Majority Key Prediction of a song

Ravi Kiran Mekala

1.Data Set

The kaggle data set being used is related to the statistics of the top 10 songs of various artist from Spotify and Youtube. It contains 26 variables for each of the songs collected from Spotify. The list of variables is:

- **Track**: name of the song, as visible on the Spotify platform.
- **Artist**: name of the artist.
- **Url_spotify**: the Url of the artist.
- **Album**: the album in wich the song is contained on Spotify.
- **Album_type**: indicates if the song is relesead on Spotify as a single or contained in an album.
- **Uri**: a spotify link used to find the song through the API.
- **Danceability**: describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable.
- **Energy**: is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
- **Key**: the key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
- **Loudness**: the overall loudness of a track in decibels (dB). Values typically range between -60 and 0 db.
- **Speechiness**: detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- **Acousticness**: a confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **Instrumentalness**: predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0
- **Liveness**: detects the presence of an audience in the recording. A value above 0.8 provides strong likelihood that the track is live.
- **Valence**: a measure 0.0 to 1.0 describing the musical positiveness covered by a track.
- **Tempo**: the overall estimated tempo of a track in beats per minute (BPM).
- **Duration_ms**: the duration of the track in milliseconds.
- Stream: number of streams of the song on Spotify.
- **Url_youtube**: url of the video linked to the song on Youtube, if it have any.
- **Title**: title of the videoclip on youtube.
- **Channel**: name of the channel that have published the video.
- **Views**: number of views.
- **Likes**: number of likes.
- **Comments**: number of comments.
- **Description**: description of the video on Youtube.
- **Licensed**: Indicates whether the video represents licensed content
- **official_video**: boolean value that indicates if the video found is the official video of the song.

Our label for the dataset is Key and we will use several Machine Learning models to predict it.

2.Data Cleaning and Preparation

- We have around 20718 total rows of which 2176 missing values in either one of the features. As it's a significant number, we end up filling the missing values for certain columns with the corresponding column's mean.
- After filling the missing values, we only have 2 rows with missing values. These rows can be removed as they are significantly less than the total data set count.
- One of the duplicate rows can be removed to avoid unnecessary redundancy.
- The features which have no significance to determine the **Key** of the song can be removed. These are removed based on domain knowledge. The features removed are *Artist, Url_spotify, Track, Album, Uri, Duration_ms, Url_youtube, Title, Channel, Description, Licensed, Official_video*.
- The non-numerical feature, *Album_type*, can be converted to numerical ones using Label Encoding and any numerical feature columns stored as string can also be converted to numbers.
- As all the features are numerical, normalization will lead to feature values between zero and one.

3.Data Analysis

As part of Data Analysis, we have seen the correlation of features with key column and did the Pearson correlation of features and found out that Valence, *Danceability, Energy, Album_Type, Loudness, Speechiness, Views, Comments and Likes* are the top positive correlated features.

In case of negatively correlated features from the data we found that *Acousticness, Album, Liveness, Stream and Licensed* are the top negatively corelated features.

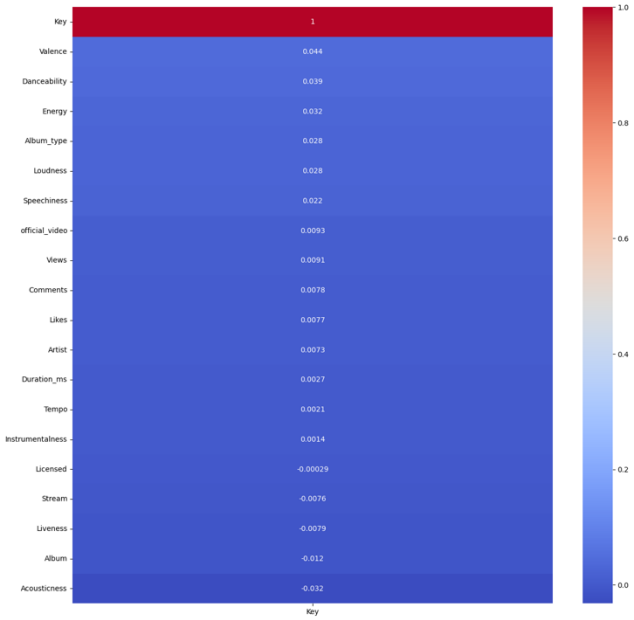


Figure: 1: Correlation with Key label

After analyzing the Key label's distribution, it is found that there is class imbalance for the multi class distribution. The highest distribution is for the Key 0 with 2195 count and the least distribution is for the Key 3 with 626. Hence, we decided to move for calculating the accuracy to weighted accuracy and, we will show confusion matrix to understand which classes are being confused after the evaluating each model.

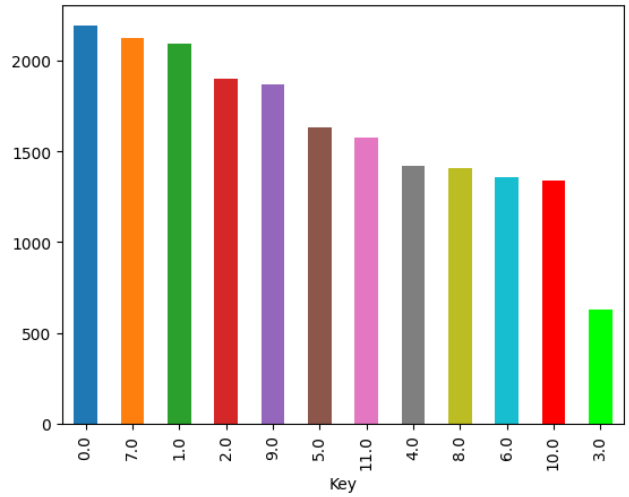


Figure 2: Class Imbalance

The distribution of numerical data columns can be seen below. Among the features, most of the top correlated features (e.g.: *Danceability, Energy, Loudness, Valence*) are almost in normal distribution.

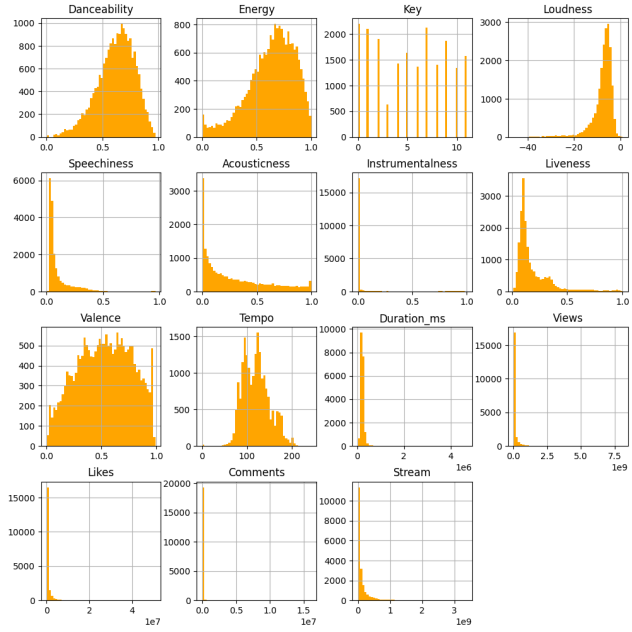


Figure 3: Features Distribution

For the top correlated features, we have shown the boxplots to understand the range of data with each Key label. Rest of the plots are shown in the code.

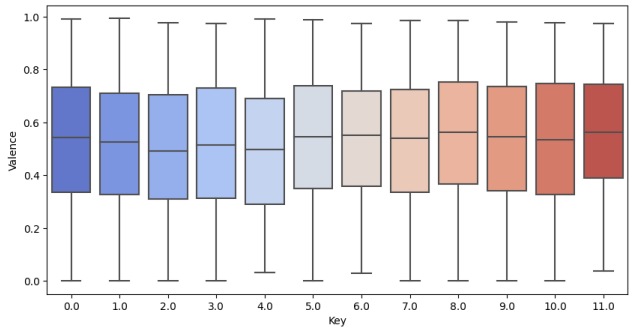


Figure 4: Valence vs Key

There are three different types of Album_types over the data like album, single, compilation. To observe the distribution of Keys over the Album_types we used seaborn countplot.

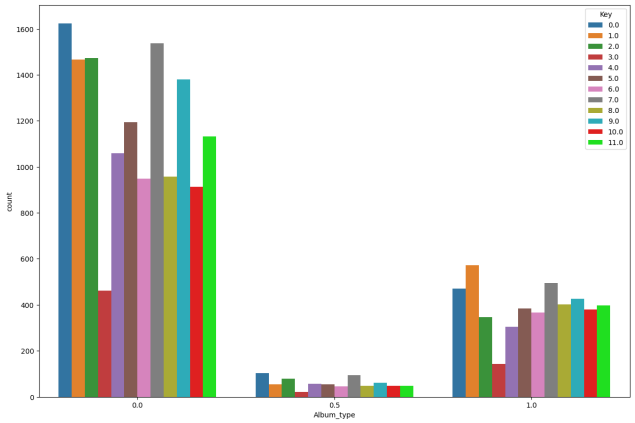
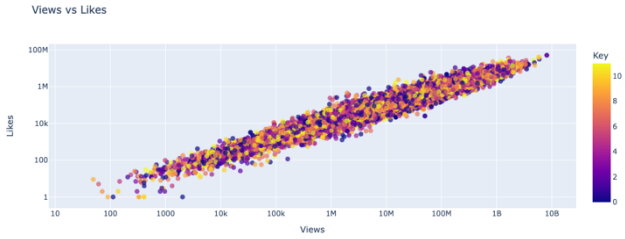
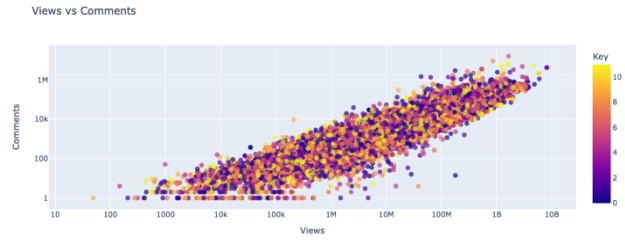
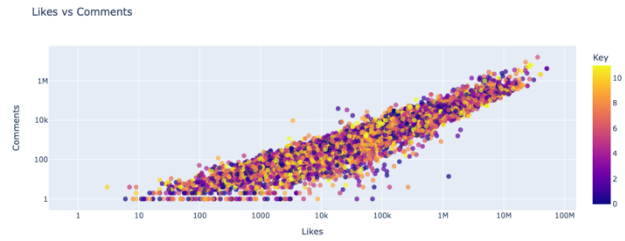


Figure 5: Album_type and Key distribution

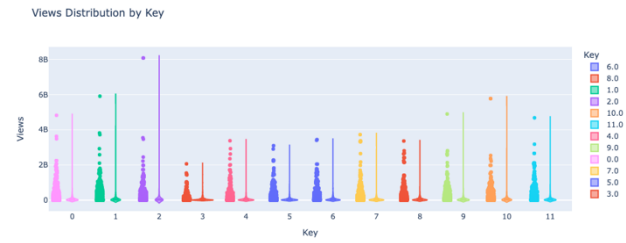
To view the distribution of Key labels over views vs likes, likes vs comments, and comments vs views we used scatter plot to show the distribution. The data is linear in case of views vs likes with correlation of 0.89 but whereas in case of views vs comments and likes vs comments the data is not linear with the correlation of 0.63 each referred from heatmap.

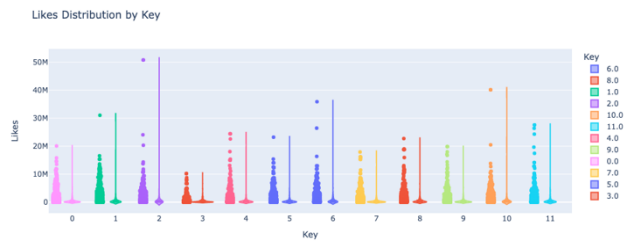


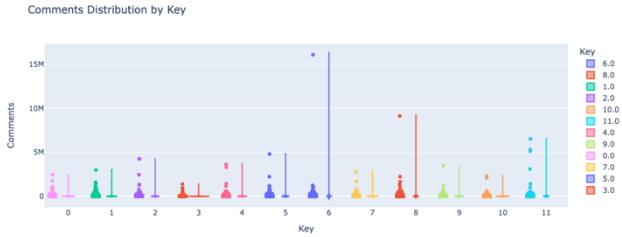




For viewing the density of views, likes, and comments over the keys we used violin plots to see the distribution.





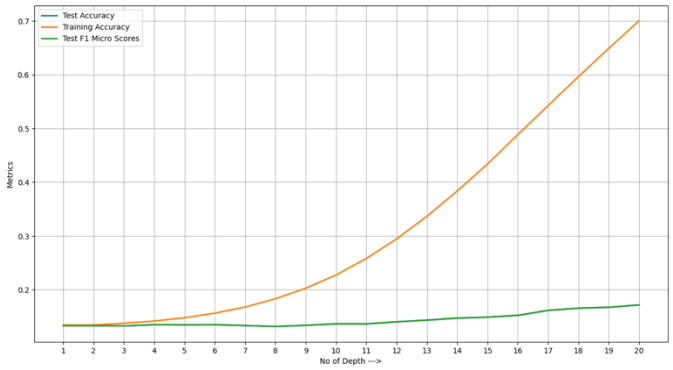


4. Random Sampling and Feature selection:

After observing that the own implementation models are taking a lot of time to get trained on the whole 20000 data points, it is always good to check if the random sampled data is also performing the same. This will help us in training all the models faster with minimal loss in the prediction accuracy.

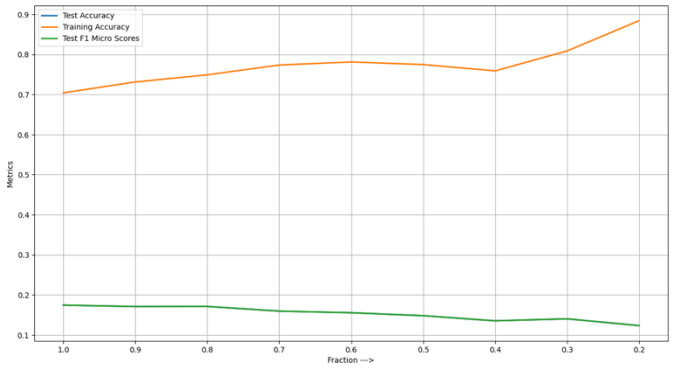
Let's do random resampling to decrease the size of the dataset. We first need a reference model to cross validate the right amount to be removed from the whole data without losing much of f1 score.

Let's consider the whole data set and train a decision tree to get a reference model which can be used for cross validating the fraction value to be used for random sampling. To get the decision tree, we need to find the best depth for the whole data. We get the best depth as 20.



Best depth = 20

With the Decision tree model of depth 20, the best fraction value for random sampling is 50% as the drop in test accuracy metric is highest from 50% to 40%.



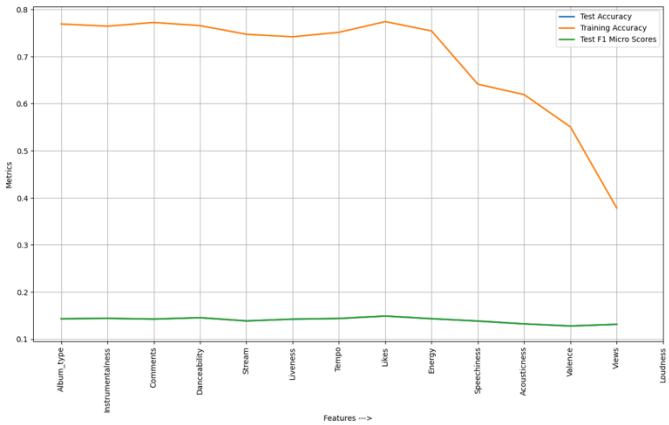
Highest drop is for fraction = 0.5

Let's find the Mutual Information Gain between all the features and the output label 'Key'. We can observe that all the features are very weakly correlated with the target variable. We will try to remove the features that are not correlated with the target variable and see if the accuracy remains same or improves.

Feature	correlation with Key
Album_type	0.000676197
Instrumentalness	0.026814914
Comments	0.030266993
Danceability	0.033935208
Stream	0.037060594
Liveness	0.041242251
Tempo	0.04223979
Likes	0.046088835
Energy	0.0470638
Speechiness	0.04929706
Acousticness	0.050087903
Valence	0.051760044
Views	0.052552044
Loudness	0.054408114

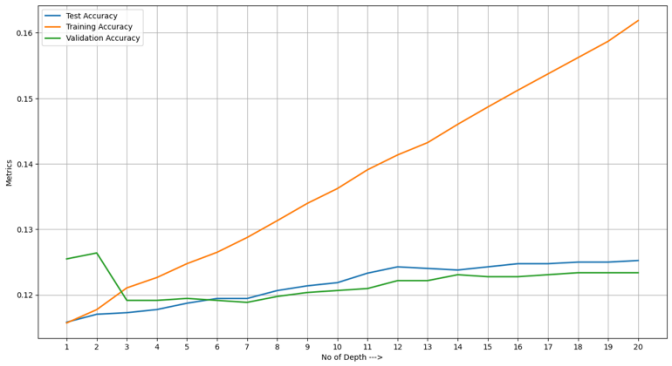
List of features and their correlation with Key label in increasing order of correlation from top to bottom.

It is observed that by removing features Liveness, Stream, Danceablility, Comments, Instrumentalness, Album_type, there is an increase in the accuracy as compared to all the other combinations. So, the only features we can consider are Tempo, Likes, Energy, Speechiness, Acousticness, Valence, Views and Loudness.
We have reduced the number of features from 14 to 8 with an improvement in the accuracy.

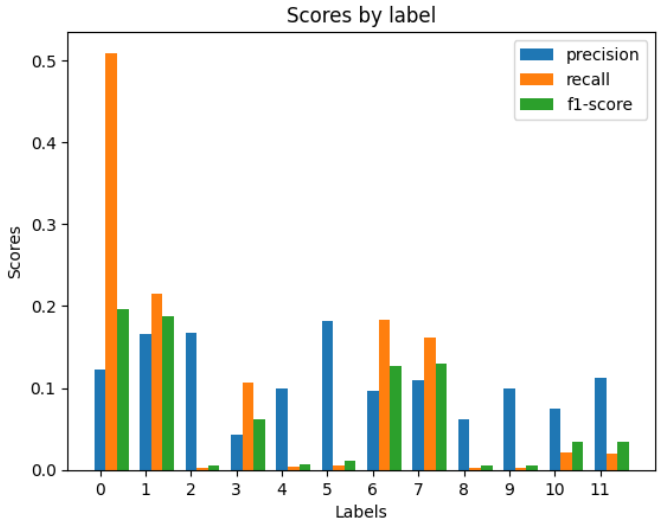


Removing all the features whose correlation is less than or equal to Likes feature is fetching us highest test accuracy with no loss in training accuracy.

5. Own Implementations:

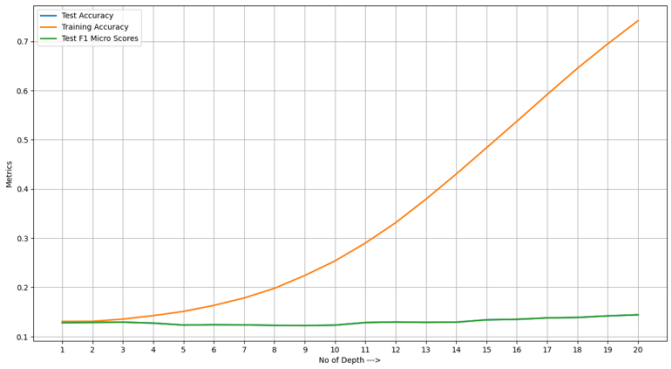


Decision Tree own implementation with best depth: 20



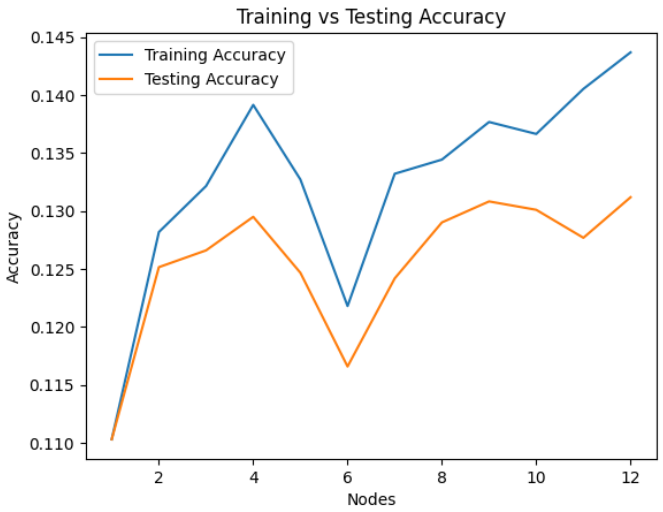
Score for each label for Gaussian Naïve Bayes Own implementation

6. Scikit Implementations:

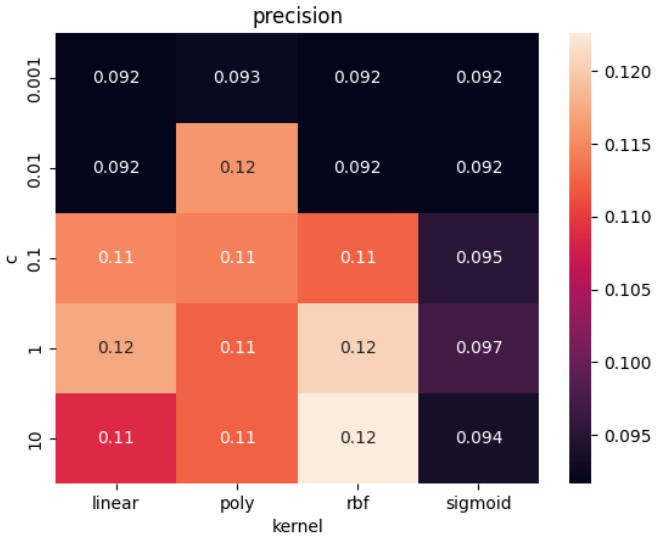


Decision Tree scikit learn implementation best depth: 20

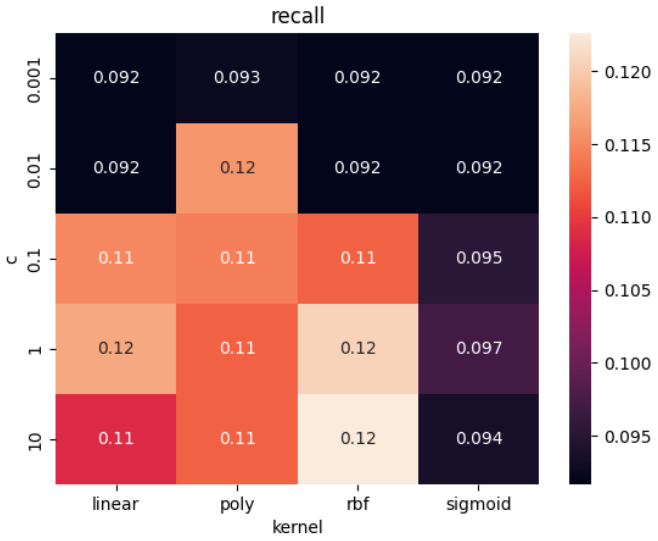
KNN scikit learn implementation best K: 2



Neural Network scikit learn implementation best number of nodes in the first hidden layer: 12

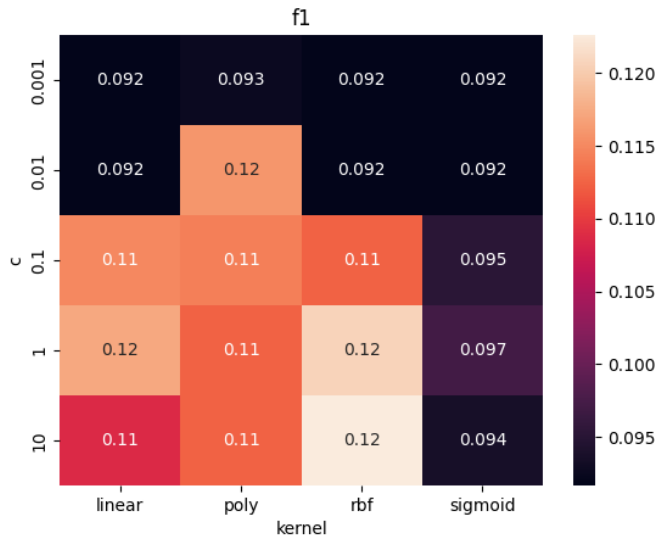


SVM Precision scikit learn implementation heat map



SVM Recall scikit learn implementation heat map

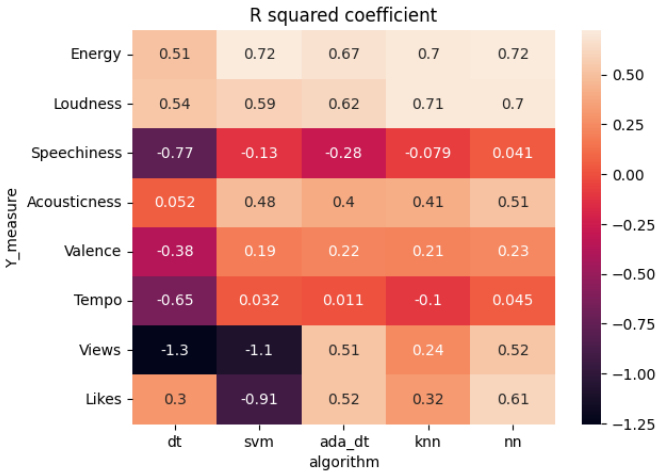
SVM scikit learn implementation.
Best parameters: c=1, kernel=rbf



SVM F1 scikit learn implementation heat map

7. Experiments

Since we have less correlation of all features with ***keys*** label, we considered the case where other features can be better labels than ***keys.*** The performance metric used here is R squared coefficient calculation which represents if the value is near to one then it is more correlated and predictable.



8.Conclusion

Algorithm	Precision	Recall	F1 score	Training Accuracy	Testing Accuracy
Logistic Regression	0.1158301	0.1158301	0.1158301	0.1395124	0.1158301
SVM	0.120656	0.1206563	0.1206563	0.1408399	0.1206563
Decision Tree	0.1322393	0.1322393	0.1322393	**0.6873038**	0.1365830
KNN	**0.1433397**	**0.1433397**	**0.1433397**	0.5870142	**0.1433397**
Neural Network	0.1216216	0.1216216	0.1216216	0.145546	0.1216216
Gaussian naïve bayes **(own)**	0.11	0.11	0.11	0.121168	0.1148648
Decision Tree **(own)**	0.1318	0.1318	0.1318	0.1457	0.1318

1. Even after feature selection using mutual information gain all the features are very weakly correlated with the target variable, none of the simple/complex models can fit the training data for a decent test f1 score. Hence all the f1 scores are very low. This implies that the data quality is not good for predicting ***keys*** label.
2. The best performing model is KNN with k=2 but it can also be observed that the model overfit the training data with high training accuracy as compared to the testing accuracy.
3. Decision tree (sklearn implementation) with depth 20 has the highest training accuracy of all. This implies that decision trees have overfit the data to a very high extent. But it still hasn't overfit the training data totally. This is because of number of continuous features in the training data which decision tree finds hard to overfit.
4. Logistic Regression is performing the poorest of all as it's a linear classifier and its performing same as Gaussian Naïve Bayes.
5. The performance of models in fact improved to an extent after removing some of the weakly correlated features. This also aided in increase of training/testing speed.
6. The precision and recall scores for all the models are same. This means that the model has an equal ability to correctly identify positive instances (true positives) and correctly exclude negative instances (true negatives).
7. All the models except Decision Trees (scikit learn implementation) and KNN haven't overfit the data. This is evident because KNN with low k(s) and Decision tree with high depth have high variance and tend to overfit. The other models have low accuracy and low variance. This suggests that weak correlations in the data are inherently challenging to capture accurately. In such cases, it may be important to reconsider the features being used.
8. We have implemented other complex models using sk-learn algorithms (NN with 3 hidden layers, SVM with rbf kernel and c as 100) and were able to note that the performance is still the same despite increase in the complexity.
9. Based on above results we can infer that predicting **views** with (decision tree, svm), **likes** with svm, **Tempo** with (decision tree, knn), **Speechiness** with (decision tree, svm) **adaboost** with (Decision tree algorithm, knn), **valence** with decision tree cannot perform better because their values are negative.
10. Knn, nn with **Speechiness** and svm, adaboost decision tress with **Tempo** as the label has r squared coefficient close to zero. The models do not capture any meaningful relationship between the features and the target variable for this case.
11. May be predicting **Energy** with (neural networks algorithm or svm) or **Loudness** with (knn or neural networks) might have given good accuracy measures because their R squared coefficient is close to one.