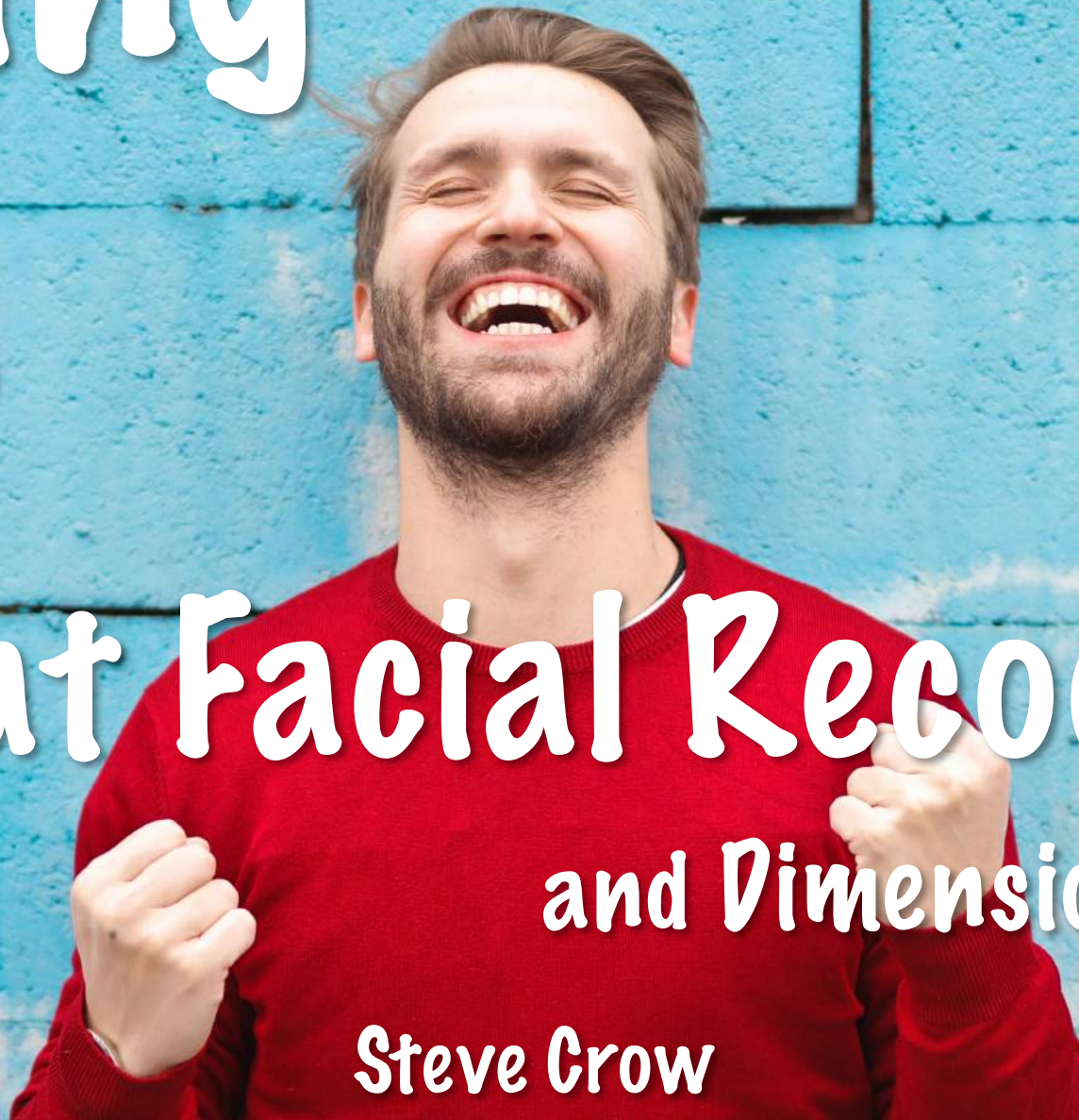
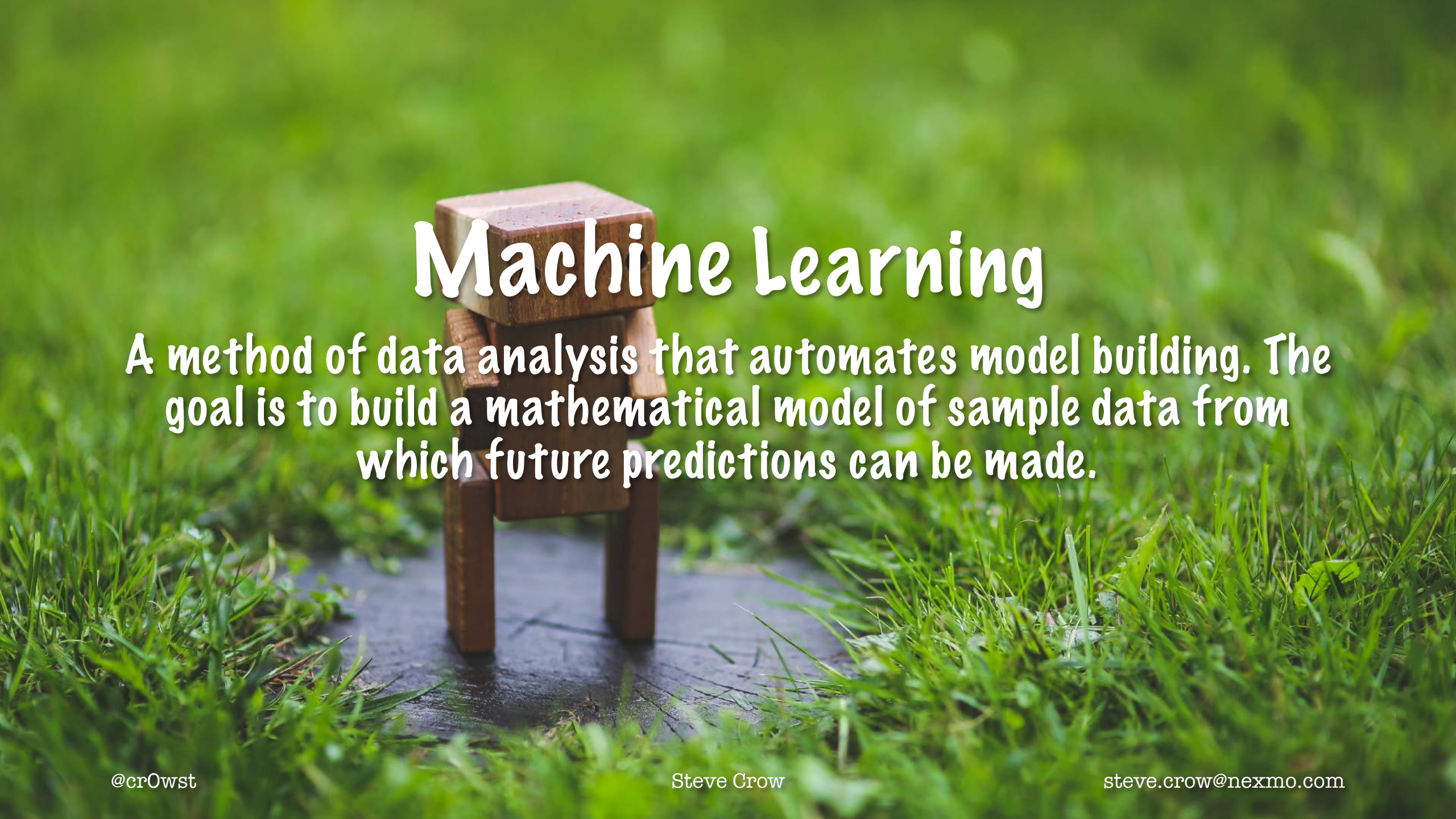
Making Faces

A Look at Facial Recognition and Dimension Reduction



Steve Crow



Machine Learning

A method of data analysis that automates model building. The goal is to build a mathematical model of sample data from which future predictions can be made.

@crOwst Steve Crow steve.crow@nexmo.com

@crOwst Steve Crow steve.crow@nexmo.com

(Semi) Supervised Learning

Supervised Learning uses a combination of input and output data in the training set to build a mathematical model.

@crOwst Steve Crow steve.crow@nexmo.com

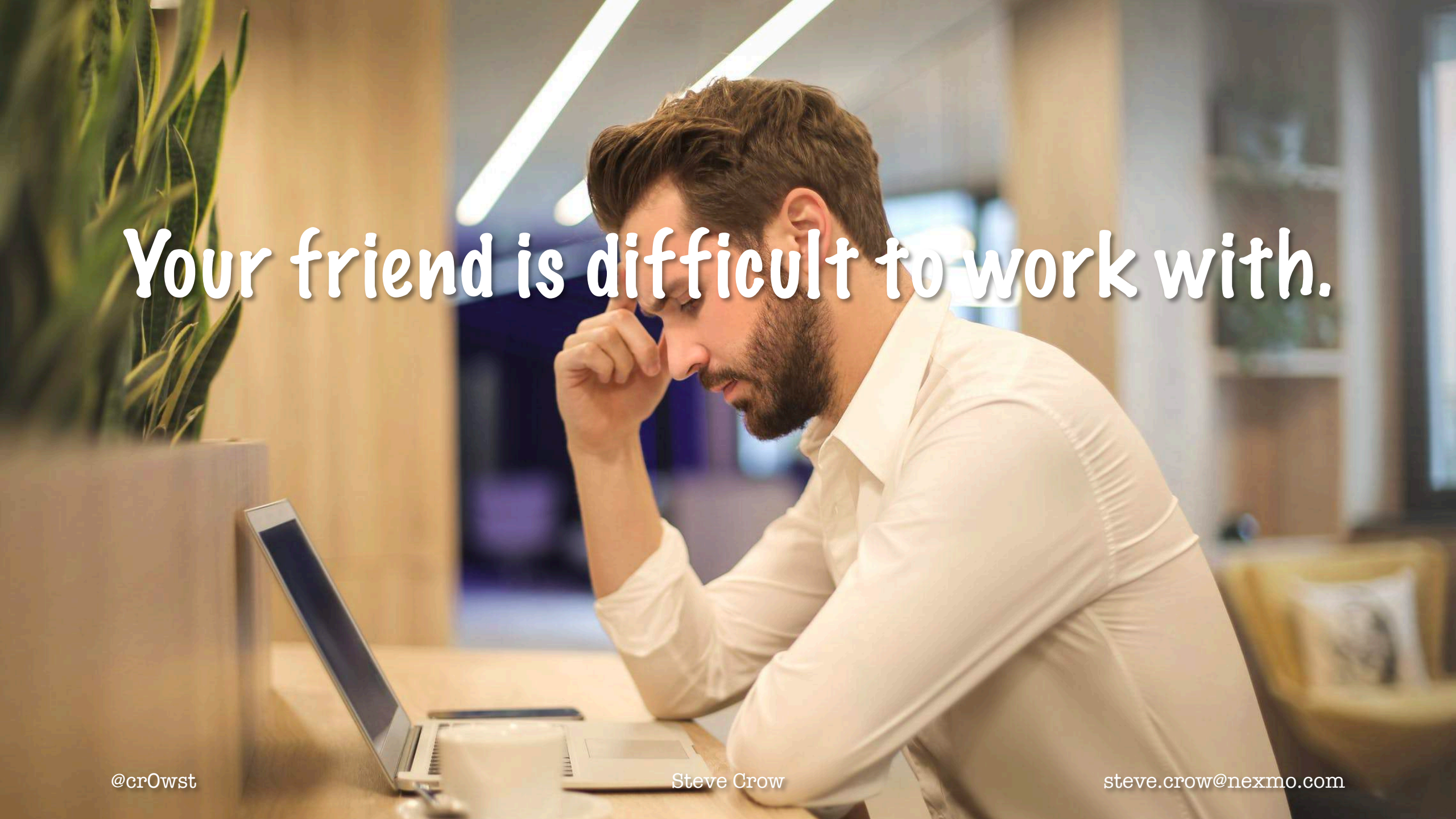
Unsupervised Learning

Unsupervised learning uses only inputs in the dataset and finds structures or grouping in the data.

@crOwst Steve Crow steve.crow@nexmo.com

@crOwst Steve Crow steve.crow@nexmo.com

@crOwst Steve Crow steve.crow@nexmo.com



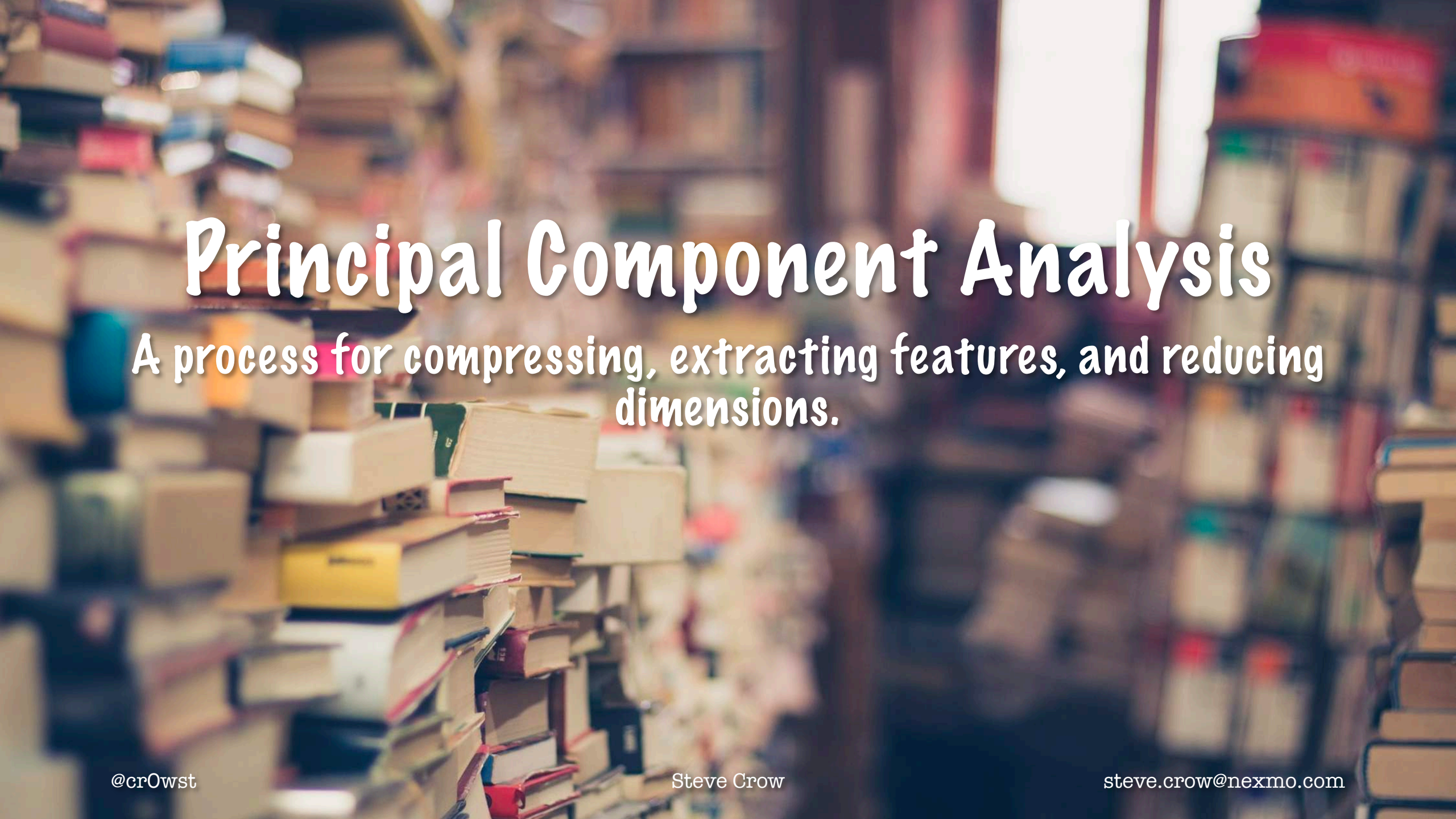
@crOwst Steve Crow steve.crow@nexmo.com

@crOwst Steve Crow steve.crow@nexmo.com


Which songs to pick?

Musical Features

accousticness, danceability, energy, instrumentalness, liveness, loudness, speachiness, valence, tempo, time signature, key, duration...



Principal Component Analysis

A process for compressing, extracting features, and reducing dimensions.

@crOwst

Steve Crow

steve.crow@nexmo.com

Feature Elimination

Reduce the feature space by eliminating features.

@crOwst Steve Crow steve.crow@nexmo.com

Feature Extraction

Create new features as combination of existing features in an attempt to reduce any sort of redundancy.

@crOwst Steve Crow steve.crow@nexmo.com

When to use PCA?

Use Principal Component Analysis when you want to reduce the number of variables, but don't necessarily know which ones to remove completely.

When to use PCA?

Use Principal Component Analysis when you want to make sure your variables are independent of one another.

When to use PCA?

Use Principal Component Analysis when you are okay with making your independent variables less interpretable.

Let's try it out!

Song Features

Spotify has an API that allows us to access song features. The features that we are going to isolate are **energy** and **loudness**.

Step 1: Create the Training Set

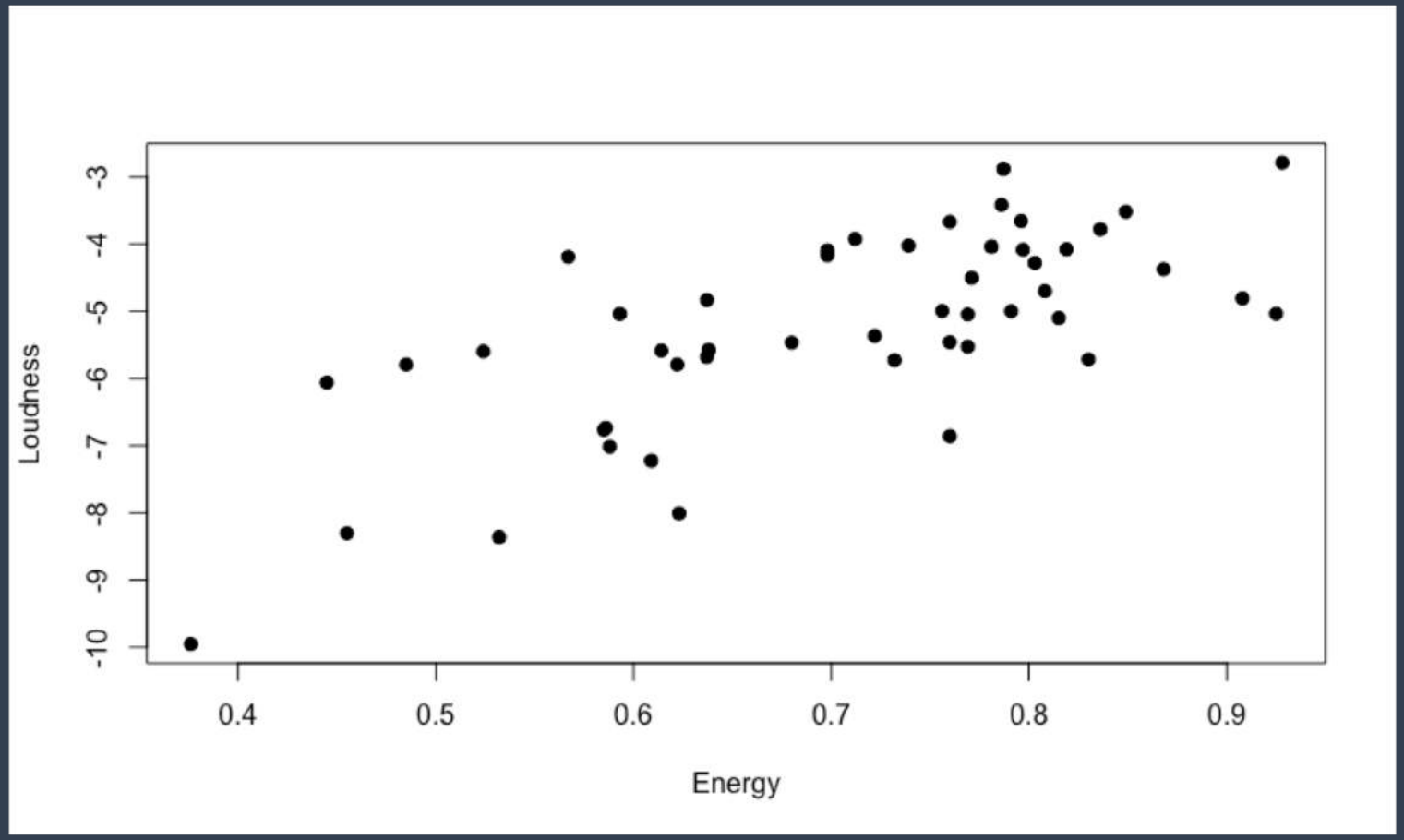
```
> source("/Users/crow/Repositories/making-faces/song example.R")
```

	name	danceability	energy	key	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo
1	Love The Way You Lie	0.749	0.925	10	-5.034	0.2270	0.24100	0.00e+00	0.5200	0.641	86.989
2	Hymn For The Weekend - Seeb Remi	0.565	0.849	0	-3.516	0.0517	0.00868	4.68e-06	0.1200	0.427	102.034
3	Uptown Funk	0.856	0.609	0	-7.223	0.0824	0.00801	8.15e-05	0.0344	0.928	114.988
4	We Can't Stop	0.613	0.622	1	-5.794	0.0334	0.00882	0.00e+00	0.3700	0.484	80.003
5	This Is What You Came For	0.630	0.928	9	-2.787	0.0331	0.19900	1.24e-01	0.1480	0.465	123.963
6	One Last Time	0.628	0.593	8	-5.036	0.0323	0.09300	1.65e-06	0.0960	0.104	125.026
7	Chandelier	0.399	0.787	1	-2.880	0.0499	0.01970	6.07e-05	0.0685	0.572	117.089
8	Never Forget You	0.583	0.732	11	-5.728	0.0457	0.00312	9.86e-06	0.2690	0.276	145.992
9	Cool for the Summer	0.584	0.614	5	-5.587	0.0363	0.00506	9.70e-05	0.0802	0.280	114.075
10	Lush Life	0.694	0.712	7	-3.923	0.0460	0.13300	0.00e+00	0.2110	0.799	98.022
11	Thinking Out Loud	0.781	0.445	2	-6.061	0.0295	0.47400	0.00e+00	0.1840	0.591	78.998
12	You Know You Like It	0.426	0.722	5	-5.369	0.1960	0.01130	1.27e-06	0.2620	0.250	196.133
13	Unforgettable	0.726	0.769	6	-5.043	0.1230	0.02930	1.01e-02	0.1040	0.733	97.985
14	Elastic Heart	0.421	0.791	9	-4.998	0.0496	0.01170	1.48e-05	0.1460	0.499	130.075
15	I Really Like You	0.622	0.808	5	-4.701	0.0418	0.00543	5.63e-05	0.2150	0.599	122.121
16	Closer	0.748	0.524	8	-5.599	0.0338	0.41400	0.00e+00	0.1110	0.661	95.010
17	Love Yourself	0.607	0.376	4	-9.954	0.4530	0.85600	0.00e+00	0.2850	0.545	102.541
18	Cake By The Ocean	0.771	0.760	4	-5.459	0.0514	0.14700	0.00e+00	0.0530	0.881	119.000
19	Budapest	0.717	0.455	5	-8.303	0.0276	0.08460	0.00e+00	0.1100	0.389	127.810
20	Sorry	0.654	0.760	0	-3.669	0.0450	0.07970	0.00e+00	0.2990	0.410	99.945
21	Bang Bang	0.706	0.786	0	-3.417	0.0909	0.26000	0.00e+00	0.3800	0.749	150.035
22	Starboy	0.678	0.588	7	-7.015	0.2760	0.14100	6.35e-06	0.1370	0.486	186.005

Step 1: Create the Training Set

```
songs = read.csv('songdata.csv')[c('energy', 'loudness')]
training_set = t(data.matrix(songs))
```

Step 1: Song Space

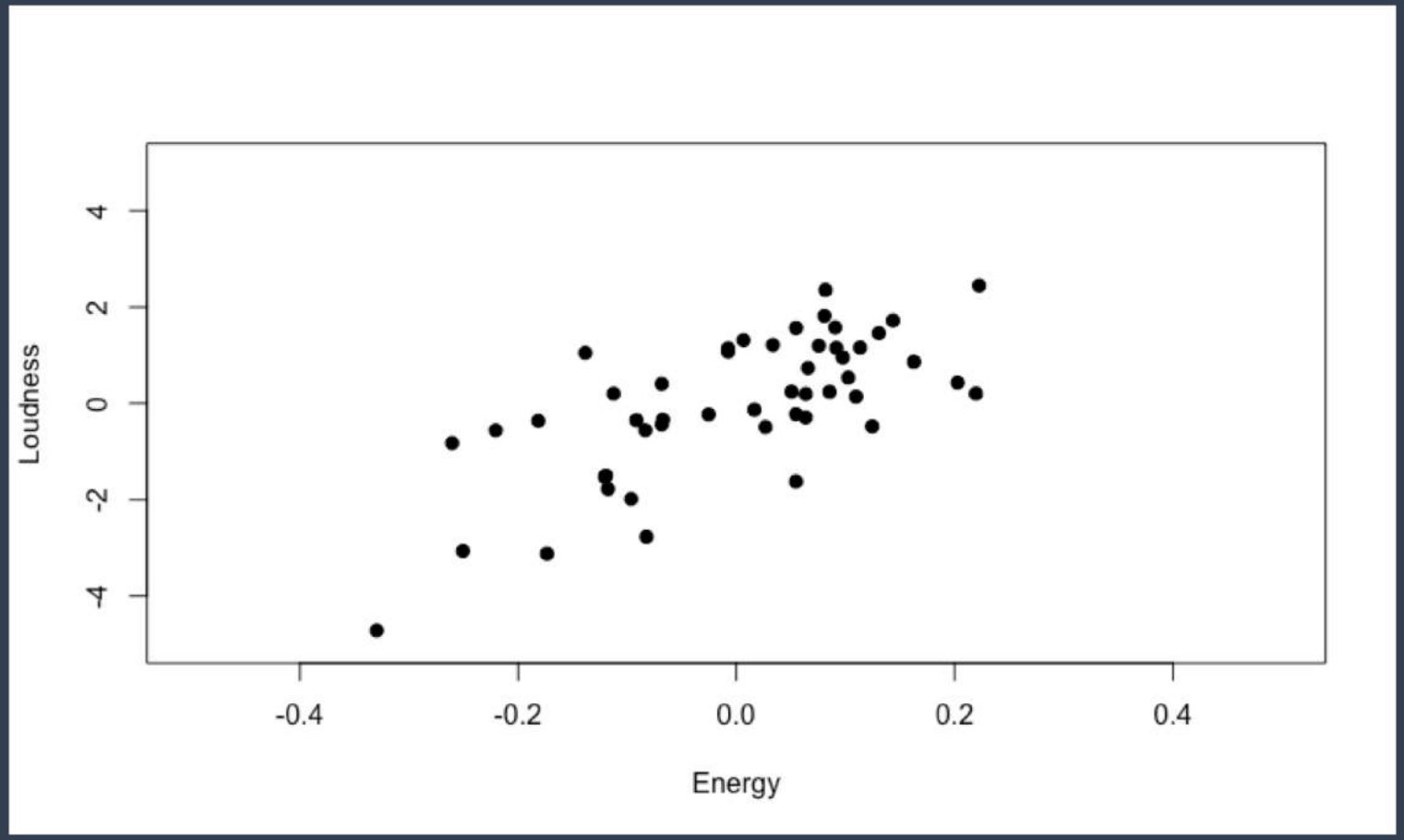


Step 2: Normalize the Data

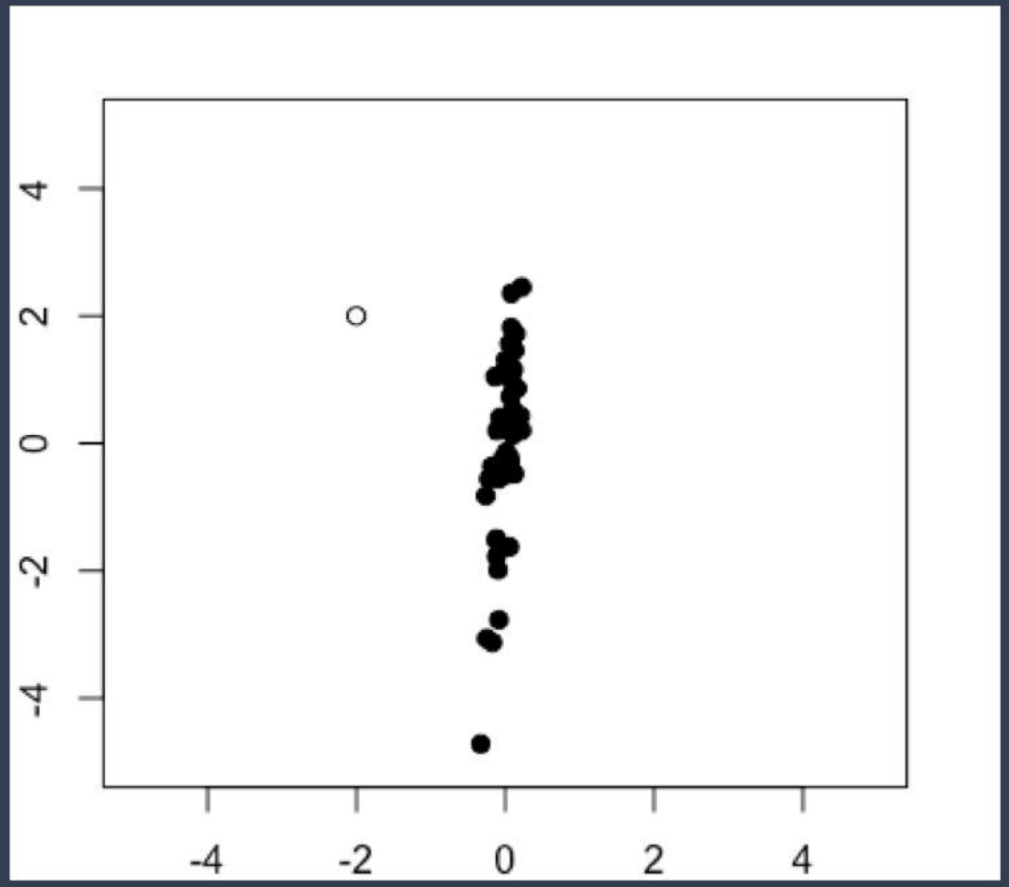
- Find the average energy and loudness
- Subtract from each energy and loudness to normalize the data.

```
centered = training_set - rowMeans(training_set)
```

Step 2: Normalize the Data

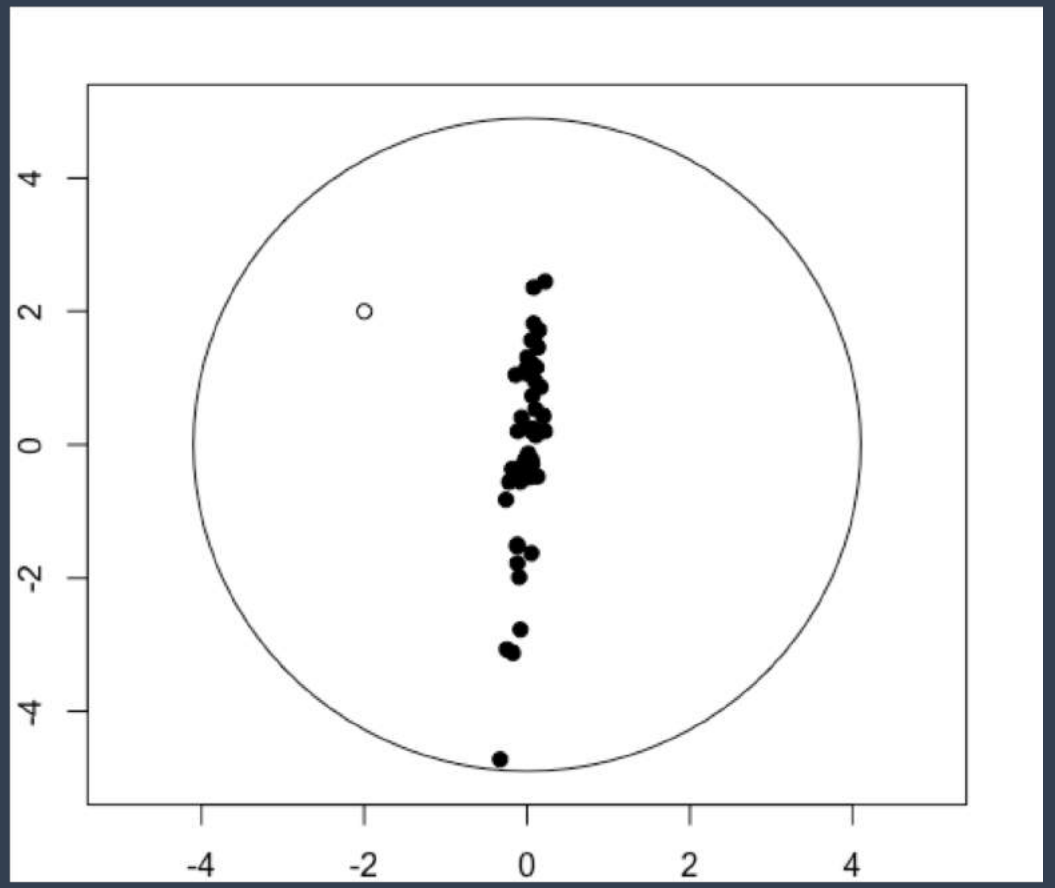


Does the song belong on the playlist?



Create a Boundary

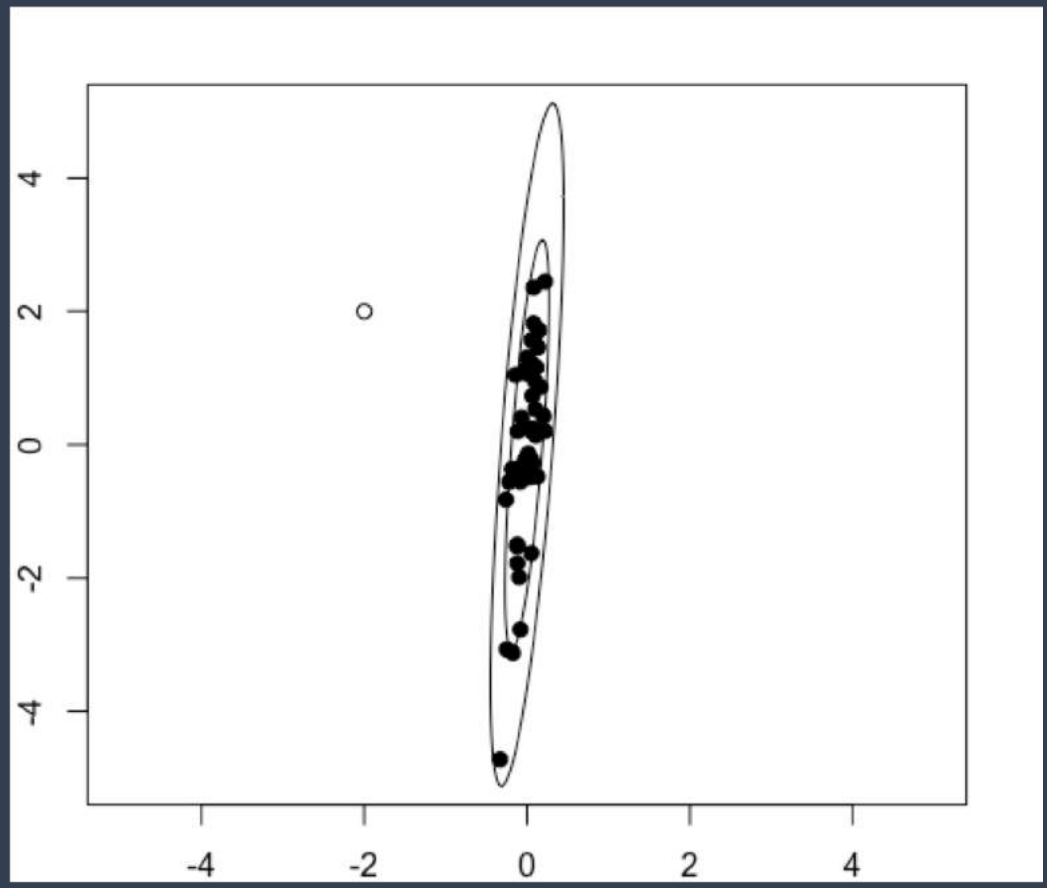
- If the point is inside of the circle then the song belongs on the playlist.
- Is there something wrong with this?



@crOwst Steve Crow steve.crow@nexmo.com

Create a Better Boundary

- If the point is inside of the ellipse then the song belongs on the playlist.
- The axes of the ellipse point to the direction of least and most variance.
- The size of the ellipse depends on how confident we want to be in fitting the data.



The Singular Value Decomposition

One of the most known and widely used matrix decompositions. All matrices have an SVD and it is commonly used in compression, denoising, and data reduction.

@crOwst Steve Crow steve.crow@nexmo.com

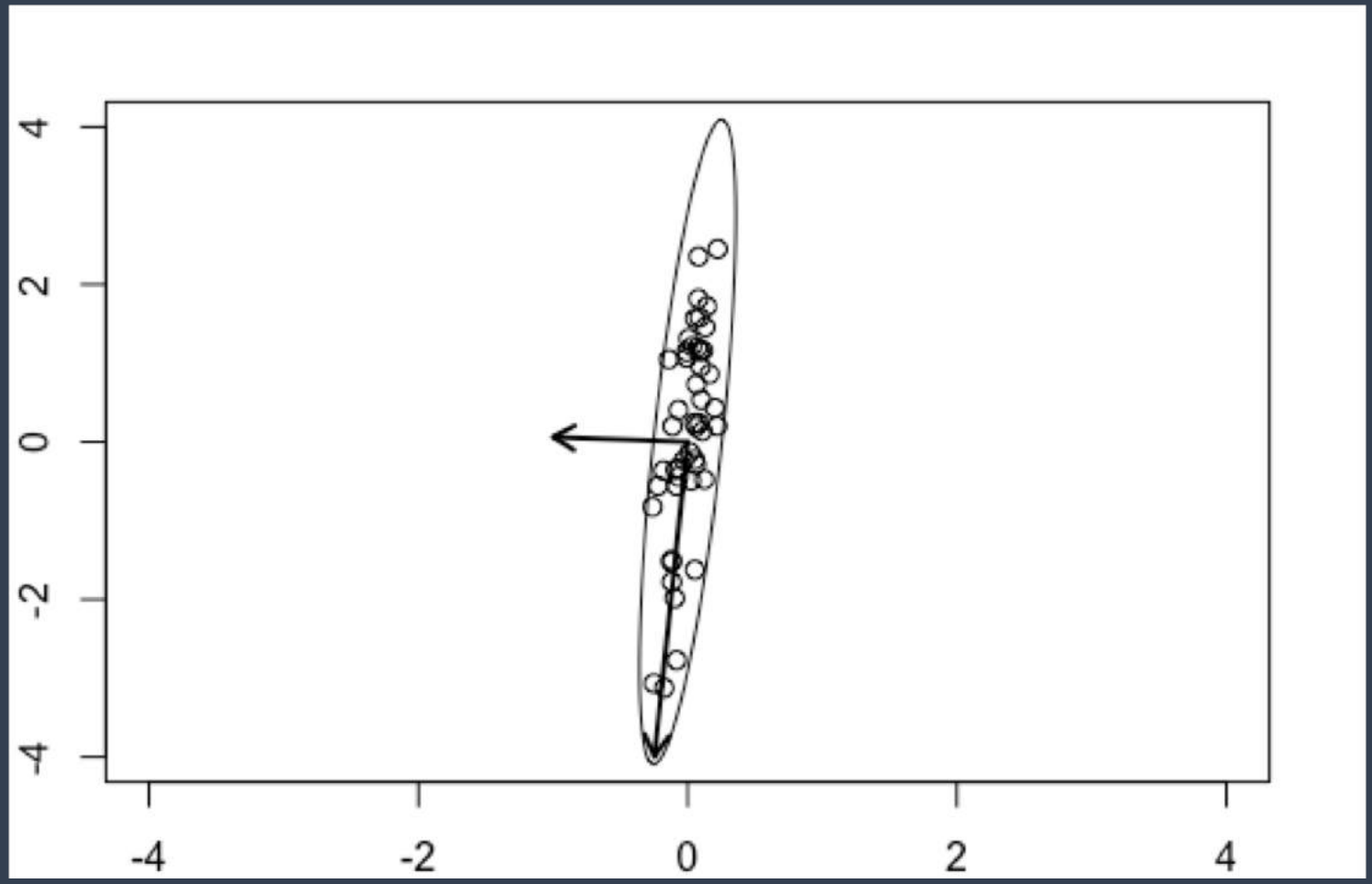
The Singular Value Decomposition

$$A = U\Sigma V^T$$

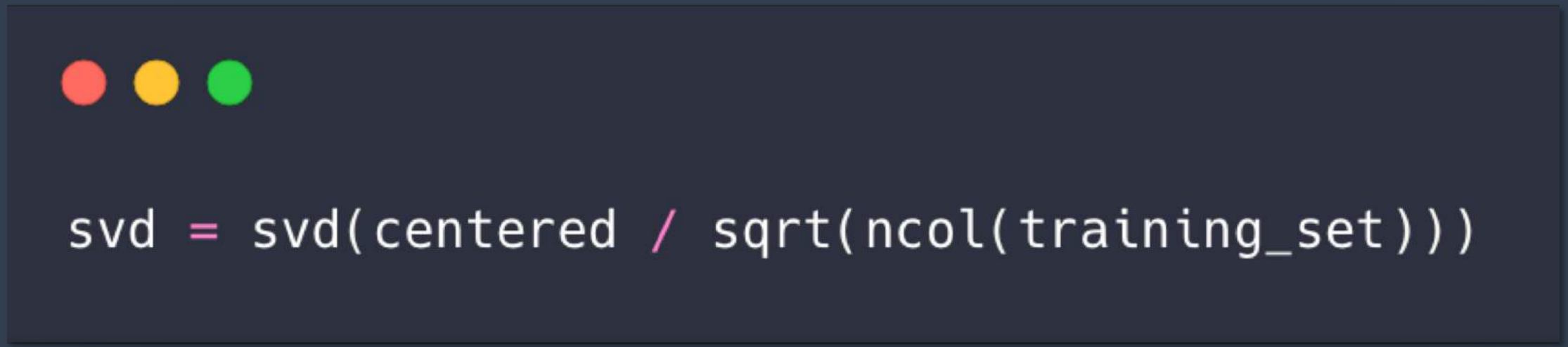
Step 3: The Singular Value Decomposition

- Let M equal our Song Space Matrix and n the number of songs.
- Then $svd\left(\frac{M}{\sqrt{n}}\right) = U\Sigma V^T$
- The columns of U are vectors which point in the direction of greatest to least variance.
- The diagonal values of Σ give insight into how significant each of the principal components are.

Step 3: The Singular Value Decomposition



Step 3: The Singular Value Decomposition



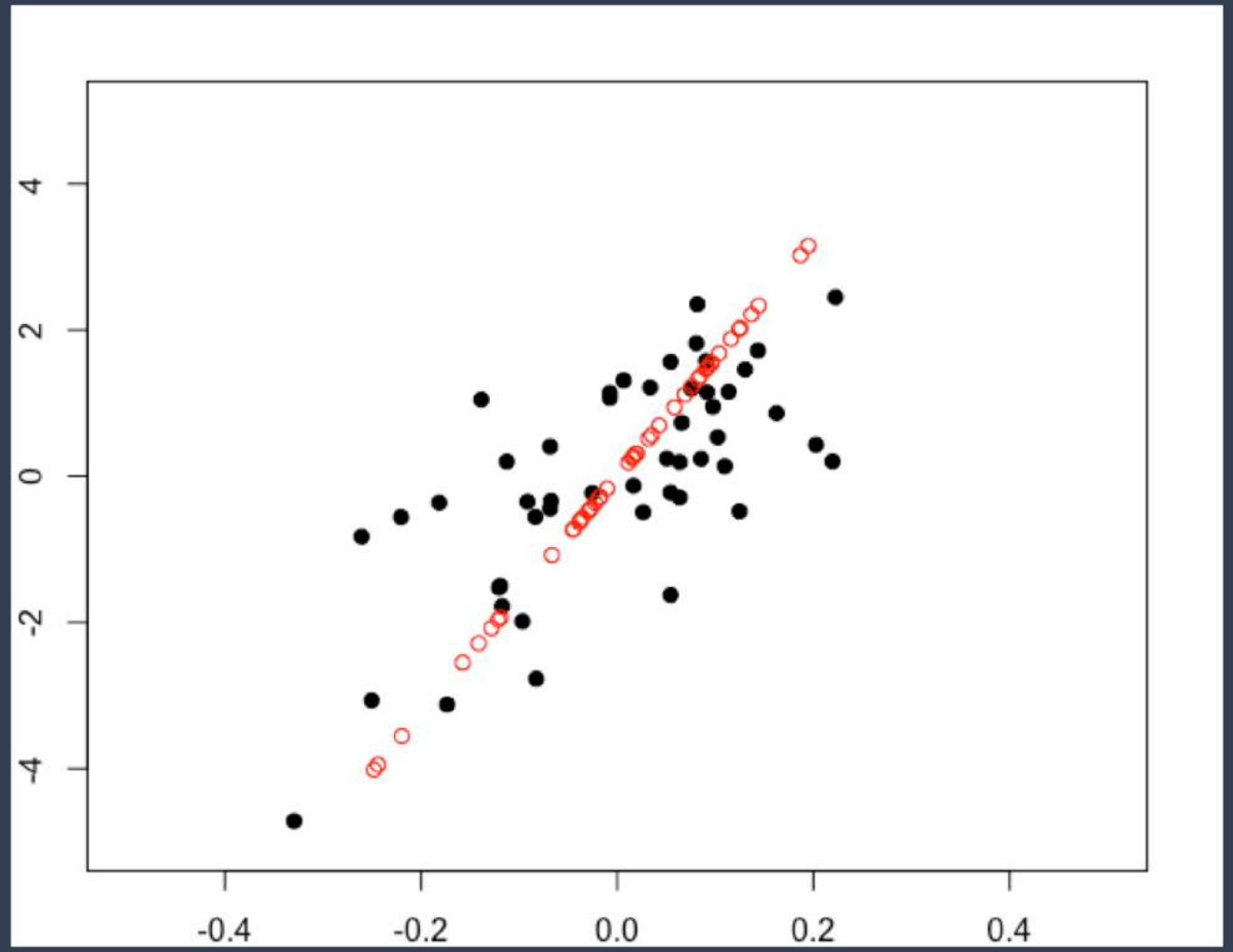
```
svd = svd(centered / sqrt(ncol(training_set)))
```

The Singular Value Decomposition

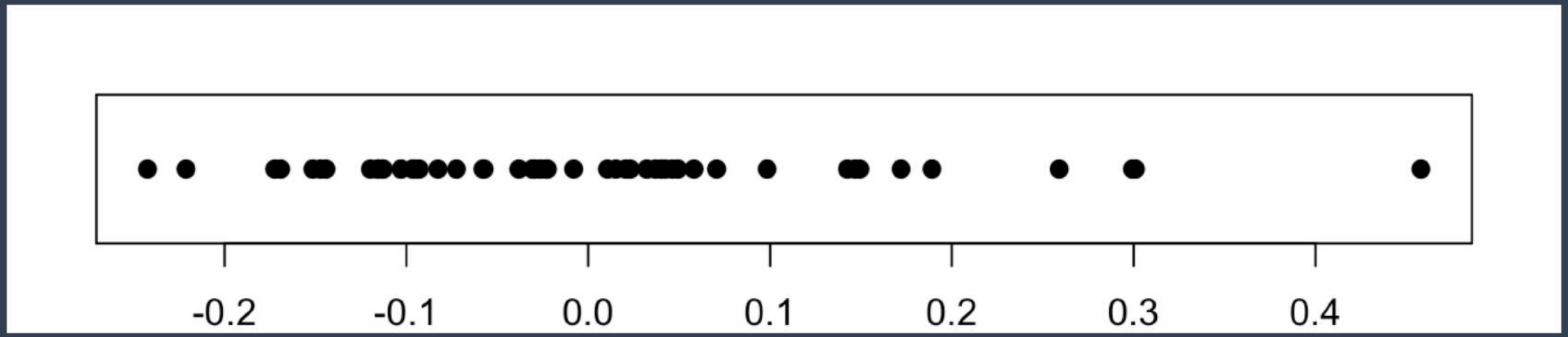
$$A = U\Sigma V^T$$

Matrix Multiplication

One Dimensional Song Data



One Dimensional Song Data



Projection

We use the following formula to project our data onto the ellipse axes:

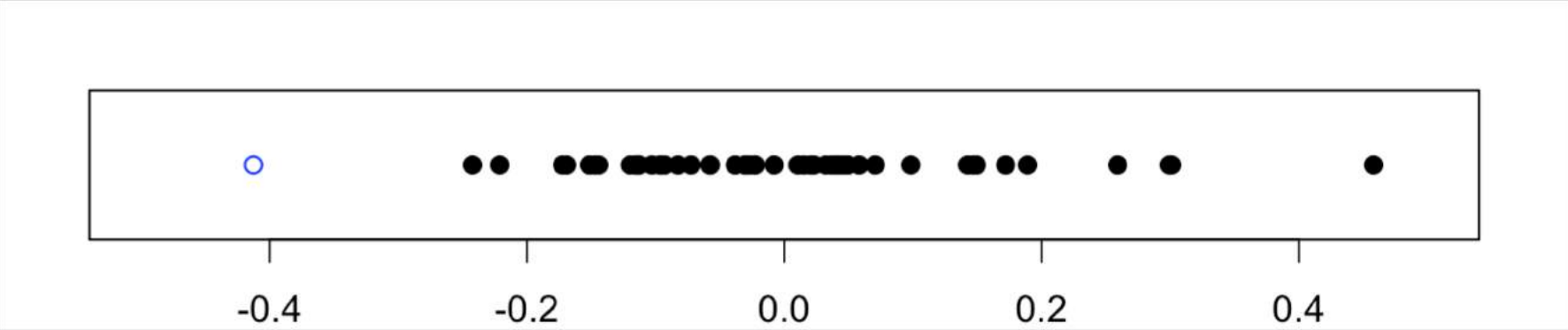
$$P = U\Sigma X$$

```
training_projections = svd$u %*% diag(svd$d) %*% centered
```

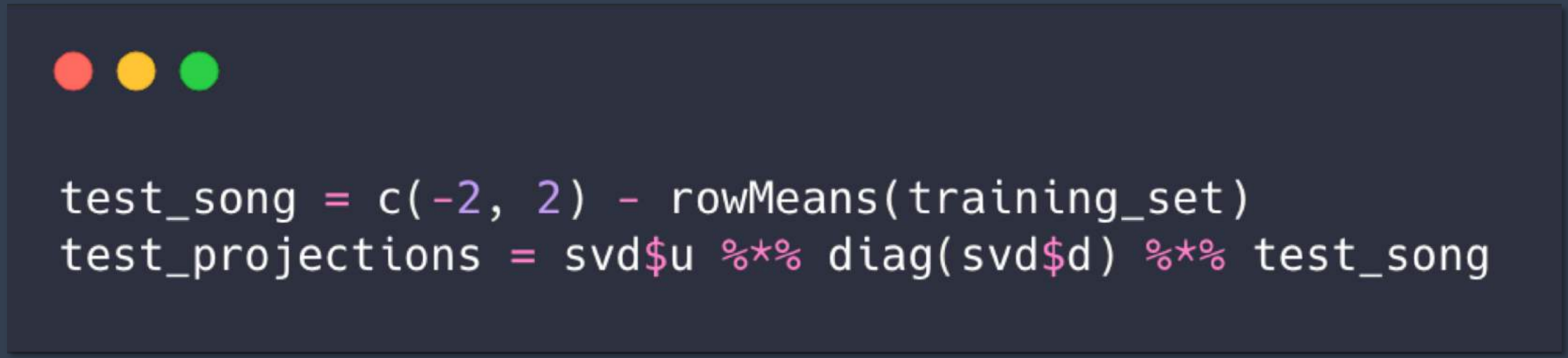
Projections

```
                [,1]      [,2]       [,3]       [,4]
energy    0.2195102 0.1435102 -0.0964898 -0.0834898
loudness  0.2020816 1.7200816 -1.9869184 -0.5579184
             [,1]       [,2]      [,3]       [,4]
PC1 -0.03816334 -0.1690882 0.1890630 0.05818034
PC2 -0.31990593 -0.2002474 0.1299837 0.11897906
```

Does this song belong?



Step 1: Center and Project the Test Song

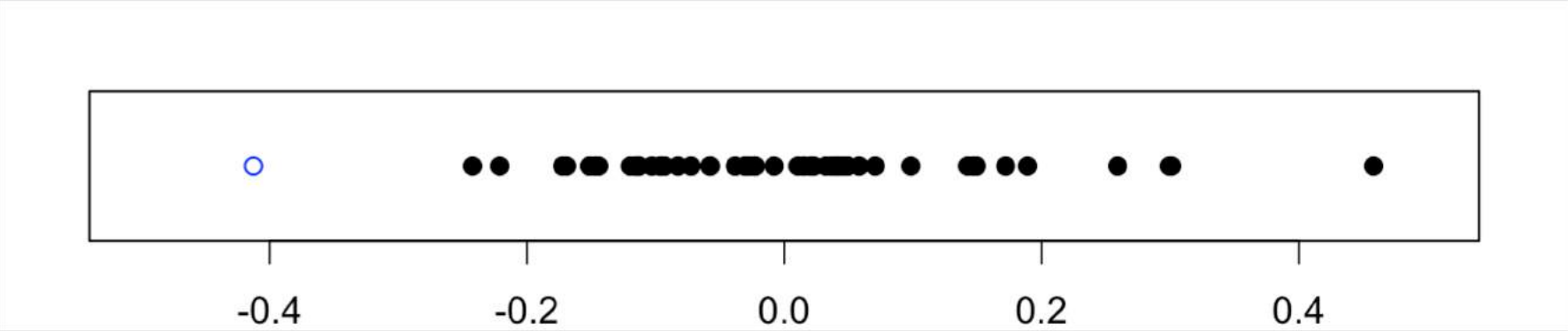


```
test_song = c(-2, 2) - rowMeans(training_set)
test_projections = svd$u %*% diag(svd$d) %*% test_song
```

Step 2: Calculate Distances

```
distances = abs(test_projections[1,] - training_projections[1,])
dist = data.frame(
  "min.dist" = min(distances),
  "min.index" = which(distances == min(distances))
  )
```

Step 3: Interpret Results




```
   min.dist min.index
1 0.1701281         5
```

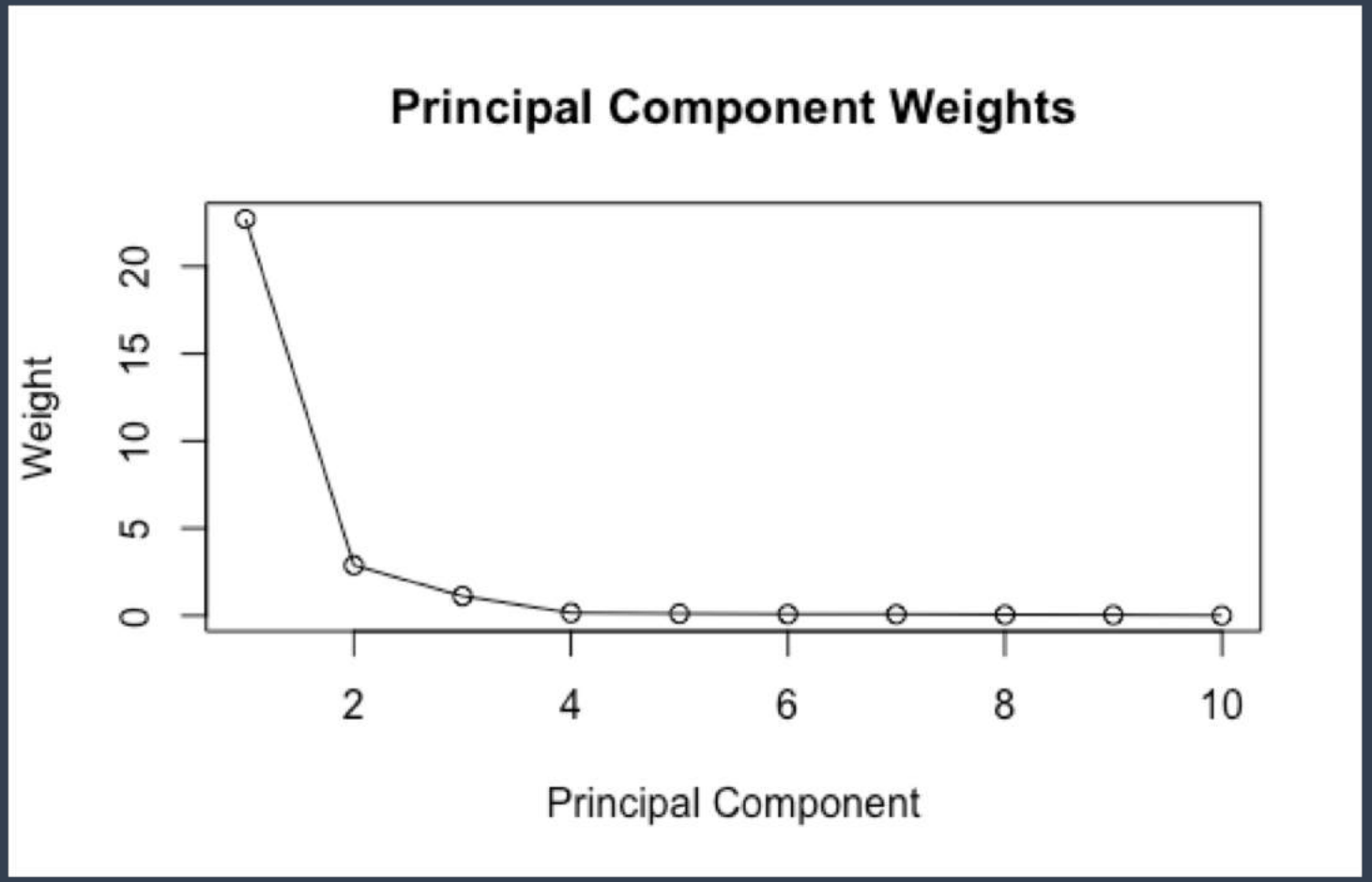
Reminder

The value of PCA is in its ability to reduce the number of dimensions we need to work with.

All Song Features

```
songs = read.csv("songdata.csv")
matrix = t(data.matrix(songs)[,-1])

centered = matrix - rowMeans(matrix)

svd = svd(centered/9)
```

All Song Features



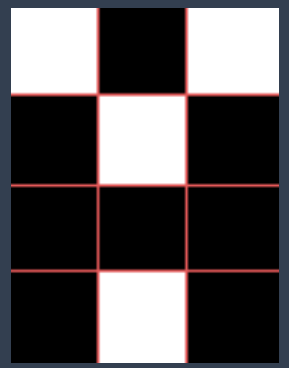
Face Dimensional Reduction

What if the number of dimensions gets even bigger?

@crOwst Steve Crow steve.crow@nexmo.com

Images as Vectors

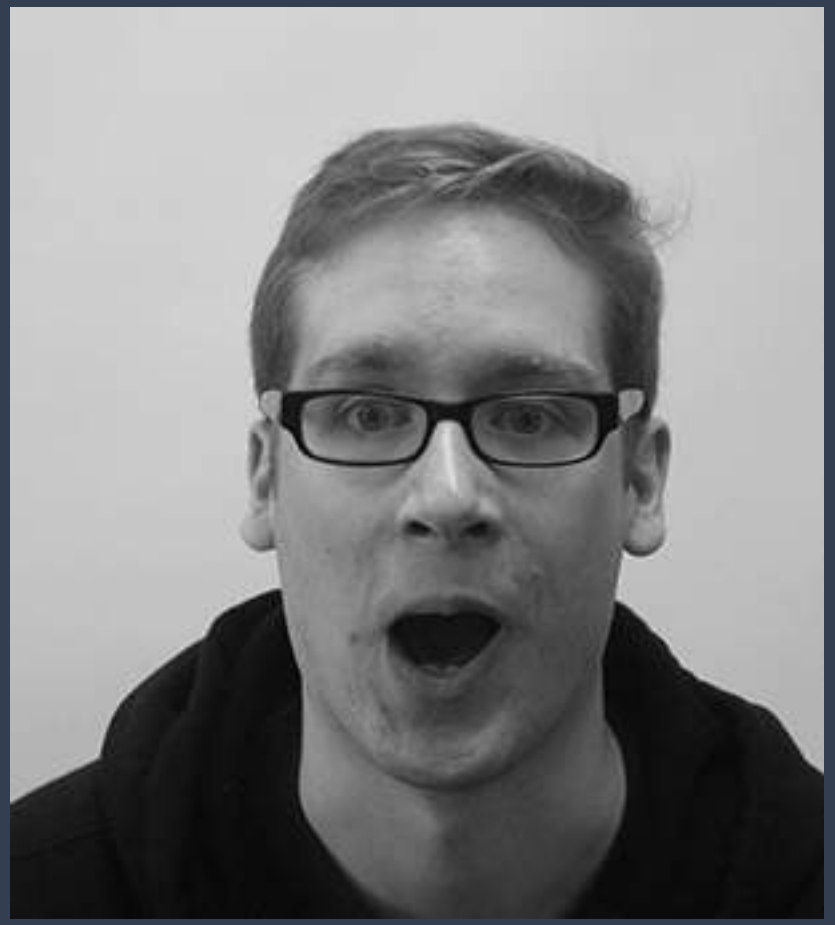
Representing Images as Vectors



This 4 by 3 image becomes a 12 by 1 image.

Each pixel represents a feature.

Face Images



Each image is 342 pixels by 305 pixels.

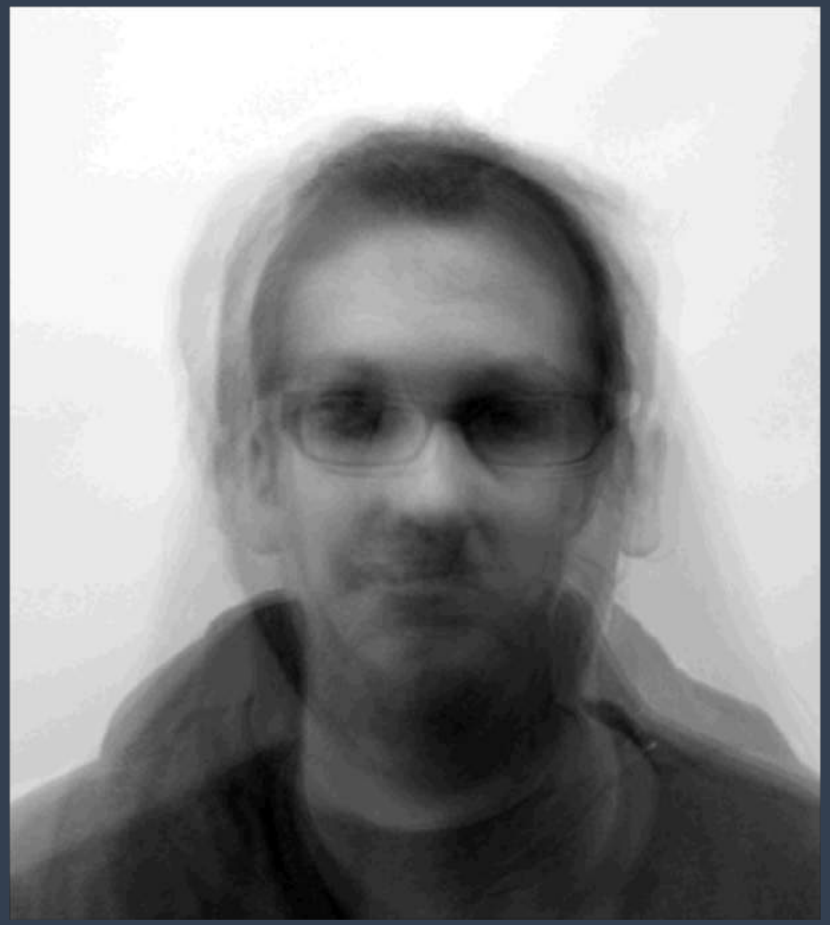
Each image has been normalized to the best of my abilities.

This will produce a vector that is 104310 pixels long.

That's 104310 features to consider.

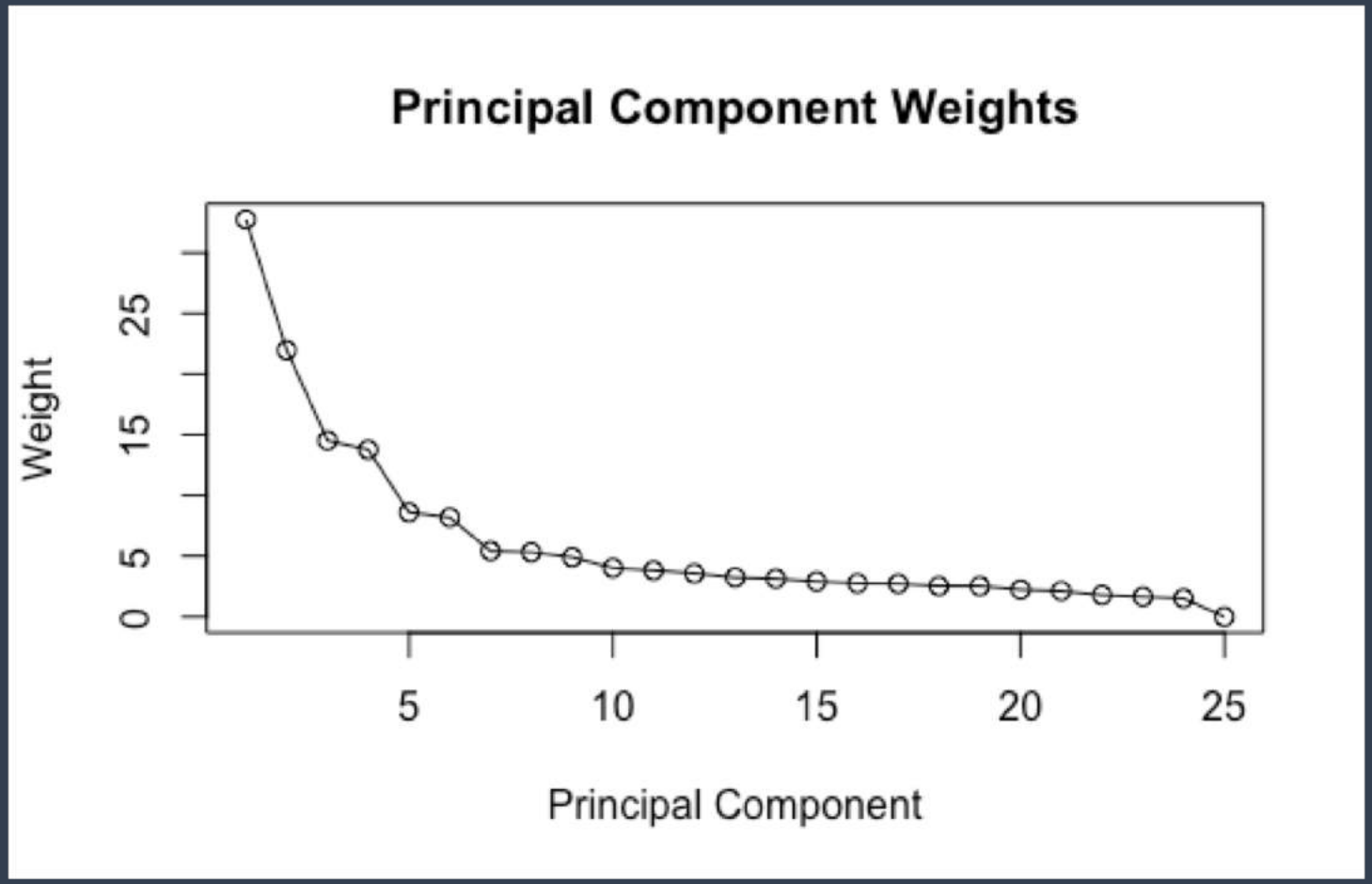
The Training Set

Normalize the Training Set



Normalize the Training Set

Principal Component Weights



Principal Component Vectors

What does that mean?

The Projections

Let's Test the Model

Image In Set

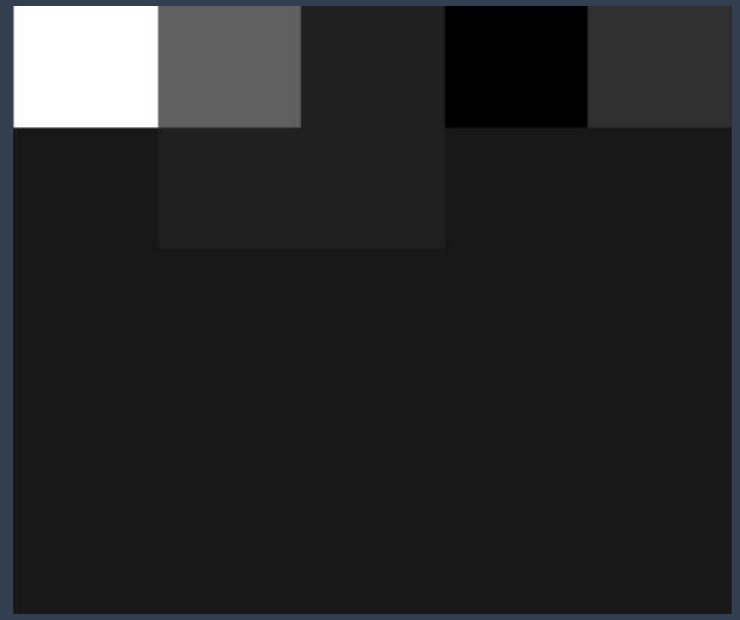
Person In Set

Not In Set

Projections

Image In Set

Person In Set



Not In Set

@crOwst

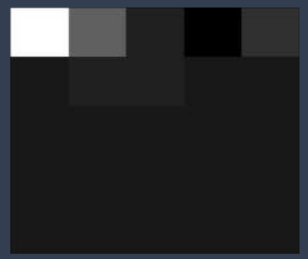
Steve Crow

steve.crow@nexmo.com

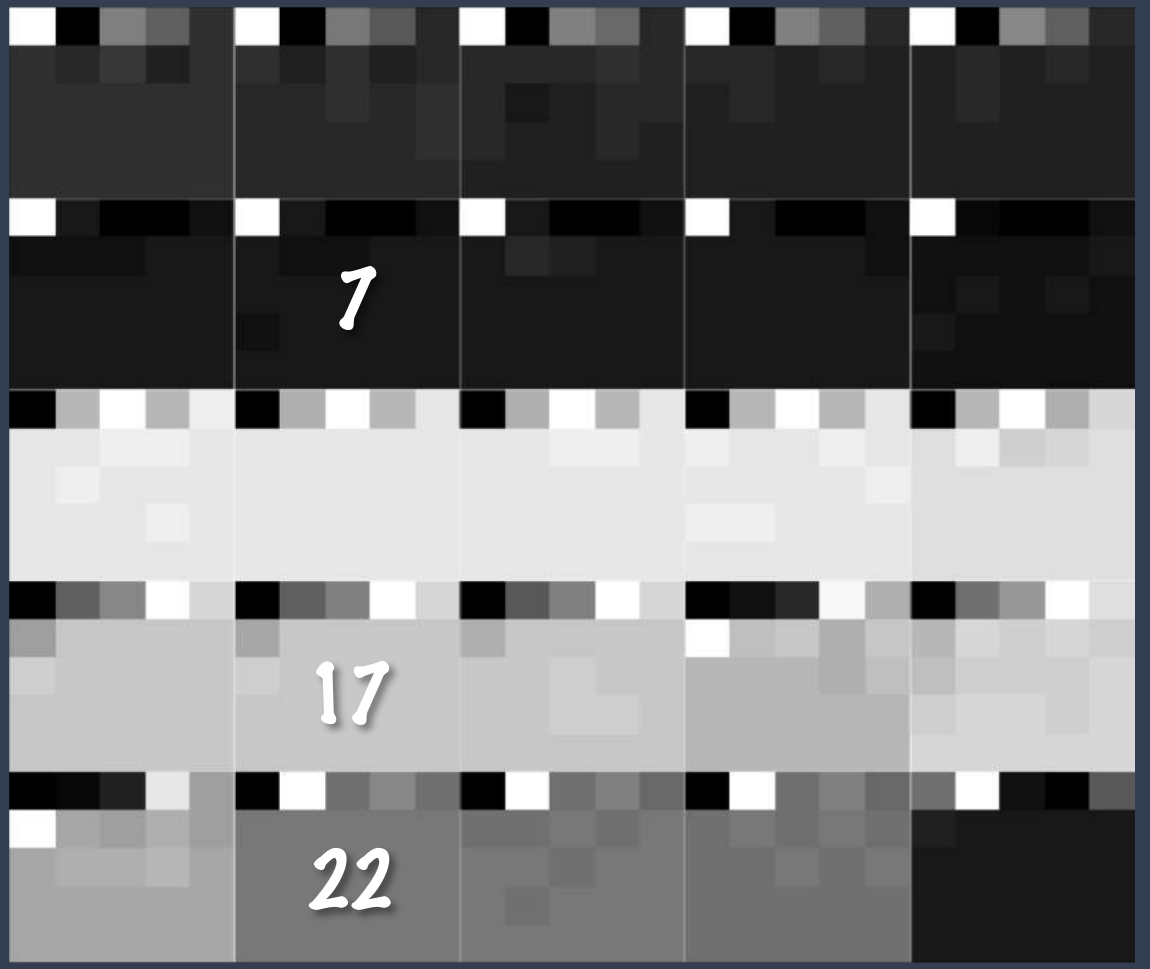
Side by Side Comparison

Image In Set
Closest to Image 17
Distance: 0

Person In Set
Closest to Image 22
Distance: 35.98396



Person Not In Set
Closest to Image 7
Distance: 527.8126



Side by Side Comparison

Image In Set
Closest to Image 17
Distance: 0

Person In Set
Closest to Image 22
Distance: 35.98396

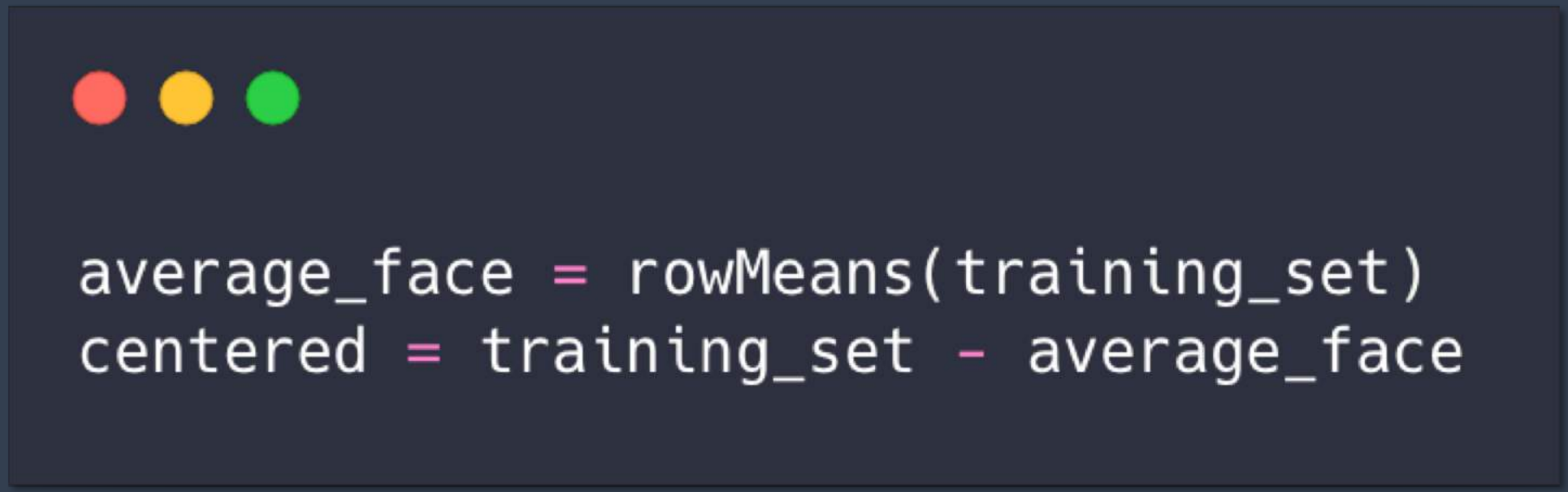
Person Not In Set
Closest to Image 7
Distance: 527.8126

Here's the Process

Step 1: Build the Training Set

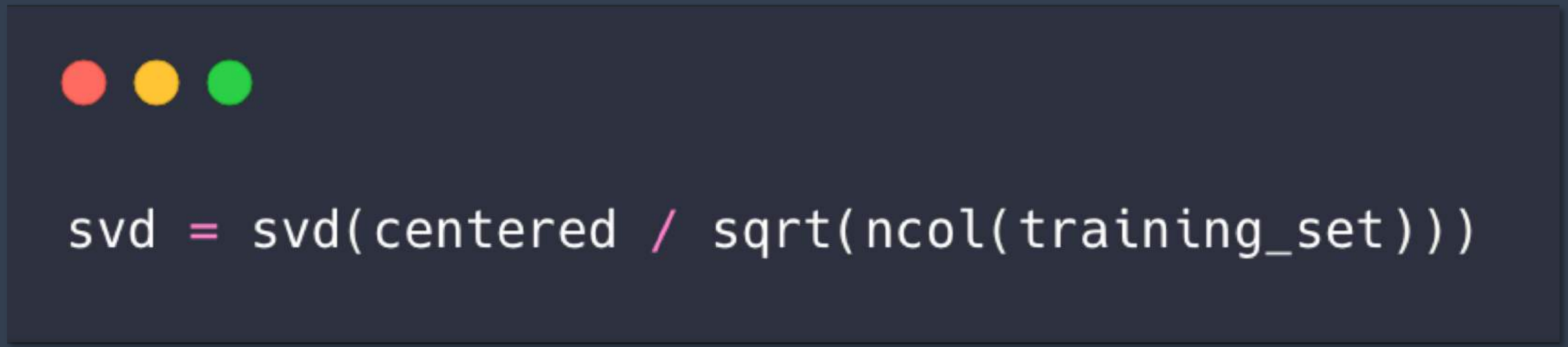
```
training_set_files = list.files(path = 'training_set', pattern = '.jpg', full.names = TRUE)
training_set_faces = map_il(training_set_files, load.image)
training_set = matrix(
  unlist(lapply(training_set_faces, function(x) x[, , , 1])),
  ncol = length(training_set_faces),
  byrow = FALSE)
```

Step 2: Mean Normalization



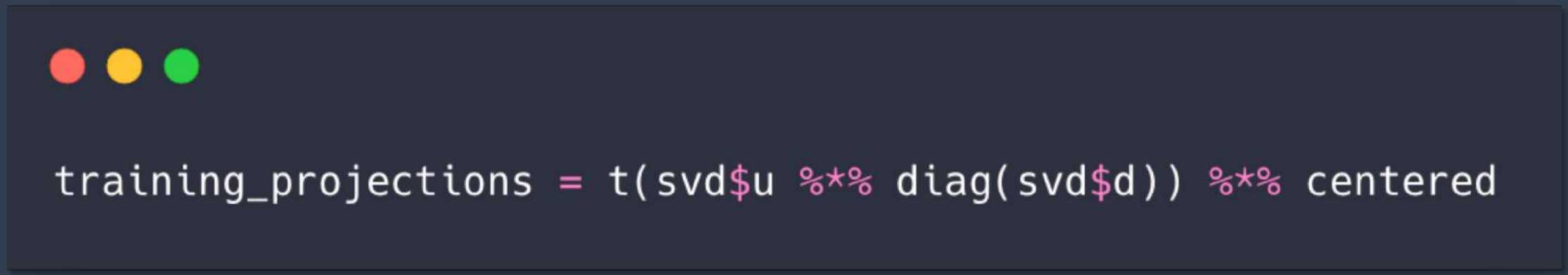
```
average_face = rowMeans(training_set)
centered = training_set - average_face
```

Step 3: Singular Value Decomposition



```
svd = svd(centered / sqrt(ncol(training_set)))
```

Step 4: Calculate the Projections



```
training_projections = t(svd$u %*% diag(svd$d)) %*% centered
```

All Together Now

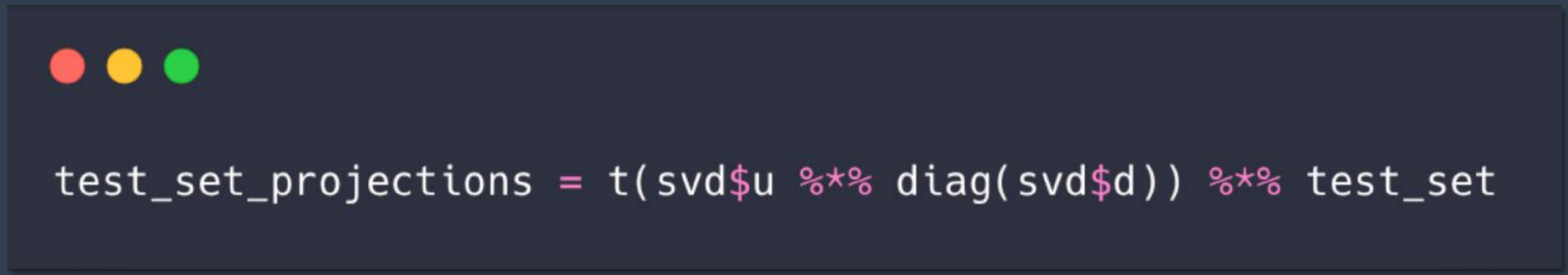
```
# Step 1 Create Training Set
training_set_files = list.files(path = 'training_set', pattern = '.jpg', full.names = TRUE)
training_set_faces = map_il(training_set_files, load.image)
training_set = matrix(
  unlist(lapply(training_set_faces, function(x) x[, , , 1])),
  ncol = length(training_set_faces),
  byrow = FALSE)

# Step 2 Perform Mean Normalization
average_face = rowMeans(training_set)
centered = training_set - average_face

# Step 3 Perform SVD
svd = svd(centered / sqrt(ncol(training_set)))

# Step 4 Project the Training Set Into Low-Rank Dimension
training_projections = t(svd$u %*% diag(svd$d)) %*% centered
```

Test The Model

Step 1: Create Test Set

```
test_files = list.files(path = 'test_set', pattern = '.jpg', full.names = TRUE)
test_faces = map_il(test_files, load.image)
test_set = matrix(
  unlist(lapply(test_faces, function(x) x[, , , 1] - average_face)),
  ncol = 3,
  byrow = FALSE)
```

Step 2: Project the Test Set

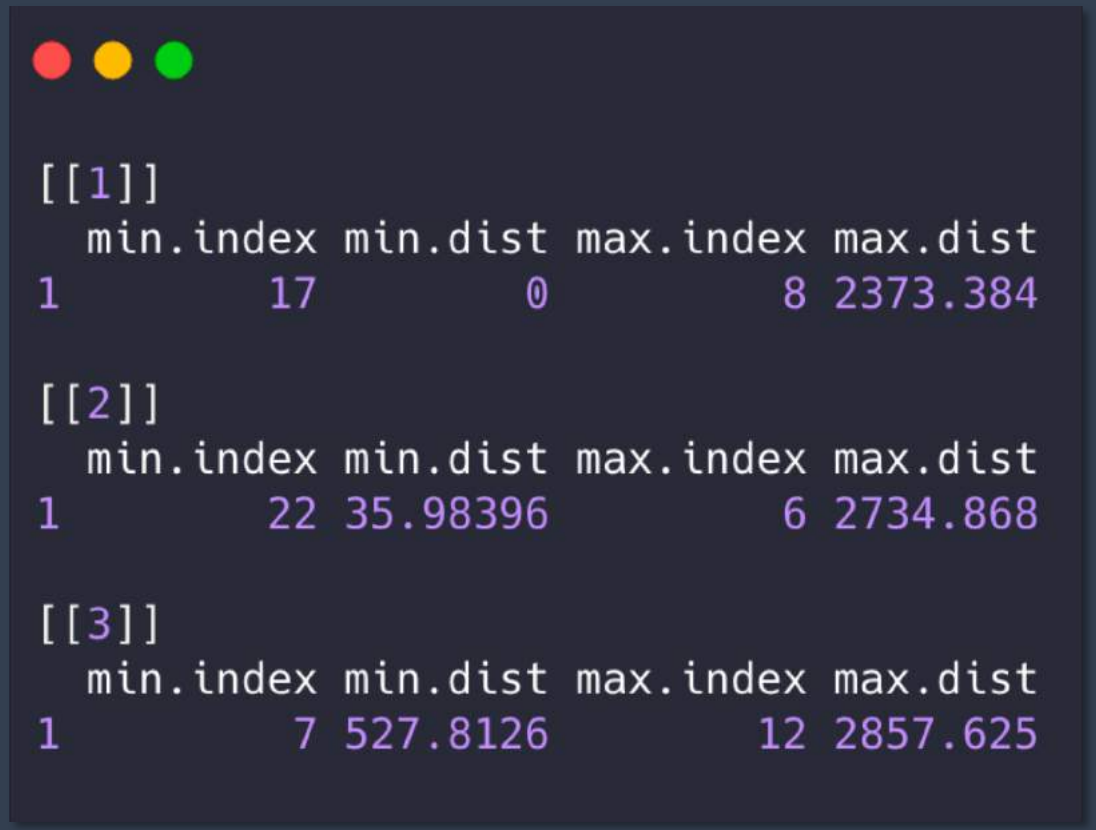


```
test_set_projections = t(svd$u %*% diag(svd$d)) %*% test_set
```

Step 3: Calculate the Distances

```
distances = apply(test_set_projections, 2, function(test) {
  dist = apply(training_projections, 2, function(x) sqrt(sum((x - test) ^ 2)))
  data.frame(
    "min.index" = which(dist == min(dist)),
    "min.dist" = min(dist),
    "max.index" = which(dist == max(dist)),
    "max.dist" = max(dist)
  )
})
```

Step 4: See the Results



```
[[1]]
  min.index min.dist max.index max.dist
1        17        0         8 2373.384

[[2]]
  min.index min.dist max.index max.dist
1        22 35.98396         6 2734.868

[[3]]
  min.index min.dist max.index max.dist
1         7 527.8126        12 2857.625
```

We Have Problems

@crOwst

Steve Crow

steve.crow@nexmo.com

Steve Crow

Java Developer Advocate
Nexmo

@cr0wst

@crOwst Steve Crow steve.crow@nexmo.com

Links and Stuff

https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/

https://github.com/cr0wst/making-faces