CAI 4104/6108 – Machine Learning Engineering:

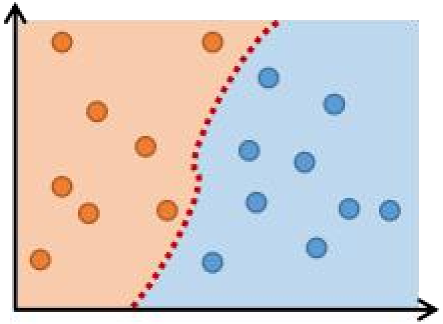
Ensembles

Prof. Vincent Bindschaedler

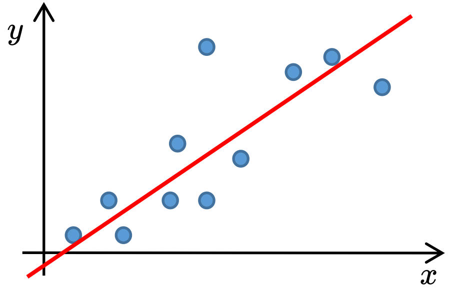
Spring 2024

Reminder: Supervised Learning

- Classification
  - Task: predict the corresponding label
  - Different types:
    - Binary classification: there are only two classes (0,1; +,-, etc.)
    - Multiclass: more than two classes
    - Multi-label: each instance can belong to more than one class
    - One-class: there is only one class, we want to distinguish it from everything else



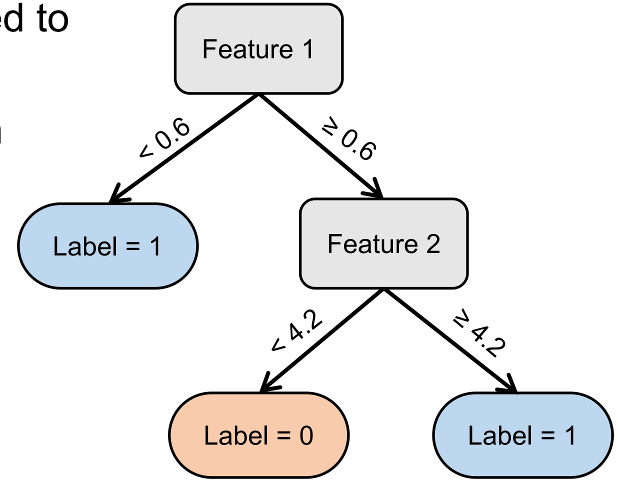
- Regression
  - Task: predict the corresponding value (typically a real number) or target
    - E.g.: you want to predict a person's future income based on their high school GPA



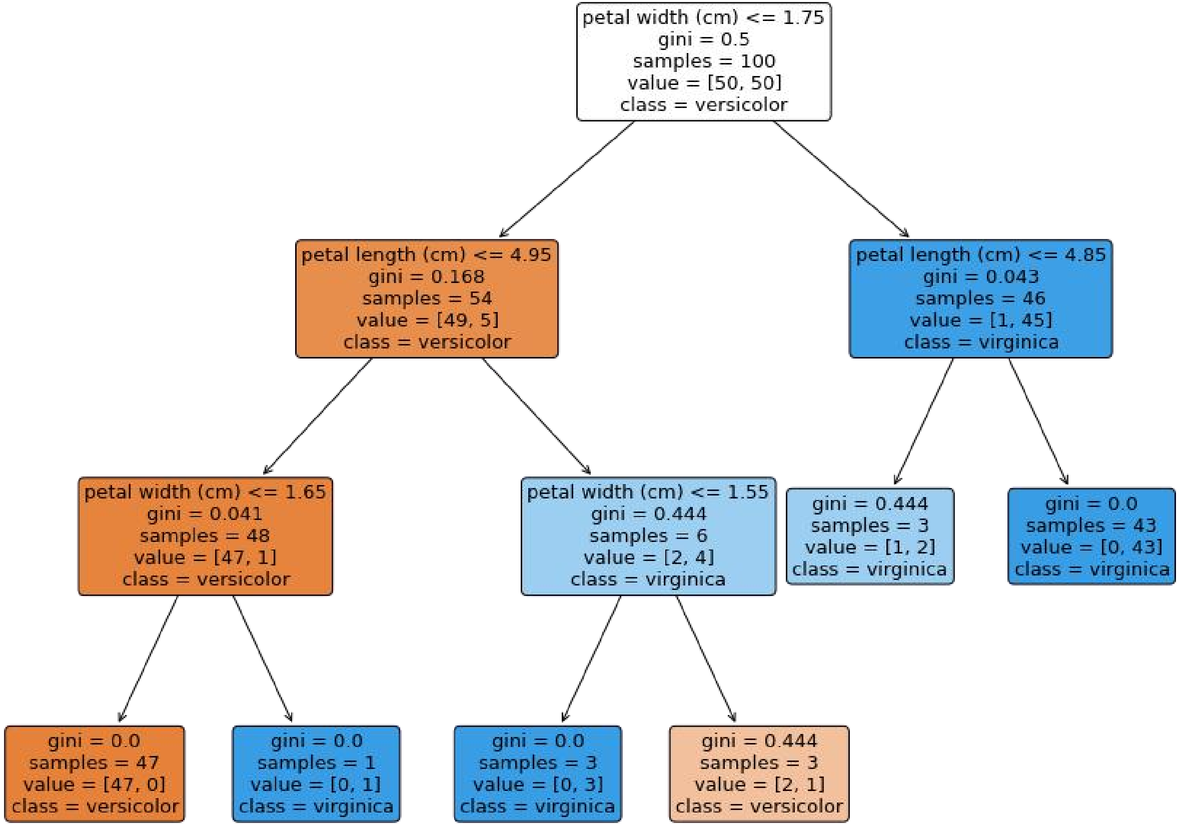
- Others:
  - Sequence-to-sequence, similarity learning/metric learning, learning to rank, etc.

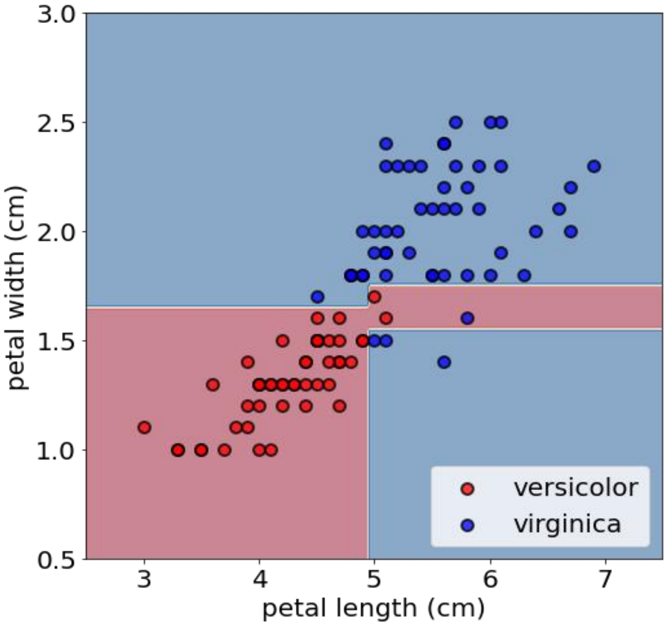
Reminder: Decision Trees

- A decision tree is
  - An *acyclic* graph (i.e., a directed rooted tree) that can be used to make predictions
  - Nonparametric model suitable for classification or regression
    - What is the other nonparametric model we have seen?
  - Prediction:
    - Start at the root
    - Traverse the tree (branching according to feature values)
    - The leaf gives the predicted label or value/target
  - How is the tree constructed from the training data?
    - There are many algorithms and many different kinds of decision trees!



Reminder: Decision Tree Example





Reminder: Overfitting & Regularization

- Decision trees make almost no assumption about the data
  - So unless we control complexity, the tree structure will be made to (over)fit the data!

- How do we avoid overfitting?
  - Prevent the tree from growing too deep (e.g., set a maximum depth)
  - Restrict splitting (e.g., set a minimum number of examples to split)
  - Pruning: after the tree is created, prune branches that do not significantly reduce the error

- Regularization hyperparameters
  - Example: Scikit-learn CART
    - `max_depth`: maximum depth of tree (default = unlimited)
    - `min_samples_split`: minimum number of examples to split a node (default = 2)
    - `min_samples_leaf`: minimum number of examples for a leaf (default = 1)

Ensemble Learning

- Motivating example:
  - We have the following models (all for the same prediction task on the same data)
    - Model A: Linear SVM — 79% accuracy
    - Model B: SVM with RBF kernel — 81% accuracy
    - Model C: Nearest Neighbor (k=3) — 76% accuracy
    - Model D: Logistic Regression — 80% accuracy
    - Model E: Decision Tree — 78% accuracy
  - Which model do we choose?
    - All of them? This is called an ensemble model.

- Why would using multiple models be better than using just one?
- How do we combine the models?

Ensemble Learning: Voting

- Voting Classifiers
  - Given a feature vector, use every classifier to predict the label
  - Hard voting: predict the majority label (statistical *mode* of all predictions)
  - Soft voting: predict label with highest probability averaged over all classifiers
    - Only feasible for models that compute/estimate probabilities

- Voting Regressors
  - Predict the average (or median) of the prediction of all the regression models

- Q: Why would the average (or majority) of many models be better than a single model?
- Q: Is it better if the models are different or similar?
- Q: Do the models have to be any good?

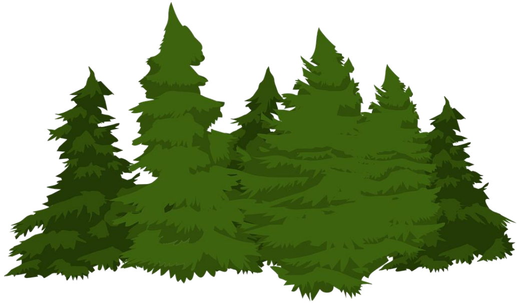
Ensemble Learning: Bagging

- Bagging (**bootstrap aggregating**)
  - Idea: train many different models of the same type
    - Each model is trained on a randomly sampled subset of the data
    - Bagging: sampling with replacement
    - Pasting: sampling without replacement
  - Typically applied to decision trees,
    - But you could apply it to any type of classifier/regressor
  - Aggregation function:
    - For classification: statistical mode
    - For regression: average
  - Effect: lowers variance (and reduces overfitting) but bias is similar
  - Variants
    - Random subspaces: pick random subsets of features (instead of examples)
    - Random patches: pick random subsets of features **and** of examples

Random Forests & Extra Trees

- Random Forests
  - Bagging with Decision Trees
  - But: when constructing a decision tree, use a random set of features to decide on best split!
    - This gives you more diversity among trees, which means an overall better model

- Extremly Randomized Trees (Extra Trees)
  - Like random forests but even more random! How?
    - When building the decision trees, pick a random threshold!



- Note: there are other types of tree-based ensembles
  - Isolation Forests: used mostly for anomaly detection
  - Embedding w/ Random Trees: for unsupervised representation learning

(Hypothesis) Boosting

- Boosting
  - Combine many weak learners into a strong learner
  - Typically, we train the weak learners sequentially (each learn corrects errors of the previous one)
  - Many variants but most popular are Gradient Boosting and AdaBoost

- Gradient Boosting
  - h_0: base model (e.g., decision tree regressor)
  - h_1: model trained on residual errors of h_0
  - h_2: model trained on residual errors of $h_0 + h_1$
  - Prediction: $h_0(\boldsymbol{x}) + h_1(\boldsymbol{x}) + h_2(\boldsymbol{x})$

- AdaBoost
  - h_0: base model (e.g., decision tree classifier)
  - h_i: model but with weights of examples misclassified by $h_{i\text{-}1}$ increased

Stacking

- Stacking
  - Alternative to voting classifiers / regressors
  - Instead of averaging or majority voting, we train a model to do the aggregation
  - Meta model (blender) is trained to combine the predictions

  - Best practice:
    - Split the training data into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$
    - Use $\mathcal{S}_1$ to train the models
    - Make predictions on instances of $\mathcal{S}_2$
    - Then use these predictions and the true labels/targets in $\mathcal{S}_2$ to train the meta model

Mixture of Experts (MoE)

- Recently reinvigorated idea in the deep learning era
  - MoE is now used often for state-of-the-art Transformers / Large Language Models (LLMs)
    - OpenAI's GPT-4 is believed to be a MoE
    - Mixtral 8x7B (from Mistral AI) is a MoE with 8 experts (7B parameters each)
- MoE Idea: divide-and-conquer strategy
  - Train a bunch of experts $f_1, f_2, \ldots, f_k$ (each expert model is usually a neural network)
  - Weighting (aka gating) function $w_1(\boldsymbol{x}), w_2(\boldsymbol{x}), \ldots, w_k(\boldsymbol{x})$, that **depends on input $\boldsymbol{x}$**
    - The weight $w_i(\boldsymbol{x})$ is the weight of $f_i(\boldsymbol{x})$ — influence of expert i on the final prediction
    - Note: this is also learned from data (trained)
  - At inference time given input $\boldsymbol{x}$, the prediction is: $\sum_i w_i(\boldsymbol{x}) f_i(\boldsymbol{x})$
- Many Variants:
  - Sparsely-gated MoE: use only the top 1 (or 2) experts according to the gating network

Next Time

- Friday (2/9): Exercise

- Upcoming:
  - Homework 2 will be out soon (due 2/14) by 11:59pm