统计图像识别与分类

图像高维数据处理的初步探索

张亦洲/盛焕新/王骋宇/戈栋炜

2023.5.30

一、选题背景

KAGGLE 竞赛项目：STEM 学科统计图像识别分类
https://www.kaggle.com/competitions/benetech-making-graphs-accessible

- 选题动机：便于阅读障碍学生分辨统计图像

比赛目标

数以百万计的学生有学习、身体或视力障碍，无法阅读传统印刷品。 这些学生无法访问科学、技术、工程和数学 (STEM) 领域的大部分教育材料。 存在使书面文字易于访问的技术。 然而，为图表等教育视觉效果这样做仍然很复杂且需要大量资源。 因此，只有一小部分教育材料可供具有这种学习差异的学习者使用——除非机器学习可以帮助弥合这种差距。

本次比赛的目标是提取 STEM 教科书中常见的四种图表所代表的数据。 您将开发一个在图形数据集上训练的自动解决方案。

您的工作将有助于使数百万有学习差异或残疾的学生能够访问阅读图表。

二、问题描述：数据情况

- 图像分类：共 60578 张图
  0: dot 点图 5131
  1: horizontal_bar 横向的柱形图 73
  2: line 线图 (折线曲线) 24842
  3: scatter (散点图) 11243
  4: vertical_bar 竖向的柱形图 19189
- 每张图 128×128 维，每一个像素点都可以用一个值去表示

```
X = pd.DataFrame(X)
X
```

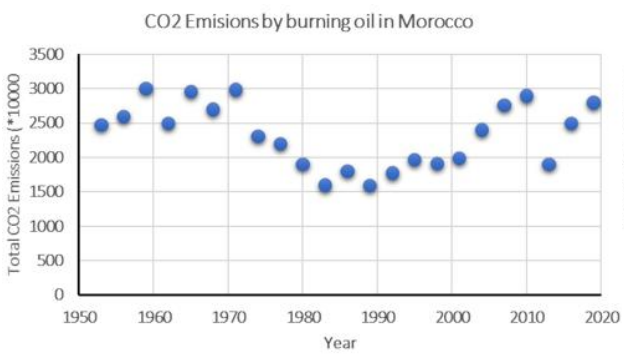
[18]:

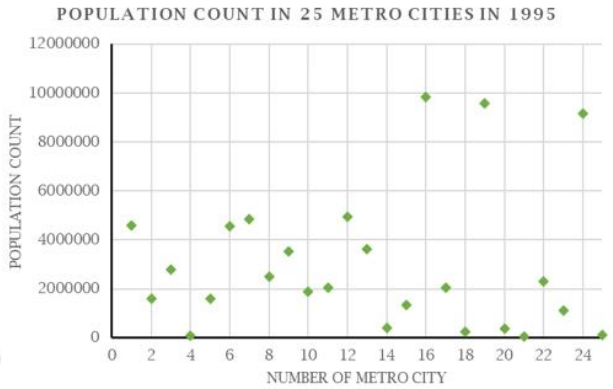
	0	1	2	3	4	5	6	7	8	9	...	16374	16375	16376	16
0	0.073684	0.073684	0.078947	0.078947	0.084211	0.084211	0.084211	0.089474	0.089474	0.094737	...	0.094737	0.094737	0.084211	0.08
1	0.749020	0.749020	0.749020	0.749020	0.749020	0.749020	0.749020	0.749020	0.749020	0.749020	...	0.749020	0.749020	0.749020	0.74
2	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
3	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
4	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
5	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
6	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
7	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00
8	0.981982	0.981982	0.986486	0.986486	0.986486	0.986486	0.986486	0.986486	0.986486	0.986486	...	1.000000	1.000000	1.000000	1.00
9	0.889764	0.889764	0.889764	0.889764	0.889764	0.889764	0.889764	0.889764	0.889764	0.889764	...	1.000000	1.000000	1.000000	1.00

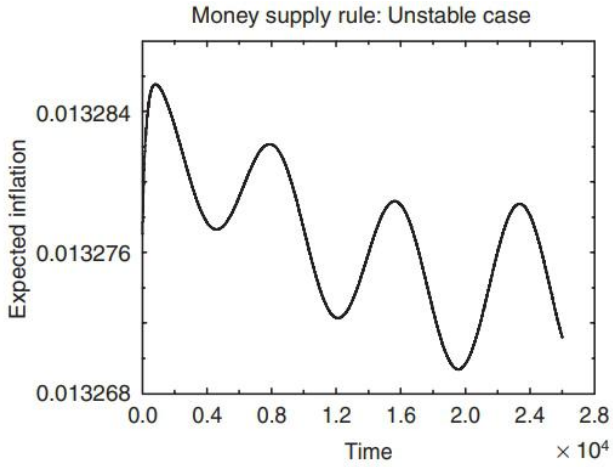
二、问题描述：存在的困难

- 数据庞大：60578 张图 $\rightarrow$ 计算量很大
- 高维特征：128×128 像素点 $\rightarrow 16384$ 个维度
- 非平衡样本：1 类图像少 $\rightarrow$ 模型训练存在偏差 $\rightarrow$ 影响预测和分类
- 同类样本差异：同一类样本差异可能会非常大
- 异类样本相似：不同类样本差异可能会非常小

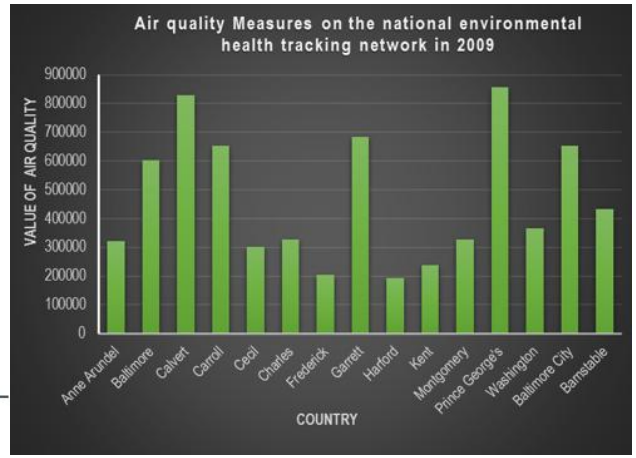
二、问题描述：图像展示


CO2 Emisions by burning oil in Morocco
Total CO2 Emissions (*10000
Year


POPULATION COUNT IN 25 METRO CITIES IN 1995
POPULATION COUNT
NUMBER OF METRO CITY


Money supply rule: Unstable case
Expected inflation
Time
× 10^4

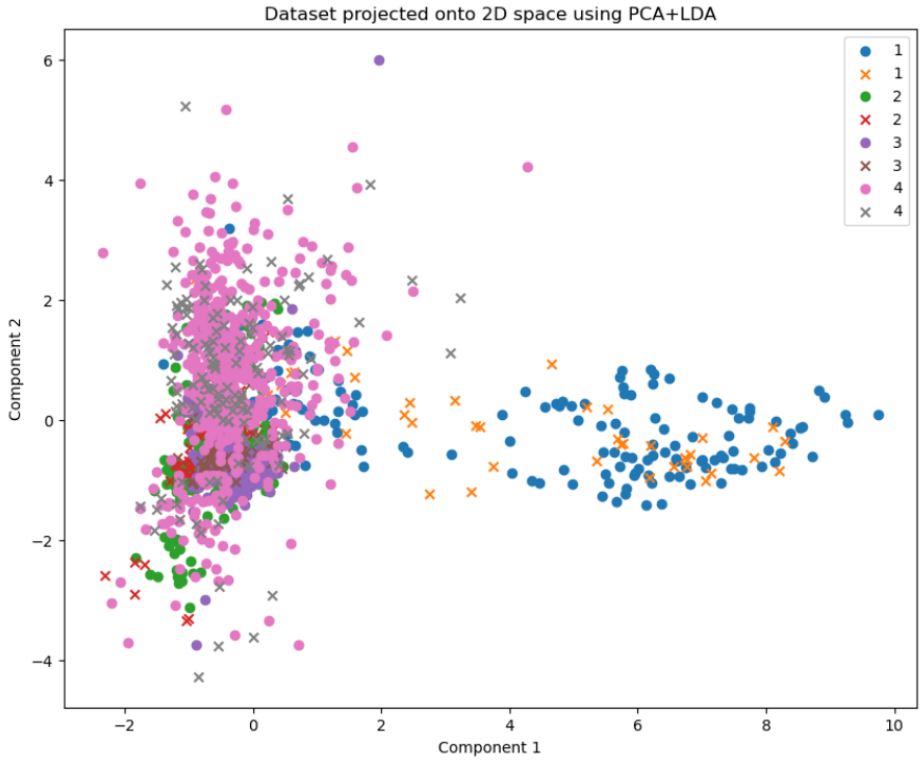

Number of Tourists visited in November 2021
No. of tourists visited (in millions)
Tarikere
Malavalli
Raisen
Lahar
Uravakonda
Savanur
Sirohi
City in India


Air quality Measures on the national environmental health tracking network in 2009
VALUE OF AIR QUALITY
COUNTRY


Stock price of CSA in 2016
Price (USD)
Month

三、初步尝试：PCA+LDA

- PCA 降维：$16384 \to 100$ LDA: 线性可分？
- 数据处理：40000 数据测试，展示 2000 数据划分结果如下



- 效果并不好...

三、初步尝试：CNN Introduction

卷积神经网络（convolutional neural network）是含有卷积层（convolutional layer）的神经网络。我们使用的卷积神经网络均使用最常见的二维卷积层。它有高和宽两个空间维度，常用来处理图像数据。

三、初步尝试：Code Representation

定义卷积层，池化层以及 Dropout 层，输出层：

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=(3, 3))
        self.conv2 = nn.Conv2d(32, 64, kernel_size=(3, 3))
        self.conv3 = nn.Conv2d(64, 128, kernel_size=(3, 3))
        self.pool = nn.MaxPool2d(kernel_size=(2, 2))
        self.dropout = nn.Dropout(p=0.25)
        self.fc1 = nn.Linear(128 * 3 * 3, 512)
        self.fc2 = nn.Linear(512, 5)
```

三、初步尝试：Code Representation

利用定义的各层搭建卷积神经网络：

```
def forward(self, x):
    x = self.conv1(x)
    x = nn.functional.sigmoid(x)
    x = self.pool(x)
    x = self.conv2(x)
    x = nn.functional.sigmoid(x)
    x = self.pool(x)
    x = self.conv3(x)
    x = nn.functional.sigmoid(x)
    x = self.pool(x)
    x = self.dropout(x)
    x = x.reshape(x.size(0), -1)
    x = nn.functional.sigmoid(self.fc1(x))
    x = self.dropout(x)
    x = nn.functional.softmax(self.fc2(x), dim=1)
    return x
```

三、初步尝试：Code Representation

模型训练：

```
# Realise a model, criterion, and optimizer
model = Net()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)


# Train the model
epochs = 30
for epoch in range(epochs):
    running_loss = 0.0
    for data, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(data)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print(f'Epoch {epoch}: Loss = {running_loss/len(train_loader)}')
```

- 也没有达到很理想的效果...

三、初步尝试：机器学习模型

- 机器学习多分类表现亮眼，应用广泛
- 树模型：DTC+RF+LGB+XGB
- 支持向量机模型：SVC

三、初步尝试：Light Gradient Boosting Machine

LGB: LightGBM

- 快速的梯度提升决策树 (GBDT)：特征聚类，互斥捆绑，无损降维
- 高效并行计算，在处理大规模数据和高维数据时表现出色
- 准确性和运行速度都很高！

```
import lightgbm as lgb
model = lgb.LGBMClassifier(learning_rate=0.01)
model.fit(X_train, y_train)
from sklearn.metrics import classification_report
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(y_pred)
```

三、初步尝试：eXtreme Gradient Boosting

XGB: XGBoost

- 不断特征分裂添加树，贪心算法最小化损失函数，近似算法泰勒展开
- 新树相当于新的一个函数去拟合之前预测的残差
- 效果优异，使用简单，竞赛常用，防止过拟合

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(
    objective='multi:softmax',
    num_class=5,
    max_depth=100,
    learning_rate=1,
    n_estimators=1000
)

xgb_model.fit(X_train, y_train)
y_pred = xgb_model.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print(y_pred)
```

三、初步尝试：Support Vector Classifier

SVC：Support Vector Classifier

- 把数据映射到更高维，找到超平面将不同类别变得线性可分
- 深度学习流行之前的主流算法，一定程度上避免过拟合

```
#惩罚系数一两万以上就收敛了
from sklearn .svm import SVC
model=SVC(kernel='rbf',
          C=25000,
          probability=True)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred,zero_division=1))
print(y_pred)
```

三、初步尝试：问题总结

- 以上方法均受非平衡数据影响：第 1 类图的数量太少了！
- PCA 降维可能不够有效：降维中损失了一些信息？

四、进一步探索: 武功秘籍

针对非平衡数据：

- 重抽样技术：第 1 类数量不够，那就多抽样几次！
- 数据增强技术：第 1 类数量不够，那就创造出样本！

针对降维方式：限于 GPU 调用条件，我们采取 CPU 计算常用方式

- 流形学习：$PCA50+\begin{cases} Tsne & 3 \\ Umap & 10 \\ DenseMap & 10 \end{cases}$

四、进一步探索：重抽样

- 固定第 1 类图片，其他类型不放回抽出等数量图片，组成平衡样本
- 重复抽样，制造出 300 个平衡样本，用于机器学习模型训练
- 集成这 300 个模型的预测结果进行投票，得出最终结果

四、进一步探索：重抽样

重抽样：

```
def resampling(x,y,re=False):
      indices_1 = np.where( y == 1)
      indices_1 = np.concatenate(indices_1)
      x=x.to_numpy()
      x_1 =  [x[idx] for idx in range(len(x)) if idx in indices_1]
      x_others = [x[idx] for idx in range(len(x)) if idx not in indices_1]
      y_others = [y[idx] for idx in range(len(y)) if idx not in indices_1]
      m = len(x_1)
      x_others_resampled = []
      y_others_resampled = []
      indices = np.random.choice(len(x_others), 8*m,replace=re)
      for idx in indices:
            x_others_resampled.append(x_others[idx])
            y_others_resampled.append(y_others[idx])
      x_resampled = x_1 + x_others_resampled
      y_resampled = [1] * m + y_others_resampled
      return x_resampled, y_resampled
```

四、进一步探索：重抽样

集成投票：

```
for i  in tqdm(range(n)):
    X_train_re,y_train_re= resampling(X_train,y_train,True)
    X_train_re=np.array(X_train_re)
    forest.fit(X_train_re, y_train_re)
    #将每次的预测结果存入二维数组,每一行代表一次预测结果
    y_predict = (forest.predict(X_test)).reshape(-1,1)
    #print("RF")
    model.fit(X_train_re, y_train_re)
    y_predict= np.hstack((y_predict,(model.predict(X_test)).reshape(-1,1)))
    #print("LGB")
    xgb_model.fit(X_train_re, y_train_re)
    y_predict= np.hstack((y_predict,(xgb_model.predict(X_test)).reshape(-1,1)))
    #print("xgb")
    res=svc_model.fit(data1,y_train_re)
    y_predict= np.hstack((y_predict,(res.predict(X_test)).reshape(-1,1)))
    #print("svm")
    if i == 0:
        y_predicts = y_predict
    else:
        y_predicts = np.hstack((y_predicts,y_predict))
y_predict = []
for i in range(y_predicts.shape[0]):
    y_predict.append(np.argmax(np.bincount(y_predicts[i])))
print('预测完成')
print(classification_report(y_test, y_predict))
```

四、数据增强：Visual Geometry Group

依赖于 VGG 网络：本质上是一种处理图像轮廓的 CNN。取 VGG 网络的倒数第 5 层作为数据增强后的数据：

```python
vgg16 = models.vgg16(pretrained=True).to(device)
features = vgg16.features[:-5]

vgg16_train_images = []
for image in tqdm(train_images):
    vgg16_image = vgg16_transform(image).to(device)
    vgg16_train_images.append(vgg16_image)

vgg16_train_features = []
for vgg16_image in tqdm(vgg16_train_images):
    vgg16_features = features(vgg16_image)
    vgg16_features = vgg16_features.detach().cpu().numpy()
    vgg16_train_features.append(vgg16_features)
```

Python

传统的图像增广方法主要使用了裁剪，旋转和混合。

四、数据增强：Code Representation

图像裁剪：

```
def random_crop(imgs, imgs_augmented):
    for img in imgs:
        h, w = img.shape[:2]
        crop_w, crop_h = w // 2, h // 2
        x1 = random.randint(0, w - crop_w)
        y1 = random.randint(0, h - crop_h)
        x2 = x1 + crop_w
        y2 = y1 + crop_h
        img = img[y1:y2, x1:x2]
        img = cv2.resize(img, (128, 128))
        imgs_augmented.append(img)
    return imgs_augmented
```

四、数据增强：Code Representation

图像旋转：

```
def random_rotation(imgs, imgs_augmented):
    for img in imgs:
        rows, cols = img.shape[:2]
        angle = random.randint(-30, 30)
        M = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)
        img = cv2.warpAffine(img, M, (cols, rows))
        img = cv2.resize(img, (128, 128))
        imgs_augmented.append(img)
    return imgs_augmented
```

四、数据增强：Code Representation

图像混合：

```python
def random_mixedtran(imgs, imgs_augmented):
    for img in imgs:
        h, w = img.shape[:2]
        crop_w, crop_h = w // 2, h // 2
        x1 = random.randint(0, w - crop_w)
        y1 = random.randint(0, h - crop_h)
        x2 = x1 + crop_w
        y2 = y1 + crop_h
        img = img[y1:y2, x1:x2]

        if np.random.rand() < 0.5:
            img = cv2.flip(img, 1)

        angle = np.random.randint(-15, 15)
        M = cv2.getRotationMatrix2D((img.shape[1]/2, img.shape[0]/2), angle, 1)
        img = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))

        img = cv2.resize(img, (128, 128))
        imgs_augmented.append(img)
    return imgs_augmented
```

四、数据增强：增强结果

训练集增强结果：总共 50144 个图像

- 0: 4069 dot
- 1: 1740 horizontal_bar
- 2: 20094 line
- 3: 8969 scatter
- 4: 15272 vertical_bar

四、进一步探索：流形学习

流形学习 (Manifold Learning)

- 从原始高维数据中发现低维拓扑结构
- 将高维情况下数据点之间的关系保存到低维当中
- 常用于数据降维及其可视化

四、进一步探索：流行学习

t-SNE: t-Stochastic Neighbor Embedding

- 使用概率表征数据间相似性，保持降维映射后拓扑结构
- 主要用于降准区分可视化，最多只能降到三维，易过拟合。

UMAP:

- 基于图论算法，以最近邻构造数据低维布局
- 与 t-SNE 效果类似，但速度更快，且能保持全局结构。

DENSEMAP:

- UMAP 的变种，考虑到了数据点的密度分布
- 可以实现有监督、半监督、无监督降维

四、进一步探索: 3D 图演示

- 将 4 万数据 PCA 降维到 100，再用三种降维方式降到 3 维。
- 认为 Umap10 降维效果最好，且比较不易过拟合

四、进一步探索: 全部数据 +UMAP10+ 机器学习

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1062
1	0.06	0.20	0.10	15
2	0.82	0.85	0.84	4848
3	0.79	0.80	0.79	2274
4	0.90	0.85	0.88	3917
accuracy			0.85	12116
macro avg	0.71	0.74	0.72	12116
weighted avg	0.86	0.85	0.85	12116

图: DTC 分类报告

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1062
1	0.00	0.00	0.00	15
2	0.84	0.91	0.88	4848
3	0.82	0.86	0.84	2274
4	0.96	0.85	0.90	3917
accuracy			0.89	12116
macro avg	0.73	0.72	0.72	12116
weighted avg	0.89	0.89	0.89	12116

图: RF 分类报告

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1062
1	0.07	0.13	0.09	15
2	0.83	0.90	0.87	4848
3	0.81	0.87	0.83	2274
4	0.97	0.83	0.89	3917
accuracy			0.88	12116
macro avg	0.73	0.74	0.74	12116
weighted avg	0.88	0.88	0.88	12116

图: LGB 分类报告

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1062
1	0.00	0.00	0.00	15
2	0.84	0.90	0.87	4848
3	0.81	0.85	0.83	2274
4	0.95	0.86	0.90	3917
accuracy			0.88	12116
macro avg	0.72	0.72	0.72	12116
weighted avg	0.88	0.88	0.88	12116

图: XGB 分类报告

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1062
1	0.33	0.07	0.11	15
2	0.83	0.89	0.86	4848
3	0.78	0.87	0.82	2274
4	0.97	0.82	0.89	3917
accuracy			0.87	12116
macro avg	0.78	0.73	0.74	12116
weighted avg	0.88	0.87	0.87	12116

图: SVC 分类报告

五、最终方案及结果: 最终解决方案

最终解决方案：

- 1、原始图像数据
- 2、将图像数据转为像素数据
- 3、数据增强调整
- 4、随机划分训练集和测试集
- 5、PCA:50
- 6、UMAP:10
- 7、机器学习分类

五、最终方案及结果: 最终结果

	precision	recall	f1-score	support
0	1.00	0.99	1.00	994
1	0.76	0.81	0.79	459
2	0.86	0.88	0.87	5045
3	0.85	0.83	0.84	2304
4	0.92	0.90	0.91	3752
accuracy			0.88	12554
macro avg	0.88	0.88	0.88	12554
weighted avg	0.88	0.88	0.88	12554

图: DTC 分类报告

	precision	recall	f1-score	support
0	1.00	1.00	1.00	994
1	0.86	0.85	0.85	459
2	0.89	0.92	0.90	5045
3	0.87	0.90	0.88	2304
4	0.96	0.90	0.93	3752
accuracy			0.91	12554
macro avg	0.92	0.91	0.91	12554
weighted avg	0.92	0.91	0.91	12554

图: RF 分类报告

	precision	recall	f1-score	support
0	0.99	1.00	1.00	994
1	0.81	0.88	0.85	459
2	0.87	0.90	0.89	5045
3	0.83	0.88	0.86	2304
4	0.97	0.88	0.92	3752
accuracy			0.90	12554
macro avg	0.90	0.91	0.90	12554
weighted avg	0.90	0.90	0.90	12554

图: LGB 分类报告

	precision	recall	f1-score	support
0	1.00	1.00	1.00	994
1	0.85	0.84	0.85	459
2	0.89	0.91	0.90	5045
3	0.87	0.88	0.87	2304
4	0.95	0.91	0.93	3752
accuracy			0.91	12554
macro avg	0.91	0.91	0.91	12554
weighted avg	0.91	0.91	0.91	12554

图: XGB 分类报告

	precision	recall	f1-score	support
0	1.00	1.00	1.00	994
1	0.78	0.88	0.83	459
2	0.86	0.89	0.87	5045
3	0.81	0.87	0.84	2304
4	0.97	0.87	0.92	3752
accuracy			0.89	12554
macro avg	0.88	0.90	0.89	12554
weighted avg	0.89	0.89	0.89	12554

图: SVC 分类报告

五、最终方案及结果: 集成学习和参数搜索

- 集成 ML 模型：让多个 ML 模型结果投票
- 集成降维：PCA7+TSNE3+UMAP10+DENSEMAP10=20 维
- 参数搜索：对五个 ML 模型做网格搜索

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1062
1	0.00	0.00	0.00	15
2	0.83	0.92	0.87	4848
3	0.82	0.85	0.84	2274
4	0.98	0.83	0.90	3917
accuracy			0.88	12116
macro avg	0.73	0.72	0.72	12116
weighted avg	0.89	0.88	0.88	12116

图: 重抽样集成结果

结论：总体准确率基本上没什么浮动 0.85-0.93。数据增强让各类的预测准确率都有所提高，其中使类型 1 的准确率从 0 上升到了大约 85%。