# Explore and Classify the Music on Spotify

110354032 徐偉鑫
110354020 曾子朋
110354016 蔡耀德
110354025 林映孝

## Abstract

We wonder whether the hits have similar characteristics, and if it has, can these be used to predict which songs will be hits in the future? Hence, we build models for automatic classification of music genres and predict potential hits through different machine learning models. We use unsupervised learning such as PCA and MCA to explore the data and visualize the data and then we use supervised learning such as Nearest Neighbor, SVM, Decision Tree, Random Forest and xgBoosting to build the classification model. These models analyze various characteristics of song, including tempo, key, valence, energy, acoustics, dance ability, and loudness. During the building model process, we tune the hyperparameters or use the kernel method to improve accuracy. At the end, we will compare all the models and choose the best among these models.

## Dataset Introduction

This dataset(Top Hits Spotify from 2000-2019) contains audio statistics of the top **2000** tracks on Spotify from 2000-2019. The data contains about **18 columns** each describing the track and its qualities.

We choose a few obscure variables to introduce:

### Dependent Variable:

- speechiness: Speechiness detects the presence of spoken words in a track.
- acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
- instrumentalness: Predicts whether a track contains no vocals.
- liveness: Detects the presence of an audience in the recording.
- valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.(e.g. sad, depressed, angry)

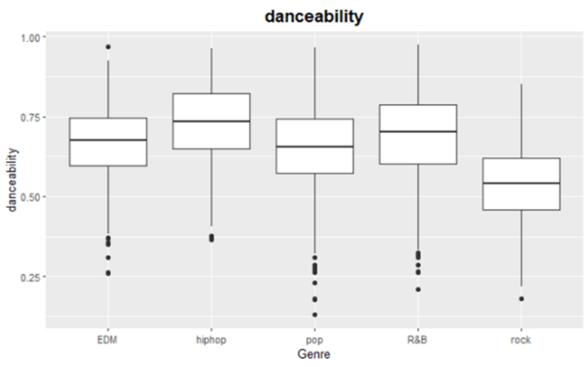
### Independent Variable:

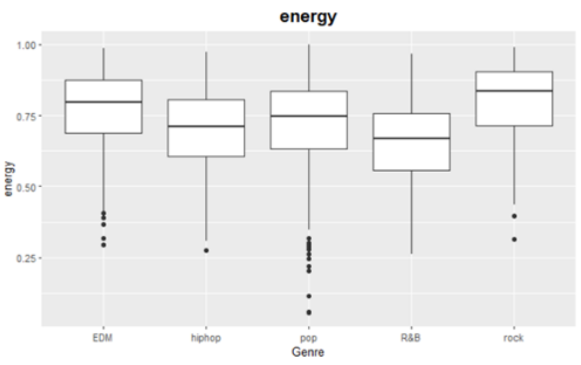
- genre: Genre of the track.(e.g. pop, rock, metal, hip hop...)

## Exploratory Data Analysis (EDA)

### Outlier Analysis

Observe the boxplots of the numeric variables among the five genres. Then find the outliers in each quantitative variable according to the median±1.5×IQR criterion. To improve the classification of genres, we delete a song that has more than six variables as outliers. (The boxplots of variables danceability and energy.)
There are 56 outliers in the data accounting for about 2.9%.

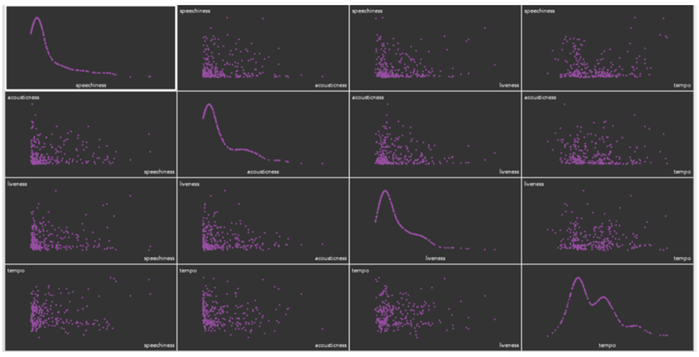


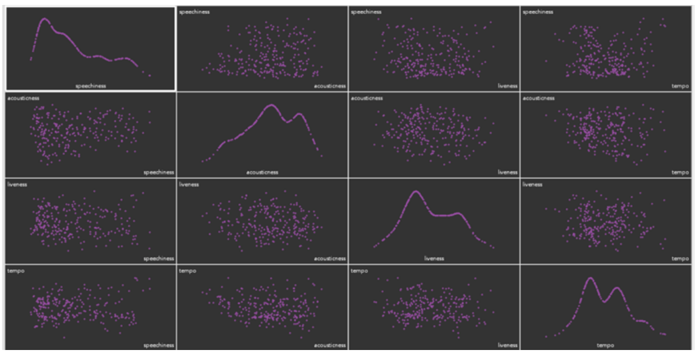


## Data Transformation

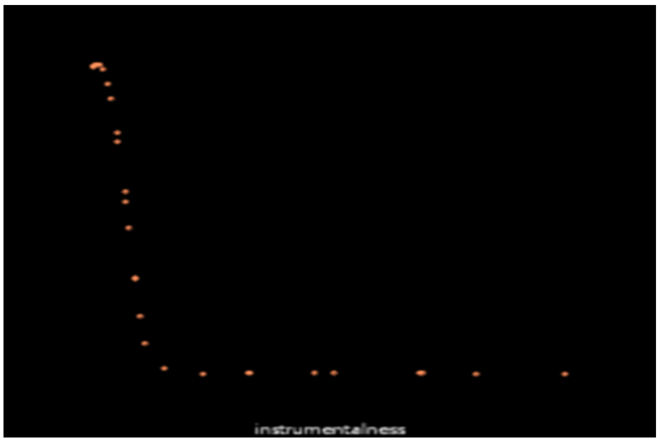
First, we delete the song name and singer name, these two variables are not what we want to discuss.

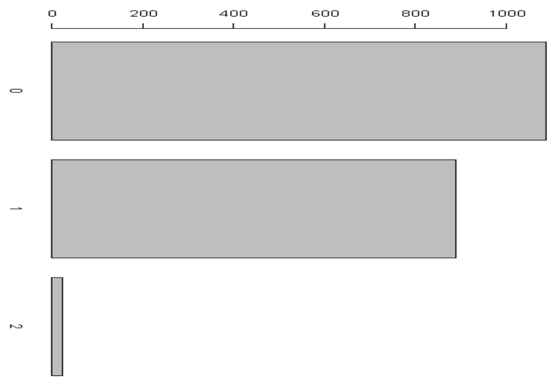
Second, after observing the density plot and scatter plot matrix of each variable, we found that the four variables are seriously skewed, so we transform the four variables (**speechiness**, **acousticness**, **liveness** and **tempo**), using log and root sign respectively.





Third, since more than half of the data of the variable "**instrument**" are concentrated in 0 and 0.1, resulting in skewed data, we convert the variables into categorical variables according to the information of the data description.We turn 0 ⇒ 0 and define it as "contains vocal content", 0.1~0.5 ⇒ 1 as "contains some vocal content", and over 0.5 ⇒ 2, defined as "contains no vocal content".





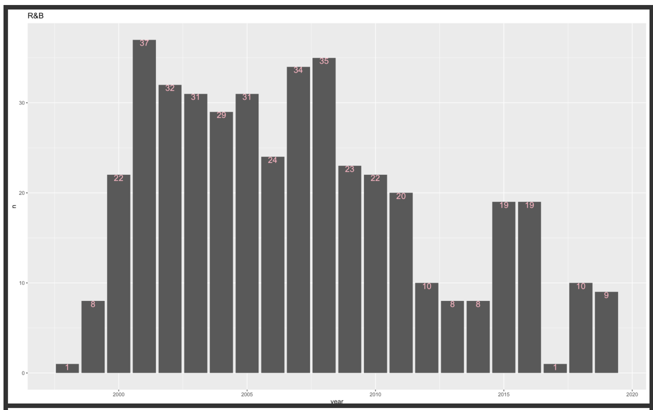
Finally we look at the category of our response variable "**genre**", it is found that many categories have 2 to 3 attributes, which makes us have as many as 59 categories, so we decided to only capture the first important music category as a representative.

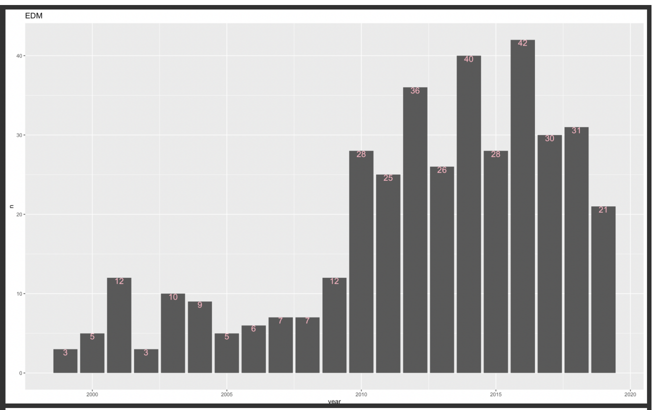
Furthermore, we remove the categories with too few observations(e.g. Folk/Acoustic, Latin, World/Traditional...).

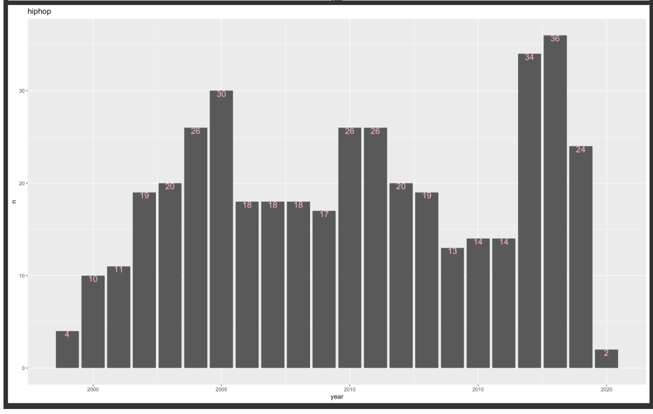
Eventually, we get five categories of genre, namely **"EDM"**, **"hiphop"**, **"pop"**, **"R & B"**, **"rock"**.

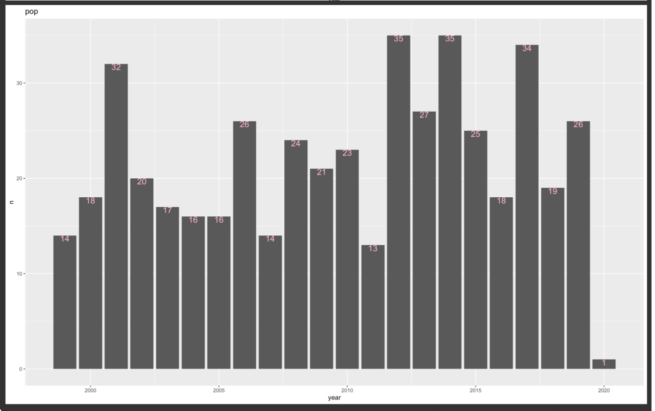
## Time Data Analysis

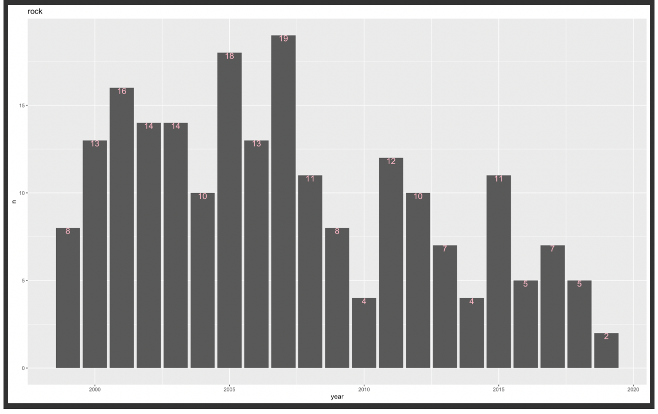
We observe how much data there is for the same genre of song in different years.











In R&B, rock, We found that the trends in songs were concentrated in the early 2000s. In EDM, We found that the trends in songs were concentrated in the late 2000s.

In pop, hiphop, We found that was very popular in every generation.

Hence, we know that R&B, rock pop was only popular in the early 2000s, EDM was only popular in the late 2000s and pop,hip hop has been popular for all generations.

## PCA and MCA

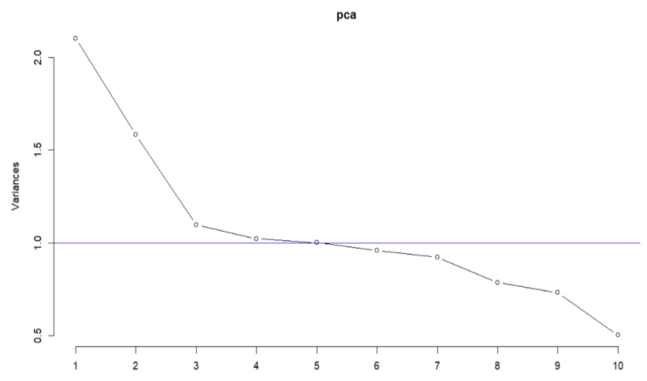
We will use PCA and MCA to explore numerical data and categorical data respectively. Our goal is to observe the genre specifically separated in groups by data visualization; also explore the data in a multidimensional perspective.

## PCA

From the summary of PCA shown below, we notice that the first 2 components only give us 33% of variance explanation and it reaches to 70% when using 6 components.

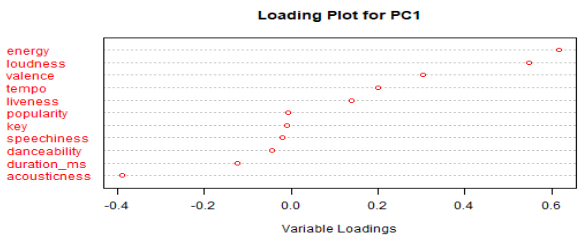
| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | ….. |
|---|---|---|---|---|---|---|---|
| **Sd** | 1.4497 | 1.2587 | 1.0478 | 1.01205 | 1.00120 | 0.98014 | ….. |
| **Prop. Var** | 0.1911 | 0.1440 | 0.0998 | 0.09113 | 0.09113 | 0.08733 | ….. |
| **Cum. Var** | 0.1911 | 0.3351 | 0.4349 | 0.52801 | 0.61914 | 0.70647 | ….. |

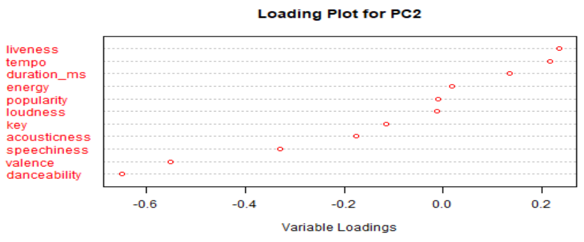
**Table: summary of PCA**

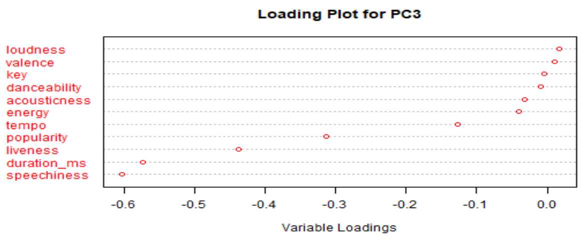


**Scree Plot for PCA**

However, the Scree Plot indicates that after PC3, the trend of changing variance is getting smooth. Since we do not need to consider the extreme high dimension issue (only 17 variables), we will visualize the data with 3 dimensions.

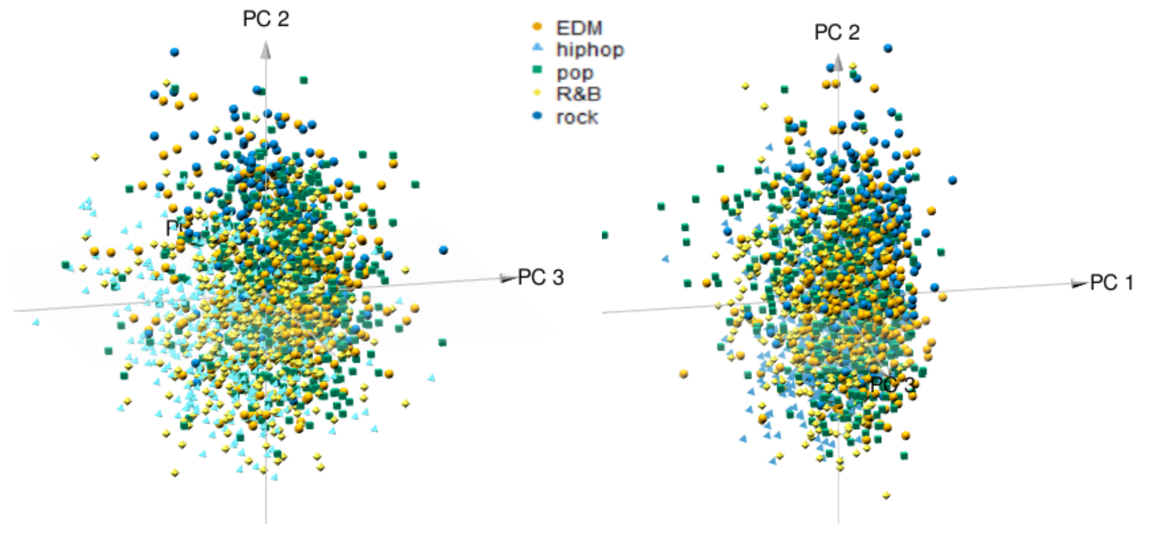


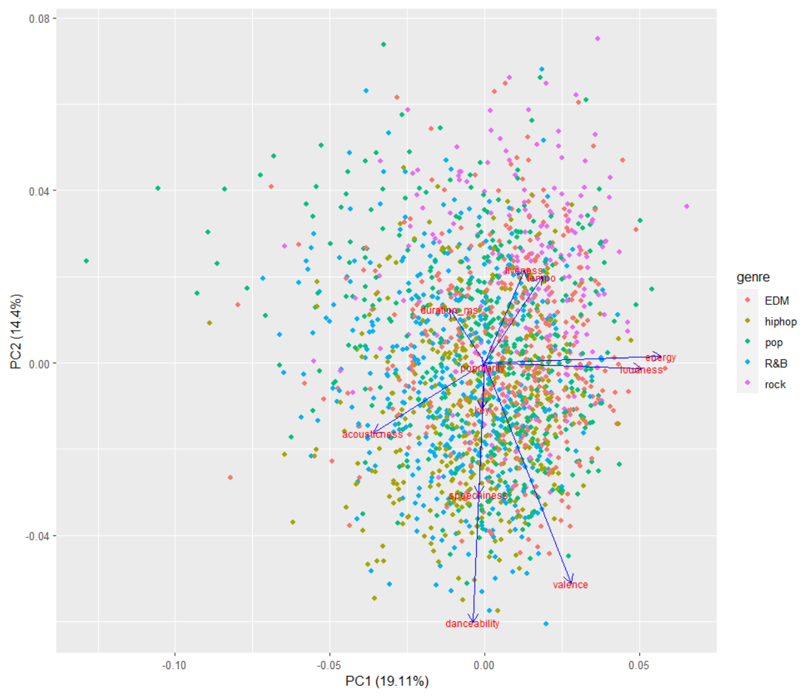




These tables give us the overview about the relationship between the variables among each principal component. For example, in PC1, negative loading variables may have similar characteristics, just like positive loading variables.

In our opinion, PC1 may represent the quantity of musicality. Key, speechiness, ms, and acousticness are highly musicality scale; energy, loudness, and liveness are more about emotion of the music.





From the 2D plot, those loading vectors verify that:

1) In PC1, **EDM** and **Rock** both have positive loading, because both kinds of music emphasize on emotion scale, like tempo, energy, and loudness.

2) On the other hand, **hip hop** and **R & B** emphasize musicality, like Key, speechiness, and acousticness.

By 2D plot, we roughly can distinguish 5 genres separately; fortunately, we can visualize the obvious 5 different genres in the 3D plot above with different angles.
In both 3D and 2D plots, **pop** music is everywhere on the plot.

## Summary

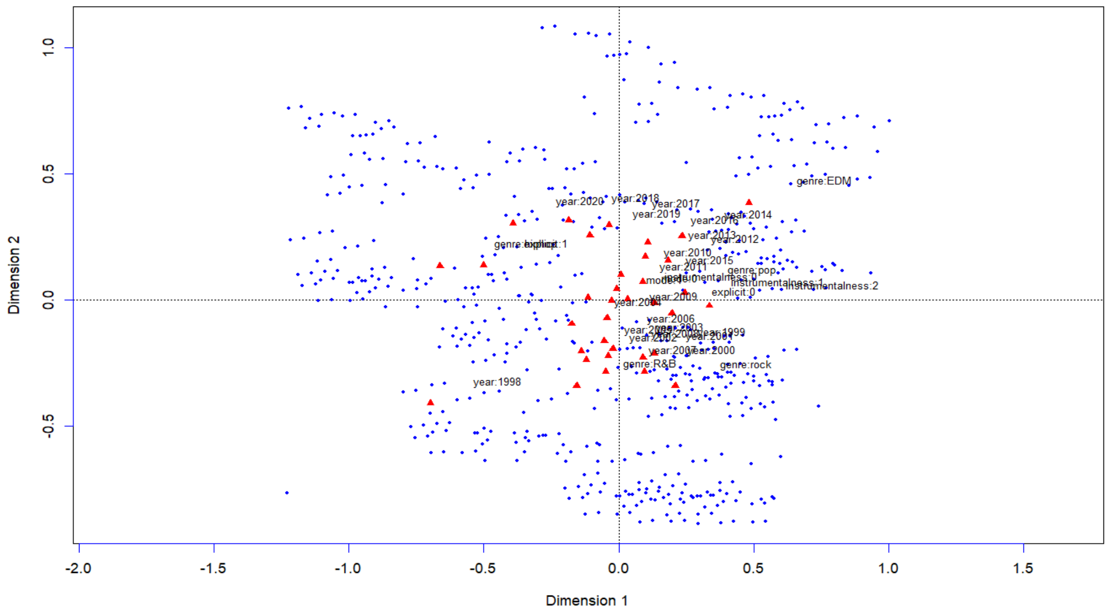
Because it is hard to distinguish the group in 2D and even it is hard to draw the boundary line in 3D either, from here we can assume that the ML model with **kernel method** may have better result of classification. Also, **linear boundaries** will not perform well.

## MCA

From the table below, after adjusting the inertia, JCA has a much better cumulative % of explanation in 2 dimensions, from 10.3% increase to 83.9%.

| | Regular MCA | | JCA | |
|---|---|---|---|---|
| **DIM** | **value** | **cum %** | **value** | **cum %** |
| 1 | 0.343675 | 5.7% | 0.059113 | |
| 2 | 0.275953 | 10.3% | 0.024862 | 83.9% |
| 3 | 0.236796 | 14.3% | 0.001872 | |
| 4 | 0.230972 | 18.1% | 0.001095 | |

From the JCA plot shown below, we roughly can see that **R & B** and **Rock** have an obvious relationship with the early 2000 year and **EDM** has an obvious relationship with the late 2000 year. we also can see explicit variable = 1 (has sexual or violence lyrix) is highly relate to Hiphop and on the other hand, explicit variable = 0 is relate to **EDM** and **pop**. In addition, four genre tags (not including pop) are far away from each other, so we can distinguish 5 different genres on this JCA plot too.



## Summary

so I can conclude that:

| | **EDM** | **Hip Hop** | **Pop** | **R&B** | **Rock** |
|---|---|---|---|---|---|
| **Year** | New | Both | Both | Old | Old |
| **Explicit** | 0 | 1 | both | 0 | 0 |

Integrate the conclusion by PCA plot and PC analysis,

1) **pop** has all the characteristics that the other 4 genres have. Therefore, it is hard for us to distinguish the boundaries for **pop** music.

2) It is hard to separate **Rock** and **R & B** in categorical data, but as mentioned in PCA, Rock has positive loading and **R & B** has negative loading.

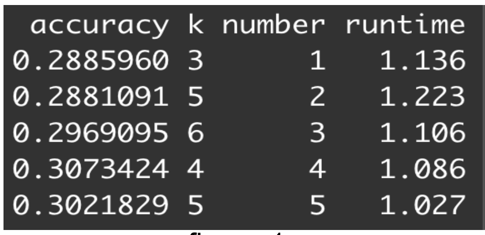
## Model

### Method

Step1: Divide the data into training and testing data where the ratio of these two data is 7:3.
Step2: Use these two data to build a model and train model.
Step3: For increasing model accuracy, we execute a model five hundreds times.
Step4: Evaluate and improve model accuracy.
Step5: Choose the best model to predict new data.

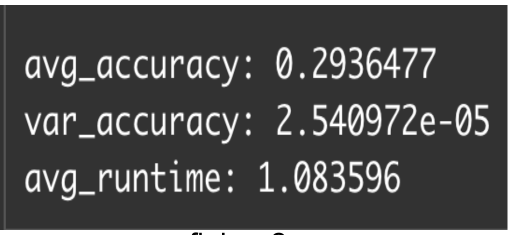
### NN:K-Nearest Neighbor ( Base Model)

We use Euclidean distance measure to see the predicted distance from the surrounding k data points, but the category variables must be converted into dummy variables, and all numerical variables need to be standardized to avoid that the units are different and the distance calculation is distorted. And then, we can check every single accuracy & running time ( figure1 ) and the mean of accuracy and running time  ( figure2 )



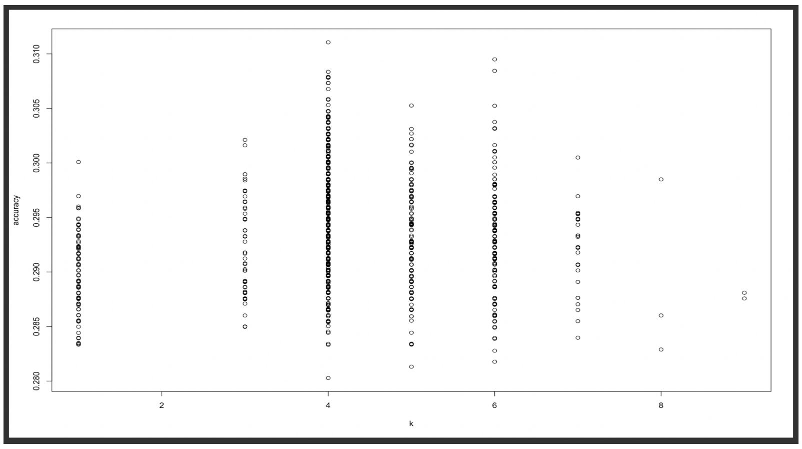
```
 accuracy k number runtime
0.2885960 3      1   1.136
0.2881091 5      2   1.223
0.2969095 6      3   1.106
0.3073424 4      4   1.086
0.3021829 5      5   1.027
```

figure 1



```
avg_accuracy: 0.2936477
var_accuracy: 2.540972e-05
avg_runtime: 1.083596
```

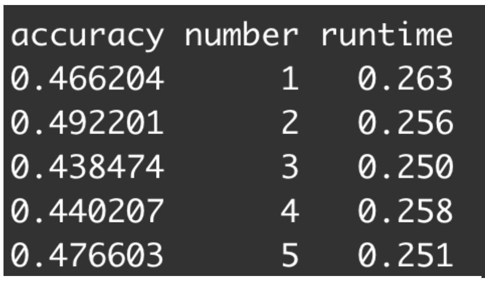
fighre 2



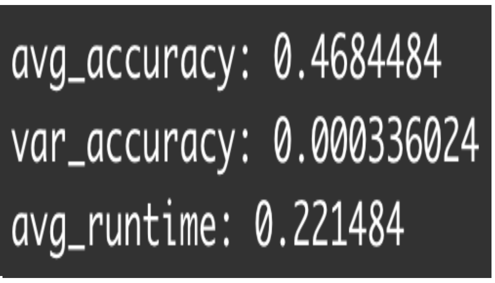
From the left graph, we found that most of the K values are at 4 & 6 in five hundreds times simulations.

### NN:Kernel-K-Nearest Neighbor (Improved Model)

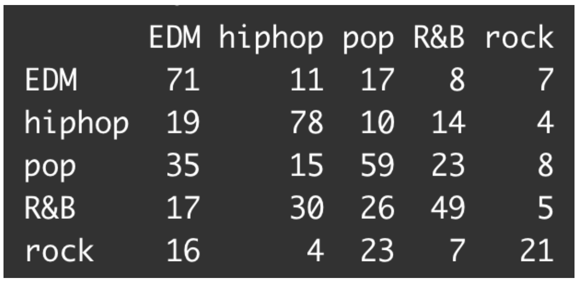
Since the accuracy of the KNN method is too low, we use Kernel-KNN which improves the computational power of linear learning machines by mapping data into a high-dimensional feature space. It can be realized by substitution of a kernel distance metric for the original one in Hilbert space. In this case, we use Minkowski distance and the Gaussian kernel method to execute the kknn model. And then, we check every single accuracy, running time (left figure) and the mean of accuracy and running time (right figure).



```
accuracy number runtime
0.466204      1   0.263
0.492201      2   0.256
0.438474      3   0.250
0.440207      4   0.258
0.476603      5   0.251
```



```
avg_accuracy: 0.4684484
var_accuracy: 0.000336024
avg_runtime: 0.221484
```


```
        EDM hiphop pop R&B rock
EDM      71     11  17   8    7
hiphop   19     78  10  14    4
pop      35     15  59  23    8
R&B      17     30  26  49    5
rock     16      4  23   7   21
```

Observed the classification table, we found that the correct number of classifications on the diagonal line. On the other hand, the correct classifications of every genre is EDM = 71, hip hop = 78, pop = 59, R&B =49 , rock =21.

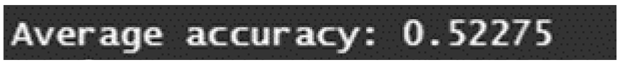
**NN conclusion:**

| | KNN | Kernel-KNN |
|---|---|---|
| **Average Accuracy** | 0.29365 | 0.46845 |
| **Vainance Accuracy** | 0.00002 | 0.00034 |
| **Average Runtime** | 1.08360 | 0.22148 |

Compare the accuracy of these two models, we discover 0.46845 is greater than 0.29365, so the kknn is better than the knn. At the same time, we compare the running time of these two models, because we discover 0.22148 is less than 1.0836, the kknn is better than the knn. Clearly, whether it's accuracy or running time, Kernel-KNN is more efficient than KNN.

## Linear SVM:

Use the **ordinal SVM** method and choose cost=100,200,300,400,500. Fitting on the training data with 10-fold cross-validation and running the loop 50 times to find the minimum of average accuracy. We can find the model has minimum mean accuracy with C=400.


```
Average accuracy: 0.52275
```

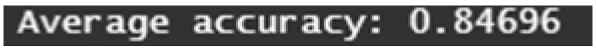
## Nonlinear SVM:

Consider two types of kernel function: RBF function and polynomial function

### RBF kenel function

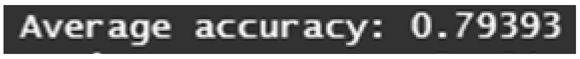
We use **Grid search** to find the best parameters (cost, gamma) over the region of [100,1000]x[0.1,1.0]. Based on 10-fold cross-validation, we can obtain an optimal cost and gamma by the training data, fit a new SVM model for all training data and compute the accuracy.

Running the loop 10 times (it takes lots of time to do the procedure), we obtain the average accuracy,variance accuracy and average runtime.


```
Average accuracy: 0.84696
```

### Polynomial kenel function

Choose a **polynomial kernel function** with coefficient=1,2,3,4,5 and degree=2,3,4,5 respectively.  Fitting on the training data with 10-fold cross-validation and running the loop 50 times to find the minimum of average accuracy. We can find the model has minimum mean accuracy with coefficient=4 and degree=3.


```
Average accuracy: 0.79393
```

| | Linear SVM | RBF SVM | Polynomial SVM |
|---|---|---|---|
| **Average Accuracy** | 0.52275 | 0.84696 | 0.79393 |

| | | | |
|---|---|---|---|
| **Vainance Accuracy** | 4e-04 | 7e-05 | 3e-04 |
| **Average Runtime** | 42.8 | 273.764 | 52.812 |

Based on the table shown above, we can find **RBF SVM has the highest average accuracy and highest average runtime**. We tried to use the Grid search method in R to find optimal parameters in the region [100,1000]x[0.1,1.0], but it is not easy to find optimal parameters in Linear SVM and Polynomial SVM (the optimizer did not converge in the same region). Hence, we choose particular parameters instead.

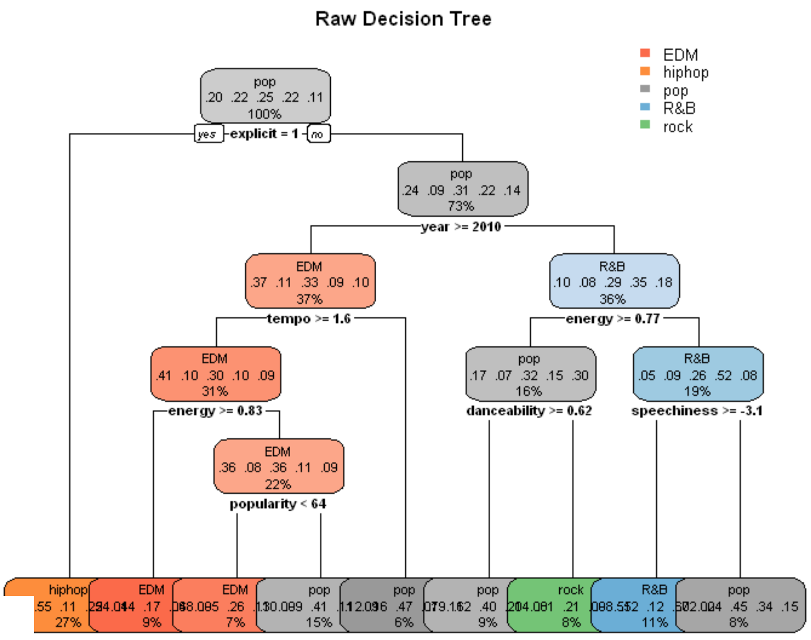
The variables are nonlinear, so the accuracy is not good in linear SVM. However, if we use nonlinear SVM methods, the accuracy has improved. In addition, the RBF kernel function with the Grid search method has slightly higher accuracy.

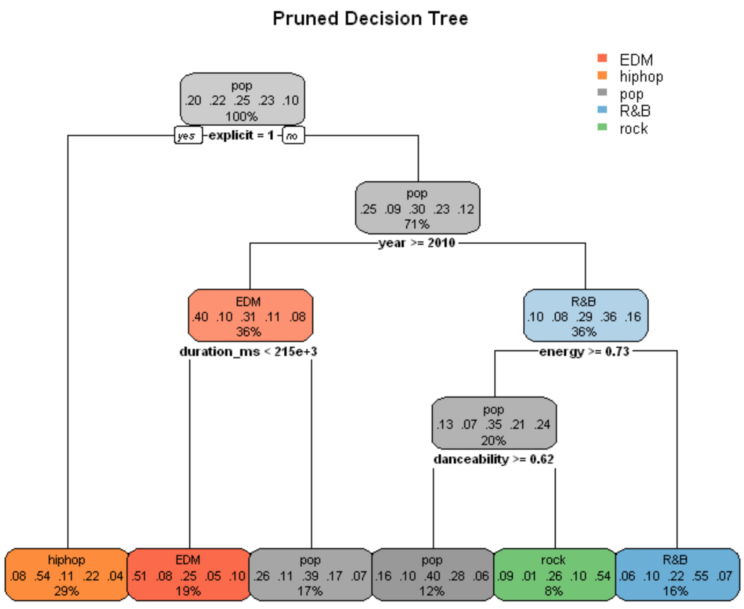
## Tree

### Decision Tree

We choose the Gini index as the impurity assessment. And use 10-fold cross-validation to calculate the overall "true error rate". and set each node to have at least 20 obs and first set the complexity parameter to 0.01.

A base decision tree with size = 9.




Pruned Decision Tree
EDM
hiphop
pop
R&B
rock
pop .20 .22 .25 .23 .10 100%
yes explicit = 1 no
pop .25 .09 .30 .23 .12 71%
year >= 2010
EDM .40 .10 .31 .11 .08 36%
duration_ms < 215e+3
R&B .10 .08 .29 .36 .16 36%
energy >= 0.73
pop .13 .07 .35 .21 .24 20%
danceability >= 0.62
hiphop .08 .54 .11 .22 .04 29%
EDM .51 .08 .25 .05 .10 19%
pop .26 .11 .39 .17 .07 17%
pop .16 .10 .40 .28 .06 12%
rock .09 .01 .26 .10 .54 8%
R&B .06 .10 .22 .55 .07 16%

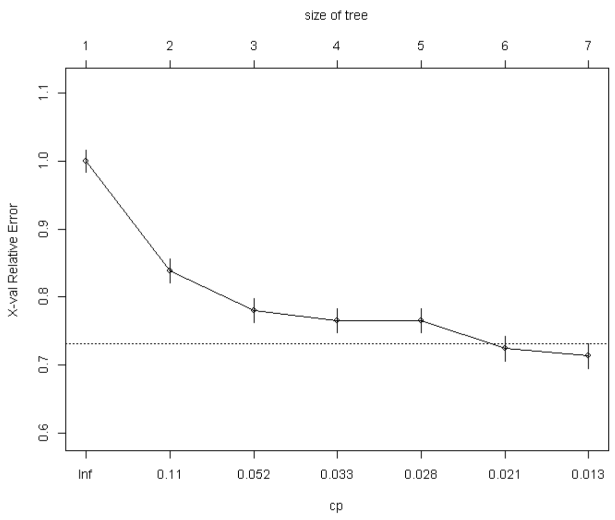
### Pruned

We decide the best tree size by the 1-SE rule. After pruning, the number of nodes has decreased, which is also in line with the purpose of pruning. In order to reduce the complexity of the model and solve overfitting.

A pruned decision tree with size = 6.

## Random Forest

We choose each parameter of **"mtry"** with a loop to find the point that minimizes the error.Then we plot a graph of the relationship between the model error and the number of decision trees, and find the best parameter **ntree**, and found that when there are about 6000 trees, it will tend to convergence



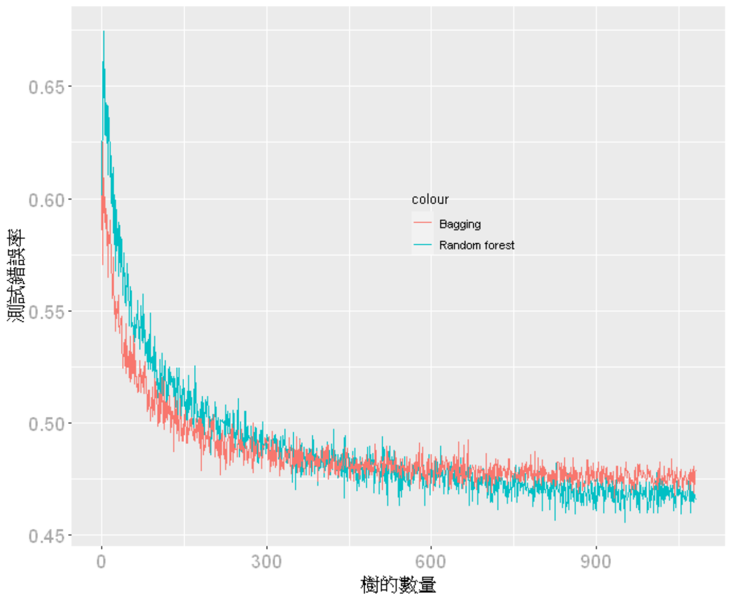
## Bagging

Bagging is the case where mtry is set equal to all explanatory variables. We found from the figure that the convergence of Bagging is similar to that of Random forest.

## RF vs. Bagging

The plot shows the convergence rate and efficiency of Random Forest and Bagging within 1000 trees.

It can be found that Bagging has a lower error rate when there are **0~300** trees at the beginning. When the number of trees is about **400~700**, the two intersect, and the error rate is similar. Bagging takes the lead **after 700**, but the random forest has a lower error rate, which is a higher prediction rate.



## Short Summary

| | Base Tree | Pruned Tree | Random Forest | Bagging |
|---|---|---|---|---|
| **Average Accuracy** | 0.46508 | 0.46428 | 0.54761 | 0.53596 |
| **Variance Accuracy** | 0.00036 | 0.00031 | 0.00032 | 0.00034 |
| **Average Runtime** | 0.08102 | 0.00466 | 8.79148 | 14.66352 |

As can be seen from this table, the highest prediction rate will be **Random Forest** (54.8%), and its running time is also shorter than Bagging.

The lowest running time is **Pruned Tree** (0.00466sec), Although the accuracy drops a little after pruning, pruning can avoid overfitting.

## XGBoosting

XGBoost is an optimized distributed gradient boosting designed to be highly efficient, flexible and portable. XGBoost provides a parallel linear and tree boosting. Therefore, for this study, we will compare these two methods.
We also apply learning tunes to help us to determine the best hyperparameters for both methods' models.

The base model for xgboosting, I choose the linear boosting with no tuning parameter.

### Linear Boosting

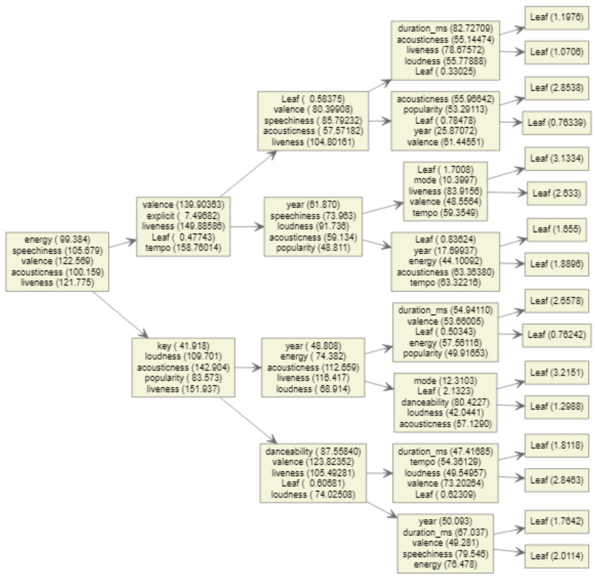
Use mlr package to determine the hyperparameters which determined by grid search:

1) **eta** ( learning rate, robustness by shrinking the weights on each step )
   range [0.01 , 0.5] = **0.493**
2) **alpha** ( L1 regularization term on weight to used of very high dimensionality )
   range [0 , 1] = **0.758**
3) **lambda** ( L2 regularization term on weights to handle the regularization )
   range [0 , 1] = **0.0599**
4) **lambda_bias** ( L2 regularization term on weights to handle the regularization )
   range [0 , 1] = **0.355**

### Tree Boosting

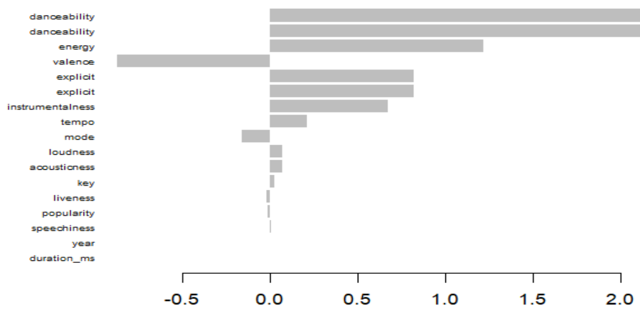
1) **eta** range [0.01 , 0.5] = **0.0634**
2) **max_depth** ( The maximum depth of a tree to control over-fitting as higher depth )
   range [3L , 10L] = **4**
3) **min_child_weight** ( the min sum of weights of all observations required in a child )
   range [1 , 10] = **1.92**
4) **subsample** ( the fraction of observations to be randomly samples for each tree )
   range [0.5 , 1] = **0.759**
5) **colsample_bytree** ( the fraction of columns to be randomly samples for each tree )
   range [0,5 ,1] = **0.949**

| | **Base** | **gblinear** | **gbtree** |
|---|---|---|---|
| **Average Accuracy** | 0.45693 | 0.45890 | 0.52411 |
| **Variance Accuracy** | 0.00031 | 0.00032 | 0.00036 |
| **Average Runtime** | 0.5007 | 0.4961 | 1.802 |


energy ( 99.384) speechiness (105.679) valence (122.569) acousticness (100.159) liveness (121.775)
valence (139.90363) explicit ( 7.49682) liveness (149.88586) Leaf ( 0.47743) tempo (158.76014)
key ( 41.918) loudness (109.701) acousticness (142.904) popularity ( 83.573) liveness (151.937)
Leaf ( 0.58375) valence ( 80.39908) speechiness ( 85.79232) acousticness ( 57.57182) liveness (104.80161)
year (61.870) speechiness (73.963) loudness (91.736) acousticness (59.134) popularity (48.811)
year ( 48.808) energy ( 74.382) acousticness (112.659) liveness (116.417) loudness ( 68.914)
danceability ( 87.55840) valence (123.82352) liveness (105.49281) Leaf ( 0.60681) loudness ( 74.02508)
duration_ms (82.72709) acousticness (55.14474) liveness (78.67572) loudness (55.77888) Leaf ( 0.33025)
acousticness (55.96642) popularity (53.29113) Leaf ( 0.78478) year (25.87072) valence (61.44551)
Leaf ( 1.7008) mode (10.3997) liveness (83.9156) valence (48.5564) tempo (59.3549)
Leaf ( 0.83624) year (17.69937) energy (44.10092) acousticness (63.36380) tempo (63.32216)
duration_ms (54.94110) valence (53.66005) Leaf ( 0.50343) energy (57.56116) popularity (49.91653)
mode (12.3103) Leaf ( 2.1323) danceability (80.4227) loudness (42.0441) acousticness (57.1290)
duration_ms (47.41685) tempo (54.36129) loudness (49.54957) valence (73.20264) Leaf ( 0.62309)
year (50.093) duration_ms (67.037) valence (49.281) speechiness (79.546) energy (76.478)
Leaf (1.1976)
Leaf (1.0706)
Leaf (2.8538)
Leaf (0.76339)
Leaf (3.1334)
Leaf (2.633)
Leaf (1.655)
Leaf (1.8896)
Leaf (2.6578)
Leaf (0.76242)
Leaf (3.2151)
Leaf (1.2988)
Leaf (1.8118)
Leaf (2.8463)
Leaf (1.7642)
Leaf (2.0114)

At the end, we will select xgboost tree as our final model; even though, the running time of gbtree is 3 times of the other 2 methods. but accuracy has improved from 0.46 to 0.52. The graph at the left is the result of the tree graph. As we can see, the depth of the tree is 4.

Also the bar chart shows the importance of variables. Danceability seems like the most important variable to this model.



## Conclusion

### EDA

We do outlier analysis and data transformation to **improve classification** results and make variables close to **non-skewed**.
According to time analysis, we know that R&B and rock pop music was popular in the early 2000s, and EDM music was popular in the near years.

In PCA, we can find that PCA1 tends to emotional scales such as Key, speechiness and acouticness and hence EDM and Rock tend to have a positive loading. Furthermore, we can find 5 different genres separated by 2D and 3D plots.

In MCA, we can find that Hiphop music tends to explicit lyrix (sexual or violence lyrix).

### Machine Learning

We use **NN, SVM, Random Forest, Tree, XGBoosting method** with seperate training data and testing data and Cross-Validation to build the model to classify the genres according to the music qualities.

1) The **NN model** with Minkowski distance and the Gaussian kernel method has higher accuracy about 46.845%.

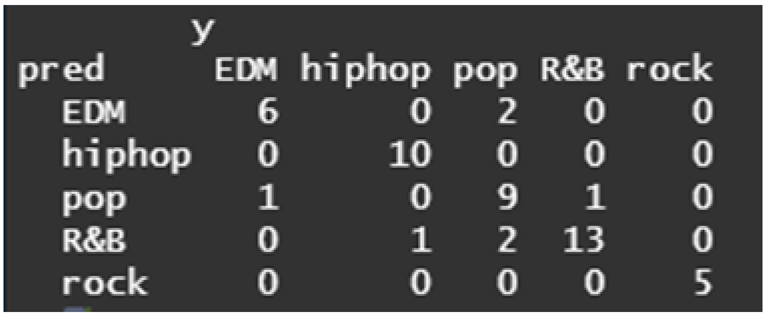
2) The **kernel SVM** model with cost=800 and gamma=0.1 has higher accuracy about 84.70%.

3) The **Random forest** with ntrees=6000 and mtry=1~15 has higher accuracy about 54.761%.

4) The **xgboost tree** with eta = 0.0634, max_depth = 0.758, min_child_weight = 1.92, subsample = 0.759, colsample_bytree = 0.949 has higher accuracy about 52.411%.

Hence, we choose the **kernel SVM model with cost=800 and gamma=0.1** as our final model.

Now we use the best accuracy model with cost=800 and gamma=0.1 to fit 50 samples from the whole data. The accuracy is 86%.
(very close to the average accuracy in the RBF method)



```
        y
pred     EDM hiphop pop R&B rock
  EDM      6      0   2   0    0
  hiphop   0     10   0   0    0
  pop      1      0   9   1    0
  R&B      0      1   2  13    0
  rock     0      0   0   0    5
```

| | Kernel-KNN | Random Forest | RBF SVM | gbtree |
|---|---|---|---|---|
| **Average Accuracy** | 0.46845 | 0.54761 | 0.84696 | 0.52411 |
| **Variance Accuracy** | 0.00034 | 0.00032 | 0.00007 | 0.00036 |
| **Average Runtime** | 0.22148 | 8.79148 | 273.764 | 1.802 |
| **Parameter Setting** | distance=1<br>kernel="gaussian" | ntree = 6000<br>mtry = min(error) | cost=800<br>gamma=0.1 | eta = 0.0634<br>max_depth = 0.758<br>min_child_weight = 1.92<br>subsample = 0.759<br>colsample_bytree = 0.949 |

## Future Work

For our study, we still have many problems haven't solve, like, "More scientific way to cluster the genre", "Song Artist and name's analysis and used in model ", "The result boundary graph for all the models", and "More runs for different building model process of different train and test dataset". We also want to do deeper research on why SVM has better results than other methods.

## References

### #KNN
https://cran.r-project.org/web/packages/kknn/kknn.pdf
https://www.kaggle.com/code/sandhyakrishnan02/knn-svm-svm-with-kernel-hyperparameter
https://cran.r-project.org/web/packages/KernelKnn/KernelKnn.pdf
https://towardsdatascience.com/make-your-knn-smooth-with-gaussian-kernel-7673fceb26b9
https://zhuanlan.zhihu.com/p/76376573

### #SVM
https://www.jianshu.com/p/446fae533b17
https://rpubs.com/markloessi/506713

### #XGBoosting
https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/
https://cran.r-project.org/web/packages/xgboost/xgboost.pdf

### #Tree
https://www.796t.com/content/1549855474.html
https://www.796t.com/content/1542572103.html
https://rpubs.com/skydome20/R-Note16-Ensemble_Learning