Statystyka Biomedyczna

Instrukcja nr 8

Klasyfikacja danych

Celem laboratorium jest zapoznanie się z wybranymi algorytmami klasyfikacji danych oraz z ich implementacją w bibliotece `scikit-learn` (http://scikit-learn.org/stable/). Materiałem do pracy jest zbiór danych *Chronic Kidney Disease* z repozytorium UCI, który zawiera dane o 400 pacjentach chorujących na przewlekłą niewydolność nerek. Cechami są tutaj różne parametry morfologiczne krwi oraz szereg danych nominalnych informujących o zdiagnozowanych chorobach (anemia, cukrzyca itp.).

1. Pobierz z serwera i wczytaj zbiór danych *Chronic Kidney Disease* (CKD) - oddzielnie plik z wektorami cech (ckd_data.npy) oraz plik zawierający etykiety kategorii (ckd_labels.npy). Należy ograniczyć zbiór wczytanych danych do cech numerycznych (o indeksach od 9 do 17).

2. Utwórz obiekt reprezentujący klasyfikator *Naive Bayes*. Za jego pomocą wykonaj klasyfikację zbioru danych CKD. Uczenie i testowanie klasyfikatora w wykonaj w trybie 10-krotnej walidacji krzyżowej. Błędy klasyfkacji dla poszczególnych foldów (podzbiorów) zapamiętaj w tablicy *errors_NB*.

3. Powtórz uczenie dla klasyfikatora *najbliższego sąsiada* ($k = 1$). Powtórz procedurę uczenia i testowania w trybie 10-krotnej walidacji krzyżowej. Błędy klasyfikacji zapamiętaj w tablicy *errors_NN*.

4. Wykonaj sparowany test Studenta i zweryfikuj hipotezę zerową o tym, że oba klasyfikatory są równie skuteczne. Przyjmij poziom ufności = 95%.

5. Wykonaj wykresy pudełkowe dla obu zmierzonych wartości błędów.

6. Podziel zbiór danych na część treningową (60%) i testową (40%). Przeprowadź binaryzację wektora etykiet katogorii - przyjmij, że klasa '*notckd'* = 0, zaś '*ckd'* = 1. Wykonaj uczenie klasyfikatora *Naive Bayes* z użyciem części treningowej, a następnie przeprowadź testowanie na części testowej z określeniem prawdopodobieństwa przynależności do jednej z klas. Na podstawie otrzymanych wyników wykreśl krzywą ROC.

Przydatne funkcje biblioteki scikit-learn:

Tworzenie klasyfikatora Naive Bayes:

```
gnb = GaussianNB()
```

Funkcja `GaussianNB` znajduje się w pakiecie `sklearn.naive_bayes`

Tworzenie klasyfikatora Naive Bayes:

```
neigh = KNeighborsClassifier(n_neighbors=1)
```

Funkcja `KNeighborsClassifier` znajduje się w pakiecie `sklearn.neighbors`

Binaryzacja wektora etykiet klas:

```
labels = label_binarize(targets, classes=['notckd', 'ckd'])
```

Funkcja `label_binarize` znajduje się w pakiecie `sklearn.preprocessing`

Podział zbioru danych na część treningową i uczącą:

```
X_train, X_test, y_train, y_test = train_test_split()
```

Funkcja `train_test_split` znajduje się w pakiecie `sklearn.cross_validation`
Funkcja `train_test_split` wymaga podania 3 argumentów: wektory cech, etykiety klas, procent zbioru danych przeznaczony na zbiór testowy. Wektor etykiet klas należy podać w postaci tablicy 1-wymiarowej. Do tego celu może być przydatna funkcja `ravel()`.

Podział zbioru danych na foldy:

```
skf = StratifiedKFold(labels, k)
```

Funkcja `StratifiedKFold` znajduje się w pakiecie `sklearn.cross_validation`

Iterację po foldach wykonuje się następująco:

```
for train, test in skf:
    X_train = data[train]
    X_test = data[test]
```

Uczenie klasyfikatora:

```
gnb.fit(X_train, y_train)
```

Testowanie klasyfikatora:

```
cls = gnb.predict(X_test)
```

Wyznaczanie poprawności klasyfikatora na zbiorze testowym:

```
accuracy = gnb.score(X_test, y_test)
```

Wyznaczanie prawdopodobieństwa przynależności do klas:

```
prob = gnb.predict_proba(X_test)
```

Wykreślanie krzywej ROC:

```
fpr, tpr, _ = roc_curve(y_test, prob[:,1]))
```

Funkcja `roc_curve` znajduje się w pakiecie `sklearn.metrics`