Resumen

Paquetes y datos

Algoritmo k-Nearest Neighbors (knn) y dilema sesgo-varianza

Error fuera de muestra y Cross-Validaton

Sesgo-Varianza, kNN, Cross-Validation

Ciencia de Datos para Políticas Públicas

Pablo Aguirre Hörmann

Resumen

Este documento utiliza datos de casas en Boston (USA) para ilustrar lo siguiente:

- El algoritmo **k-Nearest Neighbors** (**kNN**);
- El intercambio **Sesgo-Varianza**; y
- El uso de **Cross Validation** para estimar el error de predicción (fuera de muestra) y determinar el hiperparámetro óptimo, en este caso el número de "vecinos cercanos" k.

Paquetes y datos

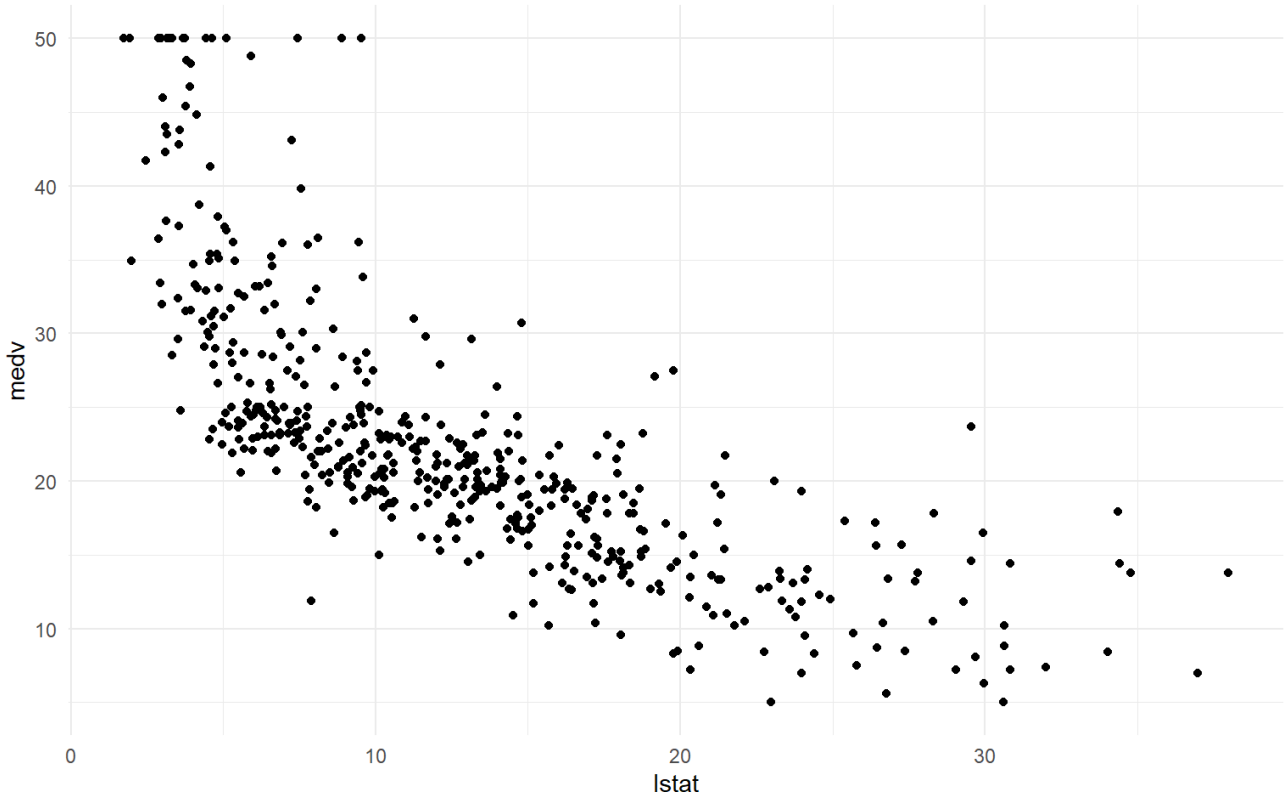
Observemos los datos disponibles.

```
library(tidyverse)
library(MASS)
library(kknn)
glimpse(Boston)
```

```
## Rows: 506
## Columns: 14
## $ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.088...
## $ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5...
## $ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87,...
## $ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.5...
## $ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.6...
## $ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9...
## $ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9...
## $ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4,...
## $ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311,...
## $ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2,...
## $ black   <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396...
## $ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17...
## $ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9,...
```

Nos enfocaremos en usar la variable `lstat` para predecir `medv`. Primero graficamos ambas variables:

```
Boston %>%
  ggplot(aes(x = lstat, y = medv)) +
  geom_point() +
  theme_minimal()
```



Algoritmo k-Nearest Neighbors (knn) y dilema sesgo-varianza

Primero estimaremos un modelo KNN usando $k = 5$ con todos los datos disponibles:

```
knn_model_5 <- kknn(medv ~ lstat,
                    train = Boston, test = Boston,
                    k = 5, kernel='rectangular')
```

Crearemos una función para graficar los resultados ya que realizaremos esta misma acción unas veces más.

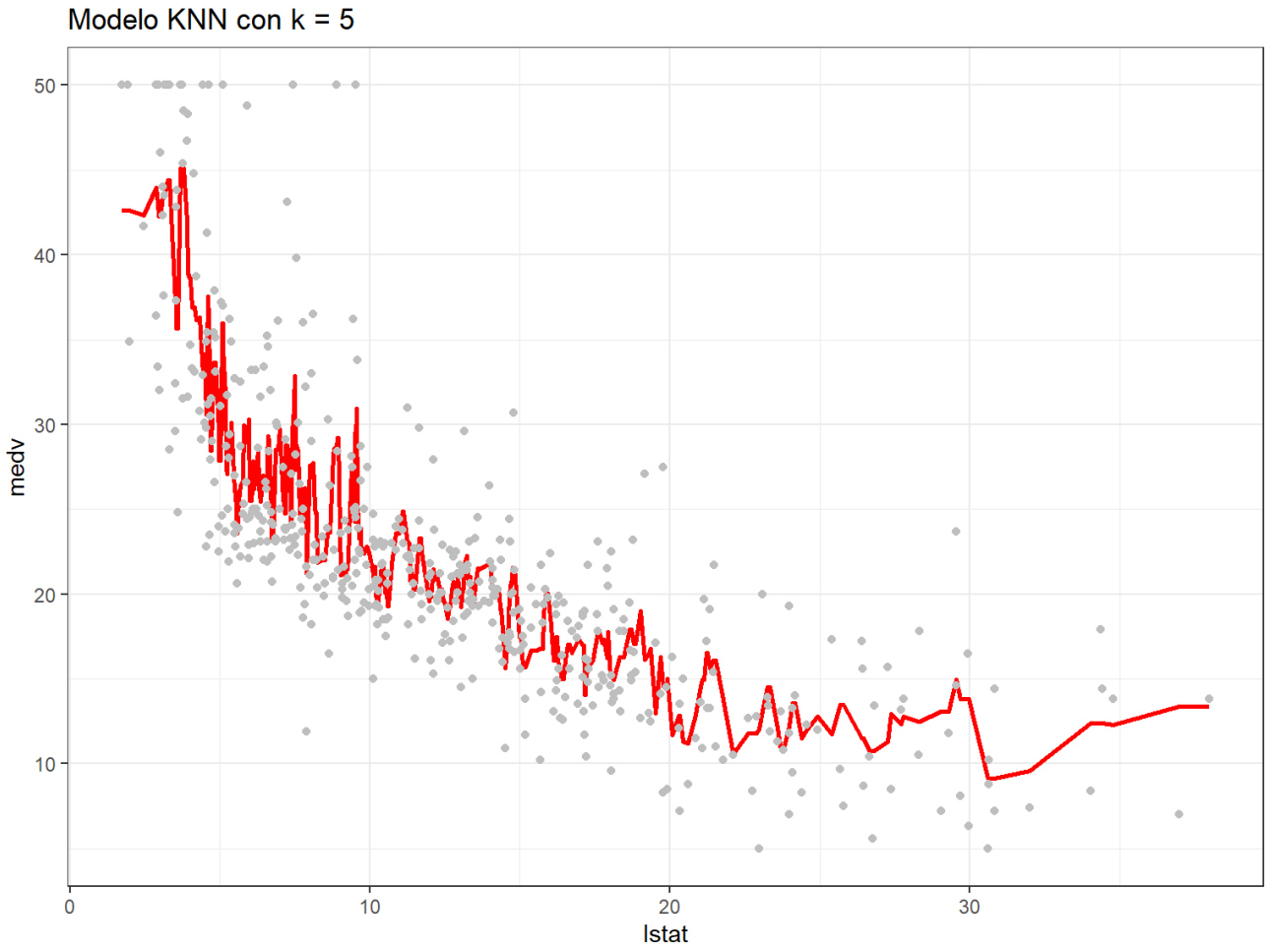
```
knn_model_5 <- kknn(medv ~ lstat,
                    train = Boston, test = Boston,
                    k = 5, kernel='rectangular')

plot_knn <- function(modelo, titulo){

  ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_line(aes(x = lstat, y = modelo$fitted.values), col = "red", size = 1) +
  geom_point(col = "grey") +
  theme_bw() +
  labs(title = titulo)

}

plot_knn(modelo = knn_model_5, titulo = "Modelo KNN con k = 5")
```

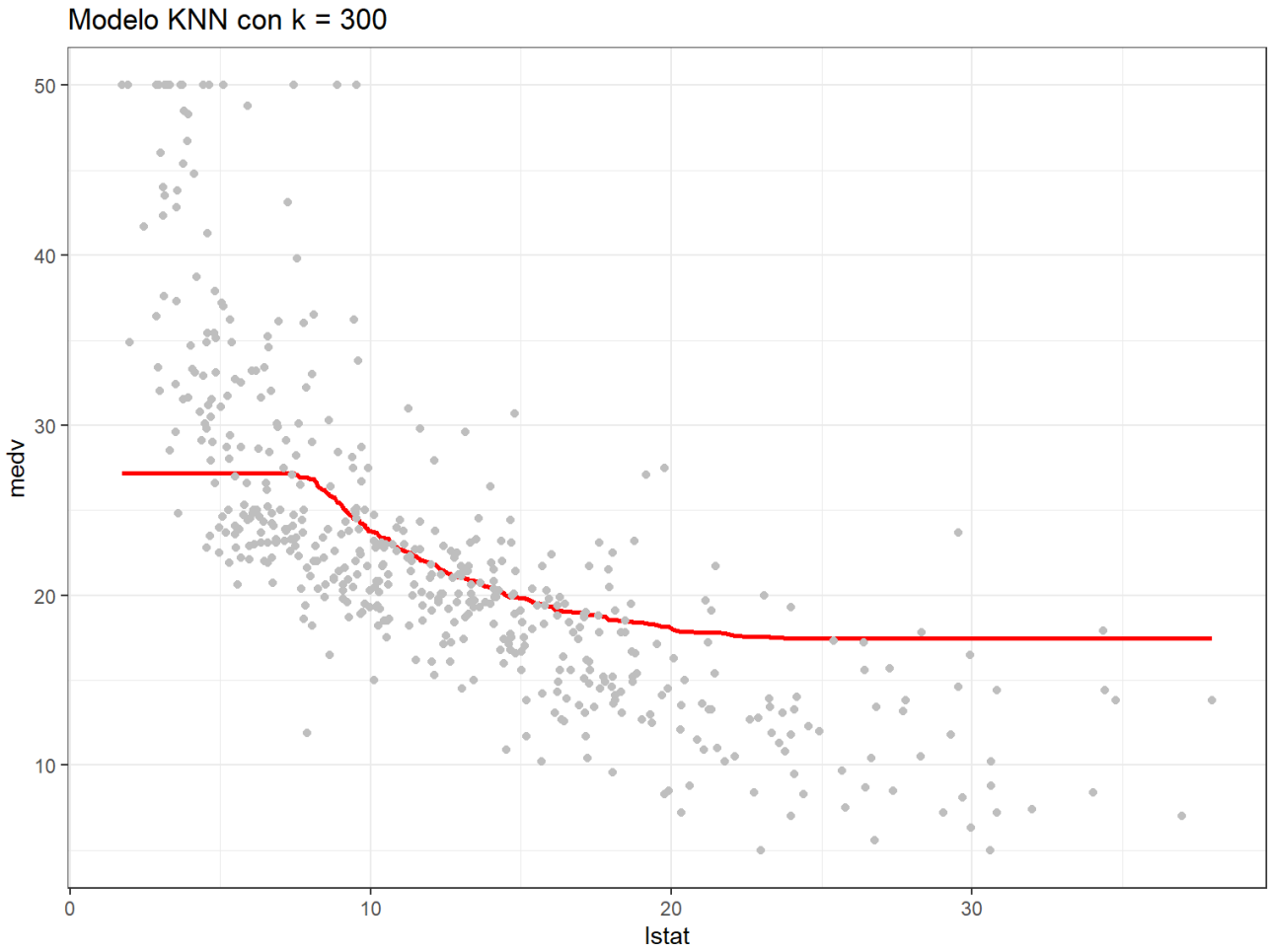


Con $k = 5$, un número bajo, tenemos una curva con muchos "saltos" que **ajusta bien los datos de entrenamiento** pero que es **muy sensible a pequeños cambios** en la variable `lstat` . Este es un modelo con **BAJO SESGO** y **ALTA VARIANZA**. No es un modelo que nos gusta.

Tratemos ahora con $k = 300$.

```
knn_model_300 <- kknn(medv ~ lstat,
                      train = Boston, test = Boston,
                      k = 300, kernel='rectangular')

plot_knn(modelo = knn_model_300, titulo = "Modelo KNN con k = 300")
```

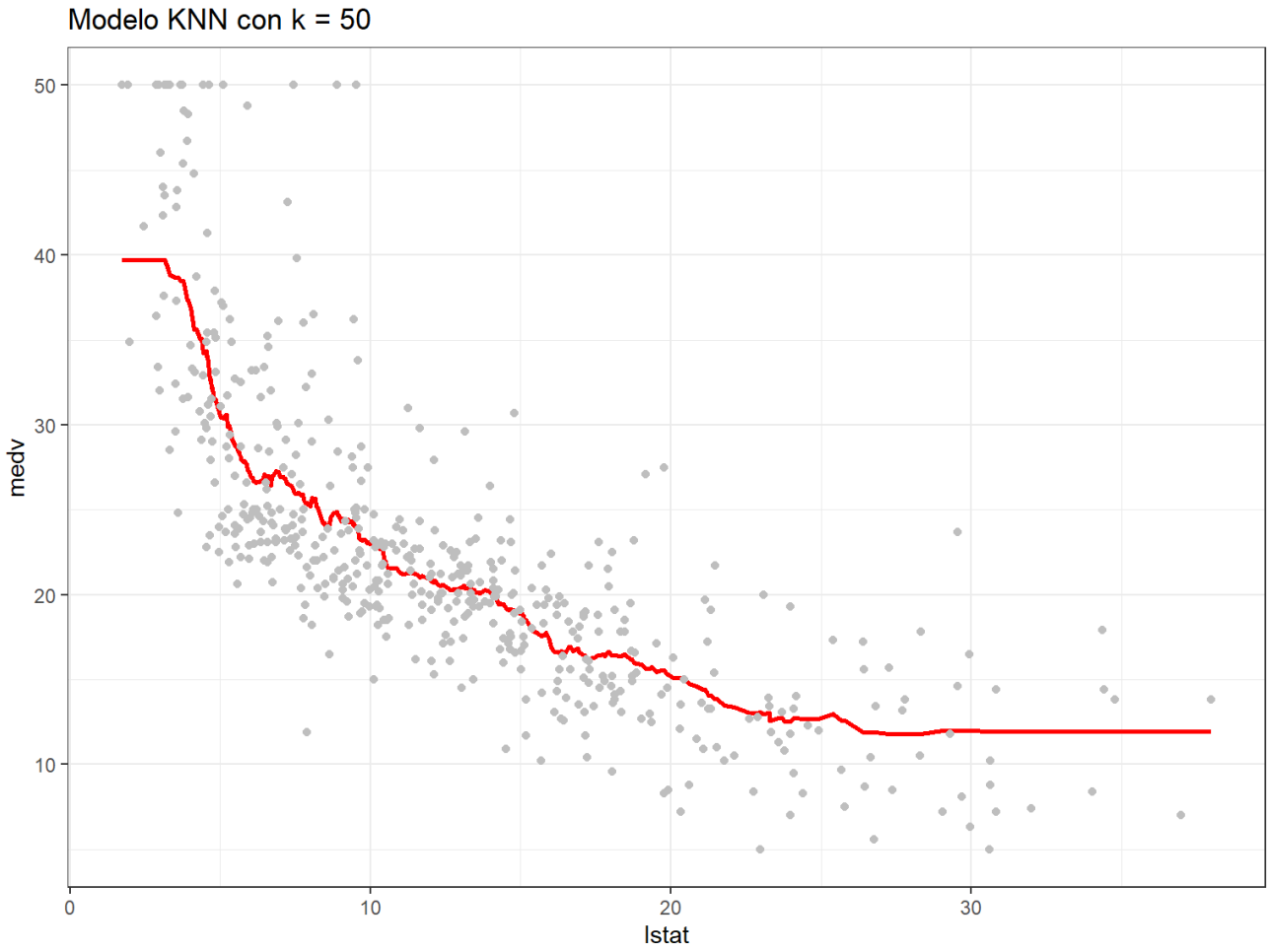


En este caso parece que nos fuimos a otro extremo. La linea parece ser **no extremadamente sensible**, pero también **muy simple y "suave"**. En otras palabras, **no responde lo suficiente** a cambios en la variable `lstat` . En este caso decimos que tenemos **ALTO SESGO Y BAJA VARIANZA**.

Por último, intentemos algo intermedio como $k = 50$.

```
knn_model_50 <- kknn(medv ~ lstat,
                      train = Boston, test = Boston,
                      k = 50, kernel='rectangular')

plot_knn(modelo = knn_model_50, titulo = "Modelo KNN con k = 50")
```



Esto pareciera ser más razonable y podríamos decir que la curva **es una buena generalización** de la relación entre ambas variables. A esta curva o modelo lo llamarías de **bajo sesgo y baja varianza**. A esto es a lo que queremos llegar.

Entonces, el mensaje principal es que a lo largo de un rango de **hyper-parameter** k de más pequeño a más grande, vemos un espectro de modelos que van desde "bajo sesgo y alta varianza" a "alto sesgo y baja varianza". A este fenómeno se le conoce como el **DILEMA SESGO-VARIANZA**, un concepto fundamental en *Machine Learning* y que aplica no solo a los modelos KNN sino que a todo tipo de modelos.

El dilema sesgo-varianza tiene relación con la **generalización de un modelo predictivo** cuando se ve enfrentado a datos con los cuales no fue entrenado/estimado. Si un modelo tiene alto sesgo o alta varianza no le irá bien con observaciones nuevas. Un buen modelo necesita tener la perfecta combinación de **bajo sesgo y baja varianza**.

Error fuera de muestra y Cross-Validaton

Para **cuantificar que tan bueno es un modelo predictivo**, necesitamos estimar el **error de predicción fuera de muestra**. En otras palabras, **medir que tan bien predice un modelo en datos que no fueorn usados para su entrenamiento/estimación**.

Una de las formas más populares para calcular el error de predicción fuera de la muestra es lo conocido como **cross-validation o validación cruzada** (Les dejo el link de Wikipedia) (http://en.wikipedia.org/wiki/Cross-validation_(statistics)%20en%20caso%20de%20que%20quieran%20leer%20un%20poco%20más%20al%20respecto).

Por otro lado, también necesitamos una métrica para evaluar que tan buen predictor es cada uno de nuestros modelos. En este caso utilizaremos la **Raiz del Error Cuadrático Medio** (**RMSE en inglés**) (http://en.wikipedia.org/wiki/Root-mean-square_deviation). Que corresponde a:

$$RSME = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y_i} - y_i)^2}{n}}$$

Lo que haremos es ver como varía el error de predicción para distintos valores de k (vecinos cercanos), en específico de 1 a 200. A menor valor de k mayor complejidad/flexibildiad del modelo. Por otro lado, haremos la estimación utilizando *k-fold cross validation* con 10 *folds*, es decir, separaremos los datos en 10 grupos y realizaremos 10 iteraciones donde 9 grupos serán utilizados para estimar el modelo correspondiente y el grupo restante será utilizado para predecir. Este proceso nos dejará con 10 estimaciones de errores de predicción que luego promediaremos para tener el valor de error de predicción para cada k. En total tendrémos 200 errores de predicción (1 para cada valor de k) y con esto podremos ver para que valor de k se obtiene el valor más pequeño de error de predicción (RMSE).

```
# Definir el máximo de valores de "k" a probar
rango_k <- 200

# Definir el número de "folds"
fold <- 10

# Asignar a cada observación un "fold"
Boston <- Boston %>%
  mutate(fold = c(rep(1:fold, 50), 1:6))

# Vector que guarda los resultados del error de predicción para cada "k"
error_total <- vector(length = rango_k)

# Iteración para cada valor de "k"
for (i in 1:rango_k){

  # Vector que guarda los resultados de error para cada "fold"
  error <- vector(length = fold)

  # Iteración para cada "fold"
  for (j in 1:fold){

    # Definir grupos de entrenamiento y test
    train <- Boston %>%
      filter(fold != j)
    test <- Boston %>%
      filter(fold == j)

    # Estimar modelo
    knn_model <- kknn(medv ~ lstat,
                      train = train, test = test,
                      k = i, kernel='rectangular')

    # Almacenar error del "fold"
    error[j] <- mean(sqrt((knn_model$fitted.values - test$medv)^2))
  }

  # Almacenar error para cada valor de "k"
  error_total[i] <- mean(error)

}
```

Calcularemos también el error de predicción “dentro de muestra”.

```
error_total_dentro <- vector(length = rango_k)

for (i in 1:rango_k){

  knn_model <- kknn(medv ~ lstat,
                    train = Boston, test = Boston,
                    k = i, kernel='rectangular')

  error_total_dentro[i] <- mean(sqrt((knn_model$fitted.values - Boston$medv)^2))

}
```

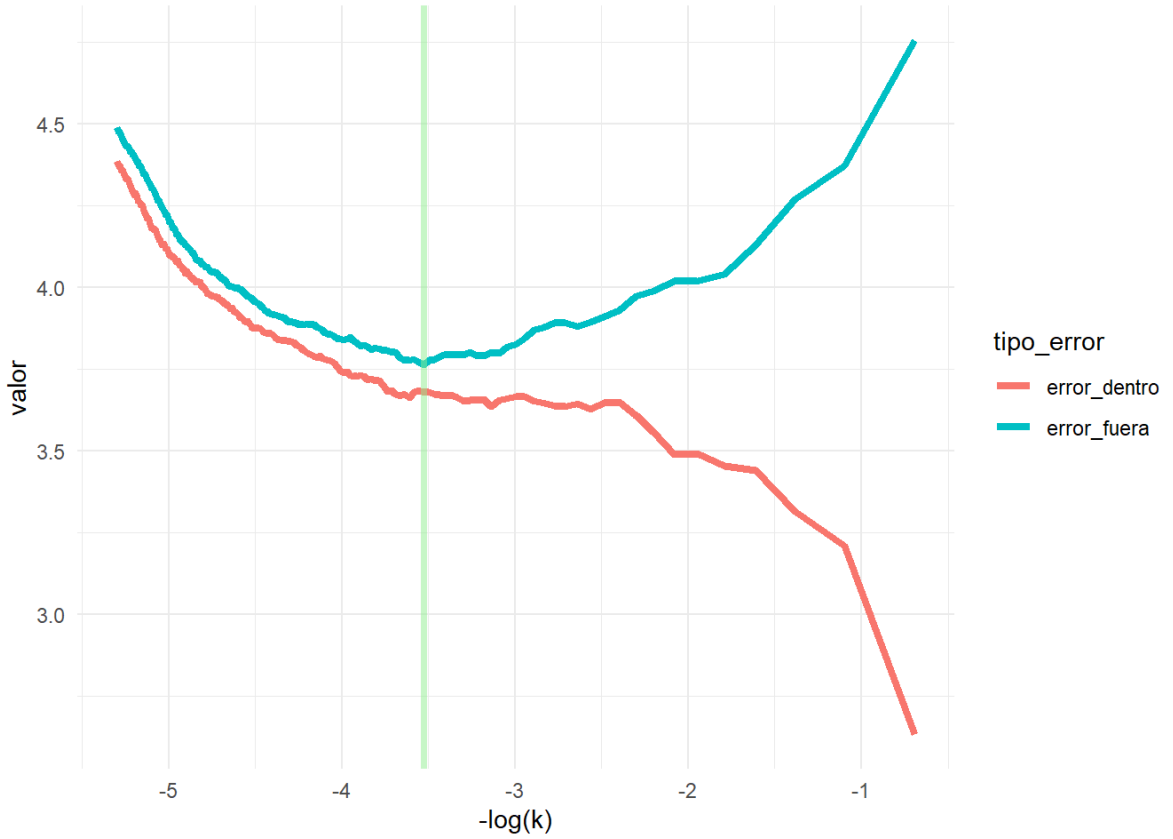
A continuación graficamos el error de predicción dentro y fuera de muestra como función del valor de k. El eje x en este caso es $-log(k)$ con el fin de que yendo de izquierda a derecha se pueda leer como "de menos a más complejo/flexible".

```
errores_dentro_muestra <- error_total_dentro %>%
  as_tibble() %>%
  rename(error_dentro = value) %>%
  mutate(k = 1:rango_k)

errores_fuera_muestra <- error_total %>%
  as_tibble() %>%
  rename(error_fuera = value) %>%
  mutate(k = 1:rango_k)

datos_errores <- errores_fuera_muestra %>%
  left_join(errores_dentro_muestra, by = "k") %>%
  relocate(k, .before = error_fuera) %>%
  pivot_longer(2:3, names_to = "tipo_error", values_to = "valor") %>%
  filter(k >= 2)

datos_errores %>%
  ggplot(aes(x = -log(k), y = valor, col = tipo_error)) +
  geom_line(size = 1.5) +
  geom_vline(xintercept = -log(which.min(error_total)), alpha = 0.5, col = "light green", si
         ze = 1.3) +
  theme_minimal()
```



Se puede ver claramente como el error de predicción dentro de muestra tiende a disminuir siempre con el aumento en la complejidad/flexibilidad del modelo (menores valores de k) mientras que el error fuera de muestra llega a un mínimo y luego comienza a subir. El valor mínimo al que se llega corresponde a un valor de k de 34 y es entonces el valor de complejidad/flexibilidad con el que se alcanza el menor error de predicción.