**Przewidywanie rozwodów**

Celem projektu jest implementacja algorytmu lasu losowego na podstawie danych dotyczących rozwodów (<http://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set>).

Dane te składają się z odpowiedzi na 54 pytania dotyczące wspólnych wartości, obowiązków w małżeństwie oraz życia codziennego. Klasyfikacja jest binarna (rozwiedziony lub nie).

Algorytm został zaimplementowany w języku *Python* z wykorzystaniem bibliotek: *numpy, pandas, random.*

**Funkcje** (plik Functions.py)**:**

* train\_test\_split(df, test\_size):

funkcja dzieląca zbiór danych na dwa podzbiory – treningowy na którym algorytm będzie się uczył oraz testowy w celu ewaluacji.

Parametry:

df – zbiór danych,

test\_size = rozmiar zbioru testowego w skali (0,1).

Zwraca: zbiór treningowy, zbiór testowy.

* calculate\_accuracy(predictions, labels):

sprawdza poprawność predykcji dokonanej przez las losowy.

Parametry:

predictions – klasy zwrócona przez algorytm,

labels – rzeczywiste klasy.

Zwraca: procentowy udział poprawnych klasyfikacji.

* sample\_data(train\_df, sample\_size):

wybiera losowe obserwacje z podanego zbioru. (Będą one wykorzystane do budowy jednego z drzew lasu losowego.)

Parametry:

train\_df – zbior z którego losowane są obiekty,

sample\_size – ilość losowanych obiektów.

Zwraca: zbiór wylosowanych obserwacji.

* unique\_check(data):

sprawdza czy wszystkie obiekty są tej samej klasy.

Parametry:

data - zbiór obiektów.

Zwraca: odpowiednio *true* lub *false*.

* classify(data):

klasyfikuje zbiór danych według najczęściej występującej klasy.

Parametry:

data – zbiór obiektów.

Zwraca: wynik klasyfikacji.

* find\_split(data, no\_of\_attributes):

funkcja wybiera podaną ilość atrybutów, która będzie wykorzystana do stworzenia pojedynczego drzewa w lesie losowym. Znajduje również atrybut maksymalizujący zdobycz informacyjną oraz wartość danego atrybutu według której obserwacje zostaną podzielone.

Parametry:

data – zbiór obiektów,

no\_of\_attributes – liczba atrybutów wykorzystana do stworzenia

pojedynczego drzewa.

Zwraca: atrybut, punkt podziału, podzbiór spełniający warunek, podzbiór niespełniający warunku.

* entropy\_calc(data):

funkcja licząca entropię podanego podzbioru.

Parametry:

data – zbiór obiektów.

* global\_entropy\_calc(data\_true, data\_false):

funkcja licząca entropię po podziale zbioru według pewnego atrybutu.

Parametry:

data\_true – część zbioru spełniająca warunek,

data\_false – część zbioru niespełniająca warunku.

* decision\_tree(df, no\_of\_attributes, counter=0):

funkcja budująca drzewo decyzyjne. W każdej swojej iteracji zwraca liść, jeśli obiekty danego podzbioru należą do tej samej klasy, w przeciwnym wypadku buduje poddrzewo.

Parametry:

df – zbiór danych;

no\_of\_attributes - liczba atrybutów wykorzystana do stworzenia

pojedynczego drzewa;

counter – wbudowany licznik iteracji

Zwraca: drzewo decyzyjne.

* predict(observation, tree):

klasyfikuje dany obiekt wykorzystując podane drzewo decyzyjne.

Parametry:

observation – pojedyncza obserwacja,

tree – drzewo decyzyjne

Zwraca: klasę danego obiektu.

* predict\_all(test\_df, tree):

klasyfikuje obiekty danego zbioru, wykorzystując funkcję *predict*.

Parametry:

test\_df – zbiór danych,

tree – drzewo decyzyjne.

Zwraca: wektor predykcji dla danego zbioru.

* random\_forest(train\_df, forest\_size, sample\_size, no\_of\_attributes):

funkcja tworząca las losowy.

Parametry:

train\_df – zbiór danych,

forest\_size – wielkość lasu,

sample\_size - ilość losowanych obiektów,

no\_of\_attribues – liczba atrybutów wykorzystana do stworzenia

pojedynczego drzewa

* predict\_forest(test\_df, forest):

funkcja wykorzystująca las losowy do klasyfikacji obiektów.

Parametry:

test\_df – zbiór danych,

forest – las losowy.

Zwraca: najczęściej występującą klasę

Plik Main.py zawiera przykładową instrukcję wywołania algorytmu w pętli. Dla stu iteracji uzyskaliśmy średnią poprawność predykcji na poziomie 96%.