Spotify Project

Introduction:

Throughout history, it is evident that humans as a species have always been known to be story tellers, whether it is drawing our stories on cave walls during the stone age or telling our stories around a campfire in the wild west; the stories we tell have always been a representation of who we are, where we come from, and who we want to be. Nowadays, storytelling has embedded many forms like books, movies, podcasts, music, etc. yet there is one form among all that stands out from the rest, and that is music. Not everybody has the time to watch movies every day, not everybody likes listening to podcasts, and not everybody enjoys reading books, yet you can rarely find a person that does not listen to at least one song per day, whether that is voluntarily through their music apps, or involuntarily when they enter a restaurant with music in the background, or watch a TV commercial with a song in it, etc. Music is everywhere in our lives, and it is the most accessible form of storytelling, where it can range from watching orchestral music with hundreds of instruments to simply listening to one person using nothing but their own voice and sounds to transform emotions and ideas to an audible form. Personally, I have a very fond relationship with music, for my taste has always been outside the trends and norms of the modern era, where most songs that I listen to are oriented towards rock and metal rather than the more common commercial hip-hop, R&B, and rap. After years of observing the music people listen to and comparing it to their overall patterns of behavior, I have come to the opinion that music is one of the many factors that affects our lives on a subconscious level, meaning that the songs we listen to everyday, their lyrics, their tones, their themes and so on do in fact have a direct impact on us and how we view

these topics and can even help us form new thoughts and opinions about them, eventually having an almost direct impact on who we are as a person. On this basis, I decided to use my data science skills and build a machine learning model that detects whether or not I would add a certain song to my playlist to see how well a primitive algorithm can predict something that is so special to me. This report will summarize my work and the results I obtained and will give my overall philosophical opinions about such algorithms and how they can tell much more about human nature than we normally think about.

1) Data Collection and Cleaning:

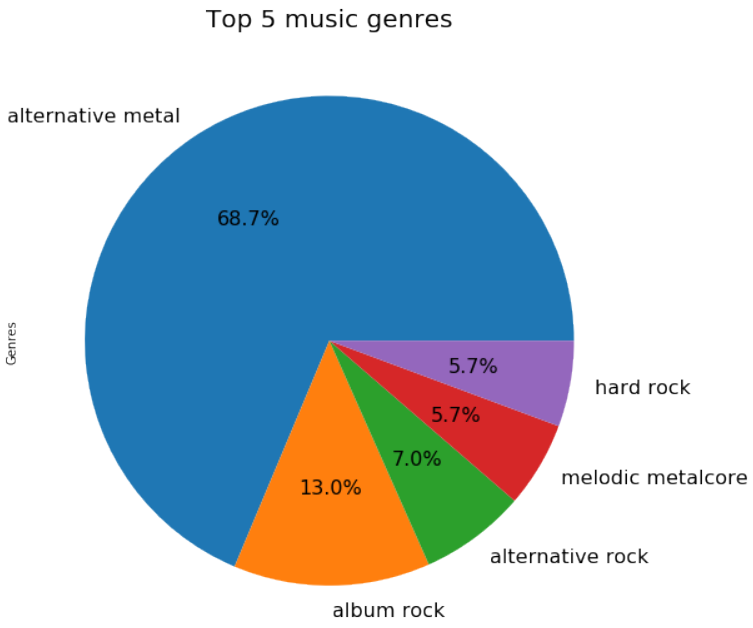
The first step in the project was to be able to transform my Spotify playlist into a data frame of rows and columns with multiple features in order to perform my analysis. Luckily, I found an amazing free website called Exportify (https://watsonbox.github.io/exportify/) where users log into their Spotify accounts, choose the playlist they would like to export as a csv file, and obtain a csv version of their playlist with the click of a button. The most impressive feature in this website is the fact that the csv returned contains very detailed information about the musical features and elements of a song, like Danceability, Energy, and Liveness, all of which are expressed in a numerical.

After importing the csv and converting it to a data frame using pandas, the next step was getting rid of entries that will cause problems to our analysis and future models. For example, some songs which are covers for actual songs (usually from YouTube artists) do not have values for the feature 'Album Name', which would have caused issues in the later stages of the project. Consequently, all cover songs were dropped from the dataset in order to avoid issues and errors in later stages.

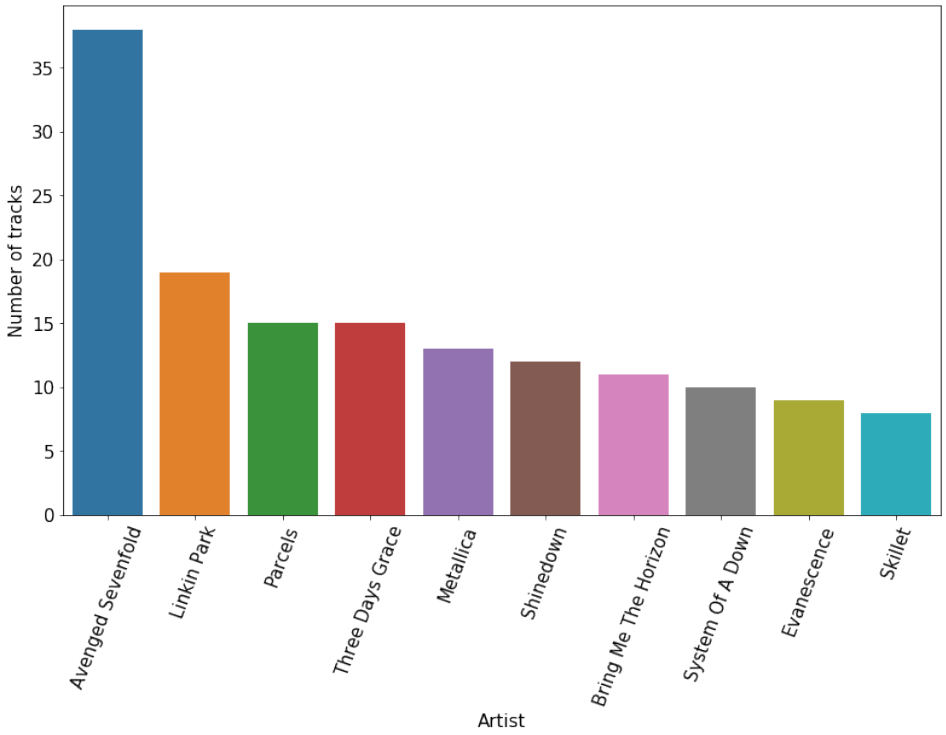
Similarly, some features that were of no use to my analysis, like a song's image URL, were dropped in order to decrease the dataset's overall size.

2) Exploratory Data Analysis:

After having imported all required data and have properly cleaned and prepared it, the time came for some basic exploratory analysis in order to get a sense of how the data is distributed among the many variables. First of all, I decided to visually represent the top 5 genres of music I listen to:

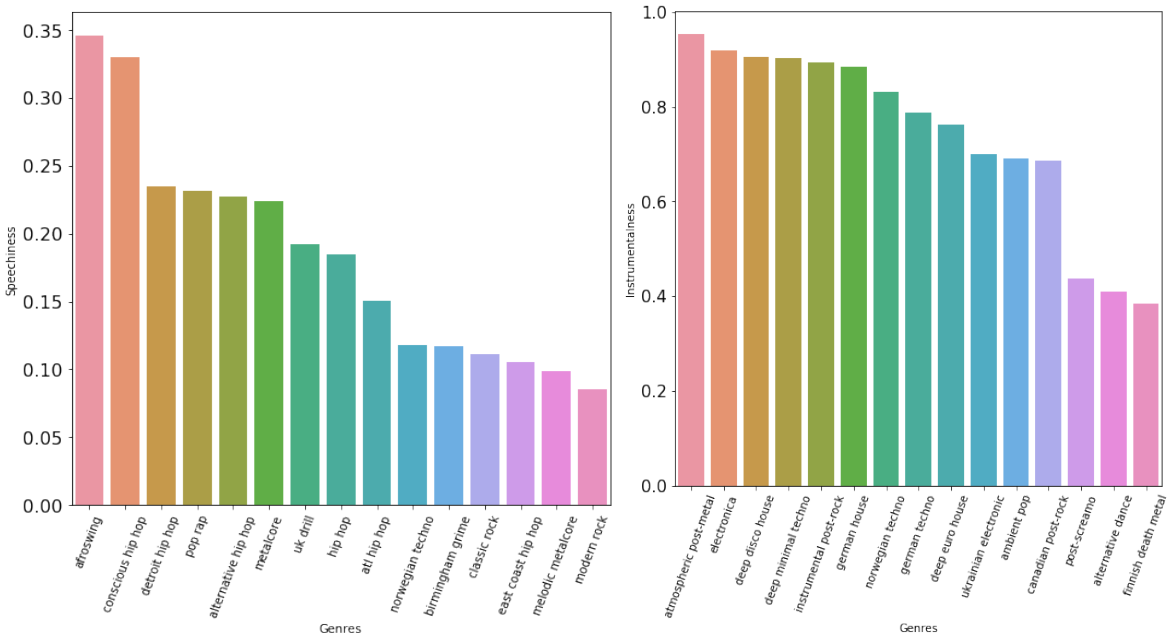


Obviously, my music taste is highly centered around rock and metal, where my top 5 genres in my playlist are all different types of rock and metal, which is an early indication of how much importance the genre of a song has in the model I will build. Furthermore, I decided to plot the artists with the greatest number of songs in the playlist:



The most relevant insight from the above graph is that the 3rd artist in the graph are Parcels, who are not even a rock band, which is important since their music takes a large chunk of non-rock music from the playlist, indicating that the number of non-rock songs that I like is in fact fewer than previously thought, since a significant number of them comes from a single band.

After obtaining a visual representation of my musical taste and preferences, I decided to relate them to the numerical musical features found in the data frame. A couple of the bar graphs are shown below:



The above 2 graphs represent the speechiness and instrumentalness of different genres. Speechiness is defined as the number of spoken words in a track, which definitely explains the fact that hip hop and rap songs are dominant in this category since it is in the nature of these songs to have many words spoken per second (for example Rap God by Eminem). On the other hand, instrumentalness represents how many instrumental sounds are involved in a song, which is definitely more present in rock and techno songs rather than traditional hip hop (for example To Live Is To Die by Metallica). The same type of graphs was created for other musical features which gave me further understandings of these features and what they mean.

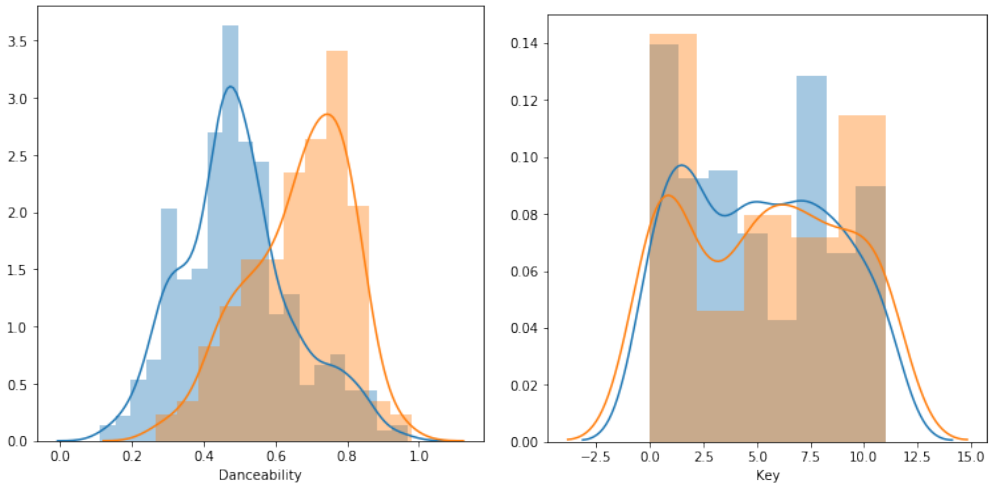
3) **Importing another Playlist:**

Up until this point, my dataset consisted of only my songs, which would be of no use on their own since my goal was to eventually use a classification algorithm which requires inputs of both 1s and 0s (yes and no) which represent whether (1) or not (0) a song would make my playlist. For this reason, I asked for a friend who has a very different music taste than mine to send me his playlist,

and I chose him in order to have a final dataset which is diverse in 1s and 0s. After importing his playlist, transforming it to a dataset and cleaning it, I merged the resulting dataset into mine and obtained a final dataset consisted of 813 entries which is more than enough for building a model.

4) Comparing Features:

In order to further understand the features of my dataset and how they change according to the genres of music, I decided to graph the density plot of the features in both playlists (mine and in my friend's) to see how they compare in addition to the fact that this will serve as a starting step to see which variables do in fact have an effect on whether or not a song would make my playlist (primitive feature selection).



The above graphs are density plots of the Danceability and Key of the songs found in both playlists, where one distribution is represented in blue and the other in orange. In brief, the more overlapping the 2 distributions of a certain variable are (like Key), the less the difference between the means of this variable hence the less the impact of this variable on the target variable. Similarly, the

further the 2 distributions of a certain variable are from each other the more they might be significant towards the target variable. After plotting the density plots for all variables, the main takeaways are:

1) The largest differences come from the danceability and valence of a song, where my playlist has lower means in both values, which makes sense since rock music is not known to be “dance friendly” and lower valences indicate darker/sadder songs, something which is reflective of the true nature of both playlists.

2) The remaining features are also logical but are close to each other, so they might not end up having a significant effect when we reach the step of feature selection. (Further analysis required)

5) Transforming Categorical Data:

Before proceeding to feature selection and model building, I had to deal with transforming categorical data to numerical data. The two categorical variables at hand were Artist Name, and Genres, both of which have hundreds of possible values, ruling out the most popular method when it comes to these kinds of problems called one hot encoding, since it creates one additional column for every possible categorical variable, leading to a very large dataset with hundreds of columns. For this reason, I decided to come up with my own strategy, which assigns each value a number corresponding to the number of times it appears in the column. For example, I have 38 Avenged Sevenfold songs, so every Avenged Sevenfold in the Artist Name column is replaced by the

number 38 (same logic applies for all artists). This way assured that enough weight is put to the number of times a certain artist and genre appears in their respective columns and ensured that all my dataset has become numerical and ready for further analysis.

6) Feature Selection:

One of the most common feature selection methods is ANOVA (analysis of variance) testing, where each variable is tested against the target variable (whether a song is in the playlist) and checks whether or not the variable does in fact have a direct effect on the target variable, all of which are done using the statistical concept of p-values (details found in full code). After performing the test on my dataset, the irrelevant features that I later dropped appeared to be:

- Energy
- Liveness
- Key

Dropping insignificant variables is important since it reduces the chance of overfitting my model, which is a common issue that should be avoided.

7) Model Building:

As previously mentioned, my main objective was to detect whether or not a song would make my playlist, making the problem a classification problem where 1 represents yes and 0 represents no. I decided to approach the problem by using the four most common classification algorithms and

compare their performances. The algorithms chosen are K Nearest Neighbor (KNN), Logistic Regression, Support Vector Machines, and Random Forest Classifiers, all of which work in different ways but have one common goal, predicting the binary value of a target variable with the lowest possible error. I split my data set into a training and a testing set (80-20 split), where I train the models and fit them on the former, and test for accuracy and performance on the latter, in order to eventually determine the top performing models. Below is a small sample for the support vector machine model and its accuracy:

Model 3: Support Vector Machine

In [30]:
```
#Import svm model
from sklearn import svm

#Create a svm Classifier
svm1 = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
svm1.fit(X_train_scaled, y_train)

#Predict the response for test dataset
svm_predictions = svm1.predict(X_test_scaled)
```

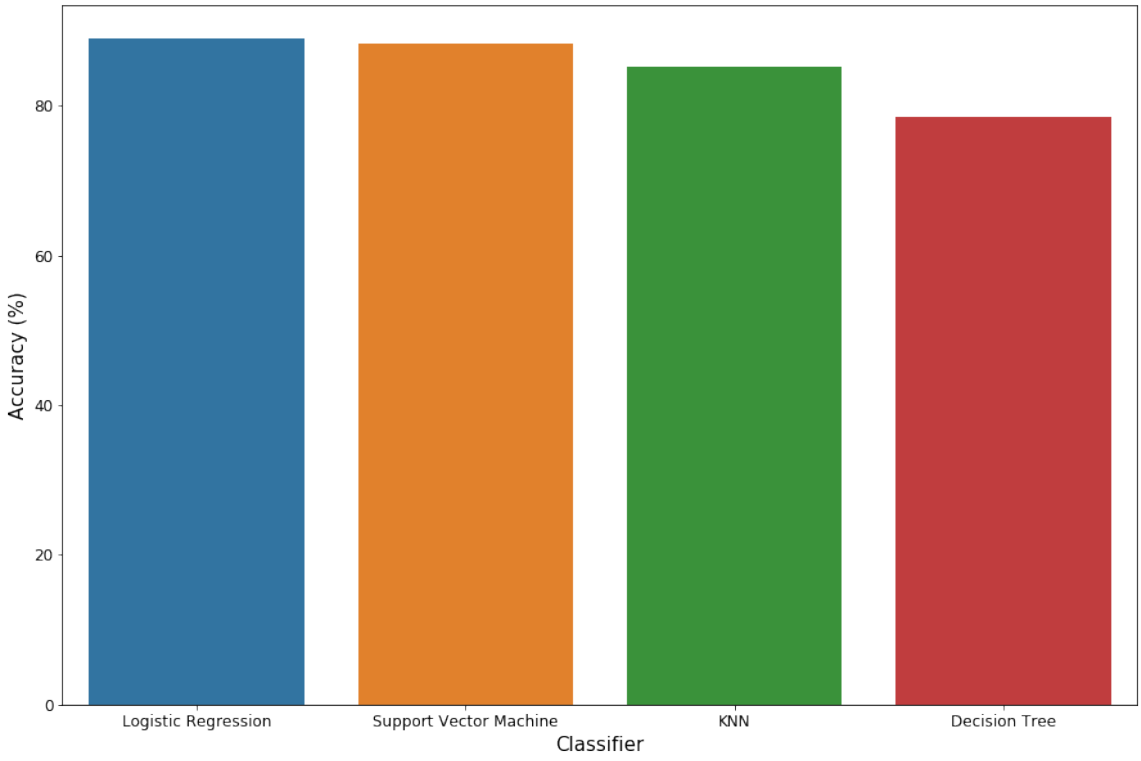
In [31]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct ?

svm_accuracy = metrics.accuracy_score(y_test, svm_predictions)

print("Accuracy:", svm_accuracy)
```

```
Accuracy: 0.8834355828220859
```

8) Model Performances and Optimization:

The two main metrics I use when evaluating ML classification models are MSE (mean squared error) and the ROC curve (Receiving Operator Characteristic Curve). MSE basically measures the average error of my model, where error is defined as misclassifying an entry. As for the ROC

curve, it focuses more on representing the true positive, true negative, false positive, and false negative rates of the model. After fitting all four models to my training set and evaluating their performances on the testing set, I plotted their accuracies in the following simple bar graph:



The models with the highest accuracies appeared to be Logistic Regression and Support Vector Machines, each with an accuracy of 88.95% and 88.34% respectively. When it comes to classification models, a certain level of accuracy cannot be generalized as "good" or "bad" since it highly depends on the nature of the problem at hand. When building a model that classifies whether or not a patient would develop a cancerous tumor or not, an 88% (round to 90%) accuracy would be weak, because misclassifying 10 out of each 100 patients (especially with a high false negative rate) is a very pool accuracy, since the results are life defining for people. However, a 90% accuracy for a model that detects whether I would like a movie or not is actually really good,

since the result have no major impact on me and the worst-case scenario is not liking a movie, which is definitely not the end of the world. In the case of this project, it is skewed more towards the latter example, where the worst-case scenario is not liking 1 out of 10 (approximately) songs that my model predicted I would, so an 88% prediction accuracy is more than enough for this problem.

Since these types of projects are for me to improve my skills in data science, I decided to practice tuning the hyperparameters of my model to see if I get an improvement in model accuracy despite being satisfied with the current accuracy. For this reason, I used grid search as a method to check for the best combination of model parameters that give the highest model accuracy for both the logistic regression and support vector machine models. After fitting both models using grid search, I selected the best versions of each and used it to evaluate it on the testing set. Check below for an example on optimizing the SVM model:

```
In [48]: # SVM Optimization

         param_grid_svm = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}

In [49]: grid = GridSearchCV(svm1,param_grid_svm,refit=True,verbose=2)
         grid.fit(X_train_scaled,y_train)
         [CV] C=0.1, gamma=1, kernel=rbf ......................................
         [CV] ....................... C=0.1, gamma=1, kernel=rbf, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=rbf ......................................
         [CV] ....................... C=0.1, gamma=1, kernel=rbf, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=rbf ......................................
         [CV] ....................... C=0.1, gamma=1, kernel=rbf, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=poly .....................................
         [CV] ....................... C=0.1, gamma=1, kernel=poly, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=poly .....................................
         [CV] ....................... C=0.1, gamma=1, kernel=poly, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=poly .....................................
         [CV] ....................... C=0.1, gamma=1, kernel=poly, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=sigmoid ..................................
         [CV] .................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=sigmoid ..................................
         [CV] .................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
         [CV] C=0.1, gamma=1, kernel=sigmoid ..................................
         [CV] .................... C=0.1, gamma=1, kernel=sigmoid, total=   0.0s
         [CV] C=0.1, gamma=0.1, kernel=rbf ....................................
         [CV] ..................... C=0.1, gamma=0.1, kernel=rbf, total=   0.0s

In [50]: print(grid.best_estimator_)

         SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)
```

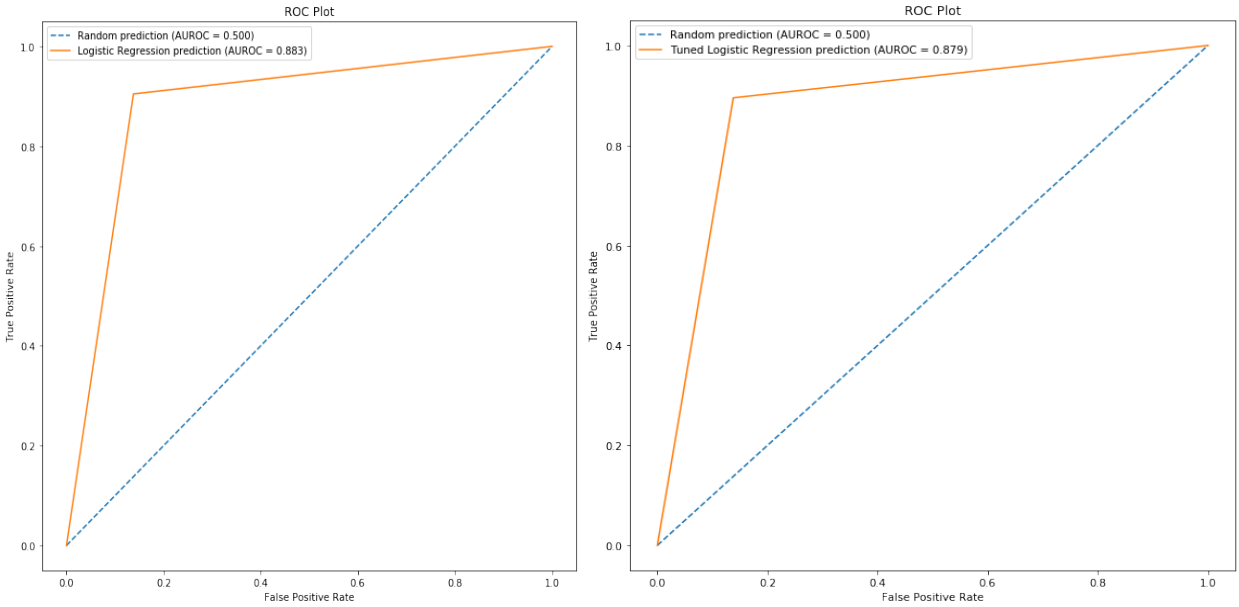
The results of my hyperparameter optimization were the following:

	Models	Accuracy (%)
0	LR	88.96
1	Tuned LR	88.34
2	SVM	88.34
3	Tuned SVM	82.82

To my surprise, the accuracy of both models decreased (very small decrease for LR) after tuning them, which might be due to the fact that the grid search is done only on the training set and returns the best possible model according to this set. I will make sure to follow up with the issue because I found it rather interesting, but as previously mentioned, an 88.96% (90%) is really good for the nature of this problem, which is why I decided to choose Logistic Regression and tuned Logistic Regression as the two options for my final model since their accuracy is very close.

Finally, I decided to plot the ROC curves of both the Logistic regression and the tuned Logistic Regression to choose which one to choose as my final model. For clarification the ROC curve plots the True Positive Rate against the False Positive rate of a model, so the closer the tip of the orange curve to the top left corner the better, which is numerically represented by the area under the curve (AUROC).

The 2 plots below compare the ROC curves of the standard and tuned logistic regression models, where the standard model has a higher area under the curve, meaning a lower false positive and higher true positive rate, which is a good thing to look for in classification models.



Overall, the standard Logistic Regression model had the higher accuracy value and the higher area under the curve compared to the tuned one, which is why I opted to use it as my final model.

9) Conclusion:

All in all, this project gave me the chance to improve my data science skills in a field that I love and have passion for, the field of art and music. I discovered that using visualizations and exploratory analysis on something you care about gives you much more significant insights than previously thought. In my case, I always knew that my taste is rock oriented, but I found out that the difference between knowing and feeling is seeing, where a couple of bar graphs and pie charts made me face the reality that I do not just "like" rock music, I actually adore it since almost no other genres appears in visualizing my top music choices. As for the machine learning part of the project, it made me realize how important it is to actually understand the industry of the project I

am working on, for it is the best way to actually understand how the algorithms are working instead of just writing a couple lines of code and applying them on a random data set. I have worked on several data science and machine learning projects up until this point, but this was by far the one I found myself actually interested and engaged in what I am doing instead of just applying code for a certain objective that I do not actually care about deep inside me. Apart from the practicing my skills benefits of this project, I reaffirmed to myself that I should always work on things I deeply care about and have passion towards, which is the only way to assure my constant will and desire to leave an impact on all projects I work on now and in the future.

10) Final Thoughts:

One of the main reasons why I have chosen Data Science and Analytics as a career path for me is the fact that it combines science and philosophy in a way that most people do not notice. To elaborate, the main objective behind almost every project is using mathematical modeling and coding (science) to derive insights to improve the business in hand, where the insights can vary from purchasing behavior of consumers for a specific product, to the number of projected points an NBA star will score in a game, and the business can also vary from a grocery store trying to attract more customers to top tier companies trying to figure out ways to increase consumer screen time on their applications/platforms. No matter the objective and purpose, there is one thing that is common among all projects, which is the fact that all the information we need and want to use is out there in nature and represented by numbers waiting to be fed to an algorithm that predicts future outcomes of the problem in hand. Personally, I am a person who likes to view the world in a very philosophical manner, where I like to look at things in an existential and deep way, which is why this project made me go down the rabbit hole of what it means for a mathematical algorithm

to be able to predict something purely subjective, like my music taste. Just like I mentioned in my introduction, music is everywhere in our lives and takes on several forms like conventional songs, sports team chants, religious choirs, and many more forms that all do one common thing, unify humans. A great example to this is watching Metallica's 1991 Moscow concert, which is known to be one of the largest concerts in history, where 1.6 million people forgot all their political and ideological differences for a couple of hours and did nothing but sing and dance along some of the greatest tracks of all time. The point that I am trying to reach is that something so emotional and relative like music can still be modeled into mathematical algorithms that were discovered by scientists throughout history, which begs me to ask myself how can something so subjective be condensed down to simple numbers and models? What does this imply about humans' emotions and consciousness if everything we feel and like can be used by numerical models discovered in nature? My model is primitive and was built for improving my skills, but what if someone with more experience than me builds an advanced AI with the same purpose, who would know more about my music taste, myself or the AI? All these questions come from one central idea: We come from nature, but so do the mathematical rules behind every machine learning and artificial intelligence model, so who will end up being the superior power down the line?