Music Genre Classification

Wesam Alsohle

CONTENTS

01 Frame the problem and look at the big picture.

02 Explore the data to gain insights.

03 Prepare the data.

CONTENTS

04 Train different models

05 Fine tune the model

06 Evaluation and solution.

Frame the problem and look at the big picture

The main aim of this problem is to predict Genre of the track. This problem comes under **supervised Machine Learning Classification.**

Explore the data to gain insights

Data Description

This classic dataset contains the Genre of the track and other attributes of almost **14,000** tracks. It's a great dataset to work with data analysis and visualization.

Features

artist: Name of the Artist.
song: Name of the Track.
popularity: popular the song.
danceability: how suitable a track is for dancing
energy: measure of intensity and activity.
key: The key of the track .
loudness: loudness of a track in (dB).
mode: Mode (major or minor) of a track.
speechiness: presence of spoken words in a track.
acousticness: A confidence measure of the track.
instrumentalness: Predicts a track contains no vocals.
liveness: Higher liveness values represent an increased probability that the track was performed live.
valence: describing the musical positiveness
tempo: the speed or pace of a given piece and derives directly from the average beat duration.
duration in milliseconds :Time of the song
time_signature : how many beats (pulses).

Target

Class Genre of the track

```
df.describe()
```

	Id	Popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_in min/ms	time_signature	Class
count	14396.000000	14063.000000	14396.000000	14396.000000	12787.000000	14396.000000	14396.000000	14396.000000	14396.000000	10855.000000	14396.000000	14396.000000	14396.000000	1.439600e+04	14396.000000	14396.000000
mean	7198.500000	44.525208	0.543105	0.662422	5.953781	-7.900852	0.640247	0.080181	0.246746	0.178129	0.195782	0.486379	122.695372	2.000942e+05	3.924354	6.695679
std	4155.911573	17.418940	0.165517	0.235967	3.200013	4.057362	0.479944	0.085157	0.310922	0.304266	0.159258	0.239476	29.538490	1.116891e+05	0.359520	3.206170
min	1.000000	1.000000	0.059600	0.001210	1.000000	-39.952000	0.000000	0.022500	0.000000	0.000001	0.011900	0.021500	30.557000	5.016500e-01	1.000000	0.000000
25%	3599.750000	33.000000	0.432000	0.508000	3.000000	-9.538000	0.000000	0.034800	0.004280	0.000088	0.097275	0.299000	99.799000	1.654458e+05	4.000000	5.000000
50%	7198.500000	44.000000	0.545000	0.699000	6.000000	-7.013500	1.000000	0.047100	0.081450	0.003920	0.129000	0.480500	120.060000	2.089410e+05	4.000000	8.000000
75%	10797.250000	56.000000	0.658000	0.861000	9.000000	-5.162000	1.000000	0.083100	0.432250	0.201000	0.256000	0.672000	141.988250	2.522470e+05	4.000000	10.000000
max	14396.000000	100.000000	0.989000	1.000000	11.000000	1.342000	1.000000	0.955000	0.996000	0.996000	0.992000	0.986000	217.416000	1.477187e+06	5.000000	10.000000

```
df.duplicated().any()
```

```
False
```

```
df.info()
```

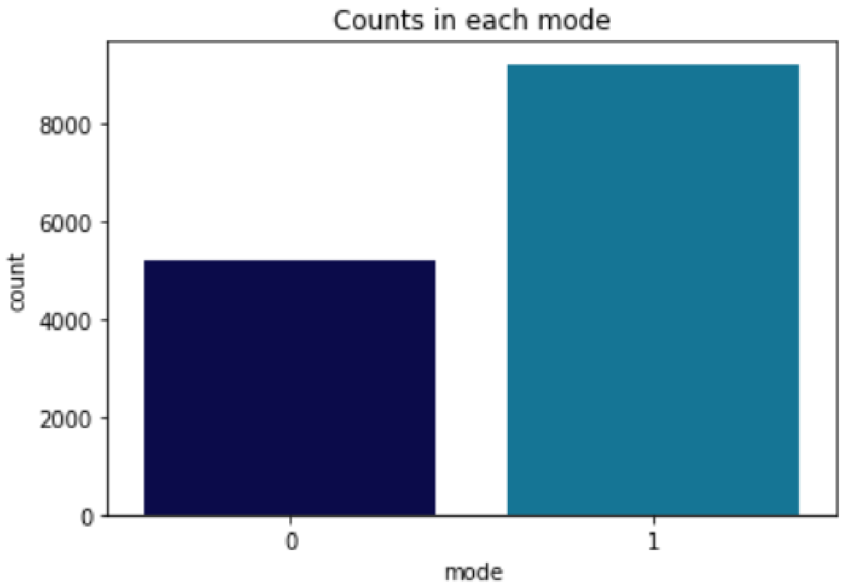
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14396 entries, 0 to 14395
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Id                  14396 non-null  int64
 1   Artist Name         14396 non-null  object
 2   Track Name          14396 non-null  object
 3   Popularity          14063 non-null  float64
 4   danceability        14396 non-null  float64
 5   energy              14396 non-null  float64
 6   key                 12787 non-null  float64
 7   loudness            14396 non-null  float64
 8   mode                14396 non-null  int64
 9   speechiness         14396 non-null  float64
 10  acousticness        14396 non-null  float64
 11  instrumentalness    10855 non-null  float64
 12  liveness            14396 non-null  float64
 13  valence             14396 non-null  float64
 14  tempo               14396 non-null  float64
 15  duration_in min/ms  14396 non-null  float64
 16  time_signature      14396 non-null  int64
 17  Class               14396 non-null  int64
dtypes: float64(12), int64(4), object(2)
memory usage: 2.0+ MB
```

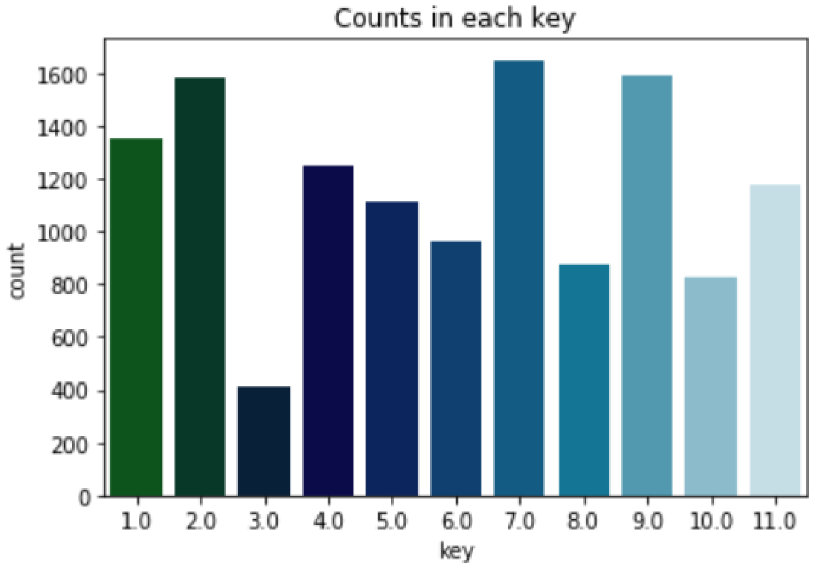
	data_type	missing_val	missing_val_ratio
Id	int64	0	0
Artist Name	object	0	0
Track Name	object	0	0
Popularity	float64	333	2
danceability	float64	0	0
energy	float64	0	0
key	float64	1609	11
loudness	float64	0	0
mode	int64	0	0
speechiness	float64	0	0
acousticness	float64	0	0
instrumentalness	float64	3541	25
liveness	float64	0	0
valence	float64	0	0
tempo	float64	0	0
duration_in min/ms	float64	0	0
time_signature	int64	0	0
Class	int64	0	0

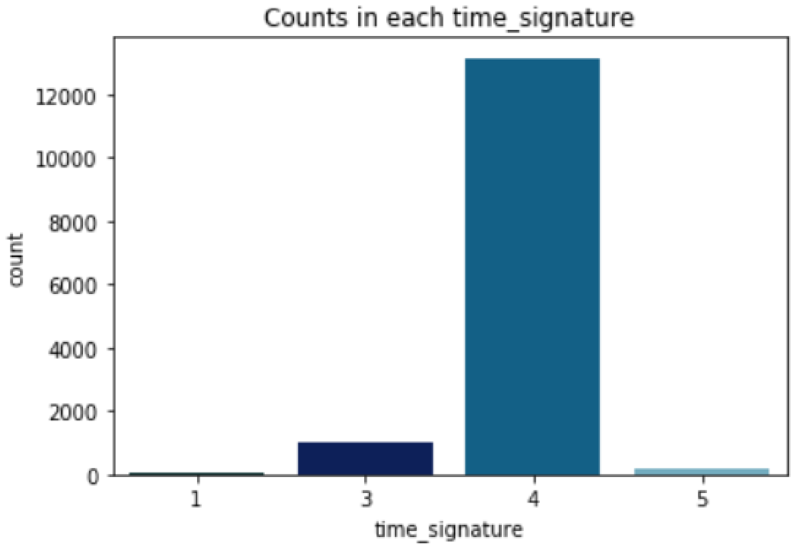
```
df.nunique()
```

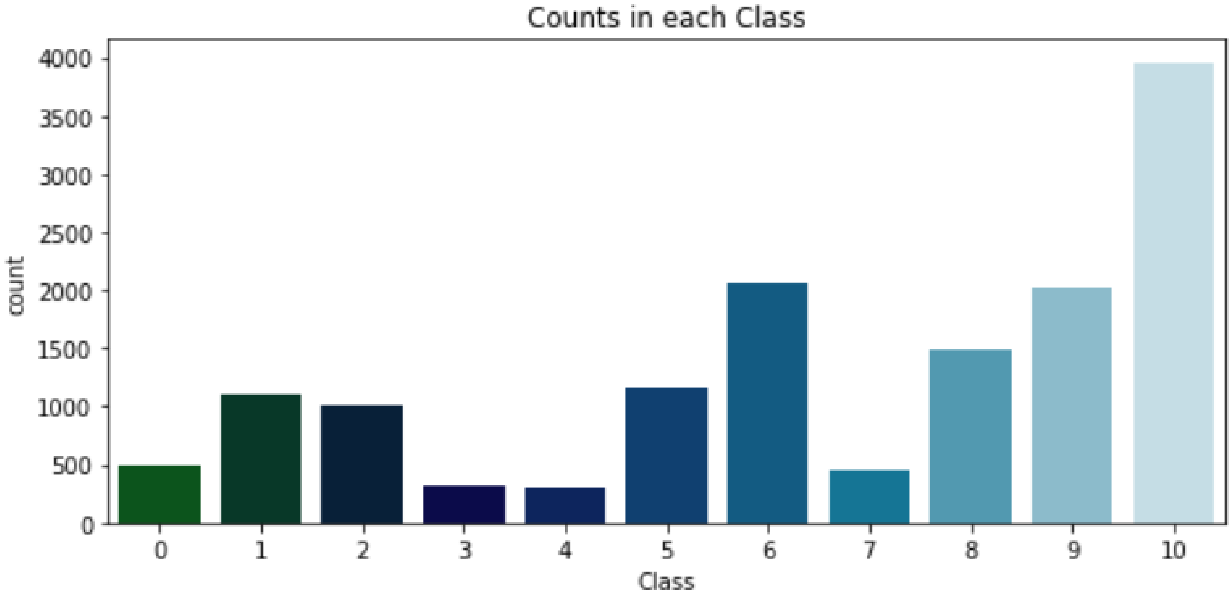
```
Id                    14396
Artist Name            7913
Track Name            12455
Popularity              100
danceability            887
energy                 1156
key                      11
loudness               8051
mode                      2
speechiness            1177
acousticness           3725
instrumentalness       3945
liveness               1407
valence                1268
tempo                 11392
duration_in min/ms    11805
time_signature            4
Class                    11
dtype: int64
```

Exploratory Data Analysis

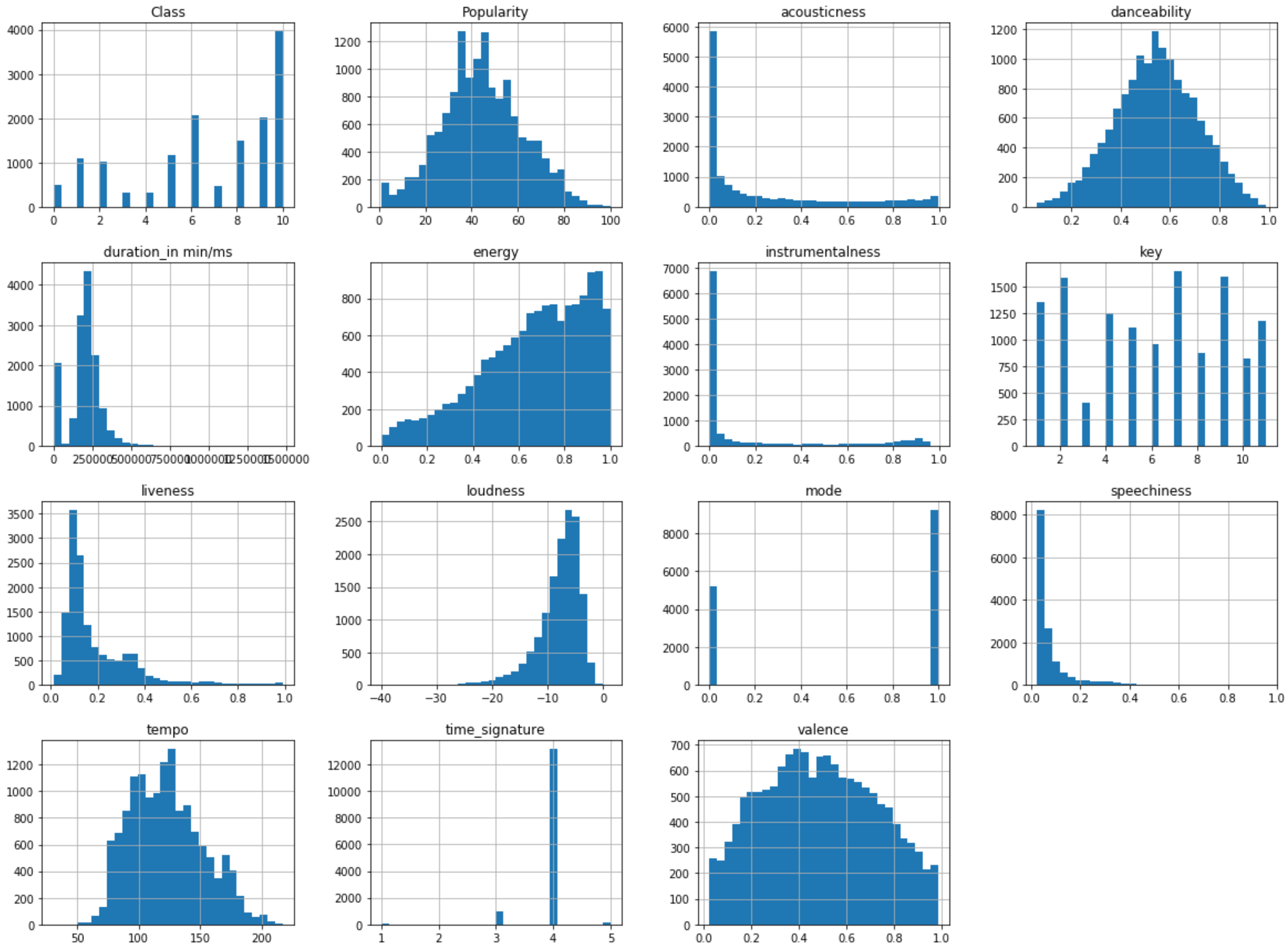




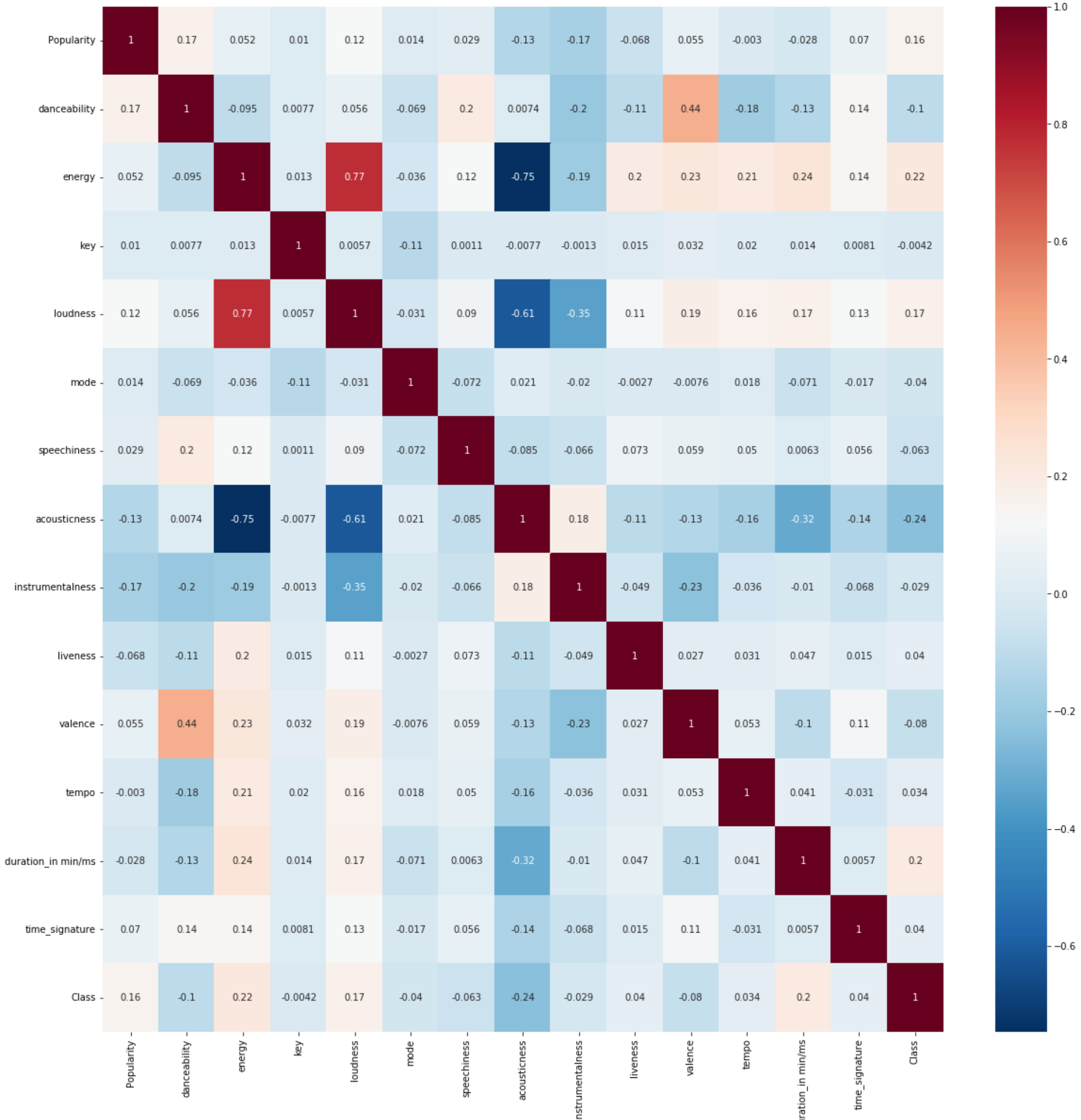




Visualize distribution of all the Continuous Predictor variables in the data using histograms



Visualize The Correlations between numerical features specally with target (class)




```
corr_matrix['Class'].abs().sort_values(ascending=False)
```

```
Class               1.000000
acousticness        0.240609
energy              0.215611
duration_in min/ms  0.203822
loudness            0.174111
Popularity          0.159484
danceability        0.101152
valence             0.080036
speechiness         0.062784
liveness            0.040101
mode                0.040092
time_signature      0.040053
tempo               0.034496
instrumentalness    0.028631
key                 0.004175
Name: Class, dtype: float64
```

Prepare the data

1

Split Data

- Into test&train
- test size = **0.25**

2

Encoder

OrdinalEncoder:

Artist Name

LabelEncoder:

Track Name

3

Missing values

Clean data with

fill all missing

values

4

outliers

Outliers all feature

by interquartile

range

5

Scaling

Apply

StandardScaler

on features

6

Feature engineering

drop id, kay that

have very **low**

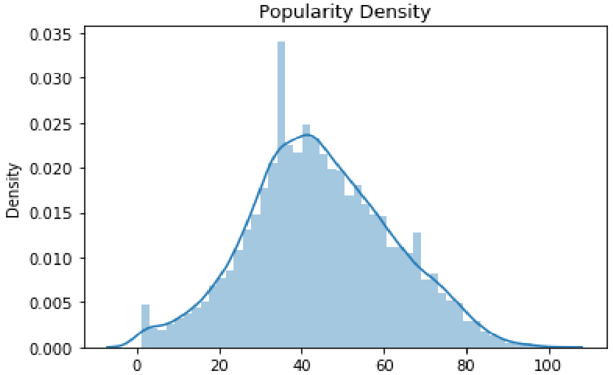
effect to predict

Genre of the track

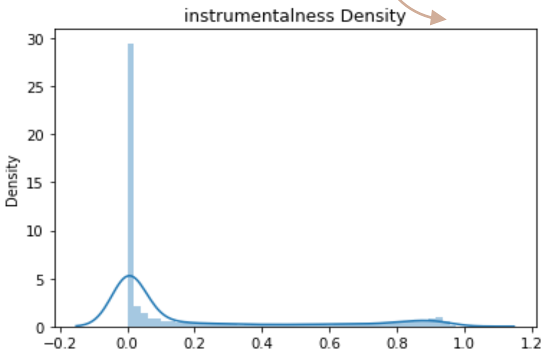
Missing values

Before

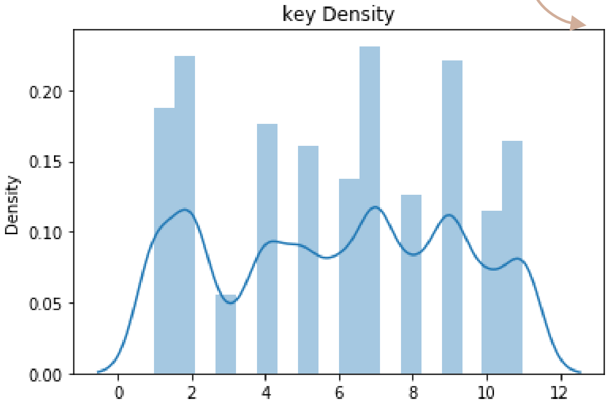
fill with min value



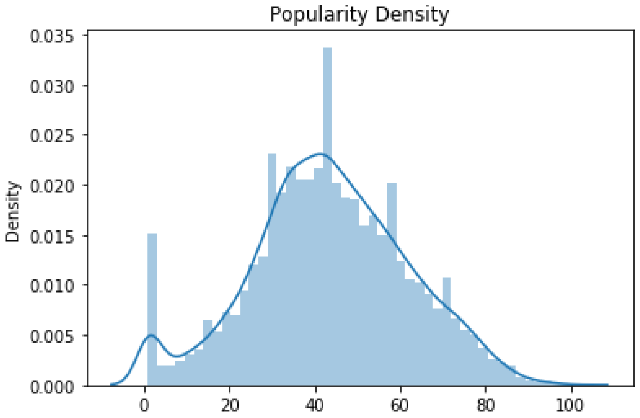
fill with mean value

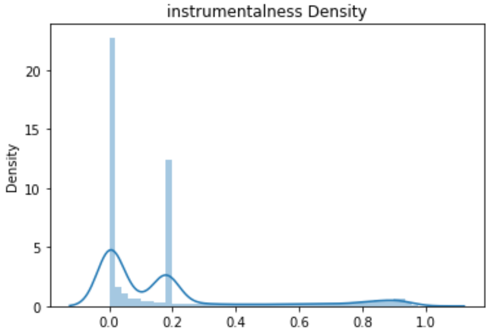


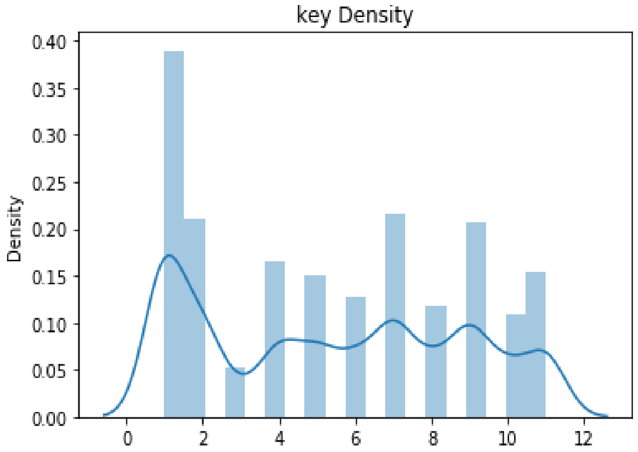
fill with min value



After

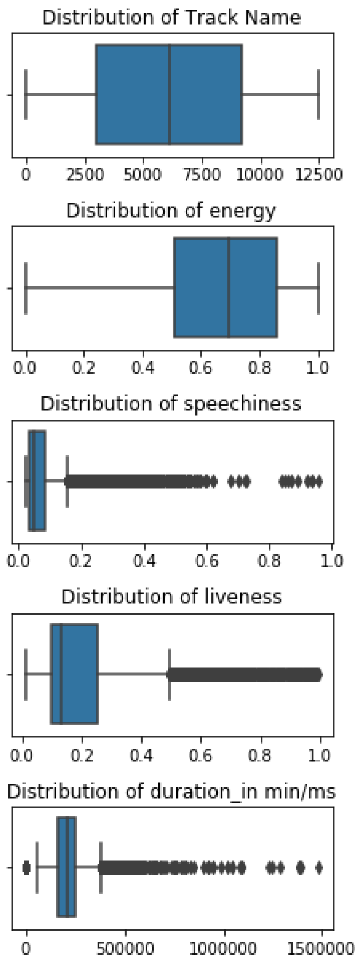




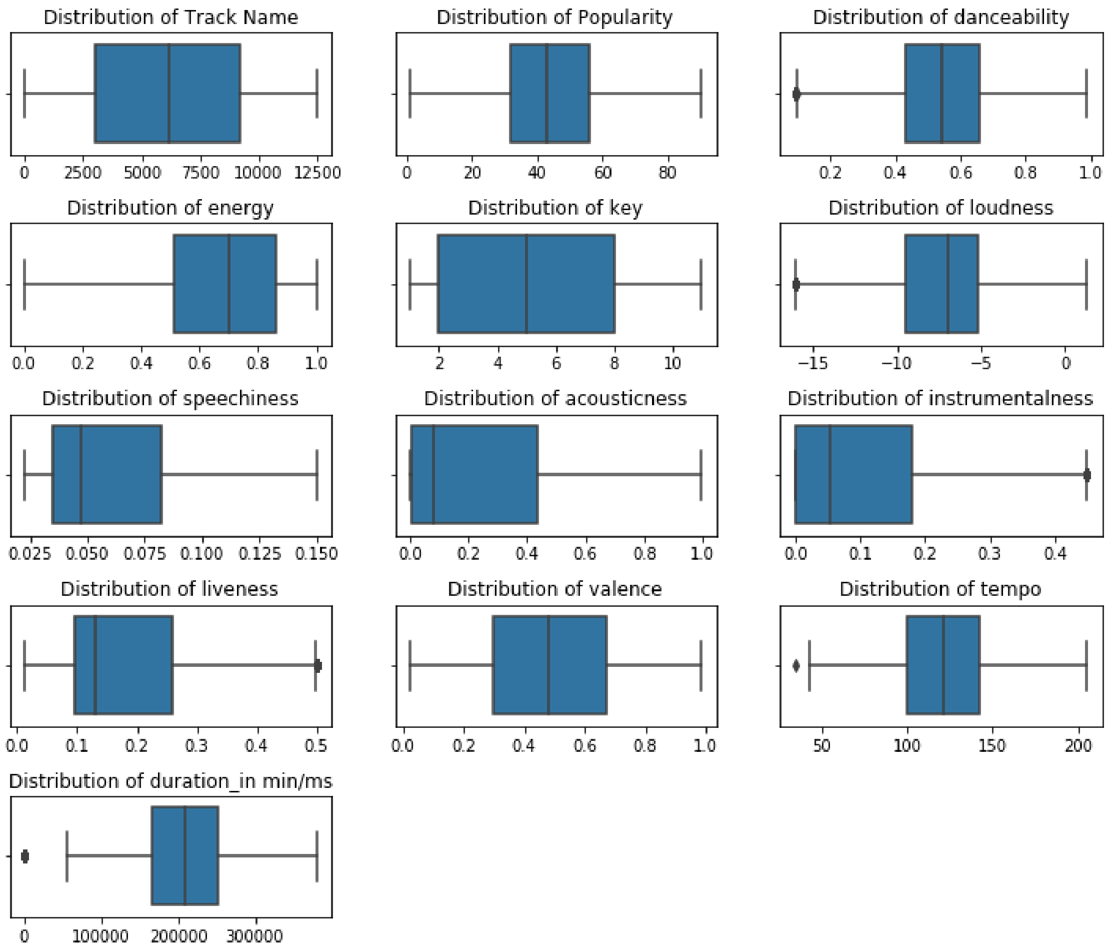


Outliers

Before

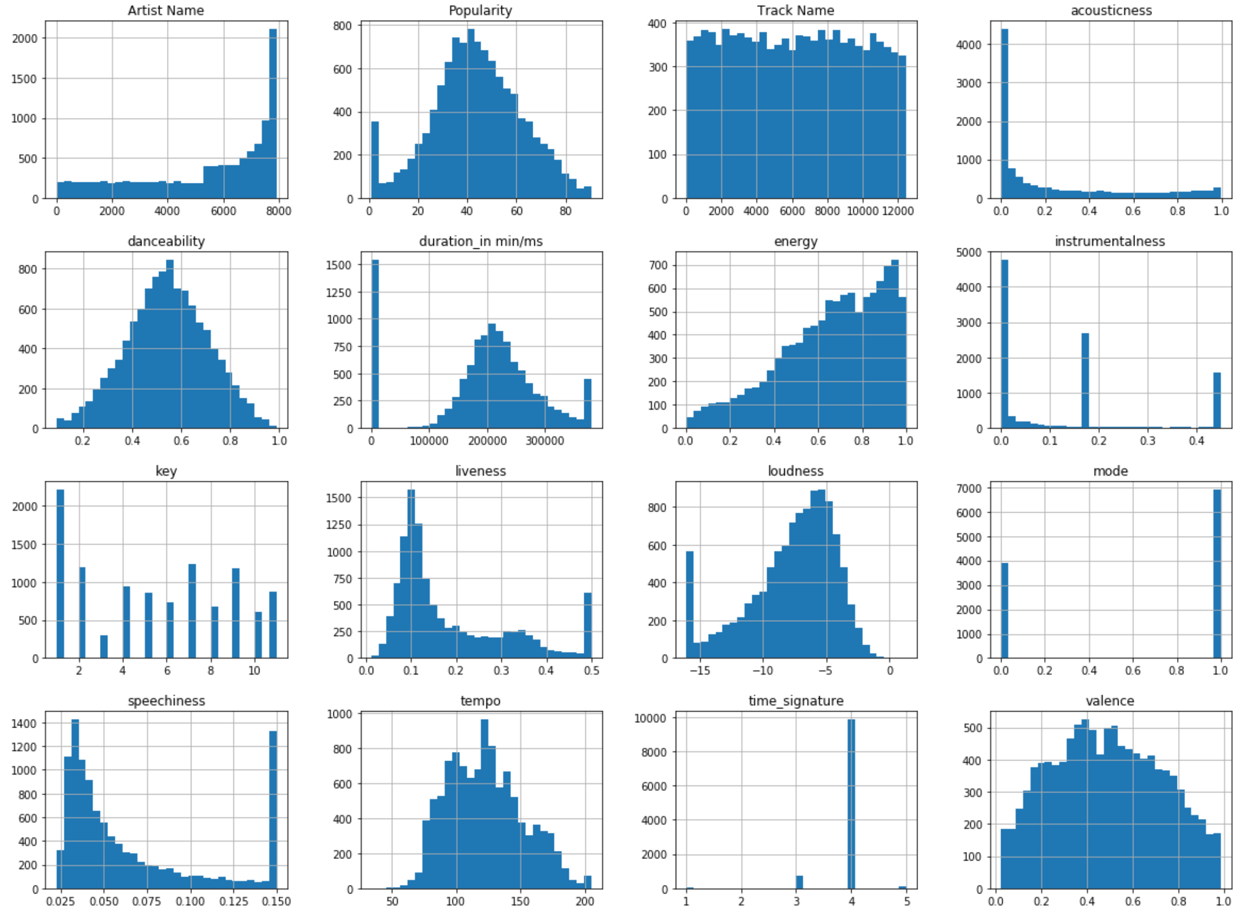


After

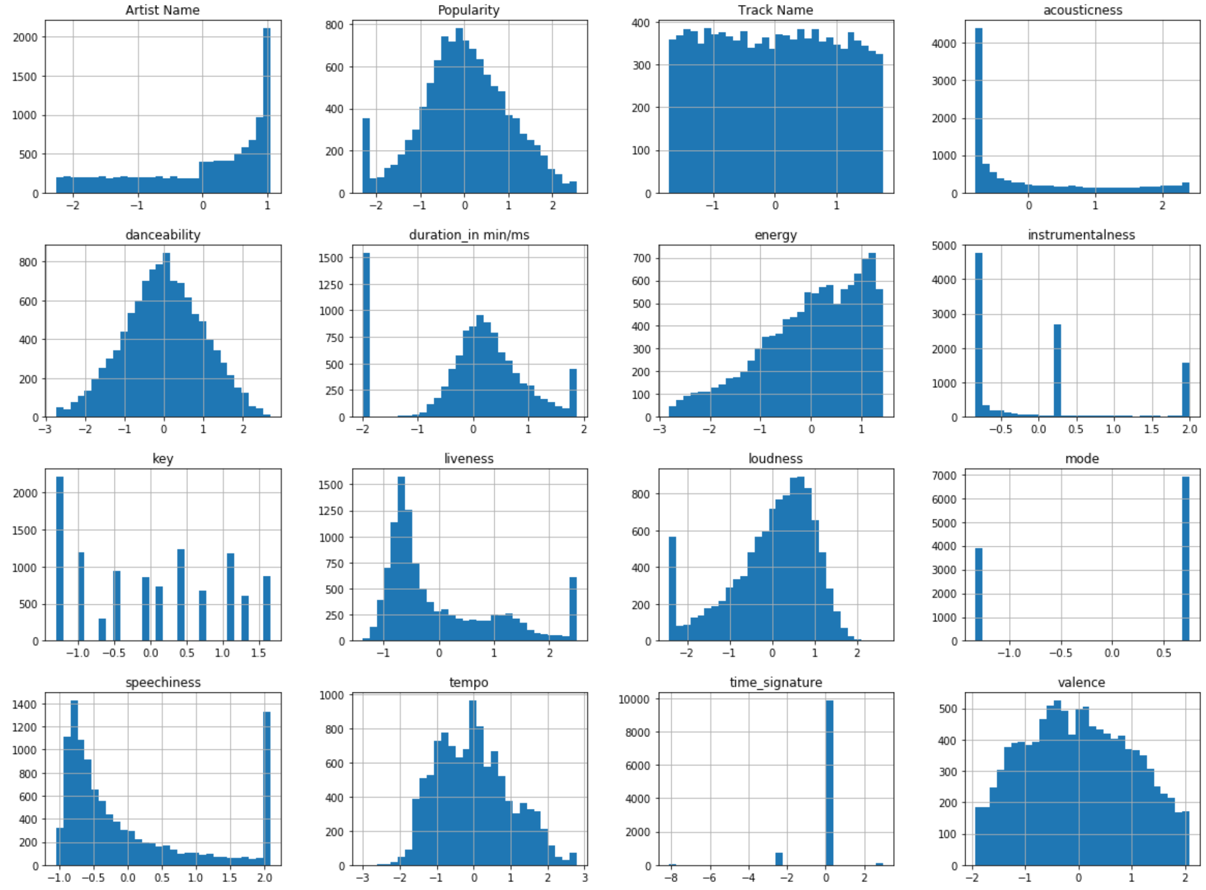


Scaling

Before



After



Train different models

Evaluation Metric
The evaluation metric for this competition is Root Mean Squared Error (**F1-score**).
The F1-score can be interpreted as a harmonic mean of the precision and recall.
F1 = 2 * (precision * recall) / (precision + recall)

SVC

kernel 'linear'
F1: 0.47

kernel 'rbf'
F1: 0.49

kernel 'poly'
F1: 0.47

Logistic Regression
F1: 0.51

KNeighbors Classifier
F1: 0.523

Decision Tree Classifier
F1: 0.47

Gradient Boosting Classifier
F1: 0.49

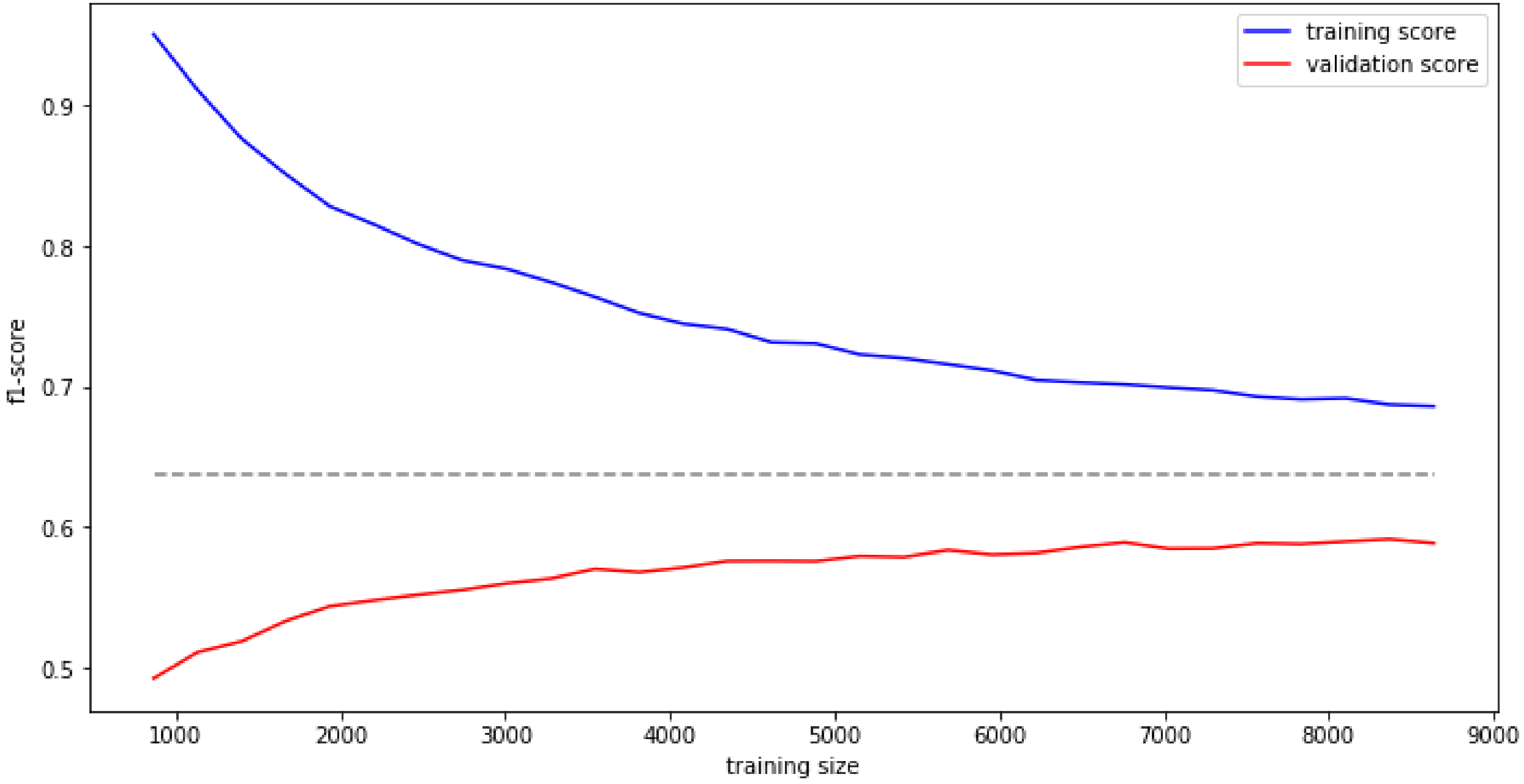
Random Forest Classifier
F1: 0.47

XGB Classifier
F1: 0.55

Fine tune the model

Choose XG**BoostingClassifier model** with best parameter.

XGBClassifier(learning_rate=0.3 ,n_estimators = 100 ,max_depth=2)

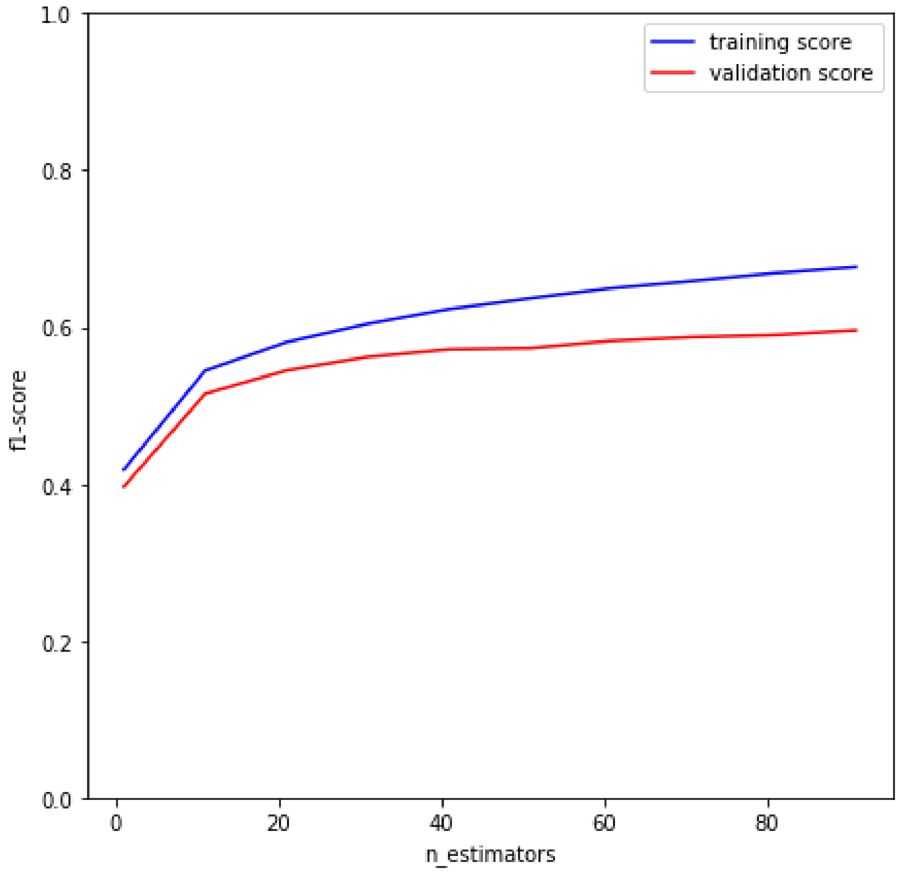


XGBClassifier
F1: 0.55

Fine tune the model

Choose XG**BoostingClassifier model** with best parameter.

XGBClassifier(learning_rate=100=srotamitse_n ,0.3 ,max_depth=2)



XGBClassifier
F1: 0.55

Thanks For Listening^^

Any questions!