全国高校大数据应用创新大赛

【大数据技术技能赛】

【企业命题赛】

算法说明书

团队名称：\_\_\_\_\_\_\_\_\_\_\_\_城建追梦队\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学校学院：\_\_\_\_\_\_\_\_\_\_\_\_河南城建学院\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

队长姓名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_刘磊\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

指导老师：\_\_\_\_\_\_\_\_\_\_\_\_\_薛冰、董国忠\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

填报日期：\_\_\_\_\_\_\_\_\_\_\_2018年6月24日\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 开发环境介绍

本项目基于Anaconda平台使用Python语言进行开发。

平台版本：Anaconda2-4.1.1

Python版本：Python 2.7.12

涉及的组件：Jupyter Notebook

涉及的模块：numpy、pandas、catboost、sklearn

平台组件介绍：

Jupyter Notebook是一种 Web 应用，能让用户将说明文本、数学方程、代码和可视化内容全部组合到一个易于共享的文档中。Notebook 已迅速成为处理数据的必备工具。其已知用途包括数据清理和探索、可视化、机器学习和大数据分析等。

Python的搭建配置过程：

使用 Anaconda安装 JupyterNoteBook，Python 是 Anaconda 附带的 Python，具体的安装过程则是按照提示一步步进行就可以了。

## 算法介绍

（通过文字和流程图介绍程序算法的开发思路和运算逻辑，详细描述程序中所涉及函数的功能及调用逻辑。）

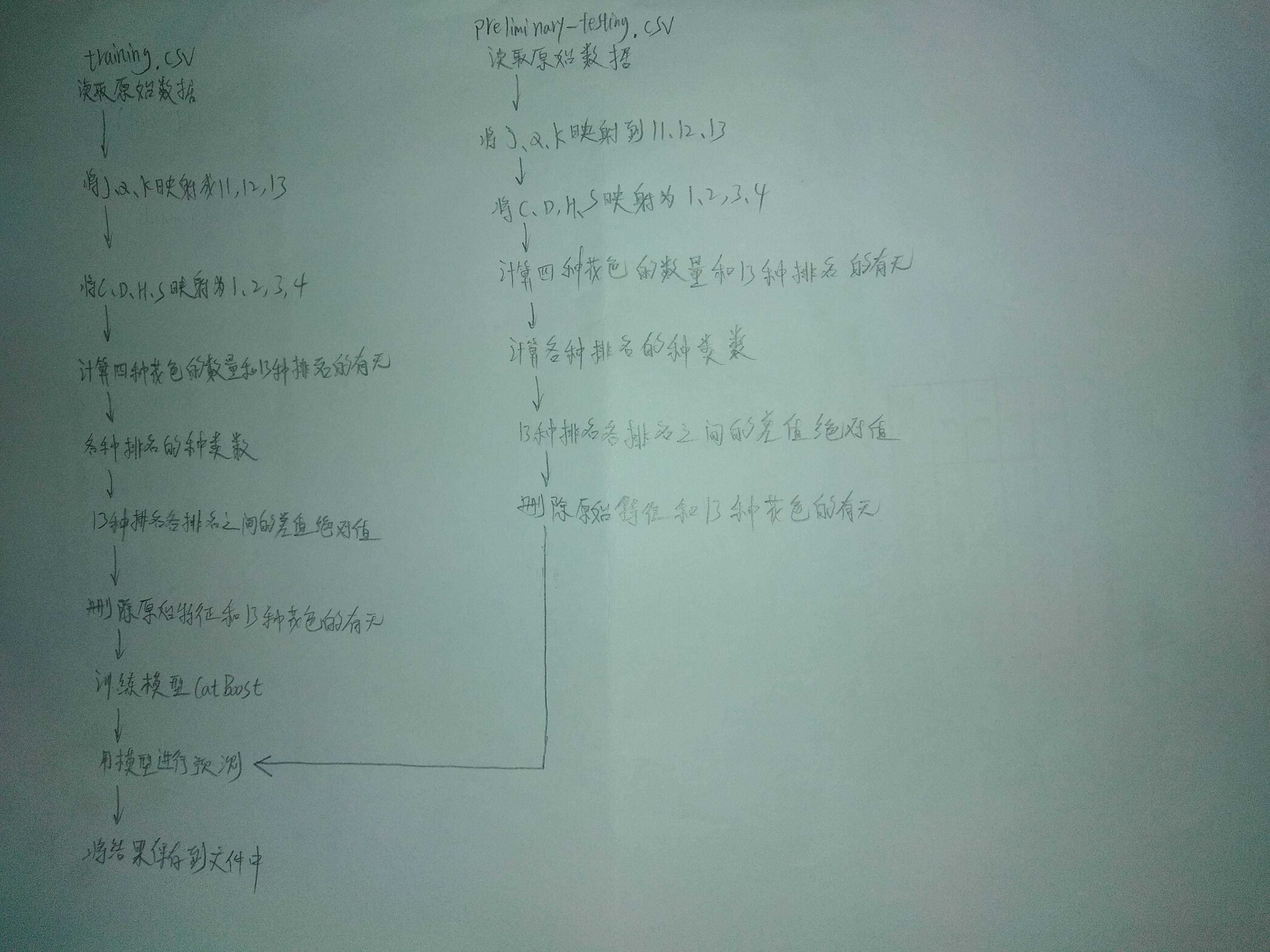
这次项目主要用的是CatBoost，CatBoost是来自Yandex的一种最近开源的机器学习算法。它可以很容易地与深入学习框架集成，比如谷歌的TensorFlow和苹果的核心ML。它可以与不同的数据类型一起工作，以帮助解决当今企业面临的广泛问题。为了使它更好，它提供了最好的级别中的准确性。

它在两方面特别强大：它可以在没有其他机器学习方法通常需要的大量数据培训的情况下产生最先进的结果，并为伴随许多业务问题而来的描述性更强的数据格式提供强大的开箱即用支持。

“CatBoost”这个名字来源于两个词“Category”和“Boosting”。如前所述，该库可以很好地处理多种类型的数据，例如音频、文本、图像(包括历史数据)。

“Boost”来自梯度增强机器学习算法，因为该库是基于梯度增强库的。梯度增强算法是一种功能强大的机器学习算法，广泛应用于欺诈检测、推荐项、预测等多种业务挑战中，并取得了良好的效果。它还可以返回非常好的结果，相对较少的数据，不像DL模型，需要从大量的数据学习。

程序算法的开发思路和运算逻辑：



涉及函数：

transform\_jkq(x):

用于将数据中J、Q、K映射成11、12、13。

map(encode\_map):

用于将将数据中C、D、H、S 映射为1、2、3、4。

bincount2D\_vectorized(a):

作用是搜索每一行中的元素数量，并根据矩阵值对它们进行编码。

用于计算四种花色的数量、13种排名的有无、各种排名的种类数。

np.abs(training['rank1'] - training['rank13'])：

用于计算13种排名各排名之间的差值的绝对值

drop(['S1','C1','S2','C2','S3','C3','S4','C4','S5','C5'], axis=1)：

删除原始特征

drop(['rank{}'.format(n) for n in range(1,14,1)], axis=1)：

删除13种花色的有无的特征

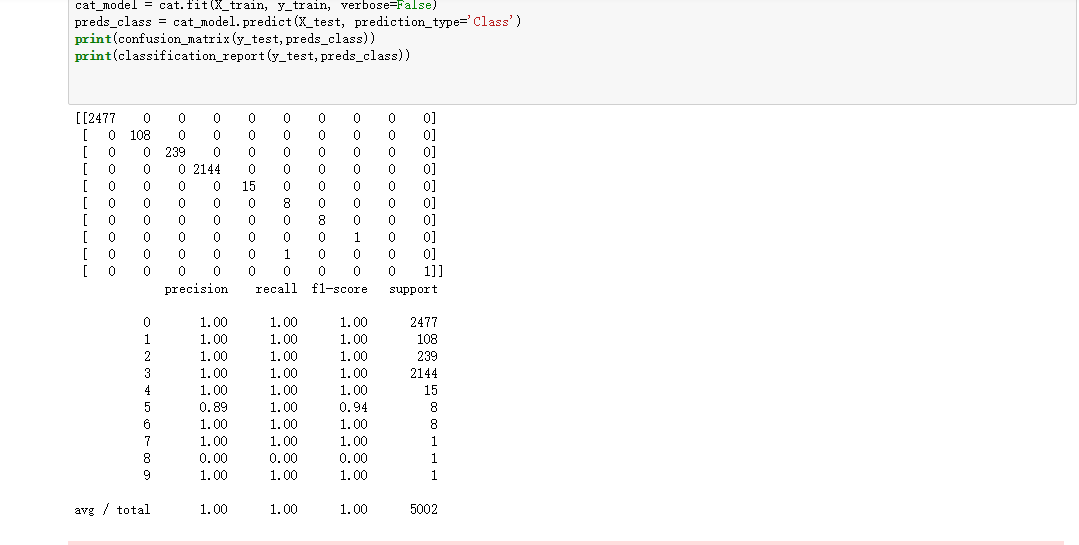
## 源代码展示

*# -\*- coding: utf-8 -\*-***import** numpy **as** np  
**import** pandas **as** pd  
**import** catboost **as** cb  
  
  
*# 原始数据路径*training\_path = **'D://training.csv'**source\_data\_path = **'D://preliminary-testing.csv'***# --------读取原始数据-------  
# training数据*training = pd.read\_csv(training\_path, names=[**'S1'**, **'C1'**, **'S2'**, **'C2'**, **'S3'**, **'C3'**, **'S4'**, **'C4'**, **'S5'**, **'C5'**, **'hand'**])  
*# preliminary-testing数据*source\_data = pd.read\_csv(source\_data\_path, names=[**'S1'**, **'C1'**, **'S2'**, **'C2'**, **'S3'**, **'C3'**, **'S4'**, **'C4'**, **'S5'**, **'C5'**])  
  
  
  
*# --------将J、Q、K映射成11、12、13------***def** transform\_jkq(x):  
 **if** x == **'J'**:  
 **return** 11  
 **elif** x == **'Q'**:  
 **return** 12  
 **elif** x == **'K'**:  
 **return** 13  
 **else**:  
 **return** x  
   
*# training数据处理*training[**'C1'**] = training[**'C1'**].apply(transform\_jkq)  
training[**'C2'**] = training[**'C2'**].apply(transform\_jkq)  
training[**'C3'**] = training[**'C3'**].apply(transform\_jkq)  
training[**'C4'**] = training[**'C4'**].apply(transform\_jkq)  
training[**'C5'**] = training[**'C5'**].apply(transform\_jkq)  
  
*# preliminary-testing数据处理*source\_data[**'C1'**] = source\_data[**'C1'**].apply(transform\_jkq)  
source\_data[**'C2'**] = source\_data[**'C2'**].apply(transform\_jkq)  
source\_data[**'C3'**] = source\_data[**'C3'**].apply(transform\_jkq)  
source\_data[**'C4'**] = source\_data[**'C4'**].apply(transform\_jkq)  
source\_data[**'C5'**] = source\_data[**'C5'**].apply(transform\_jkq)  
  
  
*# -------将C、D、H、S 映射为1、2、3、4--------*encode\_map = {**'C'**:1, **'D'**:2, **'H'**:3,**'S'**:4}  
*# training数据处理*training[**'S1'**] = training[**'S1'**].map(encode\_map)  
training[**'S2'**] = training[**'S2'**].map(encode\_map)  
training[**'S3'**] = training[**'S3'**].map(encode\_map)  
training[**'S4'**] = training[**'S4'**].map(encode\_map)  
training[**'S5'**] = training[**'S5'**].map(encode\_map)  
  
*# preliminary-testing数据处理*source\_data[**'S1'**] = source\_data[**'S1'**].map(encode\_map)  
source\_data[**'S2'**] = source\_data[**'S2'**].map(encode\_map)  
source\_data[**'S3'**] = source\_data[**'S3'**].map(encode\_map)  
source\_data[**'S4'**] = source\_data[**'S4'**].map(encode\_map)  
source\_data[**'S5'**] = source\_data[**'S5'**].map(encode\_map)  
  
  
*# --------计算四种花色的数量和13种排名的有无---------***def** bincount2D\_vectorized(a):  
 N = a.max()+1  
 a\_offs = a + np.arange(a.shape[0])[:,None]\*N  
 **return** np.bincount(a\_offs.ravel(), minlength=a.shape[0]\*N).reshape(-1,N)  
  
*# training数据处理  
# 计算四种花色的数量*S\_training = training.iloc[:,[0,2,4,6,8]].astype(int)  
S\_training = pd.DataFrame(bincount2D\_vectorized(S\_training.values),columns=[**'suitCount0'**,**'suitCount1'**,**'suitCount2'**,**'suitCount3'**,**'suitCount4'**])  
training = pd.merge(training, S\_training, how=**'left'**, left\_index=True, right\_index=True).drop([**'suitCount0'**],axis=1)  
*#计算13种排名的有无*R\_training = training.iloc[:,np.arange(1,10,2)].astype(int)  
cols = [**'rank{}'**.format(x) **for** x **in** range(0,14,1)]  
R\_training = pd.DataFrame(bincount2D\_vectorized(R\_training.values),columns=cols)  
training = pd.merge(training, R\_training, how=**'left'**, left\_index=True, right\_index=True).drop([**'rank0'**],axis=1)  
  
*# preliminary-testing数据处理  
#计算13种排名的有无*S\_source\_data = source\_data.iloc[:,[0,2,4,6,8]].astype(int)  
S\_source\_data = pd.DataFrame(bincount2D\_vectorized(S\_source\_data.values),columns=[**'suitCount0'**,**'suitCount1'**,**'suitCount2'**,**'suitCount3'**,**'suitCount4'**])  
source\_data = pd.merge(source\_data, S\_source\_data, how=**'left'**, left\_index=True, right\_index=True).drop([**'suitCount0'**],axis=1)  
*#计算13种排名的有无*R\_source\_data = source\_data.iloc[:,np.arange(1,10,2)].astype(int)  
cols = [**'rank{}'**.format(x) **for** x **in** range(0,14,1)]  
R\_source\_data = pd.DataFrame(bincount2D\_vectorized(R\_source\_data.values),columns=cols)  
source\_data = pd.merge(source\_data, R\_source\_data, how=**'left'**, left\_index=True, right\_index=True).drop([**'rank0'**],axis=1)  
  
  
*# ------各种排名的种类数------  
  
# training数据处理*R\_training = training.loc[:,[**'rank{}'**.format(n) **for** n **in** range(1,14,1)]].astype(int)  
R\_training = pd.DataFrame(bincount2D\_vectorized(R\_training.values),columns=[**'rankCount{}'**.format(n) **for** n **in** range(0,5,1)])  
training = pd.merge(training, R\_training, how=**'left'**, left\_index=True, right\_index=True).drop([**'rankCount0'**],axis=1)  
  
*# preliminary-testing数据处理*R\_source\_data = source\_data.loc[:,[**'rank{}'**.format(n) **for** n **in** range(1,14,1)]].astype(int)  
R\_source\_data = pd.DataFrame(bincount2D\_vectorized(R\_source\_data.values),columns=[**'rankCount{}'**.format(n) **for** n **in** range(0,5,1)])  
source\_data = pd.merge(source\_data, R\_source\_data, how=**'left'**, left\_index=True, right\_index=True).drop([**'rankCount0'**],axis=1)  
  
  
*# ------13种排名各排名之间的差值的绝对值-----  
  
# training数据处理*training[**'diff1\_13'**] = np.abs(training[**'rank1'**] - training[**'rank13'**])  
**for** i **in** range(2,14,1):  
 training[**'diff{}\_{}'**.format(i,i-1)] = np.abs(training[**'rank{}'**.format(i)] - training[**'rank{}'**.format(i-1)])  
*# training['diff13\_1'] = np.abs(training['rank13'] - training['rank1'])  
  
# preliminary-testing数据处理*source\_data[**'diff1\_13'**] = np.abs(source\_data[**'rank1'**] - source\_data[**'rank13'**])  
**for** i **in** range(2,14,1):  
 source\_data[**'diff{}\_{}'**.format(i,i-1)] = np.abs(source\_data[**'rank{}'**.format(i)] - source\_data[**'rank{}'**.format(i-1)])  
*# training['diff13\_1'] = np.abs(training['rank13'] - training['rank1'])  
  
  
# ------删除原始特征和13种花色的有无-----  
  
# training数据处理*training = training.drop([**'S1'**,**'C1'**,**'S2'**,**'C2'**,**'S3'**,**'C3'**,**'S4'**,**'C4'**,**'S5'**,**'C5'**], axis=1)  
training = training.drop([**'rank{}'**.format(n) **for** n **in** range(1,14,1)], axis=1)  
  
*# preliminary-testing数据处理*source\_data = source\_data.drop([**'S1'**,**'C1'**,**'S2'**,**'C2'**,**'S3'**,**'C3'**,**'S4'**,**'C4'**,**'S5'**,**'C5'**], axis=1)  
source\_data = source\_data.drop([**'rank{}'**.format(n) **for** n **in** range(1,14,1)], axis=1)  
  
  
*# --------训练模型并用模型预测数据--------*X = training.drop([**'hand'**],axis=1)  
y = training.hand  
  
params = {  
 **'l2\_leaf\_reg'**:0.8,  
 **'learning\_rate'**:0.09,  
 **'depth'**:11,  
 **'iterations'**:250  
 }  
cat = cb.CatBoostClassifier(loss\_function=**'MultiClassOneVsAll'**, random\_seed=1234)  
*# 设置模型参数*cat.set\_params(\*\*params)  
*# 训练模型*cat\_model = cat.fit(X, y, verbose=False)  
*#用模型进行预测*preds\_class = cat\_model.predict(source\_data, prediction\_type=**'Class'**)  
result = pd.DataFrame(preds\_class)  
*# 将结果转化为整型*result\_1 = result[0].apply(int)  
result\_2 = pd.DataFrame(result\_1)  
*# 将数据保存到文件dsjyycxds\_preliminary.txt中*result\_2.to\_csv(**'D://dsjyycxds\_preliminary.txt'**, index=False, header=False)

## 运行结果展示

（1）测试时的结果





（2）预测结果：

