Music Classifier – Singer Identifier

G1-MusicDeciphers

**Data Science Capstone Project   
Predictive Modeling Report**

Date:

02/12/2021

**Team Members:**

Likhil Kumar Rachuri - [lkr46@drexel.edu](mailto:lkr46@drexel.edu)

Nandakishor Reddy Pulagam - [np689@drexel.edu](mailto:np689@drexel.edu)

Nirupam Kumar Velagapudi - [nkv27@drexel.edu](mailto:nkv27@drexel.edu)

Soujanya Navuluru - [sn863@drexel.edu](mailto:sn863@drexel.edu)

Vuthej Krishna Reddy Verama Reddy - [vv334@drexel.edu](mailto:vv334@drexel.edu)

[The purpose of this report is to describe the predictive modeling on the data that you have acquired, pre-processed, and explored in DSCI 591.]

1. Define the Predictive Modeling Problem

1. Input: What are the input data and define the input data clearly?
2. Data Representation: What is the data representation?
3. Output: What are you trying to predict? Define the output clearly.

2. Predictive Models

1. What are the methods? Give a general introduction of the methods with references
2. Describe the methods with a pseudo code using the definitions in Section 1.
3. Justify the choice of the method.

3. Evaluations

1. What metrics do you use for evaluation?
2. What is your ground truth?
3. Discuss the performance and the limitation of the method.

**Appendix**

[Addition materials that are not included in the above sections.]

In the first predictive modeling report due in Week 5, you only need to use one method in the predictive modeling. Through the experience of implementing and testing the predictive modeling method, you may learn how it works and fails. In the final predictive modeling report due in Week 10, you will need to try more methods and compare with the one you use in the first report.

Table of Contributions

The table below identifies contributors to various sections of this document.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Section** | **Writing** | **Editing** |
| **1** | **Predictive Modeling Problem Definition** | **Kishor, Nirupam & Soujanya** | **Likhil & Vuthej** |
| **2** | **Predictive Models** | **Likhil & Vuthej** | **Kishor, Nirupam & Soujanya** |
| **3** | **Evaluations** | **Kishor, Nirupam & Soujanya** | **Likhil & Vuthej** |
| **4** | **Appendix** | **Likhil & Vuthej** | **Kishor, Nirupam & Soujanya** |

**Grading**

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.

**1. Define the Predictive Modeling Problem**

1. **Input: What are the input data and define the input data clearly?**
2. **Data Representation: What is the data representation?**
3. **Output: What are you trying to predict? Define the output clearly.**

**1.A - Input: What are the input data and define the input data clearly?**

‘Jio Saavn’ and ‘Deezer’, which are the online music streaming services, act as the data source for our term project. Songs from various artists and different genres have been extracted across three different languages (English, Telugu and Hindi). MP3/MP4 version of the audio data is converted into WAV format before extracting the vocals and musicals separately. Audio features are then extracted using the feature extraction techniques present in Librosa which is a python package for music and audio analysis.

Our final dataset is comprised of 22775 instances with 126 features accompanied by a target variable.

While looking at artist and genre data separately, we have

* 13116 instances for artists data
* 9659 instances for genre data

Amongst 126 features, only two features are categorical i.e. ‘song\_id’ and ‘language’. Along with these two features, the target variable ‘target’ is also categorical. Remaining 124 features are all numerical.

**Song\_id** – This is a song identifier which is unique to each entry.

**Tempo** – Tempo speaks about the frequency of the beats present inside the audio file per second. It is the rate of musical beat present in the audio and is calculated using the reciprocal of the beat period. It is defined in the units of beats per minute (BPM).

**Zero\_crossings** – It is defined as the number of times the signal in audio is crossing the zero line, which indirectly speaks about the shift of values from positive to negative or vice versa.

**Spectral\_bandwidth** (mean and variance) – It is the range of values in which the signal is not less than the half of its maximum value. This feature is also important in analyzing the signal. The values are considered into our dataset after normalization.

**Spectral\_contrast** (mean and variance) – Spectral contrast is defined as the difference of peaks and valleys in the spectrum. It considers the spectral peak, valley and their difference in each sub-band.

**Spectral\_centroids** (mean and variance) – It is a metric in digital signal processing, where spectral centroids are defined as the points where the center of mass for the spectrum is located. This is an important feature for our data as we may depend on the center of mass for each song which differentiates them accordingly.

**Spectral\_rolloff** (mean and variance) – Spectral rolloff is the range where in 85% of the spectrum energy falls into. This gives us the value where all the energy is located which helps us in differentiating the songs and thus helps in our modelling.

**Spectral\_flatness** (mean and variance) – In order to capture the presence of noise within the data, ‘Spectral Flatness’ measure can be used. It measures the amount of noise that is present in the input audio rather than the presence of toned data. This value takes the range of 0 to 1. Higher ‘Spectral Flatness’ value (1) indicates the data being similar to White-Noise whereas a lower value (zero) indicates the absence of White-Noise.

**Chroma\_stft 1 to 12** (mean and variance) – Chroma feature describes the presence of tonal content in the audio data by classifying input into twelve different pitch classes. Short-Time Fourier Transformation helps to get the information about the frequency distribution of the input data by providing a list of 12 features as output. Off the 12 features that Chroma STFT provides, mean and variance values have been considered and used in our data.

**Chroma\_cqt 1 to 12** (mean and variance) - The frequency obtained by the Constant-Q Transform directly corresponds to the central frequencies of the musical notes. At both very high frequencies and very low frequencies, it is often difficult for humans to capture and process the data. Hence by allocating more processing and more time, better resolution is obtained at these frequencies. But owing to this high processing time and complex time-frequency matrix, DFT is preferred over the CQT.

**Chroma\_cens 1 to 12** (mean and variance) – This is another variant of the chromagram feature and is robust to dynamics, timbre and articulation. This effectively helps in identifying and matching the audio data and also in performing the audio classification accurately.

**Mfcc 1 to 20** (mean and variance) – Mel-Frequency Cepstrum is a combination of many Mel-Frequency Cepstral Coefficients (in short MFCC’s) derived from the non-linear spectrum of spectrums. This feature plays a vital role in speech recognition and also in identifying the singers from the audio input in our case. Though the features obtained from MFCC are not robust to additive noise, it works well and suits the need in obtaining our goal as there is minimum or zero noise in our audio sample dataset.

**Language** – Our dataset contains attribute ‘language’ which indicates to which language the song belongs to. This attribute would be helpful to identify the song language and shows how the songs are distributed among each language. The languages that we have chosen for the project are ‘English’, ‘Hindi’ and ‘Telugu’. Our data consists of more than ten thousand instances for English language, near to six thousand instances for Telugu language and more than six thousand instances for Hindi language.

**1.B - Data Representation: What is the data representation?**

Final Data set is in the form of a csv file with 126 features. To ease the modeling process we have applied dimensionality reduction technqiue such as PCA and reduced the features from 126 to 84 by preserving the 90% variance in the data. Each song has been split into five second intervals and each instance in the data represent the spectral features of that five seconds window.

**Extraction** – Data extracted from ‘Jio Saavn’ and ‘Deezer’ APIs are in the form of JSON files along with the sings which are in MP3/MP4 format. Unwanted fields from this JSON structure have been removed and consumed only the the information that is relevant to our task.

**Storing** – Relevant information from the JSON file is then stored into a dataframe (CSV File) which then underwent pre-processing. MP3/MP4 files were converetd into WAV format before extracting the audio features.

**Processing** – Spleeter module separates the vocals from instrumentals of the WAV file. Pre-processed data (WAV) is then fed to Librosa to extract necessary audio and spectral features.

**Data Dictionary**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Is Null** | **Type** | **Description** | **Example** | **Mean Value** | **Standard Deviation** |
| song\_id | non-null | object | Id of the song | Adam Levine\_2XC5KXWV\_0 |  |  |
| zero\_crossings | non-null | int64 | Number of zero crossings in the song | 161.499023 | 125.9319 | 21.74615 |
| spectral\_centroids\_mean | non-null | float64 | mean calculated for the spectral centroids obtained | 18664 | 13895.13 | 7754.809 |
| spectral\_centroids\_var | non-null | float64 | variance calculated for the spectral centroids obtained | 0.25003098 | 0.659302 | 0.130154 |
| spectral\_rolloff\_mean | non-null | float64 | mean calculated for the spectral roll over obtained (85% of data) | 0.10603317 | 0.02408 | 0.025115 |
| spectral\_rolloff\_var | non-null | float64 | variance calculated for the spectral roll over obtained (85% of data) | 0.70996589 | 0.489446 | 0.090789 |
| spectral-bandwidth\_mean | non-null | float64 | mean calculated for the spectral bandwidth obtained | 0.12650279 | 0.022954 | 0.010368 |
| spectral\_bandwidth\_var | non-null | float64 | variance calculated for the spectral bandwidth obtained | 0.22652298 | 0.469626 | 0.129386 |
| Spectral\_contrast\_mean | non-null | float64 | mean calculated for the spectral contrast obtained | 0.09311502 | 0.028363 | 0.018005 |
| Spectral\_contrast\_var | non-null | float64 | variance calculated for the spectral contrast obtained | 0.2941936 | 0.560467 | 0.143964 |
| Spectral\_flatness\_mean | non-null | float64 | mean calculated for the spectral flatness obtained | 0.14522232 | 0.043551 | 0.02807 |
| Spectral\_flatness\_var | non-null | float64 | variance calculated for the spectral flatness obtained | 0.60158193 | 0.132647 | 0.117009 |
| tempo | non-null | float64 | Number of beats per second | 0.2352872 | 0.033433 | 0.030523 |
| mfcc\_1\_mean to mfcc\_20\_mean | non-null | float64 | mean value for mfcc feature 1 | 0.10672656 | [0.06,0.39] | [0.03,0.18] |
| mfcc\_1\_var to mfcc\_20\_var | non-null | float64 | variance value for mfcc feature 1 | 0.05563151 | [0.06,0.4] | [0.03,0.18] |
| chroma\_stft1\_mean to chroma\_stft12\_mean | non-null | float64 | mean value for stft feature 1 | 0.06300665 | [0.01,0.24] | [0.01,0.11] |
| chroma\_stft1\_var to chroma\_stft12\_var | non-null | float64 | variance value for stft feature 1 | 0.06847341 | [0.01,0.24] | [0.01,0.11] |
| chroma\_cqt1\_mean to chroma\_cqt12\_mean | non-null | float64 | mean value for cqt feature 1 | 0.01182739 | [-214,0.01] | [0.01,201] |
| chroma\_cqt1\_var to chroma\_cqt12\_var | non-null | float64 | variance value for cqt feature 1 | 0.02480159 | [-12,61] | [0.01,44] |
| chroma\_cens1\_mean to chroma\_cens12\_mean | non-null | float64 | mean value for cens feature 1 | 0.10915841 | [-6.9,6389] | [9.1,7183] |
| chroma\_cens1\_var to chroma\_cens12\_var | non-null | float64 | variance value for cens feature 1 | 2.52917933 | [0.5,1440] | [8.8,1114] |
| Language | Non-null | Object | Language in which the song is represented. | English |  |  |
| Artist/genre | non-null | object | Artist or genre respectively for dataset | English\_Adam Levine |  |  |

**1.C - Output: What are you trying to predict? Define the output clearly.**

There are two different tasks which we are attempting to resolve in this project via predictive modelling.

1. Artist Voice Detection - To identify the singer in the song.
2. Genre Classification - To classify the song based on its genre.

For these tasks, we have extracted data from the songs of various artists popular in English, Telugu and Hindi. Here, Telugu and Hindi are the regional languages of India. Similarly, songs from various genres in English, Telugu and Hindi have been extracted. For both the above-mentioned tasks, the target is a categorical variable and it consists of 12 classes for artists data and 10 classes for genre data. Values for the target variable is distributed as below..

Artist Data

* 5 English artists data (Adam Levine, Drake, Justin Beiber, Selena Gomez, The Weeknd)
* 4 Telugu artists data (Anurag, Armaan, Geetha, Sidsriram)
* 3 Hindi artists data (Arijit, Atifaslam, Sonunigam)

Genre Data

* 4 Engilish genre data (Hip-Hop Party, Rock, Romantic, Workout)
* 3 Telugu genre data (Rock, Romantic, Workout)
* 3 Hindi genre data (Rock, Romantic, Workout

**Different Classes of Target Variable for Artist Data (12 Classes in Total)**

|  |  |
| --- | --- |
| **Language and Artist Name** | **Number of Instances** |
| English\_Justin Bieber | 1599 |
| English\_Drake | 1580 |
| English\_Selena Gomez | 1471 |
| English\_Adam Levine | 1229 |
| English\_The Weeknd | 983 |
| Hindi\_sonunigam | 1326 |
| Hindi\_arijit | 1168 |
| Telugu\_geetha | 1162 |
| Hindi\_atifaslam | 1060 |
| Telugu\_armaan | 705 |
| Telugu\_anurag | 468 |
| Telugu\_sidsriram | 365 |

**Different Classes of Target Variable for Genre Data (10 Classes in Total)**

|  |  |
| --- | --- |
| **Language and Genre Name** | **Number of Instances** |
| English\_Rock | 1188 |
| English\_HipHopParty | 1017 |
| English\_Romantic | 895 |
| English\_Workout | 842 |
| Hindi\_rock | 810 |
| Hindi\_romantic | 842 |
| Hindi\_workout | 1021 |
| Telugu\_rock | 648 |
| Telugu\_romantic | 1263 |
| Telugu\_workout | 1133 |

**2. Predictive Models**

1. **What are the methods? Give a general introduction of the methods with references**
2. **Describe the methods with a pseudo code using the definitions in Section 1.**
3. **Justify the choice of the method.**

**2.A - What are the methods? Give a general introduction of the methods with references**

We have chosen 2 preliminary methods in order to perform predictive modelling for our tasks of ‘Singer Identification’ and ‘Genre Classification’. K-Nearest Neighbors Classifier (KNN) and Support Vector Machine Classifier (SVC) are preferred to be best in classification modeling.

**K-Nearest Neighbors (KNN)**

The K-nearest neighbors (KNN) algorithm is one of supervised machine learning algorithms. KNN is probably the simplest model and also frequently used model as it is easy to implement and yet performs complex tasks. The k-nearest neighbor (KNN) classifier is a non-parametric classifier which classifies the test point based on the majority vote of these "k" nearest neighbors. KNN divides complete data into different clusters based on the different classes available in the target variable. KNN assumes that similar data points are near to each other which helps in forming the clusters [1].

KNN calculates the distance between a new data point to all other data points of training dataset. To calculate the nearest neighbors, KNN uses different types of distance calculating mechanisms such as Euclidean distance, Manhattan distance and many other measurements. After calculating the distance, the model selects the K-nearest data points, where K can be any integer. Finally, it classifies the new data point to the class to which the majority of the K data points belong [2].

For example, assume there are 2 classes and the value of k to be 3. The KNN algorithm starts to calculate the distance of the new data point(assume to be X) with all the data points in the training dataset. It then finds out the 3 nearest points with least distance to data point X. The final step of the KNN classifier is to predict the class of the new data point based on the class most common among its k nearest neighbors.

Our data consists of multiple classes for which KNN works better to classify the given test data.

**Support Vector Machine (SVM)**

Support Vector Machine is a supervised machine learning model that can be used for both regression and classification tasks. Although it can be used for regression and classification, majority of its usage can be observed in classification tasks. This algorithm is built on the basis of discovering a line/hyperplane that best divides the data-points into their respective classes [3].

In Support Vector Machine algorithm, each and every data point will be plotted in an n-dimensional space where each dimension represents an individual feature of the chosen dataset. As the name suggests, ‘Support Vectors’ are the co-ordinates of the data points present in our data set, which if removed, would impact the placement of the line/hyperplane separating the classes. If the data is linearly separable, a line is employed to distinctly separate the data points into different classes, else, a hyperplane can be imagined in place of a linear line to perform this separation. Farther the data point that is closest to the hyperplane, better the separation into different classes [4].

Presence of any outliers in the data doesn’t impact the performance of the Support Vector Classifier and is robust to the outliers. For instance, presence of a data point in a class where it does not belong makes it difficult for a line/hyperplane to accurately capture that datapoint in its appropriate class. Hence the SVC ignores this data point and looks for a line/hyperplane that best segregates the classes by using the rest of the data points.

**2.B - Describe the methods with a pseudo code using the definitions in Section 1.**

**K-Nearest Neighbors Classifier Pseudo code**

classify(X, Y, x) // X = training data, Y = target class label data, x = test data

for i in X: // i= each point in train data

compute d(Xi, x) // calculating distance between each train data point to test data

select k points with least d(Xi, x) // k = nearest neighbors, k € ℕ

return max(Yi where i € selected k points) [5]

**Steps**

1. Start.
2. Select the train data along with target label.
3. Select the test data.
4. Choose the distance metrics.
5. Choose the nearest neighbor value(k) where k could be any Integer.
6. For each point in test data:
   1. find the distance to all training data points.
   2. store the distances in a list and sort it in ascending order.
   3. choose the first k points.
   4. assign a class to the test point based on the majority of classes present in the chosen points.
7. End. [6]

**Support Vector Machine Classifier Pseudo Code**

Plotting the data points by selecting total dimensions as the number of features

IF dimension is 2 i.e. number of features

Select the separator as a Line

ELSE IF dimension is greater than 2

Select the separator as a Hyperplane

Process for the selection of optimal position of the Hyperplane

IF all data-points are perfectly separable into their respective classes

Fix the position of the Hyperplane

ELSE IF all data-points are not perfectly separable into corresponding classes

Ignore the outliers i.e. data-points from one class present in other class

IF all data-points are now perfectly separable into their respective classes

Fix the position of the hyperplane

ELSE IF all data points-are not perfectly separable into corresponding classes

Ignore the outliers…

.

.

.

Repeat until all the classes are effectively separated after ignoring outliers

Place the Hyperplane

**Steps**

1. Start.
2. Split the dataset into train and test data.
3. Select the test data.
4. Plot the test data-points onto the n-dimensional plot (where n is the number of features)
5. Choose the optimal position for the line/hyperplane separating the classes.
6. For each point in test data:
   1. Plot the test data-points onto the n-dimensional plot (where n is the number of features)
   2. Choose the optimal position for the line/hyperplane separating the classes.
   3. Ignore the presence of outliers if any.
   4. Repeat above three steps until the line/hyperplane accurately separates all the classes.
7. End. [7] [8]

**Overall Implementation Process**

* Importing Support Vector Classifier from Sklearn’s Support Vector Machine
* Import test-train-split module from Sklearn
* Splitting the input dataset into train and test data sets
* Maintain the ratios using Stratification
* Separate the features from the target variable in Train and Test data
* Create an instance for the Support Vector Classifier
* Fit the features from Train data using the SVC instance
* Predict the target class for the features from Test data using the trained SVC instance

**2.C - Justify the choice of the method.**

The dataset we have generated is of 22k instances with a total of 123 features alongside. Current data is slightly imbalanced as there are more instances for the English language for both artists data and genre data. Upon performing outlier analysis we were able to identify the presence of outliers in the data.

The major criteria to classify these data points would be based on the distance. Since the algorithms which depends on distance between datapoints are KNN and SVC/SVM for classification. KNN is related to grouping of nearest datapoints at a specific distance, whereas SVC is related to the alignment of line/hyperplane in between the support vectors which divides the entire datapoints into classes which is also based on the distance.

In addition to this, for all machine learning algorithms we usually tend to refer to Scikit-learn library for its efficient documentation [9]. Based on references suggested by the Scikit-learn documentation, for any multiclass classification with datapoints less than 100k, KNN classifier is the first choice of algorithm to experiment with. Our dataset consists of 22k data points with multiclass classification. KNN fits best for our dataset for initial experiment. Because of the presence of imbalanced data and outliers, performance of KNN might be impacted and due to this fact, we had to look for another classifier. Since SVC is robust to the presence of outliers, this is our next best option to go ahead and perform classification on our data which is also suggested by the scikit documentation [9].

**Support Vector Machine (SVM)**

**Hyper-Parameters and it’s usage**

1. Random State

By setting a random seed, the data split into train and test will always be consistent, i.e. it helps in generating consistent results across multiple runs rather than producing different data-shuffling for each run.

1. Kernel

Current dataset we are having is not linearly separable and hence to distinctly represent all the classes more accurately, we will be needing a polynomial function rather than a linear function separating the classes. To achieve this, we can use a value ‘poly’ against the parameter ‘Kernel’.

1. Gamma

By changing the gamma value from high to low, the influence of the data-points which are selected as the support vectors by the model can be altered. A high gamma value dictates the influence of support vectors to smaller confinements while a low gamma value increases this boundary. By using ‘auto’ against this parameter, the reciprocal of number of features will be taken as the gamma.

1. Class Weight

If the dataset we have is made up of imbalanced data, ‘class\_weight’ parameter can be used to signify the importance of the minority group by assigning weights which are inversely proportional to the class frequency. While calculating the soft margin, this parameter can be used in assigning different weights to the classes present in our dataset.

1. Break Ties

This parameter is helpful in selecting a class when there is a tie i.e. a random data point is equally probable to be present in either of the classes. In such situations, ‘break\_ties’ parameter uses the confidence value of the ‘decision\_function\_shape’ in order to determine which class the data point should be classified with. By default ‘decision\_function\_shape’ uses one vs rest.

**K-Nearest Neighbors (KNN)**

**Hyper-Parameters and it’s usage**

1. N-neighbors

The number of neighbors considered by the model as the datapoints for model execution, This plays an important role as neighbors might be very close to each other and varying number to a smaller or greater value will effect the results. The optimum was set to 5.

1. Weights

Two weight options are available uniform and distance, In uniform all are treated equally weighed, whereas in distance, the closer ones are giving larger weight and so on.

Separating or varying the importance of datapoints is crucial for our problem as they decide the output. Better results are also obtained with the same using distance as the variable.

1. Algorithm

There are three algorithms, ball\_tree, kd\_tree and brute and the fourth option is auto. Auto decides the best algorithm based on the results from fit method. Fixing one of those algorithms was causing bias to few datapoints, so auto is best suited for our problem. Auto has also delivered the best results compared to rest three options.

1. Metric

There are two metrics to calculate the distance between the datapoints one is Minkowski and the other is Euclidean. Minkowski was of order 1, and Euclidean was of order 2. The distance calculation is important and few changes to it can vary the datapoints. So, this metric is crucial in our tuning. Standard Euclidean has given the best results comparing to Minkowski to our problem.

**3. Evaluations**

1. **What metrics do you use for evaluation?**
2. **What is your ground truth?**
3. **Discuss the performance and the limitation of the method.**

**3.A - What metrics do you use for evaluation?**

Metrics used for evaluating the model’s performance are Accuracy, Precision, Recall, F1. Out of these four metrics, Accuracy can alone give us the results, but accuracy can be misleading sometimes as it tends to get biased to the imbalanced results while training. So, to avoid the greater dependency on the accuracy we are also considering the metrics precision and recall where in both of them deal with the true positives, precision also considers the ratio of true positive to false positive whereas recall considers the ratio of true positive to the false negative. Both are different in their respective perspectives. The ratio of true positive to false positive and the ratio of true positive to false negative both are required for the evaluation.

But to consider as a solution for our problem, both can be useful, and an optimal value of both the precision and recall need to be maintained by the selected trained model. The optimal value of these both metrics can be evaluated using the F1 score, which is a harmonic mean of precision and recall. so, better the F1 value, model performance is better.

Overall, Accuracy, along with the F1 scores are considered for the evaluation of models. F1 internally depends on the precision and recall.

Upon applying both the Support Vector Machines and K-Nearest Neighbor classification algorithms by separating the artist and genre data based on language, we were able to achieve higher levels of accuracy, precision, recall and F1 since there will be less deviations in the data-points because of their similarities in the language as opposed to have the classifiers fed with data from all languages at once. Below tables provide a clear picture of the models’ performance before and after hyper-parameter tuning. These were observed on individua languages at first and then we have applied the same on all languages combined as a single input for artist data and then for genre data.

**Metric Evaluation for Artist Data (All Languages Combined)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metrics** | **Models** | | | |
| **SVC** | | **KNN** | |
| Before  Parameter Tuning | After  Parameter Tuning | Before  Parameter Tuning | After  Parameter Tuning |
| Accuracy | 0.7244 | 0.7820 | 0.6939 | 0.7252 |
| Precision | 0.7440 | 0.7858 | 0.7005 | 0.7259 |
| Recall | 0.7244 | 0.7820 | 0.6939 | 0.7252 |
| F1 | 0.7187 | 0.7817 | 0.6896 | 0.7227 |

**Metric Evaluation for Genre Data (All Languages Combined)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metrics** | **Models** | | | |
| **SVC** | | **KNN** | |
| Before  Parameter Tuning | After  Parameter Tuning | Before  Parameter Tuning | After  Parameter Tuning |
| Accuracy | 0.8442 | 0.8535 | 0.8452 | 0.8597 |
| Precision | 0.8465 | 0.8577 | 0.8494 | 0.8607 |
| Recall | 0.8442 | 0.8535 | 0.8452 | 0.8597 |
| F1 | 0.8433 | 0.8541 | 0.8447 | 0.8595 |

**Metric Evaluation for Artist Data (Language Specific Classification)**

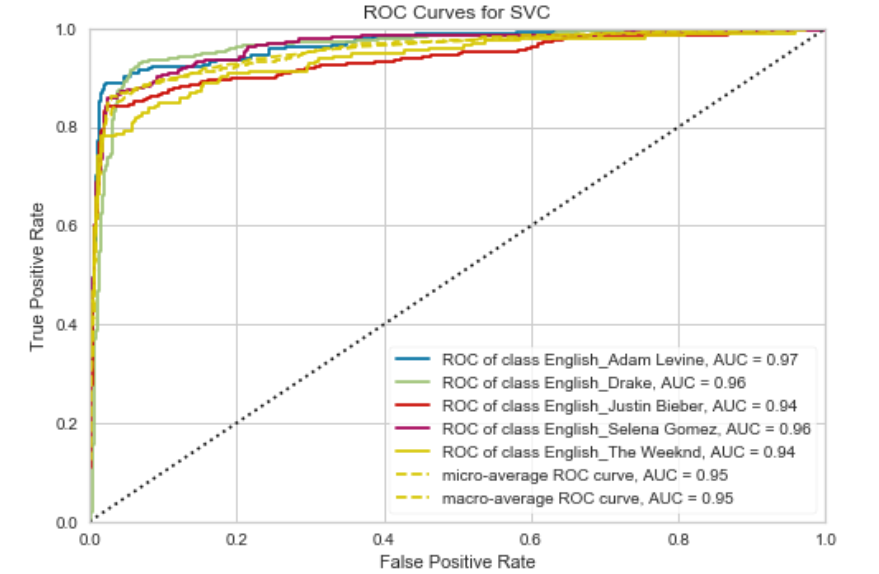
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Language** | **Metrics** | **Models** | | | |
| **SVC** | | **KNN** | |
| Before  Parameter Tuning | After  Parameter Tuning | Before  Parameter Tuning | After  Parameter Tuning |
| English | Accuracy | 0.839 | 0.865 | 0.805 | 0.824 |
| Precision | 0.848 | 0.868 | 0.811 | 0.826 |
| Recall | 0.839 | 0.865 | 0.805 | 0.824 |
| F1 | 0.839 | 0.865 | 0.804 | 0.823 |
| Telugu | Accuracy | 0.664 | 0.775 | 0.725 | 0.746 |
| Precision | 0.687 | 0.775 | 0.73 | 0.746 |
| Recall | 0.664 | 0.775 | 0.725 | 0.746 |
| F1 | 0.638 | 0.773 | 0.723 | 0.74 |
| Hindi | Accuracy | 0.81 | 0.832 | 0.755 | 0.777 |
| Precision | 0.813 | 0.833 | 0.769 | 0.782 |
| Recall | 0.81 | 0.832 | 0.755 | 0.777 |
| F1 | 0.81 | 0.832 | 0.757 | 0.778 |

**Metric Evaluation for Genre Data (Language Specific Classification)**

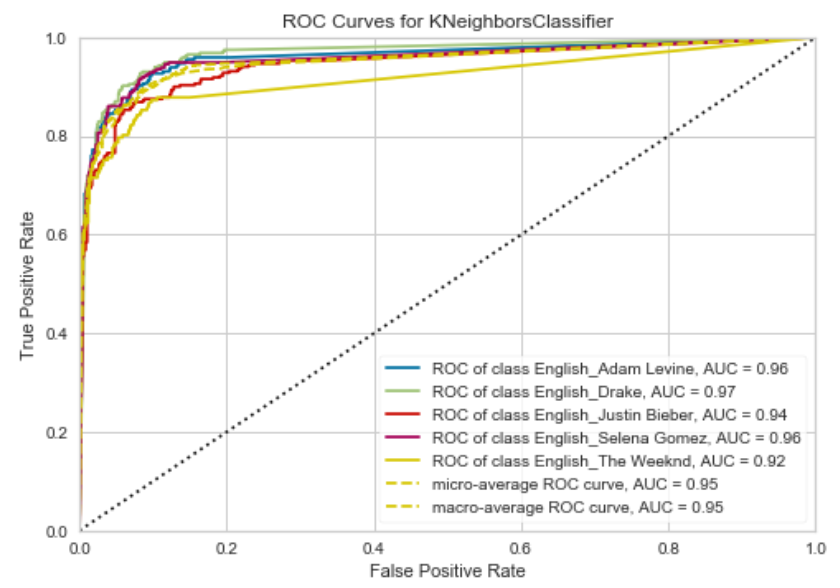
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Language** | **Metrics** | **Models** | | | |
| **SVC** | | **KNN** | |
| Before Parameter Tuning | After Parameter Tuning | Before Parameter Tuning | After Parameter Tuning |
| English | Accuracy | 0.901 | 0.944 | 0.929 | 0.932 |
| Precision | 0.905 | 0.945 | 0.929 | 0.933 |
| Recall | 0.901 | 0.944 | 0.929 | 0.932 |
| F1 | 0.901 | 0.944 | 0.928 | 0.932 |
| Telugu | Accuracy | 0.835 | 0.871 | 0.873 | 0.87 |
| Precision | 0.854 | 0.877 | 0.873 | 0.871 |
| Recall | 0.835 | 0.871 | 0.873 | 0.87 |
| F1 | 0.831 | 0.873 | 0.873 | 0.87 |
| Hindi | Accuracy | 0.942 | 0.943 | 0.943 | 0.947 |
| Precision | 0.943 | 0.944 | 0.944 | 0.947 |
| Recall | 0.942 | 0.943 | 0.943 | 0.947 |
| F1 | 0.942 | 0.943 | 0.943 | 0.947 |

**Receiver Operating Characteristic Curve – Artists Data (Individual Languages)**

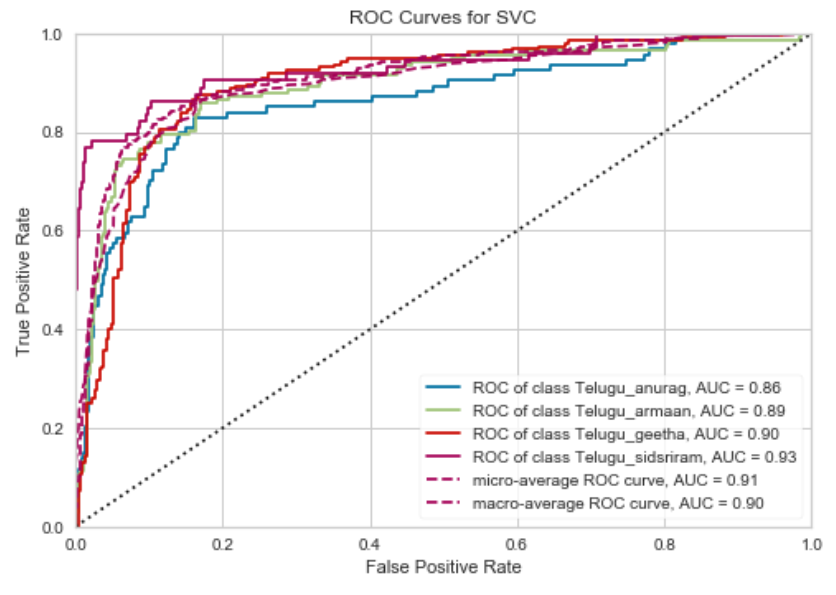
1. **English Artists – SVM**



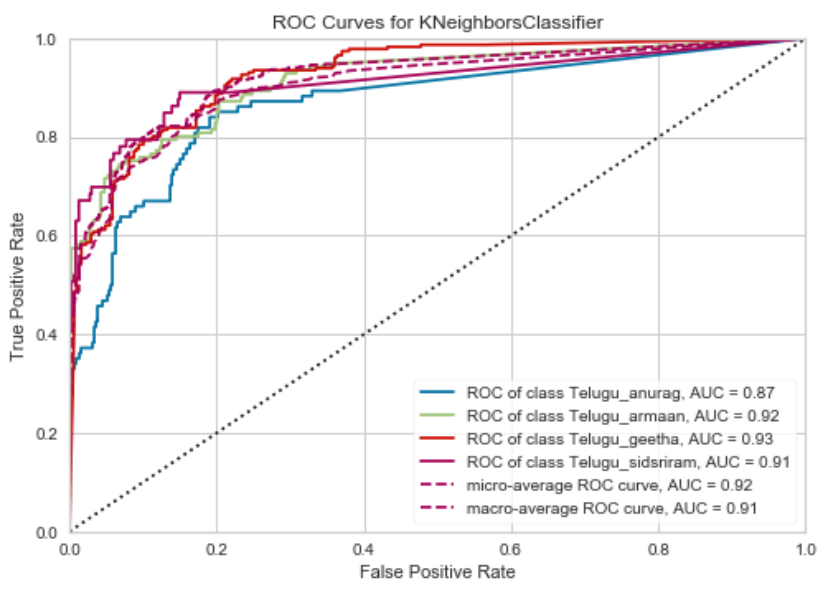
1. **English Artists – KNN**



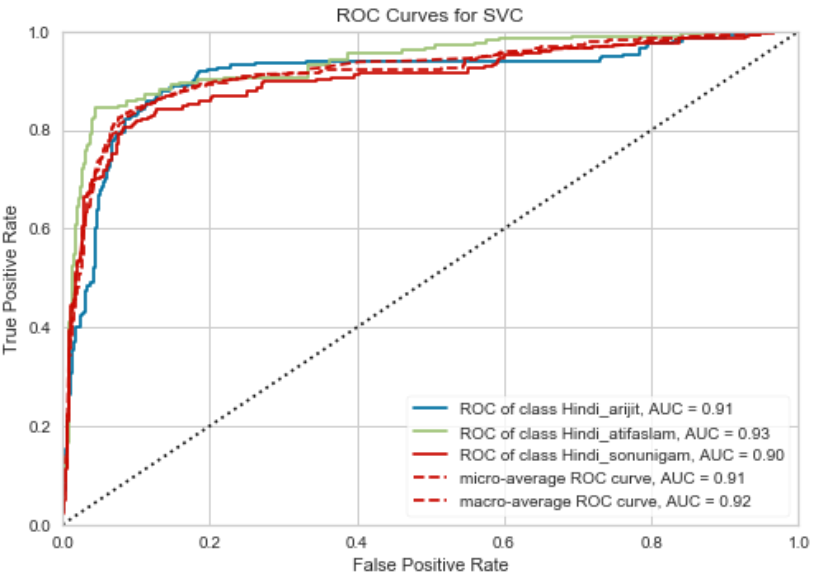
1. **Telugu Artists – SVM**



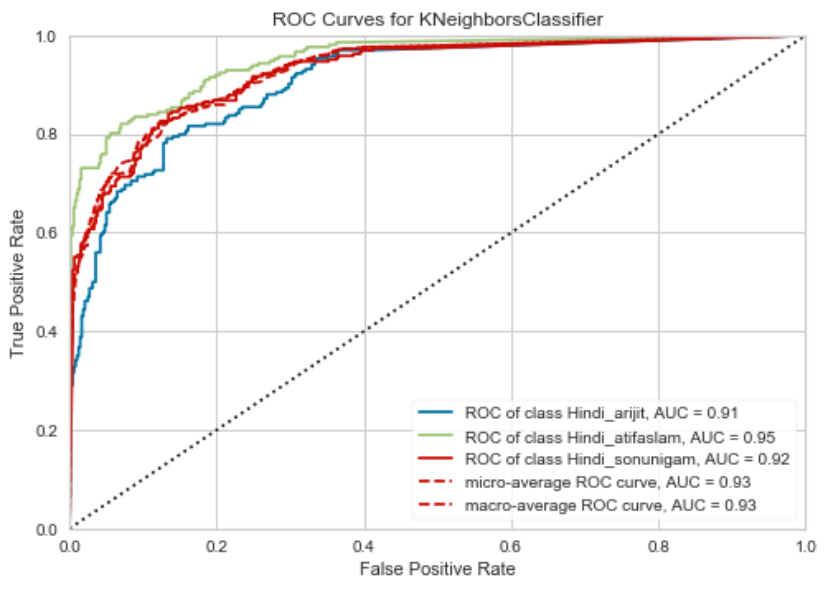
1. **Telugu Artists – KNN**



1. **Hindi Artists – SVM**

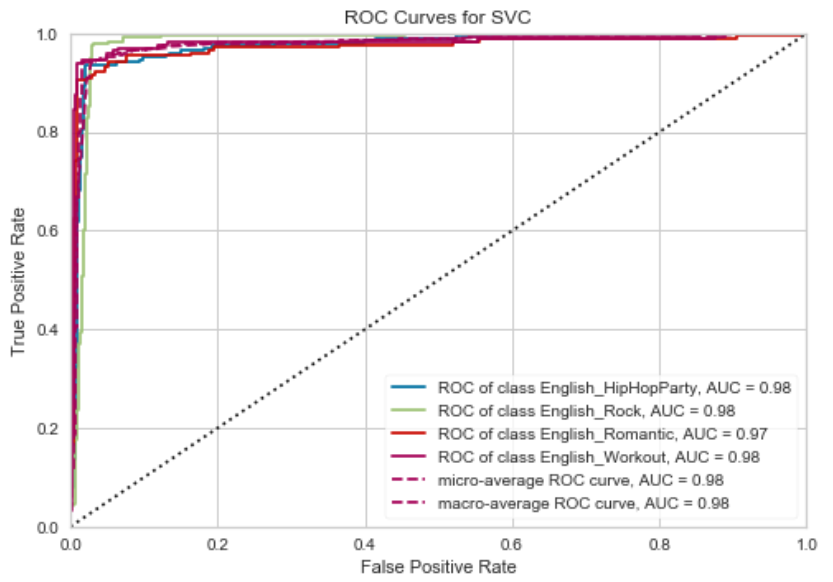


1. **Hindi Artists – KNN**

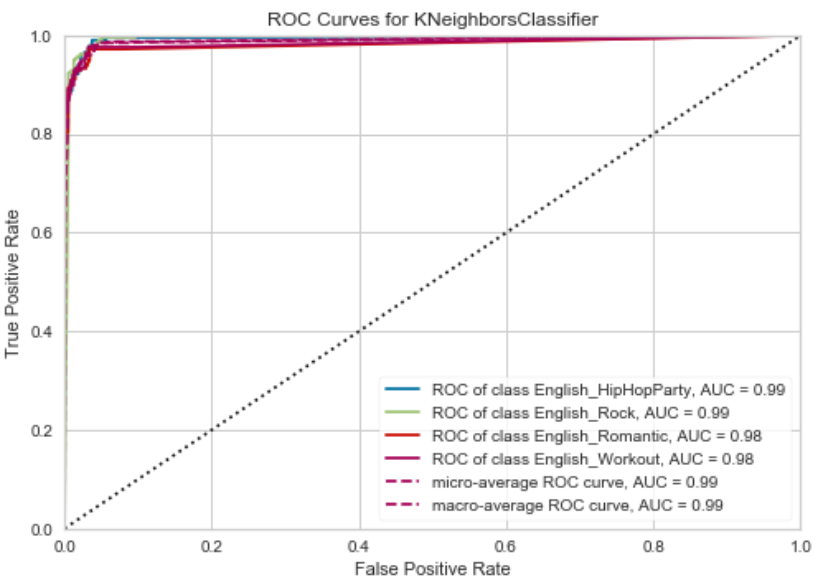


**Receiver Operating Characteristic Curve – Genre Data (Individual Languages)**

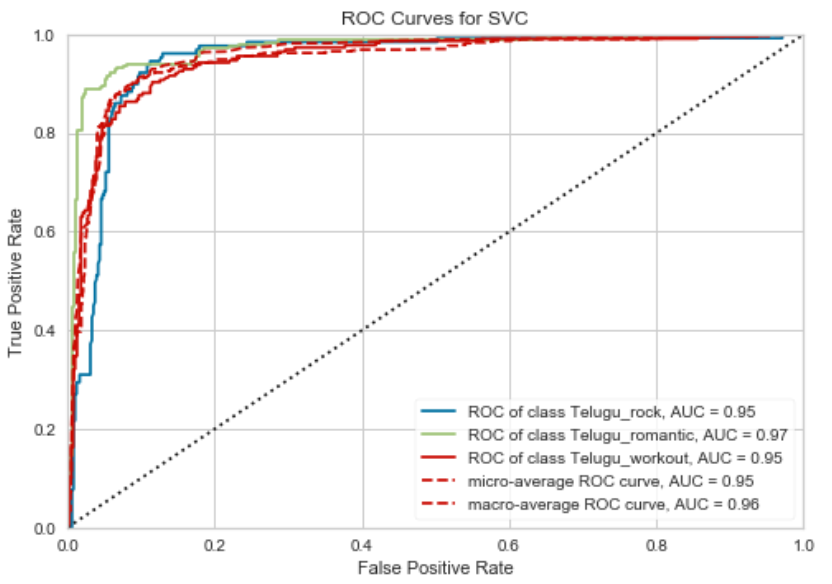
1. **English Genre – SVM**



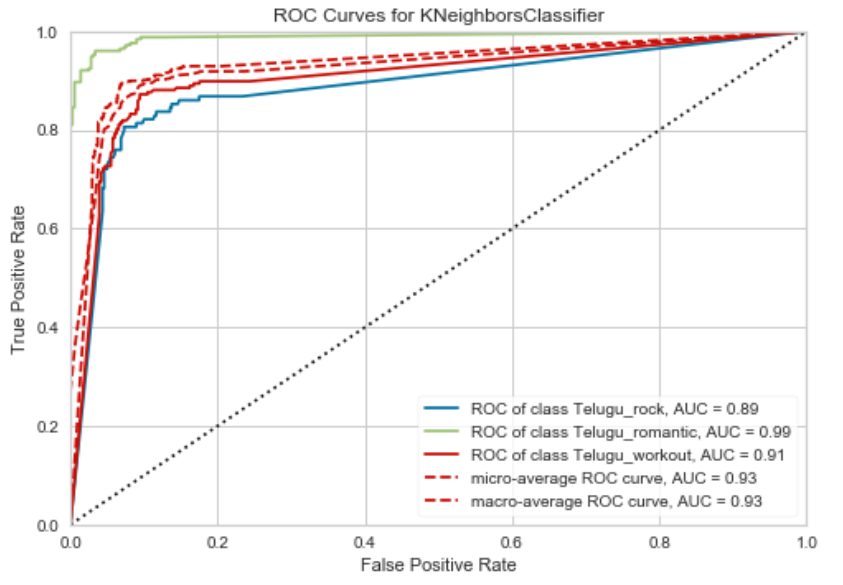
1. **English Genre – KNN**



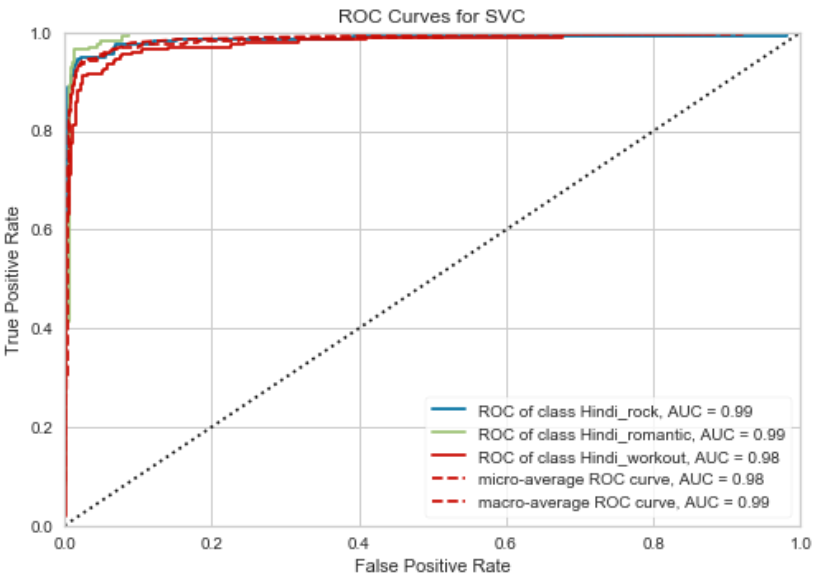
1. **Telugu Genre – SVM**



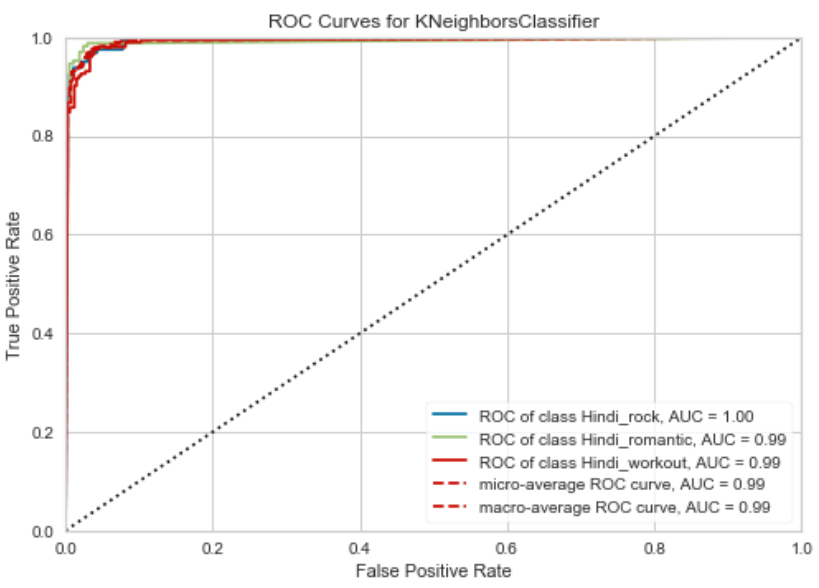
1. **Telugu Genre – KNN**



1. **Hindi Genre – SVM**



1. **Hindi Genre – KNN**

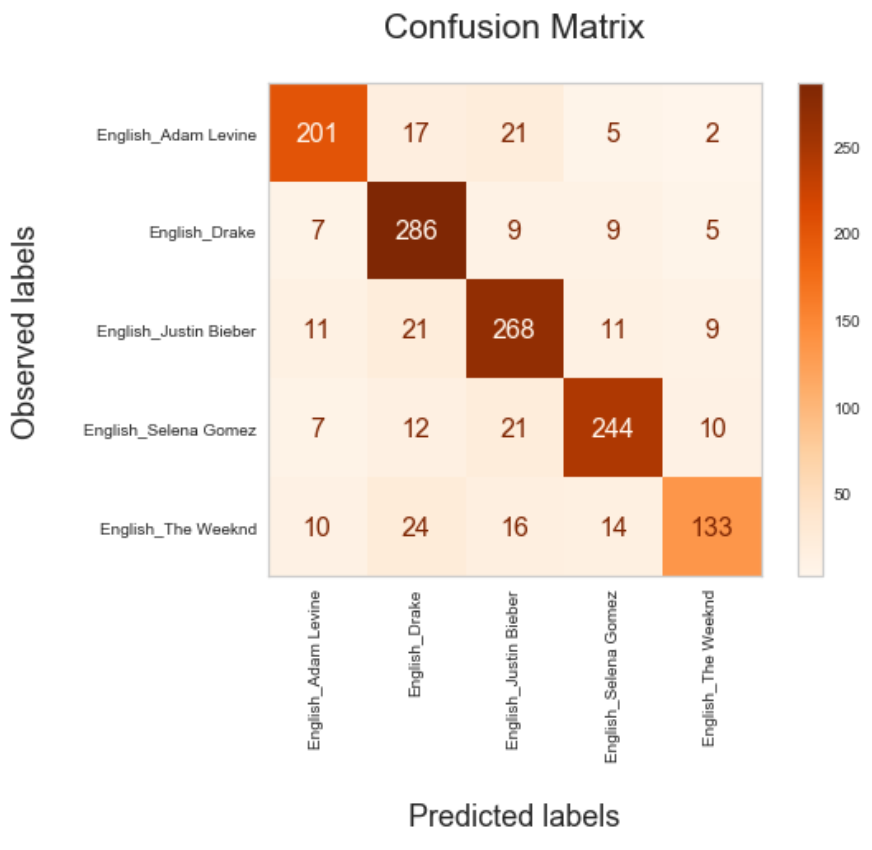


**Confusion Matrix On Artist Data (Individual Languages)**

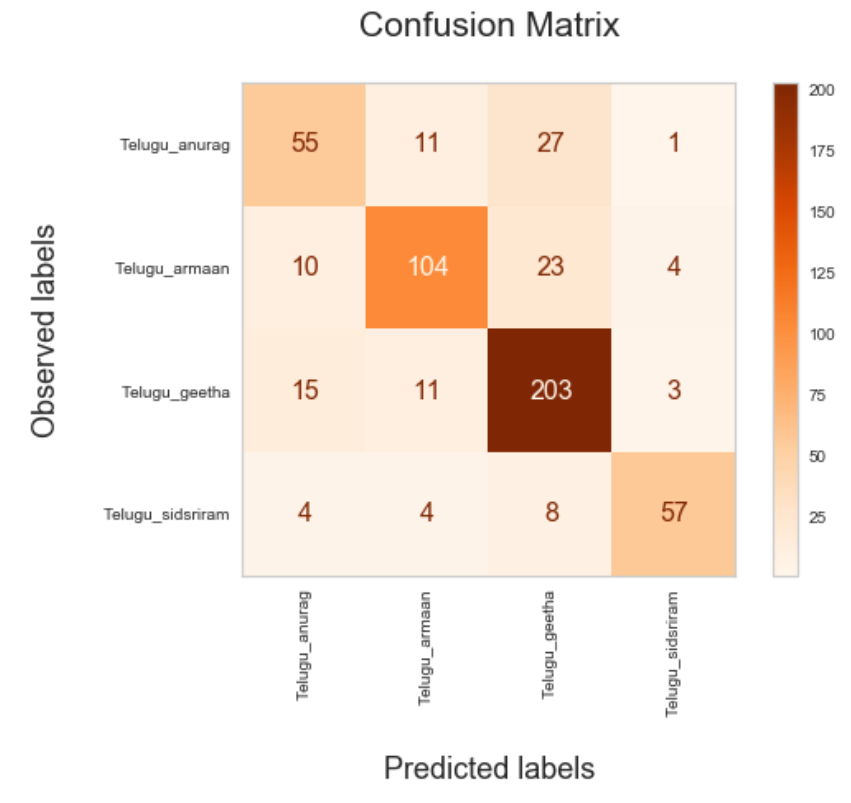
1. **English Artists – SVM**



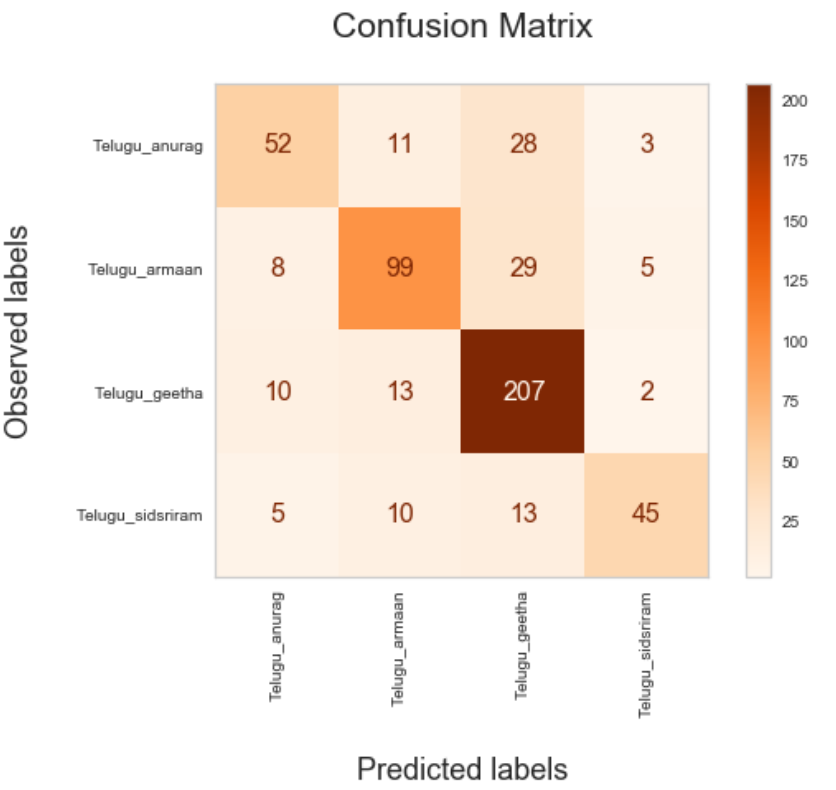
1. **English Artists – KNN**



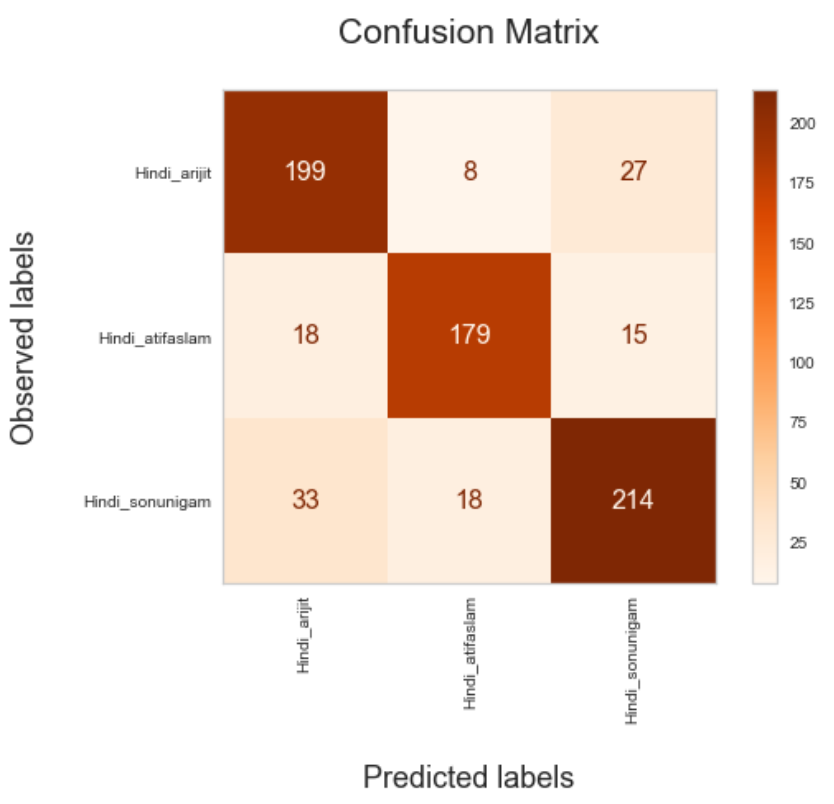
1. **Telugu Artists – SVM**



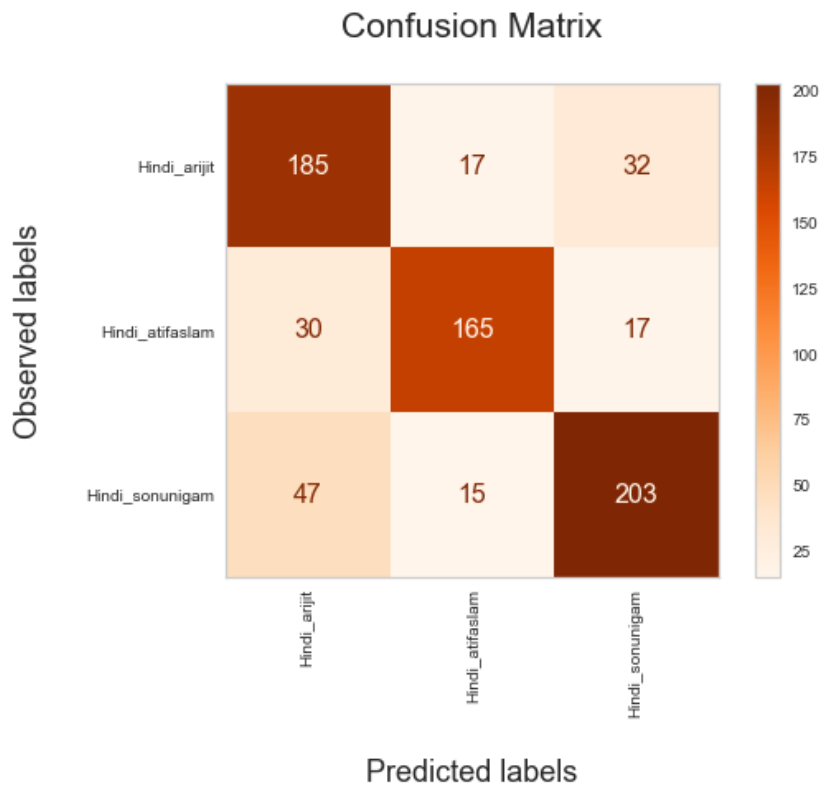
1. **Telugu Artists – KNN**



1. **Hindi Artists – SVM**

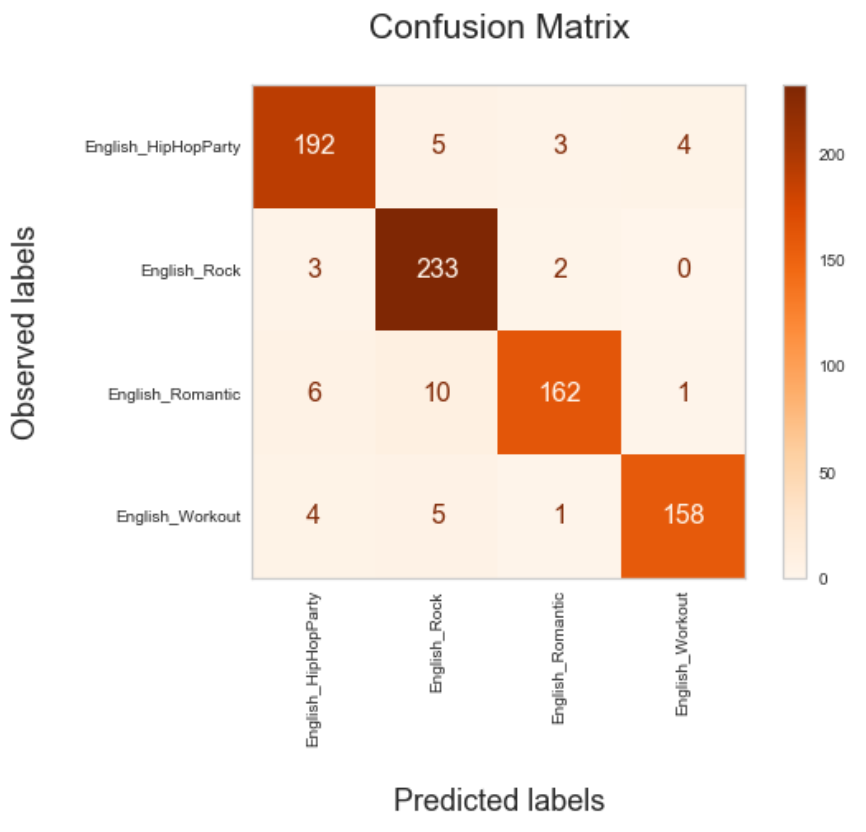


1. **Hindi Artists – KNN**

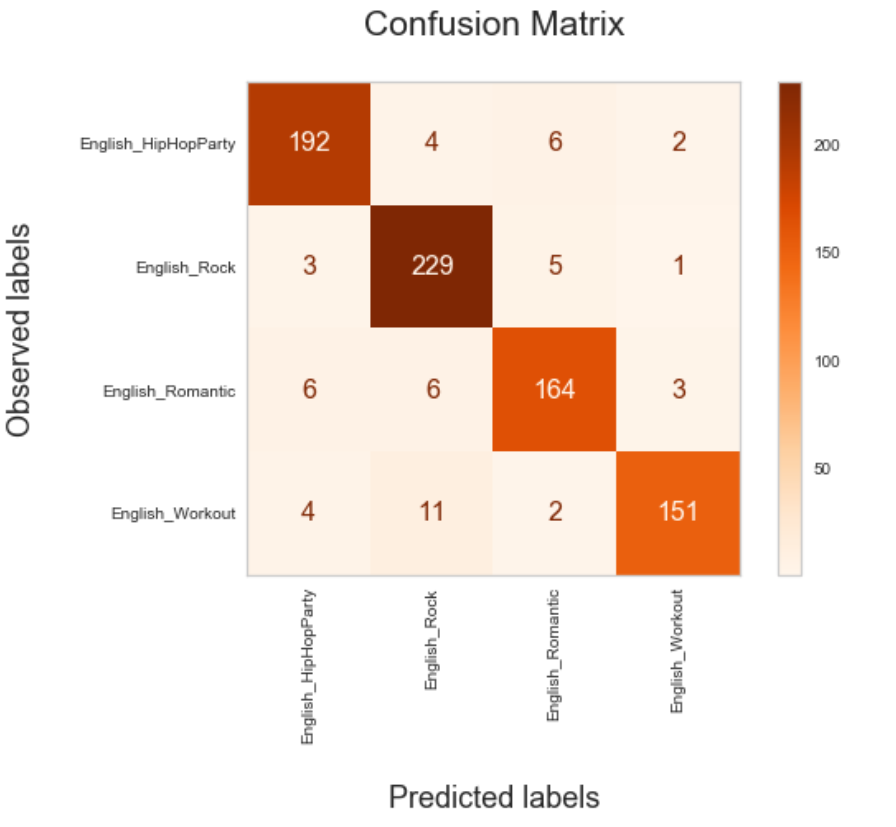


**Confusion Matrix On Genre Data (Individual Languages)**

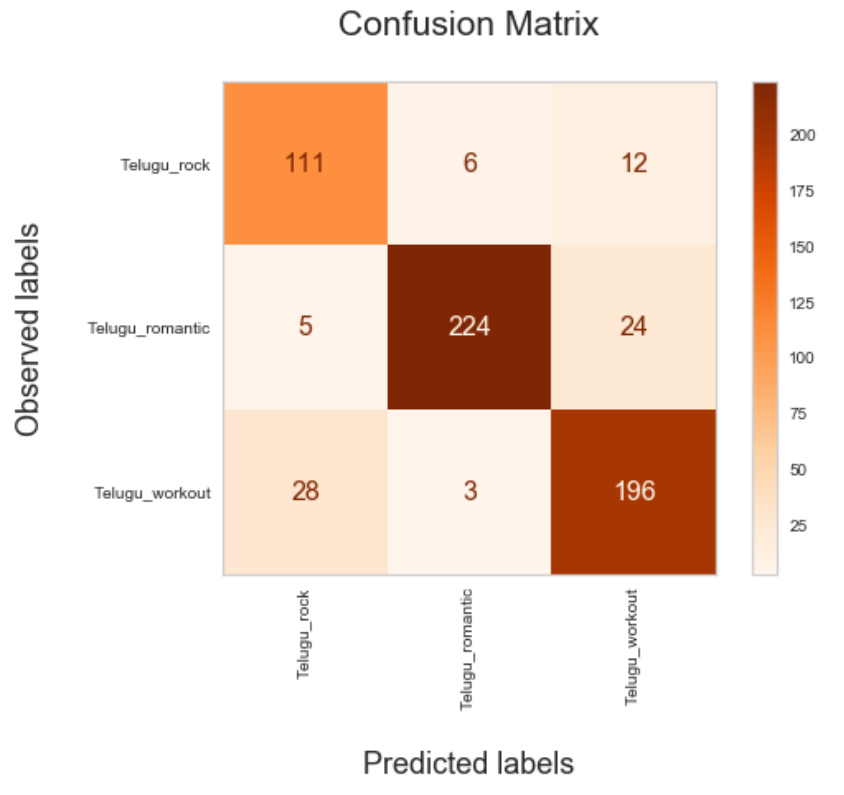
1. **English Genre – SVM**



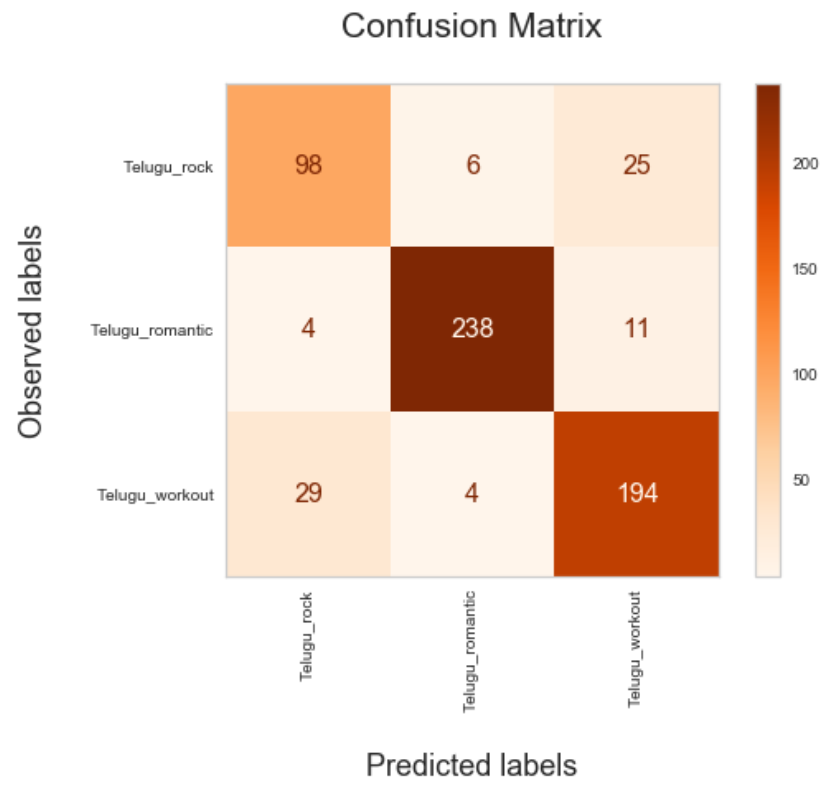
1. **English Genre – KNN**



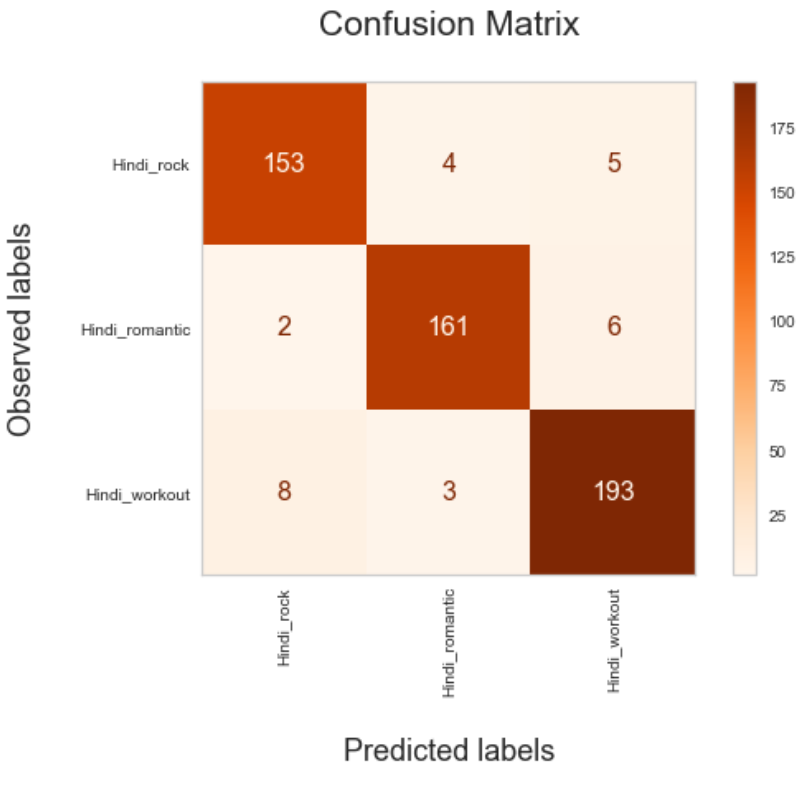
1. **Telugu Genre – SVM**



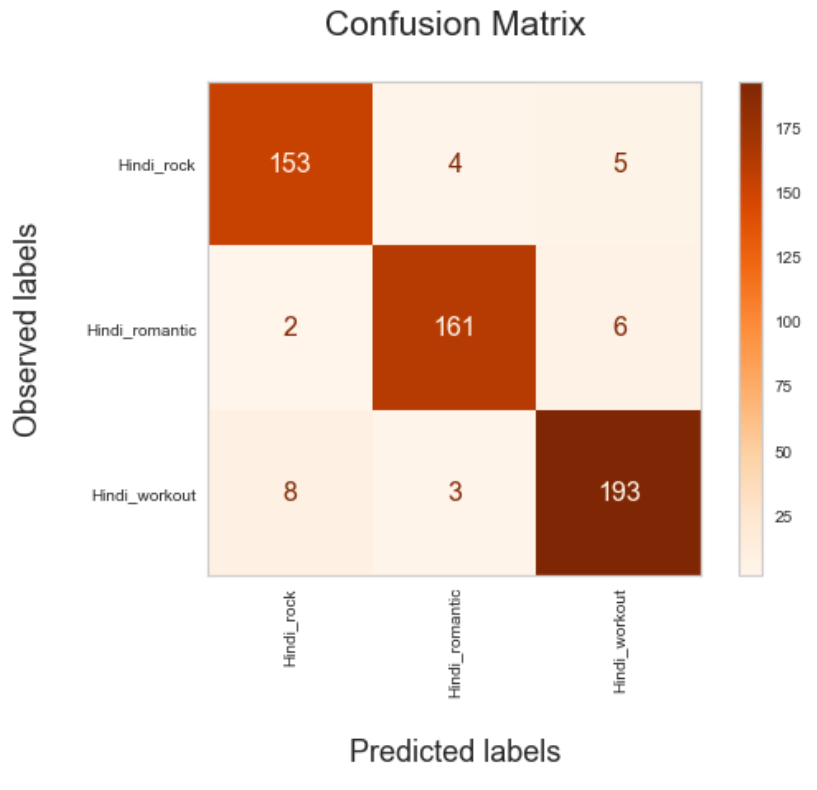
1. **Telugu Genre – KNN**



1. **Hindi Genre – SVM**



1. **Hindi Genre – KNN**



Since the results from Accuracy, Precision, Recall and F1 are similar for both KNN and SVM, we wanted to analyze ROC and confusion matrix to capture the False Positives case in order to better evaluate which model performs effectively. This analysis was performed on individual languages rather than clubbing them all to extract the effectiveness of models on different languages.

On observing the ROC curves, it is understood that for SVM mostly it is near to ideal case, whereas for KNN it has got deviated down from the ideal case. On having a look into the confusion matrices obtained from various scenarios, it can be depicted that most of the time true positives are correctly interpreted by the SVM than the KNN which lacks a bit. Hence we could observe that SVM classifier is having slightly better performance when compared to the KNN classifier when it comes to False Positive cases.

**3.B - What is your ground truth?**

The models considered are SVM and KNN, and two scenarios are currently being trained using these two models. Scenario 1 is using these models on the individual languages; Scenario 2 is using the aggregated dataset of all languages irrespective of the filtering.

We are using accuracy as metric to evaluate the ground truth, KNN on the whole dataset is giving about 90 percent of accuracy on the training dataset, whereas the SVM is giving around 94 percent of the accuracy on the training dataset. While dealing with the entire dataset feeding to the models, we are getting the error of around 6 to 7 percent of the total datapoints for SVM and around 10 to 11 percent of datapoints were misclassified while training the KNN model.

While considering the other scenario, filtering out and separating the datapoints with respect to languages and applying the SVM and KNN models. In these cases, we observed a higher accuracy of around 98 to 100. This shows the classification in these cases is being done more perfectly. The error percentage with respect to the accuracy is around 0 to 2 percent datapoints are mis classified in these scenarios.

**3.C - Discuss the performance and the limitation of the method.**

**Support Vector Machine:**

**In Artist classification**

Considering the whole dataset together, SVM is giving around 72 percent of accuracy and 71 as F1 Score, after hyperparameter tuning the results have increased to 78.2 and 78.1 respectively.

While considering the individual languages separately, accuracy and F1 are better for English dataset followed by Hindi and least is with Telugu(regional). Accuracy values range from 77 to 86.

**In genre classification**

The results are comparably higher for genre classification with respect to artist classification.

On the overall dataset accuracy is about 85 percent after hyperparameter tuning which was 84 before the tuning.

While considering the individual datasets, English and Hindi datasets are giving an accuracy of around 94 percent and Telugu is comparatively less which is around 87.

Accuracy and F1 scores are almost equal for all the SVM results.

Hyper parameter tuning is affecting a lot on the performance of the model. Before and after tuning results are greatly varying because of the support vectors alignment.

In comparison with the KNN, SVM is performing better in terms of the accuracy and F1. SVM is better performing than KNN while speaking about results in both individual languages and also considering the whole dataset

**Limitation -** SVM is very flexible and biased to tuning. As there are so many tuning parameters, accuracy is highly varying before and after tuning. The best model is only obtained after proper hyperparameter tuning and we cannot fix a single model, as change in training set or any change in dimensions could result in observable change in results.

**KNN**

**In artist classification**

Considering the whole dataset together from all languages, the accuracy obtained is around 69, which was increased to 72 after tuning.

While handling the individual languages, the results were obtained higher for English language with 82 percent of accuracy after the tuning of the model, followed by Hindi language with 77 percent of accuracy after the tuning, and Telugu or regional language stands at the last position with 72 percent of accuracy after the tuning.

**In genre classification**

Considering all the languages together, accuracy is around 85 percent after tuning, which was 84 before training.

While handling with individual languages, Hindi stands first with almost around 94 percent of accuracy, next is English with around 93 percent and Telugu is around 87 percent of accuracy.

KNN models are giving very less results in comparison with SVM after tuning, but these results are almost standing stable in comparison to tuning. So, KNN is comparatively stable.

**Limitation -** Prediction time may get increased with larger N. KNN relies on all the dimensions, it is very sensitive to all the irrelevant features and scaling is effecting the results a lot.

**References**

1. <https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm>
2. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
3. <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
4. <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
5. <https://www.researchgate.net/publication/260397165_A_Machine_Learning_Approach_for_Specification_of_Spinal_Cord_Injuries_Using_Fractional_Anisotropy_Values_Obtained_from_Diffusion_Tensor_Images/figures?lo=1>
6. <https://medium.com/@sonish.sivarajkumar/k-nearest-neighbours-knn-algorithm-9900c1427726>
7. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
8. <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
9. <https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>

**Appendix**

Owing to the large size of our notebook, instead of embedding them within this document, we will be uploading our code separately to the blackboard in the name ‘G1\_Modelling\_V1\_02.12.2021.ipynb’

Analyzing basic metrics, descriptive statistics, ROC and Confusion Matrix plots are all present in the notebook named ‘G1\_Modelling\_V1\_02.12.2021.ipynb’.

‘Artist\_Data\_PCA’ and ‘Genre\_Data\_PCA’ are the two csv files which acts as input towards this notebook.

