2024秋《数据挖掘算法与应用》课程报告

本学期《数据挖掘算法与应用》课程报告包含3项内容。

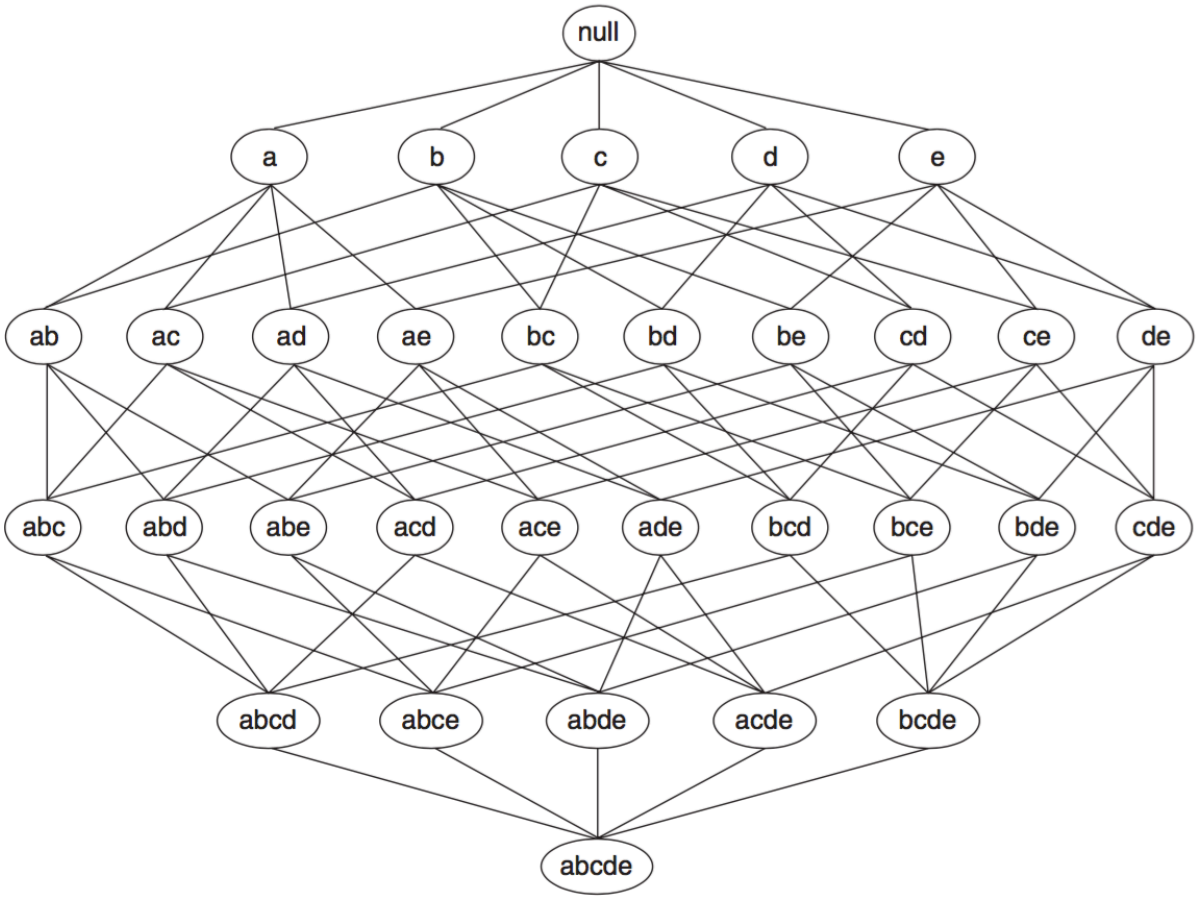
一、基础部分（20分）

已知如下数据集：

Transaction ID	Items Bought
1	a, b, d, e
2	b, c, d
3	a, b, d, e
4	a, c, d, e
5	b, c, d, e
6	b, d, e
7	c, d
8	a, b, c
9	a, d, e
10	b, d

设支持度阈值minsup = 3，在下图中标出该数据集中每个项集的类别和支持度（support）。

- F：频繁项集（frequent item）；
- I：不频繁项集（infrequent item）；
- M：极大频繁项集（maximal frequent item）；
- C：闭合频繁项集（closed frequent item）；
- P：在Apriori算法中被作为候选（candidate）生成出来，但被裁剪（prune）掉的项集。



二、拓展部分（20分）

从下面的论文列表中选择任意一篇论文，阅读该论文，并撰写一篇文献阅读报告。

1. 介绍该论文与本课程哪部分知识相关，归纳总结这部分知识的重要知识点。
2. 介绍该论文的主要研究内容和创新点。
3. 分析该论文的研究工作对推进数据挖掘相关技术的发展能起到什么作用。
4. 注意叙述的逻辑、正确性和规范性。

论文列表：

1. Qiang Huang, Pingyi Luo, and Anthony K. H. Tung. 2023. A New Sparse Data Clustering Method based on Frequent Items. Proc. ACM Manag. Data 1, 1, Article 5 (May 2023), 28 pages. https://doi.org/10.1145/3588685
2. Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. ACM Trans. Database Syst. 42, 3, Article 19 (July 2017), 21 pages. https://doi.org/10.1145/3068335

3. Shangdi Yu, Yiqiu Wang, Yan Gu, Laxman Dhulipala, and Julian Shun. ParChain: A Framework for Parallel Hierarchical Agglomerative Clustering using Nearest-Neighbor Chain. PVLDB, 15(2): 285 - 298, 2022. https://doi.org/10.14778/3489496.3489509

三、实践部分（20分）

使用`adaboost`对MAGIC gamma telescope数据集进行分类。

- 数据集：MAGIC Gamma Telescope - UCI Machine Learning Repository
- 该数据集的类别不平衡。为了平衡该数据集的类别，随机丢弃gamma“g”类的记录，使两类记录数相等。
- 随机分割数据集，使训练集占数据集的 70%，验证集占数据集的 30%。
- 应用任何必要的数据预处理方法，如特征投影（PCA降维等）、特征归一化等。
- 运用合适的指标和方法评估分类效果。

要求：

1. 详细介绍整个分类过程，包括每个步骤的作用、使用的方法、对应的代码等。
2. 注意叙述的逻辑、正确性和规范性。

参考资料：使用`scikit-learn`提供的分类器对金盏花数据集执行分类

在这个教程中，我们将使用金盏花数据集（Iris Dataset）进行分类任务，来说明如何使用`scikit-learn`分类器进行数据挖掘，如何设置训练集和测试集，如何调整超参数，以及如何评估分类模型的性能。

目标

- 了解如何加载和处理数据。
- 使用`scikit-learn`分类器进行训练和预测。
- 使用交叉验证进行模型评估。
- 调整超参数并了解如何设置训练集、验证集和测试集。
- 评估模型的性能，包括准确率、精确率、召回率等指标。

1. 加载金盏花数据集

金盏花数据集是 `scikit-learn` 提供的一个经典数据集，包含 150 个样本，分为 3 个类别（Setosa, Versicolor, Virginica）。每个样本有 4 个特征：花萼长度、花萼宽度、花瓣长度和花瓣宽度。

```
# 导入必要的库
from sklearn import datasets
import pandas as pd

# 加载金盏花数据集
iris = datasets.load_iris()

# 将数据集转换为DataFrame格式，方便查看
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target_names[iris.target]
print(df.head())
```

输出：

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
       species
0                5.1               3.5                1.4
0.2          setosa
1                4.9               3.0                1.4
0.2          setosa
2                4.7               3.2                1.3
0.2          setosa
3                4.6               3.1                1.5
0.2          setosa
4                5.0               3.6                1.4
0.2          setosa
```

2. 分割数据集：训练集和测试集

通常，我们会将数据集分为训练集和测试集。训练集用于训练模型，测试集用于评估模型的性能。`scikit-learn` 提供了 `train_test_split` 函数来方便地进行数据集划分。

```
from sklearn.model_selection import train_test_split

# 获取特征和标签
X = iris.data  # 特征
y = iris.target  # 标签

# 将数据集分割成训练集和测试集，80% 用于训练，20% 用于测试
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

print(f"训练集大小：{X_train.shape[0]}，测试集大小：{X_test.shape[0]}")
```

输出：

```
训练集大小：120，测试集大小：30
```

3. 选择分类器：使用决策树分类器

在 `scikit-learn` 中，有许多分类器可以选择。这里我们使用决策树分类器（`DecisionTreeClassifier`）进行分类。决策树是一个经典的分类方法，它通过树形结构来进行决策。

```
from sklearn.tree import DecisionTreeClassifier

# 创建决策树分类器
clf = DecisionTreeClassifier(random_state=42)

# 在训练集上训练分类器
clf.fit(X_train, y_train)

# 在测试集上进行预测
y_pred = clf.predict(X_test)
```

4. 模型评估：使用准确率和混淆矩阵

分类模型的评估通常使用以下几种指标：

- **准确率（Accuracy）**：正确分类的样本数占总样本数的比例。
- **混淆矩阵（Confusion Matrix）**：展示真实标签与预测标签的对比情况。

```python
from sklearn.metrics import accuracy_score, confusion_matrix

# 计算准确率
accuracy = accuracy_score(y_test, y_pred)
print(f"准确率: {accuracy:.2f}")

# 计算混淆矩阵
cm = confusion_matrix(y_test, y_pred)
print("混淆矩阵:")
print(cm)
```

输出:

```
准确率: 1.00
混淆矩阵:
[[15  0  0]
 [ 0 13  0]
 [ 0  0  2]]
```

5. 超参数调整：交叉验证

在训练模型时，很多模型都有超参数需要调优。对于决策树分类器来说，超参数包括 `max_depth`、`min_samples_split` 等。我们可以使用 `GridSearchCV` 或 `RandomizedSearchCV` 来进行超参数搜索。

例如，使用 `GridSearchCV` 来搜索最优的 `max_depth`：

```python
from sklearn.model_selection import GridSearchCV

# 创建参数字典
param_grid = {'max_depth': [3, 5, 10, 15, None]}

# 创建 GridSearchCV 对象
grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42),
param_grid, cv=5)

# 在训练集上进行超参数调优
grid_search.fit(X_train, y_train)

# 输出最佳参数
print(f"最佳超参数: {grid_search.best_params_}")
```

输出：

```
最佳超参数： {'max_depth': 3}
```

6. 验证集

我们使用交叉验证来评估模型的性能。交叉验证是将数据集分成多个子集，每次使用其中一个子集作为验证集，其余子集作为训练集，进行多次训练和验证。`GridSearchCV` 中的 `cv=5` 表示进行 5 次交叉验证。

7. 输出分类报告

classification_report展示了模型的分类评估结果，

```
from sklearn.metrics import classification_report

# 输出分类报告
print(classification_report(y_test, y_pred,
target_names=iris.target_names))
```

输出：

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00         2

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

8. 评估指标

(1) 混淆矩阵（Confusion Matrix）

在二分类任务中，混淆矩阵是一个 2x2 的矩阵，用于展示真实标签与预测标签的对比情况。具体如下：

	预测为正类 (Positive)	预测为负类 (Negative)
真实为正类 (True Positive, TP)	**TP**	**FN**
真实为负类 (True Negative, TN)	**FP**	**TN**

- **TP (True Positive)**: 真实为正类，且预测为正类的样本数。
- **TN (True Negative)**: 真实为负类，且预测为负类的样本数。
- **FP (False Positive)**: 真实为负类，预测为正类的样本数。
- **FN (False Negative)**: 真实为正类，预测为负类的样本数。

在多分类任务中，混淆矩阵是一个更大的矩阵，每个元素表示预测类别与真实类别的对比。

(2) 精确率（Precision）

精确率是指被预测为正类的样本中，实际为正类的比例。公式如下：

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

高精确率表示模型预测为正类时，大多数都是真正的正类。

(3) 召回率（Recall）

召回率是指所有真实为正类的样本中，模型成功预测为正类的比例。公式如下：

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

高召回率表示模型能够正确识别出大多数的正类样本。

(4) F1 分数（F1 Score）

F1 分数是精确率和召回率的调和平均数，它能够综合衡量模型的精确度和完整性，尤其适用于不平衡数据集的情况。公式如下：

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

高 F1 分数表示模型在精确率和召回率之间有良好的平衡。

(5) 计算精确率、召回率、F1分数的代码示例

```
from sklearn.metrics import precision_score, recall_score, f1_score,
confusion_matrix

# 真实标签
y_true = [0, 1, 1, 0, 1, 0, 1, 0, 1, 1]

# 预测标签
y_pred = [0, 1, 0, 0, 1, 0, 1, 1, 1, 0]

# 计算混淆矩阵
cm = confusion_matrix(y_true, y_pred)
print("混淆矩阵:\n", cm)

# 计算精确率、召回率、F1分数
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

print(f"精确率: {precision:.2f}")
print(f"召回率: {recall:.2f}")
print(f"F1 分数: {f1:.2f}")
```