Metody eksploracji danych
Laboratorium 5

Programowe użycie bibliotek Weka z poziomu języka Java
Klasyfikatory: Naive Bayes, drzewa decyzyjne, SVM (Weka: SMO) i kNN (Weka: IBk)

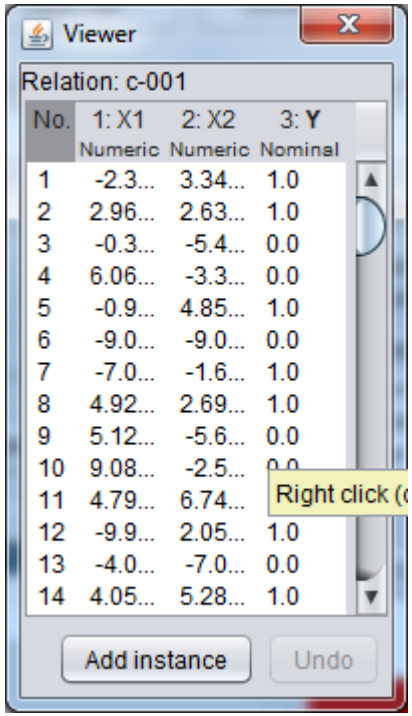
Zasoby

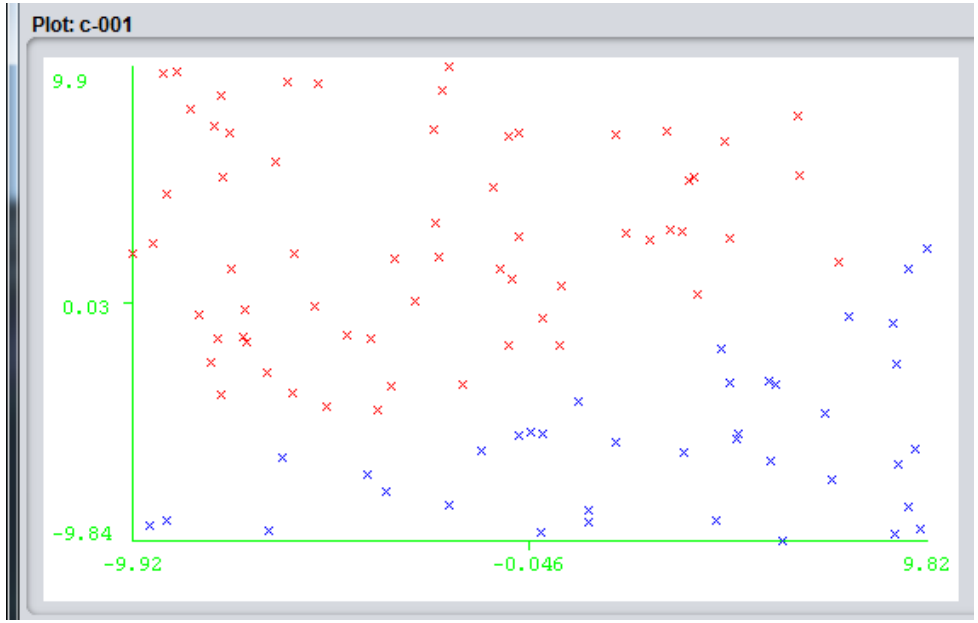
- Podczas ćwiczeń przetwarzali będziemy sztucznie wygenerowane zbiory danych c-xxx.arff
- Weka 3.8 (interfejs użytkownika i biblioteka weka.jar)
- W laboratorium jest zainstalowane oprogramowanie Eclipse Neon. Należy utworzyć projekt i dodać weka.jar jako bibliotekę używaną w projekcie
- Dokładny opis wykorzystywanych klas biblioteki znajduje się pod adresem: http://weka.sourceforge.net/doc.stable/ (ale czasem szybciej można znaleźć wpisując „Weka nazwa klasy”
- Warto też poszukać informacji pod adresem:

https://weka.wikispaces.com/Use+WEKA+in+your+Java+code

5.1 Załadowanie zbioru uczącego i klasyfikacja

- Przetwarzamy **c-001.arff**
- Uruchom Weka i otwórz plik w eksploratorze. Kolejne pliki będą miały podobną postać.





Kod do zaimplementowania

Zbiór danych jest reprezentowany przez obiekt klasy `Instances`. Przechowuje on informacje o:

- atrybutach (kolumnach tabeli), w tym etykiecie klasy
- Instancjach/obserwacjach (wierszach tabeli). Te z kolei są reprezentowane przez typ `Instance (DenseInstance, SparseInstance)`

- 5.1.1 Załaduj plik.

```
DataSource source = new DataSource("c-001.arff");
Instances data = source.getDataSet();
```

- 5.1.2 Ustaw, który atrybut ma być użyty jako etykieta klasy

```
if (data.classIndex() == -1)
data.setClassIndex(data.numAttributes()-1);
```

- 5.1.3 Utwórz i naucz klasyfikator

```
Classifier cls = new NaiveBayes();
cls.buildClassifier(data);
```

Wykorzystaj zbudowany model

- 5.1.4 Utwórz instancję (obserwację) do sklasyfikowania

```
Instance inst = new DenseInstance(3);
inst.setDataset(data);
inst.setValue(0, 1.1); // wartośc dla X1
inst.setValue(1, 2.2); // wartość dla X2
```

- 5.1.5 Sklasyfikuj instancję

```
double y = cls.classifyInstance(inst);
```

- 5.1.6 Można także wyznaczyć wartości prawdopodobieństwa

```
double[] distrib = cls.distributionForInstance(inst);
System.out.printf(Locale.US, "%d->%f %d->%f\n" ,0,
distrib[0], 1, distrib[1]);
```

Poeksperymentuj z innymi wartościami x1 i x2 …

5.2 Klasyfikacja wygenerowanego zbioru instancji

- 5.2.1 Napisz nową funkcję i powtórz kroki 5.1.1-5.1.3
- 5.2.2 Utwórz wyjściowy zbiór danych

```
List<Attribute> atts = Arrays.asList(
 new Attribute("X1"),
 new Attribute("X2"),
 new Attribute("Y", Arrays.asList("tak","nie")));

Instances result = new Instances("some-relation", new
ArrayList<> (atts),0);
result.setClassIndex(result.numAttributes()-1);
```

Klasyfikacja wygenerowanego zbioru instancji

- 5.2.3 Utwórz instancje w pętli, sklasyfikuj, dodaj do zbioru

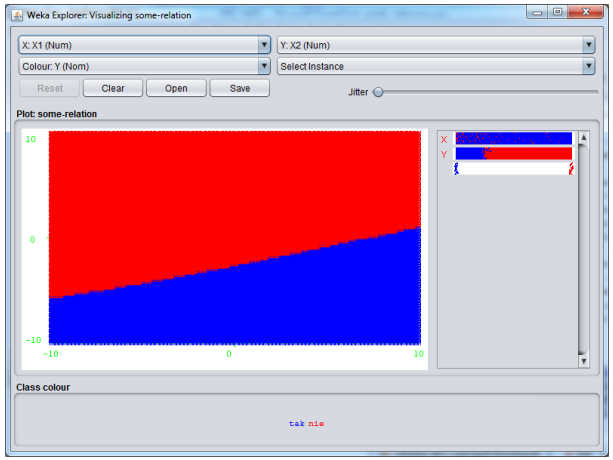
```
for(double x1=-10;x1<=10;x1+=0.1){
    for(double x2=-10;x2<=10;x2+=0.1){
       Instance inst = new DenseInstance(3);
       inst.setValue(0, x1);
       inst.setValue(1, x2);

       inst.setDataset(result);
       double y = cls.classifyInstance(inst);
       inst.setClassValue(y);
       result.add(inst);
    }
}
```

Zapis

- 5.2.4 Zapisz do pliku…

```
ArffSaver saver = new ArffSaver();
saver.setInstances(result);
saver.setFile(new File("c-001-result.arff"));
saver.writeBatch();
```

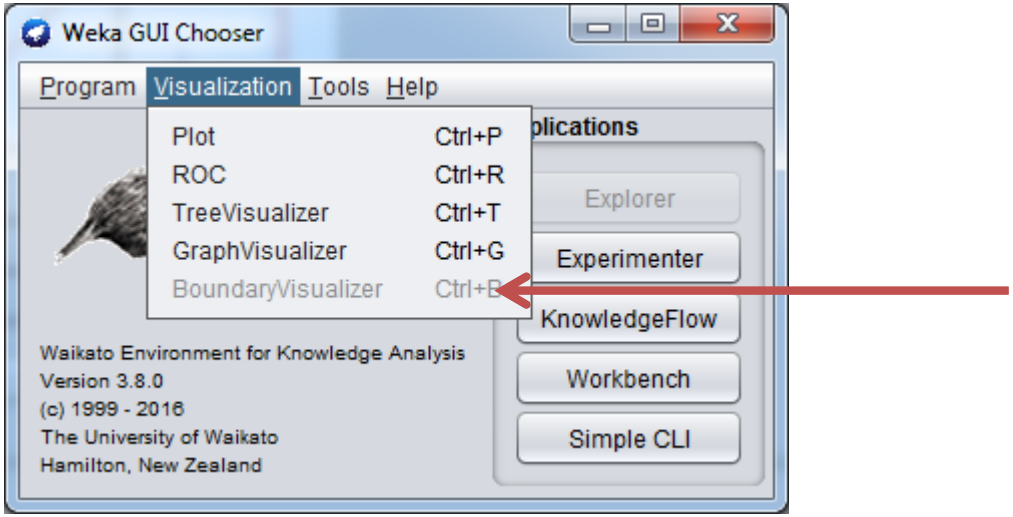
5.2.4 Załaduj do eksploratora Weka i wyświetl…



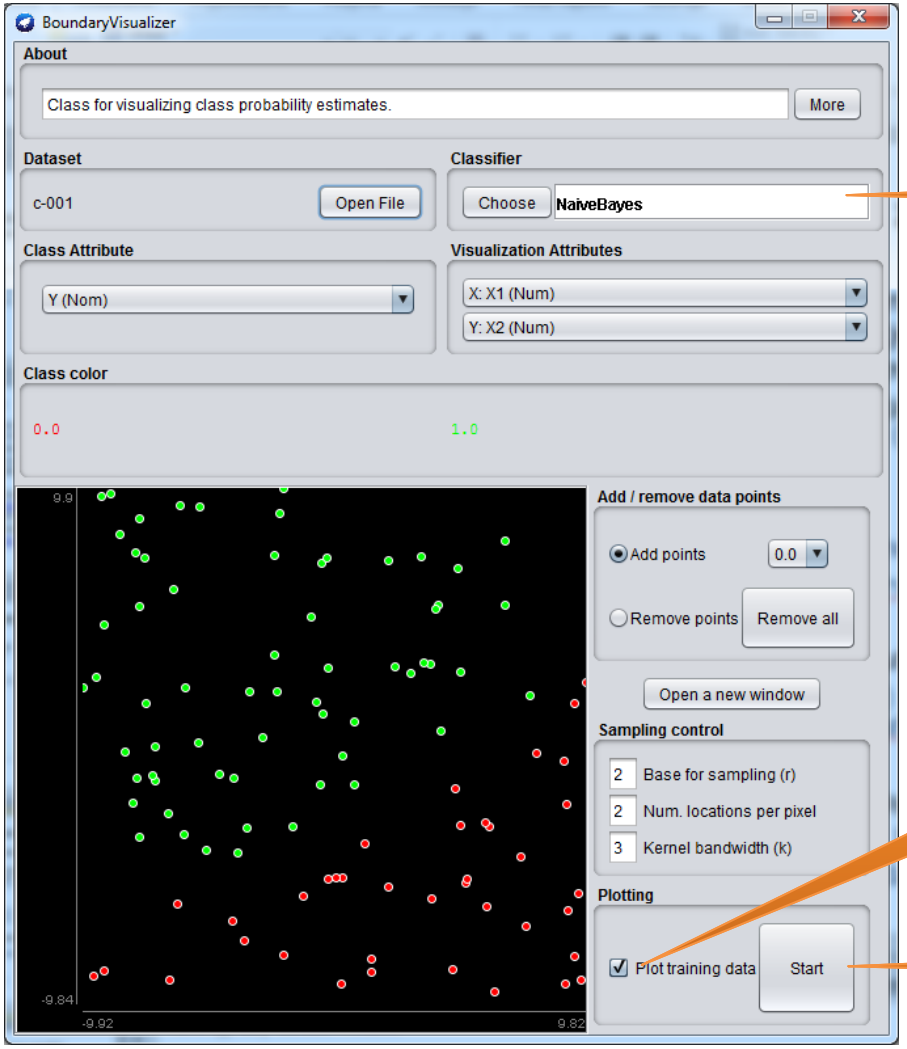
Przykłady granic klas i regionów decyzyjnych – w tekście wykładów…

5.3 Wypróbuj gotowe rozwiązanie

- W oknie głównym Weka wybierz BoundaryVisualizer



Boundary Visuzalizer



Wybierz NaiveBayes

Zaznacz tę opcję

Naciśnij Start

5.4 Ocena jakości klasyfikatora

- Zastosujemy 10-krotną walidację krzyżową i wydrukujemy wyniki
- 5.4.1 Powtórz sekwencję ładowania pliku
- 5.4.2 Utwórz klasyfikator i przeprowadź proces walidacji modelu

```
Classifier cls;
cls = new NaiveBayes();

Evaluation eval = new Evaluation(data);
eval.crossValidateModel(cls, data, 10, new Random(1));
```

Ocena jakości klasyfikatora

Wydrukuj wyniki:

```
eval.toSummaryString()
eval.confusionMatrix()


System.out.printf(Locale.US,
"[prec recall fmeasure]:\t%f\t%f\t%f\n",
eval.weightedPrecision(),
eval.weightedRecall(),
eval.weightedFMeasure());
```

5.5 Oceń jakość kolejnych klasyfiaktorów

- 5.5.1
  a. W BoundaryVisualizer wybierz drzewo decyzyjne J48
  b. Wpisz w kodzie `cls = new J48();` i zapisz wyniki precision/recall/fmeasure
- 5.5.2 Przeprowadź to samo dla Support Vector Machines (SMO)
- 5.5.3 Dla k-NN: `cls = new IBk();` lub `cls = new IBk(5);`
  Parametr k to liczba najbliższych sąsiadów branych pod uwagę

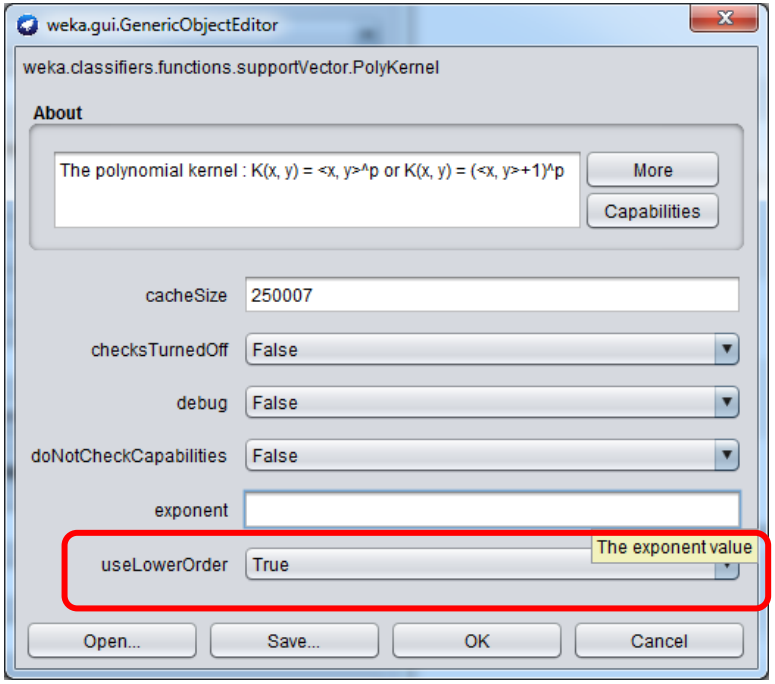
5.6 Przetwarzamy c-002.arff

- 5.6.1 W BoundaryView uruchom NaiveBayes, **równocześnie** (dla zaoszczędzenia czasu) wybierz ten klasyfikator w kodzie, uruchom i zbierz wyniki.
- 5.6.2 To samo dla J48
- 5.6.3 Wykonaj to samo dla Ibk
- Zamieść w spawozdaniu obrazki z regionami decyzyjnymi

5.7 Stosujemy SMO

- Wybierz SMO w BoundaryView i obejrzyj wyniki… (nie powinny być najlepsze)
- Uruchom kod z wybranych klasyfikatorem SMO.
- Na podstawie **wykładu 6** – ten klasyfikator potrafi jednak dobrze odtworzyć regiony decyzyjne, ale trzeba wybrać inny kernel.

- W oknie BoundaryView peksperymentuj z parametrem **Exponent** dla PolyKernel. Wybierz useLowerOrder=true.
- Nie musisz czekać na zakończenie rysowania, przerwij, jeśli wyniki są złe.
- **Exponent** to stopień wielomianu



5.8 Przenieśmy te opcje do kodu

- Utwórz klasyfikator i ustaw parametry kernela

```
SMO cls = new SMO();
double exp = 1; // to nie jest najlepsza wartość
cls.setKernel(new PolyKernel(data,1000,exp,true));
```

Uruchom dla kilku wartości `exp` (np. ciąg Fibbonacciego) i oceń wyniki

5.9 Kernel RBF

- O kernelu RBF przeczytasz w tekście **wykładu 6** (około strony 20).
- W Weka parametrem jest współczynnik gamma. Odpowiada on $\frac{1}{2\sigma^2}$ czynnikowi. W przypadku tych zbiorów danych raczej należy użyć wartości gamma > 1.
- Wypróbuj w Boundary View
- Wprowadź do kodu. Poeksperymentuj z kilkoma wartościami gamma, np. 0.01,0.1, 1, 10, 100. Potem można przeprowadzić optymalizację

```
double gamma = 0.01; // to nie jest najlepsza wartość
cls.setKernel(new RBFKernel(data, 0, gamma));
```

5.10 Plik c-003.arff

Wypełnij tabelkę i pokaż wizualizację regionów decyzyjnych dla co najmniej dwóch wybranych metod
(np. J48+ SMO/NB/Ibk)
Podaj w tabelce najlepsze parametry

Klasyfikator	Parametry	Precision	Recall	F measure
NaiveBayes				
J48				
IBk	k=5			
SMO+PolyKernel	Exp=			
SMO+RBFKernel	Gamma=			

5.11 Plik c-004.arff

Wypełnij tabelkę i pokaż wizualizację wszystkich regionów decyzyjnych dla
J48, SMO, NB, Ibk
Podaj w tabelce najlepsze parametry

Klasyfikator	Parametry	Precision	Recall	F measure
NaiveBayes				
J48				
IBk	k=5			
SMO+PolyKernel	Exp=			
SMO+RBFKernel	Gamma=			

5.15 Plik c-005.arff

Podaj w tabelce najlepsze parametry/wyniki
Dla porównania obejrzyj regiony decyzyjne.

Klasyfikator	Parametry	Precision	Recall	F measure
NaiveBayes				
J48				
IBk	k=5			
SMO+PolyKernel	Exp=			
SMO+RBFKernel	Gamma=			