॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS
CENTRE FOR MANAGEMENT
STUDIES, NOIDA

Case Study Report Component-4/5

Predictive Analytics

The objectives of this course case study are to understand predictive analytics methods; develop expertise in the use of popular software such as R for predictive analytics; and identify the most appropriate predictive analytics methods for presenting data-driven solutions.

Prepared by:

Kamal Garg- 20021021108

Yuvraj Singh Dua- 20021021295

Aftab Aalam- 20021021621

Vinayak Kumar - 20021021286

Presented to:

Dr Kriti Priya Gupta

INTRODUCTION

Objective: Based on the given data (i.e. Spotify), we have to develop an ML algorithm that can best predict the song preference based on the given features.

Data Summary:

Number of Observations: 2017

Total number of Variables: 17

About the Variables:

1. **Song_name** - Name of the song
2. **Artist_name** - Name of the artist who performed the song
3. **Acousticness** - A confidence measure from 0 to 1 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
4. **Danceability** - Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
5. **Duration_ms** - Duration of the song in milliseconds.
6. **Energy** - Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
7. **Instrumentalness** - Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
8. **Key** - The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D ♭ , 2 = D, and so on.
9. **Liveness** - Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
10. **Loudness** - The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks.
11. **Mode** - Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is represented by 0.
12. **Speechiness** - Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording, the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either

in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

13. **Tempo** - The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece, and derives directly from the average beat duration.
14. **Time_signature** - An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

Data Types:

- Continuous: 10
- Categorical: 4

Cleaning the Data:

By Using following function to check missing values. Then after the imputation we get no missing values.

```
####checking for missing values
map(s, ~sum(is.na(.)))
###imputing missing values
s<-impute(s, method = 'median/mode')
###again checking for missing values
map(s, ~sum(is.na(.)))
```

Normality Assumption Testing:

```
> skewness(s$acousticness)
[1] 1.657159
> skewness(s$danceability)
[1] -0.4192975
> skewness(s$duration_ms)
[1] 2.502925
> skewness(s$energy)
[1] -0.9123313
> skewness(s$instrumentalness)
[1] 1.951303
> skewness(s$liveness)
[1] 1.951251
> skewness(s$loudness)
[1] -2.227983
> skewness(s$speechiness)
[1] 2.307863
> skewness(s$tempo)
[1] 0.4387318
> skewness(s$valence)
[1] 0.07833138
```

```
> kurtosis(s$acousticness)
[1] 4.738323
> kurtosis(s$danceability)
[1] 2.790029
> kurtosis(s$duration_ms)
[1] 14.90885
> kurtosis(s$energy)
[1] 3.568227
> kurtosis(s$instrumentalness)
[1] 5.252605
> kurtosis(s$liveness)
[1] 7.082525
> kurtosis(s$loudness)
[1] 10.90249
> kurtosis(s$speechiness)
[1] 9.24172
> kurtosis(s$tempo)
[1] 3.038524
> kurtosis(s$valence)
[1] 1.988942
```

-Based on the above outcome of the values**, According to skewness-kurtosis approach:** The skewness and kurtosis of above all variables, which is not within the recommended range of **±1 and ±4** except few. Hence variables are not normally distributed.

Therefore: accousticness, duration, instrumentalness, liveliness, loudness, speechness have highly skewed values therefore not normal.

Treatment of the above variables:

Data cleaning (accousticness)- Since the data is not normal, Square root transformation is used to reduce non-normality.

```
> #####cleaning accousticness
> s1$acousticness<-sqrt(s1$acousticness)
```

Data cleaning (duration)- Since the data is not normal, Need to store the original data in a new variable using below function to reduced non-normality.

```
> #####cleaning duration
> outduration<-boxplot(s1$duration_ms)$out
> length(outduration)
[1] 39
> s1<-s1[-which(s1$duration_ms %in% outduration),]
> boxplot(outduration)
```

Data cleaning (instrumentalness)- Since the data is not normal so that, Need to store the original data in a new variable using below function to reduced non-normality. But still it is not working. So we are not going to take this variable in model building.

```
> #####cleaning instrumenatlness
> outinstrumentalness<-boxplot(s1$instrumentalness)$out
> length(outinstrumentalness)
[1] 373
> s1<-s1[-which(s1$instrumentalness %in% outinstrumentalness),]
> boxplot(outinstrumentalness)
> s1$instrumentalness<-log10(s1$instrumentalness+1)
> skewness(s1$instrumentalness)
[1] 3.795911
> kurtosis(s1$instrumentalness)
[1] 17.94327
```

Data cleaning (liveliness)- Since the data is not normal, log transformation is used to reduce non-normality.

```
> #####cleaning liveliness
> s1$liveness<-log10(s1$liveness)
```

Data cleaning (loudliness)- Since the data is not normal so that, Need to store the original data in a new variable using below function to reduced non-normality.

```
> ####cleaning loudliness
> outloud<-boxplot(s1$loudness)$out
> length(outloud)
[1] 85
> s1<-s1[-which(s1$loudness %in% outloud),]
> boxplot(outloud)
```

Data cleaning (speechness)- Since the data is not normal, log transformation is used to reduced non-normality.

```
> ####cleaning speeechness
> s1$speechiness<-log10(s1$speechiness)
```

After the treatment of the variables by using various techniques, we get values that are mostly under controlled.

```
> skewness(s1[c(2,3,4,5,6,8,9,11,12,14)])
    acousticness     danceability      duration_ms           energy instrumentalness
      -0.40859301      -0.32105094       0.39186842      -0.51767888       3.79591110
         liveness         loudness      speechiness           tempo          valence
       0.42552605      -0.60255641              NaN       0.45839932       0.02128715
> kurtosis(s1[c(2,3,4,5,6,8,9,11,12,14)])
    acousticness     danceability      duration_ms           energy instrumentalness
        2.506072         2.807673         3.186536         2.733793        17.943274
         liveness         loudness      speechiness           tempo          valence
        2.722031         3.203935              NaN         2.785013         2.091889
```

-Based on above outcome of the values**, According to skewness-kurtosis approach:** The skewness and kurtosis of above all variables, which are within the recommended range of **±1 and ±4**. Hence variables are normally distributed.

Except few such as (instrumentalness), so we are not going to take this variable in model building.

Also, the **boxplots** for all the variables, it can be established that most of the data does not have many outliers.

Data Partition

Set. Seed () function sets the starting number used to generate a sequence of random numbers. It ensures that you get the same result if you start with that same seed each time you run the same process. we used set.seed(100)

CreateDataPartition(Y, p, list=F) × It is used to split the dataset into training and test datasets, whereas The train function is used to estimate coefficient values for various regression and classification based modelling functions

Here, **Y** is the source dataset (If the y argument to this function is a factor, the random sampling occurs within each class); **p** is the percentage of data that should go into training set; **list=F**, prevents returning the result as a list.

```
#creating data partition
set.seed(100)
intrain<-createDataPartition(y=s1$song_preference, p=0.8, list=FALSE)
training1<-s1[intrain,]
testing1<-s1[-intrain,]
str(training1)
str(testing1)
```

Significant Predictors (Only continuous + significant IVs)

Model1.1: p-values for other variables are <0.05. Hence, all the IVs except significantly predict.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.15682    0.62517  -5.050 4.43e-07 ***
danceability  1.43193    0.36897   3.881 0.000104 ***
energy        1.88774    0.44886   4.206 2.60e-05 ***
loudness     -0.33137    0.03447  -9.613  < 2e-16 ***
speechiness   1.00951    0.18056   5.591 2.26e-08 ***
```

Confusion matrix (with positive class = 1)

```
> p1.1<-predict(model1.1, newdata = testing1)
> confusionMatrix(p1.1, testing1$song_preference, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 134  61
         1  46 116

               Accuracy : 0.7003
                 95% CI : (0.6498, 0.7474)
    No Information Rate : 0.5042
    P-Value [Acc > NIR] : 4.256e-14

                  Kappa : 0.4001

 Mcnemar's Test P-Value : 0.1759

            Sensitivity : 0.6554
            Specificity : 0.7444
         Pos Pred Value : 0.7160
         Neg Pred Value : 0.6872
             Prevalence : 0.4958
         Detection Rate : 0.3249
   Detection Prevalence : 0.4538
      Balanced Accuracy : 0.6999

       'Positive' Class : 1
```

Evaluation of various models

Selecting the performance metric

#Only continuous + significant IVs.

```
model1.1<-train(data=training1,
song_preference~danceability+energy+loudness+speechiness, method='glm',
family='binomial')

summary(model1.1)

p1.1<-predict(model1.1, newdata = testing1)

confusionMatrix(p1.1, testing1$song_preference, positive = "1")
```

Interpretation: GLM model to predict the binary response variable song_preference based on four continuous predictor variables: danceability, energy, loudness, and speechiness.

Based on the output of the summary() function, all four predictor variables are statistically significant at the 5% level, with p-values less than 0.05. Specifically, the coefficient for danceability is positive, indicating that higher danceability is associated with a higher probability of preferring a song (song_preference=1). The coefficients for energy, loudness, and speechiness are all negative, indicating that higher values of these variables are associated with a lower probability of preferring a song.

Based on the output of the confusionMatrix() function, an overall accuracy of 70% can be observed. The specificity of the model is 74.4%

Overall, the results suggest that the four predictor variables are useful in predicting the binary response variable song_preference, and the model has a reasonably good performance in terms of predicting the positive and negative cases in the testing data. However, further analysis and validation are needed to determine the robustness and generalizability of the model.

```
model2.1<-train(data=training1,
song_preference~danceability+energy+loudness+speechiness, method='naive_bayes')

p2.1<-predict(model2.1, newdata = testing1)

confusionMatrix(p2.1, testing1$song_preference, positive="1")
```

Interpretation: Based on the output of the confusionMatrix() function,an overall accuracy of 68.35%. The specificity 77.22%.

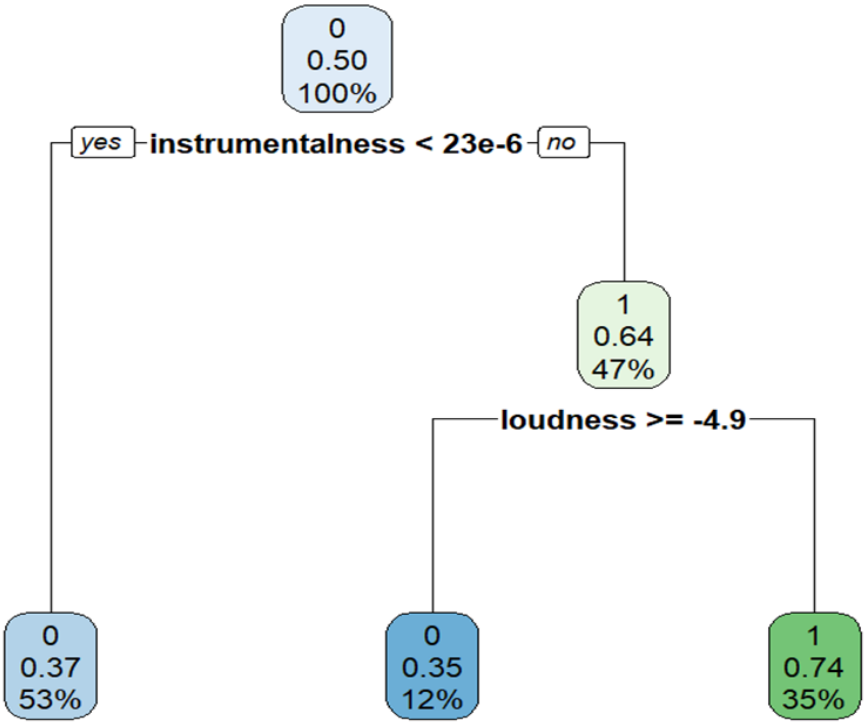
Compared to the GLM model in the previous analysis, the Naive Bayes model achieved slightly better performance in terms of specificity. This suggests that the Naive Bayes model may be a better choice for predicting the binary response variable song_preference based on the four continuous predictor variables. However, as with any model, further validation and evaluation are necessary to determine its robustness and generalizability.

#Decision Trees

```
dt<-train(data=training1, song_preference~danceability+instrumentalness+energy+loudness, method="rpart")

rpart.plot(dt$finalModel)

predsong<-predict(dt, newdata = testing1)

confusionMatrix(predsong, testing1$song_preference,positive="1")

#specificity-81 class=1
```

Output here

Interpretation: The first line of code trains the model using the rpart method and stores it in the variable 'dt'. The second line creates a visual representation of the decision tree using the rpart.plot function. The third line uses the trained model to predict the values for the testing dataset and stores the result in the variable 'predsong'.The fourth line creates a confusion matrix and calculates the specificity of the model. Specifically, it calculates the number of correctly classified positive instances divided by the total number of actual positive instances. The result shows that the specificity of the model is 81% .



#Random Forests

```
rf1<-train(song_preference~acousticness+instrumentalness+danceability+duration_ms+energy+liveness+loudness+speechiness+tempo+valence, data=training1, method="rf")

rf1

rf1$finalModel

varImp(rf1)

predsong_rf1<-predict(rf1, newdata = testing1)

confusionMatrix(predsong_rf1, testing1$song_preference, positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 152  35
         1  28 142

               Accuracy : 0.8235
                 95% CI : (0.7799, 0.8616)
    No Information Rate : 0.5042
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6469

 Mcnemar's Test P-Value : 0.4497

            Sensitivity : 0.8023
            Specificity : 0.8444
         Pos Pred Value : 0.8353
         Neg Pred Value : 0.8128
             Prevalence : 0.4958
         Detection Rate : 0.3978
   Detection Prevalence : 0.4762
      Balanced Accuracy : 0.8234

       'Positive' Class : 1
```

Interpretation: First we train the Random Forest model using the train function with the specified independent and dependent variables, and the method "rf". The trained model is stored in the variable 'rf1'.

Then we display the details of the trained model 'rf1', including the number of trees, the mtry value, and other parameters.

Then we display the variable importance measures of the trained model 'rf1', which shows the relative importance of each independent variable in the model.

The fourth line of code uses the trained model to predict the values for the testing dataset and stores the result in the variable 'predsong_rf1'.

Lastly the output consists of a confusion matrix and calculates the specificity of the model. Specifically, it calculates the number of correctly classified positive instances divided by the total number of actual positive instances. The result shows the specificity of the model is 82% and correctly classified the class 1 instances.

Result: Model with the highest specificity

```
rf6<-train(song_preference~acousticness+instrumentalness+danceability+duration_ms+energy+loudness+speechiness+tempo+valence, data=training1, method="rf")

rf6

rf6$finalModel

varImp(rf6)

predsong_rf6<-predict(rf6, newdata = testing1)

confusionMatrix(predsong_rf6, testing1$song_preference, positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 151  30
         1  29 147

               Accuracy : 0.8347
                 95% CI : (0.7921, 0.8717)
    No Information Rate : 0.5042
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6694

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8305
            Specificity : 0.8389
         Pos Pred Value : 0.8352
         Neg Pred Value : 0.8343
             Prevalence : 0.4958
         Detection Rate : 0.4118
   Detection Prevalence : 0.4930
      Balanced Accuracy : 0.8347

       'Positive' Class : 1
```

Result: Model with the highest overall accuracy

Interpretation: rf6 is an object that stores the results of the random forest classification. The output of the object includes information such as the number of trees grown, the number of variables used at each split, and the out-of-bag estimate of the error rate.

rf6$finalModel gives the final model that is built by the random forest algorithm. This includes information on the structure of each tree in the forest, the importance of each predictor variable, and other details.

varImp(rf6) shows the importance of each predictor variable in the model, measured by the mean decrease in accuracy when the variable is randomly permuted. This can help identify which variables are most influential in determining the response variable.

predsong_rf6 is a vector of predicted values for the response variable, based on the random forest model.

confusionMatrix(predsong_rf6, testing1$song_preference, positive = "1") outputs a confusion matrix that summarizes the accuracy of the predictions made by the random forest model. The positive class is defined as "1", which in this case is the class of individuals who have a preference for a certain type of music. The matrix shows the number of true positives, true negatives, false positives, and false negatives, and calculates various measures of accuracy such as sensitivity, specificity, and overall accuracy.

Description of the tasks performed by each group member

Yuvraj Singh Dua
assisted in Data cleaning.
Predictive models building, model interpretation, presentation.
assisting in final report

Kamal Garg:
Data cleaning
Model Building

Vinayak Kumar:
Final Report Making
Assisted in data cleaning and decision trees
Model interpretation

Aftab Aalam:
Final Report building
Assisted in data cleaning