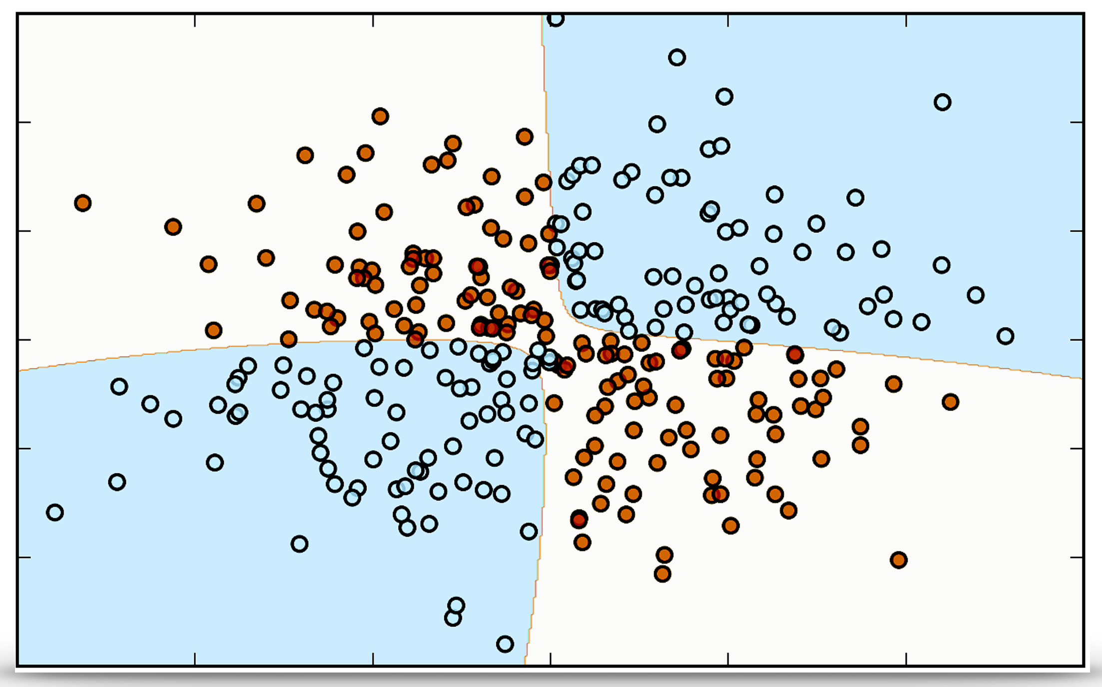
Practical Data Science

An Introduction to Supervised Machine Learning and Pattern Classification: The Big Picture



Sebastian Raschka

Michigan State University

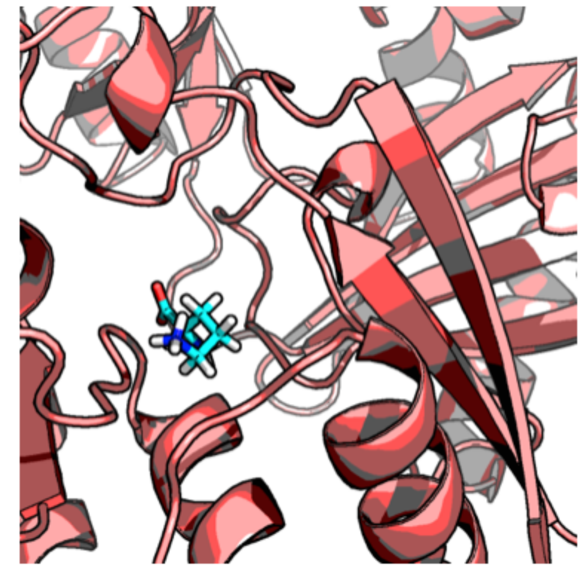
NextGen Bioinformatics Seminars - 2015

Feb. 11, 2015

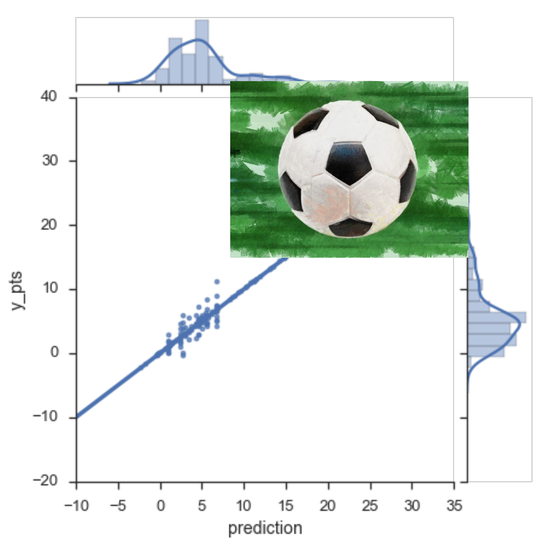
A Little Bit About Myself …

PhD candidate in Dr. L. Kuhn's Lab:

Developing software & methods for
- Protein ligand docking
- Large scale drug/inhibitor discovery



and some other machine learning side-projects …



scikit learn

kaggle

sebastianraschka

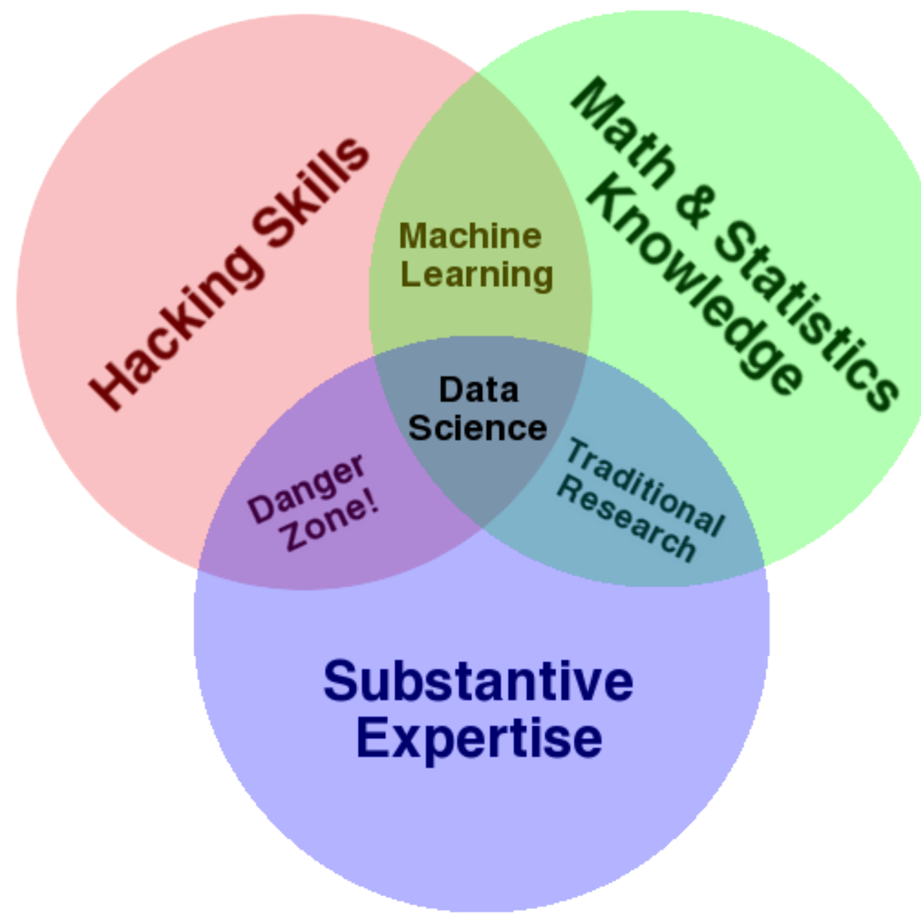
blog webapps projects books about+contact

What is Machine Learning?

"Field of study that gives computers the ability to learn without being explicitly programmed."

(Arthur Samuel, 1959)

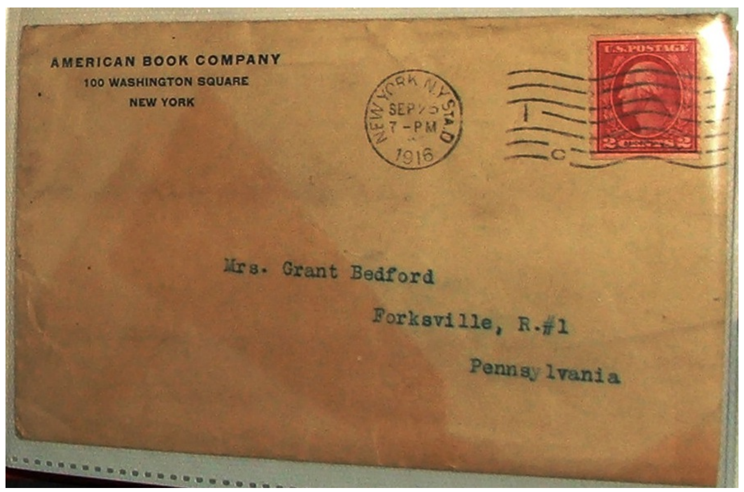
By Phillip Taylor [CC BY 2.0]



http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

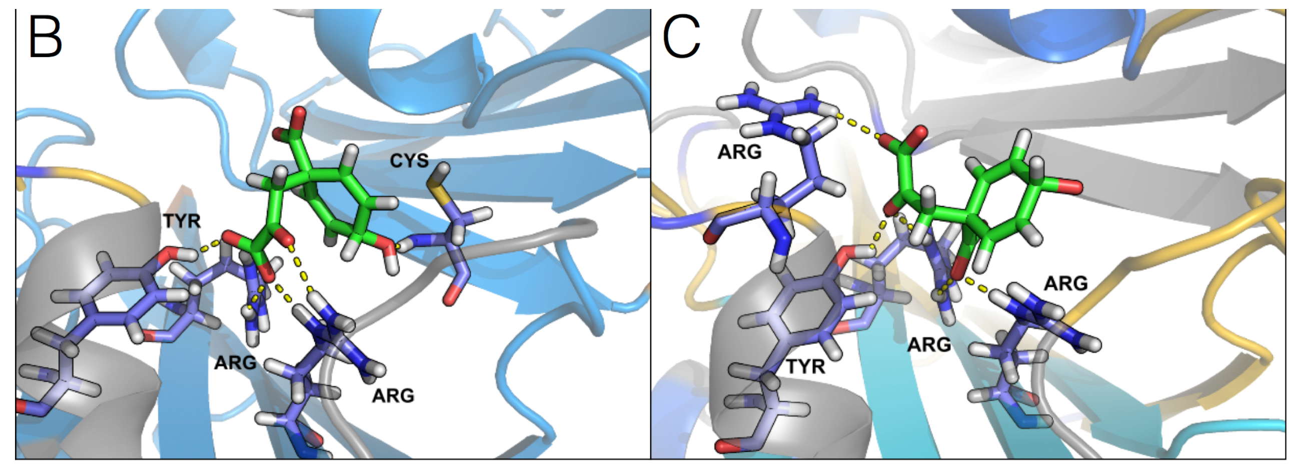
Examples of Machine Learning

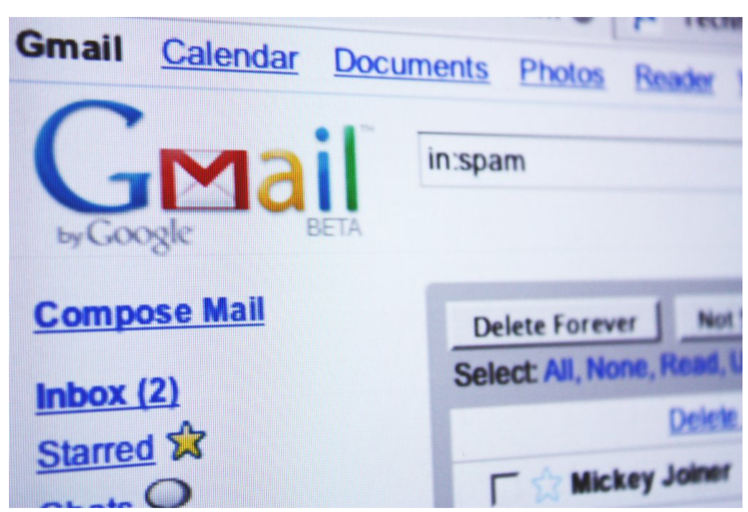
Text Recognition



http://commons.wikimedia.org/wiki/File:American_book_company_1916._letter_envelope-2.JPG#filelinks [public domain]

Biology



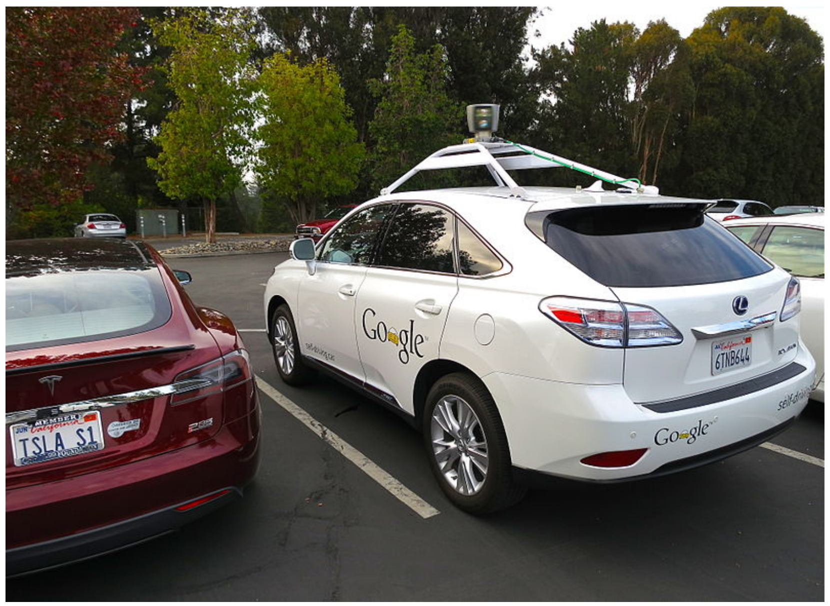


Spam Filtering

https://flic.kr/p/5BLW6G [CC BY 2.0]

Examples of Machine Learning

Self-driving cars

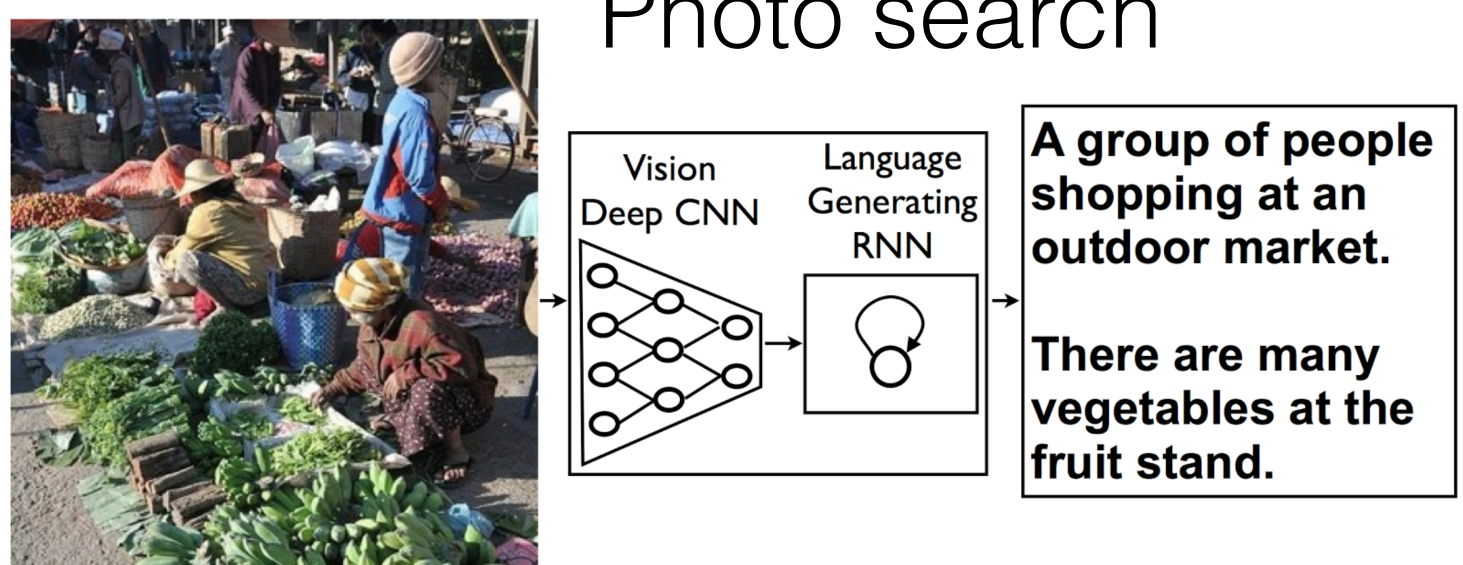


By Steve Jurvetson [CC BY 2.0]

Recommendation systems

http://commons.wikimedia.org/wiki/File:Netflix_logo.svg [public domain]

Photo search



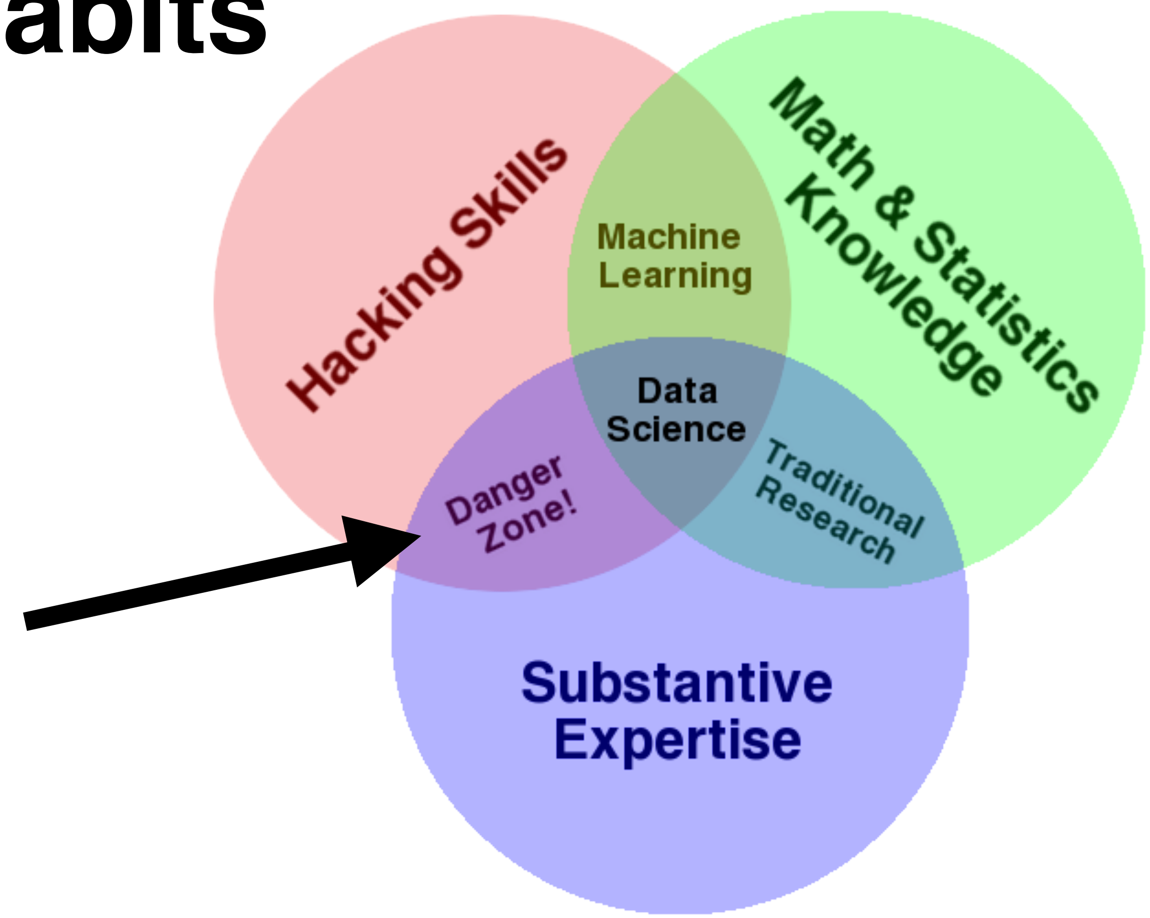
http://googleresearch.blogspot.com/2014/11/a-picture-is-worth-thousand-coherent.html

and many, many more ...

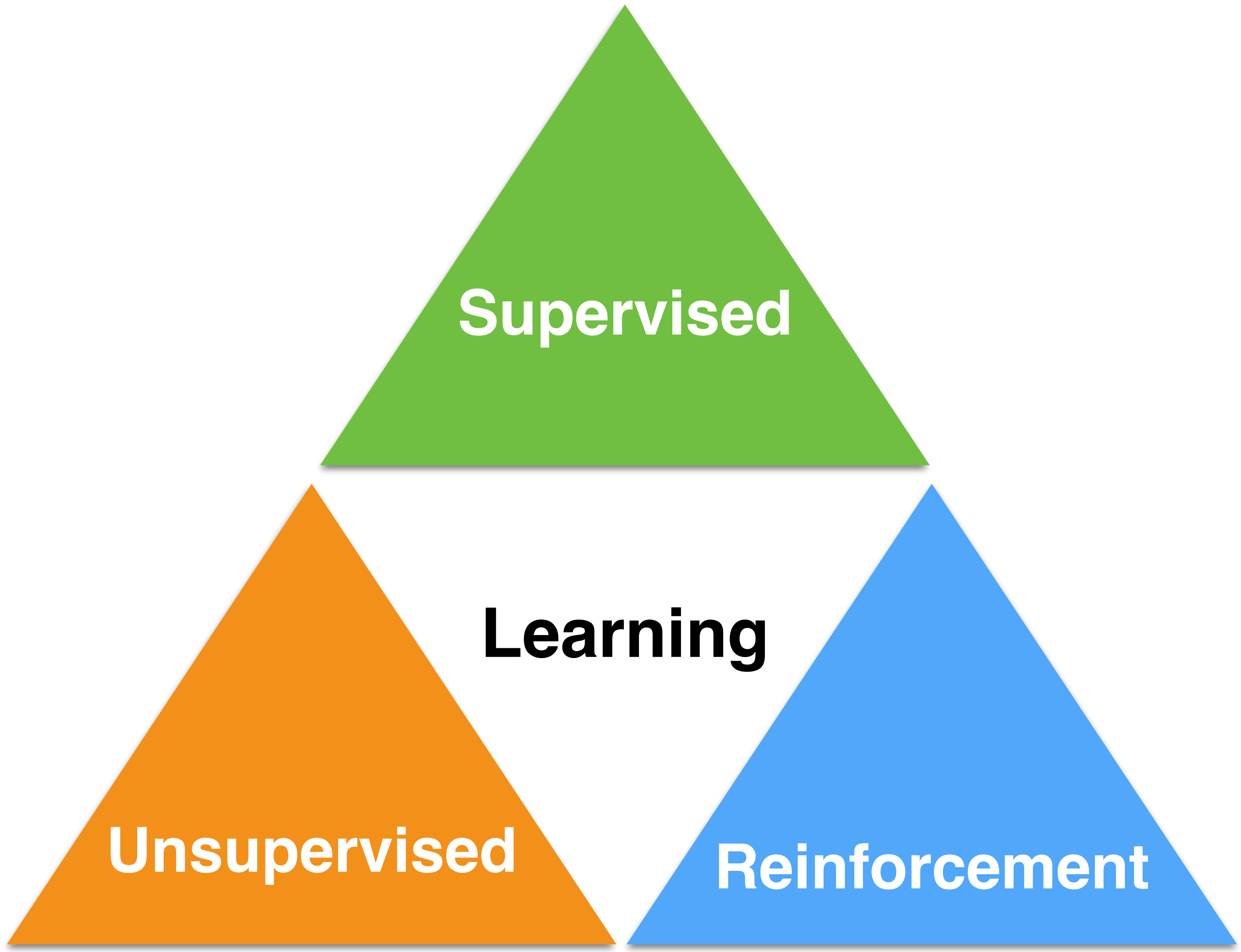
How many of you have used machine learning before?

Our Agenda

- **Concepts and the big picture**
- **Workflow**
- **Practical tips & good habits**



- Labeled data
- Direct feedback
- Predict outcome/future

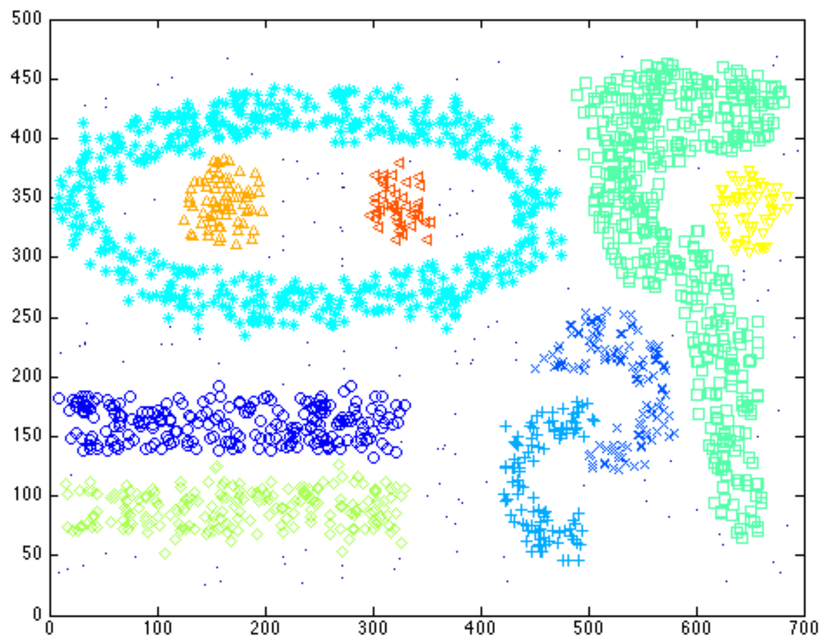


- No labels
- No feedback
- “Find hidden structure”

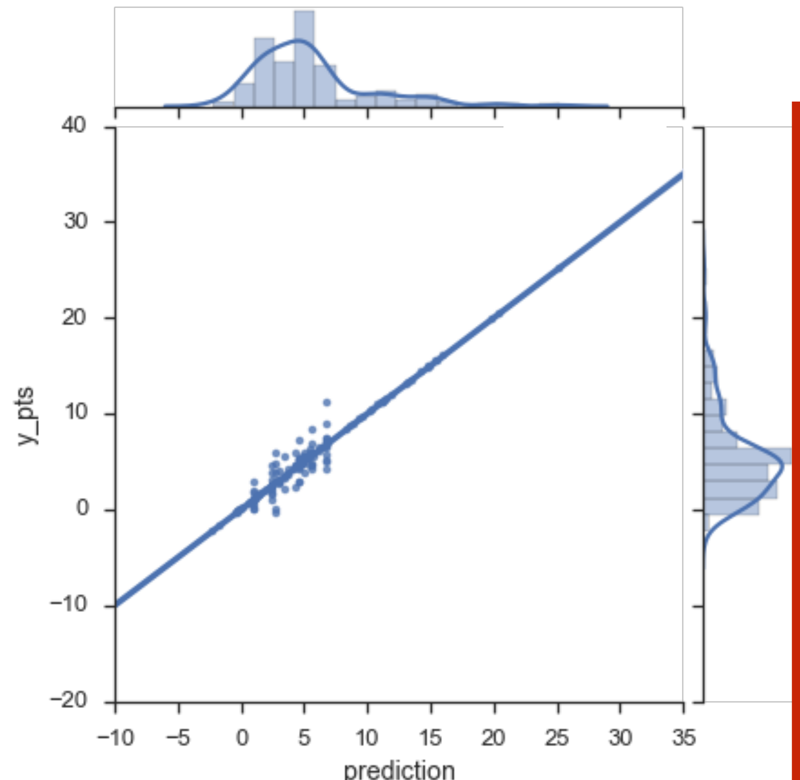
- Decision process
- Reward system
- Learn series of actions

Unsupervised Learning

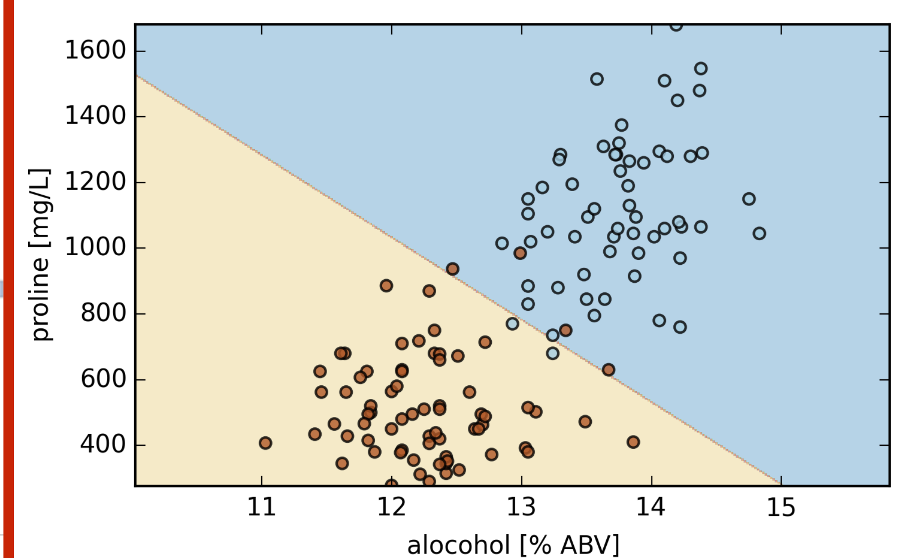
Supervised Learning



Clustering:
[DBSCAN on a toy dataset]



Regression:
[Soccer Fantasy Score prediction]



Classification:
[SVM on 2 classes of the Wine dataset]

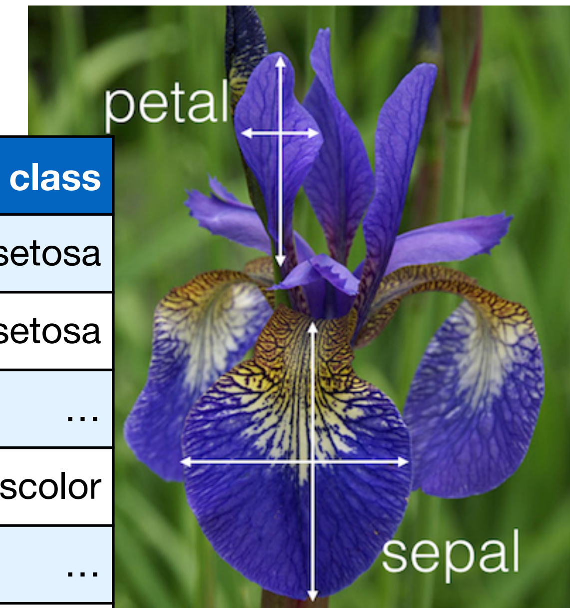
Today's topic

Nomenclature

IRIS

https://archive.ics.uci.edu/ml/datasets/Iris

Instances (samples, observations)



	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
...	...	...	...	...	...
50	6.4	3.2	4.5	1.5	veriscolor
...	...	...	...	...	...
150	5.9	3.0	5.1	1.8	virginica

Features (attributes, dimensions)

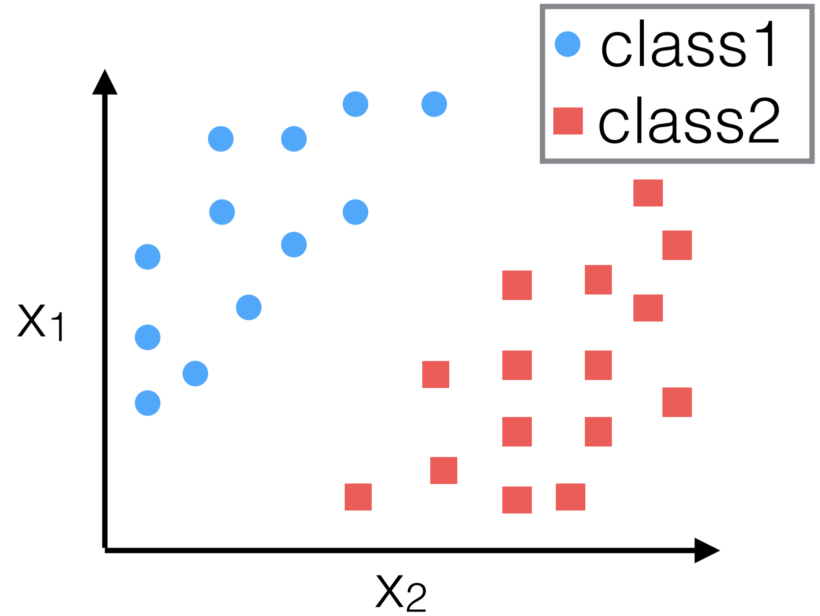
Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

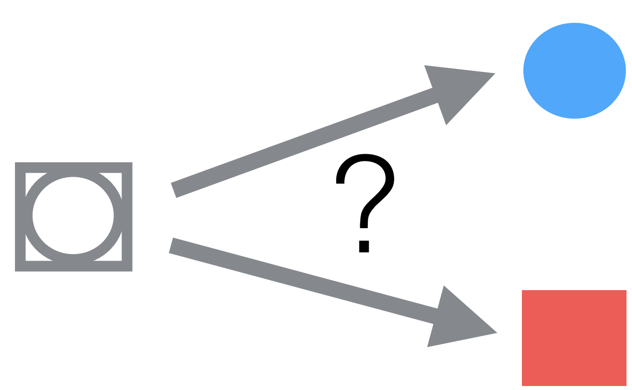
$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

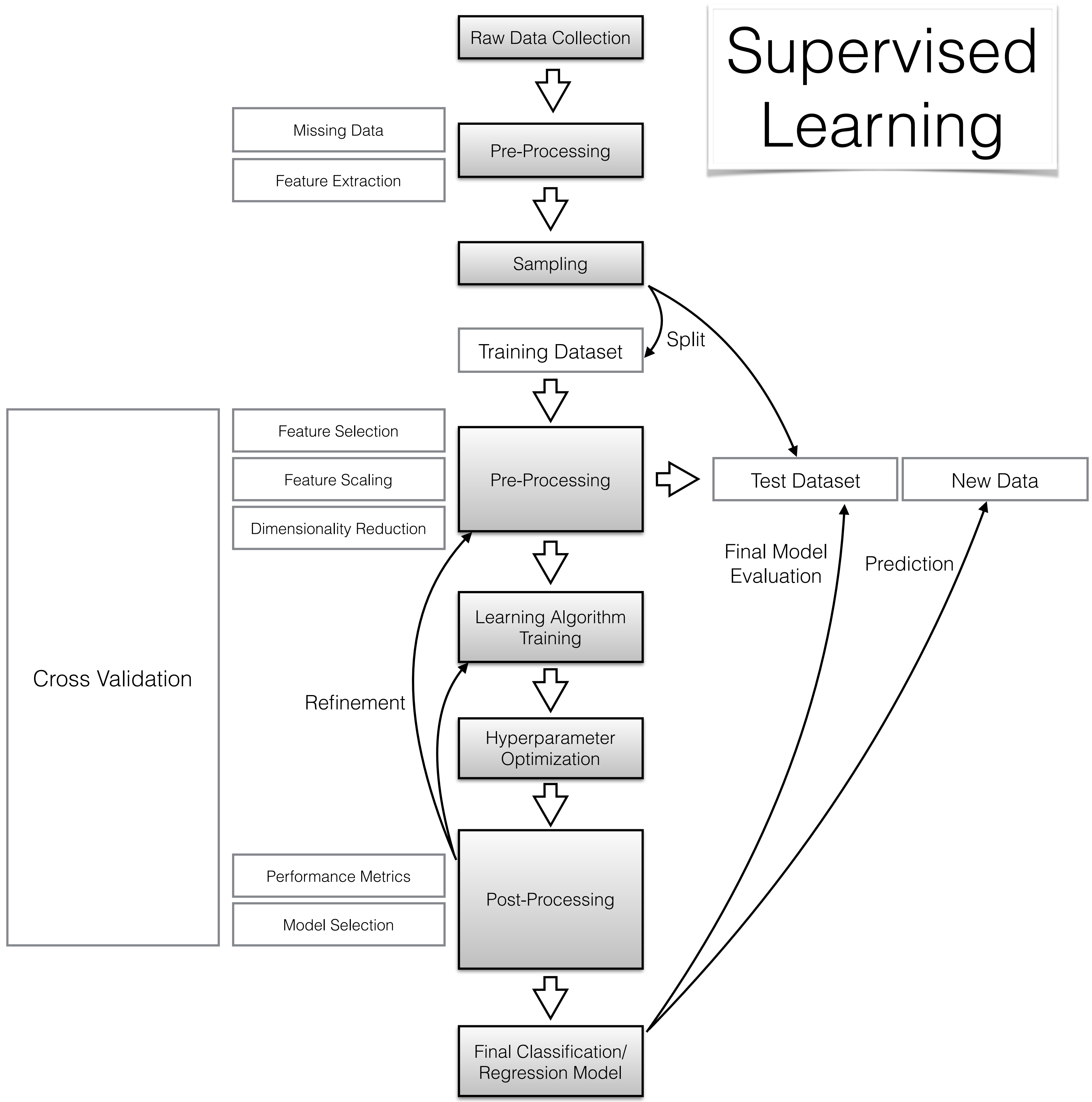
Classification

1) Learn from training data

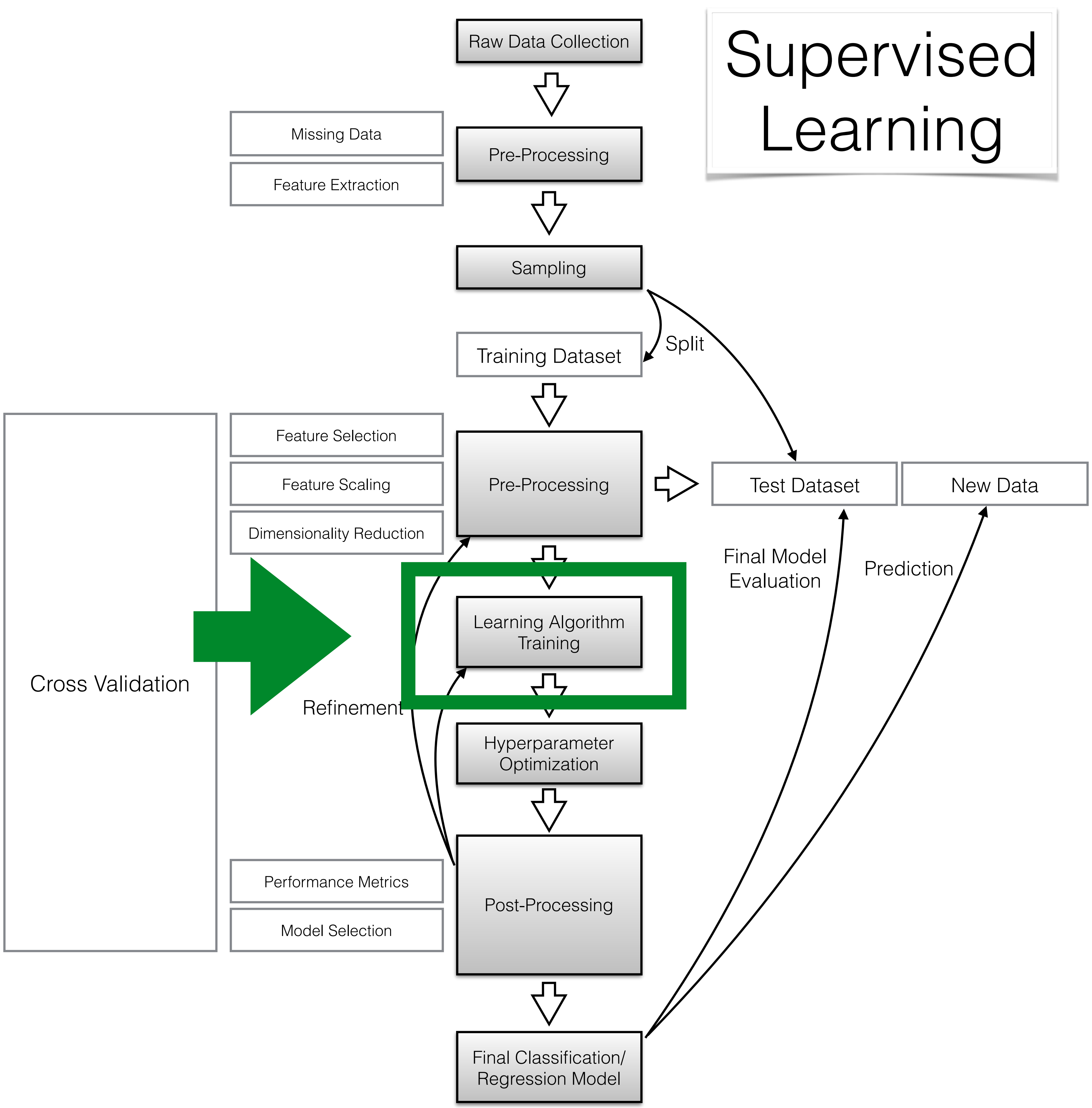


2) Map unseen (new) data





Supervised Learning
Raw Data Collection
Missing Data
Feature Extraction
Pre-Processing
Sampling
Split
Training Dataset
Cross Validation
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Test Dataset
New Data
Final Model Evaluation
Prediction
Learning Algorithm Training
Refinement
Hyperparameter Optimization
Performance Metrics
Model Selection
Post-Processing
Final Classification/ Regression Model



Supervised Learning
Raw Data Collection
Missing Data
Feature Extraction
Pre-Processing
Sampling
Split
Training Dataset
Cross Validation
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Test Dataset
New Data
Learning Algorithm Training
Refinement
Final Model Evaluation
Prediction
Hyperparameter Optimization
Performance Metrics
Model Selection
Post-Processing
Final Classification/ Regression Model

A Few Common Classifiers

Perceptron

Naive Bayes

Decision Tree

K-Nearest Neighbor

Logistic Regression

Artificial Neural Network / Deep Learning

Support Vector Machine

Ensemble Methods: Random Forest, Bagging, AdaBoost

Discriminative Algorithms

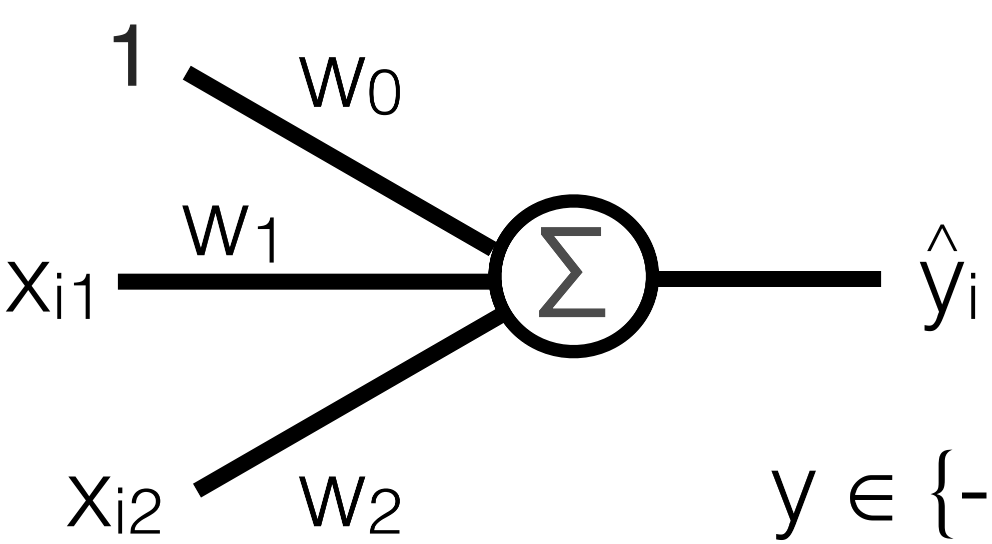
- Map $x \to y$ directly.
- E.g., **distinguish** between people speaking different languages **without learning the languages.**
- Logistic Regression, SVM, Neural Networks …

Generative Algorithms

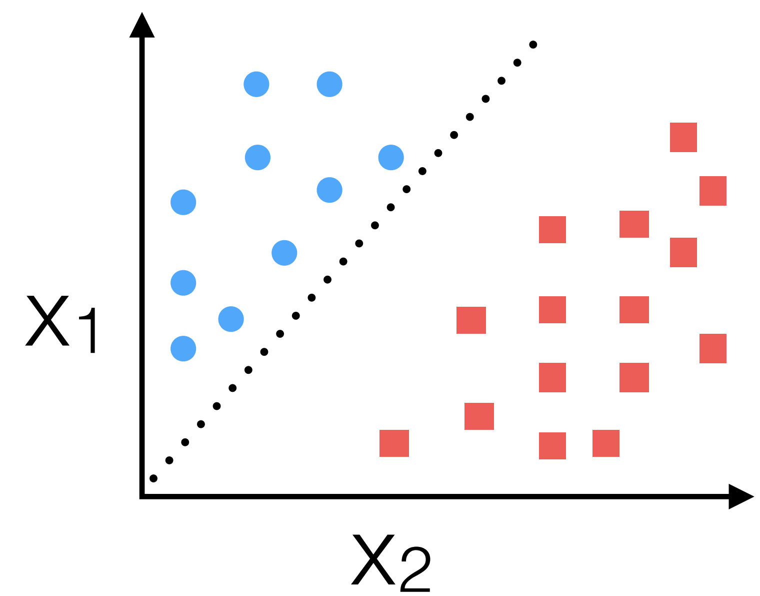
- Models a more general problem: how the data was generated.
- I.e., the distribution of the class; joint probability distribution p(x,y).
- Naive Bayes, Bayesian Belief Network classifier, Restricted Boltzmann Machine …

Examples of Discriminative Classifiers: Perceptron

F. Rosenblatt. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, **1957.**



$y \in \{-1, 1\}$



$$\hat{y} = \mathbf{w}^T \mathbf{x} = w_0 + w_1 x_1 + w_2 x_2$$

$$\hat{y}_i \begin{cases} 1 \text{ if } w^T x_i \geq \theta \\ -1 \text{ otherwise} \end{cases}$$

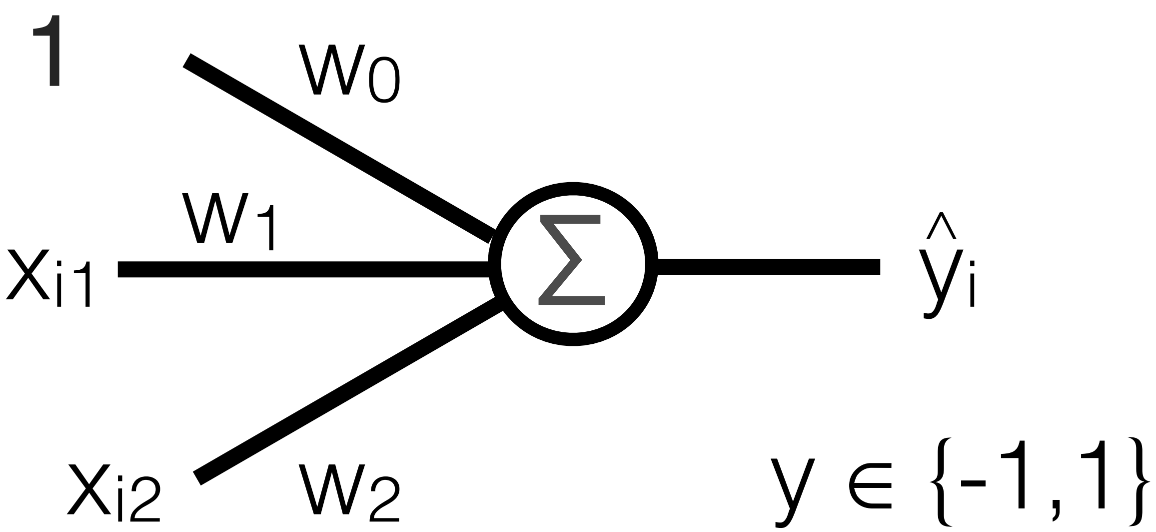
w_j = weight
x_i = training sample
y_i = desired output
$\hat{y}_i$ = actual output
t = iteration step
η = learning rate
θ = threshold (here 0)

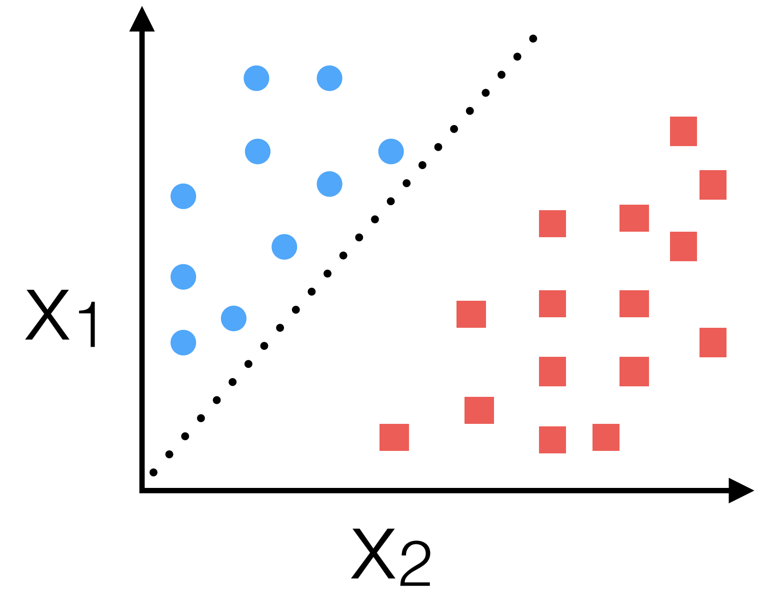
update rule:
$$w_j(t+1) = w_j(t) + \eta(y_i - \hat{y}_i)x_i$$

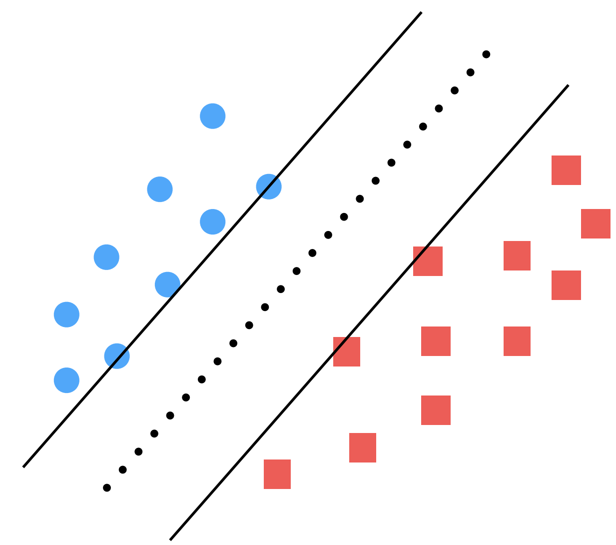
until
$t+1$ = max iter
or error = 0

Discriminative Classifiers: Perceptron

F. Rosenblatt. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory, 1957.







- Binary classifier (one vs all, OVA)
- Convergence problems (set *n* iterations)
- Modification: stochastic gradient descent
- “Modern” perceptron: Support Vector Machine (maximize margin)
- Multilayer perceptron (MLP)

Generative Classifiers: Naive Bayes

Bayes Theorem:

$$\boxed{P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j)\; P(\omega_j)}{P(\mathbf{x}_i)}}$$

$$\text{Posterior probability} = \frac{\text{Likelihood x Prior probability}}{\text{Evidence}}$$

Iris example: P("Setosa"| $\mathbf{x}_i$), $\mathbf{x}_i$ = [4.5 cm, 7.4 cm]

Generative Classifiers: Naive Bayes

Bayes Theorem: $$\boxed{P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j)\ P(\omega_j)}{P(\mathbf{x}_i)}}$$

Decision Rule: pred. class label $\omega_j \leftarrow \underset{i = 1, \ldots, m}{\operatorname{argmax}} P(\omega_j \mid \mathbf{x}_i)$

e.g., $j \in$ {Setosa, Versicolor, Virginica}

Generative Classifiers: Naive Bayes

Evidence: $P(\omega_j \mid \mathbf{x}_i) = \dfrac{P(\mathbf{x}_i \mid \omega_j)\; P(\omega_j)}{\cancel{P(\mathbf{x}_i)}}$ (cancels out)

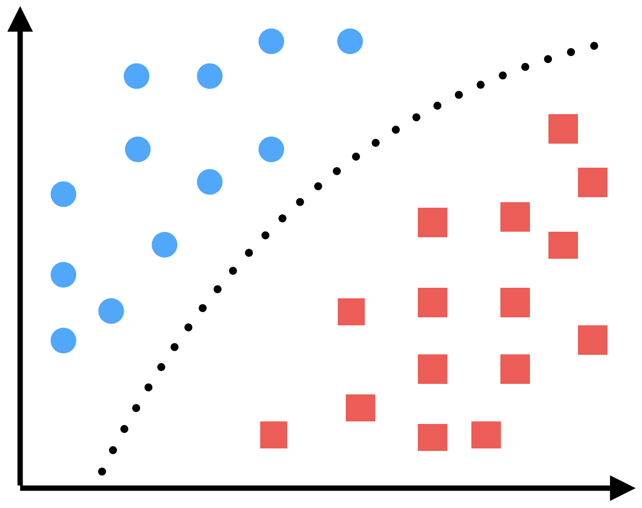
Prior probability: $P(\omega_j) = \dfrac{N_{\omega j}}{N_c}$ (class frequency)

Class-conditional probability
(here Gaussian kernel):

$$P(x_{ik} \mid \omega_j) = \frac{1}{\sqrt{(2 \pi \sigma_{\omega j}^2)}} \exp\left(- \frac{(x_{ik} - \mu_{\omega j})^2}{2\sigma_{\omega j}^2}\right)$$

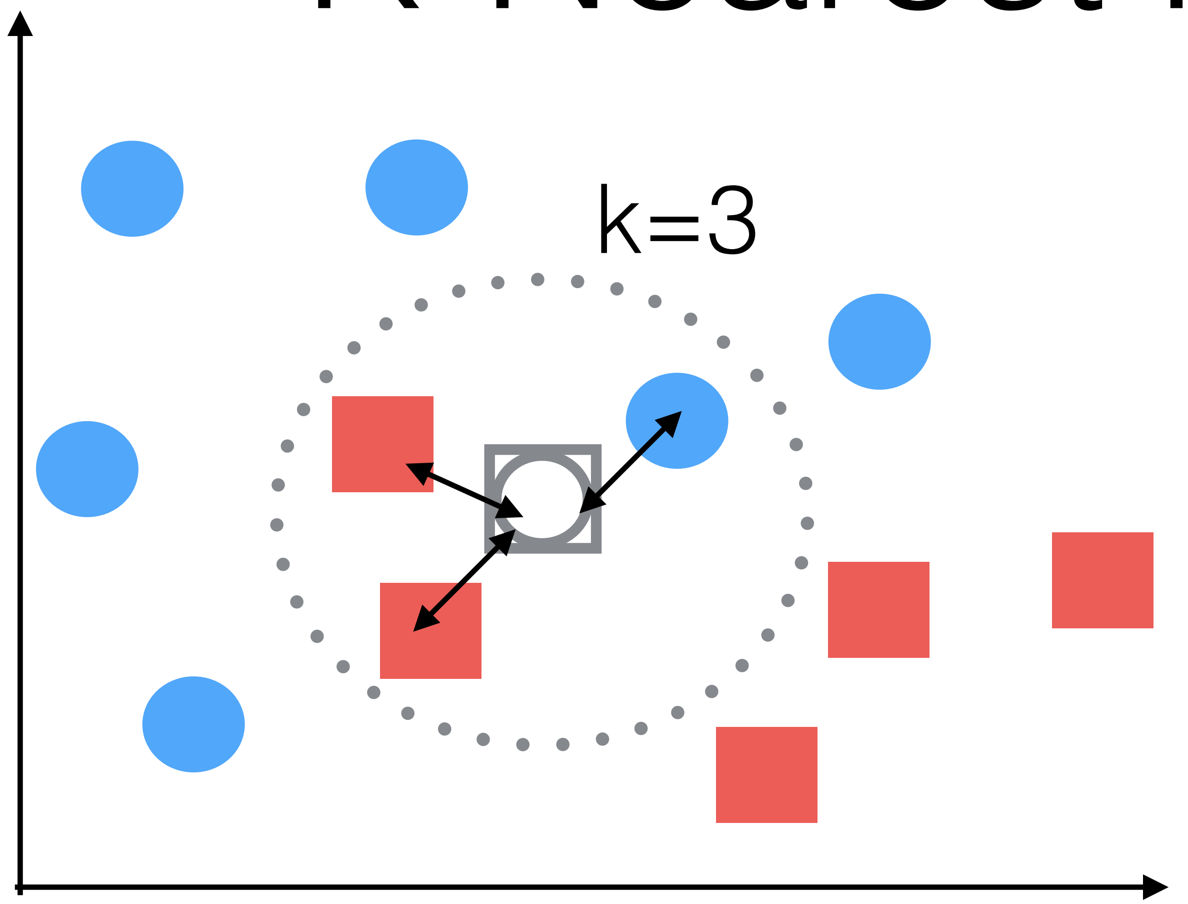
$$P(\mathbf{x}_i \mid \omega_j) = \prod_{k=1}^{d} P(x_{ik} \mid \omega_j)$$

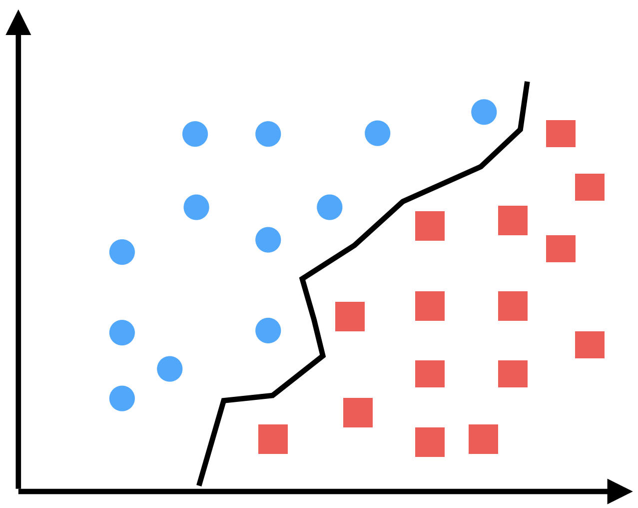
Generative Classifiers: Naive Bayes



- Naive conditional independence assumption typically violated
- Works well for small datasets
- Multinomial model still quite popular for text classification (e.g., spam filter)

Non-Parametric Classifiers: K-Nearest Neighbor

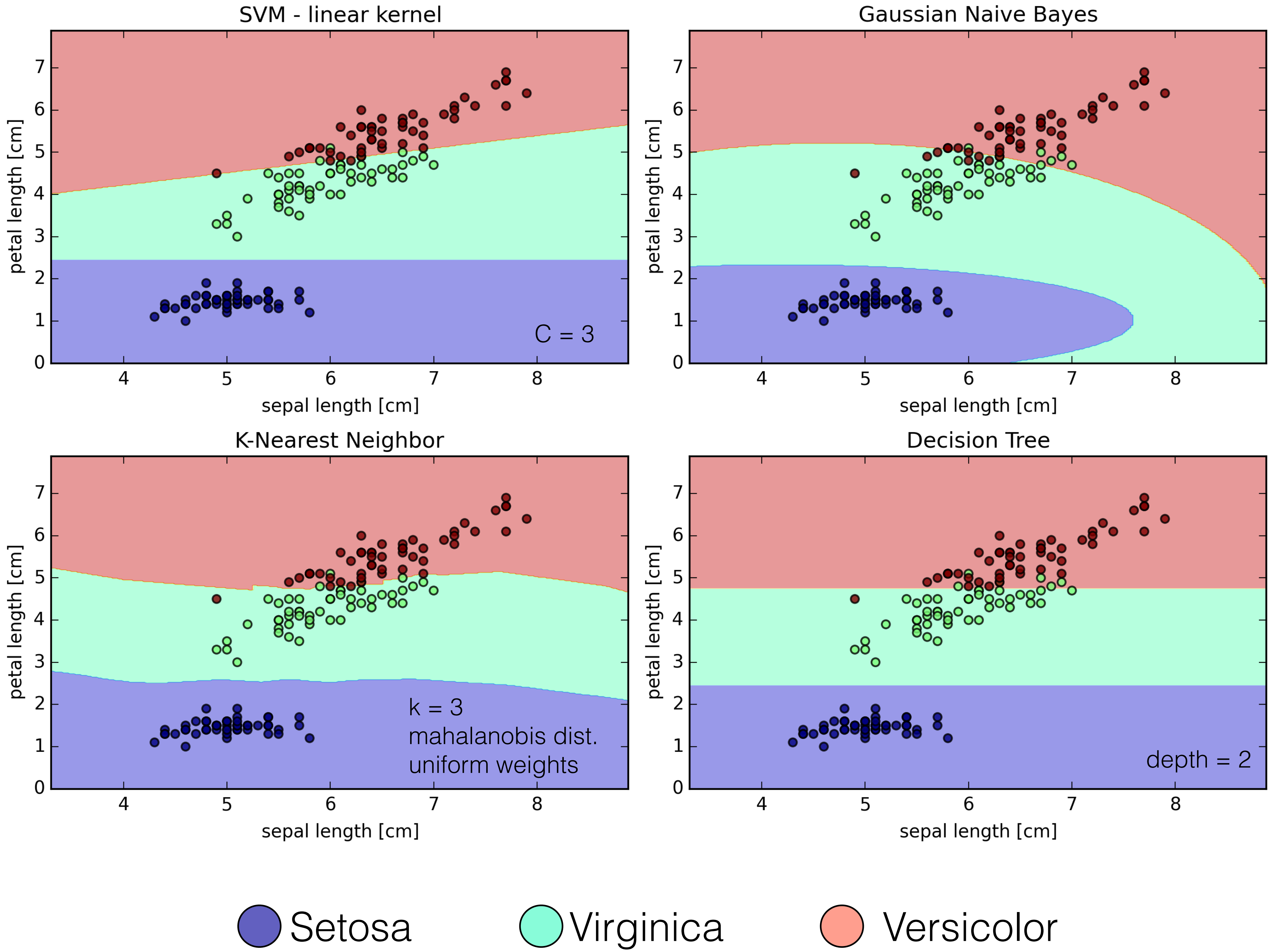




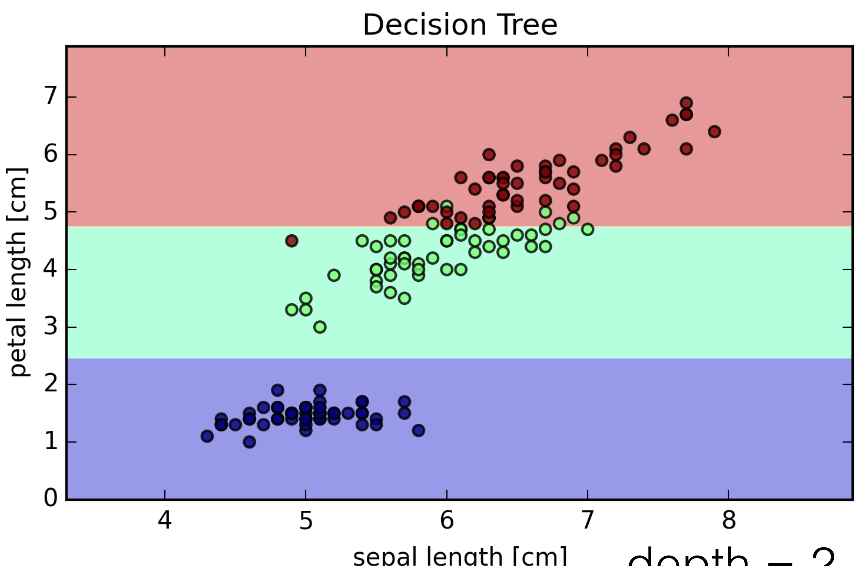
e.g., k=1

- Simple!
- Lazy learner
- Very susceptible to curse of dimensionality

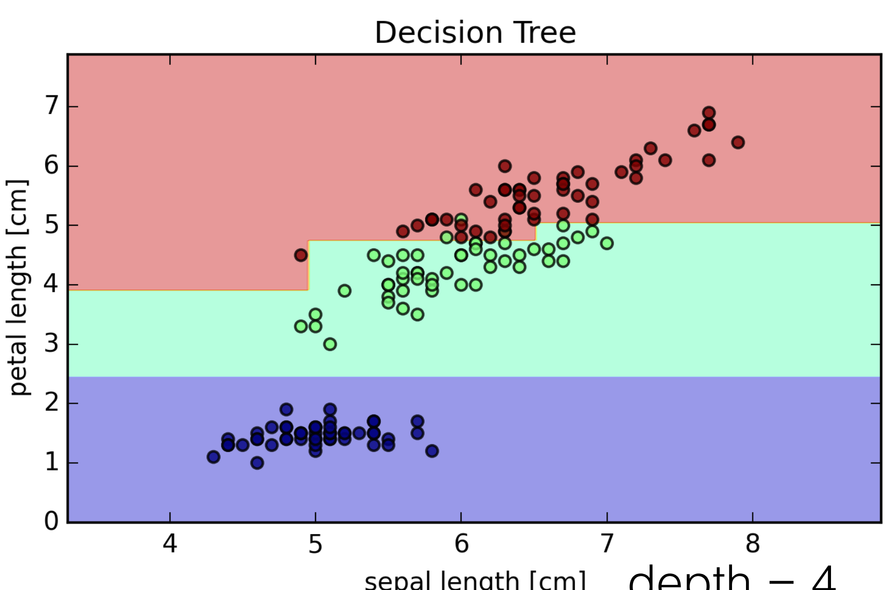
Iris Example



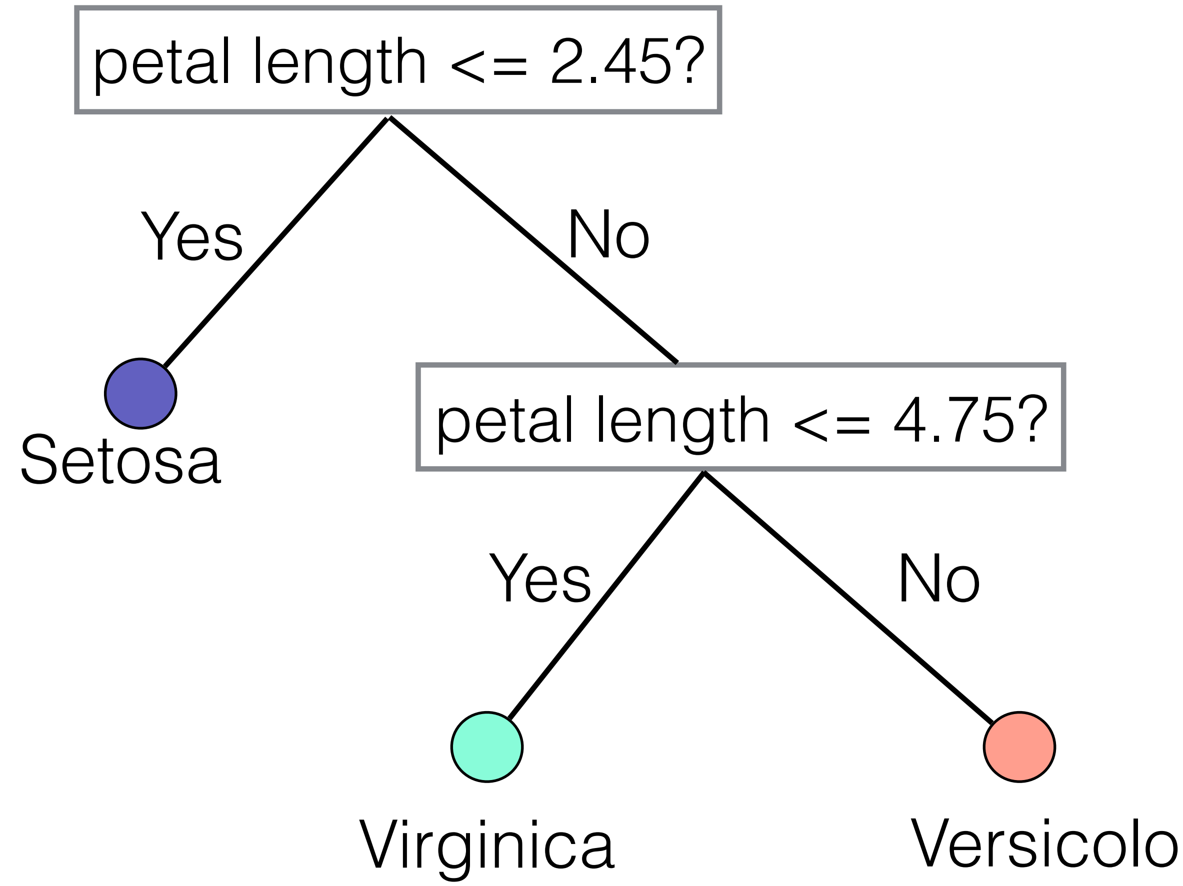
Decision Tree



depth = 2



depth = 4



$$\text{Entropy} = \sum_i -p_i \log_k p_i$$

e.g., $2\,(-\,0.5 \log_2(0.5)) = 1$

Information Gain =
entropy(parent) – [avg entropy(children)]

"No Free Lunch" :(

D. H. Wolpert. The supervised learning no-free-lunch theorems. In Soft Computing and Industry, pages 25–42. Springer, 2002.

Our model is a simplification of reality

⇩

Simplification is based on assumptions (model bias)

⇩

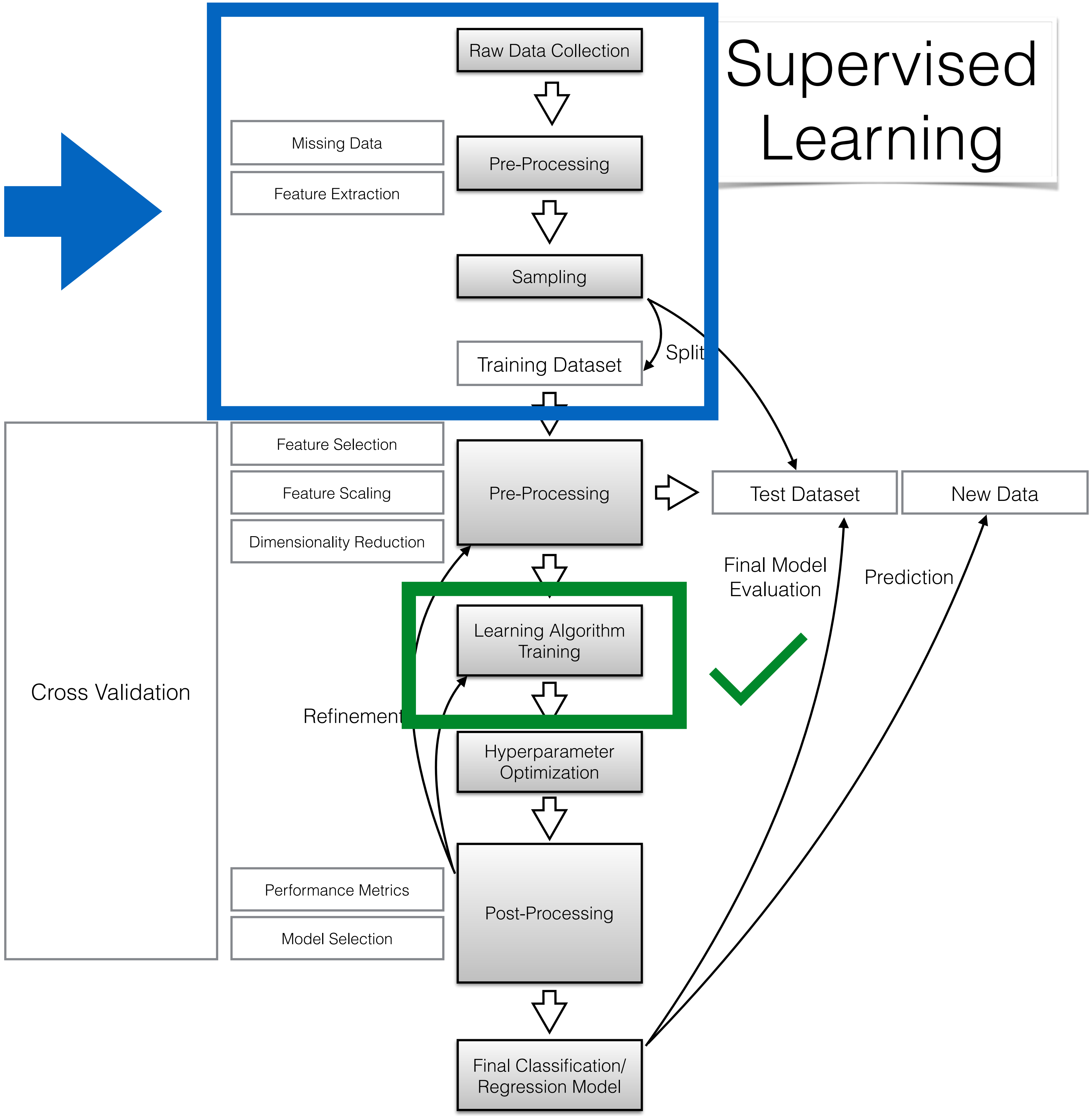
Assumptions fail in certain situations

Roughly speaking:

"No one model works best for all possible situations."

Which Algorithm?

- What is the size and dimensionality of my training set?
- Is the data linearly separable?
- How much do I care about computational efficiency?
  - Model building vs. real-time prediction time
  - Eager vs. lazy learning / on-line vs. batch learning
  - prediction performance vs. speed
- Do I care about interpretability or should it "just work well?"
- ...


Supervised Learning
Raw Data Collection
Missing Data
Feature Extraction
Pre-Processing
Sampling
Split
Training Dataset
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Test Dataset
New Data
Cross Validation
Final Model Evaluation
Prediction
Learning Algorithm Training
Refinement
Hyperparameter Optimization
Performance Metrics
Model Selection
Post-Processing
Final Classification/ Regression Model

Missing Values:

- Remove features (columns)
- Remove samples (rows)
- Imputation (mean, nearest neighbor, …)

Feature Scaling:

e.g., *standardization:*

$$z = \frac{x_{ik} - \mu_k}{\sigma_k}$$

(use same parameters for the test/new data!)

- Faster convergence (gradient descent)
- Distances on same scale (k-NN with Euclidean distance)
- Mean centering for free
- Normal distributed data
- Numerical stability by avoiding small weights

Sampling:

- Random split into training and validation sets
- Typically 60/40, 70/30, 80/20
- Don't use validation set until the very end! (overfitting)

Categorical Variables

	color	size	prize	class
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

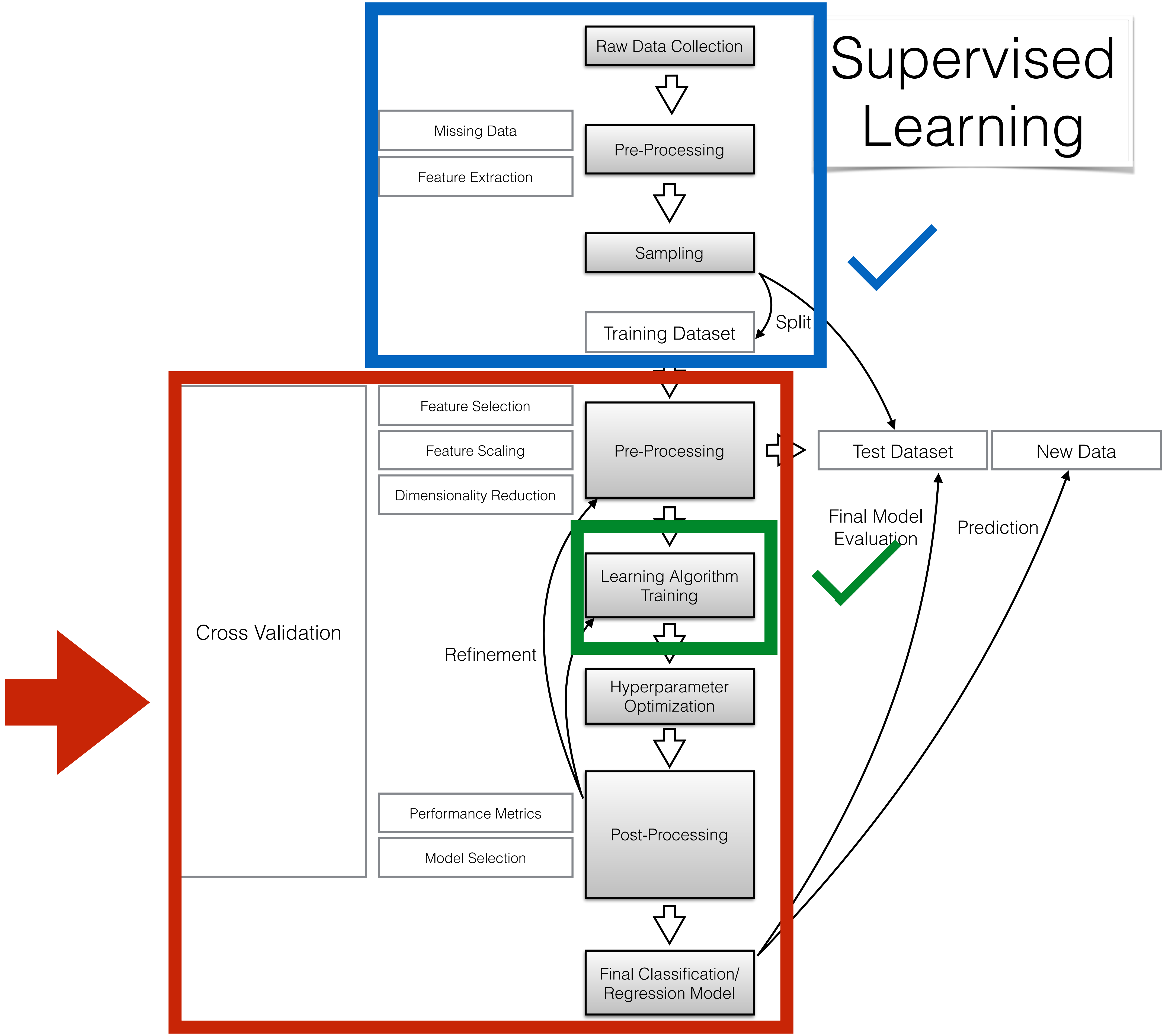
nominal

green → (1,0,0)
red → (0,1,0)
blue → (0,0,1)

ordinal

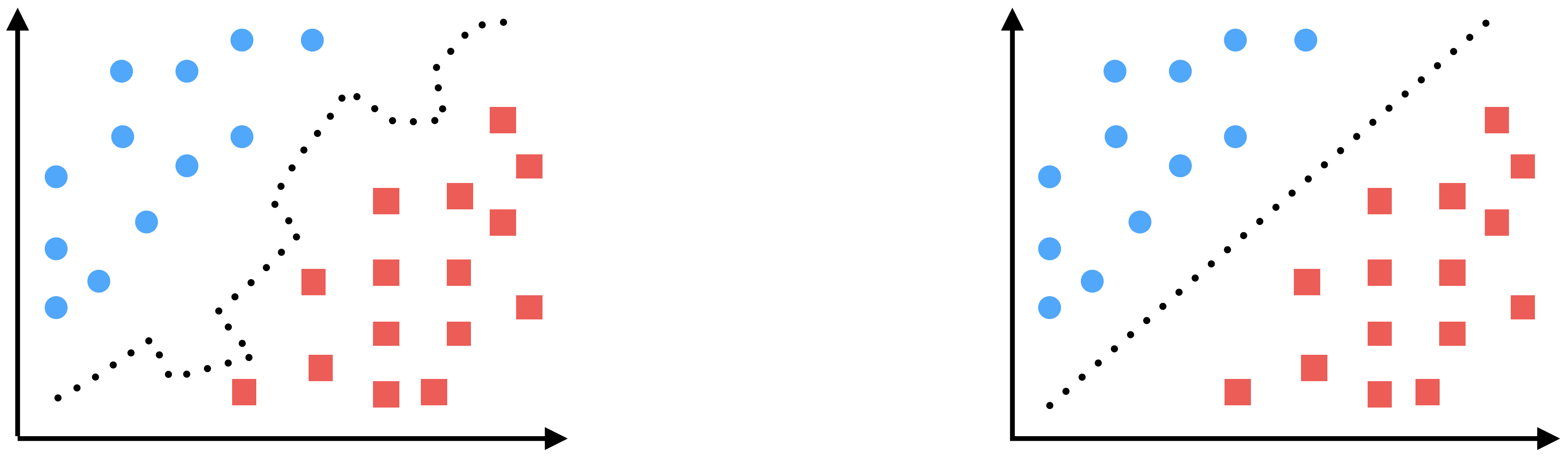
M → 1
L → 2
XL → 3

	class	color=blue	color=green	color=red	prize	size
0	0	0	1	0	10.1	1
1	1	0	0	1	13.5	2
2	0	1	0	0	15.3	3



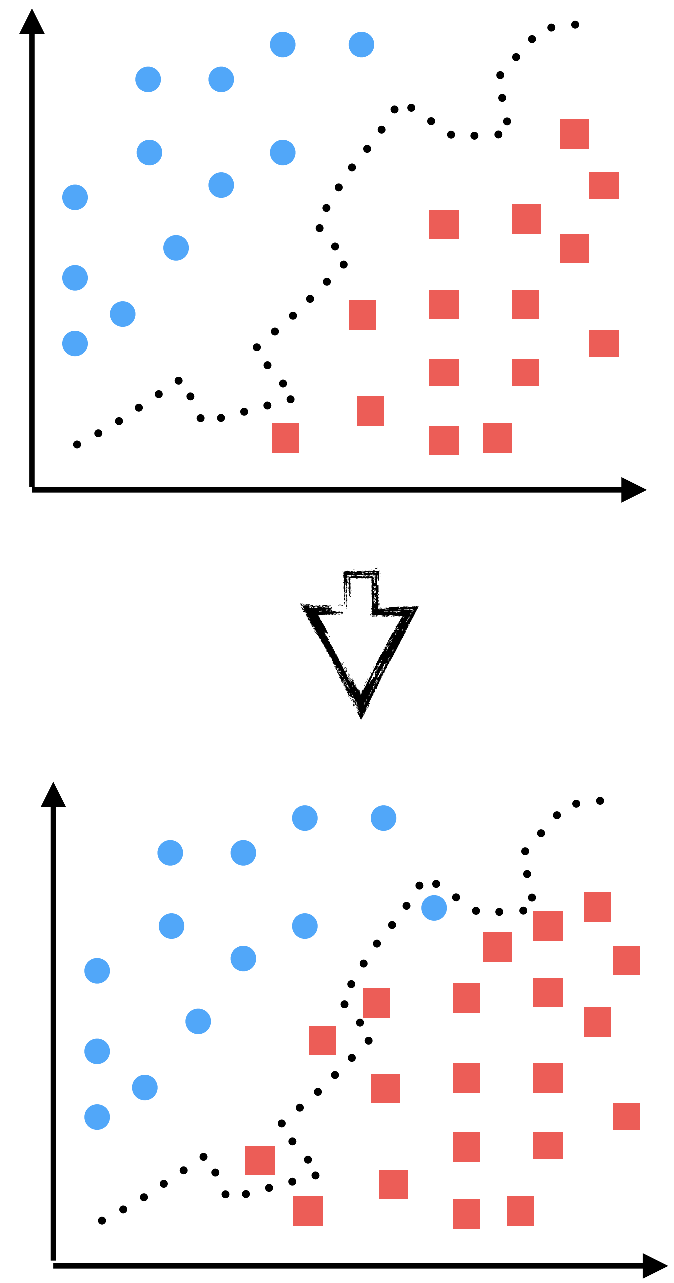
Sebastian Raschka 2014

Generalization Error and Overfitting



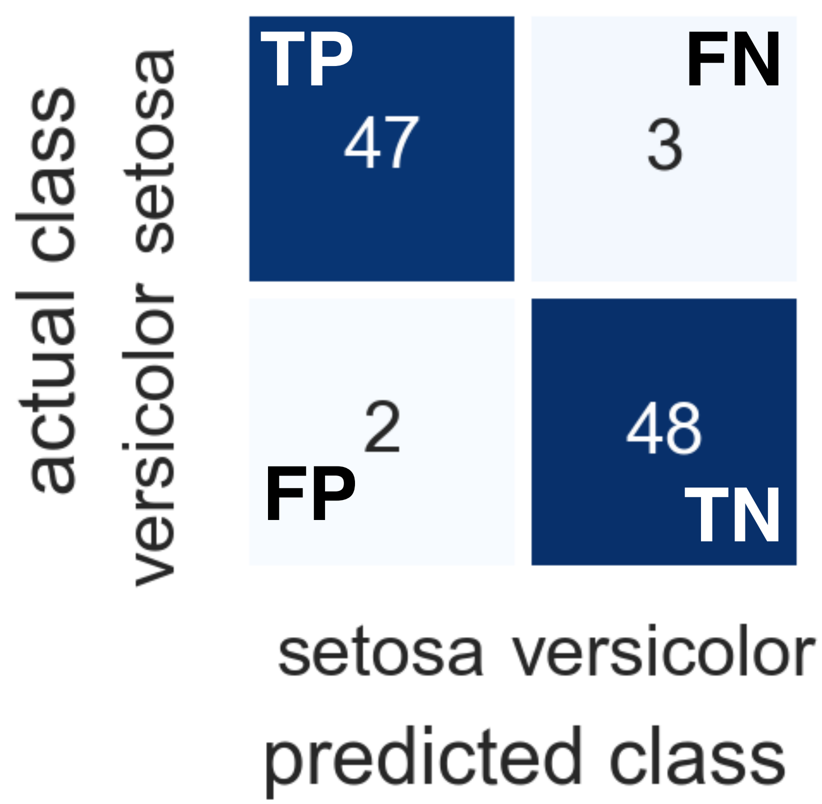
How well does the model perform on unseen data?

Generalization Error and Overfitting



Error Metrics: Confusion Matrix

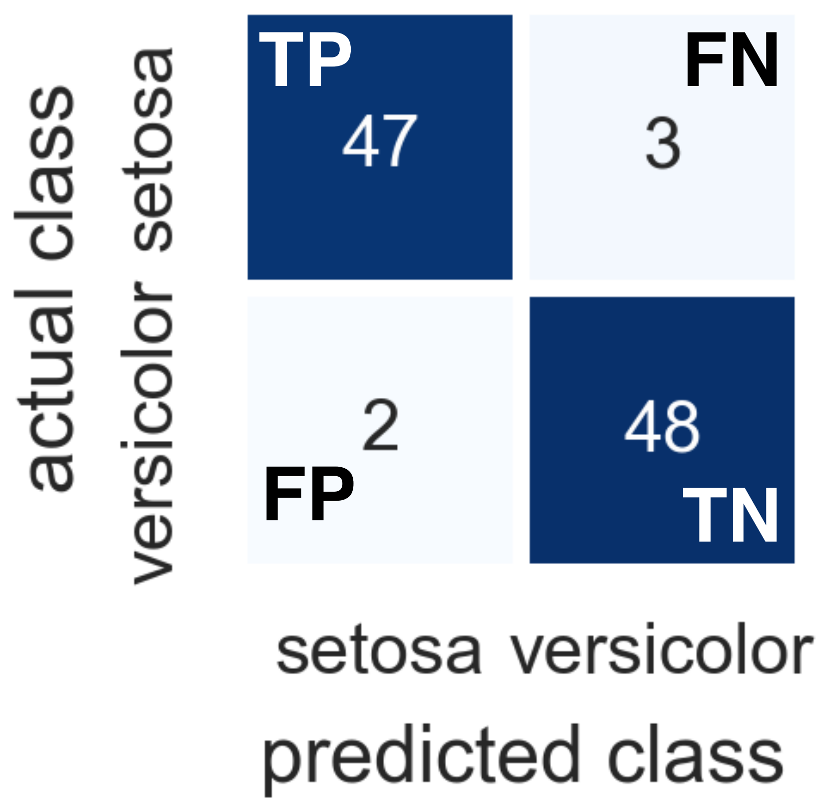
here: “setosa” = “positive”



[Linear SVM on sepal/petal lengths]

Error Metrics

here: “setosa” = “positive”



[Linear SVM on sepal/petal lengths]

“micro” and “macro” averaging for multi-class

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

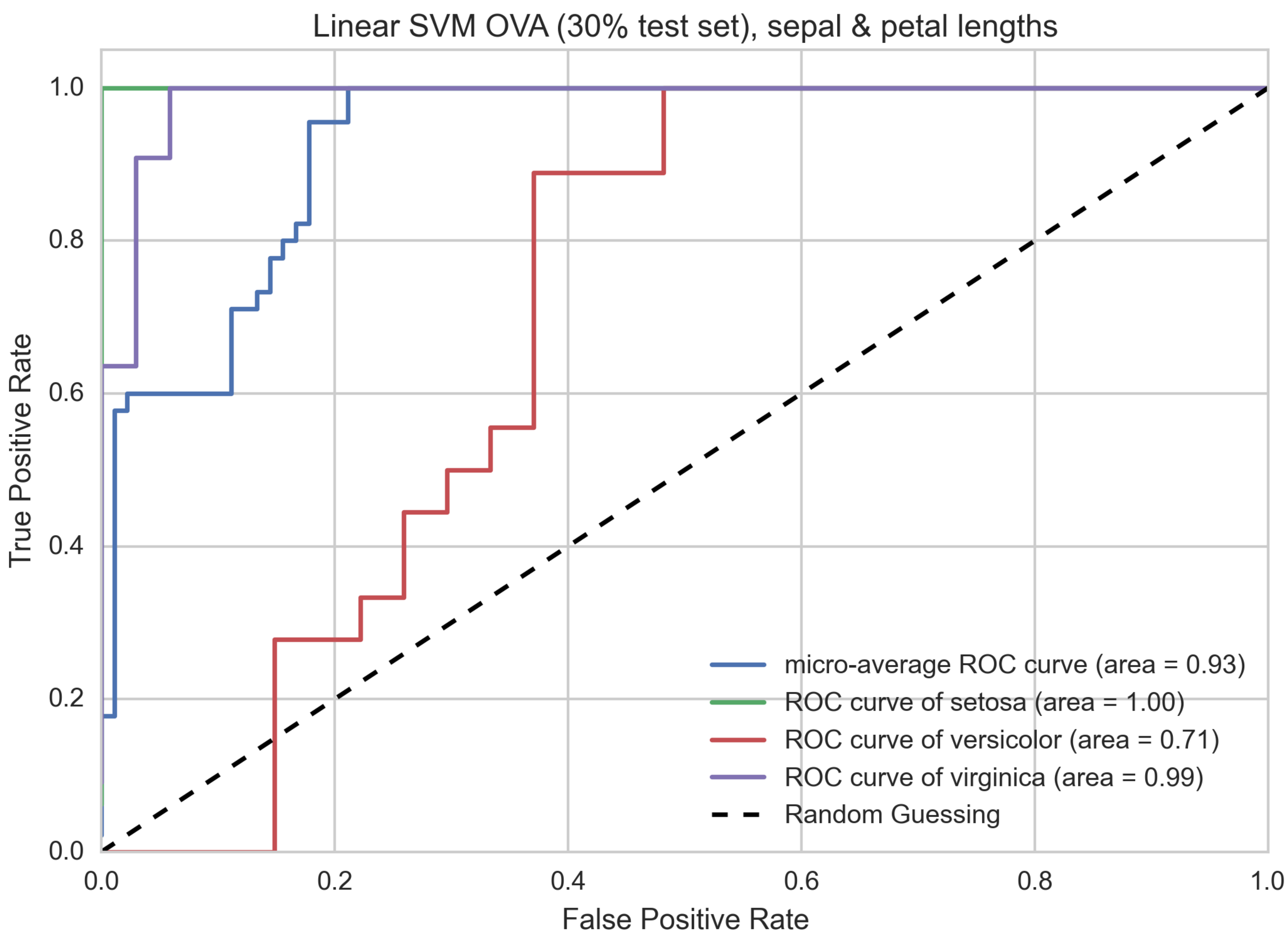
$$= 1 - \mathit{Error}$$

$$\mathit{False\ Positive\ Rate} = \frac{FP}{N}$$

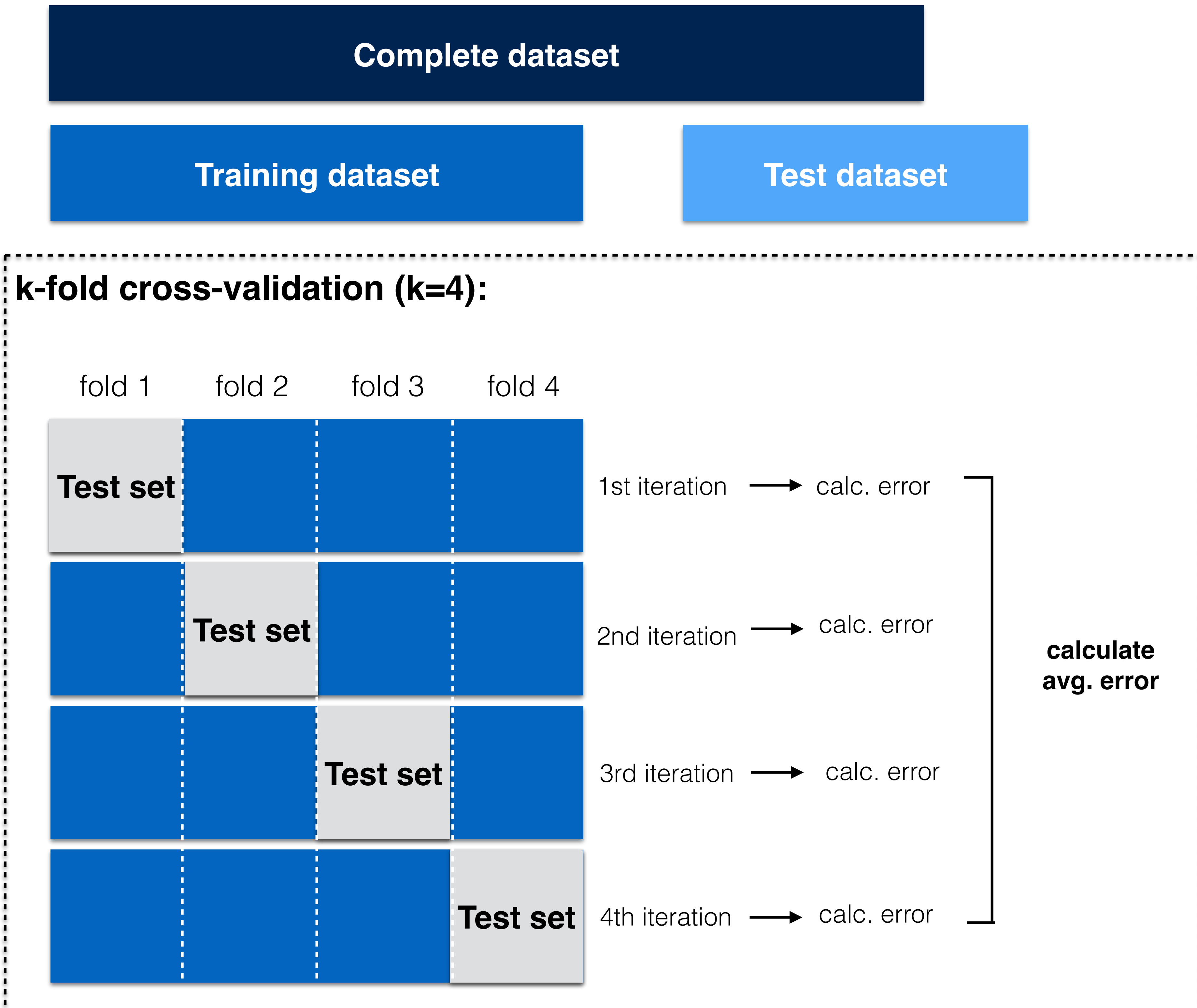
$$\underset{(\mathit{Recall})}{\mathit{True\ Positive\ Rate}} = \frac{TP}{P}$$

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

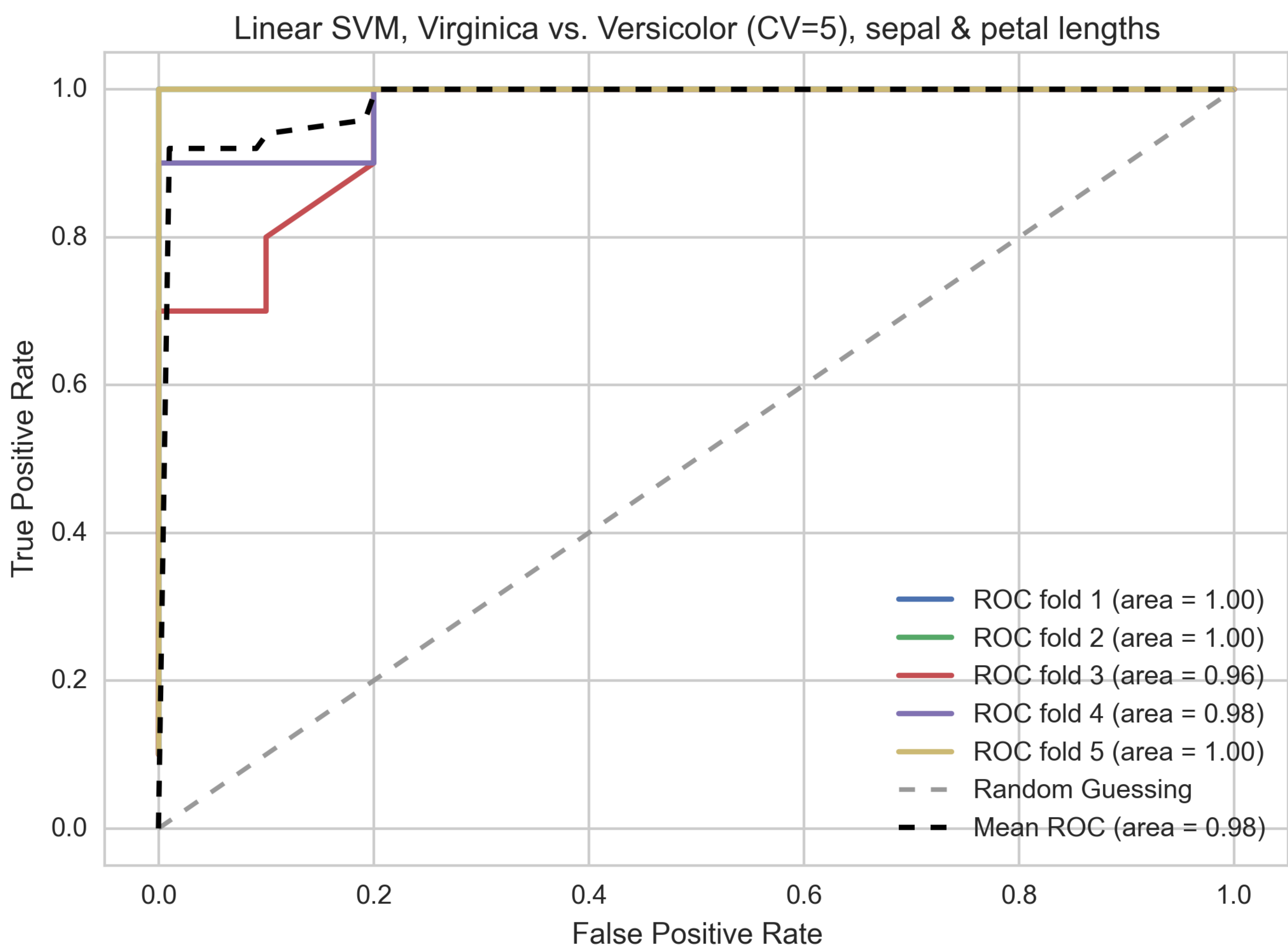
Receiver Operating Characteristic (ROC) Curves



Model Selection



k-fold CV and ROC



Feature Selection

IMPORTANT!
(Noise, overfitting, curse of dimensionality, efficiency)

- Domain knowledge
- Variance threshold
- Exhaustive search
- Decision trees
- …

Simplest example:
Greedy Backward Selection

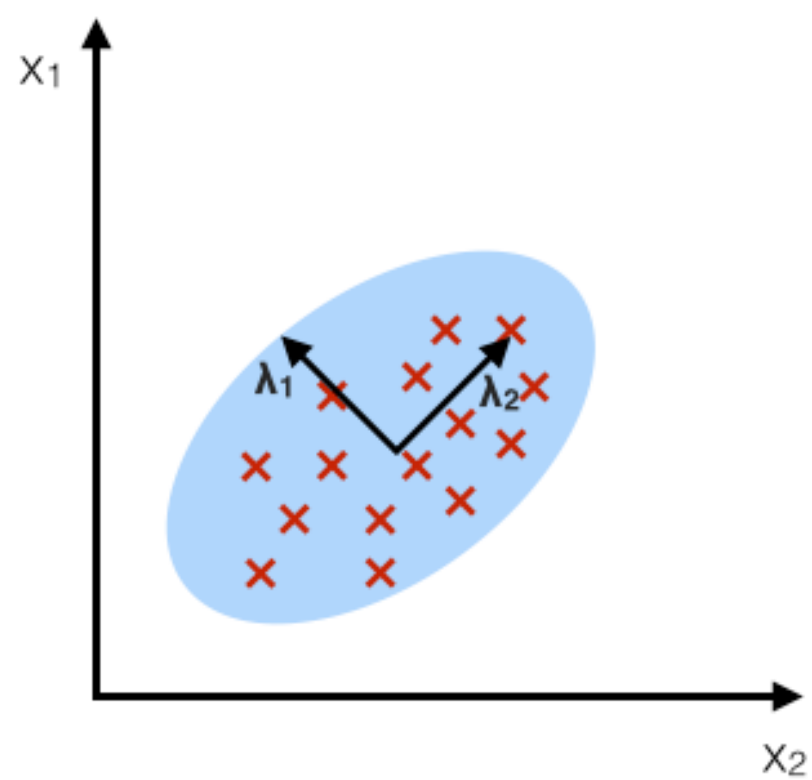
start: $\mathbf{X} = [\mathbf{x}_1, \cancel{\mathbf{x}_2}, \mathbf{x}_3, \mathbf{x}_4]$

$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_3, \cancel{\mathbf{x}_4}]$

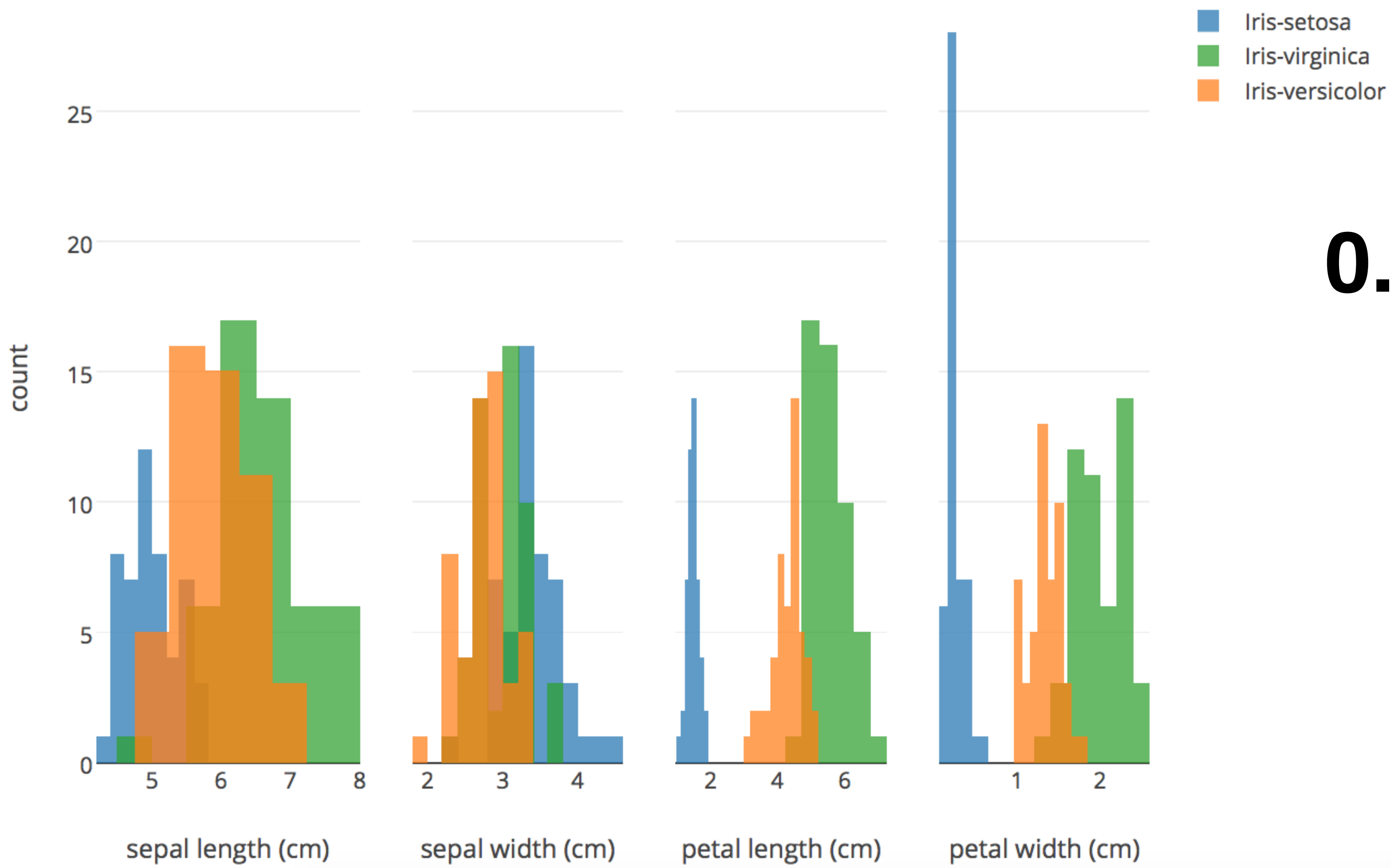
stop: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_3]$
(if $d = k$)

Dimensionality Reduction

- Transformation onto a new feature subspace
- e.g., Principal Component Analysis (PCA)
- Find directions of maximum variance
- Retain most of the information



PCA in 3 Steps



0. Standardize data

$$z = \frac{x_{ik} - \mu_k}{\sigma_k}$$

1. Compute covariance matrix

$$\sigma_{ik} = \frac{1}{n-1} \sum_i (x_{ij} - \mu_j)(x_{ik} - \mu_k)$$

$$\Sigma = \begin{bmatrix} \sigma^2{}_1 & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma^2{}_2 & \sigma_{23} & \sigma_{24} \\ \sigma_{31} & \sigma_{32} & \sigma^2{}_3 & \sigma_{34} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma^2{}_4 \end{bmatrix}$$

PCA in 3 Steps

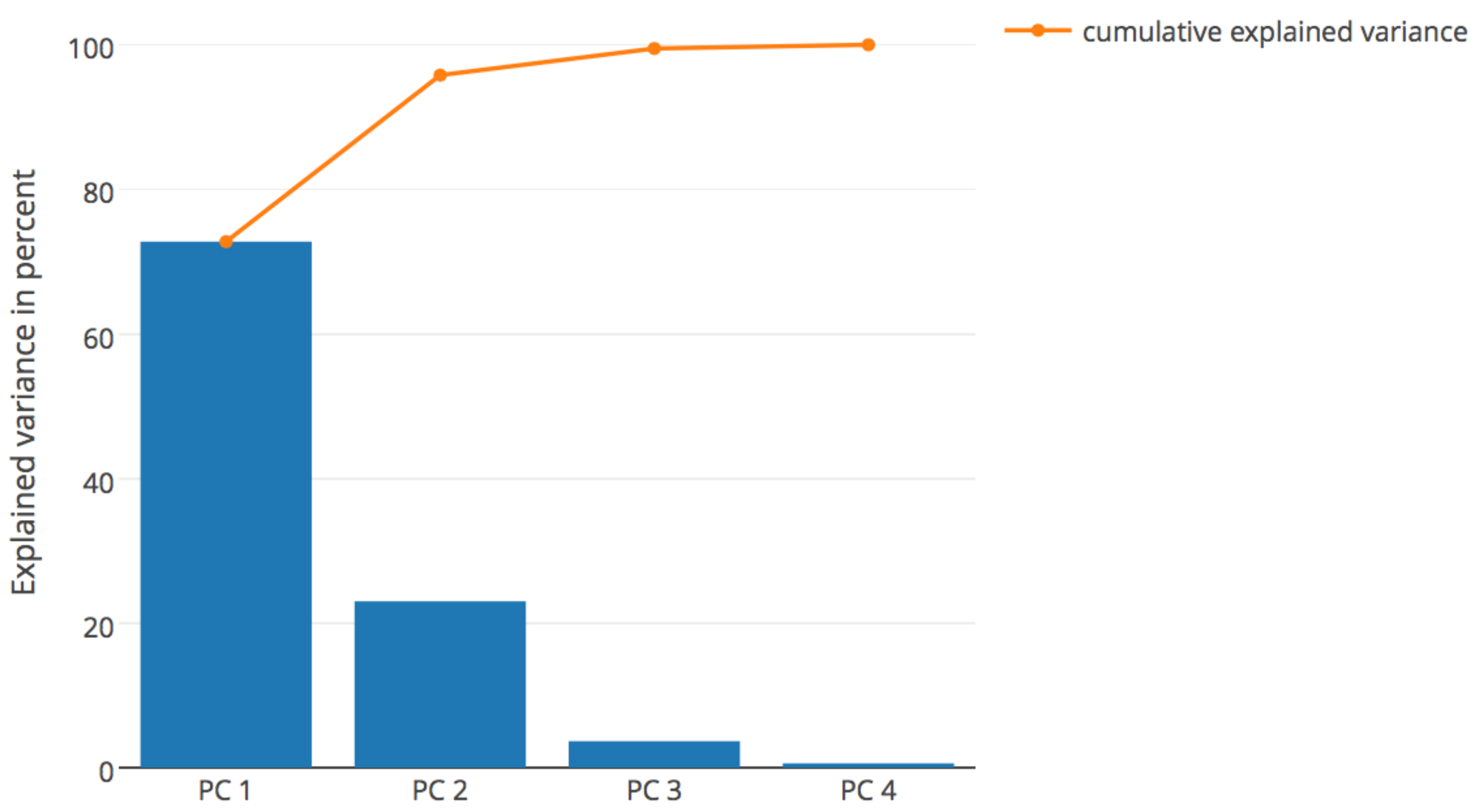
2. Eigendecomposition and sorting eigenvalues

$$\mathbf{X}\,\mathbf{v} = \lambda\,\mathbf{v}$$

```
Eigenvectors
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues
[ 2.93035378  0.92740362  0.14834223  0.02074601]
```

(from high to low)

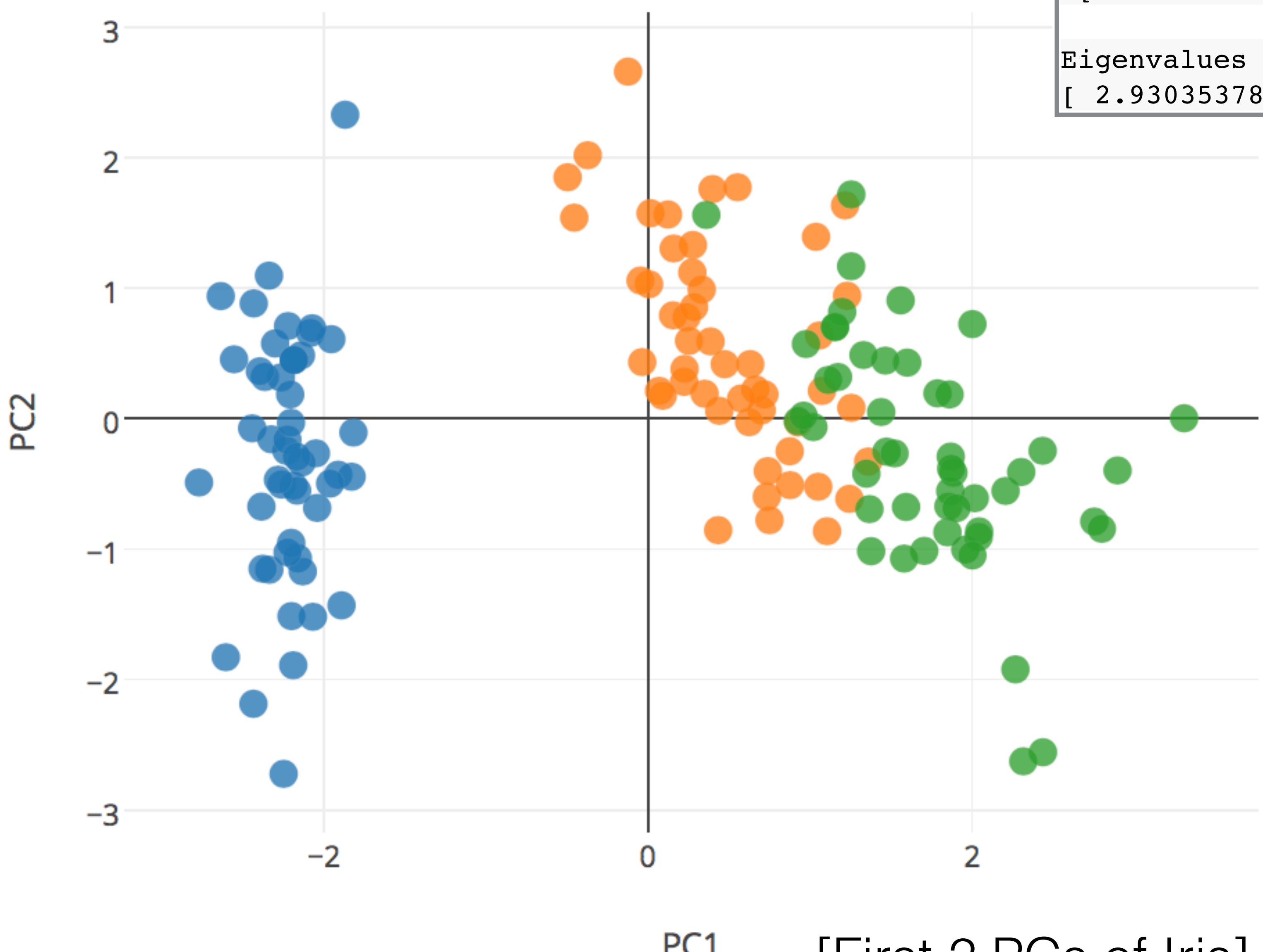


PCA in 3 Steps

3. Select top *k* eigenvectors and transform data

```
Eigenvectors
[[ 0.52237162 -0.37231836 -0.72101681  0.26199559]
 [-0.26335492 -0.92555649  0.24203288 -0.12413481]
 [ 0.58125401 -0.02109478  0.14089226 -0.80115427]
 [ 0.56561105 -0.06541577  0.6338014   0.52354627]]

Eigenvalues
[ 2.93035378  0.92740362  0.14834223  0.02074601]
```



[First 2 PCs of Iris]

Hyperparameter Optimization: GridSearch in scikit-learn

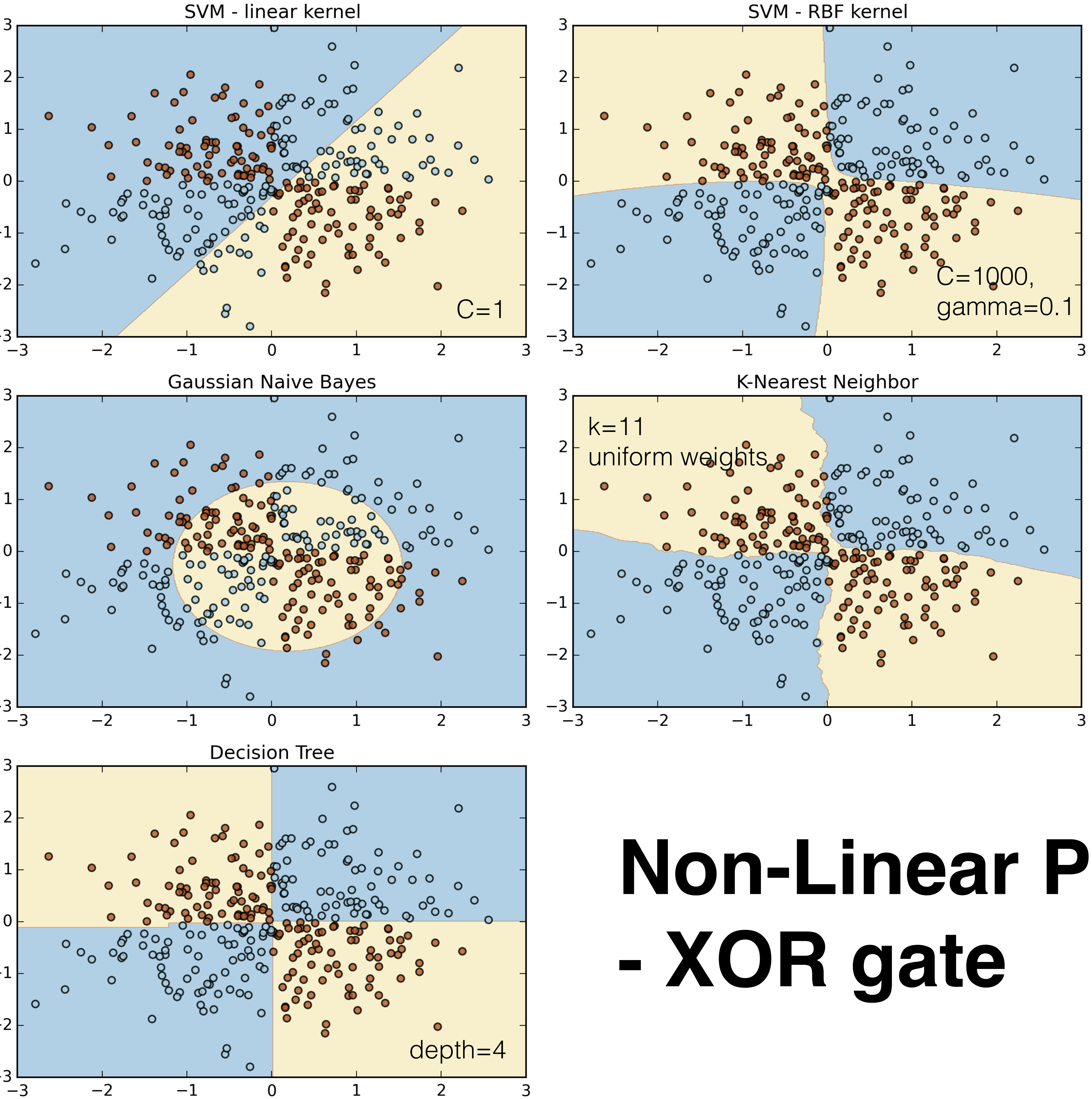
```
df = pd.read_csv('../data/iris_data.csv')
le = LabelEncoder()
X = df.iloc[:, 0:4]
y = le.fit_transform(df['class'].values)

pipeline = Pipeline([('scl', StandardScaler()),
                     ('sel', SelectKBest()),
                     ('clf', SVC(random_state=1))])

param_grid = [{'sel__k': [1, 2, 3, 4],
               'clf__C': [0.1, 1, 10, 100],
               'clf__kernel': ['linear']},
              {'sel__k': [1, 2, 3, 4],
               'clf__C': [0.1, 1, 10, 100],
               'clf__gamma':[0.0001, 0.001, 0.01, 0.1],
               'clf__kernel': ['rbf']}]

grid_search = GridSearchCV(pipeline,
                           param_grid=param_grid,
                           verbose=1,
                           cv=10,
                           scoring='accuracy',
                           n_jobs=2)
grid_search.fit(X, y)
print(grid_search.best_estimator_)
print(grid_search.best_score_)
```

```
Fitting 10 folds for each of 80 candidates, totalling 800 fits
[Parallel(n_jobs=2)]: Done   1 jobs       | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done  50 jobs       | elapsed:    0.3s
[Parallel(n_jobs=2)]: Done 200 jobs       | elapsed:    1.1s
[Parallel(n_jobs=2)]: Done 450 jobs       | elapsed:    2.6s
[Parallel(n_jobs=2)]: Done 800 out of 800 | elapsed:    4.6s finished
Pipeline(steps=[('scl', StandardScaler(copy=True, with_mean=True, with_std=True)), ('sel', SelectKBe
st(k=4, score_func=<function f_classif at 0x105e23bf8>)), ('clf', SVC(C=1, cache_size=200, class_wei
ght=None, coef0=0.0, degree=3, gamma=0.1,
  kernel='rbf', max_iter=-1, probability=False, random_state=1,
  shrinking=True, tol=0.001, verbose=False))])
0.98
```



Non-Linear Problems - XOR gate

Kernel Trick

Kernel function $\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$,

Kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$

Map onto high-dimensional space (non-linear combinations)

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \qquad \boldsymbol{x} \in I\!R^d$$

$$\Downarrow \phi$$

$$\boldsymbol{x}' = \begin{bmatrix} x_1 & x_2 & x_1x_2 & x_1^2 & x_1x_2^3 & \ldots \end{bmatrix} \qquad \boldsymbol{x} \in I\!R^k \quad (k >> d)$$

Kernel Trick

Trick: No explicit dot product!

Radius Basis Function (RBF) Kernel:

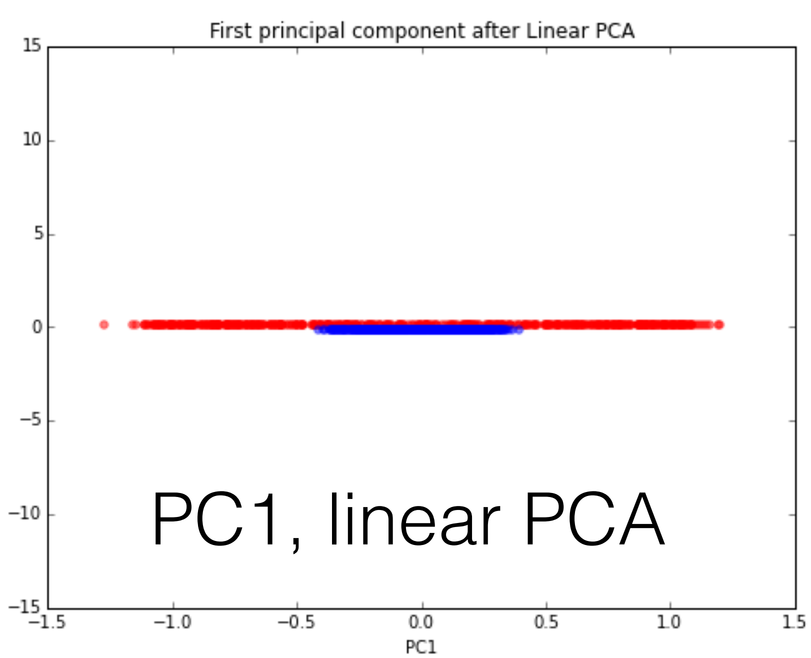
$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(- \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2} \right)$$

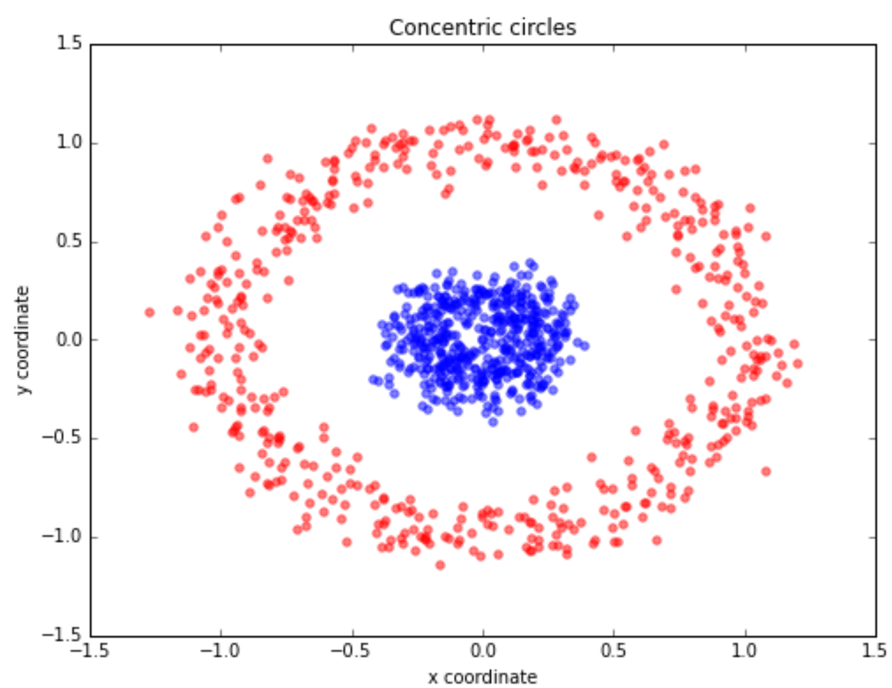
$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(- \gamma \; \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \right)$$

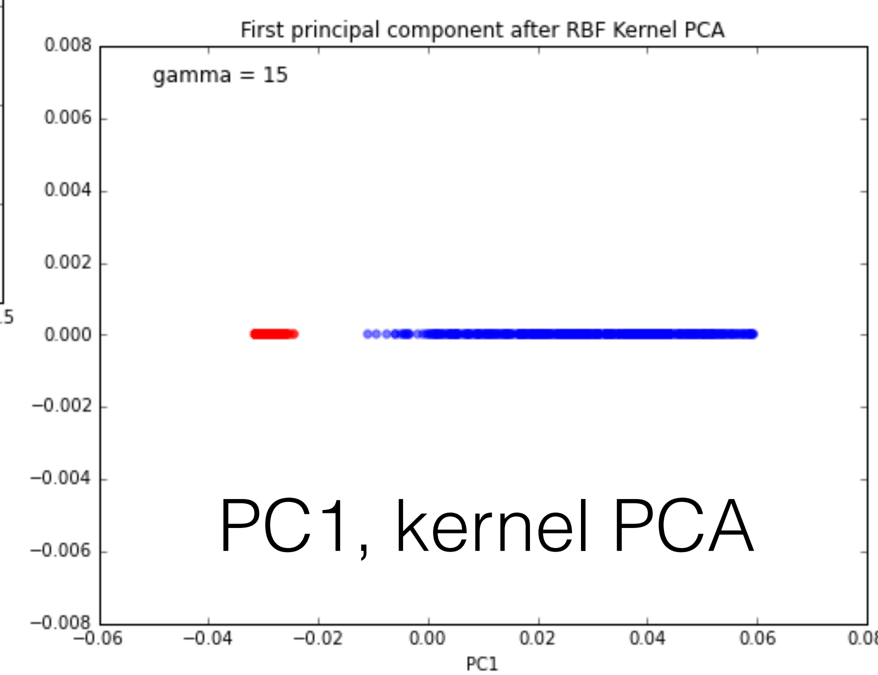
Kernel PCA

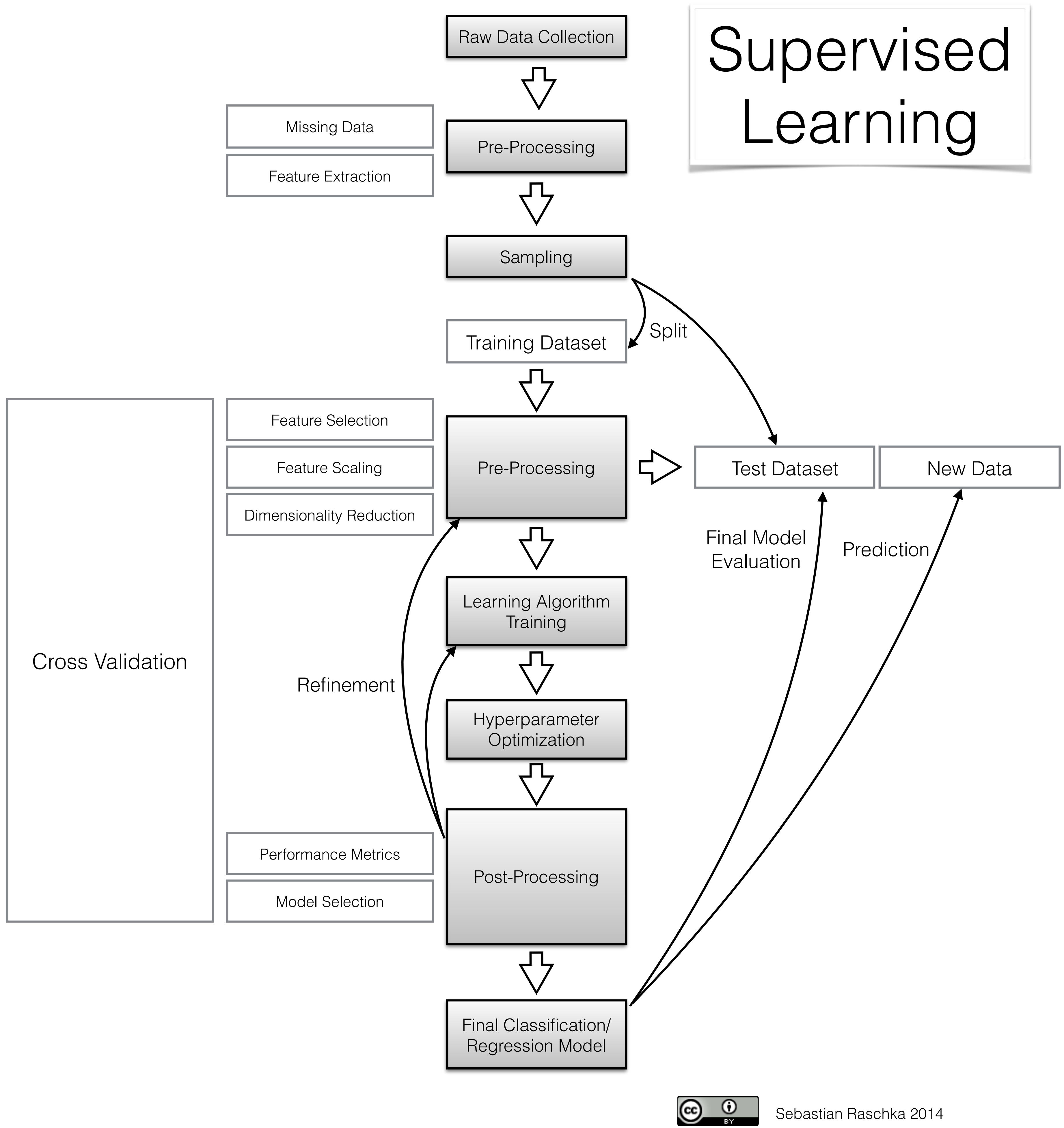
$$\text{Cov} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i$$

$$\text{Cov} = \frac{1}{N}\sum_{i=1}^{N} \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_i)$$








Supervised Learning
Raw Data Collection
Missing Data
Feature Extraction
Pre-Processing
Sampling
Split
Training Dataset
Cross Validation
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Test Dataset
New Data
Final Model Evaluation
Prediction
Learning Algorithm Training
Refinement
Hyperparameter Optimization
Performance Metrics
Model Selection
Post-Processing
Final Classification/ Regression Model
Sebastian Raschka 2014

Thanks!

Questions?

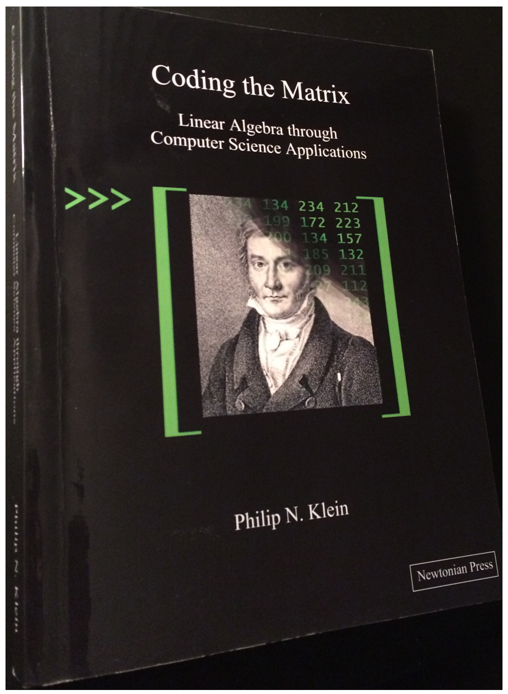
@rasbt

mail@sebastianraschka.com

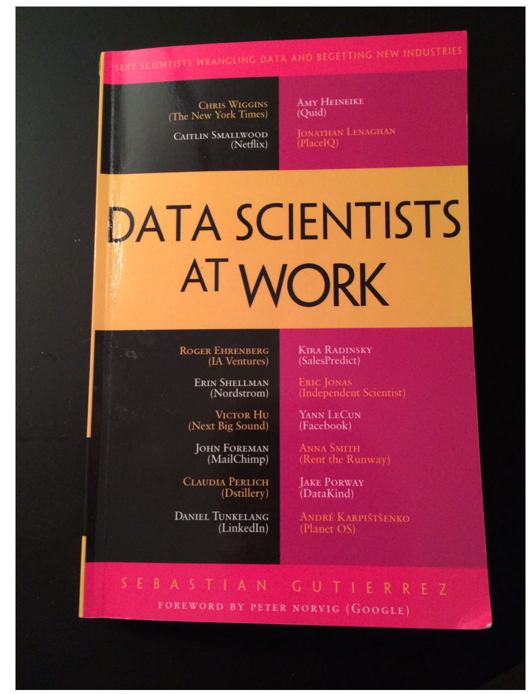
GitHub https://github.com/rasbt

Additional Slides

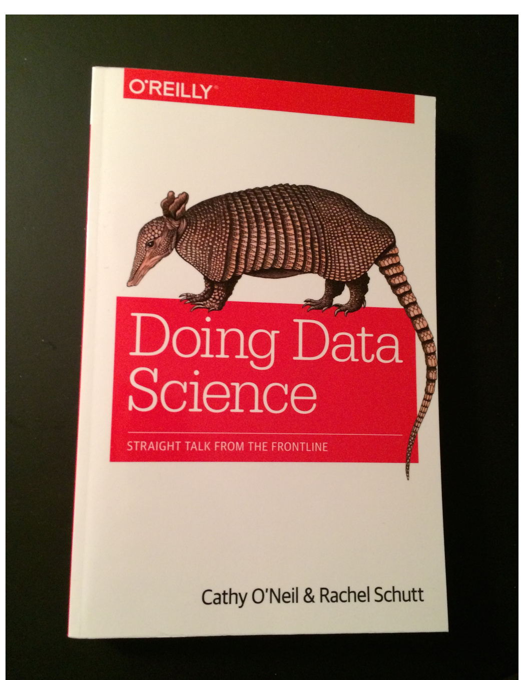
Inspiring Literature



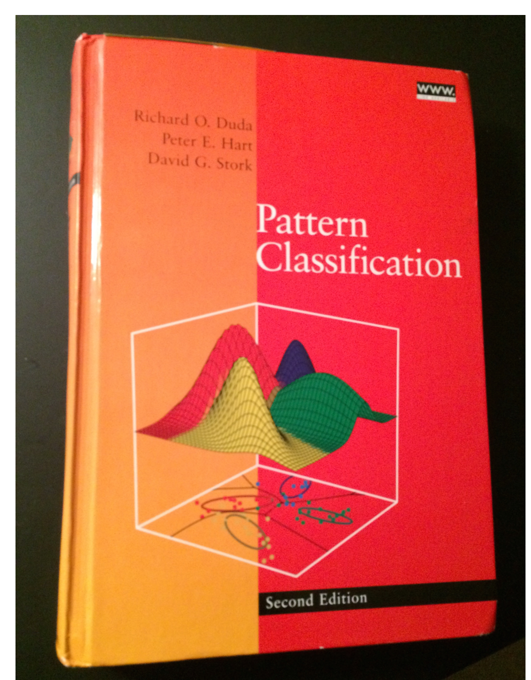
P. N. Klein. Coding the Matrix: Linear Algebra Through Computer Science Applications. Newtonian Press, 2013.



S. Gutierrez. Data Scientists at Work. Apress, 2014.

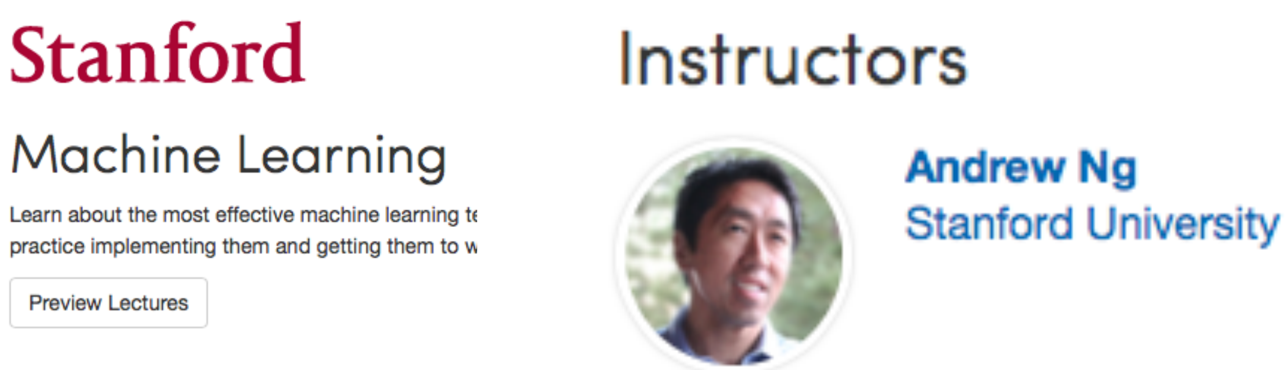


R. Schutt and C. O'Neil. Doing Data Science: Straight Talk from the Frontline. O'Reilly Media, Inc., 2013.



R. O. Duda, P. E. Hart, and D. G. Stork. Pattern classification. 2nd. Edition. New York, 2001.

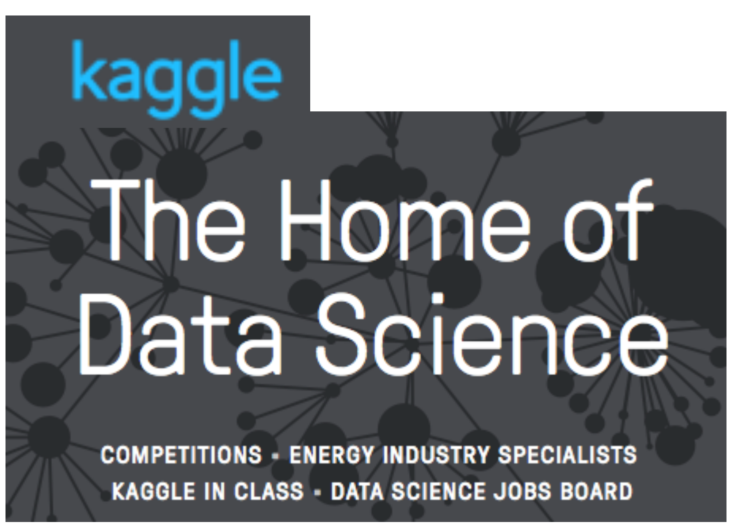
Useful Online Resources



https://www.coursera.org/course/ml

Cross Validated

http://stats.stackexchange.com



http://www.kaggle.com

My Favorite Tools

scikit learn

http://scikit-learn.org/stable/

NumPy

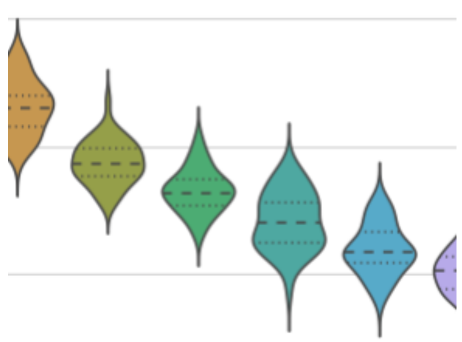
pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

http://www.numpy.org

http://pandas.pydata.org

Seaborn



http://stanford.edu/~mwaskom/software/seaborn/

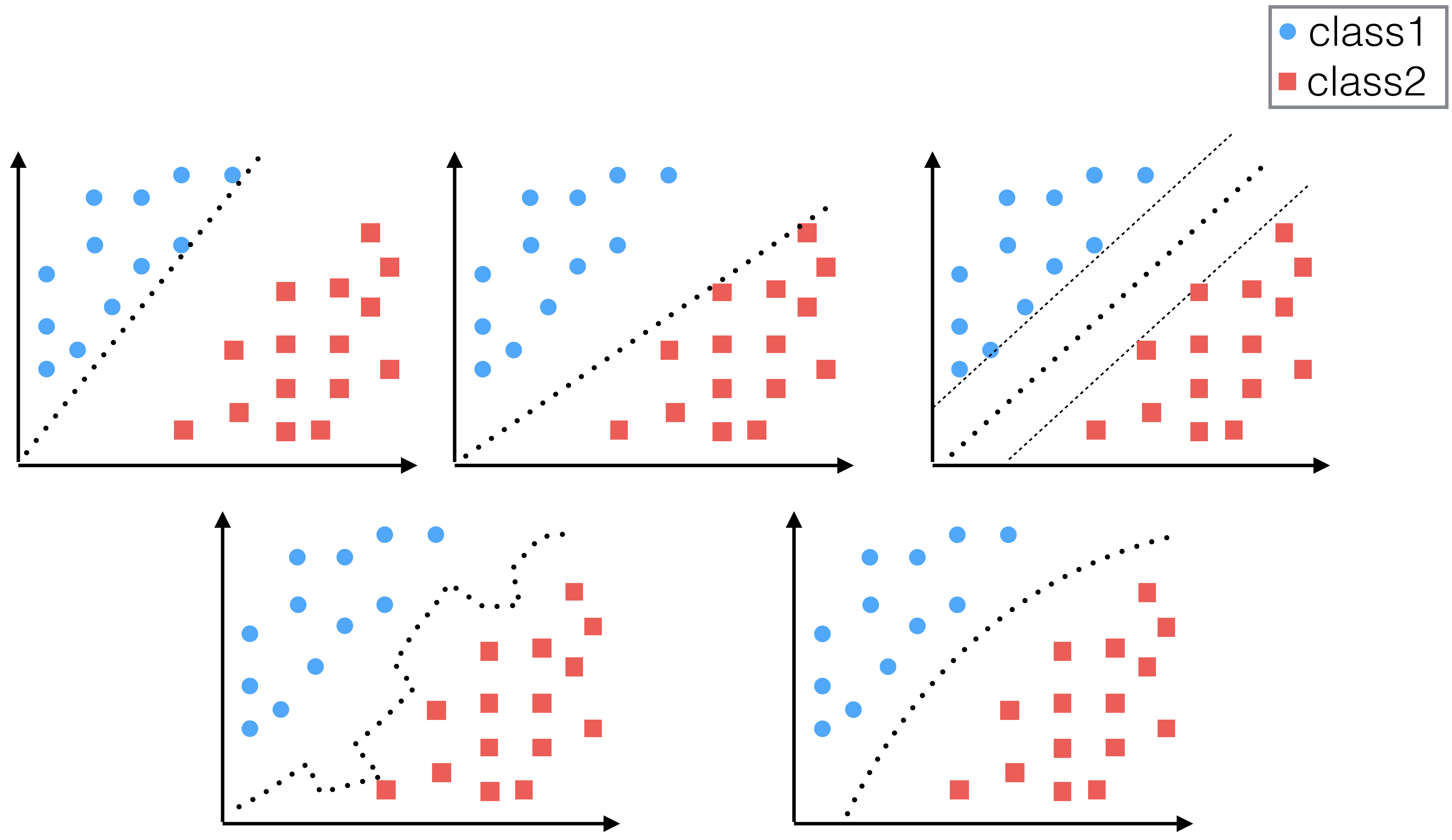
IP[y]: IPython Interactive Computing

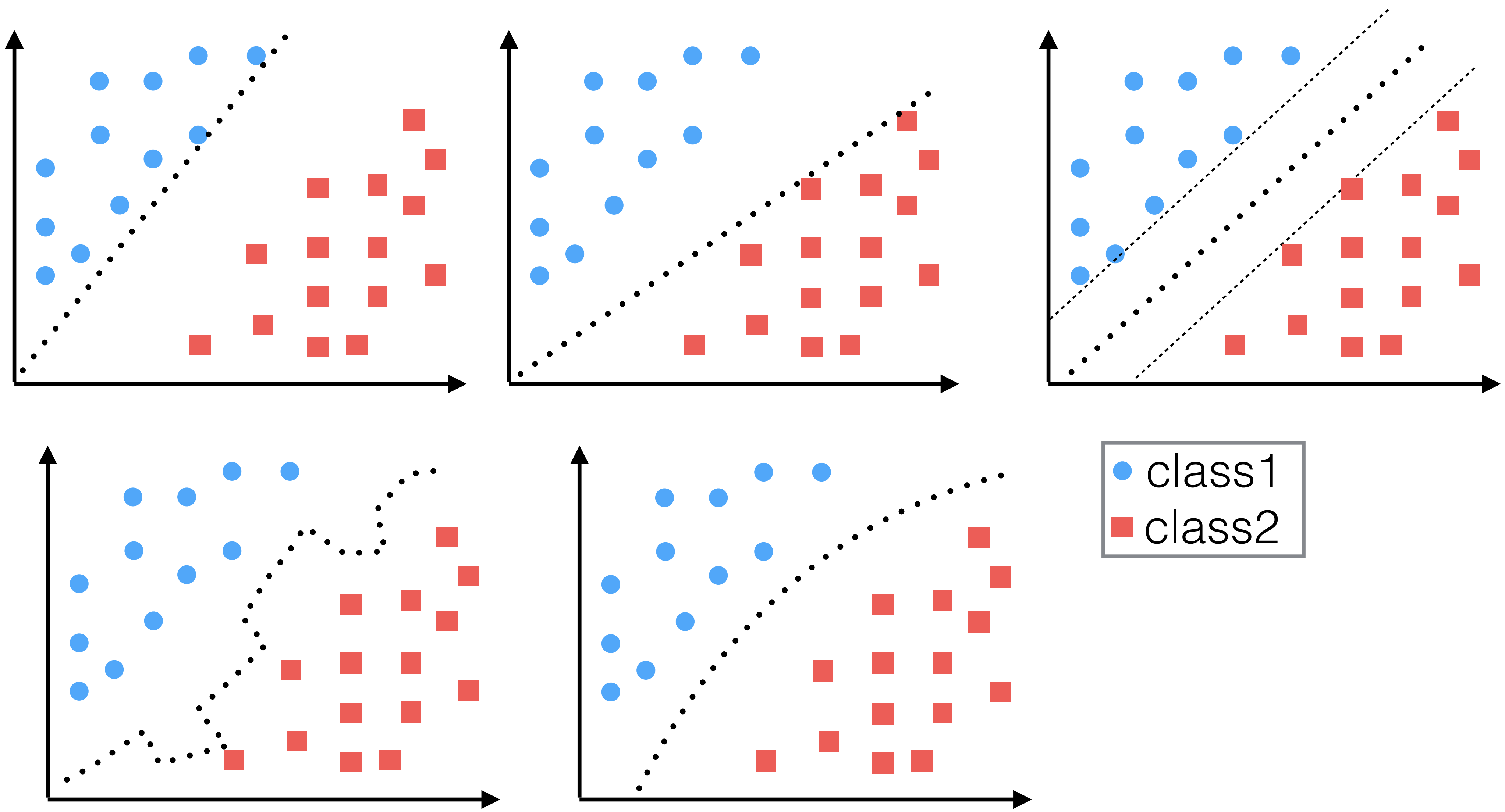
Install · Docs · Videos · News · Cite · Sponsors · Donate

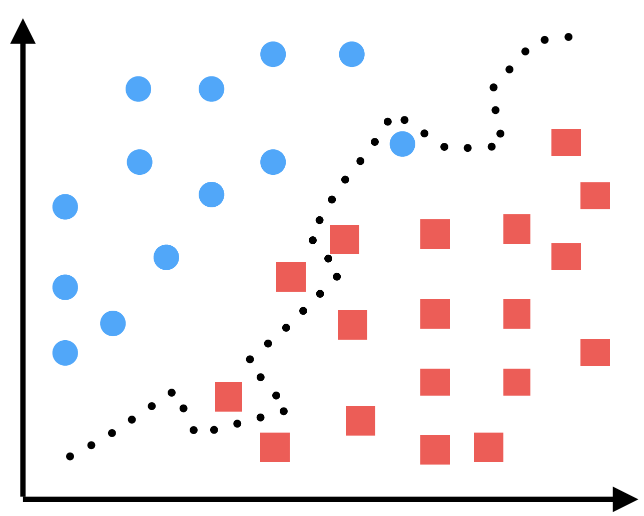
The IPython Notebook

http://ipython.org/notebook.html

Which one to pick?







Generalization error!

The problem of overfitting