Análise dos Pinguins de Palmer 2

Introdução à Ciência de Dados - Estatística | Univeridade Federal de Uberlândia

Matheus de Moraes Neves

1 Introdução

Este projeto tem como objetivo analisar o conjunto de dados *Palmer Penguins*, um conjunto coletado no Arquipélogo Palmer (Antártica). Contendo medidas e outras informações de 3 espécies catalogadas: Adelie, Chinstrap e Gentoo. Após a limpeza de dados e análise exploratória, serão construidos modelos de **KNN (K-Nearest Neighbors)**, **Random Forest** e **Support Vector Machine (SVM)** com o objetivo de classificar as 3 espécies de pinguins do arquipélogo.

Figure 1: Os 3 Pinguins de Palmer

As bibliotecas utilizadas serão *tidyverse*, *stringr*, *cowplot*, *rpart*, *rpart.plot* e *e1071*.

2 Importação e Limpeza

Importando o conjunto de dados disponibilizado pelo Professor Pedro Franklin, e visualizando as 5 primeiras linhas da tabela:

```
df = read.csv(file = "palmerpenguins.csv")

head(df)
```

```
  species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
1  Adelie Torgersen           36.6          17.8               185        3700
2  Adelie Torgersen           34.6          17.2               189        3200
3  Adelie     Dream           37.0          16.5               185        3400
4  Adelie     Dream           38.3          19.2               189        3950
5  Adelie Torgersen           45.8          18.9               197        4150
6  Adelie     Dream           41.3          20.3               194        3550
     sex year
1 female 2007
2 female 2008
3 female 2009
4   male 2008
5   male 2008
6   male 2008
```

É possível verificar as 8 colunas da tabelas, contendo informações sobre o espécie, ilha que foi catalogado, comprimento e profundida do bico em milimetros, massa corporal em gramas, sexo e ano. Como a tabela está na ordem 'Adelie', 'Gento' e 'Chinstrap' na coluna espécie, então os primeiros dados serão todos da espécie Adelia, e os últimos da espécie 'Chinstrap':

```
# Verificando os 5 últimos dados da tabela
tail(df)
```

```
      species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
328 chinstrap  Dream           51.3          19.2               193        3650
329 chinstrap  Dream           50.2          18.7               198        3775
330 chinstrap  Dream           45.2          17.8               198        3950
331 chinstrap  Dream           43.5          18.1               202        3400
332 chinstrap  Dream           46.5          17.9               192        3500
333 chinstrap  Dream           49.0          19.5               210        3950
       sex year
328   male 2007
```

```
329 female 2009
330 female 2007
331 female 2009
332 female 2007
333   male 2008
```

Como os dados estão agrupados numa ordem específica, será importante embaralharos para aplica um modelo de classificação posteriormente. Verificando como estão definidos os tipos de dados:

```
str(df)
```

```
'data.frame':   333 obs. of  8 variables:
 $ species          : chr  "Adelie" "Adelie" "Adelie" "Adelie" ...
 $ island           : chr  "Torgersen" "Torgersen" "Dream" "Dream" ...
 $ bill_length_mm   : num  36.6 34.6 37 38.3 45.8 41.3 37.6 46 39.3 39.5 ...
 $ bill_depth_mm    : num  17.8 17.2 16.5 19.2 18.9 20.3 17 21.5 20.6 17.8 ...
 $ flipper_length_mm: int  185 189 185 189 197 194 185 194 190 188 ...
 $ body_mass_g      : int  3700 3200 3400 3950 4150 3550 3600 4200 3650 3300 ...
 $ sex              : chr  "female" "female" "female" "male" ...
 $ year             : int  2007 2008 2009 2008 2008 2008 2008 2007 2007 2007 ...
```

Como *species*, *island* e *sex* estão como caracteres e não como categorias, será necessário transforma-las em fatores posteriormente. Mas antes, é necessário verificar possíveis erros de ortografia:

```
# Transformando a coluna 'species' em categorias e encontrando a frequência de
# cada categoria
summary(as.factor(df$species))
```

```
   adelie    Adelie chinstrap Chinstrap    gentoo    Gentoo
       37       109        18        50        30        89
```

Como a linguagem R é *case-sensitive* (sensível a letras maiscúlas e minuscúlas), então ele irá identificar, como por exemplo, 'adelie' e 'Adelie' como duas categorias diferentes, mesmo sendo a mesma espécie. Verificando para a outra possível categroria:

```
summary(as.factor(df$island))
```

```
   Biscoe     Dream Torgersen
      163       123        47
```

Não há erro de ortografia nessa coluna. Agora verificando para a coluna do sexo do pinguim:

```
summary(as.factor(df$sex))
```

```
female   male
   165    168
```

Também não há erro de ortografia. Manipulando os dados transformando as colunas de caracteres em colunas de categorias:

```
# Transformando os dados cuja o nome das espécies começam com letra minúscula
# e colocando-os com início em caixa alta.
df$species[str_detect(df$species, pattern = "adelie")] <- "Adelie"
df$species[str_detect(df$species, pattern = "chinstrap")] <- "Chinstrap"
df$species[str_detect(df$species, pattern = "gentoo")] <- "Gentoo"

# Transformando os campos "species" e "islando" em categorias
df = df |>
  mutate(across(c(species, island, sex), as.factor))
```

E verificando a presença de NA´s no conjunto de dados:

```
# Soma do vetor booleano de "is.na(df"), em que se não detectar NA, retornará 0
# i.e., Falso
sum(is.na(df))
```

```
[1] 0
```

Portanto, não há presença de NA´s no conjunto de dados.

3 Análise Exploratória e Gráfica

Neste capítulo serão serão analisadas medidas descritivas e serão feitos alguns gráficos.

3.1 Análise Geral

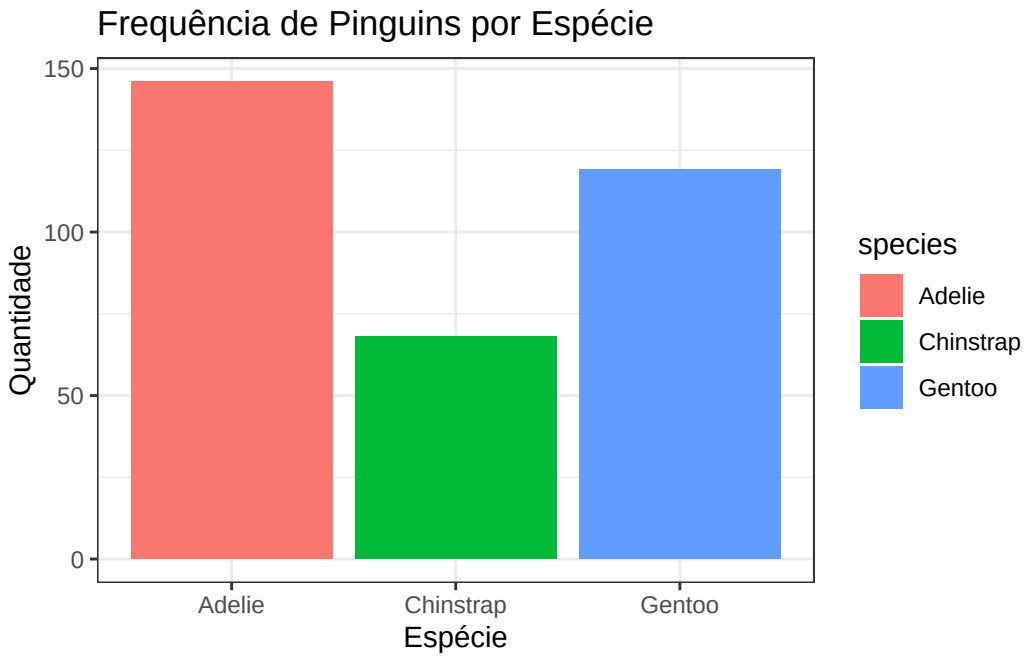
Primeiramente, um resumo estatístico geral:

```
summary(df)
```

```
      species          island    bill_length_mm  bill_depth_mm
 Adelie   :146   Biscoe   :163   Min.   :32.10   Min.   :13.10
 Chinstrap: 68   Dream    :123   1st Qu.:39.50   1st Qu.:15.60
 Gentoo   :119   Torgersen: 47   Median :44.50   Median :17.30
                                 Mean   :43.99   Mean   :17.16
                                 3rd Qu.:48.60   3rd Qu.:18.70
                                 Max.   :59.60   Max.   :21.50
 flipper_length_mm  body_mass_g       sex           year
 Min.   :172       Min.   :2700   female:165   Min.   :2007
 1st Qu.:190       1st Qu.:3550   male  :168   1st Qu.:2007
 Median :197       Median :4050                Median :2008
 Mean   :201       Mean   :4207                Mean   :2008
 3rd Qu.:213       3rd Qu.:4775                3rd Qu.:2009
 Max.   :231       Max.   :6300                Max.   :2009
```

É possível verificar que a maioria das espécies são Adelie, havendo um desequilíbrio na frequência das espécies, sendo a espécie Chinstrap com 20% de ocorrências.

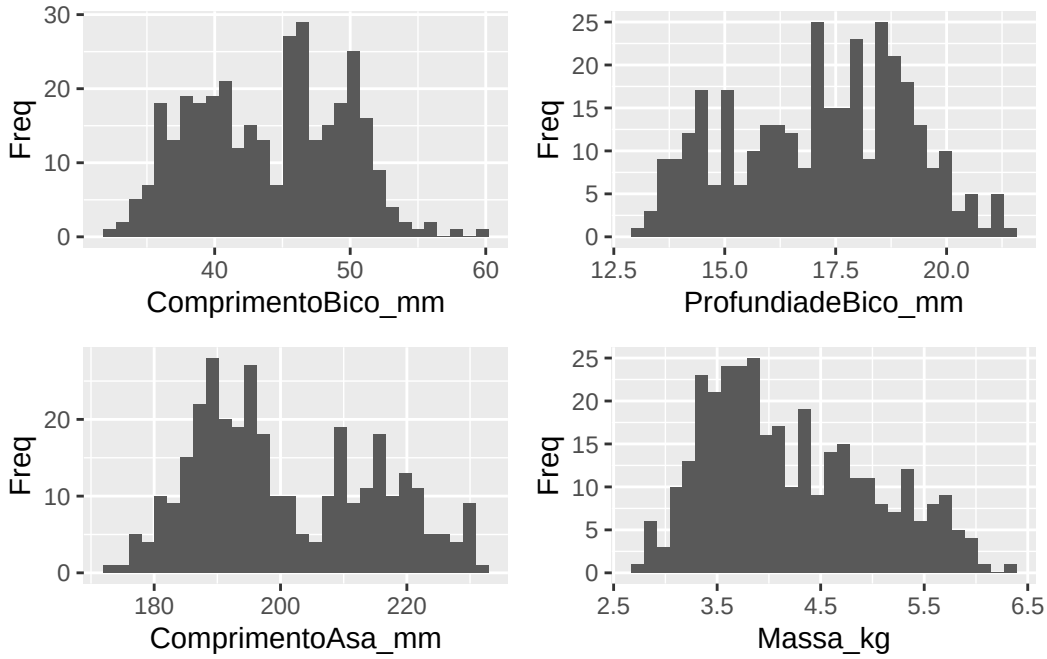
```
ggplot(df, aes(x = species, fill = species)) +
  geom_bar() +
  labs(title = "Frequência de Pinguins por Espécie",
       x = "Espécie",
       y = "Quantidade") +
  theme_bw()
```



Além disso, a maioria das ilhas em que foram catalogados é a Biscoe. A quantidade de machos e fêmeas é igual e as medidas númericas não apresentam assimetria. Analisando mais atentamente a distribuição de forma gráfica:

```
# Histograma para variável Comprimento de Bico
p1 = ggplot(df, aes(x = bill_length_mm)) +
  geom_histogram(position = "identity") +
  labs(x = 'ComprimentoBico_mm',
       y = 'Freq')

# Histograma para variável Profundiade de Bico
p2 = ggplot(df, aes(x = bill_depth_mm)) +
  geom_histogram(position = "identity") +
  labs(x = 'ProfundiadeBico_mm',
       y = 'Freq')

# Histograma para variável Comprimento de Asa
p3 = ggplot(df, aes(x = flipper_length_mm)) +
  geom_histogram(position = "identity") +
  labs(x = 'ComprimentoAsa_mm',
       y = 'Freq')

# Histograma para variável Massa Corporal (em Kg)
```

```
p4 = ggplot(df, aes(x = 0.001*body_mass_g)) +
  geom_histogram(position = "identity") +
  labs(x = 'Massa_kg',
       y = 'Freq')

plot_grid(p1,p2,p3,p4)
```



A maioria dos pinguins possuem massa corporal mais próxima de 3.5 Kg, isso se deve pelo fato de que a maioria dos pinguins são das espécie Adelie e Chinstrap, entendendo o motivo mais a frente. Além disso, alguns histogramas, principalmente sobre o Comprimento da Asa, possuem um padrão peculiar na sua distribuição, como se fossem 2 grupos diferentes. É necessário então analisar os dados agrupando por espécie, para entender melhor essas distribuições: Por qual motivo esses dados são heterogêneos?

3.2 Análise por Espécie

Verificando a média por cada espécie:

```
# Médias de cada medida numérica agrupada por espécie
group_by(df, species) |>
  summarise(M_Massa = mean(body_mass_g),
            M_ComprimentoBico = mean( bill_length_mm),
```

```
            M_ProfundidadeBico = mean(bill_depth_mm),
            M_ComprimentoAsa = mean(flipper_length_mm))
```

```
# A tibble: 3 x 5
  species   M_Massa M_ComprimentoBico M_ProfundidadeBico M_ComprimentoAsa
  <fct>       <dbl>             <dbl>              <dbl>            <dbl>
1 Adelie      3706.              38.8               18.3             190.
2 Chinstrap   3733.              48.8               18.4             196.
3 Gentoo      5092.              47.6               15.0             217.
```

É possível verificar que a espécie Gentoo possui a maior massa e comprimento de asa em média comparada à outras espécies, ao mesmo tempo que possui o bico menos profundo comparada à outras espécies. Analisando o quanto varia em cada espécie, por meio de uma medida de dispersão relativa à média, o Coeficiente de Variação. Como o CV é adimensional, então é possível utiliza-lo para comparar as diferentes medidas dos penguins (dados indo desde miliemtros até gramas).

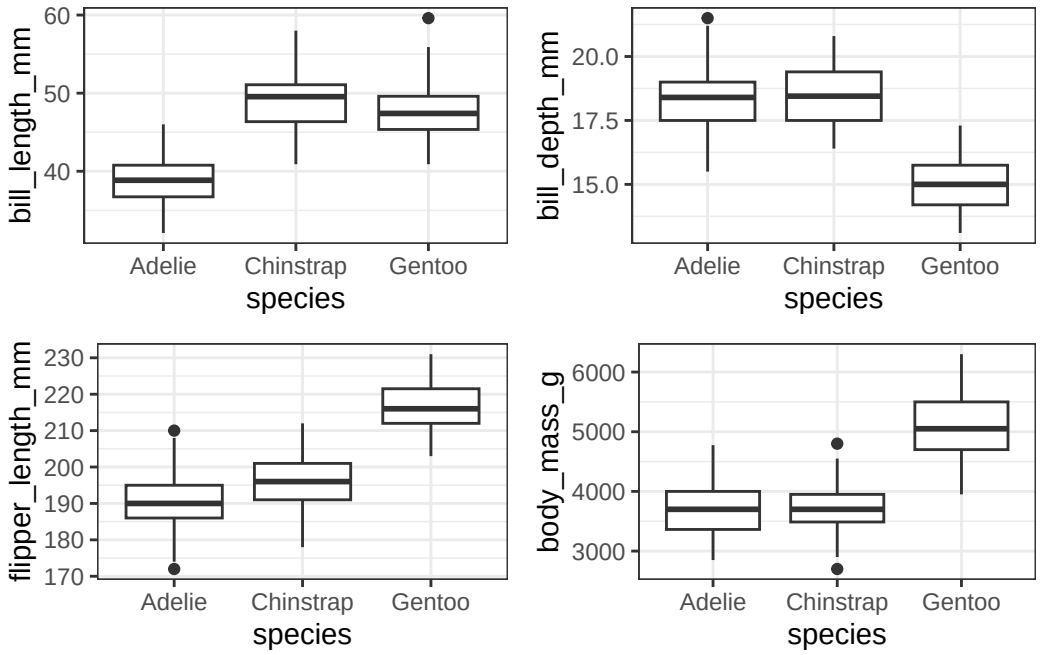
```
# Calculando o Coeficiente de Variação (CV) em % de cada espécie
group_by(df, species) |>
  summarise(CV_Massa = sd(body_mass_g)/mean(body_mass_g)*100,
            CV_ComprimentoBico = sd(bill_length_mm)/mean(bill_length_mm)*100,
            CV_ProfundidadeBico = sd(bill_depth_mm)/mean(bill_depth_mm)*100,
            CV_ComprimentoAsa = sd(flipper_length_mm)/mean(flipper_length_mm)*100)
```

```
# A tibble: 3 x 5
  species   CV_Massa CV_ComprimentoBico CV_ProfundidadeBico CV_ComprimentoAsa
  <fct>        <dbl>              <dbl>               <dbl>             <dbl>
1 Adelie       12.4                6.86                6.65              3.43
2 Chinstrap    10.3                6.84                6.16              3.64
3 Gentoo        9.85               6.53                6.57              3.03
```

O campo em que mais teve variabilidade é a massa, sendo a espécie Adelie com maior variação com 12% de CV. Essa mesma espécie segue tendo a maior variação entre as espécies nas outras medidas, exceto no comprimento da asa. A medida que menos variou das analisadas é o comprimento da asa, sendo o maior delas o de Chinstrap com 4% de variação em relação à média.

Confirmando essas primeiras impressões por meio de gráfico de boxplot:

```
bico_c = ggplot(df, mapping = aes(x=species, y=bill_length_mm)) +
  geom_boxplot()+
  theme_bw()

bico_p = ggplot(df, mapping = aes(x=species, y=bill_depth_mm)) +
  geom_boxplot()+
  theme_bw()

asa = ggplot(df, mapping = aes(x=species, y=flipper_length_mm)) +
  geom_boxplot()+
  theme_bw()

massa = ggplot(df, mapping = aes(x=species, y=body_mass_g)) +
  geom_boxplot()+
  theme_bw()

plot_grid(bico_c, bico_p, asa, massa)
```



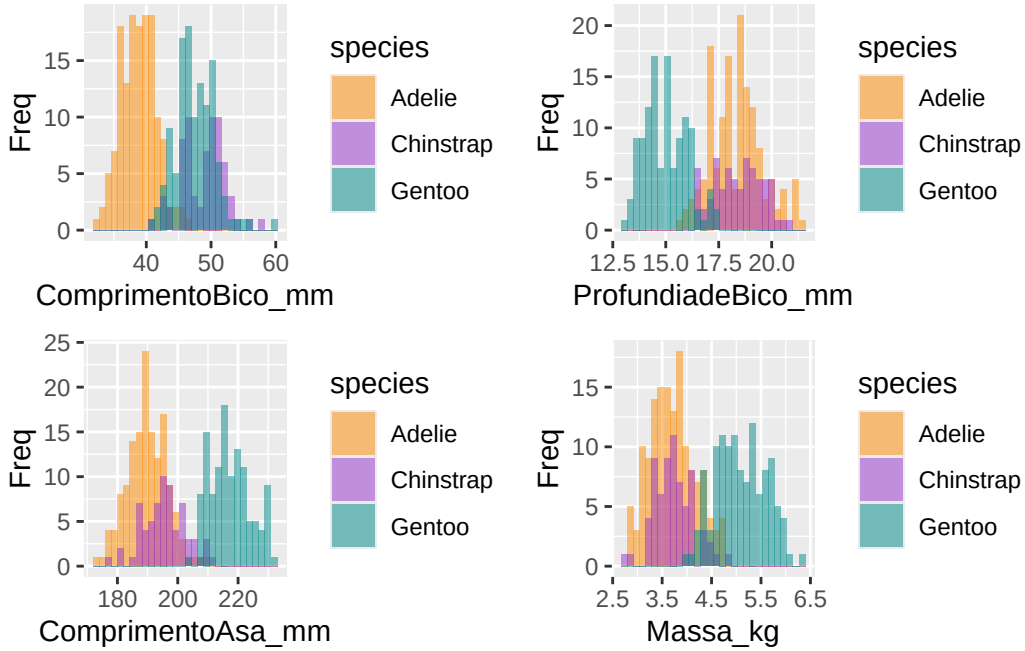
É possível verificar que a tendência do Gentoo se confirma: Seus comprimentos de bico e de asa estão em números mais elevados, enquanto sua profundida de bico é a menor das espécies. Adelie é a espécie que apresentou mais outliers em suas medidas, apresentando os menores comprimentos de bicos e asas, e menor massa corporal junto de Chinstrap. Além

disso, Chinstrap possuir o maior comprimento e profundida do bico, sendo o segundo maior comprimento de bico da espécie Gentoo.

Sintetizando as informações exploradas, Adelie é uma espécie que aparenta possuir as menores medidas de seu corpo, exceto a medida de profundiade do bico. Enquanto Gentoo é a espécie que tem as maiores proporções corporais, porém contendo a menor profundidade do bico. Chinstrap parece ser uma equílibro entre dois, com sua asa e massa sendo pequena como de Adelie, e comprimento e profundiade de bico próximas ao de Gentoo.

Ficará mais clara como a distribuição dessas medidas se comportam nas 3 espécies com um histograma:

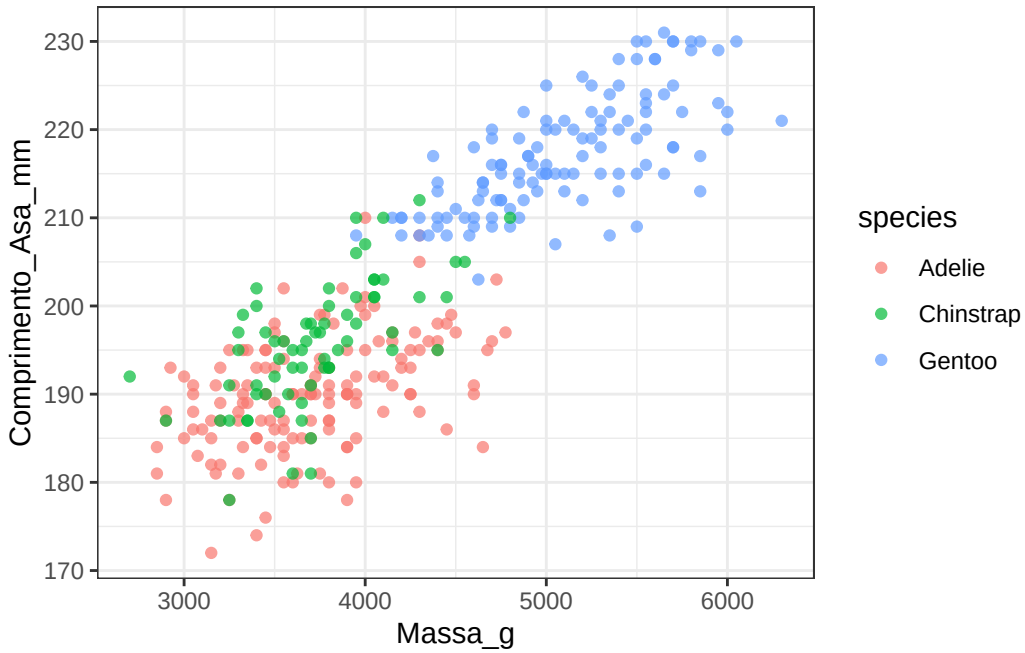
```
# Histograma para variável Comprimento de Bico por cada Espécie
p1 = ggplot(df, aes(x = bill_length_mm)) +
  geom_histogram(aes(fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4"))+
  labs(x = 'ComprimentoBico_mm',
       y = 'Freq')

# Histograma para variável Profundiade de Bico por cada Espécie
p2 = ggplot(df, aes(x = bill_depth_mm)) +
  geom_histogram(aes(fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4"))+
  labs(x = 'ProfundiadeBico_mm',
       y = 'Freq')

# Histograma para variável Comprimento de Asa por cada Espécie
p3 = ggplot(df, aes(x = flipper_length_mm)) +
  geom_histogram(aes(fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4"))+
  labs(x = 'ComprimentoAsa_mm',
       y = 'Freq')

# Histograma para variável Massa Corporal (em Kg) por cada Espécie
p4 = ggplot(df, aes(x = 0.001*body_mass_g)) +
  geom_histogram(aes(fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4"))+
  labs(x = 'Massa_kg',
       y = 'Freq')

plot_grid(p1,p2,p3,p4)
```



Por que essas variáveis apresentaram esse padrão antes? Por que esses dados eram heterogêneos (Seção 3.1)? O motivo são as espécies, as espécies são heterogêneas (diferem uma da outra), de acordo com os gráficos acima. Porém não está tão claro os padrões de agrupamento.

Verificando os agrupamentos das espécies de forma mais detalhada, com alguns exemplor de gráficos de dispersão.

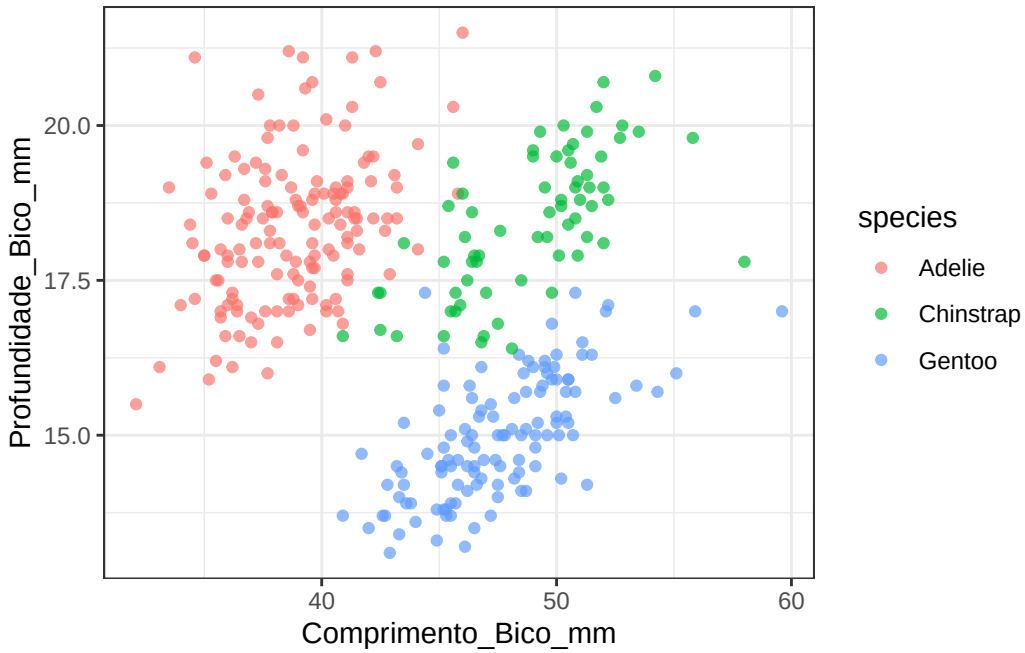
```
# Gráfico de Dispersão de Comprimento de Asa x Massa Corporal
ggplot(df, aes(x = body_mass_g, y = flipper_length_mm))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Massa_g',
       y = 'Comprimento_Asa_mm')
```



Como descrito anteriormente, Chinstrap apresenta uma tendência de se misturar em algum dos dois grupos de espécies, sendo importante dar uma atenção especial a ele quando for aplicado o modelo knn. Nesse caso, os dois grupos são "Adelia + Chinstrap" e "Gentoo".

Por outro lado, tem casos em que Chistrap possui 2 variáveis que acompanha Adelie e Chinstrap: Como comentando anteriormente, Chinstrap possui comprimento de bico elevado como Gentoo, e possui profundidade de bico elevada como Adelie, ou seja, ele possivelmente será um grupo mais destacado num gráfico de dispersão comprimento_bico x profundidade bico, diferenciando-se do gráfico anterior:

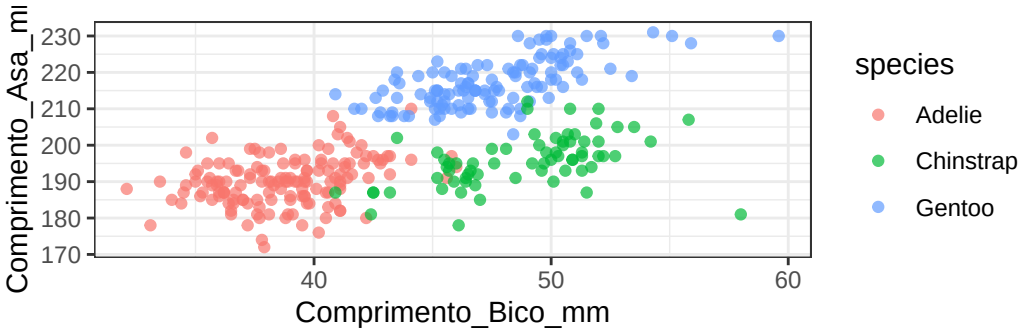
```
# Gráfico de Dispersão de Comprimento do Bico x Profundiade do Bico
ggplot(df, aes(x = bill_length_mm, y = bill_depth_mm))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Comprimento_Bico_mm',
       y = 'Profundidade_Bico_mm')
```

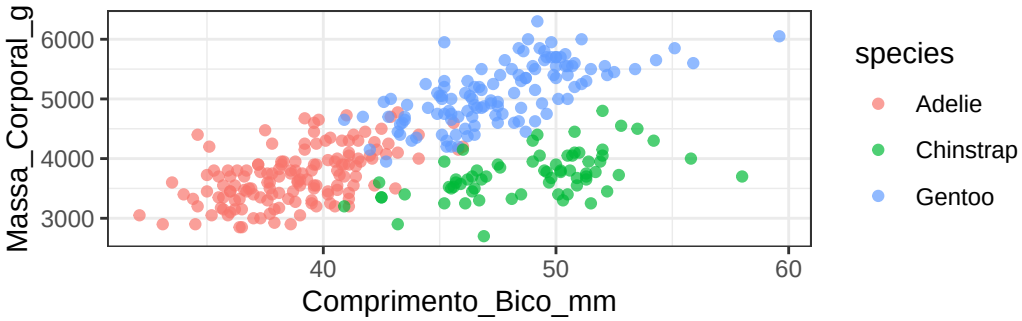


Se um novo pinguim for catalogado neste arquipélogo (um novo ponto sem cor gerado no gráfico), é relativamente simples um ser humano inferir a espécie do penguim de forma visual no gráfico acima. Isso é um bom indício para a eficácia do modelo knn.

Outros casos em que isso ocorre:

```
# Gráfico de Dispersão de Comprimento do Bico x Comprimento Asa
s1 = ggplot(df, aes(x = bill_length_mm, y = flipper_length_mm))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Comprimento_Bico_mm',
       y = 'Comprimento_Asa_mm')

# Gráfico de Dispersão de Comprimento do Bico x Massa
s2 = ggplot(df, aes(x = bill_length_mm, y = body_mass_g))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Comprimento_Bico_mm',
       y = 'Massa_Corporal_g')

plot_grid(s1,s2,nrow = 2)
```





Portanto, é possível verificar, de forma visual nos histogramas separado por espécie, quais combinações entres 2 variáveis terá 3 grupos, e não 2 grupos.

Por fim, verificando a distribuição das espécies por ilha:

```
table(df$species, df$island)
```

```
           Biscoe Dream Torgersen
  Adelie       44    55        47
  Chinstrap     0    68         0
  Gentoo      119     0         0
```

O que explica a maioria das amostras serem de penguins Adelie, eles estão estão em todas as ilhas. As outras espécies estão apenas em 1 ilha específica. Uma informação útil para fazer um modelo de classificação. A maioria dos pinguins estão na linha Biscoe.

4 Implementação do Modelo KNN

Para começar, como visto no capítulo 2, o conjunto de dados está ordenado em 'Adelie', 'Gentoo' e depois 'Chinstrap'. Como o conjunto de dados será separado em um conjunto para treino e outro para teste, será necessário embaralhar a tabela para que nenhum dos dois conjuntos esteja enviesado para alguma espécie.

```
set.seed(123) # Criando uma semente para ter o mesmo valor do embaralhamento
df = df[sample(nrow(df)),] # Embaralhando o conjunto df
n = round(nrow(df)*0.8) # Pegando 80% da qtd de linhas de df e arredondando

treino = df[1:n,] # Separando 80% de df para o conjunto treino
teste = df[-(1:n),] # Separando o resto (20%) de df para o conjunto teste
```

O conjunto treino será o nosso conjunto de "pinguins já coletados", enquanto o teste será os "novos pinguins que seriam catologados" do exemplo do capítulo 3. Um ponto do conjunto teste será colocado no gráfico de treino, será calculada sua distância euclidianda com todos os pontos do treino e será utilizado o primeiro ponto mais próximo (k = 1). Esse processo será feito para todos os valores de teste, o modelo tentará acertar as espécies desse conjunto. Como as espécies do conjunto teste já são conhecidas, então será comparada a previsão do modelo com as espécies do Teste.

4.1 Caso Bidimensional

Criando uma função para o caso de comparar apenas duas variáveis, ou seja, $||Ponto_{teste} - Ponto_{treino}||_2$

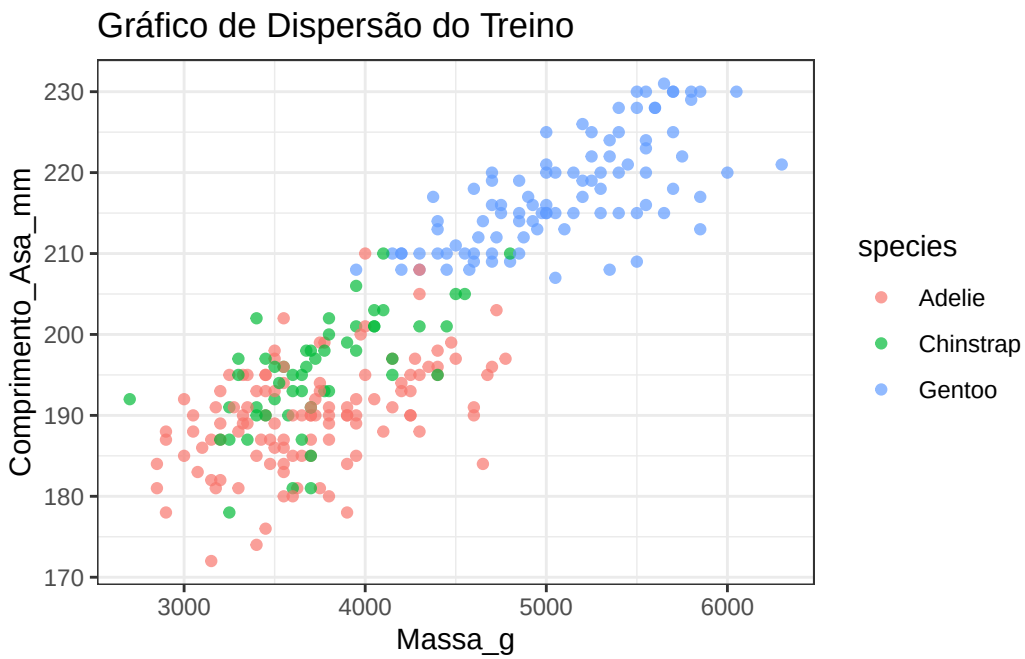
```
# Criando uma função para modularizar o código (facilitar)

knn.amador2d = function(x,y,k){
  # Vetor para armazenar as classificações das observações do teste.
  classificacao = c()

  for(i in 1:nrow(teste)){ # O(N)

    distancias = c() # Vetor para armazenar as distancias em do teste e treino

    for(j in 1:nrow(treino)){ # O(N)

      # Parada cada ponto do teste, calcula sua distancia euclidiana (norma 2)
      # com todos os pontos do treino
```

```
      distancias[j] = sqrt((treino[j,x]-teste[i,x])^2 +
                           (treino[j,y]-teste[i,y])^2)
    }
    # Irá pegar as z menores distancias
    menor = order(distancias)[k] # O(N)

    # No treino, seleciona as z espécies com menor distancia
    # e armazena no vetor classificação como caracter
    classificacao[i] = as.character(treino$species[menor])
  }

  # Acurácia do modelo (taxa de acerto).
  acuracia = mean(classificacao == teste$species)

  return(acuracia)
}
```

Uma pequena observação: A complexidade temporal de knn.amador2d é de $O(n^3)$, i.e., para cada acréscimo de no *input* na função [4], o tempo de execução (ou número de iterações) irá aumentar ao cubo (Ver Apêndice A).

Como já foi testado o modelo knn para comprimento Profundidade do Bico e Comprimento do Bico em aula, então será aplicado para outros gráficos de dispersão apresentados no capítulo 3. Primeiro para Comprimento de Asa e Massa Corporal, verificando agora pelo treino:

```
# Gráfico de Dispersão de Comprimento de Asa x Massa Corporal
ggplot(treino, aes(x = body_mass_g, y = flipper_length_mm))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Massa_g',
       y = 'Comprimento_Asa_mm',
       title = 'Gráfico de Dispersão do Treino')
```



Implementando o modelo para este caso:

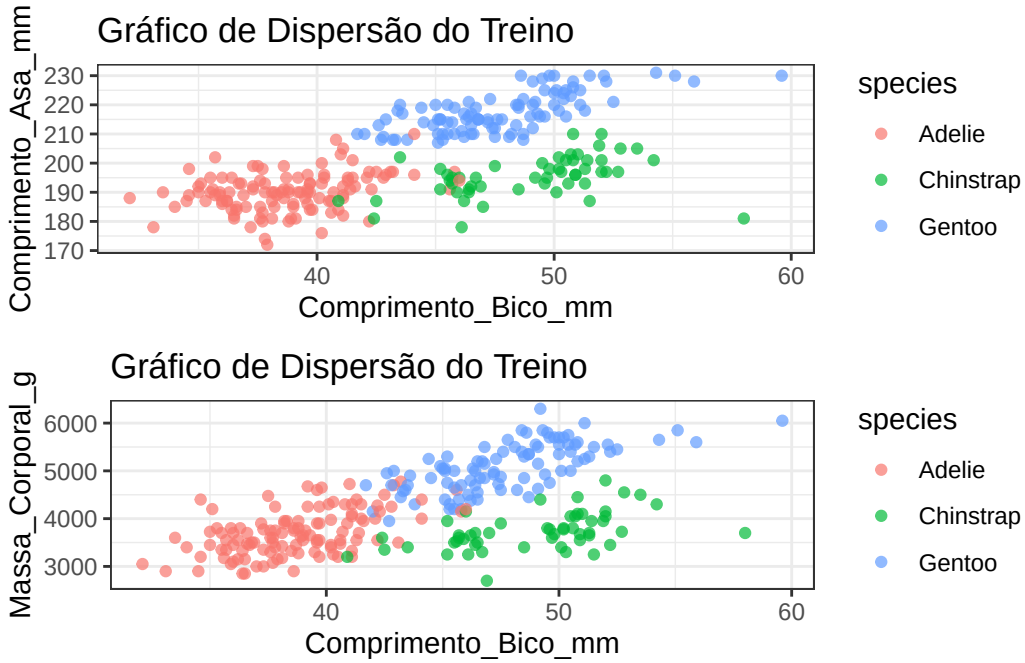
```
# Testando o modelo par k = 1 e Comprimento Asa x Massa
knn.amador2d(5,6,1)
```

```
[1] 0.7313433
```

Agora verificando para os outros gráficos de dispersão apresentados anteriormente.

```
s1 = ggplot(treino, aes(x = bill_length_mm, y = flipper_length_mm))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Comprimento_Bico_mm',
       y = 'Comprimento_Asa_mm',
       title = 'Gráfico de Dispersão do Treino')

# Gráfico de Dispersão de Comprimento do Bico x Massa
s2 = ggplot(treino, aes(x = bill_length_mm, y = body_mass_g))+
  geom_point(aes(colour = species), alpha = 0.7)+
  theme_bw()+
  labs(x = 'Comprimento_Bico_mm',
       y = 'Massa_Corporal_g',
```

```
       title = 'Gráfico de Dispersão do Treino')

plot_grid(s1,s2,nrow = 2)
```



Aplicando o modelo knn bidimensional para os dois casos:

```
print(paste("Em Comprimento_Bico x Comprimento_Asa, a acurácia foi de =",
            knn.amador2d(3,5,1)))
```

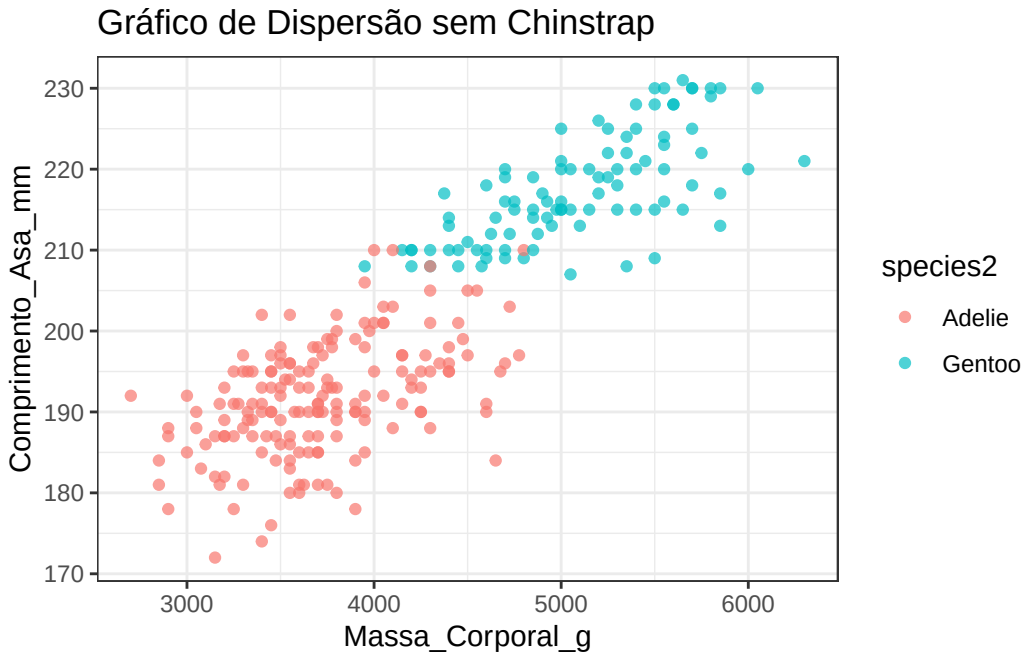
```
[1] "Em Comprimento_Bico x Comprimento_Asa, a acurácia foi de = 0.955223880597015"
```

```
print(paste("Em Comprimento_Bico x Massa_Corporal, a acurácia foi de =",
            knn.amador2d(3,6,1)))
```

```
[1] "Em Comprimento_Bico x Massa_Corporal, a acurácia foi de = 0.835820895522388"
```

É percepitível que o caso aplicando em Comprimento_Asa x Mass_Corporal tem a menor acurácia dentre os 3 testados, isso se deve porque Chinstrap está bem mistura com a espécie Adelie nesse caso. Confirmando isso, imagindo como se Chinstrap fosse a mesma espécie que Adelie:

```r
# Criando uma nova coluna em que Chinstrap seja a mesma espécie que Adelie
treino = treino |>
  mutate(species2 = ifelse(str_detect(species,pattern ="Gentoo"),"Gentoo", "Adelie"))

teste = teste |>
  mutate(species2 = ifelse(str_detect(species,pattern ="Gentoo"),"Gentoo", "Adelie"))

# Gráfico quando Chinstrap fosse a mesma espécie que Adelie
ggplot(treino,aes(x = body_mass_g,y = flipper_length_mm))+
  geom_point(aes(colour = species2),alpha = 0.7)+
  theme_bw()+
  labs(x ='Massa_Corporal_g',
       y ='Comprimento_Asa_mm',
       title = "Gráfico de Dispersão sem Chinstrap" )
```



Testando o modelo como se Chinstrap fosse a mesma espécie que Adelie:

```r
classificacao = c() # vetor para armazenar a classificação das observações do teste.
for(i in 1:nrow(teste)){
  distancias = c()
  for(j in 1:nrow(treino)){
    distancias[j] = sqrt((treino$body_mass_g[j]-teste$body_mass_g[i])^2+
                         (treino$flipper_length_mm[j]-teste$flipper_length_mm[i])^2)
```

```
  }
  menor <- order(distancias)[1]
  classificacao[i] = as.character( treino$species2[menor])
}
mean(classificacao == teste$species2) # acurácia do modelo (taxa de acerto).
```

```
[1] 0.9402985
```

```
# Retirando a nova coluna criada
treino = treino[,-9]
teste = teste[,-9]
```

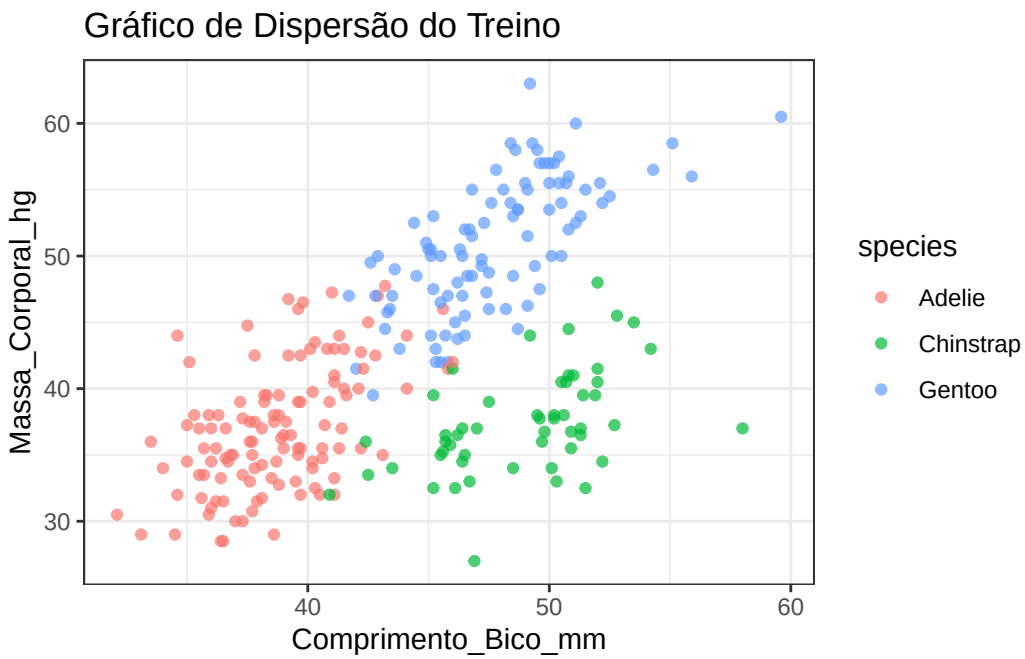
Portanto, fica provado que o modelo tem a menor acurácia para este caso por causa de Chinstrap.

Além disso, o caso Comprimento_Bico x Massa_Corporal teve 10% a menos de eficácia comparado ao caso Comprimento_Bico x Comprimento_Asa, mesmo que em ambos os casos tenham 3 grupos de cada espécie bem destacadas. Por que isso há essa diferença?

4.2 Normalização de Dados

Como são comparadas distâncias euclidianas, há um detalhe não mencionado anteriormente: As variáveis estão em escalas diferentes e até em unidades de medidas diferentes. Por exemplo, no caso Comprimento_Bico x Massa_Corporal tem escala de milhar no eixo y, e escala na dezena no eixo x, uma diferença consideravelmente grande. Aplicando no código, porém com a massa em Hectograma.

```
# Criando uma nova coluna para massa em hectograma
treino$massa_hg = treino$body_mass_g*0.01
teste$massa_hg = teste$body_mass_g*0.01

## Gráfico de Dispersão
ggplot(treino,aes(x = bill_length_mm,y = massa_hg))+
  geom_point(aes(colour =species),alpha =0.7)+
  theme_bw()+
  labs(x ='Comprimento_Bico_mm',
       y ='Massa_Corporal_hg',
       title ='Gráfico de Dispersão do Treino')
```




```
# Acurácia com Massa_corporal em nova escala
knn.amador2d(3,9,1)
```

```
[1] 0.9552239
```

```
# Removendo a nova coluna criada
treino = treino[,-9]
teste = teste[,-9]
```

Portanto, a acurácia do modelo foi afetada pela escala das variáveis. Para colocar todos os dados na mesma escala, será então necessário padronizar os dados, i.e., cada coluna numérica terá média 0 e desvio padrão igual a 1.

$$z = \frac{x - \bar{x}}{\sigma}$$

Sendo a função em R que aplica essa padronização em todos os dados de cada coluna é a função *scale(x, center = TRUE, scale = TRUE)*. Caso *scale = FALSE*, então não irá dividir pelo desvio padrão σ. Caso *center = FALSE*, então não irá centralizar as variáveis, i.e, não irá subtrair a média $\bar{x}$.

Padronizando as variáveis:

```
# Padronizando todas as variáveis numéricas, em que terão média 0 e desvio = 1
treino = treino |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))

teste = teste |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))

# Verificando como ficaram os números
head(treino[,c(1,3:6)])
```

```
      species bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
179    Gentoo      0.4331778    -1.1004732       1.064009450   0.6119779
14     Adelie     -1.1957645     1.6682152      -0.136152868  -0.5350582
195    Gentoo      0.5763815    -1.7548904       0.922813883   0.8909866
306 Chinstrap      1.7041107     1.3661764       0.287433832   0.3639701
118    Adelie     -0.4439450     0.5607398      -0.418544002   0.1159623
299 Chinstrap      1.3282010     0.9131183       0.005042699  -0.3180513
```

Testando agora os 3 últimos modelos da seção 4.1:

```
print(paste("Em Comprimento_Asa x Massa_Corporal, a acurácia foi de =",
            knn.amador2d(5,6,1)))
```

```
[1] "Em Comprimento_Asa x Massa_Corporal, a acurácia foi de = 0.761194029850746"
```

```
print(paste("Em Comprimento_Bico x Comprimento_Asa, a acurácia foi de =",
            knn.amador2d(3,5,1)))
```

```
[1] "Em Comprimento_Bico x Comprimento_Asa, a acurácia foi de = 0.955223880597015"
```

```
print(paste("Em Comprimento_Bico x Massa_Corporal, a acurácia foi de =",
            knn.amador2d(3,6,1)))
```

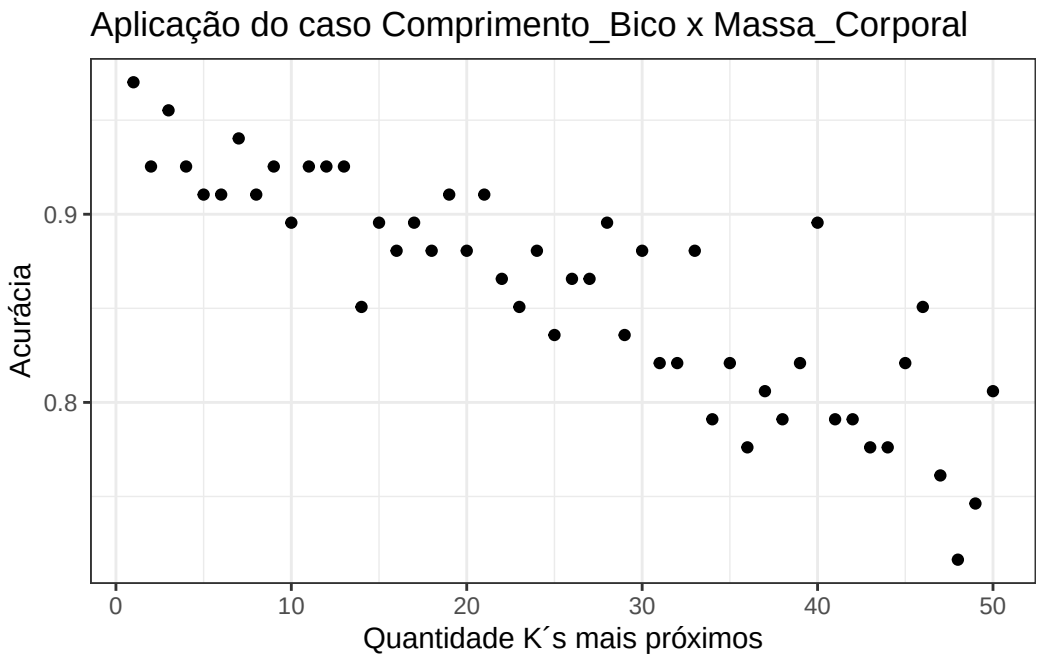
```
[1] "Em Comprimento_Bico x Massa_Corporal, a acurácia foi de = 0.970149253731343"
```

Acurácias bem mais elevadas comparadas à seção anterior.

4.3 Mais de 2 Variáveis e K $>$ 1

Agora, considerando mais de 1 ponto mais próximos ao novo penguim catalogado (k $>$ 1), é possível observar diferentes valores de acurácia para cada valor de K. Criando um gráfico de dispersão Acurácia x Quantidade de K para o caso em que teve a maior acurácia:

```
eixo_x = c() # Vetor vazio para armazenar os diferentes valores de k
eixo_y = c() # Vetor vazio para armazenar os diferentes valores da Acurácia

# Utilizando o laço para preencher os vetores
for (i in 1:50) {
  eixo_y[i] = knn.amador2d(3,6,i)
  eixo_x[i] = i
}

ggplot(mapping = aes(x = eixo_x, y = eixo_y))+
  geom_point()+
  theme_bw()+
  labs(x = 'Quantidade K´s mais próximos',
       y = 'Acurácia',
       title = 'Aplicação do caso Comprimento_Bico x Massa_Corporal')
```



Parece que quanto mais k´s são utilizados, menor fica a acurácia, sendo o maior k = 1 com a maior acurácia. Calculando uma possível correlação entre essas duas variáveis:

```
# Correlação de Pearson
cor(eixo_x,eixo_y)
```

```
[1] -0.8737928
```

Como o coeficiente de correlação é um valor próximo a -1 (-0.87), então quantidade de k´s e acurácia tem uma correlação forte negativa. Verificando a veracidade deste padrão para o caso de 3 dimensões.

```
knn.amador3d = function(x1,x2,x3,k){
  # Vetor para armazenar a classificação das observações do teste.
  classificacao = c()

  for(i in 1:nrow(teste)){
    distancias = c()
    for(j in 1:nrow(treino)){
      distancias[j] = sqrt((treino[j,x1]-teste[i,x1])^2 +
                           (treino[j,x2]-teste[i,x2])^2 +
                           (treino[j,x3]-teste[i,x3])^2)
    }

    menor = order(distancias)[k] # Irá pegar as z menores distancias

    # No treino, seleciona as z espécies com menor distancia
    # e armazena no vetor classificação como caracter
    classificacao[i] = as.character(treino$species[menor])
  }

  # Acurácia do modelo (taxa de acerto).
  acuracia = mean(classificacao == teste$species)

  return(acuracia)
}
```

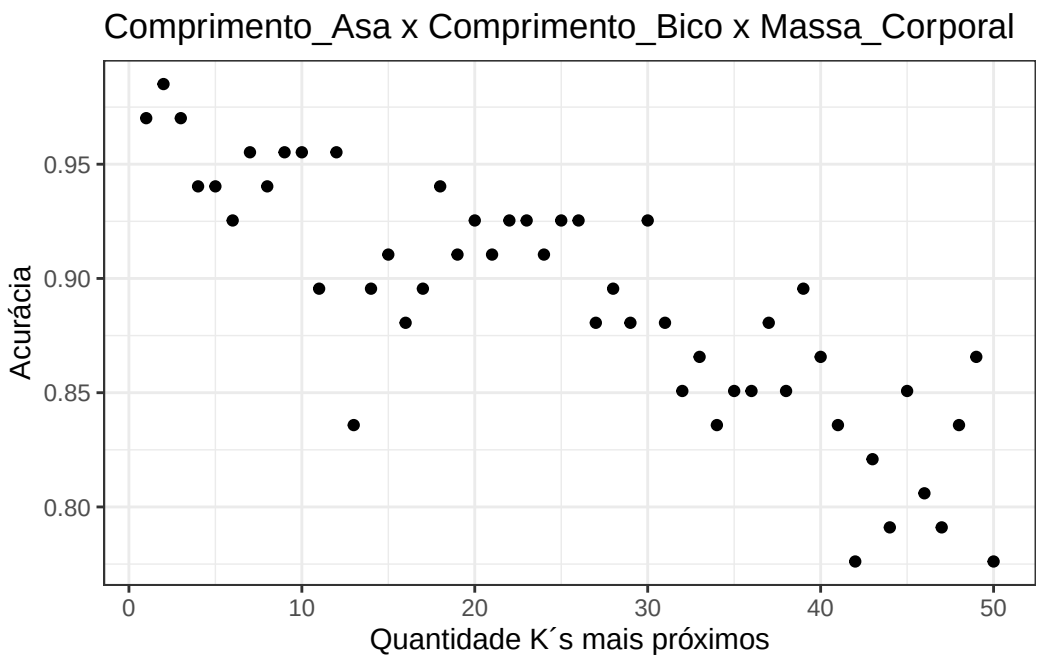
Testando para um espaço Comprimento_Asa x Comprimento_Bico x Massa_Corporal:

```
knn.amador3d(5,3,6,1)
```

```
[1] 0.9701493
```

Testando para diferente valores de k:

```
eixo_x = c() # Vetor vazio para armazenar os diferentes valores de k
eixo_y = c() # Vetor vazio para armazenar os diferentes valores da Acurácia

# Utilizando o laço para preencher os vetores
for (i in 1:50) {
  eixo_y[i] = knn.amador3d(5,3,6,i)
  eixo_x[i] = i
}

ggplot(mapping = aes(x = eixo_x, y = eixo_y))+
  geom_point()+
  theme_bw()+
  labs(x = 'Quantidade K´s mais próximos',
       y = 'Acurácia',
       title = 'Comprimento_Asa x Comprimento_Bico x Massa_Corporal')
```



E o padrão se repete mesmo para este caso tridimensional. O coeficiente de correlação será:

```r
# Correlação de Pearson
cor(eixo_x,eixo_y)
```

```
[1] -0.8417542
```

Novamente, um número muito próximo de -1, tendo uma alta correlação negativa.

E o k = 2 obteve a maior acurácia, sendo ela:

```r
# k = 2
knn.amador3d(5,3,6,2)
```

```
[1] 0.9850746
```

Por fim, testando para todas as variáveis numéricas:

```r
knn.amador4d = function(x1,x2,x3,x4,k){
  # Vetor para armazenar a classificação das observações do teste.
  classificacao = c()

  for(i in 1:nrow(teste)){
    distancias = c()
    for(j in 1:nrow(treino)){
      distancias[j] = sqrt((treino[j,x1]-teste[i,x1])^2 +
                           (treino[j,x2]-teste[i,x2])^2 +
                           (treino[j,x3]-teste[i,x3])^2 +
                           (treino[j,x4]-teste[i,x4])^2)
    }

    menor = order(distancias)[k] # Irá pegar as k menores distancias

    # No treino, seleciona as z espécies com menor distancia
    # e armazena no vetor classificação como caracter
    classificacao[i] = as.character(treino$species[menor])
  }

  # Acurácia do modelo (taxa de acerto).
  acuracia = mean(classificacao == teste$species)

  return(acuracia)
}
```

```
knn.amador4d(3,4,5,6,1)
```

```
[1] 1
```

Portanto, com essa seed, o modelo que teve a maior acurácia foi o *knn.amador4d()*. Por questões de hardware e escopo do trabalho, não serão feitos testes em outras amostras.

4.4 Melhorando ainda mais o Modelo

Como comentado na Seção 3.2, os pinguins Gentoo e Chisntrap estão em apenas 1 das ilhas. Esse fato será então incorporado no modelo em que obteve a menor acurácia dentre os apresentados: O knn bidimensional para as variáveis Comprimento da Asa e Massa Corporal:

```
classificacao = c()

for(i in 1:nrow(teste)){
  # Verifica se um pinguim é da ilha Torgersen, se verdadeiro será classificado
  # como Adelie
  if(str_detect(teste[i,2],pattern ="Torgersen")){
    classificacao[i] = "Adelie"
  }else{ # Se não, entrará no loop para calcular as distâncias
   distancias = c()
    for(j in 1:nrow(treino)){
      distancias[j] = sqrt((treino[j,5]-teste[i,5])^2 +
                           (treino[j,6]-teste[i,6])^2)
    }
  menor = order(distancias)[1]
  classificacao[i] = as.character(treino$species[menor])
  }

}

# Acurácia do modelo (taxa de acerto).
mean(classificacao == teste$species)
```

```
[1] 0.7910448
```

Como apenas os pinguins Adelies estavam na ilha Torgersen, então se um pinguim do teste ser da ilha Torgersen, ele será automaticamente classificado como Adelie. Porém, a ilha Torgersen é a ilha que tinha a minoria absoluta dos pinguins, então o modelo obteve apenas uma leve melhora na acurácia, indo de 76% para 79%.

5 Implementação da Floresta Aleatória

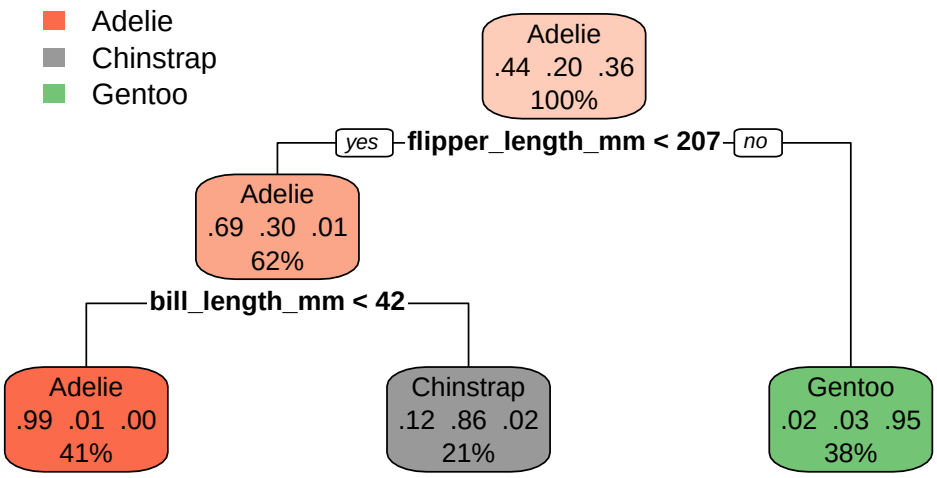
A árvore de decisão consiste em basicamente decidir caminhos diferentes baseado em variáveis presentes no data frame, criando linhas perpendiculares aos eixos das variáveis. Redefinindo o conjunto de treino e teste, agora utilizando uma função para alterar as seed facilmente.

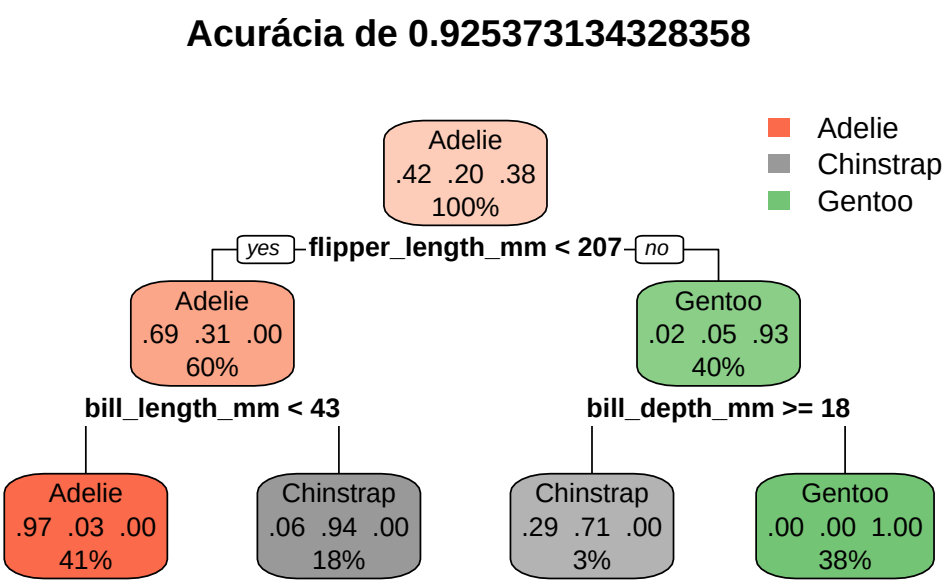
```
treino_teste <- function(dados, prop = 0.8, seed) {
  set.seed(seed)  # Agora com mais de uma seed

  n = nrow(dados) # Número total de linhas amarzenado em n

  # Índices de treino (amostra aleatória)
  indices = sample(1:n, size = round(prop * n))

  # Cria os conjuntos
  treino = dados[indices,]
  teste  = dados[-indices,]

  return(list(treino = treino, teste = teste))
}
```

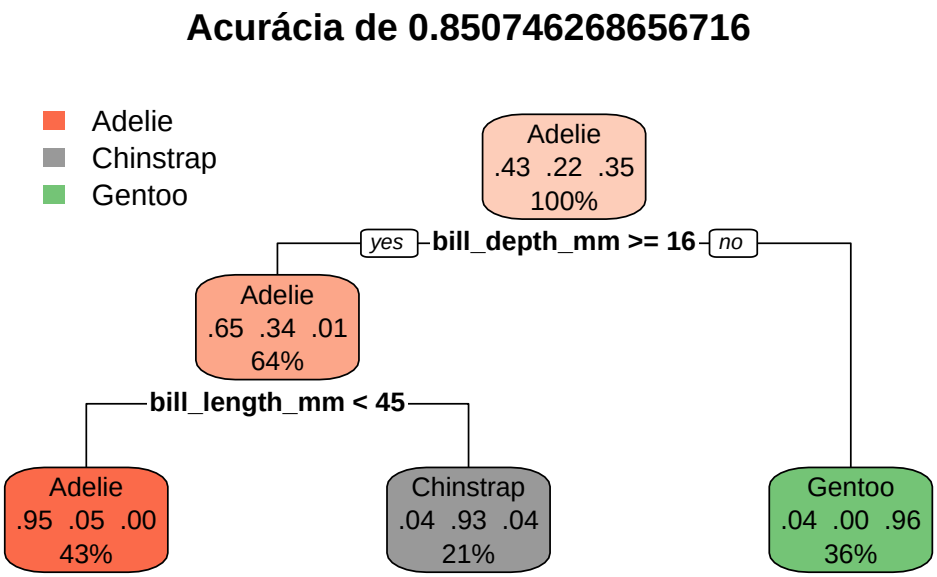
Primeiramento, testando o modelo para uma árvore apenas, na mesma seed do modelo KNN e com duas outras seeds:

```
acuracias = c() # Vetor para amazernar as 3 acurácias
vetor = c(123,200,1234) # Lista dos 3 elementos a percorrer no for

for (i in vetor) {

  # Treino e Teste recebem a função em suas colunas respectivas
  treino = treino_teste(dados = df, seed = i)$treino # Sendo i o elemento de vetor
  teste = treino_teste(dados = df,seed = i)$teste

  # Cria a árvore de decisão apenas nas variáveis numéricas
  arvore = rpart(formula = species ~ bill_length_mm + bill_depth_mm
                 + flipper_length_mm + body_mass_g,
                 data = treino,
                 method = "class")

  # Armazena a previsão da árvore
  previsao = predict(arvore, newdata = teste, type = "class")
```

```
  # Armazena a taxa de acerto de cada seed
  acuracias[i] = mean(previsao == teste$species)

  # Titulo de cada plot de ávore
  titulo = paste("Acurácia de", acuracias[i])

  rpart.plot(arvore, main = titulo)
}
```







Como pode ser observado, este exemplo de apenas 3 árvores apresentou uma grande variabilidade na estrutura e acurácia de cada uma, sendo a menor delas, a seed 1234, alcançando uma taxa de acerto de 85%. O mesmo não acontece para o KNN.

```
acuracias = c()
for (i in vetor) {

  treino = treino_teste(dados = df, seed = i)$treino
  teste = treino_teste(dados = df,seed = i)$teste

  print(paste('A acurácia na Seed',i,'foi de',knn.amador2d(3,4,1)))
}
```

```
[1] "A acurácia na Seed 123 foi de 0.925373134328358"
[1] "A acurácia na Seed 200 foi de 0.940298507462687"
[1] "A acurácia na Seed 1234 foi de 0.940298507462687"
```

Para resolver o problema da variabilidade da árvore de decisão, será implementado o modelo de Floresta Aleatória para diferentes números de ávores.

5.1 Floresta de 500 árvores

A floresta de 500 é *default* na função *randomForest()* do R. Testando sua acurácia para as mesmas seeds testadas anteriormente:

```
acuracias = c()
k = 1
vetor = c(123,1234, 200)

for (i in vetor) {

  # Treino e Teste recebem a função em suas colunas respectivas
  treino = treino_teste(dados = df, seed = i)$treino # Sendo i o elemento de vetor
  teste = treino_teste(dados = df,seed = i)$teste

  floresta500 = randomForest(formula = species ~ bill_length_mm + bill_depth_mm
                             + flipper_length_mm + body_mass_g,
                             data = treino, method = "class")

  previsao500 = predict(floresta500, newdata = teste, type = "class")
  acuracias[k] = mean(previsao500 == teste$species)

  print(paste('A acurácia na Seed',i,'foi de', acuracias[k]))
  k = k + 1
}
```

```
[1] "A acurácia na Seed 123 foi de 0.985074626865672"
[1] "A acurácia na Seed 1234 foi de 1"
[1] "A acurácia na Seed 200 foi de 0.955223880597015"
```

Anteriormente, a acurácia do modelo a acurácia do modelo de árvore na seed 1234 foi de 85%, i.e, o modelo classificou apenas 85% dos dados no conjunto teste corretamente. Porém, o modelo de floresta na mesma seed apresentou uma taxa de acerto superior, chegando a 100%.

Porém, como demonstrado na análise exploratória de dados, há um desequilíbrio na frequência das espécies de pinguins, sendo a espécie Gentoo com apenas 20% das observações. Como a acurácia não é suficiente para verificar a taxa de acerto por espécie, será feita uma análise mais aprofundada utilizando a matriz de confusão do modelo da seed de 200.

```
# Tabela de dados reais x previsto
table(teste$species, previsao500)
```

```
           previsao500
            Adelie Chinstrap Gentoo
  Adelie        32         2      0
  Chinstrap      0        14      0
  Gentoo         0         1     18
```

Sendo as linhas são os dados reais e as colunas são os dados previstos pelo modelo. No caso da previsão de Chinstrap, teve 3 falsos positivos, sendo a acurácia do modelo de Chinstrap em $14/17 = 0,824$, ou seja, dos das 17 previsões do Chinstrap, apenas 14 eram realmente Chinstrap, apenas 82% serão Chinstrap realmente. Como a taxa de falso positivo não se apresentou tão alta, e como neste caso do trabalho não há problema em ter falsos positivos para espécies, então não será feito um tratamento adequado para dados desbalanceados como apresentados em sala de aula.

5.2 Floresta de 2000 ávores

Fazendo para 2000 árvores:

```
acuracias = c()
k = 1
vetor = c(123,1234, 200)

for (i in vetor) {

  # Treino e Teste recebem a função em suas colunas respectivas
```

```
  treino = treino_teste(dados = df, seed = i)$treino # Sendo i o elemento de vetor
  teste = treino_teste(dados = df,seed = i)$teste

  floresta2000 = randomForest(formula = species ~ bill_length_mm + bill_depth_mm
                              + flipper_length_mm + body_mass_g,
                              data = treino,
                              method = "class", ntree=2000) # Para 2000 arvores

  previsao2000 = predict(floresta2000, newdata = teste, type = "class")
  acuracias[k] = mean(previsao2000 == teste$species)

  print(paste('A acurácia na Seed',i,'foi de', acuracias[k]))
  k = k + 1
}
```

```
[1] "A acurácia na Seed 123 foi de 1"
[1] "A acurácia na Seed 1234 foi de 1"
[1] "A acurácia na Seed 200 foi de 0.955223880597015"
```

Como pode ser verificado, houve um aumento na acurácia do modelo na primeira Seed. Porém, para o modelo na seed de 200 ainda permaneceu a mesma acurácia. Verificando detalhadamente por meio de sua matriz de confusão:

```
table(teste$species, previsao2000)
```

```
           previsao2000
            Adelie Chinstrap Gentoo
  Adelie        32         2      0
  Chinstrap      0        14      0
  Gentoo         0         1     18
```

Obtendo o mesmo padrão da matriz de confusão para o caso de 500 árvores.

5.3 Floresta de 100 e 50 árvores

```
acuracias = c()
k = 1
vetor = c(200,1234, 123)
```

```r
for (i in vetor) {

  # Treino e Teste recebem a função em suas colunas respectivas
  treino = treino_teste(dados = df, seed = i)$treino # Sendo i o elemento de vetor
  teste = treino_teste(dados = df,seed = i)$teste

  floresta100 = randomForest(formula = species ~ bill_length_mm + bill_depth_mm
                             + flipper_length_mm + body_mass_g,
                             data = treino,
                             method = "class", ntree=100) # Para 100 arvores

  previsao100 = predict(floresta100, newdata = teste, type = "class")
  acuracias[k] = mean(previsao100 == teste$species)

  print(paste('A acurácia na Seed',i,'foi de', acuracias[k]))
  k = k + 1
}
```

```
[1] "A acurácia na Seed 200 foi de 0.955223880597015"
[1] "A acurácia na Seed 1234 foi de 1"
[1] "A acurácia na Seed 123 foi de 0.985074626865672"
```

E testando as acurácias para 50 árvores:

```r
acuracias = c()
k = 1
vetor = c(200,123, 1234)

for (i in vetor) {

  # Treino e Teste recebem a função em suas colunas respectivas
  treino = treino_teste(dados = df, seed = i)$treino # Sendo i o elemento de vetor
  teste = treino_teste(dados = df,seed = i)$teste

  floresta50 = randomForest(formula = species ~ bill_length_mm + bill_depth_mm
                             + flipper_length_mm + body_mass_g,
                             data = treino,
                             method = "class", ntree=50) # Para 100 arvores

  previsao50 = predict(floresta50, newdata = teste, type = "class")
  acuracias[k] = mean(previsao50 == teste$species)
```

```
  print(paste('A acurácia na Seed',i,'foi de', acuracias[k]))
  k = k + 1
}
```

```
[1] "A acurácia na Seed 200 foi de 0.955223880597015"
[1] "A acurácia na Seed 123 foi de 0.985074626865672"
[1] "A acurácia na Seed 1234 foi de 0.985074626865672"
```

Verificando a matriz de confusão

```
table(teste$species, previsao50)
```

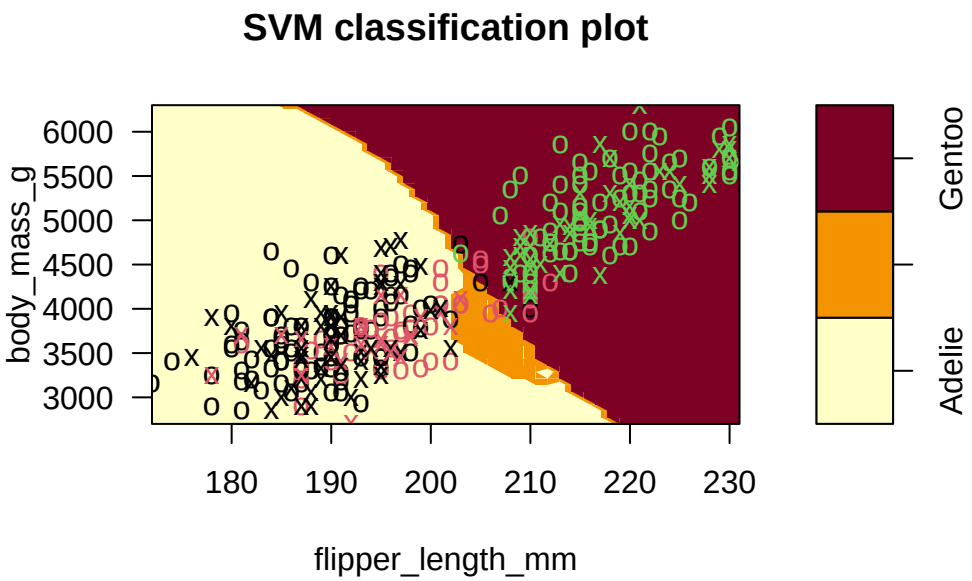
```
           previsao50
            Adelie Chinstrap Gentoo
  Adelie        31         1      0
  Chinstrap      0        10      0
  Gentoo         0         0     25
```

Para modelos com 50 árvores, a acurácia decaiu para o modelo na Seed 1234. E como pode ser observado, apenas foi adicionado um falso positivo de Chinstrap. A pouca presença dessa espécie no conjunto de dados está afetando o modelo de diferentes testes. Parece ser recomendável manter o número de árvores maior que 100, mas sem passar de 1000, pois a acurácia não se altera tanto em detrimento do custo computacional.

6 Implementação de SVM

O objetivo do SVM é encontrar a melhor fronteira de separação (hiperplano) entre diferentes classes no espaço dos dados. Testando o modelo de máquina de vetor de suporte de kernel default e em uma única *seed*, para as variáveis Comprimento de Asa e Massa Corporal, ou seja, para o caso que o modelo KNN teve a menor acurácia não passando de 79% de taxa de acerto, como visto na Seção 4.1 e 4.4:

```
treino = treino_teste(dados = df, seed = 123)$treino
teste = treino_teste(dados = df,seed = 123)$teste

# Utilizando o kernel radial default para construir o modelo
modelo_svm = svm(formula = species~ flipper_length_mm + body_mass_g, data = treino)

# Reigao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Asa,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ flipper_length_mm)
```



```r
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.8059701
```

O kernel default do modelo SVM, o radial, cria fronteiras entre os dados com curvas flexíveis. Sua acurácia mais elevada ao modelo KNN (indo de 75% para 80%) para este caso provavelmente se deve ao fato de que o kernel do modelo aumenta a dimensão do plano, convertendo dados não separáveis em dados mais separáveis nessa dimensão superior. Ele é mais útil para problemas de separação não linear. Verificando a matriz de confusão para aprofundar em como o modelo acerta:

```r
# Matriz de confusão
table(teste$species, previsao)
```

```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        30         0      0
  Chinstrap     11         2      2
  Gentoo         0         0     22
```

Como discutido anteriormente nos modelos KNN bidimensional, a espécie Chinstrap estava totalmente misturada na espécie Adelie neste gráfico. Como consequência, dos 15 pinguins que são da espécie Chinstrap, apenas 2 o modelo conseguiu inferir que era Chinstrap, em resumo: Dado que o modelo previu ser Adelie, a probabilidade do Pinguim ser realmente Adelie é de $30/41 = 0,73$, sendo a probabilidade do complementar ser exatamente a probabilidade do pinguim ser Chinstrap. E caso o pinguim seja Chinstrap, o modelo tem a probabilidade de $2/15 = 0.13$ de inferir que o pinguim seja Chinstrap. Apesar de ter aumentado a acurácia do modelo, a previsão por espécie, principalmente Chinstrap (a espécie de menor frequência), é o que tem a pior inferência sobre.

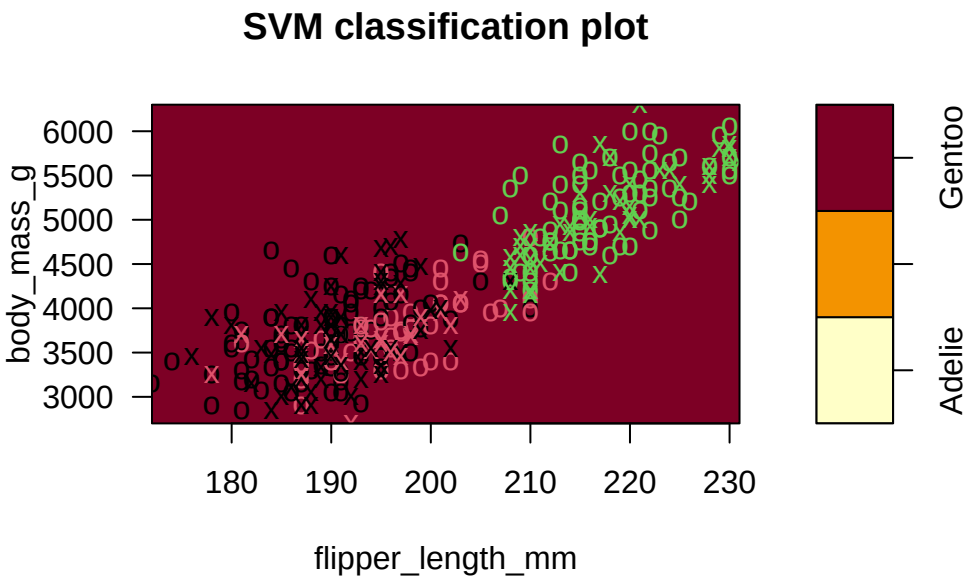
Assim como no modelo KNN houve uma melhora significativa quando se padronizou os dados, o mesmo acontece com o SVM? Será testado.

```r
# Padronizando todas as variáveis numéricas, em que terão média 0 e desvio = 1
treino = treino |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))
```

```r
teste = teste |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))
```

Testando o modelo novamente:

```r
# Utilizando o kernel radial default para construir o modelo
modelo_svm = svm(formula = species~ flipper_length_mm + body_mass_g, data = treino)

# Reigao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Asa,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ flipper_length_mm)
```




```r
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.8208955
```

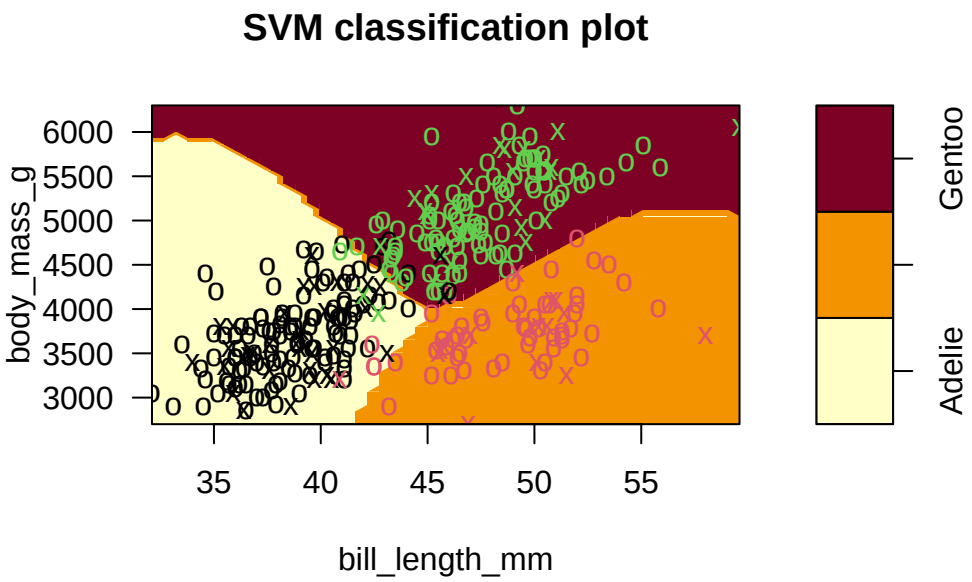
Uma melhora de 2% na acurácia total. Verificando pela matriz de confusão

```
# Matriz de confusão
table(teste$species, previsao)
```

```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        30         0      0
  Chinstrap     11         3      1
  Gentoo         0         0     22
```

Algumas propriedades se manteram, porém a probabilidade do modelo acertar dado que é Chinstrap aumentou de $0,13$ para $0,2$. Uma pequena melhora que por enquanto não justifica a padronização dos dados. Testando para um caso com menor Gini: Comprimento Bico e Massa Corporal.

```
treino = treino_teste(dados = df, seed = 123)$treino
teste = treino_teste(dados = df,seed = 123)$teste

# Utilizando o kernel radial default para construir o modelo
modelo_svm = svm(formula = species~ bill_length_mm + body_mass_g, data = treino)

# Regiao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Asa,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ bill_length_mm)
```




```
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.9253731
```

E verificando sua matriz de confusão

```
# Matriz de confusão
table(teste$species, previsao)
```

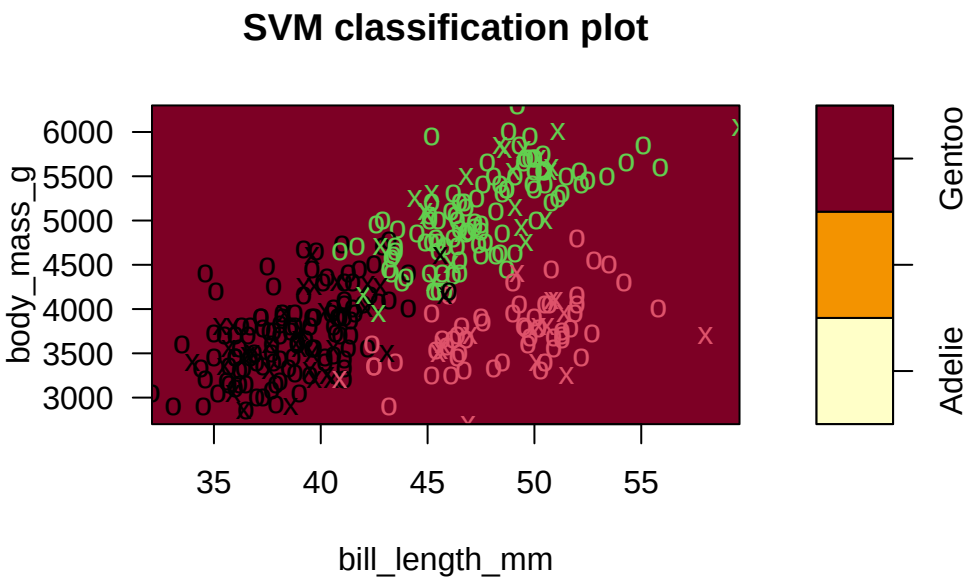
```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        26         0      4
  Chinstrap      0        14      1
  Gentoo         0         0     22
```

Padronizando as variáveis novamente:

```
# Padronizando todas as variáveis numéricas, em que terão média 0 e desvio = 1
treino_padrao = treino |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))

teste_padrao = teste |>
  mutate(bill_length_mm = scale(bill_length_mm),
         bill_depth_mm = scale(bill_depth_mm),
         body_mass_g = scale(body_mass_g),
         flipper_length_mm = scale(flipper_length_mm))
```

E repetindo o teste

```
# Utilizando o kernel radial default para construir o modelo
modelo_svm = svm(formula = species~ bill_length_mm + body_mass_g, data = treino_padrao)

# Regiao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste_padrao, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Asa,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ bill_length_mm)
```




```
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.9402985
```

E sua matriz de confusão

```
# Matriz de confusão
table(teste$species, previsao)
```

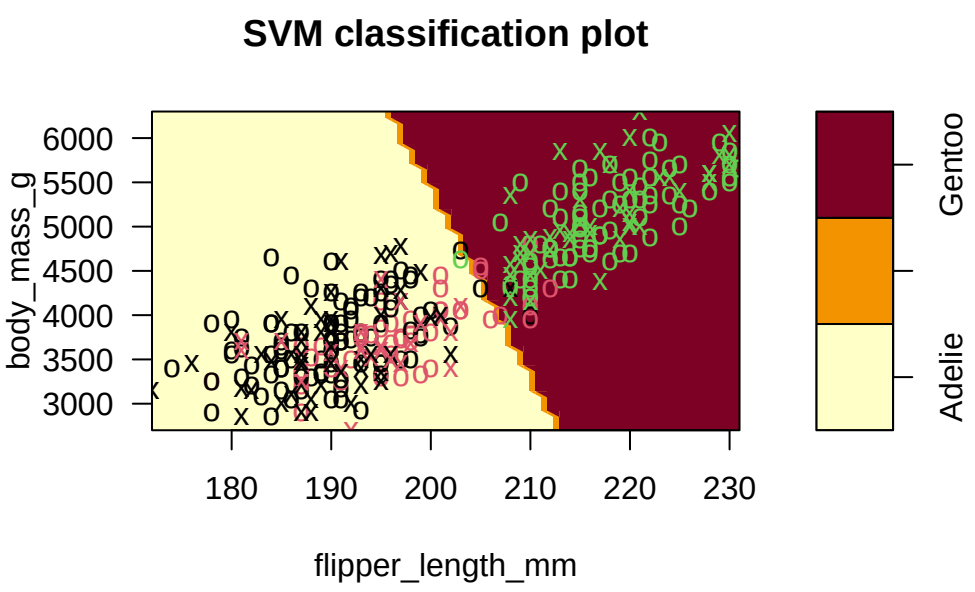
```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        27         0      3
  Chinstrap      0        14      1
  Gentoo         0         0     22
```

Como a acurácia total teve um aumento de 2% e a matriz de confusão teve apenas uma leve melhora de 1 dígito, então aparentemente não é tão importante padronizar as variáveis numéricas assim como é no KNN, mesmo que o SVM também depende de medir distâncias dos dados suportes. Além disso, padronizar apresentou uma desvantagem para visualizar os gráficos bidimensionais.

6.1 Kernel Linear

Diferente da árvore de decisão, o SVM com kernal linear consegue criar uma reta separadora na diagnal também. Será testado para o pior caso no modelo KNN bidimensional, como descrito anteriormente: Comprimento Asa e Massa Corporal.

```
# Utilizando o kernel linear default para construir o modelo
modelo_svm = svm(formula = species ~ flipper_length_mm + body_mass_g,
                 data = treino,
                 kernel = "linear")

# Regiao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Bico,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ flipper_length_mm)
```




```
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.7761194
```

Como no caso do kernel linear apenas irá criar uma "reta" separadora de dados (um hiperplano), então o melhor caso o modelo encontrou foi apenas separar ao meio conjunto "Adelia + Chinstrap" e "Gentoo", não sendo capaz de fazer fronteiras flexíveis como no kernel radial. Com a acurácia menor que o modelo anterior (de 80% para 78%), será averiguando pela matriz de confusão:

```
# Matriz de confusão
table(teste$species, previsao)
```

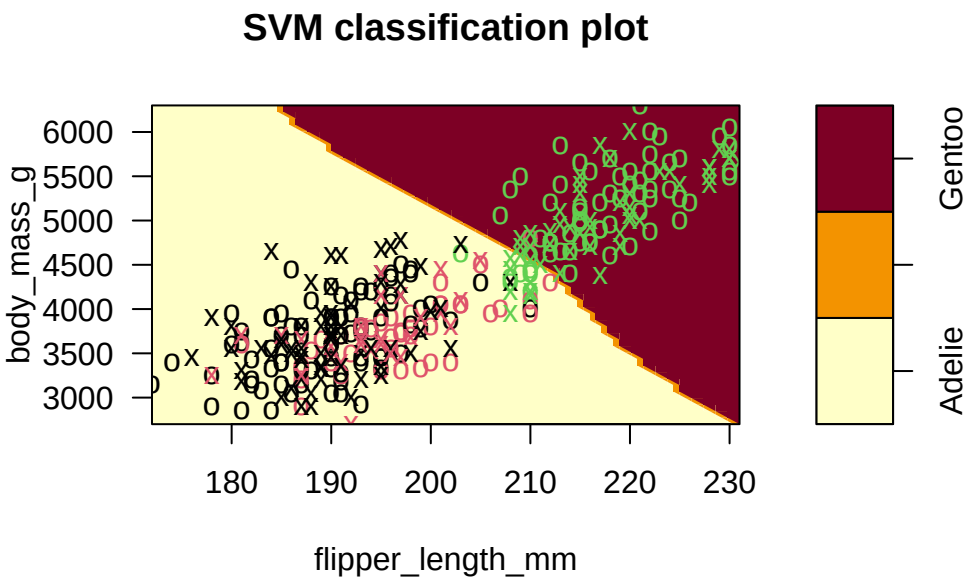
```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        30         0      0
  Chinstrap     13         0      2
  Gentoo         0         0     22
```

Dos 15 Chinstrap existentes, o modelo conseguiu classificiar 0 como Chinstrap. Confirmando o que é possível verificar de forma visual no gráfico: o modelo criou apenas duas regiões, uma de Adelie e outra de Gentoo. Como Chinstrap é uma espécie que, novamente, tem a menor ocorrência, então a acurácia não diminuiu abrubtamente.

6.2 Kernel Polinomial

Testanto para as mesmas variáveis descritas no kernel linear:

```
# Utilizando o kernel linear default para construir o modelo
modelo_svm = svm(formula = species ~ flipper_length_mm + body_mass_g,
                 data = treino,
                 kernel = "polynomial")

# Regiao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Plota o gráfico Massa_Coporal x Comprimento Bico,
# com as regiões feitas pelo modelo radial
plot(modelo_svm, df, body_mass_g ~ flipper_length_mm)
```




```
# Calculando sua acurácia
mean(previsao == teste$species)
```

```
[1] 0.761194
```

Como neste caso o modelo delimita as regiões utilizando polônmios, então há uma flexibilidade maior em como as regiões são delimitadas, comparado ao linear. Mas ainda sim, com o kernel polinomial o modelo não conseguiu criar um região para Chinstrap. Verificando pela matriz de confusão:

```
# Matriz de confusão
table(teste$species, previsao)
```

```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        30         0      0
  Chinstrap     15         0      0
  Gentoo         1         0     21
```

A tabela de confusão teve uma leve piora com um novo falso positivo de Adelie para um pinguim Gentoo.

6.3 Comparando com a Floresta

Como já foi comparado ao modelo KNN nas seções anteriores, será então comparado ao modelo da Floresta aleatória com 200 árvores, aplicando em todas as variáveis numéricas.

```
# Utilizando o kernel radial default para construir o modelo
modelo_svm = svm(formula = species~ bill_length_mm + bill_depth_mm
                              + flipper_length_mm + body_mass_g, data = treino)

# Reigao de previsao do modelo
previsao = predict(modelo_svm, newdata = teste, type = "class")

# Calculando sua acurácia do SVM
mean(previsao == teste$species)
```

```
[1] 1
```

E sua matriz de confusão:

```
table(teste$species, previsao)
```

```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        30         0      0
  Chinstrap      0        15      0
  Gentoo         0         0     22
```

E para a floresta:

```
# Floresta para Comprimento Asa e Massa Corporal
floresta100 = randomForest(formula = species ~  bill_length_mm + bill_depth_mm
                             + flipper_length_mm + body_mass_g,
                           data = treino,
                           method = "class", ntree=200) # Para 200 arvores


previsao = predict(floresta100, newdata = teste, type = "class")

# Acurácia da floresta aleatória
mean(previsao == teste$species)
```

```
[1] 0.9850746
```

E sua matriz de confusão:

```
table(teste$species, previsao)
```

```
           previsao
            Adelie Chinstrap Gentoo
  Adelie        29         1      0
  Chinstrap      0        15      0
  Gentoo         0         0     22
```

Sendo ambos os modelos com acurácias bem próximas, porém com a árvore infereindo um pinguim ser Chinstrap, sendo que ele é Adelie. Como a árvore é menos "flexível" que comparado ao svm de kernel linear, então é condizente a floresta ser um pouco mais sensível a espécies como Chinstrap. Este mesmo teste no KNN a acurácia foi de 100%.

7 Conclusão

Neste estudo foi possível verificar as diferentes características de cada espécie de pinguim do arquipélogo de Palmer. Como as espécies Chinstrap e Gentoo aparentam serem exlusivos de suas ilhas, a maioria das espécies catalogados são Adelie, sendo esta espécie bem distribuída pelo arquipélogo, além de possuir as menores medidas corporais no geral. A medida que tinha a menor variação em todas as espécies era o comprimento da asa, e não há desequelíbrio entre o número de fêmeas e machos.

Os modelos de KNN que apresentaram maior acurácia foram os modelos de mais de duas dimensões, sendo o modelo aplicado no espaço Comprimento_Asa x Comprimento_Bico x Massa_Corporal com acurácia de 98% (quando k=2), e e o modelo aplicado a todas as variáveis numéricas com eficácia igual 1 (quando k=1). O modelo que menos apresentou acurácia foi o bidimensional aplicado em Comprimento_Asa e Massa_Corporal, pois o grupo de Chinstrap não aparecia uma espécie destacada como um grupo por si mesma, mas sim se misturando com os dados da outra espécie. Modelos que aumentaram o valor de k também apresentavam acurácia descrescendo confome k crescia, tendo uma correlação negativa alta entre as variáveis acurácia e quantidade de k´s em 2 casos. Aparantemente, quanto mais variáveis for comparado, mais acurado ficou o modelo KNN.

Uma desvantagem evidente do KNN, é seu elevado custo computacional. Houve uma tentativa de melhora do modelo no caso Comprimento_Asa x Massa_Corporal na Seção 4.4, levando em conta a distribuição de Adelie em todas as ilhas, ocorrendo Adelie em uma ilha que não há nenhuma espécie de pinguim. Apesar da melhora na acurácia, ainda permaneceu como o modelo que mais errava dentre os criados, e caso tenha ocorrido imigração de Chinstrap e Gentoo em todas as ilhas do arquipélogo, a acurácia diminuirá ainda mais.

Para tentar contornar o problema apresentado na Seção 4.1 e 4.4, para o caso de Comprimento Asa e Massa Corporal, modelos de SVM com diferentes Kernel, sendo o Kernel radial o que obteve a maior acurácia, apesar de ainda apresentar dificuldades em inferir sobre a espécie Chinstrap como demonstrado pela matriz de confusão. Os SVM com kernels diferentes do radial apresentaram acurácia parecida ao modelo KNN bidimensional, junto de dificuldades de delimitar uma região para Chinstrap. Juntamente a isso, padronizar as variáveis numéricas não surtiu tanto efeito na acurácia dos modelos.

Além disso, foi explorado árvores e florestas aleatórias, demonstrando com diferentes Seeds como a árvore obteve variância alta e como a floresta aleatória corrigiu este problema. E a floresta aleatória foi testada diferentes quantidades de árvores, verificando ponderando entre acurácia e custo computacional, sendo que aumentar o número de árvores acima de mil não melhora tanto a taxa de acerto do modelo. E caso faça com o número de árvores menores que 100, começa a afetar a acurácia do modelo. Também foram discutidas a matriz de confusão de cada teste.

Por fim, foi testado todos os modelos (KNN, Floresta Aleatória e SVM) com todas as variáveis numéricas, sendo a floresta aleatória com a menor acurácia dentre os modelos apresentados (98%).

8 Apêndice A

A notação *Big O* diz respeito à escalabilidade do algortimo, i.e., como o algoritmo escala de acordo com o tamanho to input. *Big O* é uma forma de denotar desempenho do algoritmo, e não necessariamente à perfomance do algoritmo. E para analisar perfomance, será necessário uma **análise assintótica** do algoritmo. Há dois tipos de complexidade: a temporal, que diz respeito ao tempo de execução ou número de iterações, e a espcial, que diz respeito ao espaço da memória utilizado [5].

Um bom exemplo seria a função de encontrar o menor valor de um vetor ou array. Sempre considerando o pior caso [4], a função percorrerá todo o algoritmo até encontrar o menor valor. Caso o vetor tenha 10 elementos, terá 10 iterações, e caso tenha 20 elementos, terá 20 iterações, portando sua Complexidade Temporal será linear: $O(N)$. Em um array de tamanho 10 ou 20, terá apenas 1 valor que será o menor de todos, 1 valor que será armazenado na memória, portanto, sua Complexidade Espacial será constante $O(1)$.

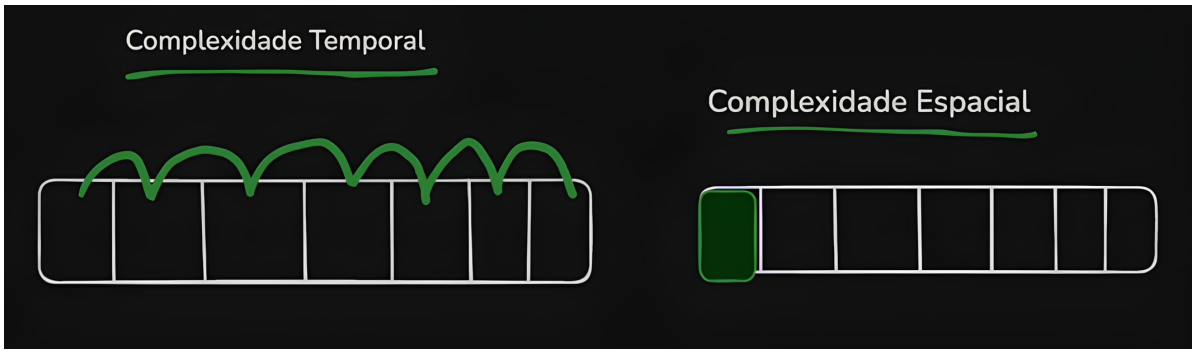


Figure 2: Os dois tipos de complexidade

9 Referências

[1] Imagem da Figura dos Pinguins: https://www.gabemednick.com/post/penguin/

[2] JAMES, Gareth et al. **An introduction to statistical learning.** New York: springer, 2013.

[3] https://www.r-bloggers.com/2021/12/how-to-use-the-scale-function-in-r/

[4] BHARGAVA, Aditya Y. **Entendendo Algoritmos: Um guia ilustrado para programadores e outros curiosos. Novatec Editora**, 2018.

[5] CORMEN, Thomas H. et al. **Introduction to algorithms.** MIT press, 2022.