Adriana Eva Fernandes da Silva 761886

```
set.seed(1)
library(readr)
library(conflicted)
library(data.table)
library(tm)
library(glmnet)
library(tidyverse)
library(dplyr)

dados <- read_csv("~/mineração de dados/atividade-1-precificacao-veiculos.csv")
```

Item A:

```
#transformando em dummies:
df_encoded <- model.matrix(~ . - 1, data = dados)

dtm.matrix <-df_encoded %>% as.matrix()
dtm.matrix <- dtm.matrix[, -c(1, ncol(dtm.matrix))]
dim(dtm.matrix)
```

```
## [1] 205 190
```

Podemos notar que, filtrando os termos da matriz de dados, teremos um total de 190 termos.

Item B:

Vamos dividir os dados em 60% em treinamento e 40% em teste para avaliar o desempenho do modelo em dados não vistos.

```
set.seed(1)
split<- sample(c("Treinamento","Teste"),prob=c(0.6,0.4),size = nrow(dados),replace = TRUE)
```

Item C:

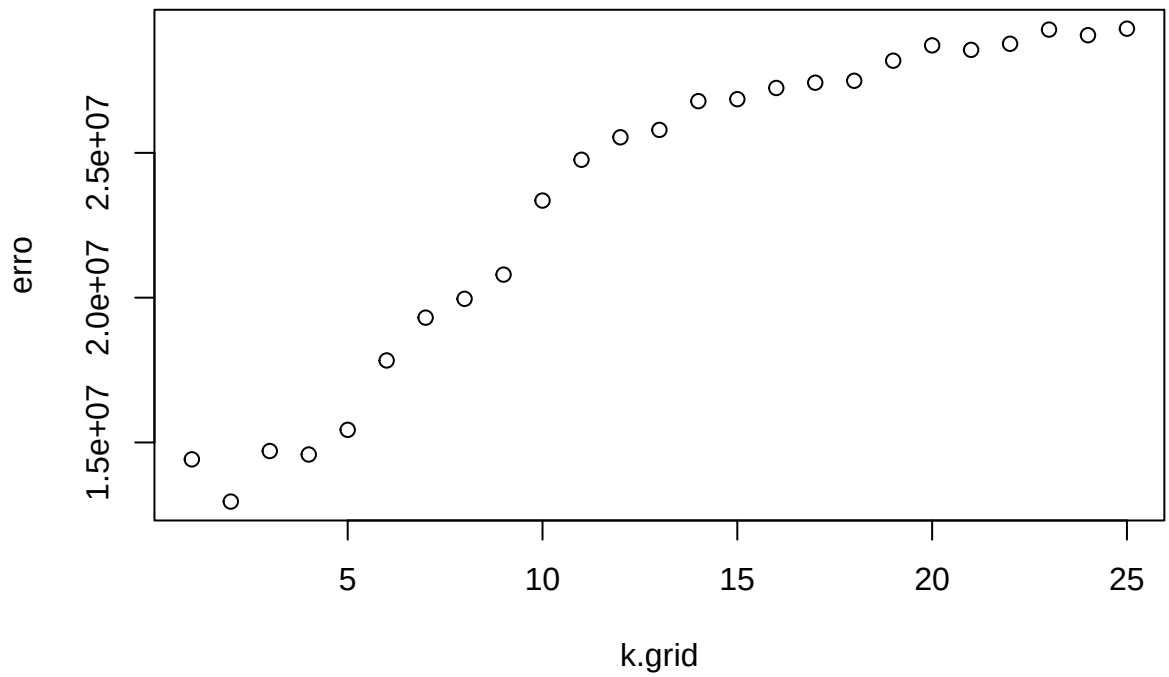
Realizando uma análise de regressão usando o método k-vizinhos mais próximos (KNN).

Com o objetivo de alcançar esse propósito, será gerado um vetor contendo múltiplos valores para a variável "k".

```
library(FNN)#pacote usado para o KNN
k.grid<- round(seq(1,25,length.out=25))


erro<- rep(NA,length(k.grid))#crio um vetor de erros para cada k do grid
for (ii in seq_along(k.grid))
{
  predito <- knn.reg(train = dtm.matrix[split=="Treinamento",],
                     y=dados$price[split=="Treinamento"],
                     k=k.grid[ii])$pred#

  erro[ii]<- mean((predito-dados$price[split== "Treinamento"])^2)  #estimando o risco
}
```

A seguir, apresenta-se a plotagem do gráfico para visualizar qual seria o melhor k:

```
plot(k.grid,erro)
```



```r
best.k<- k.grid [which.min(erro)]
best.k
```

```
## [1] 2
```

Podemos observar que o valor de "k" que resulta no menor erro é k=2.

Realizando o cálculo da predição no conjunto de teste:

```r
predito_knn<- knn.reg(train=dtm.matrix[split=="Treinamento",],
                  test = dtm.matrix[split=="Teste",],
                  y=dados$price[split=="Treinamento"],
                  k=best.k)$pred
```

Foram calculadas as predições para k=2 vizinhos mais próximos.

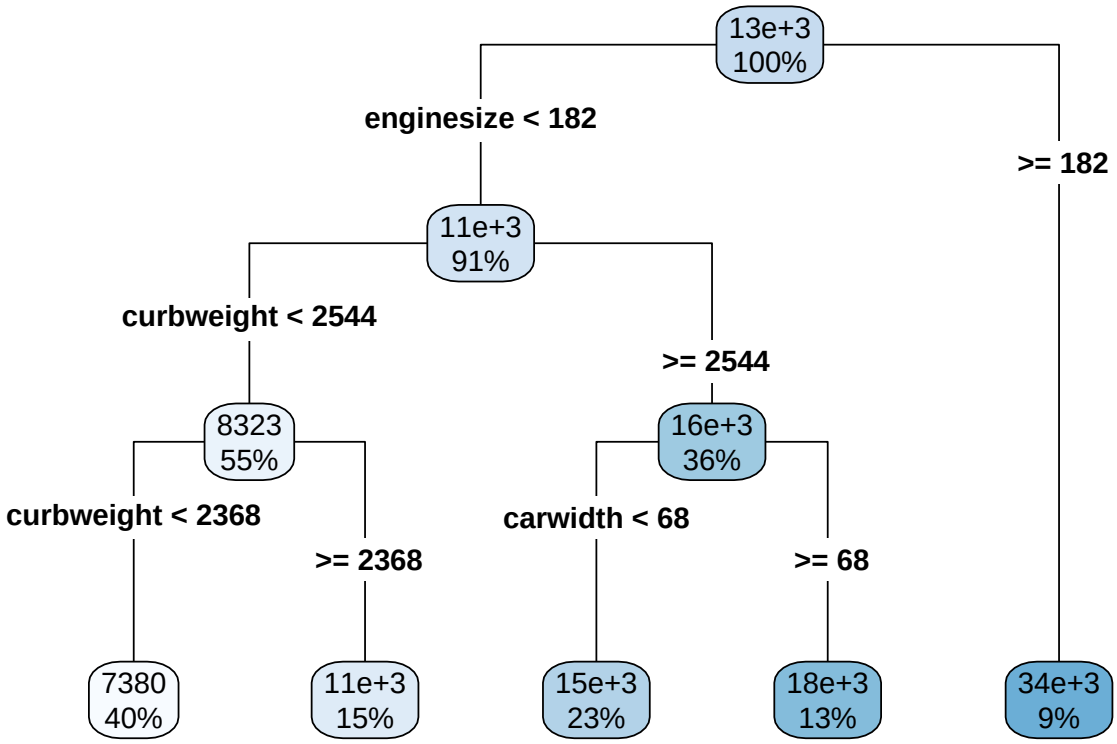
```r
riscos_knn<- (predito_knn-dados$price[split=="Teste"])^2 %>% mean()
riscos_knn
```

```
## [1] 12014891
```

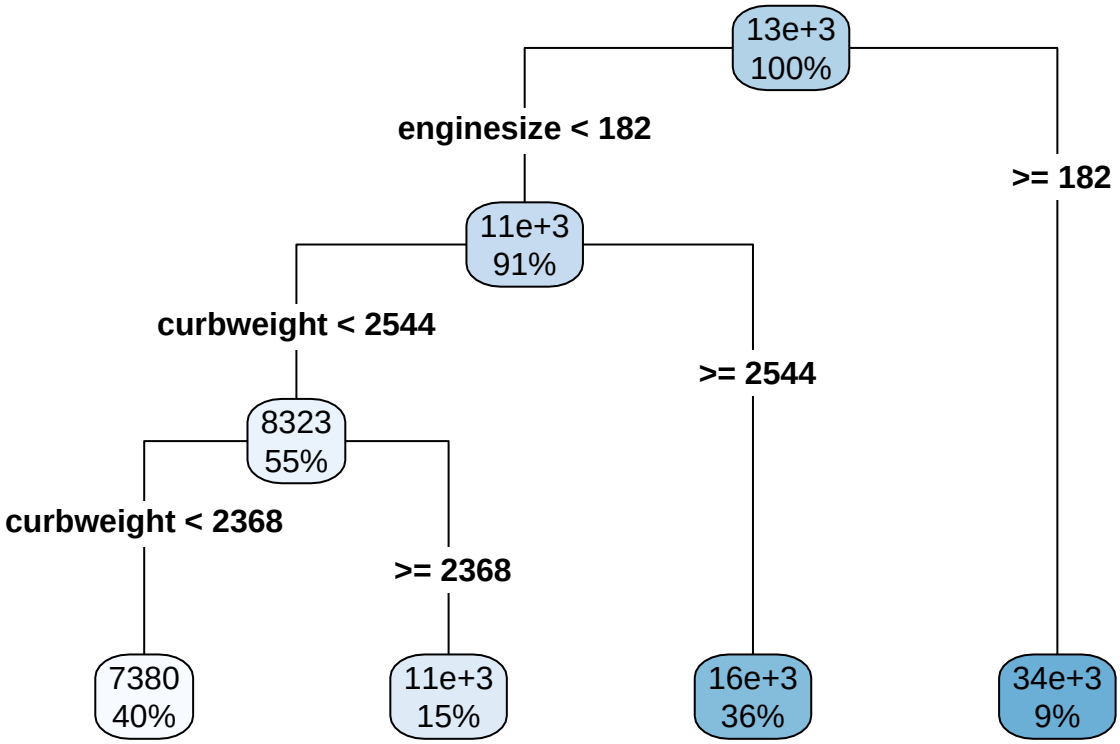
No método KNN cada vez que desejamos fazer uma previsão para um novo ponto, é necessário consultar novamente os dados de treinamento para encontrar os vizinhos mais próximos. Assim, não há um ajuste fixo, que é diferente, por exemplo do método de Mínimos quadrados e isso pode tornar o KNN mais demorado, especialmente em conjuntos de dados grandes. À medida que o valor de "k" aumenta, o tempo de processamento também aumenta, devido à necessidade de considerar mais vizinhos. Ao contrário dos métodos paramétricos, no KNN, não envolvemos coeficientes interpretáveis.

Ajustando uma regressão usando o método Árvores de Regressão:

```r
library(rpart)
library(rpart.plot)
dados.tudo<-dtm.matrix %>% as.data.frame %>% mutate(Price=dados$price)
arvore= rpart(Price ~.,data = dados.tudo[split== "Treinamento",])
rpart.plot(arvore,type=4)
```




```
#poda:

melhorCp = arvore$cptable[which.min(arvore$cptable[,"xerror"]),"CP"]

poda = prune(arvore,cp=melhorCp)

rpart.plot(poda,type=4)
```




```
predito_arvore = predict(poda,dados.tudo[split=="Teste",])
riscos_arvore<- (predito_arvore-dados.tudo$Price[split=="Teste"])^2 %>% mean()
riscos_arvore
```

```
## [1] 8011682
```

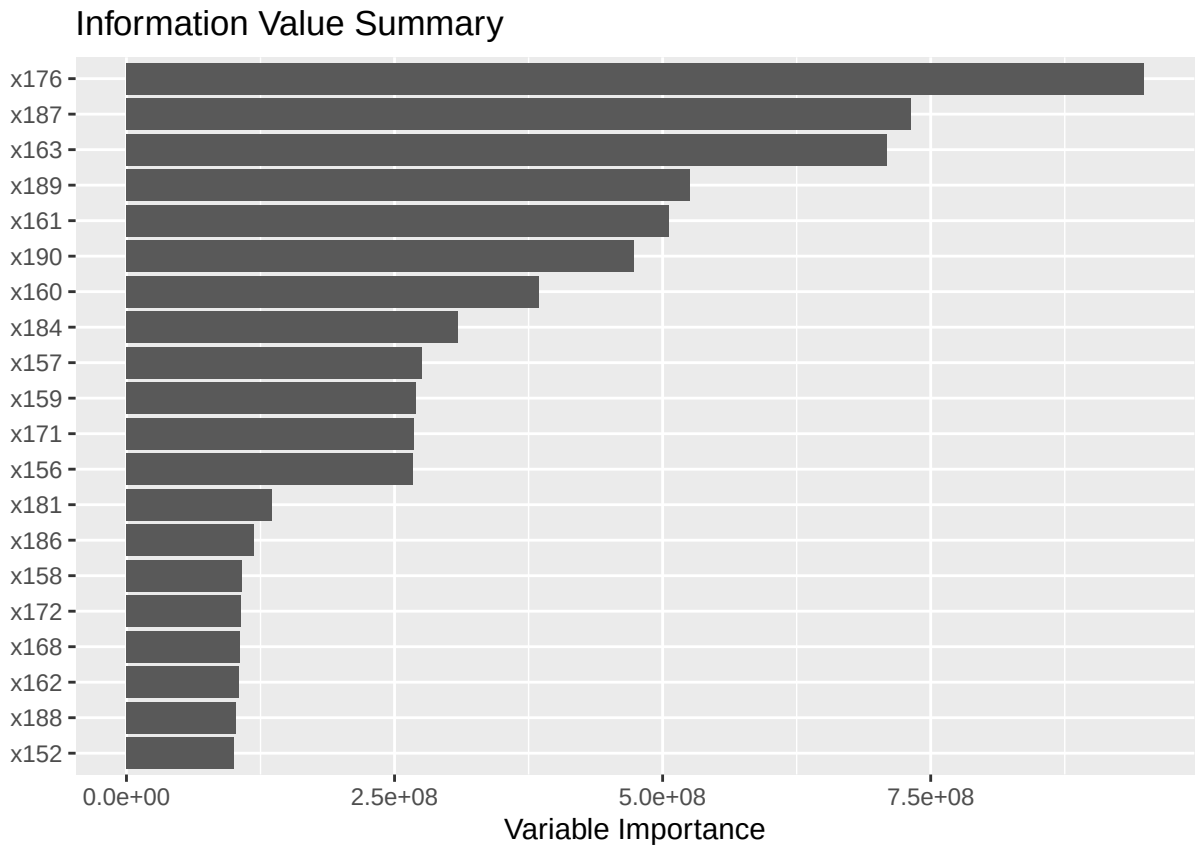
As árvores facilitam a inclusão de variáveis categóricas, realizam seleção automática de variáveis e consideram interações automaticamente. A construção de uma árvore requer uma medida de pureza baseada na soma dos erros quadráticos. Ao comparar diferentes árvores, menor pureza significa maior homogeneidade. Embora árvores maiores possam explicar qualquer conjunto de dados, sua capacidade preditiva pode ser limitada. A poda é usada após a criação de uma árvore ampla. Usando validação cruzada, determinamos o tamanho da árvore que minimiza o risco estimado.

Interpretando a árvore:

As variáveis enginesize, curbweight e carwidth têm impacto significativo nas previsões de preço dos veículos de acordo com a árvore de regressão. Com base no primeiro nó a previsão média do preço do veiculo se a variavel enginesize < 182 é de 11355.660, e se enginesize ≥ 182 a previsão média é de 34478.050. No segundo nó se curbweight <2544 a previsão média do preço do veículo é de 8322.828, se curbweight ≥ 2544 a previsão média é de 15973.840. Se curbweight <2368 o preço médio será de 7379.745, se curbweight ≥ 2368 o preço médio será 10890.110. Agora se carwidth <68 a previsão média será de 14736.210, caso contrário será de 18139.700

Ajustando uma regressão via Florestas Aleatórias:

```
library(ranger)
# Criar um vetor com 189 elementos contendo os nomes "x1", "x2", ..., "x190"
vetor_nomes <- paste0("x", 1:190)
novos_nomes<-c(vetor_nomes,"Price")
colnames(dados.tudo) <- novos_nomes

#floresta aleatoria
floresta = ranger(Price~.,data = dados.tudo[split=="Treinamento",],importance = "impurity")
predito_floresta = predict(floresta,dados.tudo[split=="Teste",])

importances<- tibble(variable=names(importance(floresta)),importance=importance(floresta)) %>%
  arrange(desc(importance))

ggplot(importances %>% top_n(n=20),
       aes(x=reorder(variable,importance),y=importance))+
  geom_bar(stat="identity",position="dodge")+ coord_flip()+
  ylab("Variable Importance")+
  xlab("")+
  ggtitle("Information Value Summary")+
  guides(fill=F)+
  scale_fill_gradient(low="red",high = "blue")
```



```r
#calculando os riscos
riscos_florestas<- (predito_floresta$predictions - dados.tudo$Price[split=="Teste"])^2 %>% mean()
riscos_florestas
```

```
## [1] 5888553
```

Esse gráfico traz a medida de importância de cada variável. A variável x176= "enginesize" tem a maior importância, seguida por x187 = "horsepower" e x163 = "curbweight", indicando que essas três variáveis tem uma contribuição mais forte na previsão do preço dos veículos.

Ajustando uma regressão via Nadaraya-Watson:

```r
library(fields)

x_treinamento <- dtm.matrix[split=="Treinamento",]
x_teste <- dtm.matrix[split=="Teste",]
# Calcule a matriz de distâncias euclidianas entre as linhas de x
dist_matrix <- rdist(x_treinamento)

y_treinamento <- dados$price[split=="Treinamento"]
y_teste <- dados$price[split=="Teste"]

banda_i<- round(seq(0.1,2,length.out=20),1)
cv_errors <- numeric(length(banda_i))


for (ii in seq_along(banda_i)){
  bandwidth <- banda_i[ii]


   # usando o kernel gaussiano
  weights <- (1 / sqrt(2 * pi * (bandwidth)^2)) * exp(-dist_matrix^2 / (2 * bandwidth^2))

    # estimativas usando o método Nadaraya-Watson
  nadaraya_watson_estimates <- rowSums(weights * y_treinamento)
  cv_errors[ii] <- mean((y_treinamento - nadaraya_watson_estimates)^2)

}


#########################################################################

# o valor de banda que resulta no menor erro de validação cruzada
best_bandwidth <- banda_i[which.min(cv_errors)]

print(paste("Melhor valor de banda:", best_bandwidth))
```

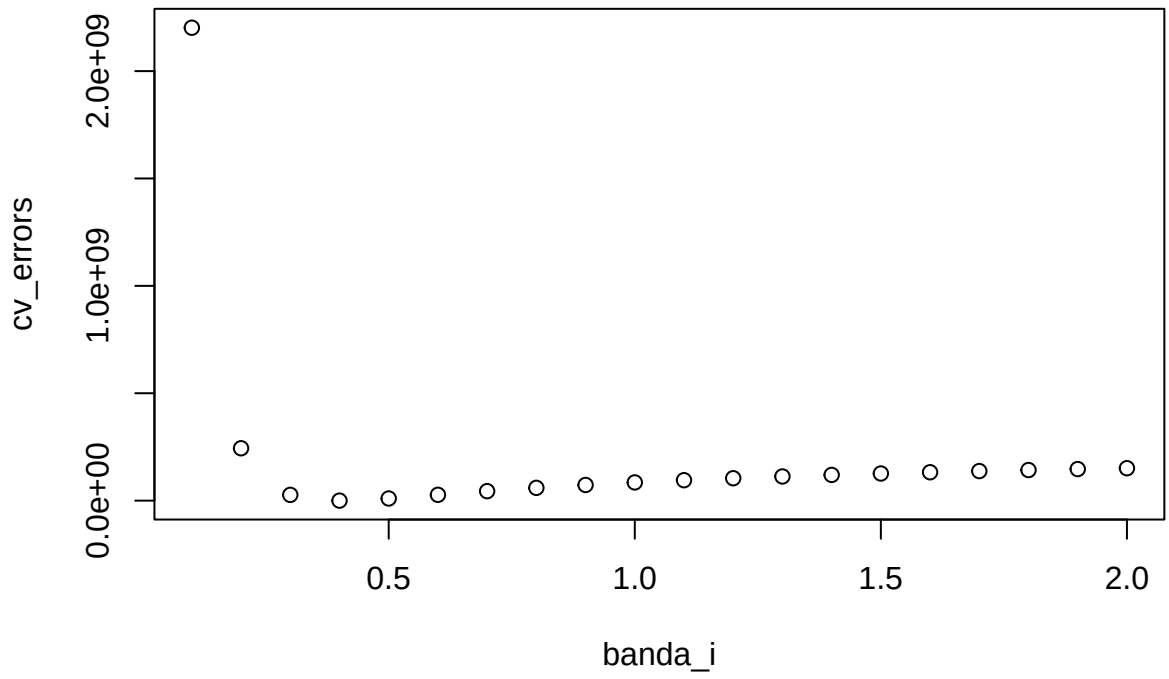
```
## [1] "Melhor valor de banda: 0.4"
```

```r
dist_matrix <- rdist(x_treinamento)

# Calculando os pesos usando o kernel gaussiano
weights_teste <- (1 / sqrt(2 * pi * (best_bandwidth)^2)) * exp(-dist_matrix^2 / (2 * best_bandwidth^2))

# Calculando as estimativas usando o método Nadaraya-Watson
predito_nw_teste<- rowSums(weights_teste * y_teste)

    # Calcule o erro de validação cruzada para o valor de banda atual
riscos_nw <- mean((y_teste - predito_nw_teste)^2)
riscos_nw
```

```
## [1] 176282734
```

Plotando o gráfico:

```r
plot(banda_i,cv_errors)
```



Item D:

Conclusão dos riscos:

```r
library(knitr)

dados_tabela <- data.frame(metodos = c("Knn", "Lasso", "Mínimos Quadrados", "Nadaraya Watson", "Árvore d
                           riscos = c(12014891, 10248768, 10238423, 176282734, 8011682,5888553))

kable(dados_tabela, format = "simple")
```

metodos	riscos
Knn	12014891
Lasso	10248768
Mínimos Quadrados	10238423
Nadaraya Watson	176282734
Árvore de Regressão	8011682
Floresta Aleatória	5888553

Ao realizar a comparação dos riscos entre os modelos utilizados, podemos concluir que Floresta Aleatória e Árvore de Regressão parecem ser os métodos mais robustos e precisos para prever os preços dos veículos. Observa-se que a Árvore apresentou o desempenho inferior em relação ao método Floresta, isso se deve ao fato de que árvores são consideradas modelos relativamente simples. Devido a essa simplicidade seu poder preditivo foi menor.

Intervalo de confiança para o risco estimado do melhor modelo encontrado para KNN:

```r
n_teste <- sum(split == "Teste")
residuos <- predito_knn - dados$price[split == "Teste"]
desvio_padrao <- sqrt(sum(residuos^2) / (n_teste))

z_critico <- qnorm(0.975)
erro_padrao <- desvio_padrao / sqrt(n_teste)
intervalo_confianca_knn <- c(riscos_knn - z_critico * erro_padrao, riscos_knn + z_critico * erro_padrao)
intervalo_confianca_knn
```

```
## [1] 12014145 12015637
```

Intervalo de confiança para o risco estimado do melhor modelo encontrado para Nadaraya Watson:

```r
n_teste <- sum(split == "Teste")
residuos <- predito_nw_teste - dados$price[split == "Teste"]
desvio_padrao <- sqrt(sum(residuos^2) / (n_teste))

z_critico <- qnorm(0.975)
erro_padrao <- desvio_padrao / sqrt(n_teste)
intervalo_confianca_nw <- c(riscos_nw - z_critico * erro_padrao, riscos_nw + z_critico * erro_padrao)
intervalo_confianca_nw
```

```
## [1] 176279271 176286197
```

Item E:

Target Encoding:

No método de "target encoding", as categorias são convertidas em valores numéricos. Para cada categoria exclusiva na variável categórica, calculamos uma estatística resumida dos valores alvo correspondentes, como média ou mediana.

```
library(dataPreparation)
dados_sem_primeira_coluna <- subset(dados, select = -1)


target_encodings <- build_target_encoding(
  data_set = dados_sem_primeira_coluna,
  cols_to_encode = c("symboling", "name", "fueltypes" ,"aspiration" , "doornumbers","carbody" ,"drivewhe
  target_col = "price")
```

```
## [1] "build_target_encoding: Start to compute encoding for target_encoding according to col: price."
```

```
library(dplyr)

categoria_valor1 <- c("3" = 17221.30,
                      "2" = 10109.28,
                      "1" = 10037.91,
                      "0" = 14366.97,
                      "-1" = 17330.68,
                      "-2" = 15781.67)

dados$symboling <- categoria_valor1[as.character(dados$symboling)]

################################################################################

categoria_valor2<- c("alfa-romero giulia" = 13495,           "alfa-romero stelvio" = 16500,  "alfa-rom
  "audi 100 ls" = 13950,                  "audi 100ls" = 17580,                    "audi fox" = 15250,
  "audi 5000" = 18920,                      "audi 4000" = 23875,             "audi 5000s (diesel)" =
  "bmw 320i" = 16677.5,                    "bmw x1" = 20970,                        "bmw x3" = 28992
 "bmw z4" = 24565,                         "bmw x4" = 30760,                       "bmw x5" = 41315,
 "dodge rampage" = 5572,                   "dodge challenger se" = 6377,         "dodge d200" = 7957,
 "dodge monaco (sw)" = 6229,             "dodge colt hardtop" = 6692,              "dodge colt (sw)" =
 "dodge coronet custom" = 8558,            "dodge dart custom" = 8921,
 "dodge coronet custom (sw)" = 12964,      "honda civic" = 8274.333,           "honda civic cvcc" = 6
 "honda accord cvcc" = 6529,              "honda accord lx" = 7295,            "honda civic 1500 gl" =
  "honda accord" = 9095,                    "honda civic 1300" = 9095,          "honda prelude" = 884
 "honda civic (auto)" = 10345,            "isuzu MU-X" = 6785,                    "isuzu D-Max"  = 9
  "isuzu D-Max V-Cross" = 8916.5,          "jaguar xj" = 32250,                   "jaguar xf" = 35550
 "mazda rx2 coupe" = 6795,                "mazda rx-4" = 8470,                   "mazda glc deluxe" =
 "mazda glc 4" = 15645,                    "mazda glc custom l" = 8495,             "mazda glc custom
"buick skyhawk" = 31600,                  "buick opel isuzu deluxe" = 34184,        "buick skylark" =
  "mitsubishi mirage" = 5389,              "mitsubishi lancer" = 6189,             "mitsubishi outla
  "mitsubishi pajero" = 8189,               "Nissan versa" = 5499,                  "nissan gt-r" =
  "nissan leaf" = 7299,                    "nissan juke" = 7799,                  "nissan note" = 79
  "nissan fuga" = 14399,                    "nissan otti" = 13499,                  "nissan teana" =
 "peugeot 504 (sw)" = 12440,               "peugeot 604sl" = 17525,                "peugeot 505s turb
```

```
 "plymouth fury gran sedan" = 7609,          "plymouth valiant" = 8921,                  "plymouth duster"
 "porsche boxter" = 37028,                  "renault 12tl" = 9295,                     "renault 5 gtl" = 9
"subaru" = 6122,                            "subaru dl" = 8590.75,                     "subaru brz" = 7775
"subaru trezia" = 7463,                     "subaru tribeca" = 10198,                  "toyota corona mark
"toyota corolla 1600 (sw)" = 7898,          "toyota carina" = 8778,                    "toyota mark ii" =
"toyota corolla tercel" = 9538,            "toyota corona liftback" = 8449,           "toyota starlet" = 7
"toyouta tercel" = 15750,                   "vokswagen rabbit" = 7775,                 "volkswagen 1131 de
"volkswagen super beetle" = 9495,           "volkswagen dasher" = 11142.5,              "vw dasher" = 1159
"vw rabbit" = 9980,                         "volkswagen rabbit" = 13295,                "volkswagen rabbit
"volvo 145e (sw)" = 14892.5,               "volvo 144ea" = 16230,                     "volvo 244dl" = 1873
"volvo 246" = 22470)

dados$name <- categoria_valor2[as.character(dados$name)]

#########################################################################
categoria_valor3<- c("gas"  = 12999.80,         "diesel" =  15838.15)
dados$fueltypes <- categoria_valor3[as.character(dados$fueltypes)]

#########################################################################
categoria_valor4<- c("std" =    12611.27,           "turbo" =   16298.17)
dados$aspiration <- categoria_valor4[as.character(dados$aspiration)]

#########################################################################
categoria_valor5<-c("two"   = 12989.92, "four" =    13501.15)
dados$doornumbers <- categoria_valor5[as.character(dados$doornumbers)]
#########################################################################

categoria_valor6<-c("convertible" =     21890.50,
                    "hatchback" =   10376.65,
                    "sedan" =   14344.27,
                    "wagon" =   12371.96,
                    "hardtop" =     22208.50    )
dados$carbody <- categoria_valor6[as.character(dados$carbody)]

#########################################################################3

categoria_valor7<- c("rwd" =    19910.809,          "fwd" =     9239.308, "4wd" = 11087.463)
dados$drivewheels <- categoria_valor7[as.character(dados$drivewheels)]

#########################################################################

categoria_valor8<- c("front" =  12961.1,            "rear" =    34528.0 )
dados$enginelocation <- categoria_valor8[as.character(dados$enginelocation)]
#########################################################################

categoria_valor9<- c("dohc" =   18116.42, "ohcv" =  25098.38,   "ohc" = 11574.05,
                     "l" =     14627.58, "rotor" =     13020.00,       "ohcf" =    13738.60, "dohcv" =
dados$enginetype <- categoria_valor9[as.character(dados$enginetype)]

#########################################################################

categoria_valor10 <- c("four" =     10285.75,
"six" =     23671.83,
```

```
"five" =    21630.47,
"three" =   5151.00,
"twelve" =  36000.00,
"two" =     13020.00,
"eight" =   37400.10)
dados$cylindernumber <- categoria_valor10[as.character(dados$cylindernumber)]

#############################################################################

categoria_valor11<- c("mpfi" =  17754.603,
"2bbl" =    7478.152,
"mfi" =     12964.000,
"1bbl" =    7555.545,
"spfi" =    11048.000,
"4bbl" =    12145.000,
"idi" =     15838.150,
"spdi" =    10990.444)
dados$fuelsystem <- categoria_valor11[as.character(dados$fuelsystem)]
```

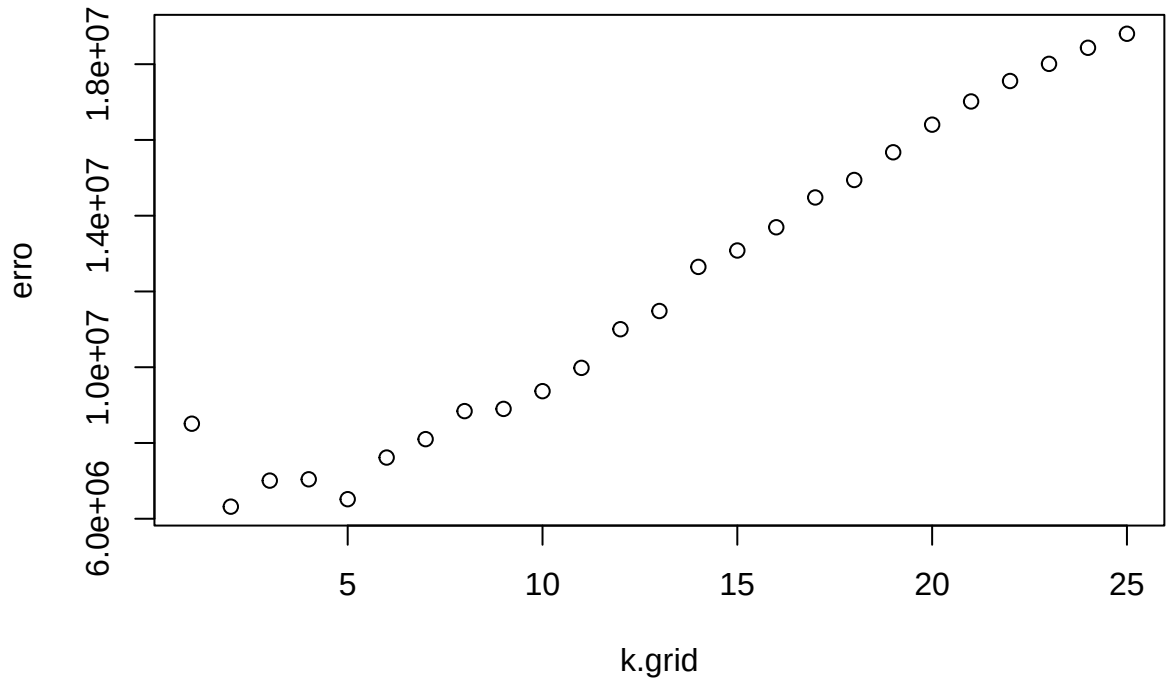
Aplicando a regressão via KNN usando o target enconding:

```
dados2 <- dados %>% as.matrix()
dados2 <- dados2[, -c(1, ncol(dados2))]
dim(dados2)
```

```
## [1] 205  24
```

```
valores_ausentes <- sum(is.na(dados2))
posicoes_ausentes <- which(is.na(dados2))
dados2 <- replace(dados2, is.na(dados2), 9982.25)
valores_ausentes <- sum(is.na(dados2))

set.seed(1)
split<- sample(c("Treinamento","Teste"),prob=c(0.6,0.4),size = nrow(dados),replace = TRUE)

library(FNN)#pacote usado para o KNN
k.grid<- round(seq(1,25,length.out=25))
valores_ausentes <- sum(is.na(dados2))
posicoes_ausentes <- which(is.na(dados2))



erro<- rep(NA,length(k.grid))#crio um vetor de erros para cada k do grid
for (ii in seq_along(k.grid))
{
  predito <- knn.reg(train = dados2[split=="Treinamento",],
                     y=dados$price[split=="Treinamento"],
                     k=k.grid[ii])$pred

  erro[ii]<- mean((predito-dados$price[split== "Treinamento"])^2)
```

```
}


plot(k.grid,erro)
```




```
best.k<- k.grid [which.min(erro)]

best.k
```

```
## [1] 2
```

```
predito_knn<- knn.reg(train=dados2[split=="Treinamento",],
                      test = dados2[split=="Teste",],
                      y=dados$price[split=="Treinamento"],
                      k=best.k)$pred

riscos_knn<- (predito_knn-dados$price[split=="Teste"])^2 %>% mean()

riscos_knn
```

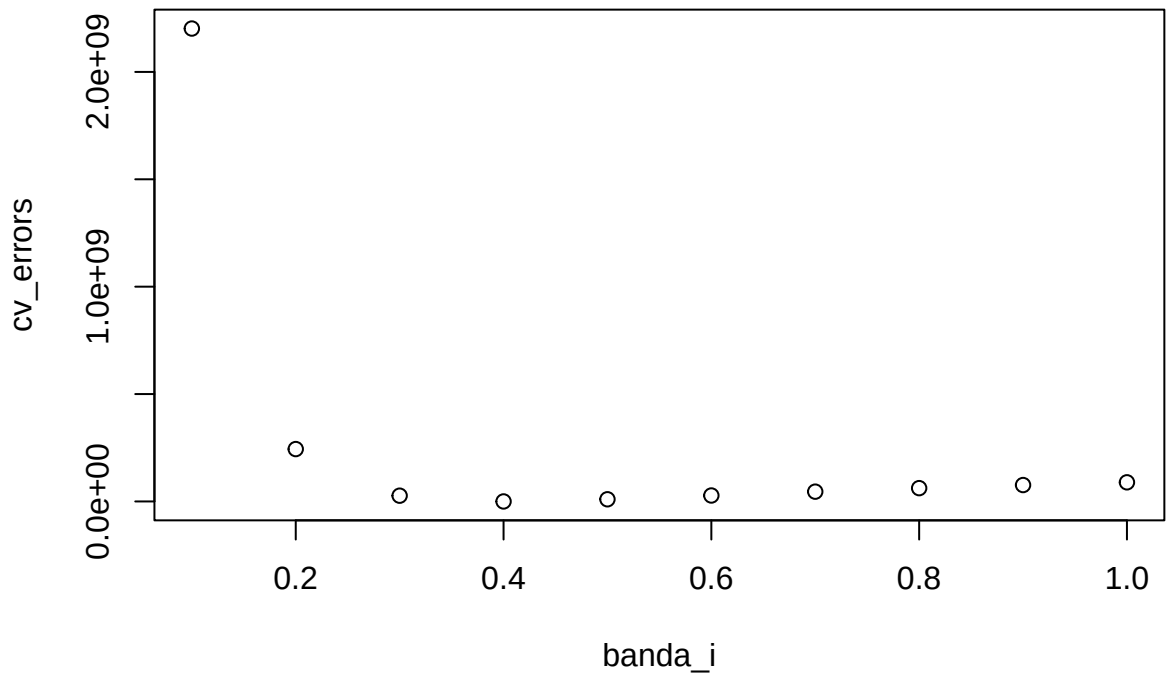
```
## [1] 5675069
```

Aplicando a regressão via Nadaraya-Watson usando o enconding:

```
library(fields)

set.seed(1)
split<- sample(c("Treinamento","Teste"),prob=c(0.6,0.4),size = nrow(dados),replace = TRUE)


x_treinamento <- dados2[split=="Treinamento",]
x_teste <- dados2[split=="Teste",]
#matriz de distâncias euclidianas entre as linhas de x
dist_matrix <- rdist(x_treinamento)
# A matriz dist_matrix conterá as distâncias euclidianas entre os pontos


y_treinamento <- dados$price[split=="Treinamento"]
y_teste <- dados$price[split=="Teste"]

banda_i<- round(seq(0.1,1,length.out=10),1)


cv_errors <- numeric(length(banda_i))



# Realizando a validação cruzada para cada valor de banda
for (ii in seq_along(banda_i)){
  bandwidth <- banda_i[ii]


    # pesos usando o kernel gaussiano
  weights <- (1 / sqrt(2 * pi * (bandwidth)^2)) * exp(-dist_matrix^2 / (2 * bandwidth^2))

    # estimativas usando o método Nadaraya-Watson
  predito_nw <- rowSums(weights * y_treinamento)

    # Calculando o risco com validação cruzada para o valor de banda atual
  cv_errors[ii] <- mean((y_treinamento - predito_nw)^2)

}




##############################################################################

# o valor de banda que resulta no menor erro de validação cruzada
best_bandwidth <- banda_i[which.min(cv_errors)]

print(paste("Melhor valor de banda:", best_bandwidth))
```

```
## [1] "Melhor valor de banda: 0.4"
```

```
plot(banda_i,cv_errors)
```



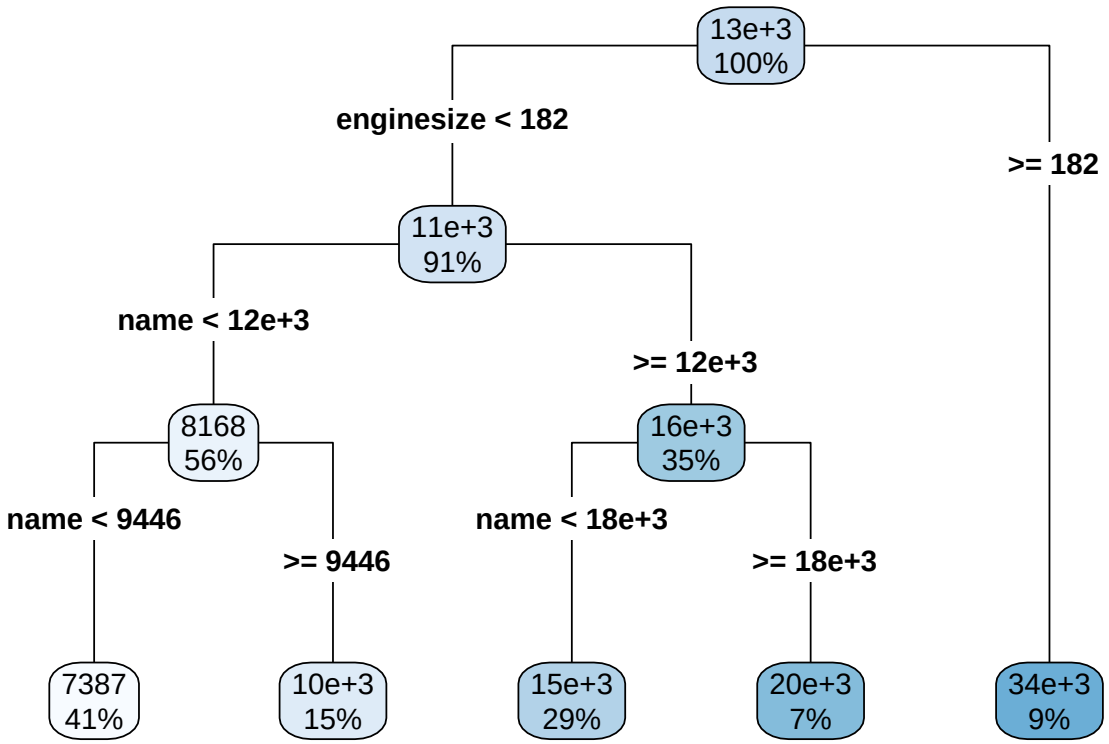

```
dist_matrix <- rdist(x_treinamento)

# Calculando os pesos usando o kernel gaussiano
weights_teste <- (1 / sqrt(2 * pi * (best_bandwidth)^2)) * exp(-dist_matrix^2 / (2 * best_bandwidth^2))

# Calculando as estimativas usando o método Nadaraya-Watson
predito_nw_teste<- rowSums(weights_teste * y_teste)

    # Calculando o erro de validação cruzada para o valor de banda atual
riscos_nw <- mean((y_teste - predito_nw_teste)^2)
riscos_nw
```

```
## [1] 176282643
```

Aplicando a regressão via Árvores usando o target enconding:

```
set.seed(1)


library(rpart)
library(rpart.plot)
```

```r
dados.tudo<-dados2 %>% as.data.frame %>% mutate(Price=dados$price)



#ajuste da arvore:

arvore= rpart(Price ~.,data = dados.tudo[split== "Treinamento",])
rpart.plot(arvore,type=4)#plota  a arvore

#poda:

melhorCp = arvore$cptable[which.min(arvore$cptable[,"xerror"]),"CP"]

poda = prune(arvore,cp=melhorCp)

rpart.plot(poda,type=4)
```




```r
#predito arvore:
predito_arvore = predict(poda,dados.tudo[split=="Teste",])

#calculo o risco
riscos_arvore<- (predito_arvore-dados.tudo$Price[split=="Teste"])^2 %>% mean()
riscos_arvore
```

```
## [1] 7425512
```

Aplicando a regressão via Florestas aleatórias usando o target enconding:

```
#precisa ser os dados em target
dim(dados)
```

```
## [1] 205  26
```

```
dados2 <- dados %>% as.matrix()
dim(dados2)
```

```
## [1] 205  26
```

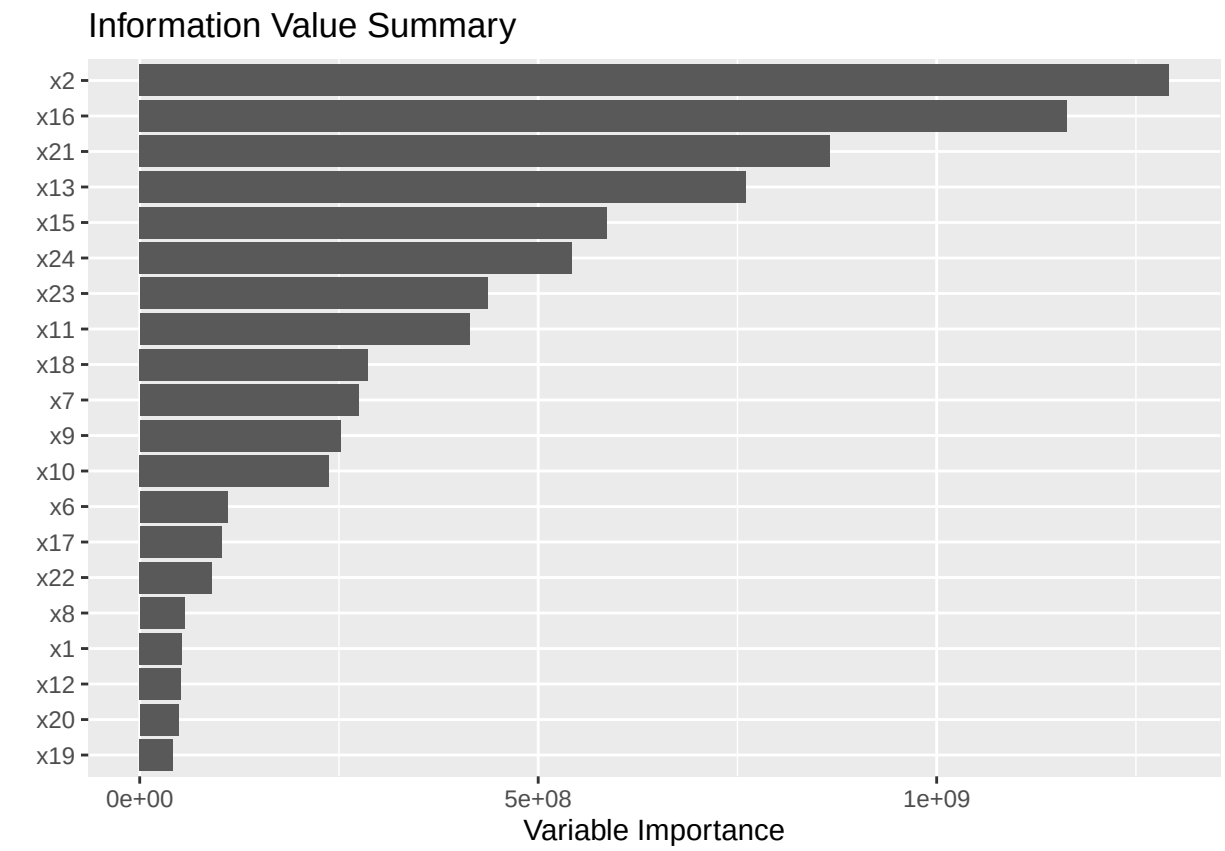
```
dados2 <- dados2[, -c(1, ncol(dados2))]
dim(dados2)
```

```
## [1] 205  24
```

```
valores_ausentes <- sum(is.na(dados2))

dados2 <- replace(dados2, is.na(dados2), 9982.25)
valores_ausentes <- sum(is.na(dados2))
set.seed(1)

split<- sample(c("Treinamento","Teste"),prob=c(0.6,0.4),size = nrow(dados),replace = TRUE)
dados.tudo<-dados2 %>% as.data.frame %>% mutate(Price=dados$price)


library(rpart)
library(rpart.plot)


# Criar um vetor com 189 elementos contendo os nomes "x1", "x2", ..., "x24"
vetor_nomes <- paste0("x", 1:24)

novos_nomes<-c(vetor_nomes,"Price")

colnames(dados.tudo) <- novos_nomes



#floresta aleatoria
floresta = ranger(Price~.,data = dados.tudo[split=="Treinamento",],importance = "impurity")


#predito floresta
predito_floresta = predict(floresta,dados.tudo[split=="Teste",])

importances<- tibble(variable=names(importance(floresta)),importance=importance(floresta)) %>%
  arrange(desc(importance))
```

```
ggplot(importances %>% top_n(n=20),
       aes(x=reorder(variable,importance),y=importance))+
  geom_bar(stat="identity",position="dodge")+ coord_flip()+
  ylab("Variable Importance")+
  xlab("")+
  ggtitle("Information Value Summary")+
  guides(fill=F)+
  scale_fill_gradient(low="red",high = "blue")
```

```
## Selecting by importance
```




```
#calculando os riscos
riscos_florestas<- (predito_floresta$predictions - dados.tudo$Price[split=="Teste"])^2 %>% mean()
riscos_florestas
```

```
## [1] 3287952
```

Conclusão:

```
library(knitr)

dados_tabela <- data.frame(metodos = c("Knn", "Nadaraya Watson", "Árvore de Regressão", "Floresta Aleat
```

```
                          riscos = c(5675069, 176282643, 7425512,3287952))

kable(dados_tabela, format = "simple")
```

metodos	riscos
Knn	5675069
Nadaraya Watson	176282643
Árvore de Regressão	7425512
Floresta Aleatória	3287952

Intervalo de confiança para a regressão via KNN usando o target enconding:

```
n_teste <- sum(split == "Teste")
residuos <- predito_knn - dados$price[split == "Teste"]
desvio_padrao <- sqrt(sum(residuos^2) / (n_teste))

z_critico <- qnorm(0.975)
erro_padrao <- desvio_padrao / sqrt(n_teste)
intervalo_confianca_knn <- c(riscos_knn - z_critico * erro_padrao, riscos_knn + z_critico * erro_padrao)
intervalo_confianca_knn
```

```
## [1] 5674556 5675581
```

Intervalo de confiança para a regressão via NW usando o target enconding:

```
n_teste <- sum(split == "Teste")
residuos <- predito_nw_teste - dados$price[split == "Teste"]
desvio_padrao <- sqrt(sum(residuos^2) / (n_teste))

z_critico <- qnorm(0.975)
erro_padrao <- desvio_padrao / sqrt(n_teste)
intervalo_confianca_nw <- c(riscos_nw - z_critico * erro_padrao, riscos_nw + z_critico * erro_padrao)
intervalo_confianca_nw
```

```
## [1] 176279180 176286106
```

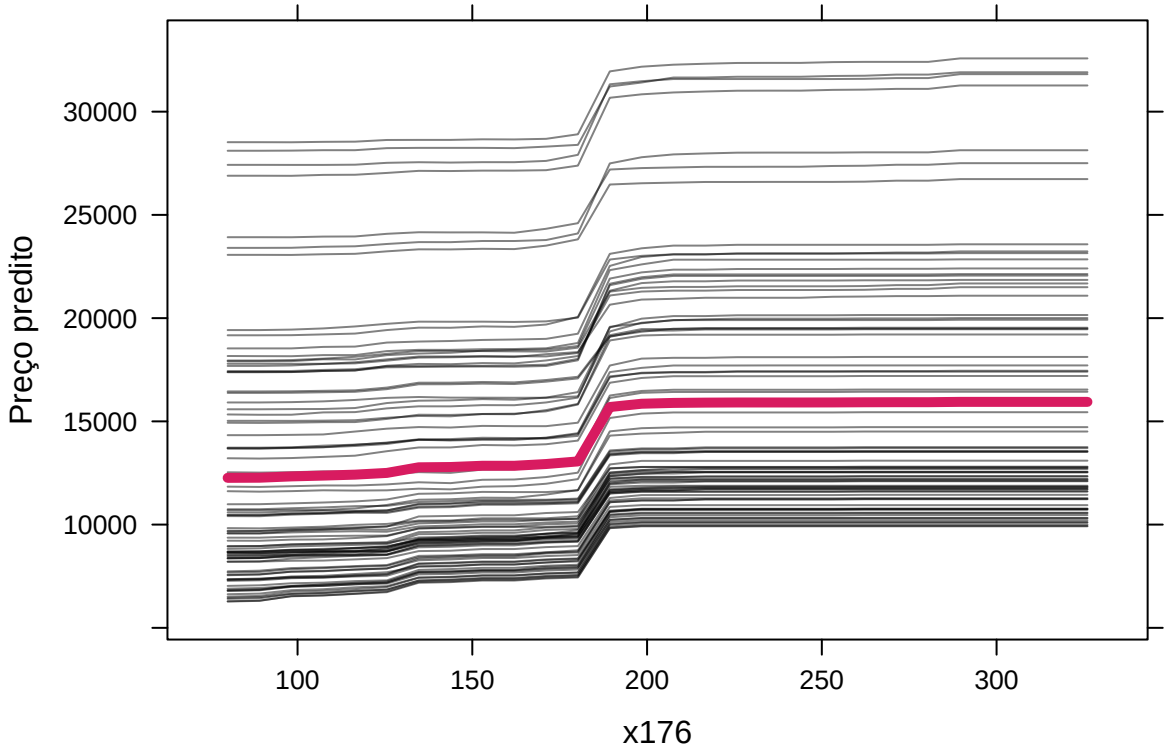
Conclusão:

Ao analisar os resultados das estimativas dos riscos para os diferentes métodos, aplicando tanto a técnica de "Target Encoding" quanto a de "Dummy Encoding", há uma redução muito evidente nas estimativas de risco ao empregar o "Target Encoding" em comparação com o "Dummy Encoding". Isso implica que o método "Target Encoding" desencadeou uma significativa redução nos riscos previstos, contribuindo assim para a capacidade de tomar decisões mais precisas e confiáveis.

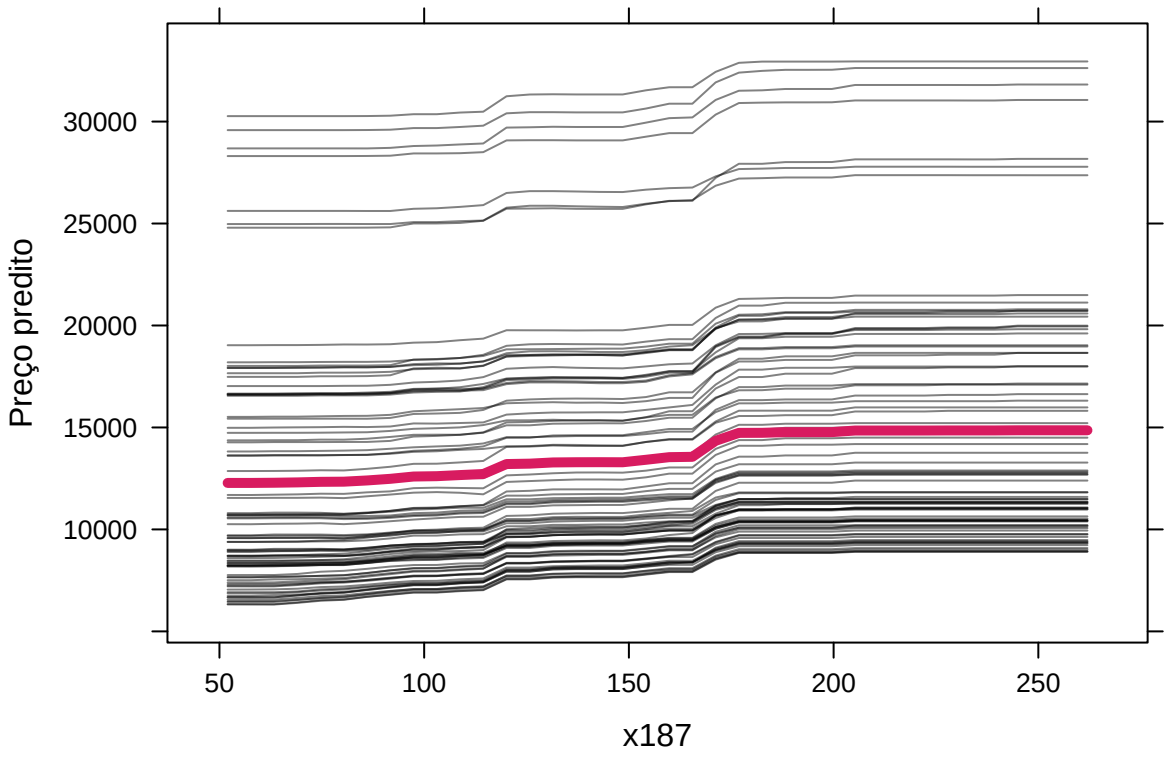
Item F

Floresta com PDP usando as variáveis: x176= "enginesize" tem a maior importância, seguida por x187 = "horsepower" e x163 = "curbweight"

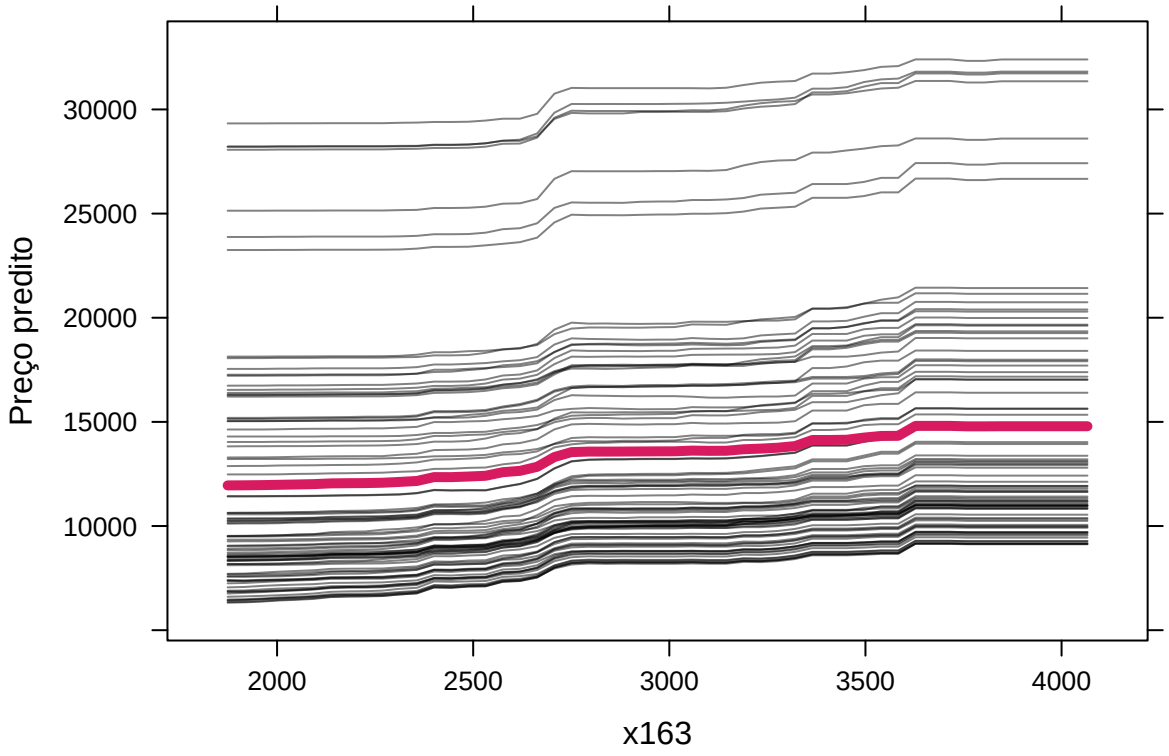
```r
library(ranger)

dados.tudo<-dtm.matrix %>% as.data.frame %>% mutate(Price=dados$price)


# Criar um vetor com 189 elementos contendo os nomes "x1", "x2", ..., "x190"
vetor_nomes <- paste0("x", 1:190)

novos_nomes<-c(vetor_nomes,"Price")

colnames(dados.tudo) <- novos_nomes


#floresta aleatoria
floresta = ranger(Price~.,data = dados.tudo[split=="Treinamento",],importance = "impurity")

library(pdp)
pdp::partial(floresta,train=as.data.frame(dados.tudo[split=="Teste",])%>% dplyr::slice(1:100)
             ,
             pred.var=
               "x176",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```



```
pdp::partial(floresta,train=as.data.frame(dados.tudo[split=="Teste",])%>% dplyr::slice(1:100)
             ,
             pred.var=
               "x187",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```




```
pdp::partial(floresta,train=as.data.frame(dados.tudo[split=="Teste",])%>% dplyr::slice(1:100)
             ,
             pred.var=
               "x163",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```



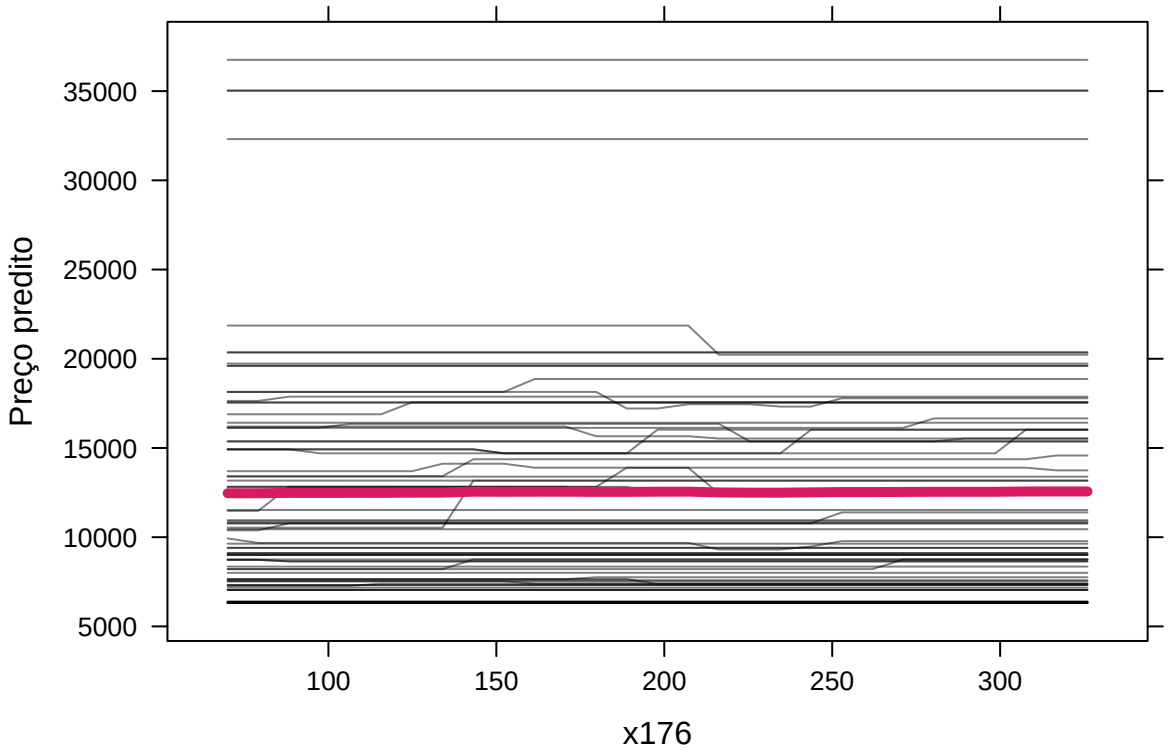
Pelos gráficos, percebemos que:

- a variável “enginesize” que corresponde à variável x176, tem uma diferença, principalmente quando o preço predito está entre 1200 e 1500.
- a variável “horsepower” que corresponde à variável 187, tem uma pequena diferença quando se aumenta o preço predito
- a variável “curbweight” que corresponde à variável 163 não se altera quando aumenta o preço predito.

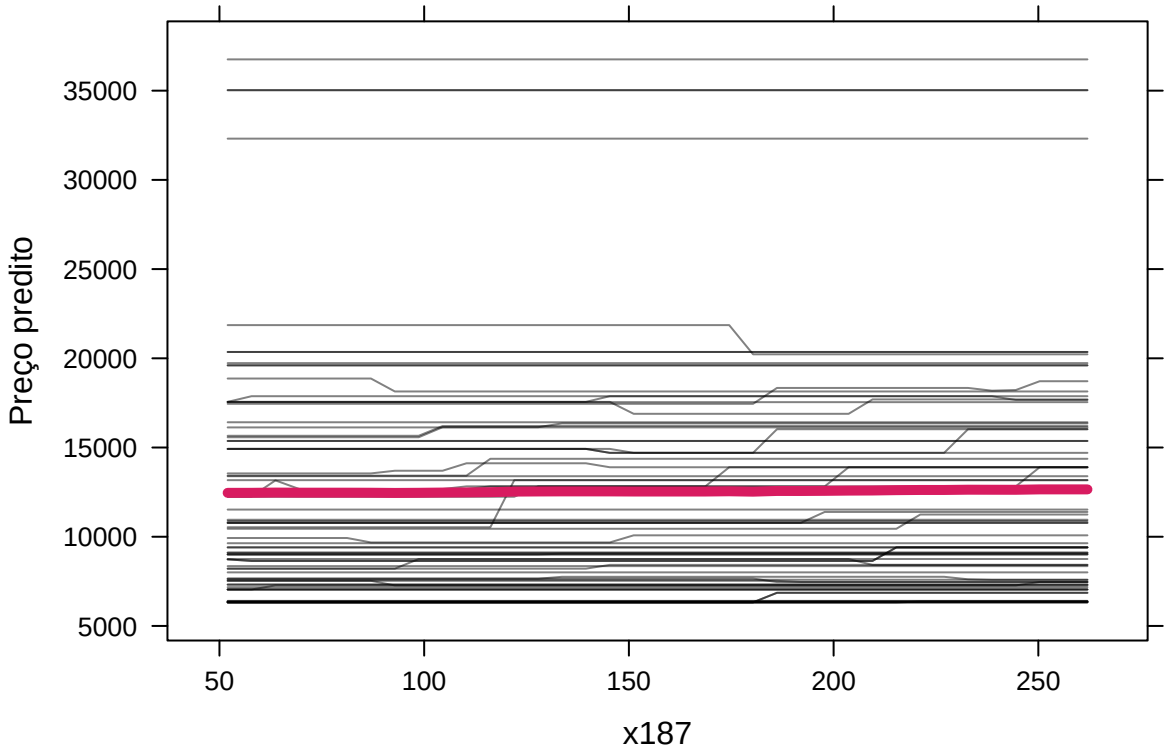
Knn com pdp usando as variáveis: x176= “enginesize” tem a maior importância, seguida por x187 = “horsepower” e x163 = “curbweight”

```
library(caret)


validationIndex <- createDataPartition(dados.tudo$Price, p=0.60, list=FALSE)
train <- dados.tudo[validationIndex,] # 60% of data to training
test <- dados.tudo[-validationIndex,] # remaining 40% for test
#Divide o conjunto de dados em treinamento e teste.


trainControl <- trainControl(method="repeatedcv", number=best.k, repeats=5)
metric <- "RMSE"  # Métrica adequada para regressão
```

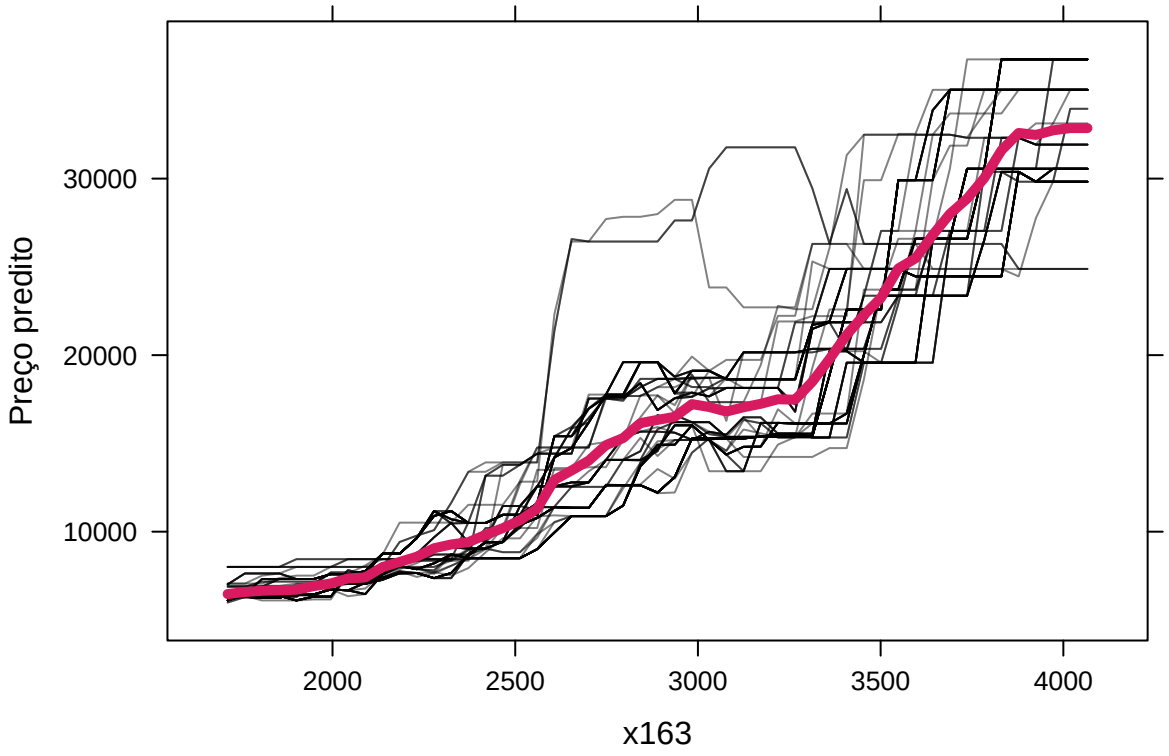
```
set.seed(7)
fit.knn <- train(Price ~ ., data=train, method="knn",
                 metric=metric, trControl=trainControl)


pdp::partial(fit.knn,train=as.data.frame(test)%>% dplyr::slice(1:200)
             ,
             pred.var=
               "x176",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```




```
pdp::partial(fit.knn,train=as.data.frame(test)%>% dplyr::slice(1:200)
             ,
             pred.var=
               "x187",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```




```
pdp::partial(fit.knn,train=as.data.frame(test)%>% dplyr::slice(1:200)
             ,
             pred.var=
               "x163",#pred.fun = predict.fun,
             ice = TRUE) %>% plotPartial(smooth = TRUE, alpha=0.5,
                                         pdp.lwd=5,pdp.col= "#D81B60",
                                         ylab="Preço predito")
```



Pelos gráficos, percebemos que:

- a variável “enginesize” que corresponde à variável x176, quase não há diferença quando o preço predito aumenta
- a variável “horsepower” que corresponde à variável 187, quase não há diferença quando se aumenta o preço predito
- a variável “curbweight” que corresponde à variável 163, há um aumento considerável quando aumenta o preço predito.