SPRAWOZDANIE

Zajęcia: Uczenie Maszynowe
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 2 Data 09.11.2024 Temat: Praktyczne Zastosowanie Drzew Decyzyjnych i Metod Ensemble w Analizie Danych Wariant drugi (2)	Bartosz Bieniek Informatyka II stopień, stacjonarne, 1 semestr, gr.A

1. Polecenie: Wariant drugi zadania

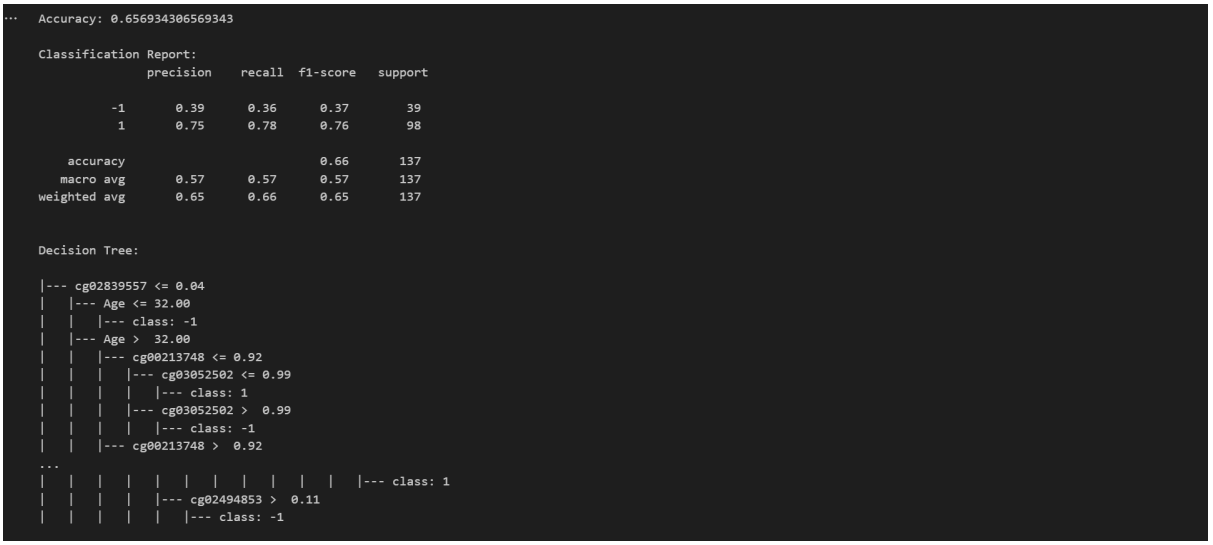
Opracować przepływ pracy uczenia maszynowego zagadnienia klasyfikacji (pojedyncze drzewo decyzyjne) oraz klasyfikacji ensemble (używając wszystkie modele) na podstawie zbioru danych według drugiego wariantu zadania.

2. Opis programu opracowanego Kod źródłowy Github

```python
# 1.Opracować przepływ pracy uczenia maszynowego zagadnienia klasyfikacji (pojedyncze drzewo decyzyjne)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.metrics import classification_report, accuracy_score

data = pd.read_csv('smokers.csv')
data.dropna()

data['Smoking Status'] = data['Smoking Status'].map({'current': 1, 'former': 0, 'never': -1})
data['Gender'] = data['Gender'].map({'f': 0, 'm': 1})

X = data.drop(columns=['GSM', 'Smoking Status'])
y = data['Smoking Status']

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

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nDecision Tree:\n")
print(export_text(clf, feature_names=list(X.columns)))
```

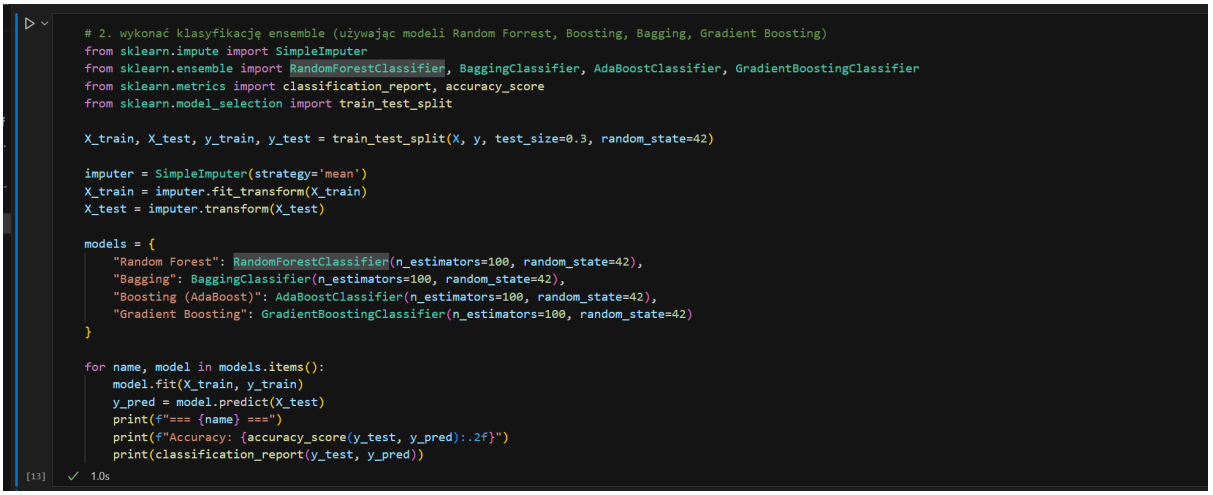
Rys. 1. Fragment programu.

Wyżej wymieniony fragment kodu opisuje przepływ uczenia maszynowego uzgadniania klasyfikacji w pojedynczym drzewie decyzyjnym. Przedstawiony program ładuje dane z pliku smokers.csv, następnie przekonwertowano wartości mnemoniczne na liczbowe dla kolumn Smoking Status oraz Gender. Wybrano odpowiednie cechy i etykiety a następnie podzielono dane na zbiory treningowe i testowe. W ostatnim kroku przed wyświetleniem wyników, utworzono i dostosowano model drzewa decyzyjnego z random_state=42 w celu powtarzalnych wyników.



Rys. 2. Wynik wykonania programu i fragment drzewa decyzyjnego.

Dokładność wyniku wynosi prawie 66%. Jego wyniki są znacznie lepsze dla klasy 1, gdzie precyzja i czułość są wysokie (odpowiednio 75% i 78%), niż dla klasy -1, gdzie oba wskaźniki wynoszą około 36-39%. Model wykorzystuje wartości cech, takich jak cg02839557 i wiek, do podejmowania decyzji. Jednak słaba skuteczność dla klasy -1 sugeruje potrzebę poprawy, np. poprzez zbalansowanie danych lub optymalizację cech.



```python
# 2. wykonać klasyfikację ensemble (używając modeli Random Forrest, Boosting, Bagging, Gradient Boosting)
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split

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

imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Bagging": BaggingClassifier(n_estimators=100, random_state=42),
    "Boosting (AdaBoost)": AdaBoostClassifier(n_estimators=100, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(n_estimators=100, random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"=== {name} ===")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(classification_report(y_test, y_pred))
```

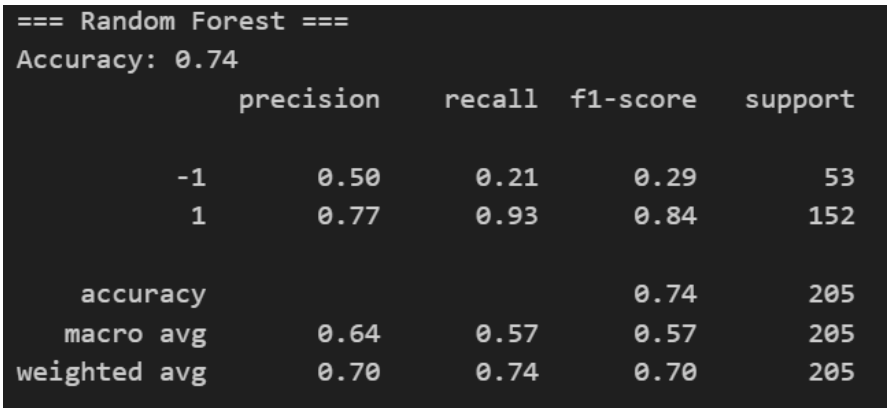
[13] ✓ 1.0s

Rys. 3. Fragment programu.

Przedstawiony program na rysunku trzecim przeprowadza klasyfikację za pomocą różnych algorytmów typu ensemble: Random Forest, Bagging, AdaBoost (Boosting) i Gradient Boosting. Na początku dane są podzielone na zbiory treningowe i testowe, a następnie brakujące wartości w cechach są uzupełniane za pomocą strategii średniej arytmetycznej klasy SimpleImputer. Każdy model z listy jest trenowany na przetworzonym zbiorze treningowym, a jego skuteczność jest oceniana na zbiorze testowym. Iterując po modelach program oblicza i wyświetla dokładność (Accuracy) oraz szczegółowe miary

jakości klasyfikacji, takie jak precyzja, czułość i F1-score. Zezwala to na porównanie, który z zastosowanych algorytmów najlepiej sprawdza się dla danego zestawu danych.

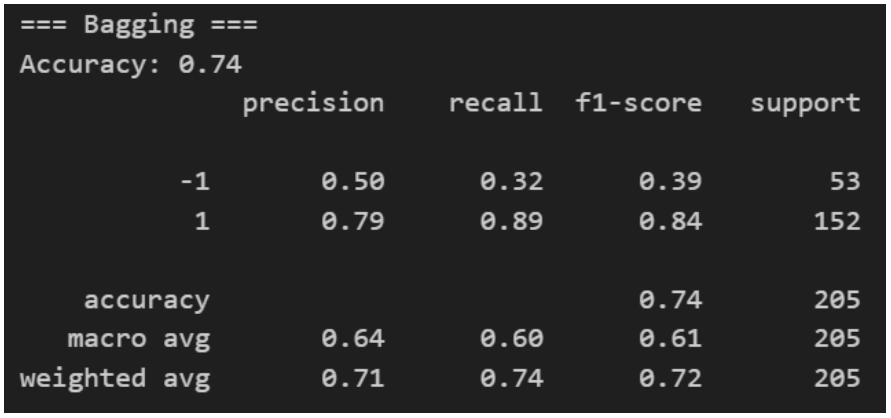
Analiza wyników:



```
=== Random Forest ===
Accuracy: 0.74
              precision    recall  f1-score   support

          -1       0.50      0.21      0.29        53
           1       0.77      0.93      0.84       152

    accuracy                           0.74       205
   macro avg       0.64      0.57      0.57       205
weighted avg       0.70      0.74      0.70       205
```

Rys. 4. Wyniki wykonania dla Random Forrest.

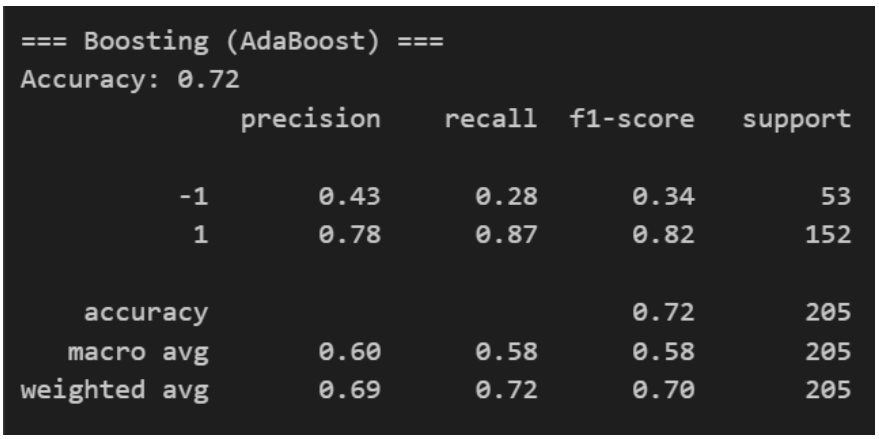
Wyniki dla modelu Random Forest wskazywały na dokładność na poziomie 74%. Klasa 1 została przewidziana z wysoką skutecznością, osiągając f1-score równy 0.84, jednak klasa -1 charakteryzowała się znacznie niższymi wartościami precyzji (0.50) i f1-score (0.29). Średnie wartości makro sugerowały nierównomierną skuteczność klasyfikacji między klasami.



```
=== Bagging ===
Accuracy: 0.74
              precision    recall  f1-score   support

          -1       0.50      0.32      0.39        53
           1       0.79      0.89      0.84       152

    accuracy                           0.74       205
   macro avg       0.64      0.60      0.61       205
weighted avg       0.71      0.74      0.72       205
```

Rys. 5. Wyniki wykonania dla modelu Bagging.

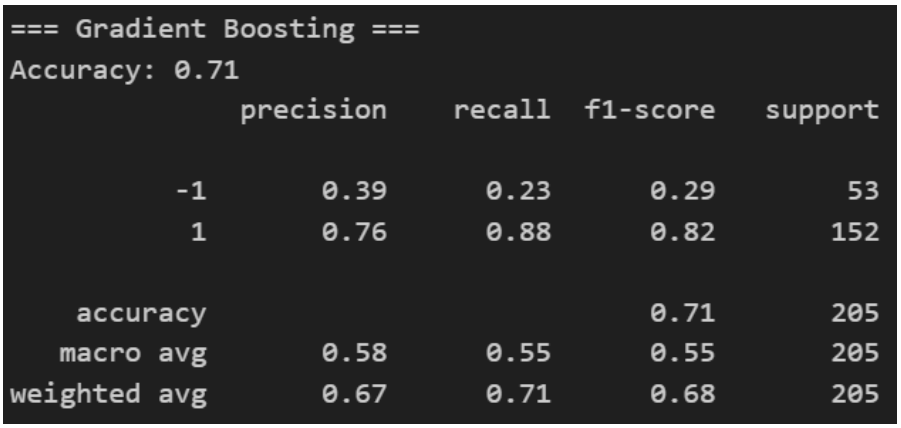
Model Bagging po wykonaniu operacji uzyskał dokładność na poziomie 0.74, tak samo jak Random Forest. Dobrze radzi sobie z klasyfikacją klasy 1, osiągając wysoką recall (89%) i precyzję (79%). W przypadku klasy -1 recall wynosi 32%, co jest lepszym wynikiem niż w przypadku Random Forest, ale wciąż pozostaje stosunkowo niska. Obydwa modele mają trudności z wykrywaniem tej klasy.



```
=== Boosting (AdaBoost) ===
Accuracy: 0.72
              precision    recall  f1-score   support

          -1       0.43      0.28      0.34        53
           1       0.78      0.87      0.82       152

    accuracy                           0.72       205
   macro avg       0.60      0.58      0.58       205
weighted avg       0.69      0.72      0.70       205
```

Rys. 6. Wyniki wykonania dla modelu Boosting.

Model AdaBoost osiągnął dokładność 0.72, co jest nieco słabszym wynikiem niż Bagging i Random Forest. Dobrze radzi sobie z klasyfikacją klasy 1, osiągając wysoką recall (87%) i precyzję (78%). Jednak w przypadku klasy -1, recall wynosi tylko 28%, co jest jednym z najniższych wyników spośród analizowanych modeli.



```
=== Gradient Boosting ===
Accuracy: 0.71
              precision    recall  f1-score   support

          -1       0.39      0.23      0.29        53
           1       0.76      0.88      0.82       152

    accuracy                           0.71       205
   macro avg       0.58      0.55      0.55       205
weighted avg       0.67      0.71      0.68       205
```

Rys. 6. Wyniki wykonania dla modelu Gradient Boosting.

Dokładność modelu Gradient Boosting to 0.71, co jest nieco słabszym wynikiem niż Bagging, Random Forest i AdaBoost. Podobnie jak poprzednie modele, dobrze radzi sobie z klasyfikacją klasy pierwszej, osiągając recall na poziomie 88% i precyzję 76%. W przypadku klasy -1, recall wynosi tylko 23%, co pokazuje trudności modelu w wykrywaniu tej klasy.

Model	Klasa -1 Recall	Klasa -1 Precision	Klasa -1 F1-score	Klasa 1 Recall	Klasa 1 Precision	Klasa 1 F1-score	Accuracy
Random Forest	21%	50%	29%	93%	77%	84%	74%
Bagging	32%	50%	39%	89%	79%	84%	74%
AdaBoost	28%	43%	34%	87%	78%	82%	72%
Gradient Boost	23%	39%	29%	88%	76%	82%	71%

Rys. 6. Zestawienie wyników w tabeli.

Z tabeli przedstawionej na rysunku szóstym można wywnioskować, że wszystkie modele mają wyższą skuteczność w klasyfikacji klasy 1, osiągając wysoką recall i precyzję. Najlepszy wynik dla klasy 1 osiąga Random Forrest (93% recall, 77% precision). Natomiast w przypadku klasy -1, zauważalne jest, że wszystkie modele mają niską wartość recall, przy czym najlepiej spisuje się model Bagging z recall na poziomie 32% i precision równym 50%.

3. Wnioski

Modele ensemble, mimo dobrej klasyfikacji klasy 1, mają duże trudności z wykrywaniem rzadziej występującej klasy -1.

Bagging i Random Forest są nieco lepsze w detekcji klasy 1, ale żaden z modeli nie osiąga zadowalających wyników dla klasy -1, co może wskazywać na potrzebę dalszej optymalizacji lub zmiany strategii klasyfikacji.