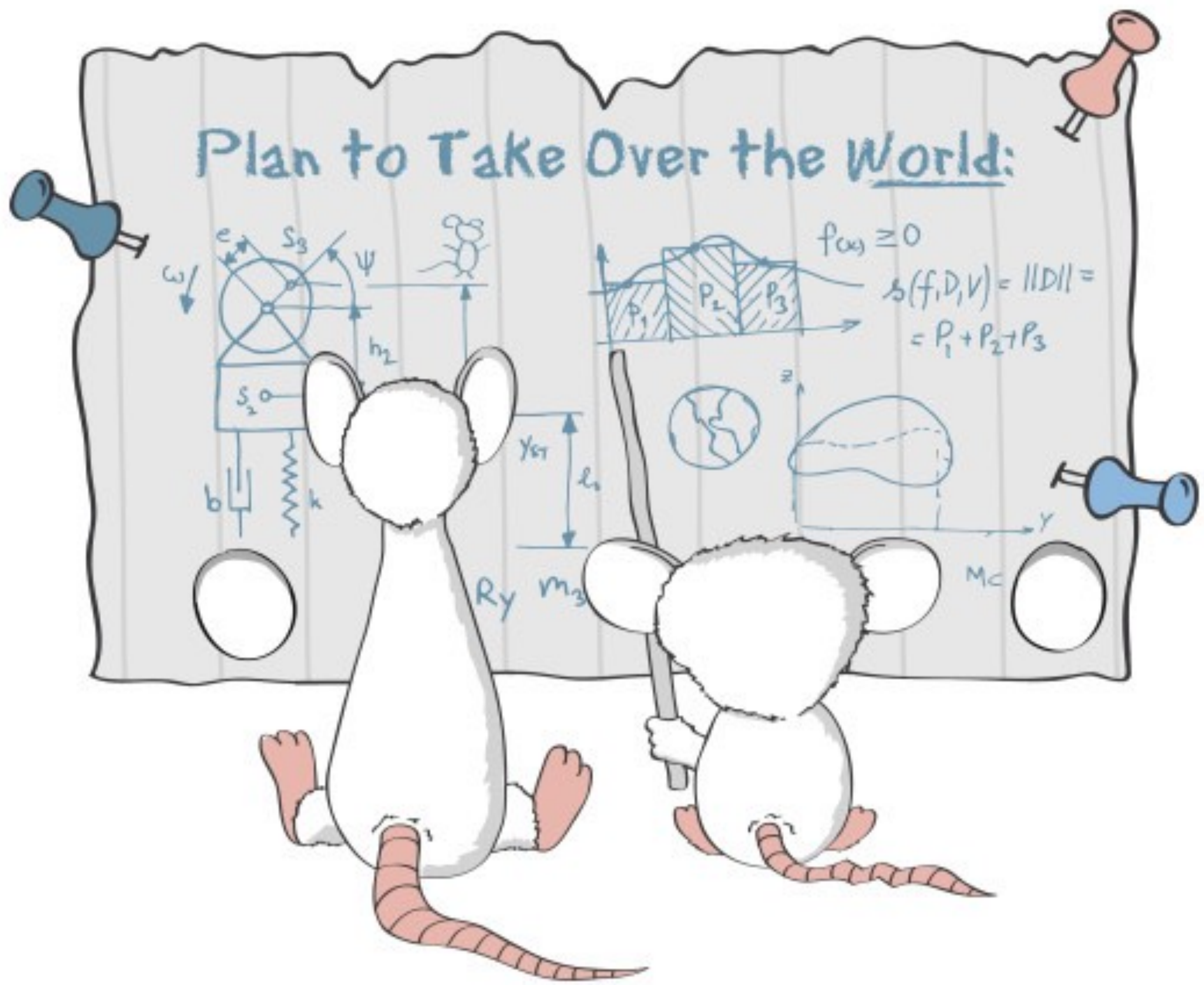


Lecture #2

F5611 Machine Learning for Astronomers
by Martin Topinka

https://github.com/toastmaker/f5611-ML4A

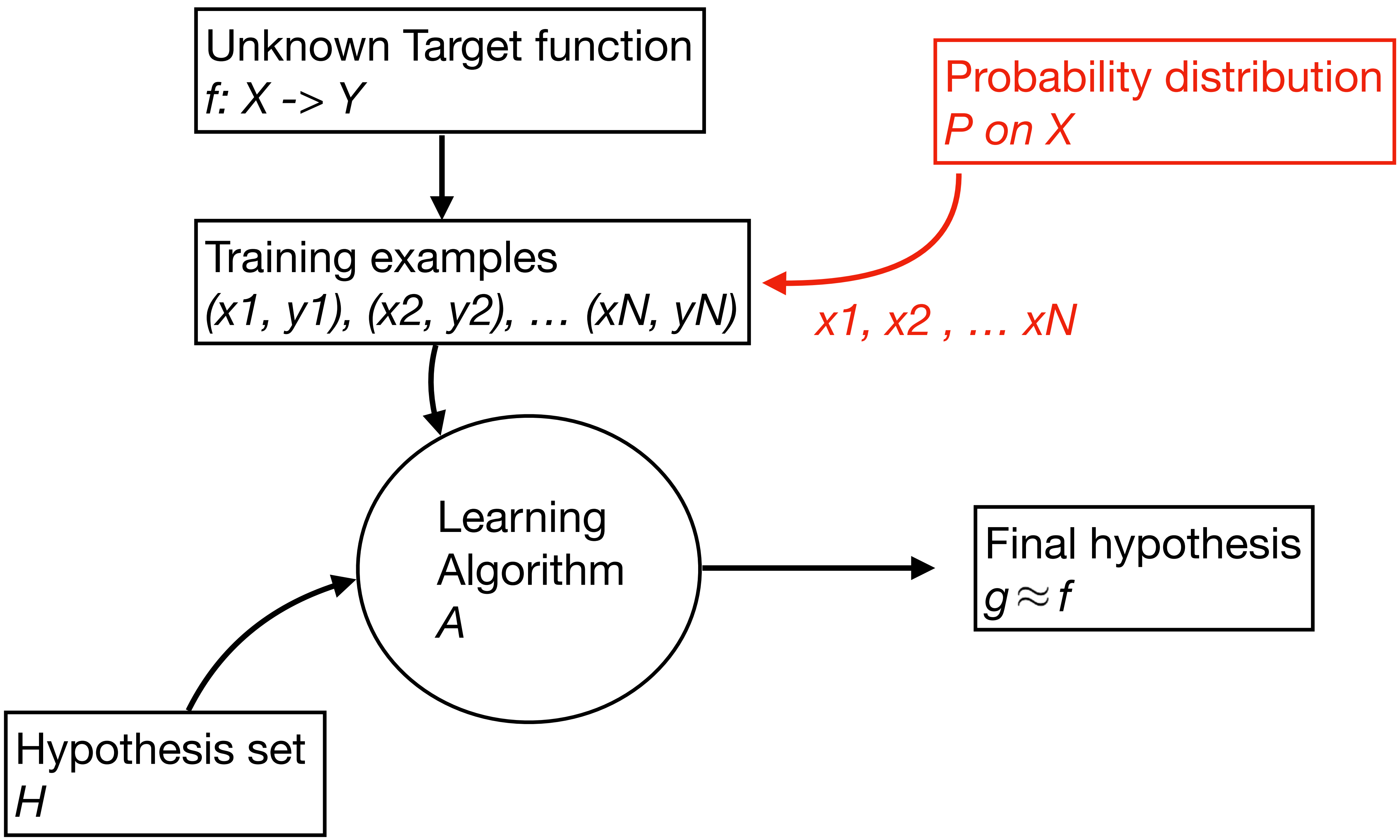
Syllabus

- Introduction to machine learning, history...
- Principles of machine learning
- Supervised, unsupervised machine learning
- Classification vs regression
- Loss function, accuracy measures
- Bias-variance tradeoff
- Model validation
- Introduction to scikit-learn and its API
- Basic machine learning algorithms (SVM, KNN, K-mean, Logistic regression, Decision Trees, Random Forest)
- Curse of dimensionality
- Feature selection, data reduction (PCA)
- Advanced algorithms (bagging, boosting, voting)
- Hands on session scikit-learn with GRB classification, QSO's vs stars…
- Hyper-parameter fine tuning
- Imbalanced classes
- Neural network, perceptron
- Deep learning neural networks
- Regularisation, dropout
- Deep learning with Convolutional Neural Networks
- Encoder-Decoder, Auto-encoder
- GAN
- Training data generators
- Introduction to Keras/TensorFlow
- Hands on session in Keras (developing a NN to classify stars/QSOs; developing a deep convNN auto-encoder for finding transients)
- Optional: Gaussian Processes

Essentials of Learning

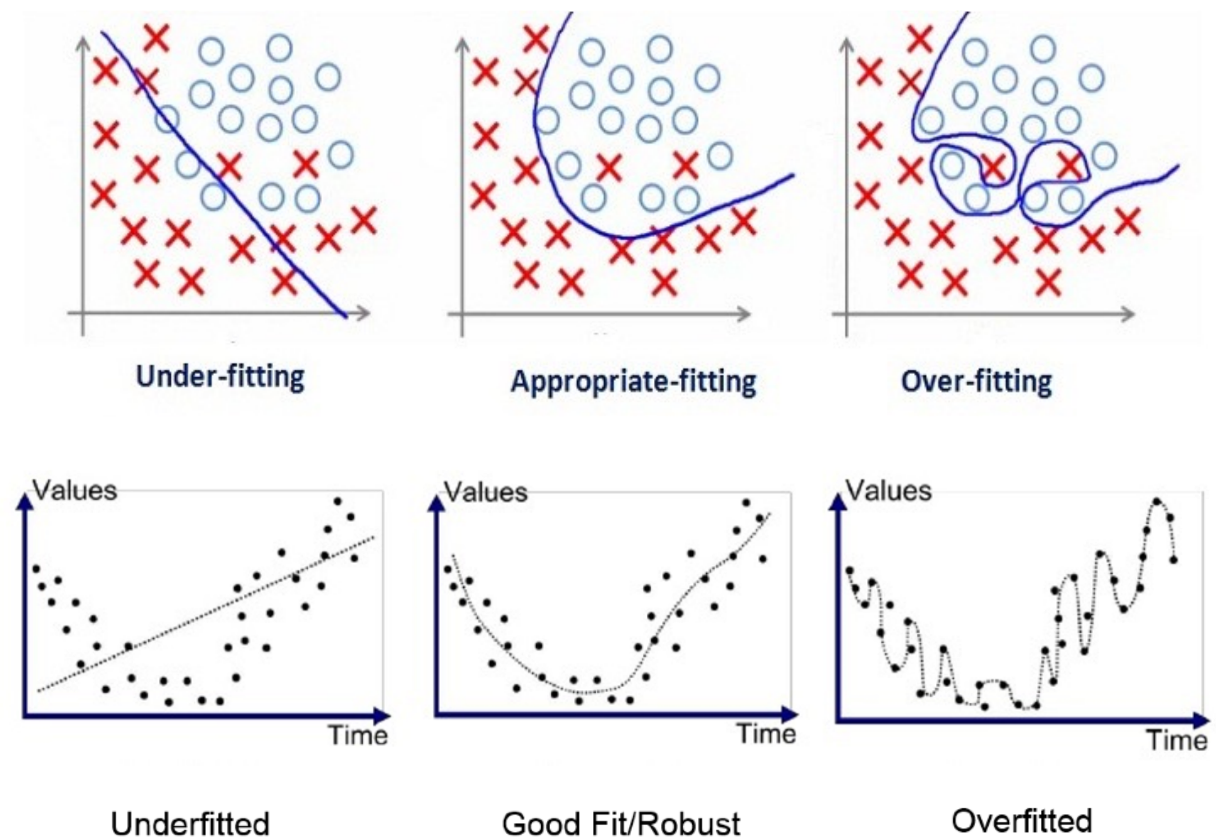
- Pattern must exist
- Mapping “target function” is unknown or expensive to calculate
- We have the data (and computing resources…)
- Data sample is representative

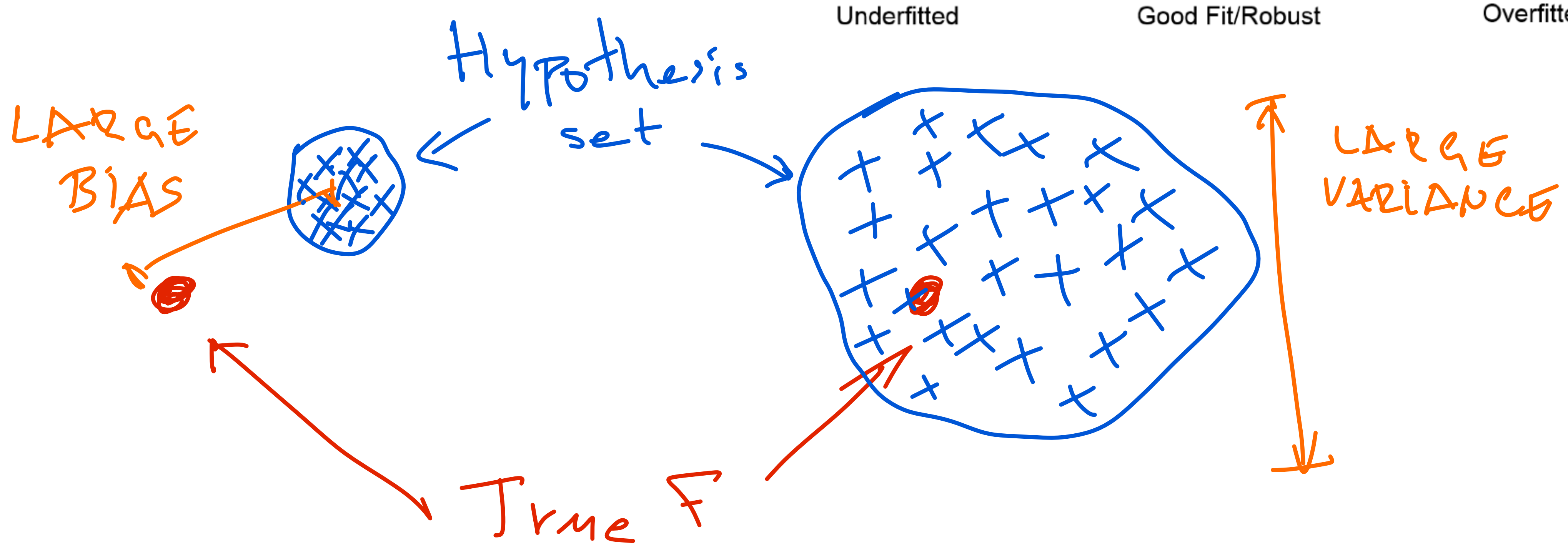
Learning Diagram



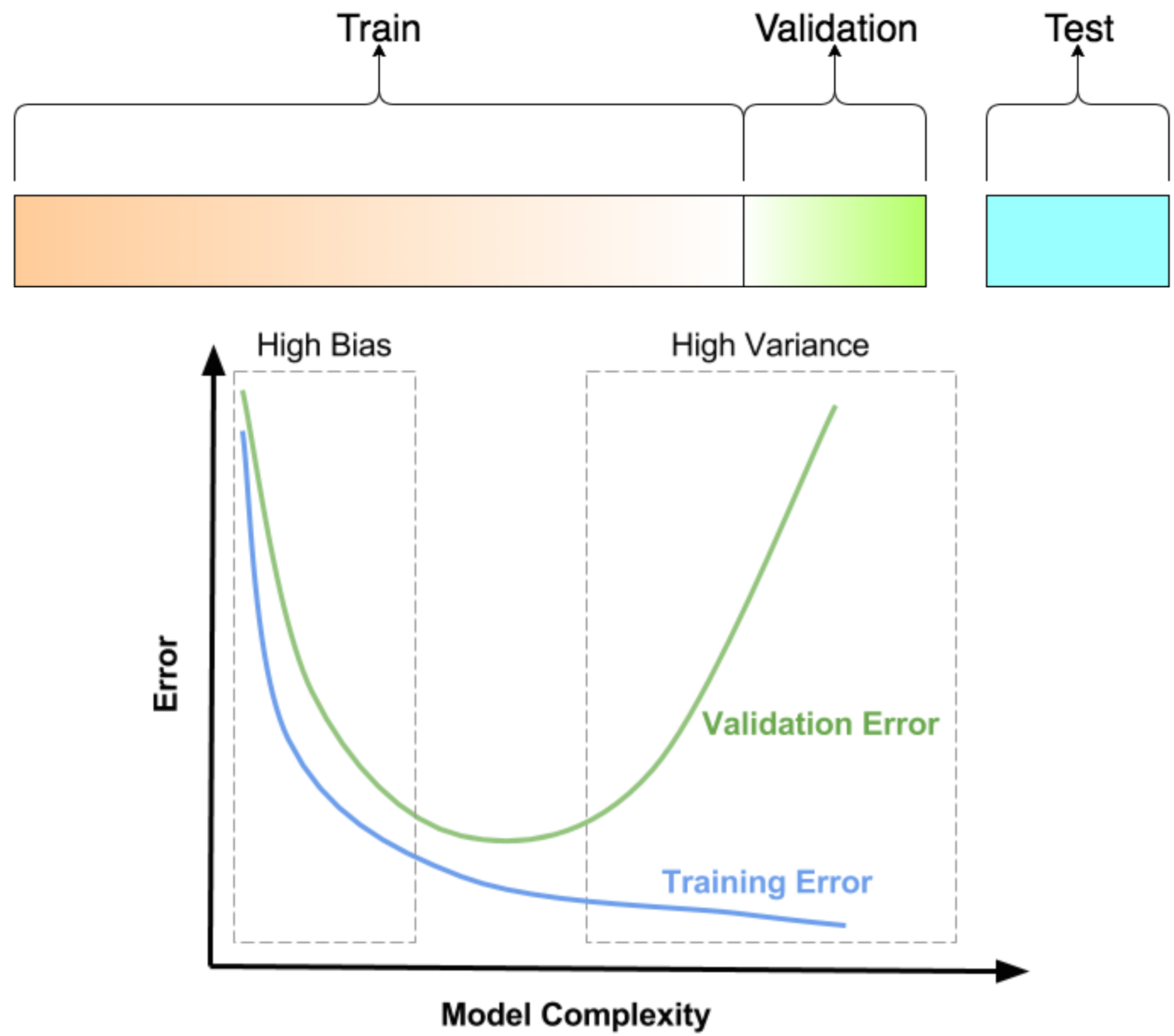
Bias - Variance Trade-off

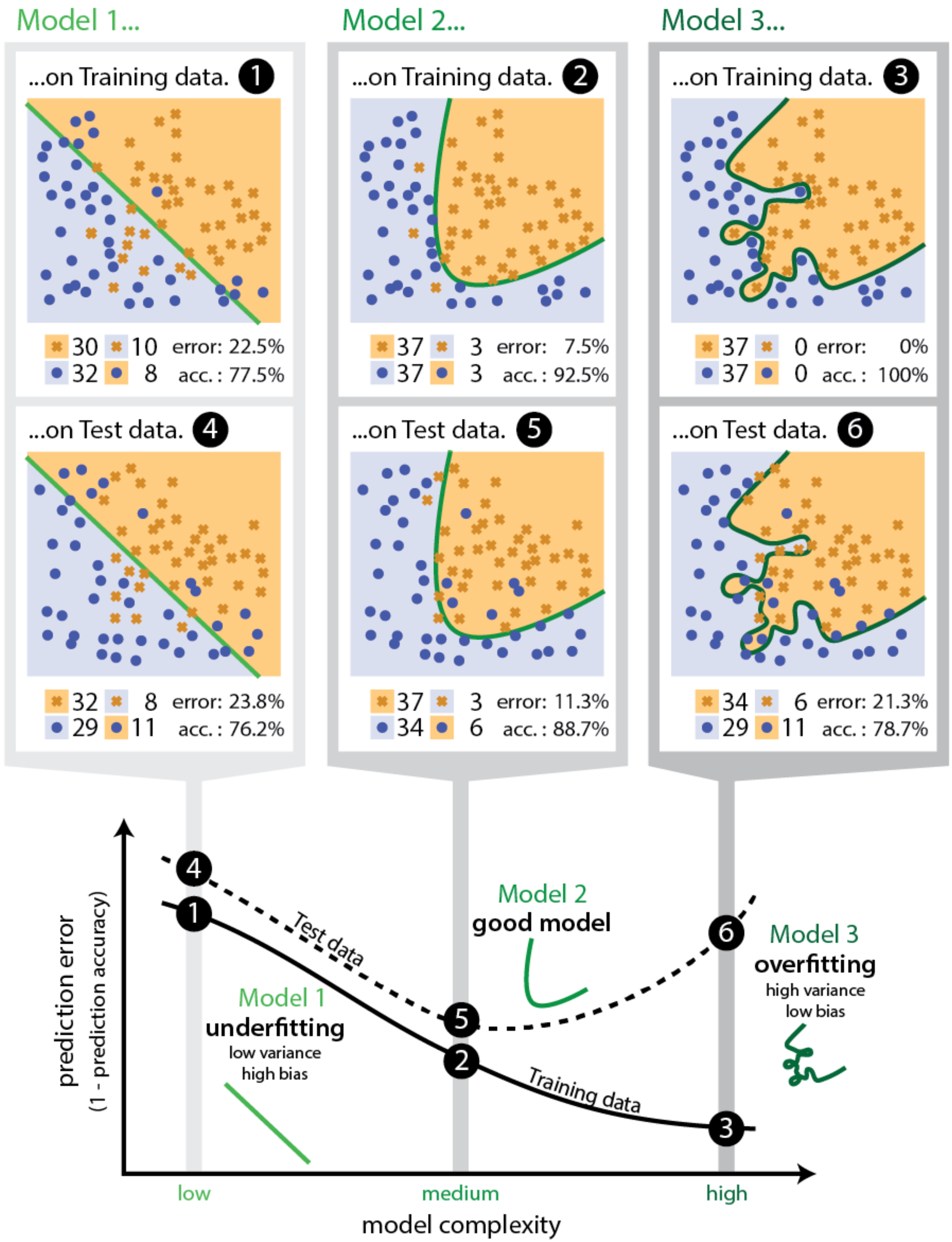
- Under-fitting/over-fitting
- in sample error vs out of sample error
- VC dimension
  Vapnik–Chervonenkis





(cross)-Validation




Model 1...
Model 2...
Model 3...
...on Training data. 1
30 10 error: 22.5%
32 8 acc.: 77.5%
...on Training data. 2
37 3 error: 7.5%
37 3 acc.: 92.5%
...on Training data. 3
37 0 error: 0%
37 0 acc.: 100%
...on Test data. 4
32 8 error: 23.8%
29 11 acc.: 76.2%
...on Test data. 5
37 3 error: 11.3%
34 6 acc.: 88.7%
...on Test data. 6
34 6 error: 21.3%
29 11 acc.: 78.7%
prediction error
(1 - prediction accuracy)
Test data
Training data
Model 1
underfitting
low variance
high bias
Model 2
good model
Model 3
overfitting
high variance
low bias
low
medium
high
model complexity

	Remedies

High Bias	• Train longer
	• Increase model complexity
	  • more features
	  • more parameters,
	  • richer architecture

High Variance	• Get more data
	• Decrease model complexity
	  • less features
	  • less parameters,
	  • simpler architecture
	• Regularization
	• Early stopping
	• Drop-out

Data Augmentation:

- When more data are needed, make up new ones! (The way of the god.)
- Translate, rotate, flip, crop, lighten/darken, add noise, de-phase, etc.



Cats

Cat?

Dogs


50% dog, 50% cat?

Machines are lazy and love shortcuts

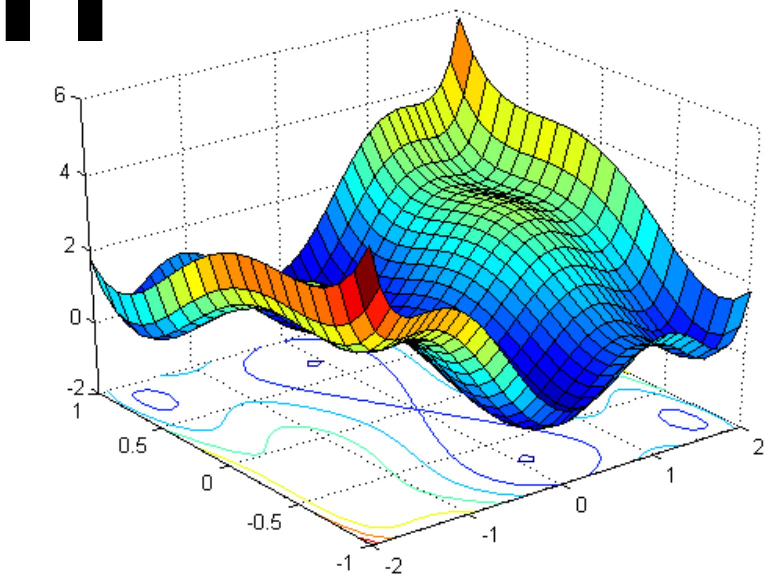
- A 1-2 years old needs about 2-4 cats to generalise how a cat looks like and to recognise new cats without over-fitting

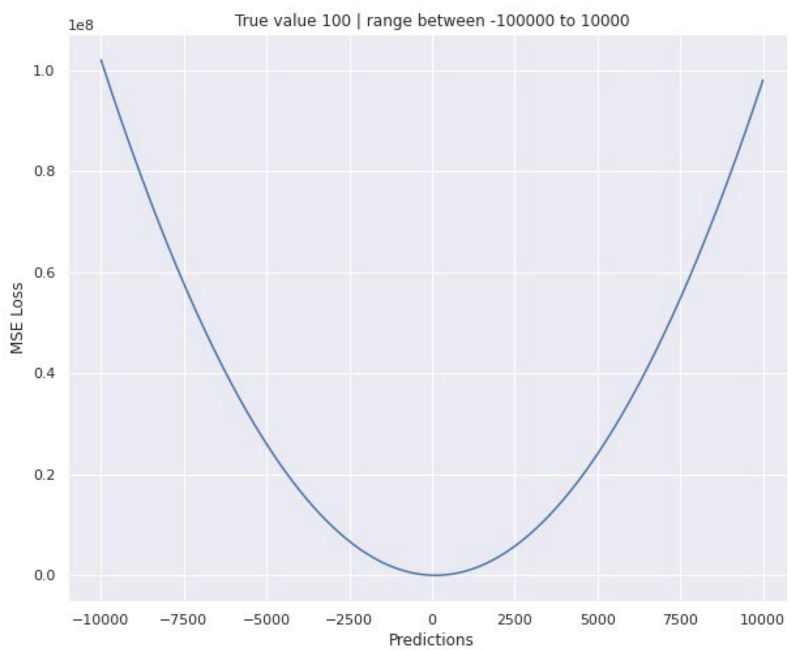
- Neural Network needs 50k objects to learn with much less neurons and still risks over-fitting

- *Garbage in, garbage out* — non-representative or poor quality data (noise, errors, outliers), irrelevant features

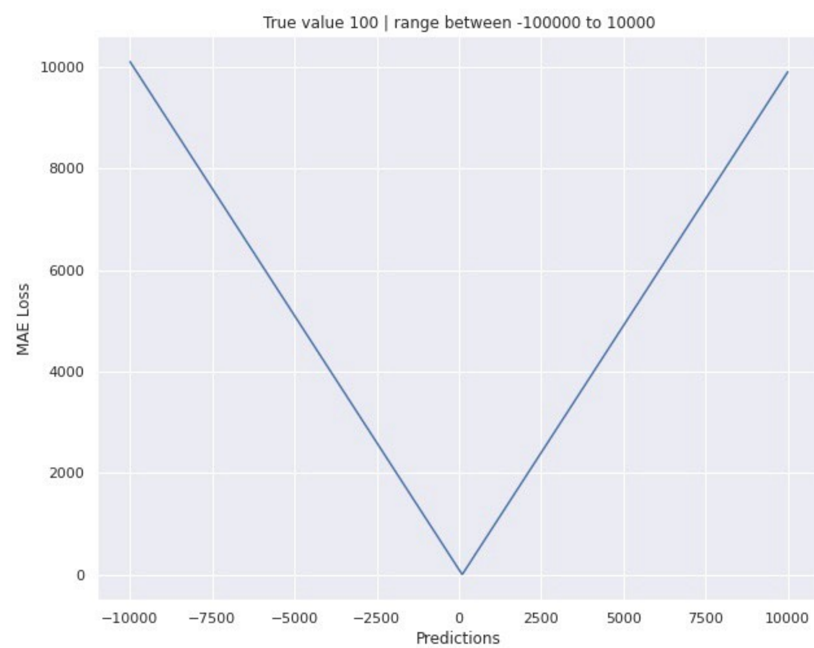
Loss function

- penalty for being wrong (L2, L1, …)
- function of model parameters

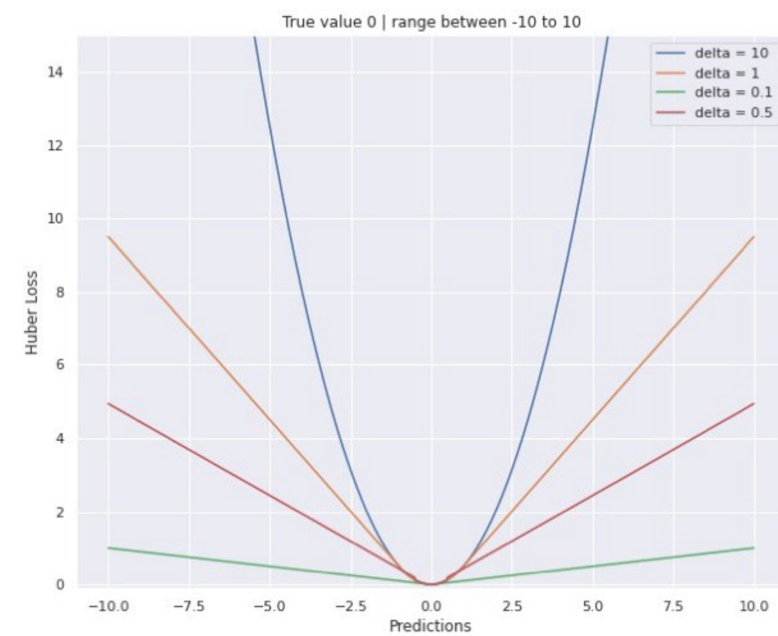




y_{true}-y_{pred}



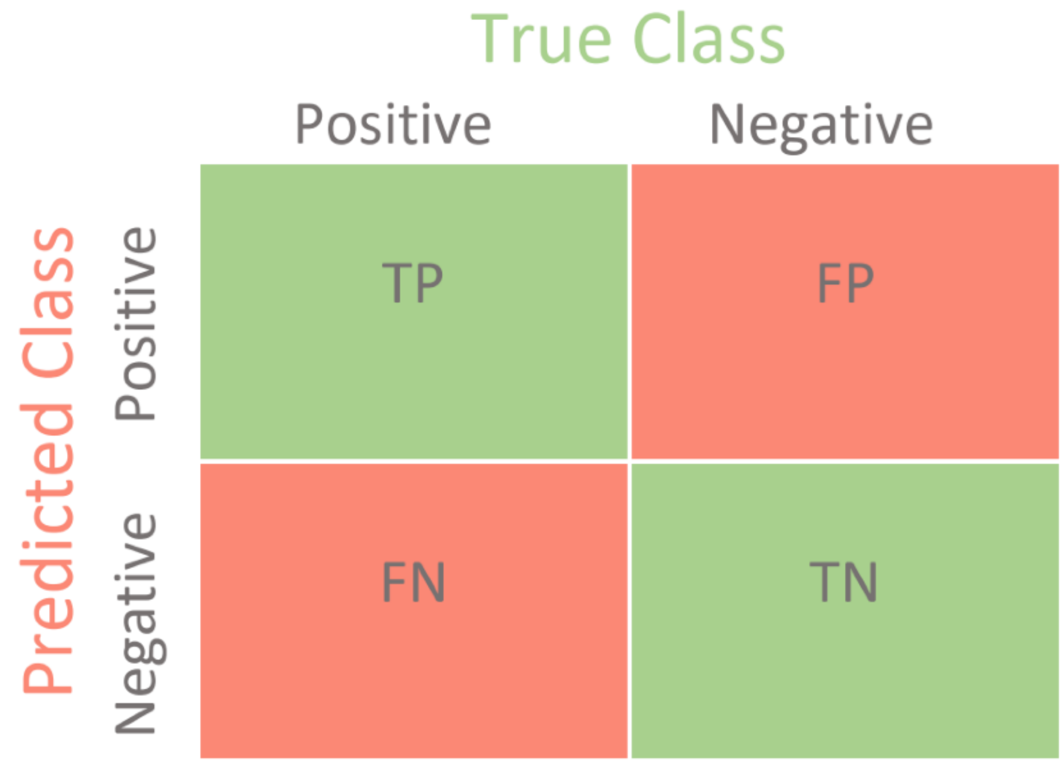
y_{true}-y_{pred}



y_{true}-y_{pred}

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Confusion matrix

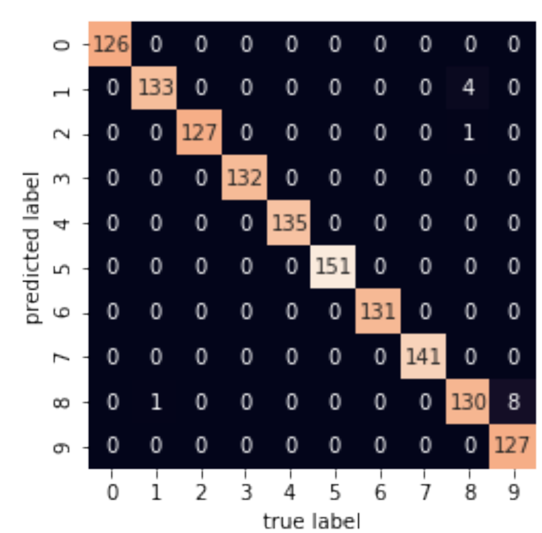


$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

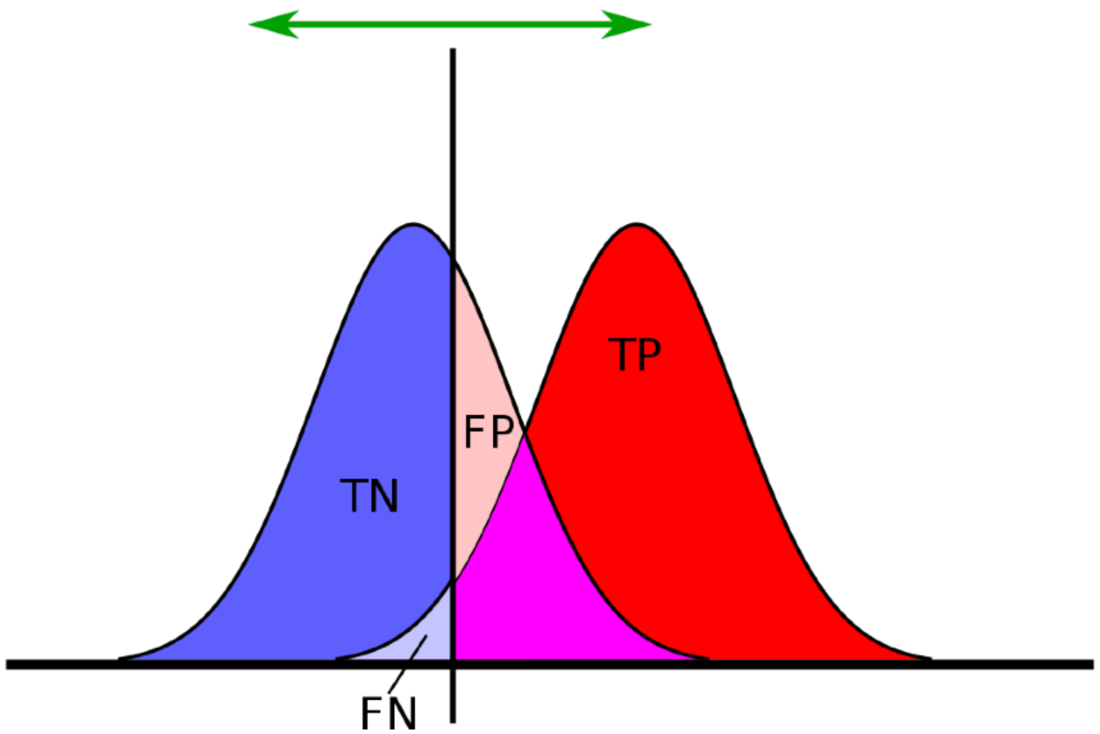
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

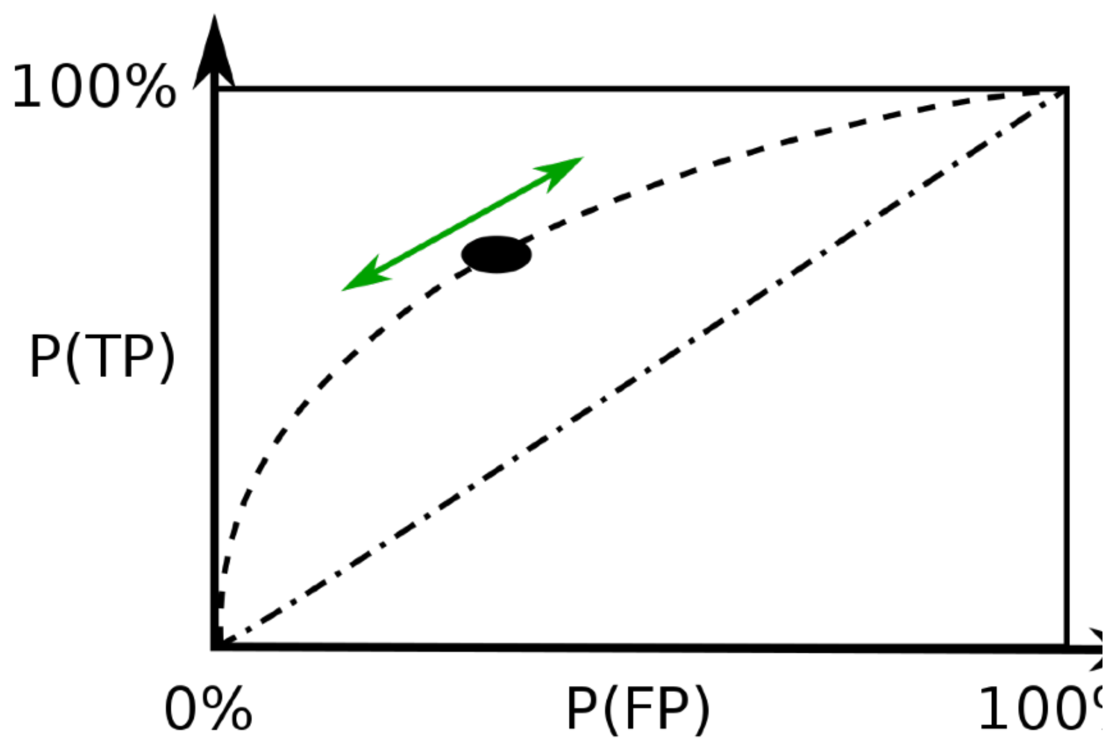
- tuneable for imbalanced classes and for weighted data

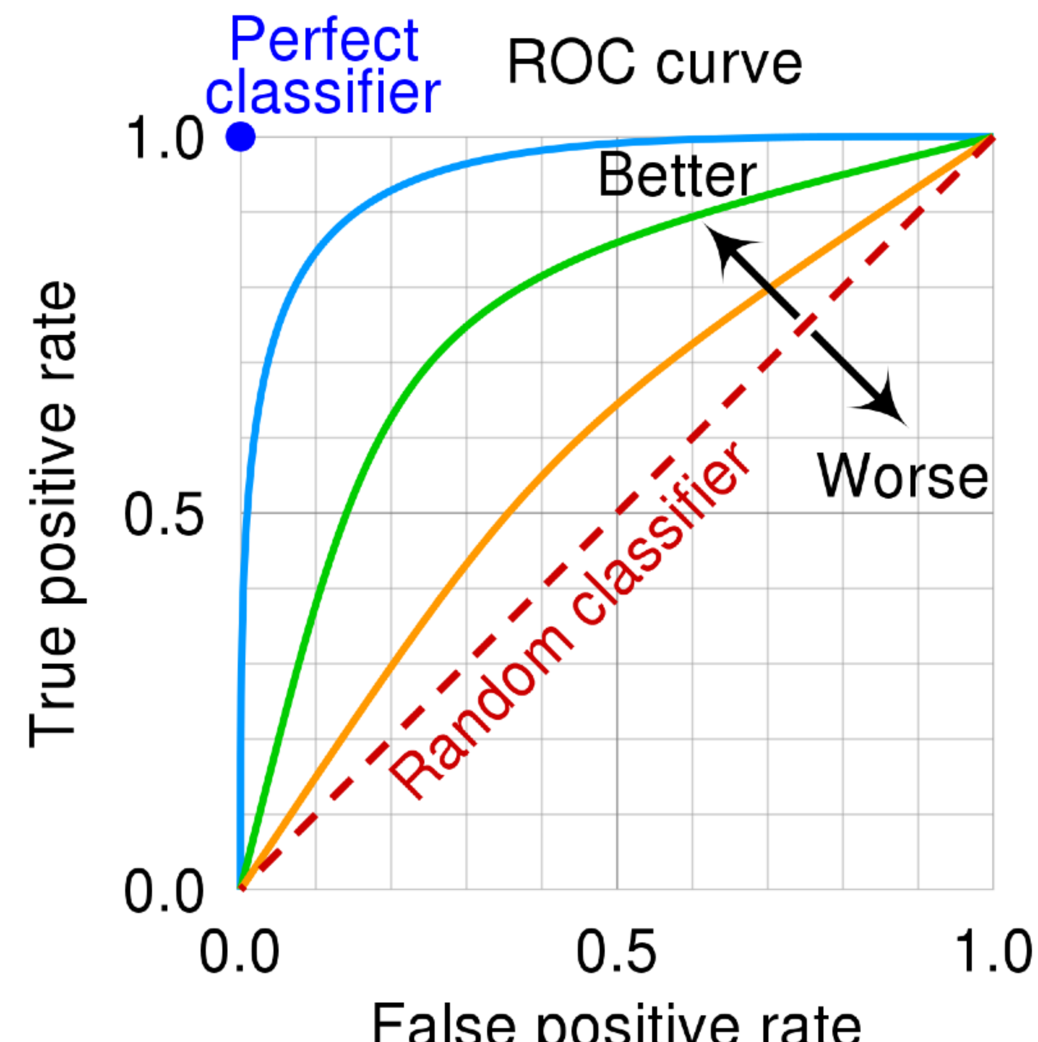


Receiver Operating Characteristic curve



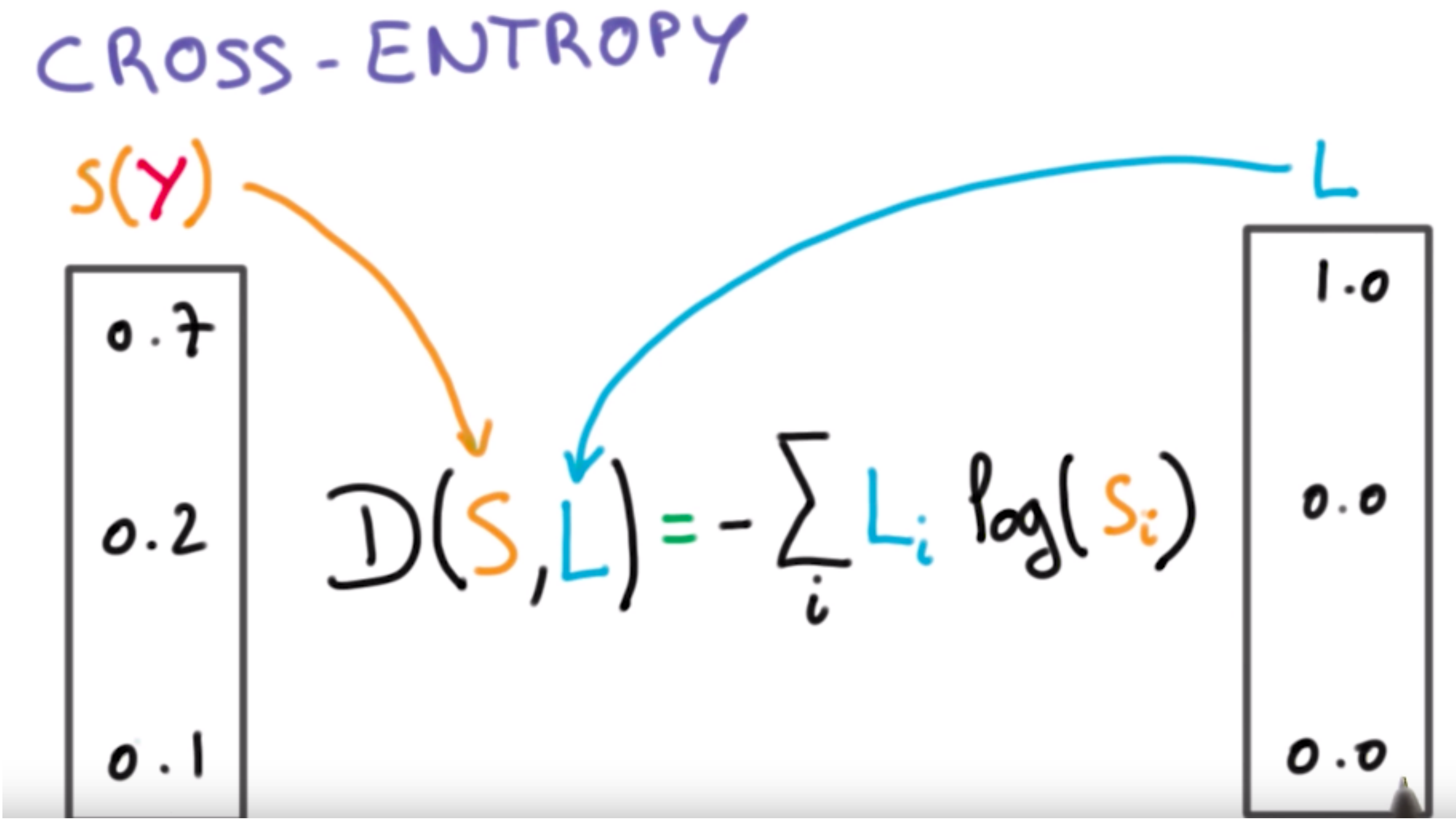
TP	FP
FN	TN





$$J = -\frac{1}{N}\left(\sum_{i=1}^{N} \mathbf{y_i} \cdot \log(\hat{\mathbf{y}}_\mathbf{i})\right)$$

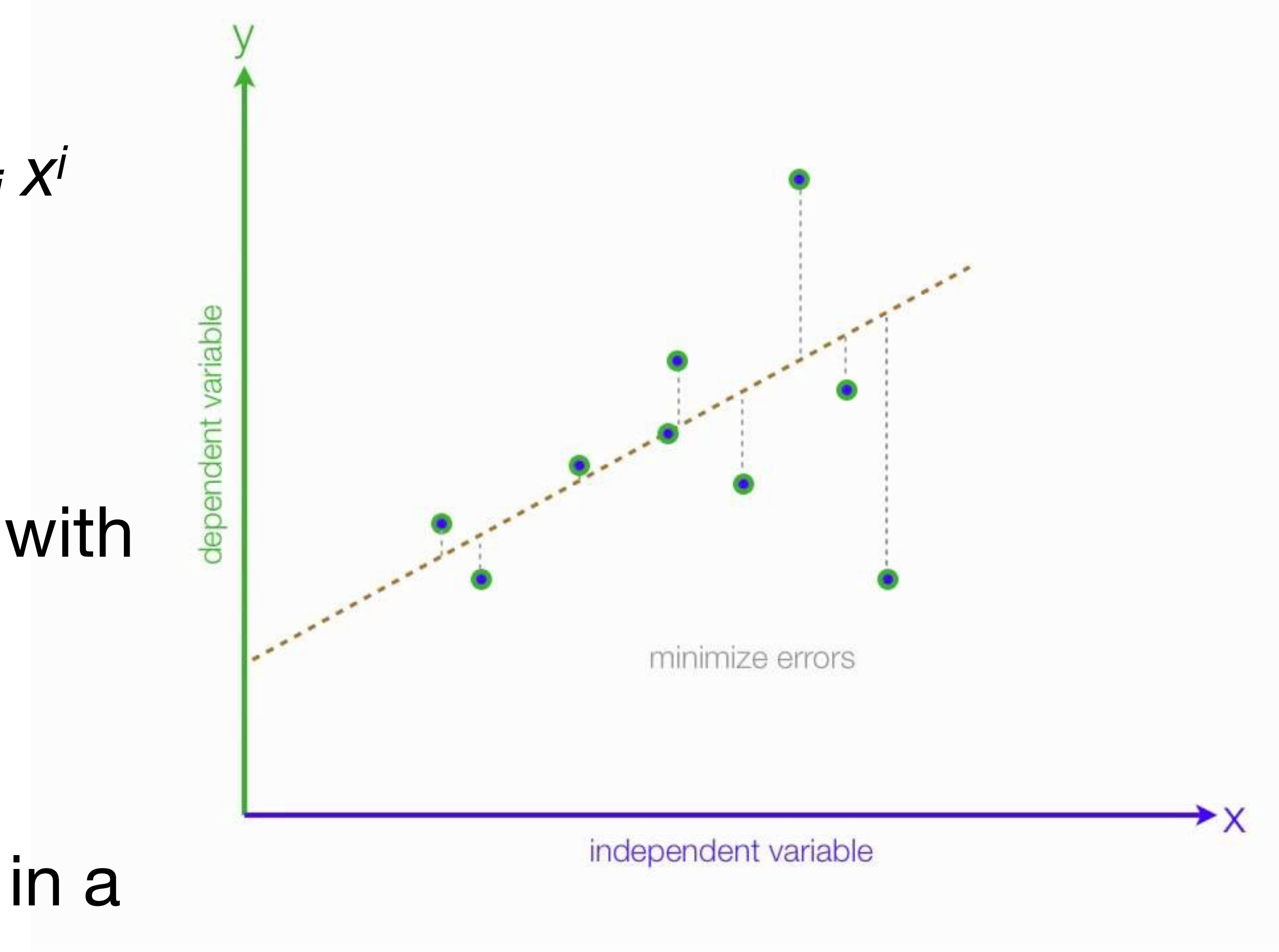
A way how to compare the predicted distribution of class probabilities to the true class probabilities



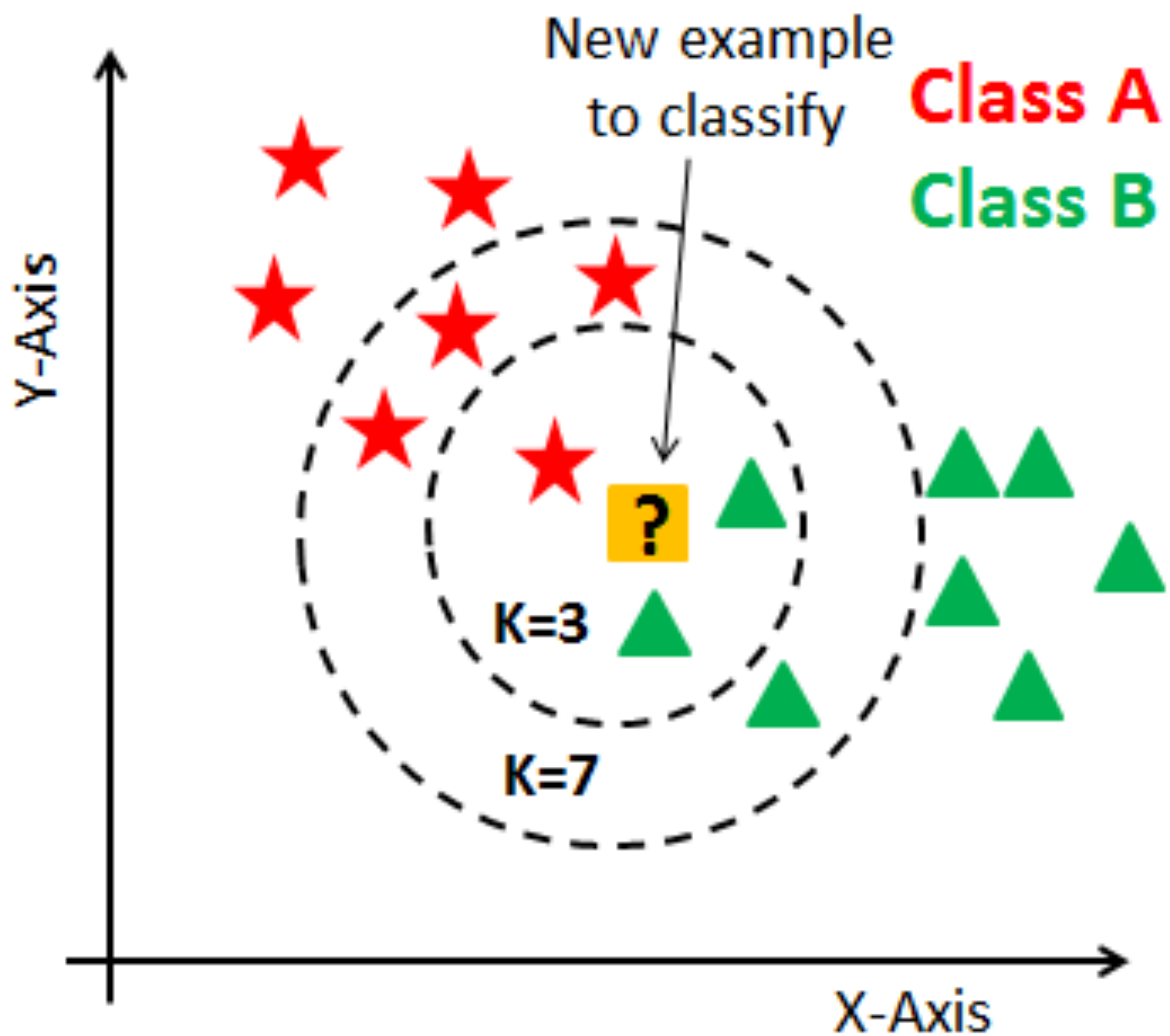
$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

Linear Regression

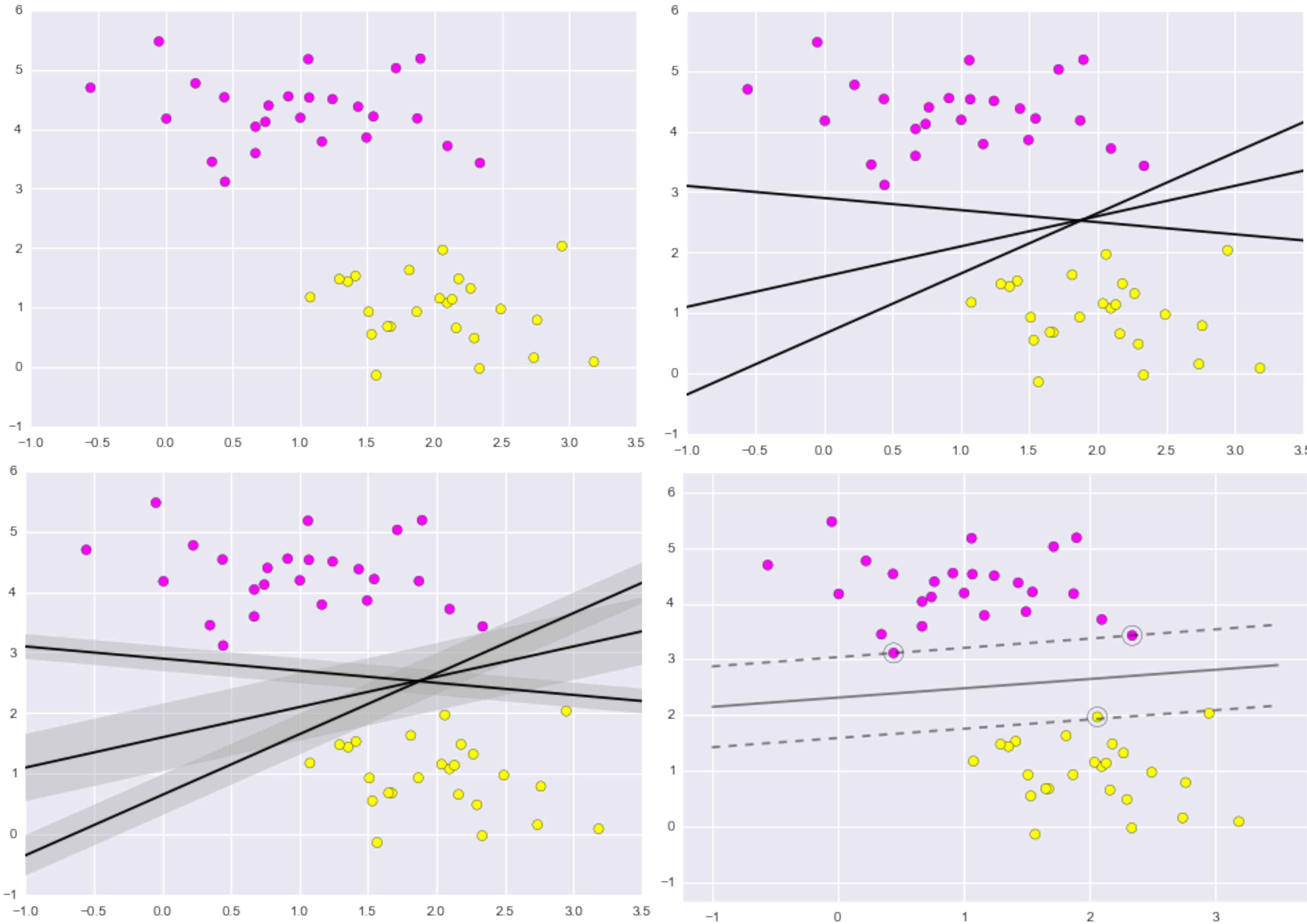
- Linear fitting in N-dim space
  $y = a\,x + b$, or more general $y = \sum a_i\, x^i$
- we have training pairs (x_i, y_i)
- **Linear in weights!** => works well with polynomials
  $a_0\, x^0 + a_1\, x^1 + a_2\, x^2 + a_3\, x^3 \ldots$
- We can transform our data points in a non-linear way and still use linear regression to find the fit
- We control the loss function *L1*, *L2*, …

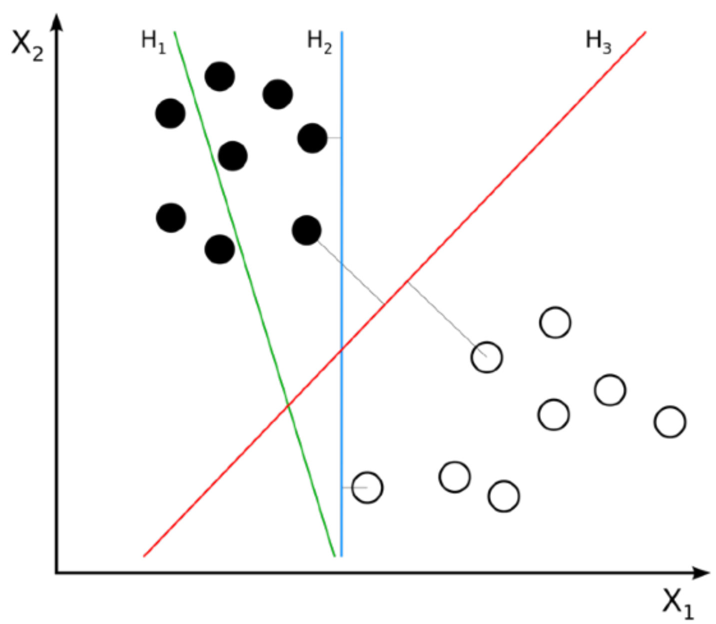


K Nearest Neighbours

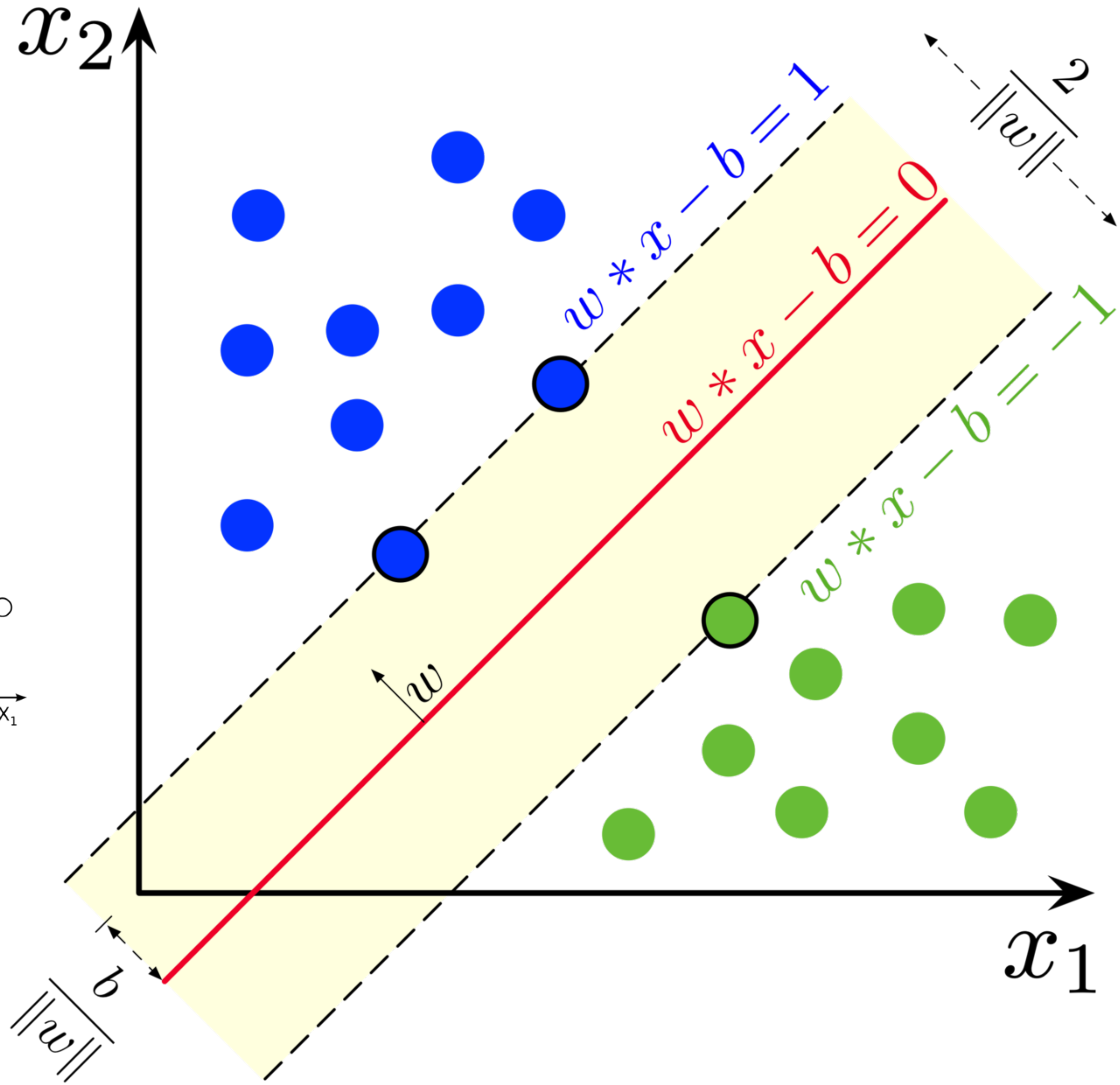


Support Vector Machine



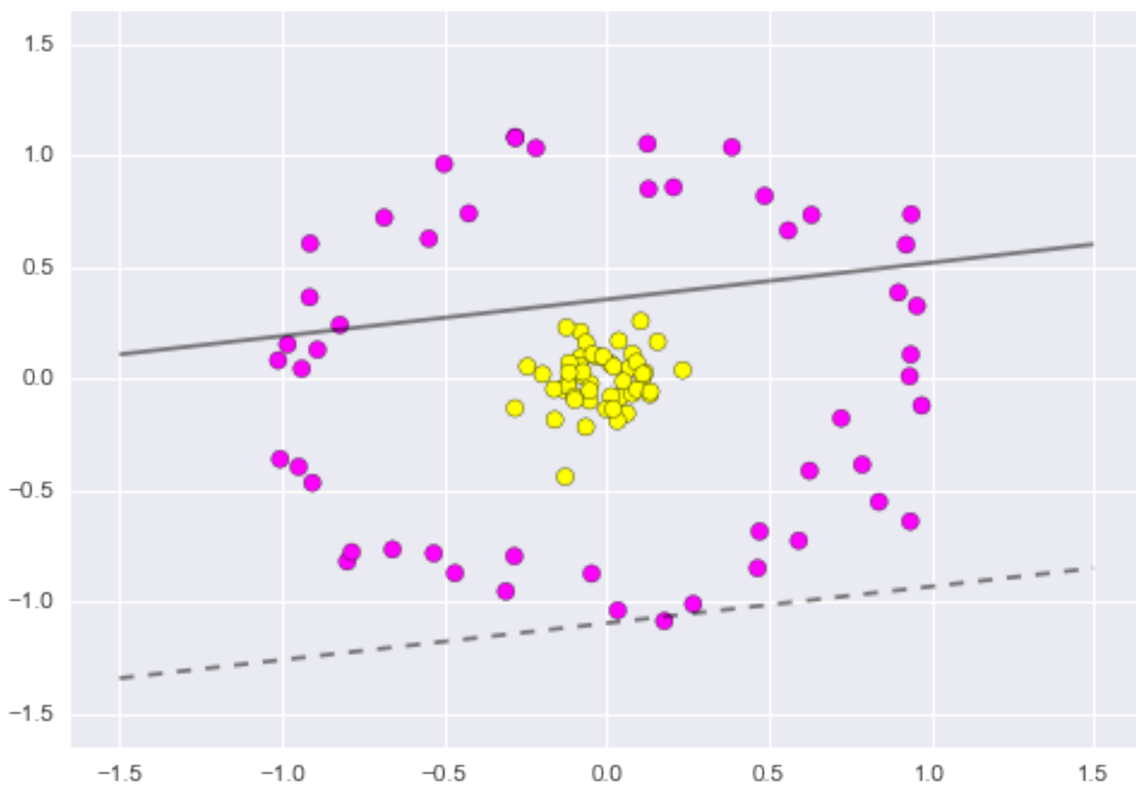


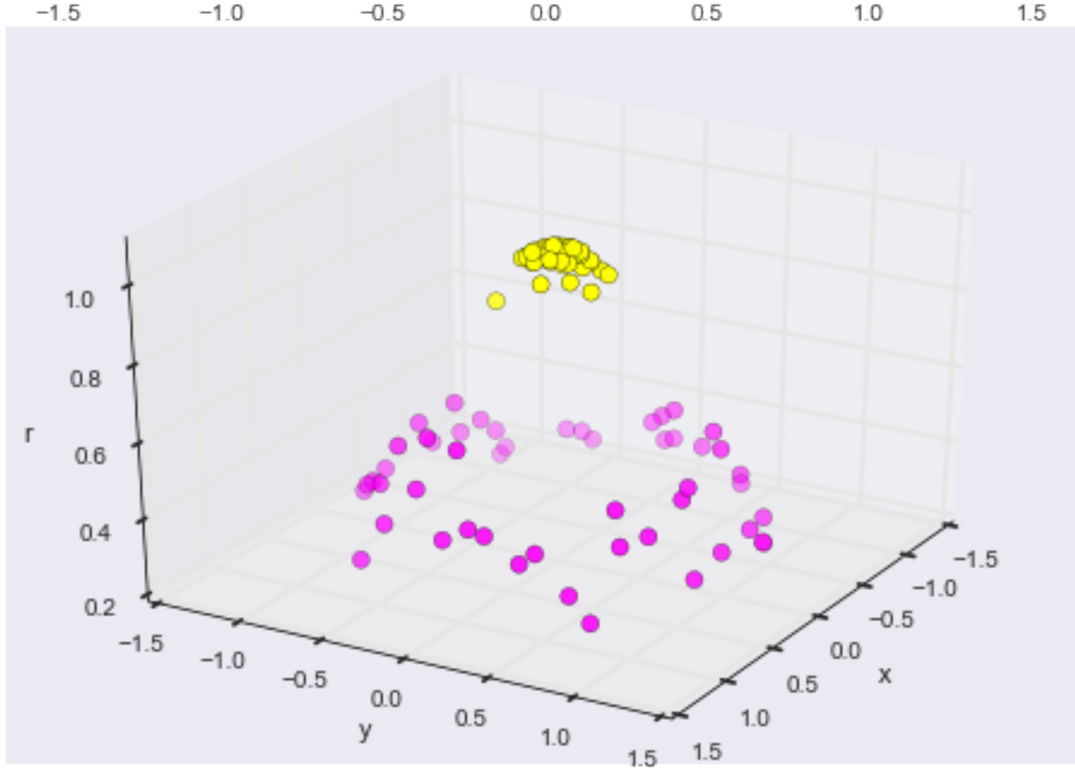
X_2
H_1
H_2
H_3
X_1

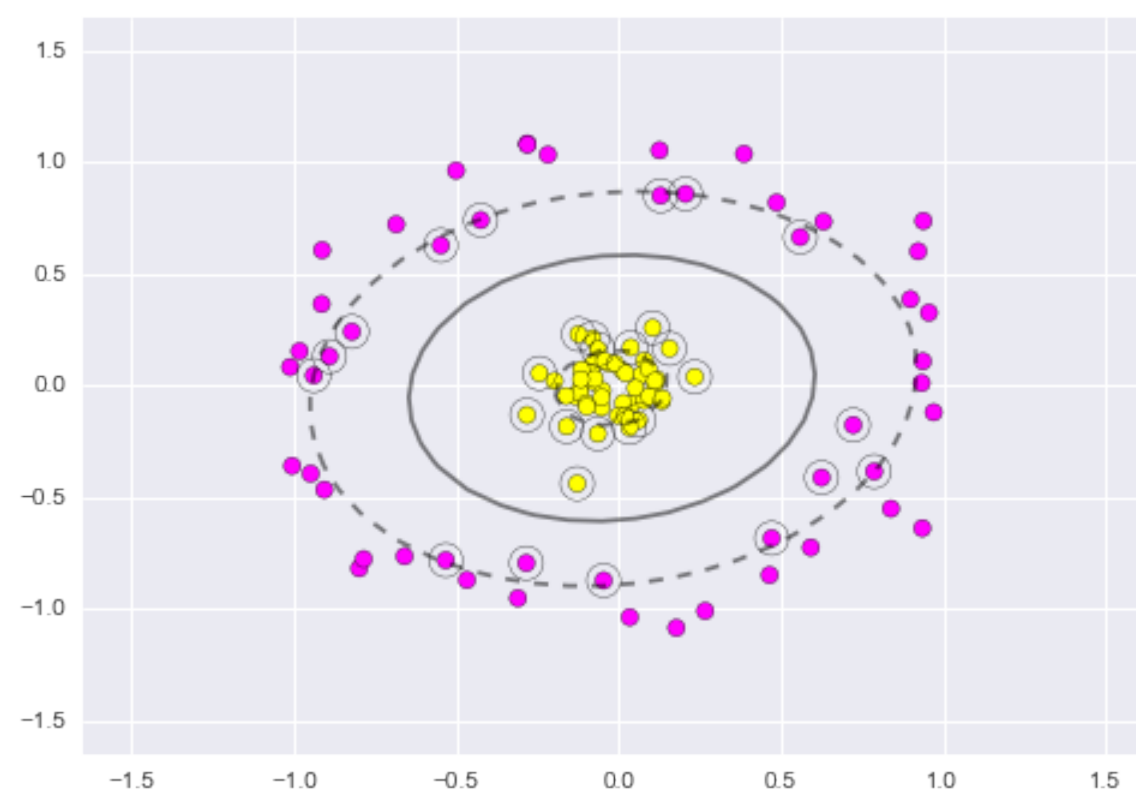


x_2
$w * x - b = 1$
$\frac{2}{\|w\|}$
$w * x - b = 0$
$w * x - b = -1$
w
x_1
$\frac{b}{\|w\|}$

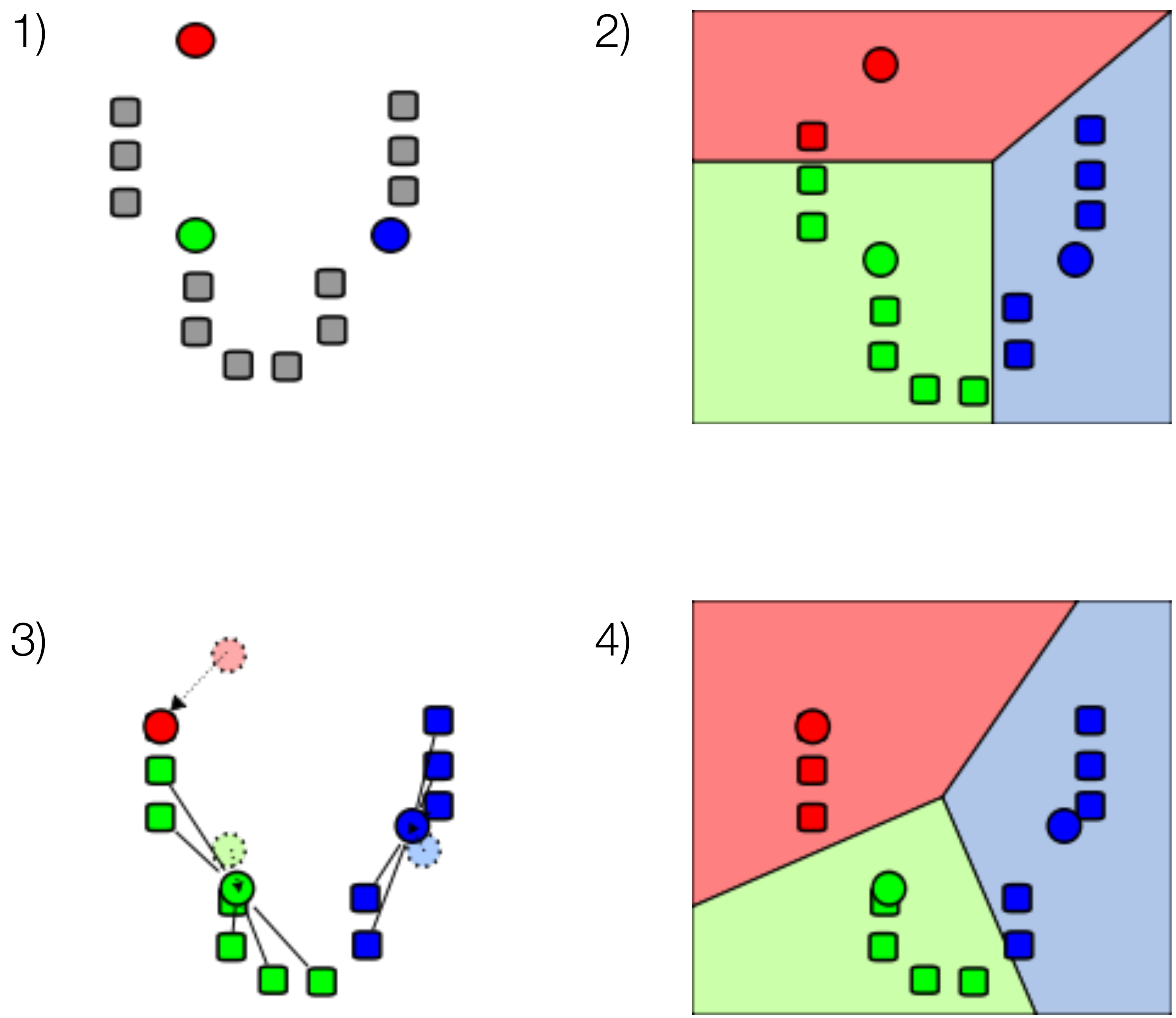
SVM with non-linear (kernel) transformation



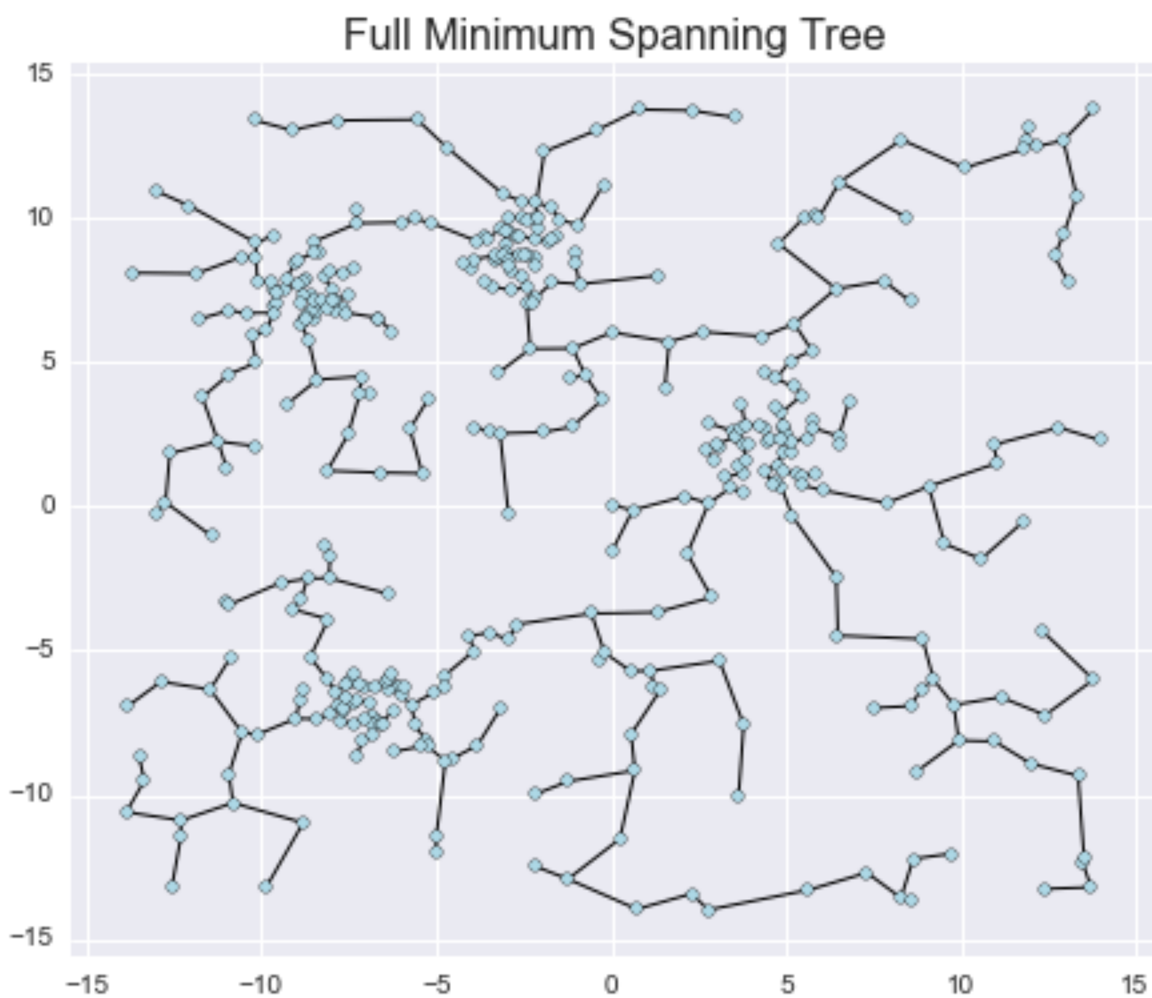


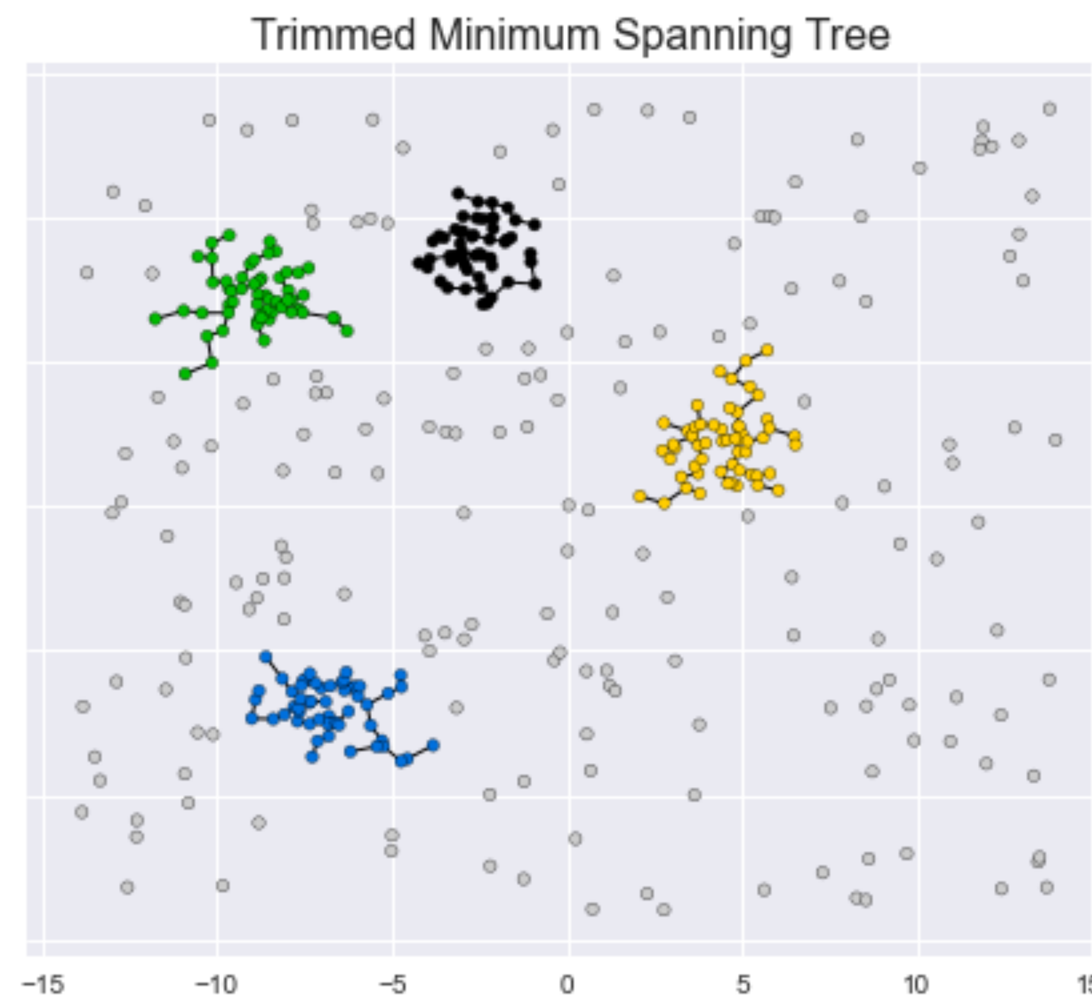


K-Mean Clustering (unsupervised)



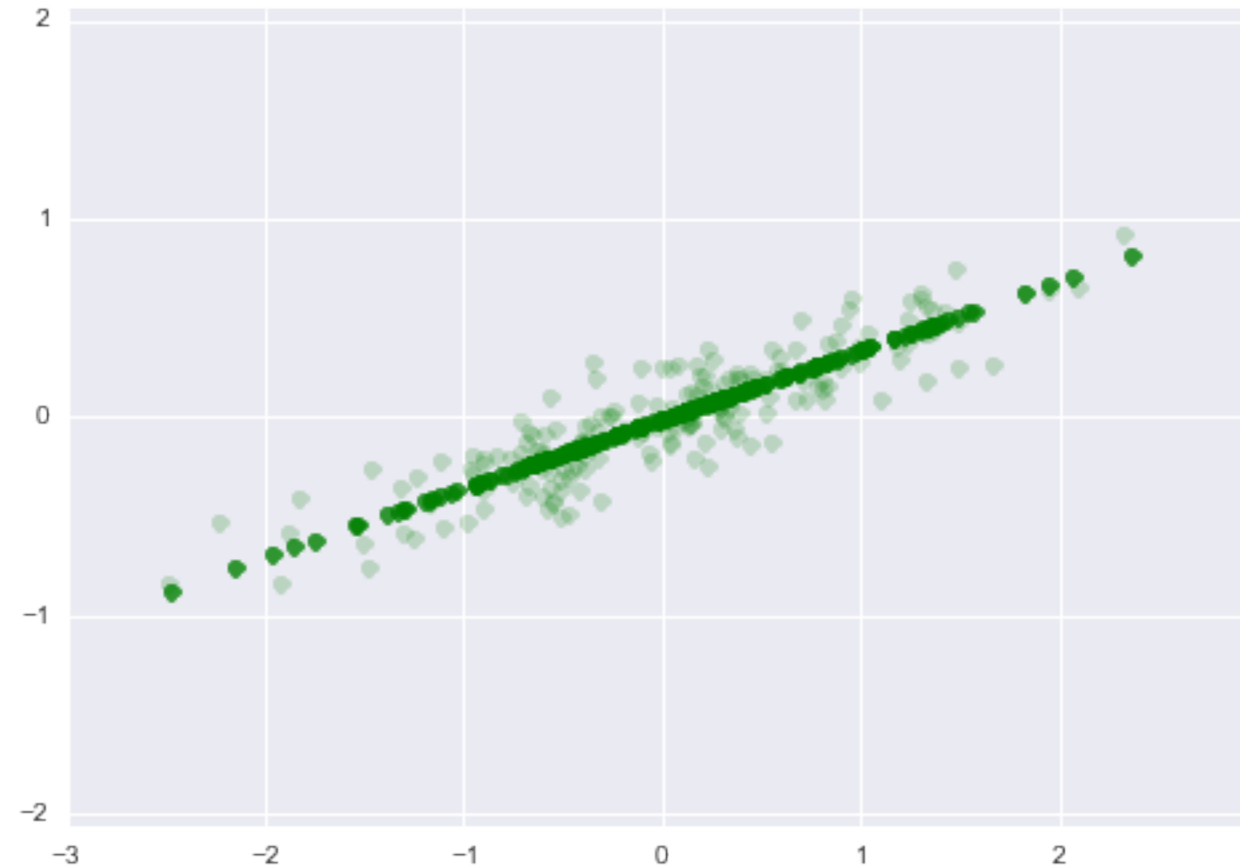
Minimal Spanning Tree Clustering (unsupervised)





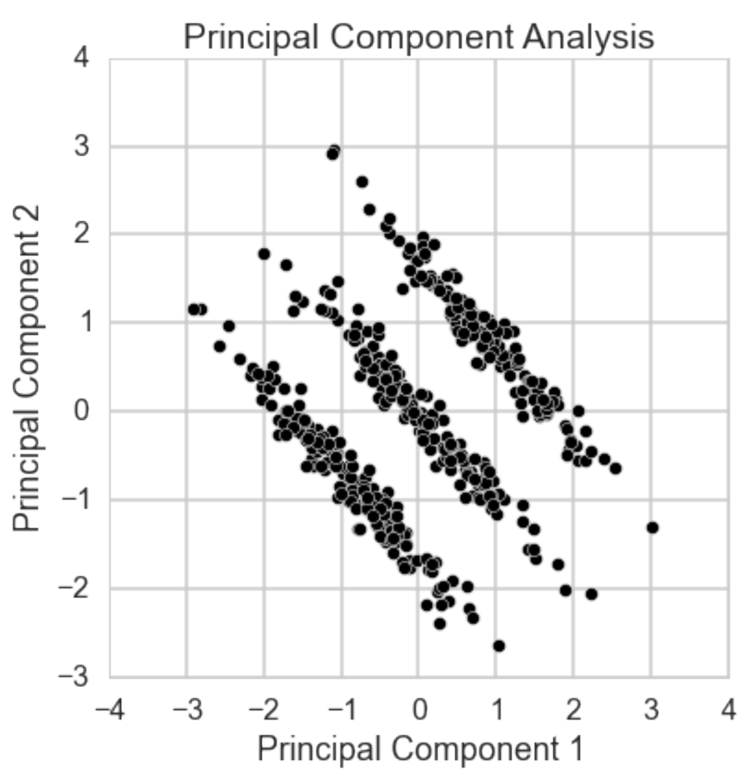
- MST is a subgraph that connects all nodes such that the sum of the graph edges is minimised.
- Chop edges larger than threshold and minimal number of points in a cluster

PCA (unsupervised)

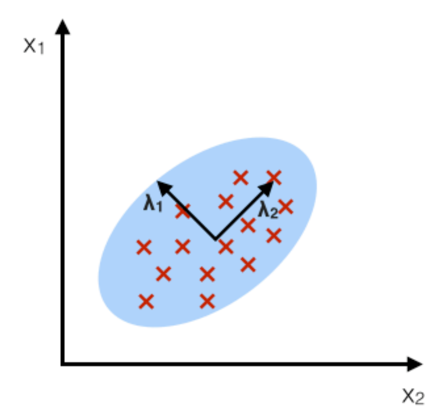


$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N}$$

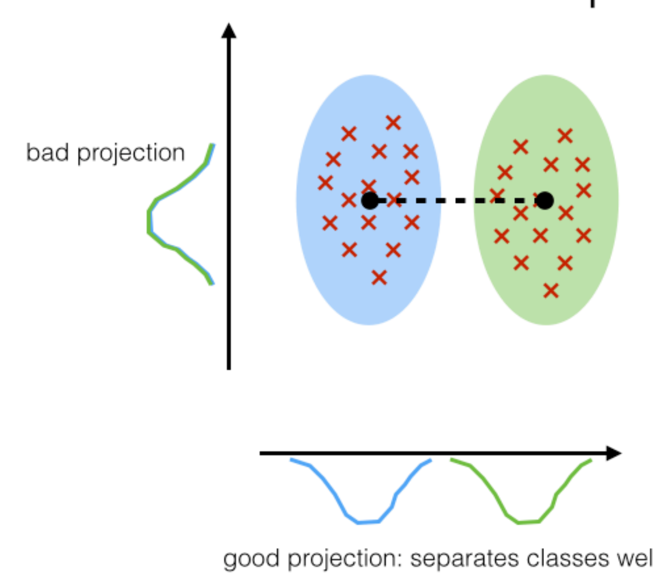
$$cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$



PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation

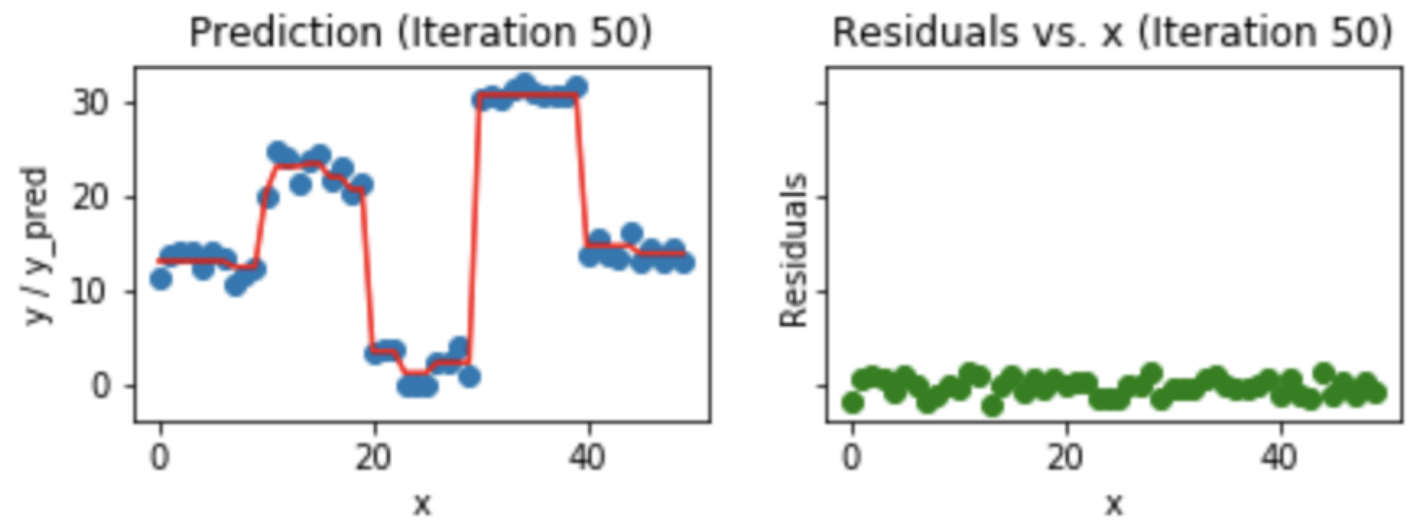


Next time: More advanced algorithms

- **Ensemble methods:**
  - Bagging (averaging lowers variance) "bootstrapping"
  - Voting (majority wins)
    (or train a simple linear regressor of the classifiers)
  - Boosting
- **Neural networks**

Top 5 (supervised)

Algorithm	Comments
Neural Networks	• Take long to train - lot of CPU • Overfits • Requires lot of data
Gradient Boosted Trees	• Fast • Overfit danger
Random Forest	• Robust to overfitting
SVM w/non-linear kernel	• Pretty good
Gaussian Processes	• non-parametric fitting



Scikit-learn

http://scikit-learn.org

Pros:

- ★ Written in Python, language #1 in astronomy today
- ★ Even complicated powerful algorithms are provided
- ★ Single and relatively simple API for all tasks
- ★ Actively being developed, open source, free
- ★ Object oriented, extensible
- ★ Great and practical online documentation with tons of examples

Cons:

- ◉ Not suitable for big data (but it can be tweaked with *dask*, *partial_fit* method, moreover, many other software packages follows a similar API concept)

Really Simple API

0) Import your model class

```
from sklearn.svm import LinearSVC
```

1) Instantiate an object and set the parameters

```
svm = LinearSVC()
```

2) Fit the model

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

3) Apply / evaluate

```
print(svm.predict(X_train))
print(y_train)
```

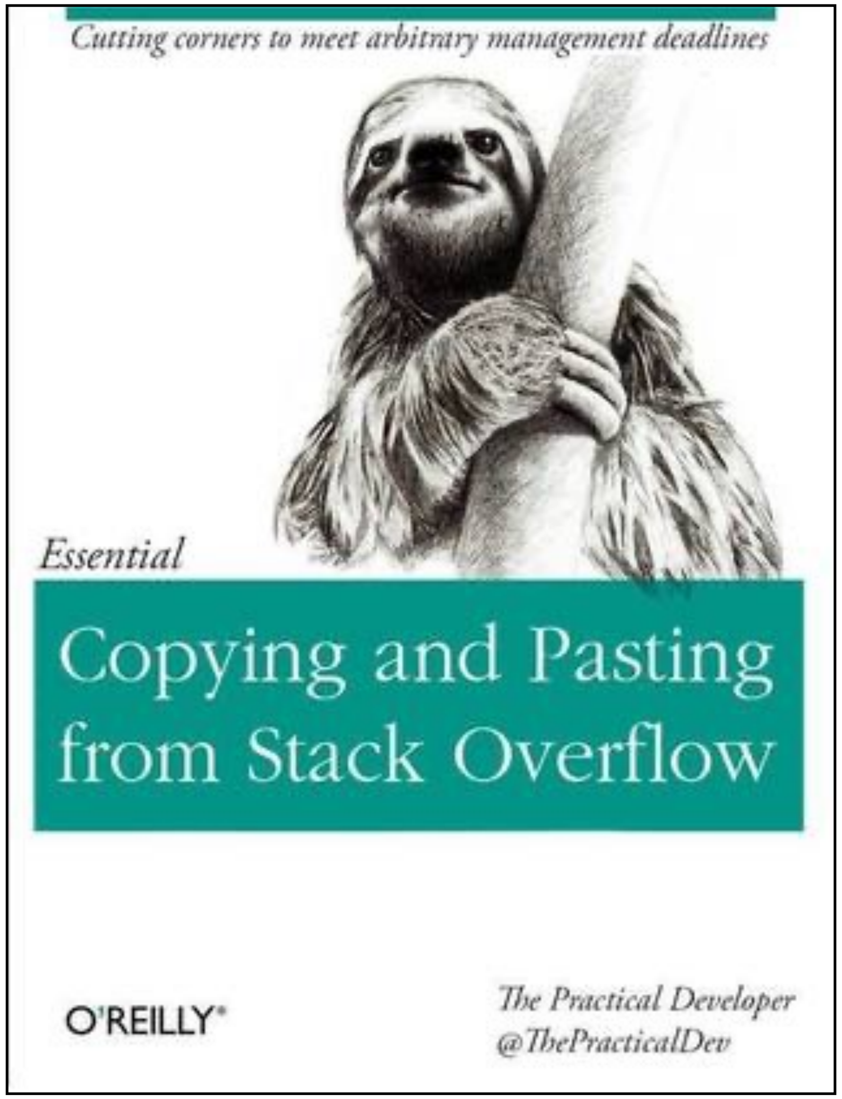
- hyperparameters specifying the model in the family of models
- feature vectors **X**

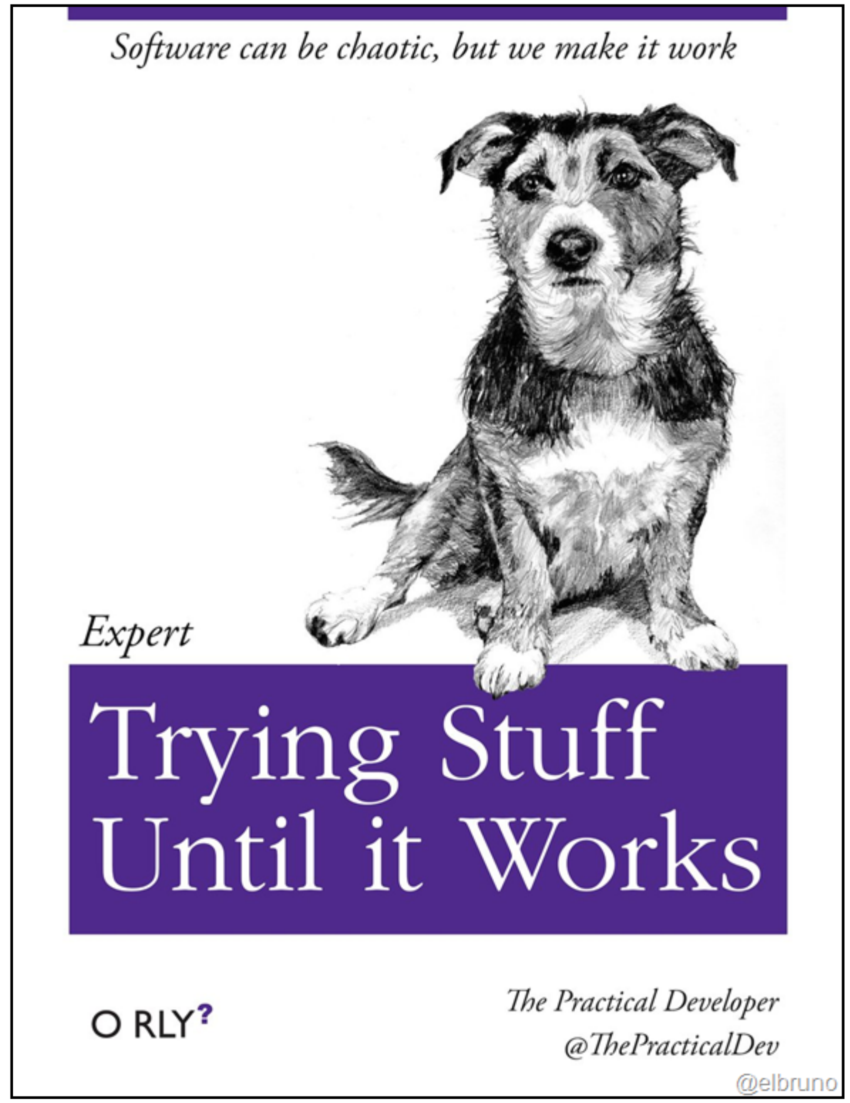
  N rows = number of points
  m columns = number of features
- values/labels **y**

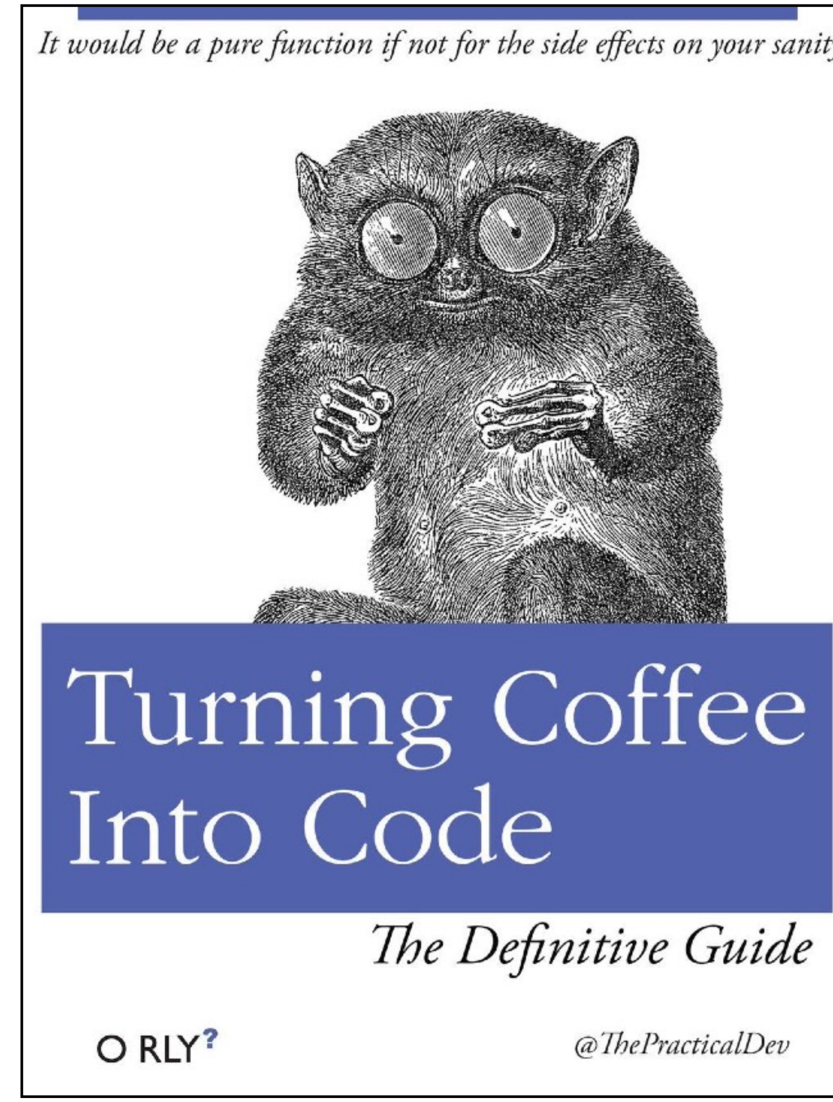
```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(n_estimators=100)
```

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







- Yaser Abu-Mostafa (Caltech): *Learning From Data*, 2012
- Željko Ivezič, Andrew Connolly, Jake Vanderplas: *Statistics, Data mining and Machine Learning in Astronomy*, 2014
- Aurelien Geron: *Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow*, 2019
- Francois Chollet: *Deep Learning with Python*, 2017
- https://www.deeplearningbook.org/

Further Reading/Watching

- scikit-learn.org documentation, great gallery, tons of examples

- Andrew Ng Standford course (I passed it and it’s really good, the code is in Matlab)
  https://www.coursera.org/learn/machine-learning

  `http://cs229.stanford.edu/syllabus-autumn2018.html`
  `http://cs231n.stanford.edu/`

- Yaser Abu-Mostafa Caltech course — #1 course online (in my humble opinion, I passed it, it requires no coding, very good but simple math explanation)
  https://work.caltech.edu/telecourse.html

- MIT 6.S191 Introduction to Deep Learning
  `http://introtodeeplearning.com`