反信用卡欺诈

本章为针对信用卡欺诈的检测技术

数据集

kaggle上Credit card fraud detection数据集。在284807次交易中，包含了492例诈骗，数据集及其不均衡。

该数据集已经做了脱敏处理及向量化，最后使用28维向量描述，分别对应v1-v28，该笔交易发生时间为Time，涉及金额为Amount。class表示该笔交易是否为欺诈。

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251412	-0.01831	0.277838	-0.11047	0.066928	0.128539	-0.18911	0.133558	-0.02105	149.62	0
0	1.191857	0.266151	0.166048	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.101288	-0.33985	0.16717	0.125895	-0.00898	0.014724	2.69	0

特征提取

标准化

V1-V28已经归一化处理过，Amount需要进行归一化处理，将数据控制在-1到1之间。

标准化和降采样

由于数据极度不均衡，常见的分类算法都会偏向数据量占优势的一方，为了避免这种情况，采用降采样方法。即，从数据量占优的数据集中**随机抽取**一定数量的样本，通常抽取的数量与数据量小的样本数量相当。

使用np.random.choice实现

```
import numpy as np
np.random.choice(a, size=None, replace=True)
#从a中抽取size的样本，有放回的抽取。
```

标准化和过采样

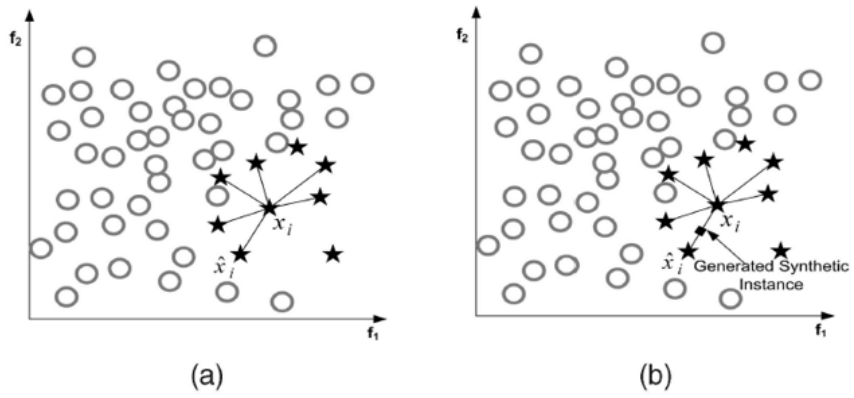
过采样保留数据占优的样本，通过一定的算法，在数据量较小的样本的基础上**生成新样本。**

smote algorithm

使用smote algorithm生成新样本。

基本原理：

SMOTE算法的思想是合成新的少数类样本，合成的策略是对每个少数类样本a，从它的最近邻中随机选一个样本b，然后在a、b之间的连线上随机选一点作为新合成的少数类样本。如图所示：



算法流程：

1、对于少数类中每一个样本a，以欧氏距离为标准计算它到少数类样本集中所有样本的距离，得到其k近邻。

2、根据样本不平衡比例设置一个采样比例以确定采样倍率N，对于每一个少数类样本a，从其k近邻中随机选择若干个样本，假设选择的近邻为b。

3、对于每一个随机选出的近邻b，分别与原样本a按照如下的公式构建新的样本:**c=a+rand(0,1)∗|a−b|**

实现:

```
# 使用imlbearn库中上采样方法中的SMOTE接口
from imblearn.over_sampling import SMOTE
# 定义SMOTE模型，random_state相当于随机数种子的作用
smo = SMOTE(random_state=42)
X_smo, y_smo = smo.fit_sample(X, y)
```

模型的训练与验证

朴素贝叶斯

```
from sklearn.naive_bayes import GaussianNB
x_train, x_test, y_train, y_test = undefsampling(data)
model = GaussianNB()  #gnb+undersample，acc=0.8985  oversample,acc=0.9741

trained_model = model.fit(x_train, y_train)
y_pred = trained_model.predict(x_test)
acc = accuracy_score(y_test, y_pred)
clf = classification_report(y_test, y_pred)
print(acc, '\n', clf)
```

XGBoosting

```
from xgboost import XGBClassifier
x_train, x_test, y_train, y_test = undefsampling(data)
model = XGBClassifier(n_estimate=150, max_depth=9)#xgb+undersample,acc=0.9238
```

MLP

```
from sklearn.neural_network import MLPClassifier
x_train, x_test, y_train, y_test = undefsampling(data)
model = MLPClassifier(hidden_layer_sizes=(5, 2), activation='relu', solver='adam',
                          shuffle=True, verbose=True) #mlp+undersample，acc=0.934
```