Лабораторна робота №2. Класифікація та регресія за допомогою бібліотеки Scikit-Learn Python

Недашківська Н.І.

УВАГА! Для отримання максимальної оцінки потрібно виконати ВСІ етапи Ходу виконання роботи та оформити звіт. Звітом може бути, наприклад, файл jupyter notebook з кодом програми і текстовими поясненнями відповідно до Ходу виконання роботи та отриманих цифр. Звіт має містити аналіз декількох моделей, підбір гіперпараметрів, значення метрик якості моделей, оцінку впливу розміру навчальної вибірки на якість моделі.

Захист роботи:

- Усно: демонстрація коду програми, яка реалізує завдання згідно з варіантом та Ходом роботи. Відповіді на питання щодо коду програми, отриманих результатів та методу класифікації/регресії, який використовувався.
- Відповідь на теоретичне питання і розв'язання задачі (письмово !) на папері за розділами 1 і 2 навчальної програми дисципліни "ІАД".

1 Хід виконання роботи:

1. Побудувати моделі класифікації або регресії згідно з варіантом.
2. Виконати прогнози на основі побудованих моделей.
3. Для кожної з моделей оцінити, чи має місце перенавчання.
4. Розрахувати додаткові результати моделей, наприклад, апостеріорні імовірності, опорні вектори або інші (згідно з варіантом).
5. В задачах класифікації побудувати границі рішень графічно для кожної з моделей.
6. В задачах класифікації розрахувати значення наступних критеріїв якості, для кожної моделі окремо на навчальній та перевірочній множинах:
    - матрицю неточностей (confusion matrix),

- точність (precision),
- повноту (recall),
- міру F1 (F1 score),
- побудувати криву точності-повноти (precision-recall (PR) curve), ROC-криву, показник AUC.

7. В задачах регресії розрахувати критерії якості для кожної моделі окремо на навчальній та перевірочній множинах:
   - коефіцієнт детермінації R^2,
   - помилки RMSE, MAE та MAPE.
8. Виконати решітчастий пошук (grid search) для підбору гіперпараметрів моделей.
9. Зробити висновки про якість роботи моделей на досліджених даних. На основі критеріїв якості вибрати найкращу модель.
10. Навчити моделі на підмножинах навчальних даних. Оцінити, наскільки розмір навчальної множини впливає на якість моделі.
11. Кожен варіант містить два набори даних. Дослідити обидва набори за наведеними вище етапами.

2 Варіанти завдань

1. 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.

   Початкові дані:

   (a) `dataset_Malicious_and_Benign_Websites`.csv

(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)
```

2. 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`.
   - Поліноміальної регресії, використовуючи pipeline, PolynomialFeatures в поєднанні з LinearRegression.

   Початкові дані:

   (а) `sklearn.datasets.load_boston`

   (б) `www.kaggle.com/rahulsah06/gooogle-stock-price`

3. 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`.

   Початкові дані:

   (а) `sklearn.datasets.load_diabetes`

   (б) `www.kaggle.com/htagholdings/property-sales`

4. 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.
- Настроїти гіперпараметри C, degree, probability за допомогою решітчастого пошуку.

Початкові дані:

(а) `sklearn.datasets.make_moons`

(б) `sklearn.datasets.fetch_covtype` - це надвеликий набір. Якщо не вийде працювати з ним цілком, достатньо сформувати з нього піднабір, обираючи елементи випадковим чином. Моделі будувати на сформованому піднаборі.

5. 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.
   - Настроїти гіперпараметри gamma, C, probability за допомогою решітчастого пошуку.

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`

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.make_moons`

   (б) `sklearn.datasets.load_digits`

7. 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.
   - Моделі нелінійної регресії SVR(kernel="rbf") .
   - Настроїти гіперпараметри epsilon і C, використовуючи решітчастий пошук.

   Початкові дані:

   (а) `sklearn.datasets.make_friedman2`

   (б) Tesla stock з 2010 року по цей день

8. 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.

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 = [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)
```

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

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

9. 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)
```
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)
```

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

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

10. 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.

    Початкові дані:

    (а) `sklearn.datasets.make_moons`

    (б) `sklearn.datasets.load_digits`

11. 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.make_moons`

    (б) `sklearn.datasets.load_digits`

12. Побудувати моделі регресії на основі методу опорних векторів:

    - 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.
    - Настроїти гіперпараметри epsilon і C, використовуючи решітчастий пошук.

    Початкові дані:

    (а) `www.kaggle.com/bogof666/shanghai-car-license-plate-auction-price`

    (б) `avocado_prices.csv`

13. 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 решітчастого пошуку.
    - Настроїти гіперпараметр probability за допомогою решітчастого пошуку.

    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`

14. 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.

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_iris`

15. 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.
- 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.
- Настроїти гіперпараметри C та degree за допомогою решітчастого пошуку.

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) `dataset_Malicious_and_Benign_Websites.csv`

16. 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`.
- Поліноміальної регресії, використовуючи pipeline, PolynomialFeatures в поєднанні з LinearRegression.

Початкові дані:

(а) `sklearn.datasets.load_diabetes`

(б) `www.kaggle.com/jainshukal/netflix-stock-price`

17. 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)
```
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)
```

(б) `sklearn.datasets.fetch_covtype` - це надвеликий набір. Якщо не вийде працювати з ним цілком, достатньо сформувати з нього піднабір, обираючи елементи випадковим чином. Моделі будувати на сформованому піднаборі.

18. 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.
- Настроїти гіперпараметри epsilon і C.

Початкові дані:

(а) `sklearn.datasets.load_boston`

(б) `www.kaggle.com/sumanthvrao/daily-climate-time-series-data`

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.
- Настроїти гіперпараметри gamma, probability і C .

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 класифікації на основі методу опорних векторів:

- Моделі NuSVC(kernel="poly"). Розглянути різні комбінації гіперпараметрів nu, degree, coef0 - управляє тим, наскільки сильно поліномами 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`).
- Настроїти гіперпараметри nu, degree, C, probability.

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 = [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)
```

(б) `sklearn.datasets.load_wine`

21. 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.

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 = [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)
```

(б) `sklearn.datasets.load_digits`

22. 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`.
    - Поліноміальної регресії, використовуючи pipeline, PolynomialFeatures в поєднанні з LinearRegression.

    Початкові дані:

    (а) `sklearn.datasets.load_diabetes`

    (б) `avocado_prices.csv`

23. 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_iris`

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

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

24. 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.

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)
```

25. 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)
```
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)
```

26. 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)
```
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)
```

27. 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_iris`

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

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

28. Побудувати моделі регресії на основі методу опорних векторів:

- Моделі sklearn.svm.LinearSVR та SVR(kernel="linear") з лінійним ядром, встановити велике значенням параметра C. При використанні LinearSVR 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 SVR(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.

- Настроїти гіперпараметри gamma, C, probability.

Початкові дані:

(а) `sklearn.datasets.make_friedman3`

(б) `www.kaggle.com/atulanandjha/national-stock-exchange-time-series`

3 Контрольні питання для захисту роботи

1. Задачі машинного навчання (МН) з учителем і без учителя.
2. Перенавчання (overfitting) моделі МН. Ємність моделі. Компроміс між систематичною помилкою і дисперсією.
3. Крива перевірки.
4. Теорема Байеса і максимальна апостеріорна гіпотеза.
5. Лінійна регресія. Обгрунтування функції помилки в задачах регресії.
6. Регуляризація із зниженням ваги. Поняття гребневої регресії. Який клас використовується для навчання за цим методом в scikit-learn? Описати його основні параметри.
7. Постановка задачі класифікації на два класи, поняття відступу, функціоналу помилок та функції втрат. Лінійний класифікатор.
8. Постановка задачі та ідея методу опорних векторів для лінійно роздільного випадку. Функціонал помилок.
9. Метод опорних векторів для лінійно роздільного випадку. Сутність і обгрунтування методу.
10. Який клас використовується для навчання моделі опорних векторів в scikit-learn? Описати його основні параметри.
11. Метод опорних векторів для лінійно нероздільного випадку. Ідея методу. Типи векторів-порушників роздільчої смуги.
12. Метод опорних векторів для лінійно нероздільного випадку. Обгрунтування методу.

13. Нелінійне узагальнення методу опорних векторів. Означення ядра і приклади. Обгрунтування методу.

14. Порівняння розв'язків за методом опорних векторів для лінійно роздільного і нероздільного випадків. Переваги і обмеження методу. Як визначається параметр C ?

15. Поняття дерева рішень. Структура дерева рішень. Типи вершин. Навести приклад.

16. Алгоритм розбиття побудови дерев рішень. Властивості алгоритму розбиття.

17. Означення ентропії. Ентропійний критерій вибору змінної розбиття та його властивості.

18. Проблема перенавчання (зверхчутливості) дерев рішень. Алгоритм C4.5 вибору змінної розбиття. Модифікований C4.5 для випадку неперервних атрибутів.

19. Як побудувати і навчити модель дерева рішень для класифікації в scikit-learn Python? Атрибути вузла дерева. Міра забрудненості Джині.

20. Алгоритм CART для класифікації. Алгоритм розбиття побудови дерев рішень.

21. Переваги і обмеження алгоритму розбиття. Регуляризація дерев рішень. Міри ефективності дерева рішень.

22. Методи зупинки побудови дерева рішень для класифікації. Як робиться зупинка побудови дерева класифікації в scikit-learn?

23. Дерева рішень для регресії в Scikit-Learn Python, атрибути вузла. Алгоритм розбиття CART для регресії.

24. Алгоритм покриття побудови дерев рішень. 1R алгоритм.

25. Основи байесівської класифікації. Максимум апостеріорної імовірності. Штраф за помилку. Середній ризик. Оцінювання апріорних імовірностей та функцій правдоподібності.

26. Наївний байесівський класифікатор (алгоритм Naive Bayes). Який клас використовується в scikit-learn для навчання моделі Naive Bayes? Описати його основні параметри.

27. Оцінка ефективності класифікації: перехресна перевірка (cross-validation), решітчатий (grid search) і рандомізований пошук. Як вони реалізовані в scikit-learn?

28. Оцінка ефективності класифікації: confusion matrix, accuracy, precision, recall, F1 score. Як вони реалізовані в scikit-learn?

29. PR-крива та ROC-крива для оцінювання якості класифікації. Як вони будуються в scikit-learn?