**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**
**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**Факультет прикладной математики и информатики**
**Кафедра математического моделирования и анализа данных**

# Дискриминантный анализ неоднородных данных

Отчет по лабораторной работе №4
студентки 3 курса 7 группы
**Летецкой М.С.**

**Преподаватель**
Малюгин В.И.

Минск, 2024

УСЛОВИЕ ЗАДАНИЯ

использовать выборку из смеси распределений (1, 3) для всех переменных;

для обучения и экзамена использовать выборки из различных классов в пропорциях 80% и 20 %, либо процедуру кросс-валидации в тех же пропорциях; использовать алгоритмы: ЛДА, КДА, деревья решений CART.

ЛИНЕЙНЫЙ ДИСКРИМИНАНТНЫЙ АНАЛИЗ

Линейный дискриминантный анализ (LDA) - это метод классификации, используемый в машинном обучении, статистике и анализе данных. Основная цель LDA - найти линейную комбинацию признаков (переменных), которая наилучшим образом разделяет два или более классов.

Основные идеи LDA:

1. Предпосылка: LDA предполагает, что данные в каждом классе имеют нормальное распределение с одинаковой ковариационной матрицей.
2. Поиск проекций: LDA пытается найти такие линейные проекции (направления), на которые классы проецируются наиболее “компактно”, т.е. с минимальным разбросом внутри классов и максимальным расстоянием между центроидами (средними значениями) разных классов.
3. Разделяющие гиперплоскости: В результате LDA строит разделяющие гиперплоскости, которые максимально разделяют классы. В двумерном случае это будут линии, в трехмерном — плоскости, и т.д.

В предыдущих лабораторных работах было выявлено, что параметры, с которыми необходимо работать, имеют нормальное распределение.

Первый этап построения модели – разделение данных на тренировочные и тестовые в пропорции, заданной в условии:

```
X = df_data.iloc[:, 0:4].values
y = df_data.iloc[:, 4].values

sc = StandardScaler()
X = sc.fit_transform(X)
le = LabelEncoder()
y = le.fit_transform(y)
X_train, X_test,y_train, y_test = train_test_split(X, y,test_size=0.2)
```

После построения и обучения модели была найдена точность её предсказаний:

```
y_pred = classifier.predict(X_test)

print('Точность : ' + str(accuracy_score(y_test, y_pred)))
```

```
Точность : 1.0
```

Для наглядности была выведена матрица ошибок. Она показывает, какую часть классификатор определил неверно. На главной диагонали количество верных определений, на остальных – ложные.

```
conf_m = confusion_matrix(y_test, y_pred)
print(conf_m)
```

```
[[11  0]
 [ 0  9]]
```

Классификационный отчет:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00         9

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Вывод: построенная модель с точностью 100% определяет тип цветка и относит к нужному классу.

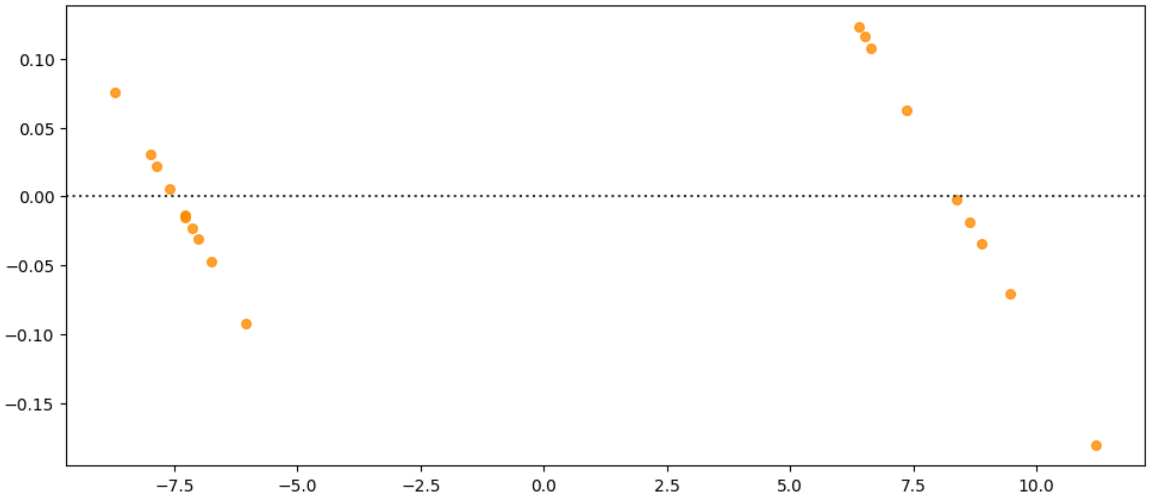
## КВАДРАТИЧНЫЙ ДИСКРИМИНАНТНЫЙ АНАЛИЗ

Квадратичный дискриминантный анализ (QDA) — это метод классификации, тесно связанный с линейным дискриминантным анализом (LDA), но с ключевым отличием: QDA предполагает, что данные в каждом классе имеют нормальное распределение, но не обязательно с одинаковыми ковариационными матрицами.

Основные идеи QDA:

1. Нормальное распределение, разные ковариационные матрицы: В отличие от LDA, QDA позволяет каждому классу иметь свою собственную ковариационную матрицу. Это делает QDA более гибким, позволяя моделировать более сложные распределения данных.
2. Квадратичные разделяющие поверхности: поскольку каждый класс может иметь свою ковариационную матрицу, разделяющие поверхности между классами в QDA являются квадратичными, а не линейными, как в LDA. Это позволяет QDA лучше моделировать нелинейные зависимости между признаками.
3. Увеличенная сложность модели: из-за дополнительной гибкости, QDA является более сложной моделью, чем LDA. Это означает, что для обучения QDA требуется больше данных, и существует больший риск переобучения (overfitting), особенно при небольшом количестве данных.

Так как данные уже разделены на тестовые и тренировочные, создаём модель и начинаем её обучение. После обучения проверяем работоспособность модели. Был построен график, на котором показано, как модель выполнила градацию данных по классам:




Точность модели:

```
print(accuracy_score(y_test, y_pred_qda))
```

```
1.0
```

Матрица ошибок:

```
print(confusion_matrix(y_test, y_pred_qda))
```

```
[[11  0]
 [ 0  9]]
```

Отчет:

```
print(classification_report(y_test, y_pred_qda))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00         9

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Вывод: модель построена корректно и с точностью 100% разбивает данные на классы.

CART-ДЕРЕВЬЯ

CART - это алгоритм обучения с учителем, который используется как для задач классификации, так и для задач регрессии. Суть CART заключается в создании древовидной структуры, в которой каждое внутреннее узловое решение (на основе признака) разделяет данные на две или более подгруппы.

Основные идеи CART:

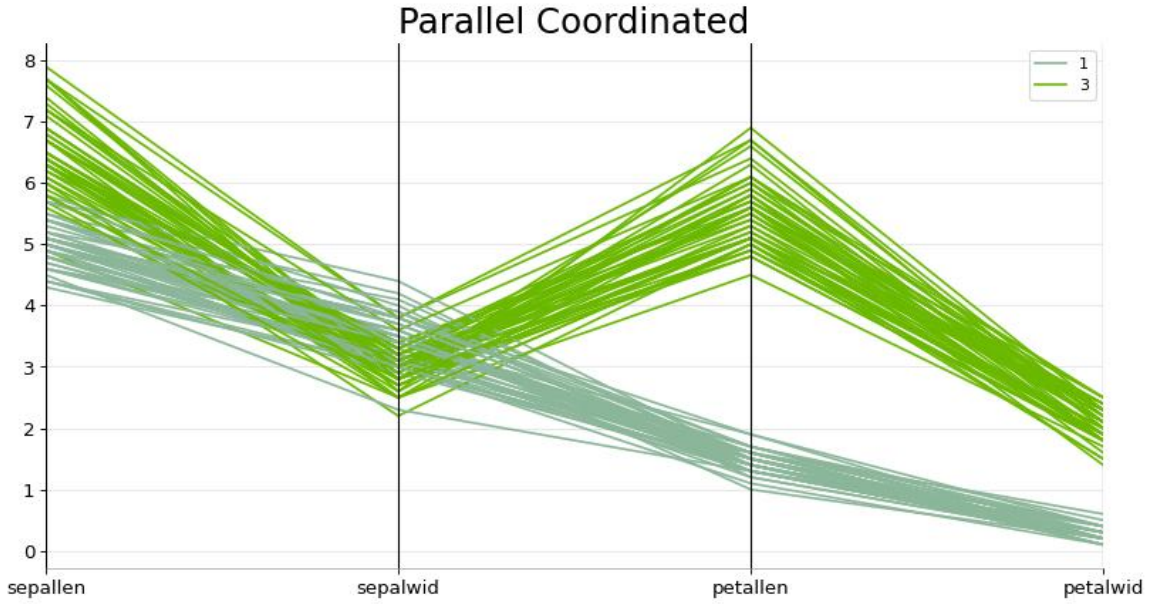
1. Древовидная структура: CART представляет собой бинарное дерево. Это означает, что каждый внутренний узел имеет ровно двух потомков (дочерних узлов).

2. Рекурсивное разделение: Алгоритм рекурсивно разделяет данные на основе значений признаков. Разделение выбирается таким образом, чтобы максимизировать однородность классов (в задаче классификации) или минимизировать дисперсию (в задаче регрессии) в дочерних узлах.
3. Простые правила разделения: В каждом узле используется простое правило разделения.
4. Листовые узлы: Процесс разделения продолжается до тех пор, пока не будут достигнуты “листовые узлы” (или конечные узлы). Каждый листовой узел содержит прогноз (класс для классификации или числовое значение для регрессии).

Процесс построения дерева CART:

1. Начальный узел: все обучающие данные находятся в корневом узле.
2. Поиск лучшего разделения: для каждого узла алгоритм перебирает все возможные значения признаков, стремясь найти разделение, которое максимизирует однородность классов (для классификации) или минимизирует дисперсию (для регрессии).
3. Критерии разделения (для классификации): например, индекс Джини.
4. Разделение узла: после того, как лучшее разделение найдено, узел разделяется на два дочерних узла.
5. Рекурсивное применение: процесс разделения рекурсивно применяется к каждому из дочерних узлов до тех пор, пока не будет достигнут критерий остановки (например, минимальное количество наблюдений в узле, максимальная глубина дерева и т.д.).
6. Присвоение прогноза: каждый листовой узел получает прогноз (например, самый часто встречаемый класс для классификации или среднее значение целевой переменной для регрессии).

Для наглядности был построен график параллельных координат:




По нему видно, что данные лучше всего разделяются по классам при помощи параметра ‘petallen’.

Было построено дерево, в качестве критерия разделения был выбран индекс Джини:

```
clf = DecisionTreeClassifier(criterion='gini', random_state=42)
clf.fit(X_train, y_train)
```

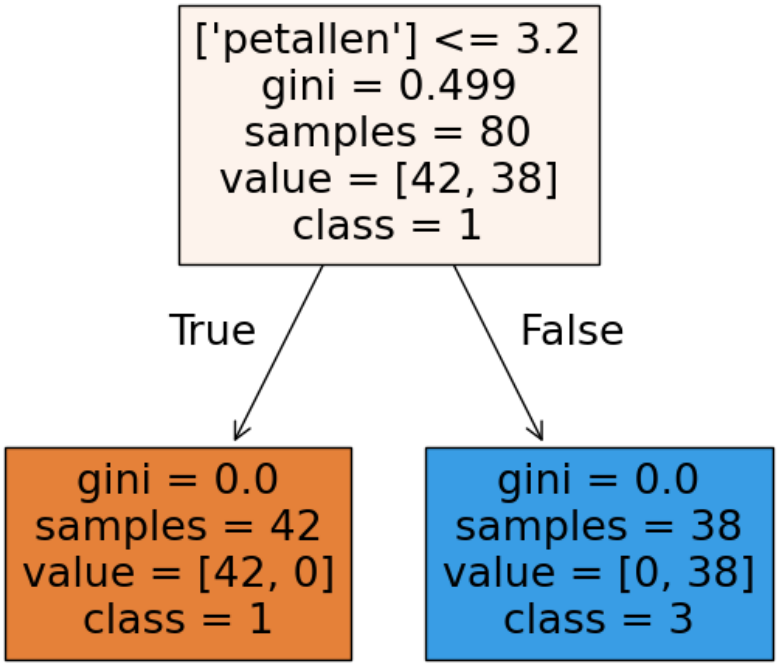
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

Индекс Джини измеряет вероятность того, что две случайные выборки из набора данных (узла дерева) будут принадлежать к разным классам. Таким образом, он показывает, насколько “нечистым” или “неоднородным” является данный набор данных с точки зрения принадлежности к разным классам.

Чем ниже индекс Джини, тем более однородными являются классы в наборе данных (или узле).

При обучении деревьев CART, индекс Джини используется для выбора наилучшего разделения на каждом шаге. Алгоритм выбирает то разделение (признак и пороговое значение), которое приводит к наименьшему значению индекса Джини (то есть к наиболее однородным дочерним узлам). Индекс Джини выступает в качестве критерия качества разделения. Чем ниже индекс Джини дочерних узлов (после разделения), тем лучше разделение.

Визуализация CART-дерева:




Оценка точности, классификационный отчет и матрица ошибок:

```
y_pred = clf.predict(X_test)
print("оценка точности:", accuracy_score(y_test, y_pred))
print("Результаты классификации:")
print(classification_report(y_test, y_pred, target_names=['1','3']))
print(confusion_matrix(y_pred, y_test))
```

```
оценка точности: 1.0
Результаты классификации:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         8
           3       1.00      1.00      1.00        12

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

[[ 8  0]
 [ 0 12]]
```

точность(precision) = 1

Данная модель с точностью 100% распределила данные по классам.

ВЫВОДЫ

Все три построенные модели с точностью 100% отнесли данные к верным классам из предоставленного для практики dataset.

Можно сделать пару общих теоретических выводов:

1. Гибкость модели:
   - LDA: самая негибкая. Она предполагает, что разделение между классами линейное. Это хорошо работает на простых данных, но плохо на сложных.
   - QDA: более гибкая, чем LDA, так как может моделировать квадратичные поверхности разделения, но все еще имеет ограничения.
   - CART: самая гибкая из трех. Древовидная структура позволяет моделировать сложные нелинейные зависимости.
2. Предположения:
   - LDA и QDA: оба предполагают, что данные распределены нормально. LDA, кроме того, предполагает, что ковариационные матрицы всех классов одинаковы.
   - CART: не делает предположений о распределении данных.
3. Переобучение:
   - LDA: наименее склонен к переобучению, особенно если данных мало.
   - QDA: более склонен к переобучению, чем LDA, особенно если данных мало и признаков много.
   - CART: может легко переобучиться, особенно если дерево слишком глубокое и не ограничено.
4. Чувствительность к выбросам:
   - LDA и QDA: относительно чувствительны к выбросам.

- CART: менее чувствителен, так как выбросы часто изолируются в отдельных узлах.

LDA подходит, когда классы линейно разделимы, данных мало. QDA подходит, когда классы нелинейно разделимы, ковариационные матрицы разные, и данных достаточно. CART подходит, когда есть нелинейные зависимости, необходимо интерпретировать модель, нужно обрабатывать разные типы данных, и можно контролировать переобучение.