# Machine Learning Engineer Nanodegree

## Capstone Project

Kibaek Jeong
March 18th, 2019

## I. Definition

### Project Overview

Music has always been a constituent of human life in every known culture, past and present. In 2017, recorded music industry have generated $8.7 billion only in the United States (Friedlander). Music industry in United States have increased 16.5% in its value from 2016, due to increase in revenue from streaming services such as Apple Music, Spotify, SoundCloud, and others. As music industry has changed mainly from physical sales to digital sales, musicians and creators are more accessible to world wide audiences. However, producing a popular music is another story. The top 10 highest-paid musicians generated $886 million in 2018 (Greenburg). So what makes their music popular? Several studies have been made such as song popularity predictor by Mohamed Nasreldin, Stephen Ma, Eric Dailey, and Phuc Dang. With highest accuracy of 63%, research was able to find that tempo, artist familiarity and mode were top 3 important factors of song popularity.

In following project, further research is done with spotify music analysis datset. Features include artist popularity, artist follower, mode, key, time signature, and other audio analysis features. Supervised learning models such as linear regressor, logistic regressor, decision tree regressor, random forest regressor and support vector machine are examined as predictor model. Additionally, neural network is examined as predictor model.

## Problem Statement

There are so many factors that affect song popularity. It is important to musicians that what factors affect song popularity and to predict how popular a song will be. When artist can predict which song in an album is most likely to be popular, artist can promote the song more effectively. Also, if a song is predicted to be very popular, streaming services could recommend the song more often, as it means that large population will like the song. Using machine learning, we will train and test the model to predict how popular a song will be. For following project, we will be examining supervised models, such as linear regressor, logistic regressor, decision tree regressor, random forest regressor, and support vector machines. Also, we will examine neural network model from Keras and decide which model performs the best for predicting song popularity.

## Metrics

Song Popularity Predictor by Mohamed Nasreldin, Stephen Ma, Eric Dailey, and Phuc Dang, used area under curve (AUC) for the accuracy score. However, AUC score is adequate for binary classification. As our project is not a binary classification, another choice of evaluation metrics is necessary. Instead, in following project, we will use Root Mean Squared Error (RMSE), representing standard deviation of the difference between predicted value and actual value. Root Mean Squared Error can be represented mathematically as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Where N is number of data points.

Another metrics we could use is R squared metric. R squared metric provides how well the regression model is fitted, where 0 means model is totally out of fit, and 1 means perfect fit. R squared can be mathematically represented as:

$$r^2 = \frac{SS_{(regression)}}{SS_{(total)}}$$

Where SS(regression) is sum squared regression error and SS(total) is sum squared total error.

Our model will provide an value of track popularity in range of 0 to 100. As our output will be continuous, it is best for us to check how close the predicted value of track popularity is to actual value of track popularity. Our model will rarely be predicting exactly same value with actual value. Therefore, using metrics that checks whether model produced exact same number with actual value wouldn't be a good choice. Instead, by using root mean square error and R squared score, as they are both great metrics for a regression problem, we will be able to observe whether our model is properly working or not.

# II. Analysis

## Data Exploration

Dataset used in this project to train the model includes following features:

- Artist popularity
- Artist followers
- Confidence mean, standard deviation, kurtosis
- Loudness mean, standard deviation, kurtosis

- Tempo mean, standard deviation, kurtosis
- Tempo confidence mean, standard deviation, kurtosis
- Key mean, standard deviation, kurtosis
- Key confidence mean, standard deviation, kurtosis
- Mode mean, standard deviation, kurtosis
- Mode confidence mean, standard deviation, kurtosis
- Time signature mean, standard deviation, kurtosis
- Time signature confidence mean, standard deviation, kurtosis
- Pitch dominance  mean, standard deviation, kurtosis (From C,C# to B)
- Pitch dominance enthropy
- Timbre  mean, standard deviation, kurtosis (1 to 12)

Note: Timbre represents quality of sound or musical note. As mentioned from Spotify, 'It is a complex notion also reffered to as sound color, texture, or tone quality, and is derived from the shape of a segment's specgtro-temporal surface, independently of pitch and loudness.' For further information, please visit: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-analysis/

Data are imported directly from Spotipy, which is provided by Spotify. Spotipy provides data such as track popularity, artist popularity, artist followers and audio analysis for each song. Audio analysis for each segment is obtained from Spotipy and statistics such as mean, standard deviation and kurtosis are calculated. Blow is table of partial data, which is preprocessed to obtain statistics.

**Partial Processed Dataset**

| | track_name | artist_name | … | track_popularity | artist_popularity | artist_followers | confidence_mean | confidence_std | confidence_kurtosis | … |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Frogbass - Original Mix | Snails | … | 45 | 55 | 86330 | 0.403243 | 0.271824 | -3.890714 | … |
| **1** | Pump This - Getter Remix | Getter | … | 25 | 62 | 201978 | 0.528772 | 0.437708 | 2.141111 | … |
| **2** | Going Gorillas | Doctor P | … | 29 | 49 | 176097 | 0.468465 | 0.380187 | 207.992583 | … |
| **3** | STOMP - Original Mix | Snails | … | 32 | 55 | 86330 | 0.471887 | 0.303001 | 9.053980 | … |

When distribution is visualized, some of the dataset were highly skewed. Log transformation and normalization was done for the dataset to eliminate high skewness. Following features were considered as highly skewed.

- artist_followers
- confidence_kurtosis
- loudness_std
- loudness_kurtosis
- tempo_kurtosis
- tempo_confidence
- key_kurtosis
- key_confidence_kurtosis
- mode_kurtosis
- mode_confidence_kurtosis
- time_signature_mean

- time_signature_kurtosis
- time_signature_confidence_mean
- time_signature_confidence_kurtosis
- C_dominance_std
- C_dominance_kurtosis
- C#_dominance_std
- C#_dominance_kurtosis
- D_dominance_std
- D_dominance_kurtosis
- D#_dominance_std
- D#_dominance_kurtosis
- E_dominance_std
- E_dominance_kurtosis
- F_dominance_std
- F_dominance_kurtosis
- F#_dominance_std
- F#_dominance_kurtosis
- G_dominance_std
- G_dominance_kurtosis
- G#_dominance_std
- G#_dominance_kurtosis
- A_dominance_std
- A_dominance_kurtosis
- A#_dominance_std
- A#_dominance_kurtosis
- B_dominance_std
- B_dominance_kurtosis
- pitch_entropy

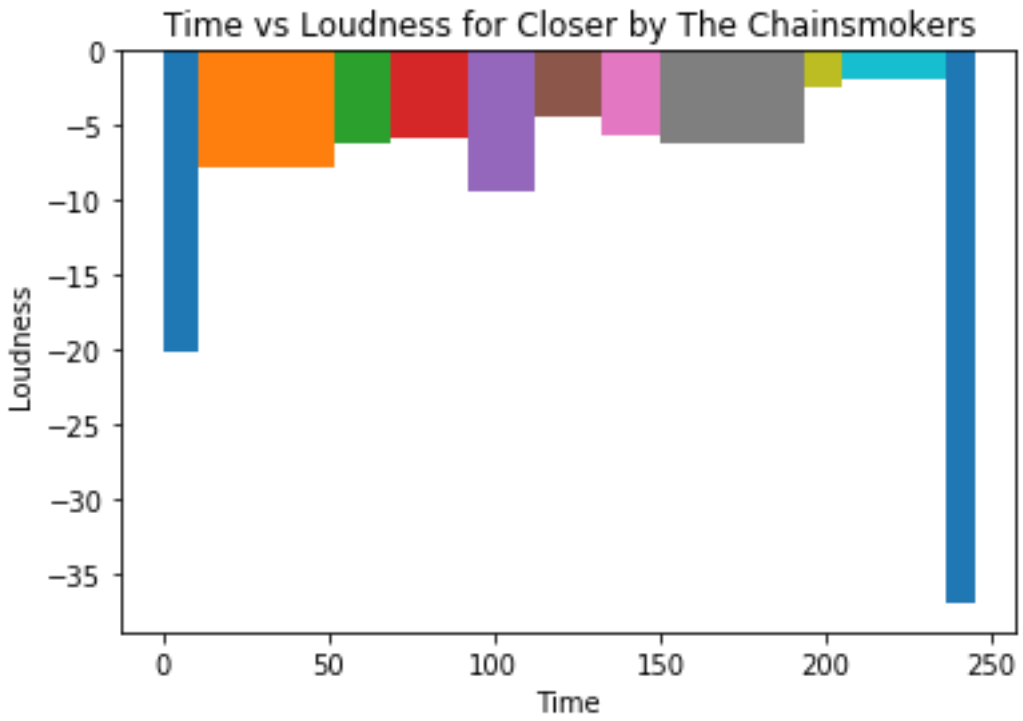
- timbre_1_std
- timbre_1_kurtosis
- timbre_2_std
- timbre_2_kurtosis
- timbre_3_std
- timbre_3_kurtosis
- timbre_4_std
- timbre_4_kurtosis
- timbre_5_std
- timbre_5_kurtosis
- timbre_6_std
- timbre_6_kurtosis
- timbre_7_std
- timbre_7_kurtosis
- timbre_8_std
- timbre_8_kurtosis
- timbre_9_std
- timbre_9_kurtosis
- timbre_10_std
- timbre_10_kurtosis
- timbre_11_std
- timbre_11_kurtosis
- timbre_12_std
- timbre_12_kurtosis

Skewed distribution of features negatively affect the performance of a learning algorithm. Therefore, logarithm transformation will be applied to these features to reduce affect of values caused by outliers.

Out of all the skewed features, artist follower, pitch dominance kurtosis, timbre kurtosis were most highly skewed. (Distribution histogram on  Most of the artists have around 0 followers. Also, most of the song pitch dominance (from C,C# to B) and timbre (from 1 to 12) has kurtosis around zero. Kurtosis is a measure of symmetry in a distribution. In our data, most of the song has distribution of each pitch dominance and timbre is shorter than normal distribution. Indicating data are light tailed and has less outliers.

## Exploratory Visualization

When data is directly collected from spotipy, it provides value of each audio analysis for each segment. Each segment is not equally divided. For example, for the loudness value vs time of the song Closer by The Chainsmokers can be seen in diagram below.



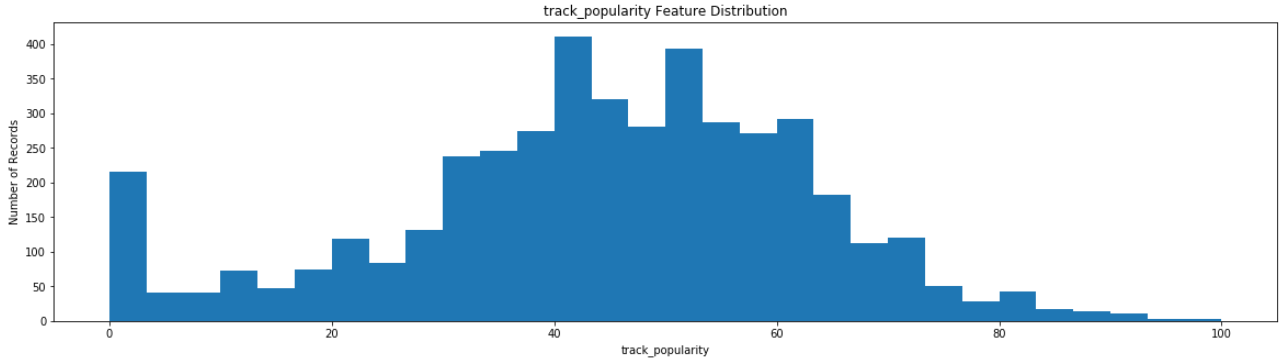

As we can see, each data point for loudness has different time length. This is because segment is divided differently. For preprocessing, this factor had to be

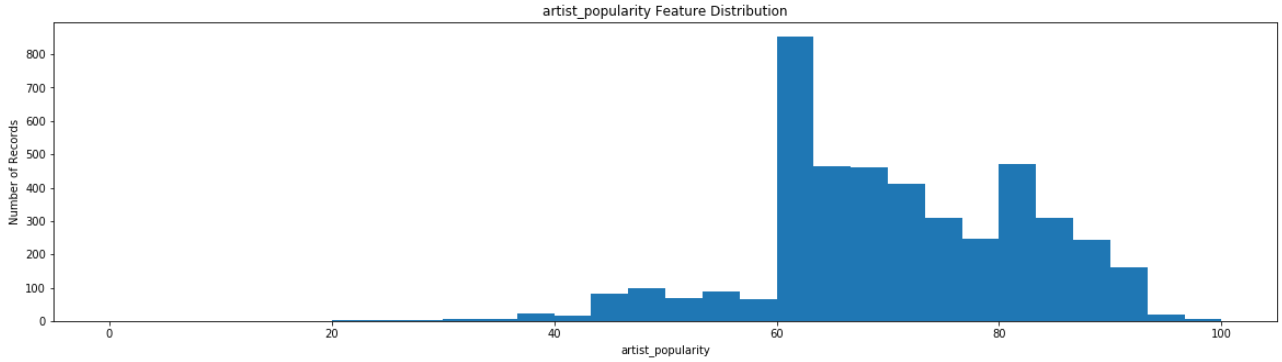
considered. Therefore, weight, time length for each segment, was applied to each data point, when collecting and preprocessing data.

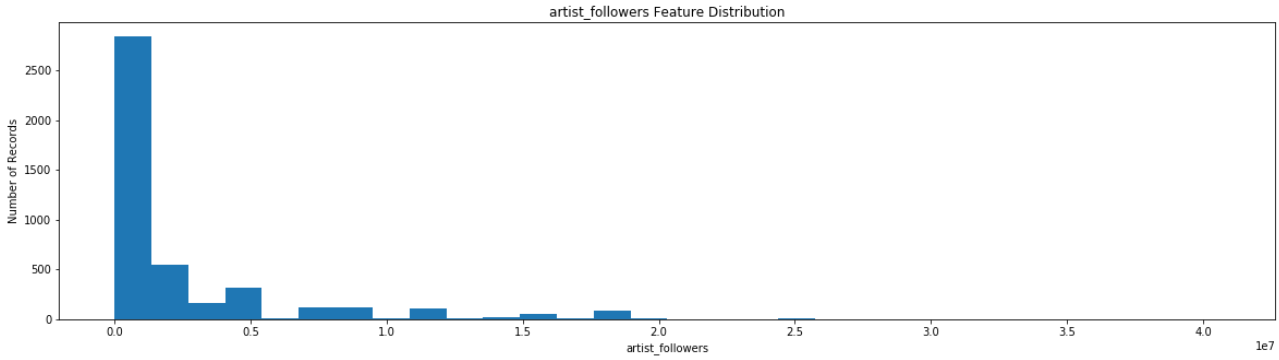
After all the data was collected for numbers of songs, statistics such as mean, standard deviation and kurtosis were calculated. Distribution of such statistics of each features are shown below.

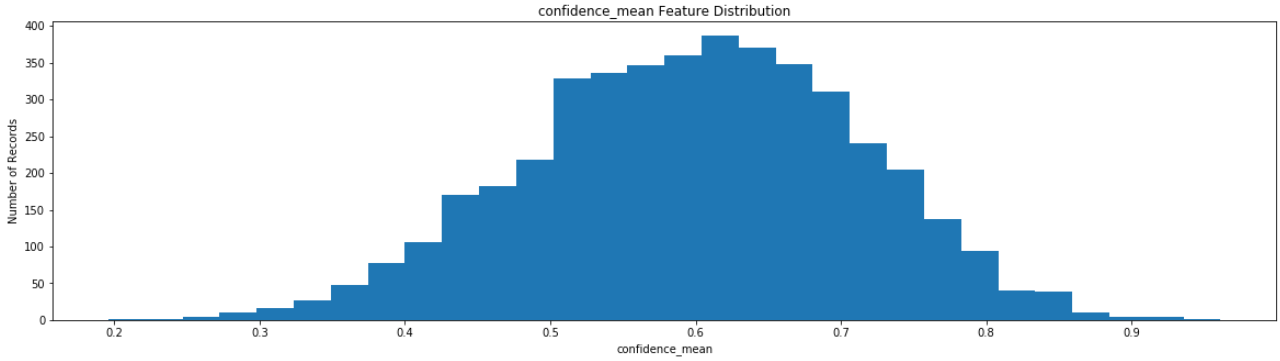
## 1. Feature distribution before log transformation

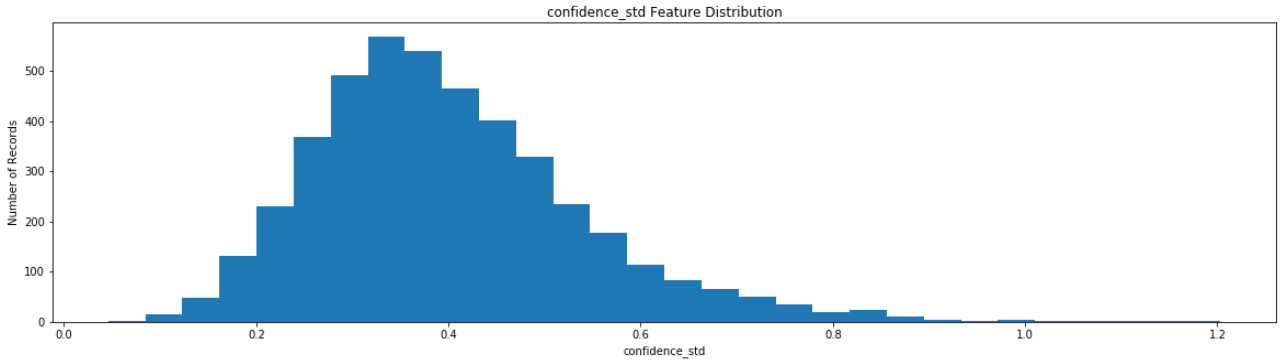
After all the data was collected for numbers of songs, statistics such as mean, standard deviation and kurtosis were calculated. Distribution of such statistics of each features are shown below.

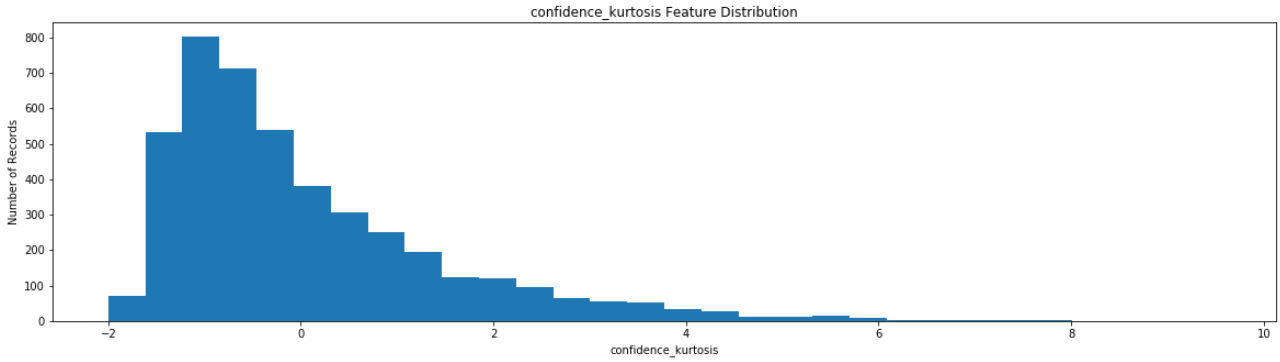


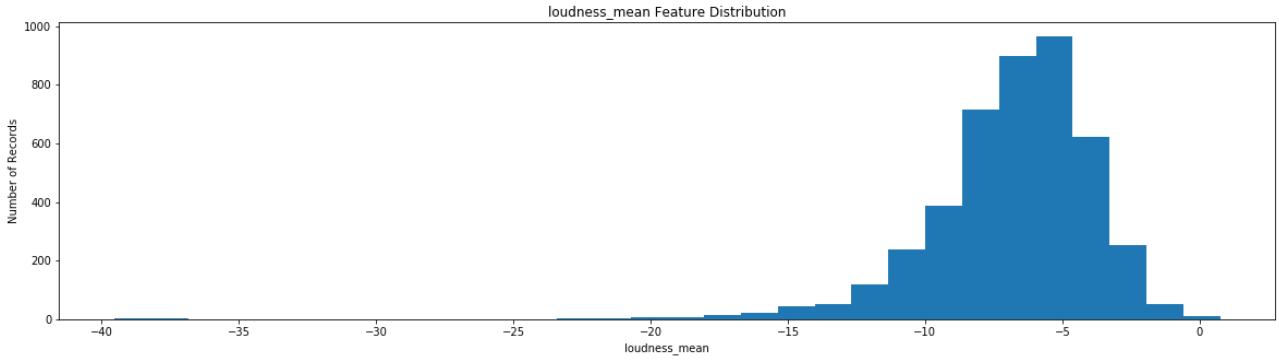



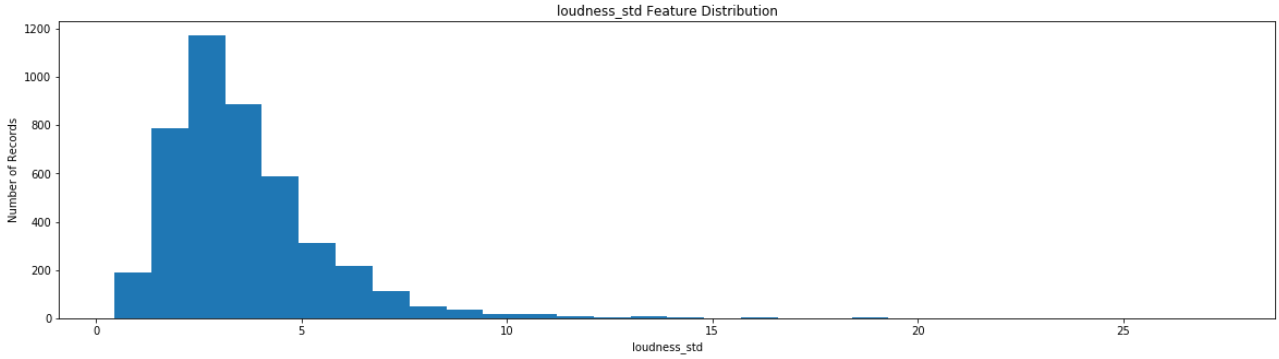

artist_followers Feature Distribution
Number of Records
artist_followers
1e7

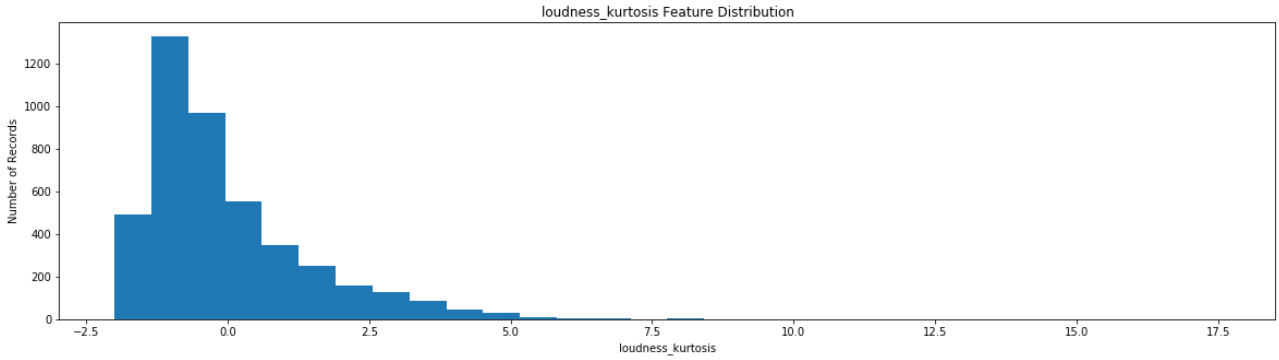

confidence_mean Feature Distribution
Number of Records
confidence_mean

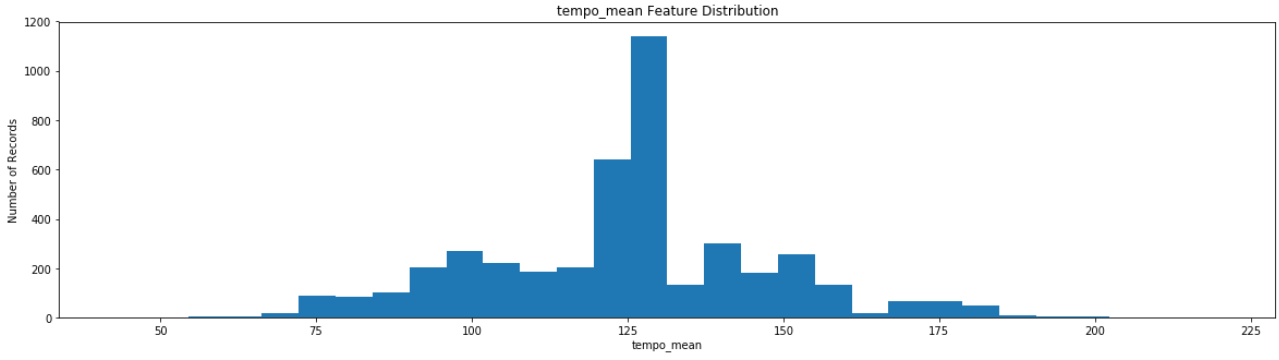

confidence_std Feature Distribution
Number of Records
confidence_std

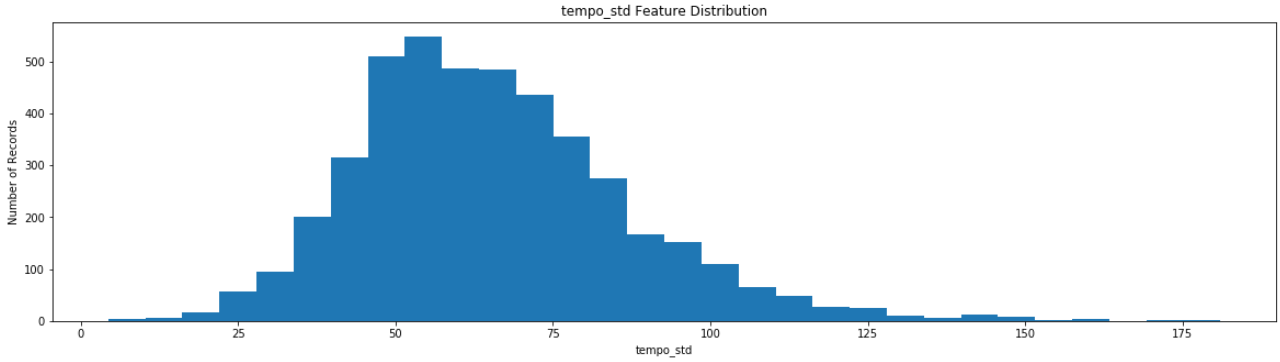

confidence_kurtosis Feature Distribution
Number of Records
confidence_kurtosis

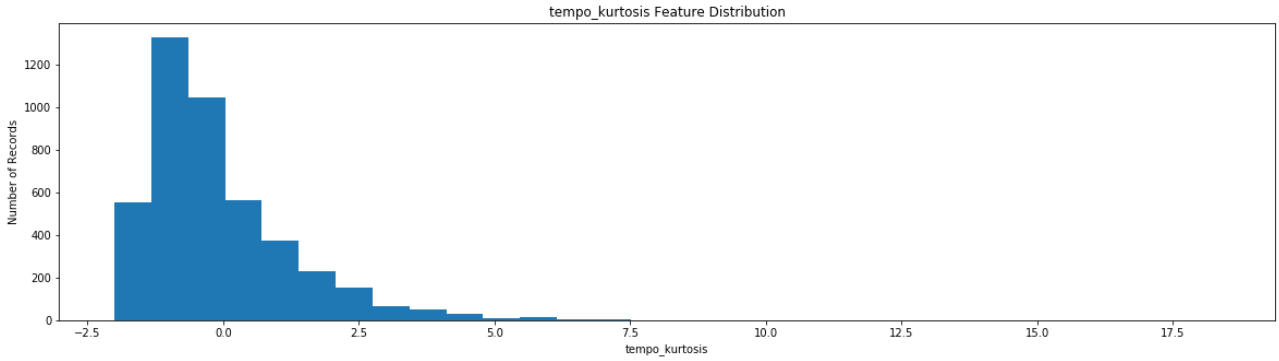

loudness_mean Feature Distribution
Number of Records
loudness_mean

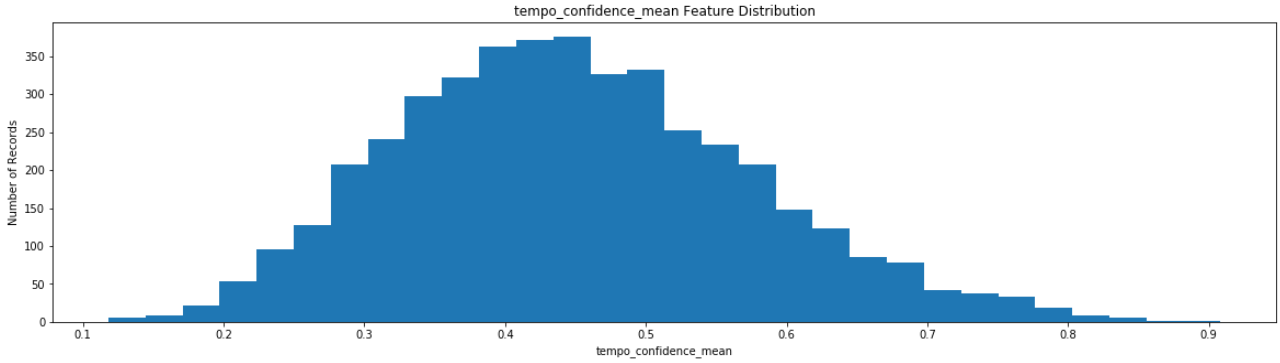

loudness_std Feature Distribution
Number of Records
loudness_std

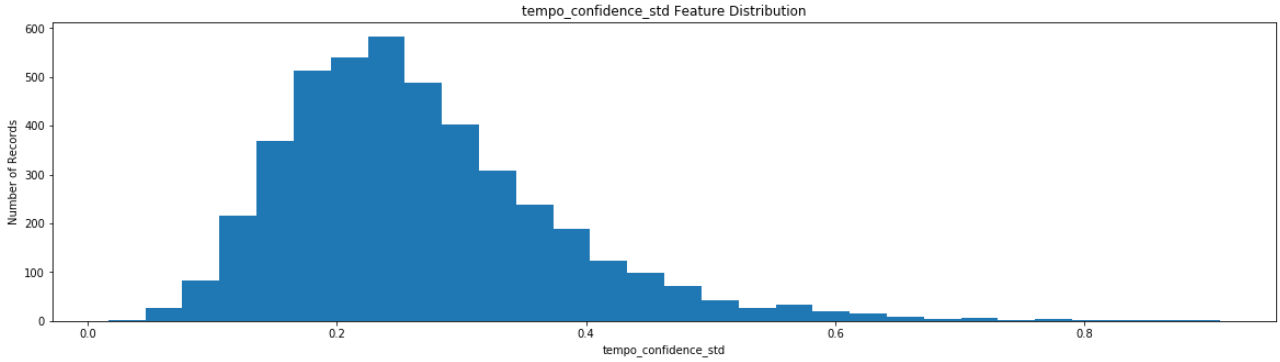

loudness_kurtosis Feature Distribution
Number of Records
loudness_kurtosis

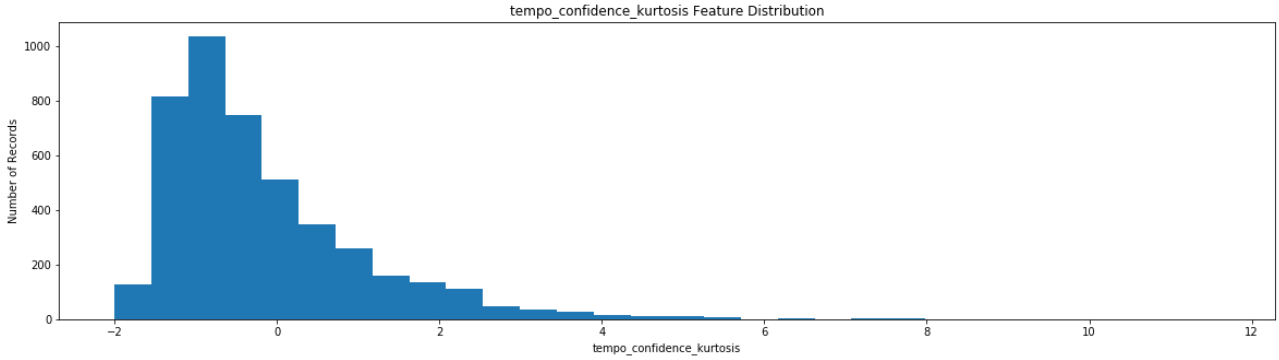

tempo_mean Feature Distribution
Number of Records
tempo_mean

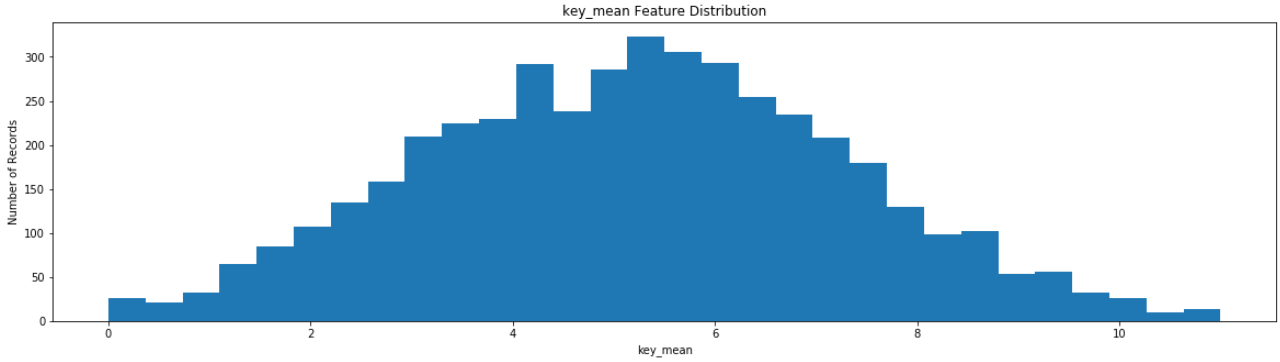

tempo_std Feature Distribution
Number of Records
tempo_std

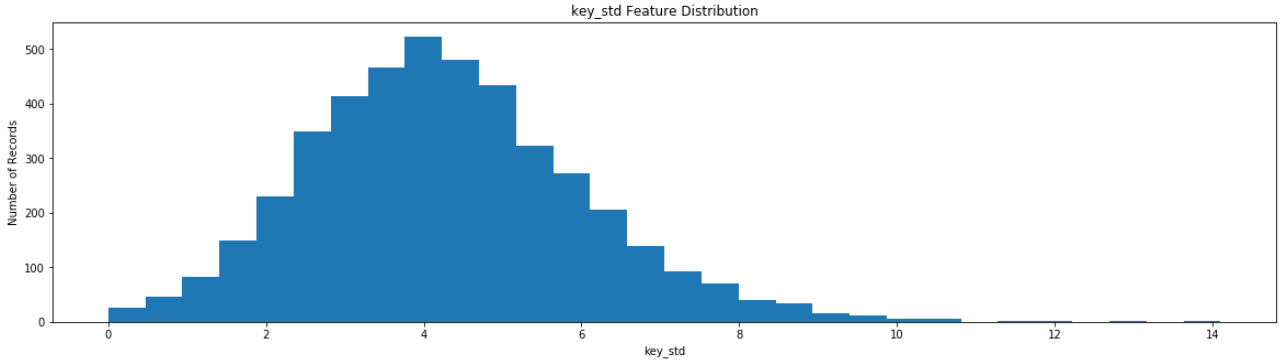

tempo_kurtosis Feature Distribution
Number of Records
tempo_kurtosis

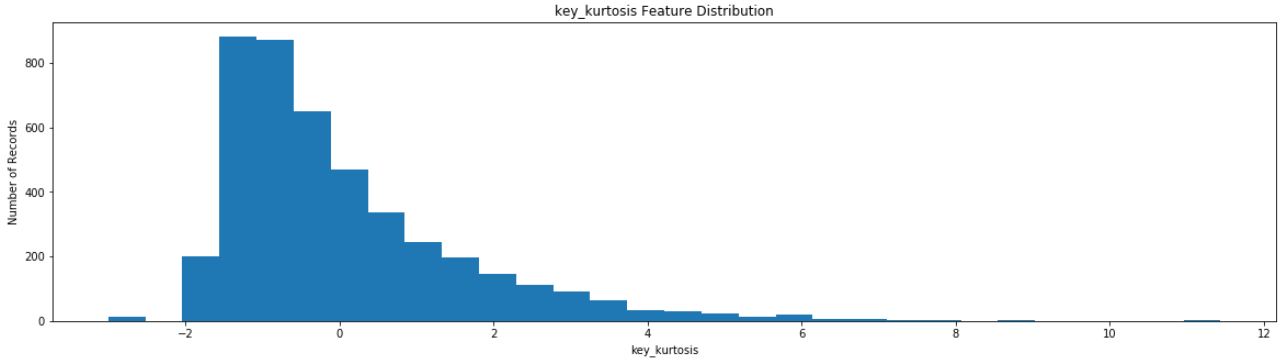

tempo_confidence_mean Feature Distribution
Number of Records
tempo_confidence_mean

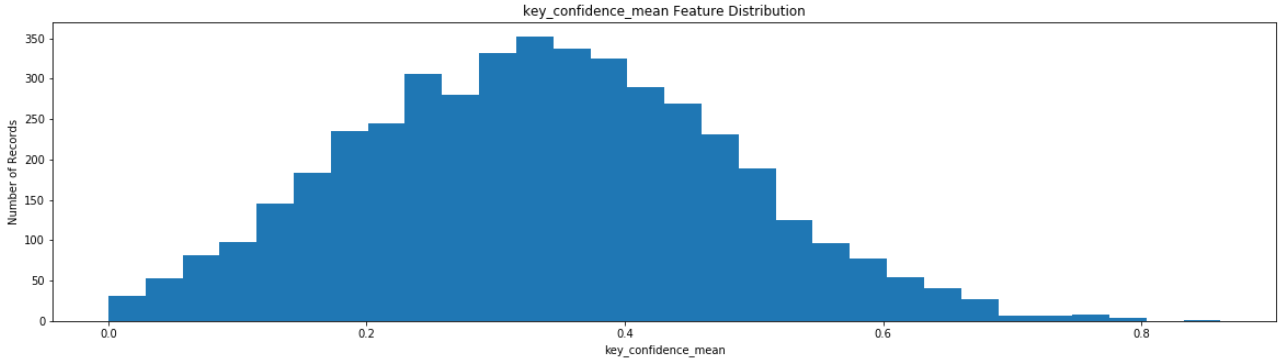

tempo_confidence_std Feature Distribution
Number of Records
tempo_confidence_std

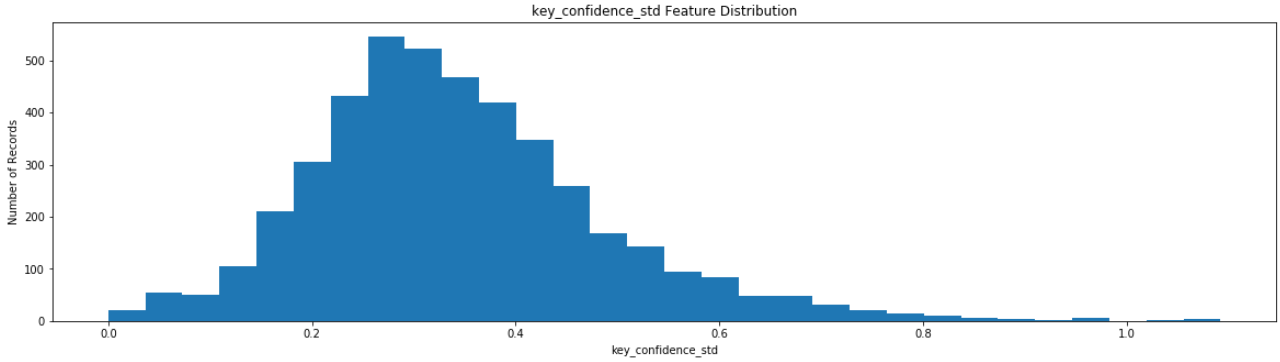

tempo_confidence_kurtosis Feature Distribution
Number of Records
tempo_confidence_kurtosis

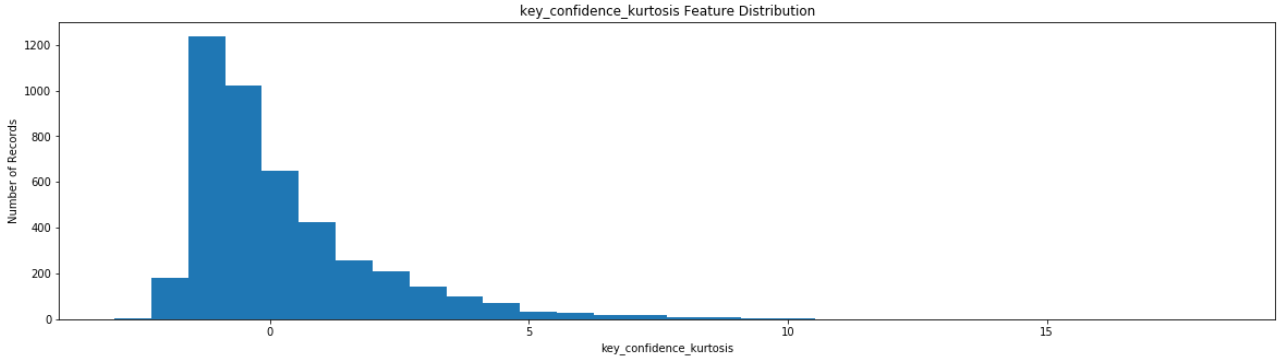

key_mean Feature Distribution
Number of Records
key_mean

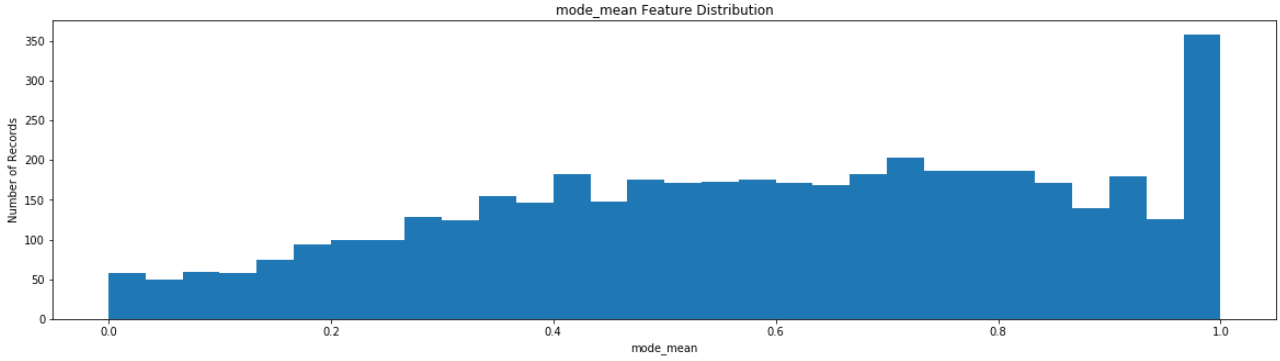

key_std Feature Distribution
Number of Records
key_std

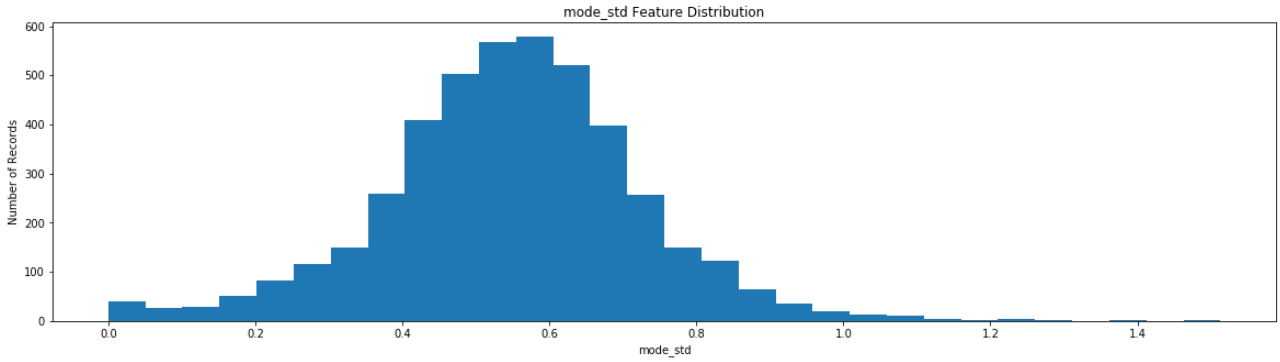

key_kurtosis Feature Distribution
Number of Records
key_kurtosis

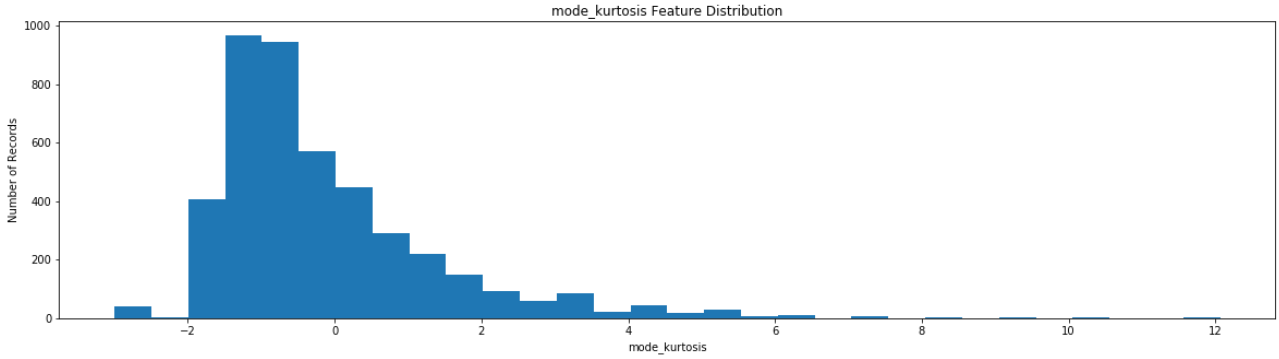

key_confidence_mean Feature Distribution
Number of Records
key_confidence_mean

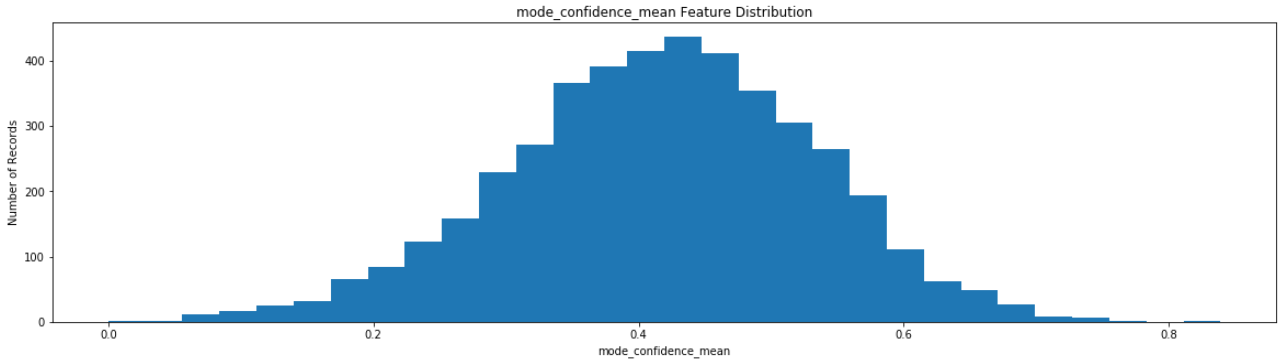

key_confidence_std Feature Distribution
Number of Records
key_confidence_std

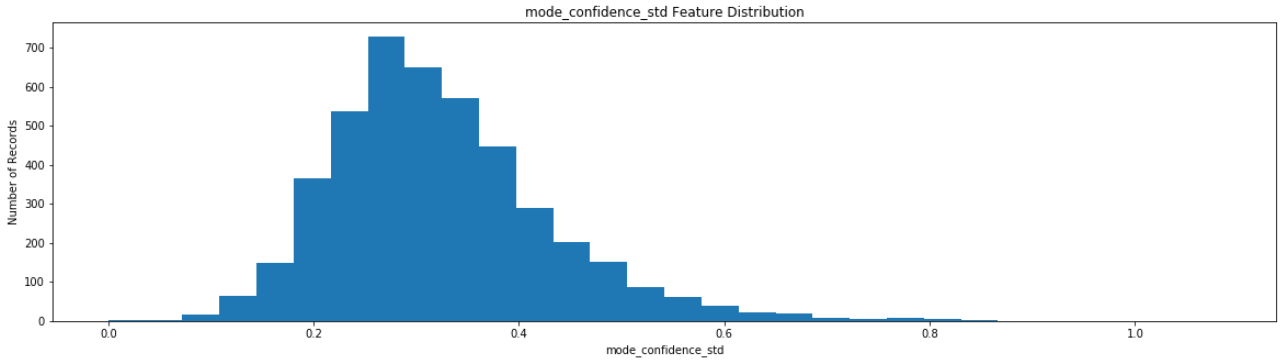

key_confidence_kurtosis Feature Distribution
Number of Records
key_confidence_kurtosis

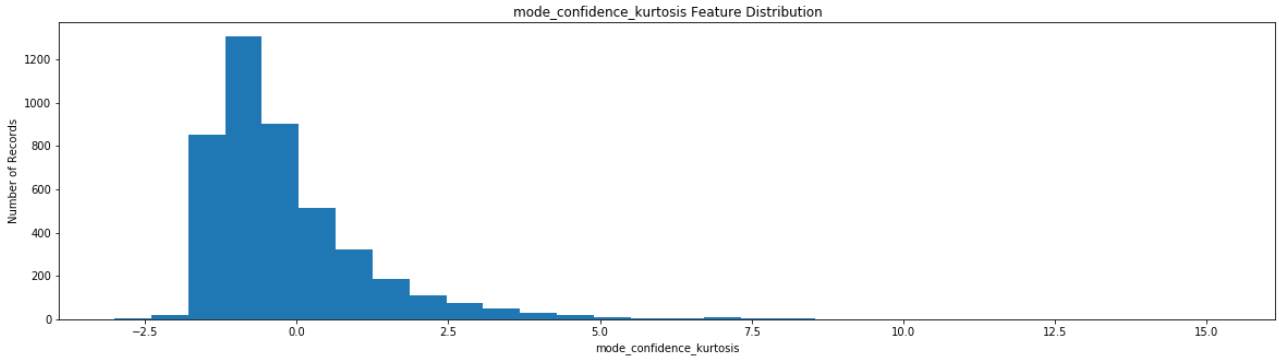

mode_mean Feature Distribution
Number of Records
mode_mean

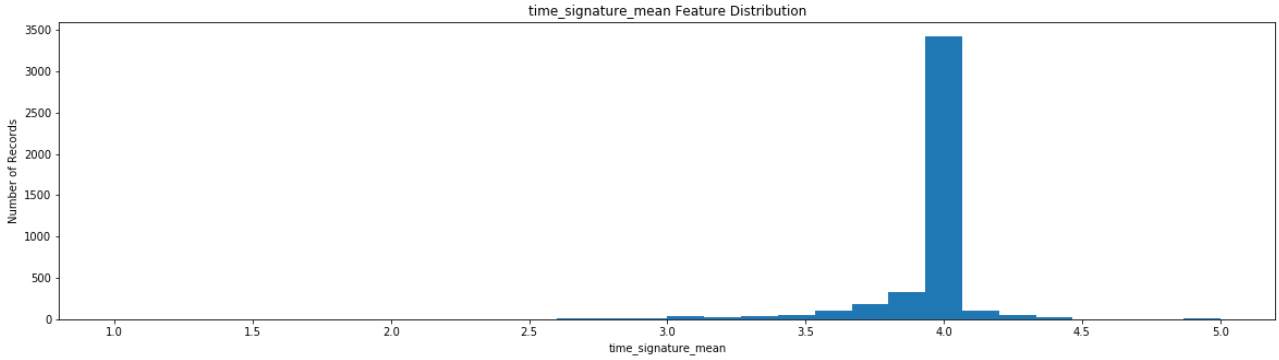

mode_std Feature Distribution
Number of Records
mode_std

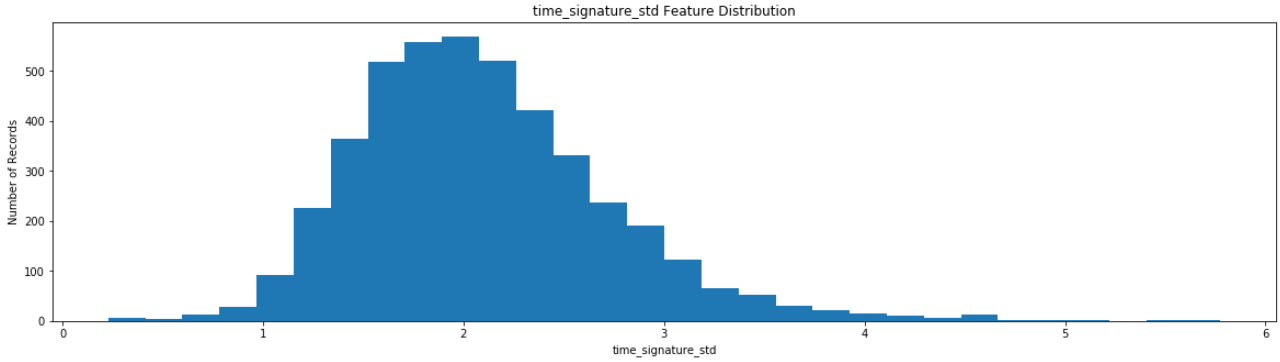

mode_kurtosis Feature Distribution
Number of Records
mode_kurtosis

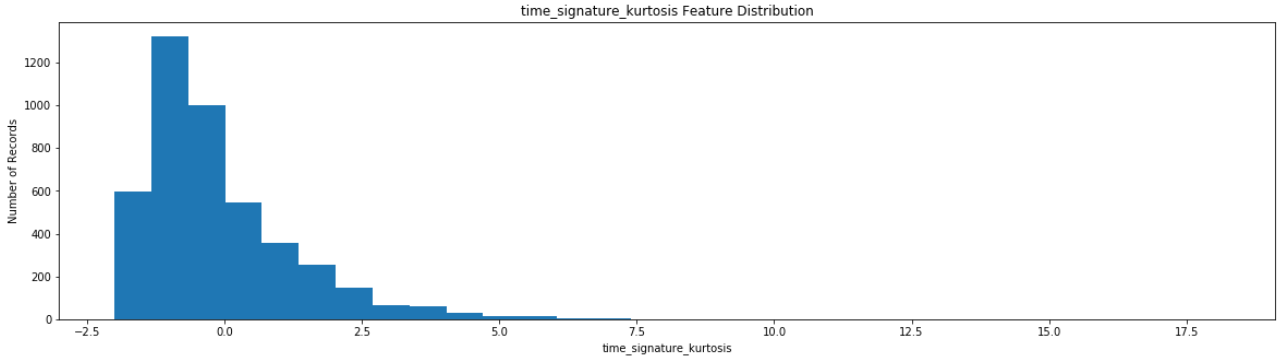

mode_confidence_mean Feature Distribution
Number of Records
mode_confidence_mean

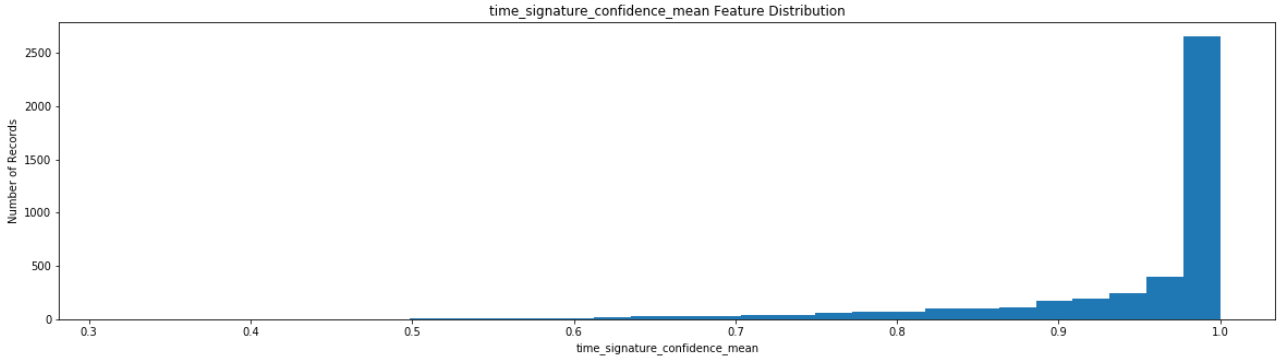

mode_confidence_std Feature Distribution
Number of Records
mode_confidence_std

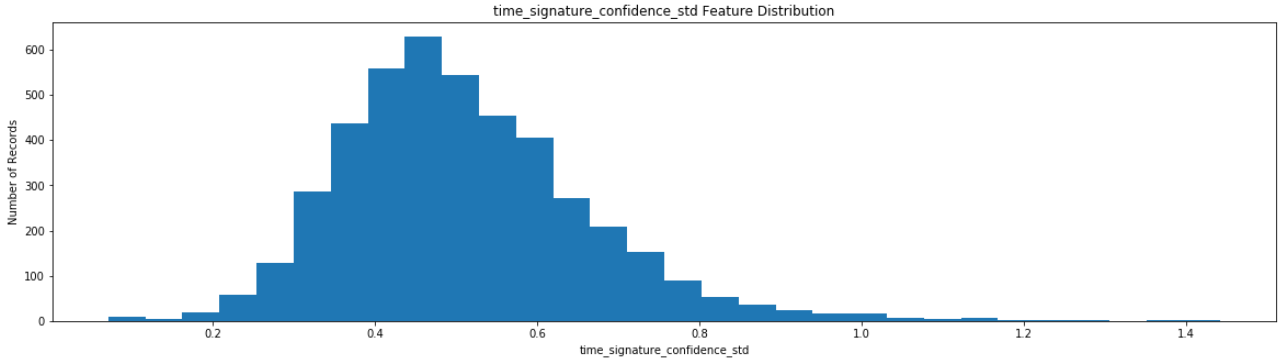

mode_confidence_kurtosis Feature Distribution
Number of Records
mode_confidence_kurtosis

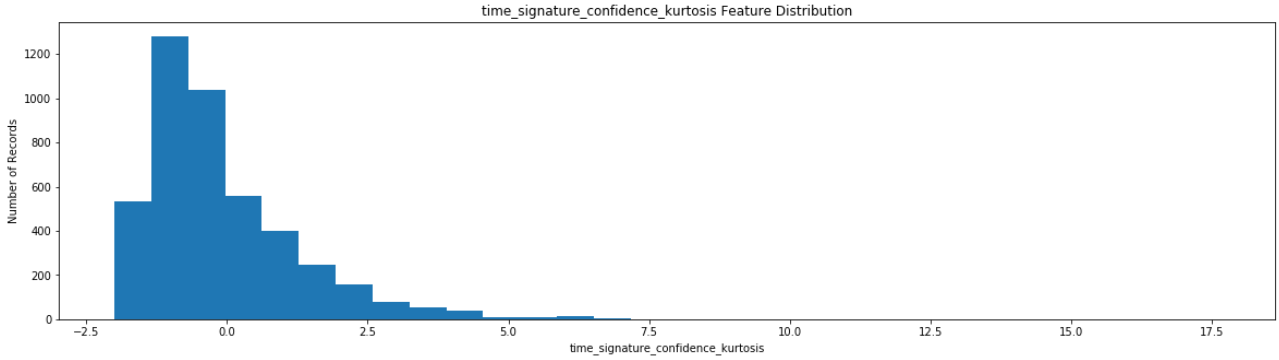

time_signature_mean Feature Distribution
Number of Records
time_signature_mean

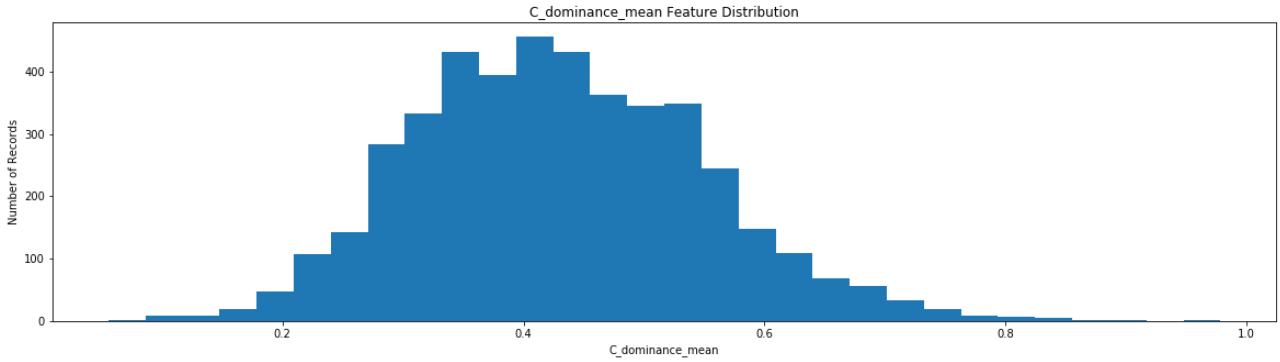

time_signature_std Feature Distribution
Number of Records
time_signature_std

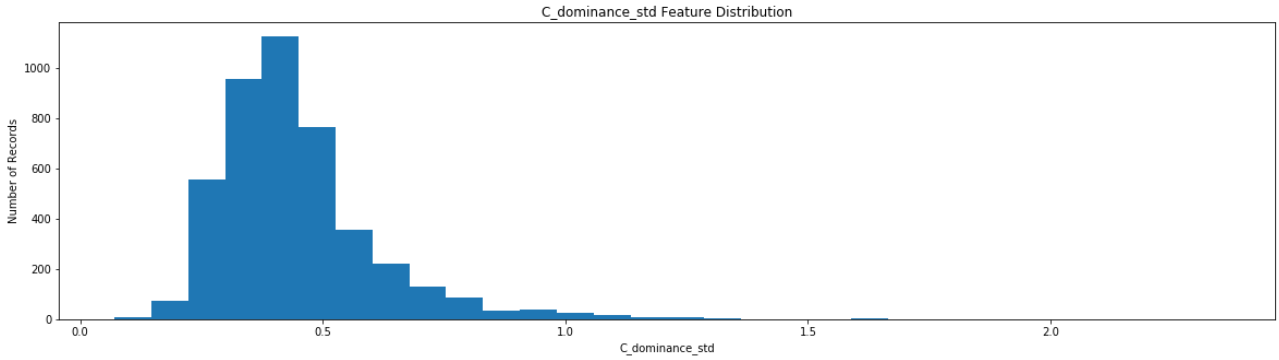

time_signature_kurtosis Feature Distribution
Number of Records
time_signature_kurtosis

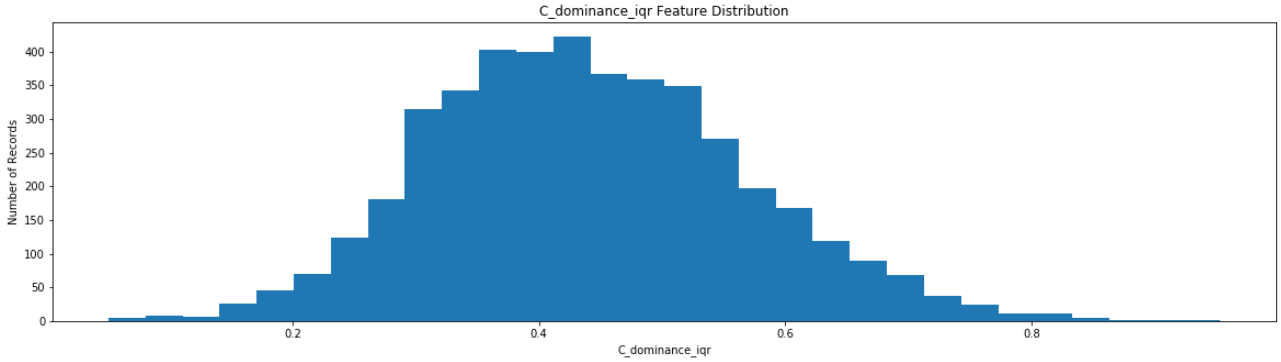

time_signature_confidence_mean Feature Distribution
Number of Records
time_signature_confidence_mean

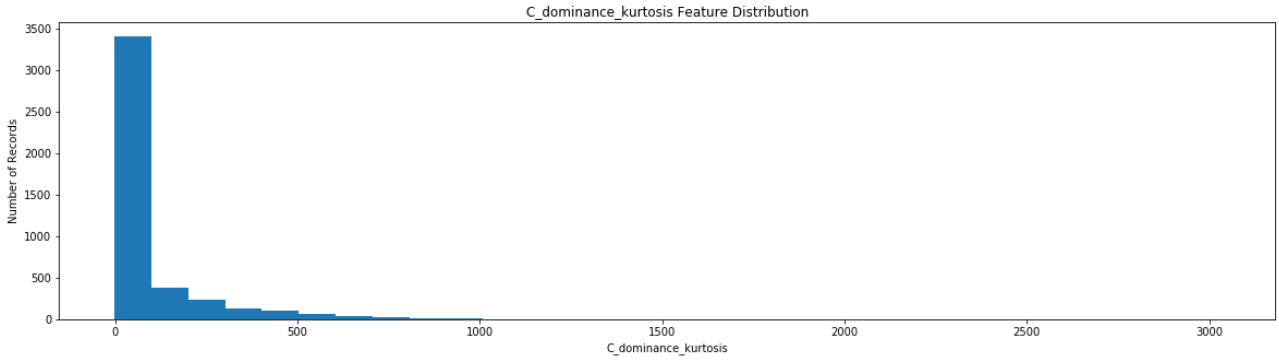

time_signature_confidence_std Feature Distribution
Number of Records
time_signature_confidence_std

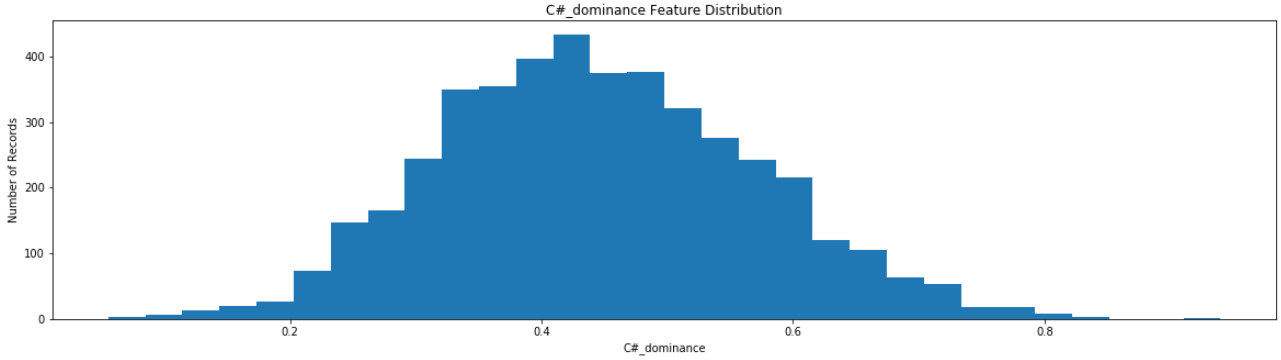

time_signature_confidence_kurtosis Feature Distribution
Number of Records
time_signature_confidence_kurtosis

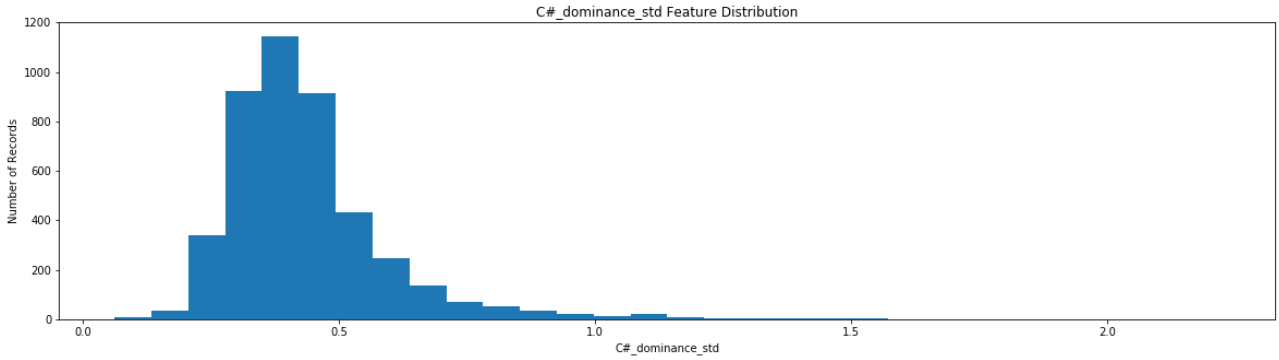

C_dominance_mean Feature Distribution
Number of Records
C_dominance_mean

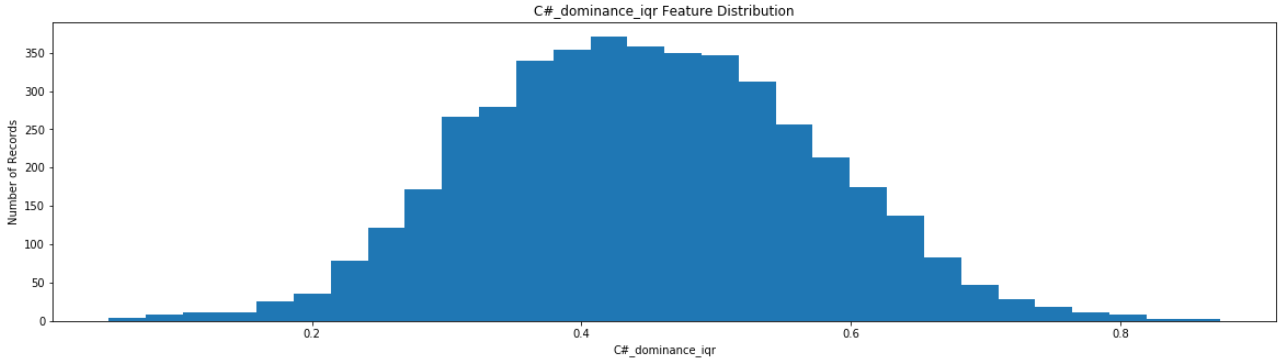

C_dominance_std Feature Distribution
Number of Records
C_dominance_std

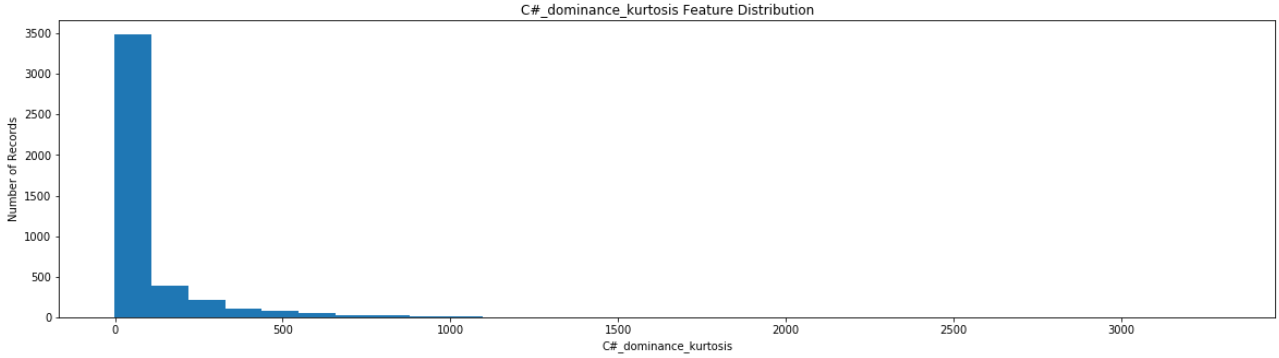

C_dominance_iqr Feature Distribution
Number of Records
C_dominance_iqr

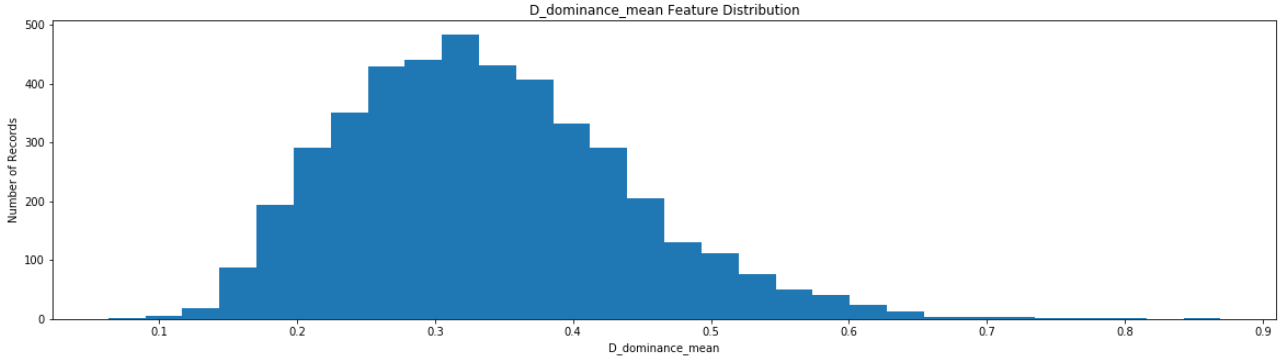

C_dominance_kurtosis Feature Distribution
Number of Records
C_dominance_kurtosis

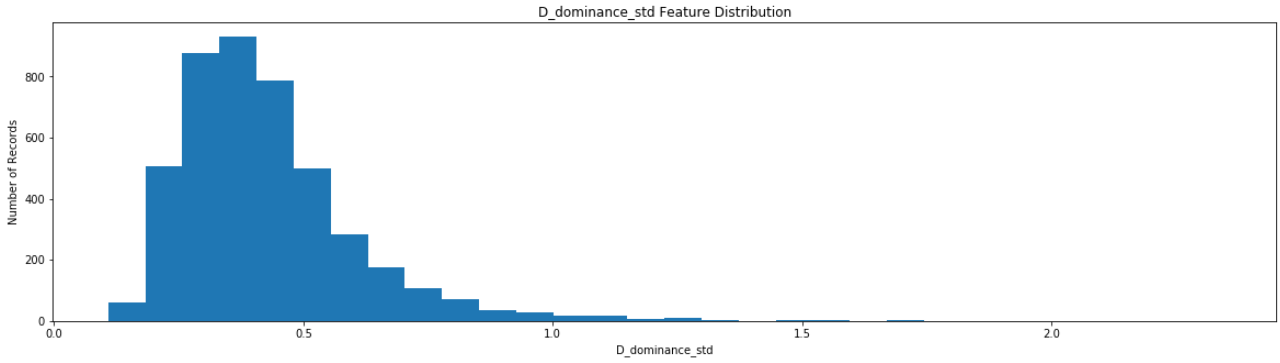

C#_dominance Feature Distribution
Number of Records
C#_dominance

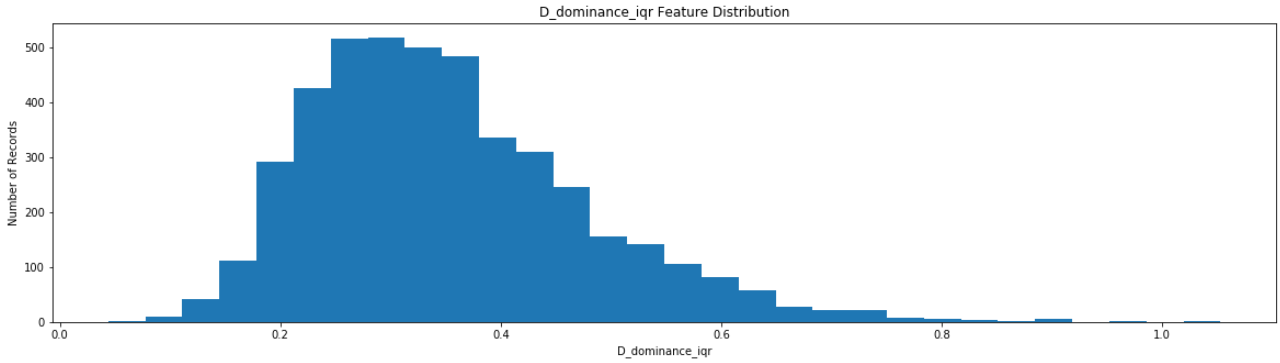

C#_dominance_std Feature Distribution
Number of Records
C#_dominance_std
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000
1200

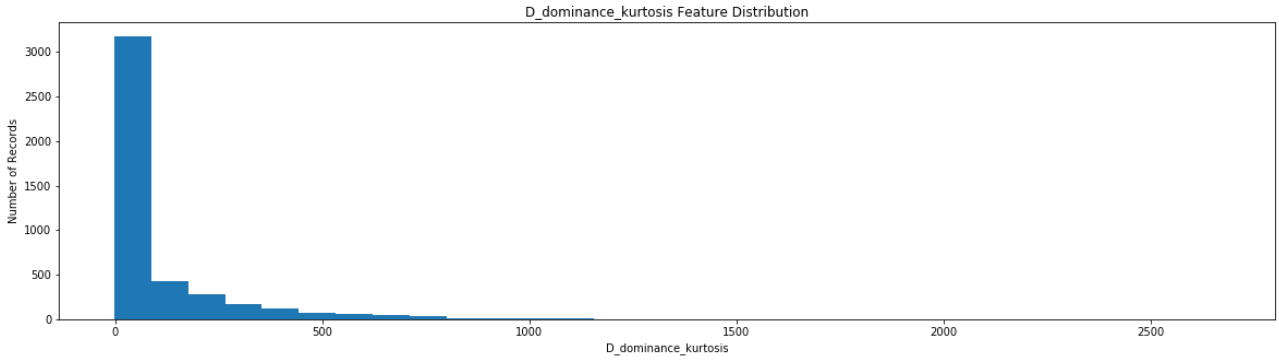

C#_dominance_iqr Feature Distribution
Number of Records
C#_dominance_iqr
0.2
0.4
0.6
0.8
0
50
100
150
200
250
300
350

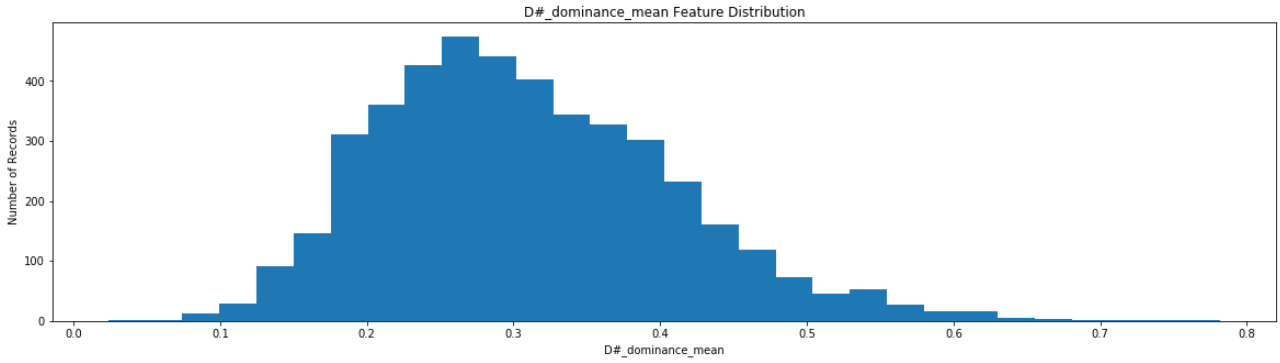

C#_dominance_kurtosis Feature Distribution
Number of Records
C#_dominance_kurtosis
0
500
1000
1500
2000
2500
3000
0
500
1000
1500
2000
2500
3000
3500

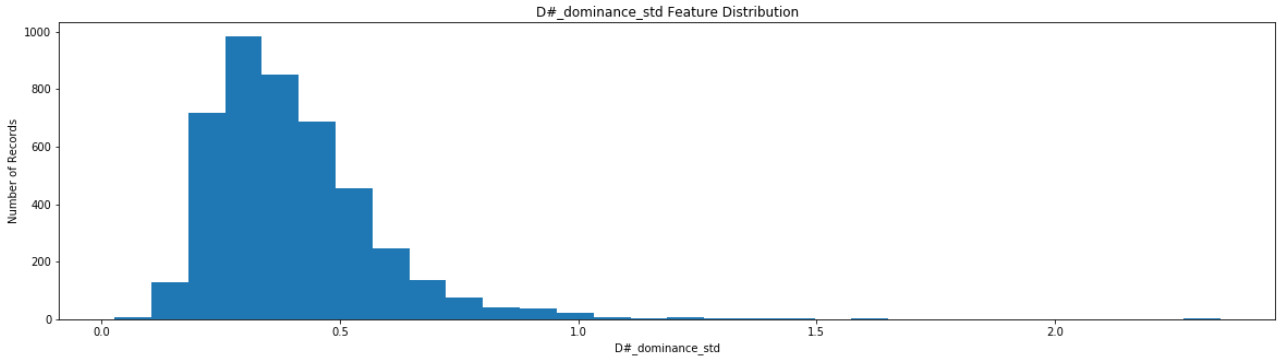

D_dominance_mean Feature Distribution
Number of Records
D_dominance_mean
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
100
200
300
400
500

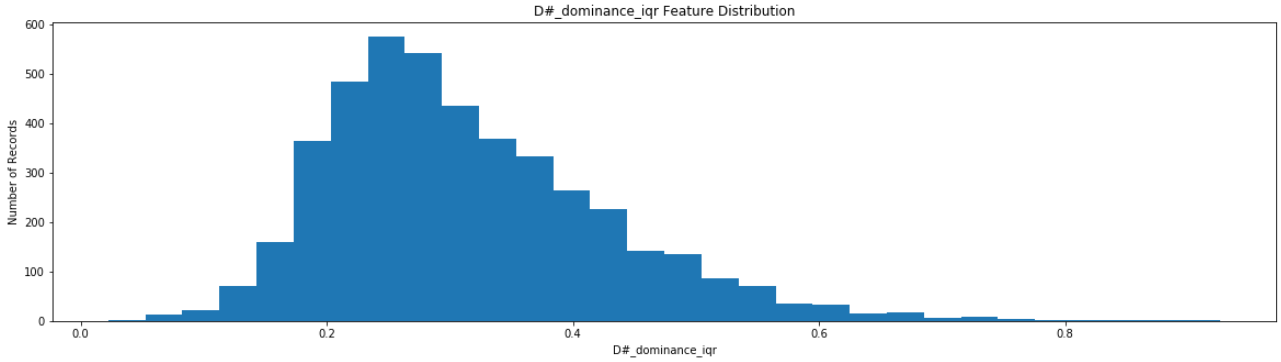

D_dominance_std Feature Distribution
Number of Records
D_dominance_std

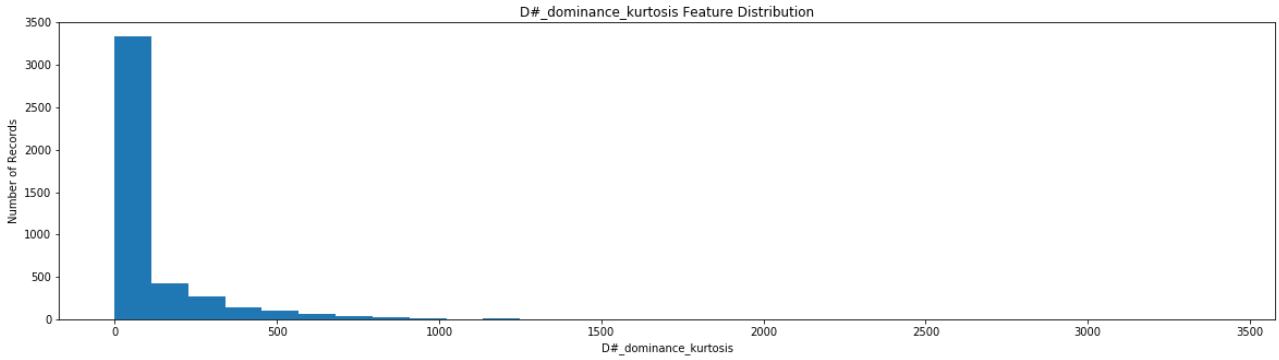

D_dominance_iqr Feature Distribution
Number of Records
D_dominance_iqr

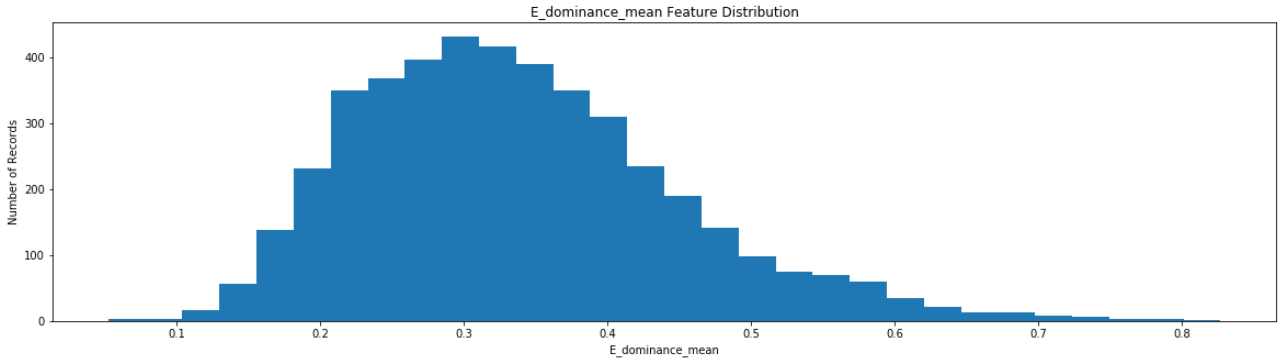

D_dominance_kurtosis Feature Distribution
Number of Records
D_dominance_kurtosis

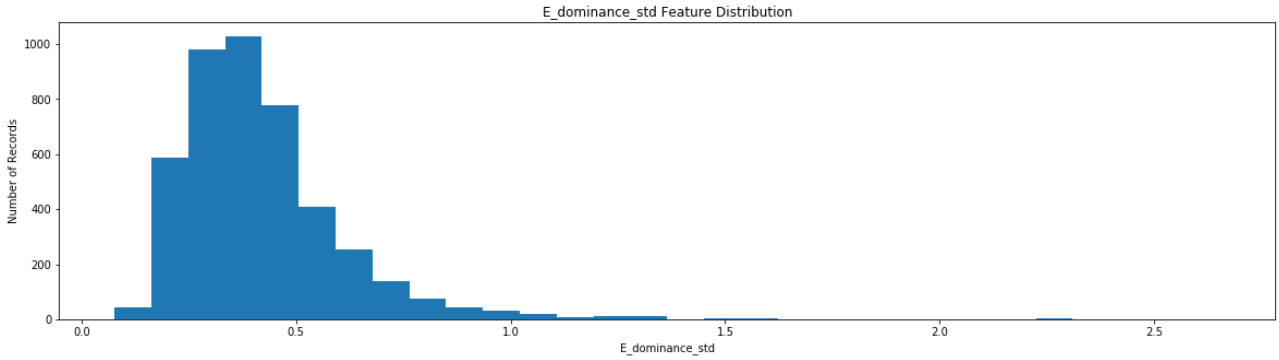

D#_dominance_mean Feature Distribution
Number of Records
D#_dominance_mean

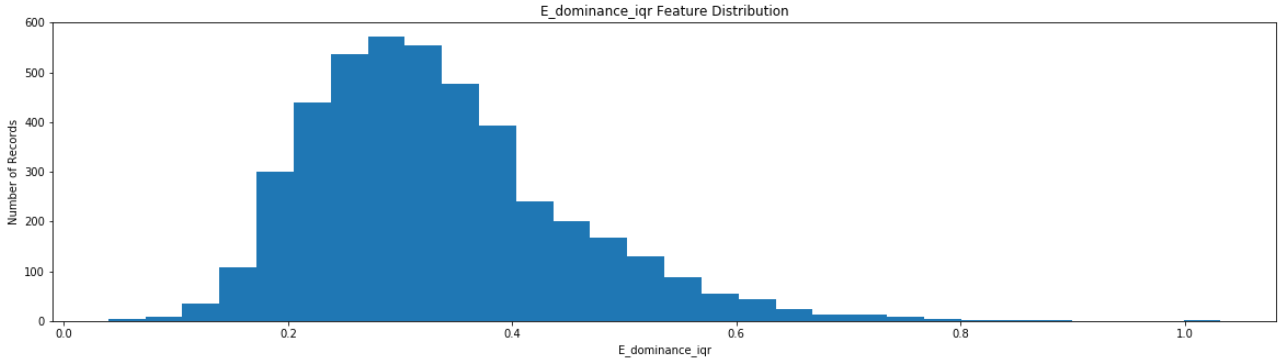

D#_dominance_std Feature Distribution
Number of Records
D#_dominance_std

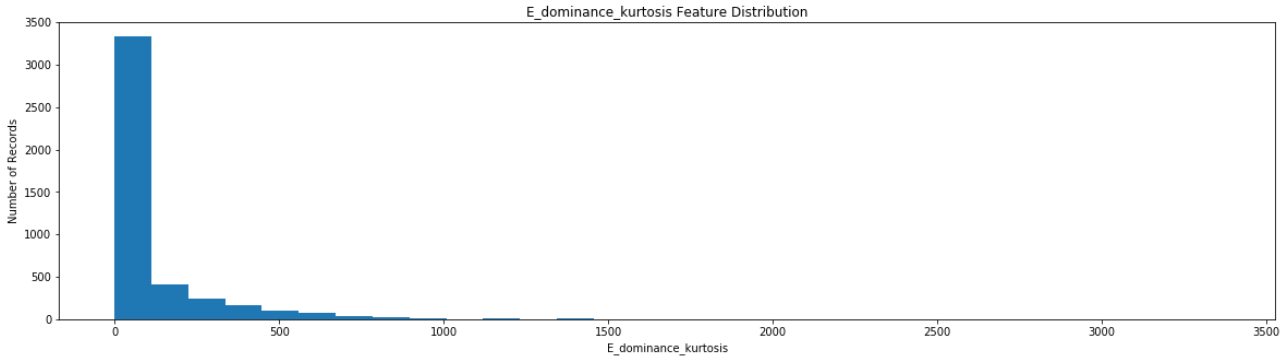

D#_dominance_iqr Feature Distribution
Number of Records
D#_dominance_iqr

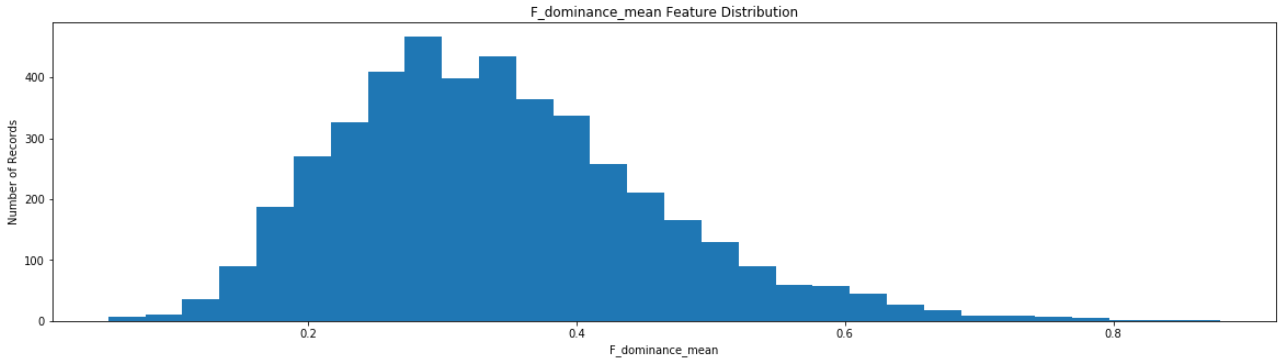

D#_dominance_kurtosis Feature Distribution
Number of Records
D#_dominance_kurtosis

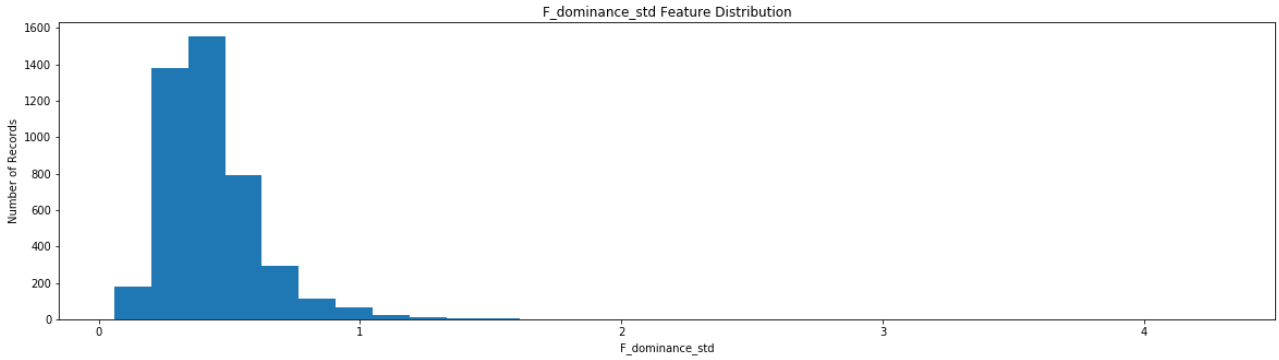

E_dominance_mean Feature Distribution
Number of Records
E_dominance_mean

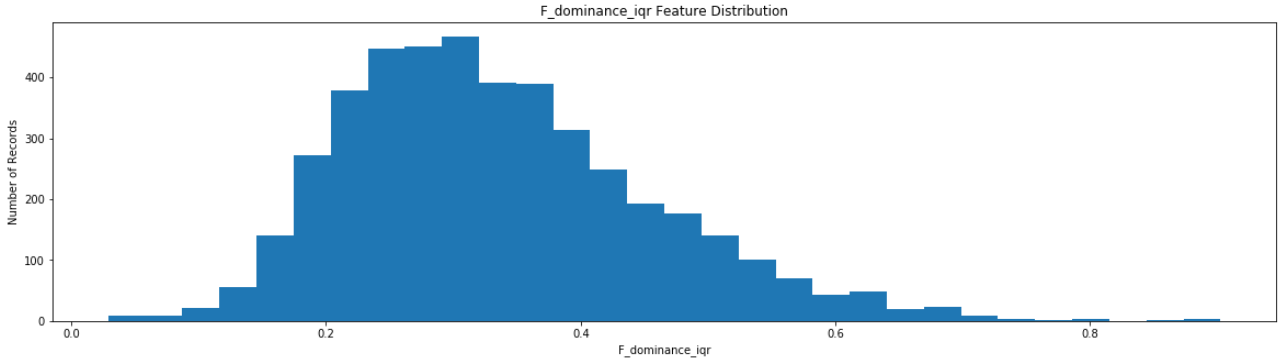

E_dominance_std Feature Distribution
Number of Records
E_dominance_std

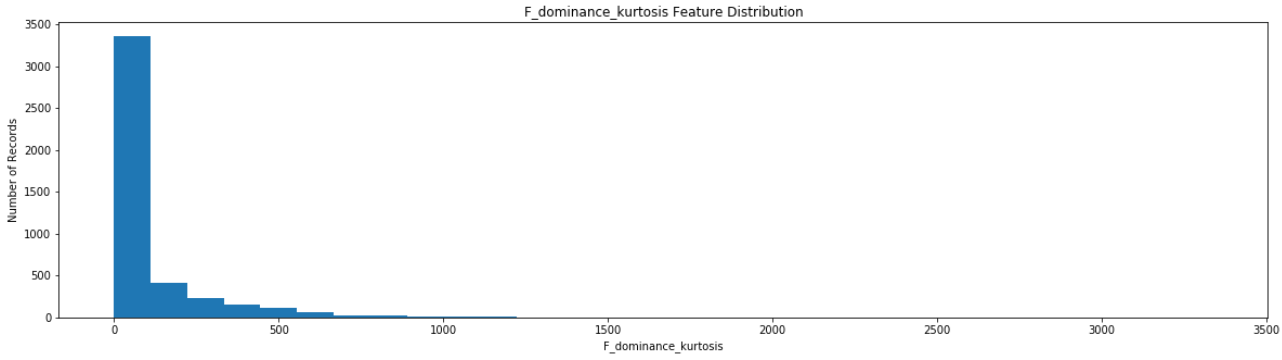

E_dominance_iqr Feature Distribution
Number of Records
E_dominance_iqr

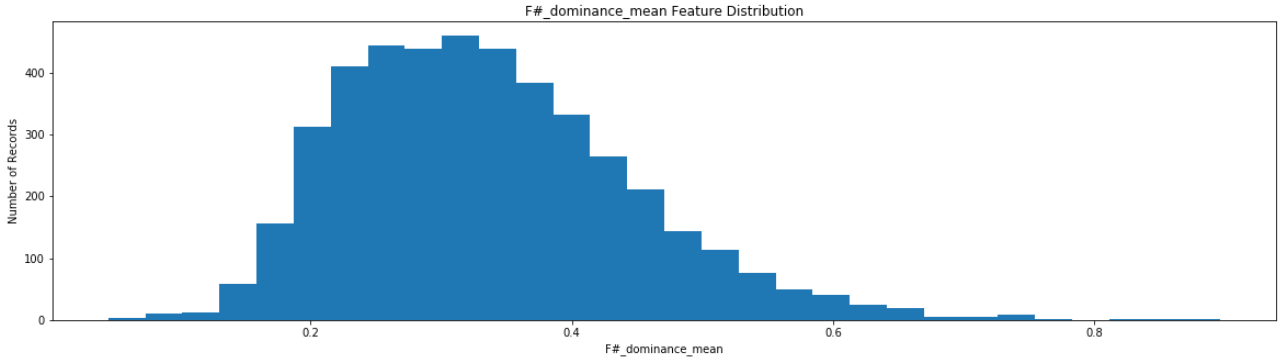

E_dominance_kurtosis Feature Distribution
Number of Records
E_dominance_kurtosis

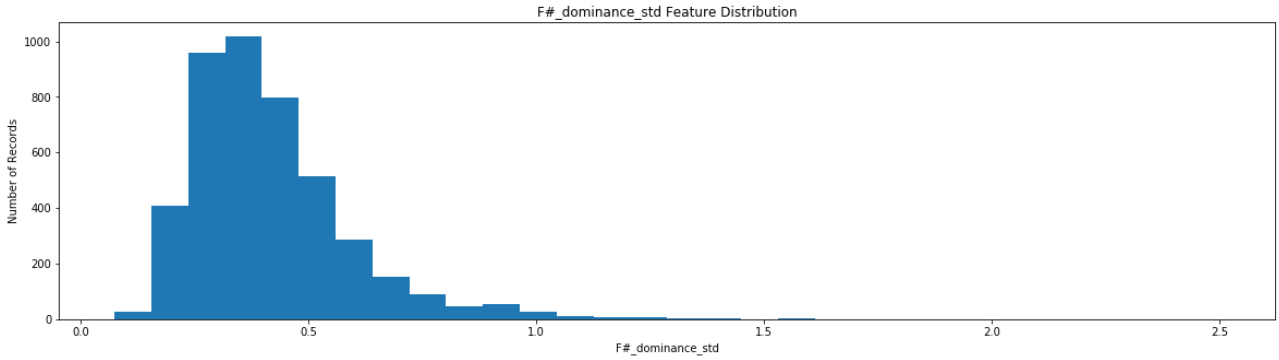

F_dominance_mean Feature Distribution
Number of Records
F_dominance_mean

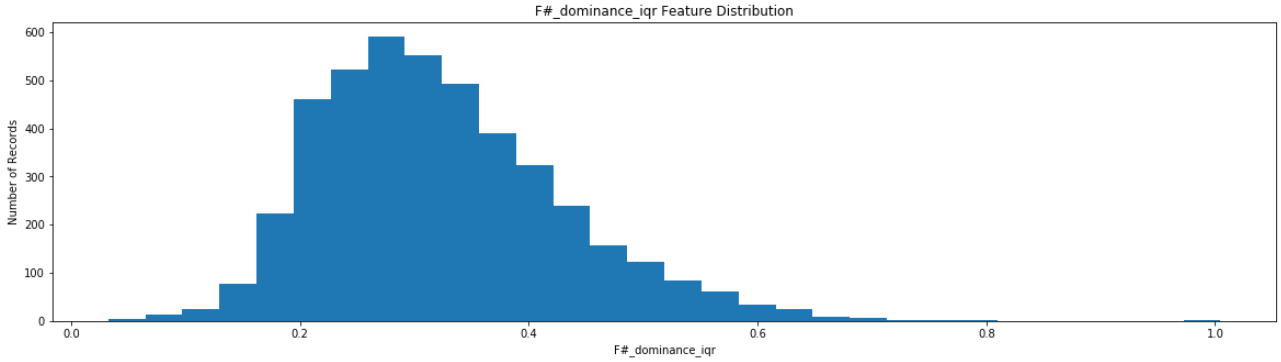

F_dominance_std Feature Distribution
Number of Records
F_dominance_std

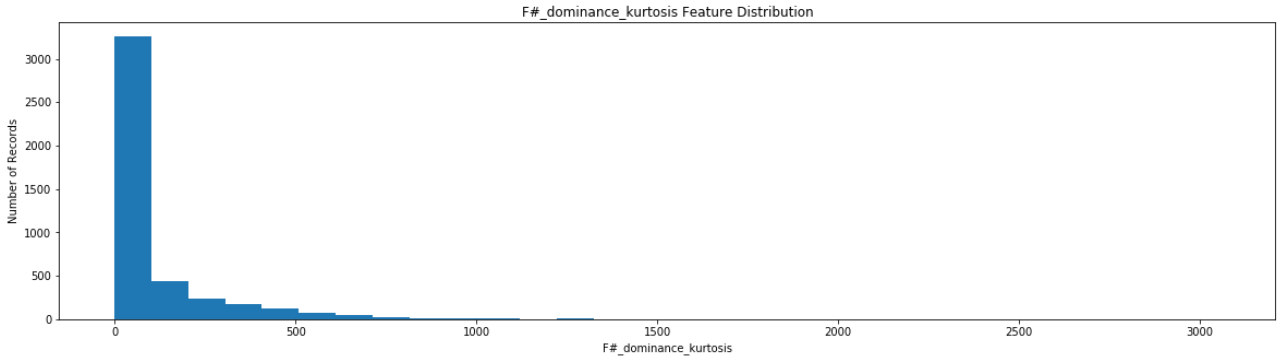

F_dominance_iqr Feature Distribution
Number of Records
F_dominance_iqr

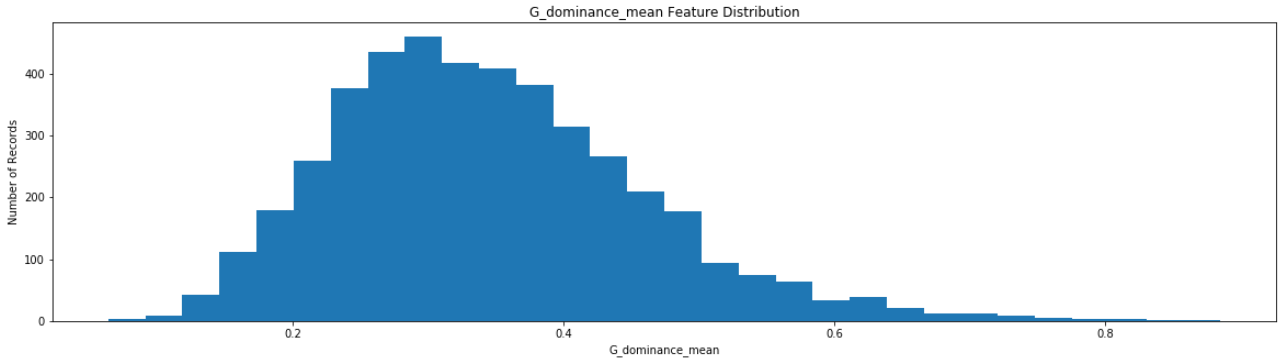

F_dominance_kurtosis Feature Distribution
Number of Records
F_dominance_kurtosis

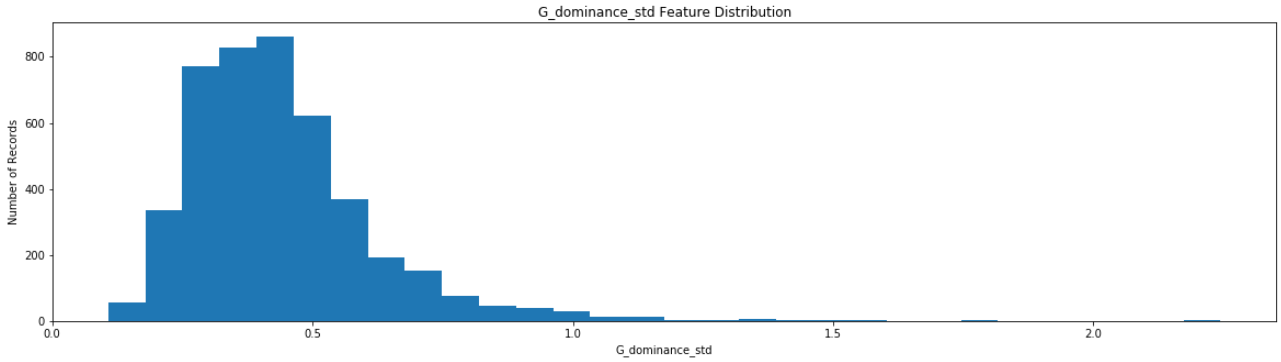

F#_dominance_mean Feature Distribution
Number of Records
F#_dominance_mean

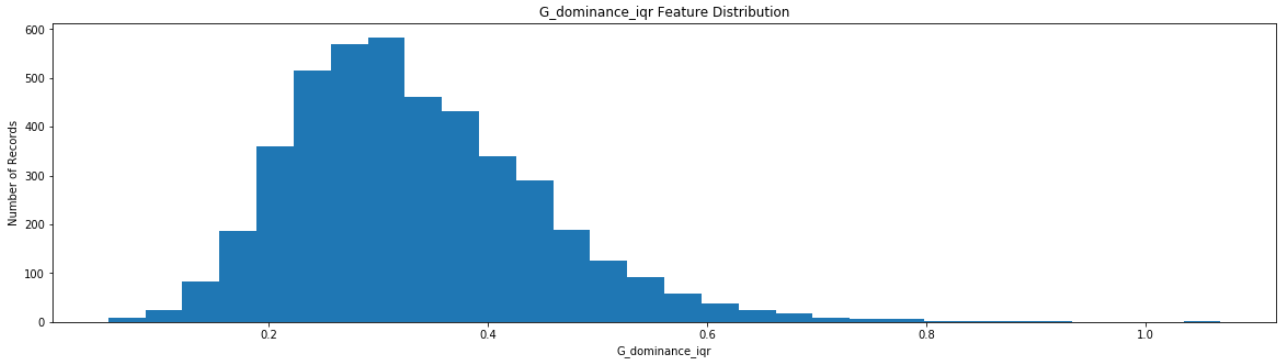

F#_dominance_std Feature Distribution
Number of Records
F#_dominance_std

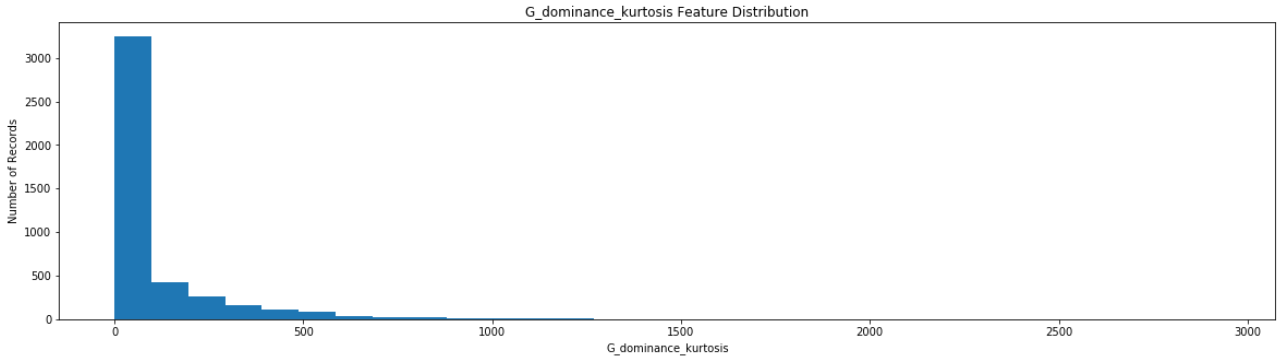

F#_dominance_iqr Feature Distribution
Number of Records
F#_dominance_iqr

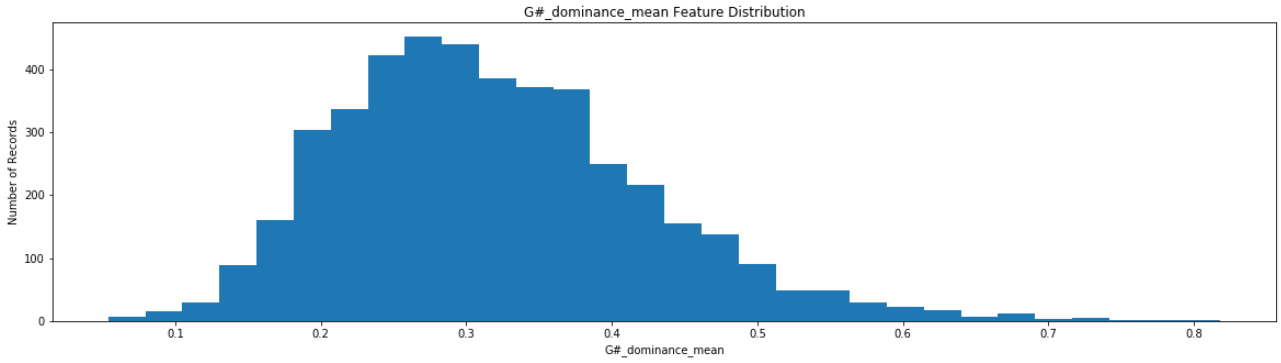

F#_dominance_kurtosis Feature Distribution
Number of Records
F#_dominance_kurtosis

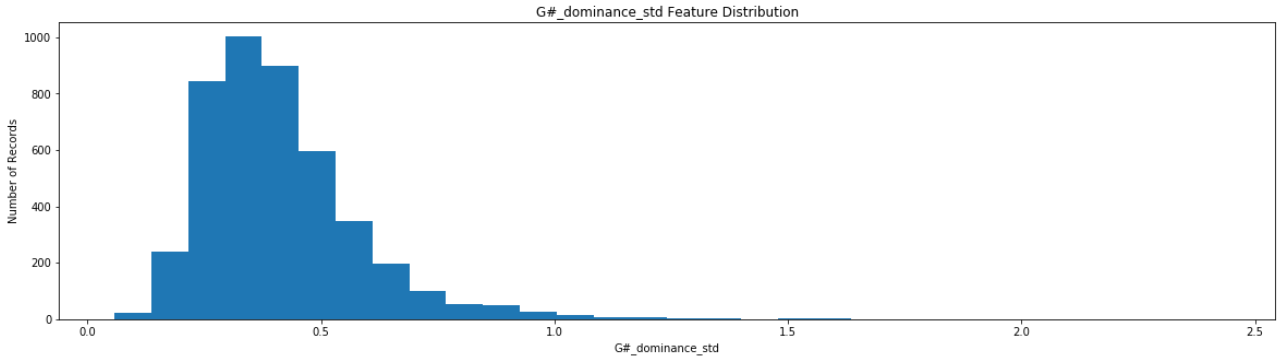

G_dominance_mean Feature Distribution
Number of Records
G_dominance_mean

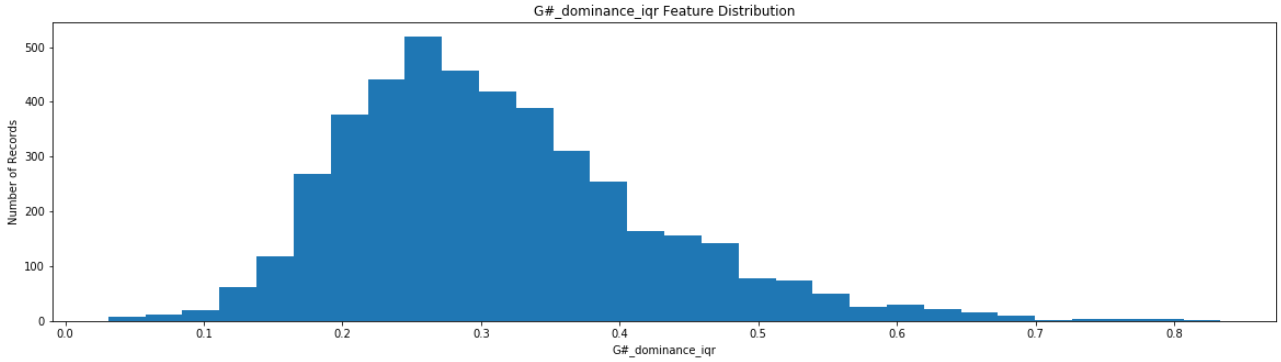

G_dominance_std Feature Distribution
Number of Records
G_dominance_std

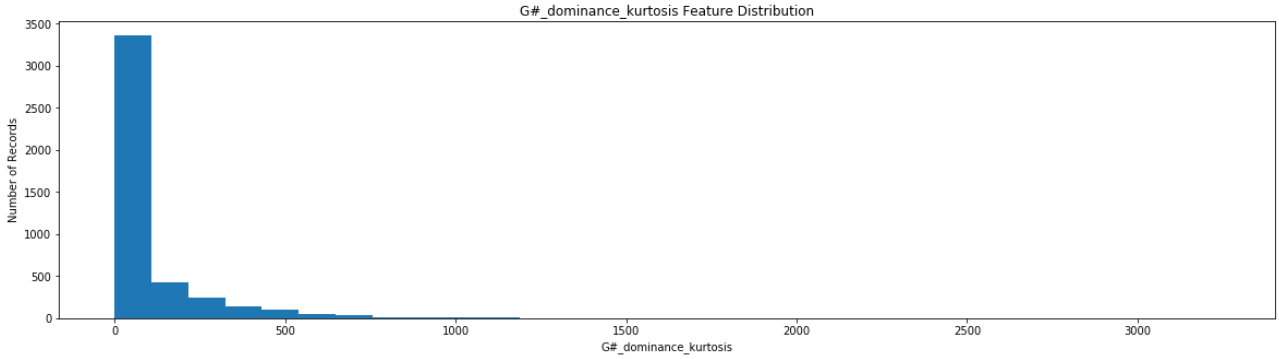

G_dominance_iqr Feature Distribution
Number of Records
G_dominance_iqr

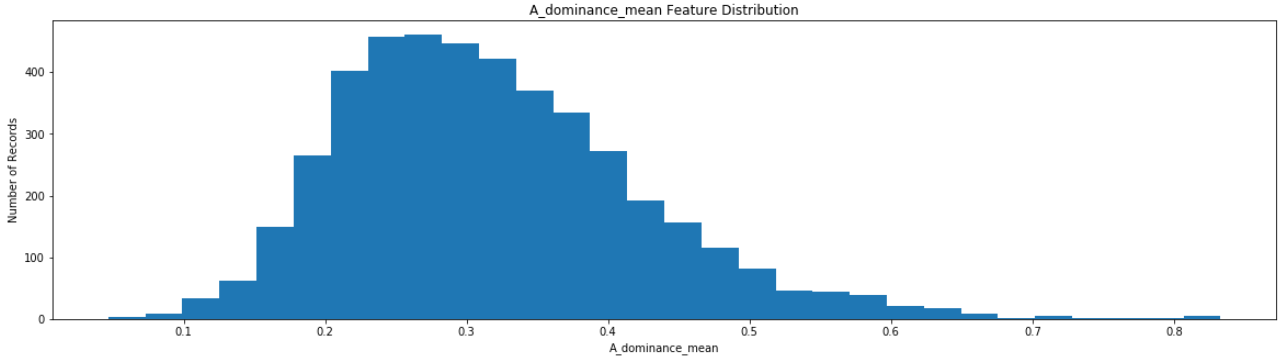

G_dominance_kurtosis Feature Distribution
Number of Records
G_dominance_kurtosis

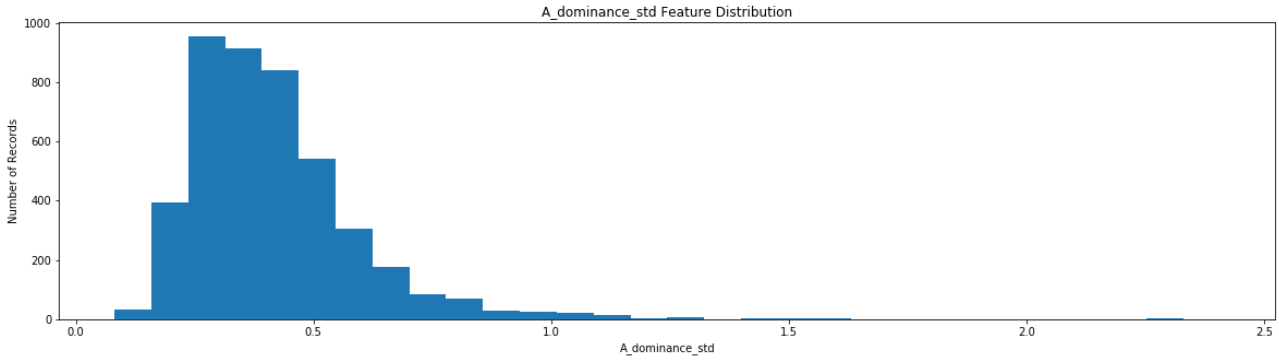

G#_dominance_mean Feature Distribution
Number of Records
G#_dominance_mean

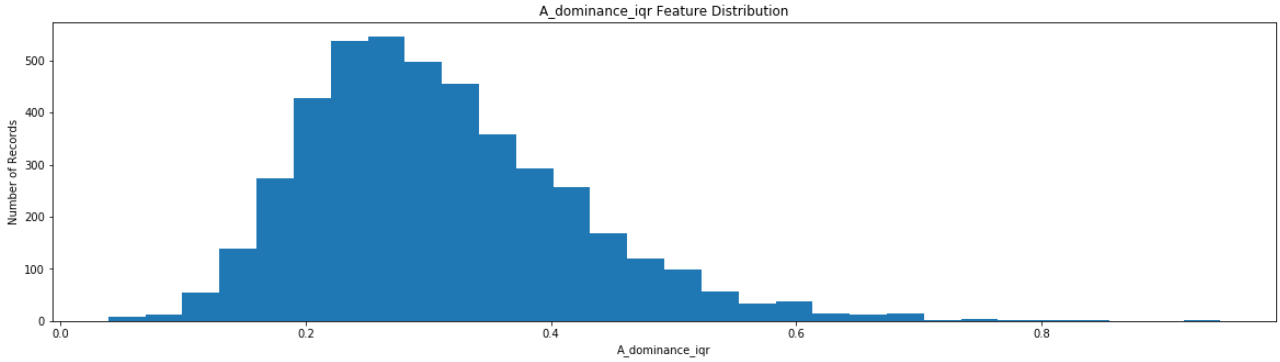

G#_dominance_std Feature Distribution
Number of Records
G#_dominance_std

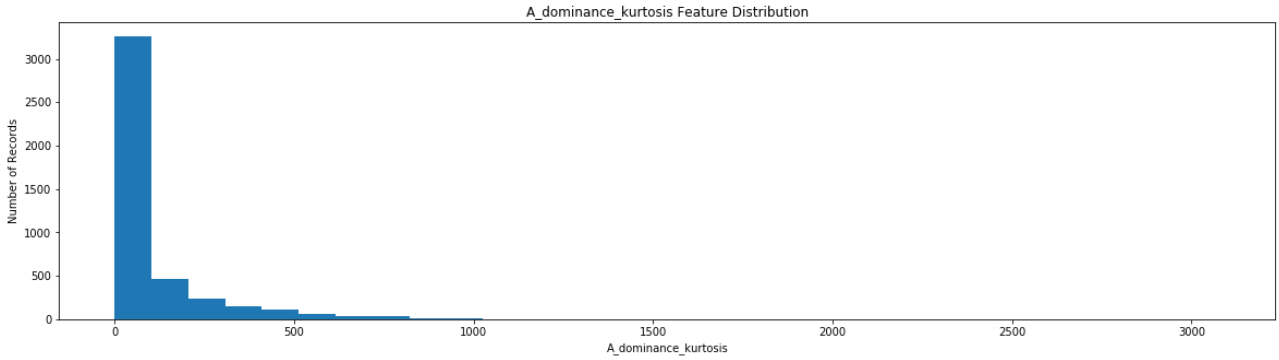

G#_dominance_iqr Feature Distribution
Number of Records
G#_dominance_iqr

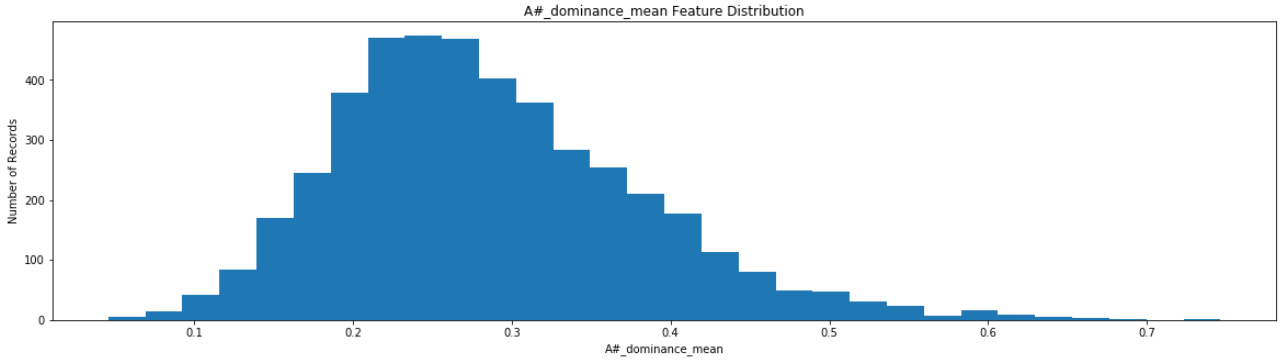

G#_dominance_kurtosis Feature Distribution
Number of Records
G#_dominance_kurtosis

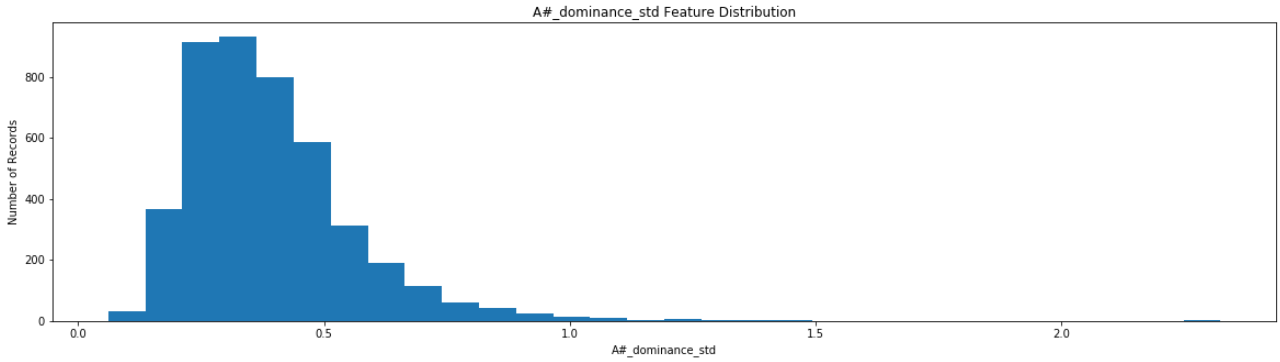

A_dominance_mean Feature Distribution
Number of Records
A_dominance_mean

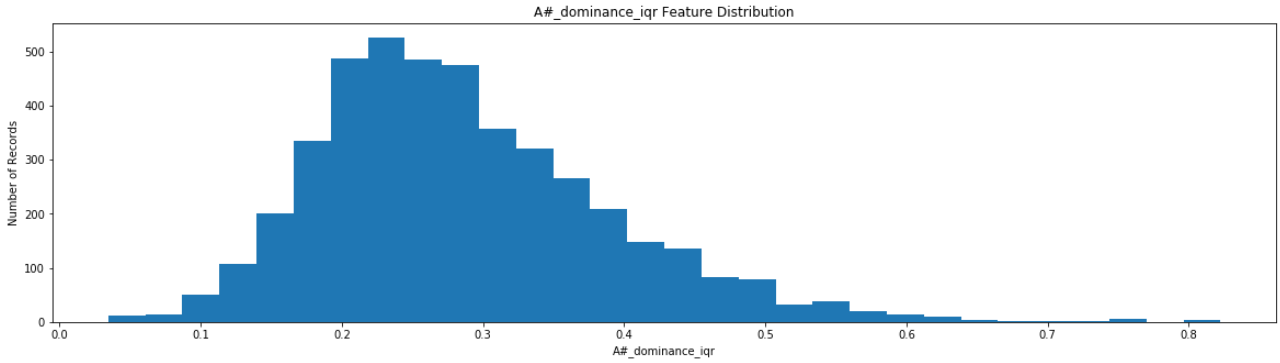

A_dominance_std Feature Distribution
Number of Records
A_dominance_std

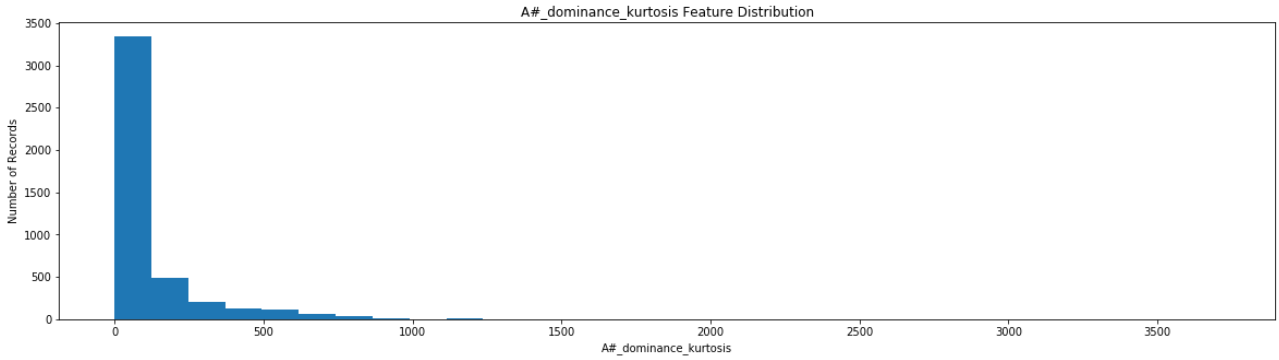

A_dominance_iqr Feature Distribution
Number of Records
A_dominance_iqr

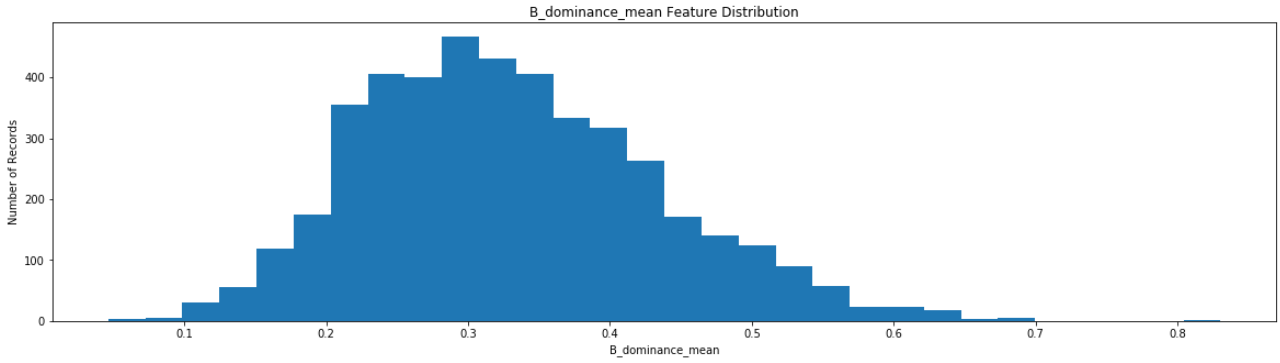

A_dominance_kurtosis Feature Distribution
Number of Records
A_dominance_kurtosis

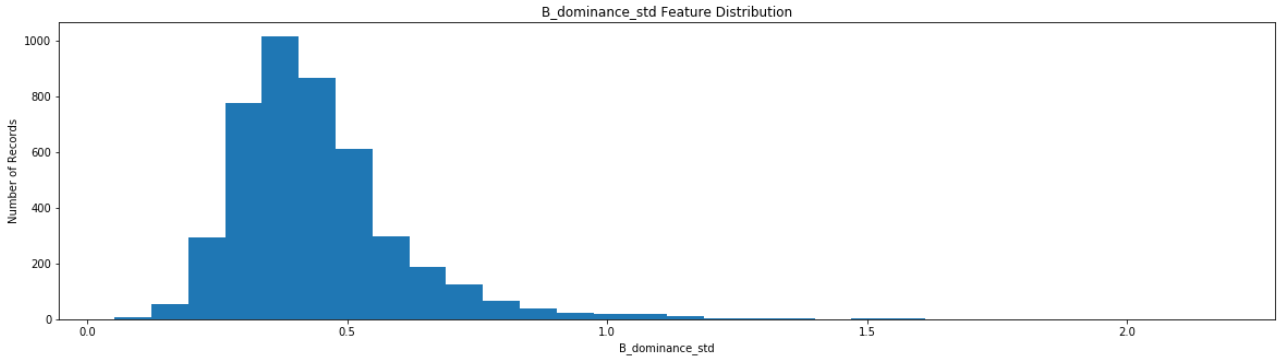

A#_dominance_mean Feature Distribution
Number of Records
A#_dominance_mean

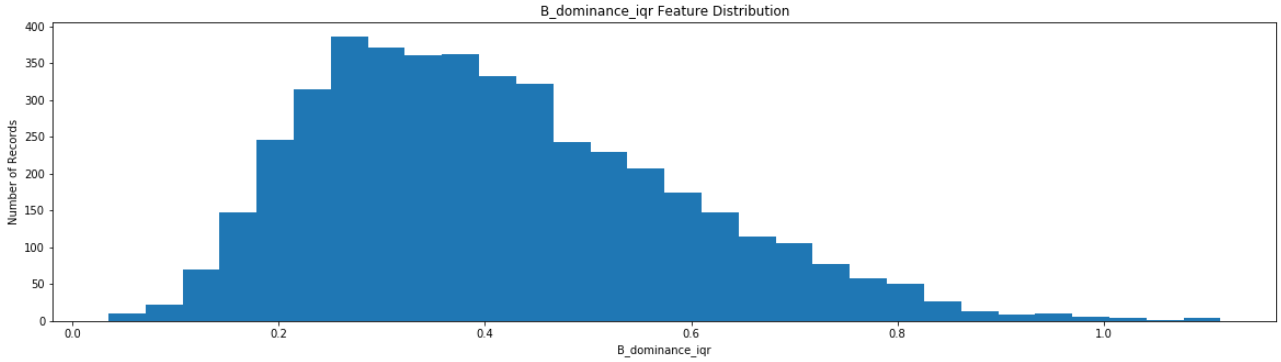

A#_dominance_std Feature Distribution
Number of Records
A#_dominance_std

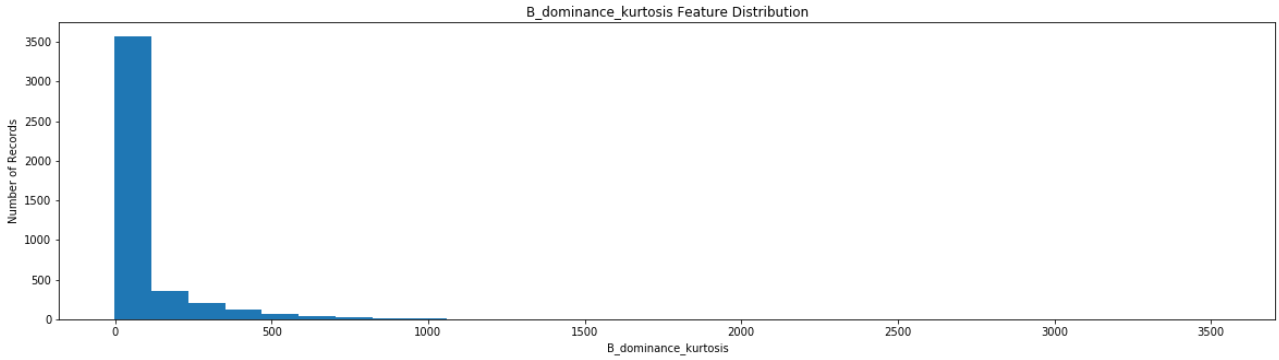

A#_dominance_iqr Feature Distribution
Number of Records
A#_dominance_iqr

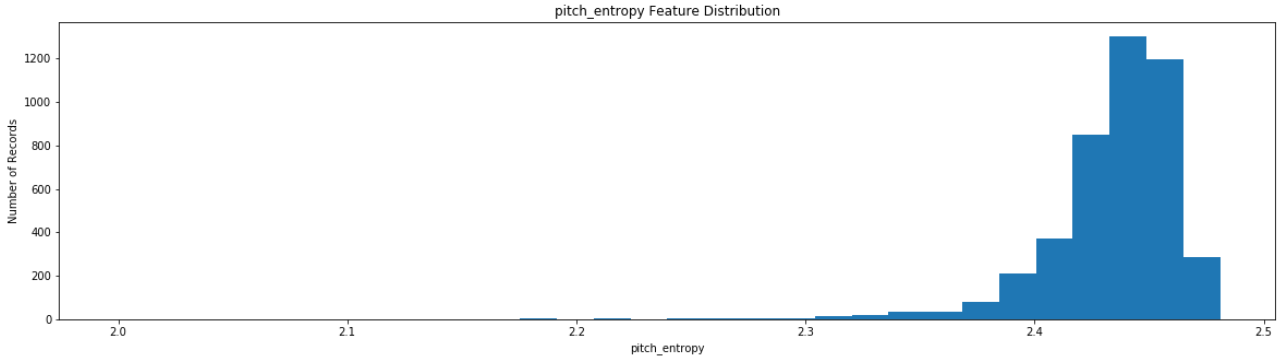

A#_dominance_kurtosis Feature Distribution
Number of Records
A#_dominance_kurtosis

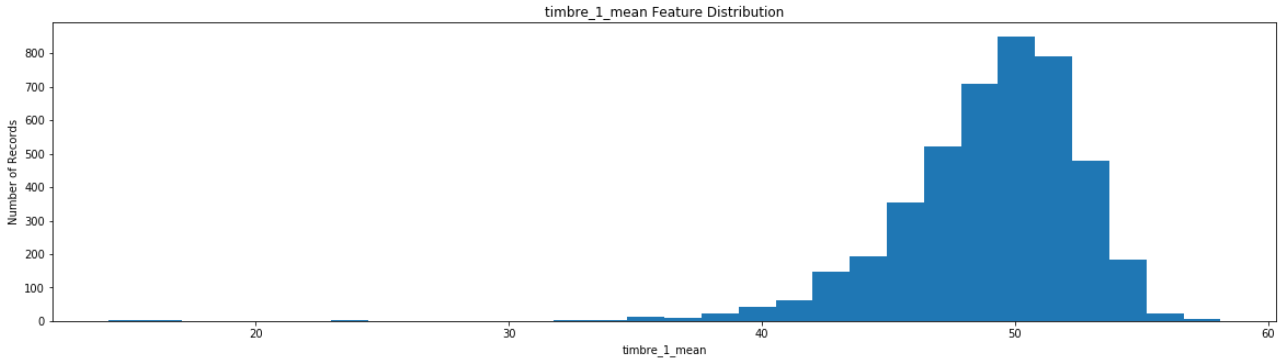

B_dominance_mean Feature Distribution
Number of Records
B_dominance_mean

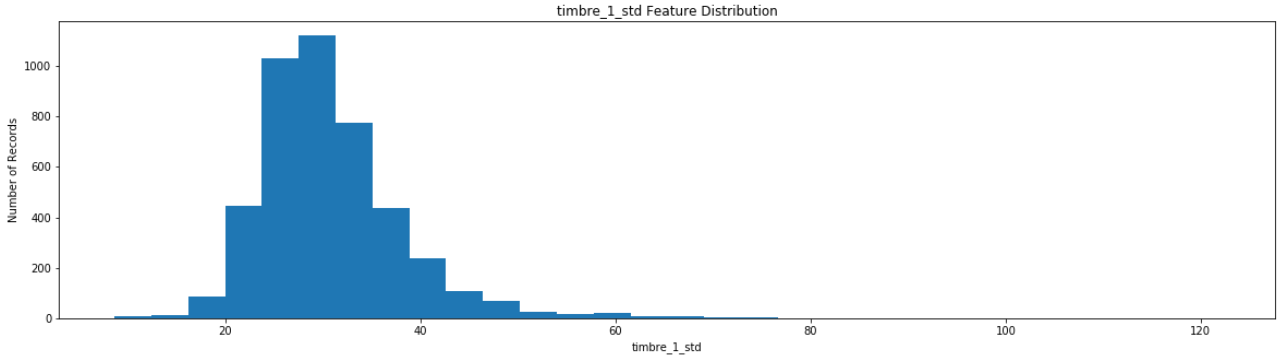

B_dominance_std Feature Distribution
Number of Records
B_dominance_std

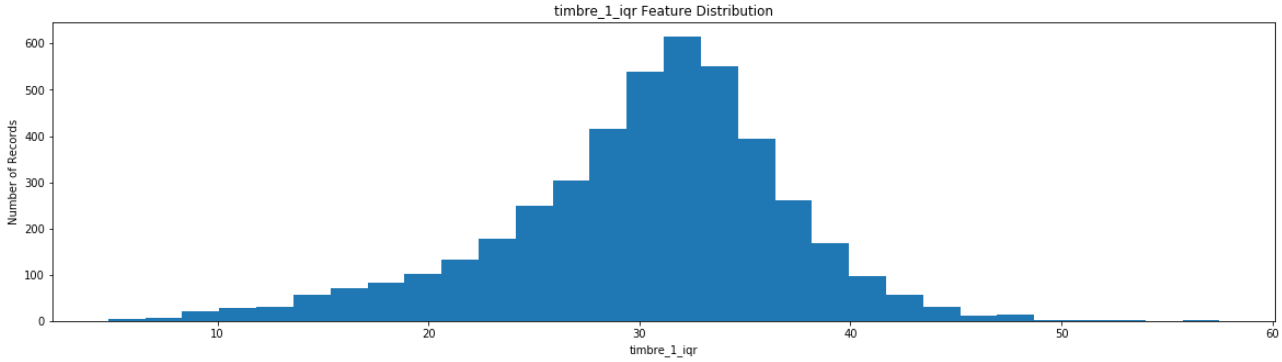

B_dominance_iqr Feature Distribution
Number of Records
B_dominance_iqr

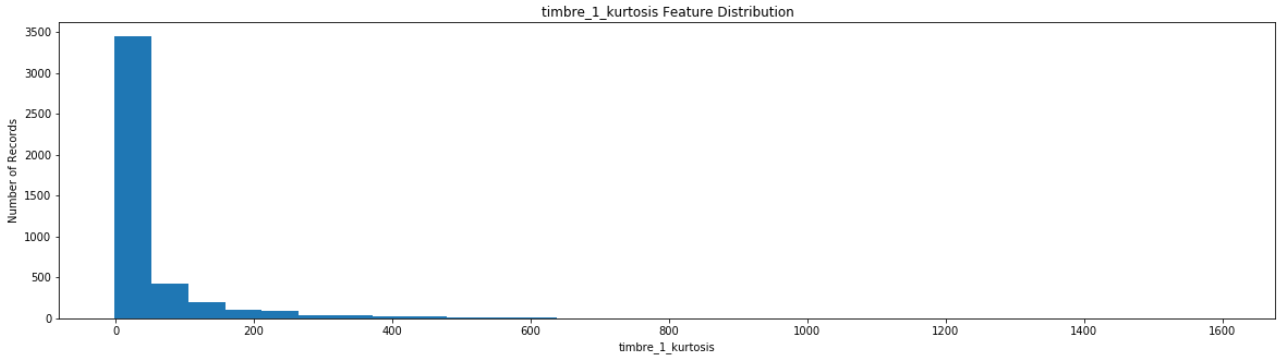

B_dominance_kurtosis Feature Distribution
Number of Records
B_dominance_kurtosis

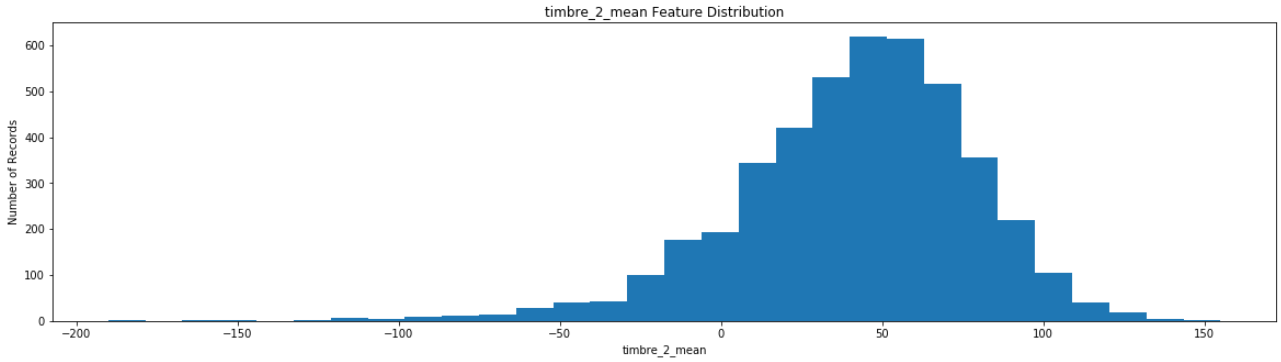

pitch_entropy Feature Distribution
Number of Records
pitch_entropy

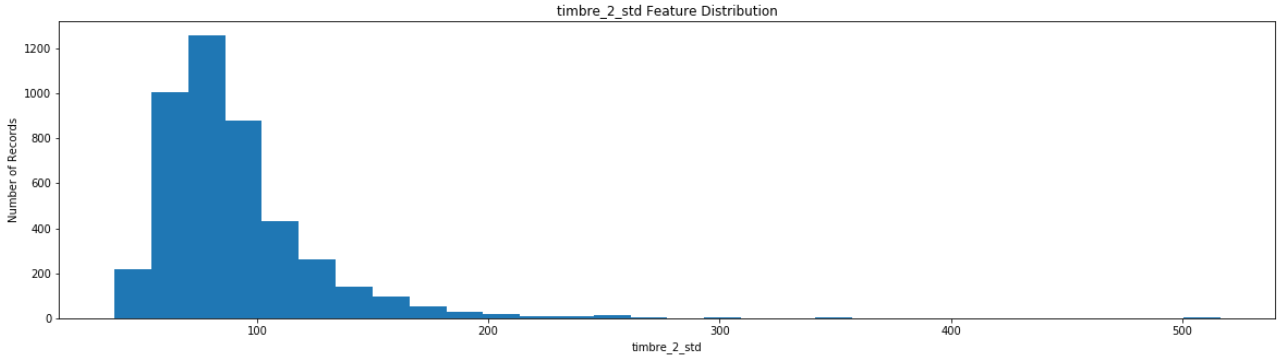

timbre_1_mean Feature Distribution
Number of Records
timbre_1_mean

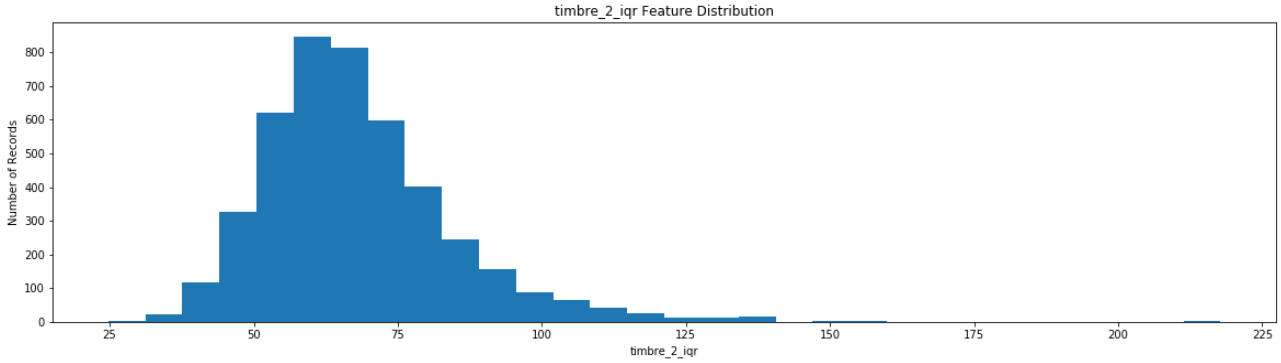

timbre_1_std Feature Distribution
Number of Records
timbre_1_std

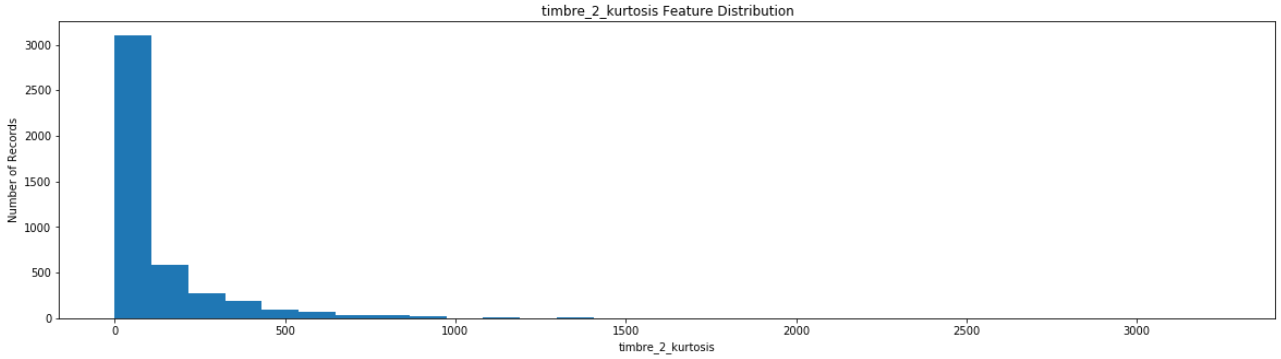

timbre_1_iqr Feature Distribution
Number of Records
timbre_1_iqr

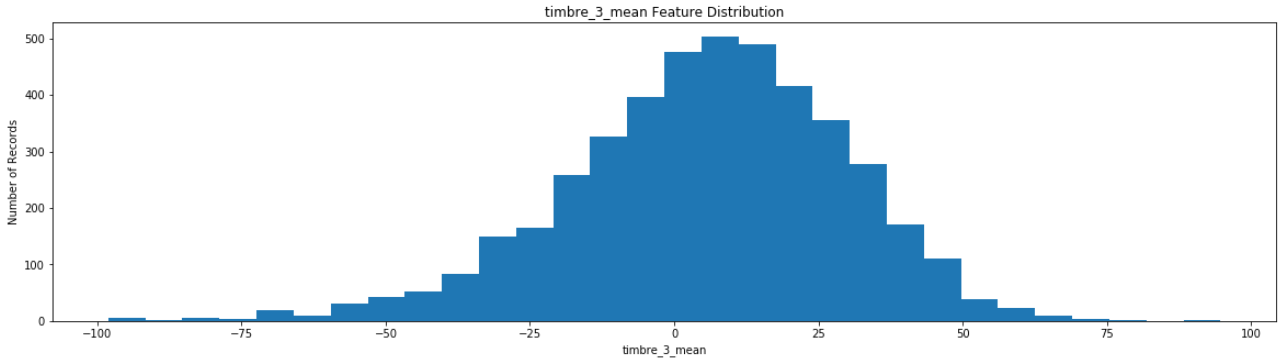

timbre_1_kurtosis Feature Distribution
Number of Records
timbre_1_kurtosis

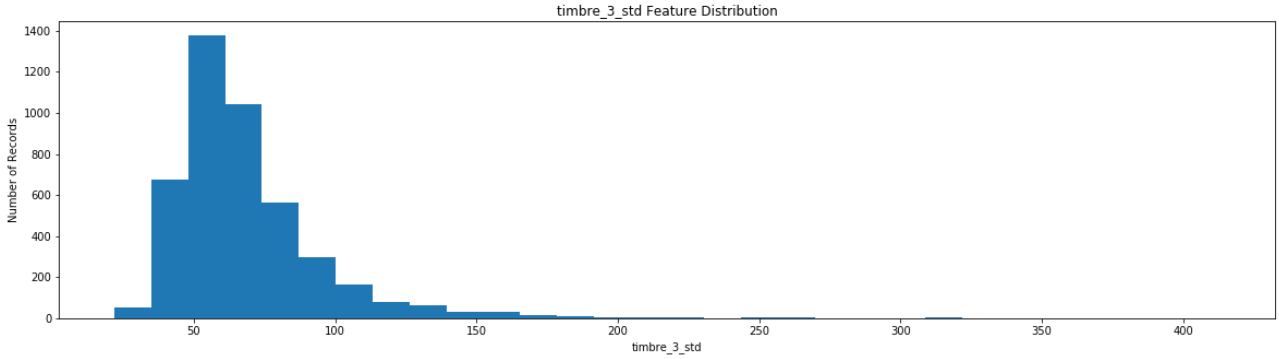

timbre_2_mean Feature Distribution
Number of Records
timbre_2_mean

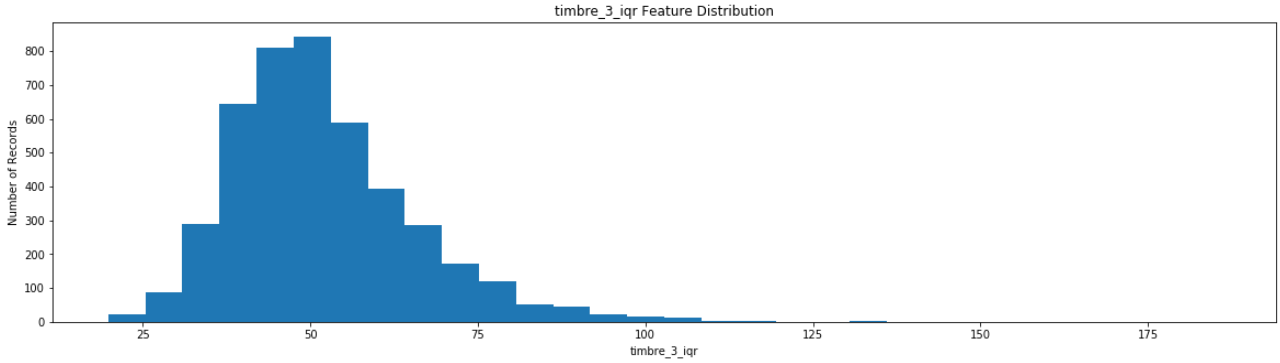

timbre_2_std Feature Distribution
Number of Records
timbre_2_std

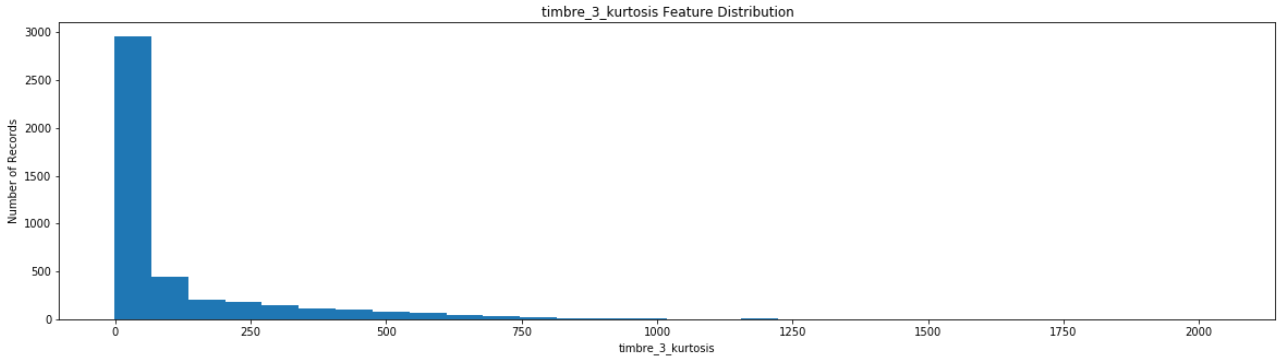

timbre_2_iqr Feature Distribution
Number of Records
timbre_2_iqr

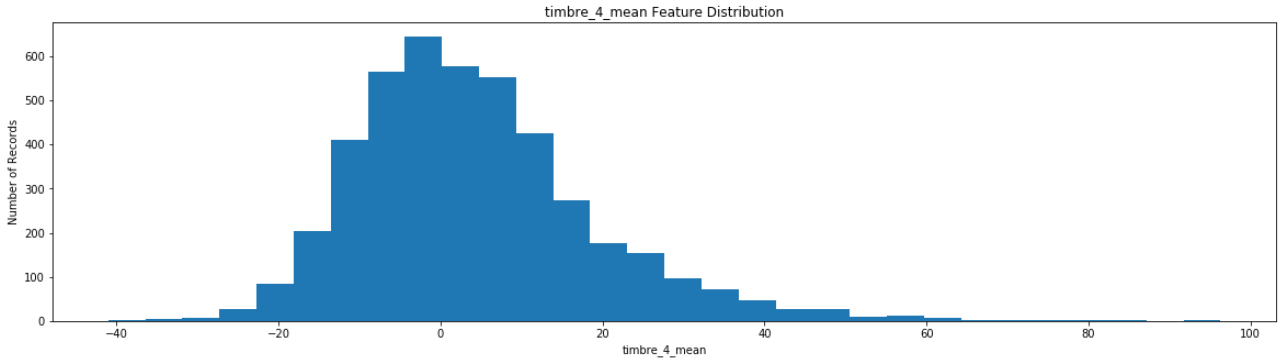

timbre_2_kurtosis Feature Distribution
Number of Records
timbre_2_kurtosis

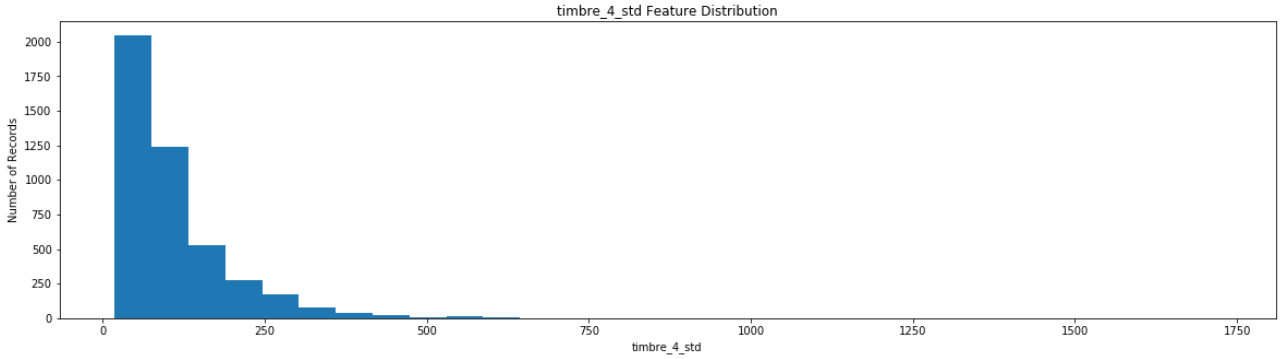

timbre_3_mean Feature Distribution
Number of Records
timbre_3_mean

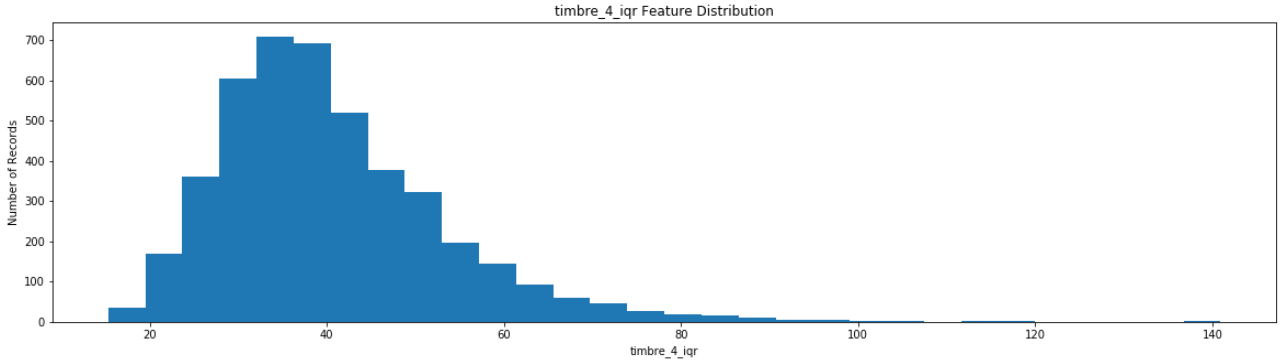

timbre_3_std Feature Distribution
Number of Records
timbre_3_std

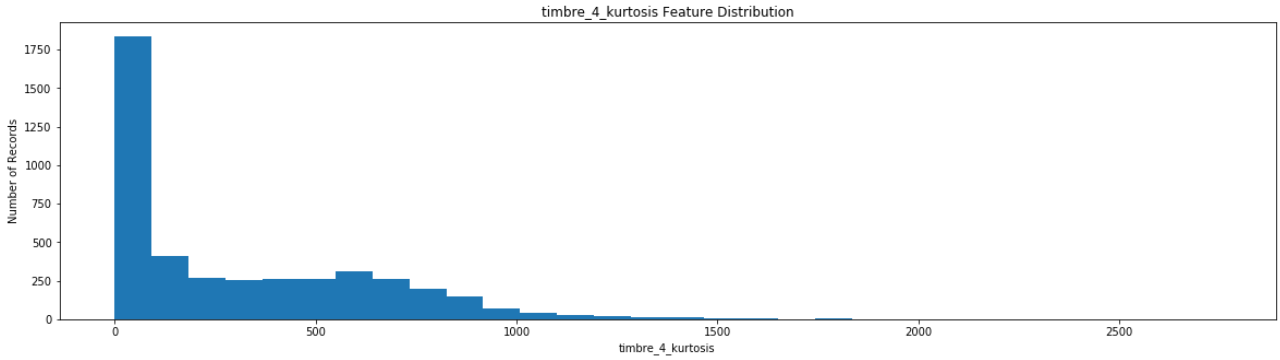

timbre_3_iqr Feature Distribution
Number of Records
timbre_3_iqr

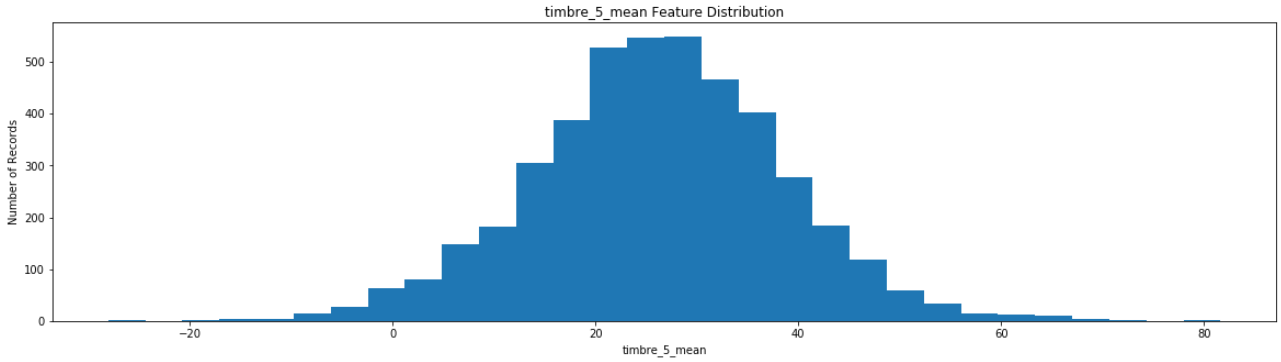

timbre_3_kurtosis Feature Distribution
Number of Records
timbre_3_kurtosis

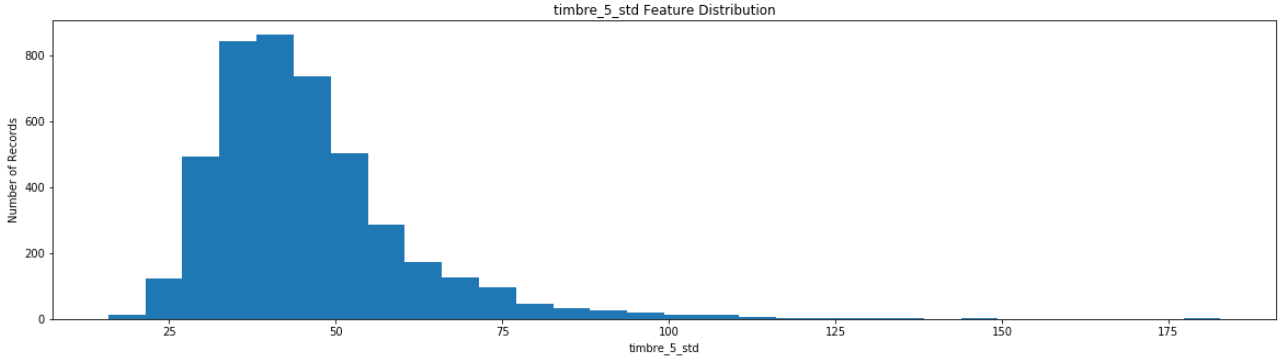

timbre_4_mean Feature Distribution
Number of Records
timbre_4_mean

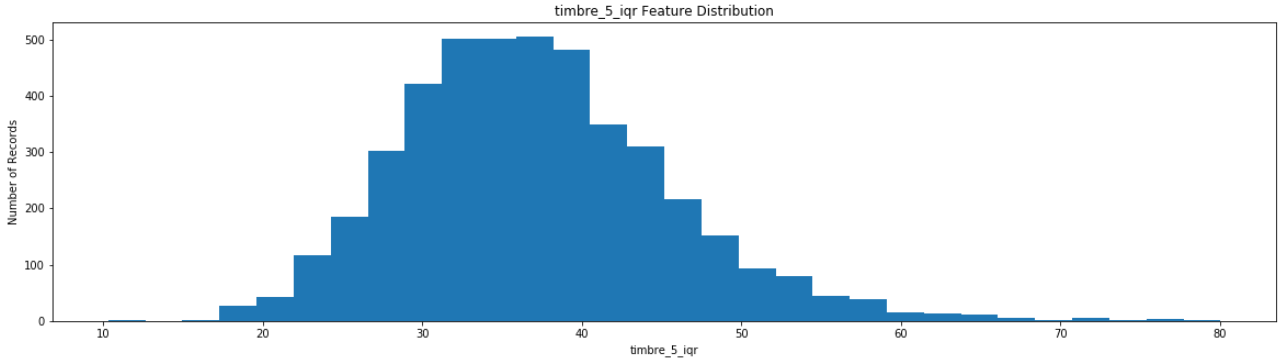

timbre_4_std Feature Distribution
Number of Records
timbre_4_std

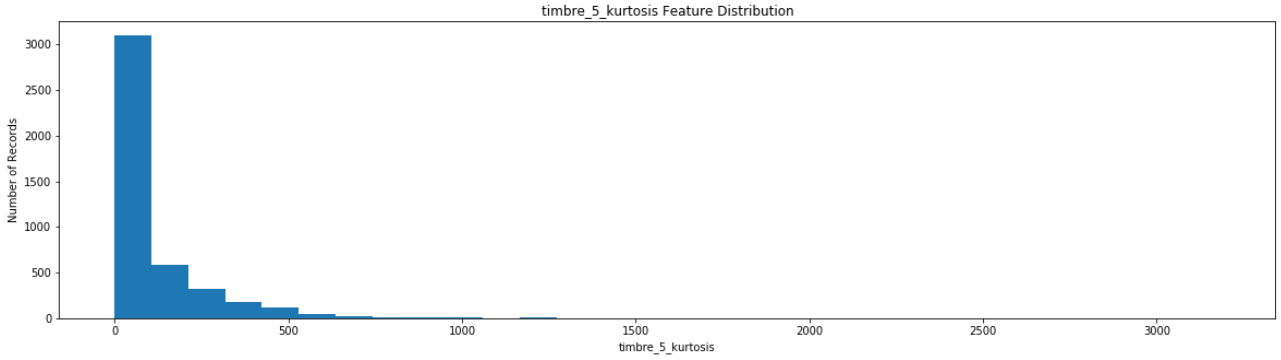

timbre_4_iqr Feature Distribution
Number of Records
timbre_4_iqr

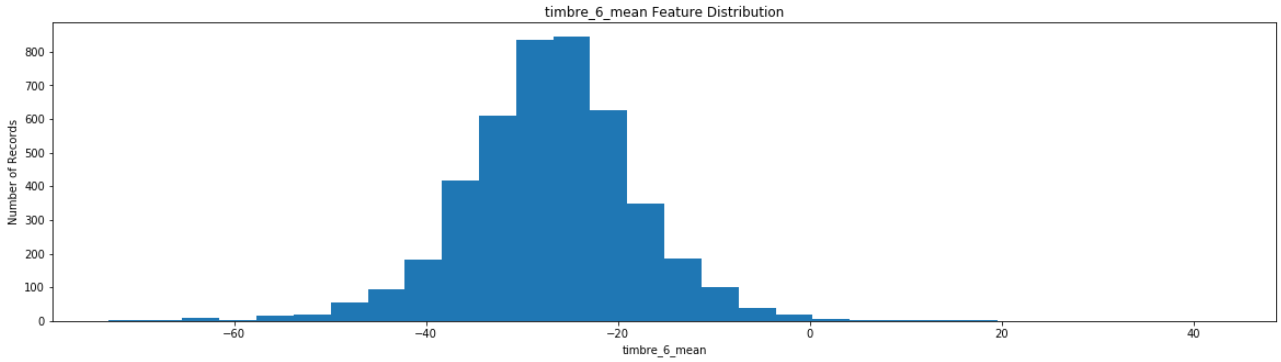

timbre_4_kurtosis Feature Distribution
Number of Records
timbre_4_kurtosis

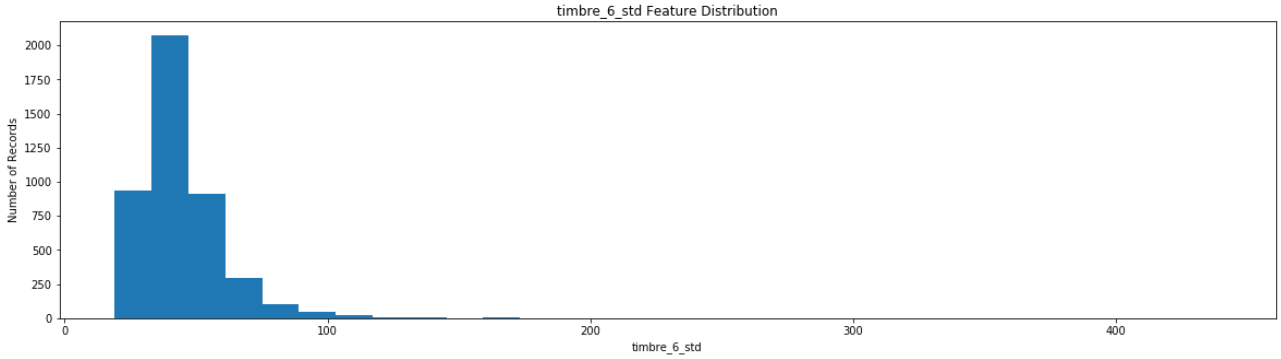

timbre_5_mean Feature Distribution
Number of Records
timbre_5_mean

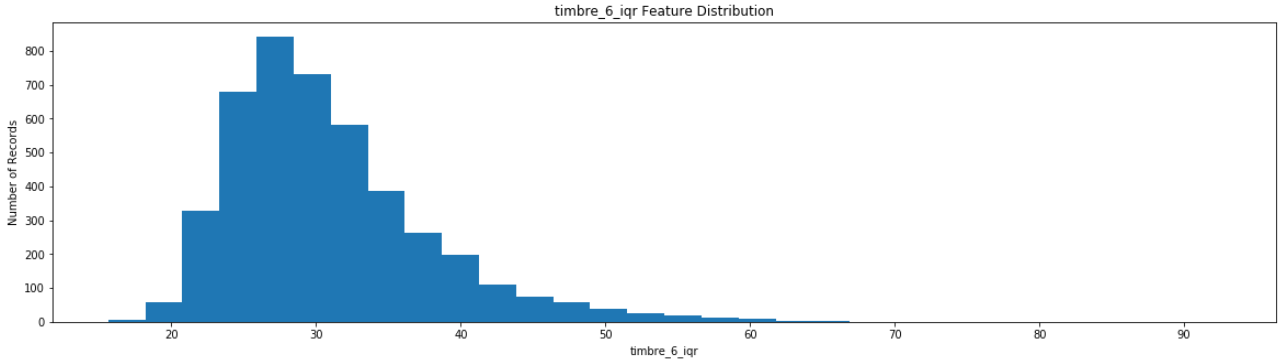

timbre_5_std Feature Distribution
Number of Records
timbre_5_std

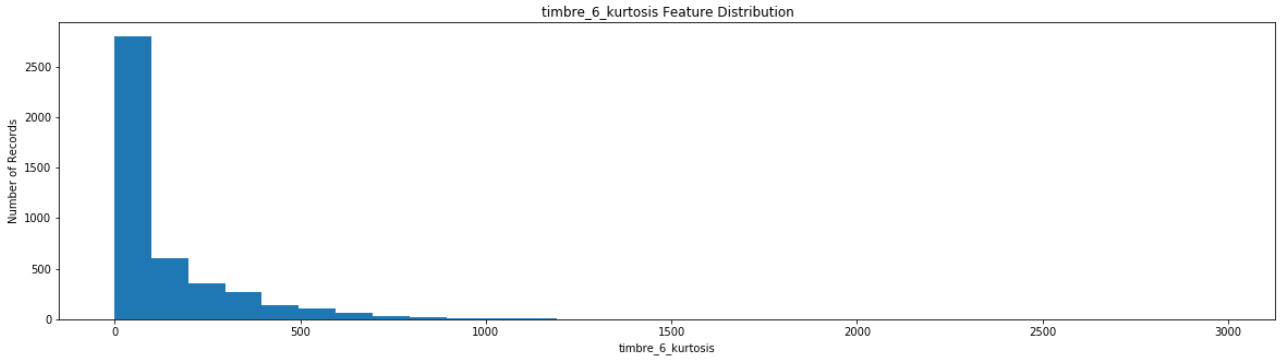

timbre_5_iqr Feature Distribution
Number of Records
timbre_5_iqr

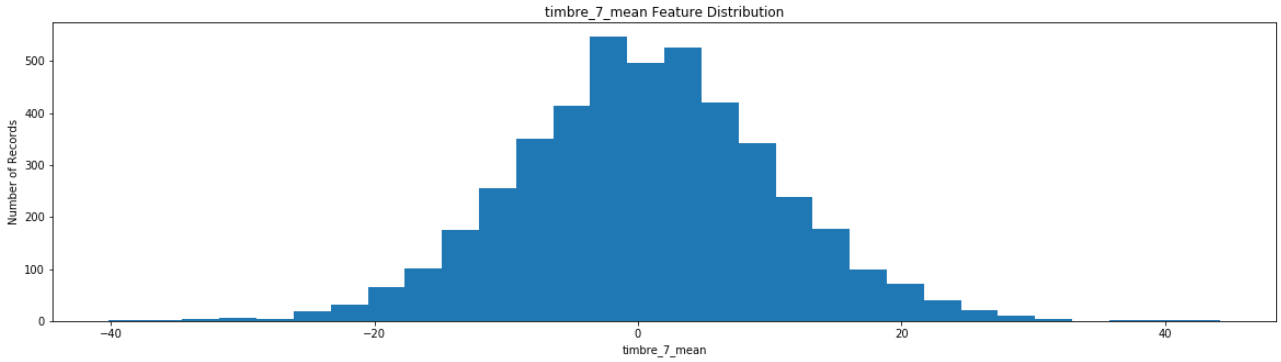

timbre_5_kurtosis Feature Distribution
Number of Records
timbre_5_kurtosis

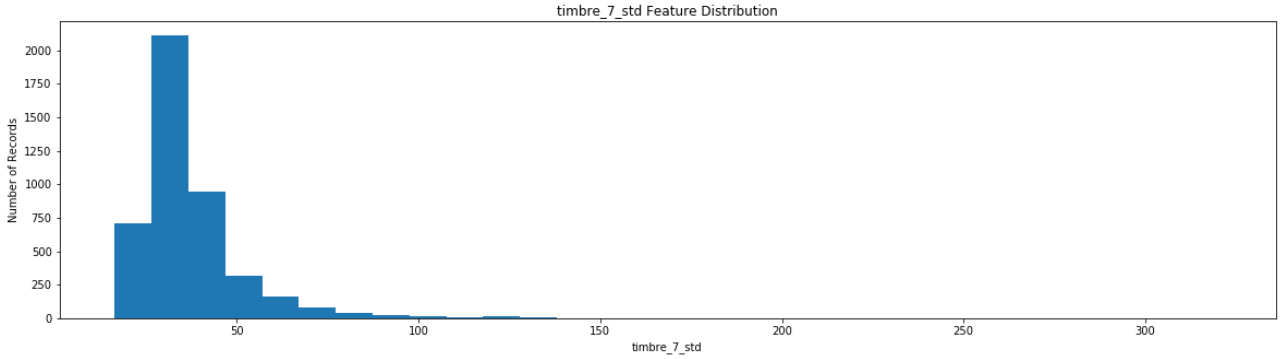

timbre_6_mean Feature Distribution
Number of Records
timbre_6_mean

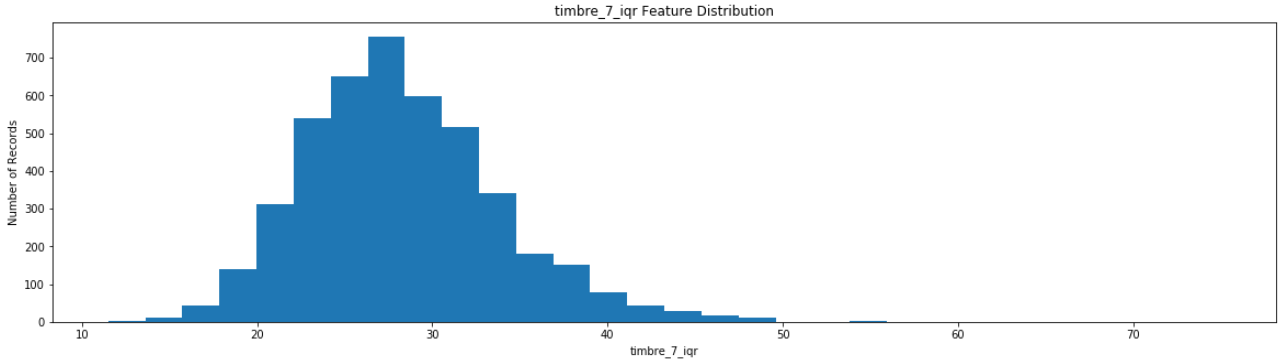

timbre_6_std Feature Distribution
Number of Records
timbre_6_std

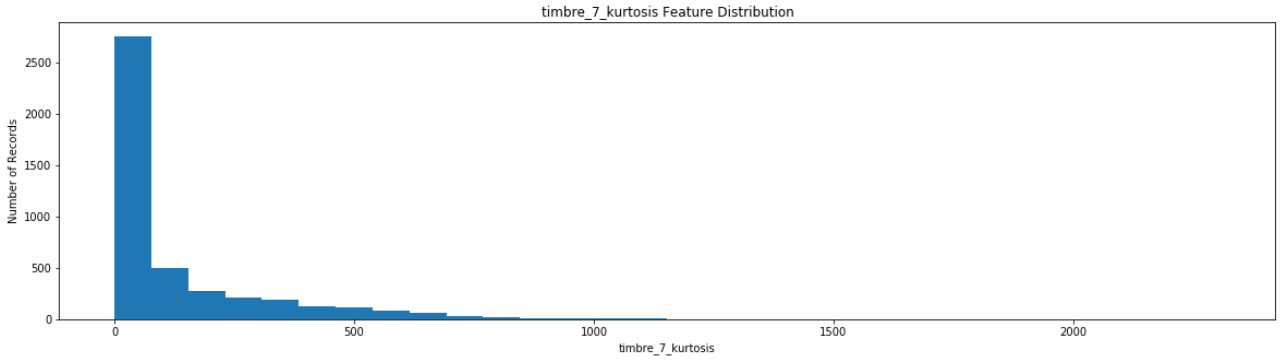

timbre_6_iqr Feature Distribution
Number of Records
timbre_6_iqr

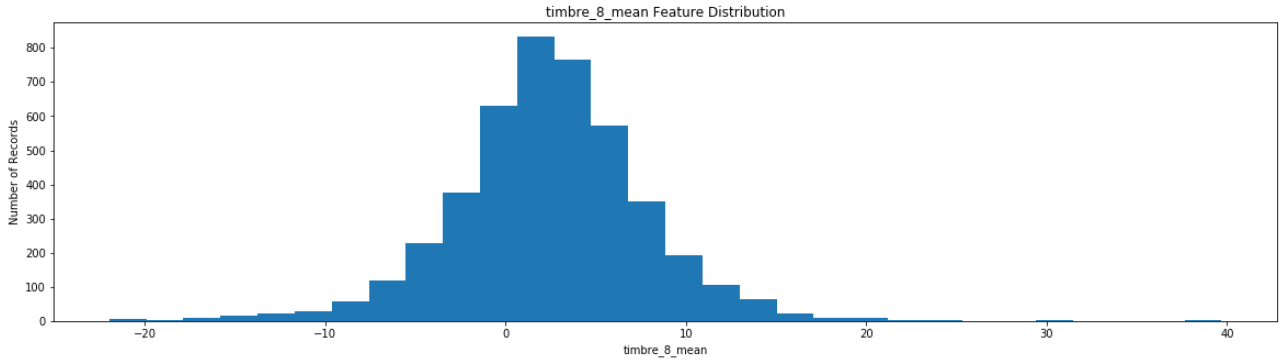

timbre_6_kurtosis Feature Distribution
Number of Records
timbre_6_kurtosis

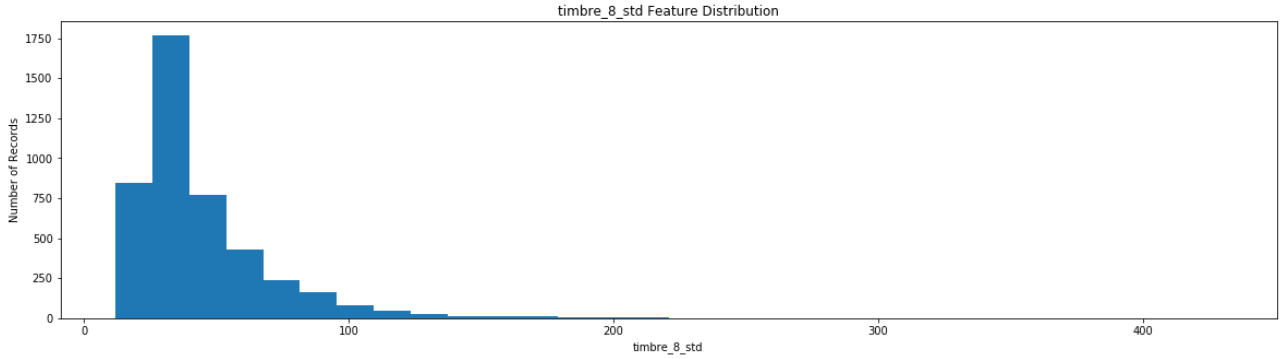

timbre_7_mean Feature Distribution
Number of Records
timbre_7_mean

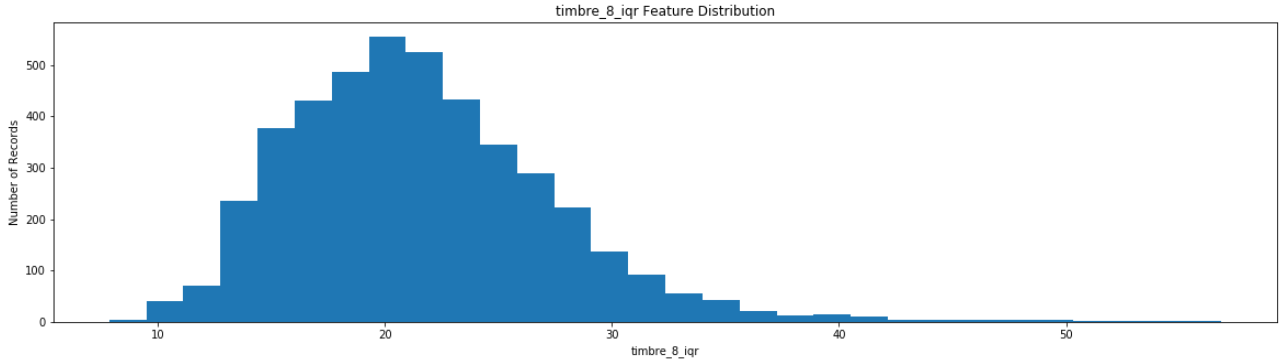

timbre_7_std Feature Distribution
Number of Records
timbre_7_std

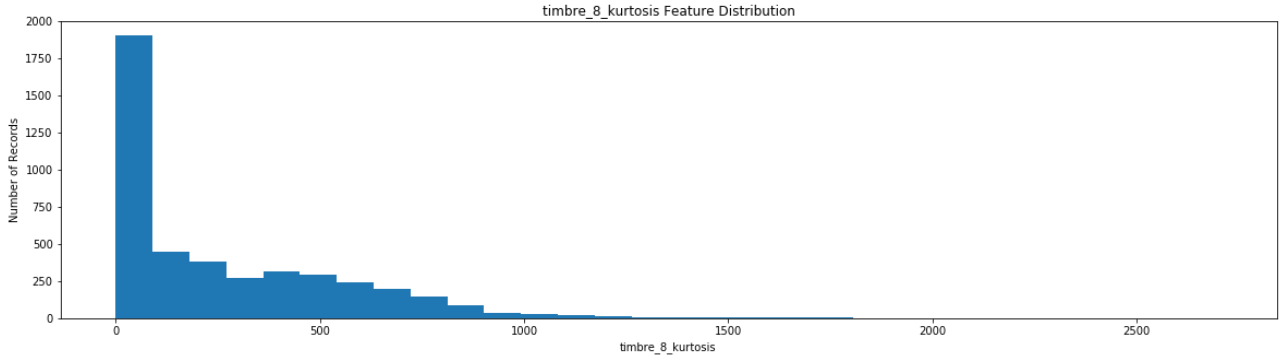

timbre_7_iqr Feature Distribution
Number of Records
timbre_7_iqr

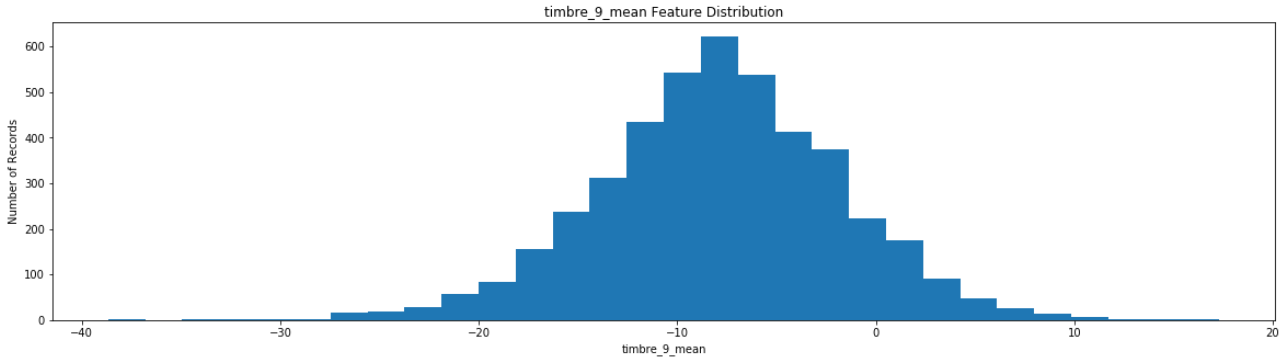

timbre_7_kurtosis Feature Distribution
Number of Records
timbre_7_kurtosis

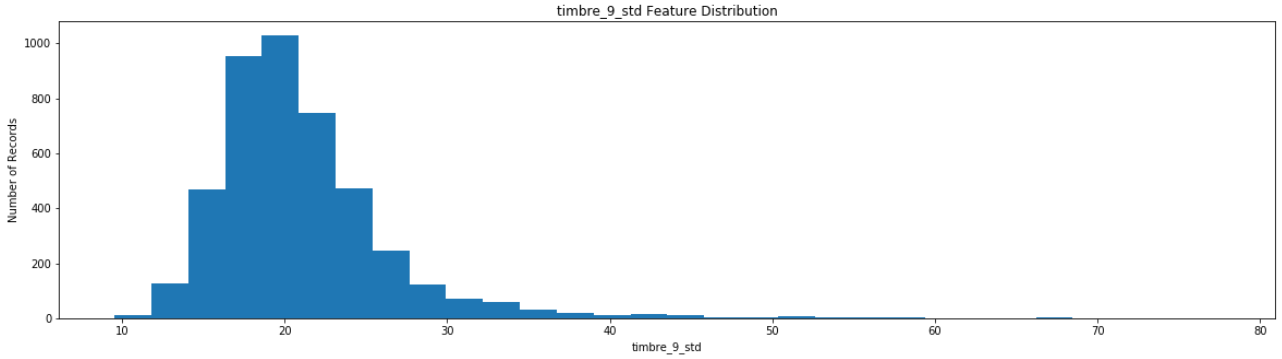

timbre_8_mean Feature Distribution
Number of Records
timbre_8_mean

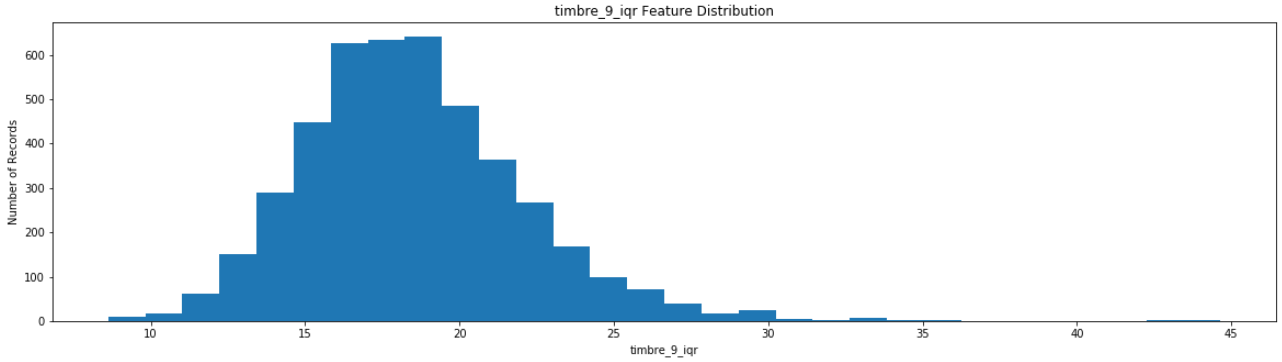

timbre_8_std Feature Distribution
Number of Records
timbre_8_std

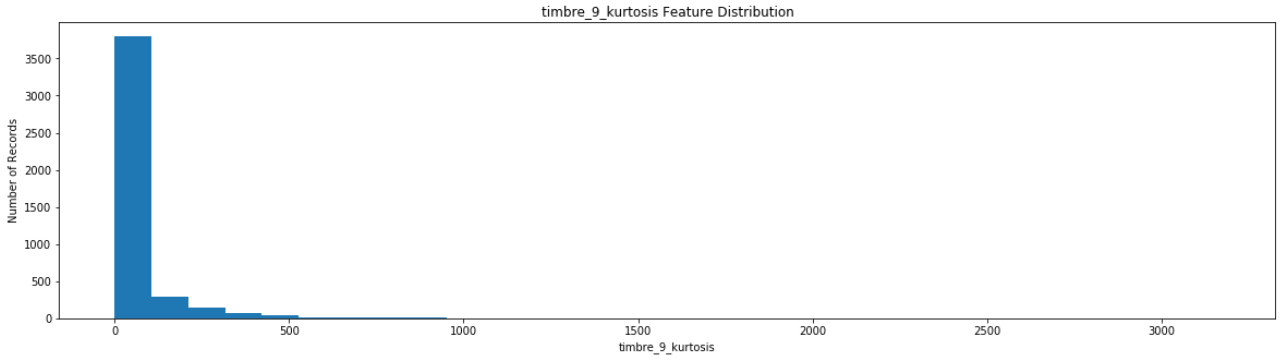

timbre_8_iqr Feature Distribution
Number of Records
timbre_8_iqr

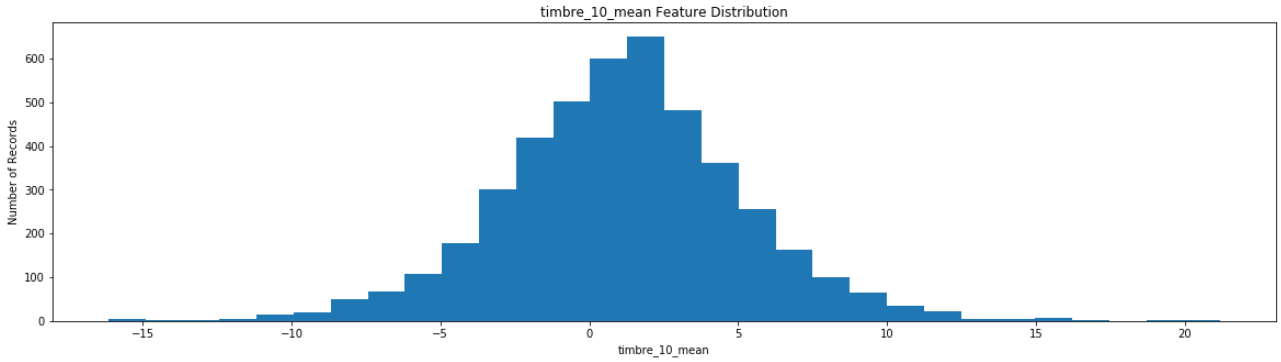

timbre_8_kurtosis Feature Distribution
Number of Records
timbre_8_kurtosis

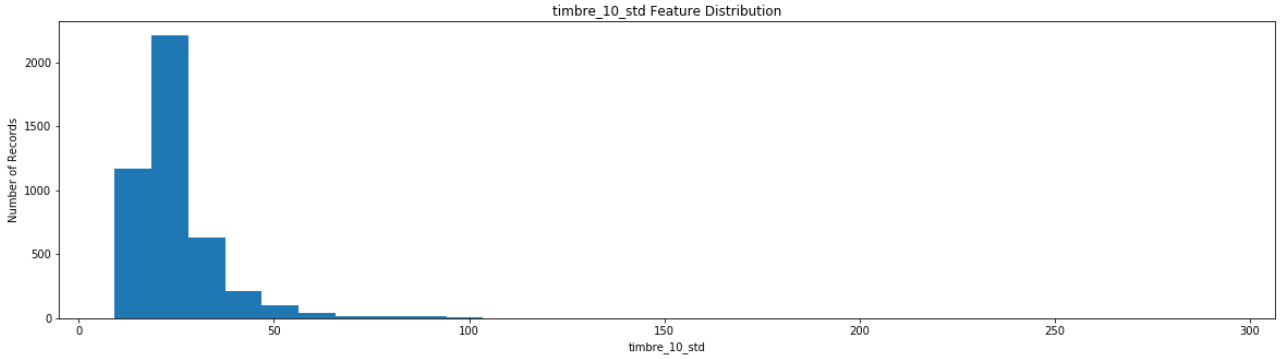

timbre_9_mean Feature Distribution
Number of Records
timbre_9_mean

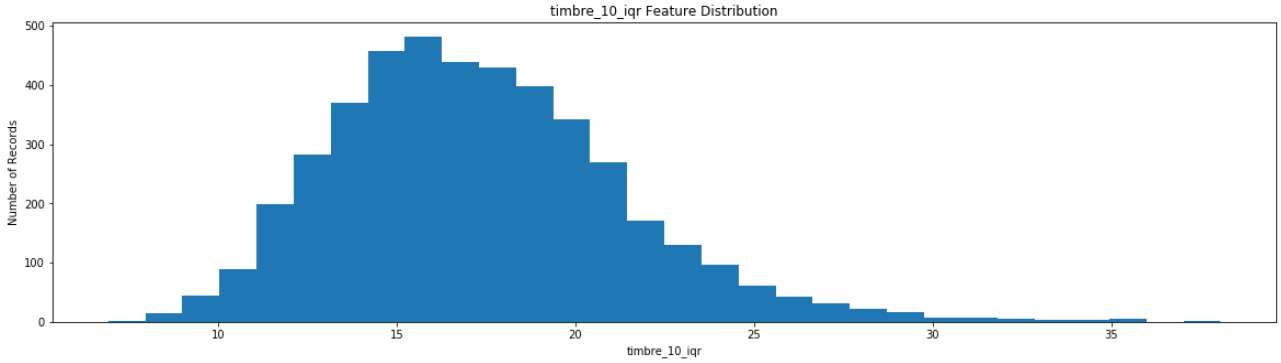

timbre_9_std Feature Distribution
Number of Records
timbre_9_std

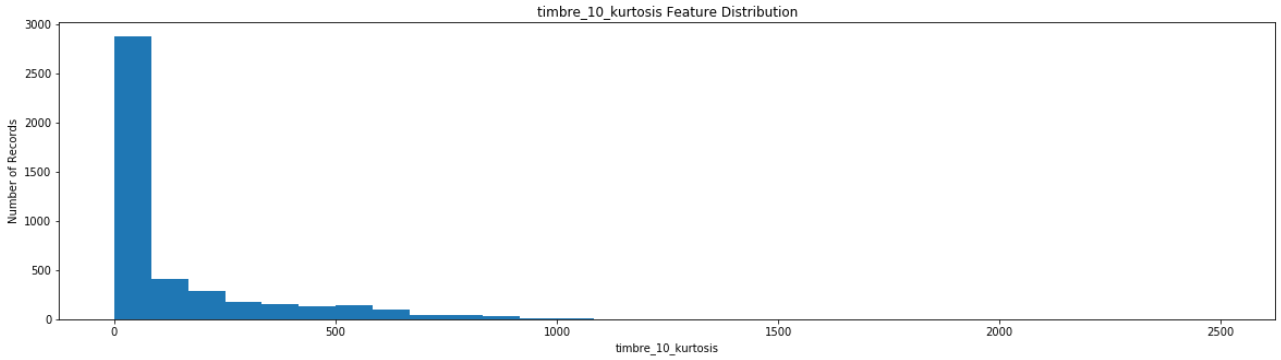

timbre_9_iqr Feature Distribution
Number of Records
timbre_9_iqr

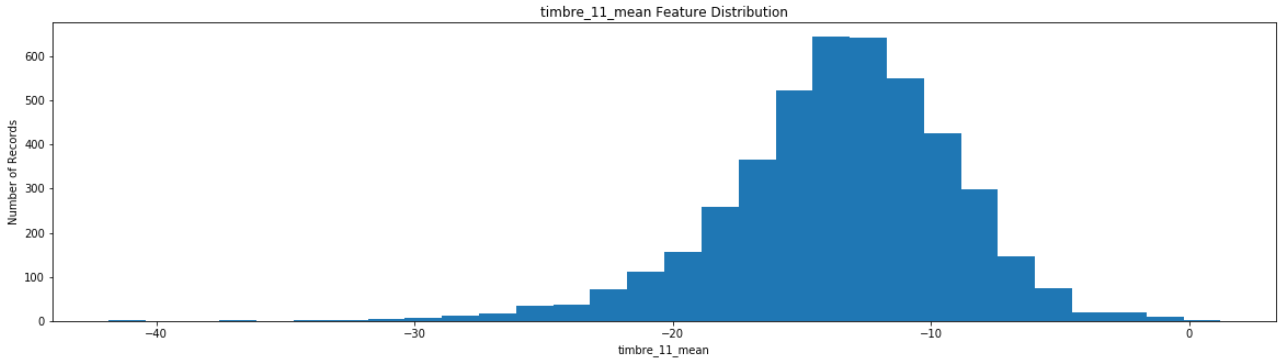

timbre_9_kurtosis Feature Distribution
Number of Records
timbre_9_kurtosis

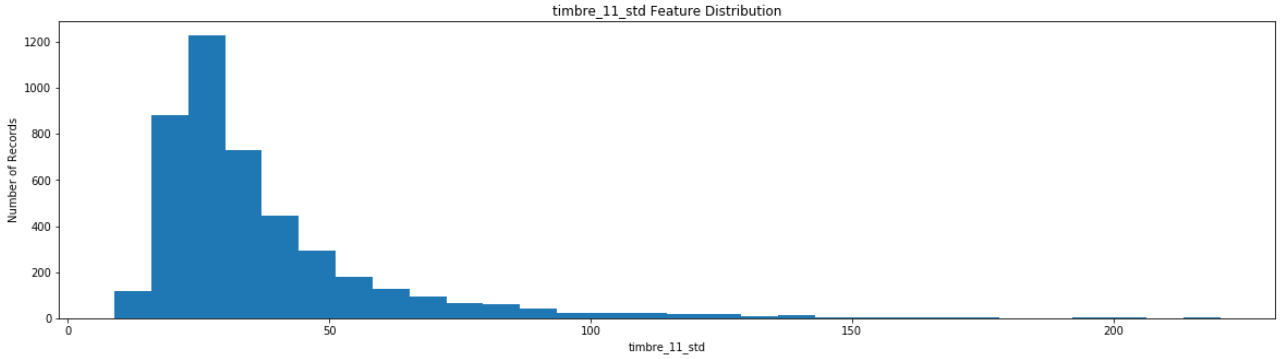

timbre_10_mean Feature Distribution
Number of Records
timbre_10_mean

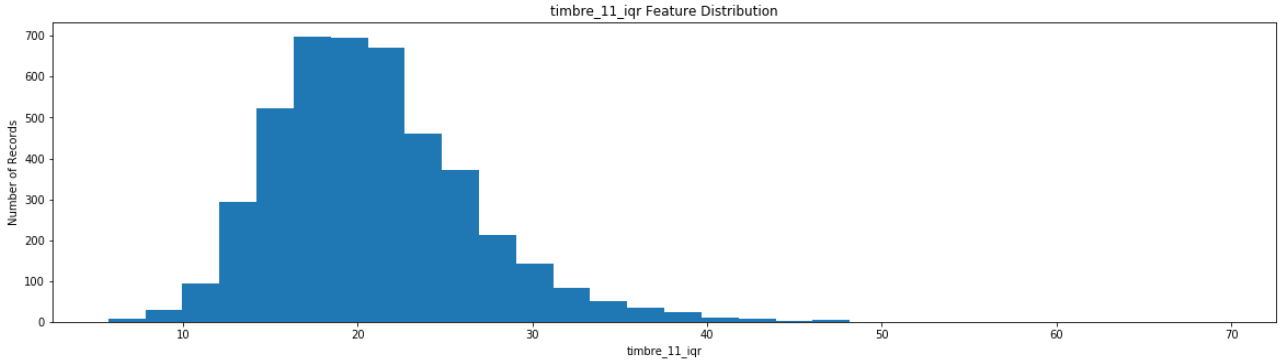

timbre_10_std Feature Distribution
Number of Records
timbre_10_std

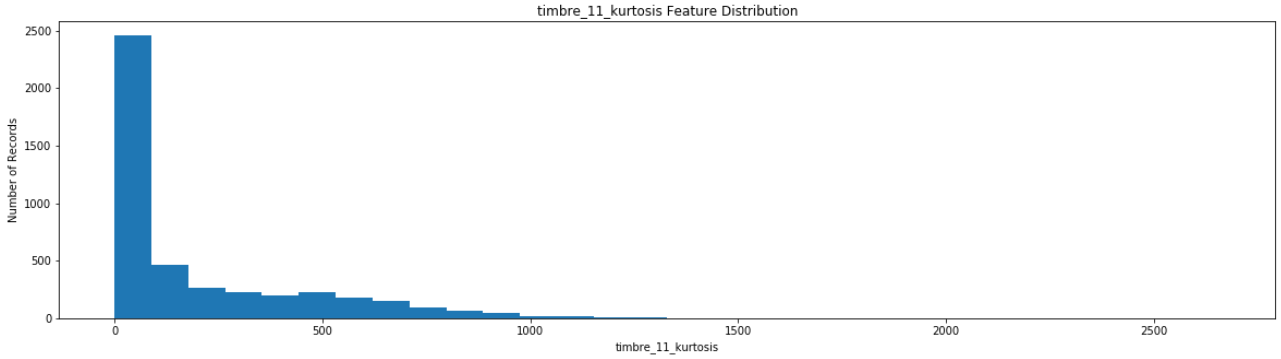

timbre_10_iqr Feature Distribution
Number of Records
timbre_10_iqr

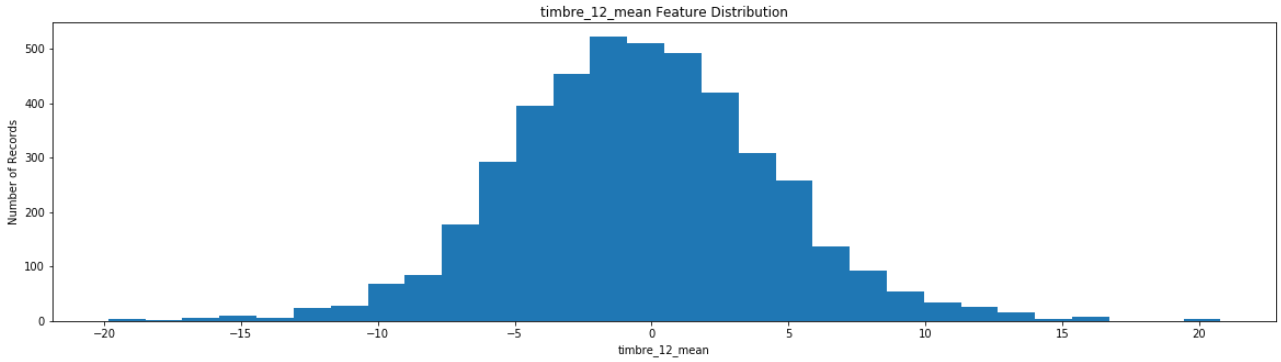

timbre_10_kurtosis Feature Distribution
Number of Records
timbre_10_kurtosis

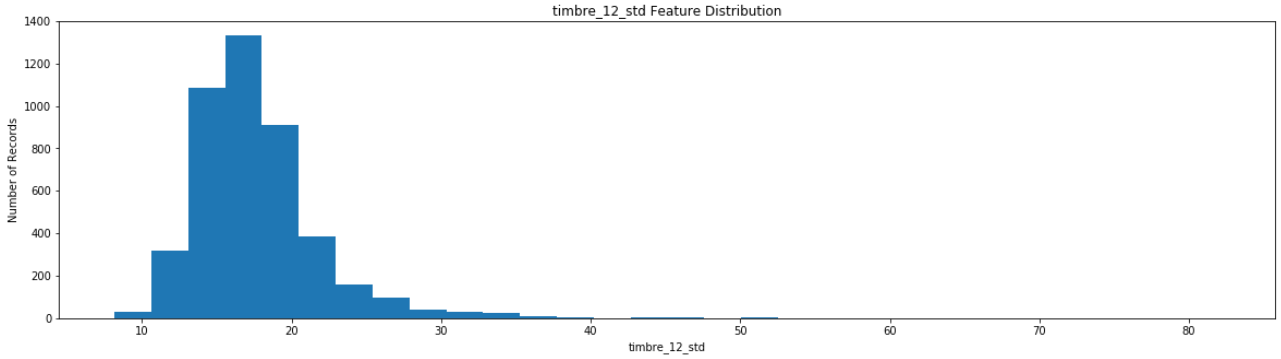

timbre_11_mean Feature Distribution
Number of Records
timbre_11_mean

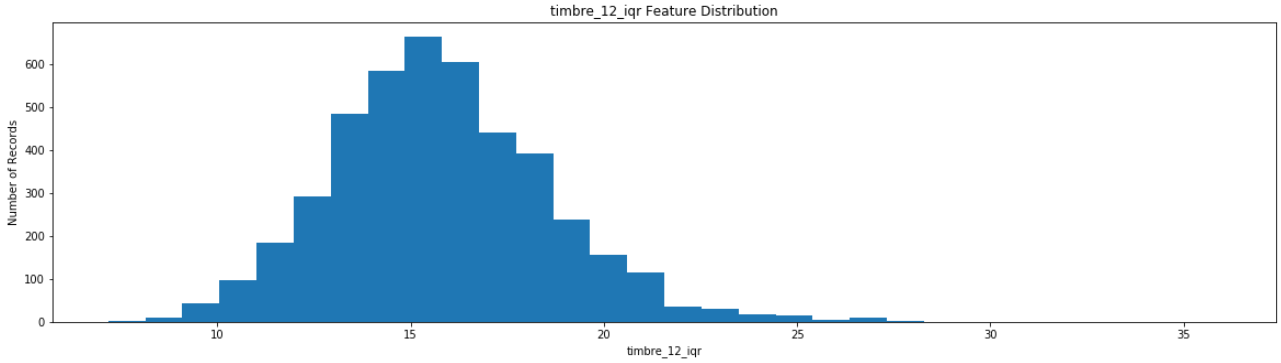

timbre_11_std Feature Distribution
Number of Records
timbre_11_std

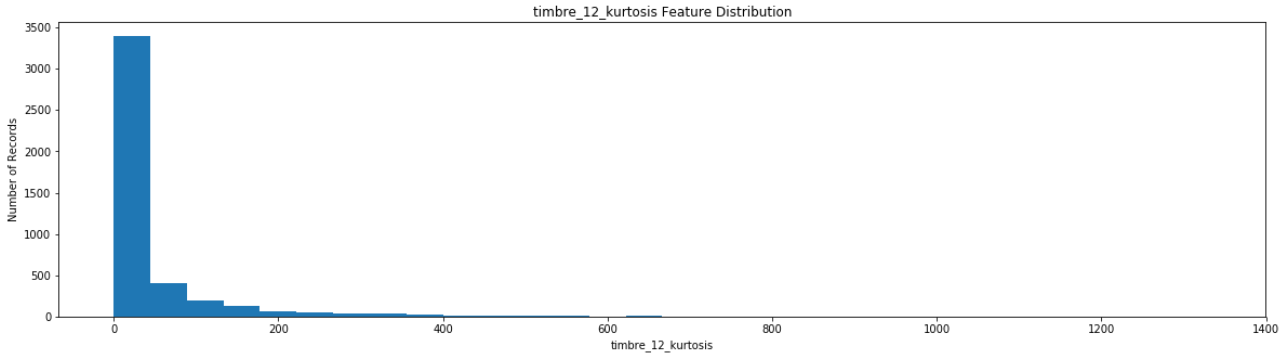

timbre_11_iqr Feature Distribution
Number of Records
timbre_11_iqr


timbre_11_kurtosis Feature Distribution
Number of Records
timbre_11_kurtosis


timbre_12_mean Feature Distribution
Number of Records
timbre_12_mean

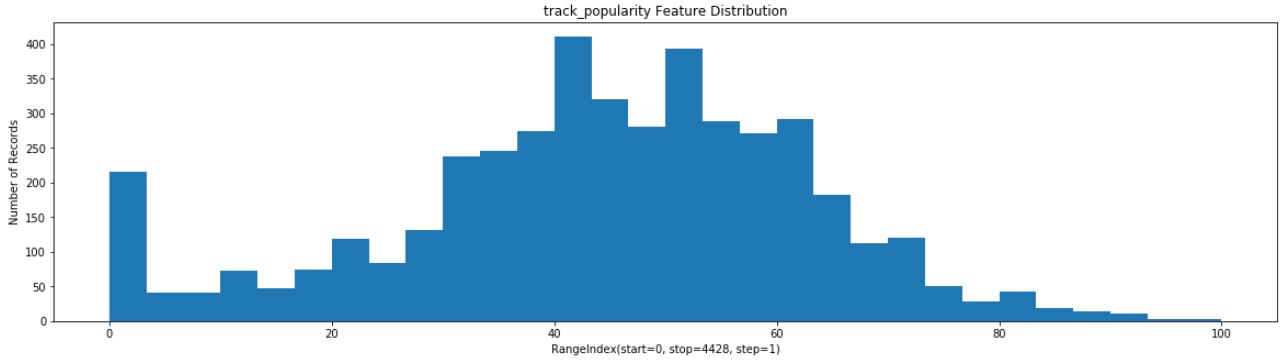

timbre_12_std Feature Distribution
Number of Records
timbre_12_std

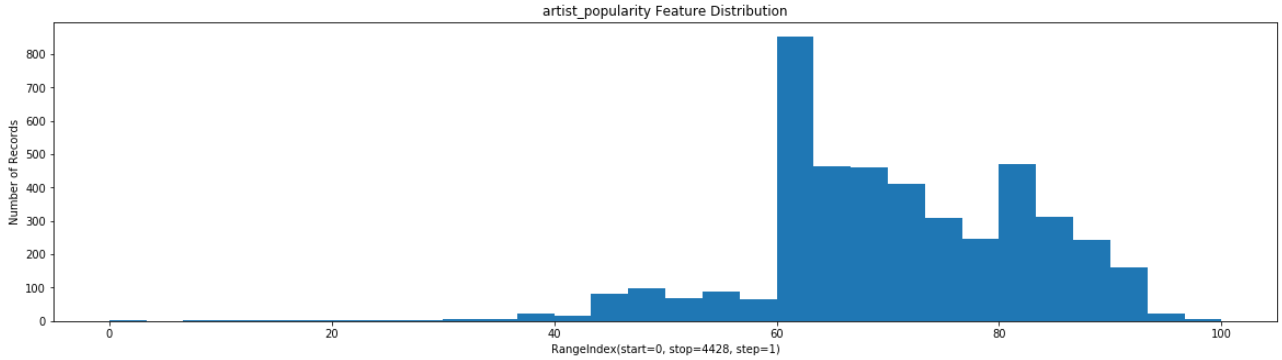

timbre_12_iqr Feature Distribution
Number of Records
timbre_12_iqr

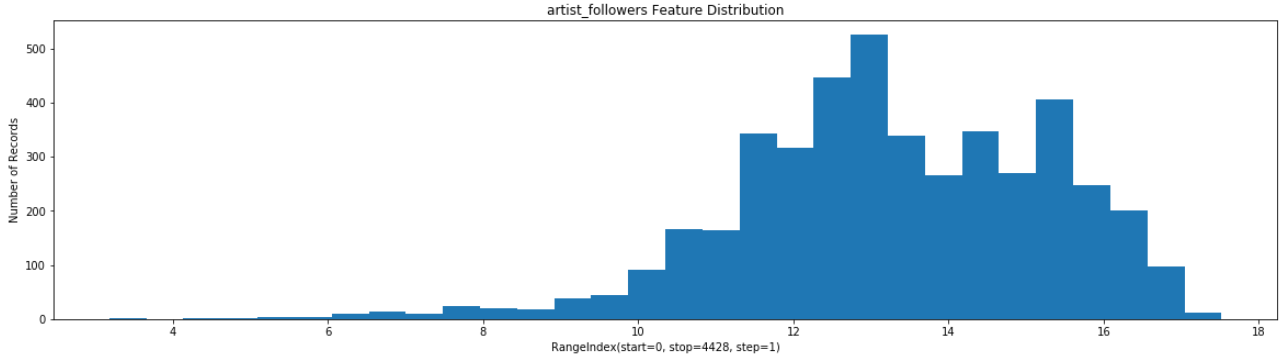

timbre_12_kurtosis Feature Distribution
Number of Records
timbre_12_kurtosis

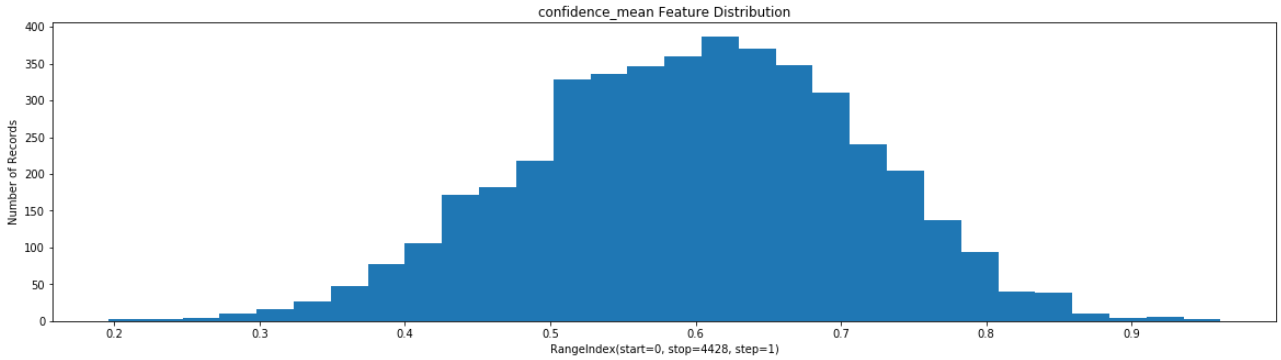
## 2. Feature distribution after log transformation

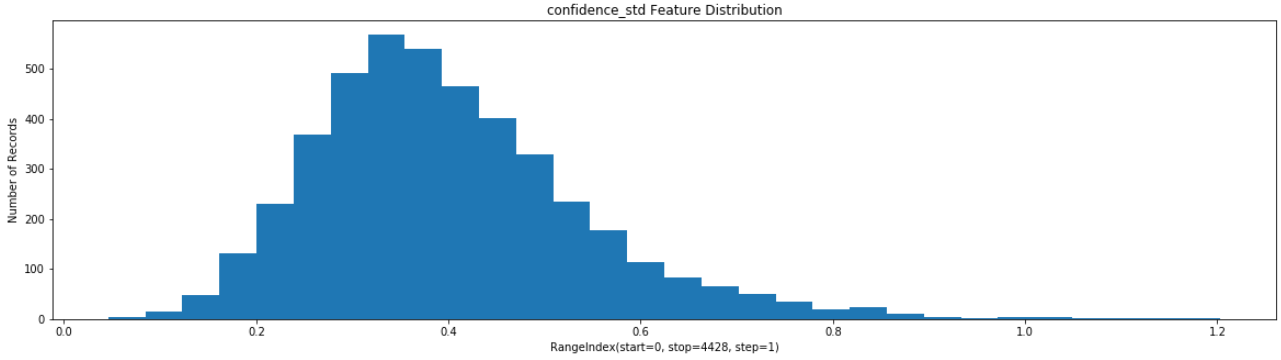
From the histogram shown above, we are able to observe that some of features are skewed. These features are log transformed, and distribution is shown as below.

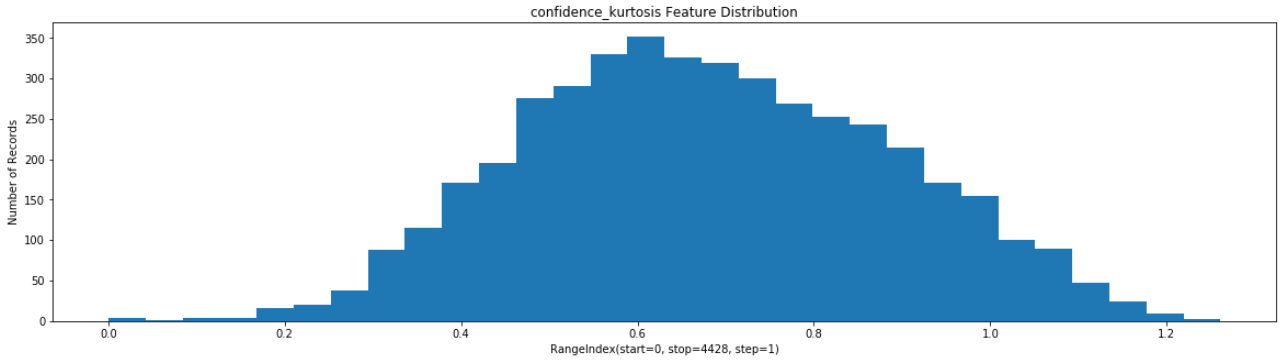


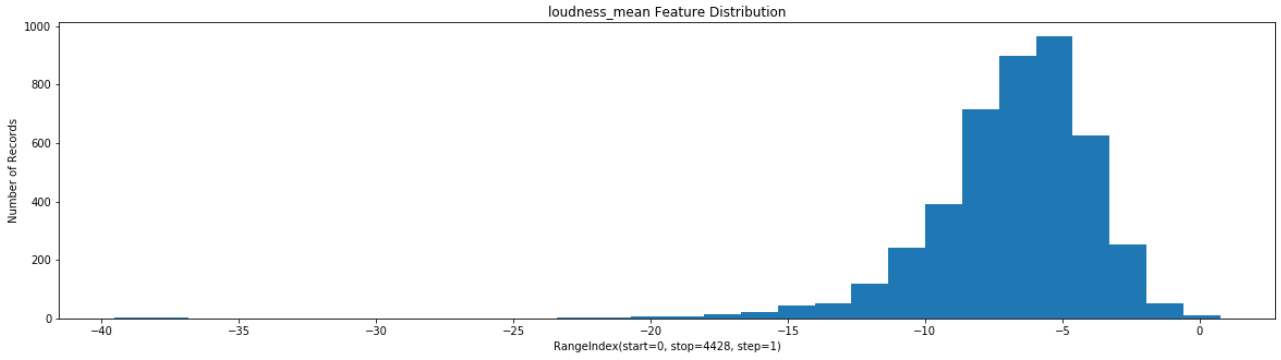



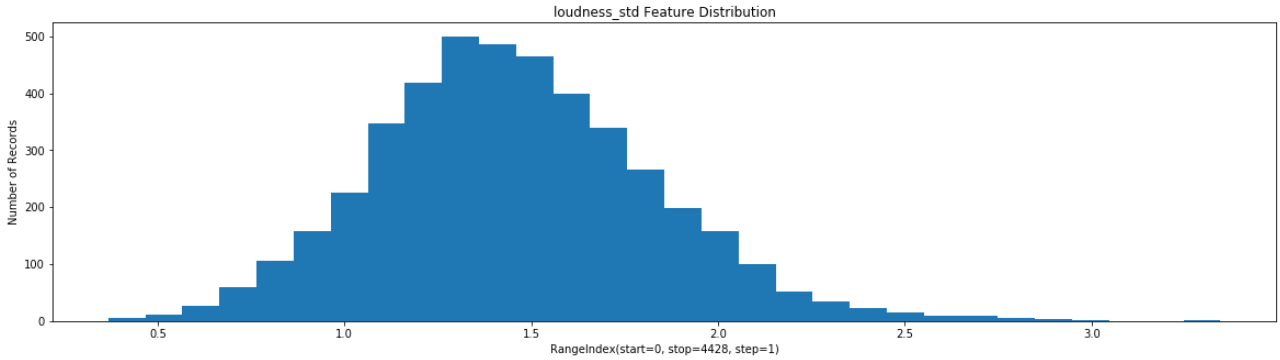



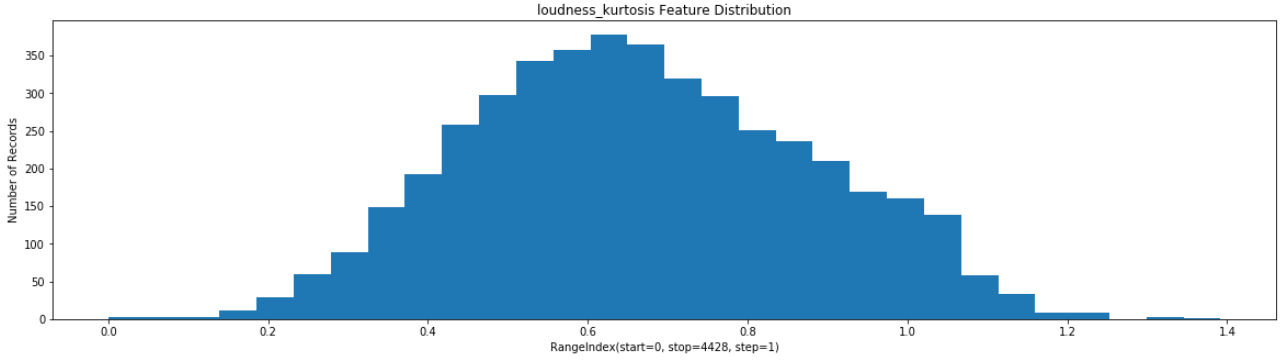

confidence_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

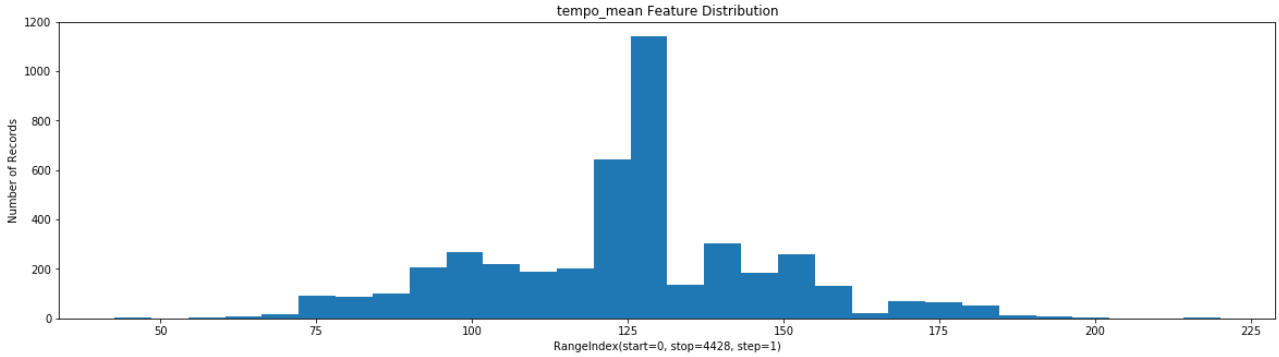

confidence_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

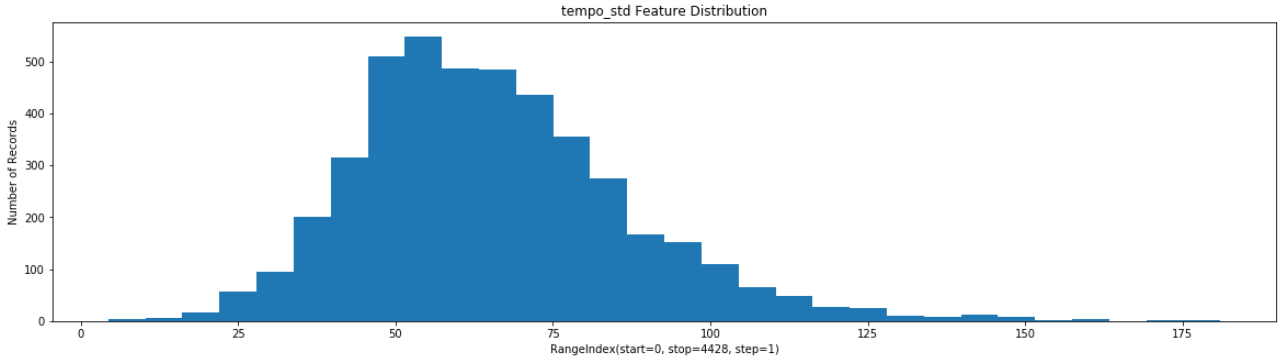

confidence_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

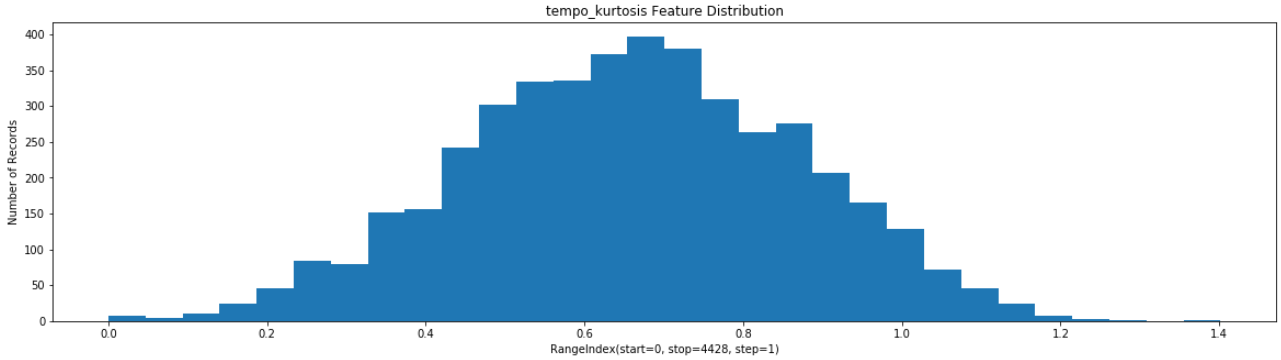

loudness_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

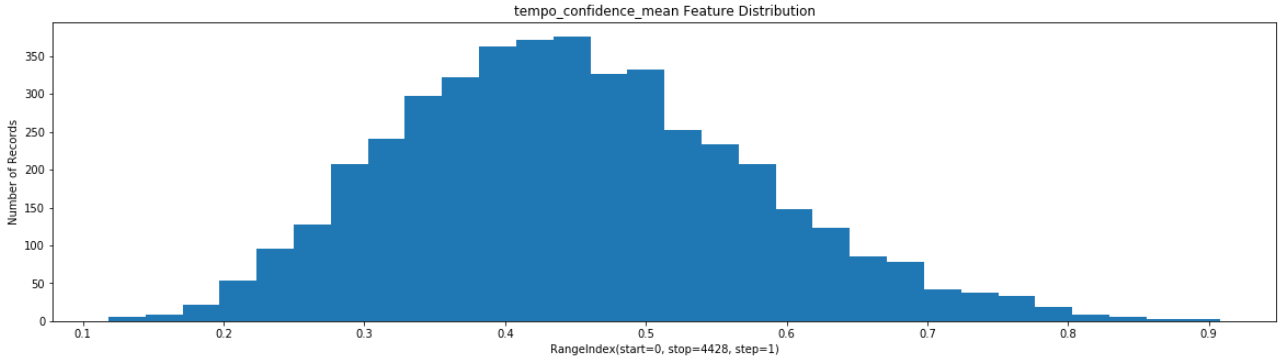

loudness_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

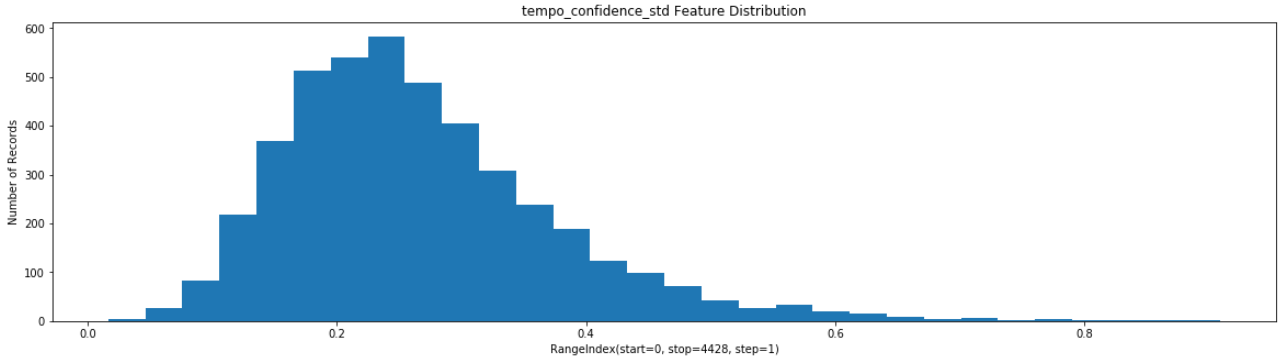

loudness_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

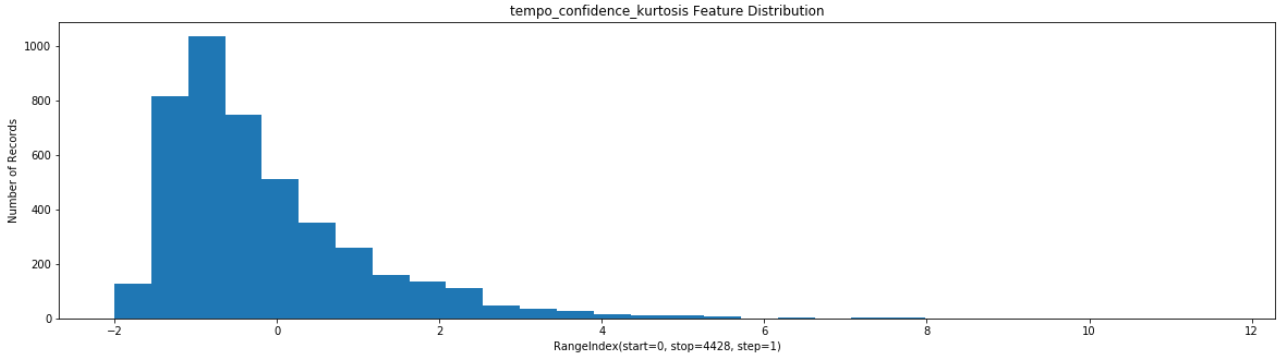

tempo_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

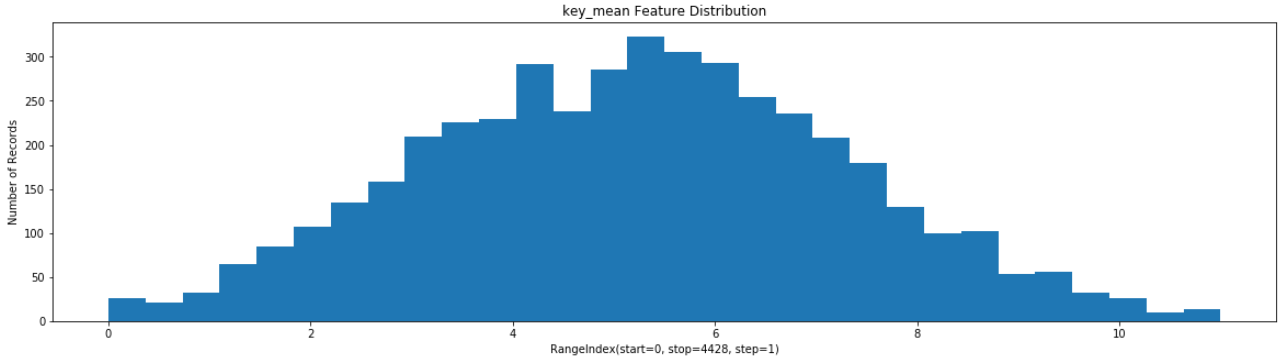

tempo_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

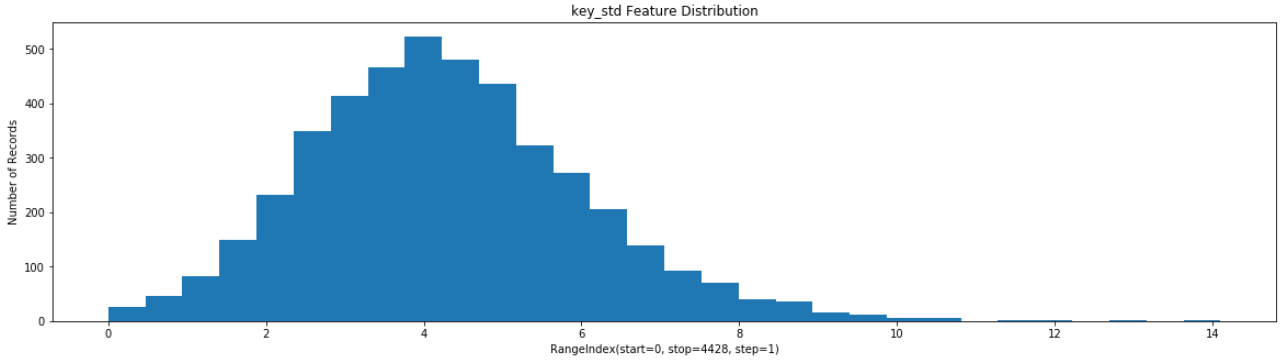

tempo_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

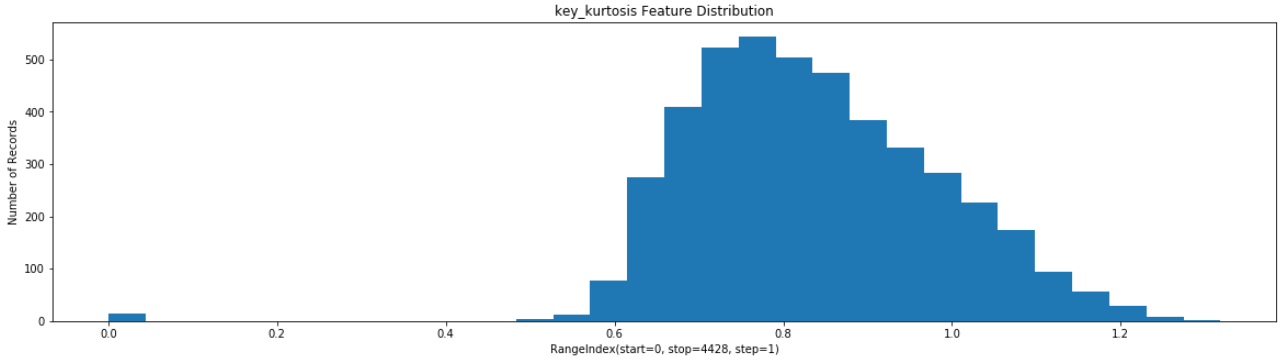

tempo_confidence_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

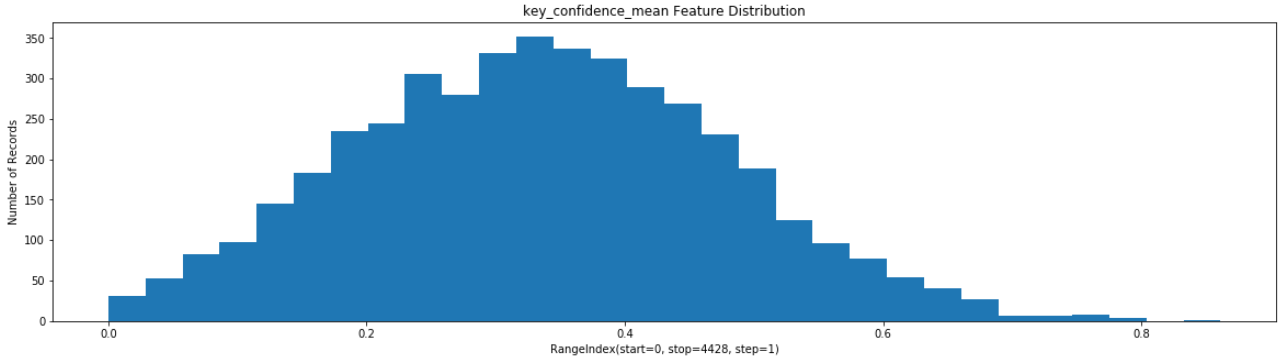

tempo_confidence_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

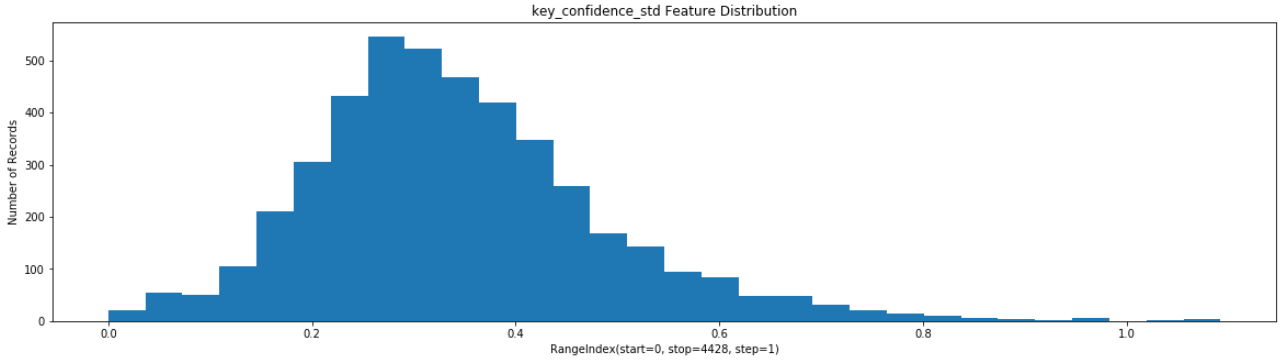

tempo_confidence_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

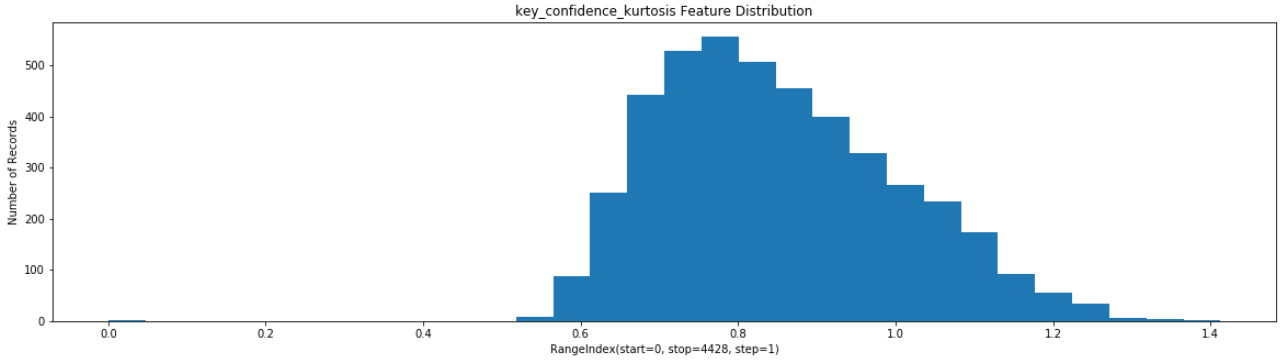

key_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

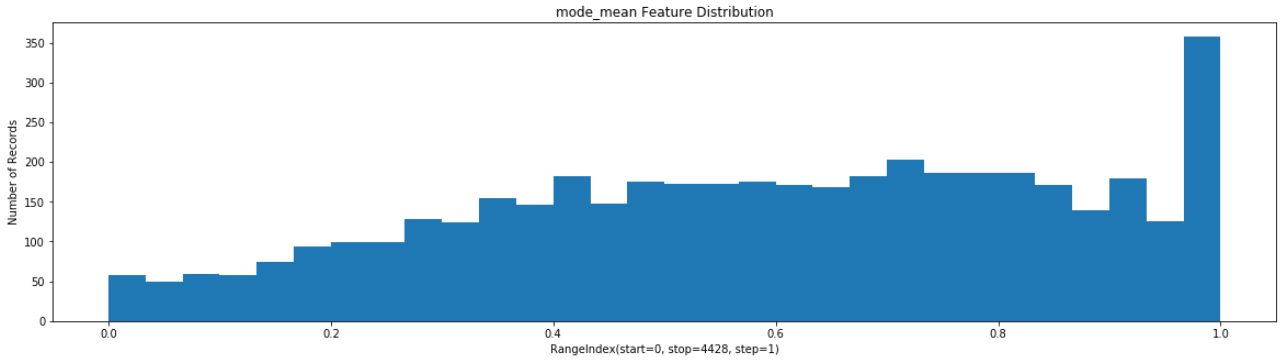

key_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

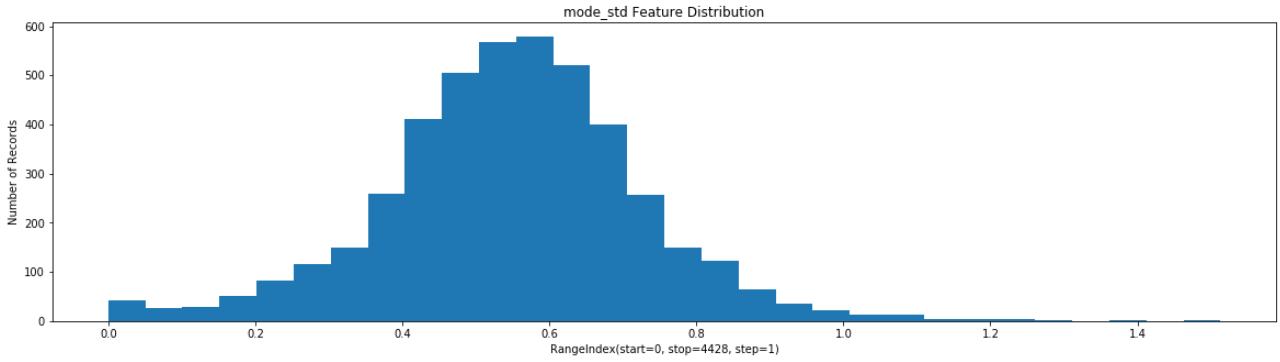

key_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

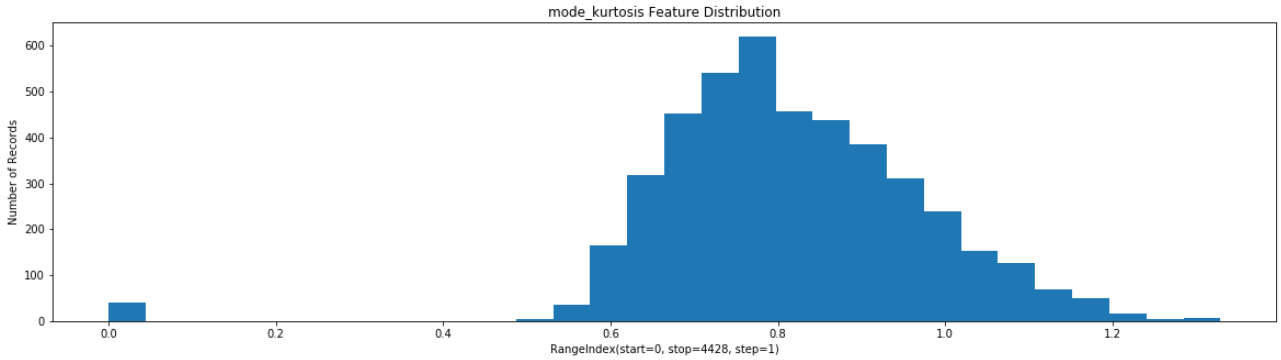

key_confidence_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

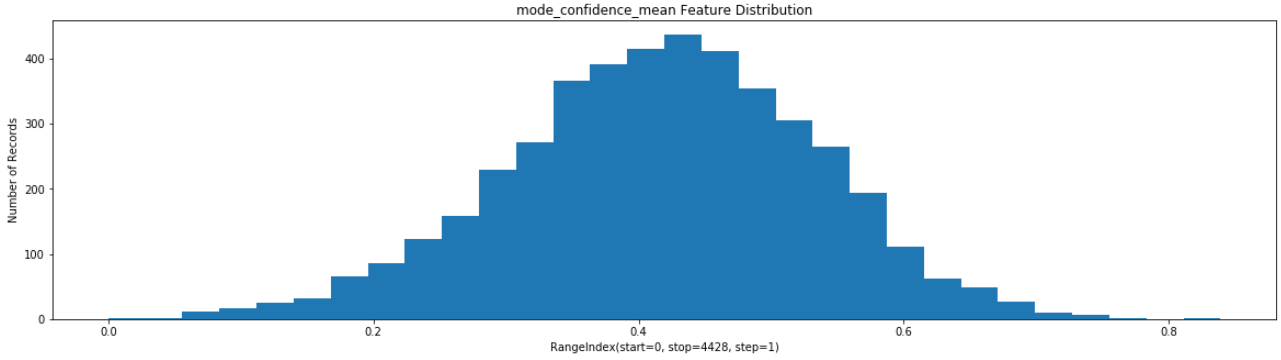

key_confidence_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

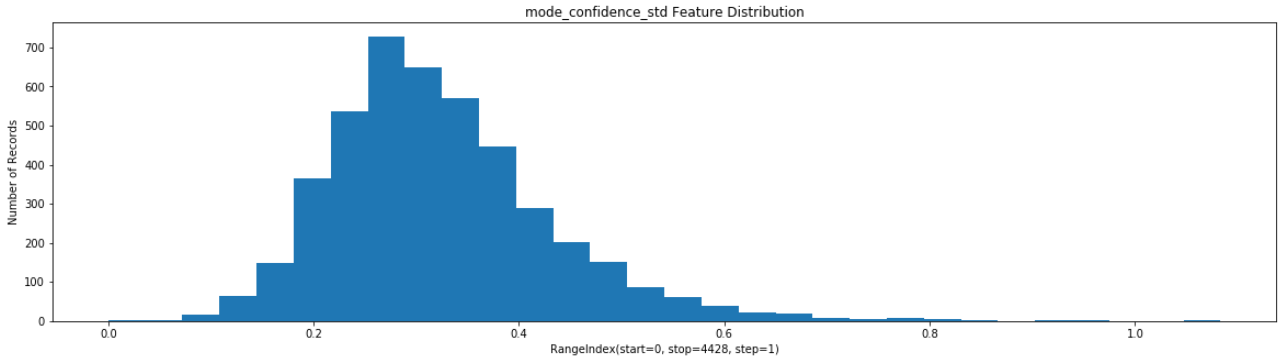

key_confidence_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

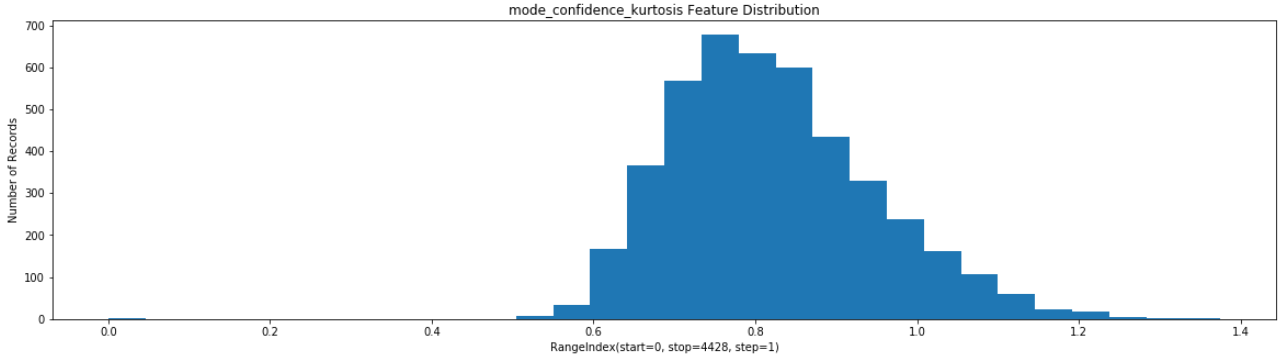

mode_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

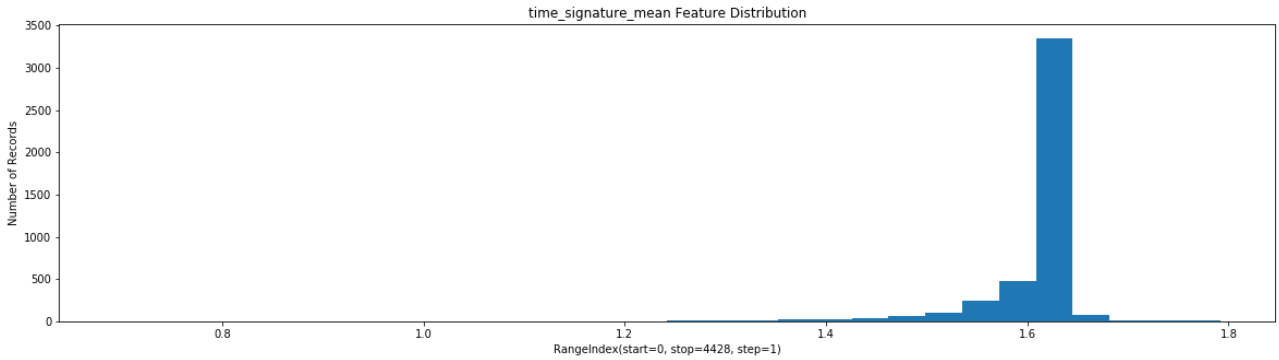

mode_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

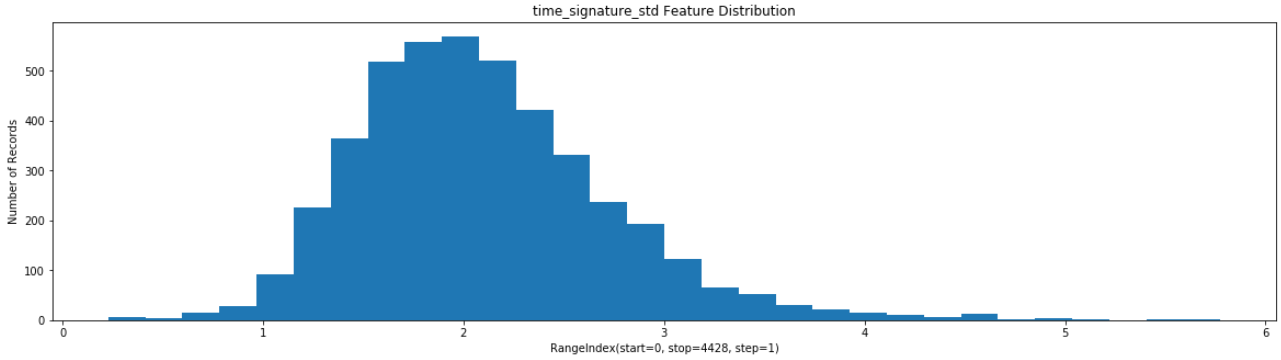

mode_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

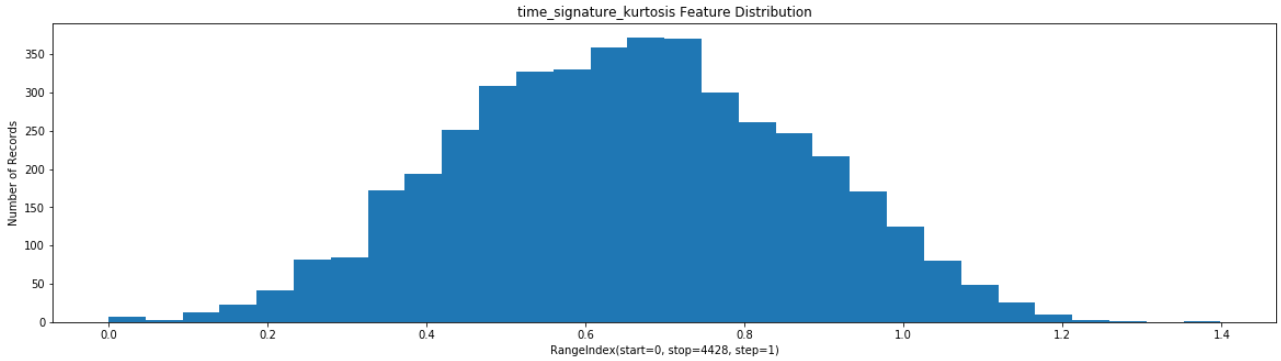

mode_confidence_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

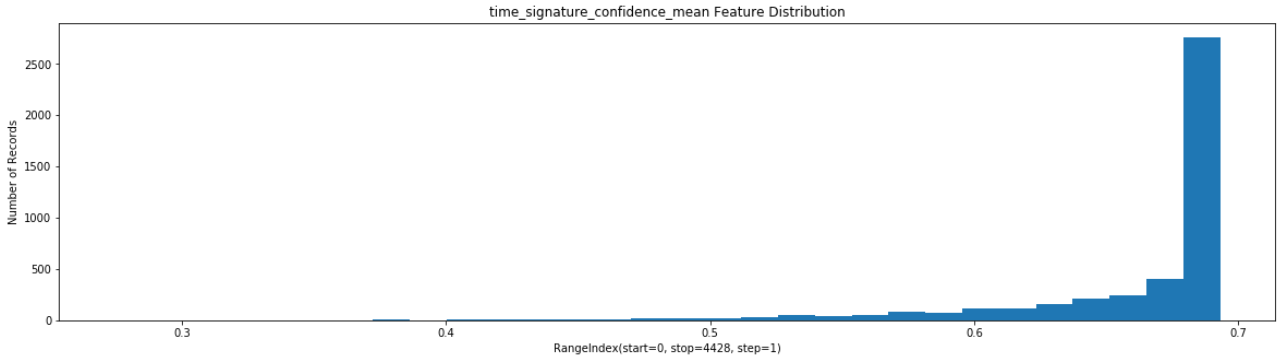

mode_confidence_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

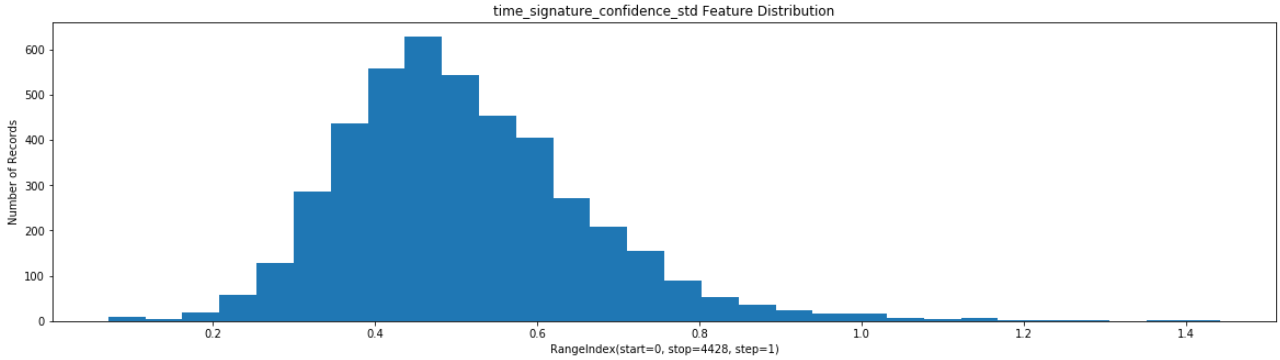

mode_confidence_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

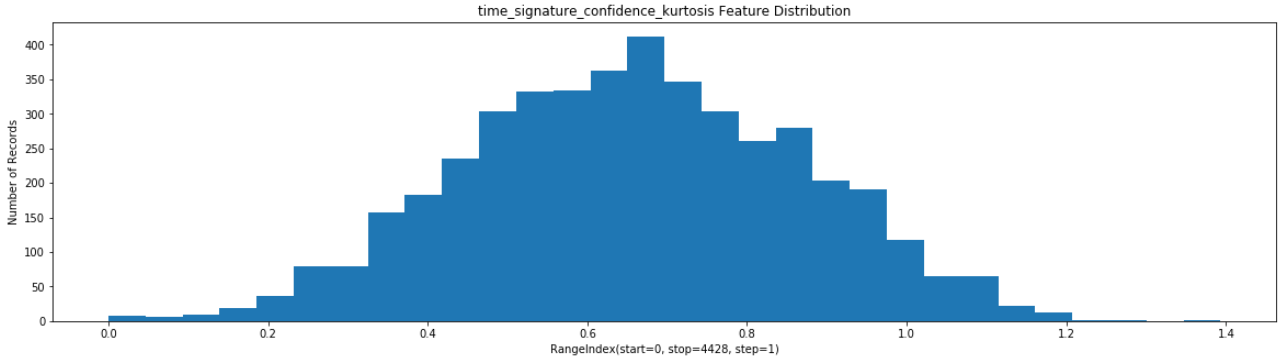

time_signature_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

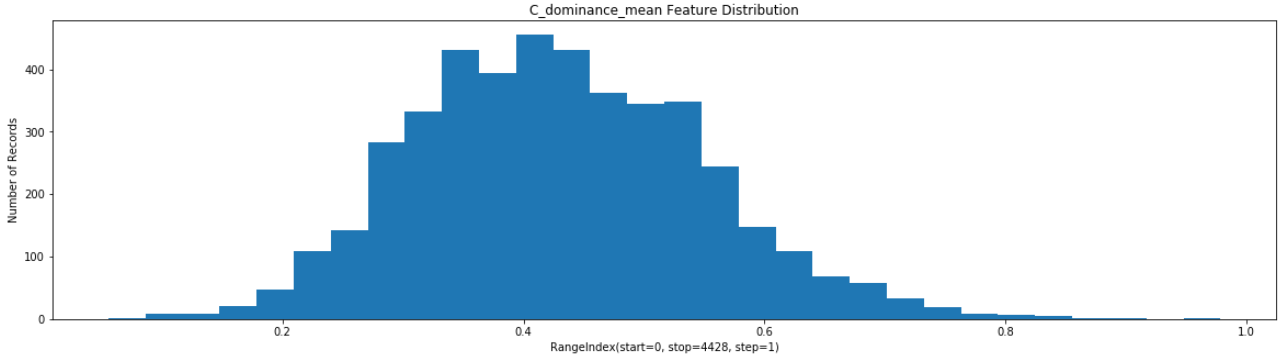

time_signature_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

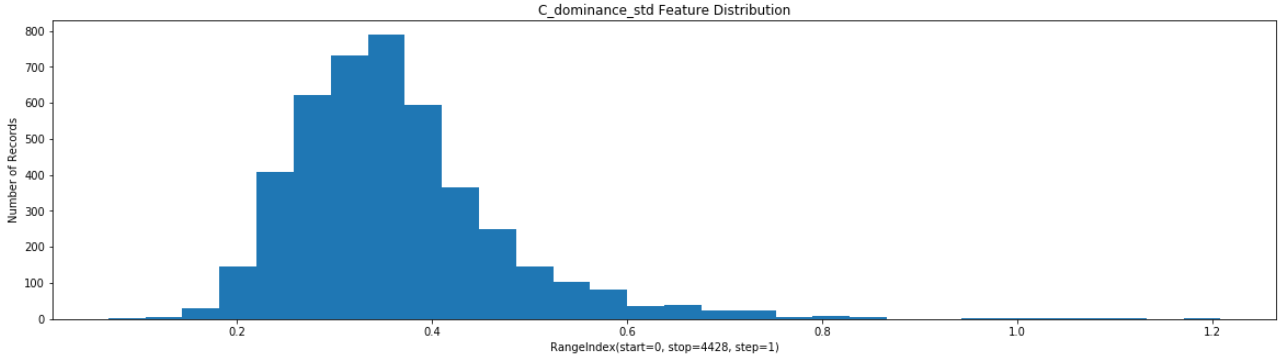

time_signature_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

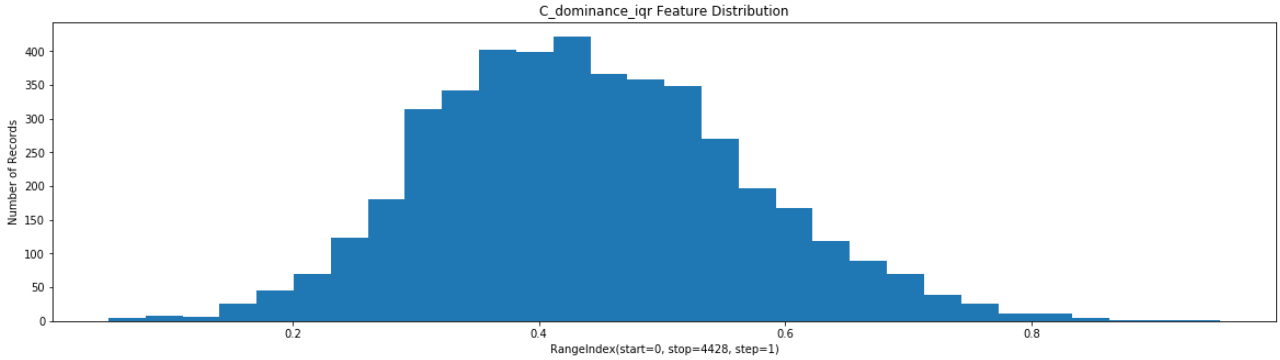

time_signature_confidence_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

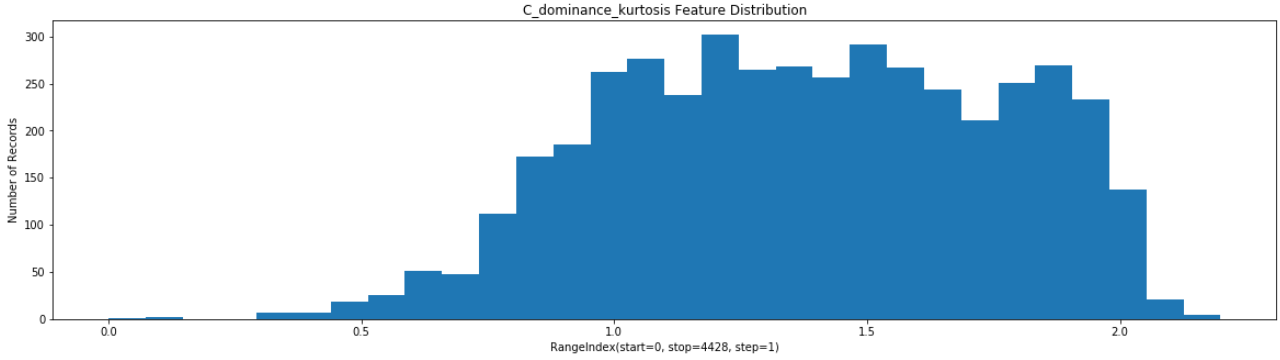

time_signature_confidence_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

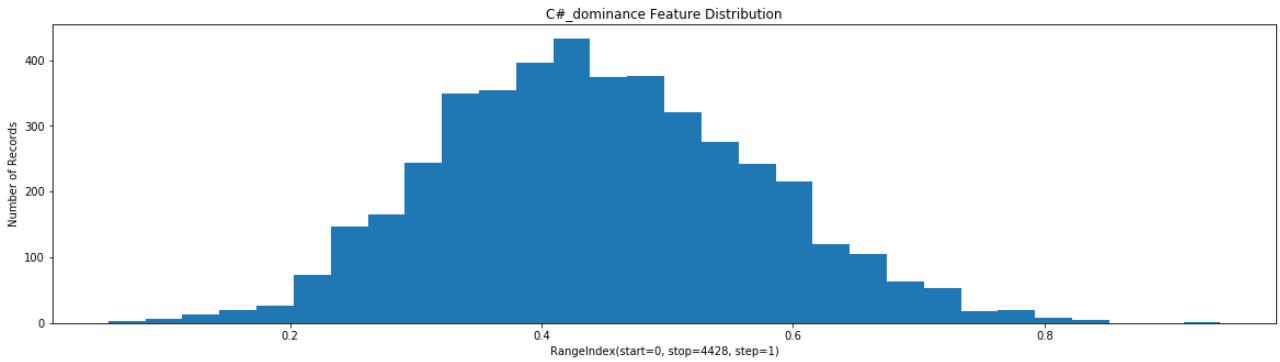

time_signature_confidence_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

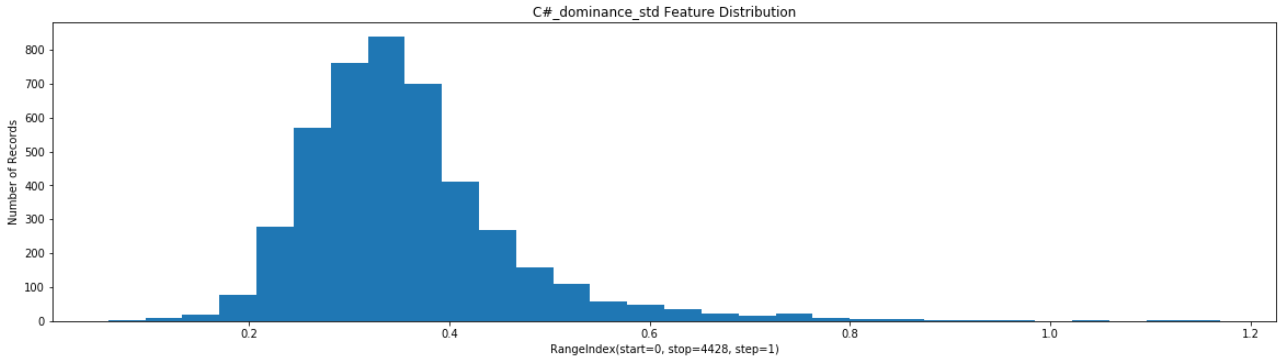

C_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

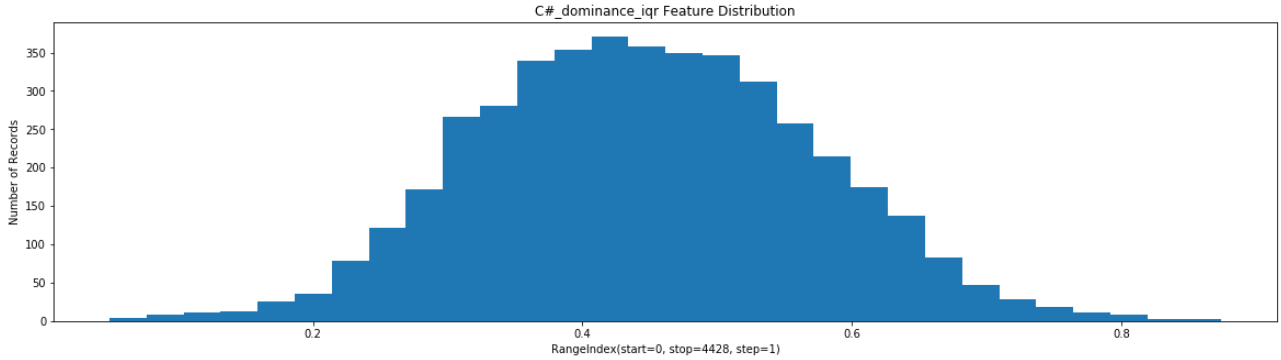

C_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

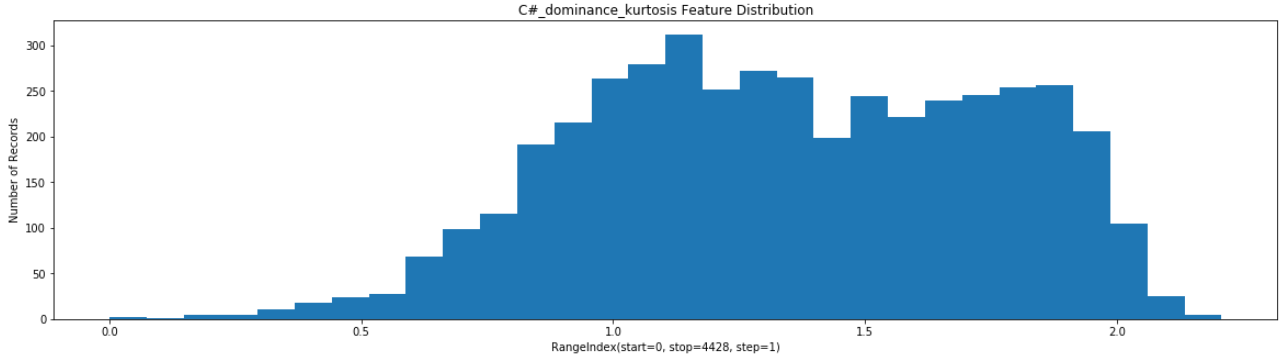

C_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

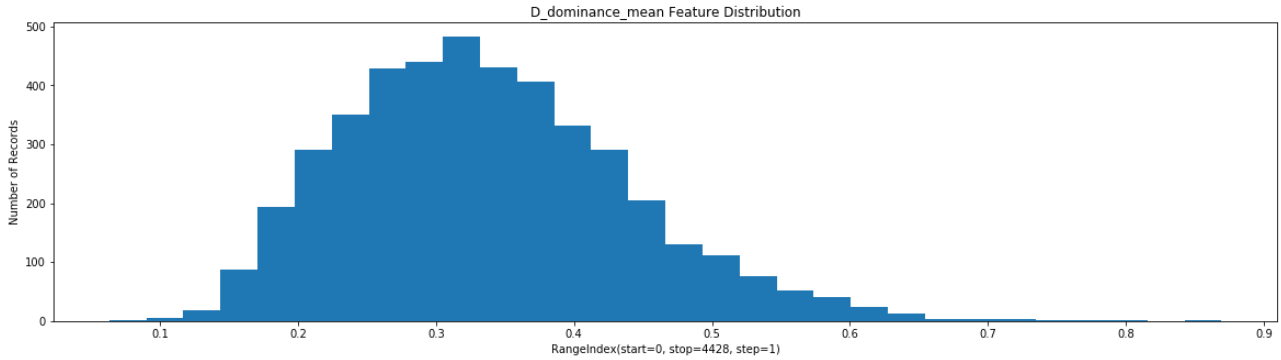

C_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

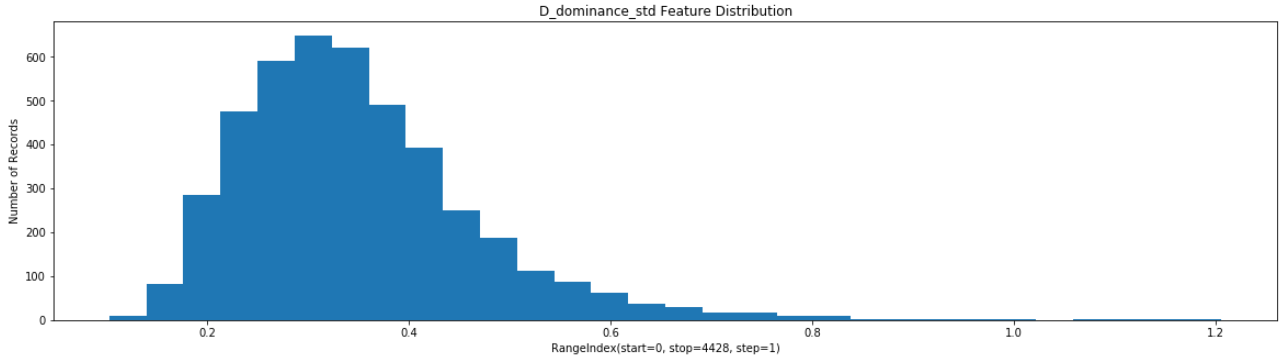

C#_dominance Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

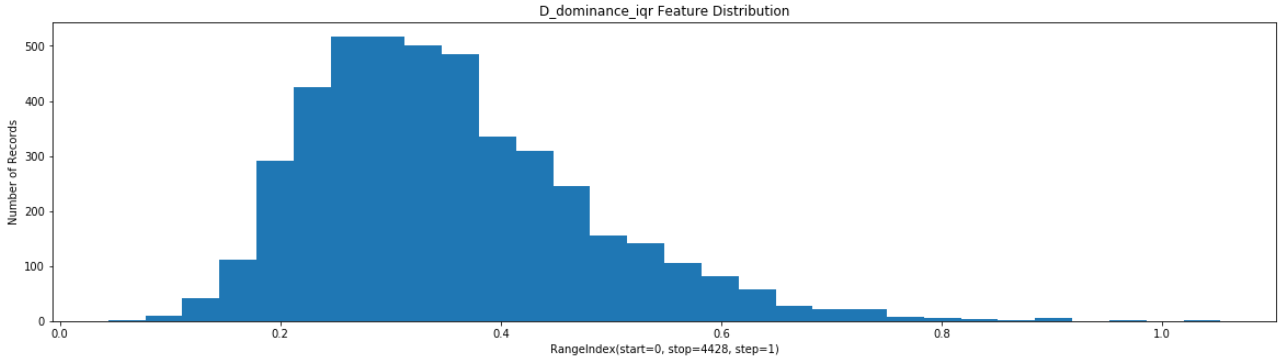

C#_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

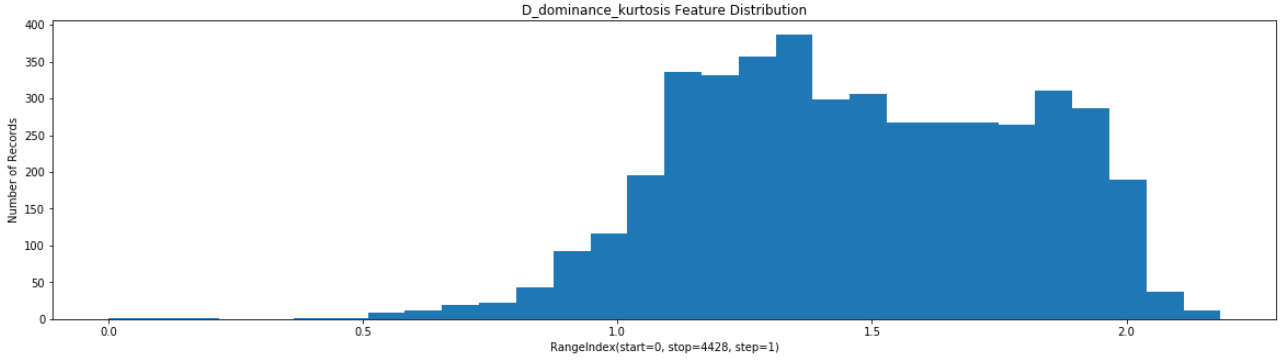

C#_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

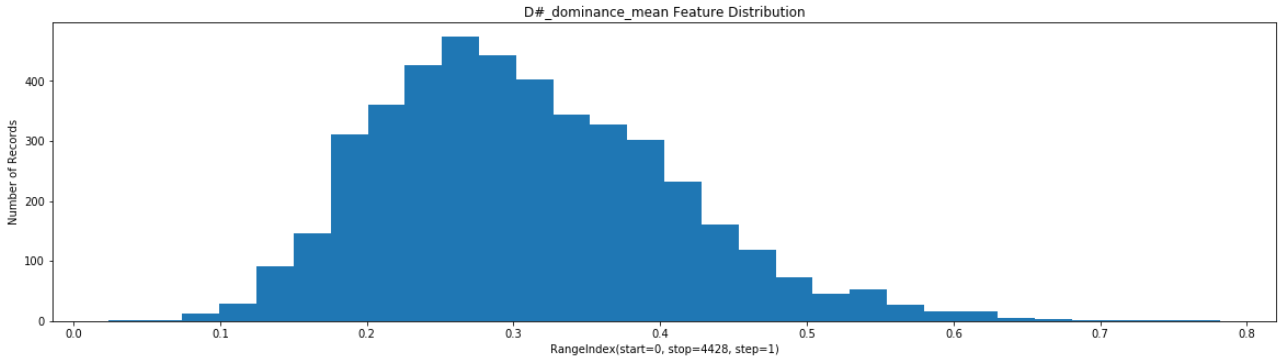

C#_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

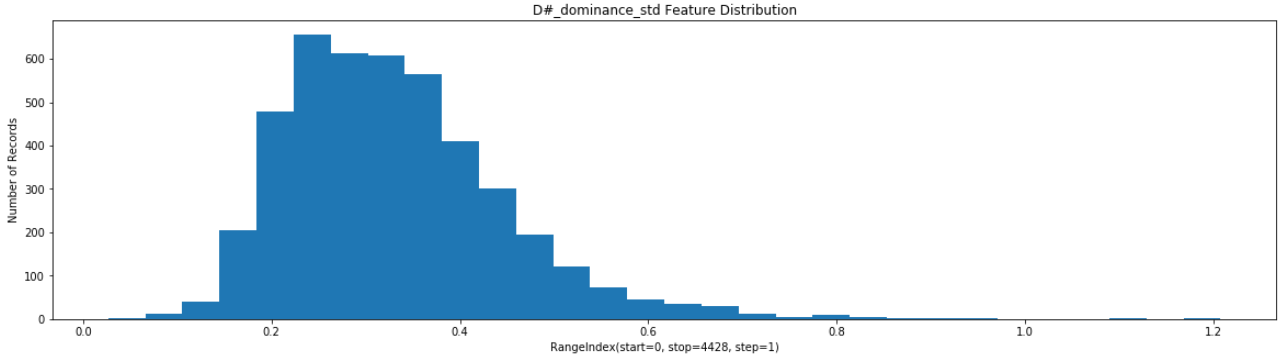

D_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

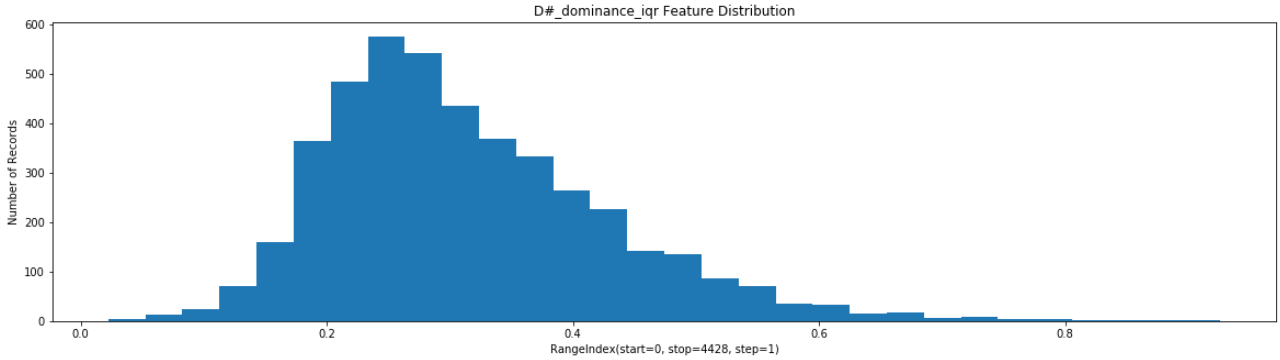

D_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

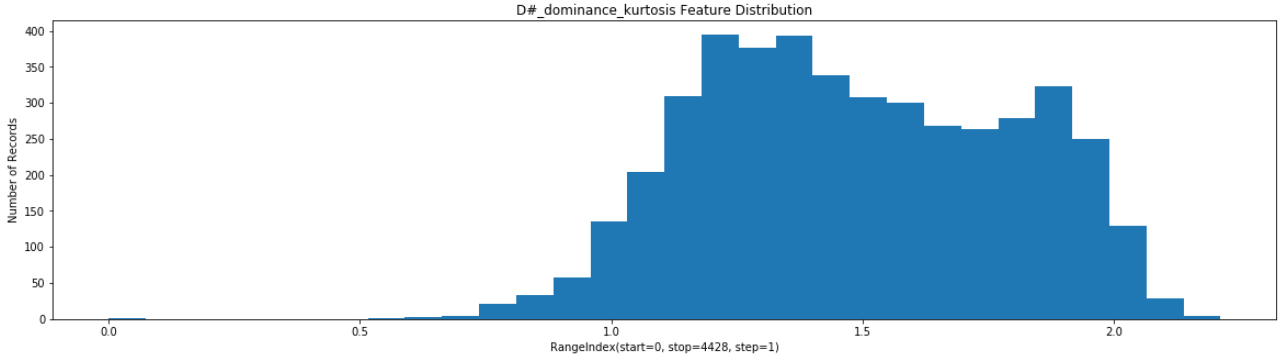

D_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

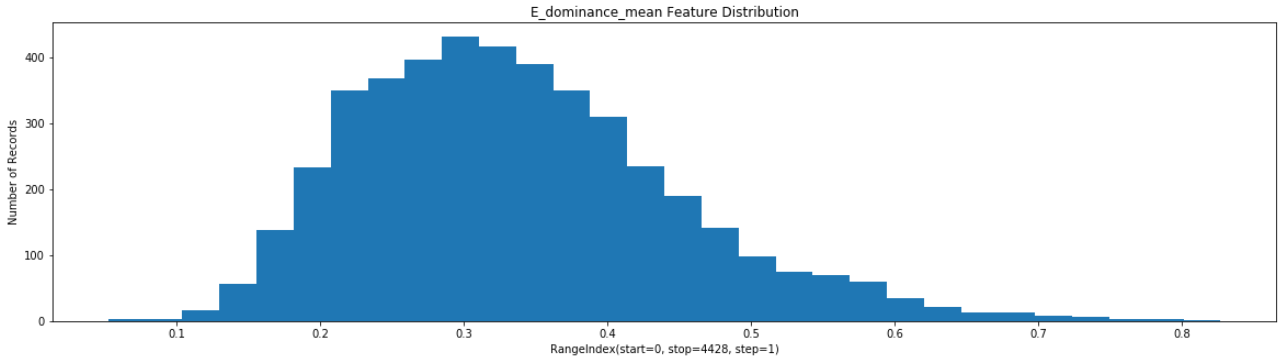

D_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

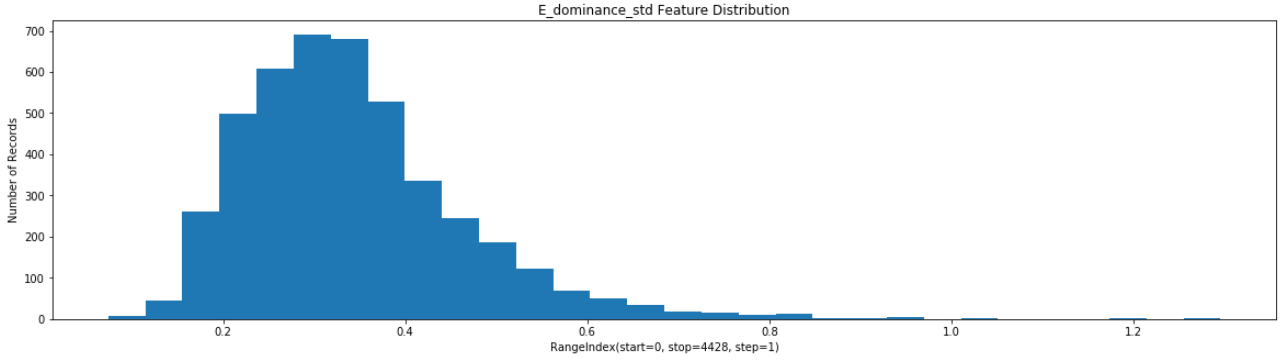

D#_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

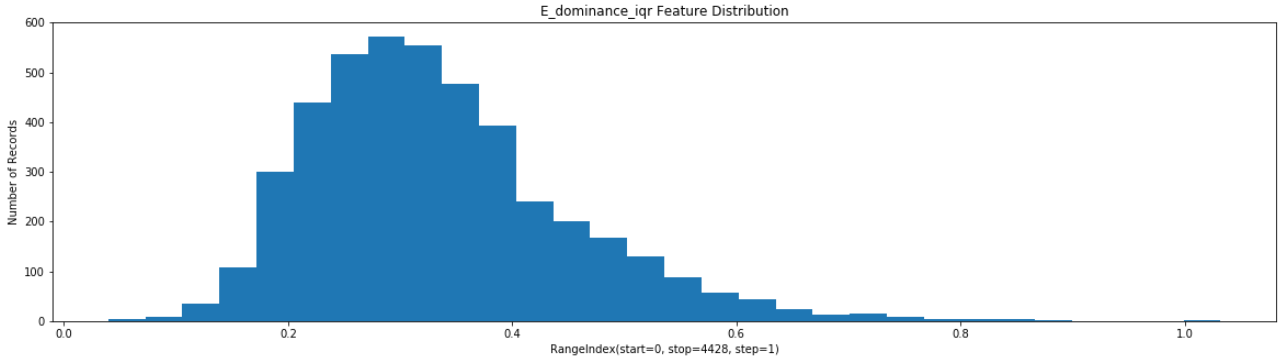

D#_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

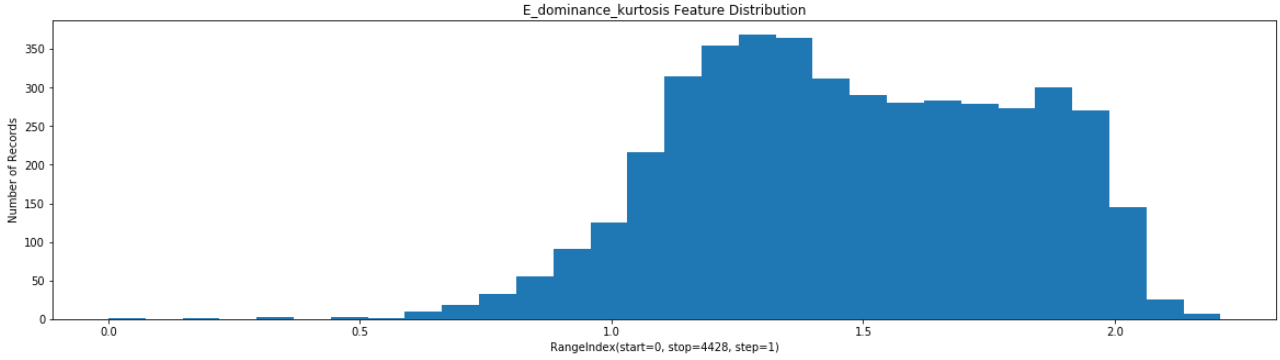

D#_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

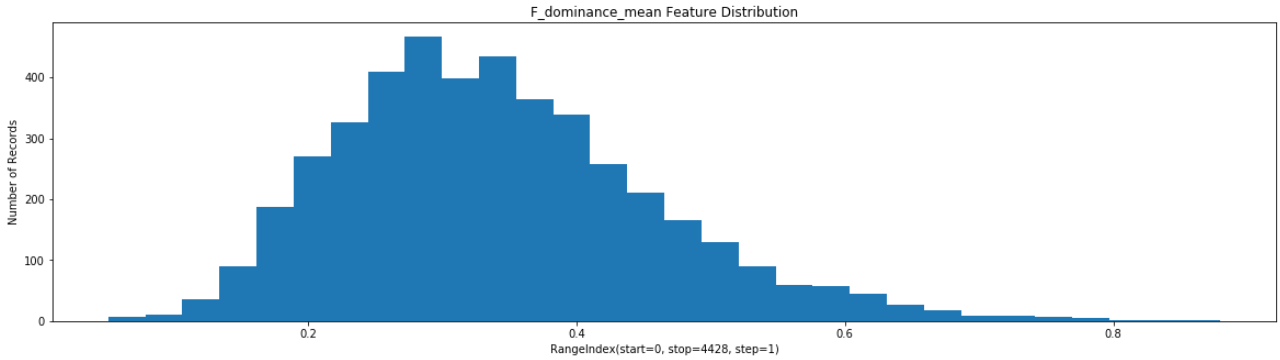

D#_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

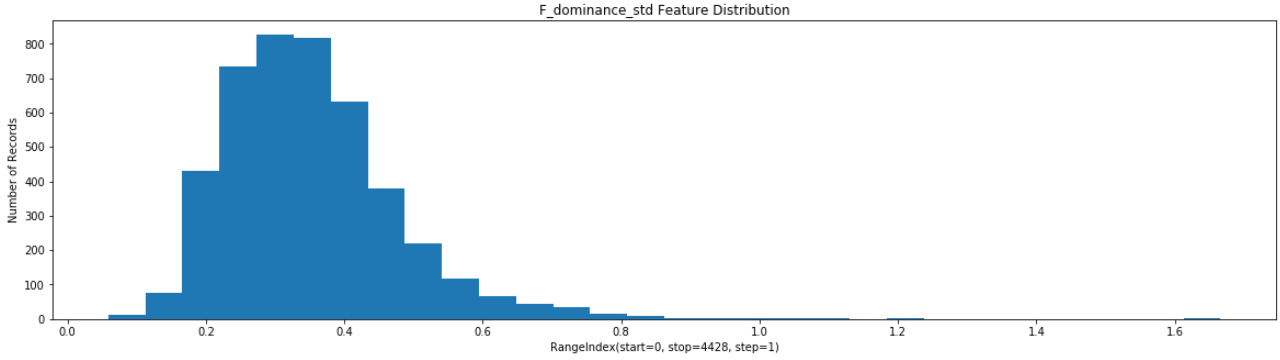

E_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

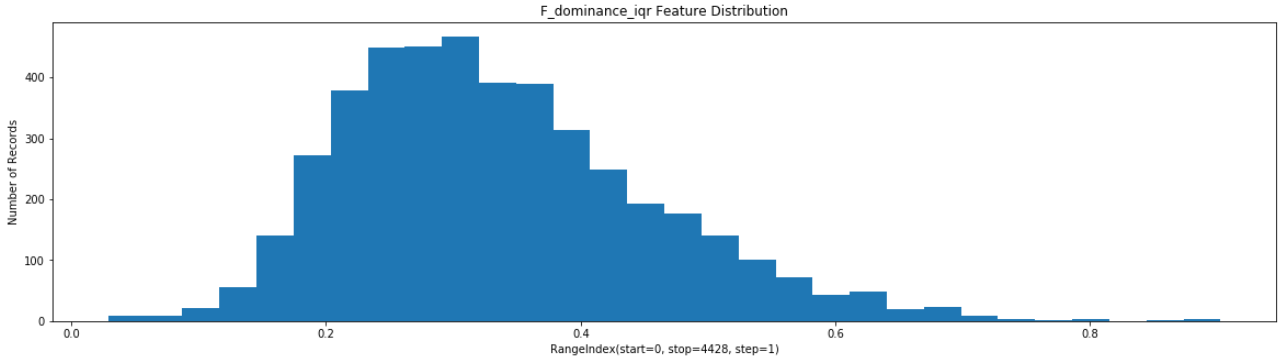

E_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

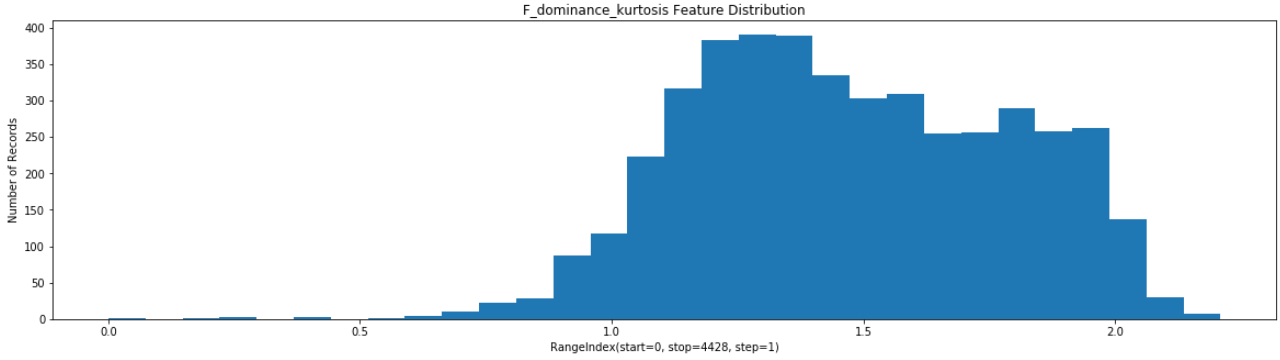

E_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

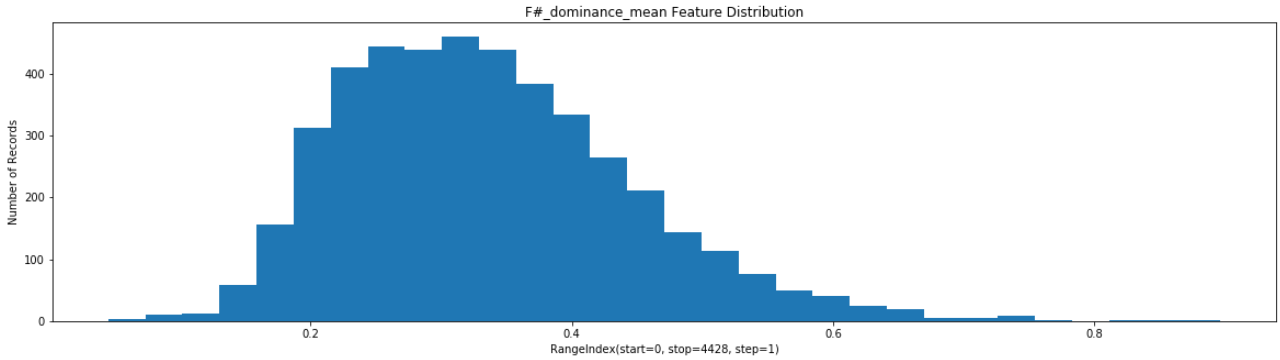

E_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

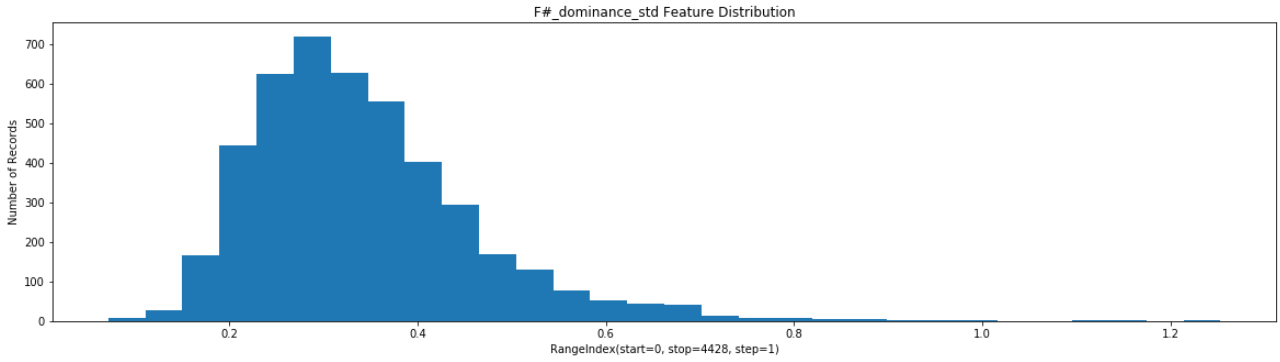

F_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

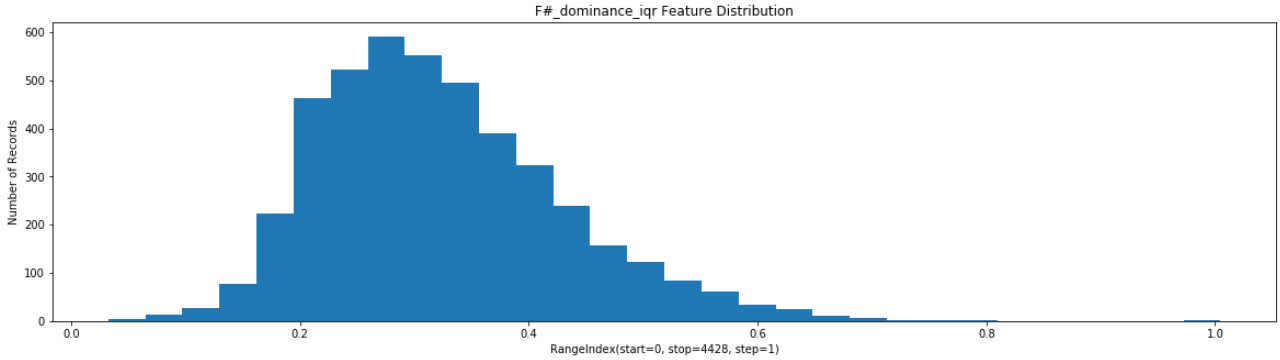

F_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

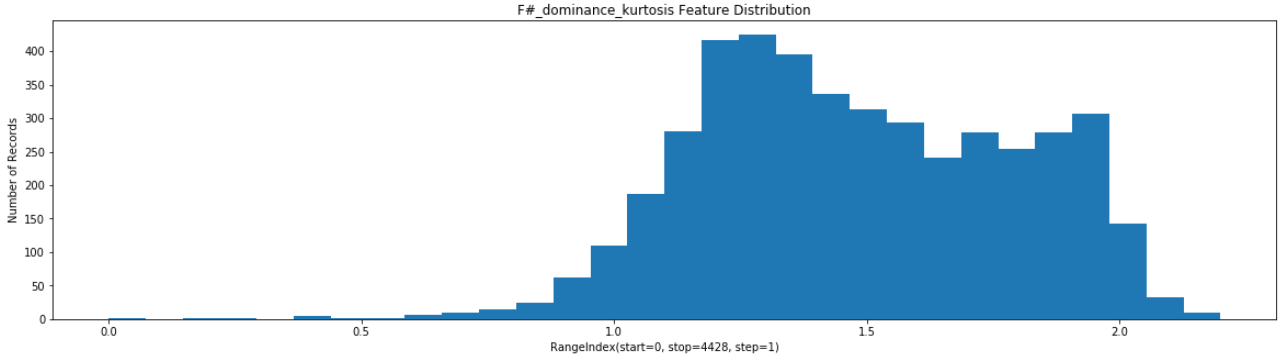

F_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

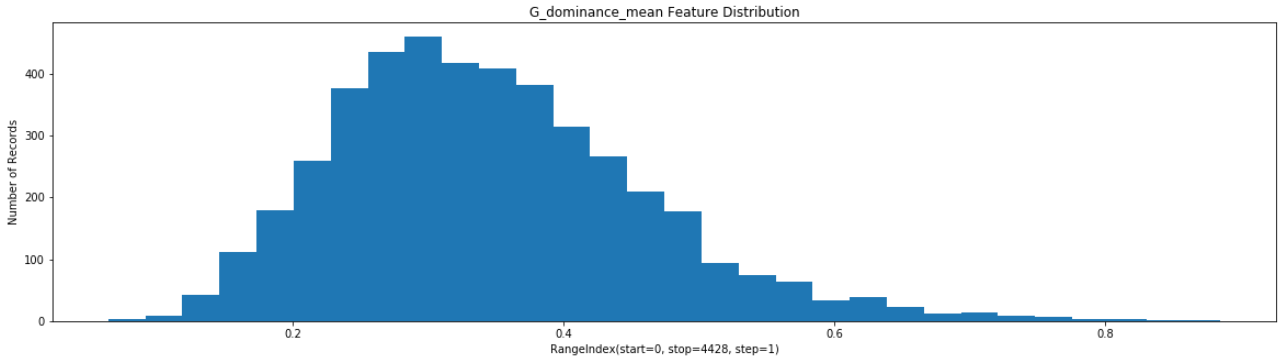

F_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

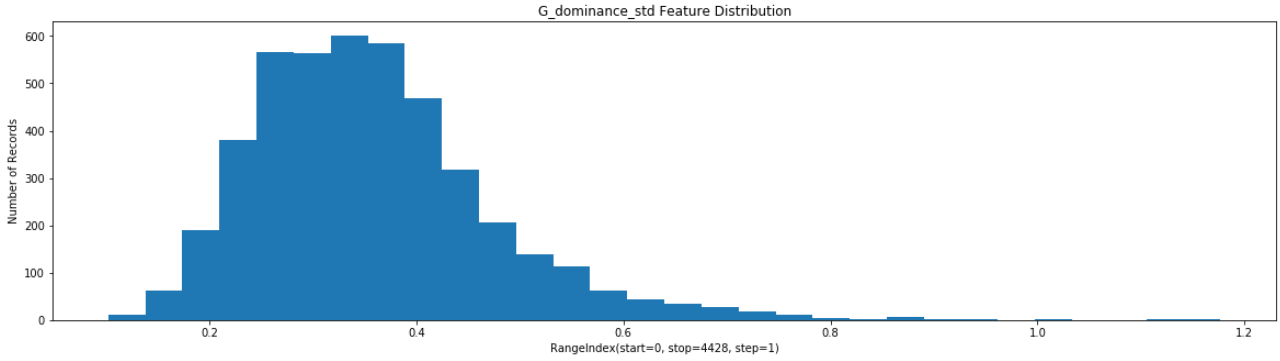

F#_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

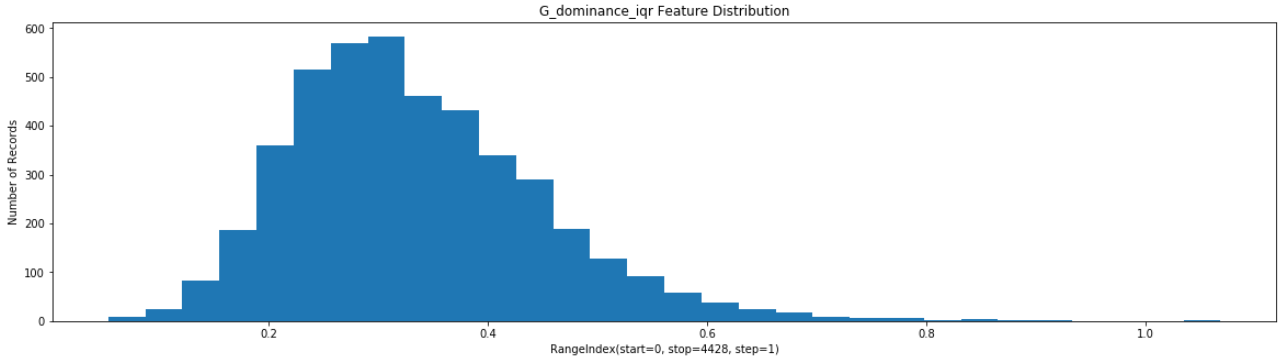

F#_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

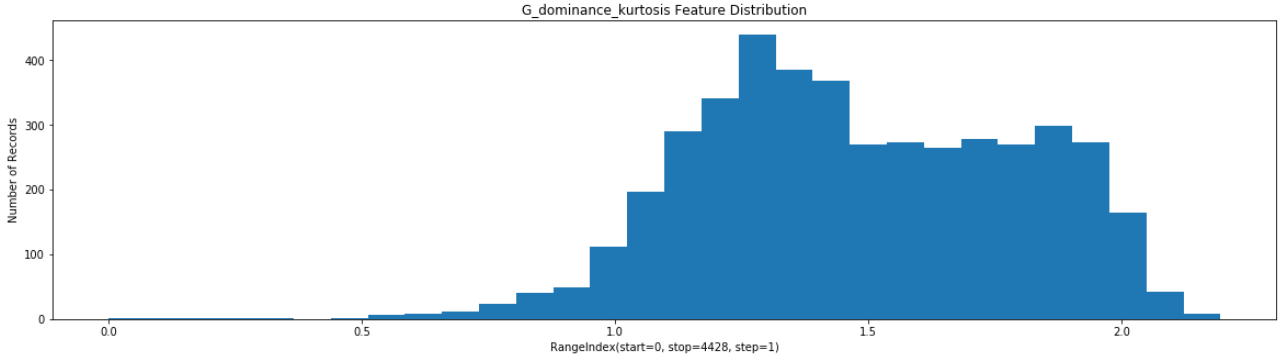

F#_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

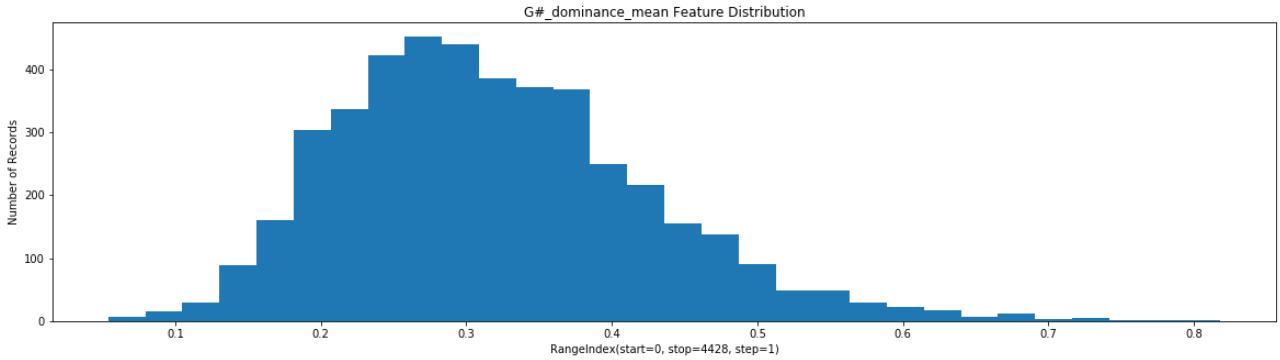

F#_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

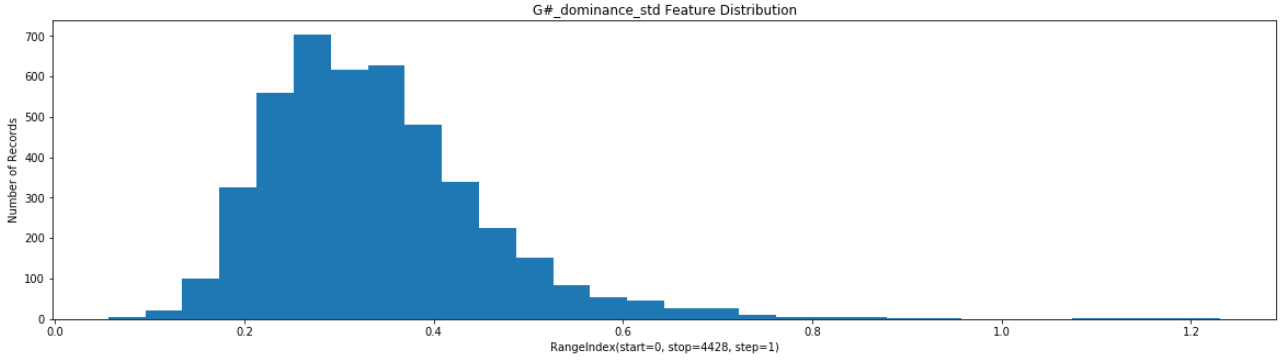

G_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

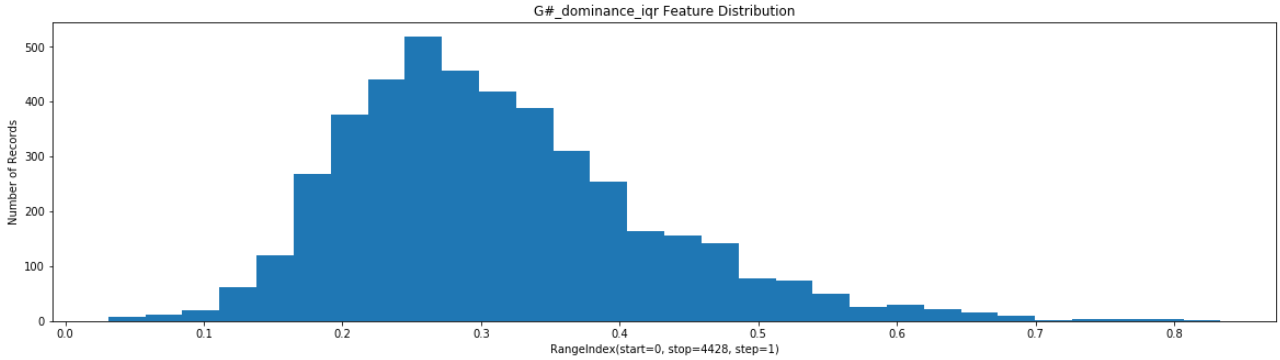

G_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

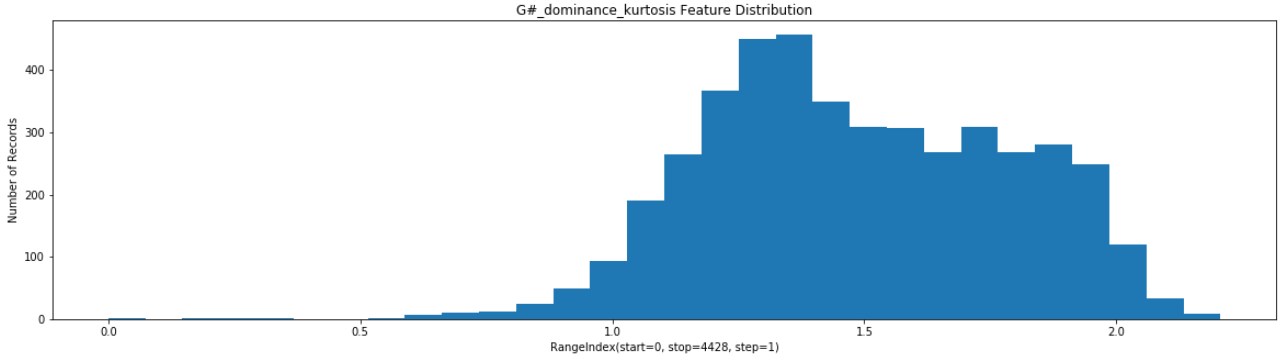

G_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

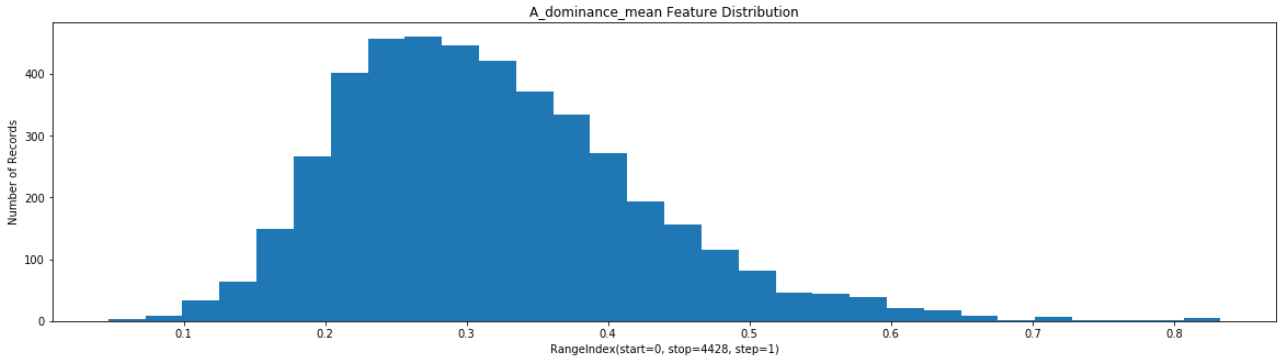

G_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

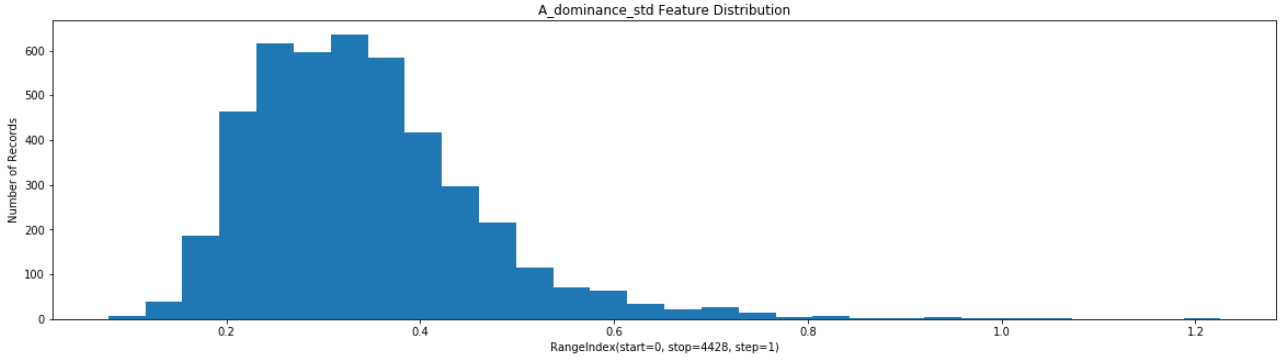

G#_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

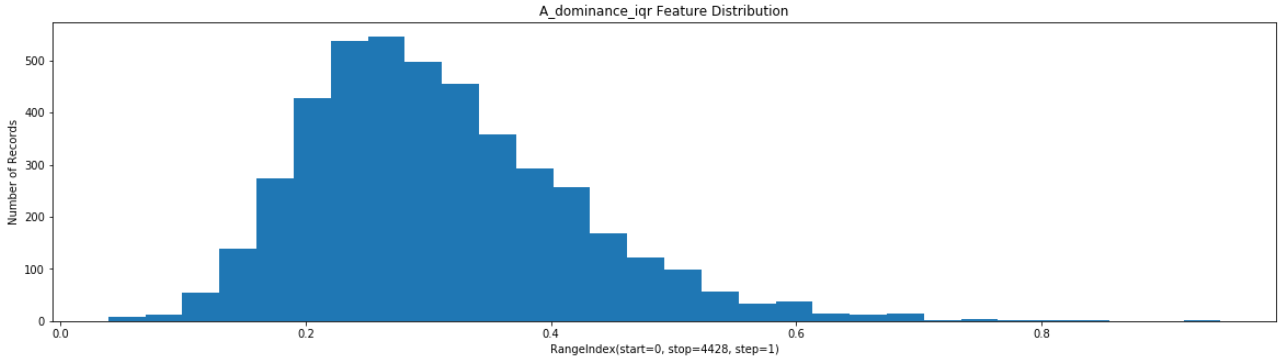

G#_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

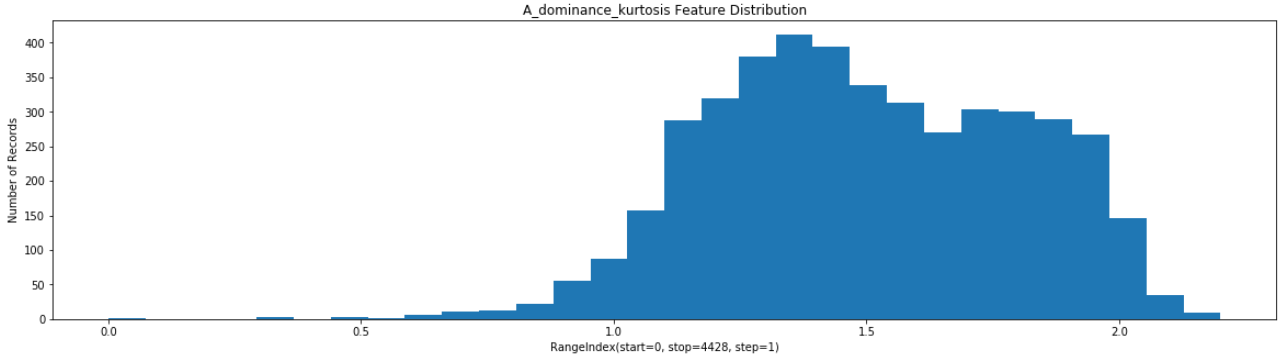

G#_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

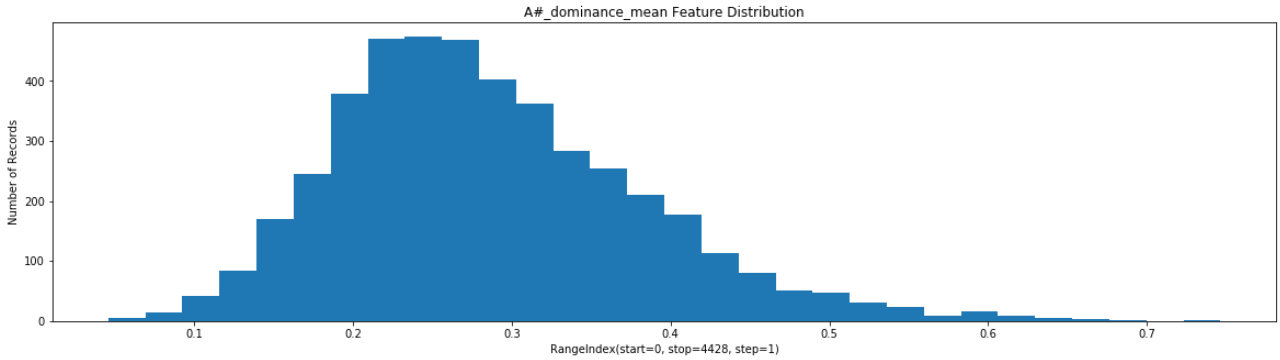

G#_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

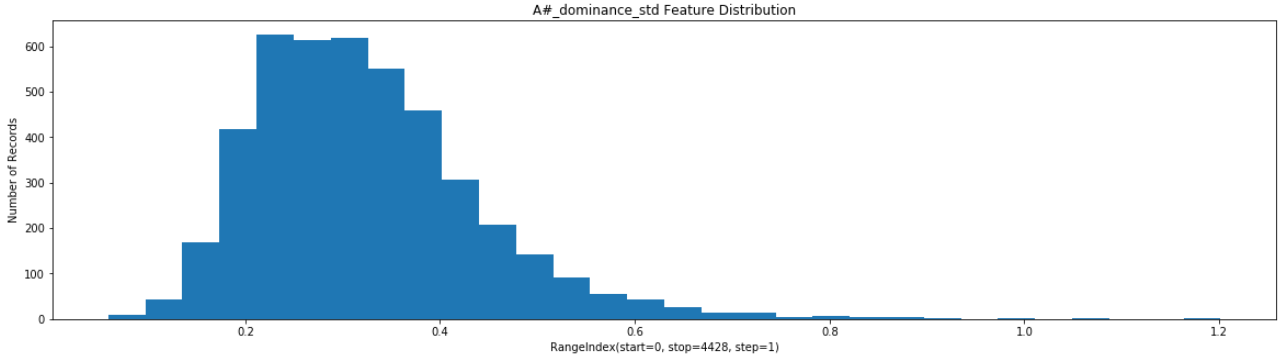

A_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

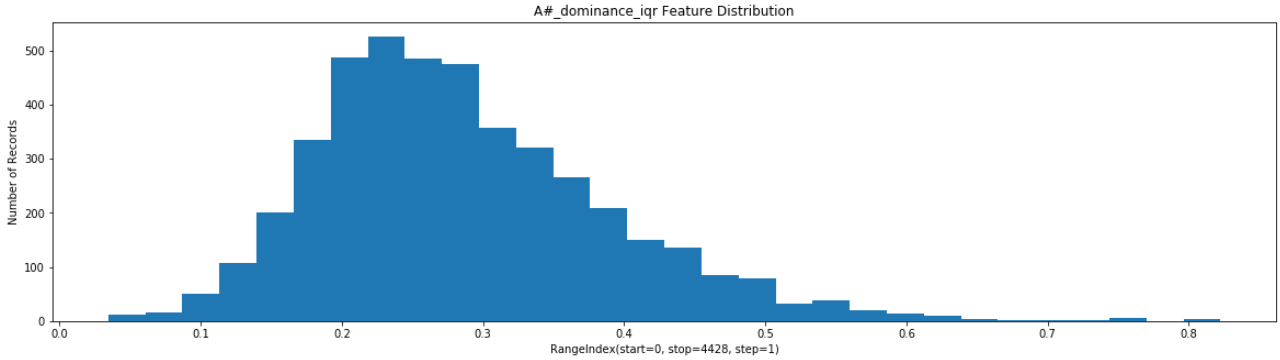

A_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

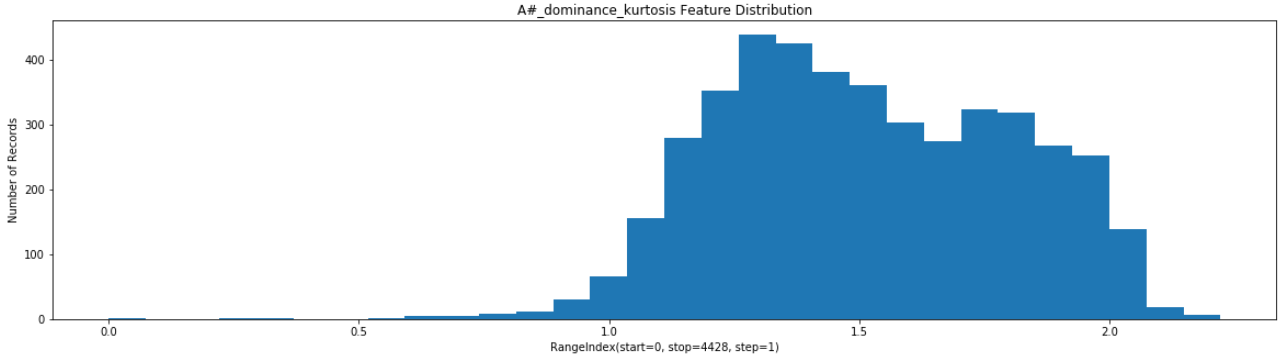

A_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

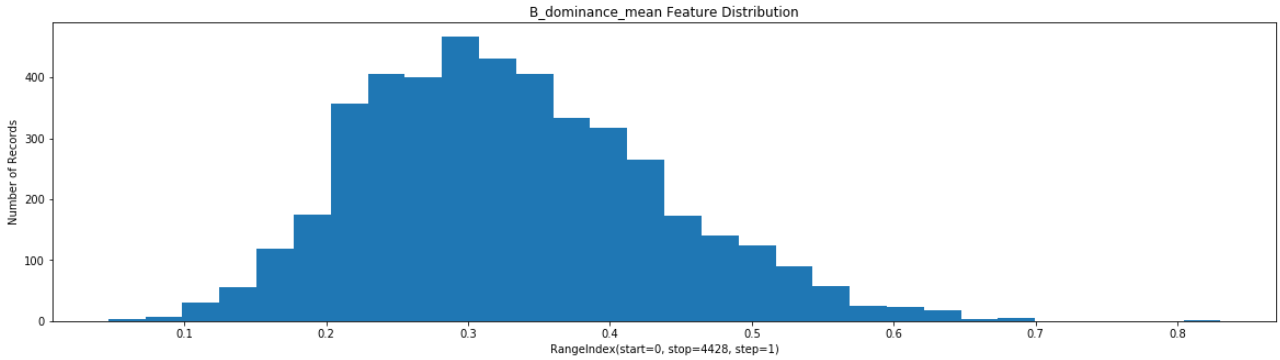

A_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

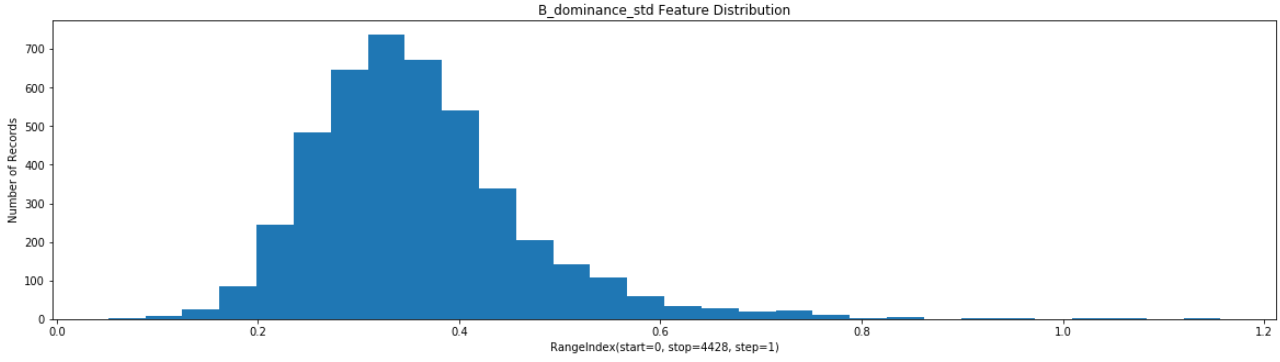

A#_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

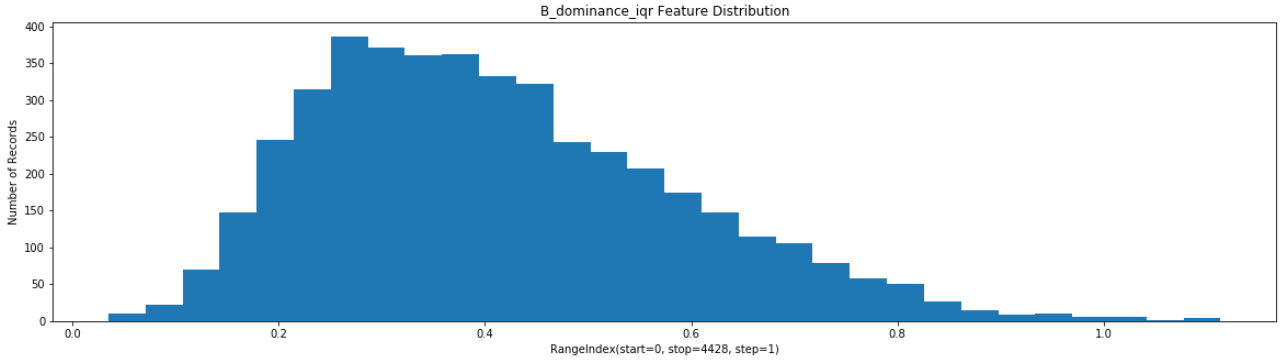

A#_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

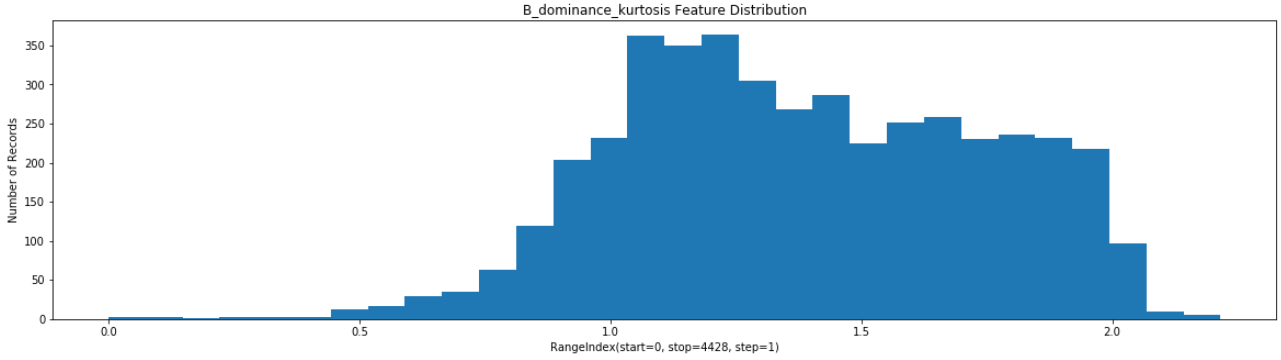

A#_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

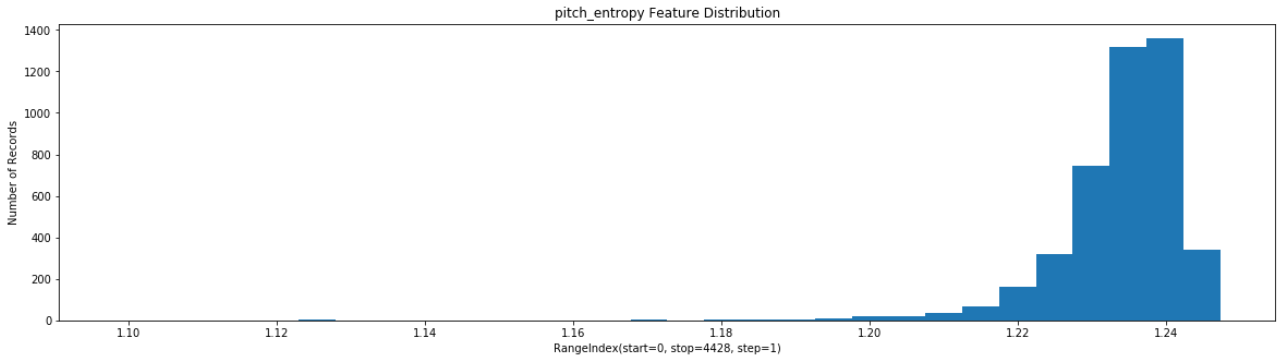

A#_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

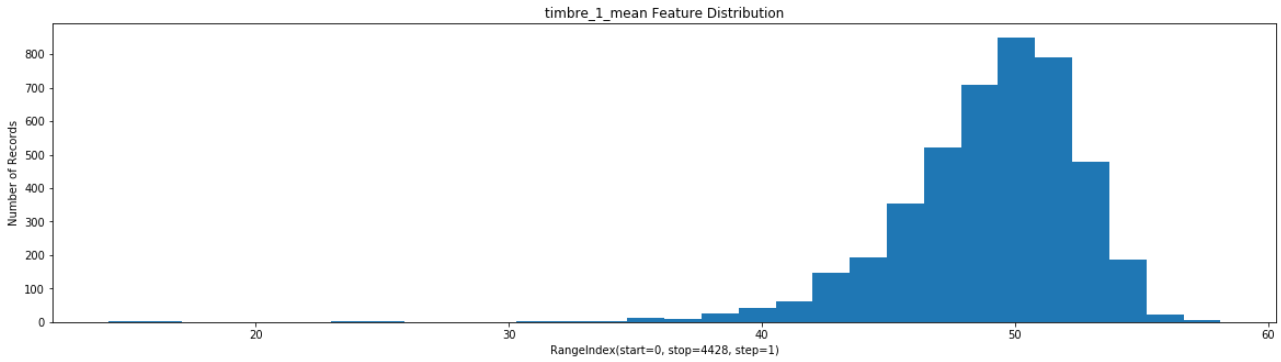

B_dominance_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

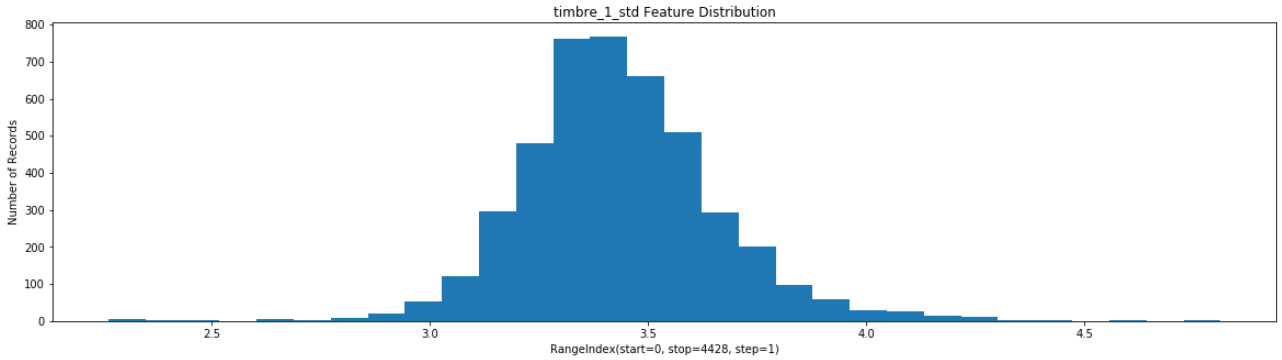

B_dominance_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

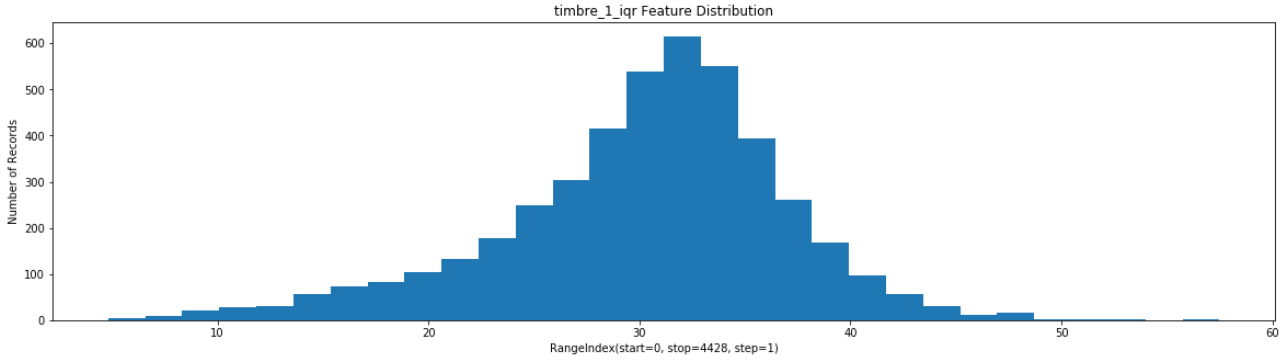

B_dominance_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

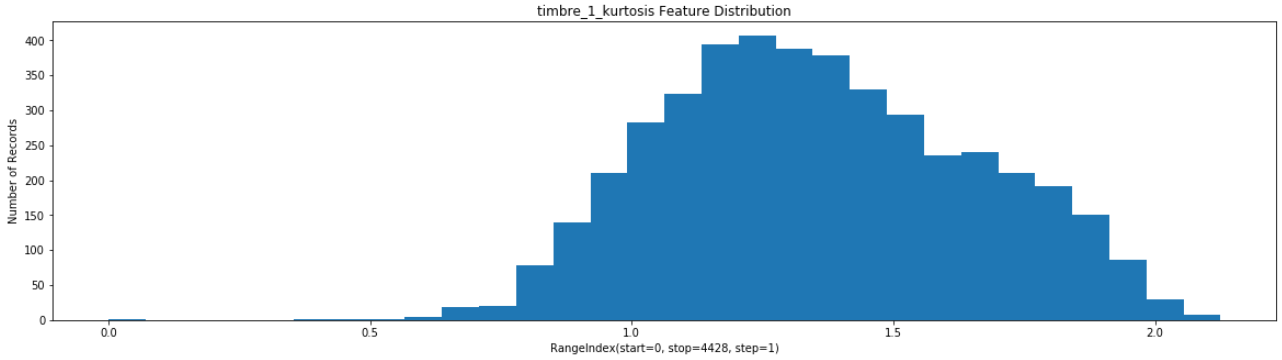

B_dominance_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

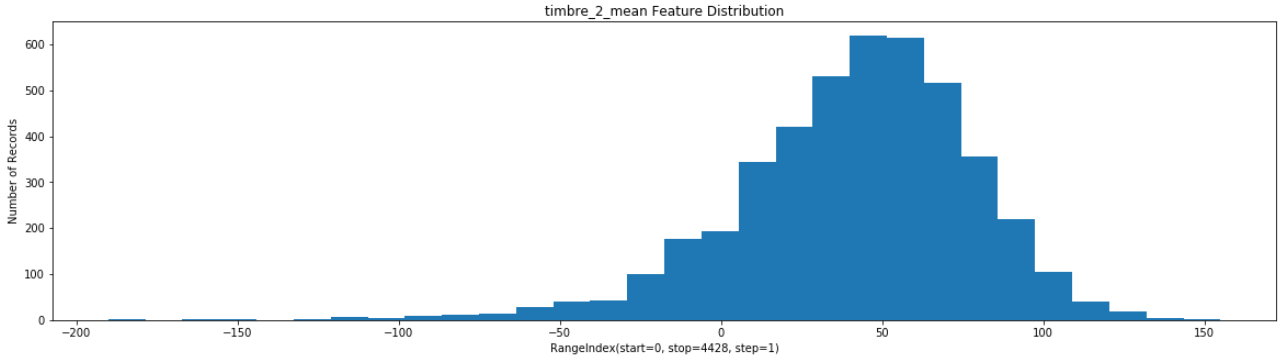

pitch_entropy Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

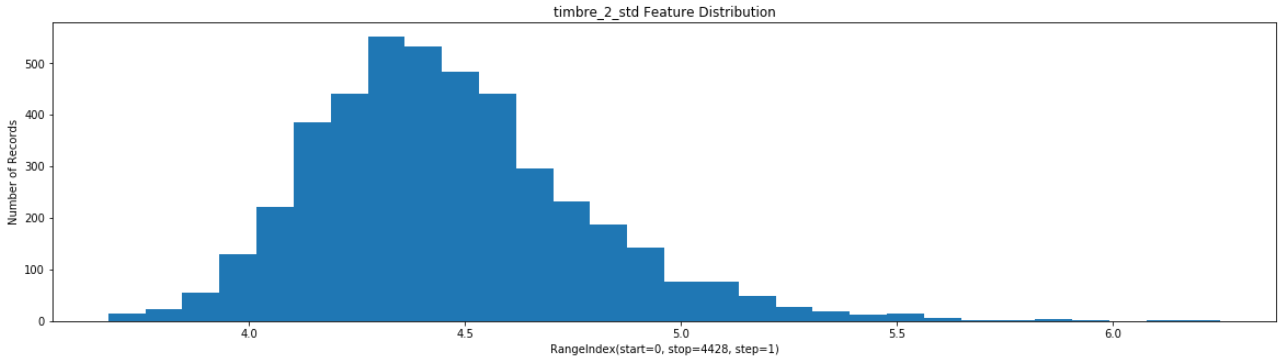

timbre_1_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

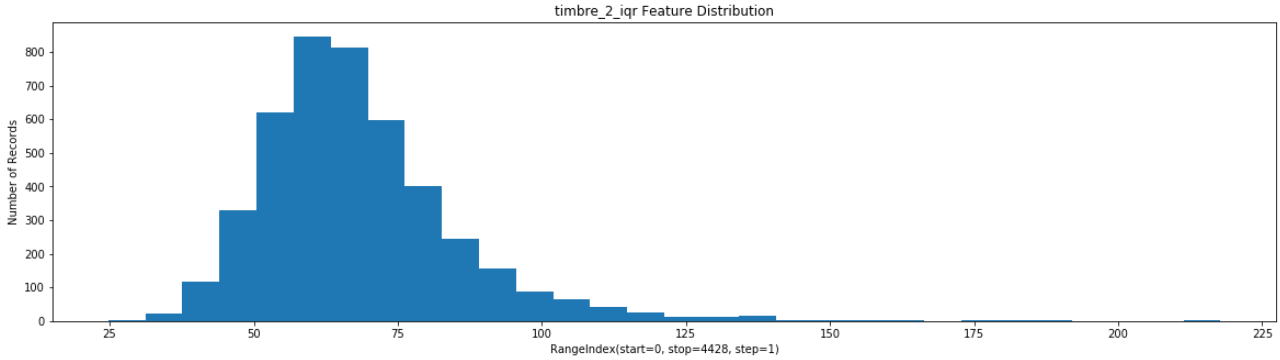

timbre_1_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

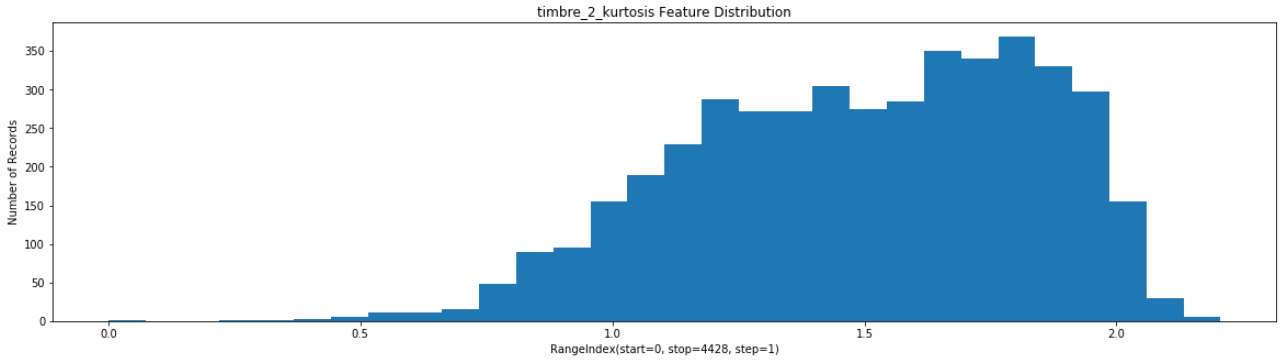

timbre_1_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

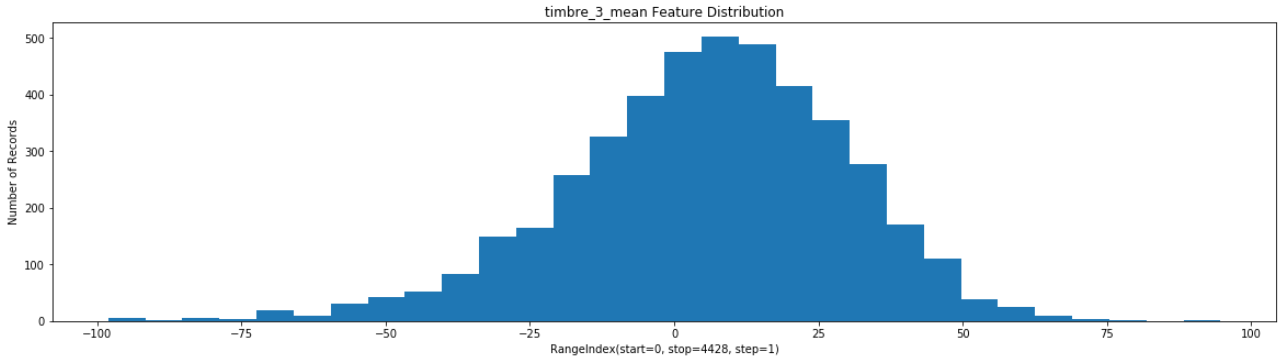

timbre_1_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

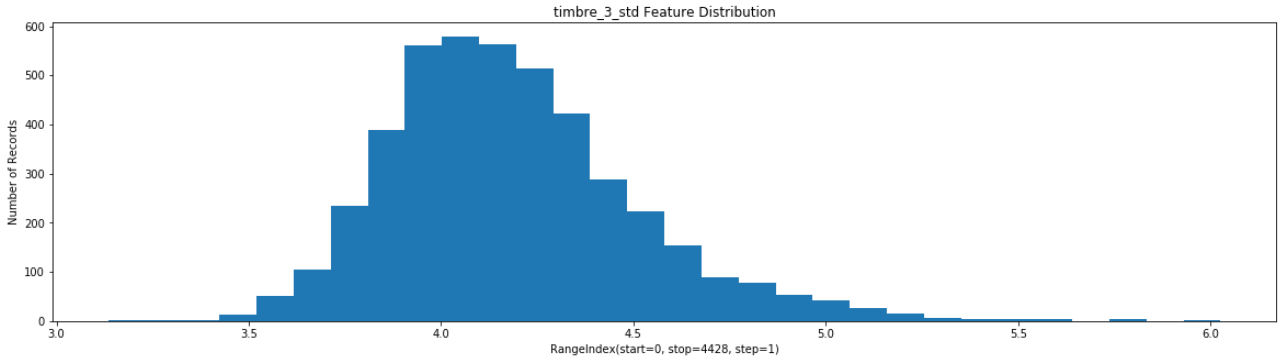

timbre_2_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

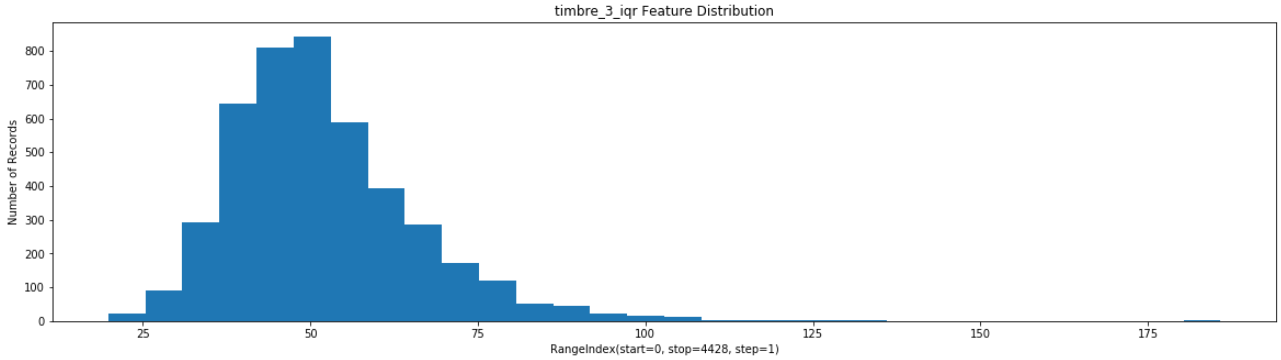

timbre_2_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

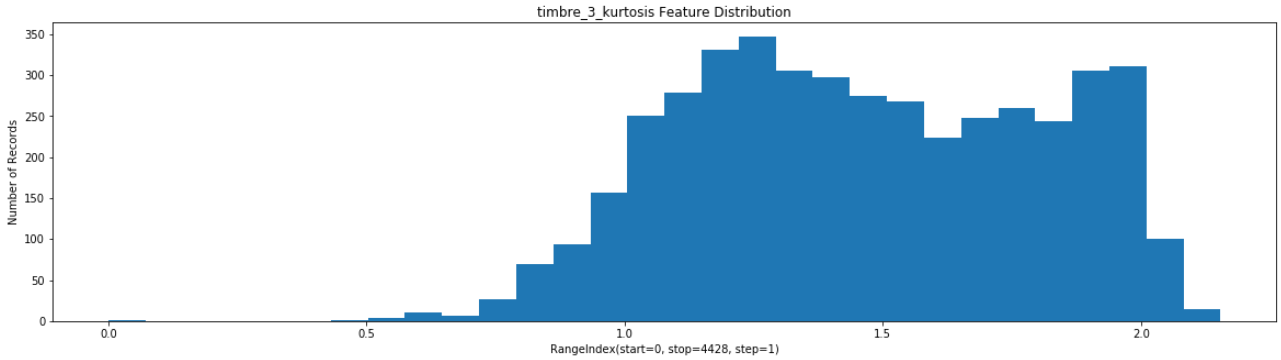

timbre_2_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

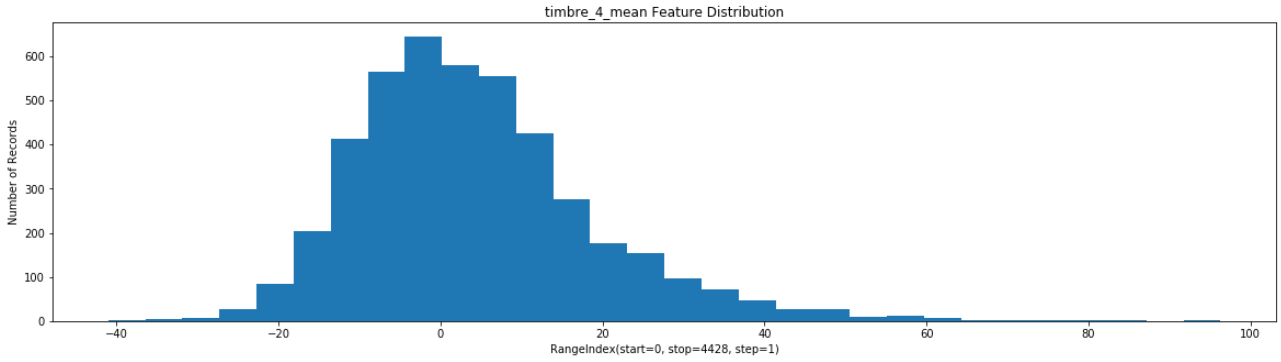

timbre_2_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

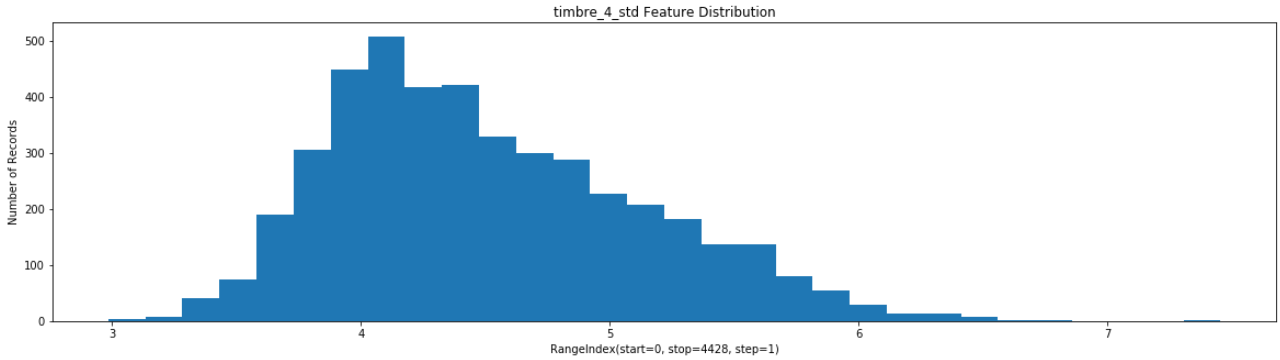

timbre_3_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

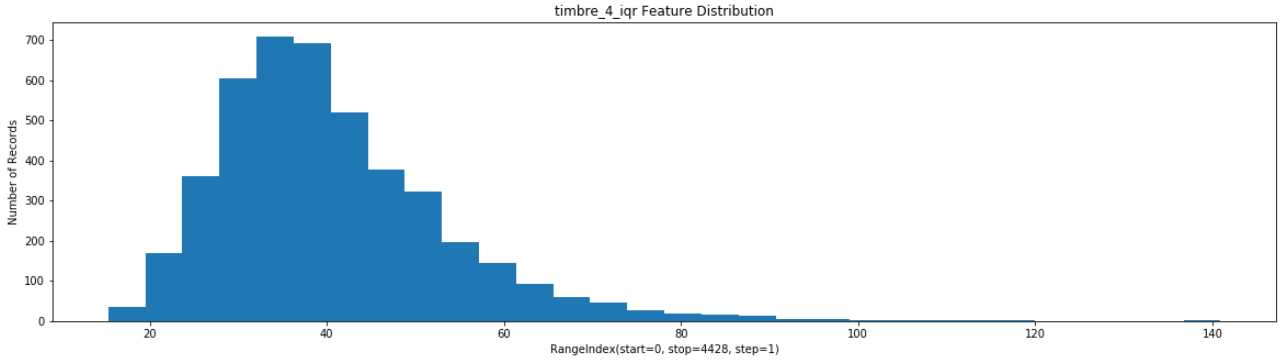

timbre_3_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

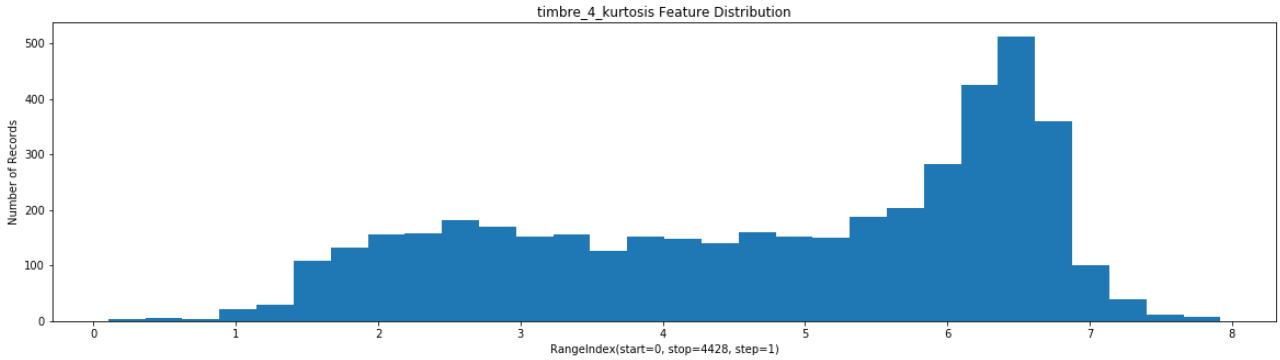

timbre_3_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

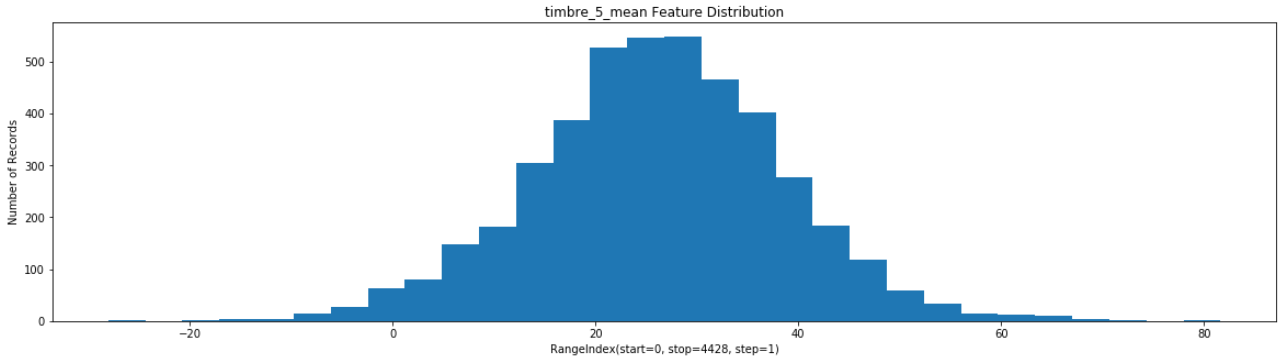

timbre_3_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

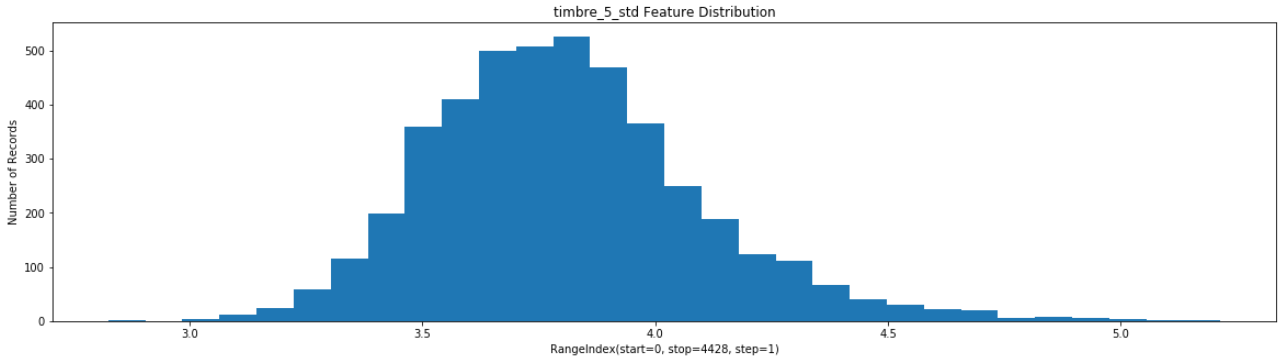

timbre_4_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

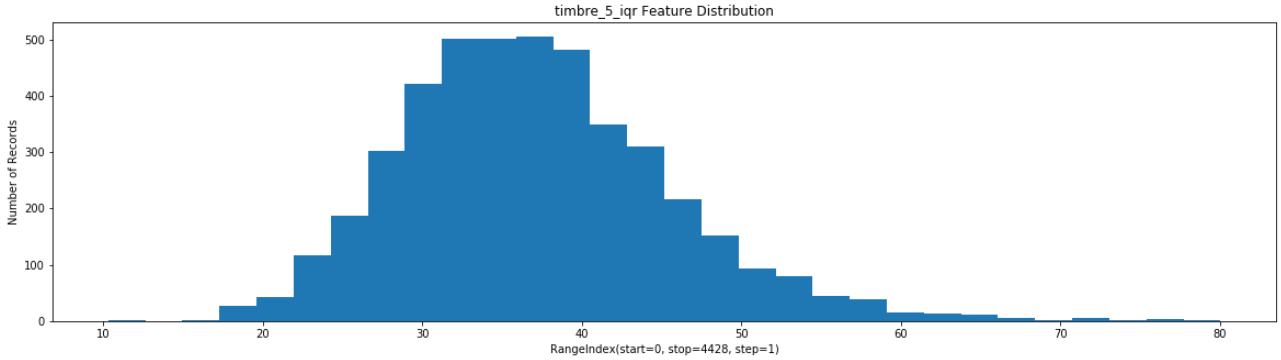

timbre_4_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

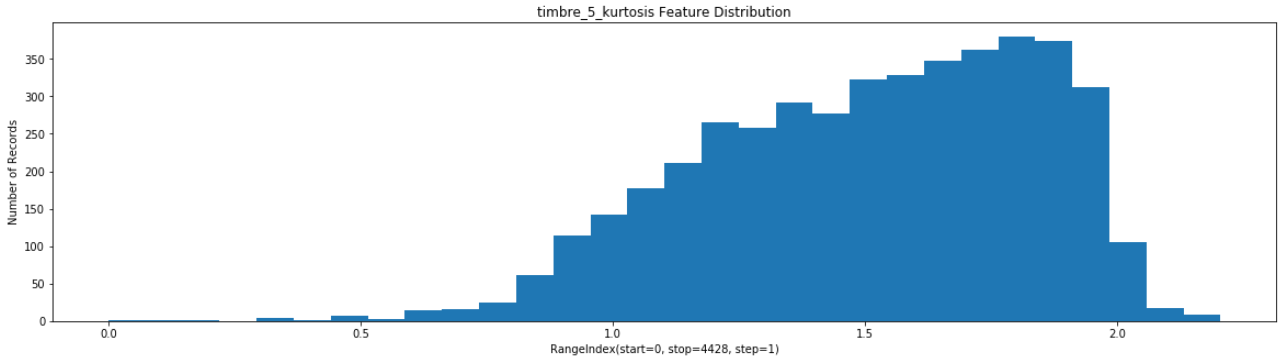

timbre_4_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

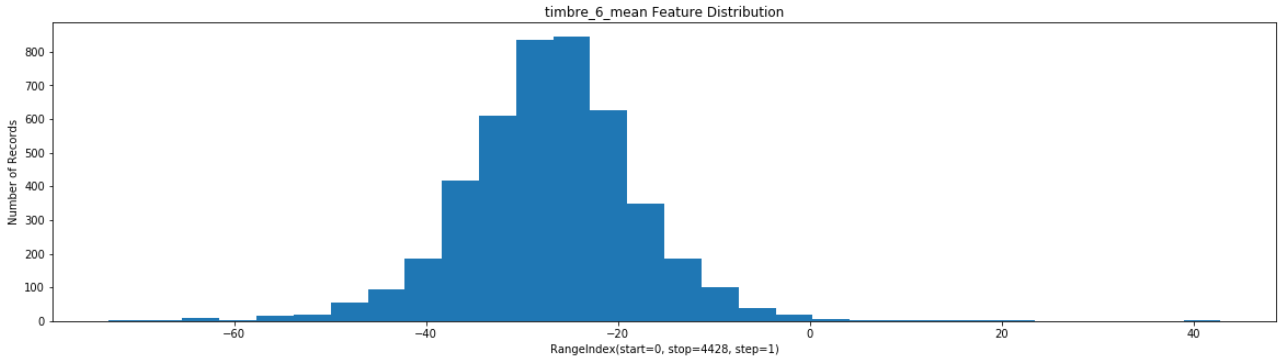

timbre_4_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

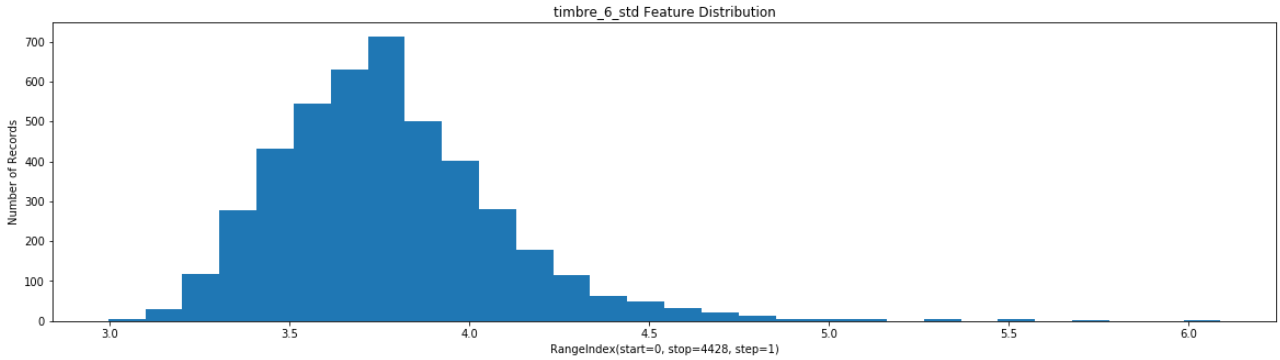

timbre_5_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

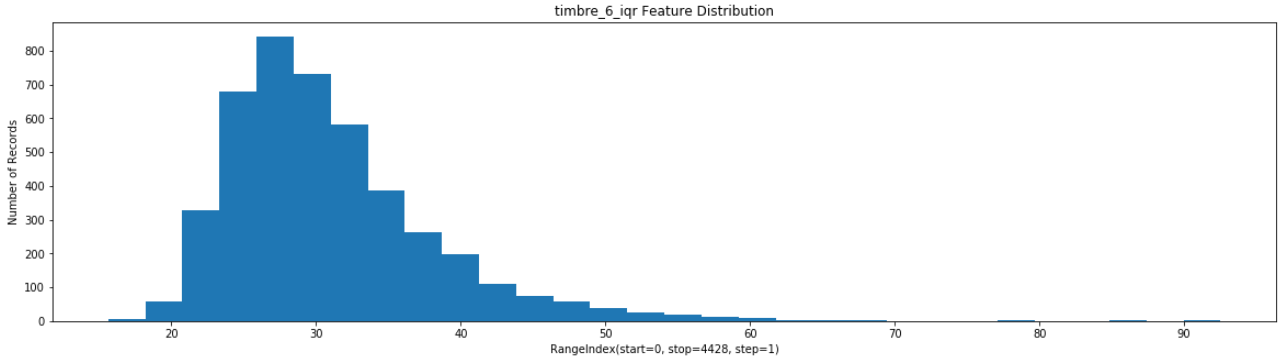

timbre_5_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

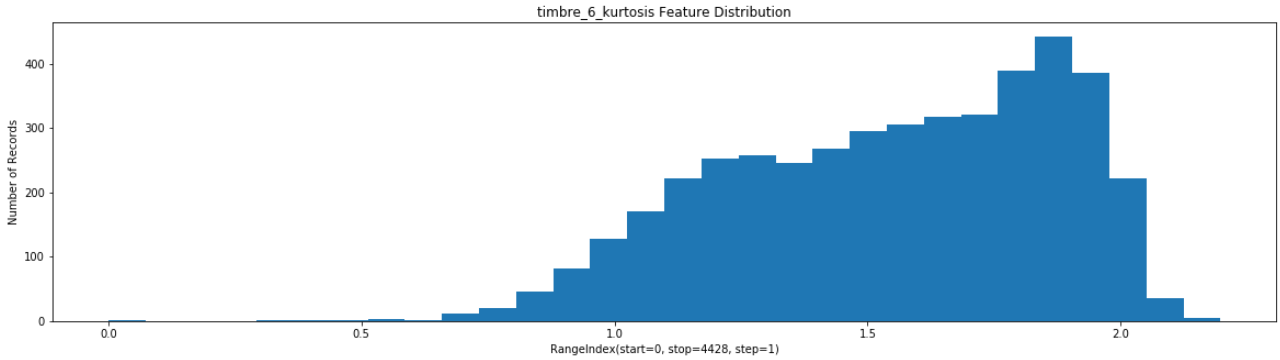

timbre_5_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

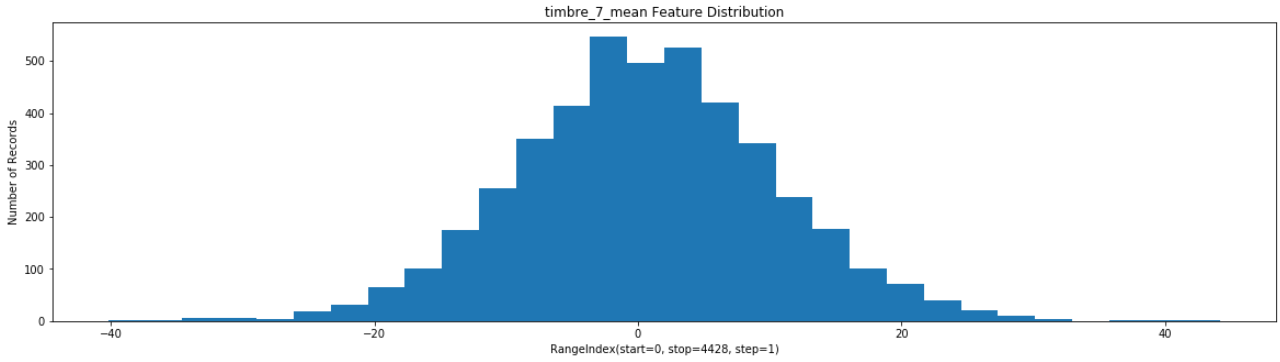

timbre_5_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

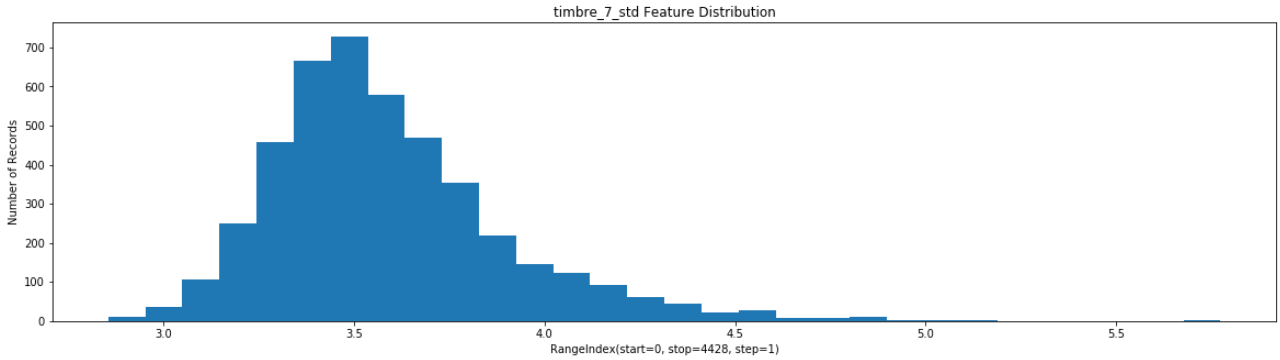

timbre_6_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

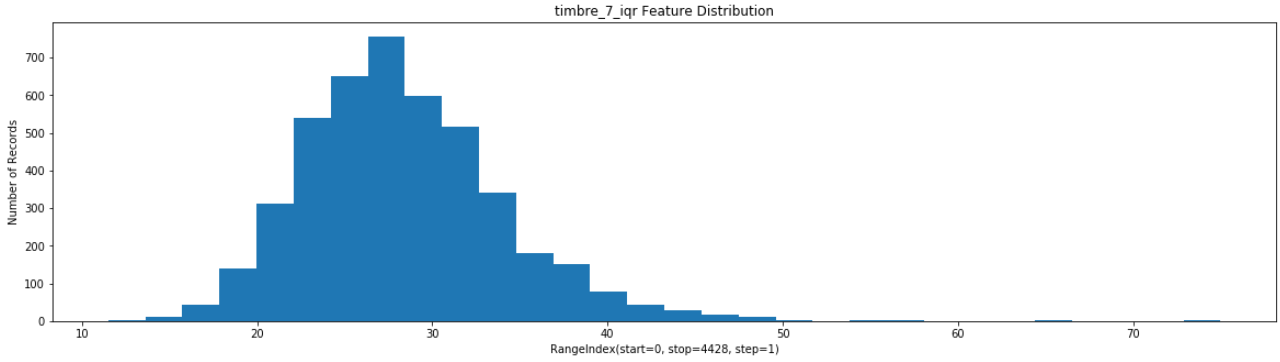

timbre_6_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

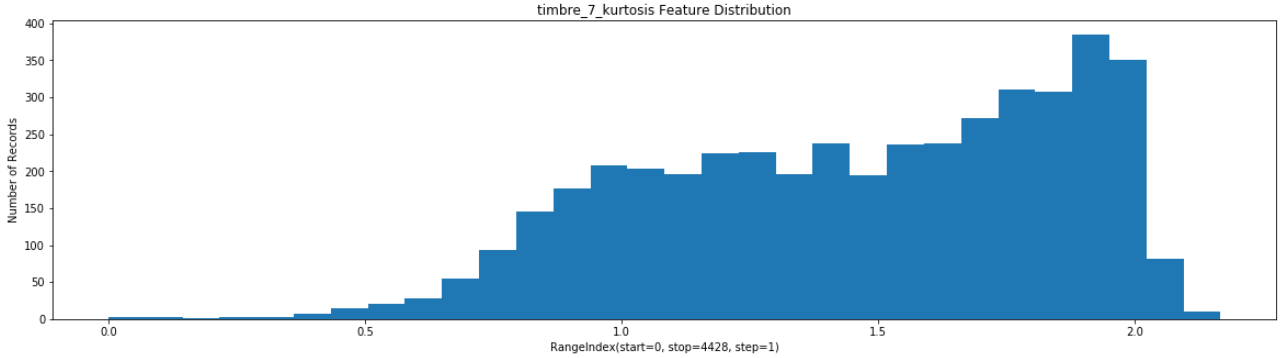

timbre_6_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

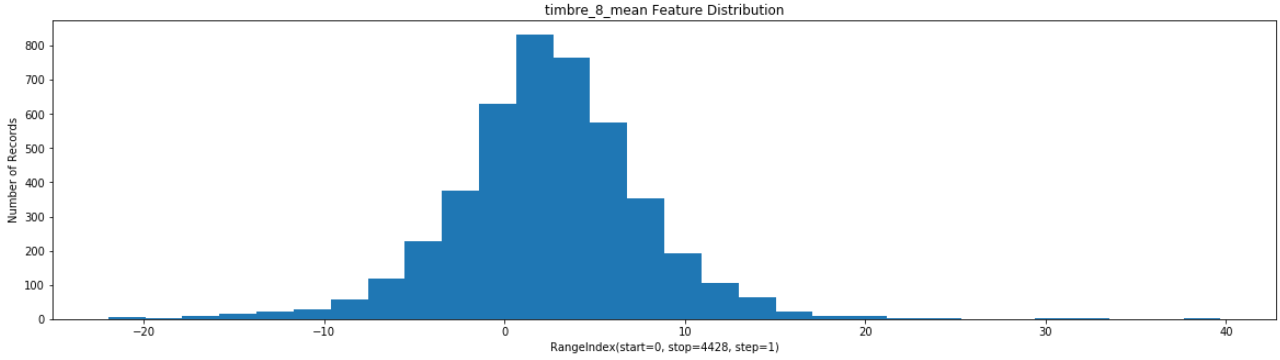

timbre_6_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

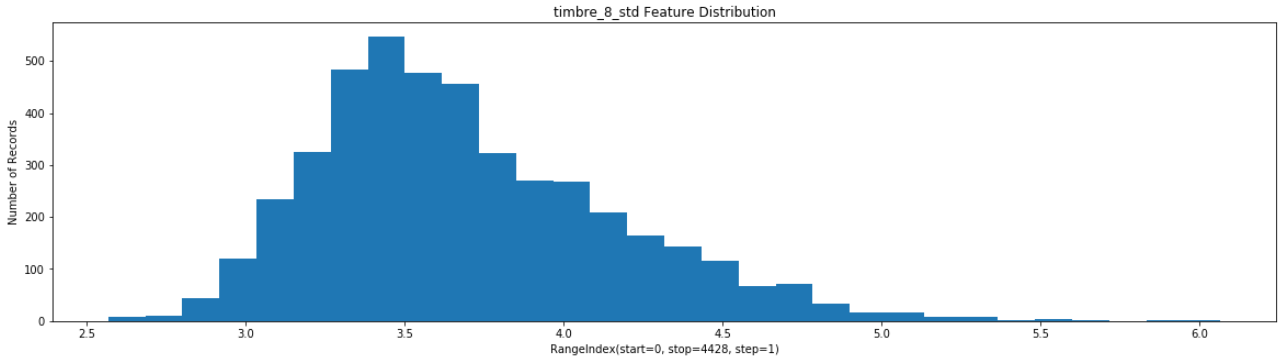

timbre_7_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

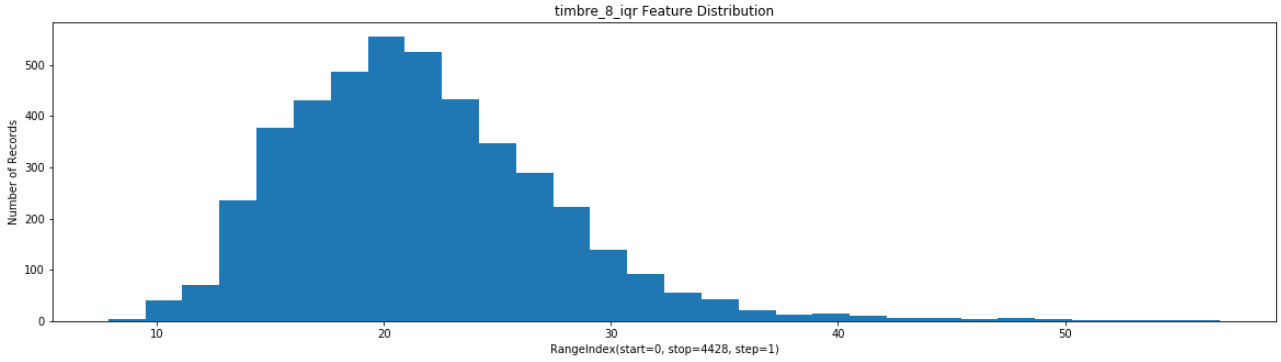

timbre_7_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

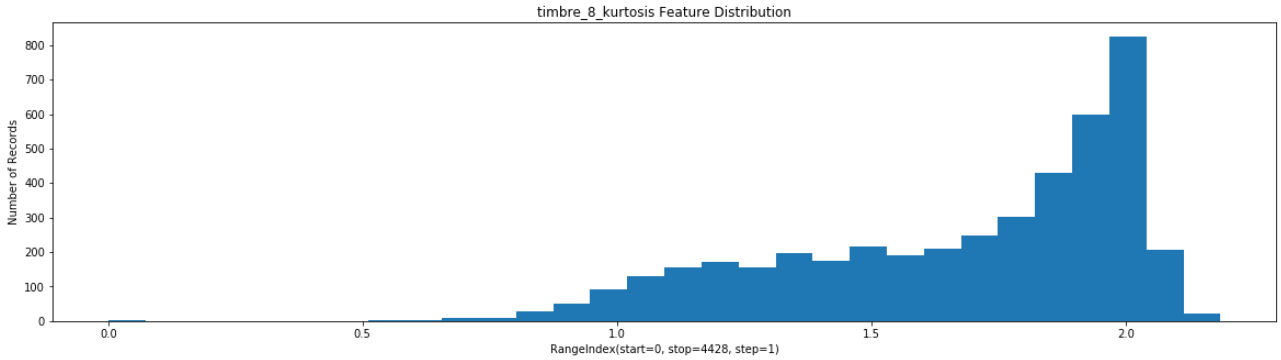

timbre_7_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

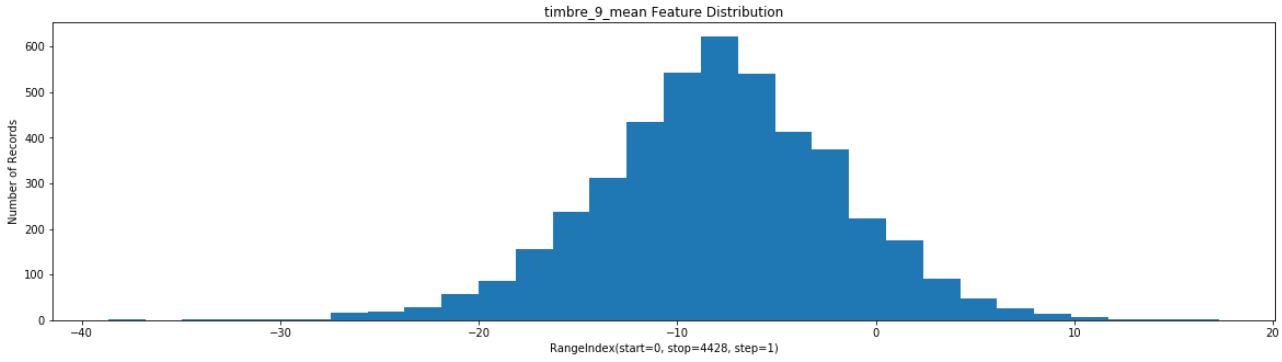

timbre_7_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

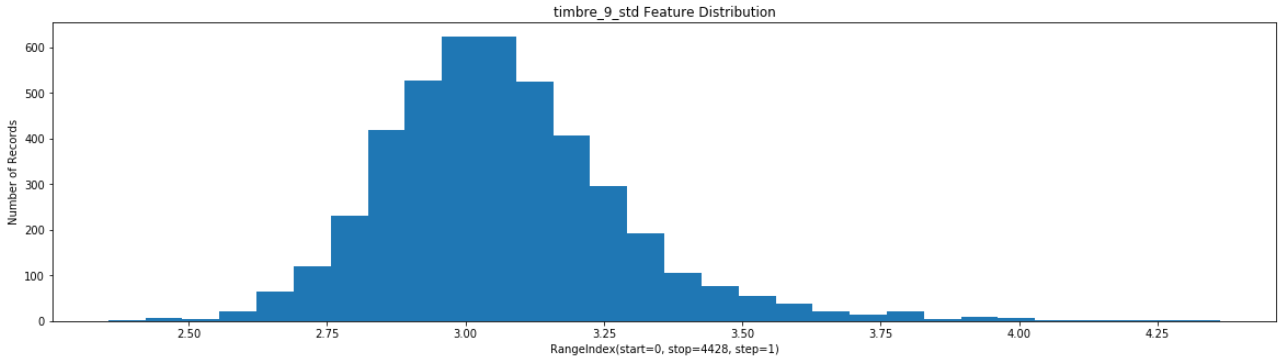

timbre_8_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

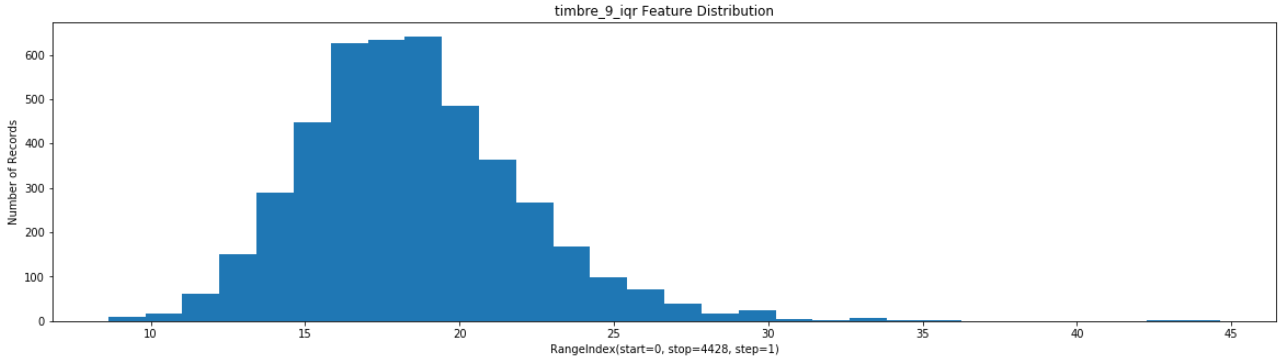

timbre_8_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

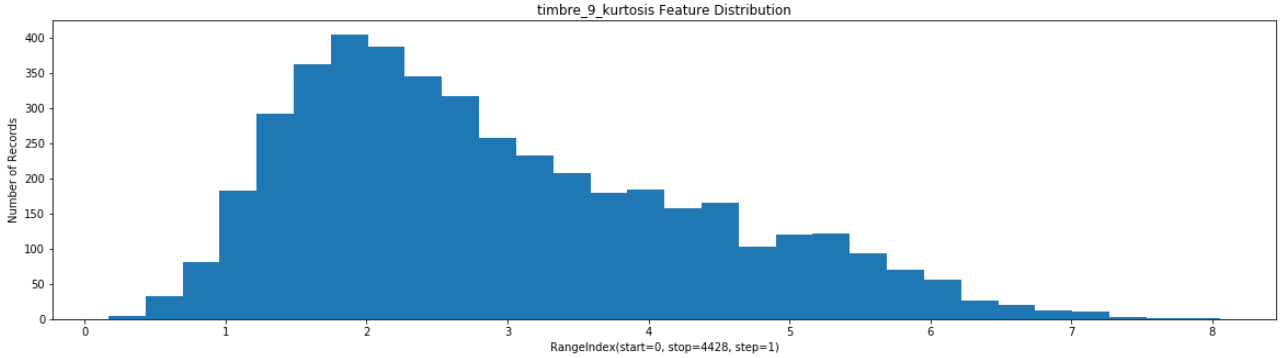

timbre_8_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

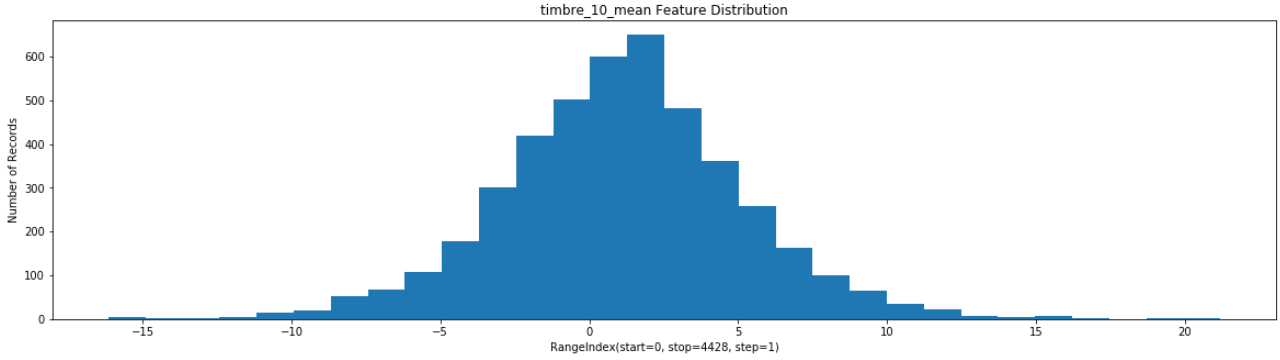

timbre_8_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

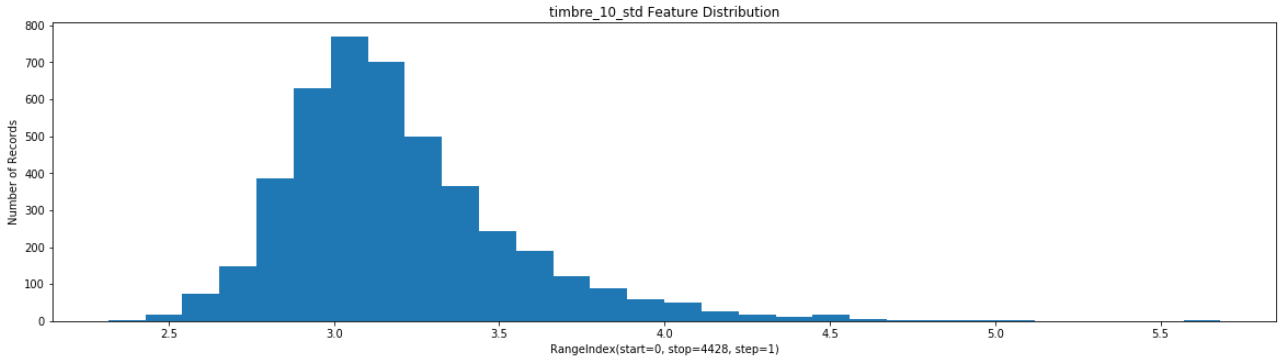

timbre_9_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

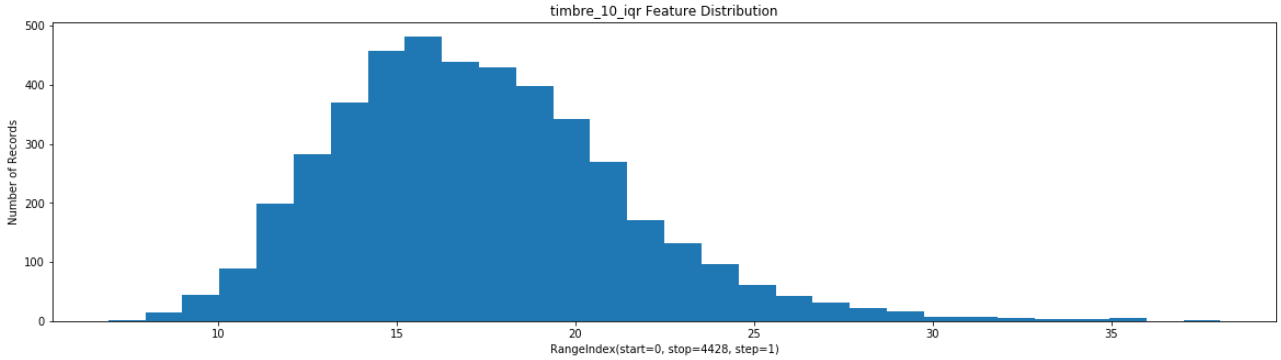

timbre_9_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

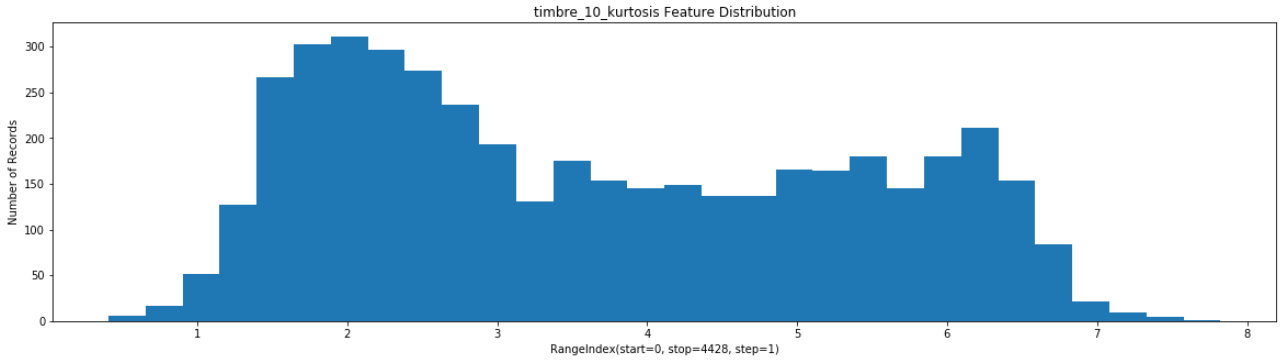

timbre_9_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

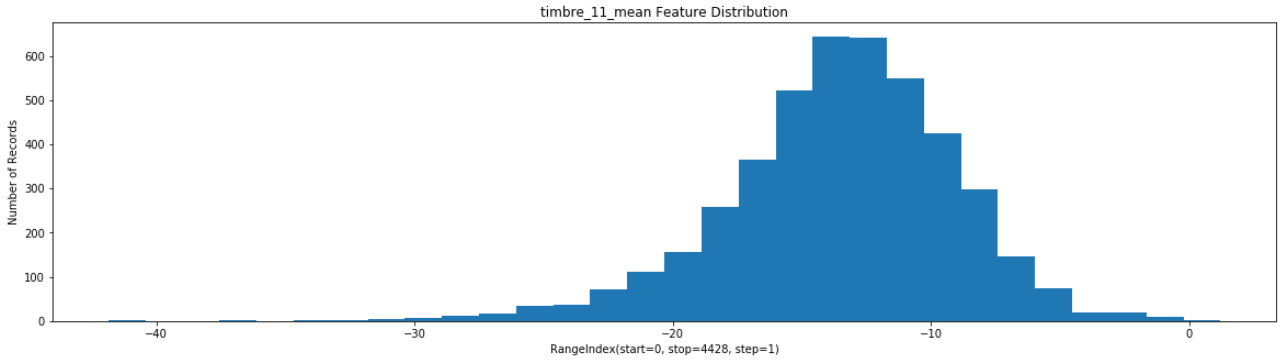

timbre_9_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

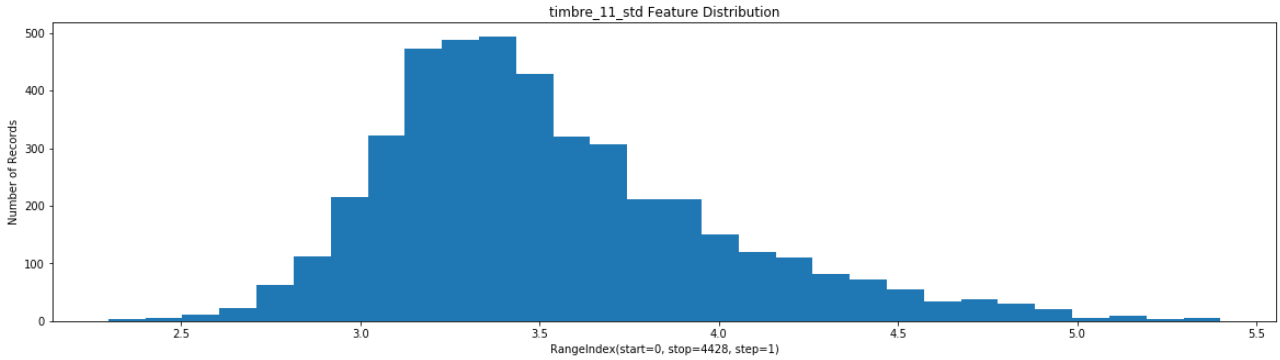

timbre_10_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

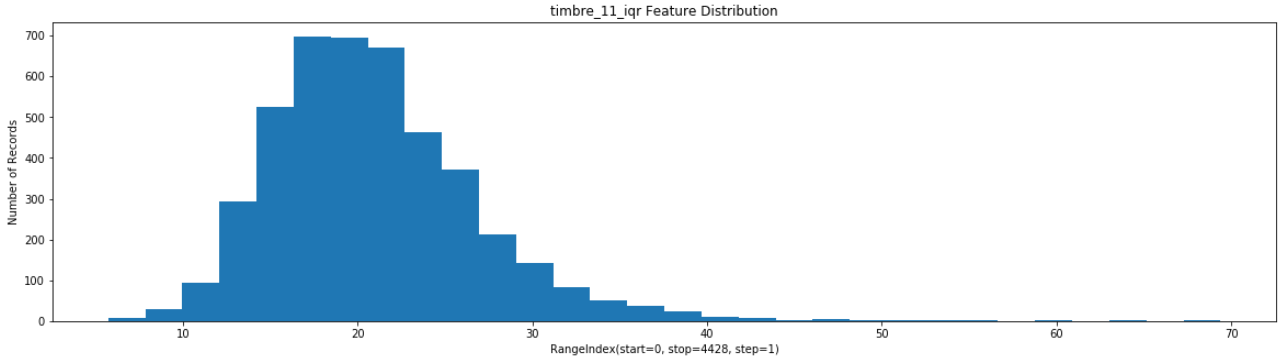

timbre_10_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

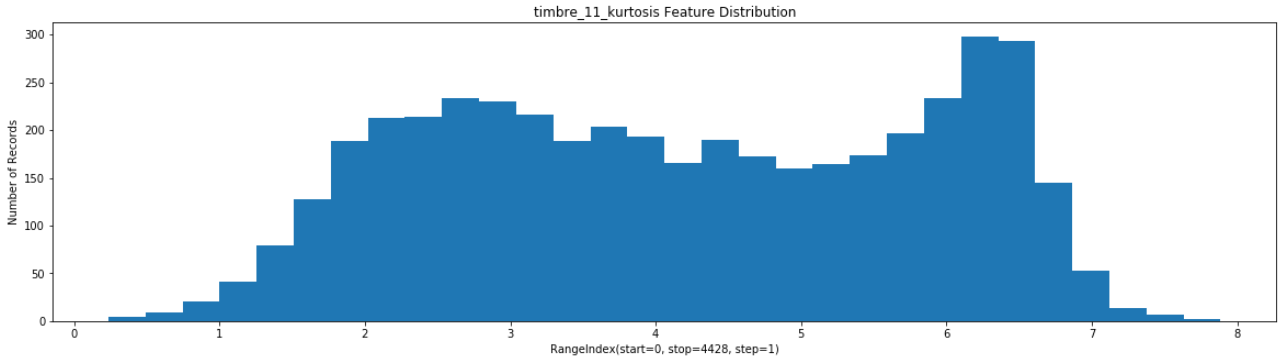

timbre_10_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

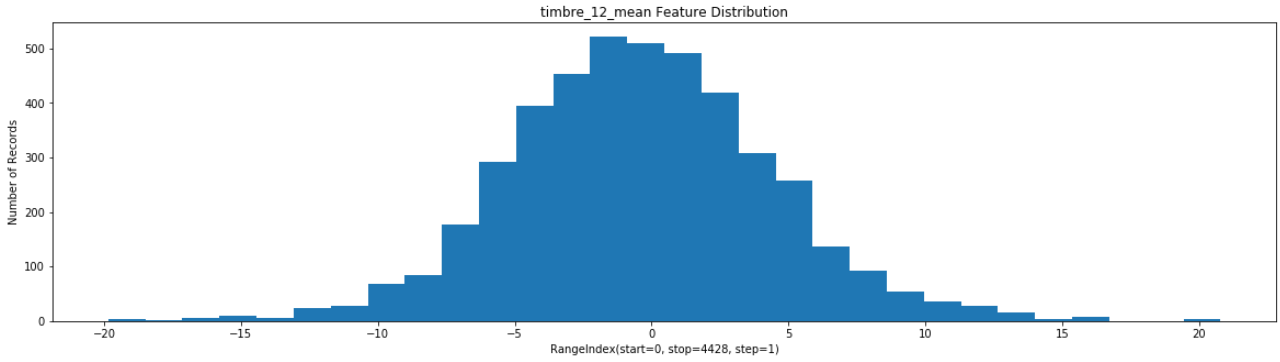

timbre_10_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

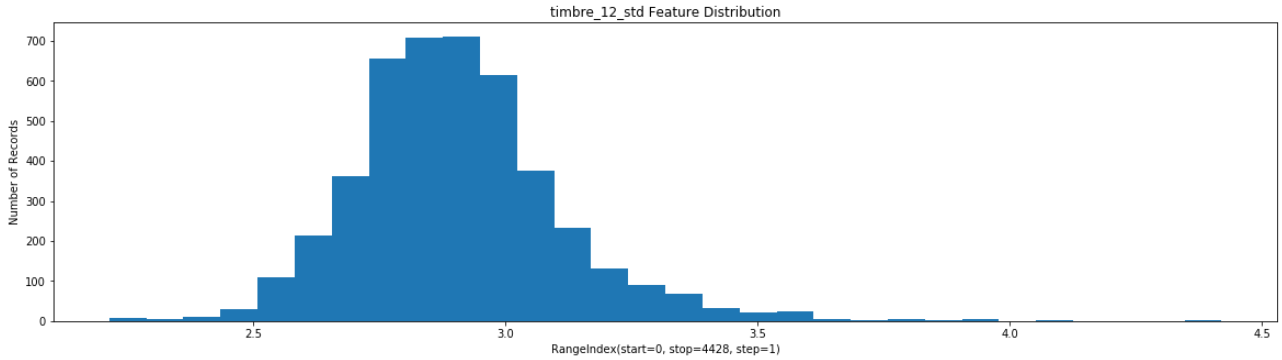

timbre_11_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

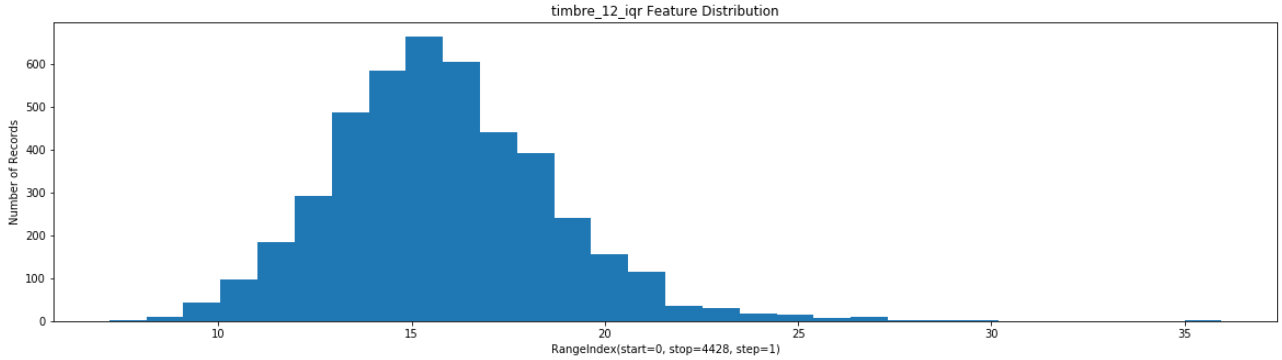

timbre_11_std Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)

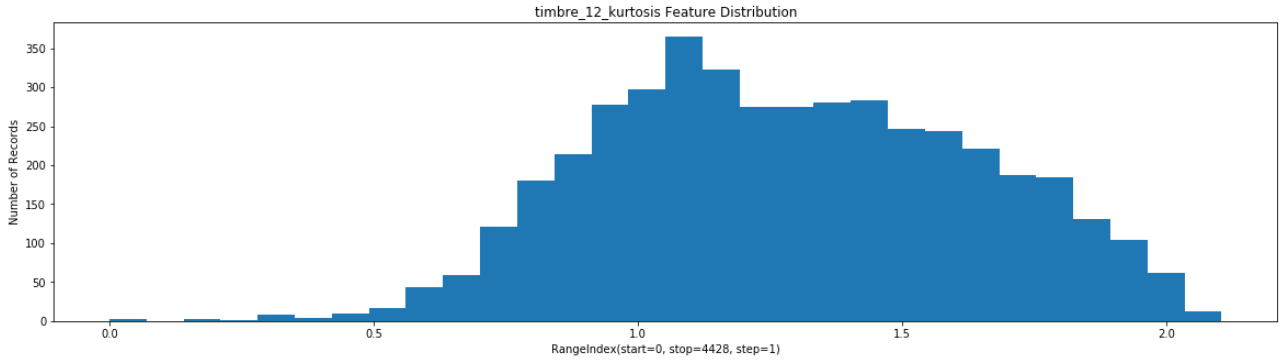

timbre_11_iqr Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)


timbre_11_kurtosis Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)


timbre_12_mean Feature Distribution
Number of Records
RangeIndex(start=0, stop=4428, step=1)




As we can see, data are much lesser skewed after log transformation.

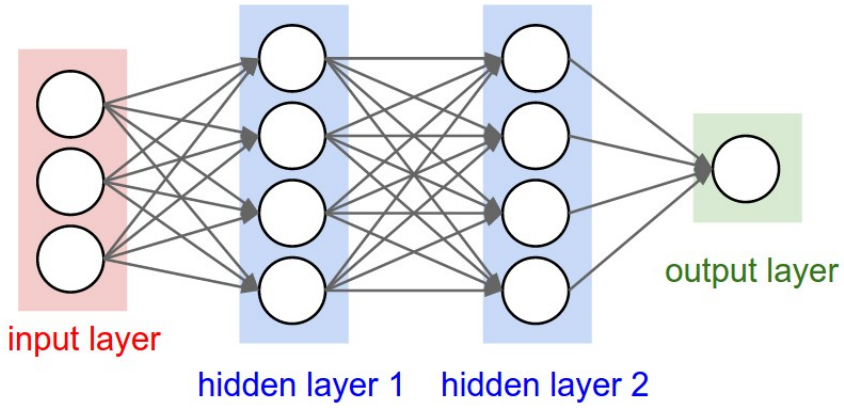
## Algorithms and Techniques

In order to predict song popularity, supervised learning models and neural network is examined. For the supervised learning models, linear regressor, logistic regressor, decision tree regressor, random forest regressor and support vector machine were examined. Using various sample size, 10, 50 and 100percent of

training dataset, all the supervised learning models were fitted and examined. Root mean squared error and R squared score were calculated. Root mean squared error was around 200 to 640 and R squared error was around 0 or negative value. Therefore, none of these supervised learning models were selected as a final model.

For following project, neural network model was chosen as final predicting model. Neural network, like how it is named, simulated our densely connected brain cells. By learning and recognizing patterns of dataset, it is able to make more accurate decisions like humans. In neural network models, there are three different units, consist of input unit, hidden unit and output unit. Input unit is where model receives information that network intends to learn and recognize about. Output unit is where model provides responds to what it learned from the input unit. Hidden units, which are in-between input unit and output unit, are consist of one or more layers connected to each other to learn and recognize patterns of input information, like how human brain is interconnected to process input data. Our model consists of fully connected layers, which every hidden units in the layers are connected to every output unit. Below is example of neural network with two fully connected layers.

Neural network with fully connected layers




Neural networks learn the pattern of data and make decision through feedback process called back propagation. Neural network model compares output produced by itself and what it should produce. In order to increase accuracy, it changes

weights of connection between each units, going backward from output layer, hidden layer to input layer. Through several back propagation, neural network improves itself to learn and recognize the patterns and reduce difference between actual and intended output. In the code, epoch indicates how many times does the model update itself to improve. For each epoch, neural network does one forward pass and one backward pass of all the training data-points.

Neural network without any refinement was able to achieve root mean squared error of 50. After several refinement, neural network with best result had configuration with 6 fully connected layers. First layer is activated with sigmoid function and composed with kernel regularizer and activity regularizer. Second layer is activated with softmax function. From third to last, all the layers are activated with relu function. Following model is then compiled with mean absolute error loss function, Adam optimizer and metrics of mean squared error and mean absolute error. Neural network was fitted with both preprocessed data and non-preprocessed data (not log transformed and not normalized) both through 100 epochs. As a result, neural network performed better with fully preprocessed dataset. Neural network was able to achieve root mean squared error of 18.73 with non-preprocessed data and root mean squared error of 16.16 with preprocessed data. Therefore using preprocessed data is selected for final model.

Preprocessing dataset was one big part to increase performance of the model. By log transforming skewed features, I was able to reduce affect of values caused by outliers. After log transformation, distribution of skewed features were normalized. Also, MinMax scaling for all the features were performed to ensure that each feature is treated equally when applied to supervised learning models and neural network model. This process of preprocessing dataset was able to achieve higher performance of final model.

### Benchmark

At first, song popularity predictor by Mohamed Nasreldin, Stephen Ma, Eric Dailey, and Phuc Dang was chosen as benchmark model. However, song popularity predictor used area under curve (AUC) as their metrics, which is not a great choice for regression problem. AUC is a good metric for a classification models. I was planning to run their code with root mean squared and R squared metric. However, code provided by Mohamed Nasreldin does not include complete dataset named cleaned_million.csv. Therefore, it was not possible for me to run the code and obtain values for comparison.

Instead, benchmark result has been set to get root mean squared error value less than 20. This is because track popularity value is ranging from 0 to 100. For the model to be performing reasonably, maximum squared error value should be less than 20 percent of range of final output.

## III. Methodology

### Data Preprocessing

Data is first collected from Spotipy. For each song, audio analysis data are given with each data points divided into segments. As mentioned before, each segment does not have same time length. As mean, standard deviation and kurtosis had to be calculated, each data points are balanced out by multiplying weight to each point. In following preprocessing, weight is set to be duration of segment divided by average segment duration.

After each data points are balanced out, mean, standard deviation and kurtosis are calculated. In result, mean, standard deviation, and kurtosis of confidence, loudness, tempo, tempo confidence, key, key confidence, mode, mode confidence, time signature, time signature, pitch dominance, and timbre are calculated.

When distribution of these audio analysis data are represented, some of the features are highly skewed. (As shown in Exploratory Visualization) Therefore, highly skewed features are log transformed.

During log transformation, as log is not defined for negative and 0 value, if all the data for a feature is positive, log(x+1) is applied. Otherwise, if any of the value in the feature includes negative value, log(absolute(feature minimum value)+x+1) is applied.

## Implementation

For supervised learning models, each models are examined with 10, 50 and 100 percent of training sample set. This is to see if there is overfitting due to size of training set. Root mean squared error and R squared score is reported for both testing set and training set. In addition, time spent for training and predicting is reported.

When all the supervised learning models are performed, root mean squared error and R squared error reported showed that all the supervised learning models performed very poorly. Root mean squared error range from 200 to 640, which is too high. Also, R squared score, which 1 indicated perfect fit and 0 indicates poor fit, is around 0 to negative value. This means that models are not working at all.

Neural network, on the other hand, is composed with 6 fully connected layers. First layer is activated with sigmoid function. Second layer is activated with softmax. Lastly, from third to last layers, all the layers are activated with rely function. Neural network model is compiled with loss function of mean absolute error, Nadam optimizer and using mean squared error and mean absolute error metrics. Opposed to other supervised learning models, neural network was able to achieve root mean squared error of 18 to 50. Therefore, Neural network is selected as final predicting model.

One of the major difficulty I faced with coding was achieving feature importance from neural network. For following project, neural network by Keras was chosen to be final model. However, Keras does not offer feature importance as built in function. Therefore eli5 was instead used to extract feature importance. Eli5 package provides permutation importance module for scikit-learn model, and Keras provides wrapper for sequential models. Using both modules together I was able to extract feature importance.

### Refinement

In order to refine the neural network model, different number of fully connected layers had been tried. From 2 layers to 8 layers, different number of layers had been tried and 6 layers had been chosen at last. When more layers are added, root mean squared error increased, this could be due to overfitting. Also when layers are reduced mean squared error increased, which would be due to underfitting.

Several activation configuration had also been examined. Relu, sigmoid, softmax, and tanh function had been tried out. Sigmoid function at first layer, softmax function at second layer and relu function at last layer as configuration has given best result. As a result, neural network was able to achieve root mean squared error from 45 to 18.

Also, comparing preprocessed data and non-preprocessed data, root mean squared error was able to reduce from 18 to 16.

## IV. Results

### Model Evaluation and Validation

By neural network refinement and data preprocessing such as log transformation and normalization, final model was able to achieve root mean squared error of 16. Compared to other supervised models, linear regressor, logistic regressor, decision

tree regressor, random forest regressor and support vector machine, neural network was able to achieve much lower root mean squared error. Supervised models achieved root mean squared error of 200 to 640, which was very high. On the other hand, neural network achieved root mean squared error of 50. Therefore, neural network was chosen as final predictor model. In order to improve further more, several configuration was examined and 6 fully connected layers with sigmoid, softmax and relu activation was chosen. For both supervised learning models and neural networks, preprocessed dataset and non-preprocessed dataset were fitted and examined. Both supervised learning and neural network models were able to achieve lower root mean squared error with fully preprocessed dataset. For neural network model, using preprocessed dataset reduced root mean squared error from 18 to 16.

Also, through testing out with several training set size, predicting model with larger training set predicted better than those with smaller training set. This indicates larger dataset would improve our predicting model. As range of track popularity is 100 and root mean squared error is around 16 and mean absolute error of 12, we are able to conclude that model is able to provide reasonable prediction.

From our final neural network model, top 20 important features were extracted, which is shown below. Most important feature for a song to be popular was artist popularity.  Next most important feature were timbre. Out of top 20 important features, 14 features were related to timbre. Timbre represents quality of sound or musical note. Mentioned from Spotify that 'It is a complex notion also reffered to as sound color, texture, or tone quality, and is derived from the shape of a segment's specgtro-temporal surface, independently of pitch and loudness.'

**Feature Importance**

## Justification

| Weight | Feature |
|---|---|
| 2.0061 ± 0.1167 | artist_popularity |
| 0.2707 ± 0.0436 | timbre_2_mean |
| 0.1741 ± 0.0334 | timbre_7_std |
| 0.1716 ± 0.0215 | G_dominance_mean |
| 0.1401 ± 0.0246 | timbre_4_mean |
| 0.1345 ± 0.0510 | artist_followers |
| 0.1344 ± 0.0306 | timbre_7_mean |
| 0.1309 ± 0.0724 | key_confidence_mean |
| 0.1238 ± 0.0126 | timbre_8_kurtosis |
| 0.1194 ± 0.0673 | timbre_8_iqr |
| 0.1084 ± 0.0452 | timbre_1_std |
| 0.1011 ± 0.0241 | timbre_5_mean |
| 0.0891 ± 0.0219 | timbre_11_std |
| 0.0772 ± 0.0181 | C_dominance_iqr |
| 0.0761 ± 0.0277 | timbre_9_mean |
| 0.0722 ± 0.0291 | timbre_8_mean |
| 0.0616 ± 0.0197 | timbre_7_kurtosis |
| 0.0606 ± 0.0282 | timbre_10_kurtosis |
| 0.0601 ± 0.0273 | timbre_8_std |
| 0.0575 ± 0.0207 | loudness_kurtosis |

As mentioned earlier, song popularity predictor by Mohamed Nasreldin, Stephen Ma, Eric Dailey, and Phuc Dang was chosen as benchmark model. However, choice of metrics in song popularity predictor was not appropriate and due to lacking source code, it was not possible for us to compare two models. Song popularity predictor used area under curve(AUC) for accuracy metric. However, AUC is a metric which is good for a classification problem. As predicting song popularity which has range of 0 to 100 (continuous), metrics such as root mean squared error, mean absolute error or R squared are more appropriate. Therefore, using source code from song popularity predictor by Mohamed Nasreldin, root mean squared error of their model was planned to be performed. Problem was that source code did not include all the dataset. Especially, dataset named cleaned_million.csv, which is very essential in this source code was missing. None of the code was able to be ran, and comparison between following model and song popularity predictor by Mohamed Nasreldin could not be done.

Instead, I have chosen to achieve root mean squared error below 20. Range of track popularity is from 0 to 100, so 20 percent of the range had been chosen to be threshold of maximum root mean squared error. At the end, final model achieved root mean squared error of 16. Therefore, I was able to conclude that predictor

model is able to give reference for users on expected popularity. When tried with existing songs, predicted popularity of a track was off from actual value of track popularity around 5 to 15 percent. This also confirms that model is performing well.

# V. Conclusion

## Free-Form Visualization

In the project, final predictor file is named as Predict_Song_Popularity(Final).ipynb.

Users are able to use following file to get final predicted popularity. Below is the output from Predict_Song_Popularity(Final).ipynb for the song ‘No Sleep’ by Bossfight. Final output provides predicted song popularity with root mean squared error and mean absolute error of the model.

```
--------------------------------------------------------------------------------

Following Song is No Sleep
by Bossfight

Predicted Track Popularity is : 37.084034/100

* Following model has RMSE(Root Mean Square Error) :  16.3609444620479
  and MAE(Mean Absolute Error) :  11.911048587773239
--------------------------------------------------------------------------------
```

Following model does not provide absolute expected song popularity, but and reference and predicted popularity. By providing root mean squared error and absolute error of the model to the user, user can be aware that predicted popularity is not an absolute value, but a reference.

## Reflection

In order to predict track popularity, data was first collected from Spotify through Spotipy. From artist popularity, artist followers to audio analysis data such as key,

mode, pitch dominance, timbre are collected. Audio analysis data points are given for each segment and each segments has different length. Therefore, audio analysis data needed preprocessing. Each audio analysis data points were multiplied to the weight, which is duration of segment divided by average duration of segments. Afterwards, mean, standard deviation and kurtosis of all the audio analysis data were calculated. In addition entropy of pitch dominance was calculated.

To check if artist popularity, artist followers and statistics of audio analysis data are well distributed, distribution histogram is plotted. As a result, it was confirmed that some of the features were highly skewed. For highly skewed features, log transformation is done. After all the skewed features are transformed, all the data are normalized using MinMaxScaler.

Training and testing dataset is spliced for both preprocessed and non-preprocessed (without log transformation and normalization) dataset in order to compare its effect on accuracy of the model. Both preprocessed and non-preprocessed dataset are trained to supervised learning models, linear regressor, logistic regressor, decision tree regressor, random forest regressor, and support vector machine. Although preprocessed dataset resulted better with supervised learning models than with non-preprocessed dataset, all the supervised learning models did poor job. Root mean squared error was around 200 to 640, which was too high.
Next, neural network was examined for predictor model. At first, root mean squared error of 50 was achieved. Therefore, neural network is chosen as final predicting model and further refinement is done. As a result, root mean squared error was reduced to 18.73. Also, like supervised learning models, using preprocessed data reduced root mean squared error to16.37. As we were able to achieve root mean squared error below 20, predictor model has been improved enough to give general reference to users on track popularity.

Following project required a lot of preprocessing data. From balancing data points as they where separated with different length of segments, to log transforming skewed data and normalizing, preprocessing was one of the most difficult part of the project. However, preprocessing the dataset definitely have improved our final model.

## Improvement

Further improvement could be done with more data points. For following project, only around 4400 data points were used. As 80 percent of data points were used as training and 20 percent of data points were used for testing, around 3500 data points were used for training. This is actually not a lot of data points. If we were able to use more than 10000 data points for training, we could have achieved much lower mean squared error.

Another possible improvement would be using time when song was released as a feature. Song popularity changes as date of song release passes by. Usually after a long time, popularity decreases. Finding when does the most songs reach its peak popularity after the release, and rate of decreasing popularity would have help finding what would be a peak popularity and average popularity during a given time period would provide much deeper insight to artists.