# regressienceMD

*Lance J. Fernando*

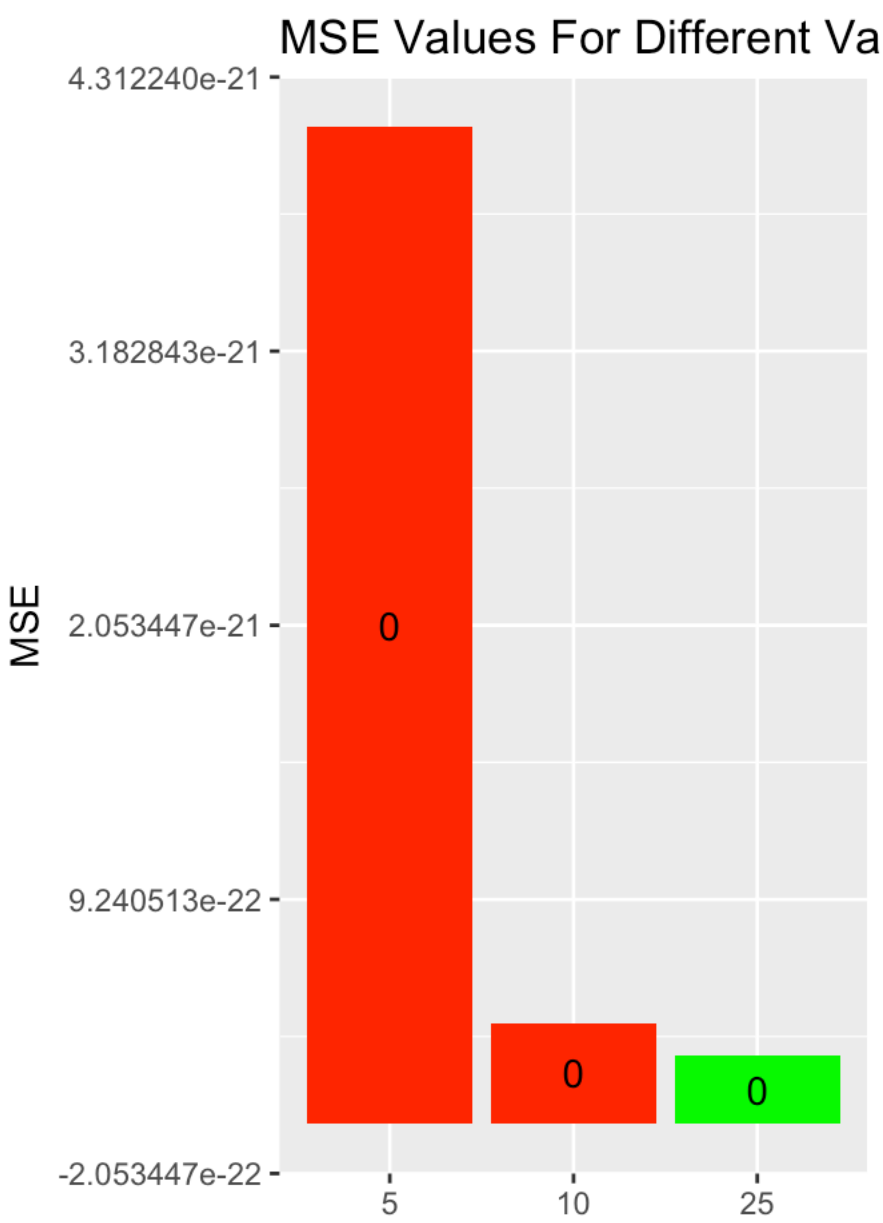
*5/8/2017*

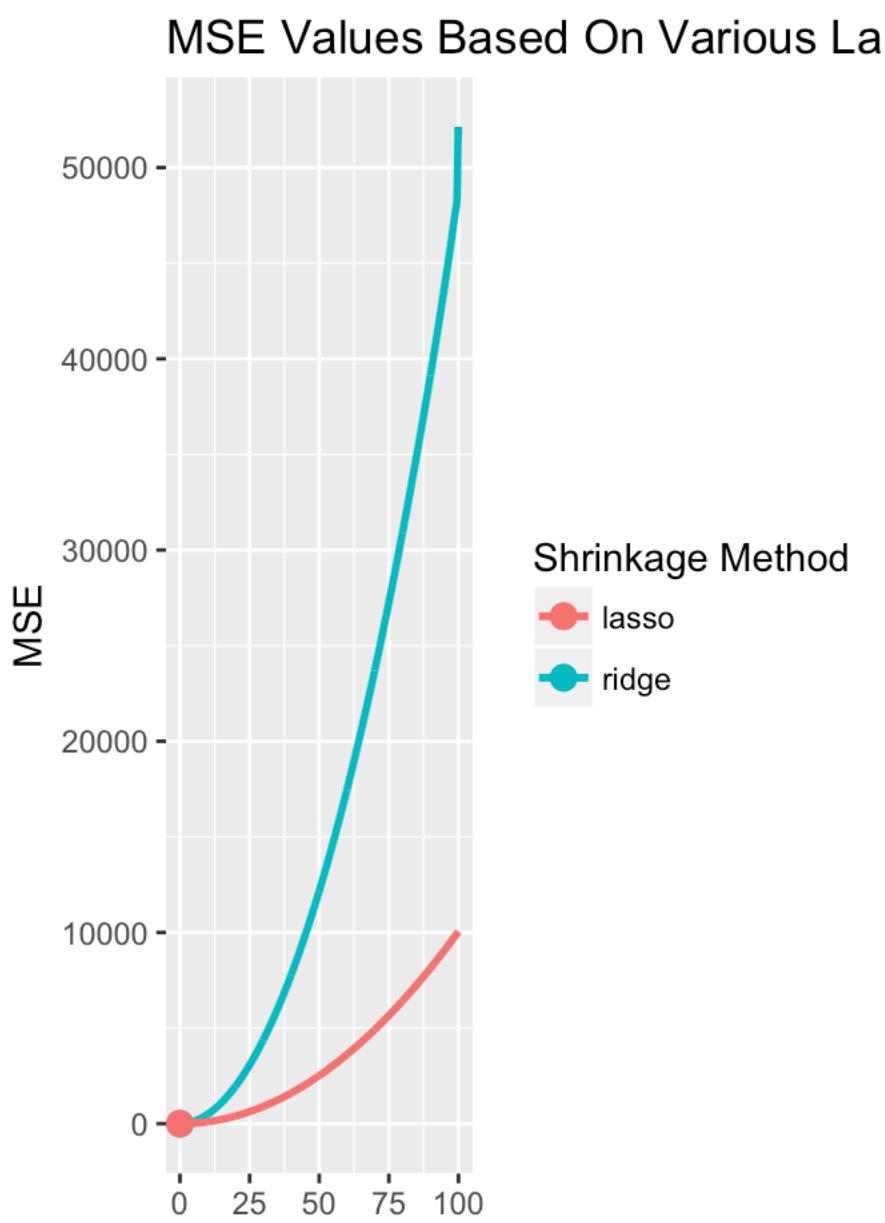
```
train <- read.csv("final.csv", header = TRUE)

regressience(X = train, y = train$SalePrice)
```

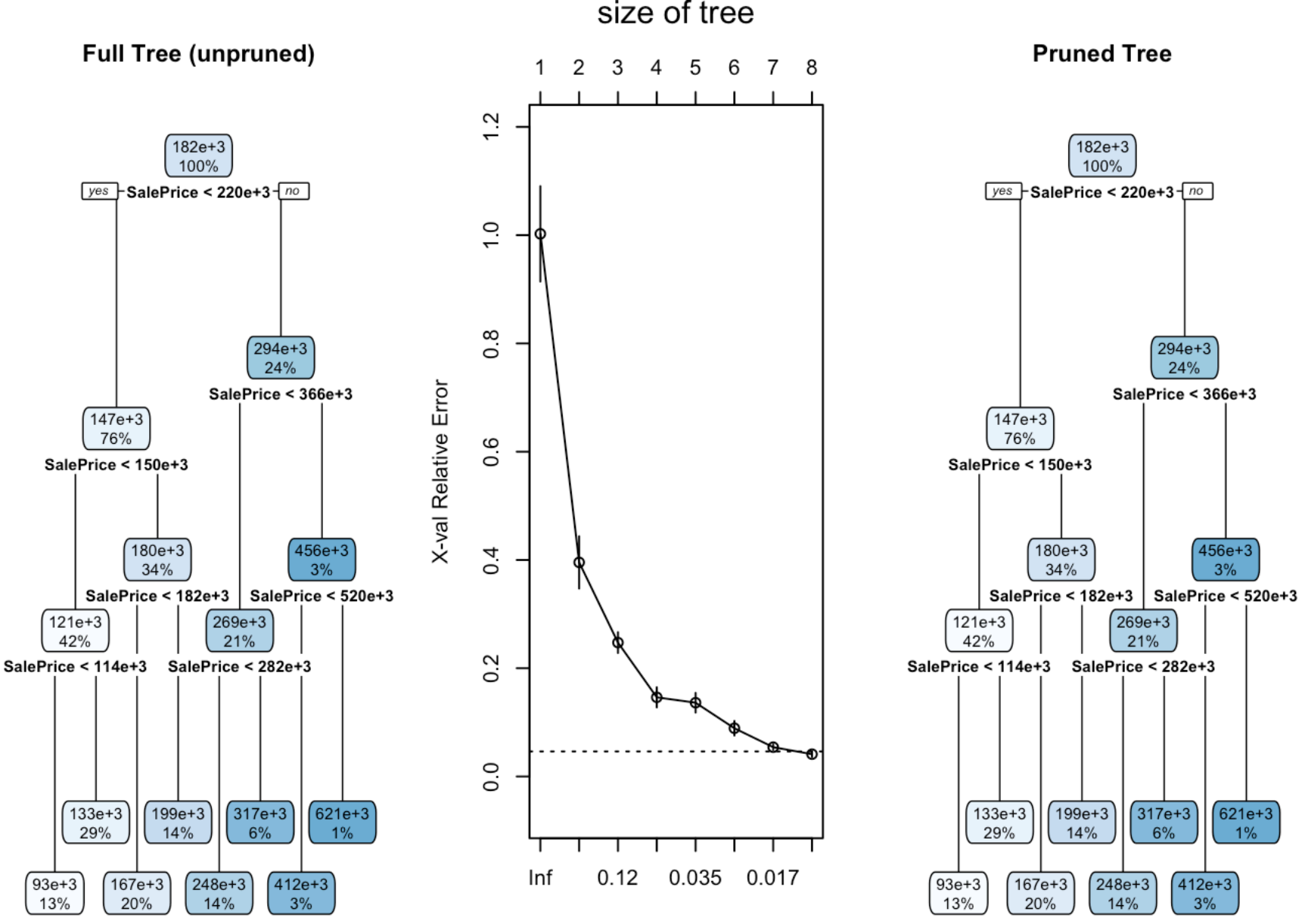
```
## Linear Model Formula: y~ LotArea + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + TotalBsmt
SF + X1stFlrSF + X2ndFlrSF + GrLivArea + BsmtFullBath + BsmtHalfBath + FullBath + Hal
fBath + BedroomAbvGr + KitchenAbvGr + TotRmsAbvGrd + Fireplaces + GarageCars + Garage
Area + WoodDeckSF + OpenPorchSF + SalePrice
## Lowest MSE: 2.79520872593919e-22
## Number of Folds: 25
##
##
##
## ~~~~~~~~Running Shrinkage Methods~~~~~~~~
##
##
## ~~~~~Ridge Regression~~~~~
## lambda grid: seq(0, 100, 0.1)
## Using 25-fold cv
## Model w/ lowest MSE
## Lambda:  0 ; MSE:  0.1606324
## Coefficients:
## 23 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)   0.3194497352
## (Intercept)   .
## LotArea      -0.0078569290
## BsmtFinSF1   -2.1192316389
## BsmtFinSF2   -0.5534351776
## BsmtUnfSF    -0.8766569592
## TotalBsmtSF   2.2800568015
## X1stFlrSF     0.1753183618
## X2ndFlrSF     0.0754725574
## GrLivArea    -0.1878698764
## BsmtFullBath  0.0035749399
## BsmtHalfBath  0.0013696302
## FullBath      0.0037366520
## HalfBath      0.0004156033
## BedroomAbvGr -0.0094530966
## KitchenAbvGr -0.0149135113
## TotRmsAbvGrd  0.0133662796
## Fireplaces    0.0015379652
```

```
## GarageCars    0.0033857008
## GarageArea    0.0050261205
## WoodDeckSF    0.0028626775
## OpenPorchSF   0.0030831260
## SalePrice     0.9999956857
##
##
##
## ~~~~~The Lasso~~~~~
## lambda grid: seq(0, 100, 0.1)
## Using 25-fold cv
## Model w/ lowest MSE
## Lambda:  0 ; MSE:  0.001846523
## Coefficients:
## 23 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)   9.306319e-03
## (Intercept)   .
## LotArea      -3.177275e-04
## BsmtFinSF1    3.190175e-03
## BsmtFinSF2    7.426796e-04
## BsmtUnfSF     1.071658e-03
## TotalBsmtSF   1.729892e-03
## X1stFlrSF     1.699493e-03
## X2ndFlrSF     1.451068e-03
## GrLivArea     4.282142e-04
## BsmtFullBath  2.027675e-04
## BsmtHalfBath  6.648602e-05
## FullBath      6.860265e-04
## HalfBath      1.253376e-04
## BedroomAbvGr -1.451817e-03
## KitchenAbvGr -1.518175e-03
## TotRmsAbvGrd  5.006605e-04
## Fireplaces    1.258397e-04
## GarageCars    8.148663e-04
## GarageArea   -1.257936e-04
## WoodDeckSF    1.633836e-04
## OpenPorchSF  -1.936286e-04
## SalePrice     9.999992e-01
```



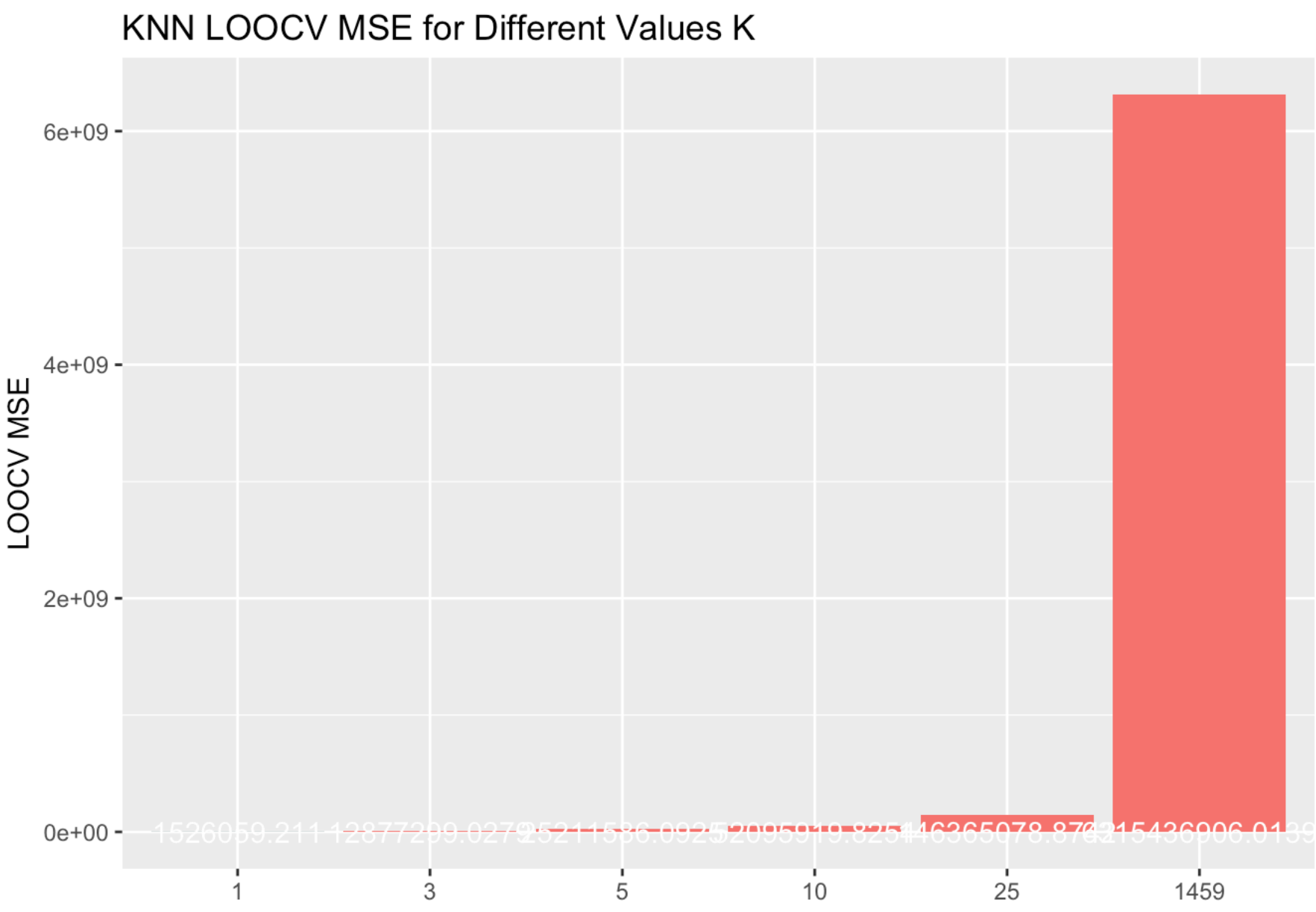




```
##
##
## ~~~~~~~~Running Regression Trees~~~~~~~~
##
## MSE of regression tree modeling using K-Fold Cross Validation: 267196597.02479
```



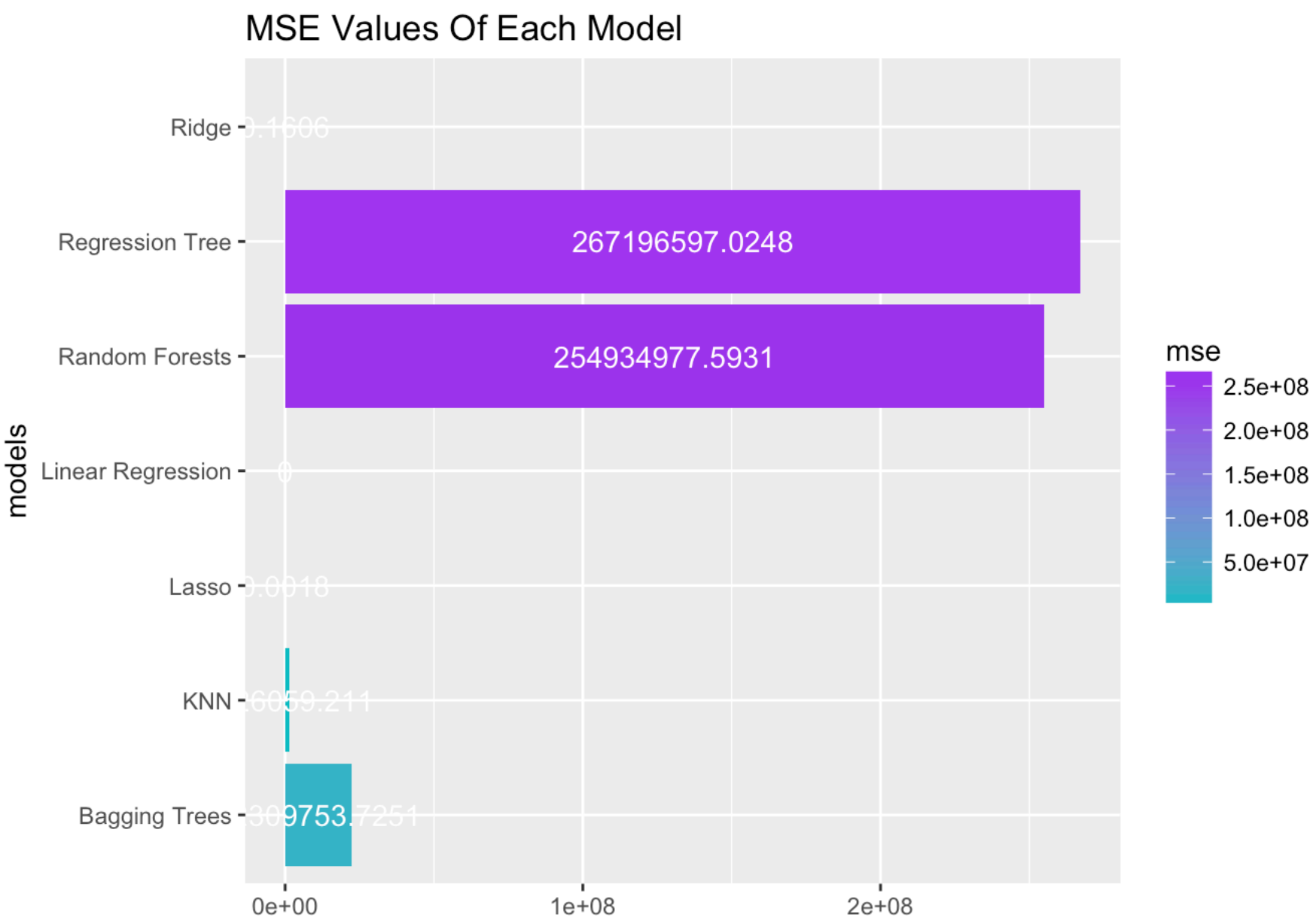

```
##
##
## ~~~~~Information For Best Cross Validated Model~~~~~
##
## Regression tree:
## rpart(formula = y ~ ., data = X, subset = train)
##
## Variables actually used in tree construction:
## [1] SalePrice
##
## Root node error: 7.674e+12/1174 = 6536632831
##
## n= 1174
##
##         CP nsplit rel error   xerror      xstd
## 1 0.606683      0  1.000000 1.002419 0.0881263
## 2 0.149583      1  0.393317 0.395520 0.0483490
## 3 0.101330      2  0.243734 0.247348 0.0192782
## 4 0.035943      3  0.142404 0.146232 0.0186536
## 5 0.033172      4  0.106461 0.136272 0.0182314
## 6 0.022474      5  0.073289 0.089125 0.0134845
## 7 0.012672      6  0.050814 0.054153 0.0050954
## 8 0.010000      7  0.038143 0.041224 0.0050373
```

```
##
## In order to prune, we choose the tree size that minimizes cv error(xerror) from th
e
## above table.
## We then grab the respective CP value and use that as the parameter for pruning
## In this case our CP value is:  0.01
##
## ~~~~~After Pruning Tree~~~~~
## n= 1174
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1174 7.674007e+12 182491.60
##    2) SalePrice< 220500 892 1.165768e+12 147083.70
##      4) SalePrice< 150250 494 2.508618e+11 120582.00
##        8) SalePrice< 113750 158 4.475575e+10  93334.13 *
##        9) SalePrice>=113750 336 3.363813e+10 133394.90 *
##      5) SalePrice>=150250 398 1.373027e+11 179977.90
##       10) SalePrice< 182450 235 2.144790e+10 166959.80 *
##       11) SalePrice>=182450 163 1.861254e+10 198746.20 *
##    3) SalePrice>=220500 282 1.852549e+12 294491.10
##      6) SalePrice< 365909.5 244 3.383277e+11 269312.90
##       12) SalePrice< 282338 168 5.391866e+10 247588.00 *
##       13) SalePrice>=282338 76 2.984336e+10 317336.30 *
##      7) SalePrice>=365909.5 38 3.663203e+11 456161.50
##       14) SalePrice< 519918.5 30 4.021195e+10 412165.60 *
##       15) SalePrice>=519918.5 8 5.027902e+10 621146.40 *
##
##
## ~~~~~~~~Running Bagging Trees~~~~~~~~
##
## CV MSE of Bagging Model: 22309753.7250603
## Using  21  Predictors to Construct Each Tree.
##
## ~~~~~~~~Running Random Forest~~~~~~~~
##
## CV MSE of Random Forest Model: 254934977.593124
## Using  5  Predictors to Construct Each Tree.
##
##
## ~~~~~~~~Running K Nearest Neighbors~~~~~~~~
##
## Testing the following values for k:
##     1   3   5   10   25   1459
##
## LOOCV MSE of best model evaluated by 1 Nearest Neighbors: 1526059
```




Nearest Neighbors

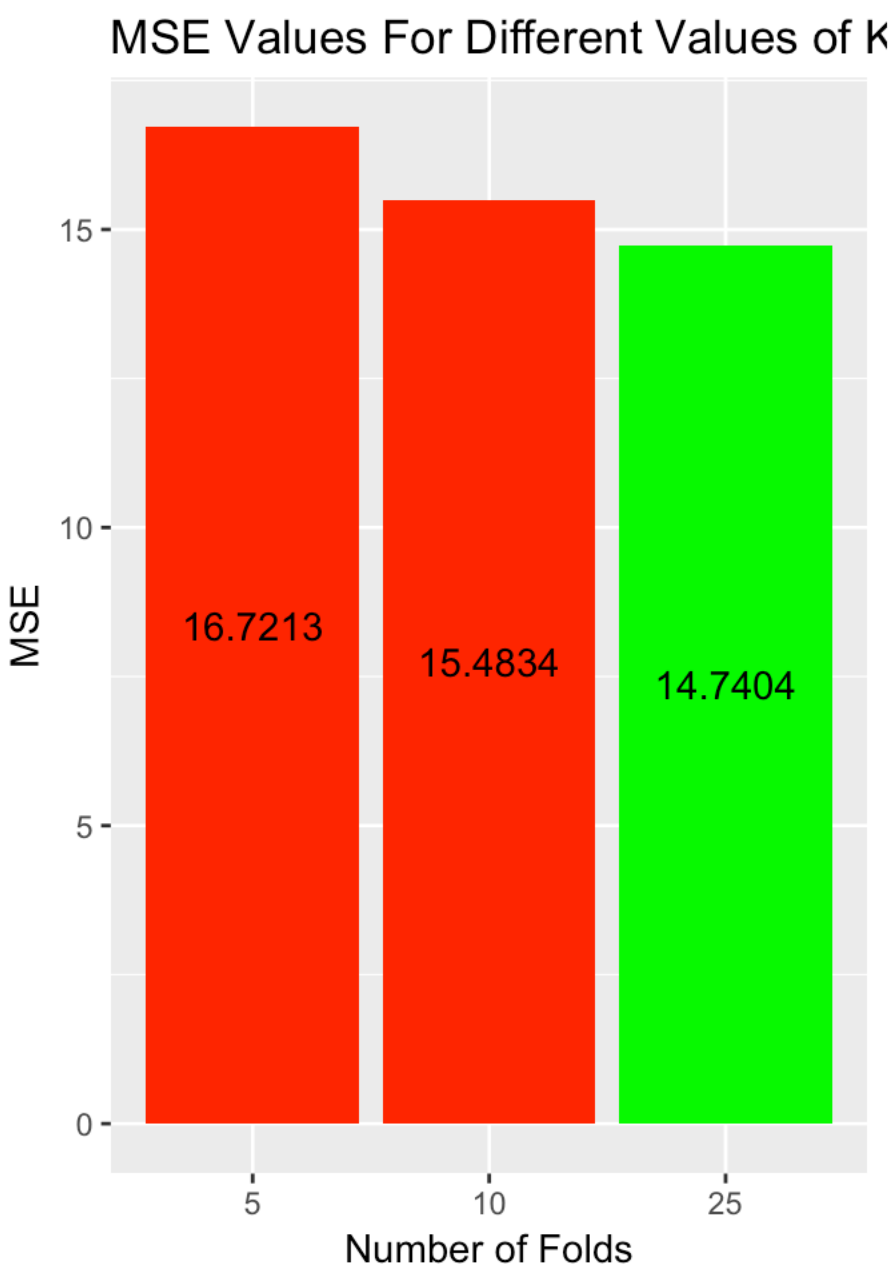
```
##
##
## ~~~~~~~~~~~~~Each Different Model's CV MSE~~~~~~~~~~~~~
##
## Full Linear Regression MSE: 2.79520872593919e-22
## Shrinkage Methods
##      Ridge Regression MSE: 0.160632421899883
##      The Lasso MSE: 0.0018465231171889
## Regression Tree Methods:
##      Single Regression Tree MSE: 267196597.02479
##      Bagging Trees MSE: 22309753.7250603
##      Random Forests MSE: 254934977.593124
## K Nearest Neighbors Model MSE: 1526059.2109589
```

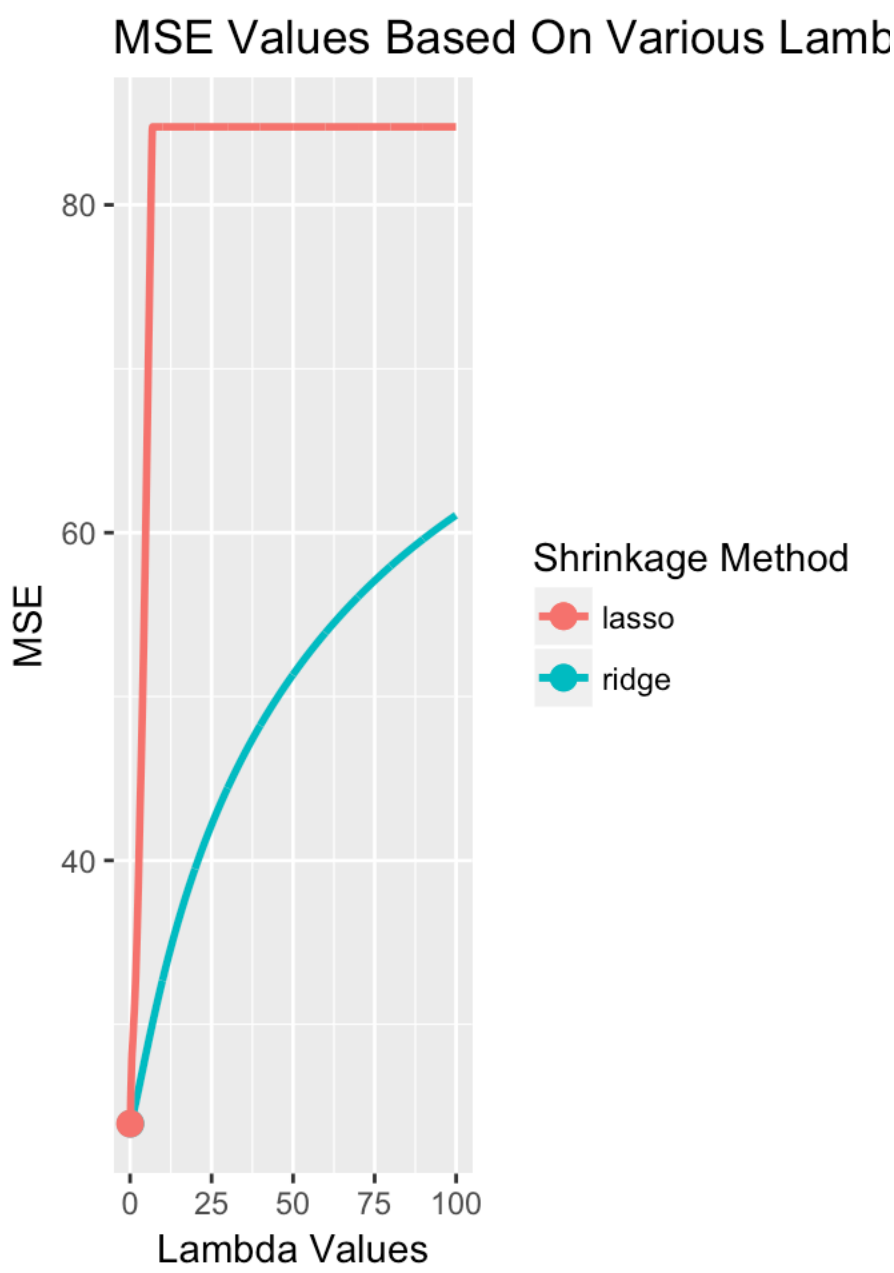



```
regressience(X = Boston[, -14], y = Boston$medv, sub.metric = "bic")
```

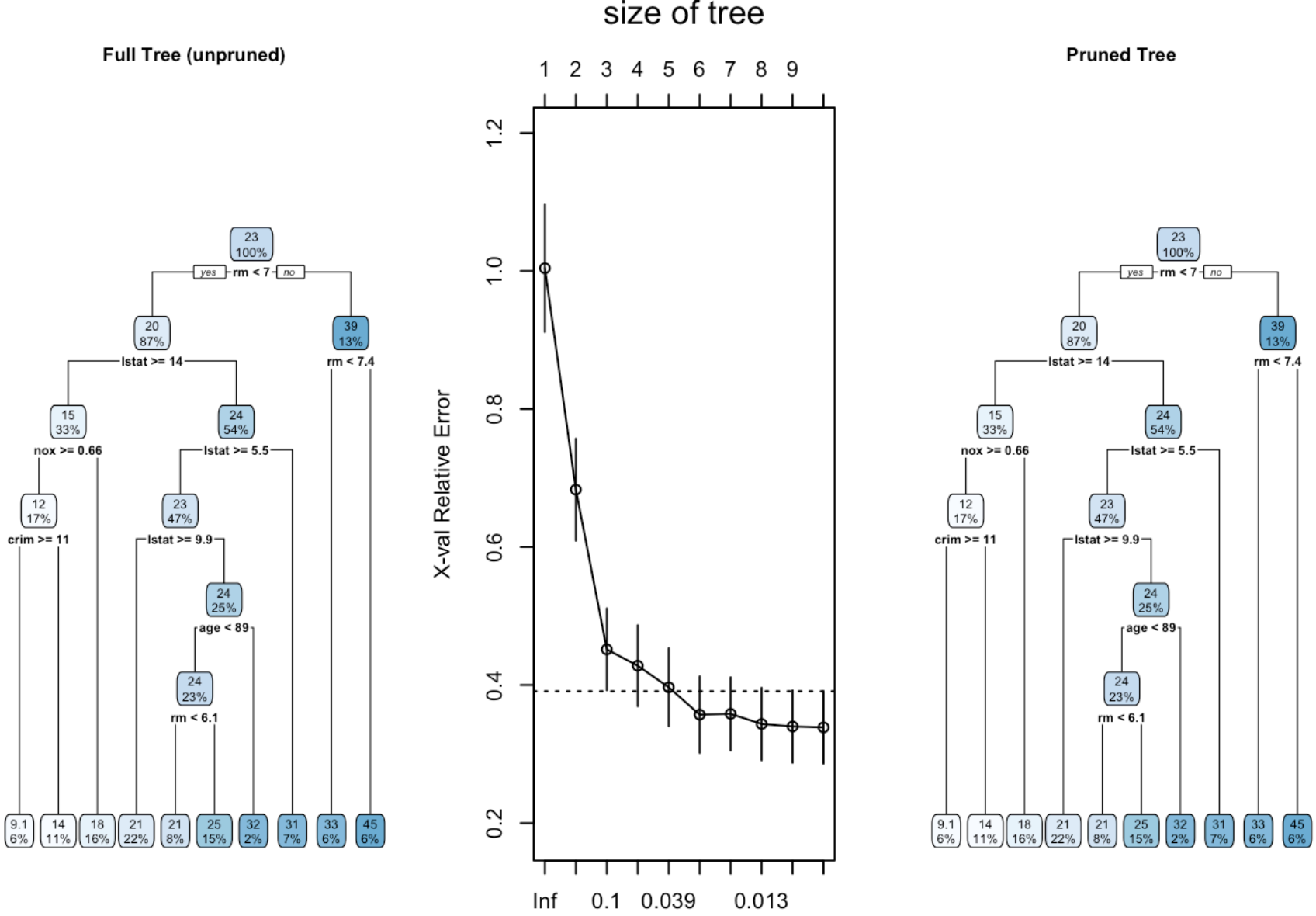
```
## Linear Model Formula: y~ crim + zn + indus + chas + nox + rm + age + dis + rad + t
ax + ptratio + black + lstat
## Lowest MSE: 14.7403624408981
## Number of Folds: 25
##
##
##
## ~~~~~~~~Running Shrinkage Methods~~~~~~~~
##
##
## ~~~~~Ridge Regression~~~~~
## lambda grid: seq(0, 100, 0.1)
## Using 25-fold cv
## Model w/ lowest MSE
## Lambda:  0.1 ; MSE:  23.93883
## Coefficients:
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)  3.453420e+01
## (Intercept)  .
```

```
## crim        -1.030248e-01
## zn           4.316131e-02
## indus        3.361661e-03
## chas         2.752657e+00
## nox         -1.652202e+01
## rm           3.870096e+00
## age         -4.313965e-04
## dis         -1.409138e+00
## rad          2.652143e-01
## tax         -1.039316e-02
## ptratio     -9.324739e-01
## black        9.279638e-03
## lstat       -5.151186e-01
##
##
##
## ~~~~~The Lasso~~~~~
## lambda grid: seq(0, 100, 0.1)
## Using 25-fold cv
## Model w/ lowest MSE
## Lambda:  0 ; MSE:  23.9395
## Coefficients:
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  3.647100e+01
## (Intercept)  .
## crim        -1.081417e-01
## zn           4.647112e-02
## indus        2.147245e-02
## chas         2.683814e+00
## nox         -1.776553e+01
## rm           3.810383e+00
## age          6.487048e-04
## dis         -1.475123e+00
## rad          3.075163e-01
## tax         -1.241994e-02
## ptratio     -9.531886e-01
## black        9.321309e-03
## lstat       -5.246481e-01
```



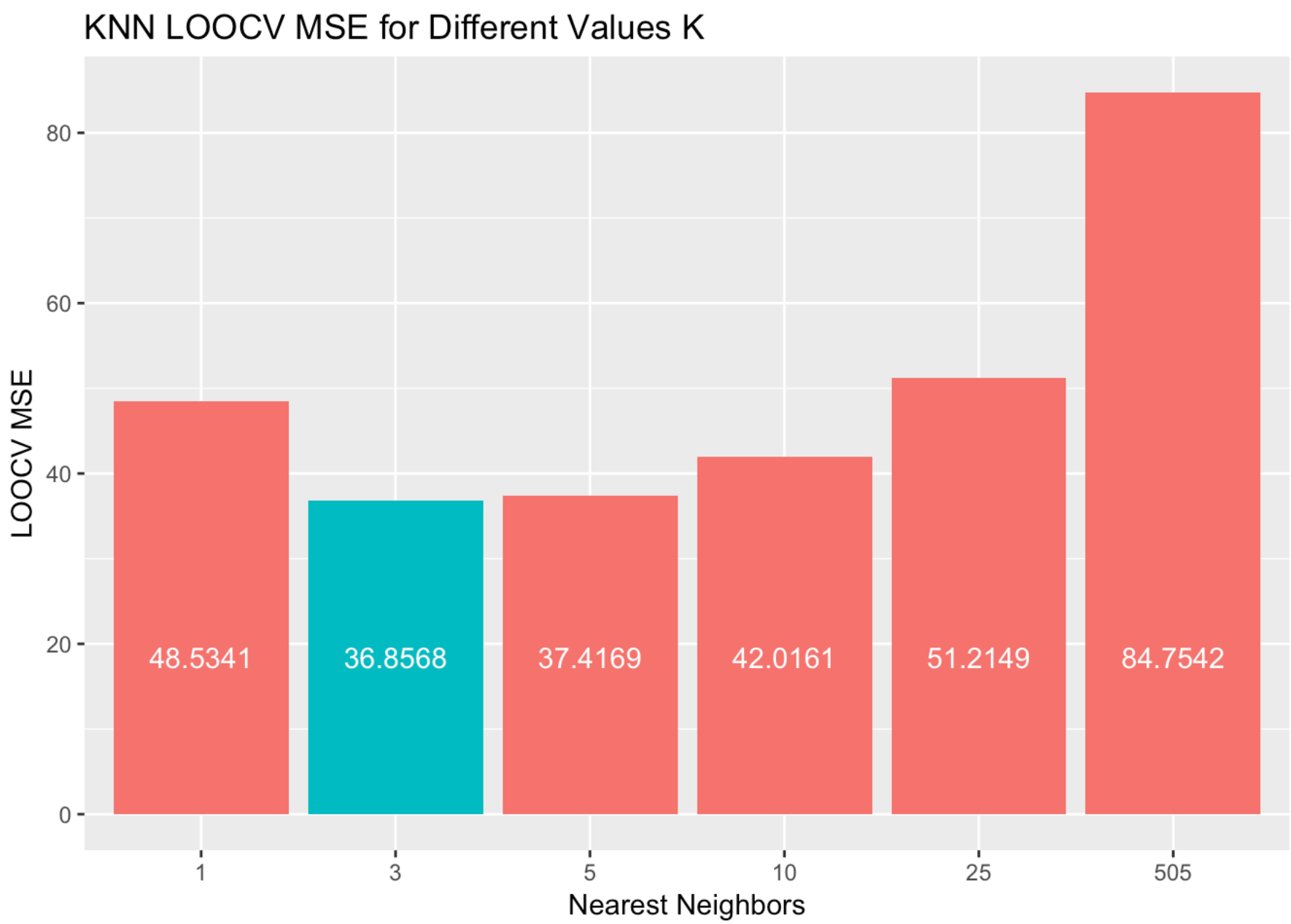

```
##
##
## ~~~~~~~~Running Regression Trees~~~~~~~~
##
## MSE of regression tree modeling using K-Fold Cross Validation: 24.4022769410863
```



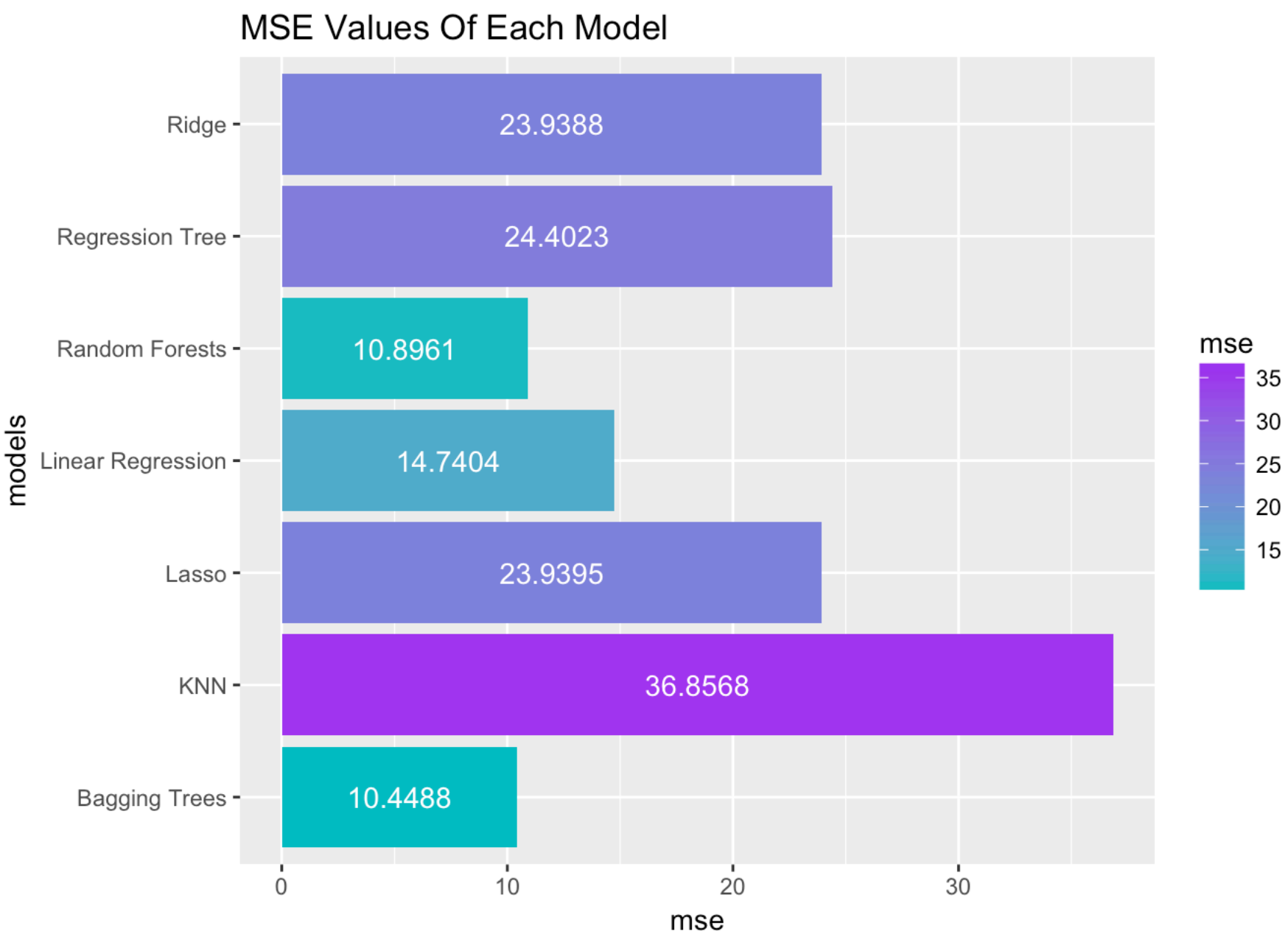

```
##
##
## ~~~~~Information For Best Cross Validated Model~~~~~
##
## Regression tree:
## rpart(formula = y ~ ., data = X, subset = train)
##
## Variables actually used in tree construction:
## [1] age   crim  lstat nox   rm
##
## Root node error: 35260/407 = 86.634
##
## n= 407
##
##          CP nsplit rel error  xerror     xstd
## 1  0.449443      0   1.00000 1.00392 0.092375
## 2  0.186143      1   0.55056 0.68309 0.073909
## 3  0.054289      2   0.36441 0.45164 0.059381
## 4  0.050093      3   0.31012 0.42793 0.058720
## 5  0.030019      4   0.26003 0.39669 0.056546
## 6  0.016820      5   0.23001 0.35707 0.055420
## 7  0.016346      6   0.21319 0.35820 0.052935
## 8  0.010578      7   0.19685 0.34347 0.052456
```

```
## 9  0.010241      8   0.18627 0.33989 0.052469
## 10 0.010000      9   0.17603 0.33858 0.052474
##
## In order to prune, we choose the tree size that minimizes cv error(xerror) from th
e
## above table.
## We then grab the respective CP value and use that as the parameter for pruning
## In this case our CP value is:  0.01
##
## ~~~~~After Pruning Tree~~~~~
## n= 407
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 407 35259.9000 22.717200
##    2) rm< 6.978 355 15644.2300 20.329010
##      4) lstat>=14.4 135  2631.4040 14.840000
##        8) nox>=0.657 69   772.5099 12.101450
##         16) crim>=11.36915 26   132.6865  9.111538 *
##         17) crim< 11.36915 43   266.8563 13.909300 *
##        9) nox< 0.657 66   800.4194 17.703030 *
##      5) lstat< 14.4 220  6449.4380 23.697270
##       10) lstat>=5.51 191  3574.3610 22.593190
##         20) lstat>=9.95 89   522.2360 20.706740 *
##         21) lstat< 9.95 102  2459.0430 24.239220
##           42) age< 89.45 94   982.4333 23.545740
##             84) rm< 6.1445 31   199.4374 20.751610 *
##             85) rm>=6.1445 63   421.8832 24.920630 *
##           43) age>=89.45 8   900.2488 32.387500 *
##       11) lstat< 5.51 29  1108.8020 30.968970 *
##    3) rm>=6.978 52  3768.3670 39.021150
##      6) rm< 7.435 26   901.4046 32.953850 *
##      7) rm>=7.435 26   952.7265 45.088460 *
##
##
## ~~~~~~~~Running Bagging Trees~~~~~~~~
##
## CV MSE of Bagging Model: 10.4488411802273
## Using  13  Predictors to Construct Each Tree.
##
## ~~~~~~~~Running Random Forest~~~~~~~~
##
## CV MSE of Random Forest Model: 10.8961261685777
## Using  4  Predictors to Construct Each Tree.
##
##
## ~~~~~~~~Running K Nearest Neighbors~~~~~~~~
##
## Testing the following values for k:
```

```
##      1    3    5    10    25    505
##
## LOOCV MSE of best model evaluated by 3 Nearest Neighbors: 36.85682
```




```
##
##
## ~~~~~~~~~~~~~~Each Different Model's CV MSE~~~~~~~~~~~~~~
##
## Full Linear Regression MSE: 14.7403624408981
## Shrinkage Methods
##      Ridge Regression MSE: 23.9388313800399
##      The Lasso MSE: 23.9394981805182
## Regression Tree Methods:
##      Single Regression Tree MSE: 24.4022769410863
##      Bagging Trees MSE: 10.4488411802273
##      Random Forests MSE: 10.8961261685777
## K Nearest Neighbors Model MSE: 36.8568181818182
```


MSE Values Of Each Model
Ridge
Regression Tree
Random Forests
Linear Regression
Lasso
KNN
Bagging Trees
models
23.9388
24.4023
10.8961
14.7404
23.9395
36.8568
10.4488
0
10
20
30
mse
35
30
25
20
15
mse