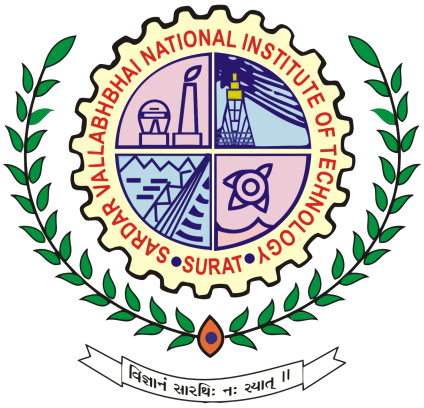


Sardar Vallabhbhai National Institute of Technology, SVNIT

Course Title: Project preliminaries

Course Code : CS405

Final Year Project 1.0

Topic :Music Genre Classification

Submitted By :
(U19CS082)
(U19CS056)
(U19CS004)
(U19CS067)
Dept.of CSE

Submitted To :
DR. BALU L. PARNE
Assistant Professor, Ph. D.
blparne@coed.svnit.ac.in
baluparne@gmail.com

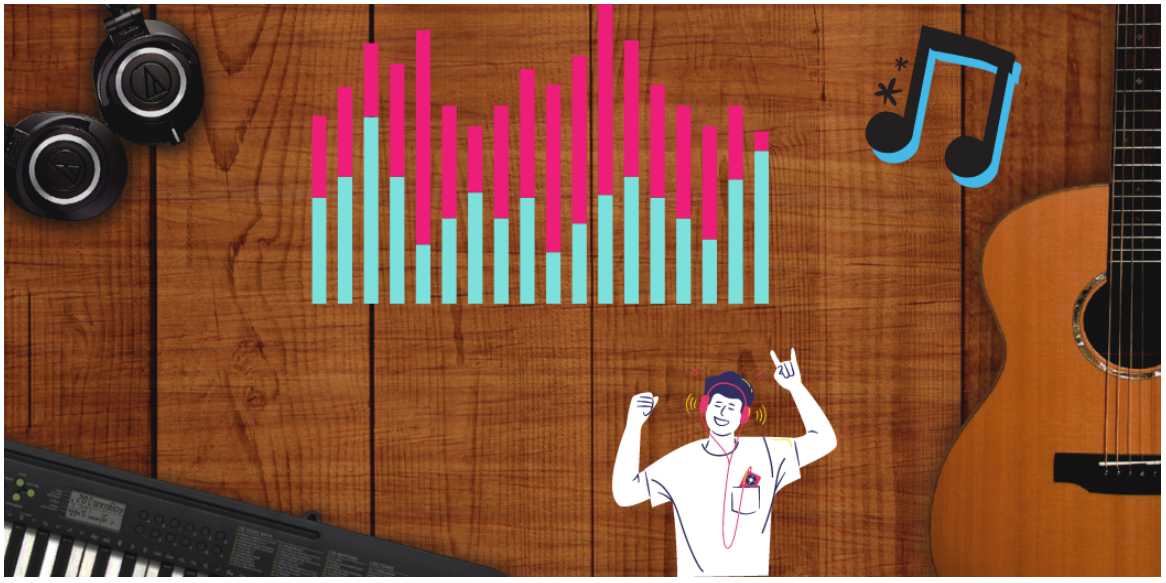
Contents

1 INTRODUCTION

Audio processing is one of the most complex tasks in data science as compared to image processing and other classification techniques. One such application is music genre classification which aims to classify the audio files in certain categories of sound to which they belong.

The application is very important and requires automation to reduce the manual error and time because if we have to classify the music manually then one has to listen out each file for the complete duration.

So To automate the process we use Machine learning and deep learning algorithms known as the K-Nearest Neighbors classification algorithm and this is what we will implement in this Project.



2 PROJECT OVERVIEW AND APPROACH

In short, we can define our project problem statement as like given multiple audio files, and the task is to categorize each audio file in a certain category like audio belongs to Disco, hip-hop, etc.

The music genre classification can be built using different approaches in which the top 4 approaches that are mostly used are listed below.

1. **Linear Regression**
2. **Naives Bayes**
3. **K-Nearest Neighbors (KNN)**
4. **Desicion Tree**
5. **Support Vector Machine (SVM)**
6. **K-means clustering algorithm**
7. **Convolutional neural network**

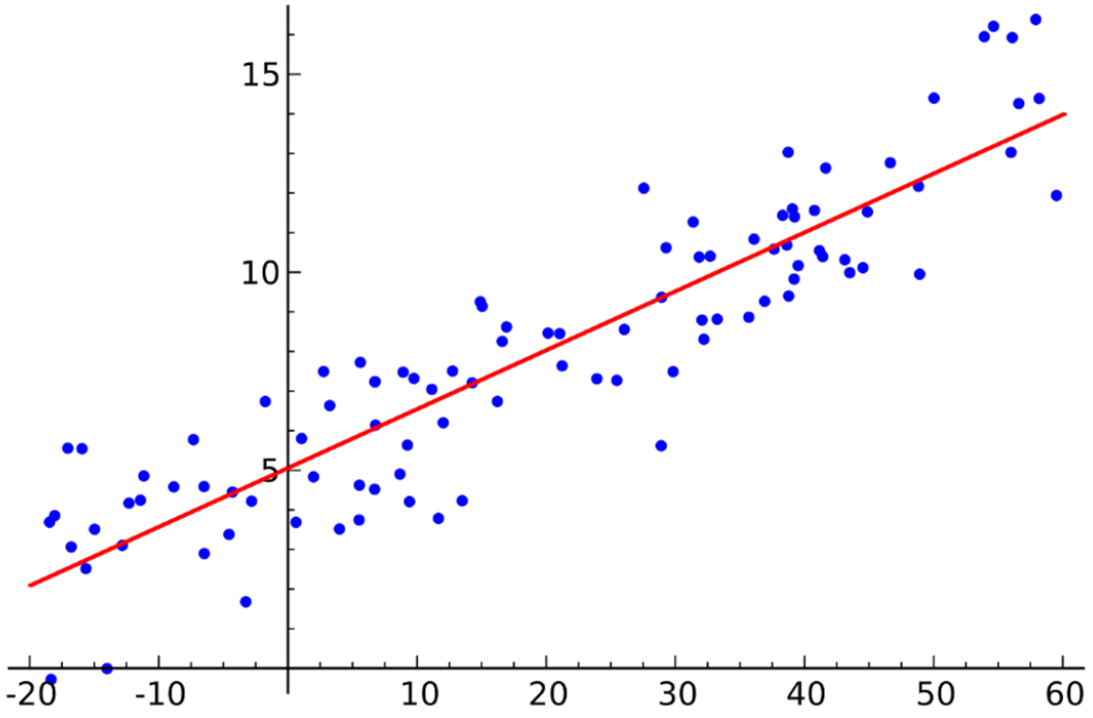
We will use K-Nearest Neighbors algorithm because various researches prove it is one of the best algorithms to give good performance and till time along with optimized models organizations uses this algorithm in recommendation systems as support.

3 DIFFERENT ALGORITHMS THAT CAN BE USED

3.1 LINEAR REGRESSION

Before knowing what is linear regression, let us get ourselves accustomed to regression. Regression is a method of modelling a target value based on independent predictors.

This method is mostly used for forecasting and finding out cause and effect relationship between variables. Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.



Linear Regression Simple linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent(x) and dependent(y) variable. The red line in the above graph is referred to as the best fit straight line. Based on the given data points, we try to plot a line that models the points the best.
The line can be modelled based on the linear equation shown below.

y = a0 + a1 * x ———————— Linear Equation

The motive of the linear regression algorithm is to find the best values for a0 and a1. Before moving on to the algorithm, let's have a look at two important concepts you must know to better understand linear regression.

LinearRegression.py

```
1  from sklearn.linear_model import LinearRegression
2  from sklearn.metrics import r2_score
3  clf = LinearRegression(normalize=True)
4  clf.fit(x_train,y_train)
5  y_pred = clf.predict(x_test)
6  print(r2_score(y_test,y_pred))
7
```

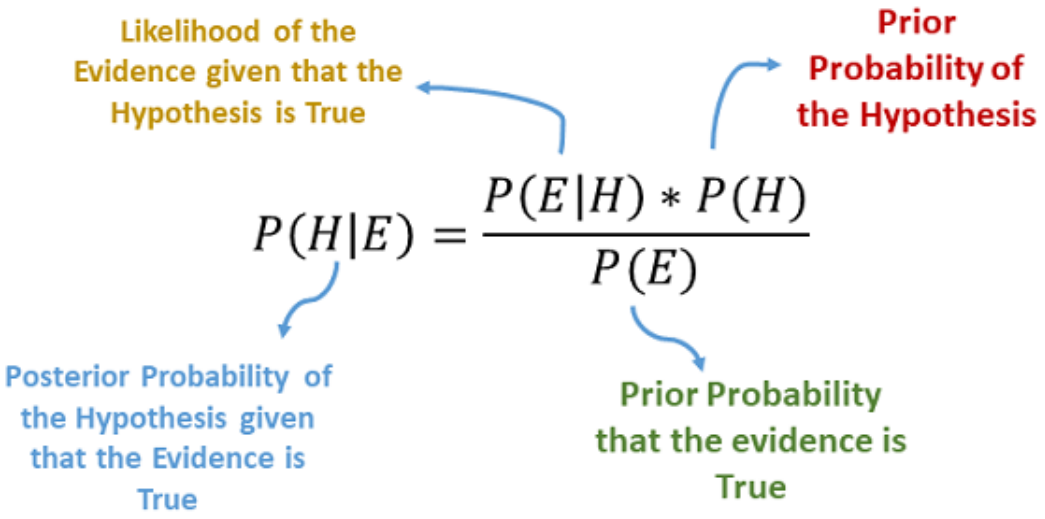
3.2 NAIVES BAYES

A classifier is a machine learning model segregating different objects on the basis of certain features of variables. It is a kind of classifier that works on the Bayes theorem. Prediction of membership probabilities is made for every class such as the probability of data points associated with a particular class.

The class having maximum probability is appraised as the most suitable class. This is also referred to as Maximum A posteriori (MAP).
The MAP for a hypothesis is:

MAP(H) = max P((H—E))
MAP(H) = max P((H—E) * (P(H))/P(E))
MAP(H) = max P((H—E) * P(H))

P (E) is evidence probability, and it is used to normalize the result. The result will not be affected by removing (E) .



NavesBayes.py

```
1  from sklearn.datasets import load_iris
2  from sklearn.model_selection import train_test_split
3  from sklearn.naive_bayes import GaussianNB
4  X, y = load_iris(return_X_y=True)
5  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,random_state=0)
6  gnb = GaussianNB()
7  y_pred = gnb.fit(X_train, y_train).predict(X_test)
8
```

3.3 KNN (K-Nearest Neighbors)

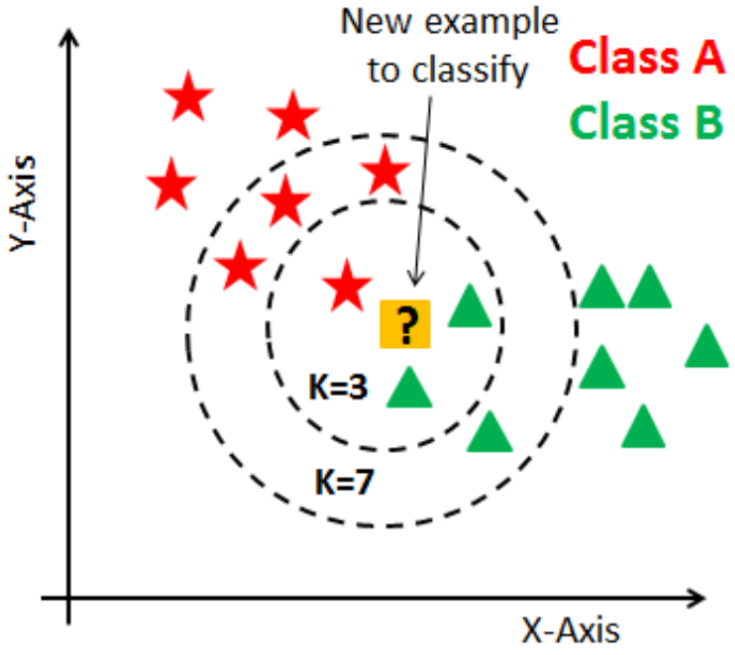
The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point.

For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k.

Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your data set.

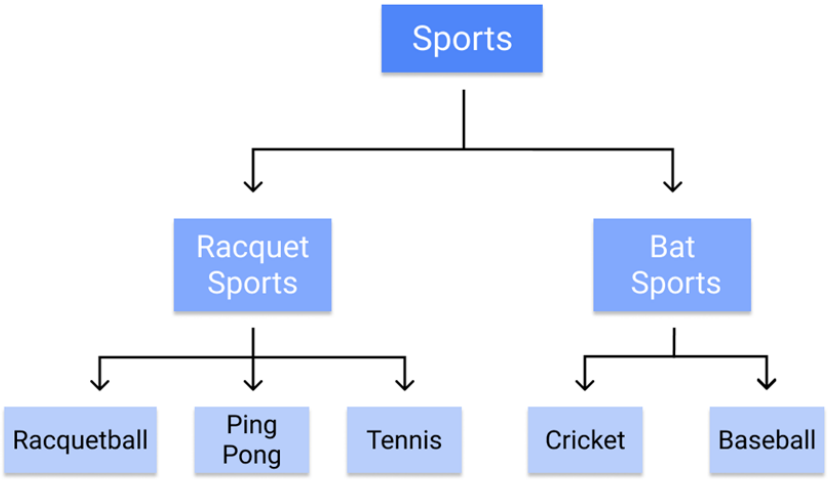


Knn.py

```
from sklearn.neighbors import KNeighborsClassifier
model_name = 'K-Nearest Neighbor Classifier'
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p=2)
knn_model = Pipeline(steps=[('preprocessor', preprocessorForFeatures), ('classifier' , knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

3.4 DECISION TREE

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it's able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision. To continue with the sports example, this is how the decision tree works:

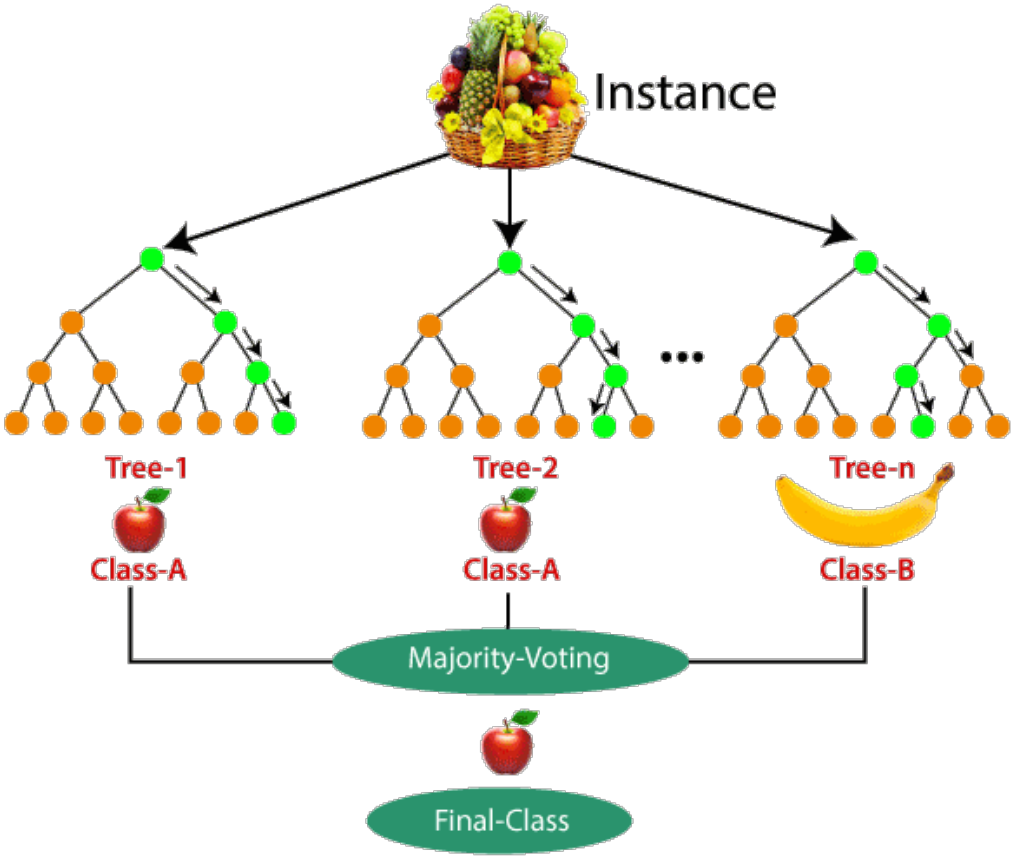


DescisionTree.py

```
1   from sklearn.datasets import load_iris
2   from sklearn import tree
3   iris = load_iris()
4   X, y = iris.data, iris.target
5   clf = tree.DecisionTreeClassifier()
6   clf = clf.fit(X, y)
7
```

3.5 Random Forest

The random forest algorithm is an expansion of decision tree, in that you first construct a multitude of decision trees with training data, then fit your new data within one of the trees as a "random forest." It, essentially, averages your data to connect it to the nearest tree on the data scale. Random forest models are helpful as they remedy for the decision tree's problem of "forcing" data points within a category unnecessarily.

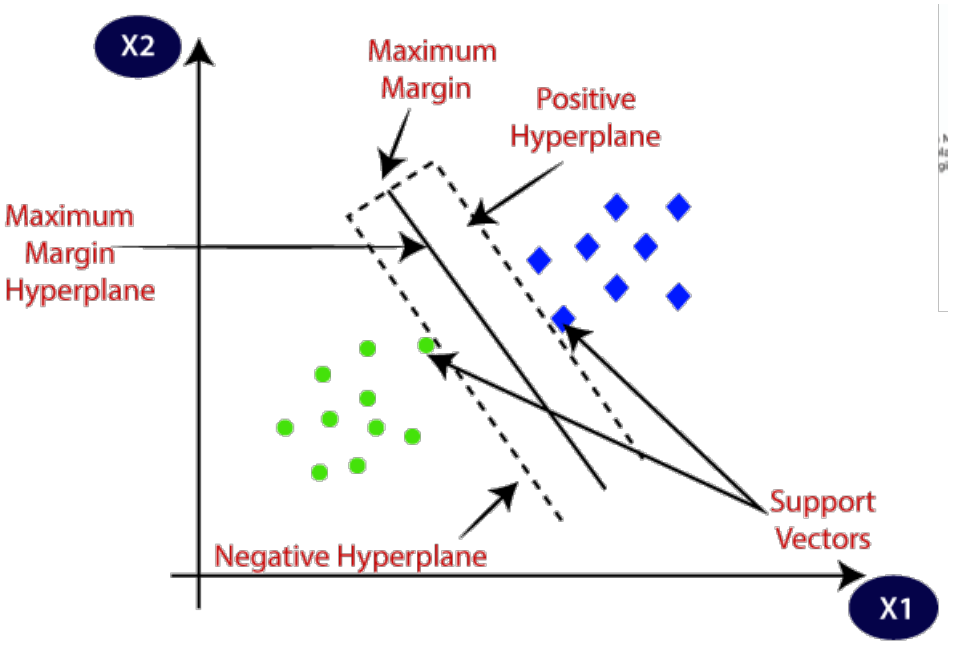


3.6 SVM (Support Vector Machine)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate

n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Example:
SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat.



Svm.py

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear')
# Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

```

4 COMPARISON OF DIFFERENT ALGORITHMS

4.1 LR vs Decision Tree:

• Decision trees supports non linearity, where LR supports only linear solutions.
• When there are large number of features with less data-sets(with low noise), linear regressions may outperform Decision trees/random forests. In general cases, Decision trees will be having better average accuracy.
• For categorical independent variables, decision trees are better than linear regression.
• Decision trees handles collinearity better than LR.

4.2 LR vs SVM:

• SVM supports both linear and non-linear solutions using kernel trick.
• SVM handles outliers better than LR.
• Both perform well when the training data is less, and there are large number of features.

4.3 LR vs KNN

• KNN is a non-parametric model, whereas LR is a parametric model.
• KNN is slow in real time as it have to keep track of all training data and find the neighbor nodes, whereas LR can easily extract output from the tuned coefficients.

4.4 KNN vs naive bayes:

• Naive bayes is much faster than KNN due to KNN's real-time execution.
• Naive bayes is parametric whereas KNN is non-parametric.

4.5 KNN vs linear regression :

• KNN is better than linear regression when the data have high SNR.

4.6 KNN vs SVM :

• SVM take cares of outliers better than KNN.
• If training data is much larger than no. of features(m¿¿n), KNN is better than SVM. SVM outperforms KNN when there are large features and lesser training data.

4.7 Decision tree vs Random Forest :

• Random Forest is a collection of decision trees and average/majority vote of the forest is selected as the predicted output.
• Random Forest model will be less prone to overfitting than Decision tree, and gives a more generalized solution.
• Random Forest is more robust and accurate than decision trees.

4.8 Decision tree vs KNN :

• Both are non-parametric methods.
• Decision tree supports automatic feature interaction, whereas KNN cant.
• Decision tree is faster due to KNN's expensive real time execution.

4.9 Decision tree vs naive Bayes :

• Decision tree is a discriminative model, whereas Naive bayes is a generative model.
• Decision trees are more flexible and easy.
• Decision tree pruning may neglect some key values in training data, which can lead the accuracy for a toss.

4.10 Decision tree vs SVM :

• SVM uses kernel trick to solve non-linear problems whereas decision trees derive hyper-rectangles in input space to solve the problem.
• Decision trees are better for categorical data and it deals collinearity better than SVM.

4.11 SVM vs Random Forest :

• Random Forest supports multi-class classification,whereas SVM needs multiple models for the same.
• Random Forest can give a probability over the prediction, whereas SVM cannot give.
• Random Forest deals categorical data better than SVM.

4.12 SVM vs Naive Bayes :

• Both performs better with low amount of training data and large features.
• If features are mutually dependent, SVM outperforms Naive Bayes.
• SVM is a discriminative model whereas NB is generative model.

5 ALGORITHM THAT WE WILL USE (K-Nearest Neighbors)

We will use K-Nearest Neighbors algorithm because various researches prove it is one of the best algorithms to give good performance and till time along with optimized models organizations uses this algorithm in recommendation systems as support. K-Nearest Neighbour KNN is a machine learning algorithm used for regression, and classification. It is also known as the lazy learner algorithm. It simply uses

a distance-based method to find the K number of similar neighbours to new data and the class in which the majority of neighbours lies, it results in that class as an output. Now let us get our system ready for project implementation.

6 Summery

In this project we are going to developed music genre classification which aims to classify the audio files in certain categories of sound to which they belong. And we have studied different algorithms for music genre classification such as Linear Regression,Naives Bayes,K-Nearest Neighbours,Decision Tree, SVM (Support Vector Machine), Random Forest etc. We compare different algorithms on different parameters and finally conclude that the best and optimal algorithm is K-Nearest Neighbours(KNN).SO that's why we will going to use K-Nearest Neighbours(KNN) algorithm.

7 Conclusion and Future Work

7.1 Conclusion

Upon the completion of Final Year Project we are able to Categories the songs according to there category.we able to understand all the algorithms which can we use for implementation and their functionality .we can understand which one of the algorithm is best among all. We have tried to understand various algorithms that we can use for implementation of music genre and the significant factors about them. This has helped us understand the different parameters that can act as a backbone to estimating the best and optimal algorithms. We have reviewed various research papers to make ourselves aware of the multiple algorithms and methodologies which has given us valuable insights to move ahead with this project and get better results. We have proposed a framework for more accurate music categorization, which we will be implementing shortly.

7.2 Future Work

We have reviewed some work on techniques for music genre and have proposed a framework for accurate categorization of musics. We will study various algorithms and their comparison based on time complexity, data-sets, accuracy and start implementing methods and algorithms after a thorough review.

Acknowledgment

We take this opportunity to express our deep sense of gratitude and are indebted to our Project guide, **DR. BALU L. PARNE**, Assistant Professor in Computer Engineering Department, SVNIT Surat for his valuable guidance, useful feedback and co-operation with kind and encouraging attitude at all stages of this work. We would also like to thank our Head of Department- **DR. RUPA G. MEHTA** , Computer Engineering Department for all the support.

Sourabh Patel
U19CS082

Vatsal
U19CS004

Sarthak
U19CS056

Jyoti
U19CS067