SVM Classification

SVM Classification

For this exploration, we will be using SVM classification to determine whether a song is a “hit” or a “miss” based on Spotify features such as danceability, energy, and so on. It is worth noting that the author of the data set based their evaluation of a song being a “hit” or “miss” based on whether the song would be considered mainstream, so a song that is considered a “miss” is not necessarily a bad song but is simply not as popular compared to others. We will be exploring and comparing the performance of SVM classification with the following SVM kernels: linear, polynomial, and radial.

Load the data

Dataset: The Spotify Hit Predictor Dataset (1960-2019) via Kaggle.

The dataset actually has six data sets, with each data set having music from a specific decade (60’s, 70’s, and so on). We’ll just use two data sets for the 2000’s and 2010’s since those are the years that we happen to be alive for so it’d be interesting to see what features contribute to a song’s popularity during this particular time period.

```r
df_00 <- read.csv("dataset-of-00s.csv", header=TRUE)
df_10 <- read.csv("dataset-of-10s.csv", header=TRUE)
df <- rbind(df_00, df_10)

str(df)
```

```
## 'data.frame':    12270 obs. of  19 variables:
##  $ track           : chr  "Lucky Man" "On The Hotline" "Clouds Of Dementia" "Heavy Metal, Raise Hell
##  $ artist          : chr  "Montgomery Gentry" "Pretty Ricky" "Candlemass" "Zwartketterij" ...
##  $ uri             : chr  "spotify:track:4GiXBCUF7H6YfNQsnBRIzl" "spotify:track:1zyqZONW985Cs4osz9wl
##  $ danceability    : num  0.578 0.704 0.162 0.188 0.63 0.726 0.365 0.726 0.481 0.647 ...
##  $ energy          : num  0.471 0.854 0.836 0.994 0.764 0.837 0.922 0.631 0.786 0.324 ...
##  $ key             : int  4 10 9 4 2 11 1 11 10 7 ...
##  $ loudness        : num  -7.27 -5.48 -3.01 -3.75 -4.35 ...
##  $ mode            : int  1 0 1 1 1 0 1 0 1 1 ...
##  $ speechiness     : num  0.0289 0.183 0.0473 0.166 0.0275 0.0965 0.071 0.0334 0.0288 0.0377 ...
##  $ acousticness    : num  3.68e-01 1.85e-02 1.11e-04 7.39e-06 3.63e-01 3.73e-01 2.85e-03 2.20e-01 5.
##  $ instrumentalness: num  0 0 0.00457 0.0784 0 0.268 0 0 0 0 ...
##  $ liveness        : num  0.159 0.148 0.174 0.192 0.125 0.136 0.321 0.193 0.0759 0.115 ...
##  $ valence         : num  0.532 0.688 0.3 0.333 0.631 0.969 0.29 0.746 0.389 0.344 ...
##  $ tempo           : num  133 93 87 148 112 ...
##  $ duration_ms     : int  196707 242587 338893 255667 193760 192720 89427 239240 253640 314286 ...
##  $ time_signature  : int  4 4 4 4 4 4 4 4 4 3 ...
##  $ chorus_hit      : num  30.9 41.5 65.3 58.6 22.6 ...
##  $ sections        : int  13 10 13 9 10 10 4 10 11 16 ...
##  $ target          : int  1 1 0 0 1 0 0 1 1 0 ...
```

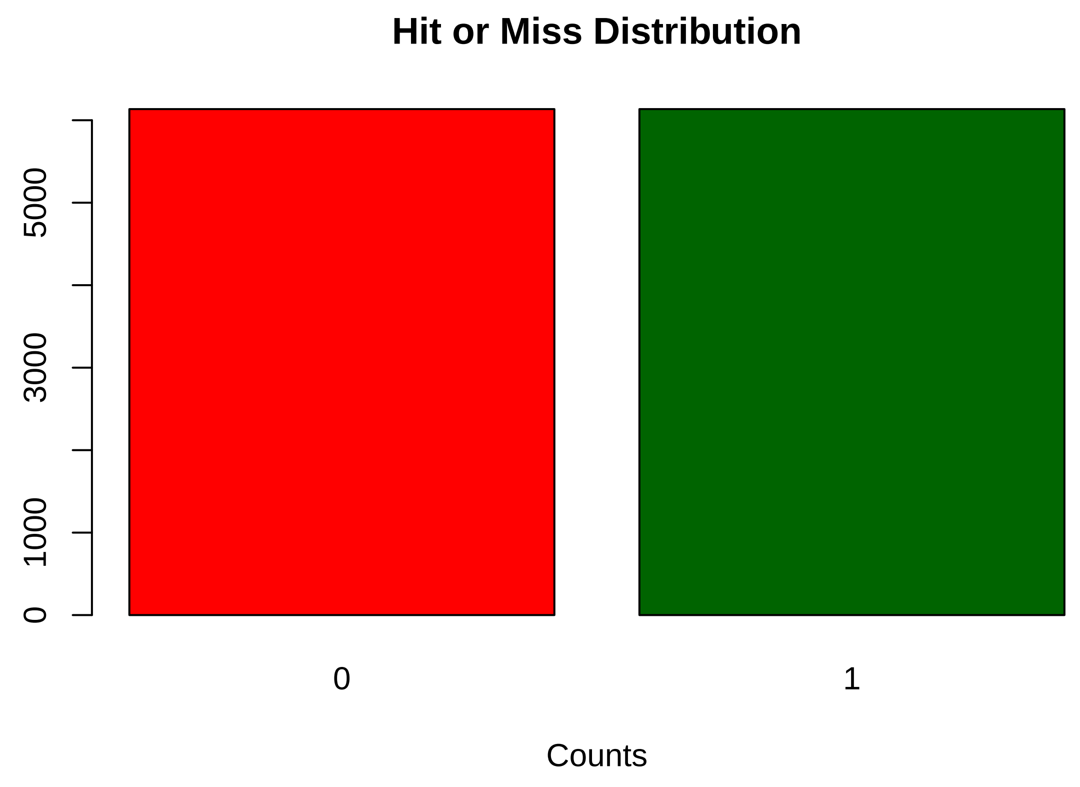
```r
df <- df[-c(1:3)]                                        # don't need the first three columns, so we can drop th
df$target <- factor(df$target, levels = c(0,1)) #encode target as a factor
```

Split Train and Test

```r
set.seed(1234)
i <- sample(1:nrow(df), 0.75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Data Exploration

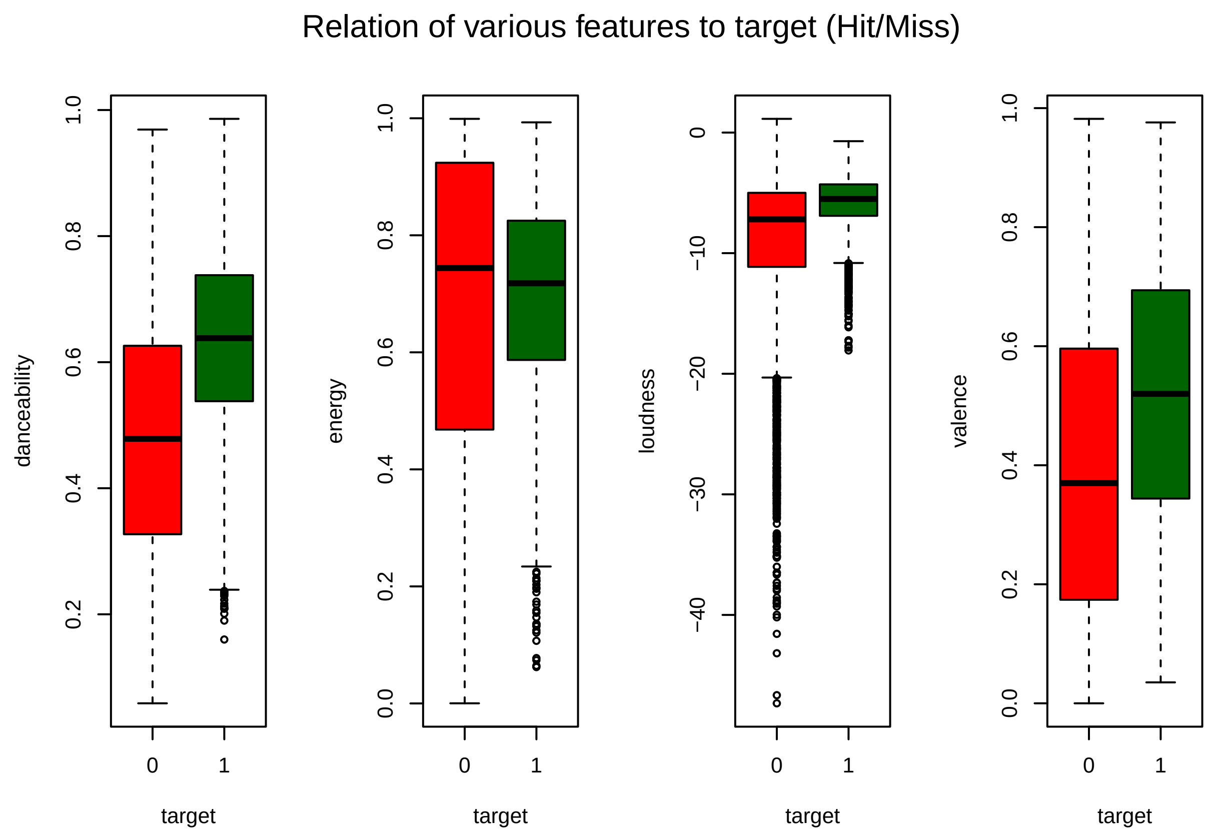
```r
# Hit or Miss Song Distribution
counts <- table(df$target)
barplot(counts, main="Hit or Miss Distribution", xlab="Counts", col=(c("red", "darkgreen")))
```



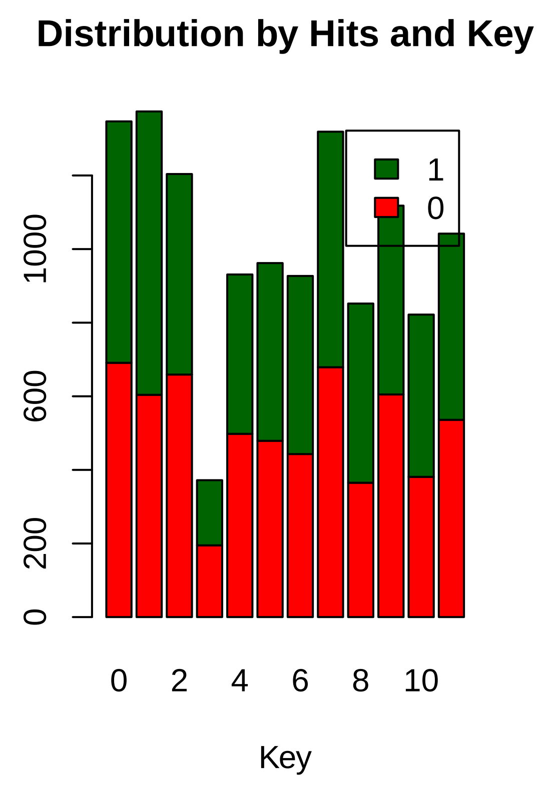
For this dataset, it looks like the hit or miss distribution is basically a fifty-fifty split so the data seems pretty balanced. Now we'll explore the other features.

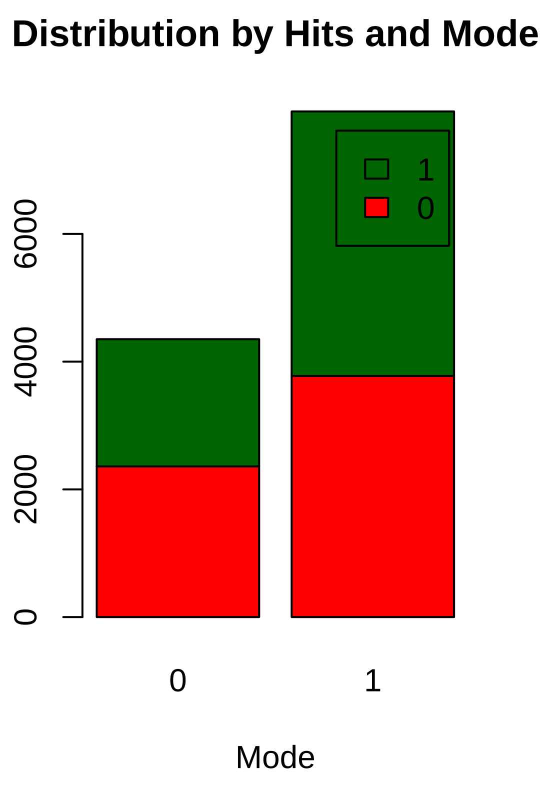
```r
# Boxplots of Various Features by Target (Hit/Miss)
par(mfrow=c(1,4))
```

```r
boxplot(danceability~target, data=train, xlab="target", ylab="danceability", col=(c("red", "darkgreen")
boxplot(energy~target, data=train, xlab="target", ylab="energy", col=(c("red", "darkgreen")))
boxplot(loudness~target, data=train, xlab="target", ylab="loudness", col=(c("red", "darkgreen")))
boxplot(valence~target, data=train, xlab="target", ylab="valence", col=(c("red", "darkgreen")))

# main title
mtext("Relation of various features to target (Hit/Miss)",
      side = 3,
      line = -2,
      outer = TRUE)
```



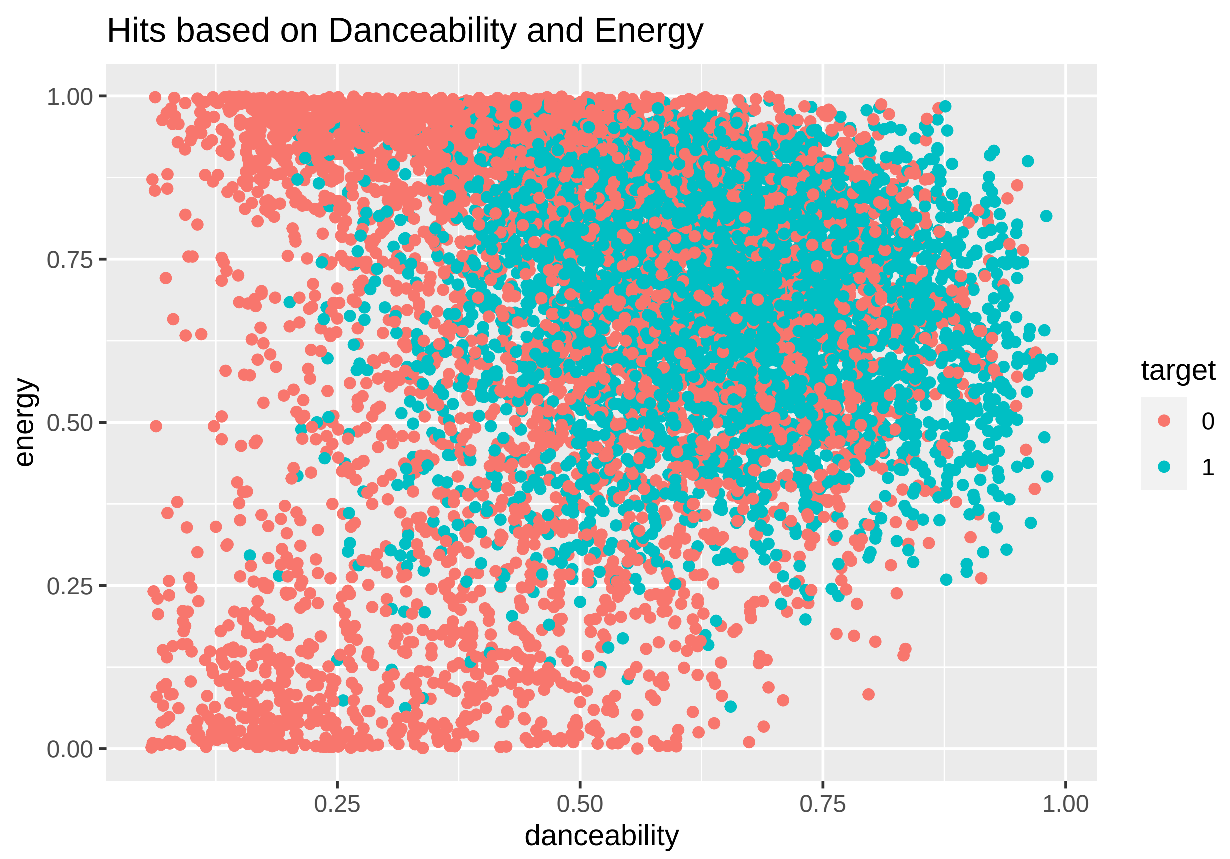
```r
par(mfrow=c(1,2))
counts <- table(df$target, df$key)
barplot(counts, main="Distribution by Hits and Key",
  xlab="Key",  col=(c("red", "darkgreen")),
  legend = rownames(counts))

counts <- table(df$target, df$mode)
barplot(counts, main="Distribution by Hits and Mode",
  xlab="Mode",  col=(c("red", "darkgreen")),
  legend = rownames(counts))
```





```r
par(mfrow=c(1,1))
```

```r
library(ggplot2)

ggplot(train) + geom_point(aes(x=danceability,y=energy,colour=target)) +
  labs(title = "Hits based on Danceability and Energy", x = "danceability", y = "energy")
```



Linear SVM

```r
library(e1071)
svm1 <- svm(target~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm1)
```

```
##
## Call:
## svm(formula = target ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  4321
##
##  ( 2160 2161 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  0 1
```

```
# evaluate on test data
pred1 <- predict(svm1, newdata=test)
table(pred1,test$target)
```

```
##
## pred1    0    1
##     0 1083  132
##     1  428 1425
```

```
acc <- mean(pred1==test$target)
print(paste("acc: ", acc))
```

```
## [1] "acc:  0.817470664928292"
```

Polynomial SVM

```
svm2 <- svm(target~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = target ~ ., data = train, kernel = "polynomial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  4027
##
##  ( 2014 2013 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
# evaluate on test data
pred1 <- predict(svm1, newdata=test)
table(pred1,test$target)
```

```
##
## pred1    0    1
##     0 1083  132
##     1  428 1425
```

```
acc <- mean(pred1==test$target)
print(paste("acc=", acc))
```

```
## [1] "acc= 0.817470664928292"
```

Radial SVM

```
svm3 <- svm(target~., data=train, kernel="radial", cost=10, scale=TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = target ~ ., data = train, kernel = "radial", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  3725
##
##  ( 1844 1881 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
# evaluate on test data
pred1 <- predict(svm1, newdata=test)
table(pred1,test$target)
```

```
##
## pred1    0    1
##     0 1083  132
##     1  428 1425
```

```
acc <- mean(pred1==test$target)
print(paste("acc=", acc))
```

```
## [1] "acc= 0.817470664928292"
```