Лабораторная работа №1 по дисциплине "Искусственный интеллект и машинное обучение"

Выполнил: студент 2-го курса Звездин Алексей Сергеевич

Группа: ПИЖ-б-о-22-1

Руководитель практики: Березина Виктория Андреевна, ассистент кафедры информационных систем и технологий института цифрового развития

Тема работы: Первичный анализ данных

Цель работы: Изучение программных средств для организации рабочего места специалиста по анализу данных и машинному обучению

1. Первичный анализ данных

Описание: Данные являются результатами химического анализа вин, выращенных в одном регионе Италии, но полученных из трех разных сортов. В ходе анализа были определены 13 компонентов, обнаруженные в каждом из трех типов вин.

Практическое применение: Тестирование различных классификационных моделей.

```
!wget https://raw.githubusercontent.com/InternetHacker1123/bd_ai/main/laba1/mush.data
```

```
--2024-05-14 11:37:54--  https://raw.githubusercontent.com/InternetHacker1123/bd_ai/main/laba1/mush.data
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5151 (5.0K) [text/plain]
Saving to: ‘mush.data’

mush.data           100%[===================>]   5.03K  --.-KB/s    in 0s

2024-05-14 11:37:55 (51.7 MB/s) - ‘mush.data’ saved [5151/5151]
```

```
import numpy as np
data_path = "mush.data"
data = np.genfromtxt(data_path, delimiter=",")
print(data)
```

```
 [7.3 4.8 6.4 1.  nan]
 [6.4 1.8 8.6 1.1 nan]
 [6.4 1.8 8.8 8.8 nan]
 [6.8 1.6 8.6 8.4 nan]
 [7.7 4.  6.8 1.4 nan]
 [6.4 4.4 8.6 1.4 nan]
 [6.4 4.8 8.8 8.8 nan]
 [6.  4.  4.8 8.8 nan]
 [6.3 4.8 8.4 1.8 nan]
 [6.7 4.8 8.6 1.4 nan]
 [6.3 4.8 8.8 1.4 nan]
 [8.8 1.7 8.8 8.3 nan]
 [6.8 4.1 8.3 1.4 nan]
 [6.7 4.4 8.7 1.8 nan]
 [6.7 4.  8.1 1.4 nan]
 [6.4 1.8 8.  8.3 nan]
 [6.8 4.  8.1 1.  nan]
 [6.1 4.4 8.4 1.4 nan]
 [8.3 4.  8.8 8.8 nan]]
```

3. Тип переменной и форма (shape)

```
print ( "Data type : ", type(data) )
print ( "Data shape : ", data.shape )
print ( data[-4:] )
```

```
Data type :  <class 'numpy.ndarray'>
Data shape :  (150, 5)
[[6.4 1.8 8.  8.3 nan]
 [6.8 4.  8.1 1.  nan]
 [6.1 4.4 8.4 1.4 nan]
 [8.3 4.  8.8 8.8 nan]]
```

4. Получение типа набора данных, строки, элемента

```
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][4]))
print('Dataset:')
print(data1)
```

```
 (6.4, 4.4, 8.6, 1.4, b'Russula-claroflava')
 (6.4, 4.8, 8.8, 8.8, b'Russula-claroflava')
 (6. , 4. , 4.8, 8.8, b'Russula-claroflava')
 (6.3, 4.8, 8.4, 1.8, b'Russula-claroflava')
 (6.7, 4.8, 8.6, 1.4, b'Russula-claroflava')
 (6.3, 4.8, 8.8, 1.4, b'Russula-claroflava')
 (8.8, 1.7, 8.8, 8.3, b'Russula-claroflava')
 (6.8, 4.1, 8.3, 1.4, b'Russula-claroflava')
 (6.7, 4.4, 8.7, 1.8, b'Russula-claroflava')
 (6.7, 4. , 8.1, 1.4, b'Russula-claroflava')
 (6.4, 1.8, 8. , 8.3, b'Russula-claroflava')
 (6.8, 4. , 8.1, 1. , b'Russula-claroflava')
 (6.1, 4.4, 8.4, 1.4, b'Russula-claroflava')
 (8.3, 4. , 8.8, 8.8, b'Russula-claroflava')]
<ipython-input-4-be1180784c4e>:1: VisibleDeprecationWarning: Reading unicode strings without specifying the encoding argument is
```

5. Указание типа столбцов при загрузке данных

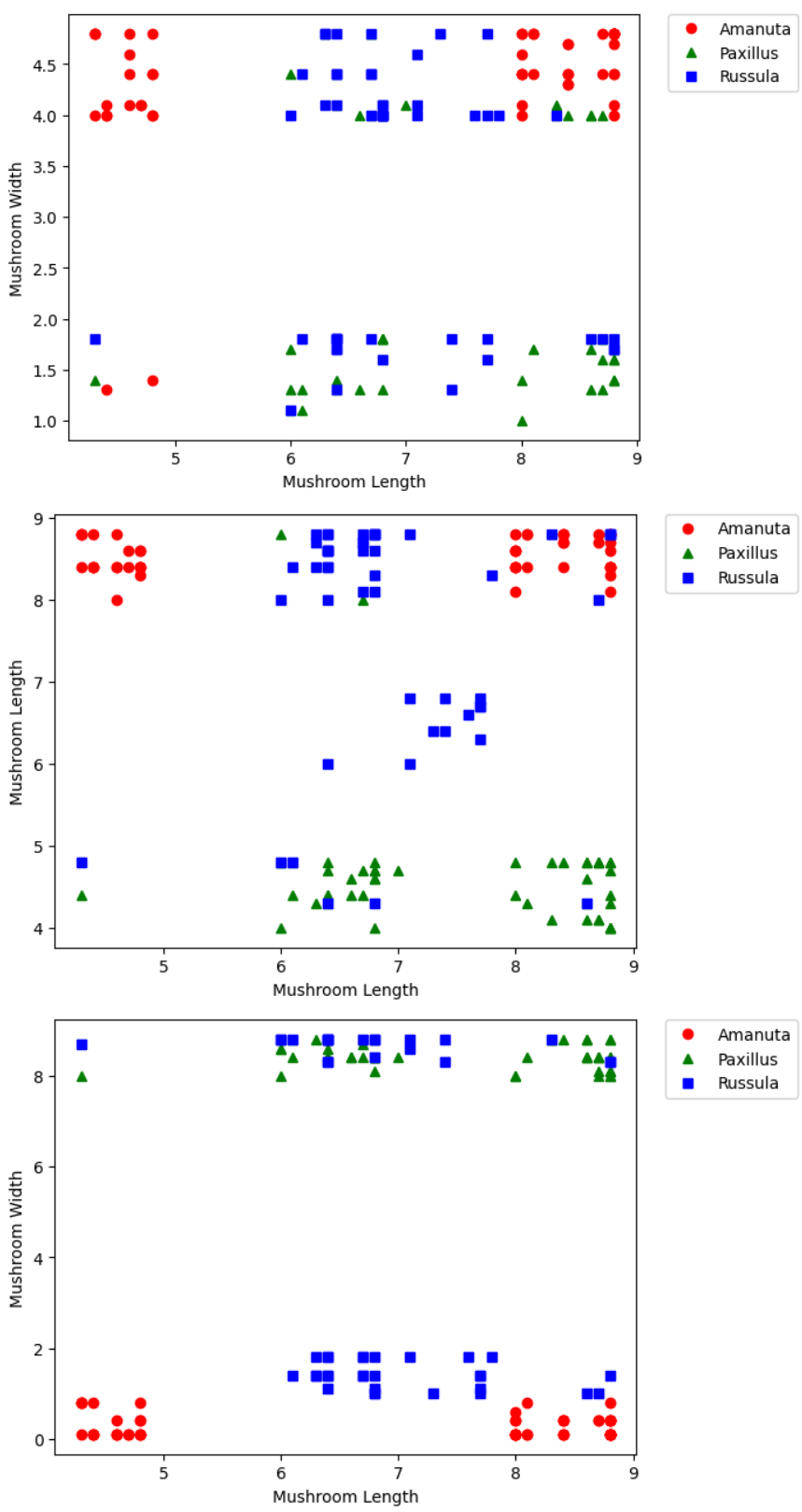
```
dt = np.dtype("f8, f8, f8, f8, U30")
data2 = np.genfromtxt(data_path, delimiter=",", dtype=dt)
print('Shape of the dataset:', data2.shape)
print('Dataset type:', type(data2))
print('A single row of the dataset is type of:', type(data2[0]))
print('Types of elements:', type(data2[0][1]), type(data2[0][4]))
print('Dataset slice:')
print(data2[:10])
```

```
Shape of the dataset: (150,)
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:
[(8.8, 4.8, 8.4, 0.1, 'Amanita-muscaria')
 (4.3, 4. , 8.4, 0.1, 'Amanita-muscaria')
 (4.7, 4.1, 8.4, 0.1, 'Amanita-muscaria')
 (4.6, 4.8, 8.8, 0.1, 'Amanita-muscaria')
 (8. , 4.6, 8.4, 0.1, 'Amanita-muscaria')
 (8.4, 4.3, 8.7, 0.4, 'Amanita-muscaria')
 (4.6, 4.4, 8.4, 0.4, 'Amanita-muscaria')
 (8. , 4.4, 8.8, 0.1, 'Amanita-muscaria')
 (4.4, 1.3, 8.4, 0.1, 'Amanita-muscaria')
 (4.3, 4.8, 8.8, 0.8, 'Amanita-muscaria')]
```

6. Построение графиков с использованием Matplotlib

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

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
apulum, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Amanuta')
popovii, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Paxillus')
pilosum, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Russula')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Mushroom Length')
plt.ylabel('Mushroom Width')

plt.figure(2)
apulum, = plt.plot(sepal_length[:50], petal_length[:50], 'ro', label='Amanuta')
popovii, = plt.plot(sepal_length[50:100], petal_length[50:100], 'g^', label='Paxillus')
pilosum, = plt.plot(sepal_length[100:150], petal_length[100:150], 'bs', label='Russula')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Mushroom Length')
plt.ylabel('Mushroom Length')

plt.figure(3)
```

```
apulum, = plt.plot(sepal_length[:50], petal_width[:50], 'ro', label='Amanuta')
popovii, = plt.plot(sepal_length[50:100], petal_width[50:100], 'g^', label='Paxillus')
pilosum, = plt.plot(sepal_length[100:150], petal_width[100:150], 'bs', label='Russula')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Mushroom Length')
plt.ylabel('Mushroom Width')

plt.show()
```



Контрольные вопросы:

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?

Язык программирования: Python часто используется в Data Science благодаря своей простоте, обширному сообществу и богатому экосистемному набору библиотек. Среды разработки: Jupyter Notebooks, Spyder, и VS Code популярны для разработки и экспериментов с кодом. Системы управления версиями: Git используется для отслеживания изменений в коде. Хранилища данных: Для хранения и управления данными могут использоваться базы данных (например, PostgreSQL, MySQL) или облачные хранилища (например, AWS S3, Google Cloud Storage). Контейнеризация и оркестрация: Docker и Kubernetes позволяют создавать и масштабировать приложения и их компоненты.

2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.

NumPy: Предоставляет поддержку для работы с многомерными массивами и математическими функциями, основной инструмент для работы с данными. Pandas: Предоставляет структуры данных для эффективной работы с табличными данными, облегчая их анализ и манипуляции. Matplotlib и Seaborn: Используются для визуализации данных и создания графиков. Scikit-learn: Предоставляет простой и эффективный инструментарий для анализа данных и реализации алгоритмов машинного обучения. TensorFlow и PyTorch: Библиотеки для глубокого обучения, позволяющие строить и обучать нейронные сети. Keras: Высокоуровневый интерфейс для TensorFlow и PyTorch, упрощающий создание и обучение моделей.

3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?

Простота использования: Python имеет чистый синтаксис и прост в изучении, что облегчает работу разработчикам. Большое сообщество: Сильное и активное сообщество обеспечивает обмен опытом, решение проблем и разработку новых инструментов. Богатая экосистема библиотек: Python обладает множеством библиотек для машинного обучения, что упрощает разработку и эксперименты с моделями. Интеграция с другими языками: Возможность интеграции с библиотеками, написанными на C/C++ (например, библиотеки для вычислений) позволяет улучшить производительность. Широкое применение: Python активно используется во многих областях, и его простота делает его привлекательным для специалистов различных профилей, включая исследователей, инженеров и аналитиков данных.