Drzewa decyzyjne do klasyfikacji akcji reklamowej w bankach

Joanna Kołodziejczyk

1 Wprowadzenie - cel zajęć

Celem zadania jest wytworzenie drzewa klasyfikacyjnego, określającego, czy klient zapisze się (tak/nie) na lokatę terminową. Zbiór danych nazywa się bank.csv a jego źródłem jest repozytorium `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing#` Dane te związane są z bezpośrednimi kampaniami marketingowymi (telefonicznymi) portugalskiej instytucji bankowej.

1. Wejściem do systemu jest zbiór danych.
2. Wynikiem jest model w postaci drzewa decyzyjnego.

2 Zadanie krok po kroku

Do zadania wykorzystany zostanie język Python i sugerowane jest środowisko PyCharm lub Spider. Konieczne jest załadowanie bibliotek.

Listing 1: Konieczne biblioteki

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score, classification_report
import graphviz as graphviz
```

1. Pandas: powerful Python data analysis toolkit - do pracy z danymi tabelarycznymi, takimi jak dane przechowywane w arkuszach kalkulacyjnych lub bazach danych. Pandas ma narzędzia do zbadania, oczyszczania i przetwarzania danych. W Pandas, tabela danych jest nazywana DataFrame. `https://pandas.pydata.org/docs/`
2. NumPy to podstawowy pakiet dla obliczeń naukowych w Python.
3. Matplotlib to obszerna biblioteka do tworzenia statycznych, animowanych i interaktywnych wizualizacji w Pythonie.

4. Seaborn to biblioteka wizualizacji danych Pythona oparta na Matplotlib. Jest przeznaczony do tworzenia wyrafinowanej grafiki statystycznej.

5. Sklearn - Scikit-learn jest biblioteką do maszynowego uczenia się typu open source, która wspiera modele nadzorowane i nienadzorowane. Dostarcza również narzędzia do dopasowywania modeli, wstępnego przetwarzania danych, wyboru i oceny modeli oraz wiele innych narzędzi. Scikit-learn dostarcza dziesiątki wbudowanych algorytmów i modeli uczenia maszynowego, zwanych estymatorami. Każdy estymator może być dopasowany do niektórych danych przy użyciu pewnej metody dopasowania.

2.1 Wczytanie zbioru treningowego - zapoznanie z danymi

Zanim przystąpi się do pracy z danymi ważne jest poznanie ich specyfiki oraz atrybutów i wartości. Opis zbioru znajduje się w repozytorium `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing#`. Plik csv znajduje się z zbiorze o nazwie bank.csv.

Wczytanie danych należy wykonać z pomocą metody `read_csv()` z biblioteki `Pandas`. Powstanie dataFrame `bank`:

Listing 2: Wczytanie danych

```
bank=pd.read_csv('../input/bank.csv')
```

Można wyświetlić początkowe wiersze z bank.

Listing 3: Zapoznanie się z danymi

```
print( bank.shape)
print( bank.head())
```

2.2 Puste komórki

Sprawdzenie, czy zbiór nie zawiera komórek pustych - null.

Listing 4: Czy w danych są wartości puste?

```
print (bank[bank.isnull().any(axis=1)].count())
```

2.3 Analiza zbioru treningowego

Analiza statystyk deskrypcyjnych na danych numerycznych. Wykonać podsumowanie z danych. W zmiennej `description`, zapisane są średnie arytmetyczne, odchylenie standardowe, mediana, kwantyle i inne.

Listing 5: Statystyki na danych

```
print(bank.describe())
```

Przykład wykresów dla zmiennej „age” klienta.

Listing 6: Statystyki i wykresy na danych

```
plt.figure(1)
plt.subplot(1, 2, 1)
```

```
sns.boxplot(x=bank["age"])
plt.subplot(1, 2, 2)
sns.set(style="ticks", color_codes=True)
sns.distplot(bank.age, bins=100)
plt.show()
```

Wnioski z analizy rozkładów. Dane o wieku są źle zbalansowane, rozkład wartości w populacji jest prawoskośny. Wszystkie wartości powyżej 75% percentyla powinny zostać uznane za wartości mocno odstające. Ewentualnie można podjąć decyzję o pozbyciu się ich lub dyskretyzacji atrybutu.

Przykład analizy zmiennej dyskretne „job" praca klienta.

Listing 7: Wykresy dla atrybutu job

```
print( bank.job.value_counts())
plt.figure(2)
sns.countplot(y='job', data=bank)
plt.show()
```

Wnioski z wizualizacji. Dane o „job" pokazują, że jedna z kategorii „unknown" jest bardzo słabo reprezentowana. Można ją usunąć posługując się poleceniem

Listing 8: Usunięcie rekordów z daną wartością atrybutu

```
bank = bank[bank.job != 'unknown']
print(bank.shape)
# ZADANIE: Policzyć ile rekordów zostało usuniętych
sns.countplot(y='job', data=bank)
plt.show()
```

Można przeanalizować atrybut „poutcome" - czyli skuteczność reklamy. Po analizie wartości i znaczenia atrubutu zamieniona zostanie wartość „other" na „unknown"

Listing 9: Połączenie unknown i other ponieważ other nie pasuje ani do success ani do failure.

```
print(bank.poutcome.value_counts())
bank['poutcome'] = bank['poutcome'].replace(['other'] , 'unknown')
print(bank.poutcome.value_counts())
```

Analiza atrybutu „contact" - czyli medium komunikacji pozwala na podjęcie decyzji o usunięciu atrybutu:

Listing 10: Ze względu na bezużyteczność contact zostanie usunięty

```
print(bank.contact.value_counts())
bank.drop('contact', axis=1, inplace=True)
```

2.4 Przygotowanie danych zgodnie z wymaganiami klasyfikatora

Implementacja algorytmu drzewa decyzyjnego w sklearn nie obsługuje wartości dyskretnych, a jedynie wartości binarne lub rzeczywiste, dlatego każdą etykietę trzeba zakodować do wartości liczbowej.

Listing 11: Zamiana etykiet na wartości numeryczne

```
le = preprocessing.LabelEncoder()
```

```
for i in range(0,len(bank.columns)):
    bank.iloc[:,i] = le.fit_transform(bank.iloc[:,i])
print (bank.head())
```

2.5 Podział na dane trenujące i testowe

Uczenie się klasyfikacji i testowanie modelu na tych samych danych jest metodologicznym błędem: model, który tylko powtarzałby etykiety próbek, które właśnie widział, miałby doskonały wynik, ale nie przewidywałby niczego użytecznego na jeszcze niewidzianych danych. Ta sytuacja jest nazywana przeuczeniem. Aby jej uniknąć, powszechną praktyką podczas przeprowadzania (nadzorowanego) eksperymentu uczenia maszynowego jest trzymanie części dostępnych danych jako zestawu testów X_test, y_test.

W podanym przykładzie następuje podział danych w proporcjach 70% zbioru danych wejściowych to zbiór uczący, 30% danych to dane testowe.

W scikit-learn losowy podział na zestawy treningowe i testowe można szybko uzyskać za pomocą funkcji **train_test_split**.

Listing 12: Rozdział danych na zbiór treningowy i testujący

```
X = bank.drop("deposit",1) #Jako zbiór próbek traktujemy wszystkie kolumny
    oprócz ostatniej
y = bank.deposit # Target - klasa - wartość oczekiwana

featureNames = X.columns.tolist() #Lista nazw atrybutów
className = ("deposit") #nazwa klasy

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=0)

print("Liczba przykładów treningowych "+str(len(X_train.index)))
print("Liczba elementów w zbiorze testowym "+str(len(X_test.index)))
```

2.6 Generowanie drzewa, gdzie miarą zabrudzenia zbioru jest entropia

Drzewa decyzyjne (DT) są nieparametryczną nadzorowaną metodą uczenia się stosowaną do klasyfikacji i regresji. Celem jest stworzenie modelu, który przewiduje wartość zmiennej docelowej poprzez uczenie się prostych zasad decyzyjnych wynikających z cech danych.

DecisionTreeClassifier może być wykorzystany do klasyfikacji do dwóch lub więcej klas. Podobnie jak w przypadku innych klasyfikatorów, DecisionTreeClassifier pobiera jako wejście dwie tablice: tablicę X oraz tablicę Y (wartości - liczby całkowite) zawierającą etykiety klasy dla próbek treningowych:

Scikit-learn używa zoptymalizowanej wersji algorytmu CART; jednakże implementacja scikit-learn nie obsługuje na razie zmiennych etykietowanych.

CART (Classification and Regression Trees) jest bardzo podobny do C4.5, ale różni się tym, że obsługuje numeryczne atrybuty wyjściowe (regresję) i nie oblicza zestawów reguł. CART konstruuje drzewa binarne przy użyciu progu, które dają największy zysk informacji (information gain) w każdym węźle.

W scikit-learn, estymatorem klasyfikacji jest obiekt, który implementuje metody fit(X, y) i predict(T). Konstruktor estymatora przyjmuje jako argumenty parametry modelu.

Funkcja accuracy_score oblicza dokładność, albo odsetek (domyślnie) albo liczbę (normalize=False) poprawnych przewidywań.

Listing 13: Drzewo klasyfikacyjne o maksymalnej głębokości z miarą entropia

```
drzewo = tree.DecisionTreeClassifier(criterion="entropy",random_state=0)
drzewo.fit(X_train,y_train)

prediction = drzewo.predict(X_train)
accuracyScore = accuracy_score(y_train, prediction)
print("Drzewo decyzyjne osiąga dokładność na zbiorze trenującym/uczącym: "+
    str(accuracyScore))

prediction = drzewo.predict(X_test)
accuracyScore = accuracy_score(y_test, prediction)
print("Drzewo decyzyjne osiąga dokładność na zbiorze testowym: "+str(
    accuracyScore))
print(classification_report(y_test.to_numpy(), prediction))

dot_data = tree.export_graphviz(decision_tree=drzewo, feature_names=
    atrybuty, class_names=["0","1"], filled=True, rounded=True)
graph = graphviz.Source(dot_data, filename="test.gv", format="png")

graph.view()
```

3 Sprawozdanie

Dotyczy badania klasyfikacji zwierząt na zbiorze Census Income Data Set `https://archive.ics.uci.edu/ml/datasets/Census+Income` Sprawozdanie powinno zawierać następujące punkty:

1. Opisać zbiór podając liczbę rekordów i atrybutów. Wykonać zestawienie atrybutów podając nazwy atrybutów i zakres przyjmowanych przez nie wartości. (1pkt)

2. Czy w danych znajdują się puste komórki? Jeżeli tak, to w jakich atrybutach i ile? (0.5pkt)

3. Drzewo o maksymalnym rozmiarze ma zwykle dobre dopasowanie do danych treningowych (przeuczenie). Ponadto trudno z niego uzyskać łatwo interpretowane reguły. Należy zatem przetestować DecisionTreeClassifier dla różnych wartości głębokości drzewa (parametr w metodzie „max_depth") i wygenerować drzewa o rozmiarach od 2 do 9. Przedstawić otrzymane wartości dokładności (accuracy) w procesie uczenia i testowania. Napisać wnioski do wyników. (1pkt)

4. Innym parametrem DecisionTreeClassifier jest miara zanieczyszczenia, która służy do wyboru węzła. Należy dokonać badania porównującego zmienny „max_depth" jak w poprzednim punkcie, z ustawionym criterion="gini". `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html` (1pkt)

5. Wybrać głębokość drzewa 3 lub 4 i zapisać słownie 3 reguły występujące w drzewku. (0.5pkt)

6. Wrócić do zadania z punktu 2.3 z analizą zbioru danych (atrybutów, wartości). Przejrzeć i przeanalizować różne atrybuty, dokonać oceny wartości, rozkładów i wizualnej. Czy dla jakiegoś atrybutu wykresy pudełkowe są warte pokazania i omówienia? To samo pytanie dotyczy histogramów „distplot" czy „countplot"? Wybrać atrybut i przedstawić wraz z komentarzem opisując przeprowadzony proces myślowy. Dokonać czyszczenia (można nawet pozbyć się atrybutu). Po zmianach dokonanych na zbiorze testowym dokonać ponownego uczenia klasyfikatora. Czy poczynione zmiany miały wpływ na jakość modelu? Zapisać wnioski. (1pkt)

Sprawozdanie (format pdf) proszę podpiąć na Teams. Sprawozdanie może być przygotowane w formie notatki w Jupyter Notebook (format .ipynb). Proszę zadbać by plik nazwany został nazwiskiem i numerem indeksu autorki/autora.

Czas wykonania zadania: tydzień.