Katedra Systemów Informacyjnych
Algorytmy analizy danych
Kierunek Informatyka, WE, sem. VII
Laboratorium nr 2

Grzegorz Cichosz

(z tego co pamiętam w momencie wykonywania laborki były problemy z środowiskiem rattle I miieliśmy)

Analizy danych z wykorzystaniem skryptów środowiska Phyton
a. Zapoznaj się ze skryptem języka Phyton zawartym w p1.txt
Co oznaczają poszczególne fragmenty skryptu, za co są odpowiedzialne.
Oczywiście większość odpowiedzi znajdziesz w komentarzach, ale sprawdź jak działają poszczególne
funkcje. W Internecie znajdziesz dokumentacje Phytona oraz dokumentacje pakietu pandas

Na początku skryptu znajduje się nagłówek informujący, że program ma być uruchamiany w Pythonie 3 oraz że plik zapisany jest w kodowaniu UTF-8, co pozwala poprawnie wyświetlać polskie znaki.

Następnie importowane są biblioteki:
pandas - do wczytywania i przetwarzania danych w postaci tabelarycznej,
numpy - do operacji numerycznych (np. wyszukiwanie unikalnych wartości, warunki),
matplotlib - do tworzenia wykresów.

Dane są wczytywane z pliku iris.data.wn za pomocą funkcji read_csv. Ustawienie header=None oznacza, że plik nie zawiera nagłówków kolumn, dlatego kolumny są numerowane od 0 do 4.

Za pomocą funkcji len(df) wypisywana jest liczba rekordów w zbiorze danych.

Następnie z kolumny 4 pobierane są etykiety klas (gatunki irysów) i przy użyciu np.unique() sprawdzane jest, jakie klasy występują w zbiorze.

Funkcja df.describe() wyświetla podstawowe statystyki opisowe danych, takie jak średnia, odchylenie standardowe, minimum, maksimum oraz kwartyle. Dzięki temu można wstępnie ocenić dane.

Polecenie df.tail(4) służy do wyświetlenia kilku ostatnich wierszy zbioru w celu sprawdzenia, czy dane zostały poprawnie wczytane.

W kolejnej części skryptu przygotowywane są dane do wykresu. Wybierane są tylko dwa gatunki (setosa i versicolor), a etykiety klas są kodowane liczbowo przy pomocy funkcji np.where. Następnie wybierane są dwie cechy (kolumny 0 i 2), które będą użyte jako osie wykresu.

Za pomocą funkcji plt.scatter rysowany jest wykres rozrzutu dla dwóch klas. Dodane są opisy osi oraz legenda, a wykres zostaje wyświetlony przy użyciu plt.show().

Kolejna część skryptu sprawdza, czy w danych występują braki. Funkcja df.isnull().sum().sum() zwraca łączną liczbę brakujących wartości. Dodatkowo wypisywane są wiersze, w których występują wartości NaN.

Brakujące dane są uzupełniane średnią arytmetyczną z danej kolumny. Skrypt przechodzi po kolejnych kolumnach i wstawia średnią w miejsca, gdzie wcześniej występowały wartości brakujące.

Po imputacji ponownie sprawdzana jest liczba braków danych, aby upewnić się, że operacja zakończyła się poprawnie.

Następnie wykonywana jest standaryzacja danych. Dla każdej cechy odejmowana jest średnia i wynik dzielony jest przez odchylenie standardowe. Dzięki temu dane mają średnią bliską 0 i odchylenie standardowe równe 1.

Na końcu skrypt zapisuje przetworzone dane do pliku wynikowego przy użyciu funkcji to_csv.
b. Otwórz nowy projekt w środowisku PyCharm
c. Utwórz nowy plik programu Python (nadaj mu nazwę np. p1 – czyli plik będzie się nazywał p1.py)
d. Skopiuj skrypt do p1.py zawartość z pliku p1.txt
e. Skopiuj do katalogu z projektem plik iris.data.wn
Jeśli skopiujesz go do innego katalogu, to musisz skorygować ścieżkę dostępu do tego pliku w skrypcie
Pythona

(odpalam to w shellu zamiast w pycharmie)

```
# zmień nazwę
mv p1.txt p1.py

#nadaj uprawnienia do uruchomienia
chmod +x p1.py

#robimy virtualne środowisko
python3 -m venv venv


#aktywujemy środowisko
source venv/bin/activate

#instalujemy zależności (robiłem to dla lab 4 przed chwilą)
pip install pandas seaborn matplotlib scipy
```

lekko sypie błędem
df.to_csv('C:\D\Edukacja\Algorytmy analizy danych\laboratorium\lab 1\iris.data.output')
między innymi dlatego że nie mam windowsa a to ścieżka z windowsa

df = pd.read_csv("iris.data.wn", header=None)
df.to_csv("iris.data.output", index=False, header=False)

poprawiono ścieżki

surowy output:

liczba rekordów we wczytamym pliku wynosi: 150
liczba etykiet we wczytamym zbiorze wynosi: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']

	0	1	2	3
count	149.000000	149.000000	149.000000	150.000000
mean	5.858389	3.051678	3.744966	1.218667
std	0.829775	0.434120	1.762348	0.814004
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.300000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	4.700000

	0	1	2	3	4
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

/home/talandar/workspace/aad/lab2/./p1.py:46: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
plt.show()
Tyle wierszy ma null: 3

	0	1	2	3	4
24	6.8	NaN	1.9	0.2	Iris-setosa
64	NaN	2.9	3.6	1.3	Iris-versicolor
108	6.7	2.5	NaN	1.8	Iris-virginica

a teraz pokażmy jak alternatywnie mozemy ich wyszukać

```
      0    1    2    3             4
24   6.8  NaN  1.9  0.2      Iris-setosa
64   NaN  2.9  3.6  1.3  Iris-versicolor
108  6.7  2.5  NaN  1.8   Iris-virginica
NaN występuje teraz:  0  razy
          0         1         2         3             4
0   -0.917052  1.036198 -1.335080 -1.251427     Iris-setosa
1   -1.158894 -0.119442 -1.335080 -1.251427     Iris-setosa
2   -1.400736  0.342814 -1.392014 -1.251427     Iris-setosa
3   -1.521657  0.111686 -1.278146 -1.251427     Iris-setosa
4   -1.037973  1.267326 -1.335080 -1.251427     Iris-setosa
..        ...       ...       ...       ...            ...
145  1.017684 -0.119442  0.828407  1.328413  Iris-virginica
146  0.534000 -1.275082  0.714539  0.837015  Iris-virginica
147  0.775842 -0.119442  0.828407  0.959864  Iris-virginica
148  0.413079  0.805070  0.942275  1.328413  Iris-virginica
149  0.050316 -0.119442  0.771473  0.714165  Iris-virginica

[150 rows x 5 columns]
```

f. Uruchom skrypt i prześledź jak działa i jakiego rodzaju informacji dostarcza jego przetworzenie
Skomentuj poszczególne informacje (wyniki przetworzonego skryptu) w sprawozdaniu

Po uruchomieniu skryptu program wypisuje podstawowe informacje o zbiorze danych iris.
Na początku wyświetlana jest liczba rekordów, co pozwala sprawdzić ile obserwacji znajduje się w zbiorze.
Następnie wypisywane są unikalne etykiety klas, dzięki czemu wiadomo jakie gatunki irysów występują w danych.
Funkcja describe() pokazuje podstawowe statystyki opisowe dla cech liczbowych, takie jak średnia, odchylenie standardowe oraz wartości minimalne i maksymalne. Pozwala to wstępnie ocenić zakresy danych oraz ich rozrzut.
Polecenie tail() służy do sprawdzenia kilku ostatnich wierszy zbioru i kontroli poprawności wczytania danych.
W kolejnej części sprawdzane są braki danych. Program wypisuje ile jest wartości brakujących oraz w których wierszach one występują. Następnie braki te są uzupełniane średnią z danej kolumny. Po tej operacji licba brakujących wartości spada do zera.
Na końcu wykonywana jest standaryzacja danych. Po tym przekształceniu cechy mają porównywalną skalę, co jest przydatne w dalszym przetwarzaniu danych.

g. Dopisz kod który dokona normalizacji danych pierwotnych z iris.data.wn a następnie zapisze te dane do pliku wynikowego

Do skryptu został dopisany fragment kodu, który wykonuje normalizację danych metodą min-max. Normalizacja została wykonana dla kolumn z cechami liczbowymi (bez kolumny z etykietą klasy). Dane po normalizacji zostały zapisane do osobnego pliku wynikowego iris.data.normalized.csv.

h. A może potrafisz przygotować wykres rozrzutu punktów dla pierwszej i drugiej współrzędnej (pierwszej I drugiej kolumny) z danych, ale po wykonanej standaryzacji

Po wykonaniu standaryzacji przygotowany został wykres rozrzutu dla pierwszej i drugiej kolumny danych. Wykres pokazuje rozkład punktów po standaryzacji oraz pozwala porównać klasy między sobą. Po standaryzacji zmienia się skala osi, jednak ogólny układ punktów pozostaje podobny jak przed przekształceniem.

```
df_std = df.copy()

for i in range(len(df_std.columns) - 1):
    col = df_std[i].astype(float)
    df_std[i] = (col - col.mean()) / col.std()

X = df_std.iloc[:, [0, 1]].values
labels = df_std.iloc[:, 4].values

plt.figure()
for cls in np.unique(labels):
    mask = labels == cls
    plt.scatter(X[mask, 0], X[mask, 1], label=str(cls))

plt.xlabel("Kolumna 1 po standaryzacji (V1)")
plt.ylabel("Kolumna 2 po standaryzacji (V2)")
plt.legend()
plt.savefig("scatter_standardized.png")
plt.close()
```