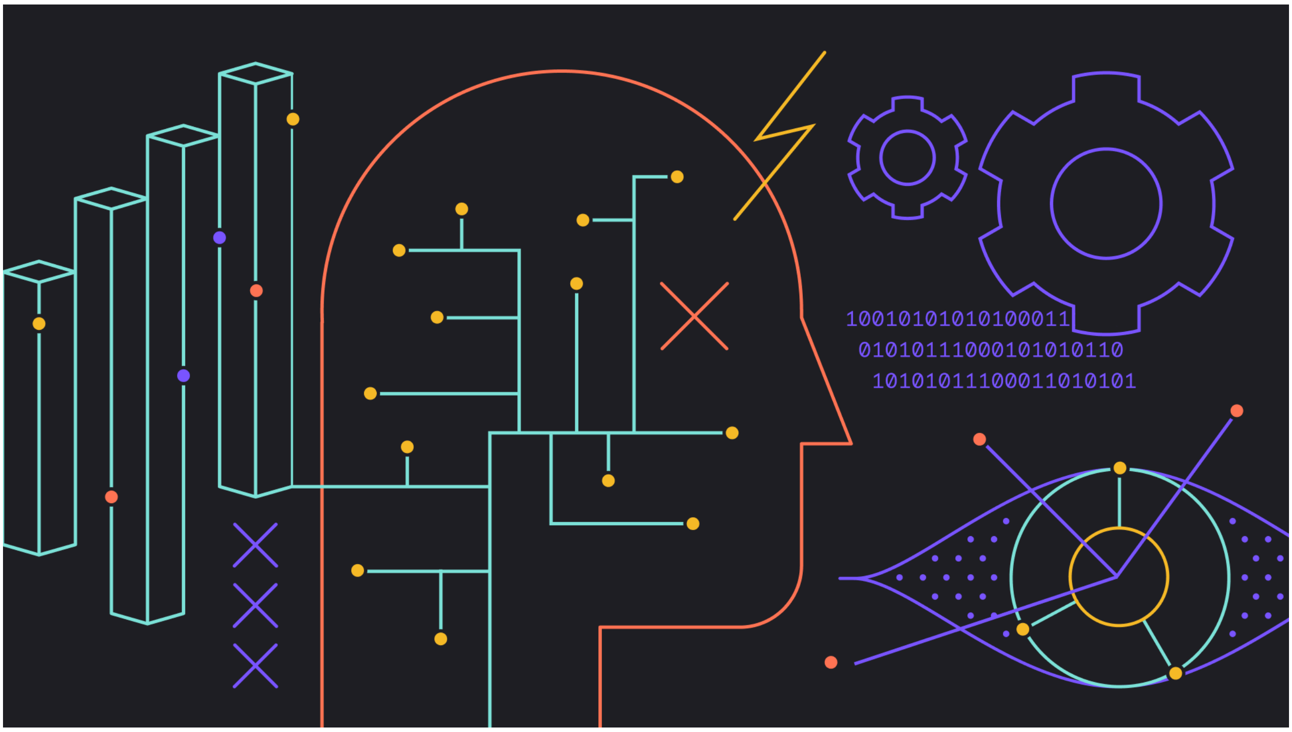
Machine Learning

Professor: Dr. Raman Kannan



Homework 2 : Ensemble Learning

Topic : Music Genre Prediction

Aakash Shetty
aks9108

Outline

1. Overview
2. Preprocessing Recap
3. Train & Test Split
4. Random Forest
5. Boosting
6. Bagging
7. Comparison
8. What does cross validation do to bias and variance?
9. Conclusion

Overview

Let's start with data, we have a 18 attribute columns, of which one is the target column 'music_genre'. In this homework we will be going through three ensemble techniques. Random Forest, Boosting and Bagging.

Given these 3 techniques we will try to compare them with each other based on accuracy, variance and bias. Variance is higher for over-fitting and bias will be higher for under-fitting.

Based on what we know random forest and bagging technique will try to reduce variance while boosting will try to reduce the bias. Let's see if we can see the difference in actual data we have worked with.

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 ...
```

Random Forest

Random forest is an ensemble model using bagging as the ensemble method and decision tree as the individual model. One random subset is used to train one decision tree. The optimal splits for each decision tree are based on a random subset of features. Each individual tree predicts the records in the test set, independently.

MODEL

```
songs_rf <- randomForest(music_genre~., data = train_set, mtry = 4)

pred_train <- predict(songs_rf)
pred_test <- predict(songs_rf, test_set)
```

TRAIN

```
confusionMatrix(pred_train, as.factor(train_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         658   304   233        94     145        198     146   87
  Anime               348  1577   207       156     159        183      30  125
  Blues               292   178  1344        50     323        157      59  479
  Classical             9   381    39      2580       5         12       3  241
  Country             343   264   422        11    1878         63      75  122
  Electronic          280   223   151        55      32       2062     130  435
  Hip-Hop             328    31    85         4     136        152    1335  137
  Jazz                204   200   471       169     126        370      61 1625
  Rap                 304    38    53         0     136         86    1526   82
  Rock                559   184   363        19     427         97      33   55
             Reference
Prediction     Rap Rock
  Alternative  243  636
  Anime         65  298
  Blues         37  606
  Classical      0   27
  Country      124  827
  Electronic   120  179
  Hip-Hop     1914   81
  Jazz          58  189
  Rap          691  119
  Rock         103  435
```

TRAIN

```
Overall Statistics

               Accuracy : 0.4235
                 95% CI : (0.4182, 0.4288)
    No Information Rate : 0.1014
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3594

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

Random Forest Random Forest Random Forest

TRAIN

Statistics by Class:

	Class: Alternative	Class: Anime	Class: Blues
Sensitivity	0.19789	0.46657	0.39905
Specificity	0.93086	0.94784	0.92761
Pos Pred Value	0.23980	0.50095	0.38128
Neg Pred Value	0.91327	0.94059	0.93247
Prevalence	0.09927	0.10091	0.10055
Detection Rate	0.01964	0.04708	0.04012
Detection Prevalence	0.08192	0.09398	0.10524
Balanced Accuracy	0.56438	0.70720	0.66333

	Class: Classical	Class: Country	Class: Electronic
Sensitivity	0.82218	0.55777	0.61006
Specificity	0.97638	0.92529	0.94671
Pos Pred Value	0.78253	0.45483	0.56231
Neg Pred Value	0.98152	0.94930	0.95581
Prevalence	0.09368	0.10052	0.10091
Detection Rate	0.07702	0.05607	0.06156
Detection Prevalence	0.09843	0.12327	0.10948
Balanced Accuracy	0.89928	0.74153	0.77838

	Class: Hip-Hop	Class: Jazz	Class: Rap	Class: Rock
Sensitivity	0.39288	0.47963	0.20596	0.12805
Specificity	0.90471	0.93862	0.92223	0.93887
Pos Pred Value	0.31763	0.46790	0.22768	0.19121
Neg Pred Value	0.92957	0.94128	0.91254	0.90513
Prevalence	0.10144	0.10115	0.10016	0.10142
Detection Rate	0.03986	0.04851	0.02063	0.01299
Detection Prevalence	0.12548	0.10368	0.09061	0.06792
Balanced Accuracy	0.64879	0.70913	0.56410	0.53346

TEST

```
confusionMatrix(pred_test, as.factor(test_set$music_genre))
```

Confusion Matrix and Statistics

Prediction \ Reference	Alternative	Anime	Blues	Classical	Country	Electronic	Hip-Hop	Jazz	Rap	Rock
Alternative	247	100	60	33	50	62	60	33	70	249
Anime	114	546	66	65	48	64	9	45	19	99
Blues	87	62	410	15	111	48	14	159	10	195
Classical	4	97	17	912	3	3	0	85	1	11
Country	122	82	115	7	626	22	19	46	44	262
Electronic	102	83	44	11	7	635	37	129	37	70
Hip-Hop	131	12	33	0	48	49	453	55	682	28
Jazz	66	79	185	63	28	118	23	514	18	61
Rap	106	8	14	0	52	29	496	18	231	37
Rock	189	39	121	5	146	38	10	15	37	146

TEST

Overall Statistics

```
               Accuracy : 0.4227
                 95% CI : (0.4135, 0.4319)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3587

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

TEST

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.21147      0.49278      0.38498
Specificity                     0.92829      0.94741      0.93060
Pos Pred Value                  0.25622      0.50791      0.36904
Neg Pred Value                  0.90972      0.94431      0.93486
Prevalence                      0.10460      0.09923      0.09538
Detection Rate                  0.02212      0.04890      0.03672
Detection Prevalence            0.08633      0.09627      0.09950
Balanced Accuracy               0.56988      0.72009      0.65779
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.82088        0.55943           0.59457
Specificity                   0.97802        0.92844           0.94850
Pos Pred Value                0.80494        0.46543           0.54978
Neg Pred Value                0.98017        0.94980           0.95675
Prevalence                    0.09950        0.10021           0.09565
Detection Rate                0.08168        0.05606           0.05687
Detection Prevalence          0.10147        0.12045           0.10344
Balanced Accuracy             0.89945        0.74393           0.77154
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.40410     0.46770    0.20104     0.12608
Specificity                 0.89667     0.93633    0.92413     0.94005
Pos Pred Value              0.30382     0.44502    0.23310     0.19571
Neg Pred Value              0.93096     0.94156    0.90978     0.90288
Prevalence                  0.10039     0.09842    0.10290     0.10371
Detection Rate              0.04057     0.04603    0.02069     0.01308
Detection Prevalence        0.13353     0.10344    0.08875     0.06681
Balanced Accuracy           0.65038     0.70201    0.56259     0.53306
```

VARIANCE & BIAS

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

1.63257517001703

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

-0.0802435966326348

Observation:

Accuracy	42%
Variance	1.632
Bias	-0.08

- We get a low accuracy of 42% with reasonable variance and bias.

- The data still continues to show low accuracy with models which goes to show that the data would need a more complex model to get good results like a neural network or transformer.

Boosting

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially, each model tries to compensate for the weaknesses of its predecessor. We have used XGBoost.

MODEL

```r
matrix_train_gb <- xgb.DMatrix(data = as.matrix(train_set[,-1]), label = as.integer(as.factor(train_set[,1])))
matrix_test_gb <- xgb.DMatrix(data = as.matrix(test_set[,-1]), label = as.integer(as.factor(test_set[,1])))

model_gb <- xgboost(data = matrix_train_gb,
                    nrounds = 50,
                    verbose = FALSE,
                    params = list(objective = "multi:softmax",
                                  num_class = 10 + 1))

predict_gb_one <- predict(model_gb, matrix_test_gb)
predict_gb <- levels(as.factor(test_set$music_genre))[predict_gb_one]
```

TEST

```r
confusionMatrix(as.factor(predict_gb), as.factor(test_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         218    96    68        25      40         59      30   20
  Anime               124   543    55        75      37         63       9   48
  Blues                92    51   432        15     124         48      15  168
  Classical            12   102    19       914       5          6       1   83
  Country             136    89   125         6     641         23      19   41
  Electronic          115    79    37        11       9        632      33  118
  Hip-Hop             118     4    28         0      48         54     509   48
  Jazz                 79    74   175        56      27        104      28  526
  Rap                 100    14    19         0      50         35     462   23
  Rock                174    56   107         9     138         44      15   24
             Reference
Prediction    Rap Rock
  Alternative  68  168
  Anime        21  120
  Blues        14  211
  Classical     2   10
  Country      42  265
  Electronic   33   78
  Hip-Hop     631   28
  Jazz         24   67
  Rap         286   31
  Rock         28  180
```

TEST

```
Overall Statistics

               Accuracy : 0.4371
                 95% CI : (0.4279, 0.4464)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3748

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

TEST

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.18664      0.49007      0.40563
Specificity                     0.94259      0.94512      0.92694
Pos Pred Value                  0.27525      0.49589      0.36923
Neg Pred Value                  0.90842      0.94390      0.93667
Prevalence                      0.10460      0.09923      0.09538
Detection Rate                  0.01952      0.04863      0.03869
Detection Prevalence            0.07093      0.09807      0.10478
Balanced Accuracy               0.56462      0.71760      0.66629
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.82268        0.57283           0.59176
Specificity                   0.97613        0.92575           0.94920
Pos Pred Value                0.79203        0.46215           0.55197
Neg Pred Value                0.98032        0.95112           0.95649
Prevalence                    0.09950        0.10021           0.09565
Detection Rate                0.08186        0.05741           0.05660
Detection Prevalence          0.10335        0.12422           0.10254
Balanced Accuracy             0.89941        0.74929           0.77048
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.45406     0.47862    0.24891     0.15544
Specificity                 0.90453     0.93702    0.92672     0.94055
Pos Pred Value              0.34673     0.45345    0.28039     0.23226
Neg Pred Value              0.93689     0.94273    0.91494     0.90588
Prevalence                  0.10039     0.09842    0.10290     0.10371
Detection Rate              0.04558     0.04711    0.02561     0.01612
Detection Prevalence        0.13147     0.10389    0.09135     0.06941
Balanced Accuracy           0.67929     0.70782    0.58782     0.54799
```

VARIANCE & BIAS

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

1.85823707196347

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

-0.0167472684936414

Observation:

Accuracy	44%
Variance	1.86
Bias	-0.016

- We get a low accuracy of 43% with reasonable variance and negative bias.

- The data still continues to show low accuracy with models.

- Variance is higher than random forest , showing a more overfitting.

- XGBoost does not need normalized features and work well if the data is nonlinear, non-monotonic, or with segregated clusters.

Bagging

Bagging is an ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement, meaning that the individual data points can be chosen more than once.

MODEL

```r
gbag <- bagging(music_genre ~ ., data = train_set, coob=TRUE)
predict_bag <- predict(gbag, newdata=test_set)
```

TEST

```r
confusionMatrix(as.factor(predict_bag), as.factor(test_set$music_genre))
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
  Alternative         214   100    67        28      71         78      66   37
  Anime               116   516    75        87      51         58      11   40
  Blues                85    59   363        23     122         55      16  157
  Classical             9    91    14       886       2          2       1   82
  Country             117    88   120         6     561         24      15   41
  Electronic          100    83    50        11      11        592      44  137
  Hip-Hop             133    17    31         0      50         49     390   52
  Jazz                 56    76   175        61      24        138      22  483
  Rap                  99     7    20         2      55         29     542   31
  Rock                239    71   150         7     172         43      14   39
             Reference
Prediction    Rap Rock
  Alternative  70  246
  Anime        25  123
  Blues        15  183
  Classical     0    5
  Country      41  240
  Electronic   37   70
  Hip-Hop     649   27
  Jazz         20   56
  Rap         246   43
  Rock         46  165
```

TEST

```
Overall Statistics

               Accuracy : 0.3955
                 95% CI : (0.3864, 0.4046)
    No Information Rate : 0.1046
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3284

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

Bagging Bagging Bagging Bagging Bagging

TEST

```
Statistics by Class:

                     Class: Alternative Class: Anime Class: Blues
Sensitivity                     0.18322      0.46570      0.34085
Specificity                     0.92368      0.94174      0.92921
Pos Pred Value                  0.21904      0.46824      0.33673
Neg Pred Value                  0.90637      0.94118      0.93041
Prevalence                      0.10460      0.09923      0.09538
Detection Rate                  0.01917      0.04621      0.03251
Detection Prevalence            0.08750      0.09869      0.09654
Balanced Accuracy               0.55345      0.70372      0.63503
                     Class: Classical Class: Country Class: Electronic
Sensitivity                   0.79748        0.50134           0.55431
Specificity                   0.97951        0.93112           0.94623
Pos Pred Value                0.81136        0.44773           0.52159
Neg Pred Value                0.97767        0.94371           0.95255
Prevalence                    0.09950        0.10021           0.09565
Detection Rate                0.07935        0.05024           0.05302
Detection Prevalence          0.09780        0.11222           0.10165
Balanced Accuracy             0.88850        0.71623           0.75027
                     Class: Hip-Hop Class: Jazz Class: Rap Class: Rock
Sensitivity                 0.34790     0.43949    0.21410     0.14249
Specificity                 0.89965     0.93762    0.91734     0.92196
Pos Pred Value              0.27897     0.43474    0.22905     0.17442
Neg Pred Value              0.92516     0.93874    0.91052     0.90284
Prevalence                  0.10039     0.09842    0.10290     0.10371
Detection Rate              0.03493     0.04326    0.02203     0.01478
Detection Prevalence        0.12520     0.09950    0.09618     0.08472
Balanced Accuracy           0.62378     0.68855    0.56572     0.53222
```

VARIANCE & BIAS

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

1.44233802971226

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

0.00644814615797958

Observation:

Accuracy	39%
Variance	1.4
Bias	0.006

- We get a low accuracy of 39% with lower variance and higher bias compared to the other two models.
- This can be because the model tries to generalize the data and but it fails to train well enough with given techniques.

Comparison

Model	*Accuracy*	*Variance*	*Bias*
Random Forest	42%	1.632	-0.08
Boosting	44%	1.86	-0.016
Bagging	39%	1.4	0.006

- XGBoost has the best accuracy but still fails to give promising results for the dataset.
- Bagging will try to decrease variance as you can see bagging technique has lower variance.
- Boosting will try to reduce the Bias mainly thus giving us lower bias than other bagging techniques.

What does cross validation do to bias and variance?

Cross Validation techniques reduce over-fitting, it reduces the variance while also trying to reduce the bias. However, we know about the tradeoff between variance and bias. As the bias increase, the variance reduce and vis-a-versa.

Conclusion

As discussed in the overview we can see the difference in the results for the given techniques. While bagging techniques have lower variance and higher bias, boosting technique has higher bias and lower variance in comparison. Ensemble techniques seem to work better than many models implemented in HW1.