CS-GY 6923 MACHINE LEARNING

Professor: Dr. Raman Kannan

Project 1: Ensemble Learning Techniques

Author: Dharu Piraba Muguntharaman

INTRODUCTION:

After running different classification techniques, we could see that the performance of the models weren't that good. Hence, to improve the performance of the model, we use ensemble learning techniques. Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

LOADING DATA AND DATA PRE-PROCESSING:

The music genre dataset is loaded using the read.csv command. To remove null rows, we use the na.omit command. We remove unwanted columns like instance id, artist name etc . The first five rows of the dataset are displayed using the dim command.

```
[1] music_data=read.csv('/content/music_genre.csv',na.strings=c("NA", "NULL"))
```

```
head(music_data)
music_data=na.omit(music_data)
music_data=music_data[-c(1,2,3,16)]
dim(music_data)
```

A data.frame: 6 × 18

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechi
	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<
1	32894	Röyksopp	Röyksopp's Night Out	27	0.00468	0.652	-1	0.941	0.79200	A#	0.115	-5.201	Minor	0
2	46652	Thievery Corporation	The Shining Path	31	0.01270	0.622	218293	0.890	0.95000	D	0.124	-7.043	Minor	0
3	30097	Dillon Francis	Hurricane	28	0.00306	0.620	215613	0.755	0.01180	G#	0.534	-4.617	Major	0
4	62177	Dubloadz	Nitro	34	0.02540	0.774	166875	0.700	0.00253	C#	0.157	-4.498	Major	0
5	24907	What So Not	Divide & Conquer	32	0.00465	0.638	222369	0.587	0.90900	F#	0.157	-6.266	Major	0
6	89064	Axel Boman	Hello	47	0.00523	0.755	519468	0.731	0.85400	D	0.216	-10.517	Minor	0

50000 · 14

Since, our dataset contains a lot of categorical variables we convert them to numeric features using one hot encoding method. Now, we display the total number of features using the dim command.

```
[5]
music_data=one_hot(as.data.table(music_data))
music_data$mode <- as.factor(music_data$mode)
music_data <- one_hot(as.data.table(music_data))
```

```
[6] music_data=select(music_data,-c(4,7))
dim(music_data)
```

50000 · 13

We remove the tempo column since it has a lot of unwanted characters and less significant. The final dataset contains 11 features and one target variable—music genre.

```
[7] music_data=select(music_data,-c(11))

[8] names(music_data)
```

'popularity' · 'acousticness' · 'danceability' · 'energy' · 'instrumentalness' · 'liveness' · 'loudness' · 'mode_Major' · 'mode_Minor' · 'speechiness' · 'valence' · 'music_genre'

SPLITTING DATA INTO TRAIN AND TEST:

In order to perform ensemble learning, the dataset must be split into training and testing datasets. We set the seed to 5596. The whole dataset is split into two datasets such that the training dataset has 80% of the original dataset and the testing dataset has 20% of the original dataset.

```
[10] set.seed(5596)
     idx=sample(1:nrow(music_data),0.8*nrow(music_data))
     train_set=music_data[idx,]
     test_set=music_data[-idx,]
```

```
dim(train_set)
dim(test_set)
```

40000 · 12
10000 · 12

ENSEMBLE LEARNING TECHNIQUES :

1. RANDOM FOREST:

Random forest is an ensemble learning technique that means that it works by running a collection of learning algorithms to increase the preciseness and accuracy of the results. Random forest works by creating multiple decision trees for a dataset and then aggregating the results.

The random forest model is installed using the install.packages() command. The train set is initialized as trdf_RF and the random forest model is called and stored in RF_model variable.

```
install.packages("randomForest")
library(randomForest)
```

```
require(randomForest)
require(caret)
formstr="music_genre~." # Formula argument with all features
trdf_RF=train_set
dim(trdf_RF)
trdf_RF$music_genre=as.factor(trdf_RF$music_genre) # Makes it run as classification
RF_model=randomForest(music_genre~.,trdf_RF)
```

```
40000 · 12
```

The RF_model output shows that the number of tress used is 500 and the number of variables at each split is 3. We could see that the OOB error rate is 46.56% which is very high. This implies that our model fails to classify the genres correctly most of the time.

```
RF_model
```

```
Call:
 randomForest(formula = music_genre ~ ., data = trdf_RF)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 46.56%
Confusion matrix:
              Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
Alternative          1482    35    93        14     506        230      339  247
Anime                 112  2878   276       295     176        178        1   62
Blues                 192   349  2121        64     355        222       13  476
Classical              93   156    94      3355      25         79        1  181
Country               325    55   221         3    2219         60       57  217
Electronic            260   189   278        26     107       2398       74  520
Hip-Hop               193     1     3         0      42         31     1410   39
Jazz                  166    58   474       255     227        589       74 1977
Rap                   157     2     3         0      51         15     2207   22
Rock                  574    17    78        11     436         30      146   88
              Rap Rock class.error
Alternative   212  869   0.6319841
Anime           1   27   0.2815776
Blues           6  214   0.4713360
Classical       0   13   0.1606205
Country        66  793   0.4474602
Electronic     46  109   0.4015473
Hip-Hop      2035  184   0.6419502
Jazz           18  164   0.5059970
Rap          1161  382   0.7097500
Rock          240 2375   0.4055069
```

The test data is initialized as tst_df and the target variable is made to a factor.

```
tstdf_RF=test_set
dim(tstdf_RF)
tstdf_RF$music_genre=as.factor(tstdf_RF$music_genre)
```

```
10000 · 12
```

The test data is passed to the model for prediction. The prediction is stored in p2. The confusion matrix is plotted to understand the performance of the model. It could be seen that the accuracy of the model is 53% which is found to be higher than most of individual classifier accuracies performed in HW1.

```
p2 <- predict(RF_model, tstdf_RF)
confusionMatrix(p2,tstdf_RF$music_genre)
```

```
Confusion Matrix and Statistics

              Reference
Prediction     Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative          369    28    67        28      79          75      53   43
  Anime                  9   726    78        36      21          35       1    8
  Blues                 22    58   514        30      46          66       1  104
  Classical              1    71    14       840       1           8       0   72
  Country              115    39    82         4     541          26       6   61
  Electronic            40    47    63        10       9         587      12  128
  Hip-Hop               83     1     3         0      25          16     379   23
  Jazz                  59    18   118        49      64         133      16  512
  Rap                   64     0     0         1      18           8     555    9
  Rock                 211     6    49         5     180          39      39   38
              Reference
Prediction     Rap Rock
  Alternative   33  149
  Anime          0    5
  Blues          3   14
  Classical      0    3
  Country       11  106
  Electronic     7    8
  Hip-Hop      535   33
  Jazz           3   20
  Rap          300   48
  Rock         108  619

Overall Statistics

               Accuracy : 0.5387
                 95% CI : (0.5289, 0.5485)
    No Information Rate : 0.1062
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4874
```

The sensitivity, specificity, recall and accuracy of the individual classes are displayed below.

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                      0.3792       0.7304       0.5202
Specificity                      0.9385       0.9786       0.9618
Pos Pred Value                   0.3994       0.7900       0.5991
Neg Pred Value                   0.9335       0.9705       0.9482
Prevalence                       0.0973       0.0994       0.0988
Detection Rate                   0.0369       0.0726       0.0514
Detection Prevalence             0.0924       0.0919       0.0858
Balanced Accuracy                0.6589       0.8545       0.7410
                     Class: Classical Class: Country Class: Electronic
Sensitivity                    0.8375         0.5498            0.5911
Specificity                    0.9811         0.9501            0.9640
Pos Pred Value                 0.8317         0.5459            0.6443
Neg Pred Value                 0.9819         0.9508            0.9553
Prevalence                     0.1003         0.0984            0.0993
Detection Rate                 0.0840         0.0541            0.0587
Detection Prevalence           0.1010         0.0991            0.0911
Balanced Accuracy              0.9093         0.7499            0.7776
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                0.3569      0.5130     0.3000      0.6159
Specificity                0.9196      0.9467     0.9219      0.9250
Pos Pred Value             0.3452      0.5161     0.2991      0.4784
Neg Pred Value             0.9233      0.9460     0.9222      0.9557
Prevalence                 0.1062      0.0998     0.1000      0.1005
Detection Rate             0.0379      0.0512     0.0300      0.0619
Detection Prevalence       0.1098      0.0992     0.1003      0.1294
Balanced Accuracy          0.6382      0.7299     0.6109      0.7704
```

The root mean square error is a parameter used to evaluate the performance of the model.The root mean square error of the random forest is displayed. The rmse value of random forest model is found to be 2.9 which is quite higher.

```
[24] #RMSE
     RMSE((as.numeric(p2) - as.numeric(tstdf_RF$music_genre)))
```

```
2.918167233042
```

The bias and variance of the test data is displayed using the bias() and variance() functions. It could be seen that the variance is 4.09 and bias as 0.2. The bias is very low compared to individual classifiers which is good whereas the variance remains the same as compared to individual classifiers.

```
#Bias and variance
print(var(as.numeric(p2), as.numeric(as.factor(tstdf_RF$music_genre))))
bias(as.numeric(p2), as.numeric(as.factor(tstdf_RF$music_genre)))
```

```
[1] 4.099508
0.2103
```

2. BAGGING:

Bagging, or bootstrap aggregation, is a technique used to reduce the variance of your predictions by combining the result of multiple classifiers modeled on different sub-samples of the same dataset. Bagging is simply a special case of a random forest with m=p. Therefore, the randomForest() function can be used to perform both random forests and bagging. Here, we call the random forest() model with mtry=11, the total number of features in our dataset. The result model is called bag_model.

```
[ ] bag_model = randomForest(music_genre~., data = trdf_RF,
                             mtry = 11,
                             importance = TRUE)
```

The bag_model output is displayed. Here, the number of trees is 500 and the number of variables tried at each split is 11. The OOB estimate of error rate is 46.87 which is higher. The confusion matrix is displayed.

```
bag_model
```

```
Call:
 randomForest(formula = music_genre ~ ., data = trdf_RF, mtry = 11,       importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 11

        OOB estimate of  error rate: 46.87%
Confusion matrix:
             Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
Alternative         1535    34   101         20     502         216     330  229
Anime                117  2843   321        254     185         188       2   72
Blues                204   315  2160         64     337         204      13  501
Classical             95   180    95       3302      30          80       1  195
Country              367    47   221          5    2202          59      52  216
Electronic           304   180   285         38     104        2317      74  528
Hip-Hop              216     0     5          0      47          28    1367   31
Jazz                 180    48   479        259     246         571      67 1952
Rap                  160     1     3          0      54          21    2183   20
Rock                 545    13    64          9     437          35     142   82
             Rap Rock class.error
Alternative  209  851   0.6188229
Anime          0   24   0.2903145
Blues          7  207   0.4616152
Classical      0   19   0.1738804
Country       63  784   0.4516932
Electronic    48  129   0.4217619
Hip-Hop     2063  181   0.6528695
Jazz          30  170   0.5122439
Rap         1165  393   0.7087500
Rock         259 2409   0.3969962
```

The test data is passed to the model for prediction. The bagged_estimate stores the predicted values of the predicted model. The confusion matrix is displayed to infer the performance of the model.

```
bagged_estimate = predict(bag_model,
                          newdata =tstdf_RF)
confusionMatrix(as.factor(bagged_estimate),as.factor(tstdf_RF$music_genre))
```

It could be seen that the accuracy of the model is 53% which is found to be higher than most of individual classifier accuracies performed in HW1. It is similar to the random forest model accuracy above and there isn't any much difference here.

```
Confusion Matrix and Statistics

              Reference
Prediction     Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative          377    28    66         27      86          80      57   49
  Anime                  8   718    75         49      11          37       1    7
  Blues                 22    74   514         34      46          75       1  107
  Classical              4    62    14        816       2          10       0   73
  Country              114    38    84          5     539          27      12   59
  Electronic            38    49    64         10      11         561      11  124
  Hip-Hop               87     1     3          0      23          19     389   21
  Jazz                  56    17   120         56      66         138      15  496
  Rap                   62     0     1          1      25          10     541   11
  Rock                 205     7    47          5     175          36      35   51
              Reference
Prediction     Rap Rock
  Alternative   36  134
  Anime          0    4
  Blues          3   10
  Classical      0    1
  Country        9  112
  Electronic    11   12
  Hip-Hop      520   31
  Jazz           3   19
  Rap          313   58
  Rock         105  624

Overall Statistics

               Accuracy : 0.5347
                 95% CI : (0.5249, 0.5445)
    No Information Rate : 0.1062
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4829
```

The sensitivity, specificity, recall and accuracy of the individual classes are displayed below.

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                      0.3875       0.7223       0.5202
Specificity                      0.9376       0.9787       0.9587
Pos Pred Value                   0.4011       0.7890       0.5801
Neg Pred Value                   0.9342       0.9696       0.9480
Prevalence                       0.0973       0.0994       0.0988
Detection Rate                   0.0377       0.0718       0.0514
Detection Prevalence             0.0940       0.0910       0.0886
Balanced Accuracy                0.6625       0.8505       0.7395
                     Class: Classical Class: Country Class: Electronic
Sensitivity                    0.8136         0.5478            0.5650
Specificity                    0.9815         0.9490            0.9634
Pos Pred Value                 0.8310         0.5395            0.6296
Neg Pred Value                 0.9793         0.9506            0.9526
Prevalence                     0.1003         0.0984            0.0993
Detection Rate                 0.0816         0.0539            0.0561
Detection Prevalence           0.0982         0.0999            0.0891
Balanced Accuracy              0.8976         0.7484            0.7642
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                0.3663      0.4970     0.3130      0.6209
Specificity                0.9211      0.9456     0.9212      0.9260
Pos Pred Value             0.3556      0.5030     0.3063      0.4837
Neg Pred Value             0.9244      0.9443     0.9235      0.9563
Prevalence                 0.1062      0.0998     0.1000      0.1005
Detection Rate             0.0389      0.0496     0.0313      0.0624
Detection Prevalence       0.1094      0.0986     0.1022      0.1290
Balanced Accuracy          0.6437      0.7213     0.6171      0.7734
```

The root mean square error of the bagging model is displayed. The rmse value of random forest model is found to be 2.9 which is quite higher. It is similar to the previous random forest model.

```
#RMSE
RMSE(as.numeric(bagged_estimate), as.numeric(tstdf_RF$music_genre))
```

```
2.90137898248402
```

The bias and variance of the test data is displayed using the bias() and variance() functions. It could be seen that the variance is 4.16 and bias as 0.2. The bias is very low compared to individual classifiers which is good whereas the variance remains the same as compared to individual classifiers. The bias is slightly lower than the previous random forest model.

```
[ ]  #BIAS AND VARIANCE
     print(var(as.numeric(bagged_estimate), as.numeric(as.factor(tstdf_RF$music_genre))))
     bias(as.numeric(bagged_estimate), as.numeric(as.factor(tstdf_RF$music_genre)))
```

```
[1] 4.16992
0.2048
```

3. K FOLD CROSS VALIDATION:

The K-fold cross-validation technique is basically a method of resampling the data set in order to evaluate a machine learning model. In this technique, the parameter K refers to the number of different subsets that the given data set is to be split into. Further, K-1 subsets are used to train the model and the left out subsets are used as a validation set.

The necessary libraries like caret,klaR are loaded to import the k fold cross validation model.

```
library(caret)

require(e1071)
install.packages("labelled")
install.packages("klaR")
library(klaR)
```

The train control specifies the type of ensemble learning technique applied. In this case it is cv which is cross validation with k value to be 10. The model is built using the dataset and naïve bayes classifier.

```
set.seed(5596)

train_control <- trainControl(method = "cv",
                              number = 10)

model <- train(music_genre~., data = music_data,
               trControl = train_control,
               method = "nb")
```

The output of the model is displayed. The accuracy of the False and true values are displayed.

```
print(model)
```

```
Naive Bayes

50000 samples
   11 predictor
   10 classes: 'Alternative', 'Anime', 'Blues', 'Classical', 'Country', 'Electronic', 'Hip-Hop', 'Jazz', 'Rap', 'Rock'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 45000, 45000, 45000, 45000, 45000, 45000, ...
Resampling results across tuning parameters:

  usekernel  Accuracy  Kappa
  FALSE      0.42318   0.3590889
   TRUE      0.37020   0.3002222

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust
 = 1.
```

The model is then used to predict using test data. The test data and the trained model is passed as input.

```
tstdf_RF=test_set
dim(tstdf_RF)
tstdf_RF$music_genre=as.factor(tstdf_RF$music_genre)
k_p=predict(model,tstdf_RF)
```

The confusion matrix of the predicted model is displayed. It could be seen that the accuracy of the model is 42% which is similar to the individual naïve bayes classifier. It is low compared to the previous two ensemble learning techniques.

```
confusionMatrix(k_p,tstdf_RF$music_genre)
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         192    23    56        15      43         77      25   32
  Anime                 3   320    67        17      14         45       0   31
  Blues                 9    35   201        14      25         25       5   56
  Classical            10   195    46       871      15         22       0  170
  Country             392   275   392        30     736        146     124  156
  Electronic           49   117    89        14      17        472       9  155
  Hip-Hop             103     2    12         0      45         57     415   72
  Jazz                 62    22   113        39      35        100      23  308
  Rap                  87     2     1         0      22         29     441    5
  Rock                 66     3    11         3      32         20      20   13
             Reference
Prediction    Rap Rock
  Alternative  22   73
  Anime         1    2
  Blues         2    8
  Classical     0   14
  Country     127  457
  Electronic    7   17
  Hip-Hop     287   16
  Jazz         19   35
  Rap         478  128
  Rock         57  255

Overall Statistics

               Accuracy : 0.4248
                 95% CI : (0.4151, 0.4346)
    No Information Rate : 0.1062
    P-Value [Acc > NIR] : < 2.2e-16
```

The sensitivity, specificity, recall and accuracy of the individual classes are displayed below.

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                      0.1973       0.3219       0.2034
Specificity                      0.9595       0.9800       0.9801
Pos Pred Value                   0.3441       0.6400       0.5289
Neg Pred Value                   0.9173       0.9291       0.9182
Prevalence                       0.0973       0.0994       0.0988
Detection Rate                   0.0192       0.0320       0.0201
Detection Prevalence             0.0558       0.0500       0.0380
Balanced Accuracy                0.5784       0.6510       0.5918
                     Class: Classical Class: Country Class: Electronic
Sensitivity                    0.8684         0.7480             0.4753
Specificity                    0.9475         0.7672             0.9474
Pos Pred Value                 0.6485         0.2596             0.4989
Neg Pred Value                 0.9848         0.9654             0.9425
Prevalence                     0.1003         0.0984             0.0993
Detection Rate                 0.0871         0.0736             0.0472
Detection Prevalence           0.1343         0.2835             0.0946
Balanced Accuracy              0.9080         0.7576             0.7114
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                0.3908      0.3086     0.4780      0.2537
Specificity                0.9335      0.9502     0.9206      0.9750
Pos Pred Value             0.4113      0.4074     0.4007      0.5312
Neg Pred Value             0.9280      0.9254     0.9407      0.9212
Prevalence                 0.1062      0.0998     0.1000      0.1005
Detection Rate             0.0415      0.0308     0.0478      0.0255
Detection Prevalence       0.1009      0.0756     0.1193      0.0480
Balanced Accuracy          0.6622      0.6294     0.6993      0.6144
```

The root mean square error of the bagging model is displayed. The rmse value of random forest model is found to be 2.8 which is quite higher. It is slightly lower than the previous two ensemble learning techniques.

```
[25] #RMSE
     RMSE(as.numeric(k_p), as.numeric(tstdf_RF$music_genre))
```

```
2.86311019697112
```

The bias and variance of the test data is displayed using the bias() and variance() functions. It could be seen that the variance is 2.7 and bias as 0.2. The bias is very low compared to individual classifiers which is good and the variance is low as compared to individual naïve bayes classifier. The bias is lower than the previous two ensemble learning techniques.

```
print(var(as.numeric(k_p), as.numeric(as.factor(tstdf_RF$music_genre))))
bias(as.numeric(k_p), as.numeric(as.factor(tstdf_RF$music_genre)))
```

```
[1] 2.753635
0.1286
```

CONCLUSION:

The dataset was used to perform ensemble learning techniques to improve the overall performance. But most of the ensemble learning models performed similar to the individual classifiers in this case. This could be due to a variety of reasons like the data used to train the models may be too simple or not diverse enough, the features are highly corelated, not training the model sufficiently. To improve the performance, we could explore the dataset with extreme data analysis and use that data to train the models. Also I think combining certain classes like Hip Hop and rock will yield sufficient results.

REFERENCES:

http://www.science.smith.edu/~jcrouser/SDS293/labs/lab14-r.html
https://www.geeksforgeeks.org/k-fold-cross-validation-in-r-programming/
https://www.datacamp.com/tutorial/decision-trees-R
https://www.statology.org/how-to-interpret-rmse/
http://inferate.blogspot.com/2015/05/k-fold-cross-validation-with-decision.html
https://www.datatechnotes.com/2018/03/classification-with-gradient-boosting.html