Praktyczne zastosowanie metod SI – Laboratorium nr 2

MATPLOTLIB - WYKRESY

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(-5, 5.1, 0.1)
b = a**2
plt.plot(a, b)
plt.show()

plt.plot(a, b, 'b-') # niebieskie punkty linia ciągła
#plt.plot(a, b, 'ro') # czerwone punkty
#plt.plot(a, b, 'k.') # czarne małe punkty
#plt.plot(a, b, 'g--') # zielona linia przerywania
#plt.scatter(a, b)

plt.xlabel(‘os x’)
plt.ylabel(‘os y’)

plt.show() # niezbędne polecenie, bez niego wykres się nie pojawi na ekranie
```

SUBPLOT

```
import matplotlib.pyplot as plt
import numpy as np

a = np.arange(-5, 5.1, 0.1)
b = a**2


plt.subplot(221) # wiersze, kolumny, numer wykresu
plt.plot(a, b)
plt.subplot(222)
plt.plot(a, b, 'b-') # niebieskie punkty linia ciągła
plt.subplot(223)
plt.plot(a, b, 'ro') # czerwone punkty
plt.subplot(224)
plt.plot(a, b, 'k.') # czarne małe punkty

plt.show()
```

Można też skorzystać z poniższej funkcji:

```
#fig, axes = plt.subplots(2, 2, subplot_kw=dict(polar=True)) axes[0, 0].plot(x, y)
#axes[0, 0]. plot(a, b) # indeks wiersza i kolumny w gridzie
…
#plt.show()
```

Przydatna może być funkcja plt.savefig() – zapisywanie wykresu do pliku.

WYKRES 3D

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = x**2 + y**2

ax.plot(X, Y, Z)
#ax.plot_wireframe(X, Y, Z)
#ax.plot_surface(X, Y, Z)

plt.show()
```

HISTOGRAM

```
X = np.random.normal(size=1000) # rozklad normalny; np.random.rand – daje rozkład jednostajny
plt.hist(x, 100) # 100 koszykow
plt.show()
```

IMSHOW / OPENCV (wyświetlanie i przetwarzanie obrazów)

```
import cv2 # instalacja przez konsolę: pip install opencv-python
import matplotlib.pyplot as plt

image = cv2.imread('zdjecie.jpg')
cv2.imshow('tytul okna', image) # najlepiej, gdy typ image to uint8
cv2.waitKey(0) # wyświetlanie okna do naciśnięcia klawisza
```

LUB

```
plt.imshow(image, cmap='gray') # wyświetlenie w skali szarości, można zrobić to też przykładowo
#pisząc np.mean(image, axis=2) Przydatna opcja: vmin, vmax do ustalenia zakresu
plt.show()
```

KONWERSJA DO SZAROŚCI

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

ZMIANA ROZMIARU OBRAZU

```
image = cv2.resize(image, (width, height))
```

Można również potraktować obraz jako macierz i zmniejszać go funkcją **block_reduce** z pakietu **scikit-image**. Mamy wtedy pełną kontrolę nad funkcją, która uśrednia wartość pikseli)

SCIPY + SCIKIT-LEARN

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()
#print(iris) # slownik z kluczami iris.keys()
X = iris.data # wartosci cech
y = iris.target # informacje o klasie

plt.scatter(X[:, 0], X[:, 1], c=y) # tak?
#plt.scatter(X[:, 0], X[:, 2], c=y) # a moze tak?

#fig = plt.figure()
#ax = fig.add_subplot(111, projection='3d')
#ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y) # a moze tak?
plt.show()
```

Rozwiązanie: znaleźć składowe główne

```
pca = PCA(n_components=3)
X_transformed = pca.fit_transform(X) # ogolnie w sklearn czy scipy – fit – uczenie, predict/transform
#jako wykorzystanie nauczonego modelu
```

Widać wyraźne rozgraniczenie między klasami.

A co jeśli nie mamy informacji o klasie? W tym przypadku można poszukać skupisk tzn. dokonać klasteryzacji. Kmeans – klasyfikacja nienadzorowana, nie potrzebujemy do uczenia próbek ze znaną klasą.

```
from sklearn.cluster import KMeans
... # wczytanie danych
kmeans = KMeans(n_clusters=3, verbose=1)
kmeans.fit(X)
y_pred = kmeans.predict(X)
```

Zarys algorytmu K-środków:

1. Wybór początkowych pozycji K centrów (losowo lub jako K punktów z danych).
2. Przypisz etykiety do próbek zgodnie z zasadą (etykieta *y* dla *i*-tej próbki), że o klasie decyduje najbliższe centrum:

$$y_i = \arg\min_j d(x_i - c_j)$$

3. Zaktualizuj położenia centrów – jako średnie arytmetyczne współrzędnych punktów z daną etykietą

Proszę poeksperymentować z parametrami kmeans (n_clusters, max_iter, tol, verbose – sterowanie wyświetlaniem informacji, algorithm), obserwując wizualizację i mierząc adjusted_rand_score (Jest to miara podobieństwa między dwoma sposobami klasteryzacji. Przyjmuje wartości z zakresu [-1, 1] – im bliżej lewego brzegu tym gorzej, im bliżej prawego tym lepiej. Losowe etykietowanie daje wartość 0.)

```
from sklearn.metrics import adjusted_rand_score
print(adjusted_rand_score(y, y_pred))
```

Proszę wyrysować w 3D znalezione centra na wykresie z próbkami (atrybut cluster_centers_) (wizualizacja po PCA)

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Proszę porównać z

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html

oraz

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans2.html#scipy.cluster.vq.kmeans2

Naiwny klasyfikator Bayesa – uczenie nadzorowane (wymaga próbek ze znaną klasą do uczenia)

Zasada działania:

Prawdopodobieństwo klasy pod warunkiem próbki obliczane jest jako:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Skąd wziąć *P(x|c)*? Możemy przyjąć, że atrybuty są niezależne. Wówczas będzie to iloczyn prawdopodobieństw:

$$\prod_{i=1}^{n} P(x_i|C)$$

A co z mianownikiem? Możemy go pominąć, ponieważ nie jest zależny od klasy, którą rozpatrujemy. (mianownik liczylibyśmy z tw. o prawdopodobieństwie całkowitym). Zatem wzór na *P(x|c)* upraszcza się. Jako klasę *y* dla próbki *x* wybieramy tą, dla której to prawdopodobieństwo jest największe (*n* – liczba atrybutów).

$$y_i = \arg\max_k \prod_{j=1}^{n} P(x_j|C_k) \cdot P(C_k)$$

Szkic użycia:

```
from sklearn.naive_bayes import GaussianNB
from sklearn import datasets
import numpy as np.
iris = datasets.load_iris()
X = iris.data # wartosci cech
y = iris.target # informacje o klasie

clf = GaussianNB()
clf.fit(X, y)
print(np.equal(y, clf.predict(X))) # clf.predict(X) wykonuje klasyfikację na X, porównujemy ją z
#oryginalnym y
```

Nie ma sensu liczenie dokładności (accuracy) ręcznie. W praktyce do tej (i innych metryk) stosuje się funkcje z pakietu sklearn.metrics:

(schemat użycia: nazwa_funkcji(y, y_predicted)

confusion_matrix – macierz pomyłek, poniżej przykład dla klasyfikacji binarnej:

		KLASY RZECZYWISTE	
		KLASA POZYTYWNA	**KLASA NEGATYWNA**
KLASY Z PREDYKCJI	**POZYTYWNA**	True Positive (TP)	False Positive (FP)
	NEGATYWNA	False Negative (FN)	True Negative (TN)

accuracy_score (ACC)– najprostsza miara, dokładność liczona jako iloraz liczby poprawnie zaklasyfikowanych przykładów do wszystkich przykładów. Dla klasyfikacji binarnej **ACC = (TP+TN) / (TP+FP+FN+TN)**

balanced_accuracy_score – podobnie jak accuracy, z tym że uwzględnia częstość klas

recall_score (czułość, TPR) – zdolność klasyfikatora do zaklasyfikowania przykładu pozytywnego jako pozytywyny. **TPR = TP / (TP+FN)**

specifity (specyficzność, TNR) – Mierzy "szansę" wykrycia przypadku negatywnego. **TNR = TN / (TN+FP)**

precision_score (precyzja, wartość predykcyjna dodatnia, PPV): **PPV = TP / (TP+FP)** – mierzy "szansę", że przypadek zaklasyfikowany jako pozytywny jest pozytywny w rzeczywistości

precision (wartość predykcyjna ujemna, NPV): **NPV = TN / (TN+FN)** – mierzy "szansę", że przypadek zaklasyfikowany jako negatywny jest negatywny w rzeczywistości

f1_score – średnia harmoniczna precyzji i czułości: **F1 = 2*precision*recall / (precision+recall) = 2TP / (2TP+FP+FN)**

roc_auc_score - pole pod krzywą ROC, rysowaną w układzie (1-specyficzność, czułość). Im większe, tym lepiej.

Uwaga: liczba klas w zbiorze iris wynosi więcej niż 2, tak więc niektóre miary (takie jak AUC są stosowane w wariancie one vs. rest – czyli jedna klasa jest traktowana jako pozytywna, a reszta jako negatywne, i tak dla każdej z klas).

ZADANIE DOMOWE

Proszę napisać skrypt, który dzieli losowo zbiór (X, y) (w tym wypadku iris) na uczący i testowy w proporcji od 90:10 do 10:90 (ze zmianą co 10), a następnie uczy 10-krotnie klasyfikator (losowy podział na dane uczącą i testową) i oblicza kilka wybranych metryk (w tym accuracy_score), a następnie uśrednia wyniki. Na koniec należy narysować wykres tych metryk w zależności od wielkości zbioru uczącego (wskazówka: można przechowywać wyniki z każdej z metryk jako listę I narysować kilka krzywych na tym samym wykresie).