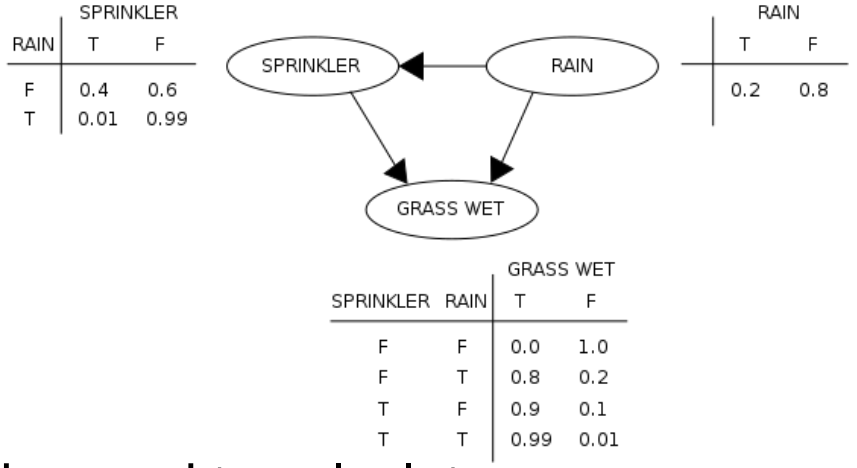
Week 4 : Trees

http://cornins.com/tip-of-the-month/taking-care-trees/

Week 3 Bayes Review

- What is Bayes' Law?
  -
- What is a Bayesian Network?
  -
- How is Naive Bayes used to predict the class of text documents?
  -

Week 3 Bayes Review



- What is Bayes' Law?
  - P(A|B) = P(B|A) P(A) / P(B)
- What is a Bayesian Network?
  - A Directed Acyclic Graph (DAG) that can be used to calculate probabilities for any variable from any other set of variable conditions in a dataset
    - Wikipedia example is fairly easy to understand
- How is Naive Bayes used to predict the class of text documents?
  - Multiply the probability of seeing each word in a doc for a given class, then multiply by the class probability. Take the highest probability as the predicted class

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Refresh - basics

- Supervised/unsupervised/reinforcement
  -
- Classification/regression
  -

Refresh - basics

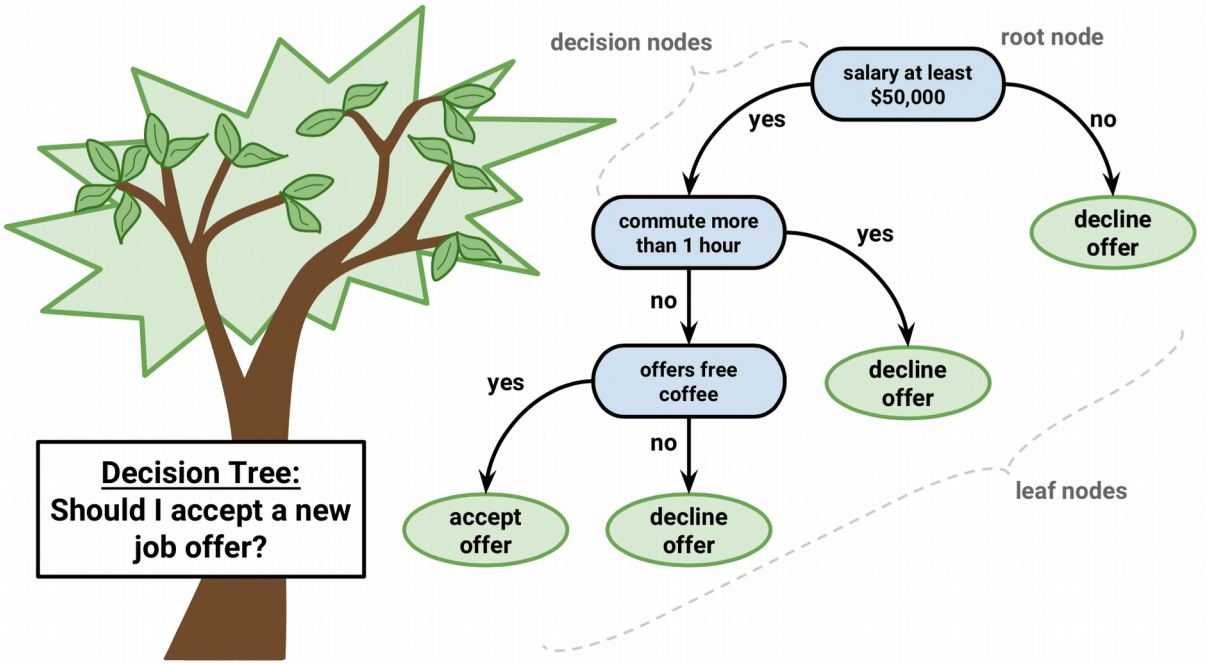
- Supervised/unsupervised/reinforcement
  - Supervised = we have targets we are trying to predict
  - Unsupervised = just data, and we don’t know the classes they should be in
  - Reinforcement = improving algorithm behavior over time by learning from feedback
- Classification/regression
  - Classification is for factor variables (limited number of classes), regression is for continuous (infinite range of numbers possible)

Week 3 review quiz

- Use a Naive Bayes model to predict the iris classes
- Create train/test sets with caret and data.table
- Use expand.grid() to search a few hyperparameter settings
- Show the best hyperparameter settings and use this model to predict on the test set
- Display the confusion matrix and interpret the results
- My solution was 24 lines with a few lines of comments
- You can just use a .R file, don’t have to do .Rmd

Decision trees

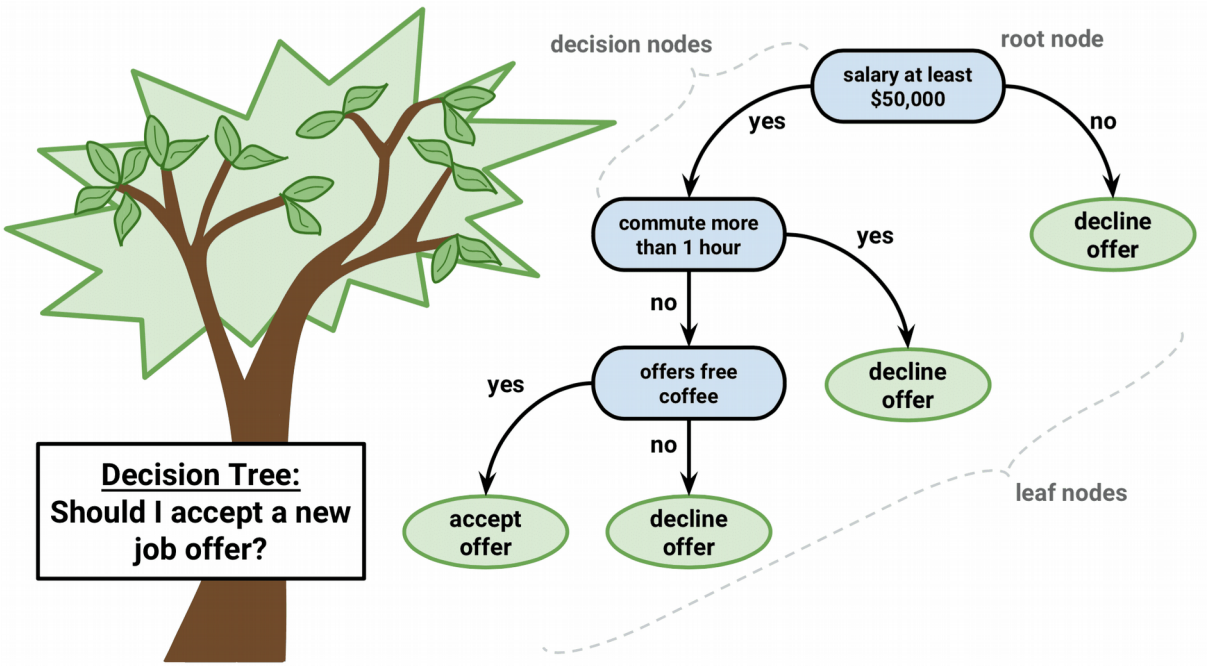
- Simple idea, use thresholds to filter data left or right



http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

Terminology

- Starts at the root and ends with leaves
- A stump is a 1-level decision tree (a single split)
  - Each row in the tree is a level



- But with real data, how do we decide what to split on?

Optimizing splitting

- some algos for splitting: ID3, C4.5, CART (there are more...)
- ID3/C4.5/C5.0 are used for classificitaion. C4.5 is an improvement on ID3, C5.0 is an improvement on C4.5, but C5.0 is not easy to use...(http://www.rulequest.com/)

Classification And Regression Trees (CART)

- “Greedy” algorithm, meaning it picks the best choice ASAP
- For regression, uses SSE = (y – y.hat)^2
- y.hat (prediction) is average of samples in that leaf
- For classification, uses Gini impurity (Gini index), which is basically class purity

Gini impurity (classification)

$$G = 1 - \sum_i P_i^2$$

- P_i is proportion of class i, so if we have all our data points in a leaf node as one class, G = 0.
- If we have 2 classes and it is a 50/5 split, then

  1 – (0.5^2 + 0.5^2) = 0.5
- 3 classes and 1/3 1/3 1/3 split:

  1 – (0.33^2 + 0.33^2 + 0.33^2) = 2/3

- For each split, try all splits of all features and pick the one that gives best Gini impurity measure (lowest number)
  - Can settle in local minima with this “greedy” method, but could use backtracking to overcome this

Entropy/Information Gain (classification) (ID3/C4.5/C5.0)

$$H = \sum_i -p_i \log_2(p_i)$$

$$IG = H_{oldNode} - \sum_{newNode} p_{newNode} H_{newNode}$$

- H is entropy – 50/50 split is -1
  - log_2() of anything < 1 is negative and increases as we approach 0
  - If we have two classes (50/50) then split to 90%/10%, entropy is now -0.4689956, so IG is around 0.53. Higher IG is better
  - H = 0 is perfect classification – log(1) = 0

Single decision tree problems

- decision.tree.demo.R file under week 4
- Single decision trees often overfit (high variance) because they are fitting the training exactly
- We can combat this with hyperparameters
  - Require minimum number of samples in each leaf node (minsplit in rpart)
  - Require our scoring metric improves by a minimum threshold on each split (cp in rpart)
  - maxdepth will limit the number of splits we can have

Bagging

- Use bootstrapping to decrease variance
- Sampling with replacement; build multiple trees on different bootstrapped samples
  - Can score the model on the ‘out-of-bag’ (oob) samples; each model evaluates the oob samples separately, and the score is averaged

Random Forests

- Bagging, but we also randomly subset the features for each split
- Each tree is called a ‘weak learner’ because it has high bias, and didn’t learn much about the data
- This allows us to tune the number of features we select from, so it can decrease variance of models
- Lots of hyperparameters:
  - Number of trees, max depth, min samples to split, number of features to use for each tree

Random Forests example

- rf.trees.example.R file under week 4
  - number of trees – more trees +variance or +bias?
  - max depth – deeper +variance or +bias?
  - min samples to split – +samples +var or +bias?
  - number of features to use for each tree – more features +variance or +bias?
- We are going to find out with the heart disease dataset

Boosting

- Gradient boosting (xgboost) is probably the most powerful tree-based method
  - Random forests are probably next in line

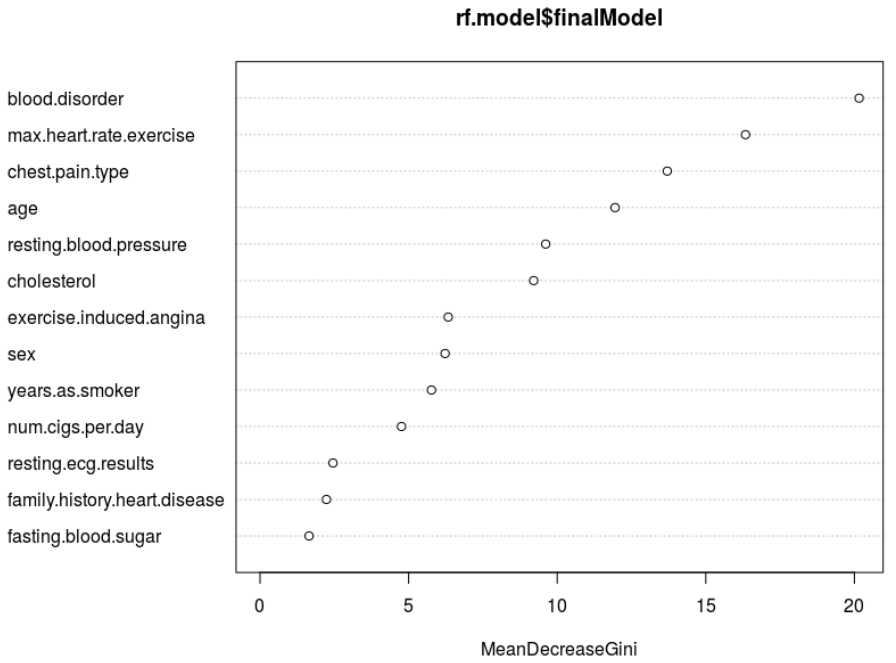
http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/

Boosting

- Fit a tree model (weak learner) to the data, F_0
- Then calculate the differences between the data and the predictions (residuals): h(x) = y – y.hat
- Fit a model to the residuals (h(x)). Ideally, h(x) would equal y – y.hat, and we would be done, but of course this never happens
- Then add the new model, F_1 (h(x)) to the current model with a learning rate, gamma. F(x) = F_0 + gamma * F_1 + …
- Rinse and repeat until we decide to stop (hit a number of trees)
- Adaboost does this with stumps, xgboost has lots more hyperparameters

Feature importances

- We can find on average which features give us the best splits (most information gain, or most reduction of Gini impurity, or most gain in R^2)
- This allows us to see which features have more predictive power



Example: classification on heart disease dataset

- Optional xgb.demo.R file under week 4 (xgboost)

Project: Trees on bank marketing dataset

- Data is here: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing
- Or under week 4 (same as downloading from UCI)
- Make at least 2 EDA plots and explain them.
- Use either random forests and/or xgboost to fit a model to the data, predicting if someone signed up (or will sign up) for a term deposit or not. Provide any possible points of improvement on the model if you don't think the one you have is optimal.
- Make a plot of the feature importances from your model and explain it.
- Report at least 2 scoring metrics (e.g. accuracy, confusion matrix, ROC/AUC score and curve, etc) and explain them.

Taking it further

- Go through the xgboost demo, and use xgboost on some data
- Try the same things in Python
- Try adaboost
- Try getting out of bag (OOB) scores for random forests