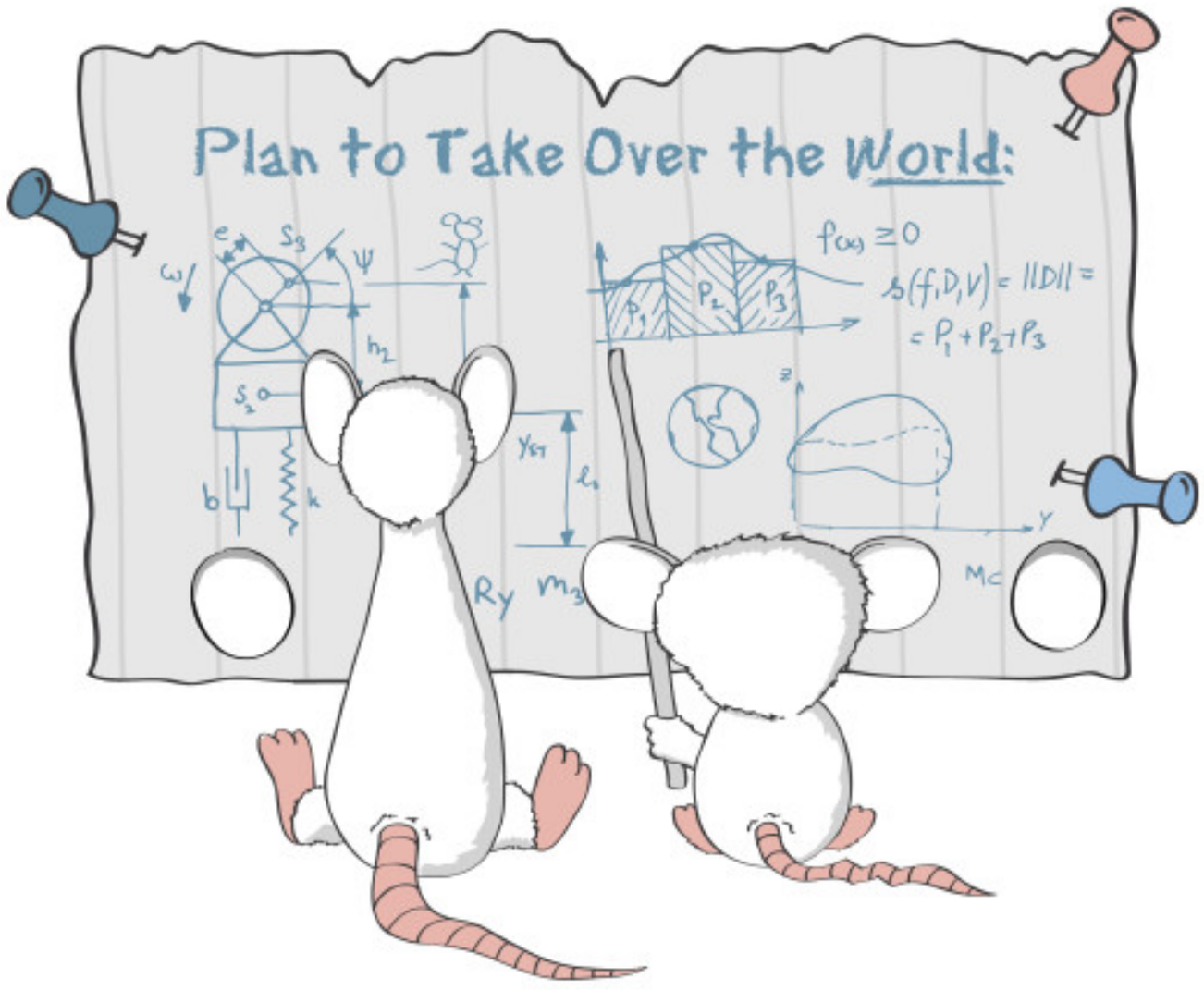


Hello to *scikit-learn*

DIAS ML course by Martin Topinka

https://github.com/toastmaker/ml-dias

Outline

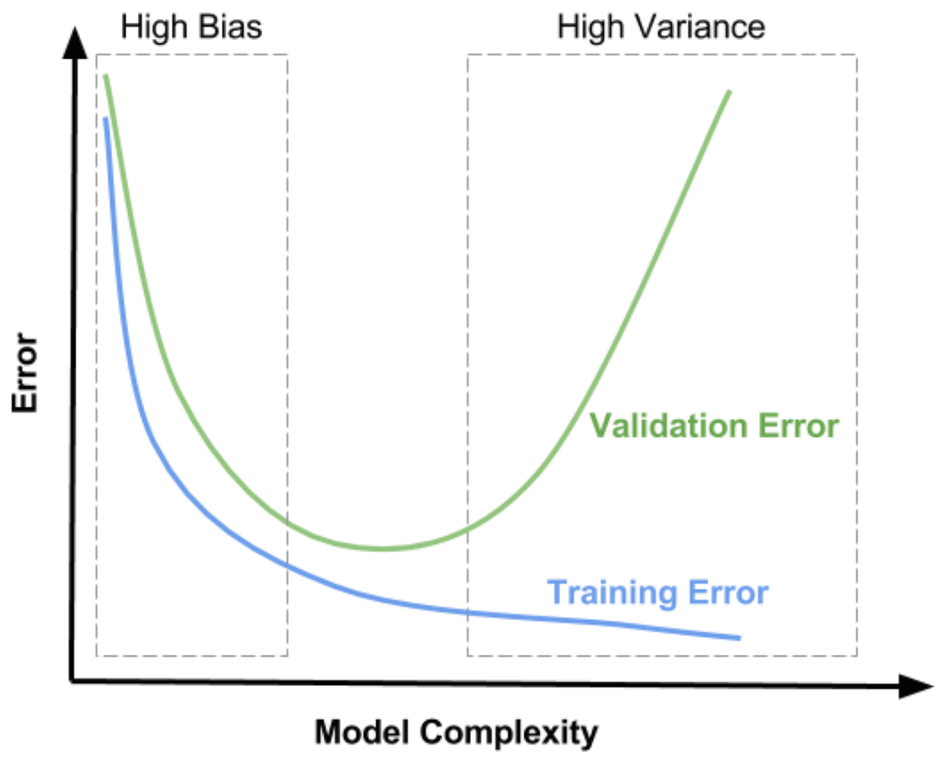
- Basic algorithms
- More advanced algorithms
  - Introduction to scikit-learn
  - Galaxy/Star classification (with PCA, feature selection, dim reduction)
  - Regression Photometric redshift with Random Forests, Gradient Boosted Trees

Recap

- Learning is possible (in a probabilistic sense) if 1) the pattern exists and 2) we have enough data points and 3) we are not unlucky = our training sample is representative

- What we want is low out of sample error = generalisation for unseen data

- Beware overfitting —> Tune model performance with cross-validation



Bias-variance trade-off

Low bias: model learned data well
Low variance: model can generalise well

	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

Linear Regression

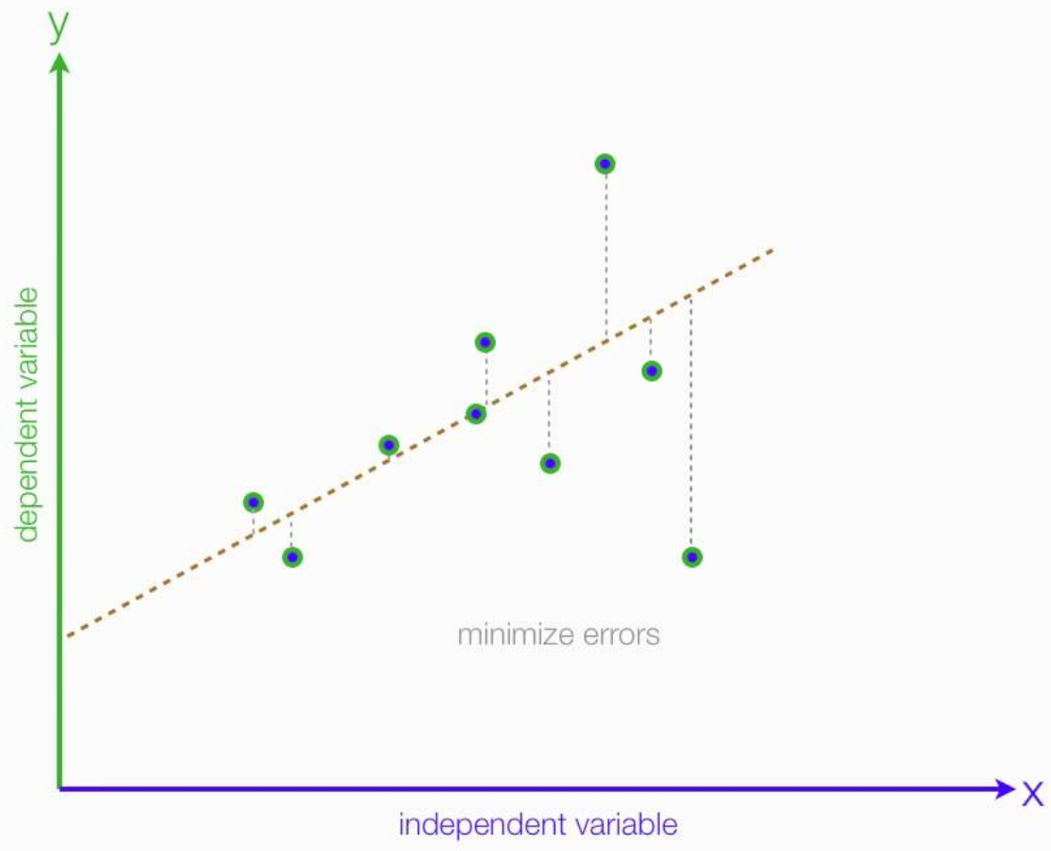
- Linear fitting in N-dim space
  $y = a\,x + b$, or more general $y = \sum a_i\, x^i$

- we have training pairs (x_i, y_i)

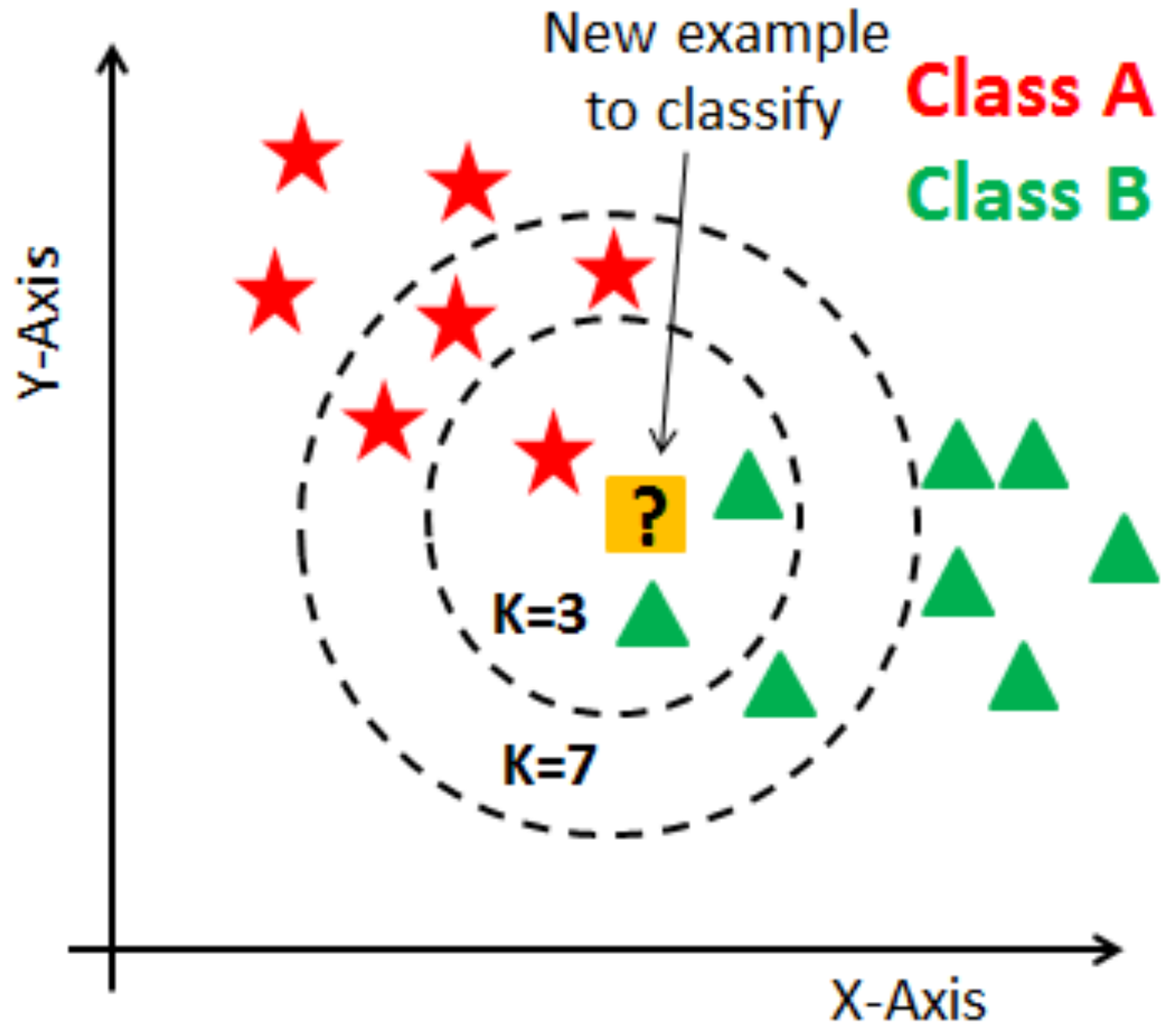
- Linear in weights! => works well with polynomials
  $a_0\, x^0 + a_1\, x^1 + a_2\, x^2 + a_3\, x^3 \ldots$

- We can transform our data points in a non-linear way and still use linear regression to find the fit

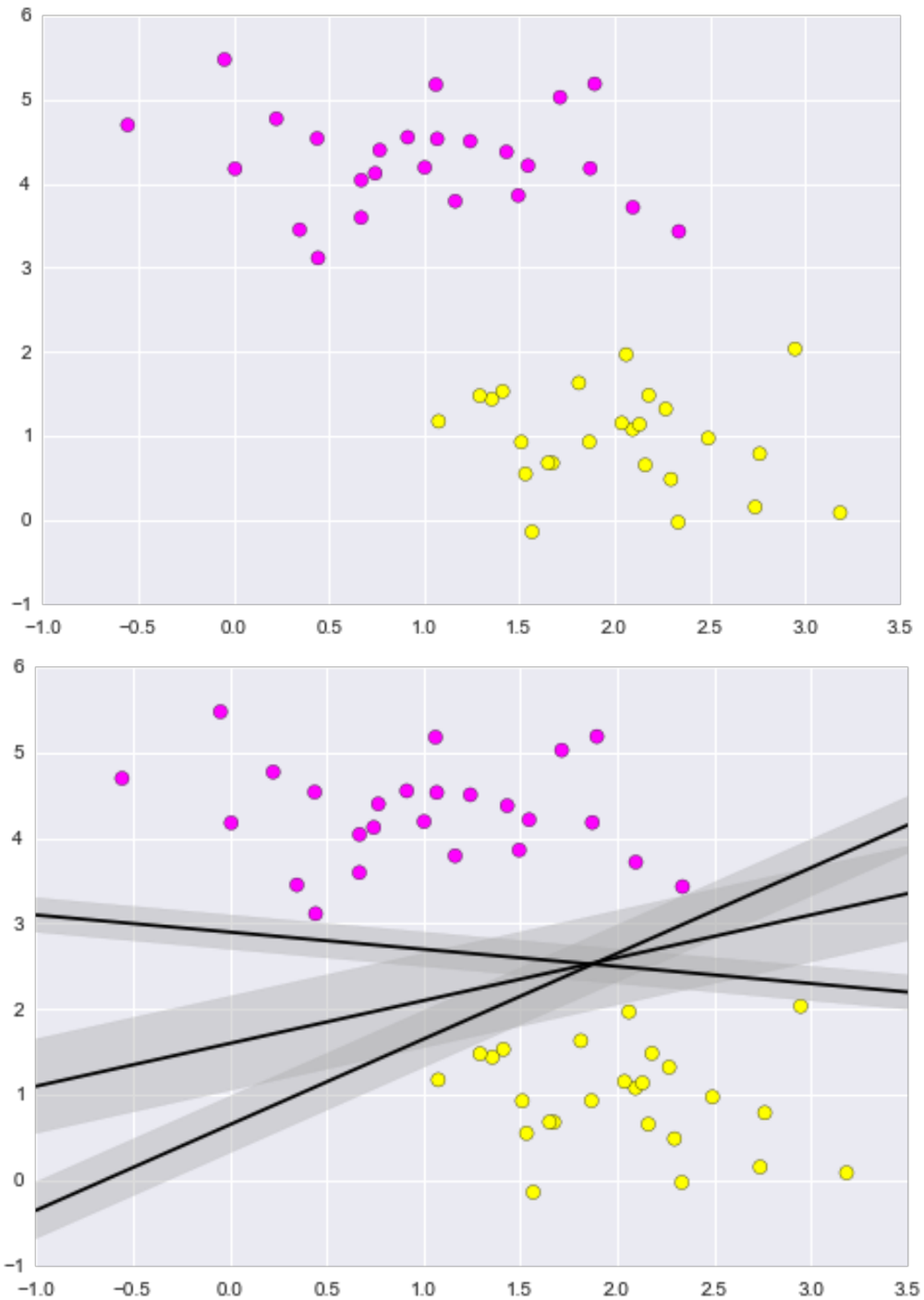
- We control the loss function *L1, L2*, …

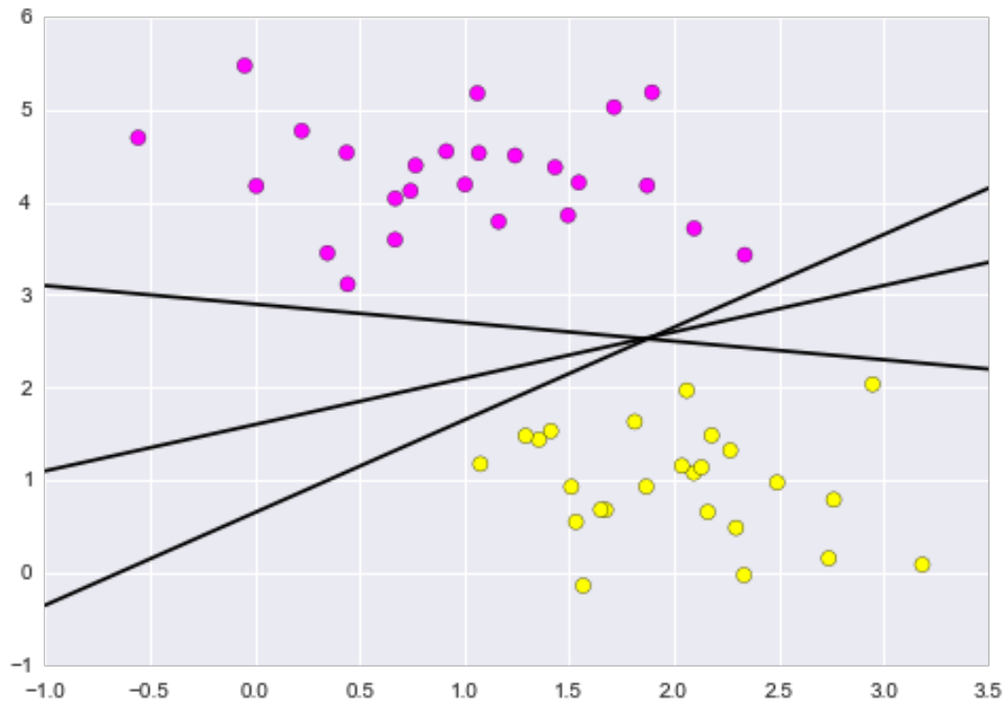


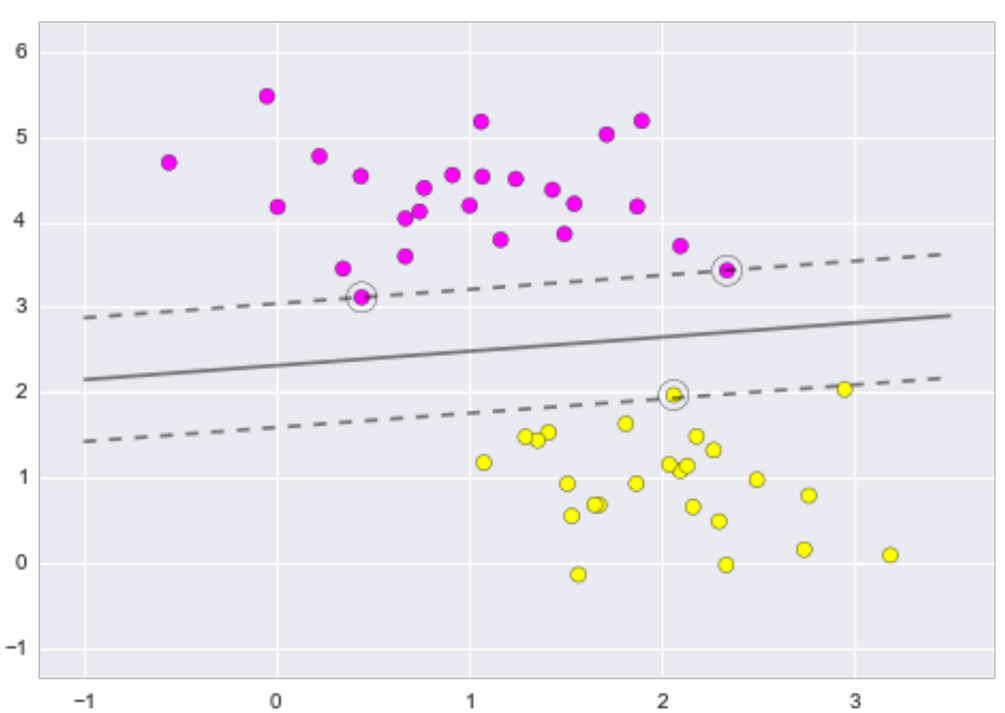
K Nearest Neighbours



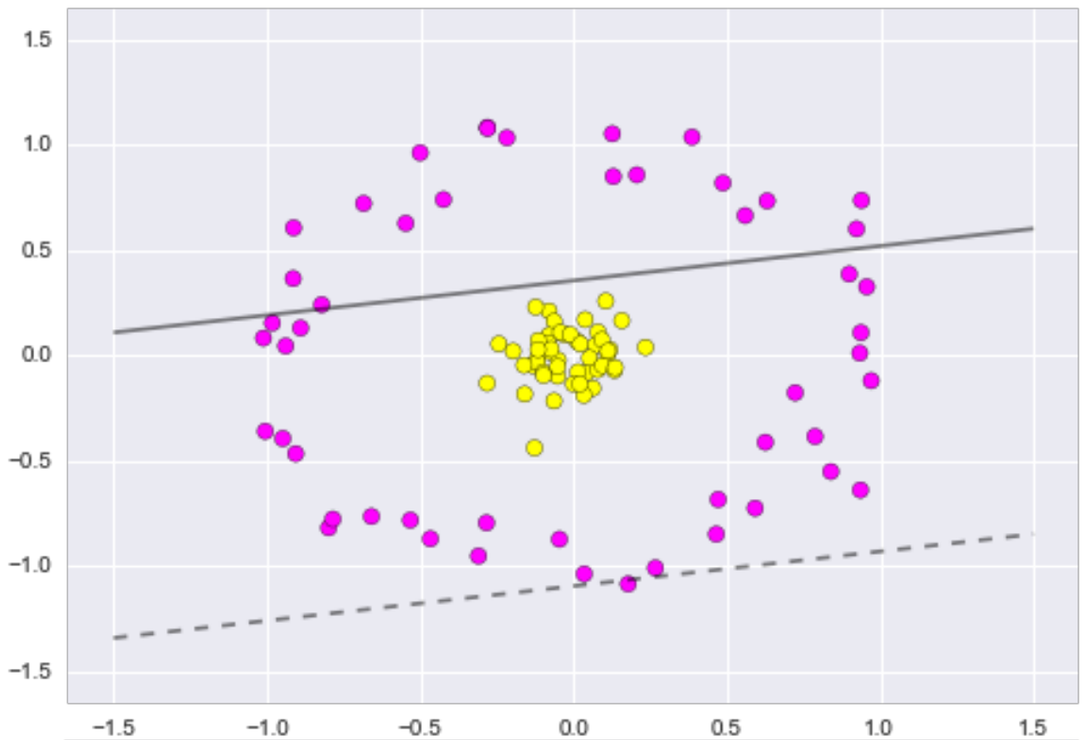
Support Vector Machine

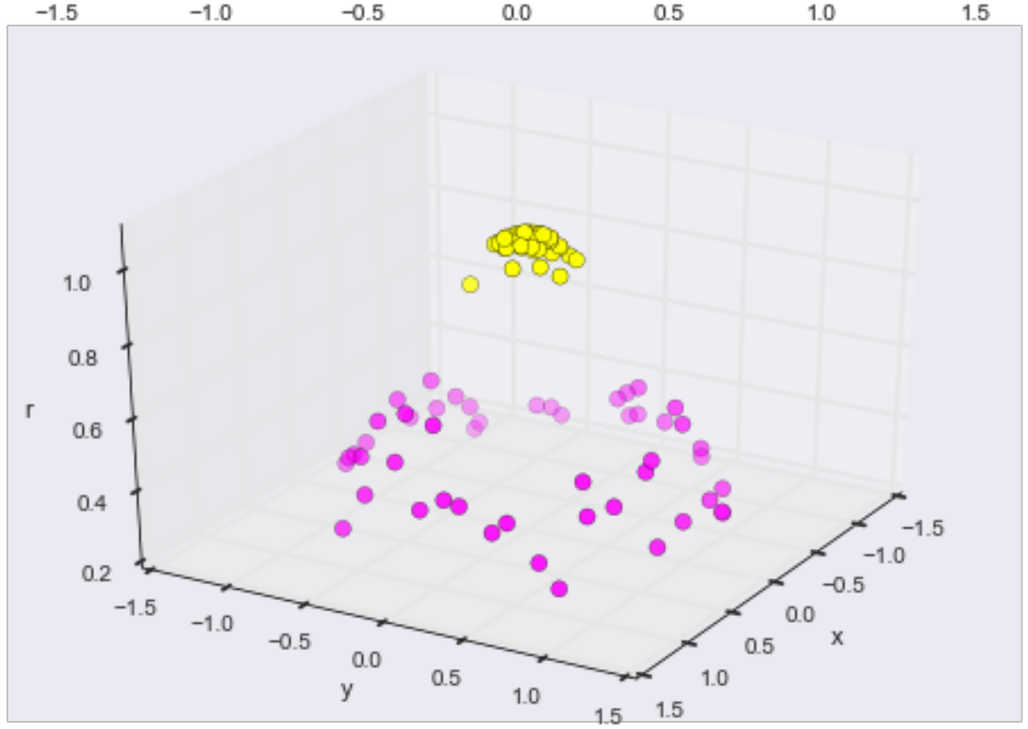


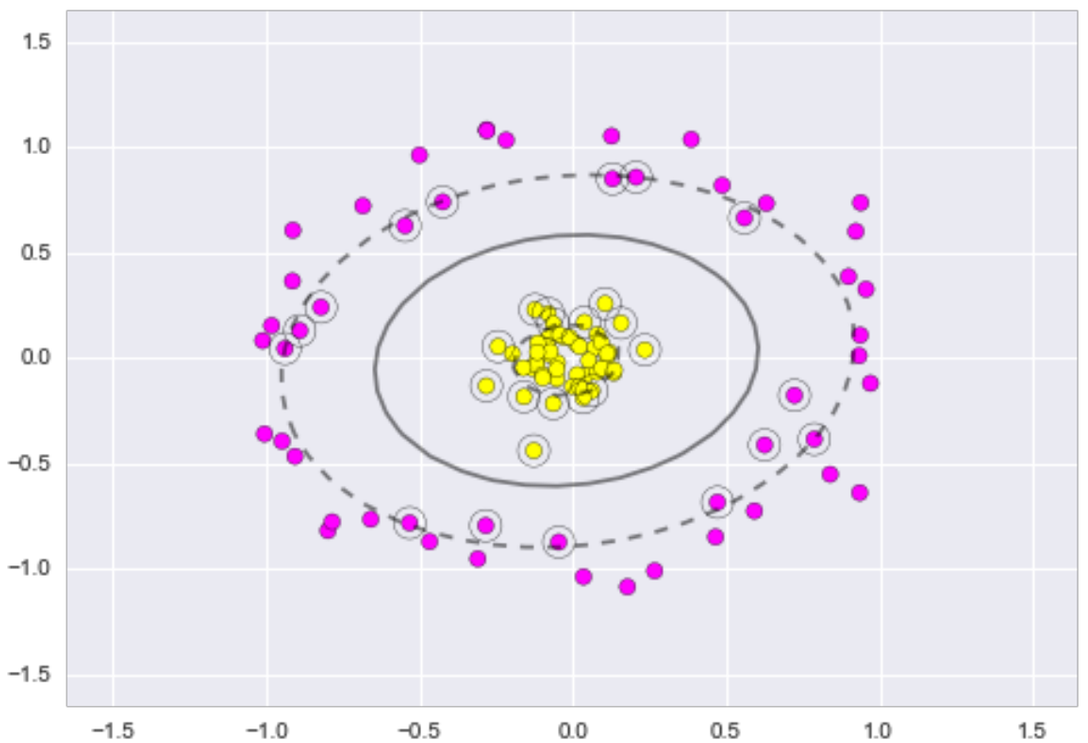




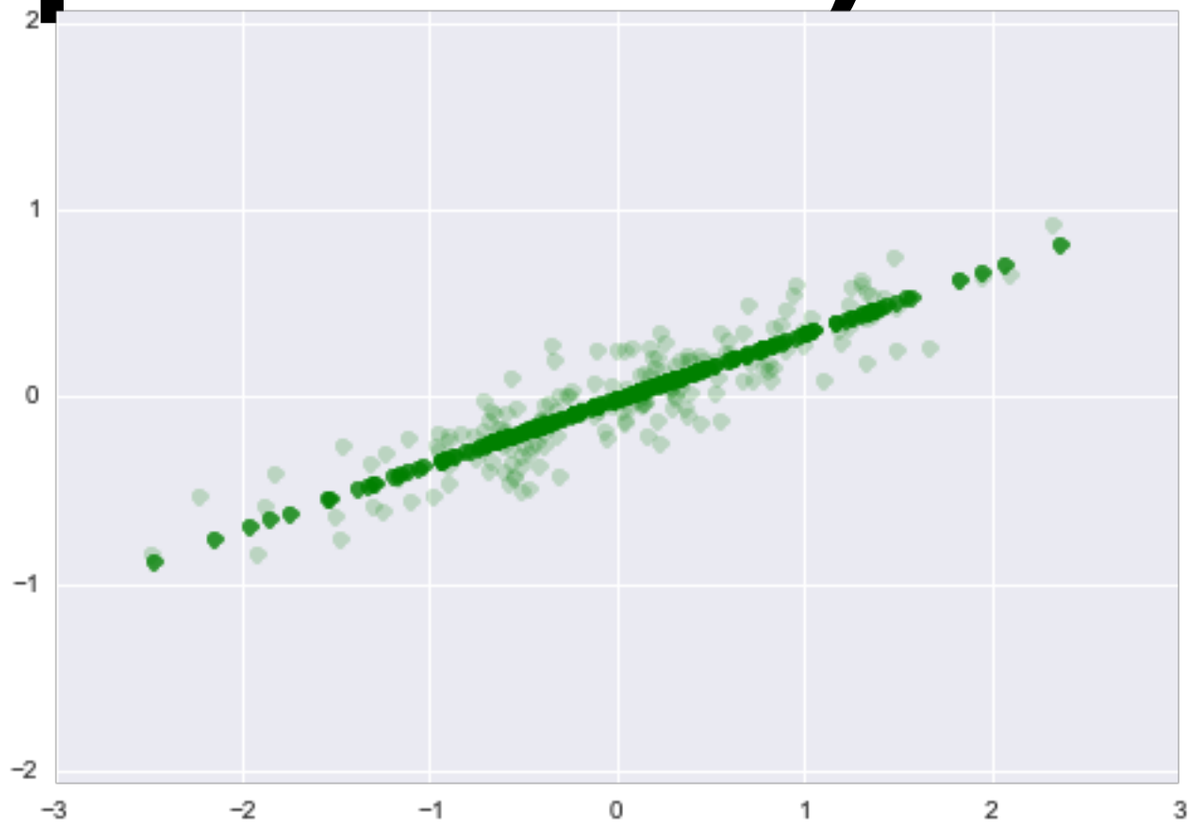
SVM with non-linear (kernel) transformation

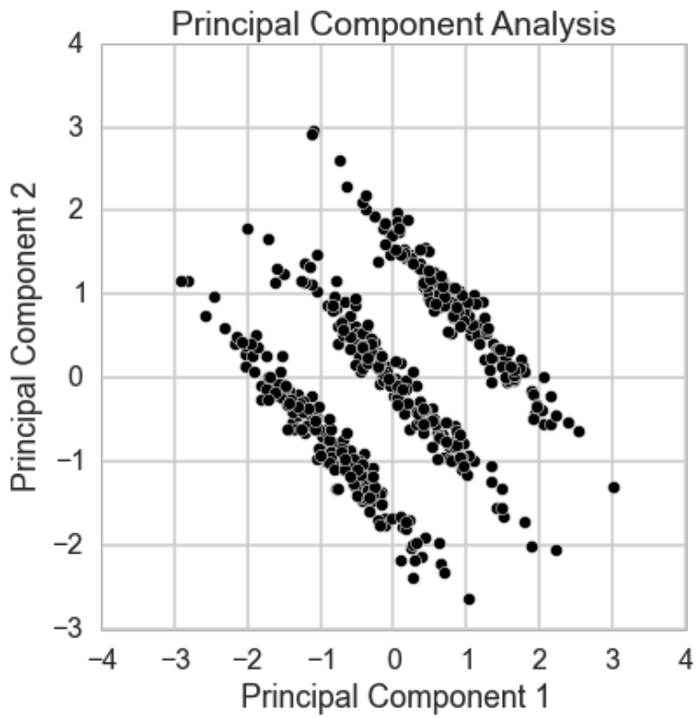




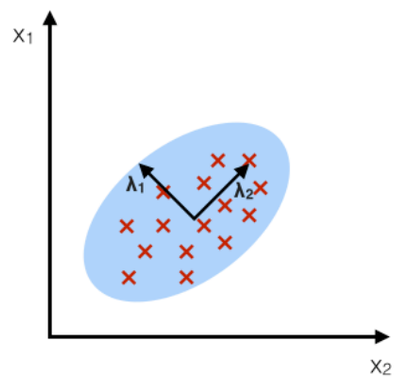


PCA (unsupervised)

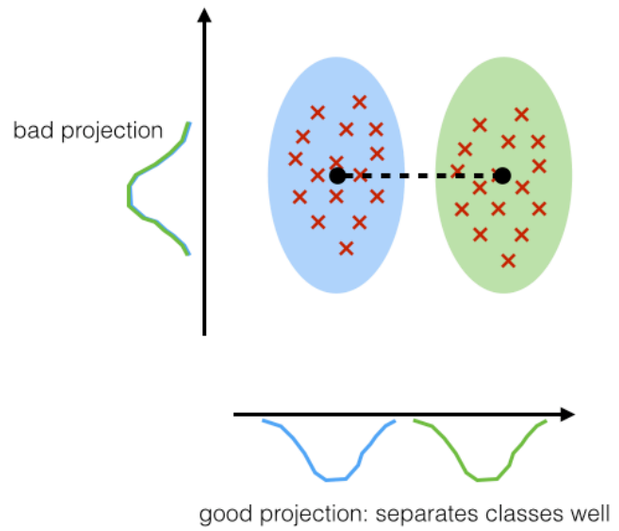




PCA:
component axes that maximize the variance



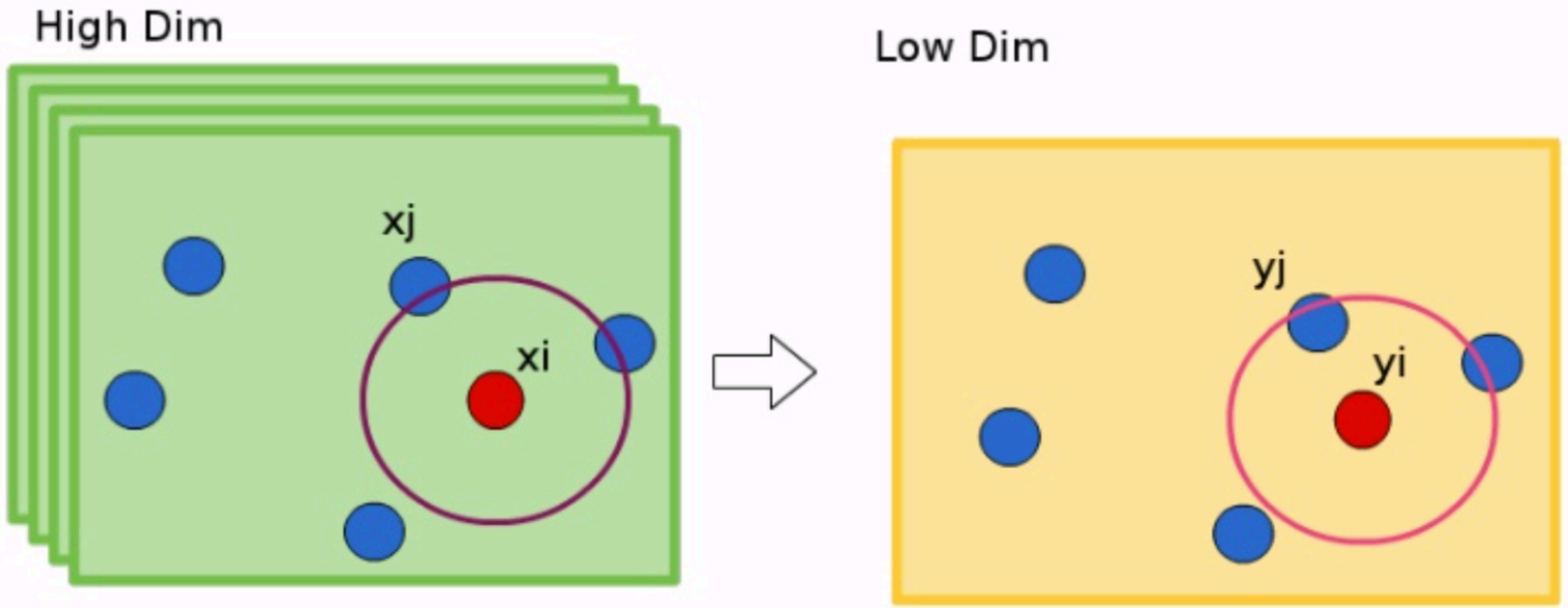
LDA:
maximizing the component axes for class-separation



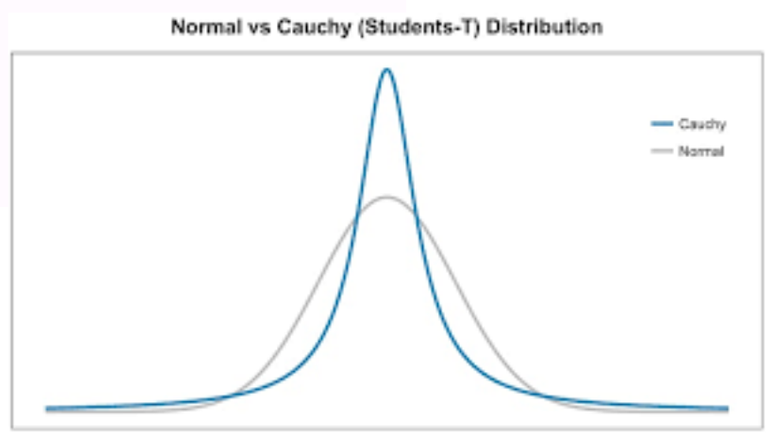
t-SNE (unsupervised)

t-distributed stochastic neighbour embedding

Measure pairwise similarities between high-dimensional and low-dimensonal objects



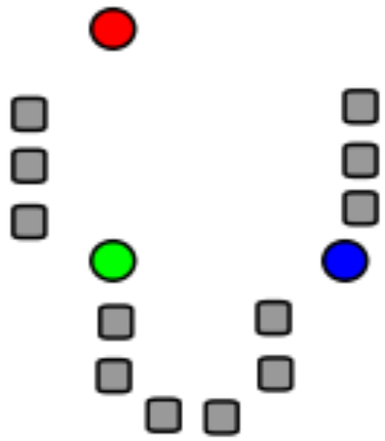
$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$



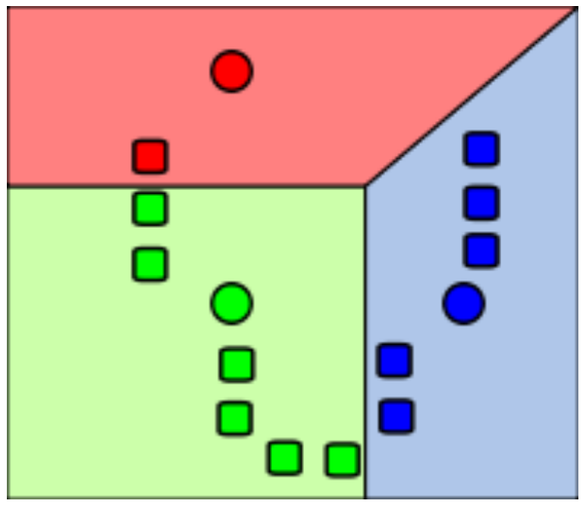
$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

K-Mean Clustering (unsupervised)

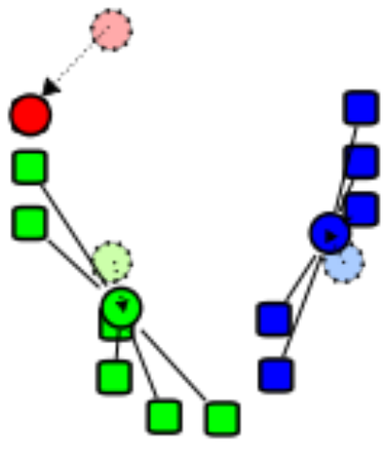
1)



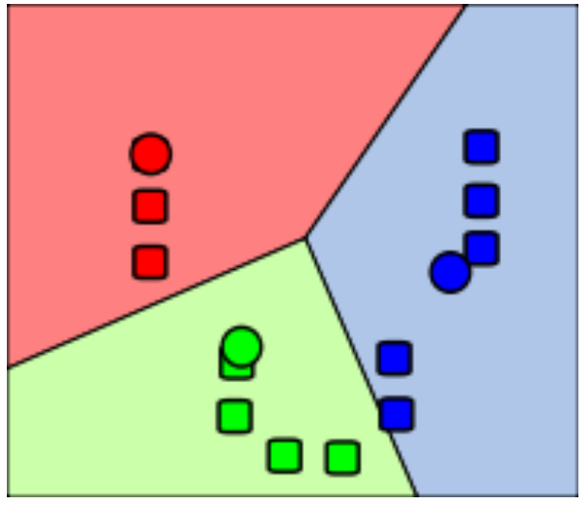
2)



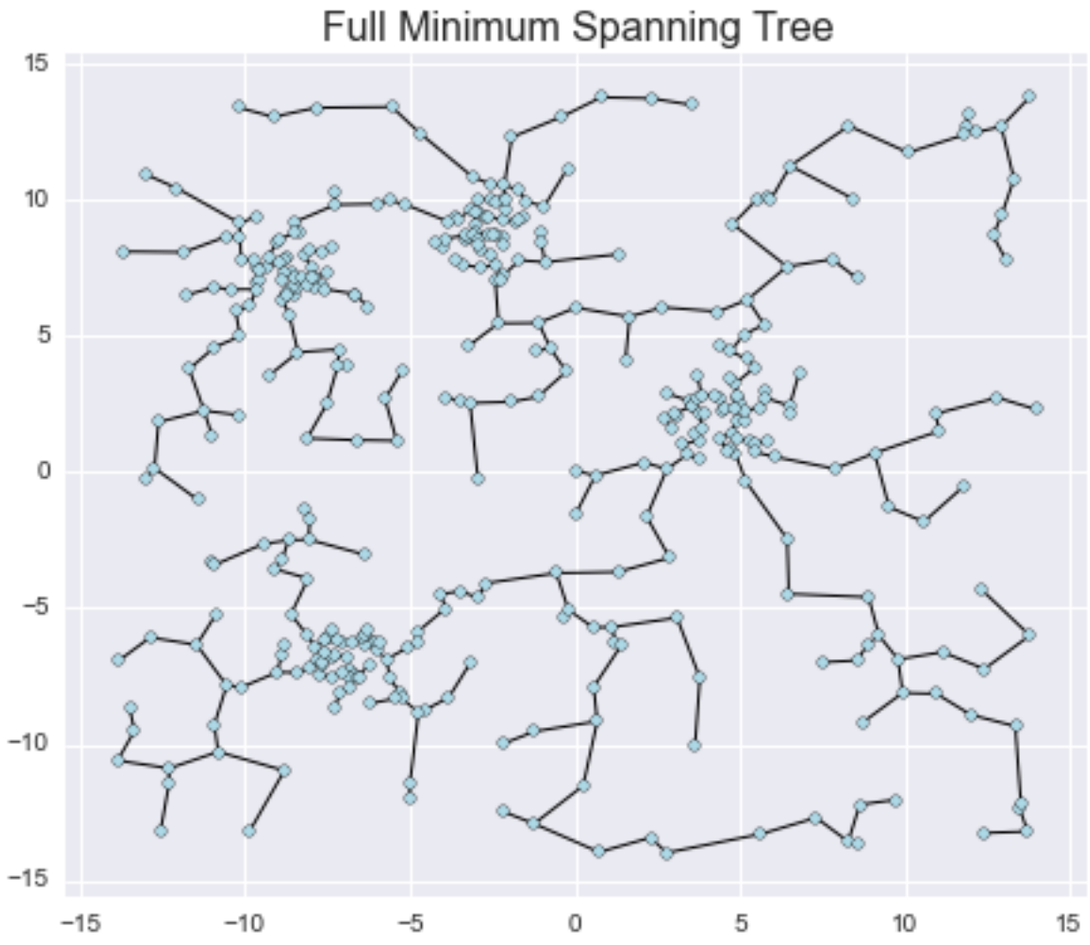
3)

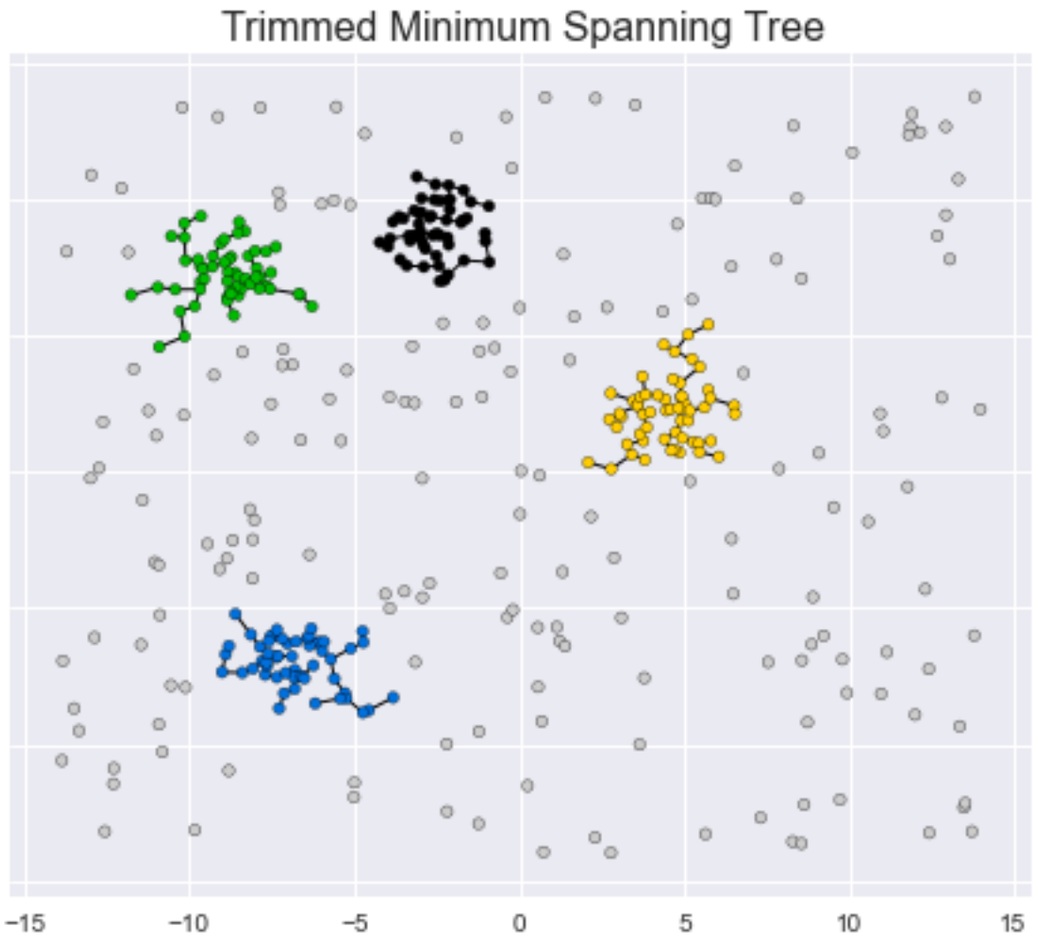


4)



Minimal Spanning Tree Clustering (unsupervised)



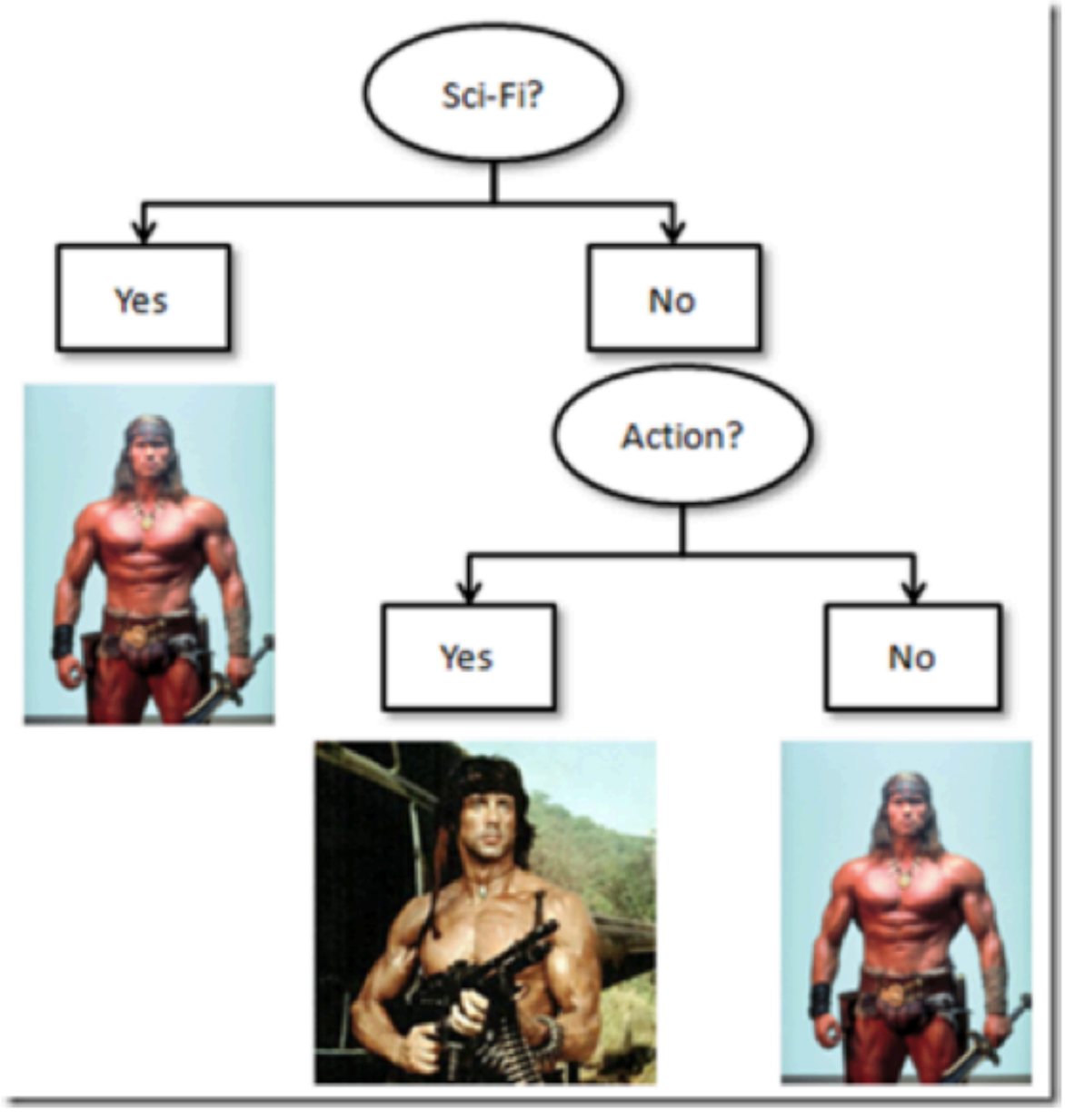


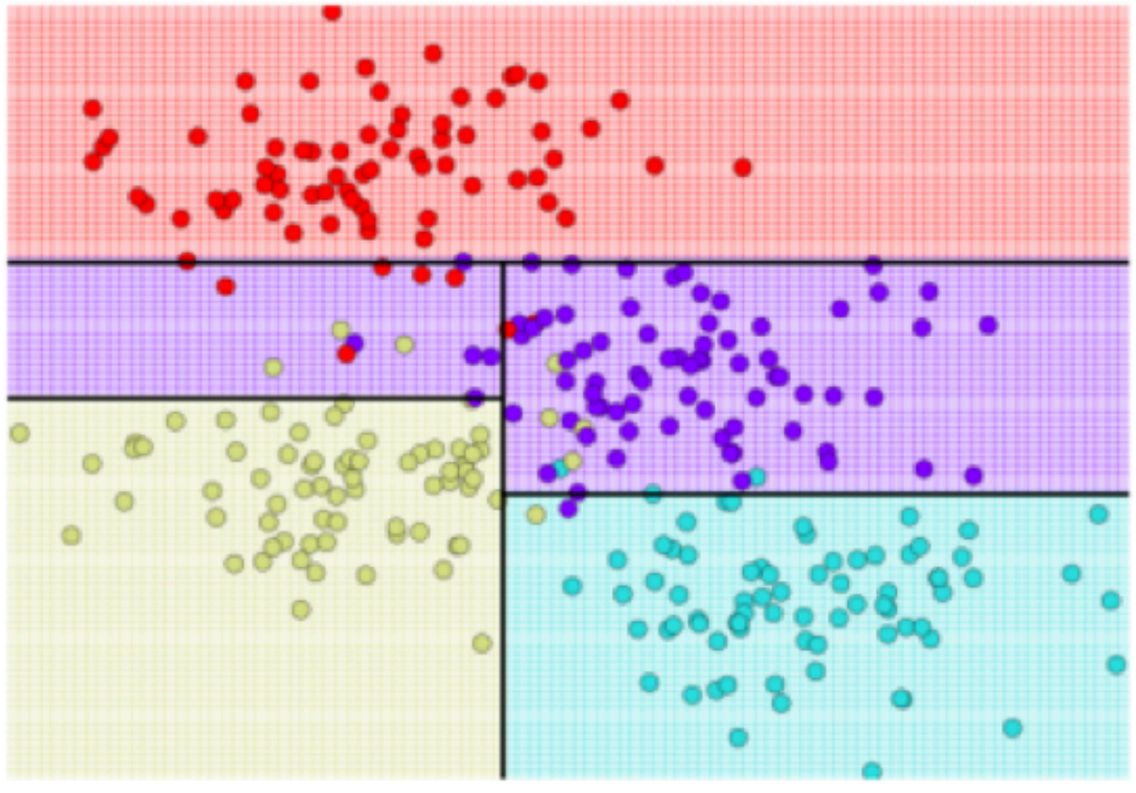
- MST is a subgraph that connects all nodes such that the sum of the graph edges is minimised.
- Chop edges larger than threshold and minimal number of points in a cluster

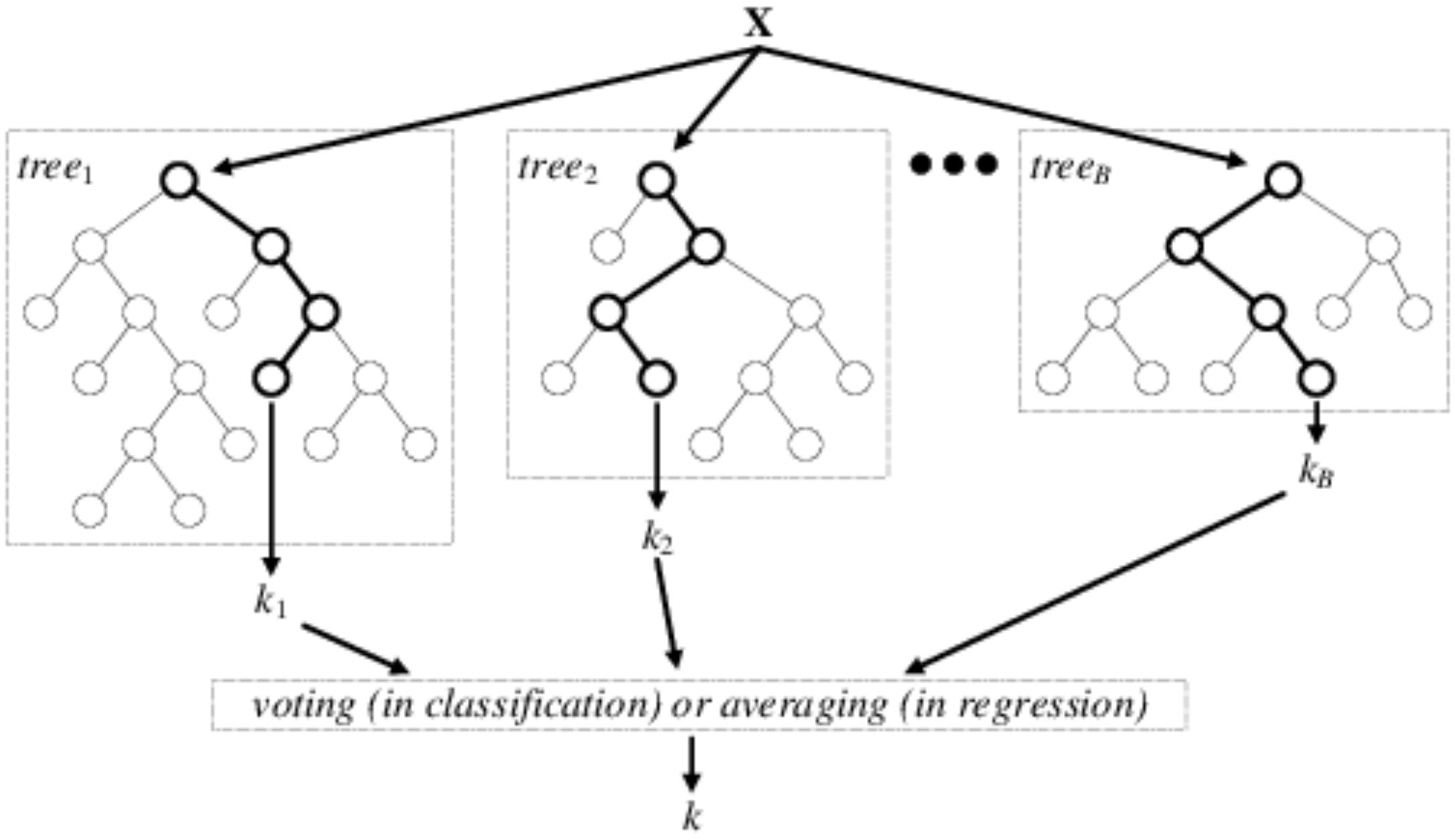
More advanced algorithms

- Ensemble methods:
  - Bagging (averaging lowers variance)
  - Voting
  - Boosting
- Neural networks

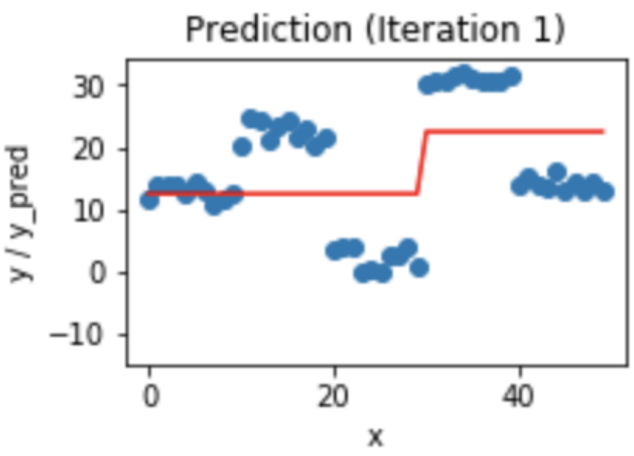
Random Forest (Ensemble of Decision Trees)

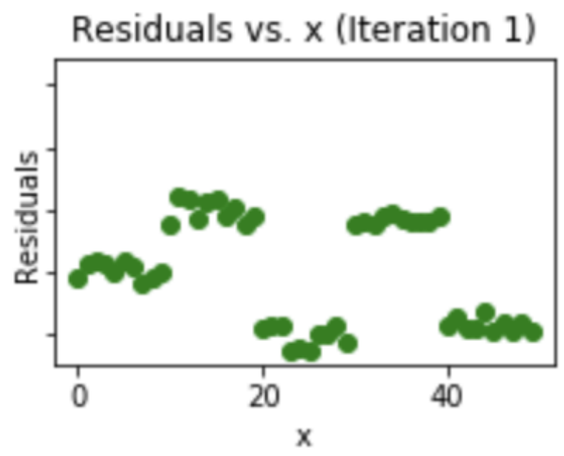


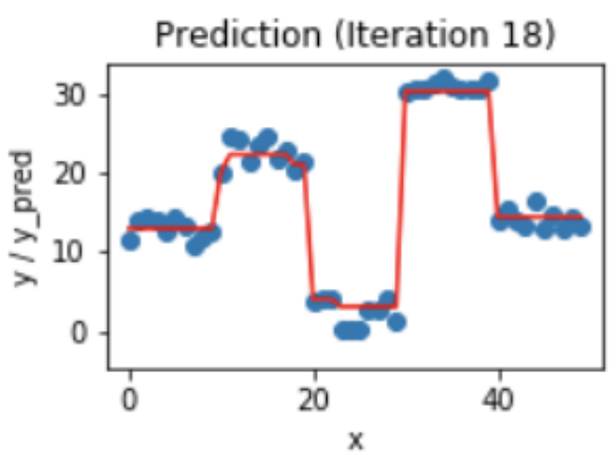


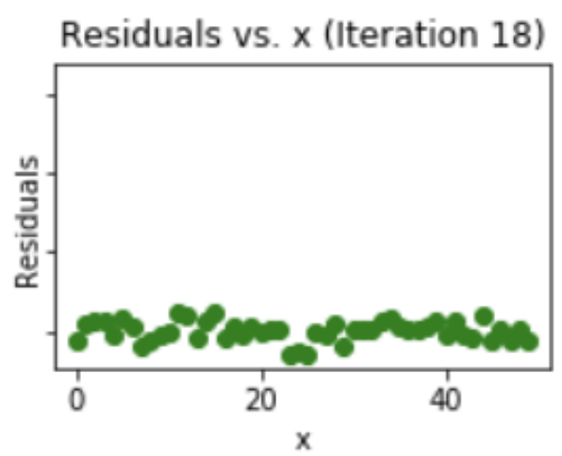

X
tree1
tree2
treeB
k1
k2
kB
voting (in classification) or averaging (in regression)
k

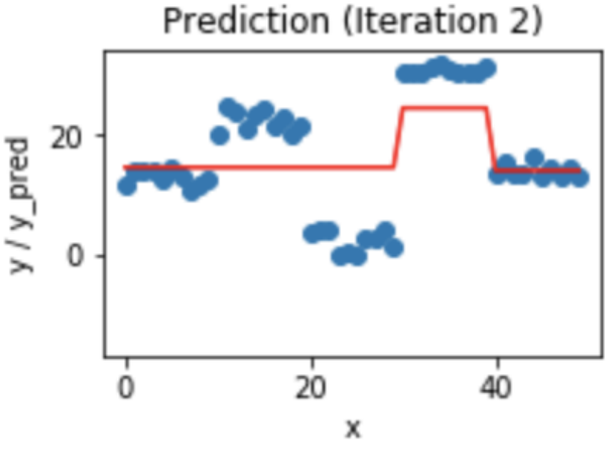
Gradient Boosted Trees

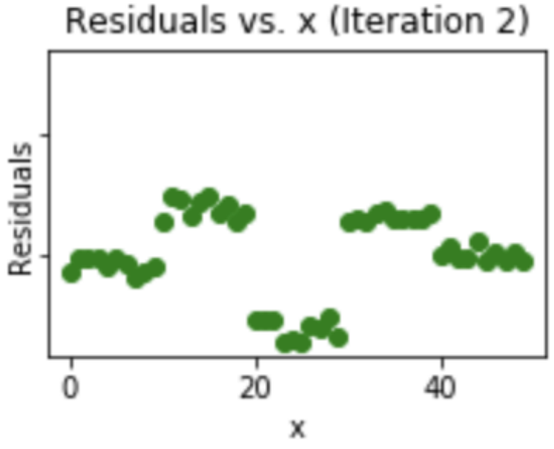


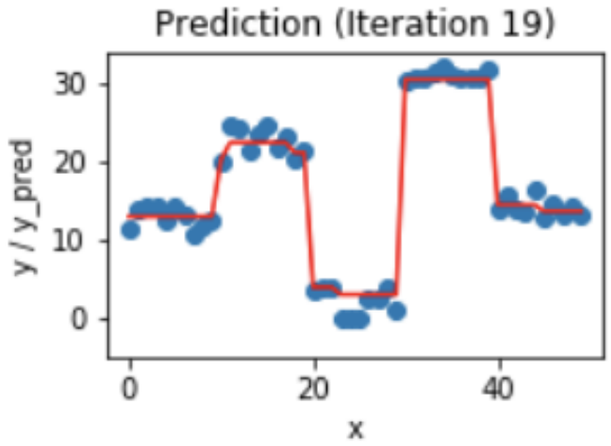


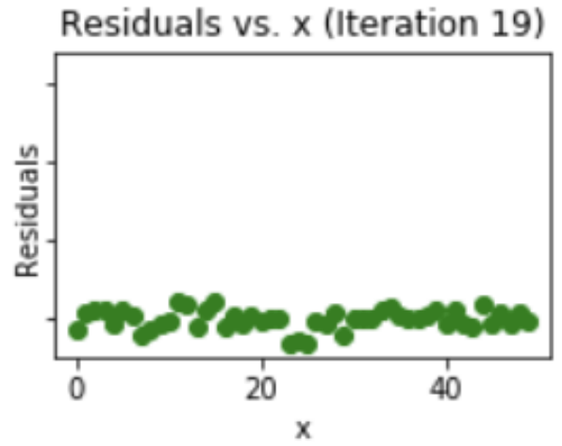


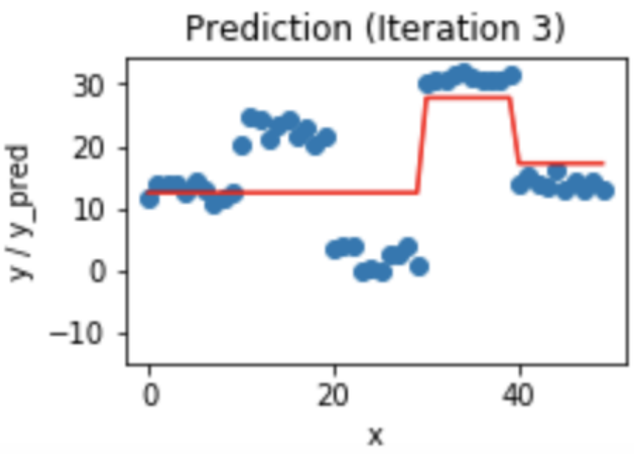


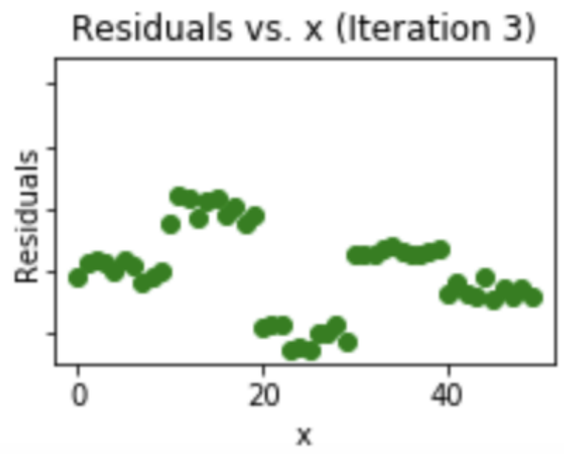


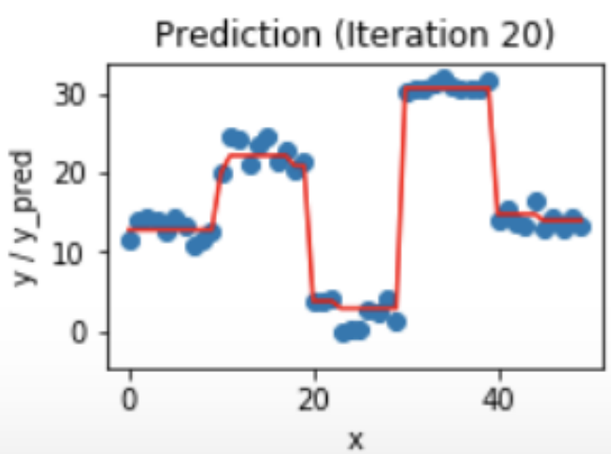


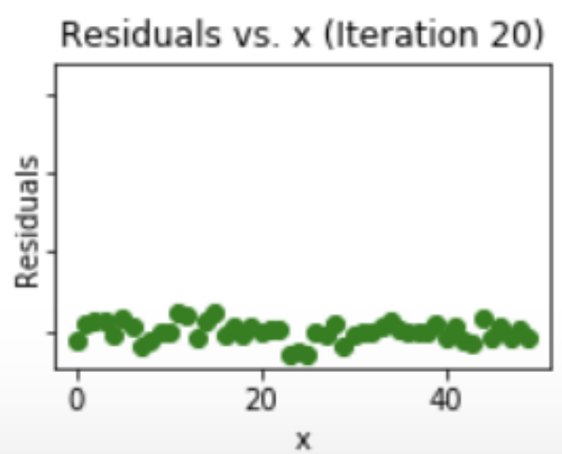




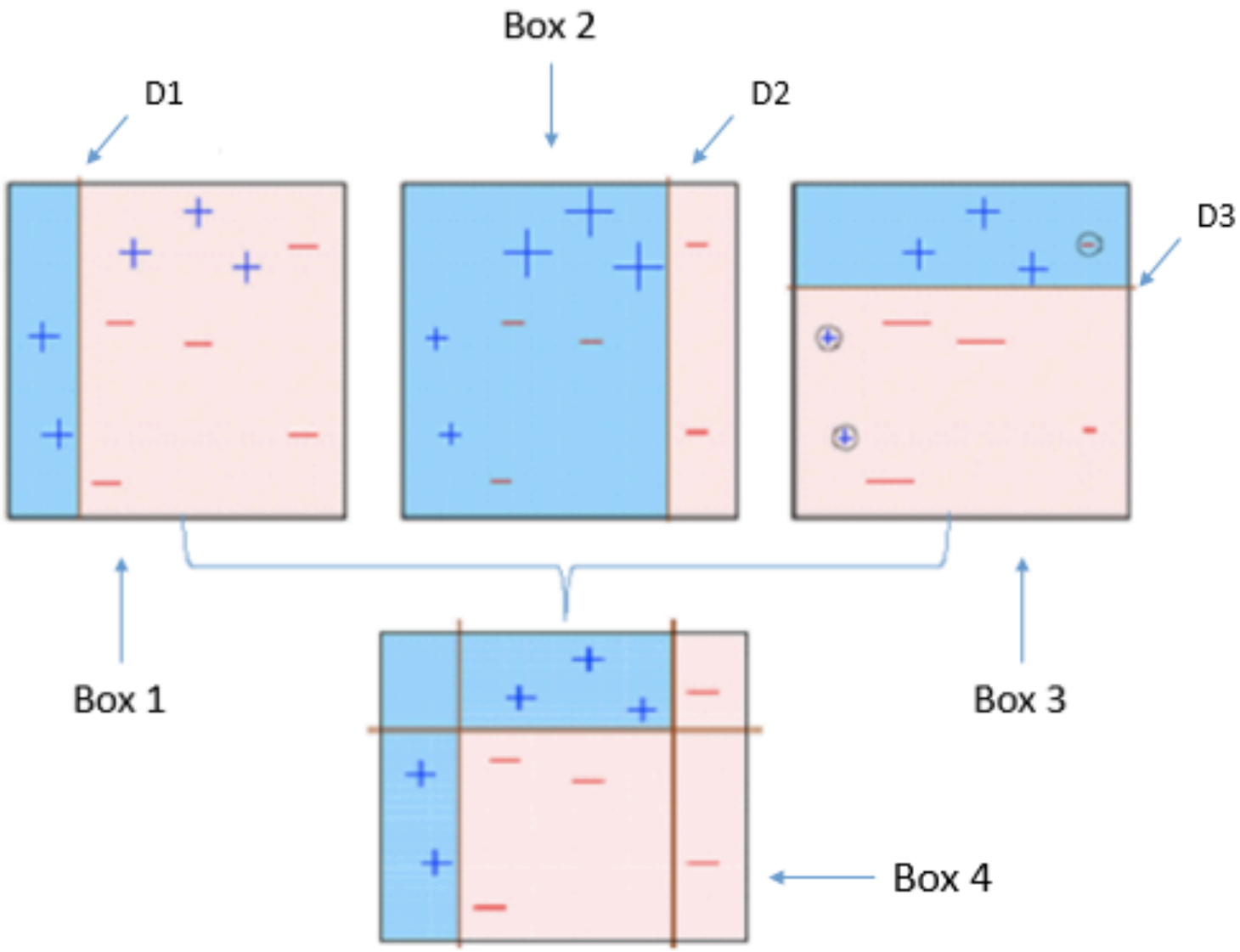




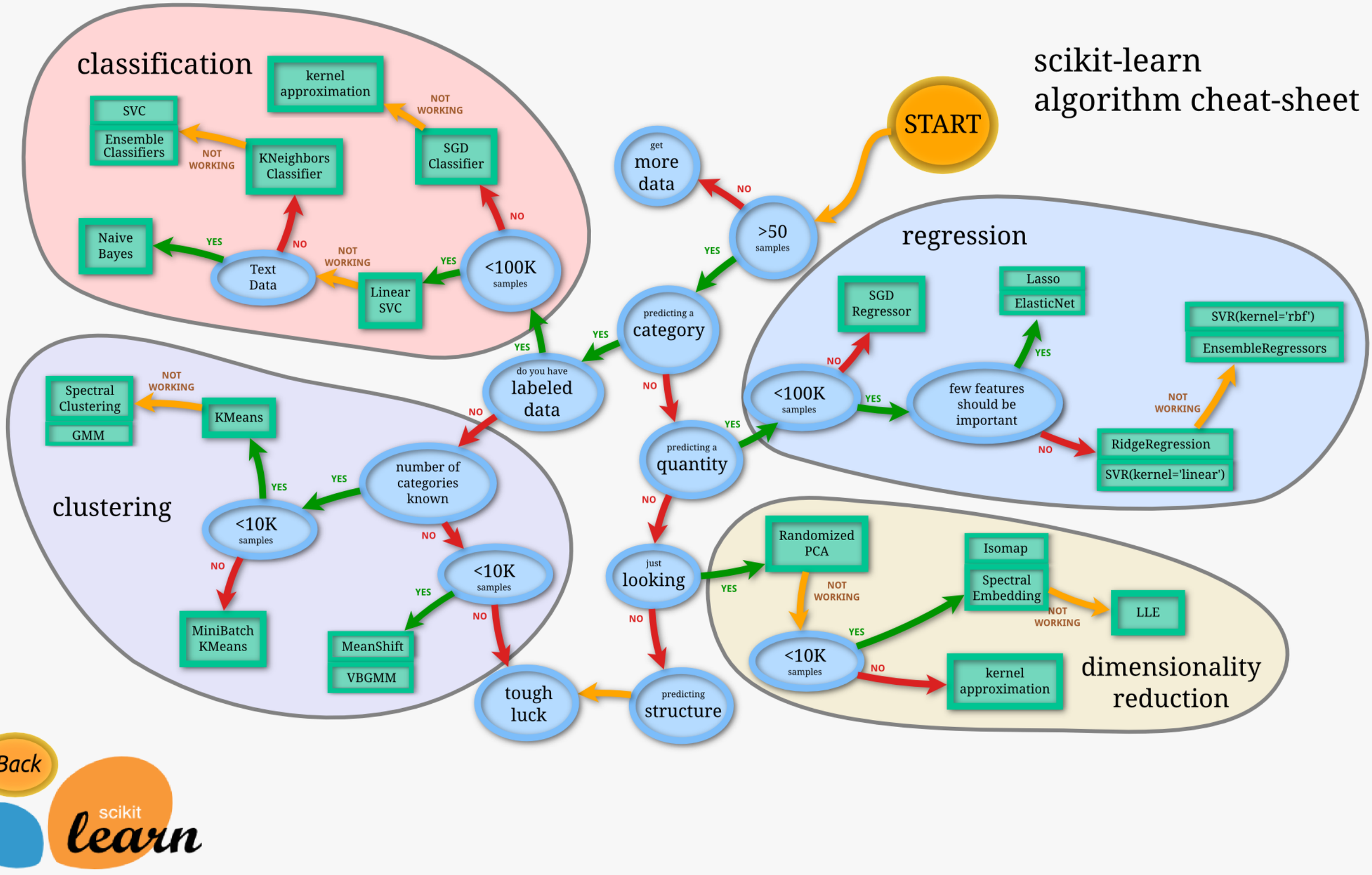




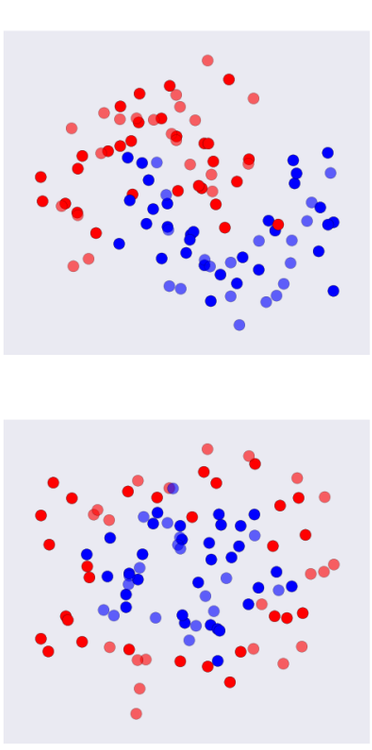
Voting (max/average) with weights (gradient descent)

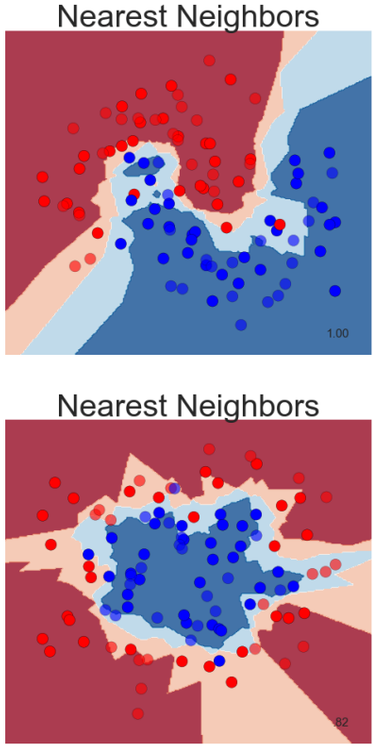


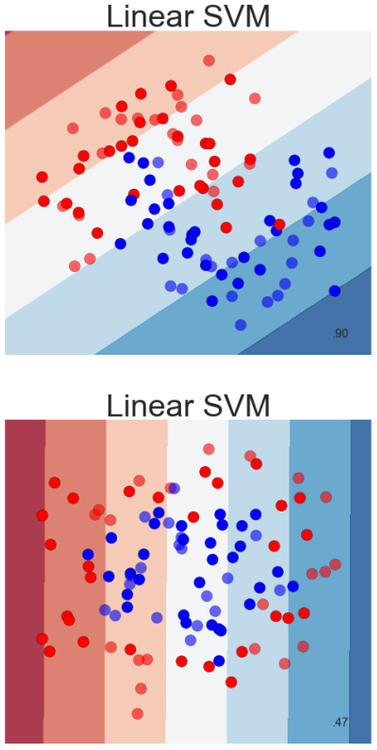
scikit-learn algorithm cheat-sheet

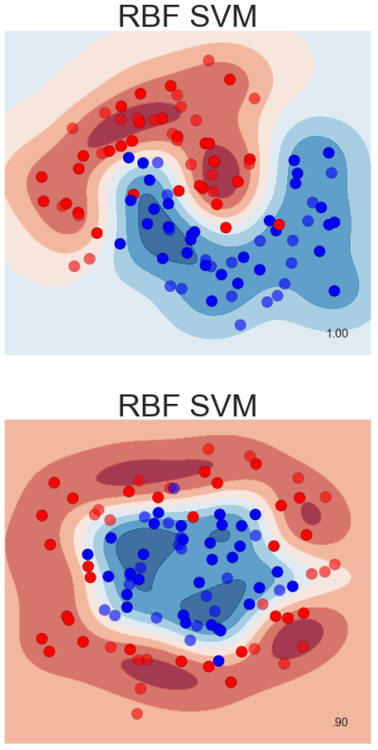


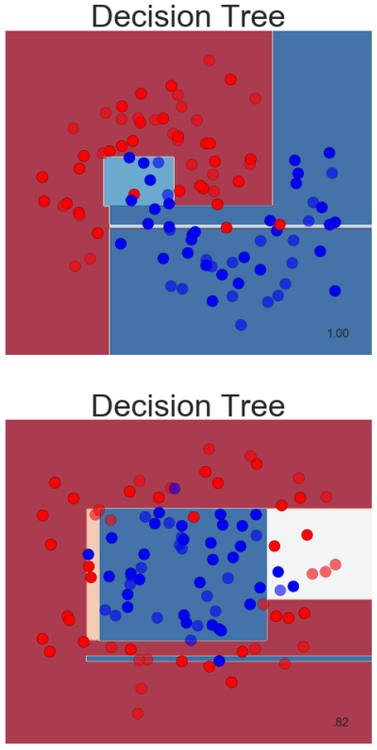
Back

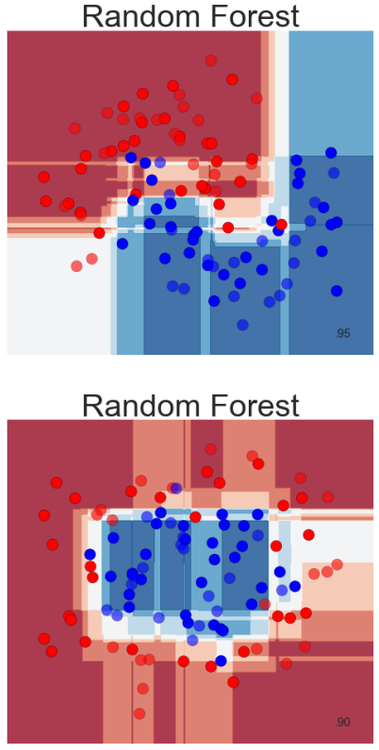



Nearest Neighbors
1.00


Linear SVM
.90


RBF SVM
1.00


Decision Tree
1.00


Random Forest
.95

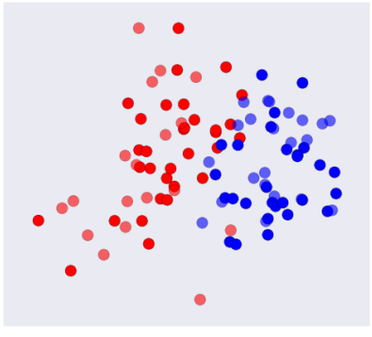
Nearest Neighbors
.82

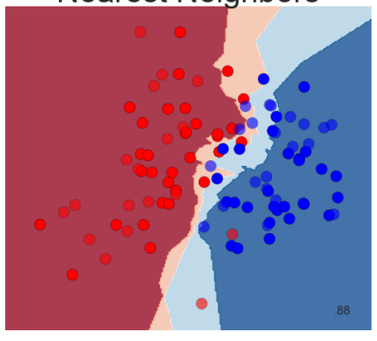
Linear SVM
.47

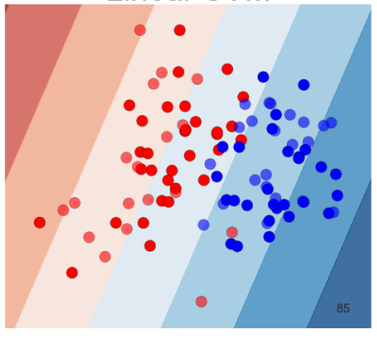
RBF SVM
.90

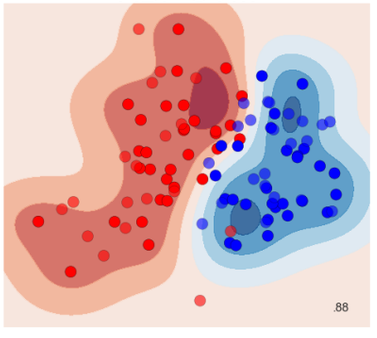
Decision Tree
.82

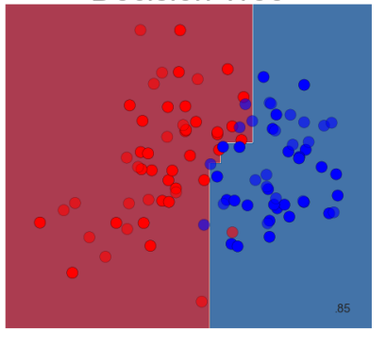
Random Forest
.90

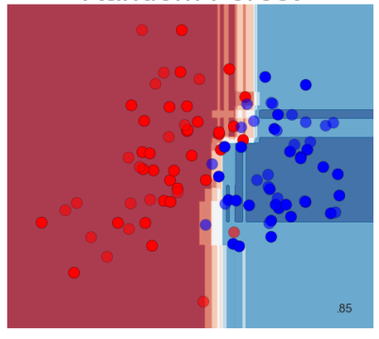



Nearest Neighbors
.88


Linear SVM
.85

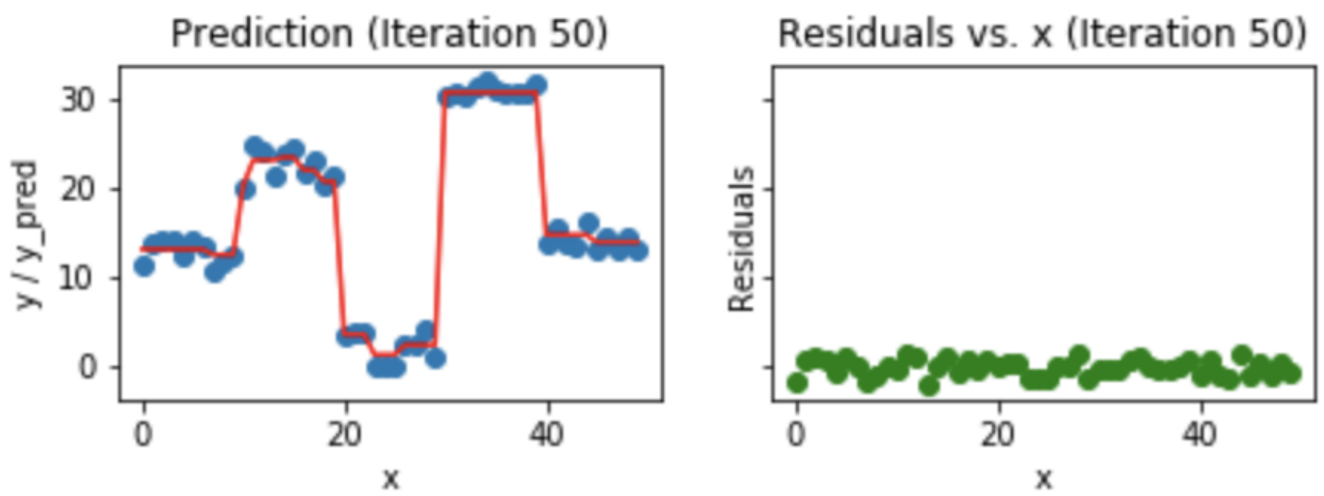

RBF SVM
.88


Decision Tree
.85


Random Forest
.85

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



Imbalanced Classes

Uneven Classes…

Modify the sets to look highly uneven. The accuracy increases. Why?

Check the weights keyword

More advanced techniques:

- weights
- sampling & averaging (smote)
- generating artificial rare cases


2/17
After 40 years as a teacher, Gary McAdams finally manages to bore a student to death
© 2015 Wulffmorgenthaler/Dist. by Universal Uclick www.gocomics.com

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)
```

Hands-on Session

- Datasets:

  - Ondrejov 2m telescope spectral lines (pca+classification)
    spectral_lines.npz
    spec=spectra, target=class

  - SDSS photometry stars + galaxies (classification)
    sdss_photo.csv
    u,g,r,i,z, u-g, g-r, r-i, i-z, target class

  - SDSS photometric redshift of quasars (regression)
    sdssphotoz.npy
    u,g,r,i,z,redshift

  - BATSE duration distribution of GRBs (clustering)
    T90 is the 5th column

to be continued…

- Neural networks
- Convolutional NN
- Auto-encoders
- Keras with Tensorflow backend

“Once you stop (machine) learning, you start dying.”

– Albert Einstein