Pandas Python Library (начало)

Pandas - это пакет Python с открытым исходным кодом, который предоставляет множество инструментов для анализа данных. Пакет поставляется с несколькими структурами данных, которые можно использовать для множества различных задач манипулирования данными. Он также имеет множество методов, которые можно использовать для анализа данных, что очень удобно при работе с данными и проблемами машинного обучения в Python.

Преимущества использования Pandas

- может представлять данные таким образом, чтобы он подходил для анализа данных через структуры данных Series и DataFrame;
- пакет содержит несколько методов для удобной фильтрации данных;
- у Pandas есть множество утилит для плавного выполнения операций ввода/вывода. Он может читать данные из различных форматов, таких как CSV, TSV, MS Excel и т. д.

Установка Pandas

Стандартный дистрибутив Python не поставляется с модулем Pandas. Чтобы использовать этот сторонний модуль, вы должны установить его.

Приятной особенностью Python является то, что он поставляется в комплекте с инструментом под названием pip, который можно использовать для установки Pandas. Для выполнения установки вам необходимо выполнить следующую команду:

```
pip install pandas
```

```
conda install pandas
```

Pandas Data Structures

Pandas имеет две основные структуры данных для хранения данных:

1. Series
2. DataFrame

Series

Series похож на одномерный массив. Он может хранить данные любого типа. Значения серии Pandas являются изменяемыми, но размер серии является неизменным и не может быть изменен.

Первому элементу в серии присваивается индекс 0, а последнему элементу присваивается индекс N-1, где N - общее количество элементов в серии.

Чтобы создать серию Pandas, мы должны сначала импортировать пакет Pandas через

```
import pandas as pd
```

```
series1 = pd.Series([1,2,3,4])
print(series1)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

Видно, что у нас есть два столбца, первый с номерами, начиная с индекса 0, а второй с элементами, которые были добавлены в серию.

Первый столбец обозначает индексы для элементов.

Тем не менее, можно получить сообщение об ошибке при попытке отображения Series. Основная причина этой ошибки заключается в том, что Pandas ищет объем отображаемой информации, поэтому вы должны предоставить системную информацию для вывода.

Можно устранить ошибку, выполнив код следующим образом:

```
import pandas as pd
import sys

sys.__stdout__ = sys.stdout

series1 = pd.Series([1,2,3,4])
print(series1)
```

Серия также может быть создана из массива NumPy. Давайте создадим массив numpy, а затем преобразуем его в серию Pandas:

```
import pandas as pd
import numpy as np
import sys

sys.__stdout__ = sys.stdout

fruits = np.array(['apple','orange','mango','pear'])
series2 = pd.Series(fruits)
print(series2)
```

Output:

```
0     apple
1    orange
2     mango
3      pear
dtype: object
```

Сначала идет импорт необходимых библиотек, в том числе и numpy. Затем мы вызвали функцию `array()` numpy для создания массива фруктов. Затем мы используем функцию Pandas `Series()` и передаем ей массив, который мы хотим преобразовать в серию. Наконец, мы вызываем функцию `print()` для отображения Series.

DataFrame

DataFrame Pandas можно рассматривать как таблицу. Он организует данные в строки и столбцы, что делает его двумерной структурой

данных. Потенциально столбцы имеют другой тип, а размер DataFrame является изменяемым и, следовательно, может быть изменен.

Чтобы создать DataFrame, вы можете создать "с нуля" или преобразовать другие структуры данных, такие как массивы Numpy, в DataFrame. Вот как можно создать DataFrame с нуля:

```
import pandas as pd
df = pd.DataFrame({
    "Column1": [1, 4, 8, 7, 9],
    "Column2": ['a', 'column', 'with', 'a', 'string'],
    "Column3": [1.23, 23.5, 45.6, 32.1234, 89.453],
    "Column4": [True, False, True, False, True]
})
print(df)
```

Output:

```
Column1 Column2  Column3  Column4
0        1       a   1.2300     True
1        4  column  23.5000    False
2        8    with  45.6000     True
3        7       a  32.1234    False
4        9  string  89.4530     True
```

В этом примере мы создали DataFrame с именем df. Первый столбец DataFrame имеет целочисленные значения. Второй столбец имеет строку, третий столбец имеет значения с плавающей запятой, а четвертый столбец имеет логические значения.

Оператор `print(df)` отобразит нам содержимое DataFrame через консоль, что позволит нам проверить его содержимое.

Однако при отображении DataFrame вы можно заметить, что в начале таблицы есть дополнительный столбец, элементы которого начинаются с 0. Этот столбец создается автоматически и помечает индексы строк.

Чтобы создать DataFrame, мы должны вызвать метод `pd.DataFrame()`, как показано в примере выше.

Можно создать DataFrame из списка или даже набора списков. Нам нужно только вызвать метод `pd.DataFrame()` и затем передать ему переменную списка в качестве единственного аргумента.

Рассмотрим следующий пример:

```
import pandas as pd
mylist = [4, 8, 12, 16, 20]
df = pd.DataFrame(mylist)
print(df)
```

Output:

```
0
0   4
1   8
2  12
3  16
4  20
```

В этом примере мы создали список с именем `mylist` с последовательностью из 5 целых чисел. Затем мы вызвали метод `DataFrame()` и передали ему имя списка в качестве аргумента. Произошло преобразование списка в DataFrame.

Затем мы вывели содержимое DataFrame. DataFrame имеет столбец по умолчанию, в котором отображаются индексы, причем первый элемент имеет индекс 0, а последний - индекс N-1, где N - общее количество элементов в DataFrame.

Вот еще один пример:

```
import pandas as pd
items = [['Phone', 2000], ['TV', 1500], ['Radio', 800]]
df = pd.DataFrame(items, columns=['Item', 'Price'], dtype=float)
print(df)
```

Output:

```
Item   Price
0  Phone  2000.0
1     TV  1500.0
2  Radio   800.0
```

Здесь мы создали список именованных предметов с набором из 3 предметов. Для каждого товара у нас есть название и цена. Затем этот список передается методу DataFrame () для его преобразования в объект DataFrame.

В этом примере также указаны имена столбцов для DataFrame. Числовые значения также были преобразованы в значения с плавающей запятой, поскольку мы указали аргумент dtype как «float».

Чтобы получить сводку данных этого элемента, мы можем вызвать метод `describe()` для переменной DataFrame, то есть df:

```
df.describe()
```

Output:

```
Price
count     3.000000
mean   1433.333333
std     602.771377
min     800.000000
25%    1150.000000
50%    1500.000000
75%    1750.000000
max    2000.000000
```

Функция `describe()` возвращает некоторые общие статистические детали данных, включая среднее, стандартное отклонение, минимальный элемент, максимальный элемент и некоторые другие детали. Это отличный способ получить снимок данных, с которыми вы работаете, если набор данных относительно вам неизвестен. Это также может быть хорошим способом быстрого сравнения двух отдельных наборов данных схожих данных.

Импорт данных

Часто приходится использовать Pandas для анализа данных, которые хранятся в файле Excel или в файле CSV. К счастью, Pandas предоставляет нам множество методов, которые мы можем использовать для загрузки данных из таких источников в Pandas DataFrame.

Импорт данных CSV

Для этого примера мы создадим CSV-файл с именем cars.csv. Файл должен содержать следующие данные:

```
Number,Type,Capacity
SSD,Premio,1800
KCN,Fielder,1500
USG,Benz,2200
TCH,BMW,2000
KBQ,Range,3500
TBD,Premio,1800
KCP,Benz,2200
USD,Fielder,1500
UGB,BMW,2000
TBG,Range,3200
```

Pandas предоставляет нам метод `read_csv`, который можно использовать для чтения значений CSV в DataFrame Pandas. Метод принимает путь к файлу CSV в качестве аргумента.

Следующий код - показвает как прочитать файл cars.csv:

```
import pandas as pd
data = pd.read_csv('cars.csv')
print(data)
```

Output:

```
Number     Type  Capacity
0    SSD   Premio      1800
1    KCN  Fielder      1500
2    USG     Benz      2200
3    TCH      BMW      2000
4    KBQ    Range      3500
5    TBD   Premio      1800
6    KCP     Benz      2200
7    USD  Fielder      1500
8    UGB      BMW      2000
9    TBG    Range      3200
```

В некоторых случаях в вашем наборе данных могут быть тысячи строк. В таком случае было бы более полезно печатать только первые несколько строк на консоли, а не печатать все строки.
Это можно сделать, вызвав метод `head()` в DataFrame, как показано ниже:

```
data.head()
```

Для наших данных выше, приведенная выше команда возвращает только первые 5 строк набора данных, что позволяет вам проверить небольшую выборку данных как показано ниже:

```
Number     Type  Capacity
0    SSD   Premio      1800
1    KCN  Fielder      1500
2    USG     Benz      2200
3    TCH      BMW      2000
4    KBQ    Range      3500
```

Метод `loc()` - полезная утилита, которая помогает нам читать только определенные строки определенного столбца в наборе данных, как показано в следующем примере:

```
import pandas as pd
data = pd.read_csv('cars.csv')

print (data.loc[[0, 4, 7], ['Type']])
```

Output:

```
Type
0   Premio
4    Range
7  Fielder
```

Здесь мы использовали метод loc () для чтения элементов только с индексами 0, 4 и 7 столбца Type.

Иногда может понадобиться прочитать только определенные столбцы. Это можно сделать с помощью метода `loc()`, как показано ниже в этом примере:

```
import pandas as pd
data = pd.read_csv('cars.csv')

print (data.loc[:, ['Type', 'Capacity']])
```

```
Type  Capacity
0   Premio      1800
1  Fielder      1500
2     Benz      2200
3      BMW      2000
4    Range      3500
5   Premio      1800
6     Benz      2200
7  Fielder      1500
8      BMW      2000
9    Range      3200
```

Здесь мы использовали метод loc () для чтения всех строк (the: part) только двух наших столбцов из набора данных, то есть столбцов типа и емкости, как указано в аргументе.

Импорт данных Excel

В дополнение к методу `read_csv`, Pandas также имеет функцию `read_excel`, которую можно использовать для чтения данных Excel в Pandas DataFrame. В этом примере мы будем использовать файл Excel с именем `worker.xlsx` с подробной информацией о работниках в компании.

Следующий код можно использовать для загрузки содержимого файла Excel в Pandas DataFrame:

```
import pandas as pd
data = pd.read_excel('workers.xlsx')
print (data)
```

```
ID    Name      Dept  Salary
0   1    John       ICT    3000
1   2    Kate   Finance    2500
2   3  Joseph        HR    3500
3   4  George       ICT    2500
4   5    Lucy     Legal    3200
5   6   David   Library    2000
6   7   James        HR    2000
7   8   Alice  Security    1500
8   9   Bosco   Kitchen    1000
9  10    Mike       ICT    3300
```

После вызова метода `read_excel` мы передали имя файла в качестве аргумента, `read_excel` читает и анализирует данные. Как и в нашем примере с CSV, эта функция может быть объединена с методом `loc()`, чтобы помочь нам прочитать определенные строки и столбцы из файла Excel.

Например:

```
import pandas as pd
data = pd.read_excel('workers.xlsx')

print (data.loc[[1,4,7],['Name','Salary']])
```

```
Name  Salary
1   Kate    2500
4   Lucy    3200
7  Alice    1500
```

Мы использовали метод `loc()` для получения значений Name и Salary элементов по индексам 1, 4 и 7.

Pandas также позволяет нам читать с двух листов Excel одновременно. Предположим, что наши предыдущие данные находятся на "Листе 1", а у нас есть некоторые другие данные на "Листе 2" того же файла Excel. Следующий код показывает, как можно прочитать с двух листов одновременно:

```
import pandas as pd
with pd.ExcelFile('workers.xlsx') as x:
    s1 = pd.read_excel(x, 'Sheet1')
    s2 = pd.read_excel(x, 'Sheet2')

print("Sheet 1:")
print (s1)
print("")
print("Sheet 2:")
print (s2)
```

```
Sheet 1:
   ID    Name      Dept  Salary
0   1    John       ICT    3000
1   2    Kate   Finance    2500
2   3  Joseph        HR    3500
3   4  George       ICT    2500
4   5    Lucy     Legal    3200
5   6   David   Library    2000
6   7   James        HR    2000
7   8   Alice  Security    1500
8   9   Bosco   Kitchen    1000
9  10    Mike       ICT    3300

Sheet 2:
   ID    Name  Age  Retire
0   1    John   55    2023
1   2    Kate   45    2033
2   3  Joseph   55    2023
3   4  George   35    2043
4   5    Lucy   42    2036
5   6   David   50    2028
6   7   James   30    2048
7   8   Alice   24    2054
8   9   Bosco   33    2045
9  10    Mike   35    2043
```

Обработка данных

Обработка данных - это процесс обработки данных для подготовки их к использованию на следующем этапе. Примеры процессов обработки данных включают слияние, группирование и объединение. Подобные манипуляции часто необходимы в data science, чтобы привести ваши данные в форму, которая хорошо работает с любым анализом или алгоритмами, которые вы собираетесь применять.

Обьединение

Библиотека Pandas позволяет нам присоединяться к объектам DataFrame с помощью функции `merge()`. Давайте создадим два DataFrames и попробуем их объединить.

Вот первый DataFrame, df1:

```
import pandas as pd

d = {
    'subject_id': ['1', '2', '3', '4', '5'],
    'student_name': ['John', 'Emily', 'Kate', 'Joseph', 'Dennis']
}
df1 = pd.DataFrame(d, columns=['subject_id', 'student_name'])
print(df1)
```

```
subject_id student_name
0          1         John
1          2        Emily
2          3         Kate
3          4       Joseph
4          5       Dennis
```

Код для создания второго DataFrame, df2:

```
import pandas as pd

data = {
    'subject_id': ['4', '5', '6', '7', '8'],
    'student_name': ['Brian', 'William', 'Lilian', 'Grace', 'Caleb']
}
df2 = pd.DataFrame(data, columns=['subject_id', 'student_name'])
print(df2)
```

```
subject_id student_name
0          4        Brian
1          5      William
2          6       Lilian
3          7        Grace
4          8        Caleb
```

Теперь нам нужно объединить два DataFrames, то есть df1 и df2, по значениям subject_id. Мы просто вызываем функцию `merge()`, как показано ниже:

```
pd.merge(df1, df2, on='subject_id')
```

```
subject_id student_name_x student_name_y
0          4         Joseph          Brian
1          5         Dennis        William
```

Есть много других способов использования функции `pd.merge()`, которые мы не будем рассматривать в этой статье, например, какие

данные должны быть объединены, как они должны быть объединены, если они должны быть отсортированы и т. д.

Группирование

Группировка - это процесс помещения данных в различные категории. Вот простой пример:

```
import pandas as pd

raw = {
    'Name': ['John', 'John', 'Grace', 'Grace', 'Benjamin', 'Benjamin',
'Benjamin',
        'Benjamin', 'John', 'Alex', 'Alex', 'Alex'],
    'Position': [2, 1, 1, 4, 2, 4, 3, 1, 3, 2, 4, 3],
    'Year': [2009, 2010, 2009, 2010, 2010, 2010, 2011, 2012, 2011, 2013,
2013, 2012],
    'Marks':[408, 398, 422, 376, 401, 380, 396, 388, 356, 402, 368, 378]
}
df = pd.DataFrame(raw)

group = df.groupby('Year')
print(group.get_group(2010))
```

```
Marks      Name  Position  Year
1    398      John         1  2010
3    376     Grace         4  2010
5    380  Benjamin         4  2010
```

В этом простом примере мы сгруппировали данные по годам, в данном случае это был 2010 год. Мы могли бы также сгруппировать любые другие столбцы, например «Имя», «Позиция» и т. д.

Конкатенация

Конкатенация - объединение данных, которое в основном означает добавление одного набора данных к другому, может быть выполнено путем вызова функции `concat()`.

Пример, как объединить DataFrames, используя два предыдущих Dataframes, каждый с двумя столбцами «subject_id» и «student_name»:

```
print(pd.concat([df1, df2]))
```

```
subject_id student_name
0          1         John
1          2        Emily
2          3         Kate
3          4       Joseph
4          5       Dennis
0          4        Brian
1          5      William
2          6       Lilian
3          7        Grace
4          8        Caleb
```

Descriptive Statistics

Как я кратко показал ранее, когда мы используем метод `description()`, мы получаем статистику для числовых столбцов, но символьные столбцы исключаются.

Давайте сначала создадим DataFrame, показывающий имена учеников и их оценки по математике и английскому языку:

```
import pandas as pd

data = {
    'Name': ['John', 'Alice', 'Joseph', 'Alex'],
    'English': [64, 78, 68, 58],
    'Maths': [76, 54, 72, 64]
}

df = pd.DataFrame(data)
print(df)
```

```
English  Maths    Name
0       64     76    John
1       78     54   Alice
2       68     72  Joseph
3       58     64    Alex
```

Нам нужно только вызвать функцию `describe()` в DataFrame и получить различные показатели, такие как среднее значение, стандартное отклонение, медиана, максимальный элемент, минимальный элемент и т. д.

```
df.describe()
```

```
English      Maths
count   4.000000   4.000000
mean   67.000000  66.500000
std     8.406347   9.712535
min    58.000000  54.000000
25%    62.500000  61.500000
50%    66.000000  68.000000
75%    70.500000  73.000000
max    78.000000  76.000000
```

Метод `describe()` полностью проигнорировал столбец «Name», поскольку он не является числовым, что нам и нужно. Это упрощает задачу для вызывающей стороны, поскольку вам не нужно беспокоиться об удалении нечисловых столбцов перед вычислением требуемой числовой статистики.

Заключение

Pandas - чрезвычайно полезная библиотека Python, особенно для науки о данных. Различные функции Pandas делают предобработку данных чрезвычайно простой. В этой статье дается краткое введение в основные функции библиотеки. В этой статье мы увидели рабочие примеры всех основных утилит библиотеки Pandas. Чтобы максимально использовать возможности Pandas, я бы посоветовал вам попрактиковаться в примерах, приведенных в этой статье, а также протестировать библиотеку с собственными наборами данных. Удачного кодирования!

P.S. Подписываемся, ставим лайки, нажимаем на колокольчик 😃