**暨南大学本科实验报告专用纸**

课程名称 人工智能原理 成绩评定

实验项目名称 多标签学习 指导教师 张佳

实验项目编号 实验项目类型 综合 实验地点 线上

学生姓名 陈宇 学号 2020101642

学院 信息科学技术学院 系 计算机系 专业 软件工程

实验时间2022 年 11月11日下午～9月20日 下午 温度 ℃湿度

# 实验内容

学习和了解多标签学习的原理，尝试使用多标签分类器来进行模型训练，并且能够对数据进行预测和模型性能评估。同时在实验中，要掌握对数据集的拆分，将训练集和测试集的数据进行一定的处理。在实验任务2中，使用多种多标签分类器进行训练，并且对比算法性能；在任务3中，对dataset这个csv文件的标签和数据进行采集，并且尝试自己动手训练模型，观察训练结果。

# 核心代码

任务1+任务2：main.py:

import pandas

from scipy.io import arff

from scipy import sparse

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from skmultilearn.adapt import BRkNNaClassifier

from skmultilearn.adapt import BRkNNbClassifier

from skmultilearn.adapt import MLkNN

from skmultilearn.adapt import MLARAM

from skmultilearn.adapt import MLTSVM

import sklearn.metrics as metrics

#load data

data, meta = arff.loadarff('yeast/yeast-train.arff')   #load data set

data = pandas.DataFrame(data)    #将数据集转换成dataframe

print(data.head())     #查看前几行

#operate on data set

X = data.iloc[:, 0:103].values # 取出训练数据

y = data.iloc[:, 103:117].values #取出标签

# 数据归一化

std = StandardScaler()

X = std.fit\_transform(X)

# 划分训练集和测试集 比例为 4:1

trainx, testx, trainy, testy = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

trainy = trainy.astype(np.float64)

testy = testy.astype(np.float64)

#train model

classifier = BRkNNaClassifier()

# train

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

#任务一:使用下载的数据集，yeast-traint.arff 全部数据做训练集，使用 yeast-test.arff 全部数据做测试集，查看分类结果

print("任务一：")

#load data

data, meta = arff.loadarff('yeast/yeast-train.arff')   #load data set

data = pandas.DataFrame(data)    #将数据集转换成dataframe

# print(data.head())     #查看前几行

#operate on data set

X = data.iloc[:, 0:103].values # 取出训练数据

y = data.iloc[:, 103:117].values #取出标签

# 数据归一化

std = StandardScaler()

X = std.fit\_transform(X)

trainx = X

trainy = y

data, meta = arff.loadarff('yeast/yeast-test.arff')   #load data set

data = pandas.DataFrame(data)    #将数据集转换成dataframe

# print(data.head())     #查看前几行

#operate on data set

X = data.iloc[:, 0:103].values # 取出训练数据

y = data.iloc[:, 103:117].values #取出标签

# 数据归一化

std = StandardScaler()

X = std.fit\_transform(X)

testx = X

testy = y

trainy = trainy.astype(np.float64)

testy = testy.astype(np.float64)

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

print('--------------------------------------')

#任务2：尝试使用 scikit-multilearn: Multi-Label Classification in Python — Multi-Label Classification for Python 下的不同多标签分类算法进行分类，对比不同算法性能

print("对于 BRkNNbClassifier分类器：")

classifier = BRkNNbClassifier()

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

print('--------------------------------------')

print("对于 MlkNN分类器：")

trainy = trainy.astype(np.int64)

testy = testy.astype(np.int64)

classifier = MLkNN()

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

print('--------------------------------------')

print("对于 MLARAm分类器：")

classifier = MLARAM()

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

# predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

print('--------------------------------------')

print("对于 MLTSVM分类器：")

classifier = MLTSVM()

trainx = sparse.csr\_matrix(trainx)

trainy = sparse.csr\_matrix(trainy)

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

# predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

print(metrics.average\_precision\_score(testy, predictions))

任务3：使用brknna分类器对dataset进行训练模型

Tast3文件如下：

import numpy as np

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

from sklearn.preprocessing import StandardScaler

from skmultilearn.adapt import BRkNNaClassifier

import sklearn.metrics as metrics

import pandas as pd

#任务3：任务 3：尝试 dataset.csv 数据集，运行出结果。

# 其中标签为 imageDiseaseCode

# commonTzPhz commonTzQxz commonTzYxz commonTzYixz

# commonTzTsz commonTzSrz commonTzXyz commonTzQyz

# commonTzTbz，

csv\_file = "dataset.csv"

csv\_data = pd.read\_csv(csv\_file, low\_memory = False)#防止弹出警告

data = pd.DataFrame(csv\_data)

# data, meta = arff.loadarff('yeast/yeast-train.arff')   #load data set

# data = pandas.DataFrame(data)    #将数据集转换成dataframe

print(data.head())     #查看前几行

#operate on data set

X = pd.concat([data.iloc[:, 1:9], data.iloc[:, 18:128]],axis=1).values # 取出训练数据

y = pd.concat([data.iloc[:, 0], data.iloc[:, 10:18]],axis=1).values #取出标签

# print(X.head())

# print('----------')

# print(y.head())

# 数据归一化

std = StandardScaler()

X = std.fit\_transform(X)

# 划分训练集和测试集 比例为 4:1

trainx, testx, trainy, testy = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

trainy = trainy.astype(np.float64)

testy = testy.astype(np.float64)

#train model

classifier = BRkNNaClassifier()

# train

classifier.fit(trainx, trainy)

# predict

predictions = classifier.predict(testx)

predictions = predictions.toarray()

# 以下值越小越好

print(metrics.hamming\_loss(testy, predictions))

print(metrics.zero\_one\_loss(testy, predictions))

print(metrics.coverage\_error(testy, predictions))

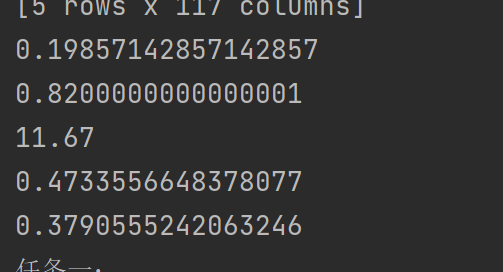
print(metrics.label\_ranking\_loss(testy, predictions))

# 该值越大越好

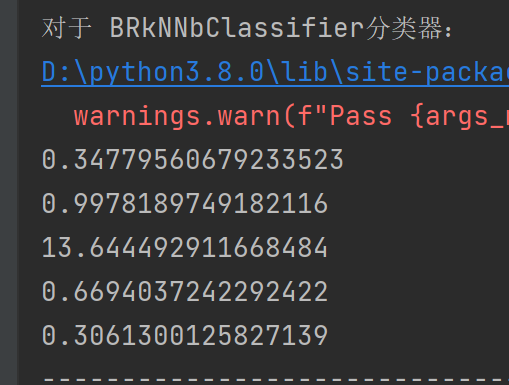
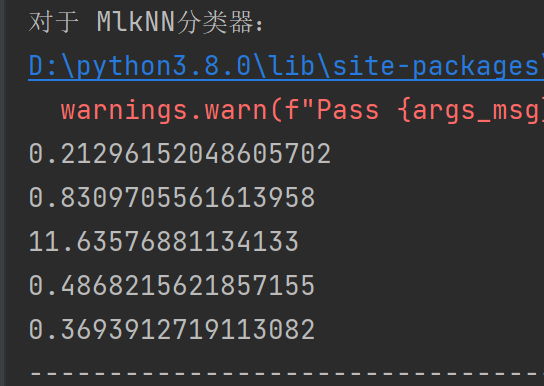
print(metrics.average\_precision\_score(testy, predictions))

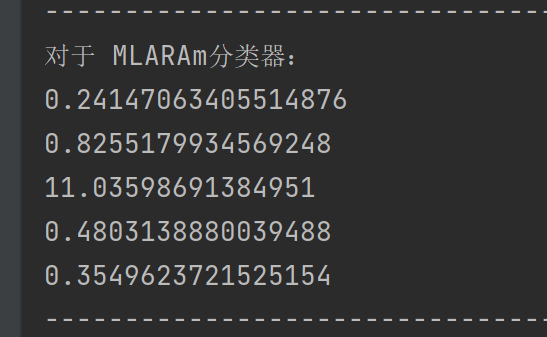
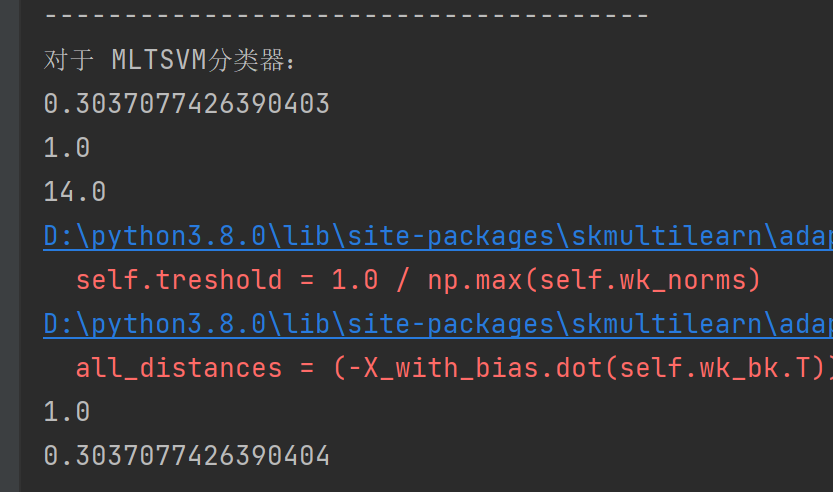
# 实验结果

任务一：

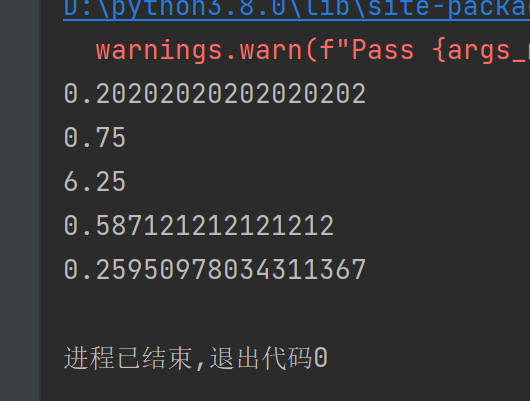


任务2：

任务3：



# 实验总结

* 在任务1中：与原数据结果对比：hammin\_loss(海明损失)：错误标签比例，01分类损失（zero\_one\_loss），覆盖误差（coverage\_error）都有一定程度减少，而label\_ranking\_loss稍微上升：说明当训练集数据越大时，模型训练效果越好。
* 在任务2中：对brknna，brknnb，mlknn，mlaram，mltsvm五个分类器对同一组数据进行训练，效果最好的是brknnb和mlknn分类器，最差运行时间最久的是mltsvm
* 在任务3中：brknna分类器分类效果一般。