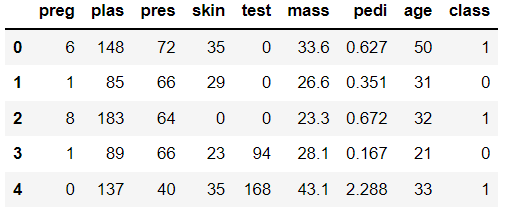
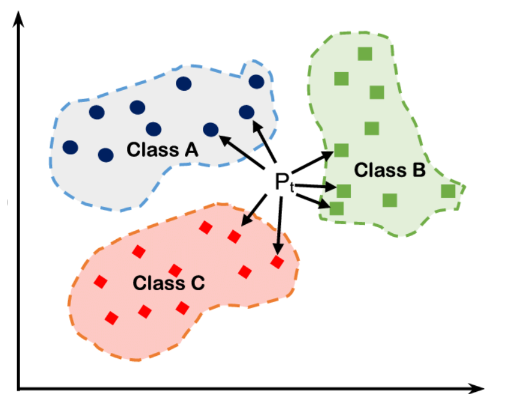
**Genaral problem definition**

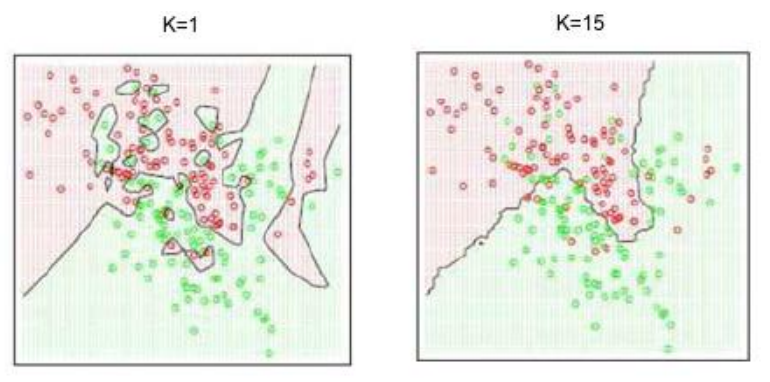
* Classification: supervised learning problem
* Need: function h(x) = y   
  x: input (features), y: output (label)
* (xi, yi) pairs (feautre-label pairs)  
  xi is d dimensional
* Example: diabetes classification  
  

(Jupyter – 7)

* Classification types: binary, multi-class
* Binary classification: y: 2 categories
* In multi-class case: one-hot encoding   
  e.g. categories a,b,c -> if cat b: (0,1,0)
* Terms
* Classification examples / applications:
  + Should we advertise the given product to the given client? (Is there a big chance that she will be interested?) (Y/N)
  + Character recognition. Which number can be seen? (0/1/2/…/9/10)
  + Is the client creditworthy? (Y/N)
  + Is the given email a spam? (Y/N)
  + Psychological profils (melancolic, flegmatic, sangvinic, coleric, …)
  + Identifying illness
  + Quality assurance
  + Sentiment analysis

**kNN**

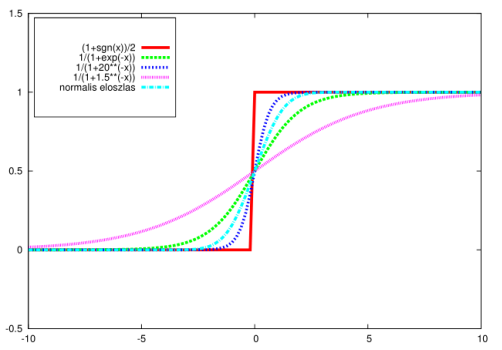
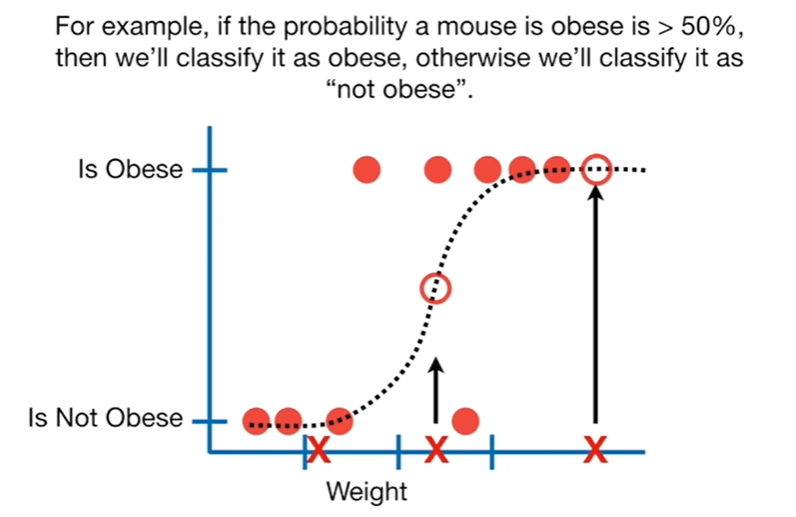
* Basic classification algorithm
* Suppose you already have datapoints and their classes
* And a new datapoint comes
* 
* Steps:
  1. Choose k
  2. Calculate the distances of the new datapoint and the others (alredy classified)
  3. Find the k smallest distances (k closest)
  4. Classify the point – majority voting
* A képen szöveg látható

  Automatikusan generált leírás
* The effect of k:  
  
* Pros, cons
* Task
  + load the iris dataset from sklearn.datasets
  + study the data
  + create X (features) and y (label)
  + use the built-in train\_test\_split method to create the train and test datasets
  + use the KNeighborsClassifier to create the knn model
  + fit the model (on the train data)
  + do prediction on the X\_test dataset (save to y\_pred)
  + calculate the accuracy of the model, use the accuracy\_score function
  + Use a for loop to check several k values (from 1 to 26). Collect the accuracy scores to a list. Finally plot the relationship between k and the accuracy.
  + Which k would you choose now? (Think of the elbow rule, just now it is upside-down)

**Most basic decision boundary**

* Picture: simple decision case
* Looking for a perfect decision line (surface in higher dimensions) = which separates the categories
* Why is this example easy?
  + Classes are far from each other
  + Can be separated by LINEAR line  
    (linear models are the most basic models)

**Logistic regression**

* Caution! Classification alg, and not regression!
* „classification via regression”
* Log.reg. turns the lin. reg. line to an S curve   
  (S shaped logistic function)
* Teach a linear regression algorithm such that   
  - if the predicted value neg. -> class 0  
  - if pos. -> class 1  
  (basically same as the signum function)  
  perdiction: TRESHOLD
* Try to make the signum function smoother (near the border we are not so sure) -> predict the probability of belonging to the class
* 
* Binary classification alg:
  + Instead of predicting a continuous variable (like size, e.g.lin.reg. pred size from weight)
  + Now we have binary categorical variable, e.g. True/False
* Datapoints in 1 dimension!! (1dim input)
*   
  (<https://www.youtube.com/watch?v=yIYKR4sgzI8>)
* Probabilistic interpretation
* Example (Jupyter)

**Stability of boundary: Max margin classification**

* Which separation line is the best?
  + Let’s add some noise to one of the datapoints
  + Which boundary is less sensitive?
* SVM classifier: Support Vector Machines
  + decision line (hyperplane)
  + support vectors
  + margin
* intuition

**Support vector machines (Linear)**

* we want a wide margin -> new objective fn with some additional constraints
* maybe it worth to misclassify one point and have better margin (we disregard that point)
* check the line which we would draw if we considered all the points
* w: the bigger the weight vector, the smaller the margin (inversely proportional)
* kNN: least compressive model  
  SVM: most compressive model
* pros and cons

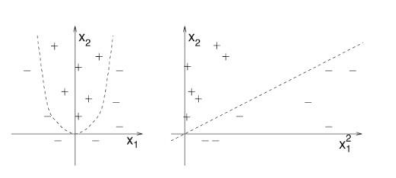
**What if decision boundary is not linear? - 1. General Linear Models**

* picture: still simple case

(for a human simple, but for a linear model …)

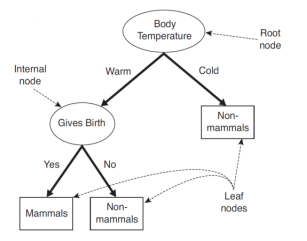
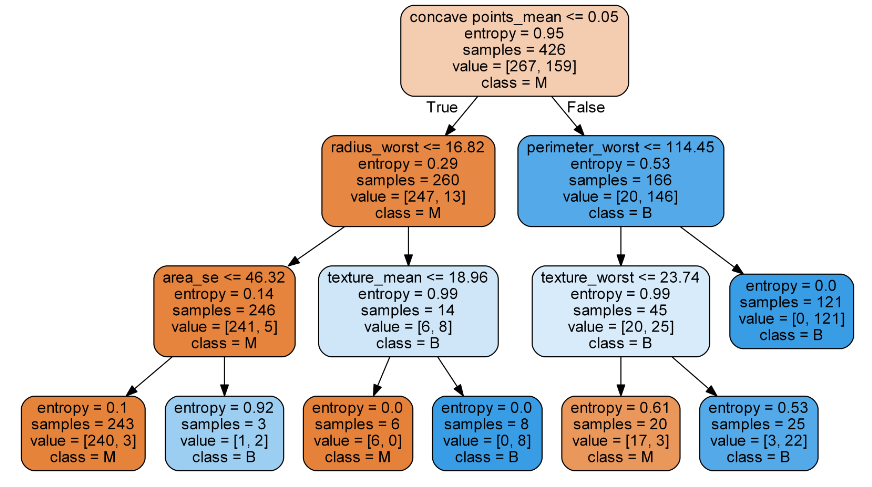
* 🡪 tricks for making it linearly separable:  
  A képen nyíl látható

  Automatikusan generált leírás



**What if decision boundary is not linear? - 2. Decision trees**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Body Temperature | Gives Birth | Label (mammel or not) |
|  |  |  |  |

* 
* Graph representations of iterative decisions
* Root and internal nodes are the features  
  (we make the decisions along them)
* Decisions = splits in the tree
* Aim: homogeneous leaf nodes – decision on the target label
* complex, locally linear decision boundary (see on the pictures)
* 

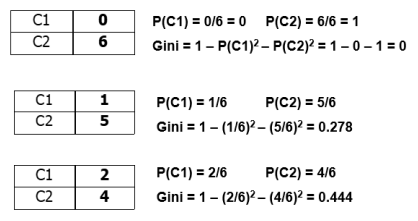
**How do we learn it?**

* Greedy algorithm

**Information gain**

* Formula shows: it is based on entropy
* Entropy: measures the degree of chaos
* More homogeneous -> lower entropy
* Bigger uncertanty -> higher entropy
* Information gain is used to decide which feature to split on
* choose the split that results in the purest daughter nodes

**GINI impurity**

* 

**An ensemble extension: Random Forest**

* Ensemble method: combines the power of more models
* RF: combines the predictions of several decision trees (average their results)
* -> more accurate output
* reduce risk of overfitting on the training data
* but slower than decision tree (trains e.g. 100 trees)
* LATER: ensemble; overfitting

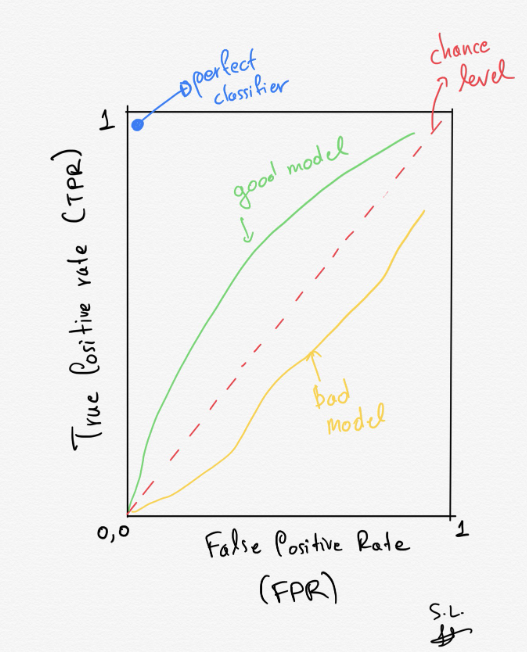
**How to measure classification**

* Stick to the binary case
* Already saw: check if prediction is is the same as the label
* But which class did we predict well   
  or which class did we fail to predict?  
  can be important! -> confusion matrix
* Example (confusion matrix): rare type of cancer
* We want to concentrate on finding those who have cancer (positive)
* But out of 2 we just found 1
* Is the model really that good? (99.9%) …

**Precision and Recall**

**F1**

**ROC and AUC and Lift chart**

* Watch at home
* Main idea:  
   

**Generalization to more than two classes**

*-Natural ectension:* There are models which uses naturally all your classes (let’s check kNN and decision tree)

*-Reduction to binary:*

* One-vs-Rest
  + e.g. classes cat, mouse, dog.
    - cat vs not cat (M and D)
    - dog vs not dog
    - mouse vs not mouse
  + k classes -> k binary classifiers
  + We can choose the "most confident" of them as the class label
  + P(datapoint is cat) > P(datapoint is dog) and P(datapoint is cat)>P(datapoint is mouse) 🡪 cat
* One-vs-One
  + Compare each class against each:  
    D-M, D-C, M-C
* Default in sklearn is the One-vs-Rest