MACHINE LEARNING 2

ADVANCED CRIME ANALYSIS

UCL

BENNETT KLEINBERG

25 FEB 2019

MACHINE LEARNING 2

TODAY

- Recap supervised machine learning
- UNsupervised ML
  - Step-by-step example
- Performance metrics
- Validation and generalisation

RECAP SUPERVISED ML

- supervised = labeled data
  - classification (e.g. death/alive, fake/real)
  - regression (e.g. income, number of deaths)
- step-wise procedure

STEPS IN SUPERVISED ML

- clarify what `outcome` and `features` are
- determine which classification algorithm to use
- train the model
  - train/test split
  - cross-validation
- fit the model

UNSUPERVISED ML

- often we don’t have labelled data
- sometimes there are no labels at all
- core idea: finding clusters in the data

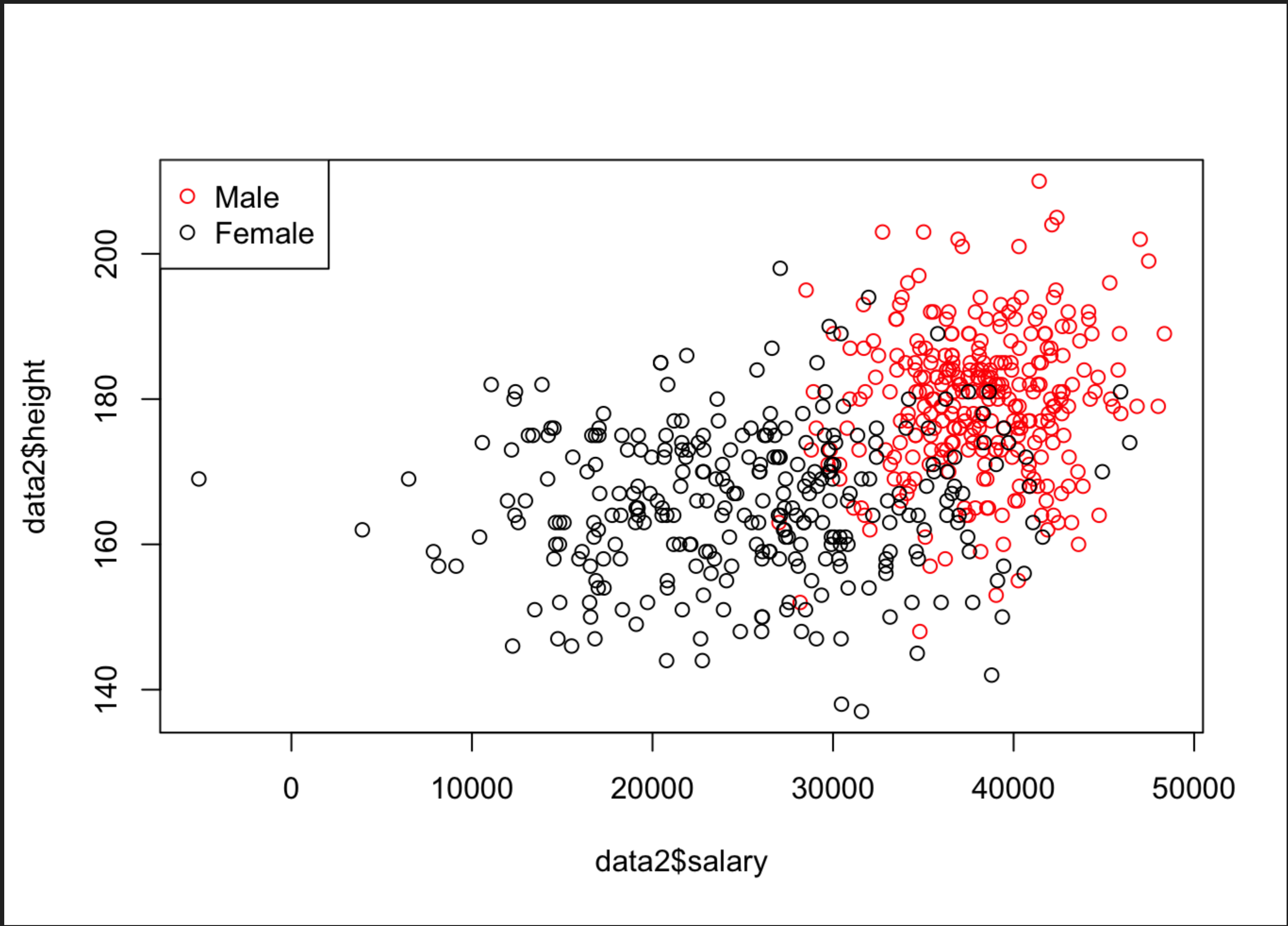
```
library(caret)
```

EXAMPLES

- grouping of online ads
- clusters in crime descriptions
- …

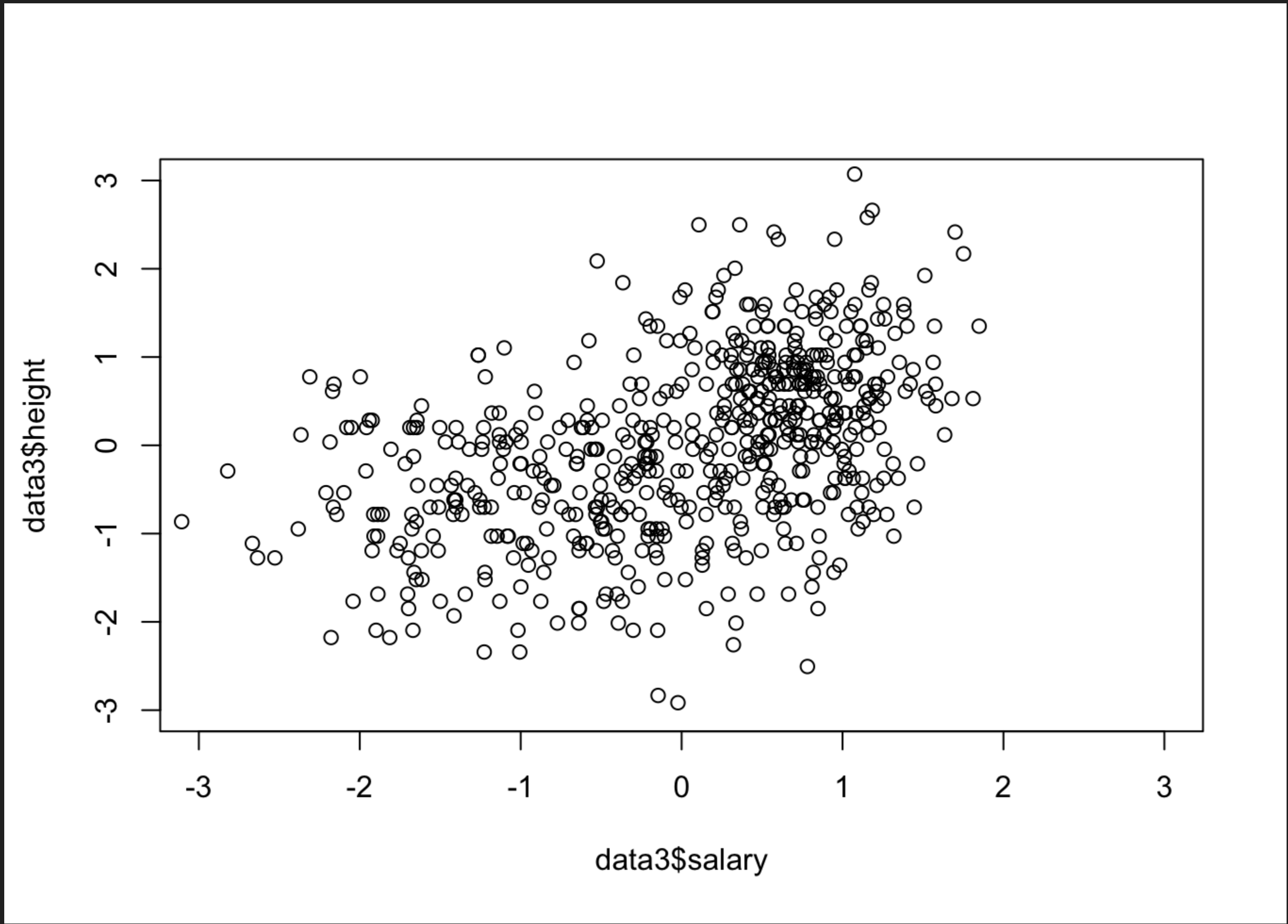
Practically everywhere.

Clustering reduces your data!


Male
Female
data2$height
200
180
160
140
0
10000
20000
30000
40000
50000
data2$salary

THE UNSUPERVISED CASE

You know nothing about groups inherent to the data.



THE K-MEANS IDEA

- separate data in set number of clusters
- find best cluster assignment of observations

STEPWISE

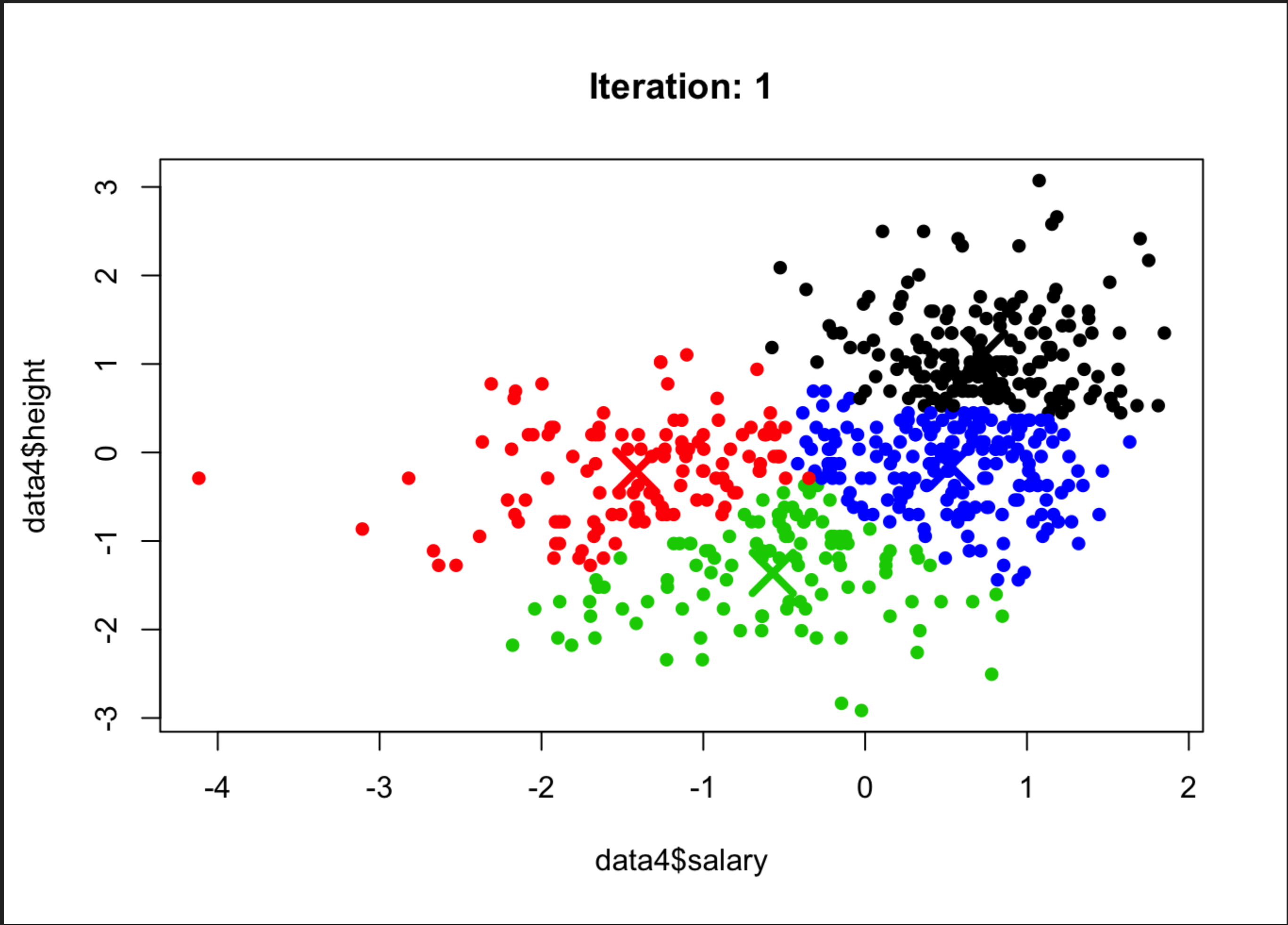
1. set the number of clusters
2. find best cluster assignment

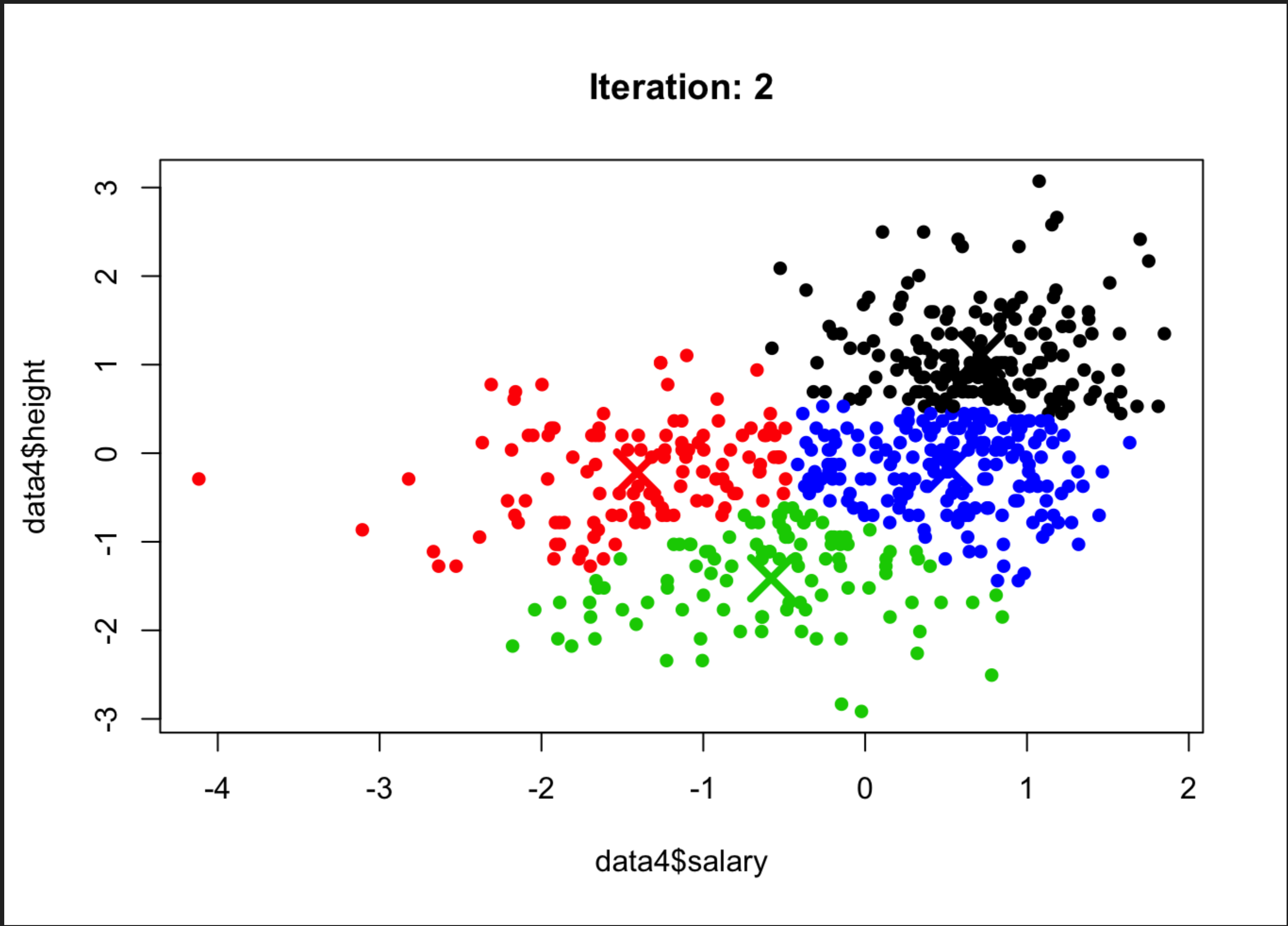
1. NO. OF CLUSTERS

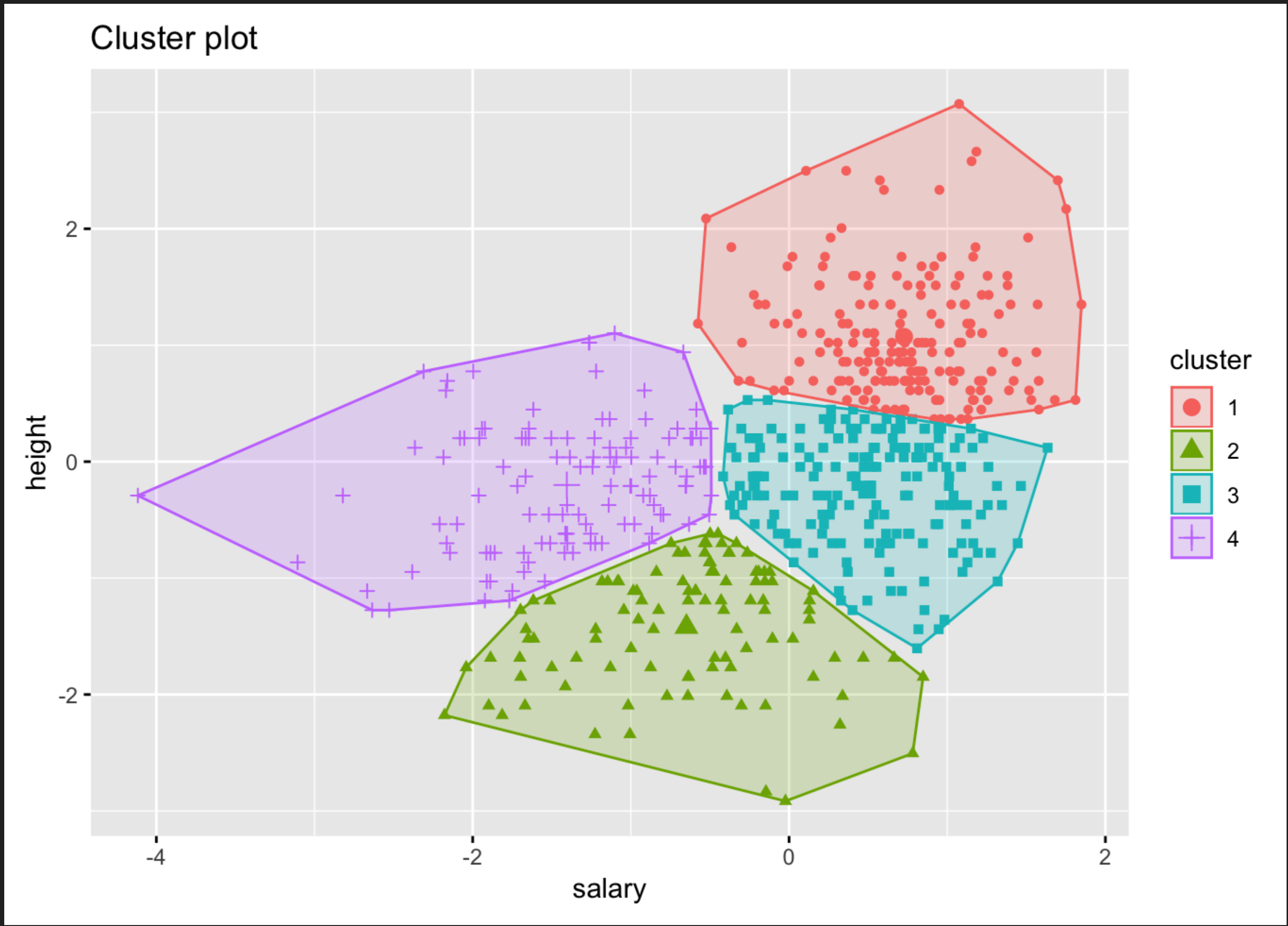
Let’s take 4.

```
unsup_model_1 = kmeans(data4
                       , centers = 4
                       , nstart = 10
                       , iter.max = 10)
```

WHAT'S INSIDE?




Iteration: 2
data4$height
3
2
1
0
-1
-2
-3
-4
-3
-2
-1
0
1
2
data4$salary



Cluster plot
height
salary
cluster
1
2
3
4
2
0
-2
-4
-2
0
2

THE K-MEANS ALGORITHM

- find random centers
- assign each observation to its closest center
- optimise for the WSS

WHAT'S PROBLEMATIC HERE?

BUT HOW DO WE KNOW HOW MANY CENTERS?

Possible approach:

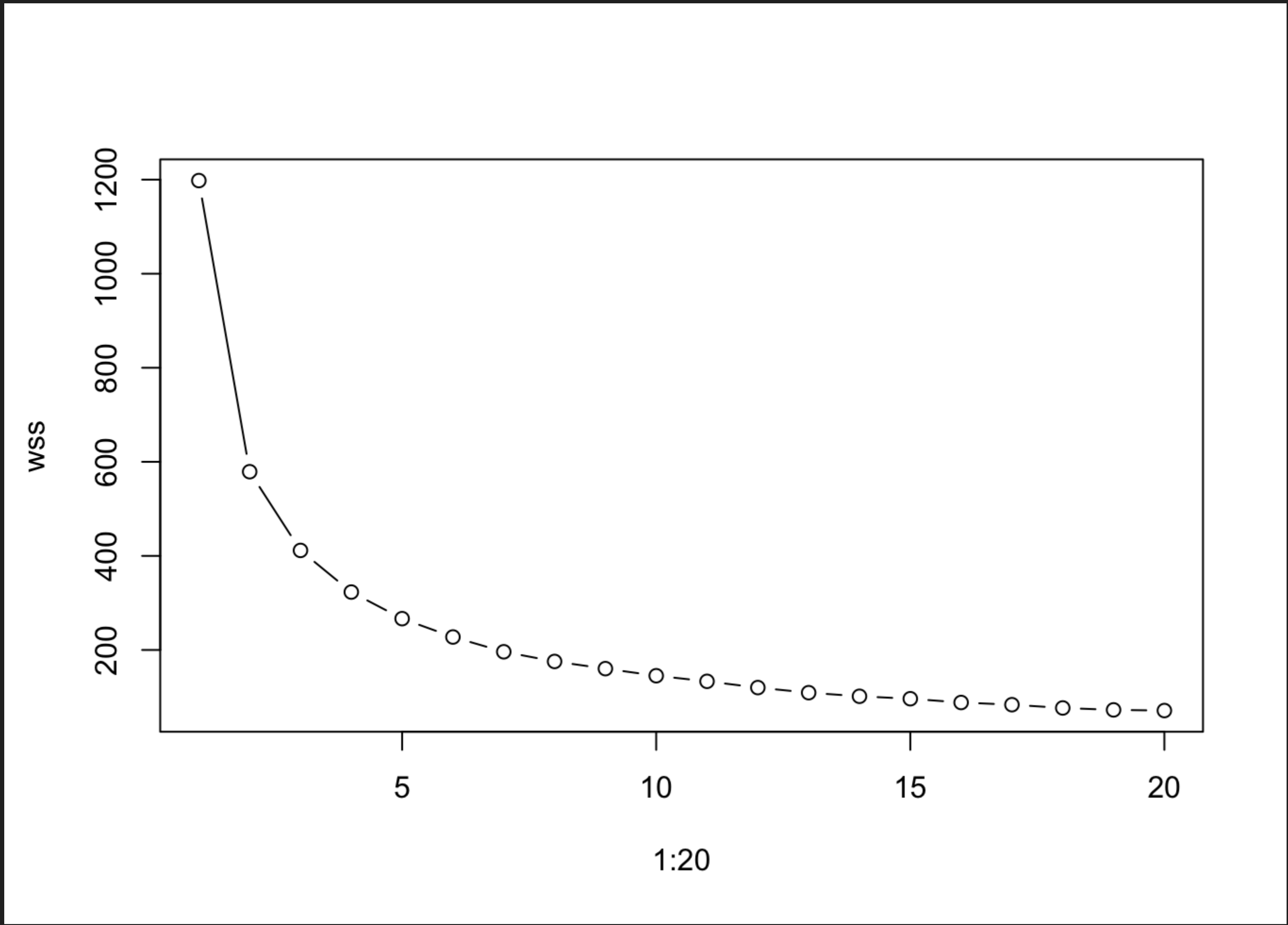
- run it for several combinations
- assess the WSS
- determine based on scree-plot

CLUSTER DETERMINATION

```
wss = numeric()
for(i in 1:20){
  kmeans_model = kmeans(data4, centers = i, iter.max = 20, nstart = 1
  wss[i] = kmeans_model$tot.withinss
}
```

SCREE PLOT (ELBOW METHOD)

Look for the inflexion point at center size *i*.

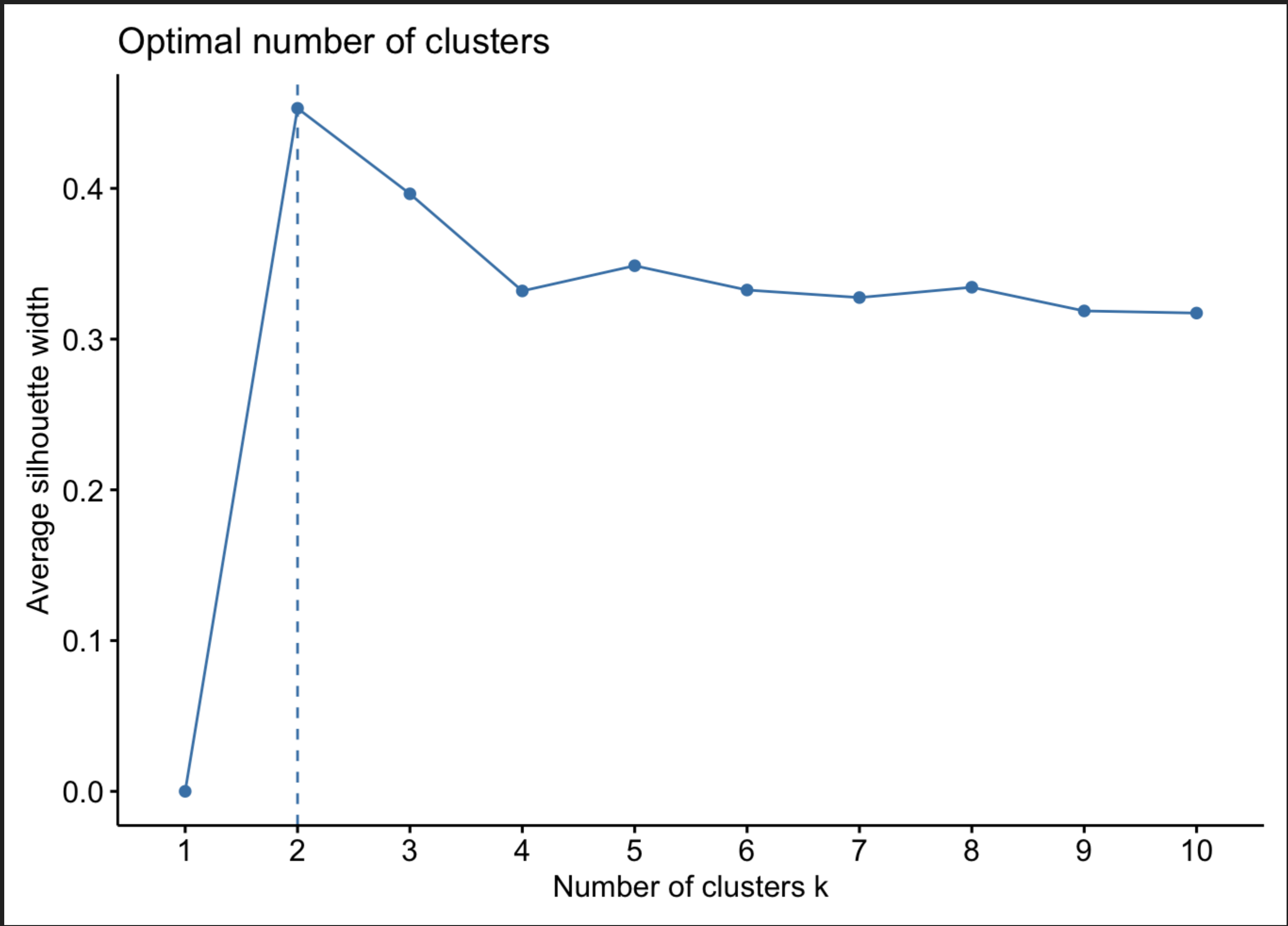


OTHER METHODS TO ESTABLISH K

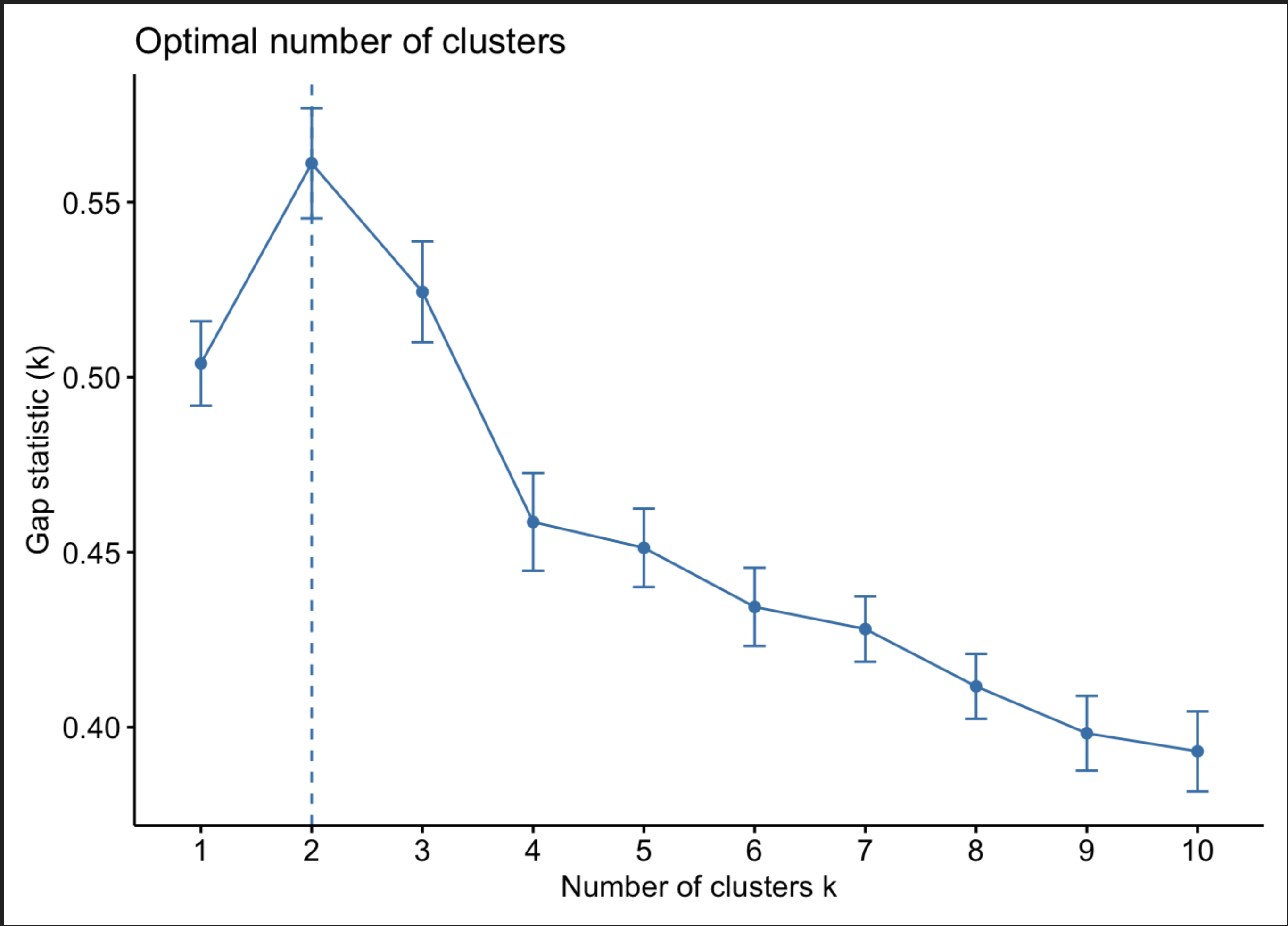
- Silhoutte method (cluster fit)
- Gap statistic

See also this tutorial.

SILHOUETTE METHOD



GAP STATISTIC

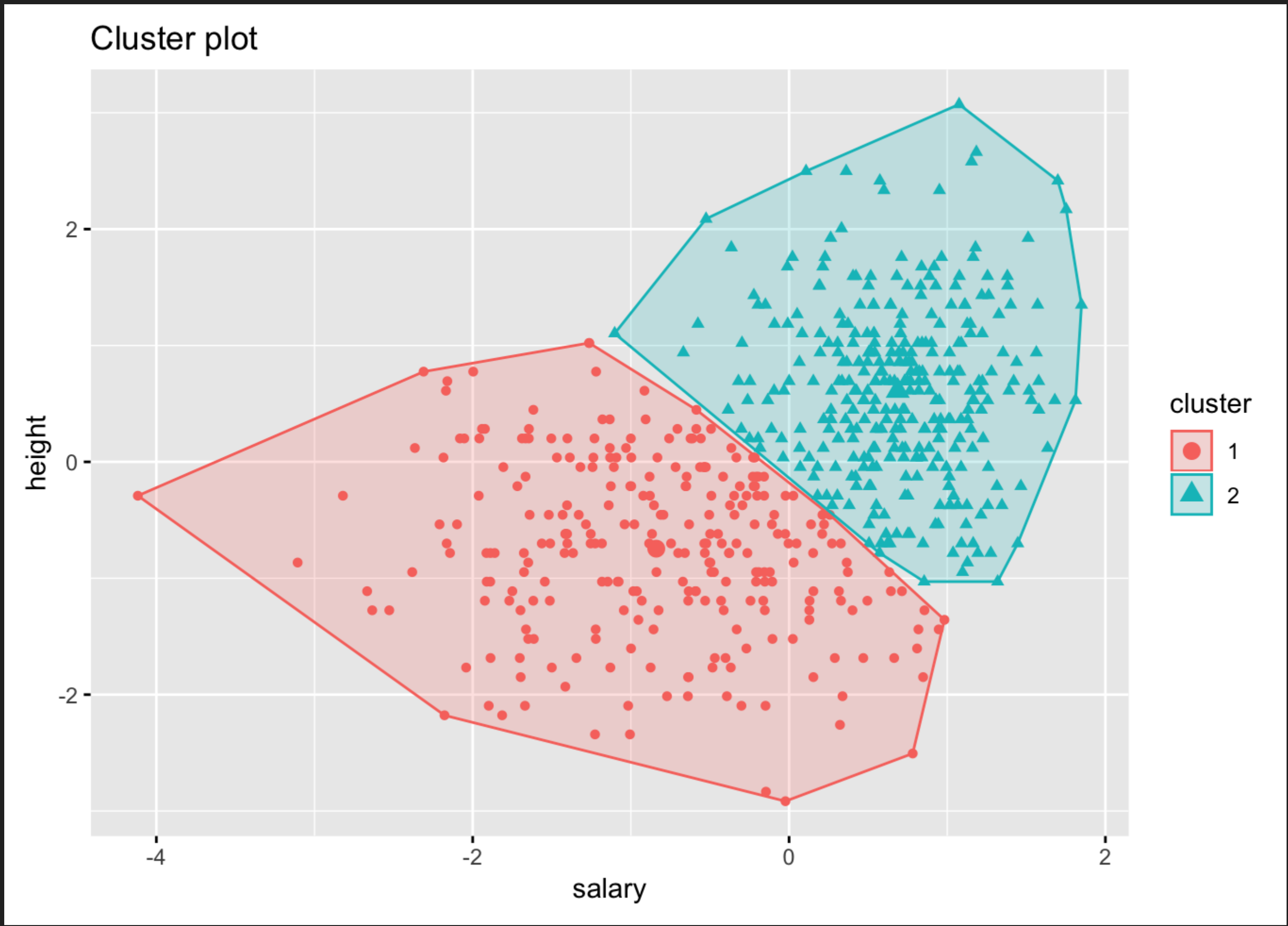


CHOOSING *K*

We settle for $k = 2$

```
unsup_model_final = kmeans(data4
                        , centers = 2
                        , nstart = 10
                        , iter.max = 10)
```

PLOT THE CLUSTER ASSIGNMENT



OTHER UNSUPERVISED METHODS

- k-means (today)
- hierarchical clustering
- density clustering

ISSUES WITH UNSUPERVISED LEARNING

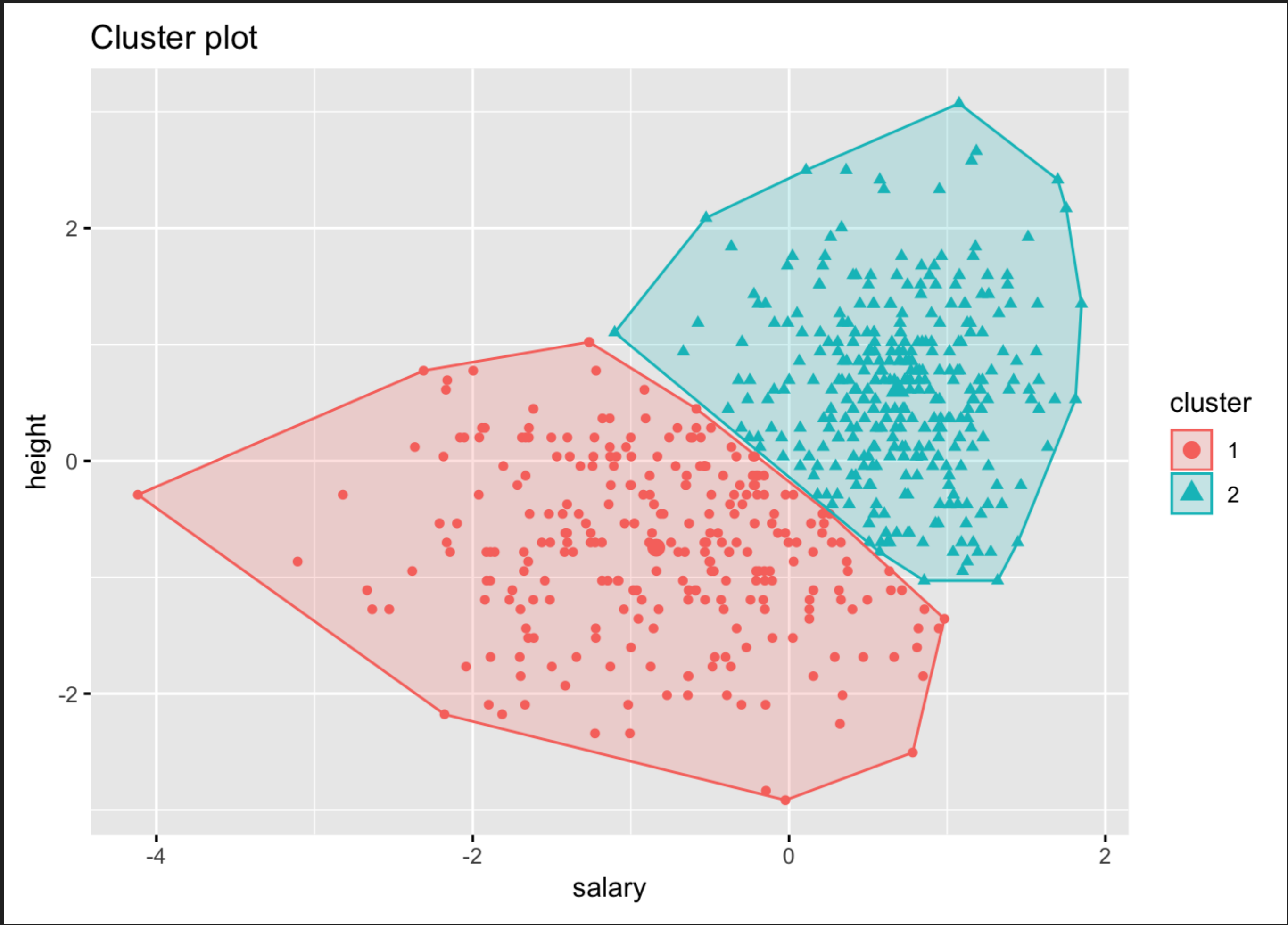
What’s lacking?

What can you (not) say?

CAVEATS OF UNSUP. ML

- there is no “ground truth”
- interpretation/subjectivity
- cluster choice

INTERPRETATION OF FINDINGS



INTERPRETATION OF FINDINGS

```
unsup_model_final$centers
```

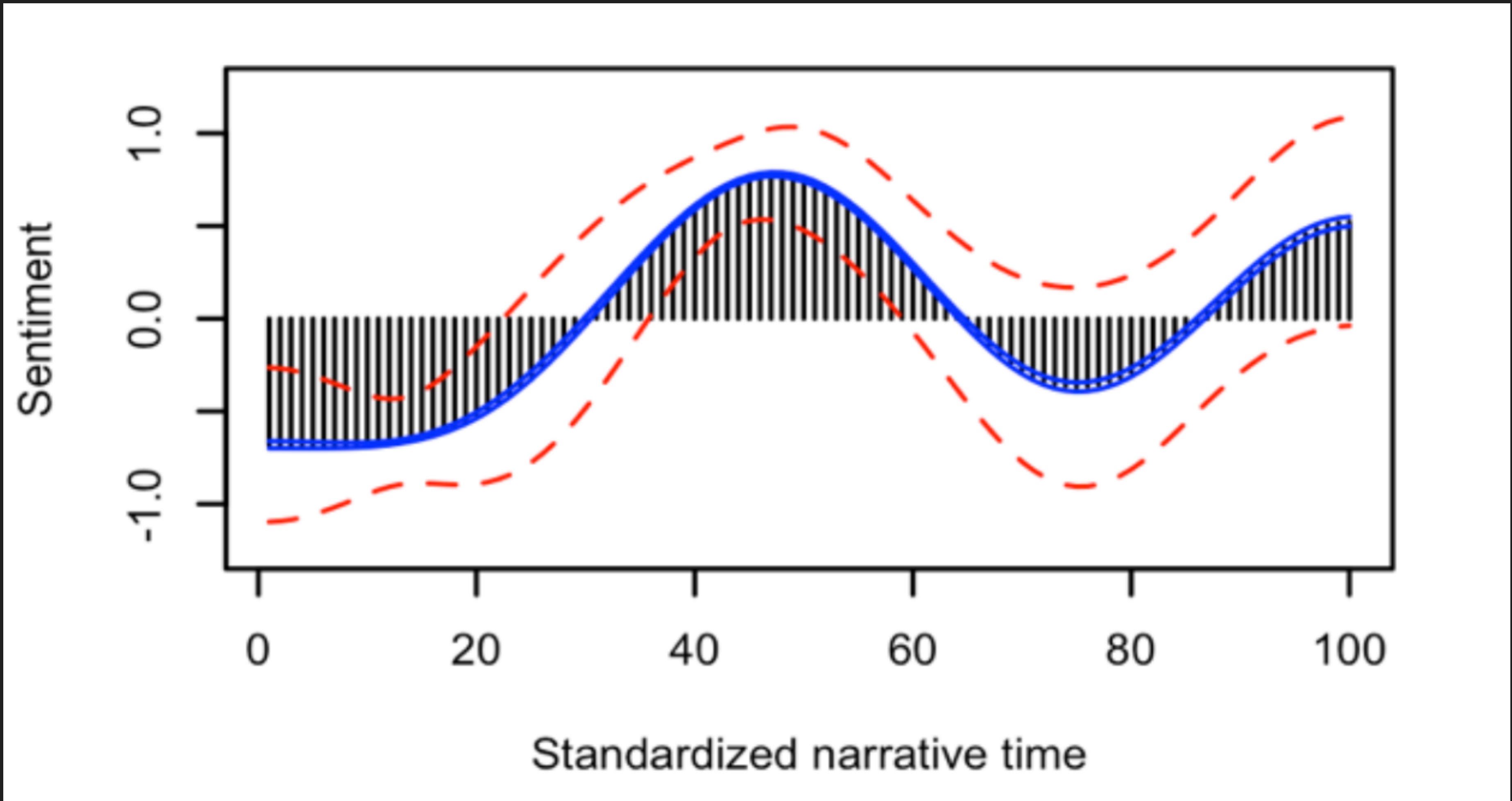
```
##       salary     height
## 1 -0.8395549 -0.7457021
## 2  0.6869085  0.6101199
```

- Cluster 1: low salary, small
- Cluster 2: high salary, tall

Note: we cannot say anything about accuracy.

See the k-NN model.

INTERPRETATION OF FINDINGS



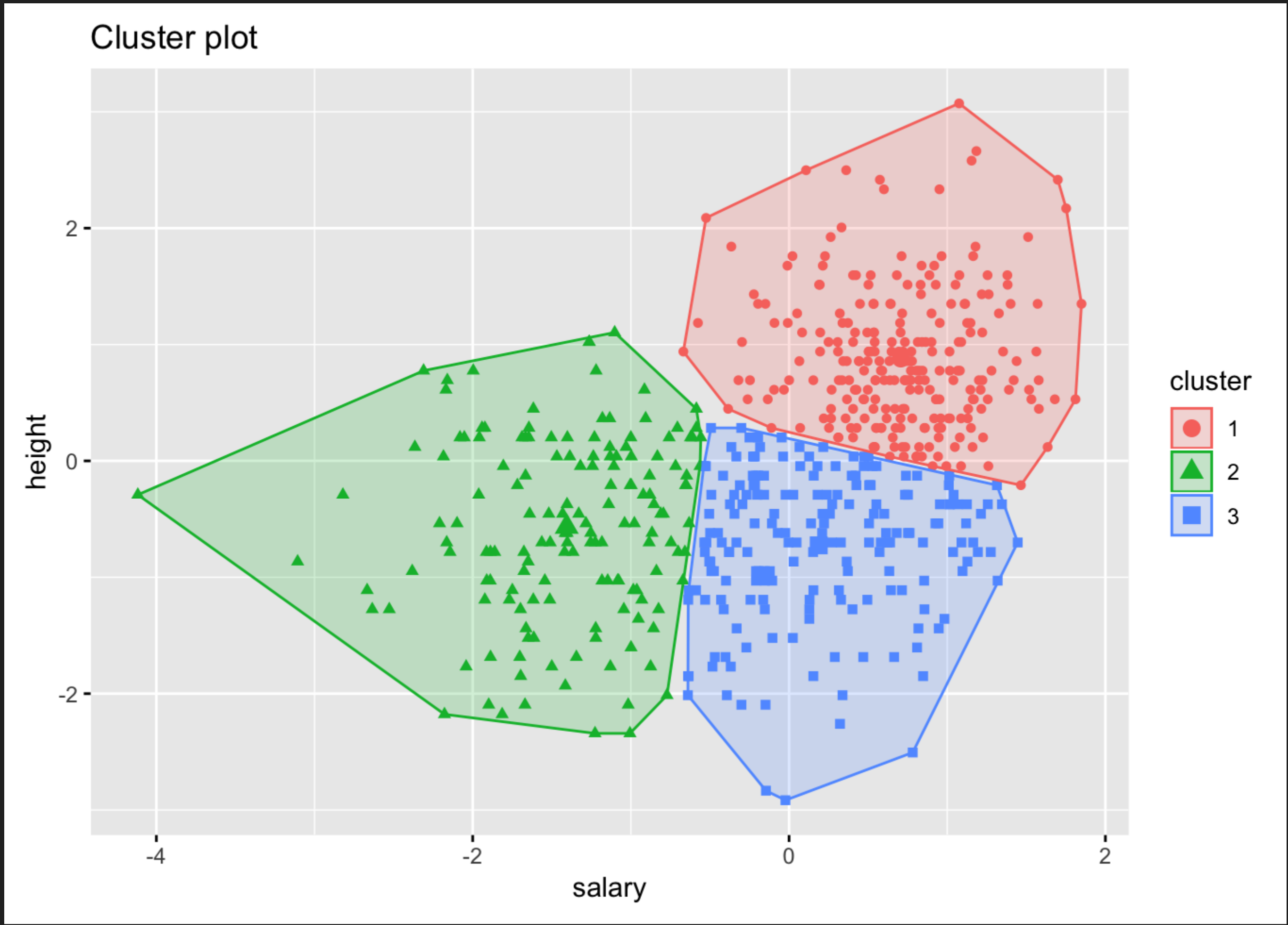
INTERPRETATION OF FINDINGS

- subjective
- labelling tricky
- researchers choice!
- be open about this

CLUSTER CHOICE

What if we chose $k = 3$?

```
km_3 = kmeans(data4, centers = 3, nstart = 10, iter.max = 10)
fviz_cluster(km_3, geom = "point", data = data4)
```



Cluster plot
height
salary
2
0
-2
-4
cluster
1
2
3

CLUSTER CHOICE

What if we chose $k = 3$?

```
km_3$centers
```

```
##       salary     height
## 1  0.7063757  0.8795474
## 2 -1.4058046 -0.5668204
## 3  0.1876933 -0.7256515
```

- Cluster 1: high salary, very tall
- Cluster 2: very low salary, small
- Cluster 3: avg salary, small

CLUSTER CHOICE

- be open about it
- make all choices transparent
- always share code and data (“least vulnerable”" principle)

PERFORMANCE METRICS FOR CLASSIFICATION TASKS

FAKE NEWS PROBLEM

STEP 1: SPLITTING THE DATA

```
set.seed(2019)
in_training = createDataPartition(y = fake_news_data$outcome
                                  , p = .7
                                  , list = FALSE
                                  )
training_data = fake_news_data[ in_training,]
test_data = fake_news_data[-in_training,]
```

STEP 2: DEFINE TRAINING CONTROLS

```
training_controls = trainControl(method="cv"
                                , number = 5
                                , classProbs = T
                                )
```

STEP 3: TRAIN THE MODEL

```
fakenews_model = train(outcome ~ .
                       , data = training_data
                       , trControl = training_controls
                       , method = "svmLinear"
                       )
```

STEP 4: FIT THE MODEL

```
model.predictions = predict(fakenews_model, test_data)
```

YOUR TASK:

Evaluate the model.

What do you do?

MODEL EVALUATION

	fake	**real**
fake	252	48
real	80	220

(252+220)/600 = 0.79

INTERMEZZO

THE CONFUSION MATRIX

CONFUSION MATRIX

	Fake	**Real**
Fake	True positives	False negatives
Real	False positives	True negatives

CONFUSION MATRIX

- true positives (TP): correctly identified fake ones
- true negatives (TN): correctly identified real ones
- false positives (FP): false accusations
- false negatives (FN): missed fakes

OKAY: LET'S USE ACCURACIES

$$acc = \frac{(TP + TN)}{N}$$

Any problems with that?

ACCURACY

Model 1

	Fake	**Real**
Fake	252	48
Real	80	220

Model 2

	Fake	**Real**
Fake	290	10
Real	118	182

PROBLEM WITH ACCURACY

- same accuracy, different confusion matrix
- relies on thresholding idea
- not suitable for comparing models (don’t be fooled by the literature!!)

Needed: more nuanced metrics

BEYOND ACCURACY

```
##         prediction
## reality Fake Real Sum
##    Fake  252   48 300
##    Real   80  220 300
##    Sum   332  268 600
```

```
##         prediction
## reality Fake Real Sum
##    Fake  290   10 300
##    Real  118  182 300
##    Sum   408  192 600
```

PRECISION

i.e. –> how often the prediction is correct when prediction class *X*

Note: we have two classes, so we get *two* precision values

Formally:

- $Pr_{fake} = \frac{TP}{(TP+FP)}$
- $Pr_{real} = \frac{TN}{(TN+FN)}$

PRECISION

```
##          prediction
## reality Fake Real Sum
##     Fake  252   48 300
##     Real   80  220 300
##     Sum   332  268 600
```

- $Pr_{fake} = \frac{252}{332} = 0.76$
- $Pr_{real} = \frac{220}{268} = 0.82$

COMPARING THE MODELS

	Model 1	**Model 2**
acc	0.79	0.79
Pr_{fake}	0.76	0.71
Pr_{real}	0.82	0.95

RECALL

i.e. –> how many of class *X* is detected

Note: we have two classes, so we get *two* recall values

Also called sensitivity and specificity!

Formally:

- $R_{fake} = \frac{TP}{(TP+FN)}$
- $R_{real} = \frac{TN}{(TN+FP)}$

RECALL

```
##          prediction
## reality Fake Real Sum
##    Fake  252   48 300
##    Real   80  220 300
##    Sum   332  268 600
```

- $R_{fake} = \frac{252}{300} = 0.84$
- $R_{real} = \frac{220}{300} = 0.73$

COMPARING THE MODELS

	Model 1	Model 2
acc	0.79	0.79
Pr_{fake}	0.76	0.71
Pr_{real}	0.82	0.95
R_{fake}	0.84	0.97
R_{real}	0.73	0.61

COMBINING PR AND R

The *F1* measure.

Note: we combine Pr and R for each class, so we get *two* F1 measures.

Formally:

- $F1_{fake} = 2 * \frac{Pr_{fake} * R_{fake}}{Pr_{fake} + R_{fake}}$
- $F1_{real} = 2 * \frac{Pr_{real} * R_{real}}{Pr_{real} + R_{real}}$

F1 MEASURE

```
##          prediction
## reality Fake Real Sum
##    Fake  252   48 300
##    Real   80  220 300
##    Sum   332  268 600
```

- $F1_{fake} = 2 * \frac{0.76 * 0.84}{0.76 + 0.84} = 2 * \frac{0.64}{1.60} = 0.80$
- $F1_{real} = 2 * \frac{0.82 * 0.73}{0.82 + 0.73} = 0.78$

COMPARING THE MODELS

	Model 1	Model 2
acc	0.79	0.79
Pr_{fake}	0.76	0.71
Pr_{real}	0.82	0.95
R_{fake}	0.84	0.97
R_{real}	0.73	0.61
$F1_{fake}$	0.80	0.82
$F1_{real}$	0.78	0.74

IN CARET

```
confusionMatrix(model.predictions, as.factor(test_data$outcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction fake real
##       fake  252   80
##       real   48  220
##
##                Accuracy : 0.7867
##                  95% CI : (0.7517, 0.8188)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5733
##  Mcnemar's Test P-Value : 0.006143
##
##             Sensitivity : 0.8400
```

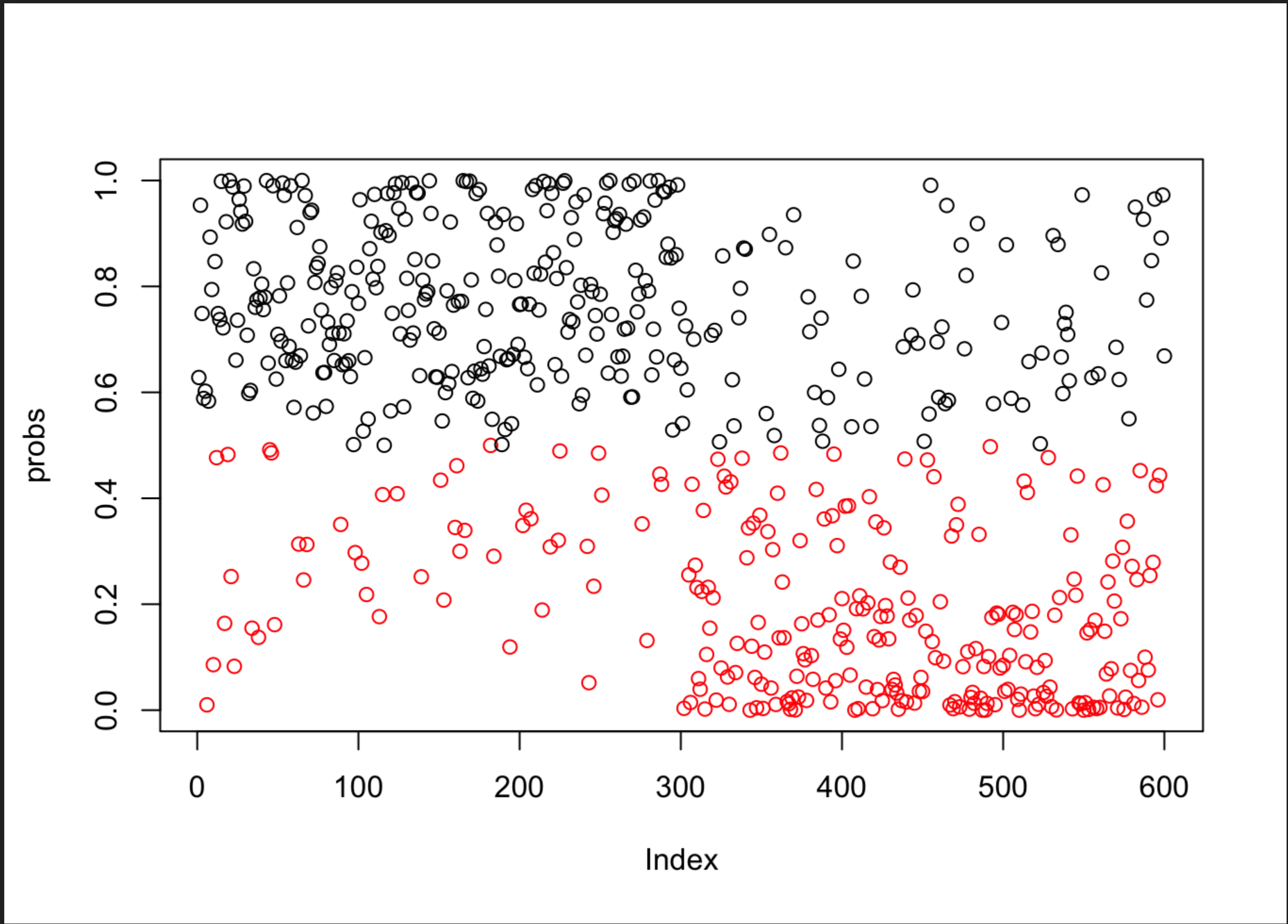
THERE’S MORE

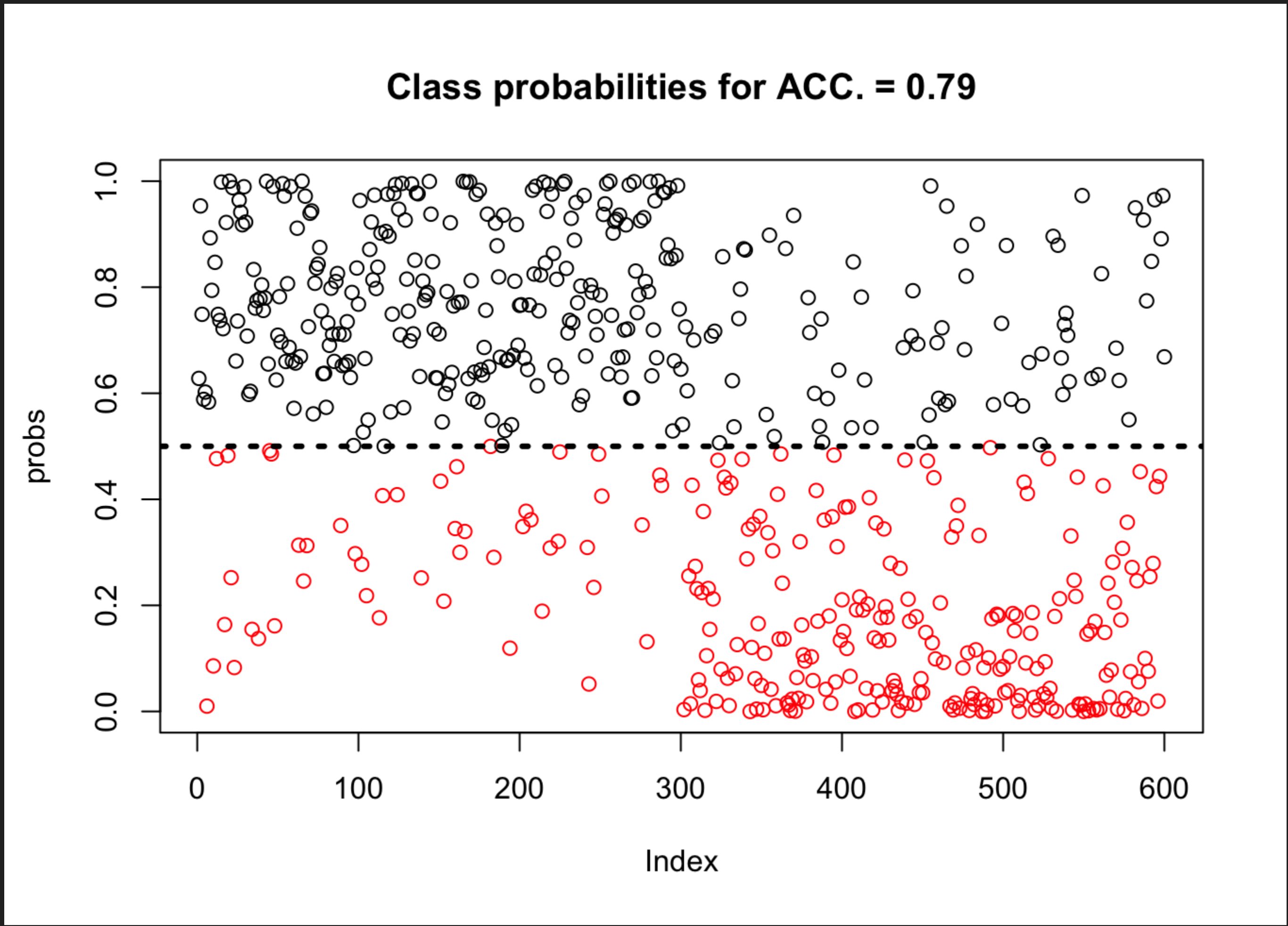
What’s actually behind the model’s predictions?

Any ideas?

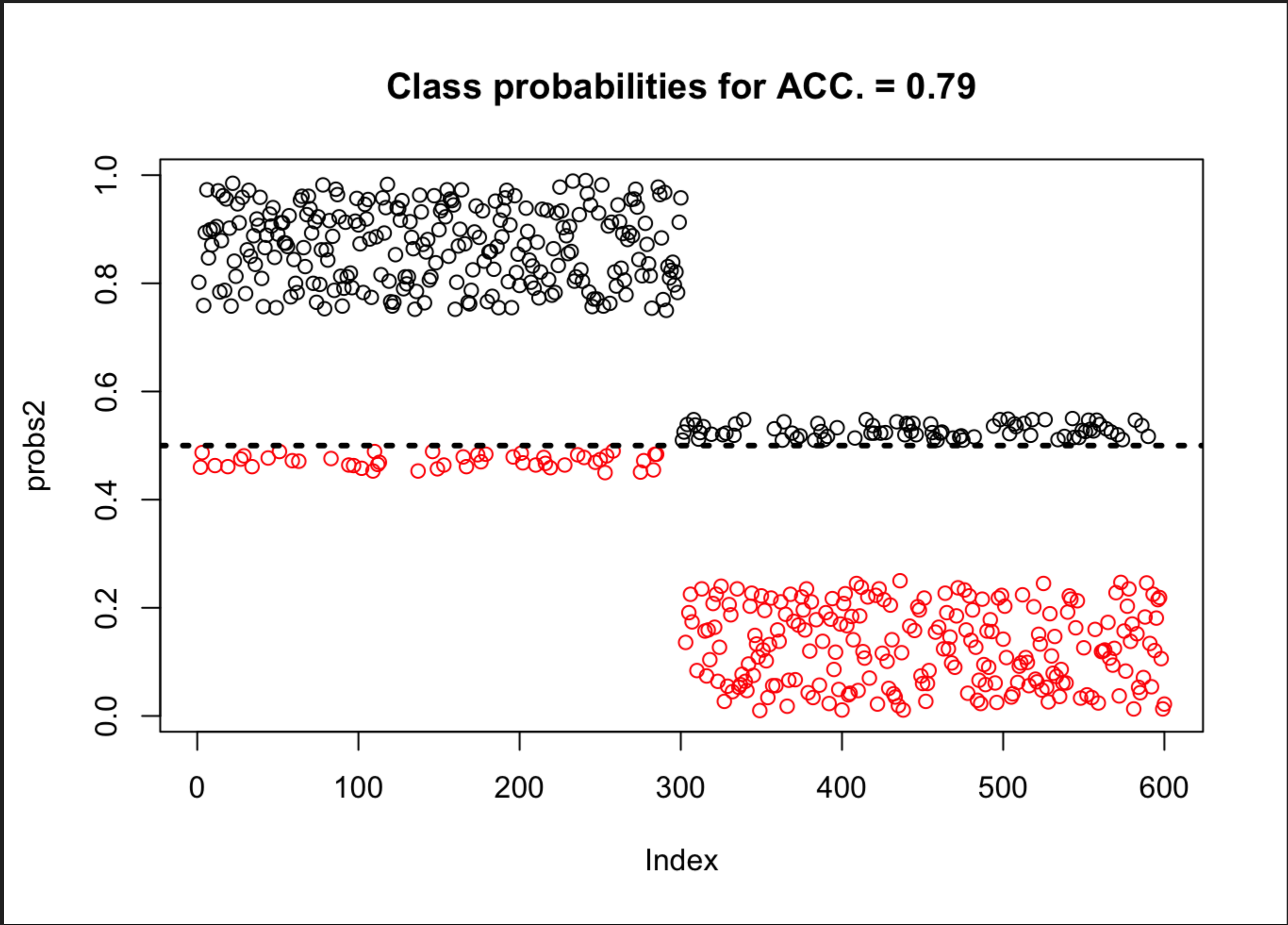
CLASS PROBABILITIES

Notice anything?




Class probabilities for ACC. = 0.79
probs
Index
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600

THE THRESHOLD PROBLEM



ISSUE!

- classification threshold little informative
- obscures certainty in judgment

Needed: a representation across all possible values

THE AREA UNDER THE CURVE (AUC)

Idea:

- plot all observed values (here: class probs)
- y-axis: sensitivity
- x-axis: 1-specificity

AUC STEP-WISE

```
threshold_1 = probs[1]
threshold_1
```

```
## [1] 0.6280156
```

```
pred_threshold_1 = ifelse(probs >= threshold_1, 'fake', 'real')
knitr::kable(table(test_data$outcome, pred_threshold_1))
```

	fake	real
fake	221	79
real	52	248

SENSITIVITY AND 1-SPECIFICITY

	fake	real
fake	221	79
real	52	248

$$Sens. = 221/300 = 0.74$$

$$Spec. = 248/300 = 0.83$$

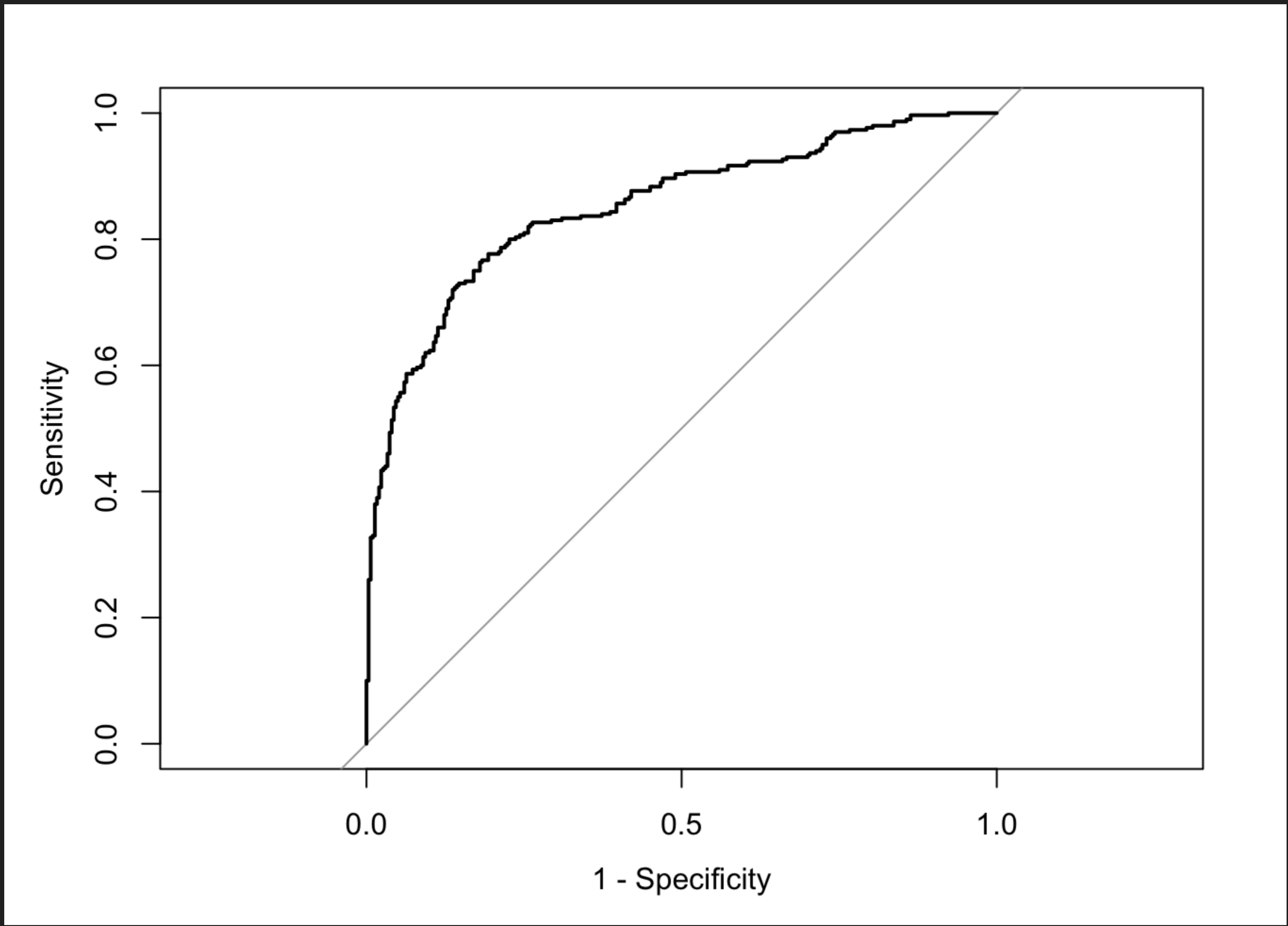
$$Sens. = 221/300 = 0.74$$

$$Spec. = 248/300 = 0.83$$

Threshold	Sens.	1-Spec
0.63	0.74	0.17

Do this for every threshold observed.

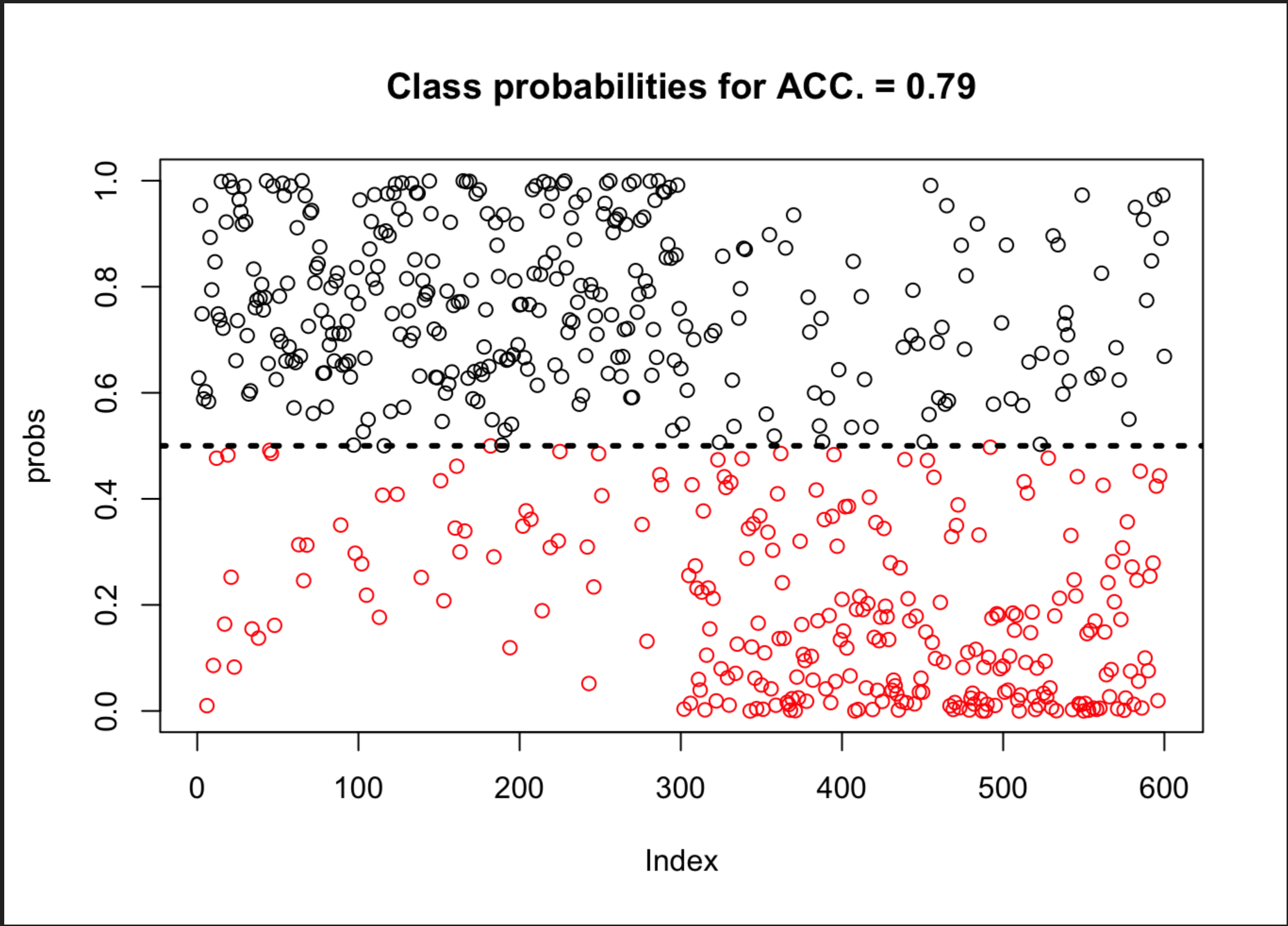
.. AND PLOT THE RESULTS:



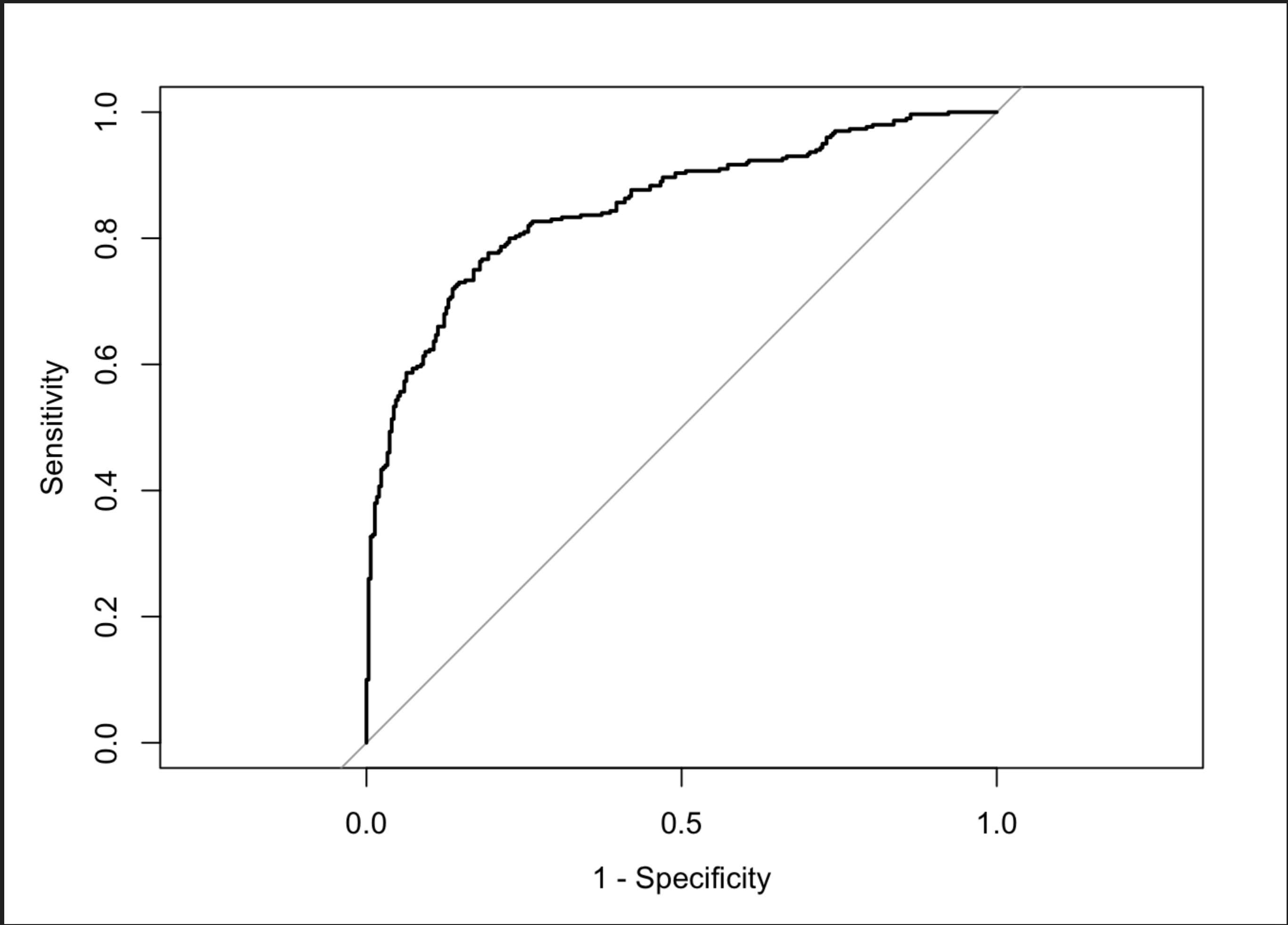
QUANTIFY THIS PLOT

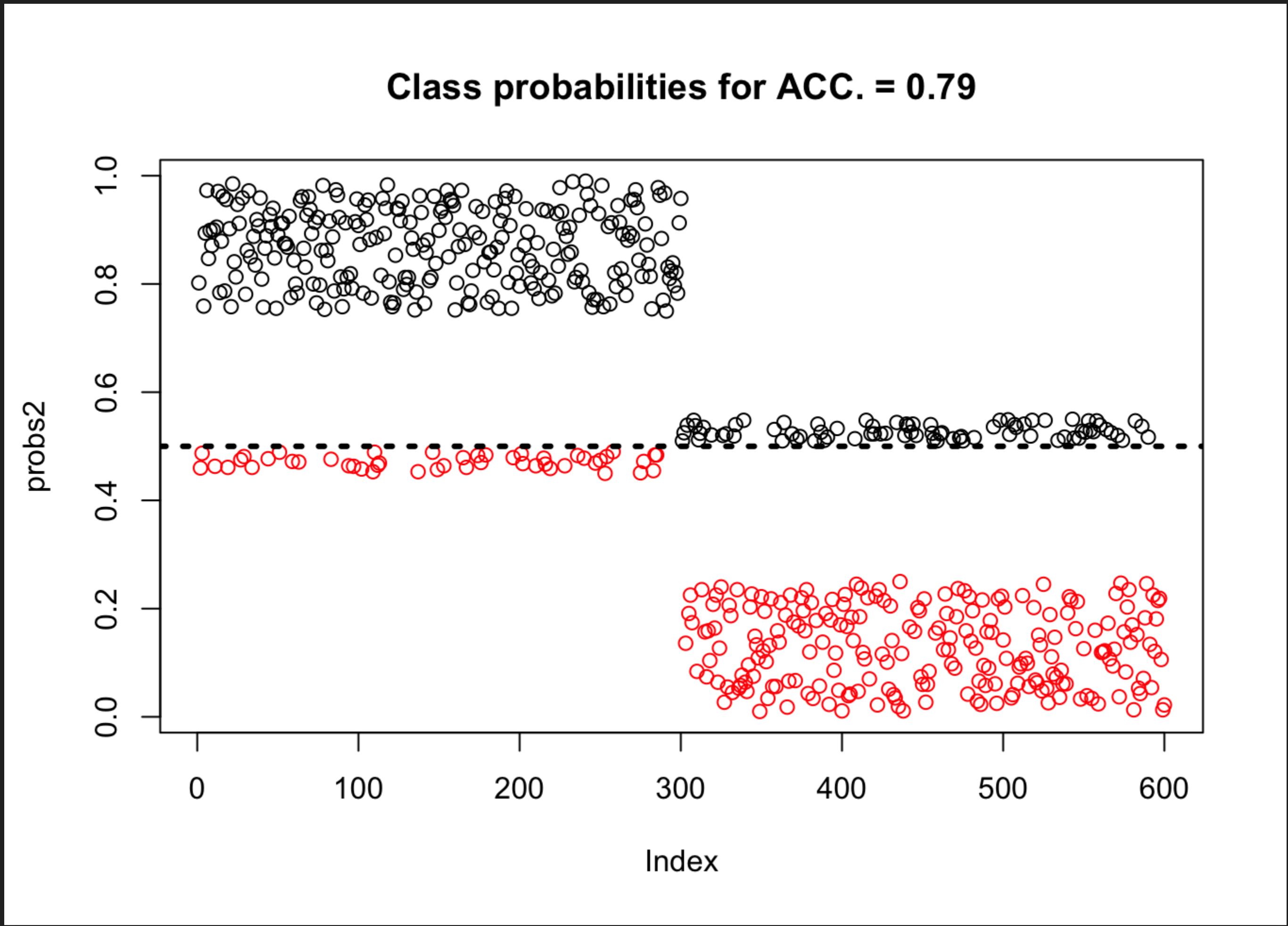
```
auc1 = roc(response = test_data$outcome
               , predictor = probs
               , ci=T)
```

WHAT IF WE COMPARE OUR TWO MODELS?



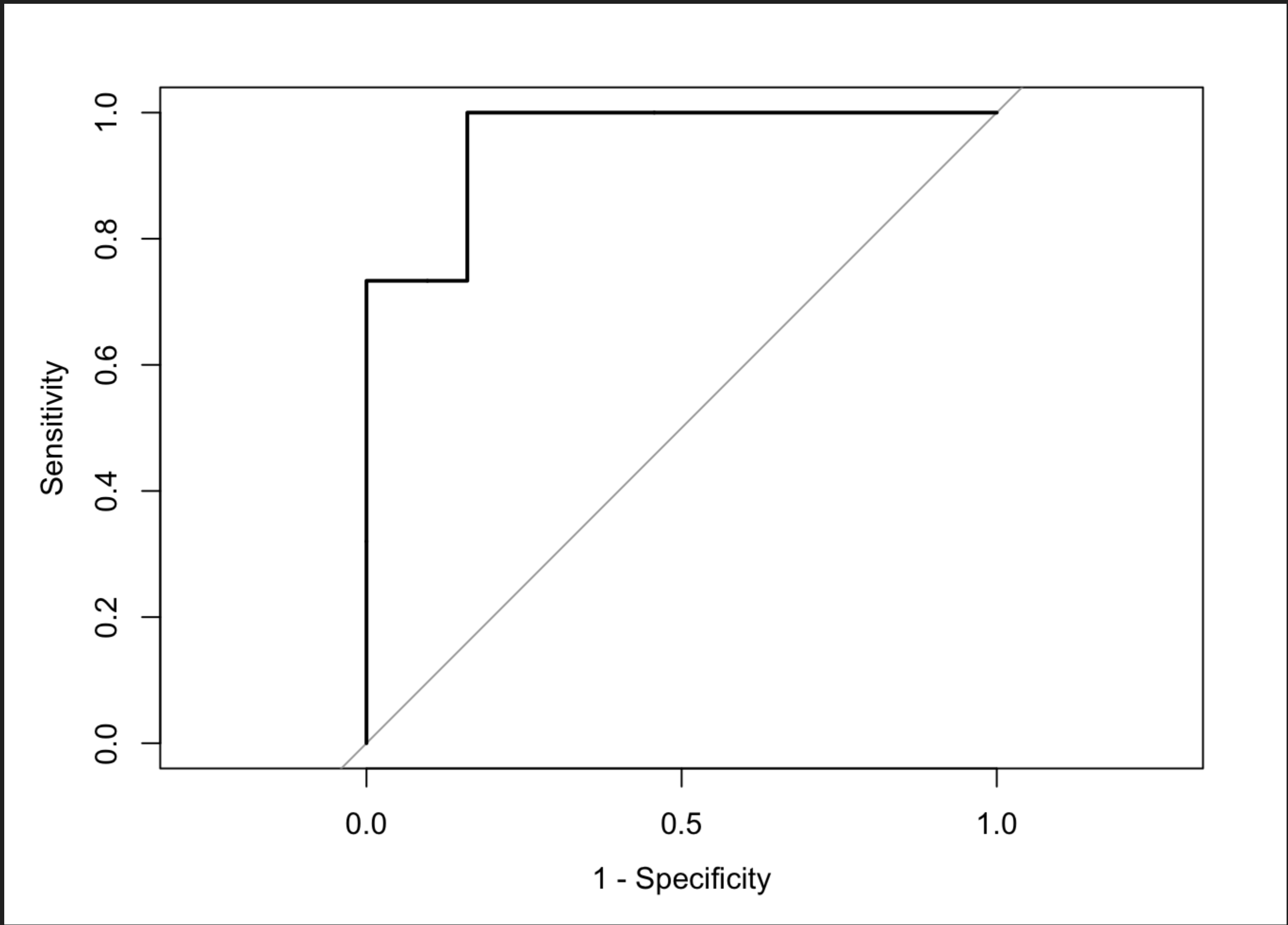
```
plot.roc(auc1, xlim=c(1, 0), legacy.axes = T)
```




Class probabilities for ACC. = 0.79
probs2
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600
Index

```r
auc2 = roc(response = test_data$outcome
                , predictor = probs2
                , ci=T)

plot.roc(auc2, xlim=c(1, 0), legacy.axes = T)
```



AUCS NUMERICALLY

```
#model 1
roc(response = test_data$outcome , predictor = probs, ci=T)
```

```
##
## Call:
## roc.default(response = test_data$outcome, predictor = probs,      c
##
## Data: probs in 300 controls (test_data$outcome fake) > 300 cases (
## Area under the curve: 0.8521
## 95% CI: 0.8216-0.8827 (DeLong)
```

```
#model 2
roc(response = test_data$outcome , predictor = probs2, ci=T)
```

```
##
## Call:
## roc.default(response = test_data$outcome, predictor = probs2,
##
## Data: probs2 in 300 controls (test_data$outcome fake) > 300 cases
## Area under the curve: 0.9573
## 95% CI: 0.9437-0.971 (DeLong)
```

RECAP

- Unsupervised ML
- Performance metrics
  - confusion matrix
  - AUC
- Validation and generalisation

OUTLOOK

Tutorial tomorrow

Week 8: Applied predictive modelling + R Notebooks

END