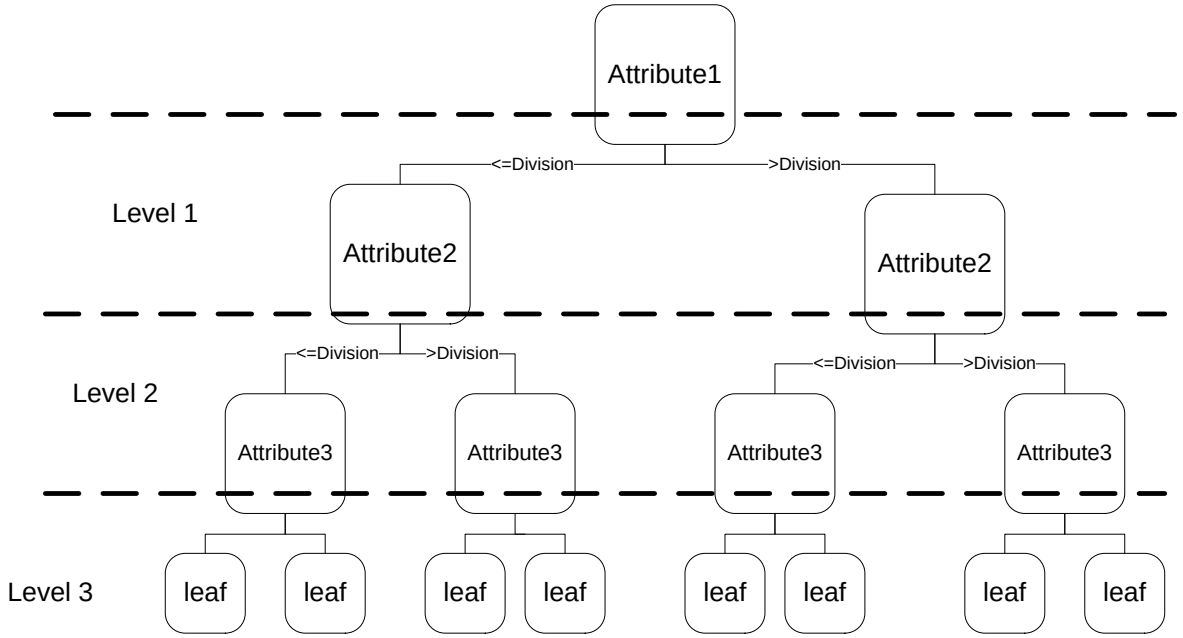
STAT745
HW# 17
Ekrem Kocaguneli

For 3 attributes and 3 levels we can think of 6 different configurations (1 feature at each level). A sample tree with that formation would look like as follows:



If we give numbers to the features of ozone dataset as follows: Solar.R = 1, Wind = 2 and Temp=3, then possible 6 combinations of 3 features to be used in each level would be as follows:

1. 1, 2, 3
2. 1, 3, 2
3. 2, 1, 3
4. 2, 3, 1
5. 3, 1, 2
6. 3, 2, 1

By using these 6 orderings, we need to choose the one that gives the minimum error. The results for 6 different orderings and the error yielded by these orderings are as follows (I used sum-squared error. Please note that I got rid of all the instances that had a NA value in dependent or independent variables.) :

- error1 = myTree(yClean, xClean, cbind(1,2,3)), where error1 = 985733.4
  - Level: 0 , Attribute: 1 , Division: 149
  - Level: 1 , Attribute: 2 , Division: 6.3
  - Level: 2 , Attribute: 3 , Division: 79
  - Level: 2 , Attribute: 3 , Division: 82
  - Level: 1 , Attribute: 2 , Division: 8
  - Level: 2 , Attribute: 3 , Division: 78
  - Level: 2 , Attribute: 3 , Division: 83

- error2 = myTree(yClean, xClean, cbind(1,3,2)), where error2 = 294860.3
    - Level: 1 , Attribute: 1 , Division: 149
    - Level: 2 , Attribute: 3 , Division: 77
    - Level: 3 , Attribute: 2 , Division: 9.2
    - Level: 3 , Attribute: 2 , Division: 6.3
    - Level: 2 , Attribute: 3 , Division: 78
    - Level: 3 , Attribute: 2 , Division: 7.4
    - Level: 3 , Attribute: 2 , Division: 8.6
- error3 = myTree(yClean, xClean, cbind(2,1,3)), where error3 = 993000.3
    - Level: 1 , Attribute: 2 , Division: 6.3
    - Level: 2 , Attribute: 1 , Division: 197
    - Level: 3 , Attribute: 3 , Division: 83
    - Level: 3 , Attribute: 3 , Division: 77
    - Level: 2 , Attribute: 1 , Division: 149
    - Level: 3 , Attribute: 3 , Division: 82
    - Level: 3 , Attribute: 3 , Division: 82
- error4 = myTree(yClean, xClean, cbind(2,3,1)), where error4 = 731731.1
    - Level: 1 , Attribute: 2 , Division: 6.3
    - Level: 2 , Attribute: 3 , Division: 77
    - Level: 3 , Attribute: 1 , Division: 238
    - Level: 3 , Attribute: 1 , Division: 197
    - Level: 2 , Attribute: 3 , Division: 84
    - Level: 3 , Attribute: 1 , Division: 81
    - Level: 3 , Attribute: 1 , Division: 229
- error5 = myTree(yClean, xClean, cbind(3,1,2)), where error5 = 249639.8
    - Level: 1 , Attribute: 3 , Division: 82
    - Level: 2 , Attribute: 1 , Division: 149
    - Level: 3 , Attribute: 2 , Division: 6.3
    - Level: 3 , Attribute: 2 , Division: 5.7
    - Level: 2 , Attribute: 1 , Division: 95
    - Level: 3 , Attribute: 2 , Division: 12
    - Level: 3 , Attribute: 2 , Division: 10.3
- error6 = myTree(yClean, xClean, cbind(3,2,1)), where error6 = 1080384
    - Level: 1 , Attribute: 3 , Division: 82
    - Level: 2 , Attribute: 2 , Division: 5.7
    - Level: 3 , Attribute: 1 , Division: 223
    - Level: 3 , Attribute: 1 , Division: 49
    - Level: 2 , Attribute: 2 , Division: 10.3
    - Level: 3 , Attribute: 1 , Division: 95
    - Level: 3 , Attribute: 1 , Division: 291

So the 6 different error values we get for 6 different orderings are as follows:

1. 985,733.4
2. 294,860.3
3. 993,000.3
4. 731,731.1
5. **249,639.8**
6. 1,080,384

The smallest error rate is given by ordering 5, which orders the features as 3, 1, 2 and it corresponds to using Temp as root, then Solar.R as the next feature and Wind as the last feature. The corresponding divisons are as follows:

Level: 1 , Attribute: 3 (Temp) , Division: 82
Level: 2 , Attribute: 1 (Solar.R) , Division: 149
Level: 3 , Attribute: 2 (Wind) , Division: 6.3
Level: 3 , Attribute: 2 (Wind) , Division: 5.7
Level: 2 , Attribute: 1 (Solar.R) , Division: 95
Level: 3 , Attribute: 2 (Wind) , Division: 12
Level: 3 , Attribute: 2 (Wind) , Division: 10.3

CODE:

```
# define the dependent and independent variables
y = airquality[,1]
x = airquality[,-1]

# firstly get rid of the instances with NA values
yClean = y[-which(is.na(y) == TRUE)]
xClean = x[-which(is.na(y) == TRUE),]
yClean = yClean[-which(is.na(xClean[,1]) == TRUE)]
xClean = xClean[-which(is.na(xClean[,1]) == TRUE),]

findMyDivision <- function(myX, myY){
 division = myX[1]
 part1 = myY[which(myX<=division)]
 part2 = myY[which(myX>division)]
 error = sum((part1-mean(part1))^2) + sum((part2-mean(part2))^2)
 for(i in 2:length(myY)){
  tmpdivision = myX[i]
  part1 = myY[which(myX<=tmpdivision)]
  part2 = myY[which(myX>tmpdivision)]
  tmperror = sum((part1-mean(part1))^2) + sum((part2-mean(part2))^2)
  if(tmperror < error){
   error = tmperror
   division = tmpdivision
  }
 }
 return(division)
}

myTree <- function(myY, myX, myAttributes){
 if(length(myAttributes)==1){
    myAttribute = myAttributes[1]
    myDivision = findMyDivision(myX[,myAttribute], myY)
    myLeft = myY[which(myX[,myAttribute]<=myDivision)]
    myRight = myY[which(myX[,myAttribute]>myDivision)]
    cat("Level: ",(4-length(myAttributes)), ", Attribute: ", myAttribute, ", Division: ", myDivision, "\n")
    return(sum((myLeft - mean(myLeft))^2) + sum((myRight + mean(myRight))^2))
 }
 else{
  myAttribute = myAttributes[1]
  myDivision = findMyDivision(myX[,myAttribute], myY)
  myLeftY = myY[which(myX[,myAttribute]<=myDivision)]
  myRightY = myY[which(myX[,myAttribute]>myDivision)]
  myLeftX = myX[which(myX[,myAttribute]<=myDivision),]
  myRightX = myX[which(myX[,myAttribute]>myDivision),]
  cat("Level: ",(4-length(myAttributes)), ", Attribute: ", myAttribute, ", Division: ", myDivision, "\n")
  return(myTree(myLeftY, myLeftX, myAttributes[-1]) +  myTree(myRightY, myRightX, myAttributes[-1]))
 }
}

error1 = myTree(yClean, xClean, cbind(1,2,3))
error2 = myTree(yClean, xClean, cbind(1,3,2))
error3 = myTree(yClean, xClean, cbind(2,1,3))
error4 = myTree(yClean, xClean, cbind(2,3,1))
error5 = myTree(yClean, xClean, cbind(3,1,2))
error6 = myTree(yClean, xClean, cbind(3,2,1))
```