20c53b4a-81d8-4d18-bc38-d36d30ae6fba

January 5, 2025

Olá,

Meu nome é Ramon. Ao ao longo do texto farei algumas observações sobre melhorias no código. Nas partes em que não houver comentários, está tudo certo, ok? Estarei aberto a feedbacks e discussões sobre o tema.

Peço que mantenha e não altere os comentários que eu fizer por aqui para que possamos nos localizar posteriormente, ok?

Mais uma coisa, vamos utilizar um código de cores para você entender os meus feedbacks no seu notebook. Funciona assim:

Comentário do revisor:

Sucesso. Tudo foi feito corretamente.

Comentário do revisor:

Alerta não crítco, mas que pode ser corrigido para melhoria geral no seu código/análise

Comentário do revisor:

Erro que precisa ser consertado, caso contrário seu projeto **não** será aceito.

Você pode me responder usando isso:

Resposta do Aluno.

0.0.1

Comentário Geral do Revisor

Obrigado por enviar seu projeto.

Parabéns, você fez um bom trabalho.

Como pontos fortes do projeto, destaco: - Código simples e eficiente (sem loop desnecessário) - Uso adequado dos recursos da lição

Há um pequeno erro de lógica. Deixei dica. Não vamos fazer nova iteração pois é algo pequeno e você acertou a tarefa em uma célula posterior.

Desejo sucesso na jornada. Parabéns pelo empenho.

1 Se liga na música

2 Conteúdo

2.1 Introdução

O trabalho de um analista é analisar dados para obter percepções valiosas dos dados e tomar decisões fundamentadas neles. Esse processo consiste em várias etapas, como visão geral dos dados, pré-processamento dos dados e testes de hipóteses.

Sempre que fazemos uma pesquisa, precisamos formular uma hipótese que depois poderemos testar. Às vezes nós aceitamos essas hipóteses; outras vezes, nós as rejeitamos. Para fazer as escolhas certas, um negócio deve ser capaz de entender se está fazendo as suposições certas ou não.

Neste projeto, você vai comparar as preferências musicais dos habitantes de Springfild e Shelbyville. Você vai estudar os dados de um serviço de streaming de música online para testar a hipótese apresentada abaixo e comparar o comportamento dos usuários dessas duas cidades.

2.1.1 Objetivo:

Teste a hipótese: 1. A atividade dos usuários é diferente dependendo do dia da semana e da cidade.

2.1.2 Etapas

Os dados sobre o comportamento do usuário são armazenados no arquivo `/datasets/music_project_en.csv`. Não há informações sobre a qualidade dos dados, então será necessário examiná-los antes de testar a hipótese.

Primeiro, você avaliará a qualidade dos dados e verá se seus problemas são significativos. Depois, durante o pré-processamento dos dados, você tentará tratar dos problemas mais críticos.

O seu projeto consistirá em três etapas: 1. Visão geral dos dados 2. Pré-processamento de dados 3. Teste da hipótese

Voltar ao Índice

2.2 Etapa 1. Visão geral dos dados

Abra os dados e examine-os.

Você precisará da `pandas`, então, importe-a.

[14]:
```
# importando pandas
import pandas as pd
```

Leia o arquivo `music_project_en.csv` da pasta `/datasets/` e salve-o na variável `df`:

[15]:
```
# lendo o arquivo e armazenando em df
df = pd.read_csv('/datasets/music_project_en.csv')
```

Imprima as primeiras 10 linhas da tabela:

[16]:
```
# obtenha as 10 primeiras 10 linhas da tabela df
print(df.head(10))
```

```
     userID                        Track            artist   genre  \
0  FFB692EC            Kamigata To Boots  The Mass Missile    rock
1  55204538  Delayed Because of Accident  Andreas Rönnberg    rock
2    20EC38            Funiculì funiculà       Mario Lanza     pop
3  A3DD03C9        Dragons in the Sunset        Fire + Ice    folk
4  E2DC1FAE                   Soul People        Space Echo   dance
5  842029A1                        Chains          Obladaet  rusrap
6  4CB90AA5                          True      Roman Messer   dance
7  F03E1C1F             Feeling This Way   Polina Griffith   dance
8  8FA1D3BE                      L'estate       Julia Dalia  ruspop
9  E772D5C0                    Pessimist               NaN   dance

        City        time        Day
0  Shelbyville  20:28:33  Wednesday
1  Springfield  14:07:09     Friday
2  Shelbyville  20:58:07  Wednesday
3  Shelbyville  08:37:09     Monday
4  Springfield  08:34:34     Monday
5  Shelbyville  13:09:41     Friday
6  Springfield  13:00:07  Wednesday
7  Springfield  20:47:49  Wednesday
8  Springfield  09:17:40     Friday
9  Shelbyville  21:20:49  Wednesday
```

Obtenha informações gerais sobre a tabela usando um comando. Você conhece o método para exibir informações gerais que precisamos obter.

[17]:
```
# obtendo informações gerais sobre os nossos dados
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65079 entries, 0 to 65078
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
```

```
---  ------    --------------  -----
 0     userID  65079 non-null  object
 1   Track     63736 non-null  object
 2   artist    57512 non-null  object
 3   genre     63881 non-null  object
 4     City    65079 non-null  object
 5   time      65079 non-null  object
 6   Day       65079 non-null  object
dtypes: object(7)
memory usage: 3.5+ MB
```

Aqui estão as nossas observações sobre a tabela. Ela contém sete colunas. Elas armazenam o mesmo tipo de dado: `object`.

De acordo com a documentação: - `' userID'` — identificação do usuário - `'Track'` — título da música - `'artist'` — nome do artista - `'genre'` — gênero da música - `'City'` — cidade do usuário - `'time'` — o tempo exato que a música foi reproduzida - `'Day'` — dia da semana

Podemos ver três problemas de estilo nos cabeçalhos da tabela: 1. Alguns cabeçalhos são escritos em letras maiúsculas, outros estão em minúsculas. 2. Alguns cabeçalhos contêm espaços. 3. `'Alguns cabeçalhos não estão muito bem descritos, como: 'city', não está explicando se é a cidade do artista, do usuário ou onde a música foi criada... 'time', não exemplifica sobre que tempo está se referindo.`

2.2.1 Escreva suas observações. Aqui estão algumas perguntas que podem ajudar:

`1. Que tipo de dados temos nas linhas? E como podemos entender as colunas?`

' Os dados que obtivemos nas linhas são dados de um determinado usuário de uma das duas cidades, que escutou um certo gênero musical em um determinado dia e horário da semana. E sobre as colunas entendemos que cada uma nos mostra um tipo de dado referente aquele respectivo usuário.

`2. Esses dados são suficientes para responder à nossa hipótese ou precisamos de mais dados?`

'Os dados que possuímos são os suficientes para fazermos a hipótese que desejamos, onde tais gêneros musicais são mais tocadas em tais cidades, em qual dia da semana os usuários usam mais a plataforma de streaming, podemos comparar qual gênero musical é mais escutado em cada cidade, porém, para obtermos todos esses dados precisos, precisamos fazer um pré-processamento de dados.

`3. Você notou algum problema nos dados, como valores ausentes, duplicados ou tipos de dados errados`

'Notei que nas primeiras 10 linhas obtivemos um valor ausente "NaN" no índice[9] na coluna Artist.'

Voltar ao Índice

2.3 Etapa 2. Pré-processamento de dados

O objetivo aqui é preparar os dados para a análise. O primeiro passo é resolver todos os problemas com o cabeçalho. E então podemos passar para os valores ausentes e duplicados. Vamos começar.

Corrija a formatação nos cabeçalhos da tabela.

2.3.1 Estilo do cabeçalho

Imprima os cabeçalhos da tabela (os nomes das colunas):

[18]:
```
# imprima os nomes das colunas
print(df.columns)
```

```
Index(['  userID', 'Track', 'artist', 'genre', '  City  ', 'time', 'Day'],
dtype='object')
```

Mude os cabeçalhos da tabela conforme as boas práticas de estilo: * Todos os caracteres precisam estar com letras minúsculas * Exclua espaços * Se o nome tiver várias palavras, use snake_case

Anteriormente, você aprendeu sobre uma maneira automatizada de renomear colunas. Vamos usá-la agora. Use o ciclo for para percorrer os nomes das colunas e transformar todos os caracteres em letras minúsculas. Após fazer isso, imprima os cabeçalhos da tabela novamente:

[19]:
```
# Percorrendo os cabeçalhos e convertendo tudo em minúsculos
new_col_names = []
for old_names in df.columns:
    name_lowered = old_names.lower()
```

Agora, usando a mesma abordagem, exclua os espaços no início e no final de cada nome de coluna e imprima os nomes das colunas novamente:

[20]:
```
# Percorrendo os cabeçalhos e removendo os espaços
name_stripped = name_lowered.strip()
```

Comentário do revisor: Aqui há um erro de lógica. veja que name_lowered é na verdade o valor da última execução do loop. Como você fez correto na 2a célula abaixo, vamos seguir

Precisamos aplicar a regra de sublinhado no lugar de espaço à coluna `userid`. Deveria ser `user_id`. Renomeie essa coluna e imprima os nomes de todas as colunas quando terminar.

[21]:
```
# Renomeando a coluna "userid"
name_snake = name_stripped.replace('userID','user_id')
```

Verifique o resultado. Imprima os cabeçalhos novamente:

[22]:
```
# verificando o resultado: a lista de cabeçalhos
new_col_names = []
for old_name in df.columns:
    name_lowered = old_name.lower()
    name_stripped = name_lowered.strip()
    name_snake = name_stripped.replace('userID','user_id')
    new_col_names.append(name_snake)

df.columns = new_col_names
```

```
print(df.columns)
```

```
Index(['userid', 'track', 'artist', 'genre', 'city', 'time', 'day'],
dtype='object')
```

Voltar ao Índice

2.3.2 Valores Ausentes

Primeiro, encontre a quantidade de valores ausentes na tabela. Você precisa usar dois métodos em sequência para obter o número de valores ausentes.

[23]:
```
# calculando o número de valores ausentes
print(df.isna().sum())
```

```
userid       0
track     1343
artist    7567
genre     1198
city         0
time         0
day          0
dtype: int64
```

Nem todos os valores ausentes afetam a pesquisa. Por exemplo, os valores ausentes em `track` e `artist` não são críticos. Você pode simplesmente substituí-los por valores padrão, como a string `'unknown'`.

Mas valores ausentes em `'genre'` podem afetar a comparação de preferências musicais de Springfield e Shelbyville. Na vida real, seria útil descobrir as razões pelas quais os dados estão ausentes e tentar corrigi-los. Mas nós não temos essa possibilidade neste projeto. Então, você terá que: * Preencha esses valores ausentes com um valor padrão * Avalie em que medida os valores ausentes podem afetar sua análise

Substitua os valores ausentes nas colunas `'track'`, `'artist'` e `'genre'` pela string `'unknown'`. Como mostramos nas lições anteriores, a melhor maneira de fazer isso é criar uma lista para armazenar os nomes das colunas nas quais precisamos fazer a substituição. Em seguida, use essa lista e percorra as colunas nas quais a substituição seja necessária e faça a substituição.

[24]:
```
# percorrendo os cabeçalhos e substituindo valores ausentes por 'unknown'
cols_to_replace = ['track','artist', 'genre']
for col in cols_to_replace:
    df[col].fillna('unknown',inplace = True)
```

Agora verifique o resultado para ter certeza de que o conjunto de dados não contenha valores ausentes após a substituição. Para fazer isso, conte os valores ausentes novamente.

[25]:
```
# contando os valores ausentes
print(df.isna().sum())
```

```
userid    0
track     0
artist    0
genre     0
city      0
time      0
day       0
dtype: int64
```

Voltar ao Índice

2.3.3 Duplicados

Encontre o número de duplicados explícitos na tabela. Lembre-se de que você precisa aplicar dois métodos em sequência para obter o número de duplicados explícitos.

[26]:
```
# contando duplicados explícitos
print(df.duplicated().sum())
```

```
3826
```

Agora descarte todos os duplicados. Para fazer isso, chame o método que faz exatamente isso.

[27]:
```
# removendo duplicados explícitos
df=df.drop_duplicates()
```

Agora vamos verificar se descartamos todos os duplicados. Conte duplicados explícitos mais uma vez para ter certeza de que você removeu todos eles:

[28]:
```
# verificando duplicados novamente
print(df.duplicated().sum())
```

```
0
```

Agora queremos nos livrar dos duplicados implícitos na coluna `genre`. Por exemplo, o nome de um gênero pode ser escrito de maneiras diferentes. Alguns erros afetarão também o resultado.

Para fazer isso, vamos começar imprimindo uma lista de nomes de gênero únicos, ordenados em ordem alfabética: Para fazer isso: * Extraia a coluna `genre` do DataFrame * Chame o método que retornará todos os valores únicos na coluna extraída

[29]:
```
# visualizando nomes de gêneros únicos
print(df['genre'].unique())
```

```
['rock' 'pop' 'folk' 'dance' 'rusrap' 'ruspop' 'world' 'electronic'
 'unknown' 'alternative' 'children' 'rnb' 'hip' 'jazz' 'postrock' 'latin'
 'classical' 'metal' 'reggae' 'triphop' 'blues' 'instrumental' 'rusrock'
 'dnb' 'türk' 'post' 'country' 'psychedelic' 'conjazz' 'indie'
 'posthardcore' 'local' 'avantgarde' 'punk' 'videogame' 'techno' 'house'
 'christmas' 'melodic' 'caucasian' 'reggaeton' 'soundtrack' 'singer' 'ska'
 'salsa' 'ambient' 'film' 'western' 'rap' 'beats' "hard'n'heavy"
```

```
'progmetal' 'minimal' 'tropical' 'contemporary' 'new' 'soul' 'holiday'
'german' 'jpop' 'spiritual' 'urban' 'gospel' 'nujazz' 'folkmetal'
'trance' 'miscellaneous' 'anime' 'hardcore' 'progressive' 'korean'
'numetal' 'vocal' 'estrada' 'tango' 'loungeelectronic' 'classicmetal'
'dubstep' 'club' 'deep' 'southern' 'black' 'folkrock' 'fitness' 'french'
'disco' 'religious' 'hiphop' 'drum' 'extrememetal' 'türkçe'
'experimental' 'easy' 'metalcore' 'modern' 'argentinetango' 'old' 'swing'
'breaks' 'eurofolk' 'stonerrock' 'industrial' 'funk' 'middle' 'variété'
'other' 'adult' 'christian' 'thrash' 'gothic' 'international' 'muslim'
'relax' 'schlager' 'caribbean' 'nu' 'breakbeat' 'comedy' 'chill' 'newage'
'specialty' 'uzbek' 'k-pop' 'balkan' 'chinese' 'meditative' 'dub' 'power'
'death' 'grime' 'arabesk' 'romance' 'flamenco' 'leftfield' 'european'
'tech' 'newwave' 'dancehall' 'mpb' 'piano' 'top' 'bigroom' 'opera'
'celtic' 'tradjazz' 'acoustic' 'epicmetal' 'hip-hop' 'historisch'
'downbeat' 'downtempo' 'africa' 'audiobook' 'jewish' 'sängerportrait'
'deutschrock' 'eastern' 'action' 'future' 'electropop' 'folklore'
'bollywood' 'marschmusik' 'rnr' 'karaoke' 'indian' 'rancheras'
'afrikaans' 'rhythm' 'sound' 'deutschspr' 'trip' 'lovers' 'choral'
'dancepop' 'retro' 'smooth' 'mexican' 'brazilian' 'ïîï' 'mood' 'surf'
'gangsta' 'inspirational' 'idm' 'ethnic' 'bluegrass' 'broadway'
'animated' 'americana' 'karadeniz' 'rockabilly' 'colombian' 'self' 'hop'
'sertanejo' 'japanese' 'canzone' 'lounge' 'sport' 'ragga' 'traditional'
'gitarre' 'frankreich' 'emo' 'laiko' 'cantopop' 'glitch' 'documentary'
'oceania' 'popeurodance' 'dark' 'vi' 'grunge' 'hardstyle' 'samba'
'garage' 'art' 'folktronica' 'entehno' 'mediterranean' 'chamber' 'cuban'
'taraftar' 'gypsy' 'hardtechno' 'shoegazing' 'bossa' 'latino' 'worldbeat'
'malaysian' 'baile' 'ghazal' 'arabic' 'popelectronic' 'acid' 'kayokyoku'
'neoklassik' 'tribal' 'tanzorchester' 'native' 'independent' 'cantautori'
'handsup' 'punjabi' 'synthpop' 'rave' 'französisch' 'quebecois' 'speech'
'soulful' 'jam' 'ram' 'horror' 'orchestral' 'neue' 'roots' 'slow'
'jungle' 'indipop' 'axé' 'fado' 'showtunes' 'arena' 'irish' 'mandopop'
'forró' 'dirty' 'regional']
```

Olhe a lista e encontre duplicados implícitos do gênero `hiphop`. Esses podem ser nomes escritos incorretamente, ou nomes alternativos para o mesmo gênero.

Você verá os seguintes duplicados implícitos: * `hip` * `hop` * `hip-hop`

Para se livrar deles, crie uma função `replace_wrong_genres()` com dois parâmetros: * `wrong_genres=` — essa é uma lista que contém todos os valores que você precisa substituir * `correct_genre=` — essa é uma string que você vai usar para a substituição

Como resultado, a função deve corrigir os nomes na coluna `'genre'` da tabela `df`, isto é, substituindo cada valor da lista `wrong_genres` por valores de `correct_genre`.

Dentro do corpo da função, use um ciclo `'for'` para percorrer a lista de gêneros errados, extrair a coluna `'genre'` e aplicar o método `replace` para fazer as correções.

[30]:
```
# função para substituir duplicados implícitos
import pandas as pd
```

```
def replace_wrong_genres(df,genre,wrong_genres, correct_genre):
    for wrong_genre in wrong_genres:
        df[genre] = df[genre].replace(wrong_genre, correct_genre)
    return df
```

Agora, chame a função `replace_wrong_genres()` e passe argumentos apropriados para que ela limpe duplicados implícitos (`hip`, `hop` e `hip-hop`) substituindo-os por `hiphop`:

[31]:
```
# removendo duplicados implícitos
def replace_wrong_genres(df,genre,wrong_genres, correct_genre):
    for wrong_genre in wrong_genres:
        df[genre] = df[genre].replace(wrong_genre, correct_genre)
    return df

duplicates = ['hip', 'hop', 'hip-hop']
name = 'hiphop'
df = replace_wrong_genres(df,'genre',duplicates,name)
print (df)
```

```
        userid                              track            artist  \
0      FFB692EC                 Kamigata To Boots  The Mass Missile
1      55204538       Delayed Because of Accident  Andreas Rönnberg
2        20EC38                  Funiculì funiculà       Mario Lanza
3      A3DD03C9             Dragons in the Sunset        Fire + Ice
4      E2DC1FAE                       Soul People        Space Echo
…         …                                  …                …
65074  729CBB09                           My Name            McLean
65075  D08D4A55  Maybe One Day (feat. Black Spade)       Blu & Exile
65076  C5E3A0D5                         Jalopiina           unknown
65077  321D0506                      Freight Train     Chas McDevitt
65078  3A64EF84        Tell Me Sweet Little Lies      Monica Lopez

            genre         city      time        day
0            rock  Shelbyville  20:28:33  Wednesday
1            rock  Springfield  14:07:09     Friday
2             pop  Shelbyville  20:58:07  Wednesday
3            folk  Shelbyville  08:37:09     Monday
4          dance  Springfield  08:34:34     Monday
…           …          …       …         …
65074        rnb  Springfield  13:32:28  Wednesday
65075     hiphop  Shelbyville  10:00:00     Monday
65076  industrial  Springfield  20:09:26     Friday
65077       rock  Springfield  21:43:59     Friday
65078    country  Springfield  21:59:46     Friday

[61253 rows x 7 columns]
```

Certifique-se que os nomes duplicados foram removidos. Imprima a lista de valores únicos da coluna

'genre' mais uma vez:

[33]:
```
# verificando valores duplicados
print(df['genre'].unique())
```

```
['rock' 'pop' 'folk' 'dance' 'rusrap' 'ruspop' 'world' 'electronic'
 'unknown' 'alternative' 'children' 'rnb' 'hiphop' 'jazz' 'postrock'
 'latin' 'classical' 'metal' 'reggae' 'triphop' 'blues' 'instrumental'
 'rusrock' 'dnb' 'türk' 'post' 'country' 'psychedelic' 'conjazz' 'indie'
 'posthardcore' 'local' 'avantgarde' 'punk' 'videogame' 'techno' 'house'
 'christmas' 'melodic' 'caucasian' 'reggaeton' 'soundtrack' 'singer' 'ska'
 'salsa' 'ambient' 'film' 'western' 'rap' 'beats' "hard'n'heavy"
 'progmetal' 'minimal' 'tropical' 'contemporary' 'new' 'soul' 'holiday'
 'german' 'jpop' 'spiritual' 'urban' 'gospel' 'nujazz' 'folkmetal'
 'trance' 'miscellaneous' 'anime' 'hardcore' 'progressive' 'korean'
 'numetal' 'vocal' 'estrada' 'tango' 'loungeelectronic' 'classicmetal'
 'dubstep' 'club' 'deep' 'southern' 'black' 'folkrock' 'fitness' 'french'
 'disco' 'religious' 'drum' 'extrememetal' 'türkçe' 'experimental' 'easy'
 'metalcore' 'modern' 'argentinetango' 'old' 'swing' 'breaks' 'eurofolk'
 'stonerrock' 'industrial' 'funk' 'middle' 'variété' 'other' 'adult'
 'christian' 'thrash' 'gothic' 'international' 'muslim' 'relax' 'schlager'
 'caribbean' 'nu' 'breakbeat' 'comedy' 'chill' 'newage' 'specialty'
 'uzbek' 'k-pop' 'balkan' 'chinese' 'meditative' 'dub' 'power' 'death'
 'grime' 'arabesk' 'romance' 'flamenco' 'leftfield' 'european' 'tech'
 'newwave' 'dancehall' 'mpb' 'piano' 'top' 'bigroom' 'opera' 'celtic'
 'tradjazz' 'acoustic' 'epicmetal' 'historisch' 'downbeat' 'downtempo'
 'africa' 'audiobook' 'jewish' 'sängerportrait' 'deutschrock' 'eastern'
 'action' 'future' 'electropop' 'folklore' 'bollywood' 'marschmusik' 'rnr'
 'karaoke' 'indian' 'rancheras' 'afrikaans' 'rhythm' 'sound' 'deutschspr'
 'trip' 'lovers' 'choral' 'dancepop' 'retro' 'smooth' 'mexican'
 'brazilian' 'ïîï' 'mood' 'surf' 'gangsta' 'inspirational' 'idm' 'ethnic'
 'bluegrass' 'broadway' 'animated' 'americana' 'karadeniz' 'rockabilly'
 'colombian' 'self' 'sertanejo' 'japanese' 'canzone' 'lounge' 'sport'
 'ragga' 'traditional' 'gitarre' 'frankreich' 'emo' 'laiko' 'cantopop'
 'glitch' 'documentary' 'oceania' 'popeurodance' 'dark' 'vi' 'grunge'
 'hardstyle' 'samba' 'garage' 'art' 'folktronica' 'entehno'
 'mediterranean' 'chamber' 'cuban' 'taraftar' 'gypsy' 'hardtechno'
 'shoegazing' 'bossa' 'latino' 'worldbeat' 'malaysian' 'baile' 'ghazal'
 'arabic' 'popelectronic' 'acid' 'kayokyoku' 'neoklassik' 'tribal'
 'tanzorchester' 'native' 'independent' 'cantautori' 'handsup' 'punjabi'
 'synthpop' 'rave' 'französisch' 'quebecois' 'speech' 'soulful' 'jam'
 'ram' 'horror' 'orchestral' 'neue' 'roots' 'slow' 'jungle' 'indipop'
 'axé' 'fado' 'showtunes' 'arena' 'irish' 'mandopop' 'forró' 'dirty'
 'regional']
```

Voltar ao Índice

2.3.4 Suas observações

`Descreva brevemente o que você reparou ao analisar duplicados, bem como a abordagem que usou para eliminá-los e os resultados que alcançou.`

'Reparei que primeiro devemos verificar quais duplicados existem, após isto, verificamos que havia um mesmo gênero musical escrito de três formas diferentes. Com isso criamos uma função com quatro argumentos,(dataframe, coluna onde queremos substituir, valores errados e valor correto que queremos acrescentar), criamos um loop 'for' e dentro dele usamos o método .replace para que fosse substituído os valores incorretos pelos corretos.

Voltar ao Índice

2.4 Etapa 3. Teste da hipótese

2.4.1 Hipótese: comparação do comportamento dos usuários nas duas cidades

A hipótese afirma que existem diferenças no consumo de música pelos usuários em Springfield e em Shelbyville. Para testar a hipótese, use os dados dos três dias da semana: segunda-feira (Monday), quarta-feira (Wednesday) e sexta-feira (Friday).

- Agrupe os usuários por cidade.
- Compare o número de músicas tocadas por cada grupo na segunda, quarta e sexta.

Execute cada cálculo separadamente.

O primeiro passo é avaliar a atividade dos usuários em cada cidade. Não se esqueça das etapas "divisão-aplicação-combinação" sobre as quais falamos anteriormente na lição. Agora seu objetivo é agrupar os dados por cidade, aplicar o método de contagem apropriado durante a etapa de aplicação e então encontrar o número de músicas tocadas por cada grupo, especificando a coluna para a qual você quer obter a contagem.

Veja um exemplo de como o resultado final deve ser: `df.groupby(by='....')['column'].method()`
Execute cada cálculo separadamente.

Para avaliar a atividade dos usuários em cada cidade, agrupe os dados por cidade e encontre o número de músicas reproduzidas em cada grupo.

[34]:
```
# Contando as músicas tocadas em cada cidade
city_number = df.groupby('city')['genre'].count()
print(city_number)
```

```
city
Shelbyville    18512
Springfield    42741
Name: genre, dtype: int64
```

Agora vamos agrupar os dados por dia da semana e encontrar a quantidade de músicas tocadas na segunda, quarta e sexta-feira. Use a mesma abordagem que antes, mas agora precisamos agrupar os dados de uma forma diferente.

[35]:
```
# Calculando as músicas escutadas em cada um desses três dias
day_count = df.groupby('day')['genre'].count()
```

```
print(day_count)
```

```
day
Friday       21840
Monday       21354
Wednesday    18059
Name: genre, dtype: int64
```

`Comente sobre suas observações aqui`

'Percebi que na Cidade de Springfield as pessoas escutam muito mais músicas que na cidade de Shelbyville e que Segunda e Sexta-feira o aplicativo streaming é muito mais usado.

Você acabou de aprender como contar entradas agrupando-as por cidade ou por dia. E agora você precisa escrever uma função que possa contar entradas simultaneamente com base em ambos os critérios.

Crie a função `number_tracks()` para calcular o número de músicas tocadas em um determinado dia **e** em uma determinada cidade. A função deve aceitar dois parâmetros:

- `day`: um dia da semana pelo qual precisamos filtrar os dados. Por exemplo, `'Monday'`.
- `city`: uma cidade pela qual precisamos filtrar os dados. Por exemplo, `'Springfield'`.

Dentro da função, você vai aplicar uma filtragem consecutiva com indexação lógica.

Primeiro, filtre os dados por dia e então filtre a tabela resultante por cidade.

Depois de filtrar os dados usando os dois critérios, conte o número de valores na coluna 'user_id' da tabela resultante. O resultado da contagem representará o número de entradas que você quer encontrar. Armazene o resultado em uma nova variável e imprima-o.

[36]:
```
# Declare a função number_tracks() com dois parâmetros: day= e city=.
def number_tracks(day,city):
    filtered_by_day = df[df['day'] == day]
    filtered_by_city = filtered_by_day[filtered_by_day['city'] == city]
    track_count = filtered_by_city['userid'].count()
    return(track_count)

    # Armazene as linhas do DataFrame em que o valor na coluna 'day' é igual ao␣
 ↪parâmetro day=

    # Filtre as linhas em que o valor na coluna 'city' é igual ao parâmetro␣
 ↪city=

    # Extraia a coluna 'user_id' da tabela filtrada e aplique o método count()

    # Retorne o número dos valores da coluna 'user_id'
```

Chame a função `number_tracks()` seis vezes, mudando os valores dos parâmetros, para que você possa recuperar os dados de ambas as cidades para cada um dos três dias.

[37]:
```
# a quantidade de músicas tocadas em Springfield na segunda-feira
print(number_tracks('Monday','Springfield'))
```

```
15740
```

[38]:
```
# a quantidade de músicas tocadas em Shelbyville na segunda-feira
print(number_tracks('Monday','Shelbyville'))
```

```
5614
```

[39]:
```
# a quantidade de músicas tocadas em Springfield na quarta-feira
print(number_tracks('Wednesday','Springfield'))
```

```
11056
```

[40]:
```
# a quantidade de músicas tocadas em Shelbyville na quarta-feira
print(number_tracks('Wednesday','Shelbyville'))
```

```
7003
```

[41]:
```
# a quantidade de músicas tocadas em Springfield na sexta-feira
print(number_tracks('Friday','Springfield'))
```

```
15945
```

[42]:
```
# a quantidade de músicas tocadas em Shelbyville na sexta-feira
print(number_tracks('Friday','Shelbyville'))
```

```
5895
```

Conclusões

`Comente sobre se a terceira hipótese está correta ou deve ser rejeitada. Explique seu raciocínio.`

'A terceira hipótese deve ser aceita, porque na quarta-feira em Springfield a atividade dos usuários foram de 11056 vezes e em Shelbyville foram de somente 7003 vezes, um pouco mais de 2/3 da cidade de Springfield. E se formos comparar também diferentes dias da semana na mesma cidade, veremos a diferença de uso do aplicativo de streaming, como por exemplo: na quarta-feira em Springfield, foram usados 11056 o aplicativo, já na sexta-feira também em Springfield, foram usados 15945 vezes.

Voltar ao Índice

3 Conclusões

`Resuma suas conclusões sobre a hipótese aqui`

'A atividade dos usuários realmente são diferentes quando comparamos o uso do aplicativo nos dias da semana e nas cidades que foram tocadas, vimos que os usuários de Springfield usam o aplicativo

mais vezes na Sexta e Segunda-feira, quase igualando as vezes que foram usadas, já em Shelbyville, na Quarta-feira, os usuários usam mais vezes o aplicativo do que no resto dos dias da semana.

#Porém caso fosse necessário fazer uma análise mais detalhada, precisaríamos de mais dados, ex: quantos habitantes existem em cada cidade, assim conseguiriamos fazer uma comparação do uso do aplicativo de cada cidade em porcentagem, proporcionalmente ao número de habitantes. Outros dias da semana que os usuários usaram o aplicativo também seria útil.

3.0.1 Importante

Em projetos de pesquisas reais, o teste estatístico de hipóteses é mais preciso e quantitativo. Observe também que conclusões sobre uma cidade inteira nem sempre podem ser tiradas a partir de dados de apenas uma fonte.

Você aprenderá mais sobre testes de hipóteses no sprint sobre a análise estatística de dados.

Voltar ao Índice