Amanda Strack
Capstone Project I: Playlist Creation Using Spotify Audio Features
https://github.com/amandastrack/Capstone-Project-I-

Problem Statement:
The right song can motivate you to run that extra mile or grind through that last chapter of studying. That's why people often have a playlist for different activities of their day. However, it can be time consuming to create these playlists. It requires sifting through many songs and then determining if it fits a certain category based on how it sounds and makes you feel.

In this project we aim to use machine learning to classify songs into different categories. We will do this by exploring different audio features of a song and classifying which playlist category the song best fits into based on these features.

My client is any music streaming company such as Spotify. The client will be able to use this model to create a data-driven product feature that enables users to more easily discover songs through pre-made playlists.

Dataset Description:
The data is extracted from the Spotify Web API (https://api.spotify.com). From the API we extract songs from different Workout, Focus, Party and Chill Playlists that are available on Spotify. We then extract another larger dataset of song audio feature details described below.

KEY	VALUE DESCRIPTION
duration_ms	The duration of the track in milliseconds.
key	The estimated overall key of the track.
mode	Mode indicates the modality of a track. Major is represented by 1 and minor is 0.
time_signature	The time signature is a notational convention to specify how many beats are in each bar.
acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.
energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
instrumentalness	Predicts whether a track contains no vocals.
liveness	Detects the presence of an audience in the recording.
loudness	The overall loudness of a track in decibels (dB).
speechiness	Speechiness detects the presence of spoken words in a track.
valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
tempo	The overall estimated tempo of a track in beats per minute (BPM).

Data Cleaning and Wrangling:

Extracting and Merging:

First, I extracted data using the Spotify API. I pulled songs from playlists in each category (workout, focus, party, chill). Then I pulled the audio feature data for each of these songs. The two datasets were merged by the song ID and the unnecessary columns were removed. Each row representing a song was labelled with its corresponding playlist category. The final data frame consists of 2148 rows with ~500 songs per genre.

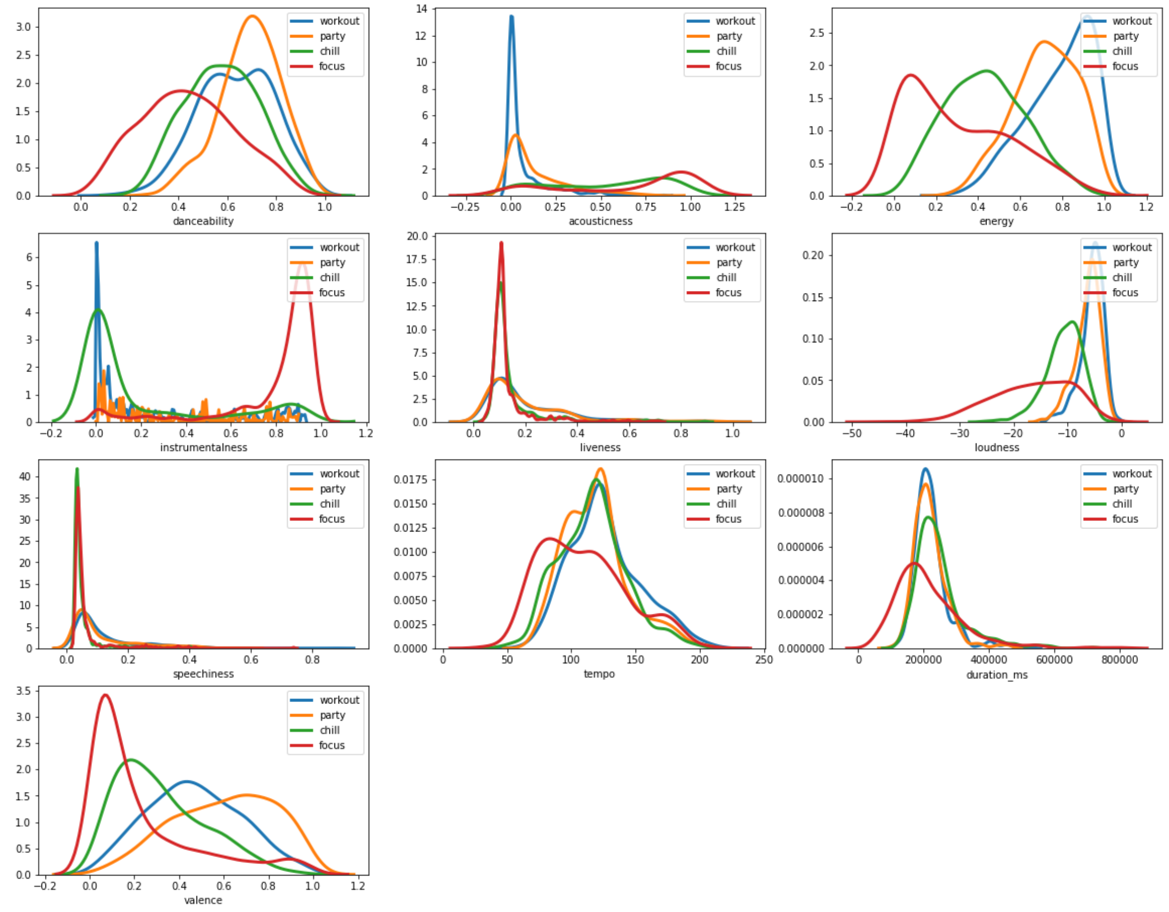
Missing Values, Duplicates and Outliers:

No missing values were found in the data. Any duplicate songs within a class was removed to avoid skewing our analysis. Outliers were inspected by plotting box plots of each of the numerical features. We found that the variables 'duration_ms' and 'speechiness' have outliers. After investigation, we know that these outliers are not any type of 'data dictionary' or 'placeholder' values. We decided to keep these outliers in the data and wait until the modeling stage to decide how to deal with them.

Exploratory Data Analysis:

Are the variables significant in terms of predicting Playlist Genre?

Through exploratory data analysis, the goal is to achieve an understanding of which variables are good candidates for our model. We want to ensure that our variables are significantly distinguishable across the four classes.



In these KDE plots you can see distinguishable distributions of most variables. For example, focus playlist songs clearly always have high instrumentalness and workout playlist songs always have low acousticness. However, for some variables such as liveness and speechiness, it is really hard to distinguish between the class distribution lines. It is difficult to draw definitive conclusions solely from graphs. Now, we further explore our data through formal hypothesis testing.

Formally Testing Differences Between Workout and Party Playlists
In the data wrangling stage we found that Workout and Party classes had many songs in common. This raises a question: Is there a statistically significant difference between Workout and Party playlists? To answer this, we use a two sample t-test. The t-test tests whether the means of two samples are significantly different.

Two sample T test at 5% significance level:
H_0: The means of workout and party playlist samples are the same.

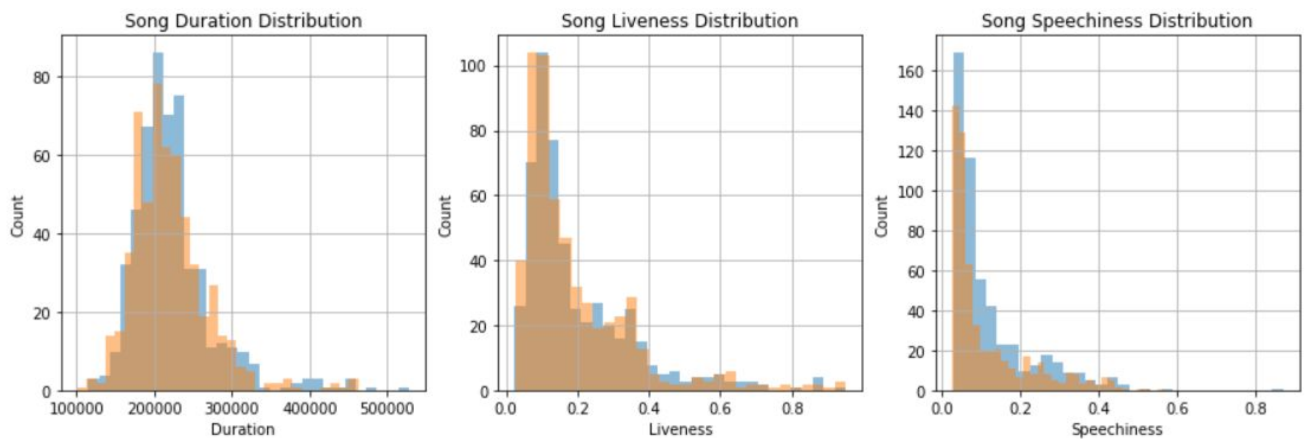
H_1: The means of workout and party playlist samples are not the same.

The table below shows variables that did not result in a significant difference. All other variables resulted in a very low p-values.

Results:

	T-Statistic	**P Value**	**Conclusion**
Liveness	0.322	≈0.748	Do not reject the null, means are the same
Speechiness	1.177	≈0.239	Do not reject the null, means are the same
Duration	.903	≈0.367	Do not reject the null, means are the same

Since these variables are so similar between Workout and Party playlists, they may be weak predictors in our model. You can see how close these distributions are in the histograms below.



The t-test is only valid for continuous variables. Therefore we use the Chi-Squared test for testing our three categorical variables (mode, key, time signature). Chi-squared tests whether there is a relationship between two categorical variables.

Chi-Squared test at 5% significance level:

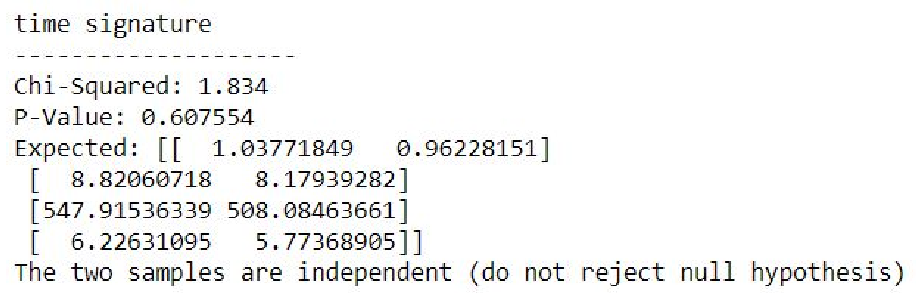
H_0: there is no relationship between the variable and playlist genre

H_1: there is a relationship between the variable and playlist genre

Results:

	Chi-Squared	**P Value**	**Conclusion**
Mode	2.408	≈ 0.1207	Do not reject the null, there is no relationship
Key	12.348	≈ 0.3381	Do not reject the null, there is no relationship
Time Signature	1.834	≈ 0.6076	Do not reject the null, there is no relationship

The test results tell us that we accept the null hypothesis that these variables do not have a significant relationship to the playlist genre. However, it is important to check that the expected cell frequencies are greater than or equal to 5, as this is one of the assumptions for the chi-squared test. After interpreting the results, we find that the expected frequencies for time signature does not fit this assumption.



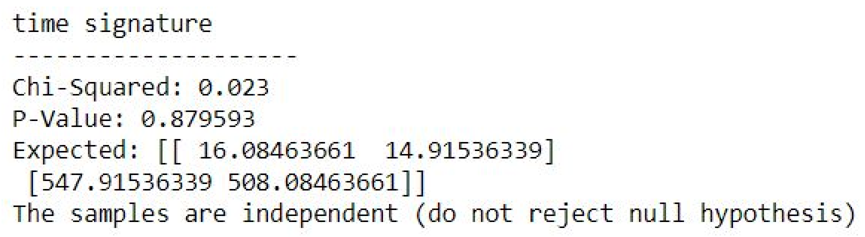
```
time signature
--------------------
Chi-Squared: 1.834
P-Value: 0.607554
Expected: [[  1.03771849   0.96228151]
 [  8.82060718   8.17939282]
 [547.91536339 508.08463661]
 [  6.22631095   5.77368905]]
The two samples are independent (do not reject null hypothesis)
```

The first two expected frequencies (1.0377 and 0.9623) are not greater than 5. The reason for the small frequency values is because of the small sample size for time signature category of 1. To fix this, we test only one category of time signature against all the others combined. Since time signature '4' is the most common value, we test it against the others to see if there is a dependence on playlist genre. A pandas method is used to create a dummy variable for the time signature ‘4’ which equals to ‘1’ if it belongs to that category and ‘0’ if it does not.

Adjusted Contingency Table:

“4” Time Signature	**party**	**workout**
0	16	15
1	548	508

We then perform a Chi-squared test with the new contingency table. The results are as follows:



```
time signature
--------------------
Chi-Squared: 0.023
P-Value: 0.879593
Expected: [[ 16.08463661  14.91536339]
 [547.91536339 508.08463661]]
The samples are independent (do not reject null hypothesis)
```

Now all of the expected frequencies are greater than 5 and the chi-squared results can be trusted. We accept the null hypothesis that there is no relationship between having a 4-beat time signature and belonging to a workout or party playlist.

In conclusion, liveness, speechiness, duration continuous variables did not test to have a significant difference among workout and party playlists. Mode, key and time signature categorical variables also did not test to have a significant relationship to distinguishing workout and party playlists. However, the amount of impact these variables will have on the final model is still indeterminate. We will keep these findings in mind when testing different variations of the final model.

Formally Testing Differences Between All Playlists
Now we will include all four genres in our hypothesis testing. We want to test that the variables are significantly different across all of our classes. For testing the continuous variables, we use the Kruskal-Wallis test. The Kruskal-Wallis test assesses for significant differences on a continuous dependent variable by a categorical independent variable (with two or more groups).

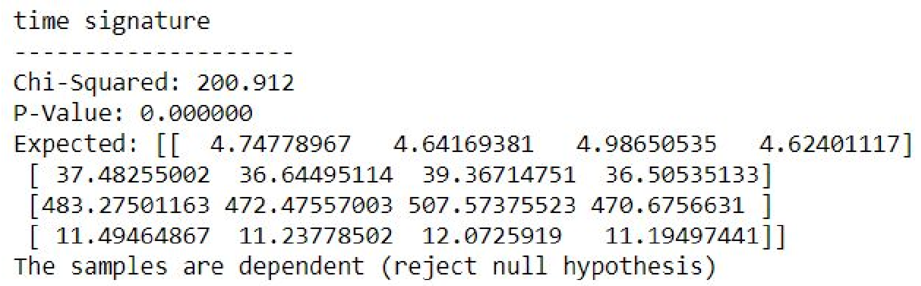
Kruskal-Wallis test at 5% significance level:

H0: the distributions of all categories are equal.

H1: the distributions of one or more categories are not equal.

For all ten numerical variables, the Kruskal-Wallis test results in a low p-value (0.000000). Therefore, we can conclude that for each of the variables, the distributions of at least one playlist is significantly different from the others. However, the test does not identify where the differences occur.

Again, we use the Chi-Squared test for testing the three categorical variables (mode, key, time signature). The results of these tests showed a low p value for all three variables. As expected from our earlier chi-squared test, the results of time signature have expected frequency values less than 5 and we cannot trust the results of the chi-squared test in this case.



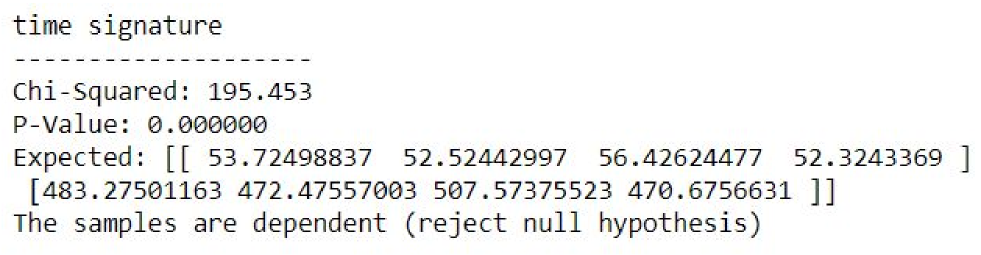
```
time signature
--------------------
Chi-Squared: 200.912
P-Value: 0.000000
Expected: [[  4.74778967   4.64169381   4.98650535   4.62401117]
 [ 37.48255002  36.64495114  39.36714751  36.50535133]
 [483.27501163 472.47557003 507.57375523 470.6756631 ]
 [ 11.49464867  11.23778502  12.0725919   11.19497441]]
The samples are dependent (reject null hypothesis)
```

As before, we test time signature '4' against all other values to see if there is an influence on playlist genre. A pandas method is used to create a dummy variable for only time signature ‘4’ which equals to ‘1’ if it belongs to that variable and ‘0’ if it does not.

Adjusted Observed Values:

"4" Time Signature	**chill**	**focus**	**party**	**workout**
0	52	132	16	15
1	485	393	548	508

With the new contingency table we get the following results:



```
time signature
--------------------
Chi-Squared: 195.453
P-Value: 0.000000
Expected: [[ 53.72498837  52.52442997  56.42624477  52.3243369 ]
 [483.27501163 472.47557003 507.57375523 470.6756631 ]]
The samples are dependent (reject null hypothesis)
```

Now all of the expected frequencies are greater than 5 and the chi-squared results can be trusted. We can reject the null hypothesis and conclude that are is relationship between having a 4-beat time signature and playlist genre. Therefore, there is a significant relationship between these categorical variables and playlist genre. However, since Chi-squared is an omnibus test, we don't know what this relationship is but we do know that these variables are not independent of each other.

Inferential Statistics Conclusion

We have now tested all of the features in our dataset. The goal of hypothesis testing was to determine which variables would be good predictors in our model. When testing the variables across all the four classes, we found that *all* variables have proved to be meaningful for our model. However, when only comparing Workout and Party playlists, we identified the following features to be weak in distinguishing the two classes.

1. Liveness
2. Speechiness
3. Duration
4. Mode
5. Key
6. Time Signature

We now have explored our data through graphs and inferential statistics. Now that we have an understanding of how each feature may play a role in predicting playlist genre, we begin to build our models.

In-Depth Analysis: Machine Learning

Supervised machine learning techniques:

1. Random Forest
    - Predict the label of a data point by ensembling decision trees
2. Support Vector Machines (SVM)
    - Defines a hyperplane that separates classes

3. Naive Bayes
    - Makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting
4. K-Nearest Neighbors
    - Predict the label of a data point by looking at the ‘K’ closest labeled data points and taking the majority vote
5. XGBoost
    - Implements gradient boosted decision trees

We began by evaluating these models with their default hyperparameters and all features. Models are evaluated using accuracy and f1 scores. We look at the f1 score in addition to the accuracy score since the accuracy score could be influenced by imbalanced classes. These performance scores are based on the test set.
Default Model Outcomes:

Model	**Accuracy Score**	**F1 Score**
Random Forest	69.81%	.70
SVM	71.39%	.72
Naive Bayes	67.38%	.68
KNN	66.95%	.68
XGBoost	**72.53%**	**.73**

XGBoost has the highest accuracy score and tied highest f1 score of the default models. We will focus on this model and move forward performing hyperparameter tuning.

Hyperparameter Tuning:

To tune the parameters we used a random search method instead of Grid Search. Grid search has proven to be too exhaustive for extensive parameter grids. Random Search is a more efficient method in this case of an XGBoost model. Cross-validation was used to evaluate the parameters.

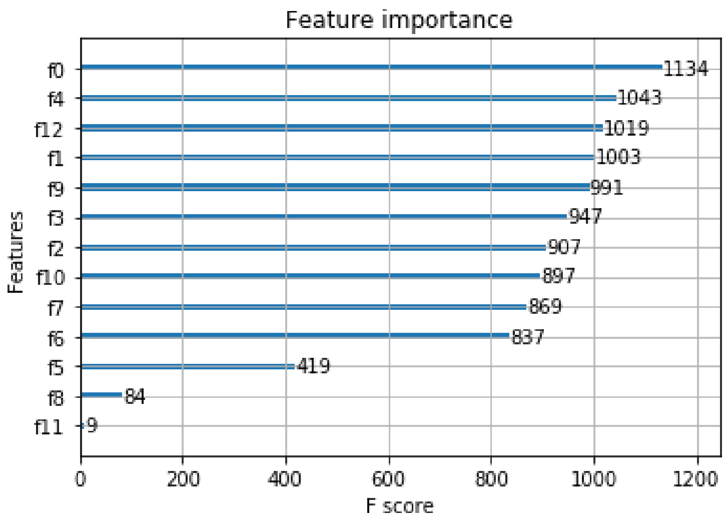
The following are the Hyperparameters we will tune: *max_depth, min_child_weight, gamma, subsample, colsample_bytree, learning_rate, n_estimators*. After implementing the optimal parameters, we get the following results from the test set.

Model	**Accuracy Score**	**F1 Score**
Tuned XGBoost	73.25%	.73

The accuracy score has improved from our initial model. The weighted F-score is the same as our initial model. Since the f1 score did not improve, we cannot conclude the tuned model as the better estimator. Accuracy score may increase because of imbalanced classes.

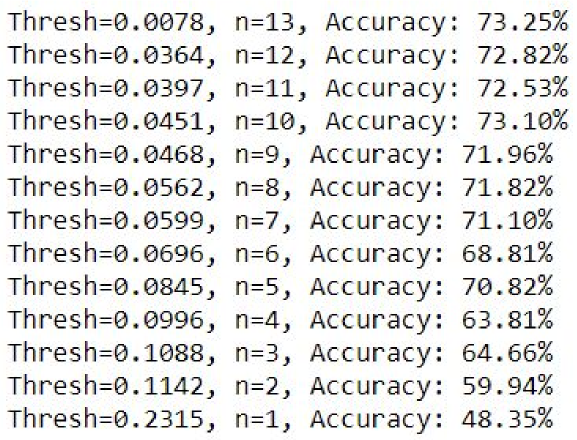
Feature Importance:

Tuning the hyperparameters for XGBoost did not improve model performance. We then see if we can further improve the model by implementing feature selection. The XGBoost library provides a built-in function to plot features ordered by their importance. 'Importance' provides a score that indicates how valuable each feature was in the construction of the boosted decision trees within the model, allowing attributes to be ranked and compared to each other.



We can see that F0 (Acousticness) is the most important feature and has the most influence on the model. F11 (Time Signature) has the lowest importance and has very little influence in predicting playlist category.

We then test multiple thresholds of features by importance. Starting with the feature with the lowest importance, we remove a feature one by one and calculate the accuracy score to see how the model changes with the new threshold. The accuracy scores below are based on the test set.

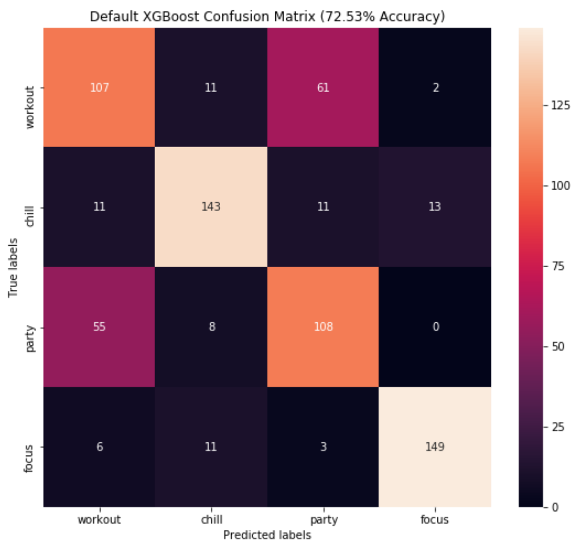


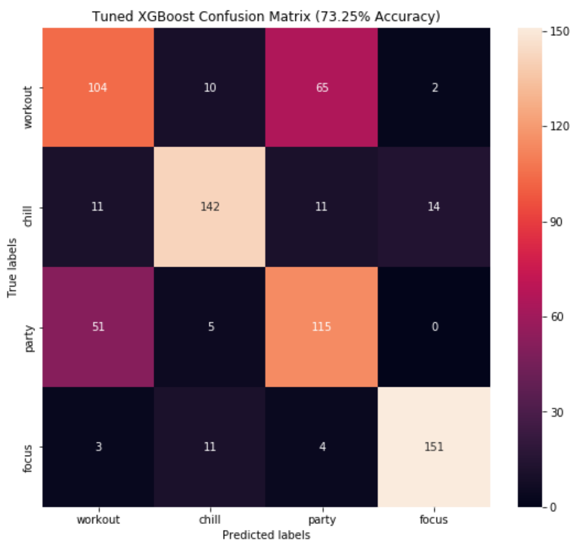
```
Thresh=0.0078, n=13, Accuracy: 73.25%
Thresh=0.0364, n=12, Accuracy: 72.82%
Thresh=0.0397, n=11, Accuracy: 72.53%
Thresh=0.0451, n=10, Accuracy: 73.10%
Thresh=0.0468, n=9, Accuracy: 71.96%
Thresh=0.0562, n=8, Accuracy: 71.82%
Thresh=0.0599, n=7, Accuracy: 71.10%
Thresh=0.0696, n=6, Accuracy: 68.81%
Thresh=0.0845, n=5, Accuracy: 70.82%
Thresh=0.0996, n=4, Accuracy: 63.81%
Thresh=0.1088, n=3, Accuracy: 64.66%
Thresh=0.1142, n=2, Accuracy: 59.94%
Thresh=0.2315, n=1, Accuracy: 48.35%
```

We can see that the accuracy is generally decreasing as the number of features decrease. Training on all 13 features gives us the best score. Therefore will use all features in our model.

Final XGBoost Model:

After tuning our hyperparameters and removing the time signature feature, we found our optimal XGBoost model at **73.25%** accuracy. Below are the confusion matrices of our original XGBoost Model and our tuned model.



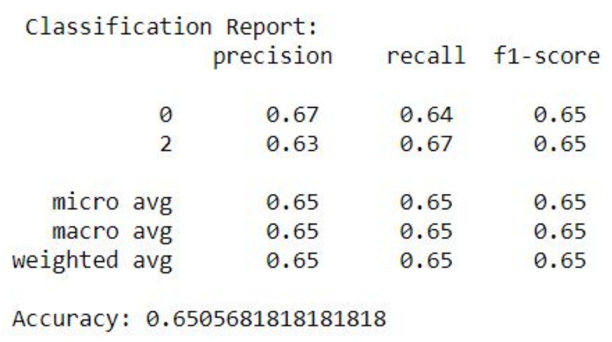


As you can see, the final tuned model is only a very slight improvement to the default XGBoost model. Notice that the scores for class '0' (workout) and class '2' (party) are consistently lower than the other classes for all our models. From the confusion matrix, you can tell that the workout and party class scores are low because our estimators often 'confuse' the two. For true labels of party, the model predicted workout 51 times. For true labels of workout, our estimator predicted party 65 times.This is to be expected since we found that workout and party distributions were very similar in our hypothesis testing stage.

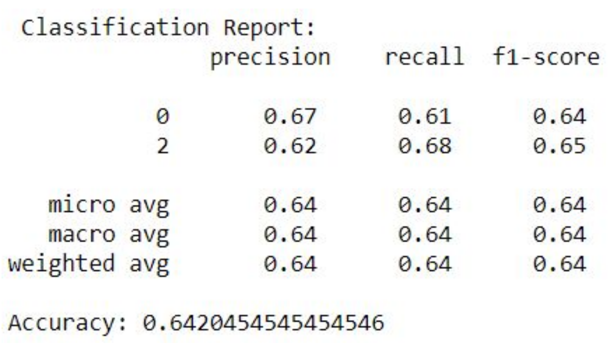
Ensembling:

Since our model is having difficulty classifying between workout and party songs, we will use an ensemble method to aid our model. Ensembling refers to the method of combining several machine learning techniques in order to improve predictions. In addition to our XGBoost model, we will train two more models on only class 0 and class 2. Then, we will get the average predicted probabilities of the ensemble of models to get our final predictions.

Random Forest and SVM had the next best performances when testing our base models. We will use these two methods in our ensemble. After training and tuning Random Forest (left) and SVM (right) on a subset of the data that only includes workout and party songs, we get the following results.



```
 Classification Report:
              precision    recall  f1-score

           0       0.67      0.64      0.65
           2       0.63      0.67      0.65

   micro avg       0.65      0.65      0.65
   macro avg       0.65      0.65      0.65
weighted avg       0.65      0.65      0.65

Accuracy: 0.6505681818181818
```

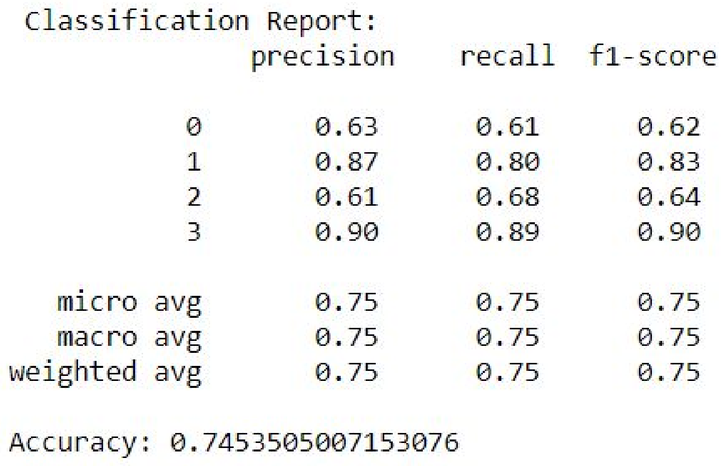


```
 Classification Report:
              precision    recall  f1-score

           0       0.67      0.61      0.64
           2       0.62      0.68      0.65

   micro avg       0.64      0.64      0.64
   macro avg       0.64      0.64      0.64
weighted avg       0.64      0.64      0.64

Accuracy: 0.6420454545454546
```

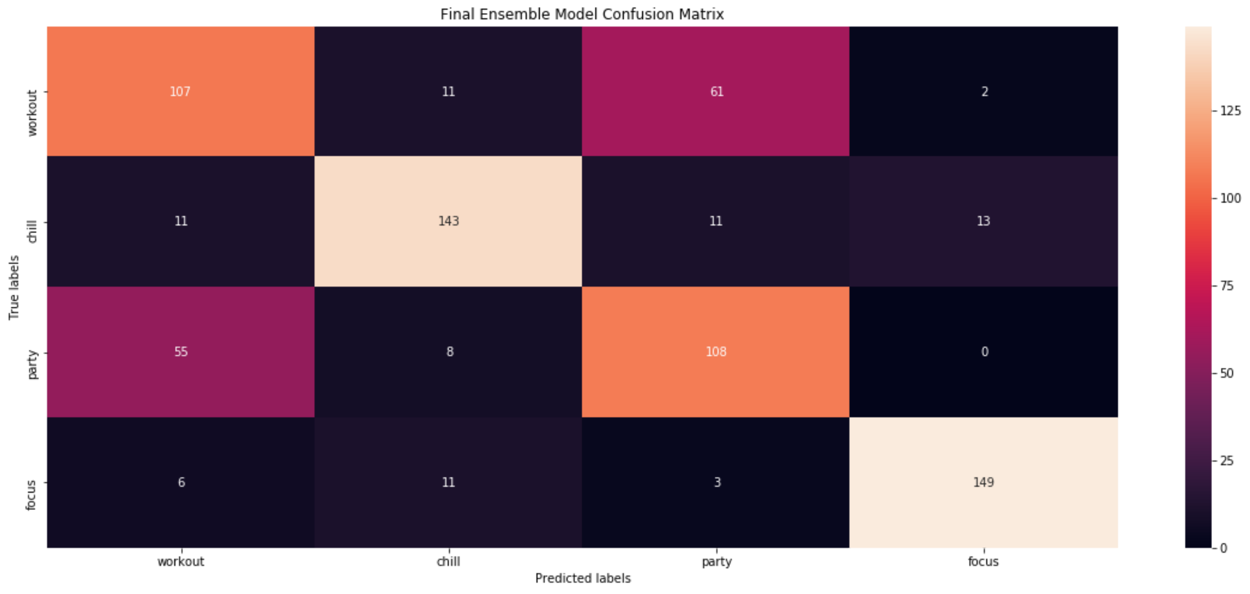
We then average the predicted probabilities outputted by the XGBoost, Random Forest and SVM models for class 0 (workout) and class 2 (party). After evaluating the predicted labels on our test set, we get a model with **74.53% accuracy** and **.75 f1-score.**

Final Ensemble Model:

Our final ensemble model improves by 1.28% accuracy in comparison to our XGBoost tuned model alone. Adding these additional models as classifiers for only workout and party playlist classes helped us reduce our model error.



```
 Classification Report:
              precision    recall  f1-score

          0       0.63      0.61      0.62
          1       0.87      0.80      0.83
          2       0.61      0.68      0.64
          3       0.90      0.89      0.90

  micro avg       0.75      0.75      0.75
  macro avg       0.75      0.75      0.75
weighted avg       0.75      0.75      0.75

Accuracy: 0.7453505007153076
```



Future Recommendations and Next Steps:

Although our ensembling method increased performance, our model confuses ‘workout’ and ‘party’ playlists frequently. This is the root cause of our model error. To only way to further improve our model is to add more features that further distinguish the distributions of ‘workout’ and ‘party’. In the future I would consider adding metadata of song artists or extracting more audio features using the python package Librosa.

In order to fully address the problem statement, our model must be put into production. Our model would be fed spotify songs beyond our dataset. I would recommend the client to build a platform that allows a user to select a category (mood, activity, genre, i.e.) which then automatically produces a playlist. Next step is to add more classes of playlist categories and re-train our model. The clients user will then have more options of playlists to access.