

# CHURN PREDICTION PARA PLATAFORMA MUSICAL

PRESENTACIÓN ENTREGAS 01 Y 02

AGUSTINA QUIRÓS  
NATALIA GRASSELLI  
NATALIA TASSIN  
RODRIGO PIZARRO

# OBJETIVO DEL PROYECTO

DESARROLLAR UN  
MODELO DE  
APRENDIZAJE  
AUTOMÁTICO PARA  
PREDECIR AQUELLOS  
CLIENTES QUE  
CANCELARÁN LA  
SUSCRIPCIÓN A UNA  
PLATAFORMA DE  
STREAMING DE MÚSICA

## LOS DATOS

SPARKIFY ES UN FALSO  
SERVICIO DE STREAMING DE  
MÚSICA INVENTADO POR  
UDACITY. EL CONJUNTO DE  
DATOS REGISTRA LA  
INFORMACIÓN DEMOGRÁFICA  
DE LOS USUARIOS Y LA  
ACTIVIDAD CON LA  
PLATAFORMA EN MARCAS DE  
TIEMPO INDIVIDUALES.

MINI\_SPARKIFY\_EVENT\_DATA.ZIP

# TAMAÑO


El dataset cuenta con 543.705 registros y 18 columnas. Cada fila representa una interacción entre algún usuario y la aplicación. Las interacciones que se muestran son entre el 01/10/18 y el 01/12/18

# TIPOS DE DATOS

La mayoría de las variables del dataset, como page, method, level, gender, etc son categóricas. Las variables numéricas identificadas fueron usserId, ts, sessionId, lenght, etc.

# LAS VARIABLES

Al estudiar el df notamos que las columnas poseen información que puede clasificarse en tres categorías diferentes: a nivel usuario, a nivel actividad y a nivel canción.

✓ 0 s  clients\_activity.shape  
(543705, 18)









```
[ ] #Observamos de qué tipo es la información que contiene el df
clients_activity.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 543705 entries, 0 to 543704
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   ts                     543705 non-null  int64
1   userId                543705 non-null  object
2   sessionId              543705 non-null  int64
3   page                  543705 non-null  object
4   auth                  543705 non-null  object
5   method                543705 non-null  object
6   status                543705 non-null  int64
7   level                 543705 non-null  object
```

```
#Observamos las columnas  
clients_activity.columns
```

```
Index(['ts', 'userId', 'sessionId', 'page', 'auth', 'method',  
       'level', 'itemInSession', 'location', 'userAgent', 'la  
       'firstName', 'registration', 'gender', 'artist', 'song  
       dtype='object'])
```

# Variables a nivel usuario

-  UserId
-  FirstName
-  LastName
-  Gender
-  Location
-  UserAgent
-  Registration
-  Level

```
[ ] #UsersId únicos  
clients_activity.userId.nunique()
```

449

449 USUARIOS  
ÚNICOS



# Variables a nivel interacción



TS



Auth



itemInSession



Page



Session Id



Method



Status

# Variables a nivel canción



Song



Length



Artist





# ¿QUÉ ES CHURN?



CHURN HACE REFERENCIA A LA TASA DE ABANDONO DE LOS CLIENTES DE LA PLATAFORMA DURANTE UN PERIODO DE TIEMPO ESPECÍFICO.

EN ESTE TRABAJO, DECIMOS QUE UN USUARIO HACE CHURN SI EL MISMO **CANCELA SU SUSCRIPCIÓN A LA PLATAFORMA.**

EN EL DATASET ESTO ESTÁ INDICADO POR LA ACCIÓN **"CANCELLATION CONFIRMATION"**

Columna  
Target

## CHURN\_USER

Indica con un 1 a los usuarios que alguna vez realizaron churn y con un 0 a los que nunca realizaron churn

```
[ ] # Agregamos la nueva columna churn_user
clients_activity_churn['churn_user'] = np.where(clients_activity_churn['userId']

[ ] #El usuario tendrá ahora en todas sus filas la marca de churn_user = 1, porque
clients_activity_churn[clients_activity_churn['userId']=='208'][['userId','date
```

	userId	date	level	churn_action	churn_user
2540	208	2018-10-01 13:25:50	free	False	1
2549	208	2018-10-01 13:29:10	free	False	1
2553	208	2018-10-01 13:30:08	free	False	1
2554	208	2018-10-01 13:30:09	free	False	1
2555	208	2018-10-01 13:30:16	free	True	1

# Balance de Clases

99  
Usuarios  
Churn

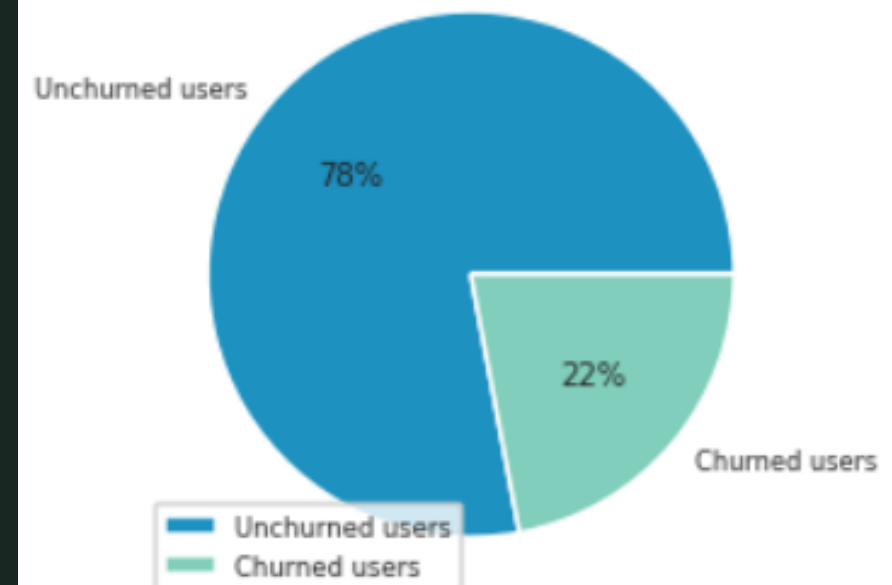


vs

350  
Usuarios  
No Churn



Usuarios Churn vs No Churn



# Usuarios Churn vs Usuarios No Churn

ANÁLISIS DEL COMPORTAMIENTO DE ESTAS CLASES



Features  
NO  
relevantes

## DISTRIBUCIONES Y/O COMPORTAMIENTOS ANÁLOGOS EN AMBAS CLASES

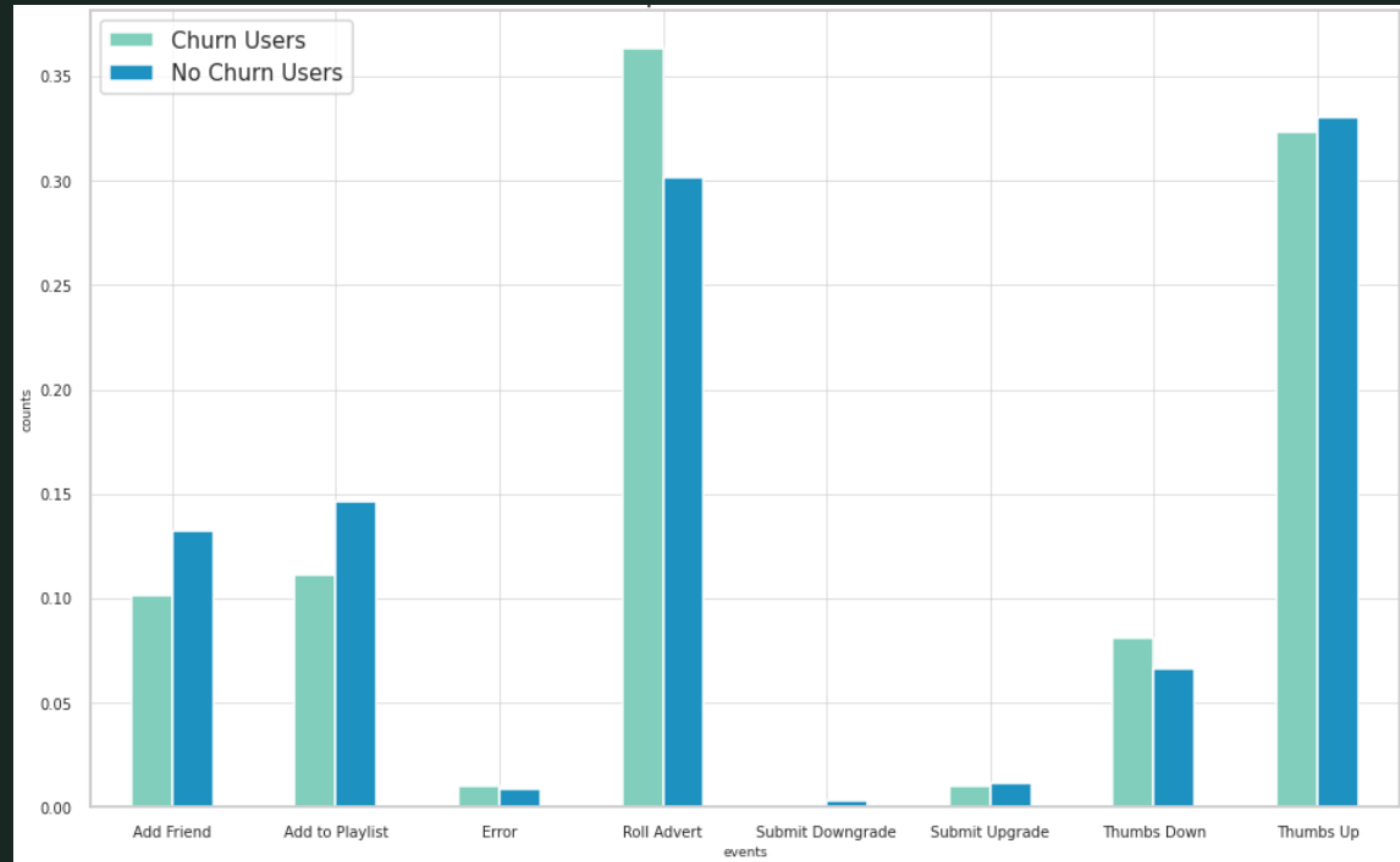
Uso a diferentes horas del día	Uso en diferentes días de la semana	Ubicación del usuario
Dispositivo utilizado	Distribución por género	Página más visitada
Artistas más populares	Canciones más populares	Interacciones con la página "Help"

O bien, las diferencias encontradas no aportan información significativa para poder predecir la acción "churn" en los usuarios.

# Features interesantes

## TIPOS DE EVENTOS

En los eventos "Roll Advert", "Thumbs Down" y en "Error" (en menor medida) se observa un MAYOR conteo en la **clase Churn** versus **clase No Churn**



# Features interesantes

¿QUE CLASE DE USUARIOS ESCUCHAN MÁS CANCIONES?

```
[ ] #medidas estadísticas para esas cantidades  
churn_song_count_df.describe()
```

	counts
count	99.000
mean	852.111
std	1054.788
min	7.000
25%	164.500
50%	439.000
75%	1243.500
max	6233.000

```
[ ] #medidas estadísticas para esas cantidades  
not_churn_song_count_df.describe()
```

	counts
count	349.000
mean	998.619
std	1176.055
min	1.000
25%	196.000
50%	601.000
75%	1316.000
max	8177.000



"La **clase No Churn** hace un uso más intensivo de esta plataforma para escuchar canciones en comparación con la **clase Churn**"

# Features interesantes

¿CUÁNTOS ARTISTAS Y CANCIONES ÚNICOS TIENE EL CONJUNTO DE DATOS?

```
[ ] print('total de artistas únicas:')

#canciones unicas en total
print('para el total usuarios', clients_activity_churn.artist.nunique())

#canciones unicas para el conjunto no churn
print('para el usuarios no churn', not_churn_df.artist.nunique())

#canciones unicas para el conjunto churn
print('para el usuarios churn', churn_df.artist.nunique())
```

```
total de artistas únicas:
para el total usuarios 21247
para el usuarios no churn 19992
para el usuarios churn 12151
```

"La **clase no churn** escucha una mayor variedad de artistas y canciones en la plataforma"

```
[ ] print('total de canciones únicas:')

#canciones unicas en total
print('para el total usuarios', clients_activity_churn.song.nunique())

#canciones unicas para el conjunto no churn
print('para el usuarios no churn', not_churn_df.song.nunique())

#canciones unicas para el conjunto churn
print('para el usuarios churn', churn_df.song.nunique())
```

```
total de canciones únicas:
para el total usuarios 80292
para el usuarios no churn 72419
para el usuarios churn 33143
```



USER ID

PAGE

SONG & ARTIST

LEVEL

SESSIONID

TS & DATE

REGISTRATION & REG\_DATE

COLUMNA TARGET: CHURN\_USER

# Columnas interesantes

Definidas en función de todo el  
análisis realizado a lo largo del TP1

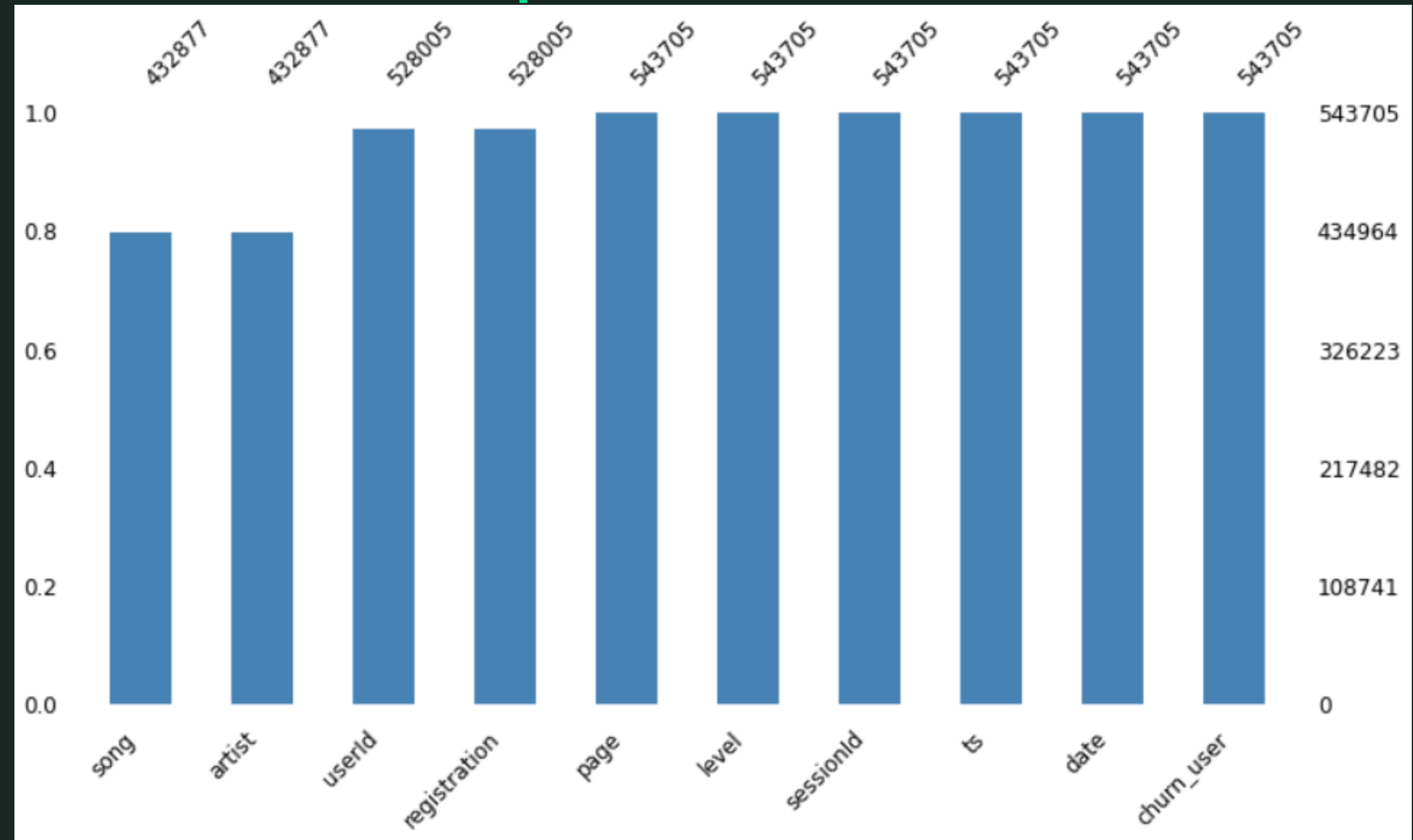
# Visualización Datos Nulos

## Cantidad total en Df

```
[7] df_churn.isna().sum().sum()
```

253056

## Proporción sobre el total



song y artist 20%  
userId y registration 2%



# Visualización Datos Nulos

## Valores nulos referidos a las canciones

clients\_activity\_format[clients\_activity\_format["song"].isnull()][["artist", "length"]]

	artist	length
3	NaN	NaN
6	NaN	NaN
7	NaN	NaN
8	NaN	NaN
9	NaN	NaN
...	...	...
543673	NaN	NaN
543678	NaN	NaN
543681	NaN	NaN
543683	NaN	NaN
543698	NaN	NaN

110828 rows × 2 columns

## Valores nulos referidos a la identificación de usuario

clients\_activity\_format[clients\_activity\_format["userAgent"].isnull()][["location", "userAgent", "lastName", "firstName", "registration", "gender"]]

	location	userAgent	lastName	firstName	registration	gender
6	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN
70	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
543561	NaN	NaN	NaN	NaN	NaN	NaN
543625	NaN	NaN	NaN	NaN	NaN	NaN
543626	NaN	NaN	NaN	NaN	NaN	NaN
543655	NaN	NaN	NaN	NaN	NaN	NaN
543656	NaN	NaN	NaN	NaN	NaN	NaN

15700 rows × 6 columns

#Queremos ver qué valores toma 'auth' cuando 'userAgent' es nulo

clients\_activity\_format[clients\_activity\_format['userAgent'].isna()][['auth']].value\_counts()

Logged Out	15606
Guest	94

## Posibles valores erróneos

Que un usuario que ve un anuncio esté clasificado como pago:

```
[ ] print('cantidad de usuarios pagos que ven anuncios:',
        df_clean[(df_clean.page == 'Roll Advert') &
                  (df_clean.level == 'Paid')].size)
```

cantidad de usuarios pagos que ven anuncios: 0

UserIds o Sessiolds negativos o nulos

```
#cantidad de userIds no positivos
print('cantidad de userIds no positivos:',
      df_clean[df_clean.userId <=0].size)

#cantidad de sessionIds no positivos
print('cantidad de sessionIds no positivos:',
      df_clean[df_clean.sessionId <=0].size)
```

cantidad de userIds no positivos: 0  
cantidad de sessionIds no positivos: 0

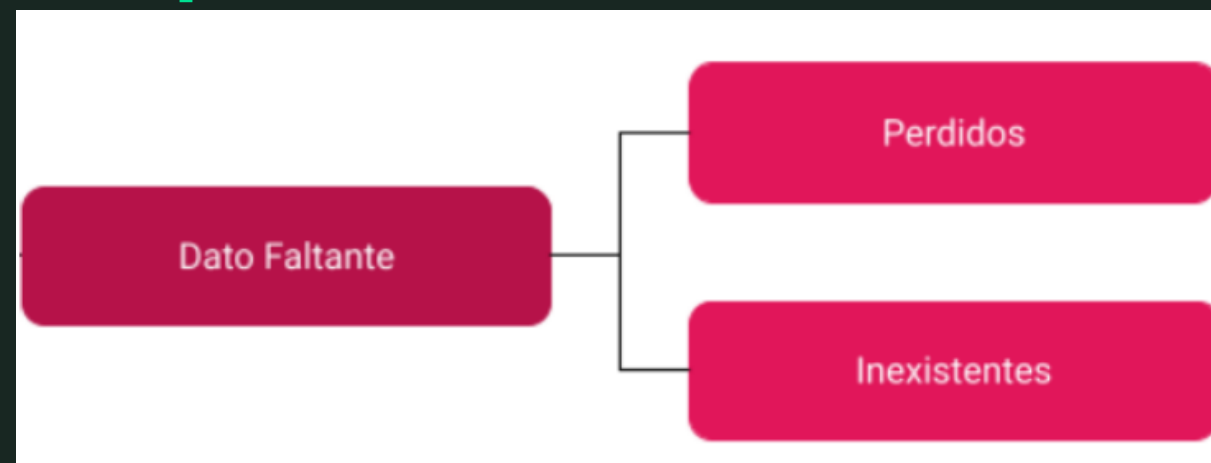
Registration dates posteriores a la fecha de la acción:

```
[ ] print('registration dates mal computadas:',
        df_clean[df_clean.registration <= df_clean.ts].size)
```

registration dates mal computadas: 0

# Tratamiento de Dato Faltante

## Tipos de Datos Faltantes



## Eliminación de variables

- **userId** -> **filas**
- **song** -> **columna**
- **artist** -> **columna**

```
df_clean.dropna(subset=['userId'],how='any',inplace=True)
df_clean['userId'].isnull().sum()
```

0

## Valores Duplicados

```
[ ] #eliminamos valores duplicados
df_clean.drop_duplicates(inplace=True)
```

```
[ ] #Ya no hay registros duplicados
df_clean.duplicated().sum()
```

0

## Dataframe luego de limpieza

```
dropped_rows = df_churn.shape[0] - df_clean.shape[0]
print('Nº filas eliminadas:', dropped_rows)
print('Porcentaje de filas eliminadas:', round(100*dropped_rows/df_churn.shape[0],2),'%')
```

Nº filas eliminadas: 15739  
Porcentaje de filas eliminadas: 2.89 %

## Validación que no haya NaNs antes de PCA

```
▶ scaled_df.isna().sum().sum()
```

👤 0



# Nuevas features

- + Cantidad de canciones únicas escuchadas por usuario
- + Cantidad de canciones escuchadas por sesión
- + Número de amigos agregados
- + Cantidad de anuncios
- + Cantidad de errores
- + Cantidad de interacciones por usuario

## COLUMNAS IRRELEVANTES

- Nombre de la canción
- Duración de la canción
- Nombre del artista

# One Hot Encodig

## COLUMNAS CATEGORICAS

```
categorical_cols = ['page', 'level']
```

**Page:** 19 grados de libertad

**Level:** 2 grados de libertad

## ESTADO DEL DATA FRAME

```
matrix
```

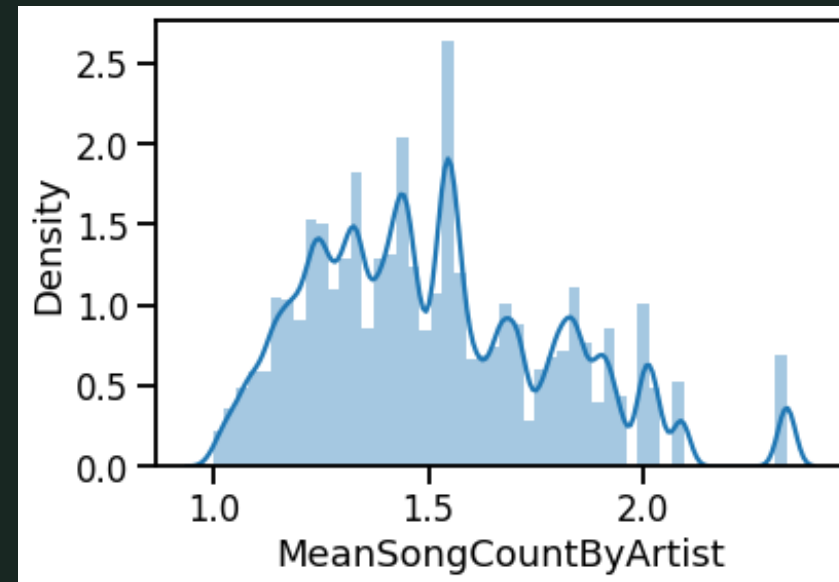
```
<527966x35 sparse matrix of type '<class 'numpy.float64'>'  
  with 7925315 stored elements in COOrdinate format>
```

## Ejemplo valores categóricos

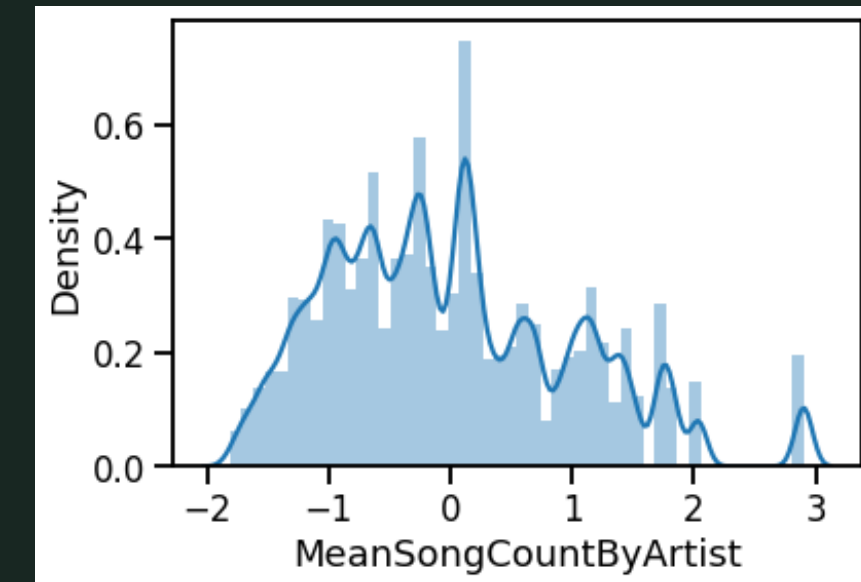
	page	level
0	NextSong	free
1	NextSong	free
2	NextSong	paid

# Estandarización

Ejemplo sin  
estandarizar



Ejemplo  
estandarizado



## POR QUE NO NORMALIZAR?

- Se pierde la proporcionalidad
- Para PCA conviene tener estandarización

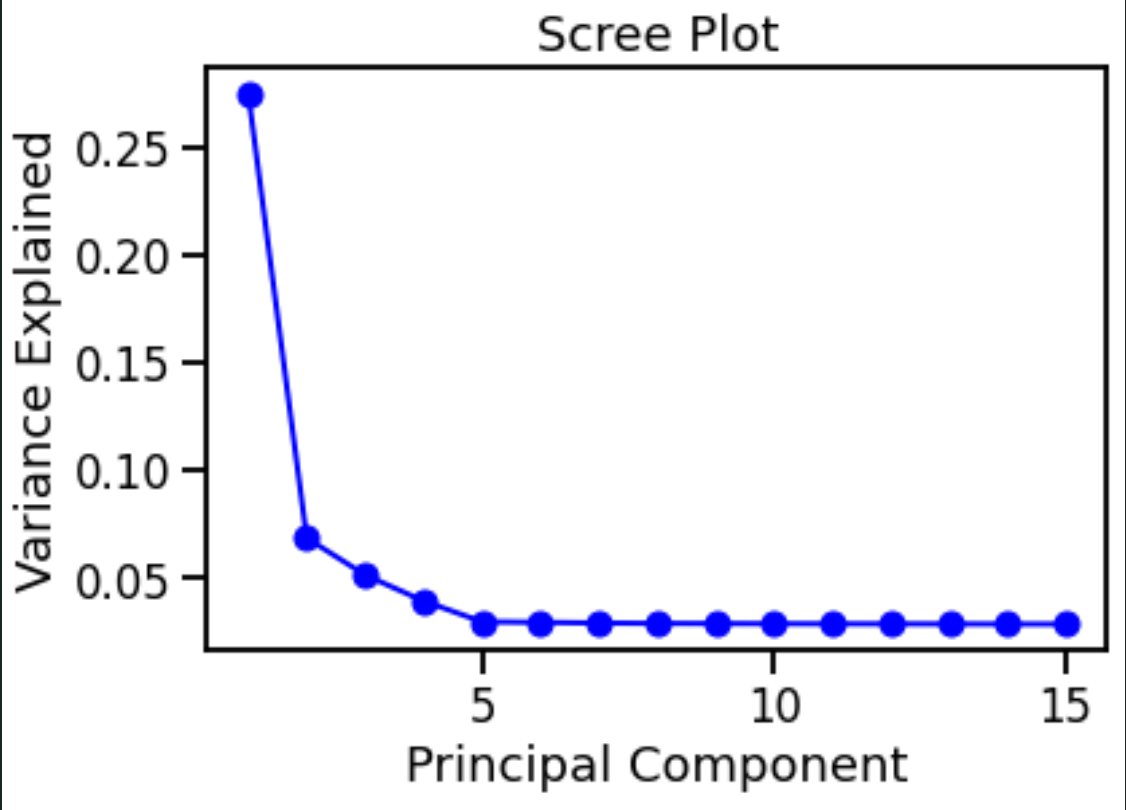
## VARIABLES NUMÉRICAS NO ESTANDARIZADAS

```
user_cols = ['userId', 'churn_user', 'sessionId', 'ts', 'registration']
```



# PCA

## APLICACIÓN



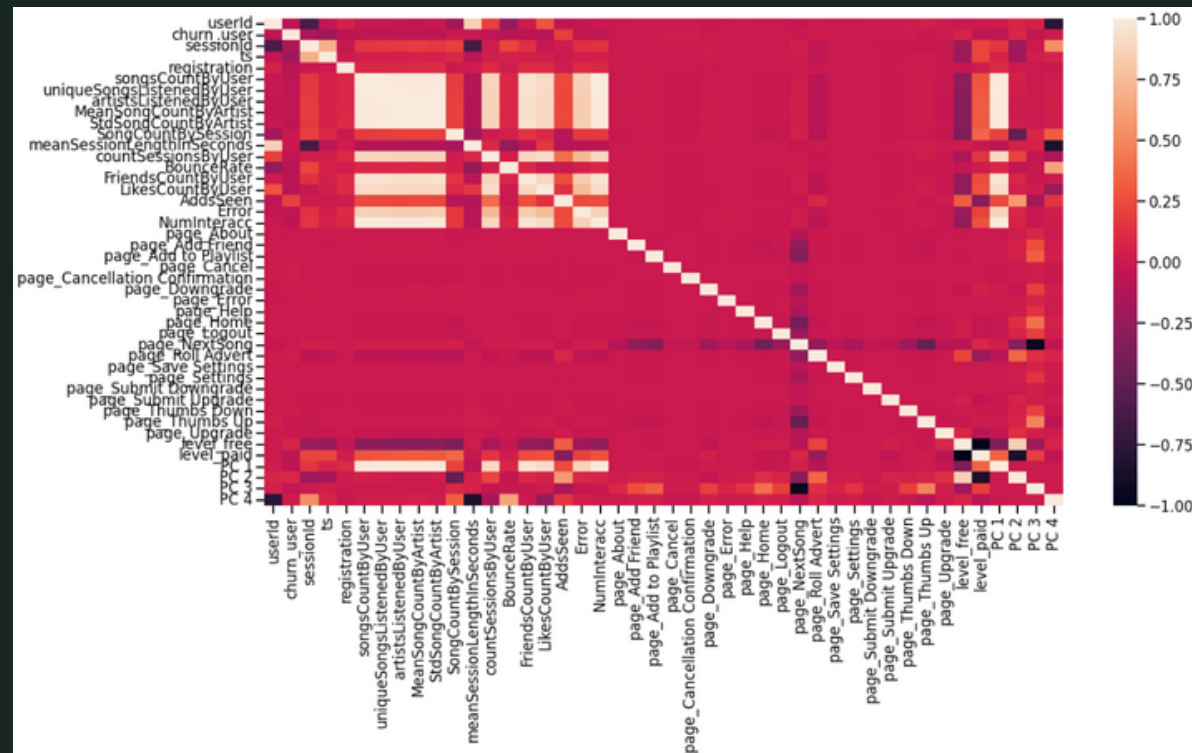
## RESULTADO

PC 1	PC 2	PC 3	PC 4
2.921311	3.118063	-1.445328	-0.175674
0.444724	2.190267	-1.259644	0.623613
-0.330131	-1.035579	-0.392073	0.140676
-0.402951	-0.039177	2.906686	0.585034
1.127700	-0.600384	-0.445908	-0.528950
...	...	...	...



# Correlaciones

## MATRIZ DE CORRELACIONES



## RESULTADO FINAL

Como data frame final, quedó una tabla con 38 columnas, entre las que se encuentra:

- La variable target.
- Los 4 componentes principales de PCA.
- El resultado del OneHot Encoding de .las variables categóricas (21 columnas)
- 8 columnas numéricas estandarizadas.
- 4 columnas con valores relativos al usuario.

## FEATURES ELIMINADAS

Las variables por encima de este valor son: `uniqueSongsListenedByUser` (0.99), `artistsListenedByUser` (0.98), `MeanSongCountByArtist` (0.98), `StdSongCountByArtist` (0.98), `FriendsCountByUser` (0.95), `LikesCountByUser` (0.91).

**¡Gracias!**