

Politechnika Warszawska
*Wydział Elektroniki i Technik
Informacyjnych*

Wstęp do Sztucznej Inteligencji
ćwiczenie 3 - Dwuosobowe gry deterministyczne

Autor:

Jarosław Jaworski 342189

Data oddania ćwiczenia:

26.04.2025

SPIS TREŚCI

1	Cel ćwiczenia	2
2	Przygotowanie zbioru danych	2
3	Ocena modelu	3
4	Wnioski	7

1 Cel ćwiczenia

Celem ćwiczenia było zaimplementowanie algorytmu lasu losowego do predykcji wystąpienia choroby serca u przykładowego pacjenta, bazując na jego cechach: płeć, typ bólu w klatce piersiowej, poziom cukru we krwi, itd.

2 Przygotowanie zbioru danych

Pierwszym krokiem przy uczeniu modeli sztucznej inteligencji jest przygotowanie tzw. *"setów"*, czyli podzestawów danych zestawu źródłowego. W tym wypadku, plik lab-4-dataset zawierał łącznie ponad 900 rekordów z danymi, które podzielono na zestaw uczący (*training_set*) oraz testujący (*test_set*). W tym celu zastosowano funkcję *train_test_split*, wchodzącą w skład biblioteki **scikit learn**. Dzięki temu dane podzielono w stosunku 4:1 (*training:test*). Dane umieszczone w kolumnach podzielono na zbiór klas wyjściowych (tu: pojedyncza kolumna z informacją binarną o występowaniu choroby) oraz zbiór atrybutów wejściowych (pozostałe kolumny, będące cechami). Ze względu na fakt, że modele AI nie mogą przyjmować napisów (*labeli*), użyto biblioteki **pandas** aby nadać każdej z występujących wartości w kategoriach odpowiednią wartość numeryczną. Całość implementacji przedstawia Wycinek kodu 1.

```
1 def load_and_prepare_data(filepath : str) -> list:
2     # Szczytanie danych z pliku
3     df = pd.read_csv(filepath)
4     # Podział na podzbiory klas i atrybutow wejsciowych
5     X = df.drop('HeartDisease', axis=1)
6     y = df['HeartDisease']
7     # Utworzenie kolumn binarnych dla kazdego labela
      kazdej kategorii atrybutow wejsciowych
```

```

8     X = pd.get_dummies(X, drop_first=True)
9     # Standardyzacja danych - nadanie sredniej i
        odchylenia standardowego dla kazdej kolumny w X
10    scaler = StandardScaler()
11    X_scaled = scaler.fit_transform(X)
12    # Zwrot listy danych podzielonych na zestaw
        treningowy i testujacy
13    return train_test_split(X_scaled, y, test_size=0.2,
        random_state=42)

```

Wycinek kodu 1: Przygotowanie danych do zadania klasyfikacji

3 Ocena modelu

Zastosowano wbudowany model biblioteki scikit-learn *RandomForestClassifier* dla zbioru danych trenujących. Dzięki temu, możliwa była prosta ocena parametrów, korzystając z interfejsu klasy, czyli jej wbudowanych metod. Wykorzystano następujące metryki klasyfikacji:

- **Dokładność** (accuracy), definiowaną jako udział poprawnych przewidywań w całym zbiorze testowym, obliczana na podstawie zależności (1)

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

,gdzie: TP, TN, FP, FN - TRUE lub FALSE + POSITIVE lub NEGATIVE

- **Precyzja**, czyli miara prawdopodobieństwa poprawnego przewidywania, obliczana na podstawie zależności (2)

$$precision = \frac{TP}{TP + FP} \quad (2)$$

- **Czułość** (recall), czyli ile faktycznie chorych osób zostało poprawnie wykrytych, obliczana na podstawie zależności (3)

$$recall = \frac{TP}{TP + FN} \quad (3)$$

- **AUC** - pole pod krzywą ROC, obliczane jako pole pod wykresem krzywej ROC (zależności czułości od wskaźnika fałszywych pozytywów).

Wartości metryk wyznaczono dla różnych wielkości zestawu trenującego, w celu oceny jakości przewidywań. Wyniki przedstawia Tabela 1.

Tabela 1. Wyniki pomiarów dla różnej wielkości zestawu trenującego

Wielkość zestawu trenującego [%]	Dokładność [%]	Precyzja [%]	Czułość [%]	AUC
90	85,87	85,96	90,74	0,9418
80	89,13	90,65	90,65	0,9342
70	87,68	90,62	88,41	0,941
60	86,41	91,3	85,52	93,99
50	87,8	90	88,6	0,938
40	87,48	89,2	88,42	0,9377

Dla wybranego rozmiaru zestawu treningowego, stanowiącego 80% całego zbioru danych, przeprowadzono próby optymalizacji modelu lasu losowego poprzez ustalenie liczby drzew jako hiperparametru. Decyzję tę podjęto na podstawie najlepszych uzyskanych wartości metryk oceny modelu.

Wykorzystując klasę *GridSearchCV* zaimplementowano funkcję, badającą najlepszy zestaw hiperparametrów dla algorytmu lasu losowego. Przedstawiono to w Wycinku kodu 2.

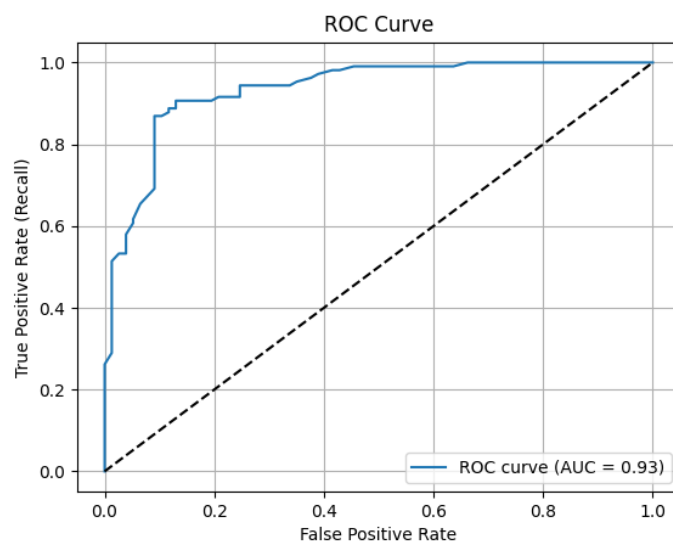
```

1 def optimize_hyperparameters(X_train, y_train):
2     # Utworzenie listy hiperparametrow
3     param_grid = {'n_estimators': [10, 50, 100, 200,
4         300, 400, 500]}
5     # Utworzenie automatycznego testera modeli, który
6         buduje las o hiperparametrze z listy param_grid i
7         ocenia na podstawie czulosci modelu
8     grid_search =
9         GridSearchCV(RandomForestClassifier(random_state=42),
10             param_grid, cv=5, scoring='recall')
11     # Przeprowadzenie wszystkich prob
12     grid_search.fit(X_train, y_train)
13     # Wyświetlenie najlepszego znalezionej modelu i
14         jego zwrocenie
15     print(f"\nNajlepsze parametry:
16         {grid_search.best_params_}")
17     return grid_search.best_estimator_

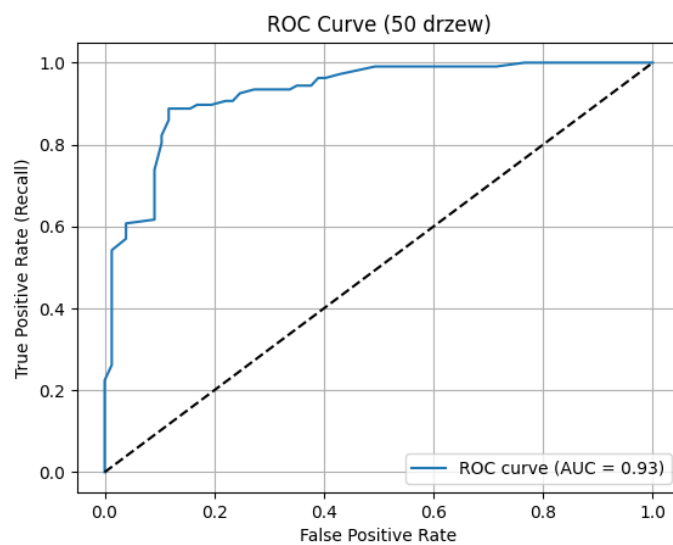
```

Wycinek kodu 2: Próba optymalizacji przez narzucenie hiperparametru - liczby drzew

Wynikiem testu było znalezienie najlepszej ilości drzew dla uzyskania najlepszego co do wartości wyniku metryki czułości, wynoszącej 100. Dla tego wyniku wykreślono krzywą ROC, przedstawioną na Rys. 1. Dla przykładu zaprezentowano również inny wykres ROC dla nieidealnego modelu na Rys. 2.



Rys. 1. Wykres ROC dla najlepszego modelu, $n = 100$



Rys. 2. Wykres ROC dla innego modelu, $n = 50$

4 Wnioski

Algorytm lasu losowego był czuły na zmiany w wielkości zestawu trenującego. Jego dokładność była co do wartości podobna, z najlepszymi wynikami pozostałych metryk dla wielkości zestawu - 80%.

Wpływ na jakość działania algorytmu ma zastosowanie hiperparametru w postaci narzucenia liczby drzew. Na podstawie dokumentacji klasy *RandomForestClassifier* wiadomo, że domyślnie liczba ta wynosi 100. Testy przeprowadzone dla kolenych liczb drzew wykazały, że była to optymalna ilość, jeżeli oceniamy modele przez pryzmat ich czułości.

Wybór kryterium oceny modeli nie był przypadkowy. Czułość z punktu widzenia medycyny powinna być najistotniejszym parametrem do maksymalizacji w modelu klasyfikującym. Jest tak, ponieważ daje ona informację ile spośród osób chorych zostało wykrytych w procesie predykcji. Tym samym uzyskuje się najlepsze pojęcie o szansie na poprawne identyfikowanie chorego przez model.