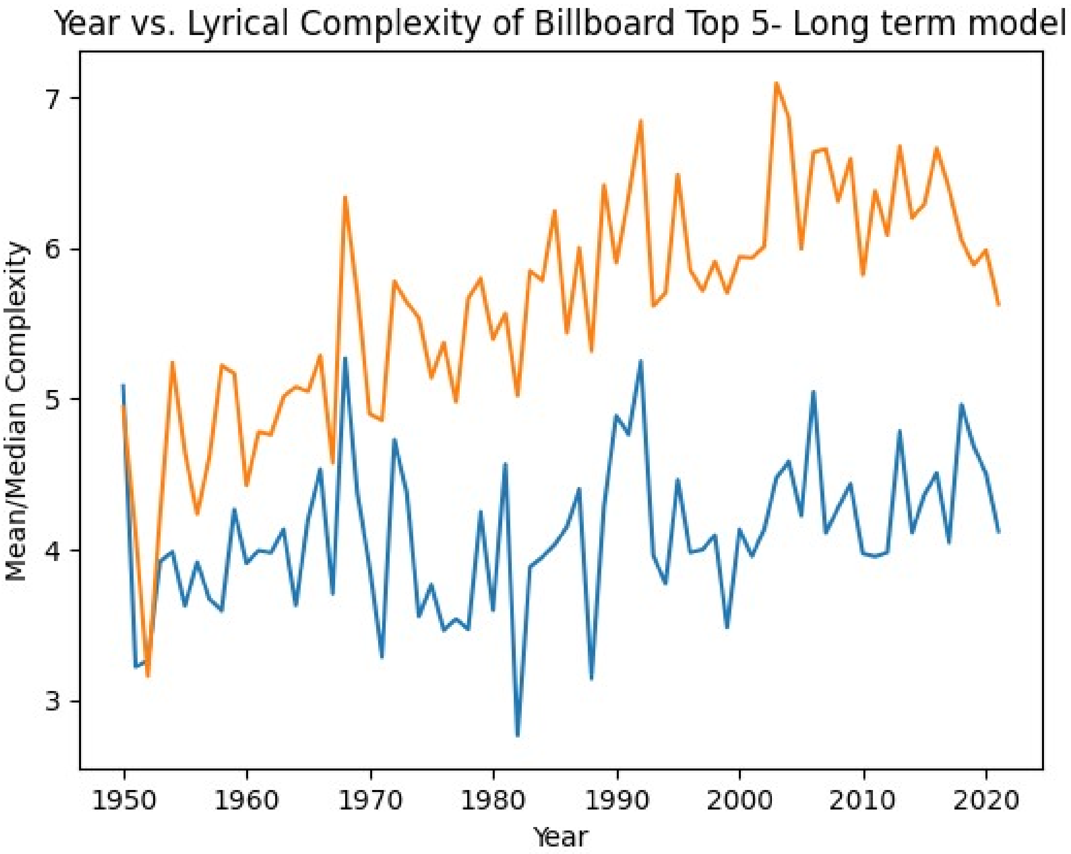
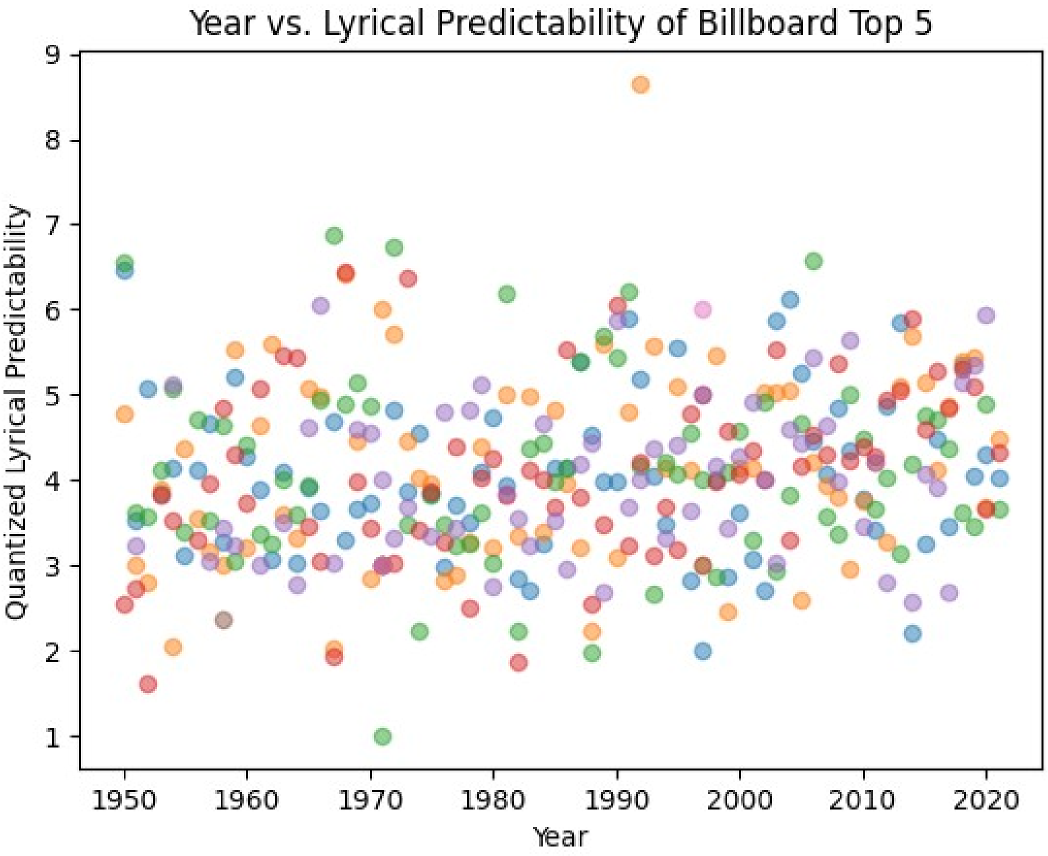
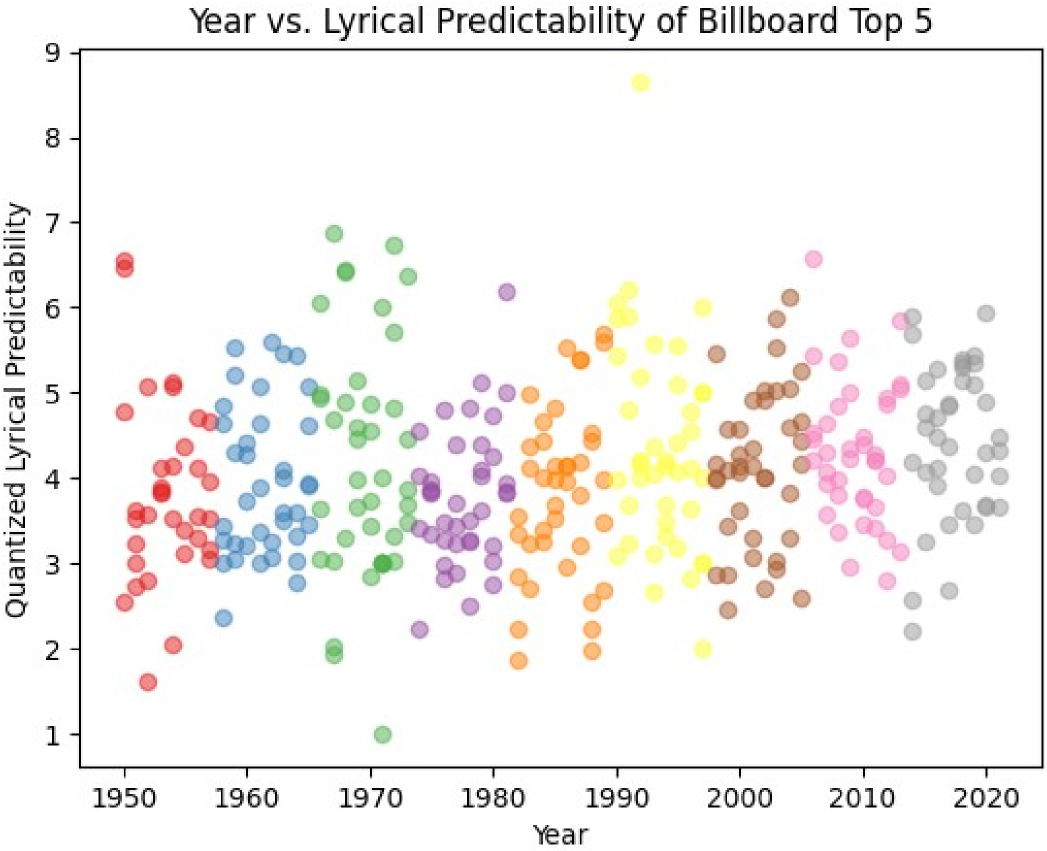
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

# 360 in bimmuda_lyrics
# 352 in quantizations - removes issue files and has incomplete years
pth = "/Users/rogerburtonpatel/home/london/bimmuda/quantizations-long"
c  = "ltmkplus.csv"
c2  = "fst-data.csv"

def mk_df(path, csv):
    file_list = os.listdir(path)

    # sort into buckets based on year
    file_list.sort()
    buckets = {}
    bucket = []
    for file in file_list:
        file_year = file[10:14]
        if not file_year in buckets:
            bucket = []
            buckets[file_year] = bucket
        buckets[file_year].append(file)
    # print(buckets)

    # keep around for debugging
    # print(len([v for value in buckets.values() for v in value]))

    # populate 'valid' slots (i.e. there exist data points) with 0s,
else NaN
    max_data_points = max([len(buckets[year]) for year in buckets])
    years_and_data = {year: np.array(
                                      [0 if i < len(buckets[year])
                                         else float('nan')
                                       for i in
range(max_data_points)])
                      for year in buckets}

    with open(csv, 'r') as f:
        for line in f:
            all_data = line.split()

    # we'll fill in the zeroes with the data, skipping NaN slots
    idx = 0
    for year in years_and_data.keys():
        for i in range(len(years_and_data[year])):
            if years_and_data[year][i] == 0:
                years_and_data[year][i] = float(all_data[idx])
                idx += 1
    df = pd.DataFrame.from_dict(years_and_data)

    # guard against data loss
    assert(np.sum(df.count()) == len(file_list) == len(all_data))
    return df

df = mk_df(pth, c)
df2 = mk_df(pth, c2)

%matplotlib inline


means = df.mean()
means2 = df2.mean()
# medians = df.median()

# Create plot
fig, ax = plt.subplots()

ax.plot(means.index.astype(int), means.values)
# Uncomment to show medians:
ax.plot(means2.index.astype(int), means2.values)

ax.set_xlabel('Year')
ax.set_ylabel('Mean/Median Complexity')
ax.set_title('Year vs. Lyrical Complexity of Billboard Top 5- Long
term model')

plt.show()
```



```
fig, ax = plt.subplots()
for index, row in df.iterrows():
    ax.scatter([int(year) for year in row.index], row.values,
alpha=0.5)

ax.set_xlabel('Year')
ax.set_ylabel('Quantized Lyrical Predictability')
ax.set_title('Year vs. Lyrical Predictability of Billboard Top 5')

plt.show()
```



```
# Define color map and colors
cmap = plt.colormaps.get_cmap('Set1')
colors = np.linspace(0, 1, len(df.columns))

fig, ax = plt.subplots()
for i, (column, color) in enumerate(zip(df.columns, colors)):
    ax.scatter([int(column)] * len(df), df[column], color=cmap(color),
alpha=0.5)

ax.set_xlabel('Year')
ax.set_ylabel('Quantized Lyrical Predictability')
ax.set_title('Year vs. Lyrical Predictability of Billboard Top 5')

plt.show()
```


Year vs. Lyrical Predictability of Billboard Top 5
Quantized Lyrical Predictability
9
8
7
6
5
4
3
2
1
1950
1960
1970
1980
1990
2000
2010
2020
Year