Prediciting Baseball Player Salaries Using Regression Trees

Czar Yobero

11/13/2017

Introduction

In this tutorial, we will construct a regression tree to predict the salaries of baseball players based on multiple features.

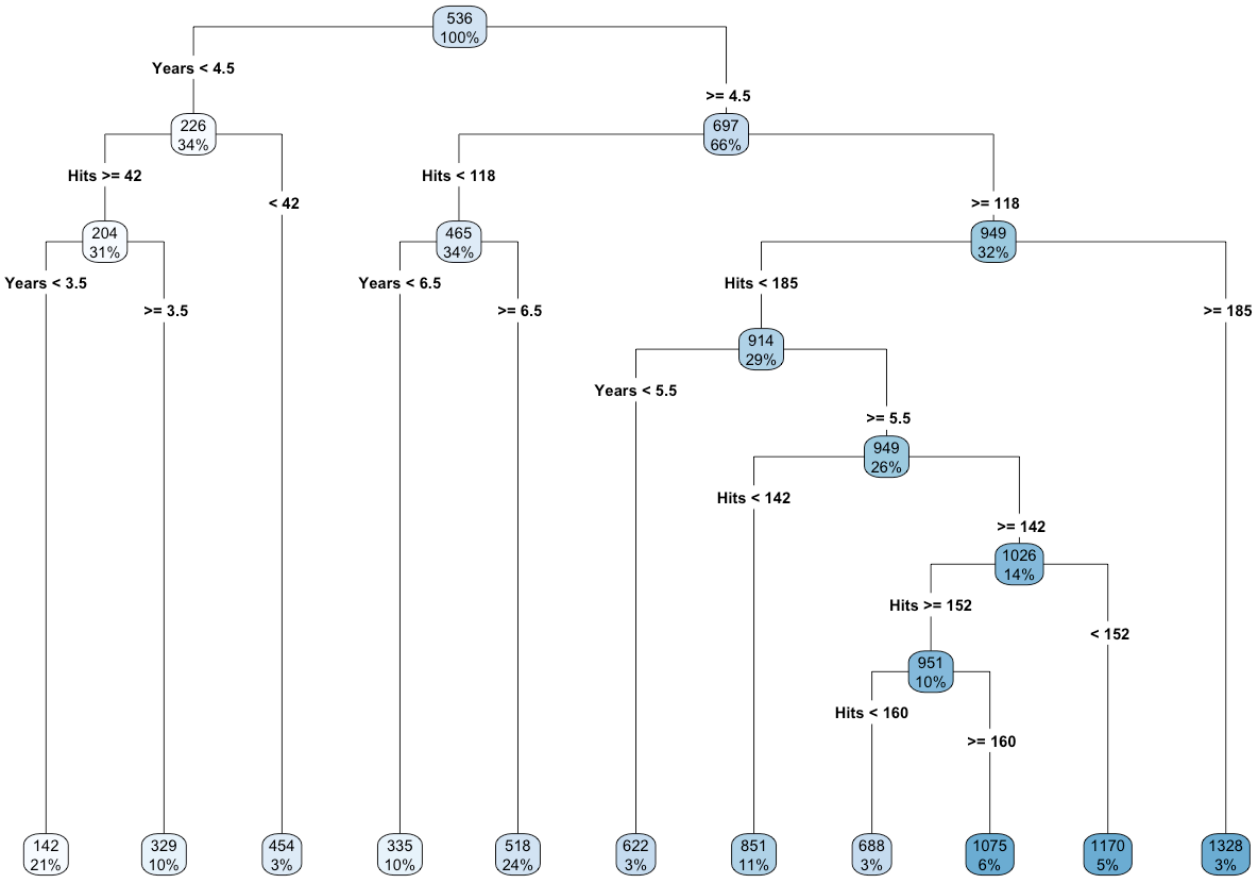
1. Decision Trees

Tree-based methods, while simple and useful for interpretation, are typically not as competitive with the best supervised learning approaches such as *polynomial regression*, *local regression*, or *general additive models (GAM)*. However, tree-based methods such as *bagging*, *random forests*, and *boosting* may make up for this shortfall. By combining a large number of trees instead of one, the model can result in dramatic improvements in terms of prediction accuracy. However, this improvement in accuracy may be at the expense of some loss in intepretation. Regression trees can be used for both classification and regression. We will obviously be focusing on the latter.

1.1 Regression Trees

In a **regression tree**, the tree arranges or segments observations into regions of a predictor space. For example, using the "Hitters" data set, which contains various statistics on baseball players, a tree might look something like the following

```
library(rpart)
library(rpart.plot)
hitters <- read.csv('/Users/czar.yobero/R/Hitters.csv')
reg.tree <- rpart(Salary ~ Years + Hits, data = hitters)
rpart.plot(reg.tree, type = 4)
```



We can interperet the above regression tree as follows: the most important factor in determining a player's salary is *years* (i.e. experience). Players who have been in the league for at least 4.5 years tend to make more than players who have only been in the league for less than that time. Hits also plays a factor in determining a player's salary (after all, we did include it in our equation). If you're a baseball fan, as you might expect, players who have more hits tend to make more than players who have less hits. We can check the importance each variable plays into the dependent variable using the following code

```
reg.tree$variable.importance
```

```
##    Years     Hits
## 15696202 14259635
```

It is evident that Years plays a relatively more important role than Hits in determining a player's salary (at least based on our simple example).

In any decision tree, observations are often grouped into regions $R_1, R_2, \ldots R_n$. These regions are commonly referred to as **terminal nodes** or **leaves** of the tree. Decision trees are generalliy plotted upside down, in the sense that the terminal nodes are at the bottom of the tree. The points along the decision tree where teh predictor space is split are referred to as **internal nodes**.

2. Building a Regression Tree

2.1 Prediction through Stratification of the Feature Space

We are now ready to begin to build a regression tree. Generally speaking, there are two main steps involved in constructing such a tree.

1. Divide the predictor space - the set of all possible values for predictors $X_1, X_2, \ldots X_p$ - into J distinct, non-overlapping regions $R_1, R_2, \ldots R_j$.
2. For each observation that falls into the region R_j, make the same prediction, which is simply the mean of the response values for the training observations in R_j.

The question now is, "How do we go about constructing the regions $R_1, R_2, \ldots R_j$?" The answer is that we choose to divide the predictor space into high-dimensional rectangles (or boxes) for simplicity and for ease of interpretation of the resulting predictive model. The goal to find boxes $R_1, \ldots R_j$ that minimizes the residual sum of squares RSS is given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the mean response ofr the etraining observations within the jth box. Unfortunately, it is computationallyi difficult, nay infeasible, to consider *every* possible partition of the feature space into J boxes. As a result, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

To perform recursive binary splitting, we first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leadds to the greatest possible reduction in RSS. The notation $\{X|X_j < s\}$ denotes *the region of predictor space in which X_j takes on a value less than s*. In other words, we consider all predictors $X_1, X_2, \ldots X_p$ and all possible values of the cutpoint s for each of the predictors, and thehn chhoose theh predictor and cutpoint such that the resulting tree has the lowest RSS. In more detail, for any j and s we define the pair of half-plains

$$R_1(j, s) = \{X|X_j < s\} \qquad \text{and} \qquad R_2(j, s) = \{X|X \geq s\}$$

where $\hat{y}_{R_1}$ is thhe mean response for the training observations in $R_1(j, s)$, and $\hat{y}_{R_2}$ is the mean response for the training observations in $R_2(j, s)$.

Next, we repeat the process and look for the best predictor and best cutpoint to split the ddata further so as to minimize the RSS withhin each of the resulting regions. However, this time around, rather than splitting thhe entire predictor space, we split one of the two previously identified regions. We now have three regions. Again, we are looking to split one of these regions further so as to minimize RSS. This process continues until a stopping criterion is reached.

2.2 Tree Pruning

Although the process above may produce goo predictors on the training set, it is likely to *overfit* the data, leaing to poor test set performance. One solution to this problem is to produce a smaller tree with fewer splits (i.e. fewer R_j's). Such a tree might lead to lower variance and better intepretation at the expense of a little bias.

An alternative process is to the one above is to build the tree only so long as the decrease in RSS due to each split exceeds some high threshohld. This strategy will result in smaller trees, but is too short-sightehd since a seemingly worthless split early on in the tree might be followed by a very good split. Therefore, a superior strategy is to grow a very large tree T_0 and then *prune* it back in order to obtain a **subtree**. But just hohw do we determine the best way to prune the tree? The answer is that we need a way to select a small set of subtrees for consideration. This is done through **cost complexity pruning**, or **weakest link pruning**.

Rather than considering *every possible subtree*, we consider a sequence of trees indexed by a non-negative tuning parameter α. Thus, the algorithm for building such a tree is as follows

1. Use *recursive binary splitting* to grow a large tree on the training data and stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply *cost complexity pruning* to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use **K-fold cross-validation** to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$.
    a. Repeat steps 1 and 2 on all but the kth fold of the training data.
    b. Evaluate the mean squared prediction error on the ata in the let-out kth fold, as a function of α.
    c. Average the results for each value of α and pick α to minimize the *average error*.
4. Return the subtree from step 2 that corresponds to the chosen value of α.

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. $|T|$ denotes the bnumber of terminal nodes of the tree T, R_m denotes the subset of predictor space corresponding to the $m$thh terminal node, and $\hat{y}_{R_m}$ is the predicted response associated with R_m (i.e. the mean of the training observations in R_m).

3. An Example: Predicting Baseball Players' Salaries

Let us now extend our baseball salaries example to build a regression tree.

```
library(MASS)
library(rpart)
set.seed(1984)
train <- sample(1:nrow(hitters), nrow(hitters)/2)
tree.baseball <- rpart(Salary ~ Hits + HmRun + Runs + RBI + Walks + Years + Errors,
subset = train, data = hitters)
summary(tree.baseball)
```

```
## Call:
## rpart(formula = Salary ~ Hits + HmRun + Runs + RBI + Walks +
##     Years + Errors, data = hitters, subset = train)
##   n=132 (29 observations deleted due to missingness)
##
##           CP nsplit rel error    xerror      xstd
## 1 0.24729229      0 1.0000000 1.0075685 0.1937932
## 2 0.17431727      1 0.7527077 0.8197971 0.1970009
## 3 0.08513476      2 0.5783904 0.8668485 0.1878783
## 4 0.03302001      3 0.4932557 0.8095991 0.1849311
## 5 0.01875615      4 0.4602357 0.7640463 0.1658833
## 6 0.01540453      5 0.4414795 0.7682444 0.1656465
## 7 0.01198533      6 0.4260750 0.7600164 0.1658597
## 8 0.01000000      7 0.4140897 0.7586088 0.1694030
##
## Variable importance
```

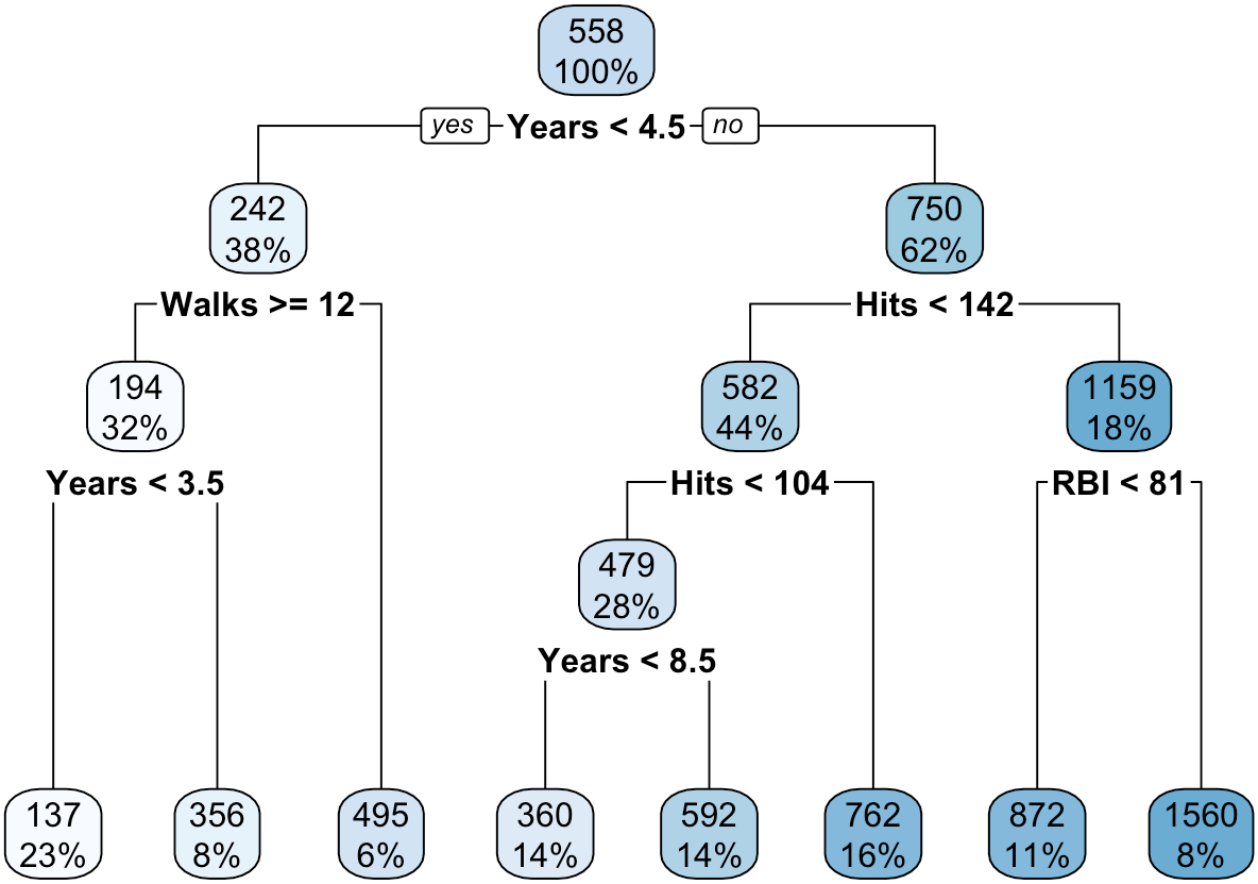
```
##   Hits  Years    RBI   Runs  HmRun  Walks Errors
##     22     22     17     16     11     10      1
##
## Node number 1: 132 observations,    complexity param=0.2472923
##   mean=557.9758, MSE=245671.6
##   left son=2 (50 obs) right son=3 (82 obs)
##   Primary splits:
##       Years < 4.5   to the left,  improve=0.2472923, (0 missing)
##       Walks < 66.5  to the left,  improve=0.2091453, (0 missing)
##       Runs  < 60    to the left,  improve=0.1663612, (0 missing)
##       Hits  < 143   to the left,  improve=0.1621618, (0 missing)
##       RBI   < 72.5  to the left,  improve=0.1408579, (0 missing)
##   Surrogate splits:
##       Hits  < 60.5  to the left,  agree=0.682, adj=0.16, (0 split)
##       RBI   < 16    to the left,  agree=0.682, adj=0.16, (0 split)
##       HmRun < 2.5   to the left,  agree=0.674, adj=0.14, (0 split)
##       Walks < 19    to the left,  agree=0.674, adj=0.14, (0 split)
##       Runs  < 29.5  to the left,  agree=0.667, adj=0.12, (0 split)
##
## Node number 2: 50 observations,    complexity param=0.01875615
##   mean=242.3267, MSE=112784.3
##   left son=4 (42 obs) right son=5 (8 obs)
##   Primary splits:
##       Walks < 12.5  to the right, improve=0.10785840, (0 missing)
##       Hits  < 44.5  to the right, improve=0.10589680, (0 missing)
##       HmRun < 1.5   to the right, improve=0.10210490, (0 missing)
##       RBI   < 14.5  to the right, improve=0.10016790, (0 missing)
##       Runs  < 23.5  to the right, improve=0.07569985, (0 missing)
##   Surrogate splits:
##       RBI  < 20.5  to the right, agree=0.92, adj=0.500, (0 split)
##       Hits < 32    to the right, agree=0.90, adj=0.375, (0 split)
##       Runs < 10.5  to the right, agree=0.90, adj=0.375, (0 split)
##
## Node number 3: 82 observations,    complexity param=0.1743173
##   mean=750.4448, MSE=228903.4
##   left son=6 (58 obs) right son=7 (24 obs)
##   Primary splits:
##       Hits  < 142.5 to the left,  improve=0.3011641, (0 missing)
##       Runs  < 60    to the left,  improve=0.2595740, (0 missing)
##       RBI   < 81    to the left,  improve=0.2506971, (0 missing)
##       Walks < 43.5  to the left,  improve=0.2365055, (0 missing)
##       HmRun < 16.5  to the left,  improve=0.1498388, (0 missing)
##   Surrogate splits:
##       Runs  < 70    to the left,  agree=0.890, adj=0.625, (0 split)
##       RBI   < 67.5  to the left,  agree=0.829, adj=0.417, (0 split)
##       Walks < 50.5  to the left,  agree=0.793, adj=0.292, (0 split)
##       HmRun < 27    to the left,  agree=0.780, adj=0.250, (0 split)
##
## Node number 4: 42 observations,    complexity param=0.01198533
##   mean=194.1905, MSE=34714.64
##   left son=8 (31 obs) right son=9 (11 obs)
##   Primary splits:
##       Years < 3.5   to the left,  improve=0.2665734, (0 missing)
##       HmRun < 24.5  to the left,  improve=0.1625136, (0 missing)
```

```
##       RBI   < 89.5  to the left,  improve=0.1597607, (0 missing)
##       Runs  < 74.5  to the left,  improve=0.1528585, (0 missing)
##       Hits  < 112.5 to the left,  improve=0.1474041, (0 missing)
##   Surrogate splits:
##       Hits   < 159.5 to the left,  agree=0.762, adj=0.091, (0 split)
##       Walks  < 64.5  to the left,  agree=0.762, adj=0.091, (0 split)
##       Errors < 21    to the left,  agree=0.762, adj=0.091, (0 split)
##
## Node number 5: 8 observations
##   mean=495.0416, MSE=446620.6
##
## Node number 6: 58 observations,    complexity param=0.03302001
##   mean=581.5489, MSE=84082.62
##   left son=12 (37 obs) right son=13 (21 obs)
##   Primary splits:
##       Hits   < 104.5 to the left,  improve=0.2195694, (0 missing)
##       Runs   < 33.5  to the left,  improve=0.1914736, (0 missing)
##       Walks  < 20.5  to the left,  improve=0.1371105, (0 missing)
##       Years  < 6.5   to the left,  improve=0.1191623, (0 missing)
##       Errors < 4.5   to the left,  improve=0.1157724, (0 missing)
##   Surrogate splits:
##       Runs   < 49.5  to the left,  agree=0.931, adj=0.810, (0 split)
##       RBI    < 54    to the left,  agree=0.776, adj=0.381, (0 split)
##       Errors < 7.5   to the left,  agree=0.759, adj=0.333, (0 split)
##       HmRun  < 17    to the left,  agree=0.741, adj=0.286, (0 split)
##       Walks  < 66.5  to the left,  agree=0.690, adj=0.143, (0 split)
##
## Node number 7: 24 observations,    complexity param=0.08513476
##   mean=1158.61, MSE=343350.4
##   left son=14 (14 obs) right son=15 (10 obs)
##   Primary splits:
##       RBI    < 81    to the left,  improve=0.3350325, (0 missing)
##       Walks  < 70.5  to the left,  improve=0.1915065, (0 missing)
##       HmRun  < 15.5  to the left,  improve=0.1835997, (0 missing)
##       Years  < 7.5   to the left,  improve=0.1288186, (0 missing)
##       Errors < 5.5   to the left,  improve=0.1015978, (0 missing)
##   Surrogate splits:
##       HmRun < 15.5  to the left,  agree=0.833, adj=0.6, (0 split)
##       Hits  < 185   to the left,  agree=0.708, adj=0.3, (0 split)
##       Runs  < 84    to the left,  agree=0.708, adj=0.3, (0 split)
##       Walks < 60.5  to the left,  agree=0.625, adj=0.1, (0 split)
##       Years < 12    to the left,  agree=0.625, adj=0.1, (0 split)
##
## Node number 8: 31 observations
##   mean=136.8871, MSE=3426.479
##
## Node number 9: 11 observations
##   mean=355.6818, MSE=87556.92
##
## Node number 12: 37 observations,    complexity param=0.01540453
##   mean=479.1847, MSE=66810.12
##   left son=24 (18 obs) right son=25 (19 obs)
##   Primary splits:
##       Years  < 8.5   to the left,  improve=0.20208470, (0 missing)
```

```
##       Walks  < 42.5  to the left,  improve=0.17910550, (0 missing)
##       Runs   < 33.5  to the left,  improve=0.13689250, (0 missing)
##       Errors < 2.5   to the left,  improve=0.08896843, (0 missing)
##       HmRun  < 8.5   to the left,  improve=0.06583124, (0 missing)
##   Surrogate splits:
##       Hits   < 79.5  to the right, agree=0.622, adj=0.222, (0 split)
##       Errors < 4.5   to the left,  agree=0.622, adj=0.222, (0 split)
##       HmRun  < 3.5   to the left,  agree=0.595, adj=0.167, (0 split)
##       Runs   < 32.5  to the right, agree=0.595, adj=0.167, (0 split)
##       Walks  < 21.5  to the left,  agree=0.595, adj=0.167, (0 split)
##
## Node number 13: 21 observations
##   mean=761.9047, MSE=63524.92
##
## Node number 14: 14 observations
##   mean=871.9626, MSE=171947.7
##
## Node number 15: 10 observations
##   mean=1559.917, MSE=307233.8
##
## Node number 24: 18 observations
##   mean=359.8056, MSE=25916.61
##
## Node number 25: 19 observations
##   mean=592.2807, MSE=79259.33
```

Plot our tree.

```
library(rpart.plot)
rpart.plot(tree.baseball)
```



According to our tree, baseball players with more experience, hihts and RBIs (runs batted in) tend to have a higher salary than their peers.

The most important factors in determining salary (according to thhe features used in our model) are

```
tree.baseball$variable.importance
```

```
##      Hits     Years       RBI      Runs     HmRun     Walks    Errors
## 9209438.4 9183650.9 7111305.5 6501789.9 4581610.2 3927343.5  503275.6
```