РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: Интеллектуальный анализ данных

Студент: Сатлихана Петрити

Группа: НПИбд-02-21

Москва 2024

Вариант 8

Breast Cancer Wisconsin (Diagnostic) Data Set

Название файла: wdbc.data

Ссылка: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Первый признак: perimeter (столбец No 5)

Второй признак: area (столбец No 6)

Класс: Outcome (столбец No 1)

Алгоритмы: K-means, Agglomerative Clustering, DBSCAN, Gaussian Mixture Model

Меры качества: F-мера, парные меры TP, FN, FP, TN, индекс Жаккара

Найти лучший алгоритм кластеризации относительно меры качества: F-мера

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

1. Считайте из заданного набора данных репозитария UCI значения двух признаков и метки класса.

```
import pandas as pd
```

```
# # Загрузка набора данных
path = '/content/drive/MyDrive/University/Viti IV/Интеллектуальный анализ данных/Lab 3/wdbc.data'
data = pd.read_csv(path, header=None)
perimeter = data[4]
area = data[5]
outcome = data[1]
print("Первый признак: perimeter (столбец No 5):")
print(perimeter.head())
print("\nВторой признак: area (столбец No 6):")
print(area.head())
print("\nКласс: Outcome (столбец No 1):")
print(outcome.head())
```

```
Первый признак: perimeter (столбец No 5):
0    122.80
1    132.90
```

```
2    130.00
3     77.58
4    135.10
Name: 4, dtype: float64

Второй признак: area (столбец No 6):
0    1001.0
1    1326.0
2    1203.0
3     386.1
4    1297.0
Name: 5, dtype: float64

Класс: Outcome (столбец No 1):
0    M
1    M
2    M
3    M
4    M
Name: 1, dtype: object
```

2. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если в признаках имеются пропущенные значения, то замените их на медианные значения того класса, к которому относится запись с пропущенным значением в признаке.

```
# Удаление записей с пропущенными метками класса
data = data.dropna(subset=[1])
# Замещение пропущенные значения в признаках на медианные значения по классам
for feature in [4, 5]:
    for label in ['M', 'B']:
        median_value = data[data[1] == label][feature].median()
        data.loc[(data[1] == label) & (data[feature].isna()), feature] = median_value
```

3. Если количество различных меток класса больше пяти, то объедините некоторые (наименее многочисленные) классы, чтобы общее количество классов не превышало пять.

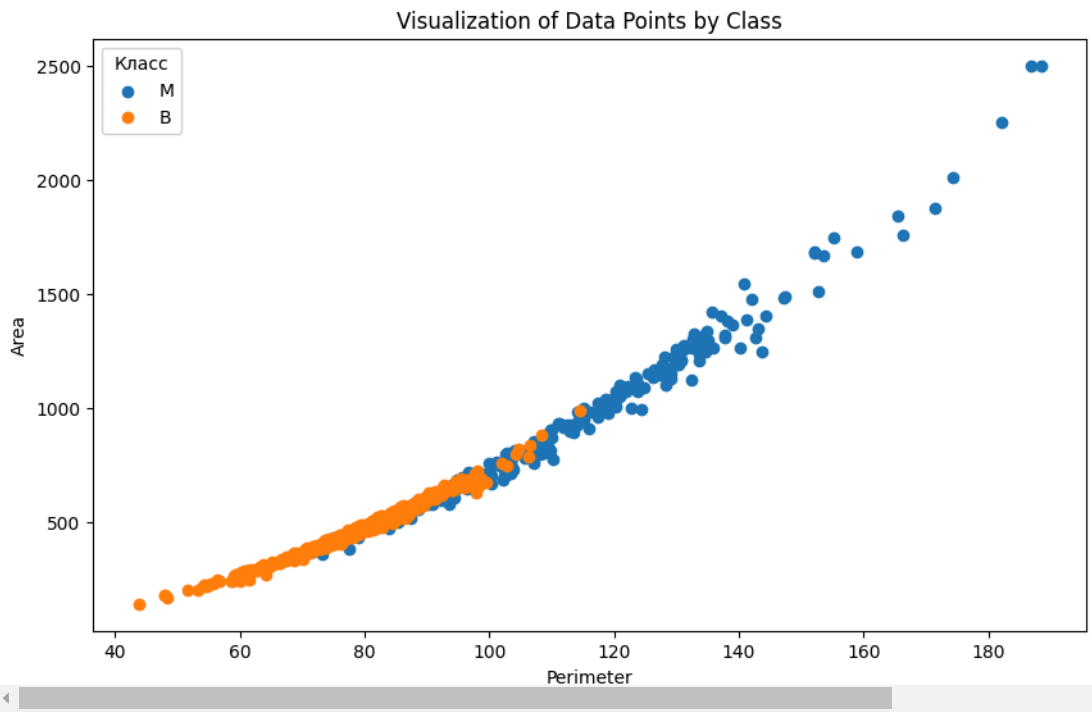
```
# Проверка количества различных меток класса
class_counts = data[1].value_counts()

# Если количество меток больше пяти, объединяем наименее многочисленные
while len(class_counts) > 5:
    # Находим наименее многочисленные классы
    least_frequent_classes = class_counts.nsmallest(len(class_counts) - 5).index
    # Объединяем наименее многочисленные классы
    data[1] = data[1].replace(least_frequent_classes, 'Other')
    # Пересчитываем количество меток
    class_counts = data[1].value_counts()

# Проверяем результат
print("Количество различных меток класса после объединения:")
print(data[1].value_counts())
```

```
Количество различных меток класса после объединения:
1
B    357
M    212
Name: count, dtype: int64
```

4. Визуализируйте набор данных в виде точек плоскости с координатами, соответствующими двум признакам, отображая точки различных классов разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

```
import matplotlib.pyplot as plt


# Создание визуализации данных
plt.figure(figsize=(10, 6))
# Уникальные значения классов
classes = data[1].unique()
# Отображение точек для каждого класса
for cls in classes:
    subset = data[data[1] == cls]
    plt.scatter(subset[4], subset[5], label=cls)
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('Visualization of Data Points by Class')
```

```
plt.legend(title='Класс')
plt.show()
```

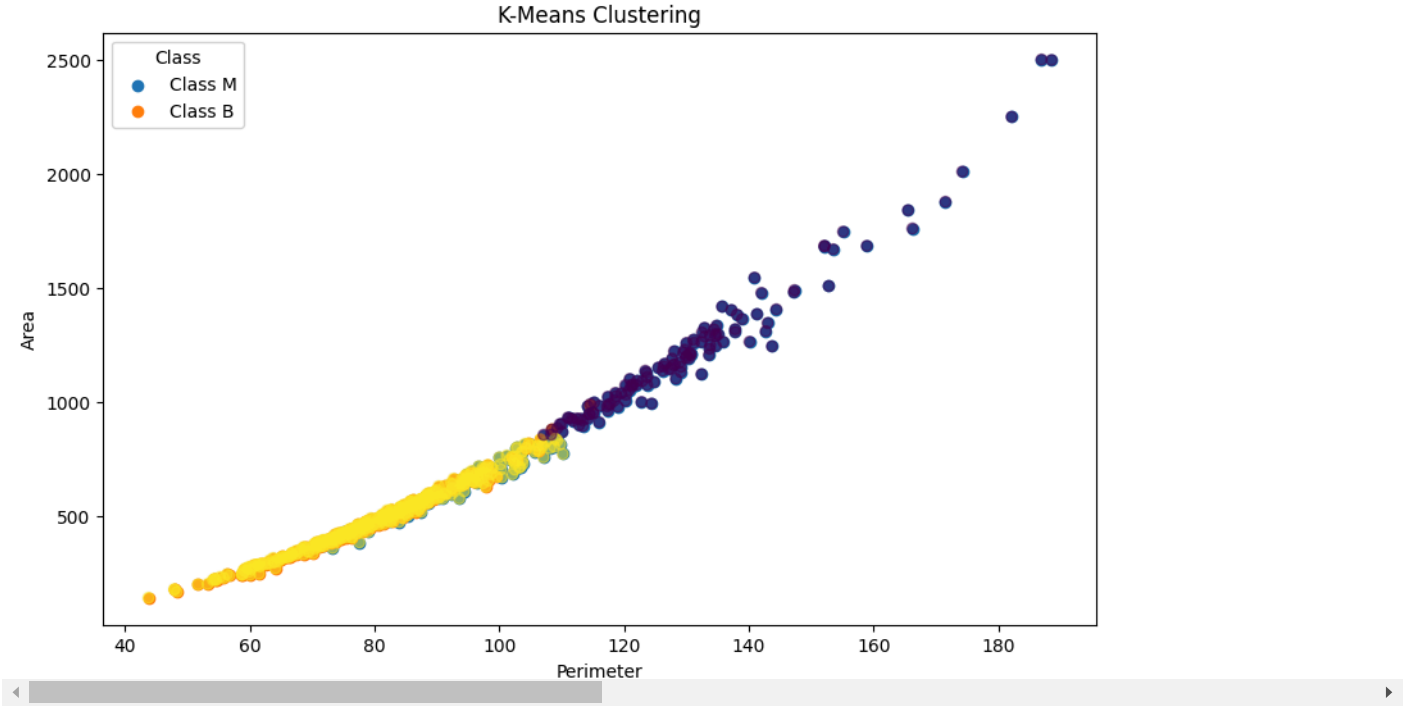


5. Проведите кластеризацию набора данных из двух признаков с помощью алгоритмов, указанных в индивидуальном задании, для случая, когда количество кластеров равно количеству классов в исходном наборе (с учетом корректировки). В случае отсутствия сходимости алгоритма измените аргументы по умолчанию или используйте для кластеризации случайную выборку из набора данных.

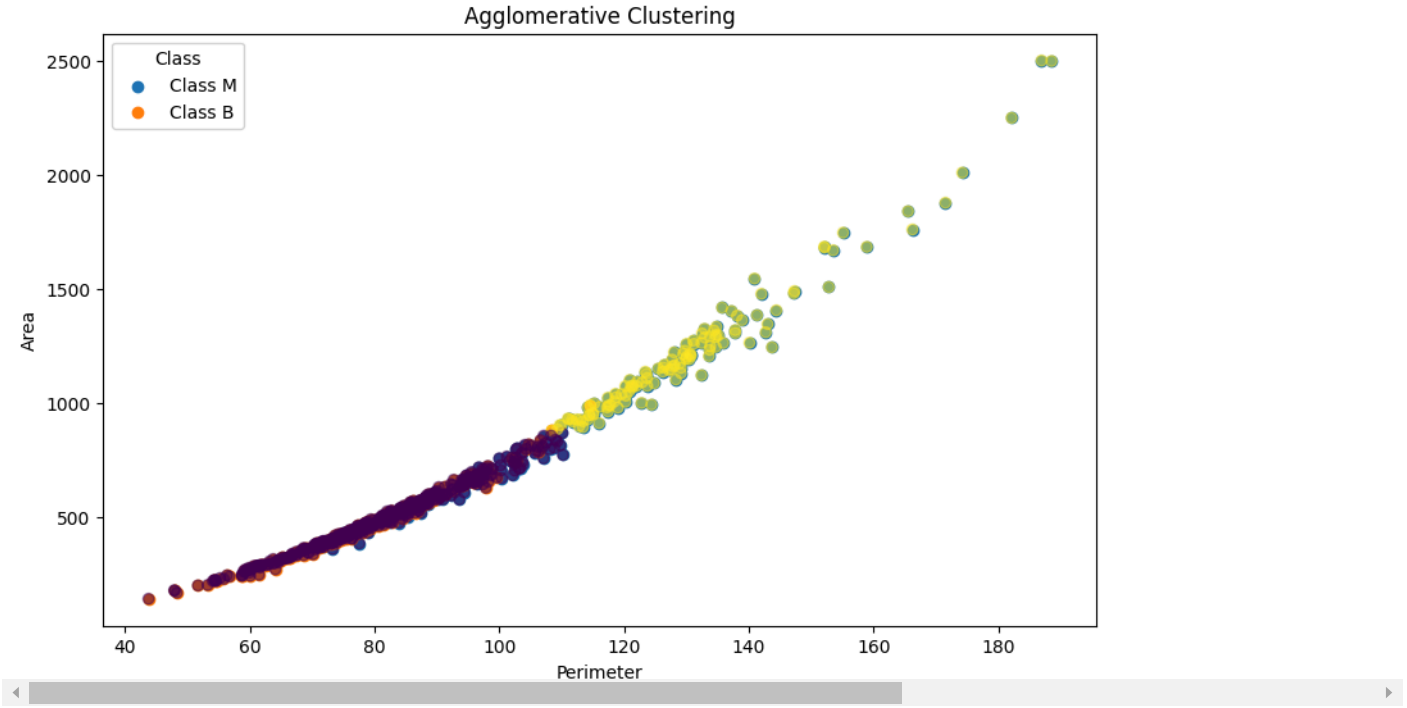
- Алгоритмы: K-means, Agglomerative Clustering, DBSCAN, Gaussian Mixture Model

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


X = data[[4, 5]].values
y_true = data[1].astype('category').cat.codes
n_classes = len(data[1].unique())
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
# K-Means
num_clusters = len(data[1].unique())
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data['KMeans_Cluster'] = kmeans.fit_predict(data[[4, 5]])
plt.figure(figsize=(10, 6))
for cls in data[1].unique():
    subset = data[data[1] == cls]
    plt.scatter(subset[4], subset[5], label=f'Class {cls}')
plt.scatter(data[4], data[5], c=data['KMeans_Cluster'], alpha=0.5, cmap='viridis')
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('K-Means Clustering')
plt.legend(title='Class')
plt.show()
```

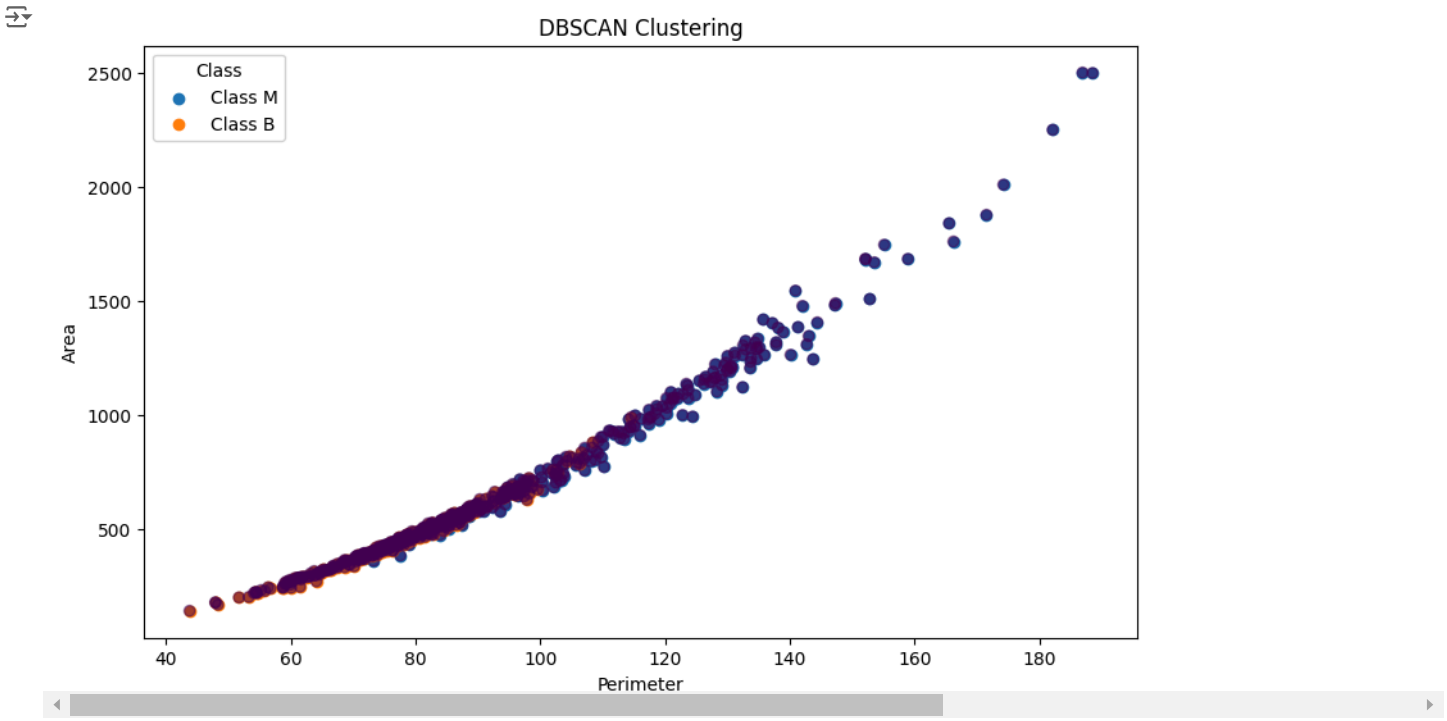
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to '
  super()._check_params_vs_input(X, default_n_init=10)
```



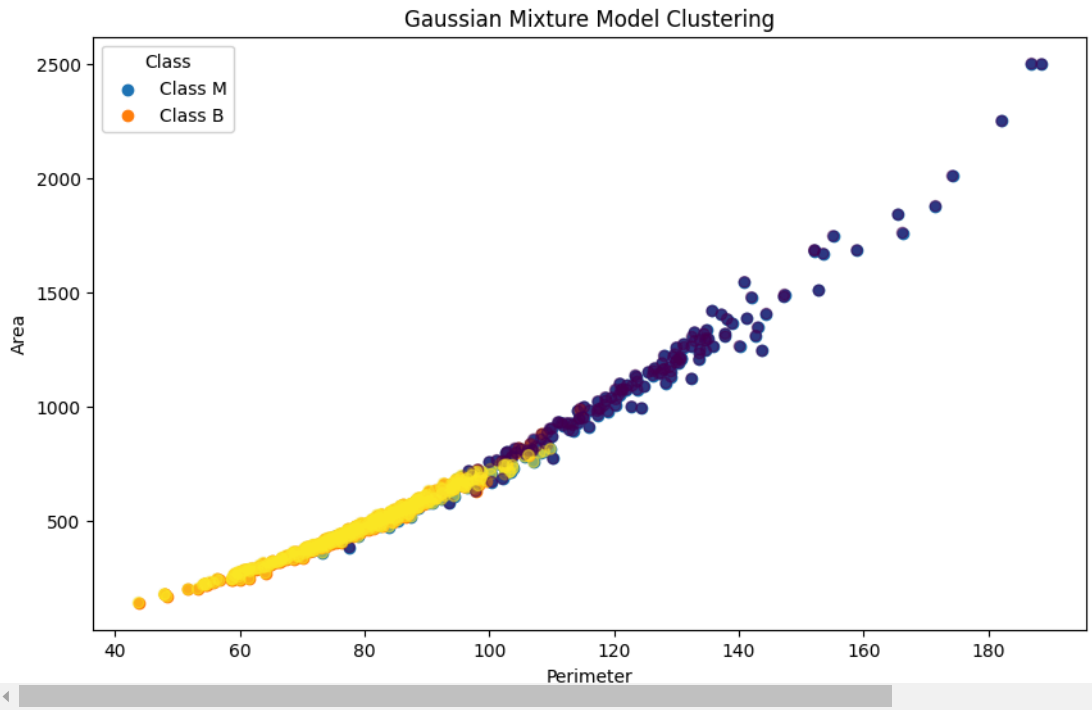

```
from sklearn.cluster import AgglomerativeClustering

# Agglomerative Clustering
agg_clustering = AgglomerativeClustering(n_clusters=num_clusters)
data['Agg_Cluster'] = agg_clustering.fit_predict(data[[4, 5]])
plt.figure(figsize=(10, 6))
for cls in data[1].unique():
    subset = data[data[1] == cls]
    plt.scatter(subset[4], subset[5], label=f'Class {cls}')
plt.scatter(data[4], data[5], c=data['Agg_Cluster'], alpha=0.5, cmap='viridis')
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('Agglomerative Clustering')
plt.legend(title='Class')
plt.show()
```



```
from sklearn.cluster import DBSCAN

# DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
data['DBSCAN_Cluster'] = dbscan.fit_predict(data[[4, 5]])
plt.figure(figsize=(10, 6))
for cls in data[1].unique():
    subset = data[data[1] == cls]
    plt.scatter(subset[4], subset[5], label=f'Class {cls}')
plt.scatter(data[4], data[5], c=data['DBSCAN_Cluster'], alpha=0.5, cmap='viridis')
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('DBSCAN Clustering')
plt.legend(title='Class')
plt.show()
```




```
from sklearn.mixture import GaussianMixture

# Gaussian Mixture Model
gmm = GaussianMixture(n_components=num_clusters, random_state=42)
data['GMM_Cluster'] = gmm.fit_predict(data[[4, 5]])
plt.figure(figsize=(10, 6))
for cls in data[1].unique():
    subset = data[data[1] == cls]
    plt.scatter(subset[4], subset[5], label=f'Class {cls}')
plt.scatter(data[4], data[5], c=data['GMM_Cluster'], alpha=0.5, cmap='viridis')
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('Gaussian Mixture Model Clustering')
plt.legend(title='Class')
plt.show()
```



6. Для каждого из алгоритмов кластеризации, указанных в индивидуальном задании, постройте матрицу сопряженности, используя функцию contingency_matrix() из scikit-learn, и найдите значения мер качества кластеризации, указанные в индивидуальном задании, на основании данных в матрице сопряженности, не используя другие функции из scikit-learn или других фреймворков.

- Меры качества: F-мера, парные меры TP, FN, FP, TN, индекс Жаккара

```
from sklearn.metrics.cluster import contingency_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
import numpy as np
import math


# Извлечение признаков и меток
X = data[[4, 5]].values  # Периметр и Площадь
y_true = data[1].astype('category').cat.codes  # Кодирование меток как целых чисел
n_classes = len(data[1].unique())

# Нормализация данных
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



# Функция для расчета TP, FN, FP, TN
def calculate_metrics(cont_matrix):
    # Общее количество пар
    n = np.sum(cont_matrix)
    tp = 0
    for i in range(cont_matrix.shape[0]):
        for j in range(cont_matrix.shape[1]):
            if cont_matrix[i, j] > 1:
                tp += math.comb(cont_matrix[i, j], 2)

    # Общее количество пар
    total_pairs = math.comb(n, 2)
    row_sums = np.sum(cont_matrix, axis=1)
    col_sums = np.sum(cont_matrix, axis=0)
    # Расчет FP и FN
    fn = np.sum([math.comb(r, 2) for r in row_sums]) - tp
    fp = np.sum([math.comb(c, 2) for c in col_sums]) - tp

    # TN = все пары - TP - FN - FP
    tn = total_pairs - tp - fn - fp
```

```
    return tp, fn, fp, tn


# Функция для расчета индекса Жаккара
def jaccard_index(cont_matrix):
    tp, fn, fp, _ = calculate_metrics(cont_matrix)
    return tp / (tp + fp + fn)


# Алгоритмы кластеризации
algorithms = {
    'K-means': KMeans(n_clusters=n_classes, random_state=42),
    'Agglomerative': AgglomerativeClustering(n_clusters=n_classes),
    'DBSCAN': DBSCAN(eps=0.5, min_samples=5),
    'Gaussian Mixture': GaussianMixture(n_components=n_classes, random_state=42)
}

# Применение каждого алгоритма и расчет мер качества
for name, algorithm in algorithms.items():
    print(f"\n{name} Кластеризация")

    # Обучение модели и прогнозирование меток кластеров
    if isinstance(algorithm, GaussianMixture):
        y_pred = algorithm.fit_predict(X_scaled)
    else:
        y_pred = algorithm.fit_predict(X_scaled)

    # Вычисление матрицы сопряженности
    cont_matrix = contingency_matrix(y_true, y_pred)
    print("Матрица сопряженности:")
    print(cont_matrix)

    # Расчет TP, FN, FP, TN
    tp, fn, fp, tn = calculate_metrics(cont_matrix)
    print(f"TP: {tp}, FN: {fn}, FP: {fp}, TN: {tn}")

    # Расчет индекса Жаккара
    ji = jaccard_index(cont_matrix)
    print(f"Индекс Жаккара: {ji:.4f}")
```

```
K-means Кластеризация
Матрица сопряженности:
[[354   3]
 [ 81 131]]
TP: 74239, FN: 11673, FP: 29067, TN: 46617
Индекс Жаккара: 0.6457

Agglomerative Кластеризация
Матрица сопряженности:
[[355   2]
 [ 92 120]]
TP: 74162, FN: 11750, FP: 32900, TN: 42784
Индекс Жаккара: 0.6242

DBSCAN Кластеризация
Матрица сопряженности:
[[  0 357]
 [  4 208]]
TP: 85080, FN: 832, FP: 74256, TN: 1428
Индекс Жаккара: 0.5312

Gaussian Mixture Кластеризация
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to '
  super()._check_params_vs_input(X, default_n_init=10)
Матрица сопряженности:
[[  8 349]
 [144  68]]
TP: 73328, FN: 12584, FP: 24884, TN: 50800
Индекс Жаккара: 0.6618
```

7. Определите алгоритм кластеризации, оптимальный с точки зрения меры качества кластеризации, указанной в индивидуальном задании.

```
def f_measure(tp, fn, fp):
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    return 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0
    # Calculate F-measure
```

```
        f_measure_score = f_measure(tp, fn, fp)
        results[name] = f_measure_score

# Determine the best algorithm
best_algorithm = max(results, key=results.get)
print(f"\nЛучший алгоритм кластеризации по F-мере: {best_algorithm} с F-мерой: {results[best_algorithm]:.4f}")
```

```
Лучший алгоритм кластеризации по F-мере: Gaussian Mixture с F-мерой: 0.7965
```

8. Для оптимального алгоритма кластеризации из предыдущего пункта визуализируйте области принятия решения и набор данных в виде точек на плоскости с координатами, соответствующими двум признакам, отображая точки различных кластеров разными цветами. Подпишите оси и рисунок, создайте легенду набора данных.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from matplotlib.colors import ListedColormap



# Обучение модели Gaussian Mixture
gmm = GaussianMixture(n_components=n_classes, random_state=42)
y_pred = gmm.fit_predict(X_scaled)

# Определение сетки для границ принятия решения
x_min, x_max = X_scaled[:, 0].min() - 1, X_scaled[:, 0].max() + 1  # Минимум и максимум по первой оси
y_min, y_max = X_scaled[:, 1].min() - 1, X_scaled[:, 1].max() + 1  # Минимум и максимум по второй оси
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),  # Создание сетки по x
                     np.arange(y_min, y_max, 0.01))  # Создание сетки по y

# Прогнозирование кластеров для каждой точки в сетке
Z = gmm.predict(np.c_[xx.ravel(), yy.ravel()])  # Прогнозирование кластеров
Z = Z.reshape(xx.shape)  # Приведение результата к форме сетки

# Визуализация границ принятия решения
plt.figure(figsize=(10, 6))
cmap_background = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])  # Цветовая карта для фона
cmap_points = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])  # Цветовая карта для точек
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap_background)  # Отображение границ кластеров
scatter = plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_pred, cmap=cmap_points, edgecolor='k')
plt.xlabel('Perimeter')
plt.ylabel('Area')
plt.title('Границы принятия решения и кластеризация с использованием модели Gaussian Mixture')
plt.legend(handles=scatter.legend_elements()[0], labels=[f'Кластер {i}' for i in range(n_classes)], title='Кластеры')

plt.show()
```