德国信贷数据建模baseline

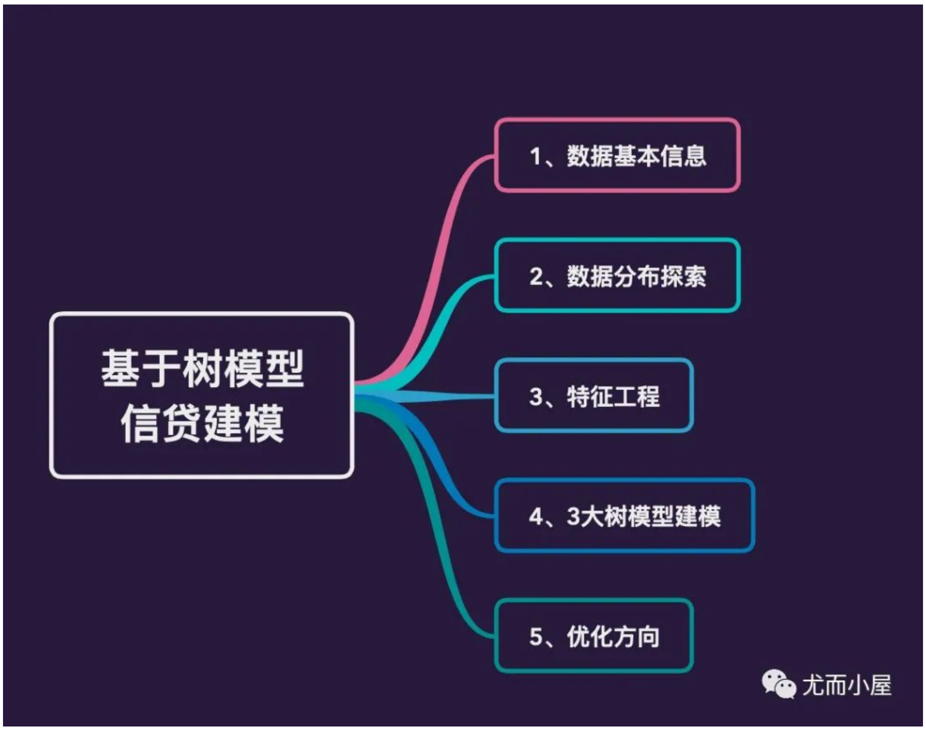
原创 尤而小屋 尤而小屋 2022-07-17 00:00 发表于北京

收录于合集
#kaggle 23 #人工智能 38 #机器学习 110 #数据可视化 61

> 公众号：尤而小屋
> 作者：Peter
> 编辑：Peter

大家好，我是Peter~

本文是基于3大树模型对一份德国信贷数据的简单建模，可以作为一份baseline，最后也提出了优化的方向。主要内容包含：



导入库

导入的库用于数据处理、可视化、建模等

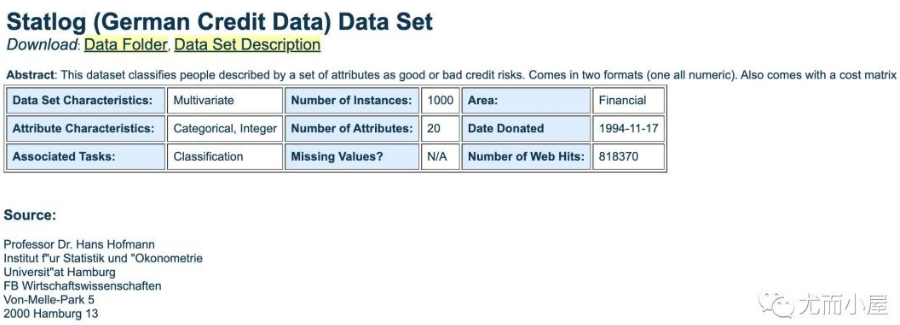
```
import pandas as pd
import numpy as np

# 1、基于plotly
import plotly as py
import plotly.express as px
import plotly.graph_objects as go
py.offline.init_notebook_mode(connected = True)
from plotly.subplots import make_subplots  # 多子图
# 2、基于matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
%matplotlib inline
```

```python
# 中文显示问题
#设置字体
plt.rcParams["font.sans-serif"]=["SimHei"]
#正常显示负号
plt.rcParams["axes.unicode_minus"]=False

# 3、基于seaborn
import seaborn as sns
# plt.style.use("fivethirtyeight")
plt.style.use('ggplot')

# 数据标准化、分割、交叉验证
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split,cross_val_score

# 模型
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor,DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# 模型评价
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix, Confusion
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, preci

# 忽略notebook中的警告
import warnings
warnings.filterwarnings("ignore")
```

数据简介

数据来自UCI官网：http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

基本信息：1000条数据 + 20个变量 + 目标变量 + 无缺失值



Statlog (German Credit Data) Data Set

Download: Data Folder, Data Set Description

Abstract: This dataset classifies people described by a set of attributes as good or bad credit risks. Comes in two formats (one all numeric). Also comes with a cost matrix

Data Set Characteristics:	Multivariate	**Number of Instances:**	1000	**Area:**	Financial
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	20	**Date Donated**	1994-11-17
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	818370

Source:

Professor Dr. Hans Hofmann
Institut f"ur Statistik und "Okonometrie
Universit"at Hamburg
FB Wirtschaftswissenschaften
Von-Melle-Park 5
2000 Hamburg 13

尤而小屋

特征变量的中文与英文含义：

- 特征向量中文：1.支票账户状态；2.借款周期；3.历史信用；4.借款目的；5.信用额度；6.储蓄账户状态；7.当前就业状态；8.分期付款占可支配收入百分比；9.性别与婚姻状态；10.他人担保信息；11.现居住地；12.财产状态；13.年龄；14.其他分期情况；15.房产状态；16.信用卡数量；17.工作状态；18.赡养人数；19.电话号码注册情况；20.是否有海外工作经历
- 特征向量对应英文：1.status_account, 2.duration, 3.credit_history, 4,purpose, 5.amount, 6.svaing_account, 7.present_emp, 8.income_rate, 9.personal_status, 10.other_debtors,

11.residence_info, 12.property, 13.age, 14.inst_plans, 15.housing, 16.num_credits, 17.job, 18.dependents, 19.telephone, 20.foreign_worker

读入数据

下载的数据没有表头，网上搜索到对应英文表头，生成DataFrame：

In [2]:

```
# 设置表头
columns = ["checking_account_status","duration","credit_history","purpose",
           "credit_amount","savings","present_employment","installment_rate",
           "personal","other_debtors","present_residence","property","age",
           "other_installment_plans","housing","existing_credits","job",
           "dependents","telephone","foreign_worker","customer_type"
          ]
```

In [3]:

```
df = pd.read_table("german.data",delimiter=' ',header=None, names=columns)

df.head()
```

Out[3]:

	checking_account_status	duration	credit_history	purpose	credit_amount	savings	present_employment	installment_rate	personal
0	A11	6	A34	A43	1169	A65	A75	4	A93
1	A12	48	A32	A43	5951	A61	A73	2	A92
2	A14	12	A34	A46	2096	A61	A74	2	A93
3	A11	42	A32	A42	7882	A61	A74	2	A93
4	A11	24	A33	A40	4870	A61	A73	3	A93

尤而小屋

5 rows × 21 columns

In [4]:

```
df.shape
```

Out[4]:

```
(1000, 21)
```

In [5]:

```
df.dtypes  # 字段类型
```

Out[5]:

```
checking_account_status    object
duration                    int64
credit_history             object
purpose                    object
credit_amount               int64
savings                    object
present_employment         object
installment_rate            int64
personal                   object
other_debtors              object
present_residence           int64
property                   object
age                         int64
other_installment_plans    object
housing                    object
existing_credits            int64
job                        object
dependents                  int64
telephone                  object
foreign_worker             object
customer_type               int64
dtype: object
```

In [6]:

```
# 不同的字段类型统计

pd.value_counts(df.dtypes.values)
```

Out[6]:

```
object    13
int64      8
dtype: int64
```

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
checking_account_status    0
duration                   0
credit_history             0
purpose                    0
credit_amount              0
savings                    0
present_employment         0
installment_rate           0
personal                   0
other_debtors              0
present_residence          0
property                   0
age                        0
other_installment_plans    0
housing                    0
existing_credits           0
job                        0
dependents                 0
telephone                  0
foreign_worker             0
customer_type              0
dtype: int64
```

不同字段下的取值统计

In [8]:

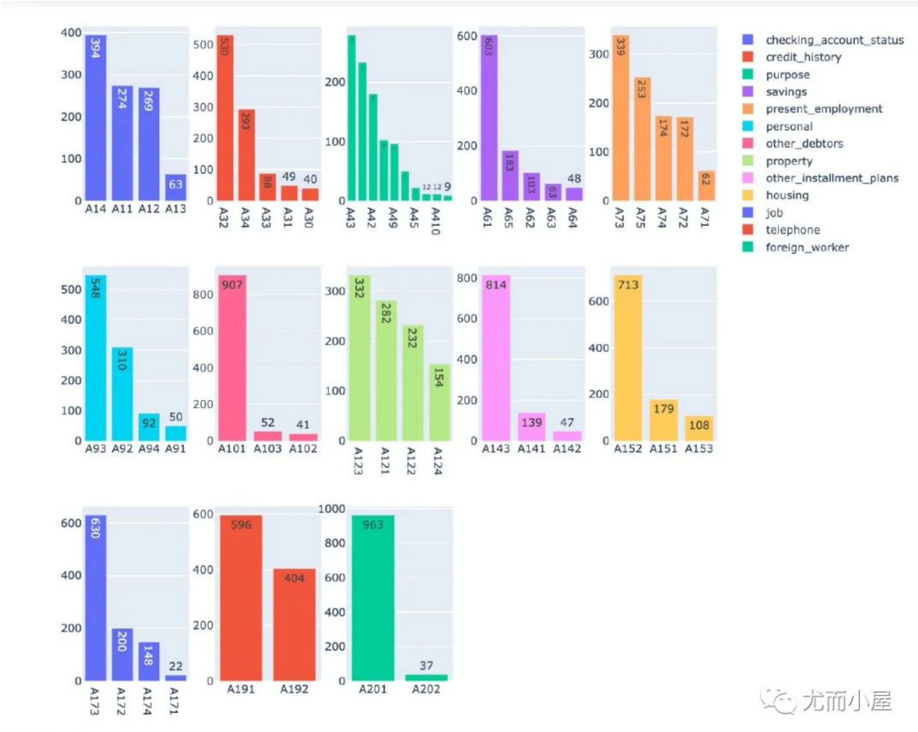
```
columns = df.columns     # 字段
columns
```

Out[8]:

```
Index(['checking_account_status', 'duration', 'credit_history', 'purpose',
       'credit_amount', 'savings', 'present_employment', 'installment_rate',
       'personal', 'other_debtors', 'present_residence', 'property', 'age',
       'other_installment_plans', 'housing', 'existing_credits', 'job',
       'dependents', 'telephone', 'foreign_worker', 'customer_type'],
      dtype='object')
```

1、针对字符类型字段的取值情况统计：

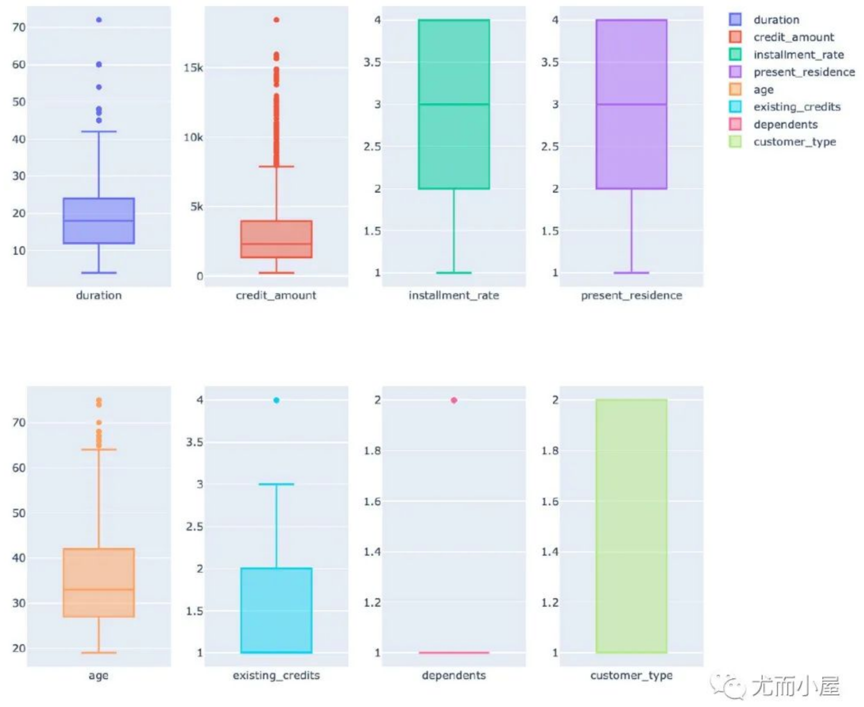
```
string_columns = df.select_dtypes(include="object").columns

# 两个基本参数：设置行、列
fig = make_subplots(rows=3, cols=5)

for i, v in enumerate(string_columns):
    r = i // 5 + 1
    c = (i+1) % 5

    data = df[v].value_counts().reset_index()

    if c ==0:
        fig.add_trace(go.Bar(x=data["index"],y=data[v],
                             text=data[v],name=v),
                      row=r, col=5)
    else:
        fig.add_trace(go.Bar(x=data["index"],y=data[v],
                             text=data[v],name=v),
                     row=r, col=c)

fig.update_layout(width=1000, height=900)

fig.show()
```



2、针对数值型字段的分布情况：

```
number_columns = df.select_dtypes(exclude="object").columns.tolist()
number_columns

# 两个基本参数：设置行、列
fig = make_subplots(rows=2, cols=4)  # 2行4列

for i, v in enumerate(number_columns):  # number_columns 长度是8
    r = i // 4 + 1
    c = (i+1) % 4
```

```
    if c ==0:
        fig.add_trace(go.Box(y=df[v].tolist(),name=v),
                 row=r, col=4)
    else:
        fig.add_trace(go.Box(y=df[v].tolist(),name=v),
                 row=r, col=c)

fig.update_layout(width=1000, height=900)

fig.show()
```



字段处理

支票状态-checking_account_status

中文含义：现有支票账户的状态

- A11：<0 DM
- A12：0 <= x <200 DM
- A13：> = 200 DM /至少一年的薪水分配
- A14：无支票账户）

In [11]:

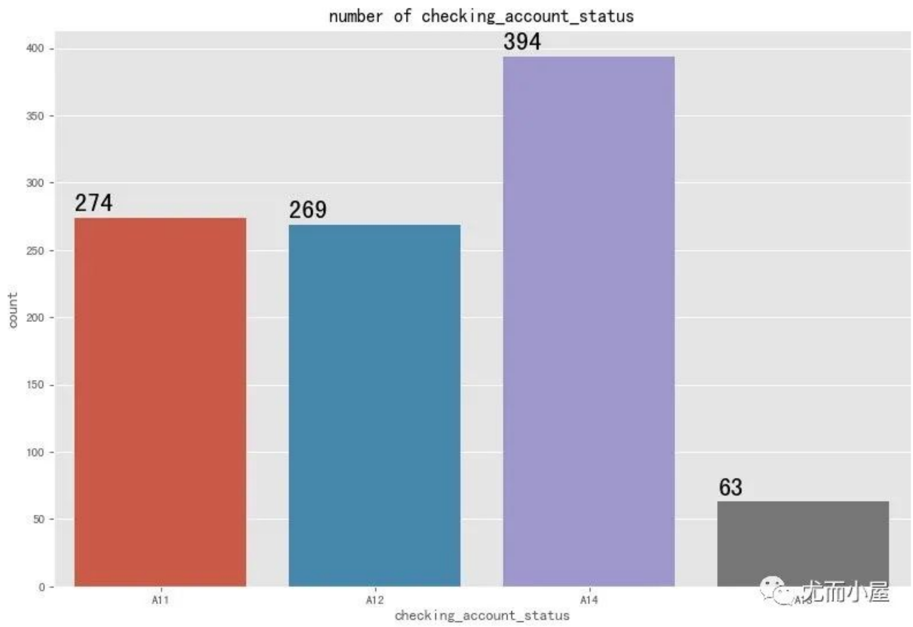
```
df["checking_account_status"].value_counts()
```

Out[11]:

```
A14    394
A11    274
A12    269
A13     63
Name: checking_account_status, dtype: int64
```

In [12]:

```
fig,ax = plt.subplots(figsize=(12,8), dpi=80)

sns.countplot(x="checking_account_status", data=df)

plt.title("number of checking_account_status")

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x(), p.get_height()+5), color='b
plt.show()
```



在这里我们根据每个人的支票账户金额的大小进行**硬编码**：

In [13]:

```
# A11: <0 DM, A12: 0 <= x <200 DM, A13: > = 200 DM /至少一年的薪水分配, A14: 无
# 编码1
cas = {"A11": 1,"A12":2, "A13":3, "A14":0}
df["checking_account_status"] = df["checking_account_status"].map(cas)
```

借款周期-duration

中文含义是：持续时间(月)

In [14]:

```
duration = df["duration"].value_counts()
duration.head()
```

Out[14]:

```
24    184
12    179
18    113
36     83
6      75
Name: duration, dtype: int64
```

In [15]:

```
fig = px.violin(df,y="duration")

fig.show()
```

信用卡历史-credit_history

中文含义

- A30：未提取任何信用/已全额偿还所有信用额
- A31：已偿还该银行的所有信用额
- A32：已到期已偿还的现有信用额
- A33：过去的还款延迟
- A34：关键帐户/其他信用额现有（不在此银行）

In [17]:

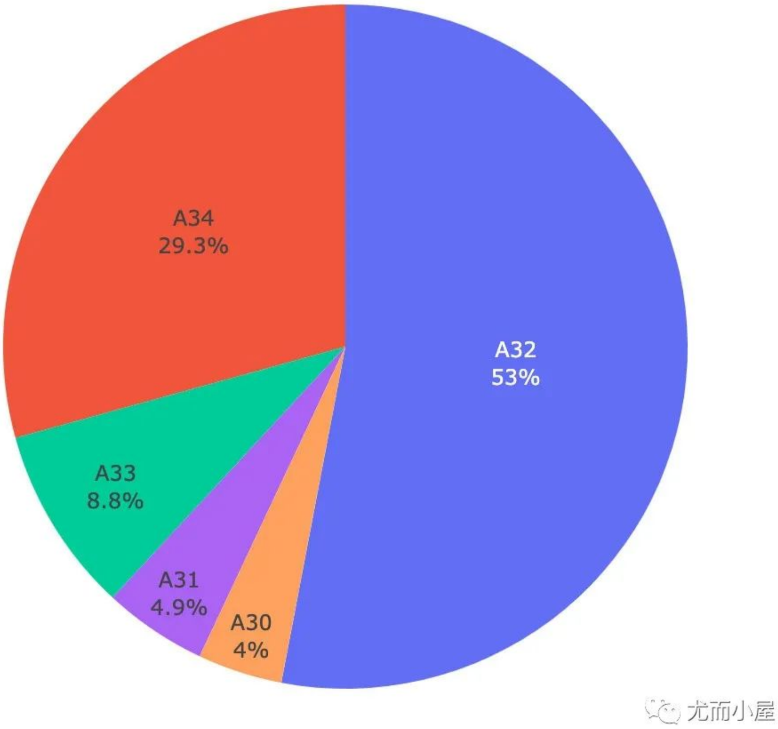
```
ch = df["credit_history"].value_counts().reset_index()
ch
```

Out[17]:

	index	credit_history
0	A32	530
1	A34	293
2	A33	88
3	A31	49
4	A30	40

In [18]:

```
fig = px.pie(ch,names="index",values="credit_history")

fig.update_traces(
    textposition='inside',
    textinfo='percent+label'
)

fig.show()
```




```
# 编码2：独热码

df_credit_history = pd.get_dummies(df["credit_history"])
df = df.join(df_credit_history)
df.drop("credit_history", inplace=True, axis=1)
```

借款目的-purpose

借款目的

In [20]:

```
# 统计每个目的下的人数，根据人数的多少来实施硬编码
purpose = df["purpose"].value_counts().sort_values(ascending=True).reset_index

purpose.columns = ["purpose", "number"]

purpose
```

Out[20]:

	purpose	number
0	A48	9
1	A44	12
2	A410	12
3	A45	22
4	A46	50

5	A49	97
6	A41	103
7	A42	181
8	A40	234
9	A43	280

尤而小屋

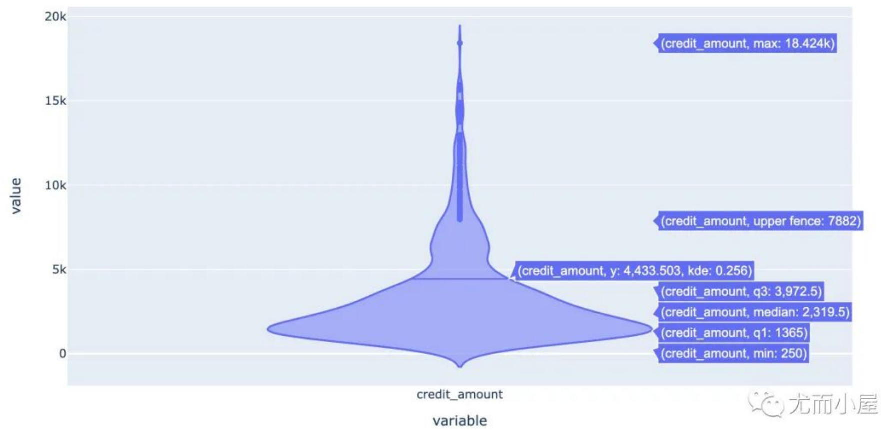
```
# 编码3
df["purpose"] = df["purpose"].map(dict(zip(purpose.purpose,purpose.index)))
```

信用额度-credit_amount

表示的是信用额度

In [22]:

```
px.violin(df["credit_amount"])
```



账户储蓄-savings

账户/债券储蓄（A61：<100 DM，A62：100 <= x <500 DM，A63：500 <= x <1000 DM，A64：>= 1000 DM，A65：未知/无储蓄账户

In [24]:

```
string_columns
```

Out[24]:

```
Index(['checking_account_status', 'credit_history', 'purpose', 'savings',
       'present_employment', 'personal', 'other_debtors', 'property',
       'other_installment_plans', 'housing', 'job', 'telephone',
       'foreign_worker'],
      dtype='object')
```

In [25]:

```
df["savings"].value_counts()
```

Out[25]:

```
A61    603
A65    183
A62    103
A63     63
A64     48
Name: savings, dtype: int64
```

In [26]:

```
# 编码6：硬编码
savings = {"A61":1,"A62":2, "A63":3, "A64":4,"A65":0}

df["savings"] = df["savings"].map(savings)
```

目前状态-present_employment

- A71：待业
- A72：<1年
- A73：1 <= x <4年
- A74：4 <= x <7年
- A75：..> = 7年

In [28]:

```
df["present_employment"].value_counts()
```

Out[28]:

```
A73    339
A75    253
A74    174
A72    172
A71     62
Name: present_employment, dtype: int64
```

In [29]:

```
# 编码7：独热码

df_present_employment = pd.get_dummies(df["present_employment"])
```

In [30]:

```
df = df.join(df_present_employment)

df.drop("present_employment", inplace=True, axis=1)
```

个人婚姻状态和性别-personal

个人婚姻状况和性别（A91：男性：离婚/分居，A92：女性：离婚/分居/已婚，A93：男性：单身，A94：男性：已婚/丧偶，A95：女性：单身）

In [31]:

```
# 编码8：独热码

df_personal = pd.get_dummies(df["personal"])
df = df.join(df_personal)

df.drop("personal", inplace=True, axis=1)
```

其他担保人-other_debtors

A101：无，A102：共同申请人，A103：担保人

In [32]:

```
# 编码9：独热码

df_other_debtors = pd.get_dummies(df["other_debtors"])
df = df.join(df_other_debtors)

df.drop("other_debtors", inplace=True, axis=1)
```

资产-property

In [33]:

```
# 编码10：独热码

df_property = pd.get_dummies(df["property"])
df = df.join(df_property)

df.drop("property", inplace=True, axis=1)
```

住宿-housing

A151:租房，A152:自有，A153:免费

In [34]:

```
# 编码11：独热码

df_housing = pd.get_dummies(df["housing"])
df = df.join(df_housing)

df.drop("housing", inplace=True, axis=1)
```

其他投资计划-other_installment_plans

A141：银行，A142：店铺，A143：无

In [35]:

```
fig,ax = plt.subplots(figsize=(12,8), dpi=80)

sns.countplot(x="other_installment_plans", data=df)

plt.title("number of other_installment_plans")

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x(), p.get_height()+5), color='b
plt.show()
```

```
# 编码12: 独热码

df_other_installment_plans = pd.get_dummies(df["other_installment_plans"])
df = df.join(df_other_installment_plans)

df.drop("other_installment_plans", inplace=True, axis=1)
```

工作-job

- A171：非技术人员-非居民
- A172:非技术人员-居民
- A173:技术人员/官员
- A174:管理/个体经营/高度合格的员工/官员

In [37]:

```
fig,ax = plt.subplots(figsize=(12,8), dpi=80)

sns.countplot(x="job", data=df)

plt.title("number of job")

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x(), p.get_height()+5), color='b
plt.show()
```

```
# 编码13：独热码

df_job = pd.get_dummies(df["job"])
df = df.join(df_job)

df.drop("job", inplace=True, axis=1)
```

电话-telephone

A191:无，A192:有，登记在客户名下

In [39]:

```
# 编码14：独热码

df_telephone = pd.get_dummies(df["telephone"])
df = df.join(df_telephone)

df.drop("telephone", inplace=True, axis=1)
```

是否国外工作-foreign_worker

A201: 有，A202: 无

In [40]:

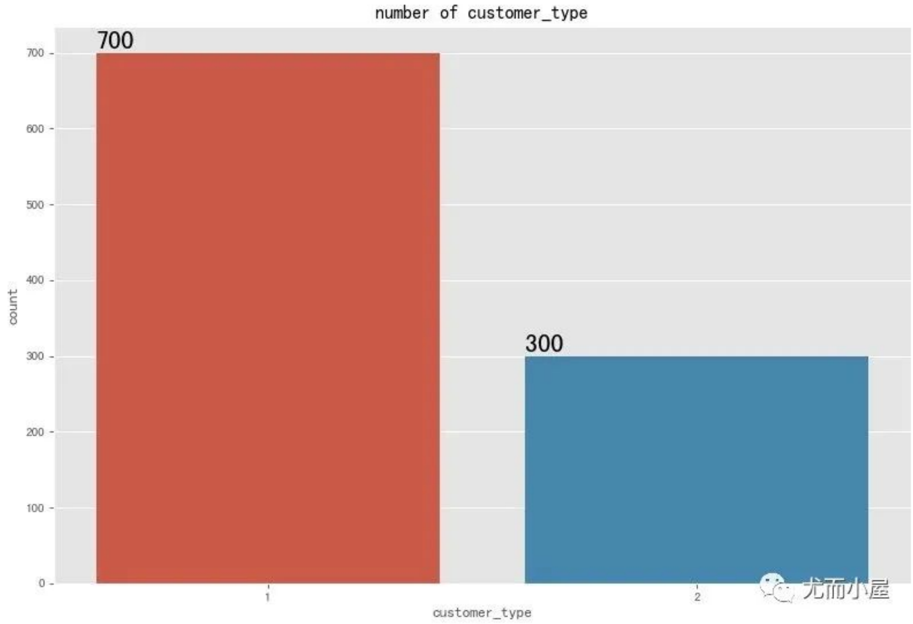
```
# 编码15：独热码

df_foreign_worker = pd.get_dummies(df["foreign_worker"])
df = df.join(df_foreign_worker)

df.drop("foreign_worker", inplace=True, axis=1)
```

两种类型顾客统计-customer_type

预测类别：1 =良好，2 =不良

In [41]:

```
fig,ax = plt.subplots(figsize=(12,8), dpi=80)

sns.countplot(x="customer_type", data=df)

plt.title("number of customer_type")

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x(), p.get_height()+5), color='b
plt.show()
```



打乱数据shuffle

In [42]:

```
from sklearn.utils import shuffle

# 随机打乱数据
df = shuffle(df).reset_index(drop=True)
```

建模

数据分割

In [44]:

```
# 选取特征
X  = df.drop("customer_type",axis=1)

# 目标变量
y = df['customer_type']
from sklearn.model_selection import train_test_split
```

In [45]:

```
# 2-8比例
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.2, rand
```

数据标准化

In [46]:

```
ss = StandardScaler()

X_train = ss.fit_transform(X_train)
```

In [47]:

```
y_train
```

Out[47]:

```
556    1
957    1
577    2
795    2
85     1
      ..
106    1
270    2
860    1
435    1
102    2
Name: customer_type, Length: 200, dtype: int64
```

In [48]:

```
# 分别求出训练集的均值和标准差

mean_ = ss.mean_  # 均值
var_ = np.sqrt(ss.var_)  # 标准差
```

将上面求得的均值和标准差用于测试集中：

In [50]:

```
# 归一化之后的测试集中的特征数据
X_test = (X_test - mean_) / var_
```

模型1：决策树

In [51]:

```
dt = DecisionTreeClassifier(max_depth=5)

dt.fit(X_train, y_train)
```

Out[51]:

```
DecisionTreeClassifier(max_depth=5)
```

In [52]:

```
# 预测
y_pred = dt.predict(X_test)
y_pred[:5]
```

Out[52]:

```
array([2, 1, 1, 2, 1])
```

In [53]:

```
# 混淆矩阵
confusion_mat = metrics.confusion_matrix(y_test,y_pred)
confusion_mat
```

Out[53]:

```
array([[450, 118],
       [137,  95]])
```

In [54]:

```
# 混淆矩阵可视化

classes = ["良好","不良"]

disp = ConfusionMatrixDisplay(confusion_matrix=confusion_mat, display_labels=c
disp.plot(
    include_values=True,            # 混淆矩阵每个单元格上显示具体数值
    cmap="GnBu",                 # matplotlib识别的颜色图
    ax=None,
    xticks_rotation="horizontal",
    values_format="d"
)

plt.show()
```

```
## auc-roc

auc_roc = metrics.roc_auc_score(y_test, y_pred)  # 测试值和预测值
auc_roc

0.5008681398737251
```

模型2：随机森林

In [56]:

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[56]:

```
RandomForestClassifier()
```

In [57]:

```
# 预测
y_pred = rf.predict(X_test)
y_pred[:5]
```

Out[57]:

```
array([1, 1, 1, 2, 1])
```

In [58]:

```
# 混淆矩阵
confusion_mat = metrics.confusion_matrix(y_test,y_pred)
confusion_mat
```

Out[58]:

```
array([[476,  92],
       [142,  90]])
```

In [59]:

```
# 混淆矩阵可视化

classes = ["良好","不良"]

disp = ConfusionMatrixDisplay(confusion_matrix=confusion_mat, display_labels=c
disp.plot(
    include_values=True,            # 混淆矩阵每个单元格上显示具体数值
    cmap="GnBu",                  # matplotlib识别的颜色图
    ax=None,
    xticks_rotation="horizontal",
    values_format="d"
)

plt.show()
```

```
## auc-roc

auc_roc = metrics.roc_auc_score(y_test, y_pred)  # 真实值和预测值
auc_roc

0.6129796017484215
```

模型3：XGboost

In [62]:

```
from xgboost.sklearn import XGBClassifier
## 定义 XGBoost模型
clf = XGBClassifier()

# X_train = X_train.values
# X_test = X_test.values
```

In [63]:

```
clf.fit(X_train, y_train)
```

Out[63]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [65]:

```
# 先转成数组再传进来
X_test = X_test.values

y_pred = clf.predict(X_test)
y_pred[:5]
```

Out[65]:

```
array([1, 1, 1, 2, 1])
```

In [66]:

```
# 混淆矩阵
confusion_mat = metrics.confusion_matrix(y_test,y_pred)
confusion_mat
```

Out[66]:

```
array([[445, 123],
       [115, 117]])
```

In [67]:

```
# 混淆矩阵可视化

classes = ["良好","不良"]

disp = ConfusionMatrixDisplay(confusion_matrix=confusion_mat, display_labels=c
disp.plot(
    include_values=True,            # 混淆矩阵每个单元格上显示具体数值
    cmap="GnBu",                    # matplotlib识别的颜色图
    ax=None,
    xticks_rotation="horizontal",
    values_format="d"
)

plt.show()
```

```
## auc-roc

auc_roc = metrics.roc_auc_score(y_test, y_pred)  # 真实值和预测值
auc_roc

0.6438805245264692
```

模型优化

基于相关系数进行特征筛选

```
# y: customer_type是目标变量

# 1、计算每个特征和目标变量的相关系数

data = pd.concat([X,y],axis=1)
```

```
corr = data.corr()
corr[:5]
```

Out[79]:

	checking_account_status	duration	purpose	credit_amount	savings	installment_rate	present_residence	age
checking_account_status	1.000000	0.035050	-0.012343	0.024561	-0.005614	-0.057942	-0.059555	-0.049058
duration	0.035050	1.000000	-0.103243	0.624984	-0.064526	0.074749	0.034067	-0.036136
purpose	-0.012343	-0.103243	1.000000	-0.154744	0.009980	0.038333	-0.070572	-0.052361
credit_amount	0.024561	0.624984	-0.154744	1.000000	-0.107538	-0.271316	0.028926	0.032716
savings	-0.005614	-0.064526	0.009980	-0.107538	1.000000	-0.000805	-0.011772	-0.017997

5 rows × 46 columns

相关系数的描述统计信息：发现整体的相关系数（绝对值）都比较小

```
In [85]: corr.describe()
```

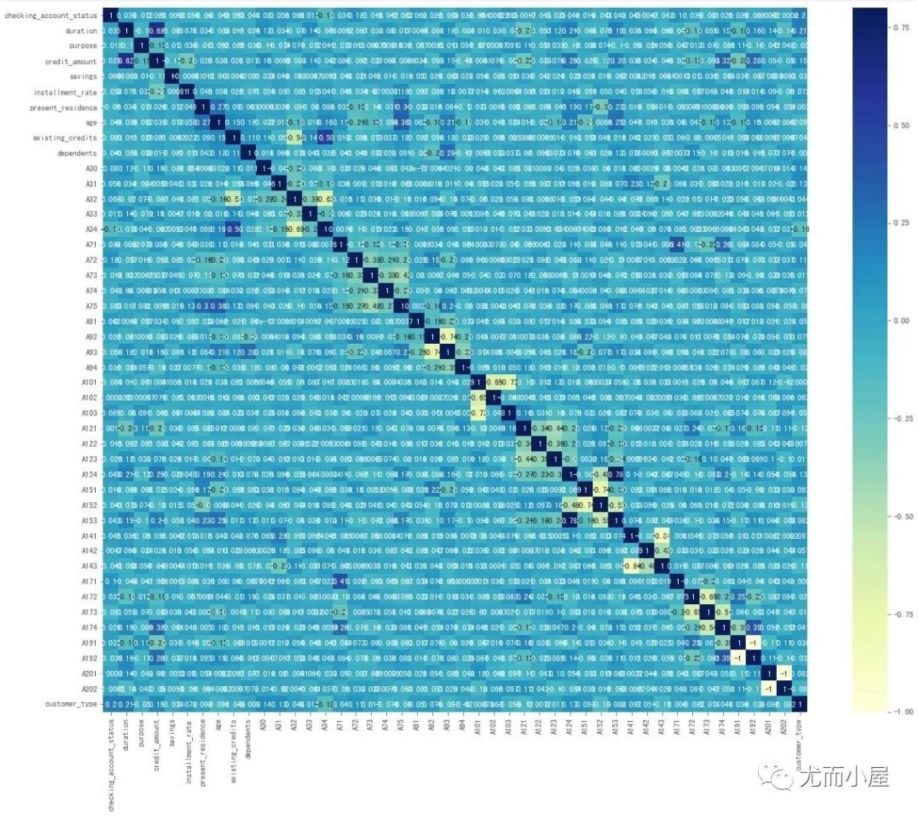
Out[85]:

	checking_account_status	duration	purpose	credit_amount	savings	installment_rate	present_residence	age	existing_credits
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000	46.000000
mean	0.026628	0.042342	0.005501	0.036310	0.015647	0.016980	0.030662	0.033603	0.028940
std	0.159009	0.199725	0.163105	0.214657	0.152528	0.163466	0.180510	0.194635	0.193339
min	-0.143082	-0.242586	-0.154744	-0.276995	-0.107538	-0.271316	-0.297547	-0.212620	-0.540354
25%	-0.041605	-0.064498	-0.067140	-0.061803	-0.024820	-0.045704	-0.040715	-0.051535	-0.028632
50%	0.003188	0.008204	-0.011866	0.011533	-0.004197	-0.000759	0.003960	0.006352	0.003549
75%	0.042369	0.108466	0.037107	0.078227	0.021749	0.042759	0.047638	0.098529	0.041989
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 46 columns

热力图

```
ax = plt.subplots(figsize=(20,16))

ax = sns.heatmap(corr,
                 vmax=0.8,
                 square=True,
                 annot=True,  # 显示数据
                 cmap="YlGnBu")
```



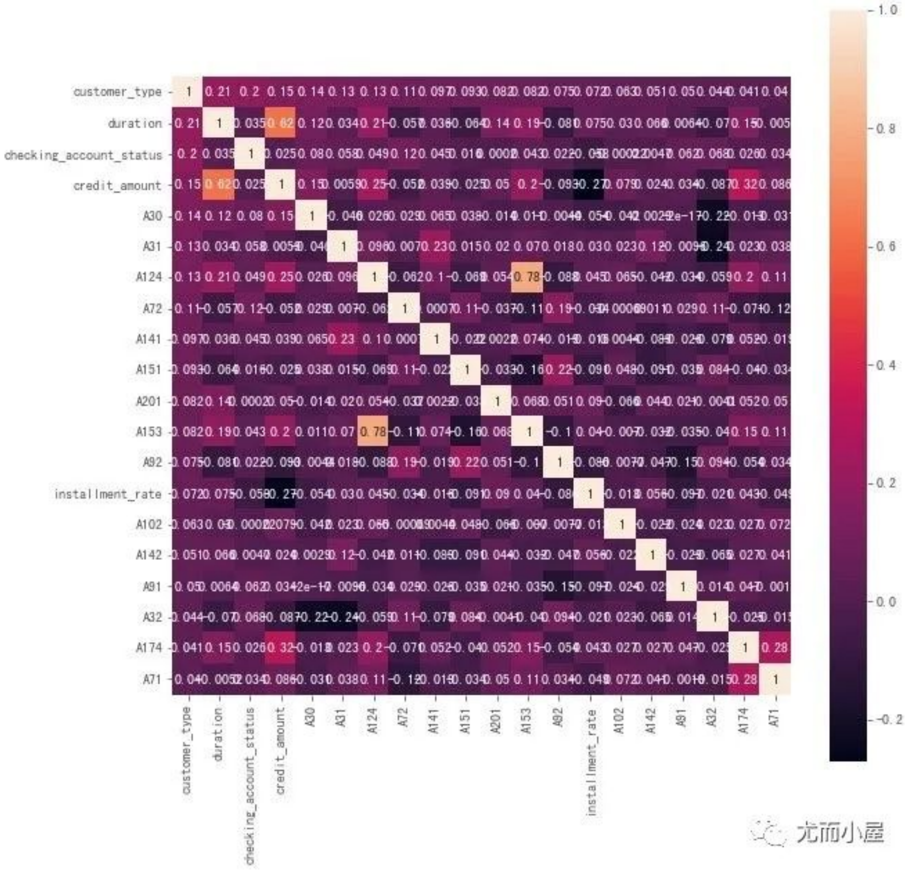
根据相关系数筛选前20个变量

```
k = 20
```

```
cols = corr.nlargest(k,"customer_type")["customer_type"].index
cols
```

```
Index(['customer_type', 'duration', 'checking_account_status', 'credit_amount'
       'A30', 'A31', 'A124', 'A72', 'A141', 'A151', 'A201', 'A153', 'A92',
       'installment_rate', 'A102', 'A142', 'A91', 'A32', 'A174', 'A71'],
      dtype='object')
```

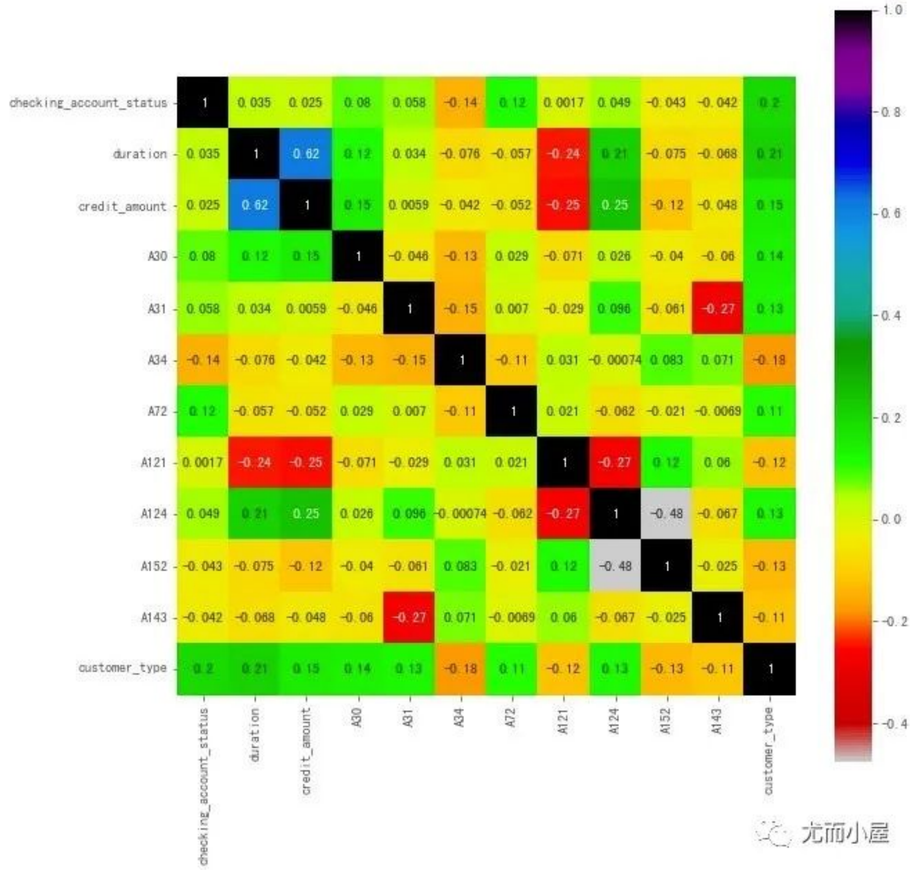
```
cm = np.corrcoef(data[cols].values.T)

hm = plt.subplots(figsize=(10,10))  # 调整画布大小
hm = sns.heatmap(data[cols].corr(),  # 前10个属性的相关系数
                 annot=True,
                 square=True)
plt.show()
```



筛选相关系数绝对值大于0.1的变量

```
threshold = 0.1

corrmat = data.corr()
top_corr_features = corrmat.index[abs(corrmat["customer_type"]) > threshold]

plt.figure(figsize=(10,10))

g = sns.heatmap(data[top_corr_features].corr(),  # 大于0.5的特征构成的DF的相关
                annot=True,
                square=True,
```

```
              cmap="nipy_spectral_r"
             )
```



新数据建模

```
# 筛选出为True的特征
useful_col = corrmat.index[abs(corrmat["customer_type"]) > threshold].tolist()
```

```
new_df = df[useful_col]
new_df.head()
```

Out[73]:

	checking_account_status	duration	credit_amount	A30	A31	A34	A72	A121	A124	A152	A143	customer_type
0	3	10	1275	0	0	0	1	0	0	1	1	1
1	1	24	4870	0	0	0	0	0	1	0	1	2
2	2	48	9960	0	0	0	1	0	0	1	1	2
3	0	4	1503	0	0	1	0	1	0	1	1	1
4	2	12	625	0	0	0	1	1	0	1	[illegible]	[illegible]

尤而小屋

数据切分

```
# 选取特征
X  = new_df.drop("customer_type",axis=1)

# 目标变量
y = new_df['customer_type']
```

```
# 3-7比例
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.3, rand
```

标准化

```
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
```

```
# 分别求出训练集的均值和标准差

mean_ = ss.mean_  # 均值
var_ = np.sqrt(ss.var_)  # 标准差

# 归一化之后的测试集中的特征数据

X_test = (X_test - mean_) / var_
```

建模

```
from xgboost.sklearn import XGBClassifier
## 定义 XGBoost模型
clf = XGBClassifier()
```

```
clf.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [80]:

```
# 先转成数组再传进来
X_test = X_test.values

y_pred = clf.predict(X_test)
y_pred[:5]
```

Out[80]:

```
array([2, 1, 2, 2, 1])
```

In [81]:

```
# 混淆矩阵
confusion_mat = metrics.confusion_matrix(y_test,y_pred)
confusion_mat
```

Out[81]:

```
array([[406,  94],
       [ 96, 104]])
```

In [82]:

```
## auc-roc

auc_roc = metrics.roc_auc_score(y_test, y_pred)  # 真实值和预测值
auc_roc
```

Out[82]:

```
0.666
```

优化方向

经过3种不同树模型的建模，我们发现模型的AUC值并不是很高。AUC 值是一个概率值，AUC 值越大，分类算法越好。可以考虑优化的方向：

1. 特征工程处理：**这个可以重点优化**。目前对原始的特征变量使用了3种不同类型编码、独热码和硬编码；有些字段的编码方式需要优化。
2. 筛选变量：相关系数是用来检测两个**连续型变量**之间**线性相关**的程度；特征变量和最终因变量的关系不一定线性相关。本文中观察到相关系数都很低，似乎佐证了这点。后续考虑结合其他方法来筛选变量进行建模。
3. 模型调优：通过**网格搜索**等优化单个模型的参数，或者通过**模型融合**来增强整体效果。

数据集获取

关注公众号【尤而小屋】，回复信贷即可领取本文数据集。

推荐阅读

通透理解Python函数传参机制！

生日快乐：尤而小屋三周岁啦！

Seaborn绘制11个柱状图

【Pandas基础】在线文件和剪贴板数据读取

粉丝的Pandas题：一题三解

5K+，Pandas读取TXT文件

> 尤而小屋，一个温馨的小屋。小屋主人，一手代码谋求生存，一手掌勺享受生活，欢迎你的光临

尤而小屋

尤而小屋，一个温馨且有爱的小屋🏠 小屋主人，一手代码谋求生存，一手掌勺享受生...

261篇原创内容

公众号

收录于合集 #kaggle 23

下一篇 · Kaggle首战金牌总结！

喜欢此内容的人还喜欢

【HackerOne】HTTP参数污染+真实案例

韬光养晦安全

【Pandas基础】DataFrame差集/交集/并集求解

尤而小屋

反后门学习：在有毒数据上训练干净的模型

隐者联盟