d7ea73a3-4e8c-461e-8c1c-52b223671cff

October 25, 2024

¡Hola, Tom!

Mi nombre es Tonatiuh Cruz. Me complace revisar tu proyecto hoy.

Al identificar cualquier error inicialmente, simplemente los destacaré. Te animo a localizar y abordar los problemas de forma independiente como parte de tu preparación para un rol como data-scientist. En un entorno profesional, tu líder de equipo seguiría un enfoque similar. Si encuentras la tarea desafiante, proporcionaré una pista más específica en la próxima iteración.

Encontrarás mis comentarios a continuación - **por favor no los muevas, modifiques o elimines**.

Puedes encontrar mis comentarios en cajas verdes, amarillas o rojas como esta:

Comentario del revisor

Éxito. Todo está hecho correctamente.

Comentario del revisor

Observaciones. Algunas recomendaciones.

Comentario del revisor

Necesita corrección. El bloque requiere algunas correcciones. El trabajo no puede ser aceptado con comentarios en rojo.

Puedes responderme utilizando esto:

Respuesta del estudiante.

Resumen de la revisión 1

Hola, Tom! Has hecho un excelente trabajo al realizar el proyecto, usaste diferentes herramientas apredidas en el curso.Ya solamente te dejo algunos comentarios para terminar de complementar los análisis.

Sigue con el excelente trabajo!

1 Déjame escuchar la música

2 Contenido

Comentario del revisor

La tabla de contenidos está bien estructurada, pero sería útil si estuviera enlazada a las secciones correspondientes, de manera que al hacer clic se pueda acceder directamente a cada una. Esto facilitaría la navegación.

Respuesta del estudiante. Hola Tonatiuh, qué tal? Si había notado que los vínculos no servían pero no le puse atención para repararlos y en realidad no sabía cómo funcionaban hasta ahora que me puse a revisarlos; ya los he reparado y ahora funcionan correctamente. Lo único que pude notar, es que los nombres de los encabezados se hicieron más grandes y no pude dejarlos del tamaño que tenían originalmente (más chicos) por ser subtítulos. Muchas gracias por la revisión! Saludos.

3 Introducción

Como analista de datos, tu trabajo consiste en analizar datos para extraer información valiosa y tomar decisiones basadas en ellos. Esto implica diferentes etapas, como la descripción general de los datos, el preprocesamiento y la prueba de hipótesis.

Siempre que investigamos, necesitamos formular hipótesis que después podamos probar. A veces aceptamos estas hipótesis; otras veces, las rechazamos. Para tomar las decisiones correctas, una empresa debe ser capaz de entender si está haciendo las suposiciones correctas.

En este proyecto, compararás las preferencias musicales de las ciudades de Springfield y Shelbyville. Estudiarás datos reales de transmisión de música online para probar la hipótesis a continuación y comparar el comportamiento de los usuarios y las usuarias de estas dos ciudades.

3.0.1 Objetivo:

Prueba la hipótesis: 1. La actividad de los usuarios y las usuarias difiere según el día de la semana y dependiendo de la ciudad.

3.0.2 Etapas

Los datos del comportamiento del usuario se almacenan en el archivo `/datasets/music_project_en.csv`. No hay ninguna información sobre la calidad de los datos, así que necesitarás examinarlos antes de probar la hipótesis.

Primero, evaluarás la calidad de los datos y verás si los problemas son significativos. Entonces, durante el preprocesamiento de datos, tomarás en cuenta los problemas más críticos.

Tu proyecto consistirá en tres etapas: 1. Descripción de los datos. 2. Preprocesamiento de datos. 3. Prueba de hipótesis.

Comentario del revisor

¡Hola! Excelente trabajo que has hecho al desarrollar la introducción. Esto es crucial para cualquier proyecto, ya que establece una guía clara sobre los pasos a seguir. Tener estos elementos bien definidos desde el principio nos permite trabajar de manera más organizada y eficiente. En un futuro lo podrías complementar con una tabla de contenido. Recuerda intentar agregar este apartado en todos tus proyectos.

Volver a Contenidos

4 Etapa 1. Descripción de los datos

Abre los datos y examínalos.

Necesitarás `pandas`, así que impórtalo.

[4]:
```
# Importar pandas
import pandas as pd
```

Lee el archivo `music_project_en.csv` de la carpeta `/datasets/` y guárdalo en la variable `df`:

[5]:
```
# Leer el archivo y almacenarlo en df
df = pd.read_csv('/datasets/music_project_en.csv', sep=',')
```

Comentario del revisor:

Has realizado un excelente trabajo al importar los datos y las bibliotecas necesarias.

Muestra las 10 primeras filas de la tabla:

[6]:
```
# Obtener las 10 primeras filas de la tabla df
df.head(10)
```

[6]:

	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
```

Obtén la información general sobre la tabla con un comando. Conoces el método que muestra la información general que necesitamos.

[7]:
```
# Obtener la información general sobre nuestros datos
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
```

Estas son nuestras observaciones sobre la tabla. Contiene siete columnas. Almacenan los mismos tipos de datos: `object`.

Según la documentación: - `' userID'`: identificador del usuario o la usuaria; - `'Track'`: título de la canción; - `'artist'`: nombre del artista; - `'genre'`: género de la pista; - `'City'`: ciudad del usuario o la usuaria; - `'time'`: la hora exacta en la que se reprodujo la canción; - `'Day'`: día de la semana.

Podemos ver tres problemas con el estilo en los encabezados de la tabla: 1. Algunos encabezados están en mayúsculas, otros en minúsculas. 2. Hay espacios en algunos encabezados. 3. Los nombres de las columnas no son muy descriptivos, falta claridad. `Detecta el tercer problema por tu cuenta y descríbelo aquí`.

5 Escribe observaciones de tu parte. Estas son algunas de las preguntas que pueden ser útiles:

1. ¿Qué tipo de datos tenemos a nuestra disposición en las filas? ¿Y cómo podemos entender lo que almacenan las columnas?

2. ¿Hay suficientes datos para proporcionar respuestas a nuestra hipótesis o necesitamos más información?

3. ¿Notaste algún problema en los datos, como valores ausentes, duplicados o tipos de datos incorrectos?

1R. En las columnas tenemos datos de tipo object o strings, son datos categóricos los cuales almacenan información descriptiva sobre los gustos preferenciales de música de usuarios de dos ciudades diferentes.

2R. Al parecer si tenemos los datos necesarios para comprobar la hipótesis, ya que tenemos una columna con la ciudad de los uduarios y otra columna con el día de la semana cuando escuchó cada una de las canciones.

3R. Si, en las columnas [‘Track’, ‘artist’ y ‘genre’] hay vaores ausentes, también se tienen 3,826 líneas duplicadas.

Comentario del revisor

Muy buen trabajo! Excelentes observaciones de tu parte.

Volver a Contenidos

6 Etapa 2. Preprocesamiento de datos

El objetivo aquí es preparar los datos para que sean analizados. El primer paso es resolver cualquier problema con los encabezados. Luego podemos avanzar a los valores ausentes y duplicados. Empecemos.

Corrige el formato en los encabezados de la tabla.

7 Estilo del encabezado

Muestra los encabezados de la tabla (los nombres de las columnas):

[8]:
```
# Muestra los nombres de las columnas
print(df.columns)
```

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

Cambia los encabezados de la tabla de acuerdo con las reglas del buen estilo: * Todos los caracteres deben ser minúsculas. * Elimina los espacios. * Si el nombre tiene varias palabras, utiliza snake_case.

Anteriormente, aprendiste acerca de la forma automática de cambiar el nombre de las columnas. Vamos a aplicarla ahora. Utiliza el bucle for para iterar sobre los nombres de las columnas y poner todos los caracteres en minúsculas. Cuando hayas terminado, vuelve a mostrar los encabezados de la tabla:

[9]:
```
# Bucle en los encabezados poniendo todo en minúsculas
new_col_names = []

for col in df.columns:
    name_lowered = col.lower()
```

```
    new_col_names.append(name_lowered)

df.columns = new_col_names

print(df.columns)
```

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

Comentario del revisor

Gran trabajo en hacer uso de un ciclo for para aplicar la función lower() en todas las columnas d ela base

Ahora, utilizando el mismo método, elimina los espacios al principio y al final de los nombres de las columnas e imprime los nombres de las columnas nuevamente:

[10]:
```
# Bucle en los encabezados eliminando los espacios
new_col_names = []

for col in df.columns:
    name_stripped = col.strip()
    new_col_names.append(name_stripped)

df.columns = new_col_names

print(df.columns)
```

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

Necesitamos aplicar la regla de snake_case a la columna `userid`. Debe ser `user_id`. Cambia el nombre de esta columna y muestra los nombres de todas las columnas cuando hayas terminado.

[11]:
```
# Cambiar el nombre de la columna "userid"
columns_new = {
    "userid": "user_id",
    }

df = df.rename(columns = columns_new)
```

Comprueba el resultado. Muestra los encabezados una vez más:

[12]:
```
# Comprobar el resultado: la lista de encabezados
print(df.columns)
```

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

Comentario del revisor

Muy buen trabajo! Ajustaste todos los elementos

Volver a Contenidos

8 Valores ausentes

Primero, encuentra el número de valores ausentes en la tabla. Debes utilizar dos métodos en una secuencia para obtener el número de valores ausentes.

[13]:
```
# Calcular el número de valores ausentes
print(df.isna().sum())
```

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

No todos los valores ausentes afectan a la investigación. Por ejemplo, los valores ausentes en `track` y `artist` no son cruciales. Simplemente puedes reemplazarlos con valores predeterminados como el string `'unknown'` (desconocido).

Pero los valores ausentes en `'genre'` pueden afectar la comparación entre las preferencias musicales de Springfield y Shelbyville. En la vida real, sería útil saber las razones por las cuales hay datos ausentes e intentar recuperarlos. Pero no tenemos esa oportunidad en este proyecto. Así que tendrás que: * rellenar estos valores ausentes con un valor predeterminado; * evaluar cuánto podrían afectar los valores ausentes a tus cómputos;

Reemplazar los valores ausentes en las columnas `'track'`, `'artist'` y `'genre'` con el string `'unknown'`. Como mostramos anteriormente en las lecciones, la mejor forma de hacerlo es crear una lista que almacene los nombres de las columnas donde se necesita el reemplazo. Luego, utiliza esta lista e itera sobre las columnas donde se necesita el reemplazo haciendo el propio reemplazo.

[14]:
```
# Bucle en los encabezados reemplazando los valores ausentes con 'unknown'
columns_to_replace = ['track', 'artist', 'genre']

for col in columns_to_replace:
    df[col].fillna('unknown', inplace=True)
```

Ahora comprueba el resultado para asegurarte de que después del reemplazo no haya valores ausentes en el conjunto de datos. Para hacer esto, cuenta los valores ausentes nuevamente.

[15]:
```
# Contar valores ausentes
print(df.isna().sum())
```

```
user_id    0
track      0
```

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

Comentario del revisor

Muy buen trabajo! Ajustaste todos los elementos con valores ausentes y los cambiaste con una etiqueta "unkown" que nos permite identificarlos

Volver a Contenidos

9 Duplicados

Encuentra el número de duplicados explícitos en la tabla. Una vez más, debes aplicar dos métodos en una secuencia para obtener la cantidad de duplicados explícitos.

[16]:
```
# Contar duplicados explícitos
print(df.duplicated().sum())
```

```
3826
```

Ahora, elimina todos los duplicados. Para ello, llama al método que hace exactamente esto.

[17]:
```
# Eliminar duplicados explícitos
df = df.drop_duplicates().reset_index(drop=True)
```

Comprobemos ahora si eliminamos con éxito todos los duplicados. Cuenta los duplicados explícitos una vez más para asegurarte de haberlos eliminado todos:

[18]:
```
# Comprobar de nuevo si hay duplicados
print(df.duplicated().sum())
```

```
0
```

Ahora queremos deshacernos de los duplicados implícitos en la columna `genre`. Por ejemplo, el nombre de un género se puede escribir de varias formas. Dichos errores también pueden afectar al resultado.

Para hacerlo, primero mostremos una lista de nombres de género únicos, ordenados en orden alfabético. Para ello: * Extrae la columna `genre` del DataFrame. * Llama al método que devolverá todos los valores únicos en la columna extraída.

[19]:
```
# Inspeccionar los nombres de géneros únicos
print(df['genre'].unique())
print()
print(df['genre'].nunique())
```

```
['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']
```

269

Busca en la lista para encontrar duplicados implícitos del género `hiphop`. Estos pueden ser nombres escritos incorrectamente o nombres alternativos para el mismo género.

Verás los siguientes duplicados implícitos: * `hip` * `hop` * `hip-hop`

Para deshacerte de ellos, crea una función llamada `replace_wrong_genres()` con dos parámetros: * `wrong_genres=`: esta es una lista que contiene todos los valores que necesitas reemplazar. * `correct_genre=`: este es un string que vas a utilizar como reemplazo.

Como resultado, la función debería corregir los nombres en la columna `'genre'` de la tabla `df`, es

decir, remplazar cada valor de la lista `wrong_genres` por el valor en `correct_genre`.

Dentro del cuerpo de la función, utiliza un bucle `'for'` para iterar sobre la lista de géneros incorrectos, extrae la columna `'genre'` y aplica el método `replace` para hacer correcciones.

[20]:
```
# Función para reemplazar duplicados implícitos

wrong_genres = ['hip', 'hop', 'hip-hop'] # Lista de nombres mal escritos.
correct_genre = 'hiphop' # El nombre correcto

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

Comentario del revisor

Excelente trabajo con la función. Con esto nos permitira reemplazar los registros duplicados con los valores que coloquemos dentro del arguneto de la función

Ahora, llama a `replace_wrong_genres()` y pásale tales argumentos para que retire los duplicados implícitos (`hip`, `hop` y `hip-hop`) y los reemplace por `hiphop`:

[21]:
```
# Eliminar duplicados implícitos
df = replace_wrong_genres(df, wrong_genres, correct_genre)
```

Asegúrate de que los nombres duplicados han sido eliminados. Muestra la lista de valores únicos de la columna `'genre'` una vez más:

[22]:
```
# Comprobación de duplicados implícitos
print(df['genre'].nunique())
```

```
266
```

Comentario del revisor

Muy buen trabajo! Remplazaste los generos incorrectos y eliminaste los duplicados.

Volver a Contenidos

10 Tus observaciones

`Describe brevemente lo que has notado al analizar duplicados, cómo abordaste sus eliminaciones y qué resultados obtuviste.`

Al analizar el Data Frame (df), de primer instancia los duplicados explicitos ascendian a la cantidad de 3,826 lineas en todo el df, los cuales fueron eliminados con el método .drop_duplicates() y comprobados nuevamente con el método .duplicated().sum() para asegurarnos de que no hubiese más duplicados.

Después de lo anterior, se buscaron duplicados implícitos en la columna de generos llamada 'genre', encontrando un total de 299 valores únicos de los cuales se tenían 3 nombres ['hip', 'hop', 'hip-hop']

que correspondían al mismo genero llamado 'hiphop'; después de corregirlos con una función que reemplazaba los tres nombres por uno sólo, quedaron un total de 266 géneros únicos.

Finalmente, se eliminaron los duplicados explicitos e implicitos para asegurar una mejor calidad de la información en el Data Frame.

Volver a Contenidos

11 Etapa 3. Prueba de hipótesis

12 Hipótesis: comparar el comportamiento del usuario o la usuaria en las dos ciudades

La hipótesis afirma que existen diferencias en la forma en que los usuarios y las usuarias de Springfield y Shelbyville consumen música. Para comprobar esto, usa los datos de tres días de la semana: lunes, miércoles y viernes.

- Agrupa a los usuarios y las usuarias por ciudad.
- Compara el número de canciones que cada grupo reprodujo el lunes, el miércoles y el viernes.

Realiza cada cálculo por separado.

El primer paso es evaluar la actividad del usuario en cada ciudad. Recuerda las etapas dividir-aplicar-combinar de las que hablamos anteriormente en la lección. Tu objetivo ahora es agrupar los datos por ciudad, aplicar el método apropiado para contar durante la etapa de aplicación y luego encontrar la cantidad de canciones reproducidas en cada grupo especificando la columna para obtener el recuento.

A continuación se muestra un ejemplo de cómo debería verse el resultado final: `df.groupby(by='....')['column'].method()`Realiza cada cálculo por separado.

Para evaluar la actividad de los usuarios y las usuarias en cada ciudad, agrupa los datos por ciudad y encuentra la cantidad de canciones reproducidas en cada grupo.

[23]:
```
# Contar las canciones reproducidas en cada ciudad
print(df.groupby('city')['track'].count())
```

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

`Comenta tus observaciones aquí` R. Al momento de agrupar las líneas por ciudad y contar el número de tracks o canciones reproducidas, podemos notar que en la ciudad de Springfield han escuchado más de doble de canciones (un total de 42,741) que en la ciudad de Shelbyville (18,512 canciones). Dato interesante.

Ahora agrupemos los datos por día de la semana y encontremos el número de canciones reproducidas el lunes, miércoles y viernes. Utiliza el mismo método que antes, pero ahora necesitamos una agrupación diferente.

[24]:
```
# Calcular las canciones reproducidas en cada uno de los tres días
print(df.groupby('day')['track'].count())
```

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

`Comenta tus observaciones aquí` R. Al agrupar las líneas por día de reporducción y contar el número de canciones, podemos notar que los viernes es cuando las personas escuchan más música (21,840 tracks) en comparación que los demás días de la semana, lunes un total de 21,354 tracks y a los miércoles es cuando escuchan menos música con un total de 18,059 tracks reproducidas; tal vez se deba a que a mitad de semana las personas se encuentran más enfocadas en su trabajo.

Ya sabes cómo contar entradas agrupándolas por ciudad o día. Ahora necesitas escribir una función que pueda contar entradas según ambos criterios simultáneamente.

Crea la función `number_tracks()` para calcular el número de canciones reproducidas en un determinado día **y** ciudad. La función debe aceptar dos parámetros:

- `day`: un día de la semana para filtrar. Por ejemplo, `'Monday'` (lunes).
- `city`: una ciudad para filtrar. Por ejemplo, `'Springfield'`.

Dentro de la función, aplicarás un filtrado consecutivo con indexación lógica.

Primero filtra los datos por día y luego filtra la tabla resultante por ciudad.

Después de filtrar los datos por dos criterios, cuenta el número de valores de la columna 'user_id' en la tabla resultante. Este recuento representa el número de entradas que estás buscando. Guarda el resultado en una nueva variable y devuélvelo desde la función.

[25]:
```
# Declara la función number_tracks() con dos parámetros: day= y city=.
def number_tracks(day, city):

    # Almacena las filas del DataFrame donde el valor en la columna 'day' es␣
 ↪igual al parámetro day=
    df_filtered = df[df['day'] == day]

    # Filtra las filas donde el valor en la columna 'city' es igual al␣
 ↪parámetro city=
    df_filtered = df_filtered[df_filtered['city'] == city]

    # Extrae la columna 'user_id' de la tabla filtrada y aplica el método␣
 ↪count()
    df_users = df_filtered['user_id'].count()

    # Devuelve el número de valores de la columna 'user_id'
    return df_users
```

Comentario del revisor

Realizaste un gran trabajo con la definición de la función. Como pudiste notar, el desarrollar funciones nos permiten encapsular lógica reutilizable, evitando la repetición de código y facilitando su uso en diferentes partes de un programa o en distintos proyectos.

Llama a `number_tracks()` seis veces, cambiando los valores de los parámetros para que recuperes los datos de ambas ciudades para cada uno de los tres días.

[26]:
```
# El número de canciones reproducidas en Springfield el lunes
Springfield_lunes = number_tracks('Monday', 'Springfield')
print(f'El número de canciones reproducidas en Springfield el día lunes:',␣
 ↪Springfield_lunes)
```

```
El número de canciones reproducidas en Springfield el día lunes: 15740
```

[27]:
```
# El número de canciones reproducidas en Shelbyville el lunes
Shelbyville_lunes = number_tracks('Monday', 'Shelbyville')
print(f'El número de canciones reproducidas en Shelbyville el día lunes:',␣
 ↪Shelbyville_lunes)
```

```
El número de canciones reproducidas en Shelbyville el día lunes: 5614
```

[77]:
```
# El número de canciones reproducidas en Springfield el miércoles
Springfield_miercoles = number_tracks('Wednesday', 'Springfield')
print(f'El número de canciones reproducidas en Springfield el día miércoles:',␣
 ↪Springfield_miercoles)
```

```
El número de canciones reproducidas en Springfield el día miércoles: 11056
```

[78]:
```
# El número de canciones reproducidas en Shelbyville el miércoles
Shelbyville_miercoles = number_tracks('Wednesday', 'Shelbyville')
print(f'El número de canciones reproducidas en Shelbyville el día miércoles:',␣
 ↪Shelbyville_miercoles)
```

```
El número de canciones reproducidas en Shelbyville el día miércoles: 7003
```

[79]:
```
# El número de canciones reproducidas en Springfield el viernes
Springfield_viernes = number_tracks('Friday', 'Springfield')
print(f'El número de canciones reproducidas en Springfield el día viernes:',␣
 ↪Springfield_viernes)
```

```
El número de canciones reproducidas en Springfield el día viernes: 15945
```

[80]:
```
# El número de canciones reproducidas en Shelbyville el viernes
Shelbyville_viernes = number_tracks('Friday', 'Shelbyville')
print(f'El número de canciones reproducidas en Shelbyville el día viernes:',␣
 ↪Shelbyville_viernes)
```

```
El número de canciones reproducidas en Shelbyville el día viernes: 5895
```

Conclusiones

`Comenta si la hipótesis es correcta o se debe rechazar. Explica tu razonamiento.`
R. La hipótesis es correcta, ya que todos los datos analisados muestran cantidades diferentes de reproducciones por día de la semana, es un hecho que ningun día y ningun usuario por ciudad, consume la misma música.

Mientras van pasando los días de la semana se muestran diferentes tendencias, siendo los miércoles los días que menos consumen música los usuarios de Springfield (MIN: 11,056 canciones); mientras que los miércoles los usuarios de Shelbyville consumen más música (MAX: 7,003 canciones).

Comentario del revisor

Hola! Excelente trabajo con la conclusión. En futuros cursos vas a poder complementar estas pruebas con pruebas estadisticas por las pruebas de diferencias de medias, además vas a poder explorar el desarrollo de gráficas que complementen este análisis. Por ejemplo en este caso podríamos desarrollar una gráfica que muestre la cantidad de canciones por día y otra que muestre la relación por ciudad. Para esto puedes aplicar la función que creaste para obtener la información necesaria para hacer estas gráficas

Volver a Contenidos

13 Conclusiones

`Resume aquí tus conclusiones sobre la hipótesis.` R. En conclusión las tendencias del consumo de música son muy variadas, no sólo los usuarios de cada ciudad, ni tampoco los días de la semana; si se realizara un análisis más profundo considerando los generos musicales, artistas, etc., tendriamos una gran variedad de consumo de la música. Sin duda el análisis de los datos es una rama muy interesante.

13.0.1 Nota

En proyectos de investigación reales, la prueba de hipótesis estadística es más precisa y cuantitativa. También ten en cuenta que no siempre se pueden sacar conclusiones sobre una ciudad entera a partir de datos de una sola fuente.

Aprenderás más sobre la prueba de hipótesis en el sprint de análisis estadístico de datos.

Comentario general

Tom, hiciste un muy buen trabajo con el proyecto, pudiste resolver todos los ejericios de una forma adecuada y se nota que hiciste uso de todas las herramientas aprendidas. Además, considero que te ayudo para aprender el uso de algunas funciones que te sirvan en los siguientes cursos. Solamente verifica lo de la tabla de contenidos y no cargar en más de una vez las bases de datos.

Exitos en lo que viene, saludos.

Respuesta del estudiante. Tonatiuh, ya he leido todos tus comentarios y he atendido las observaciones que me hiciste, tanto la reparación de los vínculos en el Índice principal, como no volver a carcgar la lista de géneros más que sólo una vez. Nuevamente te agradezco la revisión y todos los comentarios, sin duda un excelente ejercicio que me ayuda a aprender y desarrollarme en esta nueva profesión! Gracias & Saludos, Tom

Volver a Contenidos