机器学习之一：聚类实战

可预见的未来数据分析和机器学习将成为工作中必备技能，也许已经在某个项目中讨论怎么调参优化，就像过去讨论如何优雅的写 python、如何避免 C++内存泄露一样常见。

一、简单介绍聚类算法

1. 聚类的定义：

聚类就是对大量未知标注的数据集，按数据的内在相似性将数据集划分为多个类别，使类别内的数据相似度较大而类别间的数据相似度较小

2. 聚类的基本思想：

给定一个有 N 个对象的数据集，构造数据的 k 个簇，k≤n。满足下列条件：

- 每一个簇至少包含一个对象
- 每一个对象属于且仅属于一个簇
- 将满足上述条件的 k 个簇称作一个合理划分

对于给定的类别数目 k，首先给出初始划分，通过迭代改变样本和簇的隶属关系，使得每一次改进之后的划分方案都较前一次好。

3. 相似度/距离计算方法总结

- 闵可夫斯基距离Minkowski/欧式距离 $dist(X,Y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$
- 杰卡德相似系数(Jaccard) $J(A,B)=\frac{|A\cap B|}{|A\cup B|}$
- 余弦相似度(cosine similarity) $\cos(\theta)=\frac{a^Tb}{|a|\cdot|b|}$
- Pearson相似系数 $\rho_{XY}=\frac{\mathrm{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}=\frac{\sum_{i=1}^{n}(X_i-\mu_X)(Y_i-\mu_Y)}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_Y)^2}}$
- 相对熵(K-L距离) $D(p\|q)=\sum_x p(x)\log\frac{p(x)}{q(x)}=E_{p(x)}\log\frac{p(x)}{q(x)}$
- Hellinger距离 $D_\alpha(p\|q)=\frac{2}{1-\alpha^2}\left(1-\int p(x)^{\frac{1+\alpha}{2}}q(x)^{\frac{1-\alpha}{2}}dx\right)$

4. K-means 算法：

K-means 算法也被称为 k 均值，k 值的选择、距离度量及分类决策是三个基本要素

假定输入样本为 S=x1,x2,...,xm，则算法步骤为：

选择初始的 k 个类别中心 μ1μ2…μk

对于每个样本 xi，将其标记为距离类别中心最近的类别：

将每个类别中心更新为隶属该类别的所有样本的均值：

重复最后两步，直到类别中心的变化小于某阈值。

中止条件：

迭代次数/簇中心变化率/最小平方误差 MSE(Minimum Squared Error)

5. 一个简单的例子：

```
from sklearn.neighbors import NearestNeighbors
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
nbrs = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').fit(X)
distances, indices = nbrs.kneighbors(X)
```

二、项目实战

某专项测试实际业务中，海量样本为同一病毒类型，如何落地为本地能力将是挑战，所有样本都处理工作量大且重复性高，只处理高热样本会落入长尾困境，如果能将 N 个样本通过特征聚类为 K 类，报毒覆盖 K 类则理论会达到覆盖整体的能力，无论效率和产品能力、自动化上都将有收益。

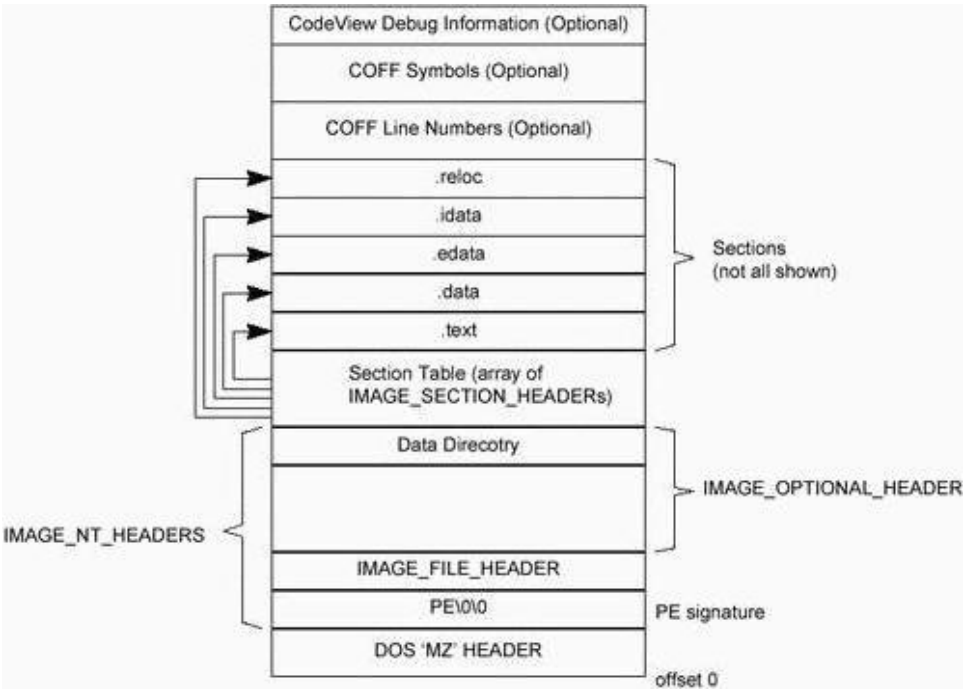
具体的思路如下：

1. 数据清洗：提取相同病毒名的文件
2. 特征提取：提取多维度文件静态特征
3. 聚类：K-means，目标聚类覆盖该类型病毒特征
4. 特征验证：k 个特征对 k 个子编写特征验证通杀性

工具包：NumPy、SkiPy、 Pandas、Skikit-Learn

1. 数据清洗：

PE 文件结构和样本特征的关系：常用的恶意文件一般都是基于格式分析，从 PE 文件格式分析来提取文件特征符合业务特征



这里使用本人在 filefuzz 项目里封装的 pe 解析模块来处理，拉取某报毒类型样本 5722 个，去除坏 PE 后解析出下列参数做为维度参数

NumberOfSections, SizeOfCode, BaseOfData, ImageBase, SizeOfImage, SizeOfHeaders, IMAGE_DATA_DIRECTORY[16], IMAGE_DIRECTORY_ENTRY_IMPORT

```python
# [IMAGE_DATA_DIRECTORY]     #_*_*_*_#
data_directory_list = []
for directory in pe.OPTIONAL_HEADER.DATA_DIRECTORY:
    dic = {}
    dic['Name'] = GetStrMd5(directory.name)
    if directory.Size:
        dic['Size'] = 1
    else:
        dic['Size'] = 0
    data_directory_list.append(dic)
peInfo.append("IMAGE_DATA_DIRECTORY\t%s"%(data_directory_list))

# [IMAGE_DIRECTORY_ENTRY_IMPORT]
directory_entry_import_list = []
for importeddll in pe.DIRECTORY_ENTRY_IMPORT:
    dic = {}
    dic['Name'] = GetStrMd5(importeddll.dll)
    dllL = []
    for importedapi in importeddll.imports:
        dllL.append(GetStrMd5(importedapi.name))
    dic['Function'] = dllL
    directory_entry_import_list.append(dic)
peInfo.append("IMAGE_DIRECTORY_ENTRY_IMPORT\t%s"%(directory_entry_import_list))
```

2. 特征提取：

去重后导入函数为 688，参数维度共 711 个

处理前特征

```
NumberOfSections	0x4
SizeOfCode	0x4b000
BaseOfCode	0x1000
BaseOfData	0x11000
ImageBase	0x400000
SizeOfImage	0x50000
SizeOfHeaders	0x1000
IMAGE_SECTION_HEADER	[{'SizeOfRawData': '0x10000', 'Name': 'b664c[redacted]', 'Misc_VirtualSize': '
IMAGE_DATA_DIRECTORY	[{'Name': '7e3[redacted]', 'Size': 0}, {'Name': 'b039[redacted]
IMAGE_DIRECTORY_ENTRY_IMPORT	[{'Function': ['a74d83c0[redacted]', '82f585[redacted]',
```

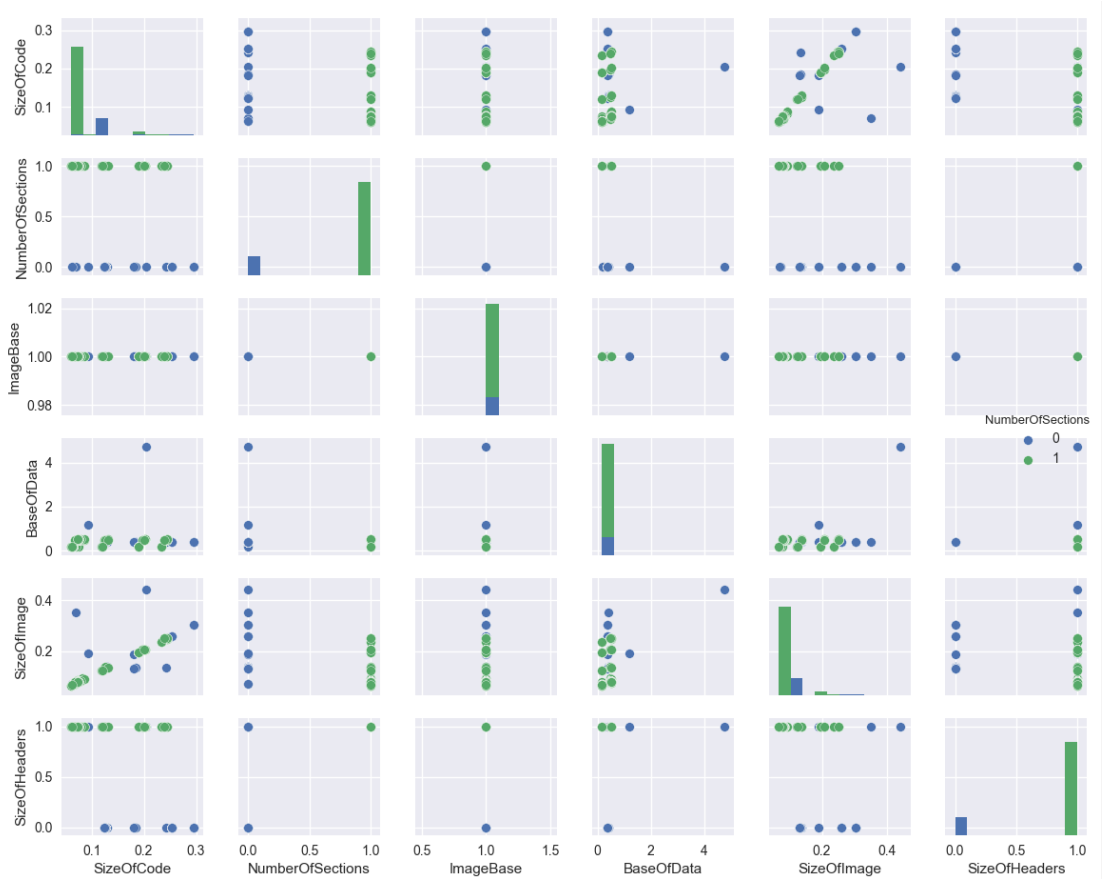
处理后特征

```
"NumberOfSections","SizeOfCode","BaseOfCode","BaseOfData","ImageBase","SizeOfImage","SizeOfHeaders","dir0","dir1","dir2","dir3","dir4","dir5",
4,258048,4096,20480,4194304,274432,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
4,1003520,4096,61440,4194304,1052672,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
10,519168,4096,49152,4194304,548864,1024,0,1,1,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,(
4,307200,4096,69632,4194304,327680,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
4,258048,4096,20480,4194304,274432,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
4,262144,4096,20480,4194304,278528,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
4,307200,4096,69632,4194304,327680,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
4,524288,4096,61440,4194304,573440,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
4,262144,4096,20480,4194304,278528,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
4,303104,4096,65536,4194304,323584,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,
4,286720,4096,61440,4194304,335872,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
4,303104,4096,65536,4194304,323584,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,
4,286720,4096,61440,4194304,335872,4096,0,1,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
```

3. 聚类：K-means

使用 pandas 加载数据后填充缺失数据，通过特征分布可视化预处理参数观察数据分布

SizeOfCodehe 和 SizeOfImage 在数据分布上区分度较好



数据加载

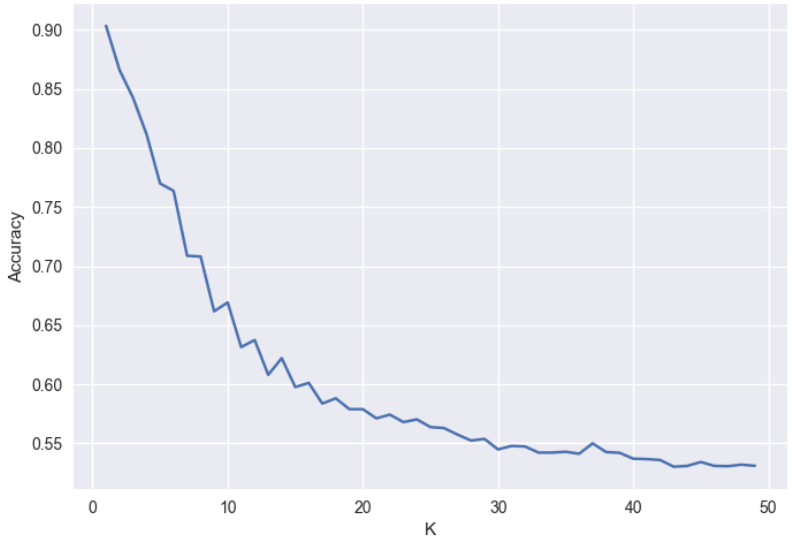
```
数据集有4746行，711列
数据预览：
   NumberOfSections  SizeOfCode  BaseOfCode  BaseOfData  ImageBase
0                 4      258048        4096       20480    4194304
1                 4     1003520        4096       61440    4194304
2                10      519168        4096       49152    4194304
3                 4      307200        4096       69632    4194304
4                 4      258048        4096       20480    4194304

   SizeOfImage  SizeOfHeaders  dir0  dir1  dir2  ...  func_678
0       274432           4096     0     1     1  ...         0
1      1052672           4096     0     1     1  ...         0
2       548864           1024     0     1     1  ...         0
3       327680           4096     0     1     1  ...         0
4       274432           4096     0     1     1  ...         0

   func_680  func_681  func_682  func_683  func_684  func_685  func
0         0         0         0         0         0         0
1         0         1         1         0         0         0
2         0         0         0         0         0         0
3         0         1         1         0         0         0
4         0         0         0         0         0         0

   func_687
0         0
1         0
2         0
3         1
4         0

[5 rows x 711 columns]
```

特征归一化

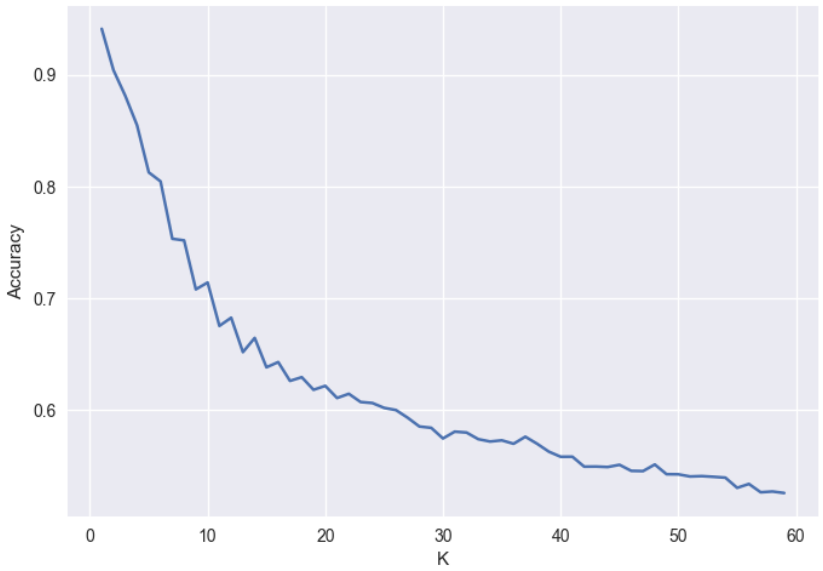
```
[-1.730956   -0.44176559  4.17773802  0.          0.4186576   0.
  4.20784992  0.4410933  -0.01451717  0.          0.          0.
 -0.01451717 -0.4410933   0.03247506  0.          0.          0.
 -0.01451717  0.          0.4410933   1.85239643  0.         -0.4410933
 -0.01451717 -0.08619409 -0.01451717 -0.08619409 -0.01451717  1.16972795
 -0.08619409  0.79113797 -0.01451717 -0.49433073 -0.02053254  0.79219443
  1.16922294 -0.01451717  0.44279628 -0.47676828 -0.08742604 -0.08619409
 -0.01451717 -0.01451717 -0.08742604 -0.01451717 -0.01451717 -0.01451717
 -0.01451717 -0.02053254 -0.01451717 -0.49400071  1.19795447 -0.01451717
 -0.02514977 -0.4764354  -0.02514977 -0.01451717 -0.01451717 -0.08619409
 -0.01451717 -0.08619409 -0.01451717 -0.08619409  0.43013803 -0.01451717
 -0.01451717 -0.08619409 -0.08619409  0.74980796 -0.08619409 -0.01451717
 -0.08619409  1.85239643 -0.01451717 -0.78481503 -0.01451717 -0.01451717
 -0.49400071  1.19795447 -0.08619409  0.79184219 -0.01451717 -0.02053254
 -0.4764354   1.19691214  0.7375071  -0.08619409 -0.08619409 -0.01451717
 -0.08619409 -0.01451717  0.79219443 -0.42910574 -0.02053254 -0.01451717
  1.19795447 -0.01451717 -0.49499065 -0.01451717  0.80493271 -0.78481503
  1.85239643 -0.01451717 -0.08619409  1.19743315 -0.48838279 -0.01451717
 -0.49400071 -0.01451717 -0.01451717 -0.08742604 -0.01451717  1.81201366
 -0.01451717  1.81308104 -0.08619409 -0.49433073  0.74877956 -0.47676828
  0.79113797  0.73104897 -0.02514977  1.85128032  0.79149004  0.80493271
 -0.8604499  -0.02514977 -0.01451717 -0.4764354   0.74980796 -0.47676828
 -0.49400071 -0.01451717  0.44245587 -0.02514977  1.85128032  1.85128032
 -0.02514977 -1.27304814  1.8490523  -0.08619409 -0.01451717 -0.02053254
 -0.02053254 -0.08619409 -0.08619409 -0.01451717 -0.08619409 -0.85379257
```

分割训练集和测试集:不同目标参数训练结果如下（x 轴表示训练次数 y 轴表示）

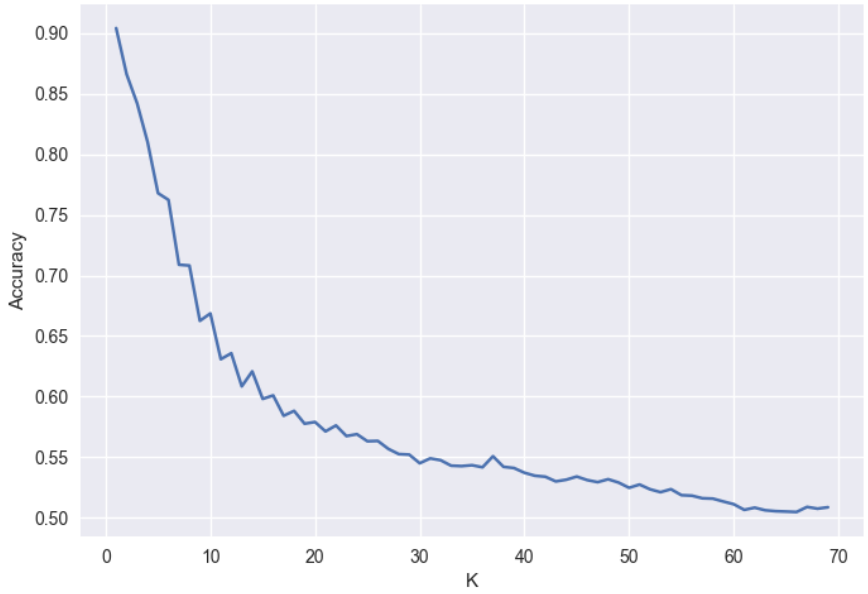
y = voice_data['NumberOfSections'].values+voice_data['SizeOfCode'].values



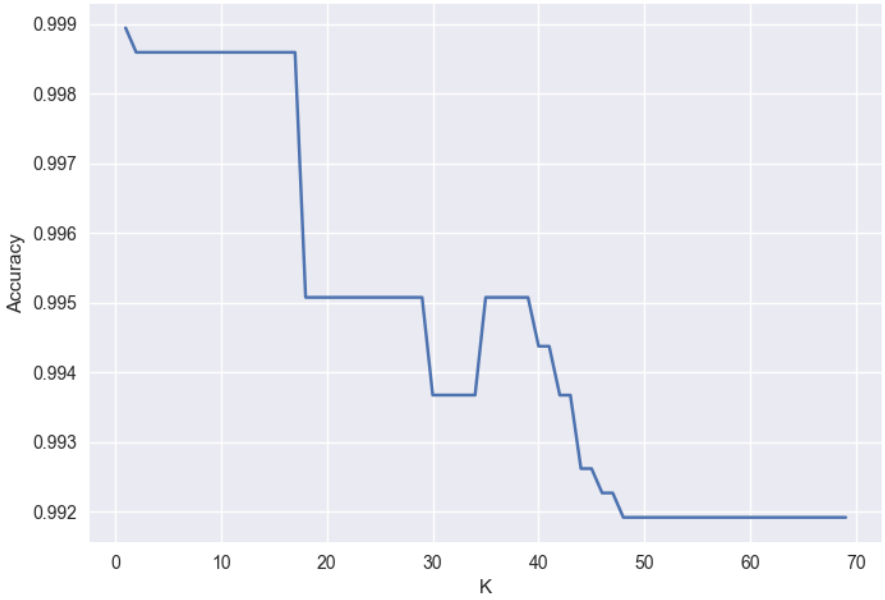
y = voice_data['NumberOfSections'].values+voice_data['SizeOfImage'].values



y = voice_data['SizeOfImage'].values+voice_data['SizeOfCode'].values



y = voice_data['NumberOfSections'].values, 此时目标参数区分度有效性最高，准确率也达到 99%



sklearn 函数介绍：

train_test_split 将给定数据集 X 和类别标签 Y，按一定比例随机切分为训练集和测试集。

X_train,X_test,y_train,y_test =

train_test_split(train_data,train_target,test_size=0.4, random_state=0)

train_data： 所要划分的样本特征集

train_target：所要划分的样本结果

test_size： 样本占比

random_state：是随机数的种子。

http://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.train_test_split.html

cross_val_score 交叉验证函数

scores = cross_val_score(clf, raw_data, raw_target, cv=5, score_func=None)

clf: 表示不同分类器,例支持向量 clf=svm.SVC(kernel='linear', C=1)

raw_data：原始数据

raw_target：原始类别标号

cv：不同的 cross validation 的方法

cross_val_score：不同划分 raw_data 在 test_data 得到分类的准确率

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

代码简要：

```
voice_data = pd.read_csv(filepath)
voice_data = process_missing_data(voice_data)
X = voice_data.iloc[:].values
y = voice_data['NumberOfSections'].values
X = preprocessing.scale(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=5)
k_range = range(1,70)
cv_scores = []
for k in k_range:
    knn = KNeighborsClassifier(k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    score_mean = scores.mean()
    cv_scores.append(score_mean)
knn_model = KNeighborsClassifier(np.argmax(cv_scores) + 1)
knn_model.fit(X_train, y_train)
```

4. 特征验证：

k 个特征对 k 个子类编写特征验证通杀性，样本处理数量从 4726 下降到 K 类集合

总结：

通过对大量同质数据的聚类，对测试集合的覆盖度和效率都有显著收益，对长尾问题解决也提供了可行的思路方法。

参考：

http://scikit-learn.org/stable/

<<计算机病毒防范艺术>>

<<统计学习方法>>

<<机器学习>>