Machine Learning

Professor: Dr. Raman Kannan

Homework 1 : Working with Individual Classifiers

Topic : Music Genre Prediction

Aakash Shetty
aks9108

Outline

1. Overview
2. Preprocessing Recap
3. Libraries Imported
4. Train & Test Split
5. Multinomial Logistic Regression
6. Support Vector Machine
7. Decision Tree
8. KNN
9. Comparison
10. Conclusion

Overview

Let's start with data, we have a 18 attribute columns, of which one is the target column 'music_genre'. In total we have 50,000 rows and 10 different genre classes among them.

We have gone over data some important steps in EDA. One of the point discussed was that it is very important to know the data. EDA has given us a good idea about our data and how it should be preprocessed so that it can be used for modeling and prediction.

Using the preprocessed data from EDA we move forward to making a logistic regression, support vector machine, decision Tree and K -nearest neighbor models. After modeling we will be using various factors to judge and observe the model that has been trained and tested. We will calculate AUC, ROC, confusion matrix, variance and bias for these models for further observations and comparisons.

We will discuss about the models and make comparisons based on observations using the metrics that has been derived from the models. Hopefully we can get some meaningful insights into the models and why it behaves the way it does.

Further, you will see multinomial logistic regression as we have multi class target variable, SVM which performs slightly better than logistic regression because it separates the classes and this reduces the risk of error on the data. Then we have decision tree which performs a little better than logistic regression but falls short when compared to SVM. Finally, we have KNN which is the best at 89% accuracy with k = 211 considering we have 50,000 rows we have considered square root of rows as K value.

Getting poor results in logistic regression, SVM and Decision tree but KNN works really well for the data. Different models works differently on a given dataset.

Preprocessing Recap

- The data used is cleaned by removing NULL values and correcting the schema incase there are data type mismatch. We also remove the outliers that may affect the training.

- Next, we made some changes to the data by encoding the key and mode attribute columns. We change the character values to numeric values for further processing. 'Major' and 'Minor' was mapped to 1 and 0 respectively in Mode while, in keys 'A', 'B', 'C#' etc. was mapped to values from 1-12.

- We also looked at important attributes for training testing in EDA which can be useful in for modeling. We disregard such columns and consider only columns which will be used for modeling. Columns such as song name, artist, popularity and instance id will not be considered for modeling.

- Important features include energy, danceability, instumentalness, loudness etc. In total we have 11 attributes to be considered for predicting a song out of 10 genres.

- We will be using the data from EDA as the input data for train and test spilt.

```
songs <- read.csv('processed_songs.csv', stringsAsFactors = TRUE)
head(songs, 10)
```

A data.frame: 10 × 18

	instance_id	artist_name	track_name	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speec
	<int>	<fct>	<fct>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<int>	
1	32894	Röyksopp	Röyksopp's Night Out	0	0.00468	0.652	-1	0.941	7.92e-01	2	0.1150	-5.201	1	
2	46652	Thievery Corporation	The Shining Path	0	0.01270	0.622	218293	0.890	9.50e-01	6	0.1240	-7.043	1	
3	30097	Dillon Francis	Hurricane	0	0.00306	0.620	215613	0.755	1.18e-02	12	0.5340	-4.617	0	
4	62177	Dubloadz	Nitro	0	0.02540	0.774	166875	0.700	2.53e-03	5	0.1570	-4.498	0	
5	24907	What So Not	Divide & Conquer	0	0.00465	0.638	222369	0.587	9.09e-01	10	0.1570	-6.266	0	
6	43760	Jordan Comolli	Clash	0	0.02890	0.572	214408	0.803	7.74e-06	3	0.1060	-4.294	0	
7	30738	Hraach	Delirio	0	0.02970	0.809	416132	0.706	9.03e-01	11	0.0635	-9.339	1	
8	84950	Kayzo	NEVER ALONE	0	0.00299	0.509	292800	0.921	2.76e-04	9	0.1780	-3.175	1	
9	56950	Shlump	Lazer Beam	0	0.00934	0.578	204800	0.731	1.12e-02	1	0.1110	-7.091	1	

Train & Test Split

We split the train data and test data into 75:25 ratio. All the features will be taken as numeric values.

Split Train and Test Data

```
set.seed(11111)
feats <- names(songs)[c(5:11,13:15,17)]
train_songs <- songs %>%
  mutate_if(is.numeric, scale)

training_songs <- sample(1:nrow(train_songs), nrow(train_songs)*.75, replace = FALSE)
train_set <- train_songs[training_songs, c('music_genre', feats)]
test_set <- train_songs[-training_songs, c('music_genre', feats)]
```

```
feats
```

'acousticness' · 'danceability' · 'duration_ms' · 'energy' · 'instrumentalness' · 'key' · 'liveness' · 'mode' · 'speechiness' · 'tempo' · 'valence'

```
str(train_set)
```

```
'data.frame':   33496 obs. of  12 variables:
 $ music_genre     : Factor w/ 10 levels "Alternative",..: 9 4 3 10 2 3 7 1 8 1 ...
 $ acousticness    : num [1:33496, 1] -0.755 1.925 1.845 -0.794 -0.888 ...
 $ danceability    : num [1:33496, 1] 0.881 -0.444 0.447 -0.601 -1.052 ...
 $ duration_ms     : num [1:33496, 1] -2.0309 0.7316 -0.8379 0.0961 -2.0309 ...
 $ energy          : num [1:33496, 1] 0.345 -1.443 -2.071 1.137 1.027 ...
 $ instrumentalness: num [1:33496, 1] -0.552 -0.417 -0.305 -0.366 -0.552 ...
 $ key             : num [1:33496, 1] -0.968 -1.256 0.476 -1.545 -0.101 ...
 $ liveness        : num [1:33496, 1] -0.3902 -0.6618 -0.5018 0.0254 -0.6023 ...
 $ mode            : num [1:33496, 1] -0.748 1.337 -0.748 -0.748 -0.748 ...
 $ speechiness     : num [1:33496, 1] -0.592 -0.531 -0.544 -0.58 -0.588 ...
 $ tempo           : num [1:33496, 1] 0.324 1.21 0.682 0.147 0.813 ...
 $ valence         : num [1:33496, 1] 0.156 -0.6638 -1.3497 1.337 -0.0754 ...
```

```
str(test_set)
```

```
'data.frame':   11166 obs. of  12 variables:
 $ music_genre     : Factor w/ 10 levels "Alternative",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ acousticness    : num [1:11166, 1] -0.809 -0.847 1.642 -0.894 -0.811 ...
 $ danceability    : num [1:11166, 1] 0.0636 0.5596 0.9936 -1.7005 0.5371 ...
 $ duration_ms     : num [1:11166, 1] -0.0111 0.1931 -2.0309 -0.285 0.8642 ...
 $ energy          : num [1:11166, 1] 0.764 1.103 -0.75 0.323 0.193 ...
 $ instrumentalness: num [1:11166, 1] -0.551 -0.55 -0.44 -0.525 2.1 ...
 $ key             : num [1:11166, 1] -0.968 -0.679 -1.545 1.631 1.053 ...
 $ liveness        : num [1:11166, 1] -0.545 -0.192 -0.13 0.894 -0.508 ...
 $ mode            : num [1:11166, 1] -0.748 -0.748 1.337 -0.748 1.337 ...
 $ speechiness     : num [1:11166, 1] 2.524 -0.291 -0.508 -0.519 -0.536 ...
 $ tempo           : num [1:11166, 1] 0.9769 0.2603 1.1319 0.5048 -0.0676 ...
 $ valence         : num [1:11166, 1] -0.928 -1.411 0.765 -1.398 -1.053 ...
```

Multinomial Logistic Regression

Since we are dealing with a multi-class dataset we cannot simply implement logistic regression we use multinomial logistic regression. We first use variable importance to check the attributes which will be important for prediction and get those attributes for logistic regression model.

```
# importance of variables to be considered
import <-rpart(music_genre~.,data=train_set)
import$variable.importance
```

acousticness: 1735.66381553462 **instrumentalness:** 1002.14509365126 **energy:** 962.311396572617 **speechiness:** 854.054137381861 **danceability:** 473.002067614859 **valence:** 131.44246504108 **tempo:** 82.7116944199987 **duration_ms:** 41.8144194582781 **liveness:** 0.161292432557005

We can see that liveness has a score below 0.5 while others are in 1000's and 100's. This attribute can be dropped while we build our model.

MODEL

```
# multinomial model (only including important attributes from variable importance)
model_lr <- multinom("music_genre~ acousticness + instrumentalness + energy + speechiness + danceability + valence +
                     tempo + duration_ms", data=train_set, MaxNWts =1000000)
```

```
# weights:  100 (81 variable)
initial  value 77127.390275
iter  10 value 59938.066874
iter  20 value 58891.944655
iter  30 value 58497.554255
iter  40 value 57579.153262
iter  50 value 56755.633503
iter  60 value 56626.681246
iter  70 value 56234.394600
iter  80 value 56068.676739
iter  90 value 55594.087335
final  value 55554.454312
converged
```

We train the model for Logistic regression and print out the confusion matrix and the stats for the model.

TRAIN

```
model_lr_train <- predict(object=model_lr, newdata=train_set, type="class")
confusionMatrix(model_lr_train, as.factor(train_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         659   412   200        53     168         277     180   72
  Anime               332   609   392       105      82         229      22  207
  Blues               194   225   985        55     392          83      81  465
  Classical            38   512    77      2604      40          34       3  437
  Country             593   493   704        40    1888         206     243  328
  Electronic          234   326    91        62      19        1633     103  415
  Hip-Hop             292    15    76         2      85         264    1750  214
  Jazz                159   236   378       169     102         251      84 1124
  Rap                 222    28    44         4     120         272     903   60
  Rock                602   524   421        44     471         131      29   66
            Reference
Prediction     Rap Rock
  Alternative  289  407
  Anime         46  263
  Blues         84  507
  Classical      0   74
  Country      305 1037
  Electronic   125  140
  Hip-Hop     1472   56
  Jazz          80  124
  Rap          866  100
  Rock          88  689
```

TRAIN

```
Overall Statistics

               Accuracy : 0.3823
                 95% CI : (0.3771, 0.3876)
    No Information Rate : 0.1014
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3138

 Mcnemar's Test P-Value : < 2.2e-16
```

TRAIN

Statistics by Class:

	Class: Alternative	Class: Anime	Class: Blues
Sensitivity	0.19820	0.18018	0.29246
Specificity	0.93179	0.94428	0.93076
Pos Pred Value	0.24255	0.26629	0.32074
Neg Pred Value	0.91338	0.91121	0.92168
Prevalence	0.09927	0.10091	0.10055
Detection Rate	0.01967	0.01818	0.02941
Detection Prevalence	0.08111	0.06828	0.09168
Balanced Accuracy	0.56499	0.56223	0.61161

	Class: Classical	Class: Country	Class: Electronic
Sensitivity	0.82983	0.56074	0.48314
Specificity	0.95998	0.86893	0.94969
Pos Pred Value	0.68185	0.32345	0.51874
Neg Pred Value	0.98201	0.94653	0.94243
Prevalence	0.09368	0.10052	0.10091
Detection Rate	0.07774	0.05636	0.04875
Detection Prevalence	0.11401	0.17426	0.09398
Balanced Accuracy	0.89490	0.71483	0.71642

	Class: Hip-Hop	Class: Jazz	Class: Rap	Class: Rock
Sensitivity	0.51501	0.33176	0.25812	0.20283
Specificity	0.91774	0.94742	0.94184	0.92106
Pos Pred Value	0.41410	0.41522	0.33066	0.22480
Neg Pred Value	0.94370	0.92647	0.91939	0.91101
Prevalence	0.10144	0.10115	0.10016	0.10142
Detection Rate	0.05225	0.03356	0.02585	0.02057
Detection Prevalence	0.12616	0.08082	0.07819	0.09150
Balanced Accuracy	0.71637	0.63959	0.59998	0.56194

We can now move to testing the model with our test data.

TEST

```
model_lr_class <- predict(model_lr, test_set, type='class')
confusionMatrix(model_lr_class, as.factor(test_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         239   147    56        22      50         94      66   23
  Anime               122   195   123        50      26         65       4   64
  Blues                59    81   313        22     125         20      29  154
  Classical            11   144    31       917      19         12       3  126
  Country             205   164   216        11     610         77      89  139
  Electronic           87   100    28        20       3        494      32  126
  Hip-Hop             103     5    24         2      44         95     550   63
  Jazz                 52    78   127        55      28         68      28  356
  Rap                  88    12    14         2      40         82     309   23
  Rock                202   182   133        10     174         61      11   25
             Reference
Prediction    Rap Rock
  Alternative 112  159
  Anime        14   93
  Blues        18  180
  Classical     2   21
  Country      96  329
  Electronic   27   58
  Hip-Hop     531   12
  Jazz         24   46
  Rap         298   30
  Rock         27  230
```

TEST

```
Overall Statistics

               Accuracy : 0.3763
                 95% CI : (0.3673, 0.3854)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.307

 Mcnemar's Test P-Value : < 2.2e-16
```

TEST

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.20462      0.17599       0.29390
Specificity                     0.92709      0.94422       0.93189
Pos Pred Value                  0.24690      0.25794       0.31269
Neg Pred Value                  0.90890      0.91230       0.92602
Prevalence                      0.10460      0.09923       0.09538
Detection Rate                  0.02140      0.01746       0.02803
Detection Prevalence            0.08669      0.06771       0.08965
Balanced Accuracy               0.56585      0.56011       0.61289
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.82538         0.54513            0.46255
Specificity                   0.96330         0.86802            0.95237
Pos Pred Value                0.71306         0.31508            0.50667
Neg Pred Value                0.98036         0.94485            0.94368
Prevalence                    0.09950         0.10021            0.09565
Detection Rate                0.08212         0.05463            0.04424
Detection Prevalence          0.11517         0.17338            0.08732
Balanced Accuracy             0.89434         0.70657            0.70746
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                0.49063     0.32393     0.25936      0.19862
Specificity                0.91249     0.94974     0.94010      0.91757
Pos Pred Value             0.38488     0.41299     0.33185      0.21801
Neg Pred Value             0.94136     0.92789     0.91712      0.90822
Prevalence                 0.10039     0.09842     0.10290      0.10371
Detection Rate             0.04926     0.03188     0.02669      0.02060
Detection Prevalence       0.12798     0.07720     0.08042      0.09448
Balanced Accuracy          0.70156     0.63683     0.59973      0.55809
```

We now get the AUC for the Model.

AUC

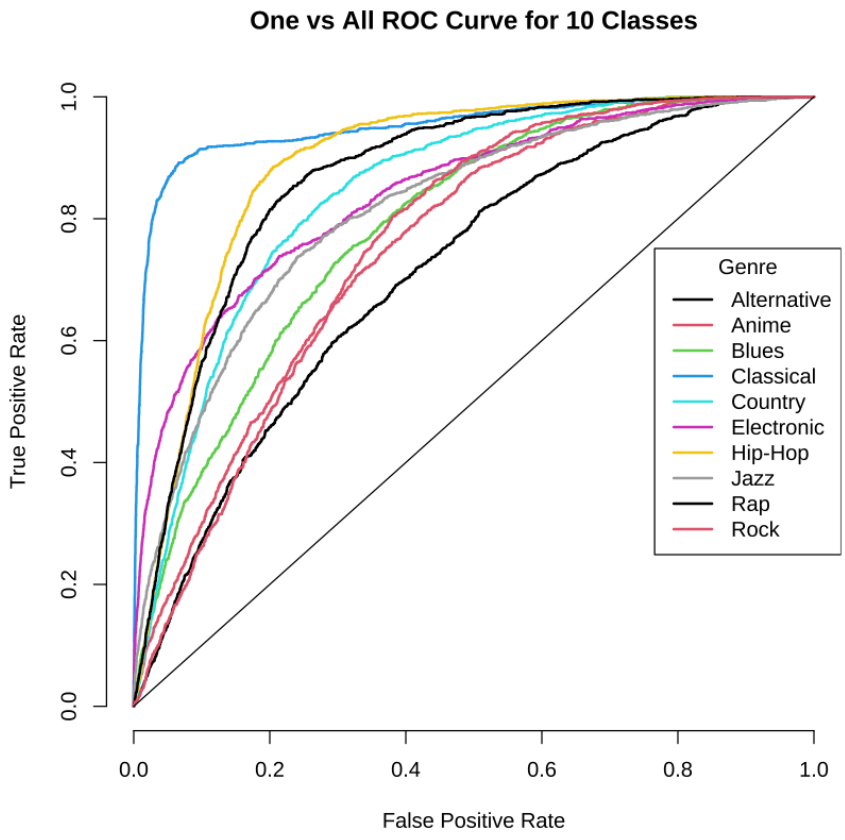
```
model_lr_test <- predict(model_lr, test_set, type='probs')
roc.multi <- multiclass.roc(test_set$music_genre, model_lr_test)
auc(roc.multi)
```

0.822067764506631

Next, Plotting the ROC:

ROC

```
multiclass_roc_plot(test_set,model_lr_test)
```



Variance:

VARIANCE

```
var(as.numeric(model_lr_class), as.numeric(test_set$music_genre))
```

1.22456240912392

Bias:

BIAS

```
bias(as.numeric(model_lr_class), as.numeric(test_set$music_genre))
```

0.00913487372380441

Observation:

Accuracy	37%
AOC	0.822
Variance	1.225
Bias	0.009

Logistic Reg.	Specificity	Precision	Sensitivity
Alternative	0.92709	0.246	0.2046
Anime	0.9442	0.257	0.1759
Blues	0.9318	0.312	0.2939
Classical	0.9633	0.713	0.8253
Country	0.5451	0.315	0.5451
Electronic	0.4625	0.506	0.4625
Hip-Hop	0.9124	0.384	0.4906
Jazz	0.9497	0.412	0.3239
Rap	0.9401	0.331	0.2593
Rock	0.9174	0.217	0.1986

- The accuracy of the model is 37%. So, the model is has more false predictions than the true predictions.
- We have higher variance than bias meaning we have a good tradeoff. A bit higher variance and low bias will give us a more generalized model.
- We can see the specificity, sensitivity and precision table, in general we have high specificity and low recall, precision. This means there are many false predictions in our model.
- Actually, by this observation we can also say that using the opposite of what the classifier outputs will give us a better prediction.

Support Vector Machine

Creating SVM model:

MODEL

```
model_svm <- svm(music_genre~., data = train_set, kernel = "radial", probability = TRUE)
model_svm
```

```
Call:
svm(formula = music_genre ~ ., data = train_set, kernel = "radial",
    probability = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  29640
```

We start with Training:

TRAIN

```
model_svm_train <- predict(model_svm, train_set[feats])
```

```
confusionMatrix(model_svm_train, train_set$music_genre)
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         887   334   278        89     181        241      64   80
  Anime               378  1419   200       119      89        185      25  128
  Blues               215   167  1387        42     352        103      70  430
  Classical            16   444    33      2674       8         17       2  296
  Country             479   322   519        23    2056        118     109  170
  Electronic          210   197   103        47      21       1901      99  327
  Hip-Hop             382    19    86         0     133        280    2366  227
  Jazz                160   190   387       127     108        266      58 1592
  Rap                 240    26    56         3     100        165     553   77
  Rock                358   262   319        14     319        104      52   61
             Reference
Prediction     Rap Rock
  Alternative  197  489
  Anime         66  286
  Blues         72  485
  Classical      0   37
  Country      172  964
  Electronic   118  139
  Hip-Hop     1797   82
  Jazz          49  175
  Rap          795  105
  Rock          89  635
```

TRAIN

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.26677      0.41982       0.41182
Specificity                     0.93527      0.95099       0.93574
Pos Pred Value                  0.31232      0.49016       0.41739
Neg Pred Value                  0.92047      0.93592       0.93435
Prevalence                      0.09927      0.10091       0.10055
Detection Rate                  0.02648      0.04236       0.04141
Detection Prevalence            0.08479      0.08643       0.09921
Balanced Accuracy               0.60102      0.68541       0.67378
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.85214        0.61063            0.56243
Specificity                   0.97190        0.90454            0.95813
Pos Pred Value                0.75815        0.41687            0.60120
Neg Pred Value                0.98452        0.95410            0.95124
Prevalence                    0.09368        0.10052            0.10091
Detection Rate                0.07983        0.06138            0.05675
Detection Prevalence          0.10530        0.14724            0.09440
Balanced Accuracy             0.91202        0.75759            0.76028
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.69629     0.46989    0.23696     0.18693
Specificity                 0.90013     0.94952    0.95604     0.94757
Pos Pred Value              0.44043     0.51157    0.37500     0.28694
Neg Pred Value              0.96331     0.94089    0.91841     0.91171
Prevalence                  0.10144     0.10115    0.10016     0.10142
Detection Rate              0.07064     0.04753    0.02373     0.01896
Detection Prevalence        0.16038     0.09291    0.06329     0.06607
Balanced Accuracy           0.79821     0.70970    0.59650     0.56725
```

Now, Testing:

TEST

```
model_svm_test <- predict(model_svm, test_set[feats])
```

```
confusionMatrix(model_svm_test, test_set$music_genre)
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         271   117    79        28      62         93      39   31
  Anime               120   432    65        58      30         75       7   35
  Blues                76    63   387        11     118         30      12  151
  Classical            10   128    14       920       6          5       1  110
  Country             180   103   150         7     654         40      34   67
  Electronic           93    66    33        14       6        550      38  115
  Hip-Hop             133     6    39         0      50        106     719   73
  Jazz                 56    83   163        64      40         79      18  462
  Rap                  82    12    13         0      38         51     234   28
  Rock                147    98   122         9     115         39      19   27
             Reference
Prediction    Rap Rock
  Alternative  68  199
  Anime        18  122
  Blues        10  183
  Classical     1    8
  Country      65  316
  Electronic   33   55
  Hip-Hop     691   26
  Jazz         17   49
  Rap         214   35
  Rock         32  165
```

TEST

```
Overall Statistics

               Accuracy : 0.4275
                 95% CI : (0.4183, 0.4368)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.364

 Mcnemar's Test P-Value : < 2.2e-16
```

TEST

	Class: Alternative	Class: Anime	Class: Blues
Sensitivity	0.23202	0.38989	0.36338
Specificity	0.92839	0.94731	0.93525
Pos Pred Value	0.27457	0.44906	0.37176
Neg Pred Value	0.91188	0.93375	0.93304
Prevalence	0.10460	0.09923	0.09538
Detection Rate	0.02427	0.03869	0.03466
Detection Prevalence	0.08839	0.08615	0.09323
Balanced Accuracy	0.58020	0.66860	0.64932
	Class: Classical	Class: Country	Class: Electronic
Sensitivity	0.82808	0.58445	0.51498
Specificity	0.97185	0.90425	0.95514
Pos Pred Value	0.76475	0.40470	0.54835
Neg Pred Value	0.98083	0.95131	0.94903
Prevalence	0.09950	0.10021	0.09565
Detection Rate	0.08239	0.05857	0.04926
Detection Prevalence	0.10774	0.14473	0.08983
Balanced Accuracy	0.89997	0.74435	0.73506
	Class: Hip-Hop	Class: Jazz	Class: Rap	Class: Rock
Sensitivity	0.64139	0.42038	0.18625	0.14249
Specificity	0.88810	0.94348	0.95078	0.93925
Pos Pred Value	0.39012	0.44811	0.30269	0.21345
Neg Pred Value	0.95688	0.93715	0.91060	0.90445
Prevalence	0.10039	0.09842	0.10290	0.10371
Detection Rate	0.06439	0.04138	0.01917	0.01478
Detection Prevalence	0.16505	0.09233	0.06332	0.06923
Balanced Accuracy	0.76475	0.68193	0.56852	0.54087

Calculate AUC:

AUC

```
roc_multi <- multiclass.roc(test_set$music_genre, model_svm_prob)
auc(roc_multi)
```

```
0.857123684198962
```

Variance:

VARIANCE

```
var(as.numeric(model_svm_test), as.numeric(test_set$music_genre))
```

```
1.65665118479512
```

Bias:

BIAS

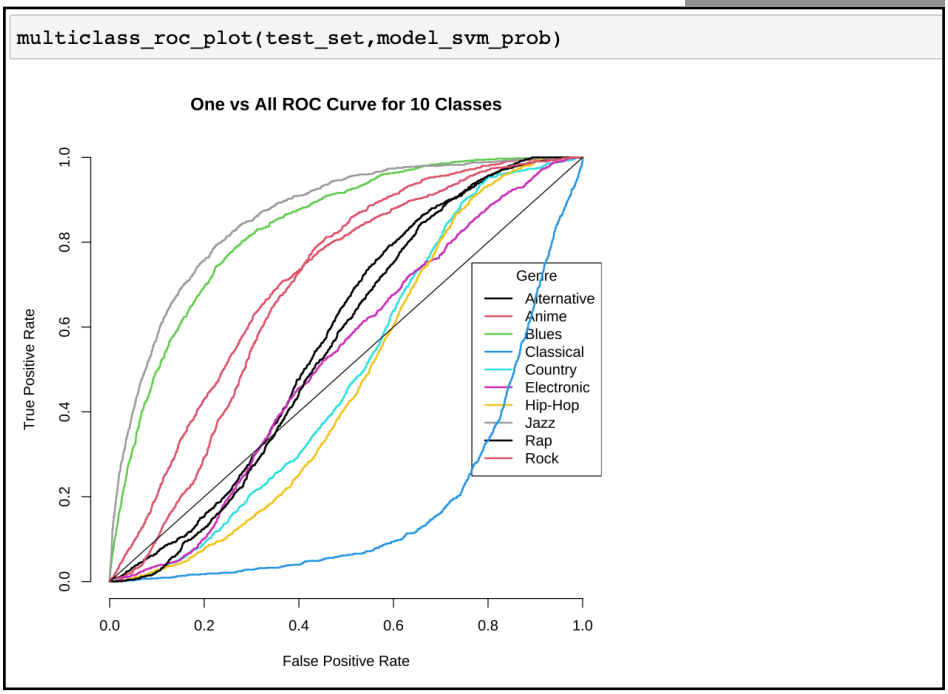
```
bias(as.numeric(model_svm_test), as.numeric(test_set$music_genre))
```

```
-0.125380619738492
```

Plot ROC:

ROC

```
multiclass_roc_plot(test_set,model_svm_prob)
```



Observation:

Accuracy	42%
AOC	0.857
Variance	1.656
Bias	-0.125

Logistic Reg.	Specificity	Precision	Sensitivity
Alternative	0.928	0.274	0.232
Anime	0.947	0.449	0.389
Blues	0.935	0.371	0.363
Classical	0.971	0.764	0.828
Country	0.904	0.404	0.584
Electronic	0.955	0.548	0.514
Hip-Hop	0.888	0.390	0.641
Jazz	0.943	0.448	0.420
Rap	0.950	0.302	0.186
Rock	0.939	0.213	0.142

- A better model than the previous Logistic regression with an accuracy of 42%.

- ROC plot has classical prediction below the diagonal line meaning the prediction for classical is opposite to what is should be.

- Even other genres are falling below diagonals in some area. We have high variance and negative bias meaning there is overfitting and will make for a bad model.

- Again, we have high specificity a low sensitivity and low precision which is not good. This means there are many false predictions in our model.

- We now know that SVMs are not suitable for large datasets because as the datasets become larger the model takes a longer time to train.

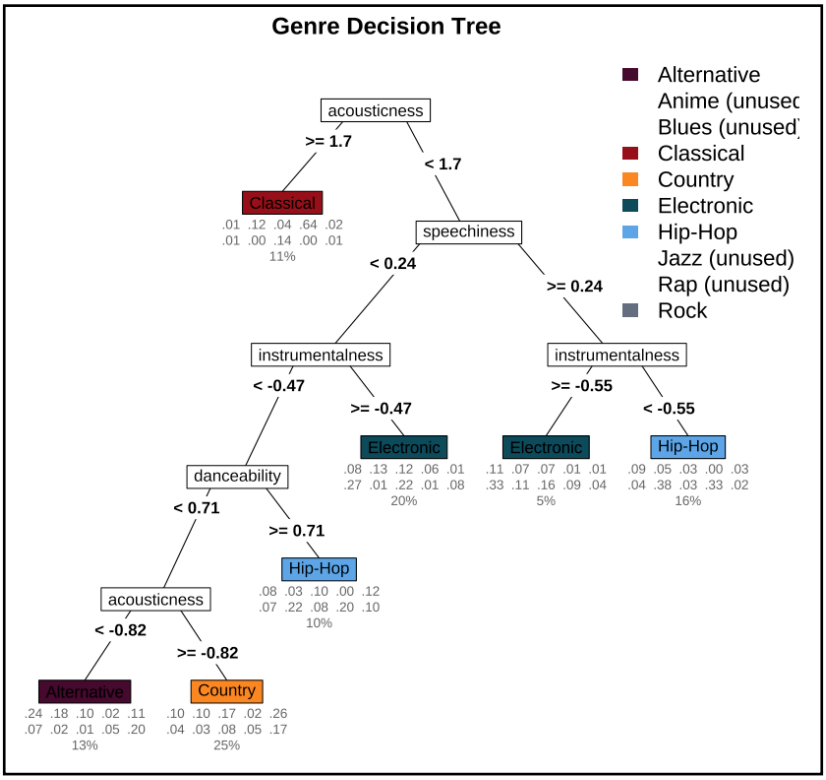
- One of the reasons SVM works better is that it separates the classes and this reduces the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.

Decision Tree

We start with plotting Decision Tree using rpart which will make a decision tree based on attributes which matter the most.

```
set.seed(1111)
model_dt <- rpart(music_genre ~ ., data = train_set)

rpart.plot(model_dt,
           type = 5,
           extra = 104,
           box.palette = list(purple = "#490B32",
               red = "#9A031E",
               orange = '#FB8B24',
               dark_blue = "#0F4C5C",
               blue = "#5DA9E9",
               grey = '#66717E'),
           leaf.round = 0,
           fallen.leaves = FALSE,
           branch = 0.3,
           under = TRUE,
           under.col = 'grey40',
           main = 'Genre Decision Tree',
           tweak = 1.2)
```



Using rpart for skips some classes and gives us unused classes which can adversely affect our model prediction accuracy. So, using ctree function for decision tree helps us get better results.

Decision Tree Training:

TRAIN

```
decision_tree <- predict(model_dt, newdata = train_set)
confusionMatrix(decision_tree, as.factor(train_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         658   296   164        85     134         135      44   86
  Anime               428  1330   281       104     185         159      29  121
  Blues               320   231  1346        48     387         178      98  411
  Classical            14   429    56      2602       9          25       6  280
  Country             367   256   442         7    1911          69      64  123
  Electronic          385   295   181        51      66        2167     228  466
  Hip-Hop             354    43   132         3     187         153    2223  150
  Jazz                203   260   461       202     131         304      84 1564
  Rap                 247    41    76         2     118          86     589   93
  Rock                349   199   229        34     239         104      33   94
             Reference
Prediction     Rap Rock
  Alternative  133  336
  Anime         85  336
  Blues        116  581
  Classical      2   34
  Country       87  866
  Electronic   235  289
  Hip-Hop     1793  129
  Jazz          65  205
  Rap          751  129
  Rock          88  492
```

TRAIN

```
Overall Statistics

               Accuracy : 0.4491
                 95% CI : (0.4438, 0.4545)
    No Information Rate : 0.1014
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3879

 Mcnemar's Test P-Value : < 2.2e-16
```

TRAIN

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.19789      0.39349      0.39964
Specificity                     0.95317      0.94262      0.92134
Pos Pred Value                  0.31772      0.43492      0.36222
Neg Pred Value                  0.91513      0.93265      0.93210
Prevalence                      0.09927      0.10091      0.10055
Detection Rate                  0.01964      0.03971      0.04018
Detection Prevalence            0.06183      0.09129      0.11094
Balanced Accuracy               0.57553      0.66806      0.66049
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.82919         0.56757           0.64112
Specificity                   0.97184         0.92429           0.92708
Pos Pred Value                0.75268         0.45587           0.49668
Neg Pred Value                0.98216         0.95031           0.95836
Prevalence                    0.09368         0.10052           0.10091
Detection Rate                0.07768         0.05705           0.06469
Detection Prevalence          0.10321         0.12515           0.13025
Balanced Accuracy             0.90051         0.74593           0.78410
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.65421     0.46163    0.22385     0.14483
Specificity                 0.90219     0.93640    0.95418     0.95452
Pos Pred Value              0.43023     0.44955    0.35225     0.26437
Neg Pred Value              0.95852     0.93923    0.91697     0.90817
Prevalence                  0.10144     0.10115    0.10016     0.10142
Detection Rate              0.06637     0.04669    0.02242     0.01469
Detection Prevalence        0.15426     0.10386    0.06365     0.05556
Balanced Accuracy           0.77820     0.69901    0.58901     0.54968
```

Decision Tree Testing:

TEST

```
predict_model<-predict(model_dt, test_set)
confusionMatrix(predict_model, as.factor(test_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         205   128    62         36      47          67      18   44
  Anime               150   371    95         53      68          62       8   38
  Blues               118    87   347         22     133          71      32  156
  Classical             9   142    29        892       8           4       2  112
  Country             131    83   142          6     575          26      25   42
  Electronic          141   108    64         15      24         587      73  178
  Hip-Hop             136    13    43          0      75          65     703   46
  Jazz                 85    91   166         79      48         112      29  415
  Rap                  80    22    30          0      46          38     215   33
  Rock                113    63    87          8      95          36      16   35
             Reference
Prediction    Rap Rock
  Alternative  37  135
  Anime        22  153
  Blues        40  187
  Classical     2   11
  Country      35  270
  Electronic   75   96
  Hip-Hop     665   49
  Jazz         26   71
  Rap         210   41
  Rock         37  145
```

TEST

```
Overall Statistics

               Accuracy : 0.3985
                 95% CI : (0.3894, 0.4077)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3321

 Mcnemar's Test P-Value : < 2.2e-16
```

TEST

```
                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.17551      0.33484      0.32582
Specificity                     0.94259      0.93547      0.91625
Pos Pred Value                  0.26316      0.36373      0.29086
Neg Pred Value                  0.90729      0.92736      0.92801
Prevalence                      0.10460      0.09923      0.09538
Detection Rate                  0.01836      0.03323      0.03108
Detection Prevalence            0.06977      0.09135      0.10684
Balanced Accuracy               0.55905      0.63516      0.62103
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.80288          0.5139           0.54963
Specificity                   0.96827          0.9244           0.92335
Pos Pred Value                0.73658          0.4307           0.43130
Neg Pred Value                0.97800          0.9447           0.95094
Prevalence                    0.09950          0.1002           0.09565
Detection Rate                0.07989          0.0515           0.05257
Detection Prevalence          0.10845          0.1196           0.12189
Balanced Accuracy             0.88558          0.7191           0.73649
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.62712     0.37762    0.18277     0.12522
Specificity                 0.89129     0.92977    0.94959     0.95104
Pos Pred Value              0.39164     0.36988    0.29371     0.22835
Neg Pred Value              0.95539     0.93190    0.91015     0.90381
Prevalence                  0.10039     0.09842    0.10290     0.10371
Detection Rate              0.06296     0.03717    0.01881     0.01299
Detection Prevalence        0.16076     0.10048    0.06403     0.05687
Balanced Accuracy           0.75920     0.65369    0.56618     0.53813
```

Calculate AUC:

AUC

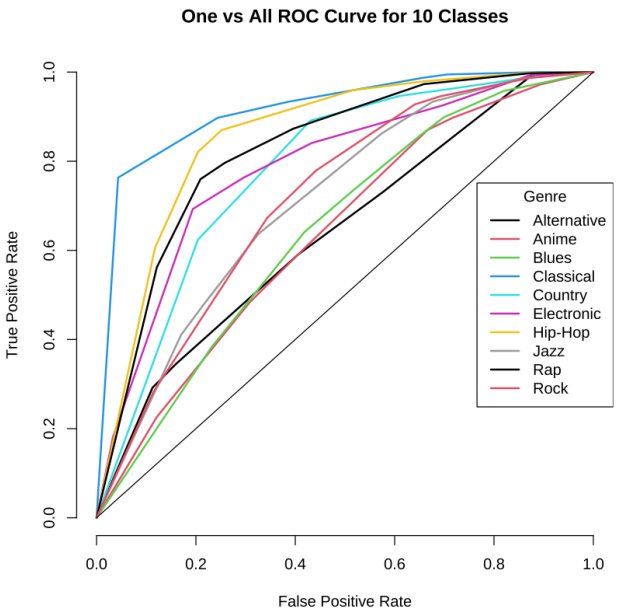
```
model_dt_prob <- predict(model_dt,select(test_set,all_of(feats)), type="prob")
roc_dt <- multiclass.roc(test_set$music_genre, model_dt_prob)
auc(roc_dt)
```

0.751355554354379

Plot ROC:

ROC

```
multiclass_roc_plot(test_set, model_dt_prob)
```



Variance:

VARIANCE

```
var(as.numeric(predict_model), as.numeric(test_set$music_genre))
```

1.71415633104751

Bias:

BIAS

```
bias(as.numeric(predict_model), as.numeric(test_set$music_genre))
```

-0.105409278165861

Observation:

Accuracy	39.8%
AOC	0.751
Variance	1.71
Bias	-0.105

Logistic Reg.	Specificity	Precision	Sensitivity
Alternative	0.942	0.263	0.175
Anime	0.935	0.363	0.334
Blues	0.916	0.290	0.325
Classical	0.968	0.736	0.802
Country	0.924	0.430	0.513
Electronic	0.923	0.431	0.549
Hip-Hop	0.891	0.391	0.627
Jazz	0.929	0.369	0.377
Rap	0.949	0.293	0.182
Rock	0.951	0.228	0.125

- Decision Trees split the target values based on attribute value test. The recursive partitioning is done until the splitting no longer add value.

- We see low accuracy in predicting target variables from the tree which can be due to the lack of relevant attributes to predict it.

- High sensitivity, low precision and low specificity indicates that the model is has many false predictions which affect the accuracy of the model.

K-Nearest Neighbor

Give that we have categorical character values as target variables so we have to covert all values to numeric values. To compute the distance we will have convert all columns to numeric value.

```
train_knn <- train_set
test_knn <- test_set

#Convert all columns to numeric
index <- 1:ncol(train_knn)

train_knn[ , index] <- lapply(train_knn[ , index], as.numeric)
test_knn[ , index] <- lapply(test_knn[ , index], as.numeric)
```

Training & Testing KNN model:

```
model_knn <- knn(train_knn, test_knn, cl = train_knn$music_genre, k = 211)

confusionMatrix(model_knn, as.factor(test_knn$music_genre))
```

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4    5    6    7    8    9   10
        1  1129   66    3    0    0    0    0    0    0    0
        2    29  947   69    5    0    0    0    0    0    0
        3    10   71  964   68    7    0    0    0    0    0
        4     0   24   29  983    7    9    0    0    0    0
        5     0    0    0   45 1048  150    1    0    0    0
        6     0    0    0    9   49  786   22    7    0    0
        7     0    0    0    1    8  120 1087   72    0    0
        8     0    0    0    0    0    3   11  936   17   10
        9     0    0    0    0    0    0    0   81 1013   31
        10    0    0    0    0    0    0    0    3  119 1117
```

```
Overall Statistics

               Accuracy : 0.8965
                 95% CI : (0.8907, 0.9021)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8849

 Mcnemar's Test P-Value : NA
```

```
Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
Sensitivity            0.9666  0.85469  0.90516  0.88479  0.93655  0.73596
Specificity            0.9931  0.98976  0.98456  0.99314  0.98049  0.99138
Pos Pred Value         0.9424  0.90190  0.86071  0.93441  0.84244  0.90034
Neg Pred Value         0.9961  0.98408  0.98995  0.98734  0.99284  0.97260
Prevalence             0.1046  0.09923  0.09538  0.09950  0.10021  0.09565
Detection Rate         0.1011  0.08481  0.08633  0.08804  0.09386  0.07039
Detection Prevalence   0.1073  0.09404  0.10030  0.09421  0.11141  0.07818
Balanced Accuracy      0.9799  0.92223  0.94486  0.93896  0.95852  0.86367
                     Class: 7 Class: 8 Class: 9 Class: 10
Sensitivity           0.96967  0.85168  0.88164     0.9646
Specificity           0.97999  0.99593  0.98882     0.9878
Pos Pred Value        0.84394  0.95803  0.90044     0.9015
Neg Pred Value        0.99656  0.98400  0.98646     0.9959
Prevalence            0.10039  0.09842  0.10290     0.1037
Detection Rate        0.09735  0.08383  0.09072     0.1000
Detection Prevalence  0.11535  0.08750  0.10075     0.1110
Balanced Accuracy     0.97483  0.92381  0.93523     0.9762
```

Variance:

```
var(as.numeric(model_knn), as.numeric(test_knn$music_genre))
```

8.38638626840372

Bias:

```
bias(as.numeric(model_knn), as.numeric(test_knn$music_genre))
```

0.00761239476983701

Observation:

Accuracy	89.6%
Variance	8.386
Bias	0.0076

Logistic Reg.	Specificity	Precision	Sensitivity
Alternative	0.993	0.942	0.966
Anime	0.989	0.901	0.854
Blues	0.984	0.860	0.905
Classical	0.993	0.934	0.884
Country	0.980	0.842	0.936
Electronic	0.991	0.900	0.735
Hip-Hop	0.979	0.843	0.969
Jazz	0.995	0.958	0.851
Rap	0.988	0.900	0.881
Rock	0.987	0.901	0.964

- The KNN-based classifier does not build any classification model. It directly learns from the training observations. It starts processing data only after it is given a test observation to classify.

- KNN however tries to create clusters and then assign them the classes, which gives us better results. All classes were mapped to numeric values.

- With a 89% accuracy this is the best model so far. ROC and AUC cannot be possible.

Comparison

Model	*Accuracy*	*AOC*	*Variance*	*Bias*
Logistic Reg.	37%	0.822	1.225	0.009
SVM	42%	0.857	1.656	-0.125
Decision Tree	39.8%	0.751	1.71	-0.105
KNN	89.6%	NA	8.386	0.0076

- As you will see multinomial logistic regression as we have multi class target variable. Crates a hyperplane for prediction. It can have different decision boundaries with different weights that are near the optimal point.

- SVM which performs slightly better than logistic regression because it separates the classes and this reduces the risk of error on the data. Also uses line and hyperplane for classification.

- Then we have decision tree which performs a little better than logistic regression but falls short when compared to SVM. Decision tree predicts the attributes that will be most helpful for classification and creates a decision tree based on these assumptions. However, it falls short of SVM and KNN.

- Finally, we have ***KNN*** which is the best at ***89.6%*** accuracy with ***k = 211*** considering we have 50,000 rows we have considered square root of rows as K value. Unlike decision tree, KNN does not make any underlying assumption about the data.

Conclusion

- We have taken a step further from EDA and moved to applying models and training out model on the data. From the observations and the experiments it is evident that different models works uniquely on different data.

- Also, we could see that KNN worked best for our data and the computation time was very reasonable. This goes to show that KNN does really work well on multi class datasets.

- SVM model took time to train due its complexity and slower run on large datasets. It also shows that SVM does not do well with large datasets in general. Even though it did a little better than decision tree it was not worth the time taken.

- Overall, it was very helpful to start with EDA before moving on to model training and classification as it made the process easier to understand and made problem easier to solve. Knowing the data in hand was very beneficial for classification.