

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизация производственных процессов»

Основы разработки СППР на языке Python

Методические указания для лабораторных работ  
для студентов заочной формы обучения

Ростов-на-Дону  
ДГТУ  
2023

УДК 681.5

Составитель: Быкадор В.С.

Методические указания. – Ростов-на-Дону : Донской гос. техн. ун-т, 2023. – 17 с.

Методические указания по дисциплине «Основы разработки СППР на языке Python» предназначены для студентов заочной формы обучения по направлению подготовки 15.04.04 «Автоматизация технологических процессов и производств» и представляют задания и порядок выполнения лабораторных работ, позволяющих студентам получить и закрепить требуемые знания по изучаемой дисциплине.

УДК 681.5

Печатается по решению редакционно-издательского совета  
Донского государственного технического университета

---

В печать \_\_\_\_\_.\_\_\_\_.20\_\_ г.  
Формат 60x84/16. Объем \_\_\_\_\_ усл. п. л.  
Тираж \_\_\_\_\_ экз. Заказ № \_\_\_\_\_.

---

Издательский центр ДГТУ  
Адрес университета и полиграфического предприятия:  
344000, г. Ростов-на-Дону, пл. Гагарина, 1

© Донской государственный  
технический университет, 2023

## Содержание

Лабораторная работа № 1 .....	4
Классификация методом $K$ ближайших соседей .....	4
Лабораторная работа № 2 .....	7
Линейный мультиклассовый классификатор .....	7
Практическая работа № 3 .....	11
Классификация на основе SVM с гауссовским ядром .....	11
Практическая работа № 4 .....	14
Классификация на основе решающего дерева .....	14
Перечень использованных информационных ресурсов .....	17

# Лабораторная работа № 1

## Классификация методом $K$ ближайших соседей

Цель — программная реализация и исследование свойств метода  $K$  ближайших соседей.

### Задание

Требуется разработать модель машинного обучения с использованием метода  $K$  ближайших соседей для СППР о сорте растения «ирис» с целью дальнейшей классификации ириса по измеренным признакам, которыми являются:

- длина чашелистиков (sepal length), см;
- ширина чашелистиков (sepal width), см;
- длина лепестков (petal length), см;
- ширина лепестков (petal width), см;

Рассматривается три класса (сорта) ириса:

- щетиный (setosa);
- разноцветный (versicolor);
- виргинский (virginica).

В массиве имеется 150 маркированных (заранее размеченных данных).

1) Подключите набор данных `sklearn.datasets.load_iris` к файлу программы через инструкцию `from...import`.

2) Загрузите набор данных `iris_dataset = load_iris()`.

Исследуйте данные, которые имеются в загруженном наборе данных, выполнив код

```
print(iris_ds.keys())
```

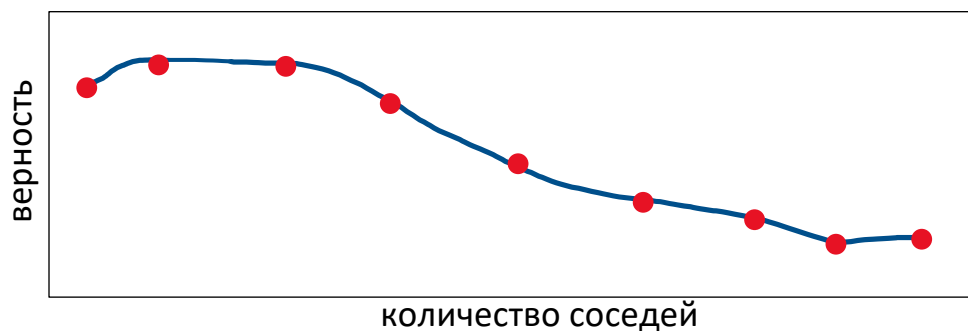
Вы получите список ключей словаря. Далее исследуйте какие данные имеются по ключам словаря, например, можно выполнить код

```
print(iris_ds['target_names'])
```

- 3) Используя функцию `train_test_split()` требуется разделить маркированные данные на два списка: обучающий набор (75% записей) и тестовый (контрольный) набор (25% записей).
- 4) Для обучения модели, реализующий алгоритм  $K$  ближайших соседей, необходимо использовать метод `fit()` экземпляра класса **KNeighborsClassifier**.
- 5) Исследуйте правильность полученной модели машинного обучения для различного количества голосующих соседей, используя метод `score()` экземпляра класса **KNeighborsClassifier**. Рекомендуется использовать следующие количества голосующих соседей:

1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110

Постройте график зависимости верности модели машинного обучения от количества принимающих участие в голосовании соседей



- 6) Заново обучите модель, но уже на фиксированном количестве голосующих соседей, дающих хорошую верность модели машинного обучения.
- 7) Проверьте работу модели машинного обучения на каких-либо произвольных значениях признаков, но взятых в пределах наблюдаемых значений (см. табл. № 1.1)

Таблица № 1.1. Статистические значений признаков

Признак, см	Min	Max	Среднее
sepal length, sl (длина чашелистика)	4.3	7.9	5.84
sepal width, sw (ширина чашелистика)	2.0	4.4	3.05
petal length, pl (длина лепестка)	1.0	6.9	3.76
petal width, pw (ширина лепестка)	0.1	2.5	1.20

Достаточно получить два различных результата, при этом соблюдайте формат входных данных sl, sw, pl, pw:

```
new_sample = [[6.5, 3.9, 5.3, 2.2]]
```

Для выполнения классификации наблюдаемого объекта используйте метод **predict()** экземпляра класса **KNeighborsClassifier**

```
>> Образец относится к классу ирисов: "Виргинский"
```

- 8) Доработайте программу до рабочего консольного приложения с предложением пользователю ввести значения признаков, на основании которых пользователь получит ответ о сорте ириса.

При этом следует учесть, что при запуске программы обучается модель, а затем выполняется цикл взаимодействия с пользователем – не нужно модель обучать при каждом вводе пользователем новых данных.

Отчет по лабораторной работе № 01.

- 1) титульный лист;
- 2) цели лабораторной работы;
- 3) листинг законченной программы;
- 4) график зависимости верности модели машинного обучения от количества принимающих участие в голосовании соседей;
- 5) консольные результаты работы модели машинного обучения на каких-либо произвольных значениях признаков, но взятых в пределах наблюдаемых значений;
- 6) выводы.

## Лабораторная работа № 2

### Линейный мультиклассовый классификатор

Цель — программная реализация и исследование свойств линейных моделей машинного обучения для мультиклассовой классификации на основе метода опорных векторов.

#### Задание

Требуется разработать линейный мультиклассовый классификатор для разделения объектов на три класса используя линейную модель, основанную на методе опорных векторов. Для выполнения данной практической работы потребуются следующие библиотеки:

- **numpy**;
- **matplotlib**;
- **sklearn**;
- **pandas**.

Линейная модель, основанная на методе опорных векторов, выполняет обучение по заранее размеченному набору данных, то есть это машинное обучение с учителем.

- 1) Загрузите данные в формате CSV в объект **pandas.DataFrame**, используя метод **pandas.read\_csv()** из файла **pr\_linear\_data.csv** (файл можно взять у преподавателя).

Формат файла имеет следующий вид (разделитель данных — символ `'\t'`):

X1	X2	y
-7.7	-8.4	2
5.5	0.7	1
-3.0	9.6	0
6.0	0.6	1
-6.5	-6.3	2
3.6	1.4	1
-2.2	10.0	0
.....		

То есть в файле содержится следующая информация:

- значения признаков объекта  $X_1$  и  $X_2$ , (которые также являются координатами двумерного пространства. Для возможности визуального контроля работы линейного классификатора рассматривается двумерный случай);
- метка класса  $y \in \{0, 1, 2\}$ .

На рисунке 2.1 показан график считанных данных из файла **pr\_linear\_data.csv**.

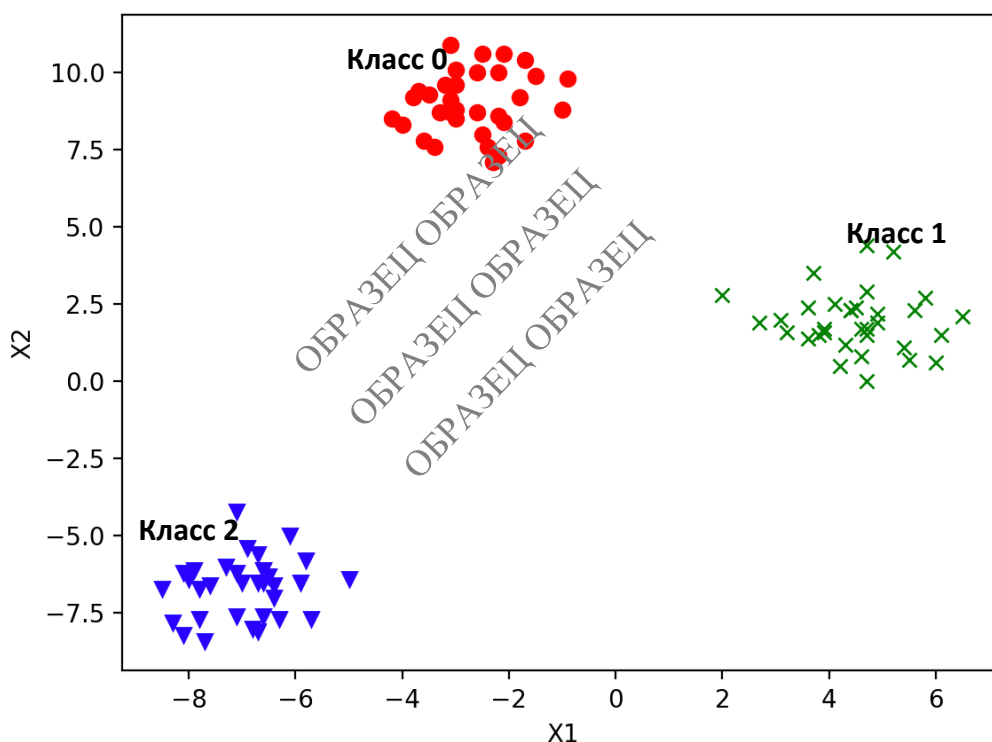


Рис. 2.1 – График считанных данных из файла **pr\_linear\_data.csv**

- 2) Разделите полученный **pandas.DataFrame** на массив признаков  $X$  и массив меток классов  $y$ .
- 3) Выполните обучение объекта **sklearn.svm.LinearSVC** - линейная модель, основанная методе опорных векторов.
- 4) Отобразите решающие границы (рис. 2.2), разделяющие объекты на три класса.

Для этого необходимо получить:

- массив коэффициентов  $w = [w_0 \ w_1 \ w_2]$  (этот массив хранится в свойстве **coef\_** объекта **sklearn.svm.LinearSVC**);

Массив  $w$  имеет следующий формат

	$w_i[0]$	$w_i[1]$
Класс 0	0,12	-1,17
Класс 1	-2,32	-0,34
Класс 2	...	...

- список коэффициентов смещений для решающих границ каждого класса  $\vartheta = [\vartheta_0 \ \vartheta_1 \ \vartheta_2]$  (свойство **intercept\_** объекта **sklearn.svm.LinearSVC**).

Решающая граница вычисляется по следующей формуле:

$$X_{2i} = \frac{-w_i[0] \cdot X_{1new} + \vartheta_i}{w_i[1]} \quad (2.1)$$

где  $X_{2i}$  –  $i$ -ая решающая граница;  
 $w_i[j]$  –  $j$ -ый ( $j = 0, 1$ ) весовой коэффициент для  $i$ -ой решающей границы;  
 $\vartheta_i$  – коэффициент смещения  $i$ -ой решающей границы.

На рисунке 2.2 показан пример отображения решающих границ для каждого из классов объектов

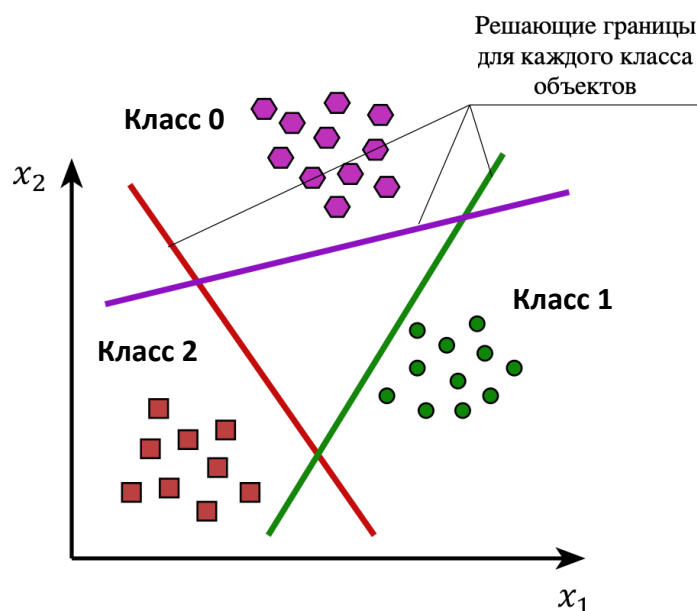


Рис. 2.2 – Пример отображения решающих границ для каждого из классов объектов

- 5) Задайте три тестовых объекта для каждой из размеченных областей (это может быть один объект у которого будут меняться значения признаков)

и загрузите его в линейный классификатор (метод `predict()` экземпляра класса `sklearn.svm.LinearSVC`). Выведите полученный от линейного классификатора класс в консоль и проверьте результат по графику (необходимо тестовый объект вывести на график).

### Отчет по лабораторной работе № 02.

- 1) титульный лист;
- 2) цели лабораторной работы;
- 3) листинг законченной программы;
- 4) графики с расположенными объектами (которые были считаны из файла), *решающими границами* (если возможно) и тестовыми объектами для каждого отдельного случая тестирования (см. рис. 2.3);
- 5) консольные результаты работы модели машинного обучения (рис. 2.3);
- 6) выводы.

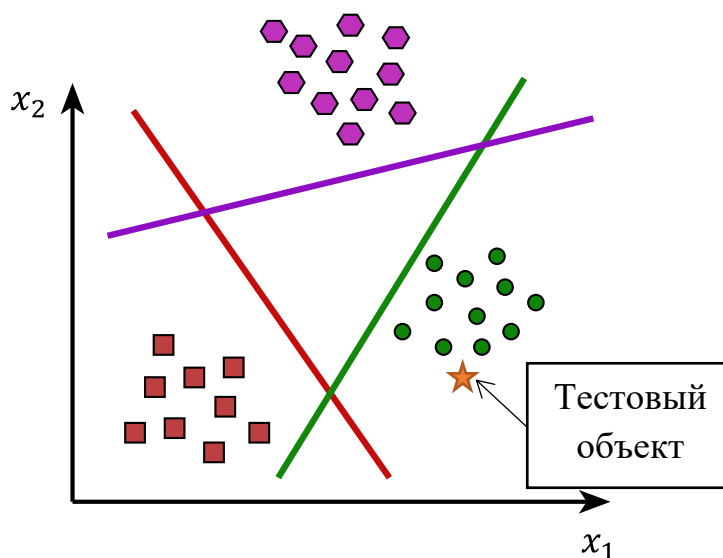


Рис. 2.3 – Примерный вид результатов работы программы машинного обучения классификации объектов

## Практическая работа № 3

### Классификация на основе SVM с гауссовским ядром

Цель — программная реализация и исследование свойств модели классификации объектов методом опорных векторов с гауссовским ядром.

#### Задание

Требуется разработать классификатор для разделения объектов на классы используя модель машинного обучения на основе метода опорных векторов с гауссовским ядром. Для выполнения данной практической работы потребуются следующие библиотеки:

- **numpy**;
- **matplotlib**;
- **sklearn**;
- **pandas**.

Модель, основанная на методе опорных векторов с ядрами, выполняет обучение по заранее размеченному набору данных, то есть это машинное обучение с учителем.

1) Загрузите данные в формате CSV в объект **pandas.DataFrame**, используя метод **pandas.read\_csv()** из файла **pr\_svm\_data\_2.csv** (файл можно взять у преподавателя).

Формат файла имеет следующий вид (разделитель данных – символ `'\t'`):

X1	X2	y
0.2	-3.8	1
0.0	-4.0	1
1.3	0.8	0
0.6	2.6	0
.....		

То есть в файле содержится следующая информация:

- значения признаков объекта  $X_1$  и  $X_2$ , (которые также являются координатами двумерного пространства. Для возможности визуального контроля работы ядерного классификатора рассматривается двумерный случай);
- метка класса  $y \in \{0, 1\}$ .

На рисунке 3.1 показан график считанных данных из файла **pr\_svm\_data\_2.csv**.

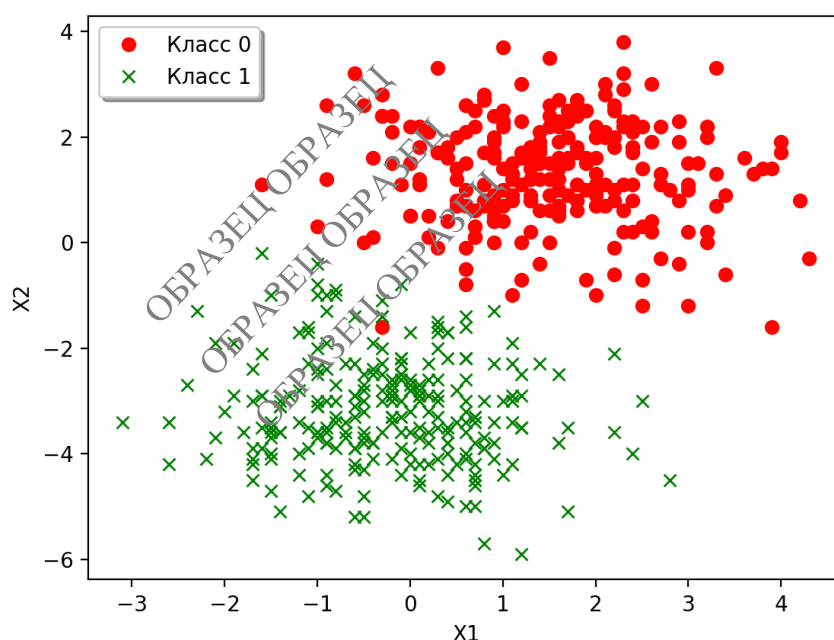


Рис. 3.1 – График считанных данных из файла **pr\_svm\_data\_2.csv**

- 2) Разделите полученный **pandas.DataFrame** на массив признаков  $X$  и массив меток классов  $y$ .
- 3) Выполните обучение объекта **sklearn.svm.SVC** - модель, основанная на методе опорных векторов с параметром **kernel='rbf'** ('rbf'-гауссовское ядро).
- 4) Выполните проверку работу машинной модели на трёх тестовых данных:
  - a.  $X_I = \{-0,4 \quad 1,65\}$ ;
  - b.  $X_{II} = \{0,6 \quad 2,45\}$ ;
  - c.  $X_{III} = \{-0,32 \quad -1,6\}$ .

Выведите прогноз, данный машинным классификатором в консоль и графики, подтверждающие правильную классификацию (см. рисунок 3.2).

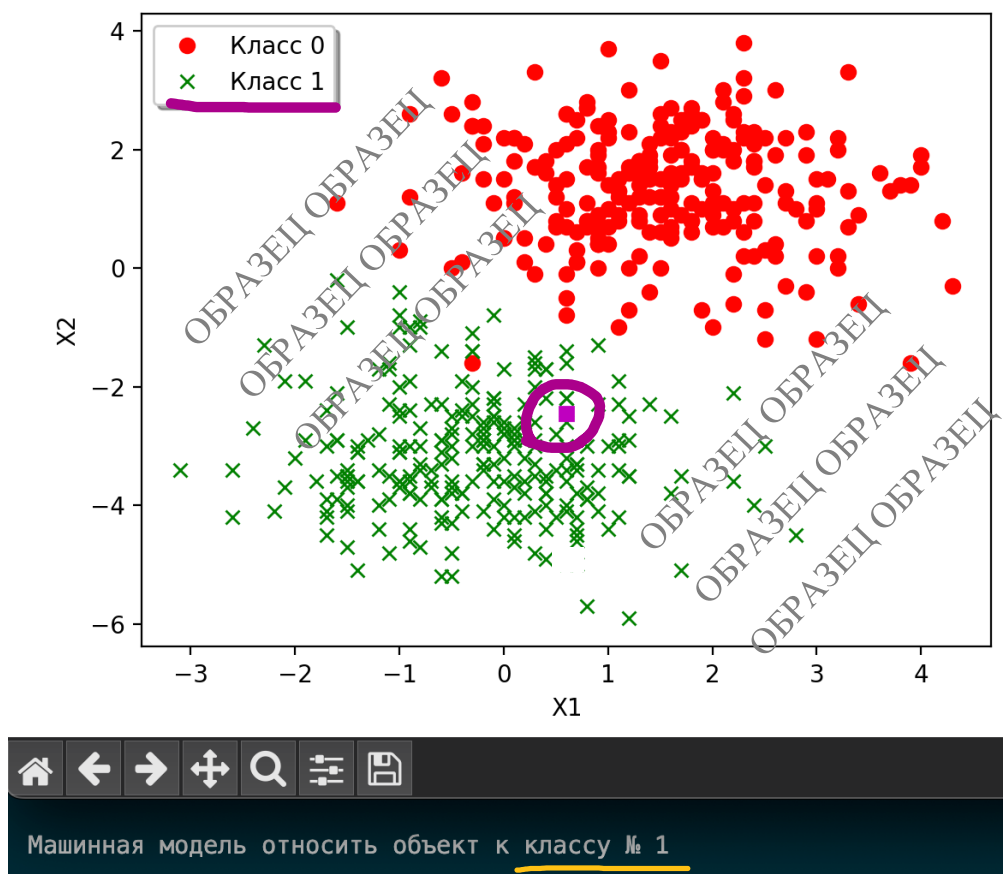


Рис. 3.2 – Результат работы машинного классификатора на данных из файла **pr\_svm\_data\_2.csv**

Отчет по лабораторной работе № 03.

- 1) титульный лист;
- 2) цели лабораторной работы;
- 3) листинг законченной программы;
- 4) графики с расположенными обучающими объектами и тестируемыми объектами  $X_I$ ,  $X_{II}$  и  $X_{III}$  с соответствующим выводом результата классификации в консоль (для примера см. рис. 3.2);
- 5) выводы.

## Практическая работа № 4

### Классификация на основе решающего дерева

Цель — программная реализация и исследование свойств модели классификации объектов методом дерева решения.

#### Задание

Требуется разработать классификатор для разделения объектов на классы используя модель машинного обучения на основе метода решающего дерева. Для выполнения данной практической работы потребуются следующие библиотеки:

- **matplotlib**;
- **sklearn**.

Модель, основанная на методе решающего дерева, выполняет обучение по заранее размеченному набору данных, то есть это машинное обучение с учителем.

- 1) Подключите набор данных **sklearn.datasets.load\_iris** к файлу программы через инструкцию **from...import**.
- 2) Загрузите набор данных **iris\_dataset = load\_iris()**. – данные о размерах лепестка, чашелистика и о сортах ириса.
- 3) Для обучения решающего дерева классификации используйте класс **sklearn.tree.DecisionTreeClassifier**.
- 4) Определите глубину решающего дерева, которая должна удовлетворят следующим требованиям:
  - глубина дерева должна быть как можно меньшей;
  - верность модели машинного обучения на тестовом наборе данных должна быть наибольшей.

Определить глубину решающего дерева лучше построив график зависимости верности модели машинного обучения решающего дерева от его глубины на тестовом наборе данных (см. рисунок 4.1), для того чтобы разделение размеченных данных на обучающие и тестовые было одним и тем же, для различных запусков программы, необходимо при инициализации

экземпляра класса **DecisionTreeClassifier** класса задать параметр **random\_state** равный какому-либо фиксированному числу.

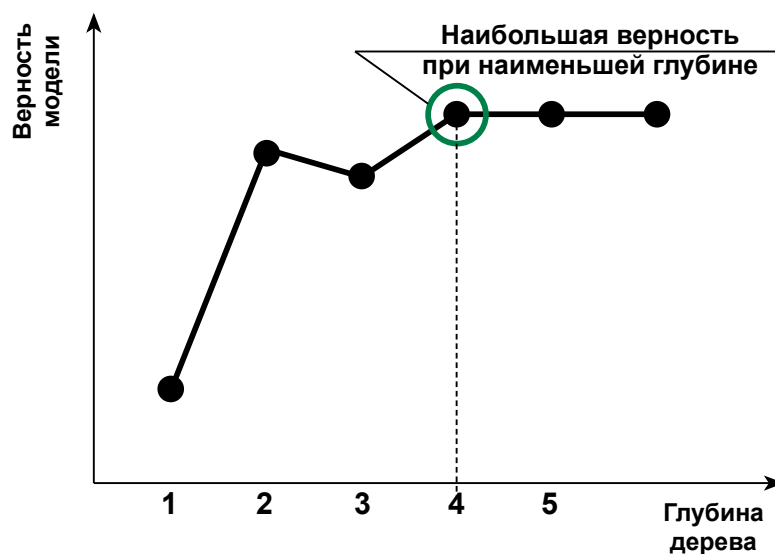


Рис. 4.1 – Зависимость верности машинной модели решающего дерева от его глубины

5) Проверьте работу модели машинного обучения на каких-либо произвольных значениях признаков, но взятых в пределах наблюдаемых значений (см. табл. № 1.1, практическая работа № 1).

Достаточно получить два различных результата, при этом соблюдайте формат входных данных **sl, sw, pl, pw**:

```
new_sample = [[6.5, 3.9, 5.3, 2.2]]
```

Для выполнения классификации наблюдаемого объекта используйте метод **predict()** экземпляра класса **DecisionTreeClassifier**

```
>> Образец относится к классу ирисов: "Виргинский"
```

6) Исследуйте работу метода **sklearn.tree.plot\_tree()** (потребуется подключить библиотеку **matplotlib**, так как метод **sklearn.tree.plot\_tree()** пользуется средствами библиотеки **matplotlib**). Метод **plot\_tree()** позволяет визуализировать дерево решений, полученное классом **DecisionTreeClassifier**. Метод **plot\_tree()** сохраняет рисунок решающего дерева в формате \*.png.

Выполните визуализацию дерева для глубин дерева: 2, 3, 4, 6 (см. пример на рисунке 4.2).

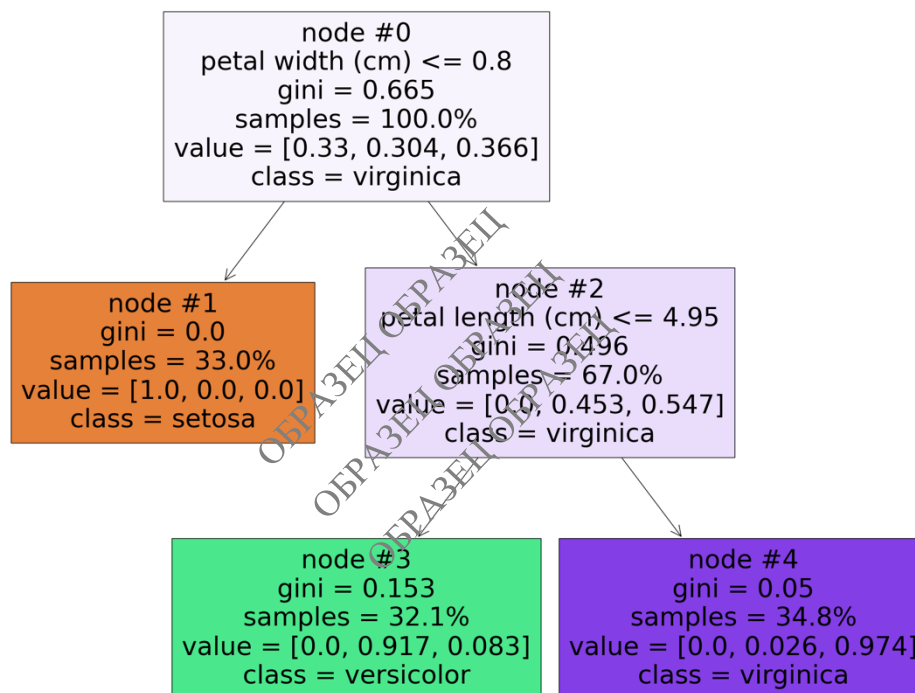


Рис. 4.2 – Пример визуализации дерева решений для глубины 2

Как вы думаете почему решающее дерево для глубин 4 и 6 выглядит одинаково?

#### Отчет по практической работе № 04.

- 1) титульный лист;
- 2) цели лабораторной работы;
- 3) листинг законченной программы;
- 4) график зависимости верности модели машинного обучения решающего дерева от его глубины (см. рис. 4.1);
- 5) консольные результаты работы модели машинного обучения на каких-либо произвольных значениях признаков, но взятых в пределах наблюдаемых значений;
- 6) график визуализации решающего дерева для заданных глубин (см. рис. 4.2);
- 7) выводы.

## Перечень использованных информационных ресурсов

1. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными.: Пер. с англ. - СПб.: ООО "Альфа-книга", 2017. - 480 с.: ил.
2. Сузи, Р.А. Язык программирования Python: учеб. пособие, М.: Интернет-Ун-т Информ. Технологий: Бином. Лаборатория знаний, 2006.
3. Буйначев, С.К., Боклаг, Н.Ю. Основы программирования на языке Python: учебное пособие, Екатеринбург: Издательство Уральского университета, 2014.
4. Сузи, Р.А. Язык программирования Python: учебное пособие, М.: Интернет- Университет Информационных Технологий (ИНТУИТ), 2016
5. Уэс, Маккинли Python и анализ данных: практическое пособие, Саратов: Профобразование, 2017.
6. Балджи, А.С., Хрипунова, М.Б. Математика на Python: учебно-методическое пособие, М.: Прометей, 2018.
7. Гуриков С. Р. Основы алгоритмизации и программирования на Python: учебное пособие, М.: Издательство "ФОРУМ", 2017.
8. Жуков Р. А. Язык программирования Python: практикум: Учебное пособие, М.: ООО "Научно- издательский центр ИНФРА-М", 2020.