Recognize genre of a song using Spotify Web API

1 Vijaya Madasu
Matriculation number:11150573
Cologne University of Applied Sciences
vijaya.madasu@smail.th-koeln.de

2 Rubaiya Kabir Pranti
Matriculation number:11146364
Cologne University of Applied Sciences
rubaiya_kabir.pranti@smail.th-koeln.de

3 Roman Schumichin
Matriculation number:11154493
Cologne University of Applied Sciences
roman.schumichin@smail.th-koeln.de

***Abstract*—The aim of this project is to recognize the genres of songs with optimal prediction results by applying several machine learning methods. For this purpose, the data extraction is done initially by using lightweight - Spotipy Python library which is used for Spotify Web API. We investigated various machine learning algorithms including Logistic Regression (LR), K-Nearest Neighbors (K-NN), Decision Tree (DT), Random Forest (RF) and Gradient Boosting (GB) classifiers. After comparing these five classifiers, the GB algorithm shows higher accuracy than rest algorithms. The results indicate that without hyperparameter regularization, models have evident performance decrease in terms of predicting genres of songs. Moreover, the cross-validation is applied to get an estimation of the performance level of our models. Then, after applying optimization techniques along with cross-validation, best hyperparameters are selected and the models are retrained for better predictions. After optimization, the prediction of genres are more accurate than before. In conclusion, this paper shows that Gradient Boosting as well as Random Forest classifiers are both highly effective at classifying genres of songs. And while classifying these genres, to avoid much computational time, only fifteen genres were selected as desired genres. We also kept only important and variant Spotify metadata (genre, popularity etc.) and Spotify audio features (tempo,valence etc.) to characterize our dataset better. Lastly, after analyzing and testing performance for each model, with evaluation metrics such as: accuracy, confusion matrix, precision, recall, f1-score; we also explored K-Means Clustering method to create clusters with matched song genres and we gave customized cluster group names for resultant clusters for better recognition of characteristics of songs for users. For future work, the project will use various neural network models to improve the performance of genre classification results.**

I. Introduction

Music genre classification is one of the most important tasks in music information retrieval studies. There have been many trials to improve the accuracy of this particular task[1]. Being a difficult and highly subjective issue, and in many cases, even humans disagree on which genre a song falls into. In this paper, we focused on predicting genre of songs based on different Spotify audio features such as danceability, valence, tempo, acousticness etc. which are known as data endpoints. Some popular classifiers such as LR, K-NN, DT, RF and GB are applied and tested for their performance to classify music genres based on those data endpoints. The purpose of this research is to find out how successful the recognition of music genres process is which is based on some metadata and features obtained from Spotify Web API.

Based on the results of tests, the GB classifier has the best classification performance with 80.46% accuracy, followed by RF with 79.67%, where both LR as well as DT have accuracy with 70.81% and K-NN with 70.6%. The contributions of this project are as follows:

• We compared five genre classification methods which can assist to recognize genres of songs by providing acceptable classification accuracy using a moderate dataset.

• In order to avoid initial accuracy performance degradation, we used fine tuning model either by individual hyperparameter optimization or by grid search technique to compute the optimum values of hyperparameters.

• We applied smote function for imbalanced training dataset before training our LR and K-NN models.

• Pipelines for LR and K-NN are also implemented with PCA (for correlation reduction among features), standard scalar (for scaling the feature values) and classifiers.

II. Data Retrieval Process

To get access to Spotify music information, we have to set up Spotify Developer account where personal Client ID and Client Secret are retrieved. In Jupyter Notebook, we have worked on a particular OAuth Flow using Spotipy(a Python library) for Spotify API. After installing, importing and setting up Spotipy, data from Spotify Web API is fetched successfully. Thereafter, the data is loaded into a data frame for exploratory data analysis and further machine learning processing where the Python Programming language is used and the Python Data Analysis Library(PANDAS) is used to process data structures and perform data analysis. Besides that, Scikit Learn which is a package containing important modules of machine learning projects is used in this project[2].

III. Methodology

*1) **Spotify Web API and Data extraction:**:* We utilised the Spotify music data set using Spotify's https://developer.spotify.com/documentation/web-api/ website. Spotify music data set has given a total of 150550 songs after first audio feature extraction,then after combining the extracted metadata set and extracted feature dataset, the combined dataset was of total 366684 data points or songs consisting of 126 genres and each has 26 features/columns. As the number of features and the large number of genres will considerably influence the complexity and computation time required for the calculation, thus we decided to keep only 15 genres in which each genre has equal or more than 850 songs.

In this section, the methods used in this project are explained. Fig. 1 shows the stages of the process of supervised learning workflow.

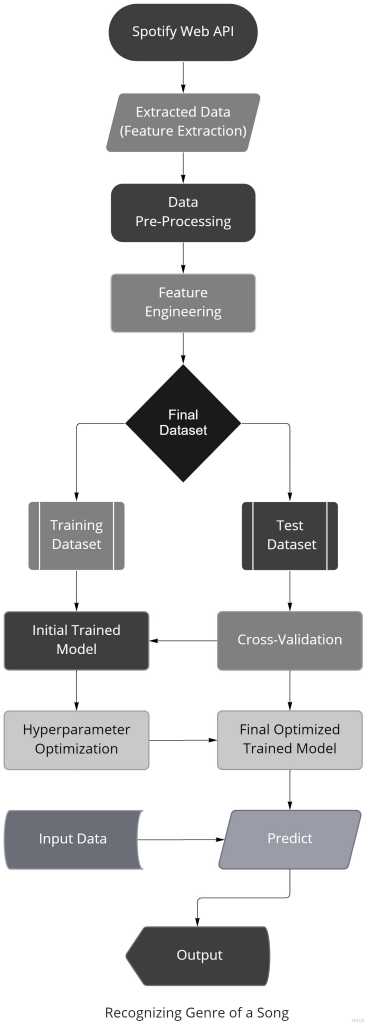


Fig. 1. Supervised Learning Workflow

*2) **Data Pre-processing:**:* As part of the data pre-processing, data cleaning is very important step to be performed. We cleaned the data as follows:

- by deleting rows with missing values(there were missing values)
- by erasing rows/songs containing duplicate data/song(duplicate tracks are dropped down to 80823 tracks/rows from 366684 songs/tracks)
- by replacing booleans with numericals(the 'explicit' feature was converted to numeric values)
- by taking desired genres from 126 to 15 genres such as study, club, comedy, bluegrass, black-metal, world-music, iranian, grindcore, heavy-metal, turkish, afrobeat, forro, country, happy, chicago-house
- after cleaning the data set, 13991 rows/songs are finally taken into consideration for further analysis

*3) **Feature Engineering:**:* The feature engineering process of drawn out data set includes converting feature columns into a set of valuable features.The previous feature extraction contains 26 columns/features named as track_name, artist_name, album_name, track_id, artist_id, type, uri, track_href, analysis_url with the object data type and popularity, duration_ms_x, duration_ms_y, key, mode, time_signature,duration_s with integer data types, as well as acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, and valence with type float data. Lastly, the most significant feature is genre column with categorical data. Therefore, feature engineering is performed as follows:

- by dropping non-numeric or insignificant rows such as track_name, artist_name, album_name, track_id, artist_id, type, uri, track_href, analysis_url etc.
- by using correlation matrix, negatively correlated feature **'loudness'** was dropped as it was affecting performance inversely.
- after feature analysis, 15 features are kept finally including 'genre' feature as labelled column.
- As part of feature engineering, feature scaling is also done within pipelines which were implemented for LR and K-NN classifiers only. Standardization technique was performed as feature scaling.

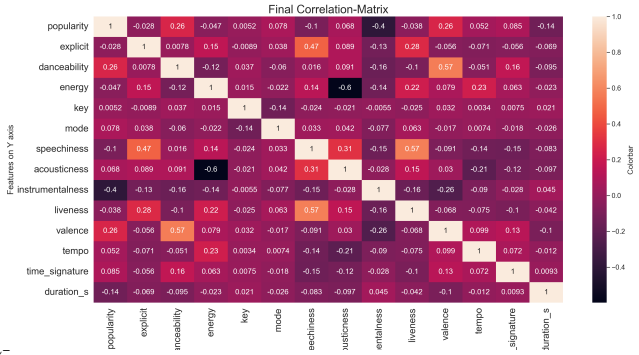


Fig. 2. Correlation Matrix for final dataset

*4) **Splitting Dataset:**:* The dataset was splitted into training and test dataset with a ratio of 80% for training set and 20% for test set with stratification of test dataset. Here, training data is the data we use to train a machine learning model to predict the desired outcome we design our model to predict. And test data is used to measure the performance, such as accuracy or efficiency, of the algorithm we are using to train the machine[3].

*5) **Classification-Training Models:**:* At this stage, the training process is done on training dataset(80%). Then data fitting is executed to fit models to data and then we analyze the accuracy of the fit.

*6) **Evaluation through Cross-Validation:**:* The evaluation criteria of recognizing genres of songs classification project takes accuracy rate as a measure. The accuracy rate is denoted as:
accuracy rate = the number of total correctly classified songs / the number of total songs on test set * 100%.
For better evaluation, cross-validation is performed to get an estimation of the performance of our models through accuracy score function.
Here, instead of using the usual train and test split, k-fold cross-validation on the training data is applied to get our initial and final predictions. This is performed by shuffling the data set randomly, then splitting it into k number of sections (folds)[4].

*7) **Hyperparameter Optimization:**:* Hyperparameter regularization is executed in two ways for our models such as:

- through manual investigation of best possible hyperparameters.
- through Scikit-Learner's GridSearchCV and RandomizedSearchCV[5] which use cross-validation to evaluate all possible combinations of hyperparameter values[4]. In addition to this, in our project, we utilized grid search approach mostly with 3-fold cross validation to tune the parameters for each of the classifiers. Due to time limitations, not all possible parameters were included in the grid search process, which was mainly limited to the amount of trees or n_estimators, the amount of features used or max_features and the maximal depth of a tree or max_depth for DT, RF and GB.

IV. Classifiers

A. Logistic Regression

Logistic Regression is one of the supervised machine learning algorithms which comes under supervised Learning technique. Categorical dependent variable (i.e. genres in our project) is predicted by logistic regression using a given set of independent variables. It is used for solving the classification problems and it has the ability to provide probabilities and classify labelled data using continuous and discrete dataset.

1) Steps performed to get optimized LR model::

- after initial training and evaluating the LR model the accuracy was not good enough
- in this purpose, to overcome the imbalanced dataset, we used SMOTE() function which was conducted outside of our pipeline
- we used StandardScaler() function to scale feature values. For example popularity and tempo have large integer values which needs to be scaled within pipeline
- we included PCA() function also to remove all the features that are correlated among each other within pipeline
- set of several parameters with different range were initiated from which grid search eventually found the best possible combinations of hyperparameters. Grid search optimization also includes cross-validation parameter along with pipeline as estimator.
- after fine tuning model with grid search, the LR model was retrained by providing test set accuracy as 70.81% which was initially 33.87%.

The final hyperparameters selected for LR model and respectively the maximum accuracies that we got with the LR classifier are as follows:

Hyperparameters	Value
C	100
penalty	l2
multiclass	ovr
solver	newton-cg
pipeline	pca, standard scalar, LR classifier

Training Accuracy	71.24%
Test Accuracy	70.81%

B. K-Nearest Neighbor

K-Nearest Neighbor is another supervised machine learning technique that takes a data point and calculates the distance between k number of labelled data points, classifying the point based on the number of votes that it gets from the k-nearest data points. Here, the 'supervised' part means that we already know what genre the song is before we test it, hence the labels[4].

The KNN model is fitted on the training set and evaluated on the test set. This process is then repeated ten times using K-Fold cross validation. If the number of nearest neighbors=k is too small, then it results in the decrease of classification accuracy rate; if the value of k is too large, it is prone to generate more noise data thus decrease the classification accuracy rate[6]. In our project, the experimental results show that initial value of k is 21, where accuracy rate reached up to 44.68%. But after applying smote function, respective pipeline same as LR classification model and applying tuning of model through grid search;the K-NN model gave accuracy of about 70.6%. It is acknowledged that training set has 100% accuracy for overfitting issue although regularization is done. Therefore, we tried other models in next steps to get the best promising model.

The final hyperparameters selected for K-NN model are as follows:

Hyperparameters	Value
leaf size	30
n neighbors	16
weights	distance
metric	minkowski
pipeline	pca, standard scalar, K-NN classifier

The maximum accuracies that we got with the K-NN classifier are:

Training Accuracy	100%
Test Accuracy	70.6%

C. Decision Tree

The Decision Tree Classifier tries to classify incoming songs based on its numerical features. In order to be able to classify it, the Decision Tree needs to be trained by a training set and then it will be evaluated by observation of the accuracy score of its performance on the test set. The goal here is to find the optimal parameters to raise the accuracy of this classifier.

1) Initial Decision Tree: In the very first attempt of a basic Decision Tree without any parameter tuning delivers an accuracy of about 67.02%.

2) Variation of maximum depth: We tried to vary the maximum depth to see how it affects the accuracy score. Here the maximum depth of 13 delivers the best accuracy.

3) Pruning by variation of ccp alpha: The ccp alpha regulates how many nodes can be erased in order to possibly enhance the performance. It turned out, that any attempt of pruning only decreases the accuracy.

4) 'Best' vs. random splitting: Here we observed which splitting we should use for the Decision Tree. There are two options available: 'Best' and random. It is simply the strategy how to split a node. With 'Best' the node will be split in the way which delivers the best local performance measured by entropy or gini-impurity. It's clear that 'best' always delivers the best accuracy.

5) gini vs. entropy: Here we observed which decision criterion should be used. There is the gini-impurity: $G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$ And the entropy: $H_i = -\sum_{\substack{k=1 \\ p_{i,k}0}}^{n} p_{i,k} log_2(p_{i,k})$

It's interesting, that the entropy performs quite a lot better in the training, but the gini performs better for the test set. It might be, that trees grown through entropy criterion tends to overfit in our case, but the difference in accuracy is only small.

6) minimum samples per split: It is possible to adjust the minimum samples to split a node. This should prevent overfitting as there will be a bigger barrier to create high detail splits and they will be less detailed. Of course big values of this parameter also can cause underfitting. The maximum accuracy occurs for 18 samples per split.

7) Conclusion: The hyperparameters we get through manually going through them is this combination:

Hyperparameters	**Value**
min. samples split	18
ccp alpha	0
max. depth	13
splitter	best
criterion	gini

The resulting accuracy on the test set is 70.81%.

8) GridSearch: To find the right parameters it is also possible to try out every possible combination of parameters in a chosen range of values. Those ranges are: min. samples split from 2 to 81, for ccp alpha from 0 to 0.04 in 4 steps and max. depth from 1 to 30.
Because GridSearch is more or less a brute force method, the computational work which needs to be done is quite big. That is why we only evaluated 3 parameters in a small area. The metric used for performance measure is a three fold cross validation. The best parameters according to the used GridSearch are 33 for minimum samples per split, no pruning(ccp alpha is 0) and maximum depth is 13.
With these paramters, we get an accuracy score of 70.03% on the test set. Here we could ask why the accuracy is lower than for the manual search. The answer is, because the grid search uses cross validations which bring different scores than the actual accuracy score on the test set we have chosen.

9) maximum achieved accuracies: The maximum accuracies that we got with the decision tree classifier are:

Training Accuracy	79.32%
Test Accuracy	70.81%

D. Random Forest

Random Forest is an ensemble model for classification which is a method for averaging multiple deep decision tree, aimed on different parts of the same set hoping to reduce the variation. This gives the expense of small increase in bias, loss of accuracy but greatly enhance performance in model. RF are gathering effort within the model to improve performance of a single random tree[7]. In our project, initially the RF model delivers an accuracy of about 79.59% without hyperparameter tuning. Later accuracy increases up to 79.67% for tuning. The RF model was regularized with a set of hyperparameters by the help of grid search. Some of the final parameters used are named as number of estimators, maximum depth, maximum features, minimum samples per split, minimum samples per leaf etc. (https://scikit-learn.org).

While implementing both individual hyperparameter optimization and grid search optimization, the later one always comes up with the best possible combination of hyperparameters as follows:

Hyperparameters	**Value**
max. depth	19
max. features	3
n estimators	1000
criterion	entropy
min. samples split	3
min. samples leaf	2
class weight	balanced_subsample

The maximum accuracies that we got with the RF classifier are:

Training Accuracy	99.46%
Testing Accuracy	79.67%

E. Gradient Boosting

At the very end of our project, we have implemented gradient boosting algorithm to predict the genres of songs, which is a kind of integrated learning method. This approach uses multi-classifiers, which create hundreds of trees. Therefore,the design of each classifier is simple and speed up the progress of our training. At the same time, the concentration of multiple classifiers can achieve a better accuracy[8]. To find the optimal parameters for GB we implemented following parameters(https://scikit-learn.org/) are as follows for which we get an accuracy score of 80.46% on the test set which was initially 78.67% without hyperparameter tuning.

1) number of estimators: The n_estimators parameter denotes the number of trees of the model. From output, number of trees or n_estimators=500 is achieved which is the optimal hyperparameter to get better accuracy which means that 500 different decision trees will be constructed in the Gradient Boosting model.

2) maximum depth: The max_depth parameter specifies the maximum depth of each tree. From execution result, max_depth=8 is perceived as the best possible hyperparameter which defines that each tree will expand until every leaf is pure. A pure leaf is one where all of the data on the leaf comes from the same class.

3) maximum features: The best maximum features value is attained from optimization too which is applied for individual tree. The result conveys that max_features=2 is best hyperparameter which provides better accuracy score.

4) minimum samples per split: The min_samples_split parameter specifies the minimum number of samples required to split an internal leaf node. The result for minimum samples per split or min_samples_split=7 which is optimal hyperparameter which means that an internal node must have at least 7 samples before it can be split to have a more specific classification.

5) minimum samples per leaf: The min_samples_leaf parameter specifies the minimum number of samples required to be at one leaf node. The maximum accuracy occurs for when minimum samples per leaf or min_samples_leaf is equal to 15 which is the best hyperparameter which means that every leaf must have at least 15 sample that it classifies.

To conclude, between individual hyperparameter optimization and grid search optimization, the later one provides best possible combination of hyperparameters for GB Model also.

The maximum accuracies that we got with the GB classifier are:

V. Additional Work

A. K-Means Clustering

In our case it might be pretty interesting if there are major clusters we can divide the genres into. In the following our attempt of K-Means Clustering will be illustrated.

Hyperparameters	**Value**
max. depth	8
max. features	2
learning rate	0.1
n estimators	500
subsample	1.0
min. samples split	7
min. samples leaf	15
criterion	friedman_mse

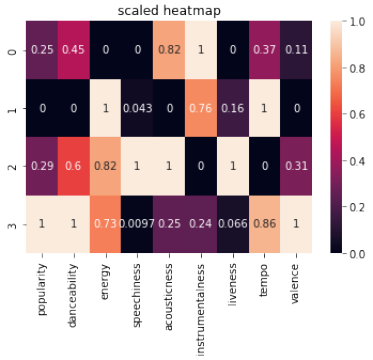
Training Accuracy	100%
Testing Accuracy	80.46%

1) Features used: The used features are the ones which should describe the character of the song in particular. The features we didn't include are explicit and the duration in seconds.
Included features: ['popularity', 'danceability', 'energy', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'tempo', 'valence'].

2) Scaling: As features like 'popularity' [from 0 to 100] are significantly higher in value than those like 'danceability' [0 to 1] the feature values need to be scaled. In our case we use the standard scaler.

3) Finding the right amount of maximum clusters: In order to find the right amount of clusters we should take a look into the inertia and the sillhouette scores. We should use the value, where we can see an elbow. In this case it is 4. The sillhouette score plot shows, that we also should use 4 clusters.

4) Means plot of resulting clusters: We can visualize the means of each cluster for each observed feature. To evaluate this a little bit easier, we can scale this table by setting up the minimum value as 0 and the maximum as 1 for each single feature.



5) Which genre belongs to which cluster?: In order to see which genre fits to which cluster we can look at the ratio between the samples labeled as the observed cluster and the observed genre and the overall existing samples for a genre. This leads to the following lists of genres for each cluster:

- Cluster 0(Acoustic Instruments): bluegrass (50%), iranian (50%), study, world-music (33%)
- Cluster 1(energetic fast Instrumentals): black-metal, club

(50%), grindcore, happy, heavy-metal, iranian (50%), world-music (33%)
- Cluster 2(speech live acoustic): only comedy
- Cluster 3(Pop-Dance): afrobeat, bluegrass (50%), chicago-house, club (50%), country, forro, turkish, world-music (33%)

Some of the genres like 'world-music', 'club', 'iranian' and 'bluegrass' do not belong clearly to one of the clusters but are pretty equally distributed between two or three of them. This may be because those genres have both described characteristics and one half might be for example a little bit more likely to acoustic-instruments cluster and some more to fast energetic instrumentals like the 'iranian' genre. Here might more clusters lead to more precise characterized clusters, which describe the contained genres better.

VI. Evaluation(Prediction)

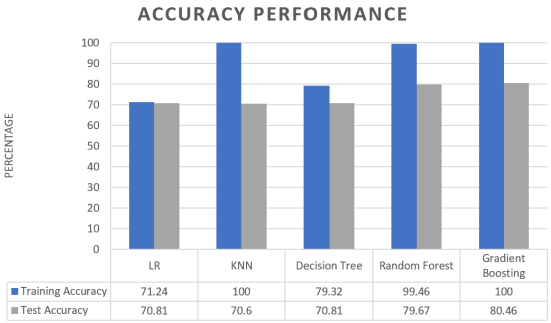
A. Accuracy

The overall maximum accuracies of every classifier are summarized in the following table:

Model	Training Accuracy	Testing Accuracy
Logistic Regression	71.24%	70.81%
KNN	100%	70.6%
Decision Tree	79.32%	70.81%
Random Forest	99.46%	79.67%
Gradient Boosting	100%	80.46%

B. Comparison graph

In the plot below, it is visible that both highest performing classifiers are respectively GB with an accuracy of 80.46% and RF with 79.67%. The lowest accuracy was achieved by the K-NN classifier with 70.6%. Besides, DT and LR had the same accuracy about 70.81% which is only a small amount of higher than K-NN.



VII. Conclusion

In this paper, we propose five machine learning classifiers' comparison of performance to recognize genres of songs successfully based on Spotify audio features, where both GB and RF classifier proved to be superior in terms of accuracy compared to DT, K-NN and LR models. In addition, the training set results of K-NN and GB indicate that these two models are overfitting on training set even after doing optimal hyperparameter regularization. However, it is comprehensive that hyperparameter tuning makes every model perform better than their initial states. As expected, comparatively GB and RF models are more accurate at recognizing genres of songs which is our primary goal to achieve in this project. In the future, we can apply CNN and RNN model (LSTM) to improve the performance of this genre classification results in a more profound way.

Future Work

As the whole initially fetched dataset from Spotify contains about 126 different genres, it might be helpful to take a look into subset of genres also which might be easier to evaluate these ML algorithms. In order to do this, we can try to create clusters of this big initial dataset and take a subset of genres which are easier to separate from each other. Furtherore, for future work, we like to analyse music genre classification based on metadata fetched from raw audio wavedata. It is practised and known that, classification based on metadata feature is relatively quicker than the extracted audio features from source like Spotify. For next work, it is intended to differentiate the performance of music genre classification based on metadata features (i.e. energy function, the spectrum, the cepstral coefficients, the fundamental frequency related features) and extracted audio features altogether, both in terms of accuracy and computational time.

References

[1] T. Giannakopoulos, A. Pikrakis, Introduction to Audio Analysis, Academic Press, 2014.

[2] D. R. Ignatius Moses Setiadi et al., "Comparison of SVM, KNN, and NB Classifier for Genre Music Classification based on Metadata," 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), 2020.

[3] J. P. Lander, M. Beigelmacher,"The Essential Guide to Quality Training Data for Machine Learning". cloudfactory.com. https://www.cloudfactory.com/training-data-guide (accessed June 13,2022).

[4] N. Thomas, "Using k-nearest neighbours to predict the genre of Spotify tracks".towardsdatascience.com.https://towardsdatascience.com/using-k-nearest-neighbours-to-predict-the-genre-of-spotify-tracks-796bbbad619f (accessed June 13,2022).

[5] Geron, Aurelien. Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. Sebastopol, CA: O'Reilly Media, 2019.

[6] M. Wu and X. Liu, "A Double Weighted KNN Algorithm and Its Application in the Music Genre Classification," 2019 6th International Conference on Dependable Systems and Their Applications (DSA), 2020.

[7] C. Chen and X. Steven, "Combined Transfer and Active Learning for High Accuracy Music Genre Classification Method," 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2021.

[8] S. Lu, B. Wang, H. Wang and Q. Hong, "A Hybrid Collaborative Filtering Algorithm Based on KNN and Gradient Boosting," 2018 13th International Conference on Computer Science Education (ICCSE), 2018.