Kelvin Nguyen
Sarah Fasanando
Sarah Kyeremeh
Zachary Brewer

1. Introduction (Sarah K)

The dataset used combines data from Spotify and the Billboard's top 100 songs from 2000-2018. It contains 20 predictors and 7,367 observations. Our goal is to to see if it is possible to predict the popularity of the song based on its energy, liveness, tempo, speechiness, acousticness,and instrumentalness. We will use SVM and KNN to create a model to predict where a song lies in terms of its popularity.

2. Questions and Models (Sarah F. and Zach B.)

a.i. The main question that we narrowed down to answer was if we could accurately predict how popular a song would be given the predictors. We believe it's important because depending on what is "trending" based on the amount of likes a particular song or group of songs, we can inform people in the music industry such as artists, information on the types of songs that are likely to be popular. The data could also be used to inform recommendations to create algorithmically selected playlists (similar to the "Discover Now" playlist made by spotify.) In order to answer this question we used KNN and SVM.

KNN and SVM are both nonlinear classification systems that can potentially divide the the data into one of 2 categories: popular (1) and not popular (0). We used KNN because it is good at handling a lot of points in a low dimensional space, and SVM due to its ability to handle a higher dimensional space.The advantages that SVM will have over KNN is that it is easier to scale with a large number of observations, is better at handling outliers, and will not be as time

consuming. The advantages that KNN has is that it tends to perform very well when it has a lot of data points, however KNN has the disadvantage of being quite sensitive to outliers .

(Zach B. Sarah F.)

b.i. The way KNN calculates the prediction for an observation is by first finding the K points closest to the current observation. It then classifies the current observation with the same class that the majority of the K points around it belong to. In math notation for the conditional probability for class "j" is

$$Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(yi = j)$$

where x_0 is the current observation, N_0 is the fraction of closest points whose class equals j, and K is the number of closest points we are using. For SVM we used linear, radial, and polynomial kernels which gave us hyperplane formulas of

$$\beta_0 + \sum_{k=1}^{p} (\sum_{i \in S} \alpha_i x_{ik}) x_k \quad , \quad exp(-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2), \gamma > 0 ,$$

and $\beta_0 + \sum_{i \in S} \alpha_i (1 + 2 \sum_{k=1}^{p} x_{ik} x_k + (\sum_{k=1}^{p} x_{ik} x_k)^2)$ respectively.

(Zach, Sarah F, Kelvin N, Sarah K)

b.ii. For our predictors we chose to use energy, liveness, tempo, speechiness, acousticness, instrumentalness, danceability, duration_ms, loudness, and valence. We chose these because we believed that they would have the biggest impact on a songs popularity since they were all characteristics that would affect the listener's "response" to the song. Especially since our goal is to predict a generic "popularity-ness" of a song. We excluded peak_pos, last_post, key and time signature from our calculations as almost all of the songs had a time signature of four and we

didn't include key because it is simply the musical scale the song was composed around which should have little impact on its popularity. Additionally, features such as peak_pos and last_pos are intertwined with our response due to it affecting our "weeks" predictor. Obviously if a song had a very high peak position, it will most likely be in the Top 100 for a long time due to mere exposure.

Due to our predictors having a variety of magnitudes it was necessary for us to scale our data. Most of our predictors had a range of zero to one, while some of our predictors, such as tempo and duration_ms, were measured in the hundreds and thousands. We determined popularity by calculating the mean number of weeks a song spent on Billboard's top 100 and stating any song that was on the list for more than the average number of weeks was popular. Then we created another column, weeksAbove, assigning a classifier to each observation; 1 meaning popular and 0 being not popular.

b.iii. (Kelvin N) We determined the optimal K for our KNN model was K = 10. For SVM our optimal tuning parameters for the linear kernel was a cost of 0.1, for the radial kernel a cost of 1 and a gamma value 0.5, and for the polynomial kernel a cost of 1 and degree of 1. Our best

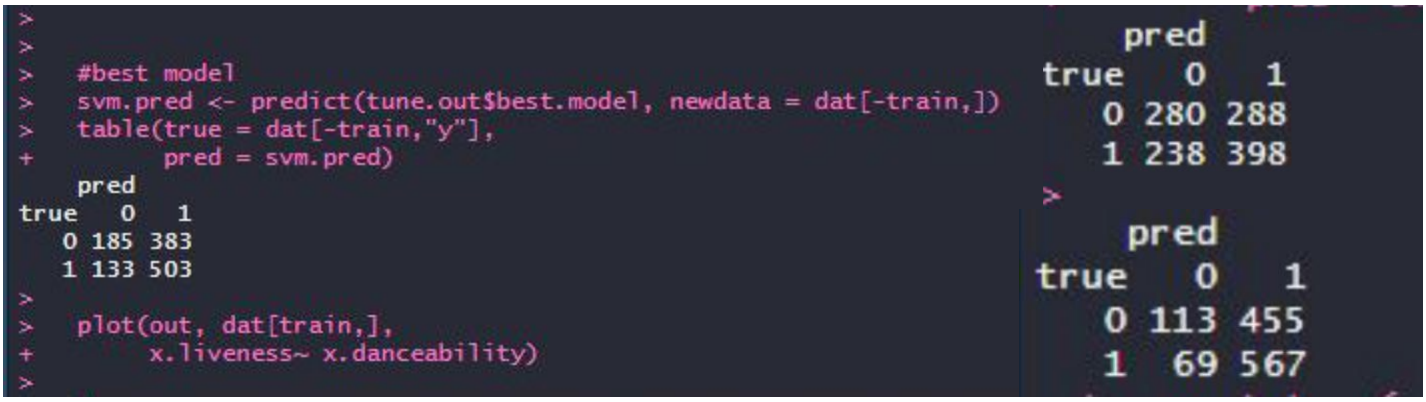
```
> #running KNN
> for (K in c(1,3,5,10,100))
+ {
+   set.seed(1)
+   knn.pred <- knn(train=train.X,
+                   test=test.X,
+                   cl=train.Y,
+                   k=K)
+
+   head(knn.pred)
+   print(mean(knn.pred != test.Y))
+ }
[1] 0.4630007
[1] 0.467074
[1] 0.4494229
[1] 0.4399185
[1] 0.4467074
>
>   # Table for the best K
>   table(knn.pred,test.Y)
        test.Y
knn.pred   0   1
       0 274 167
       1 491 541
>
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.4400823

- Detailed performance results:
   cost     error dispersion
1 1e-03 0.4504394 0.01775187
2 1e-02 0.4439872 0.01635580
3 1e-01 0.4400823 0.01744996
4 1e+00 0.4409309 0.01751807
5 5e+00 0.4409315 0.01663561
6 1e+01 0.4416100 0.01721065
7 1e+02 0.4414402 0.01758907
```

performer for SVM was the linear kernel.

(Kelvin N)

b.iv. We subdivided eighty percent of the dataset into training data and the remaining twenty percent into test data. We then trained and tested our different models. KNN had a test error rate of 43.99%, our linear kernel had a training error rate of 44%, our radial kernel SVM had a training error rate of 45%, and our polynomial kernel had a training error rate of 44.17%.



Photos clockwise: linear, radial, and polynomial. The test error rates are 35%, 35.7%, and 35.57% respectively

(Zach B)

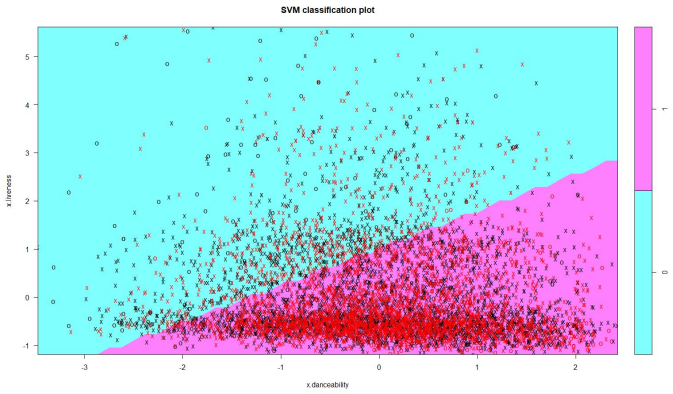
c. When comparing the test error rates of the different models it is pretty clear that SVM does a much better job with this dataset than KNN does. Our worst SVMs (radial and polynomial) outperformed KNN by 8.29% and 8.42%, and our best SVM (linear) outperformed KNN by 8.99%, a fairly significant improvement.

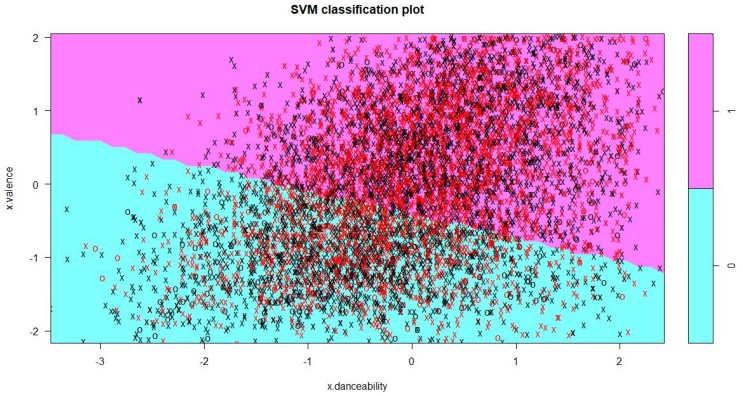
d.i. (Kelvin N) Our best models: KNN 34.8% error. SVM (Linear) 43.3% error

(Although the code says train/test, it is actually the whole data set in this portion)

```
> set.seed(1)
> knn.pred <- knn(train=train.X,
+                 test=test.X,
+                 cl=train.Y,
+                 k=10)
> head(knn.pred)
[1] 1 1 0 1 1 1
Levels: 0 1
> print(mean(knn.pred != test.Y))
[1] 0.3482688
> # Table for the best K
> table(knn.pred,test.Y)
        test.Y
knn.pred    0    1
       0 2166 1096
       1 1469 2634
```

```
Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cost = 0.1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.1

Number of Support Vectors:  6786

 ( 3394 3392 )


Number of Classes:  2

Levels:
 0 1
> table(out$fitted, dat$y)

       0    1
  0 1523 1078
  1 2112 2652
```





(Everyone)

d.ii. Overall, with a 34% and 43% error rate in our KNN and SVM models respectively, predicting song popularity based on the predictors chosen will not be very accurate. Although SVM out performed KNN during the validation stage of out project, our results on the entire dataset is flipped. Despite this, the error rate is still too high for both KNN and SVM for them to have a respectable predictive performance.These systems will not be able to classify whether a song will be popular or not; there is basically a 50% chance of predicting if we simply look at the plots and use our intuition. We would essentially have the same results from flipping a coin. We were ultimately not able to achieve good performance. Looking at early plots and the fact that our final result does not match up with our preliminary training, we can easily tell that this would have been the case. In conclusion, our data may simply be too noisy or the scope of our resources are too limited.