数据挖掘课程实验
实验2 数据降维与可视化
实验手册

计科210X 甘晴void 202108010XXX

主要放实验报告上实现 效果的python源码

几个中间**csv**文件的解释

- pca_processed.csv（经过PCA降维后的数据，用于t-SNE处理）
- selected_columns.csv（经过特征选择降维之后的数据特征名）
- selected_data.csv（经过特征选择降维之后的数据）

1、特征选择降维（预处理）代码

```
import pandas as pd
from sklearn.feature_selection import VarianceThreshold

# 读取数据集
data = pd.read_csv('实验二数据集.tsv', delimiter='\t', index_col=0)

# 转置数据以使样本在行上，特征在列上
data = data.T

# 1. 方差阈值特征选择
variance_threshold = VarianceThreshold(threshold=0.035)  # 调整阈值
data_variance_selected = variance_threshold.fit_transform(data)

# 获取选择的列索引
selected_columns = data.columns[variance_threshold.get_support()]

# 保存选择的列名到CSV文件，以逗号分隔
selected_columns_text = ','.join(selected_columns)
with open('selected_columns.csv', 'w') as file:
    file.write(selected_columns_text)
```

```
# 输出选择的列名
print("选择的列名：")
print(selected_columns)

# 输出降维后的维度
reduced_dimension = data_variance_selected.shape[1]
print(f"降维后的维度：{reduced_dimension}")

# 保存特征选择后的数据
selected_data = pd.DataFrame(data_variance_selected,
columns=selected_columns)
selected_data.to_csv('selected_data.csv', index=False)
```

2、PCA基础代码

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# 指定降维后的维度
n_components = 14  # 降维后的维度

# 创建PCA模型并进行降维
# 若n_samples >= n_features,则可以调用最大似然估计法自动选择超参数
# pca = PCA(n_components="mle")
# pca_f = PCA(n_components=0.97, svd_solver="full")可以按信息量占比自动
选择超参数
```

```
pca = PCA(n_components=n_components)
pca_result = pca.fit_transform(scaled_features)

# 将降维后的结果转换为DataFrame
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i}' for i in
range(1, n_components + 1)])



# 输出前N个主成分的累计方差解释比例
cumulative_variance_ratio =
sum(pca.explained_variance_ratio_[:n_components])
print(f'Cumulative Variance Explained by {n_components} Principal
Components: {cumulative_variance_ratio:.2%}')

# 指定要绘制的主成分
selected_components = ['PC1', 'PC2']

# 可视化降维后的数据
plt.figure(figsize=(8, 6))
plt.scatter(pca_df[selected_components[0]],
pca_df[selected_components[1]], alpha=0.5)
plt.xlabel(selected_components[0])
plt.ylabel(selected_components[1])
plt.title('PCA Visualization')
plt.show()

# 选择前m个主成分
m = 3  # 选择前5个主成分
selected_pca_df = pca_df.iloc[:, :m]
# 计算相关系数矩阵
correlation_matrix = selected_pca_df.corr()
# 绘制相关系数热力图
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f")
plt.title(f'Correlation Heatmap of the First {m} Principal
Components')
plt.show()

# 绘制前m个主成分之间的散点关系图
sns.pairplot(selected_pca_df)
```

```
plt.suptitle(f'Scatter Plot of the First {m} Principal Components')
plt.show()
```

3、PCA绘制二维图像与计算相关参数

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns


# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# 指定降维后的维度
n_components = 14  # 降维后的维度

# 创建PCA模型并进行降维
pca = PCA(n_components=n_components)
pca_result = pca.fit_transform(scaled_features)

# 将降维后的结果转换为DataFrame
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i}' for i in
range(1, n_components + 1)])

# 输出每一个主成分的方差解释比例
explained_variance_ratios = pca.explained_variance_ratio_
for i, explained_variance_ratio in
enumerate(explained_variance_ratios, 1):
    print(f'Explained Variance Ratio for PC{i}:
{explained_variance_ratio:.10%}')
```

```
# 输出前N个主成分的累计方差解释比例
cumulative_variance_ratio =
sum(pca.explained_variance_ratio_[:n_components])
print(f'Cumulative Variance Explained by {n_components} Principal
Components: {cumulative_variance_ratio:.2%}')

# 指定要绘制的主成分
selected_components = ['PC1', 'PC2']

# 创建一个新列，用于标识数据行所属的部分
pca_df['Group'] = None
pca_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
pca_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
pca_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
pca_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 根据分组使用不同颜色绘制点
plt.figure(figsize=(8, 6))
for group, color in colors.items():
    group_data = pca_df[pca_df['Group'] == group]
    plt.scatter(group_data[selected_components[0]],
group_data[selected_components[1]], c=color, label=group,
alpha=0.5)

plt.xlabel(selected_components[0])
plt.ylabel(selected_components[1])
plt.title('PCA Visualization with Grouping')
plt.legend()
plt.show()

# 选择前m个主成分
m = 5
selected_pca_df = pca_df.iloc[:, :m]
selected_pca_df['Group'] = pca_df['Group']  # 包含 'Group' 列

"""# 绘制相关系数热力图
# 计算相关系数矩阵
correlation_matrix = selected_pca_df.corr()
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm",
fmt=".2f")
plt.title(f'Correlation Heatmap of the First {m} Principal
Components')
plt.show()"""

# 绘制前m个主成分之间的散点关系图，并按组分配不同颜色
sns.pairplot(selected_pca_df, hue='Group', palette=colors)
plt.suptitle(f'Scatter Plot of the First {m} Principal Components')
plt.show()

#将PCA降维的结果保存，以便后续t-SNE的操作
# 将降维后的结果转换为DataFrame
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i}' for i in
range(1, n_components + 1)])
# 保存PCA降维后的数据到CSV文件
pca_df.to_csv('pca_processed.csv', index=False)
```

4、PCA绘制三维图像

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D

# 读取CSV文件，header=0表示有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# 指定主成分的数量
n_components = 3  # 3D图
```

```
# 创建PCA模型并进行降维
pca = PCA(n_components=n_components)
pca_result = pca.fit_transform(scaled_features)

# 将降维后的数据转换为DataFrame
pca_df = pd.DataFrame(data=pca_result, columns=[f'PC{i}' for i in
range(1, n_components + 1)])

# 输出前三个主成分的方差解释比例
explained_variance_ratios = pca.explained_variance_ratio_
for i, explained_variance_ratio in
enumerate(explained_variance_ratios, 1):
    print(f'主成分{i}的方差解释比例: {explained_variance_ratio:.10%}')

# 输出前三个主成分的累积方差解释比例
cumulative_variance_ratio =
sum(pca.explained_variance_ratio_[:n_components])
print(f'前{cumulative_variance_ratio:.2%}的主成分累积方差解释比例')

# 创建一个新列，用于标识数据行所属的部分
pca_df['Group'] = None
pca_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
pca_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
pca_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
pca_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 指定用于三维图的主成分
selected_components = ['PC1', 'PC2', 'PC3']

# 创建一个三维散点图
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# 根据“Group”分配不同颜色的数据点
for group, color in colors.items():
    group_data = pca_df[pca_df['Group'] == group]
```

```
    ax.scatter(group_data[selected_components[0]],
group_data[selected_components[1]],
group_data[selected_components[2]], c=color, label=group,
alpha=0.5)

ax.set_xlabel(selected_components[0])
ax.set_ylabel(selected_components[1])
ax.set_zlabel(selected_components[2])
ax.set_title('PCA可视化三维图')
ax.legend(loc='upper left', bbox_to_anchor=(1.0, 1.0))  #调整图例位置
ax.legend()
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.show()
```

5、ICA绘制二维图像

```
import pandas as pd
from sklearn.decomposition import FastICA
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mutual_info_score
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import kurtosis, skew  # 导入峰度和偏度函数

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# 指定降维后的维度
n_components = 3  # 降维后的维度

# 创建ICA模型并进行降维
ica = FastICA(n_components=n_components)
```

```
ica_result = ica.fit_transform(scaled_features)

# 将降维后的结果转换为DataFrame
ica_df = pd.DataFrame(data=ica_result, columns=[f'IC{i}' for i in
range(1, n_components + 1)])

# 计算信噪比（SNR）
# 假设第一个独立成分是信号，剩下的成分是噪声
signal_component = ica_result[:, 0]
noise_components = ica_result[:, 1:]
snr = np.mean(np.abs(signal_component) / np.std(noise_components,
axis=1))
print(f'Signal-to-Noise Ratio (SNR): {snr:.2f}')

# 指定要绘制的独立成分
selected_components = ['IC1', 'IC2']
# 创建一个新列，用于标识数据行所属的部分
ica_df['Group'] = None
ica_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
ica_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
ica_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
ica_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分
# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}
# 根据分组使用不同颜色绘制点
plt.figure(figsize=(8, 6))
for group, color in colors.items():
    group_data = ica_df[ica_df['Group'] == group]
    plt.scatter(group_data[selected_components[0]],
group_data[selected_components[1]], c=color, label=group,
alpha=0.5)
plt.xlabel(selected_components[0])
plt.ylabel(selected_components[1])
plt.title('ICA Visualization with Grouping')
plt.legend()
plt.show()

"""# 选择前m个独立成分
m = 5
selected_ica_df = ica_df.iloc[:, :m]
selected_ica_df['Group'] = ica_df['Group']  # 包含 'Group' 列
```

```
# 绘制前m个独立成分之间的散点关系图，并按组分配不同颜色
sns.pairplot(selected_ica_df, hue='Group', palette=colors)
plt.suptitle(f'Scatter Plot of the First {m} Independent
Components')
plt.show()"""


# 计算互信息（Mutual Information）
# 假设第一个独立成分是信号，原始信号是真实信号
true_signal = scaled_features[:, 0]
mi = mutual_info_score(true_signal, signal_component)
print(f'Mutual Information (MI) with True Signal: {mi:.4f}')

# 计算峰度和偏度
kurtosis_values = kurtosis(ica_result, axis=0)
skewness_values = skew(ica_result, axis=0)

# 创建一个DataFrame来存储结果
result_df = pd.DataFrame({'Component': ica_df.columns[:-1],
'Kurtosis': kurtosis_values, 'Skewness': skewness_values})

# 打印结果
print(result_df)
```

6、ICA绘制三维图像

```
from mpl_toolkits.mplot3d import Axes3D  # 导入3D绘图库
import pandas as pd
from sklearn.decomposition import FastICA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt


# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaler = StandardScaler()
```

```
scaled_features = scaler.fit_transform(features)

# 指定降维后的维度
n_components = 3  # 降维后的维度

# 创建ICA模型并进行降维
ica = FastICA(n_components=n_components)
ica_result = ica.fit_transform(scaled_features)

# 将降维后的结果转换为DataFrame
ica_df = pd.DataFrame(data=ica_result, columns=[f'IC{i}' for i in
range(1, n_components + 1)])

# 创建一个新列，用于标识数据行所属的部分
ica_df['Group'] = None
ica_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
ica_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
ica_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
ica_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 选择要绘制的独立成分
selected_components = ['IC1', 'IC2', 'IC3']

# 创建一个新的三维图
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# 根据分组使用不同颜色绘制点
for group, color in colors.items():
    group_data = ica_df[ica_df['Group'] == group]
    ax.scatter(group_data[selected_components[0]],
group_data[selected_components[1]],
group_data[selected_components[2]], c=color, label=group,
alpha=0.5)

ax.set_xlabel(selected_components[0])
ax.set_ylabel(selected_components[1])
ax.set_zlabel(selected_components[2])
```

```
ax.set_title('ICA 3D Visualization with Grouping')
ax.legend()

plt.show()
```

7、UMAP绘制二维图像

```
import pandas as pd
import umap
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaled_features = StandardScaler().fit_transform(features)

# 创建UMAP模型并进行降维
n_components = 2  # 降维后的维度
umap_model = umap.UMAP(n_neighbors=4, n_components=n_components)
umap_result = umap_model.fit_transform(scaled_features)

# 将降维后的结果转换为DataFrame
umap_df = pd.DataFrame(data=umap_result, columns=[f'UMAP{i}' for i
in range(1, n_components + 1)])

# 指定要绘制的UMAP成分
selected_components = ['UMAP1', 'UMAP2']

# 创建一个新列，用于标识数据行所属的部分
umap_df['Group'] = None
umap_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
umap_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
umap_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
umap_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分
```

```
# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 根据分组使用不同颜色绘制点
plt.figure(figsize=(8, 6))
for group, color in colors.items():
    group_data = umap_df[umap_df['Group'] == group]
    plt.scatter(group_data[selected_components[0]],
group_data[selected_components[1]], c=color, label=group,
alpha=0.5)

plt.xlabel(selected_components[0])
plt.ylabel(selected_components[1])
plt.title('UMAP Visualization with Grouping')
plt.legend()
plt.show()
```

8、UMAP绘制三维图像

```
import pandas as pd
import umap
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # 导入3D绘图模块
from sklearn.preprocessing import StandardScaler

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('selected_data.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 对特征进行标准化
scaled_features = StandardScaler().fit_transform(features)

# 创建UMAP模型并进行降维
n_components = 3  # 降维后的维度
umap_model = umap.UMAP(n_neighbors=4, n_components=n_components)
umap_result = umap_model.fit_transform(scaled_features)
```

```
# 将降维后的结果转换为DataFrame
umap_df = pd.DataFrame(data=umap_result, columns=[f'UMAP{i}' for i
in range(1, n_components + 1)])

# 指定要绘制的UMAP成分
selected_components = ['UMAP1', 'UMAP2', 'UMAP3']  # 选择三个成分

# 创建一个新列，用于标识数据行所属的部分
umap_df['Group'] = None
umap_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
umap_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
umap_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
umap_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 创建一个三维坐标轴
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# 根据分组使用不同颜色绘制点
for group, color in colors.items():
    group_data = umap_df[umap_df['Group'] == group]
    ax.scatter(group_data[selected_components[0]],
group_data[selected_components[1]],
group_data[selected_components[2]], c=color, label=group,
alpha=0.5)

ax.set_xlabel(selected_components[0])
ax.set_ylabel(selected_components[1])
ax.set_zlabel(selected_components[2])
ax.set_title('UMAP 3D Visualization with Grouping')
plt.legend()
plt.show()
```

9、t-SNE绘制二维图像

```
import pandas as pd
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('pca_processed.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 创建t-SNE模型并进行降维
n_components = 2  # 降维后的维度
tsne_model = TSNE(n_components=n_components, random_state=7)
tsne_result = tsne_model.fit_transform(features)

# 将降维后的结果转换为DataFrame
tsne_df = pd.DataFrame(data=tsne_result, columns=[f't-SNE{i}' for i
in range(1, n_components + 1)])

# 指定要绘制的t-SNE成分
selected_components = ['t-SNE1', 't-SNE2']

# 创建一个新列，用于标识数据行所属的部分
tsne_df['Group'] = None
tsne_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
tsne_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
tsne_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
tsne_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 根据分组使用不同颜色绘制点
plt.figure(figsize=(8, 6))
for group, color in colors.items():
    group_data = tsne_df[tsne_df['Group'] == group]
```

```
    plt.scatter(group_data[selected_components[0]],
group_data[selected_components[1]], c=color, label=group,
alpha=0.5)

plt.xlabel(selected_components[0])
plt.ylabel(selected_components[1])
plt.title('t-SNE Visualization with Grouping')
plt.legend()
plt.show()
```

10、t-SNE绘制三维图像

```
import pandas as pd
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # 导入三维绘图工具
import seaborn as sns

# 读取CSV文件，header=None表示没有列标签
data = pd.read_csv('pca_processed.csv', header=0)

# 提取特征（所有列）
features = data.iloc[:, :]

# 创建t-SNE模型并进行降维
n_components = 3  # 降维后的维度
tsne_model = TSNE(n_components=n_components, random_state=7)
tsne_result = tsne_model.fit_transform(features)

# 将降维后的结果转换为DataFrame
tsne_df = pd.DataFrame(data=tsne_result, columns=[f't-SNE{i}' for i
in range(1, n_components + 1)])

# 指定要绘制的t-SNE成分
selected_components = ['t-SNE1', 't-SNE2', 't-SNE3']

# 创建一个新列，用于标识数据行所属的部分（根据您的需求设置）
tsne_df['Group'] = None
tsne_df.loc[0:16, 'Group'] = 'MF'  # 第一部分
tsne_df.loc[16:32, 'Group'] = 'METH'  # 第二部分
```

```
tsne_df.loc[32:48, 'Group'] = 'GE'  # 第三部分
tsne_df.loc[48:64, 'Group'] = 'CNA'  # 第四部分

# 定义颜色映射
colors = {'MF': 'red', 'METH': 'blue', 'GE': 'green', 'CNA':
'purple'}

# 创建三维图像
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')  # 创建三维绘图区域

# 根据分组使用不同颜色绘制点
for group, color in colors.items():
    group_data = tsne_df[tsne_df['Group'] == group]
    ax.scatter(group_data[selected_components[0]],
group_data[selected_components[1]],
               group_data[selected_components[2]], c=color,
label=group, alpha=0.5)

ax.set_xlabel(selected_components[0])
ax.set_ylabel(selected_components[1])
ax.set_zlabel(selected_components[2])
ax.set_title('t-SNE 3D Visualization with Grouping')
plt.legend()
plt.show()
```