music_project

September 25, 2022

1 Y.Music

2 Konten

2.1 Pendahuluan

Setiap kali kita melakukan penelitian, kita perlu merumuskan hipotesis yang kemudian dapat kita uji. Terkadang kita menerima hipotesis ini; tetapi terkadang kita juga menolaknya. Untuk membuat keputusan yang tepat, sebuah bisnis harus dapat memahami apakah asumsi yang dibuatnya benar atau tidak.

Dalam proyek kali ini, Anda akan membandingkan preferensi musik kota Springfield dan Shelbyville. Anda akan mempelajari data Y.Music yang sebenarnya untuk menguji hipotesis di bawah ini dan membandingkan perilaku pengguna di kedua kota ini.

2.1.1 Tujuan:

Menguji tiga hipotesis: 1. Aktivitas pengguna berbeda-beda tergantung pada hari dan kotanya. 2. Pada senin pagi, penduduk Springfield dan Shelbyville mendengarkan genre yang berbeda. Hal ini juga ini juga berlaku untuk Jumat malam. 3. Pendengar di Springfield dan Shelbyville memiliki preferensi yang berbeda. Di Springfield, mereka lebih suka musik pop, sementara Shelbyville, musik rap memiliki lebih banyak penggemar.

2.1.2 Tahapan

Data tentang perilaku pengguna disimpan dalam berkas `/datasets/music_project_en.csv`. Tidak ada informasi tentang kualitas data, jadi Anda perlu memeriksanya lebih dahulu sebelum menguji hipotesis.

Pertama, Anda akan mengevaluasi kualitas data dan melihat apakah masalahnya signifikan. Kemudian, selama pra-pemrosesan data, Anda akan mencoba memperhitungkan masalah yang paling serius.

Proyek ini akan terdiri dari tiga tahap: 1. Ikhtisar data 2. Pra-pemrosesan data 3. Menguji hipotesis

Kembali ke Daftar Isi

2.2 Tahap 1. Ikhtisar data

Buka data di Y.Music lalu jelajahi data yang ada di sana.

Anda akan membutuhkan **pandas**, jadi Anda harus mengimpornya.

[7]:
```
# mengimpor pandas
import pandas as pd
```

Baca file `music_project_en.csv` dari folder `/datasets/` lalu simpan di variabel `df`:

[8]:
```
# membaca berkas dan menyimpannya ke df
df = pd.read_csv('/home/syaid/Downloads/music_project_en.csv')
```

Tampilkan 10 baris tabel pertama:

[9]:
```
# memperoleh 10 baris pertama dari tabel df
df.head(10)
```

[9]:

	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
```

Memperoleh informasi umum tentang tabel dengan satu perintah:

[10]:
```
# memperoleh informasi umum tentang data di df
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
```

Tabel ini berisi tujuh kolom. Semuanya menyimpan tipe data yang sama, yaitu: `objek`.

Berdasarkan dokumentasi: - `'userID'` — pengenal pengguna - `'Track'` — judul trek - `'artist'` — nama artis - `'genre'` - `'City'` — kota tempat pengguna berada - `'time'` — lama waktu lagu tersebut dimainkan - `'Day'` — nama hari

Kita dapat melihat tiga masalah dengan gaya penulisan nama kolom: 1. Beberapa nama huruf besar, beberapa huruf kecil. 2. Ada penggunaan spasi pada beberapa nama. 3. Kolom yang terdiri dari beberapa kata tidak dipisah.

Jumlah nilai kolom berbeda. Ini berarti data mengandung nilai yang hilang.

2.2.1 Kesimpulan

Setiap baris dalam tabel menyimpan data pada lagu yang dimainkan. Beberapa kolom menggambarkan lagu itu sendiri: judul, artis, dan genre. Sisanya menyampaikan informasi tentang pengguna: kota asal mereka, waktu mereka memutar lagu.

Jelas bahwa data tersebut cukup untuk menguji hipotesis. Namun, ada nilai-nilai yang hilang.

Selanjutnya, kita perlu melakukan pra-pemrosesan data terlebih dahulu.

Kembali ke Daftar Isi

2.3 Tahap 2. Pra-pemrosesan data

Perbaiki format pada judul kolom dan atasi nilai yang hilang. Kemudian, periksa apakah ada duplikat dalam data.

2.3.1 Gaya penulisan judul

Tampilkan judul kolom:

[11]:
```
# daftar nama kolom di tabel df
df.columns
```

[11]:
```
Index(['  userID', 'Track', 'artist', 'genre', '  City  ', 'time', 'Day'],
dtype='object')
```

Ubah nama kolom sesuai dengan aturan gaya penulisan yang baik: * Jika nama memiliki beberapa kata, gunakan snake_case * Semua karakter harus menggunakan huruf kecil * Hapus spasi

[12]:
```
# mengganti nama kolom

df.rename(columns={'  userID': 'user_id', 'Track': 'track', '  City  ': 'city',␣
 ↪'Day': 'day'}, inplace=True)
```

Periksa hasilnya. Tampilkan nama kolom sekali lagi:

[13]:
```
# hasil pengecekan: daftar nama kolom
df.columns
```

[13]:
```
Index(['user_id', 'track', 'artist', 'genre', 'city', 'time', 'day'],
dtype='object')
```

Kembali ke Daftar Isi

2.3.2 Nilai-nilai yang hilang

Pertama, temukan jumlah nilai yang hilang dalam tabel. Untuk melakukannya, gunakan dua metode `pandas`:

[14]:
```
# menghitung nilai yang hilang
df.isna().sum()
```

[14]:
```
user_id       0
track      1343
artist     7567
genre      1198
city          0
time          0
day           0
dtype: int64
```

Tidak semua nilai yang hilang berpengaruh terhadap penelitian. Misalnya, nilai yang hilang dalam `track` dan `artist` tidak begitu penting. Anda cukup menggantinya dengan tanda yang jelas.

Namun nilai yang hilang dalam `'genre'` dapat memengaruhi perbandingan preferensi musik di Springfield dan Shelbyville. Dalam kehidupan nyata, ini akan berguna untuk mempelajari alasan mengapa data tersebut hilang dan mencoba memperbaikinya. Tetapi kita tidak memiliki kesempatan itu dalam proyek ini. Jadi Anda harus: * Isi nilai yang hilang ini dengan sebuah tanda * Evaluasi seberapa besar nilai yang hilang dapat memengaruhi perhitungan Anda

Ganti nilai yang hilang pada `'track'`, `'artist'`, dan `'genre'` dengan string `'unknown'`. Untuk melakukannya, buat list `columns_to_replace`, ulangi dengan `for`, dan ganti nilai yang hilang di setiap kolom:

[15]:
```
# mengulang nama kolom dan mengganti nilai yang hilang dengan 'unknown'
columns_to_replace = ['track', 'artist', 'genre']
for column in columns_to_replace:
    df[column] = df[column].fillna('unknown')
```

Pastikan tidak ada tabel lagi yang berisi nilai yang hilang. Hitung kembali nilai yang hilang.

[16]:
```
# menghitung nilai yang hilang
df.isna().sum()
```

[16]:
```
user_id    0
track      0
artist     0
genre      0
city       0
time       0
day        0
dtype: int64
```

Kembali ke Daftar Isi

2.3.3 Duplikat

Temukan jumlah duplikat yang jelas dalam tabel menggunakan satu perintah:

[17]:
```
# menghitung duplikat yang jelas
df.duplicated().sum()
```

[17]:
```
3826
```

Panggil metode `pandas` untuk menghapus duplikat yang jelas:

[18]:
```
# menghapus duplikat yang jelas
df = df.drop_duplicates().reset_index(drop=True)
```

Hitung duplikat yang jelas sekali lagi untuk memastikan Anda telah menghapus semuanya:

[19]:
```
# memeriksa duplikat
print(df.duplicated().sum())
```

0

Sekarang hapus duplikat implisit di kolom `genre`. Misalnya, nama genre dapat ditulis dengan cara yang berbeda. Kesalahan seperti ini juga akan memengaruhi hasil.

Tampilkan daftar nama genre yang unik, urutkan berdasarkan abjad. Untuk melakukannya: * Ambil kolom DataFrame yang dimaksud * Terapkan metode pengurutan untuk itu * Untuk kolom yang diurutkan, panggil metode yang akan menghasilkan semua nilai kolom yang unik

[20]:
```
# melihat nama genre yang unik
df['genre'].sort_values().unique()
```

[20]:
```
array(['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans',
       'alternative', 'ambient', 'americana', 'animated', 'anime',
       'arabesk', 'arabic', 'arena', 'argentinetango', 'art', 'audiobook',
       'avantgarde', 'axé', 'baile', 'balkan', 'beats', 'bigroom',
       'black', 'bluegrass', 'blues', 'bollywood', 'bossa', 'brazilian',
       'breakbeat', 'breaks', 'broadway', 'cantautori', 'cantopop',
       'canzone', 'caribbean', 'caucasian', 'celtic', 'chamber',
       'children', 'chill', 'chinese', 'choral', 'christian', 'christmas',
       'classical', 'classicmetal', 'club', 'colombian', 'comedy',
       'conjazz', 'contemporary', 'country', 'cuban', 'dance',
       'dancehall', 'dancepop', 'dark', 'death', 'deep', 'deutschrock',
       'deutschspr', 'dirty', 'disco', 'dnb', 'documentary', 'downbeat',
       'downtempo', 'drum', 'dub', 'dubstep', 'eastern', 'easy',
       'electronic', 'electropop', 'emo', 'entehno', 'epicmetal',
       'estrada', 'ethnic', 'eurofolk', 'european', 'experimental',
       'extrememetal', 'fado', 'film', 'fitness', 'flamenco', 'folk',
       'folklore', 'folkmetal', 'folkrock', 'folktronica', 'forró',
       'frankreich', 'französisch', 'french', 'funk', 'future', 'gangsta',
       'garage', 'german', 'ghazal', 'gitarre', 'glitch', 'gospel',
       'gothic', 'grime', 'grunge', 'gypsy', 'handsup', "hard'n'heavy",
       'hardcore', 'hardstyle', 'hardtechno', 'hip', 'hip-hop', 'hiphop',
       'historisch', 'holiday', 'hop', 'horror', 'house', 'idm',
       'independent', 'indian', 'indie', 'indipop', 'industrial',
       'inspirational', 'instrumental', 'international', 'irish', 'jam',
       'japanese', 'jazz', 'jewish', 'jpop', 'jungle', 'k-pop',
       'karadeniz', 'karaoke', 'kayokyoku', 'korean', 'laiko', 'latin',
       'latino', 'leftfield', 'local', 'lounge', 'loungeelectronic',
       'lovers', 'malaysian', 'mandopop', 'marschmusik', 'meditative',
       'mediterranean', 'melodic', 'metal', 'metalcore', 'mexican',
       'middle', 'minimal', 'miscellaneous', 'modern', 'mood', 'mpb',
       'muslim', 'native', 'neoklassik', 'neue', 'new', 'newage',
       'newwave', 'nu', 'nujazz', 'numetal', 'oceania', 'old', 'opera',
       'orchestral', 'other', 'piano', 'pop', 'popelectronic',
```

```
 'popeurodance', 'post', 'posthardcore', 'postrock', 'power',
 'progmetal', 'progressive', 'psychedelic', 'punjabi', 'punk',
 'quebecois', 'ragga', 'ram', 'rancheras', 'rap', 'rave', 'reggae',
 'reggaeton', 'regional', 'relax', 'religious', 'retro', 'rhythm',
 'rnb', 'rnr', 'rock', 'rockabilly', 'romance', 'roots', 'ruspop',
 'rusrap', 'rusrock', 'salsa', 'samba', 'schlager', 'self',
 'sertanejo', 'shoegazing', 'showtunes', 'singer', 'ska', 'slow',
 'smooth', 'soul', 'soulful', 'sound', 'soundtrack', 'southern',
 'specialty', 'speech', 'spiritual', 'sport', 'stonerrock', 'surf',
 'swing', 'synthpop', 'sängerportrait', 'tango', 'tanzorchester',
 'taraftar', 'tech', 'techno', 'thrash', 'top', 'traditional',
 'tradjazz', 'trance', 'tribal', 'trip', 'triphop', 'tropical',
 'türk', 'türkçe', 'unknown', 'urban', 'uzbek', 'variété', 'vi',
 'videogame', 'vocal', 'western', 'world', 'worldbeat', 'ïîï'],
dtype=object)
```

Lihat melalui list untuk menemukan duplikat implisit dari genre `hiphop`. Ini bisa berupa nama yang ditulis secara salah atau nama alternatif dari genre yang sama.

Anda akan melihat duplikat implisit berikut: * `hip` * `hop` * `hip-hop`

Untuk menghapusnya, gunakan fungsi `replace_wrong_genres()` dengan dua parameter: * `wrong_genres=` — daftar duplikat * `correct_genre=` — string dengan nilai yang benar

Fungsi harus mengoreksi nama dalam kolom `'genre'` dari tabel `df`, yaitu mengganti setiap nilai dari daftar `wrong_genres` dengan nilai dalam `correct_genre`.

[21]:
```
# fungsi untuk mengganti duplikat implisit

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

Panggil `replace_wrong_genres()` dan berikan argumennya sehingga menghapus duplikat implisit (`hip`, `hop`, dan `hip-hop`) dan menggantinya dengan `hiphop`:

[22]:
```
# menghapus duplikat implisit

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

Pastikan nama duplikat telah dihapus. Tampilkan daftar nilai unik dari kolom `'genre'`:

[23]:
```
# memeriksa duplikat implisit
df['genre'].sort_values().unique()
```

[23]:
```
array(['acid', 'acoustic', 'action', 'adult', 'africa', 'afrikaans',
       'alternative', 'ambient', 'americana', 'animated', 'anime',
```

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

Kembali ke Daftar Isi

2.3.4 Kesimpulan

Kita mendeteksi tiga masalah dengan data:

- Gaya penulisan judul yang salah
- Nilai-nilai yang hilang
- Duplikat yang jelas dan implisit

Judul telah dibersihkan untuk mempermudah pemrosesan tabel.

Semua nilai yang hilang telah diganti dengan `'unknown'`. Tapi kita masih harus melihat apakah nilai yang hilang dalam `'genre'` akan memengaruhi perhitungan kita.

Tidak adanya duplikat akan membuat hasil lebih tepat dan lebih mudah dipahami.

Sekarang kita dapat melanjutkan ke pengujian hipotesis.

Kembali ke Daftar Isi

2.4 Tahap 3. Menguji hipotesis

2.4.1 Hipotesis 1: membandingkan perilaku pengguna di dua kota

Menurut hipotesis pertama, pengguna dari Springfield dan Shelbyville memiliki perbedaan dalam mendengarkan musik. Pengujian ini menggunakan data pada hari: Senin, Rabu, dan Jumat.

- pisahkan pengguna ke dalam kelompok berdasarkan kota.
- Bandingkan berapa banyak lagu yang dimainkan setiap kelompok pada hari Senin, Rabu, dan Jumat.

Untuk latihan, lakukan setiap perhitungan secara terpisah.

Evaluasi aktivitas pengguna di setiap kota. Kelompokkan data berdasarkan kota dan temukan jumlah lagu yang diputar di setiap kelompok.

[24]:
```
# Menghitung lagu yang diputar di setiap kota
print(df.groupby('city')['city'].count())
```

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

Springfield memiliki lebih banyak lagu yang dimainkan daripada Shelbyville. Namun bukan berarti warga Springfield lebih sering mendengarkan musik. Kota ini lebih besar, dan memiliki lebih banyak pengguna.

Sekarang kelompokkan data menurut hari dan temukan jumlah lagu yang diputar pada hari Senin, Rabu, dan Jumat.

[25]:
```
# Menghitung trek yang diputar pada masing-masing hari
print(df.groupby('day')['day'].count())
```

```
day
Friday       21840
```

```
Monday       21354
Wednesday    18059
Name: day, dtype: int64
```

Rabu adalah hari paling tenang secara keseluruhan. Tetapi jika kita mempertimbangkan kedua kota secara terpisah, kita mungkin akan memiliki kesimpulan yang berbeda.

Anda telah melihat cara kerja pengelompokan berdasarkan kota atau hari. Sekarang tulis fungsi yang akan dikelompokkan berdasarkan keduanya.

Buat fungsi `number_tracks()` untuk menghitung jumlah lagu yang diputar untuk hari dan kota tertentu. Ini akan membutuhkan dua parameter: * nama hari * nama kota

Dalam fungsi, gunakan variabel untuk menyimpan baris dari tabel asli, di mana: * Nilai kolom `'day'` sama dengan parameter `day` * Nilai kolom `'city'` sama dengan parameter `city`

Terapkan pemfilteran berurutan dengan pengindeksan logis.

Kemudian hitung nilai kolom `'user_id'` pada tabel yang dihasilkan. Simpan hasilnya ke variabel baru. Kembalikan variabel ini dari fungsi.

[26]:
```
# <membuat fungsi number_tracks()>
# Kita akan mendeklarasikan sebuah fungsi dengan dua parameter: day=, city=.
# Biarkan variabel track_list menyimpan baris df di mana
# nilai di kolom 'day' sama dengan parameter day= dan, pada saat yang sama,
# nilai pada kolom 'city' sama dengan parameter city= (terapkan pemfilteran␣
 ↪berurutan
# dengan pengindeksan logis).
# Biarkan variabel track_list_count menyimpan jumlah nilai kolom 'user_id' pada␣
 ↪track_list
# (temukan dengan metode count()).
# Biarkan fungsi menghasilkan jumlah: nilai track_list_count.

# Fungsi menghitung lagu yang diputar untuk kota dan hari tertentu.
# Pertama-tama ini akan mengambil baris dengan hari yang diinginkan dari tabel,
# kemudian memfilter baris hasilnya dengan kota yang dimaksud,
# kemudian temukan jumlah nilai 'user_id' pada tabel yang difilter,
# kemudian menghasilkan jumlah tersebut.
# Untuk melihat apa yang dihasilkan, balut pemanggilan fungsi pada print().

def number_tracks(df, day, city):
    track_list = df[df['day'] == day]
    track_list = track_list[track_list['city'] == city]
    track_list_count = track_list['user_id'].count()
    return(track_list_count)
```

Panggil `number_tracks()` enam kali, mengubah nilai parameter, sehingga Anda mengambil data di kedua kota untuk masing-masing hari tersebut.

[27]:
```
# jumlah lagu yang diputar di Springfield pada hari Senin
spr_mon = number_tracks(df=df, day='Monday', city='Springfield')
spr_mon
```

[27]: 15740

[28]:
```
# jumlah lagu yang diputar di Shelbyville pada hari Senin
shel_mon = number_tracks(df=df, day='Monday', city='Shelbyville')
shel_mon
```

[28]: 5614

[29]:
```
#  jumlah lagu yang diputar di Springfield pada hari Rabu
spr_wed = number_tracks(df=df, day='Wednesday', city='Springfield')
spr_wed
```

[29]: 11056

[30]:
```
#  jumlah lagu yang diputar di Shelbyville pada hari Rabu
shel_wed = number_tracks(df=df, day='Wednesday', city='Shelbyville')
shel_wed
```

[30]: 7003

[31]:
```
# jumlah lagu yang diputar di Springfield pada hari Jumat
spr_fri = number_tracks(df=df, day='Friday', city='Springfield')
spr_fri
```

[31]: 15945

[32]:
```
# jumlah lagu yang diputar di Shelbyville pada hari Jumat
shel_fri= number_tracks(df=df, day='Friday', city='Shelbyville')
shel_fri
```

[32]: 5895

Gunakan `pd.DataFrame` untuk membuat tabel, di mana * Nama kolom adalah: `['city', 'monday', 'wednesday', 'friday']` * Data adalah hasil yang Anda dapatkan dari `number_tracks()`

[33]:
```
# tabel dengan hasil

data = {
    'city': ['Springfield', 'Shelbyville'],
    'monday': [spr_mon, shel_mon],
    'wednesday': [spr_wed, shel_wed],
    'friday': [spr_fri, shel_fri]
}
```

```
df_result = pd.DataFrame(data)
df_result
```

[33]:

	city	monday	wednesday	friday
0	Springfield	15740	11056	15945
1	Shelbyville	5614	7003	5895

Kesimpulan

Data mengungkapkan perbedaan perilaku pengguna:

- Pada Springfield, jumlah lagu yang diputar mencapai puncaknya pada hari Senin dan Jumat, sedangkan pada hari Rabu terjadi penurunan aktivitas.
- Di Shelbyville, sebaliknya, pengguna lebih banyak mendengarkan musik pada hari Rabu.

Aktivitas pengguna pada hari Senin dan Jumat lebih sedikit.

Kembali ke Daftar Isi

2.4.2 Hipotesis 2: musik di awal dan akhir minggu

Menurut hipotesis kedua, pada Senin pagi dan Jumat malam, warga Springfield mendengarkan genre yang berbeda dari yang dinikmati warga Shelbyville.

Dapatkan tabel (pastikan nama tabel gabungan Anda cocok dengan DataFrame yang diberikan dalam dua blok kode di bawah): * Untuk Springfield — `spr_general` * Untuk Shelbyville — `shel_general`

[34]:
```
# mendapatkan tabel spr_general dari baris df,
# dimana nilai dari kolom 'city' adalah 'Springfield'

spr_general = df[df['city'] == 'Springfield']
```

[35]:
```
# mendapatkan shel_general dari baris df,
# dimana nilai dari kolom 'city' adalah 'Shelbyville'

shel_general = df[df['city'] == 'Shelbyville']
```

Tulis fungsi `genre_weekday()` dengan empat parameter: * Sebuah tabel untuk data * Nama hari * Tanda waktu pertama, dalam format 'hh:mm' * Tanda waktu terakhir, dalam format 'hh: mm'

Fungsi tersebut harus menghasilkan info tentang 15 genre paling populer pada hari tertentu dalam periode diantara dua tanda waktu.

[36]:
```
# Mendeklarasikan fungsi genre_weekday() dengan parameter day=, time1=, dan␣
 ↪time2=. Itu harus
# memberikan informasi tentang genre paling populer pada hari dan waktu␣
 ↪tertentu:

# 1) Biarkan variabel genre_df menyimpan baris yang memenuhi beberapa ketentuan:
```

```
#     - nilai pada kolom 'day' sama dengan nilai argumen hari=
#     - nilai pada kolom 'time' lebih besar dari nilai argumen time1=
#     - nilai pada kolom 'time' lebih kecil dari nilai argumen time2=
#     Gunakan pemfilteran berurutan dengan pengindeksan logis.

# 2) Kelompokkan genre_df berdasarkan kolom 'genre', lalu ambil salah satu␣
 ↪kolomnya,
#     dan gunakan metode count() untuk menemukan jumlah entri untuk masing-masing
#     genre yang diwakili; simpan Series yang dihasilkan ke
#     variabel genre_df_count

# 3) Urutkan genre_df_count dalam urutan menurun dan simpan hasilnya
#     ke variabel genre_df_sorted

# 4) Menghasilkan objek Series dengan nilai 15 genre_df_sorted pertama - 15␣
 ↪genre paling
#     populer (pada hari tertentu, dalam jangka waktu tertentu)

# tulis fungsi Anda di sini
def genre_weekday(df, day, time1, time2):

    # pemfilteran berturut-turut
    # genre_df hanya akan menyimpan baris df di mana day sama dengan day=
    genre_df = df[df['day'] == day]

    # genre_df hanya akan menyimpan baris df di mana time lebih besar dari␣
 ↪time1=
    genre_df = genre_df[genre_df['time'] > time1]

    # genre_df hanya akan menyimpan baris df di mana time lebih kecil dari␣
 ↪time2=
    genre_df = genre_df[genre_df['time'] < time2]

    # kelompokkan DataFrame yang difilter berdasarkan kolom dengan nama genre,␣
 ↪ambil kolom genre, dan temukan jumlah baris untuk setiap genre dengan metode␣
 ↪count()
    genre_df_grouped = genre_df.groupby('genre')['genre'].count()

    # kita akan mengurutkan hasilnya dalam urutan menurun (sehingga genre␣
 ↪paling populer didahulukan pada objek Series
    genre_df_sorted = genre_df_grouped.sort_values(ascending=False)

    # kita akan menghasilkan objek Series yang menyimpan 15 genre paling␣
 ↪populer pada hari tertentu dalam jangka waktu tertentu
    return genre_df_sorted[:15]
```

Bandingkan hasil fungsi `genre_weekday()` untuk Springfield dan Shelbyville pada Senin pagi (dari

pukul 07.00 hingga 11.00) dan pada Jumat malam (dari pukul 17:00 hingga 23:00):

[37]:
```
# memanggil fungsi untuk Senin pagi di Springfield (gunakan spr_general␣
 ↪alih-alih tabel df)
mon_mor_spr = genre_weekday(df=spr_general , day='Monday', time1='07:00',␣
 ↪time2='11:00')
mon_mor_spr
```

[37]:
```
genre
pop            781
dance          549
electronic     480
rock           474
hiphop         286
ruspop         186
world          181
rusrap         175
alternative    164
unknown        161
classical      157
metal          120
jazz           100
folk            97
soundtrack      95
Name: genre, dtype: int64
```

[38]:
```
# memanggil fungsi untuk Senin pagi di Shelbyville (gunakan shel_general␣
 ↪alih-alih tabel df)
mon_mor_shel = genre_weekday(df=shel_general , day='Monday', time1='07:00',␣
 ↪time2='11:00')
mon_mor_shel
```

[38]:
```
genre
pop            218
dance          182
rock           162
electronic     147
hiphop          80
ruspop          64
alternative     58
rusrap          55
jazz            44
classical       40
world           36
rap             32
soundtrack      31
rnb             27
```

```
metal           27
Name: genre, dtype: int64
```

[39]:
```
# memanggil fungsi untuk Jumat malam di Springfield
fri_eve_spr = genre_weekday(df=spr_general , day='Friday', time1='17:00',␣
 ↪time2='23:00')
fri_eve_spr
```

[39]:
```
genre
pop            713
rock           517
dance          495
electronic     482
hiphop         273
world          208
ruspop         170
classical      163
alternative    163
rusrap         142
jazz           111
unknown        110
soundtrack     105
rnb             90
metal           88
Name: genre, dtype: int64
```

[40]:
```
# memanggil fungsi untuk Jumat malam di Shelbyville
fri_eve_shel = genre_weekday(df=shel_general , day='Friday', time1='17:00',␣
 ↪time2='23:00')
fri_eve_shel
```

[40]:
```
genre
pop            256
rock           216
electronic     216
dance          210
hiphop          97
alternative     63
jazz            61
classical       60
rusrap          59
world           54
unknown         47
ruspop          47
soundtrack      40
metal           39
rap             36
```

```
Name: genre, dtype: int64
```

Kesimpulan

Setelah membandingkan 15 genre teratas pada Senin pagi, kita dapat menarik kesimpulan berikut:

1. Pengguna dari Springfield dan Shelbyville mendengarkan musik dengan genre yang sama. Lima genre teratas sama, hanya rock dan elektronik yang bertukar tempat.
2. Di Springfield, jumlah nilai yang hilang ternyata sangat besar sehingga nilai `'unknown'` berada di urutan ke-10. Ini berarti bahwa nilai-nilai yang hilang memiliki jumlah data yang cukup besar, yang mungkin menjadi dasar untuk mempertanyakan ketepatan kesimpulan kita.

Untuk Jumat malam, situasinya serupa. Genre individu agak bervariasi, tetapi secara keseluruhan, 15 besar genre untuk kedua kota sama.

Dengan demikian, hipotesis kedua sebagian terbukti benar: * Pengguna mendengarkan musik yang sama di awal dan akhir minggu. * Tidak ada perbedaan yang mencolok antara Springfield dan Shelbyville. Pada kedua kota tersebut, pop adalah genre yang paling populer.

Namun, jumlah nilai yang hilang membuat hasil ini dipertanyakan. Di Springfield, ada begitu banyak yang memengaruhi 15 teratas kita. Jika kita tidak mengabaikan nilai-nilai ini, hasilnya mungkin akan berbeda.

Kembali ke Daftar Isi

2.4.3 Hipotesis 3: preferensi genre di Springfield dan Shelbyville

Hipotesis: Shelbyville menyukai musik rap. Warga Springfield lebih menyukai pop.

Kelompokkan tabel `spr_general` berdasarkan genre dan temukan jumlah lagu yang dimainkan untuk setiap genre dengan metode `count()`. Kemudian urutkan hasilnya dalam urutan menurun dan simpan ke `spr_genres`.

[41]:
```
# pada satu baris: kelompokkan tabel spr_general berdasarkan kolom 'genre',
# hitung nilai 'genre' dengan count() dalam pengelompokan,
# urutkan Series yang dihasilkan dalam urutan menurun, lalu simpan ke spr_genres

spr_genres = spr_general.groupby('genre')['genre'].count().
 ↪sort_values(ascending=False)
```

Tampilkan 10 baris pertama dari `spr_genres`:

[42]:
```
# menampilkan 10 baris pertama dari spr_genres
spr_genres.head(10)
```

[42]:
```
genre
pop            5892
dance          4435
rock           3965
electronic     3786
```

```
hiphop          2096
classical       1616
world           1432
alternative     1379
ruspop          1372
rusrap          1161
Name: genre, dtype: int64
```

Sekarang lakukan hal yang sama pada data di Shelbyville.

Kelompokkan tabel `shel_general` berdasarkan genre dan temukan jumlah lagu yang dimainkan untuk setiap genre. Kemudian urutkan hasilnya dalam urutan menurun dan simpan ke tabel `shel_genres`:

[43]:
```
# pada satu baris: kelompokkan tabel shel_general menurut kolom 'genre',
# hitung nilai 'genre' dalam pengelompokan menggunakan count(),
# urutkan Series yang dihasilkan dalam urutan menurun dan simpan ke shel_genres

shel_genres = shel_general.groupby('genre')['genre'].count().
 ↪sort_values(ascending=False)
```

Tampilkan 10 baris pertama dari `shel_genres`:

[44]:
```
# menampilkan 10 baris pertama dari shel_genres
shel_genres.head(10)
```

[44]:
```
genre
pop             2431
dance           1932
rock            1879
electronic      1736
hiphop           960
alternative      649
classical        646
rusrap           564
ruspop           538
world            515
Name: genre, dtype: int64
```

Kesimpulan

Hipotesis terbukti benar sebagian: * Musik pop adalah genre paling populer di Springfield, seperti yang diharapkan. * Namun, musik pop ternyata sama populernya baik di Springfield maupun di Shelbyville, dan musik rap tidak berada di 5 besar untuk kedua kota tersebut.

Kembali ke Daftar Isi

3 Temuan

Kita telah menguji tiga hipotesis berikut:

1. Aktivitas pengguna berbeda-beda tergantung pada hari dan kotanya.
2. Pada senin pagi, penduduk Springfield dan Shelbyville mendengarkan genre yang berbeda. Hal ini juga ini juga berlaku untuk Jumat malam.
3. Pendengar di Springfield dan Shelbyville memiliki preferensi yang berbeda. Baik Springfield maupun di Shelbyville, mereka lebih suka musik pop.

Setelah menganalisis data, kita dapat menyimpulkan:

1. Aktivitas pengguna di Springfield dan Shelbyville bergantung pada harinya, walaupun kotanya berbeda.

Hipotesis pertama dapat diterima sepenuhnya.

2. Preferensi musik tidak terlalu berbeda selama seminggu di Springfield dan Shelbyville. Kita dapat melihat perbedaan kecil dalam urutan pada hari Senin, tetapi:

- Baik di Springfield maupun di Shelbyville, orang paling banyak mendengarkan musik pop.

Jadi hipotesis ini tidak dapat kita terima. Kita juga harus ingat bahwa hasilnya bisa berbeda jika bukan karena nilai yang hilang.

3. Ternyata preferensi musik pengguna dari Springfield dan Shelbyville sangat mirip.

Hipotesis ketiga ditolak. Jika ada perbedaan preferensi, tidak dapat dilihat dari data ini.

3.0.1 Catatan

Dalam proyek sesungguhnya, penelitian melibatkan pengujian hipotesis statistik, yang lebih tepat dan lebih kuantitatif. Perhatikan juga bahwa kamu tidak dapat selalu menarik kesimpulan tentang seluruh kota berdasarkan data dari satu sumber saja.

Anda akan mempelajari pengujian hipotesis dalam sprint analisis data statistik.

Kembali ke Daftar Isi