# Wykonanie bindingu dla GMUM.R do języka Python, zgodnego z scikit-learn

Karol Jurek
Konrad Talik
Marcin Data

31 maja 2015

## 1 Co znajdzie się w projekcie

Projekt to umożliwienie wywoływania core (C++) biblioteki w Pythonie.

To znaczy, że docelowo chcemy udostępnić w nowym środowisku wszystkie dotychczas osiągalne w R funkcjonalności pakietu GMUM. To jednak kosztuje dużo pracy (np. obsługa możliwych typów, przekazywanych między środowiskami lub inne specyficzne dla nowego środowiska przypadki).

Dodatkowo, pojawiło się dużo wyzwań typowo *koderskich*, takich jak: nauka nowego narzędzia SWIG i sposobu tworzenia interfejsów opakowujących biblioteki dla wielu języków; poprawna rozłączna kompilacja kodu w C++ i automatyczne linkowanie do nowego targetu, z zachowaniem poprzedniego targetu (R).

Z powodu wielkości biblioteki, w ramach tego projektu zrealizowany będzie początek, umożliwiający dalszą pracę w najbliższej przyszłości.

## 2 Opis funkcjonalności

Wymagania funkcjonalne projektu są następujące:

1. Z poziomu Pythona dostępne są następujące funkcjonalności modułu CEC:
   (a) CecConfiguration
      i. Podstawowe metody, umożliwiające nasetowanie konfiguracji oraz metody dostępu
   (b) CecModel
      i. get_energy() - zwraca koszt najlepszej próby algorytmu
      ii. get_assignment() - zwraca najlepszy przydział punktów do klastrów
2. Z poziomu Pythona dostępne są następujące funkcjonalności modułu SVM:
   (a) SVMConfiguration
      i. Podstawowe metody, umożliwiające nasetowanie konfiguracji oraz metody dostępu
   (b) SVMClient
      i. run() - uruchamia proces uczenia lub/i preprocessingu danych
      ii. predict() - uruchamia proces predykcji
3. Odpowiednie klasy bazowe z modułu **base** biblioteki **sklearn** zostaną rozszerzone o nowe klasyfikatory implementowane za pomocą obiektów z biblioteki GMUM. (Jeszcze nie dostępne w wersji ZERO)

Wymagania niefunkcjonalne:

1. Wspólny Makefile dla rozłącznej kompilacji core biblioteki wraz z generowaniem kodów interfejsów SWIG i obiektów dla paczki Pythona, w celu zapewnienia wydajniejszej pracy programistom biblioteki GMUM
2. Pozostałe wymagania projektowe takie jak testy i pylint

# 3 Zarys architektury

Katalog paczki **gmumpy** zawiera następujące pliki: *__init__.py _core.so core.py cec.py svm.py *.py*

Plik *_core.so* jest plikiem binarnym wygenerowanym przez proces kompilacji opakowań SWIG wraz z kodem core biblioteki. Python wykorzystuje go w automatycznie wygenerowanym opakowaniu *core.py*.

Moduły takie jak *cec.py* i *svm.py* to bezpośrednie opakowania obiektów z modułu *core*. W tych modułach wykonywane są wszelakie mechanizmy obsługi konwersji/przygotowywania danych przed przekazywaniem jej do core biblioteki (i w drugą stronę). Moduły te udostępniają klasy o tych samych nazwach, ale wzbogacone o wspomniane opakowania.

Przez **.py* rozumiemy wszelakie moduły, będące odbiciem modułów z biblioteki **sklearn** (takie jak *clustering* czy *datasets*). Będą one dodawane sukcesywnie w trakcie dodawania kolejnych funkcjonalności.

Powyższe moduły zawierać będą klasy, będące rozszerzeniem klas z modułu **sklearn.base**.