AutoML projekt 2

Prototyp pakietu AutoML

Jan Kruszewski - Bartosz Maj Bartosz - Olszewski

Grupa docelowa

- Grupa docelowa narzędzia AutoML to analitycy danych, inżynierowie uczenia maszynowego oraz badacze, którzy chcą szybko prototypować modele uczenia maszynowego bez konieczności manualnego strojenia hiperparametrów. Narzędzie jest szczególnie przydatne dla osób pracujących z klasyfikacją.

Specyfikacja narzędzia

Algorytmy klasyfikacyjne

Narzędzie obsługuje szeroki zakres algorytmów klasyfikacyjnych, takich jak:

- Decision Trees: Algorytm bazowy do interpretowalnej klasyfikacji.
- Random Forest: Wykorzystuje las losowych drzew dla poprawy dokładności.
- Support Vector Machines (SVM): Wysoka skuteczność w problemach liniowych i nieliniowych.
- XGBoost: Gradient boosting dla wysokiej wydajności.

Metody optymalizacji

- Grid Search: Wyznaczanie optymalnych parametrów poprzez przeszukiwanie wszystkich kombinacji.
- Random Search: Losowe próbkowanie przestrzeni parametrów dla szybszych wyników.
- Bayesian Optimization (do dodania): Inteligentne próbkowanie oparte na wcześniejszych wynikach.

Optymalizacja za pomocą Optuna

```
def objective_optuna(trial):
    n_estimators = trial.suggest_int("n_estimators", 50, 500)
    max_depth = trial.suggest_int("max_depth", 5, 50)
    min_samples_split = trial.suggest_int("min_samples_split", 2, 20)
    min_samples_leaf = trial.suggest_int("min_samples_leaf", 1, 10)

    model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split,
        min_samples_leaf=min_samples_leaf,
        random_state=42
    )

    score = cross_val_score(model, X_train, y_train, cv=3, scoring="accuracy").mean()
    return -score

study = optuna.create_study(direction="minimize")
study.optimize(objective_optuna, n_trials=50)

print("Najlepsze parametry Optuna:", study.best_params)
print("Najlepszy wynik Optuna:", -study.best_value)
```

Optymalizacja za pomocą Hyperopt

```
def objective_hyperopt(params):
    model = RandomForestClassifier(
        n_estimators=int(params['n_estimators']),
        max_depth=int(params['max_depth']),
        min_samples_split=int(params['min_samples_split']),
        min_samples_leaf=int(params['min_samples_leaf']),
        random_state=42
    )

    score = cross_val_score(model, X_train, y_train, cv=3, scoring="accuracy").mean()
    return -score

space = {
    'n_estimators': hp.quniform('n_estimators', 50, 500, 10),
    'max_depth': hp.quniform('max_depth', 5, 50, 1),
    'min_samples_split': hp.quniform('min_samples_split', 2, 20, 1),
    'min_samples_leaf': hp.quniform('min_samples_leaf', 1, 10, 1),
}

trials = Trials()
best_hyperopt = fmin(fn=objective_hyperopt, space=space, algo=tpe.suggest, max_evals=50, trials=trials)

print("Najlepsze parametry Hyperopt:", best_hyperopt)
```

Elastyczne przetwarzanie danych

- Obsługa brakujących danych (mean, median, most_frequent).
- Skalowanie numerycznych cech (StandardScaler, MinMaxScaler).
- Enkodowanie cech kategorycznych (OneHotEncoder).

Metryki ewaluacji

- Accuracy: Procent poprawnie sklasyfikowanych próbek.
- F1-Score: Średnia harmoniczna precyzji i czułości.
- ROC-AUC: Obszar pod krzywą ROC.
- Confusion Matrix: Macierz błędów, pokazująca poprawność predykcji.

Generowanie raportów

- Automatyczne generowanie raportów w formacie HTML, zawierających:
  - Wyniki metryk.
  - Wizualizacje, takie jak macierz pomyłek czy wykresy konwergencji.

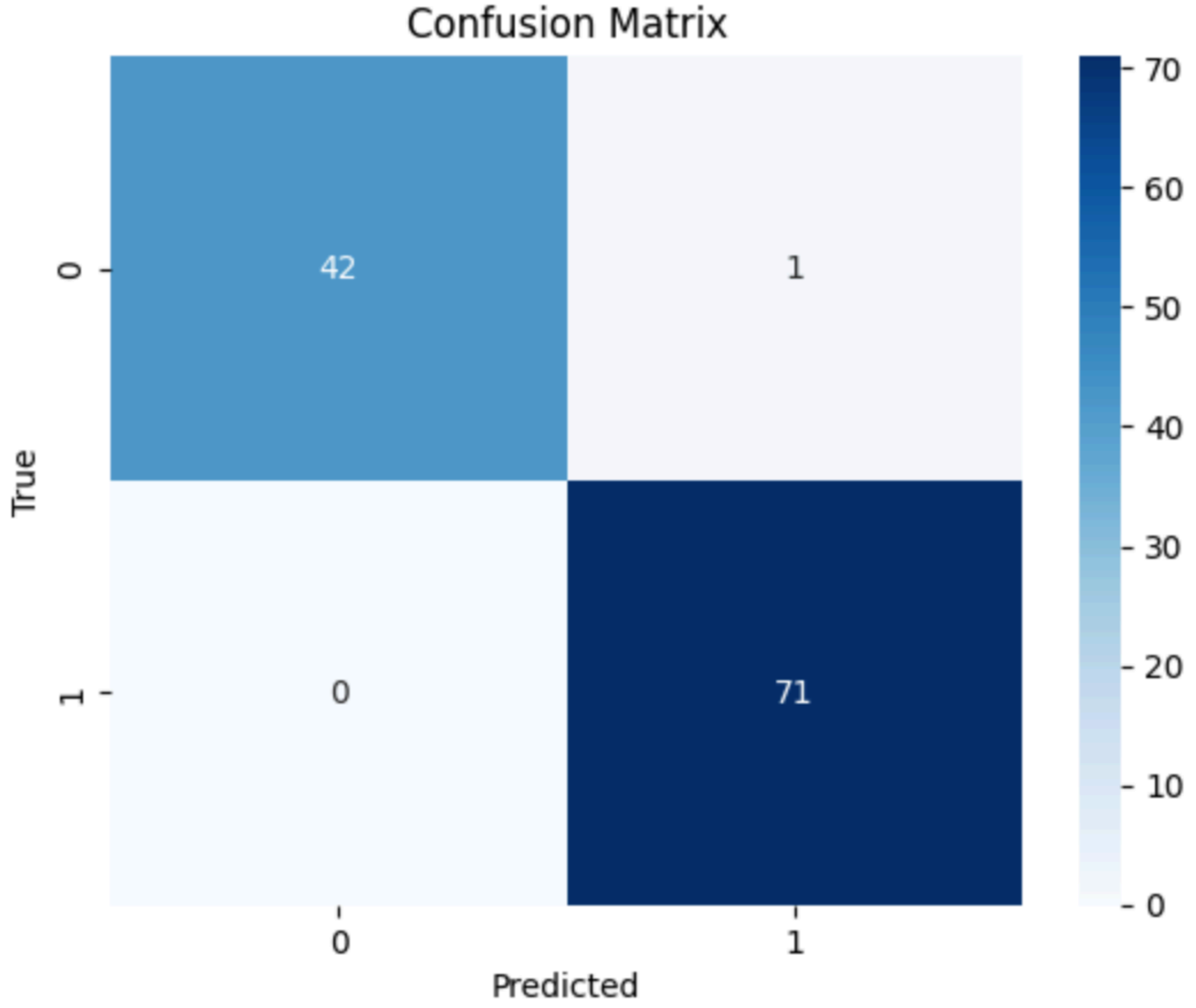
```
# ------------------------------------------------
# 2. Uruchomienie AutoMLClassifier
# ------------------------------------------------
pipeline = AutoMLClassifier(
    strategy_num="mean",
    strategy_cat="most_frequent",
    search_method="random",
    scoring="accuracy",
    n_iter=50,
    cv=5,
    random_state=42
)
```

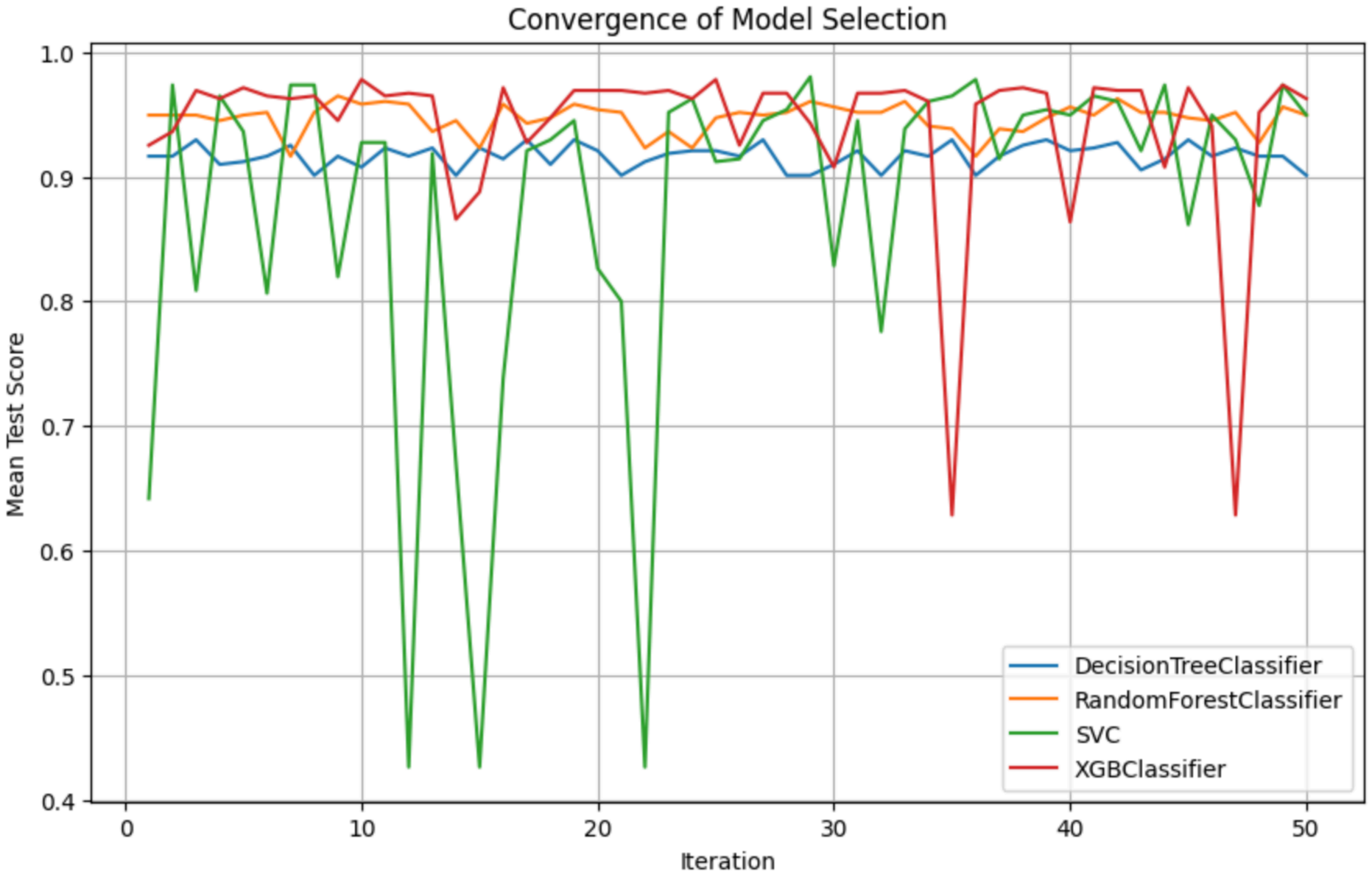
Wyniki

```
Training model: DecisionTreeClassifier
Best score for DecisionTreeClassifier: 0.929670329670329
Training model: RandomForestClassifier
Best score for RandomForestClassifier: 0.964835164835164
Training model: SVC
Best score for SVC: 0.980219780219781
Training model: XGBClassifier
Best score for XGBClassifier: 0.978021978021978
Best model: Pipeline(steps=[('model',
                SVC(C=0.01, class_weight='balanced', degree=5, gamma='auto',
                    kernel='linear', probability=True, random_state=42))])
=== Classification Report ===
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        43
           1       0.99      1.00      0.99        71

    accuracy                           0.99       114
   macro avg       0.99      0.99      0.99       114
weighted avg       0.99      0.99      0.99       114
```


Confusion Matrix
True
Predicted
0
1
42
1
0
71
70
60
50
40
30
20
10
0


Convergence of Model Selection
Mean Test Score
Iteration
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50
DecisionTreeClassifier
RandomForestClassifier
SVC
XGBClassifier