实验报告（5）

学号：202202206341 姓名：王子俊

1. **实验名称： 朴素贝叶斯分类器**
2. **实验目的**
3. 理解朴素贝叶斯分类器
4. 了解sklearn库
5. 了解pandas库
6. 进行其他分类器的比较
7. **实验内容**

在给定的数据集上，训练朴素贝叶斯模型近邻模型，输出预测结果及其准确度。

1. **实验步骤**

**1. 数据集准备及数据统计可视化**

数据加载：使用 pandas 读取数据集“肿瘤数据.xlsx”，包括6个特征维度和肿瘤性质的标签列，确保数据加载无误。

特征统计：使用直方图、箱线图对各特征的数值分布进行可视化分析。 绘制肿瘤性质（良性与恶性）的饼状图或条形图，直观展示分类分布比例。

特征之间的关系：使用热力图展示各特征之间的相关性，以便初步分析特征的重要性或冗余性。

**2. 模型训练**

数据划分： 使用 sklearn.model\_selection.train\_test\_split() 将数据集划分为训练集和测试集（例如训练集70%，测试集30%）。使用 sklearn.preprocessing.StandardScaler 对特征值进行标准化处理，确保不同特征在同一量纲上，以提高模型性能。

模型构建：选用 sklearn.naive\_bayes.GaussianNB 构建朴素贝叶斯模型，使用 fit() 函数进行模型训练。

**3. 模型预测与评估**

预测：使用训练好的模型对测试集进行预测，记录预测结果。

性能评估：使用 sklearn.metrics 提供的以下指标进行评估：准确率：accuracy\_score()，计算预测准确率。混淆矩阵：confusion\_matrix()，可视化分类结果的混淆矩阵，分析分类的正确与错误情况。分类报告：classification\_report()，展示精确率、召回率、F1分数等。

交叉验证：使用 cross\_val\_score() 方法进行交叉验证，验证模型的泛化性能。

**4. 结果分析**

性能解读：结合混淆矩阵和分类报告，分析模型对良性与恶性肿瘤分类的优劣势。对比交叉验证结果与测试集结果，分析模型是否存在过拟合或欠拟合现象。

模型优势与局限性：总结朴素贝叶斯模型在本实验中表现的优点（如训练速度快、计算简单）和可能的局限性（如假设特征独立）。

特征重要性讨论： 如果某些特征对分类贡献不大或呈现高冗余性，可讨论数据降维的可能性（如使用PCA）。

**5. 扩展探索**

特征选择：使用 sklearn.feature\_selection.SelectKBest 等方法进行特征筛选，观察特征减少对模型性能的影响。

与其他分类器对比： 选用其他简单分类器（如决策树、逻辑回归）对比朴素贝叶斯的性能。

1. **实验结果与分析**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文字体为 SimHei

plt.rcParams['axes.unicode\_minus'] = False # 解决负号显示问题

# 1. 数据加载

file\_path = '肿瘤数据.xlsx' # 确保该文件在代码工作目录下

try:

data = pd.read\_excel(file\_path)

print("数据加载成功！")

except FileNotFoundError:

print(f"未找到文件：{file\_path}")

raise

# 2. 数据检查与清洗

print("数据概览：")

print(data.head())

print("\n数据统计描述：")

print(data.describe())

print("\n检查缺失值：")

print(data.isnull().sum())

# 如果有缺失值，可选择填充或删除

# data = data.dropna() # 删除含有缺失值的行

# 或者选择填充：data.fillna(data.mean(), inplace=True)

# 3. 数据分布可视化

# 绘制肿瘤性质的分布图

plt.figure(figsize=(6, 4))

data['肿瘤性质'].value\_counts().plot(kind='bar', color=['skyblue', 'salmon'], alpha=0.7)

plt.title("肿瘤性质分布", fontsize=14)

plt.ylabel("患者数", fontsize=12)

plt.xlabel("肿瘤性质", fontsize=12)

plt.xticks(rotation=0)

plt.tight\_layout()

plt.show()

# 各特征的直方图

feature\_columns = [col for col in data.columns if col not in ['肿瘤性质']]

plt.figure(figsize=(12, 8))

for i, feature in enumerate(feature\_columns, 1):

plt.subplot(2, 3, i)

plt.hist(data[feature], bins=20, color='dodgerblue', alpha=0.7)

plt.title(feature, fontsize=12)

plt.xlabel("值", fontsize=10)

plt.ylabel("频数", fontsize=10)

plt.tight\_layout()

plt.show()

# 4. 特征之间的相关性分析

plt.figure(figsize=(10, 8))

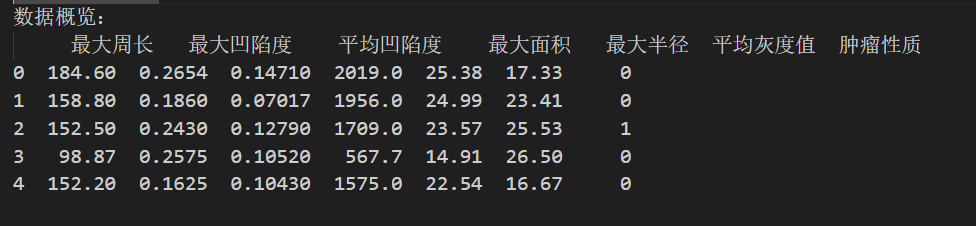
correlation\_matrix = data[feature\_columns].corr()

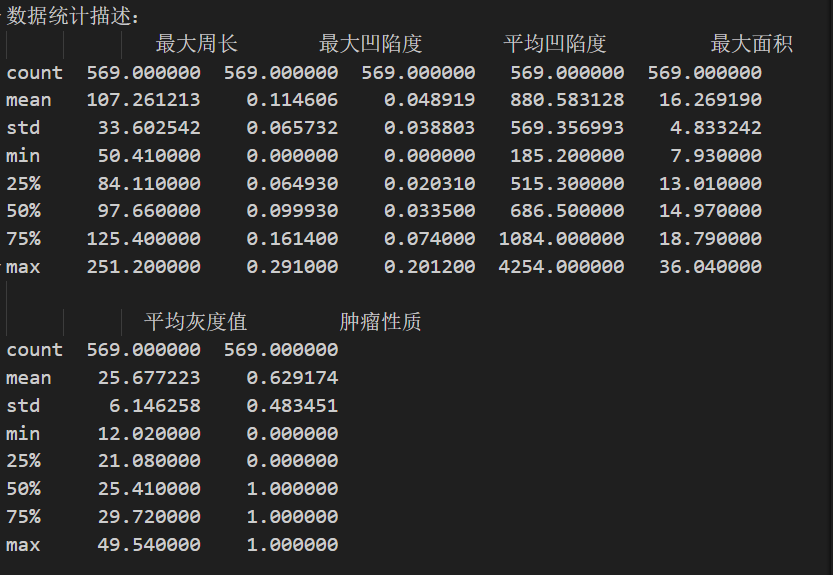
sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)

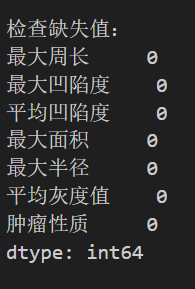
plt.title("特征相关性热力图", fontsize=14)

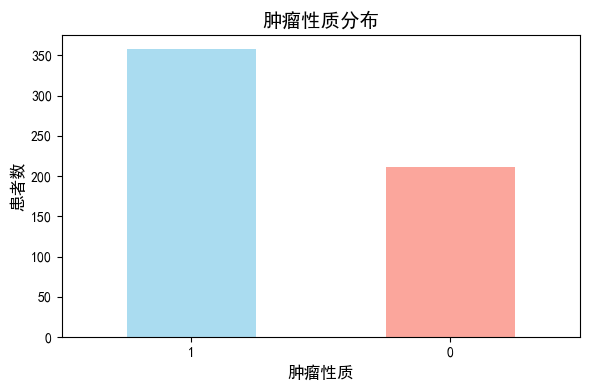
plt.tight\_layout()

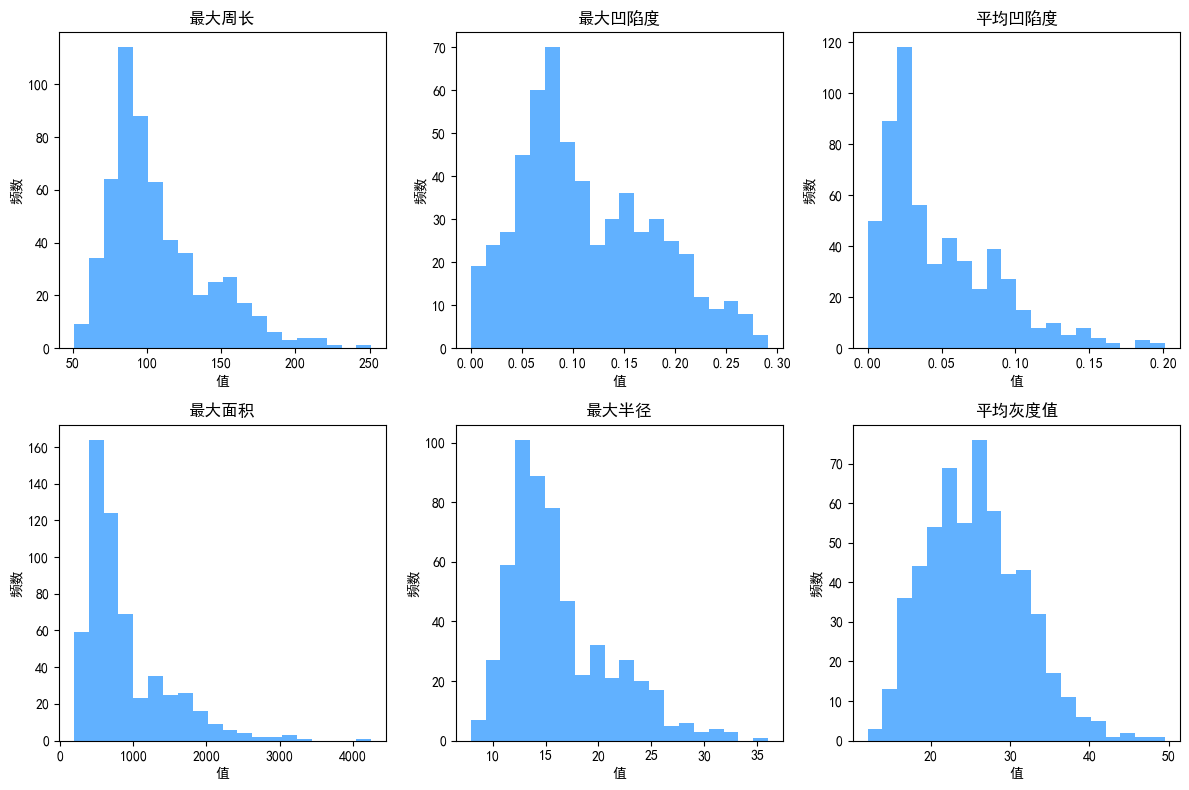
plt.show()

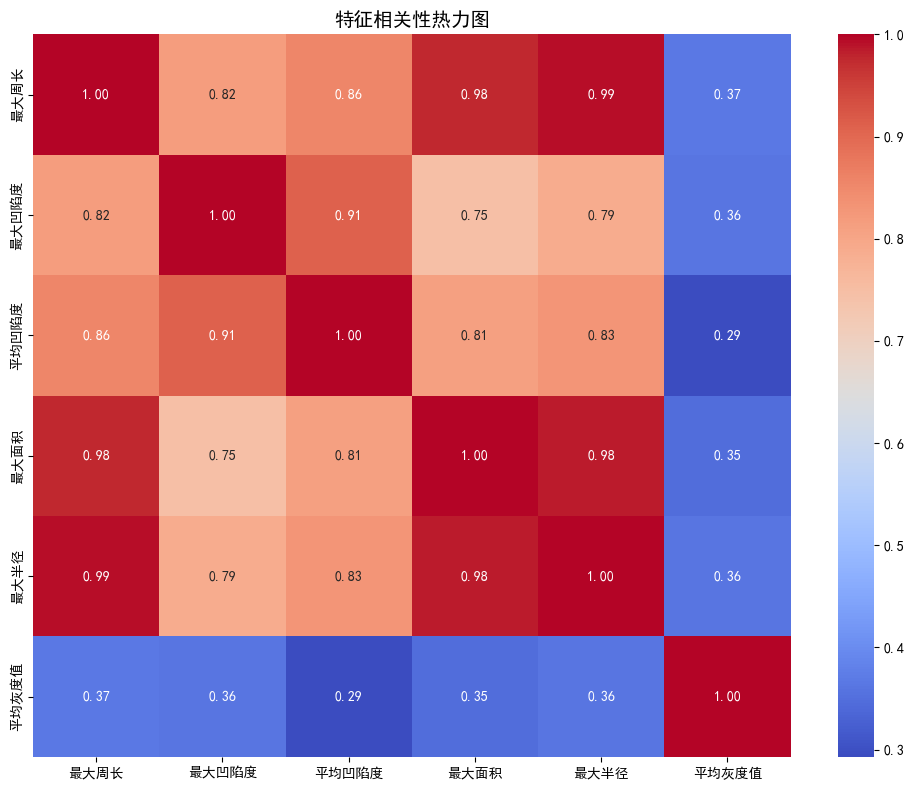












import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB

import time

# 加载数据

data = pd.read\_excel("肿瘤数据.xlsx")

# 提取特征和标签

X = data.iloc[:, :-1]  # 假设最后一列是标签

y = data.iloc[:, -1]   # 假设最后一列是标签

# 数据划分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

print(f"训练集样本数：{len(X\_train)}, 测试集样本数：{len(X\_test)}")

# 特征标准化

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# 模型训练

model = GaussianNB()

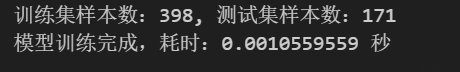
start\_time = time.time()

model.fit(X\_train, y\_train)

end\_time = time.time()

betyes\_time=end\_time-start\_time

print(f"模型训练完成，耗时：{end\_time - start\_time:.10f} 秒")



from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

import seaborn as sns

import matplotlib.pyplot as plt

# 使用模型进行预测

y\_pred = model.predict(X\_test)

# 计算准确率

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"模型预测准确率：{accuracy:.4f}")

# 混淆矩阵

cm = confusion\_matrix(y\_test, y\_pred)

print("混淆矩阵：")

print(cm)

# 可视化混淆矩阵

plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["良性", "恶性"], yticklabels=["良性", "恶性"])

plt.xlabel("预测值")

plt.ylabel("实际值")

plt.title("混淆矩阵")

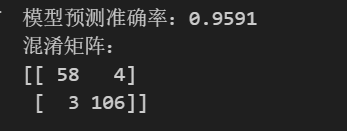
plt.show()

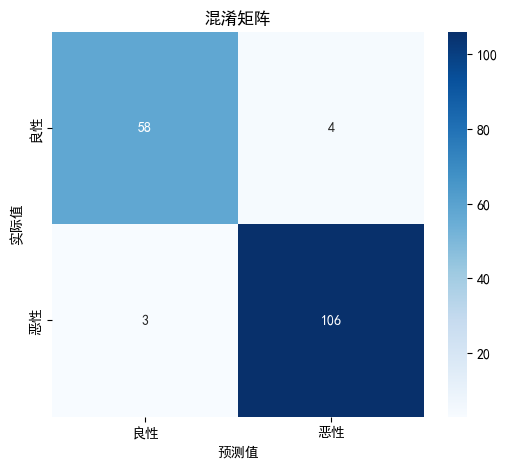
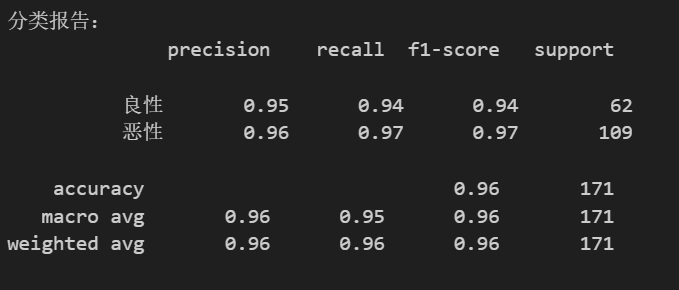
# 分类报告

report = classification\_report(y\_test, y\_pred, target\_names=["良性", "恶性"])

print("分类报告：")

print(report)



from sklearn.model\_selection import cross\_val\_score

import numpy as np

# 输出混淆矩阵的分析结果

print("混淆矩阵分析：")

true\_positive = cm[1, 1]

true\_negative = cm[0, 0]

false\_positive = cm[0, 1]

false\_negative = cm[1, 0]

print(f"正确分类良性肿瘤（True Negative, TN）：{true\_negative}")

print(f"正确分类恶性肿瘤（True Positive, TP）：{true\_positive}")

print(f"将良性误判为恶性（False Positive, FP）：{false\_positive}")

print(f"将恶性误判为良性（False Negative, FN）：{false\_negative}")

# 分析误判原因

if false\_positive + false\_negative > 0:

    print("\n误判分析：")

    print(f"误判的总样本数：{false\_positive + false\_negative}")

    print(f"误判率：{(false\_positive + false\_negative) / len(y\_test):.4f}")

else:

    print("\n模型没有误判样本。")

# 交叉验证评估模型稳定性

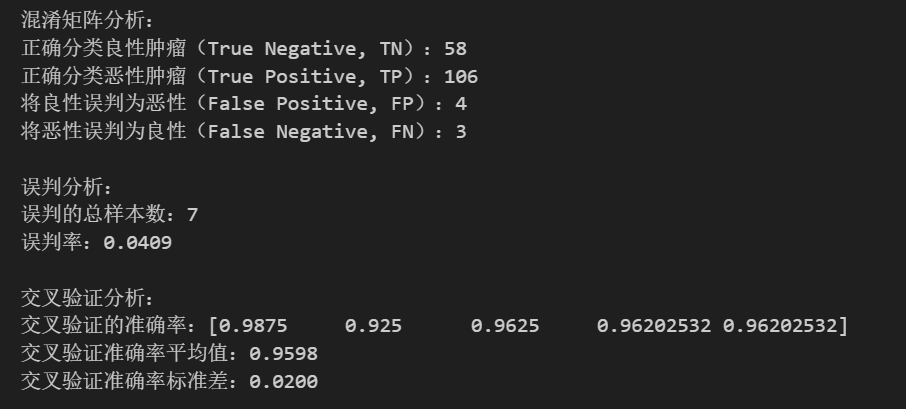
print("\n交叉验证分析：")

cv\_scores = cross\_val\_score(model, X\_train, y\_train, cv=5, scoring='accuracy')

print(f"交叉验证的准确率：{cv\_scores}")

print(f"交叉验证准确率平均值：{np.mean(cv\_scores):.4f}")

print(f"交叉验证准确率标准差：{np.std(cv\_scores):.4f}")



import time

from sklearn.feature\_selection import SelectKBest, mutual\_info\_classif

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import roc\_auc\_score

# 1. 特征选择：使用互信息法选择最重要的特征

print("特征选择：")

selector = SelectKBest(mutual\_info\_classif, k=4)  # 选择4个最相关的特征

X\_train\_selected = selector.fit\_transform(X\_train, y\_train)

X\_test\_selected = selector.transform(X\_test)

selected\_features = selector.get\_support(indices=True)

print(f"选择的特征索引：{selected\_features}")

# 2. 训练逻辑回归模型

print("\n逻辑回归模型训练：")

start\_time = time.time()

lr\_model = LogisticRegression(random\_state=42, max\_iter=1000)

lr\_model.fit(X\_train\_selected, y\_train)

end\_time = time.time()

lr\_training\_time = end\_time - start\_time

lr\_pred = lr\_model.predict(X\_test\_selected)

lr\_accuracy = accuracy\_score(y\_test, lr\_pred)

lr\_auc = roc\_auc\_score(y\_test, lr\_model.predict\_proba(X\_test\_selected)[:, 1])

print(f"逻辑回归模型准确率：{lr\_accuracy:.4f}, AUC：{lr\_auc:.4f}")

print(f"逻辑回归模型训练时间：{lr\_training\_time:.4f} 秒")

# 3. 训练随机森林模型

print("\n随机森林模型训练：")

start\_time = time.time()

rf\_model = RandomForestClassifier(random\_state=42, n\_estimators=100)

rf\_model.fit(X\_train\_selected, y\_train)

end\_time = time.time()

rf\_training\_time = end\_time - start\_time

rf\_pred = rf\_model.predict(X\_test\_selected)

rf\_accuracy = accuracy\_score(y\_test, rf\_pred)

rf\_auc = roc\_auc\_score(y\_test, rf\_model.predict\_proba(X\_test\_selected)[:, 1])

print(f"随机森林模型准确率：{rf\_accuracy:.4f}, AUC：{rf\_auc:.4f}")

print(f"随机森林模型训练时间：{rf\_training\_time:.4f} 秒")

# 4. 比较朴素贝叶斯模型的表现

nb\_pred = model.predict(X\_test)

nb\_accuracy = accuracy\_score(y\_test, nb\_pred)

nb\_auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1])

print(f"\n朴素贝叶斯模型准确率：{nb\_accuracy:.4f}, AUC：{nb\_auc:.4f}")

# 5. 总结结果

print("\n模型表现总结：")

print(f"逻辑回归准确率：{lr\_accuracy:.4f}, 随机森林准确率：{rf\_accuracy:.4f}, 朴素贝叶斯准确率：{nb\_accuracy:.4f}")

print(f"逻辑回归AUC：{lr\_auc:.4f}, 随机森林AUC：{rf\_auc:.4f}, 朴素贝叶斯AUC：{nb\_auc:.4f}")

print(f"逻辑回归训练时间：{lr\_training\_time:.4f} 秒, 随机森林训练时间：{rf\_training\_time:.4f} 秒")

# 6. 可视化比较

models = ['朴素贝叶斯', '逻辑回归', '随机森林']

accuracies = [nb\_accuracy, lr\_accuracy, rf\_accuracy]

aucs = [nb\_auc, lr\_auc, rf\_auc]

training\_times = [betyes\_time, lr\_training\_time, rf\_training\_time]  # 朴素贝叶斯训练时间可忽略不计

plt.figure(figsize=(15, 5))

# 准确率对比

plt.subplot(1, 3, 1)

plt.bar(models, accuracies, color=['blue', 'green', 'orange'])

plt.title("模型准确率比较")

plt.ylabel("准确率")

plt.ylim(0, 1)

# AUC对比

plt.subplot(1, 3, 2)

plt.bar(models, aucs, color=['blue', 'green', 'orange'])

plt.title("模型AUC比较")

plt.ylabel("AUC")

plt.ylim(0, 1)

# 训练时间对比

plt.subplot(1, 3, 3)

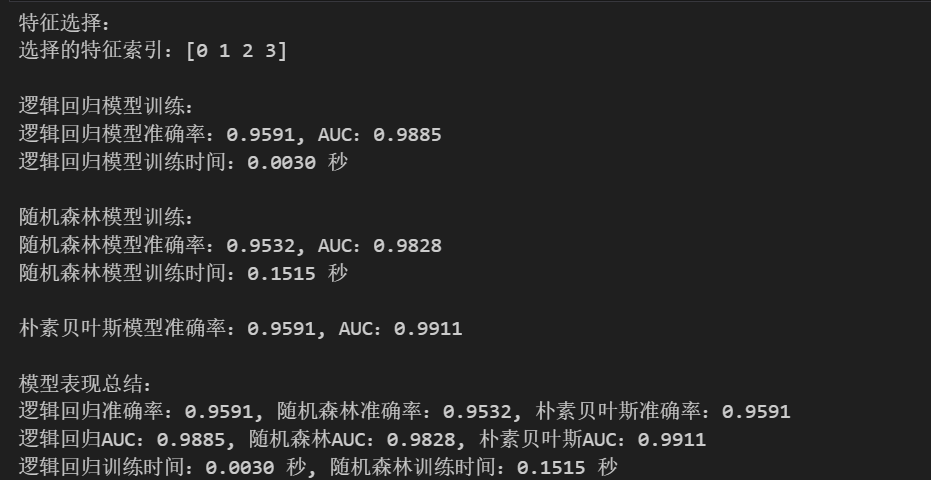
plt.bar(models, training\_times, color=['blue','green', 'orange'])

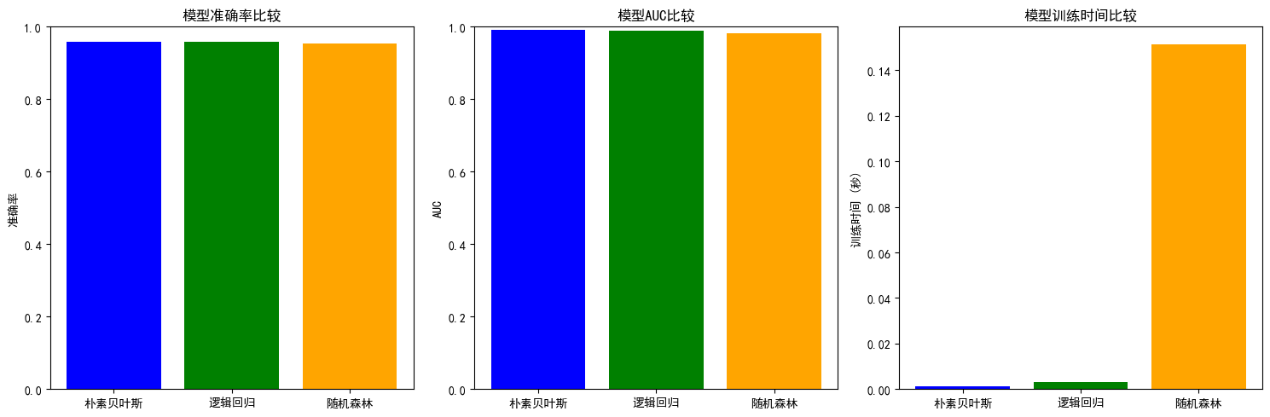
plt.title("模型训练时间比较")

plt.ylabel("训练时间 (秒)")

plt.tight\_layout()

plt.show()





从结果分析来看，模型在分类任务中表现较好，但仍有优化空间。

首先，从混淆矩阵的结果来看，模型能够准确分类大多数样本，其中正确分类的良性肿瘤为 58 个，恶性肿瘤为 106 个，总体上显示出较高的分类能力。然而，模型存在少量误判，将 4 个良性肿瘤误判为恶性肿瘤，以及 3 个恶性肿瘤误判为良性肿瘤。误判率为 0.0409，说明总体上误判情况较少，但对于恶性肿瘤误判为良性（False Negative）的情况需要特别关注，因为这可能会对患者造成更严重的后果。

交叉验证结果表明，模型的性能在不同的数据划分上具有较高的稳定性。交叉验证的准确率在 0.925 到 0.9875 之间，平均值为 0.9598，标准差为 0.0200。这表明模型在不同训练集和测试集划分下的表现波动较小，具有良好的泛化能力。

总体来看，模型对肿瘤的分类能力较为优秀，但可以考虑以下优化方向：

1. 减少误判：可以通过分析误判样本的特征，进一步优化模型，例如调整模型的决策边界或通过采样方法平衡数据集。

2. 提高召回率：特别关注恶性肿瘤的召回率，确保尽量减少 False Negative 的发生，可以尝试调节分类阈值。

3. 探索更复杂的模型：当前模型是朴素贝叶斯，可以尝试逻辑回归、随机森林或神经网络等更复杂的分类器，可能进一步提升性能。

1. **实验总结**

本次实验主要学习了朴素贝叶斯的构成原理，并通过代码实操进一步理解了算法的应用。同时，我们还将朴素贝叶斯与随机森林进行了对比分析，观察两种算法在处理同一数据集时的表现差异。实验结果显示，朴素贝叶斯具有计算高效、实现简单的优点，但在数据特征间相关性较强时，其效果可能不够理想。而随机森林由于结合了多棵决策树，在应对复杂数据时表现更为出色，但运行时间相对更长。

需要注意的是，实验过程中存在数据预处理不足的问题，导致模型的表现未能充分展现其潜力。具体来看，数据的清洗和特征处理环节还需要进一步优化，比如对缺失值、异常值的处理较为简单，特征提取也不够充分。此外，实验的结果评价仅关注了准确率，未从其他指标全面评估模型的优劣。

总体而言，本次实验帮助我们进一步理解了朴素贝叶斯和随机森林的特点和适用场景，同时也暴露了在数据处理和模型优化方面的不足。未来实验中需要更加注重数据预处理的细节，并尝试引入更多评价指标，以提升实验结果的科学性和全面性。