Klasyfikacja i regresja: Implementacja naiwnego klasyfikatora bayesowskiego w `scikit-learn`

31 października 2018

Opis pliku z zadaniami

Wszystkie zadania na zajęciach będą przekazywane w postaci plików `.pdf`, sformatowanych podobnie do tego dokumentu. Zadania będą różnego rodzaju. Za każdym razem będą one odpowiednio oznaczone:

- Zadania do wykonania na zajęciach oznaczone są symbolem $\triangle$ – nie są one punktowane, ale należy je wykonać w czasie zajęć.
- Punktowane zadania do wykonania na zajęciach oznaczone są symbolem $\diamond$ – należy je wykonać na zajęciach i zaprezentować prowadzącemu.
- Zadania do wykonania w domu oznaczone są symbolem $\star$ – są one punktowane, należy je dostarczyć w sposób podany przez prowadzącego i w wyznaczonym terminie (zwykle przed kolejnymi zajęciami).

1 Transformacja danych do formatu wspieranego przez `sklearn` △

Treść

Wykonaj poniższe kroki w celu odpowiedniego wczytania i przekształcenia danych:

1. Zapoznaj się z typami danych obsługiwanymi przez `sklearn` oraz proponowanymi metodami wczytywania danych (http://scikit-learn.org/stable/datasets/index.html#external-datasets).

2. Zapoznaj się z podstawowymi informacjami o pakietach `numpy` (http://www.numpy.org/) oraz `pandas` (https://pandas.pydata.org/).

3. Wczytaj dane z plików `.csv` dostępnych na stronie:

   - `data/grypa-train.csv`,
   - `data/grypa-test.csv`),

   za pomocą funkcji z pakietu `pandas` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html#pandas.read_csv).

   Poniżej podany jest spodziewany wynik:

```
print(data_train)

#  dreszcze katar bol_glowy goraczka grypa
#0      tak   nie    sredni      tak   nie
#1      tak   tak       nie      nie   tak
#2      tak   nie      duzy      tak   tak
#3      nie   tak    sredni      tak   tak
#4      nie   nie       nie      nie   nie
#5      nie   tak      duzy      tak   tak
#6      nie   tak      duzy      nie   nie
#7      tak   tak    sredni      tak   tak


print(data_test)

#  dreszcze katar bol_glowy goraczka
#0      nie   tak       nie      tak
```

4. Przekształć wartości nominalne na liczby całkowite:

   Poniżej podany jest spodziewany wynik:

```
print(data_train_converted)

#    dreszcze  katar  bol_glowy  goraczka  grypa
#0          1      0    sredni         1      0
#1          1      1       nie         0      1
#2          1      0      duzy         1      1
#3          0      1    sredni         1      1
#4          0      0       nie         0      0
#5          0      1      duzy         1      1
#6          0      1      duzy         0      0
#7          1      1    sredni         1      1


print(data_test_converted)

#    dreszcze  katar  bol_glowy  goraczka
#0          0      1        nie         1
```

5. Podziel wczytane dane na cechy oraz klasę, a następnie przekształć dane z obiektów `pandas` do macierzy `numpy`.

   Poniżej podany jest spodziewany wynik:

```
print(X_train)

#[[1  0  1  1]
# [1  1  0  0]
# [1  0  2  1]
# [0  1  1  1]
# [0  0  0  0]
# [0  1  2  1]
# [0  1  2  0]
# [1  1  1  1]]

print(y_train)

#[0  1  1  1  0  1  0  1]

print(X_test)

#[[0  1  0  1]]
```

6. Zaimportuj dowolny klasyfikator z pakietu `sklearn` (może to być na przykład algorytm drzew decyzyjnych `DecisionTreeClassifier`). Wytrenuj model na danych treningowych, a następnie porównaj jego działanie na danych treningowych i testowych. Wypisz predykcje modelu dla obu zbiorów danych. Przeanalizuj odpowiedź klasyfikatora. W jakim stopniu predykcja jest poprawna?

   Poniżej podany jest spodziewany wynik:

```
# Create and train a DecisionTreeClassifier
dt = DecisionTreeClassifier()
clr = dt.fit(X_train, y_train)

# Predict the test example
ypred = clr.predict(X_test)
print(ypred)
#[1]

# Predict the train examples
ypred = clr.predict(X_train)
print(ypred)
# [0 1 1 1 0 1 0 1]

# Average accuracy on the training set (known decisions)
accuracy_score(y_train, ypred)
#1.0
```

2 Implementacja naiwnego klasyfikatora bayesowskiego dla danych nominalnych *0p.⋆*

Treść

Naiwny klasyfikator Bayesa jest prostym klasyfikatorem probabilistycznym, który został przedstawiony na wykładzie. Zadaniem jest zaimplementowanie tego klasyfikatora w `pythonie` przy wykorzystaniu biblioteki `scikit-learn`. Można skorzystać z szablonu, który jest do pobrania ze strony przedmiotu.

Szczegóły zadania:

- Zakładamy, że na dziedziny cech są wartości nominalne kodowane za pomocą liczba całkowitych.
- Przetestuj rozwiązanie na zbiorze *grypa*. Poprawność można zweryfikować przeliczając prawdopodobieństwa *ręcznie*.

Kilka podpowiedzi:

- Zapoznaj się i skorzystaj z dostarczonego kodu.
- Przechowuj potrzebne informacje do obliczenia prawdopodobieństw w odpowiednich strukturach danych jako pola w klasie klasyfikatora.
- Metoda `fit` tworzy klasyfikator na podstawie danych wejściowych.
- Metoda `predict_proba` zwraca dla danych przykładów obliczone prawdopodobieństwa warunkowe klas w wektorze `numpy` .
- Metoda `predict` zwraca dla danych przykładów przewidywaną odpowiedź (tj. klasę).
- Folder `classifiers_students` zawiera szkielet klasy klasyfikatora.
- Folder `data` zawiera potrzebne dane.
- Folder `tasks_students` zawiera skrypty z prostymi testami napisanego klasyfikatora.
- Folder `utils` zawiera pożyteczne narzędzia np. do konwersji danych i ewaluacji klasyfikatorów.