Regression

Code

Ryan Gagliardi, Andrew Gerungan, Ethan Huynh, Meinhard Capucao

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Hide

```
songs <- read.csv('unpopular_songs.csv')

set.seed(1234)
i <- sample(1:nrow(songs), nrow(songs)*.80, replace=FALSE)
train <- songs[i,]
test <- songs[-i,]
```

Linear Regression The correlation here is fairly low and the mse/rmse are both fairly high. This is a result of the data set being poorly correlated and the predictors not having much of an effect on the target.

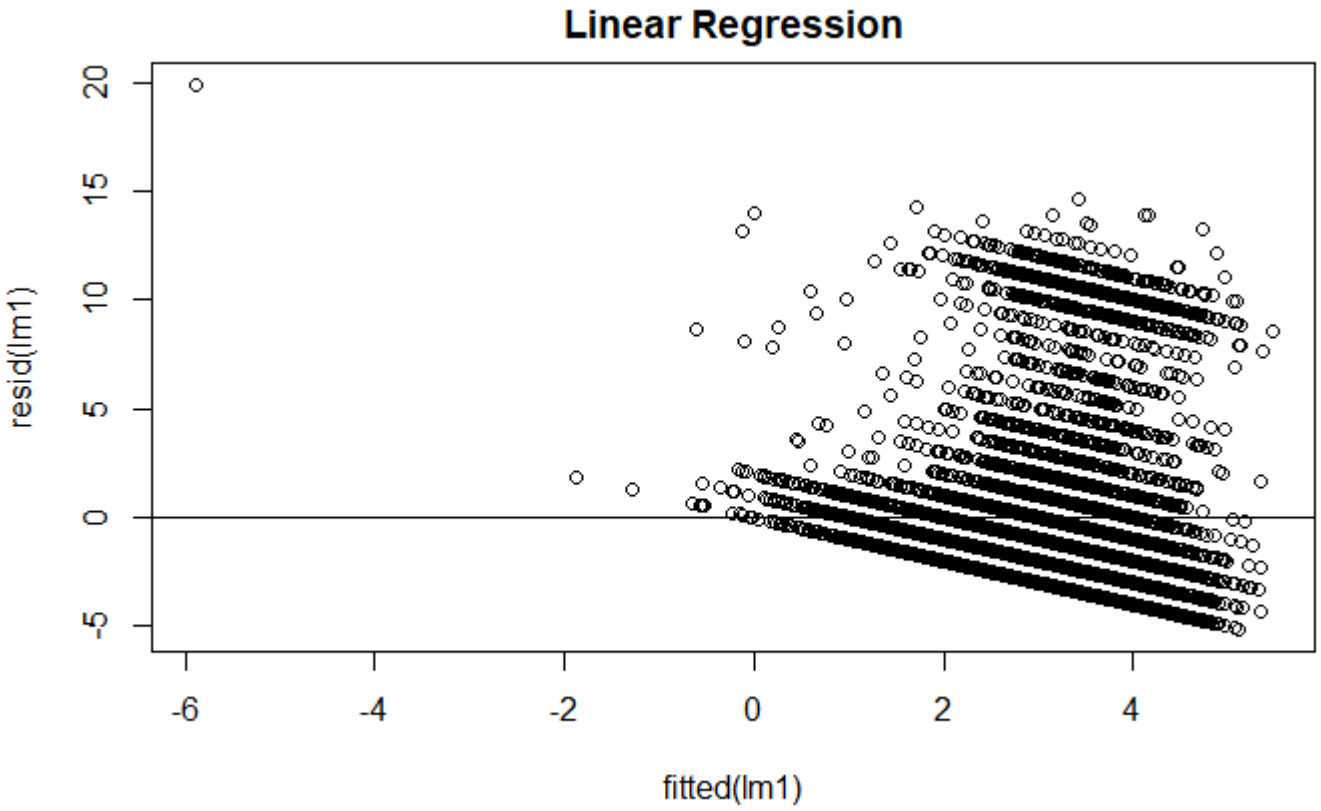
Hide

```
lm1 <- lm(popularity~danceability+energy+key+loudness+speechiness+acousticness+liveness+valence+
tempo+duration_ms, data=train)
summary(lm1)
```

```
Call:
lm(formula = popularity ~ danceability + energy + key + loudness +
    speechiness + acousticness + liveness + valence + tempo +
    duration_ms, data = train)

Residuals:
   Min     1Q Median     3Q    Max
-5.116 -2.423 -1.111  0.465 19.890

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.830e+00  3.773e-01  10.151  < 2e-16 ***
danceability  2.640e+00  2.749e-01   9.602  < 2e-16 ***
energy       -7.953e-01  2.651e-01  -3.000 0.002710 **
key          -7.751e-03  1.155e-02  -0.671 0.502085
loudness      8.660e-02  9.496e-03   9.119  < 2e-16 ***
speechiness  -2.292e-02  2.752e-01  -0.083 0.933645
acousticness -7.645e-01  1.575e-01  -4.854 1.23e-06 ***
liveness     -4.689e-01  2.437e-01  -1.925 0.054315 .
valence      -1.491e+00  1.921e-01  -7.762 9.32e-15 ***
tempo         5.004e-03  1.376e-03   3.636 0.000279 ***
duration_ms  -1.609e-06  3.979e-07  -4.045 5.28e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.893 on 8690 degrees of freedom
Multiple R-squared:  0.04975,   Adjusted R-squared:  0.04866
F-statistic:  45.5 on 10 and 8690 DF,  p-value: < 2.2e-16
```

Hide

```
plot(fitted(lm1), resid(lm1), main = "Linear Regression")
abline(0,0)
```



Hide

```
pred1 <- predict(lm1, newdata=test)
cor1 <- cor(pred1, test$popularity)
mse1 <- mean((pred1-test$popularity)^2)
rmse1 <- sqrt(mse1)
print(paste('correlation:', cor1))
```

```
[1] "correlation: 0.220409794214653"
```

Hide

```
print(paste('mse:', mse1))
```

```
[1] "mse: 15.7362109733501"
```

Hide

```
print(paste('rmse:', rmse1))
```

```
[1] "rmse: 3.9668893321279"
```

kNN Regression We see that scaled is clearly the better option when it comes to kNN regression. It doubles our correlation and decreases our mse by about 1/16th.The best K was tested and found that cor increases as K increases to 50. Because the data is not very correlated, the correlation and mse are both fairly bad. A better

correlated data set would have the correlation be closer to 1 and the mse/rmse be lower.

Hide

```
train_scaled <- train[, c(1,2,4,6:12)] # Don't include columns that are classification
means <- sapply(train_scaled, mean)
stdvs <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center=means, scale=stdvs)
test_scaled <- scale(test[, c(1,2,4,6:12)], center=means, scale=stdvs)


#Scaled Data
library(caret)
fit <- knnreg(train_scaled, train$popularity, k=50)
pred2 <- predict(fit, test_scaled)
cor_knn2 <- cor(pred2, test$popularity)
mse_knn2 <- mean((pred2 - test$popularity)^2)
print(paste("scaled cor=", cor_knn2))
```

```
[1] "scaled cor= 0.265827492955055"
```

Hide

```
print(paste("scaled mse=", mse_knn2))
```

```
[1] "scaled mse= 15.3721868993018"
```

Hide

```
print(paste("scaled rmse=", sqrt(mse_knn2)))
```

```
[1] "scaled rmse= 3.92073805543061"
```

Hide

```
#Unscaled Data
fit <- knnreg(train[,c(1,2,4,6:12)],train[,14],k=50)

pred <- predict(fit, test[,c(1,2,4,6:12)])
cor_knn1 <- cor(pred, test$popularity)
mse_knn1 <- mean((pred - test$popularity)^2)
print(paste("cor=", cor_knn1))
```

```
[1] "cor= 0.135872633654924"
```

Hide

```
print(paste("mse=", mse_knn1))
```

```
[1] "mse= 16.3067965204737"
```

Decision Tree The decision tree here is lackluster. Due to the data not being very correlated to the predictor, there is not many variables that the algorithm can find that have an impact on the outcome substantial enough to actually use it in the tree. A better data set would have many different variables being used to show how each on individually affects the predictor.

Hide

```
library(tree)
library(MASS)
tree1 <- tree(popularity~danceability+energy+loudness+speechiness+acousticness+instrumentalness+
liveness+valence+tempo+duration_ms, data = train)
summary(tree1)
```

```

Regression tree:
tree(formula = popularity ~ danceability + energy + loudness +
    speechiness + acousticness + instrumentalness + liveness +
    valence + tempo + duration_ms, data = train)
Variables actually used in tree construction:
[1] "loudness"    "duration_ms"
Number of terminal nodes:  3
Residual mean deviance:  15.26 = 132800 / 8698
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-3.9260 -2.8380 -0.9262  0.0000  0.2977 14.3000
```

Hide

```
pred <- predict(tree1, newdata=test)
print(paste('correlation:', cor(pred, test$popularity)))
```

```
[1] "correlation: 0.196462833811304"
```

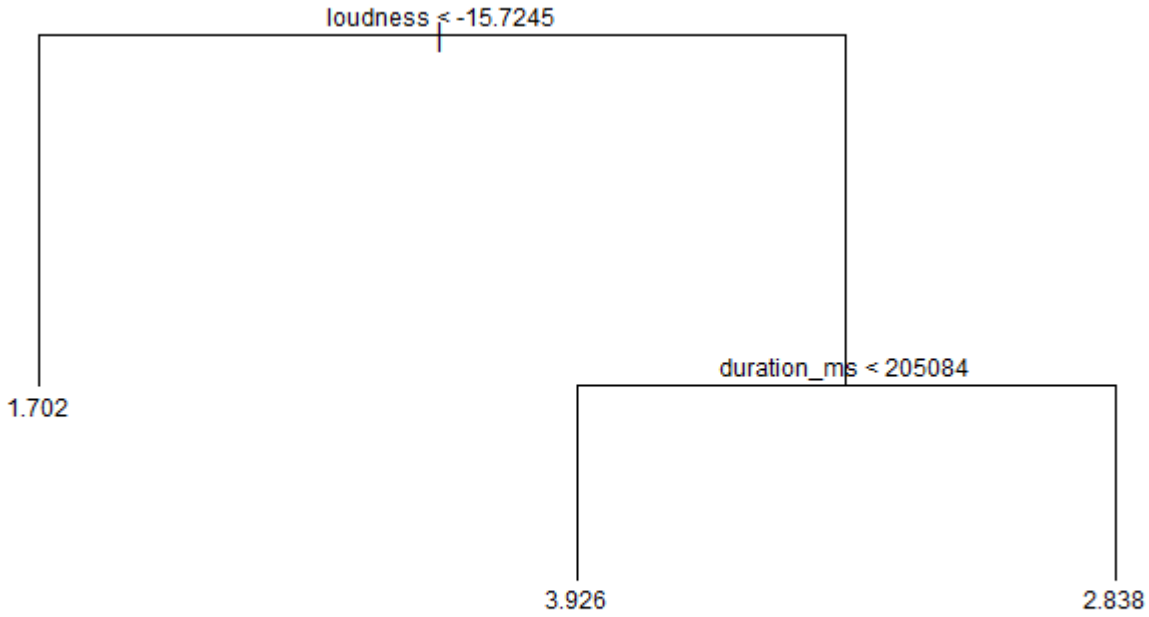
Hide

```
rmse_tree <- sqrt(mean((pred-test$popularity)^2))
print(paste('rmse:', rmse_tree))
```

```
[1] "rmse: 3.98806785427635"
```

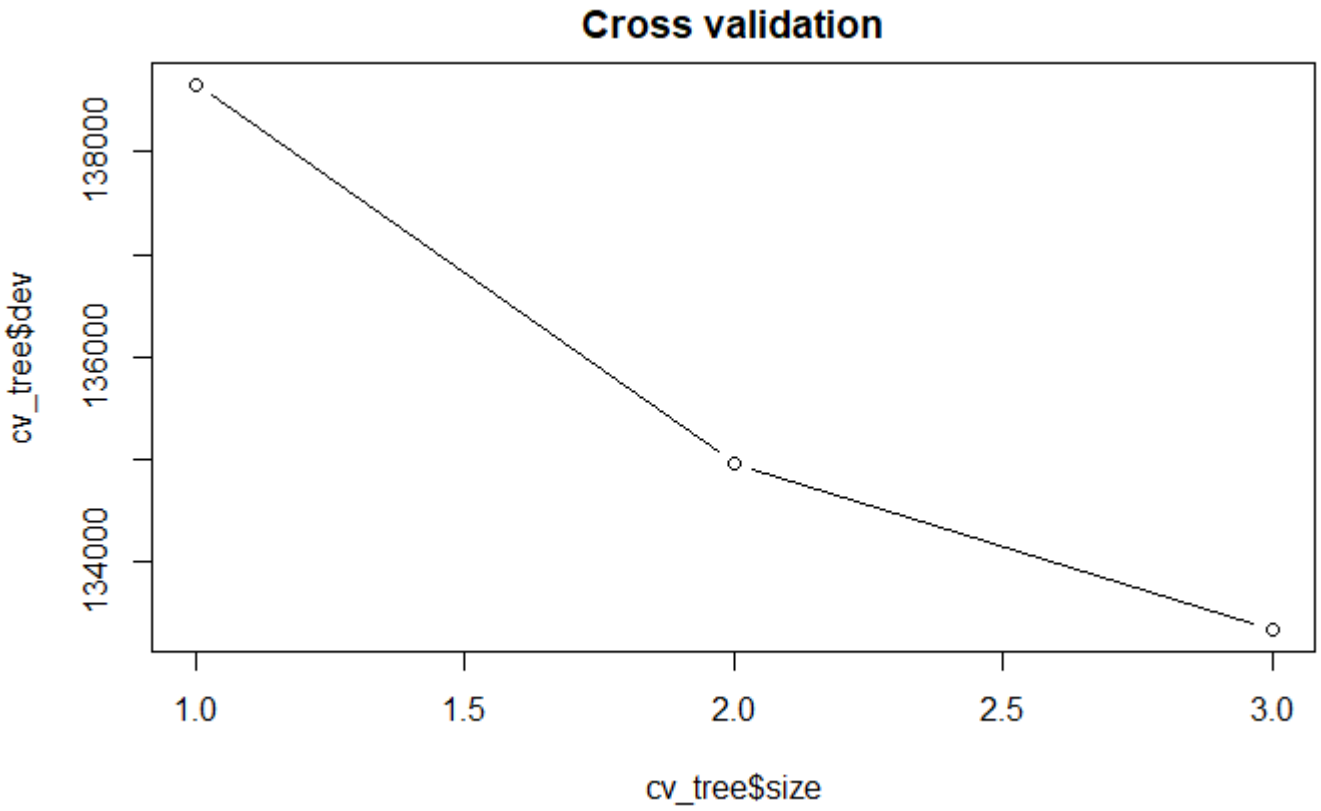
Hide

```
plot(tree1, main = "Tree")
text(tree1, cex=0.75, pretty=0)
```



Hide

```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b', main = "Cross validation")
```



All in all, the 3 different algorithms used here are all useful in their own way. Linear regression being the less accurate is because of the simplicity and ease of which the data is analyzed, simply attempting to find a line that can predict the target. kNN regression found a better correlation but was still low and had an unimpressive mse. Tree faired the worst as it did not have enough variables that were correlated enough to properly guess the outcome, so it hinged its assumption on two variables meaning its correlation and rmse are both very poor. Overall my data set was poor and in the future I need to find one that has higher correlation so that the algorithms can run more smoothly. ```