学习中的难点及解决方法

这次竞赛进行了多次尝试。

1. 首先，我观察数据发现训练集和测试集数据中有问号，但是其中有四列是没有有问号的，所以我第一次尝试就是，只提取其中没有问号的四列进行决策树训练、测试、验证，得到正确率是30%。
2. 第二次，我通过查阅资料将问号先通过data.replace(‘?’,np.nan,inplace=True),将问号改为NAN，然后在对NAN进行缺失值填充。
3. 我第一次对缺失值填充是该列的均值
4. 第二次对缺失值的填充是该列的众数
5. 然后我划分训练集为训练集和测试集，分别进行决策树，KNN，神经网络各种试验，生成CSV格式，提交。
6. 因为第一次是直接对原始数据进行训练，没有对数据进行更改，所以提交之后成功，但是后几次是先修改了数据，在提交，所以数据格式不正确，所以每次提交正确率都是0。然后我把训练集的预测值都改为了字符串类型。
7. 通过在kaggle网站观察数据，发现其中第6,9,11,12,13列数据摸一个数值出现频率极大，和其他数据有明显差异，数据不适合使用。
8. model = SelectFromModel(clf, prefit=True)

X\_new = model.transform(X)

通过代码测试，发现各列数据的重要程度是：[0.03951554 0.06141498 0.13340667 0.11424999 0.13348698 0.08063093 0.0264212 0.11025933 0.06537234 0.05404779 0.06530974 0.09048582 0.02539869]

其中第1,2,6,7,9,10,11,12,13列重要程度低。

1. 通过对数据排序之后观察，第六列数据中？太多，可以舍弃
2. 我尝试对某些列进行独热编码之后再处理，正确率还是不是很高。
3. 之后又进行神经网络的尝试，先是对缺失值用平均数进行填充，之后对平均数进行四舍五入，然后卡方检验选出重要特征，共选择9列[1,2,3,4,6,7,8,9,10]。然后实例化一个多层感知器，参数都是默认值，对数据进行训练，然后对验证集进行同样处理，带入训练器中得出结果。

核心代码分析：

首先导入库

import pandas as pd

import numpy as np

from sklearn.preprocessing import Imputer

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

from sklearn.neural\_network import MLPClassifier

之后对读取数据

col = (0,1,2,3,4,5,6,7,8,9,10,11,12,13)

data = pd.read\_csv('train.csv',names = col)

col2 = (0,1,2,3,4,5,6,7,8,9,10,11,12)

testdata = pd.read\_csv('test.csv',names = col2)

对数据中问号进行处理

先转化成nan

data.replace("?",np.nan,inplace = True)

testdata.replace("?",np.nan,inplace = True)

再对nan进行平均值填充，先实例化一个填充器

imp = Imputer(missing\_values = np.nan,strategy = 'mean')#平均数代替

newdata = imp.fit\_transform(data)

newtestdata = imp.fit\_transform(testdata)

再对数据进行四舍五入

newdata = round(pd.DataFrame(newdata))#四舍五入

newtestdata = round(pd.DataFrame(newtestdata))

之后将处理完的数据转为数组形式

newdata = np.array(newdata)#转化为数组

newtestdata = np.array(newtestdata)

#划分训练集，最后一列是标签

X = newdata[:,:-1]

#训练集X

y = newdata[:,-1:]

#训练集y

#单变量特征选择，卡方检验，检测出最优的特征

X\_new = SelectKBest(chi2, k=9).fit\_transform(X, y)

#对筛选之后的数据划分训练集和测试集

from sklearn.model\_selection import train\_test\_split

train\_X, test\_X, train\_y, test\_y = train\_test\_split(X\_new, y, test\_size=0.3)

#神经网络，实例化一个多层感知器

mlp = MLPClassifier()

对多层感知器进行训练

mlp.fit(train\_X,train\_y)

对测试集进行预测

pre\_y = mlp.predict(test\_X)

估算正确率

result = test\_y[:,0] - pre\_y

acc = 0

for i in range(1439):

if(result[i] == 0):

acc += 1

print(acc)

#对验证集进行处理

we = newtestdata[:,[1,2,3,4,6,7,8,9,10]]

对验证集进行预测

ew = mlp.predict(we)

#将数据改为int类型

ew1 = []

for i in range(len(ew)):

ew1.append(int(ew[i]))

将数据写入到CSV格式中

# test = pd.DataFrame(data = ew1,index = range(1,1799),columns=['y'])

# test.index.name = 'id'

# test.to\_csv('newfirst0321.csv')