Praktyczne Zastosowanie Drzew Decyzyjnych i Metod Ensemble w Analizie Danych

Laboratorium z Uczenia Maszynowego

Contents

1 Wstęp

Celem ćwiczenia jest zapoznanie się z praktycznym wykorzystaniem drzew decyzyjnych i metod ensemble w analizie danych. Omówimy podstawy teoretyczne, a następnie zastosujemy drzewa decyzyjne oraz techniki ensemble, takie jak bagging, Random Forest i boosting, na przykładach w językach Python i R.

2 Teoria

2.1 Indukcja drzewa decyzyjnego

Indukcja drzewa decyzyjnego to proces budowy drzewa, które klasyfikuje dane na podstawie cech wejściowych. Algorytm konstrukcji drzewa decyzyjnego można zapisać w metajęzyku programowania w następujących krokach:

- **Wejście:** Zbiór danych D, zawierający obserwacje z etykietami klasy, oraz zestaw cech X.
- **Wyjście:** Drzewo decyzyjne, które klasyfikuje obserwacje na podstawie cech.

2.1.1 Algorytm indukcji drzewa decyzyjnego

1. **Inicjalizacja:** Tworzymy korzeń drzewa, który reprezentuje cały zbiór danych D.
2. **Kryterium podziału:** Dla każdego węzła (zaczynając od korzenia) wybieramy cechę, która najlepiej rozdziela dane, używając wybranego kryterium, np. entropii, zysku informacji lub współczynnika Gini.

- Definiujemy funkcję celu $F(X_i)$, która ocenia jakość podziału danych według cechy X_i.
- Dla każdej cechy $X_i \in X$, obliczamy wartość $F(X_i)$.
- Wybieramy cechę X_{best}, która maksymalizuje $F(X_i)$, i tworzymy podział danych w węźle na podstawie wartości X_{best}.

3. **Rekursja:** Dla każdego podzbioru danych D_{sub}, wynikającego z podziału w węźle, powtarzamy krok 2, aż:
   - Wszystkie obserwacje w podzbiorze D_{sub} należą do tej samej klasy, lub
   - Osiągnięto maksymalną głębokość drzewa, lub
   - Podzbiór danych D_{sub} jest zbyt mały, by kontynuować podział.
4. **Przypisanie klasy:** Każdy węzeł liściowy otrzymuje etykietę klasy na podstawie dominującej klasy w danych, które do niego należą.

2.1.2 Pseudokod algorytmu

Poniżej przedstawiono pseudokod dla algorytmu budowy drzewa decyzyjnego.

```
def build_tree(data, features, depth=0, max_depth=None):
    if all_same_class(data) or depth == max_depth or len(data) < min_samples:
        return Leaf(class=majority_class(data))
    best_feature = select_best_feature(data, features)
    tree = Node(feature=best_feature)
    for value in unique_values(data, best_feature):
        subset = split_data(data, feature=best_feature, value=value)
```

```
8           subtree = build_tree(subset, features, depth + 1, max_depth)
9           tree.add_branch(value, subtree)
10      return tree
```

Objaśnienie funkcji:

- `build_tree` - Funkcja rekurencyjnie budująca drzewo decyzyjne.
- `all_same_class(data)` - Sprawdza, czy wszystkie próbki w zbiorze mają tę samą klasę.
- `majority_class(data)` - Zwraca dominującą klasę w zbiorze danych.
- `select_best_feature(data, features)` - Wybiera cechę o najwyższej wartości funkcji celu (np. zysku informacji).
- `split_data(data, feature, value)` - Dzieli zbiór danych na podzbiory na podstawie wartości wybranej cechy.

2.2 Kryteria wyboru cech

Najczęściej stosowane kryteria wyboru cech podczas budowy drzewa decyzyjnego to:

- **Entropia i zysk informacji** - Entropia mierzy niepewność w zbiorze danych. Zysk informacji definiuje różnicę w entropii przed i po podziale.
- **Współczynnik Gini** - Alternatywna miara, która ocenia „czystość” podziału; minimalizuje liczbę błędnych klasyfikacji.

3 Metody Ensemble

Metody ensemble łączą kilka modeli bazowych w celu poprawy wydajności i stabilności predykcji. Do najpopularniejszych metod ensemble należą:

- **Bagging (Bootstrap Aggregating)**: Technika, w której wiele modeli jest trenowanych na różnych próbkach danych, a ich wyniki są uśredniane.
- **Boosting**: Technika, która buduje kolejne modele, koncentrując się na poprawie predykcji dla obserwacji trudnych do sklasyfikowania.
- **Random Forest**: Specjalny przypadek baggingu, w którym tworzymy wiele drzew decyzyjnych, a przy każdym podziale losowo wybierana jest część cech.

4 Przygotowanie środowiska

Przed rozpoczęciem ćwiczenia należy zainstalować następujące pakiety w Pythonie:

```
pip install numpy pandas scikit-learn matplotlib
```

oraz w R:

```
install.packages("rpart")
install.packages("randomForest")
install.packages("gbm")
```

5 Analiza danych z użyciem Drzew Decyzyjnych w Pythonie

5.1 Załadowanie danych

W tym przykładzie wykorzystamy zbiór danych `Iris` do klasyfikacji.

```
import pandas as pd
from sklearn.datasets import load_iris

data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
```

5.2 Budowa Drzewa Decyzyjnego

Do stworzenia drzewa decyzyjnego użyjemy klasy `DecisionTreeClassifier` z biblioteki `scikit-learn`.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Podział danych na zbiory treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
7
8  # Inicjalizacja modelu
9  tree = DecisionTreeClassifier(max_depth=3, random_state=42)
10 tree.fit(X_train, y_train)
11
12 # Ewaluacja modelu
13 y_pred = tree.predict(X_test)
14 print("Accuracy:", accuracy_score(y_test, y_pred))
```

6 Random Forest w Pythonie

6.1 Budowa modelu Random Forest

Random Forest to metoda ensemble oparta na drzewach decyzyjnych, która losowo wybiera próbki danych i cechy przy tworzeniu każdego drzewa. W Pythonie zaimplementujemy model z użyciem klasy `RandomForestClassifier`.

```
1  from sklearn.ensemble import RandomForestClassifier
2
3  # Inicjalizacja modelu Random Forest
4  rf_model = RandomForestClassifier(n_estimators=100, max_depth=3, random_state=42)
5  rf_model.fit(X_train, y_train)
6
```

```
# Predykcja i ewaluacja modelu
y_pred_rf = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

7 Analiza danych z użyciem Bagging w R

7.1 Budowa modelu Bagging

W R zastosujemy metodę `randomForest`, która implementuje bagging.

```
library(randomForest)

#[U+FFFD]adowanie danych iris
data(iris)
set.seed(42)

# Podzia[U+FFFD]na zbiory treningowy i testowy
sample <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample, ]
test_data <- iris[-sample, ]

# Trening modelu randomForest
```

```r
model_rf <- randomForest(Species ~ ., data=train_data, ntree=100)

# Predykcja
predictions <- predict(model_rf, test_data)
accuracy <- sum(predictions == test_data$Species) / nrow(test_data)
print(paste("Accuracy:", accuracy))
```

8 Random Forest w R

8.1 Budowa modelu Random Forest w R

Random Forest w R można zbudować przy użyciu tej samej funkcji `randomForest`, co w przypadku baggingu, dostosowując odpowiednie parametry modelu.

```r
# Trening modelu Random Forest
model_rf <- randomForest(Species ~ ., data=train_data, ntree=100, mtry=2)

# Predykcja
predictions <- predict(model_rf, test_data)
accuracy <- sum(predictions == test_data$Species) / nrow(test_data)
print(paste("Random Forest Accuracy:", accuracy))
```

9 Boosting - Analiza danych w Pythonie z użyciem XGBoost

9.1 Instalacja XGBoost

W terminalu zainstaluj pakiet `xgboost`:

```
pip install xgboost
```

9.2 Budowa modelu Boosting

Przykład budowy modelu z użyciem XGBoost:

```
from xgboost import XGBClassifier

# Inicjalizacja modelu
xgb = XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1)
xgb.fit(X_train, y_train)

# Predykcja i ocena modelu
y_pred_xgb = xgb.predict(X_test)
print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
```

10 Boosting w R - Gradient Boosting

10.1 Budowa modelu Gradient Boosting w R

W R wykorzystamy pakiet `gbm` do implementacji gradient boosting.

```r
library(gbm)

# Trening modelu GBM
set.seed(42)
model_gbm <- gbm(Species ~ ., data=train_data, distribution="multinomial", n.trees=100, interaction.depth=3)

# Predykcja
predictions <- predict(model_gbm, test_data, n.trees=100, type="response")
predicted_class <- colnames(predictions)[apply(predictions, 1, which.max)]
accuracy <- sum(predicted_class == test_data$Species) / nrow(test_data)
print(paste("GBM Accuracy:", accuracy))
```

11 Wstęp

Celem ćwiczenia jest praktyczne zapoznanie się z tworzeniem modeli klasyfikacji, takich jak drzewa decyzyjne, Random Forest i boosting, przy użyciu środowiska KNIME. KNIME (Konstanz Information Miner) to graficzne środowisko open-source do analizy danych, które umożliwia

łatwe łączenie komponentów bez konieczności pisania kodu.

12 Podstawy środowiska KNIME

KNIME oferuje narzędzie do budowy przepływów pracy (ang. *workflow*) poprzez przeciąganie i łączenie węzłów. Każdy węzeł pełni określoną funkcję, taką jak ładowanie danych, wstępne przetwarzanie, trenowanie modelu lub ocena wyników.

12.1 Instalacja KNIME

Aby rozpocząć, należy pobrać i zainstalować najnowszą wersję KNIME ze strony `https://www.knime.com/downloads`. Po zainstalowaniu i uruchomieniu aplikacji można rozpocząć tworzenie przepływów pracy.

13 Wprowadzenie do Drzew Decyzyjnych w KNIME

13.1 Ładowanie danych

Aby załadować dane do KNIME, użyj węzła `File Reader`.

1. Dodaj węzeł `File Reader` do obszaru roboczego.
2. Podłącz plik danych (np. w formacie CSV) i skonfiguruj węzeł, aby zaimportować dane.
3. Uruchom węzeł, aby wczytać dane.

13.2 Wstępne przetwarzanie danych

Węzły, takie jak `Column Filter`, `Row Filter` oraz `Normalizer`, pozwalają na filtrowanie kolumn, wybór podzbioru danych lub normalizację wartości cech.

- **Column Filter:** Pozwala na wybór odpowiednich kolumn do analizy, np. usunięcie kolumn nieistotnych dla klasyfikacji.
- **Row Filter:** Umożliwia filtrowanie wybranych wierszy na podstawie określonych kryteriów.

13.3 Trenowanie Drzewa Decyzyjnego

1. Dodaj węzeł `Partitioning` do podziału danych na zbiór treningowy i testowy.
2. Dodaj węzeł `Decision Tree Learner` i połącz go ze zbiorem treningowym.
3. Wybierz zmienną docelową (kolumnę klasy) oraz skonfiguruj głębokość drzewa, jeśli jest to konieczne.
4. Dodaj węzeł `Decision Tree Predictor` i połącz go ze zbiorem testowym.
5. Uruchom oba węzły, aby uzyskać prognozy modelu drzewa decyzyjnego.

14 Random Forest w KNIME

14.1 Trenowanie modelu Random Forest

1. Dodaj węzeł `Random Forest Learner` i połącz go ze zbiorem treningowym.
2. W konfiguracji węzła wybierz zmienną docelową oraz liczbę drzew (`Number of Trees`) i maksymalną głębokość (`Max Depth`), jeśli chcesz ją ograniczyć.
3. Dodaj węzeł `Random Forest Predictor` i podłącz go do zbioru testowego.
4. Uruchom oba węzły, aby zobaczyć wyniki klasyfikacji dla modelu Random Forest.

14.2 Ocena wyników modelu

1. Dodaj węzeł `Scorer` do analizy wyników predykcji.
2. Połącz węzeł `Scorer` z wyjściem węzła predykcji (np. `Decision Tree Predictor` lub `Random Forest Predictor`).
3. Uruchom węzeł `Scorer`, aby uzyskać dokładność oraz inne miary, takie jak precyzja, recall oraz F1-score.

15 Boosting w KNIME

Boosting można zaimplementować przy użyciu węzła `Gradient Boosted Trees Learner`.

15.1 Trenowanie modelu Boosting

1. Dodaj węzeł `Gradient Boosted Trees Learner` i połącz go ze zbiorem treningowym.
2. W konfiguracji węzła wybierz kolumnę docelową oraz ustaw liczbę iteracji boostingowych (`Number of Boosting Rounds`).
3. Dodaj węzeł `Gradient Boosted Trees Predictor` i podłącz go do zbioru testowego.
4. Uruchom oba węzły, aby uzyskać predykcje modelu boostingowego.

15.2 Ocena modelu Boosting

Podobnie jak w przypadku innych modeli, dodaj węzeł `Scorer` do oceny wyników predykcji modelu boostingowego.

16 Przykładowy Przepływ Pracy

16.1 Opis przepływu pracy

Na powyższym przepływie pracy (rysunek 1) przedstawiono typowe etapy analizy:

- **Wczytanie danych** za pomocą węzła `File Reader`.
- **Wstępne przetwarzanie** za pomocą węzłów `Column Filter` oraz `Row Filter`.
- **Trenowanie i predykcja** za pomocą węzłów `Decision Tree Learner`, `Random Forest Learner` i `Gradient Boosted Trees Learner`.

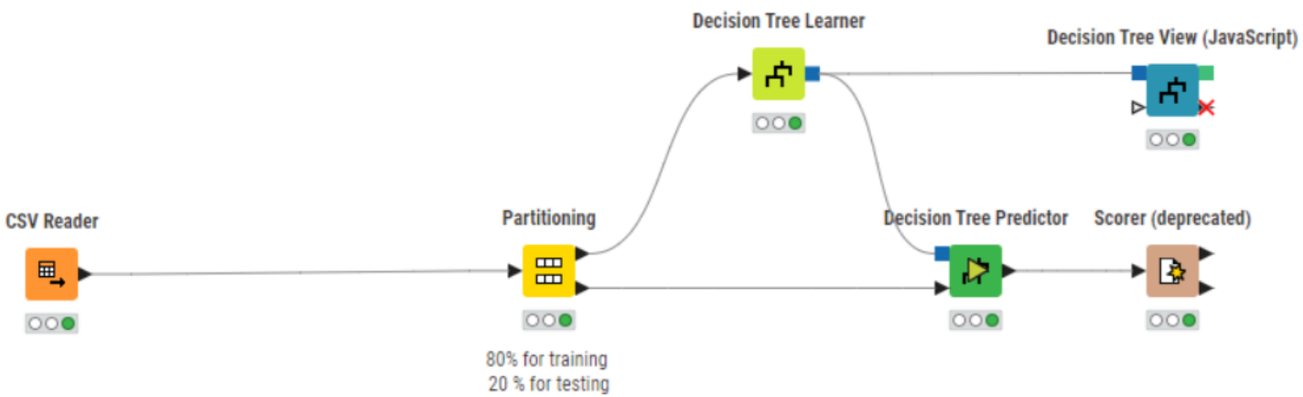

Figure 1: Przykładowy przepływ pracy dla analizy danych za pomocą Drzewa Decyzyjnego, Random Forest oraz Boosting w KNIME

- **Ewaluacja wyników** przy użyciu węzłów `Decision Tree Predictor`, `Random Forest Predictor`, `Gradient Boosted Trees Predictor` oraz `Scorer`.

17 Wnioski

KNIME pozwala na efektywne i intuicyjne tworzenie modeli klasyfikacyjnych za pomocą graficznego przepływu pracy. Modele takie jak drzewa decyzyjne, Random Forest i boosting można łatwo zaimplementować za pomocą odpowiednich węzłów KNIME, co umożliwia analizę danych bez konieczności pisania kodu.

Drzewa decyzyjne są prostymi, ale skutecznymi modelami uczenia maszynowego, szczególnie w analizie danych. Metody ensemble, takie jak bagging, Random Forest, i boosting, pozwalają na poprawę dokładności i stabilności modeli poprzez łączenie wielu słabszych klasyfikatorów. Random Forest dodaje dodatkowy element losowości w wyborze cech, co

18 Zadanie

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

1. klasyfikacja choroby Parkinsona `https://www.kaggle.com/dipayanbiswas/parkinsons-disease-speech-signal-features?select=pd_speech_features.csv`

2. smoking patients https://www.kaggle.com/thomaskonstantin/cpg-values-of-smoking-and-non-smoking-patients

3. Powikłania zawału mięśnia sercowego https://www.kaggle.com/rafatashrafjoy/myocardial-infarction-complications

4. Sygnały kardiotokografii https://www.kaggle.com/sohelranaccselab/biomedical-cardiotocography

5. Badania pH https://www.kaggle.com/zfturbo/measurements-of-urine-ph

6. Analiza cukrzycy https://www.kaggle.com/veerukhannan/diabetes

7. Choroba Alzheimera https://www.kaggle.com/madhucharan/alzheimersdisease5classdatasetadni

8. Prostate cancer https://www.kaggle.com/ashrafalsinglawi/prostate-cancer-survival-data

9. klasyfikacja choroby Parkinsona https://www.kaggle.com/dipayanbiswas/parkinsons-disease-speech-signal-features?select=pd_speech_features.csv

10. smoking patients https://www.kaggle.com/thomaskonstantin/cpg-values-of-smoking-and-non-smoking-patients

11. Powikłania zawału mięśnia sercowego https://www.kaggle.com/rafatashrafjoy/myocardial-infarction-complications

12. Sygnały kardiotokografii https://www.kaggle.com/sohelranaccselab/biomedical-cardiotocography

13. Badania pH https://www.kaggle.com/zfturbo/measurements-of-urine-ph

14. Analiza cukrzycy https://www.kaggle.com/veerukhannan/diabetes

15. Choroba Alzheimera https://www.kaggle.com/madhucharan/alzheimersdisease5classdatasetadni

16. Prostate cancer https://www.kaggle.com/ashrafalsinglawi/prostate-cancer-survival-data

Rozwiązywanie zadania opracować zarówno w KNIME jak Jupyter Notebook.

References

References

[Conway (2015)] Drew Conway, John Myles White, Uczenie maszynowe dla programistów, Helion, 2015

[Beginner's Guide (2018)] Beginner's Guide `https://www.datacamp.com/community/tutorials/r-packages-guide`

[AGH (2018)] Przykładowe rozwiązania `http://zasoby.open.agh.edu.pl/~15spkiepas/index.php/przykladowe-rozwiazania/index.html`