CCN
Machine Learning Workshop

Decision Trees

- Tree like structure
- Very powerful supervised machine learning tool, which works for both classification and regression
  - Classification: discrete values for each observation (categorical, ex cat vs. dog)
  - Regression: continuous values for each observation (housing price)
- Algorithms:
  - CART (Classification and Regression Trees)
  - ID3 (Iterative Dichotomiser 3)

CART

- CART (Classification And Regression Tree): an algorithm of Decision Tree, can also handle continuous data.
- It uses Gini impurity to decide how to split the tree. Gini=0 means all data in the specific subset are from the same class.

CART

- CART (Classification And Regression Tree): another algorithm of Decision Tree, can also handle continuous data.
- It uses Gini impurity to decide how to split the tree. Gini=0 means all data in the specific subset is from the same class.

  - Classification Tree:

    Gini Impurity $= \sum_{k=0}^{n} P(x_i) * (1 - P(x_i))$ **= 1 -** $\sum_{k=0}^{n} P(x_i)^2$

    Cost function: $J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$ (we need to minimalize cost function)

    ($P(x_i)$:proportion of x when it belongs to class i,

    k: a single feature used to split the tree, t_k: threshold for the splitting condition)

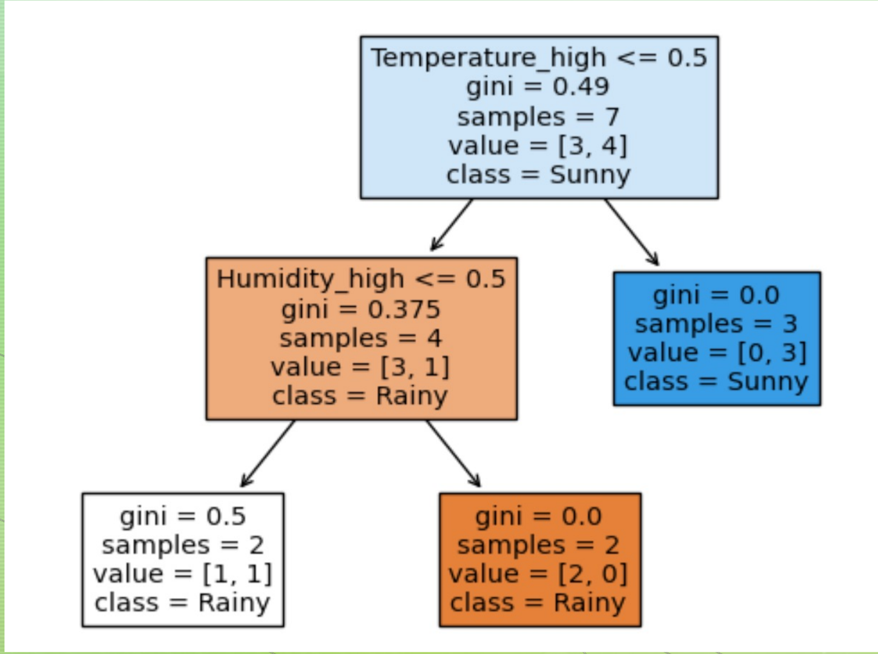
  - Regression Tree:

    Mean Squared Error: $\frac{1}{m} \sum_{i=0}^{m} (y_i - \hat{y})^2$

    Cost function: $J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}$ ($\hat{y}$: prediction value)

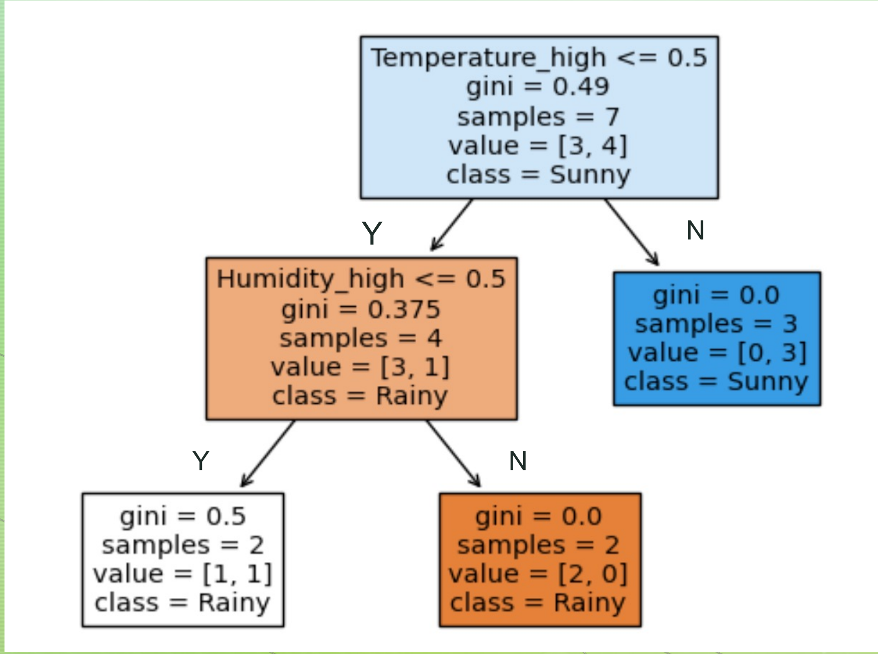
Decision Tree

Temperature	Humidity	Windy	Label
High	Low	Yes	Sunny
Low	High	Yes	Rainy
High	Low	No	Sunny
High	High	Yes	Sunny
Mild	Mild	No	Sunny
Mild	High	No	Rainy
Low	Mild	Yes	Rainy



CART

Temperature	Humidity	Windy	Label
High	Low	Yes	Sunny
High	Low	No	Sunny
High	High	Yes	Sunny
Mild	Mild	No	Sunny
Mild	High	No	Rainy
Low	High	Yes	Rainy
Low	Mild	Yes	Rainy



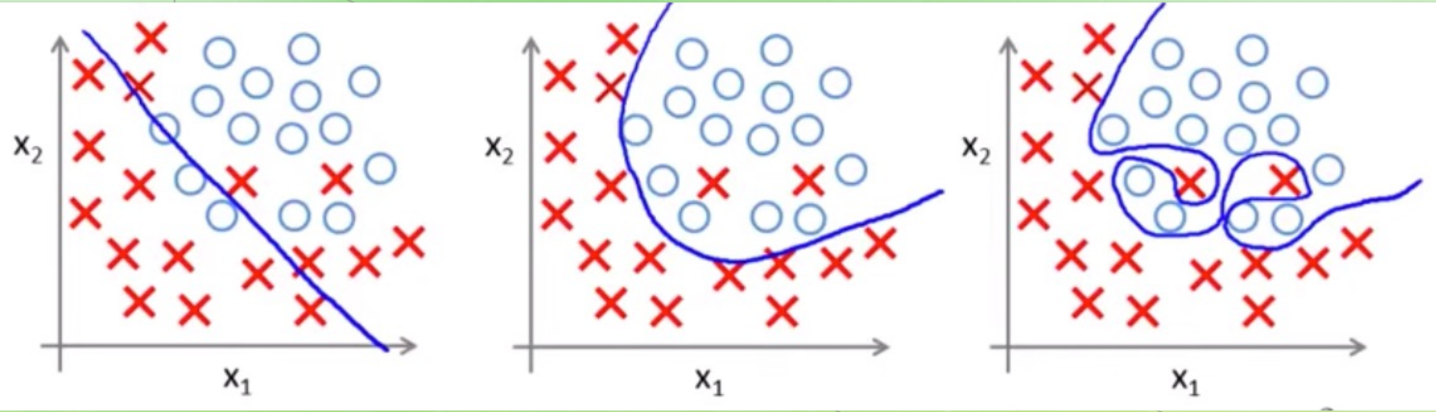
CART

Temperature	Humidity	Windy	Label
High	Low	Yes	Sunny
Low	High	Yes	Rainy
High	Low	No	Sunny
High	High	Yes	Sunny
Mild	Mild	No	Sunny
Mild	High	No	Rainy
Low	Mild	Yes	Rainy

	Temperature_high	Temperature_low	Temperature_mild	Humidity_high	Humidity_low	Humidity_mild	Windy_no	Windy_yes
0	1	0	0	0	1	0	0	1
1	0	1	0	1	0	0	0	1
2	1	0	0	0	1	0	1	0
3	1	0	0	1	0	0	0	1
4	0	0	1	0	0	1	1	0
5	0	0	1	1	0	0	1	0

Model's Over fitting and Under fitting

- Left: Underfitting (High bias)   Middle: Good model   Right: overfitting (High variance)



Reference: Andrew Ng Machine Learning

Decision Tree

- Advantages
  - ❑ Easy to understand
  - ❑ Can handle both classification and regression, less data preparing needed
- Disadvantages
  - ❑ Can be unstable depends on the dataset
  - ❑ Prone to high Variance (overfitting)
  - ❑ CART uses Greedy algorithm which might not be the best result
- For solving overfitting problem:
  - Tuning parameters(max_depth, min_samples_split, min_samples_leaf), prune branches

Random Forest

- Ensemble model
  - ❑ Use several different machine learning models or algorithms
  - ❑ By combining multiple prediction results to achieve higher accuracy
  - ❑ Each algorithm should vary in the examples they misclassify
- Random Forest algorithm
  1. Randomly resampling data in size M with repeated values to form a new dataset
  2. Use the new dataset to train N Decision Trees
  3. Randomly select a subset of features for each decision tree
  4. The final prediction will be achieved by voting from all N decision trees

Boosting

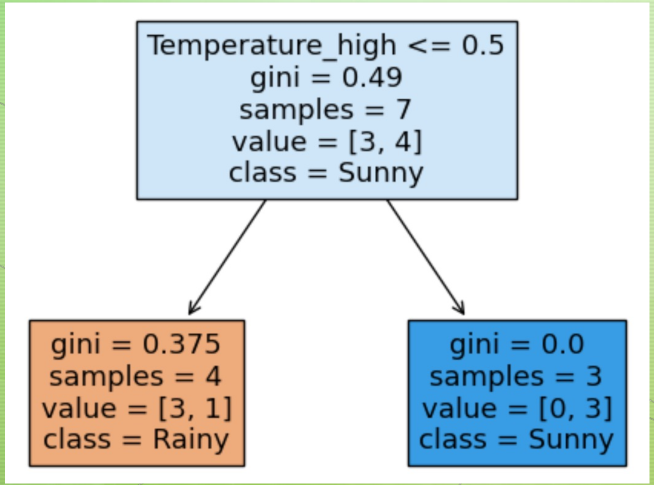
- What is Boosting
  - A type of ensemble model, which uses multiple weaker classifiers
  - Uses interactive steps to add new models
  - Some focus on training model on data mistakenly classified previously (e.g. by adding weights on them)
- Different Boosting algorithms (classifier and regressor)
  - AdaBoost (Adaptive Boosting)
  - GradientBoost
  - XGBoost (Extreme Gradient Boosting)

AdaBoost

- Adaptive Boosting
  - Generate first tree by picking the feature with least Gini when splitting the tree
  - Use Tree stumps (a node with only two leaves) as the base model by default
  - Models made correct classification get higher weights (higher weight to influence the final result)
  - Data being incorrectly classified get have higher weights in the next round while the rest get smaller weights(total weights still equals to 1) .
  - Keeps adding new trees in the same way until the number of the trees reach to what's setup in n_estimator.
- Tuning parameter
  - n_estimators
  - learning_rate
- Python Libraries

  from sklearn.ensemble import AdaBoostClassifier

  from sklearn.ensemble import AdaBoostRegressorr



Gradient Boosting

- How It works
  - Start with a naïve initial prediction, such as average of all data $\hat{y} = \bar{y}$
  - Take residuals of each observed $y_i$and predicted $\hat{y}$: **r** = $y_i - \hat{y}$
  - Generate a new Decision Tree with size as 8 to 32 leaves and apply the residuals to the tree
  - Calculate cost function (similar to means square error) to minimize the sum of residual in each tree
  - Update the predicted $\hat{y}$ value by adding the previous $\hat{y}$ + learning_rate x (reduced_residual)
  - The new tree uses new prediction to calculate residuals, so the later trees improve on the error of previous trees over time
  - Repeat these steps on the number of trees you decide to use (n_estimators)

- Tuning parameters
  - n_estimators
  - learning_rate (prevent overfitting)
  - maxdepth

XGBoost

- The name is from Xtreme Gradient Boosting. Why Extreme?
  - Use Greedy algorithm and quantile sketch to speed up the calculation
  - Use learning rate, lambda, alpha, gamma to regulate the tree to avoid overfitting.
  - Can handle vary large dataset and missing data
  - Can run parallelly, also run on GPU
  - More cool functions such as early stop function, etc

- Tuning parameters
  - n_estimators
  - learning_rate : Maximum depth of a tree
  - reg_lambda, reg_alpha: L2 and L1 regularization term on weights. Increase this value to avoid overfitting
  - subsample : ratio of the training instances
  - colsample_bytree : subsample ratio of columns when constructing each tree
  - max_depth
- References: [XGBoost Official website]

Parameter Tuning

- GridSearch

  from sklearn.model_selection import GridSearchCV

- Random Search

  from sklearn.model_selection inport RandomizedSearchCV

EEG data

- Eye opening and closing
- Split the data
  - Randomly split the data into training and testing
- Compare results of different algorithms
  - DecisionTree
  - RandomForest
  - Adaboost, GradientBoost, XGBoost

Jupyter Notebook

- Use in Hoffman2
  - Download IDRE team script:

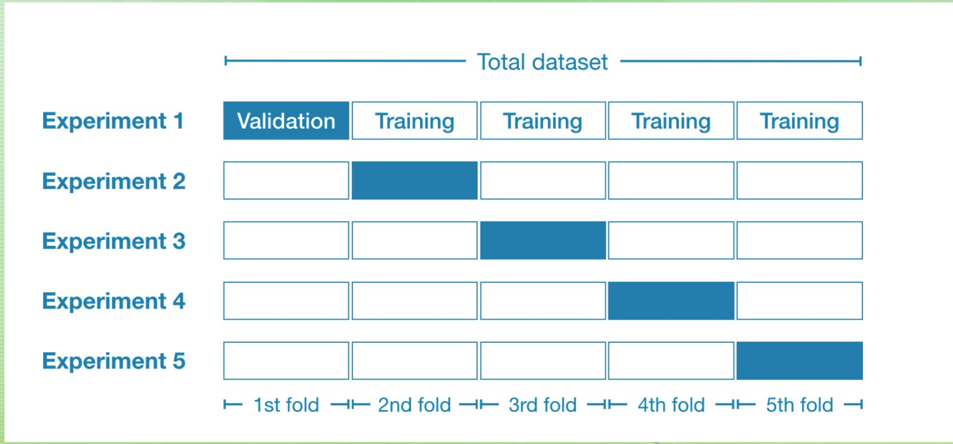
    wget https://raw.githubusercontent.com/rdauria/jupyter-notebook/main/h2jupynb
  - Run command from your laptop/desktop

    ~/h2jupynb -u USER_ID -t 3 -m 8 -p 9394
  - To use specific Conda environment

    pip install ipykernel
    python -m ipykernel install --user --name=MYCONDAENV
- Install Python in your laptop
  - Download and install Anaconda or Miniconda
  - Pip install Pandas, scikit-learn and matplotlib (or use conda env)
  - Then type in "Jupyter Notebook" from your terminal
- Colab
  - Requires Google account
  - Good for testing, using CPU node only. GPU time is limited.

Cross Validation

- A method to split data into training and testing/validation (for small and medium size dataset)



- Reference: Kaggle

Our future ML workshops

Neuroimaging Machine Learning projects

Please let me know if there's any question, feedback.

haiyanwang@mednet,ucla.edu