Machine Learning

Chapter 2: Classification I

January 2023

Contents

1

The classification problem

The 7 Steps of Machine Learning

1.1

Problem statement

Problem statement

- Given a set of *n* attributes (features), a set of *m* classes, and a set of *N* labeled training instances:

$$\{(\boldsymbol{x}[i], y[i])\} \text{ with } \boldsymbol{x}[i] \in \mathbb{R}^n, y[i] \in \{c_1, c_2, \dots, c_m\}, i = 1, \dots, N$$

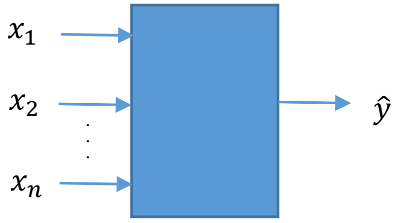
$\boldsymbol{x}$ y

age	gender	height	weight	ap_hi	ap_lo	cholest	gluc	smoke	alco	active	cardio
46.0	F	172	112	120	80	1	1	0	0	0	YES
44.0	M	170	69	120	70	1	1	0	0	1	NO
45.5	M	159	49	120	70	1	1	0	0	1	NO
39.7	F	164	48	110	70	1	2	1	1	1	YES
63.3	F	180	104	120	85	2	2	0	0	1	NO

N

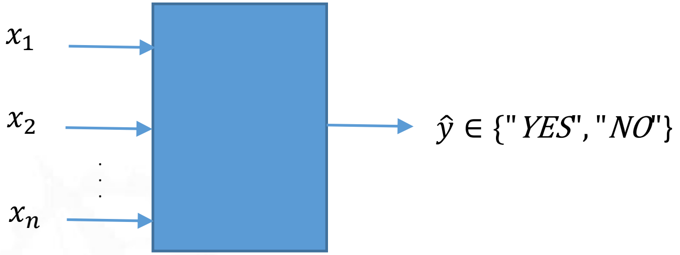
n 1

determine a classification rule that predicts the class of any instance from the value of its attributes: $\hat{y} = f(\boldsymbol{x}, \boldsymbol{w})$

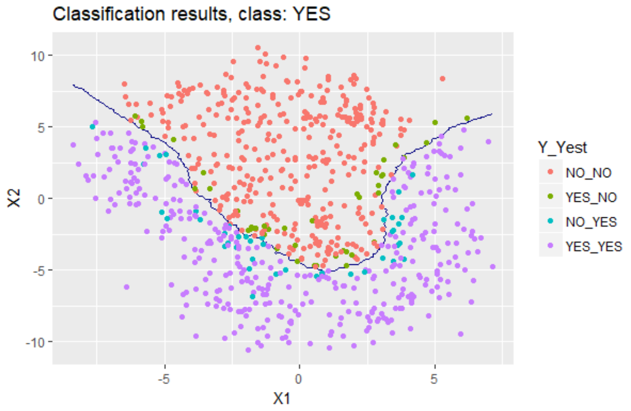


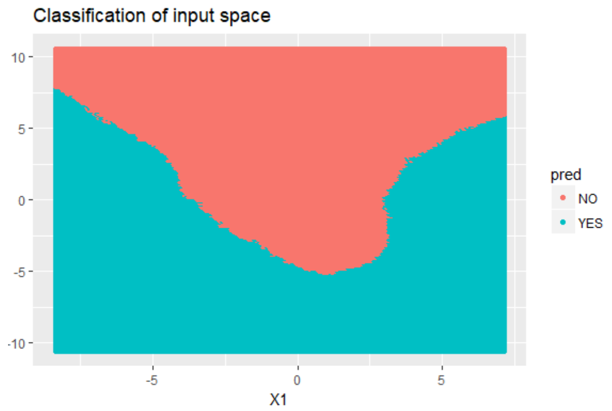
The classification problem
Problem statement

- 2 different approaches:
  - "Hard" partition:



$$\hat{y} \in \{\text{"}YES\text{"}, \text{"}NO\text{"}\}$$

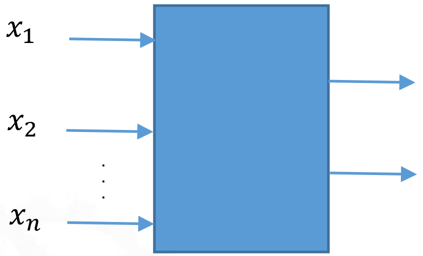




See: https://odsc.medium.com/identifying-poisonous-mushrooms-with-rule-learners-debf52748

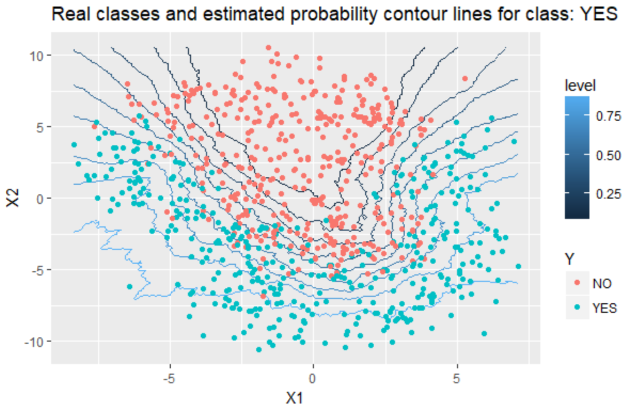
The classification problem
Problem statement

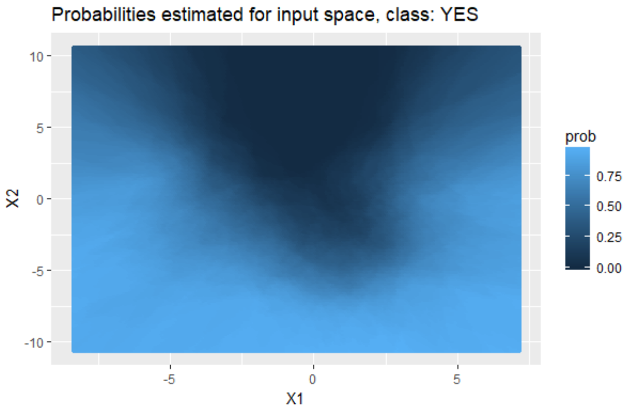
- 2 different approaches:
  - “Soft” partition:

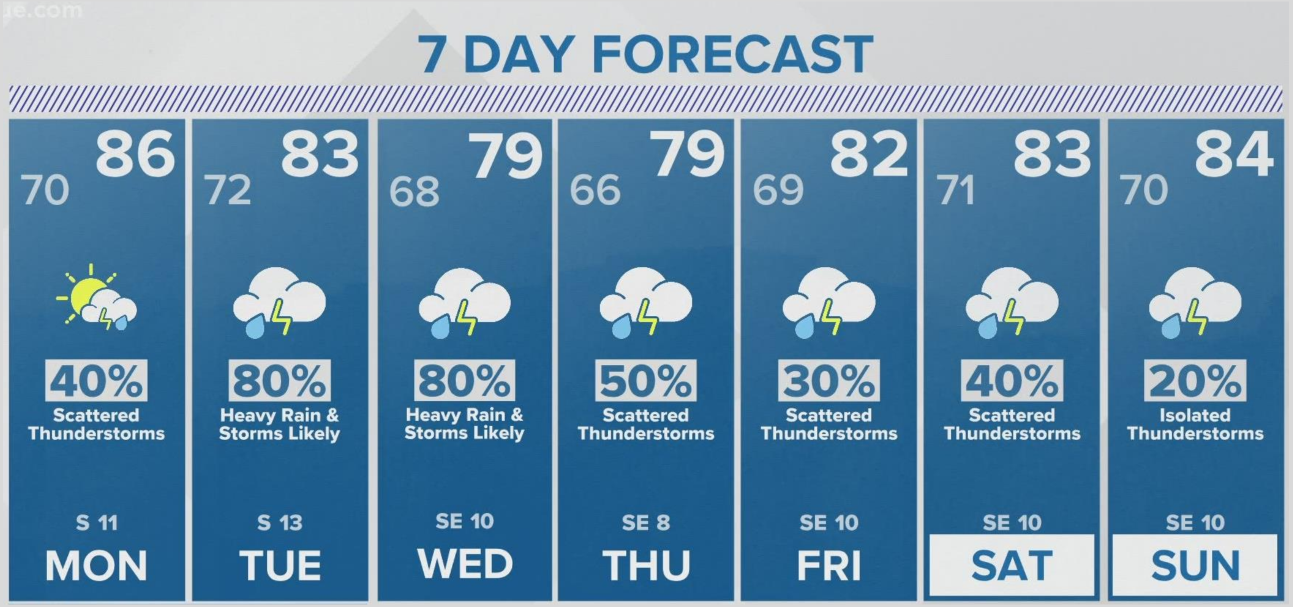


$$\hat{p}_1(\boldsymbol{x}) = \hat{P}\left(\frac{"YES"}{\boldsymbol{x}}\right)$$

$$\hat{p}_2(\boldsymbol{x}) = \hat{P}\left(\frac{"NO"}{\boldsymbol{x}}\right)$$







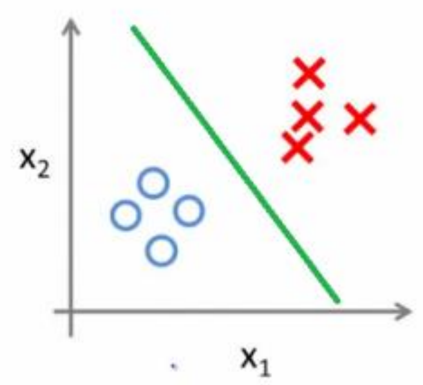
7 DAY FORECAST

MON	TUE	WED	THU	FRI	SAT	SUN
70	72	68	66	69	71	70
86	83	79	79	82	83	84
40%	80%	80%	50%	30%	40%	20%
Scattered Thunderstorms	Heavy Rain & Storms Likely	Heavy Rain & Storms Likely	Scattered Thunderstorms	Scattered Thunderstorms	Scattered Thunderstorms	Isolated Thunderstorms
S 11	S 13	SE 10	SE 8	SE 10	SE 10	SE 10

The classification problem
Problem statement

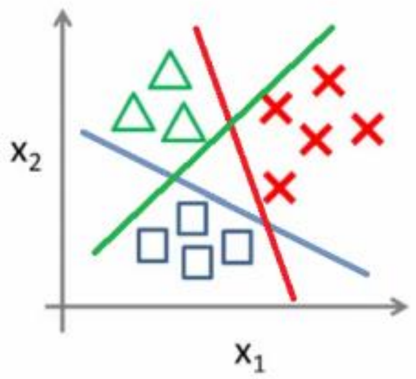
- If observations are grouped in just two categories, or classes, this is a problem of ***binary classification***.

Binary classification:



- If the number of possible categories exceeds two, this is a ***multiclass problem***.

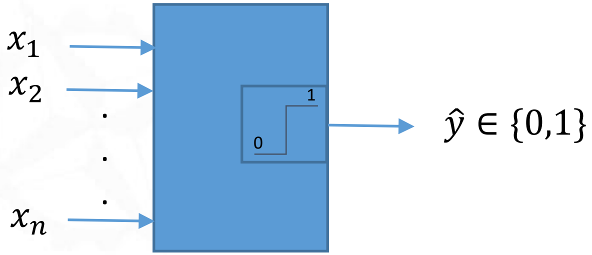
Multi-class classification:



The classification problem
Problem statement

- ***Hard partition*** for binary classification:

$$\{(\boldsymbol{x}[i], y[i])\} \text{ with } \boldsymbol{x}[i] \in \mathbb{R}^n, y[i] = \begin{cases} 0 \; if \; c_0 \\ 1 \; if \; c_1 \end{cases}, \; i = 1, \dots, N$$



MSE loss function:

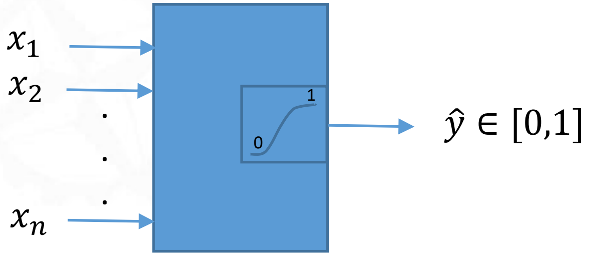
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y[i] - \hat{y}[i])^2 = \; ???$$

The classification problem

Problem statement

- ***Soft partition*** using the ***least squares approach*** for binary classification:

$$\{(\boldsymbol{x}[i], y[i])\} \text{ with } \boldsymbol{x}[i] \in \mathbb{R}^n, y[i] = \begin{cases} 0 \; if \; c_0 \\ 1 \; if \; c_1 \end{cases}, \; i = 1, \dots, N$$



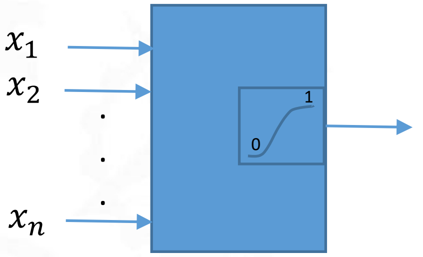
MSE loss function:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y[i] - \hat{y}[i])^2$$

The classification problem
Problem statement

- ***Soft partition*** using the ***probabilistic approach*** for binary classification:

$$\{(\boldsymbol{x}[i], y[i])\} \text{ with } \boldsymbol{x}[i] \in \mathbb{R}^n, y[i] = \begin{cases} 0 \; if \; c_0 \\ 1 \; if \; c_1 \end{cases}, \; i = 1, \ldots, N$$



$$\hat{p}(\boldsymbol{x}) = \hat{P}\left(\frac{y=1}{\boldsymbol{x}}\right)$$

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p}(x) < 0.5? \\ 1 \; if \hat{p}(x) \geq 0.5? \end{cases}$$

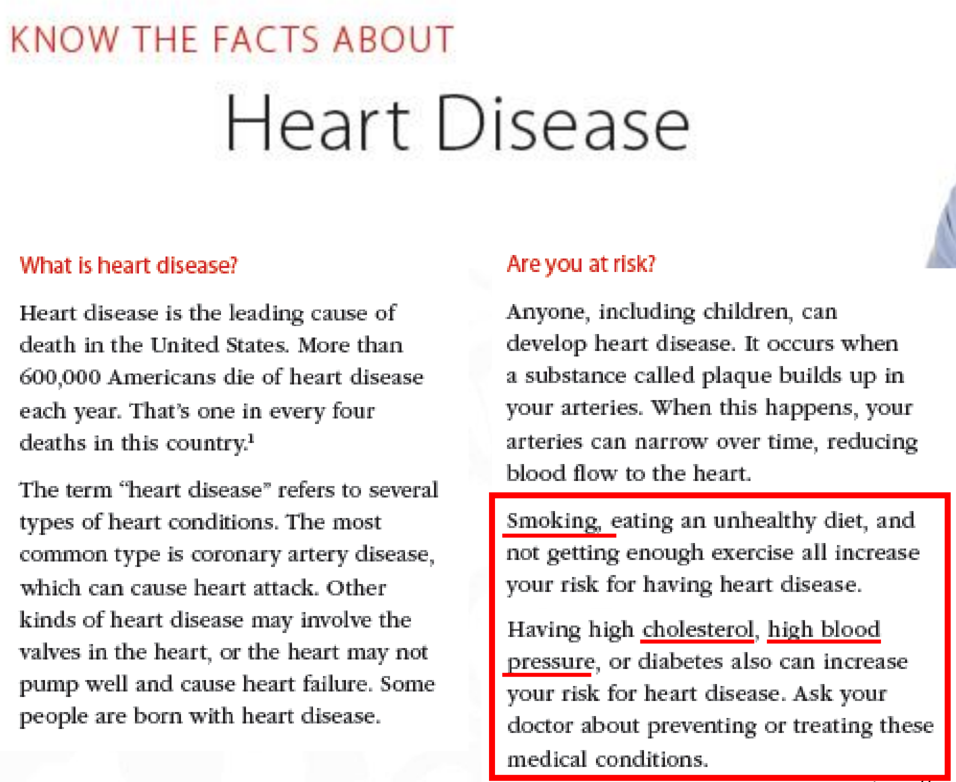
Cross-entropy loss *(→"(-1)xlog-likelihood"):*

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y[i]log\big(\hat{p}(\boldsymbol{x})\big) + (1 - y[i])log\big(1 - \hat{p}(\boldsymbol{x})\big)\right]$$

1.2

Example

The classification problem
Example: Heart Disease Prediction



KNOW THE FACTS ABOUT

Heart Disease

What is heart disease?

Heart disease is the leading cause of death in the United States. More than 600,000 Americans die of heart disease each year. That's one in every four deaths in this country.[1]

The term "heart disease" refers to several types of heart conditions. The most common type is coronary artery disease, which can cause heart attack. Other kinds of heart disease may involve the valves in the heart, or the heart may not pump well and cause heart failure. Some people are born with heart disease.

Are you at risk?

Anyone, including children, can develop heart disease. It occurs when a substance called plaque builds up in your arteries. When this happens, your arteries can narrow over time, reducing blood flow to the heart.

Smoking, eating an unhealthy diet, and not getting enough exercise all increase your risk for having heart disease.

Having high cholesterol, high blood pressure, or diabetes also can increase your risk for heart disease. Ask your doctor about preventing or treating these medical conditions.

Source: https://www.cdc.gov/heartdisease/facts.htm

The classification problem
Example: Heart Disease Prediction

STEPS:

1) Collect data

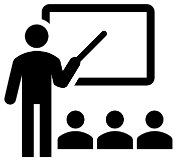
2) Preprocess (clean)

3) Choose model

4) Fit the parameters

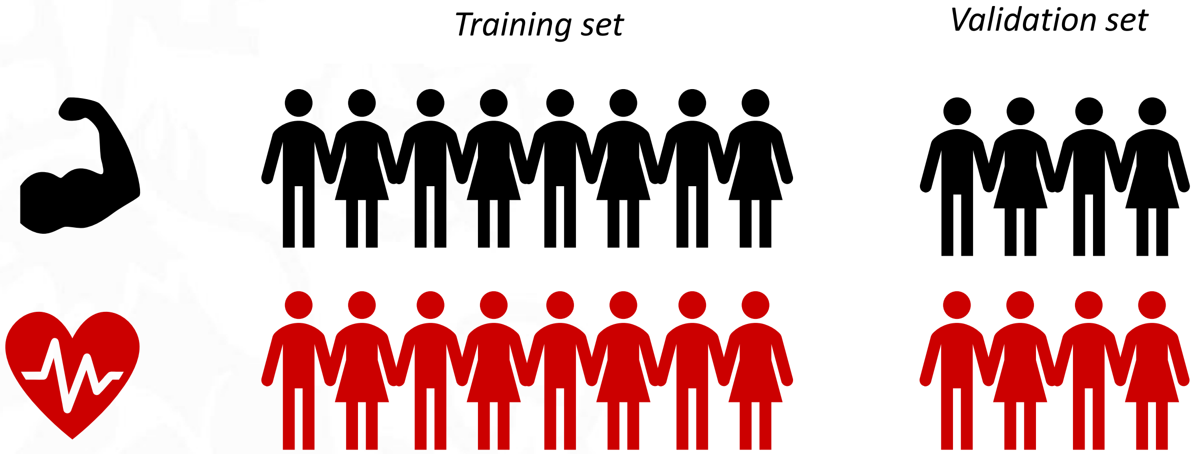
5) Generalize?

The classification problem
Example



- Solution:

  1) ***Collect a dataset (S)*** of labeled cases and split it into a *training set* and a *validation set*:

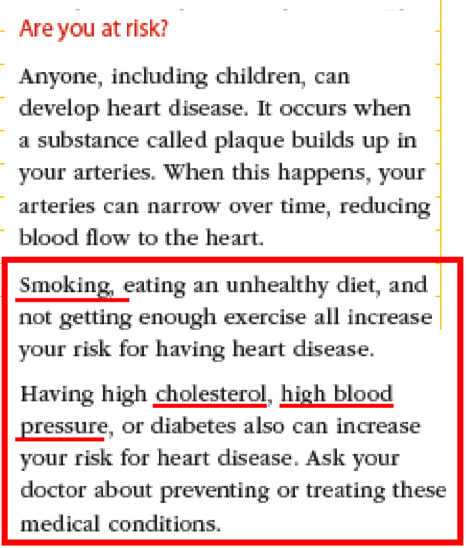


The classification problem
Example

2) Identify an appropriate set of input variables or ***features*** ($x_1, x_2,\dots x_n$), ***preprocess*** and ***clean*** the data.

Cleveland Heart Disease dataset from the UCI Repository

age	gender	height	weight	ap_hi	ap_lo	cholest	gluc	smoke	alco	active	cardio
46.0	F	172	112	120	80	1	1				
44.0	M	170	69	120	70	1	1				
45.5	M	159	49	120	70	1	1				
39.7	F	164	48	110	70	1	2				
63.3	F	180	104	120	85	2	2				
48.1	M	153	73	120	80	2	1				
51.6	M	158	70	120	80	1	1				
60.7	M	172	87	150	100	3	1				
50.1	M	163	108	150	100	3	3				
49.5	M	170	90	120	80	1	1				



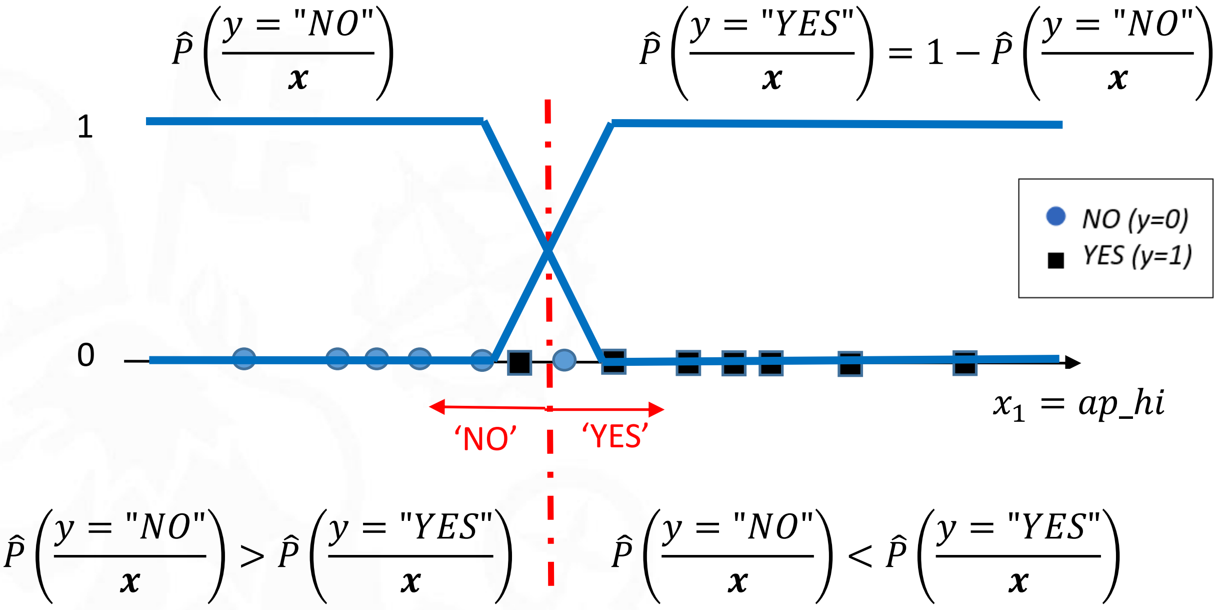
Are you at risk?

Anyone, including children, can develop heart disease. It occurs when a substance called plaque builds up in your arteries. When this happens, your arteries can narrow over time, reducing blood flow to the heart.

Smoking, eating an unhealthy diet, and not getting enough exercise all increase your risk for having heart disease.

Having high cholesterol, high blood pressure, or diabetes also can increase your risk for heart disease. Ask your doctor about preventing or treating these medical conditions.

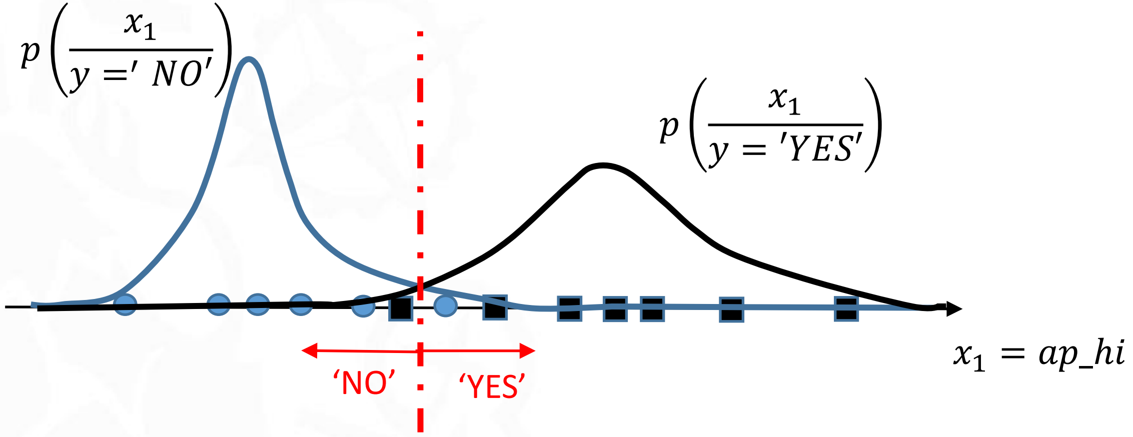
The classification problem
Example



$\hat{P}\left(\frac{y = "NO"}{\boldsymbol{x}}\right)$
$\hat{P}\left(\frac{y = "YES"}{\boldsymbol{x}}\right) = 1 - \hat{P}\left(\frac{y = "NO"}{\boldsymbol{x}}\right)$
1
0
● NO (y=0)
■ YES (y=1)
$x_1 = ap\_hi$
'NO'
'YES'
$\hat{P}\left(\frac{y = "NO"}{\boldsymbol{x}}\right) > \hat{P}\left(\frac{y = "YES"}{\boldsymbol{x}}\right)$
$\hat{P}\left(\frac{y = "NO"}{\boldsymbol{x}}\right) < \hat{P}\left(\frac{y = "YES"}{\boldsymbol{x}}\right)$

The classification problem
Example

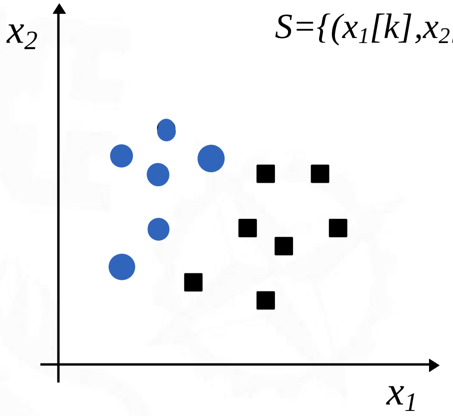
- Bayes Theorem: $P\left({}'YES'/\mathbf{x}\right) = \dfrac{p\left(\mathbf{x}/{}'YES'\right)\ P({}'YES')}{p(\mathbf{x})}$



$p\left(\frac{x_1}{y = 'NO'}\right)$
$p\left(\frac{x_1}{y = 'YES'}\right)$
'NO'
'YES'
$x_1 = ap\_hi$

The classification problem
Example

- If we add an appropriate new feature x_2:



$S=\{(x_1[k],x_2[k],y[k])\}$ = *Training set*

● *NO (y=0)*
■ *YES (y=1)*

$$\boxed{(x_1, x_2) \rightarrow y?}$$

The classification problem
Example

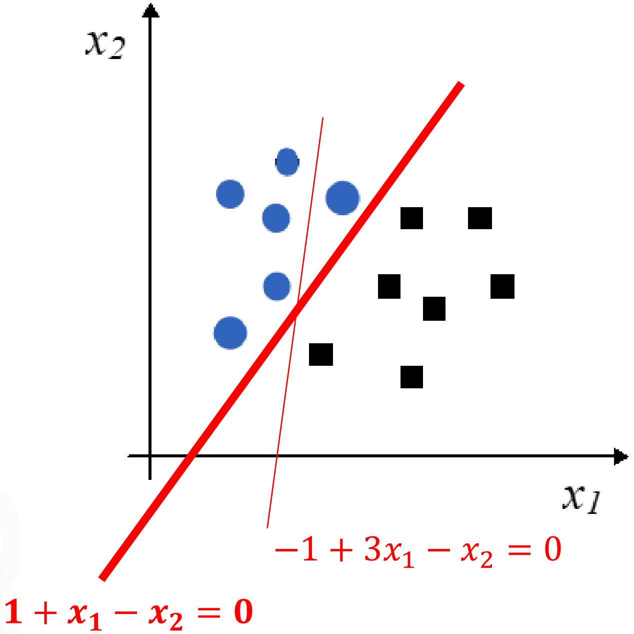
3) Select a ***classification model***:

- Classification ≡ mapping from the feature space X to the set of labels Y

- The mapping can be modeled as a mathematical function with a given number of free parameters ***w:***

$$\hat{y} = f(\mathbf{x}, \mathbf{w})$$

$$\hat{y} = sign(w_0 + w_1 x_1 + w_2 x_2)$$



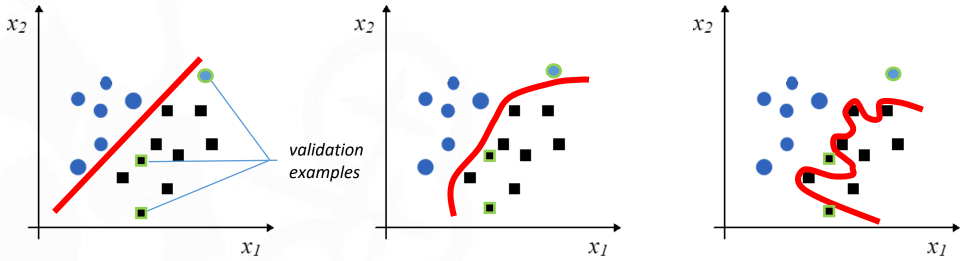
The classification problem
Example

4) ***Fit the free parameters*** of the classifier (***train***) by minimizing an error or loss function defined over the training set:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y[i] - \hat{y}[i])^2$$

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y[i] * log\big(\hat{p}(\boldsymbol{x})\big) + (1 - y[i]) * log\big(1 - \hat{p}(\boldsymbol{x})\big)\right]$$

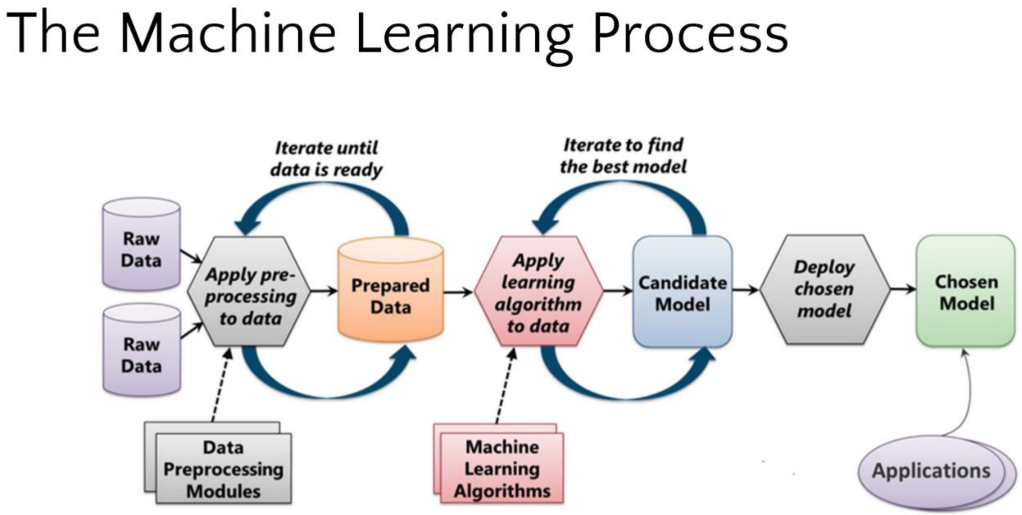
5) Estimate the ***generalization capabilities*** of the classifier using the ***validation*** set to prevent overfitting:



1.3

Preprocessing tools

The classification problem
Preprocessing: Introduction


The Machine Learning Process
Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

From "Introduction to Microsoft Azure" by David Chappell

The classification problem
Preprocessing: basic steps

Steps in Data Preprocessing

Step 1 : Import the data-set

Step 2 : Check out the **missing values**

Step 3 : Plot the data and check out for **outliers**

Step 4 : Encode the **categorical variables** (*strings are not factors*!)

Step 5 : Analyze the **continuous variables** (***feature selection***)

Step 6 : Check out for **Class Imbalances**

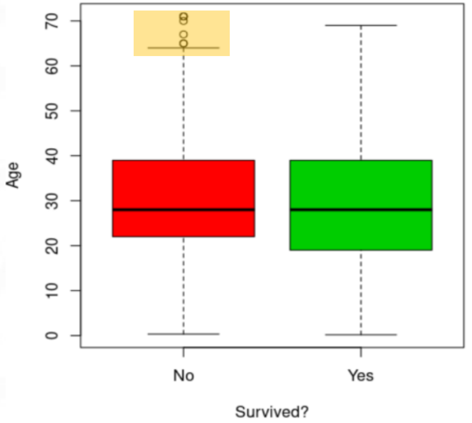
Step 7: Split the data-set into Training, Validation and Test Sets

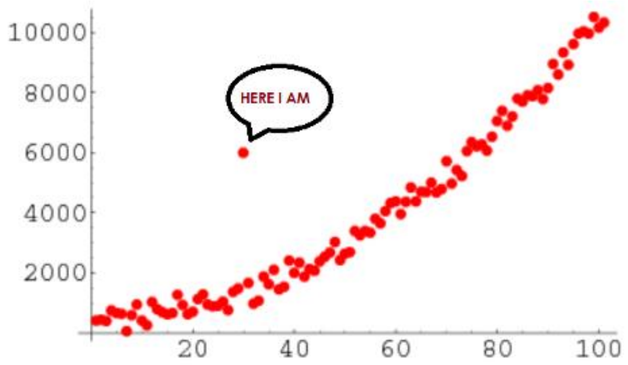
Preprocessing: Missing values

- In many cases, some predictors have no values for a given sample. (NA = not available)
- **Large data sets** → removal of samples with missing values is not a problem, assuming that the missingness is not informative.
- **Smaller data sets** → steep price in removing samples. There are two general approaches:
  - First, a few predictive models, especially tree-based techniques, can specifically account for missing data.
  - Alternatively, we can use information in the training set predictors to estimate the values of other predictors (*K*-nearest neighbor model is used very often for this purpose).

The classification problem
Preprocessing: Outliers

- ***Outliers*** can be generally defined as *"samples that are exceptionally far from the mainstream of the data"*.





How to detect them?

Plotting
- Visually
- Box-plots
- qq-plots

Fitting a model first

The classification problem
Preprocessing: Outliers

Type of outliers:

- When one or more samples are suspected to be outliers, the first step is to make sure that the values are scientifically valid and that no data ***recording errors*** have occurred.

- With small sample sizes, ***apparent outliers*** might be a result of a skewed distribution where there are not yet enough data to see the skewness.

- “True” informative outliers.

The classification problem
Preprocessing: Encoding categorical variables

- When a predictor is ***categorical (factor)***, it is common to decompose the predictor into a set of more specific variables:

5 values

Value	X_blond	X_black	X_red	X_grey
Brown	0	0	0	0
Blond	1	0	0	0
Black	0	1	0	0
Red	0	0	1	0
Grey	0	0	0	1

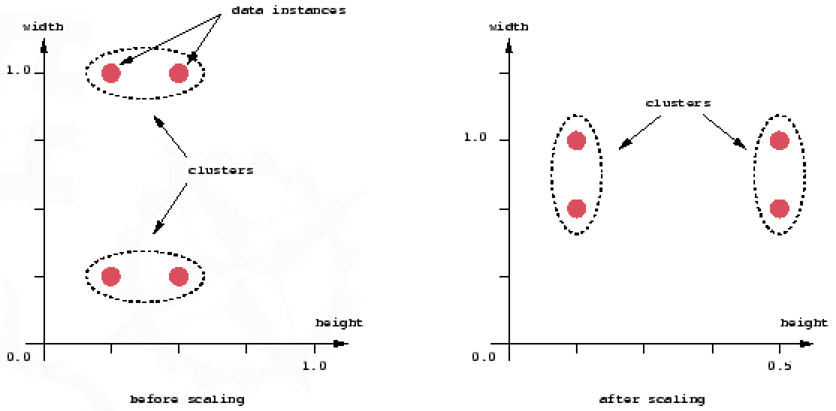
Source: https://en.wikipedia.org/wiki/Human_hair_colo

4 factors

(dummy variables)

The classification problem
Preprocessing: Scaling

- Many ***Machine Learning algorithms*** are affected by the scale of the predictors (*e.g.* distance functions).



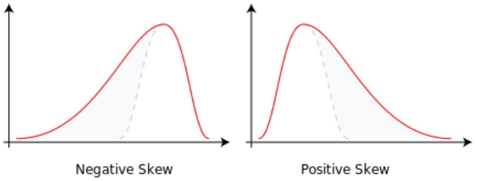
- ***Standardization***: Standard scores are also called z-values, z-scores, normal scores, and standardized variables:

$$\boxed{x^* = \frac{x - \bar{x}}{\sigma_x}}$$

The classification problem
Preprocessing: Skewness

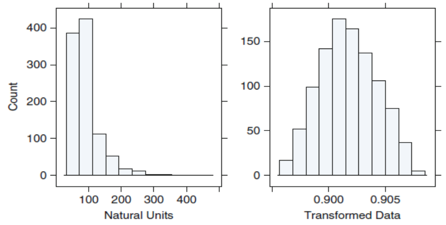
- Transformations to resolve ***skewness***:
  - An un-skewed distribution is one that is roughly symmetric.
  - A right-skewed distribution has a large number of points on the left side of the distribution (smaller values) than on the right side (larger values)

$$skewness = E\left[\left(\frac{x-\bar{x}}{\sigma_x}\right)^3\right]$$



  - The ***Box Cox transformation*** can be used to make the distribution of the variable as normal as possible (skewness →0):

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

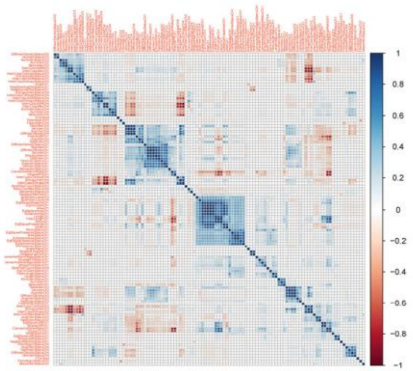


Source: (Kuhn et al., 2013)

This family can identify square transformation (λ = 2), square root (λ = 0.5), inverse (λ = −1), and others in-between.

The classification problem
Preprocessing: Collinearity

- **Fewer predictors** means decreased computational time and complexity.
- **Collinearity:** If two predictors are **highly correlated**, they are measuring the same underlying information:
  - Unstable models
  - Degraded predictive performance
  - Misleading explanations



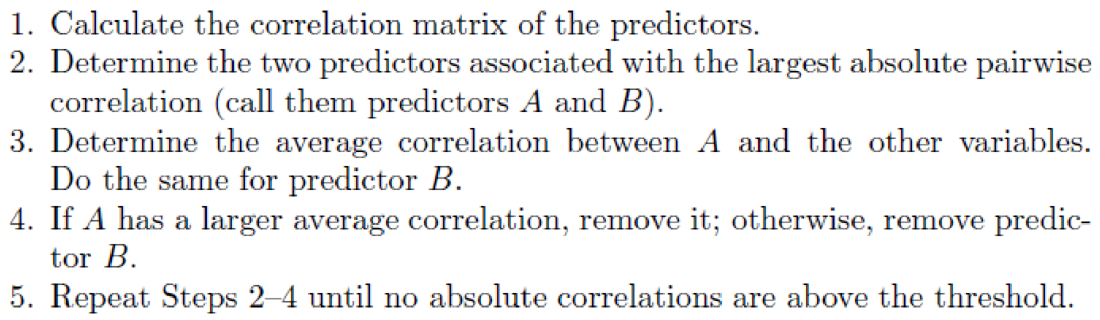
- **Removing one should** lead to a more robust, parsimonious and interpretable model.

Source: (Kuhn et al., 2013)

The classification problem
Preprocessing: Collinearity

- PCA can be used to characterize the magnitude of the problem and correct it.

- A more heuristic approach to dealing with this issue is to remove the minimum number of predictors to ensure that all pairwise correlations are below a certain threshold:



1. Calculate the correlation matrix of the predictors.
2. Determine the two predictors associated with the largest absolute pairwise correlation (call them predictors A and B).
3. Determine the average correlation between A and the other variables. Do the same for predictor B.
4. If A has a larger average correlation, remove it; otherwise, remove predictor B.
5. Repeat Steps 2–4 until no absolute correlations are above the threshold.

The classification problem
Multicollinearity

- **Variance Inflation Factor**:

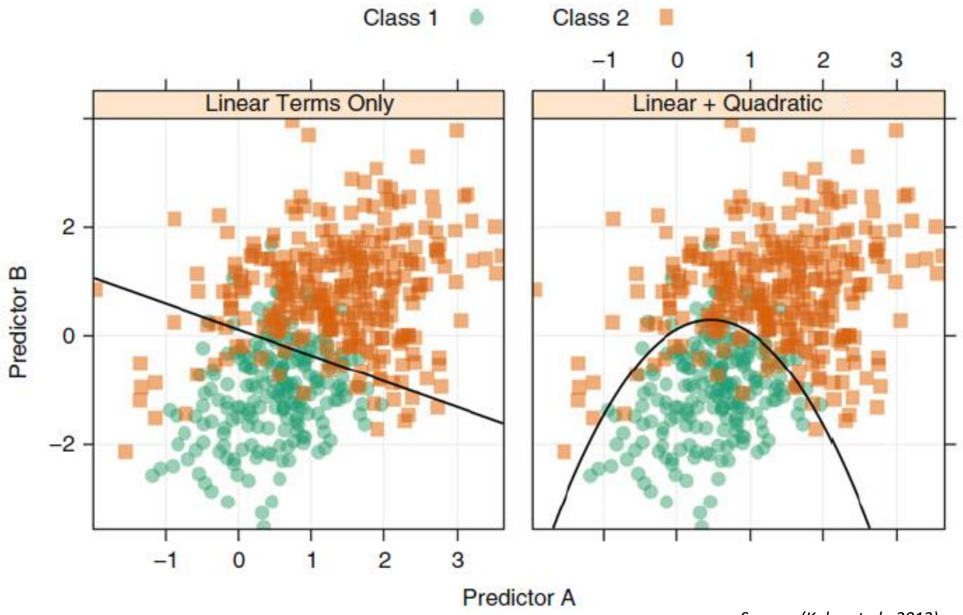
$$VIF(X_i) = \frac{1}{(1 - R_i^2)}$$

Where R_i^2 is the coefficient of determination resulting from regressing x_i on the remaining *n-1* regressor variables.

As a rule of thumb, if *VIF>10* then multicollinearity is a problem (90% of the variance of x_i is explained by the other input variables).

The classification problem
Preprocessing: Adding nonlinear terms

- It is a common practice to include nonlinear combinations of the predictors in linear models:



Source: (Kuhn et al., 2013)

The classification problem
Preprocessing: Class Imbalances

- In classification problems, a disparity in the frequencies of the observed classes can have a significant negative impact on model fitting.
- One possible solution: subsample the training data in a manner that mitigates the issues.
- Examples:
  - down-sampling: randomly subset all the classes in the training set so that their class frequencies match the least prevalent class.
  - up-sampling: randomly sample (with replacement) the minority class to be the same size as the majority class.
  - hybrid methods: Down-sample the majority class and synthesize new data points in the minority class.
  - ROSE: ROSE (Random Over-Sampling Examples)

- **NOTE: You would never want to artificially balance the Test set.**

1.4

Measuring classification performance

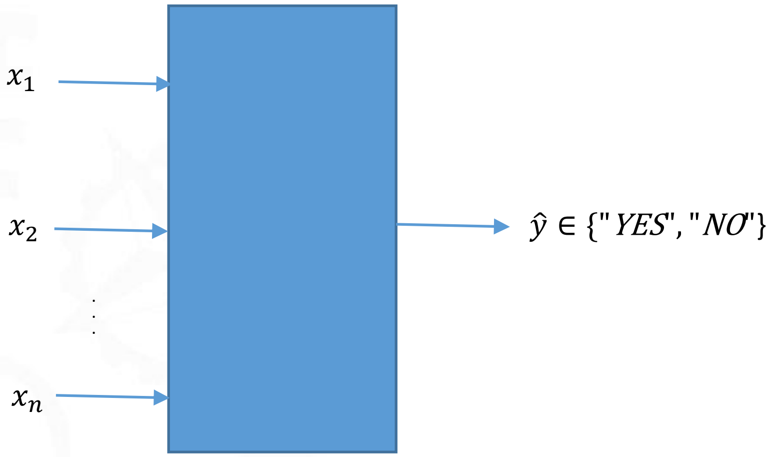
The classification problem

Measuring classification performance

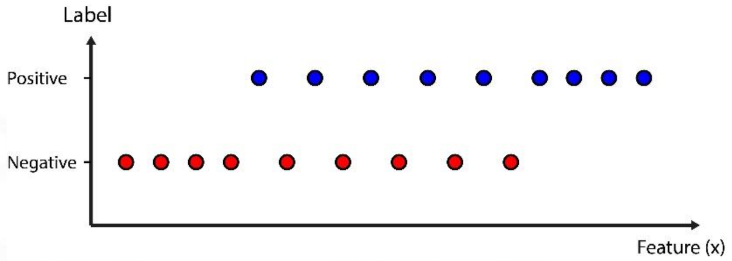
- Classification models can generate two types of predictions:
  - A predicted class in the form of a discrete category **(hard partition)**.
  - A continuous valued prediction which is usually in the form of a probability **(soft partition)**.

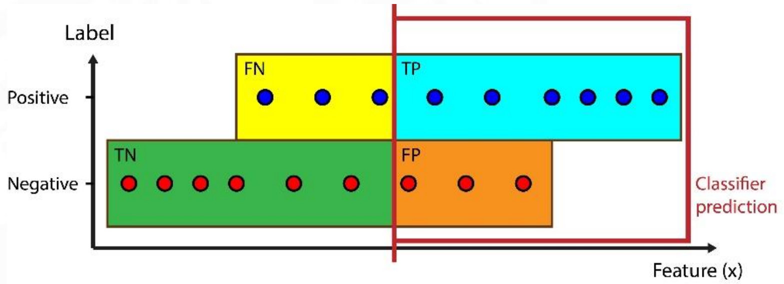
The classification problem
Problem statement

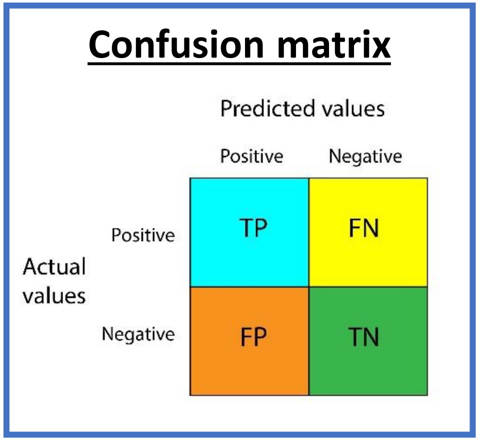
- **Hard partition approach:**



The classification problem
Measuring classification performance







Confusion matrix

		Predicted values	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

Source: https://towardsdatascience.com

The classification problem
Measuring classification performance

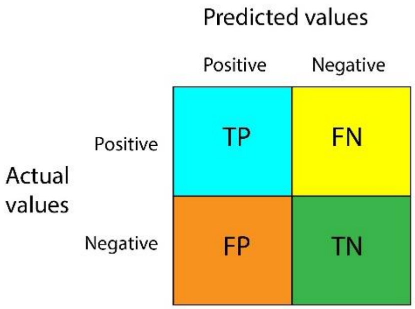
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Sensitivity \text{ } or \text{ } recall \text{ } or \text{ } TPR = \frac{TP}{TP + FN}$$

$$Specificity \text{ } or \text{ } TNR = \frac{TN}{TN + FP}$$

$$FPR = 1 - Specificity = 1 - TNR = \frac{FP}{TN + FP}$$



		Predicted values	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

Source: https://towardsdatascience.com

The classification problem
Measuring classification performance

- Example: credit scoring data = predict the quality of a customer's credit:

		Predicted values	
		Positive	Negative
Actual values	Positive	24	36
	Negative	10	130

- Overall performance $= \frac{24+130}{24+36+10+130} = 77\%$
- Sensitivity (*TPR*) $= \frac{24}{24+36} = 40\%$
- Specificity (*TNR*) $= \frac{130}{10+130} = 93\%$

This is likely due to the imbalance of the classes and a lack of a strong predictor for bad credit

The classification problem
Measuring classification performance

- We have to consider the natural frequencies of each class:

  If we had only 50 cases of class A and 450 of class B, the overall accuracy when predicting always class B will be:

$$\text{No−information rate } = \frac{450}{500} = 0.9$$

- Rather than calculate the overall accuracy and compare it to the no-information rate, other metrics can be used that take into account the class distributions of the training set samples: *Kappa statistic:*

$$Kappa = \frac{O - E}{1 - E}$$

$$O = \frac{TP + TN}{Total} \quad \text{is the overall accuracy}$$

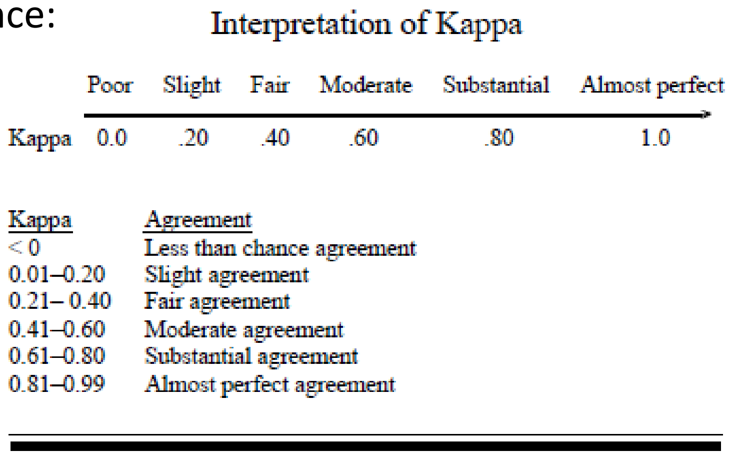
Predicted	Observed Event	Observed Nonevent	
Event	TP	FP	$P*$
Nonevent	FN	TN	$N*$
	P	N	

$$E = \frac{P^*}{Total} \times \frac{P}{Total} + \frac{N^*}{Total} \times \frac{N}{Total} \quad \text{is the probability of random or expected agreement}$$

The classification problem
Measuring classification performance

- We want to know how different the observed agreement is from the expected agreement.

- Kappa is a measure of this difference, standardized to lie on a -1 to 1 scale, where 1 is perfect agreement, 0 is exactly what would be expected by chance, and negative values indicate agreement less than chance:



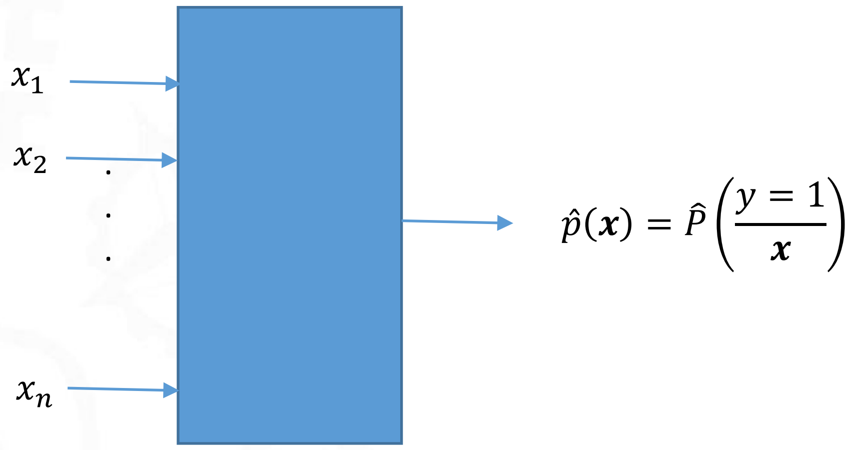
Interpretation of Kappa

	Poor	Slight	Fair	Moderate	Substantial	Almost perfect
Kappa	0.0	.20	.40	.60	.80	1.0

Kappa	Agreement
< 0	Less than chance agreement
0.01–0.20	Slight agreement
0.21– 0.40	Fair agreement
0.41–0.60	Moderate agreement
0.61–0.80	Substantial agreement
0.81–0.99	Almost perfect agreement

Measuring classification performance

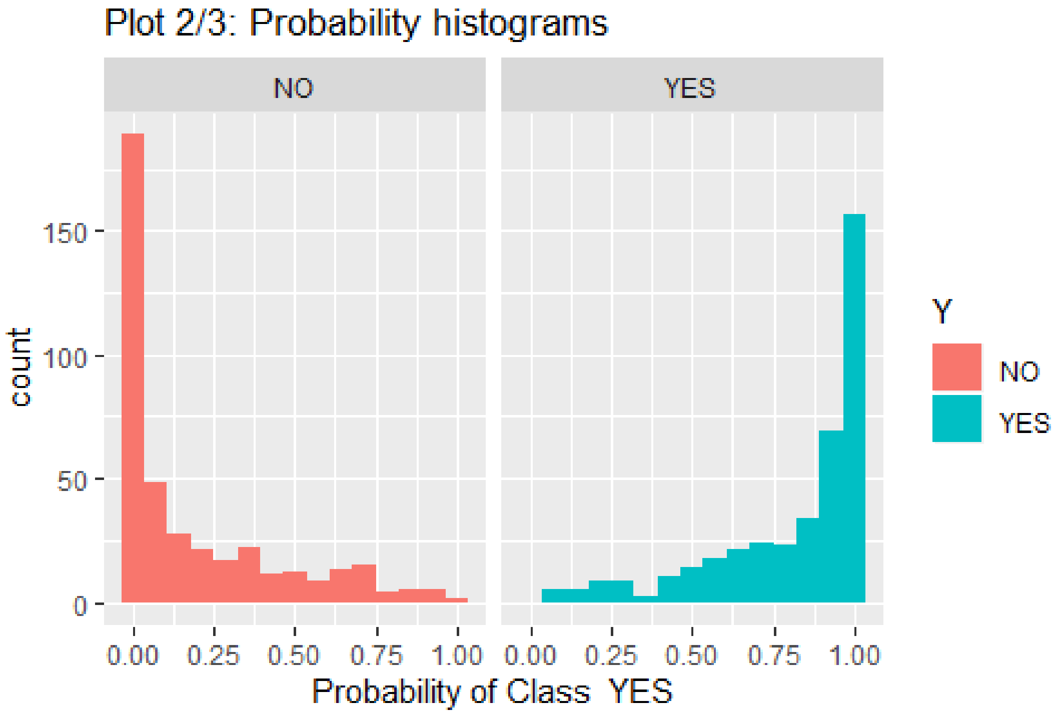
- **Soft partition approach:**



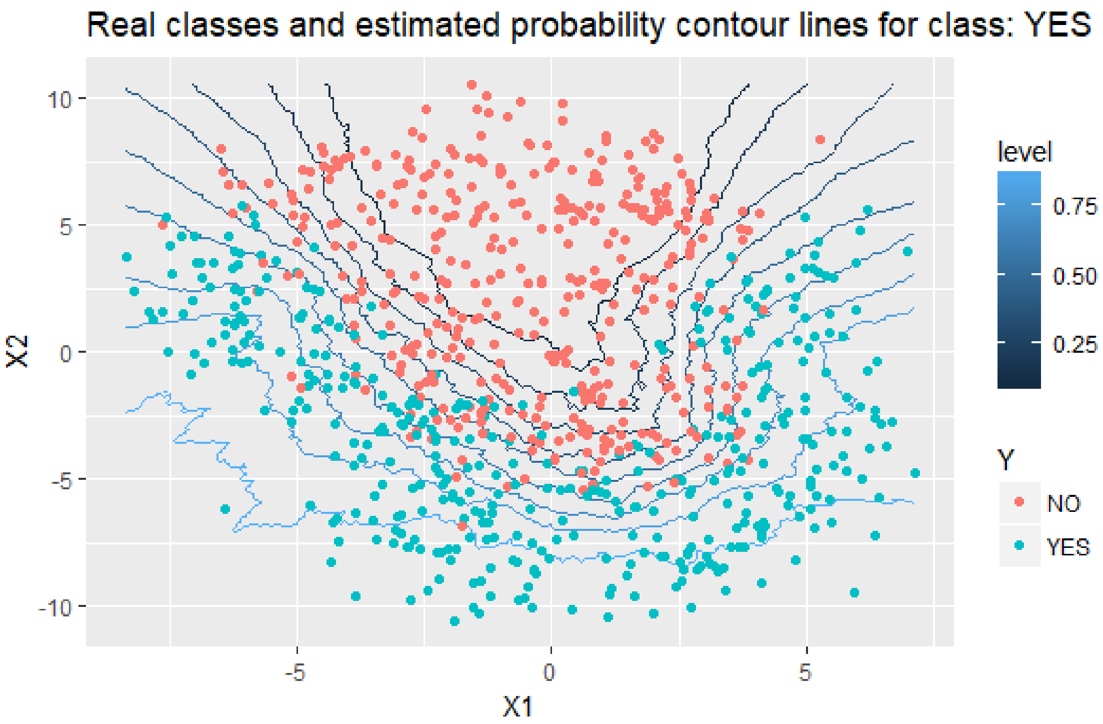
x_1
x_2
x_n
$\hat{p}(\boldsymbol{x}) = \hat{P}\left(\frac{y=1}{\boldsymbol{x}}\right)$

The classification problem
Measuring classification performance

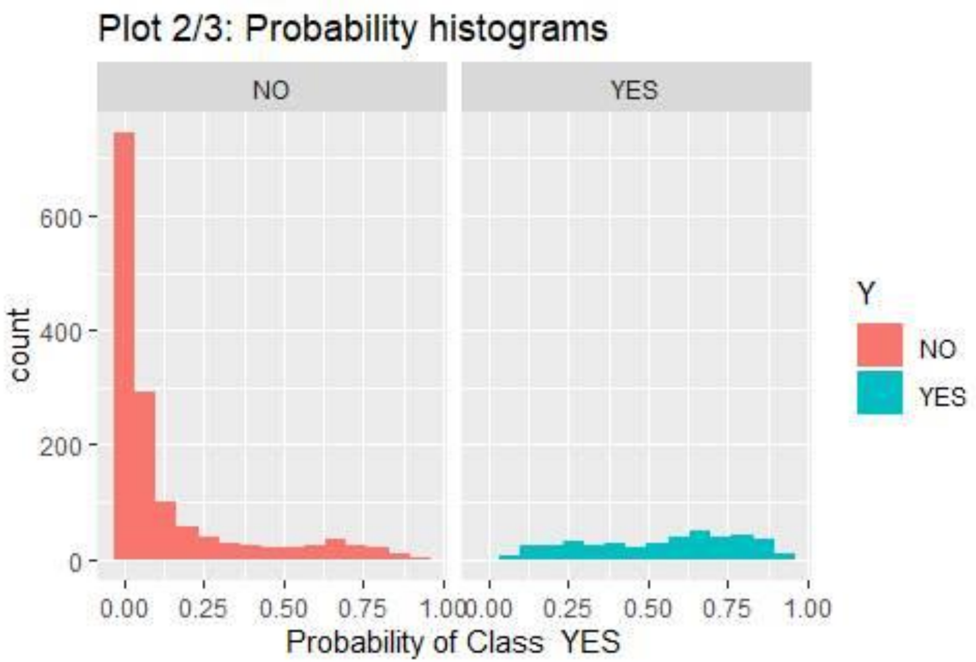
- Histograms of the estimated probability:



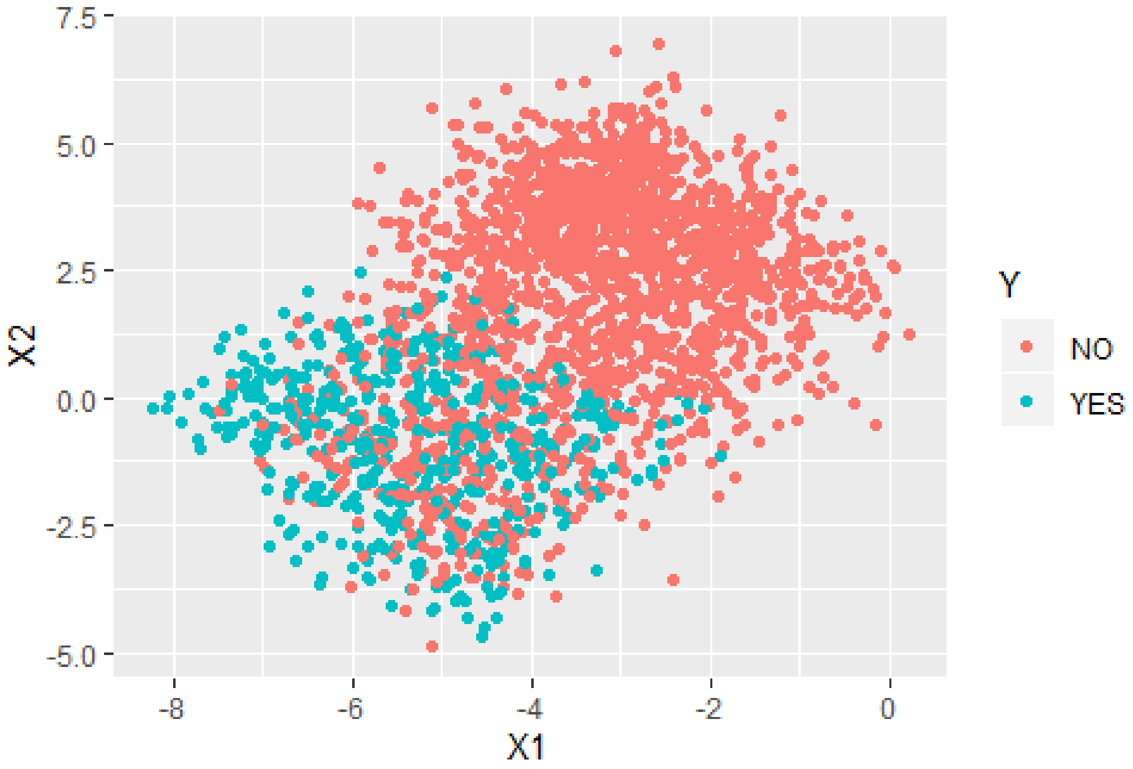
The classification problem
Measuring classification performance



The classification problem
Measuring classification performance

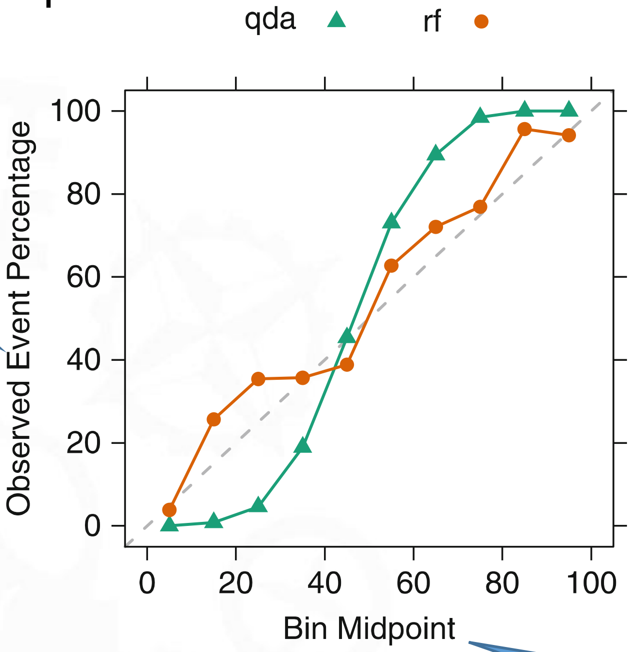


The classification problem
Measuring classification performance



The classification problem
Measuring classification performance

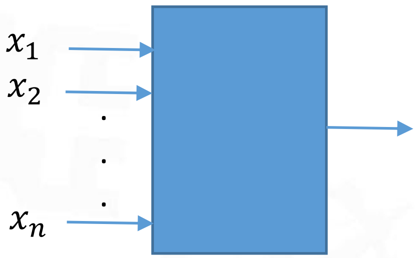
- Calibration plots:



The observed event rate of the test set

The estimated probabilities on the test set are binned in intervals of the form [0,10], (10,20], …

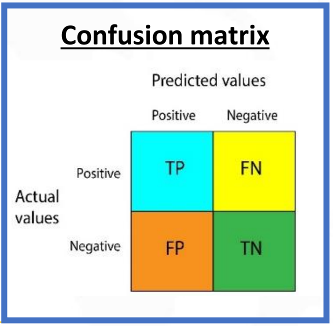
The classification problem
Measuring classification performance



$$\hat{p}(\boldsymbol{x}) = \hat{P}\left(\frac{y=1}{\boldsymbol{x}}\right)$$

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p}(x) < 0.5? \\ 1 \; if \, \hat{p}(x) \geq 0.5? \end{cases}$$

$$\hat{y} = \begin{cases} 0 \; if \; \hat{p}(x) < 0.3? \\ 1 \; if \, \hat{p}(x) \geq 0.3? \end{cases}$$



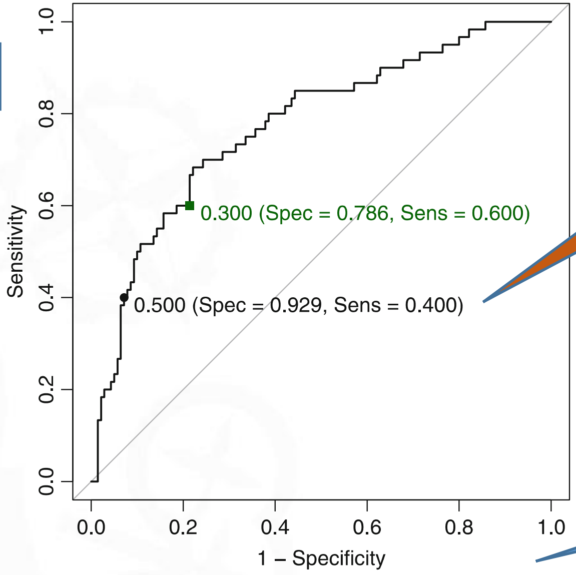
Confusion matrix

		Predicted values	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

Measuring classification performance

- **ROC curves:** The ROC curve is created by evaluating the class probabilities for the model across a continuum of thresholds. For each candidate threshold, the resulting true-positive rate (i.e., the sensitivity) and the false-positive rate (one minus the specificity) are plotted against each other.

True Positive rate (TP/(TP+FN))



By decreasing the threshold from 0.5 to 0.3, more samples will be classified as positive

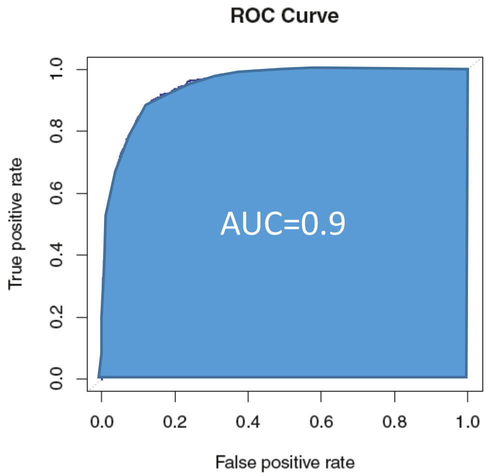
False Positive rate: (FP/(FP+TN))

		Predicted values	
		Positive	Negative
Actual values	Positive	TP	FN
	Negative	FP	TN

See: [What is a ROC Curve - How to Interpret ROC Curves - Displayr]

The classification problem
Measuring classification performance

- The area under the ROC curve (AUC) is a measure of the overall performance of a classifier.
- It is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance.
- An ideal ROC curve will hug the top left corner, so the larger the AUC, the better the classifier ($0 \leq \text{AUC} \leq 1$)
- We expect a classifier that performs no better than by chance to have an AUC of 0.5 on the test set.



2

Logistic regression

Logistic regression

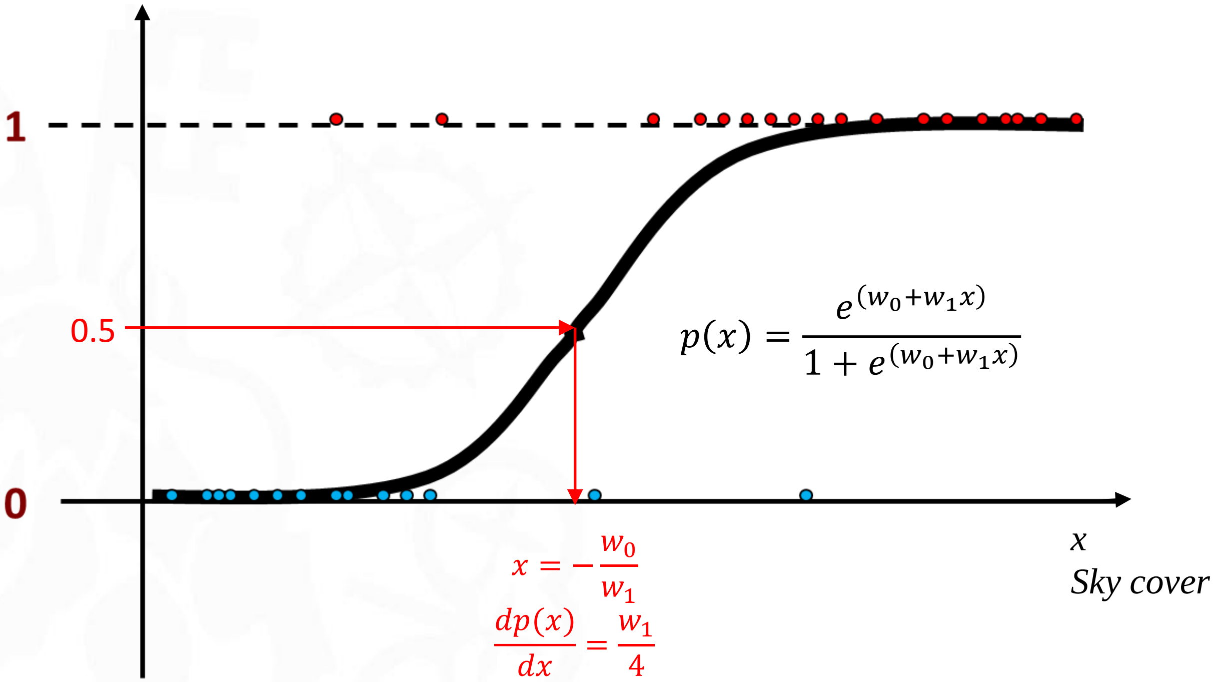
The logistic model

- Consider a classification problem where the response *Y* falls into one of two categories, 1 (rain) or 0 (no rain).

- Rather than modeling this response *Y* directly, logistic regression models the *probability* that *Y* belongs to a particular category conditioned on the value of the input variables:
$P(Y = 1 \mid X) = P(rain \mid sky\ cover)$

Logistic regression
The logistic model

$$p(x) = P(Y = 1 \mid x) = P(rain \mid sky\ cover)$$



$$p(x) = \frac{e^{(w_0 + w_1 x)}}{1 + e^{(w_0 + w_1 x)}}$$

$$x = -\frac{w_0}{w_1}$$

$$\frac{dp(x)}{dx} = \frac{w_1}{4}$$

Logistic regression
The logistic model

$$P(Y = 1 \mid x) = p(x) = \frac{e^{(w_0 + w_1 x)}}{1 + e^{(w_0 + w_1 x)}}$$

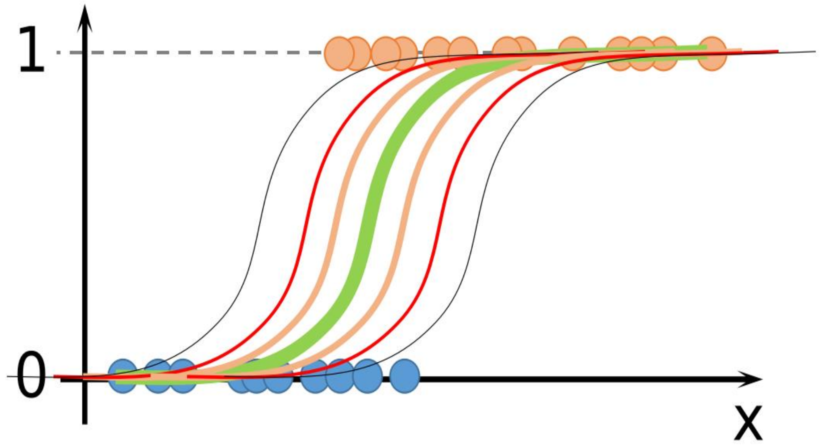
- After a bit of manipulation, we find that:

$$\frac{p(x)}{1 - p(x)} = e^{(w_0 + w_1 x)}$$

- The quantity $p(x)/[1-p(x)]$ is called the ***odds***, and can take on any value between 0 (very low probability) and ∞ (very high probability).
- Odds are traditionally used instead of probabilities in horse-racing, since they relate more naturally to the correct betting strategy.

Logistic regression

The logistic model



- The regression coefficients are estimated by maximizing the *likelihood function:*

$$l = \prod_{i:\, y_i=1} \hat{p}(\boldsymbol{x}_i) \prod_{i':\, y_{i'}=0} (1 - \hat{p}(\boldsymbol{x}_{i'}))$$

or minimizing the cross-entropy:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \left[y[i] log\left(\hat{p}(\boldsymbol{x})\right) + (1 - y[i]) log\left(1 - \hat{p}(\boldsymbol{x})\right) \right]$$

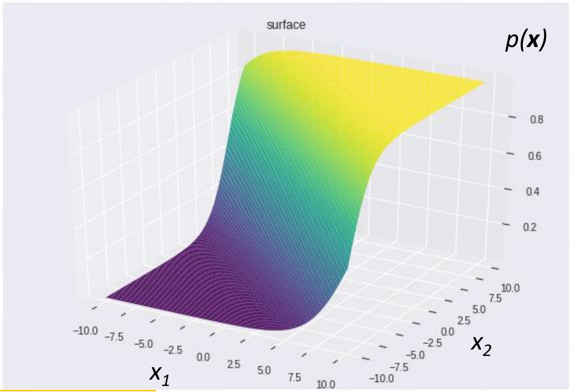
Logistic regression
Multiple logistic regression

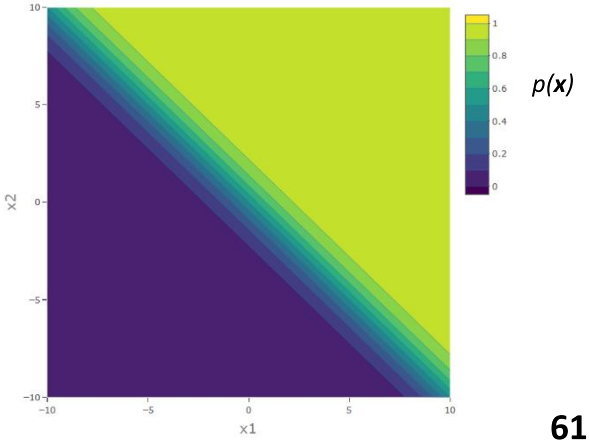
- If we consider the problem of predicting a binary response using multiple predictors, we can generalize the logistic model in the form:

$$ln\left(\frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})}\right) = w_0 + w_1x_1 + \ldots + w_nx_n$$

and therefore:

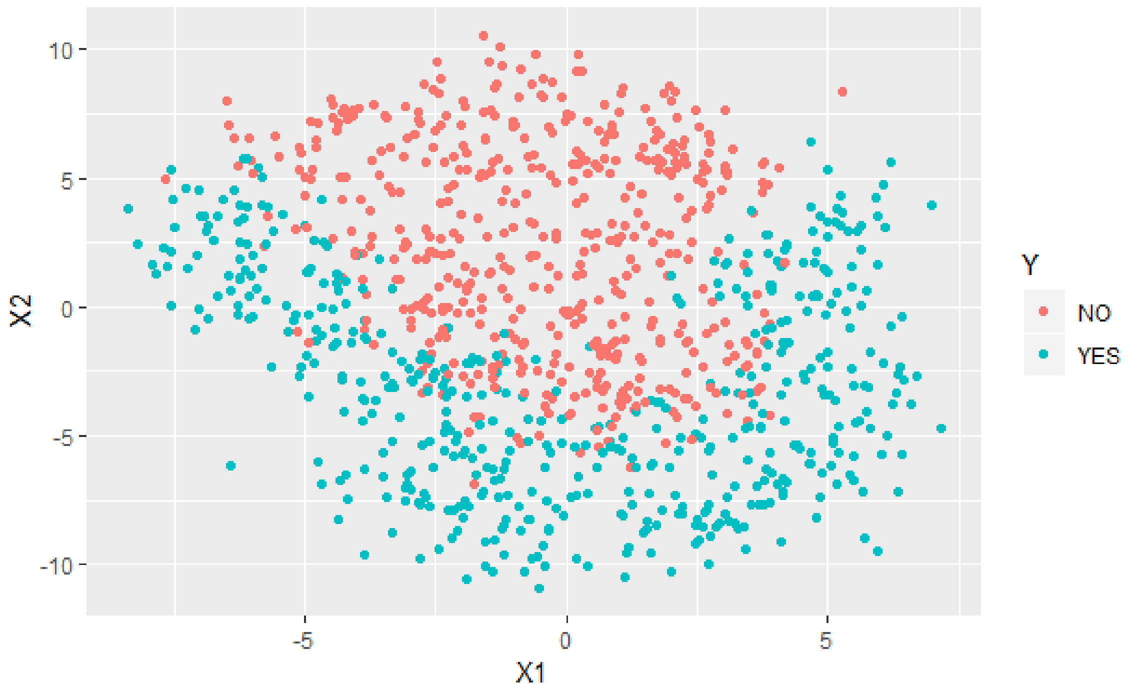
$$P(Y = 1 \mid \boldsymbol{x}) = p(\boldsymbol{x}) = \frac{e^{(w_0+w_1x_1+\ldots+w_nx_n)}}{1 + e^{(w_0+w_1x_1+\ldots+w_nx_n)}}$$





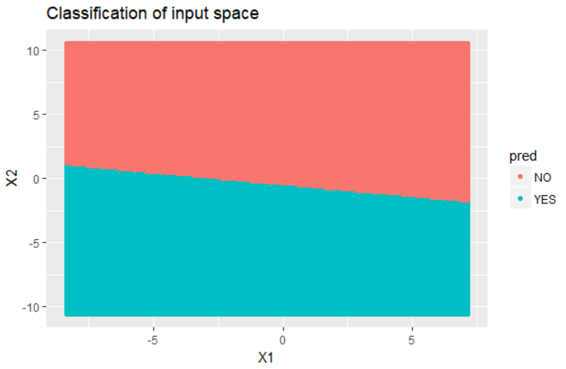
Logistic regression Example

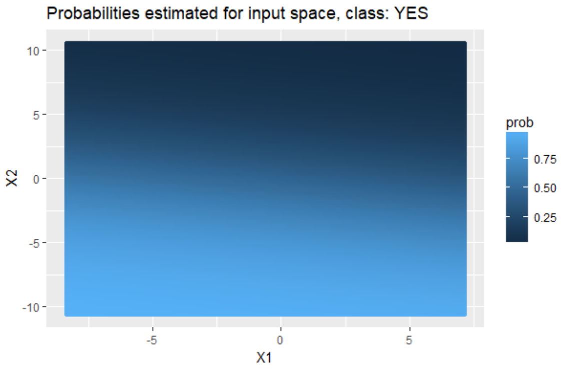
- Logistic Regression **training set**

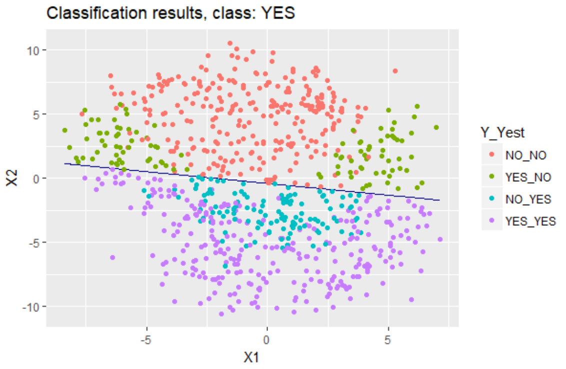


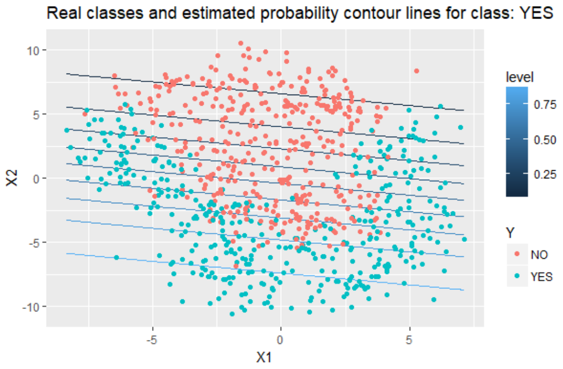
Logistic regression Example

- Logistic Regression **training results**









Logistic regression PROS vs CONS

PROS

1) Performs well when the dataset is linearly separable.
2) Less prone to over-fitting
3) Gives a measure of how relevant a predictor (coefficient size) is and direction of association (positive or negative)
4) Easy to implement, interpret and very efficient to train.

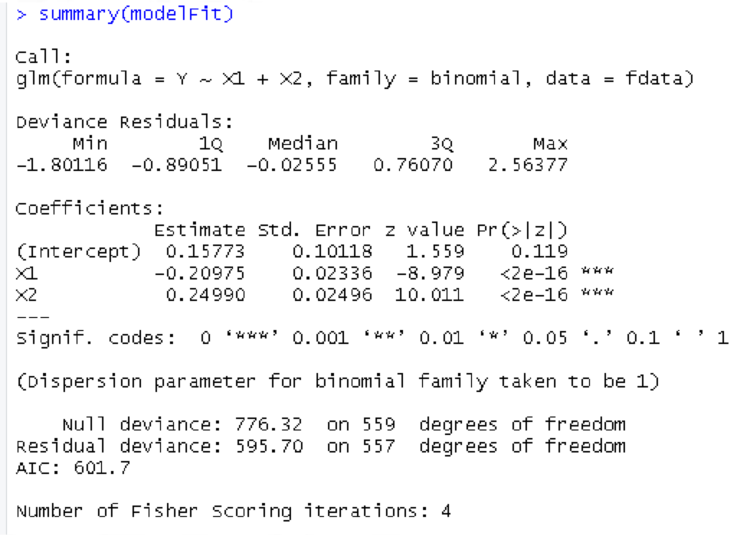
CONS

1) The assumption of linearity

Logistic regression
R implementation

- Function `glm()` in base R is used for fitting general linear models.
- Setting the input parameter "family" to binomial trains a logistic model.

```r
#Training
modelFit <- glm(Y ~ X1 + X2,   #Formula specifying the output (Y) and inputs (X1,X2)
           data = fdata_train, #Select training dataset
           family = binomial)  #'binomial' is used for logistic regression
```



```
> summary(modelFit)

Call:
glm(formula = Y ~ X1 + X2, family = binomial, data = fdata)

Deviance Residuals: 
     Min        1Q    Median        3Q       Max  
-1.80116  -0.89051  -0.02555   0.76070   2.56377  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.15773    0.10118   1.559    0.119    
X1          -0.20975    0.02336  -8.979   <2e-16 ***
X2           0.24990    0.02496  10.011   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 776.32  on 559  degrees of freedom
Residual deviance: 595.70  on 557  degrees of freedom
AIC: 601.7

Number of Fisher Scoring iterations: 4
```

Logistic regression
R implementation

- Predict function is used to compute estimates:

```
#Prediction
ProbEst <- predict(modelFit,          #Fitted model
                   newdata = fdata_val, #Predict validation set
                   type = "response")   #glm does not predict the class. It gives the
                                        #probability of the event
```

- Other option is the `lrm()` function from *rms* package.
- The function `cv.glm()` from *boot* package implements cross-validation features.

Logistic regression
R implementation - Caret

- With *caret*, the `train()` function is used and the training sentence is simplified.
- There is no need to specify `family = binomial` because it is given in the metric.

```
> set.seed(476)   #Important for replication
> LogReg.fit <- train( fdata_train[,c(“X1”,”X2”)],    #Input variables
                     y = fdata_train$Y,               #Output variable
                     method = "glm",                  #Type of model
                     metric = "ROC",                  #Metric to summarize performance
                     trControl = ctrl)                #Metric and resampling options

#Print information about the training process
LogReg.fit

#Print information about final fitted model
Summary(LogReg.fit)

#For computing estimates, two sentences are posible
# predict probabilities
> fdata_val_eval$prob = predict(LogReg.fit, type="prob" , newdata = fdata_val)

# predict classes
> fdata_val_eval$pred = predict(LogReg.fit, type="raw" , newdata = fdata_val)
```

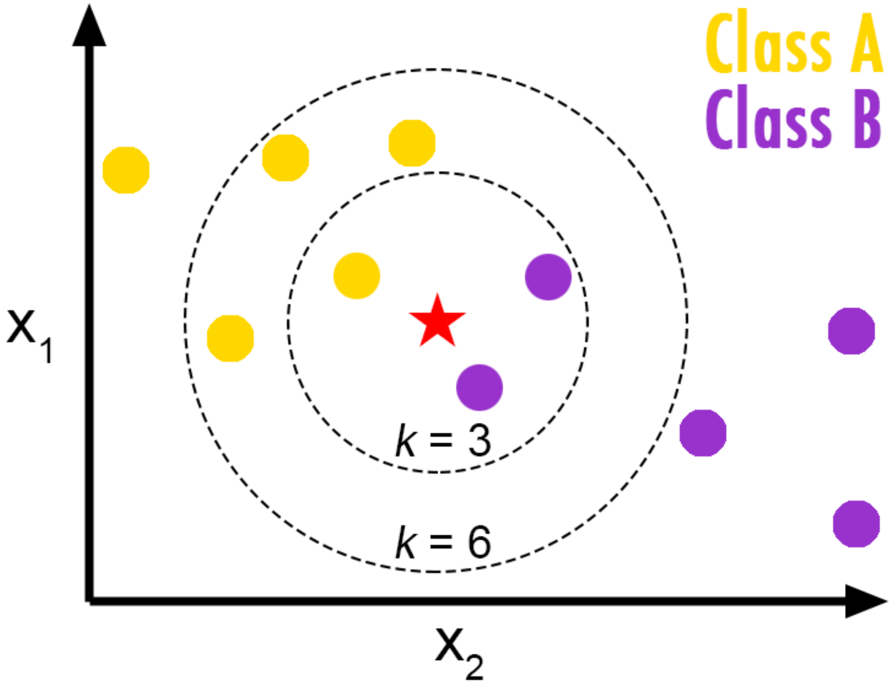
3

K-Nearest Neighbors

The kNN algorithm

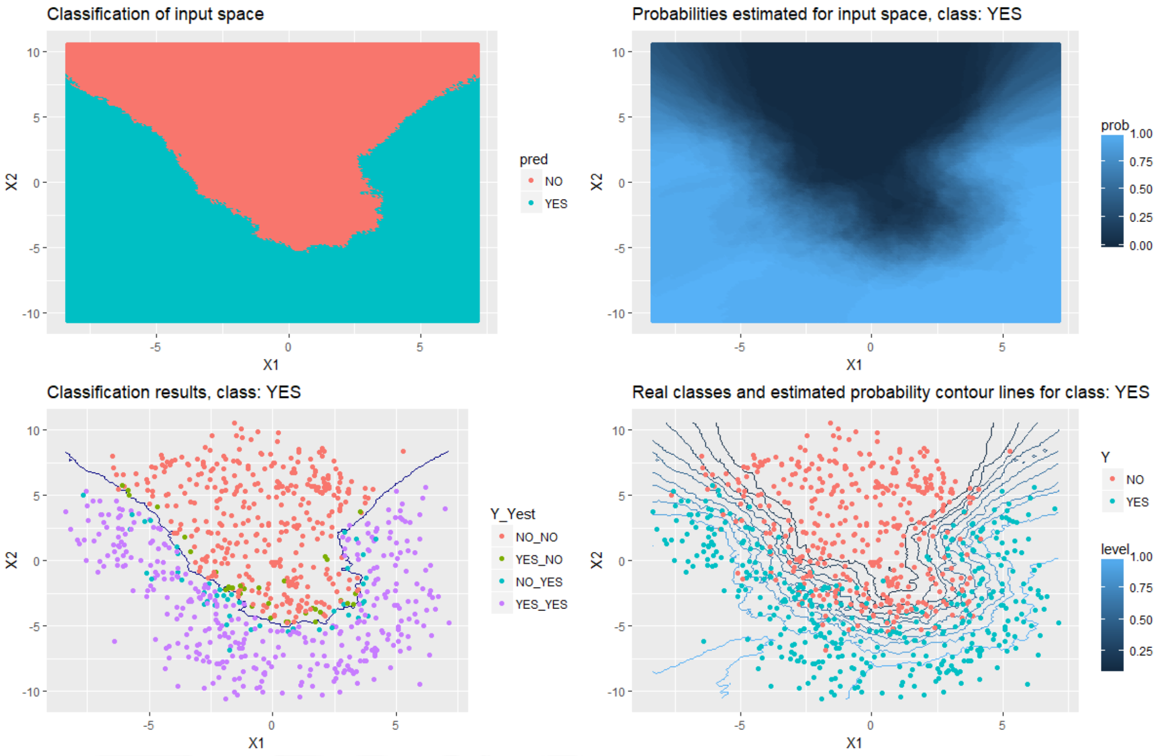
- The kNN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples.
- The letter k is a variable term implying that any number of nearest neighbors could be used.
- After choosing k, the algorithm requires a training dataset made up of examples that have been classified into several categories, as labeled by a nominal variable.
- Then, for each unlabeled record in the test dataset, k-NN identifies k records in the training data that are the "nearest" in similarity.
- The unlabeled test instance is assigned the class of the majority of the k nearest neighbors.

K-Nearest Neighbors
The kNN algorithm



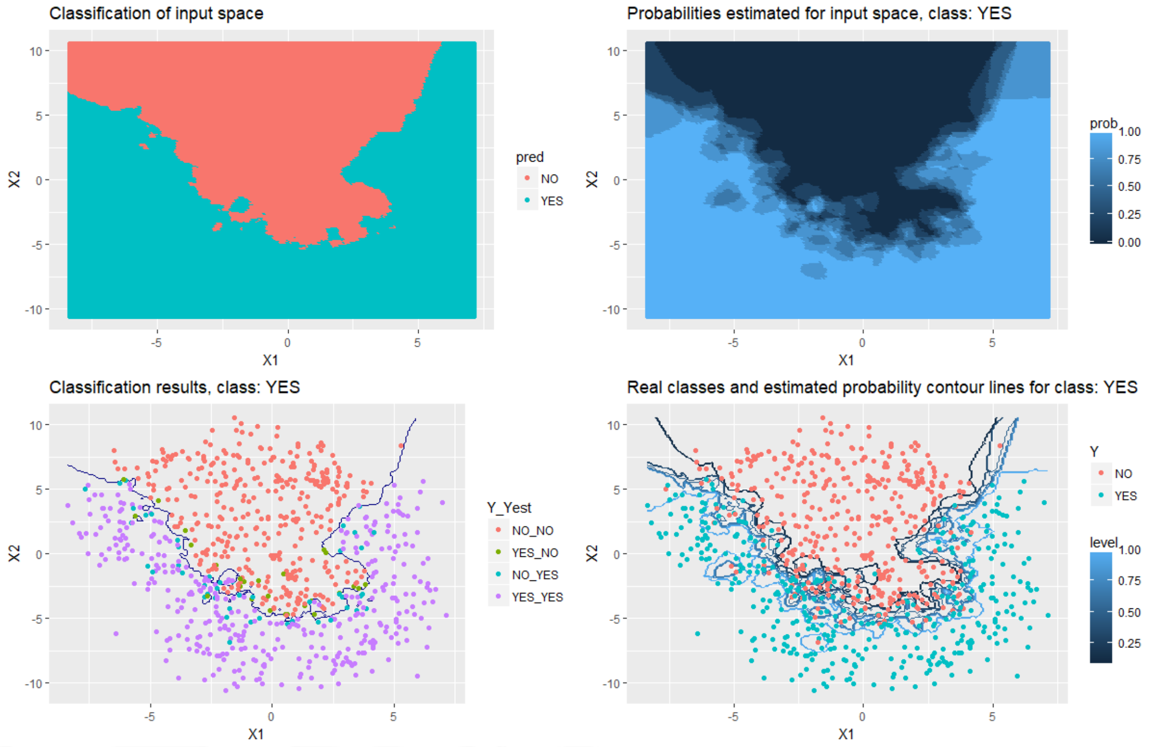
K-Nearest Neighbors
Parameter tuning example

- KNN train results **k=40.**



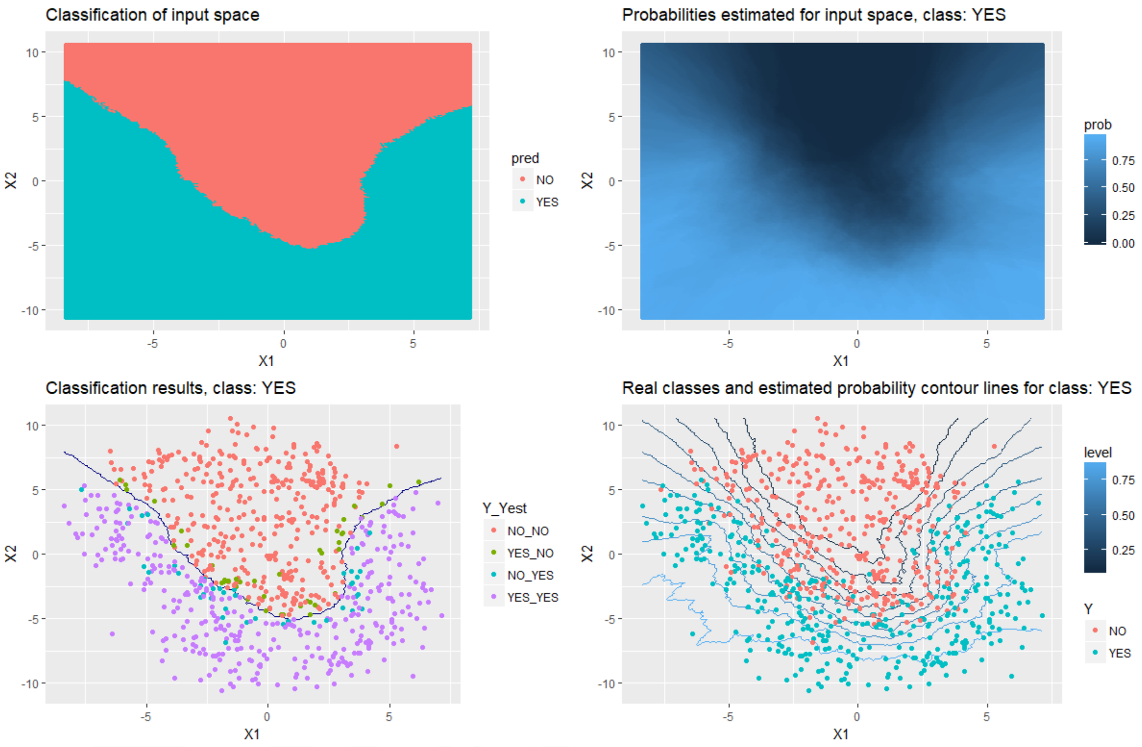
K-Nearest Neighbors
Parameter tuning example

- KNN train results **k=5**.



K-Nearest Neighbors
Parameter tuning example

- KNN train results **k=100**.



K-Nearest Neighbors PROS vs CONS

PROS

1) Simplicity (simple idea+1 **hyperparameter**)
2) No assumptions (Non-parametric)
3) No Training Step (simply tags new data based on historical data)
4) Evolves (adapts as we collect new data)
5) Easy to implement even for multiclass models
6) Can be used both for Classification and Regression

CONS

1) Slow (the curse of dimensionality)
2) K-NN needs homogeneous features
3) Optimal number of neighbors?
4) k-NN doesn't perform well on imbalanced data
5) Very sensitive to outliers as it simply choose the neighbors based on distance criteria.
6) NA treatment

K-Nearest Neighbors
R implementation: basic and caret

- Function `knn()` from *class* library.
- It does not train a model, but directly computes the estimation based on the training data.

```
library(class)
set.seed (1) #For reproducibility
Ypred_knn=knn(fdata_train[,c("X1",c("X2"))],  #Predictors associated with the training data
              fdata_val[,c("X1",c("X2"))],    #Predictors associated with the data to make predictions
              fdata_train$Y,                  #class labels for the training observations,
              k=3)                            #number of nearest neighbors to be used
```

- Using *caret*, the ususal model stucture is kept.

```
#Training the model
> modelfit = train(fdata_train[,c("X1","X2")], #Input variables
                 y = fdata_train$Y, #Output variable
                 method = "knn", #knn model
                 preProcess = c("center","scale"), #pre-processing if desired
                 tuneGrid = data.frame(.k = 3), #number of nearest neighbors to be used
                 trControl = ctrl, #Resampling settings
                 metric = "ROC") #Summary metrics

#Predictions for new data. Probabilities and classes
> Ypred_knn_prob = predict(modelfit, type="prob" , newdata = fdata_val)
> Ypred_knn_pred = predict(modelfit, type="raw" , newdata = fdata_val)
```

K-Nearest Neighbors
Parameter tuning using caret

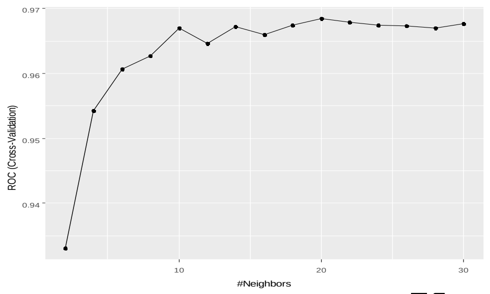
- One of the advantages of using *caret* is the optimum selection of tuning parameters.
- See `getModelInfo()` or caret web page for a list of models and associated parameters.

```
#Set trainControl with a resampling method. Example: k-fold cross-val
> ctrl = trainControl(method = “cv", number = 10,
                  summaryFunction = twoClassSummary, classProbs = TRUE,
                  savePredictions = TRUE)

#Create a grid of parameters for evaluation
#Sequence from 2 to 30 in increments of 2
> paramsGrid = data.frame(k= seq(2,30,2))

#Train model
> set.seed(50) #Important for replication
> modelfit = train(fdata_train[,c("X1","X2")], y = fdata_train$Y,
                   method = "knn", preProcess = c("center","scale"),
                   tuneGrid = paramsGrid, #use grid
                   trControl = ctrl,      #Resampling settings
                   metric = "ROC")
```

```
> modelfit
k-Nearest Neighbors

560 samples
  2 predictor
  2 classes: 'A', 'B'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 504, 504, 504, 504, 504, 504, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   2  0.9330995  0.8785714  0.8892857
   4  0.9542730  0.9107143  0.8964286
   6  0.9607143  0.9142857  0.9178571
   8  0.9626913  0.9214286  0.9035714
  10  0.9669643  0.9321429  0.9107143
  12  0.9646046  0.9357143  0.9000000
  14  0.9672194  0.9321429  0.8821429
  16  0.9659439  0.9321429  0.8892857
  18  0.9674107  0.9285714  0.8892857
  20  0.9684949  0.9392857  0.8821429
  22  0.9678571  0.9321429  0.8785714
  24  0.9674745  0.9428571  0.8678571
  26  0.9673469  0.9428571  0.8678571
  28  0.9670281  0.9464286  0.8642857
  30  0.9676658  0.9392857  0.8500000

ROC was used to select the optimal model using  the largest value.
The final value used for the model was k = 20.
```

- Use `plot` or `ggplot` for plotting the resampling profile →
- For models with more tan one tuning parameter, the expand.grid function should be used.

```
#Example of two parameter grid with names alpha and lambda
> paramsGrid <- expand.grid(alpha = c(0.1, 0.5, 0.9),
                         lambda = c(0.001, 0.01))
```



comillas.edu

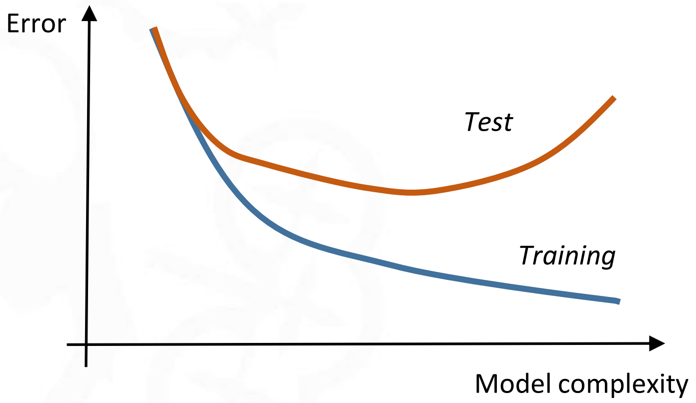
4

Validation techniques

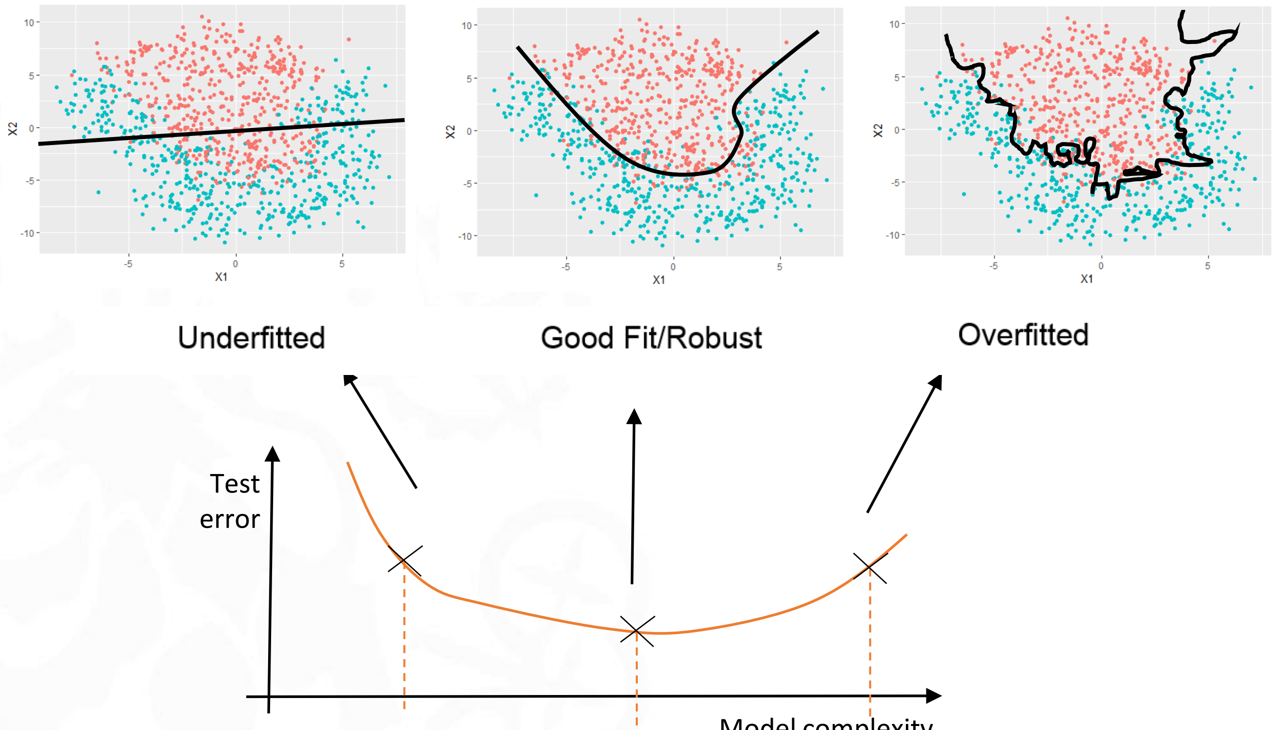
The classification problem
Training and test errors

- The ***test error*** is the average error that results from using a statistical learning method to predict the response on a new observation == measure of the ***generalization capability*** of the model.
- The ***training error*** estimated on the training data, often called ***resubstitution*** *error*, is in general very optimistic.
- This phenomenon is called ***overtraining***, and its effect is most pronounced for complex, flexible learners (as Neural Networks or complex Decision Trees):

Training and Test errors



The classification problem
Training and test errors



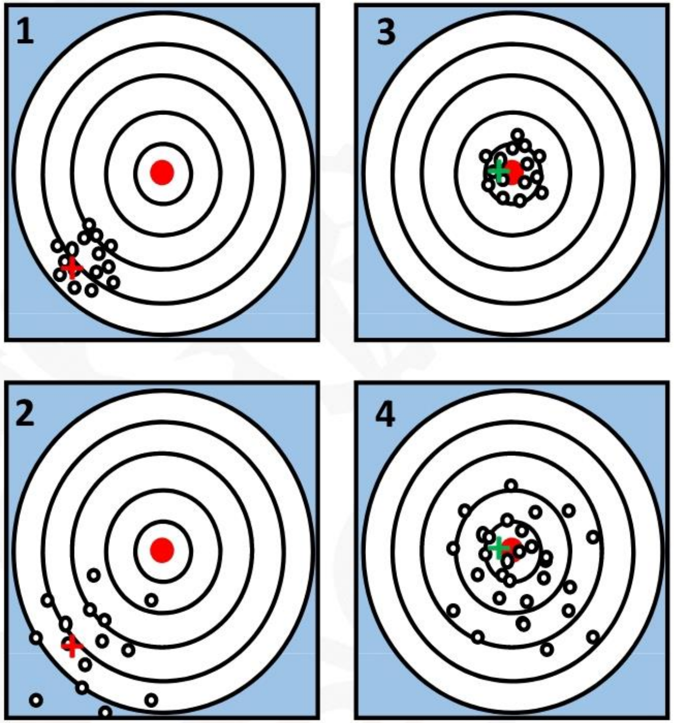
The classification problem
The bias-variance trade-off

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon)$$

Expected test error = *Variance* + *Squared Bias* + *Irreducible error*

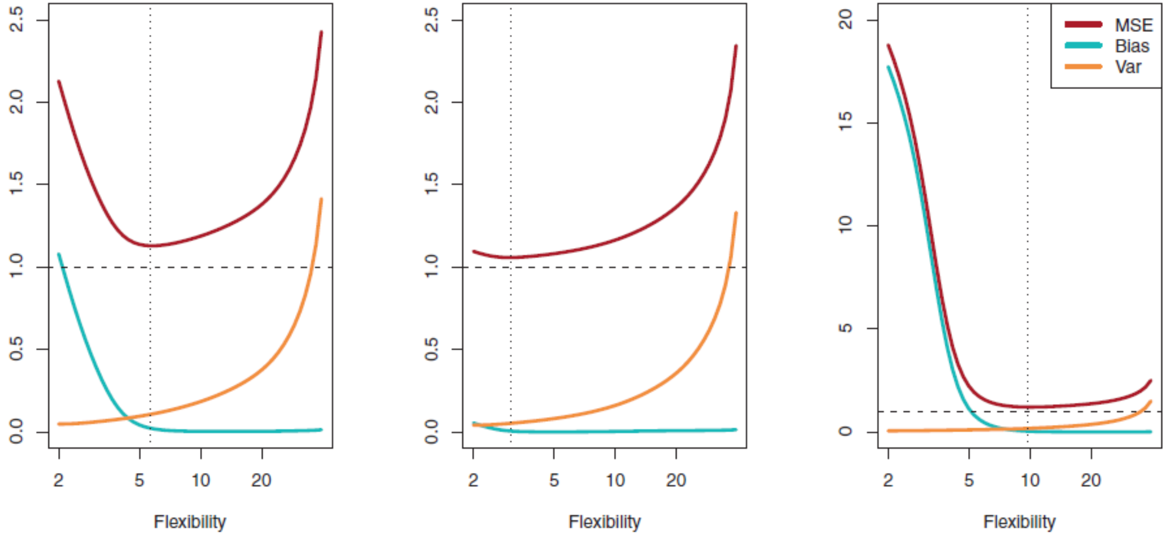
- The first term refers to the ***average test MSE*** that we would obtain if we repeatedly estimated f using a large number of training sets and tested each at x_0.
- ***Variance***: amount by which $\hat{f}$ would change if we estimated it using a different training data set: if a method has high variance then small changes in the training data can result in large changes in $\hat{f}$.
- ***Bias*** refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
- As a general rule, as we use **more flexible** methods, the **variance will increase** and the **bias will decrease**.

The classification problem
The bias-variance trade-off



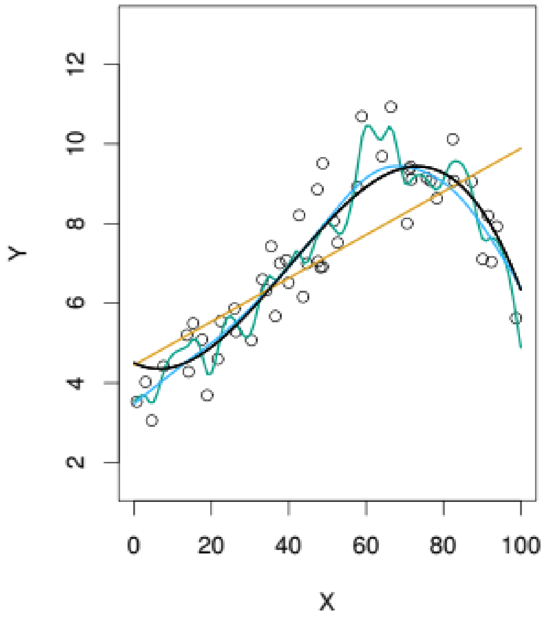
The classification problem
The bias-variance trade-off

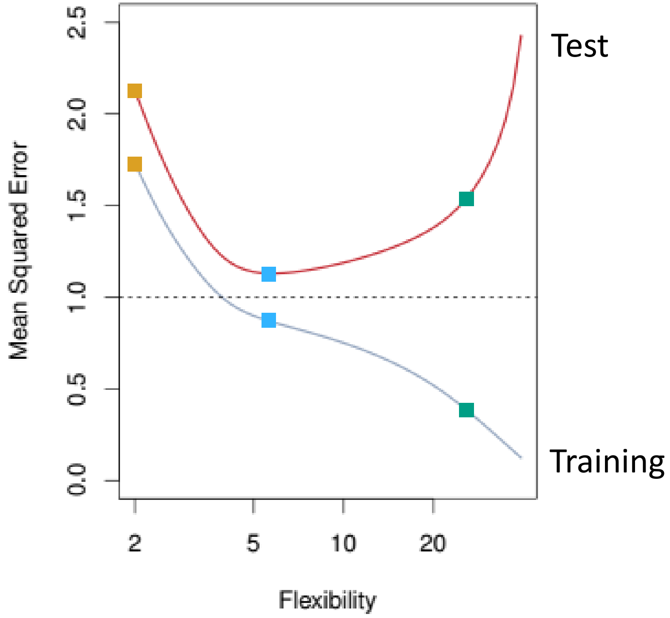
- Which one corresponds to a highly non-linear problem?



Source: (James et al., 2013)

The classification problem
The bias-variance trade-off



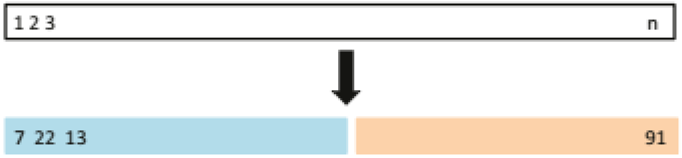


Source: (James et al., 2021)

The classification problem
The validation set approach

- **Objective**: estimate the test error rate

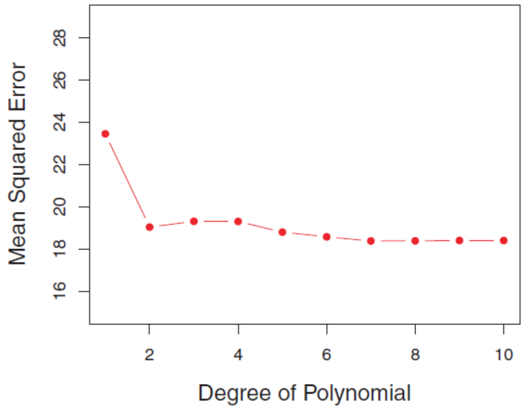
- **Solution**:
  1. Randomly divide the available set of observations into two parts, a ***training set*** (also called in-sample) and a ***validation set*** (out-of-sample or hold-out set).
  2. Fit the model on the training set
  3. Use the fitted model to predict the responses for the observations in the validation set. The resulting validation set error rate provides an estimate of the test error rate

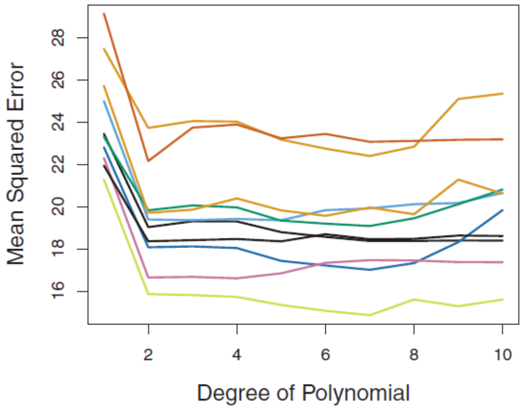


Source: (James et al., 2021)

The classification problem
The validation set approach

- How do we decide where to split the dataset into training and validation?





Left: *Validation error estimates for a single split into training and validation data sets.* Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Source: (James et al., 2021)

The validation set approach

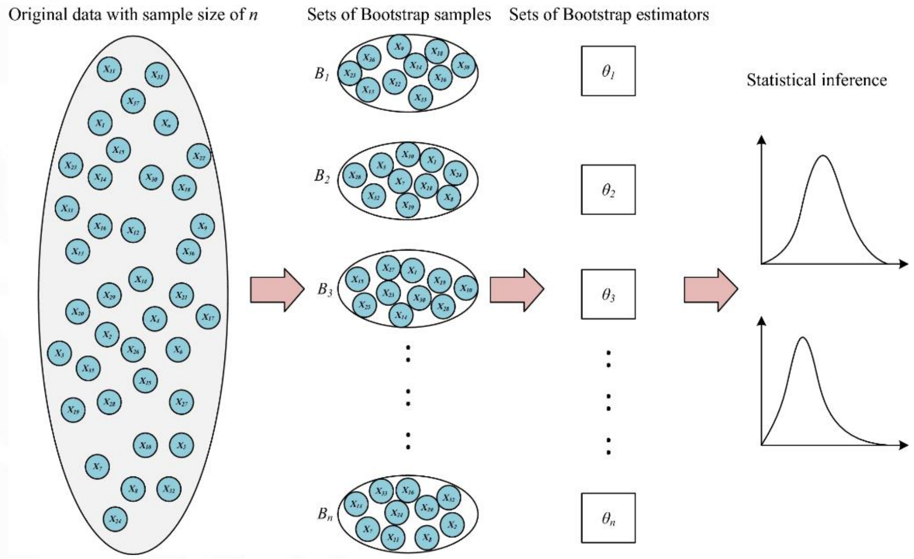
- The validation set approach has two main drawbacks:

    1. The validation estimate of the test error rate can be ***highly variable***, depending on precisely which observations are included in the training set and which observations are included in the validation set.

    2. Only a subset of the observations are used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to ***overestimate*** the test error rate for the model fit on the entire data set.

The classification problem
Resampling

- Resampling methods involve ***repeatedly drawing samples*** from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model.


Original data with sample size of n
Sets of Bootstrap samples
Sets of Bootstrap estimators
Statistical inference
B_1
B_2
B_3
B_n
θ_1
θ_2
θ_3
θ_n

From: Fatigue crack growth analysis using Bootstrap S-version finite element model

The classification problem
Resampling

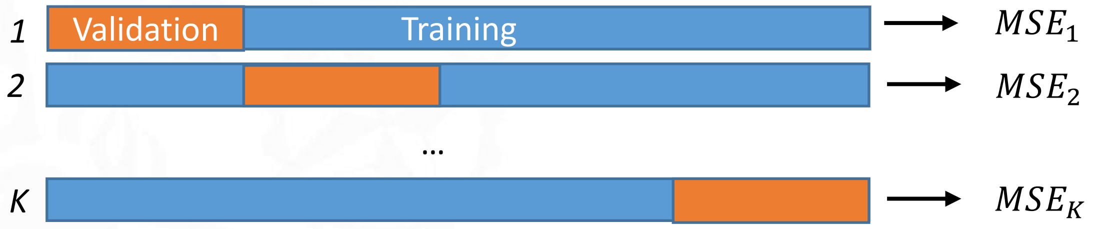
- Resampling effectively ***increases the amount of data*** without incurring the full cost of data simulation or collection.

- This trick comes at a price: datasets obtained by resampling are ***not independent***. This lack of independence can affect the quality of estimates.

- Even if the resulting datasets are not independent, theory and application experience demonstrate that resampling does produce ***good estimates*** of various statistical quantities, including the estimation ***of the predictive power*** by a classification algorithm.

The classification problem
Cross-Validation

- Cross-validation works by splitting data into *K* disjoint subsets, or *folds*.
- Use *1-1/K* of data for training and hold out *1/K* of data for validation.
- Repeat this step *K* times, that is, use every observation once for validation and (*K-1)* times for training.



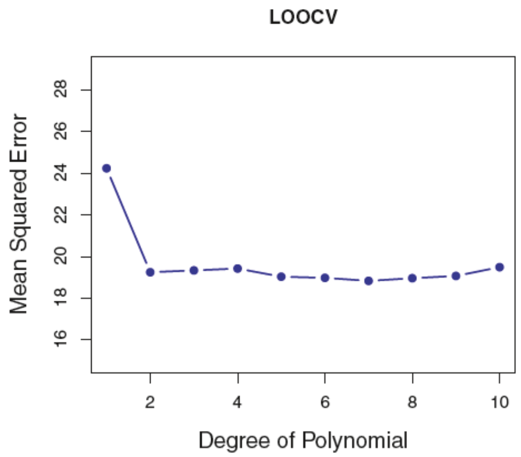
- This process results in *k* estimates of the test error, MSE_1, $\text{MSE}_2, \ldots,$ MSE_k. The *k*-fold CV estimate is computed by averaging these values:

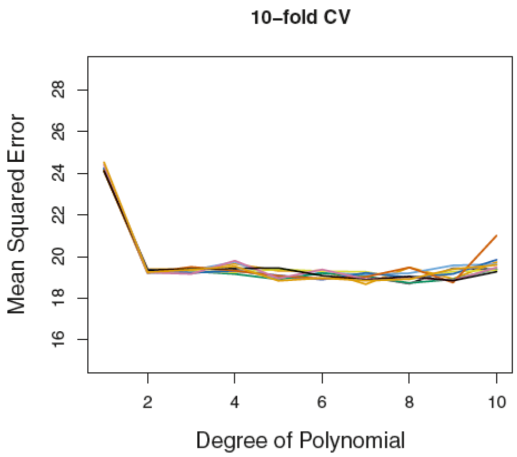
$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i .$$

The classification problem
Cross-validation

- The number of folds can vary from 2 to *N*, where *N* is the number of available observations.
- *K*=*N* is called leave-one-out cross-validation (LOOCV).
- Two-fold and leave-one-out schemes represent the two extremes, each with its advantages and disadvantages.
- The most popular choice for *K* in supervised learning is 10.

The classification problem
Cross-validation

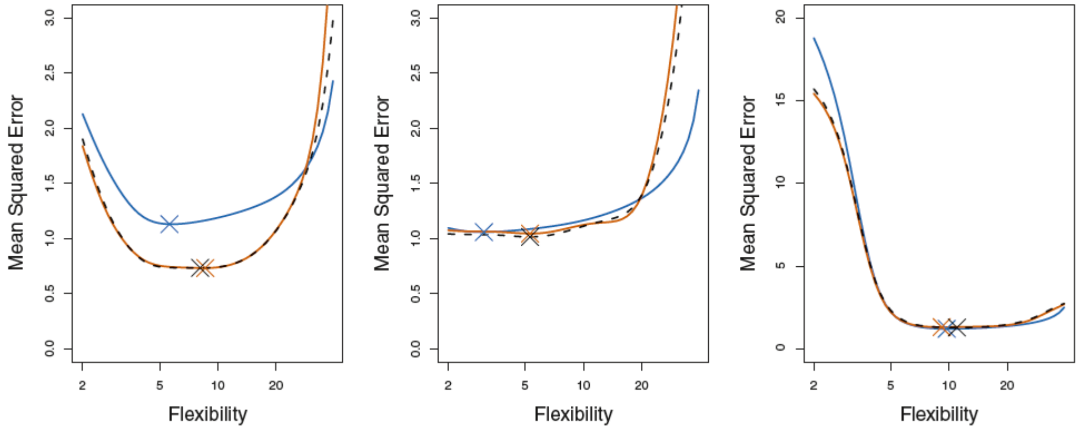




Left: The LOOCV error curve. *Right*: 10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.

Source: (James et al., 2013)

The classification problem
Cross-validation



The true test MSE is shown in blue, the LOOCV estimate is shown as a black dashed line, and the 10-fold CV estimate is shown in orange. The crosses indicate the minimum of each of the MSE curves.

Despite the fact that they sometimes underestimate the true test MSE, all of the CV curves come close to identifying the correct level of flexibility.

Source: (James et al., 2013)

5

Bibliography

Bibliography

- G. James, D. Witten, T. Hastie & R. Tibshirani (2021). *An Introduction to Statistical Learning with Applications in R*. Second Edition. Springer. See https://www.statlearning.com/)
- M. Kuhn & K. Johnson (2013). *Applied Predictive Modeling*. Springer
- T. Hastie, R. Tibshirani & J. Friedman (2009). *The Elements of Statistical Learning. Data Mining, Inference and Prediction*. 2nd Ed. Springer.

- E. Alpaydin (2014). *Introduction to Machine Learning.* 3rd Ed. MIT Press
- S. Marsland (2015), *Machine Learning: An Algorithmic Perspective*, 2nd Ed., Chapman & Hall/Crc Machine Learning & Pattern Recognition.
- T. Mitchell (1997). *Machine Learning.* McGraw-Hill.

- R. Duda, P. Hart & D. Stork (2000). *Pattern Classification*. 2nd Ed. Wiley-Interscience.
- C. Bishop (2007). *Pattern Recognition and Machine Learning*. Springer.
- S. Haykin (1999). *Neural Networks. A comprehensive foundation.* 2nd Ed. Pearson.
- W. Wei (2006). *Time Series Analysis. Univariate and Multivariate Methods.* 2nd Ed. Addison-Wesley.

Alberto Aguilera 23, E-28015 Madrid - Tel: +34 91 542 2800 - http://www.iit.comillas.edu