Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Искусственный интеллект и машинное обучение»

Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

Руководитель практики:
Березина Виктория Андреевна

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Первичный анализ данных.

Цель: изучение программных средств для организации рабочего места специалиста по анализу данных и машинному обучению.

Ход работы

Данный набор данных предназначен для построения модели классификации. Данные о 199 экземплярах зерен пшеницы, по 66 экземпляров первого класса, 68 экземпляров второго класса, 65 экземпляров третьего класса. Для каждого экземпляра измерялись семь характеристик:

1) Площадь (Area);

2) Периметр (Perimetr);

3) Компактность (Compactness);

4) Длина ядра (Kernel Length);

5) Ширина ядра (Kernel Width)

6) Коэффициент асимметрии (Asymmetry Coeff)

7) Выемка ядра (Kernel Groove)

На основании этого набора данных требуется построить правило классификации, определяющее класс зерна по данным измерений.

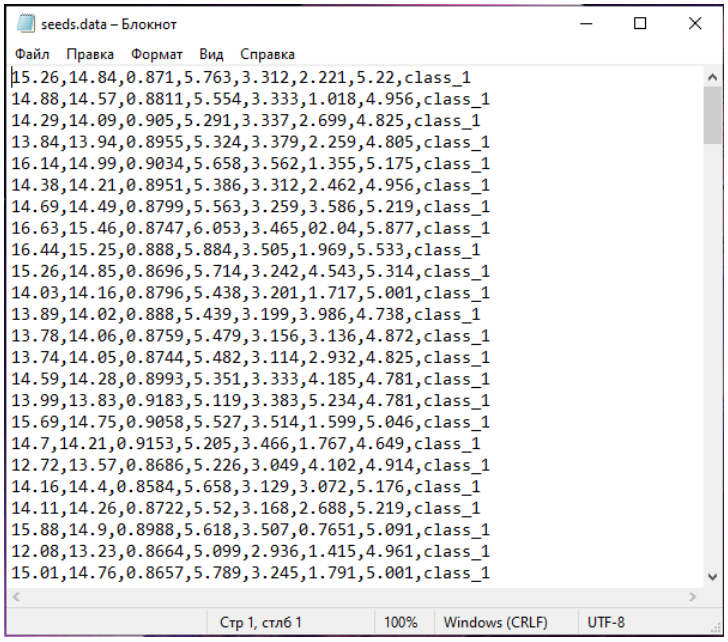

Рисунок 1. Внешний вид данных файла seeds.data

Запустим Jupyter Notebook и загрузим набором с использованием среды Python. Используем метод genfromtxt() из пакета scipy.

Лабораторная работа 1

Исследование набора данных

Подключение библиотеки NumPy и загрузка данных

```
[3] !wget https://raw.githubusercontent.com/IngaBesedina/AI_ML_1/main/SeedsData/seeds.data

--2024-02-13 12:25:35--  https://raw.githubusercontent.com/IngaBesedina/AI_ML_1/main/SeedsData/seeds.data
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10033 (9.8K) [text/plain]
Saving to: 'seeds.data'

seeds.data          100%[===================>]   9.80K  --.-KB/s    in 0s

2024-02-13 12:25:35 (62.3 MB/s) - 'seeds.data' saved [10033/10033]
```

```
[4] import numpy as np
    data_path = "seeds.data"
    data = np.genfromtxt(data_path, delimiter=",")
    print(data)

[[15.26   14.84    0.871  ...  2.221   5.22       nan]
 [14.88   14.57    0.8811 ...  1.018   4.956      nan]
 [14.29   14.09    0.905  ...  2.699   4.825      nan]
 ...
 [13.2    13.66    0.8883 ...  8.315   5.056      nan]
 [11.84   13.21    0.8521 ...  3.598   5.044      nan]
 [12.3    13.34    0.8684 ...  5.637   5.063      nan]]
```

Рисунок 2 – Загрузка данных файла seeds.data

Для поверхностного обзора исследуемой обучающей выборки используем подвыборку данных. Из представленного фрагмента видно, что data – это двумерный массив размером 199x8.

Тип переменной и форма (shape)

```
print ( "Data type : ", type(data) )
print ( "Data shape : ", data.shape )
print ( data[:10] )

Data type :  <class 'numpy.ndarray'>
Data shape :  (199, 8)
[[15.26   14.84    0.871   5.763   3.312   2.221   5.22       nan]
 [14.88   14.57    0.8811  5.554   3.333   1.018   4.956      nan]
 [14.29   14.09    0.905   5.291   3.337   2.699   4.825      nan]
 [13.84   13.94    0.8955  5.324   3.379   2.259   4.805      nan]
 [16.14   14.99    0.9034  5.658   3.562   1.355   5.175      nan]
 [14.38   14.21    0.8951  5.386   3.312   2.462   4.956      nan]
 [14.69   14.49    0.8799  5.563   3.259   3.586   5.219      nan]
 [16.63   15.46    0.8747  6.053   3.465   2.04    5.877      nan]
 [16.44   15.25    0.888   5.884   3.505   1.969   5.533      nan]
 [15.26   14.85    0.8696  5.714   3.242   4.543   5.314      nan]]
```

Рисунок 3 – Начальное исследование seeds.data

Проведем анализ типов различных значений. Получим значения восьмого столбца.

Получение типа набора данных, строки, элемента

```
[6] data1 = np.genfromtxt(data_path, delimiter=",", dtype=None)
    print('Shape of the dataset:', data1.shape)
    print('Dataset type:', type(data1))
    print('A single row of the dataset is type of:', type(data1[0]))
    print('Types of elements:', type(data1[0][1]), type(data1[0][7]))
    print('Dataset:')
    print(data1[:10])
```

```
Shape of the dataset: (199,)
Dataset type: <class 'numpy.ndarray'>
A single row of the dataset is type of: <class 'numpy.void'>
Types of elements: <class 'numpy.float64'> <class 'numpy.bytes_'>
Dataset:
[(15.26, 14.84, 0.871 , 5.763, 3.312, 2.221, 5.22 , b'class_1')
 (14.88, 14.57, 0.8811, 5.554, 3.333, 1.018, 4.956, b'class_1')
 (14.29, 14.09, 0.905 , 5.291, 3.337, 2.699, 4.825, b'class_1')
 (13.84, 13.94, 0.8955, 5.324, 3.379, 2.259, 4.805, b'class_1')
 (16.14, 14.99, 0.9034, 5.658, 3.562, 1.355, 5.175, b'class_1')
 (14.38, 14.21, 0.8951, 5.386, 3.312, 2.462, 4.956, b'class_1')
 (14.69, 14.49, 0.8799, 5.563, 3.259, 3.586, 5.219, b'class_1')
 (16.63, 15.46, 0.8747, 6.053, 3.465, 2.04 , 5.877, b'class_1')
 (16.44, 15.25, 0.888 , 5.884, 3.505, 1.969, 5.533, b'class_1')
 (15.26, 14.85, 0.8696, 5.714, 3.242, 4.543, 5.314, b'class_1')]
```

Рисунок 4 – Загрузка данных разного типа в массив

```
[ ] dt = np.dtype("f8, f8, f8, f8, f8, f8, f8, U30")
    data2 = np.genfromtxt(data_path, delimiter=",", dtype=dt)
    print('Shape of the dataset:', data2.shape)
    print('Dataset type:', type(data2))
    print('A single row of the dataset is type of:', type(data2[0]))
    print('Types of elements:', type(data2[0][1]), type(data2[0][7]))
    print('Dataset slice:')
    print(data2[:10])
```

```
Shape of the dataset: (199,)
Dataset type: <class 'numpy.ndarray'>
A single row of the dataset is type of: <class 'numpy.void'>
Types of elements: <class 'numpy.float64'> <class 'numpy.str_'>
Dataset slice:
[(15.26, 14.84, 0.871 , 5.763, 3.312, 2.221, 5.22 , 'class_1')
 (14.88, 14.57, 0.8811, 5.554, 3.333, 1.018, 4.956, 'class_1')
 (14.29, 14.09, 0.905 , 5.291, 3.337, 2.699, 4.825, 'class_1')
 (13.84, 13.94, 0.8955, 5.324, 3.379, 2.259, 4.805, 'class_1')
 (16.14, 14.99, 0.9034, 5.658, 3.562, 1.355, 5.175, 'class_1')
 (14.38, 14.21, 0.8951, 5.386, 3.312, 2.462, 4.956, 'class_1')
 (14.69, 14.49, 0.8799, 5.563, 3.259, 3.586, 5.219, 'class_1')
 (16.63, 15.46, 0.8747, 6.053, 3.465, 2.04 , 5.877, 'class_1')
 (16.44, 15.25, 0.888 , 5.884, 3.505, 1.969, 5.533, 'class_1')
 (15.26, 14.85, 0.8696, 5.714, 3.242, 4.543, 5.314, 'class_1')]
```

Рисунок 5 – Загрузка данных в массив с типом, определяемым пользователем

Построение простейшего графика для отображения различных проекций данных:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

# Данные из отдельных столбцов
Area = [] # Площадь
Perimeter = [] # Периметр
Compactness = [] # Компактность
Kernel_Length = [] # Длина ядра
Kernel_Width = [] # Ширина ядра
Asymmetry_Coeff = [] # Коэффициент ассиметрии
Kernel_Groove = [] # Выемка ядра

# Выполняем обход всей коллекции data2
for dot in data2:
    Area.append(dot[0])
    Perimeter.append(dot[1])
    Compactness.append(dot[2])
    Kernel_Length.append(dot[3])
    Kernel_Width.append(dot[4])
    Asymmetry_Coeff.append(dot[5])
    Kernel_Groove.append(dot[6])

# Строим графики по проекциям данных
plt.figure(1)
class_1, = plt.plot(Area[:67], Compactness[:67], 'ro', label='class_1')
class_2, = plt.plot(Area[67:135], Compactness[67:135], 'g^', label='class_2')
class_3, = plt.plot(Area[135:], Compactness[135:], 'bs', label='class_3')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Area')
plt.ylabel('Compactness')

plt.figure(2)
class_1, = plt.plot(Kernel_Length[:67], Kernel_Width[:67], 'ro',
label='class_1')
class_2, = plt.plot(Kernel_Length[67:135], Kernel_Width[67:135], 'g^',
label='class_2')
class_3, = plt.plot(Kernel_Length[135:], Kernel_Width[135:], 'bs',
label='class_3')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Kernel Length')
plt.ylabel('Kernel Width')

plt.figure(3)
class_1, = plt.plot(Kernel_Length[:67], Kernel_Groove[:67], 'ro',
label='class_1')
```

```
class_2, = plt.plot(Kernel_Length[67:135], Kernel_Groove[67:135], 'g^',
label='class_2')
class_3, = plt.plot(Kernel_Length[135:], Kernel_Groove[135:], 'bs',
label='class_3')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Kernel_Lengthf')
plt.ylabel('Kernel_Groove')

plt.figure(4)
class_1, = plt.plot(Perimeter[:67], Asymmetry_Coeff[:67], 'ro',
label='class_1')
class_2, = plt.plot(Perimeter[67:135], Asymmetry_Coeff[67:135], 'g^',
label='class_2')
class_3, = plt.plot(Perimeter[135:], Asymmetry_Coeff[135:], 'bs',
label='class_3')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Perimeter')
plt.ylabel('Asymmetry_Coeff')

plt.show()
```

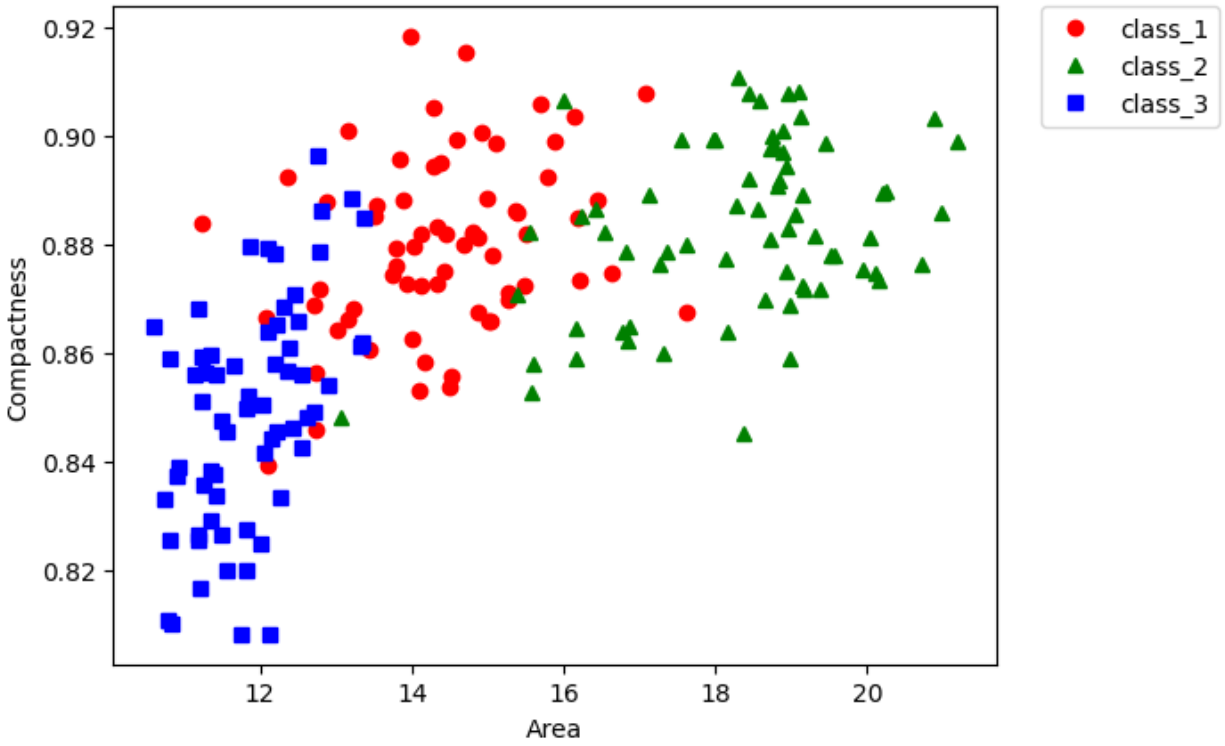

Рисунок 6 – Вывод графика для отображения различных проекций данных

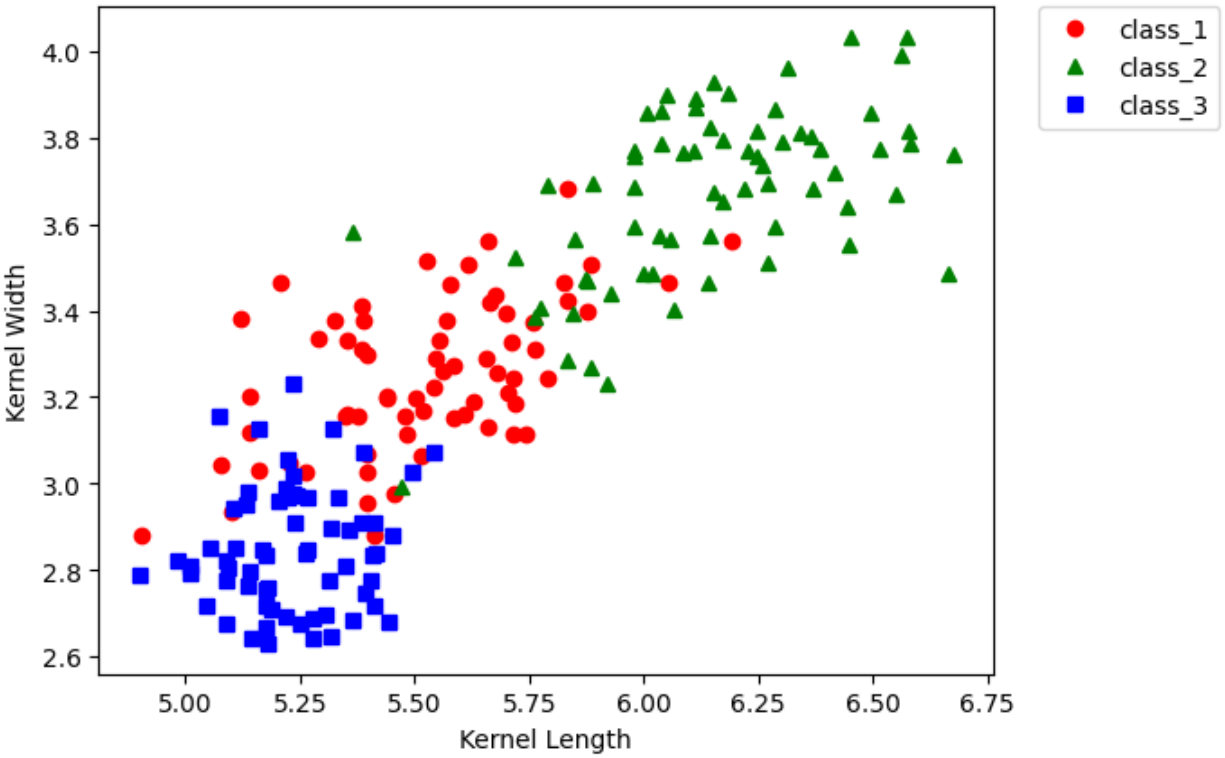

Рисунок 7 – Вывод графика для отображения различных проекций данных

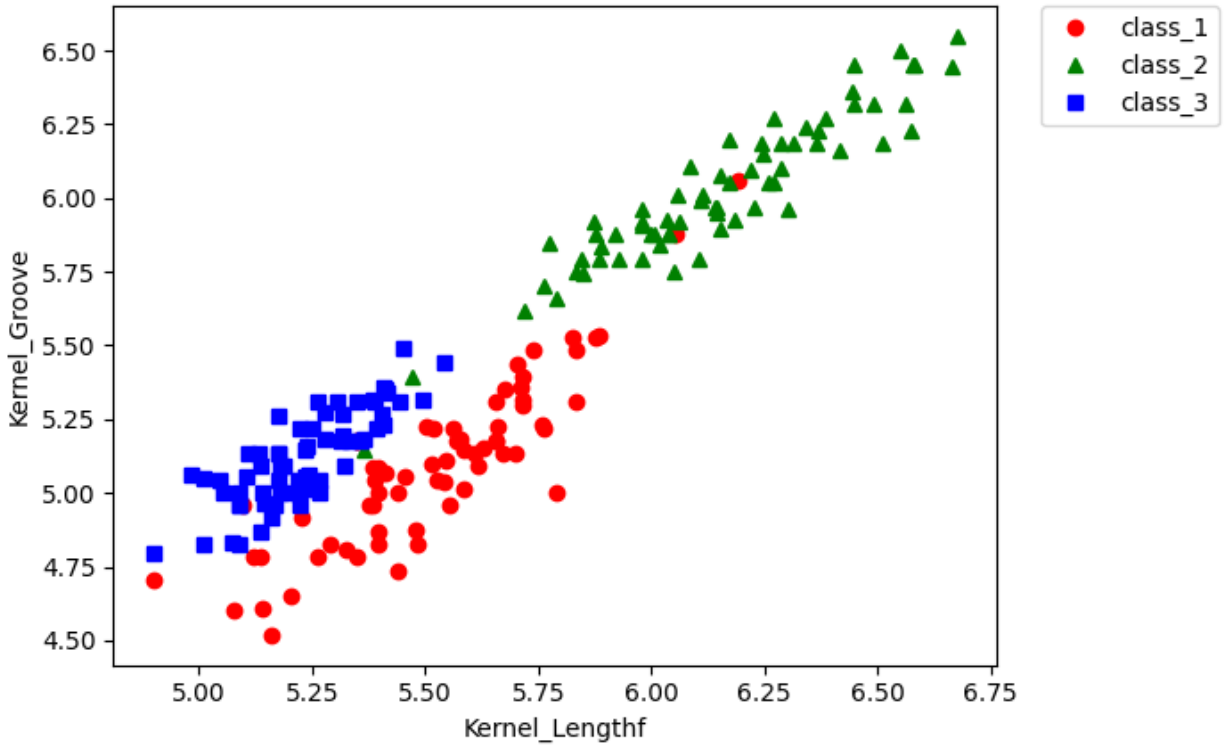

Рисунок 8 – Вывод графика для отображения различных проекций данных

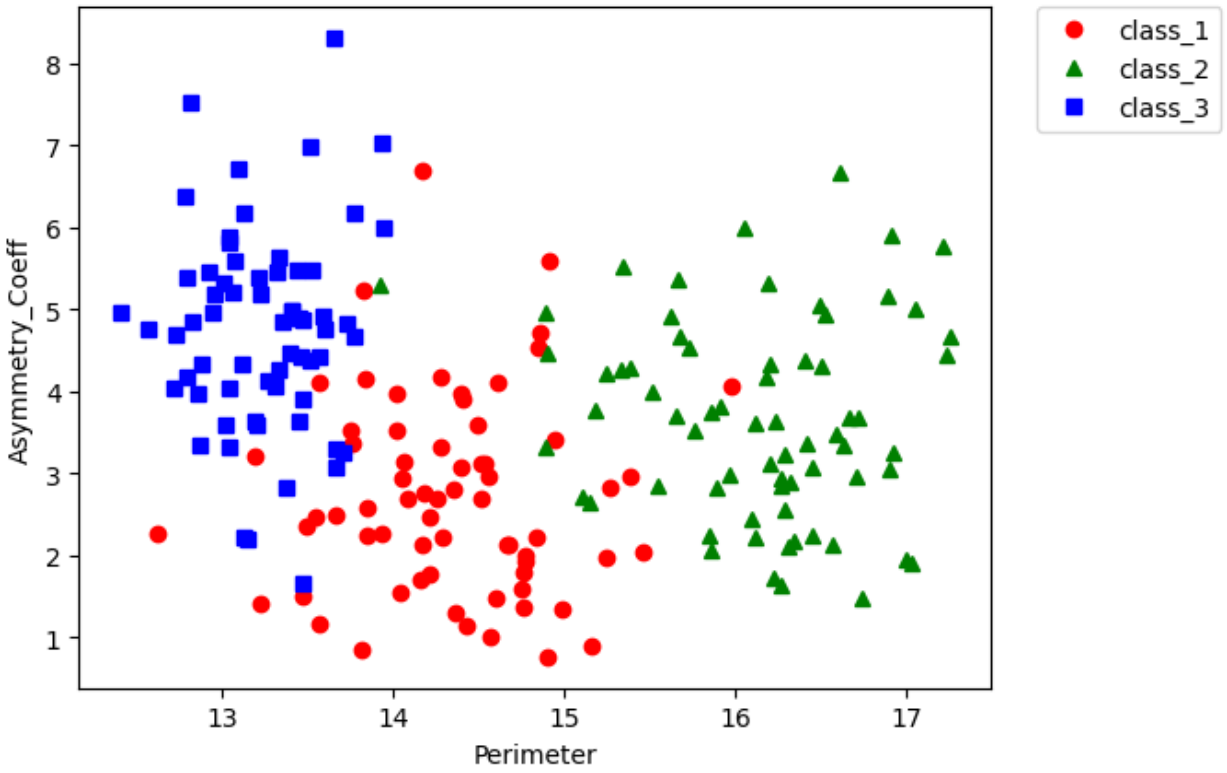

Рисунок 9 – Вывод графика для отображения различных проекций данных

Контрольные вопросы:

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science? Компьютер с высокой производительностью и достаточным объемом памяти; программное обеспечение для работы с данными; средства разработки; библиотеки для анализа данных и машинного обучения; виртуальные среды для изоляции проектов; инструменты для визуализации данных; надежное интернет-соединение;

2. **TensorFlow** — это библиотека Python, созданная Google в конце 2015 года для внутреннего использования в решениях для машинного обучения. Затем он был открыт для всего сообщества и с тех пор стал одной из крупнейших библиотек Python в области открытого исходного кода. **Основная задача:** создание моделей глубокого обучения.

Keras был разработан для исследовательского проекта, известного как Открытая нейро-электронная интеллектуальная операционная система роботов или ONEIROS, и фокусируется на более простом процессе создания

прототипов, удобном интерфейсе и модульности. Это скорее интерфейс для создания алгоритмов глубокого обучения, чем инструмент, и поддерживает как сверточные, так и рекуррентные нейронные сети. **Основная задача:** создание моделей глубокого обучения.

NumPy — это одна из первых библиотек, которая должна быть загружена для Python, независимо от того, будет ли пользователь использовать ее для машинного обучения или анализа основных данных. Эта библиотека машинного обучения Python предоставляет возможности для обработки данных и чисел. Библиотека создает объект N-мерного массива, в который пользователи могут помещать свои данные, и предлагает функции преобразования этих данных.

Объект массива также можно легко адаптировать к другим базам данных, что делает его естественным для машинного обучения. Помимо использования в научных и исследовательских приложениях, NumPy можно использовать для создания контейнера общих данных, которыми можно легко манипулировать. Библиотека поставляется с возможностями для решения задач, касающихся линейной алгебры, преобразований и случайных чисел.

3. Библиотеки python стали популярными в области машинного обучения по нескольким причинам: простота использования; многообразие библиотек; гибкость; поддержка облачных вычислений и т.д.

Вывод: в ходе выполнения практической работы были изучены программные средства для организации рабочего места специалиста по анализу данных и машинному обучению.