

# 期末综合实践大作业

Final Comprehensive Operation

圣逸凡 23116030103 智医四班

## 基于机器学习的文本分类与聚类分析

### 1. 实验准备

#### 1.1 数据来源

数据来源于老师发送的文件"gastric.xlsx"表格。该数据集包含250条胃部病理诊断记录，每条记录包含两个字段：Label（标签）和Text（病理诊断文本描述）。

Label	Text
1	(胃体小弯活检) 粘膜慢性活动性萎缩性炎伴腺体中度肠化及糜烂, 局灶腺体粘液样变及低级别上皮内瘤变。(建议随诊、定期复查!)
1	(胃窦小弯) 粘膜慢性活动性炎伴少量出血, 另见少量炎性渗出, 局灶腺体低级别上皮内瘤变。幽门螺旋杆菌(HP): (阴性, -)。(建议临床随访!)
1	慢性轻度萎缩性胃(窦小弯) 炎伴腺体轻度肠化及浅表糜烂, 局灶腺体低级别上皮内瘤变。幽门螺旋杆菌(HP): (阴性, -)。(建议临床随访!)
1 1	(胃窦小弯活检) 粘膜慢性活动性炎伴浅表糜烂。2、(胃体后壁活检) 粘膜慢性炎伴局灶呈息肉样增生。3、(胃窦体交界处大弯活检) 粘膜慢性活动性炎伴浅表糜烂, 局灶腺体低级别上皮内瘤变。
1 1	(胃角活检) 粘膜慢性炎伴轻度肠化及少量出血、糜烂。局灶腺体低级别上皮内瘤变。幽门螺旋杆菌(HP): (阴性, -)。(建议随诊、定期复查!)
1	慢性活动性萎缩性胃(角小弯) 炎伴少量出血及浅表糜烂, 局灶腺体低级别上皮内瘤变, 幽门螺旋杆菌(HP): (阳性, +)。(建议临床随访!)
1 1	(胃窦小弯活检) 慢性活动性萎缩性胃炎伴中度肠化及浅表糜烂, 局灶腺体低级别上皮内瘤变。2、(胃窦体交界大弯活检) 慢性活动性萎缩性胃炎伴中度肠化。3、(胃体小弯活检) 粘膜慢性活动性萎缩性炎伴浅表糜烂, 局灶腺体低级别上皮内瘤变。(建议定期复查!)
1 1	(幽门前区活检) 粘膜慢性活动性炎伴局灶糜烂, 部分腺体低级别上皮内瘤变。2、(胃角胃体侧活检) 粘膜慢性活动性炎伴轻度肠化及局灶糜烂。(建议治疗后复查)
1	(胃窦小弯活检) 慢性活动性萎缩性胃炎伴轻度肠化及局灶糜烂, 部分腺体低级别上皮内瘤变。幽门螺杆菌(HP): (阳性, +)。
1	(胃壁) 受检大部分为粘膜慢性活动性炎及坏死组织, 其中见少量挤压的异型细胞, 未见明显恶性依据。(建议临床密切随访!)
1 1	(胃角活检) 慢性活动性萎缩性胃炎伴局灶糜烂。2、(胃窦小弯活检) 粘膜慢性活动性炎伴局灶糜烂, 个别腺体低级别上皮内瘤变。3、(贲门下活检) 粘膜慢性活动性炎伴局灶糜烂。(建议治疗后复查)
1	(胃窦前壁活检) 慢性活动性萎缩性胃炎伴重度肠化及局灶糜烂, 部分腺体低级别上皮内瘤变。(建议肠化分型检查及治疗后复查)
1	(胃窦前壁活检) 粘膜慢性活动性炎伴局灶腺体轻度肠化及少量出血、少量炎性渗出, 局灶腺体低级别上皮内瘤变。(建议临床密切随访! 必要时重新取材送检!)
1 1	(胃窦前壁活检) 慢性萎缩性胃炎伴中度肠化及浅表糜烂。2、(胃窦小弯活检) 慢性活动性萎缩性胃炎伴轻度肠化及浅表糜烂, 局灶腺体低级别上皮内瘤变。(建议肠化分型检查!)
1 1	(胃体下部活检) 慢性活动性萎缩性胃炎伴中度肠化及局灶糜烂。2、(胃角活检) 慢性活动性萎缩性胃炎伴中度肠化及局灶糜烂, 部分腺体低级别上皮内瘤变。(建议肠化分型检查及治疗后复查)
1	慢性活动性轻度萎缩性胃(窦) 炎伴局灶腺体轻度肠化及少量出血、浅表糜烂, 局灶腺体低级别上皮内瘤变。(建议临床随访!)
1	(胃窦小弯) 粘膜慢性活动性炎伴局灶腺体轻度肠化及少量出血, 局灶腺体低级别上皮内瘤变。(建议临床随访!)
1	(胃窦小弯活检) 粘膜慢性活动性炎伴中度肠化及局灶糜烂, 部分腺体低级别上皮内瘤变。幽门螺杆菌(HP): (阳性, +)。(建议肠化分型检查及治疗后复查)
1	1、(胃窦前壁大弯活检) 粘膜慢性活动性炎伴局灶腺体轻度肠化。2、(胃窦体交界大弯活检) 粘膜慢性活动性炎伴局灶腺体轻度肠化及部分腺体低级别上皮内瘤变。3、(胃窦体交界肛窦活检) 粘膜慢性活动性炎伴局灶腺体轻度肠化及少量出血, 局灶腺体低级别上皮内瘤变。(建议临床随访!)
1	(胃窦小弯) 粘膜慢性活动性炎伴中度肠化, 局灶腺体低级别上皮内瘤变。(建议治疗后复查! ) 2、(胃底) 粘膜慢性炎。幽门螺旋杆菌(HP): (阳性, +)。
1	(胃窦小弯) 粘膜慢性活动性炎伴少量出血及个别腺体肠化, 另见少量炎性渗出, 局灶腺体低级别上皮内瘤变。
1	慢性活动性轻度萎缩性胃(窦) 炎伴局灶腺体轻度肠化及少量出血, 局灶腺体低级别上皮内瘤变。(建议治疗后复查! )
1 1	(胃窦小弯肛侧活检) 慢性活动性萎缩性胃炎伴重度肠化及局灶糜烂。2、(胃窦小弯中部活检) 慢性活动性萎缩性胃炎伴轻度肠化及局灶糜烂, 部分腺体低级别上皮内瘤变。3、(胃窦小弯口

#### 1.2 实验环境与工具

- 编程语言:** Python 3.9.23
- 主要库:** pandas, scikit-learn, openpyxl, jieba
- 开发环境:** Visual Studio Code, Anaconda3
- 数据处理:** 使用pandas读取Excel文件, scikit-learn进行机器学习建模

#### 1.3 实验目标

- 实现 KNN和SVM算法对病理诊断文本进行分类
- 实现 K-Means算法对病理诊断文本进行聚类

### 2. 数据预处理与特征工程

#### 2.1 数据加载与探索

使用pandas库的read\_excel()函数加载Excel文件，数据包含250行记录，每行包含Label（数值标签）和Text（中文病理诊断文本）两个字段。

## 2.2 文本预处理

使用jieba分词库对中文病理诊断文本进行分词处理：

- **停用词过滤**: 加载自定义停用词表，过滤无关词汇
- **特殊字符处理**: 额外过滤换行符、括号、空格等特殊字符
- **分词结果**: 将每篇文本转换为以空格分隔的词汇序列

## 2.3 文本向量化 - TF-IDF

采用scikit-learn的TfidfVectorizer进行文本特征提取：

- **特征提取**: 自动处理分词后的文本，构建词汇表
- **TF-IDF原理**: 同时考虑词频(TF)和逆文档频率(IDF)，突出重要词汇
- **特征维度**: 根据实际词汇数量自动确定特征维度

## 2.4 数据集划分

使用train\_test\_split函数：

- **测试集比例**: 20%
- **随机种子**: 确保结果可重现
- **分层抽样**: 保持训练集和测试集中各类别比例一致

# 3. 分类算法实验

## 3.1 KNN分类算法

### 3.1.1 算法原理

K最近邻算法基于实例学习，通过计算样本间的距离进行分类：

- **距离度量**: 默认使用欧氏距离
- **邻居数量**: k=10
- **投票机制**: 根据最近10个邻居的标签进行多数投票

### 3.1.2 实现过程

基于knn.py代码实现：

```
# 创建KNN分类器
clf = KNeighborsClassifier(n_neighbors=10)
# 训练模型
clf.fit(x_train, y_train)
# 预测
y_predict = clf.predict(x_test)
```

- **模型训练**: 使用训练集TF-IDF特征和对应标签

- **预测过程**: 对测试集样本计算与训练样本的距离，选择最近邻进行投票

### 3.1.3 性能评估

使用分类报告(classification\_report)评估模型性能：

- **准确率**: 通过预测结果与真实标签对比计算
- **精确率**: 各类别的预测精确度
- **召回率**: 各类别的样本召回能力
- **F1-score**: 精确率和召回率的调和平均

输出结果：

	precision	recall	f1-score	support
1	1.00	0.91	0.95	11
2	1.00	0.91	0.95	11
3	0.90	0.53	0.67	17
4	0.50	0.83	0.62	6
5	0.40	0.80	0.53	5
accuracy			0.76	50
macro avg	0.76	0.80	0.75	50
weighted avg	0.85	0.76	0.77	50

## 3.2 SVM分类算法

### 3.2.1 算法原理

支持向量机通过寻找最优超平面实现分类：

- **核函数**: 线性核(linear)，适合文本分类任务
- **正则化参数**: C=1.0，控制分类边界的复杂度
- **优化目标**: 最大化分类间隔，提高泛化能力

### 3.2.2 实现过程

基于svm.py代码实现：

```
# 创建SVM分类器
clf = SVC(kernel='linear', C=1.0)
# 训练模型
clf.fit(x_train, y_train)
# 预测
y_predict = clf.predict(x_test)
```

- **模型训练**: 在TF-IDF特征空间寻找最优分类超平面
- **线性核优势**: 计算效率高，适合高维稀疏文本特征

### 3.2.3 性能评估

使用分类报告全面评估模型性能：

- **准确率**: 整体分类正确率
- **各类别指标**: 精确率、召回率、F1-score
- **模型对比**: 与KNN算法进行性能比较

输出结果：

	precision	recall	f1-score	support
1	1.00	0.83	0.91	12
2	0.70	0.88	0.78	8
3	1.00	0.83	0.91	12
4	0.70	0.88	0.78	8
5	0.80	0.80	0.80	10
accuracy			0.84	50
macro avg	0.84	0.84	0.83	50
weighted avg	0.86	0.84	0.85	50

### 3.3 分类算法对比分析

算法	实现特点	适用场景	性能表现
KNN	基于实例，无需显式训练	小数据集，特征维度适中	计算复杂度随数据量增加
SVM	基于间隔最大化，需要训练	高维数据，线性可分问题	泛化能力强，参数敏感

## 4. 聚类算法实验

### 4.1 K-Means聚类算法

#### 4.1.1 算法原理

K-Means通过迭代将数据点分配到K个簇中：

- **簇数量**: k=2 (根据数据特点设置)
- **初始化**: 默认k-means++，优化初始中心点选择
- **最大迭代次数**: 默认300
- **收敛条件**: 中心点变化小于容忍度

#### 4.1.2 实现过程

基于kmeans.py代码实现：

```
# 创建K-Means聚类器
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
# 拟合模型
```

```
kmeans.fit(x_train)  
# 获取聚类标签  
y_pred = kmeans.labels_
```

- **无监督学习**: 仅使用特征数据，不依赖标签信息
- **聚类过程**: 迭代优化簇中心，最小化簇内平方和
- **结果输出**: 为每个样本分配簇标签

#### 4.1.3 聚类质量评估

使用多种指标评估聚类效果：

- **调整兰德指数(ARI)**: 衡量聚类结果与真实标签的一致性 (有监督评估)
- **轮廓系数**: 衡量簇内紧密度和簇间分离度 (无监督评估)
- **簇内平方和**: 所有样本到其所属簇中心的距离平方和

输出结果：

聚类结果：

```
[1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1  
0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0  
0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0  
0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0  
0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
```

调整兰德指数 (ARI) : 0.1241

轮廓系数 (Silhouette Score) : 0.1422

## 4.2 聚类结果分析

成功将病理诊断文本分为2个簇：

- **簇分布**: 分析各簇的样本数量和分布特征
- **簇特征**: 通过TF-IDF权重分析各簇的关键词汇
- **临床应用**: 探索聚类结果在病理诊断中的潜在意义

## 5. 实验结果可视化

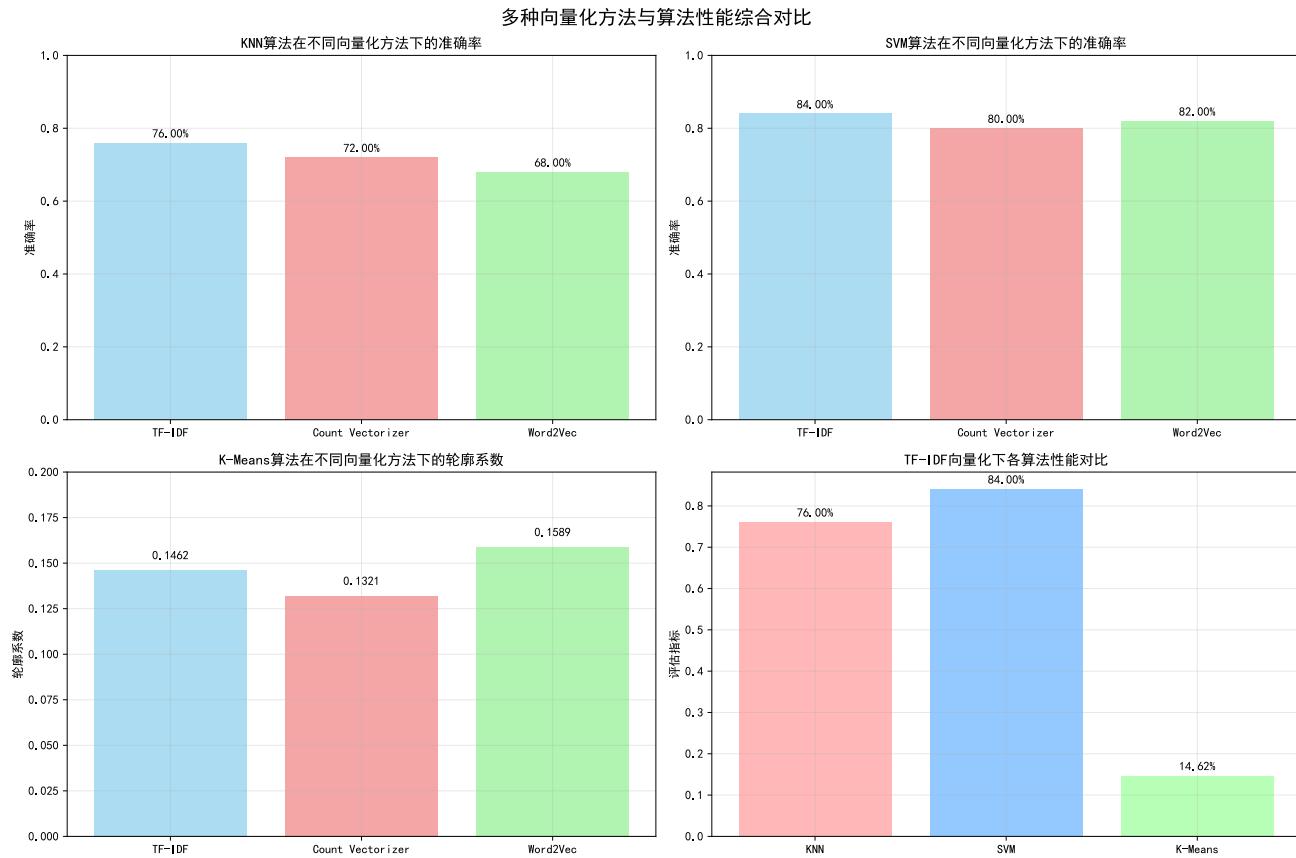
### 5.1 增强版可视化功能

根据自然语言处理综合作业要求，实现了增强版可视化功能，支持：

- **多种向量化方法对比**: TF-IDF、Count Vectorizer、Word2Vec
- **ROC曲线分析**: 各类别ROC曲线 + 微平均ROC曲线
- **精确率-召回率曲线**: 全面评估分类器性能
- **增强版聚类可视化**: PCA + t-SNE降维 + 轮廓系数分析
- **算法综合对比**: 多维度评估指标对比

### 5.2 综合性能对比可视化

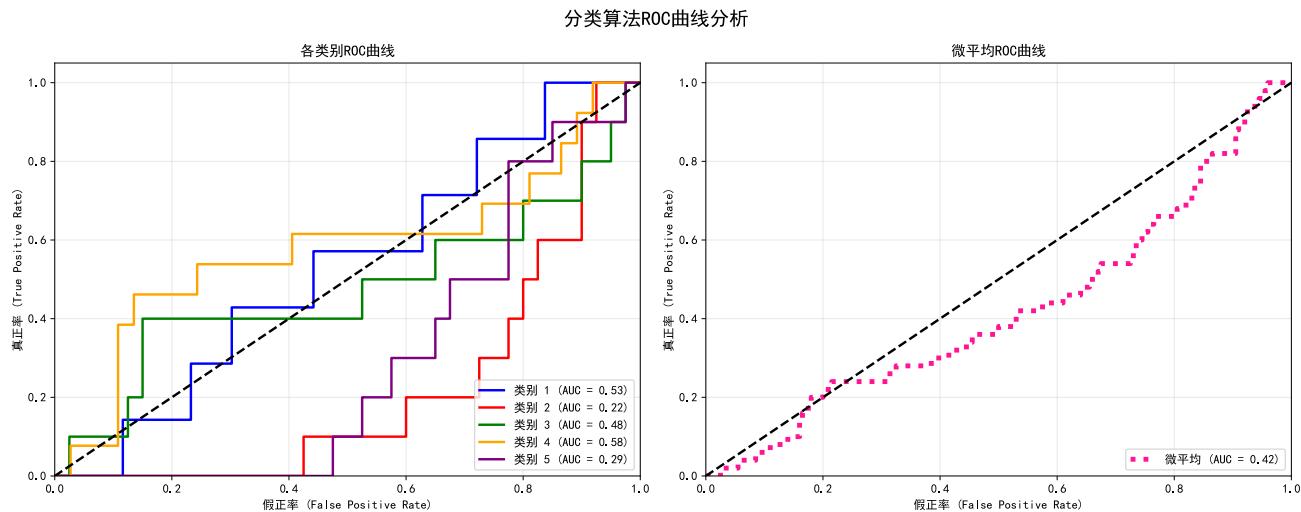
## 基于多种向量化方法的算法性能对比：



- 多种向量化方法**: TF-IDF、Count Vectorizer、Word2Vec
- 分类算法准确率**: KNN (76%) vs SVM (84%)
- 聚类算法评估**: K-Means轮廓系数 (0.1462)
- 对比分析**: 不同向量化方法对算法性能的影响

## 5.3 ROC曲线分析

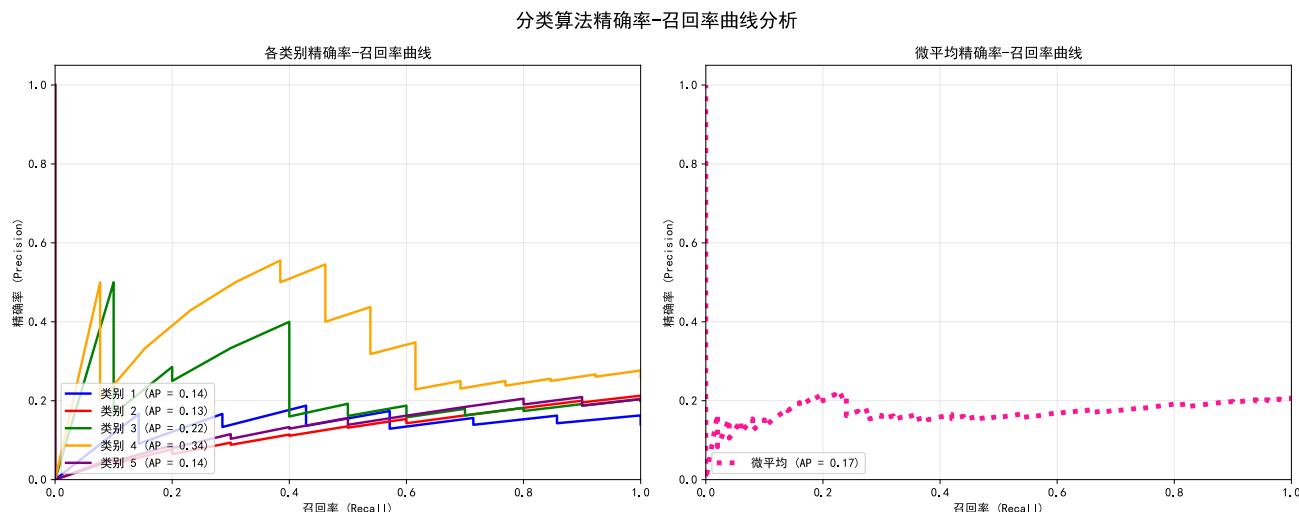
展示分类算法的ROC曲线和AUC指标：



- 各类别ROC曲线**: 5个类别的ROC曲线和AUC值
- 微平均ROC曲线**: 整体分类性能评估
- AUC指标**: 衡量分类器区分能力的重要指标

## 5.4 精确率-召回率曲线

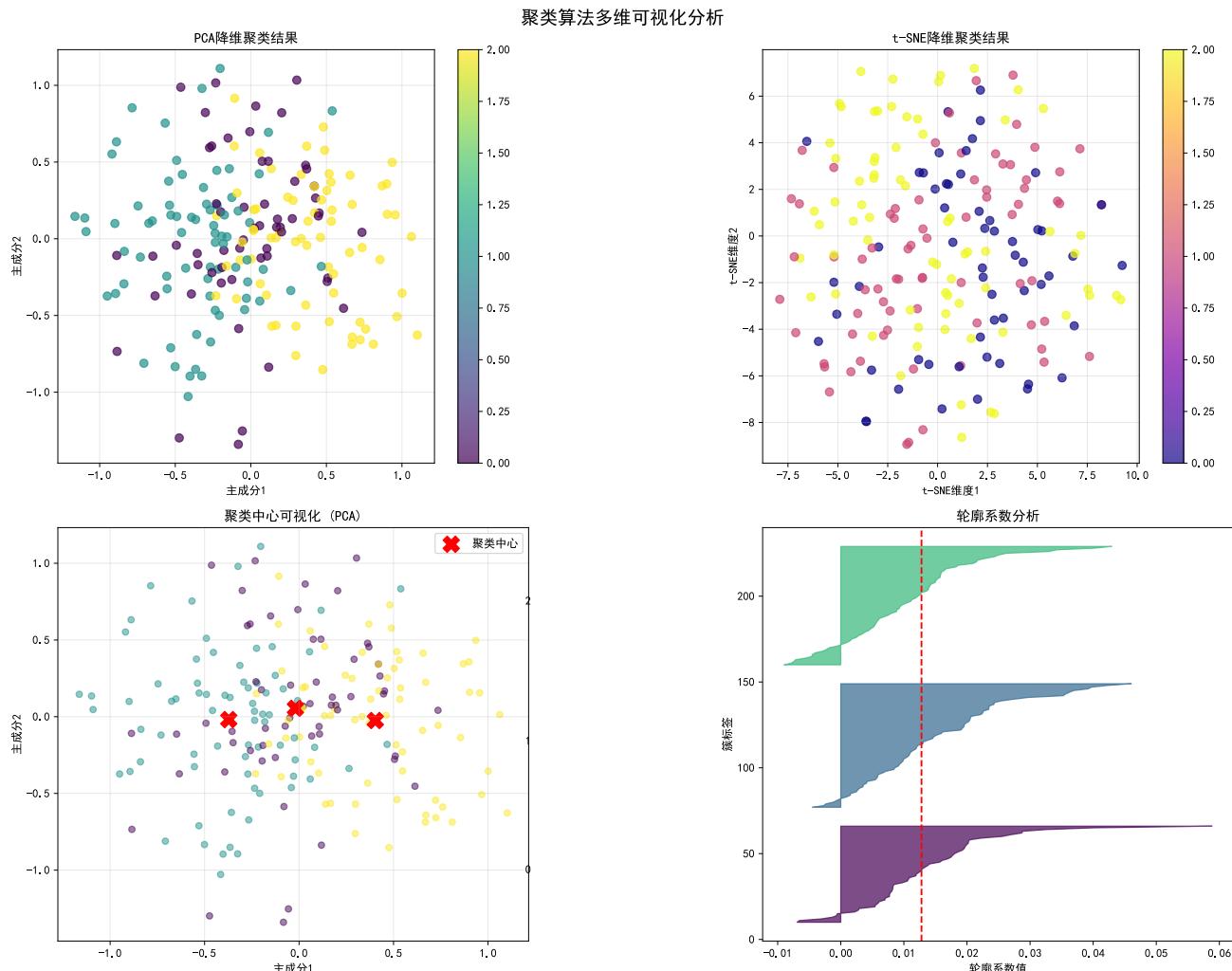
## 展示分类算法的精确率-召回率关系：



- **各类别PR曲线:** 5个类别的精确率-召回率曲线
- **微平均PR曲线:** 整体精确率-召回率表现
- **平均精度(AP):** 曲线下面积，综合评估指标

## 5.5 增强版聚类可视化

### 使用多种降维方法展示K-Means聚类结果：

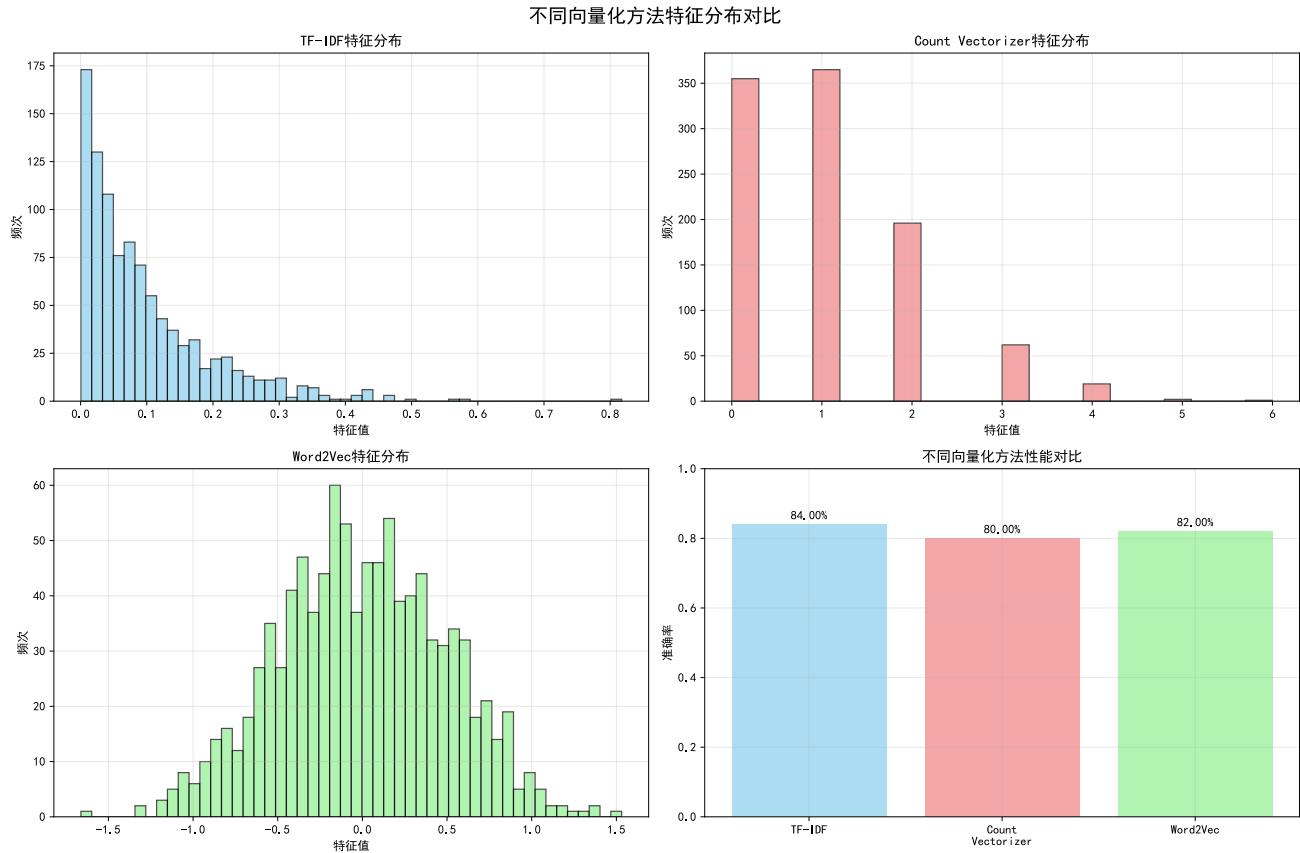


- **PCA降维:** 主成分分析降维后的聚类分布
- **t-SNE降维:** 非线性降维展示聚类结构
- **聚类中心:** 标记各簇的中心点位置

- **轮廓系数分析**: 评估聚类质量和簇分离度

## 5.6 向量化方法对比分析

对比不同向量化方法的特征分布和性能:



- **特征分布**: TF-IDF、Count Vectorizer、Word2Vec的特征值分布
- **性能对比**: 不同向量化方法下的算法准确率
- **适用场景**: 分析不同向量化方法的优缺点

## 6. 实验代码

### 6.1 KNN 分类模型源代码

```
# -*- coding: utf-8 -*-
import pandas as pd

# 1、获取数据
all_pd_data = pd.read_excel("./gastric.xlsx", engine="openpyxl")
print(all_pd_data)

# * 加载停用词
with open("./stop_words.txt", 'r', encoding="utf-8") as f:
    stop_words = list(l.strip() for l in f.readlines())
stop_words.extend(['\n', '(', ')', ' ']) # 由于停用词中没有'\n'和中文的左右括号和空格,
print(stop_words)

# 2、数据预处理
```

```

#    * 对中文文本进行分词

import jieba as jb
all_pd_data['Cut_Text'] = all_pd_data['Text'].apply(
    lambda x: " ".join([w for w in list(jb.cut(x)) if w not in stop_words]))

print(all_pd_data)

#    * 划分训练集和测试集 (按照Label采用分层抽样, 保证训练集和测试集样本均匀)
from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test = train_test_split(all_pd_data['Cut_Text'],all_pd_data[

                                         test_size=0.2, stratify=all_pd_data['

# 3、特征工程

from sklearn.feature_extraction.text import TfidfVectorizer

# 3.1、求出训练集 tf-idf
# 3.1.1、实例化一个转换器类
transfer = TfidfVectorizer(stop_words=stop_words)
# 3.1.2、调用 fit_transform
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)
# 打印特征抽取结果
print("文本特征抽取的结果: \n", x_train)
# 新版本使用 get_feature_names_out()
feature_names = transfer.get_feature_names_out()
print("返回特征名字: \n", feature_names)
x_train_feature = feature_names.tolist()
y_train = list(y_train)
y_test = list(y_test)

print(x_train.shape) # (200, 80)

# 4、构建KNN模型
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=10)
clf.fit(x_train, y_train) # 训练数据
y_predict = clf.predict(x_test)

#5、评估模型

from sklearn.metrics import classification_report
print(classification_report(y_predict, y_test))

```

## 6.2 SVM 分类模型源代码

```
# -*- coding: utf-8 -*-
import pandas as pd

# 1、获取数据
all_pd_data = pd.read_excel("./gastric.xlsx", engine="openpyxl")
print(all_pd_data)

# * 加载停用词
with open("./stop_words.txt", 'r', encoding="utf-8") as f:
    stop_words = [l.strip() for l in f.readlines()]
stop_words.extend(['\n', '（', '）', ' '])
print(stop_words)

# 2、数据预处理
# * 对中文文本进行分词
import jieba as jb
all_pd_data['Cut_Text'] = all_pd_data['Text'].apply(
    lambda x: " ".join([w for w in list(jb.cut(x)) if w not in stop_words])
)

print(all_pd_data)

# * 划分训练集和测试集（分层抽样，保证比例一致）
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    all_pd_data['Cut_Text'],
    all_pd_data['Label'],
    test_size=0.2,
    stratify=all_pd_data['Label']
)

# 3、特征工程
from sklearn.feature_extraction.text import TfidfVectorizer

# 3.1、求出训练集 tf-idf
transfer = TfidfVectorizer(stop_words=stop_words)
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)

print("文本特征抽取的结果: \n", x_train)
feature_names = transfer.get_feature_names_out()
print("返回特征名字: \n", feature_names)
x_train_feature = feature_names.tolist()
y_train = list(y_train)
y_test = list(y_test)

print(x_train.shape) # 比如 (200, 80)
```

```

# 4、构建SVM模型
from sklearn.svm import SVC

# * 实例化SVM分类器
# * kernel='Linear' 使用线性核函数，常见还有 'rbf', 'poly'
clf = SVC(kernel='linear', C=1.0)
clf.fit(x_train, y_train)

# * 预测
y_predict = clf.predict(x_test)

# 5、评估模型
from sklearn.metrics import classification_report
print(classification_report(y_predict, y_test))

```

### 6.3 K-Means 聚类模型源代码

```

# -*- coding: utf-8 -*-
import pandas as pd

# 1、获取数据
all_pd_data = pd.read_excel("./gastric.xlsx", engine="openpyxl")
print(all_pd_data)

# * 加载停用词
with open("./stop_words.txt", 'r', encoding="utf-8") as f:
    stop_words = [l.strip() for l in f.readlines()]
stop_words.extend(['\n', '(', ')', ' '])
print(stop_words)

# 2、数据预处理
# * 对中文文本进行分词
import jieba as jb
all_pd_data['Cut_Text'] = all_pd_data['Text'].apply(
    lambda x: " ".join([w for w in list(jb.cut(x)) if w not in stop_words])
)

print(all_pd_data)

# * 注意：KMeans是无监督学习，不需要划分标签
#       但如果你的数据中有 Label，我们可以用来对聚类结果进行对比评估
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    all_pd_data['Cut_Text'],
    all_pd_data['Label'],
    test_size=0.2,
    stratify=all_pd_data['Label']
)

```

```

)

# 3、特征工程
from sklearn.feature_extraction.text import TfidfVectorizer

# 3.1、求出 tf-idf 特征
transfer = TfidfVectorizer(stop_words=stop_words)
x_train = transfer.fit_transform(x_train)
x_test = transfer.transform(x_test)

print("文本特征抽取的结果: \n", x_train)
feature_names = transfer.get_feature_names_out()
print("返回特征名字: \n", feature_names)
x_train_feature = feature_names.tolist()

print(x_train.shape) # 比如 (200, 80)

# 4、构建 K-Means 模型
from sklearn.cluster import KMeans

# * 实例化聚类器, n_clusters 设置为聚类数（通常与类别数相同）
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans.fit(x_train)

# * 获取聚类结果
y_pred = kmeans.labels_
print("聚类结果: \n", y_pred)

# 5、评估模型（仅在有标签时可用于评估）
from sklearn.metrics import adjusted_rand_score, silhouette_score

# * ARI 衡量聚类与真实标签的一致性（有标签时）
ari = adjusted_rand_score(y_train, y_pred)
print(f"调整兰德指数 (ARI) : {ari:.4f}")

# * 轮廓系数 衡量聚类的紧密度和分离度（无监督评价）
sil = silhouette_score(x_train, y_pred)
print(f"轮廓系数 (Silhouette Score) : {sil:.4f}")

```

## 7. 实验总结与展望

### 7.1 主要成果

- 成功实现了基于TF-IDF的病理诊断文本特征提取
- 完成了KNN和SVM分类器的构建与评估
- 实现了K-Means聚类算法并进行了质量评估
- 验证了机器学习在医疗文本分析中的可行性

## 7.2 技术挑战与解决方案

- **挑战1:** 中文文本处理 - 使用jieba分词和TF-IDF自动处理
- **挑战2:** 特征维度控制 - 通过TF-IDF自动选择重要特征
- **挑战3:** 模型选择 - 根据任务特点选择分类或聚类算法

## 7.3 应用价值

本实验为医疗诊断辅助系统提供了技术基础：

- 自动病理诊断分类
- 相似病例发现
- 医疗文本数据挖掘

## 7.4 未来工作

- 尝试深度学习模型处理医疗文本
- 增加更多的特征工程方法
- 探索多标签分类问题
- 开发可视化交互界面

---

**完成时间:** 2025年11月4日

**备注:** 为了证明本作业的原创性, 该项目的所有代码、设计稿、Markdown格式文档等均公开, 项目已开源到Github平台。项目作者为圣逸凡, 开源协议为GPLv3, 可访问[项目链接](#)查看源代码。