*#coding:utf-8*

*# 极大似然估计 朴素贝叶斯算法*

**import** pandas **as** pd

**import** numpy **as** np

**class** NaiveBayes(object):

**def** getTrainSet(self):

dataSet = pd.read\_csv(**'D://2\_Teaching//Leature\_AI//Dataset//naivebayes\_data.csv'**)

dataSetNP = np.array(dataSet) *#将数据由dataframe类型转换为数组类型*

trainData = dataSetNP[:,0:dataSetNP.shape[1]-1] *#训练数据x1,x2*

labels = dataSetNP[:,dataSetNP.shape[1]-1] *#训练数据所对应的所属类型Y*

**return** trainData, labels

**def** classify(self, trainData, labels, features):

*#求labels中每个label的先验概率*

labels = list(labels) *#转换为list类型*

P\_y = {} *#存入label的概率*

**for** label **in** labels:

P\_y[label] = labels.count(label)/float(len(labels)) *# p = count(y) / count(Y)*

*#求label与feature同时发生的概率*

P\_xy = {}

**for** y **in** P\_y.keys():

y\_index = [i **for** i, label **in** enumerate(labels) **if** label == y] *# labels中出现y值的所有数值的下标索引*

**for** j **in** range(len(features)): *# features[0] 在trainData[:,0]中出现的值的所有下标索引*

x\_index = [i **for** i, feature **in** enumerate(trainData[:,j]) **if** feature == features[j]]

xy\_count = len(set(x\_index) & set(y\_index)) *# set(x\_index)&set(y\_index)列出两个表相同的元素*

pkey = str(features[j]) + **'\*'** + str(y)

P\_xy[pkey] = xy\_count / float(len(labels))

*#求条件概率*

P = {}

**for** y **in** P\_y.keys():

**for** x **in** features:

pkey = str(x) + **'|'** + str(y)

P[pkey] = P\_xy[str(x)+**'\*'**+str(y)] / float(P\_y[y]) *#P[X1/Y] = P[X1Y]/P[Y]*

*#求[2,'S']所属类别*

F = {} *#[2,'S']属于各个类别的概率*

**for** y **in** P\_y:

F[y] = P\_y[y]

**for** x **in** features:

F[y] = F[y]\*P[str(x)+**'|'**+str(y)] *#P[y/X] = P[X/y]\*P[y]/P[X]，分母相等，比较分子即可，所以有F=P[X/y]\*P[y]=P[x1/Y]\*P[x2/Y]\*P[y]*

features\_label = max(F, key=F.get) *#概率最大值对应的类别*

**return** features\_label

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

nb = NaiveBayes()

*# 训练数据*

trainData, labels = nb.getTrainSet()

*# x1,x2*

features = [3,**'S'**]

*# 该特征应属于哪一类*

result = nb.classify(trainData, labels, features)

print (features,**'属于'**,result)