Technical Report—Echonest Song Database

***Team*** *– Ashish Khare, Prakhar Vashisht, Tejas Magar, Shubham Malpani, Vikram Rao, Devan Kumar*

# Abstract

Predictive modeling is the process of using known results to create, process, and validate a model that can be used to forecast future outcomes. This technique attempts to answer the question "what might happen in the future?"

For our analysis of the Songs dataset, we are using the decision tree algorithm to find insights through data.

Decision trees are a type of supervision learning algorithm that repeatedly splits the sample based on certain questions about the sample. Decision trees represent several decisions followed by different chances of occurrence. It can also be used in the data exploration stage. For example, we are working on a problem where we have information available in hundreds of variables, their decision tree will help to identify the most significant variable.

In this report we are trying to predict the hotness level of the songs with the help of the parameters that defines a song, for example, familiarity level of song, loudness of a song, danceability of the song, hotness of the artist who composes the song etc. We will try to find out which of the parameters play an important role to produce a hot song or per say, a song with the highest hotness level.

# Introduction

The Million Song Dataset (MSD), a collection of one million western popular music pieces, has enabled large-scale research for many MIR applications. The dataset comes with a set of features extracted by the API of The Echonest, which include tempo, loudness, timings of fade-in and fade-out, and MFCC-like features for several segments. These datasets can help to build genre classification, recommendation systems, and cover song recognition.

Right now, a company like Pandora relies on trained musicologists to build and expand its vast music database of characteristics that the company taps into to decide what to suggest for you next.

Also, The National Science Foundation undertaking is called The Listening Machine Project. Launched eight years ago, the Project's objective was analyzing "the individual sources present in a real-world sound recording" which machines are not quite adept at doing it.

As stated above, a good amount of research has undergone to analyze music datasets to build a more accurate recommendation system, and genre classification.

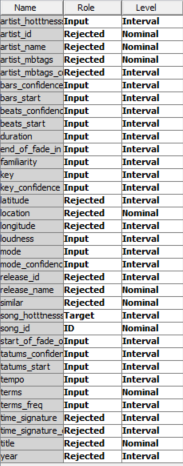
# Problem Statement

Based on some important data points related to a song which define the structure of the song like tempo, loudness, beats, fade-in time, fade-out time and other facts gathered from the Echonest database like artist hotness, danceability, and energy, we are going to predict the song hotness level of future songs.

# Dataset

We utilized a combination of different and smaller subsets of the initial 300GB total music dataset:

* The extracted size we used is 1.8GB
* 10,000 observations and 35 attributes
* Variable Role Definition:



* The target variable is ‘song\_hotttness’
* Rejected Variables: As clear from the diagram, some of the variables have been rejected from our analysis as they were not relevant for the problem statement we are trying to solve. For example: ‘artist\_name’ and ‘artist\_mb\_tags’ do not provide information related to a song, hence they have been rejected.

# Tools used

We have used SAS Enterprise Miner for creating the predictive model. Data was imported by adding a ‘File Import’ Node to the diagram and adding the dataset csv file to the node.

# Data Partitioning

Two data partitioning nodes were added to check how well the model gets trained in the 2 cases

1. Train: Validation: Test -> 60:30:10

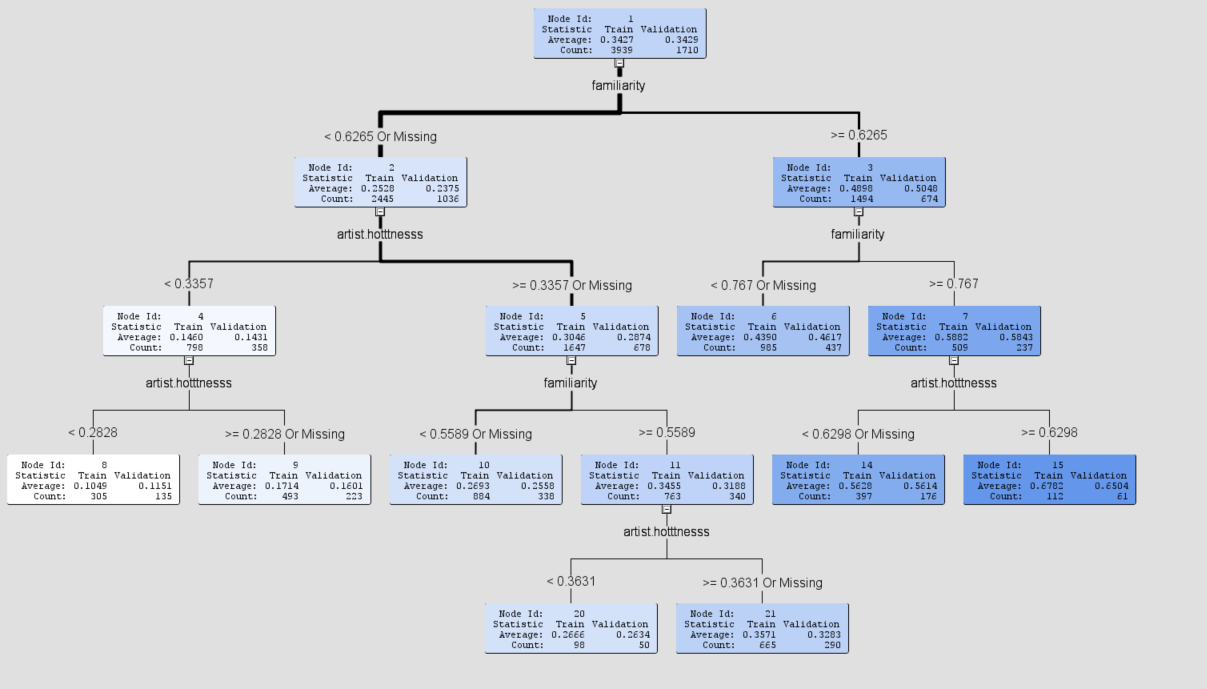
2. Train: Validation: Test -> 60:20:20

# Decision Tree

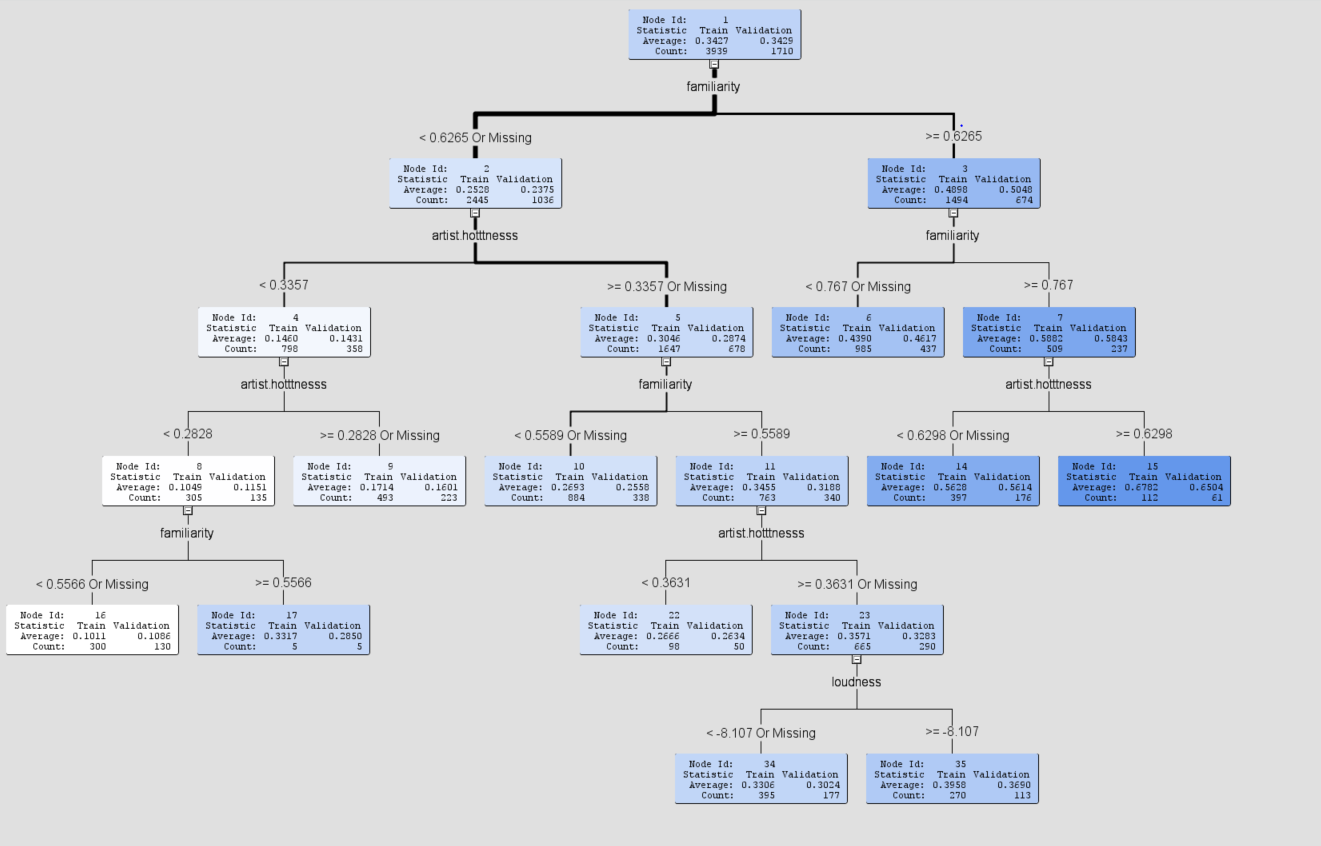
## **Step 1:**

The first step was to decide the best significance level for our decision trees(DT) which will be used for our further analysis. We took 2 different decision trees with significance factor=0.1 and significance factor=0.2 and then compared them. Here are the results:

*DT with significance level = 0.1*



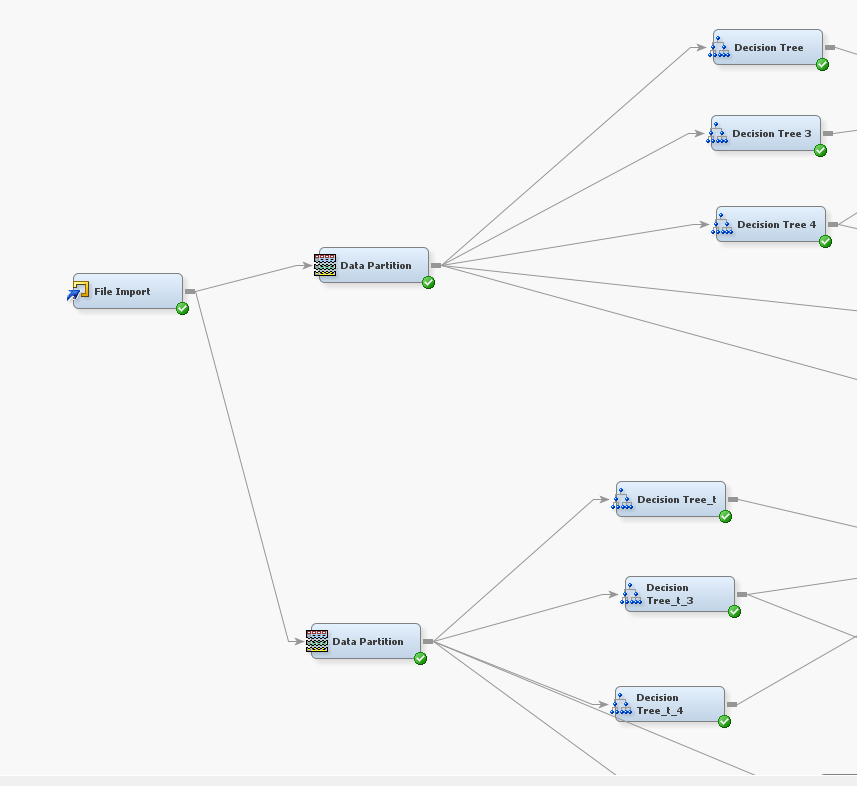
*DT with significance level = 0.2*



Based on the results, we decided to use significance factor = 0.2 as there was a not a major differnce between the 2 trees and it gave us more information about the input variables to use it for business analyisis.

## **Step 2:**

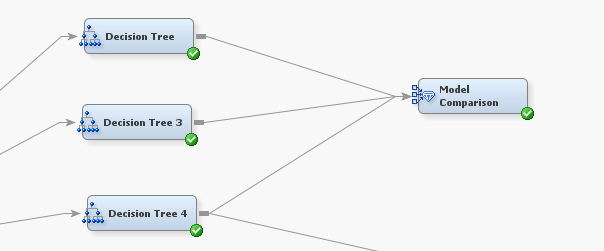
We have used 6 different decision trees in the initial analysis.



|  |  |  |
| --- | --- | --- |
|  | **Max Branches** | **Data Partition Node** |
| **Decision Tree 1** | 2 | 60:30:10 |
| **Decision Tree 2** | 3 | 60:30:10 |
| **Decision Tree 3** | 4 | 60:30:10 |
| **Decision Tree 4** | 2 | 60:20:20 |
| **Decision Tree 5** | 3 | 60:20:20 |
| **Decision Tree 6** | 4 | 60:20:20 |

## **Step 3:**

### **Model Comparison 1:**

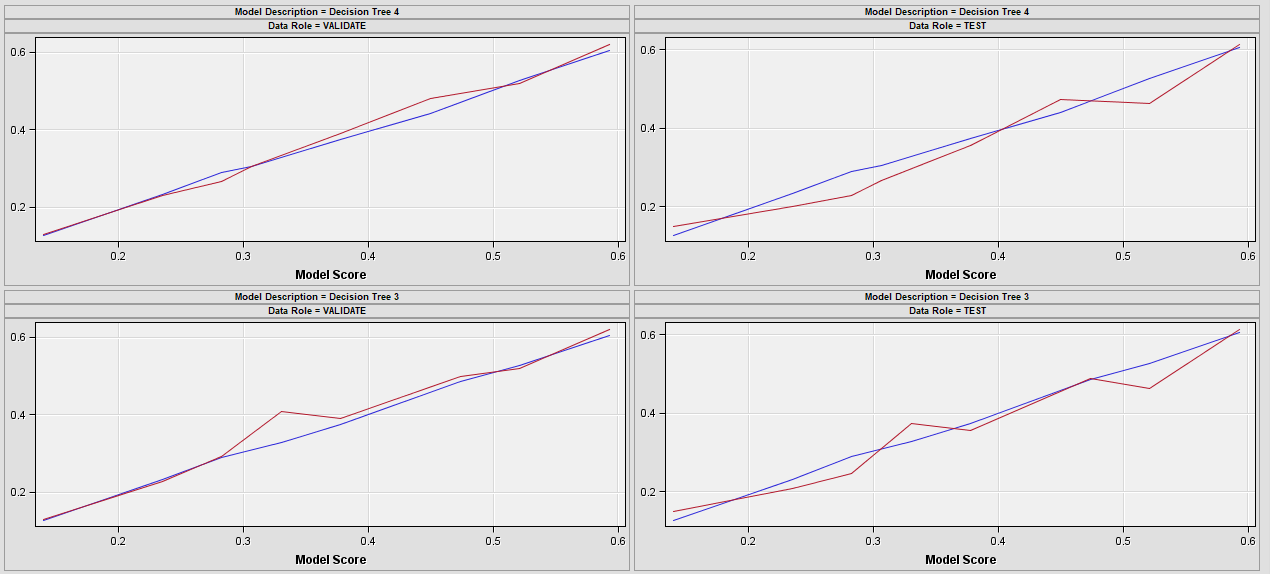


For this set of 3 trees, which differ by the max number of branches and use the first data partition node, we select the best fit tree using the results from the model comparison node.

#### ***Results of Model Comparison 1:***

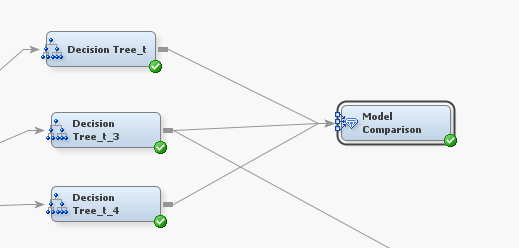
We use the results from the fit statistics and score distribution, to decide the best tree and use it in our further analysis.

We inspect the score distribution to check which trees perform better. We analyze where are the maximum gaps between the red and blue lines. For instance: A gap between the predicted and the actual value at lower levels of song hotness might not be as alarming as a significant gap between the two at higher levels of song hotness. The model might make a false prediction that a song will be hot whereas the reality might differ. This might lead to no-profitable expenses in advertisement and marketing.



Based on the diagram above, it looks like DT4 is giving more consistent results for validation and test data. Although fit statistics show that DT3 is better because of a lower average square error. For our use case, we decide to proceed with DT4.

### **Model Comparison 2:**

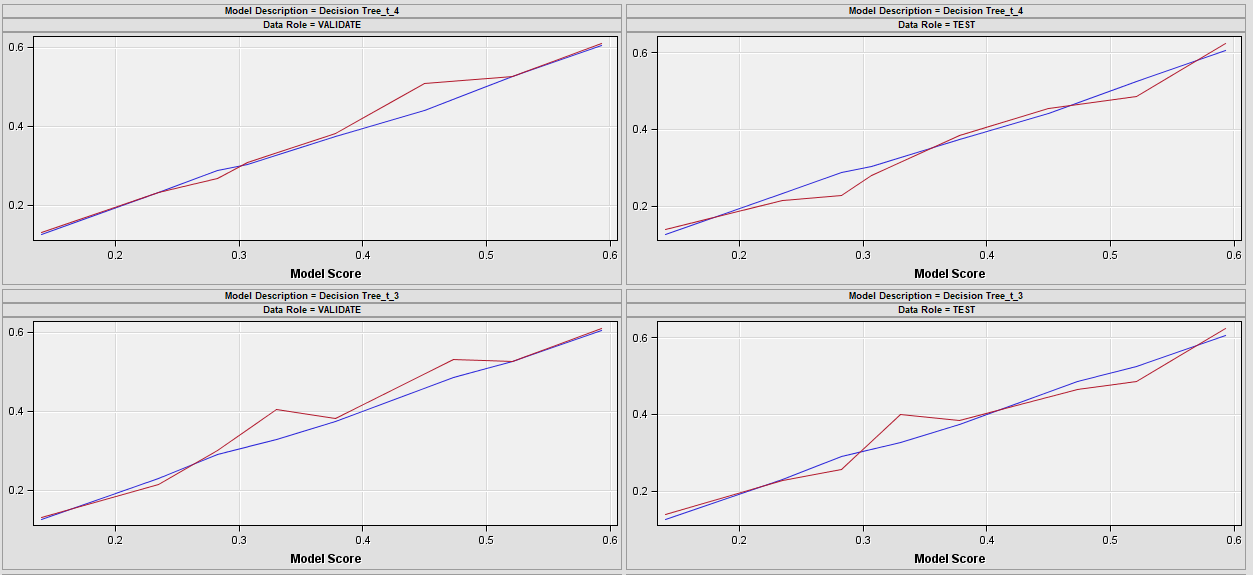
****

For this set of 3 trees, which differ by the max number of branches and use the second data partition node, we select the best fit tree using the results from the model comparison node.

#### ***Results of Model Comparison 2:***

We use the results from the fit statistics and score distribution, to decide the best tree and use it in our further analysis.

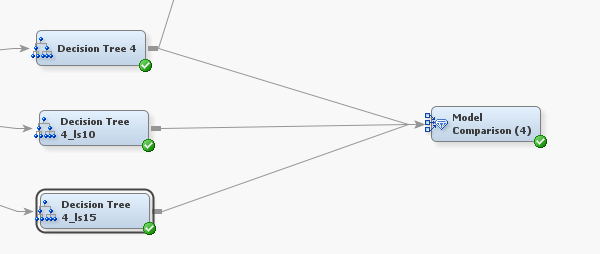
Based on the diagram above, it looks like DT\_t\_3 is giving more consistent results for validation and test data. Also, fit statistics show that DT\_t\_3 is better because of a lower average square error for both validation and test data. For this reason, we decide to proceed with DT\_t\_3.



Trees chosen from initial analysis:

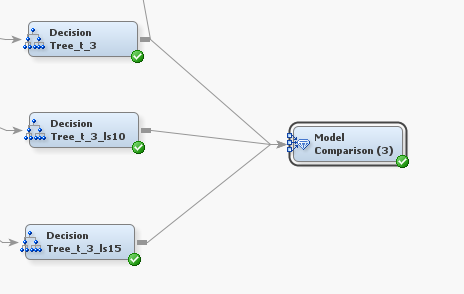
|  |  |  |
| --- | --- | --- |
|  | **Max Branches** | **Data Partition Node** |
| **Decision Tree 4** | 4 | 60:30:10 |
| **Decision Tree\_t\_3** | 3 | 60:20:20 |

## **Step 4:**



Using a similar strategy in the previous step, we now try to tweak the leaf size and make our decision tree give better results.We now create more copies of DT4 and DT\_t\_3 where we change the leaf size in {5, 10, 15} for each type of tree. We then find the best models using Model Comparison nodes.

Based on the results of the score distribution, we observe that DT4 *(max branches=4 and leaf size =5)* performs the best as the predicted values in the score distribution is relatively closer to the actual values.



We use a similar technique as the previous step to find the best fir model when we create 2 more copies of DT\_t\_3. All the decision trees now being compared differ by the leaf size{5,10,15}with data coming form the Data Partition Node 2(60:20:20).

Based on the results of the score distribution, we observe that **DT\_t\_3 *(max branches=3 and leaf size =5)*** performs the best as the predicted values in the score distribution are relatively closer to the actual values. Other trees show a large gap towards higher values due to which we do not move them forward in our analysis.

After **step 4**, we have narrowed our model down to the 2 best trees based on the below mentioned parameters: *(Max Branches, Leaf size, Data Partition Node)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Max Branches** | **Leaf Size** | **Data Partition Node** |
| **Decision Tree 4** | 4 | 5 | 60:30:10 |
| **Decision Tree\_t\_3** | 3 | 5 | 60:20:20 |

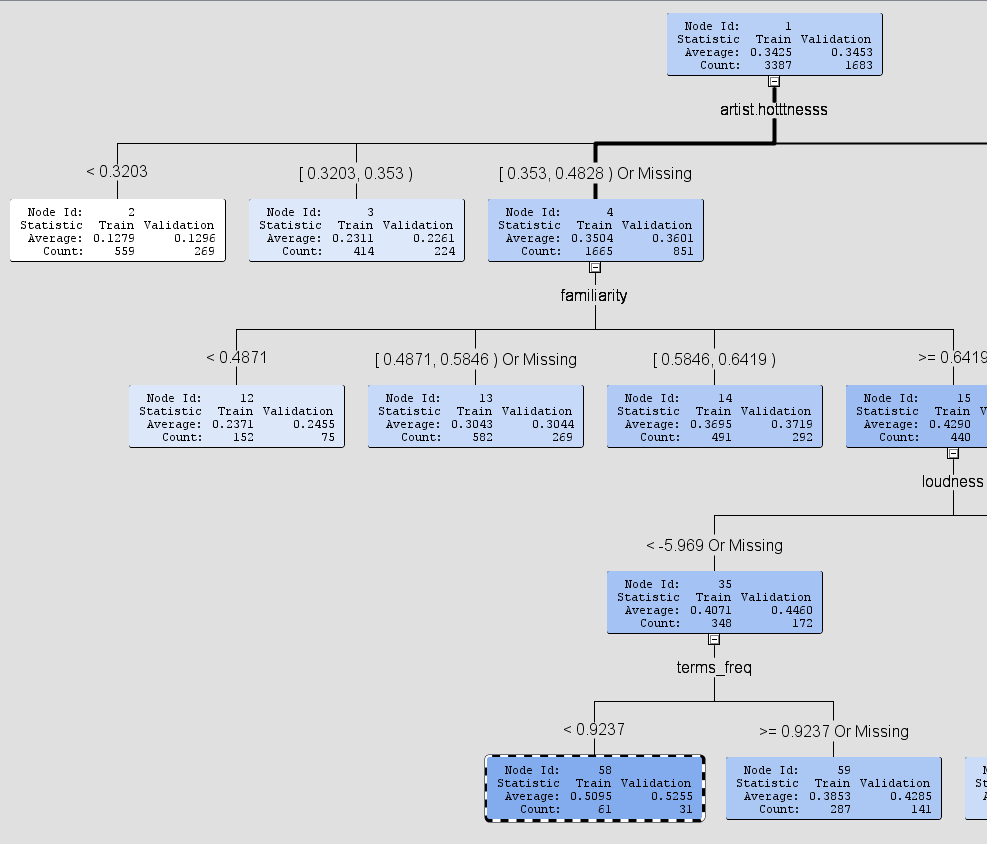
## **Step 5:**

Based on the results of the above table we see that:

1. If our data is partitioned in a 60:30:10 split, a better decision tree model to use would be **Decision Tree 4,** i.e. the one with 4 max branches and a leaf size of 5.
2. If our data is partitioned in a 60:20:20 split, a better decision tree model to use would be **Decision Tree\_t\_3,** i.e. the one with 3 max branches and a leaf size of 5.

Picking one final model from these 2 models will not give us accurate results as they both are built on different set of data (different Data Partition nodes). Hence, comparing them will not be an ideal choice.

# Key Insights

1. If the artist hotness is low, per say less than 0.5, a general notion is that the artist cannot produce popular songs. Which is again true and the same is shown as per or decision tree, but surprisingly the tree also shows that if the artist hotness is ranged from 0.353 to 0.4828, those artist can also produce songs which can be considered of a medium level hotness as indicated by the average value, only when if they follow the given parameters to produce their song, such as familiarity >= 0.6419 and loudness < -5.969 and terms\_freq > 0.9237.   
    
2. If the artist hotness is high, generally songs they produce are hot on charts, and so is shown by our decision tree, that if the artist hotness is >=0.4828 it will produce a hot song, but familiarity level should also be considered which is ideal, and so if the familiarity level is high per say >=0.7963 as per our model, songs produced will top the charts and will be considered high on hotness level.

