Pandas. Build-in methods

Analyzing DataFrames

При роботі з DataFrames в бібліотеці Pandas ви маєте великий набір функцій та операцій для аналізу та маніпулювання даними. Ось декілька загальних технік для аналізу DataFrames:

1. Вивчення DataFrame:
    - `df.head()` та `df.tail()`: Переглянути перші або останні кілька рядків DataFrame.
    - `df.shape`: Отримати розміри DataFrame (кількість рядків та стовпців).
    - `df.info()`: Отримати підсумок DataFrame, включаючи типи даних та кількість ненульових значень в кожному стовпці.
    - `df.describe()`: Створити описові статистики для числових стовпців, таких як кількість, середнє, стандартне відхилення, мінімум, максимум та квартилі.
2. Доступ та вибір даних:
    - Вибір стовпця: Використовуйте `df['назва_стовпця']` або `df.назва_стовпця` для доступу до конкретного стовпця.
    - Вибір рядка: Використовуйте булеву індексацію або умови для фільтрації рядків за певними критеріями.
    - Зрізи: Виберіть підмножину рядків або стовпців за допомогою індексації за мітками з `df.loc` або за позиціями з `df.iloc`.
3. Агрегація даних:
    - `df.groupby('стовпець')`: Згрупуйте дані за певним стовпцем.
    - Функції агрегації: Застосуйте функції, такі як `mean()`, `sum()`, `count()`, `min()`, `max()` і т.д., для розрахунку загальних статистичних показників для груп або всіх стовпців.
4. Очищення та попередня обробка даних:
    - Обробка пропущених значень: Використовуйте `df.dropna()` для видалення рядків або стовпців з

пропущеними значеннями або `df.fillna(значення)` для заповнення пропущених значень певним значенням.

- Видалення дублікатів: Використовуйте `df.drop_duplicates()` для видалення дублікатних рядків з DataFrame.
- Конвертація типів даних: Конвертуйте типи даних стовпців за допомогою функцій, таких як `df.astype()` або `pd.to_numeric()`.

1. Візуалізація:
    - Pandas інтегрується з популярними бібліотеками візуалізації, такими як Matplotlib та Seaborn. Використовуйте функції, такі як `df.plot()`, щоб створювати різні графіки, включаючи лінійні графіки, стовпчикові діаграми, точкові діаграми та гістограми.
2. Розширені операції:
    - Злиття та приєднання: Об'єднуйте кілька DataFrames на основі спільних стовпців за допомогою `pd.merge()` або `df.join()`.
    - Таблиці зведення: Створюйте зведені таблиці за допомогою `df.pivot_table()`, щоб агрегувати та перетворювати дані.
    - Сортування та ранжування: Сортуйте DataFrame за значеннями стовпців за допомогою `df.sort_values()` та ранжуйте дані за допомогою `df.rank()`.

Це лише кілька прикладів з широкого спектру операцій, які можна виконувати з DataFrames у Pandas. Бібліотека Pandas надає вичерпну документацію та ресурси для подальшого вивчення та ознайомлення з багатьма функціями та можливостями бібліотеки.

Cleaning Data

Очищення даних є важливою частиною роботи з даними. У бібліотеці Pandas є ряд функцій та методів, які допомагають в очищенні даних у DataFrame. Ось кілька основних технік очищення даних з використанням Pandas:

1. Обробка пропущених значень:
    - `df.isnull()`: Повертає DataFrame з булевими значеннями, які показують, де є пропущені значення.

- `df.dropna()`: Видаляє рядки або стовпці з пропущеними значеннями.
- `df.fillna(value)`: Заповнює пропущені значення в DataFrame певним значенням, наприклад, середнім або медіаною.

2. Обробка дублікатів:
   - `df.duplicated()`: Повертає DataFrame з булевими значеннями, які показують, де є дублікати.
   - `df.drop_duplicates()`: Видаляє дублікатні рядки з DataFrame.
3. Обробка неправильних типів даних:
   - `df.astype(dtype)`: Змінює типи даних стовпців на задані типи.
   - `pd.to_numeric()`: Конвертує значення стовпця в числовий тип.
4. Робота зі стрічками:
   - `df.str.strip()`: Видаляє пробіли з початку та кінця стрічок в стовпцях.
   - `df.str.lower()` або `df.str.upper()`: Змінює регістр стрічок в стовпцях на нижній або верхній.
5. Видалення зайвих стовпців:
   - `df.drop(columns=['column_name'])`: Видаляє стовпці з DataFrame за їхніми назвами.
6. Обробка дат:
   - `pd.to_datetime()`: Конвертує стовпець з датою/часом у тип `datetime`.
   - `df.dt.year`, `df.dt.month`, `df.dt.day`: Витягує рік, місяць, день зі стовпця з датою/часом.
7. Чистка та заміна даних:
   - `df.replace(old_value, new_value)`: Замінює значення у DataFrame на нові значення.
   - `df.dropna(subset=['column_name'])`: Видаляє рядки з пропущеними значеннями у певному стовпці.

Це лише кілька прикладів того, як Pandas можна використовувати для очищення даних у DataFrame. Залежно від специфіки вашого набору даних можуть бути необхідні інші операції чи функції. Pandas надає багато інших функцій, які можна використовувати для очищення та підготовки даних перед аналізом.

Cleaning Empty Cells

Очищення порожніх комірок:

При роботі з порожніми комірками (відсутніми значеннями) у DataFrame за допомогою Pandas у вас є кілька варіантів для їх очищення та обробки. Ось декілька типових технік:

1. Видалення рядків або стовпців:
    - `df.dropna()`: Видаляє рядки або стовпці, що містять хоча б одне порожнє значення.
    - `df.dropna(axis=1)`: Видаляє стовпці, що містять хоча б одне порожнє значення.
    - `df.dropna(subset=['назва_стовпця'])`: Видаляє рядки, які містять порожні значення саме в зазначеному стовпці(ях).
2. Заповнення порожніх значень:
    - `df.fillna(значення)`: Заповнює порожні значення певним значенням, таким як 0, середнє, медіана або будь-яке інше потрібне значення.
    - `df.fillna(method='ffill')`: Заповнює порожні значення попередніми непорожніми значеннями (вперед заповнення).
    - `df.fillna(method='bfill')`: Заповнює порожні значення наступними непорожніми значеннями (назад заповнення).
3. Інтерполяція:
    - `df.interpolate()`: Виконує інтерполяцію для оцінки пропущених значень на основі наявних значень в DataFrame.
4. Перевірка на пропущені значення:
    - `df.isnull()`: Повертає DataFrame тієї ж форми з булевими значеннями, що вказують на наявність пропущених значень.
    - `df.isnull().sum()`: Повертає кількість пропущених значень в кожному стовпці.
    - `df.isnull().any()`: Повертає булевий рядок, що показує, чи є пропущені значення в кожному стовпці.
5. Видалення дублікатів:

- `df.drop_duplicates()`: Видаляє дублікатні рядки з DataFrame, залишаючи тіль

ки перше входження.

1. Обробка спеціальних значень:
    - Іноді пропущені значення позначаються спеціальними значеннями, такими як "NA" або "N/A". Ви можете замінити такі значення за допомогою функції `replace()`.

Пам'ятайте, що вибір методу очищення залежить від конкретного контексту та особливостей ваших даних. Pandas надає гнучкість у роботі з порожніми значеннями, дозволяючи вибрати найбільш підходящий підхід для ваших завдань аналізу або моделювання.

Cleaning data

Виправлення неправильних даних є важливим етапом при роботі з даними. У бібліотеці Pandas є кілька способів виправлення неправильних даних у DataFrame. Ось декілька прикладів:

1. Заміна значень:
    - `df.replace(старе_значення, нове_значення)`: Замінює вказані значення на нові значення у всьому DataFrame.
    - `df[column_name].replace(старе_значення, нове_значення)`: Замінює вказані значення на нові значення у конкретному стовпці.
2. Використання функцій перетворення:
    - `df.apply(функція)`: Застосовує функцію до кожного елемента у DataFrame або окремому стовпцю.
    - `df[column_name] = df[column_name].apply(функція)`: Застосовує функцію до окремого стовпця та оновлює його значення.
3. Фільтрація та умовні вирази:
    - `df[умовний_вираз]`: Вибирає рядки, що задовольняють певний умовний вираз.
    - `df.loc[умовний_вираз, column_name] = нове_значення`: Замінює значення у певному стовпці для рядків, що задовольняють умовний вираз.
4. Ручне оновлення значень:

- Використовуйте індексацію за допомогою `df.loc[row_index, column_name]` для прямого доступу та оновлення певних значень у DataFrame.

Це лише кілька прикладів того, як Pandas можна використовувати для виправлення неправильних даних у DataFrame. Вибір конкретного методу залежить від типу помилки та характеристик вашого набору даних. Завжди перевіряйте та переконуйтеся, що виправлені дані відповідають вашим очікуванням та логіці аналізу.

Removing Duplicates

Видалення дублікатів є важливим кроком при очищенні та підготовці даних. У бібліотеці Pandas є кілька методів, які допомагають видалити дублікати з DataFrame. Ось кілька прикладів:

1. Видалення повних дублікатів рядків:
   - `df.drop_duplicates()`: Видаляє повторюючіся рядки з DataFrame, залишаючи тільки перше входження.
2. Видалення дублікатів за певними стовпцями:
   - `df.drop_duplicates(subset=['column_name'])`: Видаляє рядки, які мають повторення в певному стовпці(ях).
3. Видалення дублікатів з урахуванням певного підмножини стовпців:
   - `df.drop_duplicates(subset=['column1', 'column2'])`: Видаляє рядки, які мають повторення у вказаних стовпцях.
4. Заміна дублікатів значеннями за правилом:
   - `df.replace(duplicate_value, new_value)`: Замінює значення дублікатів на нове значення.

Видалення дублікатів допомагає зберегти лише унікальні записи та уникнути впливу повторюючихся даних на аналіз. Перед видаленням дублікатів завжди ретельно перевіряйте, які стовпці або комбінації стовпців визначають унікальність записів у вашому наборі даних.

Correlation

Кореляція даних є важливим поняттям при аналізі даних і дозволяє виявляти залежності між різними змінними. У бібліотеці Pandas є методи

для обчислення кореляції даних в DataFrame. Ось кілька способів розрахунку кореляції:

1. Кореляційна матриця:
    - `df.corr()`: Обчислює кореляційну матрицю для всього DataFrame, де кожен елемент представляє коефіцієнт кореляції між двома змінними. Значення кореляції може бути в межах від -1 до 1, де значення близьке до 1 вказує на позитивну кореляцію, а значення близьке до -1 - на негативну кореляцію.
2. Кореляція з певним стовпцем:
    - `df['column1'].corr(df['column2'])`: Обчислює кореляцію між двома конкретними стовпцями. Ви можете замінити `'column1'` і `'column2'` на назви ваших власних стовпців.
3. Теплова карта кореляції:
    - `import seaborn as sns` (імпортуємо бібліотеку Seaborn)
    - `sns.heatmap(df.corr(), annot=True)`: Створює теплову карту, яка візуалізує кореляційну матрицю. За допомогою параметра `annot=True` будуть відображені значення кореляції на самій тепловій карті.

Аналіз кореляції дозволяє виявити, які змінні сильно взаємозв'язані між собою, що може допомогти у встановленні патернів, прогнозуванні та видаленні зайвих змінних. Зазначимо, що кореляція не завжди означає причинно-наслідковий зв'язок між змінними, а лише вказує на статистичну залежність між ними.