Lab1- Arvores

Table of contents

# 1. Lab 1

1-) Crie um documento Word e identifique-o com o nome do laboratório, data de elaboração e o seu nome ou da dupla que o elaborou

2-) Crie um tópico para cada resultado que você considerar relevante (manipulação de dados ou resultado de algum processamento) identificando-o com um título e uma breve explicação. Os resultados podem ser imagens de gráficos gerados ou de listas de valores ou dados de resultados obtidos. Não devem ser incluídos os scripts ou instruções de processamento utilizados, inclua apenas os resultados que você considerar relevantes

3-) No final do relatório crie um último tópico denominado “Conclusões” e elabore comentários, sugestões e conclusões sobre o que você pode aprender com a elaboração deste laboratório.

# 2. Árvores de Classificação

## 2.1 Iniciando Ambiente e variáveis Globais

library(tidyverse)   
library(dplyr)  
library(ggplot2)  
library(ggExtra)  
library(hrbrthemes)  
library(RColorBrewer) # for the color palette  
library(GGally)  
library(ISLR)  
library(tree)  
library(caret)  
  
carseats <- ISLR::Carseats  
  
ALVO <- 8 #definir se sera uma venda alta ou baixa

## 2.2 Exploração dos dados

### 2.2.1 Conhecendo os dados

A base de dados escolhida é a carseats do pacote ISRL

Um quadro de dados com 400 observações nas 11 variáveis a seguir.

* Sales: Vendas unitárias (em milhares) em cada local
* CompPric: Preço cobrado pelo concorrente em cada local
* Income: Nível de renda da comunidade (em milhares de dólares)
* Advertising - Orçamento de publicidade local para a empresa em cada local (em milhares de dólares)
* Population: Tamanho da população na região (em milhares)
* Price: Preço que a empresa cobra pelas cadeirinhas em cada local
* ShelveLoc: Um fator com níveis Ruim, Bom e Médio que indica a qualidade da localização das estantes para as cadeirinhas em cada local
* Age: Idade média da população local
* Education: Nível de educação em cada local
* Urban: Fator com níveis Não e Sim para indicar se a loja está localizada em área urbana ou rural
* US: Um fator com níveis Não e Sim para indicar se a loja está nos EUA ou não

carseats |> head()

Sales CompPrice Income Advertising Population Price ShelveLoc Age Education  
1 9.50 138 73 11 276 120 Bad 42 17  
2 11.22 111 48 16 260 83 Good 65 10  
3 10.06 113 35 10 269 80 Medium 59 12  
4 7.40 117 100 4 466 97 Medium 55 14  
5 4.15 141 64 3 340 128 Bad 38 13  
6 10.81 124 113 13 501 72 Bad 78 16  
 Urban US  
1 Yes Yes  
2 Yes Yes  
3 Yes Yes  
4 Yes Yes  
5 Yes No  
6 No Yes

carseats |> str() # tipos dos dados

'data.frame': 400 obs. of 11 variables:  
 $ Sales : num 9.5 11.22 10.06 7.4 4.15 ...  
 $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
 $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
 $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
 $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
 $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
 $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
 $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
 $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
 $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
 $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...

carseats |> dim() # tamanho da tabela

[1] 400 11

carseats |> is.na() |> sum()

[1] 0

carseats |> summary()

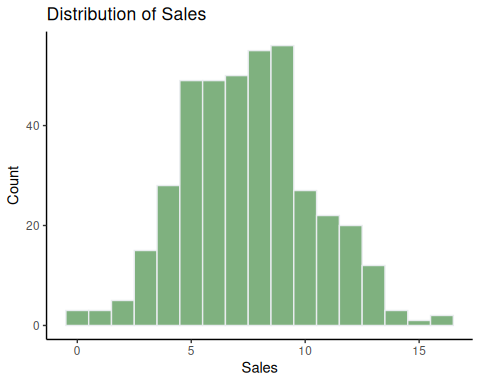
Sales CompPrice Income Advertising   
 Min. : 0.000 Min. : 77 Min. : 21.00 Min. : 0.000   
 1st Qu.: 5.390 1st Qu.:115 1st Qu.: 42.75 1st Qu.: 0.000   
 Median : 7.490 Median :125 Median : 69.00 Median : 5.000   
 Mean : 7.496 Mean :125 Mean : 68.66 Mean : 6.635   
 3rd Qu.: 9.320 3rd Qu.:135 3rd Qu.: 91.00 3rd Qu.:12.000   
 Max. :16.270 Max. :175 Max. :120.00 Max. :29.000   
 Population Price ShelveLoc Age Education   
 Min. : 10.0 Min. : 24.0 Bad : 96 Min. :25.00 Min. :10.0   
 1st Qu.:139.0 1st Qu.:100.0 Good : 85 1st Qu.:39.75 1st Qu.:12.0   
 Median :272.0 Median :117.0 Medium:219 Median :54.50 Median :14.0   
 Mean :264.8 Mean :115.8 Mean :53.32 Mean :13.9   
 3rd Qu.:398.5 3rd Qu.:131.0 3rd Qu.:66.00 3rd Qu.:16.0   
 Max. :509.0 Max. :191.0 Max. :80.00 Max. :18.0   
 Urban US   
 No :118 No :142   
 Yes:282 Yes:258

### 2.2.2 Criando a variavel alvo

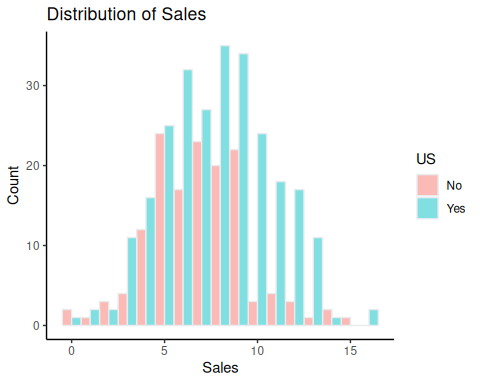
carseats <- carseats |>  
 dplyr::mutate(High = as.factor(ifelse(Sales <= ALVO,"No","Yes")))

### 2.2.3 Visualizações

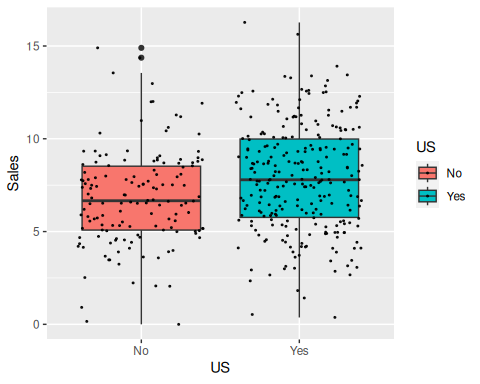
carseats |>   
 ggplot(aes(x = Sales)) +  
 geom\_histogram(binwidth = 1,color="#e9ecef",alpha = 0.5, fill = "darkgreen", position = 'identity') +   
 labs(x = "Sales", y = "Count", title = "Distribution of Sales") +  
 theme\_classic()



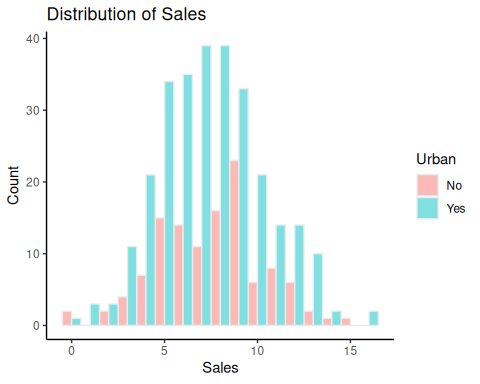
carseats |>   
 ggplot(aes(x = Sales,fill=US)) +  
 geom\_histogram(binwidth = 1,color="#e9ecef",alpha = 0.5,position = 'dodge') +   
 labs(x = "Sales", y = "Count", title = "Distribution of Sales") +  
 theme\_classic()



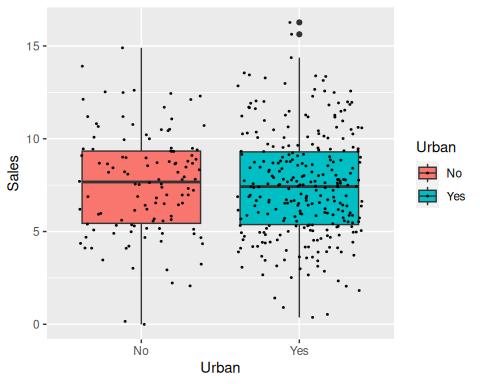
carseats |>  
 ggplot(aes(x=US,y=Sales,fill=US)) +   
 geom\_boxplot() +   
 geom\_jitter(color="black", size=0.4, alpha=0.9)



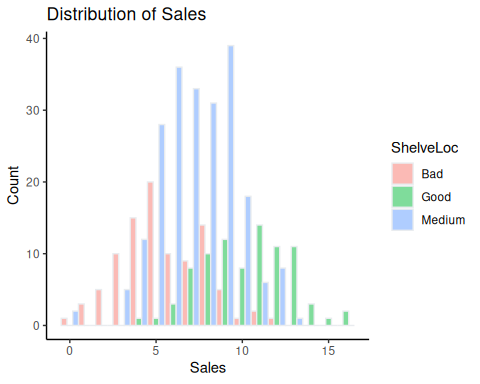
carseats |>   
 ggplot(aes(x = Sales,fill=Urban)) +  
 geom\_histogram(binwidth = 1,color="#e9ecef",alpha = 0.5,position = 'dodge') +   
 labs(x = "Sales", y = "Count", title = "Distribution of Sales") +  
 theme\_classic()



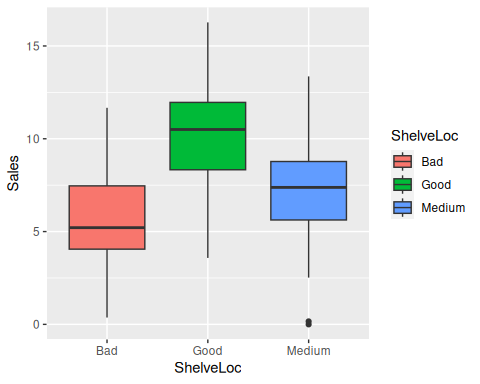
carseats |>  
 ggplot(aes(x=Urban,y=Sales,fill=Urban)) +   
 geom\_boxplot() +   
 geom\_jitter(color="black", size=0.4, alpha=0.9)



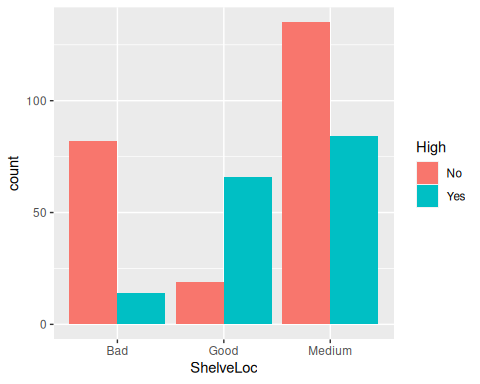
carseats |>   
 ggplot(aes(x = Sales,fill=ShelveLoc)) +  
 geom\_histogram(binwidth = 1,color="#e9ecef",alpha = 0.5,position = 'dodge') +   
 labs(x = "Sales", y = "Count", title = "Distribution of Sales") +  
 theme\_classic()

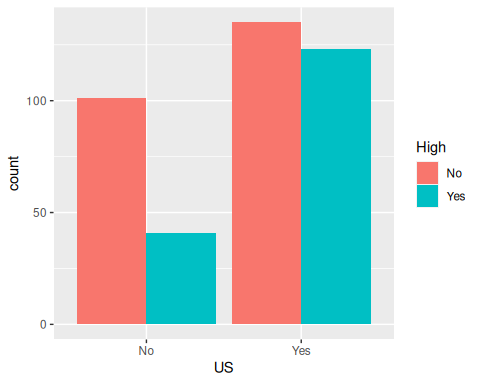


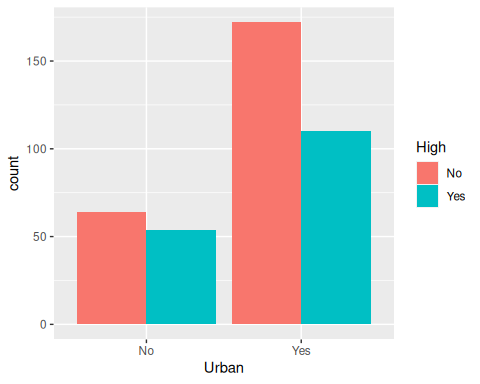
carseats |>  
 ggplot(aes(x=ShelveLoc,y=Sales,fill=ShelveLoc)) +   
 geom\_boxplot()



for(col in c("ShelveLoc","US","Urban")){  
 print(carseats |>  
 ggplot(aes(!!sym(col))) + geom\_bar(aes(fill = High),position = 'dodge'))  
}







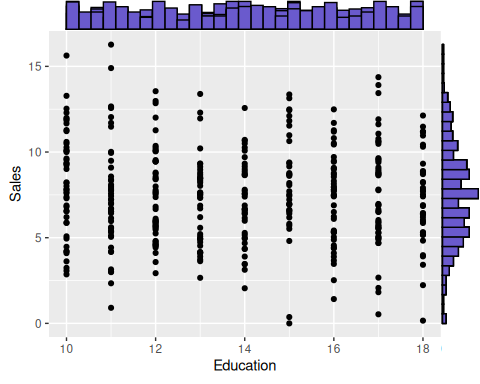
A Distribuição das vendas muda conforme a qualidade da localização nas estantes . Isso **pode** facilitar para o modelo distinguir a variavel alvo, e portanto essa seria uma boa variável.

Regiões Urbanas apresentam uma distribuição parecida com Regiões Rurais. Sem grande possibilidade de discriminação pelo modelo

As distribuições para a variável US são parecidas. Mediana das vendas é um pouco maior para vendas no Estados unidos

* Relações entre variáveis quantitativas

for(col in names(subset(carseats, select = -c(ShelveLoc,US,Urban,High,Sales)))){  
 p <- ggplot(carseats, aes(y=Sales, x=!!sym(col))) +  
 geom\_point() +  
 theme(legend.position="none")  
   
   
 p1 <- ggMarginal(p, type="histogram", size=10,fill = "slateblue")  
 print(p1)  
}



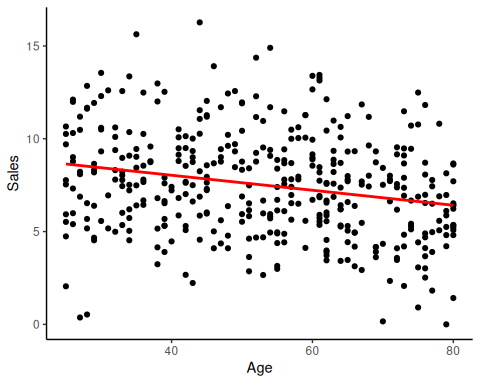
Olhando para esses gráficos. Somente “Preço x Vendas” possui uma relação linear. Verificamos que as cadeiras possuem relação elástica em suas vendas , quanto maior o preço , menor a quantidade vendida

Não aparenta haver uma alta elasticidade cruzada da demanda. Um Aumento no preço do concorrente , não aumenta a venda das cadeirinhas.

A relação entre Vendas e idade média da população parece seguir uma tendência negativa (maior a ideade média , menor a quantidade vendida, o que parece fazer sentido pois Idosos geralmente não são responsáveis por bebês). Mas não é algo muito forte

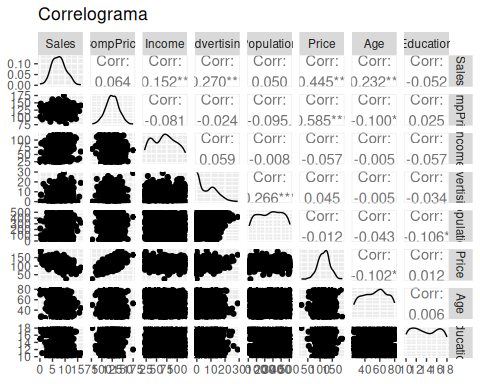
carseats |>  
ggplot( aes(x=Age, y=Sales)) +  
 geom\_point() +  
 geom\_smooth(method=lm , color="red", se=FALSE) +  
 theme\_classic()

`geom\_smooth()` using formula = 'y ~ x'



* Correlações

carseats\_numeric <-subset(carseats, select = -c(ShelveLoc,US,Urban,High))  
  
ggpairs(carseats\_numeric, title="Correlograma")

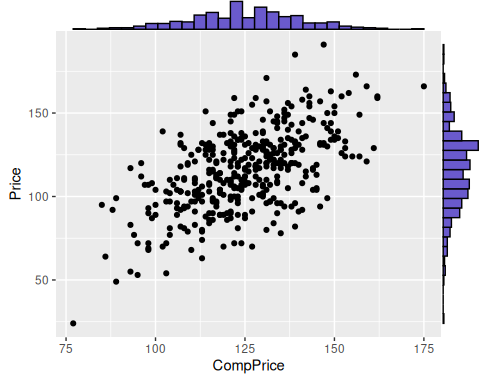


ggcorr(carseats\_numeric, method = c("everything", "pearson"),label = TRUE)



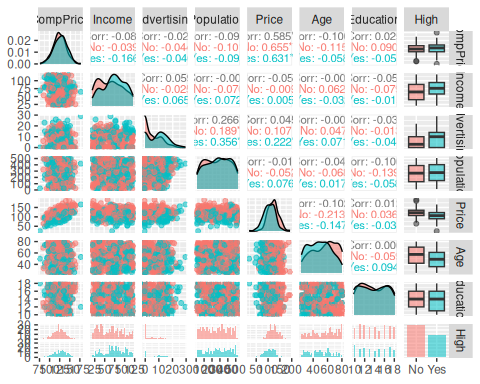
Variáveis explicativas possuem baixa correlação entre elas. Chamou atenção Preço x Preço dos concorrentes. Aparentemente tais variáveis apresentam uma relação linear, conforme a figura a seguir:

p <- ggplot(carseats, aes(y=Price, x=CompPrice)) +  
 geom\_point() +  
 theme(legend.position="none")  
  
  
p1 <- ggMarginal(p, type="histogram", size=10,fill = "slateblue")  
print(p1)

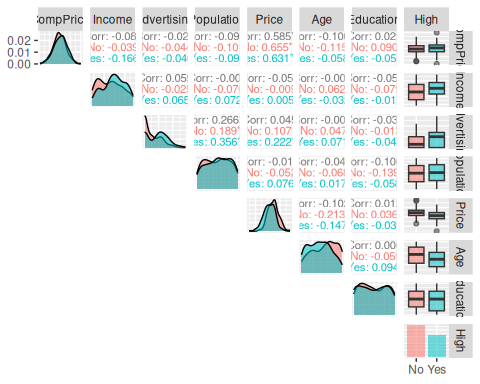


ggpairs(subset(carseats, select = -c(ShelveLoc,US,Urban,Sales)), ggplot2::aes(colour=High,alpha=0.5),upper = list(continuous = wrap("cor", size = 3)))

`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
`stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggpairs(subset(carseats, select = -c(ShelveLoc,US,Urban,Sales)), ggplot2::aes(colour=High,alpha=0.5),upper = list(continuous = wrap("cor", size = 3)), lower = "blank")



As distribuições não mudam muito dentro de cada categoria

## 2.3 Modelagem

### 2.3.1 Separando entre treino e teste

Obs: não modelando os dados completos para evitar viéses (seleção de variáveis, data leakage,etc )

carseats\_trainIndex <- createDataPartition(carseats$High, p = 0.75, list = FALSE)  
  
training\_carseats <- carseats[carseats\_trainIndex, ] |>  
 dplyr::select(-Sales)  
  
test\_carseats <- carseats[-carseats\_trainIndex, ]|>  
 dplyr::select(-Sales)  
  
print("Train dimentions: ")

[1] "Train dimentions: "

dim(training\_carseats)

[1] 300 11

print("Test dimentions: ")

[1] "Test dimentions: "

dim(test\_carseats)

[1] 100 11

head(training\_carseats)

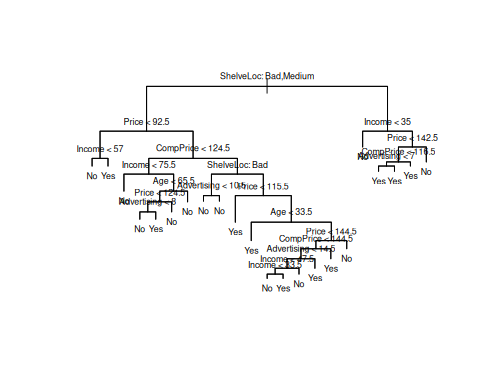
CompPrice Income Advertising Population Price ShelveLoc Age Education Urban  
1 138 73 11 276 120 Bad 42 17 Yes  
2 111 48 16 260 83 Good 65 10 Yes  
3 113 35 10 269 80 Medium 59 12 Yes  
4 117 100 4 466 97 Medium 55 14 Yes  
5 141 64 3 340 128 Bad 38 13 Yes  
6 124 113 13 501 72 Bad 78 16 No  
 US High  
1 Yes Yes  
2 Yes Yes  
3 Yes Yes  
4 Yes No  
5 No No  
6 Yes Yes

head(test\_carseats)

CompPrice Income Advertising Population Price ShelveLoc Age Education Urban  
10 132 113 0 131 124 Medium 76 17 No  
14 115 28 11 29 86 Good 53 18 Yes  
15 107 117 11 148 118 Good 52 18 Yes  
22 134 29 12 239 109 Good 62 18 No  
26 139 32 0 176 82 Good 54 11 No  
28 98 118 0 19 107 Medium 64 17 Yes  
 US High  
10 Yes No  
14 Yes Yes  
15 Yes Yes  
22 Yes Yes  
26 No Yes  
28 No No

### 2.3.2 Obtendo árvore

set.seed(42)  
options(repr.plot.width=10, repr.plot.height=8) # set plot size  
tree\_carseats\_train = tree(High ~ ., data= training\_carseats)  
plot(tree\_carseats\_train )  
text(tree\_carseats\_train , pretty=0,cex=0.55)



Olhando apenas pera esse split , temos que a variável mais importante para o modelodecidir se teremos uma venda alta ou não é a Qualidade da disposição do produto na prateleira.

Para o lado direito da árvore temos que para lojas fora dos Estados Unidos, mesmo em qualidades da prateleira média ou baixa, as vendas serão altas se o preço for menor do que o terceiro quartil (3rd Qu.:131.0 , olhando para o dataset completo nas primeiras etapas). É possível que nessas condições não seja possível a empresa repassar preços masi caros

Os recortes de idade quase sempre são para médias abaixo de 40

A Árvore *parece* não ter alta complexidade dado o seu tamanho e número de recortes (avaliação subjetiva). Provavlemente teremos que reduzir seu tamanho para melhorar a variância (como demonstraremos no tópico a seguir)

* Performance no treino

predict\_train = predict(tree\_carseats\_train, training\_carseats, type = "class")  
  
conf\_matrix\_train <- confusionMatrix(data = predict\_train, reference = training\_carseats$High)  
  
conf\_matrix\_train

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 167 12  
 Yes 10 111  
   
 Accuracy : 0.9267   
 95% CI : (0.8911, 0.9535)  
 No Information Rate : 0.59   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.848   
   
 Mcnemar's Test P-Value : 0.8312   
   
 Sensitivity : 0.9435   
 Specificity : 0.9024   
 Pos Pred Value : 0.9330   
 Neg Pred Value : 0.9174   
 Prevalence : 0.5900   
 Detection Rate : 0.5567   
 Detection Prevalence : 0.5967   
 Balanced Accuracy : 0.9230   
   
 'Positive' Class : No

* Performance no teste

predict\_test <- predict(tree\_carseats\_train,test\_carseats,type='class') #prevendo coms os o modelo do treino  
  
conf\_matrix\_test <- caret::confusionMatrix(data = predict\_test,reference = test\_carseats$High)  
  
conf\_matrix\_test

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 47 13  
 Yes 12 28  
   
 Accuracy : 0.75   
 95% CI : (0.6534, 0.8312)  
 No Information Rate : 0.59   
 P-Value [Acc > NIR] : 0.0006044   
   
 Kappa : 0.4813   
   
 Mcnemar's Test P-Value : 1.0000000   
   
 Sensitivity : 0.7966   
 Specificity : 0.6829   
 Pos Pred Value : 0.7833   
 Neg Pred Value : 0.7000   
 Prevalence : 0.5900   
 Detection Rate : 0.4700   
 Detection Prevalence : 0.6000   
 Balanced Accuracy : 0.7398   
   
 'Positive' Class : No

# checando o balanceamento dos dados  
  
test\_carseats |>   
 count(High) |>   
 mutate(prop= n/ sum(n))

High n prop  
1 No 59 0.59  
2 Yes 41 0.41

Apesar de uma boa acurácia 76% (nos dados de a proporção erade 59%-41% par avendas baixas e altas, ou seja, foi melhor do que um chute aleatório), a árvore consegue prever um pouco melhor melhor para casos em que **não temos venda alta** (de 59 acertou 47). Enquanto que para caso de ocorrência de **venda alta**, o modelo **não performou muito bem** (acertou 29 de 41). Pensando no ponto de vista de negócio, saber o que faz vender mais é melhor do que saber o que não faz vender. Esta está última característica passa a ser importante quando o custo de estocagem é alto. Faltam detalhes para saber o custo de uma venda perdida relativa ao ganho em uma venda . Ou seja , não sabemos o impacto financeiro do modelo prever *YES* , para quando na verdade era *NO*, e vice-versa

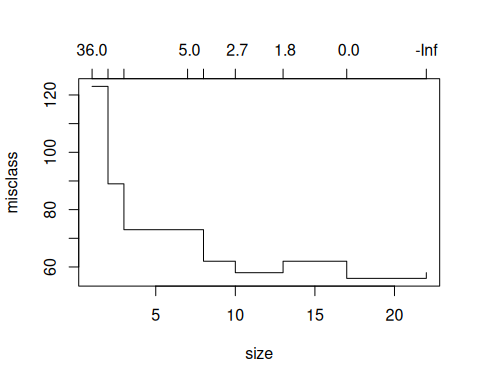
Vale ressaltar que o modelo não apresenta certa estabilidade entre treino e teste. O desempenho cai consideravlemnete na maioria das métricas. A acurácia , por exemplo, vai de 0.9167 para 0.76. Isso pode indicar um sobreajuste (overfitting) de nosso modelo, se tronou muito específico com os dados de treino, e incapaz de generalizar aos dados de teste.

### 2.3.3 Implementando Validação Cruzada (poda da árvore)

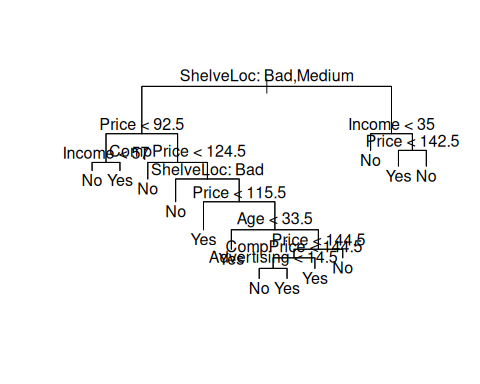
cv.carseats = cv.tree(tree\_carseats\_train, FUN = prune.misclass)  
cv.carseats

$size  
[1] 22 17 13 10 8 7 3 2 1  
  
$dev  
[1] 58 56 62 58 62 73 73 89 123  
  
$k  
[1] -Inf 0.000000 1.750000 2.666667 3.000000 5.000000 6.000000  
[8] 15.000000 36.000000  
  
$method  
[1] "misclass"  
  
attr(,"class")  
[1] "prune" "tree.sequence"

plot(cv.carseats)



prune.carseats = prune.misclass(tree\_carseats\_train, best = 12)  
plot(prune.carseats)  
text(prune.carseats, pretty=0)



prune\_tree\_pred = predict(prune.carseats, test\_carseats, type="class")  
  
prune\_conf\_matrix <- caret::confusionMatrix(data = prune\_tree\_pred,reference = test\_carseats$High)  
  
prune\_conf\_matrix

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 49 15  
 Yes 10 26  
   
 Accuracy : 0.75   
 95% CI : (0.6534, 0.8312)  
 No Information Rate : 0.59   
 P-Value [Acc > NIR] : 0.0006044   
   
 Kappa : 0.4735   
   
 Mcnemar's Test P-Value : 0.4237108   
   
 Sensitivity : 0.8305   
 Specificity : 0.6341   
 Pos Pred Value : 0.7656   
 Neg Pred Value : 0.7222   
 Prevalence : 0.5900   
 Detection Rate : 0.4900   
 Detection Prevalence : 0.6400   
 Balanced Accuracy : 0.7323   
   
 'Positive' Class : No

Acurácia ficou quase a mesma. Quanto as demais métricas, apenas a sensibilidade melhorou,as demais pioraram. A acurácia na previsão de baixas (*NO*) melhrou, porém a de vendas altas (*YES*) piorou.

## 2.4 Conclusão

Sem grandes ganhos ou reduzir a complexidade do modelo. Certamente modelos de bagging podem ajudar a melhorar o desempenho e reduzir a variância. Como não temos dados ausentes (nulos), classes desbalanceadas, grandes volumes e baixa complexidade, é *possível* que *neste caso* florestas aleatórias e boosting não trariam muitos ganhso (conforme o indicado no material), seria necessário relaizar os experimentos para comprovar esta tese. Tivemos também algumas diferenças com os resultados obtidos na apostila, já que mudamos um pouco a abordagem do problema (e.g train-test-split).

# 3. Árvores de Regressão

## 3.1 Iniciando Ambiente e variáveis Globais

library(MASS)

Attaching package: 'MASS'

The following object is masked from 'package:dplyr':  
  
 select

library(randomForest)

randomForest 4.7-1.1

Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:dplyr':  
  
 combine

The following object is masked from 'package:ggplot2':  
  
 margin

set.seed(42)  
  
boston <- MASS::Boston

## 3.2 Exploração dos dados

### 3.2.1 Conhecendo os dados

• crim: taxa de criminalidade per capita por bairro. • zn: proporção de terrenos residenciais zoneados para lotes acima de 25.000 sqft. • indus: proporção de acres comerciais e não comerciais por bairro. • chas: variável fictícia Charles River (=1 se o trecho é próximo ao rio; =0 em caso contrário). • nox: concentração de óxidos de nitrogênio (partes por 10 milhões). • rm: número médio de quartos por habitação. • age: proporção de idade das unidades ocupadas pelo proprietário construídas antes de 1940. • dis: média ponderada de distâncias para cinco centros de emprego de Boston. • rad: índice de acessibilidade às rodovias radiais da bairro. • tax: imposto propriedade de valor total taxa de imposto por $10000. • ptratio: relação aluno-professor por bairro. • black: 1000 (bk − 0,63)2 onde bk é a proporção de negros por bairro. • lstat: população de baixo status (porcentagem). • medv: valor mediano de casas ocupadas pelo proprietário em $1000s

boston |> head()

crim zn indus chas nox rm age dis rad tax ptratio black lstat  
1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90 4.98  
2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14  
3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03  
4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94  
5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33  
6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 394.12 5.21  
 medv  
1 24.0  
2 21.6  
3 34.7  
4 33.4  
5 36.2  
6 28.7

boston |> str() # tipos dos dados

'data.frame': 506 obs. of 14 variables:  
 $ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...  
 $ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...  
 $ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...  
 $ chas : int 0 0 0 0 0 0 0 0 0 0 ...  
 $ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...  
 $ rm : num 6.58 6.42 7.18 7 7.15 ...  
 $ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...  
 $ dis : num 4.09 4.97 4.97 6.06 6.06 ...  
 $ rad : int 1 2 2 3 3 3 5 5 5 5 ...  
 $ tax : num 296 242 242 222 222 222 311 311 311 311 ...  
 $ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
 $ black : num 397 397 393 395 397 ...  
 $ lstat : num 4.98 9.14 4.03 2.94 5.33 ...  
 $ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

boston |> dim() # tamanho da tabela

[1] 506 14

boston |> is.na() |> sum()

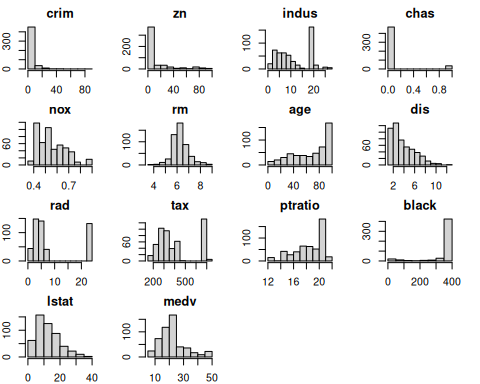
[1] 0

boston |> summary()

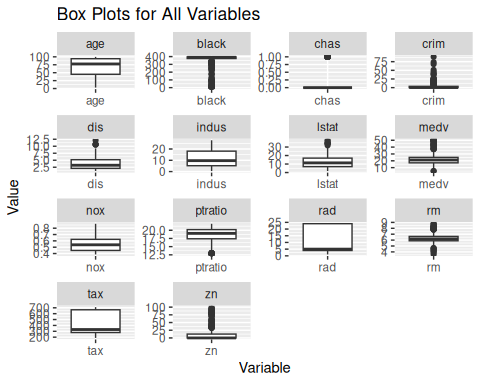
crim zn indus chas   
 Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
 1st Qu.: 0.08205 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
 Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
 Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
 Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
 nox rm age dis   
 Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
 Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
 Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
 Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
 rad tax ptratio black   
 Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 0.32   
 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.:375.38   
 Median : 5.000 Median :330.0 Median :19.05 Median :391.44   
 Mean : 9.549 Mean :408.2 Mean :18.46 Mean :356.67   
 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:396.23   
 Max. :24.000 Max. :711.0 Max. :22.00 Max. :396.90   
 lstat medv   
 Min. : 1.73 Min. : 5.00   
 1st Qu.: 6.95 1st Qu.:17.02   
 Median :11.36 Median :21.20   
 Mean :12.65 Mean :22.53   
 3rd Qu.:16.95 3rd Qu.:25.00   
 Max. :37.97 Max. :50.00

### 3.2.2 Visualizações

par(mfrow=c(4,4), mar=c(2,2,2,2))  
for (i in 1:14) {  
 hist(boston[,i], main=colnames(boston)[i])  
}

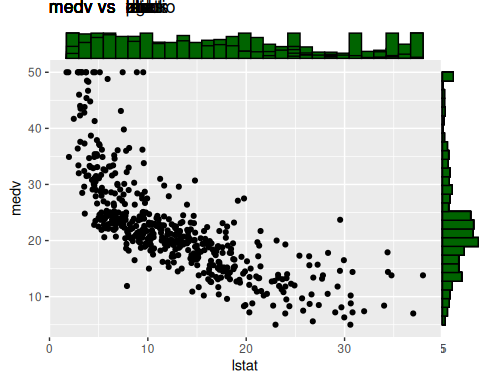


ggplot(tidyr::gather(boston), aes(x = key, y = value)) +  
 geom\_boxplot() +  
 facet\_wrap(~ key, scales = "free") +  
 labs(x = "Variable", y = "Value", title = "Box Plots for All Variables")



Muitas variáveis não se comportam como uma normal. Talvez seja nescessário alguma transformação. Há a presença de outliers, tatno em variáveis explicativas quanto na variável alvo (espera-se que modelos como Random Forest lidem bem com esse aspecto)

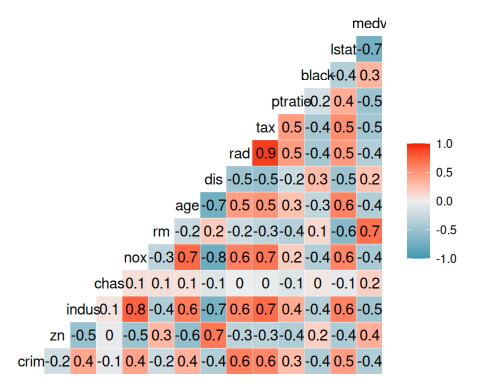
for(col in names(subset(boston,select=-c(medv)))){  
 p <- ggplot(boston, aes(y=medv, x=!!sym(col))) +  
 geom\_point() +  
 #geom\_smooth(method=lm , color="red", se=FALSE) +  
 labs(title = paste("medv vs ",col))+  
 theme(legend.position="none")   
   
   
 p1 <- ggMarginal(p, type="histogram", size=10,fill = "darkgreen")   
 print(p1)   
}



Podemos tirar as seguintes observações: \* A proximidade em relação ao “Charles River” (*chas*) não parece ter uma relação clara com a variável *medev*. Como vimos no conjunto anterior de gráficos, sua distribuição não é simétrica. Apenas uma minoria das casas estão perto do rio Charles, e mesmo dentro dessa amostra não parece haver uma relação clara no valor do imóvel. Seria interessante descartá-la do modelo para não gerar ruído

* As variáveis *age*,*rm*,*ptratio* parecem seguir uma linearidade com a variável alvo. Sendo candidatas a bons preditores
* Variáveis como *dis*, *lstat*,*tax* possuem uma relação com a variável alvo , mas que não é linear. Teoricamente, modelos de árvore consegue lidar bem com tais situações. Uma métrica de correlação tradicional (e.g pearson) talvez não capture tão bem esse aspecto

ggcorr(boston, method = c("everything", "pearson"),label = TRUE)



* *nox* e *age* são altamente positivamente correlacionadas (provalvemente regiões mais antigas eram antigos parques industriais), assim como *rad* e *tax*
* *nox* e *dis* são negativamente correlacionadas. Quanto mais longe dos centros de atividade econômica, menor será os níveis de óxido de nitrogênio (poluente), dado a menor presença de industrias ou automóveis

## 3.3 Modelagem

Obs: não retiramos ou trnasofrmamos as variáveis , veremos como o modelo reage

### 3.3.1 Separando entre treino e teste

boston\_trainIndex <- createDataPartition(boston$medv, p = 0.80, list = FALSE)  
  
training\_boston <- boston[boston\_trainIndex, ]   
  
test\_boston <- boston[-boston\_trainIndex, ]   
  
print("Train dimentions: ")

[1] "Train dimentions: "

dim(training\_boston)

[1] 407 14

print("Test dimentions: ")

[1] "Test dimentions: "

dim(test\_boston)

[1] 99 14

head(training\_boston)

crim zn indus chas nox rm age dis rad tax ptratio black lstat  
2 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14  
3 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03  
4 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94  
5 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33  
6 0.02985 0.0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 394.12 5.21  
7 0.08829 12.5 7.87 0 0.524 6.012 66.6 5.5605 5 311 15.2 395.60 12.43  
 medv  
2 21.6  
3 34.7  
4 33.4  
5 36.2  
6 28.7  
7 22.9

head(test\_boston)

crim zn indus chas nox rm age dis rad tax ptratio black  
1 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90  
9 0.21124 12.5 7.87 0 0.524 5.631 100.0 6.0821 5 311 15.2 386.63  
15 0.63796 0.0 8.14 0 0.538 6.096 84.5 4.4619 4 307 21.0 380.02  
19 0.80271 0.0 8.14 0 0.538 5.456 36.6 3.7965 4 307 21.0 288.99  
26 0.84054 0.0 8.14 0 0.538 5.599 85.7 4.4546 4 307 21.0 303.42  
33 1.38799 0.0 8.14 0 0.538 5.950 82.0 3.9900 4 307 21.0 232.60  
 lstat medv  
1 4.98 24.0  
9 29.93 16.5  
15 10.26 18.2  
19 11.69 20.2  
26 16.51 13.9  
33 27.71 13.2

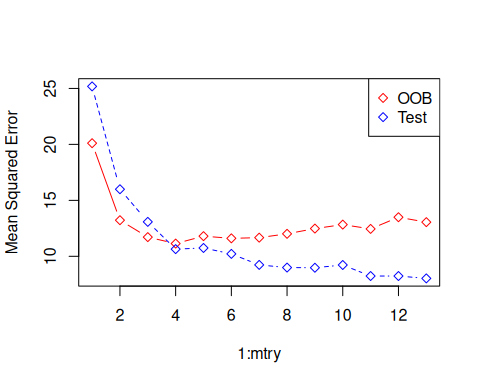
### 3.3.2 Obtendo FLoresta aleatória

rf.boston = randomForest::randomForest(medv~., data = training\_boston)  
rf.boston

Call:  
 randomForest(formula = medv ~ ., data = training\_boston)   
 Type of random forest: regression  
 Number of trees: 500  
No. of variables tried at each split: 4  
  
 Mean of squared residuals: 11.55371  
 % Var explained: 85.85

set.seed(42)  
  
oob.err = double(13)  
test.err = double(13)  
for(mtry in 1:13){  
 fit = randomForest(medv~., data = training\_boston, mtry=mtry,  
ntree = 350)  
 oob.err[mtry] = fit$mse[350]  
 pred = predict(fit, test\_boston)  
 test.err[mtry] = with(test\_boston, mean( (medv-pred)^2 ))  
}

matplot(1:mtry, cbind(oob.err,test.err), pch = 23, col = c("red",  
"blue"), type = "b", ylab="Mean Squared Error")  
  
  
  
legend("topright", legend = c("OOB","Test"), pch = 23, col = c("red",  
"blue"))



erros\_rf <- data.frame(  
 mtry = 1:mtry,  
 mse\_oob = oob.err,  
 mse\_test = test.err  
   
)  
  
 mtry\_minOOB <-erros\_rf |>  
 dplyr::filter(mse\_oob == min(mse\_oob)) |>  
 dplyr::pull(mtry)  
  
 mtry\_minTest <- erros\_rf |>  
 dplyr::filter(mse\_test == min(mse\_test)) |>  
 dplyr::pull(mtry)  
  
   
print(paste("Número de preditores que resulta no MSE Mínimo no OOB: ", mtry\_minOOB))

[1] "Número de preditores que resulta no MSE Mínimo no OOB: 4"

print(paste("Número de preditores que resulta no MSE Mínimo no Test: ", mtry\_minTest))

[1] "Número de preditores que resulta no MSE Mínimo no Test: 13"

Há um espaçamento grande entre OOB e Teste indica a capacidade preditiva do modelo. Um modelo generalizador retornmaria linhas quase sobrepostas. Não foi isso que vimos, mas não sabemos julgar o quão ruim foi o espaçamento que obtivemos.

predict\_test <- predict(rf.boston,test\_boston,type='class')   
  
predict\_train <- predict(rf.boston,training\_boston,type='class')  
  
  
KPIS\_regression\_train <- data.frame(  
 mae = caret::MAE(training\_boston$medv,predict\_train),  
 rmse = caret::RMSE(training\_boston$medv,predict\_train)  
)  
KPIS\_regression\_train

mae rmse  
1 1.001232 1.520773

KPIS\_regression\_test <- data.frame(  
 mae = caret::MAE(test\_boston$medv,predict\_test),  
 rmse = caret::RMSE(test\_boston$medv,predict\_test)  
)  
  
KPIS\_regression\_test

mae rmse  
1 2.247456 3.543298

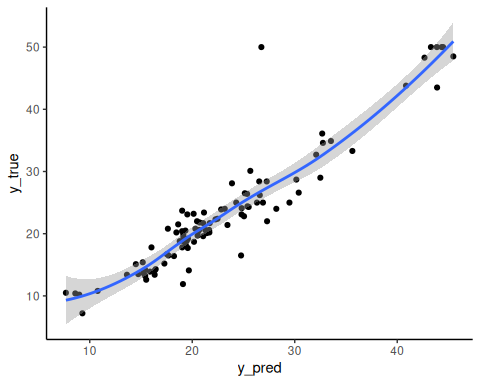
Computando métricas como MAE (Mean absolute error) e RMSE (Root mean squared error), há certa estabilidade no treino e teste

comparativo\_teste <- data.frame(y\_pred = predict\_test,  
 y\_true=test\_boston$medv)   
  
  
comparativo\_teste

y\_pred y\_true  
1 28.201862 24.0  
9 17.683856 16.5  
15 19.373667 18.2  
19 18.452933 20.2  
26 15.878485 13.9  
33 15.409832 13.2  
35 14.732325 13.5  
42 30.388272 26.6  
50 19.142043 19.4  
51 20.520920 19.7  
53 26.899800 25.0  
54 21.146790 23.4  
55 19.468757 18.9  
60 21.064757 19.6  
61 19.231960 18.7  
64 26.311803 25.0  
68 20.466557 22.0  
72 21.705123 21.7  
90 30.154667 28.7  
94 24.290541 25.0  
96 26.535410 28.4  
97 23.436140 21.4  
99 40.856573 43.8  
102 25.128680 26.5  
119 19.008267 20.4  
122 20.563637 20.3  
123 19.335673 20.5  
125 18.778470 18.8  
131 19.583599 19.2  
140 16.022080 17.8  
141 15.229614 14.0  
150 15.170887 15.4  
151 18.627756 21.5  
159 25.508156 24.3  
162 43.288670 50.0  
164 44.474537 50.0  
167 44.345083 50.0  
170 22.268292 22.3  
185 25.354812 26.4  
190 33.533373 34.9  
197 35.623157 33.3  
199 32.753190 34.6  
202 24.822693 24.1  
204 45.472597 48.5  
214 23.876127 28.1  
215 19.015793 23.7  
222 21.064992 21.7  
224 25.665812 30.1  
234 42.661133 48.3  
236 23.171902 24.0  
241 27.300473 22.0  
242 21.413337 20.1  
248 21.648179 20.5  
250 26.591220 26.2  
266 25.053143 22.8  
269 43.889480 43.5  
273 25.385597 24.4  
284 43.874200 50.0  
294 22.820078 23.9  
300 32.497927 29.0  
305 32.680737 36.1  
306 27.266840 28.4  
318 19.129343 19.8  
339 20.677503 20.6  
341 20.172667 18.7  
342 32.093720 32.7  
343 24.764975 16.5  
348 24.810977 23.1  
356 21.644453 20.6  
363 20.304191 20.8  
368 19.519693 23.1  
373 26.744902 50.0  
387 7.673899 10.5  
389 8.977009 10.2  
391 14.498826 15.1  
407 19.095438 11.9  
416 9.283283 7.2  
418 8.589464 10.4  
422 15.409903 14.2  
423 17.623411 20.8  
424 13.628886 13.4  
432 19.668836 14.1  
434 16.435035 14.3  
445 10.773815 10.8  
448 15.507756 12.6  
451 16.333275 13.4  
452 17.282701 15.2  
454 19.024465 17.8  
456 16.344177 14.1  
461 18.218204 16.4  
462 19.565066 17.7  
464 21.672287 20.2  
473 20.136349 23.2  
477 17.521963 16.7  
483 29.482914 25.0  
484 20.725645 21.8  
488 21.274605 20.6  
492 15.278881 13.6  
502 22.468547 22.4

comparativo\_teste |>   
 ggplot(aes(x=y\_pred,y=y\_true)) +   
 geom\_point() +   
 geom\_smooth() +   
 theme\_classic()

`geom\_smooth()` using method = 'loess' and formula = 'y ~ x'



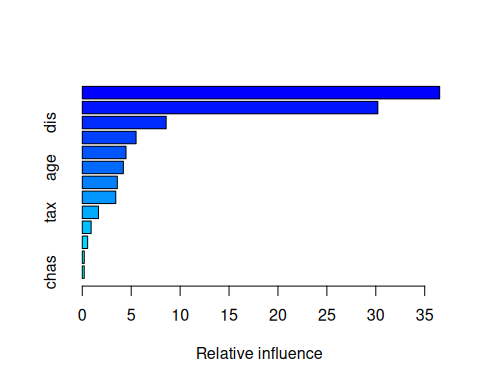
## 3.4 Bosting

require(gbm)

Loading required package: gbm

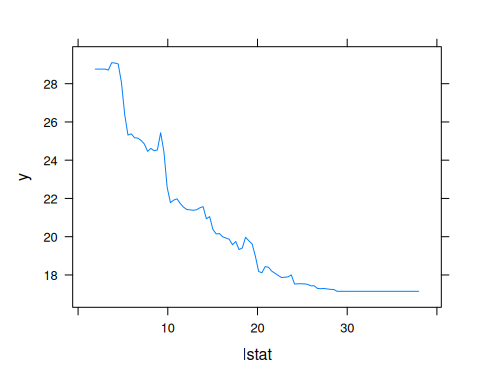
Loaded gbm 2.1.8.1

library(gbm)  
  
  
boost.boston = gbm(medv~., data = training\_boston, distribution =  
"gaussian", n.trees = 10000, shrinkage = 0.01, interaction.depth = 4)  
summary(boost.boston)

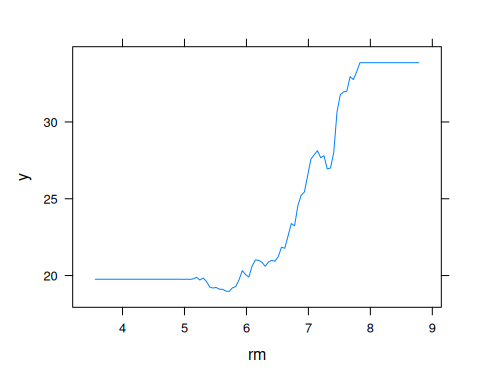


var rel.inf  
lstat lstat 36.5302064  
rm rm 30.2110808  
dis dis 8.5749440  
crim crim 5.4988901  
nox nox 4.4685884  
age age 4.2005767  
black black 3.5841535  
ptratio ptratio 3.4222336  
tax tax 1.6563053  
indus indus 0.9054813  
rad rad 0.5465131  
zn zn 0.2023577  
chas chas 0.1986690

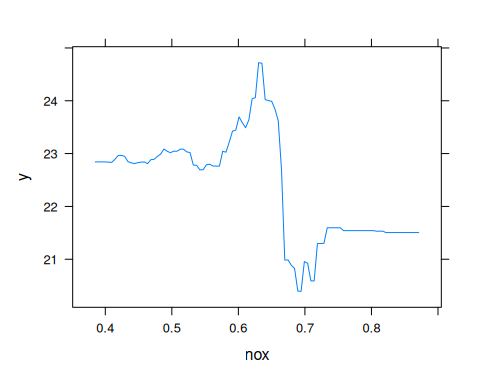
plot(boost.boston,i="lstat")



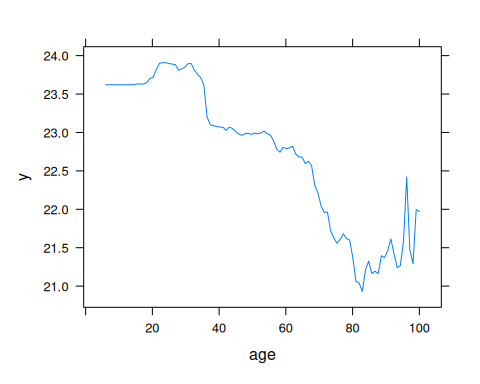
plot(boost.boston,i="rm")



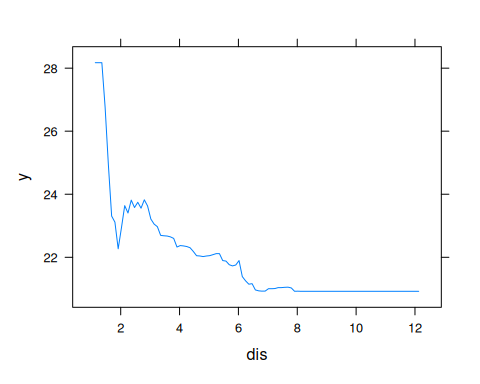
plot(boost.boston,i="nox")



plot(boost.boston,i="age")



plot(boost.boston,i="dis")

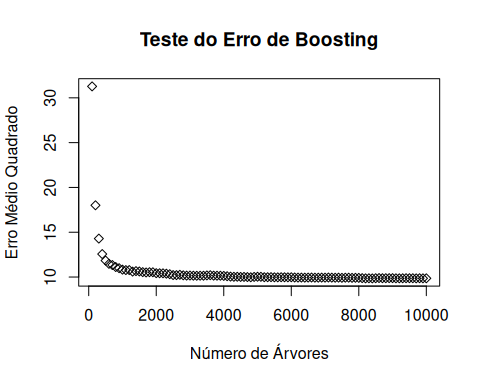


* quanto maior a proporção de pessoas de status inferior no bairro, menor o valor dos preços das habitações.
* quanto maior o número de quartos, maior o preço das habitações
* quanto maior a idade, menor o valor dos preços das habitações.
* quanto maior a distância, menor o valor dos preços das habitações.
* o nível de óxido de nitrogênio não aprece ter uma relação tão clara com o preço (picos no meio). Olhando a tendência da reta, parece que quanto maior a poluição, menor o preço. o

n.trees = seq(from = 100, to = 10000, by = 100)  
predmat = predict(boost.boston, newdata = test\_boston, n.trees =  
n.trees)  
dim(predmat)

[1] 99 100

boost.err = with(test\_boston, apply( (predmat - medv)^2, 2, mean))  
plot(n.trees, boost.err, pch = 23, ylab = "Erro Médio Quadrado", xlab  
= "Número de Árvores", main = "Teste do Erro de Boosting")



O erro se estabiliza próximo à 2000

## 3.5 Conclusão

A análise desses dados permite com que corretores ou investidores tomem boas decisões de negócio no setor imobiliário ao se atentarem a atributos como e o número de quartos , a taxa de criminalidade, distância aos centros econômicos. Esses agentes também podem utilizar os modelos de árvores para realizarem a precificação do imóvel, dado que modelos de florestas aleatórias demosntraram um baixo erro médio. Eles também podem deixar de focar em atributos irrelevantes como a ndistância ao rio Charles

Gestores públicos serão benefíciados pela análise de variáveis trazida pelo algorítimo de boosting. Por meio dela , pode-se perceber que problemáticas socieconômicas como segregação racial, criminalidade , poluição e presença de população de baixo status (desigualdade) afetam o preço dos imóveis (reflexo de todas esses problemas). Outras conclusões podem ser derivadas olhando o comportamento entre as variáveis explicativas