pyspark_Start

July 31, 2021

1 Library and dataset setup

[4]:
```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.1.2.tar.gz (212.4 MB)
     || 212.4 MB 67 kB/s
Collecting py4j==0.10.9
  Downloading py4j-0.10.9-py2.py3-none-any.whl (198 kB)
     || 198 kB 54.9 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.1.2-py2.py3-none-any.whl
size=212880768
sha256=a2d6675402f81eb7ce3e2662f185b5648c55fe4c43e047c9521a48ca0f13cc85
  Stored in directory: /root/.cache/pip/wheels/a5/0a/c1/9561f6fecb759579a7d863dc
d846daaa95f598744e71b02c77
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.1.2
```

[5]:
```
import pandas as pd
import requests
import numpy as np


### Load the required packages in the required format

%matplotlib notebook
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
plt.style.use('ggplot')
import warnings
warnings.filterwarnings("ignore")
```

```
import pyspark
```

[6]:
```
!pip install -q kaggle
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>


Saving kaggle.json to kaggle.json
```

[6]:
```
{'kaggle.json':
b'{"username":"luigirachiele","key":"47ab4d1b758be41dba8aff5c82f8f8b4"}'}
```

[7]:
```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
#!kaggle datasets list
!kaggle datasets download -d ashwinik/spotify-playlist
!unzip spotify-playlist.zip
```

```
Downloading spotify-playlist.zip to /content
 98% 180M/183M [00:02<00:00, 80.4MB/s]
100% 183M/183M [00:02<00:00, 84.4MB/s]
Archive:  spotify-playlist.zip
  inflating: README.txt
  inflating: spotify_dataset.csv
```

[28]:
```
spotify_data = pd.read_csv('/content/spotify_dataset.csv',escapechar= '.
 ↪',error_bad_lines = False,warn_bad_lines=False)
spotify_data.columns = ['user_id','artistname','trackname','playlistname']
print ("Read Succesful with shape {}".format(spotify_data.shape))
```

```
Read Succesful with shape (12774191, 4)
```

[9]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.config("spark.driver.memory","12g").
 ↪appName('Practise').getOrCreate()
```

2 Spotify API

[11]:
```
import sys
import spotipy
import spotipy.util as util
```

```
idPlaylist = '4XoMUd3cPgLeazj8ezngUu';
username = 'gixs'


def show_tracks(tracks):
    for i, item in enumerate(tracks['items']):
        track = item['track']
        print("   %d %32.32s %s" % (i, track['artists'][0]['name'],
            track['name']))


def getSinglePlaylist (playlistId, sp, username):
    return sp.user_playlist (user = username, playlistId = playlistId,␣
 ↪fields="tracks, next")




def createClient():
    if len(sys.argv) > 1:
        username = sys.argv[1]
    else:
        print("Whoops, need your username!")
        print("usage: python user_playlists.py [username]")
        sys.exit()

        token = util.prompt_for_user_token(username)

        if token:
            sp = spotipy.Spotify(auth=token)
            return sp

        else:
            print("Can't get token for", username)
            return null


token = util.prompt_for_user_token(username)

if token:
  sp = spotipy.Spotify(auth=token)

else:
  print("Can't get token for", username)

singlePlaylist = sp.user_playlist (user = username, playlist_id = idPlaylist,␣
 ↪fields="tracks, next")
```

```
tracks = singlePlaylist ['tracks']
```

```
      ␣
↪---------------------------------------------------------------------------

       ModuleNotFoundError                       Traceback (most recent call␣
↪last)

       <ipython-input-11-e9d67741bdda> in <module>()
         1 import sys
   ----> 2 import spotipy
         3 import spotipy.util as util
         4
         5 idPlaylist = '4XoMUd3cPgLeazj8ezngUu';


       ModuleNotFoundError: No module named 'spotipy'



      ␣
↪---------------------------------------------------------------------------
   NOTE: If your import is failing due to a missing package, you can
   manually install dependencies using either !pip or !apt.

   To view examples of installing some common dependencies, click the
   "Open Examples" button below.
   ---------------------------------------------------------------------------
```

3 Dataset analysis

[12]:
```
print ("Some General statistics about data are as follows:",spotify_data.info())
print ("Lets look at the summary stats about the data :",spotify_data.
 ↪describe(include ='object'))
print ("The number of rows in the datasets are as follows :",spotify_data.
 ↪shape[0])
print (" The columns in the data are as follows :",spotify_data.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12774191 entries, 0 to 12774190
Data columns (total 4 columns):
 #   Column        Dtype
---  ------        -----
```

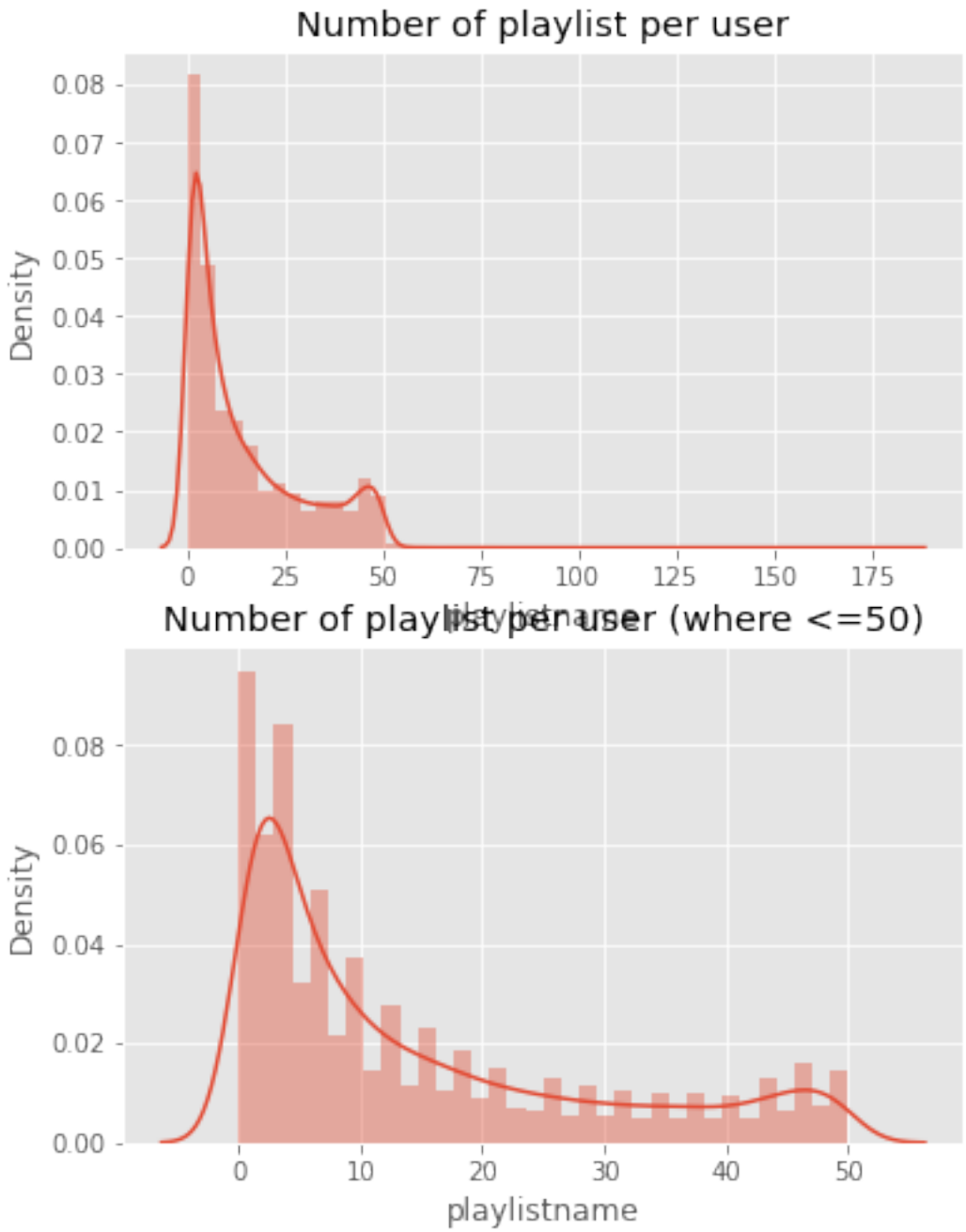
```
 0   user_id       object
 1   artistname    object
 2   trackname     object
 3   playlistname  object
dtypes: object(4)
memory usage: 389.8+ MB
Some General statistics about data are as follows: None
Lets look at the summary stats about the data :
user_id artistname trackname playlistname
count                            12774191   12741100  12774090     12634598
unique                             15897     303027   2057995       155522
top     4398de6902abde3351347b048fcdc287  Daft Punk   Starred      Starred
freq                              295274      35805     11159      1320739
The number of rows in the datasets are as follows : 12774191
 The columns in the data are as follows : Index(['user_id', 'artistname',
'trackname', 'playlistname'], dtype='object')
```

First of all, see some analysis on the data.

3.0.1 Here there is the distribution of number of playlist by *user id*.

[13]:
```
spotify_user_summary = spotify_data.groupby(['user_id'])["playlistname"].
 ↪nunique().reset_index()
```

[14]:
```
#GRAPH
### Just for better visualisation remove very high playlist
%matplotlib inline
fig, ax = plt.subplots(2, figsize=(6,8))
sns.distplot(spotify_user_summary['playlistname'], hist=True,ax = ax[0])
ax[0].set_title("Number of playlist per user")
spotify_user_summary =␣
 ↪spotify_user_summary[spotify_user_summary['playlistname'] <= 50]
sns.distplot(spotify_user_summary['playlistname'], hist=True,ax=ax[1])
ax[1].set_title("Number of playlist per user (where <=50)")
plt.show()
```

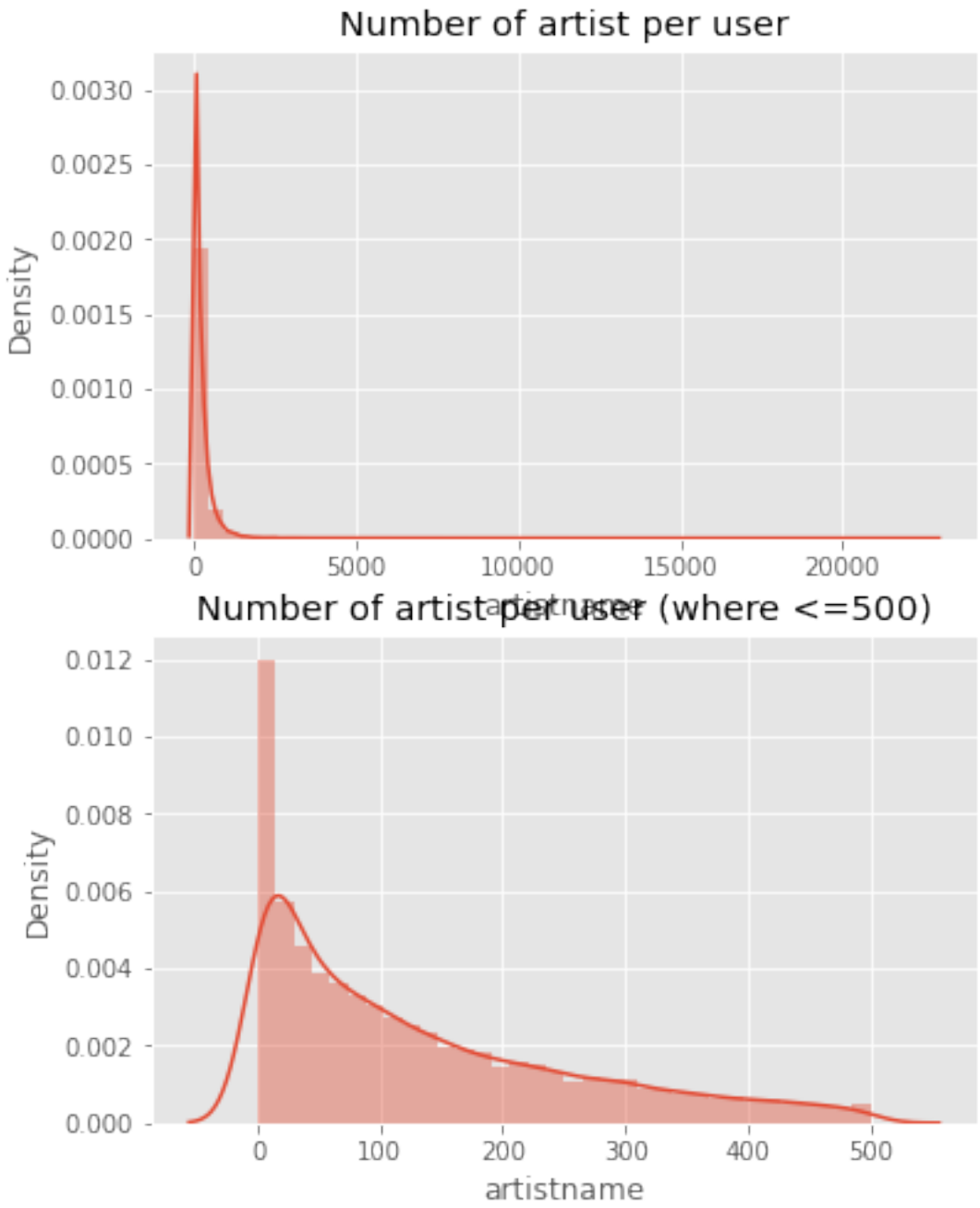


3.0.2 Then visualize the distribution of number of *artist* for each *user_id*.

[15]:
```
spotify_user_summary = spotify_data.groupby(['user_id'])["artistname"].
↪nunique().reset_index()
```

[16]:
```
#GRAPH
```

```
%matplotlib inline
### Just for better visualisation remove very high playlist
fig, ax = plt.subplots(2, figsize=(6,8))
sns.distplot(spotify_user_summary["artistname"], hist=True,ax = ax[0])
ax[0].set_title("Number of artist per user")
spotify_user_summary = spotify_user_summary[spotify_user_summary["artistname"]␣
 ↪<= 500]
sns.distplot(spotify_user_summary["artistname"], hist=True,ax=ax[1])
ax[1].set_title("Number of artist per user (where <=500)")
plt.show()
```



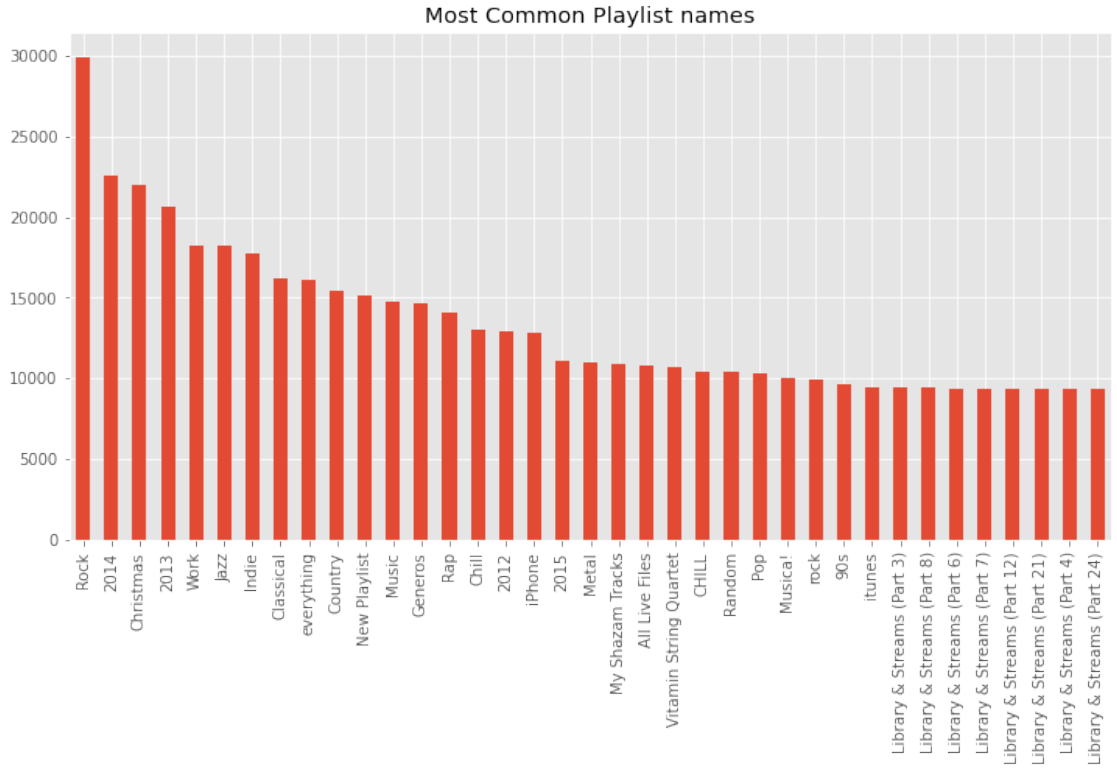
3.0.3 Then check the most common playlist names.

I removed the first 3 playlist that was given by spotify:

- Starred
- Liked from Radio
- Favorites de la radio

[33]:
```
%matplotlib inline
### Lets have a look at most common playlist
## We could have aslo created a word cloud
spotify_data['playlistname'].value_counts()[3:40].plot(kind= 'bar',title="Most␣
 ↪Common Playlist names",figsize = (12,6))
```

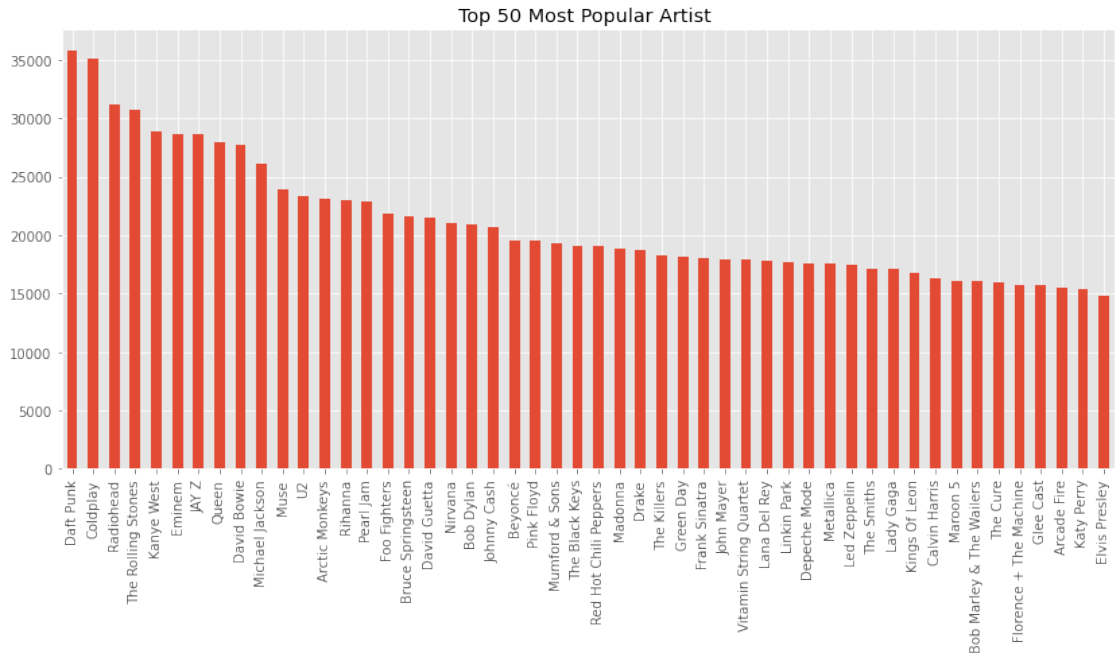
[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57fa8d7290>
```



3.0.4 Here visualize the top 50 Most Popular Artisti

[40]:
```
%matplotlib inline
spotify_data['artistname'].value_counts()[0:50].plot(kind= 'bar',title="Top 50␣
 ↪Most Popular Artist ",figsize = (14,6))
```

[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57f9e07a10>
```

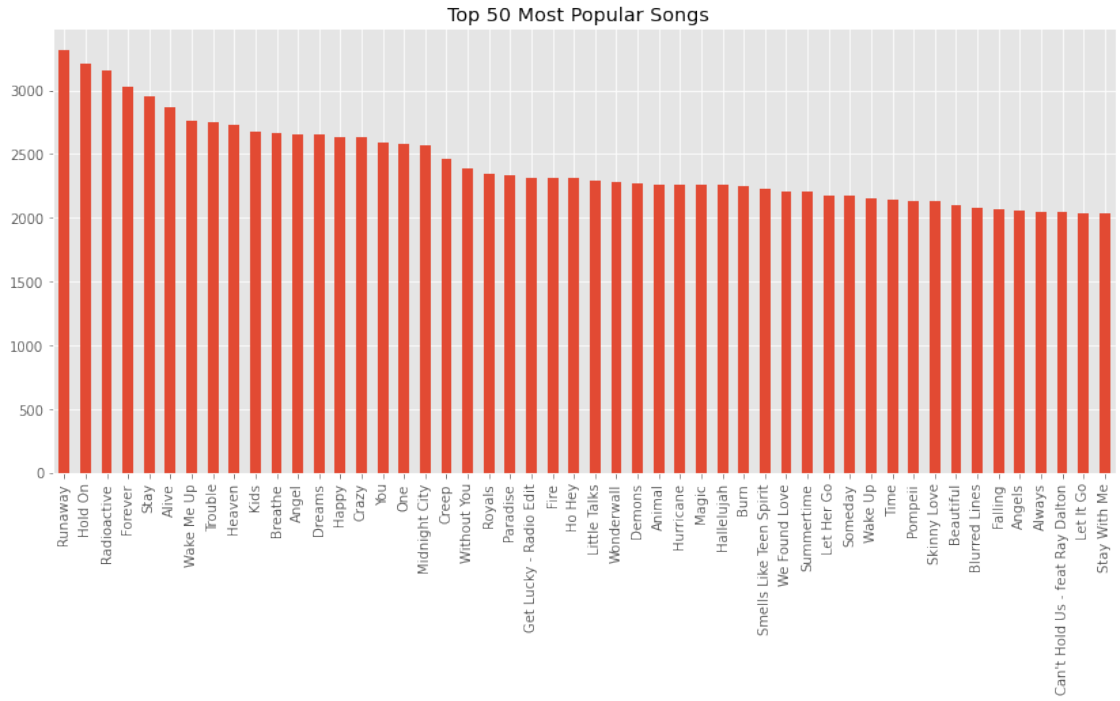


3.0.5 Here visualize the top 50 Most Popular Songs, removing the song that have common name like:

- Starred
- Intro
- Home
- Closer

[39]:
```
%matplotlib inline
### Lets have a look at most common playlist
## We could have aslo created a word cloud
spotify_data['trackname'].value_counts()[4:54].plot(kind= 'bar',title="Top 50␣
 ↪Most Popular Songs  ",figsize = (14,6))
```

[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57f9f52d90>
```



4 Data Preprocessing

In this section there is the preprocessing of data.
First of all, let's drop some useless rows:

- The rows containing null value
- The rows that contain the songs with "intro, home, starred, closer", that's beacause it will affect negatively the algorith of FP-growth.

[20]:
```
tracklist = spotify_data['trackname'].unique()
```

[21]:
```
spotify_data.dropna(inplace = True)
spotify_data = spotify_data[~spotify_data['trackname'].str.startswith("intro",␣
↪na=False)]
spotify_data = spotify_data[~spotify_data['trackname'].str.startswith("Home",␣
↪na=False)]
spotify_data = spotify_data[~spotify_data['trackname'].str.contains("Starred",␣
↪na=False)]
spotify_data = spotify_data[~spotify_data['trackname'].str.startswith("Closer",␣
↪na=False)]
```

4.1 Now lets define function which creates a dictionary and convert songs names to dictionary.

That's helpful to reduce the computation cost. Obviously *Integer* are lighter respcet to *String*.

[22]:
```
def create_dict(dataset, column):
    ''' Takes two input from user column name and dataset name and return␣
 ↪dictionary with hash map '''
    unique_list = dataset[column].unique()
    out_dict = {}
    out_dict1 = {}

    for j,i in enumerate(unique_list):
        out_dict[i.lower()] = str(j)
        out_dict1[str(j)] = i.lower()

    print ("Number of distinct in vocab is :",j)
    return (out_dict,out_dict1)
```

[23]:
```
track_map, track_map_comp= create_dict(spotify_data,'trackname')
artist_map,artist_map_comp = create_dict(spotify_data,'artistname')
```

```
Number of distinct in vocab is : 1976546
Number of distinct in vocab is : 285437
```

4.1.1 Save the dictionary as pickle file, so it will be not needed to create dict every time, but we can upload as pickle.

[24]:
```
with open('track_map_dict.pickle','wb') as track_file:
    pickle.dump(track_map,track_file)
with open('track_map_comp_dict.pickle','wb') as track_file_comp:
    pickle.dump(track_map_comp,track_file_comp)
```

[25]:
```
with open('artist_map_dict.pickle','wb') as artist_file:
    pickle.dump(artist_map,artist_file)
with open('artist_map_comp_dict.pickle','wb') as artist_file_comp:
    pickle.dump(artist_map_comp,artist_file_comp)
```

[10]:
```
### Load methods
with open('track_map_dict.pickle','rb') as dict1:
    track_dict= pickle.load( dict1)
print ("Track dict has {} observations".format(len(track_dict)))
#### Load the prcikle file for artist to numeric
with open('artist_map_dict.pickle','rb') as dict2:
    artist_dict = pickle.load(dict2)
print ("Artist dict has {} observations".format(len(artist_dict)))
```

```
Track dict has 1864427 observations
Artist dict has 277987 observations
```

[27]:
```
### Now we will use this mapping to convert names to numeric
print ("Data before mapping dict :", spotify_data.head(5))
```

```
spotify_data['trackname'] = spotify_data['trackname'].str.lower().
 ↪map(track_dict)
spotify_data['artistname'] = spotify_data['artistname'].str.lower().
 ↪map(artist_dict)
print ("Data after mapping dict :")
print (spotify_data.head(5))
```

```
Data before mapping dict :                                user_id  ...
playlistname
0  9cc0cfd4d7d7885102480dd99e7a90d6  ...  HARD ROCK 2010
1  9cc0cfd4d7d7885102480dd99e7a90d6  ...  HARD ROCK 2010
2  9cc0cfd4d7d7885102480dd99e7a90d6  ...  HARD ROCK 2010
3  9cc0cfd4d7d7885102480dd99e7a90d6  ...  HARD ROCK 2010
4  9cc0cfd4d7d7885102480dd99e7a90d6  ...  HARD ROCK 2010

[5 rows x 4 columns]
Data after mapping dict :
                           user_id artistname trackname    playlistname
0  9cc0cfd4d7d7885102480dd99e7a90d6          0   1514247  HARD ROCK 2010
1  9cc0cfd4d7d7885102480dd99e7a90d6      49541    544038  HARD ROCK 2010
2  9cc0cfd4d7d7885102480dd99e7a90d6          2         2  HARD ROCK 2010
3  9cc0cfd4d7d7885102480dd99e7a90d6      49541   1516450  HARD ROCK 2010
4  9cc0cfd4d7d7885102480dd99e7a90d6          0         4  HARD ROCK 2010
```

4.1.2 In order to get the file in the format [playlistname, array_of_track] it is necessary to create a zip_list method to aplly on the dataset.

[28]:
```
### We want to create a list of songs in zip file
def zip_list(x):
    return ([int(z) for z in x])
```

[29]:
```
spotify_summary = spotify_data.groupby(['user_id','playlistname'])['trackname'].
 ↪apply(zip_list).reset_index()
```

[30]:
```
print (" Distinct playlist after summarizing the data is :",spotify_summary.
 ↪shape[0])
print (" The data looks like this :")
print (spotify_summary.head(5))
```

```
 Distinct playlist after summarizing the data is : 229155
 The data looks like this :
                           user_id  ...
trackname
0  00055176fea33f6e027cd3302289378b  ...  [9608, 2586, 46634, 9609, 1137195,
37306, 6327...
1  0007f3dd09c91198371454c608d47f22  ...               [174795, 1537, 877677,
17529, 5292]
2  0007f3dd09c91198371454c608d47f22  ...  [1852573, 174795, 1682684, 954395,
```

```
19579, 1478...
3  0007f3dd09c91198371454c608d47f22  ...
[1509309, 1446920]
4  000b0f32b5739f052b9d40fcc5c41079  ...
[1280348, 487024]

[5 rows x 3 columns]
```

[31]:
```
### We will Dump this data in the pickle file and work in it later
with open("spotify_summary.pickle",'wb') as pick_data:
    pickle.dump(spotify_summary,pick_data)
```

4.1.3 I created also the playlist dictionary, but I didn't utilize for this script.

[32]:
```
playlist_map, playlist_map_comp= create_dict(spotify_data,'playlistname')
```

```
Number of distinct in vocab is : 155321
```

[33]:
```
with open('playlist_map_dict.pickle','wb') as playlist_file:
    pickle.dump(playlist_map,playlist_file)
with open('playlist_map_comp_dict.pickle','wb') as playlist_file_comp:
    pickle.dump(playlist_map_comp,playlist_file_comp)
```

[34]:
```
print ("Data before mapping dict :", spotify_data.head(5))
spotify_summary['playlistname'] = spotify_summary['playlistname'].str.lower().
 ↪map(playlist_map)
print ("Data after mapping dict :")
print (spotify_summary.head(5))
```

```
Data before mapping dict :                                user_id artistname
trackname    playlistname
0  9cc0cfd4d7d7885102480dd99e7a90d6          0   1514247  HARD ROCK 2010
1  9cc0cfd4d7d7885102480dd99e7a90d6      49541    544038  HARD ROCK 2010
2  9cc0cfd4d7d7885102480dd99e7a90d6          2         2  HARD ROCK 2010
3  9cc0cfd4d7d7885102480dd99e7a90d6      49541   1516450  HARD ROCK 2010
4  9cc0cfd4d7d7885102480dd99e7a90d6          0         4  HARD ROCK 2010
Data after mapping dict :
                            user_id  ...
trackname
0  00055176fea33f6e027cd3302289378b  ...  [9608, 2586, 46634, 9609, 1137195,
37306, 6327...
1  0007f3dd09c91198371454c608d47f22  ...               [174795, 1537, 877677,
17529, 5292]
2  0007f3dd09c91198371454c608d47f22  ...  [1852573, 174795, 1682684, 954395,
19579, 1478...
3  0007f3dd09c91198371454c608d47f22  ...
[1509309, 1446920]
```

```
4  000b0f32b5739f052b9d40fcc5c41079  ...
[1280348, 487024]

[5 rows x 3 columns]
```

5 FP Growth

5.1 Now setup the playlist in which the predict needs to be done.

[8]:
```
playlist_string = ["Till I Collapse", "Lose Yourself", "Love is a Laserquest",␣
↪"Madness", "Man in Black", "Notion", "The Song You Sing", "Complicated",␣
↪"Get lucky"]
```

[7]:
```
def playlist_to_int (playlist_string, dict):
  result = []
  for x in playlist_string:
    if (x.lower() in dict):
      result.append(int(dict.get(x.lower())))

  return result
```

[11]:
```
playlist_int = playlist_to_int(playlist_string, track_dict)
```

Select the needed column on dataset

[39]:
```
spotify_summary_pandas_selected = spotify_summary[['playlistname', 'trackname']]
```

[40]:
```
spotify_summary_pandas_selected
```

[40]:
```
       playlistname                                          trackname
0             45800  [9608, 2586, 46634, 9609, 1137195, 37306, 6327...
1               850                [174795, 1537, 877677, 17529, 5292]
2             95327  [1852573, 174795, 1682684, 954395, 19579, 1478...
3            134508                                  [1509309, 1446920]
4             28210                                   [1280348, 487024]
...             ...                                                 ...
229150       121343  [181459, 2313, 63638, 2890, 2954, 1682677, 167...
229151       121344  [233650, 1457, 62517, 106134, 1415671, 502479,...
229152       121345  [196633, 1531, 197104, 24277, 114148, 37306, 6...
229153       121346  [31617, 1516653, 141317, 37189, 1954879, 26123...
229154       146464  [141002, 266905, 2566, 38918, 1488929, 1850302...

[229155 rows x 2 columns]
```

[41]:
```
### We will Dump this data in the pickle file to work in it later
with open("spotify_summary_pandas_selected.pickle",'wb') as pick_data:
    pickle.dump(spotify_summary_pandas_selected,pick_data)
```

5.2 From here can start the spark code

[1]:
```
import pandas as pd
import requests
import numpy as np


### Load the required packages in the required format

%matplotlib notebook
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
plt.style.use('ggplot')
import warnings
warnings.filterwarnings("ignore")

import pyspark
```

[2]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Practise").config("spark.executor.
 ↪memory", "5g").config("spark.driver.memory", "5g").getOrCreate()
```

[3]:
```
with open("spotify_summary_pandas_selected.pickle",'rb') as pick_data:
    spotify_summary_pandas_selected = pickle.load(pick_data)
```

[4]:
```
spotify_dataframe_from_pickle = spark.
 ↪createDataFrame(spotify_summary_pandas_selected)
```

Process the FP Growth in the whole dataset is not possible for this machine. So we need to create a subset from our big main dataset:

The roole chosen for the subset is: if a song that are in the playlist in which I need to perform the prediction appears in one playlist in the dataset add this playlist to the dataset.

[5]:
```
from pyspark.sql.functions import array_contains


def create_dataframe(dataset, playlist):
  notAssegnato = True
  filtred_array = []
  for i in range (len(playlist)):
    filtred_array.append(dataset.filter(array_contains(dataset.trackname,␣
 ↪playlist[i])))
    if notAssegnato:
      result = filtred_array[0]
      notAssegnato = False
    else:
```

```
        result = result.union(filtred_array[i])
    return result
```

[12]:
```
filtred_dataset = create_dataframe(spotify_dataframe_from_pickle, playlist_int)
```

Now eliminate the row with duplicate because each item in transaction must be unique.

[13]:
```
from pyspark.sql.functions import array_distinct
filtred_dataset = filtred_dataset.withColumn("arraycol_without_dupes",␣
 ↪array_distinct("trackname"))
```

[14]:
```
filtred_dataset = filtred_dataset.
 ↪select('playlistname','arraycol_without_dupes')
filtred_dataset.cache()
```

[14]:
```
DataFrame[playlistname: string, arraycol_without_dupes: array<bigint>]
```

5.3 There is the code for the FPGrowth algorithm.

I started with an high *minSupport* but in many transaction is not a good idea -> no association rules will not be found.

Then i proceed with a low minSupport, but this machine crashes after some cycles, beacuse the Tree was too big.

My last attempt was at 0.05, that's a good minSupport, with minConfidence of 0.6. I found severeal association rules, below rappresented.

[15]:
```
from pyspark.ml.fpm import FPGrowth

fpGrowth = FPGrowth(itemsCol="arraycol_without_dupes", minSupport = 0.05,␣
 ↪minConfidence = 0.6 )
model = fpGrowth.fit(filtred_dataset)

# Display frequent itemsets.
model.freqItemsets.show()

# Display generated association rules.
model.associationRules.show()

# transform examines the input items against all the association rules and␣
 ↪summarize the
# consequents as prediction
model.transform(filtred_dataset).show()
```

```
+---------+----+
|    items|freq|
+---------+----+
|   [3339]| 249|
|[1144260]| 242|
|[1731989]| 253|
|[1951486]| 245|
```

```
|  [22670]| 235|
|[1663144]| 258|
|[1731998]| 261|
|[1528087]| 239|
|  [20720]| 265|
|  [11688]| 283|
|[1299148]| 346|
| [783311]| 270|
| [477019]| 322|
|[1619510]| 289|
| [602685]| 277|
|[1283206]| 335|
| [264868]| 296|
| [703061]| 303|
|  [11490]| 311|
|    [953]| 430|
+---------+----+
only showing top 20 rows

+--------------------+----------+------------------+------------------+---------
-----------+
|          antecedent|consequent|        confidence|              lift|
support|
+--------------------+----------+------------------+------------------+---------
-----------+
|[1273233, 1198588...| [1273225]|0.9919224555735057| 5.617703171191326|
0.13077742279020235|
|[1273233, 1198588...| [1307189]|0.9983844911147012| 5.830118390278012|
0.131629392971246|
|[1273233, 1198588...| [1307182]|0.9935379644588045| 6.329254739666332|
0.13099041533546327|
|[1273233, 1198588...| [1163492]|0.9951534733441034| 6.655620452066333|
0.13120340788072418|
|[1273233, 1198588...| [1273224]|0.9951534733441034| 6.155791248156213|
0.13120340788072418|
|[1273233, 1198588...| [1273234]|0.9951534733441034| 6.850799937464172|
0.13120340788072418|
|[1273233, 1198588...| [1307139]|0.9951534733441034|  6.75179993836787|
0.13120340788072418|
|[77923, 11218, 36...| [1860817]|
0.99581589958159|13.473647402119784|0.050692225772097976|
|[1198588, 1163492...| [1273233]|0.9887278582930756| 7.130687088611352|
0.13077742279020235|
|[1198588, 1163492...|  [361336]|0.9951690821256038| 5.847708185957083|
0.131629392971246|
|[1198588, 1163492...| [1273225]|0.9967793880837359|  5.64521016532345|
0.13184238551650693|
|[1198588, 1163492...| [1307189]|               1.0|  5.83955223880597|
```

```
0.13226837060702876|
|[1198588, 1163492...| [1273231]|0.9967793880837359| 5.176857552049934|
0.13184238551650693|
|[1198588, 1163492...| [1307139]|0.9919484702093397| 6.730055011030131|
0.13120340788072418|
|[1198588, 1163492...| [1273234]|0.9935587761674718|6.8398217802144865|
0.1314164004259851|
|[1273234, 1307139...| [1273233]|0.9967373572593801|7.1884514475158054|
0.1301384451544196|
|[1273234, 1307139...| [1273232]|               1.0| 4.825282631038027|
0.13056443024494144|
|[1273234, 1307139...| [1307182]|0.9967373572593801| 6.349636217547882|
0.1301384451544196|
|[1273234, 1307139...| [1163492]|0.9951060358890701| 6.655303188745276|
0.1299254526091587|
|[1273234, 1307139...| [1273224]|0.9967373572593801| 6.165588790952291|
0.1301384451544196|
+--------------------+---------+-----------------+-----------------+---------
-----------+
only showing top 20 rows

+------------+----------------------+--------------------+
|playlistname|arraycol_without_dupes|          prediction|
+------------+----------------------+--------------------+
|       11747|  [599689, 599690, ...|[6663, 552784, 12...|
|      150290|  [57180, 65593, 59...|[1249398, 552784,...|
|       31722|  [599355, 57180, 6...|[1249398, 552784,...|
|       50570|  [1113638, 1113639...|                  []|
|       37257|  [684554, 684626, ...|                  []|
|       24777|  [496551, 496552, ...|            [658319]|
|       65907|  [314, 574, 56057,...|                  []|
|      124241|  [31509, 1708654, ...|            [552784]|
|       24424|  [650221, 878172, ...|            [658319]|
|         751|  [31507, 31508, 31...|[1249398, 669221,...|
|       11747|  [493538, 897455, ...|[467548, 1420459,...|
|       50040|  [31509, 193050, 1...|[3699, 7177, 1121...|
|       11651|  [72045, 18093, 11...|            [658319]|
|      137650|  [1312801, 434352,...|                  []|
|      126731|  [674016, 161297, ...|                  []|
|      123967|  [2651, 656371, 17...|                  []|
|       52239|  [54362, 77612, 18...|            [658319]|
|      129773|  [458567, 803497, ...|[1273228, 1249398...|
|      115178|  [31509, 1751206, ...|                  []|
|      108224|  [233650, 1351009,...|[1249398, 1273233...|
+------------+----------------------+--------------------+
only showing top 20 rows
```

I created the playlist dataset (with the playlist taken by spotipy) and applied the model.

[16]:
```
playlist_dataset = spark.createDataFrame(
    [
        ("mine", playlist_int),  # create your data here, be consistent in the␣
 ↪types.
    ],
    ["playlistname", "arraycol_without_dupes"]  # add your column names here
)
```

[26]:
```
model.transform(playlist_dataset).show(5, False)
prediction_list = model.transform(playlist_dataset).collect()
prediction_list_isolated = prediction_list[0][2]
```

```
+------------+-----------------------------------------------------------------
-----+-----------------------------------------------------+
|playlistname|arraycol_without_dupes
|prediction                                                 |
+------------+-----------------------------------------------------------------
-----+-----------------------------------------------------+
|mine        |[36279, 179946, 77361, 1249398, 1641151, 6663, 204989, 841582,
1273228]|[3699, 7177, 11218, 407814, 783254, 918033, 1860817]|
+------------+-----------------------------------------------------------------
-----+-----------------------------------------------------+
```

[19]:
```
### Import the complementary dictonary to convert int to string

with open('track_map_comp_dict.pickle','rb') as dict1:
    track_dict_comp= pickle.load( dict1)
```

[20]:
```
def convert_prediction (prediction_list, dict):
  result = []
  for i in range(len(prediction_list)):
    result.append (dict.get(str(prediction_list[i])))

  return result
```

[21]:
```
prediction_list_string = convert_prediction(prediction_list_isolated,␣
 ↪track_dict_comp)
```

[22]:
```
print (prediction_list_string)
```

```
['crawl', 'revelry', 'manhattan', 'use somebody', 'sex on fire', 'i want you',
'be somebody']
```

[41]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 40 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
```

```
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.8 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.14 [5,092 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.14 [2,265 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Ign:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.9
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Ign:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.9
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
```

```
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Err:20 http://security.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.9
  404  Not Found [IP: 91.189.88.152 80]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Err:28 http://security.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.9
  404  Not Found [IP: 91.189.88.152 80]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive all
2017.20180305-1 [14.4 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 143 MB in 10s (15.0 MB/s)
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/r/ruby2.5/ruby2.5
_2.5.1-1ubuntu1.9_amd64.deb  404  Not Found [IP: 91.189.88.152 80]
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/r/ruby2.5/libruby
2.5_2.5.1-1ubuntu1.9_amd64.deb  404  Not Found [IP: 91.189.88.152 80]
```

```
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-
missing?
Collecting pypandoc
  Downloading pypandoc-1.6.3.tar.gz (25 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-
packages (from pypandoc) (57.2.0)
Requirement already satisfied: pip>=8.1.0 in /usr/local/lib/python3.7/dist-
packages (from pypandoc) (21.1.3)
Requirement already satisfied: wheel>=0.25.0 in /usr/local/lib/python3.7/dist-
packages (from pypandoc) (0.36.2)
Building wheels for collected packages: pypandoc
  Building wheel for pypandoc (setup.py) ... done
  Created wheel for pypandoc: filename=pypandoc-1.6.3-py3-none-any.whl
size=17315
sha256=a8017685b1016b989d478a51b41a4a751618e86e4998d0a7a57322fd591f3d6c
  Stored in directory: /root/.cache/pip/wheels/c2/66/cc/3ecb77dd76fd266946a70bb8
0b67fef8b89abf0362dabe1ad3
Successfully built pypandoc
Installing collected packages: pypandoc
Successfully installed pypandoc-1.6.3
```

[42]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!cp drive/My Drive/Colab Notebooks/pyspark_Start.ip ./
```