Classification Models for Prediction of Year of Songs

Arshdeep Kaur (V00104104) and Lok In Liang (V00104104)

M. Eng. (Telecommunications and Information Security),
Department of Electrical and Computer Engineering,
University of Victoria

Abstract

In this project, we have chosen to study the problem of classification of songs into their release years based on the audio features of the songs. The dataset used for this study is the subset of Million Song Dataset. The audio features are the timbre average and timber covariance. Various models are tested for this data such as Linear Regression, Logistic Regression, K-nearest neighbour, Naïve Bayes and Artificial Neural Network. These models are compared on various parameters.

Contents

1. **Introduction**

Million Song Dataset was collected by Columbia University and the Echo Nest as a step further in Music Information Retrieval field. It was created with a view that researchers would be able to develop algorithms that could be scaled to commercial size. This large dataset of million songs consists of metadata and audio analysis [1]. Various Music Information Retrieval tasks are recommended such as segmentation of song into segments of chorus or verse, automatic tagging, recognition of artist, categorizing song according to artist, genre, cover recognition and so on [1]. Music Information retrieval techniques help build recommender systems which can recommend songs to users based on genre of song, artist or song rating [2].

We choose the Music Information Retrieval task of categorizing the songs according to their year of release, as it will have practical applications in the recommender system. People tend to have attachment to the music that they have listened during high school or college. Moreover, such a model can also be helpful in understanding how the music has evolved over time [1].

2. **Related works**

As already mentioned above, there are many tasks that can be performed on this Million Song Dataset. But most of all the task of cover song recognition is performed as well as year prediction. Following are some of the works that have been surveyed:

Thierry Bertin-Mahieux *et al*, chose the problem of cover song recognition by using hashed chroma landmarks [3]. The practical application of such techniques can further be extrapolated to finding patterns and structure in music datasets for recommendation and classification. The dataset used is Second Hand Songs dataset which was generated for cover recognition task by matching the dataset from website secondhandsongs.com to million songs dataset. The goal was to achieve sufficient enough results on large-scale dataset which was met.

Another cover song identification was done by Peter Foster *et al*, by using information-theoretic approach to measure similarity between audio signals [4]. They proposed that continuous-valued approach should be used as a means to calculate the pairwise similarity, in place of discrete-valued approach, as former outperforms the latter. Also provides alternative techniques to normalized compression distance such as normalized compression distance with alignment which improves the performance.

Yandi Li *et al*, predicted the musical era of the songs in the Million Song Dataset as a project for the course machine learning at Stanford University [5]. In this project, they have used various models such as Naïve Bayes Classifier, Generalized Linear model, Random Forest and Gradient Boosting Machine. Comparison of these models were made on mean square error and feature selection. The results for all these models were comparable, all the methods chose similar features as important feature for prediction of year.

Another project done by Ruyu Tan *et al*, from University of California, also predicted the release years of the songs using Million Song Dataset [6]. Linear regression, Ridge regression, LASSO regression and Random Forest were the models used for the problem and the metric used to compare these models was mean absolute error. The conclusion of the project was that linear regression model performs better with least mean absolute error.

In the paper by Thierry Bertin-Mahieux *et al.*, also the creators of the Million Song Dataset, describe the creation process and its contents [1]. What are all the possible uses that researchers can make of this dataset are also suggested. Prediction of year is taken as an example of performing one of the tasks on the data. The algorithms used were k-nearest neighbour and Vowpal Wabbit. The results were compared based on average absolute error and square root of average squared error. Vowpal Wabbit outperformed the other algorithms.

Prakhar Mishra *et al*. also used various models such as linear regression, random forest, logistic regression, gradient descent, and gradient boosted trees for predicting the release year of songs in Million Song Dataset [2]. But they used Apache Spark machine learning library to implement these models. This paper proposes the use of Apache Spark framework as it greatly reduces the computational time due to its distributed computing feature.

Although the task of predicting year of a song has been taken up but still no method is provided that would work the best with this kind of data. For our problem we will also use the methods already been tried such as Linear regression, Logistic regression, Naïve Bayes, K-nearest neighbour but also artificial neural networks which proved to be very efficient for this kind of data.

3. **Data**

3.1 Description:

Dataset used for this project is a subset of Million Song Dataset (http://labrosa.ee.columbia.edu/millionsong/), which is a collaboration between LabROSA (Columbia University) and The Echo Nest. The attributes in this subset are just the audio features of the songs, that is, timbre average and timbre covariance, which are continuous values and the class is the release year of the songs that ranges from 1922 to 2011. The total instances are 515,345 and the total number of attributes are 90 where first 12 are timbre average and other 78 are timbre covariance. The train and test split of data is already mentioned along the dataset, that is, 463715 instances should be used for training and remaining 51630 should be used for testing. It is designed such a way so as to counter the "producer effect", by being sure that a song from the artist should not be both in the train and test set [7].

3.2 Data preprocessing:

Although data in the given subset needs no structuring as it has no missing data, also the dataset train/test split is already given. Only normalization of data is required. Normalization process transforms the values of attributes to range between (0,1). The reason for doing this is, so as to give all features an equal chance. Moreover, normalization helps in achieving faster convergence in gradient descent algorithm in neural networks. The figures below show data instances from two classes 'year =1999' as positive and 'year = 2000' as negative for just 5000 samples from the dataset.

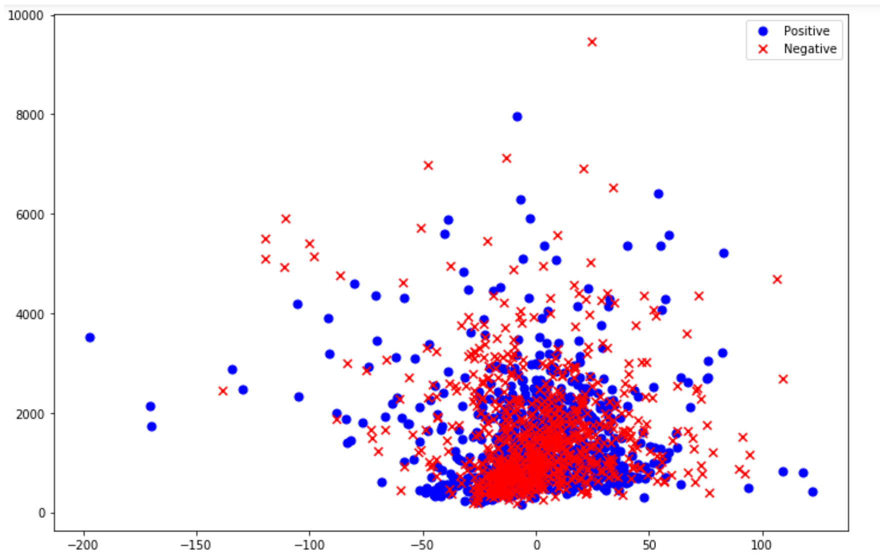


Fig. 1 (a): scatter plot of two classes before normalization

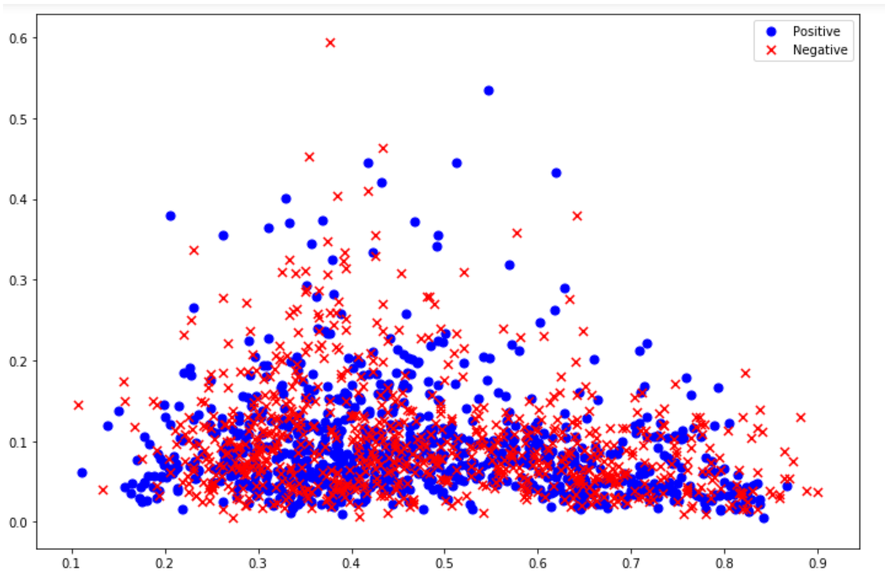


Fig. 1 (b): scatter plot of two classes after normalization

4. **Algorithms Used**

Following are the algorithms used on the subset of Million Song Dataset in order to predict the release year of the song:

4.1 Linear Regression:

This regression technique is mostly used when the attributes and the output are numeric. Output for ith instance is represented as linear combination of weights and attributes:

$$y^{(i)} = \sum_{j=0}^{k} w_j x_j^{(i)}$$

eq. (1)

where, y is the class, x are the attributes, w is the weights and j is the number of attributes. This output is the predicted class, so for linear regression we determine the weights that minimize the sum of squares of differences between actual and predicted outcome over all given instances. The sum of the squares of the differences over all instances is given by:

$$\sum_{i=1}^{n} (y^i - \sum_{j=0}^{k} w_j x_j^i)^2$$

eq. (2)

Gradient descent algorithm is used to optimize the weights by minimizing the equation (2). Linear regression is one of the simplest models used to predict numeric outcomes but it has a shortcoming of being linear. For nonlinear data it predicts the best fitting straight line which may not represent the data well but such simple linear models are foundations for more complex models [8].

4.2 Logistic Regression:

This regression describes the relationship between a dependent binary variable and one or more independent nominal or continuous variables. This type of regression uses transformation function as:

$$S(z) = \frac{1}{e^{-z}}$$

eq. (3)

Gradient descent algorithm is used to minimize the given cost function to obtain the optimized weights:

$$E(w) = \frac{1}{n} \sum_{k=1}^{n} \ln(1 + e^{-y^k w x^k})$$

eq. (4)

A new instance x is classified by using the equation (5), a threshold is selected in range (0,1) such that if the value p(y =1|x) > threshold then new instance is classified into one class and if less then to other class.

$$p(y=1|x) = \frac{1}{1+e^{-wx}}$$
eq. (5)

For multilabel classes two approaches can be used one is one vs all approach in which instances belonging to the particular class are specified as ‘1’ and all others as ‘-1’, if there are k classes it will require k number of models. Other approach that can be used is one vs one approach in which pair of classes are made and model is trained only on the instances belonging to those classes. In both cases, new instance is classified to the class that receives highest votes.

4.3 Naïve Bayes:

This algorithm is basically a classification model and helpful when the relation between the outcome and the attributes in non-deterministic. This classifier estimates the class-conditional probability for a given class ‘y’ by:

$$P(X|Y=y) = \prod_{i=1}^{d} P(X_i|Y=y)$$
eq. (6)

where, d is the number of attributes and X is set of attributes. Such conditional probabilities are calculated for all the classes and new instance is classified to the class with highest probability. Since the data used for this project is continuous, hence conditional probabilities for continuous attributes are given by gaussian distribution, for a given class ‘y’:

$$P(X_i=x_i|Y=y) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}}$$
eq. (7)

where, ‘x_i’ is the i^{th} attribute, μ is the mean of that attribute and sigma is the standard deviation of the attribute [9].

4.4 K-nearest neighbour:

This type of classifier uses lazy learners technique in which modelling of training data is not required until test instance needs to be classified. In this algorithm of nearest neighbour, distance between the test instance and all the training data is computed and the k nearest neighbours are selected and the test instance is classified to the class label based on the maximum neighbours from that class. This algorithm’s efficiency is based on choice of ‘k’ neighbours.

4.5 Artificial neural networks:

Artificial neural networks are inspired from the biological nervous system. It is collection of artificial neurons connected with directed links. This model is made of three layers, that is, input layer, hidden layer and output layer. Hidden layers can be one or more depending upon the requirement of the task. Below is an architecture of ANN:

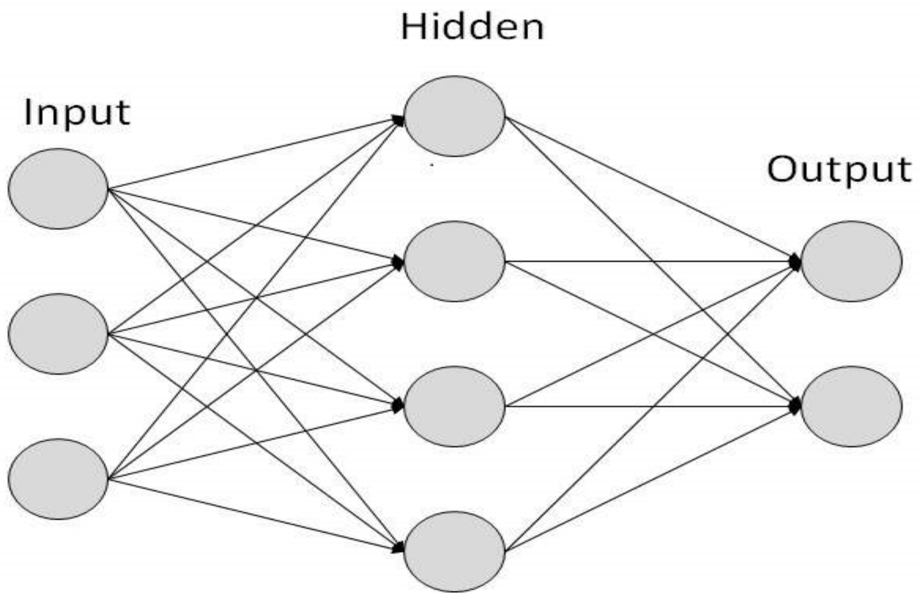


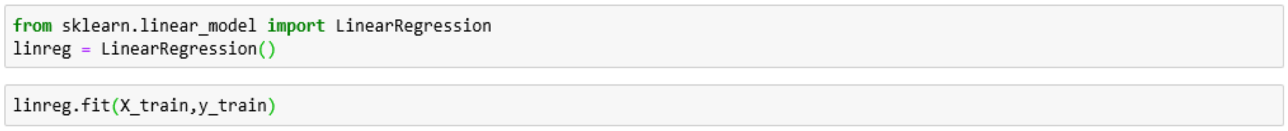
Fig. 2: architecture of Artificial Neural Network [10]

Artificial neurons or nodes in one layer are connected to nodes in the next layer by weights and output of each artificial neuron is computed by activation functions on sum of its inputs. These activation functions are mostly non-linear in nature and hence ANN can deal with more complex data. Gradient descent method is used to learn the weights of hidden and output layers by minimizing the cost function.

5. **Modelling**

All the models are run using scikit-learn, a machine learning library using python programming language [11], except for ANN which is run using Keras, which is a neural network library also written in python [11]. It can be run on top of TensorFlow. Keras is a powerful, easy to use, user-friendly API for us to build a neural network.

5.1 Linear Regression: Since it runs on continuous variables, training classes are also normalized in the range (0,1)

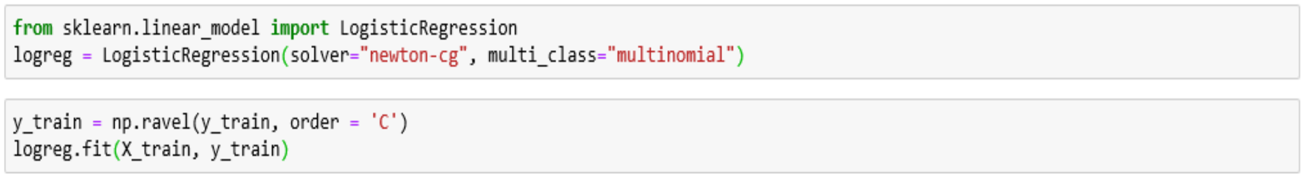


```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
```

```
linreg.fit(X_train,y_train)
```

Fig. 3: Linear regression model

5.2 Logistic Regression: For this model, we have used the solver “newton-cg” and set for multinomial regression since this data has multiple classes.

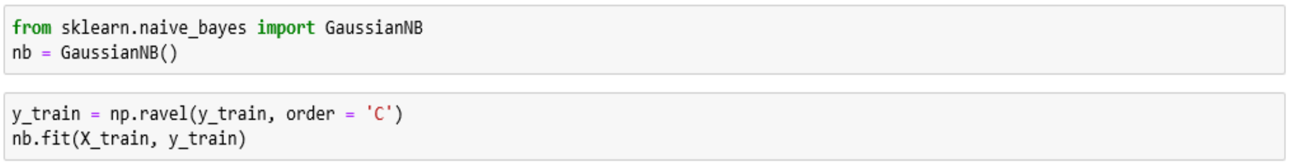


```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(solver="newton-cg", multi_class="multinomial")
```

```
y_train = np.ravel(y_train, order = 'C')
logreg.fit(X_train, y_train)
```

Fig. 4: Logistic regression model

5.3 Naïve Bayes: In this algorithm, we have used GaussianNB, since the values are continuous so gaussian probability density function will be more useful.

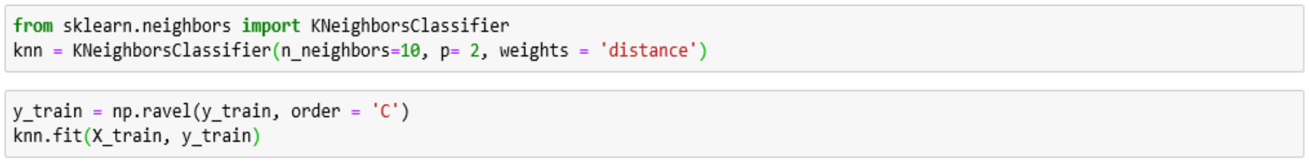


```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
```

```
y_train = np.ravel(y_train, order = 'C')
nb.fit(X_train, y_train)
```

Fig. 5: Naive Bayes model

5.4 K-nearest neighbour: as for this model, we used 10 nearest neighbors for classification and weights of the neighbours are adjusted according to the distance. Euclidean distance is used to measure the distance from the data points.



```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10, p= 2, weights = 'distance')
```

```
y_train = np.ravel(y_train, order = 'C')
knn.fit(X_train, y_train)
```

Fig. 6: K-nearest neighbour model

5.5 Artificial Neural Network: For this model we have used three layers artificial neural network. Since there are 90 attributes in the dataset, there are 90 neurons in the input layer in this network. Also, we decided to use 90 neurons in the hidden and output layer. The class values also need to be converted into vectors of 0’s and 1’s since for this model Keras Deep learning library is used which does not take integer values as output but only binary. Therefore, the output layer must be 90 neurons as each class label is represented by a vector of 0’s and 1’s called as one-hot vector. The activation function for this model is sigmoid function as it is a non-linear function, when z is a large positive number, the output of sigmoid will be 1 and when z is a small negative number, the output of sigmoid will be 0.

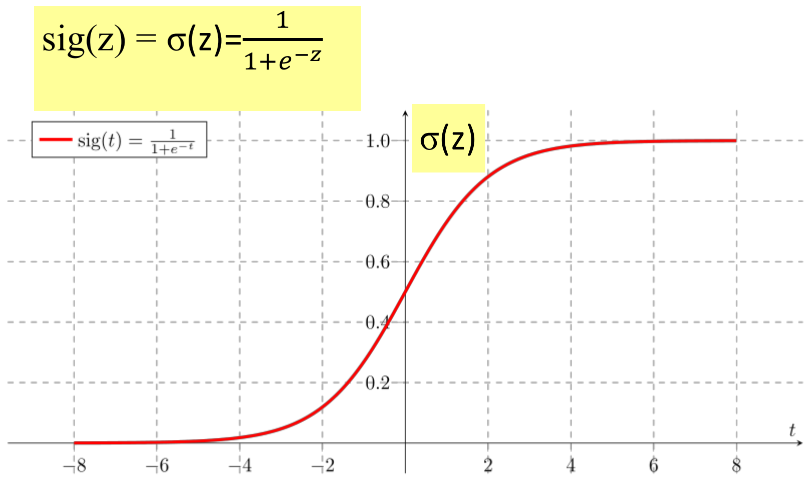


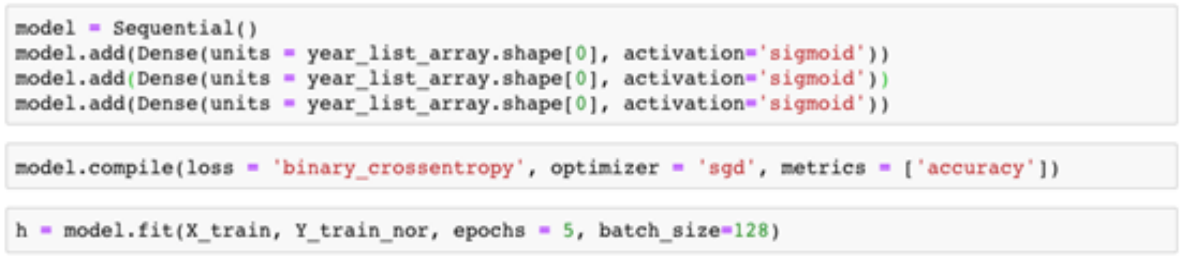
Fig. 7: Sigmoid Function

We have used the cost function binary crossentropy:

$$J(w) = -\frac{1}{N}\sum_{k=1}^{N}[\, y_k \log(\widehat{y_k}) + (1-y_k)\log(1-\widehat{y_k})\,]$$

eq. (8)

Only this cost function works best with this kind of data as the output is in form of 0's and 1's and moreover, the data is overlapping, there are instances, which are alike but categorized into different classes. For such kind of data, this cost function works quite efficiently. We even tried other cost functions but this is the only that gives the best result. SGD optimizer which is called Stochastic gradient descent, also known as incremental gradient descent is used to train the weights of the links. Further we used epochs = 5 and batch size to be 128.



```
model = Sequential()
model.add(Dense(units = year_list_array.shape[0], activation='sigmoid'))
model.add(Dense(units = year_list_array.shape[0], activation='sigmoid'))
model.add(Dense(units = year_list_array.shape[0], activation='sigmoid'))
```

```
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics = ['accuracy'])
```

```
h = model.fit(X_train, Y_train_nor, epochs = 5, batch_size=128)
```

Fig. 8: Artificial neural network model using Keras

6 Results

All the models are compared on the accuracy of their prediction, except for logistic regression as the output of this model is continuous.

Table 1 Comparison based on accuracy

Model	**Accuracy**
Logistic Regression	8.11%
Naïve Bayes	2.51%
K-NN	7.01%
ANN	**98.88%**

The models are also compared based on the mean absolute error (MAE) and mean square error(MSE). Although the output of classifiers is nominal so calculating these errors is difficult so we converted the outputs and the predicted values to continuous values in range (0,1) for this purpose.

Table 2 Comparison based on MAE and MSE

Model	**MAE**	**MSE**
Linear Regression	0.0764	0.0115
Logistic Regression	0.0823	0.0166
Naïve Bayes	0.2394	0.0883
K-NN	0.0863	0.0175
ANN	**0.0267**	**0.0107**

Classifiers can also be compared on bases of precision, recall and F-1 score. Again, linear regression cannot be compared using these measures since it deals with continuous values and also for ANN in Keras model we cannot calculate these values.

Table 3 Comparison based on classification measures

Model	**Precision**	**Recall**	**F1-score**
Logistic Regression	0.07	0.08	0.05
Naïve Bayes	0.07	0.03	0.03
K-NN	0.07	0.07	0.07

7 Conclusion

The results show a clear indication that Artificial Neural Network is the best approach for predicting years from audio features of the song. From Table1, it is clear that ANN outperforms all the models with 98.8 % accuracy. The cost function that works best here is binary crossentropy as it can classify even if different classes have same instances, some songs from different years have same features. Even in Table2, ANN gives the lowest mean absolute error and mean square error. Out of the classifiers given in the Table 3. K-NN gives the highest F1-score. So, there is scope for this classifier as well if some improvements are made such as feature selection. To sum up, ANN algorithm seems to be the step in the right direction for this problem since it can handle redundant features by assigning them really low weights, as songs even from different years can still have similar features.

8 References

1. Thierry Bertin-Mahieux, Daniel P.W. Ellis,Brian Whitman, Paul Lamere. The Million Song Dataset.12th International Society for Music Information Retrieval Conference. 2011.
2. Prakhar Mishra, Ratika Garg, Akshat Kumar, Arpan Gupta and Praveen Kumar. Song year prediction using Apache Spark. Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI). 2016.
3. Thierry Bertin-Mahieux and Daniel P. W Ellis. Large-Scale Cover Song Recognition Using Hashed Chroma Landmarks.IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. 2011.
4. Peter Foster, Simon Dixon, and Anssi Klapuri. Identifying Cover Songs Using Information-Theoretic Measures of Similarity. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 6. 2015.
5. http://cs229.stanford.edu/proj2015/139_report.pdf
6. https://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a028.pdf
7. Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
8. Ian H. Witten, Eibe Frank, Mark A. Hall, and Geoffrey Holmes. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Elsevier Science & Technology.2011
9. Pang Ni Ng Tan, Michael Steinbach, and Vipin Kumar. Introduction Data Mining. Pearson Addison Wesley. 2006.
10. https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks
11. https://www.wikipedia.org/