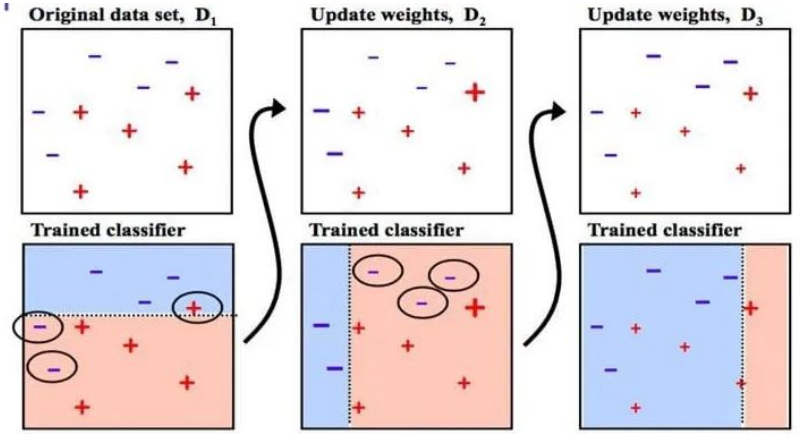
AdaBoost

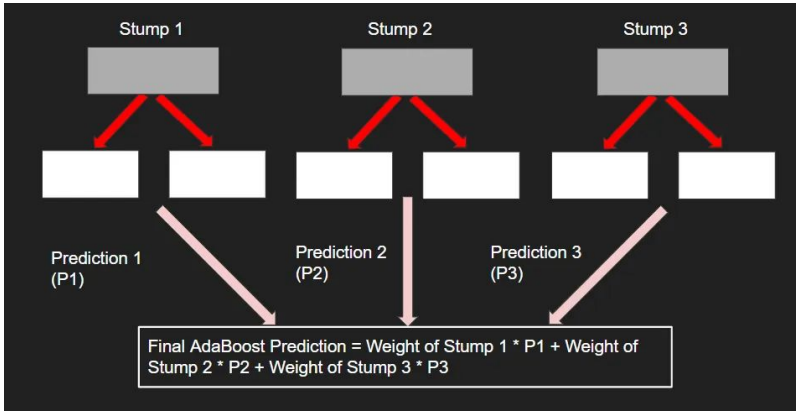
(Adaptive Boosting)

Team: little cutesy cow and horse
Members: Wendi Liao, Yicheng Lu, Xuetong Tang, Ke Zhang

Mechanism

- Boosting Method:
  - Subsequent model is trained based on errors of previous models
  - ***Adaboost: Iteratively adjust the weights of samples and classifiers***
- Ensemble Method
  - The final outcome is a weighted sum of the weak learners' predictions
- Weak Learners
  - Decision Stumps
  - Half Space
  - ***Decision Tree – Our choice!***
  - Neural Network





Mathematics

- **Representation**

$$\mathcal{E}(\mathcal{H}, T) = \{x \mapsto h_S(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right) : w \in \mathbb{R}^T, \forall t, h_t \in \mathcal{H}\}$$

- **Loss Function**
  - **Overall Loss Function – 0-1 Loss:**

$$L_S(h_S) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[h_S(x_i) \neq y_i]}$$

  - **Redefined Loss on Training Examples:**

$$\epsilon_t \overset{\text{def}}{=} L_{D^{(t)}}(h_t) \overset{\text{def}}{=} \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[h_t(x_i) \neq y_i]} \quad \text{where} \quad D^{(t)} \in \mathbb{R}^m$$

Numerical Techniques - Optimizer

- **Inputs:**
  - Training set $S = (x_1, y_1), \ldots, (x_m, y_m)$
  - Weak learner WL
  - Number of rounds T

= Number of Decision Trees to train

Numerical Technique – Optimizer

Step 1: Initialize the weights for each point in training set S with w_i = 1/m

- **for i from 1 to M, do**
  - **initialize weight uniformly**

$$D^{(1)} = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$$

Uniform Distribution D: The first model starts by treating all examples equally important.

Numerical Technique – Optimizer

Step 2: Iteration

- **for t = 1, …, T, do**
  - **Invoke weak learner h_t (decision tree)**
  - **Compute the weighted 0-1 Loss**

$$\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[y_i \neq h_t(x_i)]}$$

  - **Set the weight of weak learner as:**

$$w_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

  - **Update points' weights for all i = 1, …, m**

$$D_i^{(t+1)} = \frac{D_i^{(t)}\exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)}\exp(-w_t y_j h_t(x_j))} \text{ for all } i = 1, \ldots, m$$

- **end for**

- **Epsilon** measures the weak learner's performance by summing up the weights of the misclassified examples.
- **Indication function** checks whether the weak learner predicted the wrong label for each example

- If misclassified, yi ht(xi) = -1, weight for this example increases.
- If correctly classified, yi ht(xi) = +1, weight decreases.

Numerical Technique – Optimizer

Step 3: Output

- **the hypothesis**

$$h_S(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$$

The **sign function** decides the final class label based on whether the sum is positive or negative

Previous Work

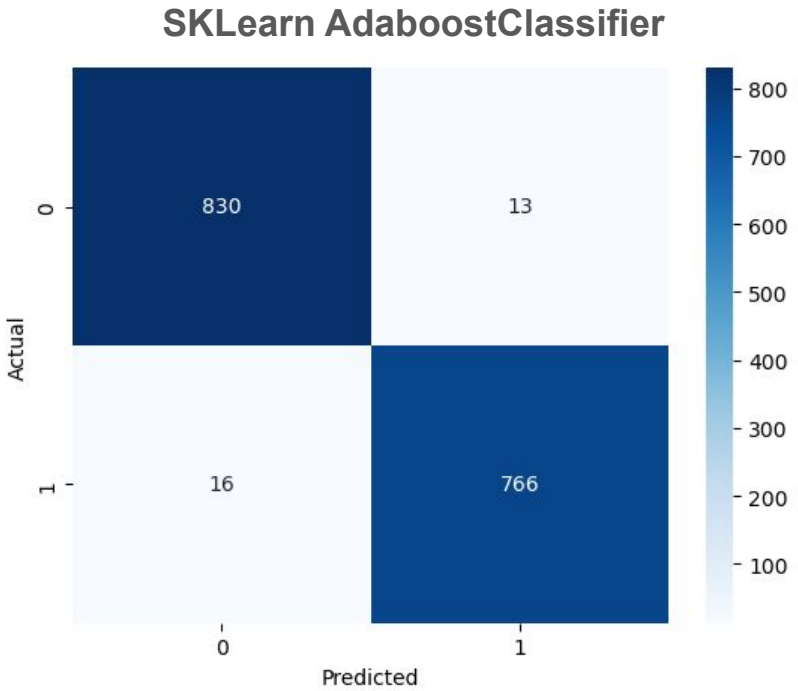
We choose scikit-learn's implementation of AdaBoost (AdaBoostClassifier) with a shallow decision tree (DecisionTreeClassifier) as the base estimator. We applied and compared scikit-learn's AdaBoost model and our AdaBoost model on the mushroom dataset:

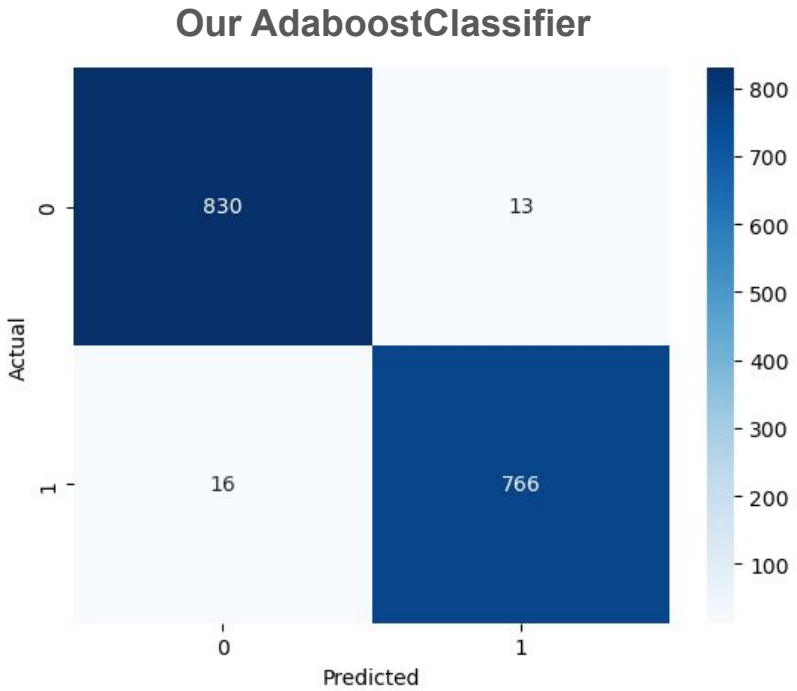
- **UCI Secondary Mushroom Dataset**
- **23 binary features describing mushroom characteristics (shape, color)**
- **a binary target indicating edibility (-1 for edible, 1 for poisonous)**
- **consists 1625 data points**

setup: 10 estimators, decision tree was set to a shallow depth of 1

Variable Name	Role	Type	Description
poisonous	Target	Categorical	
cap-shape	Feature	Categorical	bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
cap-surface	Feature	Categorical	fibrous=f,grooves=g,scaly=y,smooth=s
cap-color	Feature	Binary	brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
bruises	Feature	Categorical	bruises=t,no=f
odor	Feature	Categorical	almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
gill-attachment	Feature	Categorical	attached=a,descending=d,free=f,notched=n
gill-spacing	Feature	Categorical	close=c,crowded=w,distant=d
gill-size	Feature	Categorical	broad=b,narrow=n

Previous Work vs. Our Algorithm - Confusion matrix





Previous Work vs. Our Algorithm - Other metrics

SKLearn AdaboostClassifier

	precision	recall	f1-score	support
-1	0.98	0.98	0.98	843
1	0.98	0.98	0.98	782
accuracy			0.98	1625
macro avg	0.98	0.98	0.98	1625
weighted avg	0.98	0.98	0.98	1625

Model accuracy: 0.9821538461538462

Our AdaboostClassifier

	precision	recall	f1-score	support
-1	0.98	0.98	0.98	843
1	0.98	0.98	0.98	782
accuracy			0.98	1625
macro avg	0.98	0.98	0.98	1625
weighted avg	0.98	0.98	0.98	1625

Model accuracy: 0.9821538461538462

Results from both model are the same ~

Summary

- **Challenges**
  - **AdaBoost requires decision trees to be trained on weighted samples, where the weight of each training example is adjusted after each iteration. However, the implementation of decision tree from previous homework does not support weighted samples.**

  - **Solution: We incorporate weights into the splitting criterion**
    - 1. Ensure that weights are initialized when the decision tree is instantiated.
    - 2. In the _calc_gain function, we incorporate weights during the calculation of probabilities and splits.
    - 3. In the _split_recurs function, we ensure the left and right data splits account for the updated weights of the samples

Summary

- **Interesting point**
  - **Weighted Metrics for Splitting**
    - **Modifying decision tree splitting criteria (like Gini index or entropy) to incorporate weights is a subtle but impactful change. It teaches how core algorithms can be adapted for specific purposes.**

  - **Key Insight:**
    - **This process shows how foundational machine learning algorithms can be reengineered to work harmoniously in an ensemble setting.**