**一、简述朴素贝叶斯**的原理，三类朴素贝叶斯的比较、实现的关键操作点（不超过300字，逻辑要通，**要有小标题**，排版清晰，请不要网络大幅摘抄）

**一、原理**

朴素贝叶斯是基于贝叶斯定理与特征条件独立假设的分类算法。利用训练数据计算类别与特征的联合概率分布，然后根据特征的条件独立性假设，通过贝叶斯公式计算后验概率，最终选择具有最大后验概率的类别作为预测结果。 二、**三类朴素贝叶斯的比较** 1.高斯朴素贝叶斯（GaussianNB）：假设特征的先验概率符合高斯分布，适用于连续型数据。 2.多项式朴素贝叶斯（MultinomialNB）：适用于离散特征的分类问题，如文本分类。 3.伯努利朴素贝叶斯（BernoulliNB）：处理布尔型特征的分类问题，适用于文本分类等任务。 **三、关键操作点**

1.数据预处理：特征选择、数据清洗和特征转换。 2.概率计算：计算类别的先验概率和特征的条件概率。 3.后验概率估计：利用贝叶斯公式计算后验概率。 4.参数估计：对于高斯朴素贝叶斯，需要估计均值和方差；对于多项式和伯努利朴素贝叶斯，需要估计特征出现的概率。

**二、**概述利用**朴素贝叶斯**判断**蘑菇是否有毒**的技术思路**。**

利用朴素贝叶斯判断蘑菇是否有毒的技术思路如下：

**1.数据收集和预处理：**收集包含蘑菇特征与分类（有毒/无毒）的数据集。对数据进行预处理，包括处理缺失值、标准化数据等。**2.特征选择和提取：**从数据集中选择合适的特征来训练模型。对于蘑菇有毒分类问题，可能涉及到蘑菇的外观、气味、孢子颜色等特征。**3.模型训练：**使用朴素贝叶斯算法进行模型训练。根据训练集中的特征和标签，计算类别的先验概率和特征的条件概率。**4.模型评估：**使用测试集评估模型的性能，计算准确率、精确率、召回率等指标。**5.预测和应用：**使用经过训练的模型对新的蘑菇样本进行分类预测。基于先验概率和条件概率，计算样本属于有毒或无毒类别的后验概率，并选择具有较大后验概率的类别作为预测结果。大致演示代码如下：

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
from sklearn.model\_selection import learning\_curve  
from sklearn.model\_selection import ShuffleSplit  
from sklearn.naive\_bayes import GaussianNB, MultinomialNB  
from sklearn.model\_selection import train\_test\_split  
  
mushroom = pd.read\_csv("mushrooms.csv")  
mushroom.head()  
mushroom.isnull().sum()  
mushroom['class'].unique()  
from sklearn.preprocessing import LabelEncoder  
labelencoder = LabelEncoder()  
for col in mushroom.columns:  
 mushroom[col] = labelencoder.fit\_transform(mushroom[col])  
mushroom.head()  
x = mushroom.iloc[:,1:23]  
y = mushroom.iloc[:,0]  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state = 42)  
mnb = MultinomialNB()  
mnb.fit(x\_train,y\_train)  
print("训练集准确率：",mnb.score(x\_train,y\_train))  
print("测试集准确率：",mnb.score(x\_test,y\_test))

def plot\_learning\_curve(estimator,title,x,y,ylim = None,cv = None,n\_jobs = -1,train\_sizes = np.linspace(.1,1.0,5)):  
 plt.figure()  
 plt.title(title)  
 if ylim is not None:  
 plt.ylim(\*ylim)  
 plt.xlabel("Training examples")  
 plt.ylabel("Score")  
 train\_sizes,train\_scores,test\_scores = learning\_curve(estimator,x,y,cv = cv,n\_jobs = n\_jobs,train\_sizes = train\_sizes)  
 train\_scores\_mean = np.mean(train\_scores,axis = 1)  
 test\_scores\_mean = np.mean(test\_scores,axis = 1)  
 plt.grid()  
 plt.plot(train\_sizes,train\_scores\_mean,'o-',color = "r",label = "Training score")  
 plt.plot(train\_sizes,test\_scores\_mean,'o-',color="g",label = "Cross-validation score")  
 plt.legend(loc = "lower right")  
 return plt  
cv = ShuffleSplit(n\_splits = 30,test\_size = 0.3,random\_state = 28)  
estimators = [MultinomialNB(),GaussianNB()]  
for estimator in estimators:  
 title = estimator  
 plot\_learning\_curve(estimator,title,x,y,ylim = (0.5,1.0),cv = cv,n\_jobs = -1)  
plt.show()

**结果：**

文本

描述已自动生成

图表, 折线图

描述已自动生成

**三、利用朴素贝叶斯解决问题（预测或分类）。**

附：Sklearn常用数据集

|  |  |  |
| --- | --- | --- |
| 序号 | 数据集 | 学号尾号 |
| 1 | 鸢尾花数据集：load\_iris（） | 0 1 |
| 2 | 手写数字数据集：load\_digits（） | 2 3 |
| 3 | 乳腺癌数据集load-breast-cancer（） | 4 5 8 |
| 4 | 糖尿病数据集：load-diabetes（） | 6 7 9 |

利用朴素贝叶斯对以上数据集（学号尾号不同采用不同数据集）进行分析，找出各模型（贝努力、高斯、多项式）的朴素贝叶斯，哪个适合本数据集（准确率更高，不同模型下的准确率要列出）。

要求：

1）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

2）运行结果要截图，结果要文字解释；

3）注意排版

from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB, BernoulliNB, MultinomialNB  
from sklearn.metrics import accuracy\_score  
  
iris = datasets.load\_iris()  
X = iris.data  
y = iris.target  
  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# 初始化并拟合不同类型的朴素贝叶斯模型  
gnb = GaussianNB()  
gnb.fit(X\_train, y\_train)  
y\_pred\_gnb = gnb.predict(X\_test)  
accuracy\_gnb = accuracy\_score(y\_test, y\_pred\_gnb)  
  
bnb = BernoulliNB()  
bnb.fit(X\_train, y\_train)  
y\_pred\_bnb = bnb.predict(X\_test)  
accuracy\_bnb = accuracy\_score(y\_test, y\_pred\_bnb)  
  
mnb = MultinomialNB()  
mnb.fit(X\_train, y\_train)  
y\_pred\_mnb = mnb.predict(X\_test)  
accuracy\_mnb = accuracy\_score(y\_test, y\_pred\_mnb)  
  
print("高斯模型准确率:", accuracy\_gnb)  
print("伯努力模型准确率:", accuracy\_bnb)  
print("多项式模型准确率:", accuracy\_mnb)  
  
import matplotlib.pyplot as plt  
import numpy as np  
# 选择前两个特征  
X = iris.data[:, :2]  
y = iris.target  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
# 初始化并拟合高斯朴素贝叶斯模型  
gnb = GaussianNB()  
gnb.fit(X\_train, y\_train)  
# 绘制决策边界  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01),np.arange(y\_min, y\_max, 0.01))  
Z = gnb.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
plt.contourf(xx, yy, Z, alpha=0.8)  
# 绘制训练集样本点  
plt.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=plt.cm.Pastel1, edgecolor='k', label="Train Set")  
# 绘制测试集样本点  
plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test, cmap=plt.cm.Pastel1, marker='x', edgecolor='k', label="Test Set")  
plt.xlabel('Sepal Length')  
plt.ylabel('Sepal Width')  
plt.title('Gaussian Naive Bayes Decision Boundary')  
plt.legend()  
plt.show()

**结果：**

**文本

描述已自动生成**

**高斯模型准确率最高，可看出高斯朴素贝叶斯更适合本数据集。**

图表, 散点图

描述已自动生成

**四、[附加题].**利用朴素贝叶斯解决**自己遇到的问题**。

[解题要求同题2]

**解决红酒数据集**

from sklearn.datasets import load\_wine  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB, BernoulliNB, MultinomialNB  
from sklearn.metrics import accuracy\_score  
import matplotlib.pyplot as plt  
import numpy as np  
  
wine = load\_wine()  
X = wine.data  
y = wine.target  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# 初始化并拟合不同类型的朴素贝叶斯模型  
gnb = GaussianNB()  
gnb.fit(X\_train, y\_train)  
y\_pred\_gnb = gnb.predict(X\_test)  
accuracy\_gnb = accuracy\_score(y\_test, y\_pred\_gnb)  
  
bnb = BernoulliNB()  
bnb.fit(X\_train, y\_train)  
y\_pred\_bnb = bnb.predict(X\_test)  
accuracy\_bnb = accuracy\_score(y\_test, y\_pred\_bnb)  
  
mnb = MultinomialNB()  
mnb.fit(X\_train, y\_train)  
y\_pred\_mnb = mnb.predict(X\_test)  
accuracy\_mnb = accuracy\_score(y\_test, y\_pred\_mnb)  
print("高斯朴素贝叶斯准确率:", accuracy\_gnb)  
print("伯努利朴素贝叶斯准确率:", accuracy\_bnb)  
print("多项式朴素贝叶斯准确率:", accuracy\_mnb)  
  
# 选择前两个特征  
X = wine.data[:, :2]  
# 初始化并拟合高斯朴素贝叶斯模型  
gnb = GaussianNB()  
gnb.fit(X, y)  
# 绘制决策边界  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01),np.arange(y\_min, y\_max, 0.01))  
Z = gnb.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
plt.contourf(xx, yy, Z, alpha=0.8)  
# 绘制样本点  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Pastel1, edgecolor='k')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()

**结果：高斯朴素贝叶斯准确率最高，更适合红酒数据集。**

文本

描述已自动生成

图表

描述已自动生成