Sparkify

August 23, 2019

1 Sparkify Project Workspace

This workspace contains a tiny subset (128MB) of the full dataset available (12GB). Feel free to use this workspace to build your project, or to explore a smaller subset with Spark before deploying your cluster on the cloud. Instructions for setting up your Spark cluster is included in the last lesson of the Extracurricular Spark Course content.

You can follow the steps below to guide your data analysis and model building portion of this project.

In [145]:
```
# import libraries
from pyspark.sql import SparkSession, Window
from pyspark.sql.functions import count, col, when, isnan, udf, countDistinct, avg, wh
from pyspark.sql.functions import max as sparkMax
from pyspark.sql.types import IntegerType, DoubleType
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from pyspark.ml import Pipeline
from pyspark.ml.feature import MinMaxScaler, VectorAssembler, Normalizer, StandardScal
from pyspark.ml.classification import LogisticRegression, RandomForestClassifier, GBTC
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificat
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

In [146]:
```
# create a Spark session
spark = SparkSession.builder \
    .master("local") \
    .appName("Sparkify") \
    .getOrCreate()
```

In [147]:
```
spark.conf.set('spark.executor.memory', '32gb')
spark.conf.set('spark.driver.memory','32gb')
```

2 Load and Clean Dataset

In this workspace, the mini-dataset file is `mini_sparkify_event_data.json`. Load and clean the dataset, checking for invalid or missing data - for example, records without userids or sessionids.

```
In [148]: sparkify_data = 'mini_sparkify_event_data.json'
          df = spark.read.json(sparkify_data)

In [149]: df.head(2)

Out[149]: [Row(artist='Martha Tilston', auth='Logged In', firstName='Colin', gender='M', itemInS
           Row(artist='Five Iron Frenzy', auth='Logged In', firstName='Micah', gender='M', itemI

In [150]: df.persist()

Out[150]: DataFrame[artist: string, auth: string, firstName: string, gender: string, itemInSessi

In [151]: 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)
```

First some columns that are not useful for the model can be dropped: firstName and lastName can be dropped since the userId uniquely identifies each user. Also, the artist and song columns can be dropped since they have no effect on user churn, since it is the pick of the user.

```
In [152]: df.select([count(when(col(c)=="", c)).alias(c) for c in df.columns]).show()

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

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

```
In [153]: df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns]).s
```

```
+------+----+---------+------+-------------+--------+------+-----+--------+------+----+---------
|artist|auth|firstName|gender|itemInSession|lastName|length|level|location|method|page|registrat
+------+----+---------+------+-------------+--------+------+-----+--------+------+----+---------
| 58392|   0|     8346|  8346|            0|    8346| 58392|    0|    8346|     0|   0|        8
+------+----+---------+------+-------------+--------+------+-----+--------+------+----+---------
```

```
In [154]: #drop rows with null values
          df_clean = df.dropna(how='any',subset=['userId','sessionId'])
          df_clean = df_clean.filter(df["userId"] != "")

In [155]: df_clean.count()

Out[155]: 278154

In [156]: df.count()

Out[156]: 286500
```

3 Exploratory Data Analysis

When you're working with the full dataset, perform EDA by loading a small subset of the data and doing basic manipulations within Spark. In this workspace, you are already provided a small subset of data you can explore.

3.0.1 Define Churn

Once you've done some preliminary analysis, create a column `Churn` to use as the label for your model. I suggest using the `Cancellation Confirmation` events to define your churn, which happen for both paid and free users. As a bonus task, you can also look into the `Downgrade` events.

3.0.2 Explore Data

Once you've defined churn, perform some exploratory data analysis to observe the behavior for users who stayed vs users who churned. You can start by exploring aggregates on these two groups of users, observing how much of a specific action they experienced per a certain time unit or number of songs played.

```
In [157]: df_clean.select('auth').dropDuplicates().collect()

Out[157]: [Row(auth='Cancelled'), Row(auth='Logged In')]
```

```
In [158]: df_clean.select('gender').dropDuplicates().collect()

Out[158]: [Row(gender='F'), Row(gender='M')]

In [159]: df_clean.agg({'itemInSession':'max'}).collect()

Out[159]: [Row(max(itemInSession)=1321)]

In [160]: df_clean.agg({'length':'max'}).collect()

Out[160]: [Row(max(length)=3024.66567)]

In [161]: df_clean.select('level').dropDuplicates().collect()

Out[161]: [Row(level='free'), Row(level='paid')]

In [162]: df_clean.select('location').dropDuplicates().show(10)

+--------------------+
|            location|
+--------------------+
|     Gainesville, FL|
|Atlantic City-Ham...|
|Deltona-Daytona B...|
|San Diego-Carlsba...|
|Cleveland-Elyria, OH|
|Kingsport-Bristol...|
|New Haven-Milford...|
|Birmingham-Hoover...|
|  Corpus Christi, TX|
|         Dubuque, IA|
+--------------------+
only showing top 10 rows



In [163]: df_clean.select('page').dropDuplicates().collect()

Out[163]: [Row(page='Cancel'),
           Row(page='Submit Downgrade'),
           Row(page='Thumbs Down'),
           Row(page='Home'),
           Row(page='Downgrade'),
           Row(page='Roll Advert'),
           Row(page='Logout'),
           Row(page='Save Settings'),
           Row(page='Cancellation Confirmation'),
           Row(page='About'),
           Row(page='Settings'),
           Row(page='Add to Playlist'),
```

```
            Row(page='Add Friend'),
            Row(page='NextSong'),
            Row(page='Thumbs Up'),
            Row(page='Help'),
            Row(page='Upgrade'),
            Row(page='Error'),
            Row(page='Submit Upgrade')]
```

```
In [164]: df_clean.select('method').dropDuplicates().collect()
```

```
Out[164]: [Row(method='PUT'), Row(method='GET')]
```

```
In [165]: df_clean.select('status').dropDuplicates().collect()
```

```
Out[165]: [Row(status=307), Row(status=404), Row(status=200)]
```

```
In [166]: df_clean.select('userAgent').dropDuplicates().show(5)
```

```
+--------------------+
|           userAgent|
+--------------------+
|"Mozilla/5.0 (Mac...|
|"Mozilla/5.0 (Win...|
|Mozilla/5.0 (X11;...|
|"Mozilla/5.0 (Mac...|
|"Mozilla/5.0 (Mac...|
+--------------------+
only showing top 5 rows
```

```
In [167]: #users who made cancellation
          df_canceled = df_clean.filter(df_clean.page == 'Cancellation Confirmation')
          num_churn = df_canceled.count()
          total_users = df_clean.select('userId').dropDuplicates().count()
          print('percentage of churn: {:.2f}%'.format((num_churn/total_users)*100))
```

```
percentage of churn: 23.11%
```

```
In [168]: #Define churn column
          check_churn = udf(lambda x: 1 if x == 'Cancellation Confirmation' else 0)
          df_clean = df_clean.withColumn('churn',check_churn(df_clean.page).cast(DoubleType()) )
          df_clean.head(1)
```

```
Out[168]: [Row(artist='Martha Tilston', auth='Logged In', firstName='Colin', gender='M', itemInS
```

```
In [169]: #look and the history for user 18 who churned
          df_clean.select(["userId", "page", "ts", "level", "sessionId"]).where(df_clean.userId
```

```
Out[169]: [Row(userId='32', page='NextSong', ts=1538578897000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538579134000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538579540000, level='free', sessionId=218),
           Row(userId='32', page='Roll Advert', ts=1538579565000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538579814000, level='free', sessionId=218),
           Row(userId='32', page='Thumbs Up', ts=1538579815000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538580056000, level='free', sessionId=218),
           Row(userId='32', page='Add to Playlist', ts=1538580068000, level='free', sessionId=21
           Row(userId='32', page='NextSong', ts=1538580267000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538580477000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538581129000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538581369000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538581726000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538581924000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538582217000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538582511000, level='free', sessionId=218),
           Row(userId='32', page='Roll Advert', ts=1538582541000, level='free', sessionId=218),
           Row(userId='32', page='Logout', ts=1538582542000, level='free', sessionId=218),
           Row(userId='32', page='Home', ts=1538582763000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538582788000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538582959000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538583211000, level='free', sessionId=218),
           Row(userId='32', page='Add to Playlist', ts=1538583240000, level='free', sessionId=21
           Row(userId='32', page='NextSong', ts=1538583413000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538583682000, level='free', sessionId=218),
           Row(userId='32', page='Add to Playlist', ts=1538583714000, level='free', sessionId=21
           Row(userId='32', page='NextSong', ts=1538583896000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538584102000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538584365000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538584654000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538584876000, level='free', sessionId=218),
           Row(userId='32', page='Thumbs Up', ts=1538584877000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538585100000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538585387000, level='free', sessionId=218),
           Row(userId='32', page='NextSong', ts=1538840277000, level='free', sessionId=404),
           Row(userId='32', page='Add to Playlist', ts=1538840295000, level='free', sessionId=40
           Row(userId='32', page='NextSong', ts=1538840413000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538840792000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538840969000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538841141000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538841431000, level='free', sessionId=404),
           Row(userId='32', page='Thumbs Up', ts=1538841432000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538841623000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538841884000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538842075000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538842285000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538842573000, level='free', sessionId=404),
           Row(userId='32', page='NextSong', ts=1538842823000, level='free', sessionId=404),
```

```
Row(userId='32', page='Thumbs Up', ts=1538842824000, level='free', sessionId=404),
Row(userId='32', page='NextSong', ts=1538843073000, level='free', sessionId=404),
Row(userId='32', page='Thumbs Up', ts=1538843074000, level='free', sessionId=404),
Row(userId='32', page='NextSong', ts=1539022740000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539022944000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539023204000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539023448000, level='free', sessionId=540),
Row(userId='32', page='Roll Advert', ts=1539023476000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539023601000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539023728000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539023939000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539024197000, level='free', sessionId=540),
Row(userId='32', page='NextSong', ts=1539024477000, level='free', sessionId=540),
Row(userId='32', page='Upgrade', ts=1539024605000, level='free', sessionId=540),
Row(userId='32', page='Submit Upgrade', ts=1539024606000, level='free', sessionId=540
Row(userId='32', page='Home', ts=1539024755000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539024757000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539025168000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539025384000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539025587000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539025834000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539026033000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539026214000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539026483000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539026688000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539026906000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539027202000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539027448000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539027652000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539028009000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539028351000, level='paid', sessionId=540),
Row(userId='32', page='Thumbs Up', ts=1539028352000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539028575000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539028794000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539028935000, level='paid', sessionId=540),
Row(userId='32', page='Thumbs Down', ts=1539028936000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539029106000, level='paid', sessionId=540),
Row(userId='32', page='Add to Playlist', ts=1539029266000, level='paid', sessionId=54
Row(userId='32', page='NextSong', ts=1539029315000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539029770000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539029965000, level='paid', sessionId=540),
Row(userId='32', page='Thumbs Up', ts=1539029966000, level='paid', sessionId=540),
Row(userId='32', page='Add to Playlist', ts=1539030139000, level='paid', sessionId=54
Row(userId='32', page='NextSong', ts=1539030174000, level='paid', sessionId=540),
Row(userId='32', page='Downgrade', ts=1539030302000, level='paid', sessionId=540),
Row(userId='32', page='Home', ts=1539030575000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539030623000, level='paid', sessionId=540),
Row(userId='32', page='NextSong', ts=1539030942000, level='paid', sessionId=540),
```

```
 Row(userId='32', page='Add to Playlist', ts=1539031152000, level='paid', sessionId=54
 Row(userId='32', page='NextSong', ts=1539031179000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539031426000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539031661000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539031887000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539032235000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539032608000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539032741000, level='paid', sessionId=540),
 Row(userId='32', page='NextSong', ts=1539032978000, level='paid', sessionId=540),
 Row(userId='32', page='Downgrade', ts=1539033030000, level='paid', sessionId=540),
 Row(userId='32', page='Cancel', ts=1539033031000, level='paid', sessionId=540),
 Row(userId='32', page='Cancellation Confirmation', ts=1539033046000, level='paid', se
```

In [170]:
```
churn_users = df_canceled.select('userId').collect()
list_churn_users = [ row.userId for row in churn_users ]
```

In [171]:
```
df_sessions_per_user = df_clean.groupBy('userId').agg(countDistinct('sessionId').alias
```

In [172]:
```
df_clean.groupBy(['userId','gender']).agg(countDistinct('sessionId').alias('sessions')
                                          countDistinct(when(col('level')=='paid',col('sessionId'
                                          countDistinct(when(col('level')=='free',col('sessionId'
                                          countDistinct('song').alias('songs'), \
                                          count(when(col('page') =='Error', 1)).alias('errors'),
                                          count(when(col('page') =='Help', 1)).alias('help'), \
                                          count(when((col('status') == 404)|(col('status') == 307
                                          count(when(col("churn")>0,True)).alias('churn'))\
                                      .sort('userId').collect()
```

Out[172]:
```
[Row(userId='10', gender='M', sessions=6, paid_sessions=6, free_sessions=0, songs=629,
 Row(userId='100', gender='M', sessions=35, paid_sessions=29, free_sessions=8, songs=2
 Row(userId='100001', gender='F', sessions=4, paid_sessions=0, free_sessions=4, songs=
 Row(userId='100002', gender='F', sessions=4, paid_sessions=4, free_sessions=0, songs=
 Row(userId='100003', gender='F', sessions=2, paid_sessions=0, free_sessions=2, songs=
 Row(userId='100004', gender='F', sessions=21, paid_sessions=9, free_sessions=16, song
 Row(userId='100005', gender='M', sessions=5, paid_sessions=0, free_sessions=5, songs=
 Row(userId='100006', gender='F', sessions=1, paid_sessions=0, free_sessions=1, songs=
 Row(userId='100007', gender='F', sessions=9, paid_sessions=9, free_sessions=0, songs=
 Row(userId='100008', gender='F', sessions=6, paid_sessions=5, free_sessions=2, songs=
 Row(userId='100009', gender='M', sessions=10, paid_sessions=4, free_sessions=8, songs
 Row(userId='100010', gender='F', sessions=7, paid_sessions=0, free_sessions=7, songs=
 Row(userId='100011', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=
 Row(userId='100012', gender='M', sessions=7, paid_sessions=2, free_sessions=7, songs=
 Row(userId='100013', gender='F', sessions=14, paid_sessions=12, free_sessions=3, song
 Row(userId='100014', gender='M', sessions=6, paid_sessions=6, free_sessions=0, songs=
 Row(userId='100015', gender='F', sessions=12, paid_sessions=7, free_sessions=8, songs
 Row(userId='100016', gender='M', sessions=8, paid_sessions=7, free_sessions=2, songs=
 Row(userId='100017', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=
 Row(userId='100018', gender='M', sessions=21, paid_sessions=18, free_sessions=6, song
 Row(userId='100019', gender='M', sessions=2, paid_sessions=0, free_sessions=2, songs=
```

```
Row(userId='100021', gender='M', sessions=5, paid_sessions=0, free_sessions=5, songs=
Row(userId='100022', gender='F', sessions=20, paid_sessions=0, free_sessions=20, song
Row(userId='100023', gender='M', sessions=10, paid_sessions=9, free_sessions=2, songs
Row(userId='100024', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=
Row(userId='100025', gender='F', sessions=7, paid_sessions=6, free_sessions=2, songs=
Row(userId='101', gender='M', sessions=10, paid_sessions=8, free_sessions=3, songs=16
Row(userId='102', gender='M', sessions=8, paid_sessions=0, free_sessions=8, songs=157
Row(userId='103', gender='F', sessions=11, paid_sessions=6, free_sessions=7, songs=98
Row(userId='104', gender='F', sessions=26, paid_sessions=18, free_sessions=9, songs=1
Row(userId='105', gender='M', sessions=5, paid_sessions=5, free_sessions=1, songs=723
Row(userId='106', gender='F', sessions=12, paid_sessions=10, free_sessions=3, songs=7
Row(userId='107', gender='F', sessions=4, paid_sessions=0, free_sessions=4, songs=239
Row(userId='108', gender='F', sessions=6, paid_sessions=4, free_sessions=3, songs=598
Row(userId='109', gender='F', sessions=12, paid_sessions=10, free_sessions=4, songs=6
Row(userId='11', gender='F', sessions=16, paid_sessions=4, free_sessions=14, songs=61
Row(userId='110', gender='M', sessions=5, paid_sessions=0, free_sessions=5, songs=172
Row(userId='111', gender='F', sessions=7, paid_sessions=3, free_sessions=5, songs=656
Row(userId='112', gender='M', sessions=10, paid_sessions=0, free_sessions=10, songs=2
Row(userId='113', gender='F', sessions=24, paid_sessions=22, free_sessions=3, songs=1
Row(userId='114', gender='M', sessions=17, paid_sessions=13, free_sessions=5, songs=1
Row(userId='115', gender='F', sessions=10, paid_sessions=9, free_sessions=2, songs=15
Row(userId='116', gender='F', sessions=3, paid_sessions=0, free_sessions=3, songs=62,
Row(userId='117', gender='F', sessions=13, paid_sessions=0, free_sessions=13, songs=3
Row(userId='118', gender='F', sessions=24, paid_sessions=22, free_sessions=3, songs=2
Row(userId='119', gender='F', sessions=6, paid_sessions=0, free_sessions=6, songs=171
Row(userId='12', gender='F', sessions=9, paid_sessions=5, free_sessions=7, songs=817,
Row(userId='120', gender='F', sessions=14, paid_sessions=14, free_sessions=0, songs=1
Row(userId='121', gender='M', sessions=13, paid_sessions=5, free_sessions=9, songs=68
Row(userId='122', gender='F', sessions=4, paid_sessions=1, free_sessions=4, songs=41,
Row(userId='123', gender='M', sessions=8, paid_sessions=0, free_sessions=8, songs=150
Row(userId='124', gender='F', sessions=29, paid_sessions=29, free_sessions=0, songs=3
Row(userId='125', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=8,
Row(userId='126', gender='M', sessions=30, paid_sessions=17, free_sessions=14, songs=
Row(userId='127', gender='F', sessions=17, paid_sessions=17, free_sessions=0, songs=1
Row(userId='128', gender='M', sessions=17, paid_sessions=14, free_sessions=4, songs=1
Row(userId='129', gender='M', sessions=7, paid_sessions=5, free_sessions=3, songs=315
Row(userId='13', gender='F', sessions=33, paid_sessions=4, free_sessions=31, songs=11
Row(userId='131', gender='M', sessions=19, paid_sessions=16, free_sessions=5, songs=1
Row(userId='132', gender='F', sessions=16, paid_sessions=16, free_sessions=1, songs=1
Row(userId='133', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=32,
Row(userId='134', gender='M', sessions=4, paid_sessions=0, free_sessions=4, songs=37,
Row(userId='135', gender='F', sessions=1, paid_sessions=0, free_sessions=1, songs=6,
Row(userId='136', gender='F', sessions=22, paid_sessions=16, free_sessions=7, songs=1
Row(userId='137', gender='M', sessions=3, paid_sessions=1, free_sessions=3, songs=153
Row(userId='138', gender='M', sessions=15, paid_sessions=10, free_sessions=6, songs=1
Row(userId='139', gender='M', sessions=4, paid_sessions=4, free_sessions=1, songs=368
Row(userId='14', gender='M', sessions=11, paid_sessions=11, free_sessions=0, songs=11
Row(userId='140', gender='F', sessions=71, paid_sessions=49, free_sessions=29, songs=
```

```
Row(userId='141', gender='F', sessions=11, paid_sessions=11, free_sessions=1, songs=8
Row(userId='142', gender='M', sessions=16, paid_sessions=16, free_sessions=1, songs=1
Row(userId='143', gender='F', sessions=6, paid_sessions=0, free_sessions=6, songs=99,
Row(userId='144', gender='M', sessions=3, paid_sessions=0, free_sessions=3, songs=98,
Row(userId='145', gender='F', sessions=15, paid_sessions=15, free_sessions=0, songs=1
Row(userId='146', gender='M', sessions=17, paid_sessions=0, free_sessions=17, songs=6
Row(userId='147', gender='M', sessions=10, paid_sessions=4, free_sessions=7, songs=52
Row(userId='148', gender='M', sessions=12, paid_sessions=0, free_sessions=12, songs=3
Row(userId='149', gender='M', sessions=3, paid_sessions=0, free_sessions=3, songs=184
Row(userId='15', gender='M', sessions=15, paid_sessions=15, free_sessions=0, songs=17
Row(userId='150', gender='M', sessions=9, paid_sessions=0, free_sessions=9, songs=177
Row(userId='151', gender='M', sessions=1, paid_sessions=1, free_sessions=0, songs=137
Row(userId='152', gender='M', sessions=20, paid_sessions=9, free_sessions=12, songs=1
Row(userId='153', gender='M', sessions=9, paid_sessions=9, free_sessions=1, songs=866
Row(userId='154', gender='F', sessions=3, paid_sessions=0, free_sessions=3, songs=83,
Row(userId='155', gender='F', sessions=6, paid_sessions=5, free_sessions=2, songs=759
Row(userId='156', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=3,
Row(userId='16', gender='F', sessions=9, paid_sessions=9, free_sessions=1, songs=643,
Row(userId='17', gender='F', sessions=7, paid_sessions=5, free_sessions=3, songs=875,
Row(userId='18', gender='M', sessions=6, paid_sessions=6, free_sessions=0, songs=412,
Row(userId='19', gender='F', sessions=1, paid_sessions=1, free_sessions=0, songs=209,
Row(userId='2', gender='F', sessions=7, paid_sessions=7, free_sessions=0, songs=713,
Row(userId='20', gender='F', sessions=19, paid_sessions=16, free_sessions=7, songs=16
Row(userId='200001', gender='M', sessions=6, paid_sessions=0, free_sessions=6, songs=
Row(userId='200002', gender='M', sessions=6, paid_sessions=3, free_sessions=4, songs=
Row(userId='200003', gender='F', sessions=14, paid_sessions=3, free_sessions=14, song
Row(userId='200004', gender='M', sessions=28, paid_sessions=28, free_sessions=0, song
Row(userId='200005', gender='M', sessions=6, paid_sessions=5, free_sessions=2, songs=
Row(userId='200006', gender='F', sessions=21, paid_sessions=0, free_sessions=21, song
Row(userId='200007', gender='F', sessions=3, paid_sessions=3, free_sessions=0, songs=
Row(userId='200008', gender='F', sessions=15, paid_sessions=15, free_sessions=1, song
Row(userId='200009', gender='M', sessions=15, paid_sessions=6, free_sessions=11, song
Row(userId='200010', gender='F', sessions=6, paid_sessions=0, free_sessions=6, songs=
Row(userId='200011', gender='M', sessions=11, paid_sessions=5, free_sessions=8, songs
Row(userId='200012', gender='M', sessions=2, paid_sessions=0, free_sessions=2, songs=
Row(userId='200014', gender='M', sessions=10, paid_sessions=1, free_sessions=10, song
Row(userId='200015', gender='M', sessions=11, paid_sessions=0, free_sessions=11, song
Row(userId='200016', gender='F', sessions=7, paid_sessions=0, free_sessions=7, songs=
Row(userId='200017', gender='M', sessions=12, paid_sessions=4, free_sessions=9, songs
Row(userId='200018', gender='M', sessions=9, paid_sessions=0, free_sessions=9, songs=
Row(userId='200019', gender='M', sessions=10, paid_sessions=6, free_sessions=6, songs
Row(userId='200020', gender='M', sessions=36, paid_sessions=5, free_sessions=33, song
Row(userId='200021', gender='F', sessions=17, paid_sessions=13, free_sessions=5, song
Row(userId='200022', gender='M', sessions=8, paid_sessions=0, free_sessions=8, songs=
Row(userId='200023', gender='F', sessions=76, paid_sessions=25, free_sessions=56, son
Row(userId='200024', gender='M', sessions=8, paid_sessions=2, free_sessions=7, songs=
Row(userId='200025', gender='M', sessions=14, paid_sessions=6, free_sessions=10, song
Row(userId='21', gender='M', sessions=4, paid_sessions=4, free_sessions=0, songs=476,
```

```
Row(userId='22', gender='F', sessions=2, paid_sessions=0, free_sessions=2, songs=28,
Row(userId='23', gender='F', sessions=4, paid_sessions=1, free_sessions=4, songs=624,
Row(userId='24', gender='M', sessions=15, paid_sessions=3, free_sessions=14, songs=46
Row(userId='25', gender='F', sessions=15, paid_sessions=11, free_sessions=6, songs=16
Row(userId='26', gender='M', sessions=8, paid_sessions=1, free_sessions=8, songs=251,
Row(userId='27', gender='M', sessions=9, paid_sessions=0, free_sessions=9, songs=217,
Row(userId='28', gender='F', sessions=21, paid_sessions=6, free_sessions=16, songs=13
Row(userId='29', gender='M', sessions=34, paid_sessions=27, free_sessions=8, songs=25
Row(userId='3', gender='M', sessions=4, paid_sessions=4, free_sessions=0, songs=211,
Row(userId='30', gender='M', sessions=32, paid_sessions=5, free_sessions=29, songs=12
Row(userId='300001', gender='F', sessions=19, paid_sessions=19, free_sessions=1, song
Row(userId='300002', gender='F', sessions=18, paid_sessions=18, free_sessions=3, song
Row(userId='300003', gender='M', sessions=1, paid_sessions=0, free_sessions=1, songs=
Row(userId='300004', gender='F', sessions=3, paid_sessions=3, free_sessions=2, songs=
Row(userId='300005', gender='F', sessions=6, paid_sessions=6, free_sessions=1, songs=
Row(userId='300006', gender='M', sessions=5, paid_sessions=1, free_sessions=5, songs=
Row(userId='300007', gender='M', sessions=1, paid_sessions=1, free_sessions=0, songs=
Row(userId='300008', gender='F', sessions=18, paid_sessions=18, free_sessions=0, song
Row(userId='300009', gender='F', sessions=18, paid_sessions=18, free_sessions=1, song
Row(userId='300010', gender='M', sessions=7, paid_sessions=0, free_sessions=7, songs=
Row(userId='300011', gender='F', sessions=76, paid_sessions=69, free_sessions=10, son
Row(userId='300012', gender='M', sessions=7, paid_sessions=6, free_sessions=2, songs=
Row(userId='300013', gender='M', sessions=6, paid_sessions=6, free_sessions=0, songs=
Row(userId='300014', gender='M', sessions=7, paid_sessions=7, free_sessions=1, songs=
Row(userId='300015', gender='M', sessions=33, paid_sessions=33, free_sessions=3, song
Row(userId='300016', gender='M', sessions=13, paid_sessions=13, free_sessions=1, song
Row(userId='300017', gender='F', sessions=63, paid_sessions=63, free_sessions=0, song
Row(userId='300018', gender='M', sessions=28, paid_sessions=27, free_sessions=2, song
Row(userId='300019', gender='M', sessions=9, paid_sessions=9, free_sessions=1, songs=
Row(userId='300020', gender='F', sessions=3, paid_sessions=3, free_sessions=0, songs=
Row(userId='300021', gender='F', sessions=58, paid_sessions=58, free_sessions=1, song
Row(userId='300022', gender='M', sessions=11, paid_sessions=11, free_sessions=0, song
Row(userId='300023', gender='F', sessions=27, paid_sessions=27, free_sessions=2, song
Row(userId='300024', gender='F', sessions=1, paid_sessions=1, free_sessions=0, songs=
Row(userId='300025', gender='M', sessions=16, paid_sessions=15, free_sessions=2, song
Row(userId='32', gender='M', sessions=3, paid_sessions=1, free_sessions=3, songs=80,
Row(userId='33', gender='M', sessions=12, paid_sessions=12, free_sessions=0, songs=11
Row(userId='34', gender='M', sessions=3, paid_sessions=0, free_sessions=3, songs=53,
Row(userId='35', gender='F', sessions=38, paid_sessions=11, free_sessions=31, songs=1
Row(userId='36', gender='F', sessions=20, paid_sessions=5, free_sessions=16, songs=10
Row(userId='37', gender='M', sessions=15, paid_sessions=12, free_sessions=4, songs=12
Row(userId='38', gender='M', sessions=16, paid_sessions=12, free_sessions=6, songs=11
Row(userId='39', gender='F', sessions=107, paid_sessions=51, free_sessions=62, songs=
Row(userId='4', gender='M', sessions=22, paid_sessions=14, free_sessions=9, songs=179
Row(userId='40', gender='F', sessions=17, paid_sessions=10, free_sessions=8, songs=10
Row(userId='41', gender='F', sessions=12, paid_sessions=12, free_sessions=0, songs=16
Row(userId='42', gender='F', sessions=47, paid_sessions=42, free_sessions=6, songs=29
Row(userId='43', gender='F', sessions=4, paid_sessions=0, free_sessions=4, songs=171,
```

```
Row(userId='44', gender='F', sessions=3, paid_sessions=3, free_sessions=1, songs=410,
Row(userId='45', gender='F', sessions=16, paid_sessions=11, free_sessions=6, songs=13
Row(userId='46', gender='F', sessions=10, paid_sessions=8, free_sessions=3, songs=847
Row(userId='47', gender='M', sessions=9, paid_sessions=0, free_sessions=9, songs=200,
Row(userId='49', gender='M', sessions=10, paid_sessions=8, free_sessions=4, songs=813
Row(userId='5', gender='M', sessions=6, paid_sessions=0, free_sessions=6, songs=159,
Row(userId='50', gender='F', sessions=8, paid_sessions=2, free_sessions=7, songs=476,
Row(userId='51', gender='M', sessions=10, paid_sessions=10, free_sessions=0, songs=18
Row(userId='52', gender='F', sessions=15, paid_sessions=5, free_sessions=11, songs=99
Row(userId='53', gender='M', sessions=22, paid_sessions=11, free_sessions=12, songs=1
Row(userId='54', gender='F', sessions=37, paid_sessions=26, free_sessions=13, songs=2
Row(userId='55', gender='M', sessions=11, paid_sessions=2, free_sessions=10, songs=36
Row(userId='56', gender='M', sessions=21, paid_sessions=10, free_sessions=12, songs=6
Row(userId='57', gender='M', sessions=2, paid_sessions=0, free_sessions=2, songs=90,
Row(userId='58', gender='M', sessions=12, paid_sessions=11, free_sessions=2, songs=15
Row(userId='59', gender='M', sessions=7, paid_sessions=4, free_sessions=6, songs=680,
Row(userId='6', gender='M', sessions=24, paid_sessions=20, free_sessions=5, songs=267
Row(userId='60', gender='M', sessions=18, paid_sessions=18, free_sessions=1, songs=14
Row(userId='61', gender='M', sessions=19, paid_sessions=8, free_sessions=14, songs=14
Row(userId='62', gender='M', sessions=8, paid_sessions=8, free_sessions=0, songs=1422
Row(userId='63', gender='F', sessions=1, paid_sessions=0, free_sessions=1, songs=86,
Row(userId='64', gender='M', sessions=3, paid_sessions=0, free_sessions=3, songs=46,
Row(userId='65', gender='M', sessions=23, paid_sessions=18, free_sessions=6, songs=18
Row(userId='66', gender='F', sessions=21, paid_sessions=11, free_sessions=11, songs=9
Row(userId='67', gender='M', sessions=13, paid_sessions=9, free_sessions=5, songs=101
Row(userId='68', gender='F', sessions=2, paid_sessions=0, free_sessions=2, songs=29,
Row(userId='69', gender='F', sessions=9, paid_sessions=8, free_sessions=2, songs=1036
Row(userId='7', gender='M', sessions=7, paid_sessions=0, free_sessions=7, songs=148,
Row(userId='70', gender='M', sessions=18, paid_sessions=16, free_sessions=3, songs=13
Row(userId='71', gender='M', sessions=4, paid_sessions=3, free_sessions=2, songs=258,
Row(userId='72', gender='F', sessions=1, paid_sessions=0, free_sessions=1, songs=85,
Row(userId='73', gender='F', sessions=6, paid_sessions=6, free_sessions=1, songs=363,
Row(userId='74', gender='F', sessions=23, paid_sessions=14, free_sessions=12, songs=2
Row(userId='75', gender='F', sessions=8, paid_sessions=8, free_sessions=0, songs=750,
Row(userId='76', gender='M', sessions=6, paid_sessions=0, free_sessions=6, songs=208,
Row(userId='77', gender='F', sessions=8, paid_sessions=7, free_sessions=3, songs=971,
Row(userId='78', gender='F', sessions=9, paid_sessions=0, free_sessions=9, songs=249,
Row(userId='79', gender='M', sessions=5, paid_sessions=2, free_sessions=4, songs=239,
Row(userId='8', gender='F', sessions=7, paid_sessions=0, free_sessions=7, songs=244,
Row(userId='80', gender='F', sessions=12, paid_sessions=0, free_sessions=12, songs=35
Row(userId='81', gender='M', sessions=19, paid_sessions=18, free_sessions=3, songs=17
Row(userId='82', gender='F', sessions=17, paid_sessions=8, free_sessions=10, songs=15
Row(userId='83', gender='M', sessions=28, paid_sessions=18, free_sessions=11, songs=1
Row(userId='84', gender='F', sessions=3, paid_sessions=0, free_sessions=3, songs=71,
Row(userId='85', gender='M', sessions=41, paid_sessions=25, free_sessions=21, songs=3
Row(userId='86', gender='M', sessions=10, paid_sessions=5, free_sessions=6, songs=625
Row(userId='87', gender='M', sessions=25, paid_sessions=2, free_sessions=24, songs=72
Row(userId='88', gender='F', sessions=26, paid_sessions=14, free_sessions=13, songs=1
```

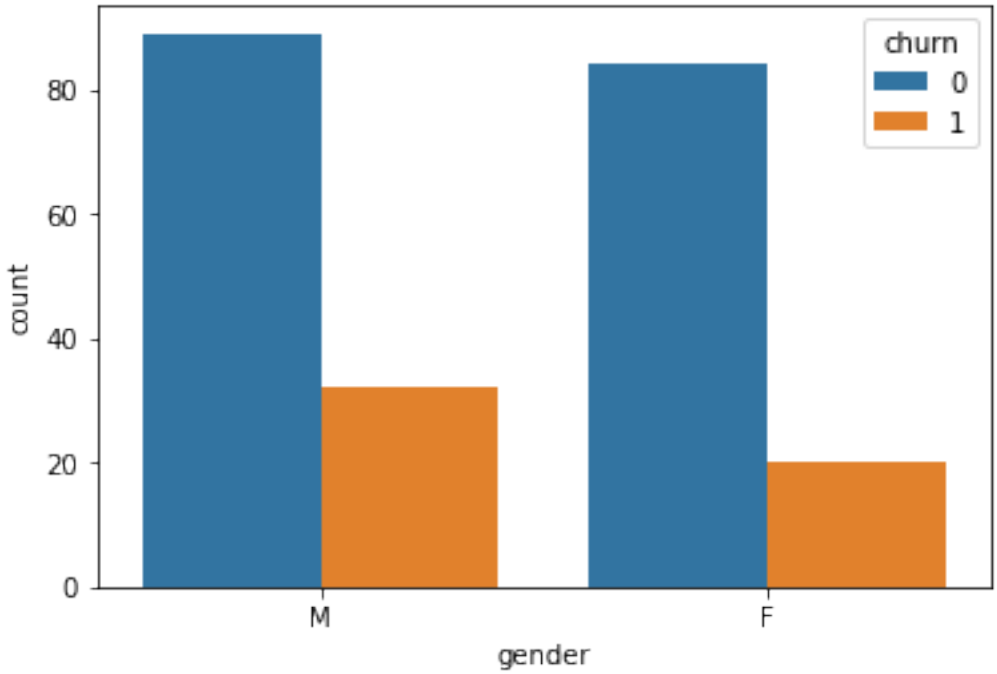
```
Row(userId='89', gender='M', sessions=5, paid_sessions=4, free_sessions=2, songs=626,
Row(userId='9', gender='M', sessions=31, paid_sessions=28, free_sessions=6, songs=230
Row(userId='90', gender='M', sessions=5, paid_sessions=0, free_sessions=5, songs=37,
Row(userId='91', gender='M', sessions=9, paid_sessions=7, free_sessions=3, songs=2218
Row(userId='92', gender='F', sessions=86, paid_sessions=60, free_sessions=28, songs=4
Row(userId='93', gender='M', sessions=12, paid_sessions=3, free_sessions=10, songs=60
Row(userId='94', gender='F', sessions=6, paid_sessions=0, free_sessions=6, songs=143,
Row(userId='95', gender='F', sessions=33, paid_sessions=10, free_sessions=24, songs=1
Row(userId='96', gender='F', sessions=19, paid_sessions=15, free_sessions=6, songs=16
Row(userId='97', gender='F', sessions=30, paid_sessions=17, free_sessions=14, songs=1
Row(userId='98', gender='M', sessions=28, paid_sessions=15, free_sessions=14, songs=2
Row(userId='99', gender='F', sessions=11, paid_sessions=7, free_sessions=5, songs=485
```

In [173]:
```
agg = df_clean.groupBy(['userId','gender']).agg(countDistinct('sessionId').alias('sess
                                    countDistinct(when(col('level')=='paid',col('sessionId'
                                    countDistinct(when(col('level')=='free',col('sessionId'
                                    countDistinct('song').alias('songs'), \
                                    count(when(col('page') =='Error', 1)).alias('errors'),
                                    count(when(col('page') =='Help', 1)).alias('help'), \
                                    count(when((col('status') == 404)|(col('status') == 307
                                    count(when(col("churn")>0,True)).alias('churn'))\
                                .sort('userId').collect()
df_agg = spark.createDataFrame(agg)
```

In [174]:
```
pd_df = df_agg.toPandas()
pd_df = pd_df.set_index('userId')
```

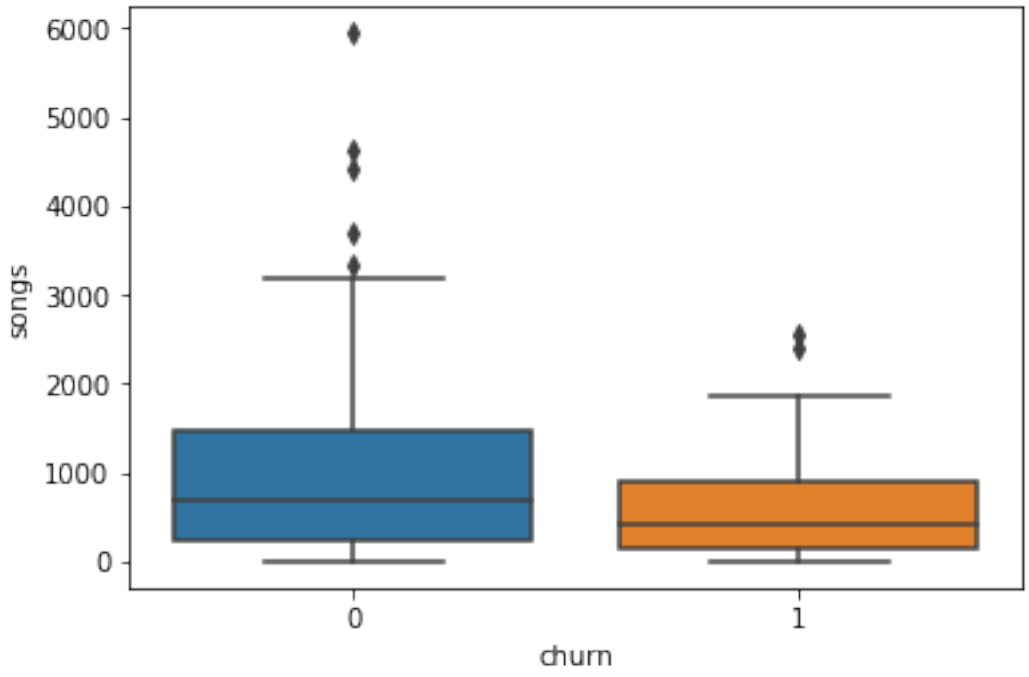
In [175]:
```
sns.countplot(x='gender', hue="churn", data=pd_df)
```

Out[175]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1f4c05a6d8>
```



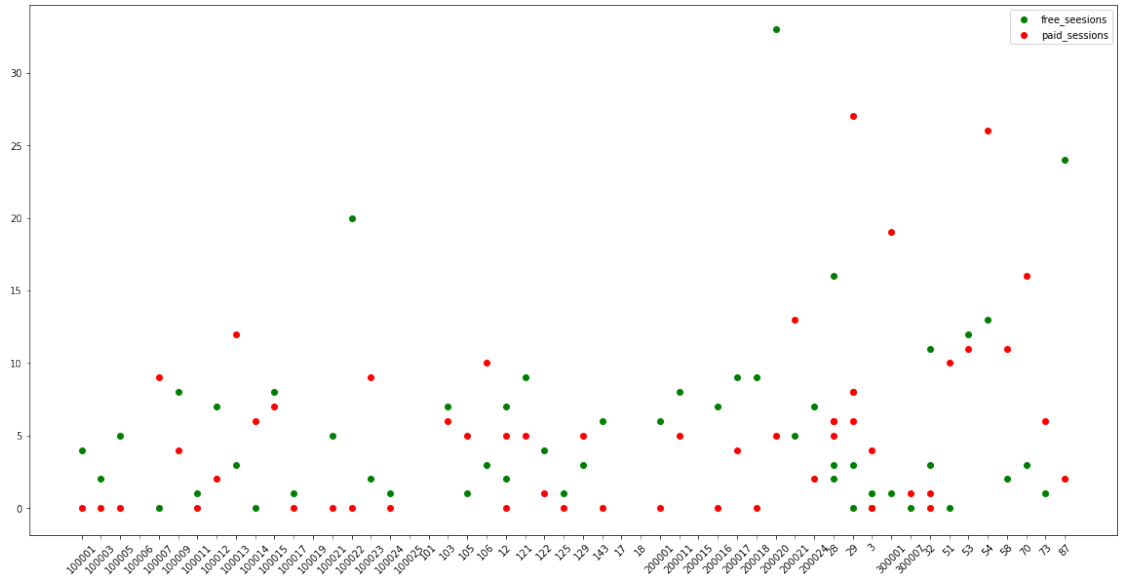

```
In [176]: sns.boxplot(x='churn',y='songs',data=pd_df)
```

```
Out[176]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f44655748>
```



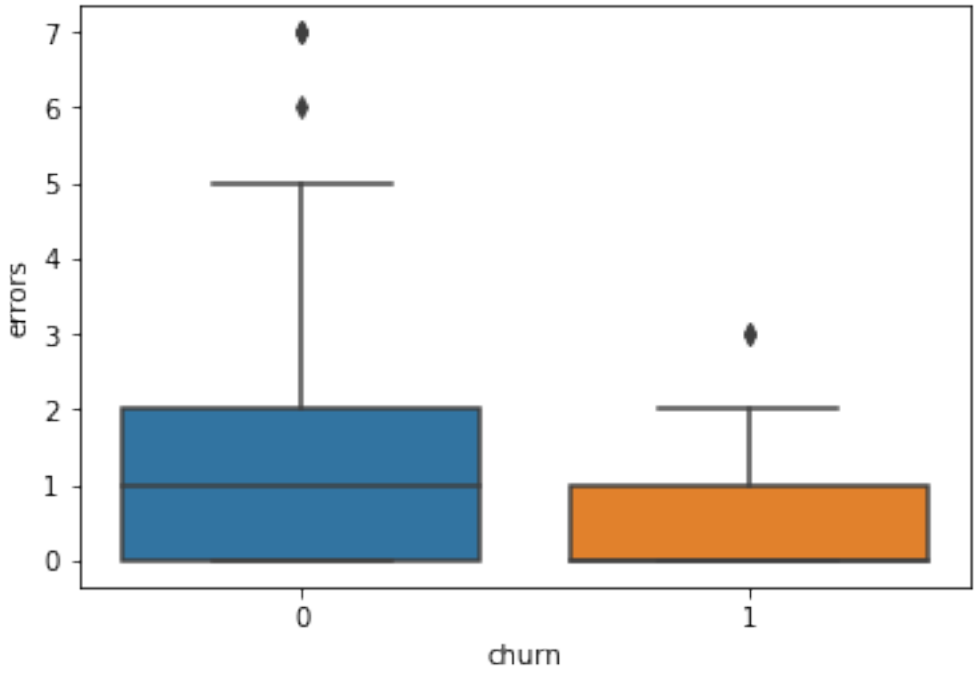
```
In [177]: pd_df_canceled = df_canceled.toPandas()
          x = pd_df_canceled['userId']
          free_sessions = [ pd_df.loc[id,['free_sessions'][0]] for id in x]
          paid_sessions = [ pd_df.loc[id,['paid_sessions'][0]] for id in x]

In [178]: plt.figure(figsize=[20,10])
          plt.scatter(x,free_sessions, c='g')
          plt.scatter(x,paid_sessions, c='r')
          plt.xticks(rotation=45)
          plt.legend(['free_seesions','paid_sessions'])
          plt.show()
```



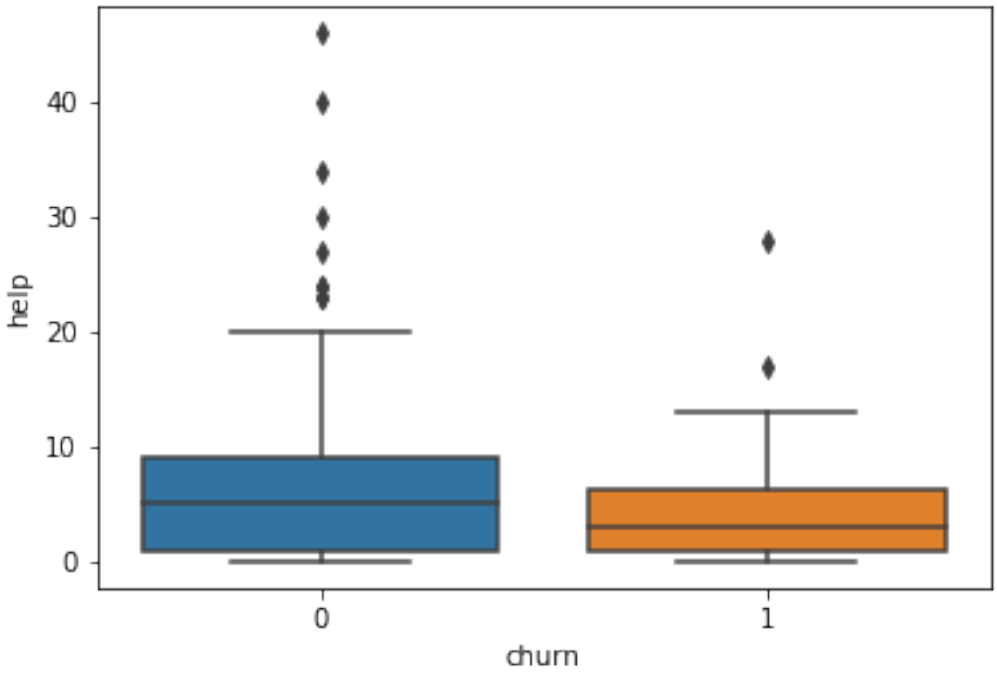

```
In [179]: sns.boxplot(x='churn',y='errors',data=pd_df)

Out[179]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f4446bb70>
```



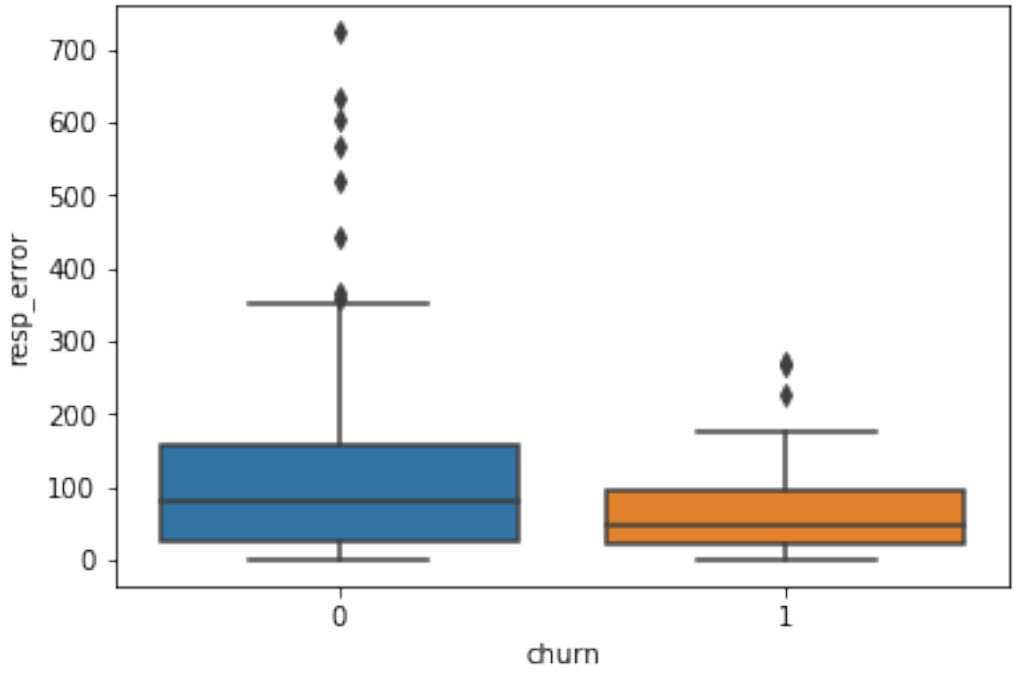

```
In [180]: sns.boxplot(x='churn',y='help',data=pd_df)
```

```
Out[180]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f443f0b38>
```



```
In [181]: sns.boxplot(x='churn',y='resp_error',data=pd_df)

Out[181]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1f4436bd30>
```



4 Feature Engineering

Once you've familiarized yourself with the data, build out the features you find promising to train your model on. To work with the full dataset, you can follow the following steps. - Write a script to extract the necessary features from the smaller subset of data - Ensure that your script is scalable, using the best practices discussed in Lesson 3 - Try your script on the full data set, debugging your script if necessary

If you are working in the classroom workspace, you can just extract features based on the small subset of data contained here. Be sure to transfer over this work to the larger dataset when you work on your Spark cluster.

To be able to work with the dataset we need to create a new dataframe with one row for each user, hence some changes needs to be conducted on the original features as follows: - artist: is replace with number of artists - auth: dropped since the information is captured by the number of paid and free sessions - firstName: This column can be dropped since we have userId to distinguish each user - gender: replaced with two columns Female and Male with values 0 or 1 - itemInSession: is replaced with the avg(itemInSession) - lastName: This column can be dropped since we have userId to distinguish each user - length: This column is replaced with the

avg(length) - level: replaced with two columns free_sessions and paid_sessions
- location: can extract the state part of the location and then add a column for each state - method: is dropped - page: replaced with columns for each page type, in each column the number of times the page was browsed - registration: replaced with col activeDuration; the number of days from registration until the last event - sessionId: replaced with two columns free_sessions and paid_sessions - song: replace with number of songs - status: replace with three columns:307,404 and 200 and each column contains a number represents the number of times the error appeared - ts: replaced with col activeDuration; the number of days from registration until the last event - userAgent: is dropped - userId: kept as is After that we will check for null values and remove any column that a null values Note: from the page columns I didn't add page_NextPage column since its the same as number of songs, and I didn't add page_CancellationConfirmation since it is the same as the ChurnColumn

```
In [182]: df_clean.groupBy(['userId','registration']).count().sort('userId').collect()
```

```
Out[182]: [Row(userId='10', registration=1538159495000, count=795),
           Row(userId='100', registration=1537982255000, count=3214),
           Row(userId='100001', registration=1534627466000, count=187),
           Row(userId='100002', registration=1529934689000, count=218),
           Row(userId='100003', registration=1537309344000, count=78),
           Row(userId='100004', registration=1528560242000, count=1245),
           Row(userId='100005', registration=1532610926000, count=216),
           Row(userId='100006', registration=1537964483000, count=44),
           Row(userId='100007', registration=1533522419000, count=520),
           Row(userId='100008', registration=1537440271000, count=940),
           Row(userId='100009', registration=1537376437000, count=671),
           Row(userId='100010', registration=1538016340000, count=381),
           Row(userId='100011', registration=1537970819000, count=23),
           Row(userId='100012', registration=1537381154000, count=600),
           Row(userId='100013', registration=1537367773000, count=1392),
           Row(userId='100014', registration=1535389443000, count=310),
           Row(userId='100015', registration=1537208989000, count=1050),
           Row(userId='100016', registration=1536854322000, count=638),
           Row(userId='100017', registration=1533247234000, count=75),
           Row(userId='100018', registration=1533812833000, count=1288),
           Row(userId='100019', registration=1536257455000, count=89),
           Row(userId='100021', registration=1537550344000, count=319),
           Row(userId='100022', registration=1537898335000, count=964),
           Row(userId='100023', registration=1536679150000, count=494),
           Row(userId='100024', registration=1536343087000, count=36),
           Row(userId='100025', registration=1532625569000, count=623),
           Row(userId='101', registration=1535066380000, count=2149),
           Row(userId='102', registration=1537915702000, count=202),
           Row(userId='103', registration=1537699856000, count=1310),
           Row(userId='104', registration=1532498424000, count=2132),
           Row(userId='105', registration=1536817381000, count=918),
           Row(userId='106', registration=1537679535000, count=1010),
           Row(userId='107', registration=1536303841000, count=315),
```

```
Row(userId='108', registration=1538215963000, count=764),
Row(userId='109', registration=1534768517000, count=861),
Row(userId='11', registration=1532554781000, count=848),
Row(userId='110', registration=1537665002000, count=235),
Row(userId='111', registration=1536372490000, count=830),
Row(userId='112', registration=1536032681000, count=292),
Row(userId='113', registration=1532920994000, count=1895),
Row(userId='114', registration=1536831228000, count=1547),
Row(userId='115', registration=1536948181000, count=2076),
Row(userId='116', registration=1534613931000, count=79),
Row(userId='117', registration=1537142824000, count=445),
Row(userId='118', registration=1537893493000, count=2868),
Row(userId='119', registration=1526838391000, count=223),
Row(userId='12', registration=1533885783000, count=1064),
Row(userId='120', registration=1532472246000, count=1842),
Row(userId='121', registration=1528403713000, count=916),
Row(userId='122', registration=1535498705000, count=62),
Row(userId='123', registration=1534370883000, count=196),
Row(userId='124', registration=1532224335000, count=4825),
Row(userId='125', registration=1533157139000, count=11),
Row(userId='126', registration=1538227408000, count=3102),
Row(userId='127', registration=1536795126000, count=2125),
Row(userId='128', registration=1535269914000, count=2067),
Row(userId='129', registration=1538289776000, count=412),
Row(userId='13', registration=1533192032000, count=1654),
Row(userId='131', registration=1533102330000, count=1863),
Row(userId='132', registration=1537054553000, count=2304),
Row(userId='133', registration=1536126401000, count=44),
Row(userId='134', registration=1537618437000, count=52),
Row(userId='135', registration=1532433959000, count=6),
Row(userId='136', registration=1537034286000, count=2567),
Row(userId='137', registration=1532450666000, count=209),
Row(userId='138', registration=1537865819000, count=2469),
Row(userId='139', registration=1535271638000, count=457),
Row(userId='14', registration=1531910878000, count=1432),
Row(userId='140', registration=1536642109000, count=6880),
Row(userId='141', registration=1535903878000, count=1102),
Row(userId='142', registration=1538120859000, count=2240),
Row(userId='143', registration=1534255113000, count=149),
Row(userId='144', registration=1535047210000, count=125),
Row(userId='145', registration=1534779204000, count=1347),
Row(userId='146', registration=1536138073000, count=837),
Row(userId='147', registration=1537522781000, count=672),
Row(userId='148', registration=1537531076000, count=518),
Row(userId='149', registration=1536980520000, count=240),
Row(userId='15', registration=1538239045000, count=2278),
Row(userId='150', registration=1532899177000, count=234),
Row(userId='151', registration=1537862870000, count=163),
```

```
Row(userId='152', registration=1540940608000, count=1682),
Row(userId='153', registration=1541223737000, count=1131),
Row(userId='154', registration=1541268021000, count=118),
Row(userId='155', registration=1541370470000, count=1002),
Row(userId='156', registration=1543247354000, count=6),
Row(userId='16', registration=1536597540000, count=819),
Row(userId='17', registration=1538333829000, count=1102),
Row(userId='18', registration=1535623466000, count=513),
Row(userId='19', registration=1537306307000, count=259),
Row(userId='2', registration=1536799770000, count=899),
Row(userId='20', registration=1536844410000, count=2195),
Row(userId='200001', registration=1538255180000, count=158),
Row(userId='200002', registration=1536269906000, count=474),
Row(userId='200003', registration=1530333492000, count=977),
Row(userId='200004', registration=1537672236000, count=2130),
Row(userId='200005', registration=1533532298000, count=174),
Row(userId='200006', registration=1536963671000, count=740),
Row(userId='200007', registration=1537971741000, count=76),
Row(userId='200008', registration=1533670697000, count=2012),
Row(userId='200009', registration=1538313468000, count=1198),
Row(userId='200010', registration=1537764007000, count=317),
Row(userId='200011', registration=1531679695000, count=848),
Row(userId='200012', registration=1537939256000, count=84),
Row(userId='200014', registration=1532364714000, count=982),
Row(userId='200015', registration=1534175704000, count=349),
Row(userId='200016', registration=1537537765000, count=268),
Row(userId='200017', registration=1537043743000, count=620),
Row(userId='200018', registration=1536956945000, count=478),
Row(userId='200019', registration=1537638755000, count=639),
Row(userId='200020', registration=1535907116000, count=1498),
Row(userId='200021', registration=1535032914000, count=1502),
Row(userId='200022', registration=1533758535000, count=456),
Row(userId='200023', registration=1537812949000, count=3769),
Row(userId='200024', registration=1537897424000, count=552),
Row(userId='200025', registration=1532696273000, count=1026),
Row(userId='21', registration=1536855067000, count=589),
Row(userId='22', registration=1537681013000, count=40),
Row(userId='23', registration=1531281160000, count=782),
Row(userId='24', registration=1536954914000, count=630),
Row(userId='25', registration=1536287099000, count=2279),
Row(userId='26', registration=1536816590000, count=337),
Row(userId='27', registration=1534245996000, count=291),
Row(userId='28', registration=1537634865000, count=1781),
Row(userId='29', registration=1536988041000, count=3603),
Row(userId='3', registration=1533886191000, count=254),
Row(userId='30', registration=1538173362000, count=1775),
Row(userId='300001', registration=1526739206000, count=2168),
Row(userId='300002', registration=1532589384000, count=1961),
```

```
Row(userId='300003', registration=1530789251000, count=35),
Row(userId='300004', registration=1534513445000, count=262),
Row(userId='300005', registration=1528772084000, count=394),
Row(userId='300006', registration=1535361448000, count=379),
Row(userId='300007', registration=1537707098000, count=143),
Row(userId='300008', registration=1535660231000, count=1672),
Row(userId='300009', registration=1534859694000, count=1721),
Row(userId='300010', registration=1536850071000, count=351),
Row(userId='300011', registration=1538336771000, count=5732),
Row(userId='300012', registration=1530306321000, count=786),
Row(userId='300013', registration=1535400650000, count=411),
Row(userId='300014', registration=1533629444000, count=361),
Row(userId='300015', registration=1528780738000, count=3051),
Row(userId='300016', registration=1534622171000, count=735),
Row(userId='300017', registration=1537082511000, count=4428),
Row(userId='300018', registration=1535591420000, count=2046),
Row(userId='300019', registration=1536158069000, count=976),
Row(userId='300020', registration=1537920589000, count=296),
Row(userId='300021', registration=1537611935000, count=4659),
Row(userId='300022', registration=1534461078000, count=446),
Row(userId='300023', registration=1530514394000, count=2210),
Row(userId='300024', registration=1536589088000, count=102),
Row(userId='300025', registration=1536935950000, count=1603),
Row(userId='32', registration=1537167593000, count=108),
Row(userId='33', registration=1529252604000, count=1529),
Row(userId='34', registration=1536659447000, count=73),
Row(userId='35', registration=1537053412000, count=2005),
Row(userId='36', registration=1533908361000, count=1399),
Row(userId='37', registration=1535268719000, count=1682),
Row(userId='38', registration=1537120757000, count=1570),
Row(userId='39', registration=1529027541000, count=9632),
Row(userId='4', registration=1538169823000, count=2442),
Row(userId='40', registration=1535525247000, count=1322),
Row(userId='41', registration=1533438580000, count=2220),
Row(userId='42', registration=1537811988000, count=4257),
Row(userId='43', registration=1536563533000, count=240),
Row(userId='44', registration=1537779419000, count=512),
Row(userId='45', registration=1536398117000, count=1776),
Row(userId='46', registration=1537767796000, count=1120),
Row(userId='47', registration=1531769420000, count=251),
Row(userId='49', registration=1534318843000, count=1091),
Row(userId='5', registration=1537456136000, count=218),
Row(userId='50', registration=1537057938000, count=651),
Row(userId='51', registration=1538080987000, count=2464),
Row(userId='52', registration=1534613601000, count=1363),
Row(userId='53', registration=1538050164000, count=2064),
Row(userId='54', registration=1532482662000, count=3437),
Row(userId='55', registration=1536649963000, count=489),
```

```
Row(userId='56', registration=1537751138000, count=955),
Row(userId='57', registration=1535062159000, count=112),
Row(userId='58', registration=1537956751000, count=2027),
Row(userId='59', registration=1537890437000, count=872),
Row(userId='6', registration=1521380675000, count=3761),
Row(userId='60', registration=1537014411000, count=1950),
Row(userId='61', registration=1537188538000, count=1976),
Row(userId='62', registration=1531804365000, count=1883),
Row(userId='63', registration=1537586184000, count=107),
Row(userId='64', registration=1537539836000, count=68),
Row(userId='65', registration=1537347211000, count=2544),
Row(userId='66', registration=1532634173000, count=1288),
Row(userId='67', registration=1530674962000, count=1322),
Row(userId='68', registration=1534359998000, count=48),
Row(userId='69', registration=1536824159000, count=1342),
Row(userId='7', registration=1536667576000, count=201),
Row(userId='70', registration=1529643103000, count=1775),
Row(userId='71', registration=1538253263000, count=329),
Row(userId='72', registration=1530763799000, count=117),
Row(userId='73', registration=1536102943000, count=457),
Row(userId='74', registration=1537365219000, count=2887),
Row(userId='75', registration=1537201808000, count=957),
Row(userId='76', registration=1538065863000, count=272),
Row(userId='77', registration=1537749786000, count=1229),
Row(userId='78', registration=1538304455000, count=324),
Row(userId='79', registration=1536919023000, count=321),
Row(userId='8', registration=1533650280000, count=334),
Row(userId='80', registration=1538316317000, count=474),
Row(userId='81', registration=1535093367000, count=2335),
Row(userId='82', registration=1536963370000, count=2049),
Row(userId='83', registration=1532771049000, count=1496),
Row(userId='84', registration=1537821607000, count=97),
Row(userId='85', registration=1534133898000, count=4370),
Row(userId='86', registration=1531760527000, count=818),
Row(userId='87', registration=1534942082000, count=1003),
Row(userId='88', registration=1536663902000, count=2465),
Row(userId='89', registration=1535774889000, count=801),
Row(userId='9', registration=1538331630000, count=3191),
Row(userId='90', registration=1533995214000, count=45),
Row(userId='91', registration=1533398428000, count=3014),
Row(userId='92', registration=1536403972000, count=7230),
Row(userId='93', registration=1537484200000, count=815),
Row(userId='94', registration=1531350022000, count=187),
Row(userId='95', registration=1538211832000, count=2091),
Row(userId='96', registration=1537149749000, count=2176),
Row(userId='97', registration=1536019842000, count=2404),
Row(userId='98', registration=1538069638000, count=2891),
Row(userId='99', registration=1531811983000, count=614)]
```

```
In [183]: states = df_clean.select("location").distinct().rdd.flatMap(lambda x: list(x)).\
                                                        flatMap(lambda loc:[loc.split(', ',1)[-1]])
                                                        flatMap(lambda state: state.split('-')).col

In [184]: states = set(states)
          states = list(states)

In [185]: to_int = udf(lambda x: int(x))
          final = df_clean.groupBy(['userId','gender','location']).agg(countDistinct('artist').a
                                              avg('itemInSession').alias('avgItemInSession'),\
                                              avg('length').alias('avgLength'),\
                                              to_int((sparkMax('ts')-sparkMax('registration'))/8640
                                              countDistinct(when(col('level')=='paid',col('sessionI
                                              countDistinct(when(col('level')=='free',col('sessionI
                                              countDistinct('song').alias('songs'), \
                                              count(when(col('status') == 404,1)).alias('resp_404')
                                              count(when(col('status') == 307,1)).alias('resp_307')
                                              count(when(col('status') == 200,1)).alias('resp_200')
                                              count(when(col('page') =='Error', 1)).alias('page_err
                                              count(when(col('page') =='Help', 1)).alias('page_help
                                              count(when(col('page') =='Cancel', 1)).alias('page_ca
                                              count(when(col('page') =='Submit Downgrade', 1)).alia
                                              count(when(col('page') =='Thumbs Down', 1)).alias('pa
                                              count(when(col('page') =='Home', 1)).alias('page_home
                                              count(when(col('page') =='Downgrade', 1)).alias('page
                                              count(when(col('page') =='Roll Advert', 1)).alias('pa
                                              count(when(col('page') =='Logout', 1)).alias('page_lo
                                              count(when(col('page') =='Save Settings', 1)).alias('
                                              count(when(col('page') =='About', 1)).alias('page_abo
                                              count(when(col('page') =='Settings', 1)).alias('page_
                                              count(when(col('page') =='Add to Playlist', 1)).alias
                                              count(when(col('page') =='Add Friend', 1)).alias('pag
                                              count(when(col('page') =='Thumbs Up', 1)).alias('page
                                              count(when(col('page') =='Upgrade', 1)).alias('page_u
                                              count(when(col('page') =='Submit Upgrade', 1)).alias(
                                              count(when(col("churn")>0,True)).alias('churn'))\
                                      .sort('userId').collect()
          df_final = spark.createDataFrame(final)

In [186]: isMale = udf(lambda x: 1 if x == 'M' else 0)
          isFemale = udf(lambda x: 1 if x == 'F' else 0)
          df_final = df_final.withColumn('Male', isMale('gender').cast(IntegerType())).withColum

In [187]: def is_in_state(state):
              def check(x):
                  if state in x.split(', ',1)[-1].split('-'):
                      return 1
                  else:
                      return 0
```

```
            return udf(check, IntegerType() )
        for state in states:
            df_final = df_final.withColumn(state, is_in_state(state)('location'))
```

```
In [188]: df_final = df_final.drop('location')
```

```
In [189]: df_final = df_final.withColumn("id", df_final["userId"].cast('int')).drop('userId')
```

```
In [190]: df_final = df_final.withColumn("active", df_final["activeDuration"].cast('int')).drop(
```

```
In [191]: df_final.show(5)
```

```
+-------+------------------+------------------+-------------+-------------+-----+--------+------
|artists|  avgItemInSession|         avgLength|paid_sessions|free_sessions|songs|resp_404|resp_3
+-------+------------------+------------------+-------------+-------------+-----+--------+------
|    565|146.23899371069183|247.94408991084703|            6|            0|  629|       0|
|   1705| 89.31487243310517|250.88659828113387|           29|            8| 2302|       3|     2
|    125|28.513368983957218|263.71234699248123|            0|            4|  129|       1|
|    184|  83.4770642201835|254.15342615384608|            4|            0|  193|       0|
|     50| 40.65384615384615|241.30233960784318|            0|            2|   51|       0|
+-------+------------------+------------------+-------------+-------------+-----+--------+------
only showing top 5 rows
```

```
In [192]: df_final.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df_final
```

```
+-------+----------------+---------+-------------+-------------+-----+--------+--------+--------
|artists|avgItemInSession|avgLength|paid_sessions|free_sessions|songs|resp_404|resp_307|resp_200
+-------+----------------+---------+-------------+-------------+-----+--------+--------+--------
|      0|               0|        0|            0|            0|    0|       0|       0|       0
+-------+----------------+---------+-------------+-------------+-----+--------+--------+--------
```

5 Modeling

Split the full dataset into train, test, and validation sets. Test out several of the machine learning methods you learned. Evaluate the accuracy of the various models, tuning parameters as necessary. Determine your winning model based on test accuracy and report results on the validation set. Since the churned users are a fairly small subset, I suggest using F1 score as the metric to optimize.

```
In [194]: df_ready = df_final
```

```
In [195]: X = df_ready.drop('churn')
          y = df_ready.select('churn')
```

```
In [196]: train, test = df_ready.randomSplit([0.7, 0.3], seed=0)
```

```
In [197]: features_col = [col for col in X.columns]
          def build_model(classifier, param):
              assembler = VectorAssembler(inputCols=features_col, outputCol="features", handleIn
              normalizer = Normalizer(inputCol="features", outputCol="normFeatures")
              pipeline = Pipeline(stages=[assembler,normalizer,classifier ])

              model = CrossValidator(
                  estimator=pipeline,
                  estimatorParamMaps=param,
                  evaluator=MulticlassClassificationEvaluator(labelCol='churn', metricName='f1')
                  numFolds=5,
              )
              return model

In [198]: classifier = RandomForestClassifier(featuresCol="normFeatures", labelCol="churn")
          classifier_param = ParamGridBuilder().build()
          classifier_model = build_model(classifier, classifier_param)

In [199]: classifier_fit_model = classifier_model.fit(train)

In [200]: pred = classifier_fit_model.transform(test)

In [201]: pred.printSchema()

root
 |-- artists: long (nullable = true)
 |-- avgItemInSession: double (nullable = true)
 |-- avgLength: double (nullable = true)
 |-- paid_sessions: long (nullable = true)
 |-- free_sessions: long (nullable = true)
 |-- songs: long (nullable = true)
 |-- resp_404: long (nullable = true)
 |-- resp_307: long (nullable = true)
 |-- resp_200: long (nullable = true)
 |-- page_error: long (nullable = true)
 |-- page_help: long (nullable = true)
 |-- page_cancel: long (nullable = true)
 |-- page_submitDowngrade: long (nullable = true)
 |-- page_thumpDown: long (nullable = true)
 |-- page_home: long (nullable = true)
 |-- page_downgrade: long (nullable = true)
 |-- page_rollAdvert: long (nullable = true)
 |-- page_logout: long (nullable = true)
 |-- page_saveSettings: long (nullable = true)
 |-- page_about: long (nullable = true)
 |-- page_settings: long (nullable = true)
 |-- page_addtoPlaylist: long (nullable = true)
 |-- page_addFriend: long (nullable = true)
 |-- page_thumpsUp: long (nullable = true)
```

```
|-- page_upgrade: long (nullable = true)
|-- page_submitUpgrade: long (nullable = true)
|-- churn: long (nullable = true)
|-- Male: integer (nullable = true)
|-- Femal: integer (nullable = true)
|-- KY: integer (nullable = true)
|-- RI: integer (nullable = true)
|-- MO: integer (nullable = true)
|-- NJ: integer (nullable = true)
|-- WA: integer (nullable = true)
|-- OR: integer (nullable = true)
|-- MA: integer (nullable = true)
|-- CA: integer (nullable = true)
|-- WI: integer (nullable = true)
|-- UT: integer (nullable = true)
|-- NY: integer (nullable = true)
|-- WV: integer (nullable = true)
|-- AL: integer (nullable = true)
|-- KS: integer (nullable = true)
|-- NV: integer (nullable = true)
|-- MT: integer (nullable = true)
|-- IA: integer (nullable = true)
|-- OH: integer (nullable = true)
|-- NE: integer (nullable = true)
|-- GA: integer (nullable = true)
|-- TN: integer (nullable = true)
|-- AR: integer (nullable = true)
|-- LA: integer (nullable = true)
|-- SC: integer (nullable = true)
|-- CT: integer (nullable = true)
|-- PA: integer (nullable = true)
|-- CO: integer (nullable = true)
|-- IN: integer (nullable = true)
|-- DC: integer (nullable = true)
|-- MI: integer (nullable = true)
|-- MN: integer (nullable = true)
|-- FL: integer (nullable = true)
|-- AZ: integer (nullable = true)
|-- OK: integer (nullable = true)
|-- AK: integer (nullable = true)
|-- DE: integer (nullable = true)
|-- id: integer (nullable = true)
|-- TX: integer (nullable = true)
|-- MD: integer (nullable = true)
|-- IL: integer (nullable = true)
|-- MS: integer (nullable = true)
|-- VA: integer (nullable = true)
|-- NC: integer (nullable = true)
```

```
 |-- NH: integer (nullable = true)
 |-- active: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- normFeatures: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
In [202]: evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="ch
          f1_score = evaluator.evaluate(pred, {evaluator.metricName: "f1"})
          print("f1: {}".format(f1_score))
```

```
f1: 0.9678544494720964
```

6 Final Steps

Clean up your code, adding comments and renaming variables to make the code easier to read and maintain. Refer to the Spark Project Overview page and Data Scientist Capstone Project Rubric to make sure you are including all components of the capstone project and meet all expectations. Remember, this includes thorough documentation in a README file in a Github repository, as well as a web app or blog post.

```
In [203]: def create_session():
              '''
                  creates a new spark session locally
                  input:()
                  output:(
                      spark: a pointer to the newly created spark session
                  )
              '''
              spark = SparkSession.builder \
                  .master("local") \
                  .appName("Sparkify") \
                  .getOrCreate()
              spark.conf.set('spark.executor.memory', '32gb')
              spark.conf.set('spark.driver.memory','32gb')
              return spark
```

```
In [204]: def load_clean_data(spark, file):
              '''
                  creates a new spark session locally
                  input:(
                      spark: The spark session,
                      file: String - The path to the file containing data
                  )
```

```
            output:(
                df_clean: DataFrame - Dataframe contining loaded data after removing null
                          values
        '''
        sparkify_datail = file
        df = spark.read.json(sparkify_data)
        df.persist()
        #drop rows with null values
        df_clean = df.dropna(how='any',subset=['userId','sessionId'])
        df_clean = df_clean.filter(df["userId"] != "")
        return df_clean
```

In [208]:
```
def is_in_state(state):
    def check(x):
        if state in x.split(', ',1)[-1].split('-'):
            return 1
        else:
            return 0
    return udf(check, IntegerType() )


def extract_features(spark, df_clean):
    '''
        Extract featured appropriate for training the model
        input:(
            spark: The spark session,
            df_clean: DataFrame - Dataframe containing clean data
        )
        output:(
            df_final: DataFrame - Dataframe contining the extracted features
    '''

    #add churn column
    check_churn = udf(lambda x: 1 if x == 'Cancellation Confirmation' else 0)
    df_clean = df_clean.withColumn('churn',check_churn(df_clean.page).cast(DoubleType(

    #create df_final with aggregate features
    to_int = udf(lambda x: int(x))
    final = df_clean.groupBy(['userId','gender','location']).agg(countDistinct('artist
                                    avg('itemInSession').alias('avgItemInSession'),\
                                    avg('length').alias('avgLength'),\
                                    to_int((sparkMax('ts')-sparkMax('registration'))/8640
                                    countDistinct(when(col('level')=='paid',col('sessionI
                                    countDistinct(when(col('level')=='free',col('sessionI
                                    countDistinct('song').alias('songs'), \
                                    count(when(col('status') == 404,1)).alias('resp_404')
                                    count(when(col('status') == 307,1)).alias('resp_307')
                                    count(when(col('status') == 200,1)).alias('resp_200')
```

```
                                    count(when(col('page') =='Error', 1)).alias('page_err
                                    count(when(col('page') =='Help', 1)).alias('page_help
                                    count(when(col('page') =='Cancel', 1)).alias('page_ca
                                    count(when(col('page') =='Submit Downgrade', 1)).alia
                                    count(when(col('page') =='Thumbs Down', 1)).alias('pa
                                    count(when(col('page') =='Home', 1)).alias('page_home
                                    count(when(col('page') =='Downgrade', 1)).alias('page
                                    count(when(col('page') =='Roll Advert', 1)).alias('pa
                                    count(when(col('page') =='Logout', 1)).alias('page_lo
                                    count(when(col('page') =='Save Settings', 1)).alias('
                                    count(when(col('page') =='About', 1)).alias('page_abo
                                    count(when(col('page') =='Settings', 1)).alias('page_
                                    count(when(col('page') =='Add to Playlist', 1)).alias
                                    count(when(col('page') =='Add Friend', 1)).alias('pag
                                    count(when(col('page') =='Thumbs Up', 1)).alias('page
                                    count(when(col('page') =='Upgrade', 1)).alias('page_u
                                    count(when(col('page') =='Submit Upgrade', 1)).alias(
                                    count(when(col("churn")>0,True)).alias('churn'))\
                            .sort('userId').collect()
df_final = spark.createDataFrame(final)

#create dummy columns for the gender column Male or Female
isMale = udf(lambda x: 1 if x == 'M' else 0)
isFemale = udf(lambda x: 1 if x == 'F' else 0)
df_final = df_final.withColumn('Male', isMale('gender').cast(IntegerType())).withC


#create dummy columns for the state of the user
#read the list of states
states = df_clean.select("location").distinct().rdd.flatMap(lambda x: list(x)).\
                                        flatMap(lambda loc:[loc.split(', ',1)[-1]])
                                        flatMap(lambda state: state.split('-')).col
states = set(states)
states = list(states)

for state in states:
    df_final = df_final.withColumn(state, is_in_state(state)('location'))
df_final = df_final.drop('location')

#convert userId column to int
df_final = df_final.withColumn("id", df_final["userId"].cast('int')).drop('userId'

#convert activeDuration column to int
df_final = df_final.withColumn("active", df_final["activeDuration"].cast('int')).d

return df_final
```

```
In [209]: def build_model(classifier, param,features_col):
              '''
                  Builds the model used for training the data
                  input:(
                      classifier: The classifier used for training the model,
                      params: Parameters to be used for training the model as returned by Param
                  )
                  output:(
                      model: The model to be used for training the data
              '''
              assembler = VectorAssembler(inputCols=features_col, outputCol="features", handleIn
              normalizer = Normalizer(inputCol="features", outputCol="normFeatures")
              pipeline = Pipeline(stages=[assembler,normalizer,classifier ])

              model = CrossValidator(
                  estimator=pipeline,
                  estimatorParamMaps=param,
                  evaluator=MulticlassClassificationEvaluator(labelCol='churn', metricName='f1')
                  numFolds=5,
              )
              return model

In [213]: def train_model(df_ready):
              '''
                  Trains the model
                  input:(
                      df_ready: DataFrame - The dataframe containinf features to train the model
                  )
                  output:(
                      classifier_fit_model: The model after training
              '''
              X = df_ready.drop('churn')
              y = df_ready.select('churn')
              train, test = df_ready.randomSplit([0.7, 0.3], seed=0)
              features_col = [col for col in X.columns]
              classifier = RandomForestClassifier(featuresCol="normFeatures", labelCol="churn")
              classifier_param = ParamGridBuilder().build()
              classifier_model = build_model(classifier, classifier_param, features_col)
              classifier_fit_model = classifier_model.fit(train)
              return classifier_fit_model

In [214]: def test_model(fit_model):
              '''
                  Test and evaluate the model using f1_score
                  input:(
                      fit_model: The trained model
                  )
                  output:(
```

```
            f1_score: float - The f1-score calculated after testing the model
        '''
        pred = fit_model.transform(test)
        evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol
        f1_score = evaluator.evaluate(pred, {evaluator.metricName: "f1"})

        return f1_score
```

In [215]:
```
spark = create_session()
sparkify_data = 'mini_sparkify_event_data.json'
df_clean = load_clean_data(spark, sparkify_data)
df_ready = extract_features(spark, df_clean)
classifier_fit_model = train_model(df_ready)
f1_score = test_model(classifier_fit_model)
print("f1: {}".format(f1_score))
```

```
f1: 0.9678544494720964
```

In [ ]: