Regression

Kyle Chan, Aidan Case, Kevin Cordano, Mason Cushing, Sagar Darnal

This data set contains over 10,000 unpopular Spotify songs. The download link can be found here.

First, we begin by reading in the data set and dividing our data into an 80/20 train/test split.

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

set.seed(1234)
train_index <- sample(1:nrow(unpopular_songs), size = 0.8*nrow(unpopular_songs), replace = FALSE)
train <- unpopular_songs[train_index, ]
test <- unpopular_songs[-train_index, ]
```

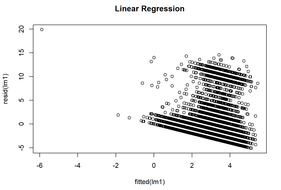
Linear Regression

Next, take in variables and return a linear regression model that predicts popularity. We will also print the summary.

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

```
##
## Call:
## lm(formula = popularity ~ danceability + energy + key + 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  9.486e-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
```

```
par(mfrow=c(1,1))
plot(fitted(lm1), resid(lm1), main = 'Linear Regression')
```



Let's determine correlation and calculate MSE/RMSE.

```
pred1 <- predict(lm1, test)
cor1 <- cor(pred1, test$popularity)
mse1 <- mean((pred1 - test$popularity)^2)
rmse1 <- sqrt(mse1)
print(paste("Correlation: ", cor1))
```

```
## [1] "Correlation:  0.220409794214653"
```

```
print(paste("MSE: ", mse1))
```

```
## [1] "MSE:  15.7362109733501"
```

```
print(paste("RMSE: ", rmse1))
```

```
## [1] "RMSE:  3.96688933212779"
```

The correlation here is low, and the MSE/RMSE are both high. We can conclude that the predictors have little correlation and effect on the target variable, popularity.

kNN Regression

Next, we will use kNN regression to predict popularity. We will use the same predictors as the linear regression model. First, we will demonstrate how kNN Regression fares without scaling. It should be noted that since, our data set also includes columns meant for classification, we will need to omit them from our train/test.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
knn1 <- knnreg(train[, c(1, 2, 4, 6:12)], train[, 14], k = 50)
pred1 <- predict(knn1, test[, c(1, 2, 4, 6:12)])
cor1 <- cor(pred1, test[, 14])
mse1 <- mean((pred1 - test[, 14])^2)
rmse1 <- sqrt(mse1)
print(paste("Correlation: ", cor1))
```

```
## [1] "Correlation:  0.135872633654924"
```

```
print(paste("MSE: ", mse1))
```

```
## [1] "MSE:  16.3067965204737"
```

```
print(paste("RMSE: ", rmse1))
```

```
## [1] "RMSE:  4.03816747058288"
```

We will now scale our data for KNN Regression. Scaling our data will improve a number of factors:

- Since kNN Regression uses a distance metric to calculate similarity, we want to ensure all features in our data set contribute equally to the distance calculation.
- kNN Regression is very sensitive to the scale of the input. Therefore, if we do not scale, our data may take longer to converge, if at all.
- Scaling assists in improving accuracy by normalizing the different scales of each feature and our target variable. It will make identifying patterns much easier in our data set.

```
train_scaled <- train[, c(1, 2, 4, 6:12)]
means <- sapply(train_scaled, mean)
sds <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center = means, scale = sds)
test_scaled <- scale(test[, c(1, 2, 4, 6:12)], center = means, scale = sds)
```

```
knn2 <- knnreg(train_scaled, train$popularity, k = 50)
pred2 <- predict(knn2, test_scaled)
cor2 <- cor(pred2, test$popularity)
mse2 <- mean((pred2 - test$popularity)^2)
rmse2 <- sqrt(mse2)
print(paste("Correlation: ", cor2))
```

```
## [1] "Correlation:  0.265827492955055"
```

```
print(paste("MSE: ", mse2))
```

```
## [1] "MSE:  15.3721868993018"
```

```
print(paste("RMSE: ", rmse2))
```

```
## [1] "RMSE:  3.92073805543061"
```

We can see that scaled is evidently better than unscaled in kNN Regression. Scaling has doubles our correlation value, and decreases the mse by a value of 1. We have experimented with K, and determined that our correlation value increases as K approaches 50. Even with this method of regression, however, we've concluded the predictors have little impact on our target variable, since the correlation and mse values are fairly insignificant.

Decision Trees

Next, we will use decision trees to predict popularity.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.2.3
```

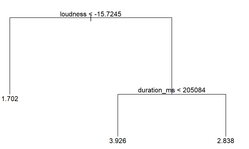
```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.2.3
```

```
tree1 <- tree(popularity ~ danceability+energy+key+key+loudness+speechiness+acousticness+liveness+valence+tempo+d
uration_ms, data = train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = popularity ~ danceability + energy + key + key +
##     loudness + speechiness + acousticness + 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
```

```
plot(tree1, main = "Tree")
text(tree1)
```




```
pred1 <- predict(tree1, test)
cor1 <- cor(pred1, test$popularity)
mse1 <- mean((pred1 - test$popularity)^2)
rmse1 <- sqrt(mse1)
print(paste("Correlation: ", cor1))
```

```
## [1] "Correlation:  0.196462833811304"
```

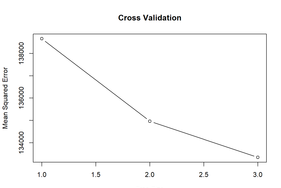
```
print(paste("MSE: ", mse1))
```

```
## [1] "MSE:  15.9046852103123"
```

```
print(paste("RMSE: ", rmse1))
```

```
## [1] "RMSE:  3.98806785427635"
```

```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type = "b", xlab = "Tree Size", ylab = "Mean Squared Error", main = "Cross Valida
tion")
```



The decision tree is not as useful for this data set. The tree does not perform well predicting the target variable, since decision trees work by creating small decision rules that can predict the target variable.

Analysis and Closing Thoughts

Overall, we see poor results across all algorithms for this data set. Linear regression performs poorly due to the simplicity of which the data is analyzed, since it just attempts to fit a line between the predictors and the target variable. kNN regression performed the best, albeit not by much, with the greatest correlation value. Our kNN regression model seemed to struggle due to the difficulty in identifying similar data points to the query point, resulting in less accuracy. Decision trees resulted in the worst performance. Since there are no clear patterns or relationships between the input features and the target variable, the algorithm struggles to identify significant splits and decision rules.

All 3 algorithms are supervised ML algorithms with unique approaches and usefulness in varying circumstances. Linear regression may be useful when there is a linear relationship between the predictors and the target variable. kNN regression is useful in handling non-linear (or those that appear not to be) relationships between its input features and the target variable. Decision trees are useful when there are more complex relationships between the features and the target variable. Additionally, decision trees are capable of handling categorical and numerical data at the same time. In the future, we will take greater care in identifying a data set with higher correlation so that the algorithms may truly demonstrate their potential.