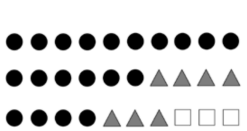
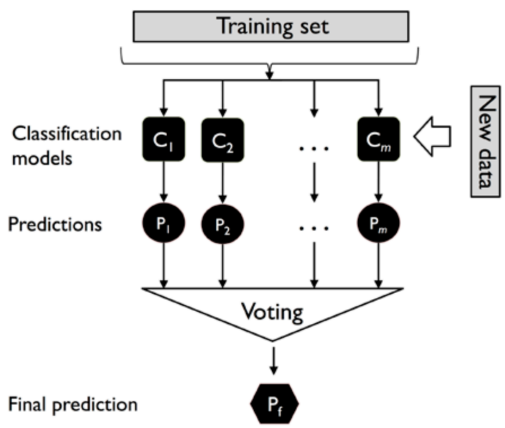
CHAPTER 7 - COMBINING DIFFERENT MODELS FOR ENSEMBLE LEARNING

LEARNING WITH ENSEMBLE

- ENSEMBLE METHOD → COMBINE CLASSIFIERS TO META-CLASSIFIER → BETTER GENERALIZATION PERFORMANCE.
- MAJORITY VOTING PRINCIPLE → CLASS LABEL WITH MORE THAN 50% OF VOTES → REFERS TO BINARY CLASS SETTINGS ONLY.
  - ↳ GENERALISE TO MULTICLASS SETTINGS → PLURALITY VOTING (UK → 'ABSOLUTE' + 'RELATIVE' VOTING) → CLASS LABEL WITH MOST VOTES





SEBASTIAN RASCHKA

- TRAINING DATASET → TRAINING m DIFFERENT CLASSIFIERS
- DEPENDING ON TECHNIQUE → USE DIFFERENT CLASSIFICATION ALGORITHM
  - ↳ DECISION TREES, SVM, LOGISTIC REGRESSION
- CAN ALSO USE SAME BASE CLASSIFICATION ALGORITHM → FIT DIFFERENT SUBSETS OF TRAINING DATASET
  - ↳ EXAMPLE → RANDOM FOREST → COMBINING DIFFERENT DECISION TREES.

⇐ GENERAL ENSEMBLE APPROACH. ↵

- PREDICT CLASS LABEL WITH MAJORITY OR PLURALITY VOTING → COMBINE PREDICTED CLASS LABELS CLASSIFIER, C_j + SELECTED CLASS LABEL, $\hat{y}$:

$$\hat{y} = \text{mode}\{C_1(x), C_2(x), \dots, C_m(x)\}$$ → MOST FREQUENT EVENT, mode = STATS.

- BINARY CLASSIFICATION + MAJORITY VOTE:

$$C(x) = \text{sign}\left[\sum_j^m C_j(x)\right] = \begin{cases} 1 & \text{if } \sum_j C_j(x) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- ENSEMBLE METHOD WORKS BETTER THAN INDIVIDUAL CLASSIFIERS → PROVEN WITH COMBINATORICS.
- ASSUMPTIONS FOR BINARY CLASSIFICATION TASK:
  - ↳ n - BASE CLASSIFIERS → EQUAL ERROR RATE, ε.
  - ↳ CLASSIFIERS INDEPENDENT + ERROR RATE NOT CORRELATED.

$$P(y \geq k) = \sum_k^n \binom{n}{k} \varepsilon^k (1-\varepsilon)^{n-k} = \varepsilon_{ensemble}$$ → ERROR PROBABILITY OF ENSEMBLE OF BASE CLASSIFIERS AS PROBABILITY MASS FUNCTION OF BINOMIAL DISTRIBUTION

↳ BINOMIAL COEFFICIENT

- E.G. 11 CLASSIFIERS ($n = 11$) + ERROR RATE 0.25 ($\varepsilon = 0.25$)

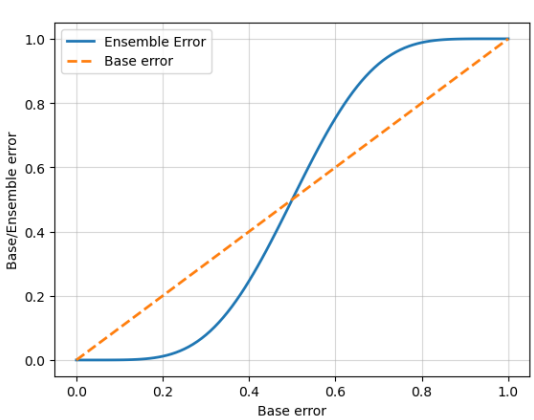
$$P(y \geq k) = \sum_{k=6}^{11} \binom{11}{k} 0.25^k (1-0.25)^{11-k} = 0.034$$ → ERROR RATE LOWER THAN OF EACH INDIVIDUAL CLASSIFIER.

BINOMIAL COEFFICIENT

- NUMBER OF WAYS WE CHOOSE K FROM SET OF SIZE n ⇒ "n choose k"

$$\frac{n!}{(n-k)!\,k!}$$

ERROR PROBABILITY OF ENSEMBLE ALWAYS BETTER



COMBINING CLASSIFIERS VIA MAJORITY VOTE

SINGLE MAJORITY VOTE CLASSIFIER

$$\hat{y} = \arg\max_i \sum_{j=1}^{m} w_j \chi_A\left(C_j(x) = i\right)$$

w_j = WEIGHT ASSOCIATED WITH BASE CLASSIFIER, C_j
$\hat{y}$ = PREDICTED CLASS LABEL OF THE ENSEMBLE
A = SET OF UNIQUE CLASS LABELS
χ_A = CHARACTERISTIC OR INDICATOR FUNCTION

- FOR EQUAL WEIGHTS → SIMPLIFIED EQUATION:

$$\hat{y} = \text{mode}\{C_1(x), C_2(x), \ldots, C_m(x)\}$$

EXAMPLE TO UNDERSTAND CONCEPT OF WEIGHTS:

- $C_1(x) \rightarrow 0$, $C_2(x) \rightarrow 0$, $C_3(x) \rightarrow 1$ $\quad \hat{y} = \text{mode}\{0, 0, 1\} = 0$
- WEIGHTS = C_1 AND $C_2 = 0.2$, $C_3 = 0.6$

↳ $\hat{y} = \arg\max_i [0.2 \times i_0 + 0.2 \times i_0, 0.6 \times i_1] = 1$

- $3 \times 0.2 = 0.6 \Rightarrow C_3$ HAS THREE TIMES THE WEIGHT THAN C_1 AND C_2. → $\hat{y} = \text{mode}\{0, 0, 1, 1, 1\} = 1$
- MODIFIED VERSION OF MAJORITY VOTE FOR PREDICTING CLASS LABELS: ← USING `predict_proba` METHOD FROM LOGISTIC REGRESSION.

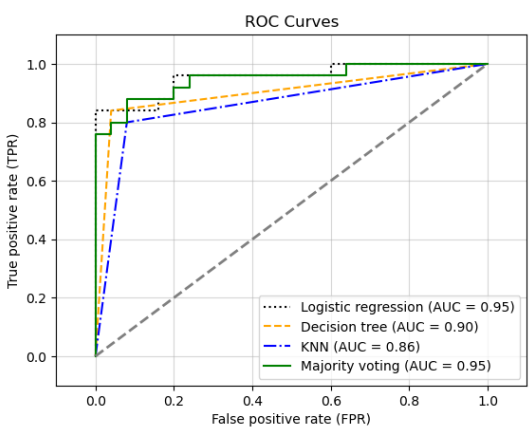
$$\hat{y} = \arg\max_i \sum_{j=1}^{m} w_j p_{ij}$$

p_{ij} = PREDICTED PROBABILITY OF jTH CLASS LABEL i

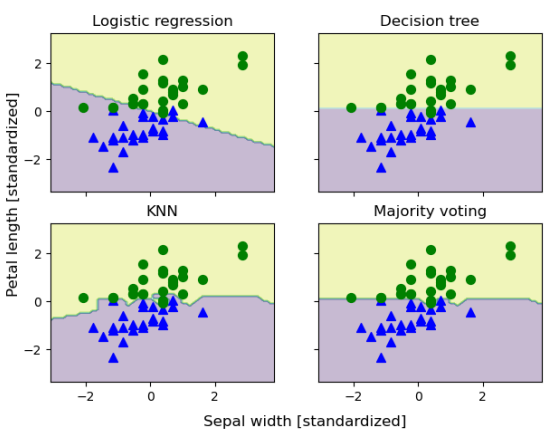
- BINARY CLASSIFICATION PROBLEM + SAME WEIGHTS
  - $C_1(x) \rightarrow [0.9, 0.1]$, $C_2(x) \rightarrow [0.8, 0.2]$, $C_3(x) \rightarrow [0.4, 0.6]$

$p(i_0 | x) = 0.2 \times 0.9 + 0.2 \times 0.8 + 0.6 \times 0.4 = 0.58$
$p(i_1 | x) = 0.2 \times 0.1 + 0.2 \times 0.2 + 0.6 \times 0.6 = 0.42$

$\hat{y} = \arg\max_i [p(i_0 | x), p(i_1 | x)] = 0$



- ROC CURVE COMPUTED TO TEST IF `MAJORITY VOTE CLASSIFIER` DOES GENERALISE WELL WITH UNSEEN DATA.



- DECISION REGIONS OF ENSEMBLE CLASSIFIER (STANDARDISED)

STACKING (ENSEMBLE METHODS)

- DAVID H. WOLPERT - "STACKED GENERALIZATION." 1992

INPUT: TRAINING DATA $D = \{x_i, y_i\}_{i=1}^{n} (x_i \in \mathbb{R}^m, y_i \in Y)$

OUTPUT: ENSEMBLE CLASSIFIER H

STEP 1: LEARN FIRST-LEVEL CLASSIFIERS. GREEN BOX = SAME AS MAJORITY VOTING

for $t \leftarrow 1$ to T do → $\{h_1, h_2, \dots h_T\}$

LEARN A BASE CLASSIFIER h_t BASED ON D

end for

STEP 2: CONSTRUCT NEW DATA SETS FROM D — FOR EACH TRAINING EXAMPLE

for $i \leftarrow 1$ to n do

CONSTRUCT A NEW DATA SET THAT CONTAINS $\{x'_i, y_i\}$, WHERE $x'_i = \{h_1(x_i), h_2(x_i), \dots h_T(x_i)\}$

↳ MODIFIED FEATURE VECTOR. ↳ PREDICTED CLASS LABELS VIA SCIKIT-LEARN

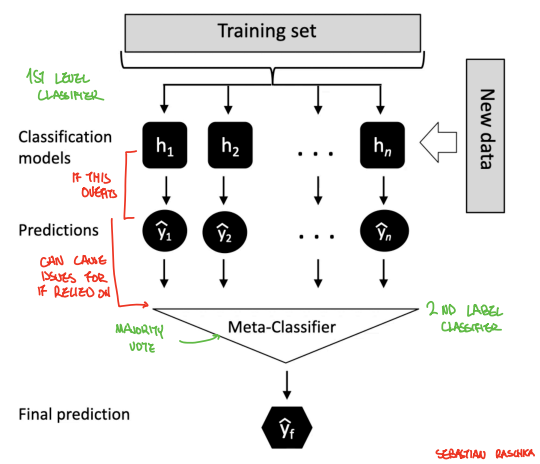
mode

end for

STEP 3: LEARN A SECOND-LEVEL CLASSIFIER → LEARN NEW CLASSIFIER BASED ON PREDICTIONS

LEARN A NEW CLASSIFIER h' BASED ON NEWLY CONSTRUCTED DATA SET.

return $H(x) = h'(h_1(x), h_2(x), \dots, h_T(x))$



SEBASTIAN RASCHKA - STACKING (VIDEO (4:)

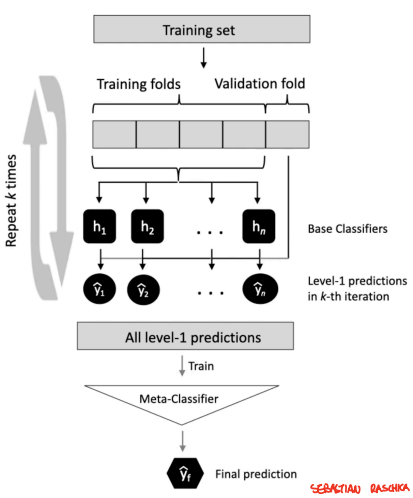
PROBLEM WITH STACKING ??

PRONE TO OVERFITTING

IMPROVEMENT OF STACKING → CROSS-VALIDATION

STACKING → K-FOLD CLASSIFICATION FOR 2ND LABEL CLASSIFIER

↳ USE K PREDICTIONS FOR



MODIFIED STACKING ALGORITHM

- INPUT + OUTPUT STAYS SAME

STEP 1: CROSS VALIDATION APPROACH PREPARATION FOR 2ND-LEVEL CLASSIFIER

RANDOMLY SPLIT D INTO K EQUAL-SIZED SUBSETS → $D = \{D_1, D_2, \dots D_K\}$

for $k \leftarrow 1$ to K do

STEP 1.1 LEARN 1ST-LEVEL CLASSIFIERS

for $t \leftarrow 1$ T do

LEARN CLASSIFIER h_{kT} FROM D/D_K

end for

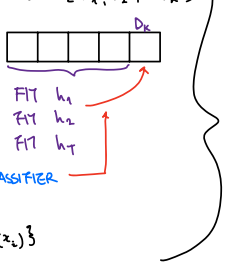
STEP 1.2 CONSTRUCT TRAINING SET FOR 2ND-LEVEL CLASSIFIER

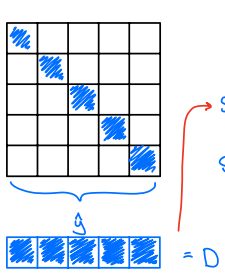
for $x_i \in D_K$ do

GET $\{x'_i, y_i\}$ WHERE $x'_i = \{h_1(x_i), h_2(x_i), \dots, h_{kT}(x_i)\}$

end for

end for





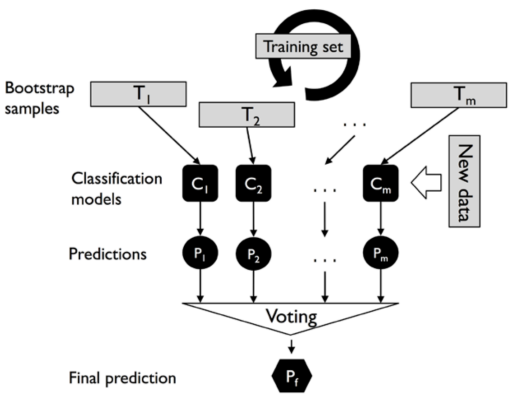
STEP 2: FIT 2ND-LEVEL CLASSIFIER

STEP 3: RE-LEARN 1ST-LEVEL CLASSIFIER

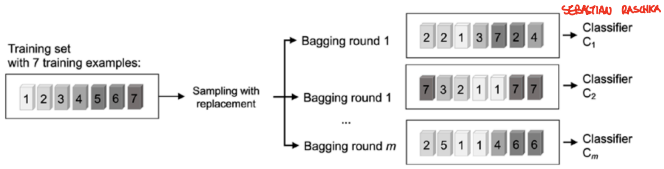
FIT ON WHOLE DATASET

BAGGING - ENSEMBLE CLASSIFIERS FROM BOOTSTRAP SAMPLES

- CLOSELY RELATED TO `MAJORITY VOTE CLASSIFIER`
- INSTEAD OF SAME TRAINING DATASET TO FIT INDIVIDUAL CLASSIFIERS → DRAW BOOTSTRAP SAMPLES (RANDOM SAMPLES WITH REPLACEMENT) FROM INITIAL TRAINING DATASET
  ↳ BOOTSTRAP AGGREGATING



SEBASTIAN RASCHKA



- RANDOM SAMPLES OBTAINED VIA BAGGING → BAGGING ROUND ...
- EACH SUBSET CONTAINS SOME DUPLICATES + SOME ORIGINAL EXAMPLES DON'T APPEAR
  ↳ SAMPLING WITH REPLACEMENT
- INDIVIDUAL CLASSIFIERS FIT TO BOOTSTRAP SAMPLES → PREDICTIONS COMBINED WITH MAJORITY VOTE.
- BAGGING RELATED TO RANDOM FOREST CLASSIFIER
  ↳ RANDOM FOREST → SPECIAL EXAMPLE → RANDOM FEATURE SUBSETS ALSO USED WHEN FITTING INDIVIDUAL DECISION TREES.

- COMPLEX CLASSIFICATION TASKS + DATASET'S HIGH DIMENSIONALITY CAN LEAD TO OVERFITTING IN SINGLE DECISION TREE → BAGGING COMES IN HANDY
- BAGGING EFFECTIVE APPROACH TO REDUCING VARIANCE OF MODEL
  ↳ INEFFECTIVE IN REDUCING MODEL BIAS → MODELS TOO SIMPLE TO CAPTURE TRENDS IN DATA.
    ↳ WHY WE WANT TO PERFORM BAGGING ON ENSEMBLE OF CLASSIFIERS WITH LOW BIAS → E.G. UNPRUNED DECISION TREES

ADAPTIVE BOOSTING - LEVERAGING WEAK LEARNERS

- ADAPTIVE BOOSTING (AdaBoost)
  ↳ ROBERT E. SCHAPIRE (1990) → 'THE STRENGTH OF WEAK LEARNABILITY.'
- BOOSTING - ENSEMBLE CONSISTS OF VERY SIMPLE BASE CLASSIFIERS → REFFERED TO AS 'WEAK LEARNERS' → SLIGHT EDGE OVER RANDOM GUESSING
  ↳ KEY CONCEPT → FOCUS ON TRAINING EXAMPLES THAT ARE HARD TO CLASSIFY
  WEAK LEARNERS SUBSEQUENTLY LEARN FROM MISCLASSIFIED EXAMPLES TO IMPROVE PERFORMANCE OF ENSEMBLE

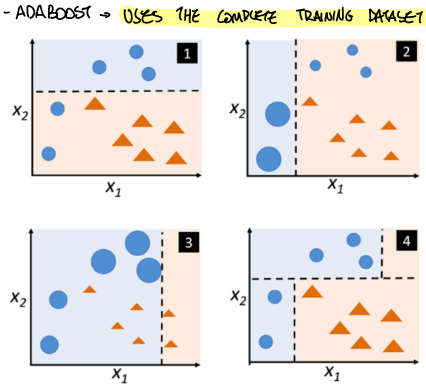
HOW ADAPTIVE BOOSTING WORKS:

- INITIAL FORMULATION - ALGO USES RANDOM SUBSETS OF TRAINING EXAMPLES FROM TRAINING DATASET WITHOUT REPLACEMENT.

FOUR KEY STEPS:

1. DRAW RANDOM SUBSET (SAMPLE) FROM TRAINING EXAMPLES, d_1 WITHOUT REPLACEMENT FROM TRAINING DATASET, D FOR WEAKER LEARNER C_1
2. SECOND RANDOM TRAINING SUBSET, d_2 WITHOUT REPLACEMENT + 50% OF EXAMPLES FROM PREVIOUSLY MISCLASSIFIED TO TRAIN WEAKER LEARNER C_2
3. FIND TRAINING EXAMPLES, d_3 IN TRAINING DATASET, D WHICH $C_1 + C_2$ DISAGREE WITH, TO TRAIN C_3
4. COMBINE C1, C2, C3 VIA MAJORITY VOTING

- LEO BREIMAN → BOOSTING CAN LEAD TO DECREASE IN BIAS + VARIANCE - COMPARED TO BAGGING
  ↳ IN PRACTICE → ADABOOST KNOWN FOR HIGH VARIANCE → TEND TO OVERFIT TRAINING DATA.
- ADABOOST → USES THE COMPLETE TRAINING DATASET TO TRAIN WEAK LEARNERS → TRAINING EXAMPLES REWEIGHTED AFTER EACH ITERATION TO LEARN FROM MISTAKES



SEBASTIAN RASCHKA - ML + PYTORCH BOOK.

1. SET WEIGHT VECTOR, w, TO UNIFORM WEIGHTS, $\sum_i w_i = 1$.
2. FOR j IN m BOOSTING ROUNDS:
   a. TRAIN WEIGHTED WEAK LEARNER, $C_j = \text{train}(X, y, w)$
   b. PREDICT CLASS LABELS, $\hat{y} = \text{predict}(C_j, X)$
   c. COMPUTE WEIGHTED ERROR RATE, $\epsilon = w \cdot (\hat{y} \neq y)$
   d. COMPUTE COEFFICIENT, $\alpha_j = 0.5 \log \frac{1-\epsilon}{\epsilon}$
   e. UPDATE WEIGHTS, $w := w \times \exp(-\alpha_j \times \hat{y} \times y)$
   f. NORMALISE WEIGHTS TO SUM TO 1, $w := w / \sum_i w_i$
3. COMPUTE FINAL PREDICTION, $\hat{y} = (\sum_{j=1}^{m} (\alpha_j \times \text{predict}(C_j, X)) > 0)$

- CONSIDERED BAD PRACTICE TO SELECT A MODEL BASED ON THE REPEATED USAGE OF THE TEST DATASET. → GENERALISATION PERFORMANCE MAY BE OVEROPTIMISTIC
- ENSEMBLE LEARNING INCREASES COMPUTATIONAL COMPLEXITY COMPARED TO INDIVIDUAL CLASSIFIERS
  ↳ IN PRACTICE ASK = IS IT WORTH THE MODEST IMPROVEMENT IN PREDICTIVE PERFORMANCE FOR INCREASED COMPUTATIONAL COST.

BOOSTING VS. STACKING

■ = BOOSTING ■ = STACKING

TRAINING ORDER - SEQUENTIAL TRAINING, PARALLEL TRAINING
CORE GOAL - REDUCE BIAS BY REPEATEDLY CORRECTING MISTAKES, EXPLOIT DIVERSITY - LEARN HOW TO MIX DIFFERENT STRONG MODELS
NEW MODEL FITTING - EMPHASIZES MISCLASSIFIED/LARGE ERROR SAMPLES. EACH BASE MODEL SEES SAME DATA, META-MODEL TRAINED ON PREDICTIONS.
COMBINATION METHOD - FINAL PREDICTIONS = WEIGHTED SUM/LARGE-ERROR SAMPLES, META-MODEL OUTPUT
PARALLELISATION - LIMITED (ITERATION DEPENDENT), BASE LEARNERS EASILY PARALLELISABLE
SENSITIVITY - CAN OVERFIT/NOISE-SENSITIVE IF NOT REGULARISED, CAN OVERFIT IF META-MODEL IS TOO FLEXIBLE → RELIES ON GOOD CV SETUP.
TYPICAL USE - GRADIENT BOOSTING TREES, XGBOOST, ADABOOST, COMPETITION/PREDICTION 'BLENDER' → COMBINE TREES, LINEAR MODELS, SVMS, NEURAL NETS, ETC ...

GRADIENT BOOSTING - TRAINING ENSEMBLE BASED ON LOSS GRADIENTS

- ANOTHER VARIANT OF BOOSTING CONCEPT → TRAINING WEAKER LEARNERS
- GRADIENT BOOSTING IS IMPORTANT → FORMS BASIS OF POPULAR ML ALGO LIKE XGBOOST → WELL KNOWN FOR KAGGLE COMPETITIONS
- GRADIENT BOOST FITS DECISION TREES IN AN ITERATIVE FASHION USING PREDICTION ERRORS
  ↳ DEEPER THAN DECISION TREE STUMPS + TYPICALLY A MAXIMUM DEPTH OF 3 TO 6. (MAX 8 - 64 LEAF NODES)
  ↳ DOES NOT USE PREDICTION ERRORS FOR ASSIGNING SAMPLE WEIGHTS → USED DIRECTLY TO FORM TARGET VARIABLE FOR FITTING NEXT TREE
  ↳ USES SAME GLOBAL LEARNING RATE FOR EACH TREE

OUTLINE OF GRADIENT BOOST ALGORITHM

- GRADIENT BOOST IS A GENERAL-PURPOSE SUPERVISED LEARNING METHOD → HERE WE'LL LOOK AT CLASSIFICATION (BINARY CLASSIFICATION EXAMPLE)
- BUILDS SERIES OF TREES, WHERE EACH TREE IS FIT ON ERROR (DIFFERENCE BETWEEN LABEL + PREDICTED VALUE)
- EACH ROUND, TREE ENSEMBLE IMPROVES → NUDGING TREE MORE IN THE RIGHT DIRECTION WITH SMALL UPDATES.
- UPDATES BASED ON LOSS GRADIENT → GOT GRADIENT BOOSTING NAME.
- STEP-BY-STEP (GENERAL ALGORITHM):

1. INITIALISE MODEL TO RETURN CONSTANT PREDICTION VALUE. DECISION TREE POST NODE (DECISION TREE WITH SINGLE LEAF NODE)

$$F_0(x) = \arg\min_{\hat{y}} \sum_{i=1}^{n} L(y_i, \hat{y})$$

$\hat{y}$ = VALUE RETURNED BY DECISION TREE
L = LOSS FUNCTION
n = n TRAINING EXAMPLES

2. FOR EACH TREE $m = 1, \dots, M$, M IS USER-SPECIFIED TOTAL NUMBER OF TREES
   a. COMPUTE DIFFERENCE BETWEEN PREDICTED VALUE $F(x_i) = \hat{y}_i$ AND CLASS LABEL y_i. SOMETIMES CALLED PSEUDO-RESPONSE OR PSEUDO-RESIDUAL
   FORMALLY → WRITE PSEUDO-RESIDUAL AS NEGATIVE OF LOSS FUNCTION WITH RESPECT TO PREDICTED VALUES:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)} \quad \text{for } i = 1, \dots, n$$

$F(x)$ = PREDICTION OF PREVIOUS TREE $F_{m-1}(x)$ (FIRST ROUND → CONSTANT VALUE FROM SINGLE NODE TREE)

   b. FIT A TREE TO PSEUDO-RESIDUAL r_{im}. R_{jm} TO DENOTE $j = 1 \dots J_m$ LEAF NODES OF RESULTING TREE IN ITERATION m.
   c. FOR EACH LEAF NODE R_{jm} → COMPUTE OUTPUT VALUE.

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \qquad \gamma_{jm} = \text{COMPUTED BY MINIMIZING LOSS FUNCTION}$$

   R_{jm} CAN CONTAIN MORE THAN ONE TRAINING EXAMPLE, HENCE SUMMATION
   d. UPDATE MODEL BY ADDING OUTPUT VALUE γ_m TO PREVIOUS TREE:

$$F_m(x) = F_{m-1}(x) + \eta \gamma_m$$

INSTEAD OF ADDING FULL PREDICTED VALUES OF CURRENT TREE γ_m TO PREVIOUS TREE F_{m-1}, SCALE γ_m BY LEARNING RATE η (SMALL STEP → TYPICALLY BETWEEN 0.01 - 1)

GRADIENT BOOST FOR CLASSIFICATION

- SINGLE TRAINING EXAMPLE, LOGISTIC LOSS:

$$L_i = -y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

- Log(odds):

$$\hat{y} = \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

- Log(odds) TO REWRITE EQUATION:

$$L_i = \log(1 + e^{\hat{y}_i}) - y_i \hat{y}_i$$

- PARTIAL DERIVATIVE OF LOSS FUNCTION WITH RESPECT TO log(odds), $\hat{y}$:

$$\frac{\partial L_i}{\partial \hat{y}_i} = \frac{e^{\hat{y}_i}}{1 + e^{\hat{y}_i}} - y_i = p_i - y_i$$

- AFTER THESE STEPS – NOW YOU ADD GRADIENT BOOST STEPS:

1. CREATE ROOT NODE → MINIMIZES LOGISTIC LOSS. LOSS MINIMIZED IF ROOT NODE RETURNS log(odds), $\hat{y}$.
2. FOR EACH TREE $m = 1, \ldots, M$, IS USER-SPECIFIED TOTAL NUMBER OF TREES
   a. CONVERT log(odds) INTO PROBABILITY USING LOGISTIC FUNCTION (THAT WE USED IN LOGISTIC REGRESSION):

$$p = \frac{1}{1 + e^{-\hat{y}}}$$

   THEN COMPUTE PSEUDO-RESIDUAL – NEGATIVE PARTIAL DERIVATIVE OF LOSS WITH RESPECT TO log(odds)
   ↳ DIFFERENCE BETWEEN CLASS LABEL AND PREDICTED PROBABILITY

   b. FIT NEW TREE TO PSEUDO-RESIDUALS

   c. FOR EACH R_{jm}, COMPUTE VALUE γ_{jm} – MINIMISES LOSS FUNCTION

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$

$$= \log(1 + e^{\hat{y}_i + \gamma}) - y_i(\hat{y}_i + \gamma)$$

RESULTS IN: $\gamma_{jm} = \dfrac{\sum_i y_i - p_i}{\sum_i p_i(1 - p_i)}$ → THIS IS ONLY FOR R_{jm}, AND NOT FULL TRAINING SET.

   d. UPDATE MODEL BY ADDING GAMMA VALUE FROM 2c WITH LEARNING RATE η:

$$F_m(x) = F_{m-1}(x) + \eta \gamma_m$$

- AFTER BOOSTING ALGO COMPLETE → PREDICT CLASS LABELS BY THRESHOLDING PROBABILITY VALUES OF FINAL MODEL, $F_m(x) = 0.5$, LIKE LOGISTIC REGRESSION. ← WILL PRODUCE NON-LINEAR DECISION BOUNDARIES UNLIKE

XG BOOST

- GRADIENT BOOSTING IS A SEQUENTIAL PROCESS → SLOW TO TRAIN (INEFFICIENT TIME WISE)
  ↳ ALTERNATIVE ⇒ XGBOOST
- XGBOOST (EXTREME GRADIENT BOOST) - PROPOSED SEVERAL TRICKS + APPROXIMATIONS → SPEED UP TRAINING PROCESS.