《数据挖掘导论》实验 4：电力窃漏电用户自动识别

一、 实验目的：

（1）掌握拉格朗日插值法进行缺失值处理的方法。
（2）掌握 LM 神经网络和 CART 决策树构建分类模型的方法。

二、 实验环境

（1）Anaconda 开发环境。
（2）IDE 为 Jupyter Notebook。
（3）库：pandas、scipy.interpolate.lagrange、sklearn。

三、 实验内容

传统的防窃漏电方法对人的依赖性太强。现需要通过在线稽查系统和现场稽查来找出窃漏电用户，并录入系统。通过这些信息提取出窃漏电用户的关键特征，构建窃漏电用户的识别模型，实现自动检查，判断用户是否存在窃漏电行为。具体实验要求如下：
（1）利用拉格朗日插值法补全用户的用电数据中的缺失值。
（2）对所有窃漏电用户及正常用户的电量、告警及线损数据和该用户在当天是否窃漏电的标识，按窃漏电评价指标进行处理并选取其中 291 个样本数据，得到专家样本，分别使用 LM 神经网络和 CART 决策树实现分类预测模型，使用混淆矩阵和 ROC 曲线对模型进行评价。

实验一

1. 打开 Python 软件，把“data/”目录下的 missing_data.xls 和 model.xls 数据放入当前工作目录。

missing_data.xls 是三个用户一个月工作日的电量数据。数据中存在缺失值。

model.xls（专家样本数据）是所有窃漏电用户及正常用户的电量、告警及线损数据和该用户在当天是否窃漏电的标识，按窃漏电评价指标进行处理并选取其中 291 个样本数据所组成。

model.xls 中各个字段的含义

Filed	Sample	Description
电量趋势下降指标	4	当天比前一天用电趋势递减，即设：

		$D(i)=\begin{cases}1, k_i < k_{i-1}\\0, k_i \geq k_{i-1}\end{cases}$ 电量趋势下降指标的值为:$T=\sum_{n=i-4}^{i+5} D(n)$
线损指标	1	衡量供电线路的损失比例。
告警类指标	1	与窃漏电相关的终端报警主要有电压缺相,电压断相，电流反极性等告警,计算发生于窃漏电相关的终端报警的次数总和作为告警类指标。
是否窃漏电	1	数据标签

2. 使用 pandas 把数据’missing_data.xls’读入当前工作目录。因为数据不包含列名，所以 header 设置为 None。

```
import pandas as pd
data=pd.read_excel('../data/missing_data.xls',header=None)
```

pandas.read_excel(io, sheetname=0, header=0, skiprows=None, skip_footer=0,index_col=None,names=None)

- 常用参数解析
  - io ： excel 路径。
  - sheetname ：返回多表。
  - header ：指定列名行，默认 0，即取第一行，数据为列名行以下的数据 若数据不含列名，则设定 header = None。
  - skiprows ：省略指定行数的数据。
  - skip_footer ：省略从尾部数的 int 行数据。
  - index_col ：指定列为索引列，也可以使用 u”strings”。
  - names ：指定列的名字。

3. 查看数据的基本情况

用 describe()函数查看数据的基本情况，结果如下图所示。

```
data.describe()
```

	0	1	2
count	19.000000	17.000000	17.000000
mean	236.262626	363.666265	553.901624
std	1.225465	57.600529	67.707729
min	234.468800	206.434900	435.350800
25%	235.494800	328.089700	514.890000
50%	235.906300	388.023000	538.347000
75%	237.192750	401.623400	611.340800
max	238.656300	416.879500	660.234700

其中 count 是非空值数，通过 len(data)可以知道数据为 21 条，因此 A 用户缺失值数为 2，B,C 用户为 4 条。此外，describe()函数提供的基本参数还有：平均值，标准差，最小值，最大值以及 1/4、1/2、3/4 分位数，更直观的展示数据。

4. 处理数据中的缺失值

处理缺失值的方法分为三类：删除记录、数据插补和不处理。常用插补法为以下几种：均值/中位数/众数插补、固定值插补、最近邻插补、回归方法、插值法。本实验中要求使用拉格朗日插值法，具体原理如下：

首先从原始数据确定因变量和自变量，取出缺失值前后 5 个数据（前后数据中遇到数据不存在或为空，直接将数据舍去，将仅有的数据组成一组），根据取出来的 10 个数据组成一组，然后采用拉格朗日多项式插值公式：

$$L_n(x) = \sum_{i=0}^{n} l_i(x) y_i$$

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

其中，x 为缺失值对应的下标序号，$L_n(x)$为缺失值的插值结果，x_i为非缺失值y_i的下标序号。对全部缺失数据依次进行插补，直到不存在缺失值为止。

5. 对读入的数据缺失值进行拉格朗日插值

1) 针对每列数据的每一个缺失值，逐个进行补数（这样可以在连续两个缺失值的情况下，使用前面一个已经补数的值来再次补数后面一个值）
2) 针对一个缺失值，构造参考组。选取前面 5 个作为前参考组，后 5 个为后参考组。如果前参考组或后参考组不足 5 个，则按实际个数构造参考组。

第一行导入需要的数据分析库包 pandas，第二行导入 scipy.interpolate 的 lagrange 拉格朗日插值函数，用于补全缺失值。

```
import pandas as pd
from scipy.interpolate import lagrange
```

第一行至第四行自定义列向量插值函数，第一行 s 为列向量，n 为被插值的位置，k 为取前后的数据个数，默认为 5。第二行，将待插值元素的前后各 k 个数据赋予 y，第三行将不存在或为空的数据，直接舍去，将仅有的数据组成一组。

```
def polyinterp_column(s,n,k=5):
    y=s.reindex(list(range(n-k,n))+list(range(n+1,n+1+k)))
    y=y[y.notnull()] #剔除空值
    return lagrange(y.index,list(y))(n) #插值并返回插值结果
```

拉格朗日函数的使用：lagrange(x, y)参数 x 和 y 分别是对应各个点的 x 值和 y 值。举例，第一行和第二行将（1,5），（2,7），（3,10），（4,3），（7,9）作为函数的输入，第三行调用拉格朗日函数将结果赋予 a，第五至第七行打印结果，

```
1.from scipy.interpolate import lagrange
2.x=[1,2,3,4,7]
3.y=[5,7,10,3,9]
4.a=lagrange(x,y)
5.print(a)
6.print(a(1),a(2),a(3))
7.print(a[0],a[2],a[3])
```

上述代码的结果如下图所示。打印 a 的值，显示出了插值的函数。a(1),a(2),a(3)提取的是输入 x 对应的值。a[1],a[2],a[3]提取的是插值函数中 x 的一次，二次和三次方的系数。

```
a的值为：
          4          3          2
0.5472 x - 7.306 x + 30.65 x - 47.03 x + 28.13
------------------------------------------------
a(1):5.000000000000007
a(2):7.000000000000014
a(3):10.00000000000005
------------------------------------------------
a[1]:-47.02777777777778
a[2]:30.65277777777778
a[3]:-7.3055555555555545
```

3) 确认缺失值在参考组的相对位置然后使用拉格朗日插值进行缺失值插值。逐个元素判断是否需要插值。第一行代码遍历数据的每一列，第二行代码遍历数据的每一行，第三行代码，判断数据是否为空，第五行代码，若数据为空，调用拉格朗日插值函数填充数据。

```
for i in data.columns:
    for j in range(len(data)):
        if(data[i].isnull())[j]:
            data[i][j]=polyinterp_column(data[i],j)
```

4) 根据插值后的值更新原始数据中相应位置的值，并导出数据。

```
data.to_excel('missing_data_filled.xls',header=None,index=False) #输出结果
```

实验二

1. 把经过预处理的专家样本数据 model.xls 数据放入当前工作目录，并使用

Pandas 读入当前工作空间。数据的签三列是特征，后一列是标签。

```
data=pd.read_excel('../data/model.xls')
```

2. 把专家样本数据 model.xls 分为两部分，一部分用于训练，一部分用于测试，一般训练数据取总数据的 80%，测试数据取 20%。第一行代码，导入随机函数 shuffle，第二行将表格转换为矩阵，第三行随机打乱数据，为训练数据和测试数据的选取做准备。

```
from random import shuffle
data = data.iloc[:,:].values
shuffle(data)
```

第一行设置训练数据比率为 0.8，第二行和第三行分别取出训练数据和测试数据。

```
p = 0.8
train = data[:int(len(data)*p),:] #前 80%为训练集
test = data[int(len(data)*p):,:] #后 20%为测试集
```

3. 使用 Scikit-Lrean 库的 sklearn.tree 的 DecisionTreeClassifier 函数以及训练数据构建 CART 决策树模型，使用 predict 函数和构建的 CART 决策树模型分别对训练和测试数据进行分类，并与真实值做对比，得到模型正确率。同时使用 sklearn.metrics 的 confusion_matrix 和 roc_curve 函数画混淆和 ROC 曲线图。

1) 实验中需要导入的包如下：

```
from sklearn.tree import DecisionTreeClassifier #导入决策树模型
from sklearn.metrics import confusion_matrix #导入混淆矩阵函数
import joblib
import matplotlib.pyplot as plt #导入作图库
from sklearn.metrics import roc_curve #导入 ROC 曲线函数
```

2) 建立决策树。第一行建立决策树模型，第二行训练数据。

```
tree = DecisionTreeClassifier()
tree.fit(train[:,:3],train[:,3]) # train[:,:3]为样本特征，train[:,3]为样本标签
```

3) 保存模型，第二项参数为将模型存放的文件路径

```
joblib.dump(tree, 'train_model.m')
```

4) 画混淆矩阵图。混淆矩阵是机器学习中总结分类模型预测结果的情形分析表，以矩阵形式将数据集中的记录按照真实的类别与分类模型作出的分类判断两个标准进行汇总。这个名字来源于它可以非常容易的表明多个类别是否有混淆。第一行使用混淆函数，使用方法后面会介绍，第二行绘制混淆矩阵。第三至六行给图绘制标签。

```
cm = confusion_matrix(train[:,3], tree.predict(train[:,:3])) #混淆矩阵
plt.matshow(cm, cmap=plt.cm.Greens) #画混淆矩阵图，配色风格使用cm.Greens
plt.colorbar() #颜色标签
for x in range(len(cm)): #数据标签
    for y in range(len(cm)):
```

```
plt.annotate(cm[x,y],xy=(x,y),horizontalalignment='center',verticalalignment='center')
```

confusion_matrix(y_true, y_pred, labels=None, sample_weight=None)

- 参数解析
  - y_true: 是样本真实分类结果，y_pred: 是样本预测分类结果。
  - labels: 是所给出的类别，通过这个可对类别进行选择。
  - sample_weight : 样本权重。

实例如下：

```
#混淆矩阵例子
from sklearn.metrics import confusion_matrix
y_true=[2,1,0,1,2,0]
y_pred=[2,0,0,1,2,1]
C=confusion_matrix(y_true, y_pred)
print("C:%s"%(C))
```

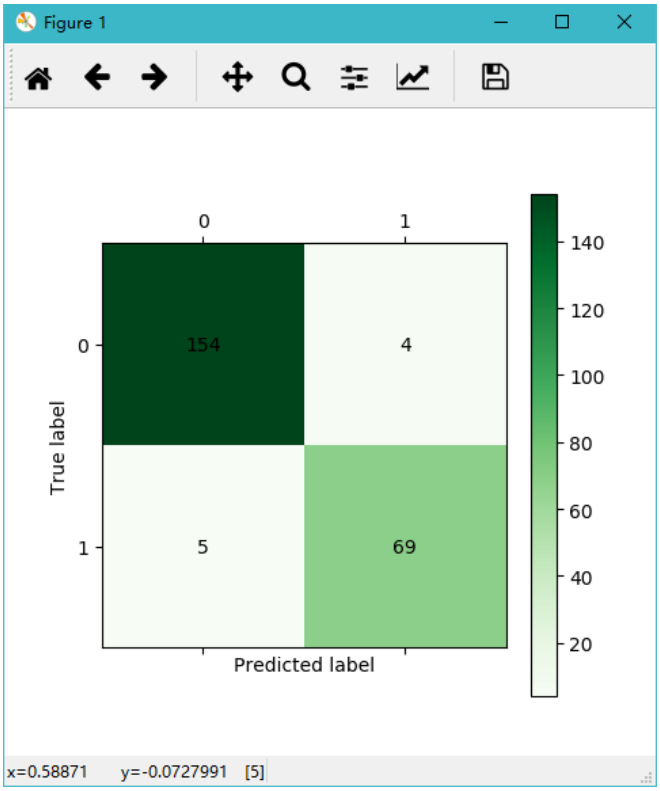
输出结果：

```
C:[[1 1 0]
 [1 1 0]
 [0 0 2]]
```

annotate(s='str' ,xy=(x,y) ,xytext=(11,12) ,...)

- 参数解析
  - s 为注释文本内容
  - xy 为被注释的坐标点
  - xytext 为注释文字的坐标位置
  - verticalalignment: 垂直对齐方式，可选‘center’、‘top’、‘bottom’、‘baseline’ 。
  - horizontalalignment:水平对齐方式,可选‘center’、‘right’、‘left’。
  - xycoords 参数 figure points 点在图左下方; figure pixels 图左下角的像素; figure fraction 左下角数字部分; axes points 从左下角点的坐标; axes pixels 从左下角的像素坐标; axes fraction 左下角部分;data 使用的坐标系统被注释的对象(默认)。
  - extcoords 设置注释文字偏移量，参数 figure points 距离图形左下角的点数量 ; figure pixels 距离图形左下角的像素数量 ; figure fraction 0,0 是图形左下角，1,1 是右上角 ; axes points 距离轴域左下角的点数量 ;axes pixels 距离轴域左下角的像素数量 ;axes fraction 0,0 是轴域左下角，1,1 是右上角;data 使用轴域数据坐标系。
  - arrowprops 箭头参数,参数类型为字典 dict，width 点箭头的宽度; headwidth 在点的箭头底座的宽度; headlength 点箭头的长度; shrink 总长度为分数“缩水”从两端; facecolor 箭头颜色
  - bbox 给标题增加外框 ，常用参数有：boxstyle 方框外形;

facecolor(简写 fc)背景颜色；edgecolor(简写 ec)边框线条颜色；edgewidth 边框线条大小



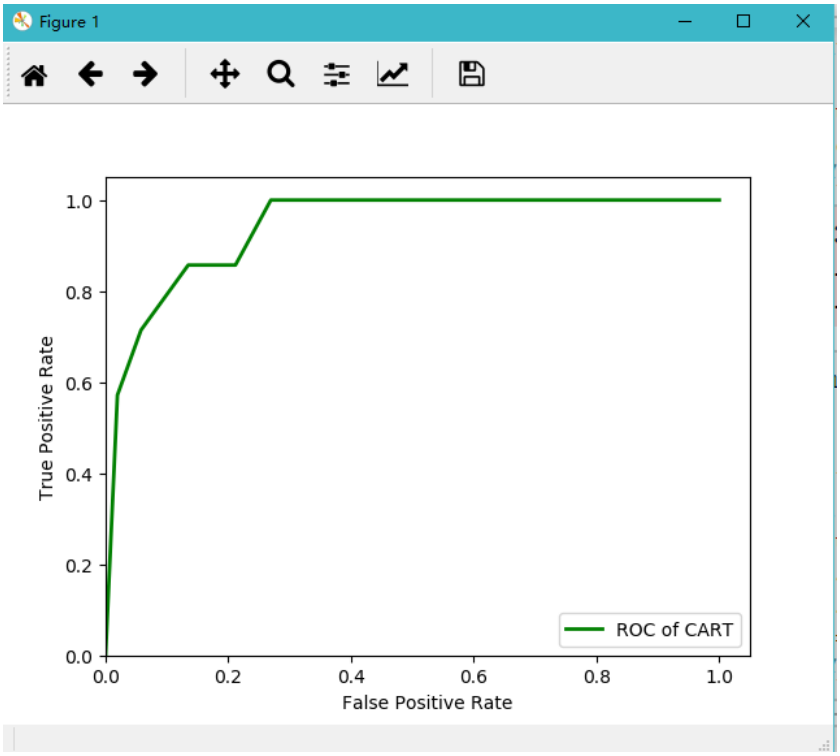
5) 导入 ROC 曲线函数，调用 roc_curve 函数得出真阳性率和假阳性率以及阈值，具体理解可以参考：

https://blog.csdn.net/u014264373/article/details/80487766

```
fpr, tpr, thresholds = roc_curve(test[:,3], tree.predict_proba(test[:,:3])[:,1],
pos_label=1)
plt.plot(fpr, tpr, linewidth=2, label = 'ROC of CART', color = 'green') #作出 ROC 曲线
```

roc_curve(y_true, y_score, pos_label=None, sample_weight=None, drop_intermediate=True)

- 参数解析
  - y_true:实际的样本标签值（这里只能用来处理二分类问题，即为{0，1}或者{true，false}，如果有多个标签，则需要使用 pos_label 指定某个标签为正例，其他的为反例）
  - y_score:目标分数，被分类器识别成正例的分数（常使用在 method="decision_function"、method="proba_predict"）
  - pos_label:指定某个标签为正例
  - sample_weight:样本的权重，可选择的
  - drop_intermediate: boolean, optional (default=True)，是否放弃一些不出现在绘制的 ROC 曲线上的次优阈值。这有助于创建更轻的 ROC 曲线



4. 使用Scikit-Learn库的neural_network的MLPClassifier函数及训练数据构建 LM 神经网络模型分别对训练和测试集进行分类，参考上一步得到模型正

1) 确率、混淆矩阵和 ROC 曲线图。实验中需要导入的包如下：

```
from sklearn.neural_network import MLPClassifier #导入神经网络模型
```

2) 建立模型

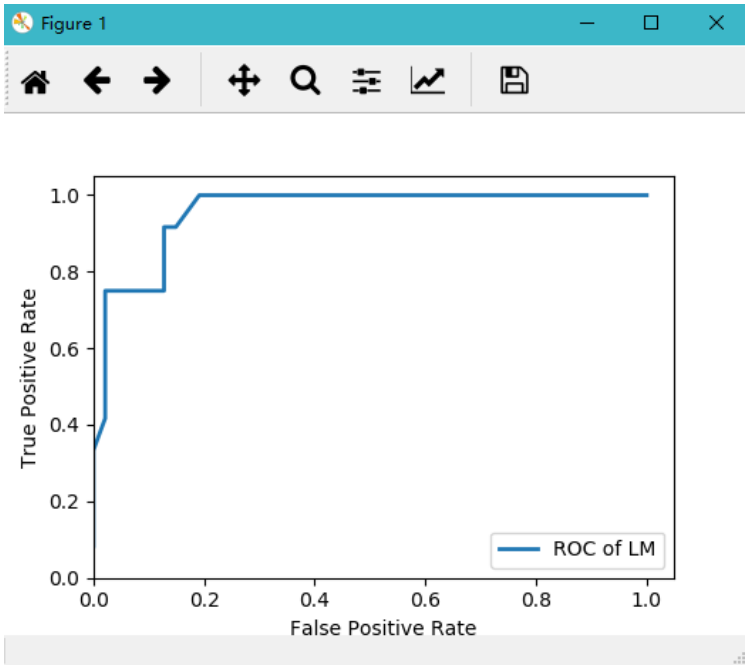
```
from sklearn.neural_network import MLPClassifier #导入神经网络函数

# 构建一个简单的神经网络，包含100个中间层节点，采用relu作为激活函数，函数具体用法参考
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

clf = MLPClassifier(max_iter=1000, hidden_layer_sizes=(100, ), activation='relu' ))

clf.fit(train[:,:3],train[:,3] # 拟合数据
```

3) 保存模型

```
joblib.dump(clf, 'train_model.m') # 保存模型
```

4) 画混淆矩阵图与导入 ROC 曲线图可以参考步骤 4，混淆矩阵如下所示：

ROC 曲线图如下所示：



5. 对比分析 CART 决策树模型和 LM 神经网络模型及其参数对数据处理结果的**影响**。