Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Катедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни

«Прикладні задачі машинного навчання»

«Класифікація, регресія і кластеризація

з використанням бібліотеки scikit-learn»

Виконав студент ІП-11 Лесів Владислав Ігорович

Перевірив Нестерук Андрій Олександрович

Київ 2023

**Лабораторна робота 3**

**Класифікація, регресія і кластеризація**

**з використанням бібліотеки scikit-learn**

**Постановка завдання.**

1. Повторити дії, описані в пункті «Часові ряди і проста лінійна регресія частина 2» даної лабораторної роботи, та порівняти з результатом попередньої лабораторної роботи;
2. Аналогічно з прикладом з лекції 7, згенерувати набір даних та класифікувати його, використавши класифікатор SVC (слайд 95);
3. Порівняти декілька класифікаційних оцінювачів, наприклад, KNeighborsClassifier, SVC та GaussianNB для вбудованого в scikit-learn одного набору даних (вибрати довільний за бажанням).

**Хід роботи.**

1. Повторюю дії, описані в пункті «Часові ряди і проста лінійна регресія частина 2».

Аналогічно до попередньої роботи, завантажую дані з файлу січневих температур в Нью-Йорку з 1895 по 2018 рік, перейменовую стовпець 'Value' в 'Temperature', видаляю 01 в кінці кожного значення дати і виводжу кілька зразків даних.

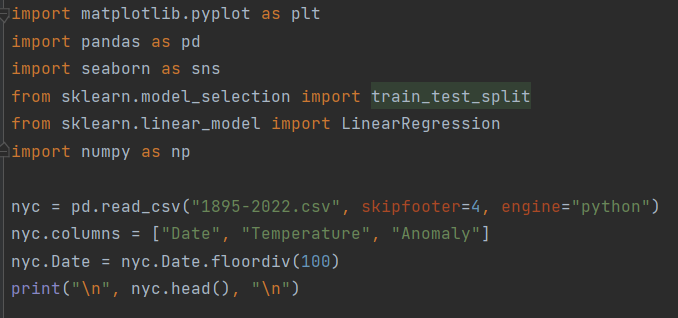


Рисунок 1 – Імпорт модулів та код обробки файлу

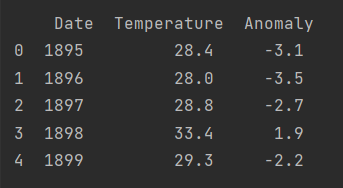


Рисунок 2 – Результат виконання коду, наведеного вище

Виконую просту лінійну регресію, використовуючи одну ознаку як незалежну змінну. У наборі даних вибираю ознаку Date. Дані розбиваю на навчальний і тестовий набори за допомогою sklearn.model\_selection.train\_test\_split(). Ключовий аргумент random\_state використовується для забезпечення відтворюваності результатів.

Для перевірки пропорції навчальних-тестових даних (75% до 25%) задаю розміри xTrain і xTest.

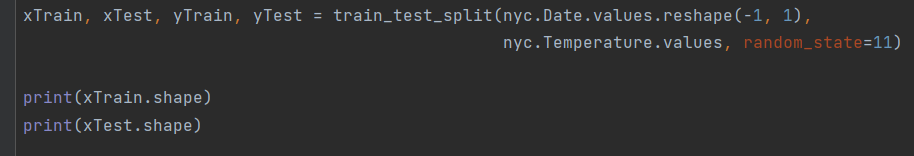


Рисунок 3 – Код формування навчального й тестового наборів



Рисунок 4 – Результат перевірки пропорцій даних

Скористаюся оцінювачем LinearRegression для простої лінійної регресії. Завантажую дані, після навчання fit повертає оцінювача. Значення кута нахилу і точки перетину з віссю, що використовуються у формулі , можуть використовуватися для прогнозування. Кут нахилу зберігається в атрибуті coeff\_ оцінювача, а точка перетину - в атрибуті intercept\_.

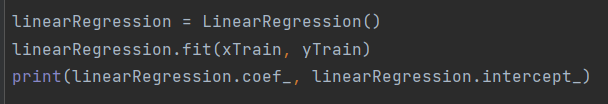


Рисунок 5 – Тренування оцінювача лінійної регресії



Рисунок 6 – Результат отримання куту нахилу та точки перетину

Протестую модель за даними з xTest і перевірю прогнози по набору даних, виводячи прогнозовані і очікувані значення для кожного п'ятого елементу.

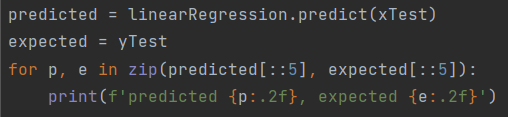


Рисунок 7 – Код тестування моделі і перевірки прогнозу

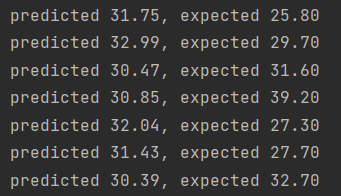


Рисунок 8 – Результат тестування моделі

Бачимо, що передбачені і очікувані значення то збігаються, то доволі сильно розходяться. Це, як і в минулій роботі, пов’язано з тим, що лінійна регресія – це пряма, тож якщо тенденція на зростання температури, прогноз також буде лінійно зростати. Водночас реальні показники – природні, тож можуть коливатися як в більшу, так і в меншу сторону залежно від безлічі умов.

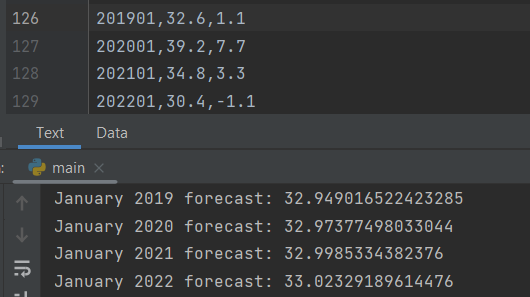


Рисунок 9 – Результати прогнозування в попередній роботі

Порівнюючи з попередньою лабораторною роботою, бачимо, що проблема все та ж, дані прогнозуються на зростання, але температура то на кілька градусів вища, то нижча. У цілому результати схожі.

Скористаємося отриманими значеннями кута нахилу і точки перетину для прогнозування середньої температури в січні 2019 року, а також оцінки середньої температури в січні 1890 року.

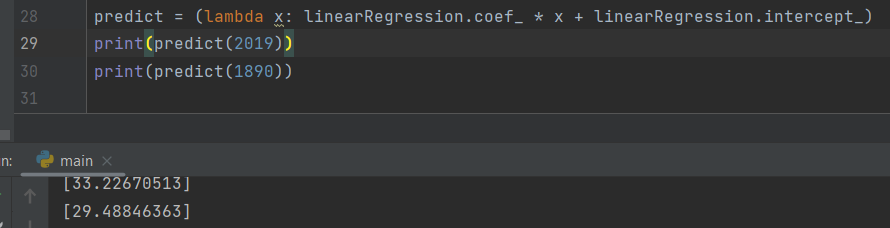


Рисунок 10 – Прогнозування температури за роком і результати прогнозування

У попередній роботі прогноз у 1890 році був 29.76 градусів, а в 2019 році – 32.95. Тобто значення дуже схожі, відрізняються на кілька десятих градуса.

Тепер побудую діаграму розкиду даних. Для виведення точок даних скористаюся методом scatterplot; зміню масштаб осі y, щоб при виведенні регресійної прямої лінійність відносин була більш очевидною.

Cтворю масив, що містить мінімальні і максимальні значення дати з nyc.Date, що стануть координатами x початкової і кінцевої точок регресійної прямої. Масив відповідних прогнозованих значень буде використовуватися в якості координат y.

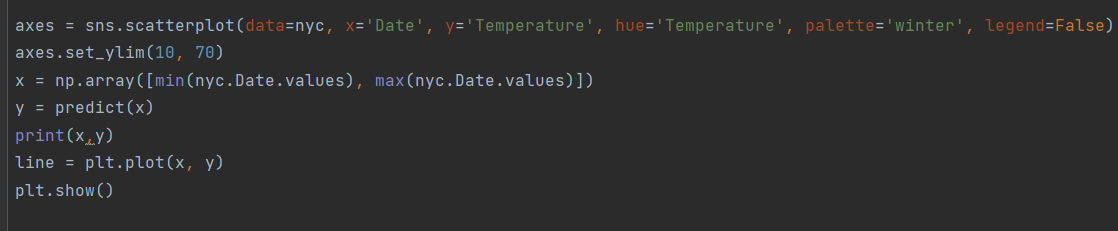


Рисунок 11 – Код для створення діаграми розкиду даних

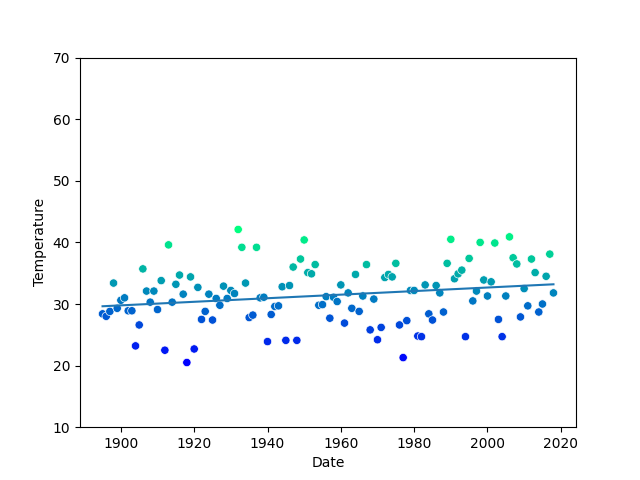


Рисунок 12 – Діаграма розкиду даних

Дійсно, отримана діаграма практично ідентична отриманій в попередній роботі.

1. Аналогічно з прикладом з лекції 7, згенерую набір даних та класифікую його, використавши класифікатор SVC (слайд 95);

Для початку я генерую 200 випадкових масивів по 2 значення у кожному, тобто наші координати. Потім виключним або розподіляю дані по тому, чи координати одного знаку, чи ні. Тобто виходить щось на кшталт 1-3 чверті й 2-4 чверті. Будуємо точки на графіку з таким розподілом.

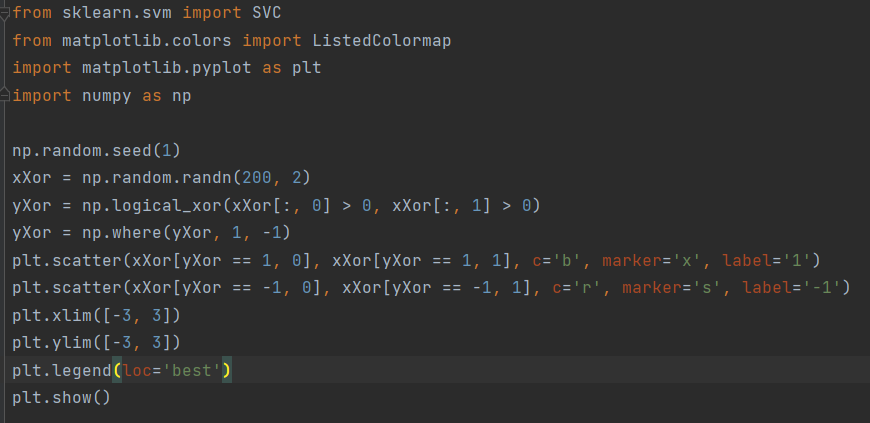


Рисунок 13 – Генерація набору даних та його розподіл

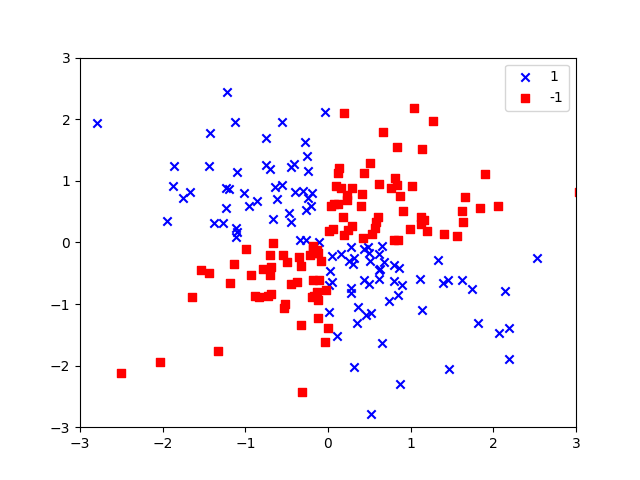


Рисунок 14 – Графік розподіленого набору даних

Далі маємо функцію розподілу точок за зонами. Знаходимо мінімальні та максимальні значення координат, далі робимо відповідні передбачення потрібним класифікатором щодо розподілу. Окреслюємо контур відповідних зон, яких у нас дві, і наносимо на графік точки у відповідних кольорах.

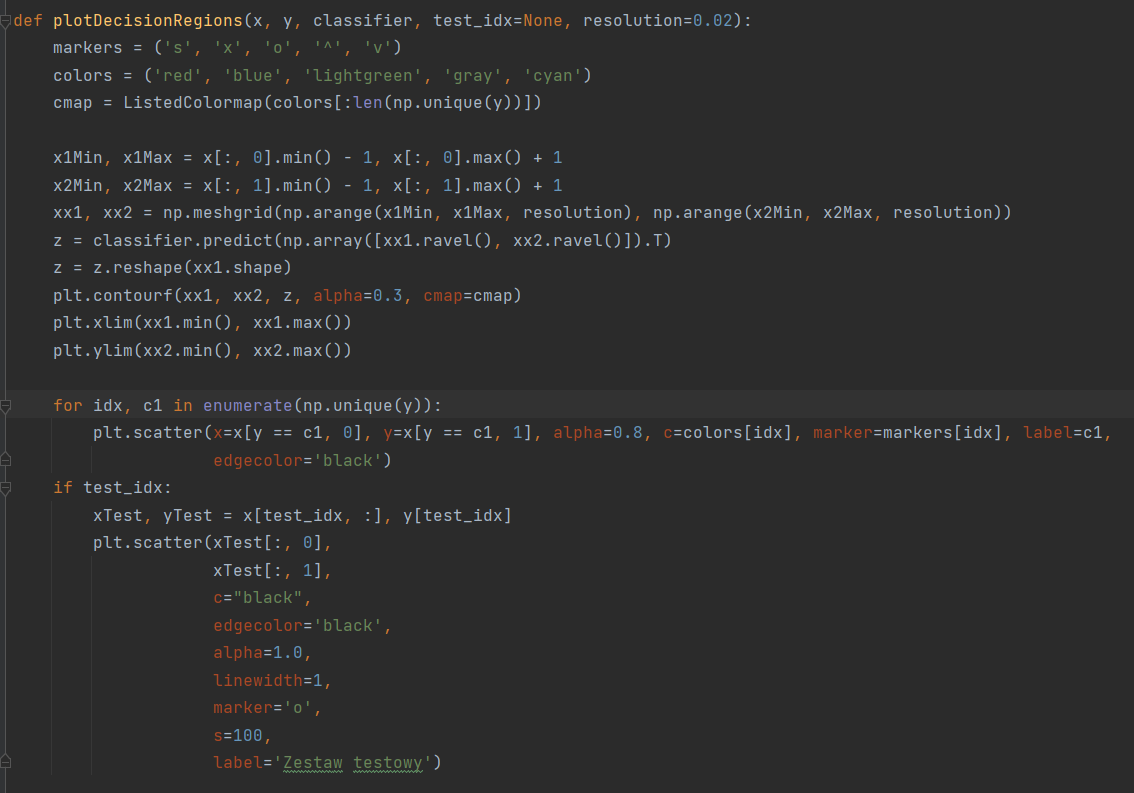


Рисунок 15 – Зонування даних на графіку

Для класифікації використовуємо класифікатор SVC з ядром “rbf”, адже класи неможливо розділити лінійно. Тож тренуємо оцінювач і викликаємо відповідну функцію.

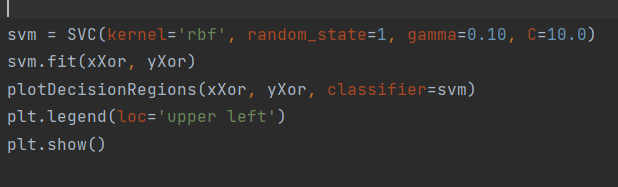


Рисунок 16 – Виконання класифікації й зонування

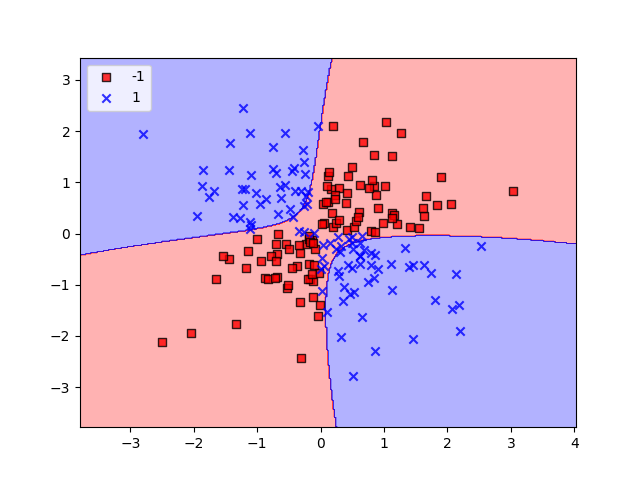


Рисунок 17 – Графік після класифікації

1. Порівнюю класифікаційні оцінювачі KNeighborsClassifier, SVC та GaussianNB для вбудованого в scikit-learn одного набору даних. Таким набором даних в мене виступає sklearn.datasets.load\_wine. Набір даних про вино, як дані має різні характеристики, а результатом – один з трьох класів вина.

Отже, завантажую дані, виділяю з них аргументи й результати.

Далі за допомогою StandartScaler() нормую вибірку x. За допомогою train\_test\_split() розділяю вибірку на навчальну та тестову частину. Наступним кроком створюю об’єкти класифікаторів, для К найближчих сусідів обираю k=5, для SCV – лінійне ядро, і відразу треную класифікатори.

Роблю передбачення для тестової вибірки, потім для кожного класифікатора порівнюю передбачення з реальним значенням і оцінюю ефективність.

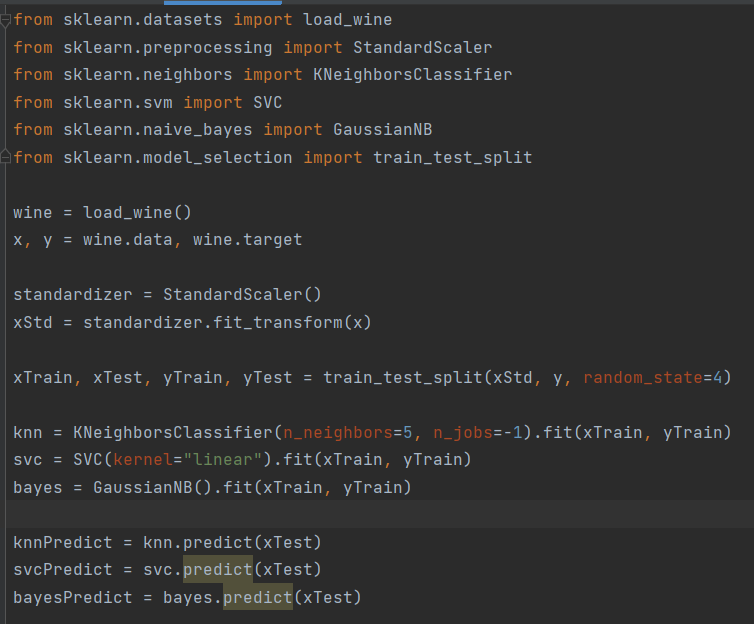


Рисунок 18 – Налаштування даних, класифікаторів та запуск передбачення

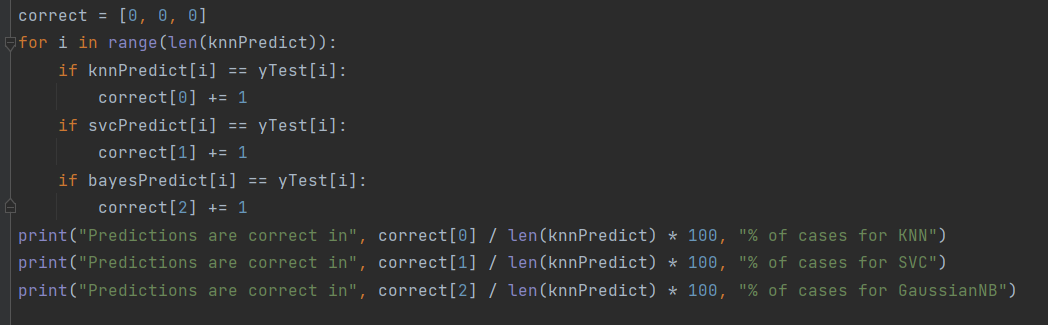


Рисунок 19 – Перевірка ефективности оцінки класифікаторів

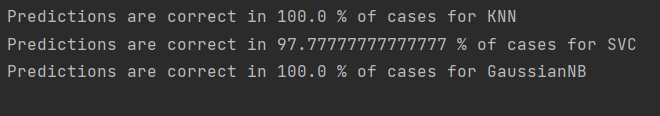


Рисунок 20 – Результат перевірки ефективности оцінки

Отже, отримали, що для даного набору даних найефективніше, на 100%, спрацювали класифікатори K-найближчих сусідів та GaussianNB. SVC спрацювало трохи гірше, але також результат оцінювання дуже високий, тобто усі 3 оцінювачі працюють ефективно для даного набору після проведених тренувань.

**Висновок.**

Отже, у цій роботі я отримав навички роботи з класифікацією, регресією і кластеризацією з використанням бібліотеки scikit-learn, підготовки, обробки даних, виведення статистики, візуалізації та вчинення відповідних дій. Класифікація є дійсно необхідною частиною машинного навчання, адже дозволяє з великою точністю визначати правильні класи у вибірці, що було досліджено й показано в лабораторній роботі. У результаті лабораторної роботи було вивчено роботу з модулем scikit-learn, побудовано графіки на основі оцінок за допомогою як лінійної регресії, так і класифікаторів, проведено дослідження й навчання за допомогою класифікаторів K-NN, SCV, GaussianNB. Використовуючи програмний засіб PyCharm, мову програмування Python з модулями, отримуємо коректний результат.