

WEEK 7: MACHINE LEARNING 1

SECU0050

BENNETT KLEINBERG

27 FEB 2020



Data Science for Crime Scientists

WEEK 7: MACHINE LEARNING 1

MACHINE LEARNING?

- core idea: a system learns from experience
- no precise instructions

TODAY

- supervised learning
- core algorithms
- performance metrics

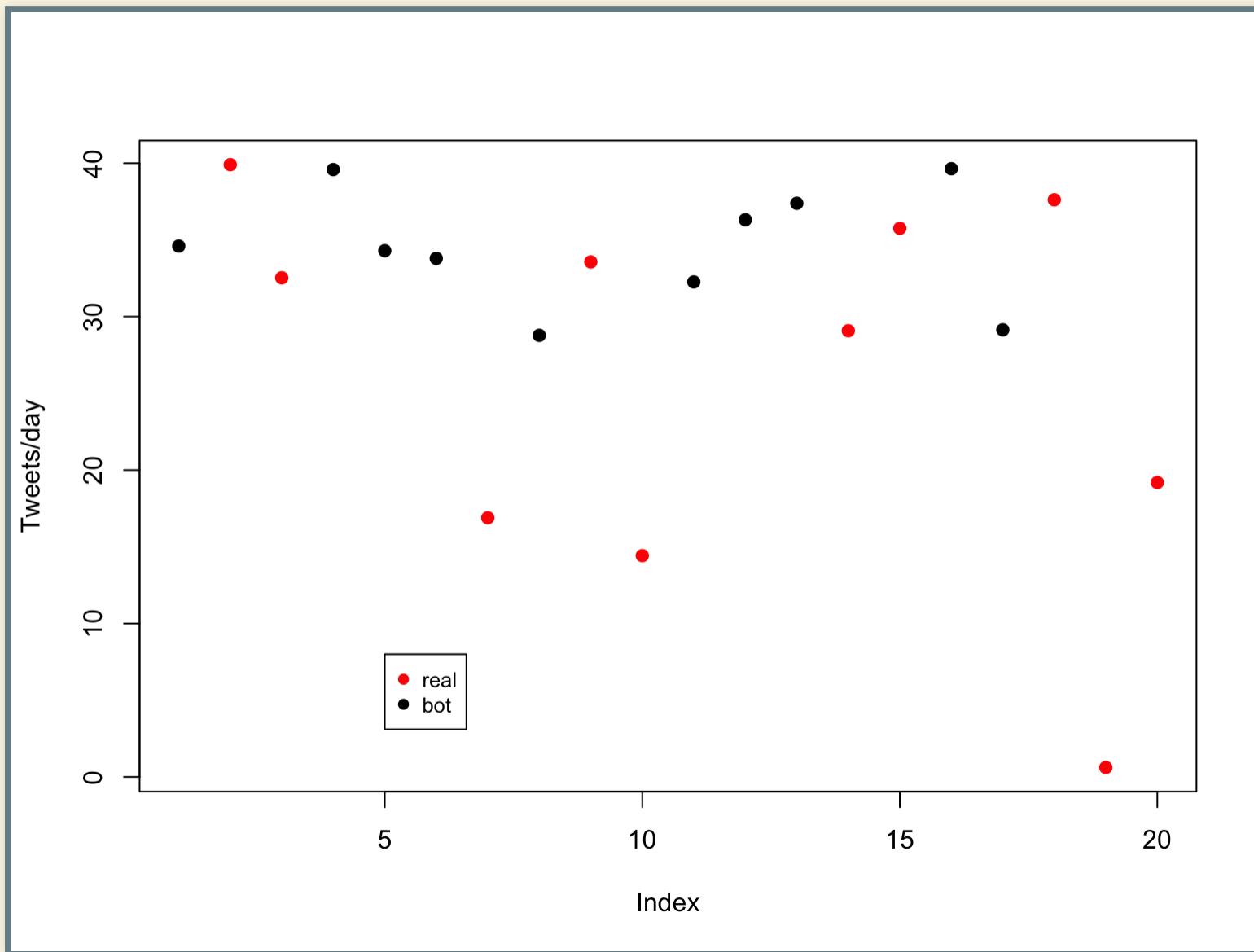
SUPERVISED LEARNING

- who is the supervisor?
- supervised = labelled data
- i.e. you know the outcome
- flipped logic

SIMULATED DATA

account	tweet_freq
bot	33.7960
bot	34.5973
bot	34.2955
bot	32.2615
real	19.1904
real	14.4237
real	37.6143
real	29.0793

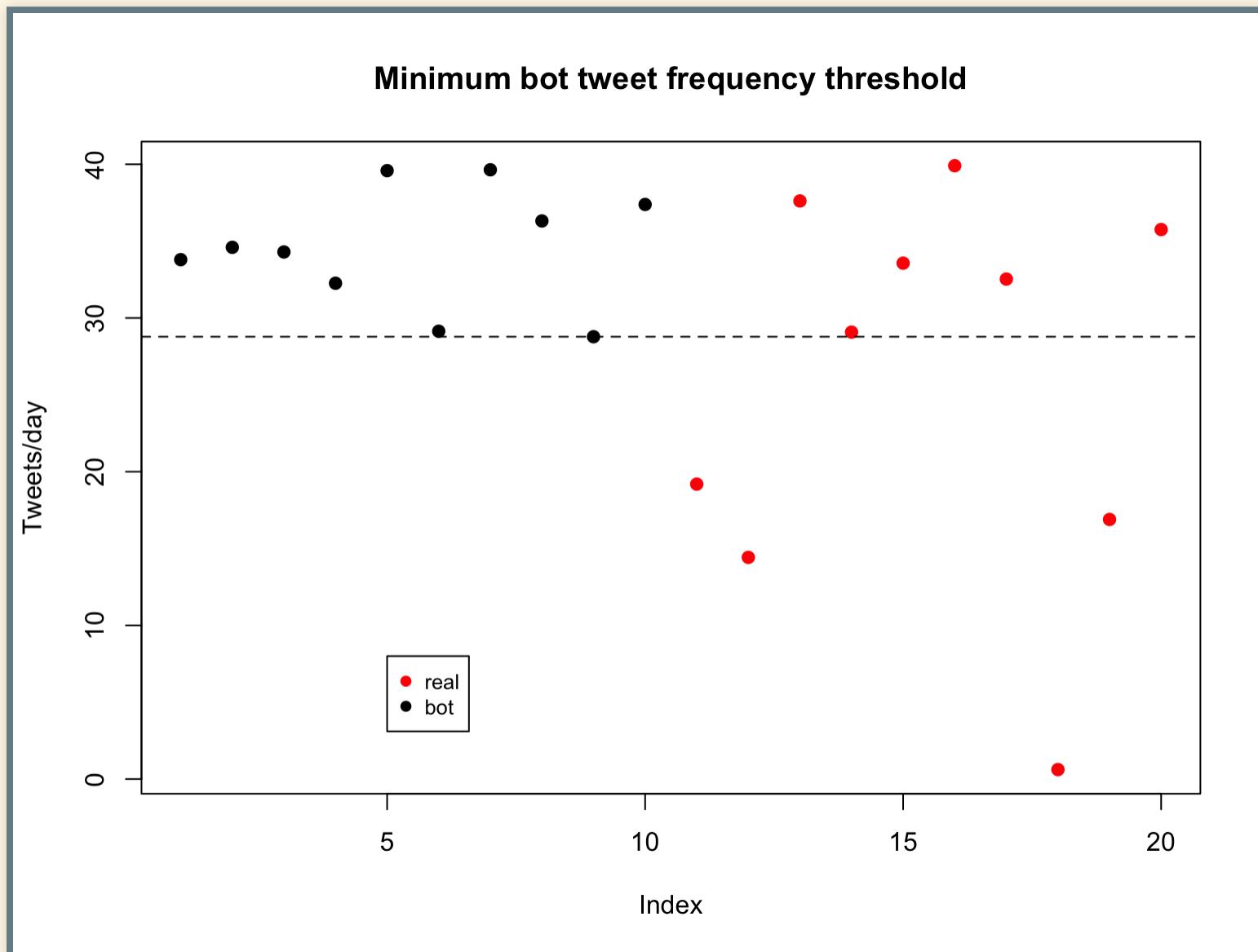
HOW TO BEST SEPARATE THE DATA INTO TWO GROUPS?



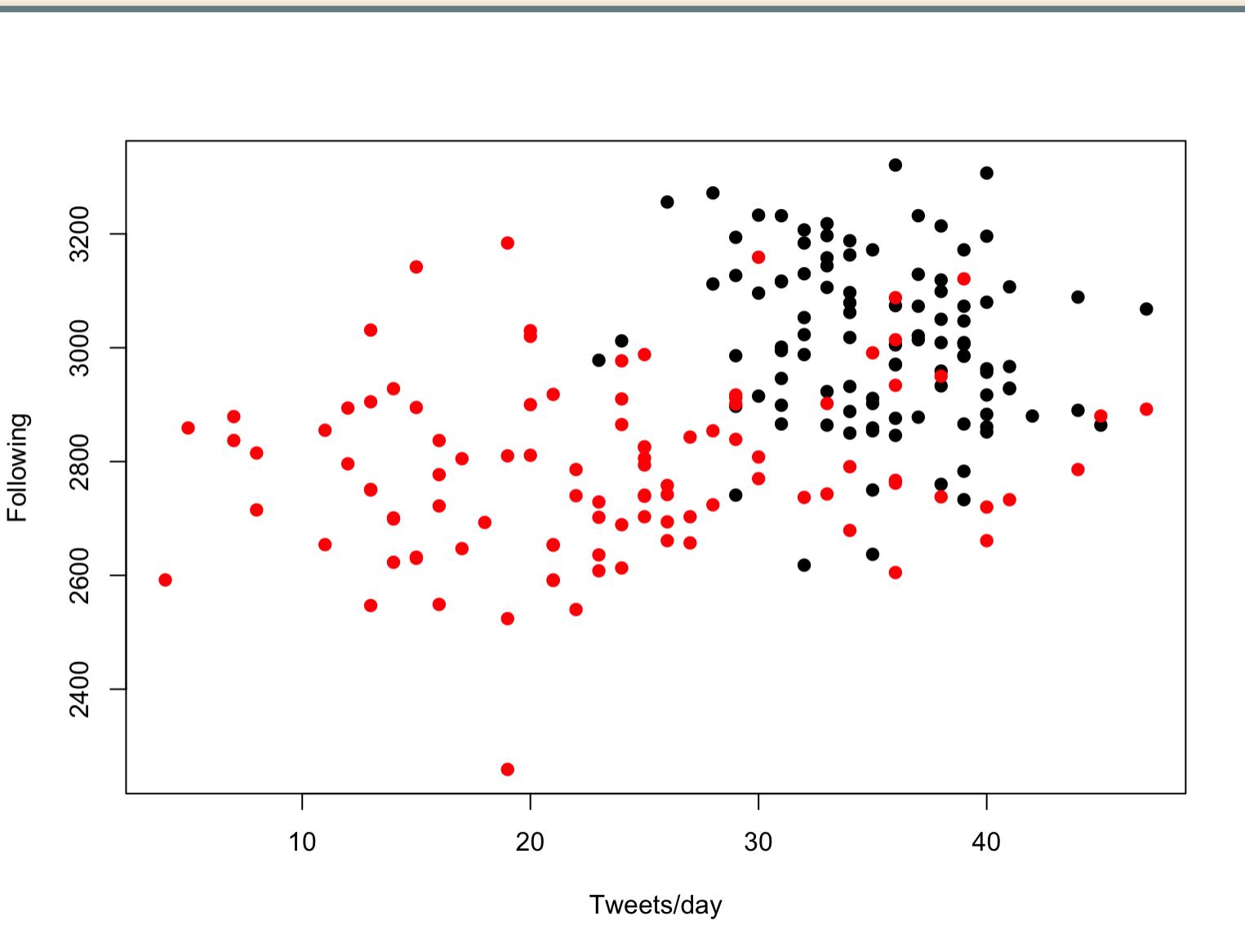
CORE IDEA

- learn relationship between
 - outcome (target) variable
 - features (predictors)
- “learning” is done through an algorithm
 - simplest algorithm: **if A then B**

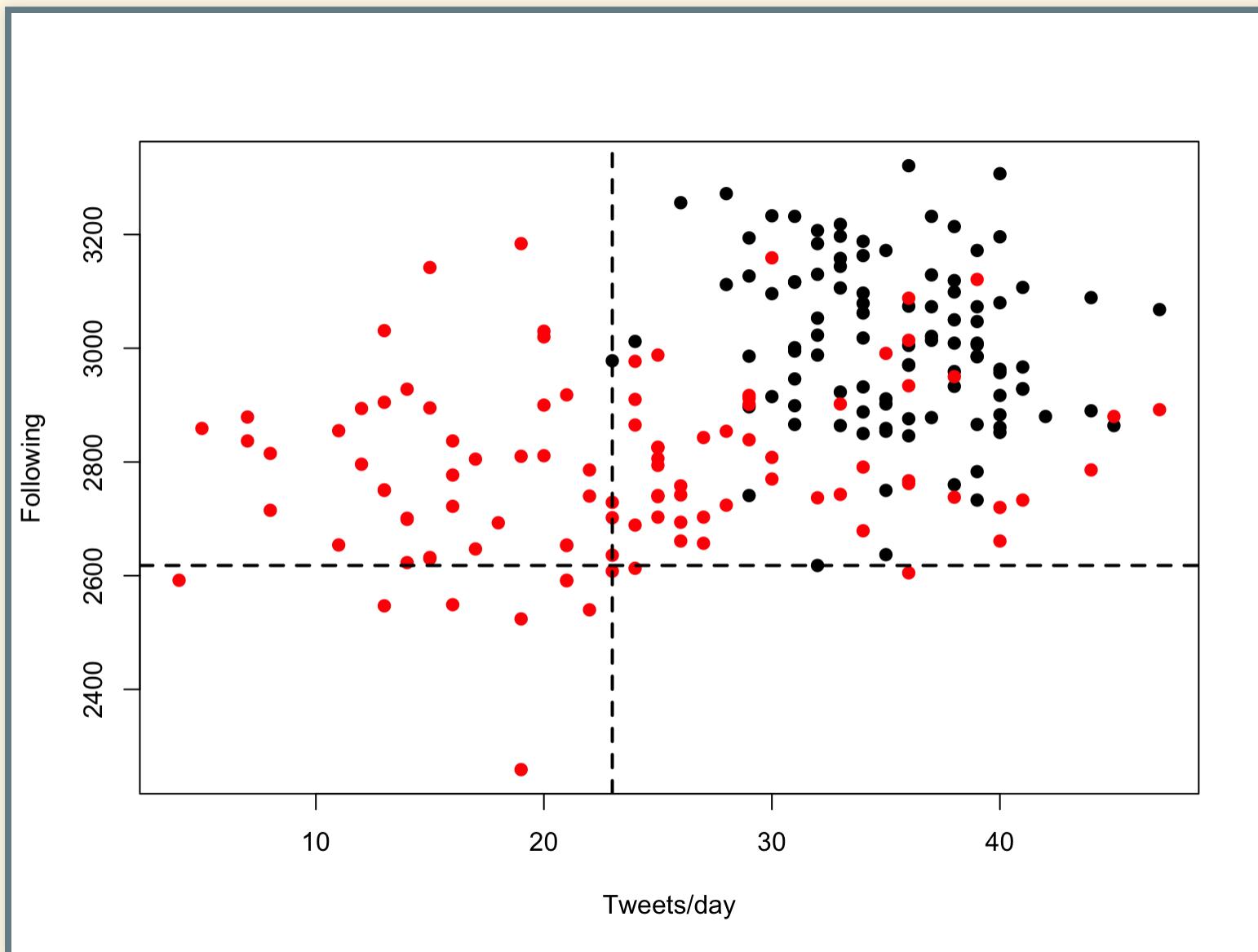
IDEA 1: MEAN THRESHOLDS



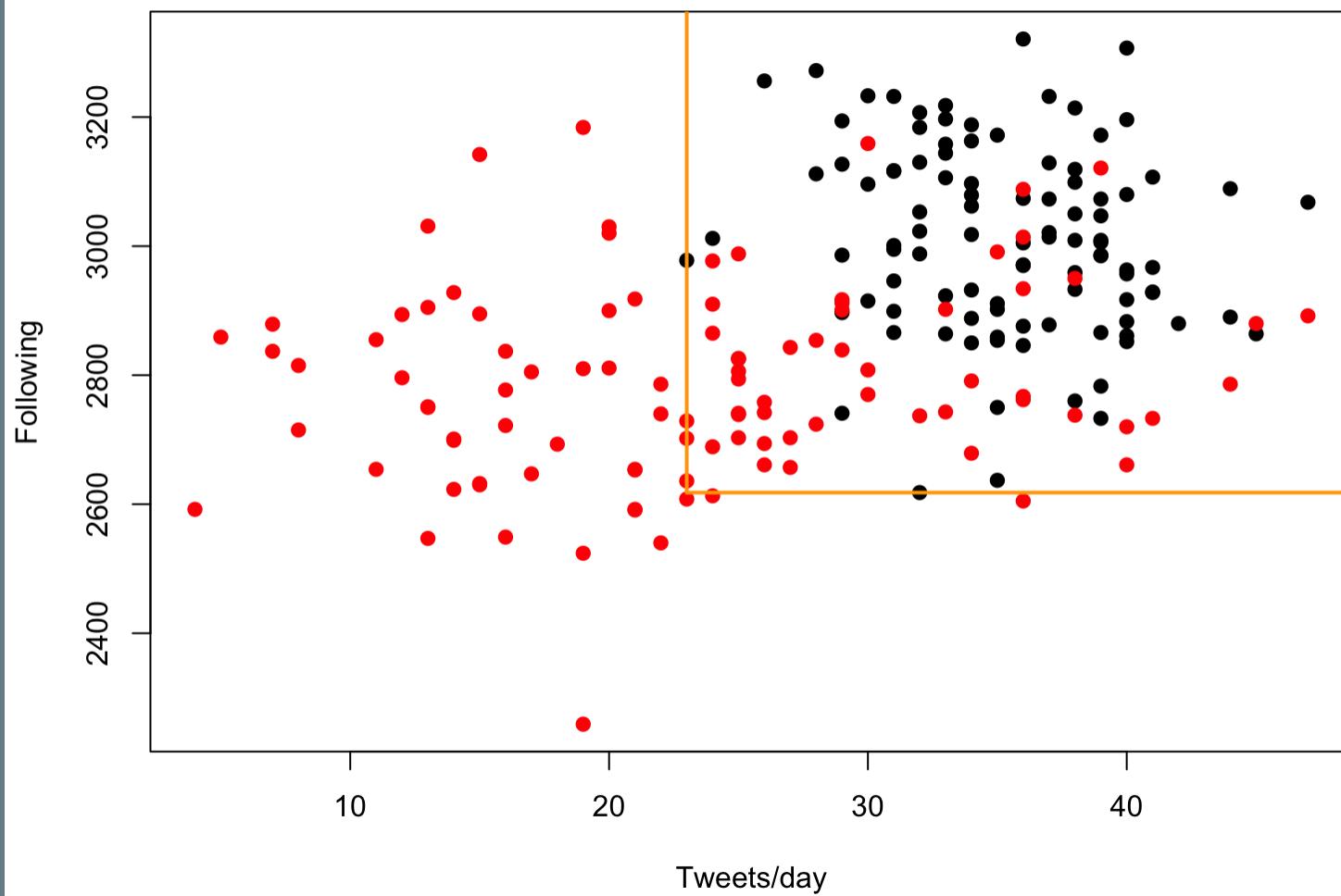
HIGHER DIMENSIONAL DATA



IDEA 2: HAND-CRAFTED RULES



Hand-crafted rule-based data separation



BUT THIS IS NOT LEARNING

- often we have no idea about the relationships
- too many predictors
- too diverse a problem
- simply unknown

STEPWISE SUPERVISED ML

- clarify what **outcome** and **features** are
- determine which classification algorithm to use
- train the model

CLASSES OF SUPERVISED LEARNING

- classification (e.g. death/alive, fake/real)
- regression (e.g. income, number of deaths)

CORE AIM OF SUPERVISED LEARNING

Learning from data to make predictions about the future.

CARET IN PRACTICE

```
my_first_model = glm(account ~ .  
                      , data = data2  
                      , family = 'binomial'  
                      )
```

We have trained a model.

= you have taught an algorithm to learn to predict real vs bot accounts based on followers and tweet frequency

**WHAT IF I TOLD YOU IT IS
REGRESSION**

EXPLANATORY VS PREDICTIVE MODELLING

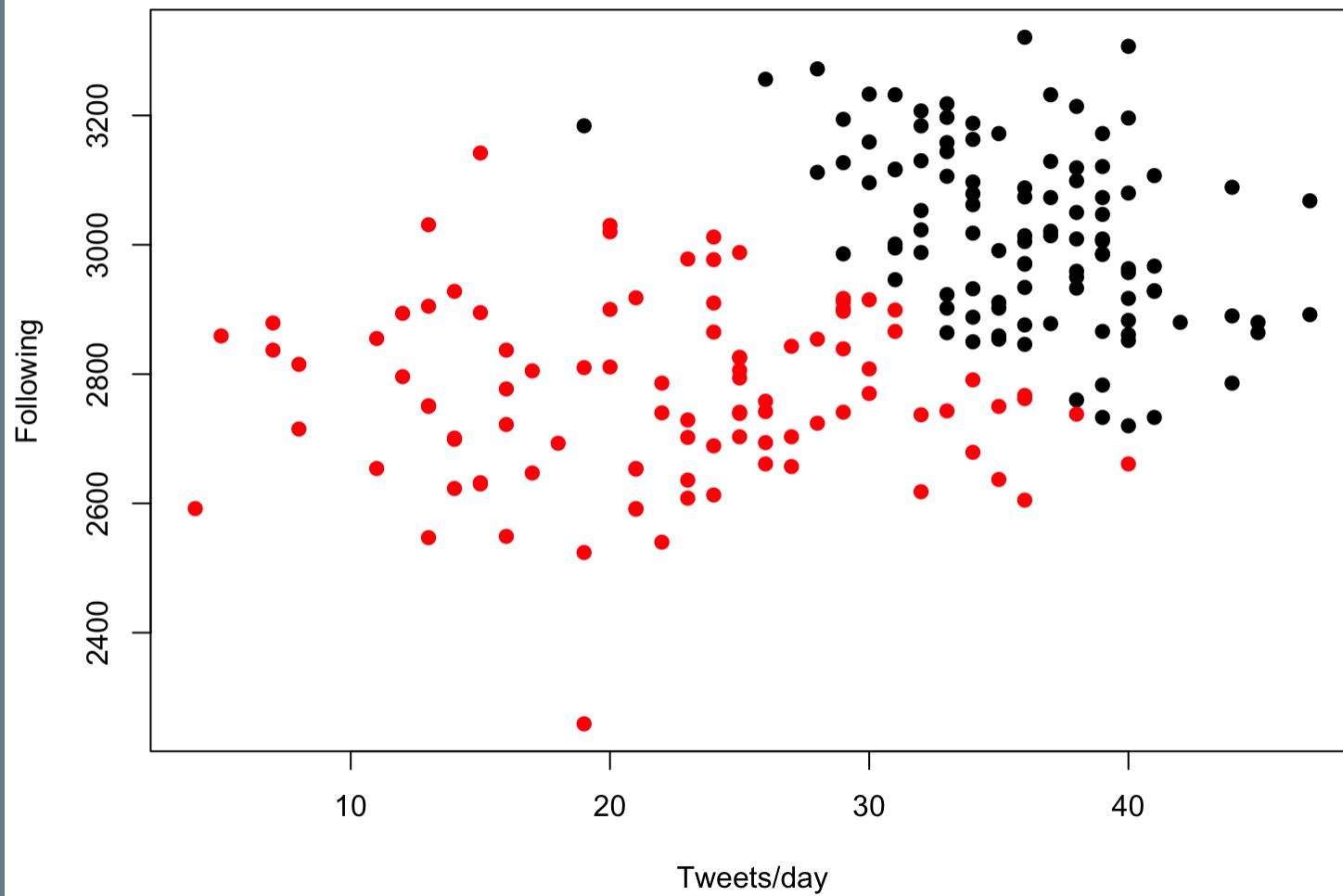
- this is where you would stop in explanatory modelling
- e.g. interpreting the coefficients
- predictive modelling focuses on the use of that model

PUTTING YOUR MODEL TO USE

```
data2$model_predictions = predict(my_first_model, data2, type = 'response')
data2$model_1 = ifelse(data2$model_predictions >= .5, 'real', 'bot')
```

	bot	real
bot	90	10
real	14	86

Algorithm-predicted account type



THE KEY CHALLENGE?

Think about what we did

PROBLEM OF INDUCTIVE BIAS

- remember: we learn from the data
- but what we really want to know is: how does it work on “unseen” data
- this is needed to estimate how good real predictions would be

How to solve this?

SPLITTING THE DATA

- split the data (e.g. 80%/20%, 60%/40%)
- use one part as TRAINING SET
- use the other as TEST SET

ENTER: CARET

```
library(caret)
```

- excellent package for ML in R
- well-documented [website](#)
- common interface for 200+ models

CARET: DATA PARTITIONING

```
set.seed(1)
in_training = createDataPartition(y = data3$account
                                  , p = .8
                                  , list = FALSE
                                  )
```

CREATING A TRAINING AND TEST SET

```
training_data = data3[ in_training, ]  
test_data = data3[-in_training, ]
```

SUPERVISED ML STEPS REVISED

- define outcome
- define features
- build model on the TRAINING SET
- evaluate model on the TEST SET

BUILDING AN SVM MODEL

```
my_second_model = train(account ~ .  
                        , data = training_data  
                        , method = "svmLinear"  
                        )
```

FIT/TEST THE SVM:

```
model_predictions = predict(my_second_model, test_data)
```

	bot	real
bot	19	1
real	4	16

BUT!

- our model might be really dependent on the training data
- we want to be more careful

Can we do some kind of safeguarding in the training data?

CROSS-VALIDATION

K-fold cross-validation

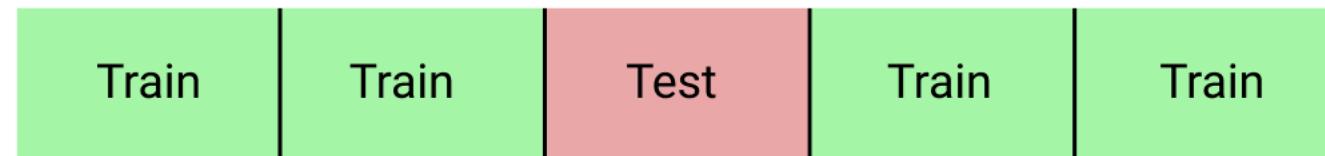
Iteration 1



Iteration 2



Iteration 3



Iteration 4



Iteration 5



img source

SPECIFYING CV IN CARET

```
training_controls = trainControl(method="cv"
                                  , number = 4
                                  , classProbs = TRUE
                                  )

my_third_model = train(account ~ .
                        , data = training_data
                        , trControl = training_controls
                        , method = "svmLinear"
                        )
```

my_third_model

```
## Support Vector Machines with Linear Kernel
##
## 160 samples
##   2 predictor
##   2 classes: 'bot', 'real'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 120, 120, 120, 120
## Resampling results:
##
##   Accuracy   Kappa
##   0.85       0.7
##
## Tuning parameter 'C' was held constant at a value of 1
```

ASSESS THE CVED MODEL

```
model_predictions = predict(my_third_model, test_data)
```

	bot	real
bot	19	1
real	4	16

ALGORITHM IN DETAIL

Naive Bayes

- rooted in Bayes' theorem
- closely related to Bayesian inference
- important for classification
- often used in NLP

BAYES RULE

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$$

In ML we are interested in:

$$P(outcome|data) = \frac{P(data|outcome)*P(outcome)}{P(data)}$$

Our scenario: text classification (normal vs spam email)

DATA

id	outcome	“cash”	“rich”	“gmail”	“happy”
1	normal	1	0	1	1
2	normal	0	0	1	0
3	spam	1	0	0	0
4	spam	1	1	0	0

PROBLEM IN BAYES TERMS

$$P(spam|text) = \frac{P(text|spam)*P(spam)}{P(text)}$$

What we know:

- $P(spam) = \frac{2}{4} = 0.5$ (prior of outcome)
- $P(text)$ is a constant (prior of data)

NEEDED: LIKELIHOOD

$P(text|spam) = P(word_1|spam) * P(word_2|spam) \dots P(word_n|spam)$

- $P('cash'|spam) = \frac{'cash'_{spam}}{'cash'_{total}} = \frac{2}{3} = 0.67$

WHAT IS $P('gmail' | \text{spam})$

id	outcome	“gmail”
1	normal	1
2	normal	1
3	spam	0
4	spam	0

PROBLEM OF ZERO PROBABILITIES

$$P('gmail' | \text{spam}) = \frac{0}{2} = 0.00$$

Results in zero for the whole of:

$$P(\text{text} | \text{spam}) = P(\text{word}_1 | \text{spam}) * P(\text{word}_2 | \text{spam}) \dots P(\text{word}_n | \text{spam})$$

LAPLACE SMOOTHING

$$P('gmail' | \text{spam}) = \frac{'gmail'_{\text{spam}} + \alpha}{'gmail'_{\text{total}} + \alpha * k}$$

For: $\alpha = 1$ and $k = 2$ (= no. of outcomes):

$$P('gmail' | \text{spam}) = \frac{1}{4} = 0.25$$

WE END UP WITH A MODEL

For any new text, we can use $P(\text{spam}|\text{text})$ to estimate whether it is spam or not.

GAUSSIAN NAIVE BAYES CLASSIFIER

"Gaussian" because this is a normal distribution

This is our prior belief

$$P(\text{class} \mid \text{data}) = \frac{P(\text{data} \mid \text{class}) \times p(\text{class})}{P(\text{data})}$$

We don't calculate this in naive bayes classifiers

ChrisAlbon

PERFORMANCE METRICS

SUPPOSE WE HAVE TWO MODELS

```
svm_model = train(account ~ .  
                  , data = training_data  
                  , trControl = training_controls  
                  , method = "svmLinear"  
                  )  
  
nb_model = train(account ~ .  
                  , data = training_data  
                  , trControl = training_controls  
                  , method = "nb"  
                  )
```

TEST SET RESULTS

SVM model

	bot	real
bot	96	104
real	102	98

NB model

	bot	real
bot	124	76
real	140	60

THE CONFUSION MATRIX

	Bot	Real
Bot	True positives	False negatives
Real	False positives	True negatives

FALSE (TRUE) POSITIVES (NEGATIVES)

- true positives (TP): correctly identified bots
- true negatives (TN): correctly identified real accounts
- false positives (FP): false accusations
- false negatives (FN): missed bots

MOST INTUITIVE: ACCURACY

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

$$acc_{svm} = \frac{(111+86)}{400} = 0.49$$

$$acc_{nb} = \frac{(124+60)}{400} = 0.47$$

Any problems with that?

ACCURACY

Model 1

	Bot	Real
Bot	100	100
Real	5	195

Model 2

	Bot	Real
Bot	150	50
Real	55	145

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 Bot Real Sum
##     Bot   100   100 200
##     Real    5   195 200
##     Sum   105   295 400
```

```
##           prediction
## reality Bot Real Sum
##     Bot   150    50 200
##     Real   55   145 200
##     Sum   205   195 400
```

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:

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

PRECISION

```
##           prediction
## reality Bot Real Sum
##     Bot   100   100 200
##     Real    5   195 200
##     Sum   105   295 400
```

- $Pr_{bot} = \frac{100}{105} = 0.95$
- $Pr_{real} = \frac{195}{295} = 0.64$

COMPARING THE MODELS

	Model 1	Model 2
acc	0.74	0.74
Pr_{bot}	0.95	0.73
Pr_{real}	0.64	0.74

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:

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

RECALL

```
##           prediction
## reality Bot Real Sum
##     Bot   100   100 200
##     Real    5   195 200
##     Sum   105   295 400
```

- $R_{bot} = \frac{100}{200} = 0.50$
- $R_{real} = \frac{195}{200} = 0.98$

COMPARING THE MODELS

	Model 1	Model 2
acc	0.74	0.74
Pr_{bot}	0.95	0.73
Pr_{real}	0.64	0.74
R_{bot}	0.50	0.75
R_{real}	0.98	0.73

COMBINING PR AND R

The F_1 measure.

Note: we combine Pr and R for each class, so we get two F_1 measures.

Formally:

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

F1 MEASURE

```
##           prediction
## reality Bot Real Sum
##     Bot   100   100 200
##     Real    5   195 200
##     Sum   105   295 400
```

- $F1_{bot} = 2 * \frac{0.95*0.50}{0.95+0.50} = 2 * \frac{0.475}{1.45} = 0.66$
- $F1_{real} = 2 * \frac{0.64*0.98}{0.64+0.98} = 2 * \frac{0.63}{1.62} = 0.77$

COMPARING THE MODELS

	Model 1	Model 2
acc	0.74	0.74
Pr_{bot}	0.95	0.73
Pr_{real}	0.64	0.74
R_{bot}	0.50	0.75
R_{real}	0.98	0.73
$F1_{bot}$	0.66	...
$F1_{real}$	0.77	...

IN CARET

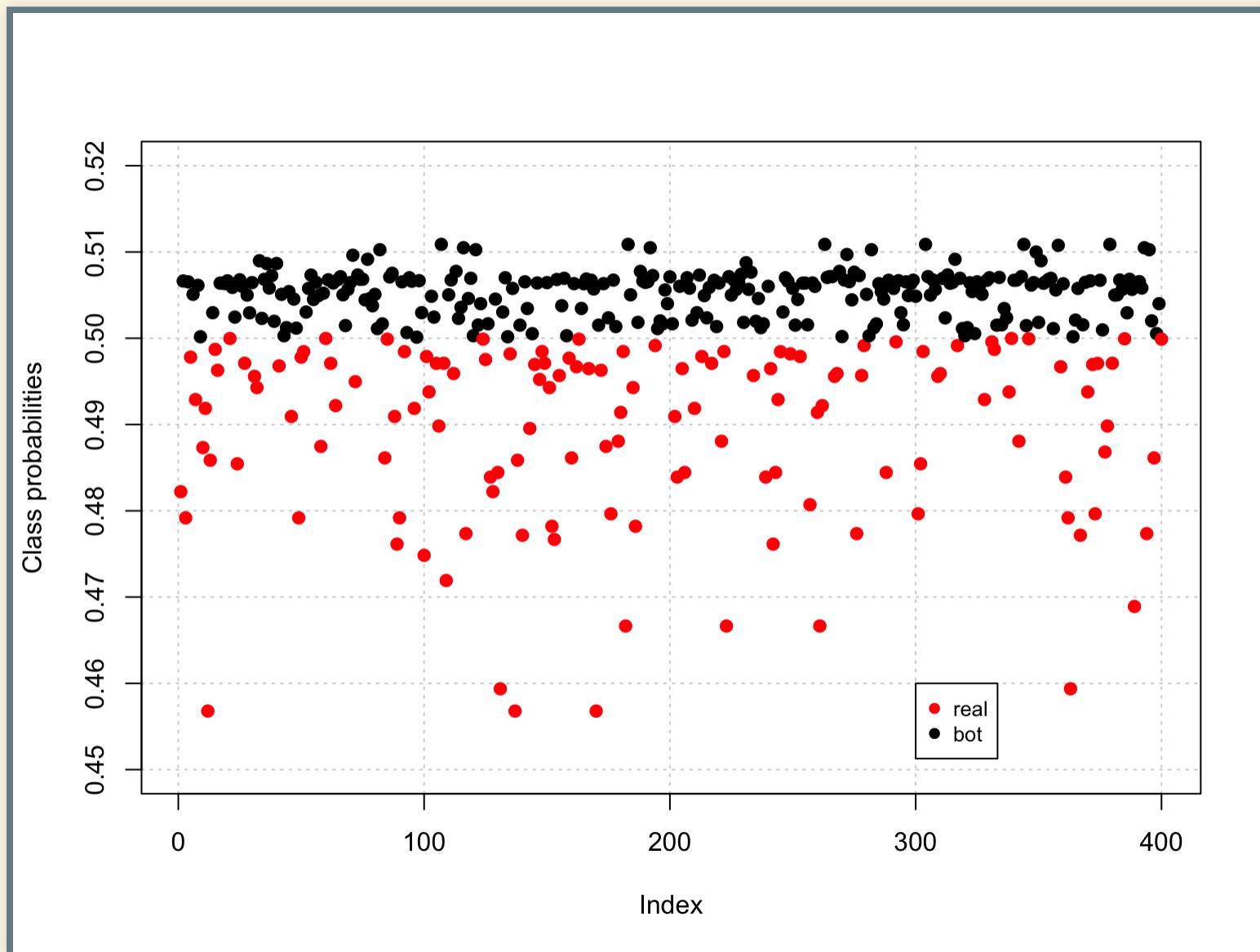
```
confusionMatrix(nb_pred, as.factor(test_data$account))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction bot  real
##           bot 124 140
##           real  76  60
##
##                   Accuracy : 0.46
##                           95% CI : (0.4104, 0.5102)
##   No Information Rate : 0.5
##   P-Value [Acc > NIR] : 0.9506
##
##                   Kappa : -0.08
##   Mcnemar's Test P-Value : 1.814e-05
##
##                   Sensitivity : 0.6200
##                   Specificity  : 0.3000
##   Pos Pred Value  : 0.4697
```

THERE'S MORE

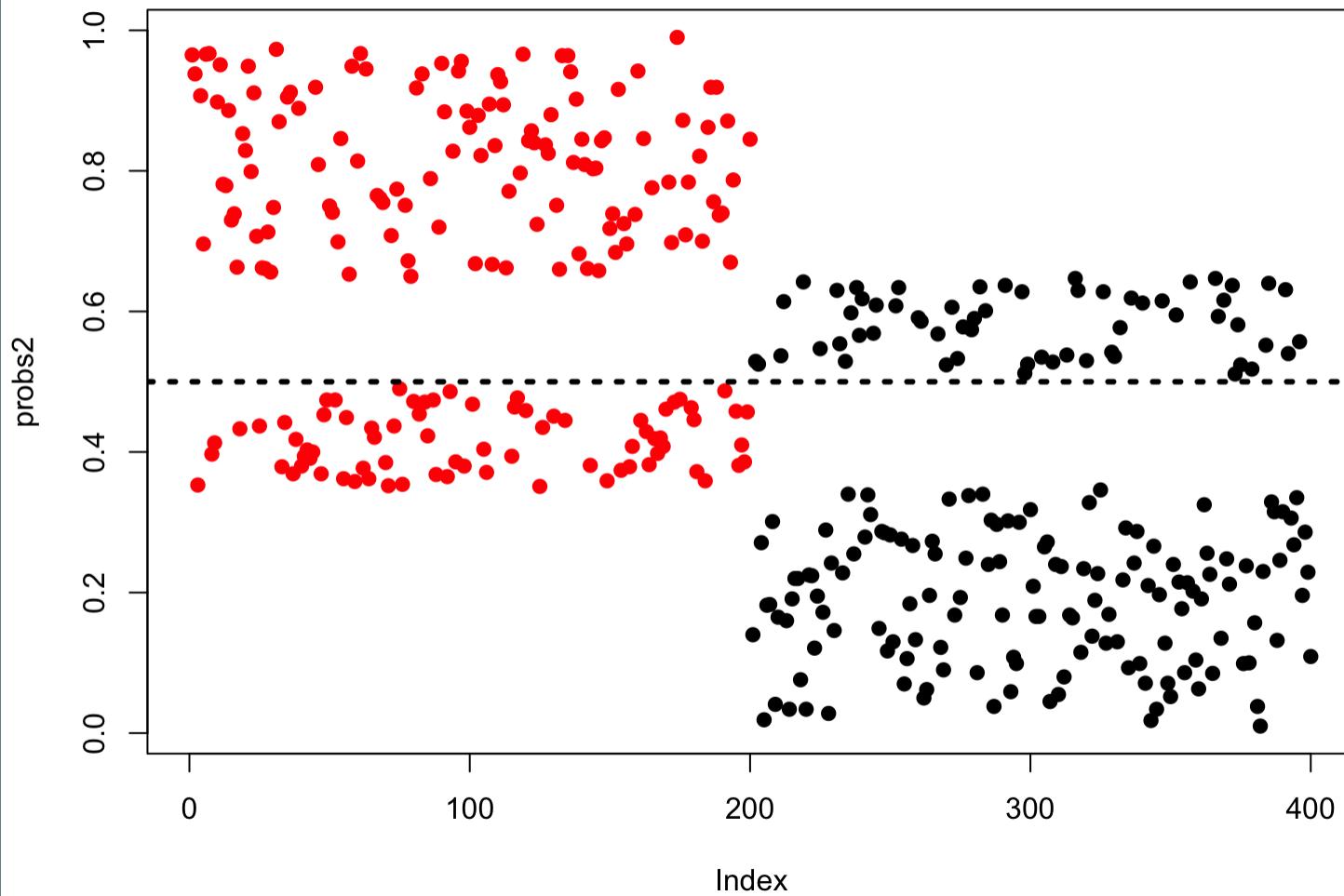
What's behind the model's predictions?

CLASS PROBABILITIES



THE THRESHOLD PROBLEM

Also class probabilities for acc = 0.46



ISSUE!

- classification threshold little informative
- obscures certainty in judgment

Needed: a representation across all possible values

THE AREA UNDER THE CURVE (AUC)

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

AUC STEP-WISE

```
#for each class probability:  
threshold_1 = probs[1]  
threshold_1
```

```
## [1] 0.4822156
```

```
pred_threshold_1 = ifelse(probs >= threshold_1, 'bot', 'real')
```

	bot	real
bot	183	17
real	188	12

SENSITIVITY AND 1-SPECIFICITY

	bot	real
bot	183	17
real	188	12

$$Sens. = 183/200 = 0.92$$

$$Spec. = 12/200 = 0.06$$

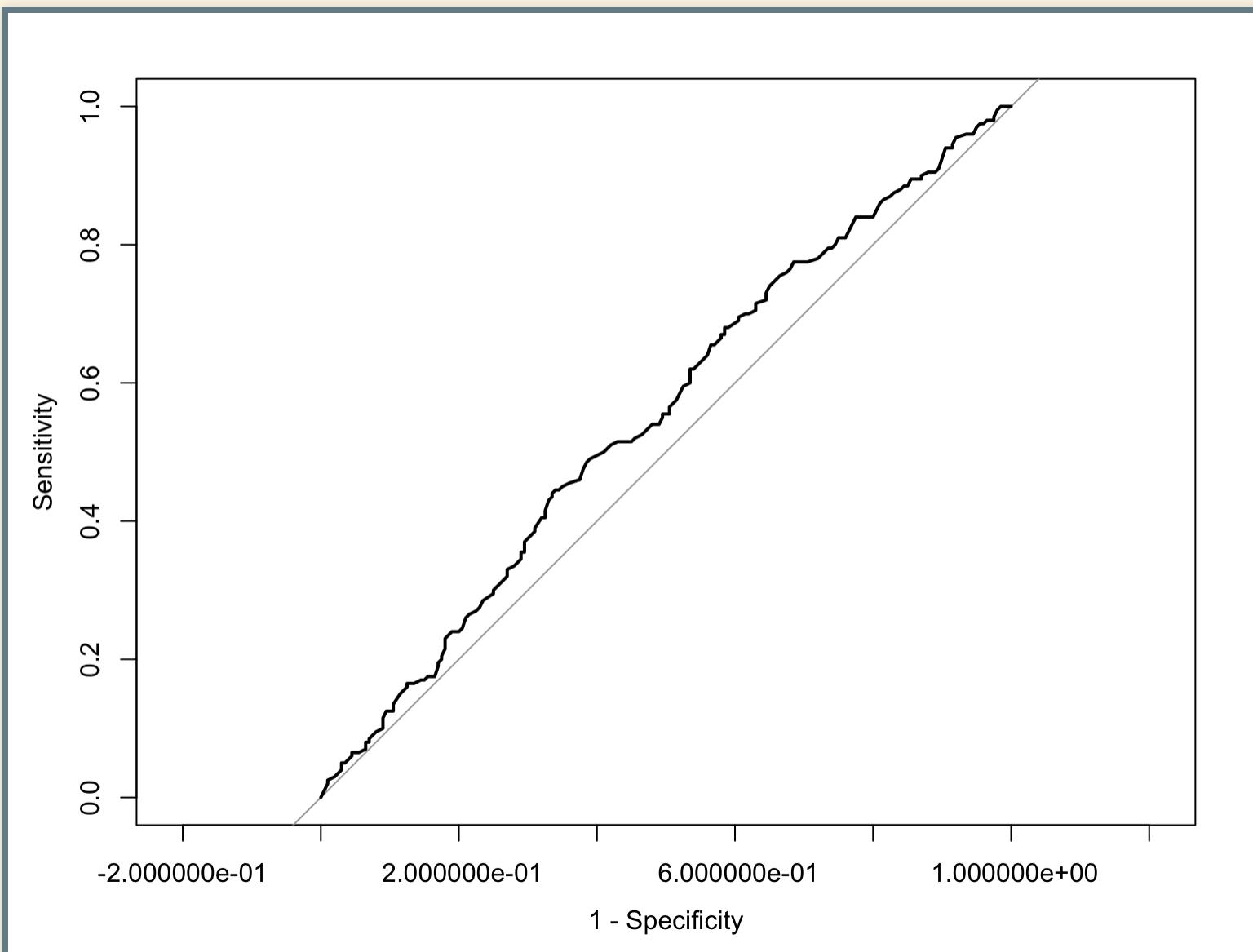
$$Sens. = 183/200 = 0.92$$

$$Spec. = 12/200 = 0.06$$

Threshold	Sens.	1-Spec
0.48	0.92	0.94

Do this for every threshold observed.

PLOT THE RESULTS:



QUANTIFY THIS PLOT

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

```
##  
## Call:  
## roc.default(response = test_data$account, predictor = probs,      ci =  
##  
## Data: probs in 200 controls (test_data$account bot) < 200 cases (test_  
## Area under the curve: 0.5534  
## 95% CI: 0.4971-0.6098 (DeLong)
```

WHAT'S NEXT?

- Today's tutorial + homework: text classification, performance metrics

Next week: Machine Learning 2