Machine Learning

Professor: Dr. Raman Kannan

Assignment : Exploratory Data Analysis

Topic : Music Genre Prediction

Aakash Shetty
aks9108

Overview

I. A Quick Glance

This is a quick overview of the EDA. The dataset is music genre prediction. The main focus is audio features that will help us classify the music genre. Now that we know what we are about to do in the EDA, we can begin with the steps. Our first step would be to load the data. Once we load the data we would like to preprocess the data to remove null values, empty rows and do some important changes to the data which can help us down the line for analytics like encodings and type conversions. Know that we have a workable data with us we can begin with exploration. We move in the direction of audio features which is given to us and try to find the density of songs at a particular feature value. It is important to make sure we have something to work with. We also delve into popularity of a genre and time it was added to database to find any important patterns that can be used. We also remove the outliers in this step.

Once we find that the audio features distribution we started with finding correlations between audio features to avoid any redundancy in the features being used. We could see great similarity in features like energy and loudness. After removing redundancy and cleaning the data we start with modeling. We prepare the data for training and testing (75-25) with the audio features and start working on decision tree, random forest and xgboost. We can see the accuracy of the overall model as well as the accuracy for each genre given by a particular model. Lastly, we talk about variable importance as every model works in a different way and gives importance to different features in uniquely. We map the features and importance value on y and x axis to see the importance given by each model to each feature.

II. Getting to Know the Dataset

The dataset is a Music Prediction dataset taken from Kaggle. It gives a list of music feature like tempo, key, energy etc all independent variables including numeric and character values. to help predict the music genre of a particular song. In total we have been given *17 columns* to work with and *1 column* for the genre. We have a total of *50,000 observations*, making it a considerably large dataset. The genres in the dataset range from *Alternative* to *Rock,* in total it has *10 genres,* so we have a multi-class classification problem. Each genre has *5000 songs*.

While it would be fun to work with more features, it would be best to select the best and most crucial features for genre prediction. Through the EDA it will be clear as to which features have been selected and why it is more important that the other features. The dataset is saved as a csv file and loaded from the same csv file.

```
Rows: 50,005
Columns: 18
$ instance_id      <dbl> 32894, 46652, 30097, 62177, 24907, 89064, 43760, 3073…
$ artist_name      <chr> "Röyksopp", "Thievery Corporation", "Dillon Francis",…
$ track_name       <chr> "Röyksopp's Night Out", "The Shining Path", "Hurrican…
$ popularity       <dbl> 27, 31, 28, 34, 32, 47, 46, 43, 39, 22, 30, 27, 31, 3…
$ acousticness     <dbl> 4.68e-03, 1.27e-02, 3.06e-03, 2.54e-02, 4.65e-03, 5.2…
$ danceability     <dbl> 0.652, 0.622, 0.620, 0.774, 0.638, 0.755, 0.572, 0.80…
$ duration_ms      <dbl> -1, 218293, 215613, 166875, 222369, 519468, 214408, 4…
$ energy           <dbl> 0.941, 0.890, 0.755, 0.700, 0.587, 0.731, 0.803, 0.70…
$ instrumentalness <dbl> 7.92e-01, 9.50e-01, 1.18e-02, 2.53e-03, 9.09e-01, 8.5…
$ key              <chr> "A#", "D", "G#", "C#", "F#", "D", "B", "G", "F", "A",…
$ liveness         <dbl> 0.1150, 0.1240, 0.5340, 0.1570, 0.1570, 0.2160, 0.106…
$ loudness         <dbl> -5.201, -7.043, -4.617, -4.498, -6.266, -10.517, -4.2…
$ mode             <chr> "Minor", "Minor", "Major", "Major", "Major", "Minor",…
$ speechiness      <dbl> 0.0748, 0.0300, 0.0345, 0.2390, 0.0413, 0.0412, 0.351…
$ tempo            <chr> "100.889", "115.00200000000001", "127.994", "128.014"…
$ obtained_date    <chr> "4-Apr", "4-Apr", "4-Apr", "4-Apr", "4-Apr", "4-Apr",…
$ valence          <dbl> 0.7590, 0.5310, 0.3330, 0.2700, 0.3230, 0.6140, 0.230…
$ music_genre      <chr> "Electronic", "Electronic", "Electronic", "Electronic…
```

III. Working with Dataset

We have to load the csv file into a variable songs.

```
songs <- read.csv('dataset/music_genre.csv', stringsAsFactors = FALSE)
```

Next, we check to see if the loaded data is correct and as we expect. head(…,10) will display the first 10 rows from our table and the column names with the type.

```
head(songs, 10)
```

A data.frame: 10 × 18

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	spee
	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	
1	32894	Röyksopp	Röyksopp's Night Out	27	0.00468	0.652	-1	0.941	7.92e-01	A#	0.1150	-5.201	Minor	
2	46652	Thievery Corporation	The Shining Path	31	0.01270	0.622	218293	0.890	9.50e-01	D	0.1240	-7.043	Minor	
3	30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	1.18e-02	G#	0.5340	-4.617	Major	
4	62177	Dubloadz	Nitro	34	0.02540	0.774	166875	0.700	2.53e-03	C#	0.1570	-4.498	Major	
5	24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	9.09e-01	F#	0.1570	-6.266	Major	
6	89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	8.54e-01	D	0.2160	-10.517	Minor	
7	43760	Jordan Comolli	Clash	46	0.02890	0.572	214408	0.803	7.74e-06	B	0.1060	-4.294	Major	
8	30738	Hraach	Delirio	43	0.02970	0.809	416132	0.706	9.03e-01	G	0.0635	-9.339	Minor	
9	84950	Kayzo	NEVER ALONE	39	0.00299	0.509	292800	0.921	2.76e-04	F	0.1780	-3.175	Minor	
10	56950	Shlump	Lazer Beam	22	0.00934	0.578	204800	0.731	1.12e-02	A	0.1110	-7.091	Minor	

We find out that some rows in the data have null values and can cause issues while performing further preprocessing steps or visualization. We can either choose to ignore null values while processing or remove the entire row with null value in any column from the table. I choose to remove the entire row because it would not affect the overall accuracy due to large number of observations still available to us. Removing few rows would be better to avoid any further null values problems.

```
songs <- na.omit(songs)
```

```
songs %>%
  count(music_genre) %>%
  knitr::kable()
```

music_genre	n
Alternative	4495
Anime	4497
Blues	4470
Classical	4500
Country	4486
Electronic	4466
Hip-Hop	4520
Jazz	4521
Rap	4504
Rock	4561

We now have table withouts nulls and can move on to further process the data by encoding character to numbers for better consumption of data and better correlation understanding. As your can see the while we run the head command we can see feature *'KEY'* have values in the form 'A#', 'C', 'C#' etc. and *'MODE'* have values in the form *'Minor'* and *'Major'*. We encode the columns *'KEY'* and *'MODE'*.

```
unique(songs$key)
```

```
'A#'  'D'  'G#'  'C#'  'F#'  'B'  'G'  'F'  'A'  'C'  'E'  'D#'
```

```
songs$mode <- as.numeric(gsub("Minor", 1, gsub("Major", 0, songs$mode)))
songs$key <- as.numeric(factor(songs$key))
```

```
unique(songs$key)
```

```
 2 ·  6 ·  12 ·  5 ·  10 ·  3 ·  11 ·  9 ·  1 ·  4 ·  8 ·  7
```

We now have data with Minor = 1 and Major = 2. Also, keys have values repressing each of them uniquely.

IV. Chi-square test of independence

We now move to complete the chi-square test for the attributes in the table. Chi-Square test in R is a statistical method which used to determine if two categorical variables have a significant correlation between them. We do this step as a precursor to the following steps in the EDA to see if we are considering the correct attributes to deal with. We can have a threshold of 0.05 for p-value to check interdependence between out audio features and genres. This chi-square test is run for all our audio feature attributes and popularity. While I also try running it for obvious non related attribute such as instance_id, it can be seen that attributes such as danceability, energy etc. have p-value considerably less than 0.5 while attribute such as instance_id have p-value greater than 0.05.

```
chisq.test(songs$music_genre, songs$popularity, correct=FALSE)
```

```
        Pearson's Chi-squared test

data:  songs$music_genre and songs$popularity
X-squared = 17952, df = 9, p-value < 2.2e-16
```

```
chisq.test(songs$music_genre, songs$key, correct=FALSE)
```

```
        Pearson's Chi-squared test

data:  songs$music_genre and songs$key
X-squared = 2195.1, df = 99, p-value < 2.2e-16
```

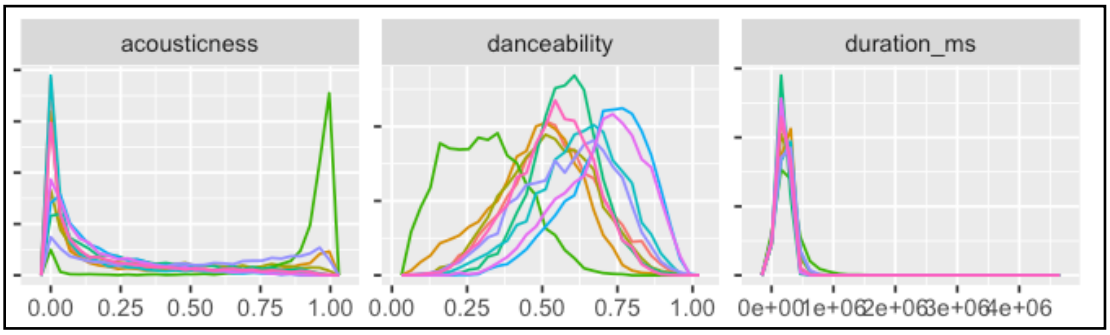
```
        Pearson's Chi-squared test

data:  songs$music_genre and songs$instance_id
X-squared = 405180, df = 405171, p-value = 0.4957
```

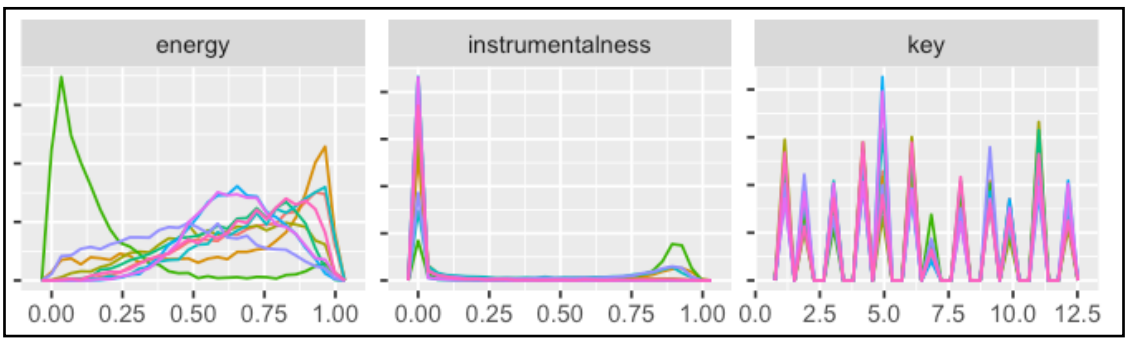
V. Exploration

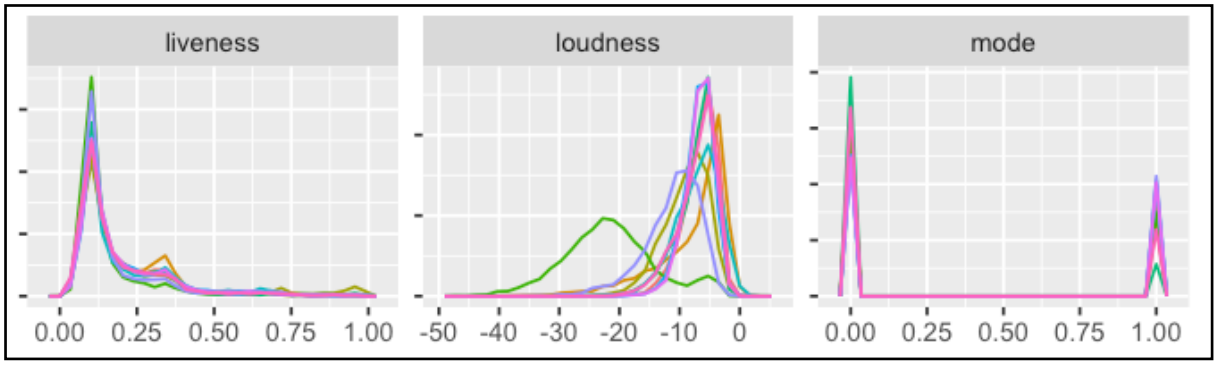
Now, we it's time to find out more about the data we have in hand. Looking at tables is not the best way to know the data and its distribution. One way of better understanding the data is visualizing it in form of graphs and charts.

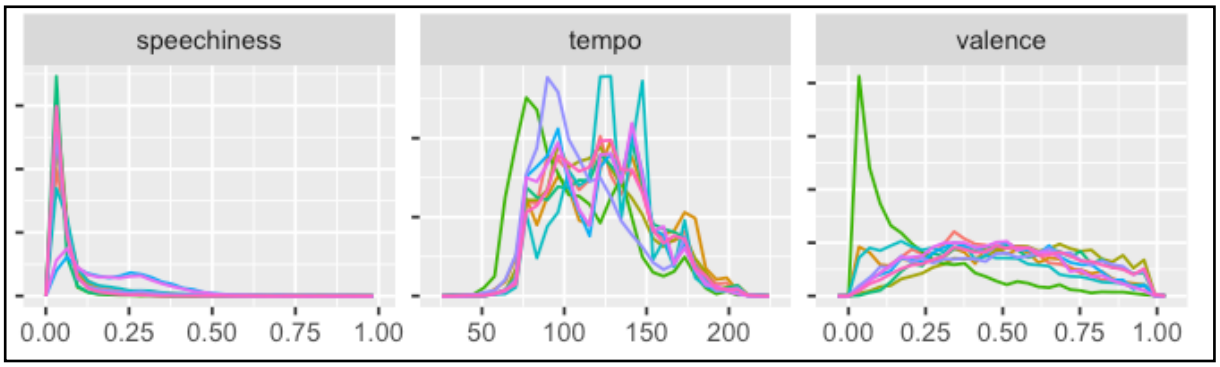
We try to display ***feature density by genre*** and see the density distribution of each audio feature with respect to each genre.

```
library(ggplot2)
library(tidyr)
feats <- names(songs)[c(5:15,17)]

img_songs <- songs %>%
  select(c('music_genre', feats)) %>%
  pivot_longer(cols = feats)

img_songs %>%
  ggplot(aes(x = value)) + geom_freqpoly(aes(color = music_genre)) + facet_wrap(~name, ncol = 3, scales = 'free') +
  labs(title = 'Audio Feature Pattern', x = '', y = '') + theme(axis.text.y = element_blank())
```







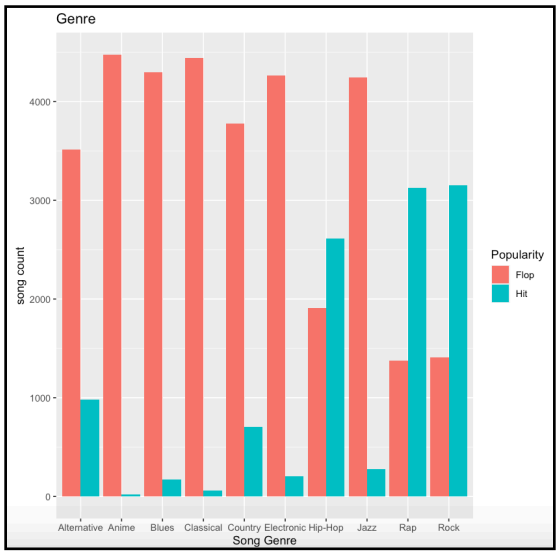


Here, we have the list of charts with different audio features and its dusty in y-axis and its values in the x-axis. The different color represent different genres. Thus, helping us understand the data better between different genres. We can see the peak of values of each genre meaning that a genre has more of that particular attribute at that value. This can be helpful while trying to classify the genres. We can take into consideration the concentrations of songs at a particular value a factor in deciding the genre.

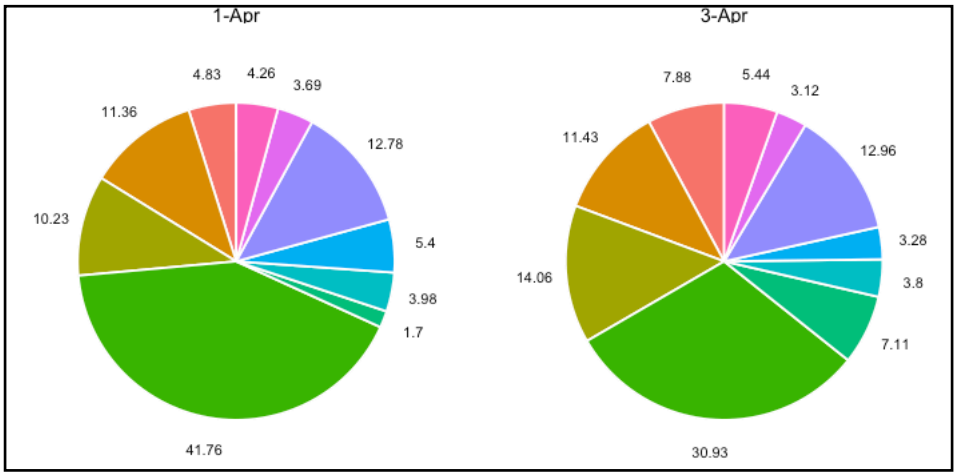
Next, we move to popularity of a song. This may not be an audio feature but we can check to see if we can find a pattern in the popularity and the music genres. We have a column in the table for popularity and has a very diverse range of values ranging from 0 - 99. The problem we face is to check if a songs was well received or not by the listeners. We can make the range from diverse to binary by using thresholding. Finding relation between ***popularity and genre***.

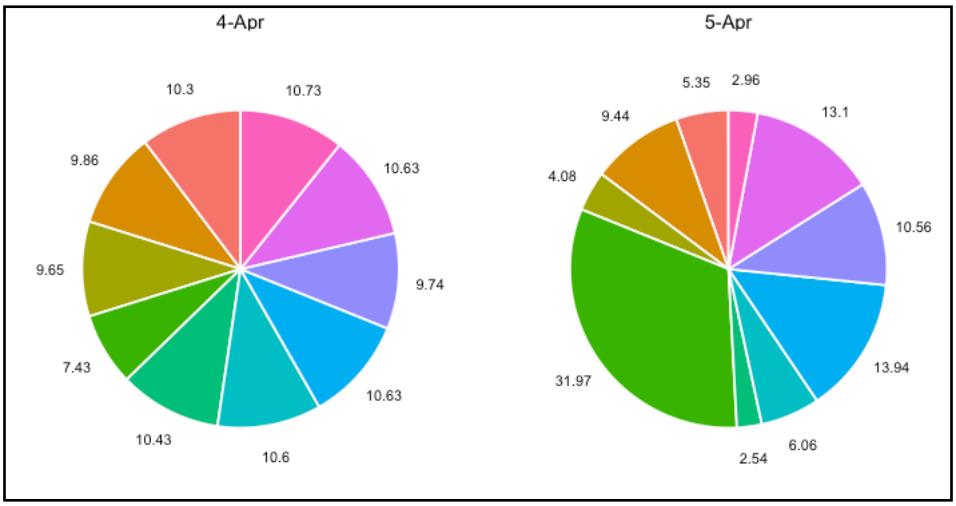
```
songs$popularity<-ifelse(songs$popularity>55,1,0)
```

```
library(tidyverse)
music_tibble <- as_tibble(songs)

genre_count_df <- music_tibble %>%
group_by(music_genre, popularity) %>%
summarise(counts = n())

genre_hit_count_total <- genre_count_df %>%
group_by(music_genre, popularity) %>%
summarise(counts = sum(counts))

genre_hit_count_total %>%
group_by(music_genre, popularity) %>%
summarize(counts = sum(counts)) %>%
  ggplot(aes(x=music_genre, y=counts, fill=as.factor(popularity)))+
  geom_bar(stat="identity", position="dodge")+labs(title="Genre", x="Song Genre", y="song count")+
  scale_fill_discrete(name = "Popularity", labels = c("Flop", "Hit"))
```



Percentage of Genre among hit songs based on date it was obtained



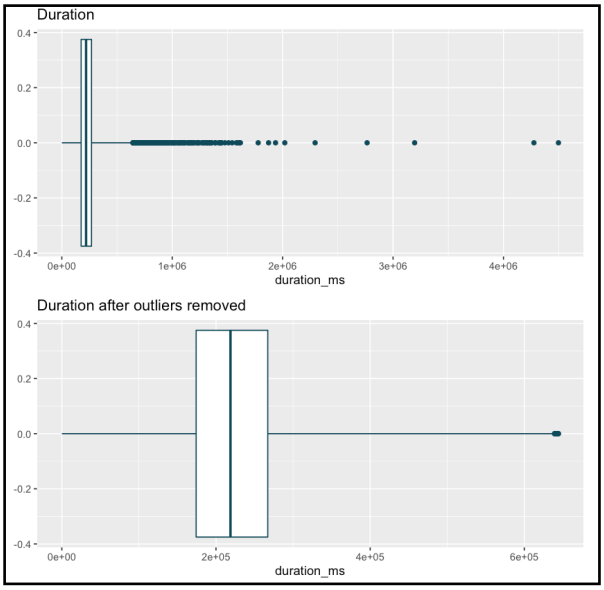


The disparity is very high among the genres, with rock, rap and hip-hop being the only genres having more hits than flops and others barely having any hits in comparison to flops. While this chart can give us the trending genres it will be very difficult to use popularity in genre prediction as the number ratios of high popularity and low popularity genres are quite similar in their respective groups. Even looking at dates the songs were added gives us no clue as to which genre can be predicted based on date added as it is not the date when the song was first published.

Moving on we set out to remove outliers from the data, it is easy for outliers to skew analysis and its easy for any data to have outliers. Based on the range we can consider what is outlier in our data what is not. We try to use a wider net by giving range 4 as lower ranges removed more data than we were expecting. Meaning more values fall outside the IQR in our case. So we increase the range in consideration.

In total we remove 358 outliers from the data.

```r
songs_out <- songs %>%
  ggplot(aes(y = duration_ms)) + geom_boxplot(color = kp_cols('dark_blue'), coef = 4) +
  coord_flip() + labs(title = 'Duration')

duration_out <- boxplot(songs$duration_ms, plot = FALSE, range = 4)$out

songs_wo_out <- songs %>%
  filter(!duration_ms %in% duration_out)

wo_out <- songs_wo_out %>%
  ggplot(aes(y = duration_ms)) + geom_boxplot(color = kp_cols('dark_blue'), coef = 4) +
  coord_flip() + labs(title = 'Duration after outliers removed')

gridExtra::grid.arrange(songs_out, wo_out, ncol = 1)
```

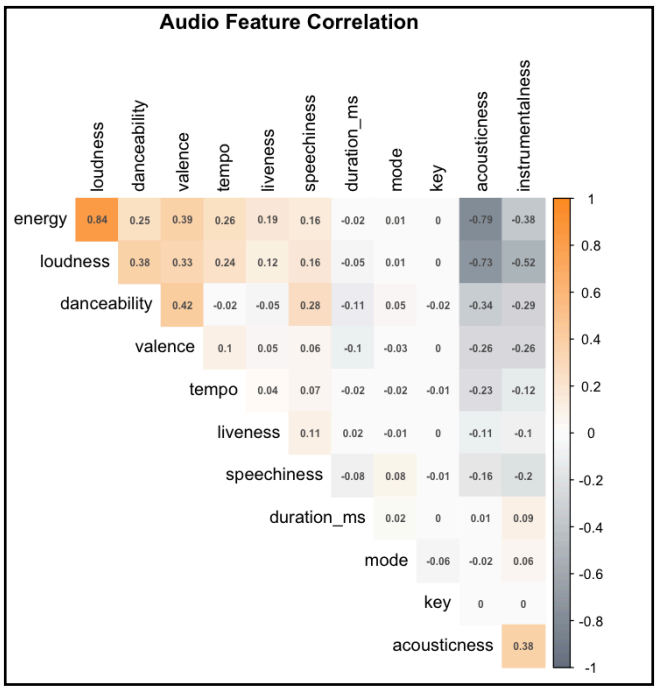


VI. Correlation

Now, that we are done with exploration and we understand what can be useful for us in genre prediction, we can move ahead with finding the correlation within the features. Are features too similar to each other? Will there be a redundancy problem? Will there be no correlation at all?

Let's visualize the correlation among the audio feature of the music. We plot each feature against each feature in the audio feature list and compare the similarity. The closer the similarity value to 1 the closer the the two features resemble.

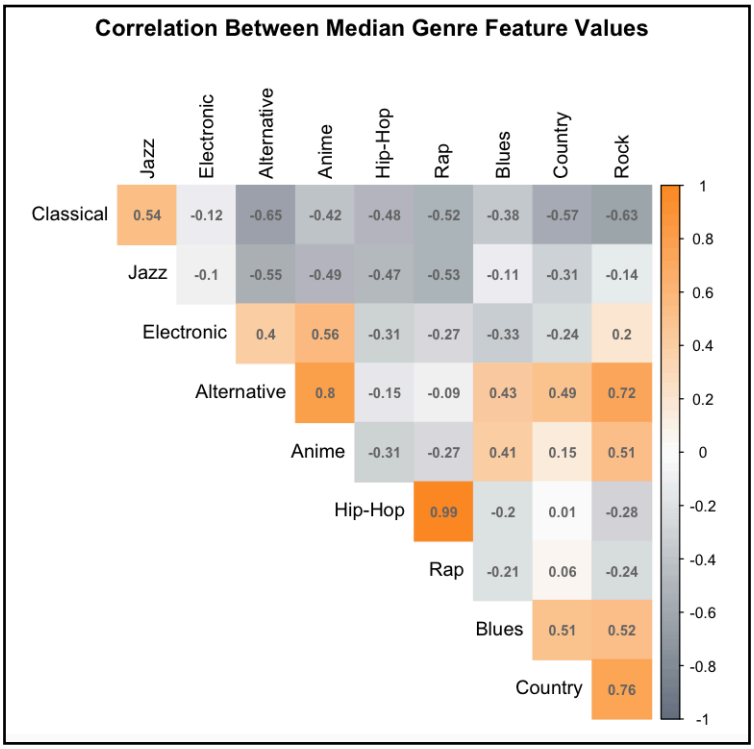
```
songs_wo_out %>%
  select(feats) %>%
  scale() %>%
  cor() %>%
    corrplot::corrplot(method = 'color', order = 'hclust', type = 'upper', diag = FALSE, tl.col = 'black',
    addCoef.col = "grey30", number.cex = 0.6, col = colorRampPalette(colors = c(kp_cols('grey'), 'white',
    kp_cols('orange')))(200), main = 'Audio Feature Correlation', mar = c(2,2,2,2))
```



As you can see in the correlation, the similarity plot makes lot of sense. Acousticness is negatively correlated to energy, loudness , valence as these all

features represent happy and more instruments being played and acousticness is the opposite. Energy and loudness is highly similar, which is understandable.

Similarly, we can have genre similarity, this doesn't take individual song variation into account, but will give us an idea which genres are similar to each other. For this we can calculate the median value for each feature values in each genre and then plot them to see similarity.



We can see some similarity between electronic and alternative. High correlation between Hip-Hop and rap, country and rock. While we can see some understandable differences in Rap and classical, Jazz and rap and Classical and rock.

VII. Data Preparation for Training

For Data preparations, we can split the data into 75:25 ratio for training and testing respectively.

```
train_songs <- songs_wo_out %>%
  mutate_if(is.numeric, scale)

training_songs <- sample(1:nrow(train_songs), nrow(train_songs)*.75, replace = FALSE)
train_set <- train_songs[training_songs, c('music_genre', feats)]
test_set <- train_songs[-training_songs, c('music_genre', feats)]

train_resp <- train_songs[training_songs, 'music_genre']
test_resp <- train_songs[-training_songs, 'music_genre']
```

Further, we will be discussing about three models in EDA for the purpose of Variable importance. This is to show that each of the model is not the same even when given same data and even when the exploration is the same.

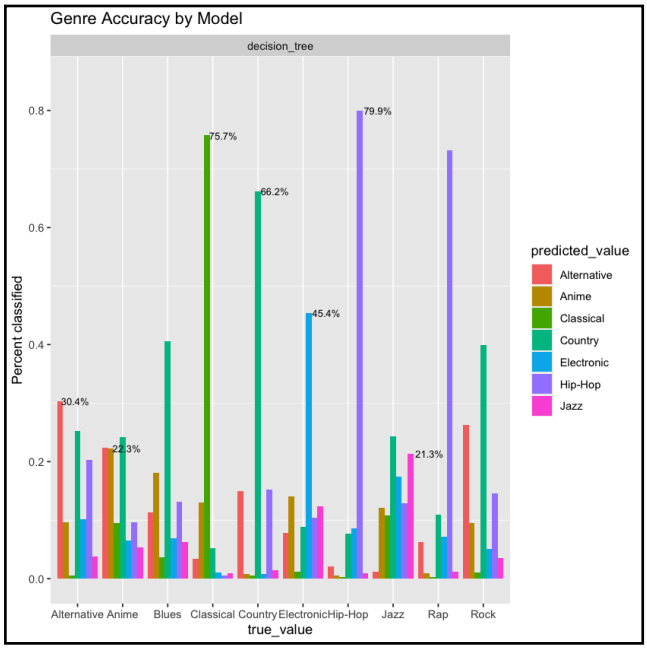
All the three model is just to show that the data explored and cleaned is workable with and can be used for real classification purposes. Thus we won't be diving into detail of modeling for this homework.

VIII. Modeling - Decision Tree

Decision tree provides a lot of information about the data. While it can be prone to outliers it is very reliable model as it does not assume anything about the data. A decision tree starts with dividing the genres based on features given. It will decide the feature most important for segmentation and keeps going down until dividing all the genres on the basis of features.

```
decision_tree <- rpart(music_genre ~ ., data = train_set)

predict_dt <- predict(object = decision_tree, newdata = test_set)
max_id <- apply(predict_dt, 1, which.max)
pred <- levels(as.factor(test_set$music_genre))[max_id]

compare_dt <- data.frame(true_value = test_set$music_genre, predicted_value = pred, model = 'decision_tree',
                         stringsAsFactors = FALSE)

model_accuracy_calc <- function(df, model_name) {
  df %>%
    mutate(match = ifelse(true_value == predicted_value, TRUE, FALSE)) %>%
    count(match) %>%
    mutate(accuracy = n/sum(n),model = model_name)
}

accuracy_dt <- model_accuracy_calc(df = compare_dt, model_name = 'decision_tree')
accuracy_dt
```

match	n	accuracy	model
<lgl>	<int>	<dbl>	<chr>
FALSE	5921	0.6628232	decision_tree
TRUE	3012	0.3371768	decision_tree



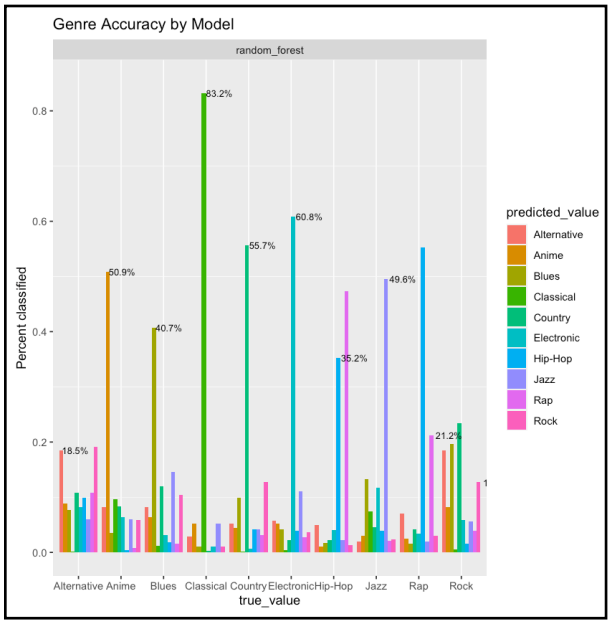
With this we get values for the false classifications and the correct classifications made by the model from the given data. We have a 34% success rate with decision tree.

IX. Modeling - Random forest

Random forest is an extension of Bagging, which it's an aggregation of trees. By bootstrap (sample with replacement) from the data and fit tree models, random forest aggregates those trees and makes significant improvement in terms of prediction. The idea of random forests is to randomly select m out of p predictors as candidate variables for each split in each tree. Commonly, m=√p in classification. The reason of doing this is that it can de-correlates the trees such that it reduces variance when we aggregate the trees. Let's fit this model on the training data and make predictions on both training and testing data.

```
library(randomForest)
model_rf <- randomForest(as.factor(music_genre) ~ ., ntree = 100, importance = TRUE, data = train_set)

predict_rf <- predict(model_rf, test_set)

compare_rf <- data.frame(true_value = test_resp,
                         predicted_value = predict_rf,
                         model = 'random_forest',
                         stringsAsFactors = FALSE)

accuracy_rf <- model_accuracy_calc(df = compare_rf, model_name = 'random_forest')
accuracy_rf
```

match	n	accuracy	model
<lgl>	<int>	<dbl>	<chr>
FALSE	6408	0.573885	random_forest
TRUE	4758	0.426115	random_forest



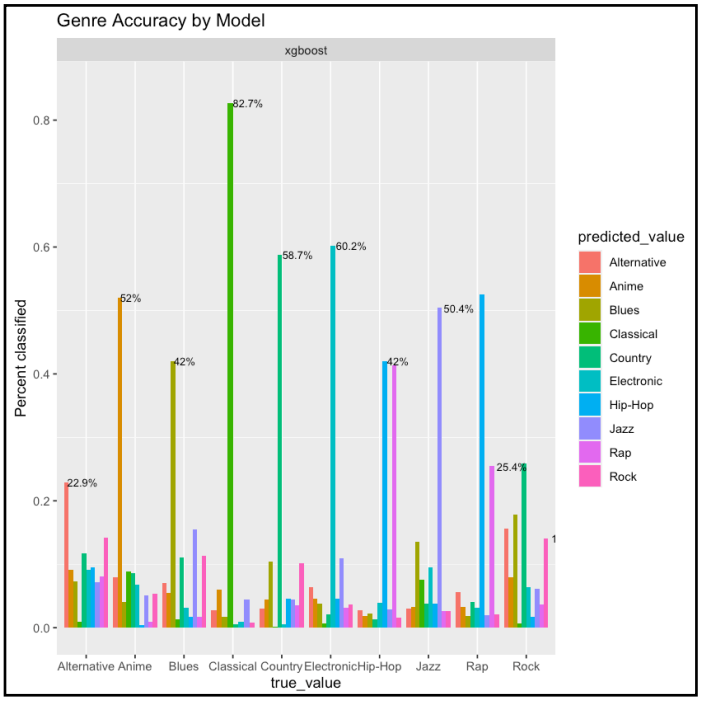
We have got an accuracy of 43% from random forest and the accuracy of the model every genre.

X. Modeling - XGBoost

Adding in the gradient descent algorithm for minimizing errors results in a gradient boosting model. Here, we'll use XGBoost, which provides parallel processing to decrease compute time as well as various other improvements. We'll use the xgboost function with most of the default hyper-parameter settings, just setting objective to handle multi-class classification.

```
library(xgboost)
matrix_train_gb <- xgb.DMatrix(data = as.matrix(train_set[,-1]), label = as.integer(as.factor(train_set[,1])))
matrix_test_gb <- xgb.DMatrix(data = as.matrix(test_set[,-1]), label = as.integer(as.factor(test_set[,1])))

model_gb <- xgboost(data = matrix_train_gb,
                    nrounds = 50,
                    verbose = FALSE,
                    params = list(objective = "multi:softmax",
                                  num_class = 10 + 1))

predict_gb <- predict(model_gb, matrix_test_gb)
predict_gb <- levels(as.factor(test_set$music_genre))[predict_gb]

compare_gb <- data.frame(true_value = test_resp,
                         predicted_value = predict_gb,
                         model = 'xgboost',
                         stringsAsFactors = FALSE)

accuracy_gb <- model_accuracy_calc(df = compare_gb, model_name = 'xgboost')
accuracy_gb
```

match	n	accuracy	model
<lgl>	<int>	<dbl>	<chr>
FALSE	6163	0.5519434	xgboost
TRUE	5003	0.4480566	xgboost

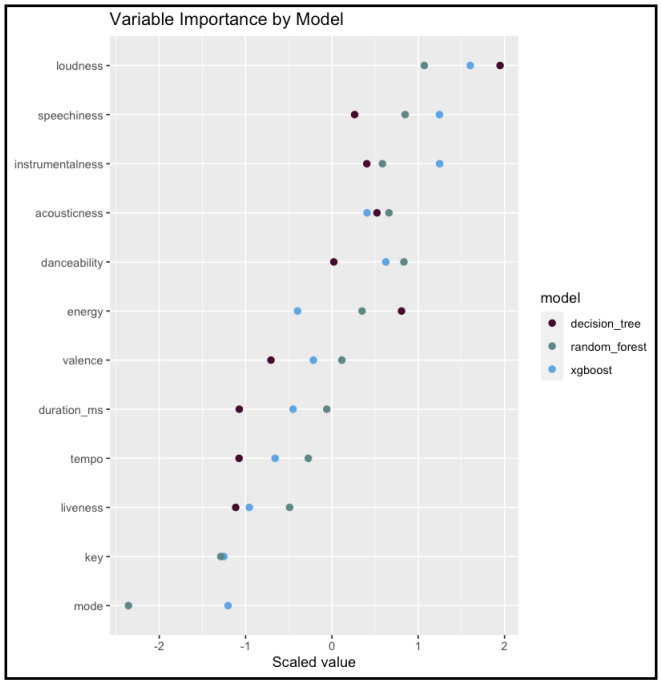


We have got an accuracy of 45% from XGBoost.

XI. Variable Importance

It is important to have a comparison between models as every model is different and takes a different approach towards variable and prediction. Decision trees provide a score for each feature based on its usefulness in splitting the data. For a random forest, we can use mean decrease in node impurity, which is the average decrease in node impurity/increase in node purity resulting from a split on a given feature. For XGBoost, we can use gain, or the improvement in accuracy contributed by a given feature.

```
imp_dt <- data.frame(imp = decision_tree$variable.imp)
imp_dt$feature <- row.names(imp_dt)

imp_rf <- data.frame(imp = imp(model_rf, type = 2))
imp_rf$feature <- row.names(imp_rf)

imp_gb <- xgb.imp(model = model_gb)

comp_imp <- imp_gb %>%
  select(Feature, Gain) %>%
  left_join(imp_dt, by = c('Feature' = 'feature')) %>%
  left_join(imp_rf, by = c('Feature' = 'feature')) %>%
  rename('xgboost' = 'Gain',
         'decision_tree' = 'imp',
         'random_forest' = 'MeanDecreaseGini')

comp_imp %>%
  mutate_if(is.numeric, scale, center = TRUE) %>%
  pivot_longer(cols = c('xgboost', 'decision_tree', 'random_forest')) %>%
  rename('model' = 'name') %>%
  ggplot(aes(x = reorder(Feature, value, mean, na.rm = TRUE), y = value, color = model)) + geom_point(size = 2) +
  coord_flip() + labs(title = 'Variable imp by Model',
                      y = 'Scaled value', x = '') + scale_color(palette = 'cool')
```



XII. Conclusion

Thus, we can predict music genres with audio features. Even though the accuracy is not the best, we have tried to work our way through the data and made changes to make the data able to work with. We can try to further improve the accuracy of models by incorporating some additional attributes provided in the table like artist names and track name. We can implement other techniques to check the accuracy of same and different models. Exploration has been a useful method to get insights and clues to get to know the data, the specific feature pattern of each genre, the relationship between audio features etc.

This EDA does give us some insights in audio feature pattern and the classification of genre, however, due to the limitation of data, model and time, the EDA is far from perfection. More data could be imported and more fancy models, like neural network could be built to extract more insights and informations from the data.