UNIVERSITAT DE BARCELONA

Introduction to Machine Learning

Master in Artificial Intelligence

UPC, UB, URV

UNIVERSITAT DE BARCELONA

Week 4

Course. Introduction to Machine Learning

Theory 4. Model Evaluation

Dr. Maria Salamó Llorente

maria.salamo@ub.edu

Dept. Mathematics and Informatics,

Faculty of Mathematics and Informatics,

University of Barcelona (UB)

Introduction to Machine Learning

Supervised Learning				Decision Learning Theory	**Unsupervised Learning**			**Machine Learning in practice**	
Non Linear Decision			Linear Decision	Basic concepts of Decision Learning Theory	Cluster Analysis	Factor Analysis	Visualization	Applications of ML	Beyond ML
Lazy Learning (K-NN, IBL, CBR)	Overfitting, model selection	Feature selection	Kernel Learning	Perceptron, SVM	Bias/Variance, VC dimension, Practical advice of how to use learning algorithms	K-Means, Fuzzy C-means, EM	PCA, ICA	Self Organized Maps (SOM) , Multi-Dimensional Scaling	RecSys	Bias and Fairness in ML

Contents

Introduction to model evaluation

Why do we evaluate a model's performance?

1. To know how well it works

- Reporting results
- Making scientific/business decisions
- Can we use it in real-life?

2. To compare models

- For choosing which models are best
- For tuning a given model's parameters

How do we evaluate a model's performance?

Definitions

- ***Model Evaluation*** is the process of assessing a property or properties of a model
  - Evaluation metrics
  - Evaluation techniques
- ***Model Selection*** is the process of choosing among many candidate models for a predictive modeling problem
  - Probabilistic measures
  - Resampling methods

How do we evaluate a model's performance?

- ***Model Evaluation*** is the process of assessing a property or properties of a model
  - It is often valuable to assess the efficacy of a model that has been learned
  - Such assessment is frequently relative– an evaluation of which of several alternative models is best suited to a specific application
  - There are many **evaluation metrics** by which a model may be assessed
    - Measures relating to *predictive efficacy*
    - Metrics based on *ROC analysis*

How do we evaluate a model's performance?

- ***Model Evaluation*** is the process of assessing a property or properties of a model
  - A learning algorithm must interpolate appropriate predictions of regions of the instance space that are not included in the training data
  - Algorithm **evaluation techniques** are designed to provide more reliable estimates of the accuracy of the models learned by an algorithm than would be obtained by assessing them on the training data

Evaluation metrics

Performance metrics

- Precision & Recall (and f1 score)
  - **Precision**: What percent of our predictions are accurate?
  - **Recall**: How many of the accurate predictions did we capture
  - **F1 score**: A single number that combines the two values above. Good for ranking/sorting…

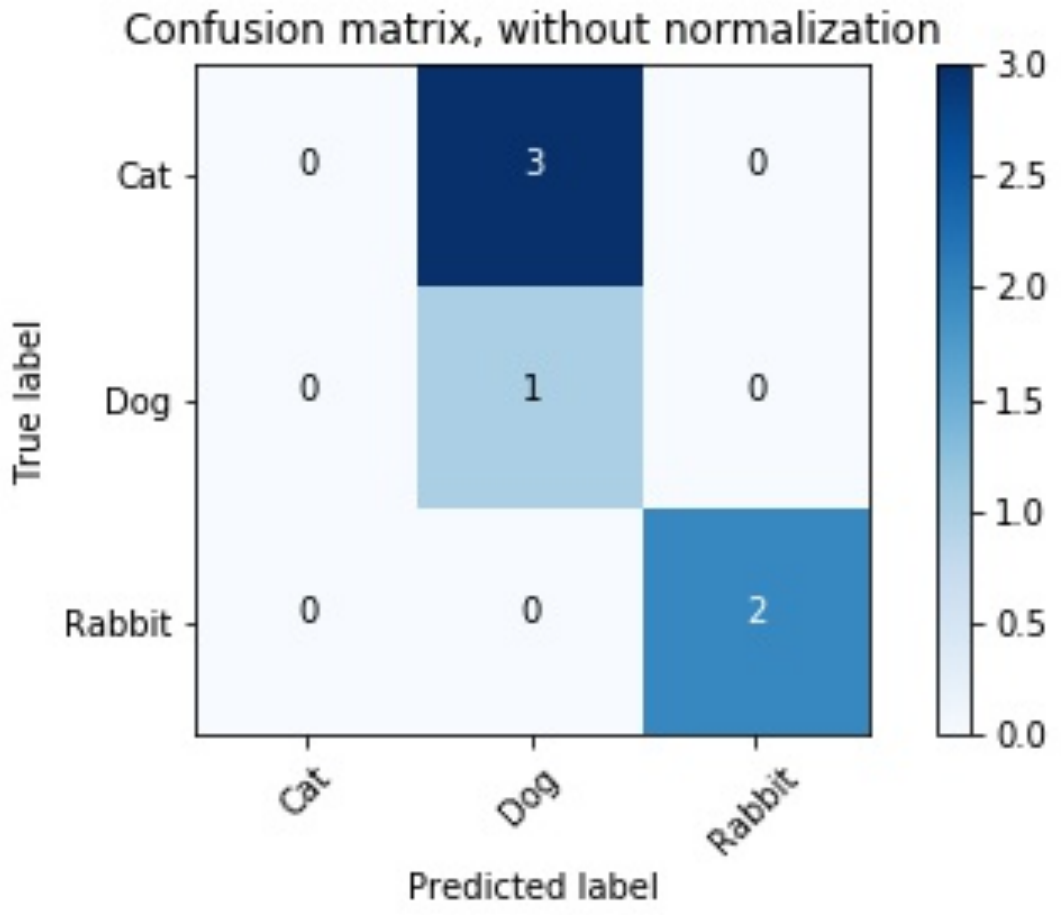
$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

  - Specificity/sensitivity – very similar to precision/recall. Often used in medicine.

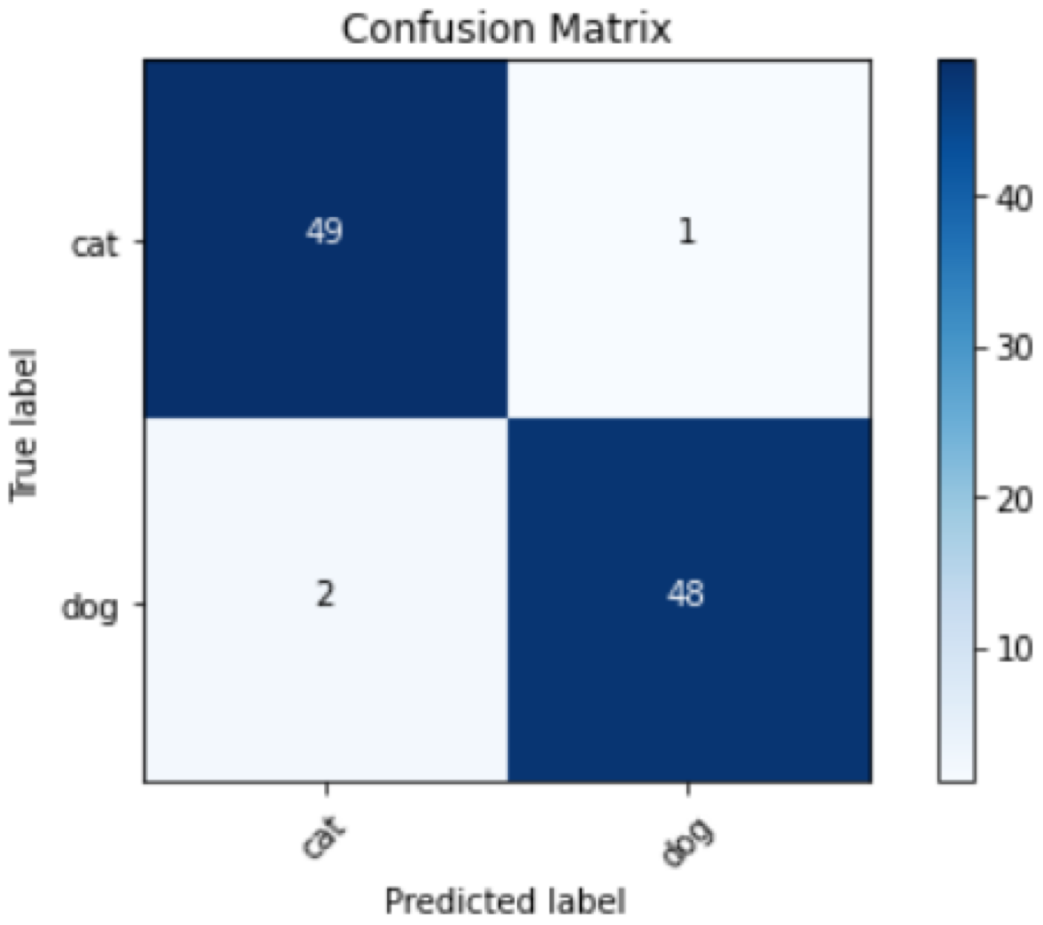
- Precision at N
  - How many accurate examples did we capture in our top N ranked examples. This is often used in information (document) retrieval

The confusion matrix

- How can we understand what types of mistakes a learned model makes?



Confusion matrix for 2-class problems



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

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

$$ErrorRate = 1 - Accuracy$$

Is accuracy an adequate measure?

- Accuracy may not be useful measure in cases where there is a large class skew
  - Is 98% accuracy good if 97% of the instances are negative?
- There are differential misclassification costs (i.e., getting a positive wrong costs more than getting a negative wrong)
  - For example, in a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
- We are more interested in a subset of high-confidence predictions

Other accuracy metrics

- Purpose: Evaluating binary classification
- Many performance terms: Precision, recall, sensitivity, specificity, true negative rate, true positive rate, PPV, NPV, type 1 error, type 2 error

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	**Specificity** = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$	

**Even this picture is not complete (precision/recall on next slide)*

Other accuracy metrics

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

$$ErrorRate = 1 - Accuracy$$

$$TP\ Rate = \frac{TP}{TP + FN}$$

$$FP\ Rate = \frac{FP}{FP + TN}$$

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

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{(1 + \beta)^2 \cdot TP}{\beta^2 \cdot TP + FP + FN}$$

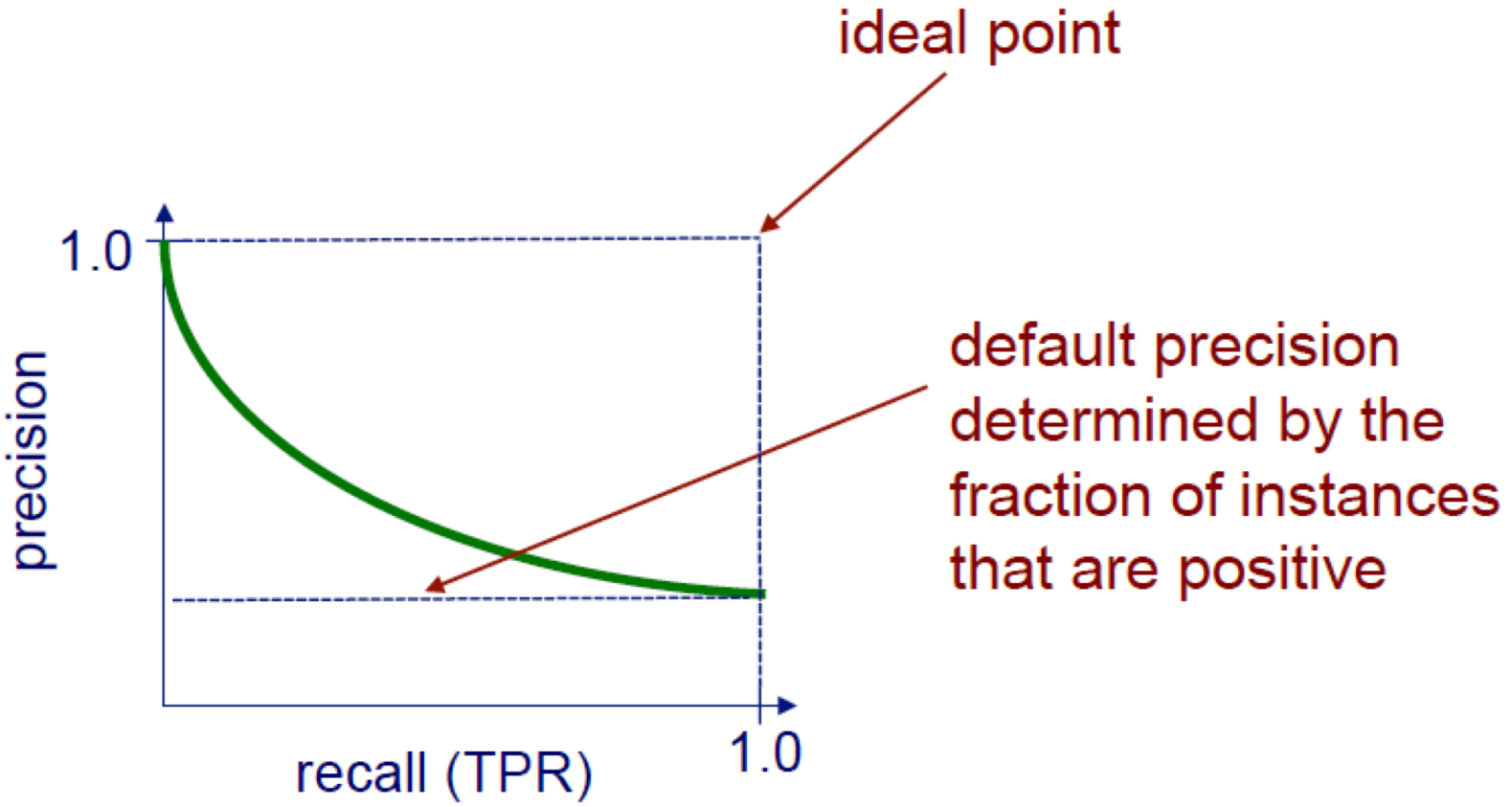
β=1, usually

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

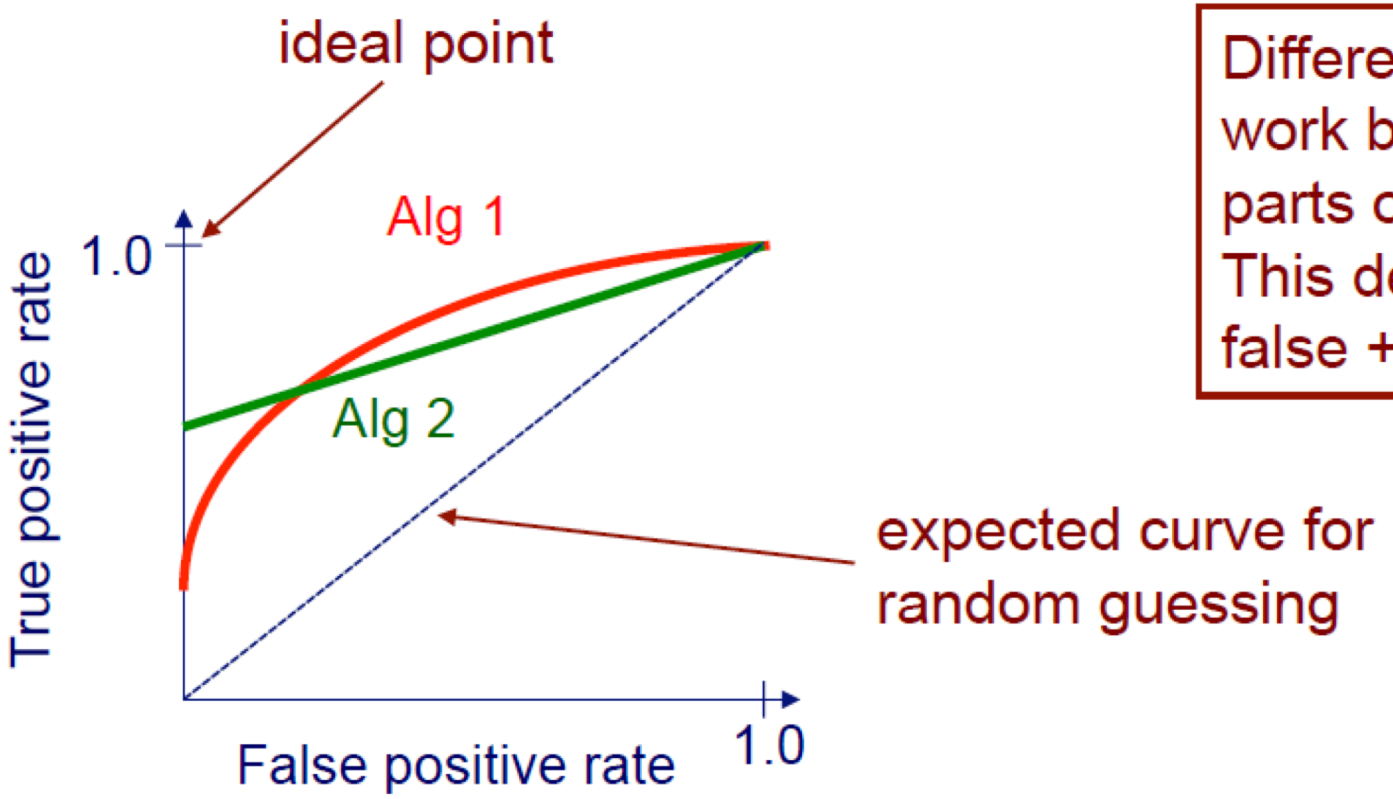
Precision/recall curves

A ***precision/recall*** (PR) curve plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied



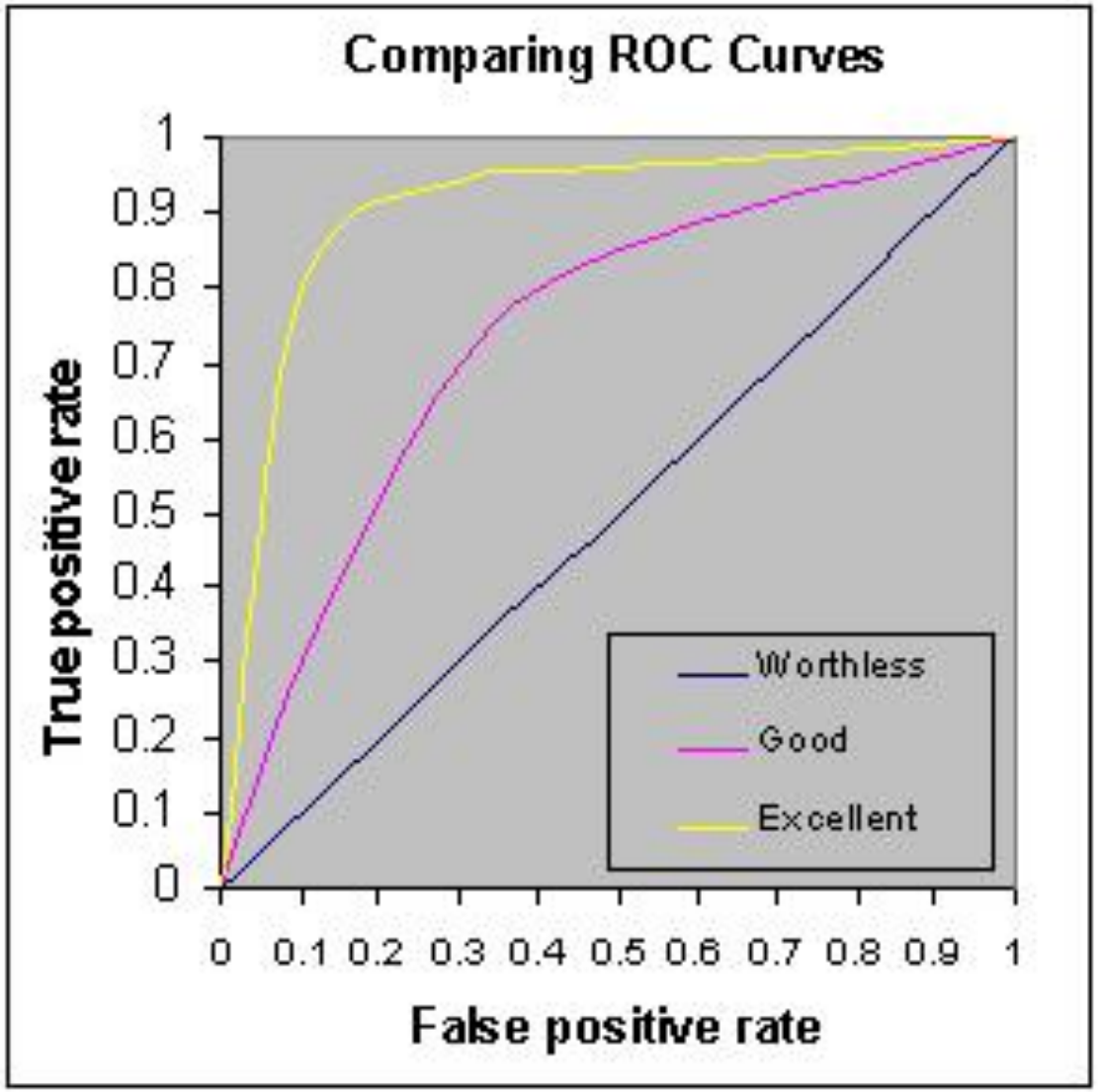
ROC Curves

A ***Receiver Operating Characteristic*** (ROC) curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied



Different methods can work better in different parts of ROC space. This depends on cost of false + vs. false -

ROC curve example



Comments on ROC and PR curves

- Both
  - Allow predictive performance to be assessed at various levels of confidence
  - Assume binary classification tasks
  - Sometimes summarized by calculating area under the curve
- ROC curves
  - Insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
  - Can identify optimal classification thresholds for tasks with differential misclassification costs
- Precision/recall curves
  - Show the fraction of predictions that are false positives
  - Well suited for tasks with lots of negative instances

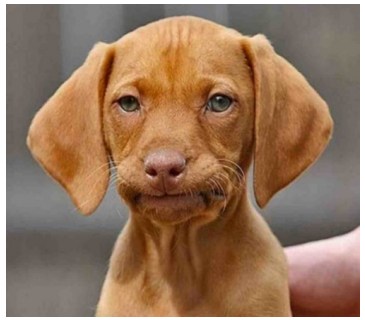
Evaluation techniques

The basic rule in model validation

Don’t test your model on data it’s been trained with!

Testing on 'realistic data'

- Can you evaluate the learned model on real-life/real use case data? → **Ideal**
- Sometimes you can only evaluate on simulated data
- Testing on biased datasets can lead to inaccurate estimation of performance



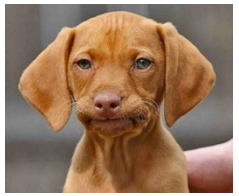
I thought our results were better

Random selection of data vs bias

- **Unbiased data**
  - Data that represents the diversity in our target distribution of data on which we want to make predictions (i.e., it is a random, representative sample. This is a good scenario)

- **Biased data**
  - Data that does not accurately reflect our target population
  - Example – we only asked people with glasses who they will vote for, instead of asking a random sample of the population
  - Take home message – training on biased data can lead to inaccuracies
  - There are methods to correct bias

Holdout test set – considerations

- Can the data be split randomly?
  - If not, it may lead to underfitting (remember me )
  - Test set and training set should be interchangeable, except for data size

- Do we have enough data?
  - Example: Guess if a car is American or Japanese based on 50 examples?

    Will the test set be big enough to accurately report our model performance?

Holdout test set: The naïve approach

The simplest evaluation strategy: Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in two sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$$

Training

Use $\mathcal{D}_{train}$ to learn the model.

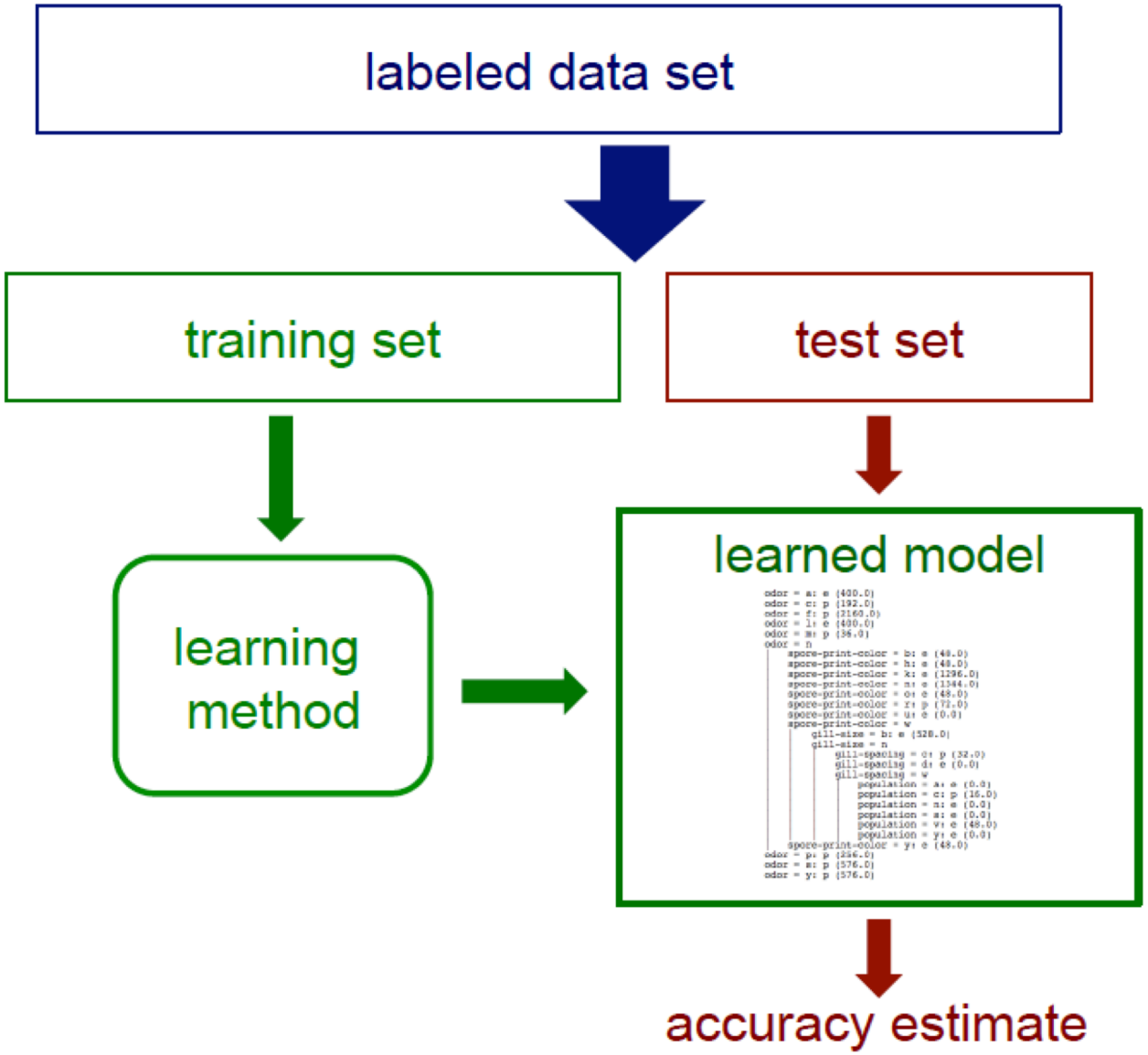
Evaluation

Use $\mathcal{D}_{test}$ to compute the performance of the method.

Exploitation

Apply the model to new data and **hope** it predicts correctly.

Holdout test set: The naïve approach



The three-way split

Simulate exploitation data.

We are given a dataset $\mathcal{D}$ and it is divided in three sets

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{validation} \cup \mathcal{D}_{test}$$

Training and model selection

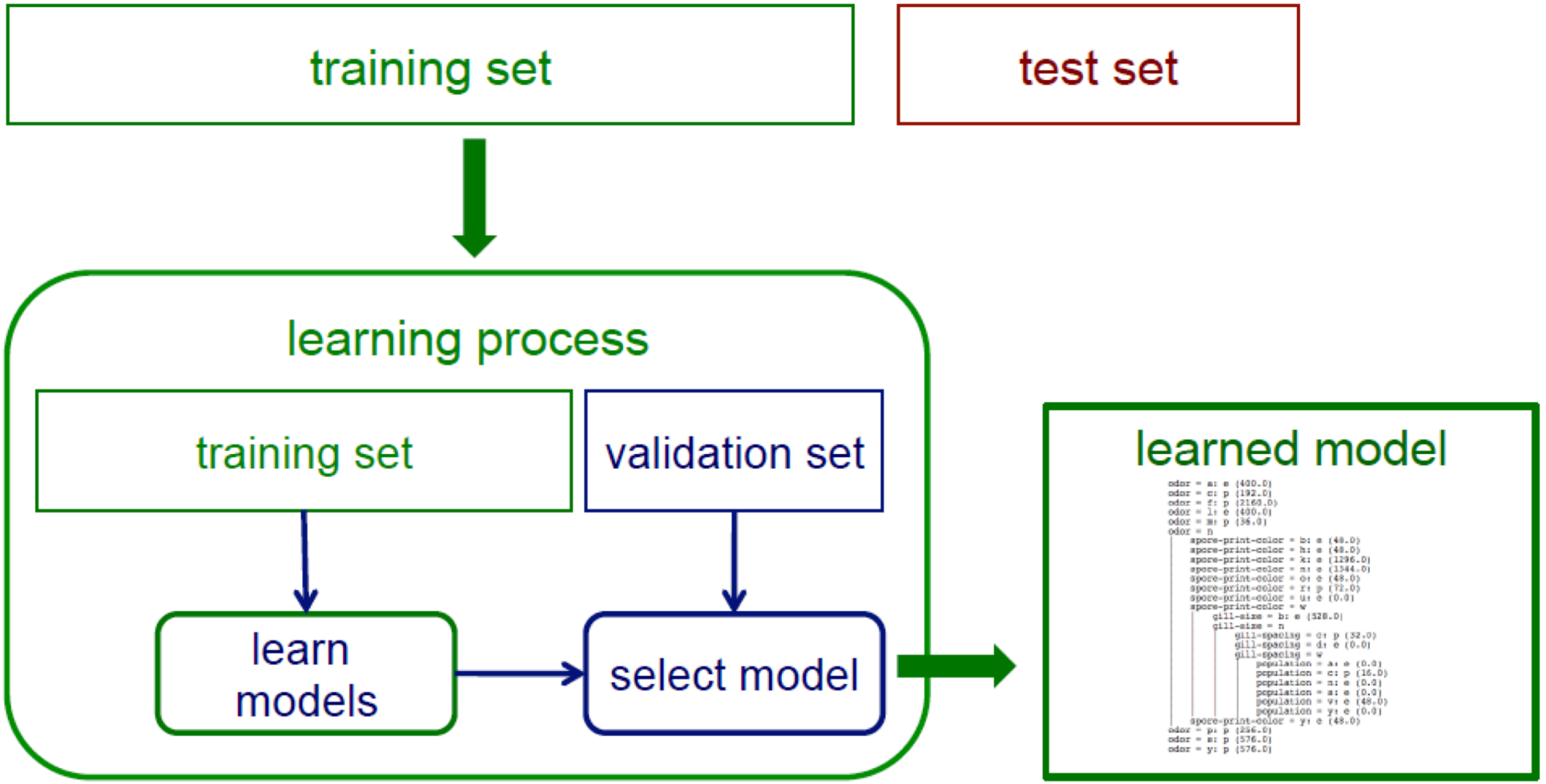
Use $\mathcal{D}_{train}$ to learn both models. Use $\mathcal{D}_{validation}$ to decide which of the two models better adapts to our problem.

Evaluation

Use $\mathcal{D}_{test}$ to compute the performance of the selected method.

Apply the selected model to new data and **hope** it predicts correctly.

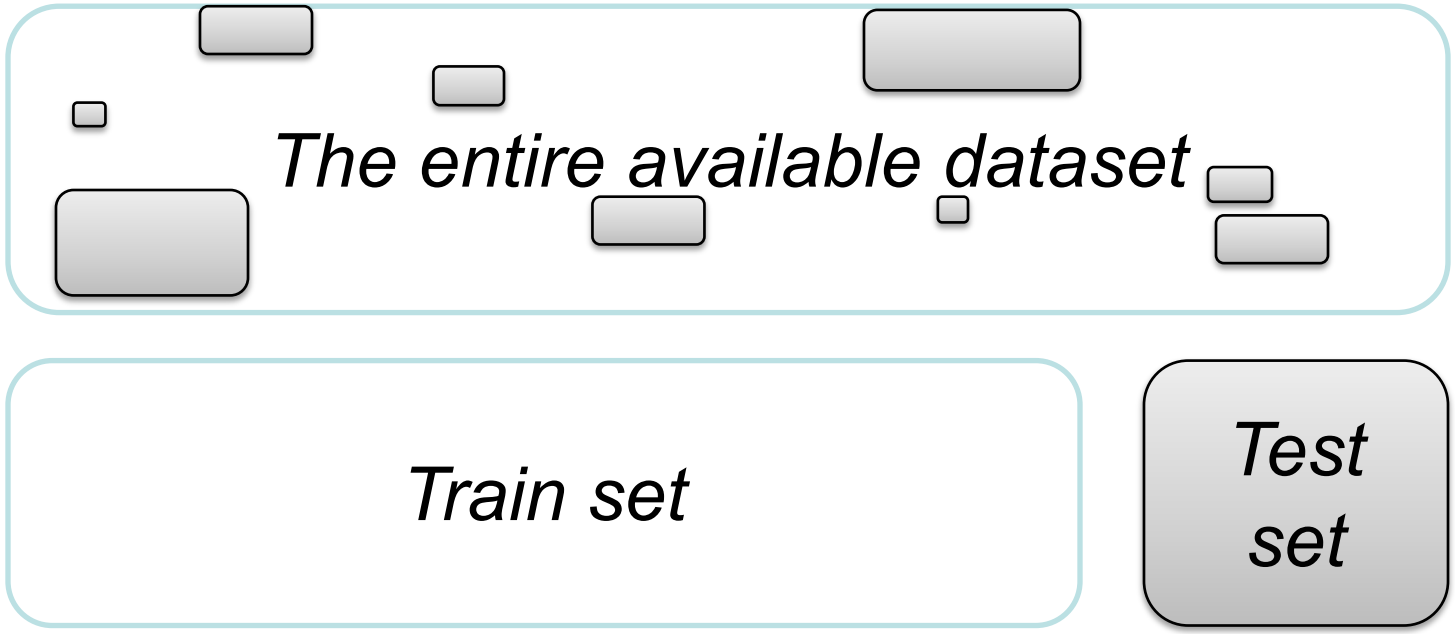
The three-way split



Partition training data into separate training/validation sets

How to perform the split?

- How many examples in each data set?
  - Training: Typically 60-80% of data
  - Test set: Typically 20-30% of your trained set
  - Validation set: Often 20% of data
- Examples
  - 3 way: Training: 60%, CV: 20%, Test: 20%
  - 2 ways: Training 70%, Test: 30%



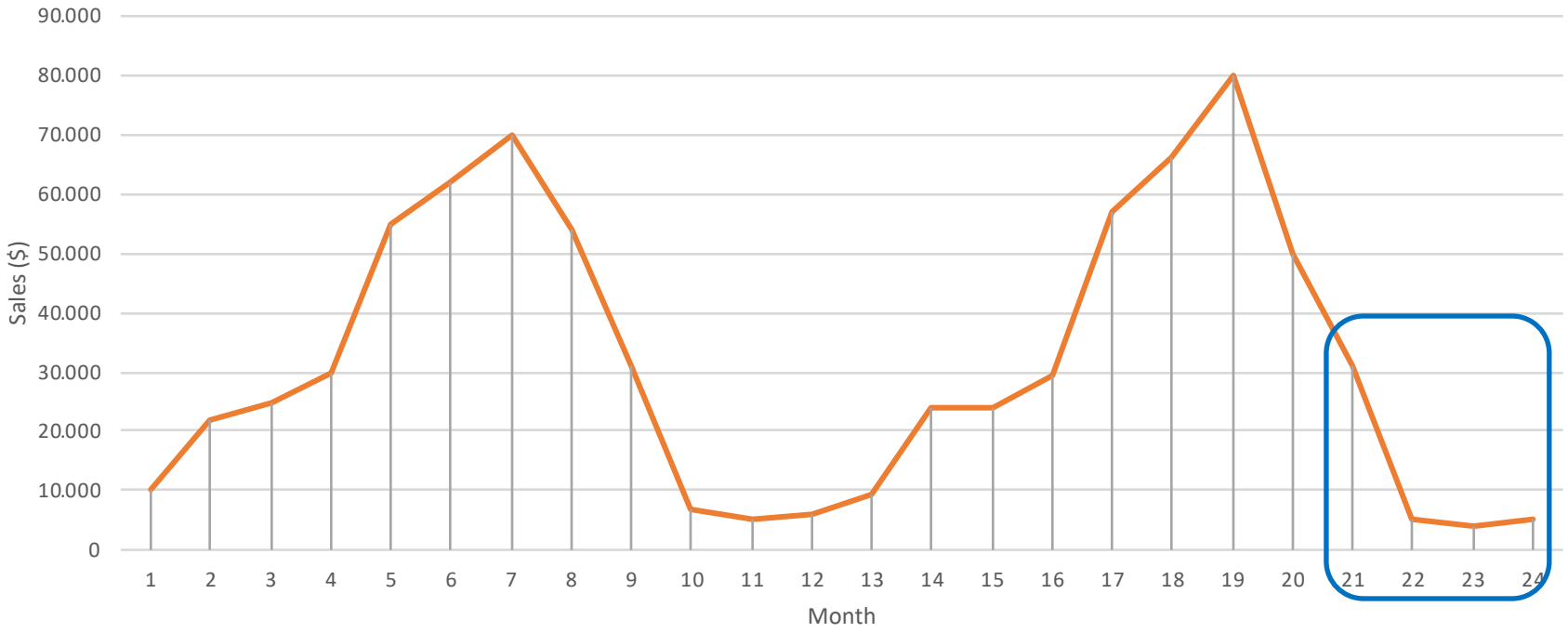
How to perform the split?

- Time based – If time is important

Can you use Jul 2011 stock price to try and predict Apr 2011 stock price?

How to perform the split?

- Some datasets are effected by seasonality



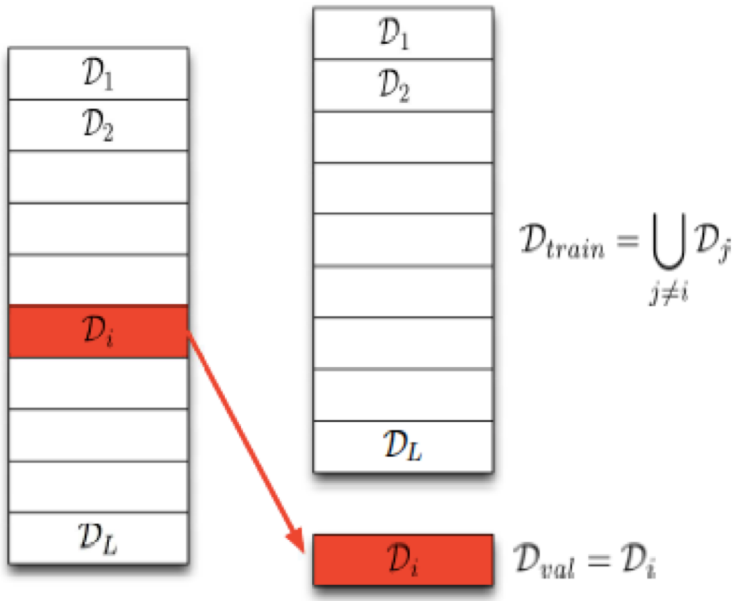
Is it a good test set?

Cross-validation

- Popular method for validating models
- Cases when it is often used
  - When a hold-out plan is difficult to design or was not designed at the beginning
  - When the dataset is small, and you want to use all the data for good model training

Cross-validation

1. **for** $i = 1..L$:



1.
2. Train the model on $\mathcal{D}_{train}$ and evaluate $e_{val}(\mathcal{D}_i)$

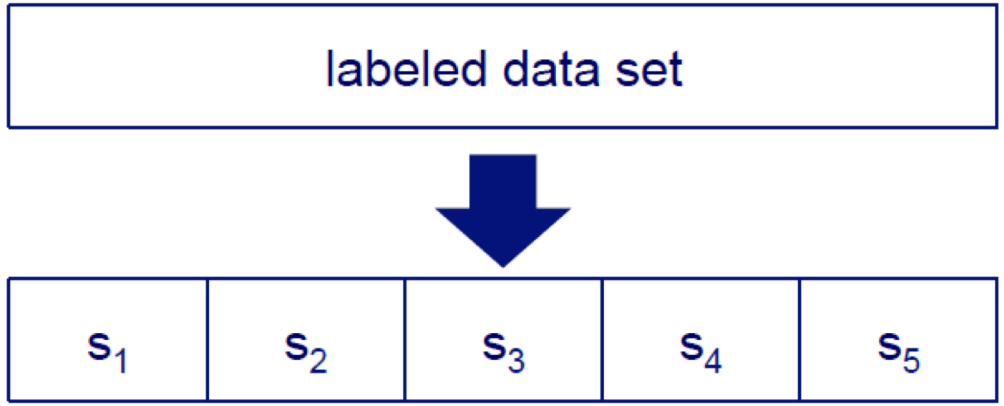
endfor

2. $E_{val} = \sum_{i=1}^{L} e_{val}(\mathcal{D}_i)$

if $L = N \implies K = 1$ the process is called **Leave-one-out**. Otherwise, the method is called L-**fold cross-validation**.

Cross-validation

partition data into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Cross-validation example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	S_2 S_3 S_4 S_5	S_1	11 / 20
2	S_1 S_3 S_4 S_5	S_2	17 / 20
3	S_1 S_2 S_4 S_5	S_3	16 / 20
4	S_1 S_2 S_3 S_5	S_4	13 / 20
5	S_1 S_2 S_3 S_4	S_5	16 / 20

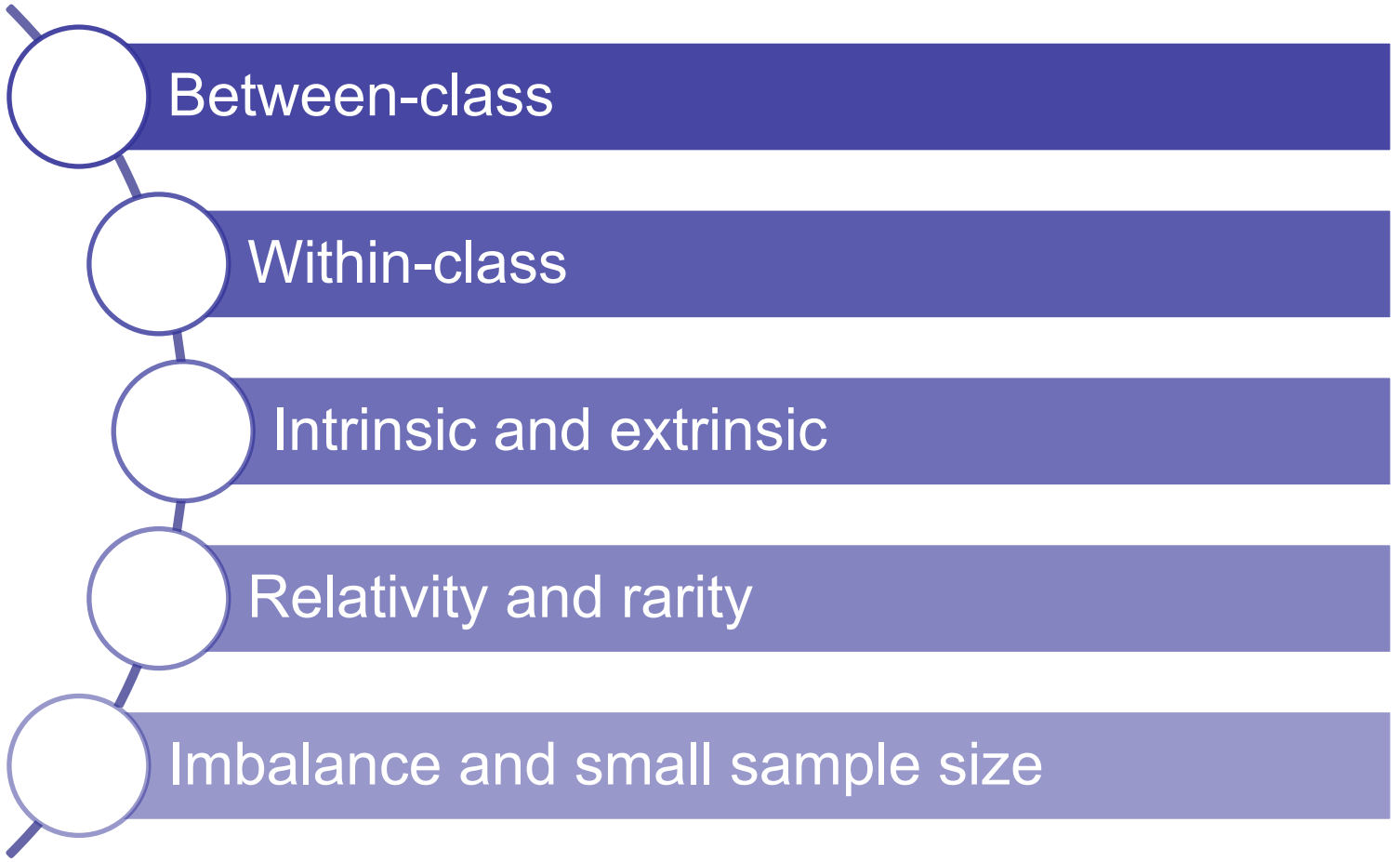
accuracy = 73/100 = 73%

Cross - Validation

How many folds are needed?

- Common choice: 5-fold or 10-fold cross validation (probably since these numbers sound nice)
- Large datasets: even a 3-Fold cross validation will be quite accurate
- Smaller datasets: we will prefer a bigger *L*
- Leave-One-Out approach (*L*=*N*). In this case the number of folds *L* is equal to the number of examples N. Used for smaller datasets

Imbalance is everywhere



The Nature of Imbalance Learning

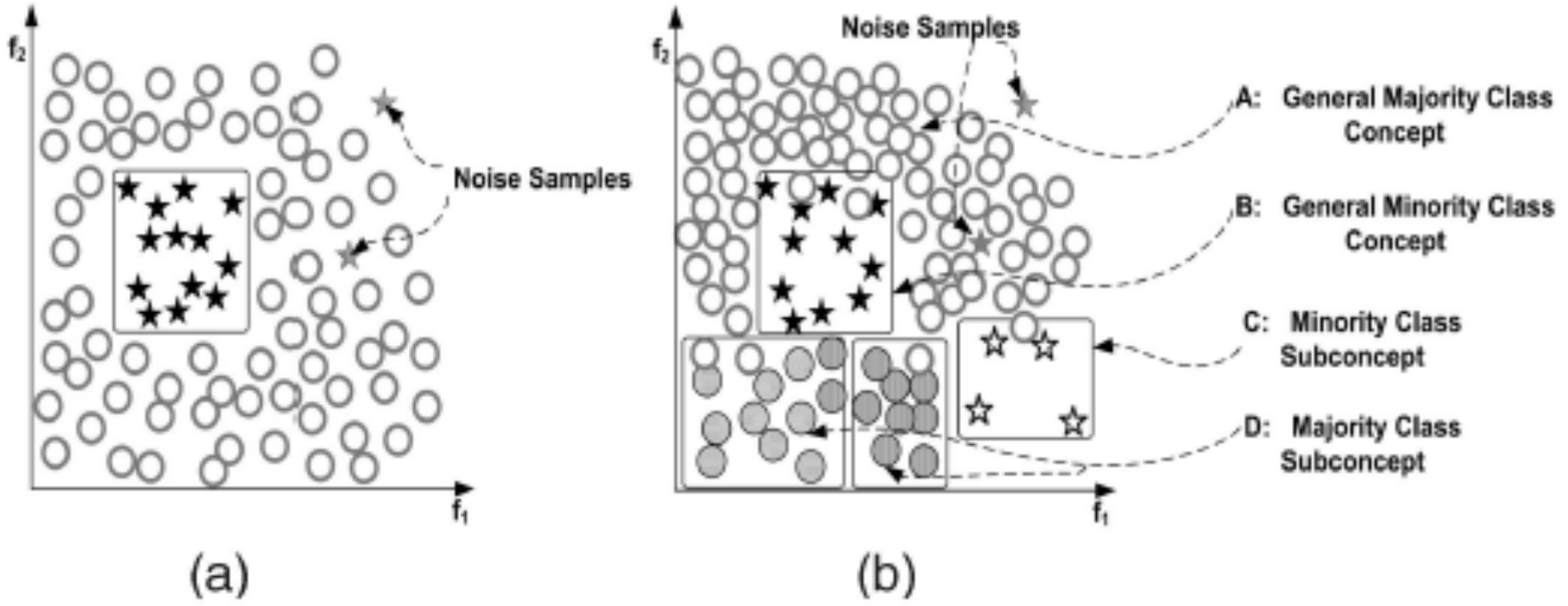
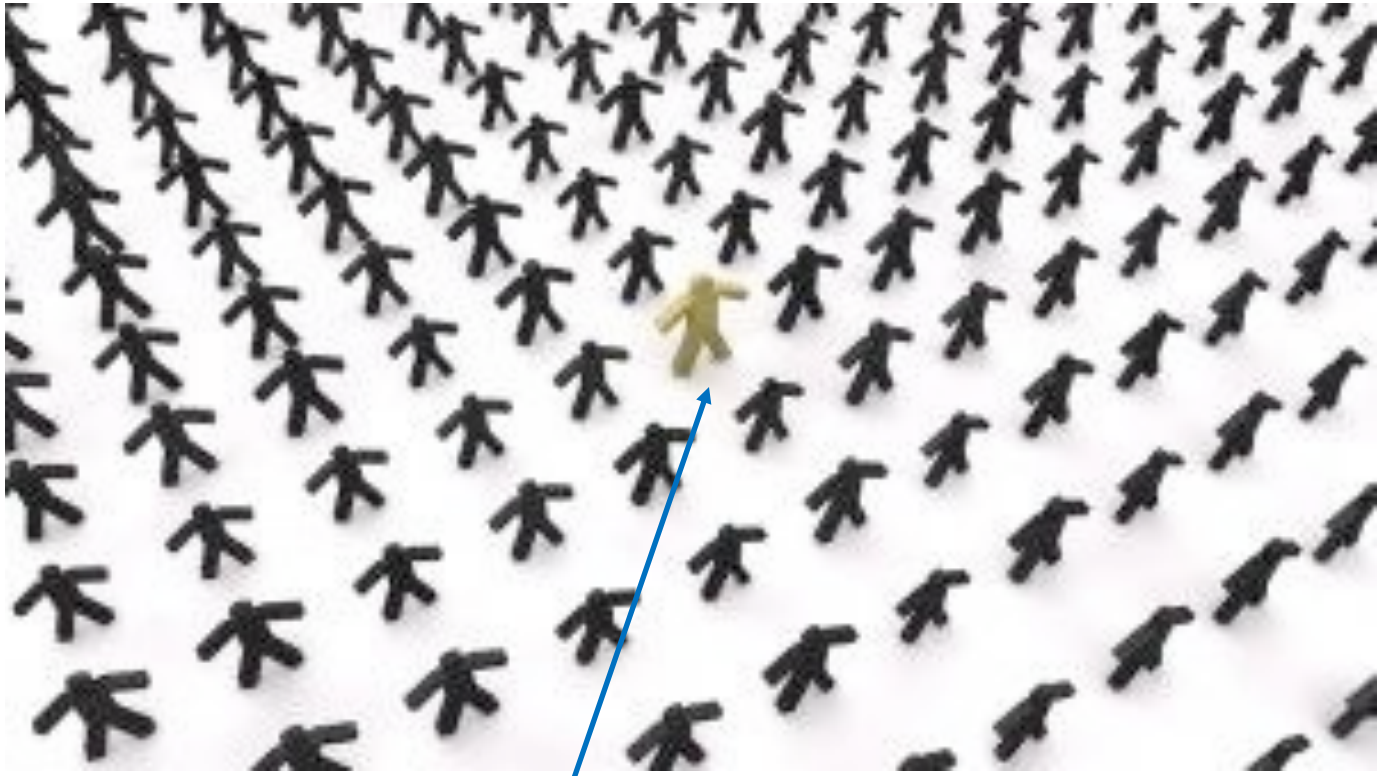


Fig. 2. (a) A data set with a between-class imbalance. (b) A high-complexity data set with both between-class and within-class imbalances, multiple concepts, overlapping, noise, and lack of representative data.

Dealing with imbalanced classes



Class 1: Short people

Class 2: Tall people

Can we build a classifier that can find the needles in the haystack?

Imbalanced Class Sizes

- Sometimes, classes have very unequal frequency
  - Cancer diagnosis: 99% healthy, 1% disease
  - eCommerce: 90% don't buy, 10% buy
  - Security: >99.99% of Americans are not terrorists
- Similar situation with multiple classes
- This creates problems for training and evaluating a model

Evaluating a model with imbalanced data

- Example: 99% of people do not have cancer
- If we simply create a 'trivial classifier' – predict that nobody has cancer, then 99% of our predictions are correct! Bad news! – we incorrectly predict nobody has cancer
- If we make only a 1% mistake on healthy patients, and accurately find all cancer patients
  - Then ~50% of people that we tell have cancer are actually healthy!

Solutions to imbalanced learning

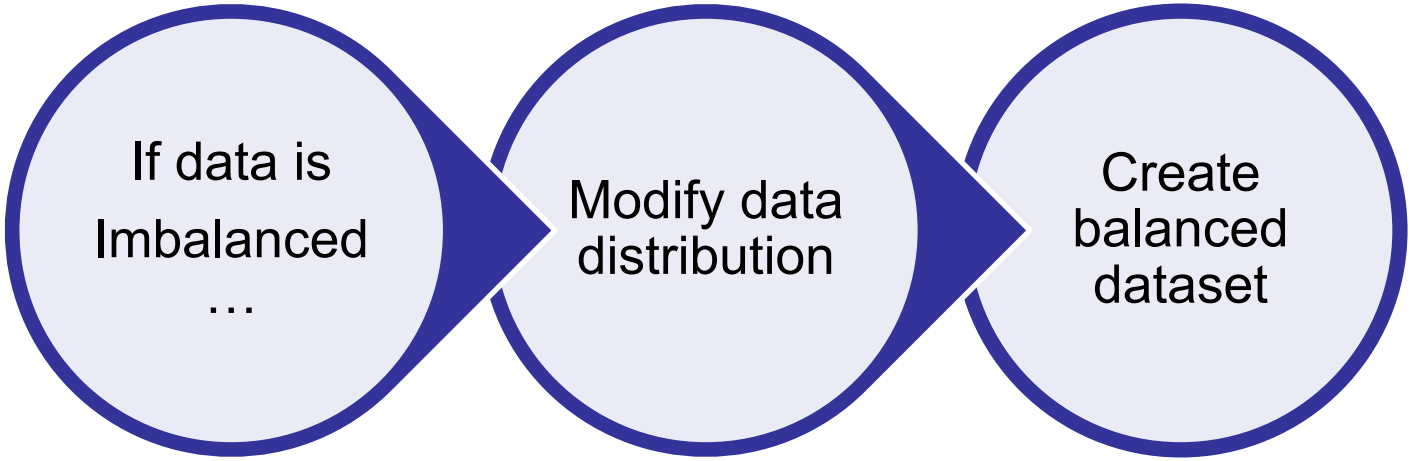
- Sampling methods
- Cost-sensitive methods
- Kernel and Active Learning methods

***Note**: it is out of the scope of this course to detail all of them.*
In this course, we will concentrate on defining basic sampling methods

TO READ: **OPTIONAL**

H. He and E. A. Garcia, "Learning from Imbalanced Data," IEEE Trans. Knowledge and Data Engineering, vol. 21, issue 9, pp. 1263-1284, 2009

Sampling methods



Create balance through sampling

Balancing the classes

- Learning with imbalanced datasets is often done by balancing the classes
- **Important: Estimate the final results using an imbalanced held-out (test) set**
- How to create a "balanced" set
  - Treat the majority class → **undersampling**
    - Down-sample
    - Bootstrap (repeat downsampling with replacement)
  - Treat the small class → **oversampling**
    - Up-Sample the small class
    - Assign larger weights to the minority class samples

Random Sampling

__S__: training data set; __S_{min}__: set of minority class samples, __S_{maj}__: set of majority class samples; __E__: generated samples

Random oversampling

- Expand the minority
- $|S'_{min}| \leftarrow |S_{min}| + |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| + |E|$
- Overfitting due to multiple "tied" instances

Random undersampling

- Shrink the majority
- $|S'_{maj}| \leftarrow |S_{maj}| - |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| - |E|$
- Loss of important concepts

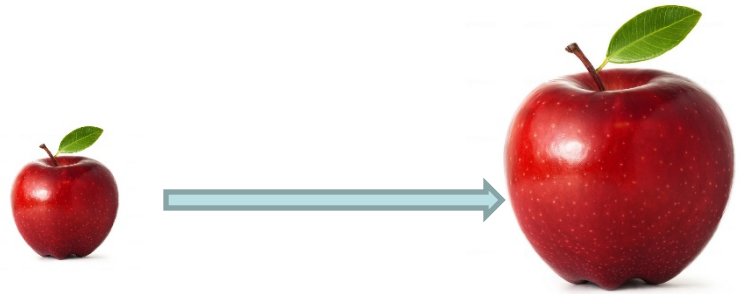
Down-sample and Up-sample

- Down-sample → undersampling
  - Sample (randomly choose) a **random subset of points** in the majority class

- Up-sample → oversampling:
  - Repeat minority points
  - Synthetic Minority Oversampling Technique (**SMOTE**)

Reweighting

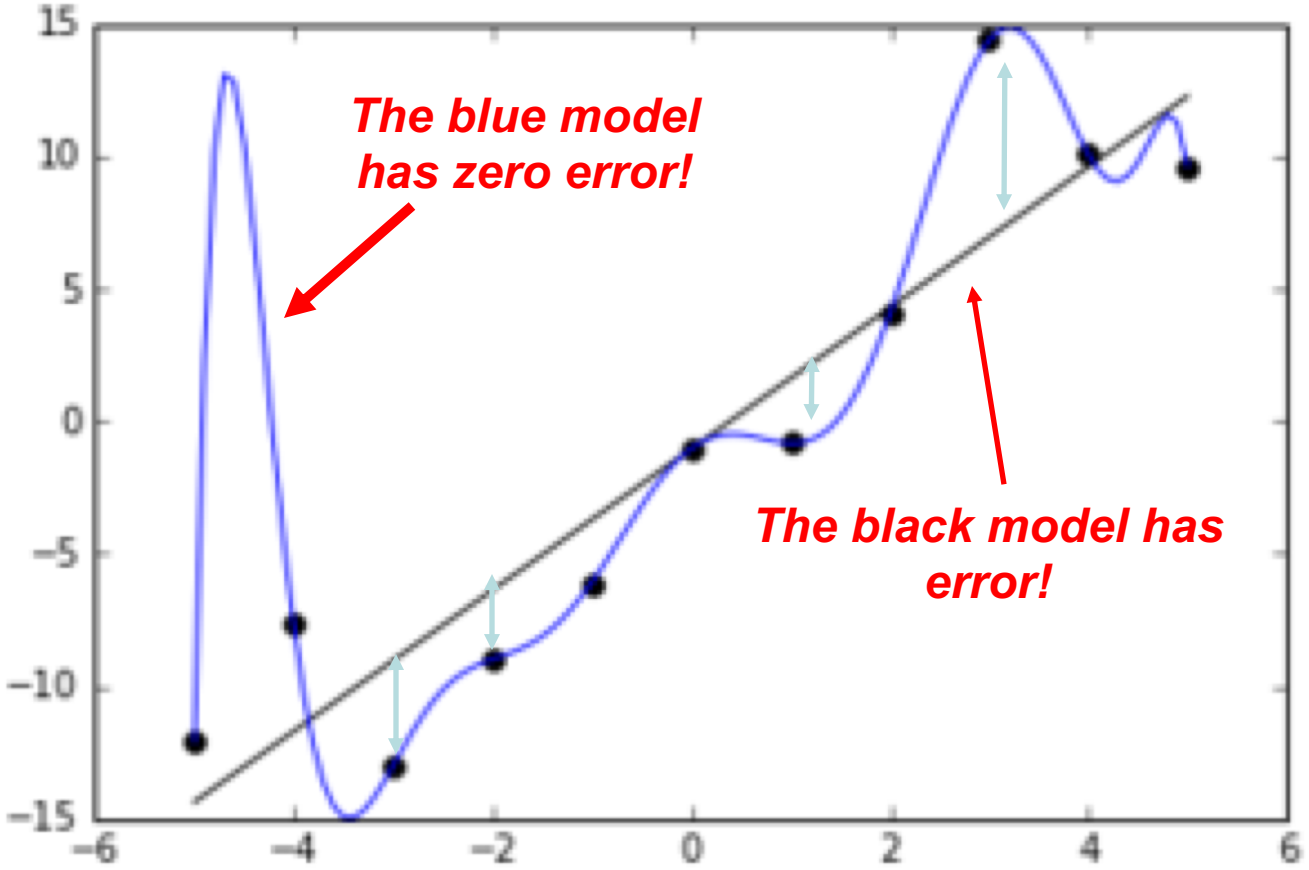
- Idea: Assign larger weights to samples from the smaller class



- A commonly used weighting scheme is linearly by class size: $w_c = \frac{n}{n_c}$
  - Where n_c is the size of the class c and $n = \sum_c n_c$ is the total sample size

Overfitting and Underfitting

Overfitting and underfitting



So which model is better : blue or black?

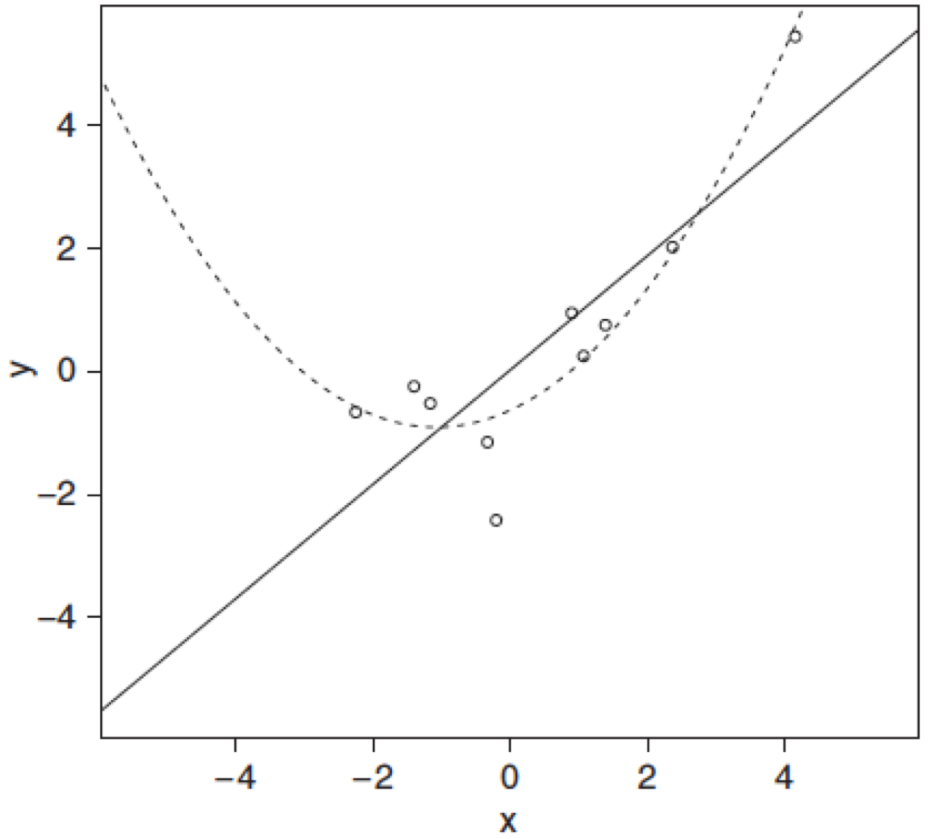
Overfitting

- A model overfits the training data when it describes features that arise from noise or variance in the data, rather than the underlying distribution from which the data were drawn

- Overfitting usually leads to loss of accuracy on out-of-sample data
- Approaches to avoid overfitting include:
  - Using low variance learners
  - Minimum description length technique
  - Pruning
  - Regularization
  - Stopping criteria

Overfitting example

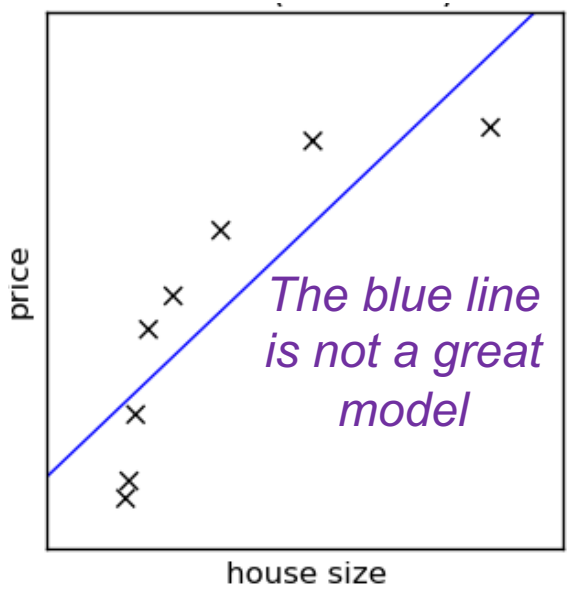
- The figure illustrates overfitting
- Linear and polynomial models fitted to random data drawn from a distribution for which the linear model is a better fit



Underfitting (high Bias)

- In underfitting, the training error and test error are high

- What does too simple mean?
  - Too few features
  - Use of features is not ‘complex’



- Examples
  - Can you predict well if an email is spam using only the word ‘free’?

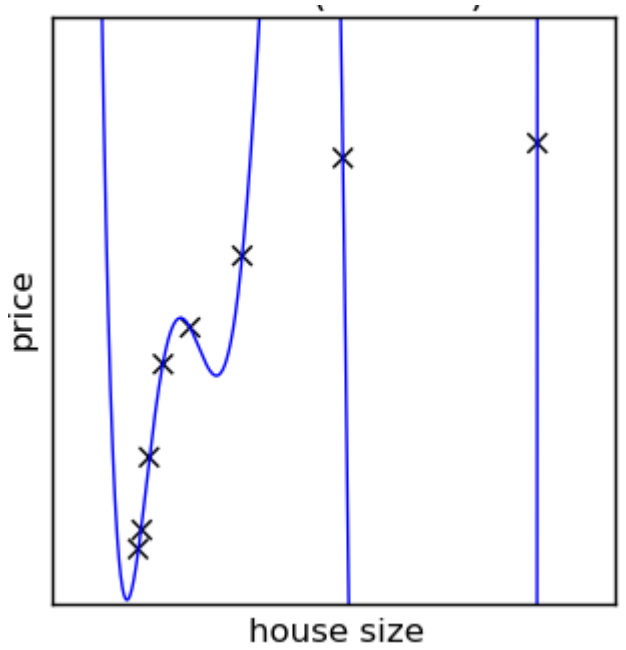
Probably not. More features (words) are needed)

  - Can you predict well housing prices using only the year it was built?

Probably not - More features are needed: size, location, #windows, etc.

Overfitting (high Variance)

- If the model overfits
  - It cannot generalize well to new data
  - It memorizes the training data
  - It has a low training error, and high test error

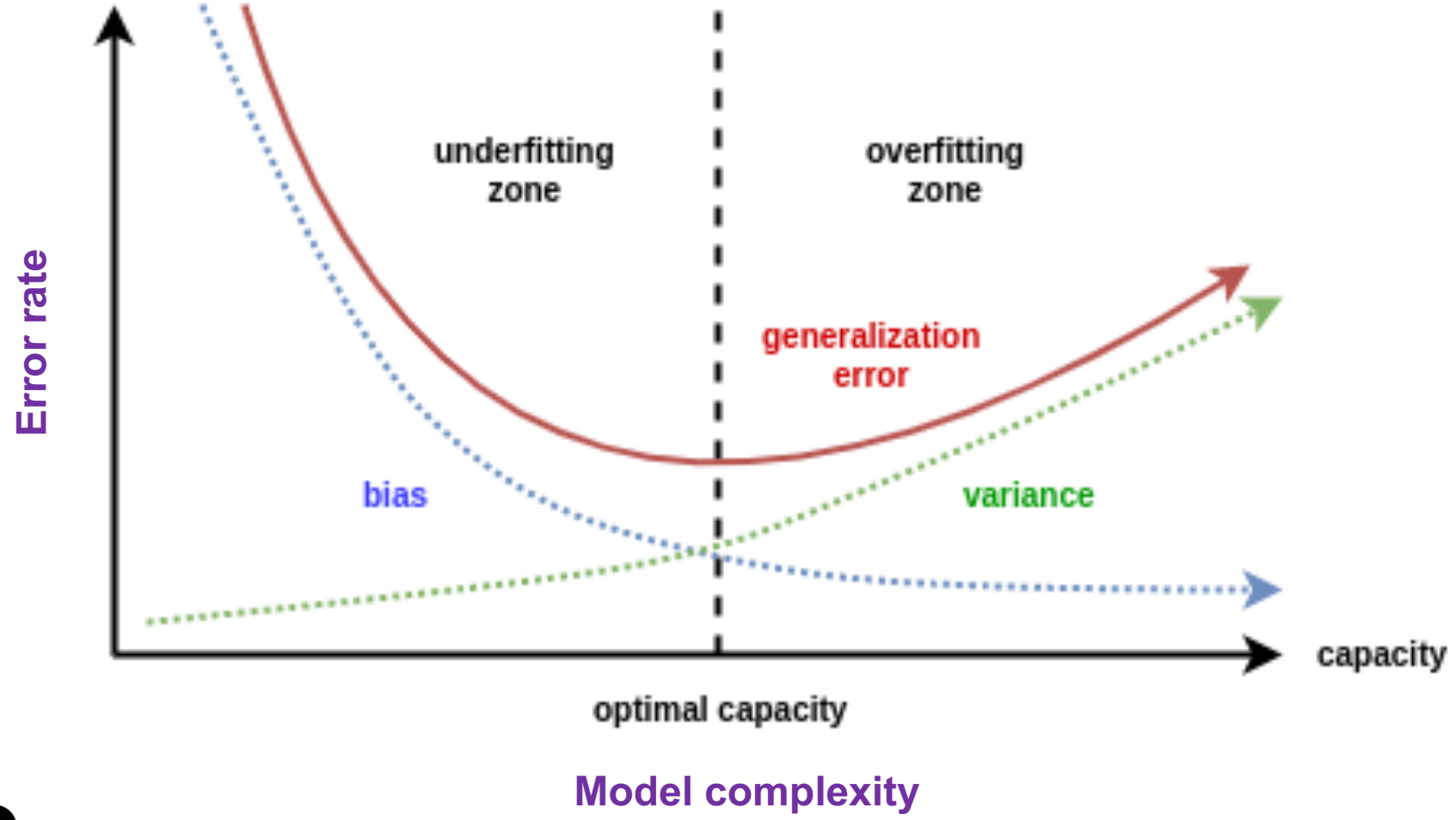


The model has no error, but it does not seem to represent the data well

Overfitting is caused by:

- Too much model complexity
  - Typically too many features
- Too little data or not enough data diversity
  - Big data: The more data we have, the more complex a model we can use
  - Diversity: Redundant data does not add information and thus does not improve the model

Bias –Variance tradeoff



TO READ:

https://djsaunde.wordpress.com/2017/07/17/the-bias-variance-tradeoff/

Addressing Overfitting

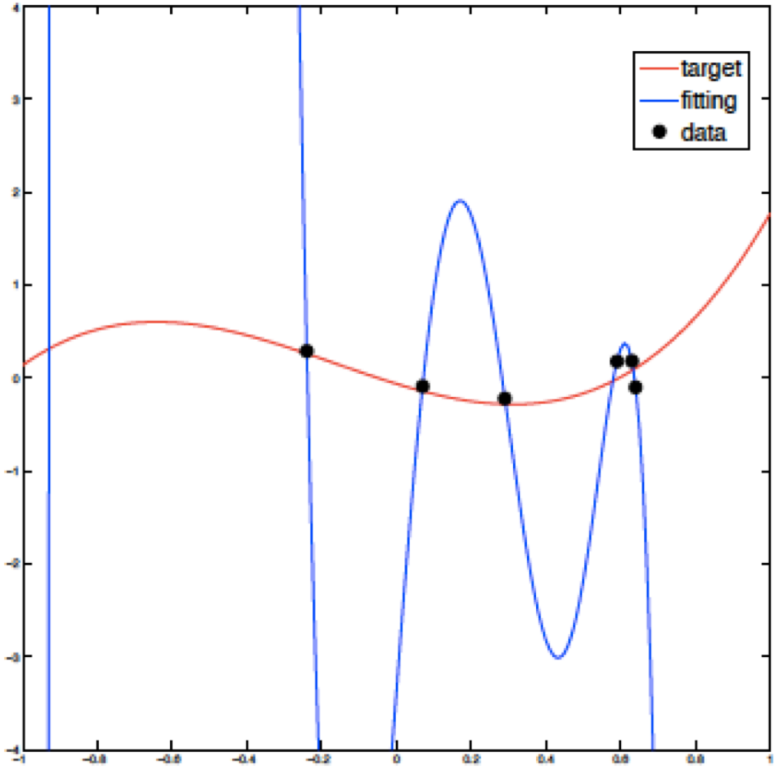
- More Data → a problem that comes up many time
- Reduce # of features or dimension
  - Select which features to keep
  - Reduce dimension
  - Select which model to use
- Regularization → Putting the brakes

Regularization

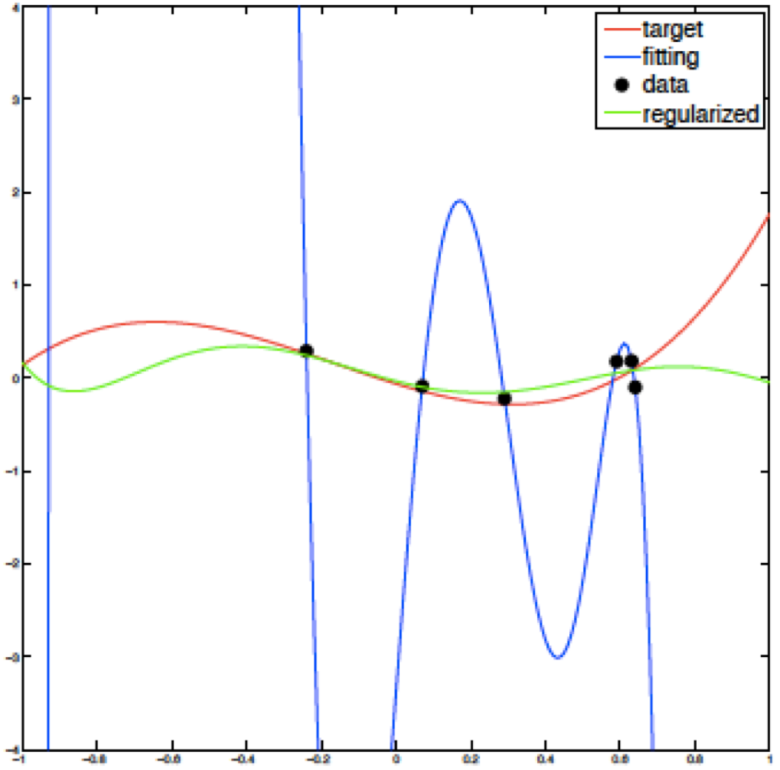
- Regularization plays a key role in many machine learning algorithms
  - Exactly fitting a model to the training data is generally undesirable, because it will fit the noise in the training examples (overfitting), and is doomed to predict (generalize) poorly on unseen data
  - In contrast, a simple model that fits the training data well is more likely to capture the regularities in it and generalize well
- A regularizer is introduced to quantify the complexity of a model
  - There is a variety of regularizers, which yield different statistical and computational properties
  - In general, there is no universally best regularizer, and a regularization approach must be chosen depending on the dataset

Regularization

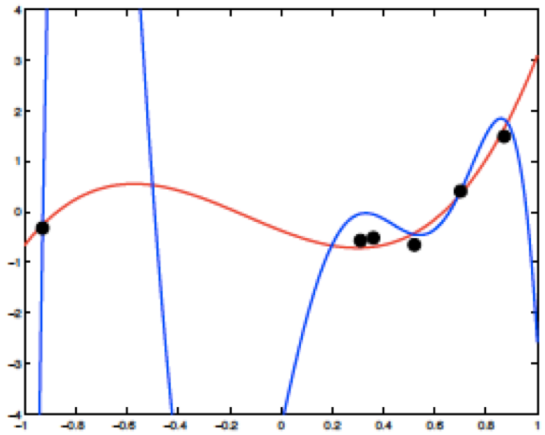
free fit



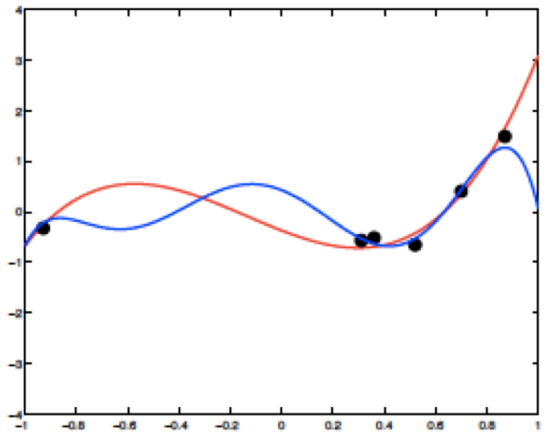
restrained fit



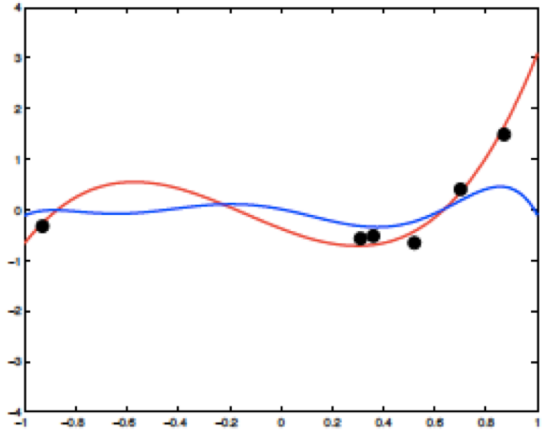
Regularization



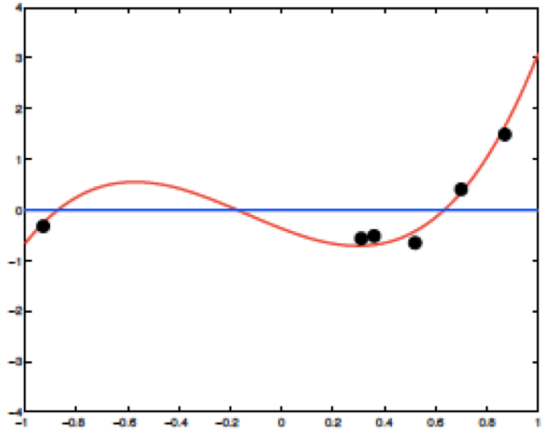
$(\lambda = 0)$



$(\lambda = 0.5)$



$(\lambda = 10)$



$(\lambda = 1000)$

Regularization formally

- In general:

Data: $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n) \in \mathbb{R}^N \times \mathbb{R}$
Estimate: $f : \mathbb{R}^N \to \mathbb{R}$

Hypothesis space $\mathcal{H}$.

$$f^* = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_i \mathcal{L}(f(x_i), y_i) + \Omega(f)$$

fit to data + complexity penalty

  - Occam's razor: "simpler" hypotheses are better
  - Regularization encodes a notion of complexity to be minimized

What else?

Look at the video: **Diagnosing bias/variance**

https://youtu.be/fZ1T7xWkImE

Look at the video: **Learning curves**

https://youtu.be/GwhzRS5uUBM

TO READ:

Evaluating Machine Learning Models

A beginner's guide to key concepts and pitfalls

Alice Zheng

Chapter 2 and 3

The remaining of the chapters are optional

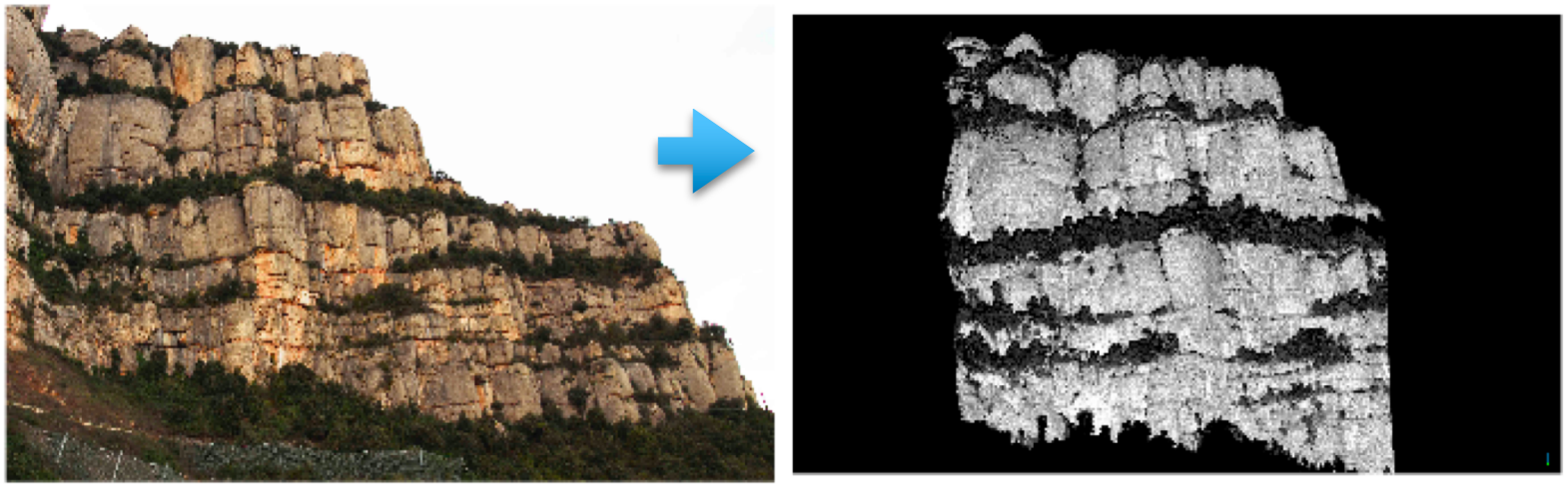
Case study: Rockfalls detection

The problem

Rock falls

Landslide

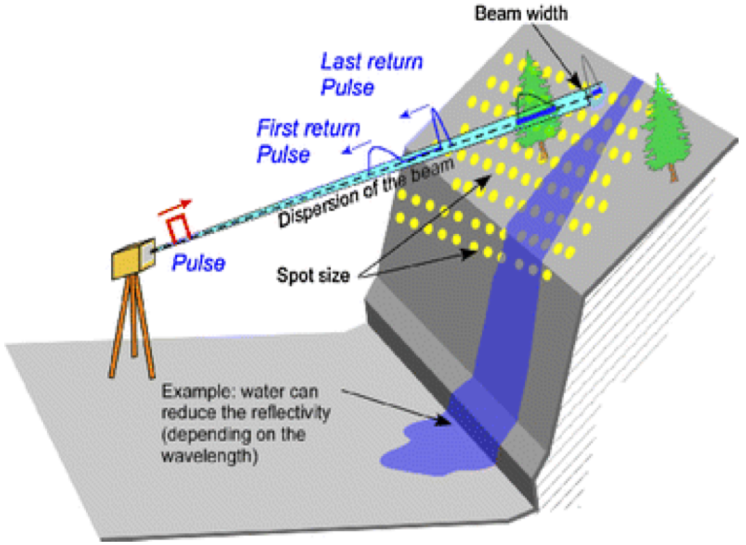
Input data



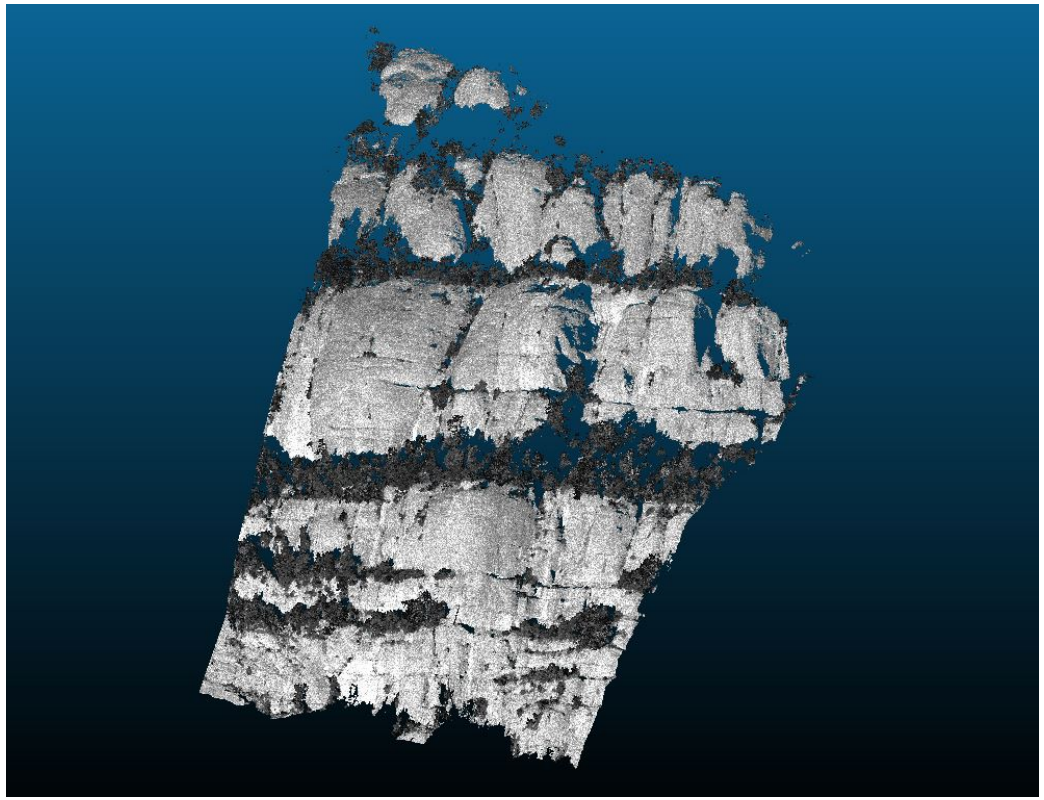
(a) Wall

(b) Point cloud of wall

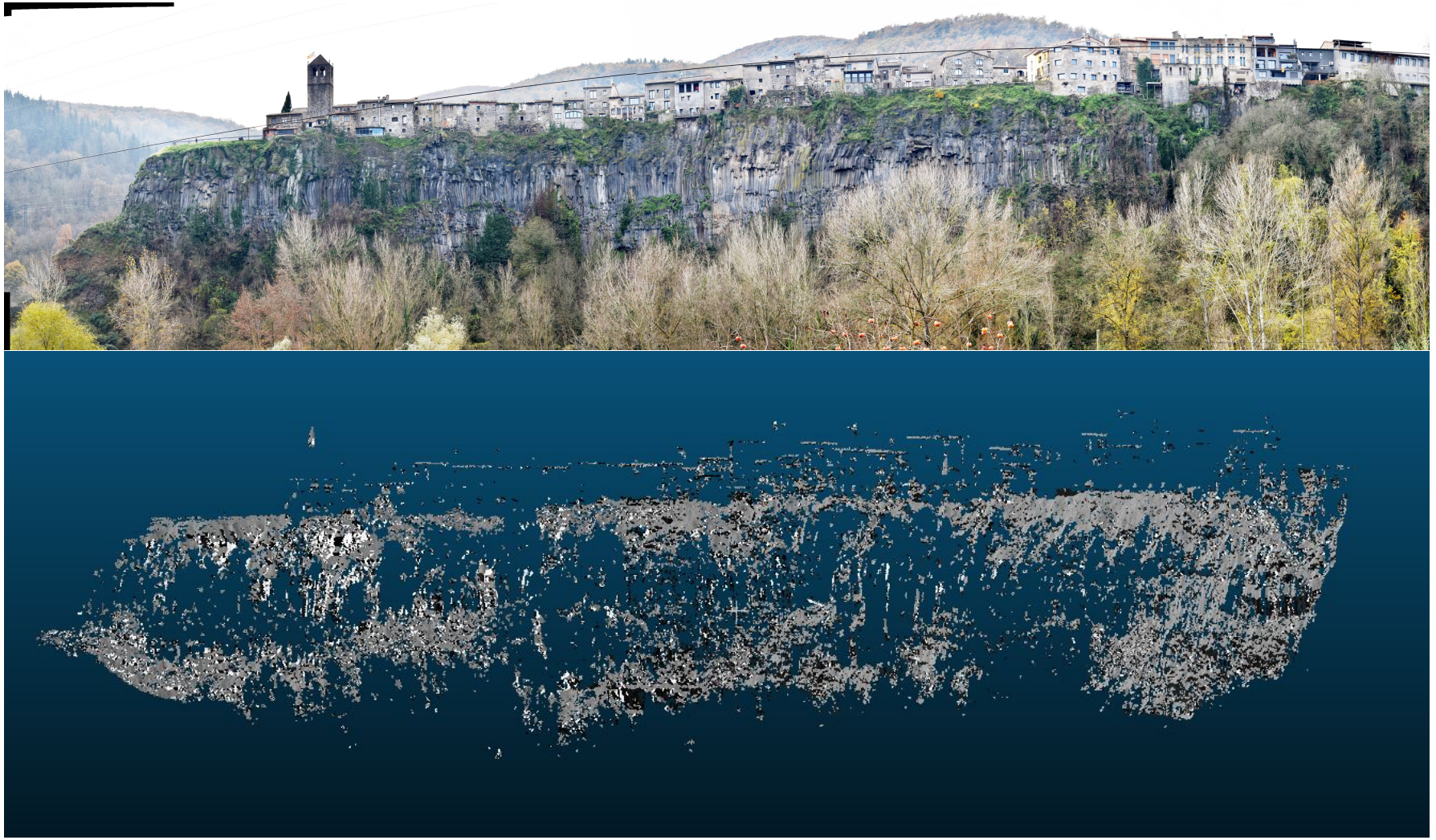
LIDAR technology, Laser Scanning 3D



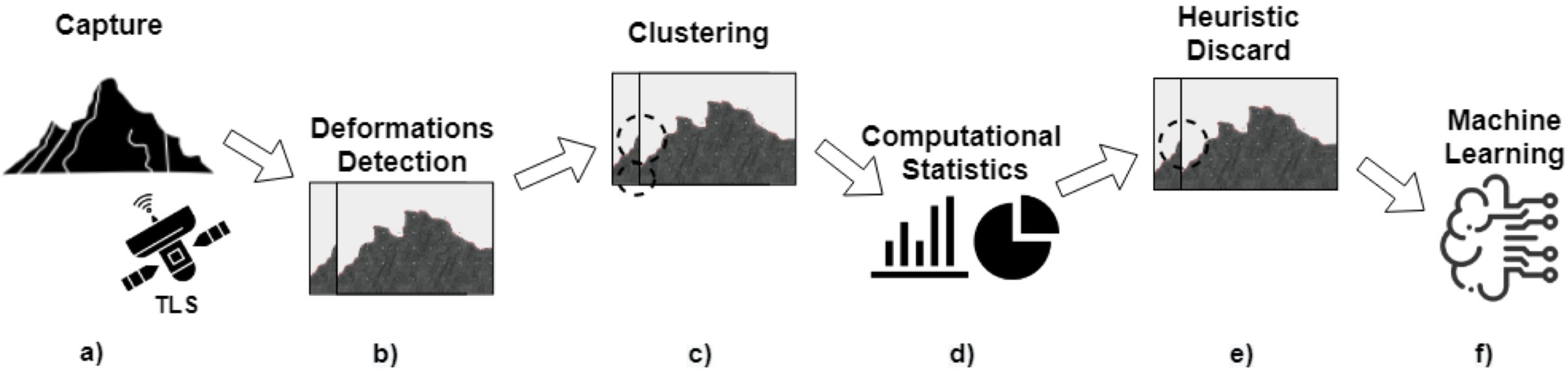
Rockfall detection Degotalls



Rockfall detection castellfollit

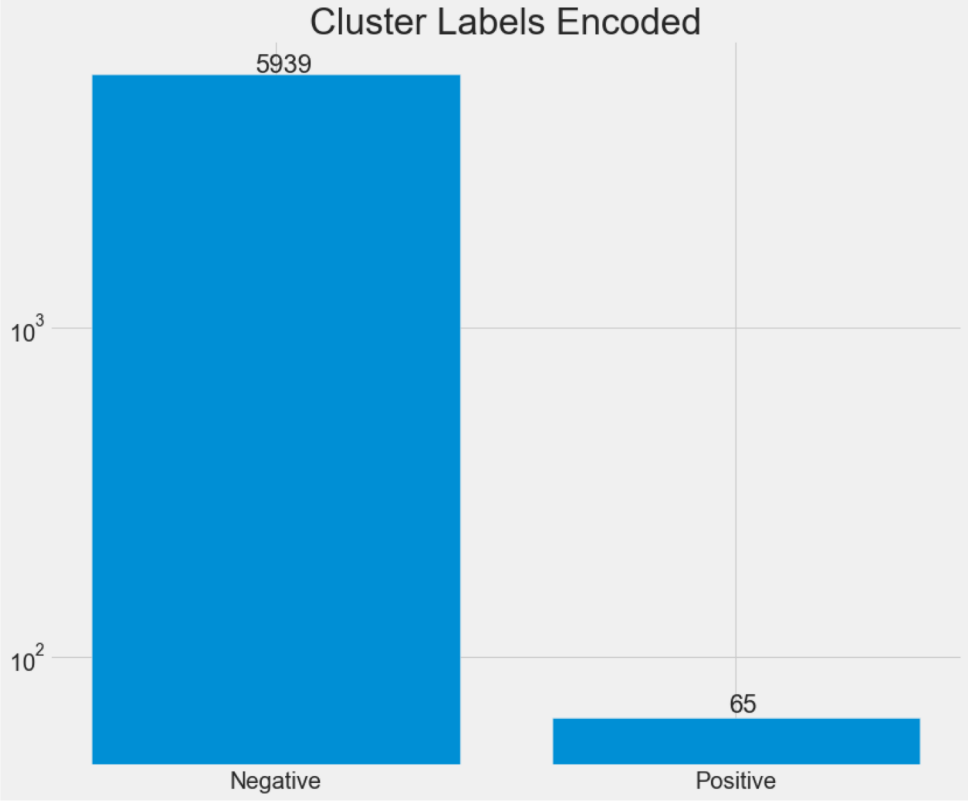


Description of the process

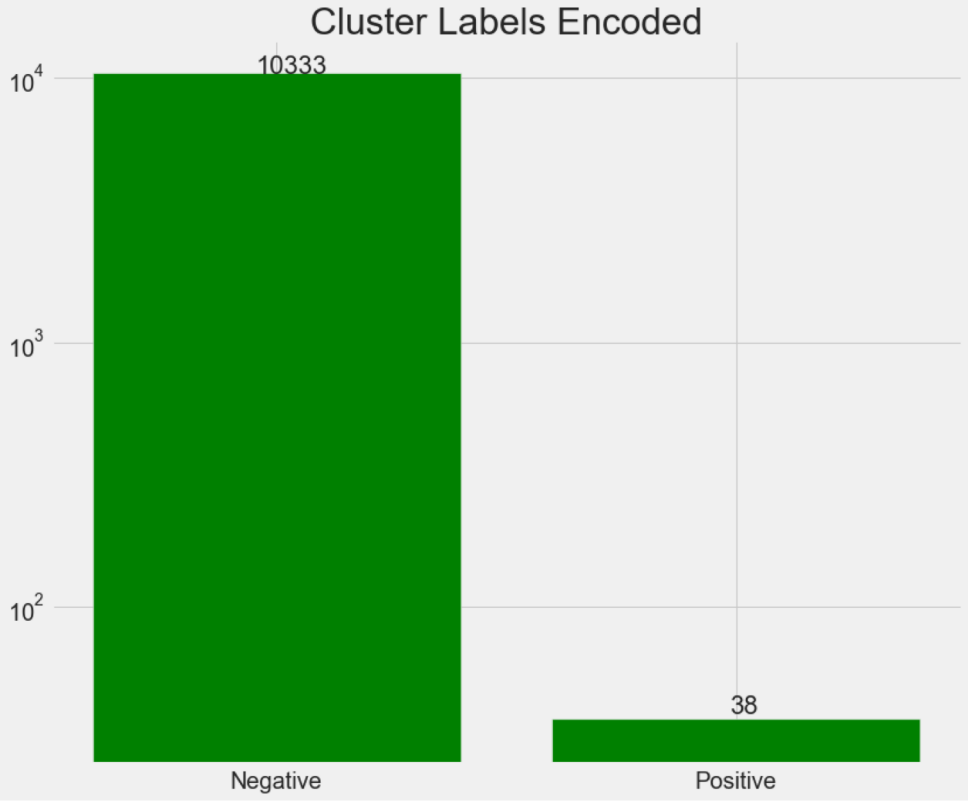


Imbalance of the data

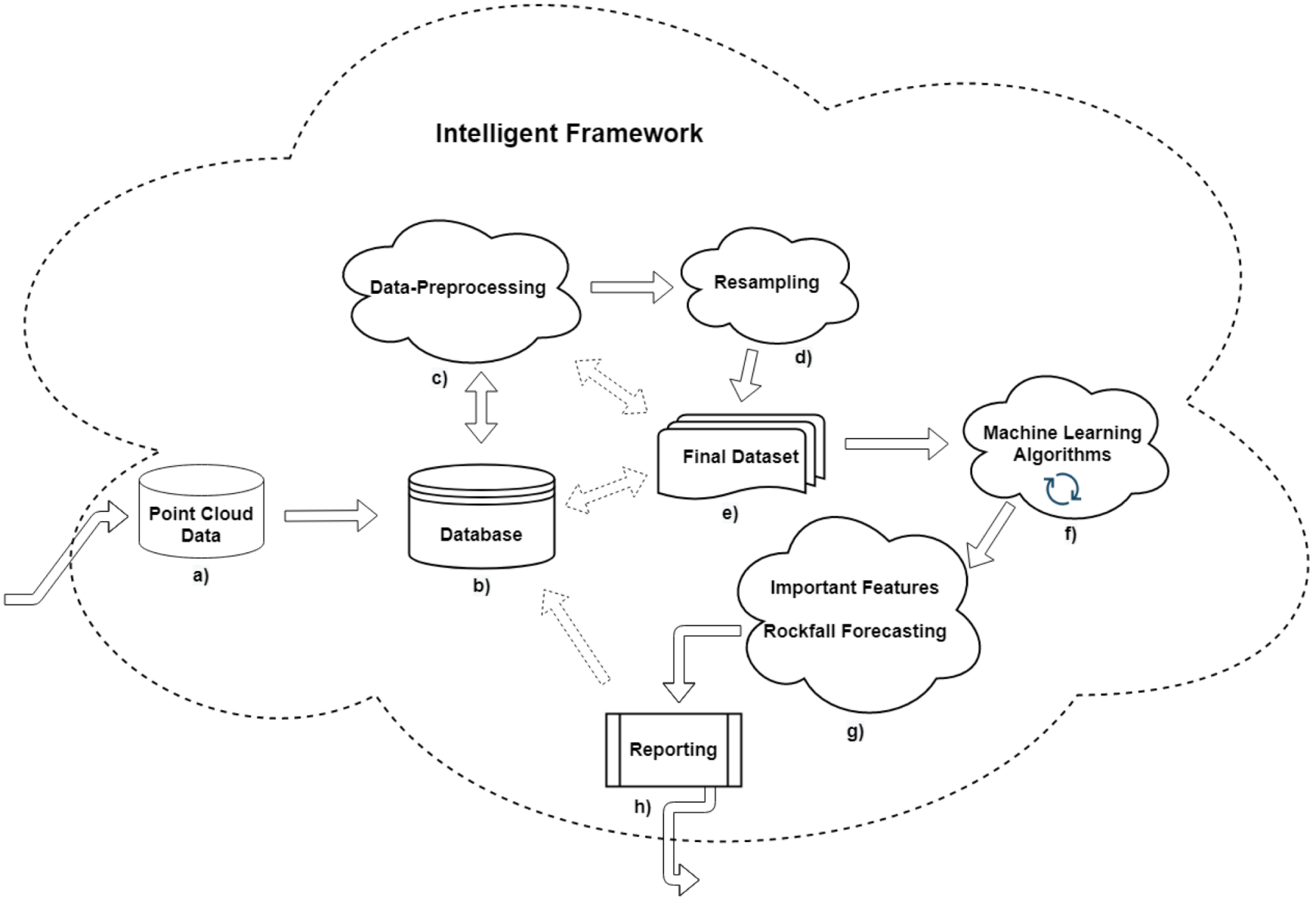
(a) Degotalls case study



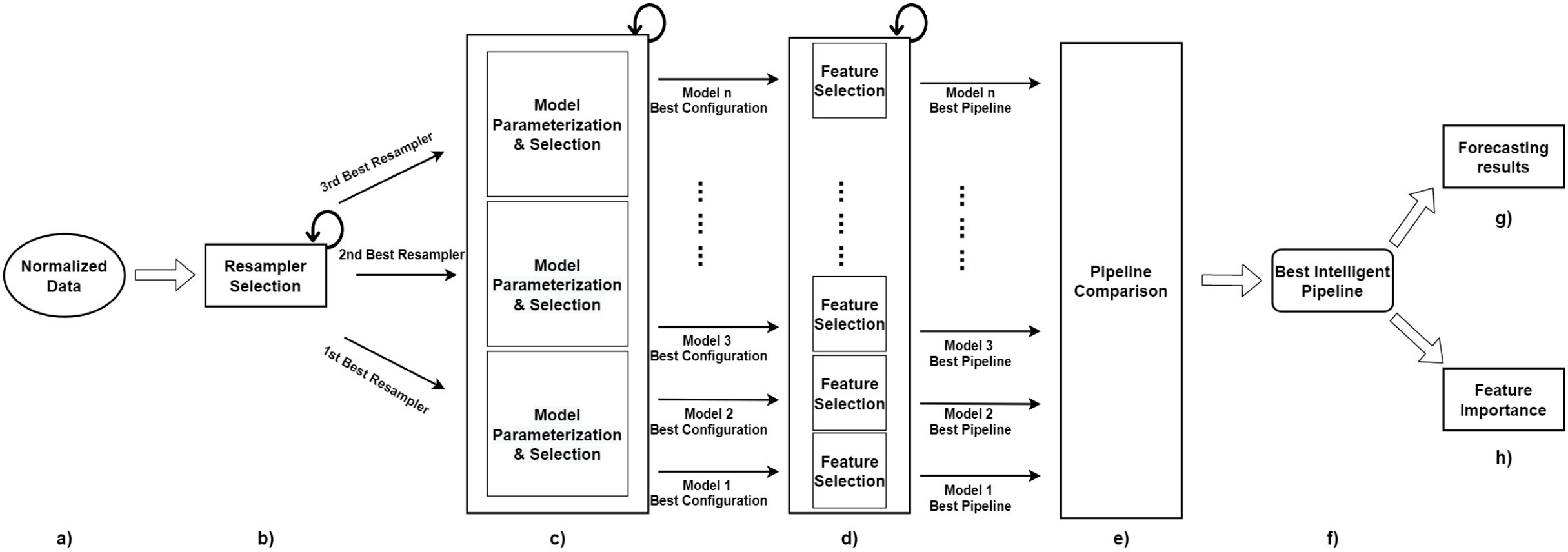
(b) Castellfollit case study



Framework


Intelligent Framework
Data-Preprocessing
c)
Resampling
d)
Point Cloud Data
a)
Database
b)
Final Dataset
e)
Machine Learning Algorithms
f)
Important Features
Rockfall Forecasting
g)
Reporting
h)

Machine learning pipeline



(a) Degotalls case study.

Resampling Method	$\overline{Acc_b}$	**Error (%)**
Cluster Centroids	0.89	3.83
ProWSyn	0.87	5.34
SMOTE-IPF	0.87	5.56
SWIM	0.86	5.65
SMOBD	0.86	6.29
Lee	0.86	5.48
ADASYN	0.86	6.28
Assembled-SMOTE	0.86	6.16
SMOTE	0.86	5.85
SMOTE-TomekLinks	0.86	6.56
CCR	0.85	7.91
G-SMOTE	0.85	7.15
LVQ-SMOTE	0.84	6.20
Polynom-fit-SMOTE	0.83	7.56
SPIDER	0.80	6.29
Cluster Representatives	0.78	7.67

(b) Castellfollit case study.

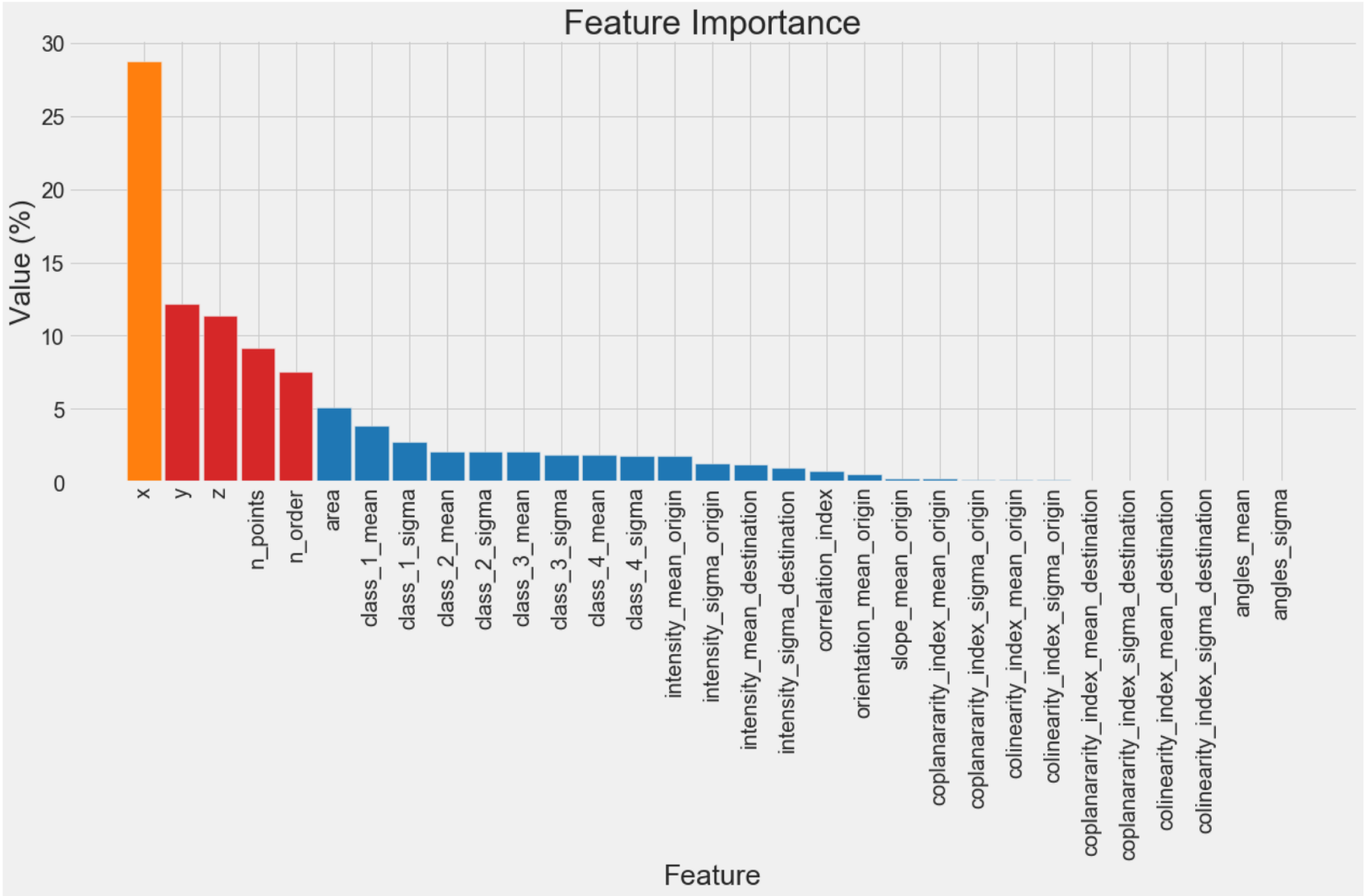
Resampling Method	$\overline{Acc_b}$	**Error (%)**
Cluster Centroids	0.82	7.78
ProWSyn	0.71	16.94
CCR	0.70	15.88
SWIM	0.70	14.90
Lee	0.68	17.52
SMOTE	0.68	18.13
LVQ-SMOTE	0.68	18.33
ADASYN	0.68	18.50
SMOBD	0.68	18.86
SMOTE-IPF	0.68	17.76
SMOTE-TomekLinks	0.68	17.33
Assembled-SMOTE	0.67	18.08
G_SMOTE	0.66	19.83
Polynom-fit-SMOTE	0.65	18.00
SPIDER	0.58	17.29
Cluster Representatives	0.56	18.28

(a) Degotalls case study.

Method	Model Parameterization		Feature Selection		
	Acc_b	Error (%)	Acc_b	Error (%)	Features
XGB-SMOTE_IPF	0.94	3.83	0.95	3.78	35
MLP-SMOTE_IPF	0.94	5.54	0.95	5.64	35
KNN-ClusterCentroids	0.94	4.29	0.95	4.09	36
XGB-ProWSyn	0.94	3.84	0.95	3.68	35
SVC-ProWSyn	0.94	5.68	0.95	4.14	17
LDA-SMOTE_IPF	0.94	4.13	0.95	4.07	31

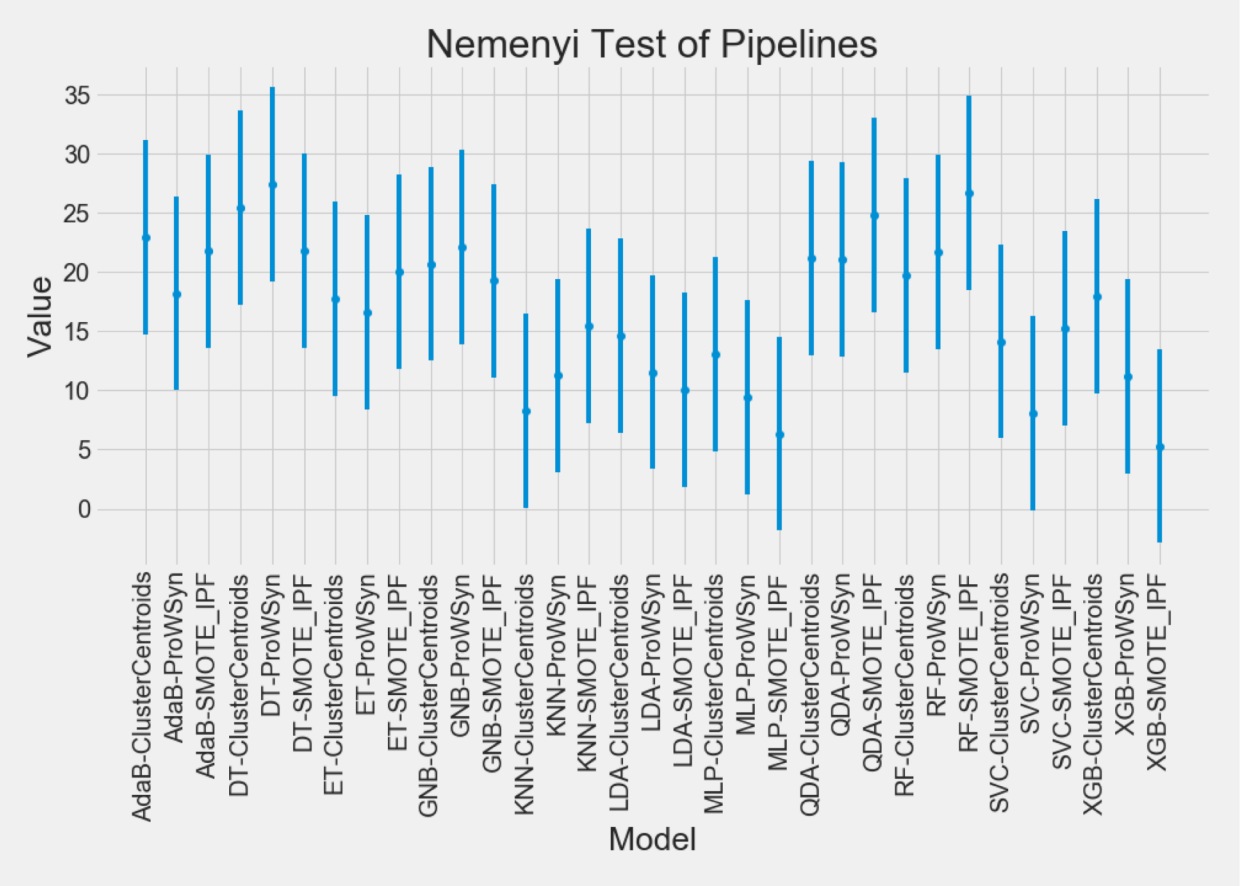
(b) Castellfollit case study.

Method	Model Parameterization		Feature Selection		
	Acc_b	Error (%)	Acc_b	Error (%)	Features
XGB-ProWSyn	0.93	8.11	0.94	6.92	30
LDA-ProWSyn	0.93	0.36	0.93	3.98	30
MLP-CCR	0.93	8.21	0.93	8.18	30
LDA-CCR	0.93	0.54	0.93	4.06	29
XGB-CCR	0.92	8.11	0.92	8.17	30
SVC-ClusterCentroids	0.91	8.18	0.91	8.26	30

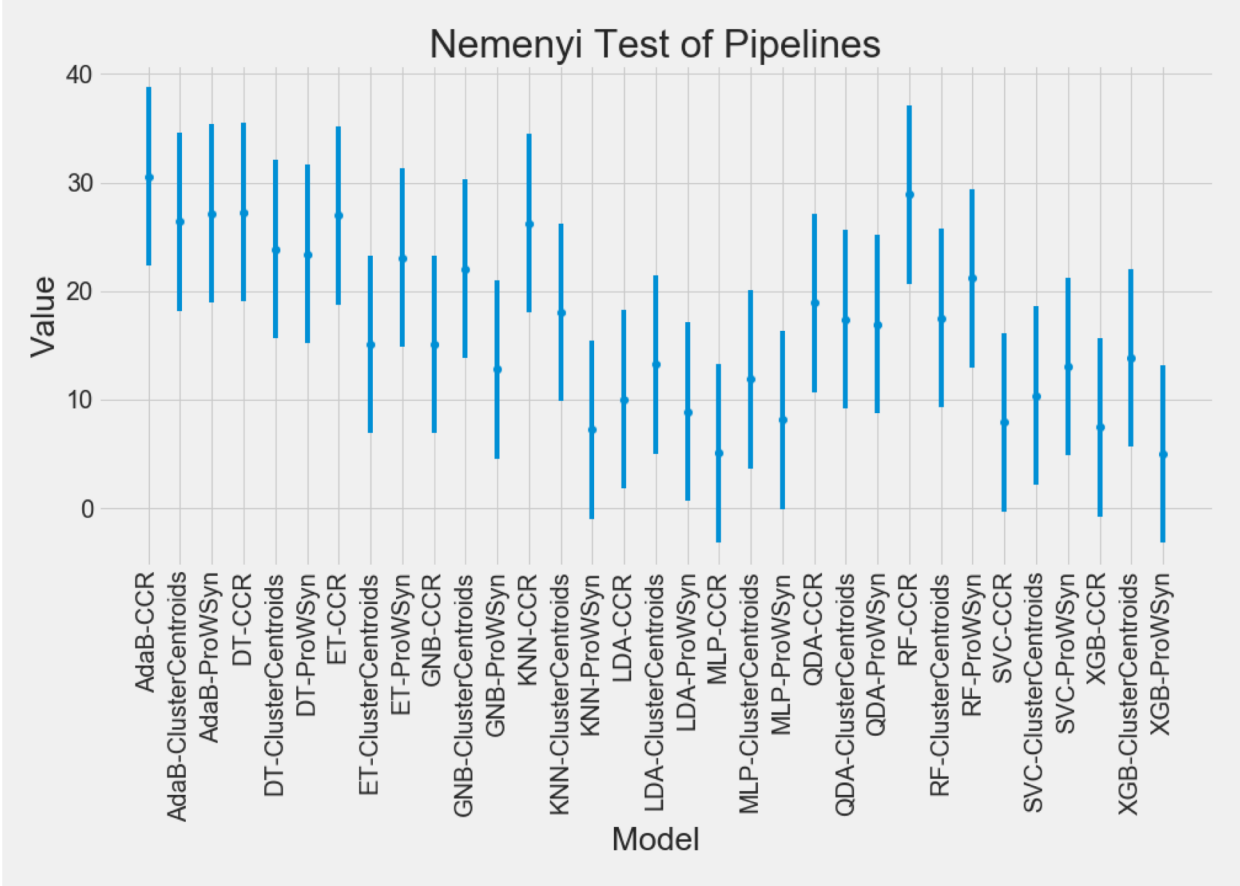

Feature Importance
Value (%)
30
25
20
15
10
5
0
x
y
z
n_points
n_order
area
class_1_mean
class_1_sigma
class_2_mean
class_2_sigma
class_3_mean
class_3_sigma
class_4_mean
class_4_sigma
intensity_mean_origin
intensity_sigma_origin
intensity_mean_destination
intensity_sigma_destination
correlation_index
orientation_mean_origin
slope_mean_origin
coplanararity_index_mean_origin
coplanararity_index_sigma_origin
colinearity_index_mean_origin
colinearity_index_sigma_origin
coplanararity_index_mean_destination
coplanararity_index_sigma_destination
colinearity_index_mean_destination
colinearity_index_sigma_destination
angles_mean
angles_sigma
Feature

Statistical results

(a) Degotalls



(b) Castellfollit



Overall Results

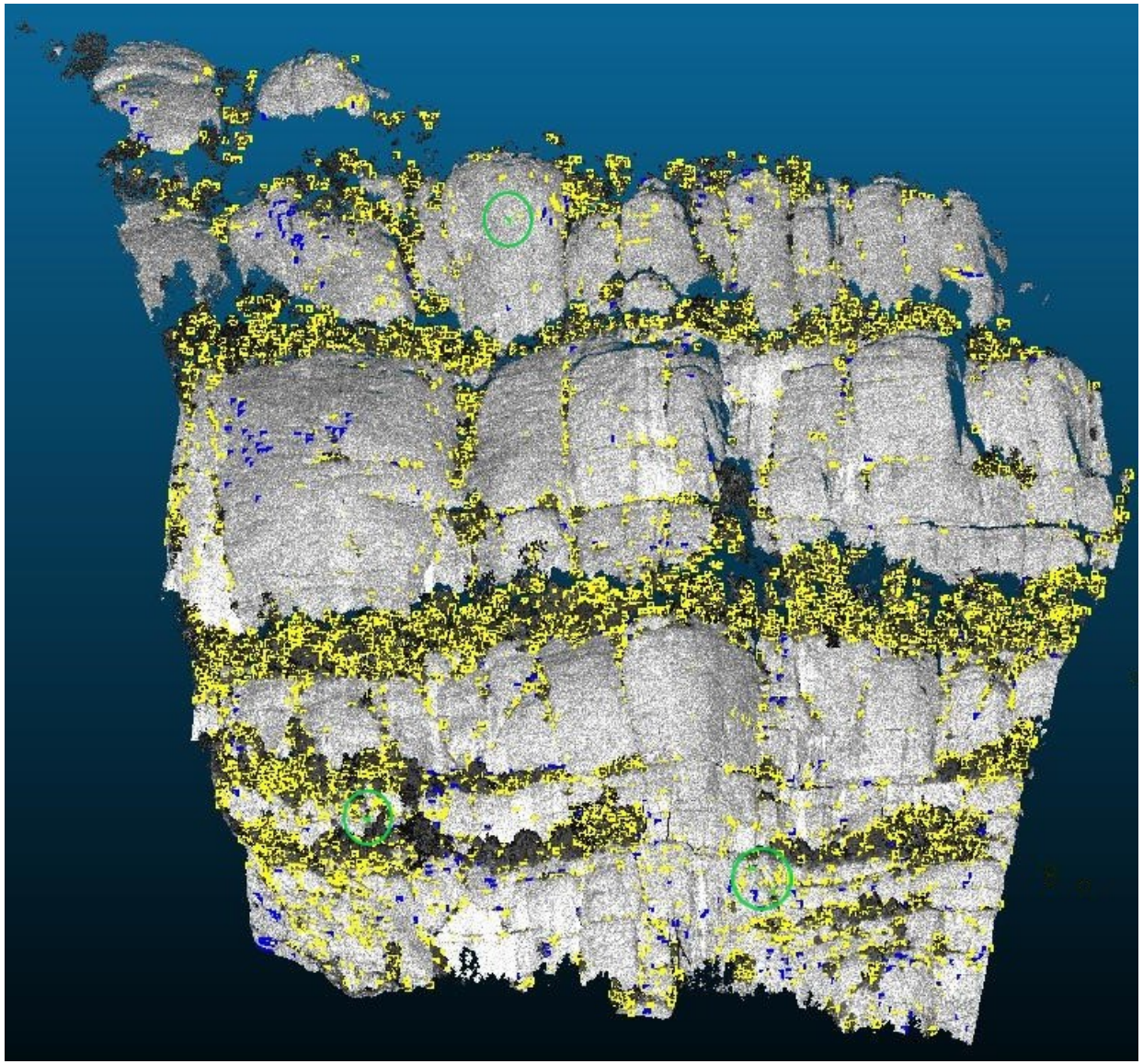
(a) Degotalls case study.

Method	$\overline{Acc_b}$	**Error (%)**	Acc_b^{best}
Baseline	0.78	7.65	0.84
+Resampling	0.85	3.25	0.89
+Model Parameterization	0.89	4.65	0.94
+Feature Selection	0.91	4.45	0.95

(b) Castellfollit case study.

Method	$\overline{Acc_b}$	**Error (%)**	Acc_b^{best}
Baseline	0.56	18.04	0.80
+Resampling	0.68	8.21	0.82
+Model Parameterization	0.79	15.93	0.93
+Feature Selection	0.82	11.75	0.94

Rock fall detection final results



Links

TO READ (optional):

End-to-End Intelligent Framework for Rockfall Detection

Thanasis Zoumpekas, Anna Puig, Maria Salamó, David García-Sellés, Laura Blanco Nuñez, Marta Guinau

https://doi.org/10.1002/int.22557 (final version)

https://arxiv.org/abs/2102.06491 (preliminar version)

https://github.com/thzou/rockfall_detection

Week 4

Course. Introduction to Machine Learning

Theory 4. Model Evaluation

Dr. Maria Salamó Llorente
maria.salamo@ub.edu
Dept. Mathematics and Informatics,
Faculty of Mathematics and Informatics,
University of Barcelona (UB)