Практикум 2. Класифікація за допомогою бібліотеки Scikit-Learn Python

Недашківська Н.І.

Варіанти завдань та початкові дані вибирати відповідно до номеру в списку групи.

УВАГА! Для отримання максимальної оцінки потрібно виконати ВСІ етапи ходу виконання роботи та оформити звіт. Звіт має містити аналіз декількох моделей, підбір значень гіперпараметрів, оцінку впливу розміру навчальної вибірки на якість моделі.

Захист роботи:

- Демонстрація програми, яка реалізує завдання згідно з варіантом.
- Письмово теоретичне питання, задача та/ або виконання іншого варіанту завдання на комп'ютері.

1 Хід виконання роботи:

1. Представити початкові дані графічно.
2. Розбити дані на навчальний і перевірочний набори.
3. Побудувати моделі класифікації або регресії згідно з варіантом.
4. Представити моделі графічно.
5. Виконати прогнози на основі моделей.
6. Для кожної з моделей оцінити, чи має місце перенавчання.
7. Розрахувати додаткові результати моделей, наприклад, апостеріорні імовірності, опорні вектори або інші (згідно з варіантом).
8. Для задач класифікації розрахувати критерії якості для кожної моделі:
    - матрицю неточностей (confusion matrix),
    - точність (precision),
    - повноту (recall),
    - міру F1 (F1 score),

- побудувати криву точності-повноти (precision-recall (PR) curve), ROC-криву, показник AUC.

9. Виконати перехресну перевірку і решітчатий пошук для підбору гіперпараметрів моделей.

10. Зробити висновки про якість роботи моделей на досліджених даних. Для кожної навчальної вибірки вибрати найкращу модель.

11. Навчити моделі на підмножинах навчальних даних. Оцінити, наскільки розмір навчальної множини впливає на якість моделі.

2 Варіанти завдань для КА-75

1. Побудувати моделі регресії:

   - Лінійної регресії з різними значеннями гіперпараметру `fit_intercept`, використовуючи клас `sklearn.linear_model.LinearRegression`.
   - Поліноміальної регресії, використовуючи pipeline, PolynomialFeatures в поєднанні з LinearRegression.
   - Гребневої регресії з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Ridge`.
   - Порівняти моделі за коефіцієнтом детермінації R^2.

   Початкові дані:

   (а) `sklearn.datasets.load_boston`

   (б)
```
def make_data(N, err=1.0, rseed=1):
    rng = np.random.RandomState(rseed)
    X = rng.rand(N, 1) ** 2
    y = 10 - 1. / (X.ravel() + 0.1)
    if err > 0:
        y += err * rng.randn(N)
    return X, y

X, y = make_data(200)
```

2. Побудувати моделі класифікації на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeClassifier з різними значеннями гіперпараметрів:

   - `max_depth` – максимальна глибина дерева,
   - `min_samples_split` – мінімальна кількість прикладів, які мають бути у вузлі, перш ніж його можна буде розщепити,

- `min_samples_leaf` – мінімальна кількість прикладів у листовому вузлі,
- `max_leaf_nodes` – максимальна кількість листових вузлів,
- `max_features` – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла.

Побудувати границі рішень графічно для кожної з моделей.

Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

Початкові дані:

(а)
```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=400, centers=4,
                       cluster_std=0.60, random_state=0)
rng = np.random.RandomState(13)
X_stretched = np.dot(X, rng.randn(2, 2))
```

(б)
```
import numpy as np
np.random.seed(0)
X = np.random.randn(300, 2)
Y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0)
```

3. Побудувати моделі класифікації на основі методу опорних векторів:

   - Моделі sklearn.svm.LinearSVC та SVC(kernel="linear") з лінійним ядром, встановити велике значення параметра C. При використанні LinearSVC звернути увагу, що навчальні дані мають бути попередньо масштабовані і центровані.
   - Моделі з лінійним ядром і меншими значеннями параметра C для даних, які частково перетинаються.
   - Моделі SVC(kernel="poly") з поліноміальними ядрами. Дослідити різні ступені degree та гіперпараметр coef0 - управляє тим, наскільки сильно поліноми високого ступеня впливають на модель порівняно з поліномами низького ступеня.
   - Побудувати границі рішень графічно для кожної з моделей.
   - Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
   - Настроїти гіперпараметри C та degree за допомогою перехресної перевірки.
   - Настроїти гіперпараметр probability за допомогою перехресної перевірки.

   Початкові дані:

(а) `sklearn.datasets.make_moons`

(б) `sklearn.datasets.fetch_covtype`

4. Побудувати моделі класифікації на основі методу опорних векторів:

   - Моделі sklearn.svm.LinearSVC та SVC(kernel="linear") з лінійним ядром, встановити велике значенням параметра C. При використанні LinearSVC звернути увагу, що навчальні дані мають бути попередньо масштабовані і центровані.
   - Моделі з лінійним ядром і меншими значеннями параметра C для даних, які частково перетинаються.
   - Моделі SVC(kernel="rbf") з ядром "гаусівська радіальна базисна функція". Розглянути різні комбінації гіперпараметрів gamma і C, такі як: gamma=0.1 і C=0.01; gamma=0.1 і C=1; gamma=0.1 і C=100; gamma=10 і C=0.01; gamma=10 і C=1; gamma=10 і C=100.

     Збільшення gamma призводить до стиснення дзвоноподібної кривої, в результаті вплив кожного прикладу зменшується; границя рішень більше звивається навколо окремих прикладів. Невелике значення gamma робить границю рішень більш гладкою. Гіперпараметр gamma діє як регуляризатор: при перенавчанні слід зменшити значення gamma.
   - Побудувати границі рішень графічно для кожної з моделей.
   - Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
   - Настроїти гіперпараметри gamma і C за допомогою перехресної перевірки.
   - Настроїти гіперпараметр probability за допомогою перехресної перевірки.

   Початкові дані:

   (а)
   ```
   import numpy as np
   np.random.seed(0)
   X = np.random.randn(300, 2)
   Y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0)
   ```

   (б) `sklearn.datasets.load_digits`

5. Побудувати моделі класифікації на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeClassifier з різними значеннями гіперпараметрів:

   - `max_depth` – максимальна глибина дерева,

- `min_samples_split` – мінімальна кількість прикладів, які мають бути у вузлі, перш ніж його можна буде розщепити,
- `min_samples_leaf` – мінімальна кількість прикладів у листовому вузлі,
- `max_leaf_nodes` – максимальна кількість листових вузлів,
- `max_features` – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла.

Побудувати границі рішень графічно для кожної з моделей.

Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

Початкові дані:

(а) `sklearn.datasets.load_digits`

(б) `sklearn.datasets.make_moons`

6. Побудувати моделі логістичної регресії:

   - Просту логістичну регресію, використовуючи `sklearn.linear_model.LogisticRegression`.
   - Поліноміальну логістичну регресію (multinomial logistic regression), встановивши гіперпараметри `multi_class = "multinomial"` та `solver = "lbfgs"`.
   - Для наведених моделей побудувати варіанти з і без регуляризації.
   - Для кожної моделі розрахувати апостеріорні імовірності для тестового прикладу, використовуючи метод `predict_proba`.

   Початкові дані:

   (а) `sklearn.datasets.load_boston`

   (б)
```
def make_data(N, err=1.0, rseed=1):
    rng = np.random.RandomState(rseed)
    X = rng.rand(N, 1) ** 2
    y = 10 - 1. / (X.ravel() + 0.1)
    if err > 0:
        y += err * rng.randn(N)
    return X, y

X, y = make_data(200)
```

7. Побудувати моделі класифікації на основі методу опорних векторів:

- Моделі NuSVC(kernel="poly"). Розглянути різні комбінації гіперпараметрів nu, degree, coef0 - управляє тим, наскільки сильно поліноми високого ступеня впливають на модель порівняно з поліномами низького ступеня.
- Моделі SVC з різними ядрами та зваженими класами для випадку незбалансованих даних (параметр `class_weight`).
- Побудувати границі рішень графічно для кожної з моделей.
- Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
- Настроїти гіперпараметри nu, degree, C за допомогою перехресної перевірки.
- Настроїти гіперпараметр probability за допомогою перехресної перевірки.

Початкові дані:

(а)
```
from sklearn.datasets import make_blobs
n_samples_1 = 1000
n_samples_2 = 100
centers = [[0.0, 0.0], [-2.0, -2.0]]
clusters_std = [2.0, 1.0]
X, y = make_blobs(n_samples=[n_samples_1, n_samples_2],
                  centers=centers,
                  cluster_std=clusters_std,
                  random_state=0, shuffle=False)
```

(б) `sklearn.datasets.load_digits`

8. Побудувати моделі регресії:

- Лінійної регресії з різними значеннями гіперпараметру `fit_intercept`, використовуючи клас `sklearn.linear_model.LinearRegression`.
- Гребневої регресії з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Ridge`.
- Лассо-регуляризації з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Lasso`.
- Еластичної регуляризації з різними значеннями alpha і `l1_ratio`, використовуючи `sklearn.linear_model.ElasticNet`.
- Порівняти моделі за коефіцієнтом детермінації R^2.

Початкові дані:

(а) `sklearn.datasets.load_boston`

(б)
```
def make_data(N, err=1.0, rseed=1):
                rng = np.random.RandomState(rseed)
                X = rng.rand(N, 1) ** 2
                y = 10 - 1. / (X.ravel() + 0.1)
                if err > 0:
                                y += err * rng.randn(N)
                return X, y

X, y = make_data(200)
```

9. Побудувати моделі класифікації на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeClassifier з різними значеннями гіперпараметрів:

   - `max_depth` – максимальна глибина дерева,
   - `min_samples_split` – мінімальна кількість прикладів, які мають бути у вузлі, перш ніж його можна буде розщепити,
   - `min_samples_leaf` – мінімальна кількість прикладів у листовому вузлі,
   - `max_leaf_nodes` – максимальна кількість листових вузлів,
   - `max_features` – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла.

   Побудувати границі рішень графічно для кожної з моделей.

   Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

   Початкові дані:

   (а) `sklearn.datasets.load_digits`

   (б)
   ```
   from sklearn.datasets import make_blobs
   n_samples_1 = 1000
   n_samples_2 = 100
   centers = [[0.0, 0.0], [2.0, 2.0]]
   clusters_std = [1.5, 0.5]
   X, y = make_blobs(n_samples=[n_samples_1, n_samples_2],
                     centers=centers,
                     cluster_std=clusters_std,
                     random_state=0, shuffle=False)
   ```

10. Побудувати моделі регресії на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeRegressor з різними значеннями гіперпараметрів `max_depth`, `min_samples_split`, `min_samples_leaf`, `max_leaf_nodes`, `max_features`.

    Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

Порівняти моделі за коефіцієнтом детермінації R^2.

Початкові дані:

(а) `sklearn.datasets.load_boston`

(б)
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = np.sin(x) + 0.1 * rng.randn(50)
```

11. Побудувати моделі класифікації на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeClassifier з різними значеннями гіперпараметрів:

    - `max_depth` – максимальна глибина дерева,
    - `min_samples_split` – мінімальна кількість прикладів, які мають бути у вузлі, перш ніж його можна буде розщепити,
    - `min_samples_leaf` – мінімальна кількість прикладів у листовому вузлі,
    - `max_leaf_nodes` – максимальна кількість листових вузлів,
    - `max_features` – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла.

    Побудувати границі рішень графічно для кожної з моделей.

    Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

    Початкові дані:

    (а) `sklearn.datasets.samples_generator.make_circles`

    (б) `sklearn.datasets.fetch_covtype`

12. Побудувати моделі логістичної регресії:

    - Просту логістичну регресію, використовуючи `sklearn.linear_model.LogisticRegression`.
    - Поліноміальну логістичну регресію (multinomial logistic regression), встановивши гіперпараметри `multi_class = "multinomial"` та `solver = "lbfgs"`.
    - Для наведених моделей побудувати варіанти з і без регуляризації.
    - Для кожної моделі розрахувати апостеріорні імовірності для тестового прикладу, використовуючи метод `predict_proba`.

    Початкові дані:

(а)
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = np.sin(x) + 0.1 * rng.randn(50)
```

(б) `sklearn.datasets.load_diabetes`

13. Побудувати моделі класифікації на основі методу дерев рішень, використовуючи sklearn.tree.DecisionTreeClassifier з різними значеннями гіперпараметрів:

    - `max_depth` – максимальна глибина дерева,
    - `min_samples_split` – мінімальна кількість прикладів, які мають бути у вузлі, перш ніж його можна буде розщепити,
    - `min_samples_leaf` – мінімальна кількість прикладів у листовому вузлі,
    - `max_leaf_nodes` – максимальна кількість листових вузлів,
    - `max_features` – максимальна кількість ознак, які оцінюються при розщепленні кожного вузла.

    Побудувати границі рішень графічно для кожної з моделей.

    Настроїти гіперпараметри дерева за допомогою перехресної перевірки.

    Початкові дані:

    (а) `sklearn.datasets.load_iris`

    (б) `sklearn.datasets.samples_generator.make_circles`

    ```
    X, y = make_circles(500, factor=.1, noise=.1)
    ```

14. Побудувати моделі регресії:

    - Лінійної регресії з різними значеннями гіперпараметру `fit_intercept`, використовуючи клас `sklearn.linear_model.LinearRegression`.
    - Гребневої регресії з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Ridge`.
    - Лассо-регуляризації з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Lasso`.
    - Еластичної регуляризації з різними значеннями alpha і `l1_ratio`, використовуючи `sklearn.linear_model.ElasticNet`.
    - Порівняти моделі за коефіцієнтом детермінації R^2.

    Початкові дані:

    (а)
    ```
    rng = np.random.RandomState(1)
    x = 10 * rng.rand(50)
    y = np.sin(x) + 0.1 * rng.randn(50)
    ```

(б) `sklearn.datasets.load_diabetes`

15. Побудувати моделі наївної байесівської класифікації за припущень:

    - Дані в кожному класі мають нормальний розподіл без коваріації між вимірами; використати клас `sklearn.naive_bayes.GaussianNB`.
    - Дані в кожному класі мають поліноміальний розподіл; використати клас `sklearn.naive_bayes.MultinomialNB`.
    - Для кожної моделі розрахувати апостеріорні імовірності для тестового прикладу, використовуючи метод `predict_proba`.

    Початкові дані:

    (а) `sklearn.datasets.load_digits`

    (б) `sklearn.datasets.make_moons`

16. Побудувати моделі класифікації на основі методу опорних векторів:

    - Моделі sklearn.svm.LinearSVC та SVC(kernel="linear") з лінійним ядром, встановити велике значення параметра C. При використанні LinearSVC звернути увагу, що навчальні дані мають бути попередньо масштабовані і центровані.
    - Моделі з лінійним ядром і меншими значеннями параметра C для даних, які частково перетинаються.
    - Моделі SVC(kernel="poly") з поліноміальними ядрами. Дослідити різні ступені degree та гіперпараметр coef0 - управляє тим, наскільки сильно поліноми високого ступеня впливають на модель порівняно з поліномами низького ступеня.
    - Побудувати границі рішень графічно для кожної з моделей.
    - Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
    - Настроїти гіперпараметри C та degree за допомогою перехресної перевірки.
    - Настроїти гіперпараметр probability за допомогою перехресної перевірки.

    Початкові дані:

    (а) `sklearn.datasets.load_iris`

    (б)
```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=400, centers=4,
                       cluster_std=0.60, random_state=0)
rng = np.random.RandomState(13)
X_stretched = np.dot(X, rng.randn(2, 2))
```

17. Побудувати моделі регресії:

    - Лінійної регресії з різними значеннями гіперпараметру `fit_intercept`, використовуючи клас `sklearn.linear_model.LinearRegression`.
    - Поліноміальної регресії, використовуючи pipeline, PolynomialFeatures в поєднанні з LinearRegression.
    - Гребневої регресії з різними значеннями гіперпараметру alpha, використовуючи `sklearn.linear_model.Ridge`.
    - Порівняти моделі за коефіцієнтом детермінації R^2.

    Початкові дані:

    (а)
    ```
    rng = np.random.RandomState(1)
    x = 10 * rng.rand(50)
    y = np.sin(x) + 0.1 * rng.randn(50)
    ```

    (б) `sklearn.datasets.load_diabetes`

18. Побудувати моделі наївної байєсівської класифікації за припущень:

    - Дані в кожному класі мають нормальний розподіл без коваріації між вимірами; використати клас `sklearn.naive_bayes.GaussianNB`.
    - Дані в кожному класі мають поліноміальний розподіл; використати клас `sklearn.naive_bayes.MultinomialNB`.
    - Для кожної моделі розрахувати апостеріорні імовірності для тестового прикладу, використовуючи метод `predict_proba`.

    Початкові дані:

    (а) `sklearn.datasets.load_iris`

    (б) `sklearn.datasets.samples_generator.make_circles`

    ```
    X, y = make_circles(500, factor=.1, noise=.1)
    ```

19. Побудувати моделі класифікації на основі методу опорних векторів:

    - Моделі sklearn.svm.LinearSVC та SVC(kernel="linear") з лінійним ядром, встановити велике значенням параметра C. При використанні LinearSVC звернути увагу, що навчальні дані мають бути попередньо масштабовані і центровані.
    - Моделі з лінійним ядром і меншими значеннями параметра C для даних, які частково перетинаються.

- Моделі SVC(kernel="rbf") з ядром "гаусівська радіальна базисна функція". Розглянути різні комбінації гіперпараметрів gamma і C, такі як: gamma=0.1 і C=0.01; gamma=0.1 і C=1; gamma=0.1 і C=100; gamma=10 і C=0.01; gamma=10 і C=1; gamma=10 і C=100.

  Збільшення gamma призводить до стиснення дзвоноподібної кривої, в результаті вплив кожного прикладу зменшується; границя рішень більше звивається навколо окремих прикладів. Невелике значення gamma робить границю рішень більш гладкою. Гіперпараметр gamma діє як регуляризатор: при перенавчанні слід зменшити значення gamma.
- Побудувати границі рішень графічно для кожної з моделей.
- Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
- Настроїти гіперпараметри gamma і C за допомогою перехресної перевірки.
- Настроїти гіперпараметр probability за допомогою перехресної перевірки.

Початкові дані:

(а) `sklearn.datasets.load_iris`

(б) `sklearn.datasets.samples_generator.make_circles`

```
X, y = make_circles(500, factor=.1, noise=.1)
```

20. Побудувати моделі регресії на основі методу опорних векторів:

- Моделі лінійної регресії на основі класу sklearn.svm.LinearSVR і SVR(kernel="linear"), дослідити різні значенням параметра epsilon. При використанні LinearSVR звернути увагу, що навчальні дані мають бути попередньо масштабовані і центровані.
- Моделі нелінійної регресії SVR(kernel="poly") з поліноміальним ядром. Розглянути поліноми різного ступеня degree та різні комбінації гіперпараметрів epsilon і C, наприклад: epsilon=0.1 і C=0.01; epsilon=0.1 і C=100.
- Побудувати границі рішень графічно для кожної з моделей.
- Вивести значення опорних векторів (атрибут `support_vectors_`) для моделей.
- Настроїти гіперпараметри epsilon і C за допомогою перехресної перевірки.

Початкові дані:

(а) `sklearn.datasets.load_boston`

(б)
```
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = np.sin(x) + 0.1 * rng.randn(50)
```