情况: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0]
(py3) ➜  credit_predict
```