College of Engineering

COMP 491 – Computer Engineering Design Project

Final Report

ANIMAL SOUND CLASSIFIER

Participant information:

I.Erdem Demir - 0053625

Project Advisor
Barış Akgün

03.01.2020

TABLE OF CONTENTS

I. ABSTRACT II

II. INTRODUCTION 1

III. SYSTEM DESIGN 2

A. MAIN PROGRAM 3

B. MFCC AUDIO FEATURE EXTRACTION AND PRE-PROCESS 3

C. MACHINE LEARNING PROCESS 4

1. RANDOM FOREST CLASSIFICATION 4

2. SVM (SUPPORT VECTOR MACHINE) CLASSIFICATION 5

3. K-MEANS CLUSTERING 6

4. DBSCAN (DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE) CLUSTERING 7

D. RESULTS 8

E. GUI 8

IV. ANALYSIS AND RESULTS 8

V. CONCLUSION 14

VI. REFERENCES 14

VII. APPENDIX 15

I. ABSTRACT

Animal Sound Classifier is a program created to help researchers with limited coding skills to classify the animal sounds they are working with, since it would take more amount of time for them to classify the sounds by listening to them one by one. The program aims to classify or cluster the input data using 4 different algorithms: Random Forest, SVM, K-Means and DBSCAN. Additionally, it is aimed to play the sounds and visualize their spectrograms. A user-friendly GUI and possibility to use different algorithms differentiates the project from the other similar programs. For the goals, the audio MFCC features are extracted and the machine learning algorithms are run. The goals are achieved with the exception of being able to get an input data to use.

II. INTRODUCTION

The aim of the design project is to design an audio classification tool that can classify the audio sets containing animal sounds. It takes an audio set as input and classifies it if the set is labeled and clusters if it is not labeled. Additionally, the program can play and plot the audio. It is created for researchers with no programming knowledge who work with animals and need to differentiate a high number of different animals' sounds.

The program does the classification and clustering via extraction of MFCC (Mel Frequency Cepstral Coefficients) features and the results are returned as heat maps for classification algorithms and visualized as 2D centroids for clustering. Also, the accuracy information is displayed.

There are several programs dedicated for classification, especially text classification being very effective.[1] However, because of the file size, audio classification could not improve much.

Most audio classification projects classify human voices or music, and environmental sound classification is not developed as much.[2] Existing animal sound classifications use sequential classifiers [3], deep convolutional networks [4], or could be used for safari rides instead of research environments [5]. However, none of these examples use clustering, nor do they use 4 different methods, 2 of them available at the same time depending on whether it is classification or clustering. Sequential classifier trials include 5 different classification, but this paper does not offer a program that is able to classify in 5 different ways. Deep convolutional networks require many steps for the user to go through and the last example only uses Random Forest. This Animal Sound Classifier project provides a software that can use Random Forest and/or SVM classification algorithms, and K-Means and/or DBSCAN clustering algorithms while keeping the number of steps for the user as minimal as possible via a user-friendly UI.

III. SYSTEM DESIGN

The program works as follows:

The user uploads the input data set to the program, chooses whether it is a labeled or unlabeled set. In the label options screen, the user can move to the info screen to choose an audio file from the set to play and/or visualize the spectrogram. If the labeled option is chosen and Next button is clicked, the user can choose to implement Random Forest, SVM or both. For random forest, the parameter number of estimators is required, and for SVM, c and kernel parameters are required. These values can be optimized if not known. 5-fold cross validation can also be used for both methods. If the unlabeled option is chosen, the user can implement the clustering methods K-Means, DBSCAN or both. For K-Means, number of clusters, and for DBSCAN, epsilon and minimum samples should be entered. These parameters can also be optimized. After the algorithm(s) are chosen, the results are visualized, and the accuracy data is printed on the result screen. A summary of the design is found in Figure 1.

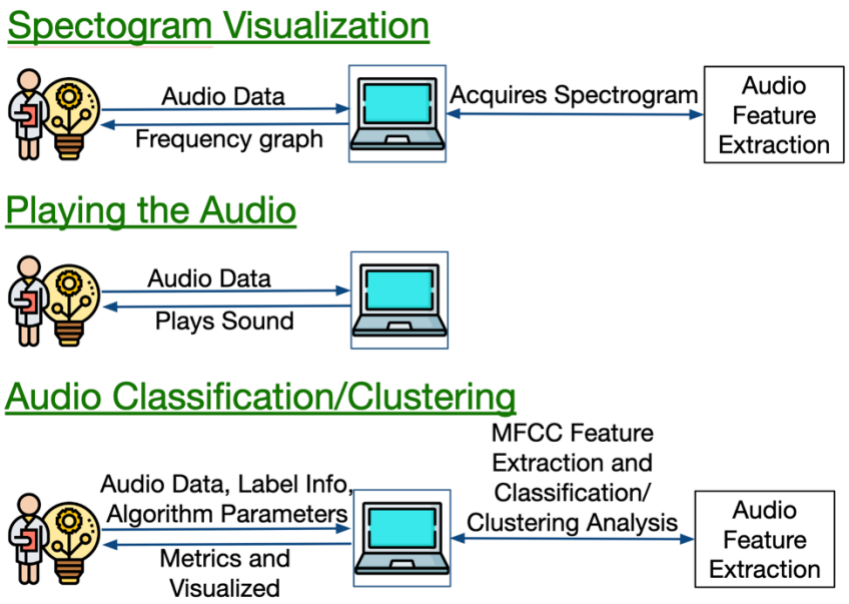

Figure 1 Project Design

A. Main Program

Animal sound classifier project proposed to deliver a classifier integrated with user-friendly GUI in which allows users to implement machine learning algorithms without elaborating on codes and only selecting input parameters with respect to the either classification or clustering algorithms and depending on preferences. Furthermore, for the research and more detailed visual and vocal analysis, there is "Details" page present and researchers may also prefer to compare those audio files with human skills. In the details page, depending on the input dataset page can display labeled dataset features: labeled bird sounds (15 different breed and 637 different songs) or can display unlabeled dataset features: mixture of unlabeled cat-dog sounds (280 data points). The details page allows users to play the animal sound or plot the Hz spectrogram which gives opportunity for researchers to compare animal sounds visually and vocally in one place. In this page, to play an animal sound the Python libraries of PyAudio and Wave are used. Additionally, Python libraries of mathplotlib.pyplot and wave libraries are used to plot frequency spectrograms.

B. MFCC Audio Feature Extraction and Pre-process

After observing the selected labeled or unlabeled audio dataset in detail, machine learning process initiates with audio feature extraction and the pre-process stages.

On audio feature extraction part, after importing audio files, MFCC (Mel Frequency Cepstral Coefficient) features of audio files extracted on lists thus created a 637 x 20 matrix (x 20 represents the extracted 20 different MFCC features of the input file) in which will be used as train dataset on Machine Learning process. While extracting MFCC features of each audio file, sample rate is set to 44.1 kHz and by using Python library of librosa.feature.mfcc, MFCC features of audio files extracted into the feature matrix. Audio feature extraction process is followed by pre-process in order to create train and test dataset from the created MFCC feature matrix. Train and test set divided as 70% train – 30% test (labeled: 446-train, 191-test; unlabeled: 196-train, 84-test) by using train_test_split method imported from Python library of scikit-

learn. Pre-process stage is completed and ready to implement preferred Machine Learning algorithms.

C. Machine Learning Process

The main reason of selecting those algorithms to execute classification and clustering on input-audio data is to try one of the most popular algorithms using all around the world to increase model accuracies and decrease error rates. Specifically, after completing a thorough literature review, it is observed that, similar audio classification projects for animal sounds are using Random Forest (RF) classifier because of the reason RF algorithm intrinsically suited for multiclass problems and it works well with a mixture of numerical and categorical features. On the other hand, for classification, SVM is strongly required due to the reason of there are several cases of SVM performs better than RF classifier and RF classifier also needs to be compared with different classifiers to see the other performance of the other classification models.

For the Machine Learning implementation of classification on labeled dataset, there are two different algorithms:

1. Random Forest Classification

By taking the input parameters from user-friendly GUI and train-test dataset from pre-process stage, and RandomForestClassifier from scikit-learn library is imported, classification process starts. If input parameter of n_estimators of the RandomForestClassifier is set by the user on the GUI, using the selected input, model is created. In addition to this, if the researcher has no information about the RF algorithm, he/she can select the is_optimized option to automatically set the n estimators parameter which maximizes the accuracy of the model.

After taking input, the classification model is created, and train dataset is fit on model then prediction is made on the test set. In order to calculate the accuracy of the model's prediction on test set accuracy_score method of scikit-learn library is used by taking y_test labels and the prediction results of the model as inputs. Moreover, to visually see the model's performance, Confusion Matrix is created and plotted as a

heatmap using the mathplotlib.pylot and seaborn libraries. Besides that, there is also an optional 5-Fold-Cross-Validation option present on the GUI. In case of selection of 5-Fold-Cross-Validation option, the model is validated by using 20% - 40% ... etc. of train dataset and the accuracy of each model on test dataset is calculated. The accuracy average of the validated model gives the actual accuracy of the Random Forest Classification.

Last but not least, in order to see the performance of the model and to compare models, RMSE(Root Mean Square Error) and MAE(Mean Absolute Error) are calculated using the scikit-learn.metrics library.

2. SVM (Support Vector Machine) Classification

By taking the input parameters from user-friendly GUI and train-test dataset from pre-process stage, and SVM from scikit-learn library is imported, classification process starts. If input parameters of kernel and c of the SVM is set by the user on the GUI, using the selected inputs, model is created. In addition to this, if the researcher has no information about the SVM algorithm, he/she can select the is_optimized option to automatically set the c and kernel(linear kernel works better) parameters which maximizes the accuracy of the model.

After taking input, the classification model is created, and train dataset is fit on model then prediction is made on the test set. In order to calculate the accuracy of the model's prediction on test set accuracy_score method of scikit-learn library is used by taking y_test labels and the prediction results of the model as inputs. Moreover, to visually see the model's performance, Confusion Matrix is created and plotted as a heatmap using the mathplotlib.pylot and seaborn libraries. Besides that, there is also an optional 5-Fold-Cross-Validation option present on the GUI. In case of selection of 5-Fold-Cross-Validation option, the model is validated by using 20% - 40% ... etc. of train dataset and the accuracy of each model on test dataset is calculated. The accuracy average of the validated model gives the actual accuracy of the SVM Classification.

Last but not least, in order to see the performance of the model and to compare models, RMSE(Root Mean Square Error) and MAE(Mean Absolute Error) are calculated using the scikit-learn.metrics library.

While working on unlabeled data, the main reason for using those clustering algorithms is besides trying most commonly used algorithms, also try those clustering algorithms for different purposes. For example, K-Means algorithm follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). K-Means algorithm will be used for well-recorded audio inputs with clean backgrounds. However, the main reason to use DBSCAN algorithm is to use for relatively unclear audio inputs to overcome the noise issue and focus on clustering depending on the densities.

For the implementation of clustering on unlabeled dataset, there are two different algorithms:

3. K-Means Clustering

K-Means clustering includes mini-pre-preprocess stage of dimensionality reduction for the MFCC feature matrix.

MFCC feature matrix has shape of 280 x 20 and 280 x 2 matrix is required for plotting purposes. Before initializing to dimensionality reduction process, input unlabeled MFCC matrix is put into Standard form then output normalized by using the scikit-learn.preprocess library's StandardScaler and normalize methods. Following the normalization process of 280 x 20 MFCC feature matrix, dimensionality reduction method of T-SNE is applied on the normalized matrix to derive 280 x 2 MFCC feature matrix. T-SNE method is imported using the sklearn.manifold library and thus 280 x 2 MFCC feature matrix is created. The output matrix gets ready for K-Means Clustering implementation. After taking input parameter of n_clusters of the K-Means is set by the user on the GUI, using the selected inputs, model is created. In addition to this, if the researcher has no information about the K-Means algorithm, he/she can select the is_optimized option to automatically set the n_clusters parameter. The selection of this parameter is done by using commonly preferred special parameter selection method of "Elbow Method". In this method, K-Means algorithm runs multiple times over a loop, with an increasing number of cluster choice and then plotting a clustering score as a function of the number of clusters.[6]

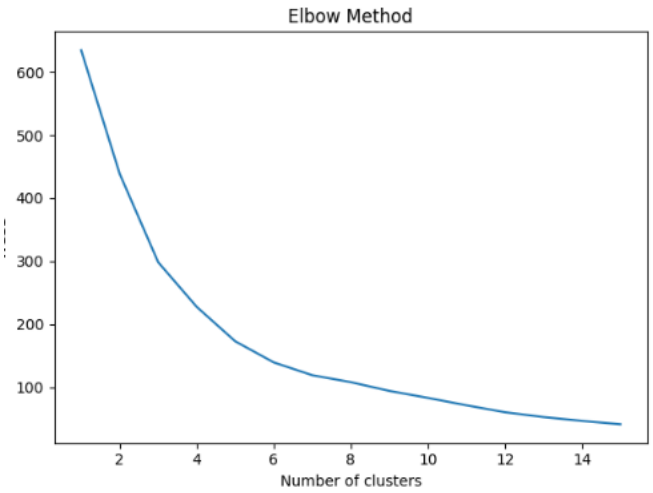

Figure 2 Optimization of Number of Cluster Parameter via Elbow Method

As it can be seen on Figure 2, the curve gets smoothing by the time n_clusters arrives to 12, therefore for the optimized parameter selection n_clusters should be selected as 12.

After taking input, the clustering model is created, 280 x 2 MFCC feature matrix is fit on the model created. Then, centroids of the datapoints calculated by the model. In order to visually see the centroids created on the normalized datapoints, both data and centroids plotted by using the mathplotlib.pyplot library.

Last but not least, in order to see the performance of the model and to compare clustering models, Silhouette coefficient is calculated, which ranges between [-1,1] and 1 is the perfectly clustered model's value.

4. DBSCAN (Density-based spatial clustering of applications with noise) Clustering

DBSCAN clustering includes mini-pre-preprocess stage of dimensionality reduction for the MFCC feature matrix.

MFCC feature matrix has shape of 280 x 20 and 280 x 2 matrix is required for plotting purposes. Before initializing to dimensionality reduction process, input unlabeled MFCC matrix is put into Standard form then output normalized by using the scikit-learn.preprocess library's StandardScaler and normalize methods. Following the normalization process of 280 x 20 MFCC feature matrix, dimensionality reduction method of T-SNE is applied on the normalized matrix to

derive 280 x 2 MFCC feature matrix. T-SNE method is imported using the sklearn.manifold library and thus 280 x 2 MFCC feature matrix is created. The output matrix gets ready for DBSCAN Clustering implementation. After taking input parameters of eps and min_samples of the DBSCAN is set by the user on the GUI, using the selected inputs, model is created. In addition to this, if the researcher has no information about the DBSCAN algorithm, he/she can select the is_optimized option to automatically set the eps and min_samples parameters. Optimized parameters are set by iteratively running DBSCAN algorithm and calculating silhouette coefficient until reaching the highest value of it.

After taking input, the clustering model is created, 280 x 2 MFCC feature matrix is fit on the model created. Then, clusters are of the datapoints calculated by the DBSCAN model. In order to visually see the clusters predicted by the DBSCAN model on the normalized datapoints, both data and centroids plotted by using the mathplotlib.pyplot library.

Last but not least, in order to see the performance of the model and to compare clustering models, Silhouette coefficient is calculated, which ranges between [-1,1] and 1 is the perfectly clustered model's value.

D. Results

After completing machine learning process, visual outputs and metrics are derived. By using visual results and metrics we can compare the performance of the models that are created and also be displayed by the researcher on GUI.

E. GUI

Kivy library is used for the implementation of the user interface. Screens are connected via screen manager. File chooser uploads the input data and recycleview is used to list the audio files.

IV. ANALYSIS AND RESULTS

A sample trial is run, preexisting audio datasets are used.

In Figure 3, the label screen is shown. For the first trial, labeled data is assumed and is chosen.

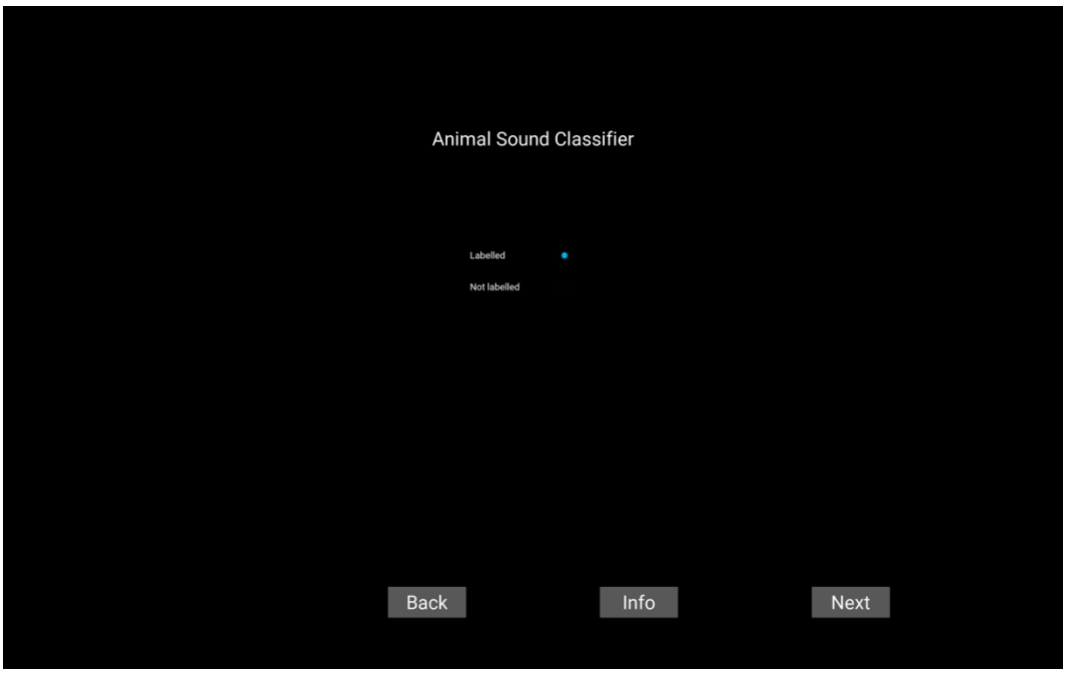

Figure 3 Label Screen

In Figure 4, the info page is visible. Here the spectrogram of an audio is visualized, and the sound is played.

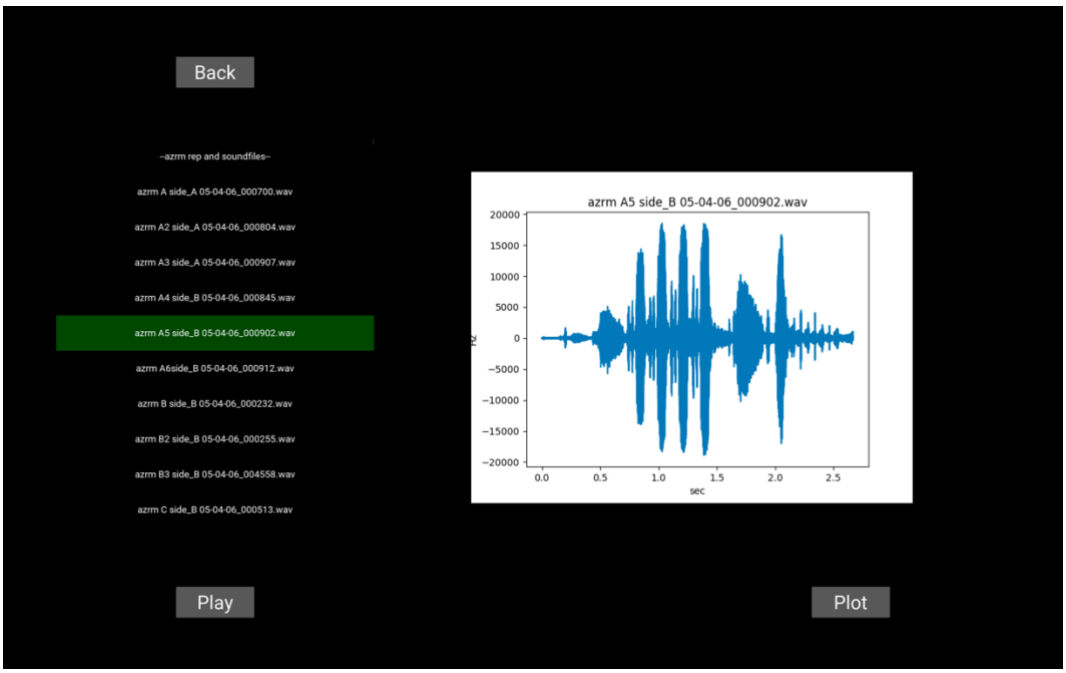

Figure 4 Info Screen

After looking at the details, the program is returned back to the label screen and the next button is clicked to show the classification algorithms screen. Both random forest and SVM are chosen and optimized, assuming the parameters are not known. 5-Fold Cross Validation is chosen for both methods. In Figure 5, the screen is shown.

Figure 5 Classification Algorithms Screen

Two classification algorithms take approximately 1 minute. In the results screens (Fig. 6 and 7), the results are visible. Accuracy of Random Forest is 68%, with 5-Fold Cross Validation the mean is 71.71%. Accuracy of SVM is 64%, with 5-Fold Cross Validation the mean is %70.12.

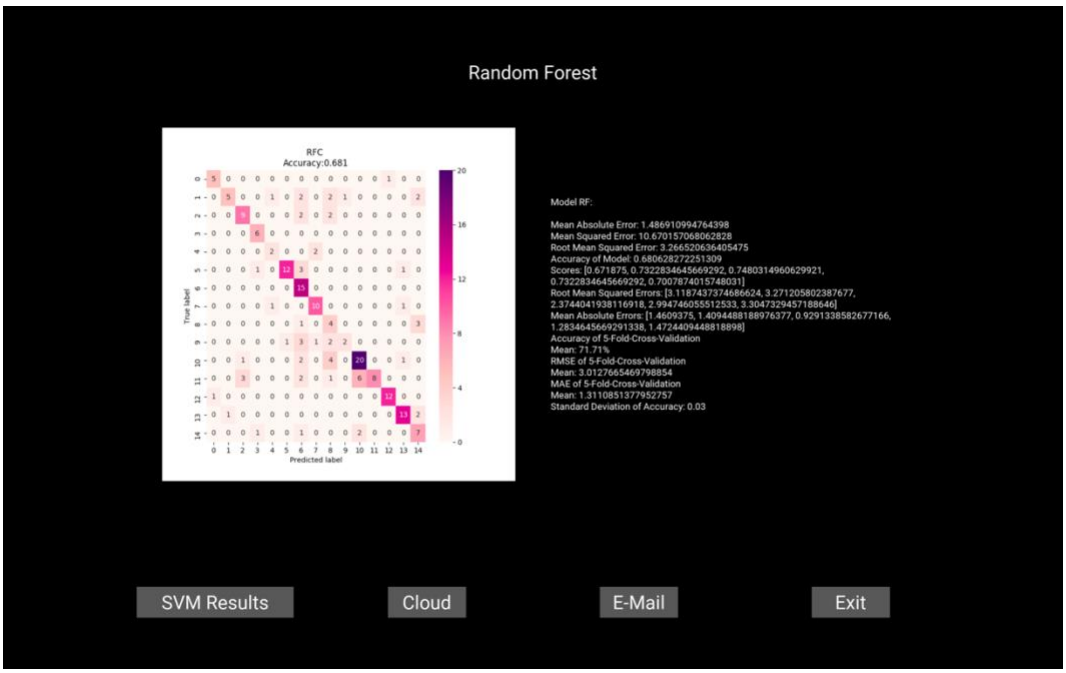

Figure 6 Random Forest Results

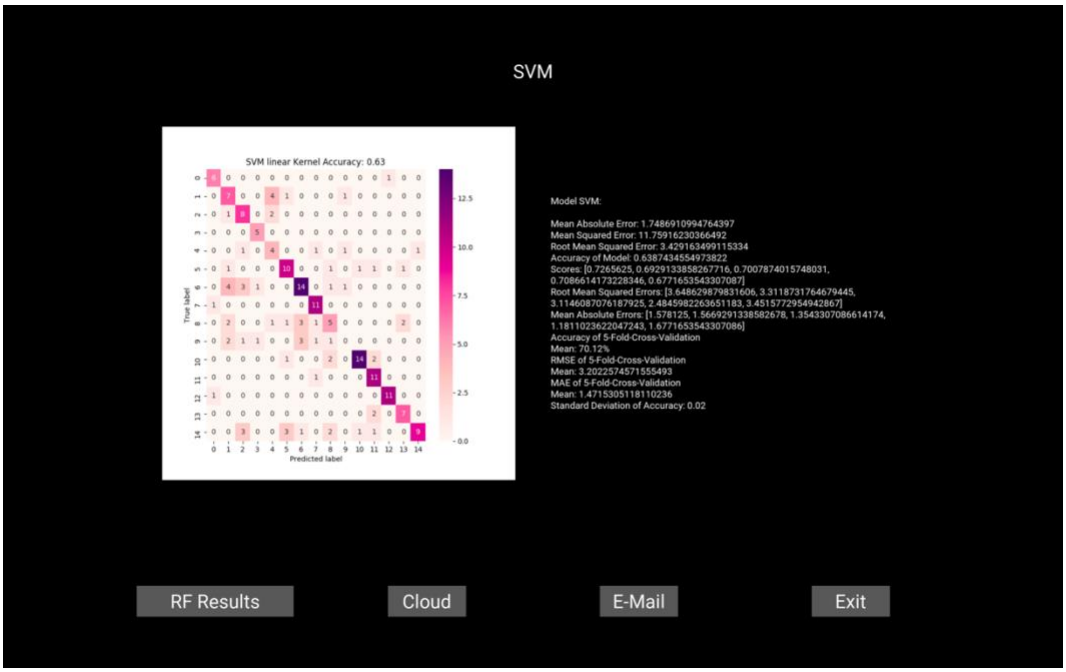

Figure 7 SVM Results

For the second trial, unlabeled data is chosen. On the clustering algorithms screen shown in Figure 8, K-Means algorithm is chosen with the number of clusters as 11, and DBSCAN algorithm is optimized.

Figure 8 Clustering Algorithms Screen

In Figure 9 and 10, the results for K-Means and DBSCAN and shown. The silhouette coefficient for K-Means is 0.52, while the silhouette coefficient for DBSCAN is 0.45.

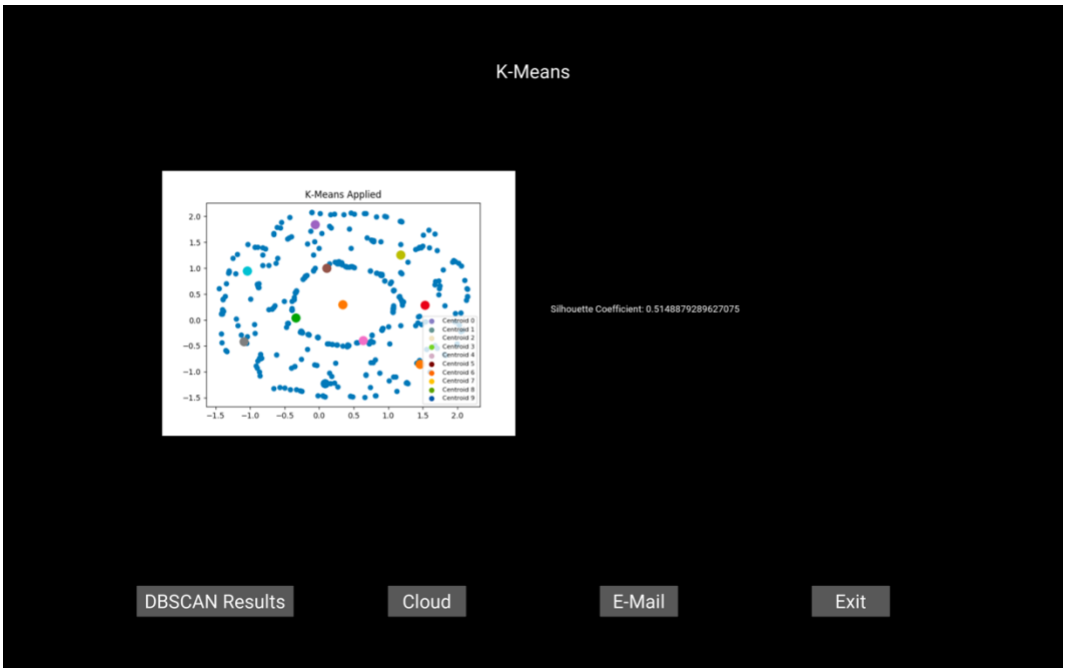

Figure 9 K-Means Result

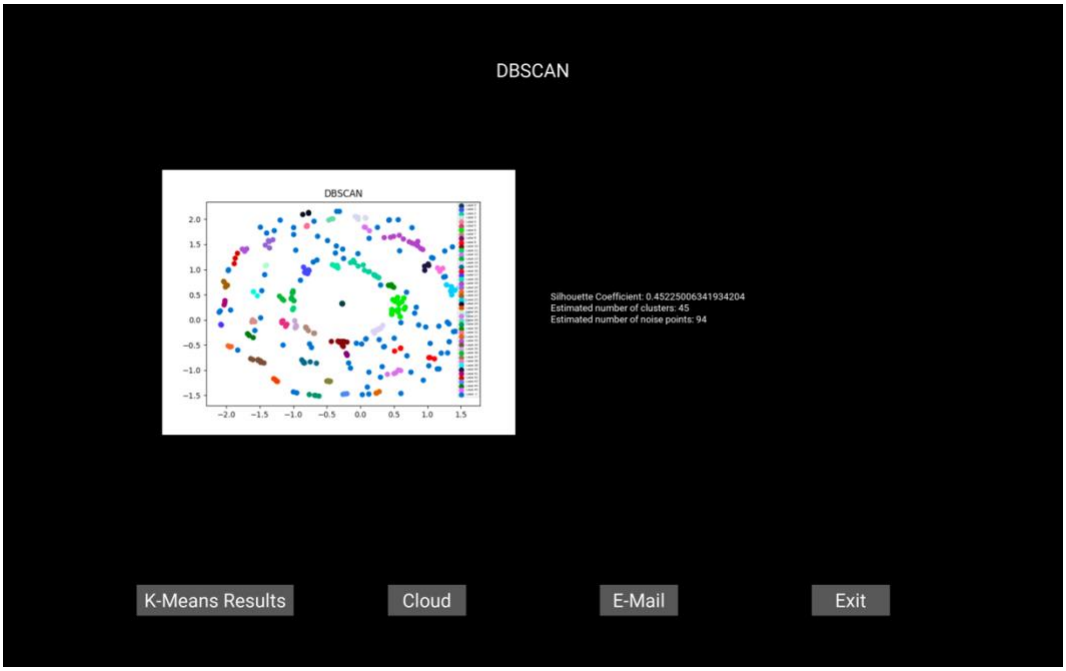

Figure 10 DBSCAN Result

The program is tried and works properly except getting the audio input, for this reason, preexisting audio sets are imported. GUI adds the location of the files in a list, but it is not used in the program because the recycleview could not be updated once the files are added. In addition to this, the conditions for the parameters (for example, for K-Means the number of clusters has to be between 1 and 15) are not set up in the program, so it can take other numbers and yield error.

Finally, since the third member is no longer in the group, the network part including the program functioning on cloud, option to upload the results to Cloud and saving the results to a database are not implemented.

The results for the optimized models are as follows:

Classification Results and Metrics:

- RF Model accuracy: 71~75%, RMSE: 3.0084, MAE:1.32
- SVM Model accuracy: 68~74%, RMSE: 3.062, MAE: 1.43

Clustering Metrics:

- DBSCAN metrics: Silhouette coefficient: 0.4626
- K-Means metrics: Silhouette coefficient: 0.5027

V. CONCLUSION

A project that can be used by researchers has been successfully developed. Extraction of MFCC features is implemented and used for the implementation of four different machine learning algorithms, Random Forest and SVM for classification, and K-Means and DBSCAN for clustering. The extraction is also used for the spectrogram visualization. In addition to this, audio can be played in the program. Initial goal of running the program on Cloud and being able to choose data features could not be implemented. Optimized results return 71-75% accuracy for Random Forest, 68-74% accuracy for SVM, 0.4626 as the silhouette coefficient for DBSCAN and 0.5027 as the silhouette coefficient for K-Means. Possible design improvements could include the 3D visualization of the clustering algorithms, a smartphone application to predict the data on the go.

VI. REFERENCES

[1] T. Joachims, *Learning to classify text using support vector machines*. Boston: Kluwer Academic Publishers, 2002.

[2] M. Smales, "Sound Classification using Deep Learning", *Medium*, 2019. [Online]. Available: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7. [Accessed: 30 December 2019].

[3] J. Romero, A. Luque and A. Carrasco, "Animal Sound Classification using Sequential Classifiers", *Proceedings of the 10th International Joint Conference on Biomedical Engineering Systems and Technologies*, 2017. Available: 10.5220/0006246002420247 [Accessed 30 December 2019].

[4] T. Oikarinen et al., "Deep convolutional network for animal sound classification and source attribution using dual audio recordings", *The Journal of the Acoustical Society of America*, vol. 145, no. 2, pp. 654-662, 2019. Available: 10.1121/1.5087827 [Accessed 30 December 2019].

[5] "adiengineer/Animal-sounds-Embedded-Classifier", *GitHub*, 2018. [Online]. Available: https://github.com/adiengineer/Animal-sounds-Embedded-Classifier. [Accessed: 30- Dec- 2019].

[6] T. Sarkar, "Clustering metrics better than the elbow-method", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/clustering-metrics-better-than-the-elbow-method-6926e1f723a6. [Accessed: 30- Dec- 2019].

VII. APPENDIX

Program source code is available at:

https://git.ku.edu.tr/akaygusuz14/animal-sound-classifier-on-cloud

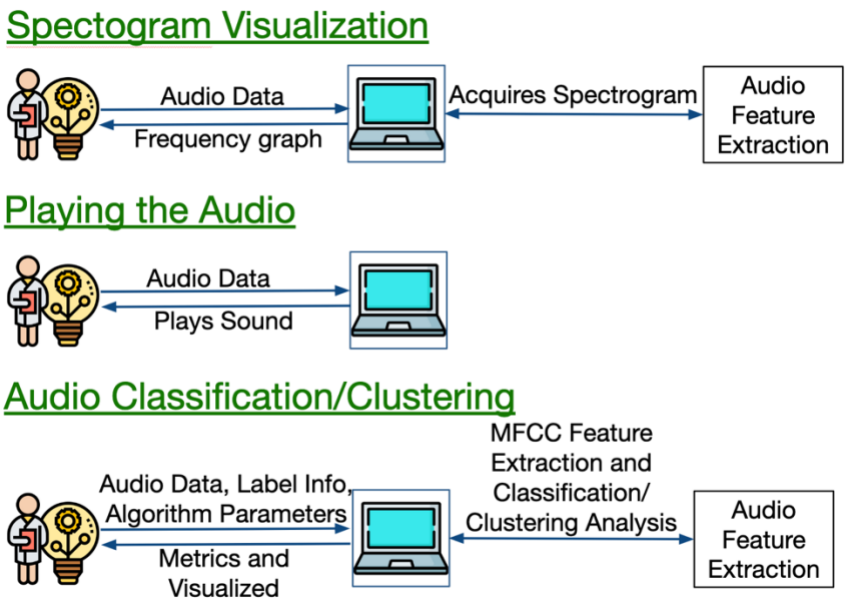

Figure 11 Project Design

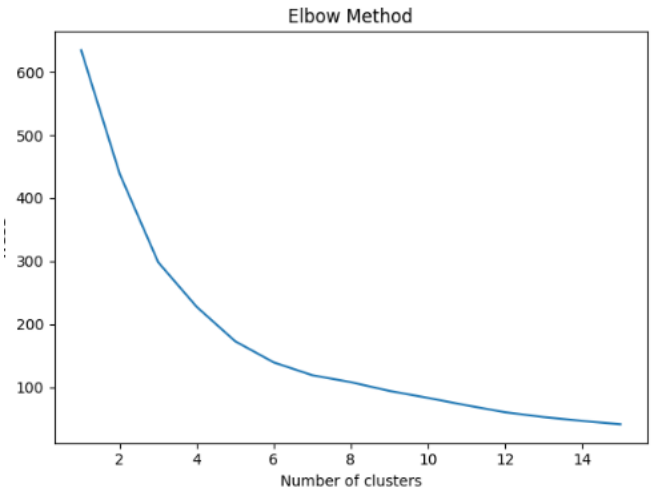

Figure 12 Optimization of Number of Cluster Parameter via Elbow Method

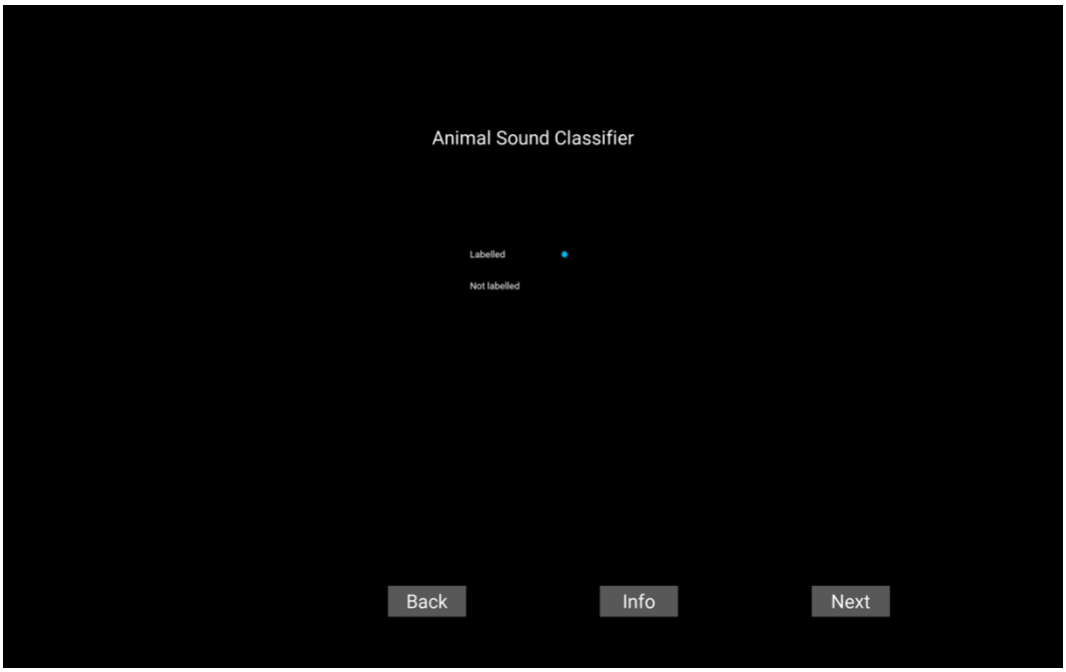

Figure 13 Label Screen

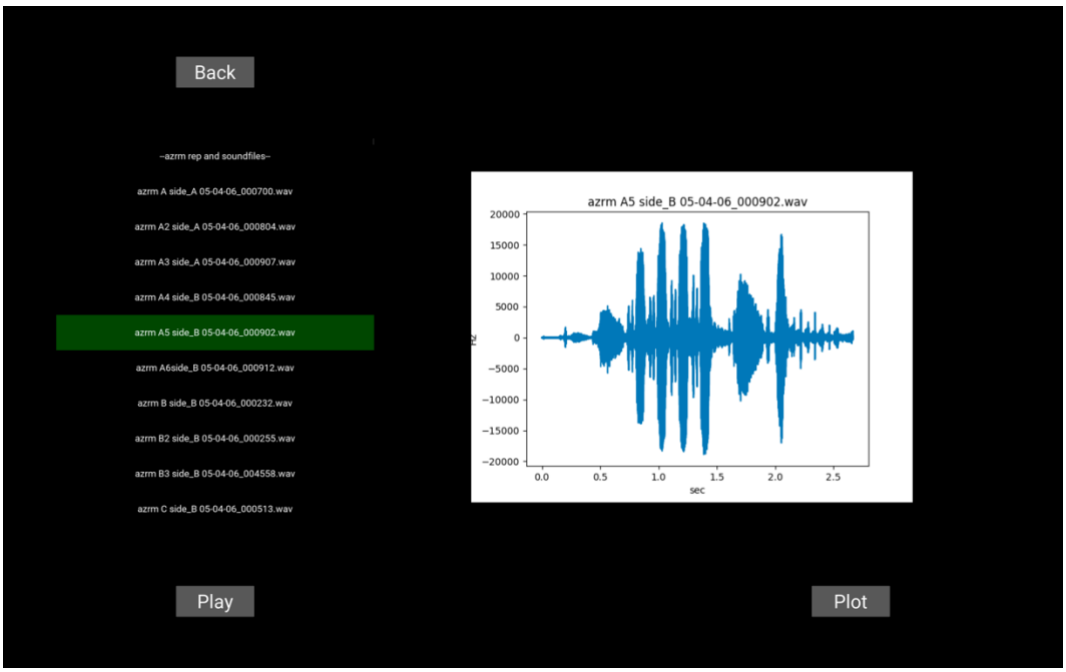

Figure 14 Info Screen

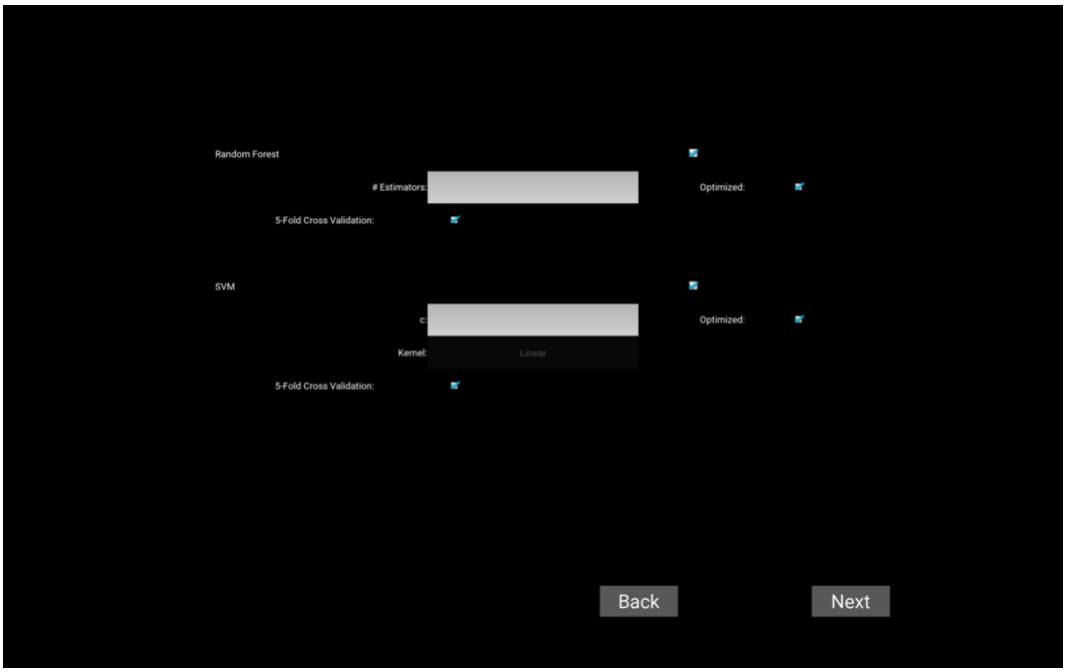

Figure 15 Classification Algorithms Screen

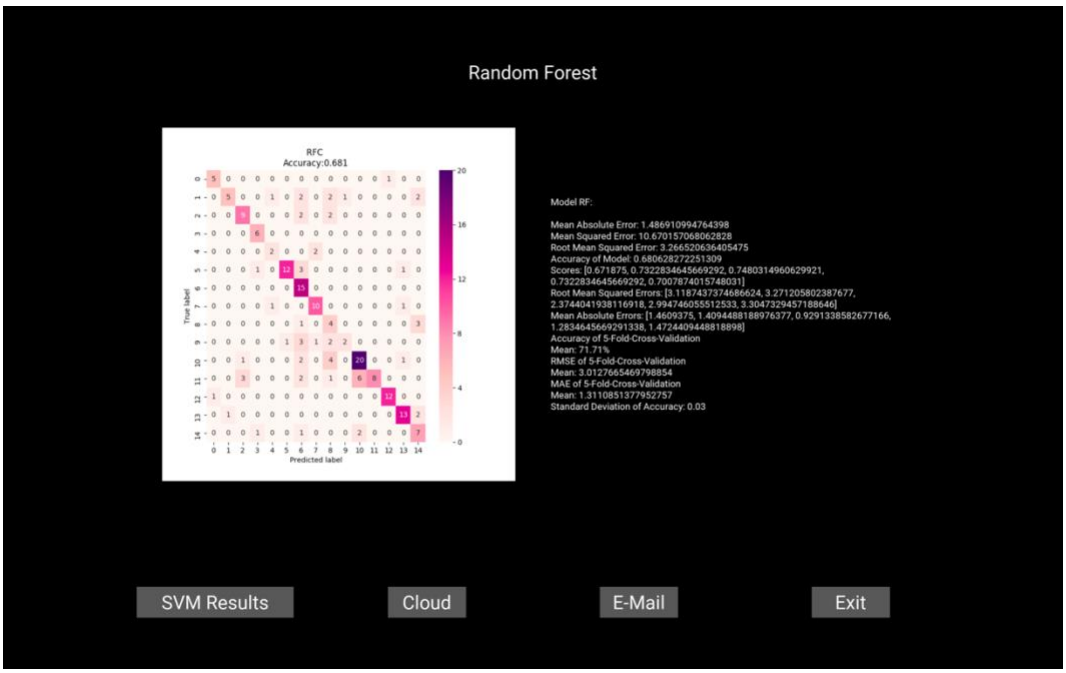

Figure 16 Random Forest Results

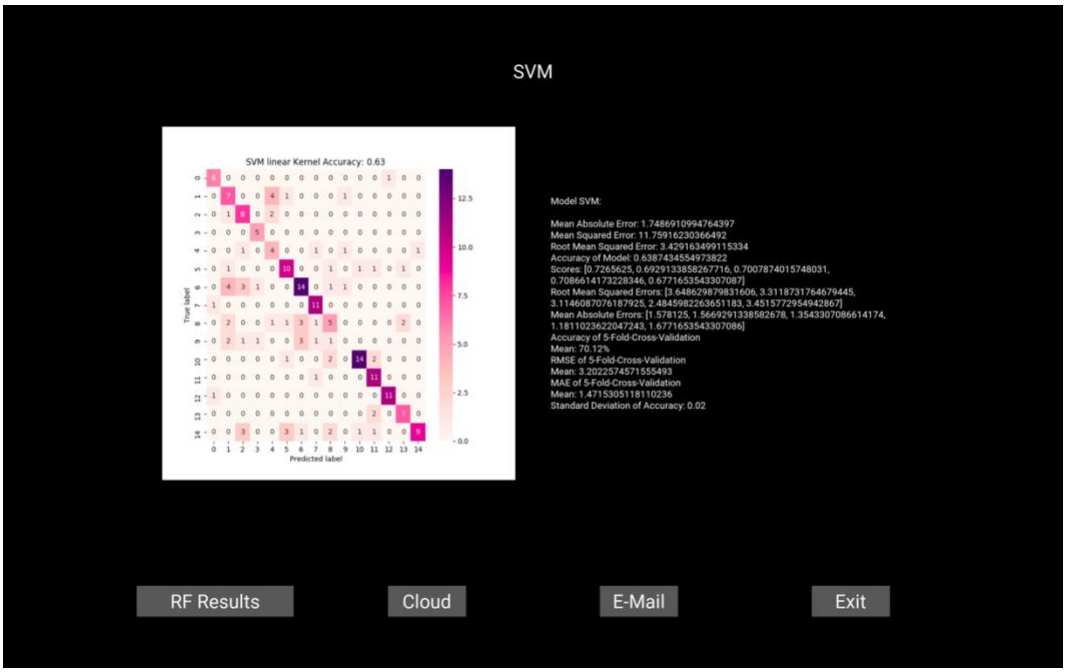

Figure 17 SVM Results

Figure 18 Clustering Algorithms Screen

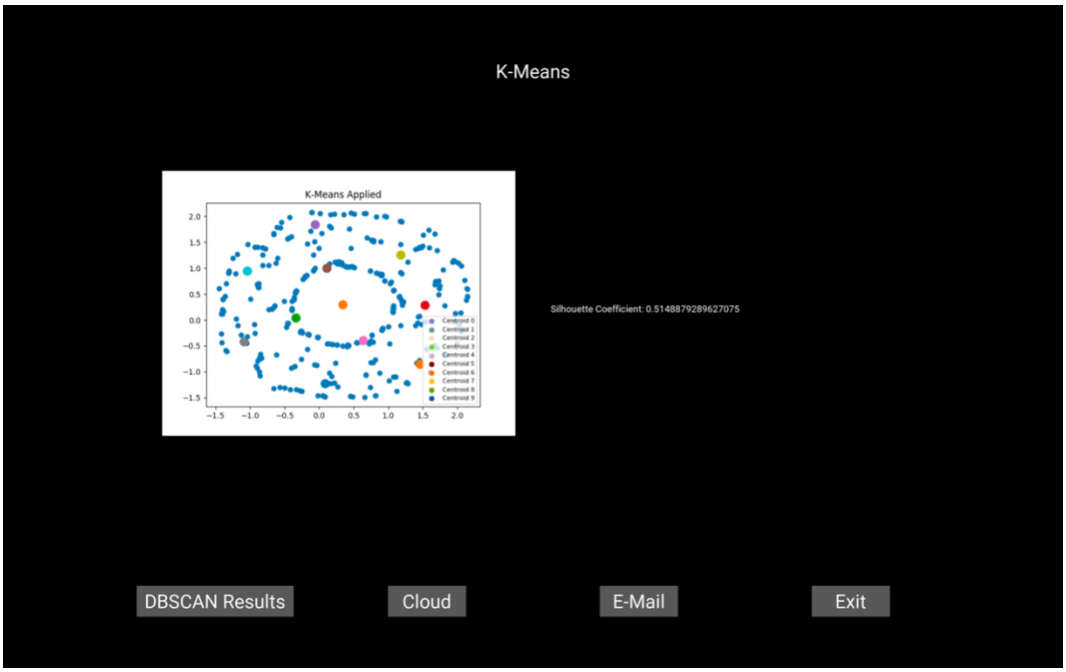

Figure 19 K-Means Result

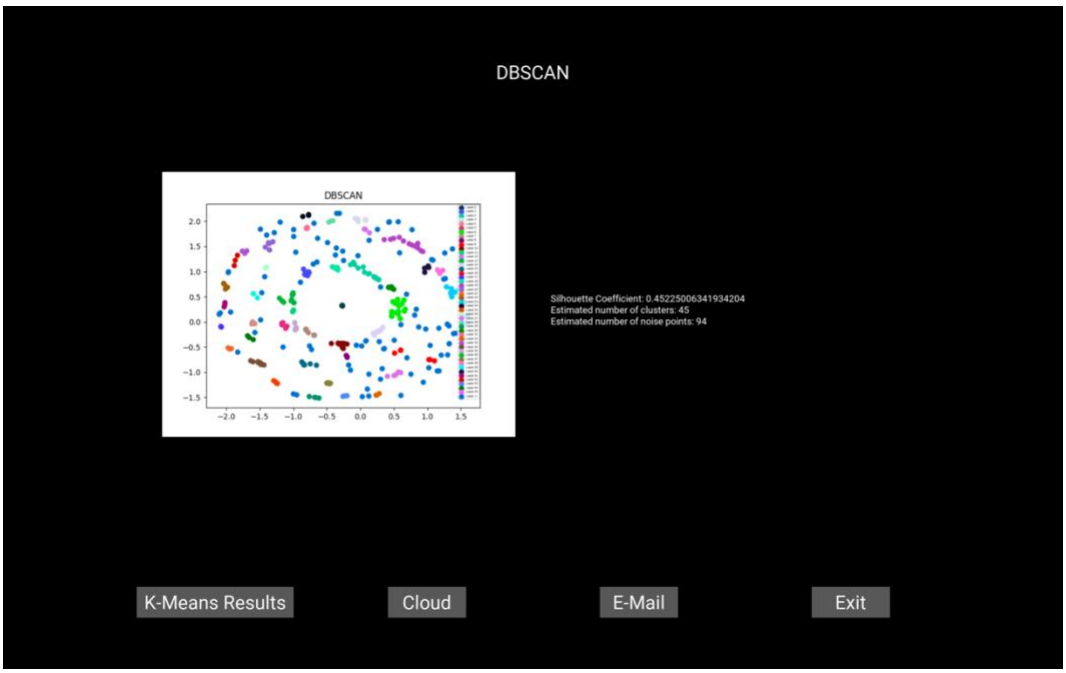

Figure 20 DBSCAN Result