Лабораторна робота №7

Кластеризація та регресія в scikit-learn

Мета роботи: Ознайомитись з побудовою моделей для вирішення задач регресії та кластеризації в scikit-learn, визначити основні оцінки цих моделей.

Короткі теоретичні відомості

Машинне навчання — це підмножина штучного інтелекту (ШІ), за допомогою якого алгоритм може навчитися передбачати значення на основі вхідних даних без явного вивчення правил. Ці алгоритми покладаються на статистичні дані, щоб робити висновки під час навчання; потім вони використовують те, чого навчилися, щоб робити прогнози.

Машинне навчання зазвичай поділяється на три категорії: навчання без учителя, навчання з учителем і навчання з підкріпленням.

Навчання без учителя використовується тоді, коли немає позначених даних, тобто для жодних даних невідомо, до якого класу вони належать.

Якщо є позначені дані, то використовується навчання з учителем.

Навчання з підкріпленням пов’язане з реакцією на зворотний зв’язок із оточення.

Найпоширенішими завданнями машинного навчання є кластеризація, класифікація та регресія.

В задачах кластеризації метою є розподілення даних за групами, при цьому групи повинні бути чітко визначеними, тобто члени групи знаходяться близько один до одного, а самі групи - відокремлені від інших груп.

Регресія, з іншого боку, призначена для передбачення числових значень; вона моделює силу і величину зв'язків між змінними.

Потрібно розділити дані на навчальний набір і набір для тестування. Для цього можна перетасувати датафрейм і вибрати верхні x% рядків для навчання, а решту залишити для тестування. Але scikit-learn надає функцію train_test_split() у модулі model_selection, яка є більш надійним і простим у використанні рішенням. Для її використання потрібно попередньо відокремити вхідні дані (X) від вихідних даних (y).

Зазвичай, обирають об’єм набору для тестування від 10% до 30% даних.

sklearn.model_selection.train_test_split()

test_size – розмір тестової вибірки, задається кількість рядків або частка. Якщо не вказаний розмір навчальної вибірки, то значення за замовчуванням – 0.25

train_size – розмір навчальної вибірки

shuffle – чи перемішувати дані перед розбиттям, за замовчуванням True

```
from sklearn.model_selection import train_test_split
X = planet_data[['mass','eccentricity', 'semimajoraxis']]
y = planet_data.period
```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

Попередня підготовка даних до використання в моделях машинного навчання включає:

- масштабування та центрування даних;
- кодування даних;
- визначення відсутніх даних та їх заповнення;
- інші перетворення даних.

Scikit-learn має модуль preprocessing для операцій з перетворення даних.

Клас StandardScaler здійснює стандартизацію даних, тобто приводить їх до стандартного нормального розподілу.

Клас MinMaxScaler здійснює нормалізацію даних, тобто перераховує всі значення в діапазон [0,1].

Кодування даних перетворює категоріальні змінні у числові. Для бінарного кодування використовується клас LabelBinarizer.

Для перетворення категоріальної характеристики з кількома значеннями використовується клас LabelEncoder або OrdinalEncoder.

Класи для роботи з відсутніми значеннями у scikit-learn знаходяться у модулі impute.

scikit-learn пропонує можливість створювати конвеєри, щоб спростити попередню обробку та гарантувати, що навчальні набори та тестувальні набори обробляються однаково.

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression

Pipeline([('scale', StandardScaler()), ('lr', LinearRegression())])

linear_model містить моделі лінійної регресії. Існує проста лінійна регресія, що має одну незалежну змінну та множинна лінійна регресія, де незалежних змінних кілька.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

from sklearn.linear_model import LinearRegression

lm = LinearRegression().fit(X_train, y_train)

lm.intercept_

lm.coef_

preds = lm.predict(X_test)

Щоб оцінити модель лінійної регресії, потрібно візуалізувати залишки (розбіжності між фактичними значеннями та прогнозами моделі); вони повинні бути зосереджені навколо нуля і мати однакову дисперсію. Можна використати ядерну оцінку щільності, щоб оцінити, чи залишки зосереджені навколо нуля, і діаграму розсіювання, щоб побачити, чи мають вони однакову дисперсію.

Метод score() лінійної регресійної моделі повертає параметр R^2, або коефіцієнт детермінації, який кількісно визначає частку дисперсії залежної

змінної, яку можна передбачити на основі незалежних змінних. Цей коефіцієнт має значення від 0 до 1, і чим ближчий він до одиниці, тим краще.

Ще одна метрика, яку пропонує scikit-learn, — це оцінка поясненої дисперсії, яка говорить нам про відсоток дисперсії, що пояснюється моделлю. Потрібно, щоб це значення було якомога ближче до 1:

```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, preds)
```

Середня абсолютна похибка (MAE) показує середню похибку моделі в будь-якому напрямку. Значення коливаються від 0 до ∞ (нескінченність), причому менші значення є кращими:

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, preds)
```

Середньоквадратична помилка (RMSE) дозволяє додатково оцінити прогнозування:

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, preds))
```

Альтернативою всім цим вимірюванням на основі середнього значення є середня абсолютна помилка, яка є медіаною залишків. Її можна використовувати в тих випадках, коли є кілька викидів у залишках, і потрібно отримати більш точной опис основної частини похибок.

```
from sklearn.metrics import median_absolute_error
median_absolute_error(y_test, preds)
```

Кластеризація використовується, щоб розділити точки даних на групи подібних точок. Точки в кожній групі більше схожі на точки зі своєї групи, ніж з інших груп.

Популярний алгоритм для кластеризації – це алгоритм k-середніх, який ітераційно призначає точки найближчій групі, використовуючи відстань від центру групи, створюючи k груп.

```
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
kmeans_pipeline = Pipeline([('scale', StandardScaler()),('kmeans', KMeans(8, random_state=0))])
kmeans_data = planet_data[['semimajoraxis', 'period']].dropna()
kmeans_pipeline.fit(kmeans_data)
```

Для того, щоб оцінити виконання задачі кластеризації у випадку навчання без учителя, потрібно використовувати показники, які оцінюють аспекти самих кластерів, наприклад, наскільки вони віддалені один від одного і наскільки близько розташовані точки в кластері. Можна порівняти кілька показників, щоб отримати більш повну оцінку ефективності.

Один з таких методів називається коефіцієнтом силуету, який допомагає кількісно визначити поділ кластерів. Він обчислюється шляхом віднімання середнього значення відстаней між кожними двома точками в кластері (a) із

середнього відстаней між точками в даному кластері та найближчого іншого кластера (b) і діленням на максимальну з двох. Ця метрика повертає значення в діапазоні [-1, 1], де -1 є найгіршим (кластери призначені неправильно), а 1 найкращим; значення поблизу 0 вказують, що кластери перекриваються. Чим вище це число, тим краще визначені (більш відокремлені) кластери:

```
from sklearn.metrics import silhouette_score
silhouette_score(kmeans_data, kmeans_pipeline.predict(kmeans_data))
```

Ще одна оцінка, яку можна використовувати для оцінки результату кластеризації, — це відношення відстаней всередині кластера (відстаней між точками в кластері) до відстаней між кластерами (відстаней між точками в різних кластерах), яке називається оцінкою Девіса-Болдіна. Значення ближчі до нуля вказують на кращі розділи між кластерами:

```
from sklearn.metrics import davies_bouldin_score
davies_bouldin_score(kmeans_data, kmeans_pipeline.predict(kmeans_data))
```

Ще одним показником є оцінка Калінського та Харабаса, або критерій співвідношення дисперсії, який є відношенням дисперсії всередині кластера до дисперсії між кластерами. Вищі значення вказують на краще визначені (більш відокремлені) кластери:

```
from sklearn.metrics import calinski_harabasz_score
calinski_harabasz_score(kmeans_data, kmeans_pipeline.predict(kmeans_data))
```

Завдання до лабораторної роботи

Написати програму, яка здійснює попередню обробку даних, навчає та тестує (при потребі) модель, що виконує завдання відповідно до варіанту, оцінити модель за допомогою відповідних метрик.

Оформити звіт. Звіт повинен містити:

— титульний лист;
— код програми;
— результати виконання коду.

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.

Варіант 1: lab2/Crime.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати рівень злочинності в даний час.
Варіант 2: lab4/diamonds.csv. Виконати кластерний аналіз на даних, описати результати.
Варіант 3: seeds.csv. Виконати кластерний аналіз на даних, описати результати.
Варіант 4: lab2/Birthweight.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати довжину дитини.
Варіант 5: customer.csv. Виконати кластерний аналіз для сегментування клієнтів.

Варіант 6: lab4/merc.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати ціну на мерседес.
Варіант 7: lab2/possum.csv. Виконати кластерний аналіз на даних, описати результати.
Варіант 8: lab4/insurance.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати ціну страховки.
Варіант 9: lab2/Birthweight.csv. Виконати кластерний аналіз на даних, описати результати.
Варіант 10: lab4/diamonds.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати ціну діаманту.
Варіант 11: Fish.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати вагу риби.
Варіант 12: lab2/Crime.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати рівень злочинності, використовуючи дані десятирічної давнини.
Варіант 13: lab2/possum.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати вік опосумів.
Варіант 14: Real estate.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати ціну будинку.
Варіант 15: lab4/StudentsPerformance.csv. Виконати кластерний аналіз оцінок учнів.