This is Edward Lu's(gl2576) Capstone Project Report.
I first imported all necessary packages and functions for the capstone questions and set np.random.seed to N-number.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
from sklearn.preprocessing import StandardScaler, scale #This is to fit it
from sklearn.decomposition import PCA #This will allow us to do the PCA efficiently
from sklearn.model_selection import train_test_split,cross_val_score, StratifiedKFold
from sklearn.linear_model import LinearRegression, ElasticNet, Lasso, LogisticRegression
from sklearn.metrics import r2_score, mean_squared_error, root_mean_squared_error, classification_report, roc_auc_score,
from scipy.special import expit # this is the logistic sigmoid function

np.random.seed(10058032)
```

✓ 0.0s Python

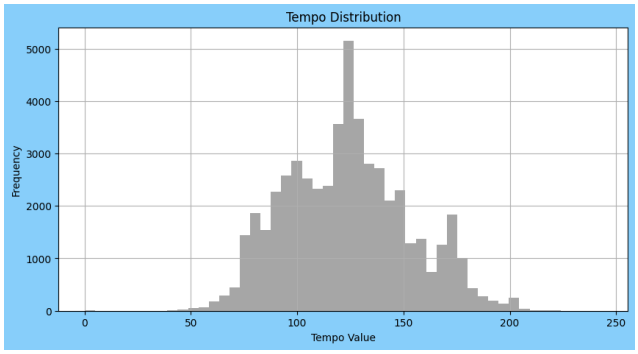
Next, I load the data using Pandas data frame.

```
# Load the data using Pandas dataframe.
data_df = pd.read_csv('spotify52kData.csv')
```

✓ 0.1s Python

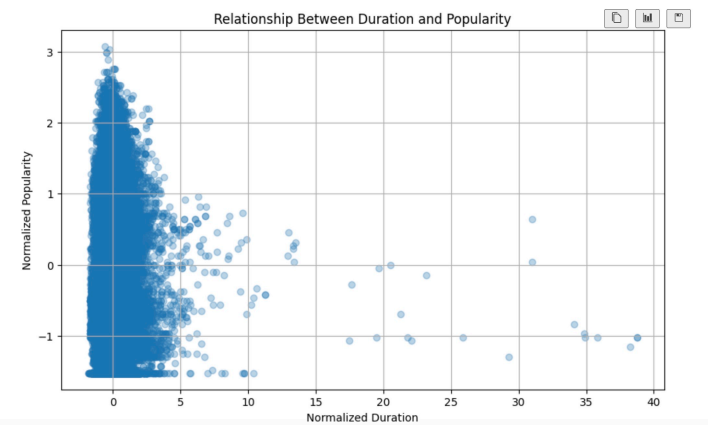
I defined the function ' root_mean_squared_error(y_true, y_pred)' to take RMSE.

Question 1
I first created a list of the 10 song features. Since the features all have numeric values, I plotted histograms for each attribute using a for loop to visualize their distributions. After seeing 10 histograms, I found *Tempo* distribution is closest to Normal Distribution with a symmetric bell curve.



Question 2
I first standardized Duration(x-axis) and popularity(y-axis) into z-score. Then I used .corr() function to calculate the correlation between the duration and popularity. Then I plotted a scatterplot with Duration(x-axis) and popularity(y-axis). Finally, I found there is approximately no correlation between duration and popularity. The absolute value of correlation is very small.

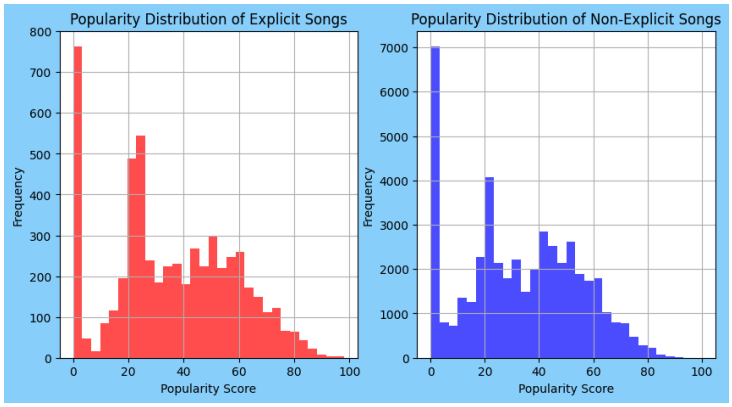
```
Correlation between Duration and Popularity:  -0.05465119593637639
```



Question 3

I created 2 groups of data based on their "explicit" labels(True/False). Then, I get the "popularity" of each song in the 2 groups. Next, I visualized the “popularity”(y-axis) of both groups to see if they are normally distributed. I found out "popularity" are not normally distributed in both groups. We cannot reduce data to sample means. So we need non-parametric test. The Spotify popularity index is not categorical data but interval data and there are 2 groups. We want to compare the medians to see if explicitly rated songs more popular than songs that are not explicit. In this case, I choose Mann Whitney U Test. I used .median() to get the medians for both groups. Then I conducted Mann Whitney U Test: H_0: There is *no* difference in 'popularity' between Explicit songs and Non-Explicit songs. H_1: Explicitly rated songs are *more* popular than songs that are not explicit. I used stats.mannwhitneyu(x,y,alternative=‘greater’) since it’s a right-tailed test.
Finally, I found p-value is extremely small, meaning we can reject the null hypothesis. So the conclusion is Explicitly rated songs are more popular than songs that are not explicit. It’s also reinforced by the median we got.

```
Median of Popularity for TRUE:  34.0
Median of Popularity for FALSE:  33.0
U Statistics:  139361273.5
P-value:  1.5339599669557339e-19
Conclusion: Explicitly rated songs are more popular than songs that are not explicit.
```

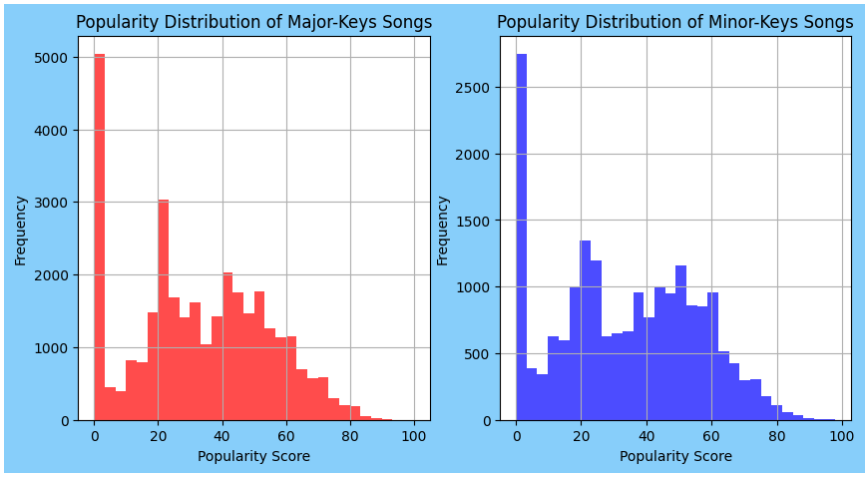


Question 4

I created 2 groups of data based on their "mode" labels(1/0). Then, I get the "popularity" of each song in the 2 groups. Next, I visualized the “popularity”(y-axis) of both groups to see if they are normally distributed. I found out "popularity" are not normally distributed in both groups. We

cannot reduce data to sample means. So we need non-parametric test. The Spotify popularity index is not categorical data but interval data and there are 2 groups. We want to compare the medians to see if songs in major key more popular than songs in minor key. In this case, I choose Mann Whitney U Test. I used .median() to get the medians for both groups. Then I conducted Mann Whitney U Test: H_0: There is *no* difference in 'popularity' between major-key songs and minor-key songs. H_1: Major-key songs are *more* popular than minor-key songs. I used stats.mannwhitneyu(x,y,alternative='greater') since it's a right-tailed test.
Finally, I found p-value is extremely large, meaning we cannot reject the null hypothesis. So the conclusion is there is no evidence to suggest that major-key songs are more popular than minor-key songs. Minor-key songs are more popular than major-key songs, which is reinforced by the median we got.

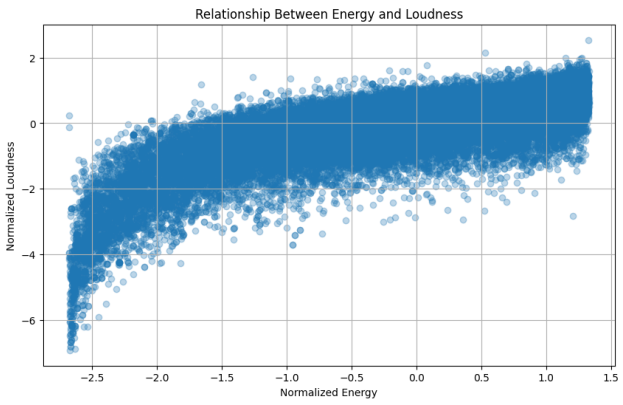
```
Median of Popularity for Major Group:  32.0
Median of Popularity for Minor Group:  34.0
U Statistics:  309702373.0
P-value:  0.9999989912386331
Conclusion: Minor-key songs are more popular than major-key songs.
There is no evidence to suggest that major-key songs are more popular than minor-key songs,
```

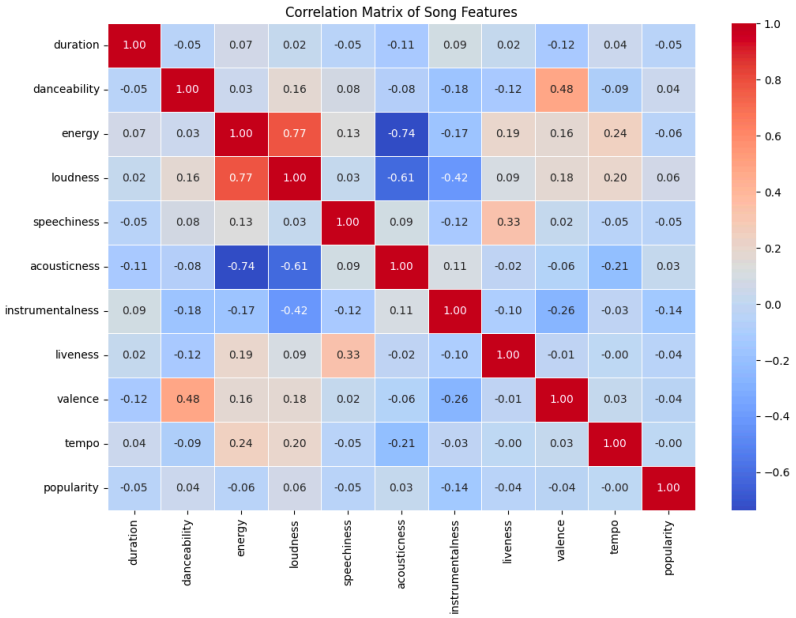


Question 5

I standardized Energy(x-axis) and Loudness(y-axis) into z-score. Then, I created a dictionary of standardized_energy and standardized_loudness. Next, I created a data frame from the dictionary named "standardized_df". Next, I used .corr() function to calculate correlation between the 2 columns('Standardized Energy' and 'Standardized Loudness'). Moreover, I created a scatterplot of standardized Energy(x-axis) and Loudness(y-axis). The scatterplot shows there is a positive correlation between energy and loudness.
Finally, I found 'energy' does reflect 'loudness' of a song. The 2 variables are positively correlated. The higher 'energy' is, the higher 'loudness' becomes.

```
Correlation between Normalized Energy and Normalized Loudness: 0.7748808291850194
```





Question 6

I first did exploratory data analysis on the 10 song features. I want to see if the features are correlated with each other. Thus, I standardized the 10 features and 'popularity' into z-score and create a data frame for the 11 columns. Next, I calculated the correlation matrix. Then, I drew a heatmap with the numeric values in each cell. Next, I calculated correlations between each feature and 'popularity'.

Based on the heatmap, I found out 'loudness' and energy are positively correlated. 'acousticness'' and 'energy' are negatively correlated. 'acousticness'' and 'loudness' are also negatively correlated. Based on the list of correlations, I found none of features have strong correlations with 'popularity'.

```
duration correlation:  -0.05465119593637639
danceability correlation:  0.03715781135740385
energy correlation:  -0.05592469066526966
loudness correlation:  0.06021003481482001
speechiness correlation:  -0.048532677084213646
acousticness correlation:  0.026233391738937687
instrumentalness correlation:  -0.1449722705373328
liveness correlation:  -0.043846029909762836
valence correlation:  -0.03576878910256893
tempo correlation:  -0.0026318311756676578
```

After EDA, I built simple linear regression for each feature and 'popularity' using for loop. I did train-test-split with test-size=0.2. Then, I trained the model with train dataset and generate y-prediction from x-test. Next, I calculated R-squared with r2_score(y_test, y_pred) and RMSE with root_mean_squared_error(y_test, y_pred). The 2 metrics are to evaluate how good the model is. Finally, I got a list of R-squared and RMSE of 10 models. Among them, I found **best single feature is instrumentalness with the highest R-squared, with R-squared=0.025, RMSE=0.98.**

Below is the list of the metrics for each model.

```
duration
R-suared: 0.003513339375693425
Root Mean Squared Error: 0.9932084217239865
----------
danceability
R-suared: 0.0015089972821005215
Root Mean Squared Error: 0.9942067940498664
----------
energy
R-suared: 0.0029483619392279836
Root Mean Squared Error: 0.9934899412140813
----------
loudness
R-suared: 0.00410290413175507
Root Mean Squared Error: 0.9929145656492276
----------
speechiness
R-suared: 0.0036787895418751715
Root Mean Squared Error: 0.9931259653668604
----------
acousticness
R-suared: 0.0004889246006493098
Root Mean Squared Error: 0.9947145123462185
----------
instrumentalness
R-suared: 0.024834252203008056
Root Mean Squared Error: 0.9825255843538242
----------
liveness
R-suared: 0.003542342968189227
Root Mean Squared Error: 0.9931939675305292
----------
valence
R-suared: 0.0007414359745734345
Root Mean Squared Error: 0.9945888546119708
----------
tempo
R-suared: 8.779679496973003e-06
Root Mean Squared Error: 0.9949534040345769
----------

Best single feature: instrumentalness with R2: 0.024834252203008056 and RMSE: 0.9825255843538242
```

Question 7

I first tried a multiple linear regression model for all features and 'popularity'. I did train-test-split with test-size=0.2. Then, I trained the model with train dataset and generate y-prediction from x-test. Next, I calculated overall R-squared with r2_score(y_test, y_pred) and overall RMSE with root_mean_squared_error(y_test, y_pred). Next, I used model.coef_ to see importance of each feature.

Finally, **I found R-squared of the model is 0.053536229699158056 and RMSE is 0.9679582948331364.** The most important feature is 'Energy'.

The multiple linear regression model only increased R-squared by approximately 0.03 and decreased RMSE by around 0.02. Thus, the multiple linear regression model did not significantly improve predictions on 'popularity'.

```
Overall model R2: 0.053536229699158056 and RMSE: 0.9679582948331364
Feature importances: {'duration': -0.043598873084253 86, 'danceability': 0.04151127670891615,
Most important feature is Energy:  0.1610866259201247
```

I also tried Lasso regression to reduced the effects of insignificant predictors. I did train-test-split and standardized x-train and x-test with StandardScaler(). Then I creased lasso regression with Lasso(alpha=0.1). After fitting the model and generated predictions, I evaluated the model with R-squared and RMSE. I found R-squared decreased a little bit and RMSE increased significantly. Also lasso coefficients increased for each variable.

```
Lasso R²: 0.05
Lasso RMSE: 21.05
Lasso Coefficients: [-0.85166859  0.72411268 -3.26796164  2.9293376  -0.82745434  0.12824851
 -2.7109007  -0.31015866 -1.84418572  0.14526953]
```

I also tried Elastic Net to see if there will be improvements. I did train-test-split and standardized x-train and x-test with StandardScaler(). Then I creased elastic net with parameters (alpha=1.0, l1_ratio=0.5, max_iter=10000, random_state=10058032). After fitting the model and generated predictions, I evaluated the model with R-squared, RMSE, and Mean Squared Error.
Finally, I found R-squared dropped by 0.02 and RMSE and MSE increased significantly.

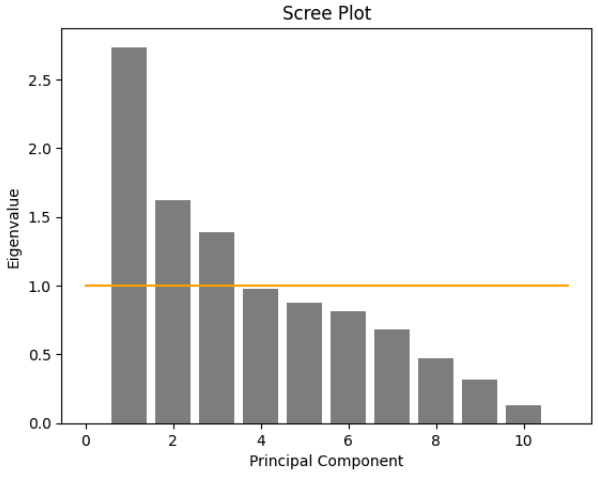
```
Elastic Net Coefficients: [-0.35636943  0.10877283 -0.82466873  0.54244536 -0.36520009  0.
 -1.76429977 -0.18687543 -0.55075985  0.        ]
R²: 0.03230213517665814
Elastic Net Mean Squared Error: 452.9141021300478
Elastic Net Root Mean Squared Error: 21.281778641129783
```

After trying 3 different models, I choose multiple linear model after evaluations.

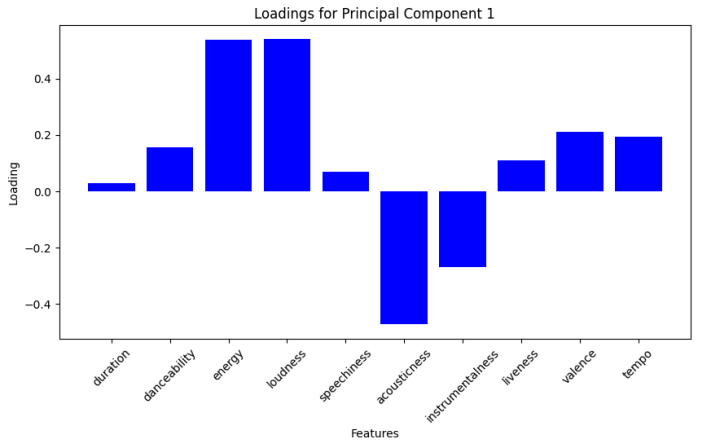
Question 8

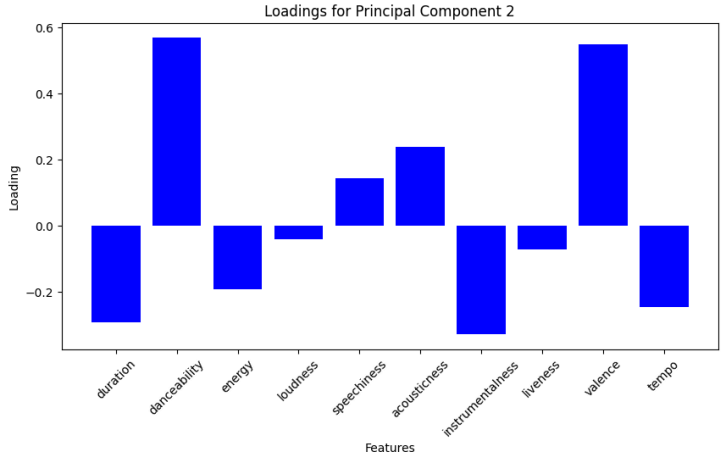
I did principle components analysis on the 10 features. I used PCA() and fitted with standardized 10 features. I got eigenvalues with .explained_variance_ , loading by .components_, rotated data by .fit_transform(standardized_df[attributes_columns]), and variance explained by eigVals/ sum(eigVals)*100. Then, I printed out variance explained by each factor.
Next, I plotted the bar chart for eigenvalues and used 3 criterion to judge principal components.
Finally, **I found 3 factor extracted by Kaiser criterion, 1 factor by elbow criterion, and 7 factors by 90% variance explained.**
The conclusion is: **The first 3 components are considered meaningful based on the Kaiser Criterion,explaining approximately 57.359% of the variance. Variance explained by each components in decreasing order: 27.339, 16.174, 13.846, 9.796, 8.752, 8.148, 6.783, 4.716, 3.131, 1.316**
To cover at least 90% of the variance, 7 components are required, as derived from the cumulative variance. V

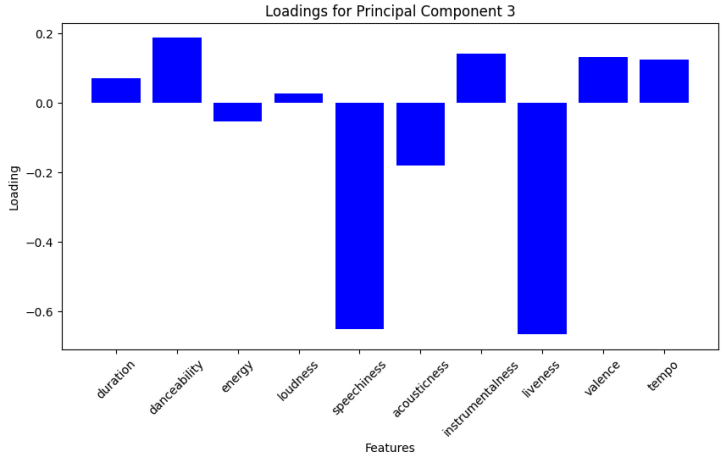
```
Number of factors extracted by Kaiser criterion: 3
Number of factors extracted by elbow criterion: 1
Number of factors to account for at least 90% variance: 7
Conclusion:
The first three components are considered meaningful based on the Kaiser Criterion,explaining approximately 57.359% of the variance.
To cover at least 90% of the variance, 7 components are required, as derived from the cumulative variance.
```



Next, I plotted loadings for the 3 principal components.







Question 9

I built a logistic regression to predict a song is in major or minor key from valence since the outcome is binary(major/minor). I did train-test-split with test-size=0.2. Then, I trained the model

with train dataset and generate Area-Under-Curve from x-test. Next, I generated predictions from x-test.
Finally, I printed the model's coefficient and intercept, classification report, and AUC-ROC for valence-mode Model.
I found Area-Under-Curve is less than 0.5, meaning the valence model cannot predict major or minor key. Also, precision is very low.

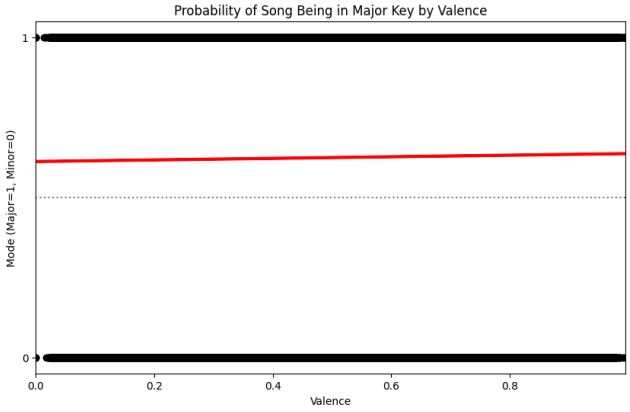
```
[[0.13078091]]
[0.44570527]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3949
           1       0.62      1.00      0.77      6451

    accuracy                           0.62     10400
   macro avg       0.31      0.50      0.38     10400
weighted avg       0.38      0.62      0.47     10400

AUC-ROC for valence-mode Model: 0.4995568203947721
```

I also plotted the sigmoid plot. Based on the plot, we can see 'valence' is not a good predictor as the same 'valence' value can predict both major and minor key.



Next, I tried building logistic regression with the other 9 song features. Using the same steps and metrics, I found the best predictor among the 10 features is 'speechiness' with the highest AUC-ROC(0.5546255762365291).

```
Feature: speechiness
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3949
           1       0.62      1.00      0.77      6451

    accuracy                           0.62     10400
   macro avg       0.31      0.50      0.38     10400
weighted avg       0.38      0.62      0.47     10400

AUC-ROC for speechiness Model: 0.5546255762365291
```

Question 10

I first created a new column in original data frame as data_df[‘genre_classical’]. I set the rows of data_df[‘genre_classical’] which ‘track_genre' is ‘classical’ to 1 and other genres to ‘0’. In this way, a logistic regression will be a good fit since the outcome is binary(1 or 0).

First, I used the same steps as previous models to build the logistic model for ‘duration’ as the predictor. Next, I generated Classification Report for Duration Model, AUC-ROC for Duration Model, and Confusion Matrix for Duration Model.

Based on the output, AUC-ROC is less than 0.5 which shows ‘duration’ is not a good predictor. The Confusion Matrix shows there are 10199 true negatives, 201 false negative and 0 for false positive and true positive. Also, based on precision and recall, ‘duration’ model did a good job in predicting non-classical songs but not classical songs.

```
Classification Report for Duration Model:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     10199
           1       0.00      0.00      0.00       201

    accuracy                           0.98     10400
   macro avg       0.49      0.50      0.50     10400
weighted avg       0.96      0.98      0.97     10400

AUC-ROC for Duration Model: 0.4274138670311547
Confusion Matrix for Duration Model: [[10199     0]
 [  201     0]]
```

Second, I built the logistic regression model with principal components. I selected 3 components and used transformed data as predictors. With the same steps, I evaluated the model with the same metrics as the ‘duration’ model.

Eventually, I found out AUC-ROC for the principal components model is close to 1, showing it’s a great model. Also, the Confusion Matrix shows there are 10170 true negatives, 187 false negative and 29 for false positive, and 14 for true positive. Precision and recall for negative outcome are also high. Even if Precision and recall for positive outcome are still low, they are higher than ‘duration’ model.

```
Classification Report for Principal Component Model:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     10199
           1       0.33      0.07      0.11       201

    accuracy                           0.98     10400
   macro avg       0.65      0.53      0.55     10400
weighted avg       0.97      0.98      0.97     10400

AUC-ROC for Principal Component Model: 0.9443819240887434
Confusion Matrix for Duration Model: [[10170    29]
 [  187    14]]
```

To avoid overfitting, I also used cross-validation methods. I used StratifiedKFold(n_splits=5, shuffle=True, random_state=10058032) and cross_val_score().
The output also shows principle components is a better predictor with higher AUC-ROC.

```
Cross-Validation AUC-ROC Scores for Duration Model: [0.41692819 0.45319142 0.37724387 0.41152206 0.43747794]
Average AUC-ROC for Duration Model: 0.4192726960784313

Cross-Validation AUC-ROC Scores for Principal Components Model: [0.94538971 0.94685956 0.93497696 0.94937672 0.94968235]
Average AUC-ROC for Principal Components Model: 0.9452570588235293
```

Therefore, the conclusion is: **principal components is a better predictor of if a song is classical music.**

Extra Credit:
I did exploratory data analysis on number of beats per measure, time_signature column.
First, I calculated the mode (most common value) of the time signatures for each genre
I found the modes of time signatures are all 4 for every genre.

```
Most common time signature for each genre:
 track_genre
acoustic         4
afrobeat         4
alt-rock         4
alternative      4
ambient          4
anime            4
black-metal      4
bluegrass        4
blues            4
brazil           4
breakbeat        4
british          4
cantopop         4
chicago-house    4
children         4
chill            4
classical        4
club             4
comedy           4
country          4
dance            4
dancehall        4
death-metal      4
deep-house       4
detroit-techno   4
disco            4
disney           4
drum-and-bass    4
dub              4
dubstep          4
edm              4
electro          4
electronic       4
emo              4
folk             4
forro            4
french           4
funk             4
garage           4
german           4
gospel           4
goth             4
grindcore        4
groove           4
grunge           4
guitar           4
happy            4
hard-rock        4
hardcore         4
hardstyle        4
heavy-metal      4
hip-hop          4
Name: time_signature, dtype: int64
```

Next, I found genres with uncommon time signatures
I found the uncommon time signatures among all genres are usually 1 and 5.

```
Genres with uncommon time signatures:
 track_genre
acoustic            [1, 5]
afrobeat            [5, 1]
alt-rock         [5, 1, 0]
alternative         [5, 1]
ambient          [5, 1, 0]
anime               [5, 1]
black-metal         [5, 1]
bluegrass           [5, 1]
blues               [5, 1]
brazil              [5, 1]
breakbeat           [5, 1]
british             [1, 5]
cantopop            [1, 5]
children            [1, 5]
chill               [5, 1]
classical           [5, 1]
club                [5, 1]
comedy              [5, 1]
country             [5, 1]
dance                  [5]
dancehall           [5, 1]
death-metal         [1, 5]
deep-house             [1]
detroit-techno      [5, 1]
disco               [5, 1]
disney           [5, 1, 0]
drum-and-bass       [5, 1]
dub                 [5, 1]
dubstep             [5, 1]
edm                    [1]
electro             [1, 5]
electronic          [1, 5]
emo                 [5, 1]
folk                [1, 5]
forro               [1, 5]
french              [1, 5]
funk                [1, 5]
garage              [1, 5]
german              [5, 1]
gospel              [5, 1]
goth                [1, 5]
grindcore           [1, 5]
groove              [5, 1]
grunge              [1, 5]
guitar           [5, 1, 0]
happy               [5, 1]
hard-rock           [5, 1]
hardcore            [5, 1]
hardstyle           [5, 1]
heavy-metal         [5, 1]
hip-hop             [5, 1]
Name: time_signature, dtype: object
```

Then, I calculated standard deviation of time signature for each genre.
Among them, I found ‘ambient’, ‘comedy’, and ‘classical’ genres have high standard deviation in time signatures.

```
Variability of time signature for each genre:
 track_genre
acoustic          0.426564
afrobeat          0.327401
alt-rock          0.281706
alternative       0.301209
ambient           0.767581
anime             0.356649
black-metal       0.539736
bluegrass         0.351002
blues             0.376295
brazil            0.315040
breakbeat         0.170023
british           0.471706
cantopop          0.293261
chicago-house     0.108940
children          0.421043
chill             0.336120
classical         0.740829
club              0.360390
comedy            0.866343
country           0.344019
dance             0.137732
dancehall         0.370693
death-metal       0.451640
deep-house        0.166649
detroit-techno    0.229843
disco             0.243331
disney            0.571569
drum-and-bass     0.192323
dub               0.229372
dubstep           0.272237
edm               0.153947
electro           0.237934
electronic        0.250533
emo               0.289072
folk              0.617074
forro             0.161015
french            0.339015
funk              0.425512
garage            0.285267
german            0.462400
gospel            0.319018
goth              0.285658
grindcore         0.666063
groove            0.292583
grunge            0.354825
guitar            0.606788
happy             0.235541
hard-rock         0.263417
hardcore          0.300816
hardstyle         0.242410
heavy-metal       0.276542
hip-hop           0.372417
Name: time_signature, dtype: float64
```

I also plotted histogram for time signature distribution.
I found the most common time signatures are 3 and 4.