

Metody eksploracji danych

Laboratorium 3

Regresja logistyczna

Regresja logistyczna

- Regresja logistyczna jest metodą klasyfikacji (patrz **wykład 4**)
 - Oczekiwane jest, że atrybut wyjściowy jest etykietą klasy (a nie liczbą)
 - Atrybuty wejściowe muszą być numeryczne
- Podczas zajęć będziemy przetwarzali plik zawierający rzeczywiste dane obejmujące:
 - ocenę i datę zaliczenia z języka C (co najmniej 3.0)
 - ocenę pierwszego zaliczenia z języka C++
 - ocenę z I terminu egzaminu (można przystąpić bez zaliczenia)
- Interesuje nas wpływ ocen z zaliczeń na wynik egzaminu w pierwszym terminie (przekształcony do problemu klasyfikacji binarnej):
 - nie zdał (2.0)
 - zdał (3.0-5.0)

No.	1: ImieNazwisko String	2: OcenaC Numeric	3: DataC Date	4: OcenaCpp Numeric	5: Egzamin Numeric
1	Dqhoil Dhxpluj	3.5	2016-01-14	4.0	3.0
2	Bhnhgpxj Lwjmq	4.5	2016-01-14	4.0	3.0
3	Wkgjnerme Djfbw	4.0	2016-01-20	3.5	2.0
4	Sredmuwt Tcimknl	4.5	2016-01-20	4.5	3.5
5	Tiowe Bqoilnqbrx	4.0	2016-01-14	4.5	3.0
6	Bvaysqv Wuyih	3.5	2016-01-14	5.0	3.0
7	Jjoaxp Ktapcy	5.0	2016-01-20	4.0	3.5
8	Mkengbtw Aainhh	3.5	2016-01-20	3.5	2.0
9	Fbfffj Muupwshu	4.0	2016-01-14	5.0	4.0

3.1 Przygotowanie danych

- Pobierz pliki
 - egzamin-cpp.csv
 - egzamin-cpp-train.csv
 - egzamin-cpp-test.csv

Dwa ostatnie powstały z podziału egzamin-cpp na dwa

- Przekształć je do postaci ARFF

Najlepiej w tym celu użyć polecenia w konsoli:

```
java -cp ..\weka.jar weka.core.converters.CSVLoader [opcje]  
egzamin-cpp-train.csv > egzamin-cpp-train.arff
```

Opcje dostępne na stronie:

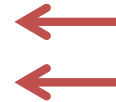
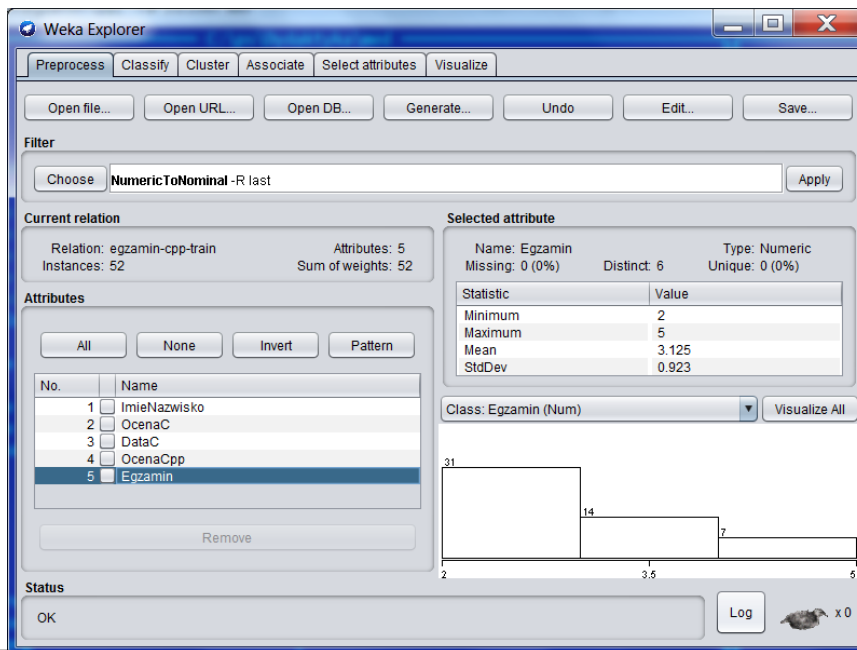
<http://weka.sourceforge.net/doc.dev/weka/core/converters/CSVLoader.html>

Należy

- wskazać separator
- podać, który atrybut jest typu String
- który atrybut jest datą i jaki jest jej format (-format "yyyy-mm-dd")
- dla pliku egzamin-cpp-test.csv użyj opcji -N 5, aby wskazać, że ostatni atrybut jest nominalny (są tam puste wartości)

3.2 Konwersja danych

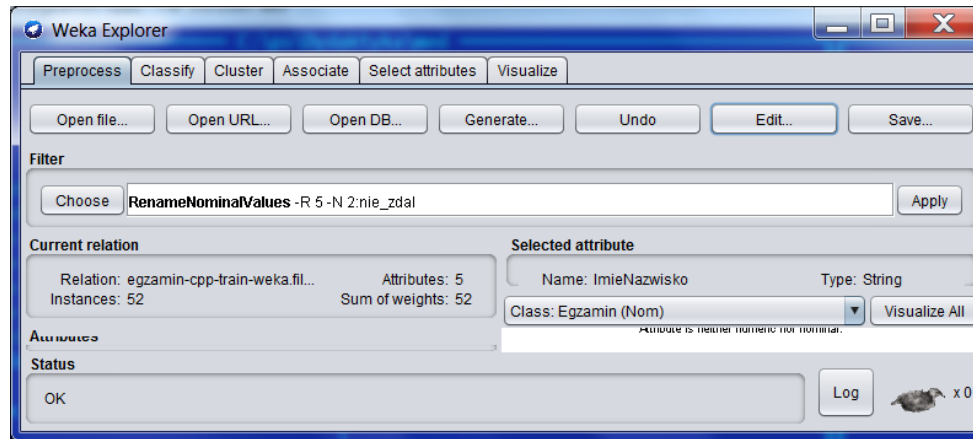
- Konwersja stosuje się do egzamin-cpp.arff i egzamin-cpp-train.arff.
- Jest ona kilkietapowa i można ją przeprowadzić w :
 - Weka Explorer wybierając kolejno stosowne filtry i klikając Apply. Na wszelki wypadek należy zapisywać pliki pośrednie.
 - Można też skonstruować knowledge flow (**zalecane**, bo łatwiej poprawić błędy i zastosować do dwóch plików).
- Etap 1: NumericToNominal



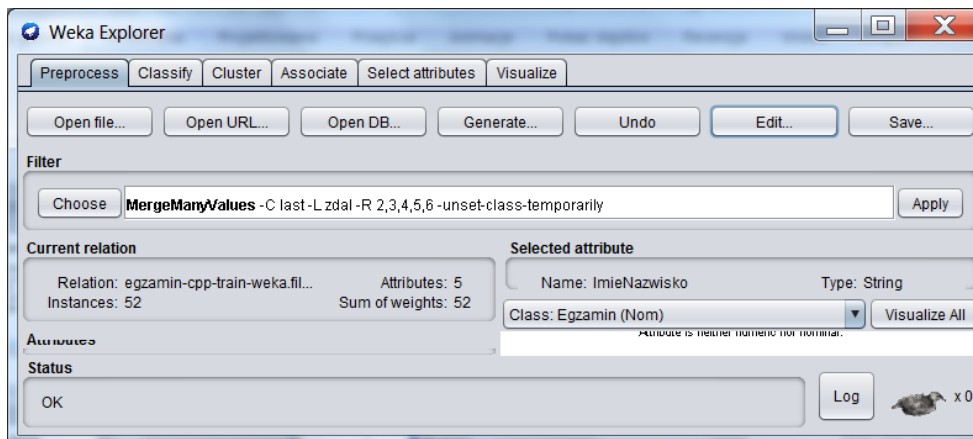
Użyj Apply i zapisz

Konwersja danych

- Etap 2: RenameNominalValues 2-> nie_zdal



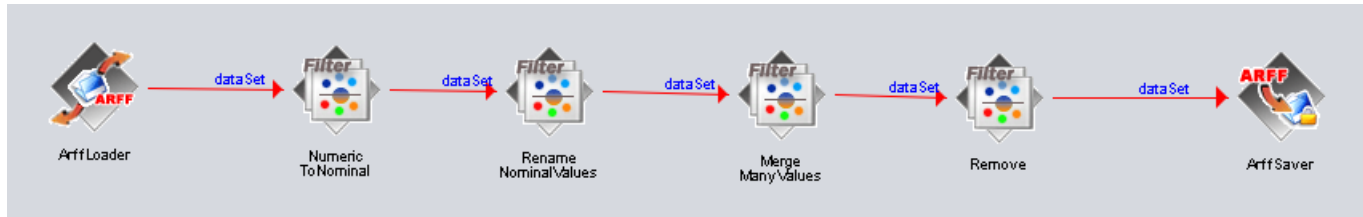
- Etap 3: MergeManyValues {3,3.5,4,4.5,5}-> zdal



Znikający przecinek?:
wprowadź {2,3,4,5,6} a
potem usuń {}

Jako alternatywa: workflow KW1

- Pamiętaj, aby uruchomić Weka korzystając z Java Oracle 1.7



- Opcje dokładnie te same
- Założmy, że w ArffSaver ustawiony jest prefiks „preprocessed”:
Pliki wyjściowe będą miały nazwy typu:
 - preprocessed-egzamin-cpp-[].arff
 - preprocessed-egzamin-cpp-train[].arff

3.2 Klasyfikator

- Otwórz plik preprocessed-egzamin-cpp- [...].arff
- Wybierz w zakładce Classify regresję logistyczną i uruchom
- Jak zinterpretujesz wyniki? Patrz **wykład 4**
 1. Podaj wzór na hiperpłaszczyznę separującą dane
 2. Podaj o ile wzrost/spadek ocen wpływa na szanse zdania/niezdania egzaminu
 3. Jak wpływa na egzamin zmiana daty wpisu zaliczenia?
 4. Zinterpretuj wyniki klasyfikacji.
 5. Porównaj wyniki testów z użyciem zbioru uczącego i walidacji krzyżowej

[<https://weka.wikispaces.com/Primer>]

The *True Positive (TP)* rate is the proportion of examples which were classified as class x , among all examples which truly have class x , i.e., how much of the class was captured correctly. It is equivalent to *Recall*. In the confusion matrix, this is the diagonal element divided by the sum over the relevant row.

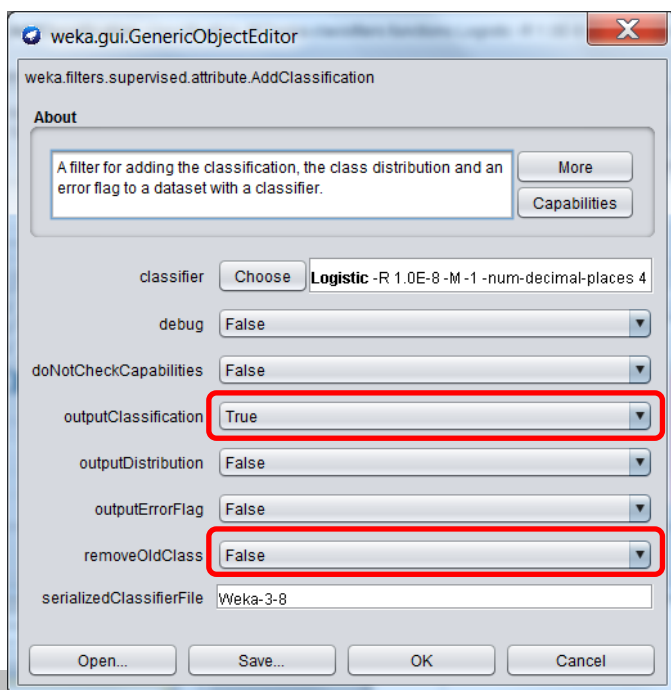
The *False Positive (FP)* rate is the proportion of examples which were classified as class x , but belong to a different class, among all examples which are not of class x . [**Element poza przekątną podzielony przez wiersz. FP i TP sumują się po przekątnych....**]

The *Precision* is the proportion of the examples which truly have class x among all those which were classified as class x .

The *F-Measure* is simply $2 * Precision * Recall / (Precision + Recall)$, a combined measure for precision and recall.

3.3 Porównaj jawnie wyniki klasyfikacji

- W Weka Explorer otwórz przetworzony plik preprocessed-egzamin-cpp- [...].arff
- W Preprocess wybierz filtr AddClassification
- Jako classifier wybierz Logistic ze standardowymi opcjami
- Zwróć uwagę na opcje filtra:



Viewer

Relation: egzamin-cpp-weka.filters.unsupervised.attribute.NumericToNominal-RI...

No. 1: OcenaC 2: Data 3: OcenaCpp 4: Egzamin 5: **classification**

	Numeric	Date	Numeric	Nominal	Nominal
1	3.5	2016-...	4.0	zdal	zdal
2	4.5	2016-...	4.0	zdal	zdal
3	4.0	2016-...	3.0	nie_zdal	nie_zdal
4	4.5	2016-...	4.5	zdal	zdal
5	4.0	2016-...	4.5	zdal	zdal
6	3.5	2016-...	5.0	zdal	zdal
7	5.0	2016-...	4.0	zdal	zdal
8	3.5	2016-...	3.0	nie_zdal	nie_zdal
9	4.0	2016-...	5.0	zdal	zdal
10	5.0	2016-...	4.5	zdal	zdal
11	5.0	2016-...	3.0	zdal	zdal
12	5.0	2016-...	3.5	zdal	zdal
13	4.5	2016-...	3.5	zdal	zdal
14	3.5	2016-...	3.0	nie_zdal	nie_zdal
15	5.0	2016-...	3.0	nie_zdal	zdal
16	4.5	2016-...	5.0	zdal	zdal
17	3.0	2016-...	3.0	nie_zdal	nie_zdal
18	4.5	2016-...	4.0	zdal	zdal

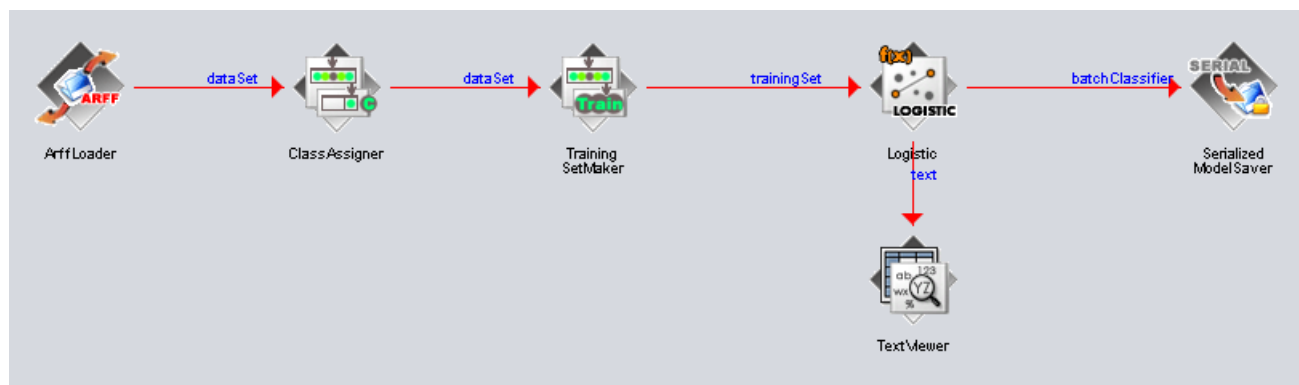
Add instance Undo OK Cancel

3.4 Klasyfikacja w Knowledge Flow

- Zbudowany zostanie model regresji logistycznej na podstawie **preprocessed-egzamin-cpp-train[...].arff**
- Model zostanie zapisany
- W drugim procesie model zostanie załadowany i zastosowany do **egzamin-cpp-test.arff**

Klasyfikacja w Knowledge Flow

- Zbuduj następujący workflow KW2

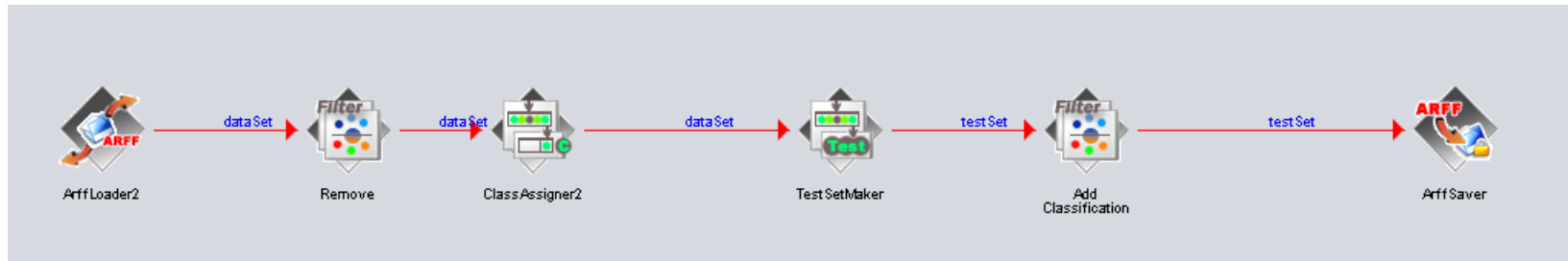


1. Załaduj plik **preprocessed-egzamin-cpp-train[...].arff**
2. Jako klasę wyjściową wybierz Egzamin
3. TrainingSetMaker i Logistic – opcje standardowe
4. Zapisz model dodając prefiks, np. **egzamin-cpp**

Model to zbudowany klasyfikator, w tym przypadku wagi dla równania $w^T [ocenaC, dataC, ocenaCpp] = 0$

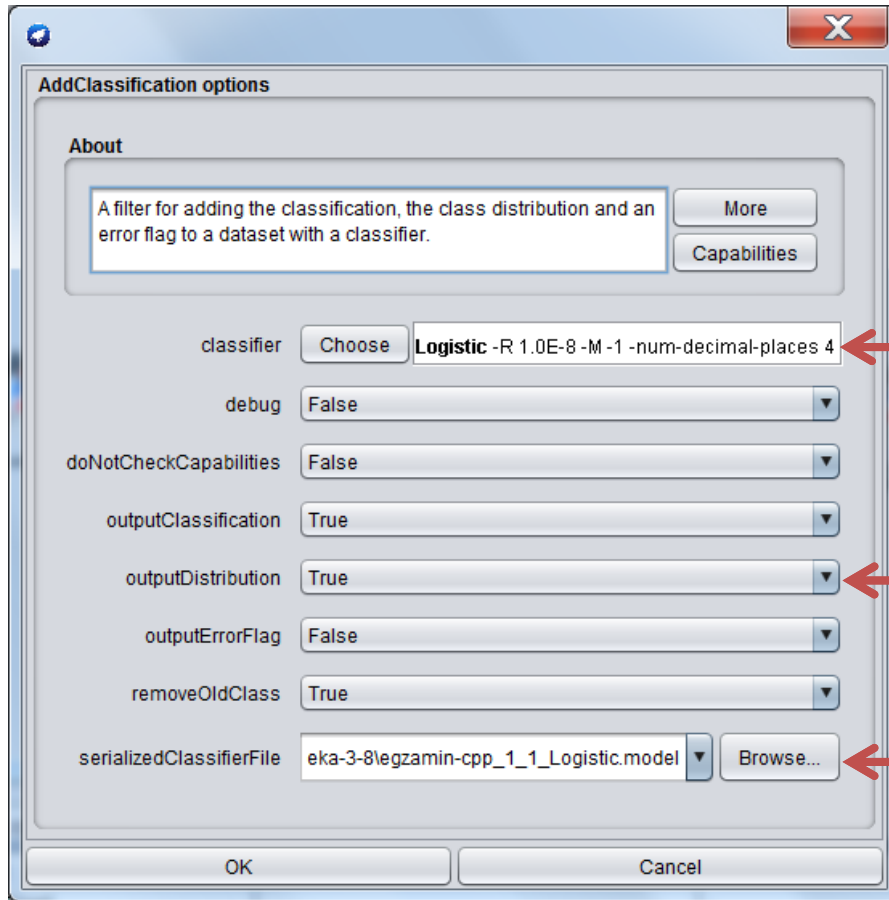
Przetwarzamy plik testowy

- Zbuduj następujący workflow KW3



1. Załaduj plik **egzamin-cpp-test.arff**
2. Usuń pierwszy atrybut
3. Jako atrybut klasy wybierz Egzamin
4. Skonfiguruj AddClassification (**patrz następna strona**)
5. Wybierz nazwę pliku wyjściowego dla ArffSaver
6. Uruchom i obejrzyj wyniki

Konfiguracja AddClassification



Wybierz Logistic, ale jeśli załadowany jest model to raczej nie ma znaczenia

Dodane zostaną prawdopodobieństwa nie_zdal/zdal

Wyberz wcześniej zapisany model

Jeżeli w trakcie wykonania pojawiły się błędy (wyjątki) typu „unary class”:
W pliku arff zamień @attribute Egzamin {*unknown*} na
@attribute Egzamin {nie_zdal,zdal}

Testy wszystkich wartości

Tak naprawdę różnych wariacji ocen wejściowych jest niewiele. Można sprawdzić je **wszystkie**.

Pobierz plik **grid.arff** Jest to wygenerowany plik z wszystkimi kombinacjami ocen (w tym 2.0 dla pierwszego zaliczenia z C++)

3.5 Test modelu na podstawie zbioru egzamin-cpp-train

- Skonfiguruj workflow KW3 tak, aby czytać z grid.arff
- Uruchom i sprawdź wyniki (nie są szczególnie wyraziste, praktycznie wszyscy zdają).
- Oceń wyznaczone przez klasyfikator prawdopodobieństwa

3.6 Test modelu na podstawie zbioru egzamin-cpp

- Skonfiguruj workflow KW2 tak, aby model był jednak generowany na podstawie **preprocessed-egzamin-cpp[].arff** (lepsze pokrycie)
- Uruchom KW3 (skonfigurowany wcześniej do odczytu grid.arff) i porównaj wyniki
- Zestaw wyniki w postaci tabelki (OcenaC, OcenaCpp, wynik egzaminu, pojedyncza [większa] wartość prawdopodobieństwa)