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 [111]:

```
# import libraries
import numpy as np
import pandas as pd
import time
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.proportion import proportions_ztest

%matplotlib inline

from pyspark.sql import SparkSession, Window
from pyspark.sql.functions import avg, stddev, split, udf, isnull, first
, col, format_number, rand
from pyspark.sql.functions import min as fmin
from pyspark.sql.functions import max as fmax
from pyspark.sql.types import IntegerType, FloatType

import re
import copy
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.proportion import proportions_ztest

%matplotlib inline

from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression, RandomForestCl
assifier, GBTClassifier, DecisionTreeClassifier, NaiveBayes
from pyspark.ml.feature import StandardScaler, VectorAssembler
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder


import re
import copy

import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.proportion import proportions_ztest

%matplotlib inline
```

In [8]:

```
sns.set(style="whitegrid")
```

In [9]:

```
# create a Spark session
spark_1 = SparkSession.builder.appName('Sparkify_local').getOrCreate()
```

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.

Reading the data

In [10]:
```
df1 = spark_1.read.json('mini_sparkify_event_data.json')
df1.show(1)
```

```
+--------------+---------+---------+------+-------------+--------+-----
----+-----+---------------+------+--------+-------------+---------+----
-----+------+-------------+--------------------+------+
|        artist|     auth|firstName|gender|itemInSession|lastName|   le
ngth|level|       location|method|    page| registration|sessionId|
song|status|           ts|           userAgent|userId|
+--------------+---------+---------+------+-------------+--------+-----
----+-----+---------------+------+--------+-------------+---------+----
-----+------+-------------+--------------------+------+
|Martha Tilston|Logged In|    Colin|     M|           50| Freeman|277.8
9016| paid|Bakersfield, CA|   PUT|NextSong|1538173362000|       29|Rock
pools|   200|1538352117000|Mozilla/5.0 (Wind...|    30|
+--------------+---------+---------+------+-------------+--------+-----
----+-----+---------------+------+--------+-------------+---------+----
-----+------+-------------+--------------------+------+
only showing top 1 row
```

In [11]:
```
df1.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)
```

```
In [12]: df1.count()
```

```
Out[12]: 286500
```

```
In [13]: df1.describe('userId').show()
```

```
+-------+------------------+
|summary|            userId|
+-------+------------------+
|  count|            286500|
|   mean| 59682.02278593872|
| stddev|109091.9499991047|
|    min|                  |
|    max|                99|
+-------+------------------+
```

```
In [14]: df1.describe('sessionId').show()
```

```
+-------+------------------+
|summary|         sessionId|
+-------+------------------+
|  count|            286500|
|   mean|1041.526554973822|
| stddev|726.7762634630741|
|    min|                 1|
|    max|              2474|
+-------+------------------+
```

```
In [15]: df1.persist()
```

```
Out[15]: DataFrame[artist: string, auth: string, firstName: string, gender: stri
ng, itemInSession: bigint, lastName: string, length: double, level: str
ing, location: string, method: string, page: string, registration: bigi
nt, sessionId: bigint, song: string, status: bigint, ts: bigint, userAg
ent: string, userId: string]
```

to aid our analysis, we will need to clean our data so it is more consistent for our use.

Cleaning the Data

As the first step we will remove the missing values and NaNs

```
In [16]: df1.filter(isnull(df1['sessionId'])).count()
```

```
Out[16]: 0
```

```
In [17]: df1.filter(isnull(df1['userId'])).count()
```

```
Out[17]: 0
```

In [18]:
```
df1.filter(df1['userId']=='').count()
```

Out[18]:
```
8346
```

In [19]:
```
df1.filter(df1['sessionId']=='').count()
```

Out[19]:
```
0
```

In [20]:
```
df1_clean = df1.filter(df1['userId']!='')
```

In [21]:
```
''' One noticeable thing we see in the data given to us is
that the time is not in a readable format so we can try and get it
into the form that is human readable
'''


time_old = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).strftime("%Y-%m-%d %H:%M:%S"))

df_clean = df1_clean.withColumn("time", time_old(df1_clean.ts))
```

In [22]:
```
df_clean.take(1)
```

Out[22]:
```
[Row(artist='Martha Tilston', auth='Logged In', firstName='Colin', gend
er='M', itemInSession=50, lastName='Freeman', length=277.89016, level
='paid', location='Bakersfield, CA', method='PUT', page='NextSong', reg
istration=1538173362000, sessionId=29, song='Rockpools', status=200, ts
=1538352117000, userAgent='Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0)
Gecko/20100101 Firefox/31.0', userId='30', time='2018-10-01 00:01:57')]
```

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 [23]:

```
# lets explore the unique values of the 'page' column
df_clean.select("page").dropDuplicates().show()
```

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

In [24]:

```
df_clean.filter(df_clean.page=="Cancellation Confirmation").select("userID").dropDuplicates().show(15)
```

```
+------+
|userID|
+------+
|   125|
|    51|
|    54|
|100014|
|   101|
|    29|
|100021|
|    87|
|    73|
|     3|
|    28|
|100022|
|100025|
|300007|
|100006|
+------+
only showing top 15 rows
```

In [25]:
```
df_clean.select(["userId", "page", "time", "level", "song", "sessionId"
]).where(df_clean.userId == "30").sort("time").show(100)
```

```
+------+---------------+-------------------+-----+--------------------+
---------+
|userId|           page|               time|level|                song|
sessionId|
+------+---------------+-------------------+-----+--------------------+
---------+
|    30|       NextSong|2018-10-01 00:01:57| paid|           Rockpools|
29|
|    30|       NextSong|2018-10-01 00:06:34| paid|   Time For Miracles|
29|
|    30|       NextSong|2018-10-01 00:11:16| paid|Harder Better Fas...|
29|
|    30|       NextSong|2018-10-01 00:14:59| paid|Passengers (Old A...|
29|
|    30|Add to Playlist|2018-10-01 00:15:05| paid|                null|
29|
|    30|       NextSong|2018-10-01 00:18:04| paid|          Fuck Kitty|
29|
|    30|       NextSong|2018-10-01 00:20:18| paid|                Jade|
29|
|    30|       NextSong|2018-10-01 00:24:01| paid|   So-Called Friends|
29|
|    30|       NextSong|2018-10-01 00:28:07| paid|           Represent|
29|
|    30|       NextSong|2018-10-01 00:31:49| paid|           Here I Am|
29|
|    30|       NextSong|2018-10-01 00:35:32| paid|Rebirthing (Album...|
29|
|    30|       NextSong|2018-10-01 00:39:25| paid|Dog Days Are Over...|
29|
|    30|       NextSong|2018-10-01 00:43:04| paid|Tomorrow Is A Lon...|
29|
|    30|       NextSong|2018-10-01 00:46:46| paid|    Halloween Spooks|
29|
|    30|       NextSong|2018-10-01 00:49:05| paid|            Stronger|
29|
|    30|       NextSong|2018-10-01 00:54:16| paid|   Dis Iz Brick City|
29|
|    30|       NextSong|2018-10-01 00:57:53| paid|          Move Along|
29|
|    30|       NextSong|2018-10-01 01:01:51| paid|           Manhattan|
29|
|    30|       NextSong|2018-10-01 01:05:15| paid|                Undo|
29|
|    30|       NextSong|2018-10-01 01:11:03| paid|     The Big Gundown|
29|
|    30|       NextSong|2018-10-01 01:15:23| paid|          Black Bird|
29|
|    30|    Thumbs Down|2018-10-01 01:15:24| paid|                null|
29|
|    30|       NextSong|2018-10-01 01:18:27| paid|              Nausea|
29|
|    30|       NextSong|2018-10-01 01:21:43| paid|           Matricide|
29|
|    30|       NextSong|2018-10-01 01:27:03| paid|             Valerie|
29|
|    30|       NextSong|2018-10-01 01:30:52| paid|           Margarita|
29|
|    30|       NextSong|2018-10-01 01:34:08| paid|   Le Jardin d'Hiver|
29|
|    30|      Thumbs Up|2018-10-01 01:34:09| paid|                null|
29|
|    30|       NextSong|2018-10-01 01:39:50| paid|Soon As I Get Hom...|
29|
|    30|      Thumbs Up|2018-10-01 01:39:51| paid|                null|
29|
|    30|       NextSong|2018-10-01 01:45:14| paid|    Vamos a la Playa|
29|
|    30|       NextSong|2018-10-01 01:48:52| paid|            Perfecta|
29|
|    30|       NextSong|2018-10-01 01:52:41| paid|             Requiem|
29|
|    30|       NextSong|2018-10-01 01:57:34| paid|     Who Can Compare|
29|
|    30|       NextSong|2018-10-01 02:02:00| paid|240 Years Before ...|
29|
|    30|       NextSong|2018-10-01 02:25:20| paid|         Rosa Pastel|
29|
|    30|    Roll Advert|2018-10-01 02:25:37| paid|                null|
29|
|    30|       NextSong|2018-10-01 02:28:25| paid|I'm Ready (Album ...|
29|
|    30|       NextSong|2018-10-01 02:31:47| paid|    No Other Saviour|
29|
|    30|       NextSong|2018-10-01 02:36:12| paid|               Hints|
29|
|    30|       NextSong|2018-10-01 02:39:00| paid|              Yellow|
29|
|    30|Add to Playlist|2018-10-01 02:39:35| paid|                null|
29|
|    30|       NextSong|2018-10-06 07:23:50| paid|Kill The Director...|
264|
|    30|     Add Friend|2018-10-06 07:23:51| paid|                null|
264|
|    30|           Home|2018-10-07 21:37:06| paid|                null|
532|
|    30|       NextSong|2018-10-07 21:37:20| paid|         Third Party|
532|
|    30|       NextSong|2018-10-07 21:43:11| paid|        It's Working|
532|
|    30|       NextSong|2018-10-07 21:47:17| paid|I'll Remember Apr...|
532|
|    30|       NextSong|2018-10-07 21:50:34| paid|  Atmosphere Station|
532|
|    30|       NextSong|2018-10-07 21:53:45| paid|        There_ There|
532|
|    30|       NextSong|2018-10-07 21:59:08| paid|'Til We Die (Albu...|
532|
|    30|       NextSong|2018-10-07 22:04:53| paid|Lies (Album Version)|
532|
|    30|       NextSong|2018-10-07 22:07:52| paid|The Ballad of Mic...|
532|
|    30|       NextSong|2018-10-07 22:11:42| paid|               Gears|
532|
|    30|       NextSong|2018-10-07 22:15:41| paid|          These Eyes|
532|
|    30|           Help|2018-10-07 22:15:49| paid|                null|
532|
|    30|           Home|2018-10-07 22:16:33| paid|                null|
532|
|    30|       NextSong|2018-10-07 22:20:12| paid|            Stronger|
532|
|    30|       NextSong|2018-10-07 22:25:23| paid| This Is Such A Pity|
532|
|    30|       NextSong|2018-10-07 22:28:47| paid|Strut (1993 Digit...|
532|
|    30|       NextSong|2018-10-07 22:32:46| paid|               Fader|
532|
|    30|       NextSong|2018-10-07 22:35:58| paid|  Se Quiere_ Se Mata|
532|
|    30|       NextSong|2018-10-07 22:39:36| paid|           Somewhere|
532|
|    30|       NextSong|2018-10-07 22:42:25| paid|              Canada|
532|
|    30|       NextSong|2018-10-07 22:46:21| paid|    Dream On Dreamer|
532|
|    30|       NextSong|2018-10-07 22:50:21| paid|HÃ  Â¡ Uma MÃ  Âºsi...
|      532|
|    30|       NextSong|2018-10-07 22:53:57| paid| Money Ain't A Thang|
532|
|    30|       NextSong|2018-10-07 22:58:11| paid|     Howlin  For You|
532|
|    30|       NextSong|2018-10-07 23:01:22| paid|           Fireflies|
532|
|    30|       NextSong|2018-10-07 23:05:07| paid|        Stadium Love|
532|
|    30|      Downgrade|2018-10-07 23:05:26| paid|                null|
532|
|    30|       NextSong|2018-10-07 23:09:19| paid|       Again & Again|
532|
|    30|       NextSong|2018-10-07 23:12:04| paid|The Trooper (1998...|
532|
|    30|       NextSong|2018-10-07 23:16:16| paid| Get Off My Elevator|
532|
|    30|       NextSong|2018-10-07 23:18:34| paid|        Speed Trials|
532|
|    30|       NextSong|2018-10-07 23:21:35| paid|             Invalid|
532|
|    30|       NextSong|2018-10-07 23:25:28| paid|           Frenchy s|
532|
|    30|       NextSong|2018-10-07 23:28:22| paid|   Between Two Lungs|
532|
|    30|       NextSong|2018-10-07 23:32:31| paid|              Hiding|
532|
|    30|       NextSong|2018-10-07 23:36:17| paid|Alone Again (Natu...|
532|
|    30|       NextSong|2018-10-07 23:39:55| paid|        You Found Me|
532|
|    30|       NextSong|2018-10-07 23:43:55| paid|            Our Song|
532|
|    30|       NextSong|2018-10-07 23:47:16| paid|           Rock Hard|
532|
|    30|       NextSong|2018-10-07 23:50:30| paid| Waiting For Tonight|
532|
|    30|       NextSong|2018-10-07 23:54:35| paid|Pioneer to The Falls|
532|
|    30|       NextSong|2018-10-08 00:00:16| paid|      You're The One|
532|
|    30|       NextSong|2018-10-08 00:04:15| paid|     Trash And Ready|
532|
|    30|       NextSong|2018-10-08 00:07:08| paid|      Keep On Movin'|
532|
|    30|       NextSong|2018-10-08 00:13:09| paid|    Quase Um Segundo|
532|
|    30|       NextSong|2018-10-08 00:16:03| paid|Costumbres Argent...|
532|
|    30|       NextSong|2018-10-08 00:19:20| paid|Nine Times Out Of...|
532|
|    30|       NextSong|2018-10-08 00:21:30| paid|           Bass Solo|
532|
|    30|       NextSong|2018-10-08 00:26:27| paid|  The Memory Remains|
532|
|    30|       NextSong|2018-10-08 00:31:06| paid|               Clock|
532|
|    30|       NextSong|2018-10-08 00:35:23| paid|I Don t Wanna Go ...|
532|
|    30|       NextSong|2018-10-08 00:37:07| paid|       She's So Cold|
532|
|    30|       NextSong|2018-10-08 00:41:20| paid|  Drops In The River|
532|
|    30|       NextSong|2018-10-08 00:45:33| paid|          Didgeridoo|
532|
|    30|       NextSong|2018-10-08 00:52:43| paid|   Sex_ Love & Money|
532|
|    30|       NextSong|2018-10-08 00:56:53| paid|      Rayando el sol|
532|
+------+---------------+-------------------+-----+--------------------+
---------+
only showing top 100 rows
```

In [26]:

```
# one way to go around is using Spark SQL to create temporary view and p
erform querying

df_clean.createOrReplaceTempView("sparkify_clean")
```

In [27]:

```
# lets take a peek into different types of authorization

spark_1.sql('''
        SELECT DISTINCT(auth)
        FROM sparkify_clean
''').show()
```

```
+---------+
|     auth|
+---------+
|Cancelled|
|Logged In|
+---------+
```

In [29]:

```
spark = spark_1
```

In [30]:

```
# now let us look for user count in each category

spark.sql('''
        SELECT auth,COUNT(DISTINCT userId) AS user_counts
        FROM sparkify_clean
        GROUP BY auth
        ORDER BY user_counts DESC
''').show()
```

```
+---------+-----------+
|     auth|user_counts|
+---------+-----------+
|Logged In|        225|
|Cancelled|         52|
+---------+-----------+
```

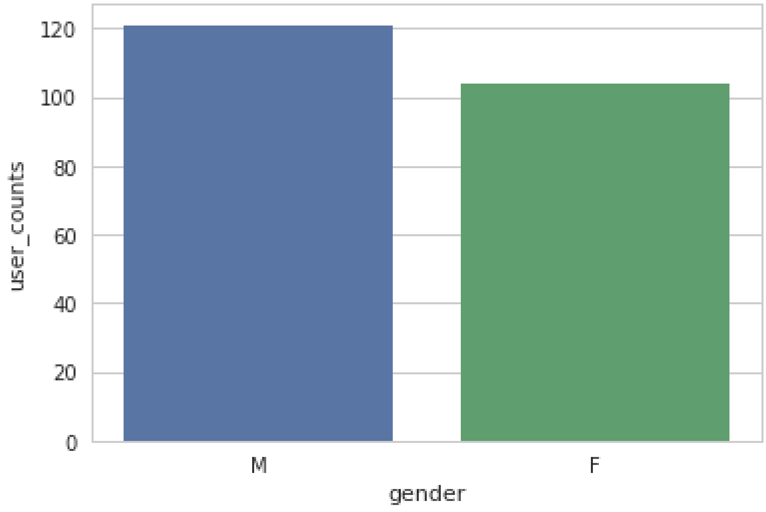
In [31]:

```
# One possible avenue of exploration is how is the gender distributed in the data
gender_data = spark.sql('''
        SELECT gender,COUNT(DISTINCT userId) AS user_counts
        FROM sparkify_clean
        GROUP BY gender
        ORDER BY user_counts DESC
''')
gender_data.show()
```

```
+------+-----------+
|gender|user_counts|
+------+-----------+
|     M|        121|
|     F|        104|
+------+-----------+
```

We can see that our data consists of 121 Males and 104 Females. Let us try and visualize it.

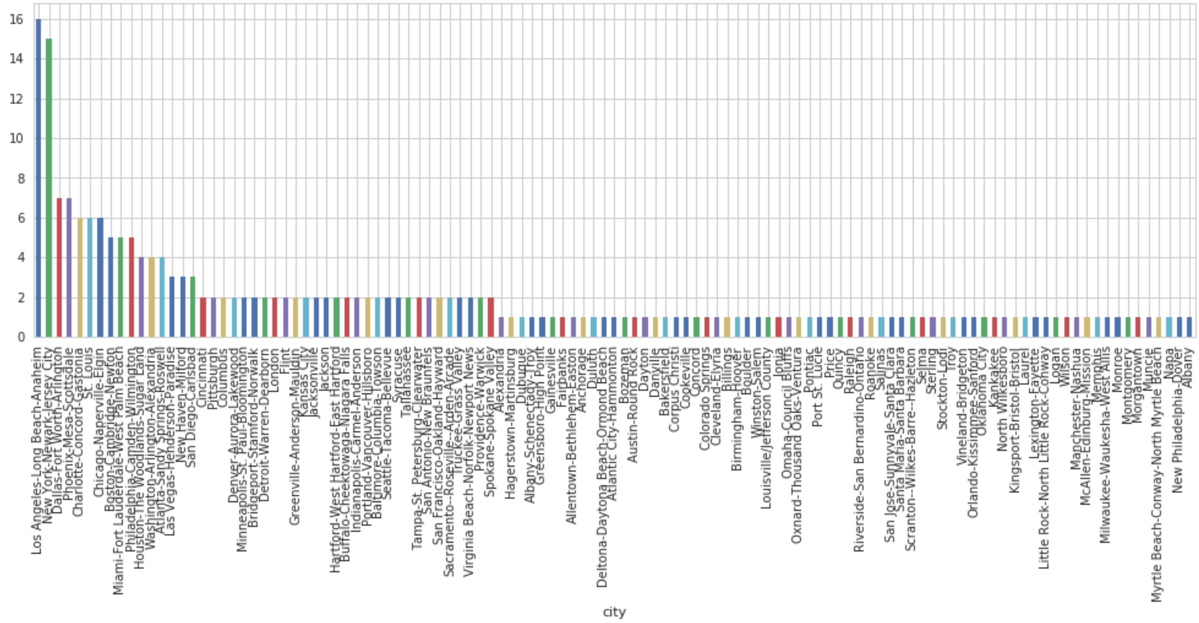
In [32]:
```
sns.barplot(x='gender',y='user_counts',data=gender_data.toPandas());
```



In [33]:
```
# Let us look into how the data is divided by location
location_data = spark.sql('''
        SELECT location,COUNT(DISTINCT userId) AS user_counts
        FROM sparkify_clean
        GROUP BY location
        ORDER BY user_counts DESC
''').toPandas()
```

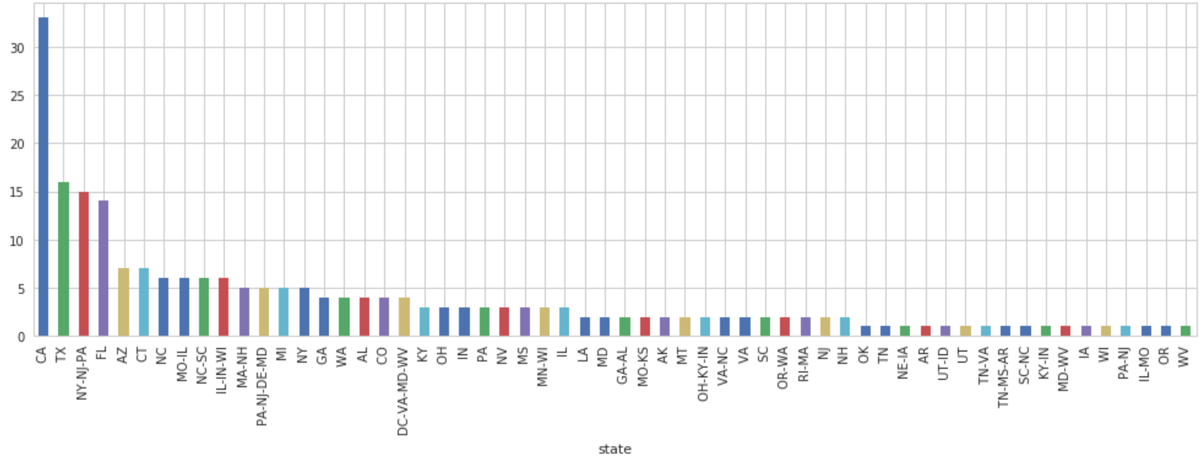
In [34]:
```
#split city and state
location_data = location_data.join(location_data['location'].str.split(
',',expand=True).rename(columns={0:'city',1:'state'})).drop('location',a
xis=1)
```

In [35]:
```
location_data.groupby('city')['user_counts'].sum().sort_values(ascending
=False).plot(kind='bar',figsize=(17,5));
```



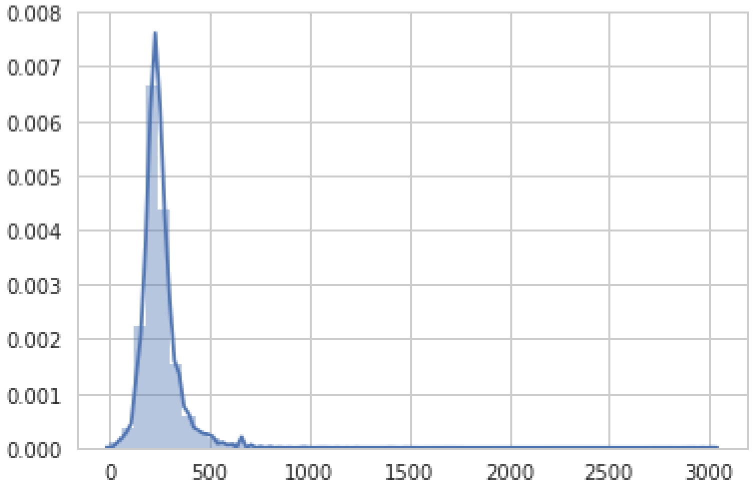
In [36]:

```
location_data.groupby('state')['user_counts'].sum().sort_values(ascending=False).plot(kind='bar',figsize=(17,5));
```



In [37]:

```
# One other way to look at data is from 'length' point of view

length_data = spark.sql('''
        SELECT Length
        FROM sparkify_clean
''')

sns.distplot(length_data.toPandas().dropna());
```



We can observe that mostly all of the lengths are between 0 - 500.

In [38]:
```
# lets look at the data from level perspective

spark.sql('''
        SELECT level, count(Distinct userID) as usr_counts
        FROM sparkify_clean
        GROUP BY level
        ORDER BY usr_counts DESC
''').show()
```

```
+-----+----------+
|level|usr_counts|
+-----+----------+
| free|       195|
| paid|       165|
+-----+----------+
```

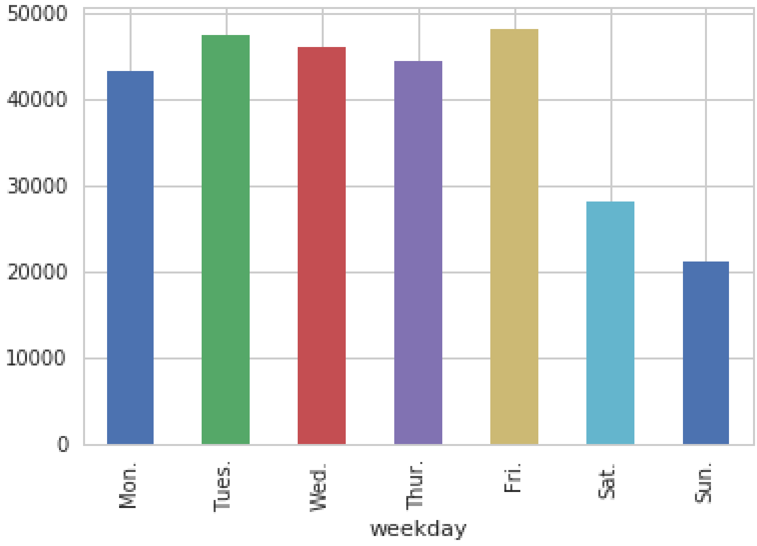
The data split shows that there are 2 distinct levels with 195 'free' and 165 'paid' accounts. We can also deduce that there are 135 users that have changed their subscription levels.

In [39]:
```
# Data split by page

spark.sql('''
        select page, count(userID) as usr_counts
        from sparkify_clean
        group by page
        order by usr_counts DESC
''').show()
```

```
+--------------------+----------+
|                page|usr_counts|
+--------------------+----------+
|            NextSong|    228108|
|           Thumbs Up|     12551|
|                Home|     10082|
|     Add to Playlist|      6526|
|          Add Friend|      4277|
|         Roll Advert|      3933|
|              Logout|      3226|
|         Thumbs Down|      2546|
|           Downgrade|      2055|
|            Settings|      1514|
|                Help|      1454|
|             Upgrade|       499|
|               About|       495|
|       Save Settings|       310|
|               Error|       252|
|      Submit Upgrade|       159|
|    Submit Downgrade|        63|
|              Cancel|        52|
|Cancellation Conf...|        52|
+--------------------+----------+
```

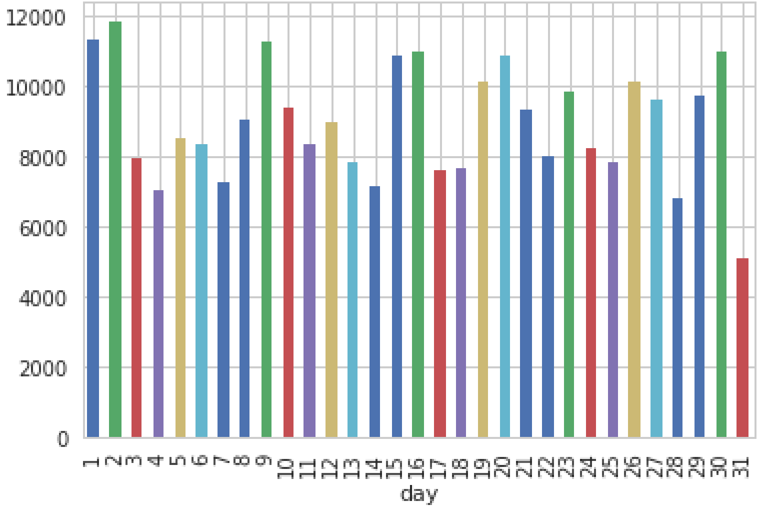
In [45]:

```
# Data cut by 'time'

time_data = spark.sql('''
        SELECT time,userId
        FROM sparkify_clean
''').toPandas()

time_data['time'] = pd.to_datetime(time_data['time'])

weekday_dict = {0:'Mon.',1:'Tues.',2:'Wed.',3:'Thur.',4:'Fri.',5:'Sat.',
6:'Sun.'}
time_data['weekday'] = time_data['time'].dt.weekday.map(weekday_dict)
time_data['day'] = time_data['time'].dt.day
time_data['hour'] = time_data['time'].dt.hour

time_data.groupby('weekday')['userId'].count().loc[list(weekday_dict.val
ues())].plot(kind='bar');
```

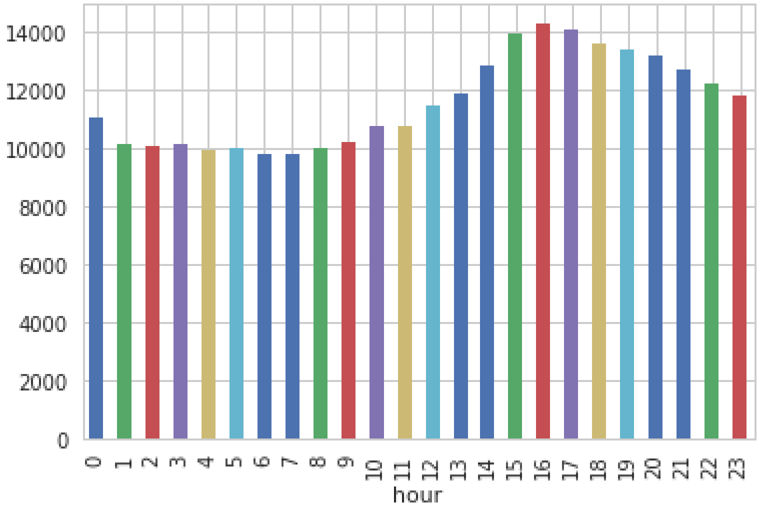


In [46]:

```
time_data.groupby('day')['userId'].count().plot(kind='bar');
```



In [47]:
```
time_data.groupby('hour')['userId'].count().plot(kind='bar');
```



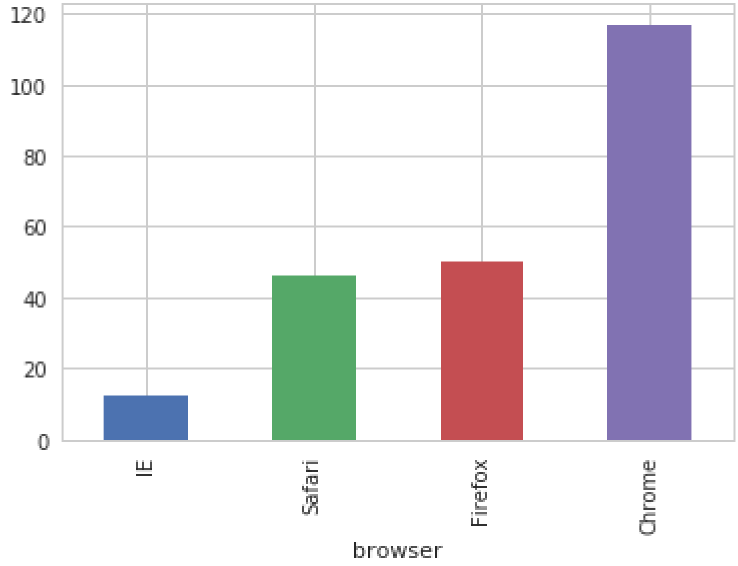
We can deduce that the user behave in some sort of periodic pattern. It can be seen that users prefer to use it late in the day like after 2 or 3 o'clock in noon which can be justified considering that people start getting bored of their jobs. Also it is observed that the user prefer to use sparkify on weekdays over weekends, which also can be explained as people like to spend time with friends and family over the weekends.

In [ ]:
```

```

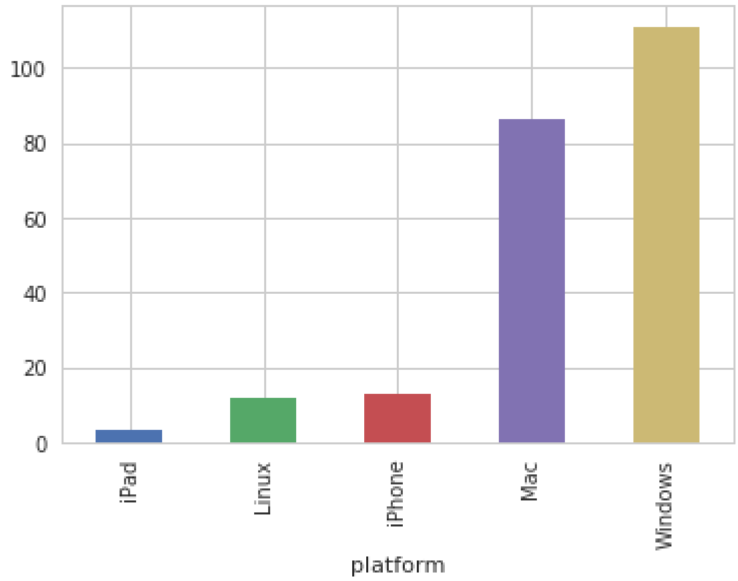
In [41]:
```
# Data cut by userAgent

userAgent_count = spark.sql('''
        select userAgent, count(Distinct userID) as usr_counts
        from sparkify_clean
        group by userAgent
        order by usr_counts DESC
''').toPandas()
```

In [42]:
```
def browser(x):
    if 'Firefox' in x:
        return 'Firefox'
    elif 'Safari' in x:
        if 'Chrome' in x:
            return 'Chrome'
        else:
            return 'Safari'
    elif 'Trident' in x:
        return 'IE'
    else:
        return np.NaN
```

In [43]:
```
userAgent_count['browser'] = userAgent_count['userAgent'].apply(browser)

dict_platform = {'compatible': 'Windows',  'iPad': 'iPad',  'iPhone': 'i
Phone',
            'Macintosh': 'Mac',  'Windows NT 5.1': 'Windows','Windows NT
 6.0': 'Windows', 'Windows NT 6.1': 'Windows',
            'Windows NT 6.2': 'Windows',  'Windows NT 6.3': 'Windows', 'X1
1': 'Linux'}

userAgent_count['platform'] = userAgent_count['userAgent'].str.extract(r
'\(([^\)]*)\)')[0].str.split(';').str[0].map(dict_platform)

userAgent_count.groupby('browser')['usr_counts'].sum().sort_values().plo
t(kind='bar');
```



In [44]:
```
userAgent_count.groupby('platform')['usr_counts'].sum().sort_values().pl
ot(kind='bar');
```



```
In [ ]:
```

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.

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 [48]: df_clean.select("page").dropDuplicates().show()
```

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

Lets us see for the customers who have selected the cancellation confirmation

In [50]:

```
df_clean.filter(df_clean.page=="Cancellation Confirmation").select("userId").dropDuplicates().show(10)
```

```
+------+
|userId|
+------+
|   125|
|    51|
|    54|
|100014|
|   101|
|    29|
|100021|
|    87|
|    73|
|     3|
+------+
only showing top 10 rows
```

In [59]:

```
# add time to see the time clear
get_time = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).str
ftime("%Y-%m-%d %H:%M:%S"))
df_clean = df_clean.withColumn("time", get_time(df_clean.ts))

df_clean.select(["userId", "page", "time", "level", "song", "sessionId"
]).where(df_clean.userId == "87").sort("time").show()
```

```
+------+-----------+-------------------+-----+--------------------+----
-----+
|userId|       page|               time|level|                song|sess
ionId|
+------+-----------+-------------------+-----+--------------------+----
-----+
|    87|       Home|2018-10-01 06:47:15| free|                null|
86|
|    87|   NextSong|2018-10-01 06:49:00| free|          Crotchopus|
86|
|    87|   NextSong|2018-10-01 07:03:31| free|  We Belong Together|
86|
|    87|   NextSong|2018-10-01 07:07:58| free|         Basic Space|
86|
|    87|   NextSong|2018-10-01 07:11:06| free|          Concertina|
86|
|    87|   NextSong|2018-10-01 07:15:23| free|Crazy Little Thin...|
86|
|    87|   NextSong|2018-10-01 07:18:33| free|        Highway Song|
86|
|    87|Roll Advert|2018-10-01 07:18:44| free|                null|
86|
|    87|   NextSong|2018-10-01 07:29:05| free|Lipstick Traces (...|
86|
|    87|   NextSong|2018-10-01 07:31:31| free|Here's Your Revol...|
86|
|    87|   NextSong|2018-10-01 07:35:22| free|    Stack Shot Billy|
86|
|    87|   NextSong|2018-10-01 07:38:43| free| I CAN'T GET STARTED|
86|
|    87|   NextSong|2018-10-01 07:47:00| free|Queen Of Memphis ...|
86|
|    87|   NextSong|2018-10-01 07:50:19| free|        Up Up & Away|
86|
|    87|   NextSong|2018-10-01 07:54:06| free|          In One Ear|
86|
|    87|   NextSong|2018-10-01 07:58:07| free|     End Of The Road|
86|
|    87|   NextSong|2018-10-01 08:01:26| free|       Sex Out South|
86|
|    87|   NextSong|2018-10-01 08:06:56| free|  Don't Shake It Off|
86|
|    87|   NextSong|2018-10-01 08:08:10| free|            Karibien|
86|
|    87|   NextSong|2018-10-01 15:12:31| free|            Ã  Â  poca|
273|
+------+-----------+-------------------+-----+--------------------+----
-----+
only showing top 20 rows
```

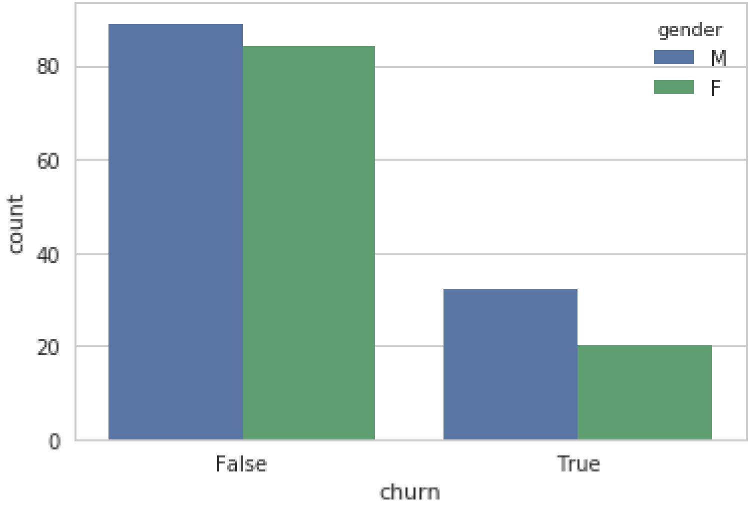
In [60]:
```
# Adding churn column

churn_usrs = df_clean.filter(df_clean.page=="Cancellation Confirmation")
.select("userId").dropDuplicates()
churn_usrs_list = [(row['userId']) for row in churn_usrs.collect()]
df_churn = df_clean.withColumn("churn", df_clean.userId.isin(churn_usrs_
list))
```

In [61]:
```
# lets start to slice the data using this data which includes churn

df_churn.dropDuplicates(["userId", "gender"]).groupby(["churn", "gender"
]).count().sort("churn").show()
```

```
+-----+------+-----+
|churn|gender|count|
+-----+------+-----+
|false|     M|   89|
|false|     F|   84|
| true|     F|   20|
| true|     M|   32|
+-----+------+-----+
```

In [62]:
```
df_pd = df_churn.dropDuplicates(["userId", "gender"]).groupby(["churn",
"gender"]).count().sort("churn").toPandas()
sns.barplot(x='churn', y='count', hue='gender', data=df_pd)
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2881ab5630>
```

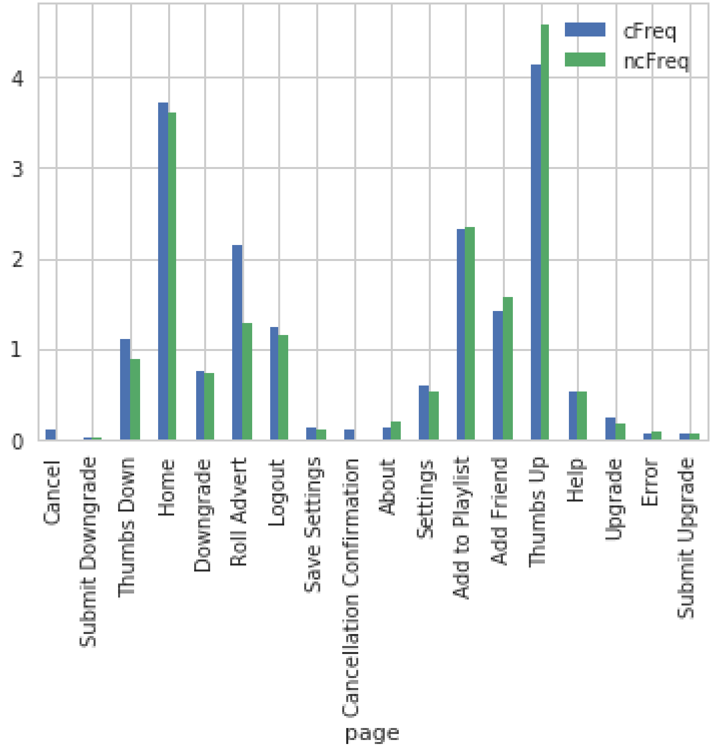


In [63]:

```
# 'page' use distribution between churn user and normal user

churn_count = df_churn.filter(df_churn.churn==1).count()
no_churn_count = df_churn.filter(df_churn.churn==0).count()

df_temp1 = df_churn.filter(df_churn.churn==1).filter(df_churn.page!="NextSong").groupby(["page"]).count()
df_temp1 = df_temp1.withColumn("cFreq", df_temp1["count"]/(churn_count/100)).sort("page")

df_temp2 = df_churn.filter(df_churn.churn==0).filter(df_churn.page!="NextSong").groupby(["page"]).count()
df_temp2 = df_temp2.withColumn("ncFreq", df_temp2["count"]/(no_churn_count/100)).sort("page")

df_pd = df_temp1.join(df_temp2, "page", "outer").drop("count").fillna(0).toPandas()
df_pd.plot.bar("page")
```

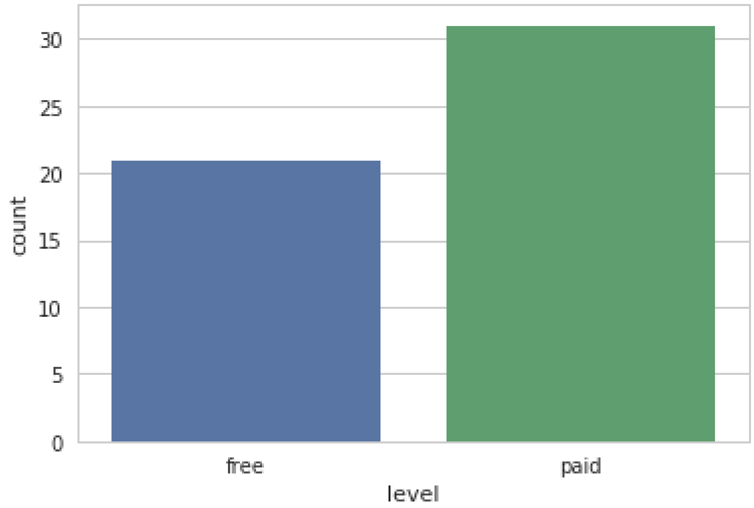
Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f288184fdd8>
```



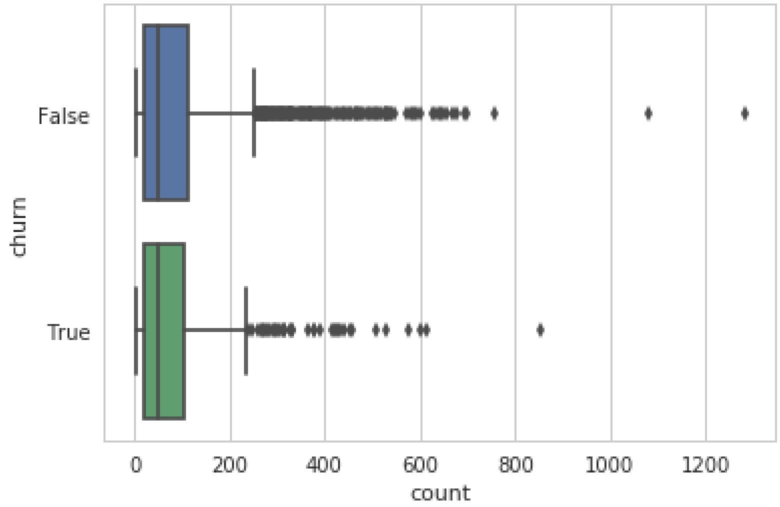
```
In [64]: # Viewing the level when the user churned


         df_pd = df_churn.filter(df_churn.page=="Cancellation Confirmation").grou
         pby("level").count().toPandas()
         sns.barplot(x="level", y="count", data=df_pd)
```

```
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2870d88c50>
```



```
In [67]: # Slicing the data by distribution in operations in each session

         df_pd = df_churn.groupby("churn", "userId", "sessionId").count().toPanda
         s()
         sns.boxplot(x='count', y='churn', orient="h", data=df_pd)
```

```
Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x7f28712aada0>
```



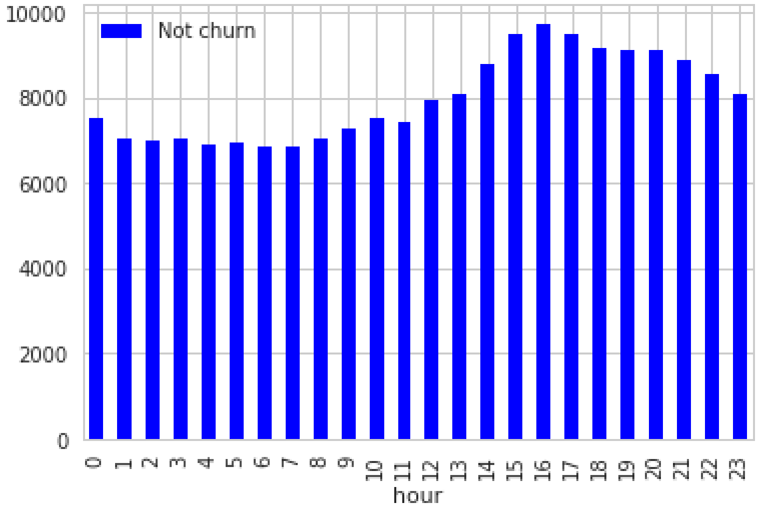
In [65]:

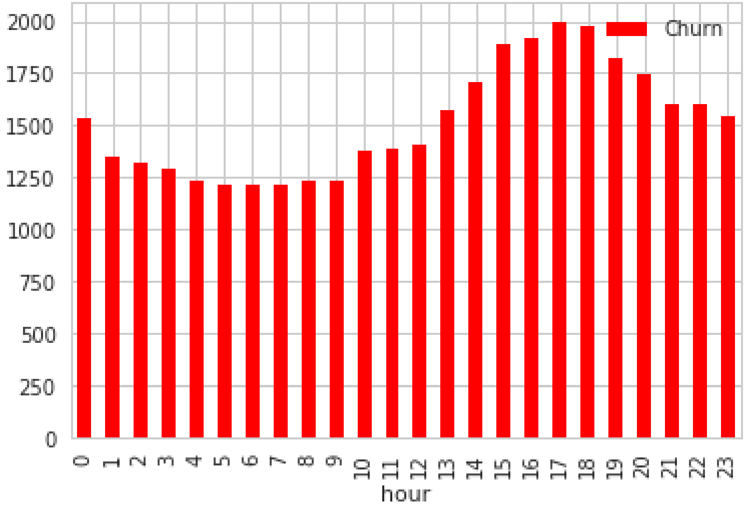
```
# the time span of the log

df_churn.select("time").describe().show()
```

```
+-------+-------------------+
|summary|               time|
+-------+-------------------+
|  count|             278154|
|   mean|               null|
| stddev|               null|
|    min|2018-10-01 00:01:57|
|    max|2018-12-03 01:11:16|
+-------+-------------------+
```

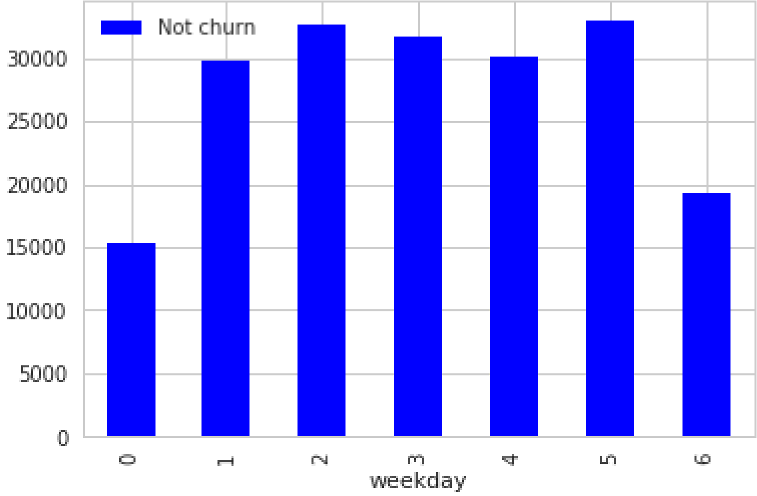
In [69]:

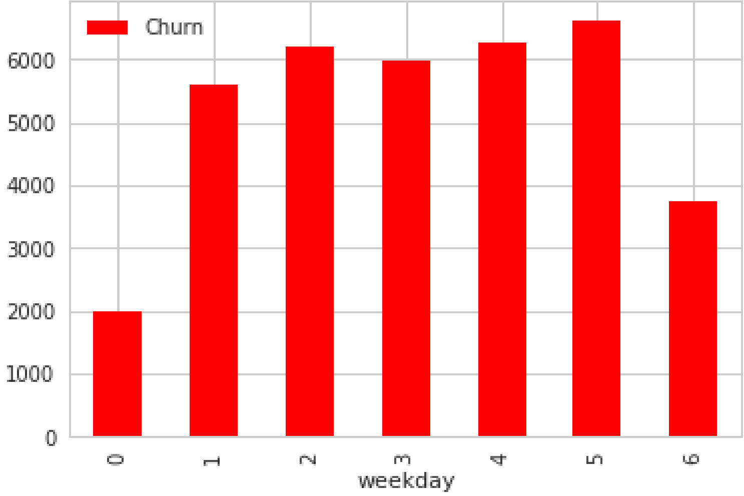
```
# Time distribution in churn data

get_hour = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).hour)
df_churn = df_churn.withColumn("hour", get_hour(df_churn.ts))

get_weekday = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).strftime("%w"))
df_churn = df_churn.withColumn("weekday", get_weekday(df_churn.ts))

get_day = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).day)
df_churn = df_churn.withColumn("day", get_day(df_churn.ts))


def plot_cnt_by_churn(time):
    """
    This function use to plot the distribution of different dimension
    """
    df_pd = df_churn.filter(df_churn.page == "NextSong").groupby("churn", time).count().orderBy(df_churn[time].cast("float")).toPandas()
    df_pd[time] = pd.to_numeric(df_pd[time])
    df_pd[df_pd.churn==0].plot.bar(x=time, y='count', color='Blue', label='Not churn')
    df_pd[df_pd.churn==1].plot.bar(x=time, y='count', color='Red', label='Churn')

plot_cnt_by_churn("hour")
```


Not churn
10000
8000
6000
4000
2000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour

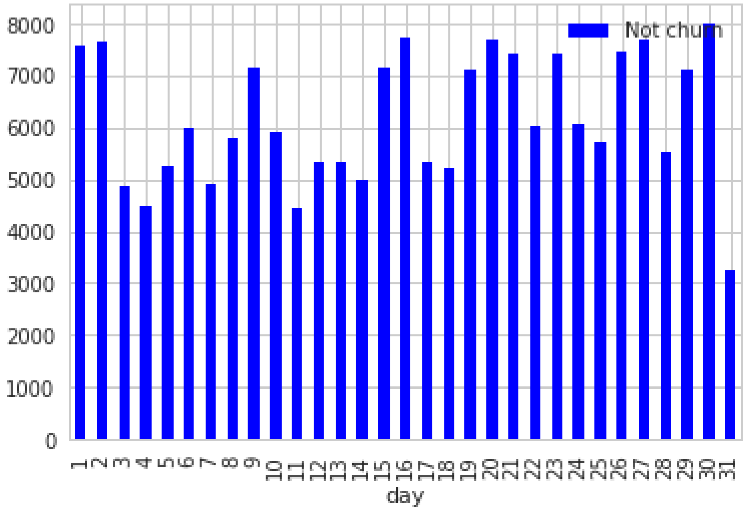

Churn
2000
1750
1500
1250
1000
750
500
250
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour

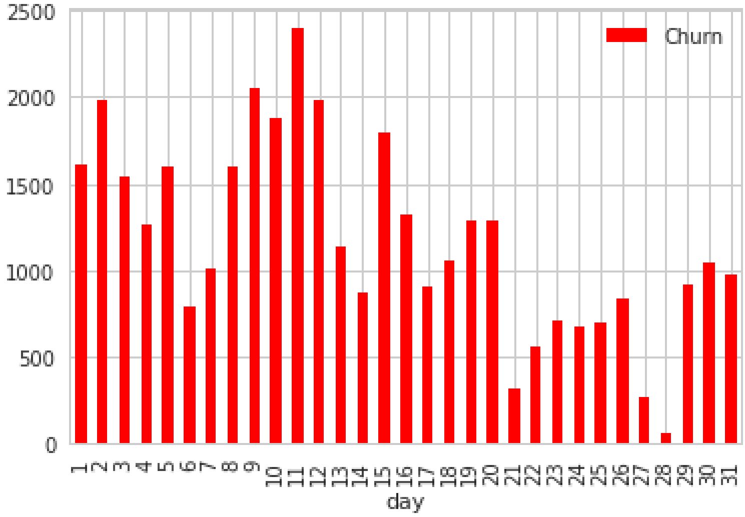
In [70]:
```
plot_cnt_by_churn("weekday")
```





In [71]:
```
plot_cnt_by_churn("day")
```





In [95]:

```
df_churn.columns
```

Out[95]:

```
['artist',
 'auth',
 'firstName',
 'gender',
 'itemInSession',
 'lastName',
 'length',
 'level',
 'location',
 'method',
 'page',
 'registration',
 'sessionId',
 'song',
 'status',
 'ts',
 'userAgent',
 'userId',
 'time',
 'churn',
 'hour',
 'weekday',
 'day']
```

In [94]:
```
df_churn.head(5)
```

Out[94]:
```
[Row(artist='Martha Tilston', auth='Logged In', firstName='Colin', gend
er='M', itemInSession=50, lastName='Freeman', length=277.89016, level
='paid', location='Bakersfield, CA', method='PUT', page='NextSong', reg
istration=1538173362000, sessionId=29, song='Rockpools', status=200, ts
=1538352117000, userAgent='Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0)
Gecko/20100101 Firefox/31.0', userId='30', time='2018-10-01 00:01:57',
churn=False, hour='0', weekday='1', day='1'),
 Row(artist='Five Iron Frenzy', auth='Logged In', firstName='Micah', ge
nder='M', itemInSession=79, lastName='Long', length=236.09424, level='f
ree', location='Boston-Cambridge-Newton, MA-NH', method='PUT', page='Ne
xtSong', registration=1538331630000, sessionId=8, song='Canada', status
=200, ts=1538352180000, userAgent='"Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.103 Safari/537.
36"', userId='9', time='2018-10-01 00:03:00', churn=False, hour='0', we
ekday='1', day='1'),
 Row(artist='Adam Lambert', auth='Logged In', firstName='Colin', gender
='M', itemInSession=51, lastName='Freeman', length=282.8273, level='pai
d', location='Bakersfield, CA', method='PUT', page='NextSong', registra
tion=1538173362000, sessionId=29, song='Time For Miracles', status=200,
ts=1538352394000, userAgent='Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.
0) Gecko/20100101 Firefox/31.0', userId='30', time='2018-10-01 00:06:3
4', churn=False, hour='0', weekday='1', day='1'),
 Row(artist='Enigma', auth='Logged In', firstName='Micah', gender='M',
itemInSession=80, lastName='Long', length=262.71302, level='free', loca
tion='Boston-Cambridge-Newton, MA-NH', method='PUT', page='NextSong', r
egistration=1538331630000, sessionId=8, song='Knocking On Forbidden Doo
rs', status=200, ts=1538352416000, userAgent='"Mozilla/5.0 (Windows NT
6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.103
Safari/537.36"', userId='9', time='2018-10-01 00:06:56', churn=False, h
our='0', weekday='1', day='1'),
 Row(artist='Daft Punk', auth='Logged In', firstName='Colin', gender
='M', itemInSession=52, lastName='Freeman', length=223.60771, level='pa
id', location='Bakersfield, CA', method='PUT', page='NextSong', registr
ation=1538173362000, sessionId=29, song='Harder Better Faster Stronge
r', status=200, ts=1538352676000, userAgent='Mozilla/5.0 (Windows NT 6.
1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0', userId='30', time='201
8-10-01 00:11:16', churn=False, hour='0', weekday='1', day='1')]
```

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.

In [73]:

```
# Feature 1: number of days since registration


user_max_ts = df_churn.groupby("userId").max("ts").sort("userId")
user_reg_ts = df_churn.select("userId", "registration").dropDuplicates()
.sort("userId")
user_reg_days = user_reg_ts.join(user_max_ts, user_reg_ts.userId == user
_max_ts.userId).select(user_reg_ts["userId"], ((user_max_ts["max(ts)"]-u
ser_reg_ts["registration"])/(1000*60*60*24)).alias("regDay"))

user_reg_days.show(10)
```

```
+------+------------------+
|userId|            regDay|
+------+------------------+
|    10| 51.76265046296297|
|   100| 64.87377314814815|
|100001| 44.80021990740741|
|100002|160.47207175925925|
|100003|22.748113425925926|
|100004|172.44008101851853|
|100005| 85.19559027777778|
|100006| 9.127164351851851|
|100007|115.38761574074074|
|100008| 68.22856481481482|
+------+------------------+
only showing top 10 rows
```

In [85]:

```
# Feature 2: gender


user_gender = df_churn.select("userId", "gender").dropDuplicates()
user_gender = user_gender.replace(["M", "F"], ["0", "1"], "gender")
user_gender = user_gender.select("userId", user_gender.gender.cast("int"
))

user_gender.show(10)
```

```
+------+------+
|userId|gender|
+------+------+
|    44|     1|
|    46|     1|
|    41|     1|
|    72|     1|
|300023|     1|
|    39|     1|
|100010|     1|
|    40|     1|
|    94|     1|
|    35|     1|
+------+------+
only showing top 10 rows
```

In [77]:

```
# Feature 3: number of songs per session

user_session_songs = df_churn.filter(df_churn.page=="NextSong").groupby("userId", "sessionId").count()
user_session_songs_avg = user_session_songs.groupby("userId").agg(avg(user_session_songs["count"]).alias("avgSessionSongs")).sort("userId")


user_session_songs_avg.show(10)
```

```
+------+------------------+
|userId|   avgSessionSongs|
+------+------------------+
|    10|112.16666666666667|
|   100| 78.88235294117646|
|100001|             33.25|
|100002|             48.75|
|100003|              25.5|
|100004|              47.1|
|100005|              38.5|
|100006|              26.0|
|100007|              47.0|
|100008|128.66666666666666|
+------+------------------+
only showing top 10 rows
```

In [100]:

```
# Feature 4: number of sessions


user_session_count = df_churn.select("userId", "sessionId").dropDuplicates().groupby("userId").count()
user_session_count = user_session_count.withColumnRenamed("count", "sessionCount")

user_session_count.show(5)
```

```
+------+------------+
|userId|sessionCount|
+------+------------+
|100010|           7|
|200002|           6|
|   125|           1|
|    51|          10|
|   124|          29|
+------+------------+
only showing top 5 rows
```

In [87]:

```
# Feature 5: frequency of use of pages

# get all the type of page
page_list = [(row['page']) for row in df_churn.select("page").dropDuplic
ates().collect()]

# must remove the column which will cause data leakage
page_list.remove("Cancel")
page_list.remove("Cancellation Confirmation")

# caculate the total page each user view
user_page_view_count = df_churn.groupby("userId").count()
user_page_view_count = user_page_view_count.withColumnRenamed("count",
"pageCount")


for page in page_list:
    col_name = "count" + page.replace(" ", "")
    view_count = df_churn.filter(df_churn.page==page).groupby("userId").
count()
    view_count = view_count.withColumnRenamed("count", col_name).withCol
umnRenamed("userId", "userIdTemp")
    user_page_view_count = user_page_view_count.join(view_count, user_pa
ge_view_count.userId==view_count.userIdTemp, "left").drop("userIdTemp")
user_page_view_count = user_page_view_count.sort("userId")
user_page_view_count = user_page_view_count.fillna(0)

col_list = user_page_view_count.columns
col_list.remove("userId")
col_list.remove("pageCount")
freq_sql = "select userId"
for col in col_list:
    col_name = col.replace("count", "freq")
    sql_str = ", (" + col + "/(pageCount/100)) as " + col_name
    freq_sql = freq_sql + sql_str
freq_sql = freq_sql + " from user_page_view_count"

user_page_view_count.createOrReplaceTempView("user_page_view_count")
col_list = user_page_view_count.columns
col_list.remove("userId")
col_list.remove("pageCount")
freq_sql = "select userId"
for col in col_list:
    col_name = col.replace("count", "freq")
    sql_str = ", (" + col + "/(pageCount/100)) as " + col_name
    freq_sql = freq_sql + sql_str
freq_sql = freq_sql + " from user_page_view_count"

user_page_view_freq = spark.sql(freq_sql)
```

```
In [89]: user_page_view_freq.show(20)
```

```
+------+--------------------+------------------+------------------+----
---------------+-------------------+------------------+----------------
---+-------------------+-------------------+------------------+--------
----------+------------------+------------------+-------------------+--
------------------+-------------------+--------------------+
|userId| freqSubmitDowngrade|    freqThumbsDown|          freqHome|
freqDowngrade|     freqRollAdvert|        freqLogout|   freqSaveSetting
s|          freqAbout|       freqSettings| freqAddtoPlaylist|     freqA
ddFriend|      freqNextSong|      freqThumbsUp|           freqHelp|
freqUpgrade|          freqError|   freqSubmitUpgrade|
+------+--------------------+------------------+------------------+----
---------------+-------------------+------------------+----------------
---+-------------------+-------------------+------------------+--------
----------+------------------+------------------+-------------------+--
------------------+-------------------+--------------------+
|    10|                 0.0|0.5031446540880503|3.7735849056603774| 0.8
805031446540881|0.12578616352201258|1.3836477987421383|0.12578616352201
258|0.25157232704402516| 0.8805031446540881|1.1320754716981132|1.509433
9622641508| 84.65408805031447| 4.654088050314465|0.12578616352201258|
0.0|                0.0|                 0.0|
|   100|0.031113876789047916|0.8400746733042938| 3.266957062850031| 0.9
334163036714375| 0.7778469197261979| 1.088985687616677| 0.1555693839452
396|0.37336652146857496|0.34225264467952704| 1.897946484131923|1.524579
9626633478| 83.44741754822651| 4.604853764779091| 0.5600497822028625|0.
031113876789047916|0.09334163036714374|0.031113876789047916|
|100001|                 0.0|  1.06951871657754|  5.88235294117647|
0.0| 7.4866310160427805|3.7433155080213902|                0.0|
0.0|   0.53475935828877|1.6042780748663101|  1.06951871657754| 71.12299
465240642|  4.27807486631016|   0.53475935828877|    1.06951871657754|
0.53475935828877|                 0.0|
|100002|                 0.0|               0.0|2.7522935779816513| 0.9
174311926605504| 1.3761467889908257|0.4587155963302752|
0.0|                0.0|                0.0| 2.293577981651376|0.458715
5963302752| 89.44954128440367| 2.293577981651376|                0.0|
0.0|                0.0|                 0.0|
|100003|                 0.0|               0.0| 8.974358974358974|
0.0| 11.538461538461538| 3.846153846153846|                0.0|
0.0|                0.0| 2.564102564102564|               0.0| 65.38461
538461539| 3.846153846153846|  1.282051282051282|                 0.0|
0.0|                 0.0|
|100004|  0.1606425702811245|0.8835341365461847| 5.301204819277109| 0.8
032128514056225|  6.907630522088354|1.5261044176706828| 0.1606425702811
245|                0.0| 0.8835341365461847|1.8473895582329318|1.526104
4176706828| 75.66265060240964| 2.811244979919679|0.48192771084337355|
0.642570281124498| 0.1606425702811245| 0.24096385542168677|
|100005|                 0.0|1.3888888888888888| 6.944444444444444|
0.0|  8.333333333333332|1.3888888888888888|                0.0|
0.0| 0.9259259259259258|1.3888888888888888|1.3888888888888888| 71.29629
629629629|3.2407407407407405| 0.9259259259259258|  1.8518518518518516|
0.0|                 0.0|
|100006|                 0.0| 4.545454545454546| 4.545454545454546|
0.0|  6.818181818181818| 2.272727272727273|                0.0|
0.0|  2.272727272727273| 2.272727272727273| 9.090909090909092| 59.09090
909090909| 4.545454545454546|                0.0|                 0.0|
0.0|                 0.0|
|100007|                 0.0|1.1538461538461537| 3.846153846153846|  1.
346153846153846| 0.9615384615384615|0.9615384615384615| 0.1923076923076
```

```
923|               0.0| 0.5769230769230769|1.7307692307692306| 3.26923
0769230769| 81.34615384615384|3.6538461538461537| 0.5769230769230769|
0.0|               0.0|                 0.0|
|100008| 0.10638297872340426|0.6382978723404255| 2.553191489361702| 1.0
638297872340425|  2.127659574468085|0.7446808510638298|
0.0| 0.3191489361702127| 0.3191489361702127|3.1914893617021276|1.808510
6382978722| 82.12765957446808|3.9361702127659575| 0.6382978723404255|
0.425531914893617|               0.0|                 0.0|
|100009| 0.14903129657228018|1.1922503725782414| 3.427719821162444| 0.7
451564828614009|  6.259314456035767|1.9374068554396424|0.14903129657228
018|0.14903129657228018| 0.5961251862891207|1.7883755588673622|1.043219
0760059612| 77.1982116244113| 3.427719821162444| 0.8941877794336811|
0.5961251862891207|               0.0| 0.14903129657228018|
|100010|                0.0|1.3123359580052494|2.8871391076115485|
0.0| 13.648293963254593|1.3123359580052494|               0.0|0.262467
19160104987|                0.0| 1.837270341207349|1.0498687664041995|
72.17847769028872|4.4619422572178475| 0.5249343832020997|  0.5249343832
020997|                0.0|                0.0|
|100011|                0.0|4.3478260869565215|17.391304347826086|
0.0|  8.695652173913043|              0.0|                0.0|
0.0| 4.3478260869565215| 8.695652173913043|               0.0|47.826086
956521735|              0.0|                0.0|                 0.0|
0.0|                 0.0|
|100012| 0.16666666666666666|              1.5|               4.5| 0.6
66666666666666|  6.333333333333333|                1.0|
0.0|0.16666666666666666|               0.0|               2.0|0.333333
3333333333| 79.33333333333333|              3.0| 0.3333333333333333|
0.16666666666666666|               0.0| 0.16666666666666666|
|100013|                0.0|1.0775862068965518| 3.807471264367816| 0.9
339080459770115| 2.8017241379310347|0.9339080459770115|0.21551724137931
036|0.14367816091954022| 0.5028735632183908|2.2270114942528734|2.011494
2528735633|            81.25|2.8017241379310347|  0.646551724137931|
0.28735632183908044|0.14367816091954022| 0.07183908045977011|
|100014|                0.0|0.9677419354838709| 2.258064516129032| 0.9
677419354838709| 0.6451612903225806|0.9677419354838709|
0.0|               0.0| 0.3225806451612903| 2.258064516129032|1.935483
8709677418| 82.90322580645162| 5.483870967741935| 0.6451612903225806|
0.0|               0.0|                 0.0|
|100015| 0.09523809523809523|0.7619047619047619| 4.476190476190476|0.47
619047619047616|  6.571428571428571|1.7142857142857142|0.09523809523809
523|0.38095238095238093| 0.6666666666666666|2.0952380952380953|1.333333
3333333333| 76.19047619047619|3.3333333333333335| 0.5714285714285714|
0.7619047619047619|0.09523809523809523| 0.19047619047619047|
|100016| 0.15673981191222572|0.7836990595611285|  3.29153605015674| 0.9
404388714733543| 2.5078369905956115|1.4106583072100314|
0.0|0.15673981191222572|0.15673981191222572|0.9404388714733543|2.037617
5548589344| 83.07210031347962| 3.918495297805643| 0.4702194357366771|
0.0|0.15673981191222572|                0.0|
|100017|                0.0|1.3333333333333333|               4.0|
0.0| 18.666666666666668|              0.0|                0.0|
0.0|               0.0|1.3333333333333333|              0.0| 69.33333
333333333|2.6666666666666665|               0.0|                 0.0|
0.0|                 0.0|
|100018| 0.15527950310559005|0.6987577639751552|3.3385093167701863| 0.6
987577639751552|  6.211180124223602|1.3198757763975155| 0.2329192546583
851|0.07763975155279502| 0.6987577639751552|2.4068322981366457|1.785714
2857142856| 77.79503105590062| 3.571428571428571| 0.5434782608695652|
0.3105590062111801|                0.0| 0.15527950310559005|
+------+--------------------+------------------+------------------+----
---------------+-------------------+------------------+----------------
---+-------------------+-------------------+------------------+--------
----------+------------------+------------------+-------------------+--
------------------+-------------------+--------------------+
only showing top 20 rows
```

In [90]:
```
#  Feature 6: how many singers have the user heard


user_artist_count = df_churn.filter(df_churn.page=="NextSong").select("userId", "artist").dropDuplicates().groupby("userId").count()
user_artist_count = user_artist_count.withColumnRenamed("count", "aritstCount")

user_artist_count.show(10)
```

```
+------+-----------+
|userId|aritstCount|
+------+-----------+
|100010|        252|
|200002|        339|
|   125|          8|
|    51|       1385|
|   124|       2232|
|     7|        142|
|    15|       1302|
|    54|       1744|
|   155|        643|
|   132|       1299|
+------+-----------+
only showing top 10 rows
```

In [91]:

```
#. Label: churn


user_churn = df_churn.select("userId", "churn").dropDuplicates()
user_churn = user_churn.select("userId", user_churn.churn.cast("int"))

user_churn.show(10)
```

```
+------+-----+
|userId|churn|
+------+-----+
|    19|    0|
|100005|    1|
|200007|    0|
|300007|    1|
|    50|    0|
|200002|    0|
|    30|    0|
|     8|    0|
|100011|    1|
|100012|    1|
+------+-----+
only showing top 10 rows
```

In [101]:

```
# Putting all the features into a dataframe


# put all the features dataframe into a list
features_list = []
features_list.append(user_reg_days)
features_list.append(user_session_songs_avg)
features_list.append(user_session_count)
features_list.append(user_gender)
features_list.append(user_page_view_freq)
features_list.append(user_artist_count)
features_list.append(user_churn)
```

In [102]:

```
# prepare the final dataframe to join all the other features
df_final = df_churn.select("userId").dropDuplicates()

def features_merge(df1, df2):
    """
    This function is used to merge the feature using left join
    input: two data frame to be merged
    output: merged dataframe
    """
    df2 = df2.withColumnRenamed("userId", "userIdTemp")
    df = df1.join(df2, df1.userId == df2.userIdTemp, "left").drop("userI
dTemp")
    return df
```

In [103]:

```
# use function to merge the features in the list
for feature in features_list:
    df_final = features_merge(df_final, feature)

# sort and view the final dataframe
df_final = df_final.sort("userId")
df_final.persist()
df_final.show(5)
```

```
+------+------------------+------------------+------------+------+-----
---------------+------------------+------------------+-----------------
-+-------------------+------------------+-------------------+----------
---------+-------------------+------------------+------------------+---
--------------+-----------------+-------------------+------------------
--+-------------------+--------------------+-----------+-----+
|userId|            regDay|   avgSessionSongs|sessionCount|gender| freq
SubmitDowngrade|    freqThumbsDown|          freqHome|     freqDowngrad
e|     freqRollAdvert|        freqLogout|   freqSaveSettings|
freqAbout|       freqSettings| freqAddtoPlaylist|     freqAddFriend|
freqNextSong|     freqThumbsUp|           freqHelp|         freqUpgrade
|          freqError|   freqSubmitUpgrade|aritstCount|churn|
+------+------------------+------------------+------------+------+-----
---------------+------------------+------------------+-----------------
-+-------------------+------------------+-------------------+----------
---------+-------------------+------------------+------------------+---
--------------+-----------------+-------------------+------------------
--+-------------------+--------------------+-----------+-----+
|    10| 51.76265046296297|112.16666666666667|           6|     0|
0.0|0.5031446540880503|3.7735849056603774|0.8805031446540881|0.12578616
352201258|1.3836477987421383|0.12578616352201258|0.25157232704402516|
0.8805031446540881|1.1320754716981132|1.5094339622641508|84.65408805031
447|4.654088050314465|0.12578616352201258|                 0.0|
0.0|                 0.0|        565|    0|
|   100| 64.87377314814815| 78.88235294117646|          35|     0|0.031
113876789047916|0.8400746733042938| 3.266957062850031|0.933416303671437
5| 0.7778469197261979| 1.088985687616677| 0.1555693839452396|0.37336652
146857496|0.34225264467952704| 1.897946484131923|1.5245799626633478|83.
44741754822651|4.604853764779091| 0.5600497822028625|0.0311138767890479
16|0.09334163036714374|0.031113876789047916|       1705|    0|
|100001| 44.80021990740741|             33.25|           4|     1|
0.0|  1.06951871657754|  5.88235294117647|               0.0| 7.4866310
160427805|3.7433155080213902|               0.0|                0.0|
0.53475935828877|1.6042780748663101|  1.06951871657754|71.1229946524064
2| 4.27807486631016|   0.53475935828877|    1.06951871657754|   0.53475
935828877|                 0.0|        125|    1|
|100002|160.47207175925925|             48.75|           4|     1|
0.0|               0.0|2.7522935779816513|0.9174311926605504| 1.3761467
889908257|0.4587155963302752|               0.0|                0.0|
0.0| 2.293577981651376|0.4587155963302752|89.44954128440367|2.293577981
651376|                0.0|                 0.0|                0.0|
0.0|        184|    0|
|100003|22.748113425925926|              25.5|           2|     1|
0.0|               0.0| 8.974358974358974|               0.0| 11.538461
538461538| 3.846153846153846|               0.0|                0.0|
0.0| 2.564102564102564|               0.0|65.38461538461539|3.846153846
153846|  1.282051282051282|                 0.0|                0.0|
0.0|         50|    1|
+------+------------------+------------------+------------+------+-----
---------------+------------------+------------------+-----------------
-+-------------------+------------------+-------------------+----------
---------+-------------------+------------------+------------------+---
--------------+-----------------+-------------------+------------------
--+-------------------+--------------------+-----------+-----+
only showing top 5 rows
```

In [ ]:

```
```

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 [104]:

```
# let's check how much data do we have.

df_final.groupby("churn").count().show()
```

```
+-----+-----+
|churn|count|
+-----+-----+
|    1|   52|
|    0|  173|
+-----+-----+
```

In [105]:

```
output = "final_data.csv"

df_final.write.save(output, format="csv", header=True)

df_final = spark.read.csv(output, header=True)
df_final.persist()
```

Out[105]:

```
DataFrame[userId: string, regDay: string, avgSessionSongs: string, sess
ionCount: string, gender: string, freqSubmitDowngrade: string, freqThum
bsDown: string, freqHome: string, freqDowngrade: string, freqRollAdver
t: string, freqLogout: string, freqSaveSettings: string, freqAbout: str
ing, freqSettings: string, freqAddtoPlaylist: string, freqAddFriend: st
ring, freqNextSong: string, freqThumbsUp: string, freqHelp: string, fre
qUpgrade: string, freqError: string, freqSubmitUpgrade: string, aritstC
ount: string, churn: string]
```

In [106]:

```
# Lets convert all the features to numeric.

num_features_list = df_final.columns[1:]
for f in num_features_list:
    f_name = f + "Num"
    df_final = df_final.withColumn(f_name, df_final[f].cast("float"))
    df_final = df_final.drop(f)
```

In [107]:

```
# Lets train the features into a vector


assembler = VectorAssembler(inputCols=df_final.columns[1:-1], outputCol=
"NumFeatures")
data = assembler.transform(df_final)

scaler = StandardScaler(inputCol="NumFeatures", outputCol="ScaledNumFeat
ures", withStd=True)
scalerModel = scaler.fit(data)
data = scalerModel.transform(data)

data = data.select(data.churnNum.alias("label"), data.ScaledNumFeatures.
alias("features"))

train, validation = data.randomSplit([0.9, 0.1], seed=42)
train = train.cache()
```

In [108]:

```
lr =  LogisticRegression()
paramGrid = ParamGridBuilder() \
    .addGrid(lr.elasticNetParam,[0.0, 0.1, 0.5, 1.0]) \
    .addGrid(lr.regParam,[0.0, 0.05, 0.1]) \
    .build()

crossval = CrossValidator(estimator=lr,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=3)
cvModel_lr = crossval.fit(train)

cvModel_lr.save('cvModel_lr.model')

cvModel_lr.avgMetrics
```

Out[108]:

```
[0.8005825382361194,
 0.7373409099969138,
 0.7331116591529545,
 0.8005825382361194,
 0.7288125875568267,
 0.7284167395575718,
 0.8005825382361194,
 0.7372280827767593,
 0.6951150985956462,
 0.8005825382361194,
 0.7119161587284445,
 0.6935331299204026]
```

In [109]:

```
dt = DecisionTreeClassifier()
paramGrid = ParamGridBuilder() \
    .addGrid(dt.impurity,['entropy', 'gini']) \
    .addGrid(dt.maxDepth,[2, 3, 4, 5, 6, 7, 8]) \
    .build()

crossval_dt = CrossValidator(estimator=dt,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=3)

cvModel_dt = crossval_dt.fit(train)

cvModel_dt.save('cvModel_dt.model')

cvModel_dt.avgMetrics
```

Out[109]:

```
[0.809391103951177,
 0.7501008119218004,
 0.7243218701425422,
 0.7125347861131004,
 0.7199490636115637,
 0.7336895542612214,
 0.7310787550540219,
 0.8245202237499156,
 0.7383282194663868,
 0.7446536981714925,
 0.752821690006628,
 0.753965808893903,
 0.7417978017336555,
 0.7417978017336555]
```

In [112]:

```
gbt = GBTClassifier()

paramGrid = ParamGridBuilder() \
    .addGrid(gbt.maxIter,[3, 10, 20]) \
    .addGrid(gbt.maxDepth,[2, 4, 6, 8]) \
    .build()

crossval_gbt = CrossValidator(estimator=gbt,
                              estimatorParamMaps=paramGrid,
                              evaluator=MulticlassClassificationEvaluator(),
                              numFolds=3)

cvModel_gbt = crossval_gbt.fit(train)

cvModel_gbt.save('cvModel_gbt.model')

cvModel_gbt.avgMetrics
```

Out[112]:

```
[0.823855068054212,
 0.7378619062152055,
 0.753965808893903,
 0.7417978017336555,
 0.7948841221807386,
 0.7418057214032604,
 0.753965808893903,
 0.7417978017336555,
 0.7855099580040472,
 0.7338696709530763,
 0.753965808893903,
 0.7417978017336555]
```

In [113]:

```
stratified_train = train.sampleBy('label', fractions={0: 99/349, 1: 1.0
}).cache()
stratified_train.groupby("label").count().show()
```

```
+-----+-----+
|label|count|
+-----+-----+
|  1.0|   44|
|  0.0|   40|
+-----+-----+
```

In [114]:

```
lrs =  LogisticRegression()
paramGrid = ParamGridBuilder() \
    .addGrid(lrs.elasticNetParam,[0.0, 0.1, 0.5, 1.0]) \
    .addGrid(lrs.regParam,[0.0, 0.05, 0.1]) \
    .build()

crossval_lrs = CrossValidator(estimator=lrs,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=3)
cvModel_lrs = crossval_lrs.fit(stratified_train)
cvModel_lrs.avgMetrics
```

Out[114]:

```
[0.584848484848485,
 0.5685105018438352,
 0.5588924777779267,
 0.584848484848485,
 0.5362999712979521,
 0.5616072777198097,
 0.584848484848485,
 0.5717316878442198,
 0.510604560712087б,
 0.584848484848485,
 0.582304606627584 4,
 0.5174310801761782]
```

In [115]:

```
cvModel_lrs.save('cvModel_lrs.model')
```

In [116]:

```
dts = DecisionTreeClassifier()
paramGrid = ParamGridBuilder() \
    .addGrid(dts.impurity,['entropy', 'gini']) \
    .addGrid(dts.maxDepth,[2, 3, 4, 5, 6, 7, 8]) \
    .build()
crossval_dts = CrossValidator(estimator=dts,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=3)
cvModel_dts = crossval_dts.fit(stratified_train)
cvModel_dts.avgMetrics
```

Out[116]:

```
[0.5995466555416493,
 0.6720397283443197,
 0.5931161073096557,
 0.5588591026091027,
 0.6150359347281962,
 0.6061778499278501,
 0.6061778499278501,
 0.5835235065245915,
 0.6264957264957265,
 0.5459702496983199,
 0.5365808014422584,
 0.5365808014422584,
 0.5365808014422584,
 0.5365808014422584]
```

In [117]:

```
cvModel_dts.save('cvModel_dts.model')
```

In [118]:

```
gbts = GBTClassifier()
paramGrid = ParamGridBuilder() \
    .addGrid(gbts.maxIter,[3, 10, 20]) \
    .addGrid(gbts.maxDepth,[2, 4, 6, 8]) \
    .build()
crossval_gbts = CrossValidator(estimator=gbts,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=3)
cvModel_gbts = crossval_gbts.fit(stratified_train)
cvModel_gbts.avgMetrics
```

Out[118]:

```
[0.6027873524138243,
 0.5499410164960218,
 0.5365808014422584,
 0.5365808014422584,
 0.6568068172187166,
 0.5945591087526572,
 0.5365808014422584,
 0.5365808014422584,
 0.66054381348499,
 0.6262132928799596,
 0.5365808014422584,
 0.5365808014422584]
```

In [119]:

```
cvModel_gbts.save('cvModel_gbts.model')
```

In [120]:

```
# Use the validate data to evaluate the best model through F1 score.


results = cvModel_lr.transform(validation)

tp = results.filter("label = 1 and prediction = 1").count()
fp = results.filter("label = 0 and prediction = 1").count()
fn = results.filter("label = 1 and prediction = 0").count()
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1 = 2*precision*recall / (precision+recall)
print(precision)
print(recall)
print(f1)
```

```
1.0
0.75
0.8571428571428571
```

In [121]:

```
results = cvModel_lrs.transform(validation)
```

In [122]:

```
tp = results.filter("label = 1 and prediction = 1").count()
fp = results.filter("label = 0 and prediction = 1").count()
fn = results.filter("label = 1 and prediction = 0").count()
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1 = 2*precision*recall / (precision+recall)
print(precision)
print(recall)
print(f1)
```

```
0.6363636363636364
0.875
0.7368421052631579
```

In [123]:

```
cvModel_lrs.bestModel.coefficients
```

Out[123]:

```
DenseVector([4.8203, 11.436, -16.637, 5.6112, -13.3528, -107.9358, -27
0.1181, -87.8711, -572.666, -119.6136, -44.9646, -27.1968, -233.8042, -
172.517, -235.4224, -953.0062, -285.0545, -59.4815, -55.8425, -30.8607,
-27.3215, 21.452])
```

In [124]:

```
df_final.columns
```

Out[124]:

```
['userId',
 'regDayNum',
 'avgSessionSongsNum',
 'sessionCountNum',
 'genderNum',
 'freqSubmitDowngradeNum',
 'freqThumbsDownNum',
 'freqHomeNum',
 'freqDowngradeNum',
 'freqRollAdvertNum',
 'freqLogoutNum',
 'freqSaveSettingsNum',
 'freqAboutNum',
 'freqSettingsNum',
 'freqAddtoPlaylistNum',
 'freqAddFriendNum',
 'freqNextSongNum',
 'freqThumbsUpNum',
 'freqHelpNum',
 'freqUpgradeNum',
 'freqErrorNum',
 'freqSubmitUpgradeNum',
 'aritstCountNum',
 'churnNum']
```

In [ ]:

In [ ]:

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 [ ]: