Model Creation

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [ ]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:
```
#Read datasets and add 'isLiked' column with default values to both dataframes
liked = pd.read_csv("/content/drive/MyDrive/MusicMachineLearningProject/databa
disliked = pd.read_csv("/content/drive/MyDrive/MusicMachineLearningProject/dat
liked["isLiked"] = True
disliked["isLiked"] = False
```

In [ ]:
```
#Drop any rows with missing values (ie. local files)
liked.dropna(inplace=True)
disliked.dropna(inplace=True)
```

In [ ]:
```
#Merge the dataframes, drop duplicate songs
merged = pd.concat([liked,disliked])
merged.drop_duplicates(subset=["Spotify ID"], inplace=True)
merged.reset_index(drop=True, inplace=True)
```

In [ ]:
```
#Split Data
X = merged.drop(['Spotify ID', 'Artist IDs', 'Track Name', 'Album Name',
       'Artist Name(s)', 'Release Date', 'Duration (ms)', 'Instrumentalness',
y = merged['isLiked']

#Create scaler for normalization
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

#Train model using logistic regression and test size of 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
model = LogisticRegression(random_state=101, max_iter=1000)
model.fit(X_train, y_train)

#Create a confusion matrix
predictions = model.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, predictions)
cnf_matrix
```

Out[23]:
```
array([[27, 31],
       [ 8, 87]])
```

In [ ]:
```
#Display report
target_names = ['Disliked', 'Liked']
print(metrics.classification_report(y_test, predictions, target_names=target_
```

```
              precision    recall  f1-score   support

    Disliked       0.77      0.47      0.58        58
       Liked       0.74      0.92      0.82        95

    accuracy                           0.75       153
   macro avg       0.75      0.69      0.70       153
weighted avg       0.75      0.75      0.73       153
```

The model has an accuracy of 75%. That is, 75% of all songs evaluated were evaluated correctly. This is higher than I really imagined. I was expecting an accuracy of 40%-60%, close to that of a coin flip, but the model seems to have exceeded that. Soon I hope to evaluate this model against two more datasets. First, a cumulative playlist of all my friends (Rap being likely the most popular here along side branches into several other, distinct genres). The goal here will be to simply see how good the model is a picking out new songs and see if it is viable to be a time saver. Second, is to run it against a playlist of my sister, moslty just for fun and curiosity. The model has very little data from the genres she listens to so I am curious both what it would do with this data that largely doesn't conform to the "liked" or "disliked" data as well as see if it is still able to pick out songs that I would like

Additionally I hope to improve this number potentially using the artist names or seeking out another model type. I believe that 80% accuracy would be impressive for the nature of the data as simply being much higher than I could expect.