Sparkify

September 20, 2022

1 Sparkify Churn Prediction

2 Problem Definition

We want to create a churn prediction classifier.

We will use data from a fictive company called Sparkify, a music streaming company. The dataset contains all kinds of events created by the users who interacted with the platform. Such as when they logged in, when they are listening to music, and when they are unsubscribing from the platform.

The goal of this notebook is to leverage this data to create a model that can forecast customer churn. Therefore, we can understand better why the users are leaving the platform and we can adapt and improve their overall experience.

3 Import Libraries

[1]:
```
import findspark
findspark.init()

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.window import Window

from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression, GBTClassifier,␣
↪NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.sql.types import LongType
```

4 Define Spark Session

[2]:
```
spark = SparkSession.\
    builder.\
    appName("Sparkify Churn Prediction").\
    getOrCreate()
```

```
22/09/20 08:27:27 WARN Utils: Your hostname, iusztin-MS-7C91 resolves to a
loopback address: 127.0.1.1; using 192.168.0.170 instead (on interface enp42s0)
22/09/20 08:27:27 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/opt/spark/jars/spark-unsafe_2.12-3.2.1.jar) to constructor
java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of
org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/09/20 08:27:27 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

5 Load and Clean Dataset

[3]:
```
EVENT_DATA_LINK = "mini_sparkify_event_data.json"
df = spark.read.json(EVENT_DATA_LINK)
df.persist()

df.printSchema()
```

```
root
 |-- artist: string (nullable = true)
 |-- auth: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- itemInSession: long (nullable = true)
 |-- lastName: string (nullable = true)
 |-- length: double (nullable = true)
 |-- level: string (nullable = true)
 |-- location: string (nullable = true)
 |-- method: string (nullable = true)
```

```
 |-- page: string (nullable = true)
 |-- registration: long (nullable = true)
 |-- sessionId: long (nullable = true)
 |-- song: string (nullable = true)
 |-- status: long (nullable = true)
 |-- ts: long (nullable = true)
 |-- userAgent: string (nullable = true)
 |-- userId: string (nullable = true)
```

We can observe that all the columns are `strings` or `longs`.

[4]:
```
df.show(n=10)
```

```
+--------------------+---------+---------+------+-------------+--------+--------
-+-----+--------------------+------+---------------+-------------+---------+----
----------------+------+-------------+--------------------+------+
|              artist|     auth|firstName|gender|itemInSession|lastName|
length|level|            location|method|           page|
registration|sessionId|                song|status|           ts|
userAgent|userId|
+--------------------+---------+---------+------+-------------+--------+--------
-+-----+--------------------+------+---------------+-------------+---------+----
----------------+------+-------------+--------------------+------+
|      Martha Tilston|Logged In|    Colin|     M|           50|
Freeman|277.89016| paid|     Bakersfield, CA|   PUT|
NextSong|1538173362000|       29|           Rockpools|
200|1538352117000|Mozilla/5.0 (Wind…|    30|
|    Five Iron Frenzy|Logged In|    Micah|     M|           79|
Long|236.09424| free|Boston-Cambridge-…|   PUT|       NextSong|1538331630000|
8|               Canada|   200|1538352180000|"Mozilla/5.0 (Win…|     9|
|        Adam Lambert|Logged In|    Colin|     M|           51| Freeman|
282.8273| paid|     Bakersfield, CA|   PUT|       NextSong|1538173362000|
29|   Time For Miracles|   200|1538352394000|Mozilla/5.0 (Wind…|    30|
|              Enigma|Logged In|    Micah|     M|           80|
Long|262.71302| free|Boston-Cambridge-…|   PUT|       NextSong|1538331630000|
8|Knocking On Forbi…|   200|1538352416000|"Mozilla/5.0 (Win…|     9|
|           Daft Punk|Logged In|    Colin|     M|           52|
Freeman|223.60771| paid|     Bakersfield, CA|   PUT|
NextSong|1538173362000|       29|Harder Better Fas…|
200|1538352676000|Mozilla/5.0 (Wind…|    30|
|The All-American …|Logged In|    Micah|     M|           81|
Long|208.29995| free|Boston-Cambridge-…|   PUT|       NextSong|1538331630000|
8|       Don't Leave Me|   200|1538352678000|"Mozilla/5.0 (Win…|     9|
|The Velvet Underg…|Logged In|    Micah|     M|           82|
Long|260.46649| free|Boston-Cambridge-…|   PUT|       NextSong|1538331630000|
8|          Run Run Run|   200|1538352886000|"Mozilla/5.0 (Win…|     9|
|        Starflyer 59|Logged In|    Colin|     M|           53|
```

```
Freeman|185.44281| paid|     Bakersfield, CA|   PUT|
NextSong|1538173362000|       29|Passengers (Old A…|
200|1538352899000|Mozilla/5.0 (Wind…|    30|
|                null|Logged In|    Colin|     M|           54| Freeman|
null| paid|     Bakersfield, CA|   PUT|Add to Playlist|1538173362000|       29|
null|   200|1538352905000|Mozilla/5.0 (Wind…|    30|
|            Frumpies|Logged In|    Colin|     M|           55|
Freeman|134.47791| paid|     Bakersfield, CA|   PUT|
NextSong|1538173362000|       29|          Fuck Kitty|
200|1538353084000|Mozilla/5.0 (Wind…|    30|
+--------------------+---------+---------+------+-------------+--------+--------
-+-----+--------------------+------+---------------+-------------+---------+----
---------------+------+-------------+--------------------+------+
only showing top 10 rows
```

5.1 Check for Empty Values

5.1.1 Check for NaNs

[5]:
```python
df.select([F.count(F.when(F.isnan(c), c)).alias(c) for c in df.columns]).show()
```

```
[Stage 2:>                                                         (0 + 12) / 12]

+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|     0|   0|        0|     0|            0|       0|     0|    0|       0|
0|   0|           0|        0|   0|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
```

There are no **nans** within the data.

5.1.2 Check for Nones

[6]:
```python
df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.columns]).show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId| song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+---------+------+
```

```
| 58392|   0|      8346|  8346|            0|     8346| 58392|    0|     8346|
0|   0|         8346|         0|58392|     0|   0|      8346|      0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+--------+------+
```

Instead of `nans` the missing entries are filled with `Nones`.

5.1.3 Check for Empty Strings

[7]:
```python
df.select([F.count(F.when(F.col(c) == "", c)).alias(c) for c in df.columns]).
↪show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+--------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+--------+------+
|     0|   0|        0|     0|            0|       0|     0|    0|       0|
0|   0|            0|        0|   0|     0|  0|        0|  8346|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+--------+------+
```

[8]:
```python
df.filter(F.col("userId") != "").select([F.count(F.when(F.isnull(c), c)).
↪alias(c) for c in df.columns]).show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+--------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId| song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+--------+------+
| 50046|   0|        0|     0|            0|       0| 50046|    0|       0|
0|   0|            0|        0|50046|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+--------+------+
```

[9]:
```python
df.filter(F.col("userId") != "").select("auth").groupby("auth").count().show()
```

```
+---------+------+
|     auth| count|
+---------+------+
|Cancelled|    52|
|Logged In|278102|
+---------+------+
```

We can observe that some `users` have the `id` as an `empty string`. The number of open user ids, `8346`, equals the number of None properties, such as: `firstName`, `lastName`, `registration`, `userAgent`. Also, those events have the `auth` state `Cancelled` or `Logged In`. This means that the users who do not have an ID are those who `haven't registered`, yet, into the platform.

When the `artist`, `song`, and `length` columns are `None`, the user stays on pages that do not include listening to music.

5.2 Check for Other Unwanted Symbols

[10]:
```
df.select([F.count(F.when(F.col(c) == "-", c)).alias(c) for c in df.columns]).
↪show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|     0|   0|        0|     0|            0|       0|     0|    0|       0|
0|   0|           0|        0|   0|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
```

[11]:
```
df.select([F.count(F.when(F.col(c) == "NaN", c)).alias(c) for c in df.columns]).
↪show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|     0|   0|        0|     0|            0|       0|     0|    0|       0|
0|   0|           0|        0|   0|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
```

[12]:
```
df.select([F.count(F.when(F.col(c) == "None", c)).alias(c) for c in df.
↪columns]).show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
```

```
-+----+------------+---------+----+------+---+--------+------+
|     0|   0|         0|      0|             0|        0|      0|     0|        0|
0|   0|           0|         0|   0|      0|   0|         0|      0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+--------+------+
```

None of those symbols can be found within the dataset.

[13]:
```python
df.select([F.count(F.when(F.col(c).cast("int").isNull(), c)).alias(c) for c in␣
 ↪df.columns]).show()
```

```
+------+------+---------+------+-------------+--------+------+------+--------+--
----+------+------------+---------+------+------+---+--------+------+
|artist|  auth|firstName|gender|itemInSession|lastName|length|
level|location|method|  page|registration|sessionId|  song|status|
ts|userAgent|userId|
+------+------+---------+------+-------------+--------+------+------+--------+--
----+------+------------+---------+------+------+---+--------+------+
|286277|286500|   286500|286500|            0|  286500| 58392|286500|
286500|286500|286500|        8346|        0|286110|     0|   0|   286500|  8346|
+------+------+---------+------+-------------+--------+------+------+--------+--
----+------+------------+---------+------+------+---+--------+------+
```

There are no `sessionIds` that couldn't be cast to an int type.

5.3 Unregistered Users

5.3.1 Check Unregistered Users Page Distribution

[14]:
```python
df.filter(F.col("userId") == "").select("page").groupby("page").count().show()
```

```
+-------------------+-----+
|               page|count|
+-------------------+-----+
|               Home| 4375|
|              About|  429|
|              Login| 3241|
|               Help|  272|
|              Error|    6|
|           Register|   18|
|Submit Registration|    5|
+-------------------+-----+
```

Because there is no valuable activity for churn prediction performed by empty users, we can drop them out of the DataFrame.

5.4 Drop Unregistered Users

[15]:
```
cleaned_df = df.filter(F.col("userId") != "")
cleaned_df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.
 ↪columns]).show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId| song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+---------+------+
| 50046|   0|        0|     0|            0|       0| 50046|    0|       0|
0|   0|           0|        0|50046|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+-----+------+---+---------+------+
```

5.5 Empty Artists

5.5.1 Check Empty Artists Distribution

[16]:
```
df.filter(F.isnull(F.col("artist"))).select(["artist", "song", "userId",␣
 ↪"page"]).show()
```

```
+------+----+------+---------------+
|artist|song|userId|           page|
+------+----+------+---------------+
|  null|null|    30|Add to Playlist|
|  null|null|     9|    Roll Advert|
|  null|null|     9|      Thumbs Up|
|  null|null|    54|      Downgrade|
|  null|null|    54|      Thumbs Up|
|  null|null|     9|    Thumbs Down|
|  null|null|     9|           Home|
|  null|null|     9|         Logout|
|  null|null|    74|      Thumbs Up|
|  null|null|      |           Home|
|  null|null|      |           Help|
|  null|null|      |           Home|
|  null|null|      |          Login|
|  null|null|     9|           Home|
|  null|null|    30|    Thumbs Down|
|  null|null|     4|         Logout|
|  null|null|      |           Home|
|  null|null|      |          Login|
|  null|null|     4|           Home|
|  null|null|    74|Add to Playlist|
```

```
+-------+----+------+---------------+
only showing top 20 rows
```

[17]:
```python
df.filter(F.isnull(F.col("artist"))).select(["artist", "song", "userId",␣
↪"page"]).groupby("page").count().show()
```

```
+--------------------+-----+
|                page|count|
+--------------------+-----+
|              Cancel|   52|
|    Submit Downgrade|   63|
|         Thumbs Down| 2546|
|                Home|14457|
|           Downgrade| 2055|
|         Roll Advert| 3933|
|              Logout| 3226|
|       Save Settings|  310|
|Cancellation Conf…|   52|
|               About|  924|
|            Settings| 1514|
|               Login| 3241|
|     Add to Playlist| 6526|
|          Add Friend| 4277|
|           Thumbs Up|12551|
|                Help| 1726|
|             Upgrade|  499|
|               Error|  258|
|      Submit Upgrade|  159|
|            Register|   18|
+--------------------+-----+
only showing top 20 rows
```

We can observe that when the artist is null, the users spend time on different pages than actually listening to music. In this case, such information is valuable for understanding the behavior of registered users.

5.5.2 Impute empty values

[18]:
```python
# Fill the length of the song with 0.
# Fill the artist and the song with a string constant to signal that those␣
↪pages don't have such information.
cleaned_df = cleaned_df.fillna({
    "length": 0,
    "artist": "unknown",
    "song": "unknown"
})
```

```
cleaned_df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.
↪columns]).show()
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|metho
d|page|registration|sessionId|song|status| ts|userAgent|userId|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
|     0|   0|        0|     0|            0|       0|     0|    0|       0|
0|   0|           0|        0|   0|     0|  0|        0|     0|
+------+----+---------+------+-------------+--------+------+-----+--------+-----
-+----+------------+---------+----+------+---+---------+------+
```

Great! Now our data no longer has any `None` values. The last cleaning step is to check the types of the variables.

5.6 Check Types

[19]:
```
cleaned_df.printSchema()
```

```
root
 |-- artist: string (nullable = false)
 |-- auth: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- itemInSession: long (nullable = true)
 |-- lastName: string (nullable = true)
 |-- length: double (nullable = false)
 |-- level: string (nullable = true)
 |-- location: string (nullable = true)
 |-- method: string (nullable = true)
 |-- page: string (nullable = true)
 |-- registration: long (nullable = true)
 |-- sessionId: long (nullable = true)
 |-- song: string (nullable = false)
 |-- status: long (nullable = true)
 |-- ts: long (nullable = true)
 |-- userAgent: string (nullable = true)
 |-- userId: string (nullable = true)
```

[20]:
```
cleaned_df.show(n=5)
```

```
+----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+
```

```
|          artist|     auth|firstName|gender|itemInSession|lastName|
length|level|            location|method|    page| registration|sessionId|
song|status|           ts|           userAgent|userId|
+-----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+
|  Martha Tilston|Logged In|    Colin|     M|           50| Freeman|277.89016|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|
Rockpools|   200|1538352117000|Mozilla/5.0 (Wind…|    30|
|Five Iron Frenzy|Logged In|    Micah|     M|           79|    Long|236.09424|
free|Boston-Cambridge-…|   PUT|NextSong|1538331630000|        8|
Canada|   200|1538352180000|"Mozilla/5.0 (Win…|     9|
|    Adam Lambert|Logged In|    Colin|     M|           51| Freeman| 282.8273|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|   Time For
Miracles|   200|1538352394000|Mozilla/5.0 (Wind…|    30|
|          Enigma|Logged In|    Micah|     M|           80|    Long|262.71302|
free|Boston-Cambridge-…|   PUT|NextSong|1538331630000|        8|Knocking On
Forbi…|   200|1538352416000|"Mozilla/5.0 (Win…|     9|
|       Daft Punk|Logged In|    Colin|     M|           52| Freeman|223.60771|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|Harder Better
Fas…|   200|1538352676000|Mozilla/5.0 (Wind…|    30|
+-----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+
only showing top 5 rows
```

All the types are looking all right. We can proceed to EDA.

6 Exploratory Data Analysis

6.0.1 Define Churn

We will consider that a user is churn when the subscription is canceled.

[21]:
```
cleaned_df.select(F.count(F.when(F.col("page") == "Cancellation Confirmation",␣
↪"page")).alias("Cancellation Confirmation")).show()
```

```
+-------------------------+
|Cancellation Confirmation|
+-------------------------+
|                       52|
+-------------------------+
```

[22]:
```
# First, create the churn event based on the visited page.
# The churnEvent will be different for every event/visited page.
```

```
labeled_df = cleaned_df.withColumn("churnEvent", F.when(F.col("page") ==␣
↪"Cancellation Confirmation", 1).otherwise(0))
labeled_df.show(n=5)
```

```
+----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+----------+
|          artist|     auth|firstName|gender|itemInSession|lastName|
length|level|            location|method|    page| registration|sessionId|
song|status|           ts|           userAgent|userId|churnEvent|
+----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+----------+
|  Martha Tilston|Logged In|    Colin|     M|           50| Freeman|277.89016|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|
Rockpools|   200|1538352117000|Mozilla/5.0 (Wind…|    30|         0|
|Five Iron Frenzy|Logged In|    Micah|     M|           79|    Long|236.09424|
free|Boston-Cambridge-…|   PUT|NextSong|1538331630000|        8|
Canada|   200|1538352180000|"Mozilla/5.0 (Win…|     9|         0|
|    Adam Lambert|Logged In|    Colin|     M|           51| Freeman| 282.8273|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|   Time For
Miracles|   200|1538352394000|Mozilla/5.0 (Wind…|    30|         0|
|          Enigma|Logged In|    Micah|     M|           80|    Long|262.71302|
free|Boston-Cambridge-…|   PUT|NextSong|1538331630000|        8|Knocking On
Forbi…|   200|1538352416000|"Mozilla/5.0 (Win…|     9|         0|
|       Daft Punk|Logged In|    Colin|     M|           52| Freeman|223.60771|
paid|     Bakersfield, CA|   PUT|NextSong|1538173362000|       29|Harder Better
Fas…|   200|1538352676000|Mozilla/5.0 (Wind…|    30|         0|
+----------------+---------+---------+------+-------------+--------+---------+--
---+--------------------+------+--------+-------------+---------+---------------
-----+------+-------------+--------------------+------+----------+
only showing top 5 rows
```

[23]:
```
# Now create the churn label which is unique for every user.
labeled_df = labeled_df.withColumn("churn", F.sum("churnEvent").over(Window.
↪partitionBy("userId")))
labeled_df = labeled_df.withColumn("churn", F.when(F.col("churn") >= 1, 1).
↪otherwise(0))
labeled_df.filter(F.col("userId") == "122").show(n=100)
```

```
+--------------------+---------+---------+------+-------------+---------+-------
--+-----+------------------+------+--------------------+-------------+---------+-
-------------------+------+-------------+--------------------+------+----------+
-----+
|               artist|     auth|firstName|gender|itemInSession| lastName|
length|level|          location|method|                page|
```

```
registration|sessionId|                 song|status|           ts|
userAgent|userId|churnEvent|churn|
+--------------------+---------+---------+------+-------------+--------+-------
--+-----+-----------------+------+--------------------+-------------+---------+-
-------------------+------+-------------+--------------------+------+----------+
-----+
|             unknown|Logged In|    Molly|     F|            1|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|                Home|1535498705000|      403|
unknown|   200|1538657235000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            2|Patterson|
0.0| free|Memphis, TN-MS-AR|   PUT|              Logout|1535498705000|      403|
unknown|   307|1538657236000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            3|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|                Home|1535498705000|      611|
unknown|   200|1539173538000|Mozilla/5.0 (X11;…|   122|         0|    1|
|         Danger Doom|Logged In|    Molly|     F|
4|Patterson|160.36526| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      611|Old School Rules …|
200|1539173664000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Florence + The Ma…|Logged In|    Molly|     F|            5|Patterson|
131.3171| free|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
611|    Kiss With A Fist|   200|1539173824000|Mozilla/5.0 (X11;…|   122|
0|    1|
|       Groove Armada|Logged In|    Molly|     F|
6|Patterson|309.68118| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      611|       Serve Chilled|
200|1539173955000|Mozilla/5.0 (X11;…|   122|         0|    1|
|The All-American …|Logged In|    Molly|     F|
7|Patterson|194.79465| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      611|Mona Lisa (When T…|
200|1539174264000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Metallica / Maria…|Logged In|    Molly|     F|
8|Patterson|279.11791| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      611|  The Memory Remains|
200|1539174458000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            9|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|         Roll Advert|1535498705000|      611|
unknown|   200|1539174569000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            0|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|                Home|1535498705000|      691|
unknown|   200|1539886900000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Atlanta Rhythm Se…|Logged In|    Molly|     F|
1|Patterson|217.80853| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      691|           Doraville|
200|1539886903000|Mozilla/5.0 (X11;…|   122|         0|    1|
|           Lady GaGa|Logged In|    Molly|     F|
2|Patterson|274.18077| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      691|           Alejandro|
```

```
200|1539887120000|Mozilla/5.0 (X11;…|   122|         0|    1|
|       Guns N' Roses|Logged In|    Molly|     F|            3|Patterson|
184.0322| free|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
691|    Live And Let Die|   200|1539887394000|Mozilla/5.0 (X11;…|   122|
0|    1|
|        Cyndi Lauper|Logged In|    Molly|     F|
4|Patterson|228.88444| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|      691|Girls Just Want T…|
200|1539887578000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            5|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|         Roll Advert|1535498705000|      691|
unknown|   200|1539887683000|Mozilla/5.0 (X11;…|   122|         0|    1|
|           Cat Power|Logged In|    Molly|     F|            6|Patterson|
142.8371| free|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
691|          Sea Of Love|   200|1539887806000|Mozilla/5.0 (X11;…|   122|
0|    1|
|   The Rockin Rebels|Logged In|    Molly|     F|            7|Patterson|
138.1873| free|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
691|         Wild Weekend|   200|1539887948000|Mozilla/5.0 (X11;…|   122|
0|    1|
|             unknown|Logged In|    Molly|     F|            8|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|                Home|1535498705000|      691|
unknown|   200|1539888010000|Mozilla/5.0 (X11;…|   122|         0|    1|
|   Cage The Elephant|Logged In|    Molly|     F|            0|Patterson|
228.0224| free|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
1029|Back Against The …|   200|1540055184000|Mozilla/5.0 (X11;…|   122|
0|    1|
|             unknown|Logged In|    Molly|     F|            1|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|                Home|1535498705000|     1029|
unknown|   200|1540055184000|Mozilla/5.0 (X11;…|   122|         0|    1|
|            De-Phazz|Logged In|    Molly|     F|
2|Patterson|220.99546| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|       Belle de Jour|
200|1540055412000|Mozilla/5.0 (X11;…|   122|         0|    1|
|   King Biscuit Time|Logged In|    Molly|     F|
3|Patterson|227.52608| free|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|      Time To Get Up|
200|1540055632000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            4|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|             Upgrade|1535498705000|     1029|
unknown|   200|1540055690000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            5|Patterson|
0.0| free|Memphis, TN-MS-AR|   GET|             Upgrade|1535498705000|     1029|
unknown|   200|1540055771000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            6|Patterson|
0.0| free|Memphis, TN-MS-AR|   PUT|      Submit Upgrade|1535498705000|     1029|
unknown|   307|1540055772000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            7|Patterson|
```

```
0.0| paid|Memphis, TN-MS-AR|   GET|               Home|1535498705000|     1029|
unknown|   200|1540055776000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|            8|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|         Add Friend|1535498705000|     1029|
unknown|   307|1540055777000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Afro-Cuban All Stars|Logged In|    Molly|     F|            9|Patterson|
288.1824| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|MarÃ Âa Caracole…|   200|1540055859000|Mozilla/5.0 (X11;…|   122|
0|    1|
|       Spiritualized|Logged In|    Molly|     F|
10|Patterson|344.99873| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|Why Don't You Smi…|
200|1540056147000|Mozilla/5.0 (X11;…|   122|         0|    1|
|            Cyberfit|Logged In|    Molly|     F|
11|Patterson|303.15057| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|           Pojo Pojo|
200|1540056491000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Alison Krauss / U…|Logged In|    Molly|     F|
12|Patterson|171.04934| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|            Restless|
200|1540056794000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             Cartola|Logged In|    Molly|     F|           13|Patterson|
127.242| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|            Tive Sim|   200|1540056965000|Mozilla/5.0 (X11;…|   122|
0|    1|
|             unknown|Logged In|    Molly|     F|           14|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|          Thumbs Up|1535498705000|     1029|
unknown|   307|1540056966000|Mozilla/5.0 (X11;…|   122|         0|    1|
|Thao with The Get…|Logged In|    Molly|     F|
15|Patterson|193.74975| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|           Geography|
200|1540057092000|Mozilla/5.0 (X11;…|   122|         0|    1|
|          Kanye West|Logged In|    Molly|     F|
16|Patterson|278.07302| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|    Through The Wire|
200|1540057285000|Mozilla/5.0 (X11;…|   122|         0|    1|
|The Lonely Island…|Logged In|    Molly|     F|           17|Patterson|
192.9922| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|             Boombox|   200|1540057563000|Mozilla/5.0 (X11;…|   122|
0|    1|
|Soulja Boy Tell'e…|Logged In|    Molly|     F|           18|Patterson|
194.2722| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|Kiss Me Thru The …|   200|1540057755000|Mozilla/5.0 (X11;…|   122|
0|    1|
|      More Fire Crew|Logged In|    Molly|     F|           19|Patterson|
353.4624| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|         Never Trust|   200|1540057949000|Mozilla/5.0 (X11;…|   122|
0|    1|
```

```
|         Donna Lewis|Logged In|    Molly|     F|
20|Patterson|240.95302| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|I Love You Always…|
200|1540058302000|Mozilla/5.0 (X11;…|   122|         0|    1|
|  Renegade Soundwave|Logged In|    Molly|     F|
21|Patterson|224.67873| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|  Pocket Porn (1990)|
200|1540058542000|Mozilla/5.0 (X11;…|   122|         0|    1|
|         Cat Stevens|Logged In|    Molly|     F|
22|Patterson|225.17506| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|           Sad Lisa|
200|1540058766000|Mozilla/5.0 (X11;…|   122|         0|    1|
|            La Renga|Logged In|    Molly|     F|
23|Patterson|307.35628| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|Triste CanciÃ Â³n…|
200|1540058991000|Mozilla/5.0 (X11;…|   122|         0|    1|
|              Eagles|Logged In|    Molly|     F|
24|Patterson|241.78893| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|Take The Devil (L…|
200|1540059298000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|          25|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|         Thumbs Down|1535498705000|     1029|
unknown|   307|1540059299000|Mozilla/5.0 (X11;…|   122|         0|    1|
|        Modest Mouse|Logged In|    Molly|     F|
26|Patterson|246.17751| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|          Dashboard|
200|1540059539000|Mozilla/5.0 (X11;…|   122|         0|    1|
|   59 Times the Pain|Logged In|    Molly|     F|
27|Patterson|144.95302| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|         Found Home|
200|1540059785000|Mozilla/5.0 (X11;…|   122|         0|    1|
|         Suzi Quatro|Logged In|    Molly|     F|
28|Patterson|239.72526| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|      Daytona Demon|
200|1540059929000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|          29|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|           Thumbs Up|1535498705000|     1029|
unknown|   307|1540059930000|Mozilla/5.0 (X11;…|   122|         0|    1|
|            No Doubt|Logged In|    Molly|     F|
30|Patterson|241.81506| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|            Running|
200|1540060168000|Mozilla/5.0 (X11;…|   122|         0|    1|
|        Lisac Josipa|Logged In|    Molly|     F|          31|Patterson|
271.882| paid|Memphis, TN-MS-AR|   PUT|            NextSong|1535498705000|
1029|       Le~aj od suza|   200|1540060409000|Mozilla/5.0 (X11;…|   122|
0|    1|
|         Kim Burrell|Logged In|    Molly|     F|
32|Patterson|249.67791| paid|Memphis, TN-MS-AR|   PUT|
```

```
NextSong|1535498705000|     1029|        Just as I Am|
200|1540060680000|Mozilla/5.0 (X11;…|   122|         0|    1|
|              Xzibit|Logged In|    Molly|     F|
33|Patterson|262.29506| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|       Chamber Music|
200|1540060929000|Mozilla/5.0 (X11;…|   122|         0|    1|
|        Brad Paisley|Logged In|    Molly|     F|
34|Patterson|266.91873| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|    She's Everything|
200|1540061191000|Mozilla/5.0 (X11;…|   122|         0|    1|
|        New Radicals|Logged In|    Molly|     F|
35|Patterson|219.19302| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|  Someday We'll Know|
200|1540061457000|Mozilla/5.0 (X11;…|   122|         0|    1|
|               3OH!3|Logged In|    Molly|     F|           36|Patterson|
192.522| paid|Memphis, TN-MS-AR|   PUT|           NextSong|1535498705000|
1029|My First Kiss (Fe…|   200|1540061676000|Mozilla/5.0 (X11;…|   122|
0|    1|
|      Robert Johnson|Logged In|    Molly|     F|
37|Patterson|154.09587| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|I?'m A Steady Rol…|
200|1540061868000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|           38|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|          Thumbs Up|1535498705000|     1029|
unknown|   307|1540061869000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|           39|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|     Add to Playlist|1535498705000|     1029|
unknown|   200|1540061987000|Mozilla/5.0 (X11;…|   122|         0|    1|
|         Bob Newhart|Logged In|    Molly|     F|
40|Patterson|367.98649| paid|Memphis, TN-MS-AR|   PUT|
NextSong|1535498705000|     1029|Introducing Tobac…|
200|1540062022000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|           41|Patterson|
0.0| paid|Memphis, TN-MS-AR|   GET|           Settings|1535498705000|     1029|
unknown|   200|1540062049000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Logged In|    Molly|     F|           42|Patterson|
0.0| paid|Memphis, TN-MS-AR|   PUT|             Cancel|1535498705000|     1029|
unknown|   307|1540062050000|Mozilla/5.0 (X11;…|   122|         0|    1|
|             unknown|Cancelled|    Molly|     F|           43|Patterson|
0.0| paid|Memphis, TN-MS-AR|   GET|Cancellation Conf…|1535498705000|     1029|
unknown|   200|1540062068000|Mozilla/5.0 (X11;…|   122|         1|    1|
+--------------------+---------+---------+------+-------------+---------+-------
--+-----+------------------+------+--------------------+-------------+---------+-
-------------------+------+-------------+--------------------+------+----------+
-----+
```

6.0.2 Explore Data

[24]:
```
palette = sns.color_palette("Set2")
sns.set_palette(palette)
```

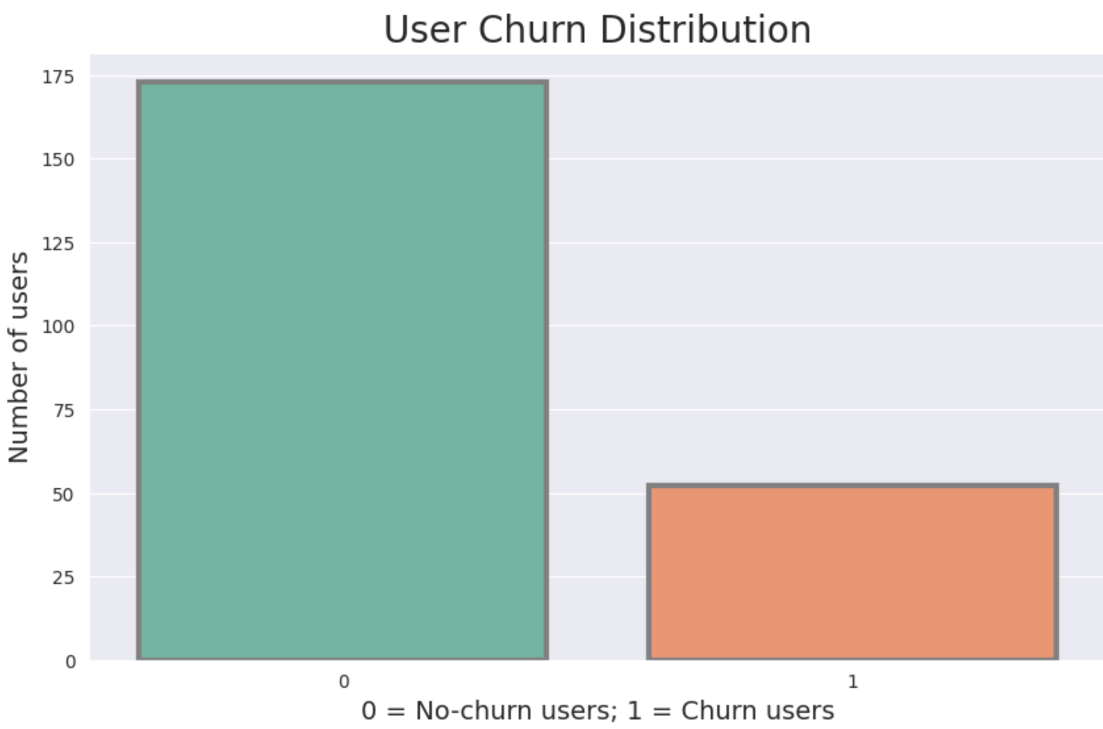
Sample the Dataset for EDA This is useful for when the full dataset is used.

[25]:
```
mini_df_count = 278154
# Keep the number of records from the mini dataframe as a point of reference.␣
↪If we use only the mini dataframe, then all the records will be used.
sampling_fraction = 278154 / labeled_df.count()
print(f"Sampling fraction for EDA: {sampling_fraction * 100:.2f}%")
eda_df = labeled_df.sample(withReplacement=False, fraction=sampling_fraction,␣
↪seed=42)
eda_df.show(n=1)
```

```
Sampling fraction for EDA: 100.00%
+--------------------+---------+---------+------+-------------+---------+-------
--+-----+--------------------+------+--------+-------------+---------+----------
----------+------+-------------+--------------------+------+----------+-----+
|              artist|     auth|firstName|gender|itemInSession| lastName|
length|level|            location|method|    page| registration|sessionId|
song|status|           ts|           userAgent|userId|churnEvent|churn|
+--------------------+---------+---------+------+-------------+---------+-------
--+-----+--------------------+------+--------+-------------+---------+----------
----------+------+-------------+--------------------+------+----------+-----+
|Sleeping With Sirens|Logged In| Darianna|     F|
0|Carpenter|202.97098| free|Bridgeport-Stamfo…|   PUT|NextSong|1538016340000|
31|Captain Tyin Knot…|   200|1539003534000|"Mozilla/5.0 (iPh…|100010|
0|    0|
+--------------------+---------+---------+------+-------------+---------+-------
--+-----+--------------------+------+--------+-------------+---------+----------
----------+------+-------------+--------------------+------+----------+-----+
only showing top 1 row
```

Users Churn Distribution

[26]:
```
user_churn_distribution = eda_df.select(["userId", "churn"]).distinct().
↪groupby("churn").count().toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=user_churn_distribution, x="churn", y="count", errorbar=("pi",␣
↪50), capsize=.4, errcolor=".5", linewidth=3, edgecolor=".5", ax=ax)
ax.set_title("User Churn Distribution", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Number of users", fontsize=14)
fig.savefig("images/user_churn_distribution.jpg");
```



The distribution is highly skewed towards the `no-churn` users. That is why we will use the `F1` score to find the best model in the cross validation step.

Possible Pages

[27]:
```
eda_df.select("page").distinct().show()
```

```
+--------------------+
|                page|
+--------------------+
|              Cancel|
|    Submit Downgrade|
|         Thumbs Down|
|                Home|
|           Downgrade|
|         Roll Advert|
|              Logout|
|       Save Settings|
|Cancellation Conf…|
|               About|
|            Settings|
|     Add to Playlist|
|          Add Friend|
|            NextSong|
|           Thumbs Up|
```

```
|                Help|
|             Upgrade|
|               Error|
|      Submit Upgrade|
+--------------------+
```

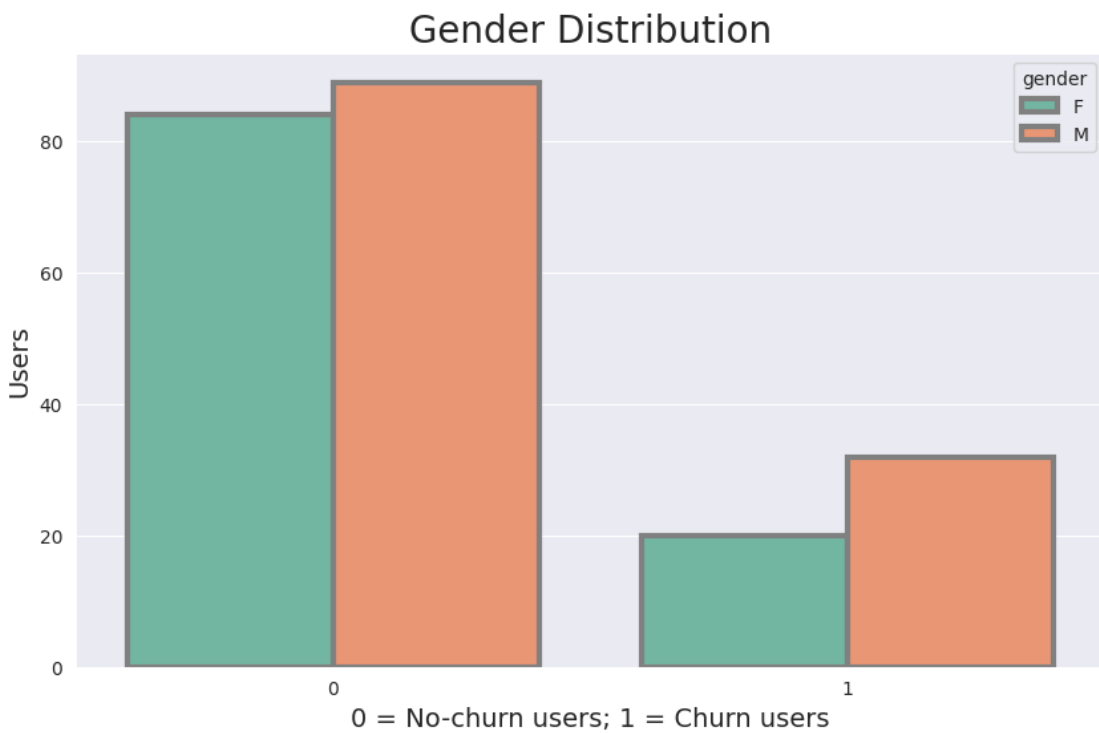
[28]:
```
eda_df.filter(F.col("artist") != "unknown").select("page").distinct().show()
```

```
+--------+
|    page|
+--------+
|NextSong|
+--------+
```

The only page where music is played is called `NextSong`.

Gender Distribution

[29]:
```
gender_distribution = eda_df.select(["churn", "userId", "gender"]).distinct().
 ↪groupby(["churn", "gender"]).count().toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=gender_distribution, x="churn", y="count", hue="gender",␣
 ↪errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3, edgecolor=".5",␣
 ↪ax=ax)
ax.set_title("Gender Distribution", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Users", fontsize=14)
fig.savefig("images/gender_distribution.jpg");
```



Visually it seems that more males are in the churn group than females. But, we won't use this as a feature because the difference is not that big (to be 100%, we should statistically check the difference between the two distributions), and we don't want to bias the model towards gender.

Time Delta Since Registration

[30]:
```
registration_delta_distribution = eda_df.
↪alias("registration_delta_distribution")
registration_delta_distribution = registration_delta_distribution.
↪withColumn("timeSinceRegistration", F.col("ts") - F.col("registration"))
registration_delta_distribution = registration_delta_distribution \
    .select(["churn", "userId", "timeSinceRegistration"]) \
    .groupby(["churn", "userId"]) \
    .agg(F.max(F.col("timeSinceRegistration")).alias("timeSinceRegistration")) \
    .toPandas()
registration_delta_distribution
```

[30]:

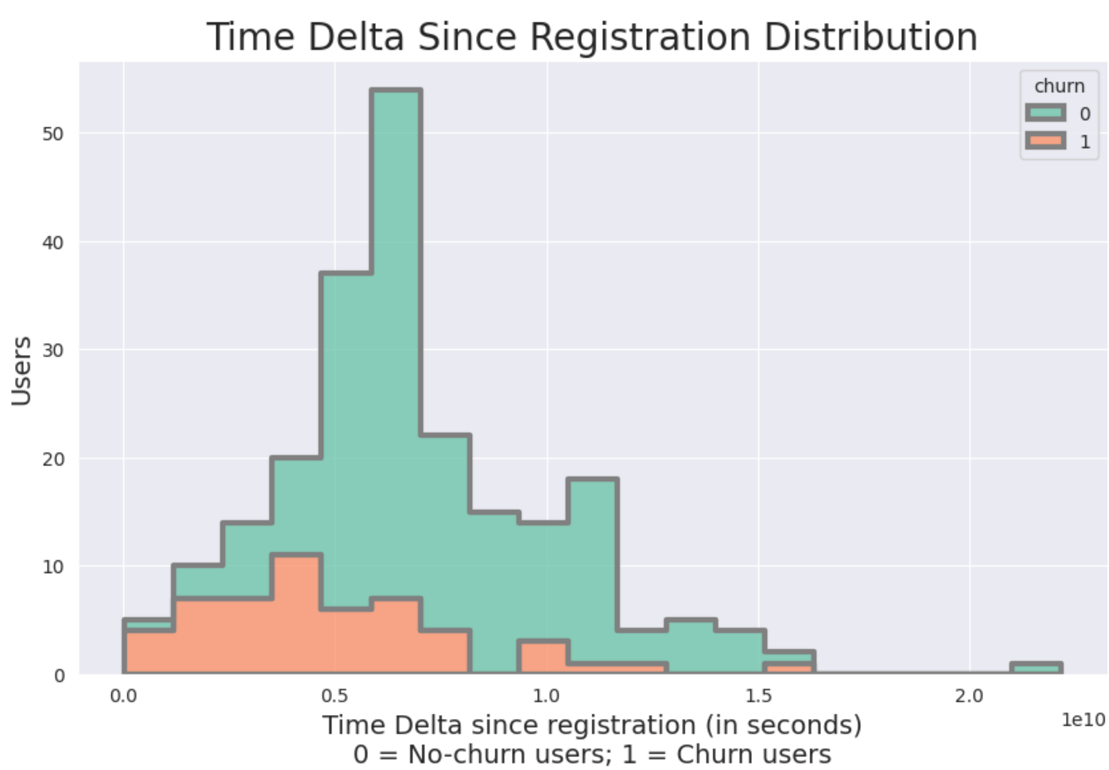
	churn	userId	timeSinceRegistration
0	0	100	5605094000
1	0	100004	14898823000
2	1	100005	7360899000
3	1	100006	788587000
4	1	100007	9969490000
..	…	…	…

```
220        0       94              11431432000
221        0       95               5379812000
222        0       97               7550792000
223        0      115               6431581000
224        0   200025              10108410000

[225 rows x 3 columns]
```

[53]:

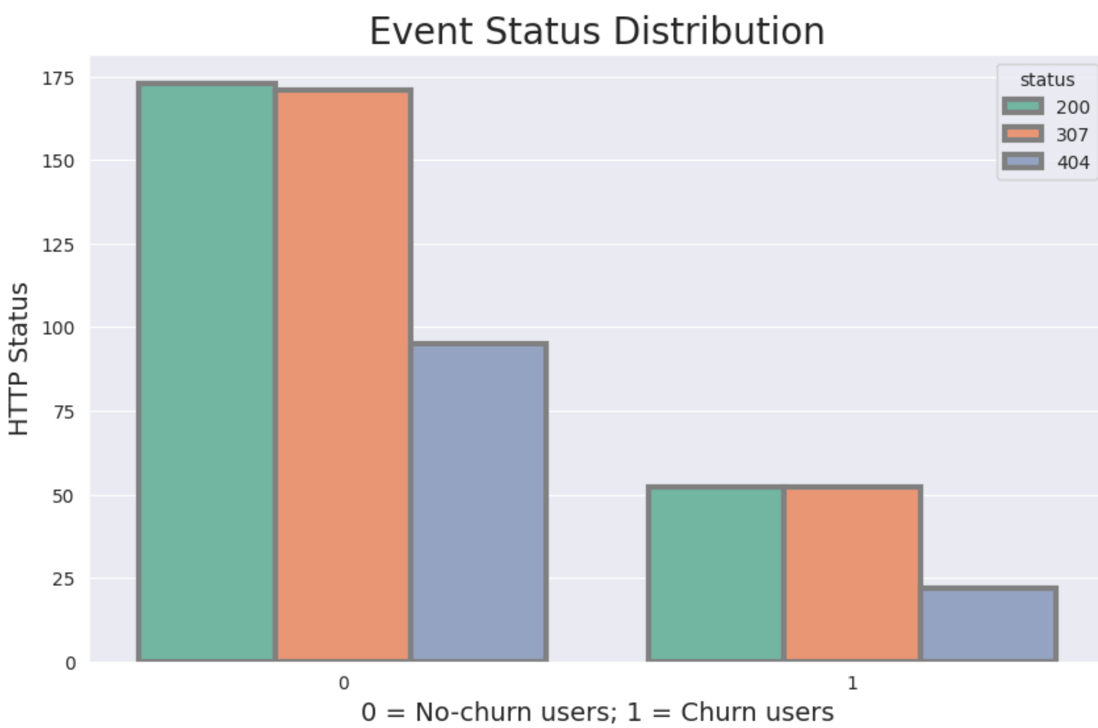
```
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.histplot(
    data=registration_delta_distribution,
    x="timeSinceRegistration",
    hue="churn",
    multiple="stack",
    linewidth=3,
    edgecolor=".5",
    element="step",
    ax=ax,
)
ax.set_title("Time Delta Since Registration Distribution", fontsize=20)
ax.set_xlabel("Time Delta since registration (in seconds)\n0 = No-churn users;␣
 ↪1 = Churn users", fontsize=14)
ax.set_ylabel("Users", fontsize=14)
fig.savefig("images/registration_delta_distribution.jpg");
```



It looks like the timedelta since registration, for churn users, is right skewed. While the no-churn one is pretty normal. The mean of the timedelta is a good predictor.

Event Status Distribution

[32]:
```
event_status_distribution = eda_df.select(["churn", "userId", "status"]).
↪distinct().groupby(["churn", "status"]).count().toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=event_status_distribution, x="churn", y="count", hue="status",␣
↪errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3, edgecolor=".5",␣
↪ax=ax)
ax.set_title("Event Status Distribution", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("HTTP Status", fontsize=14)
fig.savefig("images/event_status_distribution.jpg");
```

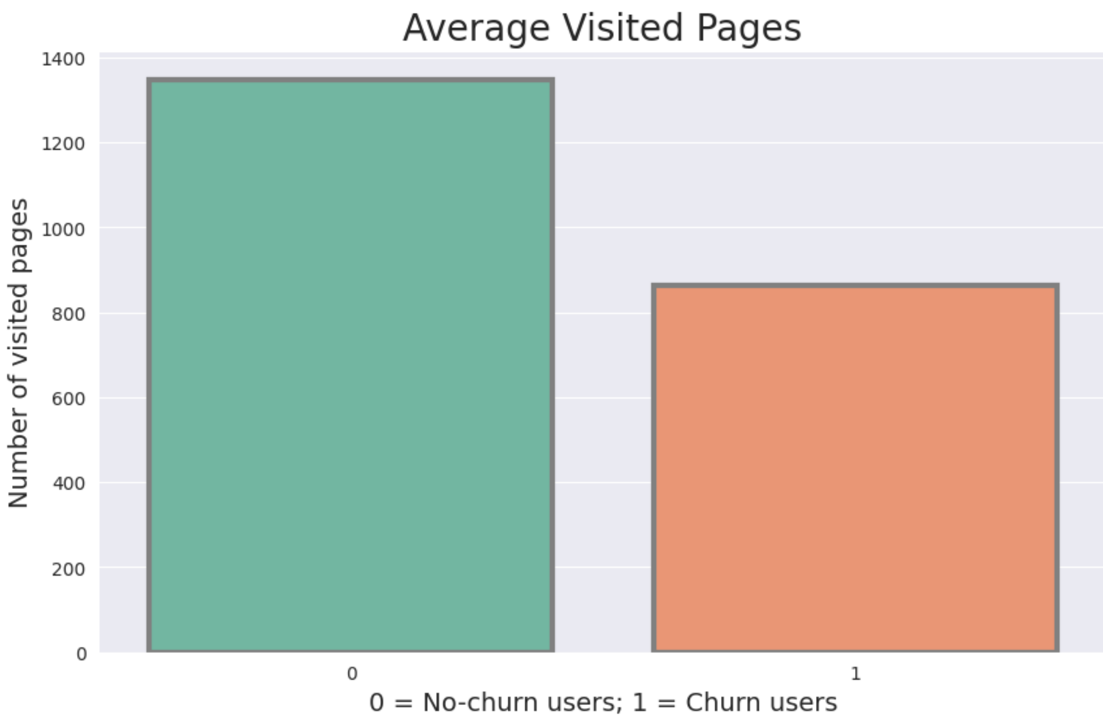


The only status type that could mess with the user experience is `404`. But as we can see, it is evenly distributed between the `churn` and `no-churn` groups.

NOTE: A more robust verification can be performed with a proportion statistical test.

Distribution of the Average Number of Visited Pages by Every User

[33]:
```
visited_pages_distribution = eda_df.groupby(["churn", "userId"]).count().
 ↪groupby("churn").agg(F.avg("count").alias("Average Visited Pages")).
 ↪toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=visited_pages_distribution, x="churn", y="Average Visited␣
 ↪Pages", errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3,␣
 ↪edgecolor=".5", ax=ax)
ax.set_title("Average Visited Pages", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Number of visited pages", fontsize=14)
fig.savefig("images/visited_pages_distribution.jpg");
```

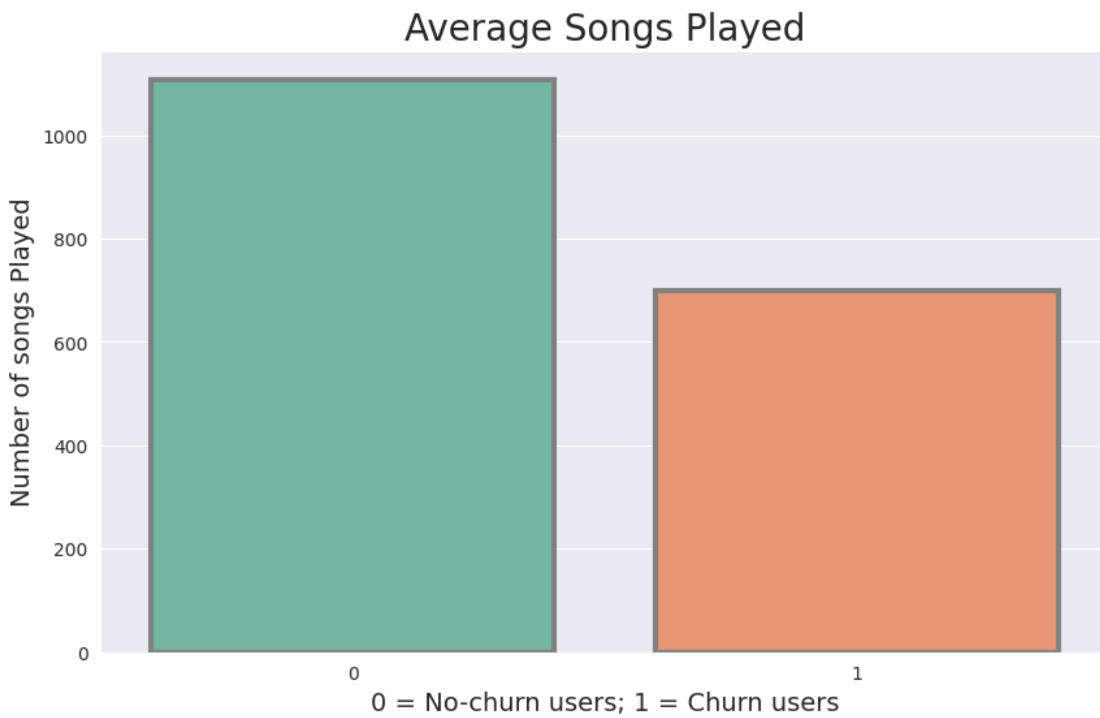


Users that remained on the platform, on average, visited more pages.

Distribution of the Average Number of Songs Listened by Every User

[34]:
```
listened_songs_distribution = eda_df.where(F.col("artist") != "unknown").
 ↪groupby(["churn", "userId"]).count().groupby("churn").agg(F.avg("count").
 ↪alias("Average Songs Played")).toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=listened_songs_distribution, x="churn", y="Average Songs␣
 ↪Played", errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3,␣
 ↪edgecolor=".5", ax=ax)
ax.set_title("Average Songs Played", fontsize=20)
```

```
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Number of songs Played", fontsize=14)
fig.savefig("images/listened_songs_distribution.jpg");
```

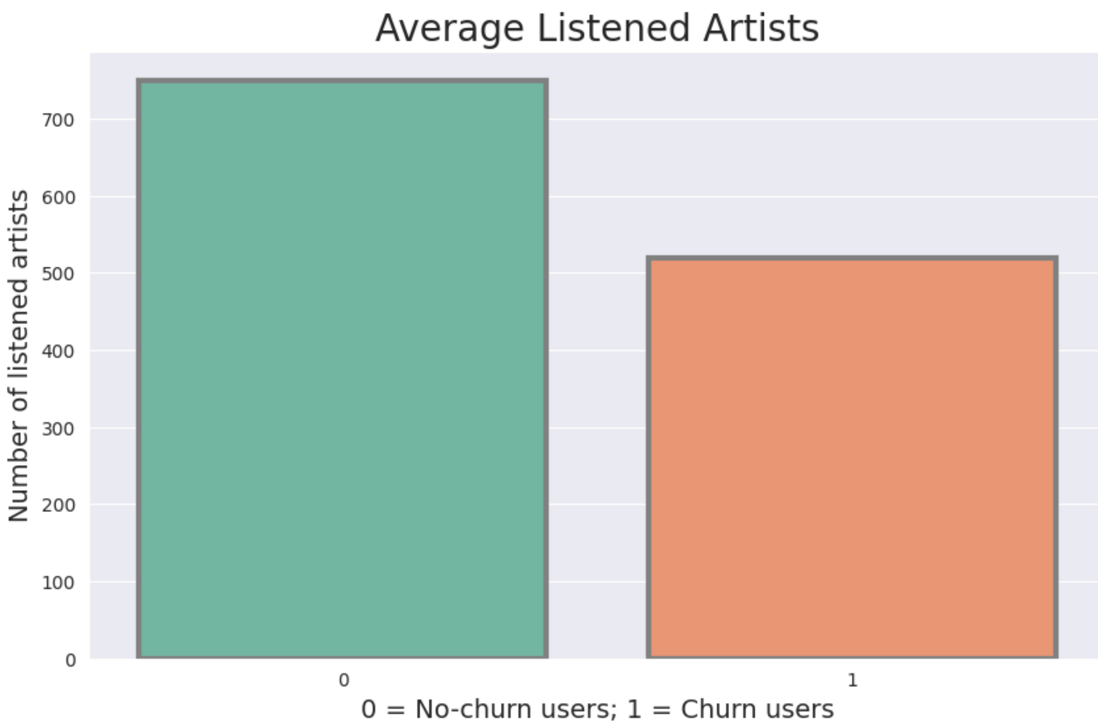


Users that remained on the platform, on average, are listening to more songs.

Distribution of the Average Number of Artists Listened by Every User

[35]:
```
listened_artists_distribution = eda_df \
    .select(["artist", "userId", "churn"]) \
    .where(F.col("artist") != "unknown") \
    .distinct() \
    .groupby(["churn", "userId"]) \
    .count() \
    .groupby("churn") \
    .agg(F.avg("count").alias("Average Listened Artists")) \
    .toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=listened_artists_distribution, x="churn", y="Average Listened␣
 ↪Artists", errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3,␣
 ↪edgecolor=".5", ax=ax)
ax.set_title("Average Listened Artists", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Number of listened artists", fontsize=14)
```

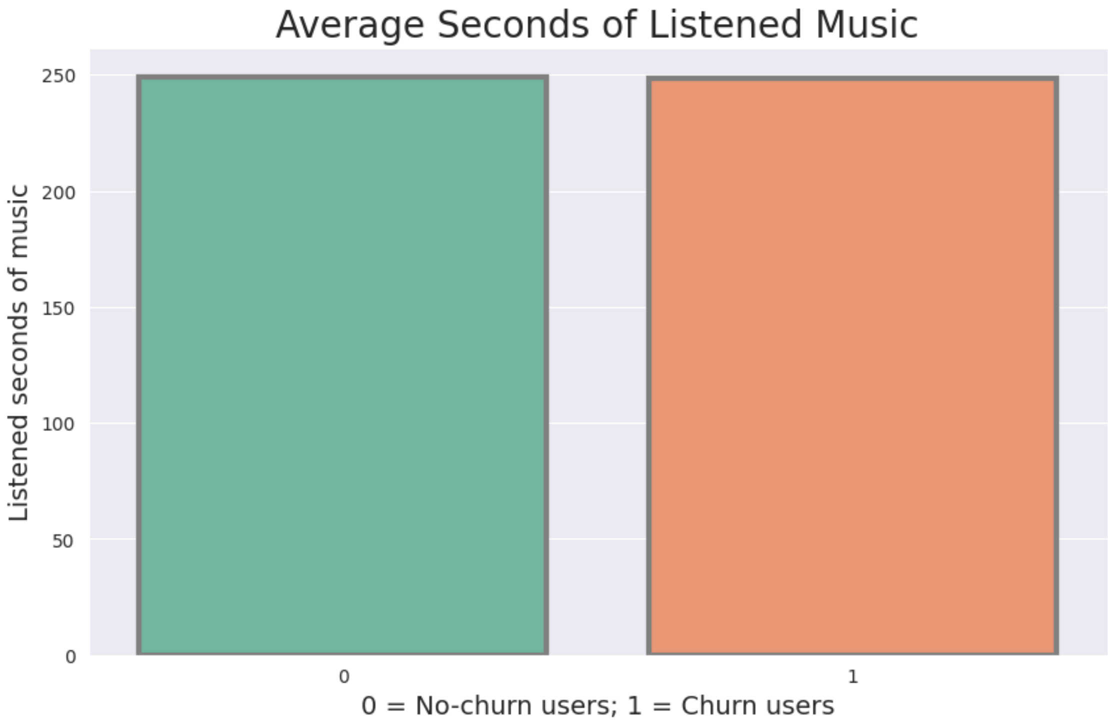
```
fig.savefig("images/listened_artists_distribution.jpg");
```



On average, users that remain on the platform are listening to a wider variety of artists.

Average Seconds of Listened Music

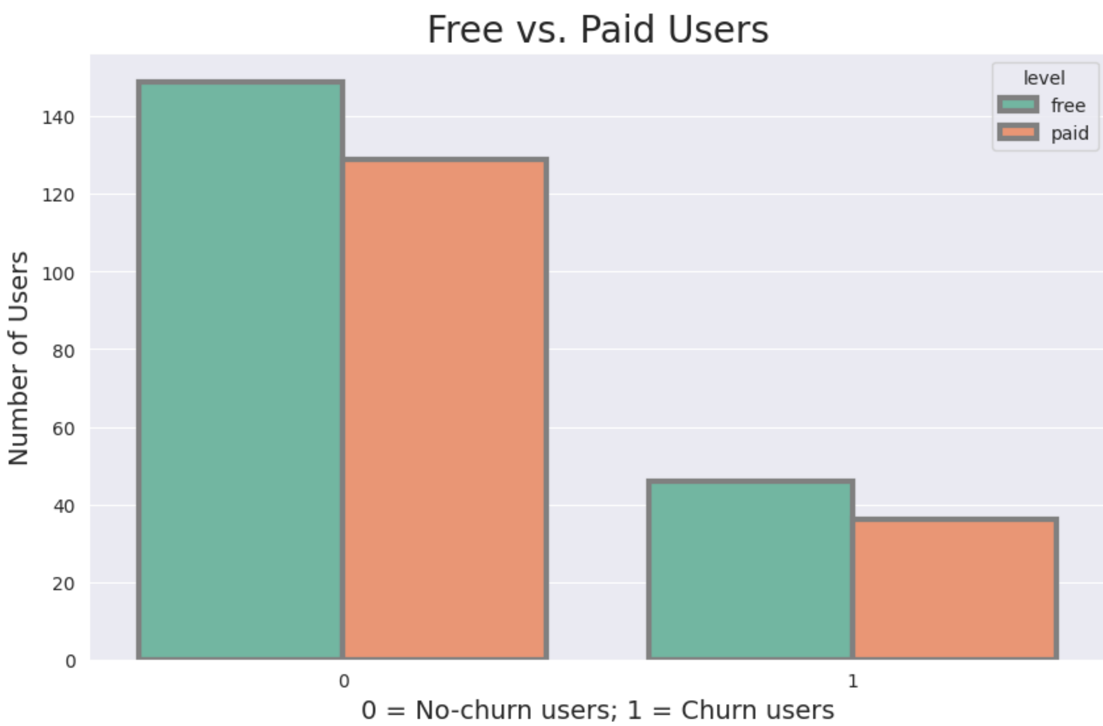
[36]:
```
average_listened_seconds_distribution = eda_df \
    .select(["artist", "userId", "churn", "length"]) \
    .where(F.col("artist") != "unknown") \
    .groupby(["churn", "userId"]) \
    .agg(F.avg(F.col("length")).alias("averageLength")) \
    .groupby("churn") \
    .agg(F.avg(F.col("averageLength")).alias("Average Seconds of Listened␣
↪Music")) \
    .toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=average_listened_seconds_distribution, x="churn", y="Average␣
↪Seconds of Listened Music", errorbar=("pi", 50), capsize=.4, errcolor=".5",␣
↪linewidth=3, edgecolor=".5", ax=ax)
ax.set_title("Average Seconds of Listened Music", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Listened seconds of music", fontsize=14)
fig.savefig("images/average_listened_seconds_distribution.jpg");
```



The average seconds of listened music is similar for both churn and no-churn groups. Therefore, this feature won't have much predictive power.

Different Level of Users

[37]:
```
levels_distribution = eda_df \
    .select(["userId", "level", "churn"]) \
    .distinct() \
    .groupby("churn", "level") \
    .agg(F.count(F.col("userId")).alias("Levels of Users")) \
    .toPandas()
fig, ax = plt.subplots(1, 1, figsize=(10, 6))
sns.barplot(data=levels_distribution, x="churn", y="Levels of Users",␣
 ↪hue="level", errorbar=("pi", 50), capsize=.4, errcolor=".5", linewidth=3,␣
 ↪edgecolor=".5", ax=ax)
ax.set_title("Free vs. Paid Users", fontsize=20)
ax.set_xlabel("0 = No-churn users; 1 = Churn users", fontsize=14)
ax.set_ylabel("Number of Users", fontsize=14)
fig.savefig("images/levels_distribution.jpg");
```



[38]:
```
levels_distribution
```

[38]:
```
   churn level  Levels of Users
0      0  free              149
1      0  paid              129
2      1  paid               36
3      1  free               46
```

[39]:
```
no_churn_users = levels_distribution[levels_distribution["churn"] == 0]
paid_users_proportion = no_churn_users.iloc[1]["Levels of Users"] /␣
 ↪no_churn_users["Levels of Users"].sum()
print(f"Paid users proportion for no-churn users: {paid_users_proportion*100:.
 ↪2f}%")
```

```
Paid users proportion for no-churn users: 46.40%
```

[40]:
```
no_churn_users = levels_distribution[levels_distribution["churn"] == 1]
paid_users_proportion = no_churn_users.iloc[0]["Levels of Users"] /␣
 ↪no_churn_users["Levels of Users"].sum()
print(f"Paid users proportion for churn users: {paid_users_proportion*100:.
 ↪2f}%")
```

```
Paid users proportion for churn users: 43.90%
```

There is no huge difference for the users that are paying for a subscription between the churn and no-churn groups.

Note: A more robust verification can be done with a proportion statistical test.

7 Feature Engineering

Aggregate Data at the User Level The churn classification will be performed at the user level. Therefore, we need to aggregate the data for every user. All the features will be a result of this aggregation.

[41]:
```
def count_with_condition(condition):
    """Utility function to count only specific rows based on the 'condition'."""
    return F.count(F.when(condition, True))


def count_distinct_with_condition(condition, values):
    """Utility function to count only distinct & specific rows based on the␣
 ↪'condition'."""
    return F.count_distinct(F.when(condition, values))
```

Based on the EDA step, we chose the features that divide the most the `churn` and `no-churn` user groups: - The total number of visited pages. - The total number of songs played. - The total number of total artists. - Timestamp since registration (in seconds)

[58]:
```
aggregated_df = labeled_df.groupby("userId").agg(
    F.count("page").alias("numPagesVisited"),
    count_with_condition(F.col("page") == "NextSong").alias("numTotalPlays"),
    count_distinct_with_condition(F.col("artist") != "unknown", F.
 ↪col("artist")).alias("numTotalArtists"),
    F.max(F.col("ts") - F.col("registration")).
 ↪alias("timedeltaSinceRegistration"),
    F.max("churn").alias("churn")
)
aggregated_df.show(n=5)
```

```
+------+---------------+-------------+---------------+--------------------------
+------+
|userId|numPagesVisited|numTotalPlays|numTotalArtists|timedeltaSinceRegistration
|churn|
+------+---------------+-------------+---------------+--------------------------
+------+
|100010|            381|          275|            252|
4807612000|     0|
|100014|            310|          257|            233|
7351206000|     1|
|100021|            319|          230|            207|
5593438000|     1|
```

```
|   101|           2149|          1797|            1241|
4662657000|    1|
|    11|            848|           647|             534|
10754921000|    0|
+------+---------------+-------------+---------------+-------------------------
+------+
only showing top 5 rows
```

Create the Feature Vector and Labels

[59]:
```python
assembler = VectorAssembler(inputCols=["numPagesVisited", "numTotalPlays", "timedeltaSinceRegistration", "numTotalArtists"], outputCol="unscaled_features")
engineered_df = assembler.transform(aggregated_df)
engineered_df = engineered_df.select(F.col("unscaled_features"), F.col("churn").alias("label"))
engineered_df.show()
```

```
+--------------------+-----+
|   unscaled_features|label|
+--------------------+-----+
|[381.0,275.0,4.80...|    0|
|[310.0,257.0,7.35...|    1|
|[319.0,230.0,5.59...|    1|
|[2149.0,1797.0,4...|    1|
|[848.0,647.0,1.07...|    0|
|[4825.0,4079.0,1...|    0|
|[11.0,8.0,6.16177...|    1|
|[2304.0,1928.0,5...|    0|
|[2469.0,2070.0,5...|    0|
|[2278.0,1914.0,4...|    0|
|[118.0,84.0,2.062...|    0|
|[1002.0,820.0,2.0...|    0|
|[474.0,387.0,6.05...|    0|
|[3603.0,3028.0,5...|    1|
|[4428.0,3632.0,6...|    0|
|[2464.0,2111.0,1...|    1|
|[3437.0,2841.0,9...|    1|
|[1342.0,1125.0,6...|    0|
|[201.0,150.0,6.28...|    0|
|[964.0,681.0,4.14...|    1|
+--------------------+-----+
only showing top 20 rows
```

8 Modeling

We will train and test three models: * Logistic Regression * Naive Bayes * Gradient Boosting Tree

We will use cross-validation with 3 folds to find the best hyper-parameters.

Because the labels are highly imbalanced, we will use the `F1 score` to evaluate the models. The F1 score metric is using under the hood the precision and recall which are taking into consideration the unbalanced distribution issue.

[44]:
```
def run(pipeline, paramGrid, train_df, test_df):
    """
    Main function used to train & test a given model.
    The training step uses cross-validation to find the best hyper-parameters␣
 ↪for the model.

    :param pipeline: Model pipeline.
    :param paramGrid: Parameter grid used for cross-validation.
    :param train_df: Training dataframe.
    :param test_df: Testing dataframe.
    :return: the best model from cross-validation.
    """

    fitted_model = fit_model(paramGrid, pipeline, train_df)
    evaluate_model(fitted_model, test_df)

    return fitted_model


def fit_model(paramGrid, pipeline, train_df):
    """
    Function that trains the model using cross-validation.
    Also, it prints the best validation results and hyper-parameters.

    :param paramGrid: Parameter grid used for cross-validation.
    :param pipeline: Model pipeline.
    :param train_df: Training dataframe.
    :return: the best model from cross-validation.
    """

    crossval = CrossValidator(
        estimator=pipeline,
        estimatorParamMaps=paramGrid,
        evaluator=MulticlassClassificationEvaluator(metricName="f1", beta=1.0),
        parallelism=3,
        numFolds=3
    )

    fitted_model = crossval.fit(train_df)
```

```
    print_best_validation_score(fitted_model)
    print_best_parameters(fitted_model)

    return fitted_model


def create_pipeline(model):
    """
    Create a pipeline based on a model.

    :param model: The end model that will be used for training.
    :return: the built pipeline.
    """

    scaler = StandardScaler(inputCol="unscaled_features", outputCol="features")
    pipeline = Pipeline(stages=[scaler, model])

    return pipeline


def print_best_validation_score(cross_validation_model):
    """Prints the best validation score based on the results from the␣
 ↪cross-validation model."""
    print()
    print("-" * 60)
    print(f"F1 score, on the validation split, for the best model: {np.
 ↪max(cross_validation_model.avgMetrics) * 100:.2f}%")
    print("-" * 60)


def print_best_parameters(cross_validation_model):
    """Prints the best hyper-parameters based on the results from the␣
 ↪cross-validation model."""

    parameters = cross_validation_model.getEstimatorParamMaps()[np.
 ↪argmax(cross_validation_model.avgMetrics)]

    print()
    print("-" * 60)
    print("Best model hyper-parameters:")
    for param, value in parameters.items():
        print(f"{param}: {value}")
    print("-" * 60)


def evaluate_model(model, test_df):
```

```
    """Evaluate the model on the test set using F1 score and print the results.
 ↪"""

    predictions = model.transform(test_df)
    evaluator =  MulticlassClassificationEvaluator(metricName="f1", beta=1.0)
    metric = evaluator.evaluate(predictions)

    print()
    print("-" * 60)
    print(f"F1 score, on the test set is: {metric*100:.2f}%")
    print("-" * 60)

    return metric
```

8.1 Split the Data

[45]:
```
train_df, test_df = engineered_df.randomSplit([0.8, 0.2], seed=42)
```

8.2 Logistic Regression

[46]:
```
lr = LogisticRegression()
pipeline = create_pipeline(lr)

paramGrid = ParamGridBuilder() \
    .addGrid(lr.maxIter, [10, 25, 50])  \
    .addGrid(lr.regParam, [0.05, 0.1, 0.2]) \
    .addGrid(lr.elasticNetParam, [0.05, 0.1, 0.2]) \
    .build()
```

[47]:
```
run(pipeline, paramGrid, train_df.alias("train_df_lr"), test_df.
 ↪alias("test_df_lr"));
```

```
22/09/20 08:29:09 WARN BlockManager: Asked to remove block broadcast_2699, which
does not exist
22/09/20 08:29:13 WARN BlockManager: Asked to remove block
broadcast_2845_piece0, which does not exist


------------------------------------------------------------
F1 score, on the validation split, for the best model: 72.79%
------------------------------------------------------------

------------------------------------------------------------
Best model hyper-parameters:
LogisticRegression_c11f8516b4ea__maxIter: 10
LogisticRegression_c11f8516b4ea__regParam: 0.05
LogisticRegression_c11f8516b4ea__elasticNetParam: 0.05
```

```
------------------------------------------------------------

------------------------------------------------------------
F1 score, on the test set is: 59.52%
------------------------------------------------------------
```

8.3 Naive Bayes

[48]:
```python
nb = NaiveBayes()
pipeline = create_pipeline(nb)

paramGrid = ParamGridBuilder() \
    .addGrid(nb.smoothing, [0.5, 1, 2])  \
    .build()
```

[49]:
```python
run(pipeline, paramGrid, train_df.alias("train_df_nb"), test_df.
 ↪alias("test_df_nb"));
```

```
------------------------------------------------------------
F1 score, on the validation split, for the best model: 67.26%
------------------------------------------------------------

------------------------------------------------------------
Best model hyper-parameters:
NaiveBayes_4cf4924048e5__smoothing: 0.5
------------------------------------------------------------

------------------------------------------------------------
F1 score, on the test set is: 59.52%
------------------------------------------------------------
```

8.4 Gradient Boosting

[50]:
```python
gbt = GBTClassifier()
pipeline = create_pipeline(gbt)

paramGrid = ParamGridBuilder() \
    .addGrid(gbt.maxIter, [10, 20, 30]) \
    .addGrid(gbt.stepSize, [0.05, 0.1]) \
    .build()
```

[51]:
```python
run(pipeline, paramGrid, train_df.alias("train_df_gbt"), test_df.
 ↪alias("test_df_gbt"));
```

```
]22/09/20 08:31:14 WARN BlockManager: Asked to remove block
```

```
broadcast_5847_piece0, which does not exist
22/09/20 08:31:17 WARN BlockManager: Asked to remove block
broadcast_5904_piece0, which does not exist
22/09/20 08:31:17 WARN BlockManager: Asked to remove block broadcast_5904, which
does not exist
22/09/20 08:33:20 WARN BlockManager: Asked to remove block
broadcast_8718_piece0, which does not exist
22/09/20 08:33:26 WARN BlockManager: Asked to remove block broadcast_8847, which
does not exist

------------------------------------------------------------
F1 score, on the validation split, for the best model: 76.92%
------------------------------------------------------------

------------------------------------------------------------
Best model hyper-parameters:
GBTClassifier_4d7710265fa4__maxIter: 30
GBTClassifier_4d7710265fa4__stepSize: 0.1
------------------------------------------------------------

------------------------------------------------------------
F1 score, on the test set is: 81.78%
------------------------------------------------------------
```

9 Conclusion

After the cleaning & feature engineering steps, we trained three models: * Logistic Regression * Naive Bayes * Gradient Boosting

The `LR` model is a good baseline for classification. `GBT` is a gradient tree-based model which usually performs better on complex data with less feature engineering.

The `GBT` model performed better than the Logistic Regression and the Naive Bayes. Probably, because it is a more complex model that can understand non-linear relationships better. It has an `F1 score` of `84.73%` on the test split, based on the limited number of features we used, it is a good start. We could do better, but it is a good start to see that the model can pick up patterns within the data.

To further improve the model, we can do the following: * add more features * solve the label imbalance issue * use the `Downgrade` event to generate more `churn` labels * more hyper-parameter tuning (because some intervals used in cross-validation are hitting the lower or upper edges).