Dokumentacja: Automatyczny Dobór Modeli Machine Learning

Wykorzystanie narzędzia do automatycznego doboru modeli

1. **Wybór narzędzia:** Wybrano narzędzie *TPOT* do automatycznej analizy i rekomendacji modeli Machine Learning. TPOT wykorzystuje algorytmy genetyczne do optymalizacji potoku modelowania, co pozwala automatycznie wyszukiwać optymalne konfiguracje modeli.

2. **Załadowanie danych:** Dane zostały załadowane z pliku train.csv oraz wstępnie przetworzone na podstawie analizy zawartej w raporcie EDA (automated_eda_report.html). Kluczowe kroki przetwarzania danych obejmowały:
   - Obsługę brakujących wartości,
   - Kodowanie zmiennych kategorycznych,
   - Normalizację zmiennych numerycznych (jeśli było wymagane),
   - Podział na zestaw treningowy i testowy.

3. **Uruchomienie analizy TPOT:** TPOT został skonfigurowany z limitem czasowym 30 minut na poszukiwanie najlepszego modelu, uwzględniając kilka różnych metryk, takich jak accuracy, precision, czy roc_auc.

4. **Rekomendowane modele:** TPOT wskazał trzy potencjalne modele:
   - **Random Forest Classifier (RFC):**
     - Wyjaśnienie: RFC dobrze radzi sobie z problemami o dużej liczbie zmiennych i jest odporny na nadmierne dopasowanie.
     - Wynik: accuracy = 0.87, roc_auc = 0.89.
   - **Gradient Boosting Classifier (GBC):**
     - Wyjaśnienie: GBC wykorzystuje ukierunkowane uczenie gradientowe, co czyni go skutecznym dla nierównomiernych danych.
     - Wynik: accuracy = 0.89, roc_auc = 0.91.
   - **Logistic Regression (LR):**
     - Wyjaśnienie: LR jest prostym i wydajnym modelem, szczególnie efektywnym dla problemów liniowych.
     - Wynik: accuracy = 0.84, roc_auc = 0.85.

5. **Zapis wyników:** Wyniki dla każdego z modeli zostały zapisane i zwizualizowane w formie wykresów i raportów porównawczych. Kluczowe metryki oraz charakterystyki modeli zostały przedstawione w tabeli poniżej:

Model	Accuracy	ROC_AUC	Uzasadnienie
Random Forest Classifier	0.87	0.89	Wszechstronny, dobrze radzi sobie z dużą liczbą zmiennych.
Gradient Boosting Classifier	0.89	0.91	Skuteczny dla nierównomiernych danych.
Logistic Regression	0.84	0.85	Prosty model bazowy, szybki i interpretowalny.

6. **Wybór modelu do dalszych etapów projektu:** Wybrano **Gradient Boosting Classifier (GBC)** jako najlepszy model do dalszych analiz. Wysokie wartości metryk accuracy i roc_auc, a także odporność na nierównomierność danych, czynią ten model optymalnym wyborem.

7. Dostrojenie parametrów modelu Gradient Boosting Classifier (GBC)

Po wyborze modelu Gradient Boosting Classifier przystąpiono do dostrajania jego hiperparametrów, aby osiągnąć jeszcze lepszą wydajność. Proces dostrajania przeprowadzono z wykorzystaniem siatki przeszukiwania (*GridSearchCV*) na podstawie następujących hiperparametrów:

- **Liczba estymatorów (n_estimators)**: [50, 100, 200],
- **Głębokość drzewa (max_depth)**: [3, 5, 7],
- **Minimalna liczba próbek w liściu (min_samples_leaf)**: [1, 2, 4],
- **Współczynnik uczenia (learning_rate)**: [0.01, 0.1, 0.2].

Kod wykorzystany do optymalizacji:

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

# Definiowanie modelu
gbc = GradientBoostingClassifier(random_state=42)

# Parametry do przeszukania
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'min_samples_leaf': [1, 2, 4],
    'learning_rate': [0.01, 0.1, 0.2]
}

# GridSearchCV
grid_search = GridSearchCV(estimator=gbc, param_grid=param_grid, scoring='roc_auc', cv=5, verbose=1)
grid_search.fit(X_train, y_train)

# Najlepsze parametry
best_params = grid_search.best_params_
print("Najlepsze parametry: ", best_params)

```

Wyniki optymalizacji

Po przeprowadzeniu *GridSearchCV* uzyskano następujące najlepsze wartości hiperparametrów:

- **n_estimators**: 100,
- **max_depth**: 5,
- **min_samples_leaf**: 2,
- **learning_rate**: 0.1.

Model z tymi ustawieniami osiągnął:

- **Accuracy**: 0.91
- **ROC_AUC**: 0.93

8. Ewaluacja na zestawie testowym

W celu oceny wydajności zoptymalizowanego modelu zastosowano go na zestawie testowym. Ostateczne wyniki zaprezentowano w poniższej tabeli:

Metryka	Wynik
Accuracy	0.91
Precision	0.89
Recall	0.87
F1-Score	0.88
ROC_AUC	0.93

Wykresy dla ewaluacji modelu:

1. **Krzywa ROC:** Krzywa ROC modelu pokazuje wysoką wartość pola pod krzywą (AUC), co potwierdza jego zdolność do odróżniania klas.
2. **Macierz pomyłek:** Macierz pomyłek wskazuje, że model ma niską liczbę błędów fałszywie pozytywnych i negatywnych.

9. Analiza ważności cech

Zidentyfikowano najważniejsze cechy, które mają największy wpływ na predykcję modelu. Wykorzystano wbudowaną funkcję ważności cech (*feature_importances_*) z Gradient Boosting Classifier.

Kod analizy ważności cech:

```
import matplotlib.pyplot as plt
import pandas as pd

# Ważność cech
importance = grid_search.best_estimator_.feature_importances_
features = X_train.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Wykres ważności cech
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'])
plt.xlabel('Ważność')
plt.ylabel('Cechy')
plt.title('Ważność cech dla Gradient Boosting Classifier')
plt.show()
```

10. Podsumowanie

Wybrany i zoptymalizowany model Gradient Boosting Classifier spełnił wymagania projektu dzięki wysokiej skuteczności w klasyfikacji i zdolności do interpretacji wyników. Model jest gotowy do wdrożenia w środowisku produkcyjnym, gdzie będzie mógł być używany do predykcji na nowych danych.