SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 7 Data 21.12.2024 Temat: Klasyfikacja danych przy użyciu algorytmów uczenia maszynowego. Wariant 8	Imię Nazwisko Hubert Mentel Informatyka II stopień, niestacjonarne, 1 semestr, gr.1a

1. Zadanie:

Celem ćwiczenia jest zapoznanie się z metodami klasyfikacji danych przy użyciu algorytmów uczenia maszynowego. Studenci będą mieli możliwość wyboru różnych zbiorów danych dostępnych publicznie, takich jak MNIST, Iris, czy Breast Cancer, aby przeprowadzić klasyfikację i porównać efektywność różnych algorytmów.

Ćwiczenie obejmuje:

• Przygotowanie danych do klasyfikacji,

• Implementację różnych algorytmów klasyfikacyjnych,

• Optymalizację modeli,

• Porównanie wyników za pomocą odpowiednich miar jakości,

• Wizualizację wyników klasyfikacji.

Wariant 8: Zbiór danych Pima Indians Diabetes

Zbiór Pima Indians Diabetes zawiera dane o zdrowiu i stylu życia kobiet z plemienia Pima. Celem zadania jest klasyfikacja na podstawie 8 cech (np. wiek, BMI, liczba ciąż) w celu przewidywania, czy dana osoba ma cukrzycę. Link do danych:
https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Pliki dostępne są na GitHubie pod linkiem:
https://github.com/HubiPX/NOD/tree/master/Zadanie%207

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

# 1. Wczytanie zbioru danych
df = pd.read_csv('diabetes.csv')
print(df.to_string())
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1

2. Przygotowanie danych

X = dane.drop("Outcome", axis=1) # cechy

y = dane["Outcome"] # etykiety

3. Podział na zbiór treningowy i testowy

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

4. Normalizacja danych

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

5. Trenowanie klasyfikatorów

Regresja logistyczna

log_reg = LogisticRegression(random_state=42)

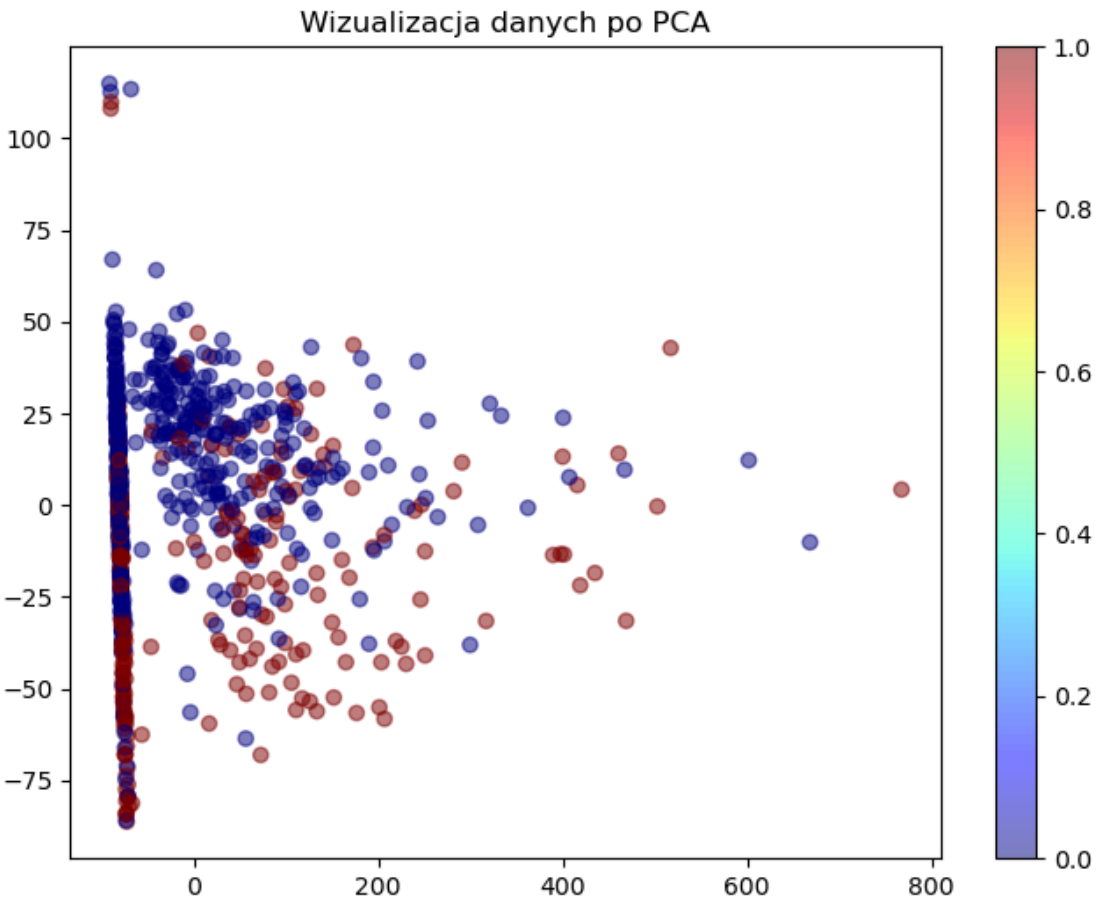
log_reg.fit(X_train, y_train)

SVM

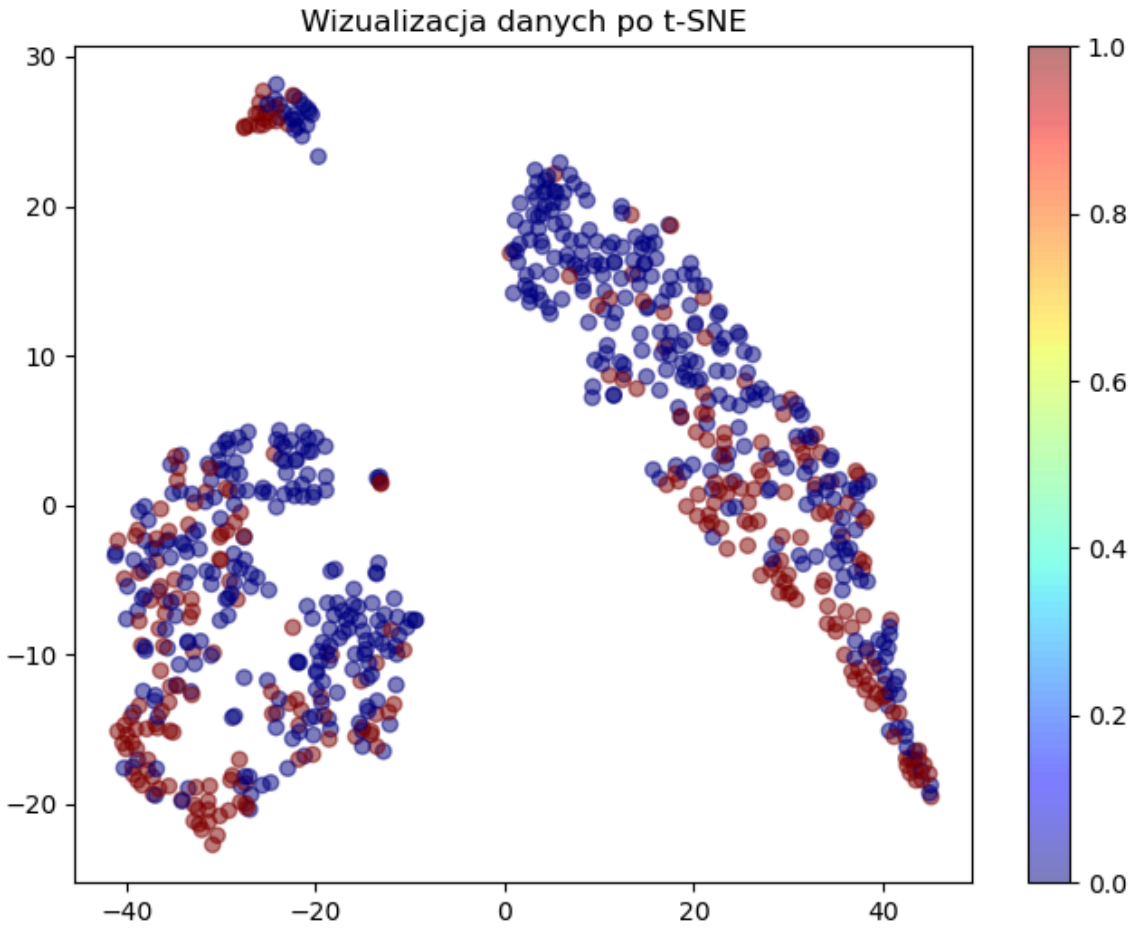
```
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)

# k-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# 6. Predykcja i ocena wyników
print("Regresja logistyczna:")
y_pred = log_reg.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

print("\nSVM:")
y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

print("\nk-Nearest Neighbors:")
y_pred = knn.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# 7. Redukcja wymiarowości (PCA)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

```
# Wizualizacja wyników PCA
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='jet', alpha=0.5)
plt.colorbar()
plt.title("Wizualizacja danych po PCA")
plt.show()

# 8. Wizualizacja wyników t-SNE
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)

# Wizualizacja wyników t-SNE
plt.figure(figsize=(8, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y, cmap='jet', alpha=0.5)
plt.colorbar()
plt.title("Wizualizacja danych po t-SNE")
plt.show()
```

```
Regresja logistyczna:
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154

[[79 20]
 [18 37]]

SVM:
              precision    recall  f1-score   support

           0       0.81      0.82      0.81        99
           1       0.67      0.65      0.66        55

    accuracy                           0.76       154
   macro avg       0.74      0.74      0.74       154
weighted avg       0.76      0.76      0.76       154

[[81 18]
 [19 36]]

k-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.75      0.80      0.77        99
           1       0.58      0.51      0.54        55

    accuracy                           0.69       154
   macro avg       0.66      0.65      0.66       154
weighted avg       0.69      0.69      0.69       154

[[79 20]
 [27 28]]
```



Wykres przedstawia dane po redukcji wymiarowości do dwóch głównych komponentów za pomocą metody PCA (Principal Component Analysis). Każdy punkt na wykresie reprezentuje jeden przykład w zbiorze danych, a kolor punktu wskazuje jego przynależność do jednej z dwóch klas (0 lub 1) — w tym przypadku, wynik testu na cukrzycę (Outcome). Celem tej wizualizacji jest przedstawienie, jak dane rozkładają się w przestrzeni 2D po redukcji wymiarów, co może pomóc w ocenie, czy klasy są dobrze separowane.



Na wykresie przedstawiono dane po zastosowaniu techniki t-SNE (t-Distributed Stochastic Neighbor Embedding), która również redukuje wymiarowość danych, ale w sposób bardziej ukierunkowany na zachowanie lokalnych zależności między przykładami. W podobny sposób jak na wykresie PCA, kolor punktu oznacza klasę (0 lub 1) przypisaną do danego przykładu. t-SNE jest szczególnie użyteczne w przypadku skomplikowanych danych, ponieważ zachowuje bardziej szczegółową strukturę lokalną i może lepiej pokazać skupiska i różnice między klasami.

3. Wnioski

Celem tego ćwiczenia było zapoznanie się z metodami klasyfikacji danych przy użyciu różnych algorytmów uczenia maszynowego. W ramach zadania wykorzystano dane dotyczące cukrzycy, a konkretne modele – regresja logistyczna, maszyny wektorów nośnych (SVM) oraz k-Nearest Neighbors (k-NN) – zostały zastosowane w celu klasyfikacji wyników testów na cukrzycę. Celem było porównanie wydajności tych algorytmów w zakresie dokładności klasyfikacji, precyzji, czułości i F1-score dla dwóch klas: osób z cukrzycą i bez cukrzycy.

Wyniki eksperymentu pokazały, że regresja logistyczna i maszyny wektorów nośnych osiągnęły podobną dokładność, z nieznaczną przewagą SVM, który lepiej radził sobie w rozróżnianiu przypadków negatywnych (klasa 0). Z kolei model k-Nearest Neighbors okazał się najmniej skuteczny, szczególnie w wykrywaniu przypadków pozytywnych (cukrzyca), co wskazuje na potrzebę dalszej optymalizacji tego algorytmu w tego typu zadaniach. Choć SVM i regresja logistyczna wykazały się lepszymi wynikami, to wciąż pozostaje przestrzeń do poprawy, szczególnie w zakresie wykrywania osób z cukrzycą.

Porównując te algorytmy, można stwierdzić, że nie ma jednoznacznie najlepszego modelu dla każdego przypadku. W zależności od priorytetów – np. jeśli zależy nam na wykryciu jak największej liczby przypadków cukrzycy – warto przeanalizować skuteczność różnych algorytmów i rozważyć dalsze dostosowanie ich parametrów. Eksperymenty takie jak ten pomagają zrozumieć, jak różne algorytmy reagują na dane i jakie mają ograniczenia, co stanowi cenną wiedzę przy wyborze metod klasyfikacji w realnych problemach.