**实验二 手写体数字识别**

**一、实验目的**

通过实验基本理解常用的分类模型，掌握scikit-learn编程库中分类模块的使用，并能用classification\_report对分类结果进行评价。

**二、实验步骤**

1、用load\_digits导入数据；

2、用train\_test\_split拆分数据；

3、用StandardScaler对数据标准化；

4、分别用LogisticRegression, SGDClassifier, LinearSVC, SVC, NuSVC, KNeighborsClassifier, GaussianNB, BernoulliNB, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier模型训练模型，对测试集训练及预测；

5、用classification\_report分别对各模块的预测结果进行评价。

**三、实验代码**

# 数据读取  
# 从sklearn.datasets里导入手写体数字加载器  
from sklearn.datasets import load\_digits

X, y = load\_digits(return\_X\_y=True)  
print(X.shape)  
print(y.shape)

# 从通过数据加载器获得手写体数字的数码图像数据，并储存在digits变量中  
digits = load\_digits()  
# 监视数据规模和特征维度  
print(digits.data.shape)

import matplotlib.pyplot as plt  
plt.figure()  
plt.matshow(digits.images[100])  
print(digits.images[100])  
plt.show()

# 数据分割  
# 从sklearn.model\_selection中导入rain\_test\_split用于分割数据  
from sklearn.model\_selection import train\_test\_split

# 随机选取75%的数据作为训练样本；其余25%的数据作为测试样本  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)  
print(y\_train.shape)  
print(y\_test.shape)

# 数据标准化  
# 从sklearn.preprocessing里导入数据标准化模块  
from sklearn.preprocessing import StandardScaler  
  
ss = StandardScaler() # 仍然需要对训练和测试的特征数据进行标准化  
X\_train = ss.fit\_transform(X\_train)  
X\_test = ss.transform(X\_test)

# LogisticRegression

# 数字识别及预测  
# 从sklearn.linear\_model里导入LogisticRegression  
from sklearn.linear\_model import LogisticRegression  
  
# 初始化  
lgr = LogisticRegression(multi\_class='auto', solver='liblinear')  
# 进行模型训练  
lgr.fit(X\_train, y\_train)  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = lgr.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of LogisticRegression is:', lgr.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# SGDClassifier

# 数字识别及预测  
# 从sklearn.linear\_model里导入SGDClassifier  
from sklearn.linear\_model import SGDClassifier  
  
# 初始化  
sgdc = SGDClassifier()  
# 进行模型训练  
sgdc.fit(X\_train, y\_train)  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = sgdc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of SGDClassifier is:', sgdc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# LinearSVC

# 数字识别及预测  
# 从sklearn.svm里导入基于线性假设的支持向量机分类器LinearSVC  
from sklearn.svm import LinearSVC  
  
# 初始化线性假设的支持向量机分类器LinearSVC  
lsvc = LinearSVC(max\_iter=10000)  
# 进行模型训练  
lsvc.fit(X\_train, y\_train)  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = lsvc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of Linear SVC is:', lsvc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# SVC

# 数字识别及预测  
# 从sklearn.svm里导入支持向量机分类器SVC。  
from sklearn.svm import SVC  
  
# 初始化支持向量机分类器SVC。  
svc = SVC()  
  
# 进行模型训练  
svc.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中。  
y\_predict = svc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评。  
from sklearn.metrics import classification\_report  
print('The Accuracy of SVC is', svc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做  
# 更加详细的分析  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# NuSVC

# 数字识别及预测  
# 从sklearn.svm里导入核支持向量机分类器NuSVC。  
from sklearn.svm import NuSVC  
  
# 初始化核支持向量机分类器NuSVC。  
nusvc = NuSVC()  
  
# 进行模型训练  
nusvc.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中。  
y\_predict = nusvc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评。  
from sklearn.metrics import classification\_report  
print('The Accuracy of NuSVC is', nusvc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做  
# 更加详细的分析  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# KNeighborsClassifier

# 数字识别及预测  
# 从sklearn.neighnors里导入KNeighborsClassifier  
from sklearn.neighbors import KNeighborsClassifier  
  
# 初始化KNeighborsClassifier  
neighbor = KNeighborsClassifier()  
# 进行模型训练  
neighbor.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = neighbor.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of KNeighborsClassifier is:',  
 neighbor.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# GaussianNB

# 数字识别及预测  
# 从sklearn.naive\_bayes里导入GaussianNB  
from sklearn.naive\_bayes import GaussianNB  
  
# 初始化GaussianNB  
gsbayes = GaussianNB()  
# 进行模型训练  
gsbayes.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = gsbayes.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of GaussianNB is:', gsbayes.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# BernoulliNB

# 数字识别及预测  
# 从sklearn.naive\_bayes里导入BernoulliNB  
from sklearn.naive\_bayes import BernoulliNB  
  
# 初始化BernoulliNB  
bnlbayes = BernoulliNB()  
# 进行模型训练  
bnlbayes.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = bnlbayes.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of BernoulliNB is:', bnlbayes.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# DecisionTreeClassifier

# 数字识别及预测  
# 从sklearn.tree里导入DecisionTreeClassifier  
from sklearn.tree import DecisionTreeClassifier  
  
# 初始化DecisionTreeClassifier  
dtc = DecisionTreeClassifier()  
# 进行模型训练  
dtc.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = dtc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of DecisionTreeClassifier is:', dtc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# RandomForestClassifier

# 数字识别及预测  
# 从sklearn.ensemble里导入RandomForestClassifier  
from sklearn.ensemble import RandomForestClassifier  
  
# 初始化RandomForestClassifier  
rfc = RandomForestClassifier()  
# 进行模型训练  
rfc.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = rfc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of RandomForestClassifier is:', rfc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

# GradientBoostingClassifier

# 数字识别及预测  
# 从sklearn.ensemble里导入GradientBoostingClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
  
# 初始化GradientBoostingClassifier  
gbc = GradientBoostingClassifier()  
# 进行模型训练  
gbc.fit(X\_train, y\_train)  
  
# 利用训练好的模型对测试样本的数字类别进行预测，  
# 预测结果储存在变量y\_predict中  
y\_predict = gbc.predict(X\_test)

# 性能分析  
# 使用模型自带的评估函数进行准确性测评  
from sklearn.metrics import classification\_report  
print('The Accuracy of GradientBoostingClassifier is:',  
 gbc.score(X\_test, y\_test))  
  
# 依然使用sklearn.metrics里面的classification\_report模块对预测结果做更加详细的分析  
  
print(  
 classification\_report(y\_test,  
 y\_predict,  
 target\_names=digits.target\_names.astype(str)))

**四、实验体会**