**Predictive Modelling - Assignment**



**School of Information Technology**

**Year 2**

**IT2362- Predictive Modelling**

**Prepared by:** Yap Suen Hong (200562J)

**Module Group:** DBA2001

**Assignment Type:** Predictive Modelling Assignment

**Lecturer: Mr Chee Yong Law**

Date of Submission: 02/02/2022

**Data Preparation**

For my first step, I print out the data frame to see what columns they have and understand what datas I am working with. I will have to do data preparation before doing prediction on the data.

To start my cleaning, I saw my dataset have lots of null value, therefore I use isnull() function to find out all the null value and .sum() function to check which are the columns that contain null value so I can remove it.

After finding out that column 'Popularity','key','instrumentalness' are the column that contains null value. I will do KNN – imputation to replace the null value, it is imputed by finding the samples in the training set “closest” to it and averages these nearby points to fill in the value. I have decided to do KNN – imputation instead of other methods such as elimination because I do not want to remove any data unnecessary as I might not have enough data for the prediction later on. After doing KNN – imputation, the column name is changed to 0, 1 and 2, therefore I have to rename the column back to 'Popularity','key','instrumentalness'.

After doing KNN-imputation, I will have to do feature scaling, I do feature scaling because some feature are much larger values compared to others, the larger value will dominate and affects the results more than the other features. I have used Min-Max Scaling to rescale the features.

Once I am done with feature scaling, I create a boxplot to check for outliers (Figure A1). After finding out I got lots of outliers I will remove them. I have decided to remove them by comparing their z-score. I will first call scipy.stats.zscore() to get a NumPy array containing the z-score of each value. Next I will call numpy.abs() to convert each element to its absolute value. Use the syntax (array < 3).all(axis=1) with array as the previous result to create a boolean array. Filter the original DataFrame with this result. The shape for original DF is (17996, 17) after removing outliers the DF shape is (16482, 17), 1514 data got removed. I created another box plot to compare with the first boxplot to show that outliers got removed (Figure A2)

After removing outliers, I have to remove duplicate data. In this situation, duplicate data means that they have the same artist name and track name, therefore I use drop\_duplicates() to remove data with same artist name and track name only keeping the last one. With this, I am done with data preparation and I will move on to data understanding

**Data Understanding**

Since I am predicting for their class, I first have to check how many data each classes have, so I created a bar chart showing how many data each classes have (Figure A3). With this, I can learn that class 1 has the most number of data while class 7 have the least.

After seeing how many data are there in each class, I want to see the Percentage of each class occupy, therefore I created a pie chart to have a better visualization (Figure A4)

Next I calculate the Mean, Median and Mode. They each tell me what value in a data set is typical or representative of each column. (Figure A19)

I also calculated the minimum and maximum of each column in the dataset (Figure A20).

After calculating of the minimum and maximum, I have decided o do a heatmap (Figure A21). A heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors.

**Predictive Modelling**

**Splitting Data Into Training and Testing set**

After data understanding, I can finally start on predictive modelling. First I have to remove columns that are not needed for training model, hence I will remove 'Artist Name', 'Track Name' as they are object not float. I have determined class as my label and this question is a classification problem. I split my data into X and Y, Y containing my label which is ‘class’ and X containing all the other columns. I tried to use Y to do predictive modelling but I cant as there would be an error, the error is ‘ValueError: Unknown label type: 'continuous'’. After researching, I found out that am passing floats to a classifier which expects categorical values as the target vector. I need to change it to a list and convert it by using sklearn LabelEncoder() function. After transforming it I can finally split it into training and testing set for my classification models.

**Decision Tree**

For my first model, I use Decision Tree. A decision tree is a machine learning algorithm that partitions the data into subsets. The partitioning process starts with a binary split and continues until no further splits can be made. Various branches of variable length are formed. I import DecisionTreeClassifier from sklearn.tree, and fit X an Y training data to the classifier. I use get\_depth() to return the depth of the decision tree, in this case 29. After getting the depth I print the importance of each feature using feature\_importances\_. In order to have a better visualization, I created a bar chart (Figure A5) using matplotlib.pylot and seaborn. After creating the barchart, I created a Confusion Matrix, a Classification Report (Figure A6) and find out their Accuracy Score which is 0.410 (3s.f.). Next, I will have to do find out the best depth value for the tree to prevent underfitting or overfitting, therefore I use matplotlib to create a line chart (Figure A7) to show the tree level for the best accuracy. Based on the plot, we can see that as we increase the depth from 2, the accuracy of the prediction increases. However, once pass a certain point, the improvement stops and the accuracy actually becomes worse. This shows that we have start to overfit. So the best result for accuracy is around a depth of 9. Therefore I have print the Confusion Matrix, Classification Report (Figure A8) and Accuracy Score which is 0.475 (3s.f) for the model when the depth is 9.

**Logistic Regression**

For Logistic Regression, I have to import LogisticRegression from sklearn.linear\_model, after importing the classifier, I have to fit my X and Y training data to it. Once I am done, I will I will print my Confusion Matrix, Classification Report (Figure A16) and Accuracy Score which is 0.541 (3s.f).

**K-Nearest Neighbor**

For my third model, I have decided to use K-Nearest Neighbor algorithm . The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. To start off, I have to find the best value for K in the KNN algorithm to have the best accuracy, hence I have to derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate. After creating 2 line chart (Figure A9 & 10) we will have the smallest error which is 0.556 (3s.f.) at K=30 and maximum accuracy 0.444(3s.f.) at K=30 hence the best value for K is 30. To do KNN algorithm, I have import KNeighborsClassifier from sklearn.neighbors, after fitting the train dataset into the classifier, I will print the Confusion Matrix, Classification Report (Figure A11) and Accuracy Score which is 0.442 (3s.f)

**Naive Bayes**

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. To start off I import Guassian Naive Bayes model from sklearn.naive\_bayes, once I created a Guassian Classifier, I have to train the model using X and Y train data set. After training it, I will print the Confusion Matrix, Classification Report (Figure A12) and Accuracy Score which is 0.445 (3s.f).

**Random Forest**

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model’s prediction. In short, random forest combines several decision trees. To begin, I import RandomForestClassifier from sklearn.ensemble and fit the X and Y training dataset into the classifier. After fitting into the classifier, I will print the Confusion Matrix, Classification Report (Figure A13) and Accuracy Score which is 0.541 (3s.f). Next. I have to see which features are the most important using feature\_importances\_. Once I got the list of the importance of each features, I will create them into a bar chart (Figure A14) using matplotlib for better visualization. As you can see, the important features for random forest classifier and decision tree classifier is different, random forest classifier bar chart would be more accurate since it is many decision tree and they chose the best one, hence it will be the most accurate.

**Gradient Boost**

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. The Gradient Boosting Classifier depends on a loss function. A custom loss function can be used, and many standardized loss functions are supported by gradient boosting classifiers, but the loss function has to be differentiable. To start gradient boosting, I will first import GradientBoostingClassifier from sklearn.ensemble. I will then fit my X and Y train data to the classifier. Once I am done, I will set different learning rates, so that I can compare the performance of the classifier's performance at different learning rates. I will be looking at the classifier's accuracy on the validation set. Base on the output printed, 0.5 gives us the best performance on both validation and training. I will then print the Confusion Matrix, Classification Report (Figure A15) and Accuracy Score which is 0.507 (3s.f).

**Comparing Different Models**

For this part, I will take into consideration each Classifiers’ Classification report average of their precision, recall and f1-score, I would also take into consideration their accuracy score. I will create a table for better visualization.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Accuracy |
| Decision Tree | 0.51 | 0.48 | 0.48 | 0.475 |
| Logistic Regression | 0.50 | 0.48 | 0.49 | 0.493 |
| K-Nearest Neighbor | 0.49 | 0.39 | 0.41 | 0.442 |
| Naïve Bayes | 0.45 | 0.51 | 0.46 | 0.445 |
| Random Forest | 0.60 | 0.55 | 0.55 | 0.541 |
| Gradient Boost | 0.51 | 0.50 | 0.49 | 0.507 |

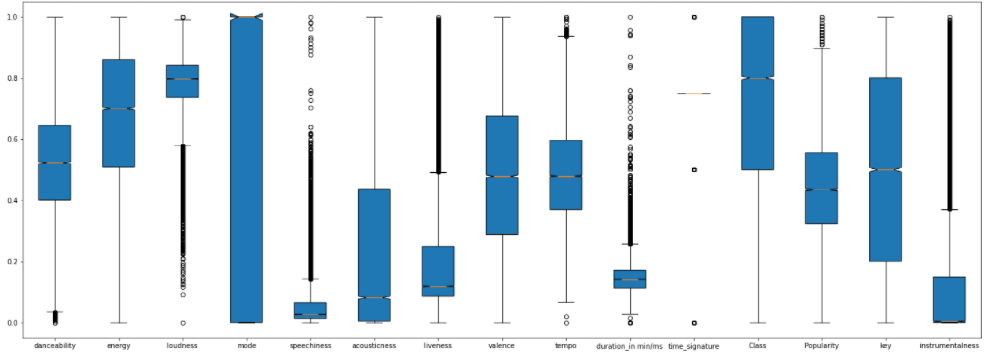
As you can tell from the table, Random Forest have the highest Precision, Recall, F1-Score and Accuracy. However, 1 bad thing about random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time prediction. A more accurate prediction requires more trees, which results in a slower model. So the second best classifier to use would be Gradient Boost as it has the second highest Precision, Recall, F1-Score and Accuracy.

**Doing Prediction**

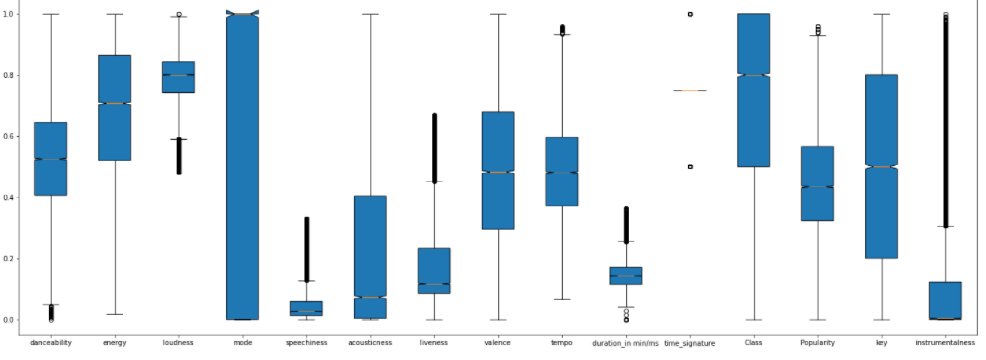
I random generated a dataset of 100 datas (Figure 17) in order to do prediction using the models and see what class they assign to for each data (Figure 18). The number I generated is between 0 to 0.9 as after doing feature scaling all the numbers in the dataset is between 0 to 0.9.

**Figures**

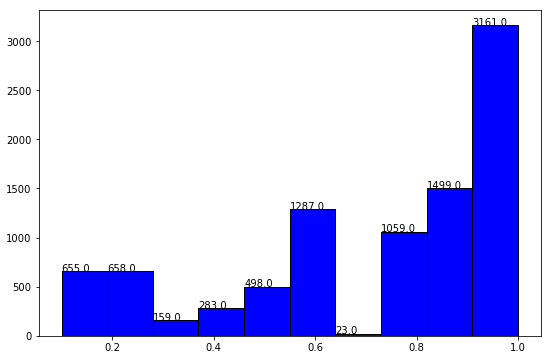
**Figure A1**

****

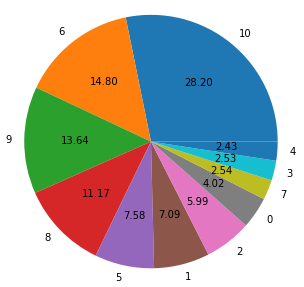
**Figure A2**

****

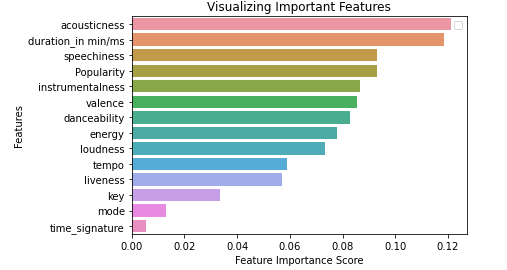
**Figure A3**

****

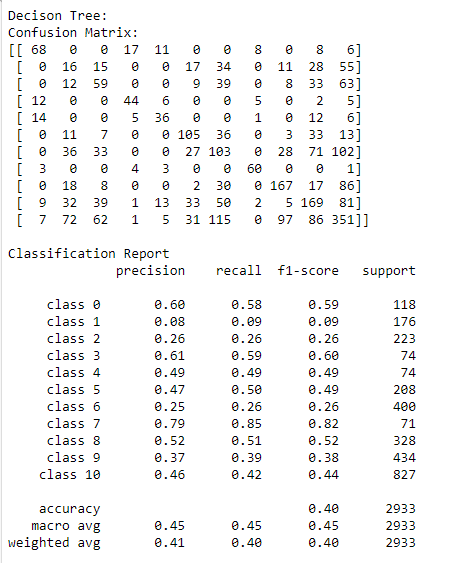
**Figure A4**

****

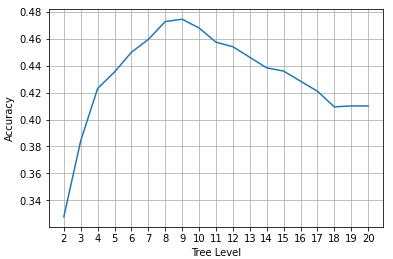
**Figure A5**

****

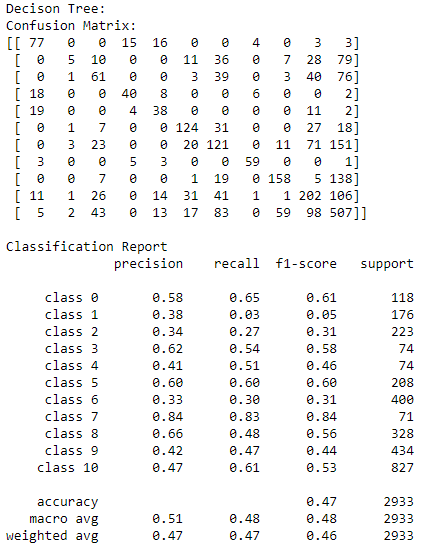
**Figure A6**

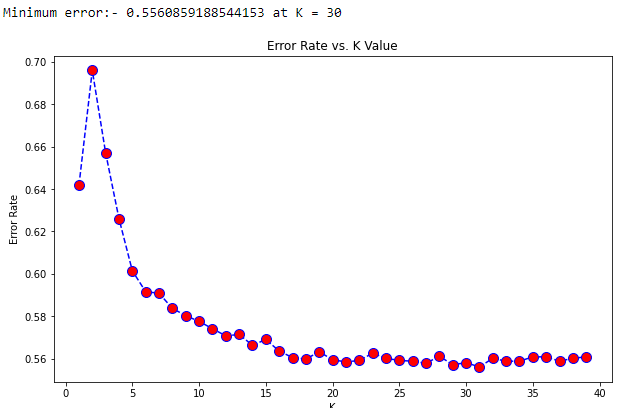


**Figure A7**

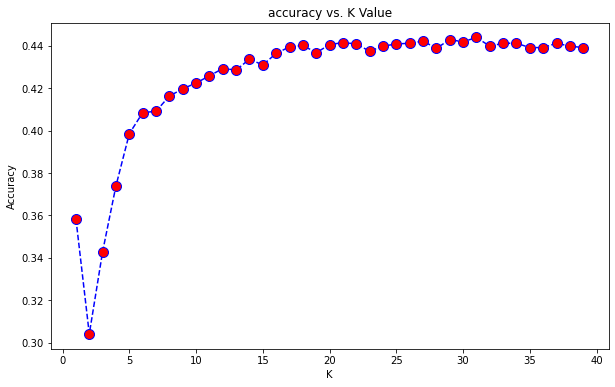
****

**Figure A8**

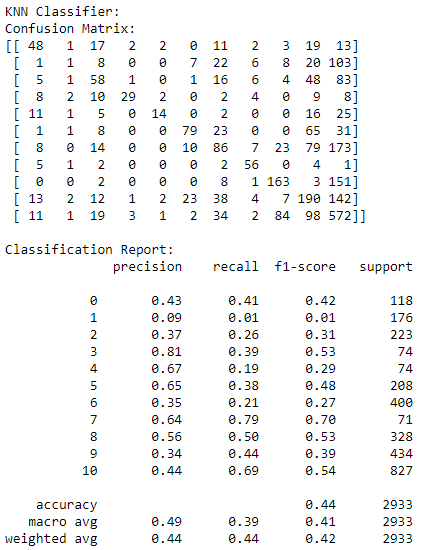
****

**Figure A9** ****

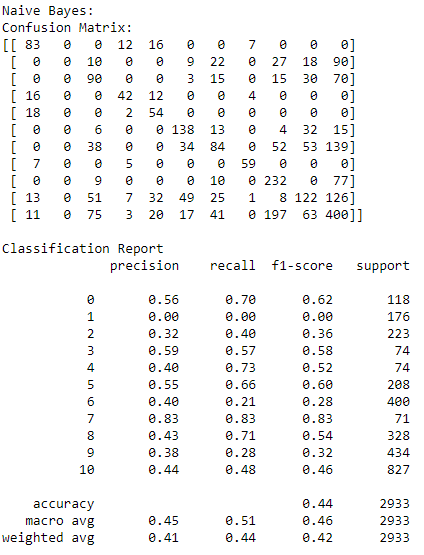
**Figure A10**

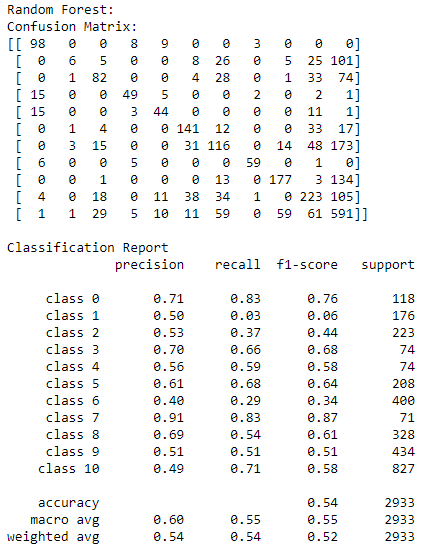
**Figure A11**

****

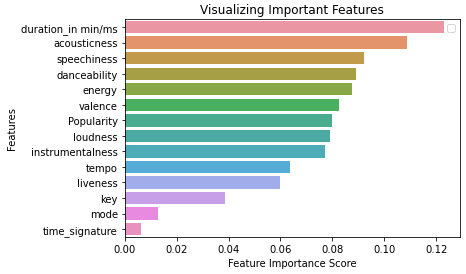
**Figure A12**

****

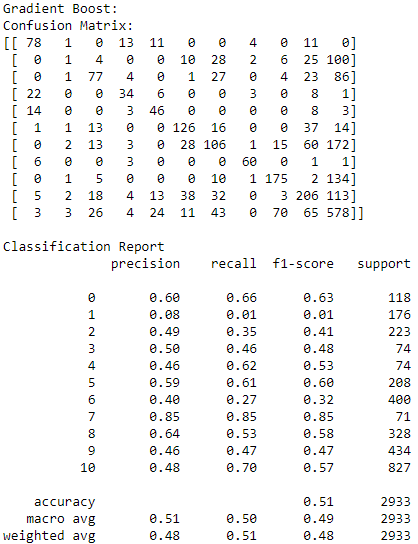
**Figure A13**

****

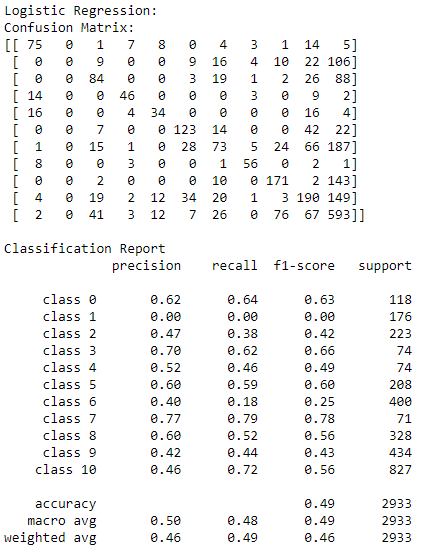
**Figure A14**

****

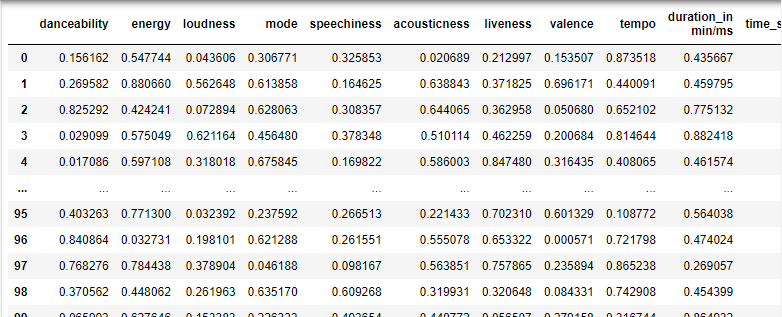
**Figure A15**

****

**Figure A16**

****

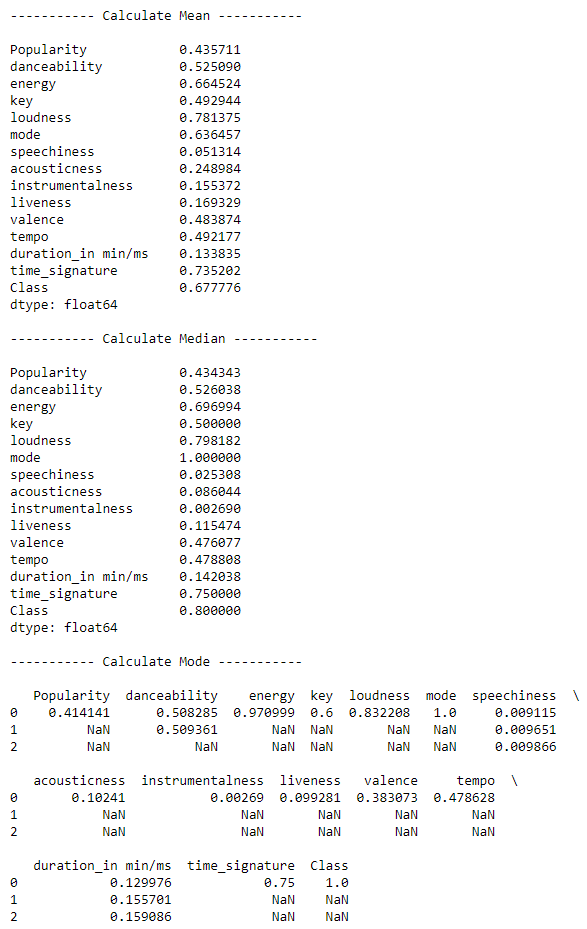
**Figure A17**

****

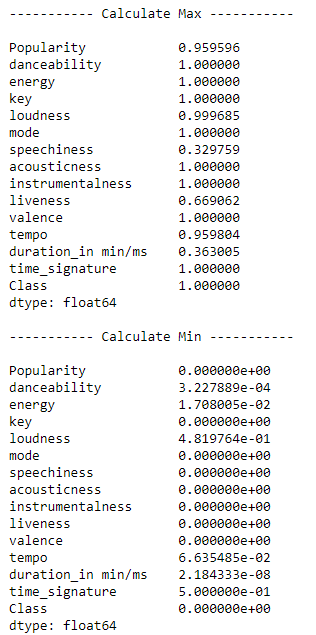
**Figure A18**

****

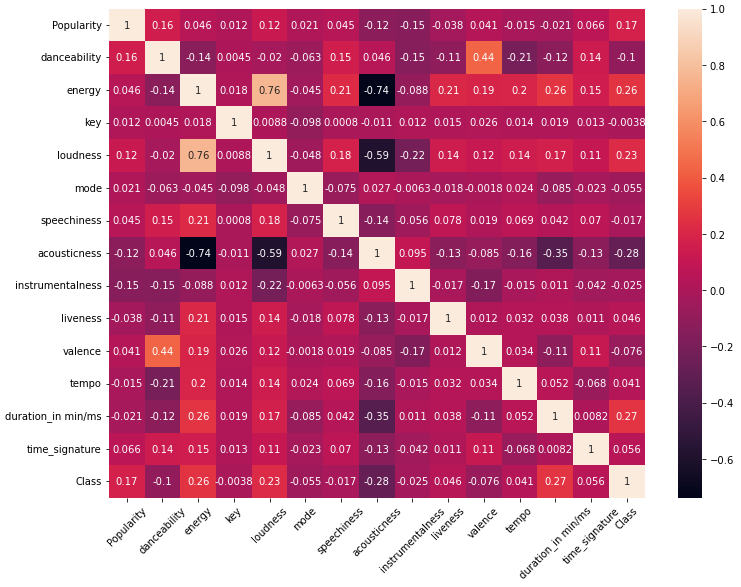
**Figure A19**

****

**Figure A20**

****

**Figure A21**

****