MTAT.03.319

Business Data Analytics

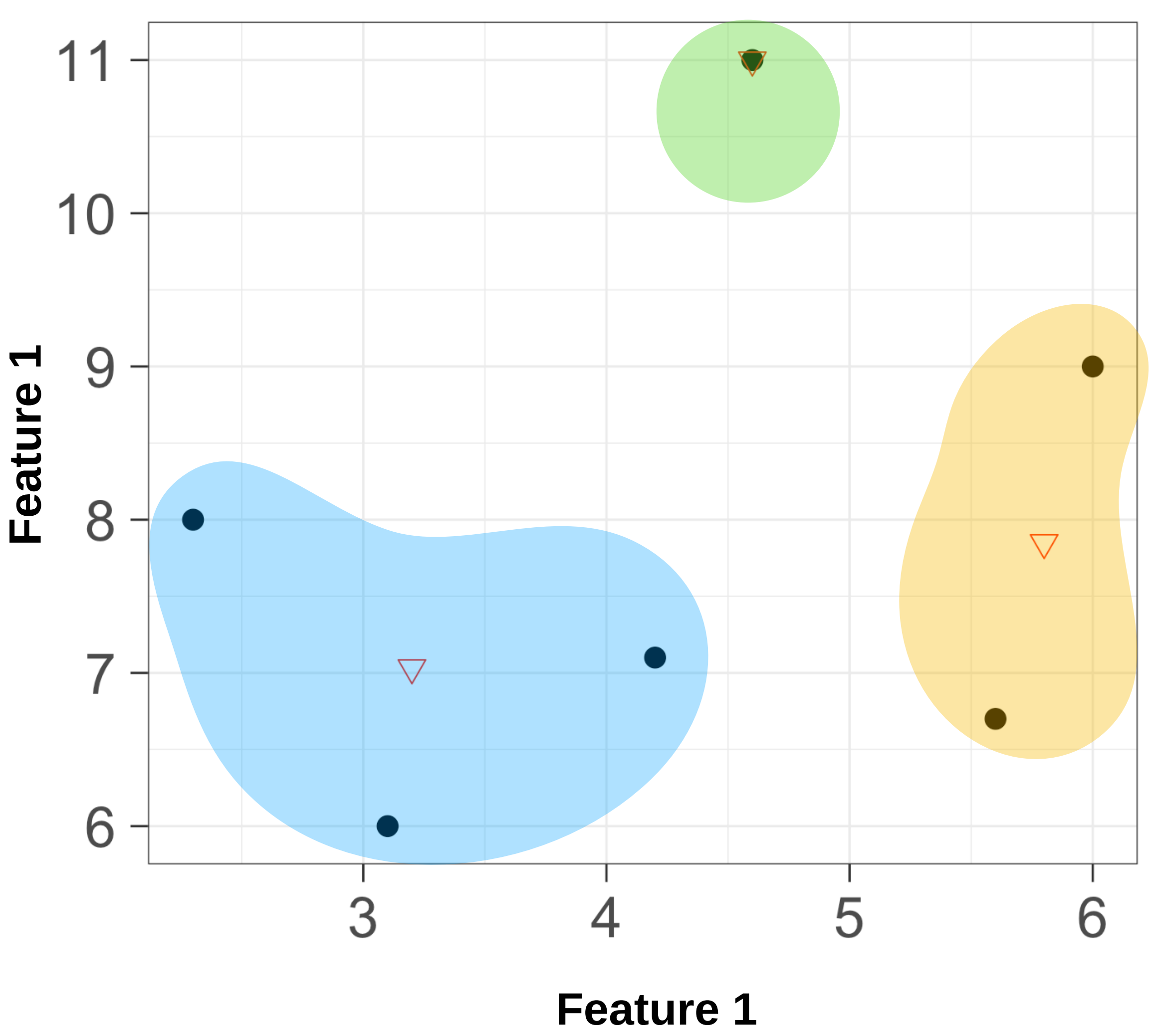
Lecture 5: Customer Lifecycle Management – Classification Problems

Rajesh Sharma
https://css.cs.ut.ee/

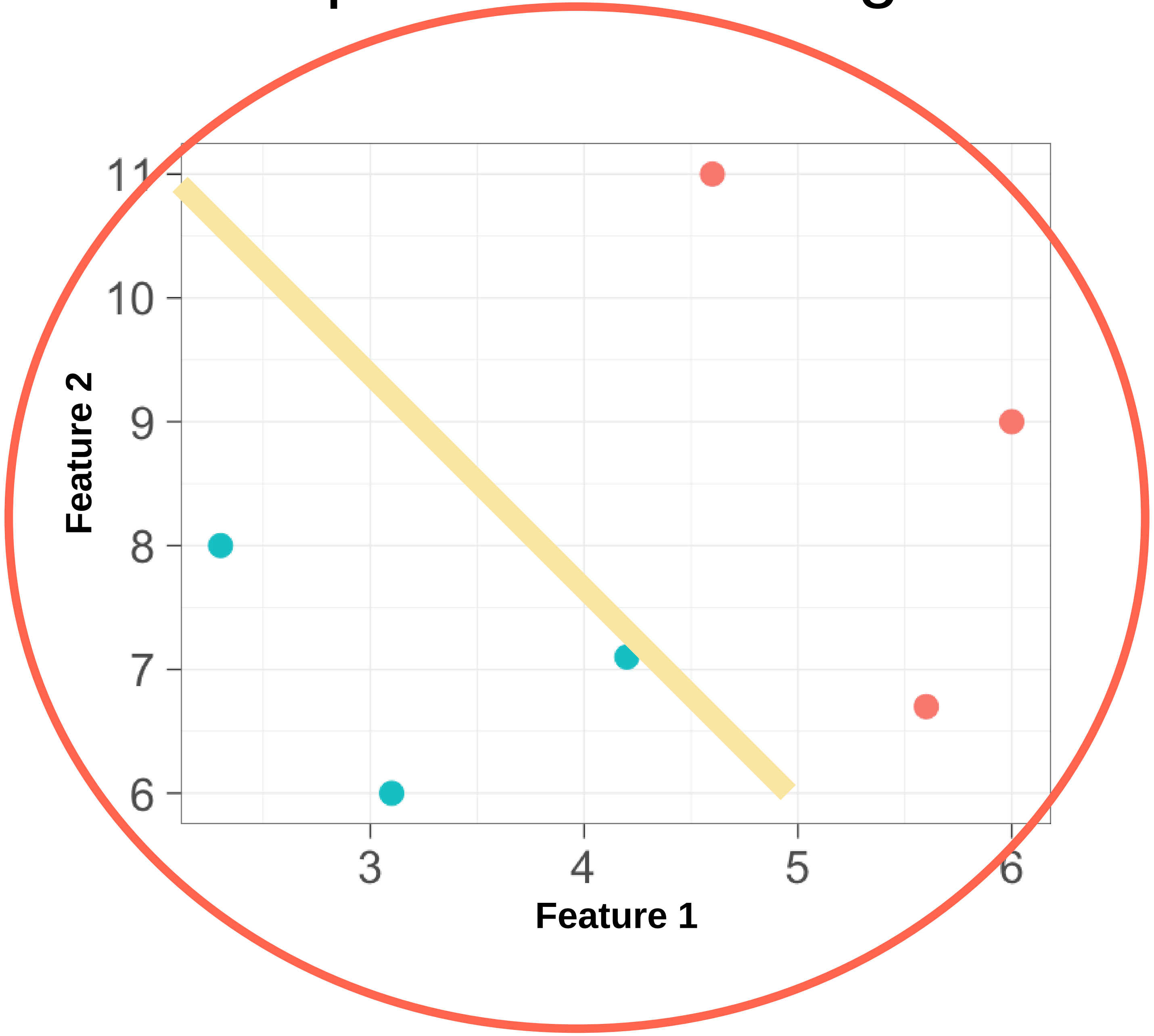
Outline

- Classification and its applications to CLM
- Classification with logistic regression
- Classification with decision trees
- Black-box classification models
- Measuring the quality of classification models
- Pitfalls: class imbalance & overfitting

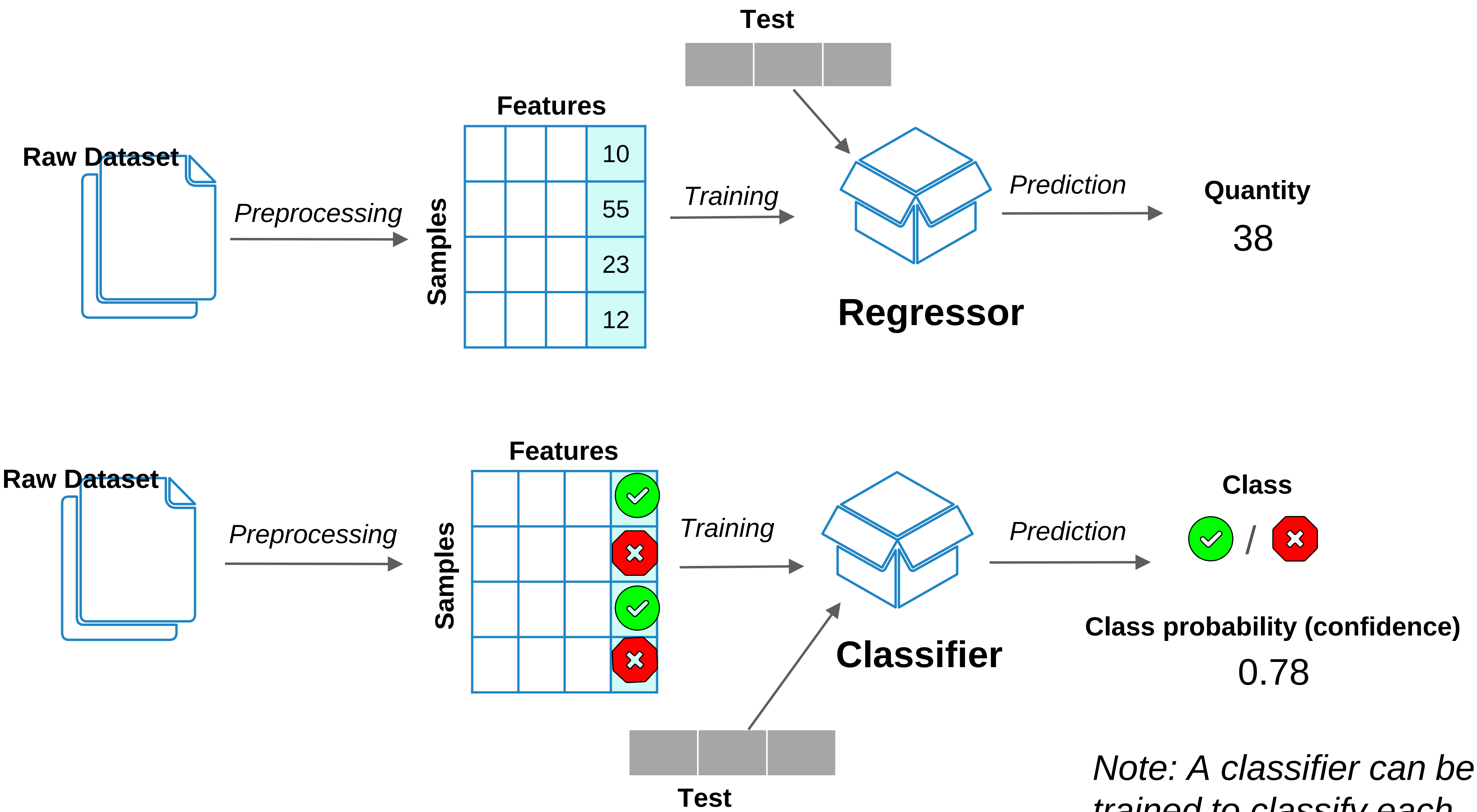
Unsupervised learning



Supervised learning

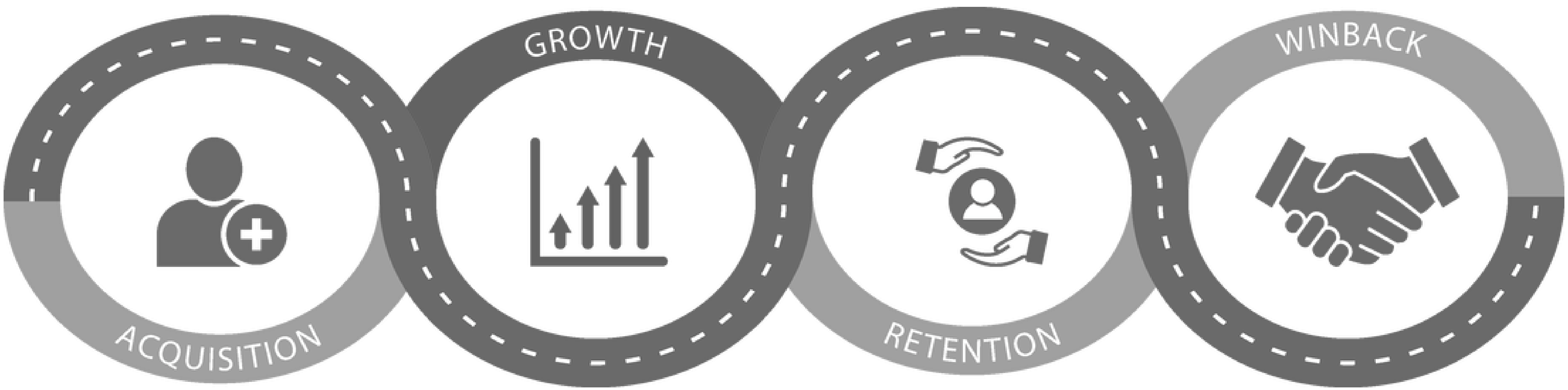


Supervised Learning
Regression vs. classification



Note: A classifier can be trained to classify each sample into more than two classes.

Classification in Customer Lifecycle Management

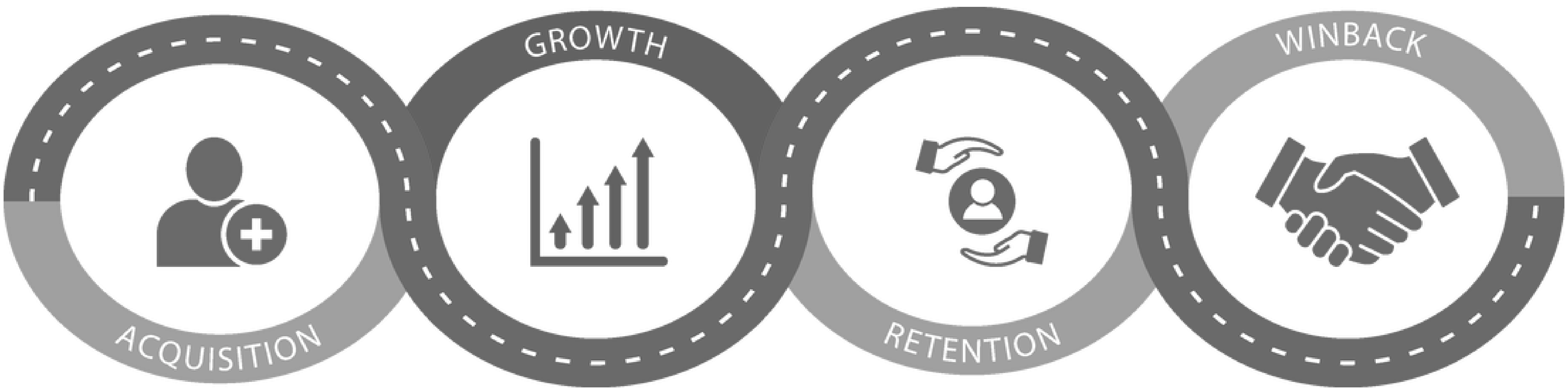


Lead propensity

Cross-sell/up-sell propensity

Response modeling / buying propensity

Classification in Customer Lifecycle Management



Lead propensity

Cross-sell/up-sell propensity

Response modeling / buying propensity

Fault/Complaint prediction

<u>Churn propensity</u>

Win-back propensity

What is churn?

event-based

subscription-based

Less obvious:

when used last time, bought, answered email, etc

Obvious:

when they stop their subscription, terminate contract, etc

Why care?

For most companies, the customer acquisition cost (cost of acquiring a new customer) is multiple times higher than the cost of retaining an existing customer

Bringing a new customer to the level of profitability as an old customer takes time

Churners/switchers often can be avoided with proactive measures

How to decrease churn?

Short-term
Operational intervention

Long-term
Tactical intervention

How to decrease short-term churn?

- Recurrent payments (automatic subscription renewal)

- Reminder emails (e.g. account expiration)

- Providing extra services and knowledge

  (e.g. figure out where clients have troubles
  and provide help with that)

Use scoring model to determine the intervention type and moment

Logistic regression

is not regression!*

$$y_i \in \{1, ..C\}$$

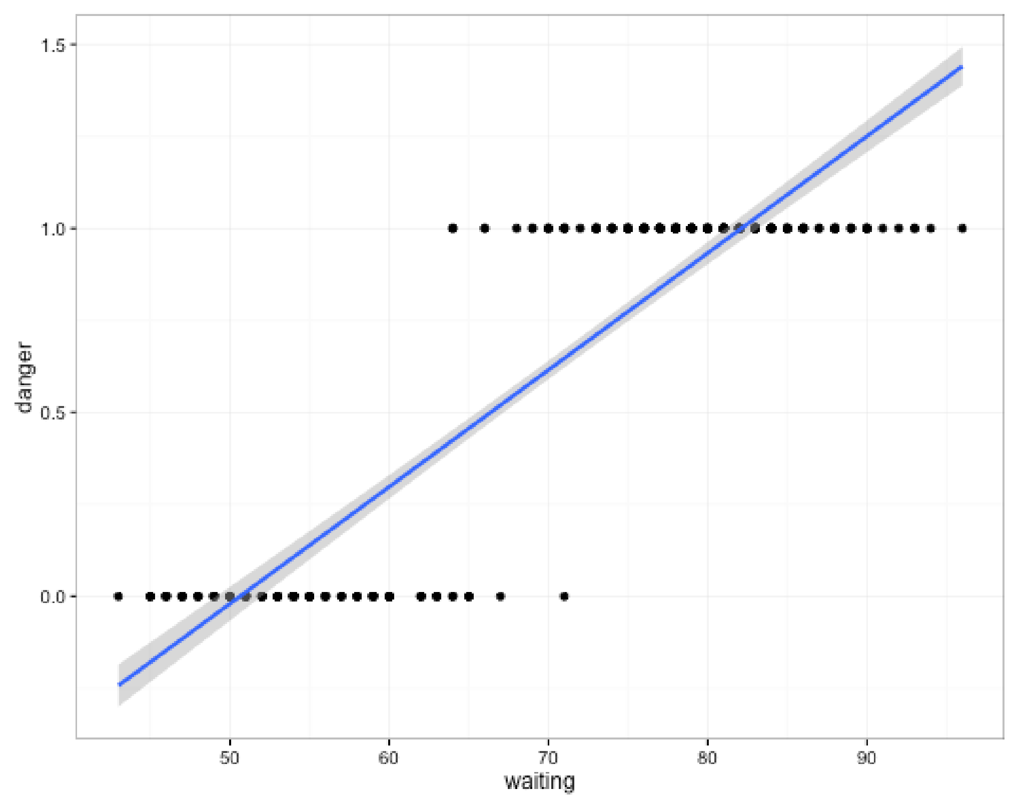
it is classification

* it is called so due to its similarity to linear regression

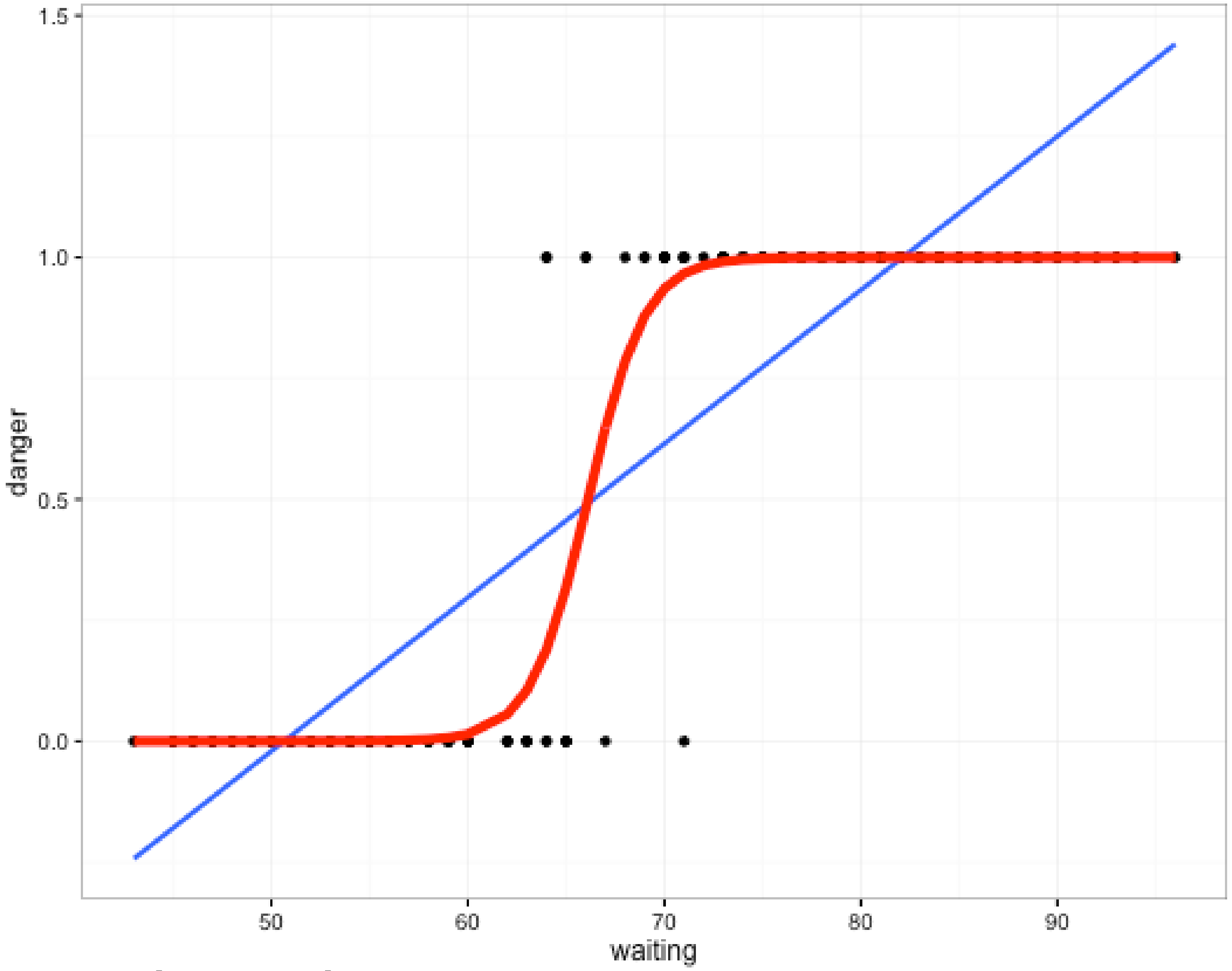
Binary logistic regression

means that y is binary: (0,1)

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

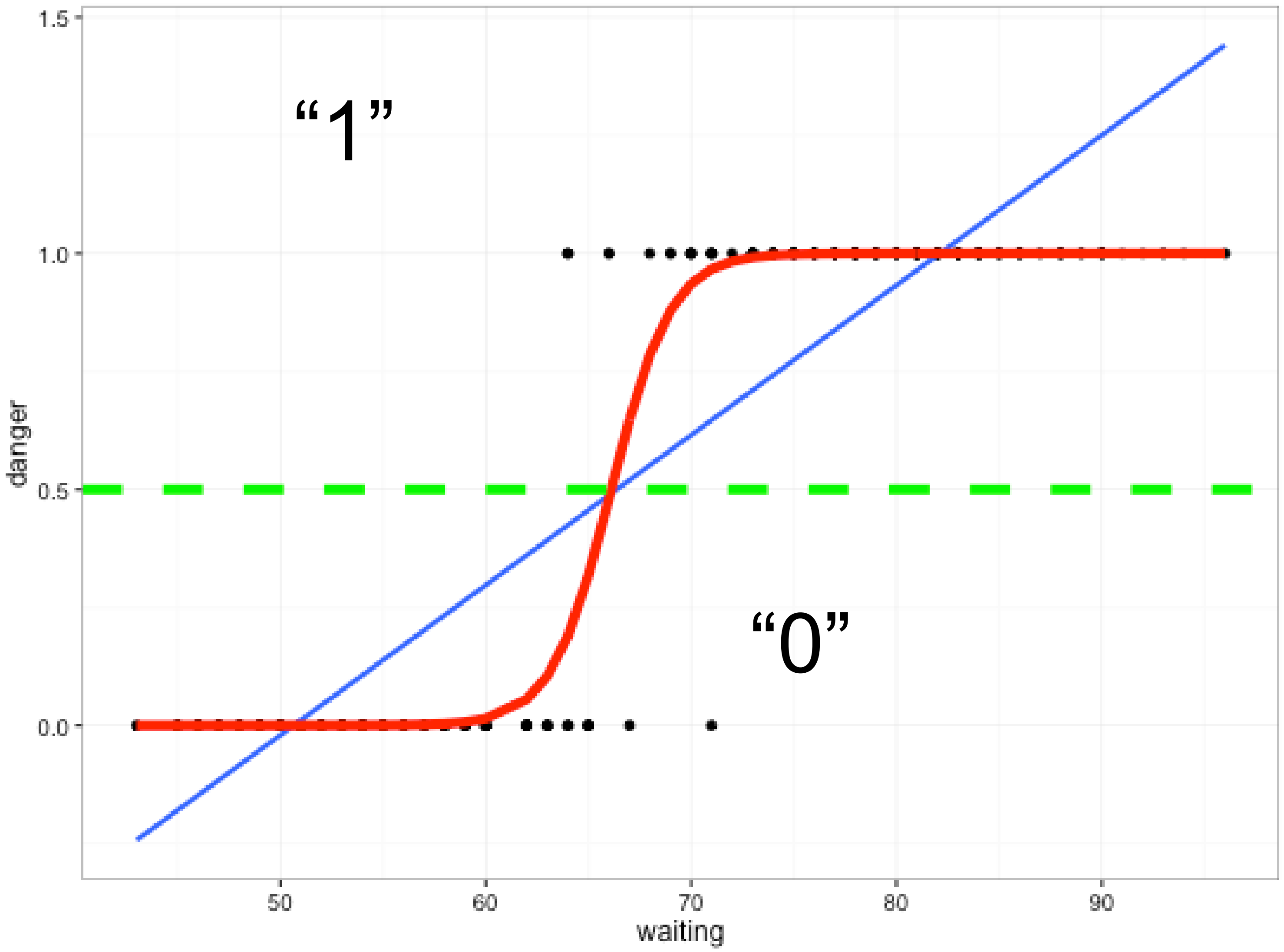


Binary logistic regression



- sigmoid means S-shaped
- also known as squashing function since it maps the line to [0,1],
- which is necessary if the output needs to be interpreted as probability

Binary logistic regression



If we threshold the output at 0.5, we create a decision rule of the form

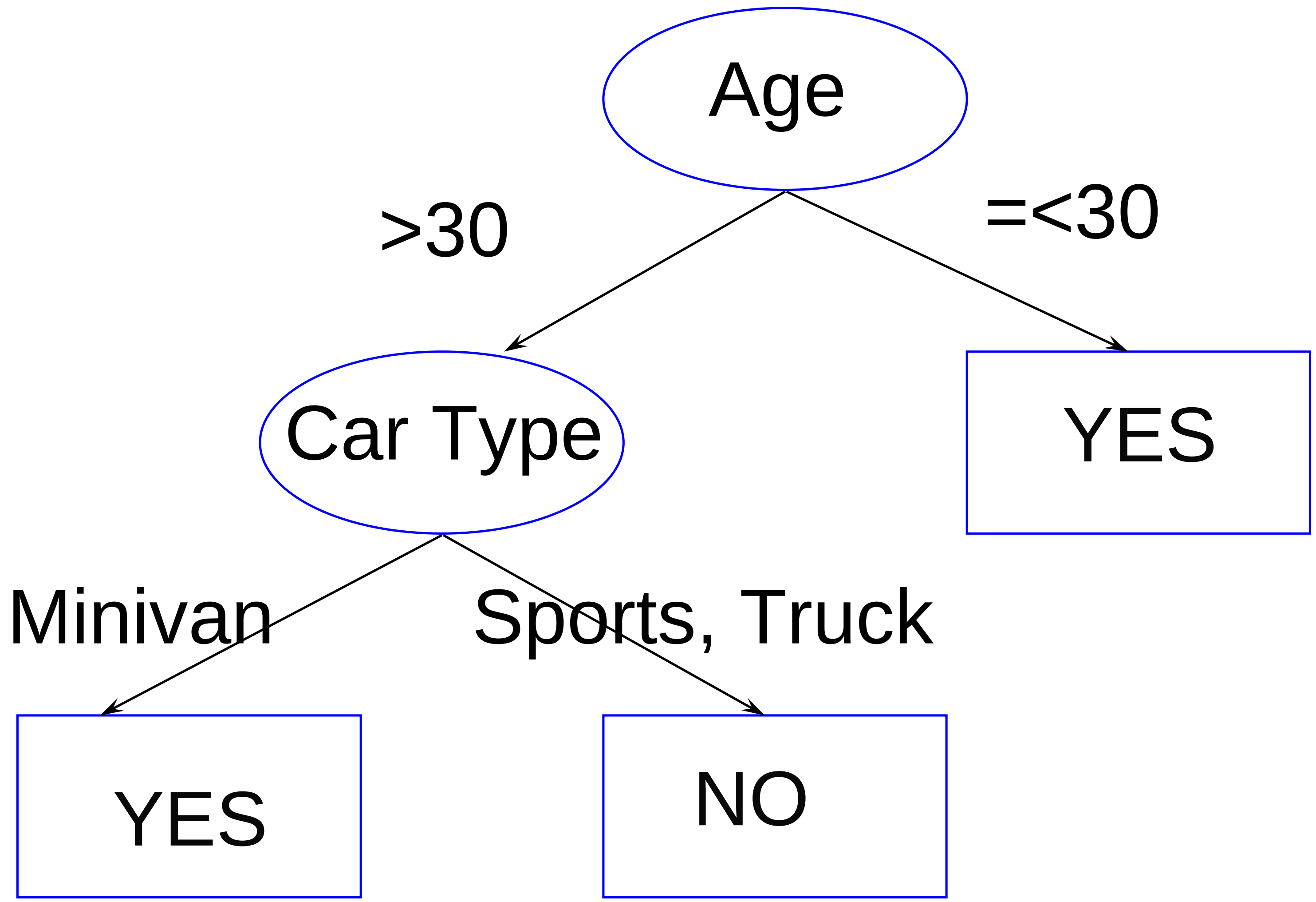
$$\hat{y} = 1 \Leftrightarrow p(y = 1|\mathbf{x}) > 0.5$$

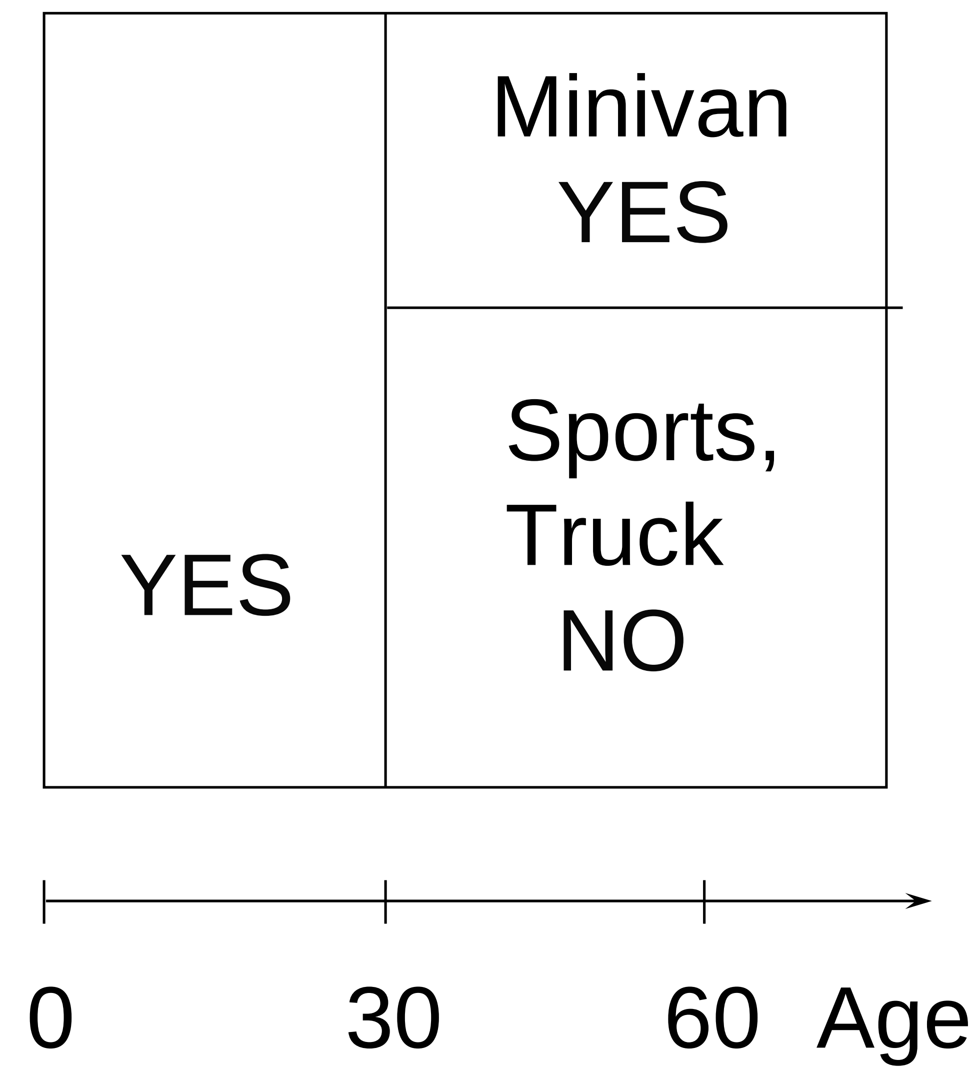
Outline

- Classification and applications to CLM
- Classification with logistic regression
- Classification with decision trees
- Black-box classification models
- Measuring the quality of classification models
- Pitfalls: class imbalance & overfitting

Decision Trees

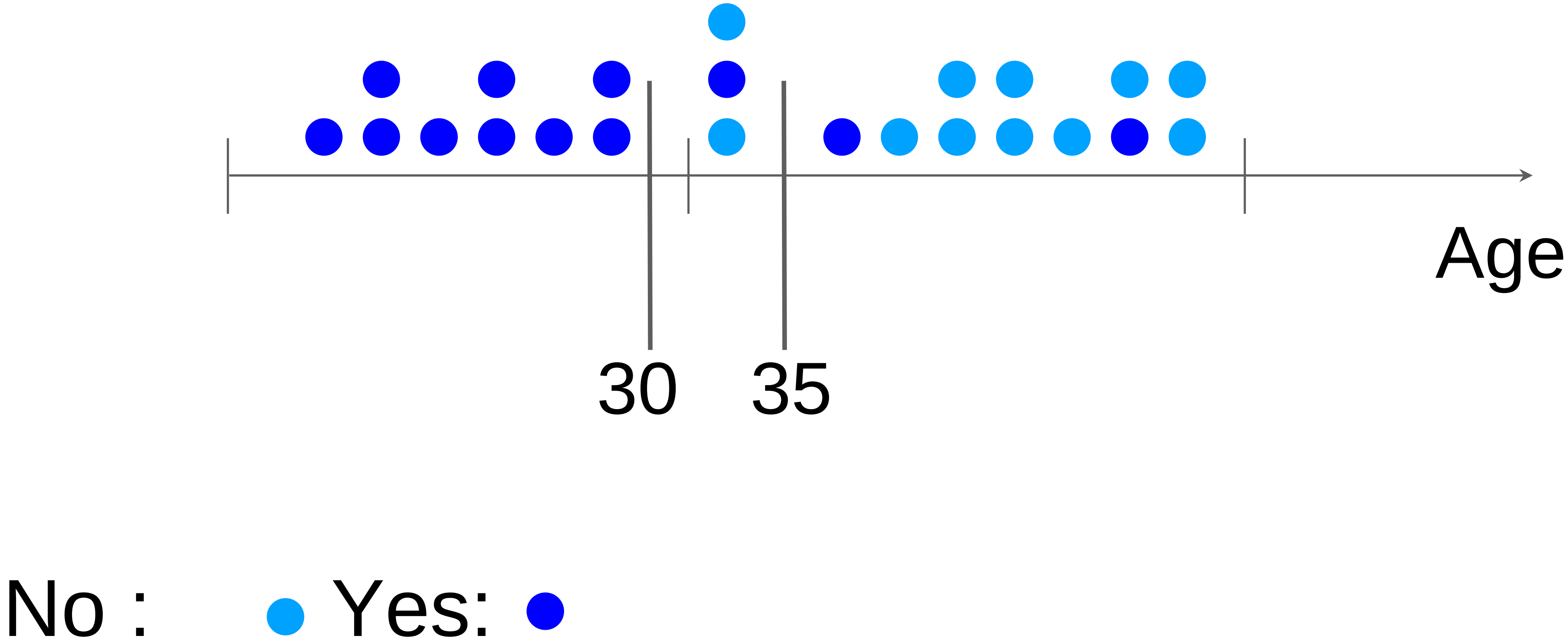
Who is cool ?





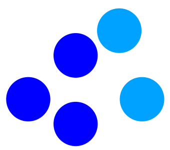
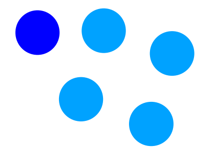
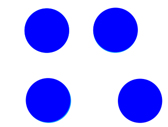
Decision Tree: Split Selection

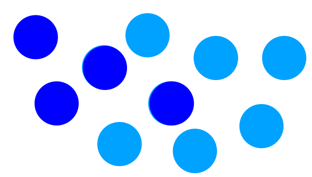
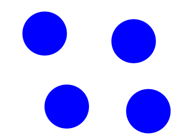
- Numerical or ordered attributes: Find a split point that separates the (two) classes

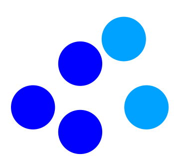
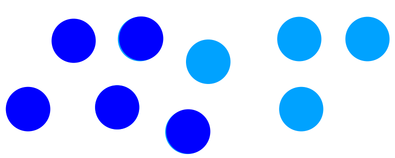


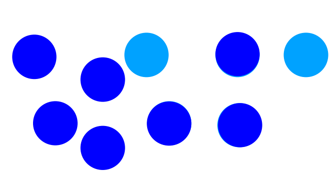
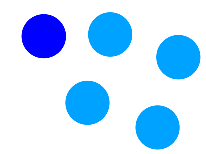
Decision Tree: Split Selection

Categorical attributes: How to group?

Sport:  Truck:  Minivan: 

(Sport, Truck):  (Minivan): 

(Sport):  (Truck, Minivan): 

(Sport, Minivan):  (Truck) 

Which split should we pick?

Hint: the one with the higher “purity” → impurity measure

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition

Decision Tree

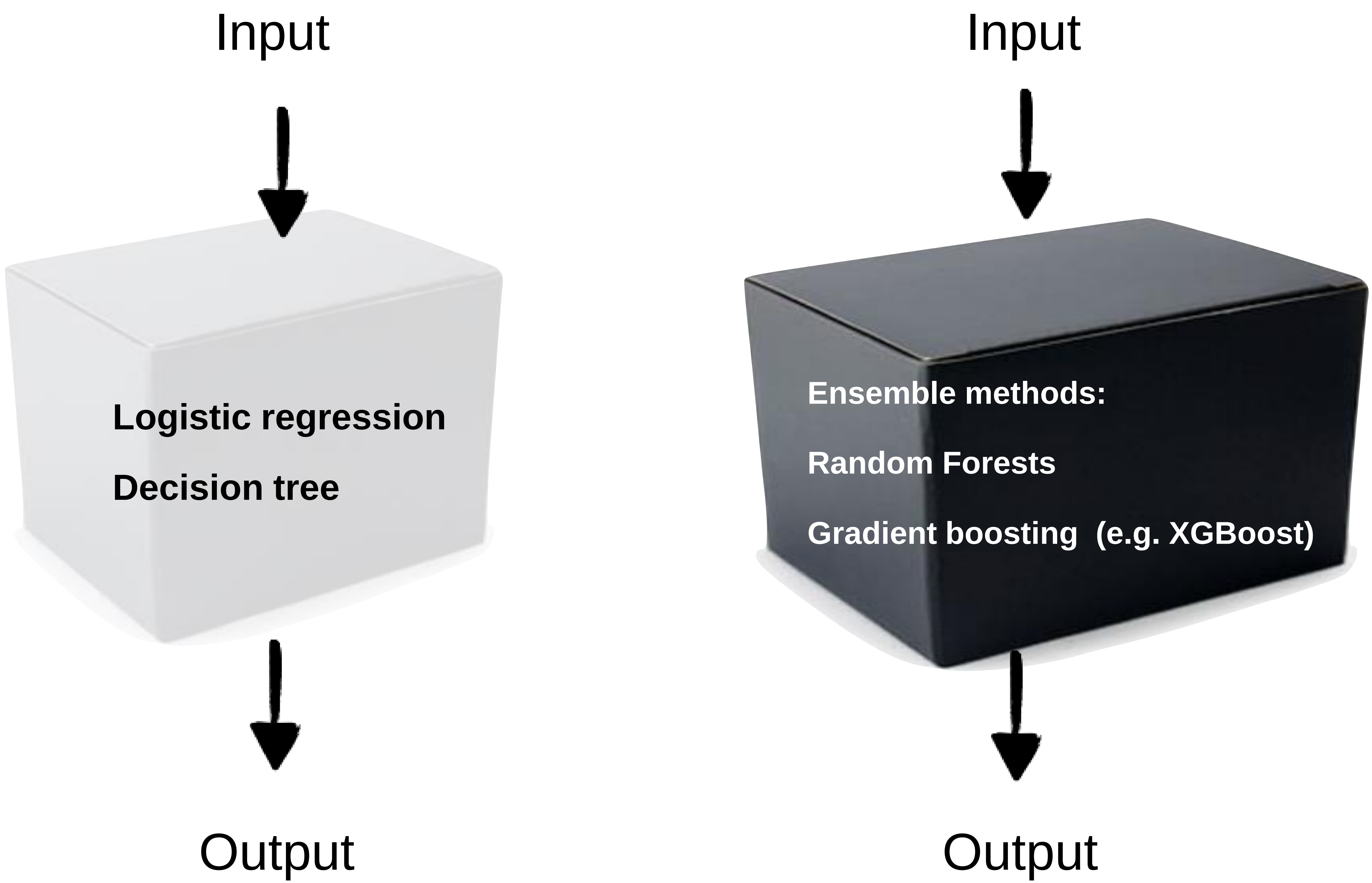
Advantages

1. Easy to Understand
2. Useful in Data exploration
3. Very fast to calculate (even for large datasets)

Disadvantages

1. Low accuracy (like logistic regression)
2. Over fitting (more on this later)

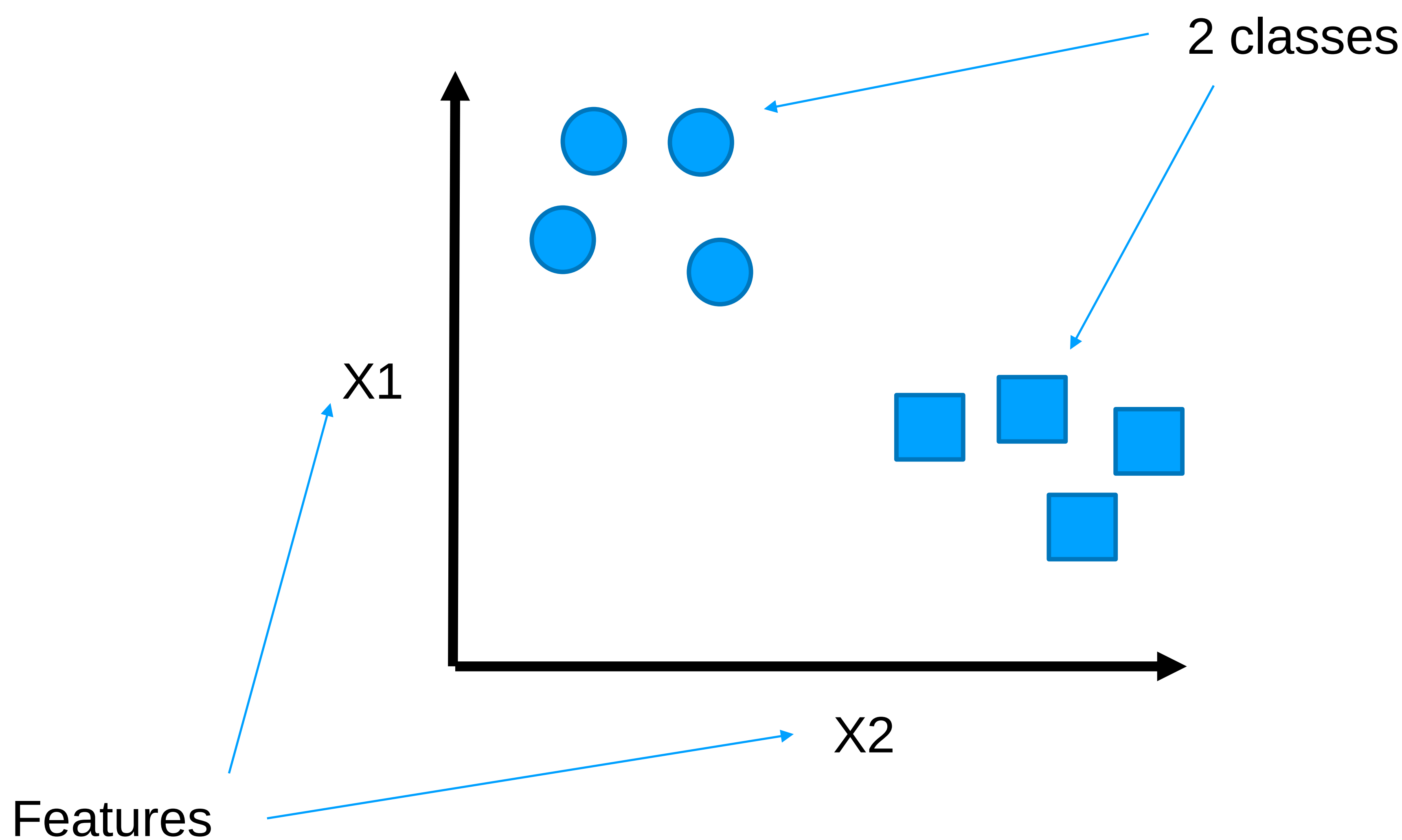
White-Box vs. Black-Box Models



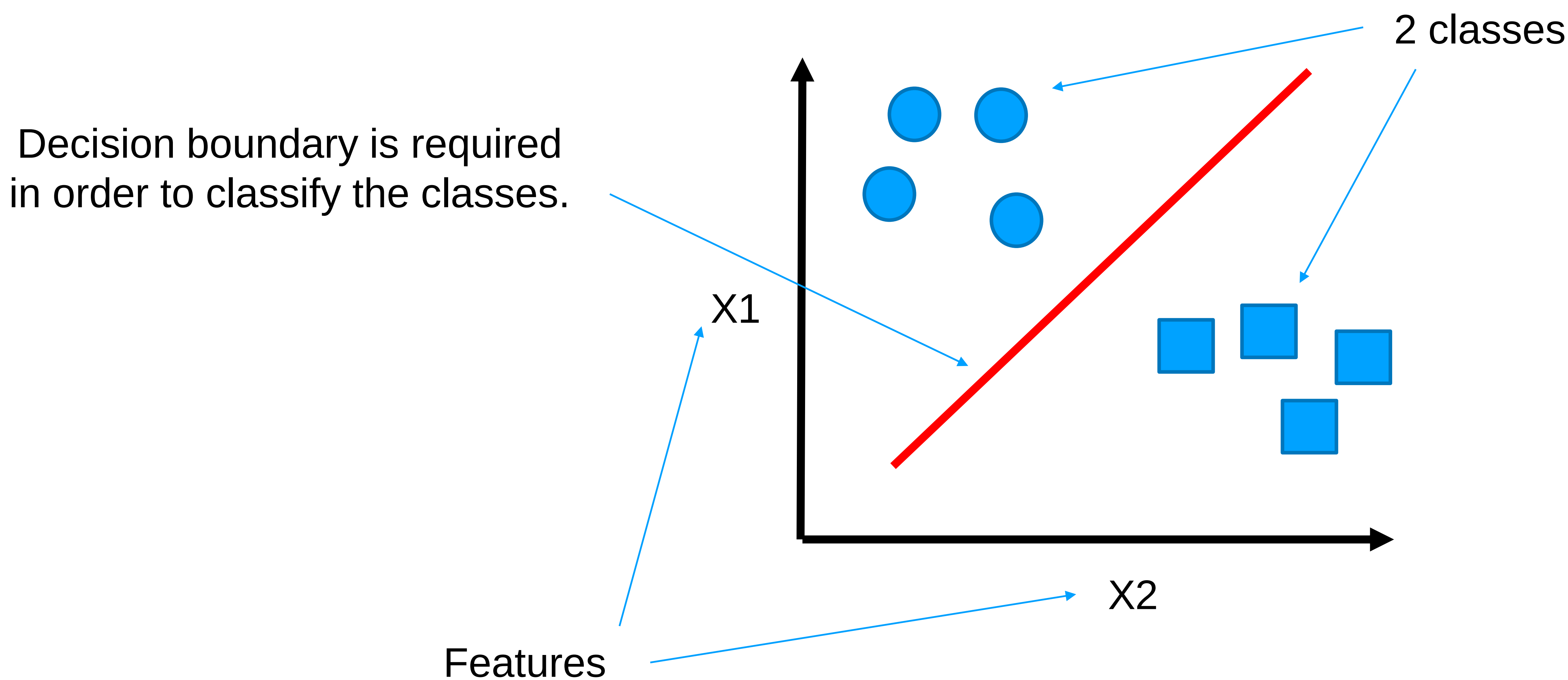
Outline

- Classification and applications to CLM
- Classification with logistic regression
- Classification with decision trees
- Black-box classification models
- Measuring the quality of classification models
- Pitfalls: class imbalance & overfitting

Support Vector Machine (SVM)

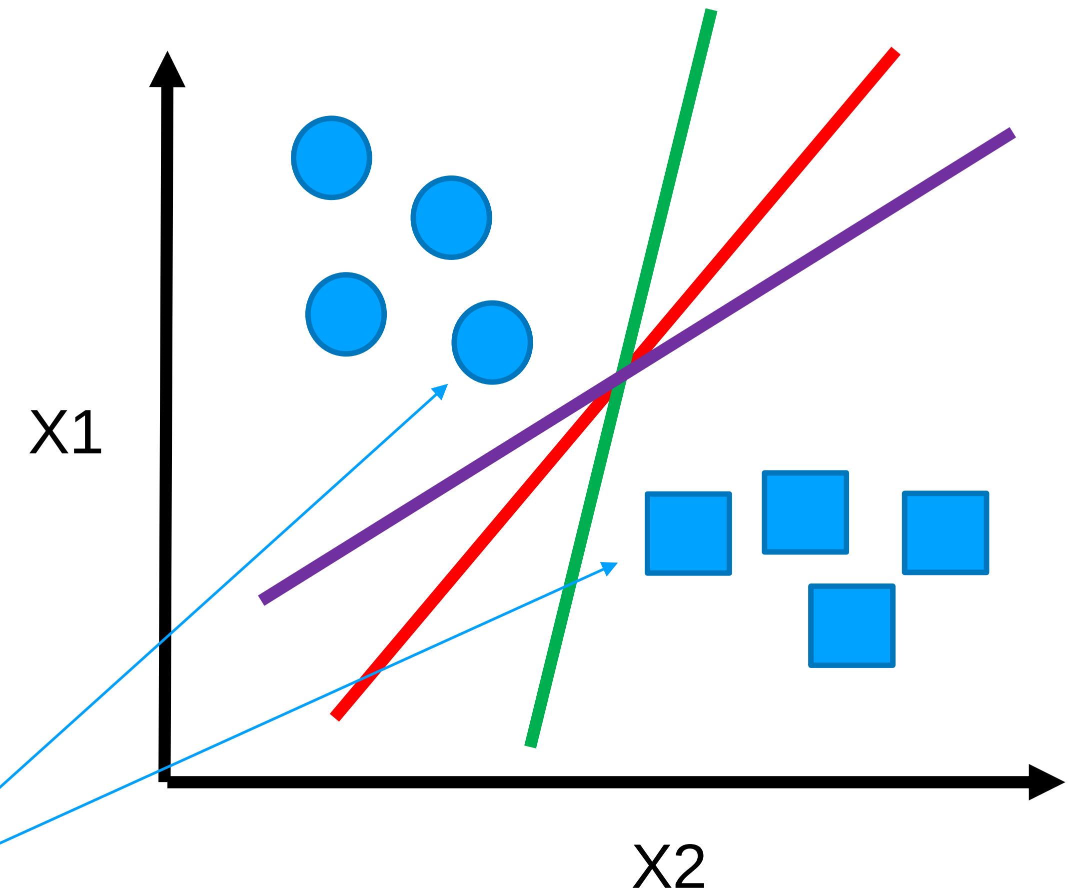


Support Vector Machine (SVM)



Support Vector Machine (SVM)

Using the support vectors we can select the best line to divide the data



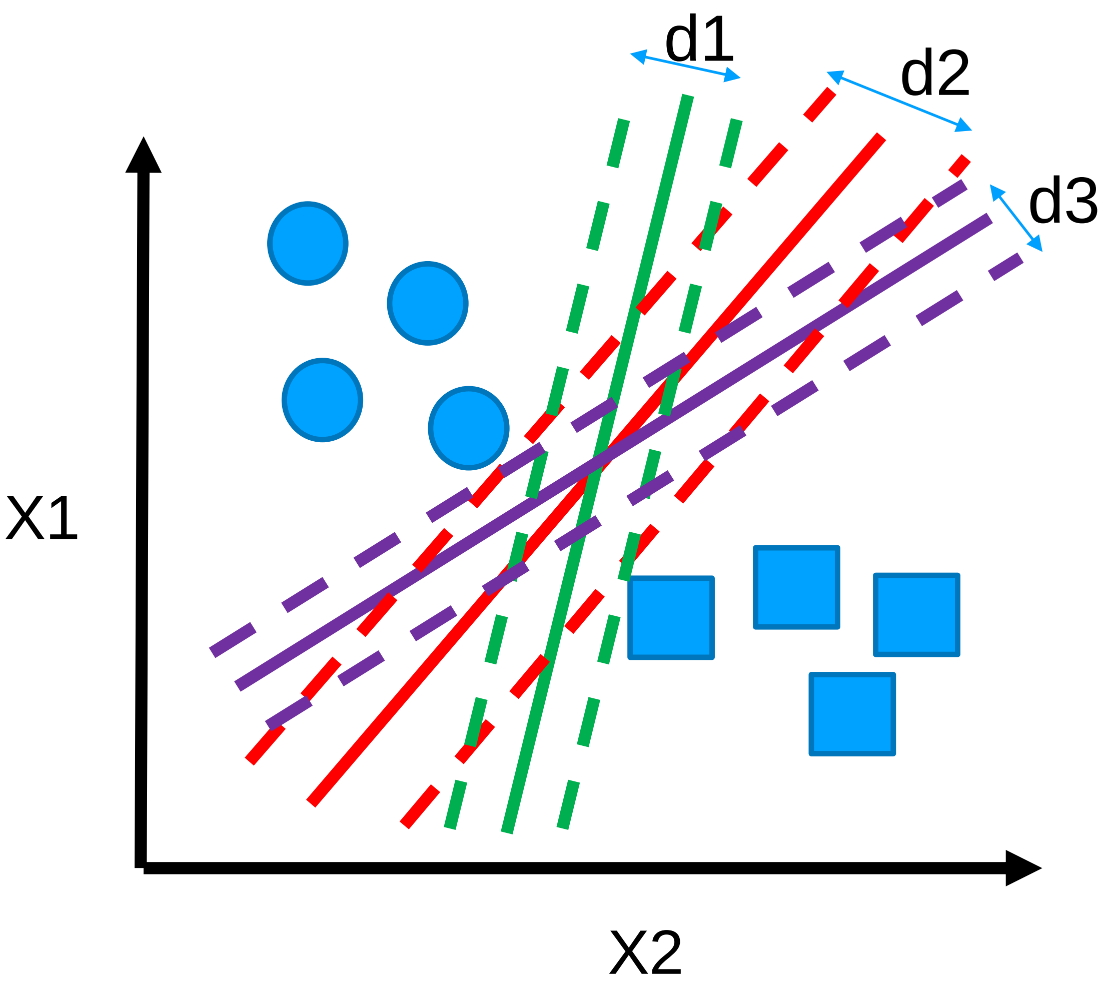
Support Vectors are the points which are very close to each a dividing line

Support Vector Machine (SVM)

Margin distance: d1, d2, d3

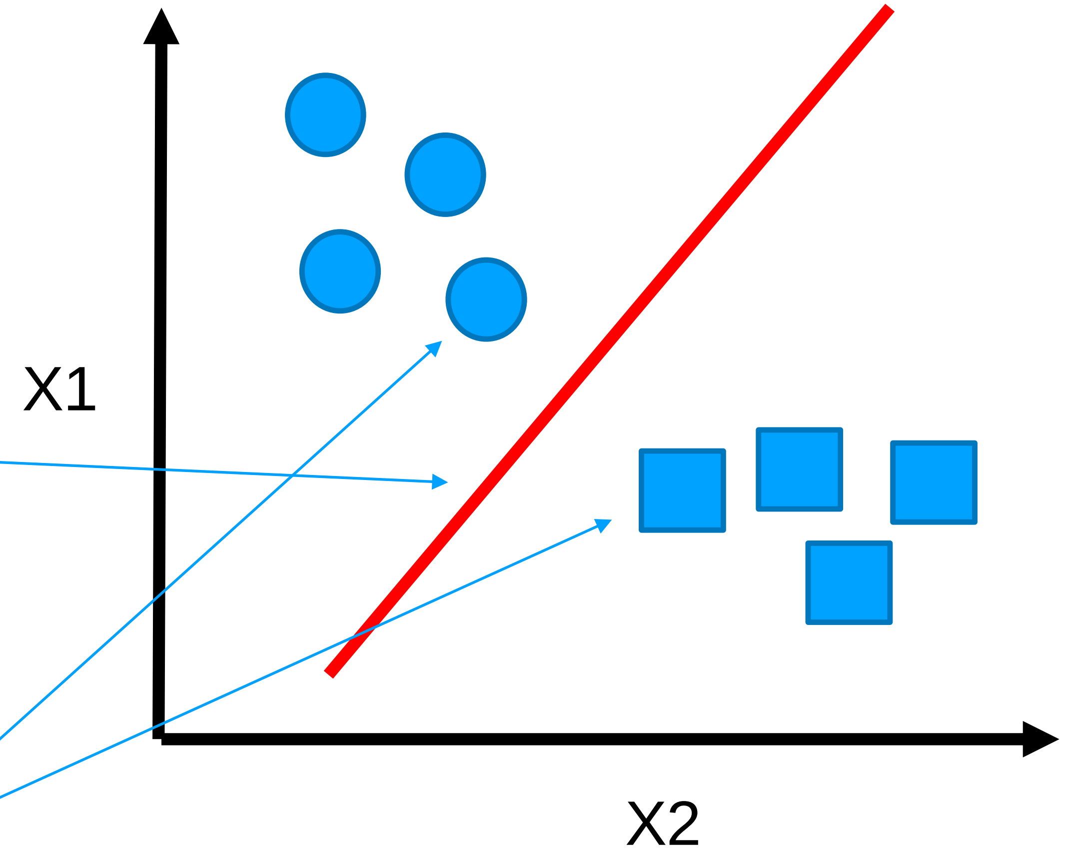
Margin is distance between the support vectors and a dividing line

The best line will always have the greatest margin distance between the support vectors



Support Vector Machine (SVM)

Using the support vectors we can select the best line to divide the data



Hyperplane

(Hyperplane has maximum distance to the support vectors of any class)

Support Vectors are the points which are very close to each a dividing line

SVM

Understanding Kernel SVM

2 Classes

3 Dimensions (Features)

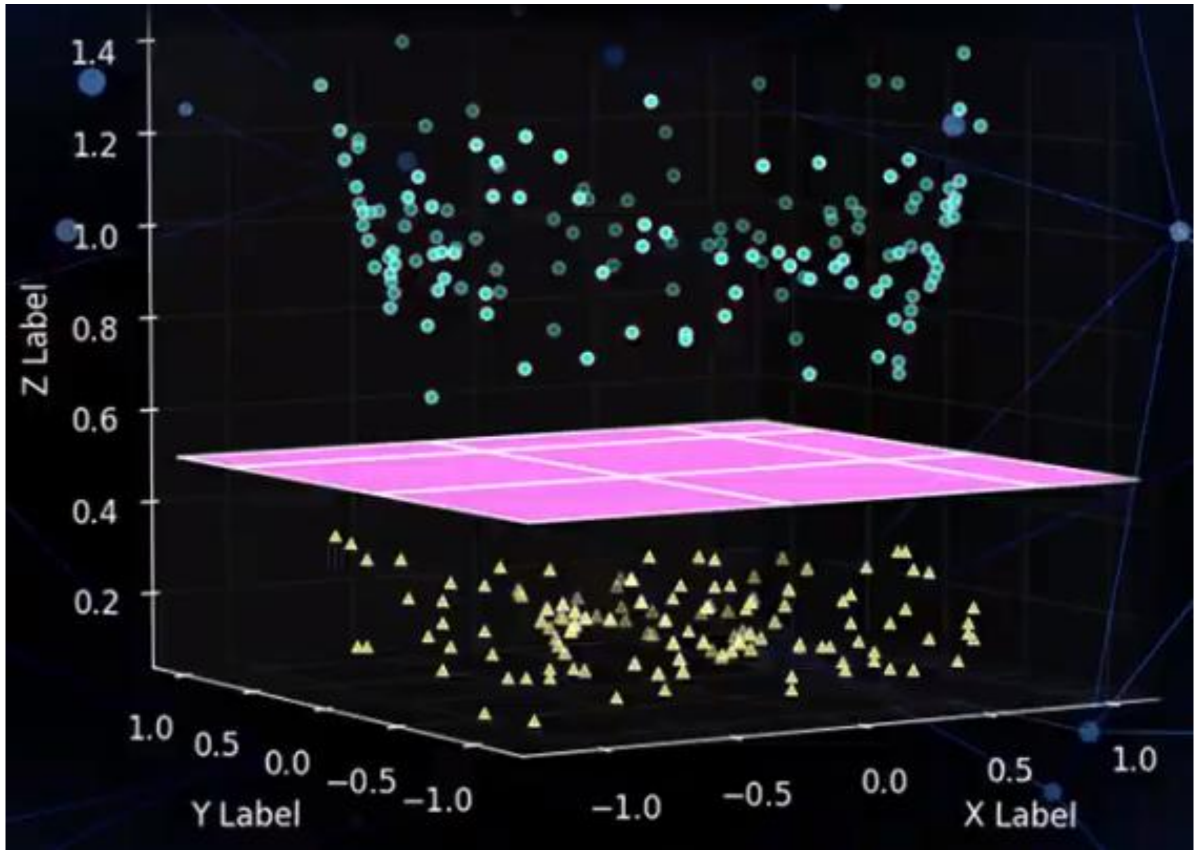
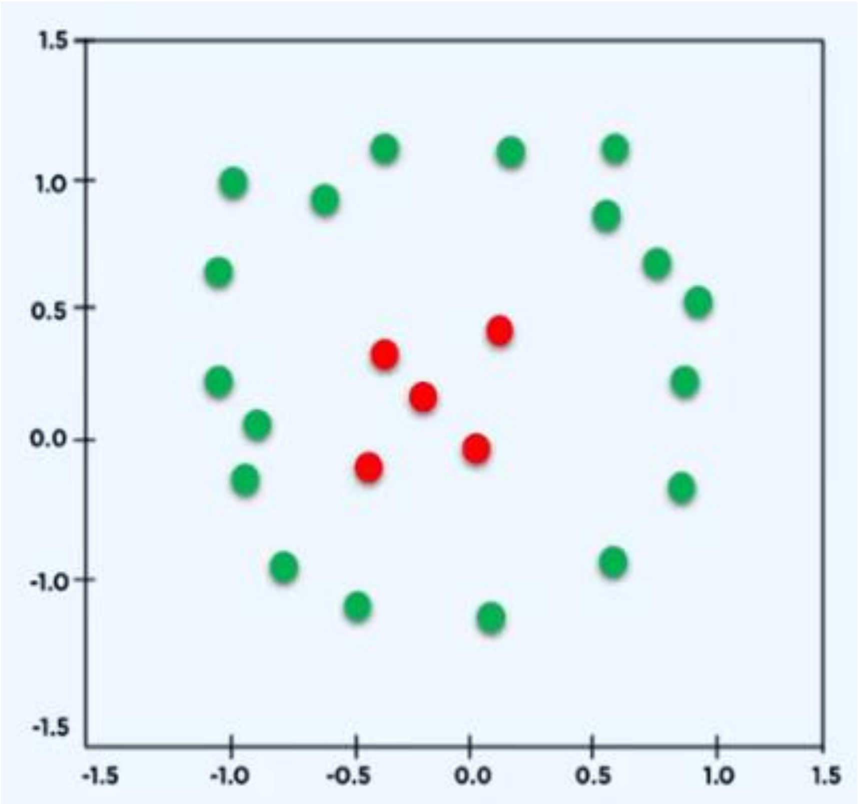
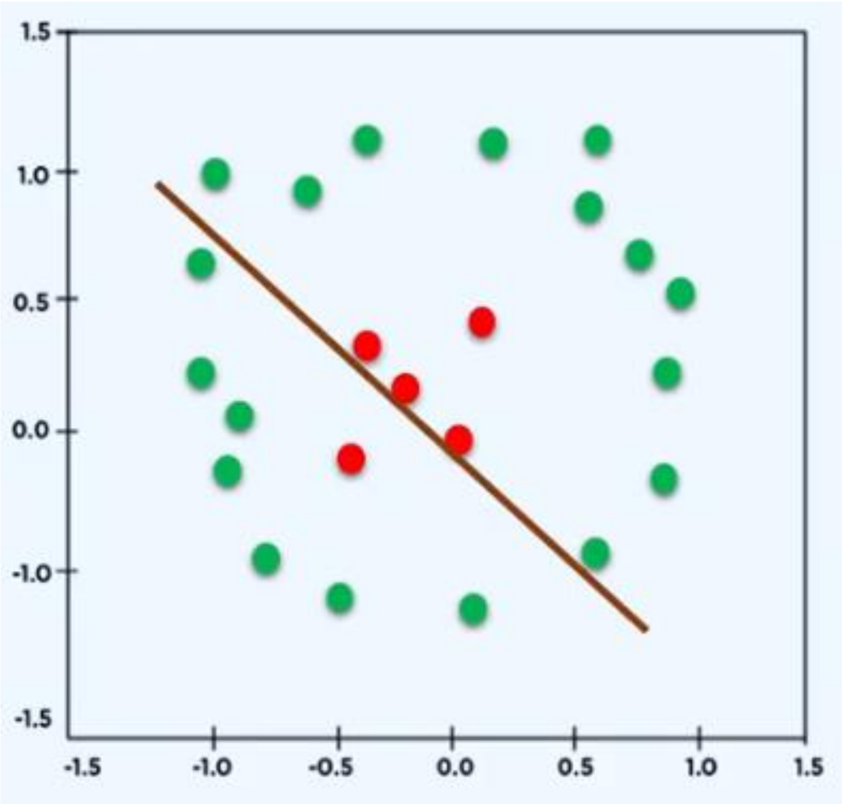


Image source: Simplilearn

How to differentiate non linear classes type distribution?

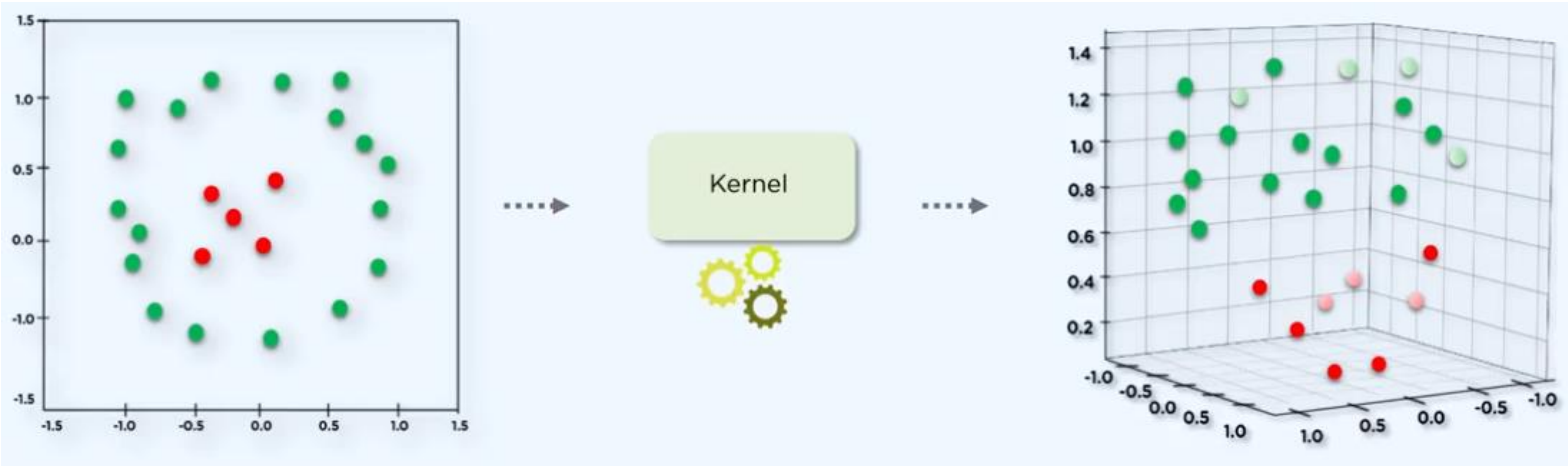


Original Data



Linear Classifier

SVM: Kernel Transformation



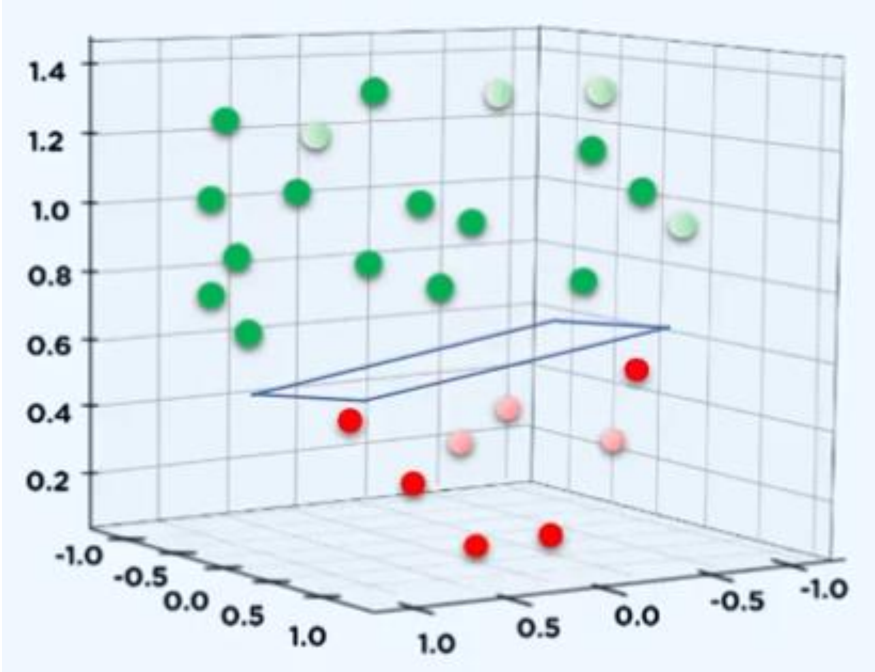
2D plot

3D plot

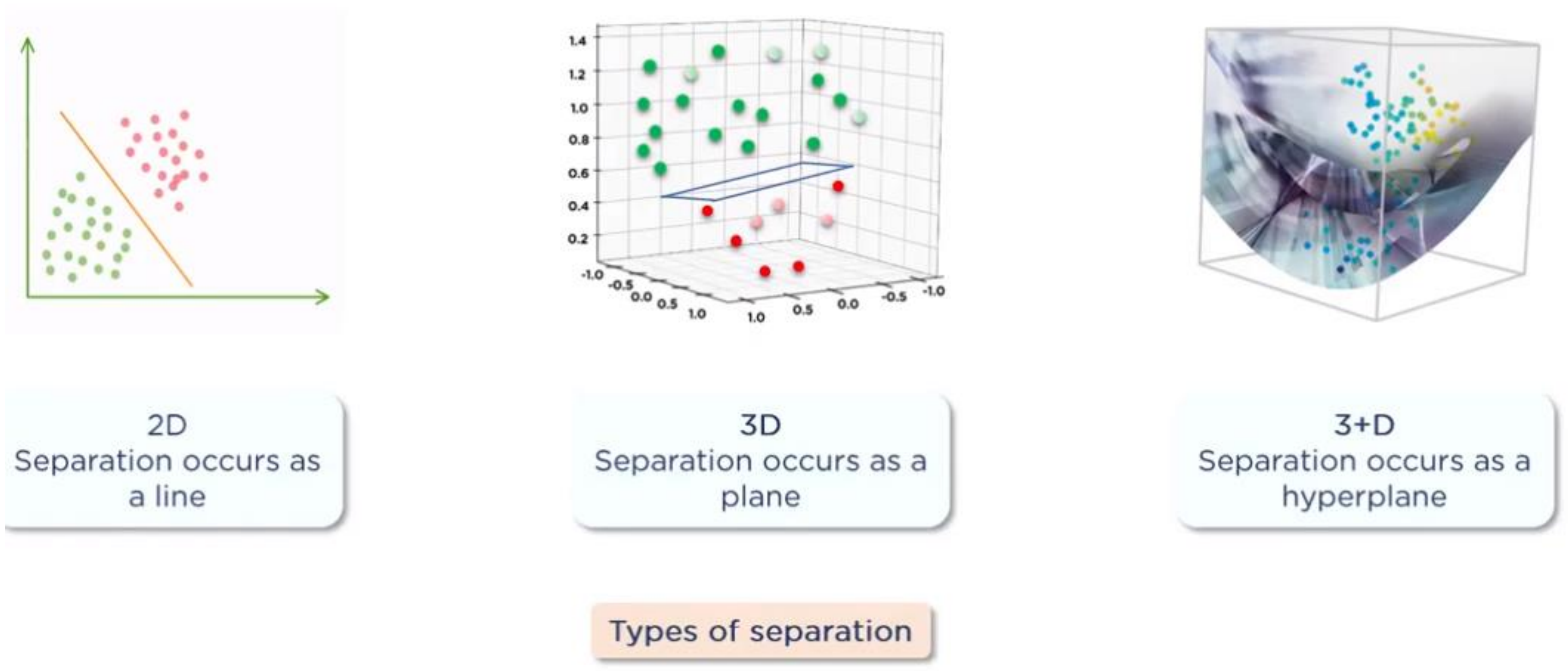
Kernel Examples:
1) Sigmoid function
2) Gaussian
3) polynomial

SVM Kernel

Once the data is in 3 Dimensions, SVM separates the data in the graph using a 2D plane



Understanding SVM Kernel



One of the favorite method for high dimensional (features) data problem

Ensemble methods

- Idea: Train multiple classifiers on subsets of the data or subsets of features and have them compete
- Examples:
  - Random forests
  - XGboost

Decision Tree

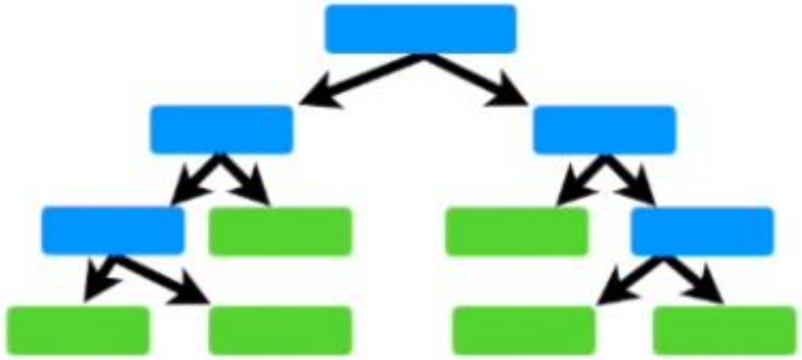
Step 1: Take the whole dataset

Step 2: Take the feature with more important information gain and divide the data

Step 3: Take the next feature with more important information gain and divide the data again and so on ..

Problem setting

F1	F2	F3	Salar y
			?



Random Forest

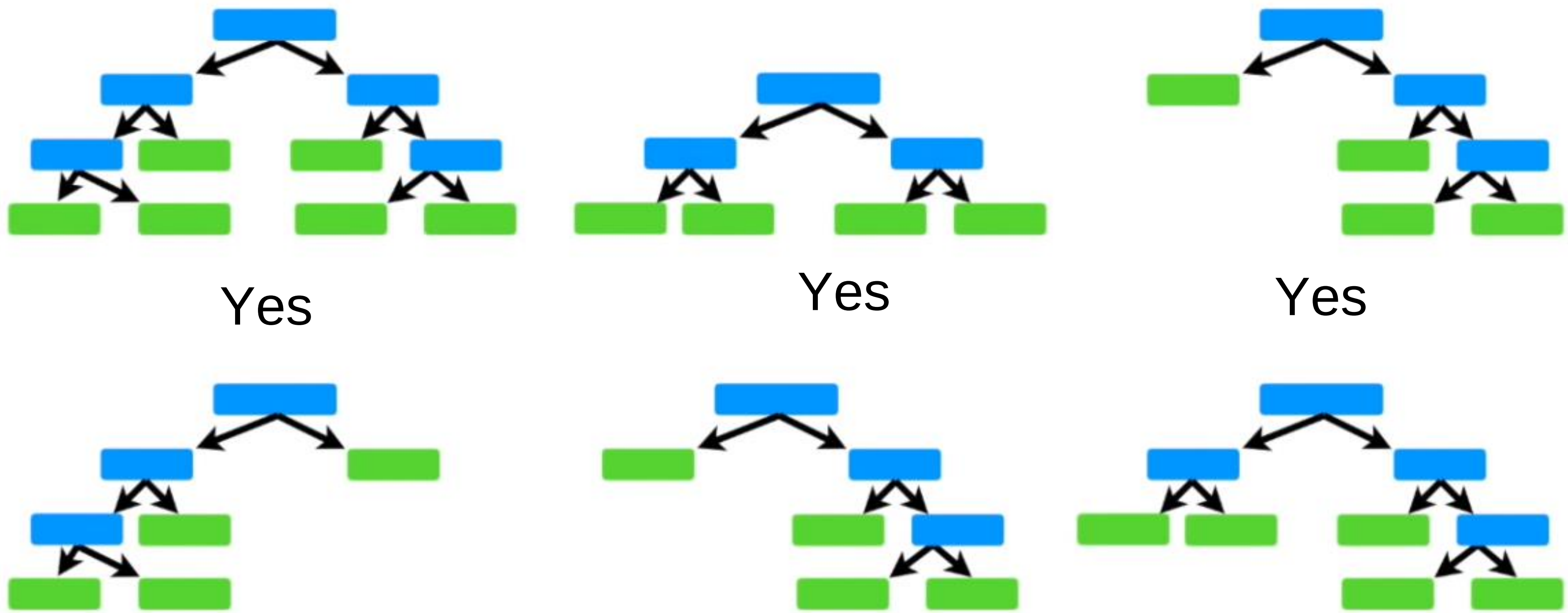
F1	F2	F3	Salary
			?

1) Multiple parallel Trees

2) Bootstrap dataset

3) Random Selection of features at every step

4) Voting



Yes Yes Yes

No Yes No

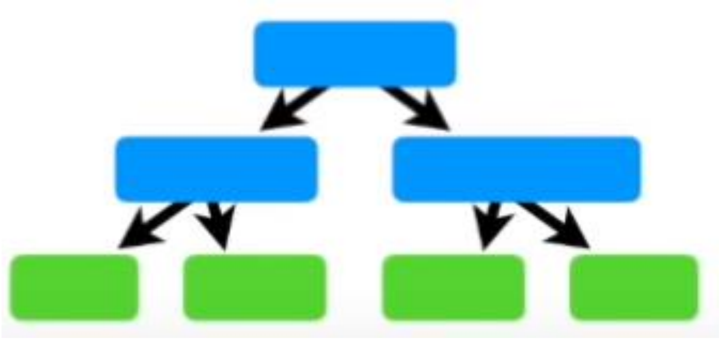
Gradient Boosting Machine

Gradient Boost builds fixed sized trees based on previous trees error

F1	F2	F3	Salar y
			?

Gradient Boost scales all the tree by the same amount.

Gradient Boost builds another tree based on the errors made by the previous tree



Gradient Boost scales the tree.

Gradient Boost builds another tree based on the errors made by the previous tree

Gradient Boost scales the tree.

:

Gradient Boosting Machine

Gradient Boost builds fixed sized trees based on previous trees error

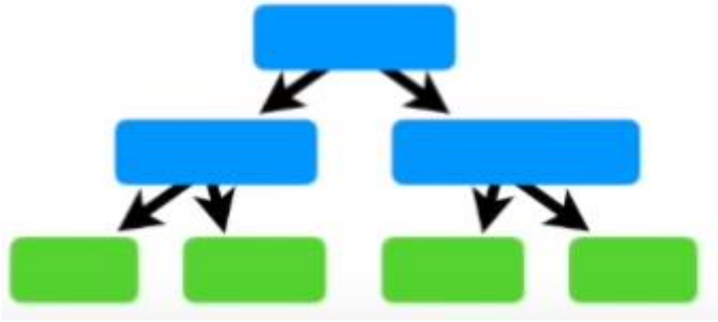
Gradient Boost scales all the tree by the same amount.

Gradient Boost builds another tree based on the errors made by the previous tree.

Gradient Boost scales the tree.

Gradient Boost builds another tree based on the errors made by the previous tree

Gradient Boost scales the tree.
:
STOPING CONDITION: Gradient Boost continues to build trees in this fashion until
1) it has made the number of trees you asked for **or**
2) addition trees fail to improve the fit.



Ensemble methods

- Idea: Train multiple classifiers on subsets of the data or subsets of features and have them compete
- Examples:
  - Random forests
  - XGBoost, LightGBM (state of the art)
- Why is it powerful?
  - Because we can play with the parameters of the ensemble, e.g. number of trees, depth, etc.

  → This is called *hyperparameter optimization*
  - Because they (generally) do their own *feature selection*
    - Except when the number of features is large (e.g. > square root of # samples), then use Boruta or BoostARoota or similar technique for feature selection

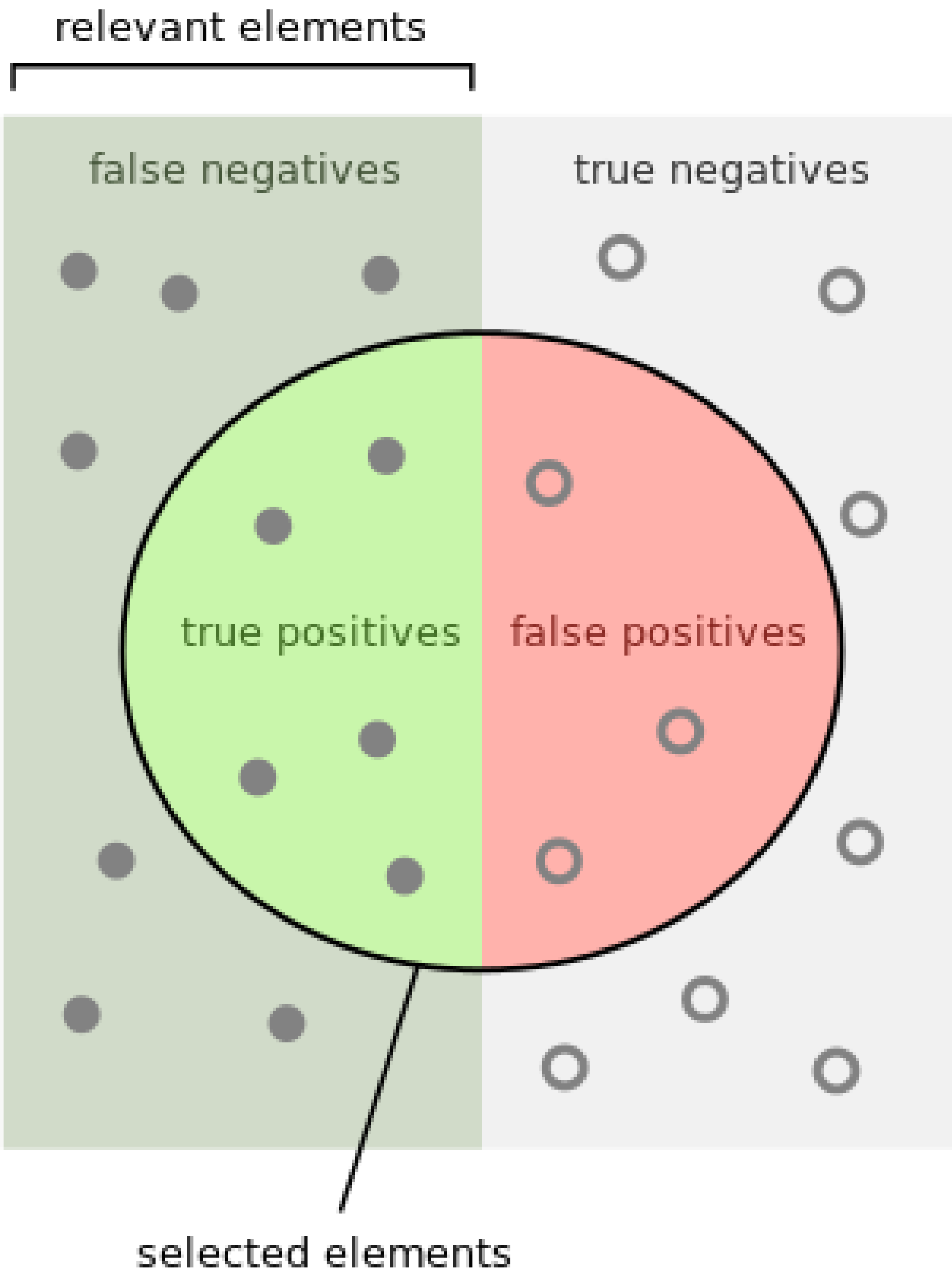
Outline

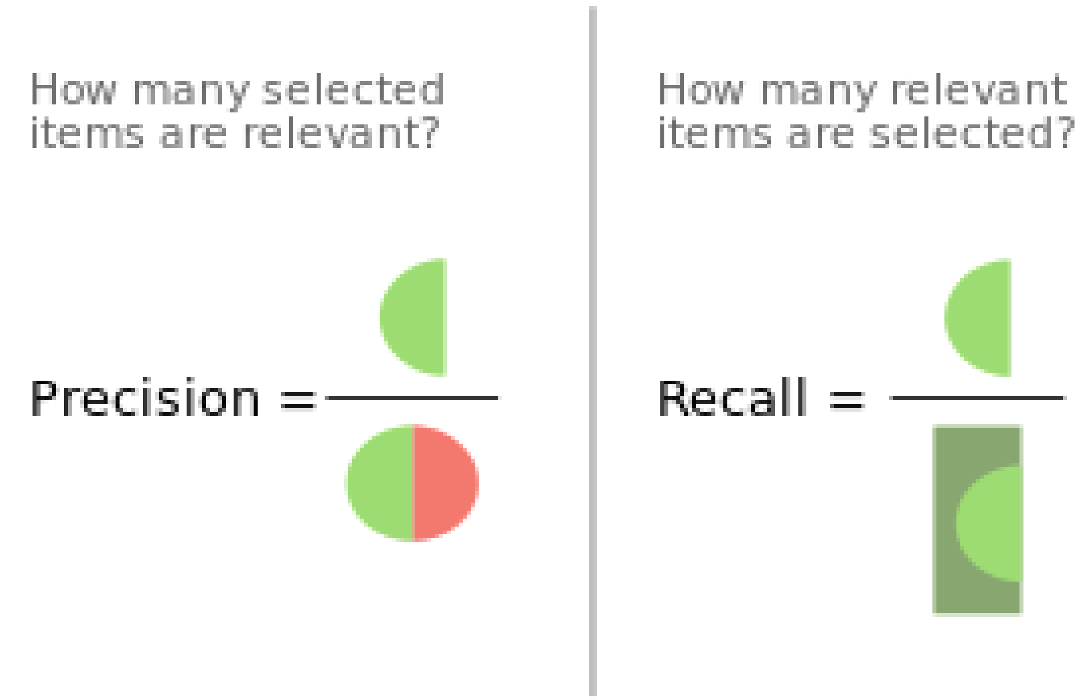
- Classification and applications to CLM
- Classification with logistic regression
- Classification with decision trees
- Black-box classification models
- Measuring the quality of classification models
- Pitfalls: class imbalance & overfitting

Quality of Classifiers

		True condition	
		Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Quality of classifiers





$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

Quality of Classifiers

Features **Actual class** **Probability of positive class**

		churn	prediction
1	1	0	0.3
2	2	0	0.6
3	3	1	0.9
4	4	0	0.1
5	5	1	0.9

It depends on the class probability threshold!

Calculate a confusion matrix

Quality of Classifiers

Features **Actual class** **Probability of positive class**

		churn	prediction
1	1	0	0.3
2	2	0	0.6
3	3	1	0.9
4	4	0	0.1
5	5	1	0.9

Threshold= 0.5
i.e., if prediction >= 0.5
then churn (1)
else no churn (0)

Predicted

Actual

	1 (pos)	**0 (neg)**
1 (pos)	2	0
0 (neg)	1	2

Quality of Classifiers

Features **Actual class** **Probability of positive class**

		churn	prediction
1	1	0	0.3
2	2	0	0.6
3	3	1	0.9
4	4	0	0.1
5	5	1	0.9

confusion matrix

Predicted

Actual

	1 (P)	**0 (N)**	
1 (P)	2	0	2
0 (N)	1	2	3
	3	3	

$$\text{Precision} = \frac{tp}{tp + fp}$$

Precision = 2/3

Recall = 2/2

$$\text{Recall} = \frac{tp}{tp + fn}$$

Quality of Classifiers

Features **Actual class** **Probability of positive class**

		churn	prediction
1	1	0	0.3
2	2	0	0.6
3	3	1	0.9
4	4	0	0.1
5	5	1	0.9

Threshold= 0.8
i.e., if prediction >= 0.8
then churn (1)
else no churn (0)

Actual/Predicted	**1 (pos)**	**0 (neg)**
1 (pos)	2	0
0 (neg)	0	3

Now: what is the precision and the recall?

High Precision or High Recall ?

Spam Vs. Ham

- We should better predict all the spams as spams.
- We should not classify non spam as spam (Interview emails)
- We want *FP* to be low but *TP* to be high
- High Precision $\text{Precision} = \frac{tp}{tp + fp}$

Fraud Vs. Legitimate

- We should better predict all the frauds as frauds.
- We should not make fraud transactions as non fraud
- We want to keep *FN* as low as possible but TP to be high.
- High Recall $\text{Recall} = \frac{tp}{tp + fn}$

Quality of Classifiers

- **Accuracy:** Ratio of correctly predicted observation to the total observations
  - Accuracy = (TP+TN)/(TP+FP+FN+TN)
  - Intuitive, but terrible in case of **class imbalance**
- **F Score:** Weighted average of Precision and Recall.
  - F-Score = 2*(Recall * Precision) / (Recall + Precision)
  - More resilient to **class imbalance**

Actual/Predicted	**1 (pos)**	**0 (neg)**
1 (pos)	2	1
0 (neg)	0	3

Outline

- Classification and applications to CLM
- Classification with logistic regression
- Classification with decision trees
- Black-box classification models
- Measuring the quality of classification models
- Pitfalls: class imbalance & overfitting

Pitfall 1: Class imbalance

A credit card company collects data about OK vs fraudulent transactions

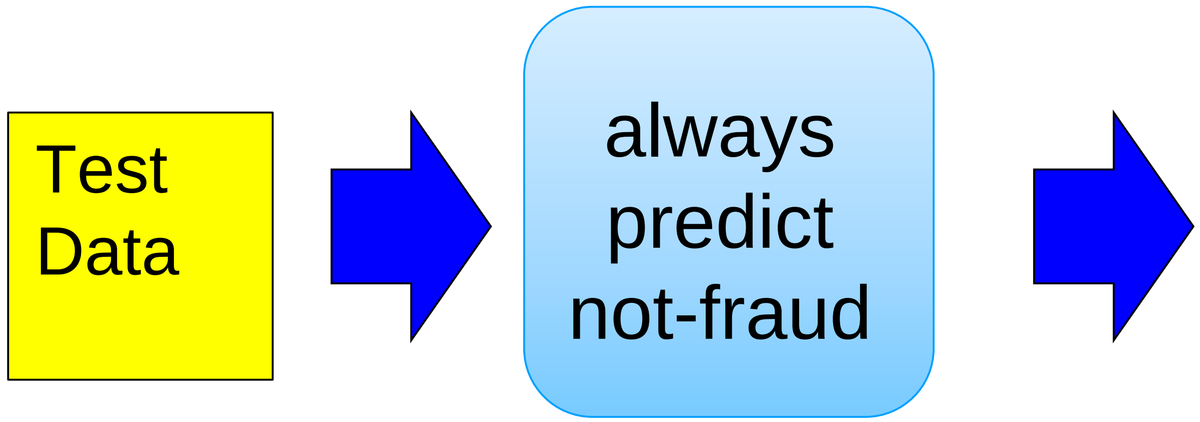
OK transactions make up 99.999% of transactions

We train a classifier to classify transactions into OK and fraudulent

We get an accuracy of 99.999%

Why?

Class imbalance



99.997% accuracy

99.997% precision

0 recall!

Why does the classifier learn this?

Because twisting the classifier to catch the bad guys, makes us also “catch” some good guys

… so no visible improvement

Slide by: David Kauchak

Dealing with class imbalance

- Use a quality measure that is as robust as possible
  - F-Score is OK in extreme cases (like the previous one)
    - Less robust in more subtle cases, e.g. 90-10 class imbalance
  - Area Under the Precision Recall Curve even better (AUPRC)

ROC – Robust Measurement of Classifier Quality

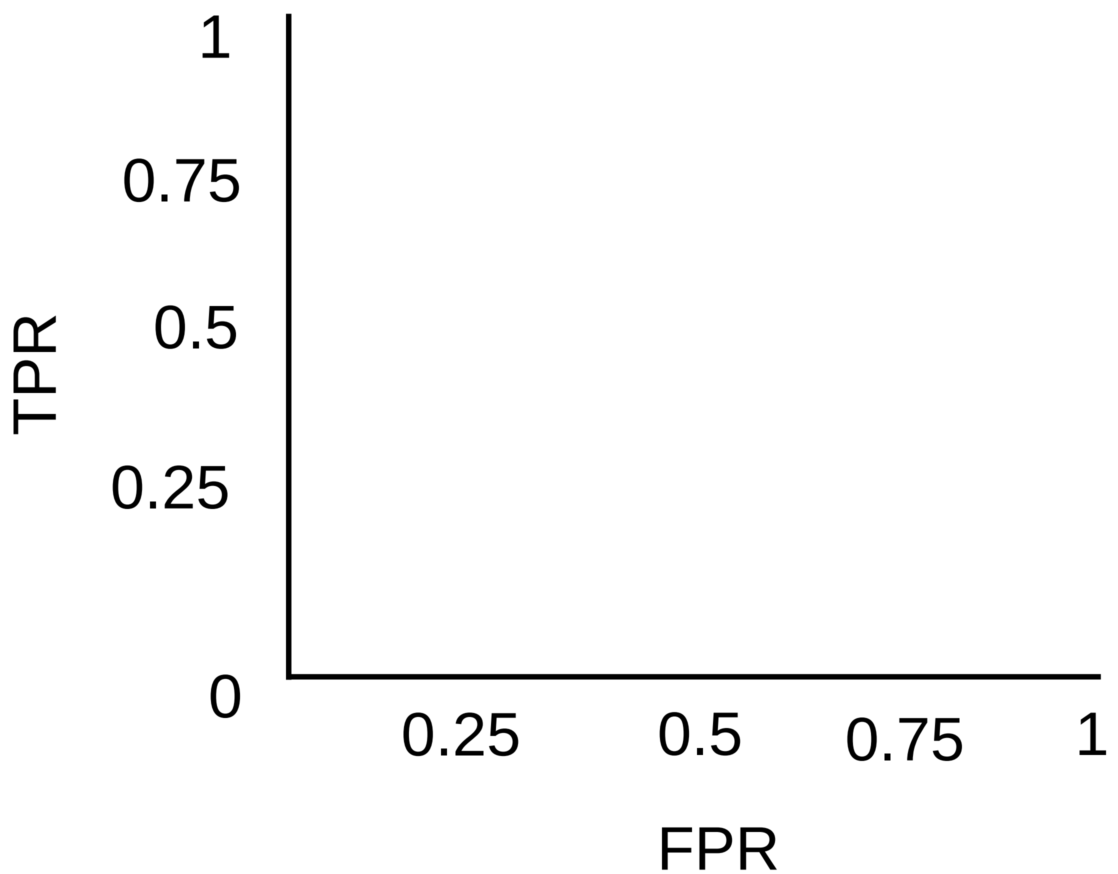
ROC Curve and AUC:

Another metric robust against imbalanced dataset

True Positive Rate (TPR) = Sensitivity = $\text{Recall} = \dfrac{tp}{tp + fn}$

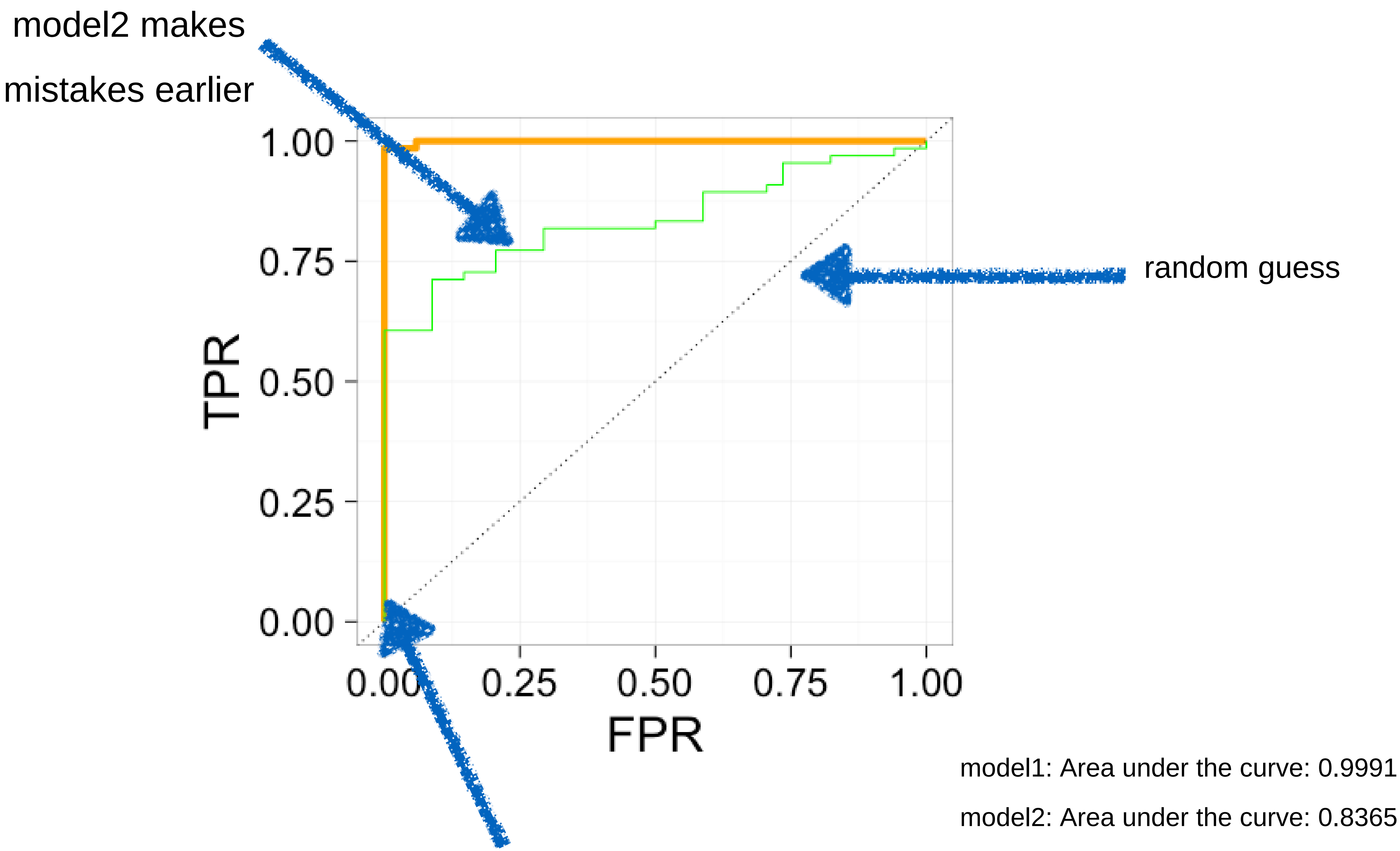
False Positive Rate (FPR) = *fp/(fp + tn)*

Let's plot the ROC curve for the following example



	client_id	churn	prediction
1	1	0	0.3
2	2	0	0.6
3	3	1	0.9
4	4	0	0.1
5	5	1	0.9

Intuition behind ROC



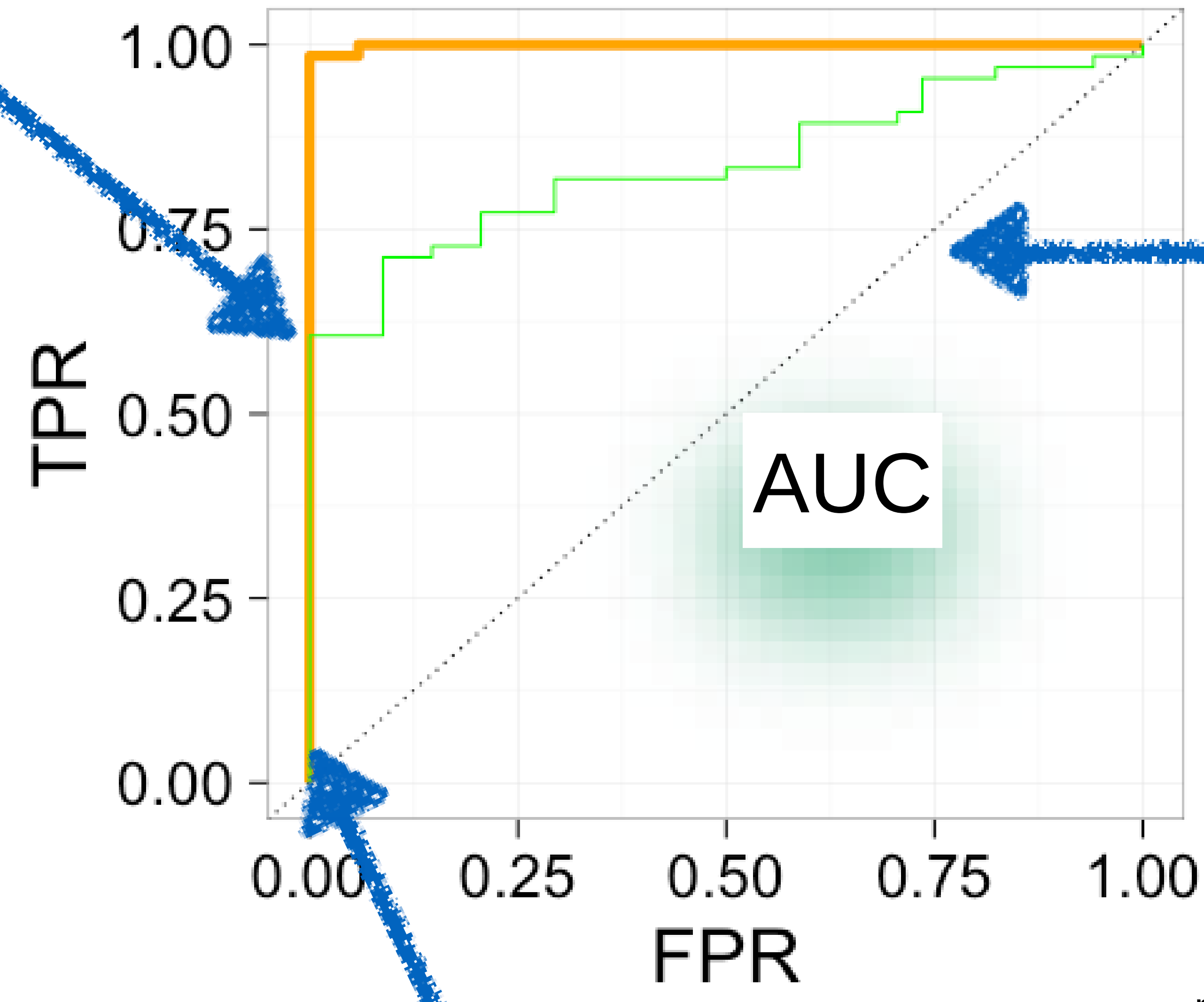
both our model, and the second one predict correctly

when they are confident in their scores

Intuition behind ROC

model2 makes

mistakes earlier



random guess

model1: Area under the curve: 0.9991

model2: Area under the curve: 0.8365

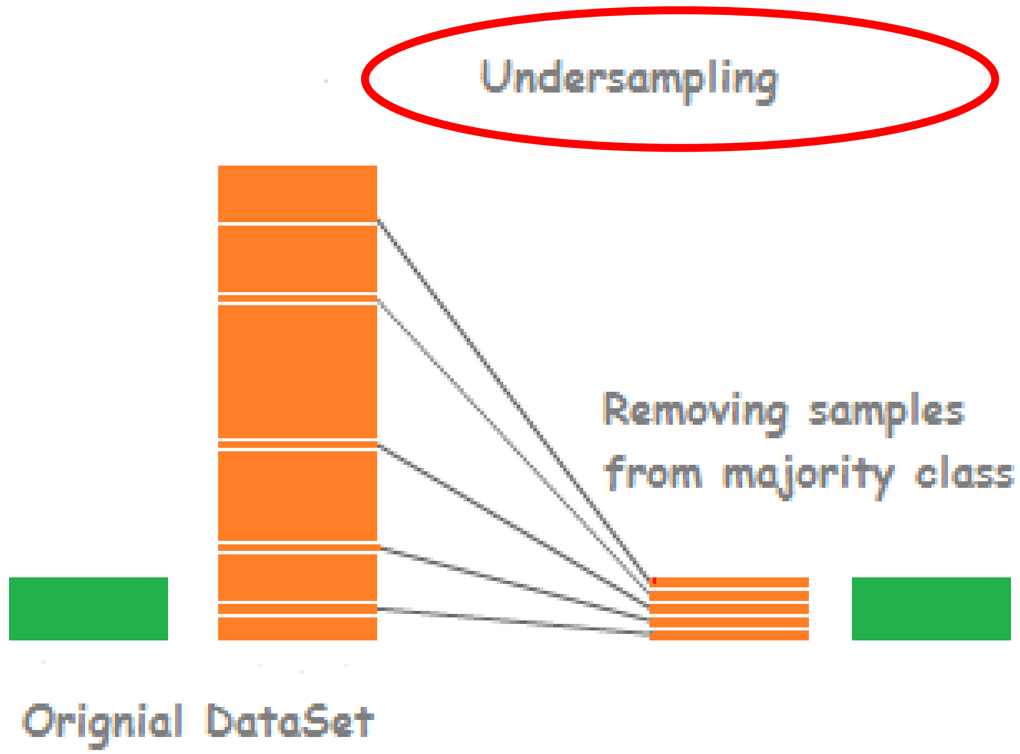
both our model, and the second one predict correctly

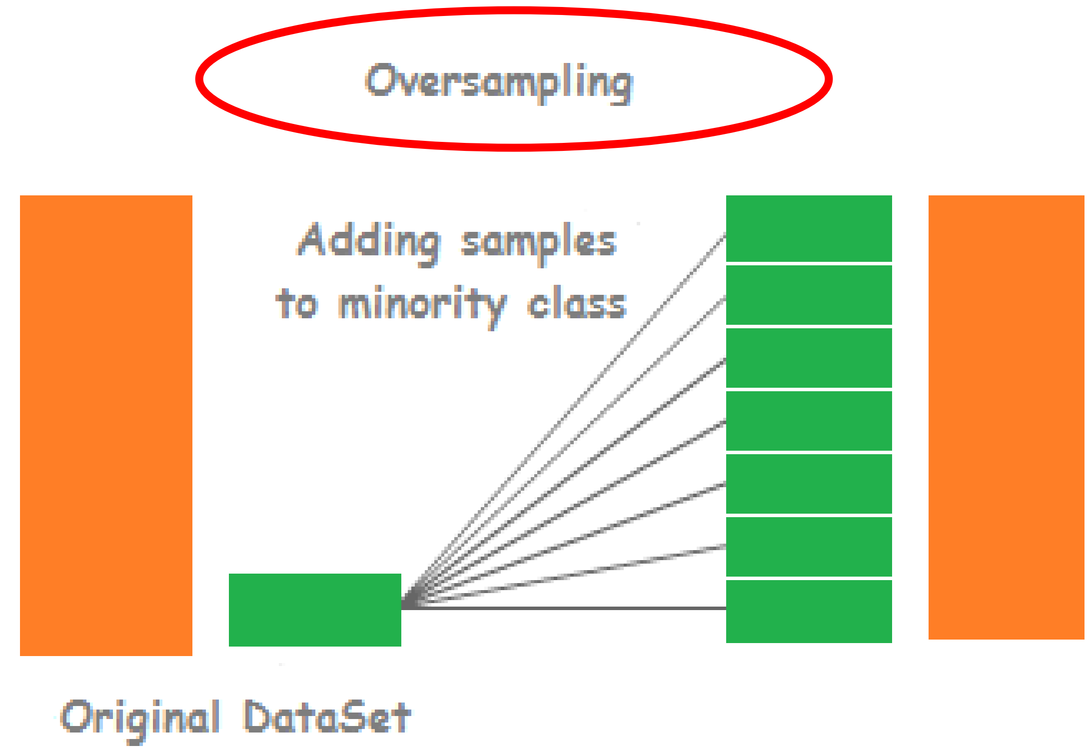
when they are confident in their scores

Dealing within class imbalance

- Use a quality measure that is as robust as possible
  - F-Score is OK in extreme cases (like the previous one)
  - Less robust in more subtle cases, e.g. 90-10 class imbalance
  - ROC and AUC (next) are better…
  - Area Under the Precision Recall Curve even better (AUPRC)

Dealing within class imbalance

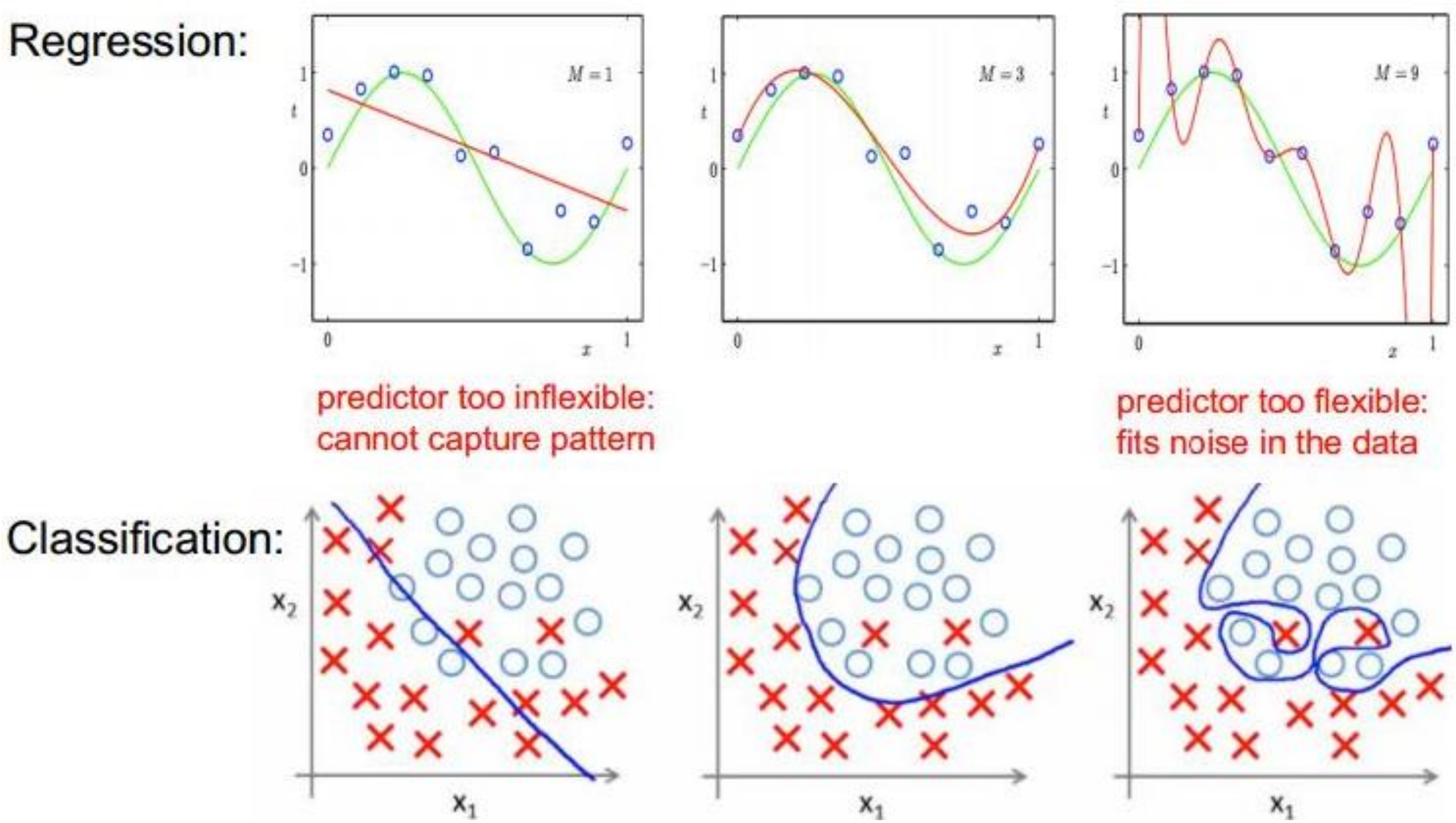




- Deliberately throw away <u>training data</u>
  - E.g. we have 6000 positive samples, 500 negative for training
  - We randomly select 500 of the 6000 positive samples

- We introduce (duplicating) minority class samples.
- Synthetic generation of minority class data, e.g. SMOTE

Pitfall 2: Under- and overfitting



http://www.inf.ed.ac.uk/teaching/courses/iaml/slides/eval-2x2.pdf

How to detect overfitting

- Compute AUC (or F-score) on the training set and on the testing set
- The larger the diff, the more likely overfitting has taken place
- E.g. AUC on training = 0.97, AUC on testing = 0.85

Detecting overfitting more rigorously (and rather avoiding it)

K-Folds cross validation

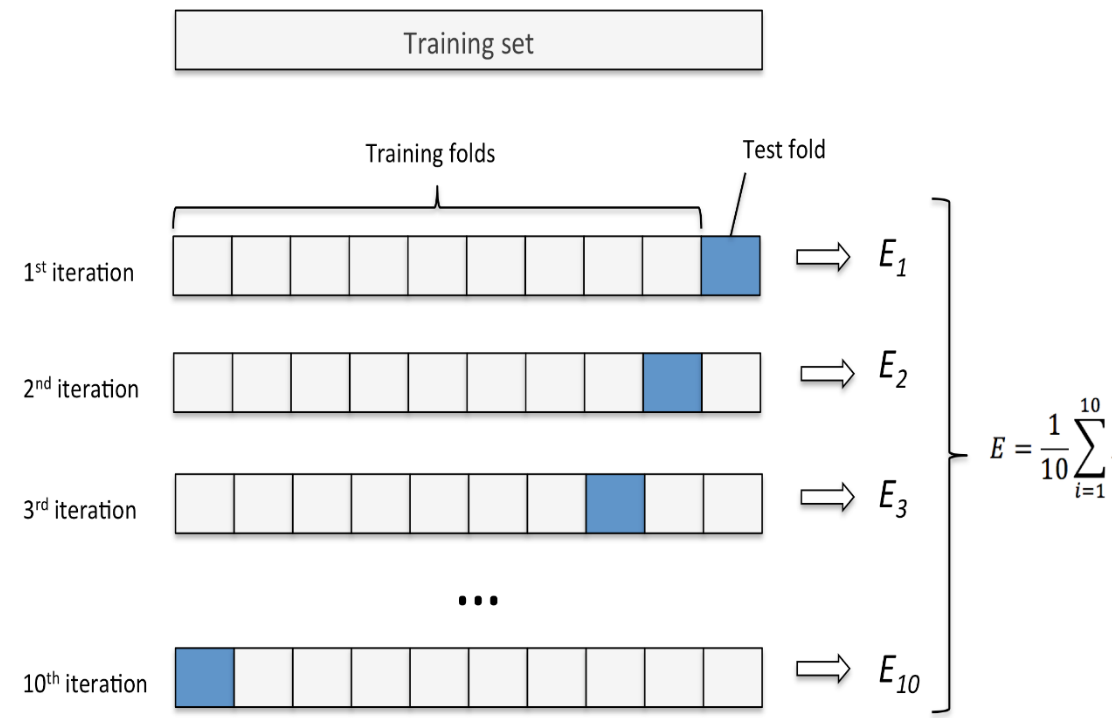
Step1: We split our data into K different splits.

Step2: We use (K-1) folds for training and Kth for testing.

Step3: We do the step 2 for every different K set.

Step4: Finally we do average of the model quality across all the models.

➔ The measure you get is more reflective of the likely performance in practice…



E = Output (Error or Accuracy) of your model

Source: http://karlrosaen.com/ml/learning-log/2016-06-20/

K-Fold cross validation (Variation)

Whole data

Part 1 Part 2 Part 3 Part 4 Part 5

Part 1 Part 2 Part 3 Part 4

K-Fold cross validation

Part 5

Testing

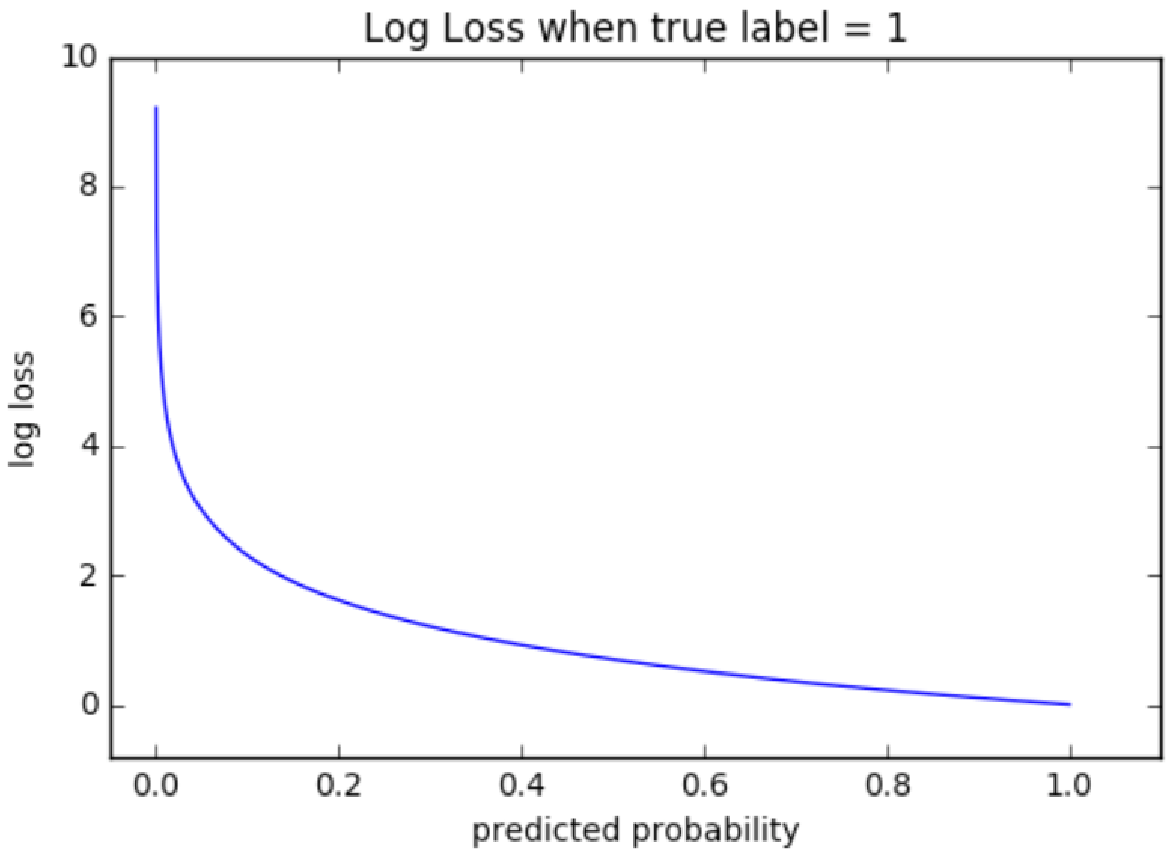
LogLoss function

Logarithmic loss measures the performance of a classification model where the prediction input is a probability value between 0 and 1

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log p_{ij}$$

- N is the number of samples or instances
- M is the number of possible labels,
- y_{ij} is a binary indicator of whether or not label j is the correct classification for instance i
- p_{ij} is the model probability of assigning label j to instance i.

2 class log function = $-\frac{1}{N}\sum_{i=1}^{N}[y_i \log p_i + (1-y_i)\log(1-p_i)].$

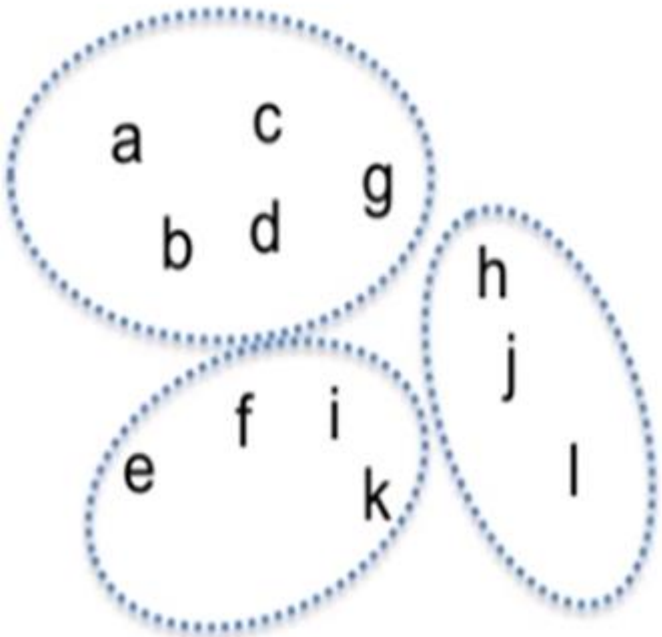


A perfect model would have a log loss of 0. Log loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high log loss

Source: http://wiki.fast.ai/index.php/Log_Loss

Measuring clustering Algorithms' quality

Step 1: Apply a clustering algorithm



Step 2: Sample set of pairs x_i, x_j

Step 3: Ask humans to check the pairs

Pair = Human decision	**Outcome Type**
a,b = Yes	True Positive (TP)
c,d = No	False Positive (FP)
e,h = Yes	False Negative (FN)
g,h = No	True Negative (TN)

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

Rand Index (RI) = Accuracy

Summary

- Classification algorithms
  - White Box: Logistic Regression, Decision Trees
  - Black Box: SVM, Random Forest, Boosting algorithms.
- Quality metrics of classification models: Accuracy, Precision, Recall, F1, AUC ROC, Logloss
- Pitfalls: class imbalance & overfitting