CSCI B 565 Data Mining Bonus Homework

Ganesh Nagarajan, gnagaraj@indiana.edu

December 17, 2015

All work herein is solely mine.

Solutions

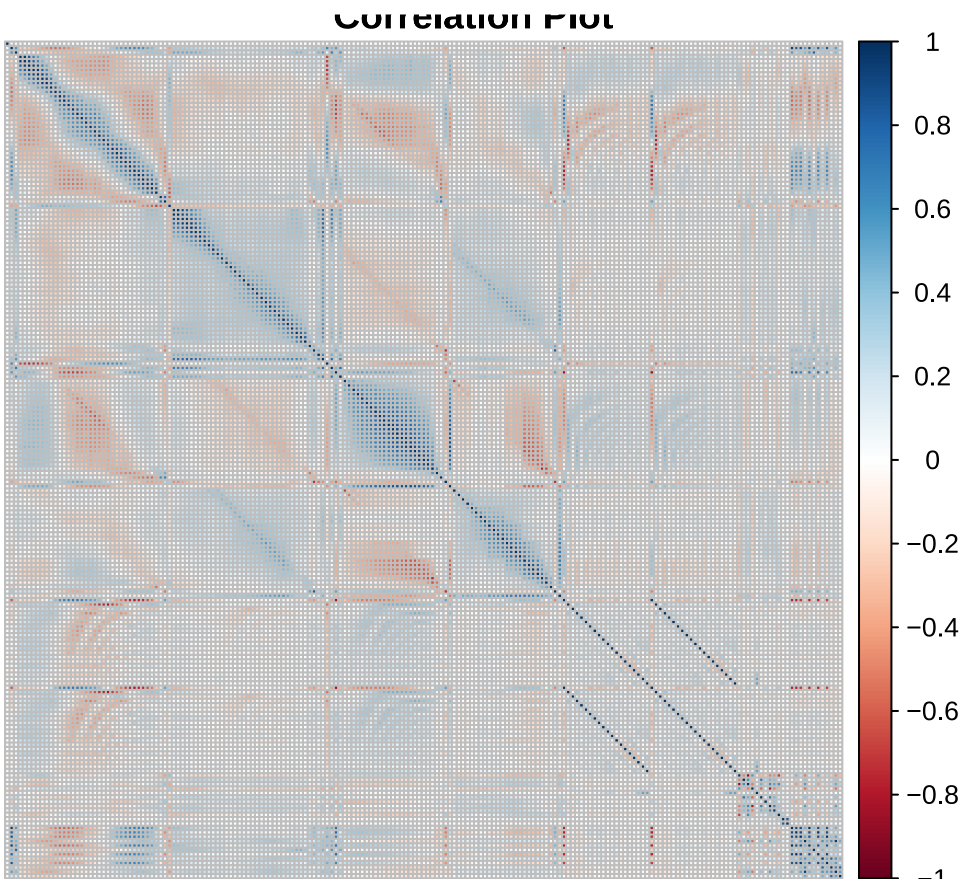
1. Relative Frequencies of these labels

```
# Load the Train Dataset to R
tuneds <- read.csv("genresTrain.csv")
# Create a Frequency Table with the Genre
table(tuneds[,192])
```

```
##
##     Blues Classical      Jazz     Metal       Pop      Rock
##      1596      3444      3003       924      1575      1953
```

2. Correlation of these features

```
library(corrplot)
num.tuneds<- tuneds[-192]
# Find Correlation
cor.tune <- cor(num.tuneds)
# Plot Correlation
corrplot(cor.tune,method = c("square"),title = "Correlation Plot",tl.pos = c("n"))
```




```
cor.upper.tune <- cor.tune
# Make the Lower triangular matrix of the correlation matrix 0
cor.upper.tune[lower.tri(cor.upper.tune)] <- 0
# Since the diagonals are always correlated, make them 0
diag(cor.upper.tune) <- 0
#Find the labels which have a perfect correlation
cno <- which(cor.upper.tune == 1,arr.ind = TRUE)
print("Elements which have perfect correlation")
```

```
## [1] "Elements which have perfect correlation"
```

```
cno
```

```
##             row col
## PAR_MFCC2   129 149
## PAR_MFCC3   130 150
## PAR_MFCC4   131 151
## PAR_MFCC5   132 152
## PAR_MFCC6   133 153
## PAR_MFCC7   134 154
## PAR_MFCC8   135 155
## PAR_MFCC9   136 156
## PAR_MFCC10 137 157
## PAR_MFCC11 138 158
## PAR_MFCC12 139 159
```

```
## PAR_MFCC15 142 162
## PAR_MFCC16 143 163
## PAR_MFCC17 144 164
## PAR_MFCC18 145 165
## PAR_MFCC19 146 166
## PAR_MFCC20 147 167
```

```r
cno1 <- which(cor.upper.tune > 0.9,arr.ind = TRUE)
print("Elements which have correlation with > 0.9")
```

```
## [1] "Elements which have correlation with > 0.9"
```

```r
cno1
```

```
##             row col
## PAR_ASE1      4   5
## PAR_ASE2      5   6
## PAR_ASE3      6   7
## PAR_ASE4      7   8
## PAR_ASE5      8   9
## PAR_ASE6      9  10
## PAR_ASE29    32  33
## PAR_ASE30    33  34
## PAR_ASEV1    39  40
## PAR_ASEV2    40  41
## PAR_ASEV3    41  42
## PAR_ASEV4    42  43
## PAR_ASEV5    43  44
## PAR_SFM11    88  89
## PAR_SFM12    89  90
## PAR_SFM13    90  91
## PAR_SFM14    91  92
## PAR_SFM15    92  93
## PAR_SFM18    95  96
## PAR_SFM19    96  97
## PAR_MFCC1   128 148
## PAR_MFCC2   129 149
## PAR_MFCC3   130 150
## PAR_MFCC4   131 151
## PAR_MFCC5   132 152
## PAR_MFCC6   133 153
## PAR_MFCC7   134 154
## PAR_MFCC8   135 155
## PAR_MFCC9   136 156
## PAR_MFCC10  137 157
## PAR_MFCC11  138 158
## PAR_MFCC12  139 159
## PAR_MFCC13  140 160
## PAR_MFCC14  141 161
## PAR_MFCC15  142 162
## PAR_MFCC16  143 163
## PAR_MFCC17  144 164
## PAR_MFCC18  145 165
```

```
## PAR_MFCC19   146 166
## PAR_MFCC20   147 167
## PAR_SC         2 180
## PAR_SC         2 184
## PAR_ZCD      180 184
## PAR_1RMS_TCD 181 186
## PAR_2RMS_TCD 182 188
## PAR_3RMS_TCD 183 190
```

- It can be seen that about 47 of these features are correlated. An analysis is done in the next section considering and omitting these features.

3. There were about 60 performers
4. 15-20 pieces were performed by each performer
5. The first segment of the 20 segments with 191 features are available.
6. Building the Classifier

- **Building a Naive Bayes Classifier with Feature Engineering and Normalization of the Data**
- A Naive bayes classifier was implemented for the training set. As discussed in the previous section, there are about 47 correlated features. This section removes these correlated features except one and the data is normalized by using the standardization formula, $\hat{x} = \frac{x-\overline{x}}{\sigma}$

```r
# Remove the correlated columns
tune.ds <- tuneds[,c(cno[,2],cno1[,2])*-1]

#Normalization function
normalize <- function(df){
  v_cols<-ncol(df)
  df_normalized <- df
  for (i in 1:v_cols){
    df_normalized[,i]<-(df_normalized[,i]-mean(df_normalized[,i]))/sd(df_normalized[,i])
  }
  return(df_normalized)
}

library(e1071)

#Normalize the dataset
df_norm.tune <- normalize(tune.ds[-147])
df_norm.tune["GENRE"]<-tune.ds[147]

# Build a Naive Bayes Model
s<-naiveBayes(GENRE~.,data=df_norm.tune)
# Create the confusion Matrix for Train dataset
table(predict(s,df_norm.tune),df_norm.tune[,147])
```

```
##
##             Blues Classical Jazz Metal  Pop Rock
##   Blues      1296        21  382    51   82  158
##   Classical    43      3131 1070    12   22   34
##   Jazz         13       267 1226     9   38   39
##   Metal       104         0   26   766   22  349
```

```
##   Pop           61         13  140    39 1314  109
##   Rock          79         12  159    47   97 1264
```

```
# Load the Test Dataset
trainds <- read.csv("genresTest.csv")
# Remove same correlated columns
c_train.ds <- trainds[,c(cno[,2],cno1[,2])*-1]
# Normalize the test dataset
df_normal<-normalize(c_train.ds)
# Read Base line from KNN
baseline<-read.csv("genresBaseline.txt")
# Confusion Matrix between the Naive Bayes prediction and the baseline KNN prediction.
table(predict(s,df_normal[-1,]),baseline[,1])
```

```
##
##              Blues Classical Jazz Metal  Pop Rock
##   Blues          3         0    2     0    2    0
##   Classical     29       167   93     8   35   23
##   Jazz          88      2739 1664     4   28   51
##   Metal        188         3   42   109   11  118
##   Pop          266        75  562    40  707  303
##   Rock         269       146  509    40   95 1849
```

- **Building a Naive Bayes Classifier with Feature Engineering**
- This test run is without normalizing the data,

```
library(corrplot)
tuneds <- read.csv("genresTrain.csv")
num.tuneds<- tuneds[-192]
cor.tune <- cor(num.tuneds)
cor.upper.tune <- cor.tune
cor.upper.tune[lower.tri(cor.upper.tune)] <- 0
diag(cor.upper.tune) <- 0
cno <- which(cor.upper.tune == 1,arr.ind = TRUE)
cno1 <- which(cor.upper.tune > 0.9,arr.ind = TRUE)
# Removing correlations
tune.ds <- tuneds[,c(cno[,2],cno1[,2])*-1]

library(e1071)
# Train the Model
s<-naiveBayes(GENRE~.,data=tune.ds)
# Confiusion Matrix of the Training set
table(predict(s,tune.ds),tune.ds[,147])
```

```
##
##              Blues Classical Jazz Metal  Pop Rock
##   Blues       1311        21  382    51   88  158
##   Classical     25      3131 1070    12   13   34
##   Jazz          12       267 1227     9   28   39
##   Metal        104         0   25   766   22  349
##   Pop           65        13  140    39 1327  109
##   Rock          79        12  159    47   97 1264
```

```r
trainds <- read.csv("genresTest.csv")
c_train.ds <- trainds[,c(cno[,2],cno1[,2])*-1]
baseline<-read.csv("genresBaseline.txt")
# COnfusion matrix for the Naive Bayes classifier output and the KNN output.
table(predict(s,c_train.ds[-1,]),baseline[,1])
```

```
##
##              Blues Classical Jazz Metal  Pop Rock
##   Blues        616        62  360    75  203  202
##   Classical     11      2418  540     3    6   25
##   Jazz          13       618 1635     4   24   54
##   Metal         60         0   13    74    0   66
##   Pop           85        20  169    16  569  254
##   Rock          58        12  155    29   76 1743
```

- **Building a Naive Bayes Classifier straight from data**
- No feature enginnering or standardization

```r
library(corrplot)
tuneds <- read.csv("genresTrain.csv")

library(e1071)
s<-naiveBayes(GENRE~.,data=tuneds)
# Confusion Matrix for Train Dataset
table(predict(s,tuneds),tuneds[,192])
```

```
##
##              Blues Classical Jazz Metal  Pop Rock
##   Blues       1311        14  403    39   74  160
##   Classical     31      3133  977    10   23   16
##   Jazz          14       278 1261    11   38   51
##   Metal         86         0   19   776   24  339
##   Pop           58         8  165    37 1319   94
##   Rock          96        11  178    51   97 1293
```

```r
trainds <- read.csv("genresTest.csv")
baseline<-read.csv("genresBaseline.txt")
# COnfusion Matrix for the Naive Bayes Ouput and Knn baseline output
table(predict(s,trainds[-1,]),baseline[,1])
```

```
##
##              Blues Classical Jazz Metal  Pop Rock
##   Blues        581        45  403    72  189  209
##   Classical      8      2362  416     4   15   20
##   Jazz          17       702 1716     8   32   56
##   Metal         55         0   13    73    0   73
##   Pop          108        14  162    12  577  198
##   Rock          74         7  162    32   65 1788
```

7. Conclusions

- Standardization or normalization, atleast with this example created lot of noise and aberrant results.
- Removing correlated columns had almost same accuracy as the full dataset. Infact, the correlatied columns removed dataset had more matching to blues for the output of the KNN than the full dataset.
- Matching the full dataset has the most match with the KNN baseline model.

Regarding the quality of the model, the full blown naive bayes has 7097 values classified out of 10269 values of test dataset. This brings up the accutacy of 69%.

8. An 1:NN algorithm was used for base solution. The proposed model is compared with the results of the baseline soution in the above question. ###References and acknowledgements
9. Packages Corrplot 0.73 and e1071 1.6-7 packages were used for this assignment.