**Лабораторна робота №3**

**Тема:** Класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn

В даній лабораторній роботі буде використовуватися популярна бібліотека машинного навчання scikit-learn. Бібліотека scikit-learn, також звана sklearn, надає найбільш ефективні алгоритми машинного навчання, зручно упаковані в формі оцінювачів (estimators). Всі оцінювачі вміщені, тому подробиці і математичне обґрунтування роботи всіх цих алгоритмів не помітні розробнику. І вас це не повинно турбувати - людина може вести машину, не знаючи всіх подробиць роботи двигуна, системи передачі, системи гальмування або системи рульового управління. Уявіть, як ви входите в ліфт і вибираєте потрібний поверх або вмикаєте телевізор і вибираєте канал. Розбираєтеся ви в усіх подробицях того, як працює це устаткування або, скажімо, як функціонує програмне забезпечення вашого смартфона?

З scikit-learn і невеликим об'ємом коду Python можна швидко створити потужні моделі для аналізу даних, вилучення закономірностей з даних і, що найважливіше, побудови прогнозів. Ми будемо використовувати scikit-learn для навчання моделей на підмножині даних з подальшим тестуванням для перевірки того, як працює кожна модель. Після того як ваші моделі пройдуть навчання, ми застосуємо їх для побудови прогнозів на підставі даних, які їм ще не зустрічалися. Результати часто вражають. раптово ваш комп'ютер, який використовувався в основному для всяких рутинних завдань, починає проявляти зачатки інтелекту. Scikit-learn містить інструменти, що автоматизують процеси навчання і тестування моделей. І хоча ви можете задати параметри для настройки моделей з можливим зростанням їх ефективності, в цьому розділі ми зазвичай використовуємо налаштування моделей за замовчуванням, домагаючись при цьому вражаючих результатів. Також існують такі інструменти, як auto-sklearn (https://automl.github.io/auto-sklearn), автоматизують багато завдань, які вирішуються за допомогою scikit-learn.

**Якого оцінювача scikit-learn слід вибрати для проекту**

Важко заздалегідь визначити, які моделі найкраще підійдуть для ваших даних, тому зазвичай аналітик випробує багато моделей і вибирає ту, яка покаже найкращі результати. Як ви незабаром побачите, scikit-learn спрощує це завдання. Популярний підхід полягає в запуску багатьох моделей і виборі найкращого варіанту (-ів). Як же оцінити, яка модель показує найкращі результати?

Для цього потрібно поекспериментувати з безліччю різних моделей з різними видами наборів даних. Зазвичай знати подробиці складних математичних алгоритмів оцінювачів sklearn не потрібно, але з набуттям досвіду ви почнете уявляти, які алгоритми краще підходять для певних типів завдань і наборів даних. Втім, навіть маючи в своєму розпорядженні певним досвідом, навряд чи вам вдасться інтуїтивно вгадати найкращу модель для кожного нового набору даних. З цієї причини scikit-learn допомагає легко «випробувати їх усі». Створення і використання кожної моделі займає всього кілька рядків коду. Моделі видають інформацію про свою ефективності, що дозволяє порівняти результати і вибрати модель (-і) з найкращою ефективністю.

**Типи класифікаторів:**

Scikit-Learn дає доступ до безлічі різних алгоритмів класифікації. Ось основні з них:

• Метод k-найближчих сусідів (K-Nearest Neighbors);

• Метод опорних векторів (Support Vector Machines);

• Класифікатор дерева рішень (Decision Tree Classifier)

• Випадковий ліс (Random Forests);

• Наївний байесовский метод (Naive Bayes, GaussianNB);

• Лінійний дискримінантний аналіз (Linear Discriminant Analysis);

• Логістична регресія (Logistic Regression);

**Приклади завдань класифікації**

Завдання класифікації - ця будь-яке завдання, де потрібно визначити тип об'єкта з двох і більше існуючих класів. Такі завдання можуть бути різними: визначення, кішка на зображенні або собака, або визначення якості вина на основі його кислотності і вмісту алкоголю.

Залежно від завдання класифікації ви будете використовувати різні типи класифікаторів. Наприклад, якщо класифікація містить якусь бінарну логіку, то до неї найкраще підійде логістична регресія.

У міру накопичення досвіду вам буде простіше вибирати відповідний тип класифікатора. Однак хорошою практикою є реалізація декількох відповідних класифікаторів і вибір найбільш оптимального і продуктивного.

**Набори даних, включені в поставку scikit-learn**

У табл. перераховані набори даних, включені в поставку scikit-learn. Також надається можливість завантаження наборів даних з інших джерел, включаючи 20 000+ наборів даних, доступних на сайті openml.org.

Таблиця: Набори даних, включені в поставку scikit-learn

|  |  |
| --- | --- |
| «Іграшкові» набори даних | «Реальні» набори даних |
| Ціни на будинки в Бостоні  іриси  діабет  Оптичне розпізнавання рукописних цифр  Linnerrud  розпізнавання вин  Діагностика раку грудей (Вісконсін) | особи Оліветті  Тексти 20 новинних груп Помічені особи для розпізнавання типи  лісопосадок RCV1  Kidcup 99 California Housing |

**Часові ряди і проста лінійна регресія частина 2**

Раніше проста лінійна регресія була використана в прикладі з часовими рядами середньої січневої температури в Нью-Йорку за період з 1895 по 2018 рік. Крім того, в цьому прикладі використовувалася функція regplot бібліотеки Seaborn для побудови діаграми розкиду даних з відповідної регресійної прямої, а функція linregress модуля scipy.stats - для обчислення кута нахилу регресійної прямої і точки перетину з віссю. Отримані значення були використані для прогнозування майбутніх і оцінки минулих температур. В цій роботі розглянемо:

* Використання оцінювача scikit-learn для повторної реалізації простої лінійної регресії, що була реалізована в лабораторній роботі №2.
* Використання функції scatterplot бібліотеки Seaborn для графічного виведення даних і функції plot бібліотеки Matplotlib для виведення регресійної прямої.
* Використання значень кута нахилу і точки перетину, обчислених оцінювачем scikit-learn, для побудови прогнозів.

Пізніше ми розглянемо множинну лінійну регресію (яка називається також лінійної регресією).

**Завантаження середніх температур в колекцію DataFrame**

IPython слід запускати з ключем --matplotlib:



Завантажте дані з файлу ave\_hi\_nyc\_jan\_1895-2018.csv (це файл січневі температури в Нью-Йорку з 1895 по 2018 рік , який Ви завантажили з <https://www.ncdc.noaa.gov/cag/> в лабораторній роботі 3), перейменуйте стовпець 'Value' в 'Temperature', видаліть 01 в кінці кожного значення дати і виведіть кілька зразків даних:



**Розбиття даних для навчання і тестування**

Буде використовуватися оцінювач LinearRegression з sklearn.linear\_model. За замовчуванням він використовує **всі** числові ознаки в наборі даних, виконуючи *множинну лінійну регресію.*

Виконаємо просту лінійну регресію, використовуючи одну ознаку як незалежну змінну. У наборі даних необхідно вибрати одну ознаку (Date) з набору даних.

При виборі одного стовпчика в двовимірному DataFrame результат являє собою одновимірну колекцію Series. Однак оцінювачі scikit-learn вимагають, щоб в якості навчальних і тестових даних використовувалися двовимірні масиви (або двовимірні структури, подібні до масивами, наприклад списки списків або колекції pandas DataFrame). Щоб використовувати одномірні масиви з оцінювачем, необхідно перетворити їх з одновимірного масиву з n елементами в двовимірний масив з **n рядками і одним стовпцем**.

Дані розбиваємо на навчальний і тестовий набори. Ключовий аргумент random\_state використовується для забезпечення відтворюваності результатів:



Вираз nyc.Date повертає колекцію Series для стовпця Date, атрибут якої values повертає масив NumPy зі значеннями колекції. Для перетворення одновимірного масиву в двовимірний викличемо метод reshape масиву. Зазвичай в двох аргументах передається точна кількість рядків і стовпців, але перший аргумент -1 означає, що метод reshape повинен обчислити кількість рядків на підставі кількості стовпців (1) і кількості елементів (124) в масиві. Перетворений масив містить тільки один стовпець, тому reshape робить висновок, що кількість рядків дорівнює 124: розмістити 124 елемента в один стовпець можна, тільки розподіливши їх по 124 рядків.

Для перевірки пропорції навчальних і тестових даних (75% до 25%) задамо розміри X\_train і X\_test:



**Навчання моделі**

У scikit-learn немає окремого класу для простої лінійної регресії, тому що проста лінійна регресія є окремим випадком множинної лінійної регресії, тому ми скористаємося оцінювачем LinearRegression:



Після навчання оцінювача, fit повертає оцінювача і IPython виводить строкове представлення. Описи налаштувань за замовчуванням доступні за адресою:

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

Щоб знайти регресійну пряму з найкращого підгонкою до даних, оцінювач LinearRegression в ітеративному режимі регулює кут нахилу і точку перетину для мінімізації суми квадратів відстаней точок даних від лінії.

Після цього значення кута нахилу і точки перетину з віссю, що використовуються у формулі *y = mx + b,* можуть використовуватися для прогнозування. Кут нахилу зберігається в атрибуті coeff\_ оцінювача *(m* у формулі), а точка перетину – в атрибуті intercept\_ (*b* у формулі):



Пізніше ці значення будуть використані для виведення регресійної прямої і прогнозування для конкретних дат.

**Тестування моделі**

Протестуємо модель за даними з X\_test і перевіримо прогнози по набору даних, виводячи прогнозовані і очікувані значення для кожного п'ятого елементу



**Прогнозування майбутніх температур і оцінка минулих температур**

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. Лямбда-вираз в наступному фрагменті реалізує формулу:

*y = mx + b*

Значення coef\_ використовується замість *m*, а значення intercept\_ - замість *b*.



**Візуалізація набору даних з регресійній прямий**

Тепер побудуємо діаграму розкиду даних за допомогою функції scatterplot бібліотеки Seaborn і функції plot бібліотеки Matplotlib. Для виведення точок даних скористаємося методом scatterplot з колекцією DataFrame з ім'ям nyc:



Ключові аргументи:

* data - колекція DataFrame (nyc) з виведеними даними;
* x і y - імена стовпців nyc, які є джерелом даних по осях x і y відповідно. В даному випадку x містить ім'я стовпця 'Date', а y - 'Temperature'. Відповідні значення стовпців утворюють пари координат x-y, що наносяться на діаграму;
* hue - стовпець, дані якого використовуються для визначення кольорів точок ( 'Temperature'). У нашому прикладі колір особливої ролі не грає, але ми хотіли зробити діаграму більш привабливою;
* palette - колірна карта Matplotlib, по якій вибираються кольору точок;
* legend = False - на діаграмі розкиду даних не повинні виводитися умовні позначення. За замовчуванням використовується значення True, але в нашому прикладі умовні позначення не потрібні.

Змінимо масштаб осі y, щоб при виведенні регресійної прямої лінійність відносини була більш очевидною:



Перейдемо до висновку регресійної прямої. Почнемо зі створення масиву, що містить мінімальні і максимальні значення дати з nyc.Date. Вони стануть координатами x початкової і кінцевої точок регресійної прямої:



В результаті передачі predict масиву x у фрагменті [26] буде отримано масив відповідних прогнозованих значень, які будуть використовуватися в якості координат y:



Нарешті, функція plot бібліотеки Matplotlib малює лінію по масивам x та y, що зображує координати x та y точок відповідно:



Отримана діаграма розкиду даних з регресійній прямий зображена на наступній діаграмі. Вона практично ідентична тій, що Ви отримали в лабораторній роботі №2



**Надмірна / недостатня підгонка**

При створенні моделі потрібно в першу чергу прагнути до того, щоб ця модель видавала точні прогнози для даних, які їй поки невідомі. Дві поширені проблеми, що перешкоджають точному прогнозуванню,

* надмірна і недостатня підгонка:

**Недостатня підгонка** відбувається в тому випадку, якщо модель занадто проста для побудови прогнозів на підставі тренувальних даних. Наприклад, лінійна модель (скажімо, проста лінійна регресія) використовується в задачі, яка насправді вимагає нелінійної моделі. Наприклад, температури істотно змінюються протягом чотирьох пор року. Якщо ви спробуєте створити узагальнену модель, яка може прогнозувати температури круглий рік, модель простої лінійної регресії призведе до недостатньої підгонці даних.

**Надмірна підгонка** відбувається при надмірній складності моделі. Крайній випадок такого роду - модель, що запам'ятовує свої навчальні дані. Таке рішення прийнятно, якщо нові дані будуть повністю співпадати з навчальними, але зазвичай це не так. При побудові прогнозів на підставі моделі з надмірною підгонкою для нових даних, які збігаються з навчальними, будуть зроблені ідеально точні прогнози, але така модель не буде знати, що робити з даними, які їй ще не зустрічалися.

Додаткову інформацію про надмірну і недостатню підгонці можна знайти тут:

<https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>

**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 згенеруйте набір даних та класифікуйте 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).**
4. **Зробити звіт про роботу, який включає:**

**а. Титульна сторінка з інформацією про виконавця, темою та номером лабораторної роботи,**

**б. Постановку завдання**

**в. Скріни коду та скріни результату виконання з коментарями**

**г. Висновок**

1. **Надіслати звіт(.docx або .doc) на @zeit\_13 (Telegram) або aonesterukr@gmail.com (e-mail). В повідомленні обов’язково вказати Ваше ПІБ, групу, назву предмету (скорочено)**
2. **Записати захист роботи на відео та надіслати його разом зі звітом на вище вказані адреси. Під час захисту роботи показати роботу програмного коду! (не звіт) та пояснити свої дії.**