ML Fundamentals

Instituto Balseiro

Instituto Balseiro
23/08/2022

Lecture 3
Error

ML Fundamentals – Lecture 3

- Probability recap
- EDA
- Error
  - Variance vs. bias
  - Loss and optimizers
  - Train/validation/test
  - Overfitting and underfitting
  - Regularization
  - Performance metrics
  - Resampling

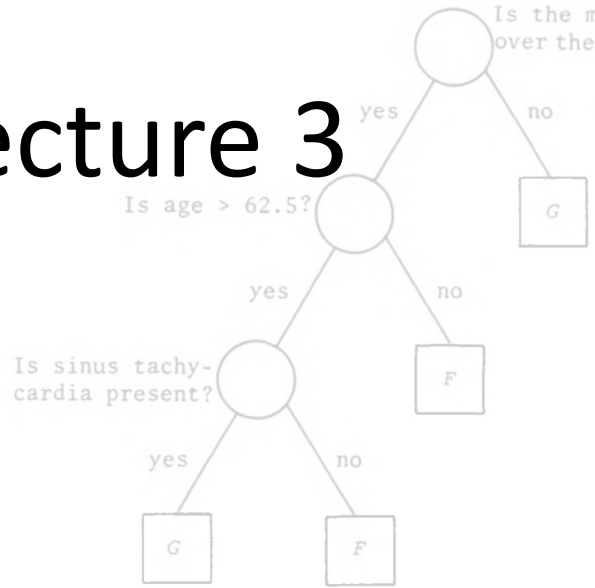
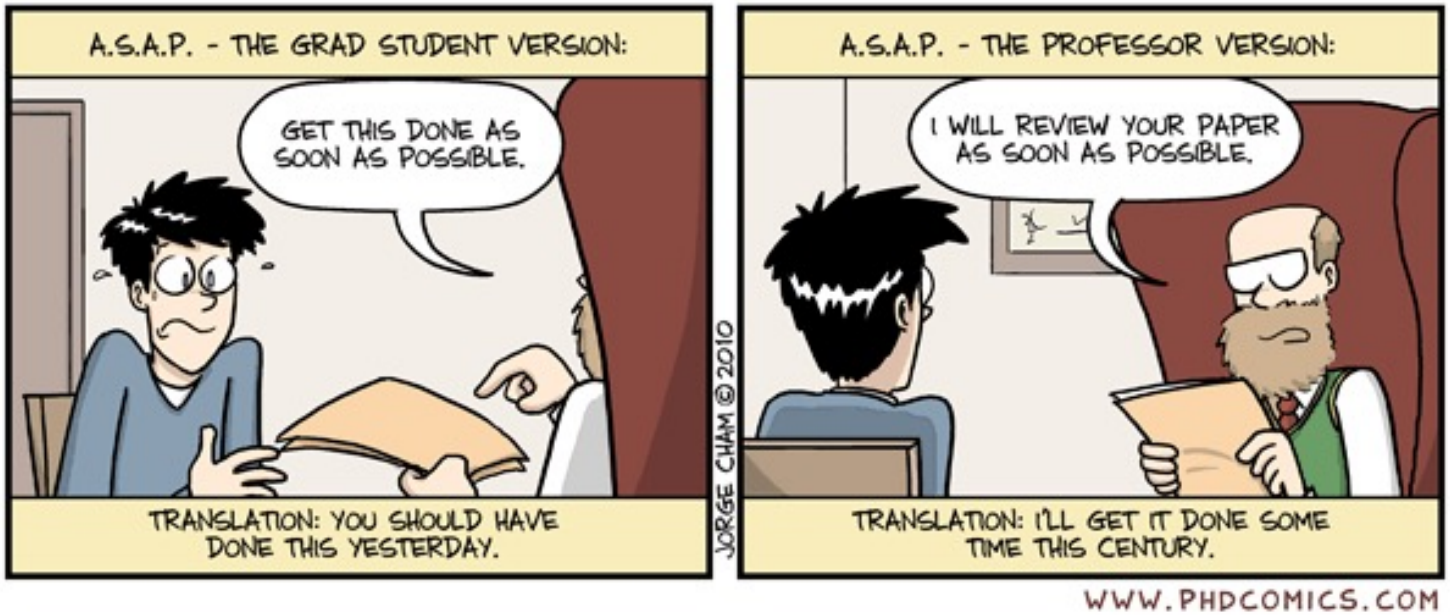


FIGURE 1.1

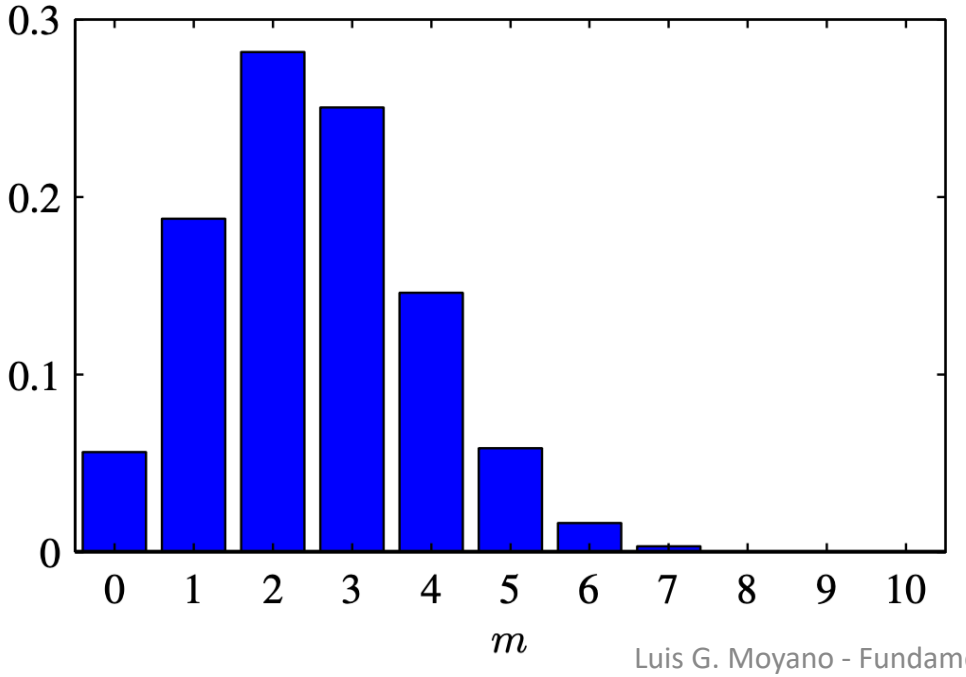
Supervised vs. Unsupervised

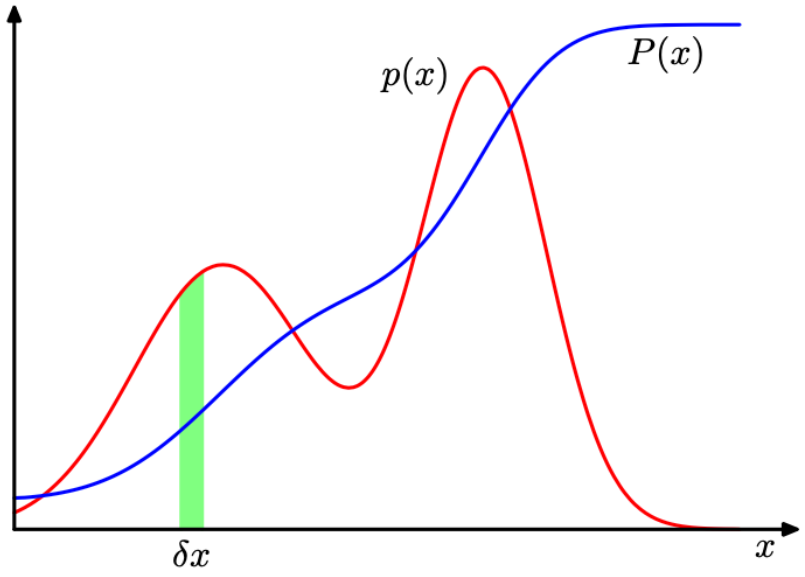
- Labels = ground truth or supervisor
- predictive vs. descriptive approach
- supervisors are resource expensive
- unlabeled data is way more common



Probability recap

- Random variables: x
- Probability distributions: p(x)
- Cumulative function: P(x)





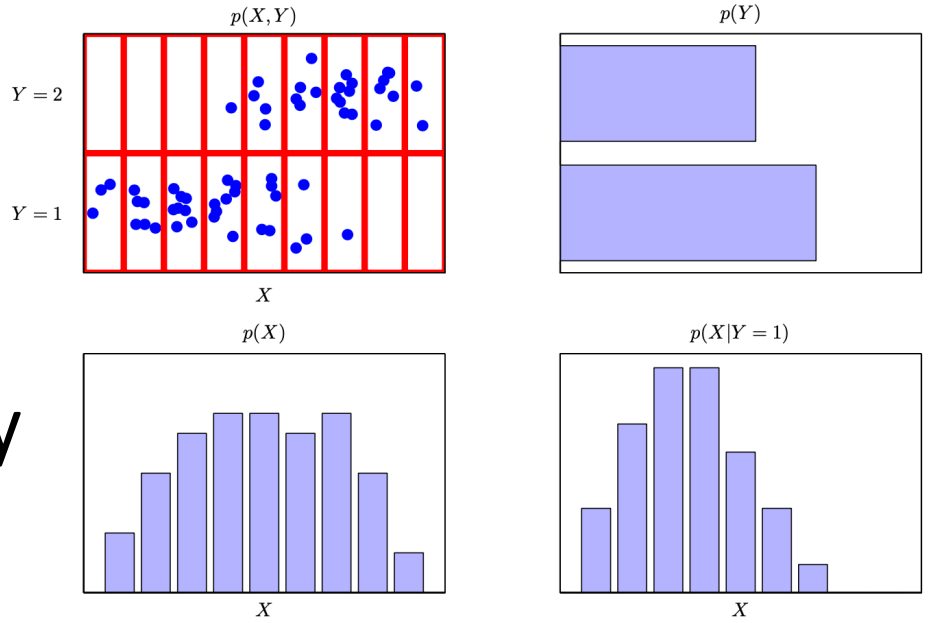
Probability recap

- Discrete
  - The domain of P must be the set of all possible states of x.
  - $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$.
  - $\sum_{x \in \mathrm{x}} P(x) = 1$.
- Continuous
  - The domain of p must be the set of all possible states of x.
  - $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$.
  - $\int p(x)dx = 1$.

Probability recap

- Marginal probability

sum rule $p(x) = \int p(x, y)dy.$

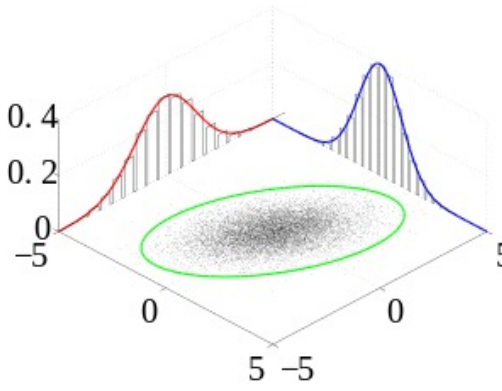


- Conditional probability

product rule $P(\mathrm{y} = y \mid \mathrm{x} = x) = \dfrac{P(\mathrm{y} = y, \mathrm{x} = x)}{P(\mathrm{x} = x)}.$

- Chain rule of conditional probabilities

$$P(\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(n)}) = P(\mathrm{x}^{(1)}) \Pi_{i=2}^{n} P(\mathrm{x}^{(i)} \mid \mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(i-1)}).$$



Probability recap

- Expectation

$$\mathbb{E}_{\mathbf{x}\sim P}[f(x)] = \sum P(x) f(x),$$

$$\mathbb{E}_{\mathbf{x}\sim p}[f(x)] = \int p(x) f(x) dx.$$

- Variance

$$\mathrm{Var}(f(x)) = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2 \right].$$

Supervised vs. Unsupervised

$$p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \qquad p(\mathbf{x}_i|\boldsymbol{\theta})$$

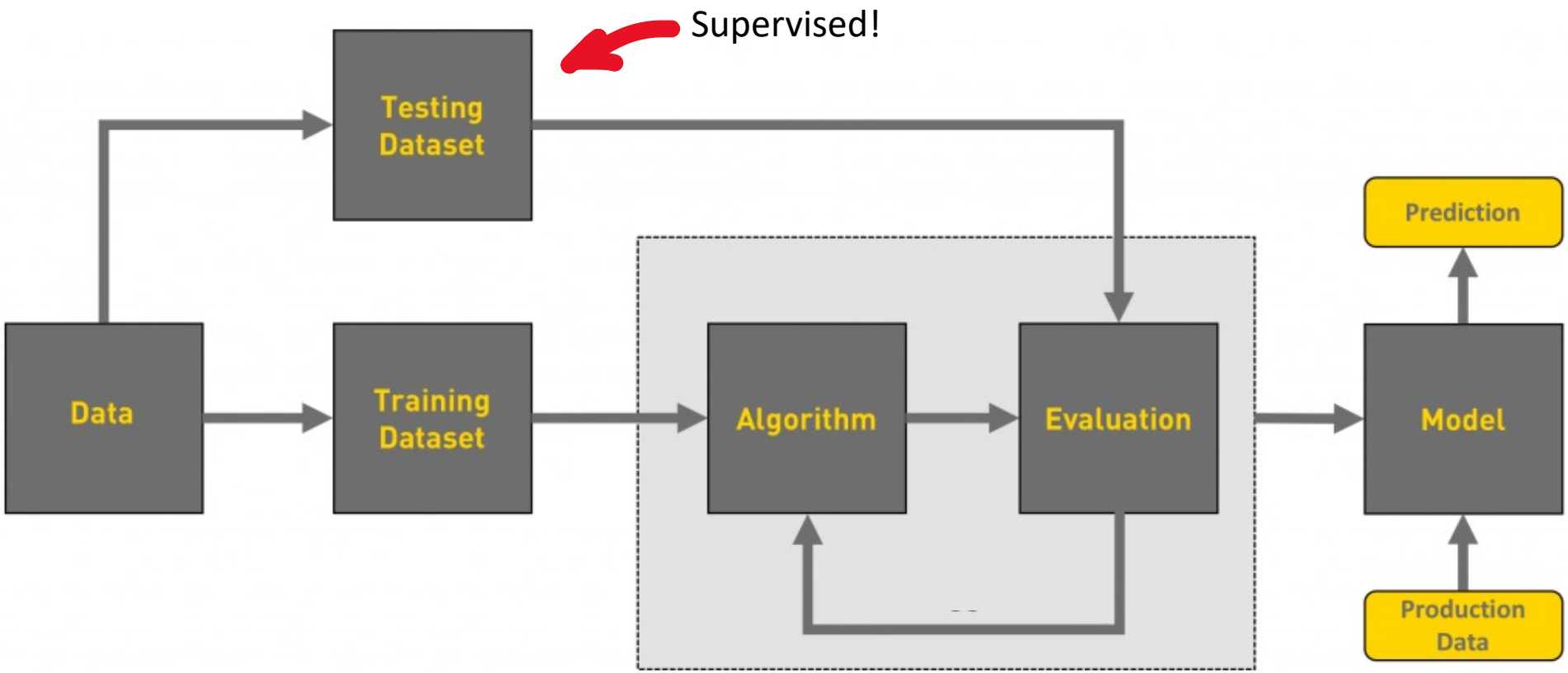
- Unsupervised from supervised:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(\mathrm{x}_i \mid \mathrm{x}_1, \ldots, \mathrm{x}_{i-1}).$$

- Supervised from unsupervised:

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')}.$$

ML canonical end-to-end workflow



EDA: Exploratory data analysis (in a nutshell)

- Load and inspect data

```
import numpy as np
import pandas as pd
import pickle
```

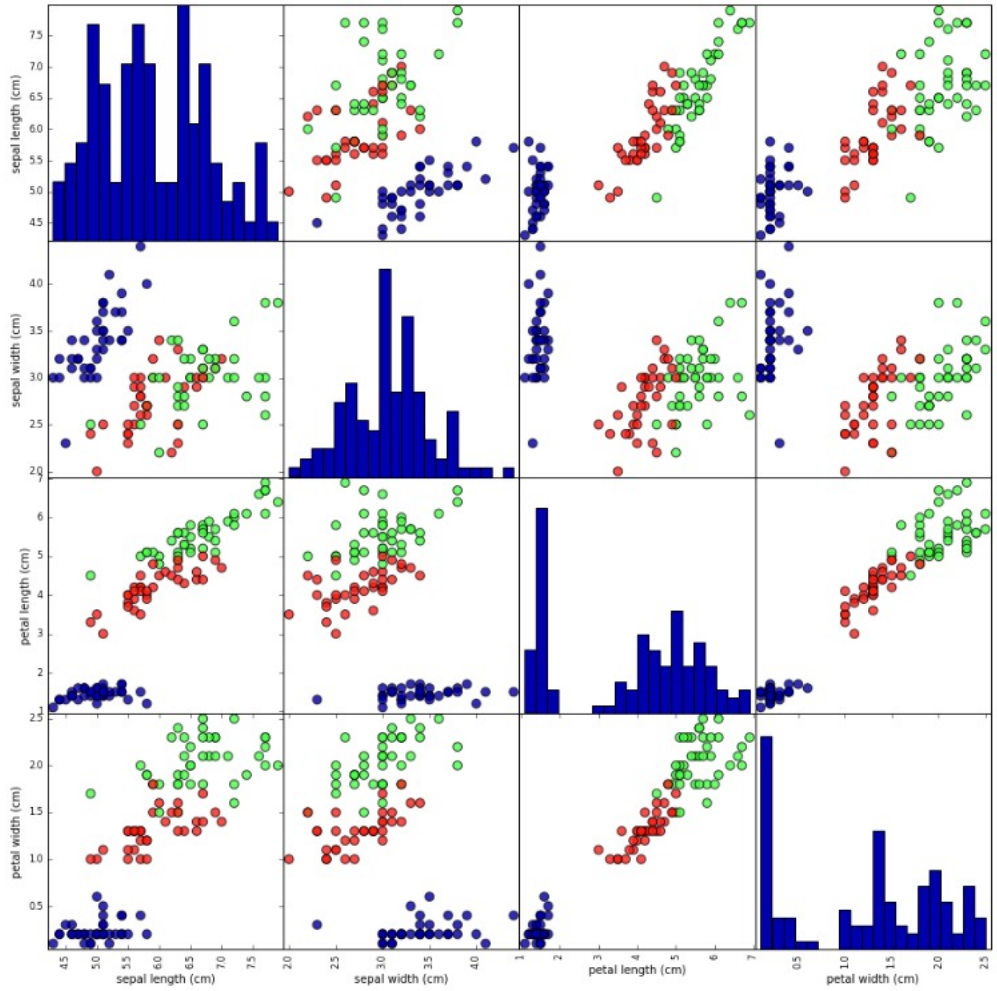
```
>>> pdDf = pd.read_csv('100 Sales Record.csv')
>>> pdDf.head()
```

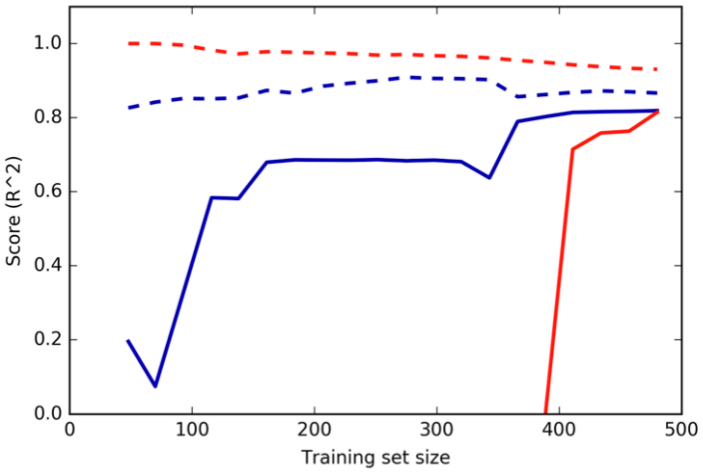
```
with open('test.pkl','wb') as f:
    pickle.dump(pdDf, f)
```

```
with open("test.pkl", "rb") as f:
    d4 = pickle.load(f)

>>> d4.head()
```

EDA: Exploratory data analysis (in a nutshell)





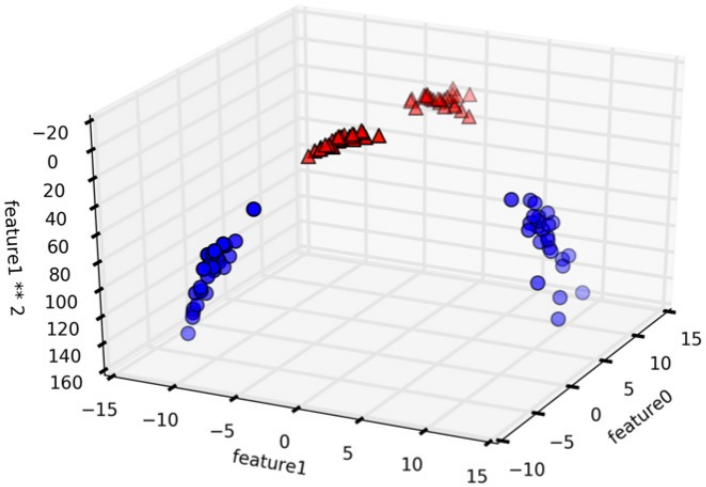
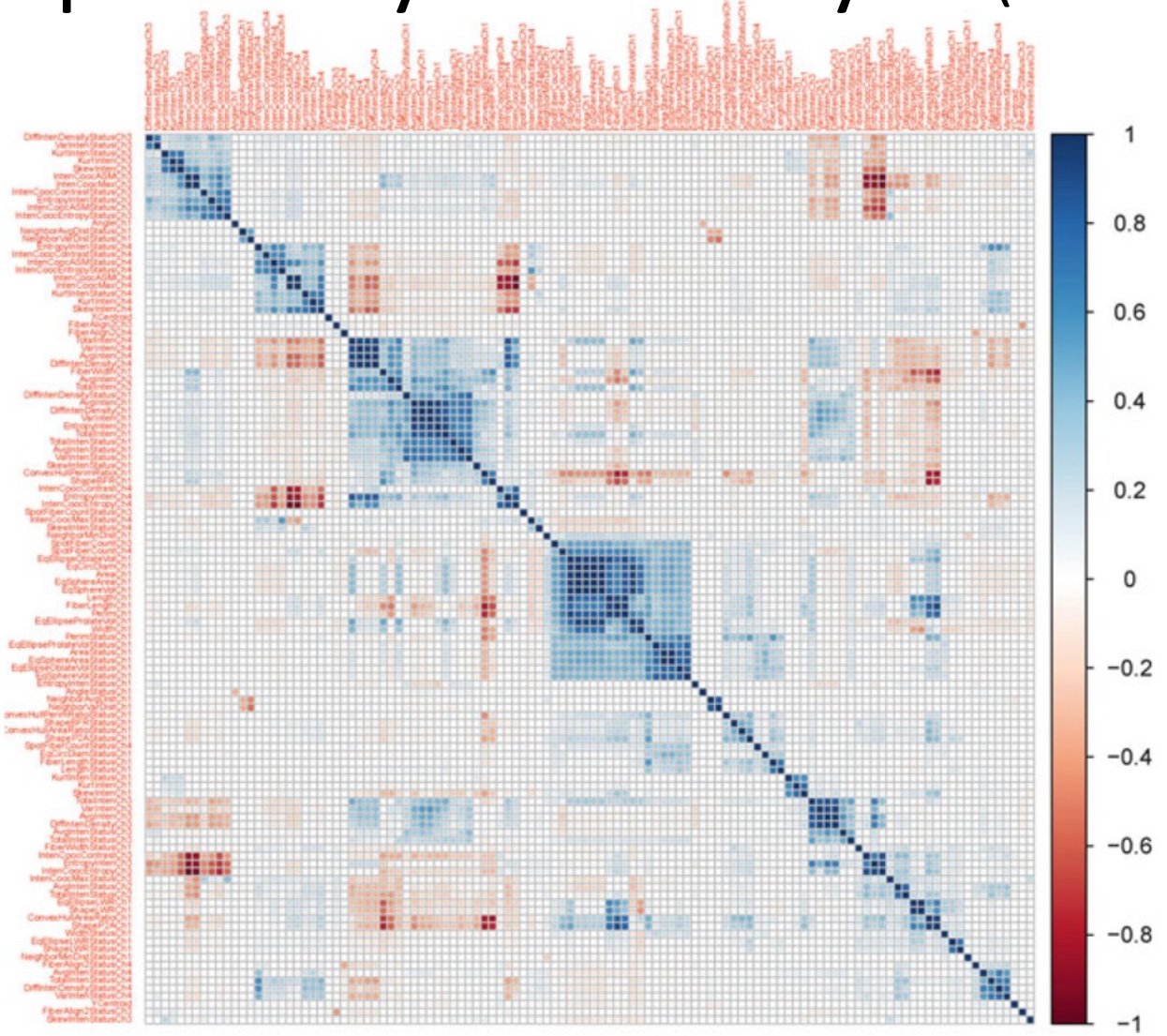
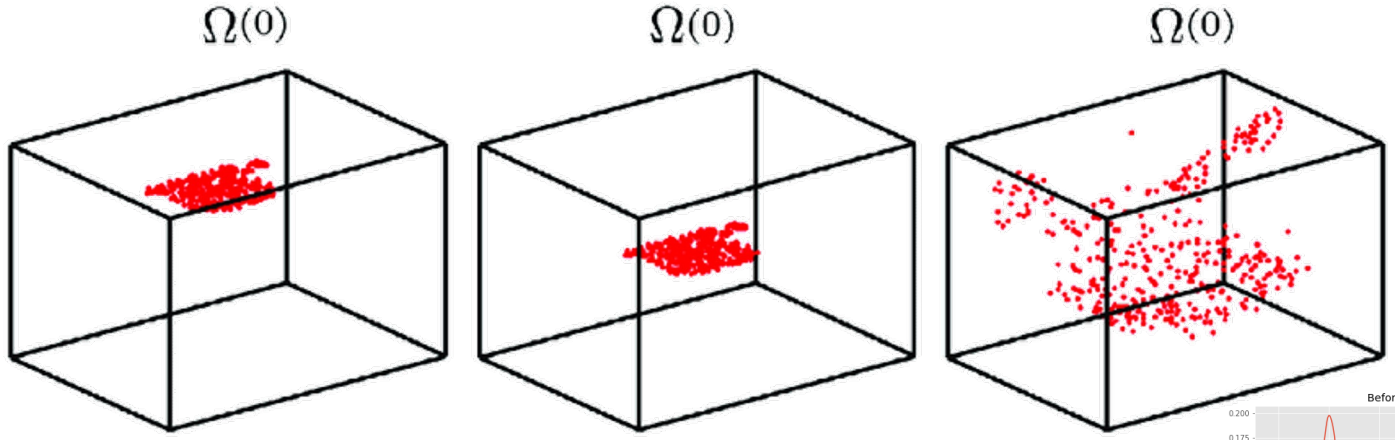


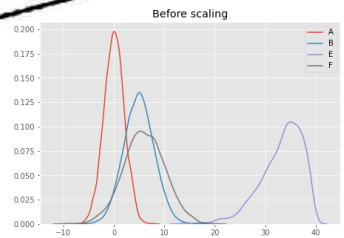
Figure 1-3. Pair plot of the Iris dataset, colored by class label

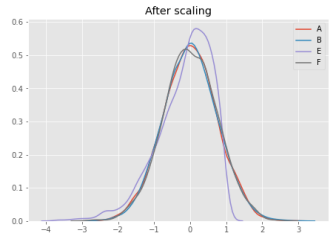
EDA: Exploratory data analysis (in a nutshell)

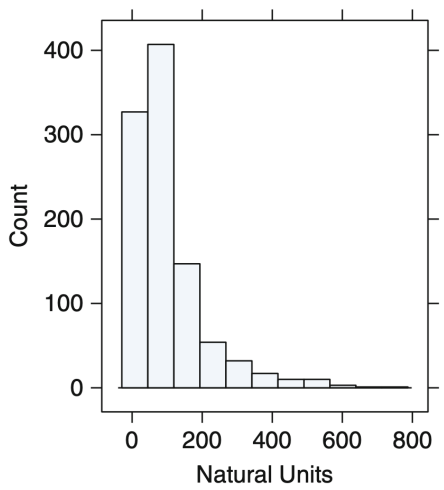


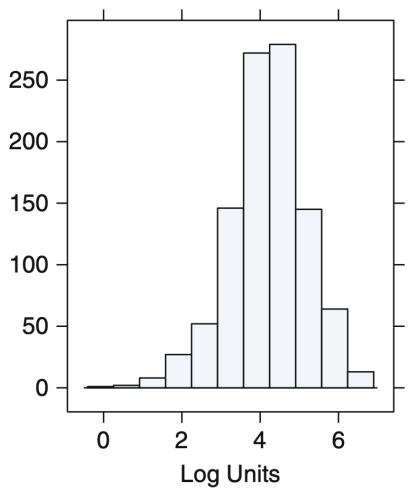
EDA: Exploratory data analysis (in a nutshell)

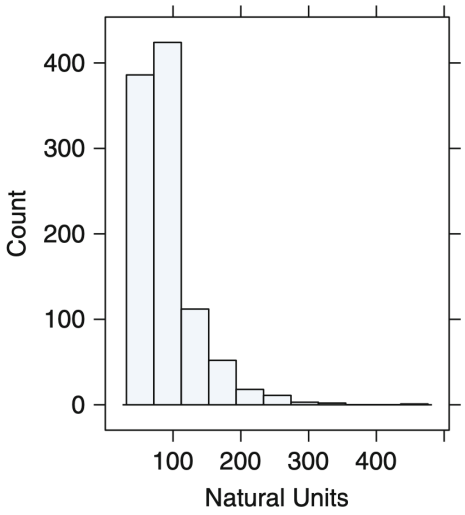


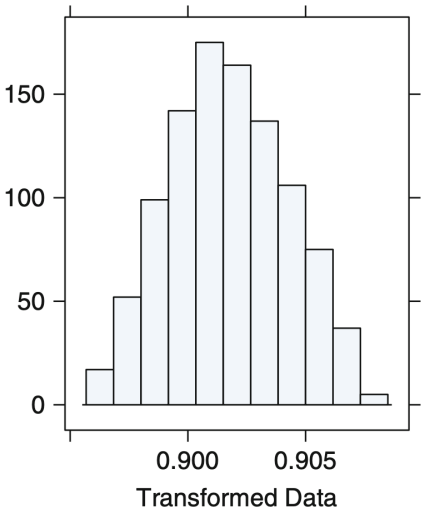




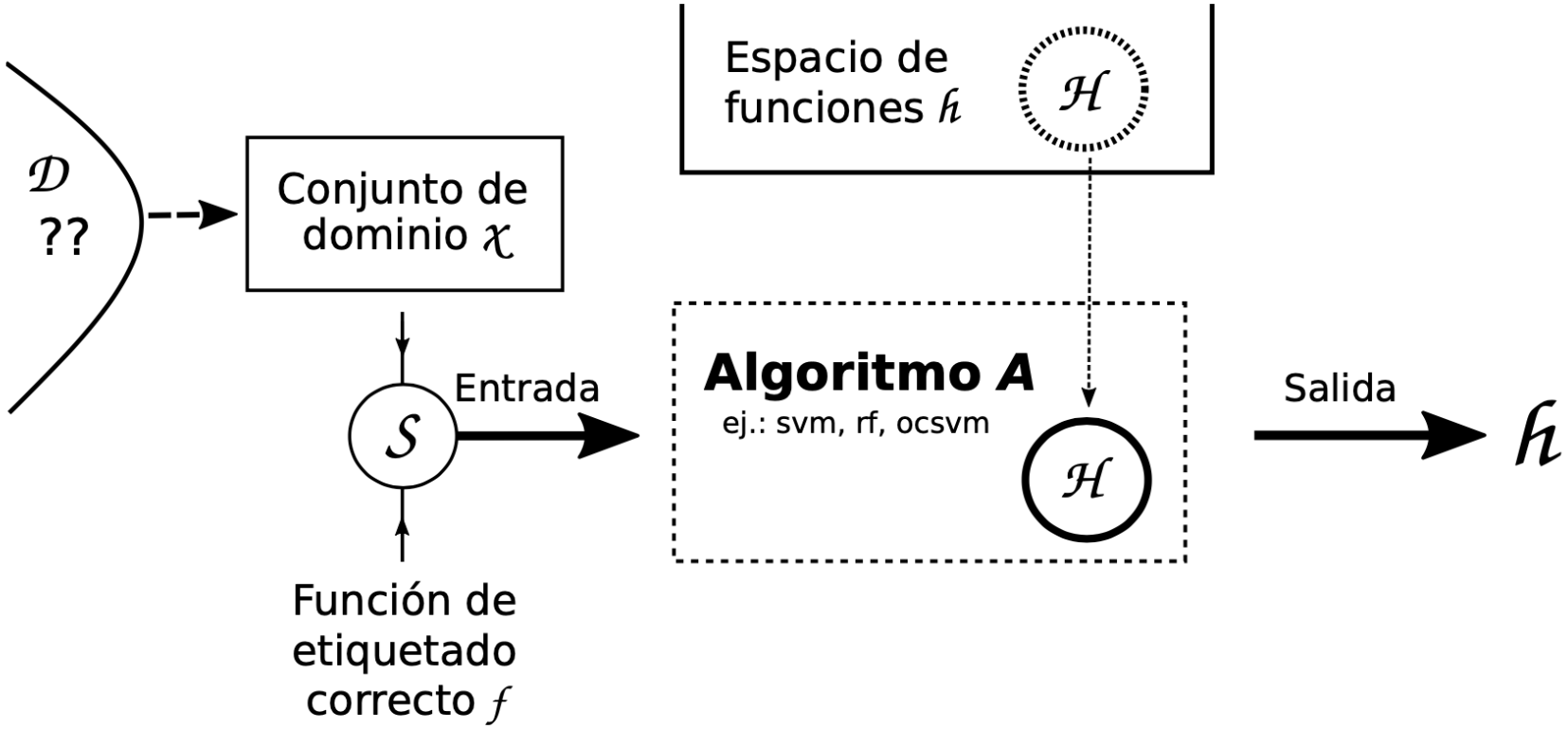






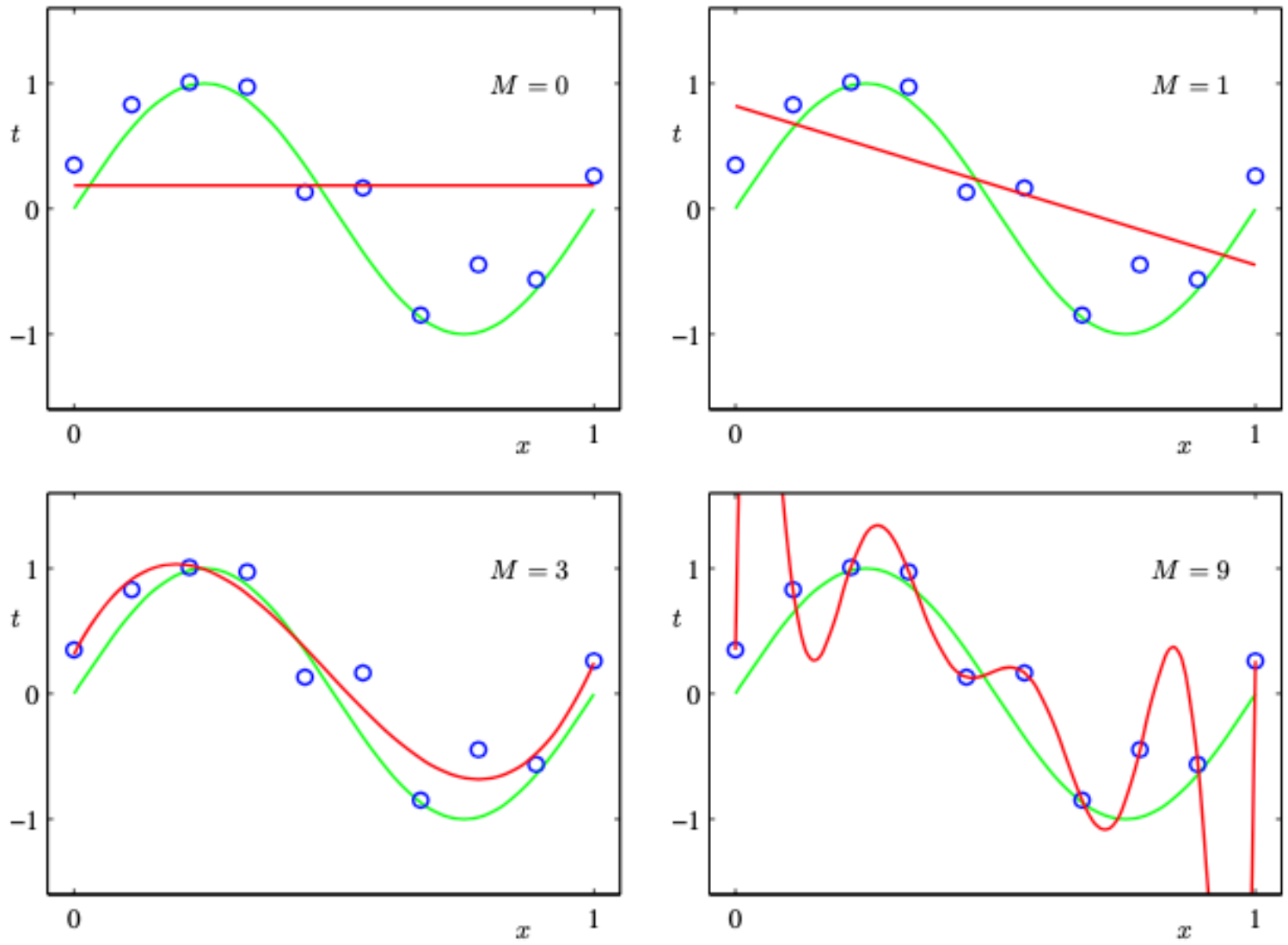


Supervised Learning conceptual model



Técnicas de Aprendizaje Automático Aplicadas a Simulaciones Numéricas de Colisiones de Material Granular Poroso, Daniela N. Rim, Seminario de investigación y/o desarrollo, FCEN, UNCUyo, 2019. Basado en 2014 - Shalev-Shwartz - Understanding machine learning= From theory to algorithms.

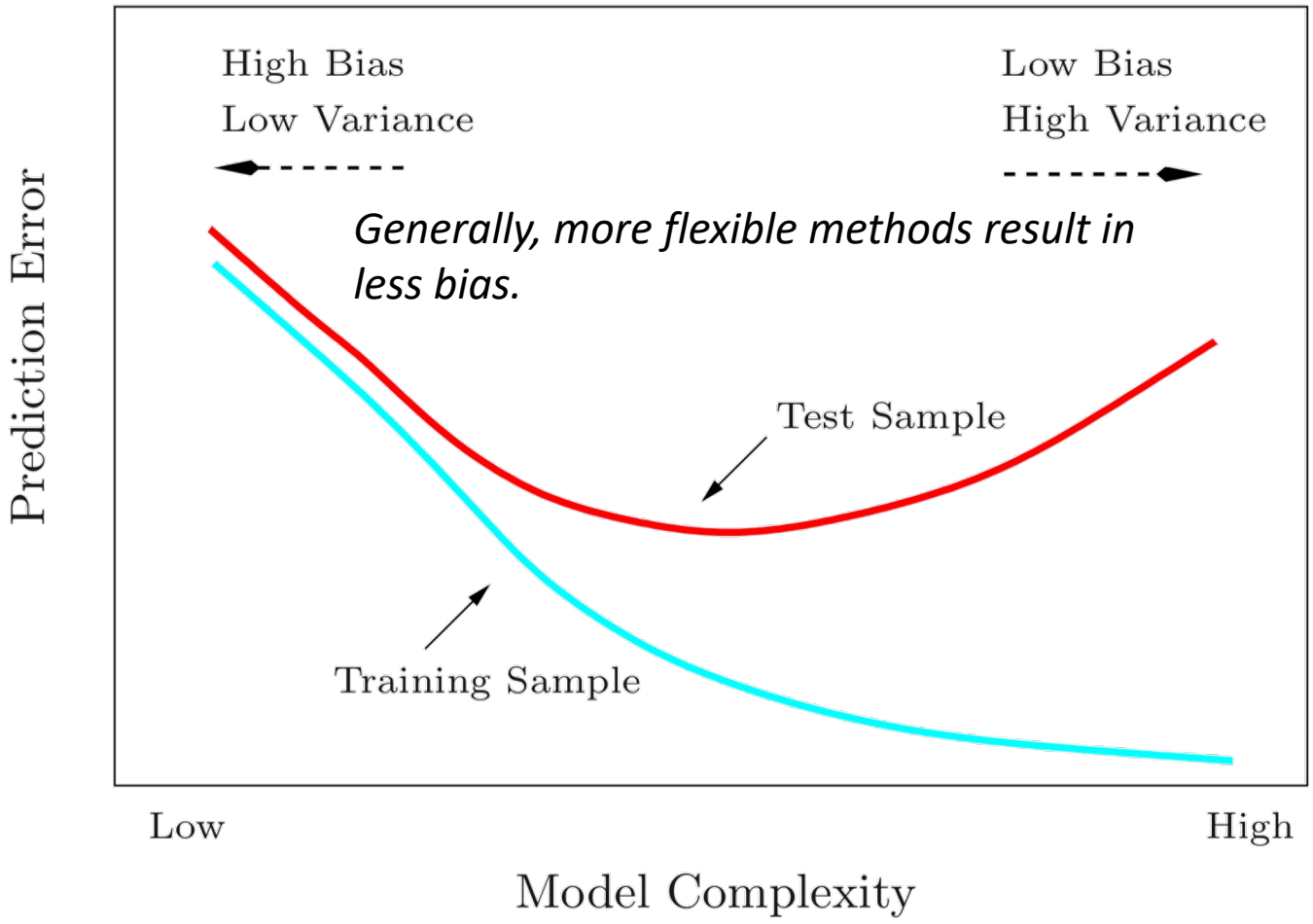
Complexity, capacity, flexibility, df



The Bias–Variance Tradeoff

$$
\begin{aligned}
\mathrm{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\varepsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - \mathrm{E}\hat{f}(x_0)]^2 \\
&= \sigma_\varepsilon^2 + \mathrm{Bias}^2(\hat{f}(x_0)) + \mathrm{Var}(\hat{f}(x_0)) \\
&= \mathrm{Irreducible\ Error} + \mathrm{Bias}^2 + \mathrm{Variance}.
\end{aligned}
$$

The Bias–Variance Tradeoff



$$\mathrm{E}\left[\left(y-\hat{f}\left(x\right)\right)^{2}\right]=\left(\mathrm{Bias}\left[\hat{f}\left(x\right)\right]\right)^{2}+\mathrm{Var}\left[\hat{f}\left(x\right)\right]+\sigma^{2}$$

where

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right]=\mathrm{E}\left[\hat{f}\left(x\right)\right]-\mathrm{E}\left[f(x)\right]$$

and

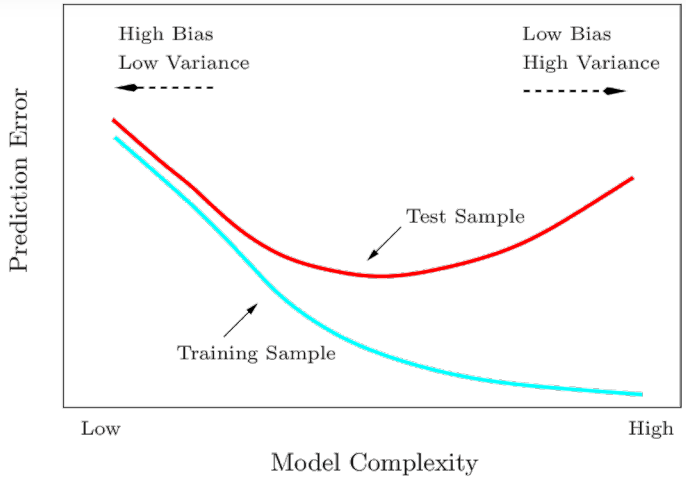
$$\mathrm{Var}\left[\hat{f}\left(x\right)\right]=\mathrm{E}[\hat{f}\left(x\right)^{2}]-\mathrm{E}[\hat{f}\left(x\right)]^{2}.$$

Towards a good model

- Model training: estimating the parameters of the model that lower a given loss function for that particular training data. *Optimization error/Loss/Cost.*

- Model selection: estimating the performance of different models in order to choose the best one: *validation*.

- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data. *Test.*

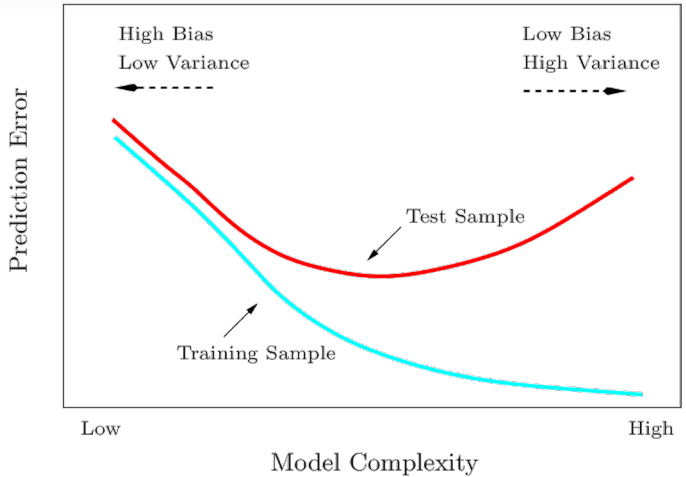


Towards a good model

- Model training: estimating the parameters of the model that lower a given loss function for that particular training data. *Optimization error/Loss/Cost.*

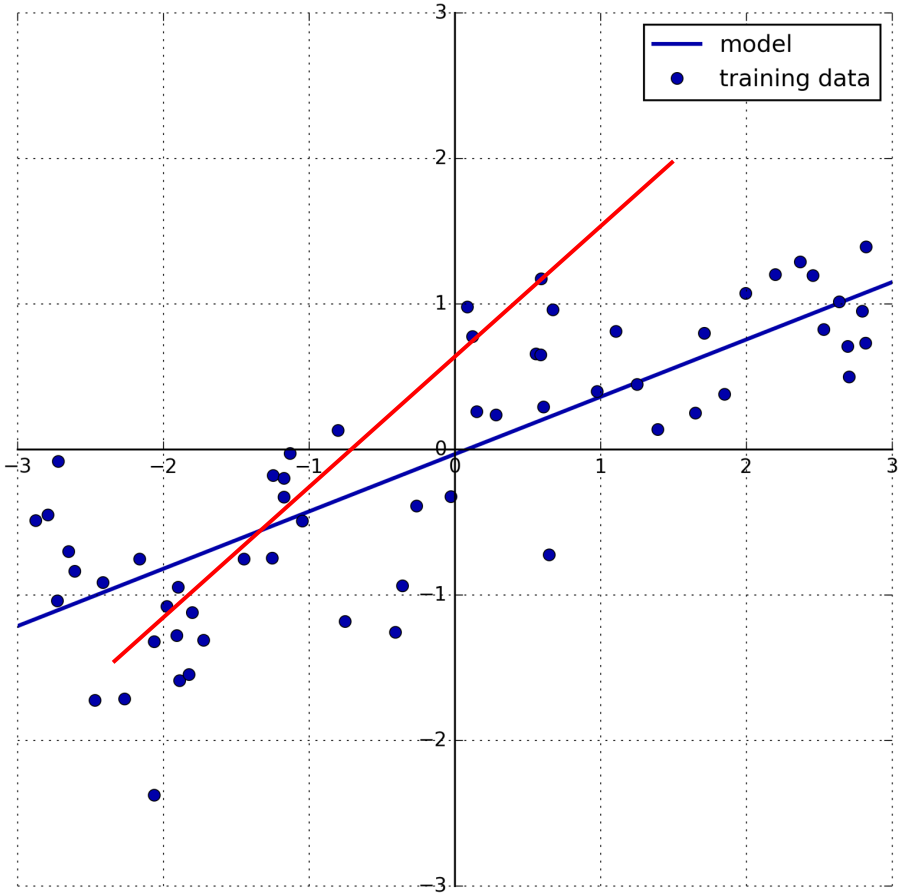
- Model selection: estimating the performance of different models in order to choose the best one: *validation*.

- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data. *Test.*



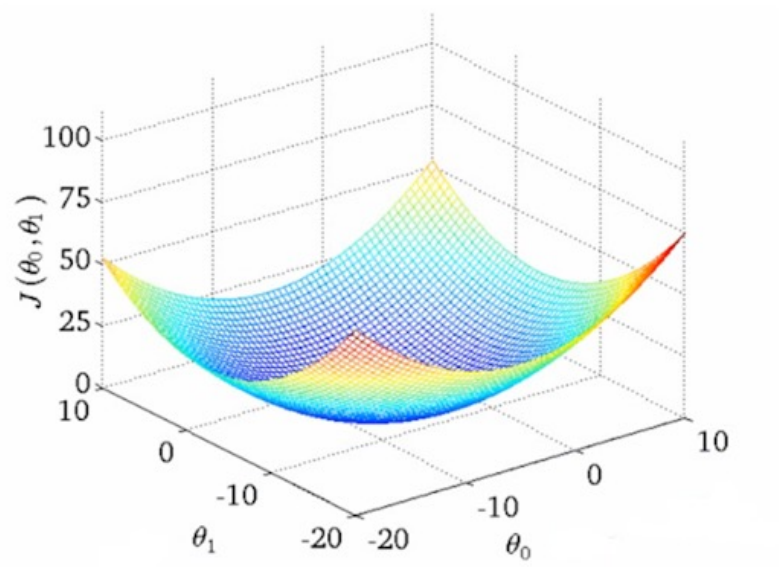
Loss

aka, cost function

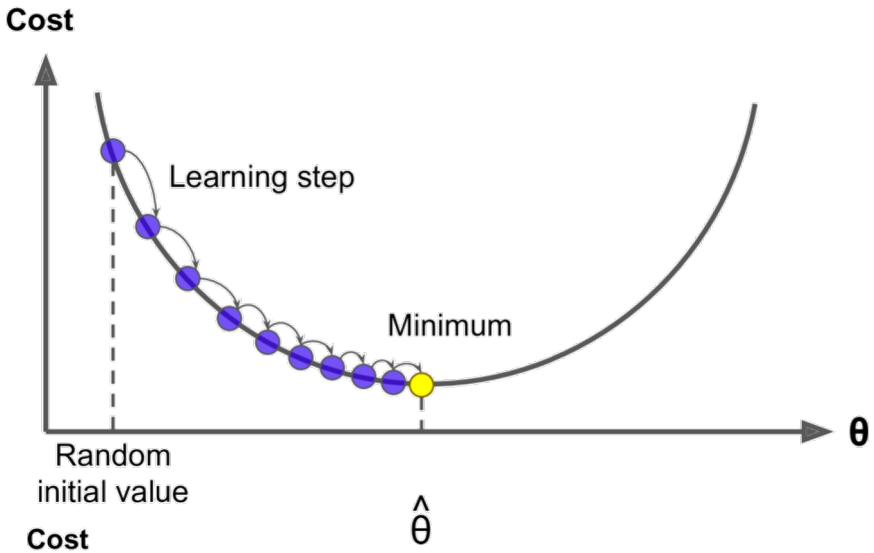


Mean Squared Error Loss: **MSE(θ)**

$$J(b_0, b_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$



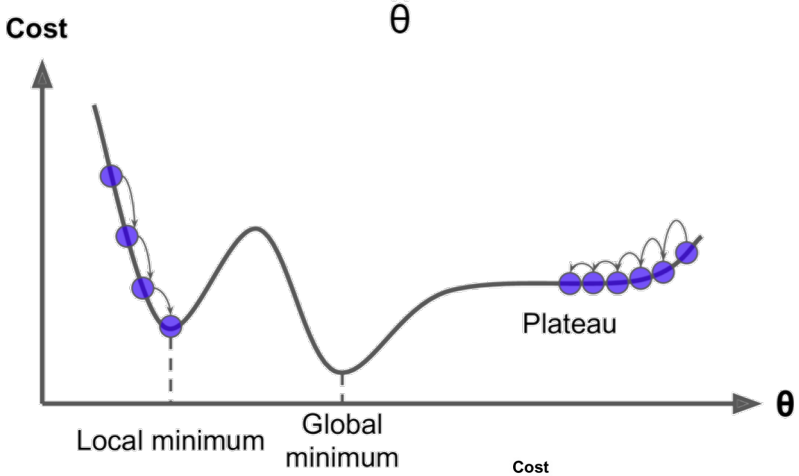
Gradient descent

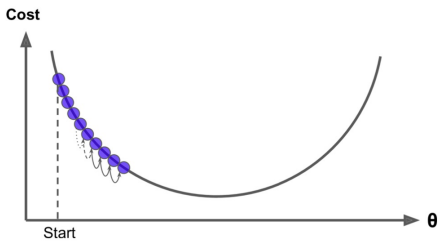


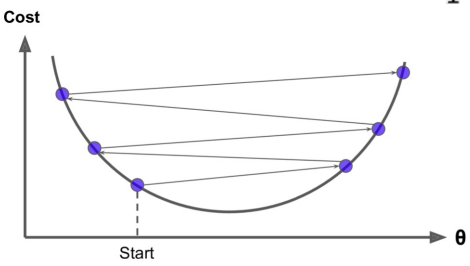
$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

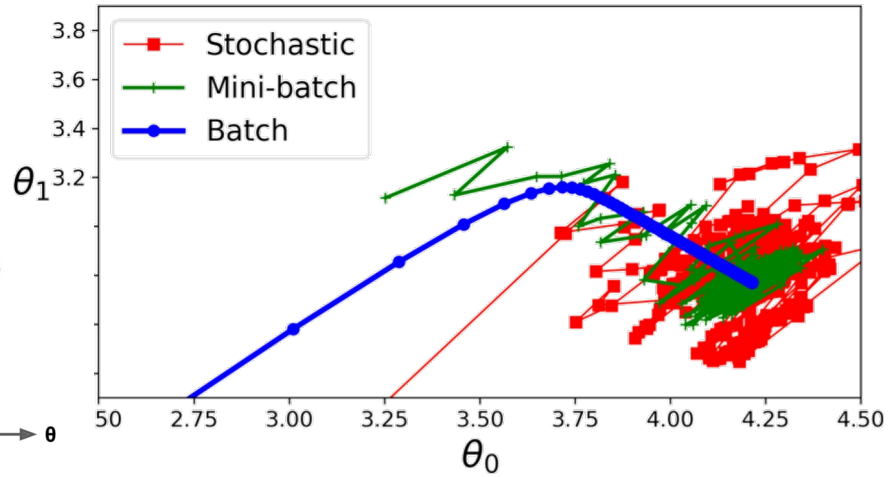
$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

Learning Rate







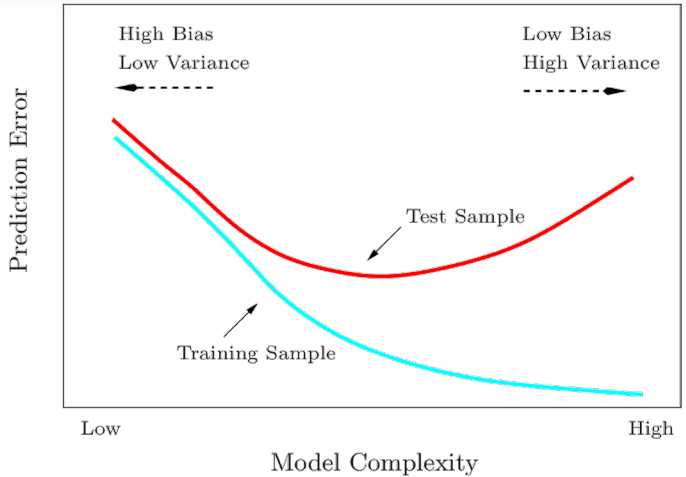


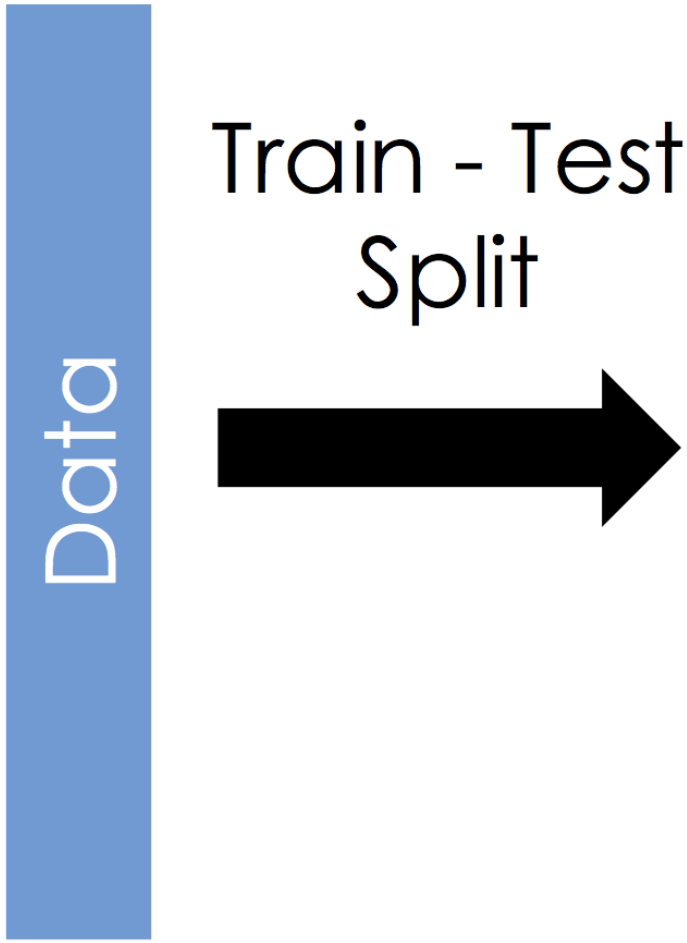
Towards a good model

- Model training: estimating the parameters of the model that lower a given loss function for that particular training data. *Optimization error/Loss/Cost.*

- Model selection: estimating the performance of different models in order to choose the best one. V*alidation error*.

- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data. *Test error.*





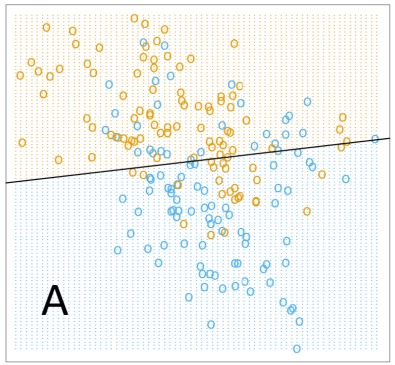
Parámetros

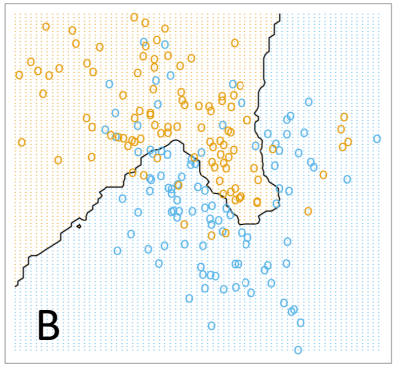
Y si hay "suerte" y los casos difíciles quedan en Train?

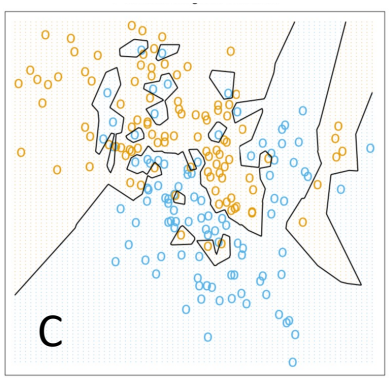
~~Hiperparámetros~~

Cualquier decisión sobre los modelos hecha a partir de Test, filtra información de los datos hacia el modelo, y por lo tanto deja de ser una medida independiente del desempeño del mismo. Pasa inclusive con la exploración visual (EDA).

Under/Overfitting

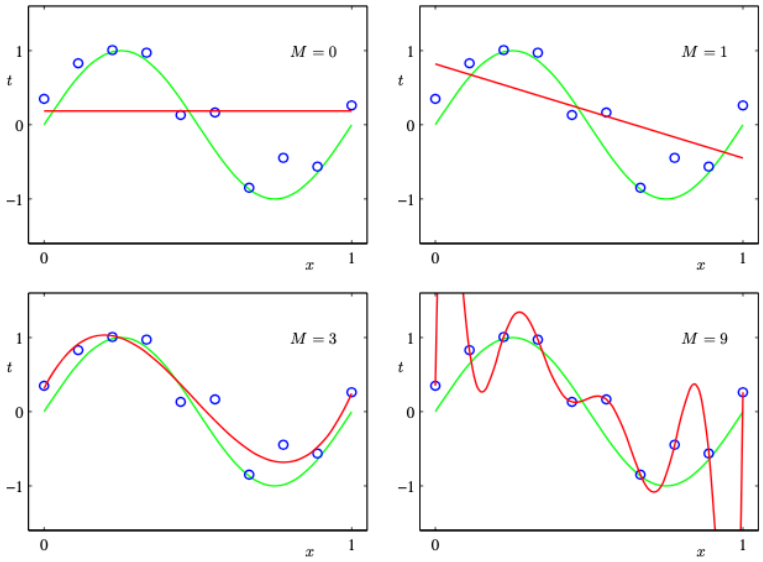




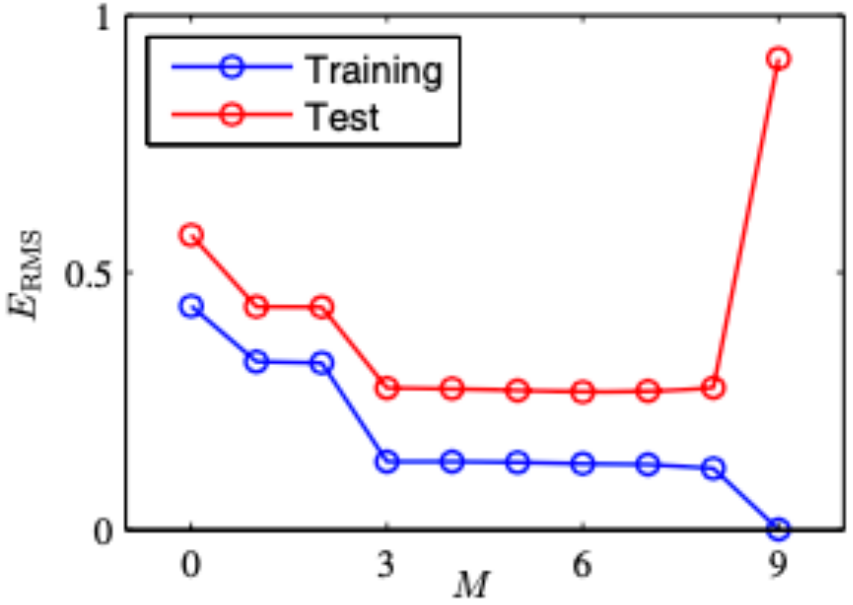


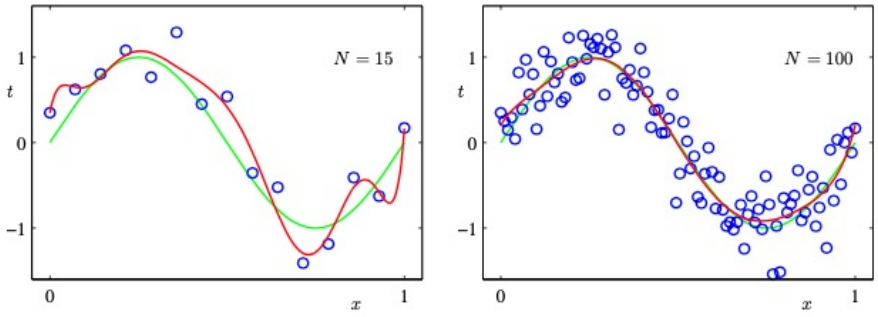
- Very low training error
- Training error much lower than test error
- High variance

Complexity/capacity/flexibility/df



	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43



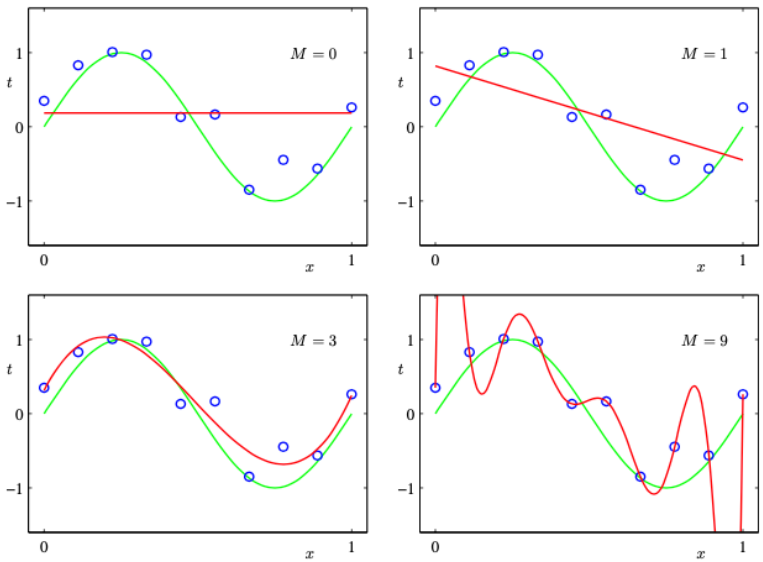


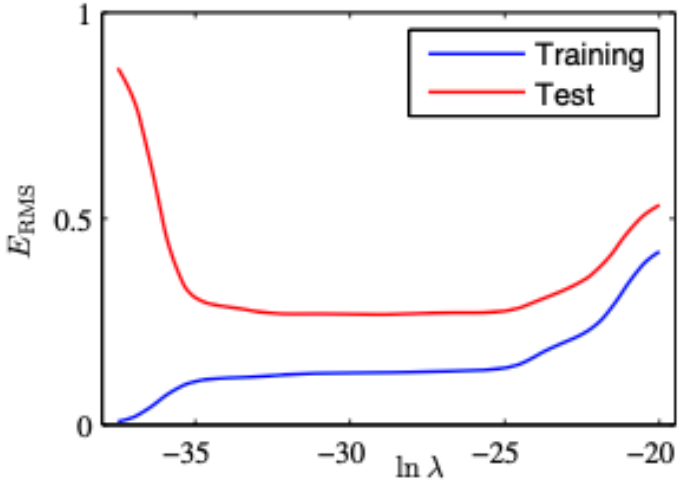
Overfit? More data!

Regularization aka, Shrinkage

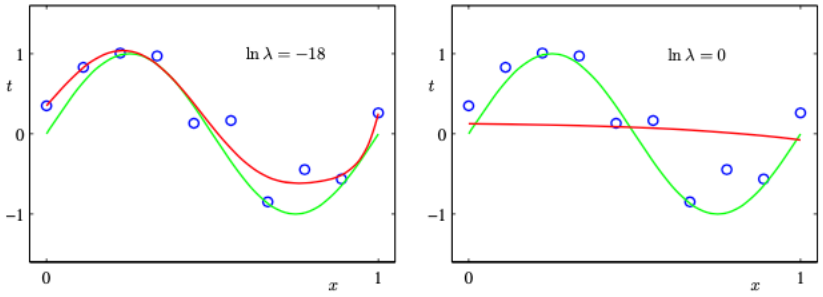
Penalty!

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$





	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

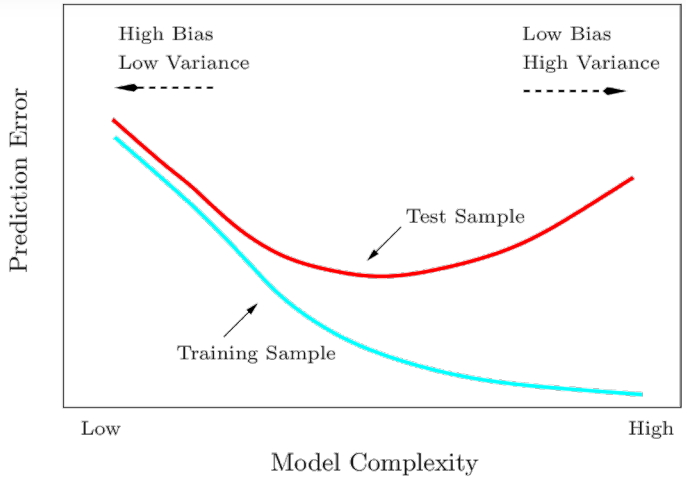


Towards a good model

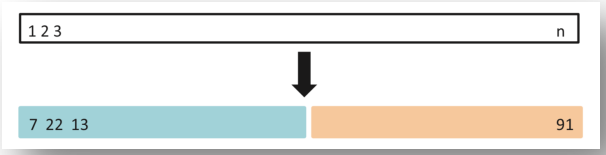
- Model training: estimating the parameters of the model that lower a given loss function for that particular training data. *Optimization error/Loss/Cost.*

- Model selection: estimating the performance of different models in order to choose the best one. V*alidation error*.

- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data. Test error.

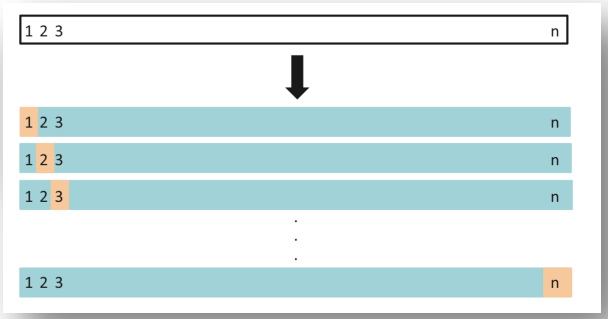


Resampling methods



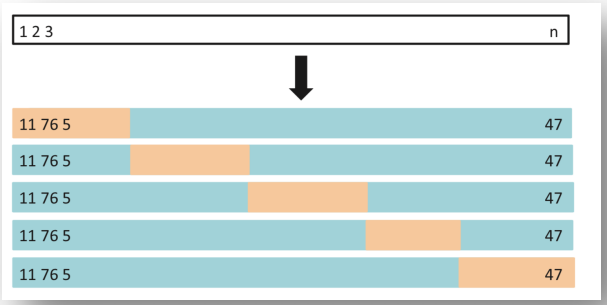
Validation set

model_selection.LeaveOneOut

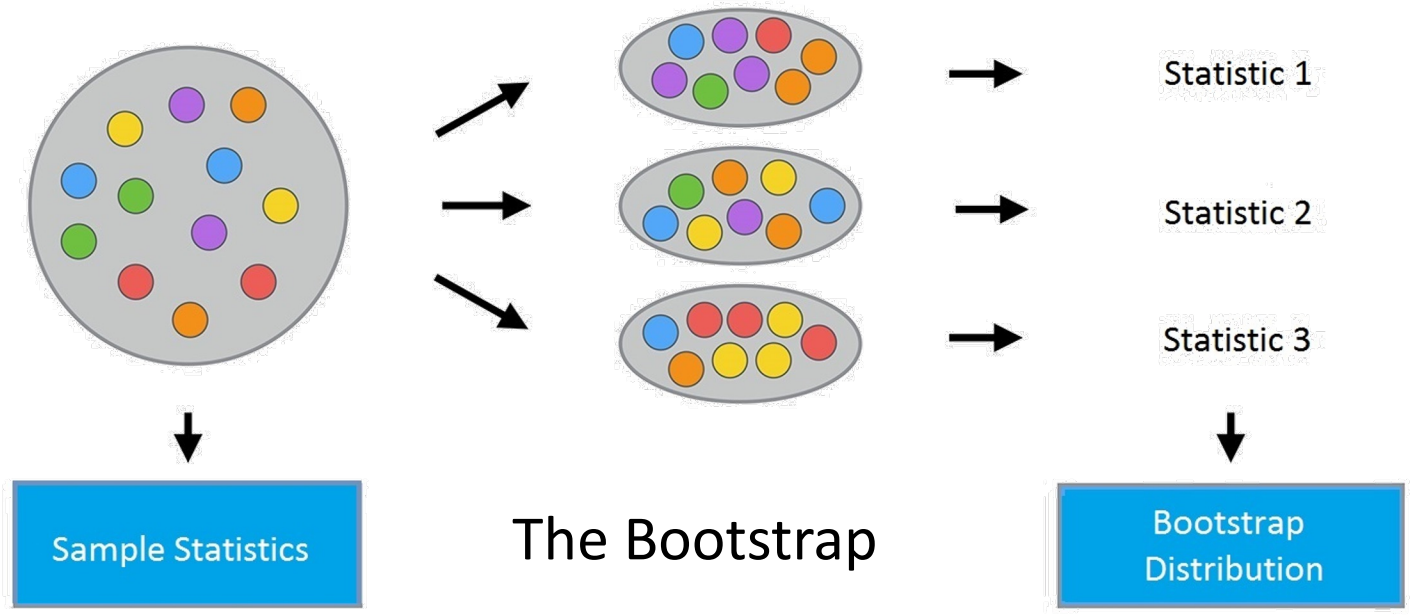


LOOCV

model_selection.Kfold
model_selection.cross_validate



K-fold cross-validation



The Bootstrap

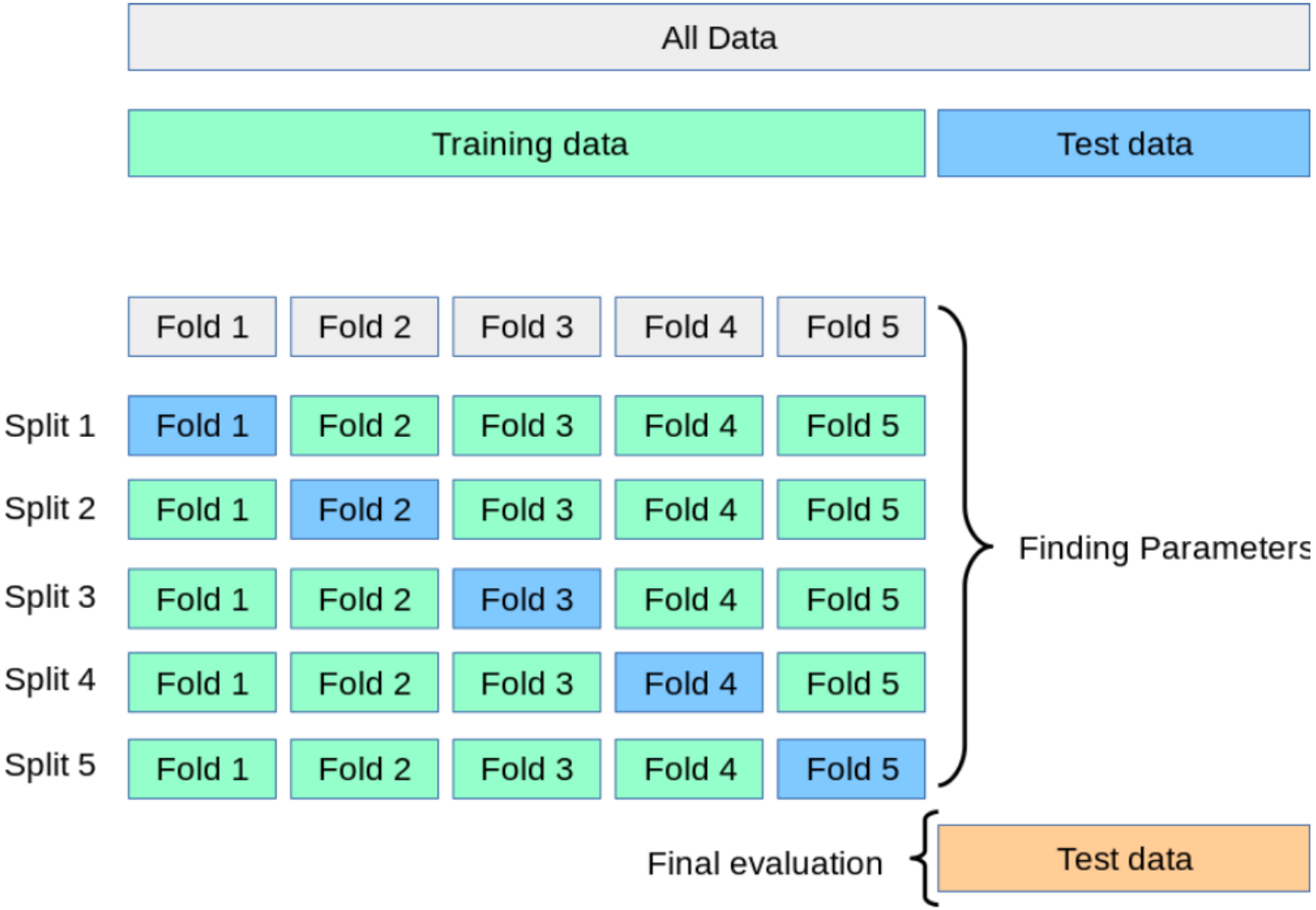
Cross-validation

- Statistical (resampling) method to estimate the generalization error
- More stable and accurate than only train/test
- Gives an idea of the *variance* of the performance metric
- More resources needed to compute

sklearn.model_selection: Model Selection

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

k-fold CV


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

k-fold CV with scikit learn

In[4]:

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

iris = load_iris()
logreg = LogisticRegression()

scores = cross_val_score(logreg, iris.data, iris.target)
print("Cross-validation scores: {}".format(scores))
```

Out[4]:

```
Cross-validation scores: [ 0.961  0.922  0.958]
```

In[5]:

```
scores = cross_val_score(logreg, iris.data, iris.target, cv=5)
print("Cross-validation scores: {}".format(scores))
```

Out[5]:

```
Cross-validation scores: [ 1.     0.967  0.933  0.9    1.   ]
```

CV nos da una idea de la varianza del score

In[6]:

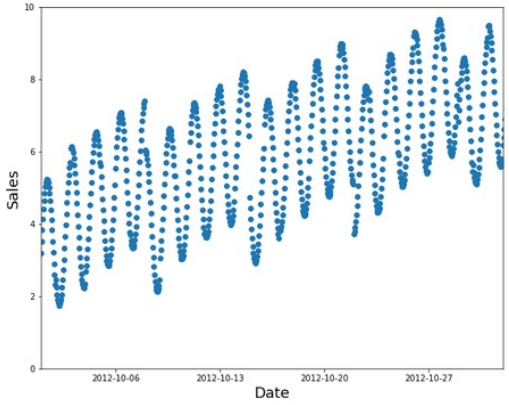
```
print("Average cross-validation score: {:.2f}".format(scores.mean()))
```

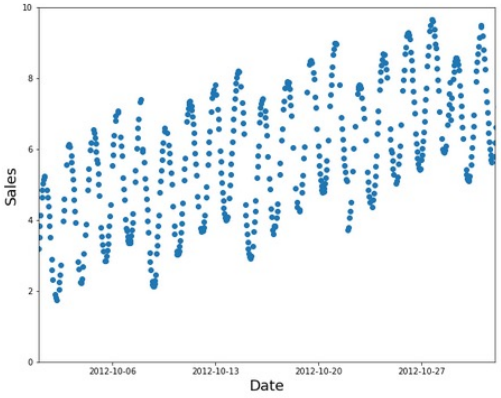
Out[6]:

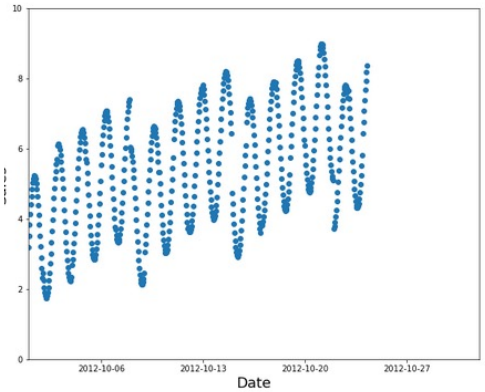
```
Average cross-validation score: 0.96
```

CV caveats: Choosing a good validation set

- Validation set should mimic test set
  - unseen data?
  - ordered data?





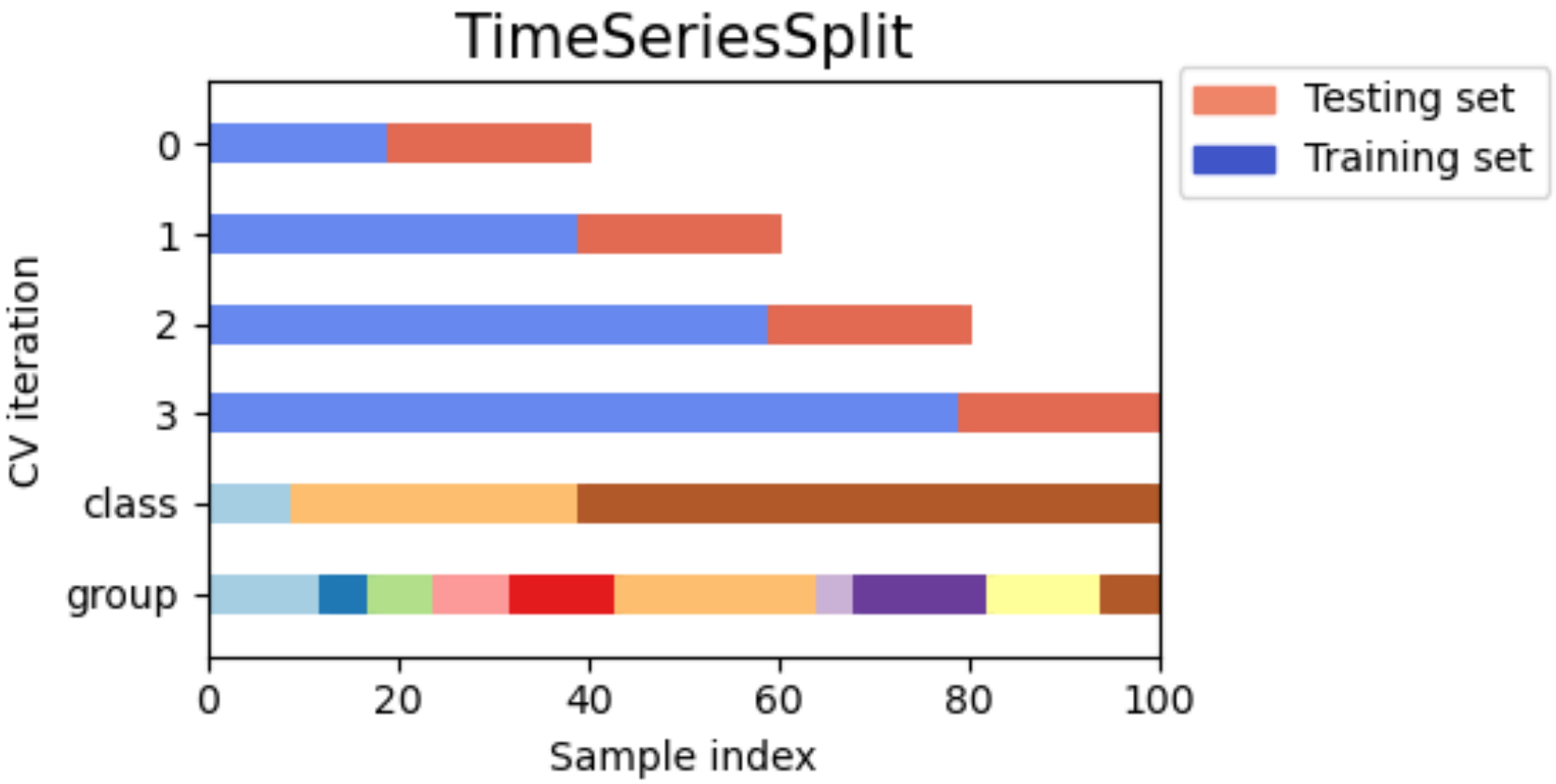


Training

Validation

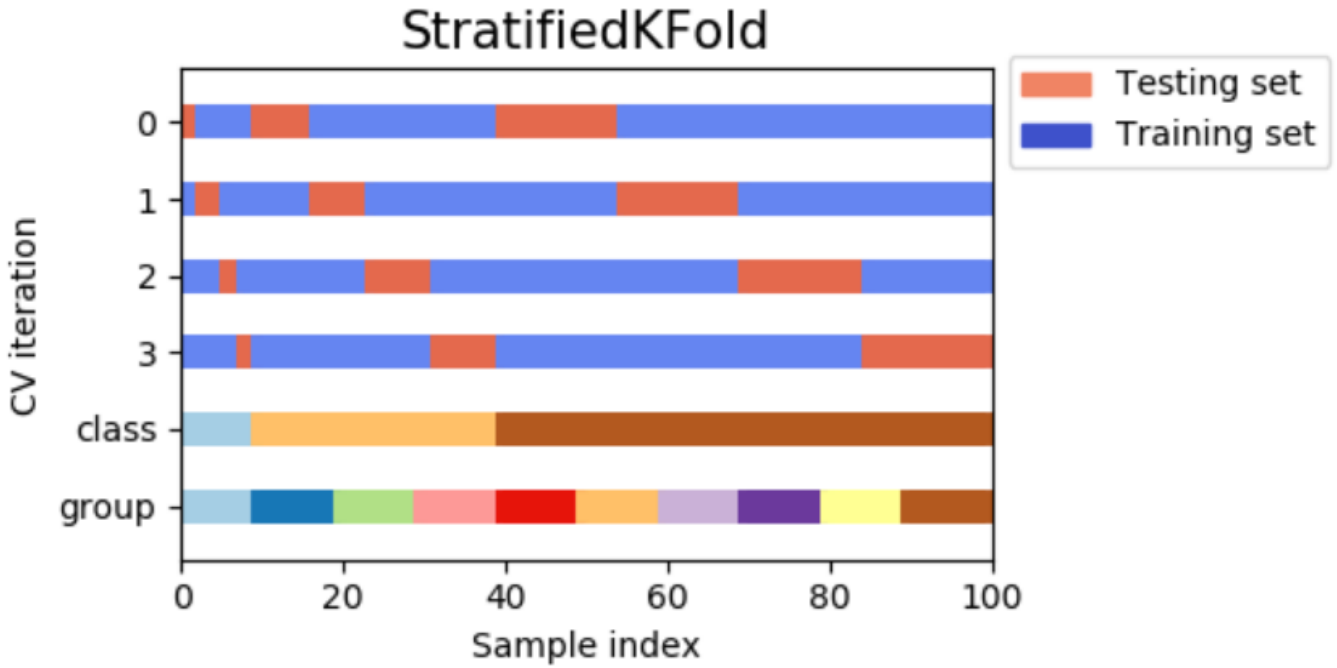
[How (and why) to create a good validation set, by Rachel Thomas (2017)]

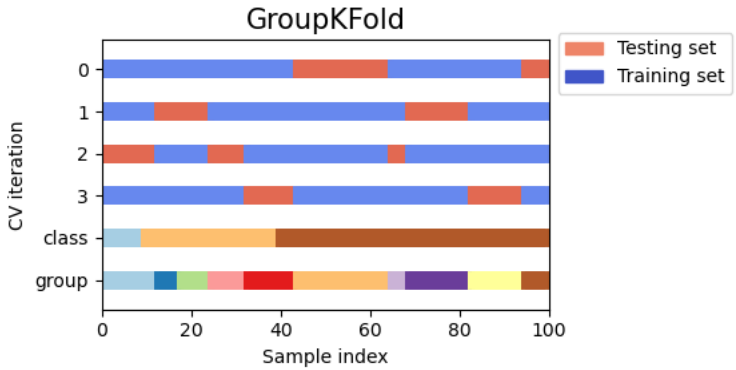
Times series shouldn't be shuffled

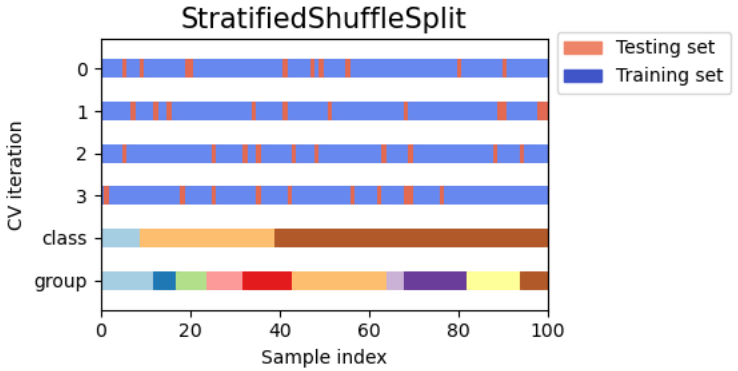


Stratified CV, etc.

Perform a k-CV within the different classes of target variables







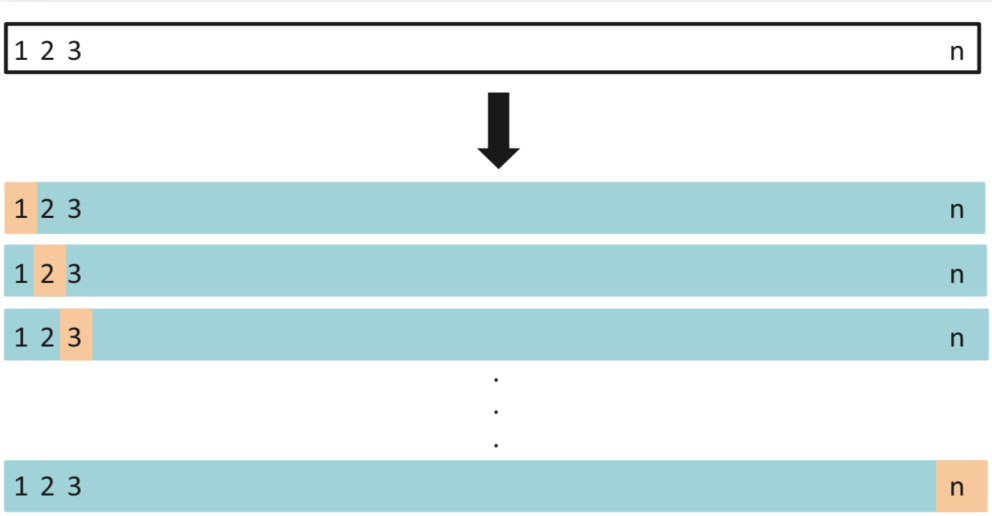
Leave-one-out CV

In[13]:

```
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()
scores = cross_val_score(logreg, iris.data, iris.target, cv=loo)
print("Number of cv iterations: ", len(scores))
print("Mean accuracy: {:.2f}".format(scores.mean()))
```

Out[13]:

```
Number of cv iterations:  150
Mean accuracy: 0.95
```



Hyperparameter fine-tuning

- Parameters vs. hyperparameters
- Systematic search of best hyperp. values
- Do not touch test partition
- Grid search vs. Random search

	C = 0.001	C = 0.01	…	C = 10
gamma=0.001	SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	…	SVC(C=10, gamma=0.001)
gamma=0.01	SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	…	SVC(C=10, gamma=0.01)
…	…	…	…	…
gamma=100	SVC(C=0.001, gamma=100)	SVC(C=0.01, gamma=100)	…	SVC(C=10, gamma=100)

Grid search + CV

In[18]:

```
# naive grid search implementation
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, random_state=0)
print("Size of training set: {}   size of test set: {}".format(
      X_train.shape[0], X_test.shape[0]))

best_score = 0

for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
    for C in [0.001, 0.01, 0.1, 1, 10, 100]:
        # for each combination of parameters, train an SVC
        svm = SVC(gamma=gamma, C=C)
        svm.fit(X_train, y_train)
        # evaluate the SVC on the test set
        score = svm.score(X_test, y_test)
        # if we got a better score, store the score and parameters
        if score > best_score:
            best_score = score
            best_parameters = {'C': C, 'gamma': gamma}

print("Best score: {:.2f}".format(best_score))
print("Best parameters: {}".format(best_parameters))
```

Out[18]:

```
Size of training set: 112   size of test set: 38
Best score: 0.97
Best parameters: {'C': 100, 'gamma': 0.001}
```

Grid search + CV

In[21]:

```
    for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
        for C in [0.001, 0.01, 0.1, 1, 10, 100]:
            # for each combination of parameters,
            # train an SVC
            svm = SVC(gamma=gamma, C=C)
            # perform cross-validation
            scores = cross_val_score(svm, X_trainval, y_trainval, cv=5)
            # compute mean cross-validation accuracy
            score = np.mean(scores)
            # if we got a better score, store the score and parameters
            if score > best_score:
                best_score = score
                best_parameters = {'C': C, 'gamma': gamma}
    # rebuild a model on the combined training and validation set
    svm = SVC(**best_parameters)
    svm.fit(X_trainval, y_trainval)
```

Grid search + CV with scikit-learn

In[24]:

```
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}
```

In[25]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
```

In[26]:

```
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, random_state=0)
```

In[27]:

```
grid_search.fit(X_train, y_train)
```

Performance metrics

Regression performance metrics

- Root Mean Square Error

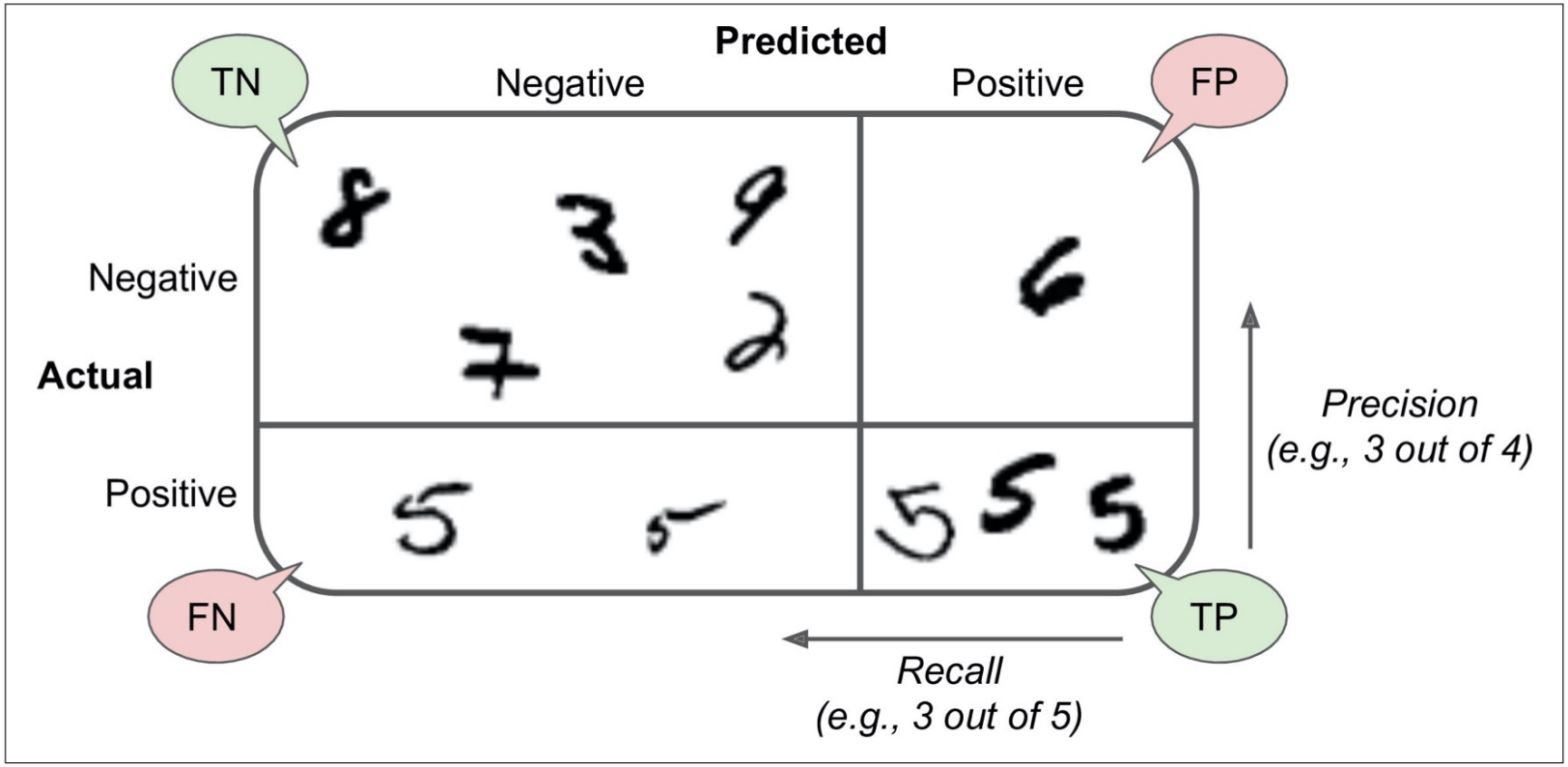
$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

- Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

- R-Squared and *adjusted* R-Squared

Classification - Confusion matrix



$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Confusion matrix

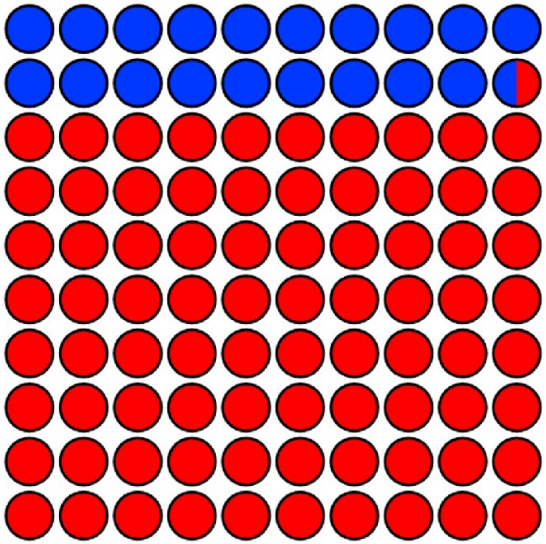
Sources: [5][6][7][8][9][10][11][12][13] view · talk · edit

		Predicted condition			
	Total population = P + N	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P+N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

(Beware of) Accuracy

Or, is the model better than the majority class?

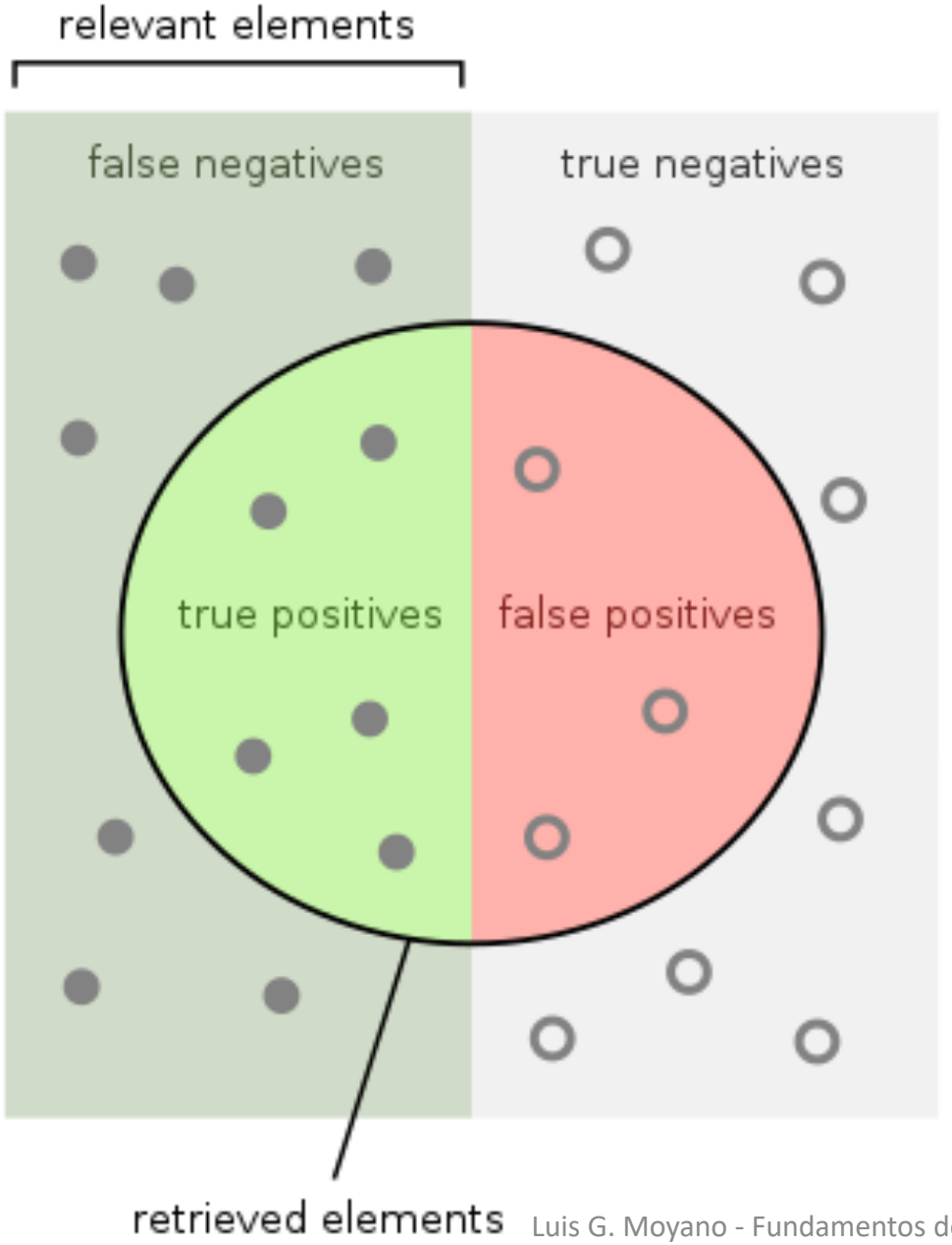
$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ total\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$



- Class imbalance
- Majority class = 80%
- Accuracy = 79% ...is model good?

- If model predicts with accuracy lower than 80%, better to always say 'red'!

Precision and Recall



How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

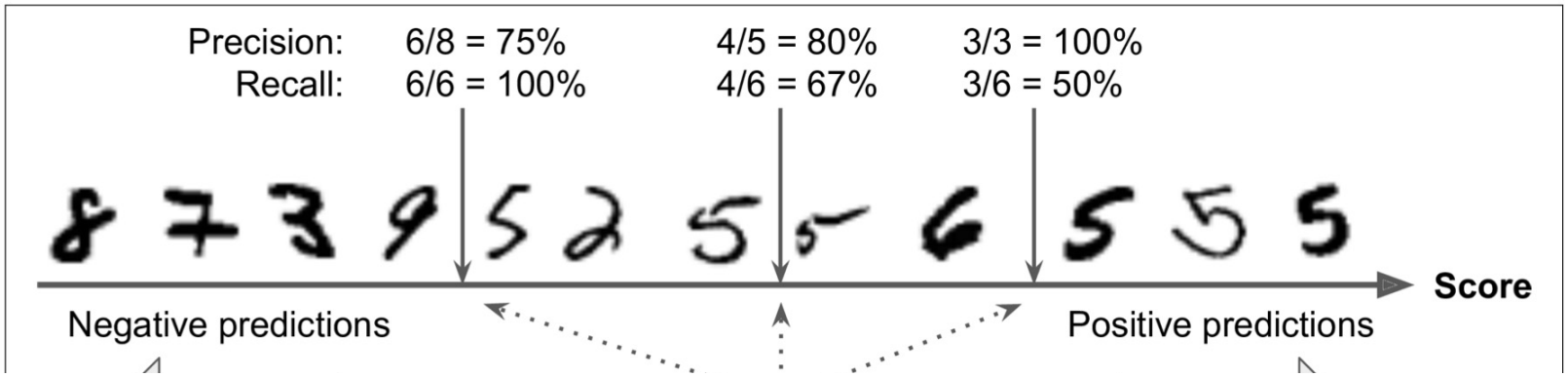
Validity

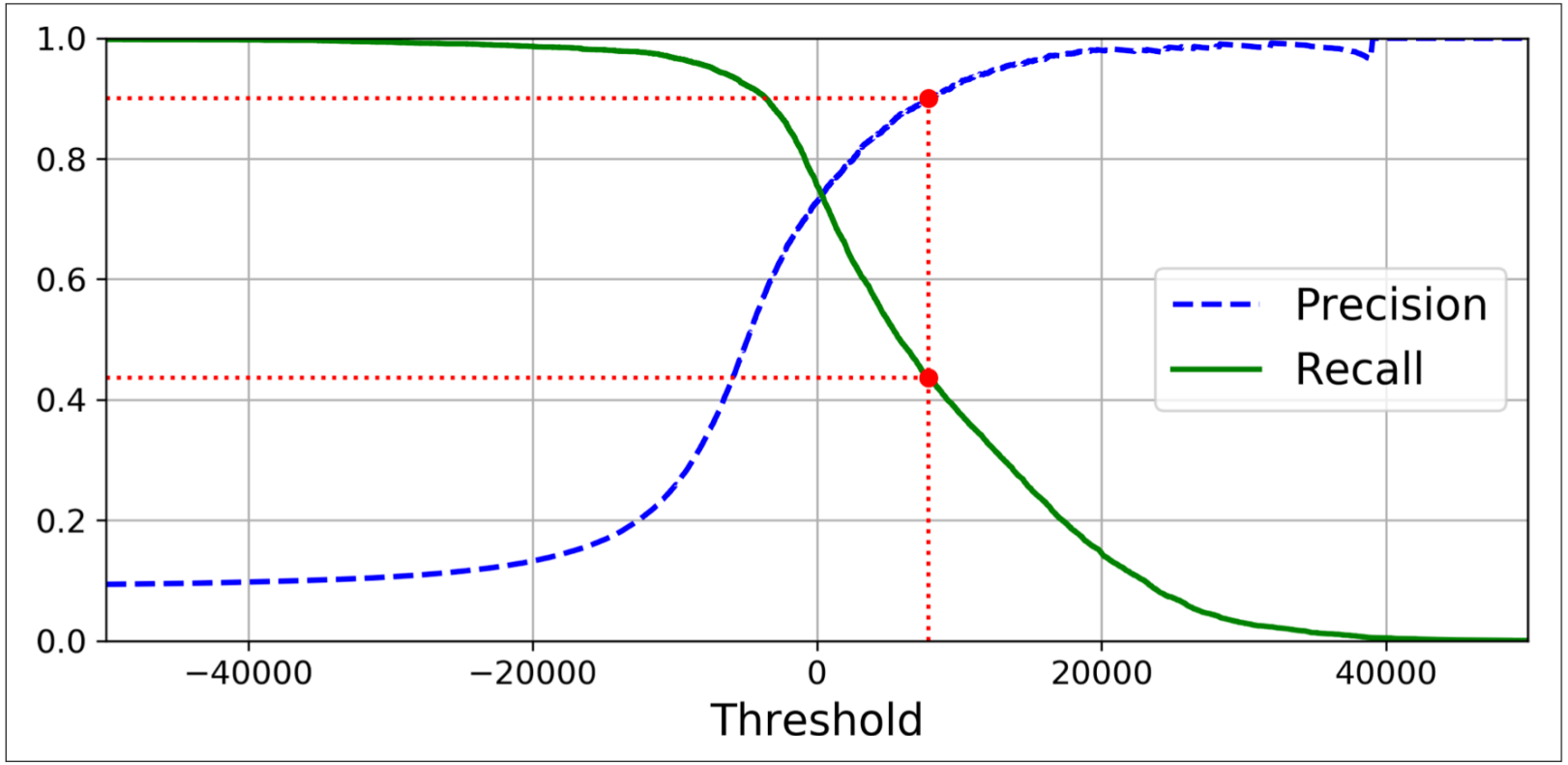
How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Completeness

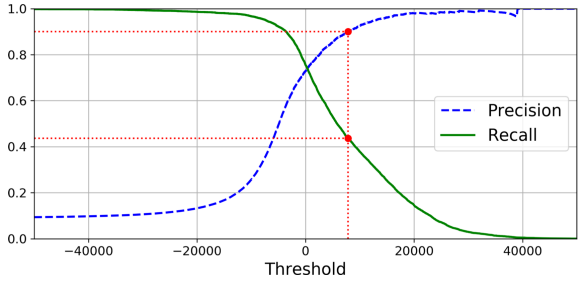
Precision and Recall

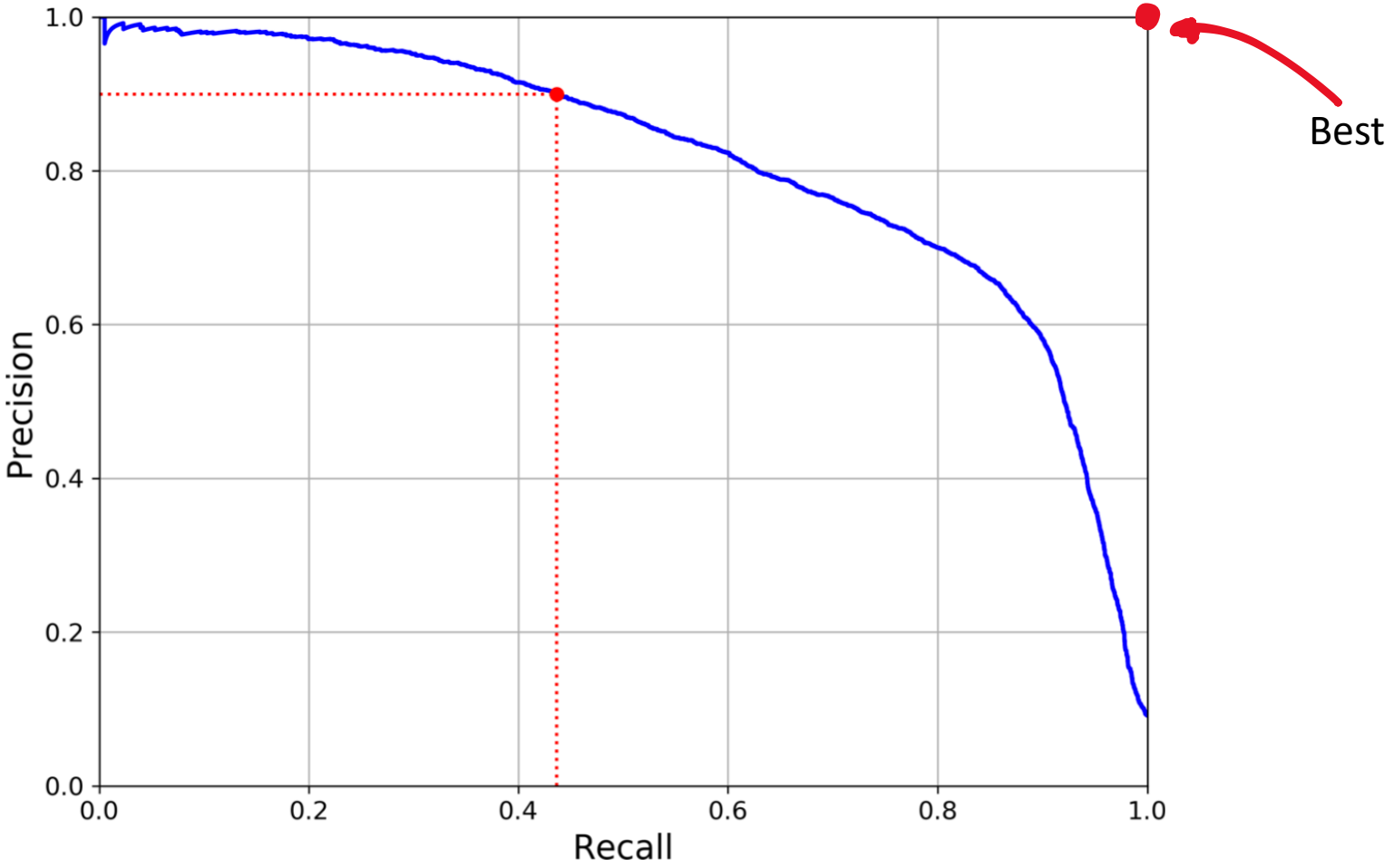




Precision-Recall curve

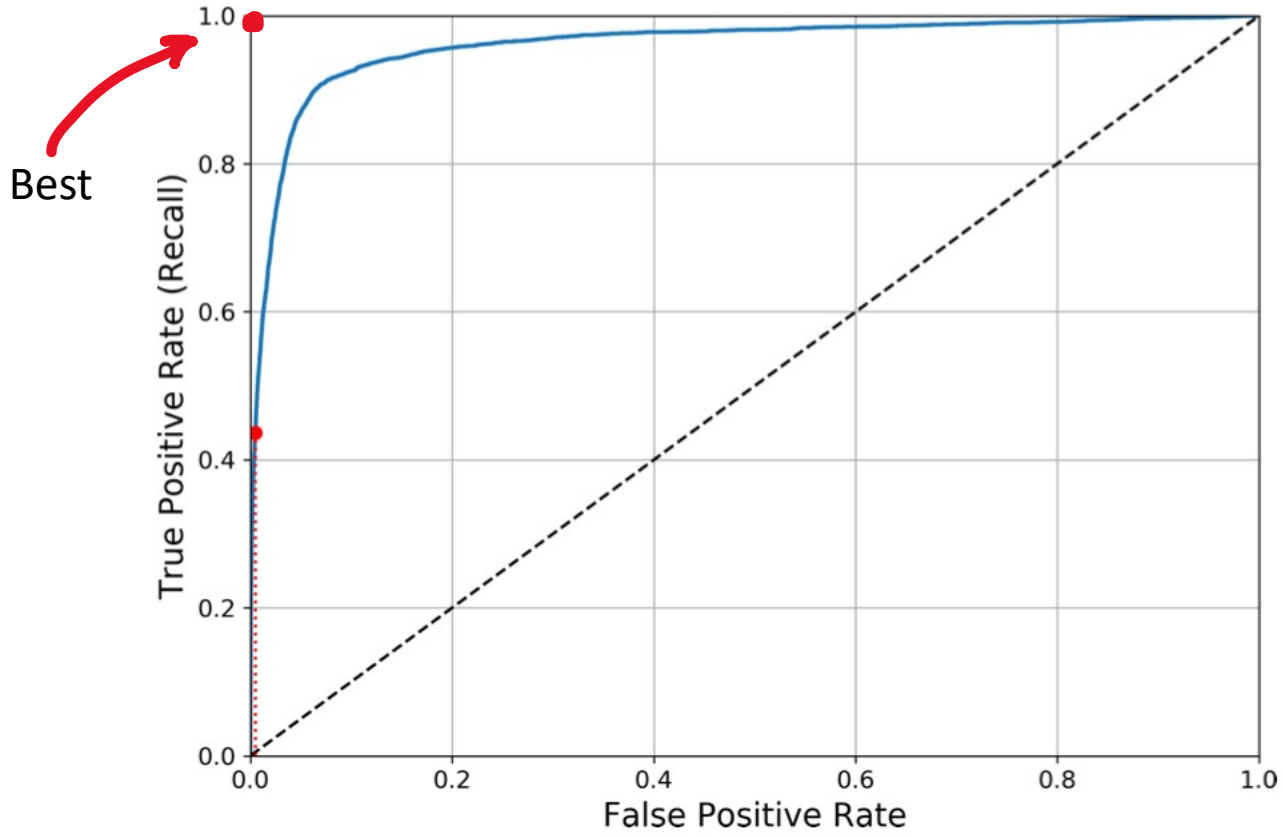
Select the precision/ recall tradeoff that fits your needs





ROC curve and AUC

true positive rate (another name for recall) against the *false positive rate, i.e.* the ratio of negative instances that are incorrectly classified as positive, or 1-*sensitivity.*



AUC=Area under the curve, metric for model comparison

Other single metrics: Gini impurity index, Akaike information criterion (AIC), etc

F-score

Balanced F1-score:

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

- Harmonic mean of Precision and Recall
- Convenient way of summing up both in a single number
- Defined between 0 (both low) and 1 *(both high)
- Particular case of F_{beta}: $F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}}.$
- Will further discuss when talking about *class imbalance*

Resumen de esta clase

- Probability recap
- EDA
- Error
  - Variance vs. bias
  - Loss and optimizers
  - Train/validation/test
  - Overfitting and underfitting
  - Regularization
  - Performance metrics
  - Resampling

Próxima clase: **Data**