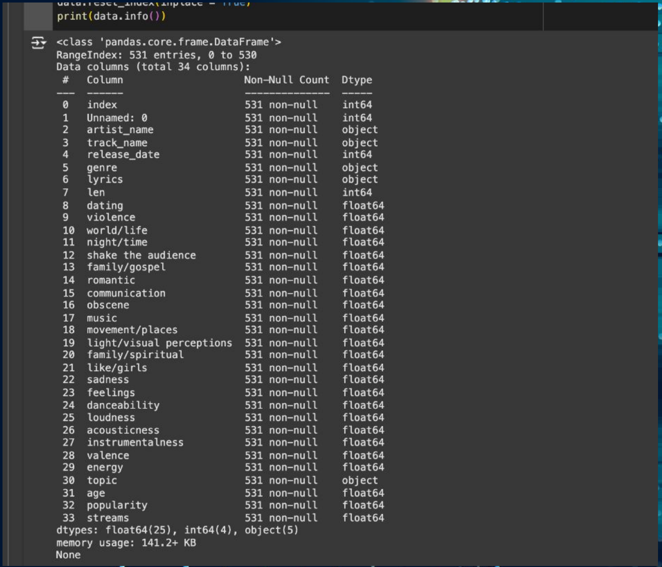
Record Label Analysis

Why Care ?

Gain insight into what kind of artists a record label should sign/put money into by assessing the type of music that performed well in the year 2019. Models will test the relevance certain variables have when predicting popularity and a hit.

Introduction to Data

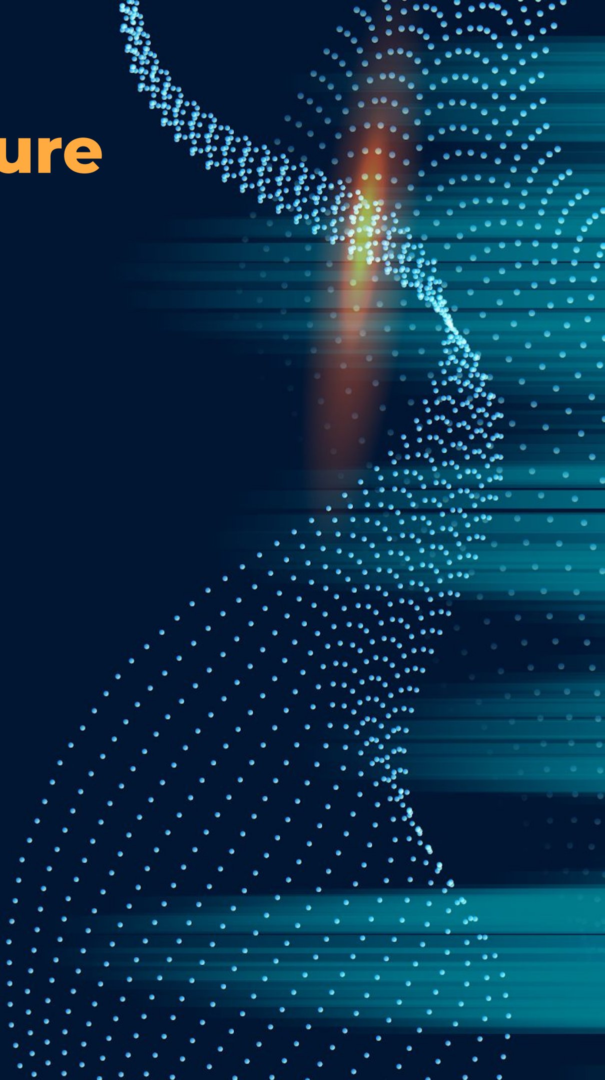
- **Data:**
  - List of most popular songs from 1950-2019
    - Sentiment Analysis
  - Two Columns Added:
    - Popularity - Spotify API
    - Streams - Manually (Youtube API had vague restrictions)



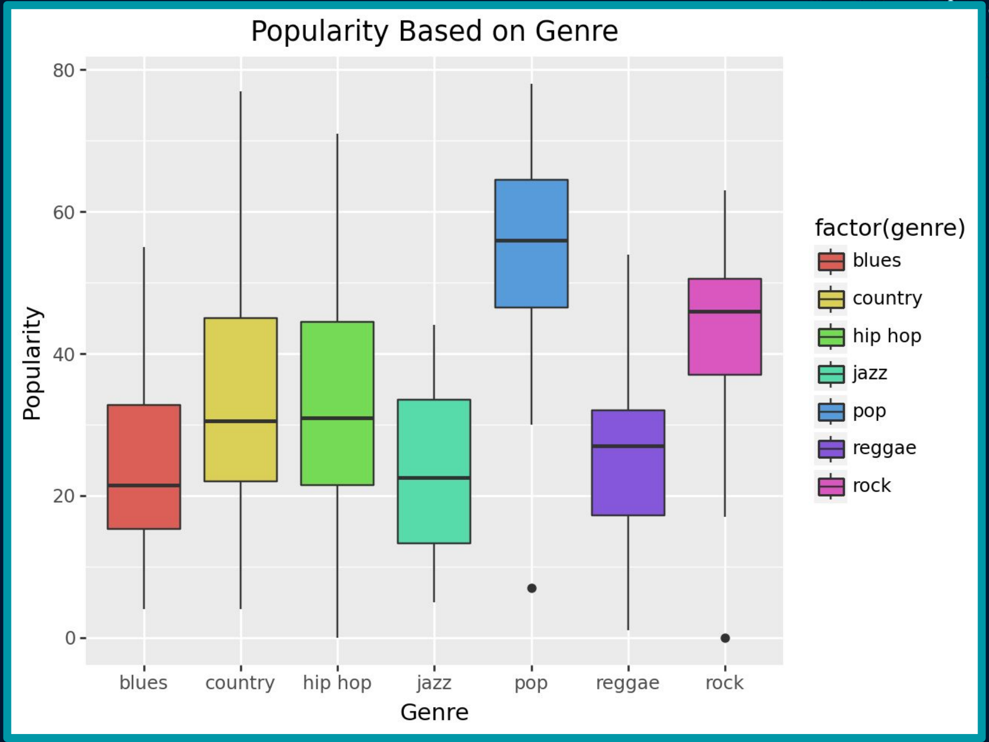
```
print(data.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 531 entries, 0 to 530
Data columns (total 34 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   index                     531 non-null    int64
 1   Unnamed: 0                531 non-null    int64
 2   artist_name               531 non-null    object
 3   track_name                531 non-null    object
 4   release_date              531 non-null    int64
 5   genre                     531 non-null    object
 6   lyrics                    531 non-null    object
 7   len                       531 non-null    int64
 8   dating                    531 non-null    float64
 9   violence                  531 non-null    float64
 10  world/life                531 non-null    float64
 11  night/time                531 non-null    float64
 12  shake the audience        531 non-null    float64
 13  family/gospel             531 non-null    float64
 14  romantic                  531 non-null    float64
 15  communication             531 non-null    float64
 16  obscene                   531 non-null    float64
 17  music                     531 non-null    float64
 18  movement/places           531 non-null    float64
 19  light/visual perceptions  531 non-null    float64
 20  family/spiritual          531 non-null    float64
 21  like/girls                531 non-null    float64
 22  sadness                   531 non-null    float64
 23  feelings                  531 non-null    float64
 24  danceability              531 non-null    float64
 25  loudness                  531 non-null    float64
 26  acousticness              531 non-null    float64
 27  instrumentalness          531 non-null    float64
 28  valence                   531 non-null    float64
 29  energy                    531 non-null    float64
 30  topic                     531 non-null    object
 31  age                       531 non-null    float64
 32  popularity                531 non-null    float64
 33  streams                   531 non-null    float64
dtypes: float64(25), int64(4), object(5)
memory usage: 141.2+ KB
None
```

Summary of Data Cleaning/Feature Engineering

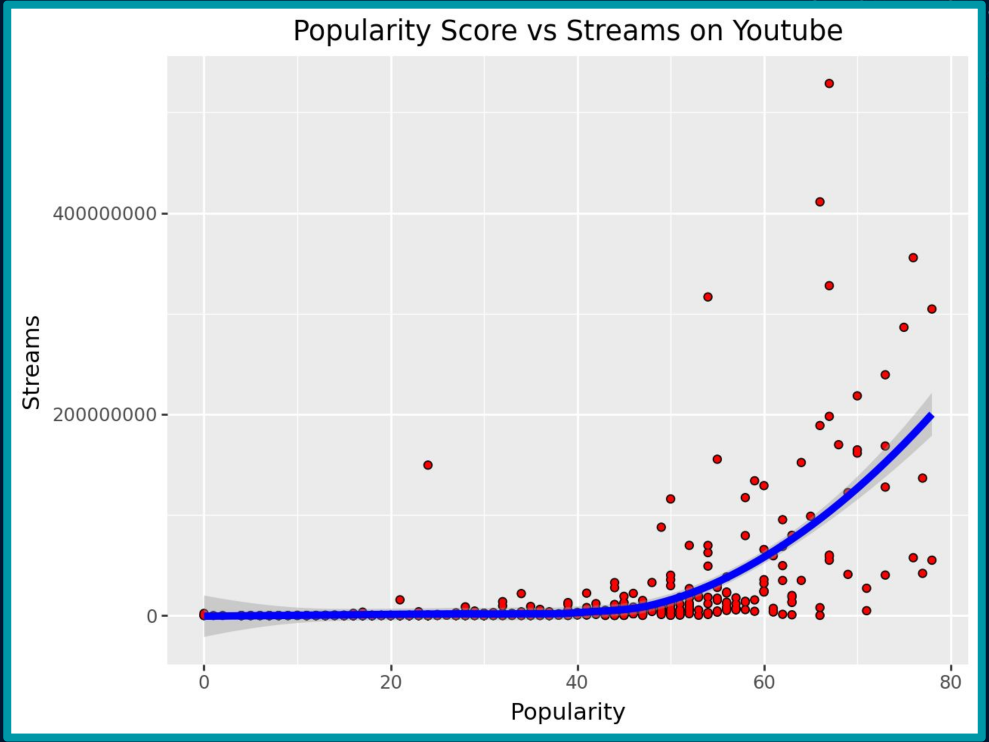
1. Cut all data from songs not in 2019
2. Separate Features from Popularity (Hit for Logistic)
   a. >=1mn
3. Encode Categorical, Scale Continuous
4. Transform Features
5. Split Data



Summary Plot 1



Summary Plot 2

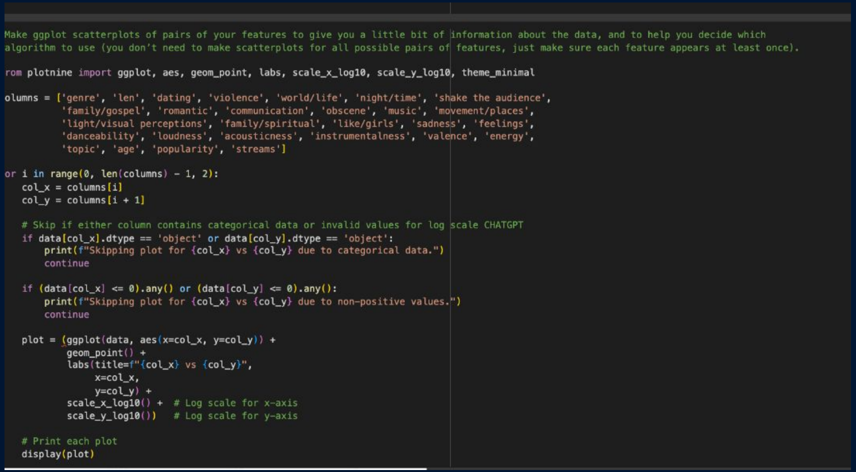


Models

Beta Regression, Clustering, ElasticNet, PCA

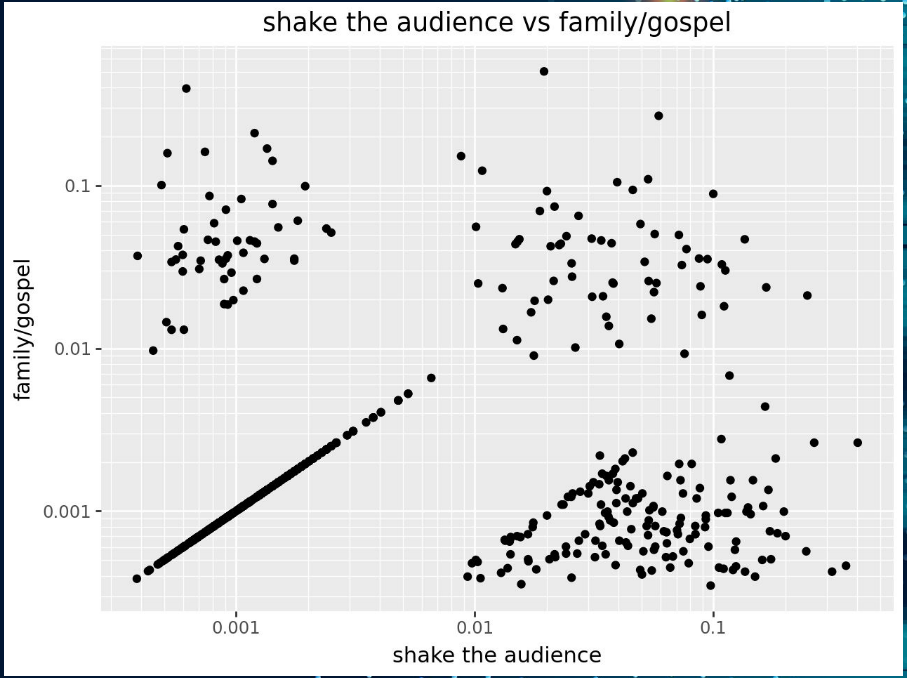
Clustering Model

- **What is Clustering?**
  - Find groups in data
- How to determine best type of model?



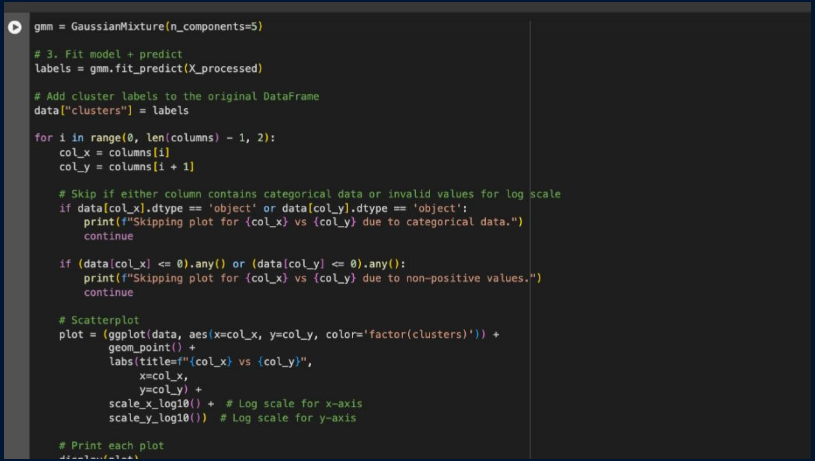
Make ggplot scatterplots of pairs of your features to give you a little bit of information about the data, and to help you decide which algorithm to use (you don't need to make scatterplots for all possible pairs of features, just make sure each feature appears at least once).

```
rom plotnine import ggplot, aes, geom_point, labs, scale_x_log10, scale_y_log10, theme_minimal

olumns = ['genre', 'len', 'dating', 'violence', 'world/life', 'night/time', 'shake the audience',
          'family/gospel', 'romantic', 'communication', 'obscene', 'music', 'movement/places',
          'light/visual perceptions', 'family/spiritual', 'like/girls', 'sadness', 'feelings',
          'danceability', 'loudness', 'acousticness', 'instrumentalness', 'valence', 'energy',
          'topic', 'age', 'popularity', 'streams']

or i in range(0, len(columns) - 1, 2):
   col_x = columns[i]
   col_y = columns[i + 1]

   # Skip if either column contains categorical data or invalid values for log scale CHATGPT
   if data[col_x].dtype == 'object' or data[col_y].dtype == 'object':
       print(f"Skipping plot for {col_x} vs {col_y} due to categorical data.")
       continue

   if (data[col_x] <= 0).any() or (data[col_y] <= 0).any():
       print(f"Skipping plot for {col_x} vs {col_y} due to non-positive values.")
       continue

   plot = (ggplot(data, aes(x=col_x, y=col_y)) +
           geom_point() +
           labs(title=f"{col_x} vs {col_y}",
                x=col_x,
                y=col_y) +
           scale_x_log10() +  # Log scale for x-axis
           scale_y_log10())   # Log scale for y-axis

   # Print each plot
   display(plot)
```

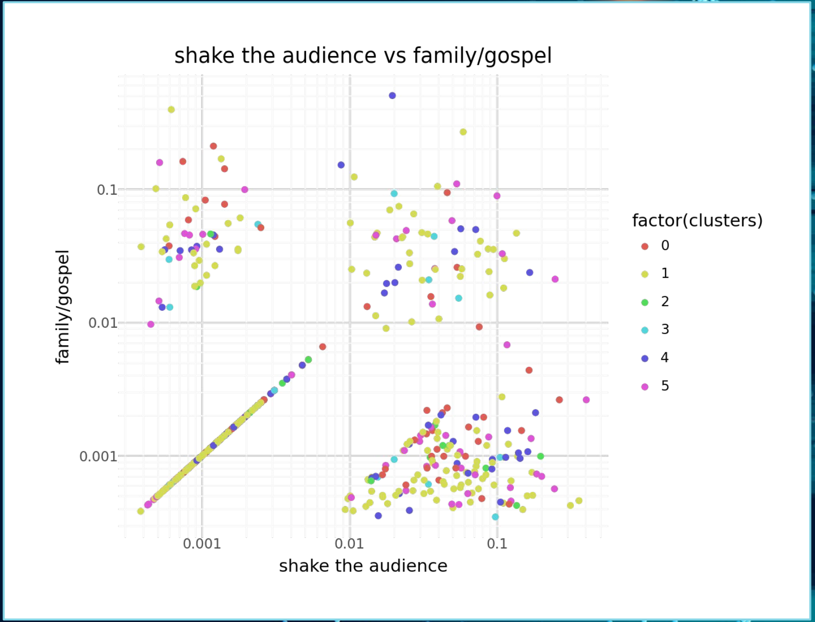


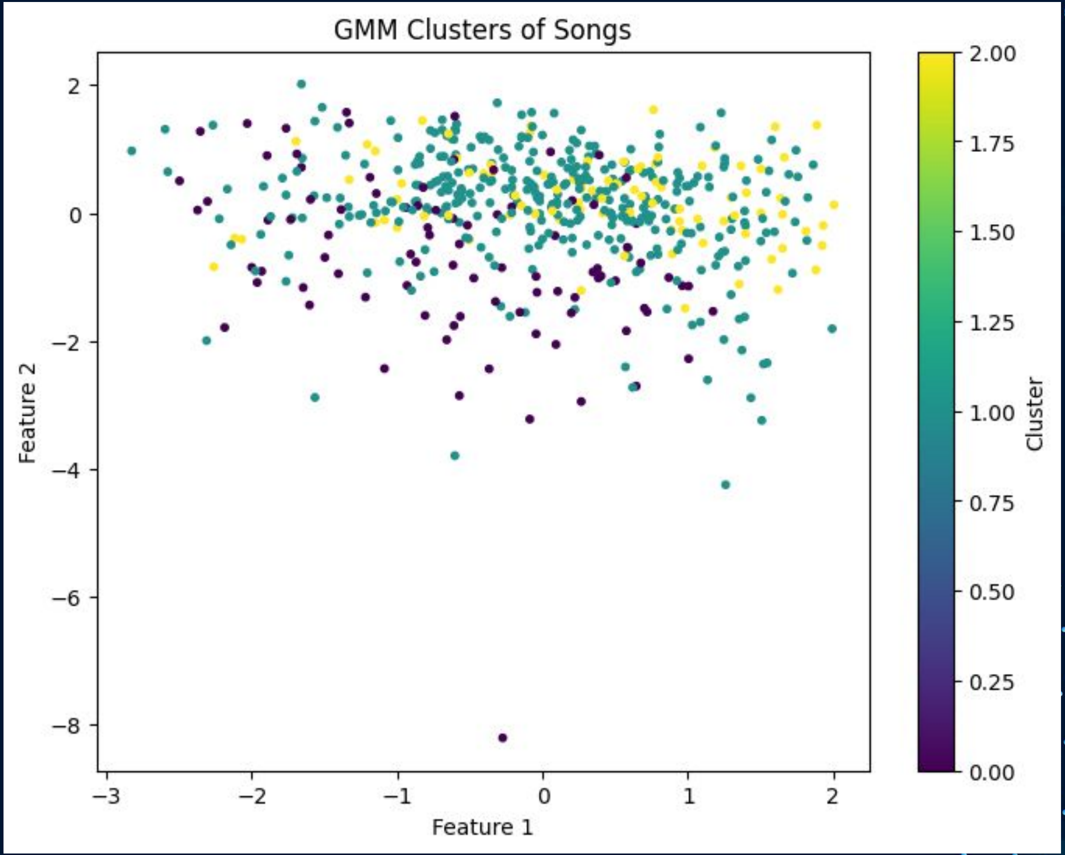
Clustering Model

- GMM
  - Multiple clusters in data (6)
  - Probabilistic
  - Variances can differ
- Log Likelihood
  - 20.01 (not great)

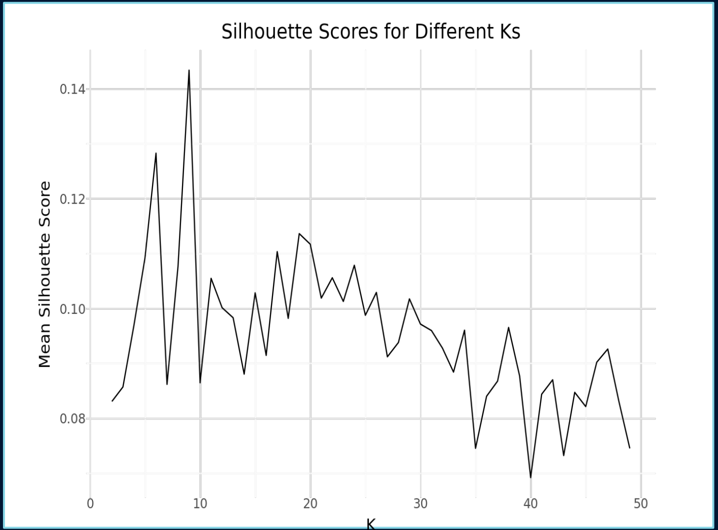


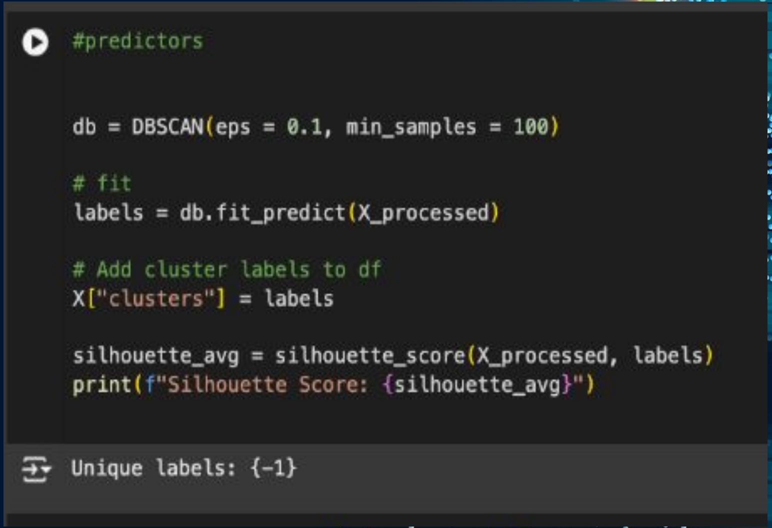
```python
gmm = GaussianMixture(n_components=5)

# 3. Fit model + predict
labels = gmm.fit_predict(X_processed)

# Add cluster labels to the original DataFrame
data["clusters"] = labels

for i in range(0, len(columns) - 1, 2):
    col_x = columns[i]
    col_y = columns[i + 1]

    # Skip if either column contains categorical data or invalid values for log scale
    if data[col_x].dtype == 'object' or data[col_y].dtype == 'object':
        print(f"Skipping plot for {col_x} vs {col_y} due to categorical data.")
        continue

    if (data[col_x] <= 0).any() or (data[col_y] <= 0).any():
        print(f"Skipping plot for {col_x} vs {col_y} due to non-positive values.")
        continue

    # Scatterplot
    plot = (ggplot(data, aes(x=col_x, y=col_y, color='factor(clusters)')) +
            geom_point() +
            labs(title=f"{col_x} vs {col_y}",
                 x=col_x,
                 y=col_y) +
            scale_x_log10() +  # Log scale for x-axis
            scale_y_log10())  # Log scale for y-axis

    # Print each plot
```





Avg. cluster means for popularity





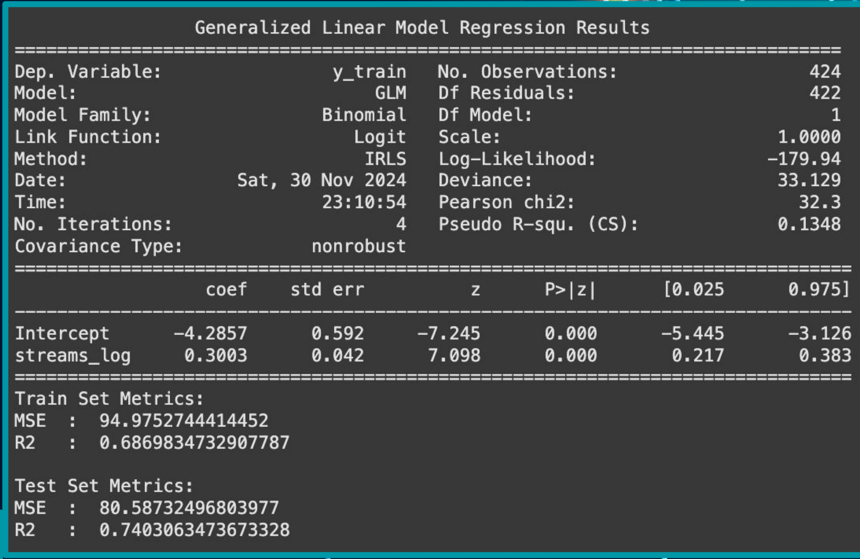
```
#predictors

db = DBSCAN(eps = 0.1, min_samples = 100)

# fit
labels = db.fit_predict(X_processed)

# Add cluster labels to df
X["clusters"] = labels

silhouette_avg = silhouette_score(X_processed, labels)
print(f"Silhouette Score: {silhouette_avg}")
```

```
Unique labels: {-1}
```

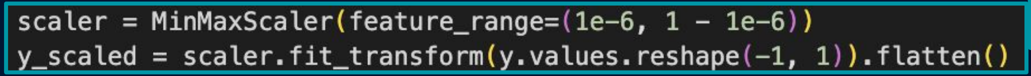
DBSCAN and KMeans Results

Beta Regression

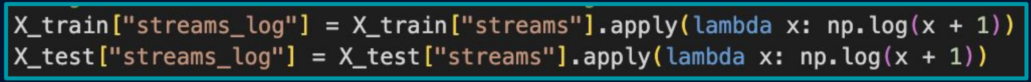
- Why beta regression? (What is beta regression)
- Necessary transformations made to "popularity" column in order to fit the requirements of a beta regression model
- Consistent performance between training and testing sets indicates the model is properly fit and can generalize well on new (unseen) data



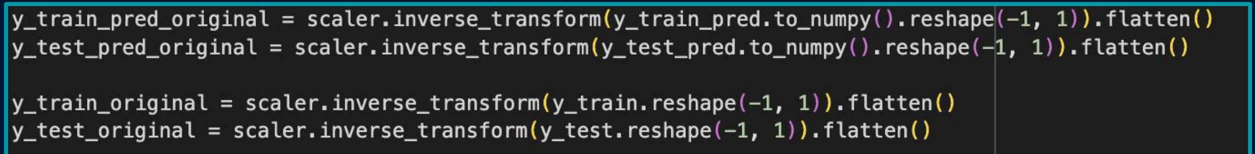
```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                y_train   No. Observations:                  424
Model:                            GLM   Df Residuals:                      422
Model Family:                Binomial   Df Model:                            1
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -179.94
Date:                Sat, 30 Nov 2024   Deviance:                       33.129
Time:                        23:10:54   Pearson chi2:                     32.3
No. Iterations:                     4   Pseudo R-squ. (CS):             0.1348
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.2857      0.592     -7.245      0.000      -5.445      -3.126
streams_log    0.3003      0.042      7.098      0.000       0.217       0.383
==============================================================================
Train Set Metrics:
MSE  :  94.9752744414452
R2   :  0.6869834732907787

Test Set Metrics:
MSE  :  80.58732496803977
R2   :  0.7403063473673328
```



```
scaler = MinMaxScaler(feature_range=(1e-6, 1 - 1e-6))
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).flatten()
```



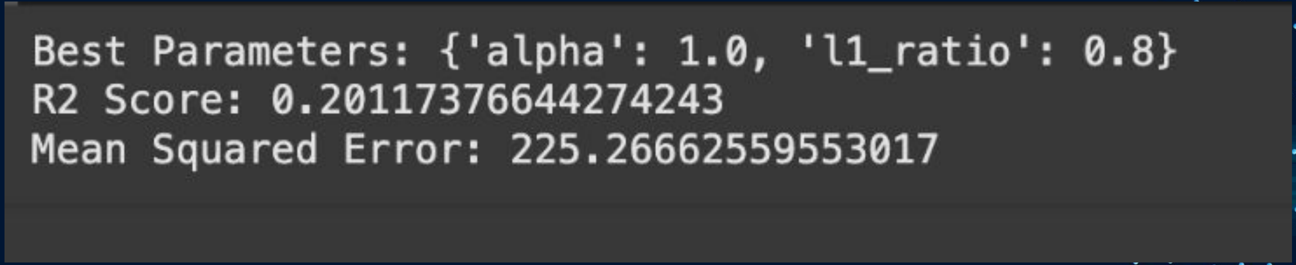
```
X_train["streams_log"] = X_train["streams"].apply(lambda x: np.log(x + 1))
X_test["streams_log"] = X_test["streams"].apply(lambda x: np.log(x + 1))
```



```
y_train_pred_original = scaler.inverse_transform(y_train_pred.to_numpy().reshape(-1, 1)).flatten()
y_test_pred_original = scaler.inverse_transform(y_test_pred.to_numpy().reshape(-1, 1)).flatten()

y_train_original = scaler.inverse_transform(y_train.reshape(-1, 1)).flatten()
y_test_original = scaler.inverse_transform(y_test.reshape(-1, 1)).flatten()
```

Regression Using ElasticNet

- Goal:
  - Wanted to figure out what makes a song popular and predict its popularity based on its features
- Why Linear Regression using ElasticNet?
  - A combination of Lasso and Ridge regression. It’s great because it not only selects the most important features but also balances them, ensures no overfitting
- Shows predictive power, but it also suggests that other factors, like marketing or cultural trends, likely play a role in popularity

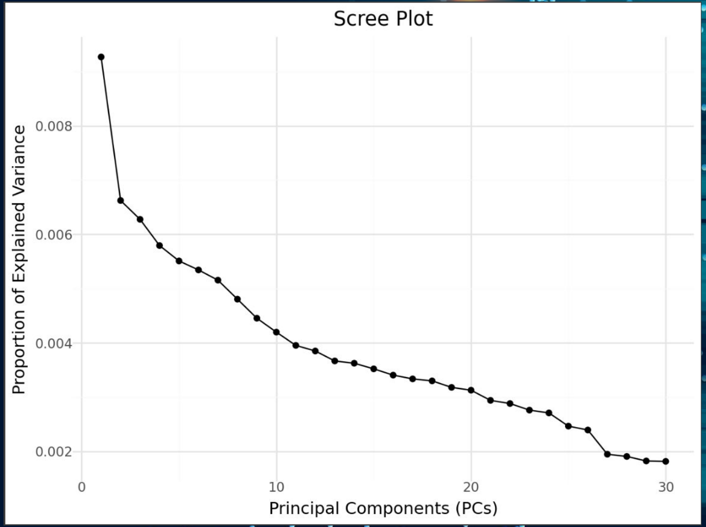


```
Best Parameters: {'alpha': 1.0, 'l1_ratio': 0.8}
R2 Score: 0.20117376644274243
Mean Squared Error: 225.26662559553017
```

PCA

Why PCA?

- PCA ensures the models focus on the most meaningful aspects of the data, reducing complexity
- Linear Regression predicts continuous probability, and Logistic regression predicts binary probability



Comparison of Models

Beta Regression, Clustering, ElasticNet, PCA

Beta Regression

- Test MSE of 80.58 and test R2 of 0.74 indicate passable performance with moderate error
- log features decreases skewness and can increase accuracy but also can negatively impact interpretability

Clustering

- Visually, GMM did not create good clusters
- Low Log Likelihood
  - Tried DBSCAN and KMeans but data did not do well
- Variables lowly correlated indicating clustering is not ideal

ElasticNet

R2 of .20 and MSE of 225

Leftover 80% of the variability unexplained

Limited by the quality and scope of the input features

Best for feature interpretability only

Limited predictive power

PCA

Popularity - R2 of .22 and MSE of 220

Hits - Accuracy of .64 and ROC AUC of .69

Classification:

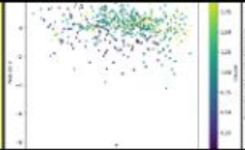
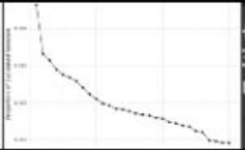
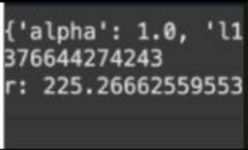
Non-hits - 70% precise, 69% recall

Hits - 55% precise, 56% recall

Used 30 PCs

Conclusion

Should This Model be Implemented?

Very Good	
Good	
Still Good	
Awful	  

```
{'alpha': 1.0, 'l1
376644274243
r: 225.26662559553
```

Outside of Beta Regression, no model performed exceedingly well. This is displayed in the tier list above. Streams is the most important variable when predicting popularity. Features are lowly correlated.