**Autor: Zuzanna Wojtyniak**

Raport przyrostu indywidualnego z przedmiotu „ Sztuczna inteligencja”

Temat projektu: Automatyczny kelner

Metoda uczenia: Drzewo decyzyjne

Problem: Wybór produktu przez klienta

Drzewo decyzyjne to nic innego jak graficzny sposób wspierania procesu decyzyjnego. Drzewo stosowane jest w teorii decyzji i ma sporo zastosowań. Może zarówno rozwiązać problem decyzyjny, jak i stworzyć plan.

Do mojego projektu wykorzystałam decision-tree-id3. Jest to moduł stworzony do tworzenia drzew decyzyjnych wykorzystujący algorytm ID3.

Wymagania do instalacji decision-tree-id3:

- Python (>= 2.7 or >= 3.3)

- NumPy (>= 1.6.1)

- Scikit-learn (>= 0.17)

Instalacja:

pip install decision-tree-id3

or clone the project using:

git clone https://github.com/svaante/decision-tree-id3.git

cd decision-tree-id3

python setup.py install

Kiedy instalacja zakończy się pomyślnie, można podpiąć owy moduł do naszego projektu:

from id3 import Id3Estimator, export\_graphviz

W algorytmach konstrukcji drzew jednym z kluczowych elementów jest wybór kolejności cech, według których, na poszczególnych etapach, będzie dokonywany podział zbioru obiektów.

Hierarchiczność podejmowania decyzji jest cechą, która wyróżnia drzewo decyzyjne od innych metod.

Wybór atrybutów opiera się na liczeniu entropii, dzięki czemu wiemy dla którego atrybutu jest najwięcej przyrostu informacji. Entropia jest to miara informacji, im mniejsza tym jest więcej informacji. Za pomocą algorytmu decyzyjnego tworzymy korzeń drzewa i rozdzielamy dane wobec cechy mającej najwięcej informacji. Poprzez wielokrotne iteracje możemy powtarzać procedurę rozdzielania danych w każdym potomnym węźle, aż uzyskamy same liście.

Opis algorytmu:

1. Oblicz entropię dla każdego atrybutu

2. Wybierz atrybut A z najniższą entropią

3. Podziel zbiór przykładów uczących ze względu na wartość atrybutu A na rozłączne podzbiory

4. Dodaj do drzewa krawędzie z warunkami:

jeśli A=a1 to ... (poddrzewo 1)

jeśli A=a2 to ... (poddrzewo 2)

...

5. Dla każdego poddrzewa wykonaj kroki od 1.

6. W każdej iteracji jeden atrybut jest usuwany. Algorytm zatrzymuje się,

gdy do rozpatrzenia nie pozostanie juz żaden atrybut lub wszystkie przykłady

w danym podrzewie mają tą samą wartość atrybutu decyzyjnego.

W moim projekcie do stworzenia korzystam z trzech tabel przechowujących kolejno:

- 12 nazw cech zestawu uczącego:

feature\_names = ["kwasne", "gorzkie", "szybkie", "pitne", "slodkie", "lekkostrawne", "na wynos", "kaloryczne", "zdrowe", "ekskluzywne", "tanie", "smaczne"]

- 150 przypadków testowych

feature\_names = X = np.array([

[10,10,1,9,9,9,9,1,1,9,4,"tak"],

[9,5,7,7,9,7,0,0,8,1,4,"przepyszne"],

[9,7,7,9,10,0,0,10,9,0,0,"przepyszne"],

[0,0,1,3,3,3,9,9,0,3,3,"domowej roboty"],…

*Wartości przypadków testowych są w skali od 0-10. Każdy z tych wartości odpowiada po kolei nazwie konkretnej cechy zestawu uczącego.*

- oraz ich decyzje zbioru

y = np.array(["chlodnik",

"ciasto czekoladowe",

"frytki",

"krokiety z miesem",

…

Następne linie kodu odpowiadają odpowiednio za tworzenie drzewa oraz na zobrazowaniu go

# tworzenie drzewa decyzyjnego

clf = Id3Estimator()

# fit - synonim do "find patterns in data"

clf.fit(X, y, check\_input=True)

export\_graphviz(clf.tree\_, "test.dot", feature\_names)…

Drzewo to jest zobrazowane (przy użyciu GraphViz ) w pliku PDF. Możemy również zobrazować do plików w formatach takich jak GIF, PNG, SVG czy PostScript. Graphviz to zestaw narzędzi do tworzenia diagramów za pomocą grafów. Tworzymy to poniższym poleceniem w wierszu poleceń:

dot -Tpdf tree.dot -o tree.pdf

Drzewo decyzyjne w projekcie „Automatyczny kelner” odpowiada za decyzje jaki produkt ma polecić kelner klientowi. Kelner pyta się jaki produkt klient pragnie, a następnie dostosowując się do odpowiedzi klienta proponuje mu dany produkt.

Bardzo przydatną funkcją w moim projekcie jest funkcja predict, która wyświetla decyzje na podstawie przypadków testowych.

Źródła:

<https://svaante.github.io/decision-tree-id3/>

<http://graphviz.org/>