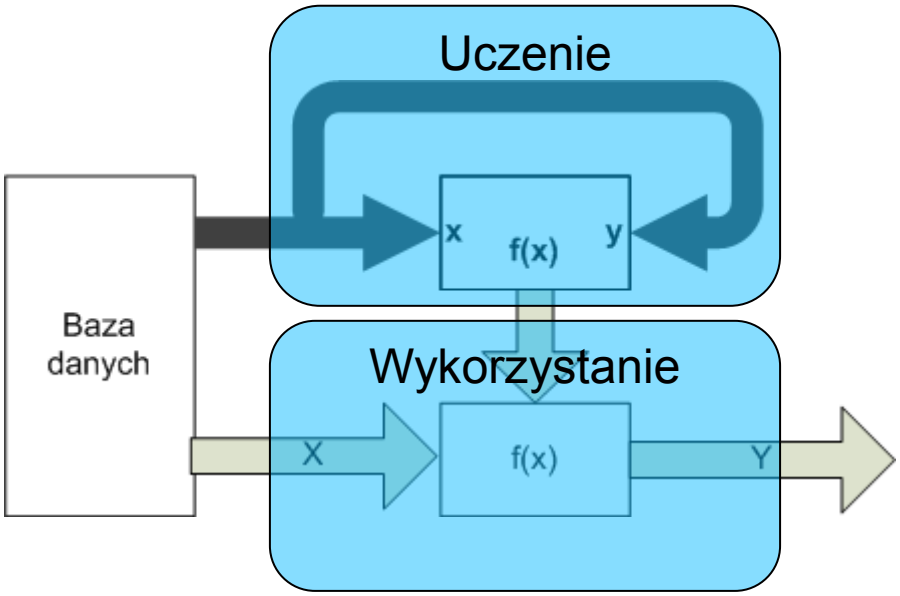
Algorytm najbliższego sąsiada i metody selekcji i edycji danych

Wstęp

Jednym z najprostszych algorytmów posiadających zdolność uczenia się jest algorytm najbliższego sąsiada. Jego cechą charakterystyczną jest niezwykła prostota procesu uczenia. Proces ten sprowadza się do zapamiętani wszystkich przypadków zbioru uczącego, co powoduje że czas uczenia czyli indukcji wiedzy jest niemal zerowy. Podejmowanie przez system decyzji sprowadza się do znalezienia najbliższego wektora w zapamiętanym zbiorze danych uczących w stosunku do wektora dla którego system ma podjąć decyzję.

Schematycznie całkowity proces indukcji wiedzy i późniejszego jej wykorzystania można przedstawić jako:



Gdzie proces uczenia odbywa się na podstawie zbioru **T** = [(**x**$_1$,y_1),(**x**$_2$,y_2)**…,(x**$_n$,y_n)], gdzie (**x**,y) – oznacza parę składającą się z pojedynczego wektora uczącego oraz odpowiadającego mu wyjścia . Wektor **X**$_{n+1}$ jest wektorem dla którego chcemy podjąć decyzję czyli wyznaczyć odpowiadającą mu wartość y_{n+1}

Dla algorytmu najbliższego sąsiada (ang. Nearest neighbor - NN) jak wspominano wyżej polega na zapamiętaniu zbioru **T**, natomiast predykcję można wyrazić wzorem:

$$j = \underset{i=1..n}{argmin}\, D(\boldsymbol{x}_i, \boldsymbol{x}_{n+1})$$

$$y_{n+1} \Leftarrow y_j$$

Dla modelu NN pojawia się jednak problem w przypadku istnienia szumu w danych. Wówczas koniecznym elementem jest kontrola generalizacji czyli stopnia uogólnienia jakie jest w stanie zaoferować algorytm NN.

Typowym rozwiązaniem jest tzw. algorytm kNN, gdzie w zamian z pojedynczy najbliższy wektor szuka się *k* najbliższych wektorów i dokonuje procesu głosowania znalezionych najbliższych sąsiadów. W najprostszej wersji dla problemu regresyjnego wyznacza się średnią z najbliższych sąsiadów, natomiast dla problemu klasyfikacyjnego każdy najbliższy sąsiad używany jest do podjęcia decyzji o przypisaniu odpowiedniej etykiety, a następnie analizuje się jaka etykieta była najczęściej wybierana.

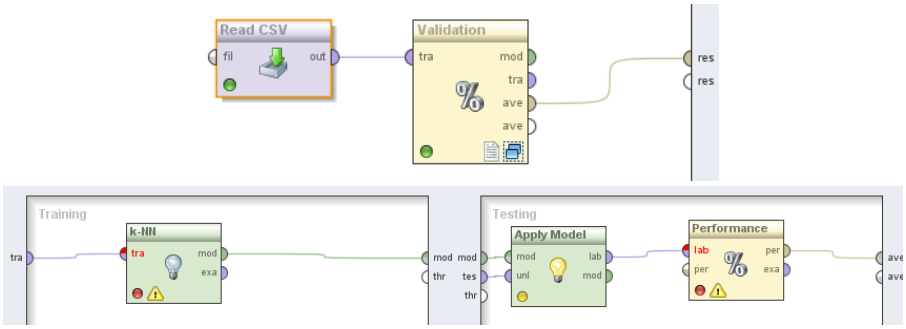
Rozwiązanie to w zależności od wyboru parametru *k* pozwala na kontrolę stopnia generalizacji modelu, tak iż im wyższa wartość *k* tym złożoność modelu maleje, gdzie złożoność modelu rozumiana jest tutaj jako złożoność kształtu granicy decyzji.

W tym przypadku pozostaje jednak problem dużej złożoności obliczeniowej, ponieważ im zbiór danych treningowych będzie rósł, tym również będzie rosła złożoność obliczeniowa, ponieważ chcąc podjąć decyzję dla danego wektora **x,** konieczne jest wyznaczenie odległości do *n* wektorów uczących. Dlatego też alternatywnymi metodami poprawy jakości klasyfikatora kNN jest wstępna selekcja wektorów uczących, tym samym zmniejszając liczbę wektorów uczących zmniejsza się również złożoność obliczeniowa. Do tej grupy algorytmów należą między innymi algorytmy takie jak ENN, RENN, All kNN, CNN.

Oraz algorytmy optymalizacji położenia wektorów uczących takie jak algorytm LVQ.

Zadania do wykonania

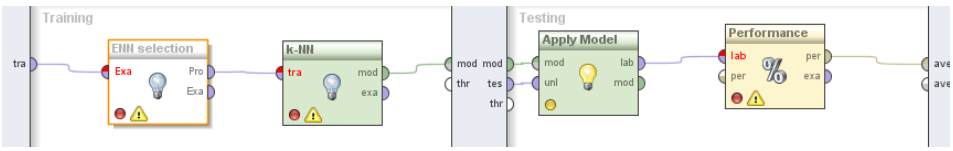
1) Zbuduj model predykcyjny wykorzystujący algorytm kNN i przetestuj wpływ parametru *k* na jakość uzyskanych wyników w teście krzyżowym. Operacje wykonaj zgodnie ze schematami:



Obliczenia wykonaj dla *Zbior5.csv*

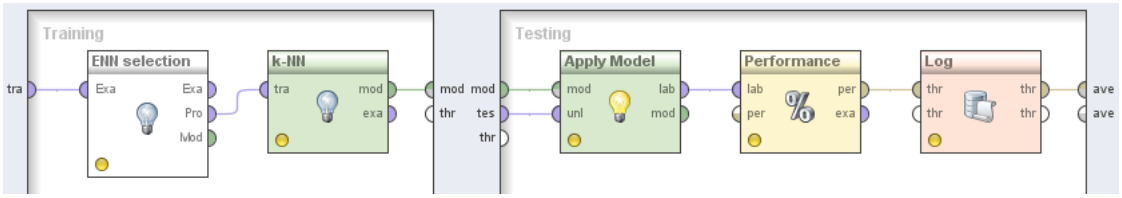
2) Rozszerz zbudowany model o wykorzystanie algorytmów selekcji wektorów prototypowych. W tym celu jeśli nie jest zainstalowany to doinstaluj dodatek Instance Selection and Prototype Based Rules
Obliczenia wykonaj dla *Zbior5.csv*

3) Dodaj przed operatorem k-NN operator selekcji prototypów ENN i zbadaj wpływ wartości parametru *k* algorytmu ENN na jakość uzyskanych wyników oraz liczbę wybranych prototypów. Dla algorytmu *k-NN* ustaw *k=1*

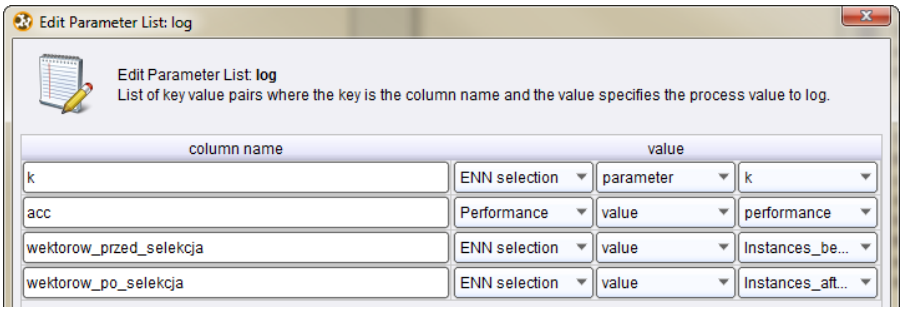


Badając liczbę wybranych prototypów umieść Operator logowania na końcu procesu testowania i loguj: liczbę wektorów przed i po zredukowaniu z wykorzystaniem algorytmu

ENN oraz dokładność (jeśli do obliczeń wykorzystywany jest operator Loop Parameters wówczas loguj również wartość zmienianych parametrów, czyli parametr *k modelu ENN Selection*)



Aby uzyskać wymagany efekt bloczek logowania skonfiguruj jak poniżej:



4) Jako algorytm selekcji zastosuj algorytm IB3 i powtórz operacje jak z zadania 3

> UWAGA poniższe algorytmy nie wymagają konfiguracji, więc wynikiem ich zastosowania będzie jedna para liczb (dokładność i kompresja) dla każdego zbioru danych

5) Jako algorytm selekcji zastosuj algorytm CNN i powtórz operacje jak z zadania 3 jednak zwróć uwagę, iż ten algorytm nie wymaga żadnej konfiguracji!!

6) Jako algorytm selekcji zastosuj algorytm RNG i powtórz operacje jak z zadania 3 jednak zwróć uwagę, iż ten algorytm nie wymaga żadnej konfiguracji!!

7) Jako algorytm selekcji zastosuj algorytm ELH i powtórz operacje jak z zadania 3 jednak zwróć uwagę, iż ten algorytm nie wymaga żadnej konfiguracji!!

8) W sprawozdaniu opisz wpływ parametru *k* dla algorytmów kNN, ENN, IB3 na jakość uzyskanych wyników (wykresy przedstawiające dokładność w funkcji parametru *k* dla różnych zbiorów danych).
   Dla algorytmu kNN sprawdź *k* w zakresie 1:10
   Dla algorytmów ENN i IB3 sprawdź *k* w zakresie 3:10

9) W sprawozdaniu opisz wpływ parametru *k* dla algorytmu ENN na stopień kompresji (wykresy przedstawiające kompresję w funkcji parametru *k*), gdzie kompresja zdefiniowana jest jako

$$\frac{\text{liczba\_przypadków\_po\_selekcji}}{\text{liczba\_przypadków\_przed\_selekcją}} \cdot 100\%$$

10) Dla algorytmów RNG, CNN, ELH wyznacz wartość kompresji

11) Dla algorytmów kNN, ENN, IB3 wybierz najlepszą wartość parametru *k.* Najlepsza wartość parametru k to taka wartość parametru k dla którego uzyskano największą dokładność. Dla tego parametru k odczytaj również kompresję.

12) Na jednym wykresie dla każdego zbioru danych przedstaw zależność dokładności w funkcji kompresji dla wszystkich algorytmów ACC = f(Compression). Dla algorytmów kNN, ENN oraz IB3 zastosuj wyniki z 10.

13) Dla każdego z zadań skomentuj uzyskane wyniki.