Abhishek Patwardhan

D17A - 57

SMA Experiment 5

Importing Libraries and Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime


import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv("Spotify.csv")
df.head()
```

	Profile Link	Profile Ima
0	https://www.google.com/maps/contrib/1097199965...	https://lh3.googleusercontent.com/a-/ACB-R5TLG
1	https://www.google.com/maps/contrib/1179812806...	https://lh3.googleusercontent.com/a-/ACB-R5T_c
2	https://www.google.com/maps/contrib/1013997448...	https://lh3.googleusercontent.com/a-/AC R5Rm_
3	https://www.google.com/maps/contrib/1022967930...	https://lh3.googleusercontent.com/a-/ACB-R5Re3
4	https://www.google.com/maps/contrib/1164622418...	https://lh3.googleusercontent.com/a/AGNmyxZi_\

Cleaning The Dataset

==> Removing the words "review" or "reviews" from the reviewCount column

```
df1 = df
df1['Review Count'] = df1['Review Count'].str.replace(r'\D+', '', regex=True)
df1['Review Count'] = df1['Review Count'].replace(np.nan, 1)


df1['Review Count'] = df1['Review Count'].astype(int)
```

==> Changing the Stars URLs to proper numeric values and adding them to give the final start count

```
# create a dictionary to map string values to integer values
string_to_num = {'https://maps.gstatic.com/consumer/images/icons/2x/ic_star_rate_14.png': 1,
 'https://maps.gstatic.com/consumer/images/icons/2x/ic_star_rate_empty_14.png': 0}
```

```
# map the 'stars' column to integer values using the dictionary
df1['ReviewStar'] = (df1['Star 1'].map(string_to_num) +
 df1['Star 2'].map(string_to_num) +
 df1['Star 3'].map(string_to_num) +
 df1['Star 4'].map(string_to_num) +
 df1['Star 5'].map(string_to_num))


col = df1.pop('ReviewStar')
df1.insert(3, 'ReviewStars', col)
df1 = df1.drop(['Star 1', 'Star 2', 'Star 3', 'Star 4', 'Star 5'], axis=1)
```

==> Mapping the string date values to their corresponding date values (measured strictly,from today)

```
df1['Date'].unique()
```

```
     array(['2 months ago', '5 years ago', '6 months ago', '3 years ago',
            '5 months ago', 'a year ago', '9 months ago', '4 months ago',
            '10 months ago', '11 months ago', 'a month ago', '4 years ago',
            '2 years ago', '9 years ago', '6 years ago', '11 years ago',
            '7 years ago', '3 months ago', '7 months ago', 'a week ago',
            '3 weeks ago', '8 years ago', '8 months ago'], dtype=object)
```

```
from datetime import timedelta
from dateutil.relativedelta import relativedelta


def parse_relative_time(s):
 s = s.split()
 if s[0] == 'a':
   n = 1
 else:
   n = int(s[0])
 if s[1].startswith('month'):
   delta = relativedelta(months=-n)
 elif s[1].startswith('year'):
   delta = relativedelta(years=-n)
 elif s[1].startswith('week'):
   delta = timedelta(weeks=-n)
 else:
   delta = timedelta(days=-n)
 return datetime.today() + delta


df1['Date'] = df1['Date'].apply(parse_relative_time)
```

==> Filling the null values in each column

```
# Replacing the NaN values of review column with NA value

df1['Review'] = df1['Review'].replace(np.nan, "NA")
```

==> Exporting the Refined Dataset

```
df1.head()
```

	Profile Link	Profile Image	Name
0	https://www.google.com/maps/contrib/1097199965...	https://lh3.googleusercontent.com/a-/ACB-R5TLG...	Calewan
1	https://www.google.com/maps/contrib/1179812806...	https://lh3.googleusercontent.com/a-/ACB-R5T_o...	Oleksandr Kucherenko

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 399 entries, 0 to 398
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Profile Link   399 non-null    object        
 1   Profile Image  399 non-null    object        
 2   Name           399 non-null    object        
 3   ReviewStars    399 non-null    int64         
 4   Designation    354 non-null    object        
 5   Review Count   399 non-null    int64         
 6   Date           399 non-null    datetime64[ns]
 7   Review         399 non-null    object        
dtypes: datetime64[ns](1), int64(2), object(5)
memory usage: 25.1+ KB
```

==> Results of EDA

```
df1.describe()
```

	ReviewStars	Review Count
count	399.000000	399.000000
mean	4.228070	65.829574
std	1.383904	146.147482
min	1.000000	1.000000
25%	4.000000	2.000000
50%	5.000000	15.000000
75%	5.000000	59.000000
max	5.000000	1731.000000

Sentiment Analysis

```
df3 = df1
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.downloader.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
sia = SentimentIntensityAnalyzer()
```

```
df3['sentiment'] = df3['Review'].apply(lambda x: sia.polarity_scores(x)['compound'])
```

```
print(f"Mean Sentiment Score: {df3['sentiment'].mean()}")
```

```
Mean Sentiment Score: 0.11109598997493735
```

```
# Print the sentiment label for each row
def get_sentiment_label(score):
 if score >= 0.05:
   return 'Positive'
 elif score <= -0.05:
   return 'Negative'
 else:
   return 'Neutral'


df3['sentiment_label'] = df3['sentiment'].apply(get_sentiment_label)
print(df3['sentiment_label'].value_counts())
```

```
Neutral     288
Positive     92
Negative     19
Name: sentiment_label, dtype: int64
```

Dicrepancy Observed

Here, due to the fact that the exact date of the review being posted is unavailable, but the strings give information like "a week ago, a month ago, 4 years ago" hence numerous multiple values which lie in a certain time period (say 4 years ago) are all mapped to the same year, month, or week (if today is March 10, then all "4 years ago" values will be mapped to the March 10 of that year)

```
plt.figure(figsize=(16,5))
# plt.scatter(df3.index, df3['sentiment'])
plt.plot(df3.index, df3['sentiment'])
plt.title("Sentiment Variation")
plt.xlabel('Index')
plt.ylabel('Sentiment Score')
plt.show()
```