Song Qualities

The Spotify API is a service provided by Spotify that makes it possible for users to easily draw data about the songs on Spotify.

In addition to information about the album, artists, and so on, Spotify also provides some measures of the audio features of each song. This includes things like how loud the song is, how long it lasts, and what time signature it is in; as well as estimates of more subjective measures like the "acousticness" or "danceability".

Here you can find a dataset containing 100 songs each from 14 different artists. This data represents a small subest of the data that has been pulled from the Spotify API and put on Kaggle for you here; you can also find more in-depth descriptions of the song features at that Kaggle site.

Our goal in this lab will be to construct models with interesting interpretations from these audio features.

Question 0

Read and clean the data.

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

df_songs = pd.read_csv("https://www.dropbox.com/s/hijzbof7nnche09/top_artists_
df_songs
```

Out[1]:

	Acousticness	Danceability	Duration	Energy	Explicit	Instrumentalness	Liveness	Loudr
0	0.631000	0.399	213492	0.491	True	0.000000	0.1710	-8.
1	0.099900	0.758	230484	0.888	True	0.033600	0.6610	-4.
2	0.161000	0.529	195720	0.690	True	0.000000	0.4980	-7
3	0.528000	0.766	188703	0.531	False	0.000665	0.1100	-10.
4	0.042900	0.828	211643	0.482	False	0.000008	0.0807	-8.
...	...	...	...	...	...	...	...	
1395	0.407000	0.526	273467	0.665	False	0.000271	0.1690	-4
1396	0.002650	0.652	225400	0.783	False	0.000002	0.2680	-6
1397	0.000266	0.368	293733	0.631	False	0.055900	0.0682	-12
1398	0.751000	0.536	35973	0.229	False	0.000000	0.6290	-24
1399	0.410000	0.545	278893	0.947	False	0.000705	0.0385	-7.

1400 rows × 13 columns

In [2]:
```
df_new = df_songs.drop('Explicit', axis="columns")
df_new
```

Out[2]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness	Mo
0	0.631000	0.399	213492	0.491	0.000000	0.1710	-8.992	
1	0.099900	0.758	230484	0.888	0.033600	0.6610	-4.899	
2	0.161000	0.529	195720	0.690	0.000000	0.4980	-7.870	
3	0.528000	0.766	188703	0.531	0.000665	0.1100	-10.899	
4	0.042900	0.828	211643	0.482	0.000008	0.0807	-8.469	
...	...	...	...	...	...	...	...	
1395	0.407000	0.526	273467	0.665	0.000271	0.1690	-4.997	
1396	0.002650	0.652	225400	0.783	0.000002	0.2680	-6.706	
1397	0.000266	0.368	293733	0.631	0.055900	0.0682	-12.191	
1398	0.751000	0.536	35973	0.229	0.000000	0.6290	-24.760	
1399	0.410000	0.545	278893	0.947	0.000705	0.0385	-7.456	

1400 rows × 12 columns

Question 1

Fit a *Logistic Regression* model to predict whether or not a song will contain Explicit language or not. Report the confusion matrix of the resulting model.

(Hint: Add `solver="newton-cg"` as an argument to your `LogisticRegression()` function, to avoid a warning.)

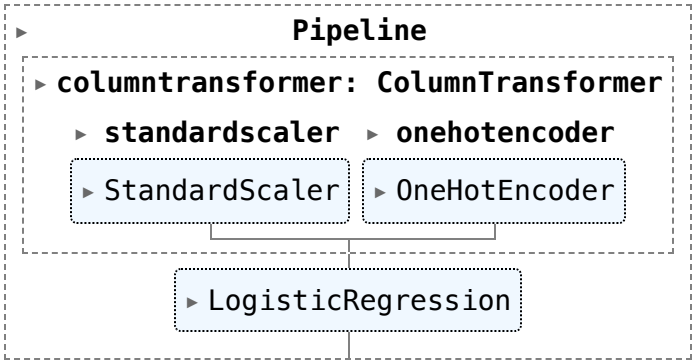
In [3]:
```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
```

In [4]:
```
## YOUR CODE HERE

ct = make_column_transformer(
    (StandardScaler(), ["Acousticness", "Danceability", "Duration", "Energy", '
    (OneHotEncoder(), ["Mode", "TimeSignature"]),
    remainder="drop"
)
pipeline = make_pipeline(
  ct,
  LogisticRegression(penalty=None, solver='newton-cg'))

pipeline.fit(df_new, df_songs['Explicit'])
```

Out[4]:



In [5]:
```
pred_explicit = pipeline.predict(df_new)
pred_explicit
```

Out[5]:
```
array([False, False,  True, ..., False, False, False])
```

In [6]:
```
pd.DataFrame(pred_explicit)
```

Out[6]:

	0
0	False
1	False
2	True
3	True
4	True
...	...
1395	False
1396	False
1397	False
1398	False
1399	False

1400 rows × 1 columns

In [7]:
```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(df_songs['Explicit'], pred_explicit), columns =
```

Out[7]:

	False	True
False	900	71
True	218	211

Question 2

Print out a data frame showing the *coefficient* estimates for each feature.

What **categorical** feature was most important, and what do the coeffient estimates tell you about the categories?

What **quantitative** feature was most important, and what does the coefficient estimate tell you?

In [8]:
```
df_new
```

Out[8]:

	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness	Mo
0	0.631000	0.399	213492	0.491	0.000000	0.1710	-8.992	
1	0.099900	0.758	230484	0.888	0.033600	0.6610	-4.899	
2	0.161000	0.529	195720	0.690	0.000000	0.4980	-7.870	
3	0.528000	0.766	188703	0.531	0.000665	0.1100	-10.899	
4	0.042900	0.828	211643	0.482	0.000008	0.0807	-8.469	
...	...	...	...	...	...	...	...	
1395	0.407000	0.526	273467	0.665	0.000271	0.1690	-4.997	
1396	0.002650	0.652	225400	0.783	0.000002	0.2680	-6.706	
1397	0.000266	0.368	293733	0.631	0.055900	0.0682	-12.191	
1398	0.751000	0.536	35973	0.229	0.000000	0.6290	-24.760	
1399	0.410000	0.545	278893	0.947	0.000705	0.0385	-7.456	

1400 rows × 12 columns

In [9]:
```
pipeline.named_steps["logisticregression"].coef_
```

Out[9]:
```
array([[-0.16121819,  0.65967116,  0.05076077,  0.02682568, -0.09661532,
        -0.13878952,  0.35057856,  0.96317377,  0.01737785, -0.32006564,
        -0.59753742, -0.49560583, -6.27361678,  2.87149527,  0.4225499 ,
         0.57882983,  1.30759852]])
```

In [10]:
```
## YOUR CODE HERE

# Hint: use the code below to find the transformed feature names
# pipeline.named_steps['columntransformer'].get_feature_names_out()

coefficients = pd.DataFrame({
  "Coefficients": pipeline['logisticregression'].coef_[0],
  "Column": pipeline.named_steps['columntransformer'].get_feature_names_out()
  })

coefficients.sort_values(by="Coefficients", ascending=True)
```

Out[10]:

	Coefficients	Column
12	-6.273617	onehotencoder__TimeSignature_0
10	-0.597537	onehotencoder__Mode_0
11	-0.495606	onehotencoder__Mode_1
9	-0.320066	standardscaler__Valence
0	-0.161218	standardscaler__Acousticness
5	-0.138790	standardscaler__Liveness
4	-0.096615	standardscaler__Instrumentalness
8	0.017378	standardscaler__Tempo
3	0.026826	standardscaler__Energy
2	0.050761	standardscaler__Duration
6	0.350579	standardscaler__Loudness
14	0.422550	onehotencoder__TimeSignature_3
15	0.578830	onehotencoder__TimeSignature_4
1	0.659671	standardscaler__Danceability
7	0.963174	standardscaler__Speechiness
16	1.307599	onehotencoder__TimeSignature_5
13	2.871495	onehotencoder__TimeSignature_1

-YOUR INTERPRETATION HERE-

Based on the values of the coefficients presented in the above DataFrame, the categorical variable that was most important in determining whether a song is explicit or not is Time Signature. When the value of the TimeSignature increases by 1, the log-odds of a song being explicit increases by approximately 2.87, which means that faster songs are likely more explicit. On the other hand, the other categorical variable of Mode had a very low coefficient, meaning that when the Mode is 1, the log-odds of a song being explicit decrease by approximately 0.49.

Regarding the quantitative features, the Speechiness of the song has the most impact on whether it is explicit or not, with a coefficient of approximately 0.96 indicating that the log-odds of a song being explicit increase by about 0.96 when the Speechiness score increases by 1.

Question 3

If we were to use this model in the future, to predict whether a song has explicit language based on its audio qualities, what **accuracy** would we probably expect to get? What **f1 score**?

In [11]:
```
## YOUR CODE HERE
X_train = df_new
y_train = df_songs["Explicit"]
cross_val_score(
    pipeline,
    X=X_train, y=y_train,
    scoring="accuracy",
    cv=10
).mean()
```

Out[11]:
```
0.7735714285714287
```

In [12]:
```
cross_val_score(
    pipeline,
    X=X_train, y=y_train,
    scoring="f1_macro",
    cv=10
).mean()
```

Out[12]:
```
0.6877275532488859
```

Question 4

Here you will find a dataset with only one row, consisting of the audio features for all songs (as of 2019) by the band Twenty One Pilots.

Use your model to predict whether these songs will contain explicit language. Did it correctly predict? Why do you think that is?

In [13]:
```
top = pd.read_csv("https://www.dropbox.com/scl/fi/j9yz0nqnm4d0sqafineyb/twenty-
top
## YOUR CODE HERE
```

Out[13]:

	Name	X1	Album	Artist	Acousticness	Danceability	Duration	Energy	Explicit	Ins
0	Ode to Sleep	0	Vessel (with Bonus Tracks)	twenty one pilots	0.00891	0.446	308388	0.760	False	
1	Holding on to You	1	Vessel (with Bonus Tracks)	twenty one pilots	0.24100	0.608	263733	0.776	False	
2	Migraine	2	Vessel (with Bonus Tracks)	twenty one pilots	0.11000	0.690	239015	0.678	False	
3	House of Gold	3	Vessel (with Bonus Tracks)	twenty one pilots	0.44500	0.680	163880	0.693	False	
4	Car Radio	4	Vessel (with Bonus Tracks)	twenty one pilots	0.09800	0.766	267720	0.519	False	
...	...	...	...	...	...	...	...	...	...	
65	The Run and Go - Bonus Commentary	65	Vessel	twenty one pilots	0.75200	0.746	86844	0.121	False	
66	Fake You Out - Bonus Commentary	66	Vessel	twenty one pilots	0.74600	0.624	63467	0.205	False	
67	Guns for Hands - Bonus Commentary	67	Vessel	twenty one pilots	0.65200	0.677	204575	0.201	False	
68	Trees - Bonus Commentary	68	Vessel	twenty one pilots	0.58400	0.683	64525	0.182	False	
69	Truce - Bonus Commentary	69	Vessel	twenty one pilots	0.69200	0.731	124184	0.232	False	

70 rows × 17 columns

In [14]:
```
top_clean = top.drop("Name", axis="columns")
top_clean = top_clean.drop("Album", axis="columns")
top_clean = top_clean.drop("Explicit", axis="columns")
top_clean = top_clean.drop("Artist", axis="columns")
top_clean
```

Out[14]:

	X1	Acousticness	Danceability	Duration	Energy	Instrumentalness	Liveness	Loudness	M
0	0	0.00891	0.446	308388	0.760	0.000108	0.2320	-3.608	
1	1	0.24100	0.608	263733	0.776	0.000000	0.0713	-3.635	
2	2	0.11000	0.690	239015	0.678	0.000000	0.2460	-6.520	
3	3	0.44500	0.680	163880	0.693	0.000000	0.2080	-7.234	
4	4	0.09800	0.766	267720	0.519	0.000000	0.0855	-7.355	
...	...	...	...	...	...	...	...	...	...
65	65	0.75200	0.746	86844	0.121	0.000000	0.8420	-26.296	
66	66	0.74600	0.624	63467	0.205	0.000000	0.4360	-27.435	
67	67	0.65200	0.677	204575	0.201	0.000000	0.1270	-27.235	
68	68	0.58400	0.683	64525	0.182	0.000000	0.2620	-29.028	
69	69	0.69200	0.731	124184	0.232	0.000000	0.3210	-27.719	

70 rows × 13 columns

In [15]:
```
pipeline.predict(top_clean)
```

Out[15]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

Looking at the Twenty-One Pilots data and comparing it to the predictions, it appears that my pipeline predicted 56/70 (80%) of the explicitness of the songs correctly.

Submission Instructions

- Restart this notebook and run the cells from beginning to end.
  - Go to Runtime > Restart and Run All.

In [ ]:
```
# @markdown Run this cell to download this notebook as a webpage, `_NOTEBOOK.h

import google, json, nbformat

# Get the current notebook and write it to _NOTEBOOK.ipynb
raw_notebook = google.colab._message.blocking_request("get_ipynb",
                                                      timeout_sec=30)["ipynb"]
with open("_NOTEBOOK.ipynb", "w", encoding="utf-8") as ipynb_file:
  ipynb_file.write(json.dumps(raw_notebook))

# Use nbconvert to convert .ipynb to .html.
!jupyter nbconvert --to html --log-level WARN _NOTEBOOK.ipynb
```

```
# Download the .html file.
google.colab.files.download("_NOTEBOOK.html")
```

- Open `_NOTEBOOK.html` in your browser, and save it as a PDF.
  - Go to File > Print > Save as PDF.
- Double check that all of your code and output is visible in the saved PDF.
- Upload the PDF to Gradescope.
  - Please be sure to select the correct pages corresponding to each question.