Politechnika Białostocka

Sprawozdanie z pracowni specjalistycznej

Sztuczna inteligencja

Ćwiczenie numer: 5-6

Temat: **Drzewa decyzyjne w uczeniu maszynowym**

Wykonujący ćwiczenie: **Jarosław Klepadło**

Adrian Choroszewicz

Michał Piotr Busiński

Konrad Łupiński

Studia stacjonarne

Kierunek: Informatyka

Semestr: IV

Grupa zajęciowa: 8

Prowadzący ćwiczenie: mgr inż. Dariusz Jankowski

Data wykonania ćwiczenia: 24.03.2025

1. **Treści zadań:**
    1. **Podaj dwa przykłady tablic decyzyjnych takich, że zastosowanie wskaźnika A powoduje zbudowanie innego drzewa decyzyjnego niż zastosowanie heurystyki opartej na wskaźniku B. W pierwszym przykładzie heurystyka oparta na wskaźniku A powinna dawać lepsze rezultaty według kryterium C niż zastosowanie wskaźnika B, a w drugim przykładzie powinno być odwrotnie.**

       Wariant 1: A = Gini, B = Gain Ratio, C = mała liczba węzłów

    2. **Przeprowadź klasyfikację danych podanych przez prowadzącego zajęcia oraz zbadaj wpływ parametrów (wybranych przez prowadzącego zajęcia) na jakość klasyfikacji.**

2. Wyniki

Zadanie 1

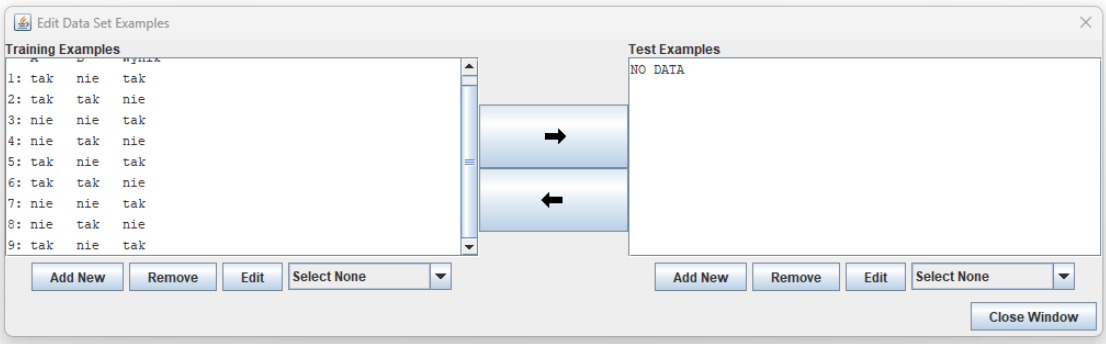
Podaj dwa przykłady tablic decyzyjnych takich, że zastosowanie wskaźnika A powoduje zbudowanie innego drzewa decyzyjnego niż zastosowanie heurystyki opartej na wskaźniku B. W pierwszym przykładzie heurystyka oparta na wskaźniku A powinna dawać lepsze rezultaty według kryterium C niż zastosowanie wskaźnika B, a w drugim przykładzie powinno być odwrotnie.

Wariant 1: A = Gini, B = Gain Ratio, C = mała liczba węzłów

1. Przygotowanie danych

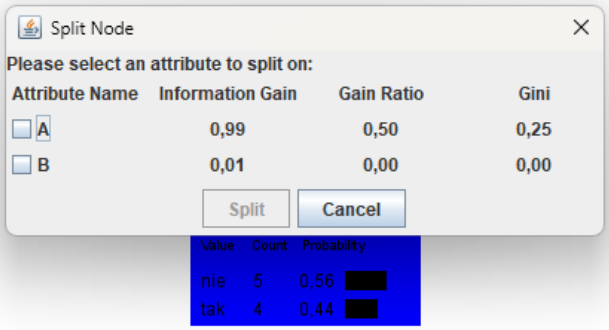
Stworzono dwie proste tablice decyzyjne danych do zadania. Do sprawdzenia róznych rodzajów heurystyk.

2. Przykład 1:



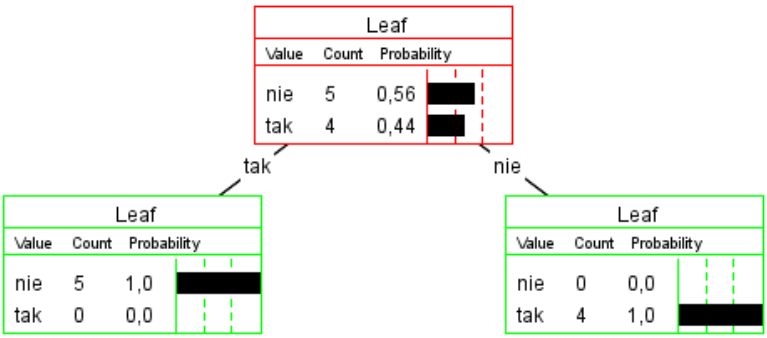
Rys 1. Dane w przykładzie pierwszym

Mając wprowadzone dane należy wyliczyć współczynnik *Gini* oraz *Gain Ration* dla podanych danych. Program dtree łatwo nam to wylicza.



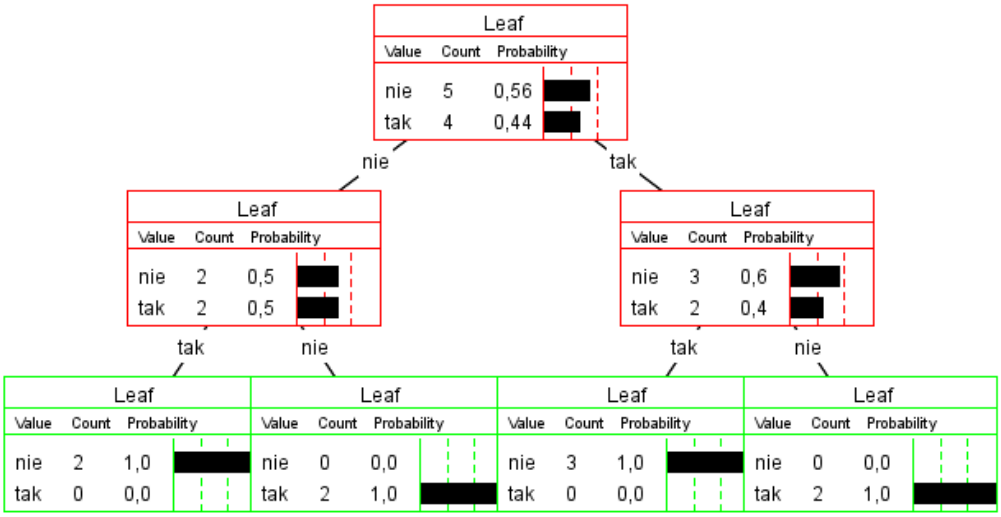
Rys 2. Współczynniki dla przykładu pierwszego

Wybrano potem opcję podziału poprzez Gini i symulujemy nasze wyniki.



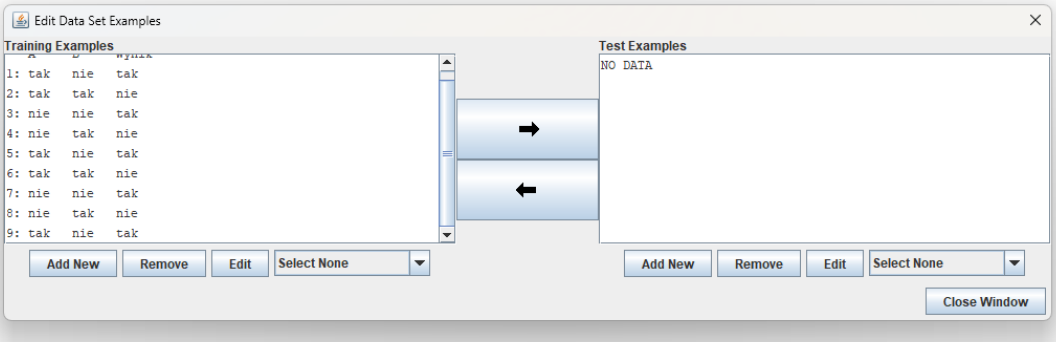
Rys 3. Tablica decyzyjna podziału gini dla pierwszego przykładu

Następnie wybrano podział poprzez gain ratio.



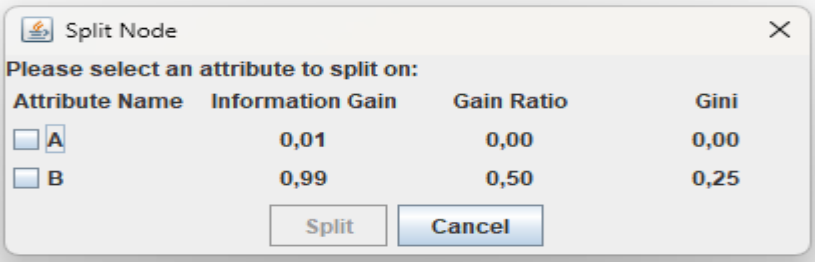
Rys 4. Tablica decyzyjna podziału gain ratio dla pierwszego przykładu

3. Przykład 2:



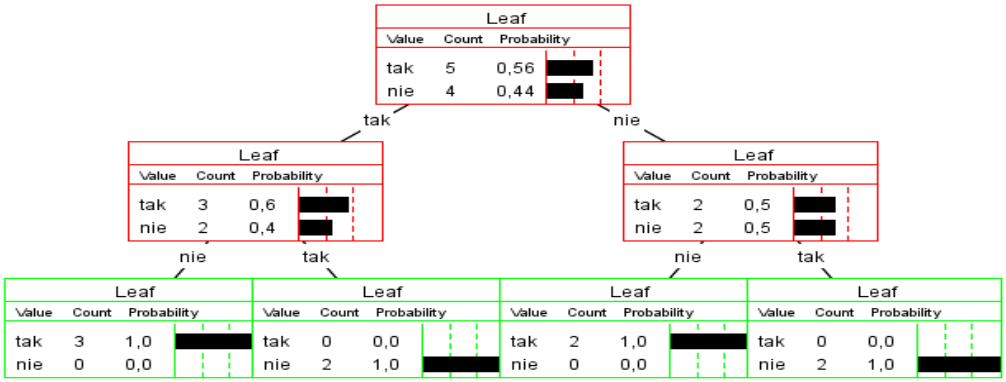
Rys 5. Dane dla przykładu drugiego

Mając wprowadzone dane należy wyliczyć współczynnik *Gini* oraz *Gain Ration* dla podanych danych. Program dtree łatwo nam to wylicza.



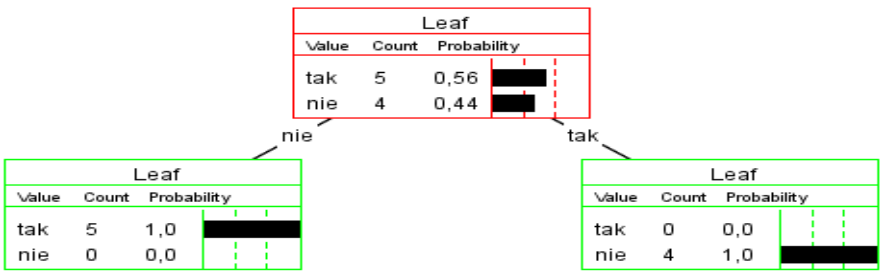
Rys 6. Współczynniki dla przykładu drugiego

Wybrano potem opcję podziału poprzez Gini i symulujemy nasze wyniki.



Rys 7. Tablica decyzyjna podziału gini dla drugiego przykładu

Następnie wybrano podział poprzez gain ratio.



Rys 8. Tablica decyzyjna podziału gain ratio dla drugiego przykładu

Zadanie 2

Przeprowadź klasyfikację danych podanych przez prowadzącego zajęcia oraz zbadaj wpływ parametrów (wybranych przez prowadzącego zajęcia) na jakość klasyfikacji.

1. Instalacja i import bibliotek

pandas i numpy do pracy z danymi,

seaborn i matplotlib do wizualizacji,

scikit-learn do uczenia maszynowego,

kagglehub do pobierania danych z Kaggle.

2. Wczytanie i podgląd danych

import kagglehub

path = kagglehub.dataset_download("yasserh/titanic-dataset")

df = pd.read_csv(path + "/Titanic-Dataset.csv")

df.head()

Tutaj pobierany jest zestaw danych o pasażerach Titanica bezpośrednio z Kaggle przy użyciu kagglehub. Po pobraniu dane są wczytywane do ramki DataFrame z wykorzystaniem pandas.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Rys 9. Pierwsze 5 wierszy Titanic Dataset

df.head() wyświetla pierwsze 5 wierszy, umożliwiając szybki podgląd struktury danych. Znajdują się tam kolumny takie jak Name, Sex, Age, Survived.

Z danych wynika, że przykładowo:

- pasażer z ID 1 (Braund, Mr. Owen Harris) nie przeżył,

- pasażerka z ID 2 (Cumings, Mrs. John Bradley) przeżyła.

3. Kodowanie zmiennych tekstowych

```
encoder = LabelEncoder()

for column in df.columns:
    df[column] = encoder.fit_transform(df[column])
```

Wszystkie kolumny są kodowane do wartości liczbowych.

To ważne, ponieważ modele uczące się nie radzą sobie ze stringami (np. male czy female), więc zamieniane są one na liczby.

LabelEncoder przypisuje kolejne liczby całkowite do unikalnych wartości w kolumnach.

Przykładowo, male może zostać zakodowany jako 1, a female jako 0.

4. Przygotowanie danych do trenowania

```
X = df.drop(columns=["Survived","Cabin","Name","PassengerId","Ticket","Fare","Embarked"])

y = df["Survived"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

X zawiera dane wejściowe (cechy) – bez zbędnych kolumn, takich jak nazwiska czy numery biletów.

y zawiera etykietę docelową, czyli informację, czy dana osoba przeżyła (1) czy nie (0).

Dane dzielone są na zbiór treningowy i testowy w proporcji 70/30.

```
X_train
```

	Pclass	Sex	Age	SibSp	Parch
748	0	1	24	1	0
45	2	1	88	0	0
28	2	0	88	0	0
633	0	1	88	0	0
403	2	1	36	1	0
...	...	...	...	...	...
476	1	1	45	1	0
190	1	0	42	0	0
736	2	0	63	1	3
462	0	1	62	0	0
136	0	0	24	0	2

623 rows × 5 columns

Rys 10. Zbiór treningowy

X_test

	Pclass	Sex	Age	SibSp	Parch
625	0	1	77	0	0
566	2	1	24	0	0
459	2	1	88	0	0
804	2	1	35	0	0
338	2	1	59	0	0
...	...	...	...	...	...
184	2	0	8	0	2
607	0	1	35	0	0
624	2	1	27	0	0
103	2	1	44	0	0
387	1	0	48	0	0

268 rows × 5 columns

Rys 11. Zbiór testowy

Ten podział jest kluczowy, aby móc sprawdzić później, jak dobrze model radzi sobie na danych, których nie widział wcześniej.

5. Sprawdzenie i wypełnienie brakujących danych

X.isna().values.any()

wynik: False

To sprawdzenie mówi, czy są jakieś braki (NaN) w zbiorze cech.

Wynik False sugeruje, że po kodowaniu tekstowym nie występują żadne braki danych.

Zastosowano także:

imp = SimpleImputer(missing_values=np.nan, strategy='mean')

imp.fit(X_train)

X_train = imp.transform(X_train)

X_test = imp.transform(X_test)

Na wszelki wypadek stosuje się wypełnianie braków średnią wartością w kolumnie.

To dobra praktyka, szczególnie gdy dane mają wartości NaN.

6. Trenowanie drzewa decyzyjnego

model = DecisionTreeClassifier(criterion='gini')

model.fit(X_train, y_train)

Tworzony jest model drzewa decyzyjnego (DecisionTreeClassifier) z domyślnym kryterium podziału gini.

Model trenowany jest na zbiorze treningowym X_train i y_train.

Drzewo decyzyjne to model, który podejmuje decyzje poprzez ciąg warunków logicznych (np. Sex <= 0.5).

7. Wizualizacja drzewa

plt.figure(figsize=(12,8))

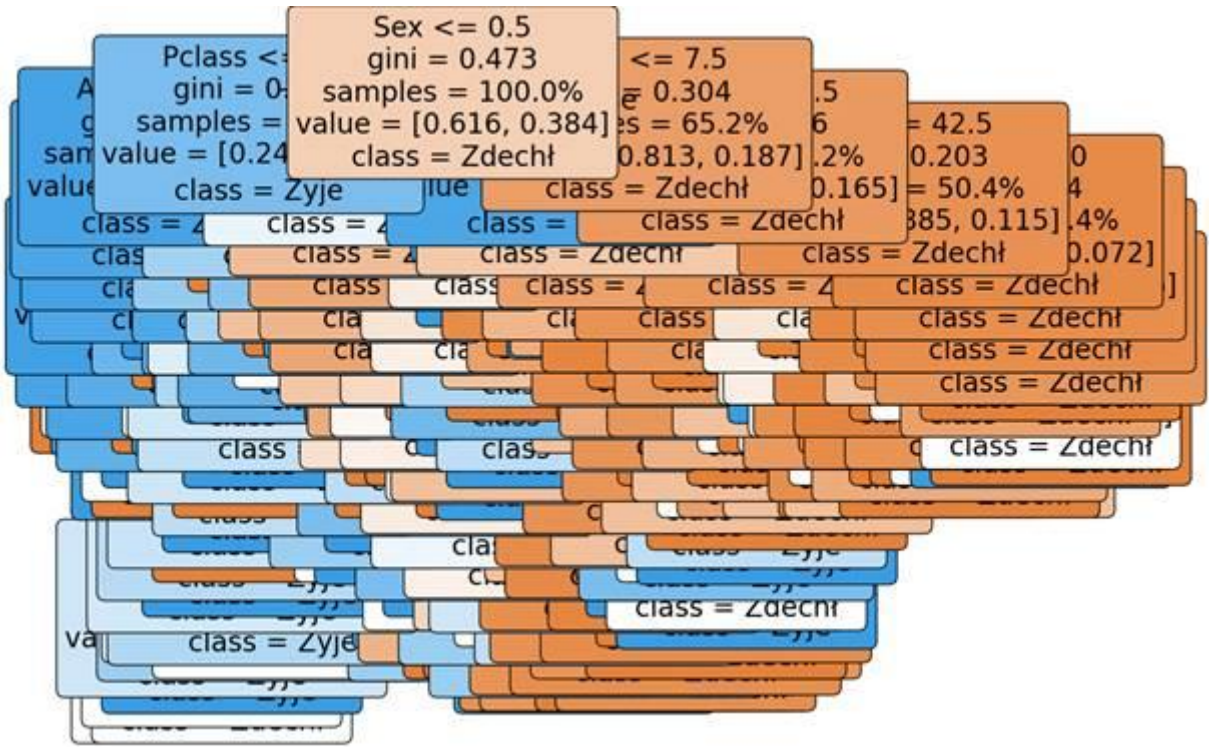
plot_tree(model, feature_names=X.columns, class_names=["Zdechł", "Żyje"], filled=True)

plt.title("Struktura Drzewa Decyzyjnego")

plt.show()

Model zostaje wizualnie przedstawiony jako drzewo, w którym widać, jakie cechy mają największy wpływ na decyzję (np. płeć, wiek, klasa).

Każdy węzeł pokazuje warunek, a liście końcowe wskazują, do jakiej klasy przypisywana jest osoba (żyje/zdechł).



Rys 12. Struktura Drzewa Decyzyjnego

To bardzo pomocne, by zrozumieć, jak działa model.

Opcja filled=True koloruje węzły w zależności od dominującej klasy.

8. Klasyfikacja danych testowych i dokładność modelu

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Dokładność modelu: {accuracy:.2f}")

Wynik: Dokładność modelu: 0.79

Model przewiduje wyniki dla danych testowych, a następnie obliczana jest dokładność.

Wynik 0.79 oznacza, że model poprawnie przewidział los pasażerów w 79% przypadków.

Dokładność to miara globalna, nie mówi nam nic o konkretnych błędach modelu.

9. Macierz pomyłek

conf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["Zdechł", "Żyje"], yticklabels=["Zdechł", "Żyje"])

plt.xlabel("Przewidywane")

plt.ylabel("Rzeczywiste")

plt.title("Macierz Pomyłek")

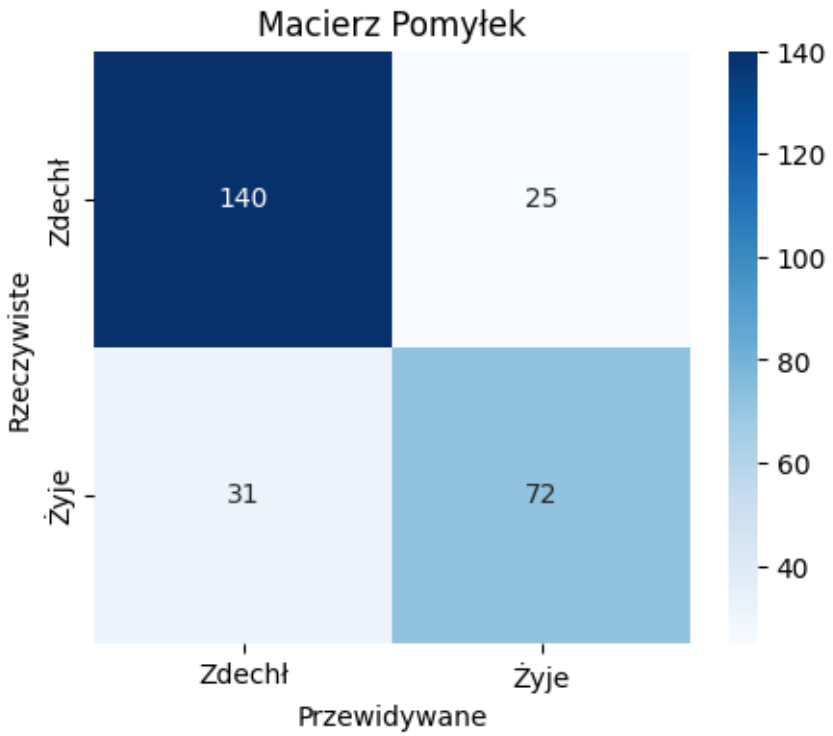
plt.show()

Macierz pomyłek, przy użyciu naszych danych, pokazuje:

- ile osób przewidziano poprawnie jako zmarłych,

- ile przeżyło, ale zostało błędnie sklasyfikowanych,

- oraz odwrotnie.



Rys 13. Macierz pomyłek

To bardzo pomocne narzędzie, by lepiej zrozumieć, gdzie model się myli.

Na przykład może się okazać, że model dobrze rozpoznaje osoby, które zginęły, ale gorzej radzi sobie z przewidywaniem tych, którzy przeżyli.

3. Wnioski

Zadanie 1

Przedstawione przykłady obrazują, że wybór wskaźnika heurystycznego ma istotny wpływ na strukturę i rozmiar drzewa decyzyjnego. Wskaźnik Gini często preferuje szybkie, silne podziały o niskim stopniu skomplikowania, co sprzyja małej liczbie węzłów. Z kolei Gain Ratio jest bardziej wyważony i uwzględnia różnorodność atrybutów, co w niektórych przypadkach prowadzi do bardziej kompaktowych drzew. Dobór wskaźnika należy więc uzależnić od charakteru danych i priorytetów modelu, minimalizacja liczby węzłów może być osiągnięta różnymi heurystykami w zależności od konkretnej sytuacji.

Zadanie 2

Analiza danych pasażerów Titanica wykazała, że odpowiednie przygotowanie danych, w tym kodowanie zmiennych kategorycznych i usunięcie zbędnych kolumn, miało kluczowe znaczenie dla skuteczności modelu. Zastosowanie drzewa decyzyjnego umożliwiło nie tylko uzyskanie dokładności 79%, ale także przejrzystą interpretację, które cechy najbardziej wpływały na przeżycie. Pomimo braku braków w danych, użycie wypełnienia wartości średnich zwiększyło stabilność modelu. Macierz pomyłek pokazała, że model dobrze przewiduje zgony, choć nieco słabiej radzi sobie z rozpoznawaniem osób, które przeżyły. Wnioski te potwierdzają, że nawet prosty model może być skuteczny przy dobrze przemyślanej analizie danych.