TP 6: Clasificación

Grupo 12: Barragán, Rossi, Fontana Walser

2022-11-10

Primera Parte: Distinción entre venta y alquiler en base a identifcación de fronteras

Carga de Librerias, Lectura del dataset y su estructura

```
require(GGally)
```

```
## Loading required package: GGally
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
require(tidyverse) # entorno tidy
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ✔ purrr   0.3.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
require(dplyr) # manejo de datos
require(sf) # simple features
```

```
## Loading required package: sf
```

```
## Warning: package 'sf' was built under R version 4.2.2
```

```
## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
require(ggmap) # mapas estaticos
```

```
## Loading required package: ggmap
```

```
## Warning: package 'ggmap' was built under R version 4.2.2
```

```
## ℹ Google's Terms of Service: <https://mapsplatform.google.com>
## ℹ Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```
require(hrbrthemes) # aesthetics
```

```
## Loading required package: hrbrthemes
```

```
## Warning: package 'hrbrthemes' was built under R version 4.2.2
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
require(ggridges) # density plot with ridges
```

```
## Loading required package: ggridges
```

```
## Warning: package 'ggridges' was built under R version 4.2.2
```

```
require(rje) # función powerSet
```

```
## Loading required package: rje
```

```
## Warning: package 'rje' was built under R version 4.2.2
```

```
##
## Attaching package: 'rje'
##
## The following object is masked from 'package:dplyr':
##
##     last
```

```
dataset <- read_csv("ar_properties.csv")
```

```
## Rows: 388891 Columns: 24
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (12): id, ad_type, l1, l2, l3, l4, l5, currency, price_period, title, p...
## dbl   (8): lat, lon, rooms, bedrooms, bathrooms, surface_total, surface_cove...
## lgl   (1): l6
## date  (3): start_date, end_date, created_on
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Aplicando filtros

Nos quedamos con las variables de interés y filtramos los datos faltantes.

```
  # Creamos las variables lprice, lsurf y es_venta para analizar despues
datos <- dataset %>%
  filter(currency == "USD", l1 == "Argentina", l2 == "Capital Federal", l3 %in% c("Centro / Microcentro", "Puerto
Madero", "Boedo")) %>%
  mutate(lprice = log(price,base = 10), lsurf = log(surface_total,base = 10), es_venta = operation_type == 'Venta
') %>%
  filter(!is.na(lsurf), lsurf != "-inf", lsurf != "-Inf")
head(datos)
```

```
## # A tibble: 6 × 27
##   id      ad_type start_date end_date   created_on   lat   lon l1    l2    l3
##   <chr>   <chr>   <date>     <date>     <date>     <dbl> <dbl> <chr> <chr> <chr>
## 1 HdjpKr… Propie… 2019-04-14 2019-04-15 2019-04-14 -34.6 -58.4 Arge… Capi… Boedo
## 2 oO3WJx… Propie… 2019-04-14 2019-04-29 2019-04-14 -34.6 -58.4 Arge… Capi… Cent…
## 3 KAsKSt… Propie… 2019-04-14 9999-12-31 2019-04-14 -34.6 -58.4 Arge… Capi… Puer…
## 4 /HUjBB… Propie… 2019-04-14 2019-08-17 2019-04-14 -34.6 -58.4 Arge… Capi… Puer…
## 5 pMLGL7… Propie… 2019-04-14 2019-08-07 2019-04-14 -34.6 -58.4 Arge… Capi… Puer…
## 6 5hddYN… Propie… 2019-04-14 2019-07-10 2019-04-14 -34.6 -58.4 Arge… Capi… Cent…
## # … with 17 more variables: l4 <chr>, l5 <chr>, l6 <lgl>, rooms <dbl>,
## #   bedrooms <dbl>, bathrooms <dbl>, surface_total <dbl>,
## #   surface_covered <dbl>, price <dbl>, currency <chr>, price_period <chr>,
## #   title <chr>, property_type <chr>, operation_type <chr>, lprice <dbl>,
## #   lsurf <dbl>, es_venta <lgl>
```

1 - Análisis del precio y la superficie total

Empecemos haciendo histogramas del precio y la superficie, distinguiendo según la propiedad esté en venta o no. Graficamos la media de cada conjunto, y buscamos el punto de corte que maximiza la accuracy al querer clasificar los datos según el siguiente criterio: tras observar dónde se ubica la media de cada conjunto, si el precio de una propiedad se halla a la derecha del punto de corte “x”, diremos que está en venta, y si se halla a la izquierda, diremos que no está en venta.

```
esVentaMean <- mean(datos$lprice[datos$es_venta == TRUE])
noEsVentaMean <- mean(datos$lprice[datos$es_venta == FALSE])

esVentaMean
```

```
## [1] 5.453083
```

```
noEsVentaMean
```

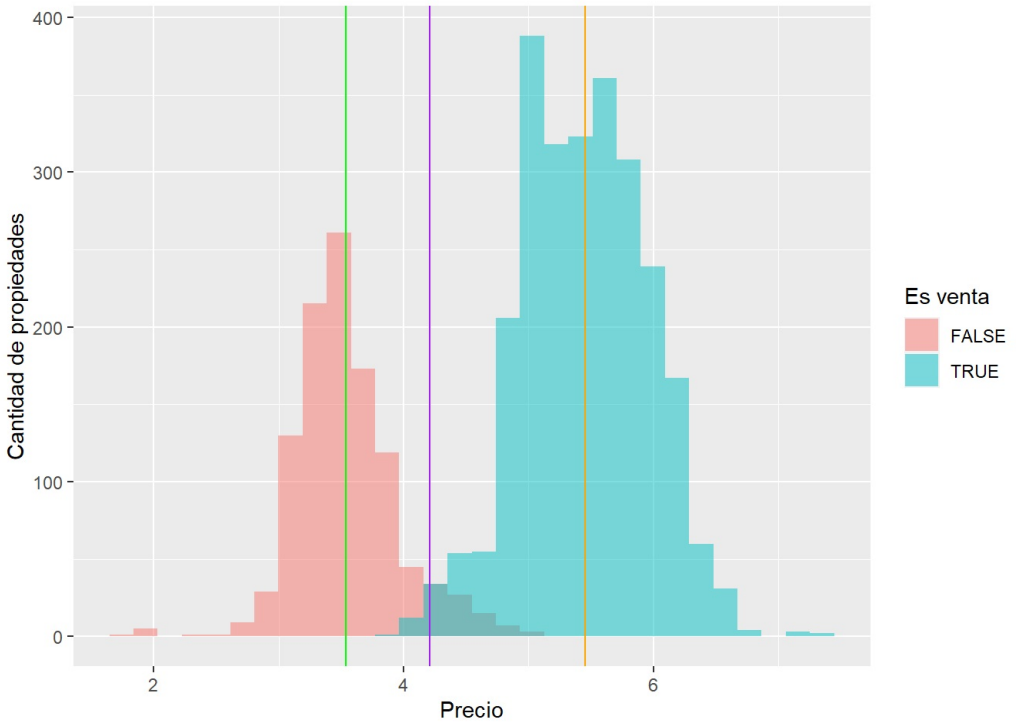
```
## [1] 3.536307
```

```
#Definimos la función que calcula la accuracy de la clasificación según el criterio elegido.
accuracyPrice <- function(x) {
  nObs <- length(datos$lprice)
  bien <- 0
  matriz <- matrix(NA, nrow = nObs, ncol = 2)
  for(i in 1:nObs) {
    price <- datos$lprice[i]
    if(price >= x) {
      if(datos$es_venta[i] == TRUE) {
        bien <- bien +1
      }
    } else {
      if(datos$es_venta[i] == FALSE) {
        bien <- bien +1
      }
    }
  }
  return (bien/nObs)
}

ptoCorte <- optimize(accuracyPrice, c(noEsVentaMean, esVentaMean), maximum = TRUE)$maximum

accuracy_precio <- accuracyPrice(ptoCorte)
print(accuracy_precio)
```

```
## [1] 0.9747322
```

```
#Graficamos, en violeta, el punto de corte obtenido, y en naranja y verde, las medias del precio de las propiedad
es que están en venta
#y de las que no, respectivamente.
ggplot(data = datos,
  mapping = aes(
  x = lprice,
  fill = es_venta)) +
  labs(x = "Precio", y = "Cantidad de propiedades", fill = "Es venta") +
  geom_histogram(alpha = 0.5, position = "identity") +
  geom_vline(aes(xintercept = ptoCorte), color = I('purple')) +
  geom_vline(aes(xintercept = esVentaMean), color = I('orange')) +
  geom_vline(aes(xintercept = noEsVentaMean), color = I('green'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
esVentaMean <- mean(datos$lsurf[datos$es_venta == TRUE])

noEsVentaMean <- mean(datos$lsurf[datos$es_venta == FALSE])

esVentaMean
```

```
## [1] 2.008843
```

```
noEsVentaMean
```

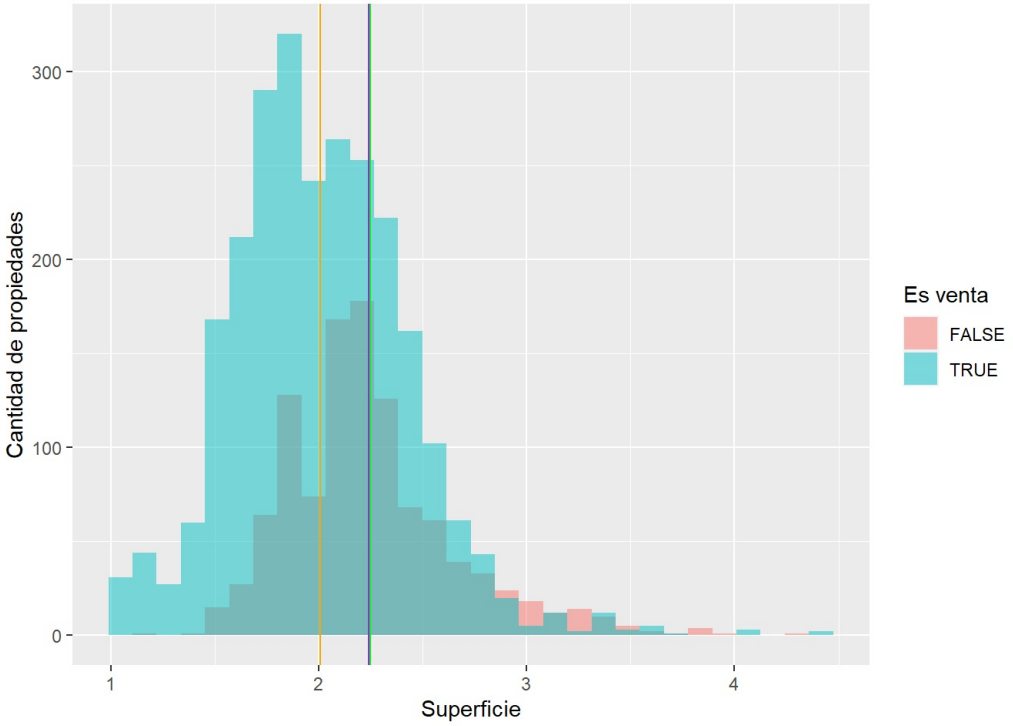
```
## [1] 2.246127
```

```
#Definimos la función que calcula la accuracy de la clasificación según el criterio elegido.
accuracySurf <- function(x) {
  nObs <- length(datos$lsurf)
  bien <- 0
  for(i in 1:nObs) {
    surf <- datos$lsurf[i]
    if(surf >= x) {
      if(datos$es_venta[i] == FALSE) {
        bien <- bien +1
      }
    } else {
      if(datos$es_venta[i] == TRUE) {
        bien <- bien +1
      }
    }
  }
  return (bien/nObs)
}

ptoCorte <- optimize(accuracySurf, c(esVentaMean, noEsVentaMean), maximum = TRUE)$maximum

accuracy_superficie <- accuracySurf(ptoCorte)
print(accuracy_superficie)
```

```
## [1] 0.6380115
```

```
#Graficamos, en violeta, el punto de corte obtenido, y en naranja y verde, las medias de la superficie de las pro
piedades que están en venta y de las que no, respectivamente.
ggplot(data = datos,
  mapping = aes(
  x = lsurf,
  fill = es_venta)) +
  labs(x = "Superficie", y = "Cantidad de propiedades", fill = "Es venta") +
  geom_histogram(alpha = 0.5, position = "identity") +
  geom_vline(aes(xintercept = ptoCorte), color = I('purple')) +
  geom_vline(aes(xintercept = esVentaMean), color = I('orange')) +
  geom_vline(aes(xintercept = noEsVentaMean), color = I('green'))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



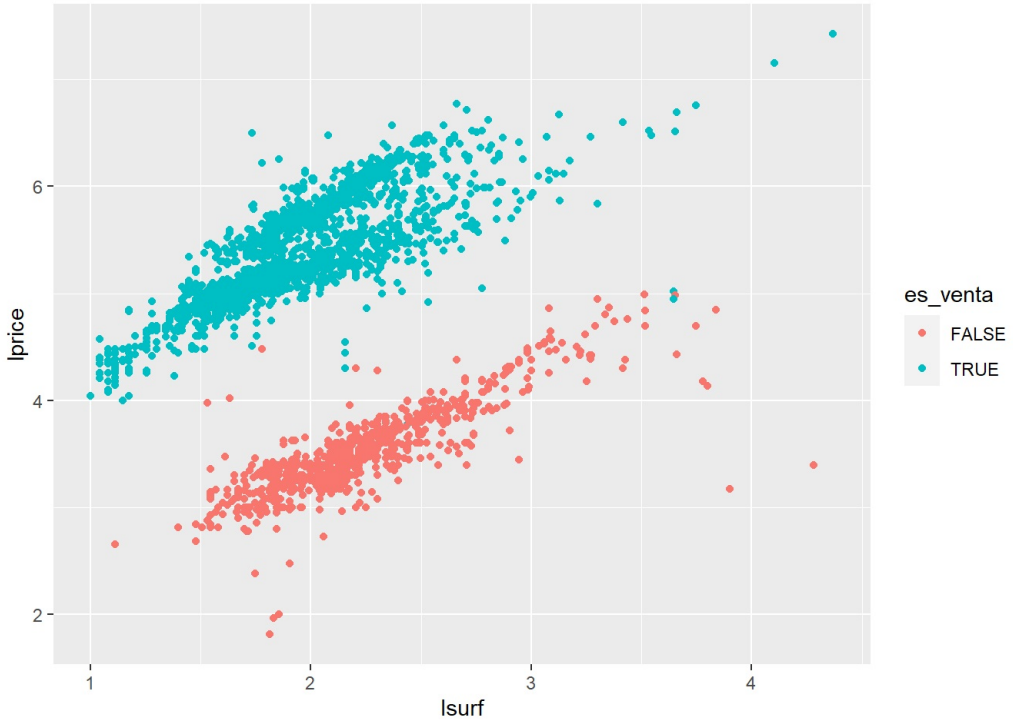
Notemos que los puntos de corte obtenidos en base al precio de las propiedades primero y considerando la superficie de las propiedades posteriormente, son distintos.

Para el caso particular en que consideramos la superficie, podemos ver que el histograma correspondiente a las propiedades que no están en venta queda casi totalmente por debajo del construido con aquellas que sí lo están. Por lo tanto, no consideramos que la superficie por sí sola sea una buena variable en base a la cual distinguir si una propiedad está en venta o no.

Por el contrario, si observamos el histograma construido con el precio de las propiedades, podemos ver que existen dos gráficos marcadamente separados, y por lo tanto el criterio escogido para clasificar cobra más sentido, es decir que el punto de corte obtenido en base al precio de una propiedad debe ser una mejor medida para decidir si la misma se encuentra en venta o no.

2 - Mejorando la clasificación

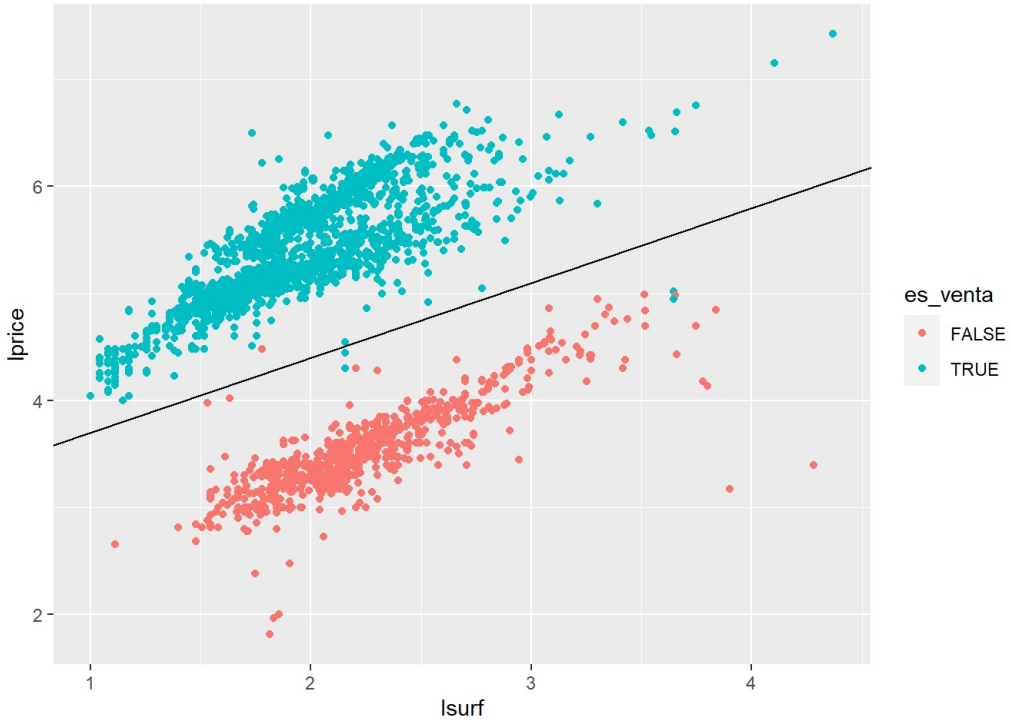
```
ggplot(data = datos, mapping = aes(x = lsurf, y = lprice, color = es_venta)) + geom_point()
```




```
accuracy <- function(pendiente_ordenada) {
  nObs <- length(datos)
  bien <- 0
  pendiente <- pendiente_ordenada[1]
  ordenada <- pendiente_ordenada[2]
  for(i in 1:nObs) {
    surf <- datos$lsurf[i]
    price <- datos$lprice[i]
    punto <- c(surf, price)
    puntoRecta <- c(surf, pendiente*surf+ordenada)
    if(punto[2] <= puntoRecta[2]) {
      if(datos$es_venta[i] == FALSE) {
        bien <- bien +1
      }
    } else {
      if(datos$es_venta[i] == TRUE) {
        bien <- bien +1
      }
    }
  }
  return (bien/nObs)
}

vals <- optim(par = c(1,3), fn = accuracy, control = list(fnscale = -1)) # fnscale en -1 hace que el optim busque el maximo y no el minimo, y la semilla seleccionada a ojo es el vector (1,3)
pendiente <- vals$par[1]
ordenada <- vals$par[2]

accuracy_lprice_lsurf <- accuracy(vals$par)
print(accuracy_lprice_lsurf)
```

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

```
ggplot(data = datos, mapping = aes(x = lsurf, y = lprice, color = es_venta)) +
  geom_point() +
  geom_abline(color = "black", slope = pendiente, intercept = ordenada)
```



Imprimimos la accuracy obtenida al clasificar unicamente con la variable precio y la obtenida al clasificar con ambas variables y nos fijamos cual es mejor

```
accuracy_precio
```

```
## [1] 0.9747322
```

```
accuracy_lprice_lsurf
```

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

Vemos que mejora la clasificacion utilizando ambas variables conjuntamente dado que la accuracy arrojada es mayor

3 - ¿Es única la recta calculada en el gráfico anterior?

Si queremos investigar la unicidad de la recta obtenida en el punto anterior, podemos ver que por ejemplo, si construimos otra recta con igual pendiente pero de ordenada ligeramente distinta, obtenemos la misma accuracy pero la recta que separa a los datos es distinta. Por lo tanto, la recta que maximiza la accuracy, no es unica.

```
accuracy(c(pendiente, ordenada+0.0001))
```

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

```
cantidad_filas_dataset <- 20
datos_accuracy <- data.frame(
            'Pendiente' = rep(NA, cantidad_filas_dataset),
            'Ordenada' = rep(NA, cantidad_filas_dataset),
            'Accuracy' = rep(NA, cantidad_filas_dataset)
)

cont <- 1
for (i in 0:3) {
  for (j in 2:6) {
    print(c(i,j))
    print(accuracy(c(i,j)))
    datos_accuracy$Pendiente[cont] <- i
    datos_accuracy$Ordenada[cont] <- j
    datos_accuracy$Accuracy[cont] <- accuracy(c(i,j))
    cont <- cont + 1
  }
}
```

```
## [1] 0 2
## [1] 0.6666667
## [1] 0 3
## [1] 0.7037037
## [1] 0 4
## [1] 1
## [1] 0 5
## [1] 0.8148148
## [1] 0 6
## [1] 0.4444444
## [1] 1 2
## [1] 1
## [1] 1 3
## [1] 0.962963
## [1] 1 4
## [1] 0.3333333
## [1] 1 5
## [1] 0.3333333
## [1] 1 6
## [1] 0.3333333
## [1] 2 2
## [1] 0.3703704
## [1] 2 3
## [1] 0.3333333
## [1] 2 4
## [1] 0.3333333
## [1] 2 5
## [1] 0.3333333
## [1] 2 6
## [1] 0.3333333
## [1] 3 2
## [1] 0.3333333
## [1] 3 3
## [1] 0.3333333
## [1] 3 4
## [1] 0.3333333
## [1] 3 5
## [1] 0.3333333
## [1] 3 6
## [1] 0.3333333
```

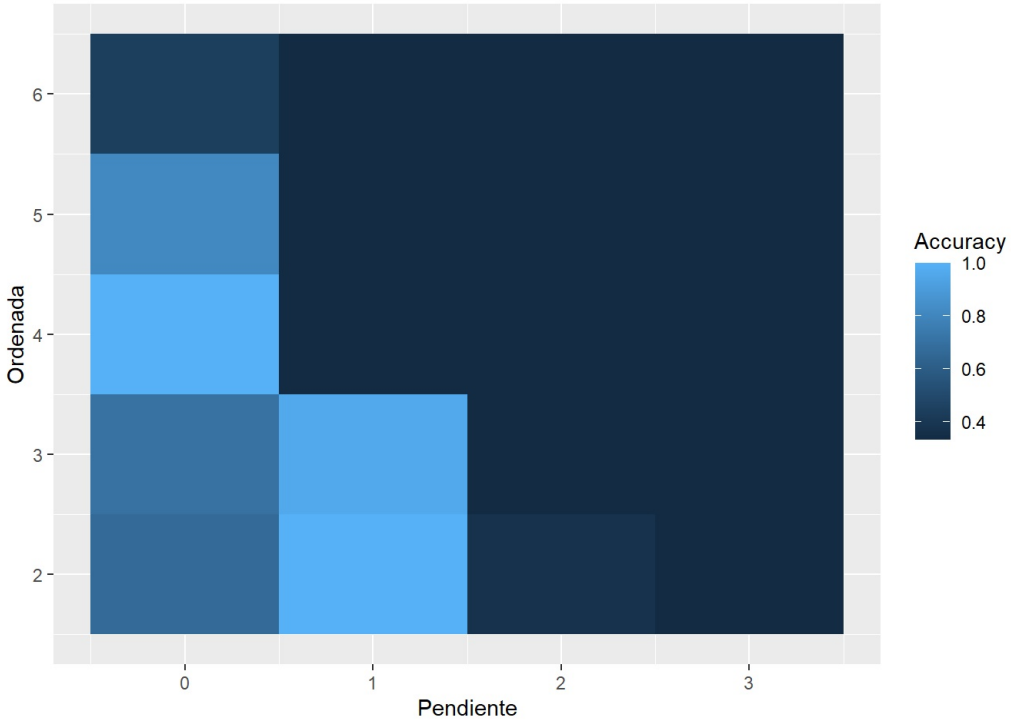
```
datos_accuracy
```

```
##    Pendiente Ordenada  Accuracy
## 1          0        2 0.6666667
## 2          0        3 0.7037037
## 3          0        4 1.0000000
## 4          0        5 0.8148148
## 5          0        6 0.4444444
## 6          1        2 1.0000000
## 7          1        3 0.9629630
## 8          1        4 0.3333333
## 9          1        5 0.3333333
## 10         1        6 0.3333333
## 11         2        2 0.3703704
## 12         2        3 0.3333333
## 13         2        4 0.3333333
## 14         2        5 0.3333333
## 15         2        6 0.3333333
## 16         3        2 0.3333333
## 17         3        3 0.3333333
## 18         3        4 0.3333333
## 19         3        5 0.3333333
## 20         3        6 0.3333333
```

Vemos que las rectas (0,4) y (1,2) tienen la misma accuracy que nuestra recta del punto anterior, eso nos dice que dicha recta no es única.

Observemoslo en el siguiente grafico

```
ggplot(datos_accuracy, aes(x = Pendiente, y = Ordenada, fill = Accuracy)) +
  geom_tile()
```



Segunda Parte: Clasificación de especies arbóreas en base a las características de las mismas

Lectura del dataset y su estructura

```
datos2 <- read_csv("arbolado-en-espacios-verdes.csv")
```

```
## Rows: 51502 Columns: 17
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (8): nombre_com, nombre_cie, tipo_folla, espacio_ve, ubicacion, nombre_f...
## dbl (9): long, lat, id_arbol, altura_tot, diametro, inclinacio, id_especie, ...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(datos2)
```

```
## # A tibble: 6 × 17
##    long   lat id_arbol altura_…¹ diame…² incli…³ id_es…⁴ nombr…⁵ nombr…⁶ tipo_…⁷
##   <dbl> <dbl>    <dbl>     <dbl>   <dbl>   <dbl>   <dbl> <chr>   <chr>   <chr>
## 1 -58.5 -34.6        1         6      35       0      53 Washin… Washin… Palmera
## 2 -58.5 -34.6        2         6      35       0      53 Washin… Washin… Palmera
## 3 -58.5 -34.6        3         6      35       0      53 Washin… Washin… Palmera
## 4 -58.5 -34.6        4        17      50       0      65 Ombú    Phytol… Árbol …
## 5 -58.5 -34.6        5        17      50       0      65 Ombú    Phytol… Árbol …
## 6 -58.5 -34.6        6        17      50       0      65 Ombú    Phytol… Árbol …
## # … with 7 more variables: espacio_ve <chr>, ubicacion <chr>, nombre_fam <chr>,
## #   nombre_gen <chr>, origen <chr>, coord_x <dbl>, coord_y <dbl>, and
## #   abbreviated variable names ¹altura_tot, ²diametro, ³inclinacio,
## #   ⁴id_especie, ⁵nombre_com, ⁶nombre_cie, ⁷tipo_folla
```

1 - Breve análisis exploratorio del dataset

Nos quedamos con las variables de interés y filtramos los datos faltantes.

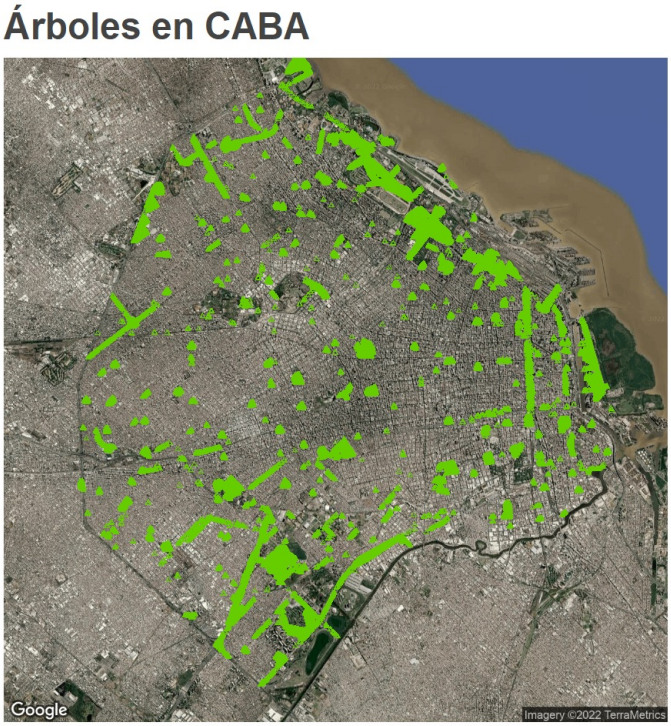
```
datos_arboles <- datos2 %>%
  select(nombre_com, lat, long, espacio_ve, tipo_folla, diametro, inclinacio, altura_tot)
```

Creamos un mapa con la distribución de árboles en la Ciudad de Buenos Aires. Además analizamos la distrubución de las alturas en un gráfico de densidad.

```
#registro clave para API de Google Maps
ggmap::register_google(key = "AIzaSyAT0QpI9NMPFrJNNUKAiaDhvkr0ymNFWEA")


#retrieve mapa de caba, localizada con un centro
mapa_caba <- get_map(location = c(-58.44538,-34.61743), zoom = 12, maptype = 'satellite')
```

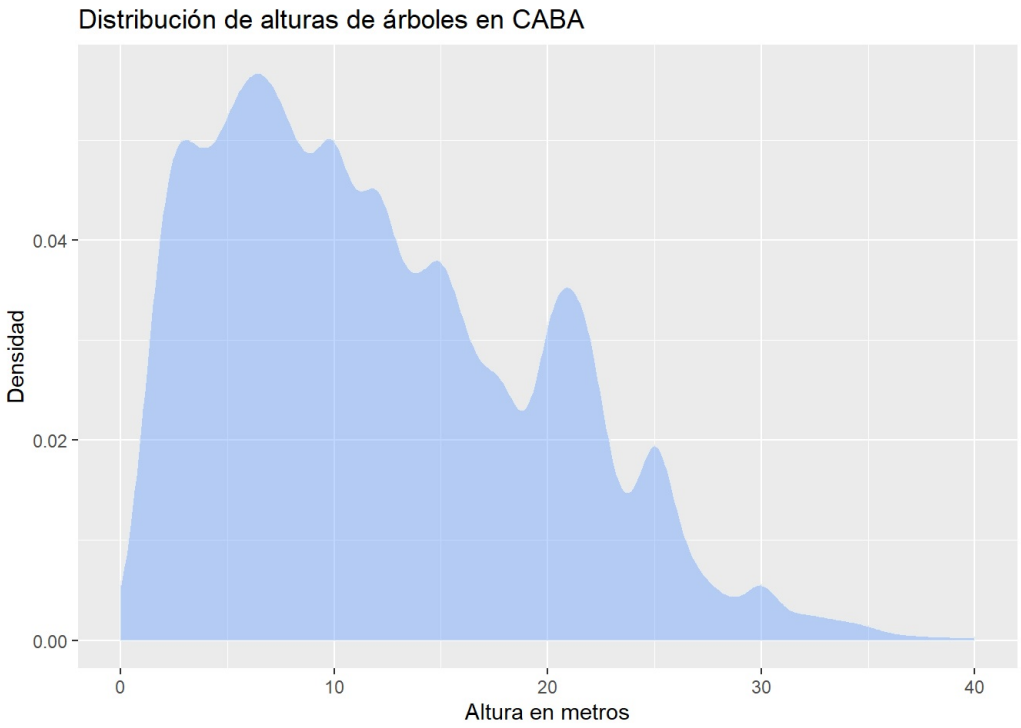
```
## i <https://maps.googleapis.com/maps/api/staticmap?center=-34.61743,-58.44538&zoom=12&size=640x640&scale=2&mapt
ype=satellite&language=en-EN&key=xxx>
```

```
#mapa scatter
ggmap(mapa_caba) +
  geom_point(data = datos_arboles, aes( x = long, y = lat), size = 0.5, color = "chartreuse3", shape = 2) +
  labs(title = "Árboles en CABA")+
  theme(text = element_text(color = "#444444")
        ,plot.title = element_text(size = 18, face = 'bold')
        ,axis.text = element_blank()
        ,axis.title = element_blank()
        ,axis.ticks = element_blank()
        ) +
  guides(fill = guide_legend(override.aes= list(alpha = 1)))
```




```
#density plot alturas

datos_arboles %>% ggplot(aes(altura_tot), xlab = "Altura en metros", ) +
  geom_density(fill = "#619CFF",color="#e9ecef", alpha = 0.4) +
  labs(title = "Distribución de alturas de árboles en CABA",
       x = "Altura en metros",
       y = "Densidad") +
  xlim(0,40)
```

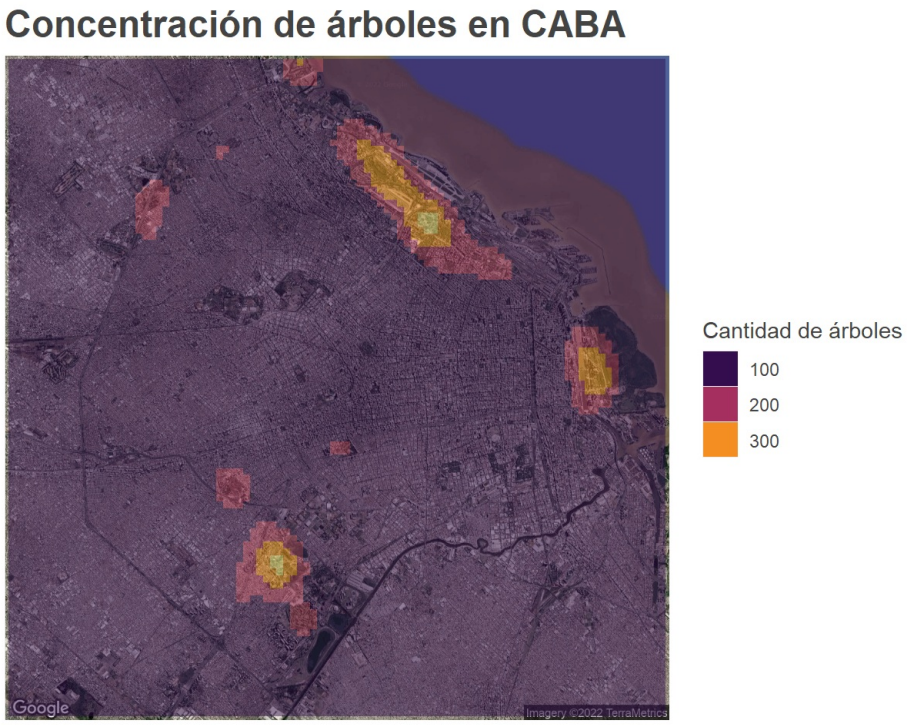
```
## Warning: Removed 78 rows containing non-finite values (stat_density).
```



Vemos que la mayoría de los árboles poseen una altura entre 5 y 15 metros.

El primer mapa no es muy útil ya que muchos puntos están superpuestos y no es posible apreciar la concentración de árboles. Por eso realizamos un mapa de calor, que muestra las zonas con mayor densidad de árboles plantados.

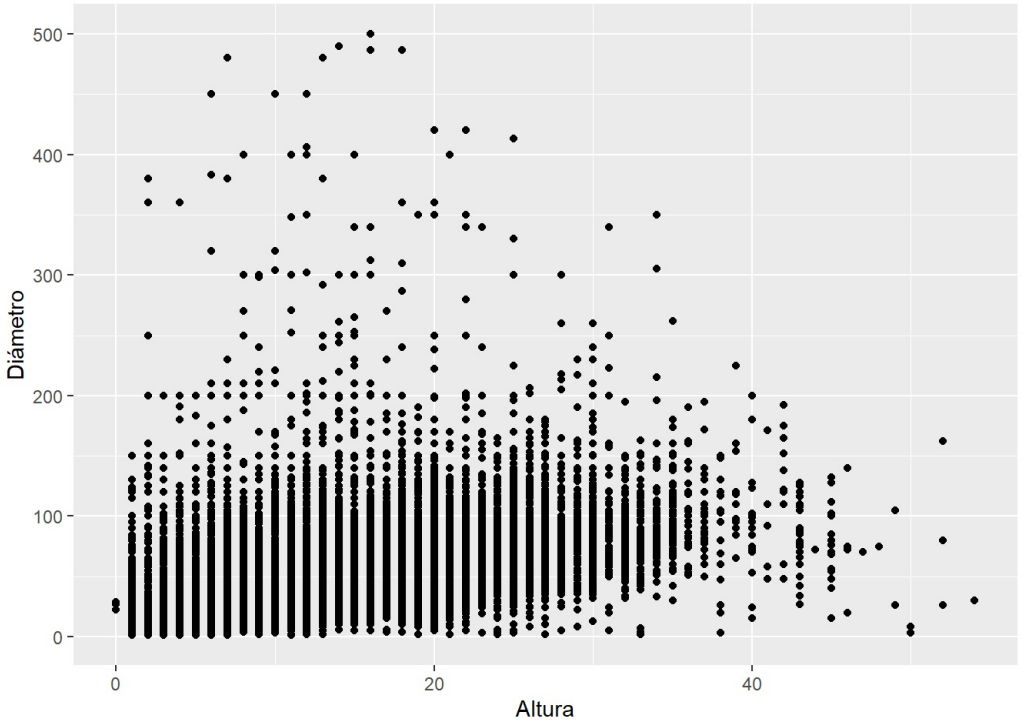
```
ggmap(mapa_caba) +
  stat_density2d(data = datos_arboles, aes(x = long, y = lat, fill = ..density..), geom = "tile", contour = F, al
pha = .5) +
  scale_fill_viridis_b(option = "inferno") +
  labs(title = "Concentración de árboles en CABA",
       fill = str_c('Cantidad de árboles')) +
  theme(text = element_text(color = "#444444")
        ,plot.title = element_text(size = 18, face = 'bold')
        ,axis.text = element_blank()
        ,axis.title = element_blank()
        ,axis.ticks = element_blank()
        ) +
  guides(fill = guide_legend(override.aes= list(alpha = 1)))
```



Observamos principalmente 3 zonas con alta densidad de árboles, que son en general parques y espacios verdes. La región ubicada más al norte corresponde a los Lagos de Palermo y a toda la zona del Rosedal. La región del sur corresponde al Parque de la Ciudad, y la del este a los parques ubicados en Puerto Madero.

Veamos ahora si existe alguna relación entre la altura y el diámetro de los árboles.

```
datos_arboles %>% ggplot(aes(x = altura_tot, y = diametro )) +
  geom_point() +
  labs(x = "Altura", y = "Diámetro")
```



No parece existir relación entre altura y diámetro de un árbol, pues hay una gran cantidad de árboles con la misma altura que tiene diámetros muy distintos. Sí podemos notar que los valores de las alturas fueron introducidos en números enteros, por eso los surcos en el scatter plot.

Analicemos ahora qué tipo de follaje y que árbol es más frecuente en la ciudad.

```
tabla_tipo_folla <- data.frame(table(datos_arboles$tipo_folla))
tabla_tipo_folla <- tabla_tipo_folla[order(tabla_tipo_folla$Freq, decreasing = TRUE),]
colnames(tabla_tipo_folla) <- c("Tipo de Follaje", "Frecuencia")
tabla_tipo_folla
```

```
##                 Tipo de Follaje Frecuencia
## 3  Árbol Latifoliado Caducifolio      29528
## 4      Árbol Latifoliado Perenne      12777
## 2         Árbol Conífero Perenne       3369
## 10                       Palmera       2911
## 9                 No Determinado       2115
## 7                Arbusto Perenne        333
## 6             Arbusto o Herbacea        203
## 1     Árbol Conífero Caducifolio        166
## 5             Arbusto Caducifolio         97
## 8                     Miscelaneo          3
```

```
tabla_nombre_com <- data.frame(table(datos_arboles$nombre_com))
tabla_nombre_com <- tabla_nombre_com[order(tabla_nombre_com$Freq, decreasing = TRUE),]
colnames(tabla_nombre_com) <- c("Nombre Común", "Frecuencia")
head(tabla_nombre_com)
```

```
##             Nombre Común Frecuencia
## 146            Eucalipto       4112
## 323           Tipa blanca       4031
## 185             Jacarandá       3255
## 258 Palo borracho rosado       3150
## 86              Casuarina       2719
## 169     Fresno americano       2166
```

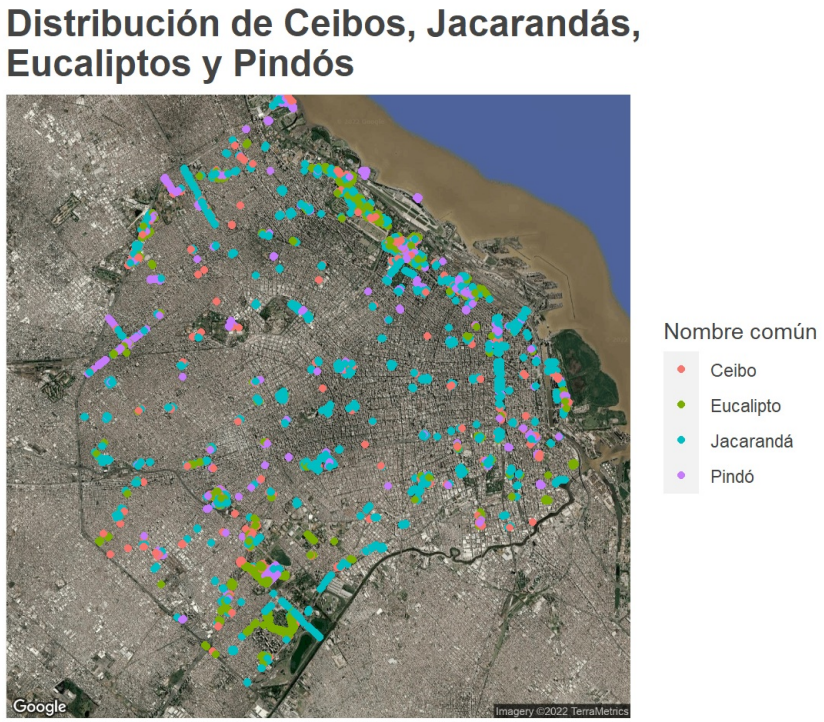
Vemos que el Árbol Latifoliado Caducifolio es el follaje más común. También se ve que el Eucalipto predomina en la ciudad. Le sigue la Tipa Blanca y el Jacarandá.

2 - Analizando los Ceibos, Jacarandás, Pindós y

Eucaliptos

Nos enfocamos en dichas especies. Veamos la distribución geográfica de cada una.

```
especies <- c("Ceibo","Jacarandá","Eucalipto", "Pindó")
datos_arboles2 <- datos_arboles %>% filter(nombre_com %in% especies)

ggmap(mapa_caba) +
  geom_point(data = datos_arboles2, aes( x = long, y = lat, color = nombre_com), size = 1.5, shape = 19) +
  labs(title = str_c("Distribución de Ceibos, Jacarandás,\n", "Eucaliptos y Pindós"),
       color = "Nombre común")+
  theme(text = element_text(color = "#444444")
        ,plot.title = element_text(size = 18, face = 'bold')
        ,axis.text = element_blank()
        ,axis.title = element_blank()
        ,axis.ticks = element_blank()
        ) +
  guides(fill = guide_legend(override.aes= list(alpha = 1)))
```

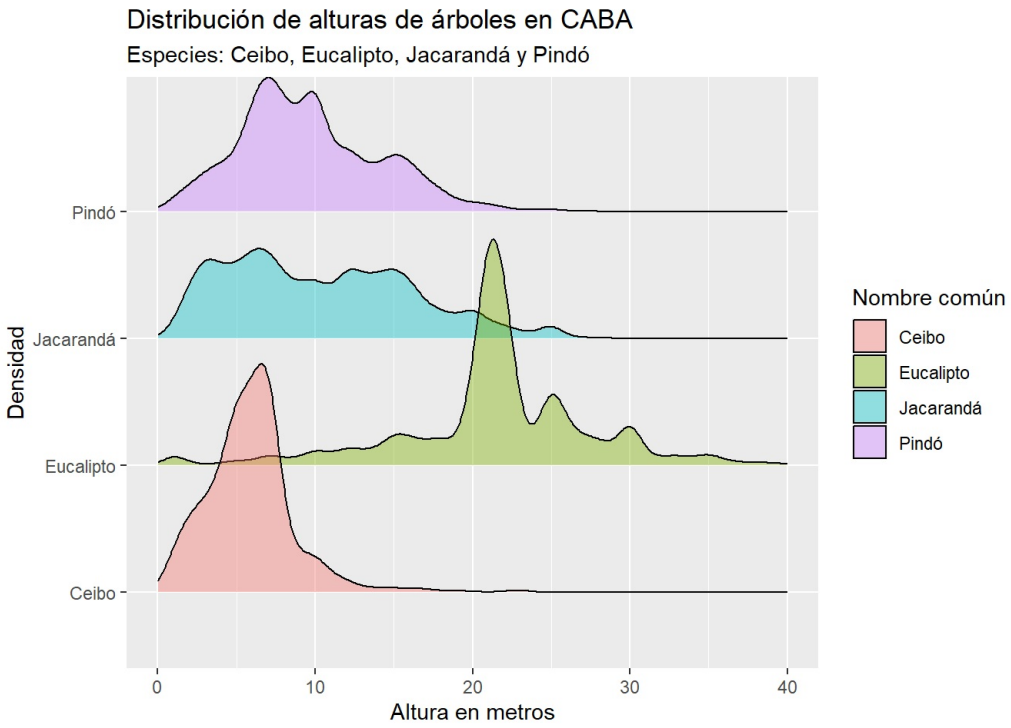


Veamos ahora la distribución de las alturas de cada especie.

```
#density plot alturas

datos_arboles2 %>% ggplot(aes(x = altura_tot, y = nombre_com, fill = nombre_com)) +
  geom_density_ridges(alpha = 0.4) +
  labs(title = "Distribución de alturas de árboles en CABA",
       subtitle = "Especies: Ceibo, Eucalipto, Jacarandá y Pindó",
       x = "Altura en metros",
       y = "Densidad",
       fill = "Nombre común") +
  xlim(0,40)
```

```
## Picking joint bandwidth of 0.752
```

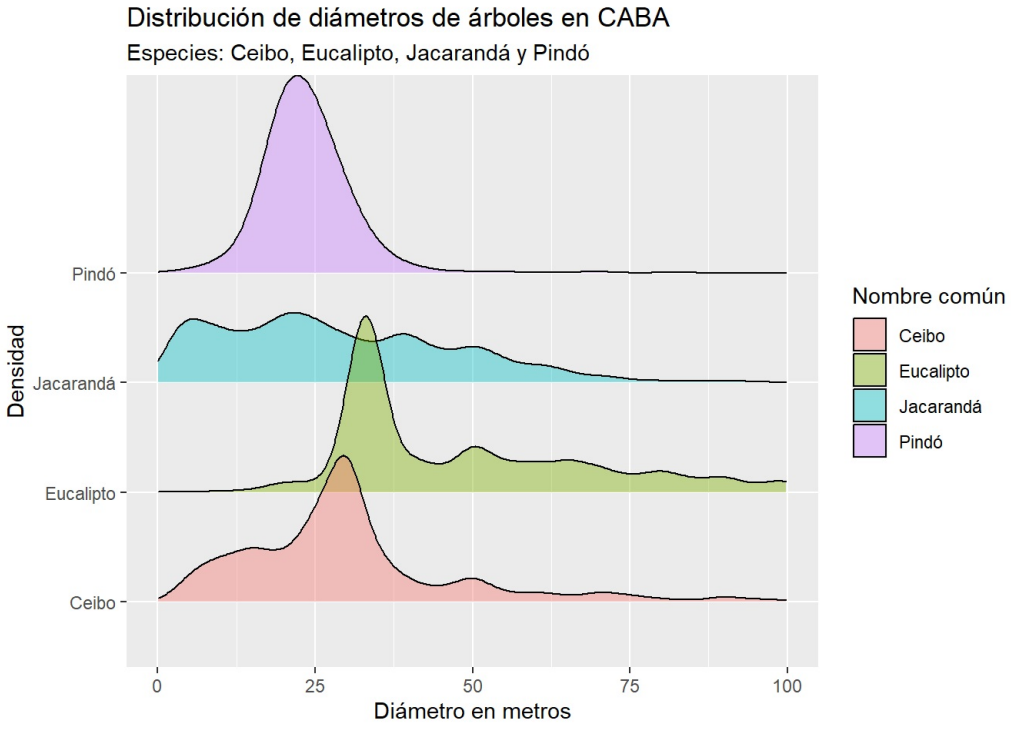
```
## Warning: Removed 34 rows containing non-finite values (stat_density_ridges).
```




```
datos_arboles2 %>% ggplot(aes(x = diametro, y = nombre_com, fill = nombre_com)) +
  geom_density_ridges(alpha = 0.4) +
  labs(title = "Distribución de diámetros de árboles en CABA",
       subtitle = "Especies: Ceibo, Eucalipto, Jacarandá y Pindó",
       x = "Diámetro en metros",
       y = "Densidad",
       fill = "Nombre común") +
  xlim(0,100)
```

```
## Picking joint bandwidth of 2.56
```

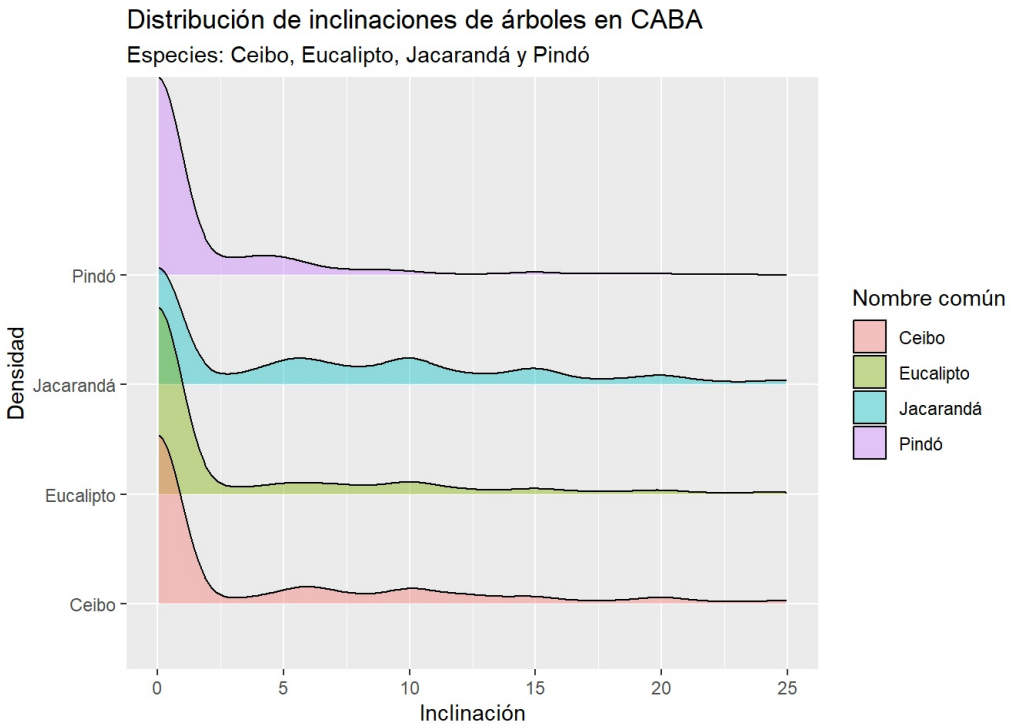
```
## Warning: Removed 331 rows containing non-finite values (stat_density_ridges).
```




```
datos_arboles2 %>% ggplot(aes(x = inclinacio, y = nombre_com, fill = nombre_com)) +
  geom_density_ridges(alpha = 0.4) +
  labs(title = "Distribución de inclinaciones de árboles en CABA",
       subtitle = "Especies: Ceibo, Eucalipto, Jacarandá y Pindó",
       x = "Inclinación",
       y = "Densidad",
       fill = "Nombre común") +
  xlim(0,25)
```

```
## Picking joint bandwidth of 0.951
```

```
## Warning: Removed 203 rows containing non-finite values (stat_density_ridges).
```



Se puede observar que en los Eucaliptos predominan las alturas superiores a 20m, siendo una de las especies más altas. Vemos que, en cambio, los Ceibos no suelen superar los 15m de altura. Además, considerando tanto la altura como el diámetro, son los Jacarandás los que presentan una distribución más uniforme a lo largo de los interavalos considerados. Es decir, los Jacarandás tienen más variabilidad en altura y diámetro que el resto de las especies, que muestras comportamientos más tendenciosos.

3 y 4 - Clasificación basada en los k vecinos más cercanos

```
require('class')
```

```
## Loading required package: class
```

```
require('psych')
```

```
## Loading required package: psych
```

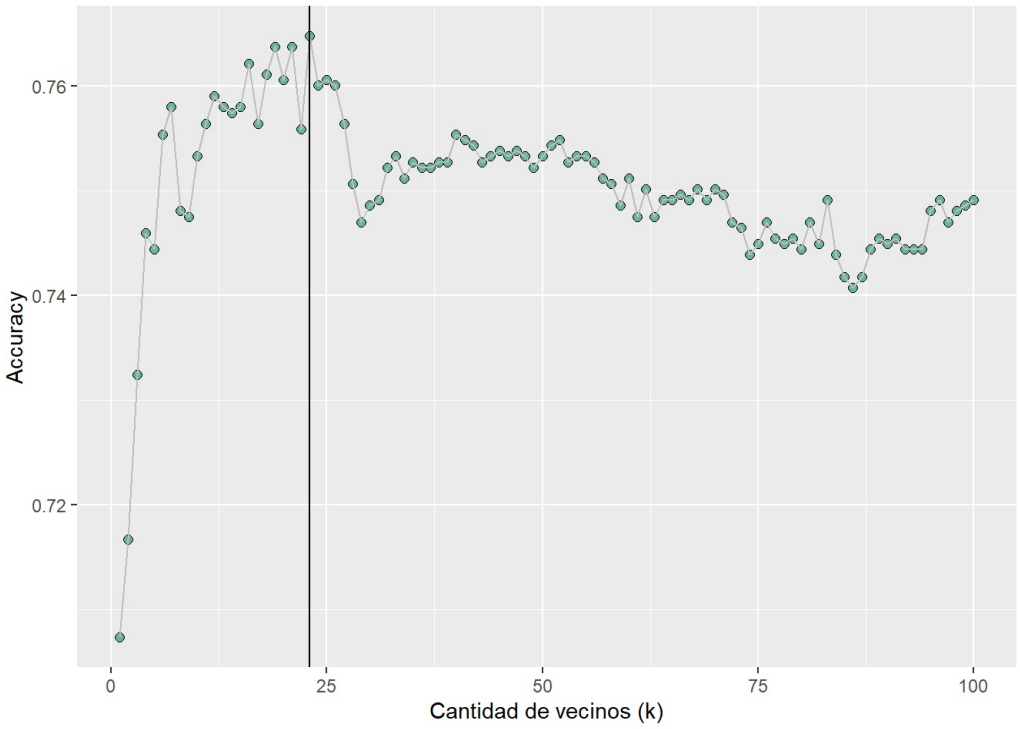
```
## Warning: package 'psych' was built under R version 4.2.2
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:rje':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
datos_arboles3 <- datos_arboles2 %>% select(nombre_com, altura_tot, diametro, inclinacio)

#normalizacion (ademas saco la columna de nombre porq es lo q quiero predecir)

normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }

datos_arboles3_n <- as.data.frame(lapply(datos_arboles3 [,2:4], normalize))

#intervalos de training y de testeo
set.seed(123)
dat_d <- sample(1:nrow(datos_arboles3_n),size=nrow(datos_arboles3_n)*0.8,replace = FALSE)

#seleccion random del 80% de los datos
train_set <- datos_arboles3_n[dat_d,] # 80%
test_set <- datos_arboles3_n[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles3[dat_d,1]
test_nombres <-datos_arboles3[-dat_d,1]


#creo df de accuracy y k-vecinos

k_i <- c() #nro de vecinos (=i)
acc_i1 <- c()

#relleno con un for que pase por todos los vecinos

for (i in 1:100){
  k_i <- append(k_i, i) #aca solo agrego la iteracion(o sea el nro de vecinos)
  knn_i <- knn(train = train_set, test=test_set, cl = train_nombres$nombre_com, k=i)  #modelo
  acc_i1 <-  append(acc_i1,mean(test_nombres$nombre_com == knn_i)) #accuracy

}

ggplot(data = data.frame(acc_i1), aes(x=k_i, y = acc_i1)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "Cantidad de vecinos (k)", y = "Accuracy") +
  geom_vline(aes(xintercept=which.max(acc_i1)))
```

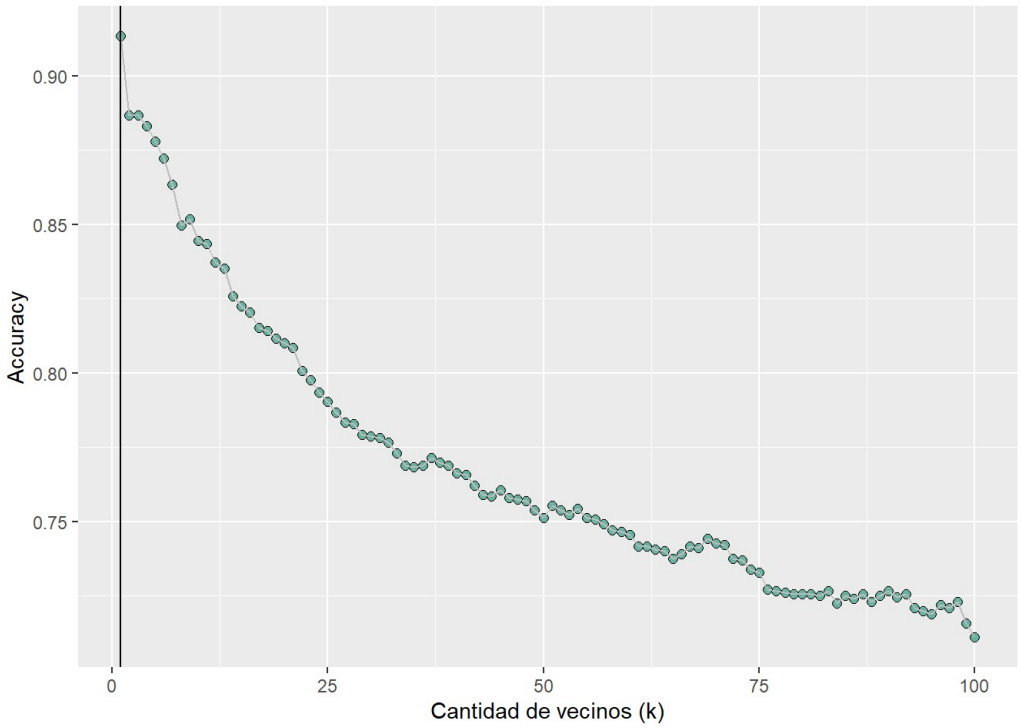


Podemos observar que el número de vecinos óptimo es 23. La accuracy varía porque, a pesar de que siempre usamos los mismos datos, la cantidad de vecinos con la cual se clasifica es distinto. El criterio de knn se basa únicamente en los k vecinos más cercanos de cada observación del datdaset, y no en su totalidad como conjunto. Por eso puede ser que si cambia el k, cambie la clasificación de un mismo punto.

5 - Comparación de clasificadores

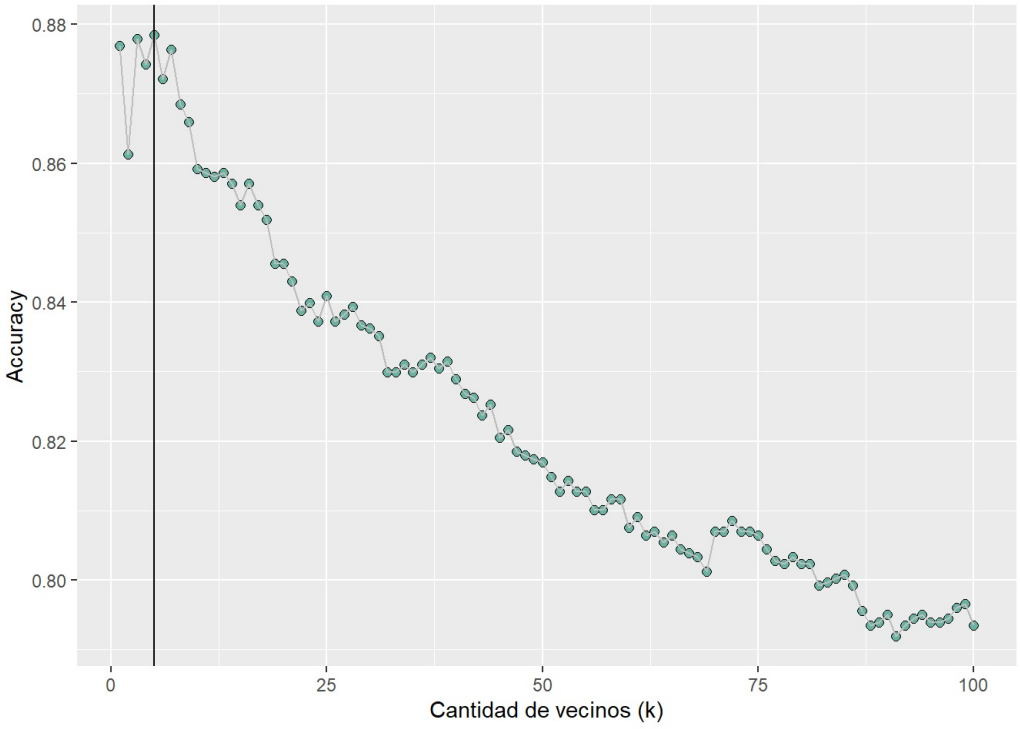
Realizamos otro clasificador con la ubicación geográfica de cada árbol.

```
datos_arboles5 <- datos_arboles2 %>% select(nombre_com, lat, long)

#normalizacion (ademas saco la columna de nombre porq es lo q quiero predecir)

normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }

datos_arboles5_n <- as.data.frame(lapply(datos_arboles5 [,2:3], normalize))

#intervalos de training y de testeo
set.seed(123)
dat_d <- sample(1:nrow(datos_arboles5_n),size=nrow(datos_arboles5_n)*0.8,replace = FALSE)

#seleccion random del 80% de los datos
train_set <- datos_arboles5_n[dat_d,] # 80%
test_set <- datos_arboles5_n[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles5[dat_d,1]
test_nombres <-datos_arboles5[-dat_d,1]

k_i <- c()
acc_i2 <- c()

#relleno con un for que pase por todos los vecinos

for (i in 1:100){
  k_i <- append(k_i, i) #aca solo agrego la iteracion(o sea el nro de vecinos)
  knn_i <- knn(train = train_set, test=test_set, cl = train_nombres$nombre_com, k=i)  #modelo
  acc_i2 <-  append(acc_i2,mean(test_nombres$nombre_com == knn_i)) #accuracy

}

ggplot(data = data.frame(acc_i2), aes(x=k_i, y = acc_i2)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "Cantidad de vecinos (k)", y = "Accuracy")  +
  geom_vline(aes(xintercept=which.max(acc_i2)))
```



Vemos que para este clasificador el k óptimo es 1. El clasificador se está fijando cuál es el árbol que tiene más cercano. Vemos que la accuracy es de aproximadamente el 91%, quizás los árboles están ubicados en grupitos de su misma especie y por eso tiene tan buena accuracy.

```
datos_arboles6 <- datos_arboles2 %>% select(nombre_com, lat, long, altura_tot, inclinacio, diametro)

#normalizacion (ademas saco la columna de nombre porq es lo q quiero predecir)

normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }

datos_arboles6_n <- as.data.frame(lapply(datos_arboles6[,2:6], normalize))

#intervalos de training y de testeo
set.seed(123)
dat_d <- sample(1:nrow(datos_arboles6_n),size=nrow(datos_arboles6_n)*0.8,replace = FALSE)

#seleccion random del 80% de los datos
train_set <- datos_arboles6_n[dat_d,] # 80%
test_set <- datos_arboles6_n[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles6[dat_d,1]
test_nombres <-datos_arboles6[-dat_d,1]

k_i <- c()
acc_i3<- c()

#relleno con un for que pase por todos los vecinos

for (i in 1:100){
  k_i <- append(k_i, i) #aca solo agrego la iteracion(o sea el nro de vecinos)
  knn_i <- knn(train = train_set, test=test_set, cl = train_nombres$nombre_com, k=i)  #modelo
  acc_i3 <-  append(acc_i3,mean(test_nombres$nombre_com == knn_i)) #accuracy

}
ggplot(data = data.frame(acc_i3), aes(x=k_i, y = acc_i3)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "Cantidad de vecinos (k)", y = "Accuracy")  +
  geom_vline(aes(xintercept=which.max(acc_i3)))
```



Para este clasificador, el número óptimo de vecinos es 5 y tiene una accuracy del 88% aproximadamente. Quizás el buen desempeño del clasificador que sólo usa la ubicación se ve reducida cuando agregamos las demás variables.

6 - Árbol de decisión como clasificador

Utilizando las variables altura_tot, inclinación y diámetro realizamos distintos árboles de decisión para optimizar la accuracy, variando el minbucket.

```
require(rpart)
```

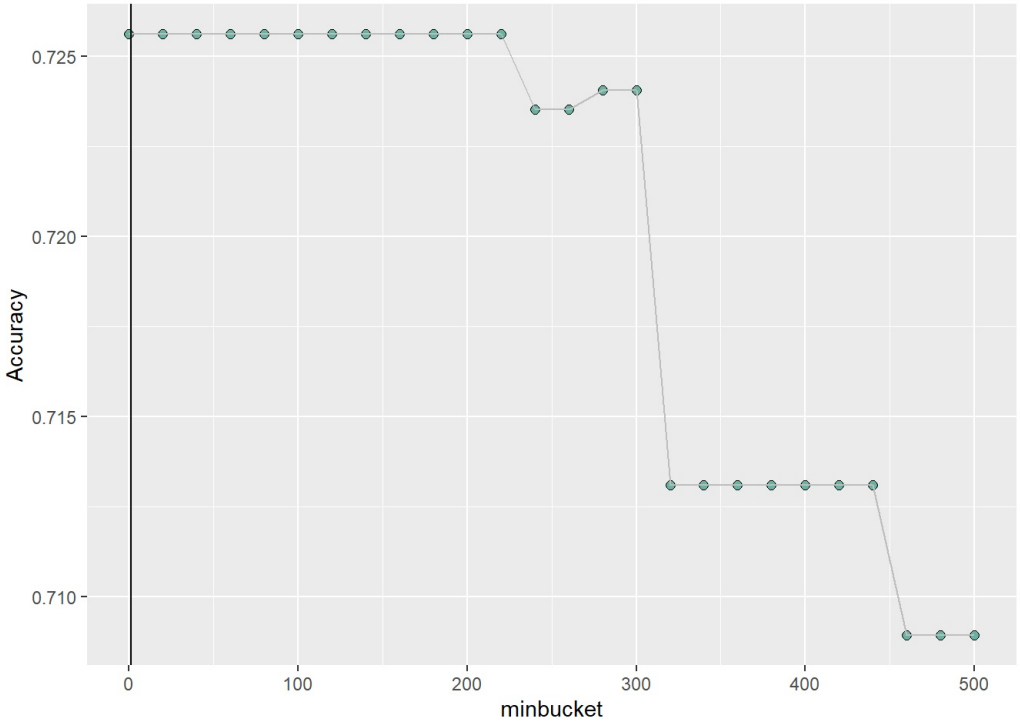
```
## Loading required package: rpart
```

```
require(rpart.plot)
```

```
## Loading required package: rpart.plot
```

```
## Warning: package 'rpart.plot' was built under R version 4.2.2
```

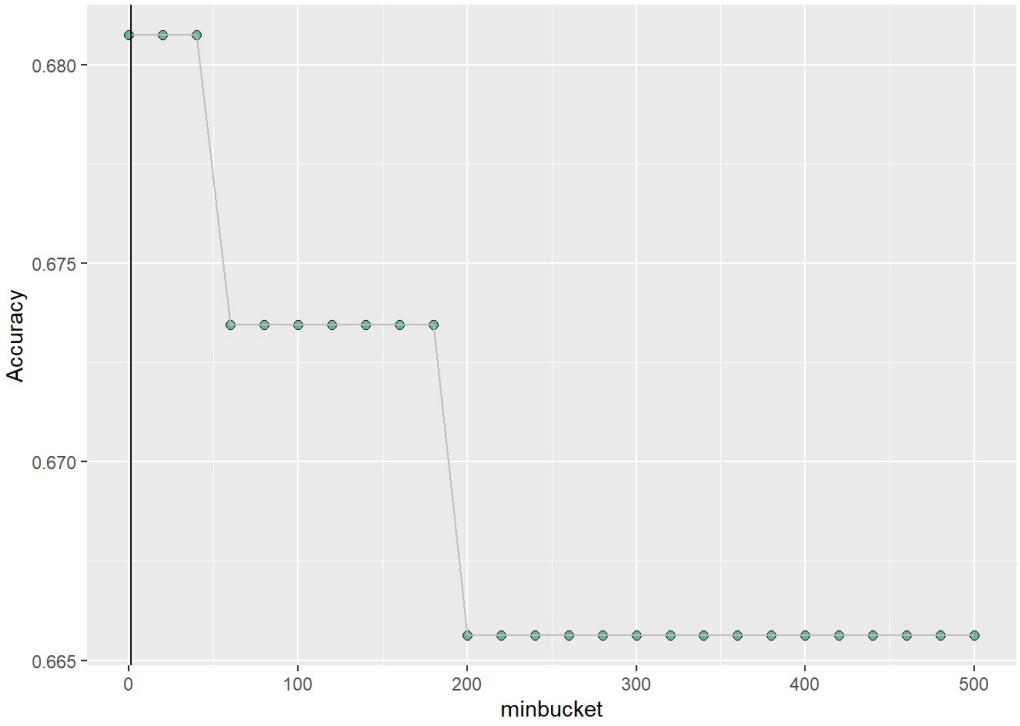
```
#usamos datos_arboles3
#seleccion random del 80% de los datos

train_set <- datos_arboles3[dat_d,] # 80%
test_set <- datos_arboles3[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles3[dat_d,1]
test_nombres <-datos_arboles3[-dat_d,1]

#un for para iteran en max depth

accs1 <- c()
for (i in seq(0,500,20)){
  fit <- rpart(nombre_com ~ altura_tot + inclinacio + diametro, data = train_set, method = 'class', minbucket = i
)
  df <- data.frame(test_set$nombre_com, predict(fit, newdata = test_set, type = "class"))
  accs1 <-  append(accs1,mean(df[,2] == df[,1]))
}

ggplot(data = data.frame(accs1), aes(x = seq(0,500,20), y = accs1)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "minbucket", y = "Accuracy") +
  geom_vline(aes(xintercept=which.max(accs1)))
```



Vemos que este árbol tiene una accuracy del 72.6%, siempre y cuando se utilice un minbucket entre 0 y 200.

Veamos ahora qué sucede con un árbol que sólo considera la ubicación geográfica.

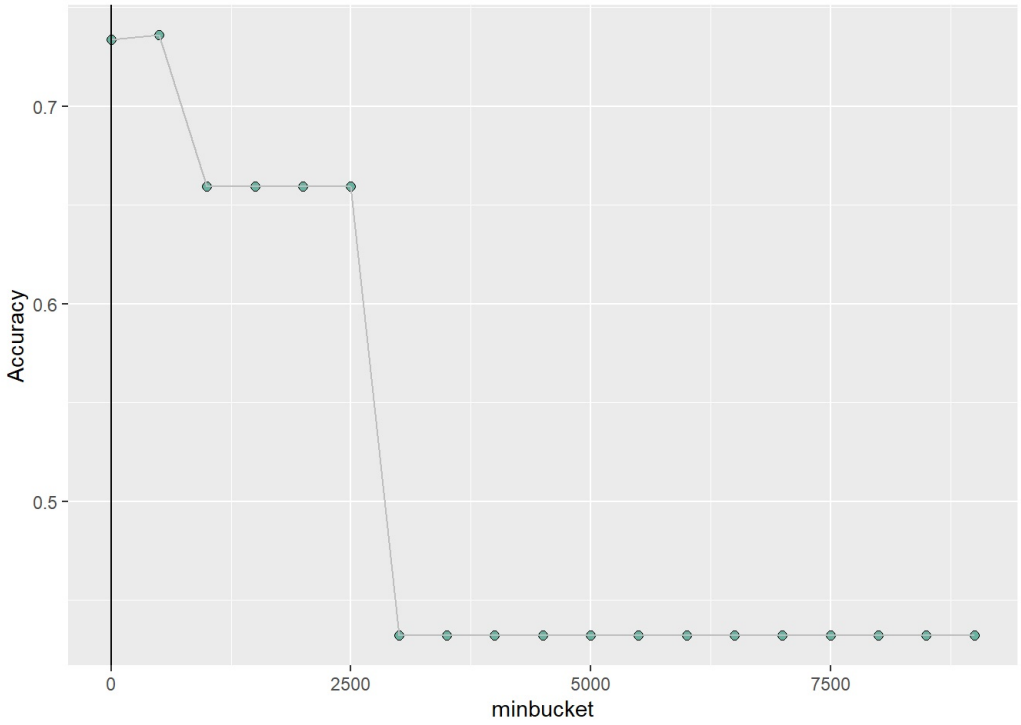
```
train_set <- datos_arboles5[dat_d,] # 80%
test_set <- datos_arboles5[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles5[dat_d,1]
test_nombres <-datos_arboles5[-dat_d,1]

#un for para iteran en max depth

accs2 <- c()
for (i in seq(0,500,20)){
  fit <- rpart(nombre_com ~ lat + long, data = train_set, method = 'class', minbucket = i)
  df <- data.frame(test_set$nombre_com, predict(fit, newdata = test_set, type = "class"))
  accs2 <-  append(accs2,mean(df[,2] == df[,1]))
}

ggplot(data = data.frame(accs2), aes(x = seq(0,500,20), y = accs2)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "minbucket", y = "Accuracy") +
  geom_vline(aes(xintercept=which.max(accs2)))
```



Este árbol tiene una accuracy del 68%, siempre y cuando se utilice un minbucket entre 0 y 30.

Veamos ahora qué sucede con un árbol que considera la ubicación geográfica, la altura, la inclinación y el diámetro.

```
train_set <- datos_arboles6[dat_d,] # 80%
test_set <- datos_arboles6[-dat_d,] # 20%

#creo otro df con los nombres, que es lo que quiero predecir
train_nombres <- datos_arboles6[dat_d,1]
test_nombres <-datos_arboles6[-dat_d,1]

#un for para iterar en max depth

accs3 <- c()
for (i in seq(0,9000,500)){
  fit <- rpart(nombre_com ~ lat + long + altura_tot + inclinacio + diametro, data = train_set, method = 'class',
minbucket = i)
  df <- data.frame(test_set$nombre_com, predict(fit, newdata = test_set, type = "class"))
  accs3 <-  append(accs3,mean(df[,2] == df[,1]))
}

ggplot(data = data.frame(accs3), aes(x = seq(0,9000,500), y = accs3)) +
  geom_point(shape=21, color="black", fill="#69b3a2", size=2) + geom_line(color="grey") +
  labs(x = "minbucket", y = "Accuracy") +
  geom_vline(aes(xintercept=which.max(accs3)))
```

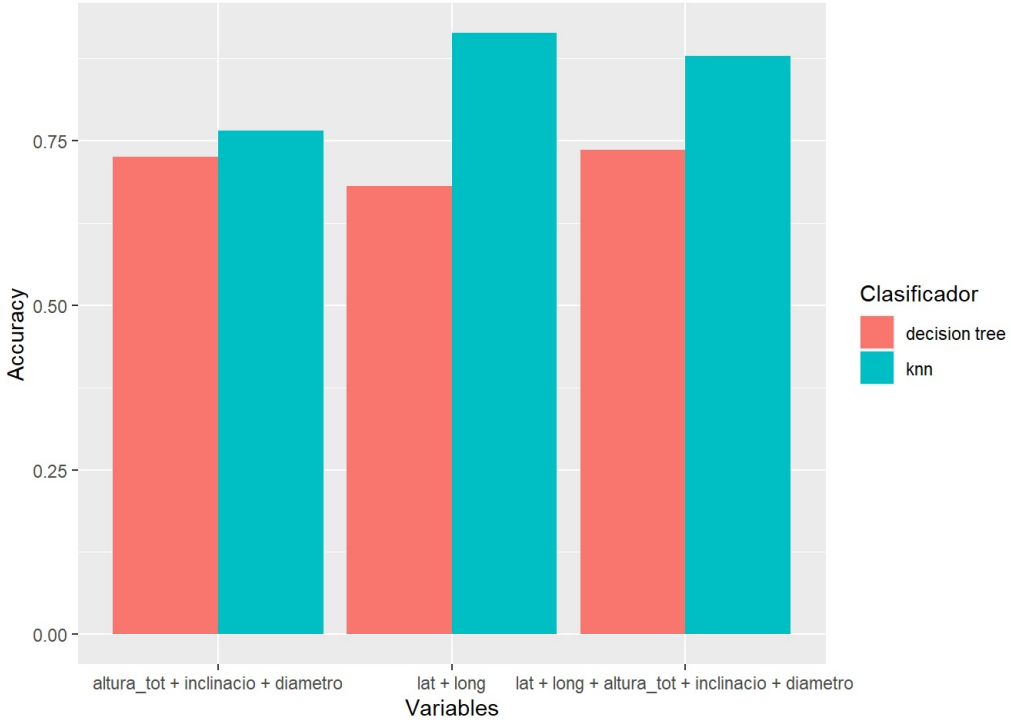


Este árbol tiene una accuracy del 73.6%, siempre y cuando se utilice un minbucket entre 0 y 500.

Si tenemos en cuenta la accuracy óptima de cada árbol, el más preciso es el que utiliza la ubicación, la altura, la inclinación y el diámetro.

Comparemos las accuracies de los clasificadores de k-vecinos más cercanos con los árboles de decisión, considerando además las tres combinaciones de variables.

```
clasificador <- c("knn","knn","knn", "decision tree", "decision tree", "decision tree")
variables <- c("altura_tot + inclinacio + diametro","lat + long", "lat + long + altura_tot + inclinacio + diametr
o","altura_tot + inclinacio + diametro","lat + long", "lat + long + altura_tot + inclinacio + diametro")
accu <- c(acc_i1[which.max(acc_i1)],acc_i2[which.max(acc_i2)], acc_i3[which.max(acc_i3)],accs1[which.max(accs1)],
accs2[which.max(accs2)], accs3[which.max(accs3)])
d <- data.frame(clasificador, variables, accu)

ggplot(d, aes(fill=clasificador, y=accu, x=variables)) +
  geom_bar(position='dodge', stat='identity') +
  labs(x = "Variables", y = "Accuracy", fill = "Clasificador")
```



Vemos que el que mejor desempeño tiene es el de k-vecinos más cercanos utilizando sólo la ubicación geográfica de los árboles. Esto podría deberse, como dijimos anteriormente, a que los árboles quizás se ubican en pequeños grupos de su misma especie, y como estamos comparando con un solo vecino, la accuracy es muy buena (91%).