03

Implementando a distância entre usuários no nosso dataset

Transcrição

Continuando o nosso projeto, já aprendemos a calcular a distância entre dois usuários com base nas notas que eles deram aos filmes. De volta ao conjunto de dados que nomeamos como notas, temos a informação de que o usuário 1 avaliou vários filmes com diversas notas, e o mesmo para os demais usuários.

Se quisermos calcular a distância entre o usuário 1 e o usuário 4, primeiro teremos que extrair as notas de cada um deles para então efetuarmos essa operação.

Portanto, vamos fazer notas.query("usuarioId==1") para filtrarmos apenas as notas do usuário 1, e teremos como resultado uma series de 16 avaliações que atribuiremos a uma variável notas_do_usuario. Se essas são as notas do usuário 1, sabemos que não iremos precisar da coluna usuarioId, assim como não precisaremos da coluna momento. Portanto, filtraremos somente as colunas que nos serão úteis (filmeId e nota), passando-as dentro de uma lista.

Também podemos nos livrar do índice, já que, na prática, quando estamos trabalhando com dataframes, é interessante utilizarmos como índice aquilo que faz sentido naturalmente. Nesse caso, para as avaliações de um usuário, só teremos uma nota para cada filme, e portanto podemos definir filmeId como o próprio índice. Faremos isso com set_index("filmeId"). Vamos sobrescrever a variável notas_do_usuario com essas alterações.

Uma das vantagens de trabalharmos dessa forma é a possibilidade de buscarmos um elemento diretamente pelo seu índice (por exemplo, com notas_do_usario.loc[]), sem a necessidade de utilizarmos uma query().

Agora, queremos ser capazes de generalizar essa construção para qualquer usuário do conjunto, e não somente o usuário 1. Para isso, definiremos a função notas_do_usuario() recebendo usuario como parâmetro. O retorno dessa função será uma variável notas_do_usuario.

O identificador do usuário na query() não poderá ficar fixo. Portanto, faremos notas.query("usuarioId==%d" % usuario) para termos um inteiro interpolado com o id do usuário.

Após executarmos essa célula, poderemos fazer notas_do_usuario(1) para imprimir aquelas 16 notas do primeiro usuário. Poderemos repetir esse processo para qualquer usuário listado no conjunto.

Com isso, conseguiremos definir uma variável usuario1 como sendo o retorno de notas_do_usuario(1); e usuario4 como sendo o retorno de notas_do_usuario(4).

Temos, então, dois dataframes que queremos juntar. Para isso, faremos usuario1.join(usuario4). A função join(), por padrão, utiliza o índice para fazer essa junção, criando três "colunas": o índice (que é o filme), a nota do usuário 1 e a nota do usuário 4.

Porém, se rodarmos essa função, receberemos uma mensagem de erro indicando que as colunas se sobrepõem. Isso porque existe a coluna "nota" nos dois dataframes. Teremos, então, que especificar um sufixo para ser adicionado ao nome dessa coluna.

Portanto, faremos `usuario1.join(usuario4, lsuffix="_esquerda", rsuffix="_direita")`. Dessa forma, estaremos especificando que o usuário da esquerda ("l" de "left", que é o usuario1) receberá o sufixo `_esquerda`, e o usuário da direita ("r" de "right", o usuario4) receberá o sufixo `_direita`.

Agora, além da coluna filmeId, teremos as colunas nota_esquerda e nota_direita. Nesse dataframe, temos uma linha para cada filme que o usuario assistiu, e as notas de cada um dos usuários. Quando não foram encontradas notas do usuario4, como em 307 e 1257, temos NaN, que significa "not a number".

Note que, se existirem filmes que somente o usuario4 assistiu, eles não aparecerão. Mas não tem problema, pois realmente queremos comparar somente os filmes que ambos os usuários assistiram, removendo aqueles que foram assistidos por apenas um deles. Inclusive, vamos executar um dropna() para remover as linhas com NaN, limpando os nossos dados. Atribuiremos esse conjunto a uma variável diferencas.

Podemos, então, fazer diferencas['nota_esquerda] e diferencias[nota_direita] para obtermos os vetores de cada usuário, o que nos permitirá calcular a distância entre eles.

Vamos redefinir a função distancia(a,b) para distancia_de_vetores(a,b). Então, calcularemos distancia_de_vetores(diferencas['nota_esquerda], diferencias[nota_direita]). Não importa se esse vetor tem duas dimensões ou quinze, a função executará o cálculo da mesma forma.

Como resultado, teremos que a distância entre o usuario1 e o usuario 4 é 3.04. Agora que já temos como calcular a distância entre dois usuários, vamos criar uma função que facilitará esse processo.

Definiremos, então, a função distancia_de_usuarios(usuario_id1, usuario_id2), que claramente recebe o id de dois usuários. Primeiro, precisaremos pegar as notas desses usuários, e felizmente já criamos uma função para isso. Portanto, faremos notas1 = notas_do_usuario(usuario_id1) e notas 2 = notas_do_usuario(usuario_id2).

Em seguida, calcularemos as diferencas, ou seja, juntaremos os dois dataframes e removeremos as linhas que não possuem valores nas duas colunas. Faremos isso com `notas1.join(notas2, lssufix="_esquerda", rsuffix="_direita").dropna()`. Por fim, retornaremos a distancia_de_vetores() baseada nessas diferenças.

Agora, se executarmos distandia_de_usuarios(1,4), receberemos o mesmo valor que encontramos anteriormente: 3.04. Portanto, já somos capazes de, com apenas uma função, calcular a distância entre dois usuários quaisquer dentro do nosso dataset do MovieLens, ou de outro sistema que contenha algum tipo de avaliação.

Existem várias funções de distância que podemos implementar, e ainda iremos refinar esta a medida em que construirmos nosso algoritmo.