Statistical Learning with Boosting

Matthias Schonlau, Ph.D.

University of Waterloo, Canada

Outline

- Example: Ethyl data
  - Regression Trees (CART)
    - Overfitting
  - MART algorithm to Boosting
    - Tuning parameters
  - Comparison: Gaussian Linear regression vs boosting
- Example: Propensity scoring
  - Death penalty data
  - Comparison: Logistic regression vs boosting

Ethyl Acrylate

- Y= Kinematic viscosity of a lubricant
- x1=temperature (C)
- X2=pressure in atmospheres (atm)
- N=50

- Data Reference: Bates and Watts, Nonlinear regression analysis and its application, Wiley.

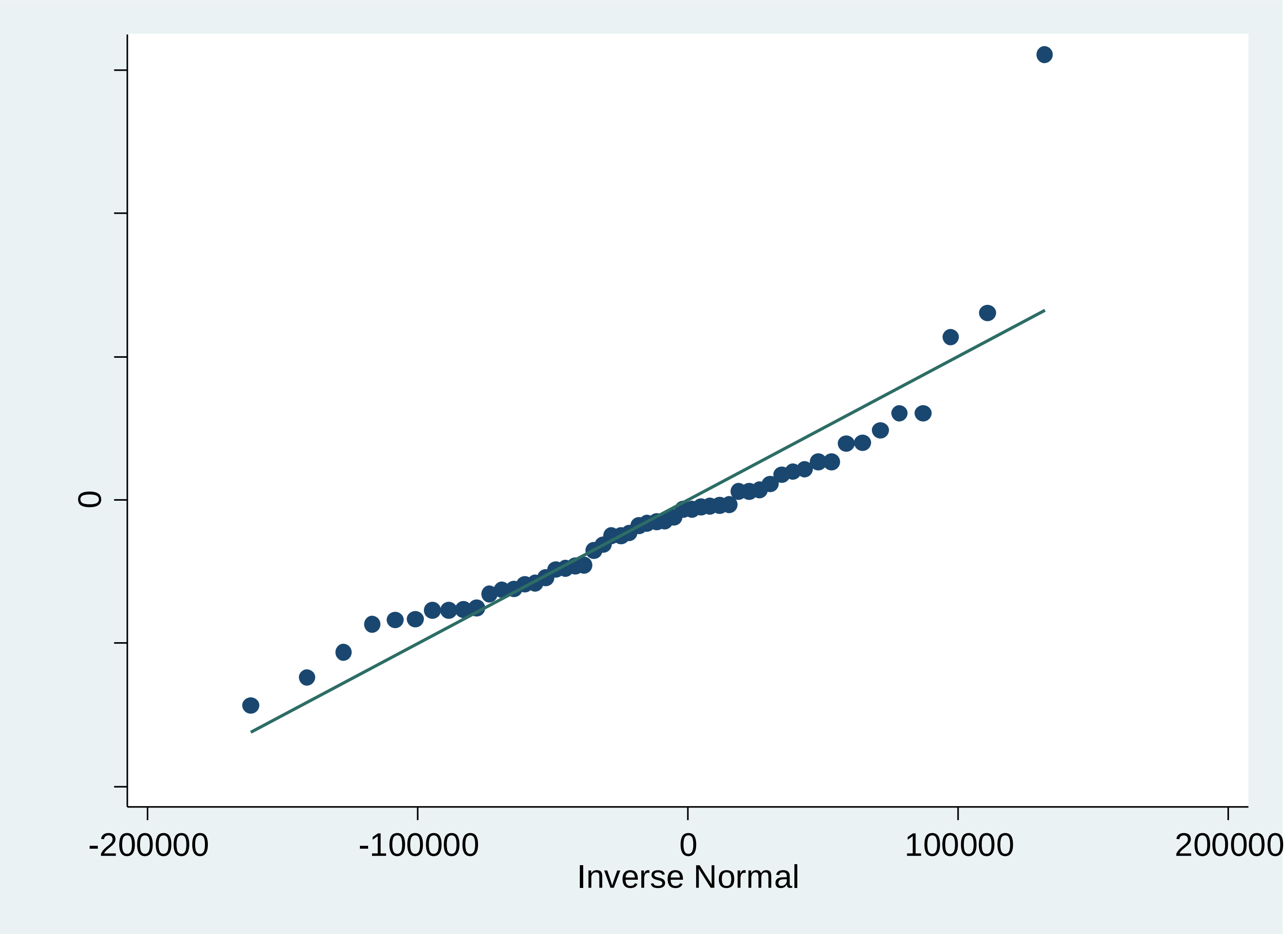
Linear regression

```
. regress viscosity pressure temp if train

      Source |       SS           df       MS      Number of obs   =        30
-------------+----------------------------------   F(2, 27)        =      9.76
       Model |  1.2685e+11         2  6.3424e+10   Prob > F        =    0.0006
    Residual |  1.7547e+11        27  6.4991e+09   R-squared       =    0.4196
-------------+----------------------------------   Adj R-squared   =    0.3766
       Total |  3.0232e+11        29  1.0425e+10   Root MSE        =     80617

------------------------------------------------------------------------------
   viscosity |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    pressure |   30.23023   7.539026     4.01   0.000     14.76143    45.69903
        temp |  -551.8822   415.7719    -1.33   0.196    -1404.976    301.2113
       _cons |  -5727.932   33660.81    -0.17   0.866    -74794.21    63338.34
------------------------------------------------------------------------------
```

- Regressing on 30 random observations
- Conclusion: temp does not matter

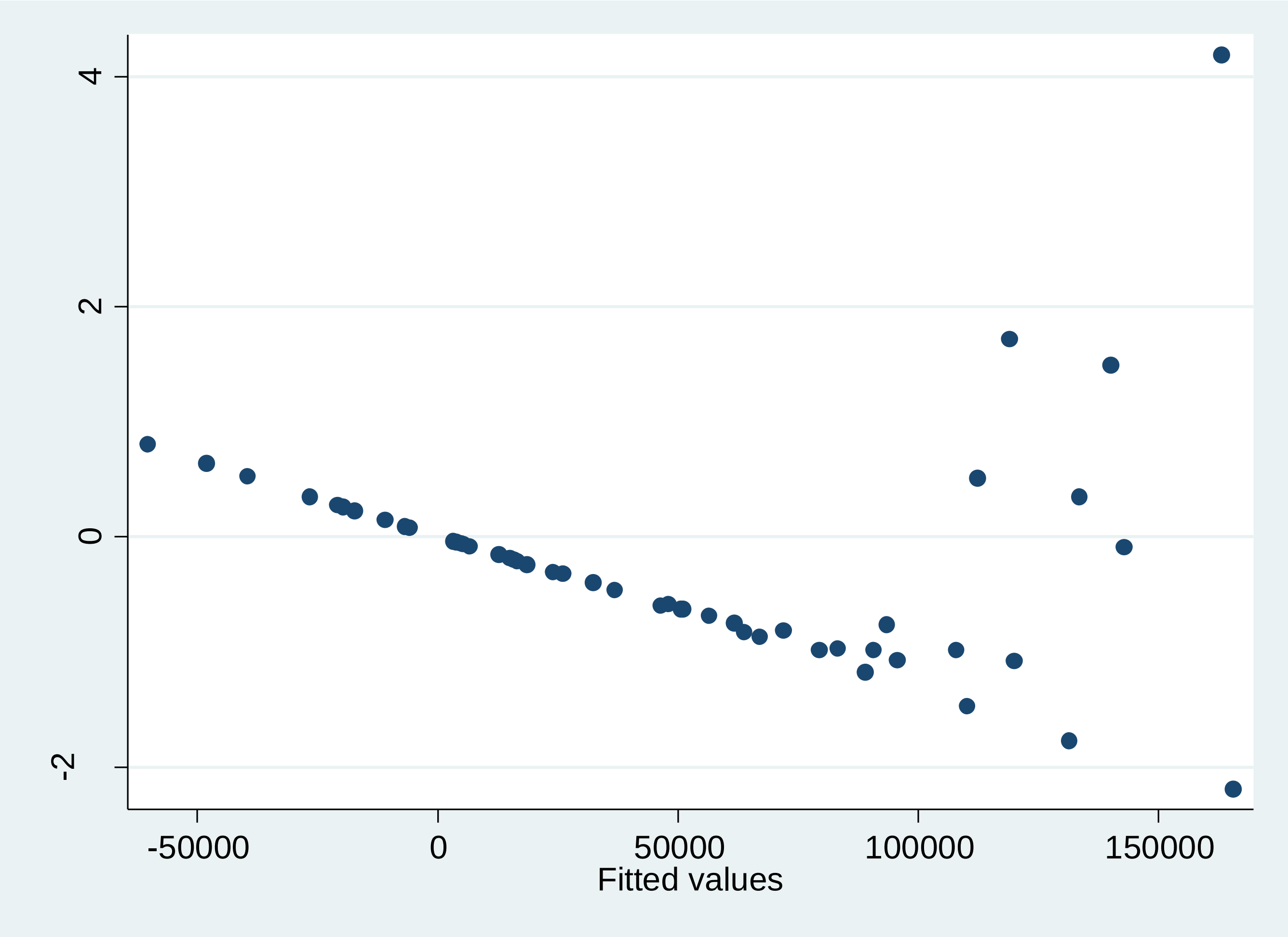
Linear regression diagnostic : qq plot

If the normality assumption were true, all points would be on the line

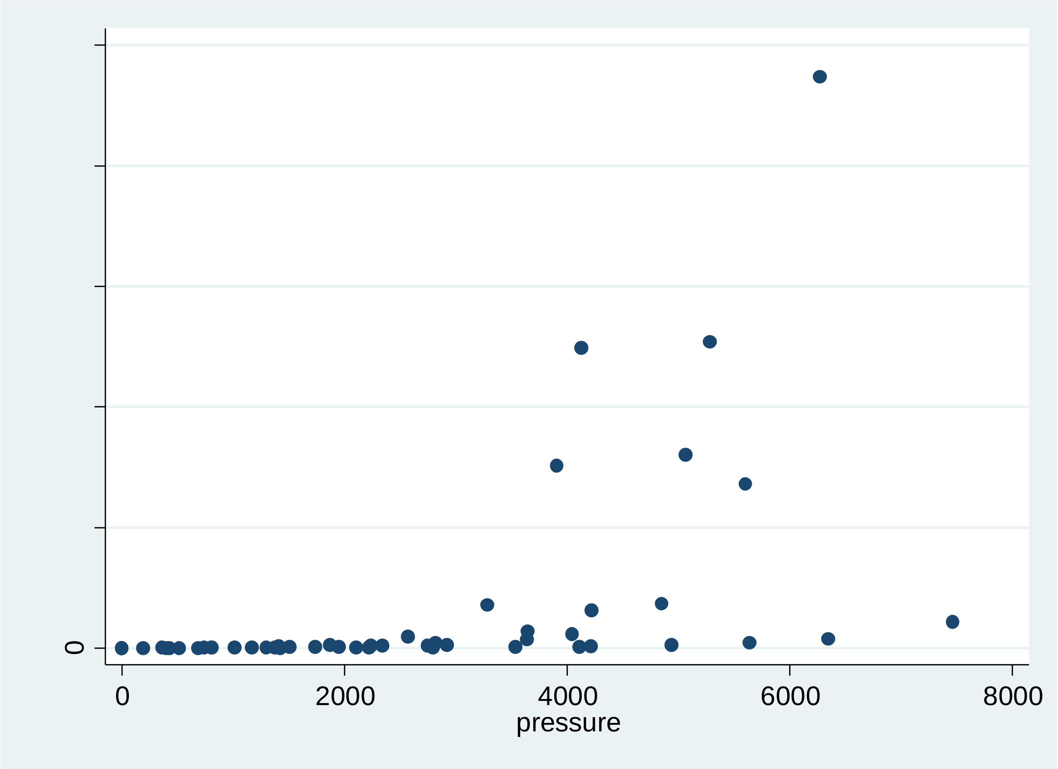


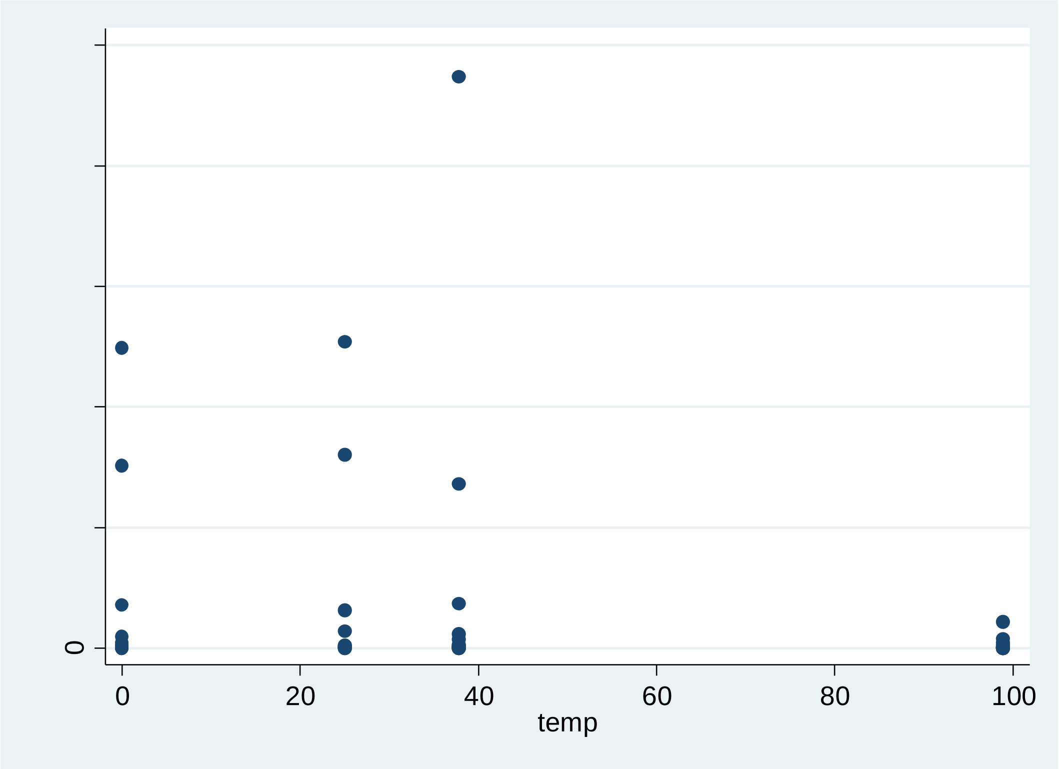
Linear regression diagnostic: constant variance plot

If the assumption of constant variance were true, there should be no pattern here.



Viscosity as a function of pressure and temperature





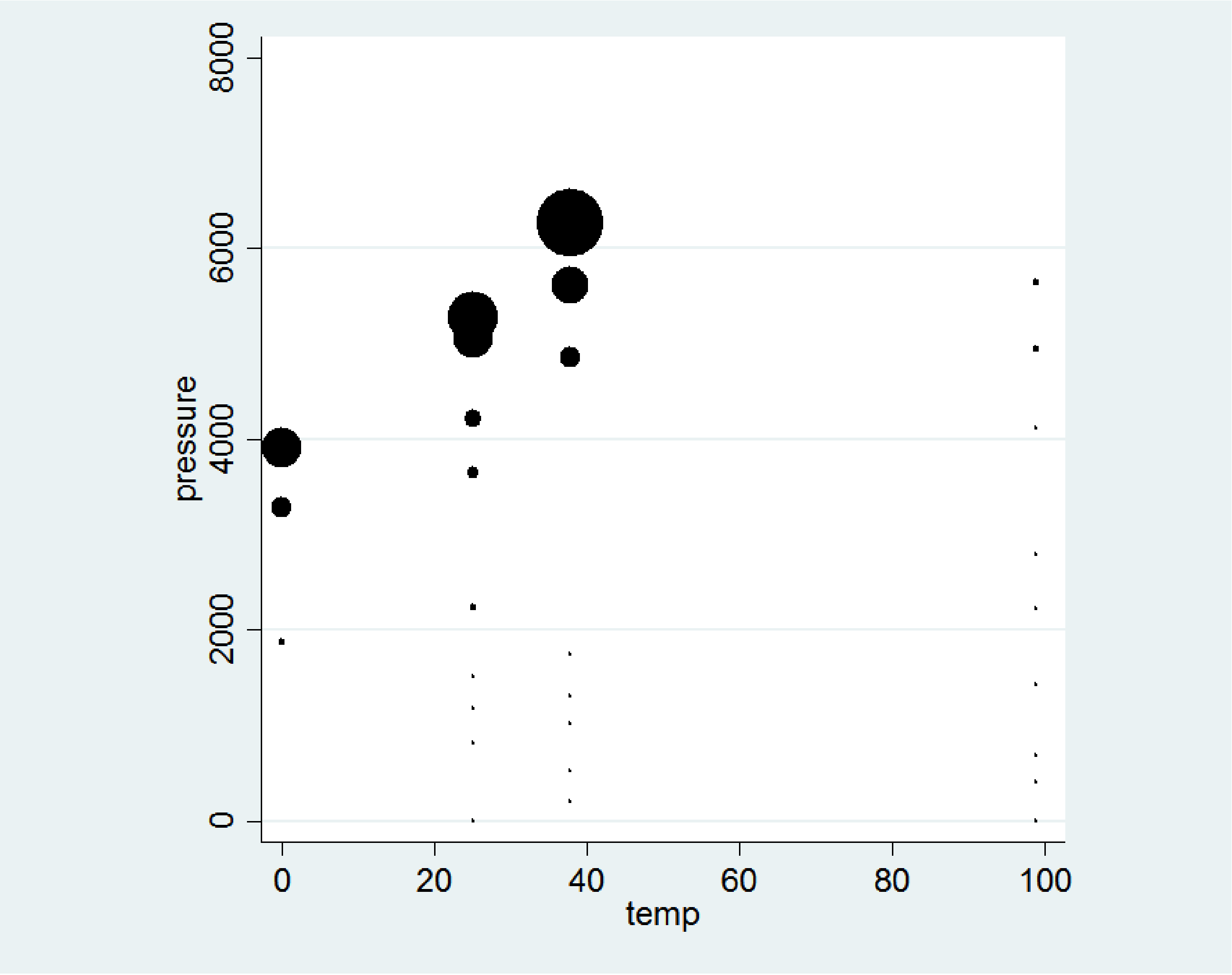
CART: **C**lassification **a**nd **R**egression **T**rees

We now introduce a different method.

Emphasis is on concepts rather than mathematical rigour.

CART: The data

Larger circle means greater viscosity (y).



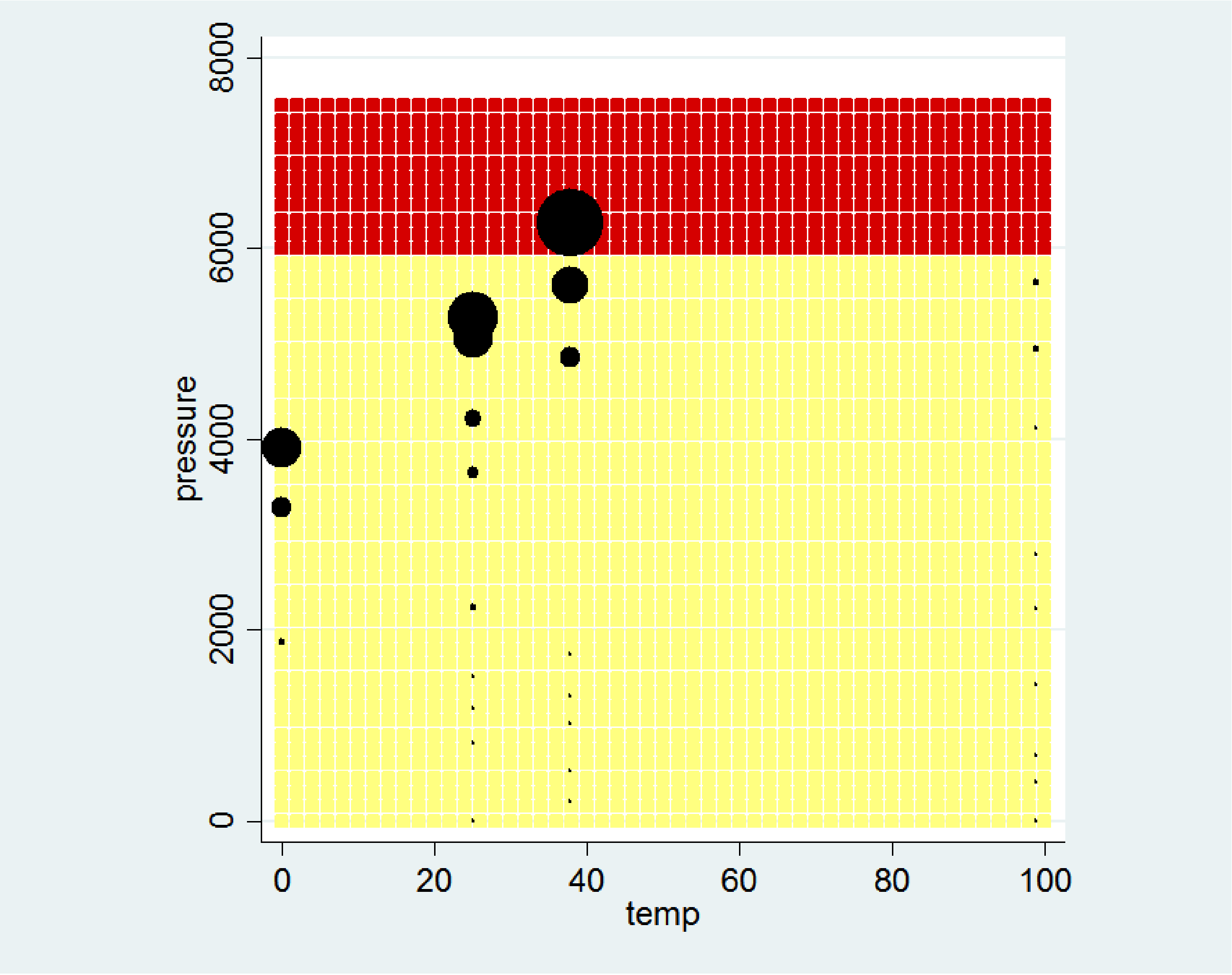
CART: split=1

The predicted values of viscosity are constant in the red area , and constant in the yellow area.

i.e. one split =two predicted values

For Gaussian distribution, the predicted value is the average viscosity of all obs in the same area.

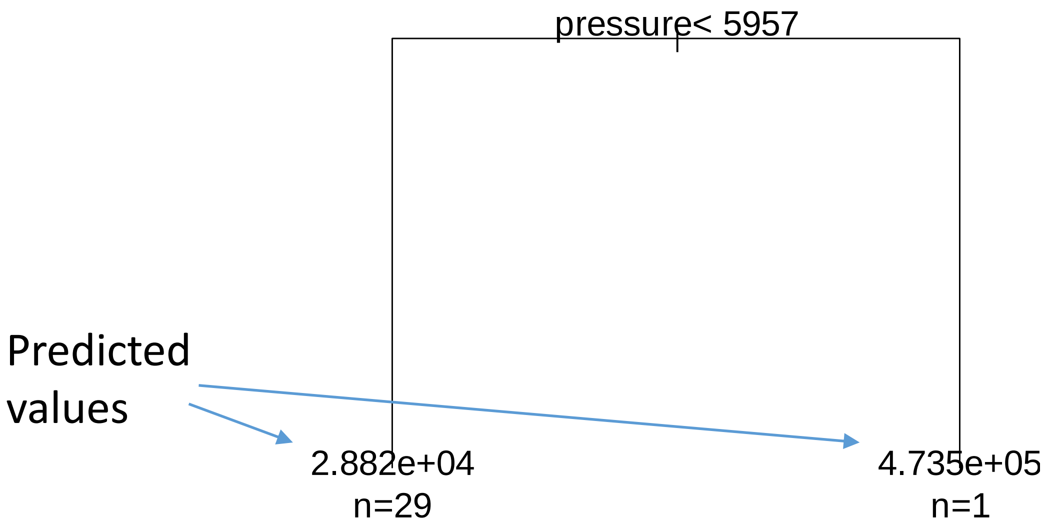
The best split is chosen across all variables and all values for each variable such that residual sums of squares a minimized.

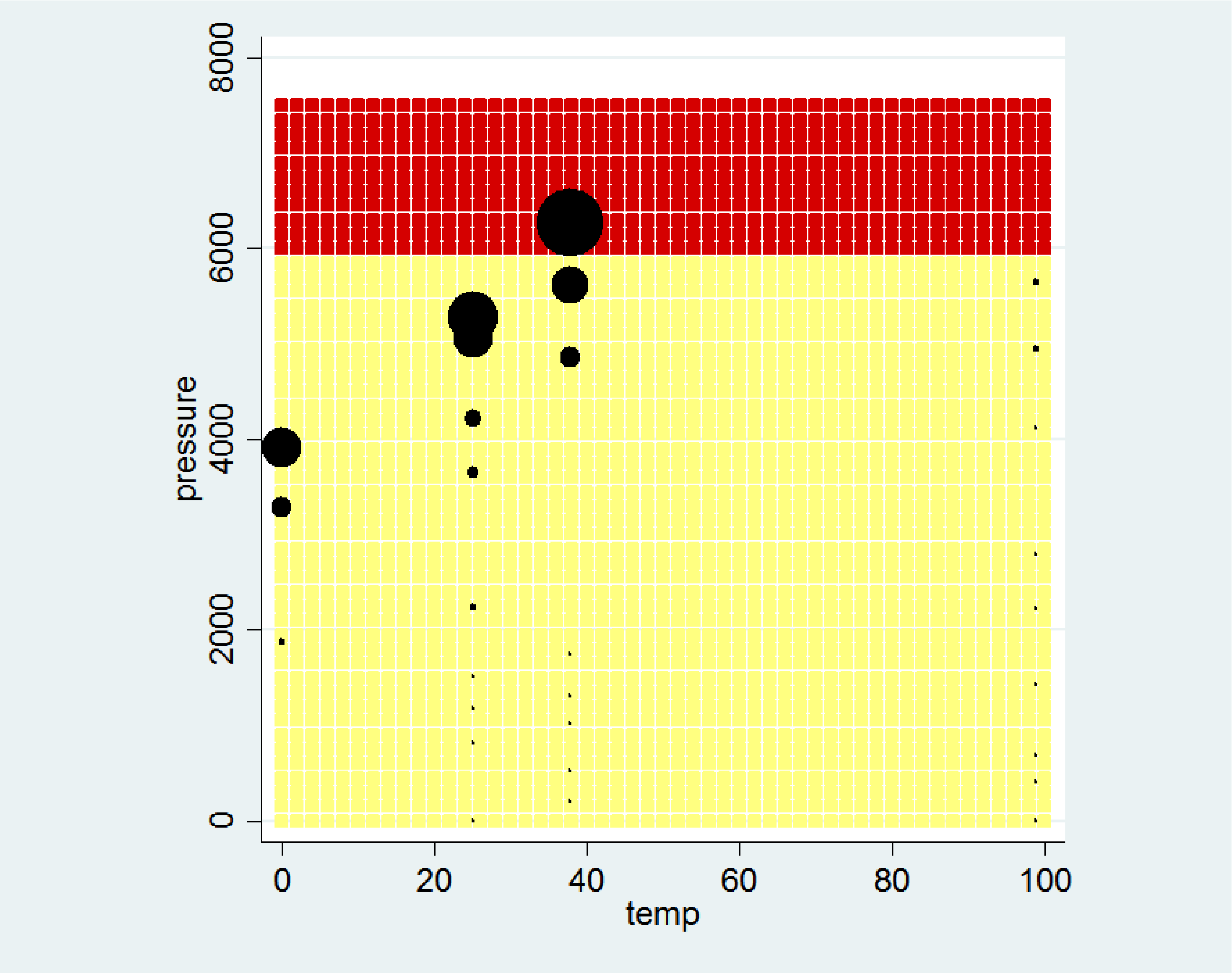


The plot is a scatterplot with plotting symbols of squares.

CART: split=1

Corresponding tree

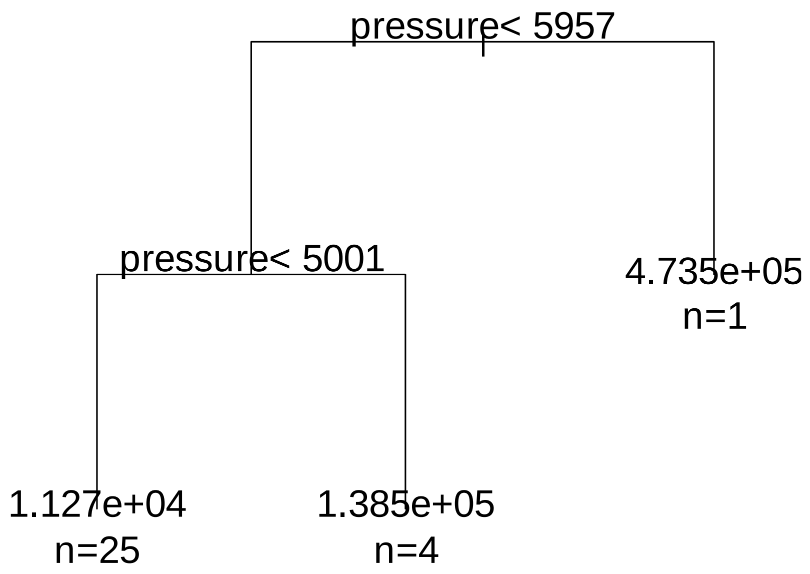


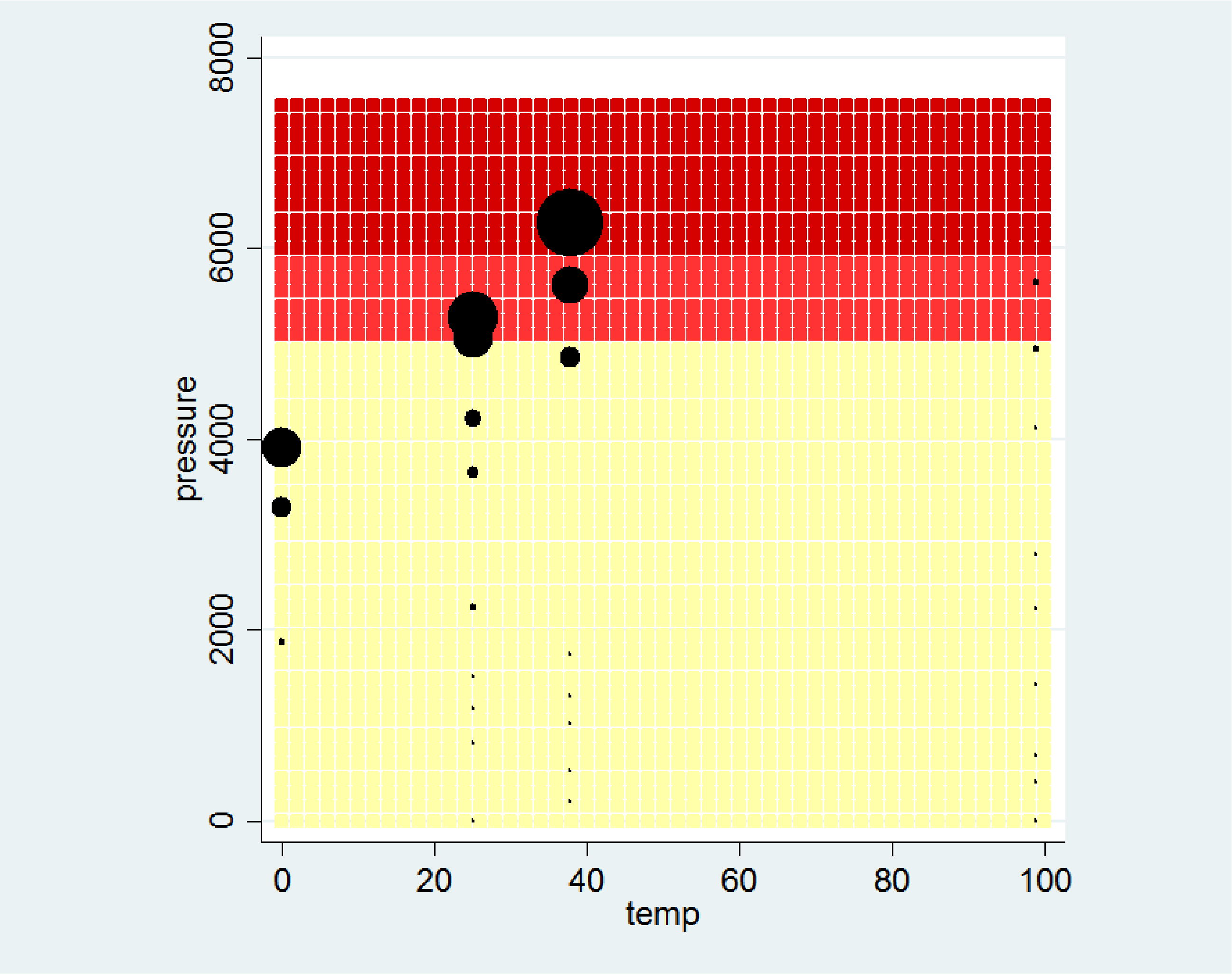


The tree graph was produced in R as Stata has no such program. The exact split point may appear to be slightly different on these graphs.

CART: split=2

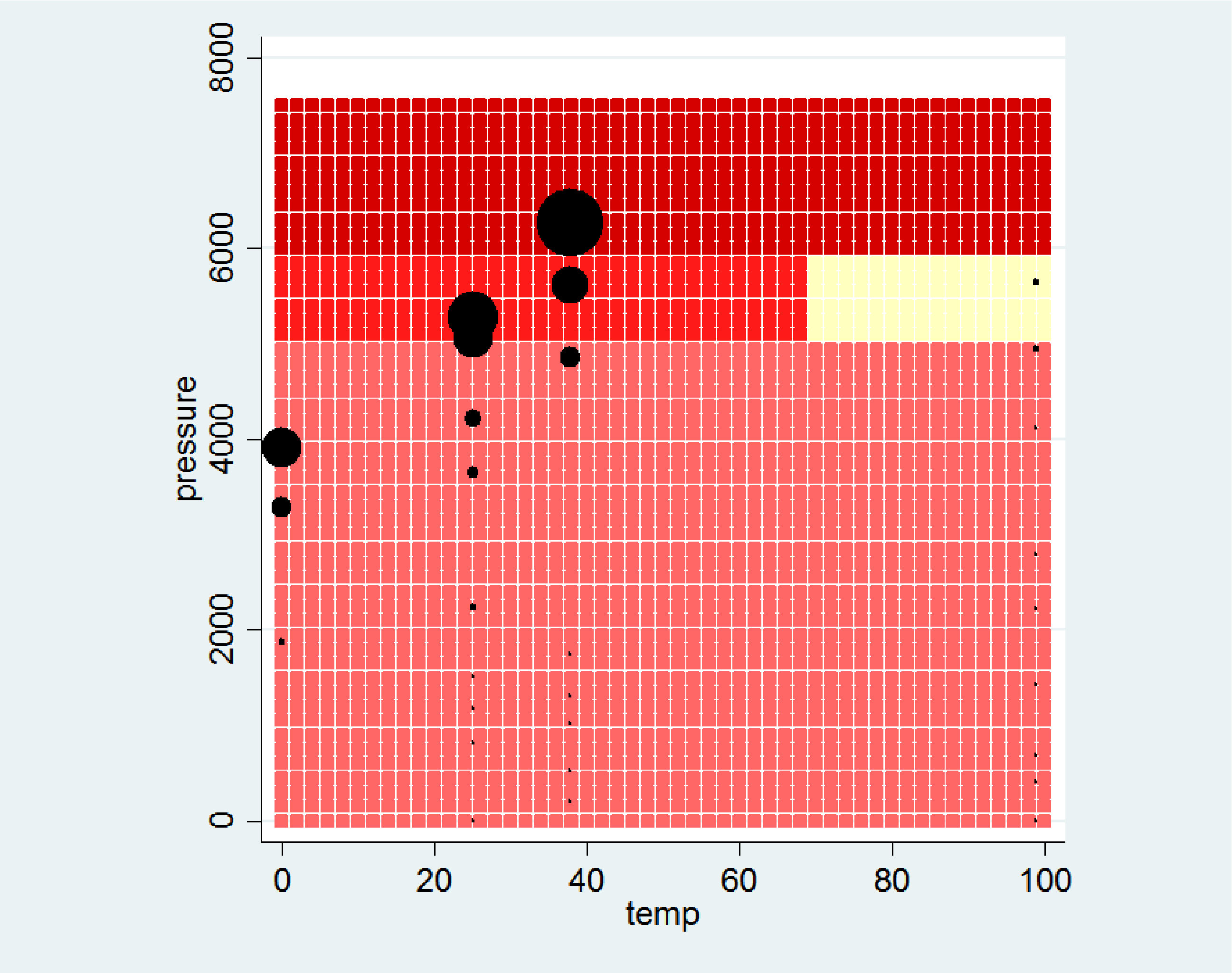
Tree





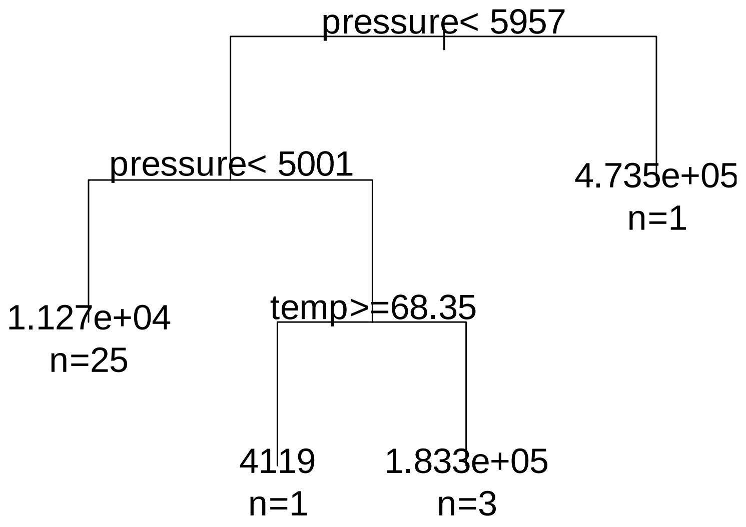
CART: split=3

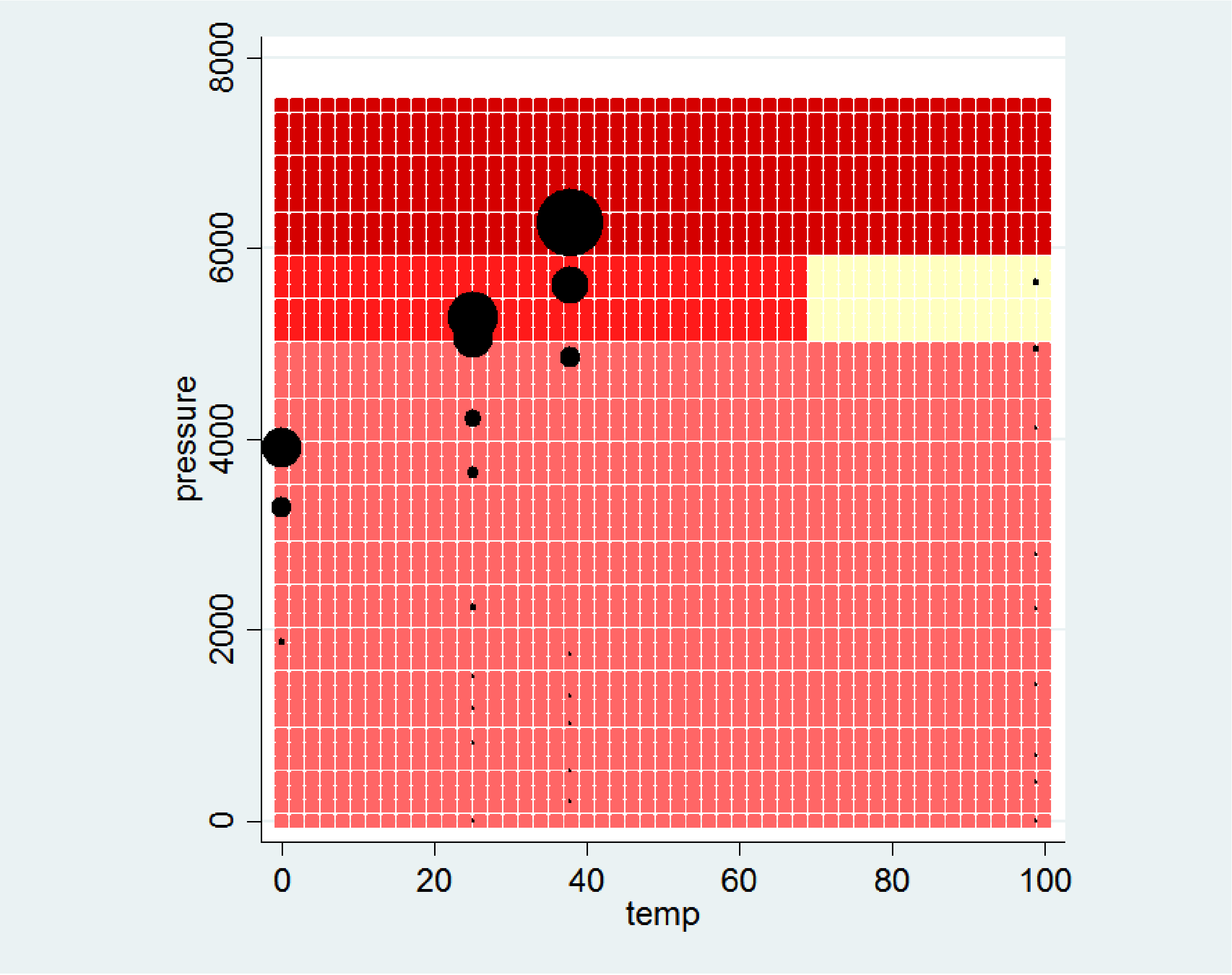
- The intial 2 splits were on pressure
- This split is on temperature in the middle pressure region.
- Because the split depends on the region for pressure, this represents a two-level interaction
- Potential for interactions= number of splits



CART: split=3

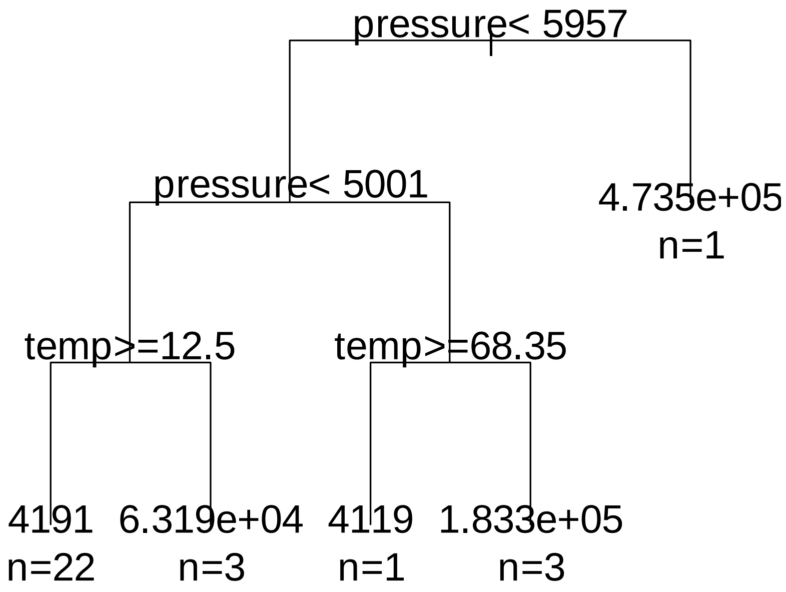
.


pressure< 5957
pressure< 5001
4.735e+05
n=1
1.127e+04
n=25
temp>=68.35
4119
n=1
1.833e+05
n=3

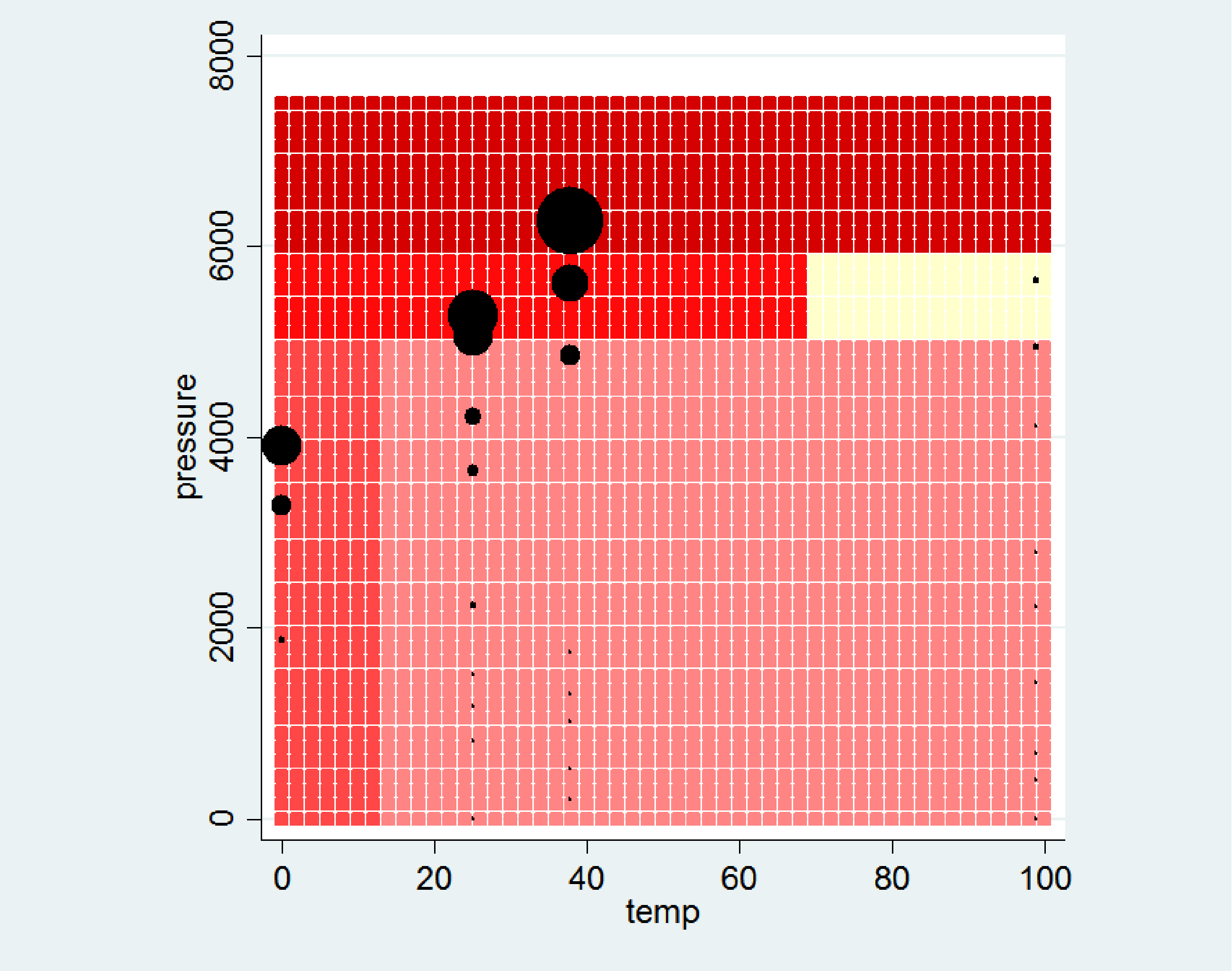

pressure
0
2000
4000
6000
8000
0
20
40
60
80
100
temp

CART: split=4

.



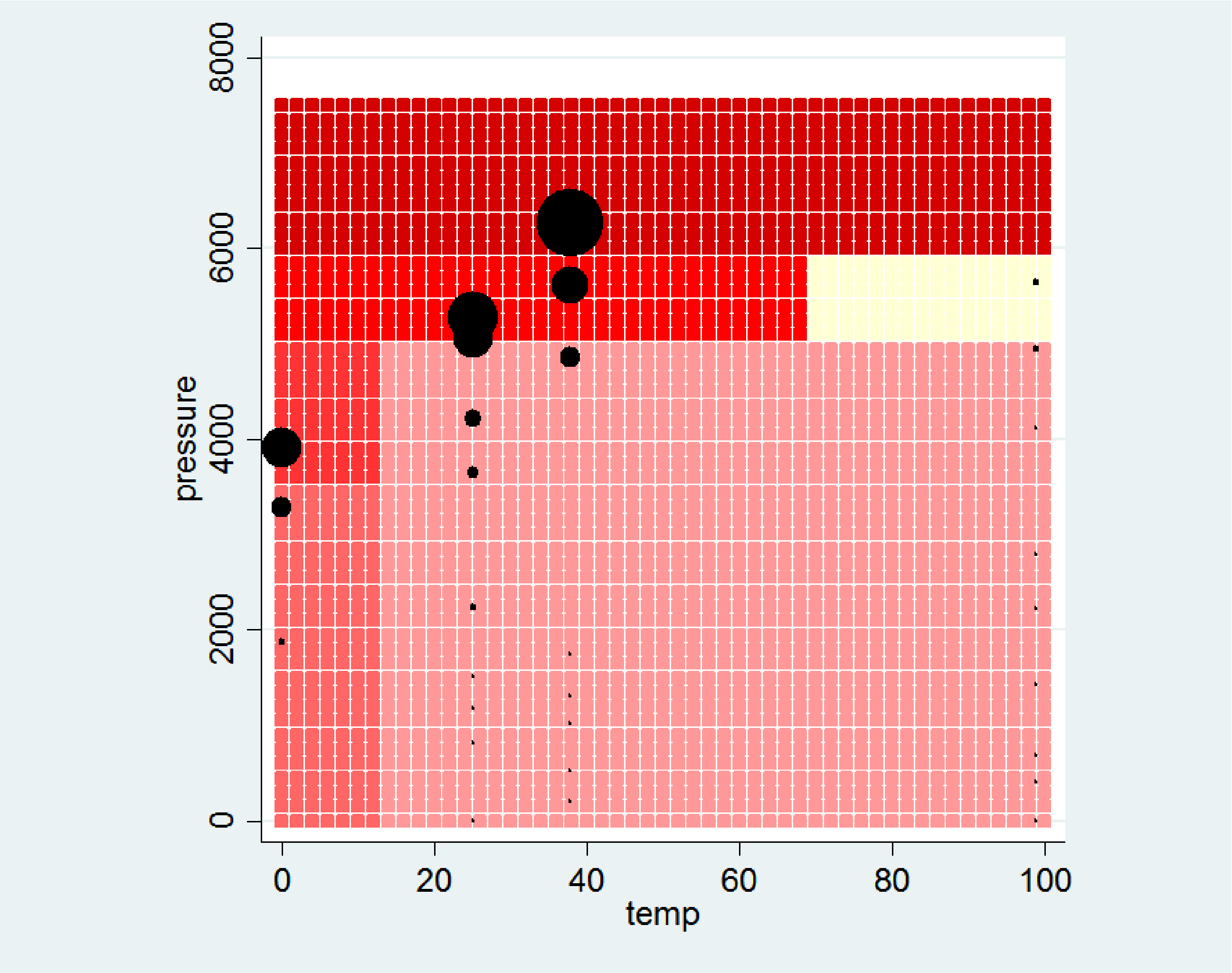
pressure< 5957
pressure< 5001
4.735e+05
n=1
temp>=12.5
temp>=68.35
4191
n=22
6.319e+04
n=3
4119
n=1
1.833e+05
n=3



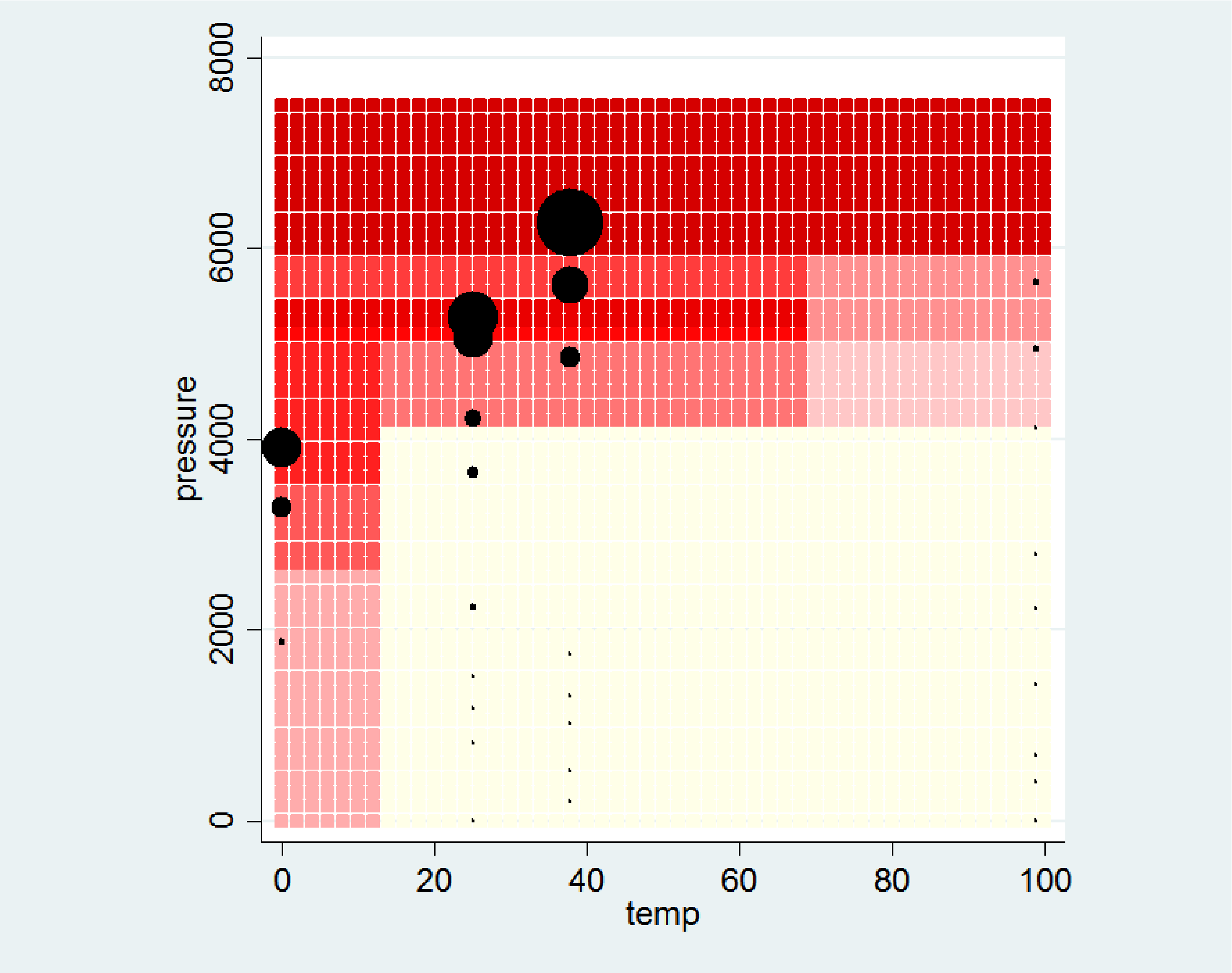
pressure
0
2000
4000
6000
8000
0
20
40
60
80
100
temp

CART: split=5

More intense shades of red correspond to larger predicted values.

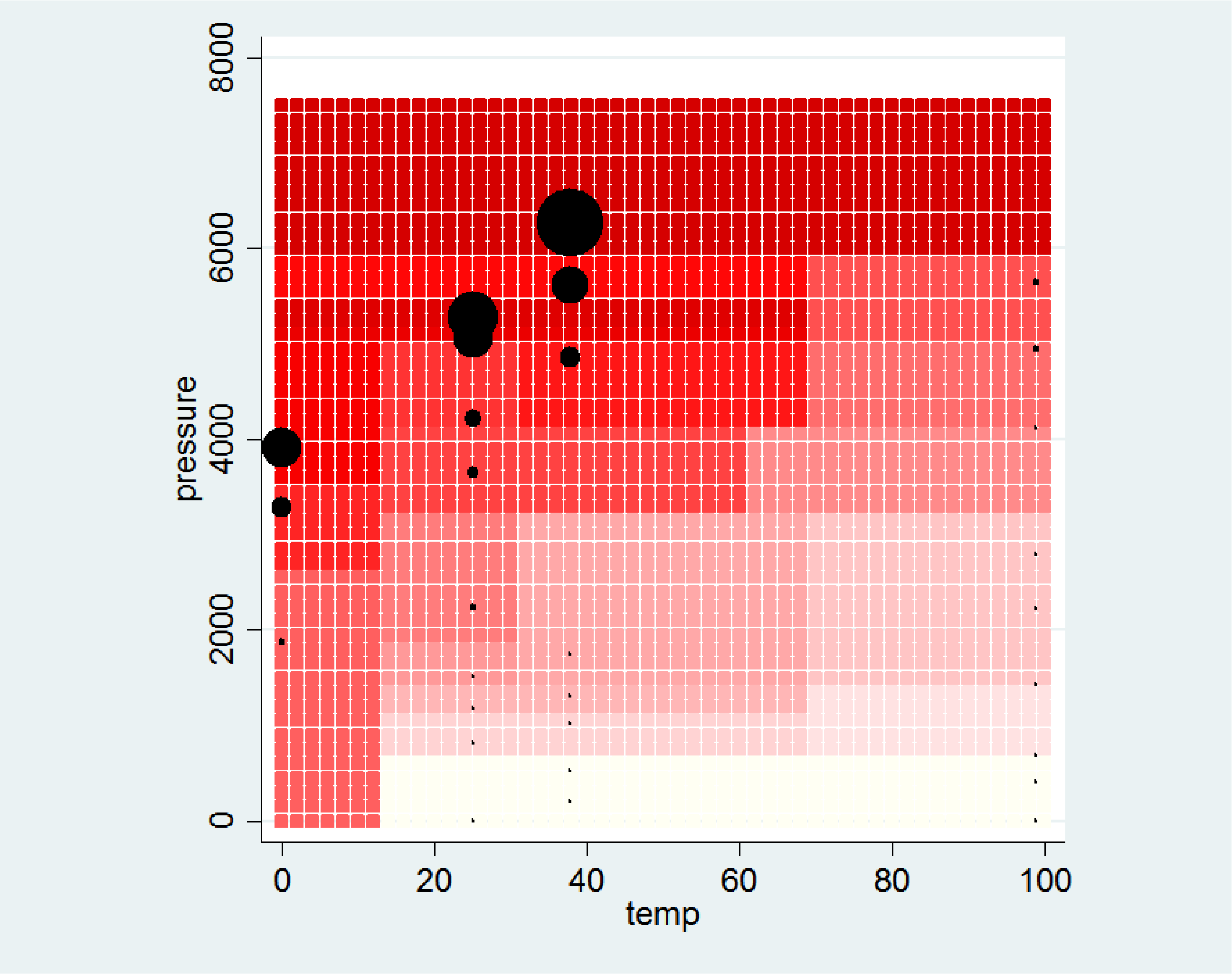


CART: split=10



CART: split=20

.

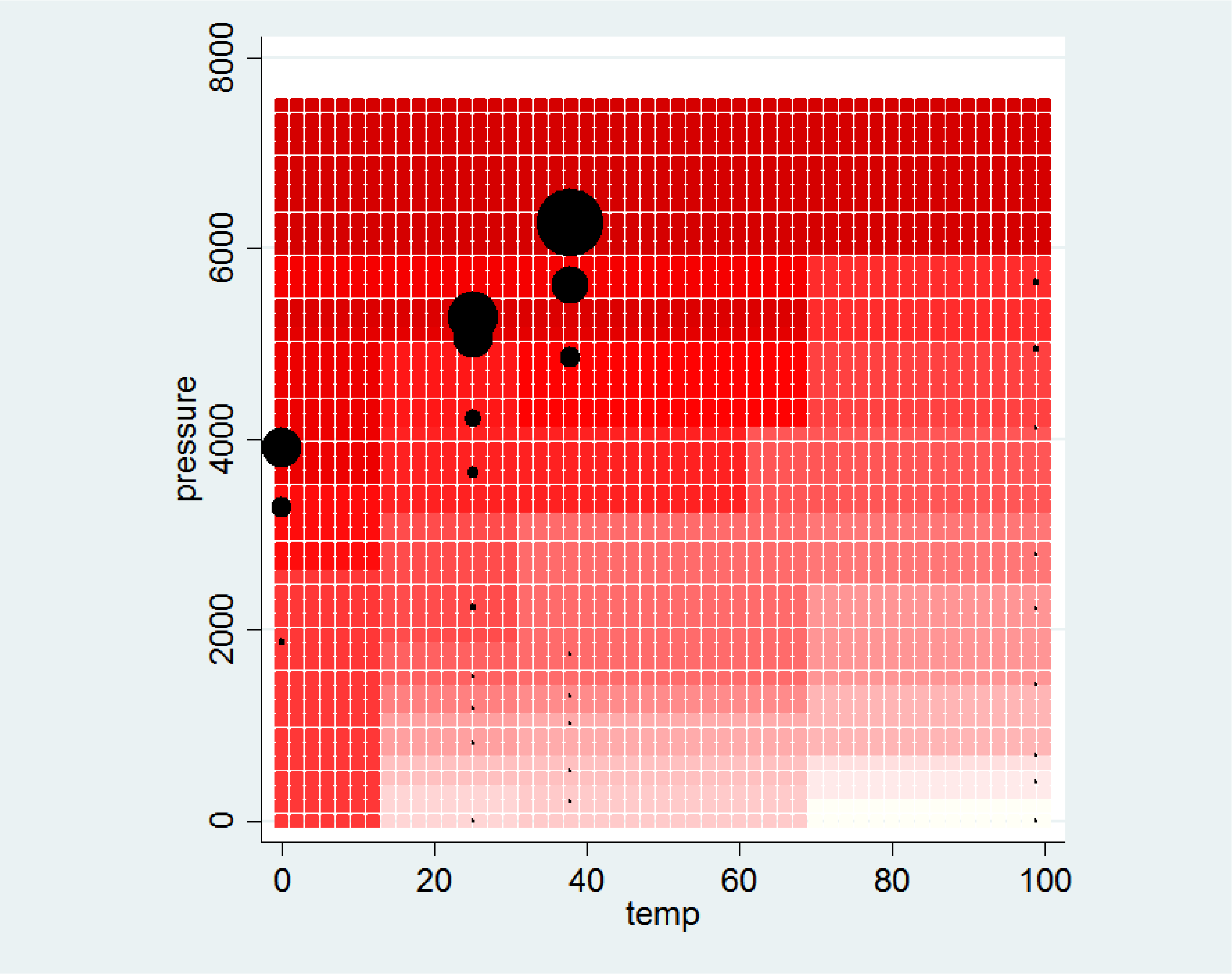


CART: split=28

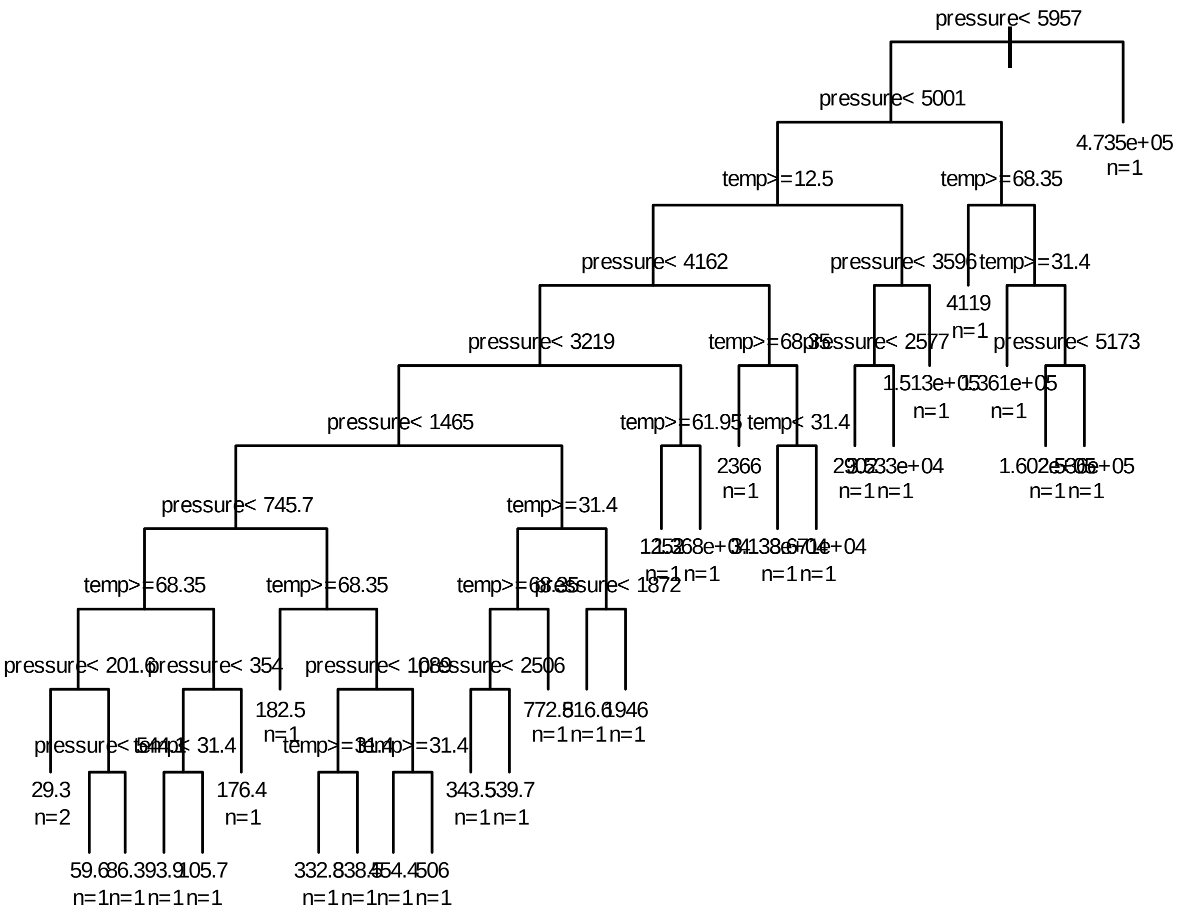
28 splits result into 29 regions.

N=30

almost each observation gets its own region.



CART: all possible 29 splits



- Every data point has its own terminal leaf.
- Perfect prediction (on this data)
- The tree appears unbalanced because mostly small regions are chopped off the one large region

Overfitting

- Overfitting occurs when the model “hugs” the data so well that it predicts perfectly on the given data - but not a fresh data set
- Overfitting is much less of a danger in Linear models because the linearity constraint mostly prevents overfitting.
    - Overfitting may occur in linear models when too many interactions or polynomial (squared etc) terms are requested.

Strategy against Overfitting

- Split the data into two data sets: training and test data
- Develop a model using the training data only
- Assess the model using the test data only
  - Here: choose the number of potential interactions (=splits) based on performance on the test data

Strategy against Overfitting

- Two data sets don’t necessarily have to be equal size
  - e.g. 80% training vs 20% testing is allowed
  - The (small) Ethyl data were split into 30 and 23 data points.
- Training and Test data should be random
  - complications arise when data are sorted
  - (e.g. all high y-values first)

(Gradient) Boosting

We next introduce boosting as approximating a function sequentially with a sequence of regression trees.

One a regression tree is added, it is never changed.

Boosting for squared error loss

1. Initialize f_0 with a constant value $$f_0(x) = \frac{1}{n}\sum_i y_i$$
2. For m=1 to M:
   a) Compute residuals $$r_{im} = y_i - f(x_i)$$

   b) Fit a regression tree (with J terminal nodes) to the residuals giving terminal leaves R_{jm}

   a) For each terminal node, compute fitted values
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} \left(y_i - [f_{m-1}(x_i) + \gamma]\right)^2$$

   d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output f_M as estimate of f

   $$\hat{f}(x) = f_M(x)$$

Boosting for squared error loss

- We want to approximate the function slowly rather than all at once.
  - Typical values for m are in the 100's or in the 1000's.
- Any one tree does not need to be complicated (few splits).
- Any one tree only needs to improve the approximation a little bit.

- By introducing an arbitrary loss function this algorithm can be generalized.

MART algorithm for (gradient) boosting

For an arbitrary loss function L

1. Initialize f_0 with a constant value $f_0(x) = \arg\ \min_\gamma \sum_{i=1}^{n} L(y;\gamma)$
2. For m=1 to M:
   a) Compute pseudo residuals $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]$

   b) Fit a regression tree (with J terminal nodes) to the pseudo residuals giving terminal leaves R_{jm}
   a) For each terminal node, compute fitted values

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

   d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
3. Output f_M as estimate of f $\hat{f}(x) = f_M(x)$

Tuning parameters

- Parameters that one can change or “tune” are called tuning parameters
- Boosting two naturally arising tuning parameters:
  - M, the number of iterations or trees
  - J, the number of leaves per regression
    - Equivalently specify J-1 splits or interactions
- And two optional tuning parameters
  - Fraction of the data used for bagging
  - extent of shrinking

Number of trees/iterations M

- Monitor predictive performance on a test data set .
- Initially, predictive performance will increase and then degrade.
- Choose M as the number of iterations that maximizes performance on the test data
    - This is done automatically in my implementation of boost.

Number of splits/ interactions

- Interaction=1
  - can only split on a single variable.
- Interaction=2
  - Could split on two variables
  - Could split on a single variable at two different values
- And so forth

- Typically good values for interaction are between 3 and 10
- The same value is used for all trees.

Shrinking

- To avoid overfitting one might want to fit slowly. Shrink the fitted value by multiplying it with ν

$$f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jkm} \mathrm{I}(\mathrm{x} \in \mathrm{R}_{jkm}) \quad , \quad 0 \le \nu \le 1$$

Where γ_{jkm} is the constant predicted value in region R_{jkm}

- Shrinking slows down the learning process but substantially increases runtime
- v=1 correspond to no shrinking
- Typical values of v are 0.1, 0.01, 0.001.

Bagging

- Goal: reduce variance
- **Bag**ging = **B**ootstrap **Ag**gregation
- Method:
  - Create bootstrap replicates of the data and fit a model to each replicate.
  - Average predictions over all models
- Properties:
  - Stabilizes unstable methods (e.g. CART)
  - Easy to implement, can be parallelized
  - No interpretation (Black box method)

A Bagged version of boosting

- Bagging can be applied to boosting as follows:
- For each iteration, fit the model to a random subset of the “residuals” r_{im}
- This is not “pure” bagging, because there is only one bagged data set at each iteration.
  - No averaging needed
- A typical value for bagging is 0.5 (50%) .

Comparison on Mean squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y-\hat{y}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{r}^2$$

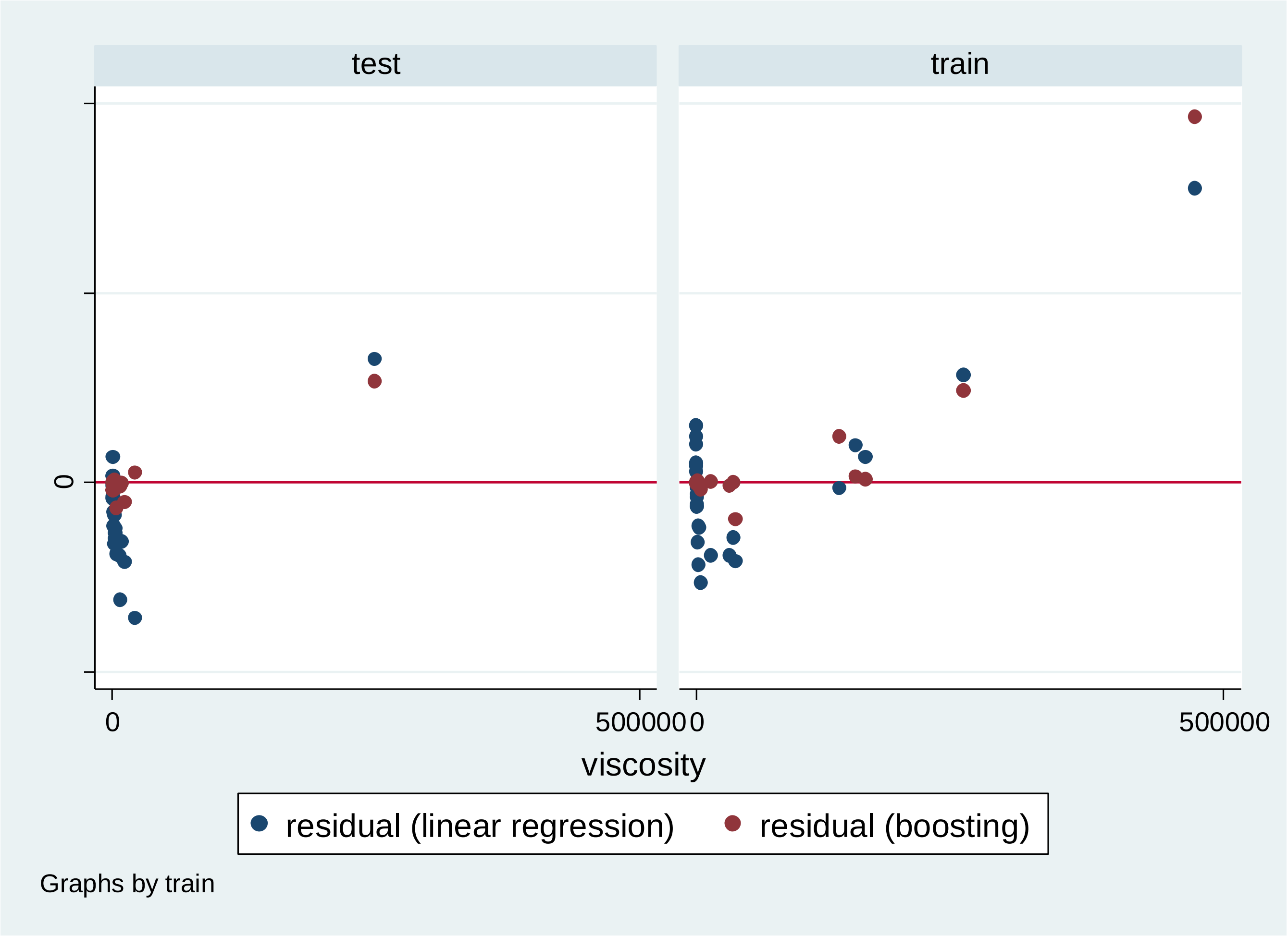
- The mean squared error is the average of the squared residuals.
  - To compare to boosting, we do not use the usual (n-p) in the denominator as in linear regression

Calculating the MSE in stata :

```
boost viscosity temp pressure if train, dist(normal) pred(predb)  shrink(0.1) inter(3) bag(0.5)
bysort train: egen mse=total(((y-predb)^2)/`trainn')
```

Boosting has much smaller residuals

- linear regression MSE= 31.5 *10^8
- Boosting MSE= 4.4 *10^8
- Both MSE's refer to the test data (N=23)



Does boosting still do better when regressing on log(y)?

- You might notice that y spans several orders of magnitude. The standard recommendation then is to regress on the log transformed variable log(y).

Result:

- linear regression MSE= 0.80
- Boosting MSE= 0.14
- Both MSE’s refer to the test data (N=23)

- If first log-transforming the response , boosting still does better

Example: Propensity scoring

Statistical learning is useful for Propensity scoring

- Propensity scoring is a tool for causal inference.
- A propensity score is the predicted value of a logistic regression.

$$\log \frac{p_i}{1-p_i} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

Where p_i is the probability of being in the treatment group.

- The goal is to estimate the probabilities well
  - (and to create “balance”)
  - The goal is not explanation of the relationship between x and treatment.
- This problem is well suited for black-box statistical learning methods

Death Penalty in the USA

Question: Is the decision to seek the death penalty arbitrary?

i.e., is it related to the race of the defendant or victim after adjusting for case characteristics?

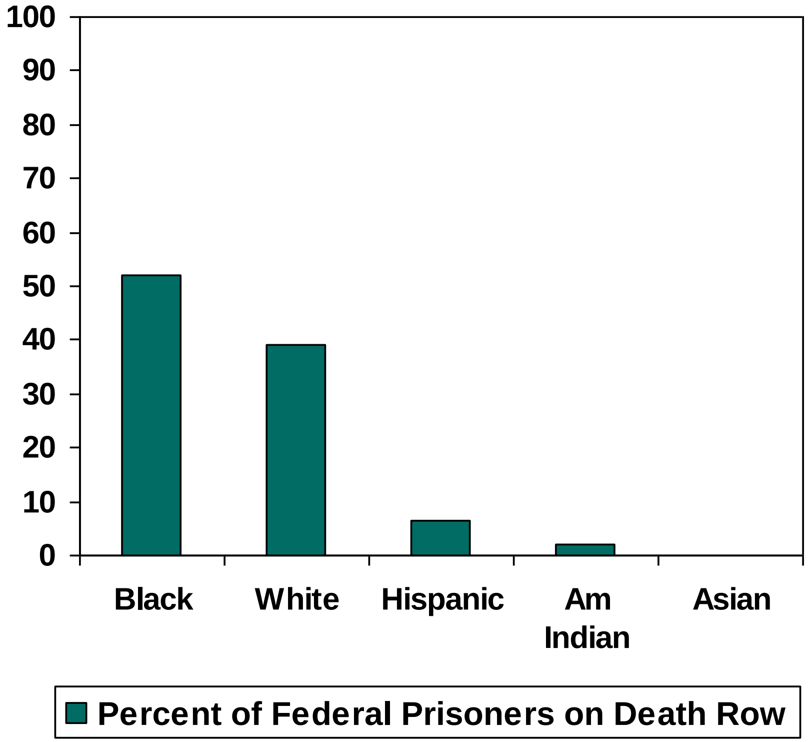
Data from 1995-2000 (federal cases only)

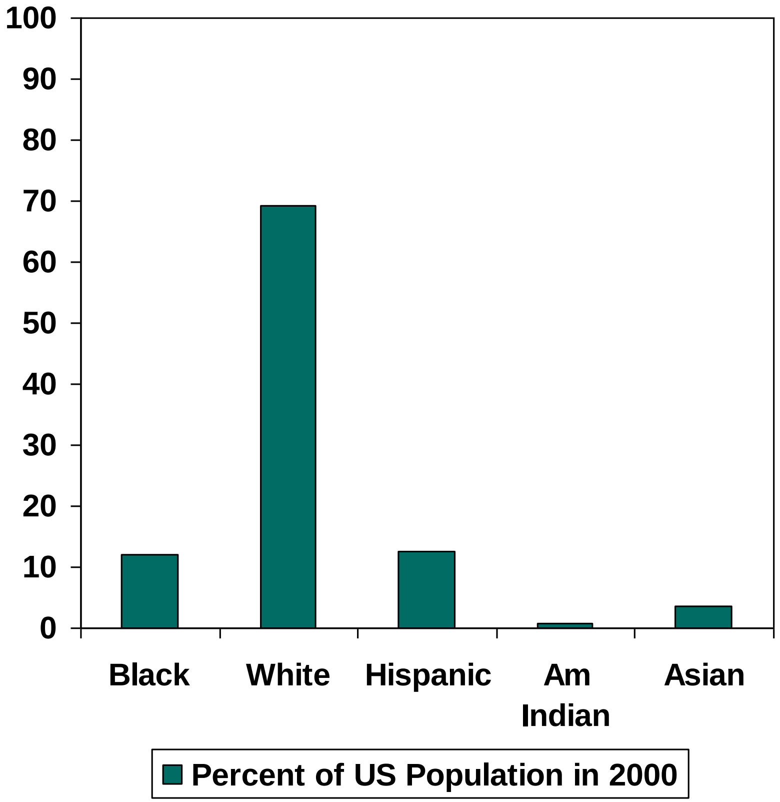
N=403

Death Penalty in the USA : Background

- A Capital Crime May Be Prosecuted Under Either State or Federal Law
- The vast majority of capital crimes are prosecuted under state law
- Federal cases tend to be more complex, involving…
  - multiple defendants
  - multiple victims
  - elaborate ongoing criminal activities

Death Penalty in the USA

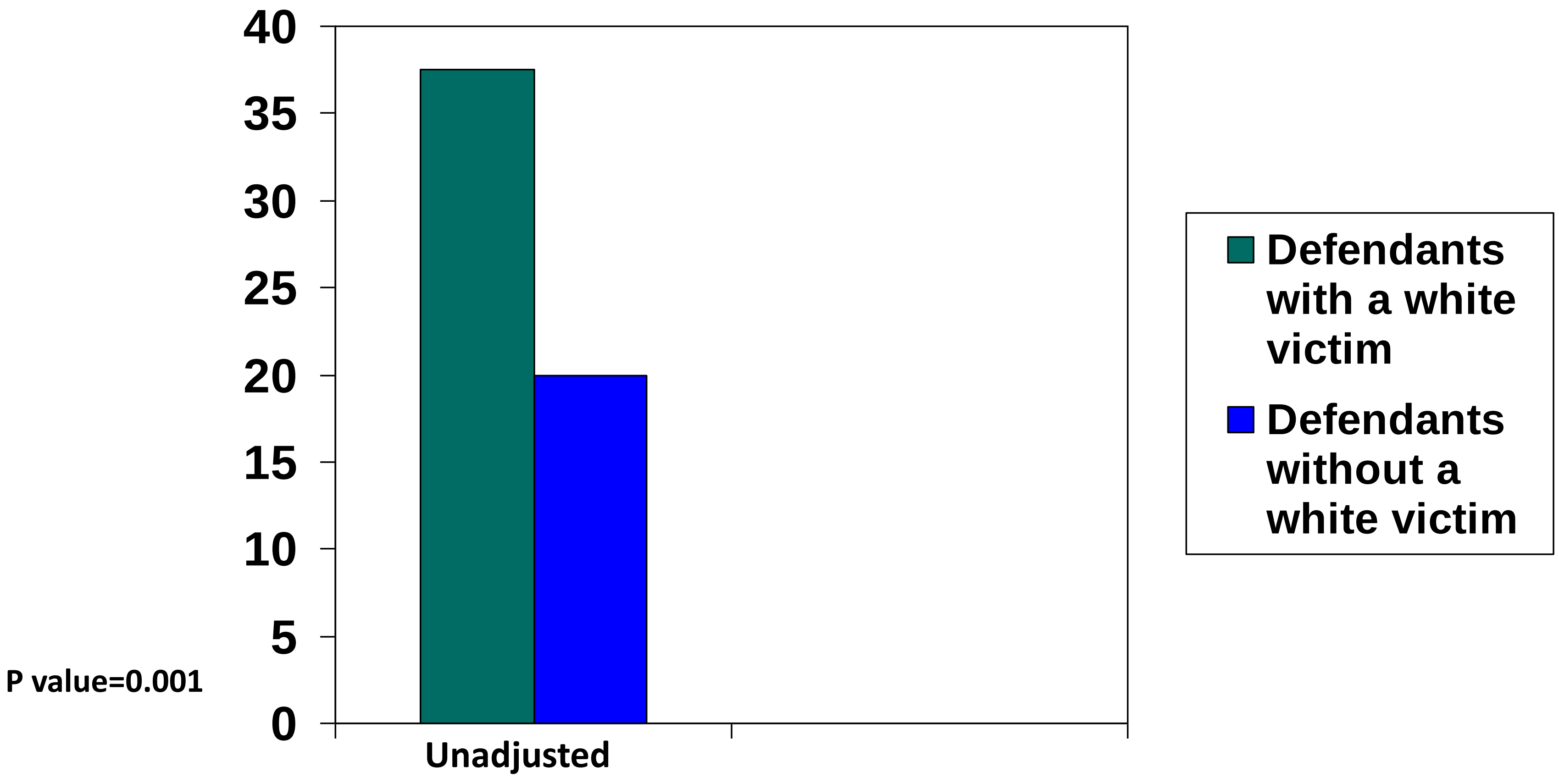




If race is a factor for the death penalty for either *defendant* or *victim*, then there is racial inequity

The Raw Numbers Seem to Point to Racial Inequity in the Federal System

Without Adjusting for Characteristics of the Crime, Large Significant Race Effects Exist



Percentage of defendants for which the Attorney General sought the death penalty

To Assess Racial Inequity, We Needed to Compare Cases with Similar Crimes

- For each defendant with a white victim, we found a defendant without a white victim who had committed a comparable crime
- Comparable crimes are those with similar heinousness
- Any differences in decisions to seek the death penalty could not be due to observed characteristics of the crime
- Victim Race was the only remaining factor

Death Penalty

- Logistic regression
- 103 x-variables
- The large number of x-variables create some problems.

```
. logistic $treat `xvar' $race2  if train

note: mfvconsentagrc_any != 0 predicts failure perfectly
      mfvconsentagrc_any dropped and 16 obs not used

note: evidprvwpn_any dropped because of collinearity

Logistic regression                             Number of obs     =        403
                                                LR chi2(100)      =     277.19
                                                Prob > chi2       =     0.0000
Log likelihood = -114.06293                     Pseudo R2         =     0.5485

------------------------------------------------------------------------------
      whtvic | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    v1female |   2.308138   1.693927     1.14   0.254     .5477353    9.726415
    sympvic1 |   .5215818   .4338366    -0.78   0.434     .1021686     2.66273
   crimedoer |   .0445804   .0842127    -1.65   0.100     .0010996     1.80747
    vunder17 |   3.443886   3.738736     1.14   0.255     .4101768    28.91521
     vover60 |   1744.724   3655.227     3.56   0.000      28.7372    105927.6
    vmarried |   1.424964   .8657659     0.58   0.560      .433151    4.687796
vworkcrim_~y |   .9180076   1.036818    -0.08   0.940     .1003414    8.398703
     vulvic1 |   .0023684   .0039253    -3.65   0.000      .000092     .060986
vfmilitary~y |   15.34566   21.74452     1.93   0.054     .9546595    246.6736
vfprison_any |   4.97e+09          .        .       .            .           .
   vdisabled |   4.107626   7.226138     0.80   0.422     .1306642    129.1294
vfabusedd_~y |   .6833503   .7480495    -0.35   0.728     .0799559    5.840314
vfinforman~y |   10.66977   8.082034     3.13   0.002     2.417657    47.08852
shotstabal~m |   .0694539   .0738036    -2.51   0.012     .0086534    .5574504
```

Death Penalty: Stepwise logistic regression

- Stepwise regression to reduce number of x-variables
- Backward stepwise regression:

```
. sw, pr(.1): logistic $treat `xvar' $race2  if train
between-term collinearity, variable evidprvwpn_any
```

  - Same problem when trying forward stepwise regression “sw, pe(.1): “
- Remove evidprvwpn_any, and then run stepwise again

```
//  remove "evidprvwpn_any“ and replace with " "
local xvar2= regexr("`xvar'","evidprvwpn_any"," ")
sw, pe(.1): logistic $treat `xvar2' $race2  if train
```

Death Penalty: Stepwise logistic regression

- Variables removed

```
. sw, pe(.1): logistic $treat `xvar2' $race2  if train
note: mfvconsentagrc_any dropped because of estimability
note: o.mfvconsentagrc_any dropped because of estimability
note: 16 obs. dropped because of estimability
                           begin with empty model
p = 0.0000 <  0.1000  adding  crimedoer
p = 0.0006 <  0.1000  adding  vfinformant_any
p = 0.0012 <  0.1000  adding  mfeqdefagrc_any
p = 0.0079 <  0.1000  adding  vover60
p = 0.0065 <  0.1000  adding  csslow_anyr
p = 0.0083 <  0.1000  adding  bkkidnap_anyr
p = 0.0018 <  0.1000  adding  akconceal_anyr
p = 0.0105 <  0.1000  adding  pubdang
p = 0.0175 <  0.1000  adding  evidweapon_any
p = 0.0147 <  0.1000  adding  shotstaball_anym
p = 0.0107 <  0.1000  adding  vulvic1
p = 0.0169 <  0.1000  adding  mfminpartagrc_any
p = 0.0108 <  0.1000  adding  agghagrc_sumwm
p = 0.0135 <  0.1000  adding  agghplanagrc_any
p = 0.0215 <  0.1000  adding  alcohol_dummy
p = 0.0043 <  0.1000  adding  mfimpcapagrc_any
p = 0.0268 <  0.1000  adding  anyclaim
p = 0.0434 <  0.1000  adding  agghprevofagrc_any
p = 0.0305 <  0.1000  adding  agghprevdthagrc_any
p = 0.0382 <  0.1000  adding  vfprison_any
p = 0.0223 <  0.1000  adding  evidwitness_any
p = 0.0693 <  0.1000  adding  headinjdis_dummy
p = 0.0935 <  0.1000  adding  idcnt4
p = 0.0746 <  0.1000  adding  agghprevfireagrc_any
p = 0.0851 <  0.1000  adding  timelong
```

Death Penalty: Stepwise logistic regression

- Variables remaining

```
Logistic regression                               Number of obs     =        403
                                                  LR chi2(25)       =     190.85
                                                  Prob > chi2       =     0.0000
Log likelihood =  -157.2345                       Pseudo R2         =     0.3777
```

whtvic	Odds Ratio	Std. Err.	z	P>|z|	[95% Conf.	Interval]
crimedoer	.2683635	.1012328	-3.49	0.000	.1281241	.5621032
vfinformant_any	8.80676	3.822697	5.01	0.000	3.76132	20.62016
mfeqdefagrc_any	.200342	.0768202	-4.19	0.000	.0944894	.4247771
vover60	86.72239	81.27761	4.76	0.000	13.81566	544.366
csslow_anyr	3.11581	1.19883	2.95	0.003	1.465768	6.623337
bkkidnap_anyr	.0936096	.0556223	-3.99	0.000	.0292107	.2999852
akconceal_anyr	3.982008	1.823835	3.02	0.003	1.622691	9.771662
pubdang	.4328603	.1870553	-1.94	0.053	.1855715	1.009681
evidweapon_any	2.845765	.9937227	2.99	0.003	1.435374	5.642001
shotstaball_anym	.3097303	.1336872	-2.72	0.007	.1329181	.7217438
vulvic1	.0749186	.0510093	-3.81	0.000	.019726	.2845389
mfminpartagrc_any	6.954793	3.957542	3.41	0.001	2.279912	21.21535
agghagrc_sumwm	2.106977	.3701996	4.24	0.000	1.493146	2.973154
agghplanagrc_any	.3071287	.1391332	-2.61	0.009	.1263908	.7463206
alcohol_dummy	.0938077	.0653772	-3.40	0.001	.0239338	.3676757
mfimpcapagrc_any	8.693805	7.190865	2.61	0.009	1.71854	43.9805
anyclaim	2.193278	.7342245	2.35	0.019	1.138006	4.227105
agghprevofagrc_any	17.94478	22.92028	2.26	0.024	1.468	219.3563
agghprevdthagrc_any	.0242166	.0326784	-2.76	0.006	.0017198	.3409902
vfprison_any	14.67863	12.84712	3.07	0.002	2.640533	81.598
evidwitness_any	.4360173	.1428879	-2.53	0.011	.2293798	.828805
headinjdis_dummy	2.282991	1.122871	1.68	0.093	.8706578	5.986333
idcnt4	.3800507	.1844	-1.99	0.046	.1468384	.9836561
agghprevfireagrc_any	.3256272	.2060958	-1.77	0.076	.0941843	1.125804
timelong	.1412831	.1605992	-1.72	0.085	.0152235	1.311194
_cons	1.009922	.4700196	0.02	0.983	.4056368	2.514425

Death Penalty

- Boosting
  - One might do a grid search for best values of tuning parameters. This would lead to even better results.

```
. * data were randomly sorted
. boost $treat `xvar' $race2,  dist(logistic) inter(5) predict(pred) shrink(0.01) train(0.7)
Distribution=logistic
predict=pred
Trainfraction=.7 Shrink=.01 Bag=.5 maxiter=20000 Interaction=5
xy_pred
Fitting ...
Predicting ...
bestiter= 1855
Test R2= .42504793
trainn= 420
Train R2= .99810995
```

Death Penalty : Prediction Accuracy

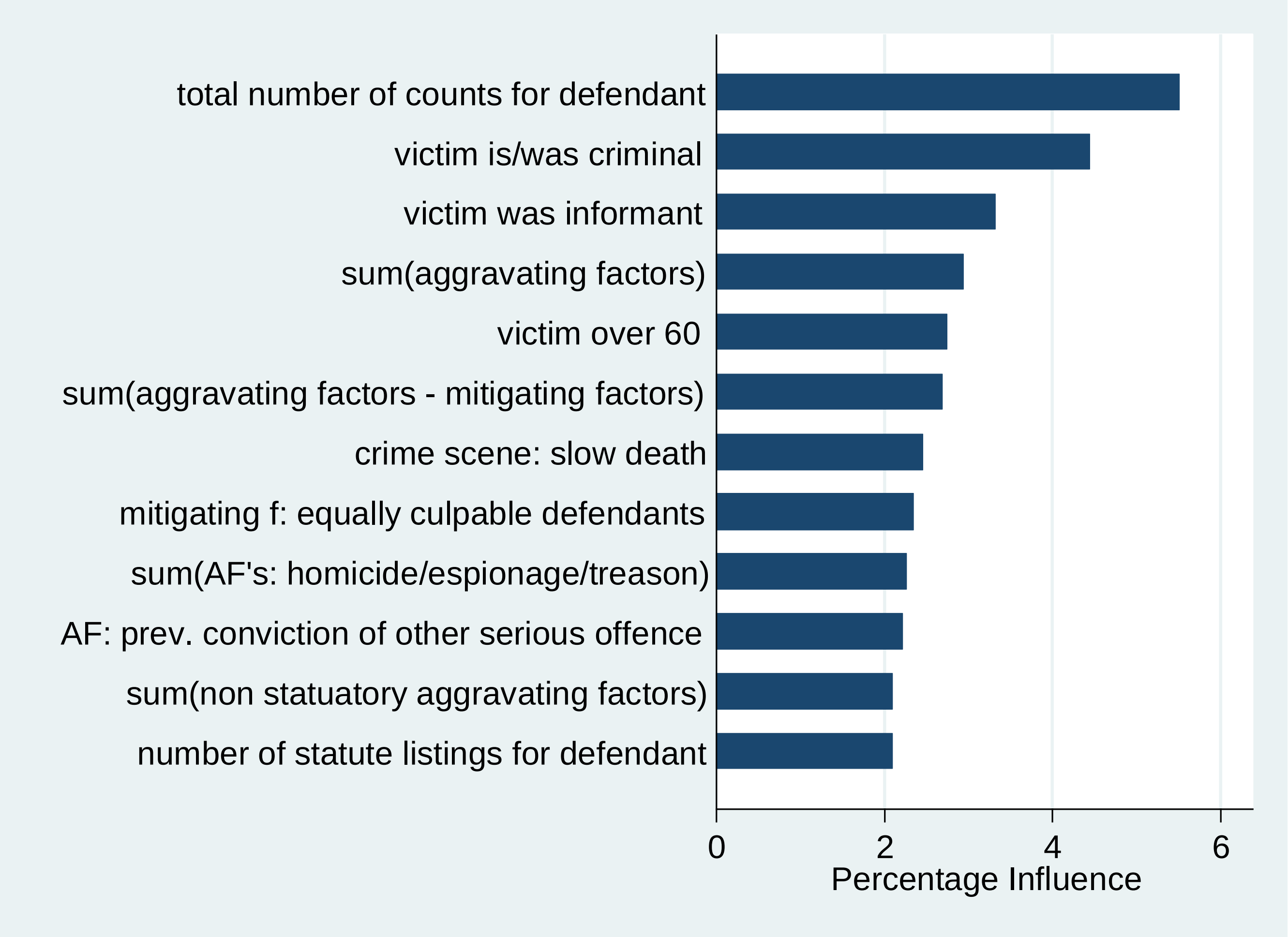
Whitevic	Logistic regression	Stepwise logistic	Boosting
Accuracy Training data	.835	.799	.997
Accuracy test data	.740	.729	.851

Influence

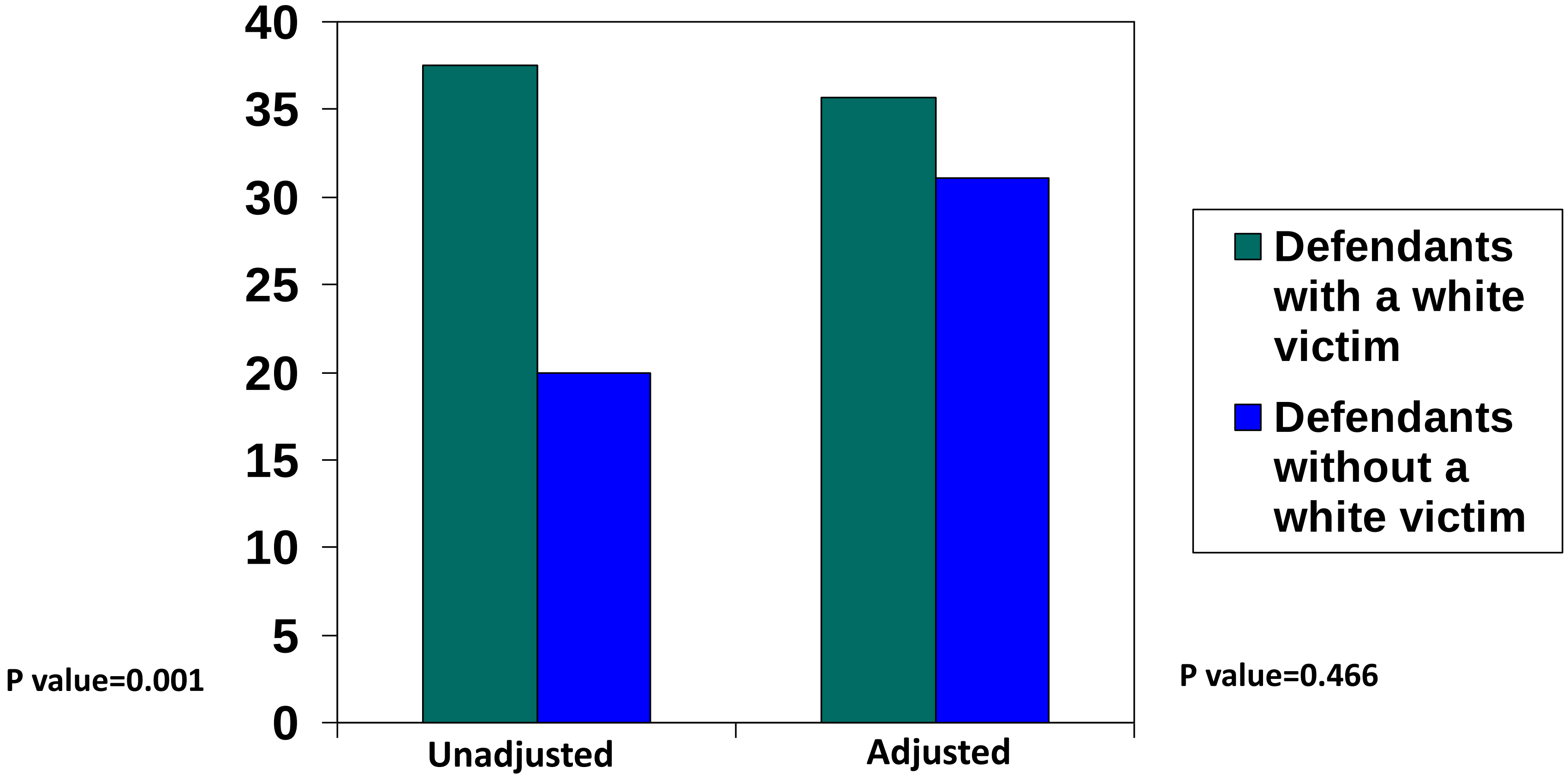
- Boosting is a black-box method.
- The relationship between x variables and y is not easily explainable.
- However, we can get aggregate measures of the influence of individual variables.
- The influence of a variable is the percentage the sums of squares explained by this variable of the total explained across all trees.

Death Penalty: Influence

- Influence on predicting "white victim"
- Listing only variables with at least 2% influence
- Interpretation: e.g. "victim is/was criminal" is predictive of "white victim".
  - Correlation could be positive or negative



Adjusting for Characteristics of the Crime, Significant Race Effects Disappeared



Percentage of defendants for which the Attorney General sought the death penalty

Boosting in Computer science

- The MART algorithm to boosting is the most popular boosting algorithm in Statistics.
- Developed by Statisticians at Stanford
- Analogous to Generalized linear models (GLM) it has versions for various distributions
  - GLM includes logistic regression, Gaussian regression, Poisson regression …
  - Boosted regression includes logistic boosted regression, Gaussian boosted regression, Poisson boosted regression …

Boosting in Computer science

- Many computer scientists with interests in machine learning, including professors, have never heard of this algorithm.
- Historical reasons:
  - The first boosting algorithm, Adaboost, was invented by Computer scientists (Schapire and Freund) for Bernoulli outcomes.
- Philosophical reasons:
  - Boosting is viewed as combining many week learning algorithm algorithms to one strong one.
- There are many different boosting algorithms in computer science.

Appendix

How to use the boost command for least squares CART predictions

- Specify a single tree: maxiter(1)
- Eliminate bagging and shrinking: bag(1) shrink(1)
- Specify the desired number of splits (or interactions) : inter(`inter’)
- Use all data for training [train(1)]
  - Boosting internally splits the data into training and test data

```
boost viscosity temp pressure, dist(normal) pred(pred1) train(1)
maxiter(1)  shrink(1) inter(`inter') bag(1)
```

References for the Death penalty study

Race And The Decision To Seek The Death Penalty In Federal Cases

Report #   ISBN: 0-8330-39966-0

Full report on the web:

http://www.rand.org/pubs/technical_reports/TR389/index.html

Public Data:

ICPSR 4355

http://www.icpsr.org/access/restricted/index.html

(requires a verification process to prevent abuse of the data)