Sprint 3 phyton proyect

August 12, 2024

1 ¡Hola Jafet!

Mi nombre es Ezequiel Ferrario, soy code reviewer en Tripleten y tengo el agrado de revisar el proyecto que entregaste.

Para simular la dinámica de un ambiente de trabajo, si veo algún error, en primer instancia solo los señalaré, dándote la oportunidad de encontrarlos y corregirlos por tu cuenta. En un trabajo real, el líder de tu equipo hará una dinámica similar. En caso de que no puedas resolver la tarea, te daré una información más precisa en la próxima revisión.

Encontrarás mis comentarios más abajo - **por favor, no los muevas, no los modifiques ni los borres**.

¿Cómo lo voy a hacer? Voy a leer detenidamente cada una de las implementaciones que has llevado a cabo para cumplir con lo solicitado. Verás los comentarios de esta forma:

Comentario del revisor

Si todo está perfecto.

Comentario del revisor

Si tu código está bien pero se puede mejorar o hay algún detalle que le hace falta. Se aceptan uno o dos comentarios de este tipo en el borrador, pero si hay más, deberá hacer las correcciones. Es como una tarea de prueba al solicitar un trabajo: muchos pequeños errores pueden hacer que un candidato sea rechazado.

Comentario del revisor

Si de pronto hace falta algo o existe algún problema con tu código o conclusiones.

Puedes responderme de esta forma:

Respuesta del estudiante

Hola, muchas gracias por tus comentarios y la revisión.

¡Empecemos!

Comentario general #1

Jafet, entregaste un buen proyecto ya que hasta el punto de corte en general supiste manejarte a traves de todos los metodos.

A su vez, el error que se observa en los encabezados hace que se produzcan otros. De igual manera el proyecto esta encaminado y tus analisis me gustaron mucho.

Quedo atento a tu correccion, saludos.

Comentario general #2

Jafet, las correcciones estuvieron excelentes por lo que finalizas un muy buen proyecto.

Por lo expuesto anteriormente, el trabajo queda **aprobado**.

Exitos en lo que viene, saludos.

---

2 Déjame escuchar la música

3 Contenido

Comentario del revisor

Muy bien la tabla de contenidos pero debe estar linkeada a las secciones (al clickear debe llevarnos a esa seccion) de esta menera es mas facil desplazarse.

Como consejo, si realizas bien todas las secciones (con su respectivo #) podes generarlo automáticamente desde jupyter lab. Para hacerlo, en la pestaña de herramientas de jupyter lab clickeas en el **botón de los puntos y barras** (Table of contents) te generara automáticamente una tabla de contenidos linkeable y estética. A la **derecha** del seleccionador del tipo de celda (code/markdown)

3.1 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.

3.1.1 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.

3.1.2 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.

Volver a Contenidos

3.2 Etapa 1. Descripción de los datos

Abre los datos y examínalos.

Necesitarás `pandas`, así que impórtalo.

[3]:
```
import pandas as pd # Importar pandas
```

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

[4]:
```
df = pd.read_csv('/datasets/music_project_en.csv') # Leer el archivo y␣
↪almacenarlo en df
```

Muestra las 10 primeras filas de la tabla:

[5]:
```
df.head(10) # Obtener las 10 primeras filas de la tabla df
```

[5]:

	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
```

Comentario del revisor

Esta bien. Pero en este caso remplaza el print() y utiliza el metodo sin nada (llamalo solo). Es decir llamas a la variable y al metodo correspondiente y ejecutas.

- *El ejemplo seria: nombre_df.metodo(10)*

Esto hara que sea mas legible y estetico. LISTO

Comentario del revisor #2

Corregido, muy bien.

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

[6]:
```
df.info() # Obtener la información general sobre nuestros datos
```

```
<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. UserID no posee el formato standar de separación entre palabras con el guión bajo.

3.2.1 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?` En nuestros datos tenemos el numero unico de identificador de usuario, tenemos el nombre de las canciones, tenemos el nombre del artista, tenemos los tipos de genero musical, tenemos los nombres de las ciudades en donde se encuentra el usuario, tenemos la hora exacta que se reprodujo la canción y tenemos el día de la semana.

2. `¿Hay suficientes datos para proporcionar respuestas a nuestra hipótesis o necesitamos más información?` Por el momento tenemos los datos necesarios para hacer el analisis comparativo entre los preferencias y habitos de repreducción musical entre los habitantes de Springfield y Shellbyville.

3. `¿Notaste algún problema en los datos, como valores ausentes, duplicados o tipos de datos incorrectos?` Que hay valores NaN en 3 columnas: Track, Artist & Genre

Comentario del revisor

En este caso nos enfrentamos con nulos en 3 columnas (comenta cuales).

Al metodo info podemos interpretarlo de la siguiente manera:

`RangeIndex: 65079 entries` -> Este parametro nos indica cuantas entrada de datos hay.

Asi, todas aquellas columnas que poseean menos datos **no nulos** que ese numero, son columnas que poseen datos faltantes.

Comenta esas 3 columnas en el analisis. LISTO

Comentario del revisor #2

Corregido.

Volver a Contenidos

3.3 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.

3.3.1 Estilo del encabezado

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

[7]:
```
print(df.columns) # Muestra los nombres de las columnas
```

```
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:

[8]:
```
df.columns = [column.lower() for column in df.columns] # Bucle en los␣
↪encabezados poniendo todo en minúsculas
```

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:

[9]:
```
df.columns = [column.strip() for column in df.columns] # Bucle en los␣
↪encabezados eliminando los espacios
```

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.

[10]:
```
df.rename(columns={'userid': 'user_id'}, inplace=True) # Cambiar el nombre de␣
↪la columna "userid"
```

Comprueba el resultado. Muestra los encabezados una vez más:

[11]:
```
print(df.columns) # Comprobar el resultado: la lista de encabezados
```

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

Comentario del revisor

Cuidado, en este caso no estas cambiando los encabezaados, recorda hacer un bucle en cada celda, estas llamando variables que no tienen un bucle. # LISTO!

Comentario del revisor #2

Corregido.

Volver a Contenidos

3.3.2 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.

[12]:
```
mis_val = df.isna().sum() # Calcular el número de valores ausentes
print(mis_val)
```

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

Comentario del revisor

No importes pandas ya que lo has hecho al principio, no constituye una buena practica hacerla todo el tiempo. Otra cosa que no se debe hacer es cargar nuevamente el dataset(con el read). Ya que estaras todo el tiempo cancelando tu progreso con el mismo, en el momento que lo haces reinicias todos los cambios, por esto es que se carga solo al principio.

Cuando hacemos un notebook lo que hacemos al principio se mantiene ejecutado a lo largo de todo el proyecto, por eso realizamos por etapa el codigo y no todo en una sola celda. # Perfecto, gracias maestro

Comentario del revisor #2

Corregido.

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.

[13]:
```
columns_to_replace = ['track', 'artist', 'genre'] # Bucle en los encabezados␣
 ↪reemplazando los valores ausentes con 'unknown'
for col in columns_to_replace:
    df[col] = df[col].fillna('unknown')
```

Comentario del revisor

No importes pandas ya que lo has hecho al principio, no constituye una buena practica hacerla todo el tiempo. Otra cosa que no se debe hacer es cargar nuevamente el dataset(con el read). Ya que

estaras todo el tiempo cancelando tu progreso con el mismo, en el momento que lo haces reinicias todos los cambios, por esto es que se carga solo al principio.

Cuando hacemos un notebook lo que hacemos al principio se mantiene ejecutado a lo largo de todo el proyecto, por eso realizamos por etapa el codigo y no todo en una sola celda. LISTO

Comentario del revisor #2

Corregido, perfecto.

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

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

Volver a Contenidos

3.3.3 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]:
```
print(df.duplicated().sum()) # Contar duplicados explícitos
```

```
3826
```

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

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

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

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

```
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]:
```
genre_column = df['genre']    # Inspeccionar los nombres de géneros únicos
unique_genres = genre_column.unique()
print(unique_genres)
```

```
['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']
```

Comentario del revisor

En este caso, ordenar estos valores hara que sea mucho mas claro encontrar errores y correcciones. Siempre es mejor para la legibilidad.

Contando la cantidad de elementos que tenemos (antes y despues) tambien es parametro para verificar si el error esta corregido.

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]:
```
def replace_wrong_genres(df, column, wrong_genres, correct_genre):  #␣
↪Función para reemplazar duplicados implícitos
    for wrong_genre in wrong_genres:
        df[column] = df[column].replace(wrong_genre, correct_genre)
    return df
```

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]:
```
# Lista de duplicados implícitos y el nombre correcto del género
duplicates = ['hip', 'hop', 'hip-hop']
correct_genre = 'hiphop'

# Llama a la función para reemplazar los géneros incorrectos
df = replace_wrong_genres(df, 'genre', duplicates, 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]:
```
print(df['genre'].unique()) # Comprobación de duplicados implícitos
```

```
['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']
```

Comentario del revisor

En este caso, ordenar estos valores hara que sea mucho mas claro encontrar errores y correcciones. Siempre es mejor para la legibilidad.

Contando la cantidad de elementos que tenemos (antes y despues) tambien es parametro para verificar si el error esta corregido.

Volver a Contenidos

3.3.4 Tus observaciones

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

Al principio pude ver cuantos datos duplicados tenia la tabla para posteriormente eliminarlos tanto duplicados explicitos como los implicitos. Quitar esos datos nos permite tener una tabla integra y limpia con los datos.

Comentario del revisor

Buenas observaciones.

Volver a Contenidos

3.4 Etapa 3. Prueba de hipótesis

3.4.1 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]:
```
city_grouped = df.groupby(by='city')['track'].count()# Contar las canciones␣
↪reproducidas en cada ciudad
print(city_grouped)
```

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

Por alguna razón, se reproduce mas del doble de canciones en Springfield que en Shelbyville.

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.

[26]:
```
df_filtered = df[df['day'].isin(['Monday', 'Wednesday', 'Friday'])] # Calcular␣
↪las canciones reproducidas en cada uno de los tres días
day_grouped = df_filtered.groupby('day')['track'].count()
print(day_grouped)
```

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

Comentario del revisor

En este caso el error se genera ya que no cambiaste de forma efectiva la columna `day` y te quedo con mayuscula, corrigiendo el punto de arriba y ejecutando todo el proyecto deberia correr el codigo. Listo!

Comentario del revisor #2

Corregido, muy bien.

Aparentemente los usuarios suelen reproducir menos musica los miercoles, a comparación de los lunes y viernes por alguna razón.

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.

[30]:
```
def number_tracks(day, city): # Declara la función number_tracks() con dos␣
 ↪parámetros: day= y city=.

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

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

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

    return track_count     # Devolve el número de valores de la columna 'user_id'
```

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.

[31]:
```
springfield_monday = number_tracks('Monday', 'Springfield') # El número de␣
 ↪canciones reproducidas en Springfield el lunes
print(springfield_monday)
```

```
15740
```

[32]:
```
# El número de canciones reproducidas en Shelbyville el lunes
shelbyville_monday = number_tracks('Monday', 'Shelbyville')
print(shelbyville_monday)
```

```
5614
```

[33]:
```
# El número de canciones reproducidas en Springfield el miércoles
springfield_wednesday = number_tracks('Wednesday', 'Springfield')
print(springfield_wednesday)
```

```
11056
```

[34]:
```
# El número de canciones reproducidas en Shelbyville el miércoles
shelbyville_wednesday = number_tracks('Wednesday', 'Shelbyville')
print(shelbyville_wednesday)
```

```
7003
```

[35]:
```
# El número de canciones reproducidas en Springfield el viernes
springfield_friday = number_tracks('Friday', 'Springfield')
print(springfield_friday)
```

```
15945
```

[36]:
```
# El número de canciones reproducidas en Shelbyville el viernes
shelbyville_friday = number_tracks('Friday', 'Shelbyville')
print(shelbyville_friday)
```

```
5895
```

Conclusiones

La hipotesis es correcta, ya que los usuarios tienden a reproducir canciones de la plataforma en diferente cantidad dependiendo del día de la semana. Por alguna razón, en Springfield el comportamiento de la tendencia es: Tanto los lunes como los viernes el registro sobrepasan las 15,000 reproducciones y los miercoles regisra solo por encima de las 11,000 reproducciones, es decir un poco menos. Por otra parte, en Shelbyville por alguna razón el comportamiento es lo contrario, ya que en los dias miercoles se reproducen mas canciones que van por encima de las 7,000 reproducciones a diferencia de los dfias lunes y viernes que se reproducen menos canciones que rondan por las 5,000 unidades.

`Comenta si la hipótesis es correcta o se debe rechazar. Explica tu razonamiento.`

Volver a Contenidos

4 Conclusiones

`Resume aquí tus conclusiones sobre la hipótesis.`

Conociendo la información cuantitativa de el comportamiento de reproducción de música los dias lunes, miercoles y viernes en ambas ciudades, podemos establecer ahora hipotesis sobre las posibles causas que originan estos fenomenos, y por ende podemos ver la imagen completa de cuales son las razones por la que el comportamiento ocurre y se repite y eso nos permitira tener un punto de vista amplio y sustentado con información precisa sobre los datos para tomar desiciones.

Comentario del revisor #2

Muy buenas conclusiones.

4.0.1 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.

Volver a Contenidos