Sparkify

January 1, 2023

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 [1]:
```
# import libraries
from pyspark.sql import SparkSession, Window
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType,NumericType,LongType,IntegerType
from pyspark.sql.functions import desc,asc,sum,mean,avg,max
import pyspark.sql.functions as F
from pyspark.sql.functions import col, row_number
import numpy as np
from sklearn.metrics import f1_score,accuracy_score
from pyspark.sql.functions import col, row_number, when
import seaborn as sns
from pyspark.ml.feature import StandardScaler, VectorAssembler
import matplotlib.pyplot as plt
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression, RandomForestClassifier, GBTCla
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.feature import CountVectorizer, IDF, Normalizer, StandardScaler, StringI
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
import time
from pyspark.ml.stat import Correlation
import pandas as pd
```

In [2]:
```
# create a Spark session
spark = SparkSession.builder.appName('Sparkify').getOrCreate()
spark.sparkContext.getConf().getAll()
```

Out[2]:
```
[('spark.rdd.compress', 'True'),
 ('spark.app.name', 'Sparkify'),
```

```
 ('spark.serializer.objectStreamReset', '100'),
 ('spark.master', 'local[*]'),
 ('spark.executor.id', 'driver'),
 ('spark.submit.deployMode', 'client'),
 ('spark.driver.host', '3867cc290bab'),
 ('spark.driver.port', '45475'),
 ('spark.app.id', 'local-1672563710498'),
 ('spark.ui.showConsoleProgress', 'true')]
```

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 [3]: event_data = spark.read.json('mini_sparkify_event_data.json')
```

```
In [4]: event_data.printSchema()
        event_data.count()
        print("the data subset has {} rows and {} columns ".format(event_data.count(), len(event
```

```
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)

the data subset has 286500 rows and 18 columns
```

```
In [5]: #Null/Empty Check on every column in the dataframe (COMMENT IN LATER)
        print({col:event_data.filter(event_data[col].isNull()).count() for col in event_data.col
        print({col:event_data.filter(event_data[col] == '').count() for col in event_data.column
        # in 8346 the userId is empty, and the following fields are null firstName, gender, last
        # verify assumption that this is a specific type of event
```

```
        #event_data.select("*").where(event_data.userId == '').take(1000)
        #event_data.filter((event_data.auth !='Logged Out') & (event_data.userId == '')).show()
        #if the user is 'Logged Out' or a 'Guest' the records have no relevant information for u
        event_data_cleaned = event_data.filter((event_data.auth != 'Logged Out') & (event_data.a
        print("The data subset has {} rows and {} columns ".format(event_data.count(), len(event
```

```
{'artist': 58392, 'auth': 0, 'firstName': 8346, 'gender': 8346, 'itemInSession': 0, 'lastName':
{'artist': 0, 'auth': 0, 'firstName': 0, 'gender': 0, 'itemInSession': 0, 'lastName': 0, 'length
The data subset has 286500 rows and 18 columns
```

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.

In [6]:
```
#explore all possible event types
event_data.select("page").dropDuplicates().sort("page").show(100,truncate= False)
```

```
+------------------------+
|page                    |
+------------------------+
|About                   |
|Add Friend              |
|Add to Playlist         |
|Cancel                  |
|Cancellation Confirmation|
|Downgrade               |
|Error                   |
|Help                    |
|Home                    |
|Login                   |
|Logout                  |
|NextSong                |
|Register                |
|Roll Advert             |
|Save Settings           |
|Settings                |
|Submit Downgrade        |
|Submit Registration     |
|Submit Upgrade          |
|Thumbs Down             |
|Thumbs Up               |
|Upgrade                 |
+------------------------+
```

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.

```
In [7]: #define churn
        is_churn = udf(lambda x: int(x=="Cancellation Confirmation"))
        event_data_cleaned = event_data_cleaned.withColumn("churn", is_churn("page").cast("integ
        #partitionBy userId and add st every event by a user is labeled with the churn flag of t
        event_data_cleaned = event_data_cleaned.withColumn("user_churn", max('churn').over(Windo
```

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.

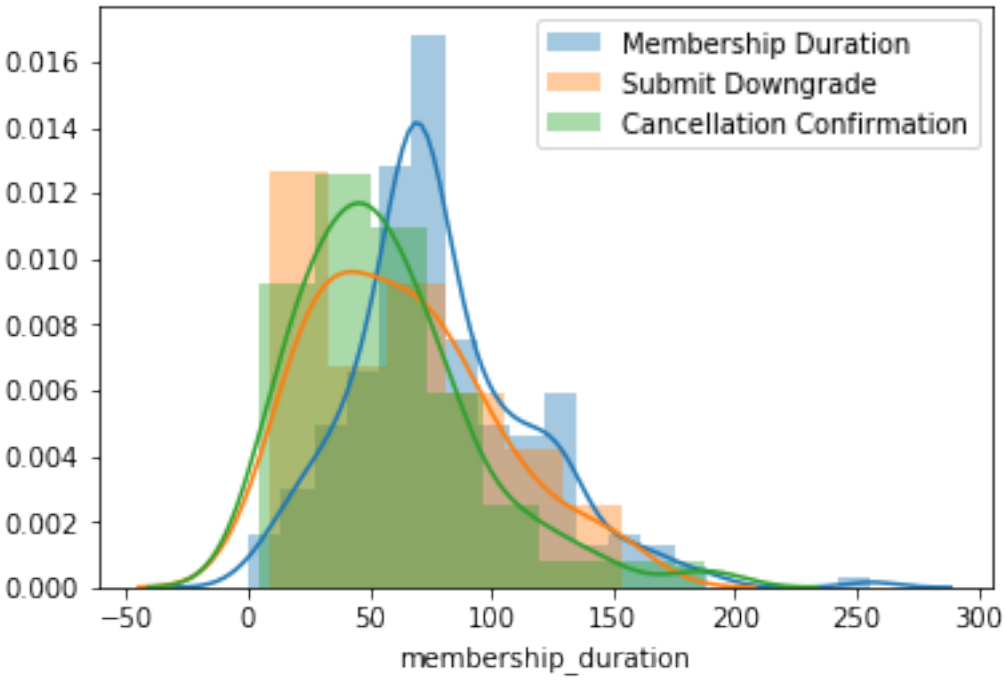
```
In [8]: #check number of unique users
        print('There are logs from {} different users in the dataset'.format((event_data_cleaned
        #use 'registration' and 'ts' timestamp to calculate a for how long a user is already a m
        get_membership_duration = udf(lambda registration,current: float((current-registration)/
        event_data_cleaned = event_data_cleaned.withColumn('membership_duration',
                                                           get_membership_duration(event_data_cl
```

```
There are logs from 225 different users in the dataset
```

```
In [9]: #check when (in the users lifecycle a Downgrade or App Cancelation occurs)
        df = event_data_cleaned.select(['membership_duration','userId']).groupBy('userId') \
        .max('membership_duration').withColumnRenamed('max(membership_duration)','membership_dur
        sns.distplot(df.membership_duration)

        df_downgrade = event_data_cleaned.select(['membership_duration','userId']).where(event_d
        .groupBy('userId').max('membership_duration').withColumnRenamed('max(membership_duration
        sns.distplot(df_downgrade.membership_duration)
        plt.legend(labels=['Membership Duration', 'Submit Downgrade'])

        df_cancel = event_data_cleaned.select(['membership_duration','userId']).where(event_data
        .groupBy('userId').max('membership_duration').withColumnRenamed('max(membership_duration
        sns.distplot(df_cancel.membership_duration)
        plt.legend(labels=['Membership Duration', 'Submit Downgrade','Cancellation Confirmation'

        event_data_cleaned.select(avg("membership_duration")).show()
```

```
+------------------------+
|avg(membership_duration)|
+------------------------+
```

```
|       64.32194755423255|
+------------------------+
```



There are activities from 225 users in the provided dataset. The average user is on the platform since 64.3 days. Cancellation of membership (leaving sparkify) or Downgrade from premium level are both left skewed (especially relative to the average membership duration) Generally speaking the chance that a new user leave Sparkify is highest in the first few weeks

```
In [10]: # split up into activity logs from churned users and users that are still active
         df_churned_users = event_data_cleaned.filter(event_data_cleaned.user_churn == 1).orderB
         df_active_users = event_data_cleaned.filter(event_data_cleaned.user_churn == 0).orderBy

         #Explore User Journey of first user in the set of churned users
         df_churned_users.select("page").where(event_data_cleaned.userId == '100009').show(187)

         def calculate_page_ratio(df,page_name):
             return df.filter(df.page == page_name).count()/df.count()

         # analyse all columns that might differ in these two groups (churned users might have s
         #print("[churned] error percentage: {}".format(calculate_page_ratio(df_churned_users,'E
         #print("[active] error percentage: {}".format(calculate_page_ratio(df_active_users,'Err
         #print("[churned] roll advert percentage: {}".format(calculate_page_ratio(df_churned_us
         #print("[active] roll advert percentage: {}".format(calculate_page_ratio(df_active_user
         #print("[churned] settings percentage: {}".format(calculate_page_ratio(df_churned_users
```

```
        #print("[active] settings percentage: {}".format(calculate_page_ratio(df_active_users,'
        #print("[churned] help percentage: {}".format(calculate_page_ratio(df_churned_users,'He
        #print("[active] help percentage: {}".format(calculate_page_ratio(df_active_users,'Help
        #print("[churned] add friend percentage: {}".format(calculate_page_ratio(df_churned_use
        #print("[active] add friend percentage: {}".format(calculate_page_ratio(df_active_users
        #print("[churned] thumbs down percentage: {}".format(calculate_page_ratio(df_churned_us
        #print("[active] thumbs down percentage: {}".format(calculate_page_ratio(df_active_user
        #print("[churned] thumbs up percentage: {}".format(calculate_page_ratio(df_churned_user
        #print("[active] thumbs up percentage: {}".format(calculate_page_ratio(df_active_users,
```

```
+---------------+
|           page|
+---------------+
|       NextSong|
|       NextSong|
|      Thumbs Up|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|    Roll Advert|
|    Roll Advert|
|       NextSong|
|    Roll Advert|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|        Upgrade|
|       NextSong|
|       NextSong|
|       Settings|
|       NextSong|
|       NextSong|
|         Logout|
|           Home|
|       NextSong|
|       NextSong|
|    Roll Advert|
|       NextSong|
|       NextSong|
|       NextSong|
```

```
|       NextSong|
|    Roll Advert|
|    Roll Advert|
|       NextSong|
|    Thumbs Down|
|       NextSong|
|       NextSong|
|       NextSong|
|         Logout|
|           Home|
|       NextSong|
|      Thumbs Up|
|       NextSong|
|       NextSong|
|Add to Playlist|
|       NextSong|
|       NextSong|
|         Logout|
|           Home|
|       NextSong|
|       NextSong|
|         Logout|
|           Home|
|       NextSong|
|       NextSong|
|         Logout|
|           Home|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|    Roll Advert|
|       NextSong|
|       NextSong|
|       NextSong|
|    Roll Advert|
|    Roll Advert|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|    Roll Advert|
|       NextSong|
|       NextSong|
```

| Thumbs Down|
| NextSong|
| Logout|
| Home|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| Help|
| Home|
| NextSong|
| NextSong|
| NextSong|
| Thumbs Down|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| Roll Advert|
| NextSong|
| Roll Advert|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| NextSong|
| Home|
| NextSong|
| Roll Advert|
| NextSong|
| Roll Advert|
| NextSong|
| Roll Advert|
| NextSong|
| NextSong|
| Add Friend|
| NextSong|
| Roll Advert|
| NextSong|

```
|Add to Playlist|
|       NextSong|
|       NextSong|
|       NextSong|
|      Thumbs Up|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|        Upgrade|
| Submit Upgrade|
|           Home|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|           Home|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       Settings|
|  Save Settings|
|       Settings|
|           Home|
|      Downgrade|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
|       NextSong|
```

```
|       NextSong|
|Add to Playlist|
|       NextSong|
|           Help|
|       NextSong|
|     Add Friend|
|     Add Friend|
|       NextSong|
+---------------+
only showing top 187 rows
```

A significant difference in the two groups can be observed in the number of ads ('Roll Advert') the users see, how the users rate the songs ('Thumps up/down'), how often they visit the 'Settings' page and the number of erros a user experiences. While the churned users saw more ads, rated the songs more negatively (more thumbs down and less thumbs up) and visited the Settings page more often than the active users the active users actually experienced more errors!

4 Feature Engineering

activity: number of total events of the user divided by his/her membership duration, aims to show how active the user is on the app

total_friends: number of friends the user has on the app

ads_frequency: how frequent the user sees ads in the app

level: the level of the user (premium vs. free) represented as a binary value

membership_duration: number of days past from the users registration to his/her last recorded action in the user log

gender: the gender of the user represented as a binary value

```
<li>rating: the difference of thumbs up and thumbs down issued by the user (the value is negativ
```

```
In [11]: df_friends = event_data_cleaned.where(event_data_cleaned.page == 'Add Friend') \
                                    .groupBy('userId').count().withColumnRenamed('count', 'total_fr
         event_data_cleaned = event_data_cleaned.join(df_friends,"userId","left")


         df_ads = event_data_cleaned.where(event_data_cleaned.page == 'Roll Advert') \
                                   .groupBy('userId').count().withColumnRenamed('count', 'ads_playe
         event_data_cleaned = event_data_cleaned.join(df_ads,"userId","left")


         df_thumbs_up = event_data_cleaned.where(event_data_cleaned.page == 'Thumbs Up') \
                                   .groupBy('userId').count().withColumnRenamed('count', 'thumbs_up
         event_data_cleaned = event_data_cleaned.join(df_thumbs_up,"userId","left")

         df_thumbs_down = event_data_cleaned.where(event_data_cleaned.page == 'Thumbs Down') \
                                   .groupBy('userId').count().withColumnRenamed('count', 'thumbs_do
```

```
        event_data_cleaned = event_data_cleaned.join(df_thumbs_down,"userId","left")

        df_total_events = event_data_cleaned.groupBy('userId').count().withColumnRenamed('count
        event_data_cleaned = event_data_cleaned.join(df_total_events,"userId","left")
```

In [12]:
```
        #fill the new numeric features with 0 (no recorded action of this type)
        event_data_cleaned = event_data_cleaned.fillna(0,subset = ['ads_played','total_friends'
        #show new features
        event_data_cleaned.select (['userId','ads_played','total_friends','thumbs_up','thumbs_d
```

```
+------+----------+-------------+---------+-----------+
|userId|ads_played|total_friends|thumbs_up|thumbs_down|
+------+----------+-------------+---------+-----------+
|100010|        52|            4|       17|          5|
|200002|         7|            4|       21|          6|
|   125|         1|            0|        0|          0|
|   124|         4|           74|      171|         41|
|    51|         0|           28|      100|         21|
|     7|        16|            1|        7|          1|
|    15|         1|           31|       81|         14|
|    54|        47|           33|      163|         29|
|   155|         8|           11|       58|          3|
|100014|         2|            6|       17|          3|
|   132|         2|           41|       96|         17|
|   154|        10|            3|       11|          0|
|   101|         8|           29|       86|         16|
|    11|        39|            6|       40|          9|
|   138|        17|           41|       95|         24|
|300017|        11|           63|      303|         28|
|100021|        30|            7|       11|          5|
|    29|        22|           47|      154|         22|
|    69|         3|           12|       72|          9|
|   112|        21|            7|        9|          3|
+------+----------+-------------+---------+-----------+
only showing top 20 rows
```

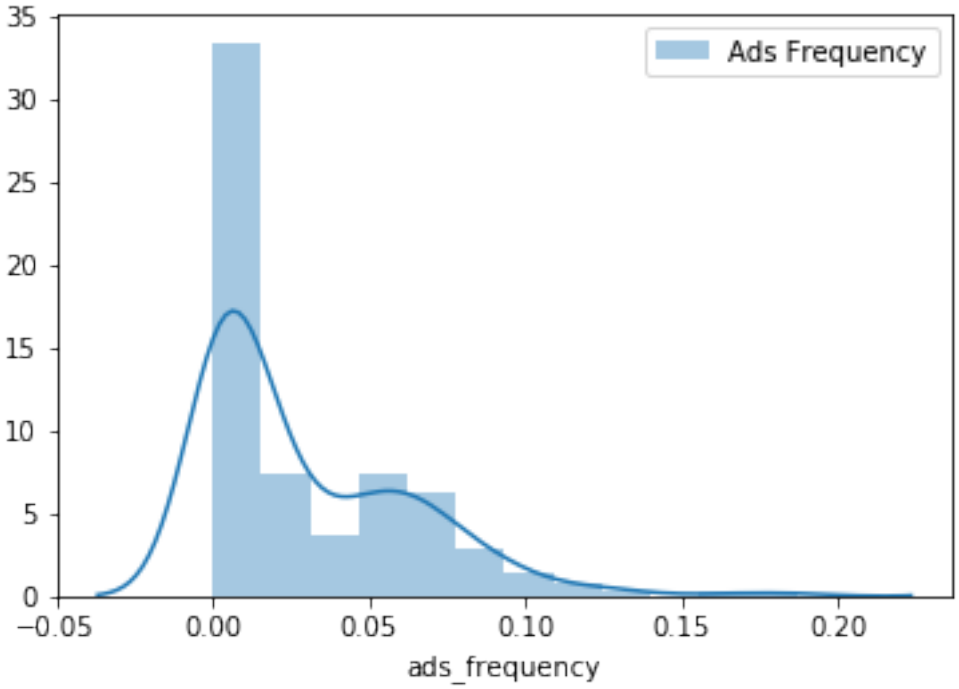
In [13]:
```
        #calculate frequency of ad impressions instead of total count
        event_data_cleaned = event_data_cleaned.withColumn('ads_frequency', \
                                                           (event_data_cleaned['ads_played'] / event
```

In [14]:
```
        #calculate the rating by subtracting the number of given thumbs_down from the number of
        event_data_cleaned = event_data_cleaned.withColumn('rating', \
                                                           (event_data_cleaned['thumbs_up'] - event_
```

In [15]:
```
        print(event_data_cleaned.count())
        event_data_cleaned.show()
```

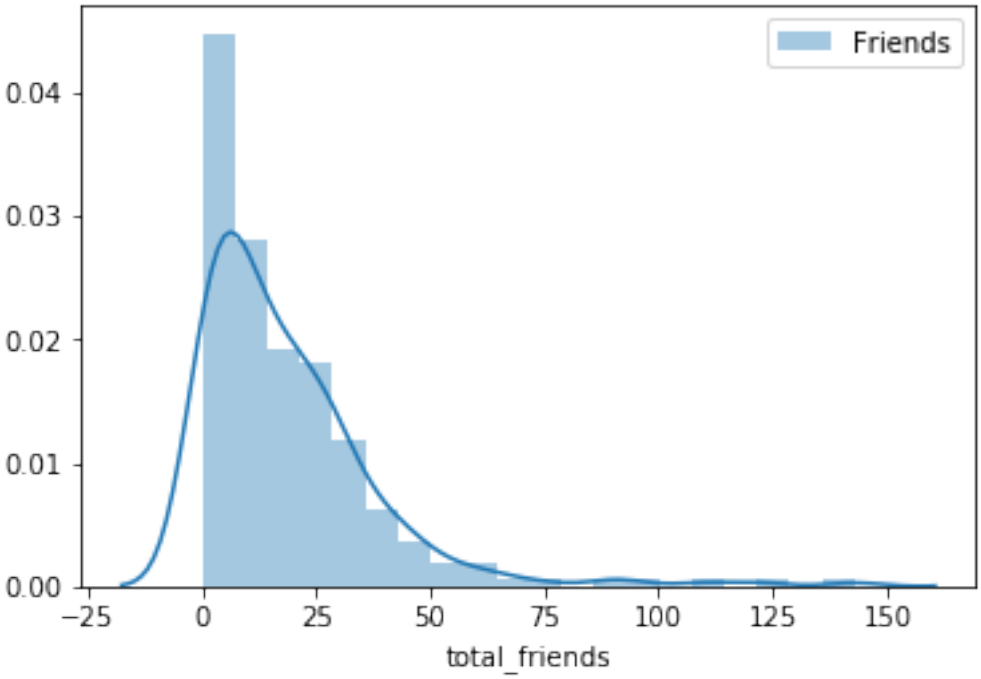
```
278154
+------+--------------------+---------+---------+------+-------------+---------+---------+-----+
|userId|              artist|     auth|firstName|gender|itemInSession| lastName|   length|level|
+------+--------------------+---------+---------+------+-------------+---------+---------+-----+
|100010|Sleeping With Sirens|Logged In| Darianna|     F|            0|Carpenter|202.97098| free|
|100010|Francesca Battist...|Logged In| Darianna|     F|            1|Carpenter|196.54485| free|
|100010|              Brutha|Logged In| Darianna|     F|            2|Carpenter|263.13098| free|
|100010|                null|Logged In| Darianna|     F|            3|Carpenter|     null| free|
|100010|         Josh Ritter|Logged In| Darianna|     F|            4|Carpenter|316.23791| free|
|100010|               LMFAO|Logged In| Darianna|     F|            5|Carpenter|183.74485| free|
|100010|         OneRepublic|Logged In| Darianna|     F|            6|Carpenter|224.67873| free|
|100010|       Dwight Yoakam|Logged In| Darianna|     F|            7|Carpenter| 239.3073| free|
|100010|                null|Logged In| Darianna|     F|            8|Carpenter|     null| free|
|100010|      The Chordettes|Logged In| Darianna|     F|            9|Carpenter|142.41914| free|
|100010|Coko featuring Ki...|Logged In| Darianna|     F|           10|Carpenter| 249.3122| free|
|100010|            The Cure|Logged In| Darianna|     F|           11|Carpenter| 52.27057| free|
|100010|                null|Logged In| Darianna|     F|           12|Carpenter|     null| free|
|100010|Kid Cudi Vs Crookers|Logged In| Darianna|     F|           13|Carpenter|162.97751| free|
|100010|                null|Logged In| Darianna|     F|           14|Carpenter|     null| free|
|100010|            Yeasayer|Logged In| Darianna|     F|           15|Carpenter|323.44771| free|
|100010|             Ben Lee|Logged In| Darianna|     F|           16|Carpenter|245.78567| free|
|100010|                null|Logged In| Darianna|     F|           17|Carpenter|     null| free|
|100010|  ? & The Mysterians|Logged In| Darianna|     F|           18|Carpenter|128.10404| free|
|100010|                null|Logged In| Darianna|     F|           19|Carpenter|     null| free|
+------+--------------------+---------+---------+------+-------------+---------+---------+-----+
only showing top 20 rows
```

```
In [16]: event_data_cleaned_pandas = event_data_cleaned.toPandas()

In [17]: event_data_cleaned_pandas = event_data_cleaned_pandas.drop_duplicates(subset=['userId']

In [18]: # check distribution of new columns
         sns.distplot(event_data_cleaned_pandas.ads_frequency)
         plt.legend(labels=['Ads Frequency'])

Out[18]: <matplotlib.legend.Legend at 0x7fb2b46cecc0>
```



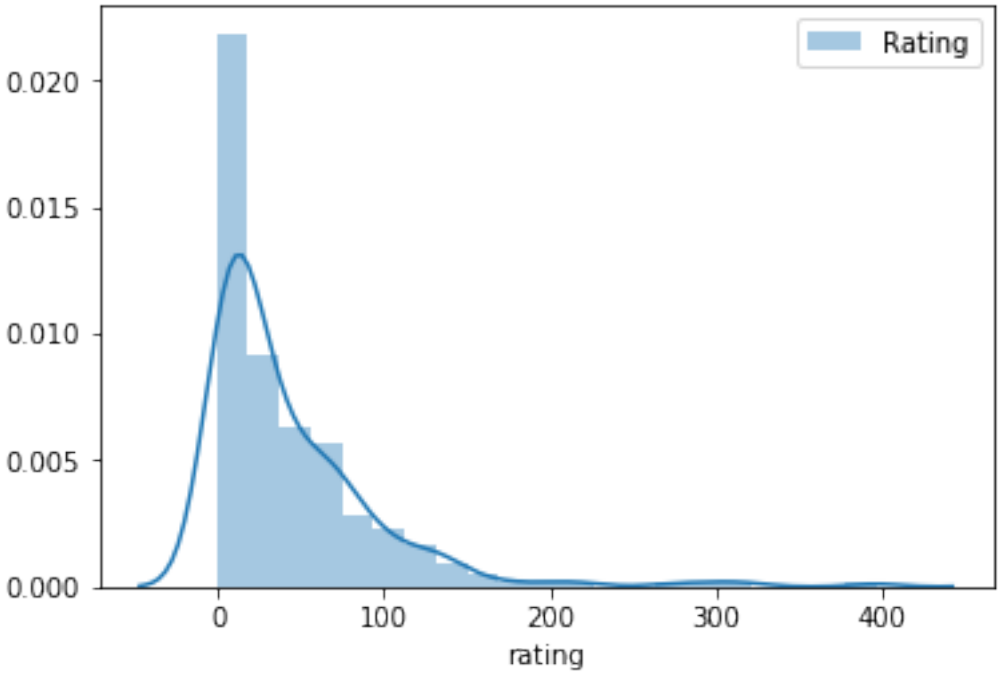
In [19]:
```
# check distribution of new columns
sns.distplot(event_data_cleaned_pandas.total_friends)
plt.legend(labels=['Friends'])
```

Out[19]: <matplotlib.legend.Legend at 0x7fb2e878b4e0>



In [20]:
```
# check distribution of new columns
sns.distplot(event_data_cleaned_pandas.rating)
plt.legend(labels=['Rating'])
```

Out[20]:
```
<matplotlib.legend.Legend at 0x7fb2b46ceb38>
```



In [21]:
```
#use entry with the highest membership duration = last user state before churning/not c
user_data_logs = event_data_cleaned.select('userId','gender','level','user_churn','memb

#https://stackoverflow.com/questions/48829993/groupby-column-and-filter-rows-with-maxim
user_data_final = user_data_logs.withColumn("row", row_number().over(Window.partitionBy
  .filter(col("row") == 1).drop("row")
```

In [22]:
```
user_data_final.write.json('sparkify_user_data3.json')
```

5 Modeling

In [23]:
```
user_data = spark.read.json('sparkify_user_data3.json')
```

In [24]:
```
user_data.show()
user_data = user_data.withColumn("gender", when(col("gender") == "M", 0).otherwise(when
```

```
        user_data = user_data.withColumn("level", when(col("level") == "free", 0).otherwise(whe
        user_data = user_data.drop('ts')
        user_data = user_data.select(*(col(c).cast("float").alias(c) for c in user_data.columns
        user_data = user_data.withColumnRenamed('user_churn','label')
        user_data = user_data.withColumn("label",user_data.label.cast('integer'))
        user_data = user_data.withColumn('activity',(user_data['total_events'] / user_data['mem
        user_data = user_data.fillna(0,subset = ['activity'])
        user_data.orderBy('activity').show()
```

```
+--------------------+------+-----+-------------------+------+------------+-------------+------
|       ads_frequency|gender|level|membership_duration|rating|total_events|total_friends|
+--------------------+------+-----+-------------------+------+------------+-------------+------
| 0.06907630522088354|     F| paid|              172.|    24|      1245.|         19.|1543459
|0.010787992495309569|     F| paid|              125.|    67|      2132.|         23.|1543348
|0.001538461538461...|     M| paid|               71.|    75|      1950.|         27.|1543190
| 0.08333333333333333|     F| free|              100.|     2|        48.|          7.|1543000
| 0.06666666666666667|     M| free|              101.|     0|        45.|          0.|1542806
|0.011283043197936816|     M| paid|               62.|   114|      3102.|         33.|1543592
|0.002683843263553...|     M| free|              120.|    52|      1863.|         26.|1543548
|0.012645348837209303|     F| paid|               80.|   202|      6880.|        143.|1543606
|0.003629764065335753|     F| paid|               13.|    27|      1102.|         12.|1539465
|0.010687022900763359|     F| paid|               42.|    43|      1310.|         25.|1541340
|0.028735632183908046|     M| paid|              113.|     7|       174.|          3.|1543306
|0.001273885350318...|     M| paid|               75.|    44|      1570.|         21.|1543606
| 0.01739788199697428|     F| paid|               78.|    55|      1322.|         23.|1542348
|0.005321722302854378|     M| paid|               95.|    69|      2067.|         28.|1543506
| 0.04891304347826087|     M| paid|               28.|     6|       552.|          9.|1540353
|0.024296675191815855|     F| paid|              135.|    22|       782.|         15.|1543022
|4.504504504504504...|     F| paid|              110.|    66|      2220.|         36.|1543009
| 0.06716417910447761|     F| free|               55.|    14|       268.|          0.|1542356
|0.036062378167641324|     M| paid|              116.|    21|      1026.|         24.|1542804
| 0.05714285714285714|     M| free|               99.|     2|        35.|          0.|1539362
+--------------------+------+-----+-------------------+------+------------+-------------+------
only showing top 20 rows
```

```
+-------------+------+-----+-------------------+------+------------+-------------+--------+-----
|ads_frequency|gender|level|membership_duration|rating|total_events|total_friends|  userId|label
+-------------+------+-----+-------------------+------+------------+-------------+--------+-----
|   0.16666667|   0.0|  0.0|                0.0|   0.0|         6.0|          0.0|   156.0|    0
|          0.0|   1.0|  0.0|               72.0|   0.0|         6.0|          0.0|   135.0|    0
|   0.09090909|   0.0|  0.0|               71.0|   0.0|        11.0|          0.0|   125.0|    1
|  0.057142857|   0.0|  0.0|               99.0|   2.0|        35.0|          0.0|300003.0|    0
|   0.06666667|   0.0|  0.0|              101.0|   0.0|        45.0|          0.0|    90.0|    0
|  0.083333336|   1.0|  0.0|              100.0|   2.0|        48.0|          7.0|    68.0|    0
|          0.1|   1.0|  0.0|               61.0|   3.0|        40.0|          3.0|    22.0|    0
|  0.057692308|   0.0|  0.0|               65.0|   2.0|        52.0|          0.0|   134.0|    0
|   0.03797468|   1.0|  0.0|               93.0|   1.0|        79.0|          2.0|   116.0|    0
|   0.08547009|   1.0|  0.0|              124.0|   5.0|       117.0|          3.0|    72.0|    0
```

```
|   0.18666667|   0.0|  0.0|               78.0|   1.0|        75.0|          0.0|100017.0|    1
|   0.04109589|   0.0|  0.0|               71.0|   2.0|        73.0|          0.0|    34.0|    0
|  0.022727273|   0.0|  0.0|               39.0|   2.0|        44.0|          1.0|   133.0|    0
|  0.040358745|   1.0|  0.0|              188.0|   6.0|       223.0|          4.0|   119.0|    0
|  0.032258064|   1.0|  1.0|               52.0|   2.0|        62.0|          1.0|   122.0|    1
|        0.072|   0.0|  0.0|               98.0|   3.0|       125.0|          4.0|   144.0|    0
|   0.05357143|   0.0|  0.0|               87.0|   1.0|       112.0|          0.0|    57.0|    0
|    0.0952381|   0.0|  0.0|               65.0|   2.0|        84.0|          3.0|200012.0|    0
|  0.083333336|   0.0|  0.0|               27.0|   5.0|        36.0|          0.0|100024.0|    1
|  0.013761468|   1.0|  1.0|              160.0|   5.0|       218.0|          1.0|100002.0|    0
+-------------+------+-----+-------------------+------+------------+-------------+--------+-----
only showing top 20 rows
```

```
In [25]: def train_model(data,classifier,params,feature_names):
             """ Fits classifier to given data using corss validation appraoch
             Args:
               data (DataFrame): data used to train the classifier
               classifier (pyspark.ml.classification): PySpark classifier object
               params (ParamGrid): parameter grid used in the cross validation approach fo fine
               feature_names (list): names of all features of the dataframe that should be used
             """
             assembler = VectorAssembler(inputCols = feature_names, outputCol='numerical_feature
             scaler = StandardScaler(inputCol="numerical_features", outputCol="features")
             pipeline = Pipeline(stages=[assembler, scaler, classifier])


             crossval= CrossValidator(estimator=pipeline,
                                      estimatorParamMaps=params,
                                      evaluator=BinaryClassificationEvaluator(),
                                      numFolds=3
                                      )
             start = time.time()
             # fit the cross validation linear support vector machine model
             model = crossval.fit(training)
             end = time.time()
             print(f'Model fitting took {end-start} seconds')
             return model

In [26]: def evaluate_model(results):
             """ Method for evaluation the results of any model. Calculates f1_score and accurac
             Args:
               results (DataFrame): dataframe containing the predictions of the model and the or
             """
             correct_predictions = (results.filter(results.label == results.prediction).count())
             total_predictions = (results.count())
             churn_predictions = (results.filter( '1.0' == results.prediction).count())
```

```
            results_pd = results.toPandas()
            print('f1_score: {}'.format(f1_score(results_pd.label.values,results_pd.prediction.
            print('accuracy: {}'.format(accuracy_score(results_pd.label.values,results_pd.predi
            print('predicted user churn ration: {}'.format(churn_predictions/total_predictions)
```

In [27]:
```
(training, test) = user_data.randomSplit([0.70, 0.30], seed=42)
#show distribution of user churn in test and training data to get an idea of the baseli
#baseling = performance of model thatl always guesses 0 = user does not churn
print(training.filter(col('label') == 0).count()/training.count())
print(test.filter(col('label') == 0).count()/test.count())
```

```
0.7884615384615384
0.7246376811594203
```

In [28]:
```
user_data.count()
```

Out[28]: 225

In [29]:
```
# define names of columnes the model should use for learning
feature_names = ['gender','ads_frequency','level','membership_duration','rating','total
```

In [30]:
```
#Baseline
base_model = test.withColumn('prediction', F.lit(0))
evaluate_model(base_model)
```

```
f1_score: 0.0
accuracy: 0.7246376811594203
predicted user churn ration: 0.0
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: UndefinedMetricWa
  'precision', 'predicted', average, warn_for)
```

In [31]:
```
#Logistic Regression
lr_model = LogisticRegression()
params =ParamGridBuilder()\
.addGrid(lr_model.maxIter, [10,50,100,150]) \
.build()

lr_model_result = train_model(training,lr_model,params,feature_names)
evaluate_model(lr_model_result.transform(test))
```

```
Model fitting took 47.73698568344116 seconds
f1_score: 0.2727272727272727
accuracy: 0.7681159420289855
predicted user churn ration: 0.043478260869565216
```

```
In [32]: #Gradient Boost Classifier
         gbc_model = GBTClassifier()
         params = ParamGridBuilder() \
                           .addGrid(gbc_model.maxIter ,[20, 100]) \
                           .addGrid(gbc_model.maxDepth, [5, 10]) \
                           .build()


         gbc_model_result = train_model(training,gbc_model,params,feature_names)
         evaluate_model(gbc_model_result.transform(test))

Model fitting took 602.1364877223969 seconds
f1_score: 0.39999999999999997
accuracy: 0.6956521739130435
predicted user churn ration: 0.2318840579710145


In [33]: #Random Forst Classifier
         rf_model = RandomForestClassifier()
         params = ParamGridBuilder() \
                           .addGrid(rf_model.numTrees ,[20, 40]) \
                           .addGrid(rf_model.maxDepth, [5, 10]) \
                           .build()
         rf_model_result = train_model(training,rf_model,params,feature_names)
         evaluate_model(rf_model_result.transform(test))

Model fitting took 24.033241510391235 seconds
f1_score: 0.5185185185185185
accuracy: 0.8115942028985508
predicted user churn ration: 0.11594202898550725
```

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 [ ]:
```