Klasyfikacja i regresja:
Walidacja krzyżowa

28 listopada 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 Prawidłowa realizacja walidacji krzyżowej *10p.*◇

Treść

Walidacja krzyżowa jest metodą estymacji jakości algorytmów uczących na małych zbiorach danych. Pozwala ona także na odpowiednie dobranie hiperparametrów danego algorytmu uczącego. Jednak jej błędne wykorzystanie może doprowadzić do błędnych wniosków. Zadaniem do wykonania podczas zajęć jest analiza dwóch scenariuszów walidacji krzyżowej:

1. Selekcja najlepszych parametrów na zbiorze uczącym i uruchomienie walidacji krzyżowej,
2. Selekcja najlepszych parametrów podczas walidacji krzyżowej.

Należy zastanowić się, czy oba scenariusze są poprawne. Jeżeli nie, to który jest błędny.

W celu realizacji zadania wykonań następującego kroki:

1. Zapoznaj się z kodem zamieszczonym w pliku `compare_cv_scenarios.py` (plik jest dostępny na stronie przedmiotu oraz wydrukowany poniżej).
2. Zapoznaj się z opisem metod `SelectFromModel` oraz `SelectKBest`.
3. Uruchom eksperyment dla syntetycznych danych. Wypróbuj różne ustawienia liczby przykładów, liczby cech oraz liczby wybieranych cech. Liczbę iteracji (`fold`) pozostaw bez zmian.
4. Czy udało się zaobserwować nieprawidłowości?
5. Odpowiednio rozszerz kod, aby wyniki otrzymać wielokrotnie i odpowiednio je uśrednić.
6. Wyciągnij ostateczne wnioski.

```
1  import numpy as np
2  from sklearn.metrics import confusion_matrix
3  from sklearn.linear_model import LogisticRegression
4  from sklearn.model_selection import cross_val_predict
5  from sklearn.pipeline import Pipeline
6  from sklearn.feature_selection import SelectFromModel,
       SelectKBest
7
8
9  def create_random_data(n_instances, n_features):
10     X = 2*np.random.rand(n_instances, n_features) - 1
11     scores =  2*np.random.rand(n_instances) - 1
12     where_positive = scores > 0
13     where_negative = scores <= 0
14     scores[where_positive] = 1
15     scores[where_negative] = 0
```

```
    y = np.array(scores.ravel().tolist(), dtype=np.int32)
    return X, y

def print_evaluation(y, predicted):
    print("True: {0}/{1} = {2}".format(sum(y), len(y), float(sum
        (y))/len(y)))
    print("Confusion matrix")
    cm = confusion_matrix(y, predicted)
    print(cm)
    correct = cm[0, 0] + cm[1, 1]
    incorrect = cm[1, 0] + cm [0, 1]
    print(float(correct)/(correct + incorrect))

def selection_outside_cv(X, y, folds, sfm = SelectFromModel(
    LogisticRegression())):
    print("Scenario: Selection + CV(LR)")
    sfm.fit(X, y)
    X_selected = sfm.transform(X)

    predicted = cross_val_predict(LogisticRegression(),
        X_selected, y, cv=folds)
    print_evaluation(y, predicted)

def selection_inside_cv(X, y, folds, sfm = SelectFromModel(
    LogisticRegression())):
    print("Scenario: CV(Selection, LR)")
    clf = Pipeline([
        ('feature_selection', sfm),
        ('classification', LogisticRegression())
    ])
    predicted = cross_val_predict(clf, X, y, cv=folds)
    print_evaluation(y, predicted)


def main():
    n_instances = 1000
    n_features = 1000
    folds = 10
    n_top_features = 2

    X, y = create_random_data(n_instances, n_features)

    #use one of the methods for feature selection
    #sfm = SelectFromModel(LogisticRegression())
    sfm = SelectKBest(k=n_top_features)

    selection_outside_cv(X, y, folds, sfm)
    selection_inside_cv(X, y, folds, sfm)

if __name__ == "__main__":
    main()
```

compare_cv_scenarios.py