Lecture 9: Predictive Modeling

Predictive Modeling Framework

- In many situations, our goal is to describe a scientific process or phenemenon. To do so, we fit a model, which we hope is a reasonable approximation to reality that allows inferences. In this situation, simplicity or parsimony is important.

- In other situations, the goal is to predict (or forecast) future values. Generally, in these situations a prediction (point estimate or distribution) are generated for upcoming scenarios, but the underlying model parameters are not of interest.

- In other situations, the goal may seem to be prediction, but there is still a desire to describe the underlying scientific process. Resolving this requires clear communication.

Loss Functions

- With the prediction setting, often times the problem will come with a specific criteria that we seek to minimize.

- For instance when predicting continuous quantities,

- With categorical, or binary data, a zero-one loss is common where there is zero loss for a correct prediction and a loss of one for an incorrect prediction.

- In both situations, you can also have non-symmetric loss functions where one kind of prediction error is more "costly".

- The loss functions that we previously discussed are focused on point predictions, but they can also evaluate predictive distributions or prediction intervals. One example is the Kaggle March Madness prediction competition, where rather than a prediction for each basketball game, contestants produce a winning probability.

  Then the log loss function is used: $y \times log(p) + (1 - y) \times log(1 - p)$, where y is the binary response.

Test / Training and Cross - Validation

- An important concept in predictive modeling is overfitting. Which enough parameters in a model, it is possible to get perfect predictions.

- Two common approaches for evaluating predictive models are cross-validation and the test/training set.

- With the test / training set,

- Cross-validation partitions your data into k groups. Then all of the observations in the first $k-1$ groups are used to fit a model and predictions are made for the k^{th} group. This process continues until all of the observations have been predicted once (and used in the model fitting $k-1$ times).

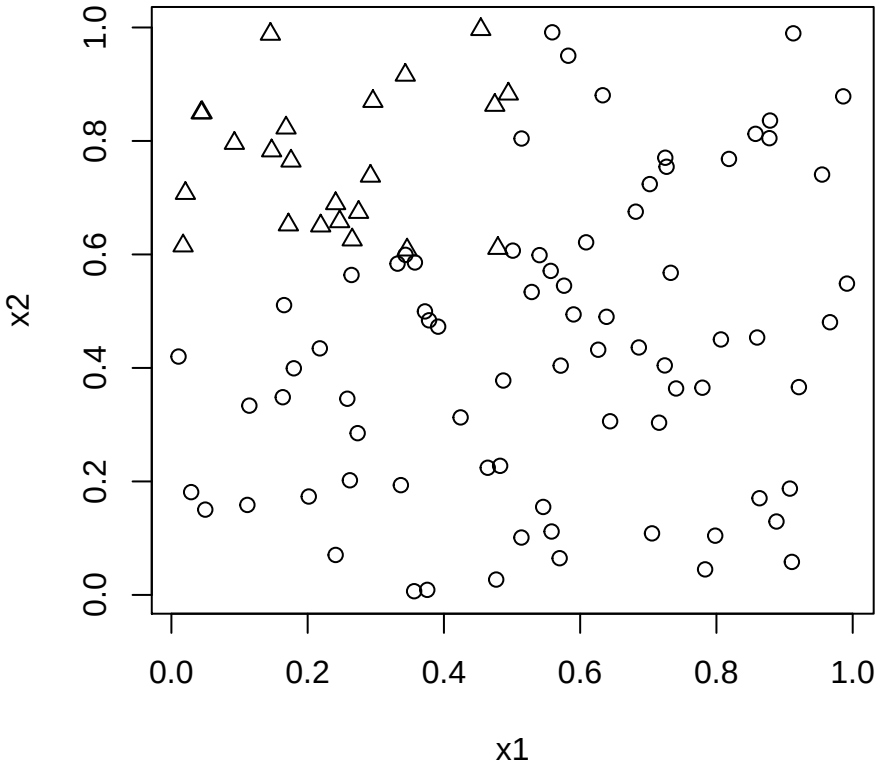
Tree Methods

- "Statistical Modeling: The Two Cultures" by Leo Breiman https://projecteuclid.org/download/pdf_1/euclid.ss/1009213726, which focuses on "algorithmic models" that do not have an underlying probabilistic or data generating mechanism.

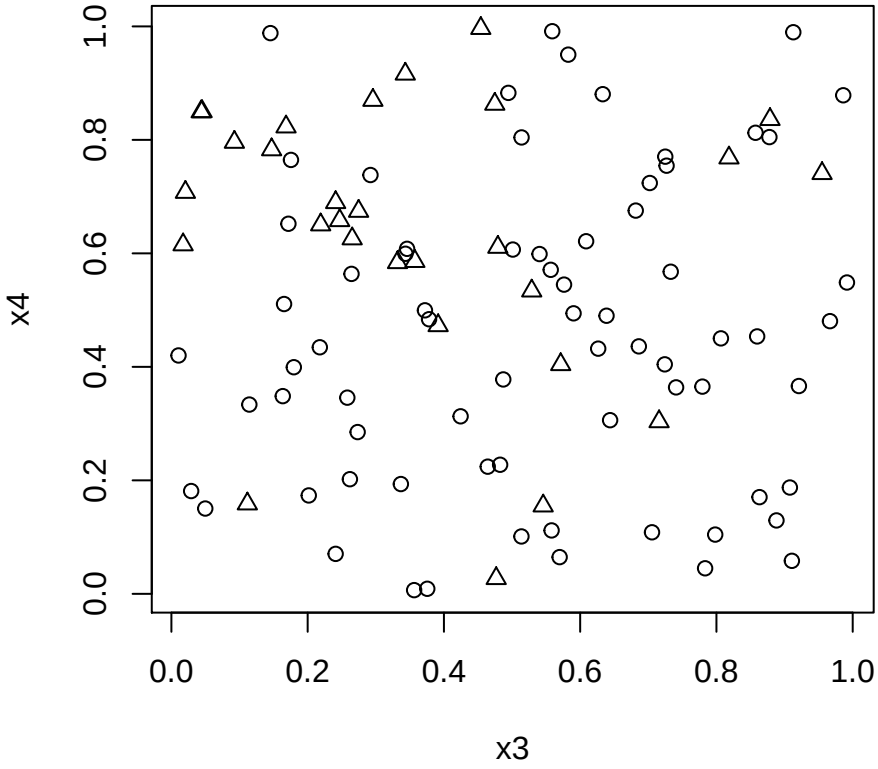
- One such example of an algorithmic model is a decision tree - also known as Classification And Regression Tree (CART).

- Classification and Regression Trees partition the predictor space into distinct sets that have different models or predictions.

- Consider the figure below and write out a statistical model for predicting triangles or circles *or* sketch a boundary.

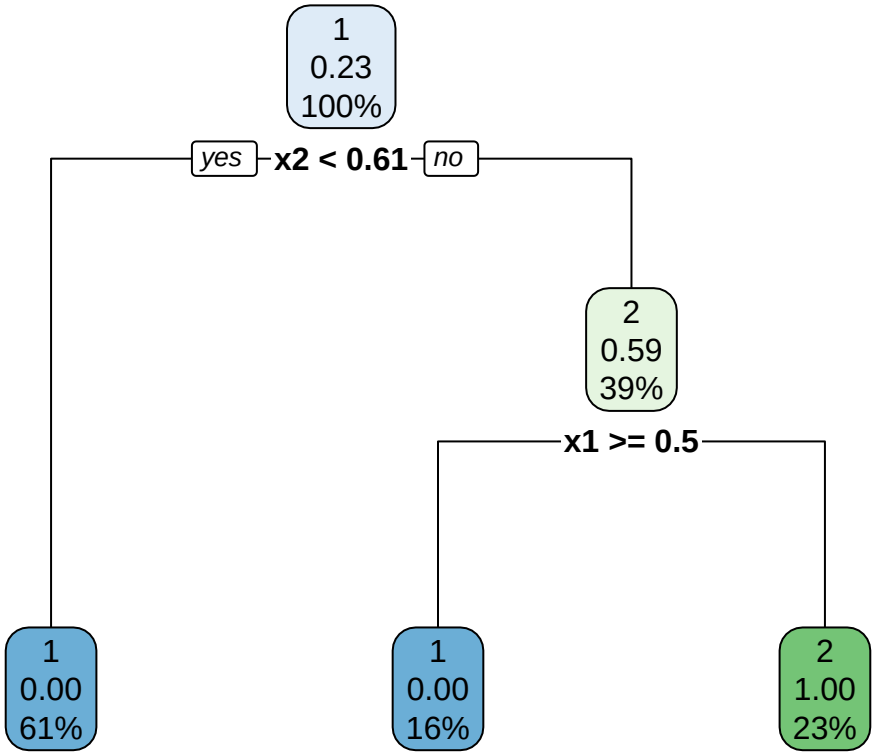


- Now, how about the figure below?

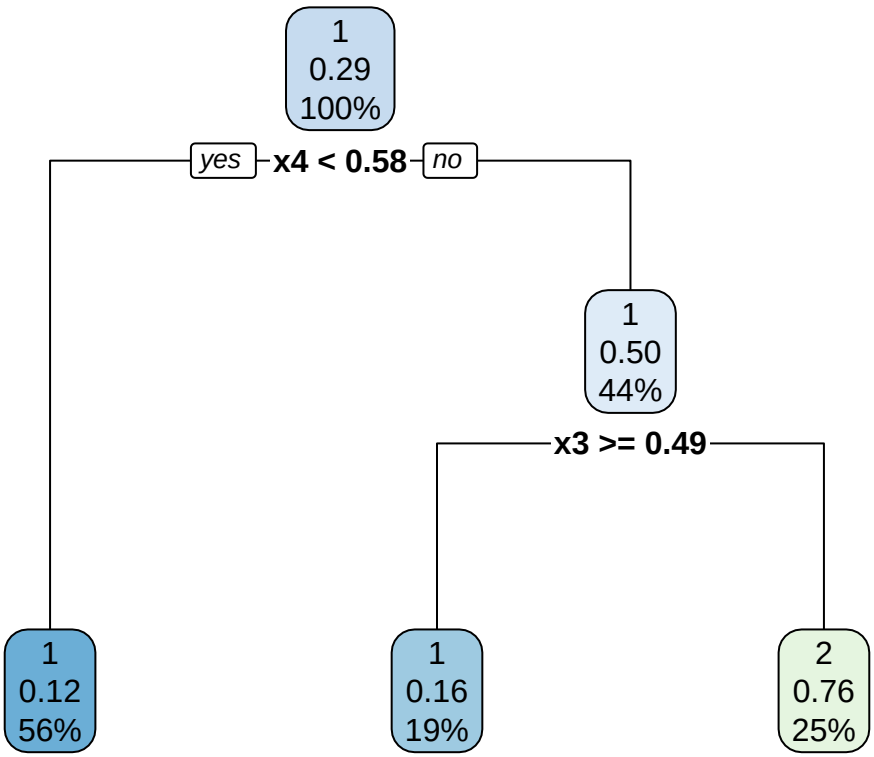


- Using R, we can fit a decision tree with `rpart`.

```
tree.df <- data.frame(y1 = point.type1, x1 = x1, x2 = x2,
                      y2 = point.type2, x3 = x3, x4 = x4)
tree1 <- rpart(y1 ~ x1 + x2, data = tree.df, method = 'class')
rpart.plot(tree1)
```




```
tree2 <- rpart(y2 ~ x3 + x4, data = tree.df, method = 'class')
rpart.plot(tree2)
```



- These examples were both classification trees, where a value of one or zero is predicted for each partition.
- Regression trees are also possible, where each terminal node would have an indpendent tree.

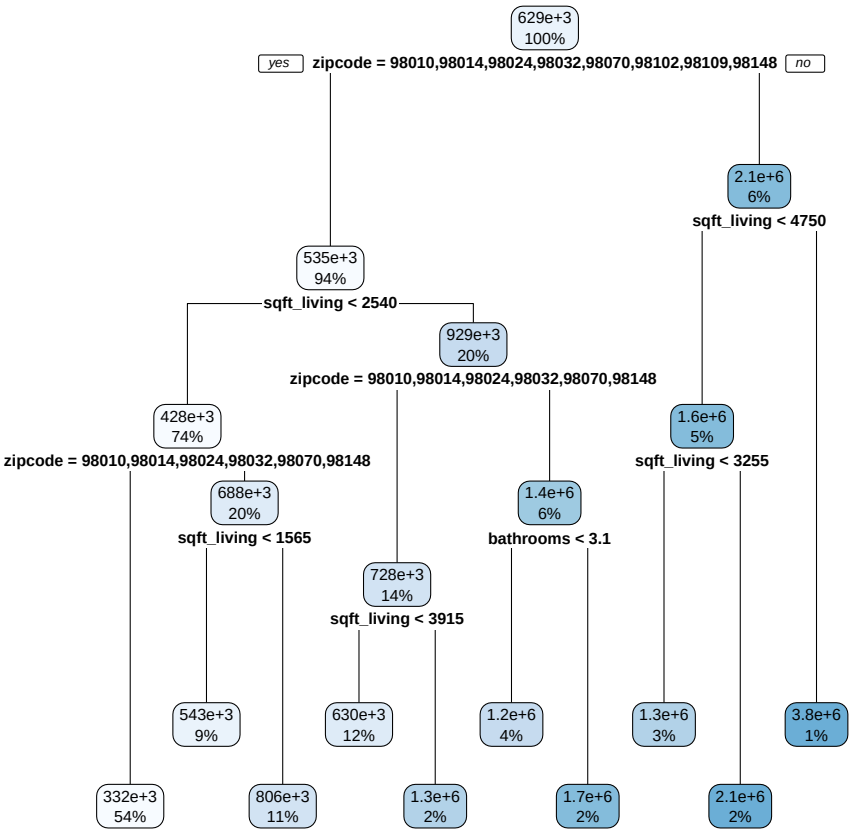
Ensemble Methods

- Ensemble methods combine several *weak* learners to reach a consensus decision.
- Consider a set of independent models that each correctly predict a binary outcome 60 percent of the time.
- Then $Pr[Y = 1|M_1 = 1] = .6$
- $Pr[Y = 1|M_1 = 1, M_2 = 1] = \frac{.6^2}{.6^2+.4^2} =$
- $Pr[Y = 1|M_1 = 1, M_2 = 1, \ldots, M_5 = 1] =$
- A common example of an ensemble method is a random forest. A random forest is composed of a set of decision trees; hence, the forest moniker.
- The random forest attempts to create "pseudo-independent" trees by randomly selecting a subset of the covariates to include and a random samples, with replacement, or the observations.
- In the classification setting, a majority rule algorithm is implemented for a final, consensus choice.
- The performance of a random forest is based on two criteria: the strength of the predictions for each individual tree and the correlation between those trees.

```
set.seed(02182020)
Seattle <- read_csv('http://math.montana.edu/ahoegh/teaching/stat408/datasets/SeattleHousing.csv')
Seattle <- Seattle %>%
  mutate(log_price = log(price), zipcode = as.factor(zipcode))
num_houses <- nrow(Seattle)
Seattle_test_ids <- sample(1:num_houses,round(num_houses * .75))
Seattle_train_ids <- (1:num_houses)[!(1:num_houses) %in% Seattle_test_ids]
Seattle_test <- Seattle %>% slice(Seattle_test_ids)
Seattle_train <- Seattle %>% slice(Seattle_train_ids)
```

```
library(randomForest)
lm_pred <- lm(price ~ bedrooms + bathrooms + sqft_living +
                floors + waterfront + zipcode, data = Seattle_test)
tree_pred <- rpart(price ~ bedrooms + bathrooms + sqft_living +
                     floors + waterfront + zipcode, data = Seattle_test)
rf_pred <- randomForest(price ~ bedrooms + bathrooms + sqft_living +
                          floors + waterfront + zipcode, data = Seattle_test)

rpart.plot(tree_pred)
```




```
mape_lm <- mean(abs(predict(lm_pred, Seattle_train) - Seattle_train$price))
mape_tree <- mean(abs(predict(tree_pred, Seattle_train) - Seattle_train$price))
mape_rf <- mean(abs(predict(rf_pred, Seattle_train) - Seattle_train$price))
```

The mean absolute error for the tree model is

The mean absolute error for the tree model is

The mean absolute error for the random forest model is