山西大学计算机与信息技术学院

**实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓 名 | 李京 | 学 号 | 202202501110 | 专业班级 | 计算机科学与技术2202 | |
| 课程名称 | 人工智能 | | | | 实验日期 | 2024.10.28 |
| 成 绩 |  | 指导教师 | 李琳 | | 批改日期 |  |
| 实验名称 | | 实验四 机器学习 | | | | |
| 一、实验目的：   1. 通过本实验，学习决策树和神经网络。 2. 构建企鹅分类和乳腺癌预测模型。   二、实验内容：   1. **决策树方法：**使用决策树算法构建企鹅分类模型，包括数据预处理、模型训练、预测及评估。 2. **神经网络方法：**利用BP神经网络构建乳腺癌预测模型，涵盖数据分割、模型搭建、训练、预测和结果分析。   三、实验平台：  MacBook Air M3  Microsoft Visual Studio Code  Mamba environment + python 3.12.6  四、实验步骤：  **决策树方法实验步骤：**   1. **导入必要的库**：    * 导入数据处理库（NumPy, Pandas）。    * 导入绘图库（Matplotlib, Seaborn）。    * 导入决策树相关的库（sklearn.tree, sklearn.model\_selection）。 2. **数据准备**：    * 使用Pandas的read\_csv函数读取企鹅数据集，并转化为DataFrame格式。    * 选择部分特征以简化模型。 3. **数据预处理**：    * 使用.info()查看数据整体信息。    * 使用.head()和.tail()查看数据的头部和尾部。    * 用.fillna(-1)处理缺失值。    * 将类别标签转化为数字。 4. **数据划分**：    * 选择类别为0和1的样本，忽略类别为2的样本。    * 使用train\_test\_split函数将数据划分为训练集和测试集。 5. **模型训练**：    * 从sklearn中导入决策树模型。    * 定义决策树模型，并设置参数以防止过拟合。    * 使用训练集数据训练模型。 6. **模型评估**：    * 使用训练好的模型在训练集和测试集上进行预测。    * 使用metrics.accuracy\_score评估模型的准确度。    * 使用confusion\_matrix查看混淆矩阵。    * 使用热力图对混淆矩阵结果进行可视化。 7. **模型可视化**：    * 使用Graphviz库将决策树模型可视化并保存。   # %%  ## 基础函数库  import numpy as np  import pandas as pd  import graphviz  ## 绘图函数库  import matplotlib.pyplot as plt  import seaborn as sns  ## 我们利用 Pandas 自带的 read\_csv 函数读取并转化为 DataFrame 格式，这里要注意修改一下你自己文件的位置，要不然导入不进来  data = pd.read\_csv('./penguins\_raw.csv')  ## 为了方便我们仅选取四个简单的特征，有兴趣的同学可以研究下其他特征的含义以及使用方法  data = data[['Species','Culmen Length (mm)','Culmen Depth (mm)','Flipper Length (mm)','Body Mass (g)']]  # %%  ## 利用.info()查看数据的整体信息  data.info()  ## 进行简单的数据查看，我们可以利用 .head() 头部.tail()尾部  data.head()  data = data.fillna(-1)  data.tail()  ## 其对应的类别标签为'Adelie Penguin', 'Gentoo penguin', 'Chinstrap penguin'三种不同企鹅的类别。  data['Species'].unique()  '''为了方便我们将标签转化为数字  'Adelie Penguin (Pygoscelis adeliae)' ------0  'Gentoo penguin (Pygoscelis papua)' ------1  'Chinstrap penguin (Pygoscelis antarctica) ------2  '''  def trans(*x*):  if *x* == data['Species'].unique()[0]:  return 0  if *x* == data['Species'].unique()[1]:  return 1  if *x* == data['Species'].unique()[2]:  return 2  data['Species'] = data['Species'].apply(trans)  # %%  ## 为了正确评估模型性能，将数据划分为训练集和测试集，并在训练集上训练模型，在测试集上验证模型性能。  from sklearn.model\_selection import train\_test\_split  ## 选择其类别为 0 和 1 的样本 （不包括类别为 2 的样本）  data\_target\_part = data[data['Species'].isin([0,1])][['Species']]  data\_features\_part = data[data['Species'].isin([0,1])][['Species','Culmen Length (mm)','Culmen Depth (mm)','Flipper Length (mm)','Body Mass (g)']]  ## 测试集大小为 20%， 80%/20%分（请补全缺失代码）  # 测试集大小为 20%， 80%/20%分（请补全缺失代码）  from sklearn.model\_selection import train\_test\_split  x\_train, x\_test, y\_train, y\_test = train\_test\_split(  data\_features\_part,  data\_target\_part,  *test\_size*=0.2,  *random\_state*=42,  *stratify*=data\_target\_part # 确保类别分布一致  )  ## 从 sklearn 中导入决策树模型  from sklearn.tree import DecisionTreeClassifier  from sklearn import tree  ## 定义 决策树模型  clf = DecisionTreeClassifier(*criterion*='entropy')  # 在训练集上训练决策树模型（请补全缺失代码）  # 定义 决策树模型，添加减枝参数  clf = DecisionTreeClassifier(  *criterion*='entropy',  *max\_depth*=5, # 设置树的最大深度  *min\_samples\_split*=10, # 设置内部节点再划分所需最小样本数  *min\_samples\_leaf*=5 # 设置叶子节点最小样本数  )  # 在训练集上训练决策树模型（请补全这句缺失的代码）  clf.fit(x\_train, y\_train)  # %%  import graphviz  dot\_data = tree.export\_graphviz(clf, *out\_file*=None)  graph = graphviz.Source(dot\_data)  graph.render("penguins")  ## 在训练集和测试集上分布利用训练好的模型进行预测  train\_predict = clf.predict(x\_train)  test\_predict = clf.predict(x\_test)  from sklearn import metrics  ## 利用 accuracy（准确度）【预测正确的样本数目占总预测样本数目的比例】  # 评估模型效果  print('The accuracy of the Logistic Regression is:',metrics.accuracy\_score(y\_train,train\_predict))  print('The accuracy of the Logistic Regression is:',metrics.accuracy\_score(y\_test,test\_predict))  ## 查看混淆矩阵 (预测值和真实值的各类情况统计矩阵)  confusion\_matrix\_result = metrics.confusion\_matrix(test\_predict,y\_test)  print('The confusion matrix result:\n',confusion\_matrix\_result)  # 利用热力图对于结果进行可视化  plt.figure(*figsize*=(8, 6))  sns.heatmap(confusion\_matrix\_result, *annot*=True, *cmap*='Blues')  plt.xlabel('Predicted labels')  plt.ylabel('True labels')  plt.show()  **神经网络方法实验步骤：**   1. **导入必要的库**：    * 导入乳腺癌数据集。    * 导入BP神经网络模型。    * 导入数据预处理和模型评估相关的库。 2. **数据准备**：    * 从sklearn.datasets中导入乳腺癌数据集。 3. **数据预处理**：    * 将数据分割为训练集和测试集。    * 使用StandardScaler进行数据标准化。 4. **模型搭建**：    * 建立BP神经网络模型，设置隐藏层大小、激活函数、优化器和最大迭代次数。 5. **模型训练**：    * 使用训练集数据训练BP神经网络模型。 6. **模型预测与可视化**：    * 使用训练好的模型进行预测。    * 可视化真实数据和预测数据。 7. **模型评估**：    * 显示预测分数。    * 打印分类报告，包括准确率、召回率和F1分数。    * 绘制混淆矩阵并进行可视化。   # %%  # 导入乳腺癌数据集  from sklearn.datasets import load\_breast\_cancer  # 导入 BP 模型  from sklearn.neural\_network import MLPClassifier  # 导入训练集分割方法  from sklearn.model\_selection import train\_test\_split  # 导入预测指标计算函数和混淆矩阵计算函数  from sklearn.metrics import classification\_report, confusion\_matrix  # 导入绘图包  import matplotlib.pyplot as plt  import seaborn as sns  from mpl\_toolkits.mplot3d import Axes3D  from sklearn.preprocessing import StandardScaler  # 导入乳腺癌数据集  cancer = load\_breast\_cancer()  # %%  # 分割数据为训练集和测试集  cancer\_data = cancer['data']  print('cancer\_data 数据维度为：', cancer\_data.shape)  cancer\_target = cancer['target']  print('cancer\_target 标签维度为：', cancer\_target.shape)  cancer\_names = cancer['feature\_names']  cancer\_desc = cancer['DESCR']  # 分为训练集与测试集  cancer\_data\_train, cancer\_data\_test, cancer\_target\_train, cancer\_target\_test = train\_test\_split(  cancer\_data, cancer\_target, *test\_size*=0.2, *random\_state*=42  )  # 数据标准化  scaler = StandardScaler()  cancer\_data\_train = scaler.fit\_transform(cancer\_data\_train)  cancer\_data\_test = scaler.transform(cancer\_data\_test)  # %%  # 建立 BP 模型, 采用 Adam 优化器，relu 非线性映射函数  BP = MLPClassifier(  *hidden\_layer\_sizes*=(100,), # 隐藏层大小，可以根据需要调整  *activation*='relu', # 激活函数  *solver*='adam', # 优化器  *max\_iter*=1000, # 最大迭代次数，确保收敛  *random\_state*=42  )  # 进行模型训练  BP.fit(cancer\_data\_train, cancer\_target\_train)  # %%  # 进行模型预测  predict\_train\_labels = BP.predict(cancer\_data\_train)  # 可视化真实数据  fig = plt.figure()  ax = fig.add\_subplot(111, *projection*='3d')  scatter = ax.scatter(  cancer\_data\_train[:, 0],  cancer\_data\_train[:, 1],  cancer\_data\_train[:, 2],  *marker*='o',  *c*=cancer\_target\_train,  *cmap*='viridis',  *alpha*=0.7  )  plt.title('True Label Map')  plt.show()  # 可视化预测数据  fig = plt.figure()  ax = fig.add\_subplot(111, *projection*='3d')  scatter = ax.scatter(  cancer\_data\_train[:, 0],  cancer\_data\_train[:, 1],  cancer\_data\_train[:, 2],  *marker*='o',  *c*=predict\_train\_labels,  *cmap*='viridis',  *alpha*=0.7  )  plt.title('Cancer with BP Model')  plt.show()  # %%  # 显示预测分数  print("预测准确率: {:.4f}".format(BP.score(cancer\_data\_test, cancer\_target\_test)))  predict\_test\_labels = BP.predict(cancer\_data\_test)  # 进行预测结果指标统计 统计每一类别的预测准确率、召回率、F1 分数  print(classification\_report(cancer\_target\_test, predict\_test\_labels))  # 绘制混淆矩阵  cm = confusion\_matrix(cancer\_target\_test, predict\_test\_labels)  sns.heatmap(cm, *annot*=True, *fmt*='d', *cmap*='Blues')  plt.xlabel('Predicted')  plt.ylabel('Actual')  plt.title('Confusion Matrix')  plt.show()    五、实验结果：    六、实验体会：  通过这次实验，我体会到了决策树和神经网络在分类问题上的应用，以及如何通过数据预处理和模型调优来提升模型性能。同时，我也学习到了如何使用Python中的库来处理数据、训练模型，并进行结果的可视化分析。 | | | | | | |
| 教 师  评 语 |  | | | | | |