Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Лабораторная работа №3

по дисциплине

«Методы машинного обучения»

Выполнил:

студент группы ИУ5И-23М

Цзян Юхуэй

Москва — 2024 г.

Цель лабораторной работы

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание

1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.

2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

① масштабирование признаков (не менее чем тремя способами);

② обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);

③ обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);

④ отбор признаков:

a. один метод из группы методов фильтрации (filter methods);

b. один метод из группы методов обертывания (wrapper methods);

c. один метод из группы методов вложений (embedded methods).

Основной раздел кода

Для масштабирования признаков мы используем набор данных IRIS. При этом использовались три метода масштабирования признаков: масштабирование по мини-максимуму, нормализация и нормализация соответственно.

```
import pandas as pd

from sklearn.datasets import load_files

# 下载并加载 IMDB 电影评论数据集

!wget -O aclImdb_v1.tar.gz https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

!tar -xf aclImdb_v1.tar.gz

# 加载训练数据

train_data = load_files('aclImdb/train', categories=['pos', 'neg'], shuffle=True, random_state=42)

test_data = load_files('aclImdb/test', categories=['pos', 'neg'], shuffle=True, random_state=42)

# 分离数据和标签

X_train, y_train = train_data.data, train_data.target

X_test, y_test = test_data.data, test_data.target
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.svm import LinearSVC

from sklearn.linear_model import LogisticRegression

from sklearn.pipeline import make_pipeline

from sklearn.metrics import classification_report

# 创建不同的管道

pipelines = {

    'CountVectorizer + LinearSVC': make_pipeline(CountVectorizer(), LinearSVC(random_state=42)),

    'CountVectorizer + LogisticRegression': make_pipeline(CountVectorizer(), LogisticRegression(random_state=42)),

    'TfidfVectorizer + LinearSVC': make_pipeline(TfidfVectorizer(), LinearSVC(random_state=42)),
```

```
    'TfidfVectorizer + LogisticRegression': make_pipeline(TfidfVectorizer(), LogisticRegression(random_state=42))
}
# 训练和评估每个管道
for name, pipeline in pipelines.items():
    print('='*80)
    print(f'Training and evaluating pipeline: {name}')
    print('='*80)
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)
    print(classification_report(y_test, y_pred))
    print('-' * 80)
```

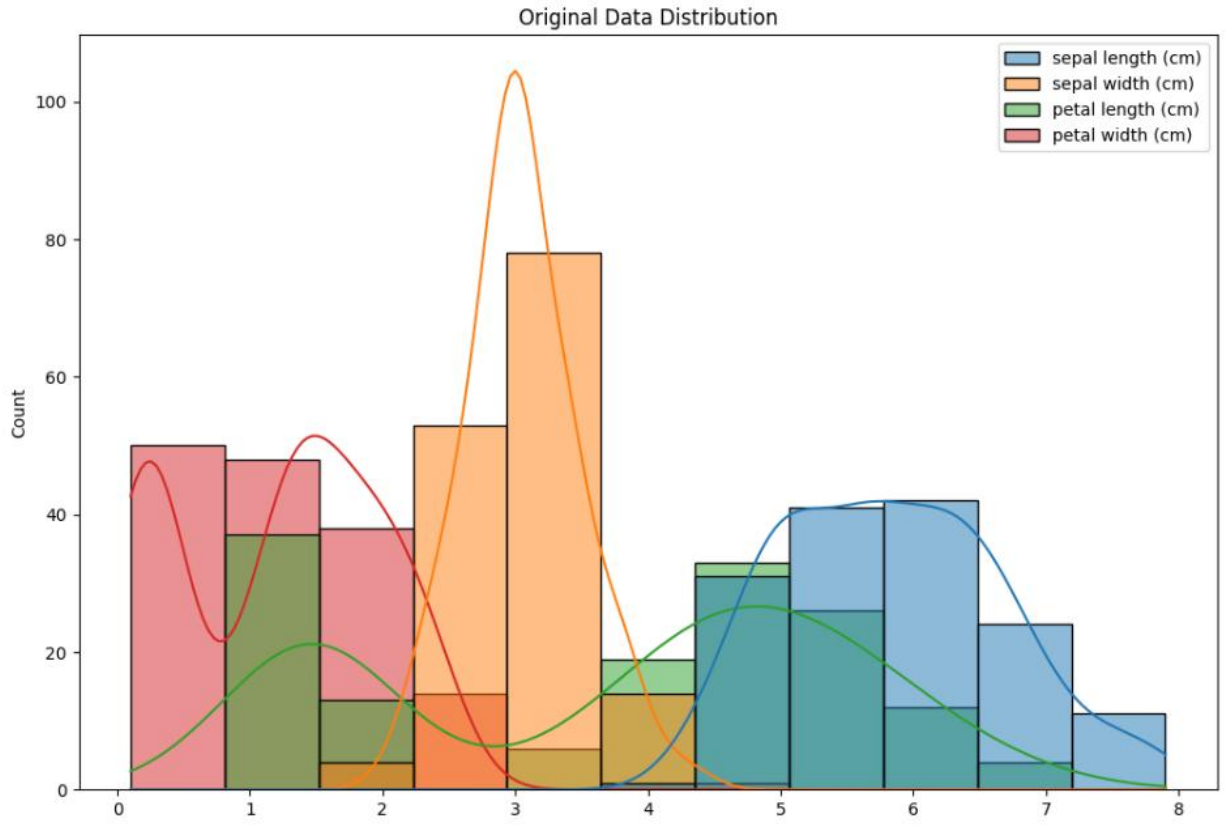

Рис 1. Распределение исходных данных.

Минимальный Максимальный Масштаб:

```
# 最小最大缩放

min_max_scaler = MinMaxScaler()

iris_min_max_scaled = min_max_scaler.fit_transform(iris_df)

# 绘制最小最大缩放后的数据分布

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

sns.histplot(data=pd.DataFrame(iris_min_max_scaled, columns=iris.feature_names), kde=True)

plt.title('Min-Max Scaled Data Distribution')

plt.show()
```

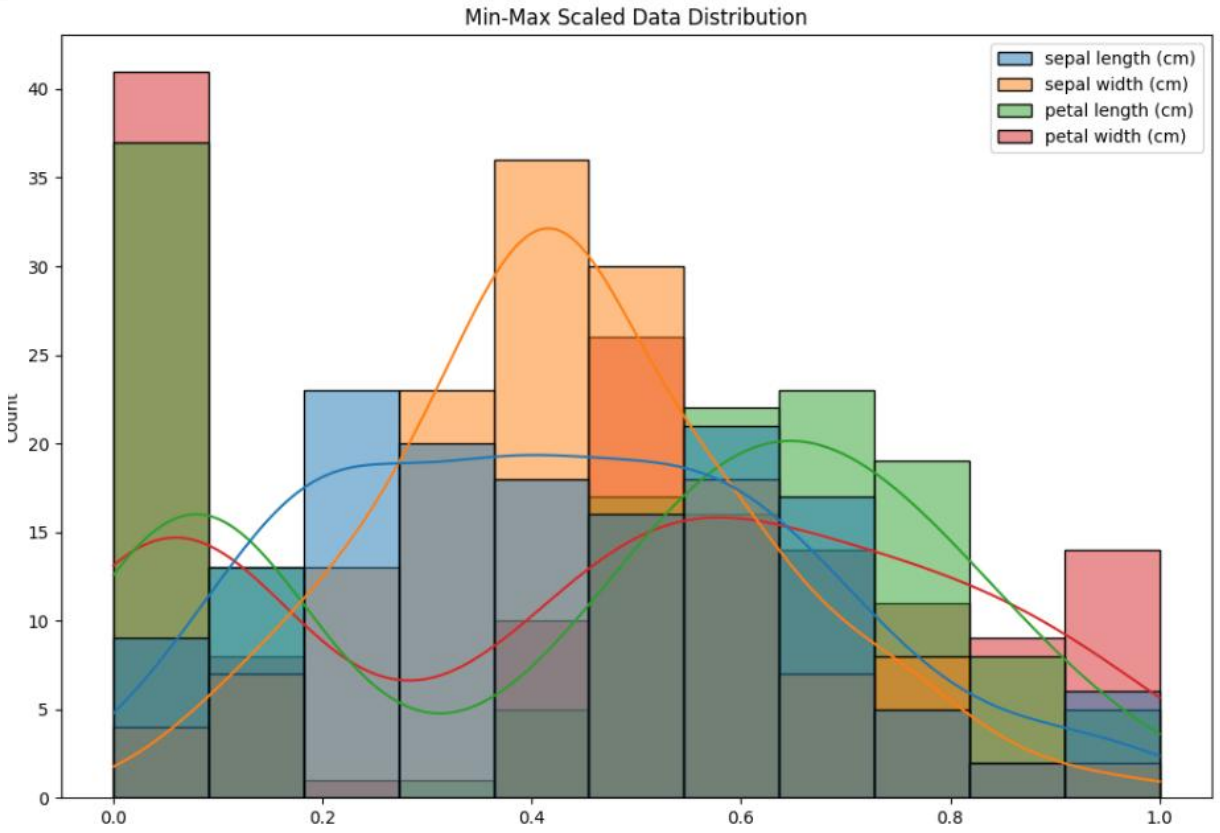

Рис 2. Распределение данных после масштабирования min-max.

Стандартизированный:

```
# 标准化
```

```
standard_scaler = StandardScaler()

iris_standard_scaled = standard_scaler.fit_transform(iris_df)

# 绘制标准化后的数据分布

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

sns.histplot(data=pd.DataFrame(iris_standard_scaled, columns=iris.feature_names), kde=True)

plt.title('Standardized Data Distribution')

plt.show()
```

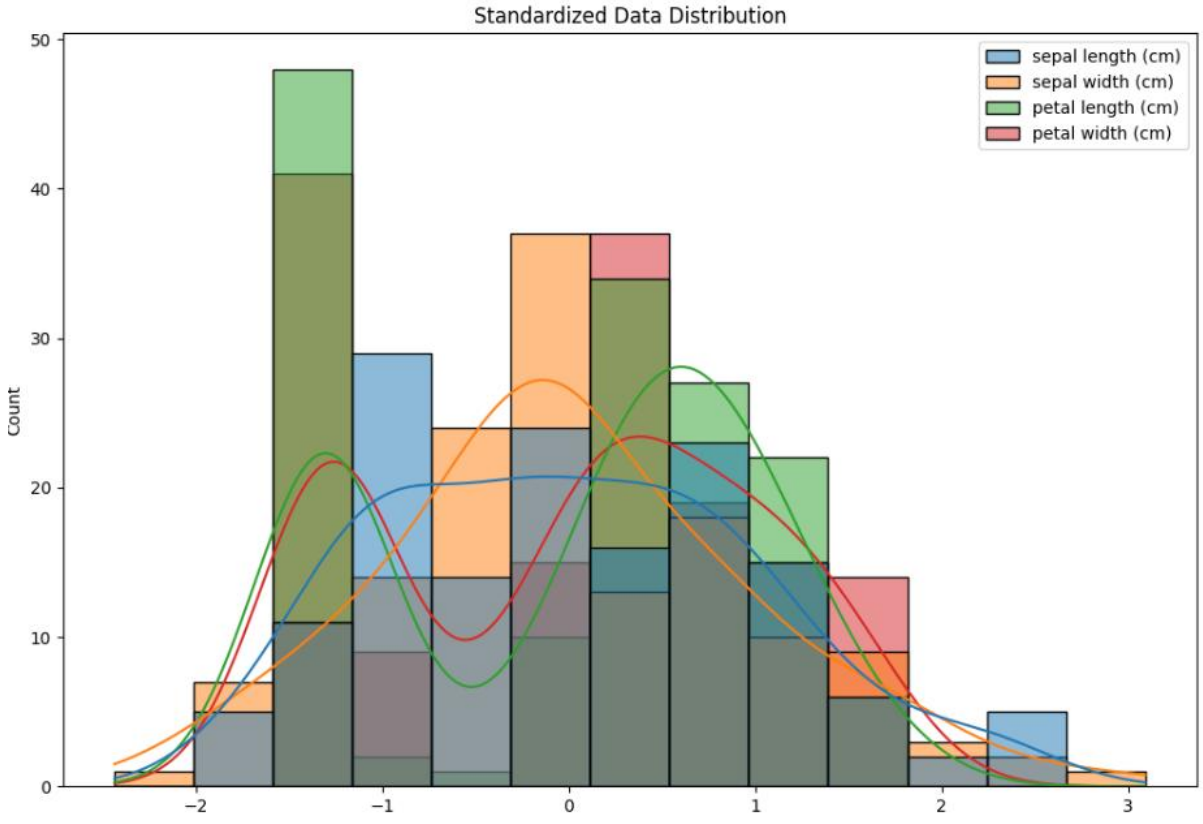

Рис 3. Распределение данных после нормализации.

Нормализация:

```
# 归一化

normalizer = Normalizer()

iris_normalized = normalizer.fit_transform(iris_df)
```

```
# 绘制归一化后的数据分布

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

sns.histplot(data=pd.DataFrame(iris_normalized, columns=iris.feature_names), kde=True)

plt.title('Normalized Data Distribution')

plt.show()
```

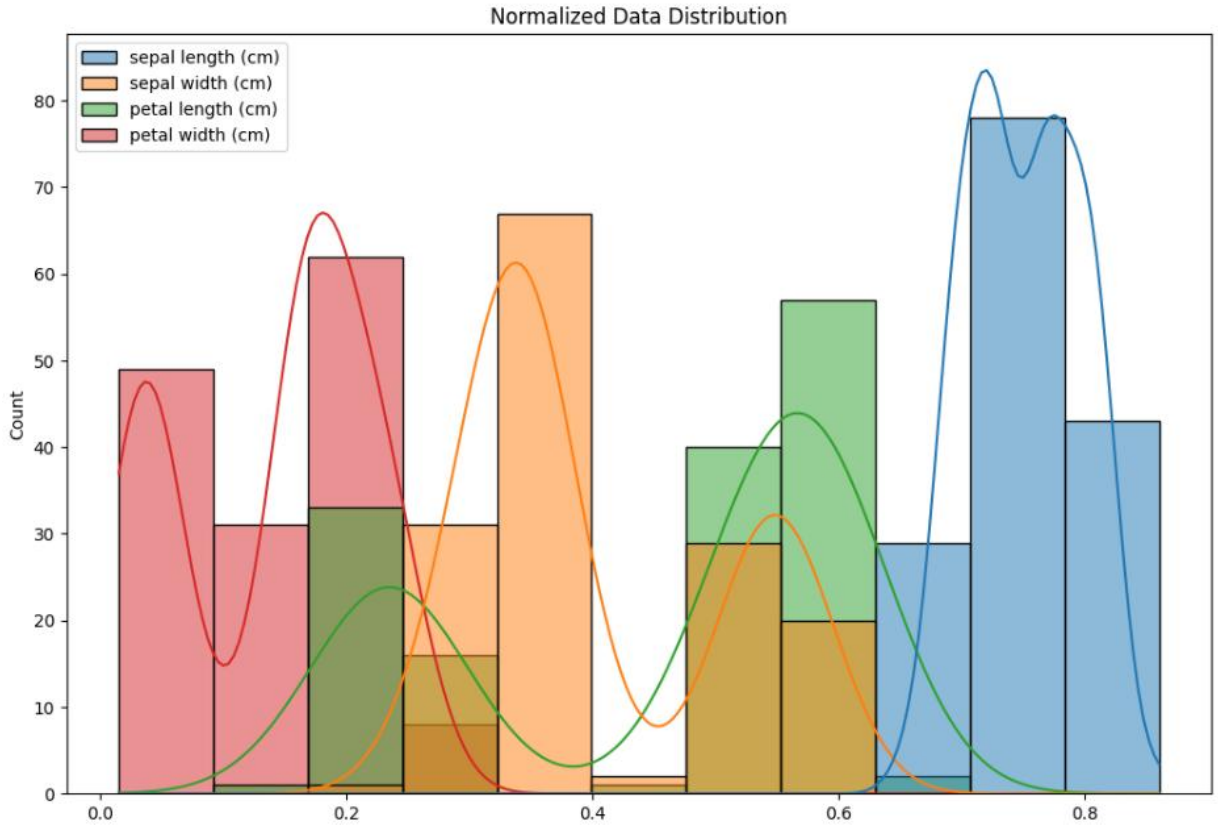

Рис 4. Нормализованное распределение данных.

Часть 2. Обработку выбросов для числовых признаков

Для эмиссионной обработки числовых признаков мы используем набор данных IRIS.

```
import numpy as np

# 计算 z-scores

iris_z_scores = np.abs((iris_df - iris_df.mean()) / iris_df.std())

# 设置阈值，定义异常值

threshold = 3

# 绘制原始数据的箱线图

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

sns.boxplot(data=iris_df)

plt.title('Original Data Boxplot')

plt.show()
```

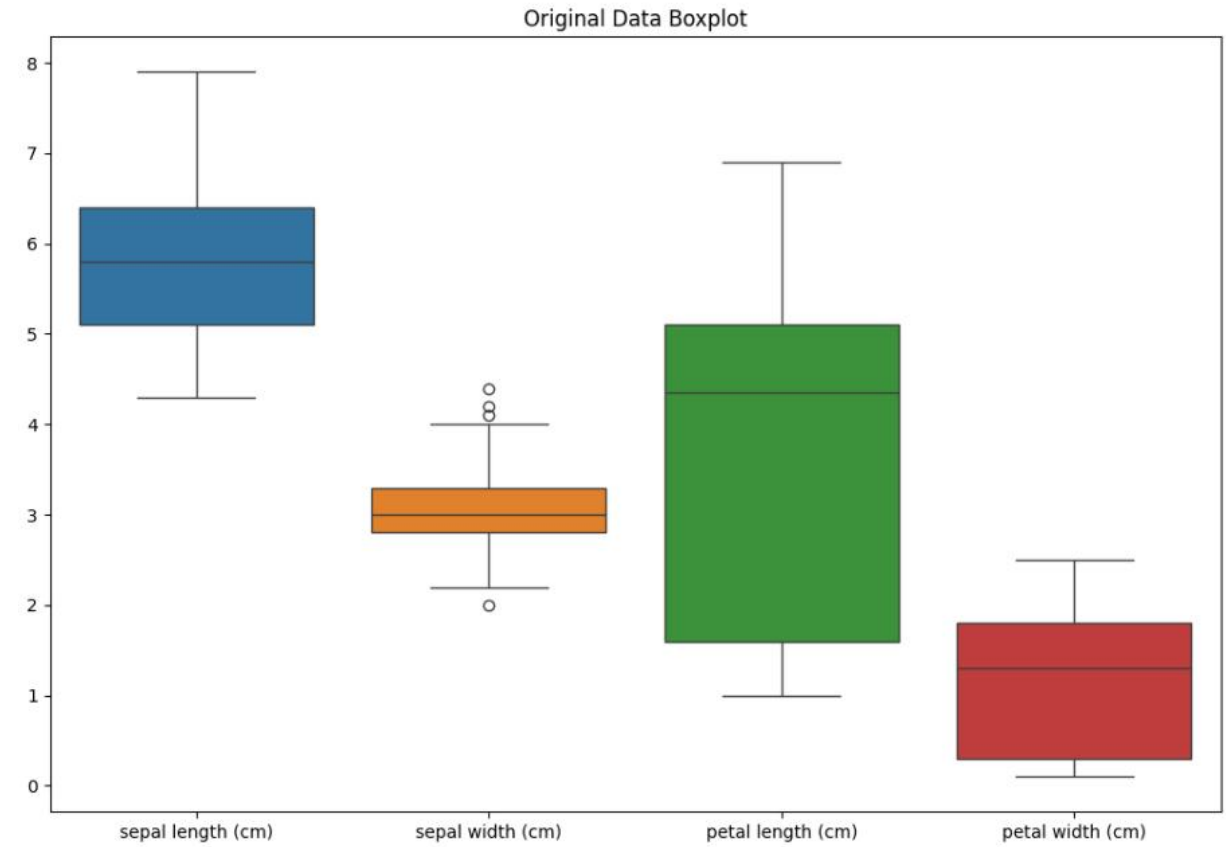

Рис 5. Боксплот исходных данных.

Удалить провалы:

```
# 删除异常值

iris_no_outliers = iris_df[(iris_z_scores < threshold).all(axis=1)]

# 绘制删除异常值后的数据箱线图

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

sns.boxplot(data=iris_no_outliers)

plt.title('Data Boxplot After Removing Outliers')

plt.show()
```

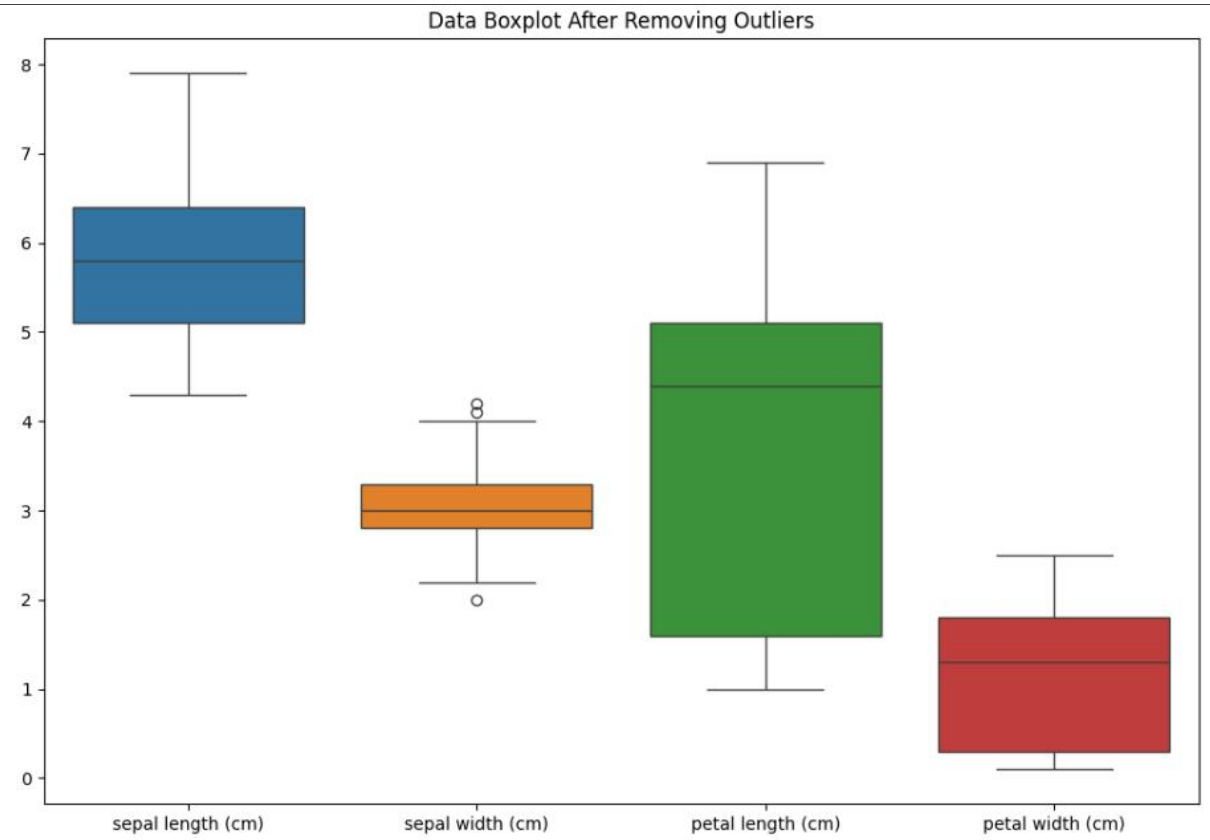

Рис 6. Бокс-диаграмма после удаления выбросов.

Заменить провалы медианой:

```
# 替换异常值（用中位数替换）
```

```
iris_replaced_outliers = iris_df.copy()

median = iris_df.median()

for col in iris_df.columns:

    iris_replaced_outliers[col] = np.where(iris_z_scores[col] > threshold, median[col], iris_df[col])

# 绘制替换异常值后的数据箱线图

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

sns.boxplot(data=iris_replaced_outliers)

plt.title('Data Boxplot After Replacing Outliers')

plt.show()
```

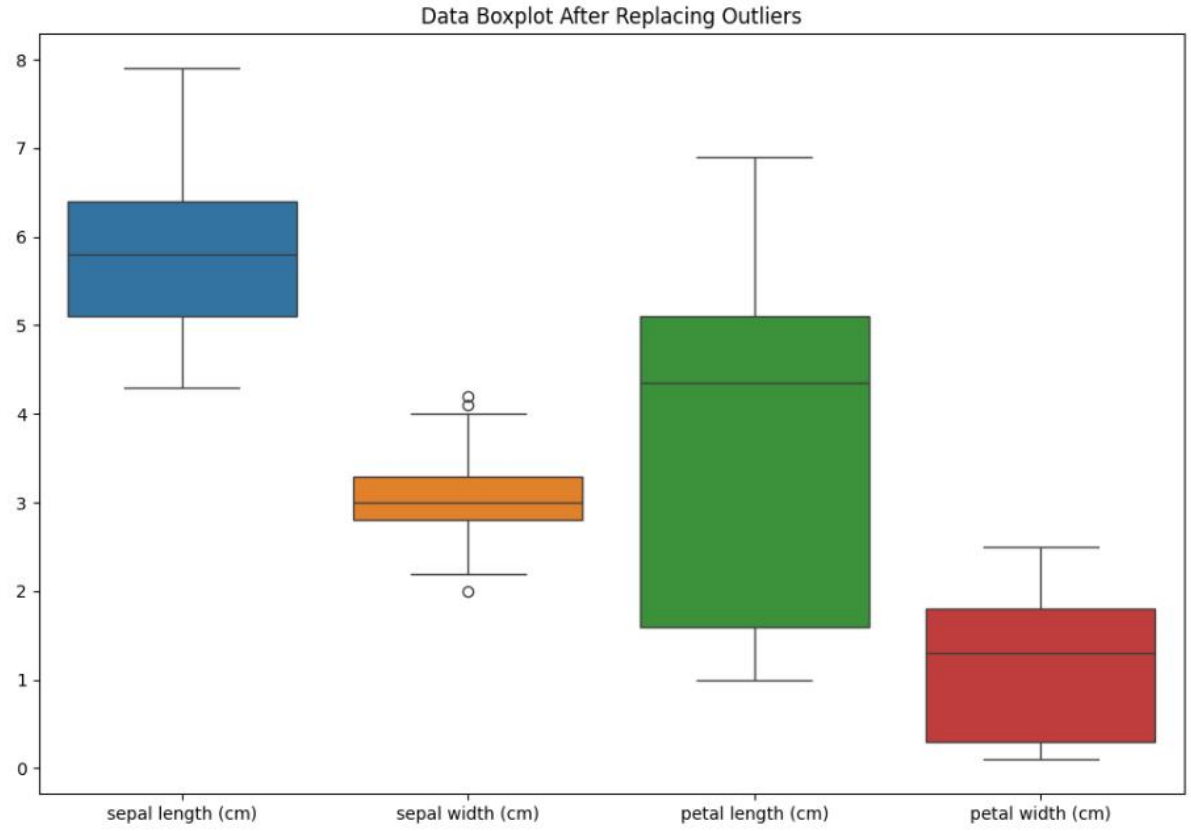

Рис 7. Боксплот после замены промахов.

Часть 3. Обработку по крайней мере одного нестандартного признака

Для обработки нестандартных признаков мы использовали набор данных "Breast Cancer Wisconsin (Diagnostic) Data Set" из библиотеки машинного обучения UCI. Мы смоделировали идентификатор пациента（patient_id）, чтобы добавить нестандартные признаки в набор данных.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load_breast_cancer

from sklearn.feature_selection import SelectKBest, chi2, RFE

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear_model import LogisticRegression

# 加载 Breast Cancer Wisconsin 数据集

breast_cancer = load_breast_cancer()

breast_cancer_df = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)

# 添加一个非标准特征（如患者 ID，这里我们模拟一个）

breast_cancer_df['patient_id'] = np.arange(1, breast_cancer_df.shape[0] + 1)

# 显示添加非标准特征后的数据分布

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

sns.histplot(data=breast_cancer_df[['mean radius', 'patient_id']], kde=True)

plt.title('Data Distribution with Non-Standard Feature (patient_id)')

plt.show()
```

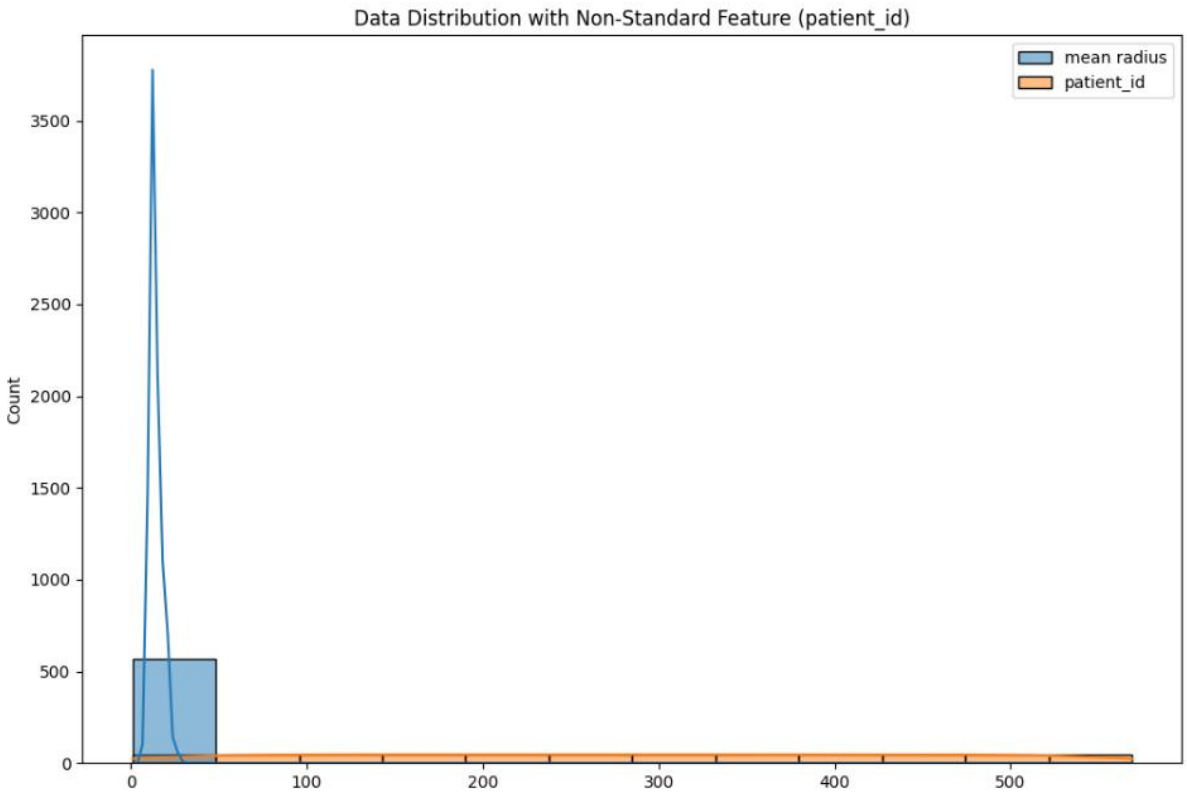

Рис 8. Распределение данных с нестандартной характеристикой (patient_id).

Часть 4. Отбор признаков

Мы рассчитали значимость признаков с помощью теста хи-квадрат. Методы фильтрации:

```
# 过滤方法
select_k_best = SelectKBest(chi2, k=10)
X_k_best = select_k_best.fit_transform(X, y)
selected_features_k_best_indices = select_k_best.get_support(indices=True)
selected_features_k_best = breast_cancer_df.columns[selected_features_k_best_indices]
plt.subplot(1, 3, 1)
sns.barplot(x=select_k_best.scores_[selected_features_k_best_indices], y=selected_features_k_best)
plt.title('Feature Importance (Filter Method)')
```

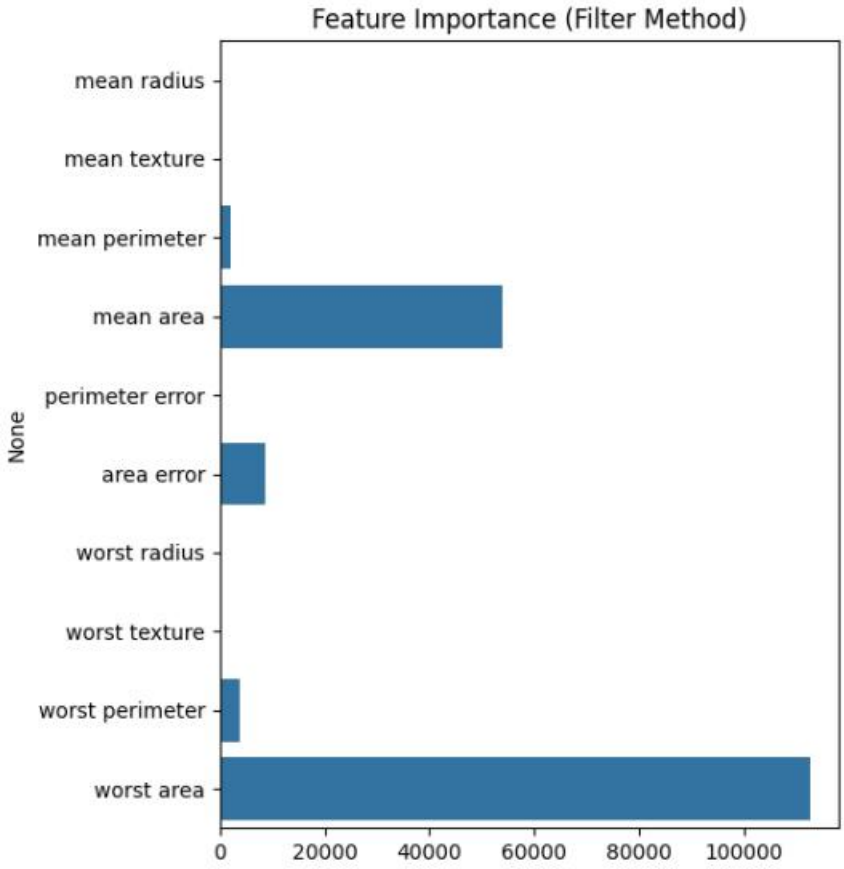

Рис 9. Важность использования фильтров для выбора характеристик.

Для отбора признаков использовалась модель логистической регрессии и алгоритм исключения признаков. Методы упаковки:

```
# 包裹方法
log_reg = LogisticRegression(max_iter=1000)
```

```
rfe = RFE(log_reg, n_features_to_select=10)
X_rfe = rfe.fit_transform(X, y)
selected_features_rfe_indices = rfe.get_support(indices=True)
selected_features_rfe = breast_cancer_df.columns[selected_features_rfe_indices]
plt.subplot(1, 3, 2)
sns.barplot(x=sorted_coefs[sorted_indices][:len(selected_features_rfe_indices)], y=selected_features_rfe[sorted_indices])
plt.title('Feature Importance (Wrapper Method)')
```

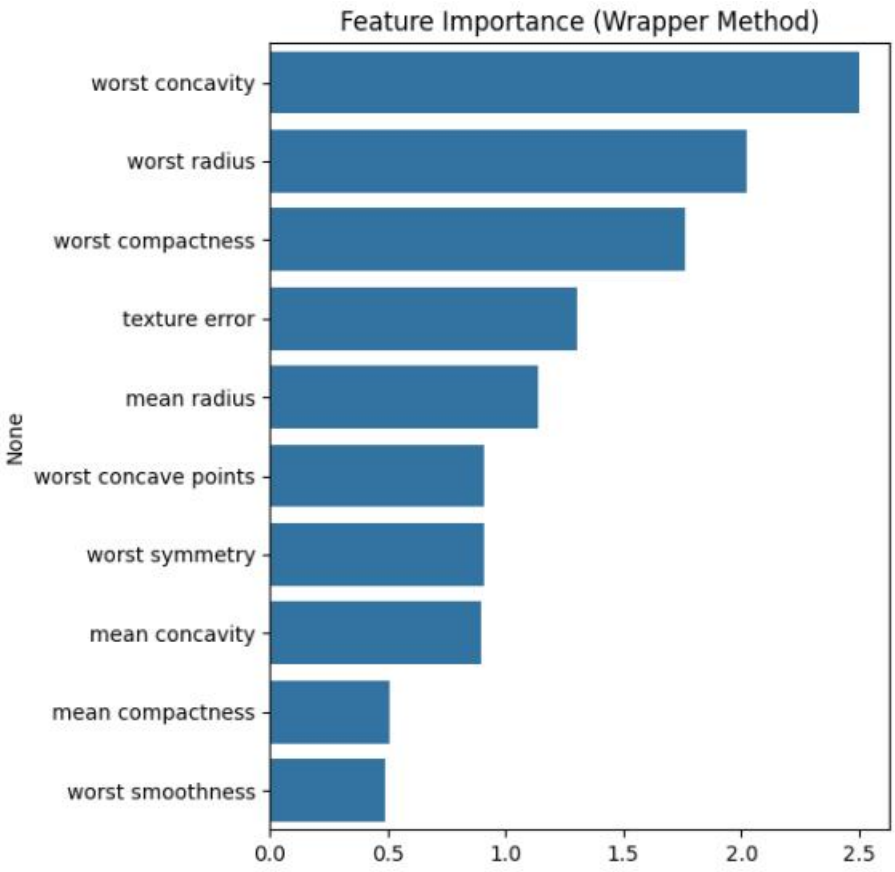

Рис 10. Важность использования методов парцеллирования для отбора признаков.

Мы использовали модель случайного леса и отбирали признаки по их важности. Методы встраивания:

```
# 嵌入方法
plt.subplot(1, 3, 3)
sns.barplot(x=importances[indices], y=selected_features_rf)
plt.title('Feature Importance (Embedded Method)')
plt.tight_layout()
```

```
plt.show()
```

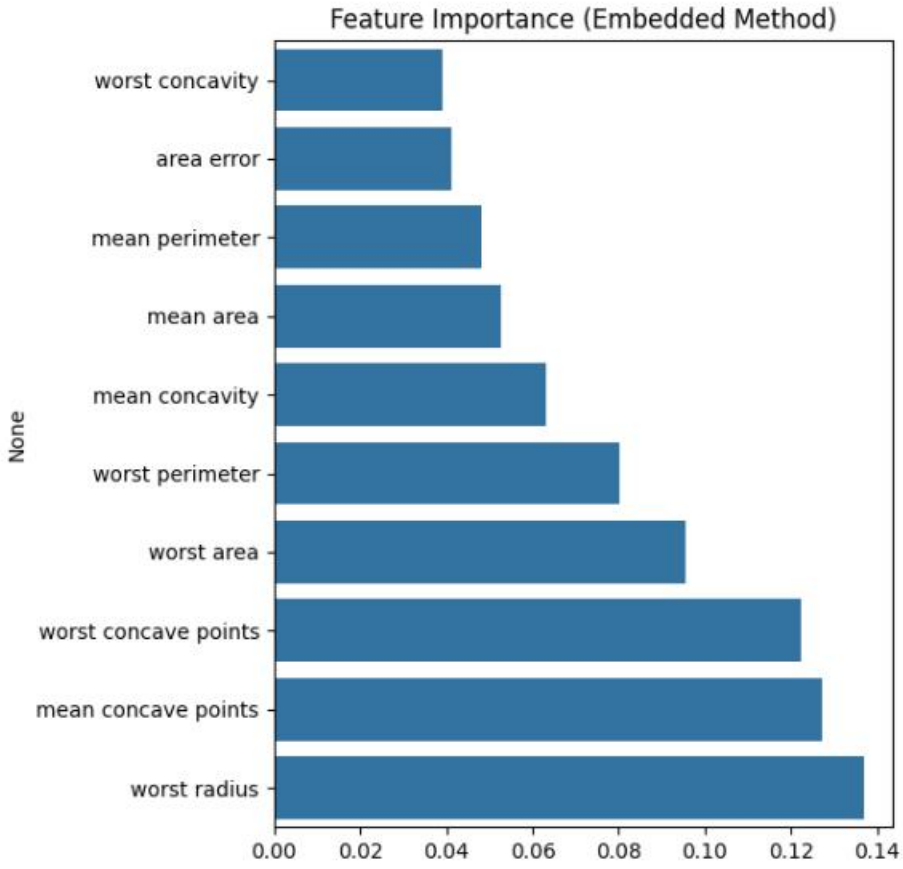

Рис 11. Важность использования методов встраивания для отбора признаков.