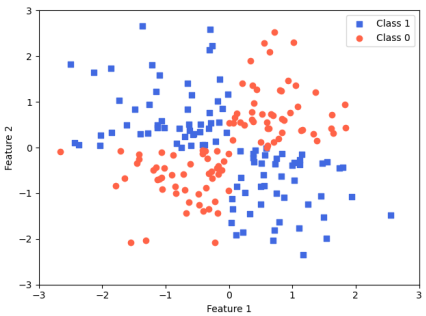
STILL CHAPTER 3

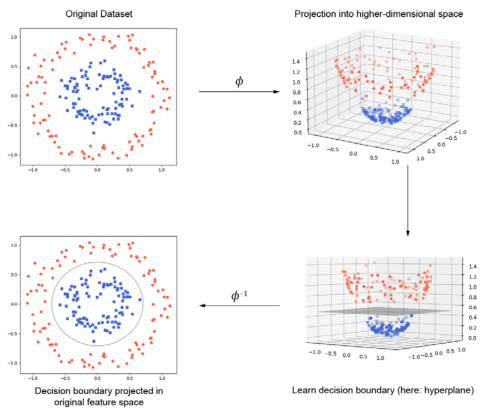
KERNEL METHODS FOR LINEARLY INSEPARABLE DATA

- SVMS CAN BE KERNELIZED FOR NON-LINEAR CLASSIFICATION
- KERNEL SVM → MOST COMMON VARIANT OF SVM



- XOR DATASET WITH NOISE
  ↳ CAN'T SEPARATE EXAMPLES FROM NEGATIVE + POSITIVE CLASSES.
- IDEA OF KERNEL METHOD → CREATE NONLINEAR COMBINATIONS OF ORIGINAL FEATURES TO PROJECT THEM ONTO HIGHER-DIMENSIONAL SPACE VIA MAPPING FUNCTION ϕ

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$



USING KERNEL TRICK TO FIND SEPARATING HYPERPLANES IN A HIGH-DIMENSIONAL SPACE

- NON-LINEAR PROBLEM → MAPPING FUNCTION (ϕ) → TRAIN LINEAR SVM TO CLASSIFY DATA IN NEW FEATURE SPACE → ϕ (MAPPING FUNCTION) → TRANSFORM NEW UNSEEN DATA TO CLASSIFY IT USING LINEAR SVM.
- HOWEVER → MAPPING FUNCTION IS COMPUTATIONALLY VERY EXPENSIVE, ESPECIALLY HIGH-DIMENSIONAL DATA
  ↳ USE KERNEL TRICK
  - QUADRATIC PROGRAMMING TO TRAIN SVM. → REPLACE DOT PRODUCT $x^{(i)T}x^{(j)} = \phi(x^{(i)})^T \phi(x^{(j)})$
  - TO SAVE EXPENSIVE DOT PRODUCT → WE DEFINE KERNEL FUNCTION:

$$\kappa(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^T \phi(x^{(j)})$$

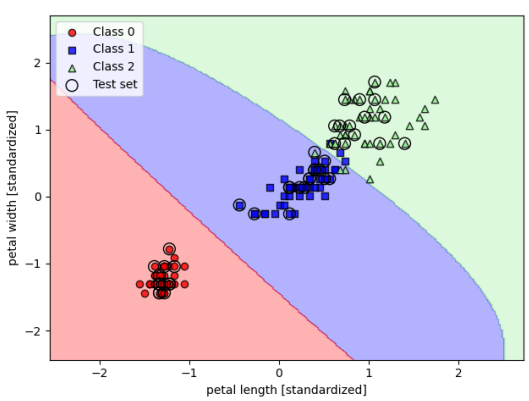
- ONE OF MOST WIDELY USED KERNELS → RADIAL BASIS FUNCTION (RBF) → CALLED GAUSSIAN KERNEL

$$\kappa(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2}\right)$$

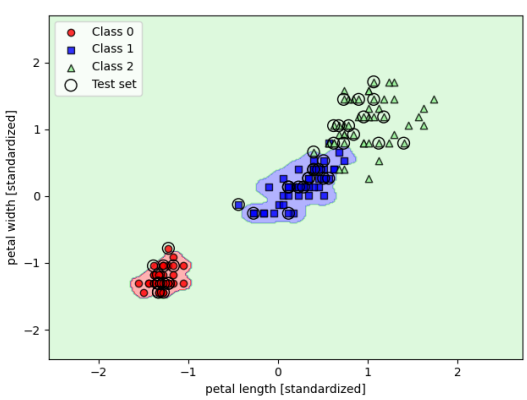
$\gamma = \frac{1}{2\sigma^2}$ = FREE PARAMETER TO BE OPTIMIZED

SIMPLIFIED ⇒ $\kappa(x^{(i)}, x^{(j)}) = \exp\left(-\gamma \|x^{(i)} - x^{(j)}\|^2\right)$

- KERNEL → INTERPRETED AS SIMILARITY FUNCTION BETWEEN PAIR OF EXAMPLES.
- MINUS SIGN → INVERTS DISTANCE INTO SIMILARITY SCORE → FALLS BETWEEN RANGE (0,1)



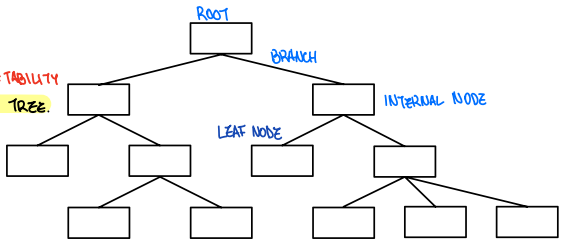
γ IS SMALL → RBF KERNEL SVM MODEL RELATIVELY SOFT



γ IS HIGH - FITS DATA BUT WILL HAVE HIGH GENERALIZATION ERROR

- γ PLAYS AN IMPORTANT ROLE IN CONTROLLING OVERFITTING OR VARIANCE WHEN ALGORITHM IS TOO SENSITIVE TO FLUCTUATION IN TRAINING DATASET.

DECISION TREE LEARNING



- DECISION TREE CLASSIFIER → ATTRACTIVE MODEL WHEN WE CARE ABOUT INTERPRETABILITY
  ↳ BREAK DOWN DATA BY MAKING DECISIONS BASED ON QUESTIONS → SHAPE LIKE A TREE.
- MODEL LEARNS CLASS LABELS FROM FEATURES BY SERIES OF QUESTIONS ASKED.
- START AT ROOT → SPLIT DATA ON FEATURE THAT RESULTS IN LARGEST INFORMATION GAIN (IG)
- TRAINING EXAMPLES IN EACH NODE BELONG TO THE SAME CLASS.
  ↳ SPLITTING HAPPENS UNTIL ALL LEAVES ARE "PURE"
- WE WANT TO PRUNE TREE BY SETTING LIMITS → TREE CAN BECOME VERY DEEP → PRONE TO OVERFITTING.

MAXIMISING INFORMATION GAIN (IG)

- SPLITTING TREE AT MOST INFORMATIVE FEATURES → DEFINE OBJECTIVE FUNCTION → OPTIMISE ALGORITHM.

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

- f = FEATURE TO PERFORM SPLIT.
- D_p AND D_j = DATASET OF PARENT AND jTH CHILD NODE
- I = IMPURITY MEASURE
- N_p = TOTAL NUMBER OF TRAINING EXAMPLES AT PARENT NODE
- N_j = NUMBER OF EXAMPLES IN jTH CHILD NODE.

- INFORMATION GAIN → DIFFERENCE BETWEEN IMPURITY OF PARENT NODE + SUM OF CHILD NODE IMPURITIES
- SIMPLICITY + REDUCE COMBINATIONAL SEARCH SPACE → MOST LIBRARIES → IMPLEMENT BINARY DECISION TREES
  ↳ EACH PARENT NODE → SPLIT INTO TWO CHILD NODES, D_{left}, D_{right}:

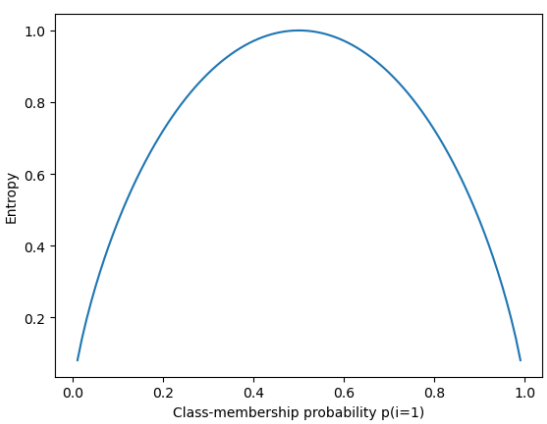
$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

- THREE IMPURITY MEASURES OR SPLITTING CRITERIA USED IN BINARY DECISION TREES:
  - GINI IMPURITY (I_G)
  - ENTROPY (I_H)
  - CLASSIFICATION ERROR (I_E)

ENTROPY

- FOR ALL NON-EMPTY CLASSES ($p(i|t) \neq 0$):

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$



- $p(i|t)$ = PROPORTION OF EXAMPLES THAT BELONG TO i FOR PARTICULAR NODE, t.
- ENTROPY = 0 IF ALL EXAMPLES AT A NODE BELONG TO SAME CLASS.
- ENTROPY = MAXIMAL - UNIFORM CLASS DISTRIBUTION.

GINI IMPURITY

- CRITERION TO MINIMIZE THE PROBABILITY OF MISCLASSIFICATION:

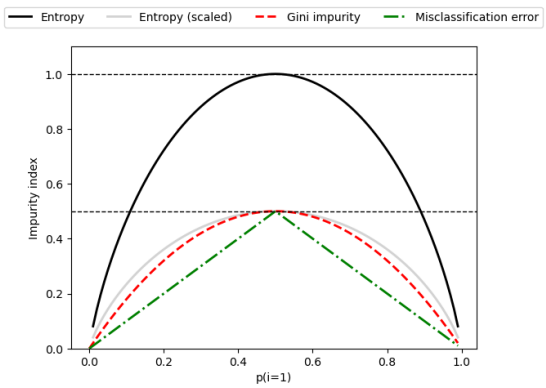
$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

- GINI IMPURITY = MAXIMAL IF ALL THE CLASSES ARE PERFECTLY MIXED
- IN PRACTICE BOTH GINI IMPURITY AND ENTROPY → TYPICALLY YIELD VERY SIMILAR RESULTS.
- NOT WORTH EVALUATING TREES USING DIFFERENT IMPURITY CRITERIA → EXPERIMENT RATHER WITH DIFFERENT PRUNING CUT-OFFS.

CLASSIFICATION ERROR

$$I_E(t) = 1 - \max\{p(i|t)\}$$

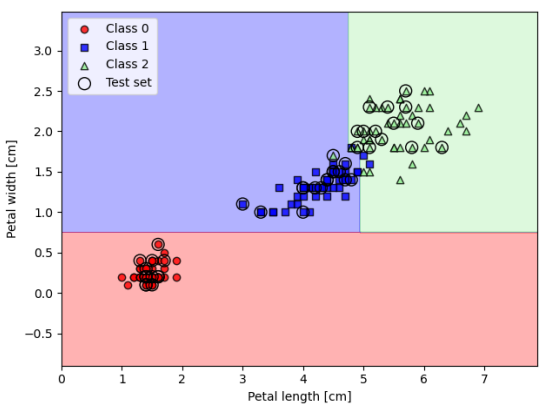
- USEFUL FOR PRUNING → NOT RECOMMENDED FOR GROWING DECISION TREES
  → LESS SENSITIVE TO CHANGES IN CLASS PROBABILITIES OF THE NODE.



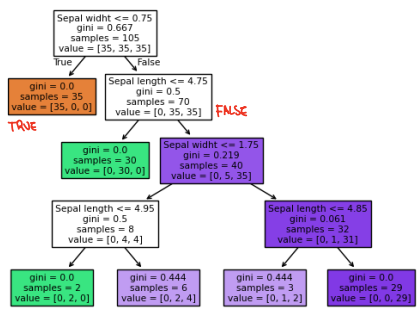
- DIFFERENT IMPURITY INDICIES FOR DIFFERENT CLASS-MEMBERSHIP PROBABILITIES IN RANGE 0,1.

BUILDING A DECISION TREE (SCIKIT-LEARN)

- DECISION TREES CAN BUILD COMPLEX DECISION BOUNDARIES → DIVIDING FEATURE SPACE INTO RECTANGLES



- SCIKIT-LEARN DECISION TREE → MAX DEPTH OF 4, GINI IMPURITY.
- FEATURE SCALING NOT REQUIRED FOR DECISION TREES.



CAN VISUALISE DECISION TREE MODEL AFTER TRAINING WITH SCIKIT-LEARN

`from sklearn import tree`

COMBINING MULTIPLE DECISION TREES VIA RANDOM FOREST

- ENSEMBLE METHOD → BECAME POPULAR → GOOD CLASSIFICATION PERFORMANCE + ROBUSTNESS TOWARD OVERFITTING.
- RANDOM FOREST → GOOD SCALABILITY + EASY TO USE.
  - ↳ ENSEMBLE OF DECISION TREES.
  - ↳ IDEA BEHIND RANDOM FOREST → AVERAGE MULTIPLE (DEEP) DECISION TREES THAT INDIVIDUALLY SUFFER FROM HIGH VARIANCE
    - ↳ BUILD A MORE ROBUST MODEL → BETTER GENERALIZATION PERFORMANCE + LESS SUSCEPTIBLE TO OVERFITTING.
  - FOUR SIMPLE STEPS:
    1. DRAW RANDOM BOOTSTRAP SAMPLE SIZE n (RANDOMLY CHOOSE n EXAMPLES FROM TRAINING DATASET WITH REPLACEMENT.)
    2. GROW DECISION TREE FOR BOOTSTRAP SAMPLE.
       - ↳ EACH NODE:
         a. RANDOMLY SELECT d FEATURES WITHOUT REPLACEMENT.
         b. SPLIT NODE USING OBJECTIVE FUNCTION
    3. REPEAT 1-2 k TIMES
    4. AGGREGATE PREDICTIONS OF EACH TREE → MAJORITY VOTING

WITH AND WITHOUT REPLACEMENT

- WITH → CHOSEN FEATURE IS PUT BACK → CAN BE CHOSEN MULTIPLE TIMES.
- WITHOUT → CHOSEN FEATURE NOT PUT BACK → FEATURE CAN'T BE DRAWN AGAIN.

- RANDOM FOREST → ADVANTAGE → DON'T HAVE TO WORRY ABOUT CHOOSING GOOD HYPERPARAMETERS.
  - ↳ HOWEVER → DOESN'T OFFER SAME LEVEL OF INTERPRETABILITY AS DECISION TREE.
- DON'T NEED TO PRUNE RANDOM FOREST.
- ONLY PARAMETER TO CARE ABOUT → NUMBER OF TREES, k → LARGER NUMBER OF TREES = BETTER PERFORMANCE OF RANDOM FOREST CLASSIFIER → AT COST OF INCREASED COMPUTATIONAL COST.
- SOME HYPERPARAMETERS CAN BE OPTIMISED (LESS COMMON IN PRACTICE) → SIZE, n + NUMBER OF FEATURES, d
  - ↳ CONTROLS BIAS-VARIANCE TRADEOFF OF RANDOM FOREST.

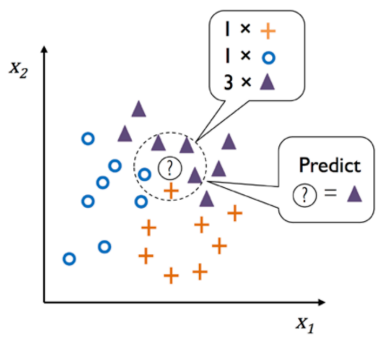
- INCREASING BOOTSTRAP SAMPLE → INCREASED DIVERSITY AMONG INDIVIDUAL TREES.
- SHRINKING BOOTSTRAP SAMPLE → INCREASE RANDOMNESS → REDUCE OVERFITTING ⇒ BUT LOWER OVERALL PERFORMANCE.
- `RANDOMFORESTCLASSIFIER` (SCIKIT-LEARN) → BOOTSTRAP SAMPLE CHOSEN EQUAL TO NUMBER OF TRAINING EXAMPLES IN ORIGINAL TRAINING SET.

K-NEAREST NEIGHBORS

- K-NEAREST NEIGHBOR (KNN) CLASSIFIER → "LAZY" → NOT BECAUSE OF ITS SIMPLICITY
  → KNN DOESN'T LEARN A DISCRIMINATIVE FUNCTION, MEMORISES TRAINING DATASET INSTEAD.

PARAMETRIC VS. NON-PARAMETRIC MODELS

- PARAMETRIC MODEL → ESTIMATE PARAMETERS FROM TRAINING DATA → LEARN FUNCTION TO CLASSIFY NEW DATA POINTS (WITHOUT NEED OF ORIGINAL TRAINING SET)
  ↳ EXAMPLES: PERCEPTRON, LOGISTIC REGRESSION, LINEAR SVM
- NON-PARAMETRIC → CAN'T BE CHARACTERIZED BY A FIXED SET OF PARAMETERS + NUMBER OF PARAMETERS CHANGE WITH AMOUNT OF TRAINING DATA.
  ↳ EXAMPLES: DECISION TREE CLASSIFIER/ RANDOM FOREST, KERNEL SVM
- KNN → SUB-CATEGORY OF NON-PARAMETRIC MODELS → INSTANCE BASED LEARNING → MEMORISE DATASETS
  → LAZY LEARNING IS SPECIAL CASE → ZERO COST DURING LEARNING PROCESS.

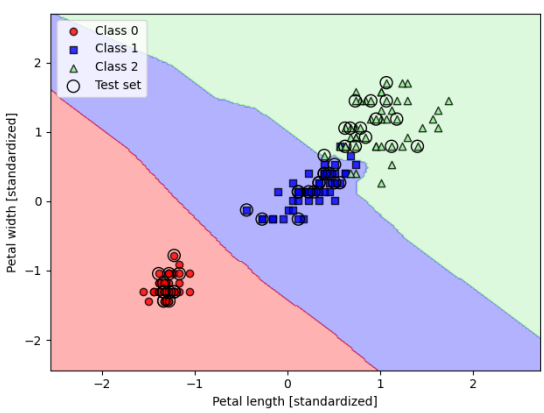


- KNN ALGORITHM IN 3 STEPS:
  1. CHOOSE NUMBER OF k + DISTANCE METRIC
  2. FIND K-NEAREST NEIGHBOR OF DATA THAT WE WANT TO CLASSIFY
  3. ASSIGN NEW CLASS BY MAJORITY VOTE

- FINDS K EXAMPLES IN TRAINING DATASET THAT ARE CLOSEST TO THE POINT WE WANT TO CLASSIFY.
- CLASS LABEL OF DATA POINT → DETERMINED BY MAJORITY VOTE.

ADVANTAGES + DISADVANTAGES OF MEMORY-BASED APPROACH

- ADVANTAGE: NEW TRAINING DATA → CLASSIFIER ADAPTS ASAP.
- DISADVANTAGE: COMPUTATIONAL COMPLEXITY GROWS LINEARLY WITH NUMBER OF EXAMPLES IN TRAINING DATASET. → LIMITED STORAGE CAPABILITIES.
- EFFICIENT DATA STRUCTURES FOR MEMORY BASED APPROACH = K-d TREE, BALL TREE
- MOST OF THE TIMES WE WORK WITH SMALL-MEDIUM SIZED DATASET → MEMORY BASED APPROACH → GOOD CHOICE FOR REAL-WORLD PROBLEMS.



- IN CASE OF TIES → SCIKIT-LEARN KNN → PREFER NEIGHBOR WITH CLOSER DISTANCE TO DATA RECORD TO BE CLASSIFIED.
  ↳ IF SIMILAR LENGTHS → WHAT IS FIRST IS CHOSEN.
- IMPORTANT TO FIND GOOD BALANCE FOR K
  ↳ EUCLIDEAN DISTANCE MEASURE REAL-VALUE EXAMPLES
  ↳ HAVE TO STANDARDIZE DATA SO THAT EACH FEATURE CONTRIBUTES EQUALLY TO DISTANCE.

→ MINKOWSKI DISTANCE → GENERALIZATION OF EUCLIDEAN + MANHATTAN DISTANCE

$$d\left(x^{(i)}, x^{(j)}\right) = \sqrt[p]{\sum_k \left|x_k^{(i)} - x_k^{(j)}\right|^p}$$

p = 1 = EUCLIDEAN p = 2 = MANHATTAN

- KNN VERY SUSCEPTIBLE TO OVERFITTING → CURSE OF DIMENSIONALITY
  ↳ FEATURE SPACE BECOMES INCREASINGLY SPARSE FOR INCREASING NUMBER OF DIMENSIONS OF A FIXED DATASET.
  → EVEN YOUR CLOSEST NEIGHBORS ARE FAR AWAY
  → USE FEATURE SELECTION + DIMENSIONALITY REDUCTION → AVOID CURSE OF DIMENSIONALITY.