Boosted Trees

Alex Wood, Wyatt Lansford, Luke Kim, and Mitchell Neat

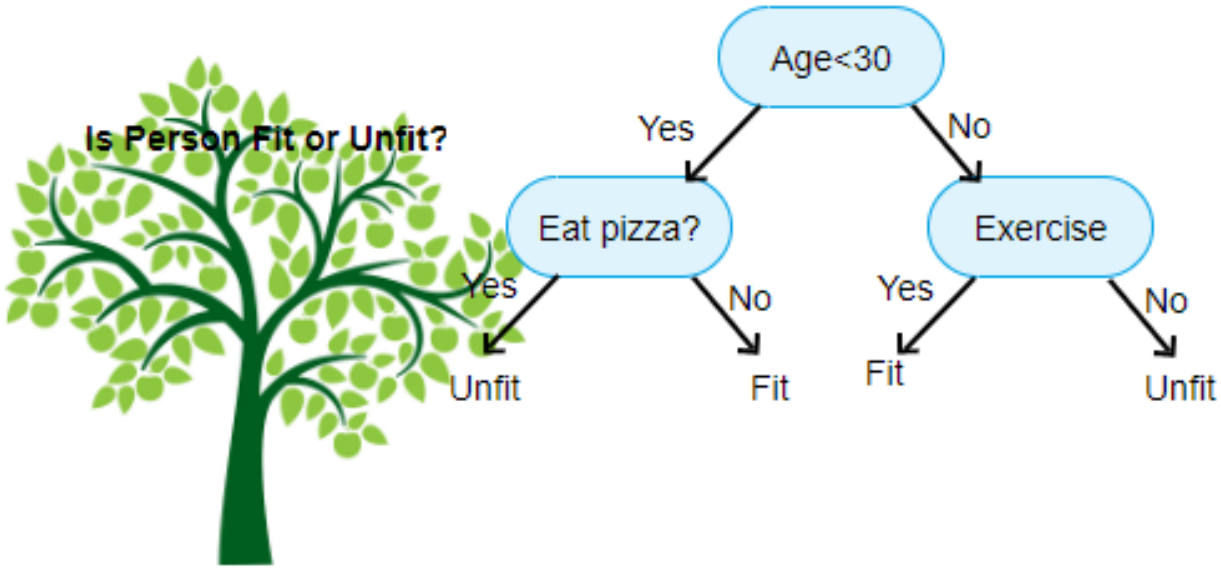
12/3/2019

Outline

- Boosted Trees
- Gradient Boosted Trees
  - Parameters
  - Math/Algorithm Details
  - Visual Examples
- Bostom Crime Data
  - Regression
  - Trees
  - Gradient Boosted Trees
- Summary

What are Decision Trees?

- Take set of features and split data recursively based on features.
- Splits are chosen to minimize entropy.



What is Boosting?

- Combining weak learners (trees) into strong classifiers.
- Key Aspects:
  - weighting data
  - misclassification
  - regularization

Combined: Boosted Decision Trees

- Iteratively adding learners (trees) to minimize this objective function of the decision tree.

$$O(x) = \Sigma_i L(y_i, y_i) + \Sigma_t \Omega(f_t)$$

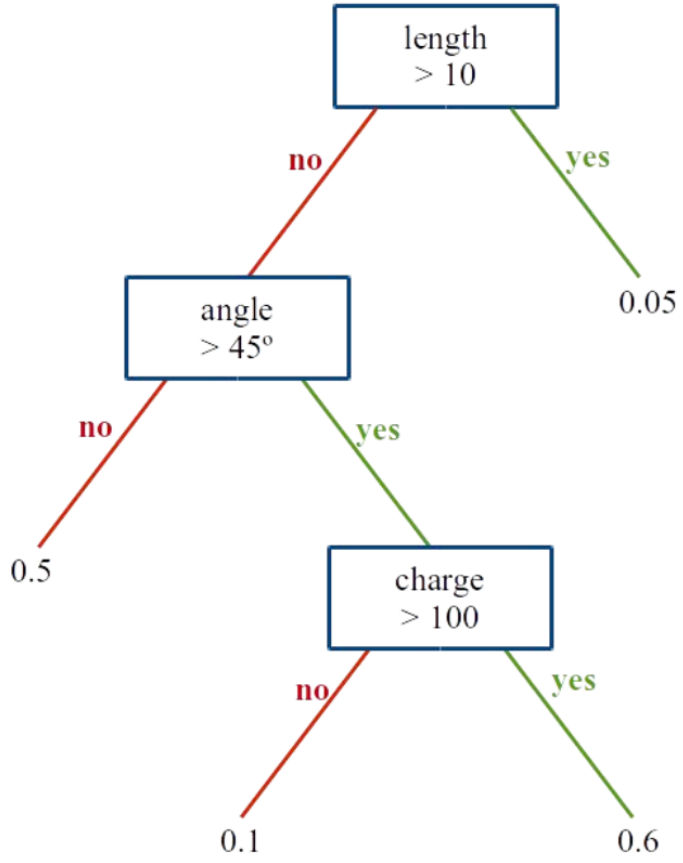
- $L(y_i, y_i)$ is the loss function
- $\Omega(f_t)$ is the regularization function

Types of Boosted Trees

- Ada Boost
  - One of the original boosted tree methods
  - Adapts a set of user-specified weak learners to create a strong learner
- Gradient Boosting
  - Uses gradient descent to optimize decision tree
  - Builds weak learners to predict loss of the previous step
- XGBoost
  - Specific subtype of gradient boosted trees
  - "eXtreme Gradient Boosting"

Generic Tree Parameters

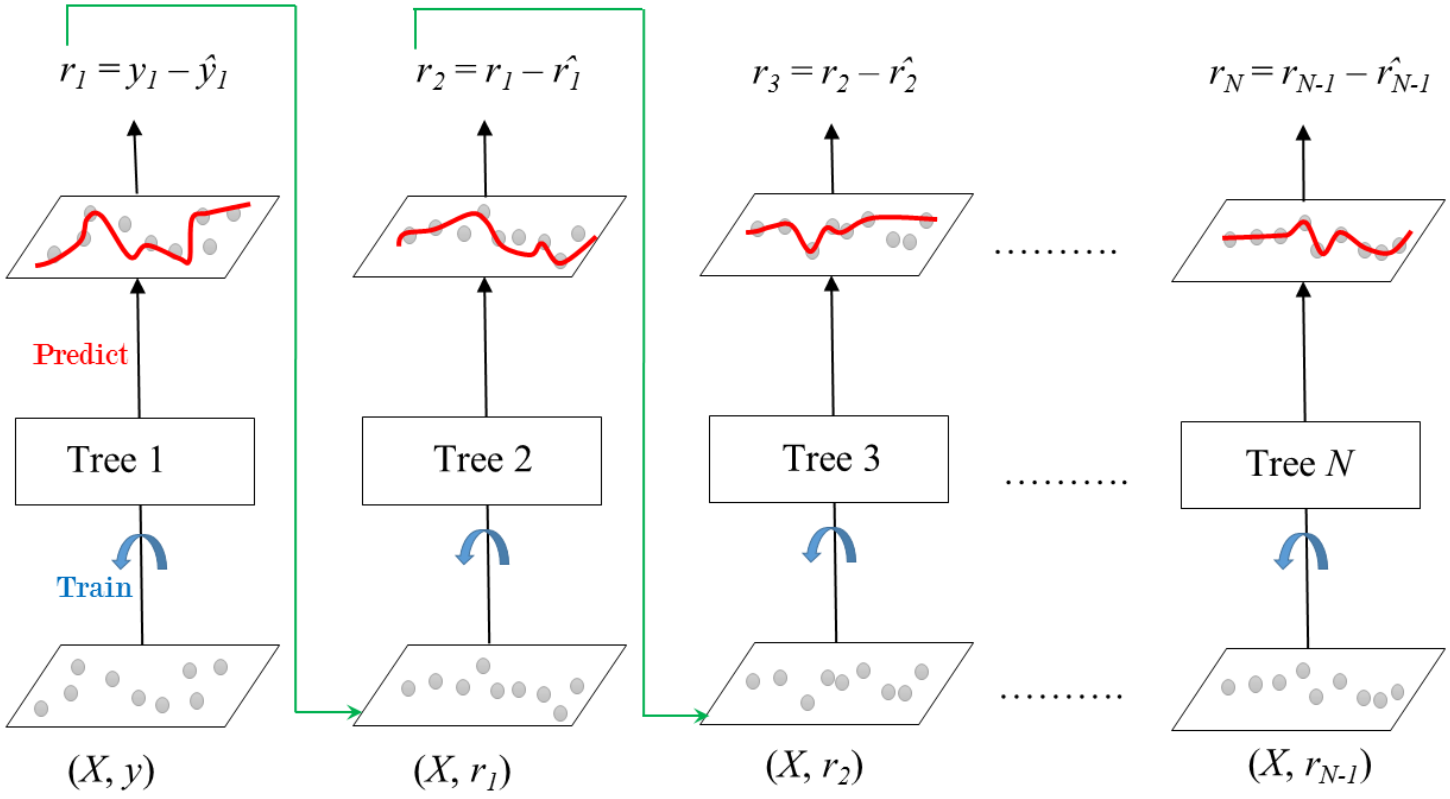
- Maximum depth
- Maximum features
- Number of Trees
- Minimum samples per leaf



GBT Parameters

- Loss Function
- Learning Rate
- Subsample Size
- Number of Trees
  - Generic tree parameters apply here

Gradient Boosted Tree



GBT Algorithm

- Step 1: Initialize Model Values
  - Values should be constants

$$F_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma)$$

- Step 2: Fit Weak Learners
  - (more details on next slide)
- Step 3: Output $F_M(x)$
- Step 4: Prune (optional)

Step 2 in Detail

1. Compute pseudo-residuals

$$r_{im} = -[\frac{\delta L(y_i, F(x_i))}{\delta F(x_i)}]_{F(x)=F_{m-1}(x)} \text{ for } i = 1, ..., n.$$

2. Fit base learner (tree) to psuedo-residuals

   Train the tree using training set (x_i, r_{im}) for $i = 1, ..., n$

3. Compute multiplier γ_m by solving 1D optimization problem

$$\gamma_m = argmin \sum_{i=1}^{n} L(y_i, F_{m-1} + \gamma h_m(x_i))$$

4. Update model

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Detailed Algorithm

Algorithm 4: Gradient tree boosting

Input : Data set $\mathcal{D}$.
A loss function L.
The number of iterations M.
The learning rate η.
The number of terminal nodes T

Initialize $\hat{f}^{(0)}(x) = \hat{f}_0(x) = \hat{\theta}_0 = \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta)$;

for $m = 1,2,..,M$ **do**

  $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$;

  Determine the structure $\{\hat{R}_{jm}\}_{j=1}^{T}$ by selecting splits which maximize $Gain = \frac{1}{2}\left[\frac{G_L^2}{n_L} + \frac{G_R^2}{n_R} - \frac{G_{jm}^2}{n_{jm}}\right]$;

  Determine the leaf weights $\{\hat{w}_{jm}\}_{j=1}^{T}$ for the learnt structure by $\hat{w}_{jm} = \arg\min_{w_j} \sum_{i \in \hat{I}_{jm}} L(y_i, \hat{f}^{(m-1)}(x_i) + w_j)$;

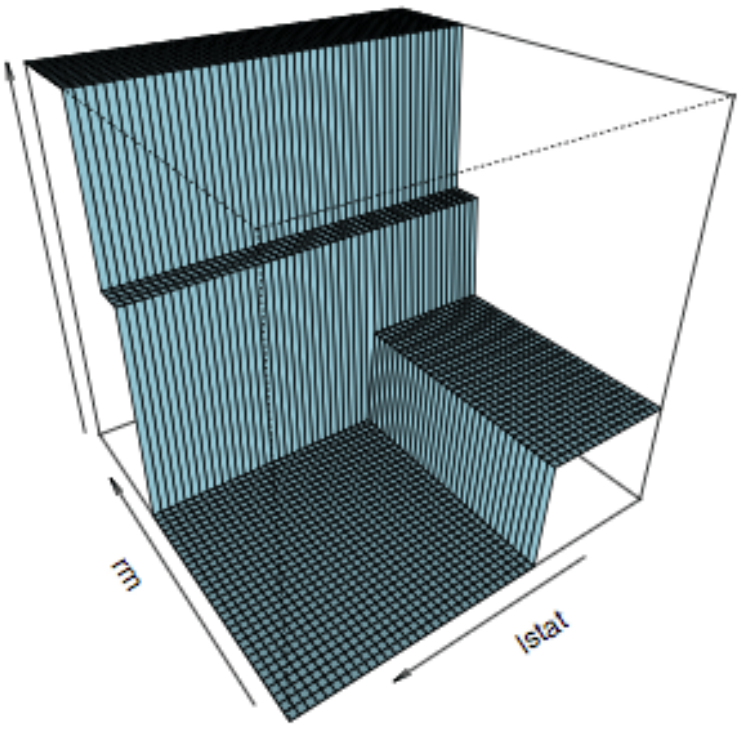
  $\hat{f}_m(x) = \eta \sum_{j=1}^{T} \hat{w}_{jm} \mathrm{I}(x_i \in \hat{R}_{jm})$;

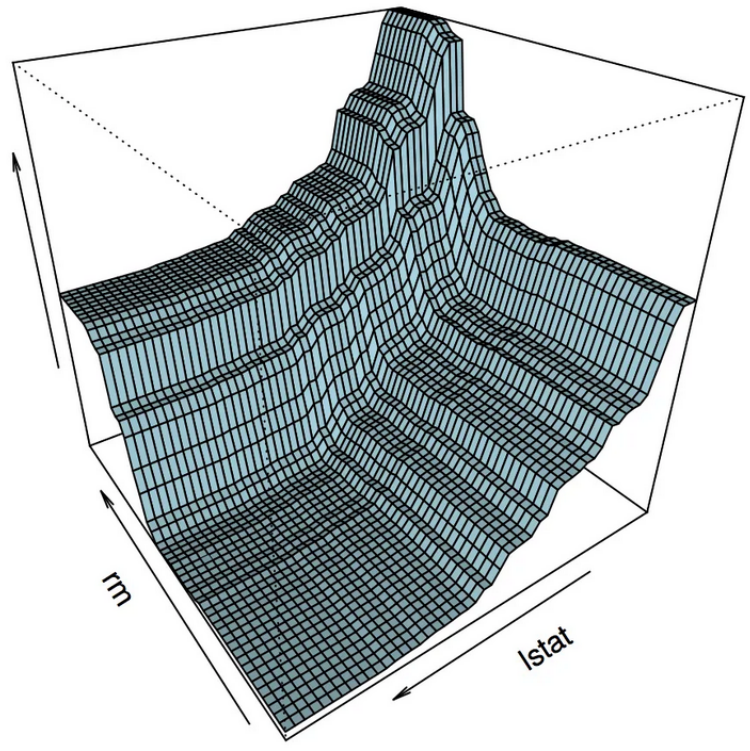
  $\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$;

end

Output: $\hat{f}(x) \equiv \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$

Visual Example





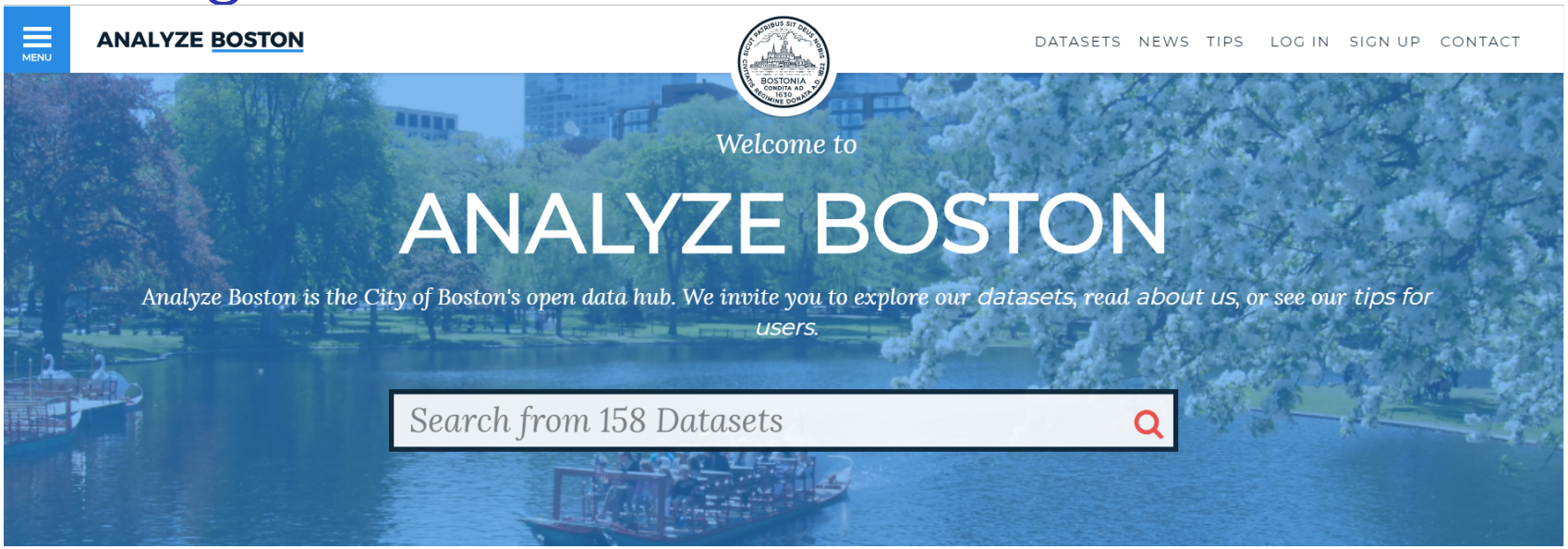
Why?

- Tree Boosting 'beats' the *curse of dimensionality* by not relying on any distance metric, as well as employing a large amount of inherent regularization through boosting parameters, tree parameters, and randomization parameters
  - Information becomes diluted if there are too many dimensions represented.
- Deeper trees allow capturing of interactions of features automatically.

Why?

- Boosting fits the data against the errors of the previous trees, allowing for a self-correcting of incorrect predictions
- Using optional methods such as gradient descent allows you to minimize much more complicated loss functions
  - If using a learning rate, this also introduces the concept of shrinkage
  - Shrinkage brings slower convergence and therefore generally improved performance
- Boosting trees generally perform well for both classification and regression

Introducing the Dataset

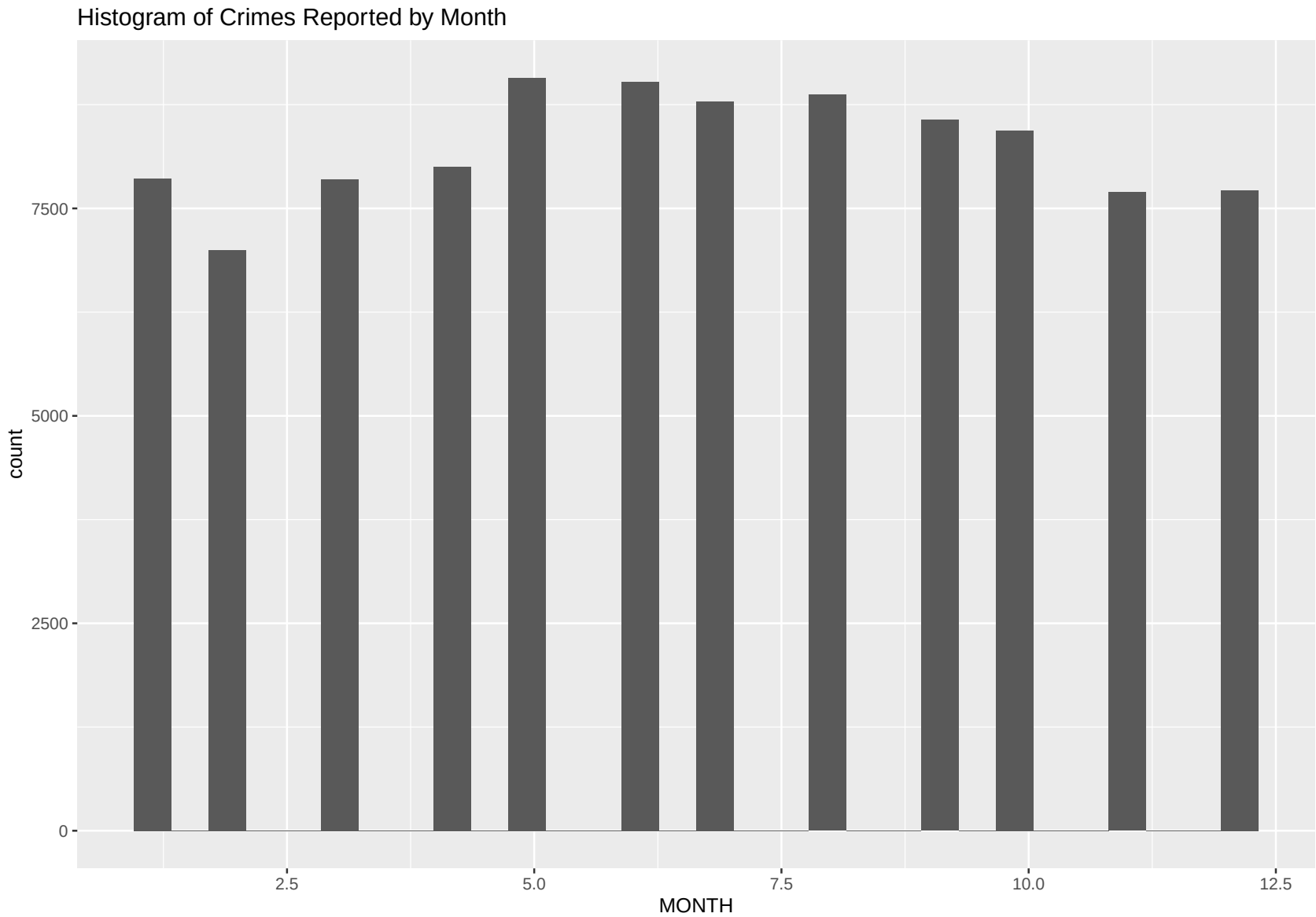


OFFENSE_DESCRIPTION	OCCURRED_ON_DATE	STREET
INVESTIGATE PERSON	2018-04-30 09:00:00	HAZLETON S
LARCENY ALL OTHERS	2018-03-06 08:00:00	HYDE PARK A
HARASSMENT	2018-10-31 12:00:00	PRIMROSE ST
HARASSMENT	2018-04-09 08:43:00	ATLANTIC AV
PROPERTY - MISSING	2018-01-01 00:00:00	COMMONWEA
HARASSMENT	2018-10-01 09:15:00	CEDRUS AVE

Crime incident reports provided by Boston Police Department

Exploratory Analysis

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```

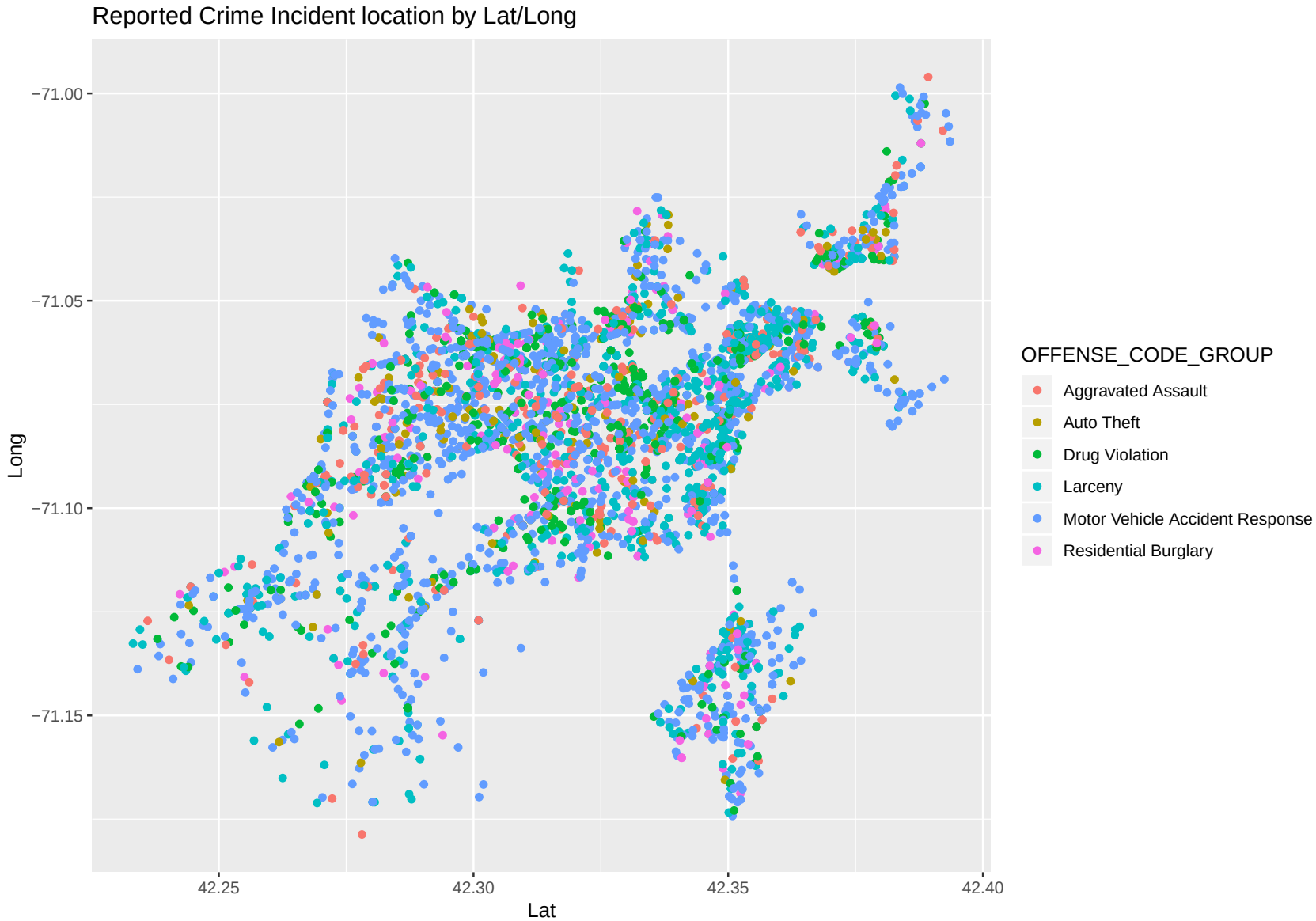


Exploratory Analysis (cont.)

There are 61 different types of offense code group. We have picked 6 interesting crimes reported in our data.

Var1	Freq
Motor Vehicle Accident Response	1936
Larceny	1365
Drug Violation	802
Aggravated Assault	404
Residential Burglary	235
Auto Theft	212

Exploratory Analysis (cont. . .)



Question we want to answer. . .

Could we predict what kind of offense(OFFENSE CODE GROUP)?

Regression Attempts

Trying to fit regression models to predict which crimes are being commited did not have great results. When trying to predict based on location we have an r squared of

```
## [1] 2.415002e-05
```

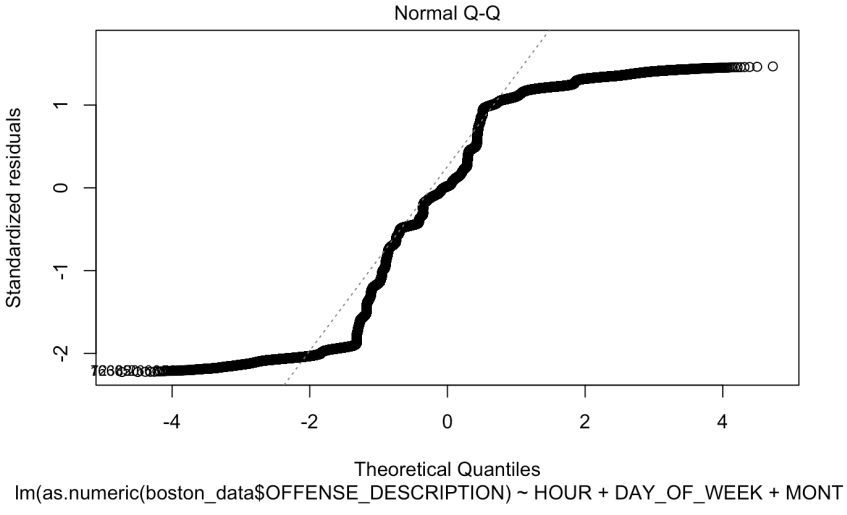
We then thought time may be a significant factor so we tried to predict type of crime based on the the month, day of week, and time of day. This again was unsuccessful and resulted in a r squared value of

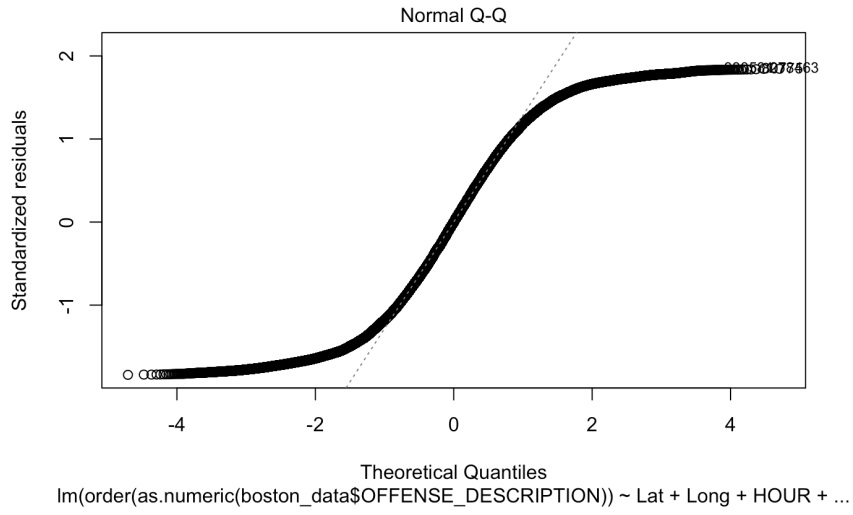
```
## [1] 0.0003162341
```

Finally we chose to make a model with any many factors from the data that we could. This is the best regression we could come up with and it still performed very poorly with an r squared value of

```
## [1] 0.003845766
```

Regression Attempts (cont. . .)





Regression is not the best choice for classification.

Regular Tree Fit

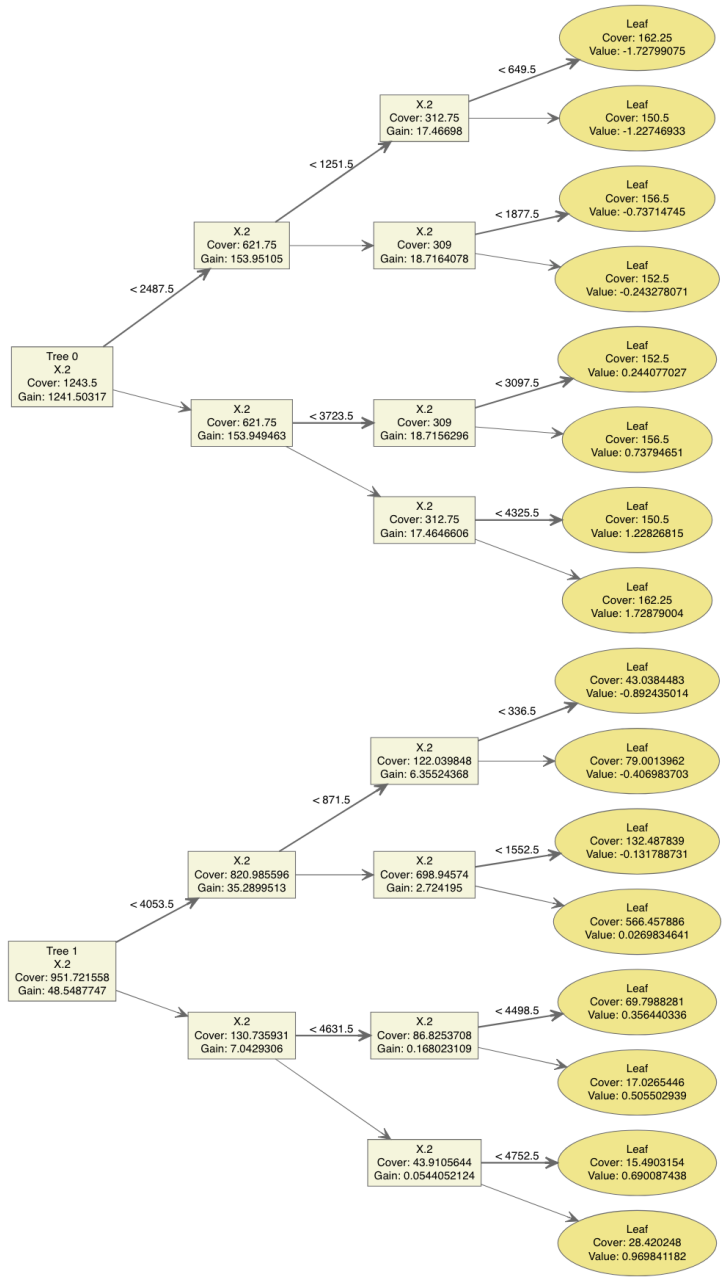
```
##   X INCIDENT_NUMBER OFFENSE_CODE                OFFENSE_CO
## 1 1      I192074532          619                          
## 2 2      I192073672          619                          
## 3 3      I192059691          522             Residential 
## 4 4      I192031665         3831 Motor Vehicle Accident 
## 5 5      I192026192          617                          
## 6 6      I192018124          413               Aggravated
##                     OFFENSE_DESCRIPTION DISTRICT REPORTI
## 1                    LARCENY ALL OTHERS       D4       
## 2                    LARCENY ALL OTHERS       C6       
## 3     BURGLARY - RESIDENTIAL - NO FORCE      D14       
## 4 M/V - LEAVING SCENE - PROPERTY DAMAGE      C11       
## 5           LARCENY THEFT FROM BUILDING       C6       
## 6        ASSAULT - AGGRAVATED - BATTERY       A1       
##      OCCURRED_ON_DATE YEAR MONTH DAY_OF_WEEK HOUR    UCR_
## 1 2018-10-15 00:00:00 2018    10      Monday    0   Part
## 2 2018-08-15 00:01:00 2018     8   Wednesday    0   Part
## 3 2018-08-05 01:00:00 2018     8      Sunday    1   Part
```

GBT Model

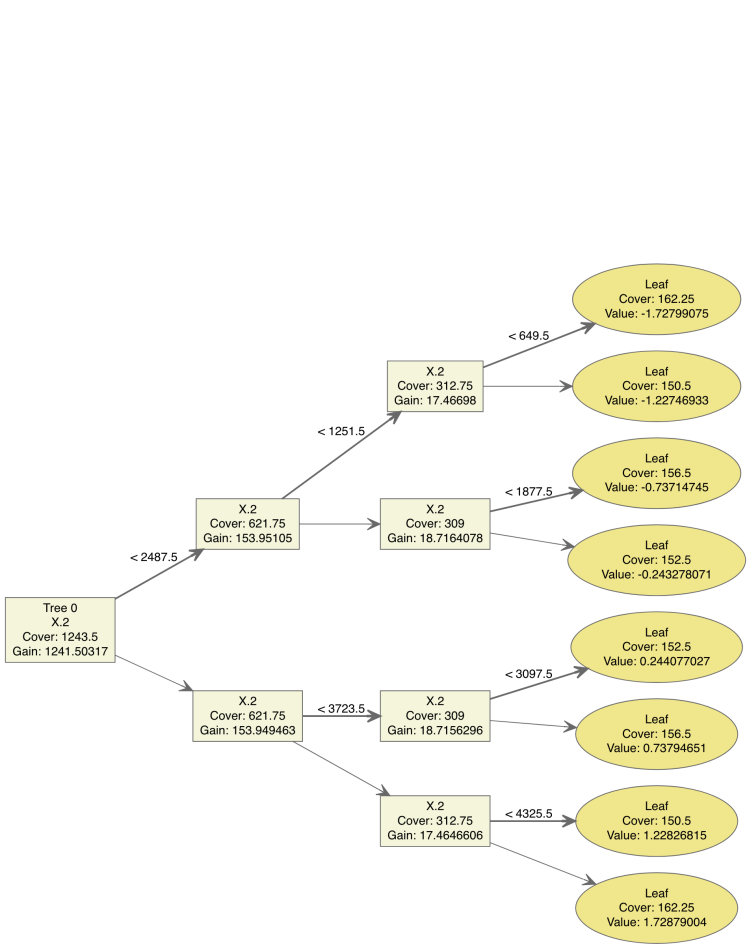
```
matdat <- data.matrix(test)

bst <- xgboost(data = matdat, label =
(test$X)/length(test$X), nrounds = 2, objective =
"binary:logistic")
```

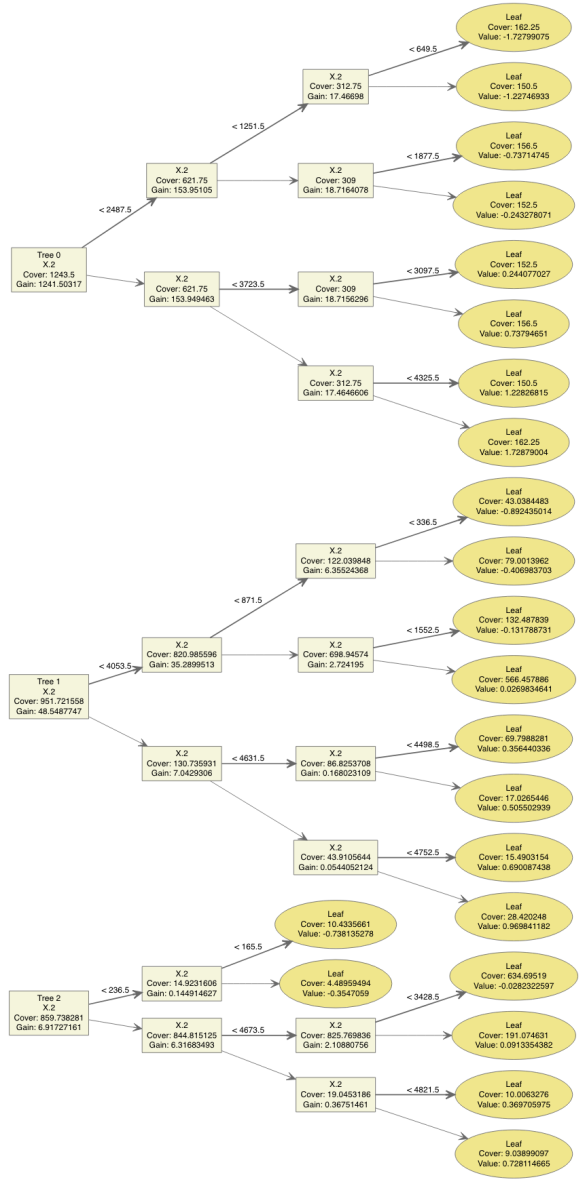
- data: training dataset
- label: vector of response values ([0:1])
- nrounds: max number of boosting iterations
- objective: returns gradients with given prediction and dtrain

GBT Plot


Tree 0
X.2
Cover: 1243.5
Gain: 1241.50317
< 2487.5
X.2
Cover: 621.75
Gain: 153.95105
< 1251.5
X.2
Cover: 312.75
Gain: 17.46698
< 649.5
Leaf
Cover: 162.25
Value: -1.72799075
Leaf
Cover: 150.5
Value: -1.22746933
X.2
Cover: 309
Gain: 18.7164078
< 1877.5
Leaf
Cover: 156.5
Value: -0.73714745
Leaf
Cover: 152.5
Value: -0.243278071
X.2
Cover: 621.75
Gain: 153.949463
< 3723.5
X.2
Cover: 309
Gain: 18.7156296
< 3097.5
Leaf
Cover: 152.5
Value: 0.244077027
Leaf
Cover: 156.5
Value: 0.73794651
X.2
Cover: 312.75
Gain: 17.4646606
< 4325.5
Leaf
Cover: 150.5
Value: 1.22826815
Leaf
Cover: 162.25
Value: 1.72879004
Tree 1
X.2
Cover: 951.721558
Gain: 48.5487747
< 4053.5
X.2
Cover: 820.985596
Gain: 35.2899513
< 871.5
X.2
Cover: 122.039848
Gain: 6.35524368
< 336.5
Leaf
Cover: 43.0384483
Value: -0.892435014
Leaf
Cover: 79.0013962
Value: -0.406983703
X.2
Cover: 698.94574
Gain: 2.724195
< 1552.5
Leaf
Cover: 132.487839
Value: -0.131788731
Leaf
Cover: 566.457886
Value: 0.0269834641
X.2
Cover: 130.735931
Gain: 7.0429306
< 4631.5
X.2
Cover: 86.8253708
Gain: 0.168023109
< 4498.5
Leaf
Cover: 69.7988281
Value: 0.356440336
Leaf
Cover: 17.0265446
Value: 0.505502939
X.2
Cover: 43.9105644
Gain: 0.0544052124
< 4752.5
Leaf
Cover: 15.4903154
Value: 0.690087438
Leaf
Cover: 28.420248
Value: 0.969841182

Changing Depth





Differences

Expected Loss of Basic Tree: 0.6170084

Expected Loss of Gradient Boosted Tree: 0.2500000

Summary

Boosted Trees are a powerful machine learning method extending from decision trees, that allows for a great improvement in model accuracy and precision.

The general idea is to "compute a sequence of decision trees in which each successive tree is built for the prediction results of the preceeding tree."