1. System ilustrujący działanie wybranych algorytmów optymalizacji.
   Cele:
   - Wizualizacja problemów programowania matematycznego
   - Demonstracja działania algorytmów optymalizacji lokalnej (Greedy, Steepest, Symulowane wyżarzanie, Przeszukiwanie Tabu) oraz optymalizacji ewolucyjnej

   Technologia:
   - Python 3
   - Źródła będą przechowywane w podanym repozytorium (Bitbucket)

   Materiały:
   - Program może działać podobnie do programu OptiVis (http://www.alife.pl/opt/p/index.html)

2. System ilustrujący działanie wybranych algorytmów uczenia maszynowego (**9 projektów**).
   Cele:
   - Stworzenie grupy programów o wspólnym szablonie (interfejsie) ilustrujących wybrane algorytmy uczenia maszynowego.
   - Każdy program może działać niezależnie, ale jest też podłączony do wspólnego “centrum sterowania) - w tym sensie architektura podobna do Weki
   - W stosunku do istniejących http://orange.biolab.si/ i http://www.shogun-toolbox.org/ różnica jest taka, że cel jest edukacyjny (możliwość wykonywania eksperymentów po to, żeby zrozumieć jak “w środku” działają algorytmy wedle opisu ćwiczeń), więc algorytmy muszą udostępniać kilka kluczowych wewnętrznych stanów/zmiennych (które w normalnych, praktycznych zastosowaniach nie są istotne dla użytkowników).

   Technologia:
   - Python 3
   - Źródła będą przechowywane w podanym repozytorium (Bitbucket)

   Każdy projekt składa się z implementacji wybranego algorytmu/algorytmów na podstawie artykułu naukowego opisującego ten algorytm (z ew. pomocą dostępnych źródeł, o ile licencja pozwala i po uzgodnieniu z prowadzącą) oraz stworzenia wizualizacji obrazującej działanie algorytmu. Programy powinny umożliwiać wykonywanie poleceń zawartych w opisach zajęć z linków poniżej; opisy te można uaktualnić i dostosować do nowych implementacji algorytmów.

   Projekty:

1) Implementacja klasy nadrzędnej pozwalającej na skorzystanie z zaimplementowanych algorytmów (wspólny interfejs), oraz obliczającej i wizualizującej wskaźniki jakości klasyfikacji i sposób działania algorytmów.
2) Algorytm eliminacji kandydatów (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-version-spaces).
3) Indukcja drzew decyzyjnych metodami ID3, C4, C4.5 (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-decision-tree-induction)

4) Generowanie drzew decyzyjnych (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-decision-trees)
5) Generowanie reguł decyzyjnych (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-decision-rules)
6) Klasyfikatory Bayes'owskie (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-bayes)
7) Klasyfikatory k-NN i IBL (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-kNN-IBL)
8) Przestrzeń atrybutów i jej transformacje (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-attribute-space)
9) Uczenie nadzorowane sztucznych sieci neuronowych, wersja podstawowa (http://www.cs.put.poznan.pl/mkomosinski/site/?q=ml-neural-networks-supervised-learning)