Python ile Veri Analizi

Pandalar Bölüm-2

Yapılacaklar: Aktivitenin sonuna kadar gidin

Normal İfadeler (REGEX)

- Normal ifade (normal ifade olarak kısaltılır) , metinde bir arama düzenini belirten bir karakter dizisidir. Genellikle bu tür modeller, dizi arama algoritmaları tarafından dizilerdeki "bul" veya "bul ve değiştir" işlemleri veya girdi doğrulaması için kullanılır.

`'\'`Normal ifadeler , özel biçimleri belirtmek veya özel anlamlarını çağırmadan özel karakterlerin kullanılmasına izin vermek için ters eğik çizgi karakterini ( ) kullanır. Python'un düzenli ifade kalıpları için ham dize gösterimi, özel karakterlerin kullanılması sorununu çözer. ve `r"\n"` içeren iki karakterli bir dize iken , yeni bir satır içeren tek karakterli bir dizedir.`'\''n'"\n"`

Ortak Desenler

`\d`ile arasında herhangi bir sayısal rakam `0.9`

`\D` Ondalık basamak olmayan herhangi bir karakterle eşleşir. Bu, 'in tam tersidir `\d`.

`\w` Herhangi bir harf, sayısal rakam veya alt çizgi karakteri. (Bunu eşleşen "kelime" karakterleri olarak düşünün.)

`\W` Harf, sayısal rakam veya alt çizgi karakteri olmayan herhangi bir karakter.

`\s` Herhangi bir boşluk, sekme veya yeni satır karakteri. (Bunu beyaz boşluk karakterleriyle eşleşen olarak düşünün.)

`\S` Boşluk, sekme veya yeni satır olmayan herhangi bir karakter.

Ortak Meta Karakterler

`"[]"` Bir dizi karakter `"[a-m]"`

`"\"` Özel bir diziyi işaret eder (özel karakterlerden kaçmak için de kullanılabilir)

`"."` Herhangi bir karakter (yeni satır karakteri hariç)

`"^"` İle başlar `"^hello"`

"$" ile biter "world$"

"*" Sıfır veya daha fazla olay

"+" Bir veya daha fazla olay

"{}" Tam olarak belirtilen olay sayısı

"|" Ya ya da "falls|stays"

"()" Yakala ve gruplandır

Ham Dize ("r / R")

- Python raw string is created by prefixing a string literal with 'r' or 'R'.
- Python raw string treats backslash (\) as a literal character. This is useful when we want to have a string that contains backslash and don't want it to be treated as an escape character

print("Backslash: \\") ------ Backslash: \
print(r"Backslash: \\") ----- Backslash: \\

Common Python RegEx Functions

re.search(pattern, string, flags=0)

Scan through string looking for a match to the pattern, returning a Match object, or None if no match was found

text = "A78L41K"
num = re.search("78", text)
num.group() ---->> '78'

re.match(pattern, string, flags=0)

Try to apply the pattern at the start of the string, returning a Match object, or None if no match was found.

If you want to locate a match anywhere in string, use search() instead of match()

text = "A78L41K"
alpha = re.match("\D\d\d", text)
alpha.group() ----->> 'A78'

re.fullmatch(pattern, string, flags=0)

Try to apply the pattern to all of the string

alpnum = re.fullmatch("\D\d+\D\d+\D", text)
alpnum.group() --->> 'A78L41K'

re.findall(pattern, string, flags=0)

Return a list of all non-overlapping matches in the string.

text = "O 1, t 10, o 100. 100000"
re.findall("\d{1}", text) ---------->> ['1', '1', '0', '1', '0', '0', '1', '0', '0', '0', '0', '0']

`re.sub(pattern, repl, string, count=0, flags=0)`

Return the string obtained by replacing the leftmost non-overlapping occurrences of the pattern in string by the replacement repl. repl can be either a string or a callable; if a string, backslash escapes in it are processed. If it is a callable, it's passed the Match object and must return a replacement string to be used.

text = "2004-959-559 # This is Phone Number"
re.sub("\D", "", text) ----->> '2004959559'

`re.split(pattern, string, maxsplit=0, flags=0)`

Split the source string by the occurrences of the pattern, returning a list containing the resulting substrings.

text = "ab56cd78_de fg3hıi49"
re.split("\D+", text) ------>> ['', '56', '78', '3', '49']

Pandas Functions Accepting RegEx

s = pd.Series(['a3aa', 'b4aa', 'c5aa'])

print(s)

```
0    a3aa
1    b4aa
2    c5aa
```

- **count()**: Count occurrences of pattern in each string of the Series/Index

`pandas.Series.str.count(pat, flags=0)`

Count occurrences of pattern in each string of the Series/Index.

This function is used to count the number of times a particular regex pattern is repeated in each of the string elements of the Series.

```
s.str.count('a')
```

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

- **replace()**: Replace the search string or pattern with the given value

`pandas.Series.str.replace(pat, repl, n=- 1, case=None, flags=0, regex=None)`

Replace each occurrence of pattern/regex in the Series/Index.

Equivalent to str.replace() or re.sub(), depending on the regex value.

```
s.str.replace('\d','R', regex=True)
```

```
0    aRaa
1    bRaa
2    cRaa
dtype: object
```

- **contains():** Test if pattern or regex is contained within a string of a Series or Index. Calls re.search() and returns a boolean

```
pandas.Series.str.contains(pat, case=True, flags=0, na=None, regex=True)
```

Test if pattern or regex is contained within a string of a Series or Index.

Return boolean Series or Index based on whether a given pattern or regex is contained within a string of a Series or Index.

```
s.str.contains('b')
```

```
0    False
1     True
2    False
dtype: bool
```

- **findall():** Find all occurrences of pattern or regular expression in the Series/Index. Equivalent to applying re.findall() on all elements

```
pandas.Series.str.findall(pat, flags=0)
```

Find all occurrences of pattern or regular expression in the Series/Index.

Equivalent to applying re.findall() to all the elements in the Series/Index.

```
s = pd.Series(['a3a8a', 'b4a7a', 'c5a6a'])
s
```

```
0    a3a8a
1    b4a7a
2    c5a6a
dtype: object
```

```
s.str.findall('\d')
```

```
0    [3, 8]
1    [4, 7]
2    [5, 6]
dtype: object
```

- **match():** Determine if each string matches a regular expression. Calls re.match() and returns a boolean

```
pandas.Series.str.match(pat, case=True, flags=0, na=None)
```

Determine if each string starts with a match of a regular expression.

```
s = pd.Series(['&a3a8a', 'b4a&7a', 'c5a6a'])
s
```

```
0    &a3a8a
1    b4a&7a
2     c5a6a
dtype: object
```

```
s.str.match('&')
```

```
0     True
1    False
2    False
dtype: bool
```

- **split():** Split strings around given separator/delimiter and accepts string or regular expression to split on

```
pandas.Series.str.split(pat=None, n=- 1, expand=False, *, regex=None)
```

Split strings around given separator/delimiter.

Splits the string in the Series/Index from the beginning, at the specified delimiter string.

```
s = pd.Series(['&a3a8a', 'b4a&7a', 'c5a6a'])
s
```

```
0    &a3a8a
1    b4a&7a
2     c5a6a
dtype: object
```

```
s.str.split('&')
```

```
0    [, a3a8a]
1    [b4a, 7a]
2      [c5a6a]
dtype: object
```

- **extract():** Extract capture groups in the regex pat as columns in a DataFrame and returns the captured groups

```
pandas.Series.str.extract(pat, flags=0, expand=True)
```

Extract capture groups in the regex pat as columns in a DataFrame.

For each subject string in the Series, extract groups from the first match of regular expression pat.

```
s = pd.Series(['&a3a+?8a', 'b4a&+^7a', 'c5a6a'])
s
```

```
0    &a3a+?8a
1    b4a&+^7a
2       c5a6a
dtype: object
```

```
s.str.extract('(\w+)')
```

	0
0	a3a
1	b4a
2	c5a6a

Bu soru buraya gömülemez.

Bu soru buraya gömülemez.

Previous

Next

You have completed 77% of the lesson

77%

Jump to...

Bu sayfada kullanıcı turunu sıfırla