Machine Learning for All: Trees

Anastasiya Yarygina

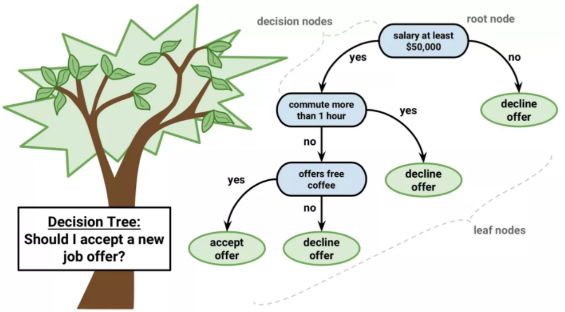
Saturday, January 19, 2019

Trees

In this class we will cover the following topics:

- Decision trees: Using tree-logic to make predictions
  - Regression and Classification Single-tree models
  - Random Forest
  - Boosting
- Examples:
  - Iris, R
  - Boston Housing, Kaggle
  - California Housing Prices, Kaggle

What is a Decision Tree?



Decision Tree classifier, **Image credit:** www.packtpub.com

Tree-logic uses series of steps to come to a conclusion. Each decision is a **node**, and the final prediction is a **leaf node**.

Decision Trees in ML

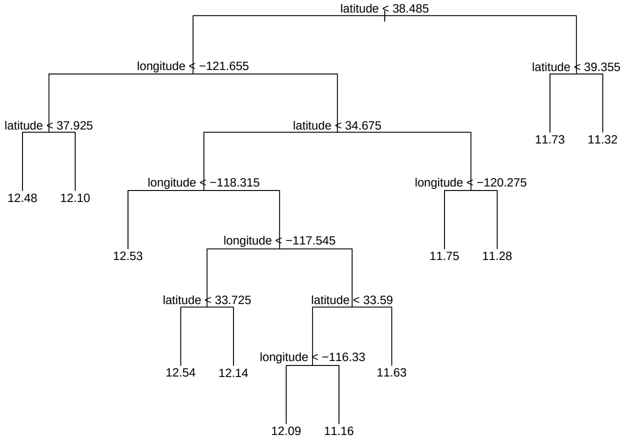
- **Decision Tree Algorithm** is a supervised learning algorithm that can be used for solving
  - **regression** (continuous response variable) and
  - **classification** (categorical or factor response variable) problems.
- Classically, the name of this algorithm is **Decision Tree**
  - Some platforms like R use a modern term **CART** (Classification and Regression Trees)
- Objective: obtain **predictions**
  - of the **reponse variable** Y (dependent variable or output)
  - from the **input variables** X_1, X_2, **. . .** X_n (features, predictors).

Predictions using Decision Trees

- **Key Idea**: Decision Tree **splits** the data into
  - two or more **homogeneous data segments**
  - based on the **best splitter**, which is a variable taken from the inputs X_1, X_2, ... X_n.
- Every time we split the sample we make a decision. Each decision is a **decision node**, and the final prediction is a **leaf node**.

Exmpale: California Housing dataset

We can fit a tree that predicts for each property **log price** using as inputs **longitude** and **latitude**.

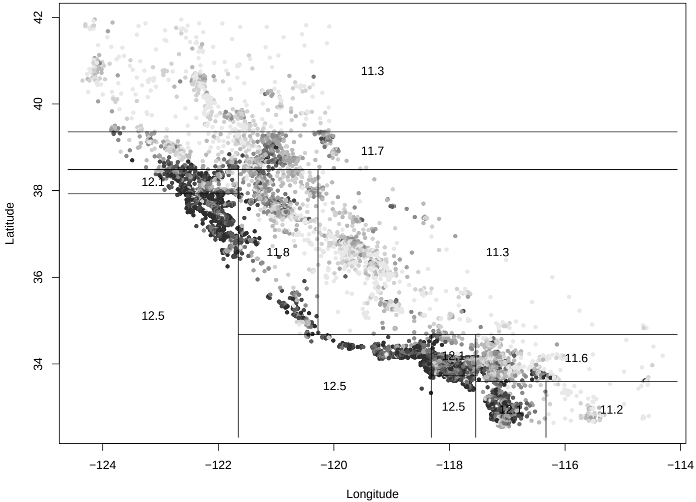


Exmpale: California Housing dataset

- The tree has 11 **decision nodes**, which are the nodes where the splitting of the data takes place.
- And there are 12 **leaf nodes**, which means that the data space was partitioned in to 12 **homogeneous regions**.
- How do these **homogeneous regions** look like?

Exmpale: California Housing dataset

Overlay log price of properties on predicted partitions. Darker dots represent more expensive properties.



Exmpale: California Housing dataset

The tree model divided the **predictor space** (longitude and latitude in this case) into 12 distinct and non-overlapping rectangular **regions** R_1, R_2, . . . ,R_j, . . . R_{12}.

If there are more than two inputs, the data space is split in some kind of hyper-rectangles.

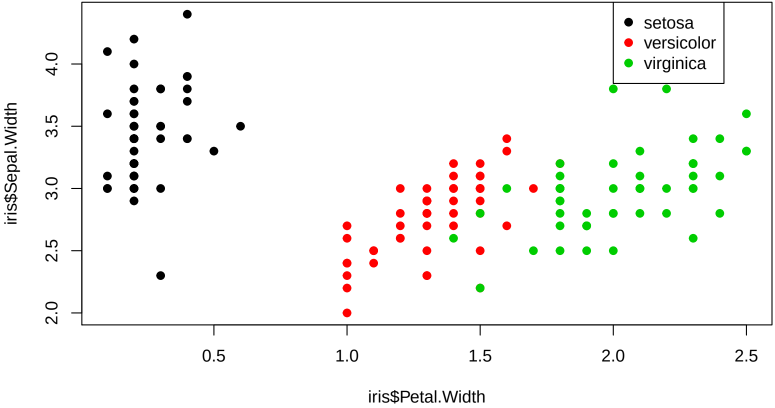
For every observation that falls into a given region R_j, the model assigns its **predicted value**, which is the **mean of the response** Y (log price in this case) for all observations in region R_j.

The regions with the log average value 12.5 are LA and the Bay Area.

Exmpale: *iris* dataset

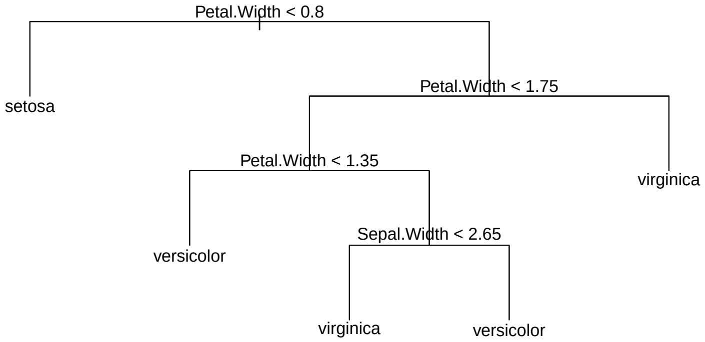
What happens if our problem is a **classification problem**?

iris dataset: sepal and petal length and width, 150 plants, 3 species - Setosa, Versicolor, Virginica.

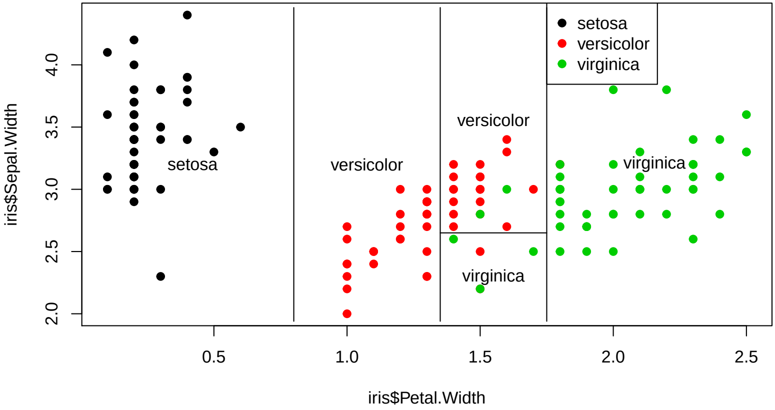


Exmpale: *iris* dataset

Fit a tree model that predicts species using as inputs sepal and petal width.



Exmpale: *iris* dataset



Partitions are defined by the classification tree. The fist node classifies plants with petal width < 0.8 as setosa. Next, all plants with petal width > 1.75 are virginica.

Decision Tree Algorithm

To get homogeneous segments, the model makes **optimal splits**.

Each **optimal split** is made:

- at **certain value of some predictor** X_i,
- so that the child set to the left of the split and the child set to the right of the split are **as homogeneous in response** Y **as possible**.

In regression trees **homogeneity** is measured by the **Sum of Squared Errors** (SSE)[1]:

$sum(y - prediction_{left})^2 + sum(y - prediction_{right})^2$

Each **optimal split minimizes the SSE** to the left and to the right of the split.

[1] Different metrics called **Gini Impurity** is used in classification trees. I found this post about Gini Impurity particularly didactic.

Decision Tree Algorithm

Decision trees are fit in a **top-down**, **greedy** approach, which is also known as a recursive binary splitting.

- **Top-down**: it starts at the top of the tree
- **Greedy**: at each step the best split is made at that particular step, we do not look ahead and pick the split that will lead to a better tree in some future step.

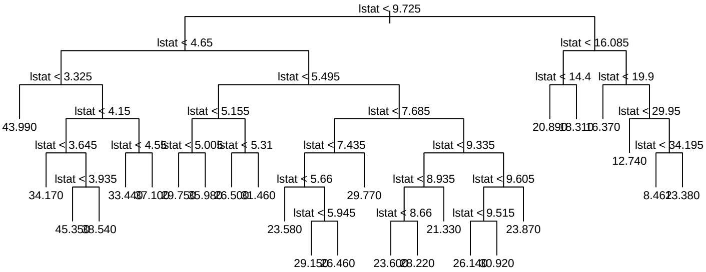
Each split improves the fit of the tree (think of R^2 and adding new variables in a regression model).

The algoritm stops when:

- improvement in the fit is below some predefined threshold (default 0.01)
- number of observations in leafs is below some predefined threshold (default 5)

Practice: *tree* package and Boston Housing dataset

Fit a tree to predict median value of properties using low income status as predictor.




```
## The big tree size is:  26
```

Prunning

Tree models are very flexible and tend to overfit.

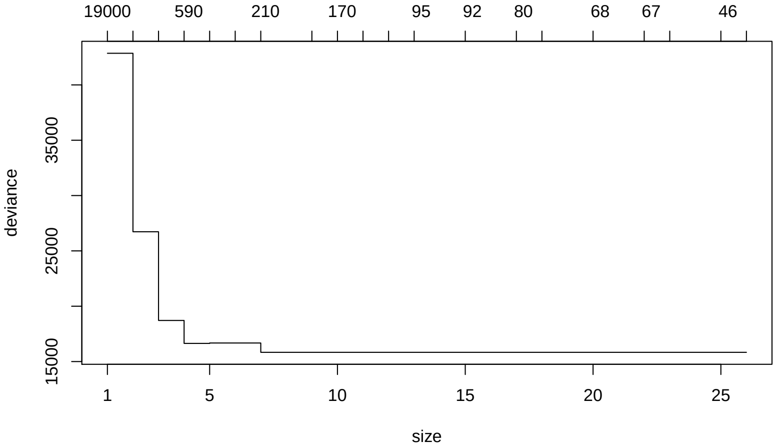
To avoid oferfitting trees are **prunned** by removing nodes and brunches from the bottom up.

At each step we remove the split that contributes least to improvement in the fit.

Pruning produces a set of **candidate trees** of different sizes.

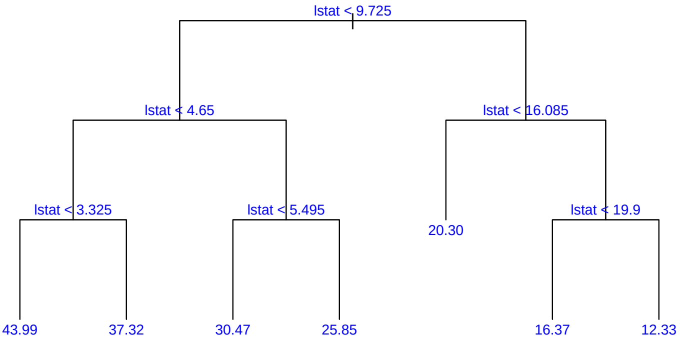
We use **Cross Validation** to choose the tree with the best fit (i.e., with the smallest SSE or smallest deviance).

Practice: *tree* package and Boston Housing dataset




```
## CV and choose the size that minimizes deviance
```

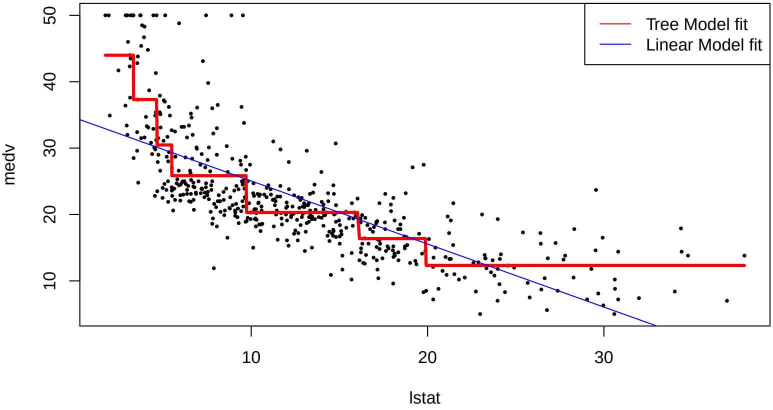
Practice: *tree* package and Boston Housing dataset




```
## the size of the prunned tree is 7
```

Practice: *tree* package and Boston Housing dataset

Compare Tree fit and Linear Model fit



Aggregating Trees: Bagging

- Decision Tree algorithms are effective in choosing a single tree.
- What is better than one tree? Many trees!!!
- To improve predictions we can:
  - fit many tree models from the same data
  - and average predictions across these models.
- This is excatly what **Bagging** (or Bootstrap aggregation) procedures do. The steps are the following:
  - Sample (Bootstrap) B subsets of the data
  - Fit a tree to each subset to get B fitted trees
  - For regression trees: average predictions across trees
  - For classification trees: take the most commonly occurring class

Aggregating Trees: Random Forest

Random Forest is a special case of Bagging. It provides an improvement over bagged trees by way of a small tweak:

- Random Forest builds B trees on bootstrapped samples.
- But, for each split it randomly choose a sample of m predictors of all available p predictors (default $m = \sqrt{p}$).

Random Forest tuning parameters are B and m.

Boosting

Boosting builds many decision trees, but unlike in Bagging, Boosting trees are grown **sequentially**. The steps are the following:

- Fit the model **tree#1** on the original data and save the residuals
- Fit the model **tree#2** on the residuals
- Update the initial model: **tree#3 = tree#1 + tree#2**
- Update the residuals
- Fit a new model **tree#4** on the residuals
- ...
- Repeat the process for a specified number of iterations

Updated trees are **weighted** or **shrunk** by the **shrinkage parameter** λ which controls the rate at which algorithm learns (default = 0.001 to 0.01).

Other tuning parameters: d the number of splits in each tree and B the number of trees to grow.

Takeawyas

- Decision Trees are simple and interpretable predictive models.
- However, they tend to overfit.
- **Ensembling methods** such as Random Forest and Boosting are good for improving predictive capacity of trees. They work growing many trees and combining predictions of the resulting ensemble of trees.
- Random Forest and Boosting are among the sate-of-the art methods for supervised learning. However, they are computationally intensive and their results are difficult to interpret.

Practice: Regression Trees using California Housing data

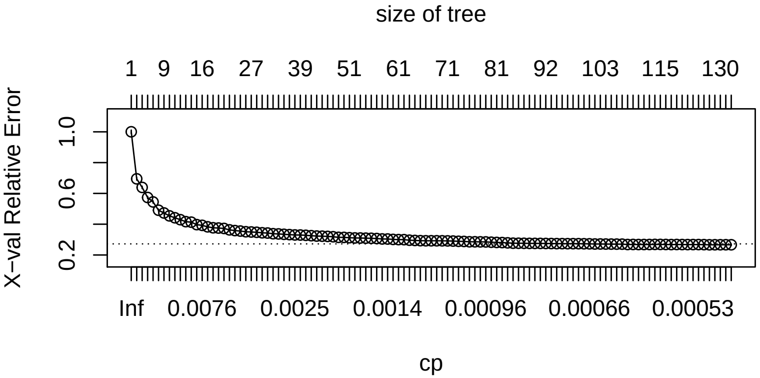
Objectives:

- Build Single Tree, Random Forest (RF), Boosting models
- Fit linear model
- Compare predictive capacity using Out of Sample (OOS) Mean Root Squared Error (MRSE).

We fit models on **training partition** and we evaluate their predictive capacity on **testing partition**.

Fit Single Tree model using *rpart* package

```
## size of big tree:  134
```




```
## the best size is: 128
```

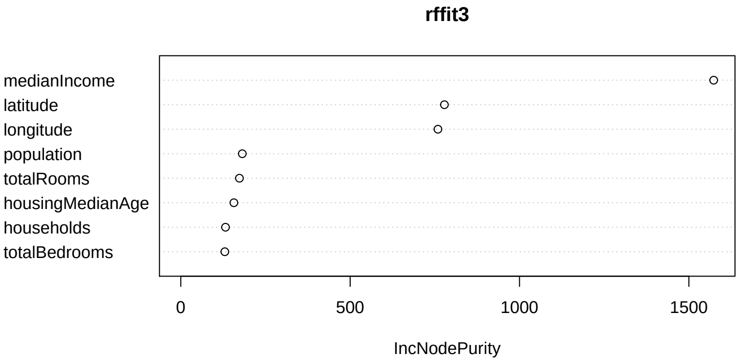
```
## RMSE Single Tree Model:  0.2939057
```

Fit Random Forest model using *randomForest* package

Ideally, fit many models with different tuning parameters. Choose the one with the best OOS RMSE.

```
## RMSE on test for RF m=3 ntree=50:  0.2472154
```

```
## Variable Importance RF
```



Fit Boosting model using *gbm*[2] package

Ideally, fit many models with different tuning parameters. Choose the one with the best OOS RMSE.

```
## RMSE on test for Boosting 4; 1000; 2:  0.2402014

## Variable Importance Boosting

##                                var   rel.inf
## medianIncome           medianIncome 45.594766
## longitude                 longitude 17.294545
## latitude                   latitude 17.137733
## population               population  5.435569
## totalBedrooms         totalBedrooms  4.283103
## totalRooms               totalRooms  3.733635
## households               households  3.392255
## housingMedianAge housingMedianAge  3.128394
```

[2] GBM: Gradient Boost Machine

Fit Linear model and compare OOS RMSE

- Linear model OOS predictive capacity:

```
## RMSE on test for linear model:  0.3485098
```

- Now, let's compare OOS predictive capacity of all models

```
##       rmse_rpart  rmse_rf3 rmse_gbm3   rmse_lm
## [1,]   0.2939057 0.2472154 0.2402014 0.3485098
```

- Which model does the best job?

Extra practice: Classification Trees using *iris* dataset

Objectives:

- Build Single Tree, Random Forest (RF), Boosting models
- Fit multinomial model
- Compare predictive capacity using OOS **Accuracy**

Fit models on **training partition** and evaluate their predictive capacity on **testing partition**.

Fit Single Tree model

- Classification table Single Tree model

```
##               rpartfitpred
##                setosa versicolor virginica
##    setosa         30          0         0
##    versicolor      0         20         1
##    virginica       0          1        23
```

Fit RF model

- Classification table RF

```
##               rfritpred
##                setosa versicolor virginica
##   setosa           30          0         0
##   versicolor        0         20         1
##   virginica         0          1        23
```

Fit Boosting model

- Classification table Boosting model

```
##              gbmfitpredcat
##                1  2  3
##   setosa      30  0  0
##   versicolor   0 19  2
##   virginica    0  1 23
```

Fit Multinomial model

- Classification table Multinomial model

```
##             mnfitpred
##              setosa versicolor virginica
##   setosa         30          0         0
##   versicolor      0         20         1
##   virginica       0          0        24
```

Compare OOS Accuracy

- Now, compare OOS predictive capacity of all models

```
##       rpart_acc    rf_acc gbm_acc    mn_acc
## [1,] 0.9733333 0.9733333    0.96 0.9866667
```

- Which model does the best job?