第12章　反欺诈模型实例

Jupyter金融应用从入门到实践

场景介绍

金融欺诈是指以获取非法收益为目的，通过非常规手段进行交易的行为，包括网络诈骗、电话诈骗、账户盗用等。通过对欺诈交易行为进行识别，能够有效支持银行风险管理，降低操作风险。在线的反欺诈是互联网金融必不可少的一部分，本章涉及的反欺诈案例，来自消费金融领域的信用卡盗刷。

由于数据集过大，仅抽取 10%的数据进行建模。采样前可通过设定随机数种子，保证随机采样的结果可重复。采样后共 28481 条数据，其中负样本 49 条。该数据集数据类型只有 float64 和 int64，共 31 个变量，无缺失值。

[In]

```
# 设定随机数种子
np.random.seed(seed=2020)
# 读取数据
data=pd.read_csv(r"./dataset/creditcardfraud.csv")
# 采样
sample=data.sample(frac=0.1)
sample.shape
```

[Out]

```
(28481, 31)
```

特征工程

	Time	Amount	V1	V2	V3
count	28481.000000	28481.000000	28481.000000	28481.000000	28481.000000
mean	95081.280749	87.202118	0.026534	0.011319	-0.007368
std	47483.822980	231.158456	1.919344	1.582511	1.493663
min	1.000000	0.000000	-35.698345	-40.978852	-29.468732
25%	54476.000000	5.850000	-0.894757	-0.606529	-0.897072
50%	85329.000000	22.470000	0.038758	0.070498	0.184776
75%	139690.000000	78.000000	1.322692	0.813802	1.025286
max	172784.000000	7712.430000	2.418802	15.876923	4.101716

通过对特征的统计可以发现，特征Time和特征Amount相对于其他特征取值［均值（mean）或最大值（max）］特别大，离散程度［标准差（std）］也特别高。

这是由于特征Time是按秒计，取值过大，可将时间从单位秒转换为单位小时。而对于交易金额（Amount），可采用对数压缩取值范围，注意取对数的时候要留意非法零值，因此可对Amount+1取对数，即ln(Amount+1)。

经过特征转换，特征Time和Amount的数据规格与特征V1至V28仍有一定差别，需要对其进行特征缩放，将特征缩放至同一个规格。这是因为在聚类算法中，使用了距离来度量样本的相似性，但是特征变量的量纲和数值的量级不一样，对输出变量的影响程度就不一样。例如，使用欧式距离的一些距离算法，可能在某些取值较大的差异特征上计算出远超一般水平的距离数据。原始数据经过特征缩放处理后，可以消除特征间的规格差异，各变量处于同一数量级，适合进行综合对比评价。主要的方法有3种：正则化（Normalization），最小值-最大值归一化（Min-Max Scaling）以及Z-score标准化（Z-score Normalization）。

通过 L1 正则可将数据的分布变成一个半径为 1 的圆。

[In]

```
# 正则化
from sklearn.preprocessing import Normalizer
x_cols=sample.columns.tolist()  # 解释变量
x_cols.remove('Class')
# 正则化
sample[x_cols]  = Normalizer().fit_transform(sample[x_cols])
```

[Out]

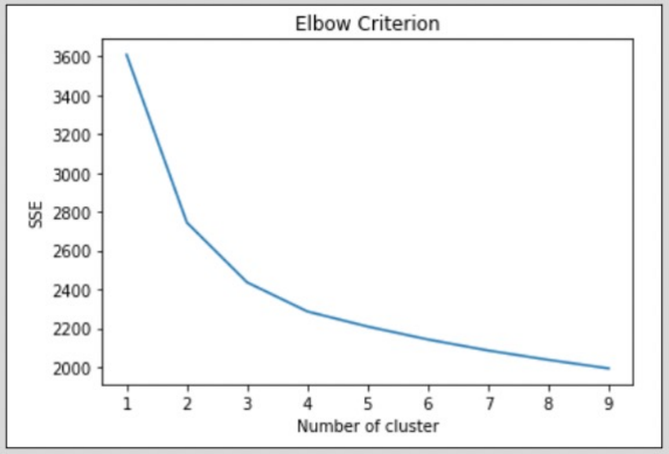
本章使用K均值聚类方法来处理信用卡反欺诈数据集。在这个数据集里，每一条样本代表一条信用卡交易记录，是否存在欺诈由变量Class给出。在训练模型时，可以放弃这一变量。

在代码的执行结果中，从二类到四类变化时，组内的平方总和有一个明显的下降趋势。特别是四类之后，下降的速度减弱，暗示着聚成二类到四类可能对数据来说是一个很好的拟合。

[In]

```
# 通过 Elbow Criterion 确定 K
sse = {}
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(sample[x_cols])
    sse[k] = kmeans.inertia_
# 画出组内的平方和和提取的聚类个数的对比
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.title('Elbow Criterion')
plt.show()
```

[Out]



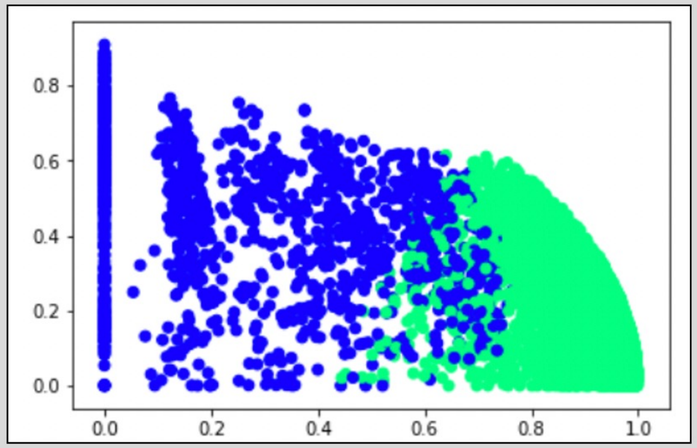
使用 sklearn.cluster 中的 Kmeans()函数可以完成建模及训练，聚类效果如运行结果图所示，在特征 Time 和 Amount 的二维图上，模型对数据有一定的分离效果和区分度，特别是在输出结果中，在浅色的簇上表现出明显的簇内集中。

[In]

```
# K 均值聚类建模
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)  # 创建模型
kmeans.fit(sample[x_cols]) # 使用无标签数据训练

# 聚类效果可视化
plt.scatter(sample['Time'], sample['Amount'],
        c = KMeans(n_clusters = 2).fit_predict(sample[x_cols]),cmap =plt.cm.winter)
plt.show()
```

[Out]



[In]
```
1.    # 统计标签
2.   Y_true=sample['Class'].values
3.   Y_predict= kmeans.fit_predict(sample[x_cols])
4.   from collections import Counter
5.   print('实际标签分布：',Counter(Y_true))
6.   print('预测标签分布：',Counter(Y_predict))
```

[Out]
```
实际标签分布： Counter({0: 28432, 1: 49})
预测标签分布： Counter({1: 27020, 0: 1461})
```

可以通过 NumPy 中的 Where 函数，做位运算交换 0 和 1。

[In]
```
1.    # 交换标签的 0 和 1
2.   Y_predict=np.where((Y_predict==0)|(Y_predict==1), Y_predict^1, Y_predict)
```

[Out]

通过分类报告可以看出，虽然总体上预测准确度（Total Precision）接近 100%，并且 F1-Score 也达到了 97%，但这是由标签的不平衡分布引起的。如果只关注负样本的表现，召回率只有 27%，这意味着在 49 个负样本中，只有大约 13 个负样本被准确预测到。

谢谢！