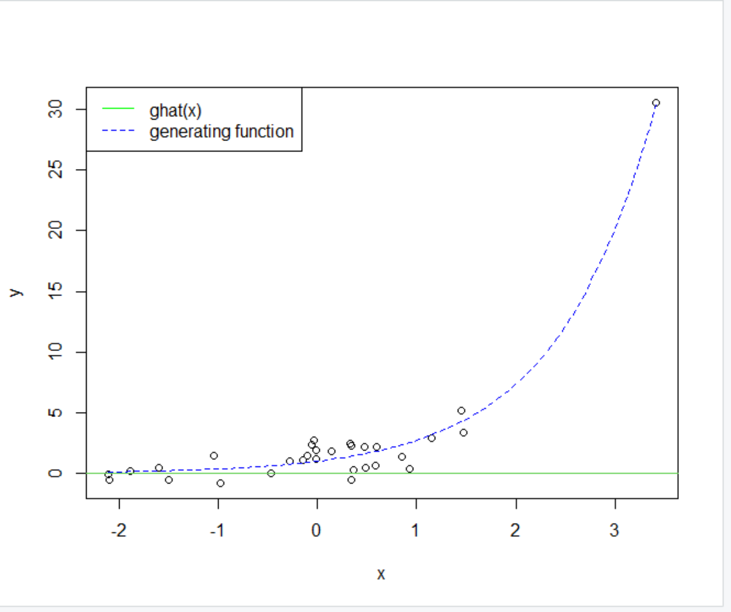
James Yang

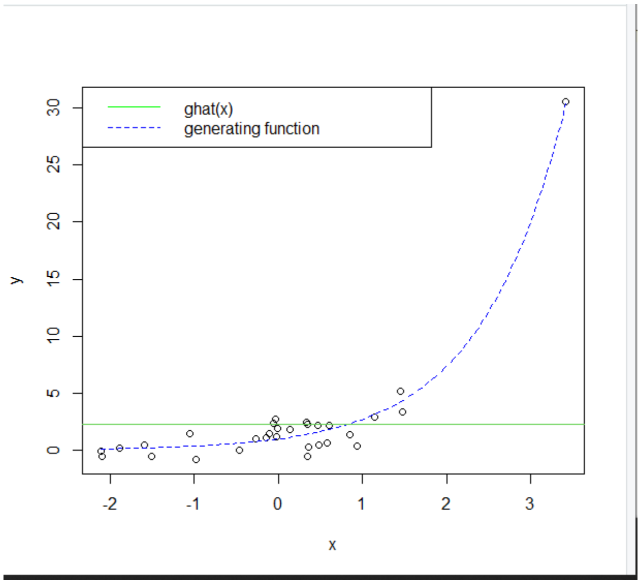
DATA 558

Hw4

1. Giving g(x) = e^x, we can give plot a graph of each.
   a. ĝ(x) = 0

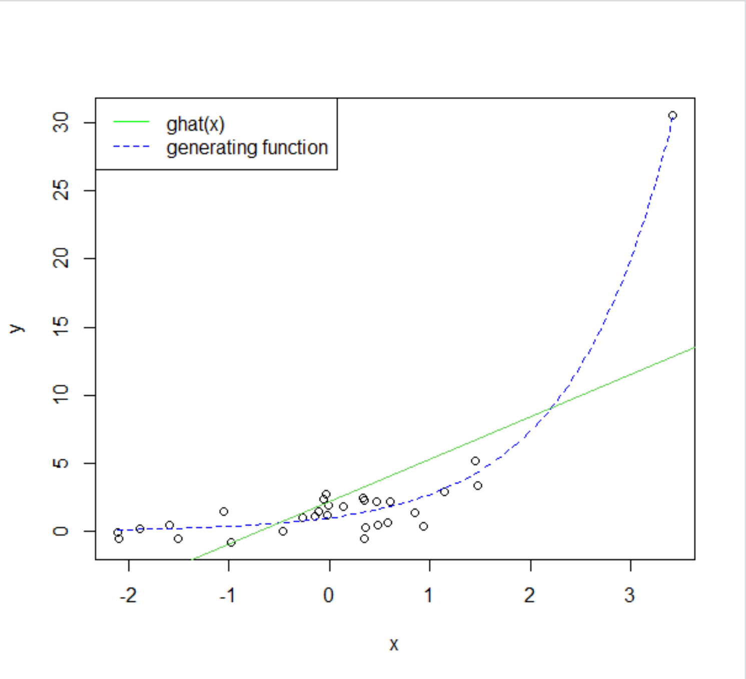


   b. ĝ(x) = $\frac{1}{n}\sum_{i=1}^{n} y_i$
   = 2.256



   c. ĝ(x) = ax+b
   After running lm(y ~ x), we get the coefficients to be:

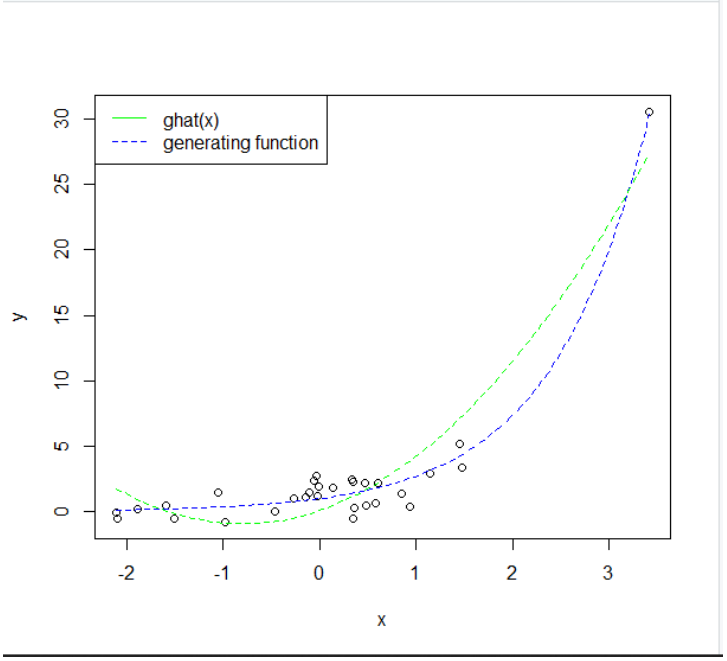
```
(Intercept)           x
   2.194379    3.105231
```



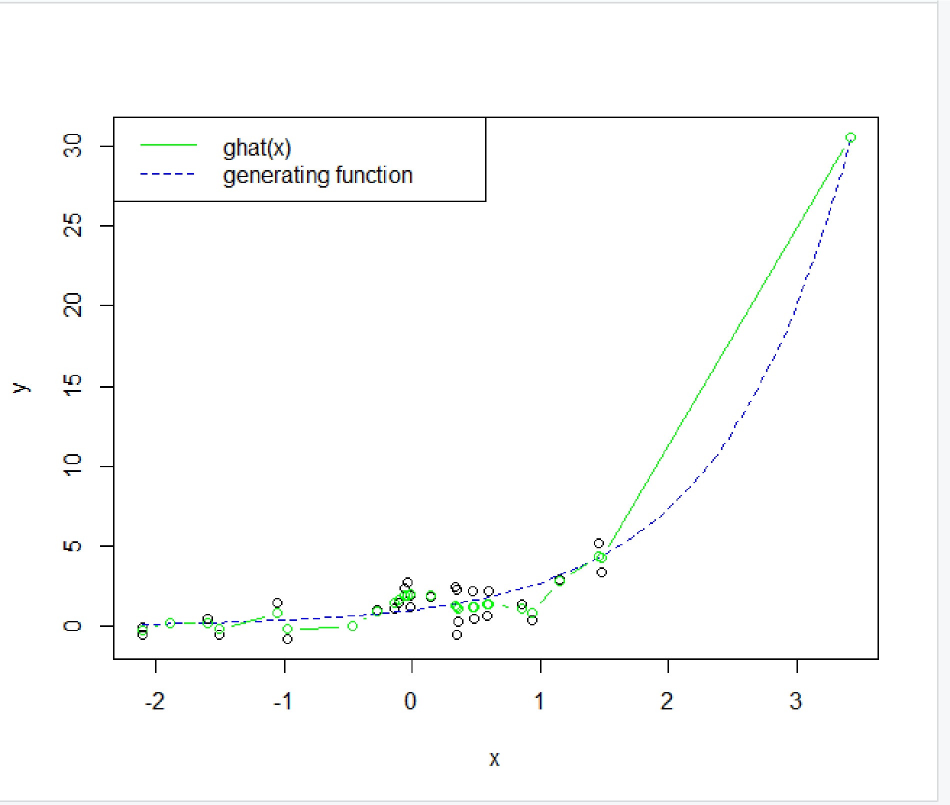
d. ĝ(x) = $ax^2 + bx + c$

After running a lm with y ~ x + x^2, the equation is:

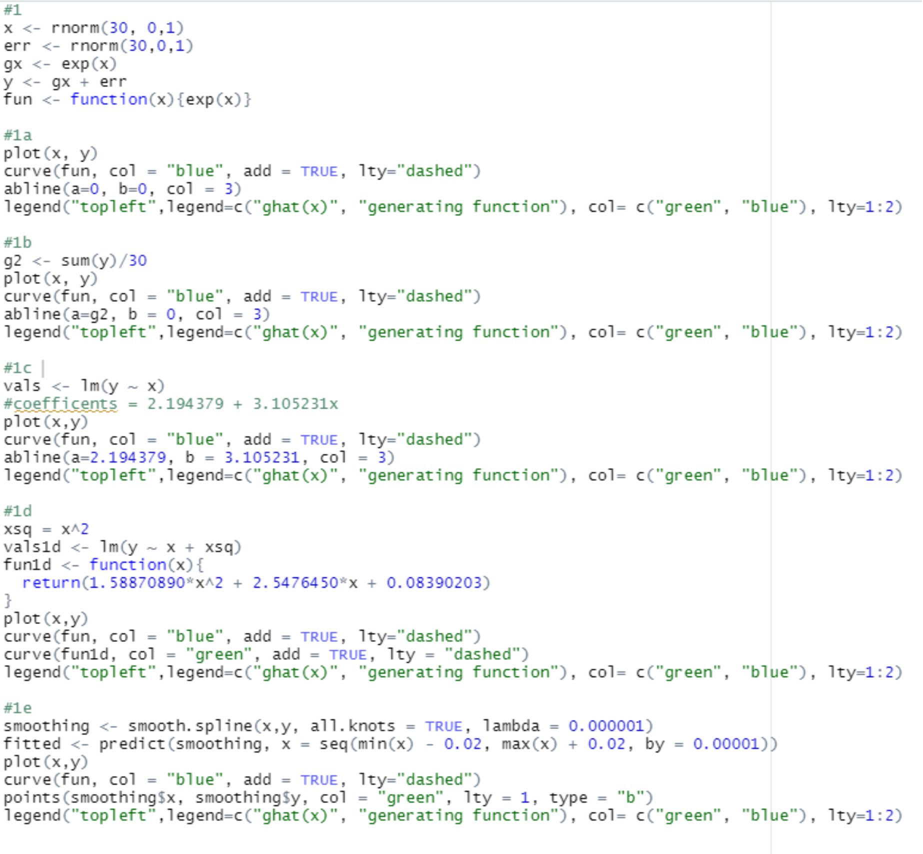
1.58870890*x^2 + 2.5476450*x + 0.08390203



e.



Using this code:

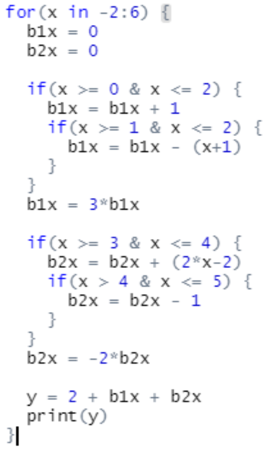


```
#1
x <- rnorm(30, 0,1)
err <- rnorm(30,0,1)
gx <- exp(x)
y <- gx + err
fun <- function(x){exp(x)}

#1a
plot(x, y)
curve(fun, col = "blue", add = TRUE, lty="dashed")
abline(a=0, b=0, col = 3)
legend("topleft",legend=c("ghat(x)", "generating function"), col= c("green", "blue"), lty=1:2)

#1b
g2 <- sum(y)/30
plot(x, y)
curve(fun, col = "blue", add = TRUE, lty="dashed")
abline(a=g2, b = 0, col = 3)
legend("topleft",legend=c("ghat(x)", "generating function"), col= c("green", "blue"), lty=1:2)

#1c
vals <- lm(y ~ x)
#coefficents = 2.194379 + 3.105231x
plot(x,y)
curve(fun, col = "blue", add = TRUE, lty="dashed")
abline(a=2.194379, b = 3.105231, col = 3)
legend("topleft",legend=c("ghat(x)", "generating function"), col= c("green", "blue"), lty=1:2)

#1d
xsq = x^2
vals1d <- lm(y ~ x + xsq)
fun1d <- function(x){
  return(1.58870890*x^2 + 2.5476450*x + 0.08390203)
}
plot(x,y)
curve(fun, col = "blue", add = TRUE, lty="dashed")
curve(fun1d, col = "green", add = TRUE, lty = "dashed")
legend("topleft",legend=c("ghat(x)", "generating function"), col= c("green", "blue"), lty=1:2)

#1e
smoothing <- smooth.spline(x,y, all.knots = TRUE, lambda = 0.000001)
fitted <- predict(smoothing, x = seq(min(x) - 0.02, max(x) + 0.02, by = 0.00001))
plot(x,y)
curve(fun, col = "blue", add = TRUE, lty="dashed")
points(smoothing$x, smoothing$y, col = "green", lty = 1, type = "b")
legend("topleft",legend=c("ghat(x)", "generating function"), col= c("green", "blue"), lty=1:2)
```

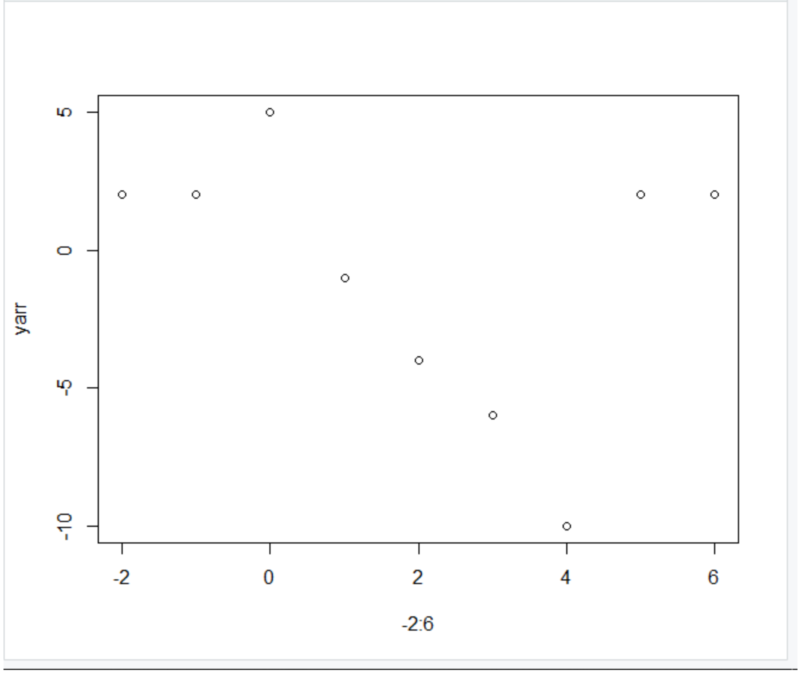
2. We will plot it out after calculating x = [-2,6]:

X	Y
-2	2
-1	2
0	5
1	-1
2	-4
3	-6
4	-10
5	2
6	2

Using this code:



```r
for(x in -2:6) {
  b1x = 0
  b2x = 0

  if(x >= 0 & x <= 2) {
    b1x = b1x + 1
    if(x >= 1 & x <= 2) {
      b1x = b1x - (x+1)
    }
  }
  b1x = 3*b1x

  if(x >= 3 & x <= 4) {
    b2x = b2x + (2*x-2)
    if(x > 4 & x <= 5) {
      b2x = b2x - 1
    }
  }
  b2x = -2*b2x

  y = 2 + b1x + b2x
  print(y)
}
```



It appears that the curve is constant from -2 and -1 where y = 2. From 0 to 1, y is linear with y = -6x + 5. From 1 to 2, it is linear with y = -3x + 2. From 2 to 3 it is linear with y =

-2x. From 3 to 4 it is linear with y = -4x + 6. From 4 to 5 it is linear with y = 12x – 58. From 5 to 6 it is constant with y = 2.

3.

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\psi)^3_+$$

So we know that f(x) is a piece wise polynomial

because $x > \psi$,

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \psi)^3$$

$$= (\beta_0 - \beta_4 \psi^3) + (\beta_1 + 3\psi^2 \beta_4)x + (\beta_2 - 3\beta_4 \psi)x^2 + (\beta_3 + \beta_4)x^3$$

If we split the left hand side and right hand side:

$$f_L(\psi) = \beta_0 + \beta_1 \psi + \beta_2 \psi^2 + \beta_3 \psi^3$$

$$f_R(\psi) = (\beta_0 - \beta_4 \psi^3) + (\beta_1 + 3\psi^2 \beta_4)\psi + (\beta_2 - 3\beta_4 \psi)\psi^2 + (\beta_3 + \beta_4)\psi^3$$

$$f'_L(\psi) = \underline{\beta_1 + 2\beta_2 \psi + 3\beta_3 \psi^2}$$

$$f'_R(\psi) = \beta_1 + 3\beta_4 \psi^2 + 2(\beta_2 - 3\beta_4 \psi) + 3(\beta_3 + \beta_4)\psi^2$$

$$= \underline{\beta_1 + 2\beta_2 \psi + 3\beta_3 \psi^2}$$

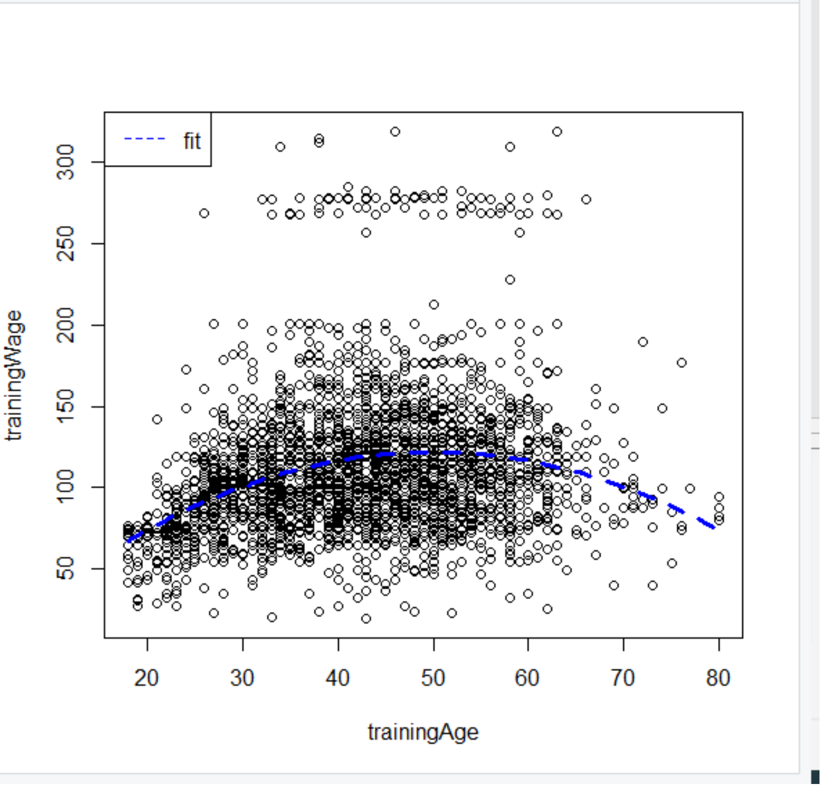
$$f''_L(\psi) = 2\beta_2 + 6\beta_3$$

$$f''_R(\psi) = 2\beta_2 + 6\beta_3$$

They're equivalent and $f''(x)$ is continous @ ψ which means f(x) is a cubic spline ✓

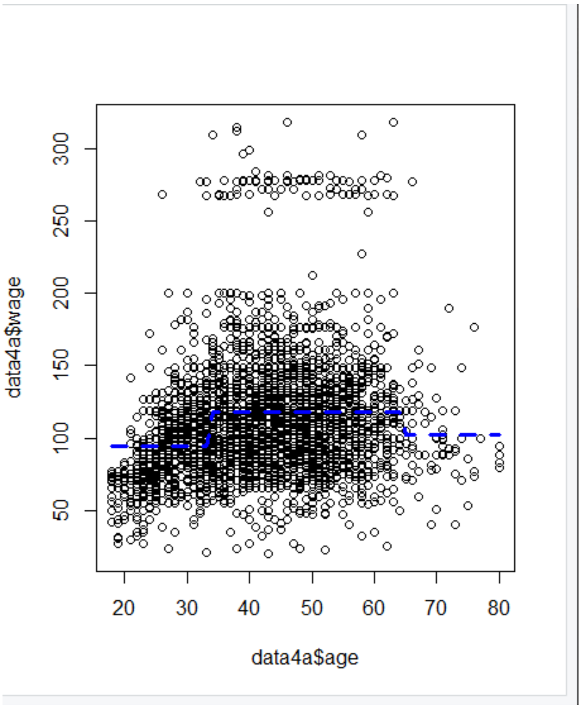
4. The MSEs will be labeled at the bottom of the problem.

a. After splitting data in 2800 training and 200 testing, polynomial function ended up being:
Y = -0.05321429*x^2 + 5.31585967*x -10.87610589

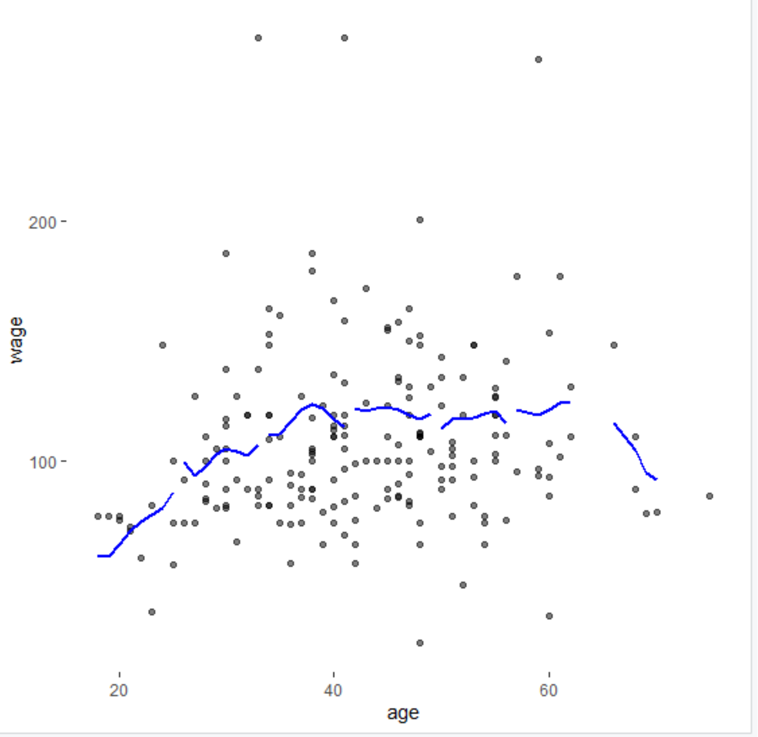


Producing a graph like this with the blue dashed line being the fitted values.

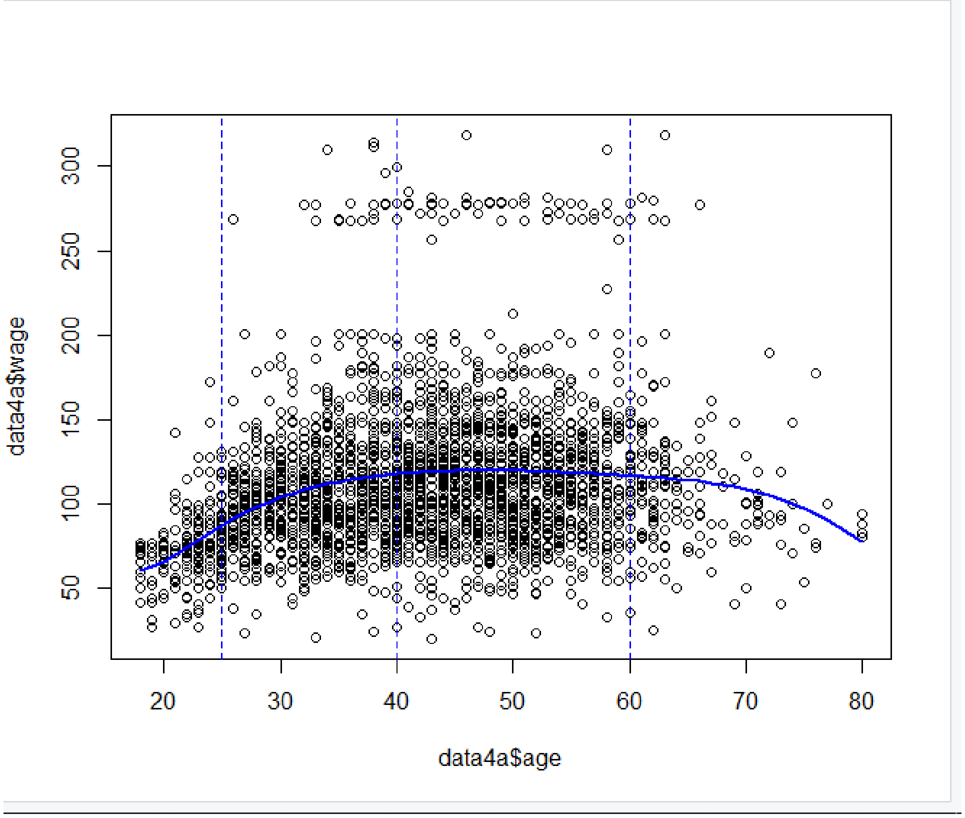
b.



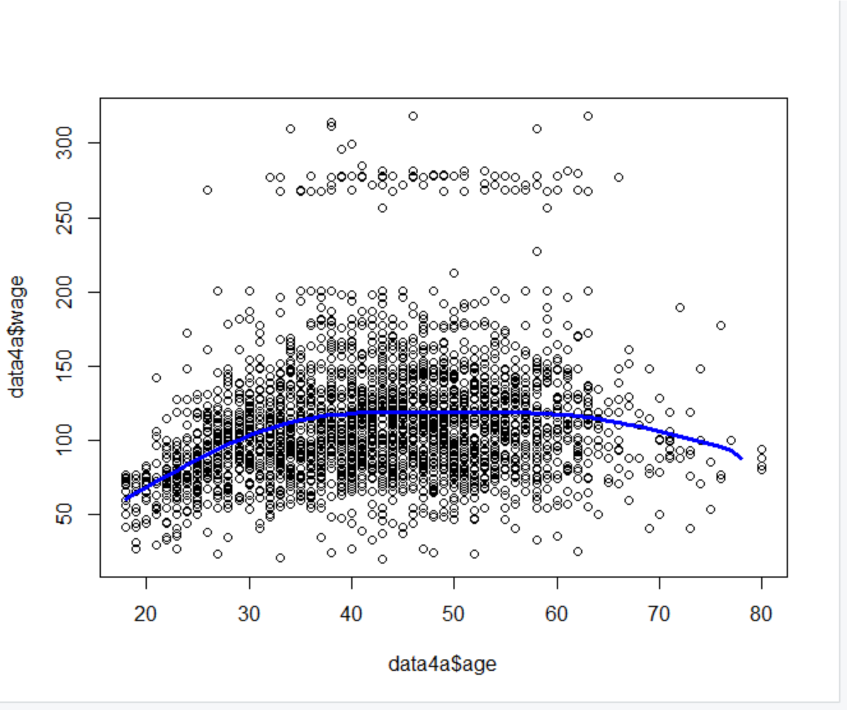
c.


wage
200
100
20
40
60
age

d.


data4a$wage
300
250
200
150
100
50
20
30
40
50
60
70
80
data4a$age

e.



After analyzing all of the errors:

MSE 4a: 1609.01

MSE 4b: 1731.71

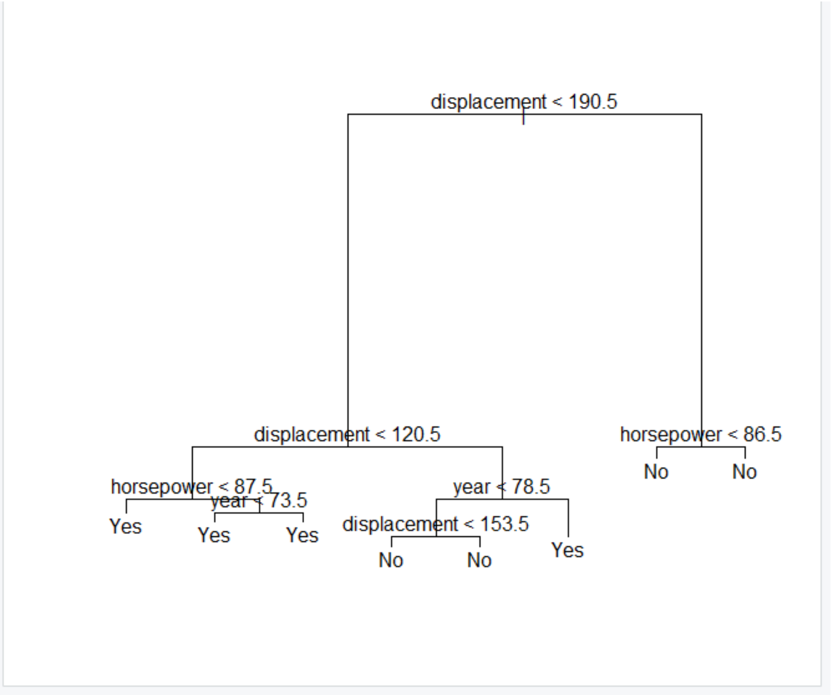
MSE 4c: 1532.32

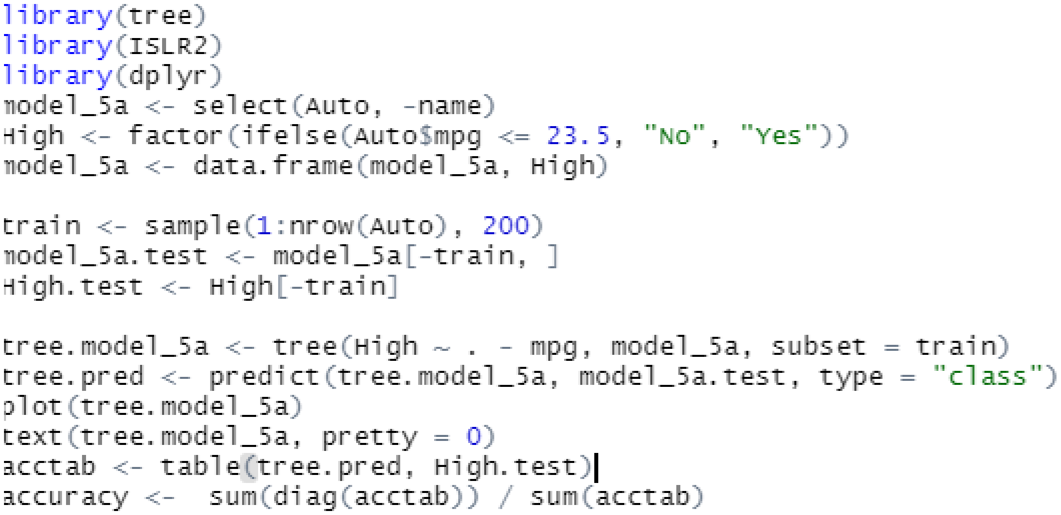
MSE 4d: 1590.30

The best approach for this problem according to MSE was the piecewise polynomial. This is interesting but it could be because the dataset is fitting the curvature of step function in a polynomial manner. However, after reevaluating the MSE values, they do fluctuate a bit but keep a consistent lowest value for 4c.

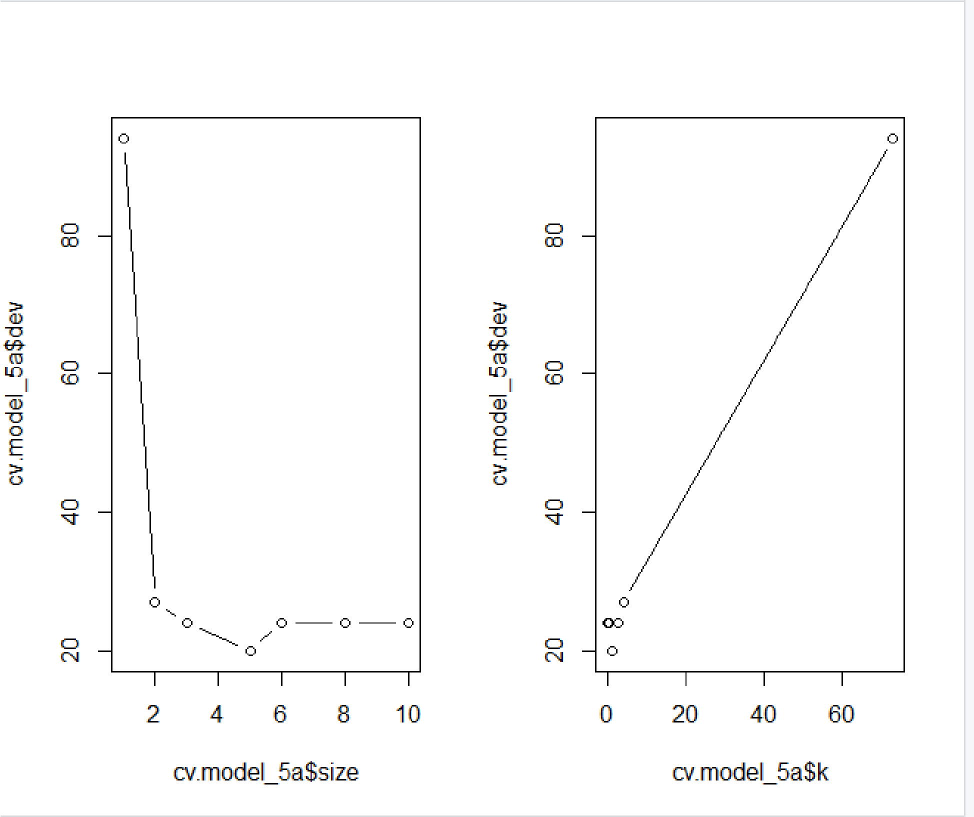
5.

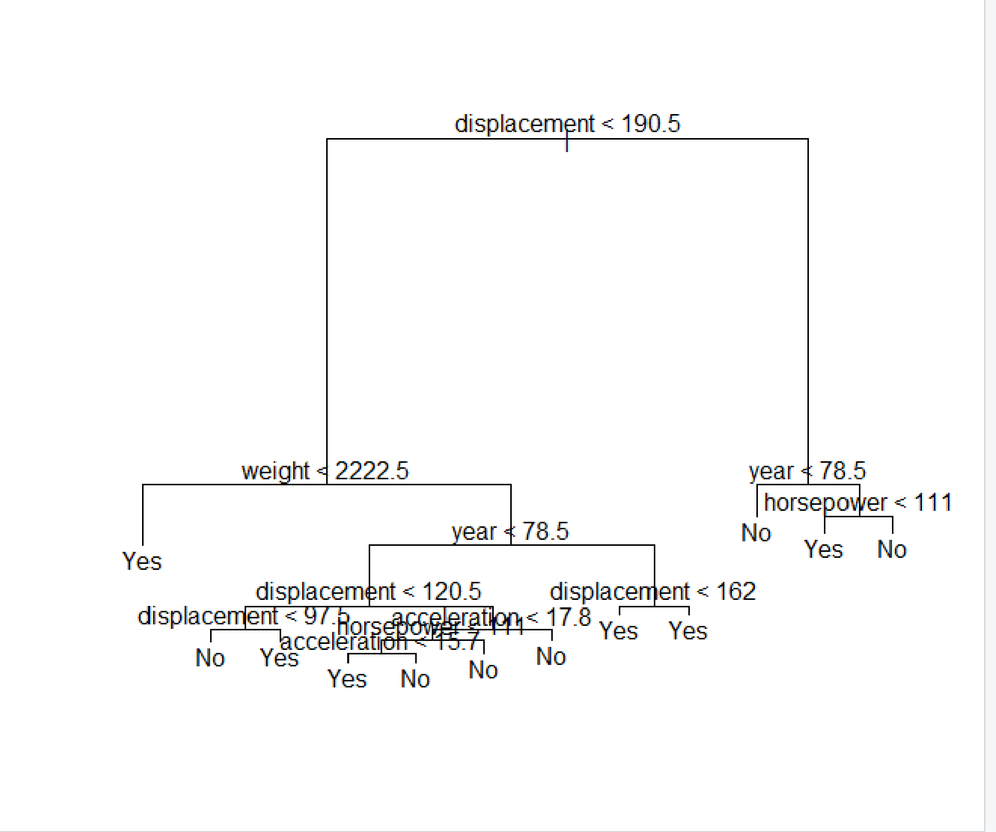
a. When fitting a regression tree without pruning, we get these results. **Its accuracy predicts 85.4%**



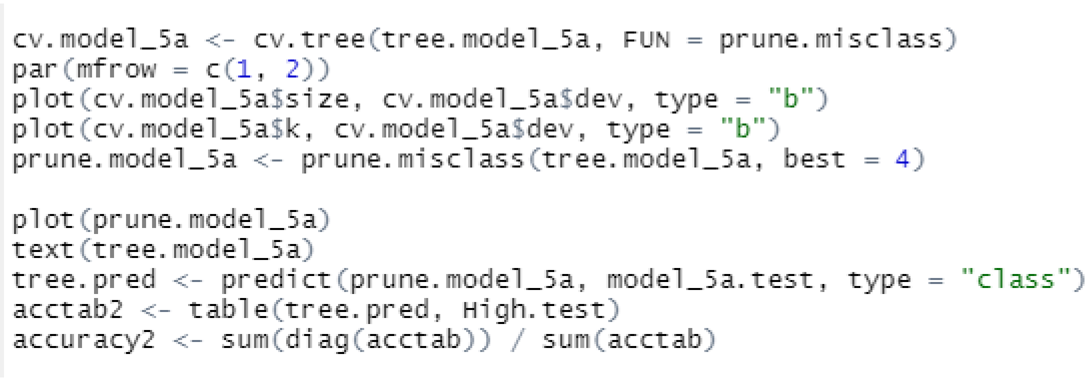


```
library(tree)
library(ISLR2)
library(dplyr)
model_5a <- select(Auto, -name)
High <- factor(ifelse(Auto$mpg <= 23.5, "No", "Yes"))
model_5a <- data.frame(model_5a, High)

train <- sample(1:nrow(Auto), 200)
model_5a.test <- model_5a[-train, ]
High.test <- High[-train]

tree.model_5a <- tree(High ~ . - mpg, model_5a, subset = train)
tree.pred <- predict(tree.model_5a, model_5a.test, type = "class")
plot(tree.model_5a)
text(tree.model_5a, pretty = 0)
acctab <- table(tree.pred, High.test)
accuracy <-  sum(diag(acctab)) / sum(acctab)
```

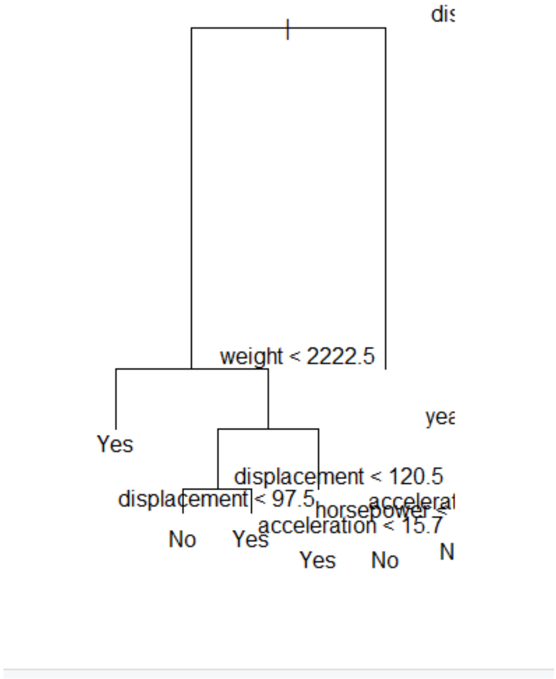




When pruning with a best of 4, we get a slightly higher accuracy of 89.58%

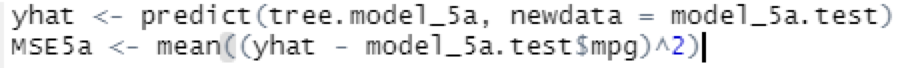


```
cv.model_5a <- cv.tree(tree.model_5a, FUN = prune.misclass)
par(mfrow = c(1, 2))
plot(cv.model_5a$size, cv.model_5a$dev, type = "b")
plot(cv.model_5a$k, cv.model_5a$dev, type = "b")
prune.model_5a <- prune.misclass(tree.model_5a, best = 4)

plot(prune.model_5a)
text(tree.model_5a)
tree.pred <- predict(prune.model_5a, model_5a.test, type = "class")
acctab2 <- table(tree.pred, High.test)
accuracy2 <- sum(diag(acctab)) / sum(acctab)
```



When pruning with a best of 4, it gives a higher accuracy rating than when we simply created a tree regression. This means the pruning process has produced a more interpretable tree, but also has slightly improved classification accuracy. This makes sense because as the value of best increases, we obtain a larger pruned tree that can give us lower classification accuracy but higher interpretability.

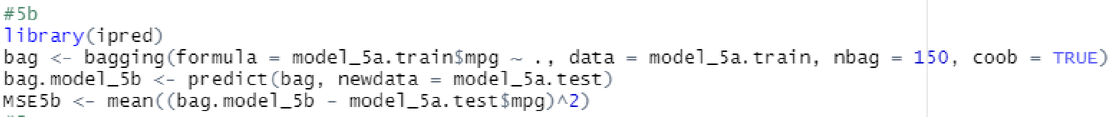
After calculating the MSE,



```
yhat <- predict(tree.model_5a, newdata = model_5a.test)
MSE5a <- mean((yhat - model_5a.test$mpg)^2)
```

We get that the MSE is about **20.31**, meaning the model does not fit the data very well with regression trees.

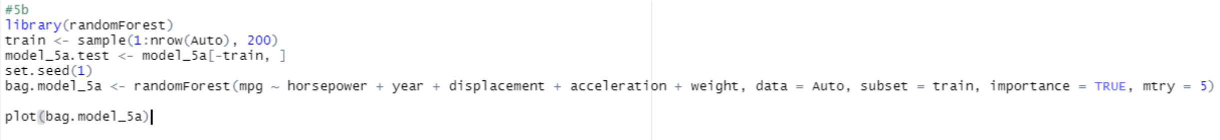
b. Using this code:



```
#5b
library(ipred)
bag <- bagging(formula = model_5a.train$mpg ~ ., data = model_5a.train, nbag = 150, coob = TRUE)
bag.model_5b <- predict(bag, newdata = model_5a.test)
MSE5b <- mean((bag.model_5b - model_5a.test$mpg)^2)
```

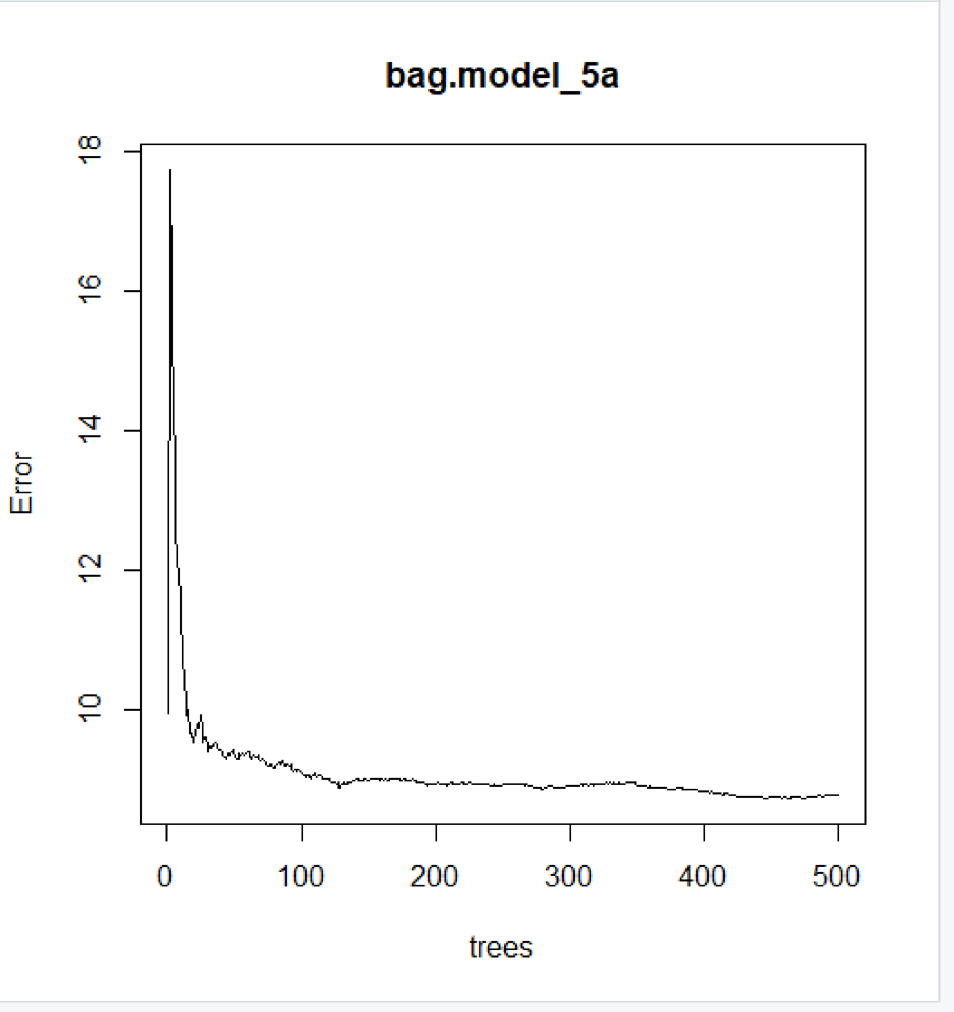
We get that the MSE is about **17.64** which means the bagging did improve the model accuracy a bit. The tuning parameters used were the number of bags = 150 and that the coob = TRUE.

c. When fitting a random forest model:



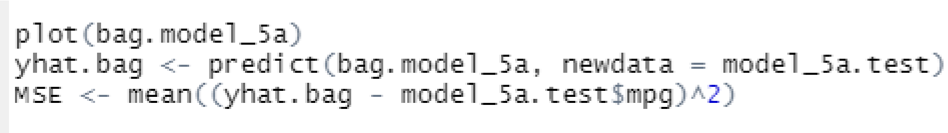
```
#5b
library(randomForest)
train <- sample(1:nrow(Auto), 200)
model_5a.test <- model_5a[-train, ]
set.seed(1)
bag.model_5a <- randomForest(mpg ~ horsepower + year + displacement + acceleration + weight, data = Auto, subset = train, importance = TRUE, mtry = 5)

plot(bag.model_5a)
```

With an mtry of 5:



Mtry = 5

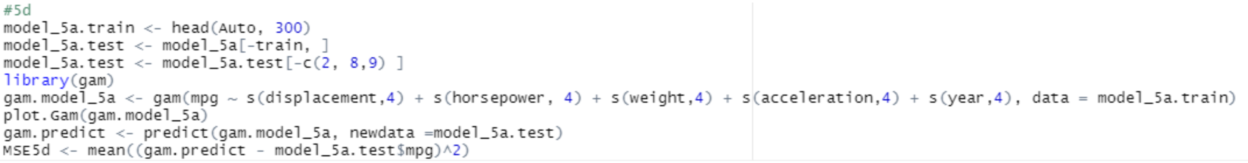
Obviously as the ntrees progress, we see that the error decreases. When the trees increase though, it makes the model less interpretable which kind of defeats the purpose of having a regression tree.



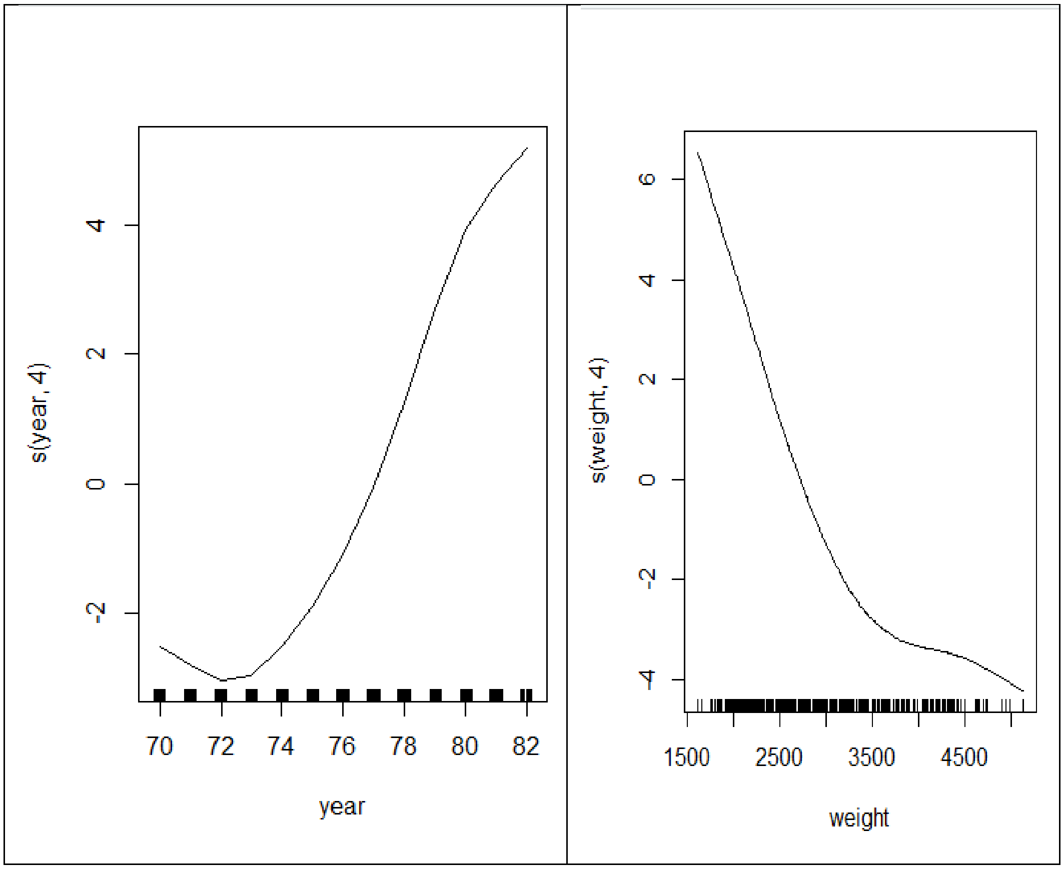
```
plot(bag.model_5a)
yhat.bag <- predict(bag.model_5a, newdata = model_5a.test)
MSE <- mean((yhat.bag - model_5a.test$mpg)^2)
```

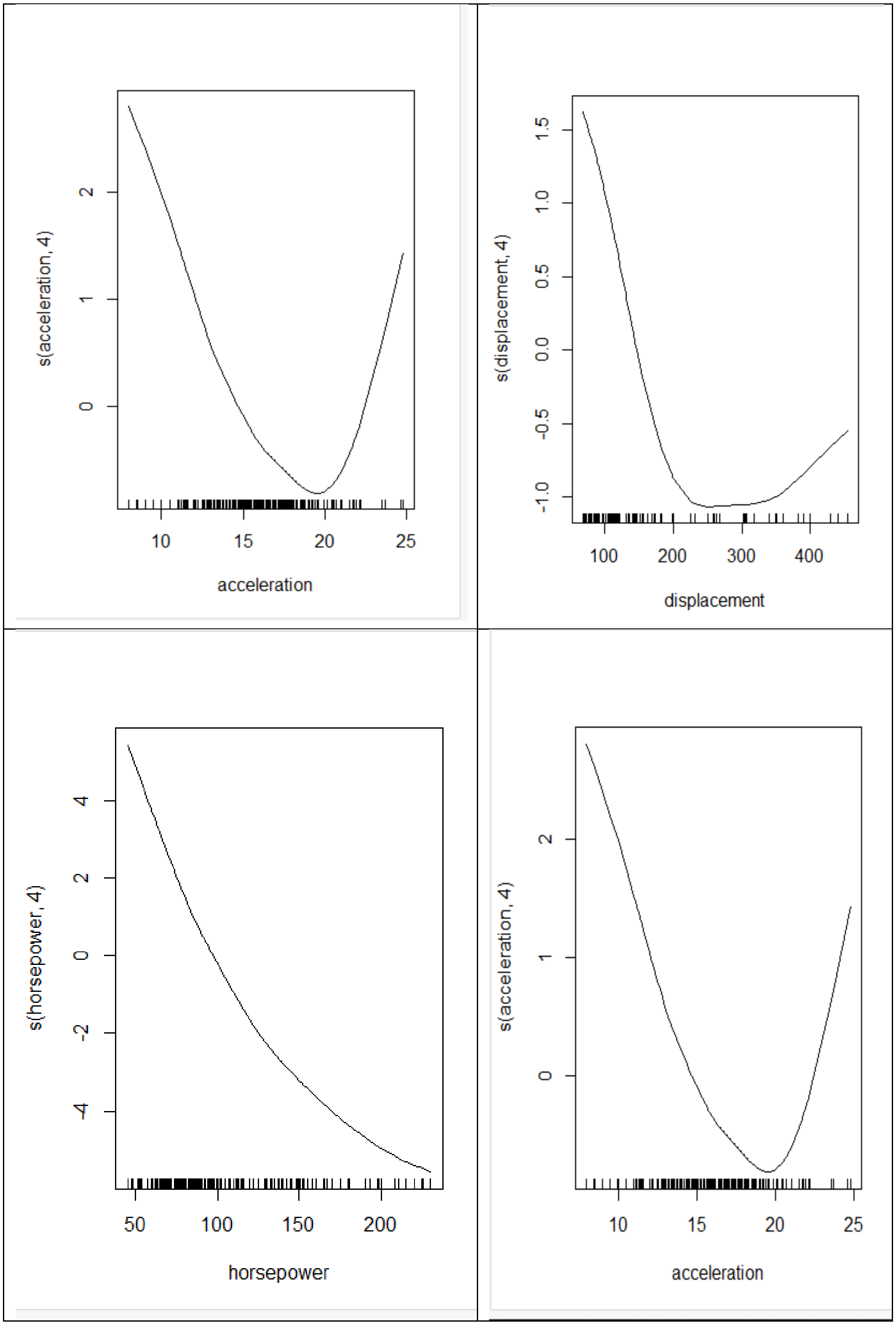
When calculating the MSE, we get **8.425** with an mtry of 5.

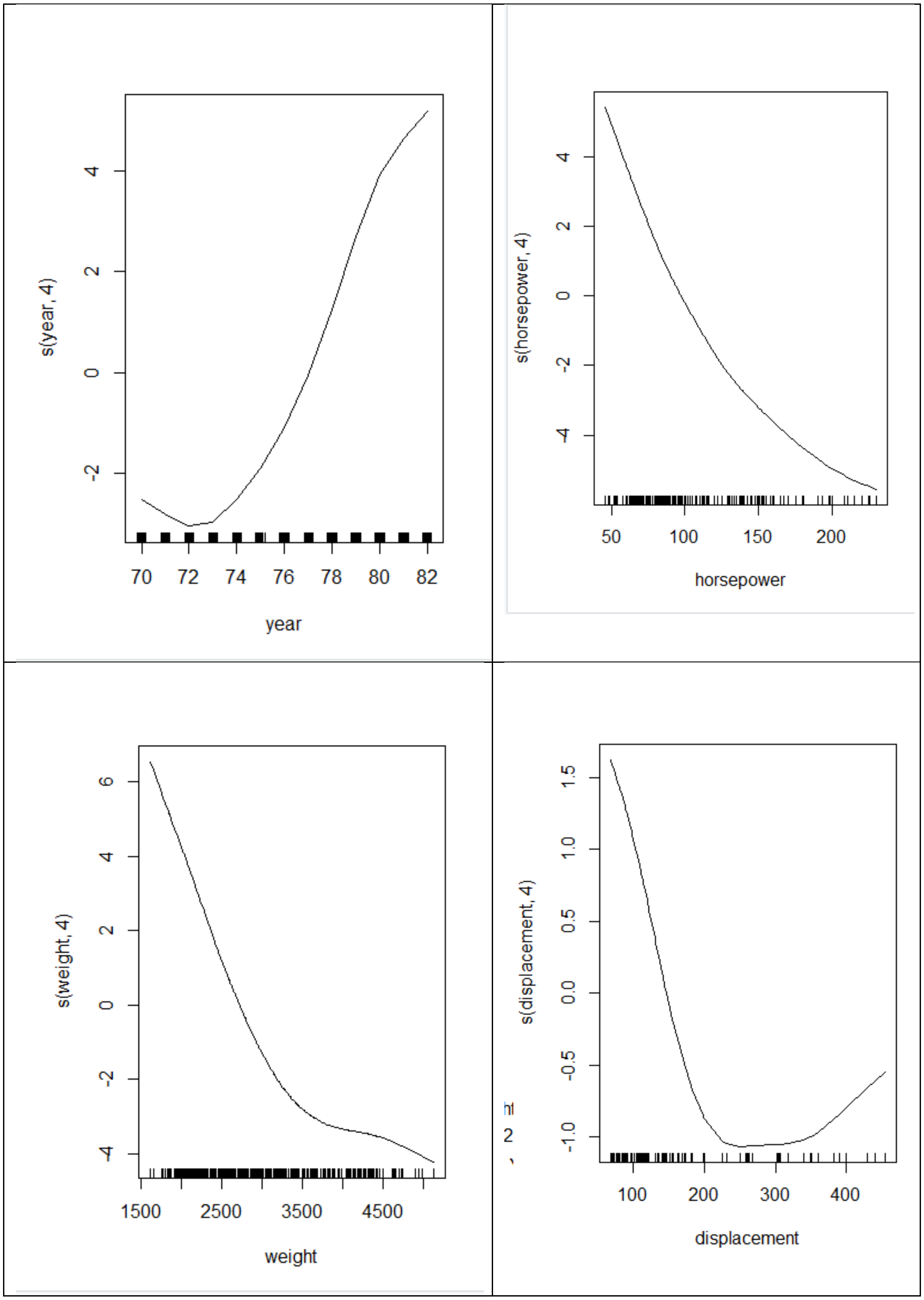
d. Using this code:

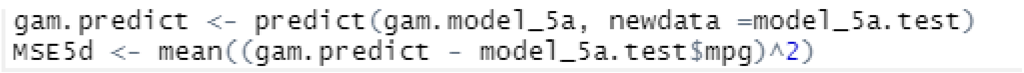


```
#5d
model_5a.train <- head(Auto, 300)
model_5a.test <- model_5a[-train, ]
model_5a.test <- model_5a.test[-c(2, 8,9) ]
library(gam)
gam.model_5a <- gam(mpg ~ s(displacement,4) + s(horsepower, 4) + s(weight,4) + s(acceleration,4) + s(year,4), data = model_5a.train)
plot.Gam(gam.model_5a)
gam.predict <- predict(gam.model_5a, newdata =model_5a.test)
MSE5d <- mean((gam.predict - model_5a.test$mpg)^2)
```



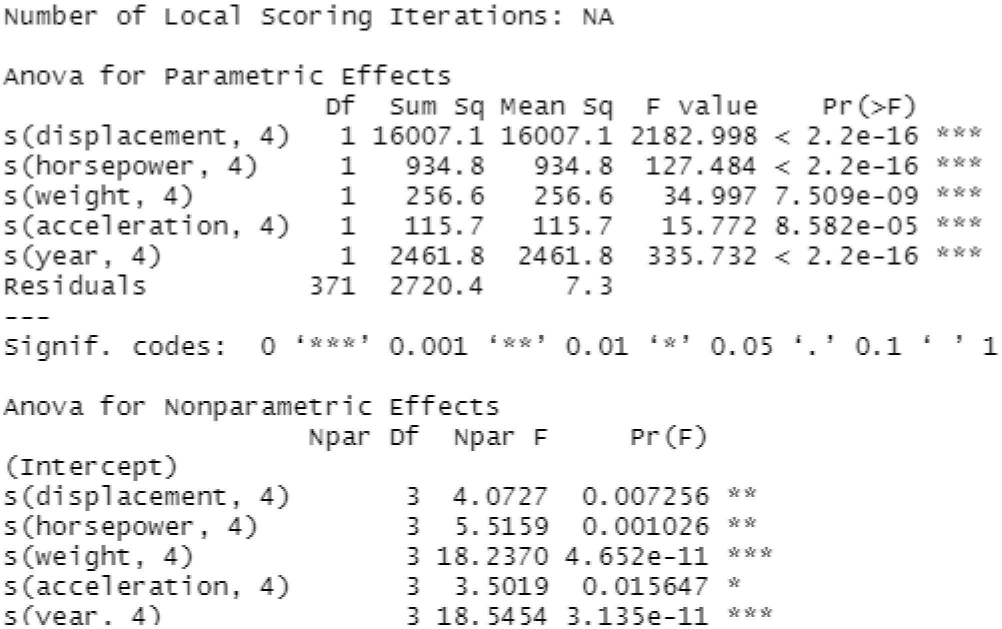

s(acceleration, 4)
0
1
2
10
15
20
25
acceleration
s(displacement, 4)
-1.0
-0.5
0.0
0.5
1.0
1.5
100
200
300
400
displacement
s(horsepower, 4)
-4
-2
0
2
4
50
100
150
200
horsepower
s(acceleration, 4)
0
1
2
10
15
20
25
acceleration





```
gam.predict <- predict(gam.model_5a, newdata =model_5a.test)
MSE5d <- mean((gam.predict - model_5a.test$mpg)^2)
```

After calculating the MSE on the GAM model, we find that the MSE is **7.504** which is less than the regression tree, random forest, and bagging. So it is the best model so far. Also, when doing summary(gam.model_5a), we get that the pvalues were less than 0.05 which indicates that they are signficant in estimating mpg.



```
Number of Local Scoring Iterations: NA

Anova for Parametric Effects
                     Df  Sum Sq Mean Sq  F value    Pr(>F)    
s(displacement, 4)   1 16007.1 16007.1 2182.998 < 2.2e-16 ***
s(horsepower, 4)     1   934.8   934.8  127.484 < 2.2e-16 ***
s(weight, 4)         1   256.6   256.6   34.997 7.509e-09 ***
s(acceleration, 4)   1   115.7   115.7   15.772 8.582e-05 ***
s(year, 4)           1  2461.8  2461.8  335.732 < 2.2e-16 ***
Residuals          371  2720.4     7.3                       
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Anova for Nonparametric Effects
                   Npar Df  Npar F     Pr(F)    
(Intercept)                                     
s(displacement, 4)       3  4.0727  0.007256 ** 
s(horsepower, 4)         3  5.5159  0.001026 ** 
s(weight, 4)             3 18.2370 4.652e-11 ***
s(acceleration, 4)       3  3.5019  0.015647 *  
s(year, 4)               3 18.5454 3.135e-11 ***
```

e. Considering both accuracy and interpretability of the fitted models, I prefer the random forest model. Even though it didn’t produce the best MSE, the error to tree graph made the most sense as it became clear when the optimal point of number trees would arise with the greatest reduction of error. With regression trees in this model, it becomes highly complex and increasingly less interpretable and its important to find when that sweet spot occurs which is what the bagged regression tree model tells us. The GAM model was the best model for accuracy, but I don’t think it was as interpretable as the random forests model.

There is an article written by Arrieta, Rodriguez, Ser about the concepts, taxonomies, opportunities and challenges of AI for interpretability.

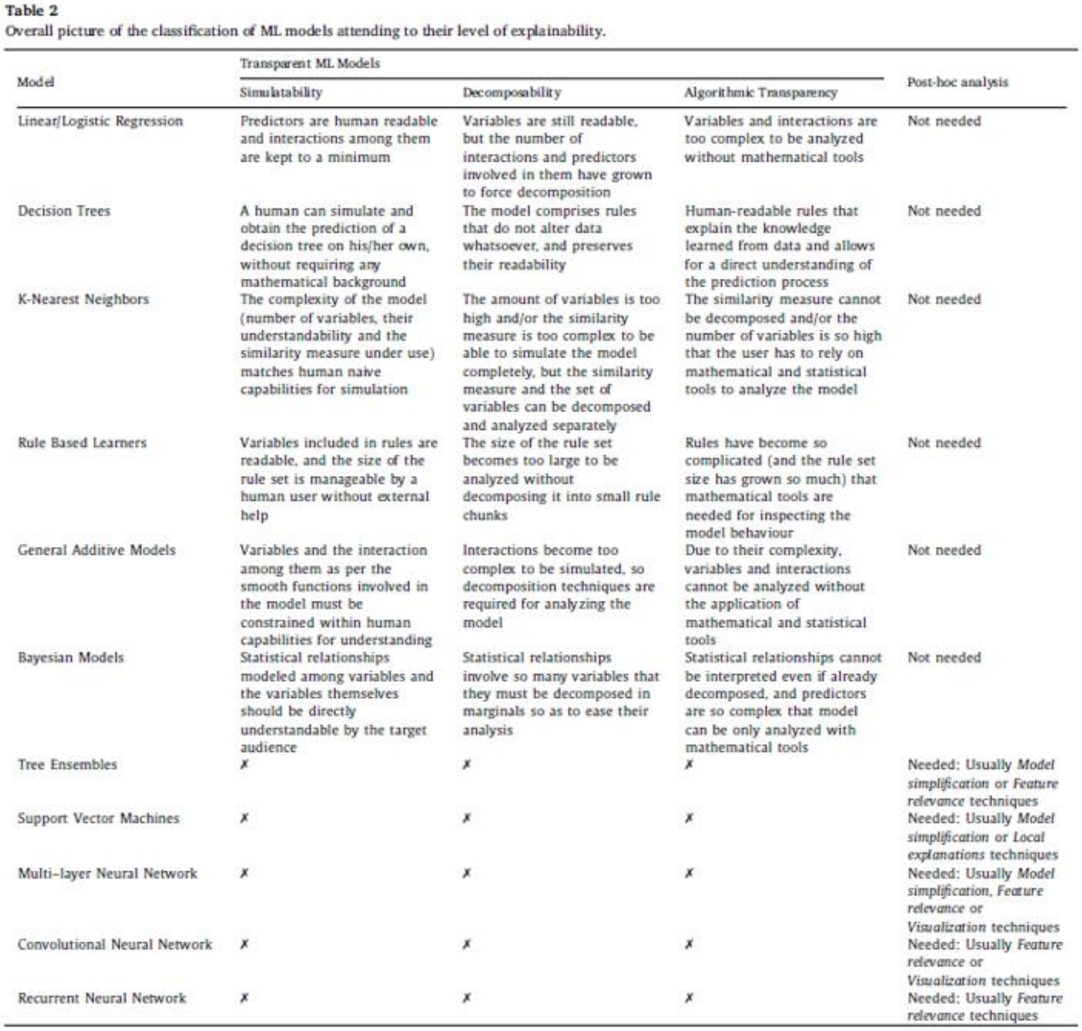


Table 2
Overall picture of the classification of ML models attending to their level of explainability.

Model	Transparent ML Models			Post-hoc analysis
	Simulatability	Decomposability	Algorithmic Transparency	
Linear/Logistic Regression	Predictors are human readable and interactions among them are kept to a minimum	Variables are still readable, but the number of interactions and predictors involved in them have grown to force decomposition	Variables and interactions are too complex to be analyzed without mathematical tools	Not needed
Decision Trees	A human can simulate and obtain the prediction of a decision tree on his/her own, without requiring any mathematical background	The model comprises rules that do not alter data whatsoever, and preserves their readability	Human-readable rules that explain the knowledge learned from data and allows for a direct understanding of the prediction process	Not needed
K-Nearest Neighbors	The complexity of the model (number of variables, their understandability and the similarity measure under use) matches human naive capabilities for simulation	The amount of variables is too high and/or the similarity measure is too complex to be able to simulate the model completely, but the similarity measure and the set of variables can be decomposed and analyzed separately	The similarity measure cannot be decomposed and/or the number of variables is so high that the user has to rely on mathematical and statistical tools to analyze the model	Not needed
Rule Based Learners	Variables included in rules are readable, and the size of the rule set is manageable by a human user without external help	The size of the rule set becomes too large to be analyzed without decomposing it into small rule chunks	Rules have become so complicated (and the rule set size has grown so much) that mathematical tools are needed for inspecting the model behaviour	Not needed
General Additive Models	Variables and the interaction among them as per the smooth functions involved in the model must be constrained within human capabilities for understanding	Interactions become too complex to be simulated, so decomposition techniques are required for analyzing the model	Due to their complexity, variables and interactions cannot be analyzed without the application of mathematical and statistical tools	Not needed
Bayesian Models	Statistical relationships modeled among variables and the variables themselves should be directly understandable by the target audience	Statistical relationships involve so many variables that they must be decomposed in marginals so as to ease their analysis	Statistical relationships cannot be interpreted even if already decomposed, and predictors are so complex that model can be only analyzed with mathematical tools	Not needed
Tree Ensembles	✗	✗	✗	Needed: Usually *Model simplification* or *Feature relevance* techniques
Support Vector Machines	✗	✗	✗	Needed: Usually *Model simplification* or *Local explanations* techniques
Multi-layer Neural Network	✗	✗	✗	Needed: Usually *Model simplification*, *Feature relevance* or *Visualization* techniques
Convolutional Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* or *Visualization* techniques
Recurrent Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* techniques

They have assembled this idea that as the models go down the list, they become more difficult to interpret to a human. Decision trees and regression trees are some of the easiest models for humans to interpret, and the trade off that the GAM model had in this exercise did not seem worth it. They divide the ideas into simulatability, decomposability, and algorithmic transparency. GAM algorithmic transparency drives at an exponential slope when it comes to interpretability for humans, therefore making it less interpretable and more difficult as more predictors come into play. The same goes for regression trees, but they are way more interpretable at the beginning states of tree nodes (especially when pruning). That’s why I believe random forests are somewhat in the middle of interpretability and they fit this problem quite well.