Entrega Regresión SVM

Arturo González Moya

09 junio, 2021

Contents

1 Introducción

El conjunto de datos "cadata2.csv" recoge información sobre las variables usando todos los grupos de bloques en California del Censo de 1990. En esta muestra, un grupo de bloques en promedio incluye 1425.5 individuos que viven en un área geográficamente compacta. Naturalmente, el área geográfica incluida varía inversamente con la densidad de población. Calculamos las distancias entre los centroides de cada grupo de bloques medidos en latitud y longitud. Se excluyeron todos los grupos de bloques que informaron cero entradas para las variables independientes y dependientes.

En este trabajo se tratará de ajustar el siguiente modelo (donde *ln(median house value)* es la variable dependiente)

$$\ln(\text{median house value}) = a_1 + a_2 * \text{MEDIAN INCOME} + a_3 * \text{MEDIAN INCOME}^2 + a_4 * \text{MEDIAN INCOME}^3$$
$$+ a_5 * \ln(\text{MEDIAN AGE}) + a_6 * \ln(\text{TOTAL ROOMS/POPULATION}) + a_7 * \ln(\text{TOTAL BEDROOMS/POPULATION})$$
$$+ a_8 * \ln(\text{POPULATION/HOUSEHOLDS}) + a_9 * \ln(\text{HOUSEHOLDS})$$

mediante métodos de SVM y otros métodos de regresión.

2 Exploración y limpieza de los datos

Comenzamos la exploración de los datos. Lo primero que haremos será cargar el fichero de datos *cadata2.csv*.

```
datos_ca2 <- read_csv("cadata2.csv")
```

```
## Parsed with column specification:
## cols(
##   median_house_value = col_double(),
##   median_income = col_double(),
##   housing_median_age = col_double(),
##   total_rooms = col_double(),
##   total_bedrooms = col_double(),
##   population = col_double(),
##   households = col_double(),
##   latitude = col_double(),
##   longitude = col_double()
## )
```

```
dim(datos_ca2)
```

```
## [1] 20640     9
```

```
summary(datos_ca2)
```

```
##  median_house_value median_income     housing_median_age  total_rooms
##  Min.   : 14999     Min.   : 0.4999   Min.   : 1.00      Min.   :    2
##  1st Qu.:119600     1st Qu.: 2.5634   1st Qu.:18.00      1st Qu.: 1448
##  Median :179700     Median : 3.5348   Median :29.00      Median : 2127
##  Mean   :206856     Mean   : 3.8707   Mean   :28.64      Mean   : 2636
##  3rd Qu.:264725     3rd Qu.: 4.7432   3rd Qu.:37.00      3rd Qu.: 3148
##  Max.   :500001     Max.   :15.0001   Max.   :52.00      Max.   :39320
##  total_bedrooms     population      households        latitude
##  Min.   :   1.0   Min.   :    3   Min.   :   1.0   Min.   :32.54
##  1st Qu.: 295.0   1st Qu.:  787   1st Qu.: 280.0   1st Qu.:33.93
##  Median : 435.0   Median : 1166   Median : 409.0   Median :34.26
##  Mean   : 537.9   Mean   : 1425   Mean   : 499.5   Mean   :35.63
##  3rd Qu.: 647.0   3rd Qu.: 1725   3rd Qu.: 605.0   3rd Qu.:37.71
##  Max.   :6445.0   Max.   :35682   Max.   :6082.0   Max.   :41.95
##    longitude
##  Min.   :-124.3
##  1st Qu.:-121.8
##  Median :-118.5
##  Mean   :-119.6
##  3rd Qu.:-118.0
##  Max.   :-114.3
```

Este conjunto de datos tiene 20640 observaciones y 9 variables que son las siguientes:

- **median_house_value**: Valor medio de la vivienda (variable numérica).
- **median_income**: Valor ingresos medio (variable numérica).

- **housing_median_age**: Edad media de la vivienda (variable numérica).
- **total_rooms**: Habitaciones totales de la vivienda (variable numérica).
- **total_bedrooms**: Dormitorios totales de la vivienda (variable numérica).
- **population**: Población (variable numérica).
- **households**: Hogares (variable numérica).
- **latitude**: Latitud de la vivienda (variable numérica).
- **longitude**: Longitud de la vivienda (variable numérica).

2.1 Análisis exploratorio de los datos

Veamos primero si exiten valores perdido en el conjunto de datos.

```
anyNA(datos_ca2)
```

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

Podemos observar que no. Además, vemos que todas nuestras variables son numéricas. Pasamos por lo tanto a añadir al conjunto de datos las variables que necesitamos para realizar el modelo anteriormente descrito.

```
datos_ca2 <- mutate(datos_ca2,ln_median_house_value = log(median_house_value),
                    median_income_2 = median_income * median_income,
                    median_income_3 = median_income * median_income * median_income,
                    ln_housing_median_age = log(housing_median_age),
                    ln_tot_rooms_population = log(total_rooms/population),
                    ln_tot_bedrooms_population = log(total_bedrooms/population),
                    ln_population_households = log(population/households),
                    ln_households = log(households))
```

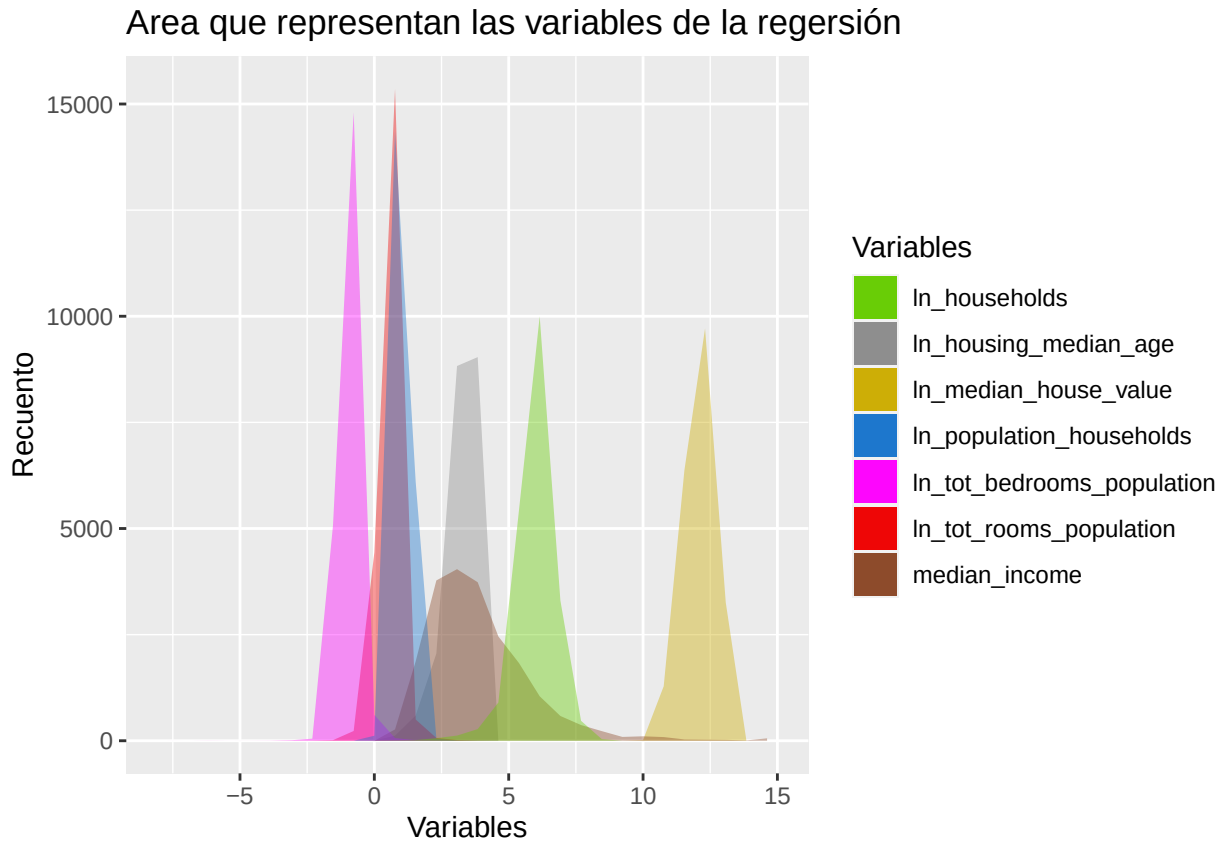
2.1.1 Visualización de los datos

Ya que las variables que vamos a usar en la regresión estan fijadas, vamos a realizar una visualización rápida de ellas. Primero veremos todas las variables que influyen en la regresión (menos la que están al cuadrado y al cubo) en un mismo gráfico.

```
fill <- c("ln_median_house_value" = "gold3", "ln_housing_median_age" = "gray55",
          "median_income" = "sienna4", "ln_tot_rooms_population" = "red2",
          "ln_tot_bedrooms_population" = "magenta1" ,
          "ln_population_households" = "dodgerblue3", "ln_households" = "chartreuse3")
ggplot(datos_ca2)+
  geom_area(mapping = aes(x=ln_median_house_value, fill = "ln_median_house_value"),
            stat = "bin", alpha = 0.4)+
  geom_area(mapping = aes(x=ln_housing_median_age, fill = "ln_housing_median_age"),
            stat = "bin", alpha = 0.4)+
  geom_area(mapping = aes(x=median_income, fill = "median_income"),
            stat = "bin", alpha = 0.4)+
  geom_area(mapping = aes(x=ln_tot_rooms_population, fill = "ln_tot_rooms_population"),
            stat = "bin", alpha = 0.4)+
```

```
  geom_area(mapping = aes(x=ln_tot_bedrooms_population, fill = "ln_tot_bedrooms_population"),
            stat = "bin", alpha = 0.4)+
  geom_area(mapping = aes(x=ln_population_households, fill = "ln_population_households"),
            stat = "bin", alpha = 0.4)+
  geom_area(mapping = aes(x=ln_households, fill = "ln_households"),
            stat = "bin", alpha = 0.4)+
  xlab("Variables") +
  ylab("Recuento") +
  ggtitle("Area que representan las variables de la regersión")+
  scale_fill_manual("Variables",values = fill)+
  theme(legend.position = "right")
```

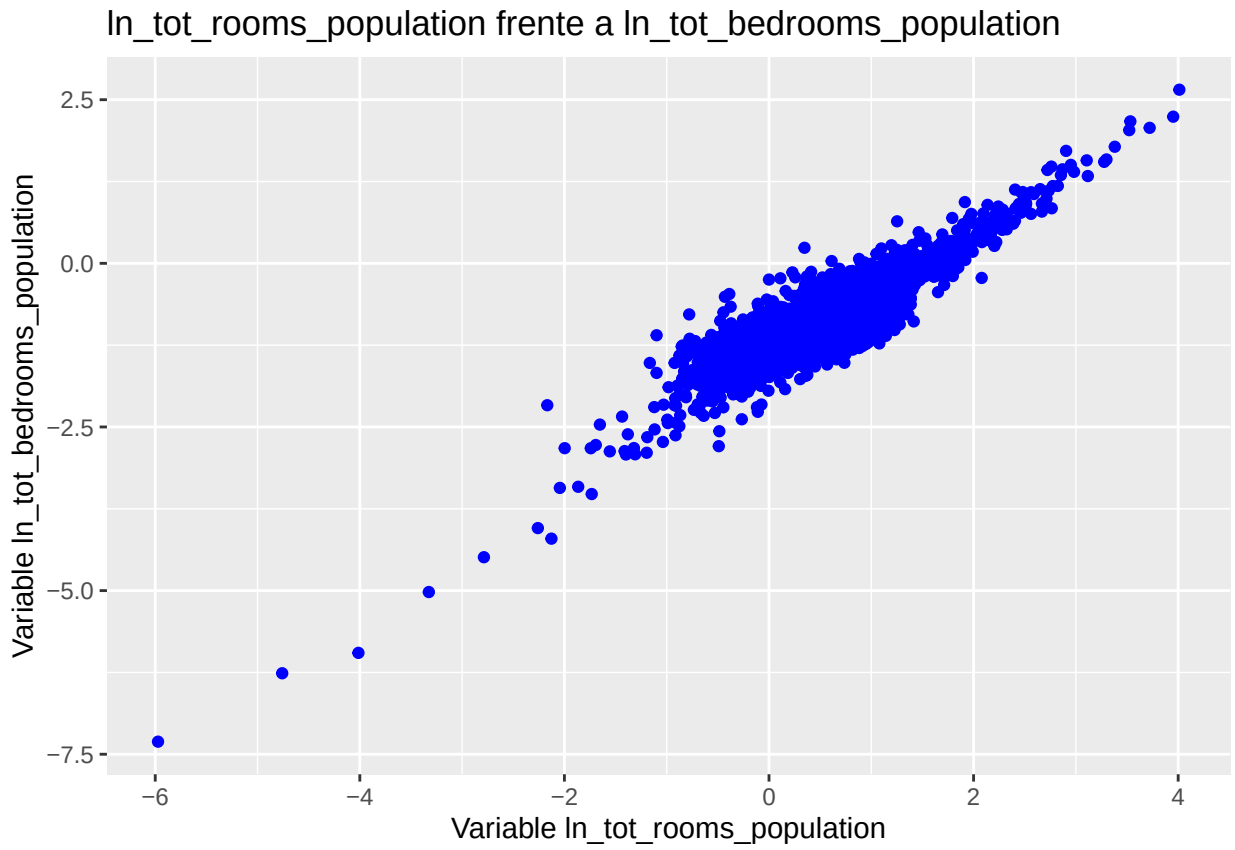
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using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



Este gráfico representa en que parte se acumulan más los datos en las diferentes variables. Podemos ver que las variables *ln_population_households* y *ln_tot_rooms_population* tienen valores muy similares. La variable *ln_median_house_value* (que es la variable independiente de la regresión) es la que tiene los valores más altos.

En el siguiente gráfico veremos representada la variable *ln_tot_rooms_population* frente a la variable *ln_tot_bedrooms_population*.

```
ggplot(datos_ca2, mapping = aes(x=ln_tot_rooms_population, y = ln_tot_bedrooms_population))+
  geom_jitter(color = "blue")+
  xlab("Variable ln_tot_rooms_population") +
  ylab("Variable ln_tot_bedrooms_population") +
  ggtitle("ln_tot_rooms_population frente a ln_tot_bedrooms_population")
```



Podemos ver que en el gráfico podemos trazar una linea recta que aproxime bien los puntos, por lo que nos dice que estas variables estarán muy correlacionadas.

3 Modelos

Antes de comenzar con los modelos, vamos a seleccionar solo las variables que infuyen en él.

```
datos_ca2 <- dplyr::select(datos_ca2, c(ln_median_house_value,
                                        median_income,
                                        median_income_2,
                                        median_income_3,
                                        ln_housing_median_age,
                                        ln_tot_rooms_population,
                                        ln_tot_bedrooms_population,
                                        ln_population_households,
                                        ln_households))
```

3.1 Creación del conjunto de entrenamiento y del conjunto de test

El 75% de los datos originales se utilizan como conjunto de entrenamiento, mientras que el resto (el 25%) se utilizan como conjunto de prueba.

```
set.seed(3456)
trainIndex <- sample(1:nrow(datos_ca2), round(0.75*nrow(datos_ca2), 0))
ca2_train <- datos_ca2[trainIndex, ]
ca2_test <- datos_ca2[-trainIndex, ]
```

3.2 Creación de modelos

3.2.1 Modelos con SVR

Dentro de los modelos que utilizan máquinas de soporte vectorial para regresión, utilizaremos principalmente 2 tipos que son la epsilon-SVR y la nu-SVR. Utilizaremos 3 tipos de kernel diferentes (el lineal, el polinomial y el radial) y los modelos los realizaremos con diferentes costes para ver cual tiene un menor MSE sobre el conjunto de test.

3.2.1.1 Kernel lineal Comenzamos realizando los modelos con diferentes costes utilizando epsilon-SVR con kernel lineal. Iremos guardando los resultados de los coeficientes y de los errores en un data frame. La función `svm` del paquete `e1071` no devuleve directamente los coeficientes de la regresión y para calcularlos hay que hacer lo que podemos ver en este enlace.

```
n <- names(ca2_train)
f <- as.formula(paste("ln_median_house_value ~",
                      paste(n[!n %in% "ln_median_house_value"],
                            collapse = " + "))) # Formula de la regresión

costes <- c(0.001,0.01,0.05,0.1,1,5,10)

df = data.frame()

###### Kernel lineal eps-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "eps-regression", cost = costes[i] ,
               kernel = "linear")
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Realizaremos el estudio de los errores una vez tengamos hayamos añadido todos los modelos de SVR al data frame, pero antes echemosle un vistazo para ver como se va creando.

```
head(df)
```

```
##   median_income median_income_2 median_income_3 ln_housing_median_age
## 1     0.8671352       0.1721901     -0.34500240             0.1653663
## 2     1.3302697      -0.1514361     -0.32501460             0.1708148
## 3     1.5916496      -0.5717580     -0.11513257             0.1702153
## 4     1.6561642      -0.6825569     -0.05822194             0.1694307
## 5     1.7375047      -0.8249035      0.01476748             0.1694911
## 6     1.7426028      -0.8325905      0.01817456             0.1695251
##   ln_tot_rooms_population ln_tot_bedrooms_population ln_population_households
## 1              -0.2998627                  0.2023846               -0.2368648
## 2              -0.5713808                  0.4431543               -0.2248361
## 3              -0.6205255                  0.4964284               -0.2119503
## 4              -0.6278630                  0.5036522               -0.2111343
## 5              -0.6348735                  0.5122252               -0.2087571
## 6              -0.6352714                  0.5126926               -0.2087237
##   ln_households     intercept       mse
## 1    0.10559432 -0.0233436692 0.1321158
## 2    0.08678495 -0.0002793556 0.1244349
## 3    0.08049650  0.0029864108 0.1241798
## 4    0.07820701  0.0033856949 0.1242272
## 5    0.07698062  0.0048284013 0.1243773
## 6    0.07665629  0.0046411626 0.1243867
```

Podemos ver que las columnas se corresponden con los coeficientes de las variables de la regresión, el intercepto y el MSE. En las filas se encuentran cada uno de los modelos creados. Al final les añadiremos los nombre a las filas ya que ahora no aparecen.

Continuamos realizando los modelos con diferentes costes utilizando ahora nu-SVR con kernel lineal.

```
###### Kernel lineal nu-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "nu-regression", cost = costes[i] ,
               kernel = "linear")
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Se han añadido 7 modelos más al data frame con los coeficientes de las regresiones y los errores de la predicciones.

3.2.1.2 Kernel radial Pasamos a construir modelos con el kernel radial, comenzado con el método epsilon-SVR.

```
###### Kernel radial eps-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "eps-regression", cost = costes[i] ,
               kernel = "radial")
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
```

```
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Se han añadido 7 modelos más al data frame con los coeficientes de las regresiones y los errores de la predicciones.

Continuamos realizando los modelos con diferentes costes utilizando ahora nu-SVR con kernel radial.

```
###### Kernel radial nu-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "nu-regression", cost = costes[i] ,
               kernel = "radial")
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Se han añadido 7 modelos más al data frame con los coeficientes de las regresiones y los errores de la predicciones.

3.2.1.3 Kernel polinomial

Por último, vamos a construir modelos con el kernel polinomial de grado 3, comenzado con el método epsilon-SVR.

```
###### Kernel polinomial eps-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "eps-regression", cost = costes[i] ,
               kernel = "polynomial", degree = 2)
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Se han añadido 7 modelos más al data frame con los coeficientes de las regresiones y los errores de la predicciones.

Realizamos los últimos modelos con diferentes costes utilizando ahora nu-SVR con kernel polinomial de grado 3.

```
###### Kernel polinomial nu-reg
for (i in 1:length(costes)) {
  modelo = svm(f, data = ca2_train, type = "nu-regression", cost = costes[i] ,
```

```
                kernel = "polynomial", degree = 3)
  pred <-predict(modelo, ca2_test)
  w = t(modelo$coefs) %*% modelo$SV
  b = modelo$rho
  w = cbind(w,"intercept" = b)
  mse <- mean((pred-ca2_test$ln_median_house_value)^2)
  w = cbind(w,"mse" = mse)
  df= rbind(df, w)
}
```

Se han añadido los 7 últimos modelos de SVR al data frame con los coeficientes de las regresiones y los errores de las predicciones.

Ya que cuando hemos ido guardando los modelos en el data frame no les hemos añadido el nombre en la fila, lo haremos ahora.

```
nombre <- c(rep("lineal_",14),rep("radial_",14),rep("polinomial_",14))
coste_nombre <- c(rep(c("0.001","0.01","0.05","0.1","1","5","10"), 6))
metodo <- c(rep(c(rep("eps_", 7),rep("nu_", 7)), 3))

rownames(df) <- paste0(metodo, nombre, coste_nombre)
```

Con todos los modelos de SVR, echemos un vistazo a la tabla.

```
df
```

```
##                     median_income median_income_2 median_income_3
## eps_lineal_0.001     8.671352e-01       0.1721901   -3.450024e-01
## eps_lineal_0.01      1.330270e+00      -0.1514361   -3.250146e-01
## eps_lineal_0.05      1.591650e+00      -0.5717580   -1.151326e-01
## eps_lineal_0.1       1.656164e+00      -0.6825569   -5.822194e-02
## eps_lineal_1         1.737505e+00      -0.8249035    1.476748e-02
## eps_lineal_5         1.742603e+00      -0.8325905    1.817456e-02
## eps_lineal_10        1.744986e+00      -0.8360943    1.977259e-02
## nu_lineal_0.001      8.101948e-01       0.1829397   -3.039386e-01
## nu_lineal_0.01       1.287794e+00      -0.0642132   -3.671798e-01
## nu_lineal_0.05       1.536215e+00      -0.4521339   -1.850038e-01
## nu_lineal_0.1        1.604365e+00      -0.5657383   -1.295271e-01
## nu_lineal_1          1.718427e+00      -0.7753247   -1.676284e-02
## nu_lineal_5          1.739698e+00      -0.8144646    4.000733e-03
## nu_lineal_10         1.742283e+00      -0.8168067    4.786787e-03
## eps_radial_0.001     5.514176e+00       4.6409907    3.537815e+00
## eps_radial_0.01      2.318682e+01      21.9435847    1.994272e+01
## eps_radial_0.05      5.217670e+01      55.6438486    5.735981e+01
## eps_radial_0.1       6.708513e+01      72.5006279    7.552654e+01
## eps_radial_1         1.371922e+02     157.8390148    1.733964e+02
## eps_radial_5         1.456766e+02     167.4027948    1.811929e+02
## eps_radial_10        1.386235e+02     161.9163999    1.727574e+02
## nu_radial_0.001      4.867155e+00       4.1650126    3.243713e+00
## nu_radial_0.01       2.111927e+01      20.1668783    1.849859e+01
## nu_radial_0.05       4.785955e+01      50.1380810    5.123784e+01
## nu_radial_0.1        6.182632e+01      65.2432314    6.729908e+01
## nu_radial_1          1.248492e+02     140.6853646    1.533001e+02
```

```
## nu_radial_5           1.309856e+02     145.3504234    1.557774e+02
## nu_radial_10          1.252621e+02     139.7821026    1.476186e+02
## eps_polinomial_0.001  6.998094e+00       4.9965179    2.873196e+00
## eps_polinomial_0.01   6.305862e+01      42.5747784    2.305080e+01
## eps_polinomial_0.05   3.068381e+02     205.2932499    1.102914e+02
## eps_polinomial_0.1    6.047886e+02     404.2338152    2.170097e+02
## eps_polinomial_1      4.879598e+03    3266.4999922    1.748652e+03
## eps_polinomial_5      1.480367e+04    9931.4534438    5.297299e+03
## eps_polinomial_10     2.041409e+04   13702.7702382    7.301544e+03
## nu_polinomial_0.001   4.547858e+00       3.1589362    1.755546e+00
## nu_polinomial_0.01    3.422303e+01      22.3696972    1.140782e+01
## nu_polinomial_0.05    1.361353e+02      84.1615630    3.954038e+01
## nu_polinomial_0.1     2.516029e+02     153.8649748    7.083057e+01
## nu_polinomial_1       1.837910e+03    1110.8060052    4.941284e+02
## nu_polinomial_5       7.528206e+03    4553.8708537    2.008524e+03
## nu_polinomial_10      1.392264e+04    8437.6914304    3.721788e+03
##                      ln_housing_median_age ln_tot_rooms_population
## eps_lineal_0.001                 0.1653663              -0.2998627
## eps_lineal_0.01                  0.1708148              -0.5713808
## eps_lineal_0.05                  0.1702153              -0.6205255
## eps_lineal_0.1                   0.1694307              -0.6278630
## eps_lineal_1                     0.1694911              -0.6348735
## eps_lineal_5                     0.1695251              -0.6352714
## eps_lineal_10                    0.1694719              -0.6358214
## nu_lineal_0.001                  0.1597062              -0.2580444
## nu_lineal_0.01                   0.1683853              -0.5638262
## nu_lineal_0.05                   0.1669725              -0.6174883
## nu_lineal_0.1                    0.1675234              -0.6251232
## nu_lineal_1                      0.1655223              -0.6375298
## nu_lineal_5                      0.1653293              -0.6389947
## nu_lineal_10                     0.1650007              -0.6406067
## eps_radial_0.001                 1.0790317               1.6134863
## eps_radial_0.01                  3.2222377               2.5910048
## eps_radial_0.05                  3.8045383               7.0667161
## eps_radial_0.1                   1.2130451              10.4381753
## eps_radial_1                    -4.0358603              11.7943681
## eps_radial_5                    -3.4131119              35.1956866
## eps_radial_10                   -1.2432907              47.5977391
## nu_radial_0.001                  0.7851549               1.5121037
## nu_radial_0.01                   2.9807562               2.2584426
## nu_radial_0.05                   2.6871685               6.0072699
## nu_radial_0.1                    0.3206125               7.1884376
## nu_radial_1                     -4.8488273               8.3123208
## nu_radial_5                     -8.2227933              26.8818132
## nu_radial_10                    -3.2504506              33.7126482
## eps_polinomial_0.001             0.8779154               3.9315745
## eps_polinomial_0.01             10.5043330              33.8353797
## eps_polinomial_0.05             52.5866238             161.0877522
## eps_polinomial_0.1             103.8343262             319.0533438
## eps_polinomial_1              1295.0742325            2723.3483018
## eps_polinomial_5              9236.5551584            9712.8418998
## eps_polinomial_10            20311.6760884           15856.0220119
## nu_polinomial_0.001              0.8883049               2.0982546
## nu_polinomial_0.01               8.8264916              14.0279491
```

```
## nu_polinomial_0.05              42.3479258              57.5245763
## nu_polinomial_0.1               85.9804941             110.9076435
## nu_polinomial_1                846.1284780             907.4655556
## nu_polinomial_5               4062.3829975            4095.8748739
## nu_polinomial_10              7987.0191161            7526.8958628
##                       ln_tot_bedrooms_population ln_population_households
## eps_lineal_0.001                       0.2023846            -2.368648e-01
## eps_lineal_0.01                        0.4431543            -2.248361e-01
## eps_lineal_0.05                        0.4964284            -2.119503e-01
## eps_lineal_0.1                         0.5036522            -2.111343e-01
## eps_lineal_1                           0.5122252            -2.087571e-01
## eps_lineal_5                           0.5126926            -2.087237e-01
## eps_lineal_10                          0.5132331            -2.087064e-01
## nu_lineal_0.001                        0.1784797            -2.328933e-01
## nu_lineal_0.01                         0.4384118            -2.281549e-01
## nu_lineal_0.05                         0.4930794            -2.186228e-01
## nu_lineal_0.1                          0.5017748            -2.158224e-01
## nu_lineal_1                            0.5123719            -2.143221e-01
## nu_lineal_5                            0.5166889            -2.120201e-01
## nu_lineal_10                           0.5179614            -2.115625e-01
## eps_radial_0.001                       0.8434460            -1.879288e+00
## eps_radial_0.01                        1.0038981            -5.551096e+00
## eps_radial_0.05                       -1.8518413            -7.922844e+00
## eps_radial_0.1                        -3.1865367            -9.804988e+00
## eps_radial_1                         -38.2411647             1.028537e+01
## eps_radial_5                         -39.0864712             8.704079e+00
## eps_radial_10                        -41.4265053             5.319707e+00
## nu_radial_0.001                        0.6813632            -1.567726e+00
## nu_radial_0.01                         0.7345658            -4.763760e+00
## nu_radial_0.05                        -1.8066517            -7.035136e+00
## nu_radial_0.1                         -5.6462488            -7.290232e+00
## nu_radial_1                          -39.5225510             1.776077e+01
## nu_radial_5                          -33.5770129             1.968433e+01
## nu_radial_10                         -37.0307362             2.285595e+01
## eps_polinomial_0.001                   2.0000038            -2.974007e+00
## eps_polinomial_0.01                   16.5064830            -2.567131e+01
## eps_polinomial_0.05                   74.8197665            -1.198587e+02
## eps_polinomial_0.1                   150.3364707            -2.397291e+02
## eps_polinomial_1                    1854.2489450            -2.728866e+03
## eps_polinomial_5                   11621.0884076            -1.550153e+04
## eps_polinomial_10                  25501.2995771            -3.266232e+04
## nu_polinomial_0.001                    1.2025342            -2.068795e+00
## nu_polinomial_0.01                    10.0418551            -1.744364e+01
## nu_polinomial_0.05                    49.0259443            -8.133050e+01
## nu_polinomial_0.1                     99.7051000            -1.601815e+02
## nu_polinomial_1                     1011.7415600            -1.481071e+03
## nu_polinomial_5                     5010.3338110            -7.000068e+03
## nu_polinomial_10                    9654.8163820            -1.346169e+04
##                       ln_households     intercept       mse
## eps_lineal_0.001      1.055943e-01 -0.0233436692 0.1321158
## eps_lineal_0.01       8.678495e-02 -0.0002793556 0.1244349
## eps_lineal_0.05       8.049650e-02  0.0029864108 0.1241798
## eps_lineal_0.1        7.820701e-02  0.0033856949 0.1242272
## eps_lineal_1          7.698062e-02  0.0048284013 0.1243773
```

```
## eps_lineal_5          7.665629e-02  0.0046411626 0.1243867
## eps_lineal_10         7.681333e-02  0.0047119918 0.1244017
## nu_lineal_0.001       1.002545e-01 -0.0039448225 0.1343531
## nu_lineal_0.01        8.431815e-02  0.0014083096 0.1246274
## nu_lineal_0.05        7.964659e-02  0.0037604535 0.1243413
## nu_lineal_0.1         7.794159e-02  0.0037784805 0.1243504
## nu_lineal_1           7.607982e-02  0.0046691786 0.1244719
## nu_lineal_5           7.493758e-02  0.0048403992 0.1244893
## nu_lineal_10          7.493319e-02  0.0050051606 0.1244927
## eps_radial_0.001      1.295273e+00 -0.2189103593 0.1879357
## eps_radial_0.01       4.834151e+00 -0.3877684844 0.1221816
## eps_radial_0.05       9.435085e+00 -0.4565492503 0.1082407
## eps_radial_0.1        1.280934e+01 -0.4332849211 0.1056513
## eps_radial_1          3.513979e+01 -0.4269261158 0.1020321
## eps_radial_5          7.325033e+01 -0.4356463075 0.1022869
## eps_radial_10         8.641820e+01 -0.4911348968 0.1034360
## nu_radial_0.001       1.061805e+00 -0.1774354415 0.1969699
## nu_radial_0.01        4.251708e+00 -0.3846100413 0.1256445
## nu_radial_0.05        9.075876e+00 -0.4374675547 0.1093485
## nu_radial_0.1         1.141707e+01 -0.4200900565 0.1064517
## nu_radial_1           2.778935e+01 -0.3625134691 0.1021670
## nu_radial_5           5.891263e+01 -0.3589448825 0.1016856
## nu_radial_10          6.527296e+01 -0.4230627457 0.1024811
## eps_polinomial_0.001  1.481924e+00 -0.0203563111 0.2923887
## eps_polinomial_0.01   1.388009e+01  0.0204143177 0.2812626
## eps_polinomial_0.05   6.609793e+01  0.0367650695 0.2891082
## eps_polinomial_0.1    1.316947e+02  0.0413373106 0.2910190
## eps_polinomial_1      1.258862e+03  0.0820136547 0.2499569
## eps_polinomial_5      6.156678e+03  0.1927217996 0.1854983
## eps_polinomial_10     1.258909e+04  0.2689406935 0.1639668
## nu_polinomial_0.001   9.633454e-01  0.0102940527 0.6557654
## nu_polinomial_0.01    9.336144e+00  0.0310411729 1.3705859
## nu_polinomial_0.05    4.447879e+01  0.0515032087 1.4627173
## nu_polinomial_0.1     8.830128e+01  0.0555272883 1.3136952
## nu_polinomial_1       7.809297e+02  0.0579153763 1.3852566
## nu_polinomial_5       3.576100e+03  0.0516147809 1.1172810
## nu_polinomial_10      7.025470e+03  0.0650690594 0.9443754
```

Veamos cúal es el modelo que menor error tiene sobre el conjunto de prueba.

```
which.min(df$mse)
```

```
## [1] 27
```

```
mse_minimo_svr <- df[which.min(df$mse),ncol(df)]

nombre_mejor_modelo <- row.names(df)[which.min(df$mse)]
nombre_mejor_modelo
```

```
## [1] "nu_radial_5"
```

El mejor modelo considerando el error sobre el conjunto de prueba es el número 27 del data set, que corresponde con el modelo de nu-SVR, utilizando kernel radial y con un coste de 5. Este error es de 0.1016856. Estudiemos los coeficientes de regresión obtenidos con este modelo.

```
df[which.min(df$mse),-ncol(df)]
```

```
##             median_income median_income_2 median_income_3 ln_housing_median_age
## nu_radial_5      130.9856        145.3504        155.7774             -8.222793
##             ln_tot_rooms_population ln_tot_bedrooms_population
## nu_radial_5                26.88181                  -33.57701
##             ln_population_households ln_households  intercept
## nu_radial_5                 19.68433      58.91263 -0.3589449
```

Vemos que las variables *median_income*, *median_income_2*, *median_income_3*, *ln_tot_rooms_population*, *ln_population_households* y *ln_households* tienen un coeficiente positivo, mientras que las variables *ln_housing_median_age*, *ln_tot_bedrooms_population* y el intercepto son negativos. Los coeficientes más grandes en valor absoluto son los correspondientes a las variables *median_income*, *median_income_2* y *median_income_3*.

3.2.2 Otros método de regresión

Una vez finalizados los modelos de SVR, vamos a pasar a utilizar otros métodos de regresión como pueden ser el arbol de regresión, la regresión lineal, los bosques aleatorios o las redes neuronales.

3.2.2.1 Arbol regresión El primero método de regresión que aplicaremos será el arbol de regresión.

```
set.seed(3456)
tree2 <- rpart(f, data = ca2_train, cp = 1e-8)
pred <- predict(tree2, newdata = ca2_test)
mse_reg_tree <- mean((pred-ca2_test$ln_median_house_value)^2)
mse_reg_tree
```

```
## [1] 0.1419695
```

El error obtenido con el arbol de regresión es de 0.1419695. Podemos observar que este error es mayor que el obtenido con el mejor modelo de SVR que era de 0.1016856.

3.2.2.2 Regreseion lineal Pasamos ahora a realizar una regresión lineal. Veremos los coeficientes de las regresión y el error cometido en el conjunto de prueba.

```
reg.model<-lm(ln_median_house_value~.,data=ca2_train)
pred<-predict(reg.model,ca2_test)

squaredError.reg <- (pred-ca2_test$ln_median_house_value)^2
mse_reg <- mean(squaredError.reg)
summary(reg.model)$coefficients[,1]
```

```
##               (Intercept)             median_income
##             11.4586097377              0.4816533795
##           median_income_2           median_income_3
##             -0.0172265056             -0.0001331981
##     ln_housing_median_age   ln_tot_rooms_population
##              0.1571041603             -0.8497950495
```

```
## ln_tot_bedrooms_population   ln_population_households
##               0.7889357938              -0.4235721251
##              ln_households
##               0.0523676723
```

```
mse_reg
```

```
## [1] 0.1234448
```

Vemos que el error obtenido con el conjunto de prueba en la regresión lineal es de 0.1234448, que es mayor que el error obtenido con el mejor modelo de SVR.

Recordemos cuales eran los coeficientes del mejor modelo de SVR.

```
df[which.min(df$mse),-ncol(df)]
```

```
##             median_income median_income_2 median_income_3 ln_housing_median_age
## nu_radial_5      130.9856        145.3504        155.7774             -8.222793
##             ln_tot_rooms_population ln_tot_bedrooms_population
## nu_radial_5                26.88181                  -33.57701
##             ln_population_households ln_households  intercept
## nu_radial_5                 19.68433      58.91263 -0.3589449
```

Vemos que en la regresión lineal, los coeficientes son mucho mas pequeños en valor absoluto que los coeficientes obtenidos con el mejor modelo de SVR.

3.2.2.3 Bosques aleatorios Pasamos ahora a realizar un modelos utilizando bosques aleatorios. El número de arboles utilizado será de 500.

```
set.seed(3456)
rf3 <- randomForest(f, data = ca2_train, ntree = 500)
pred <- predict(rf3, newdata = ca2_test)
mse_reg_rf <- mean((pred-ca2_test$ln_median_house_value)^2)
mse_reg_rf
```

```
## [1] 0.1062535
```

Podemos observar que el error obtenido sobre el conjunto de prueba es de 0.1062535. Vemos que este error es ligeramente mayor que el obtenido con el mejor modelo de SVR (que era de 0.1016856).

3.2.2.4 Redes neuronales Por último, se utilizará una red neuronal para intentar estimar el valor medio de la vivienda. Este red tendrá un tamaño de 30. El número de iteraciones máximas para calcular el peso de las neuronas será de 500.

```
set.seed(3456)
nn1 <- nnet(f, data = ca2_train, size = 30, maxit = 500, linout = TRUE)
```

```
## # weights:  301
## initial  value 2476390.638283
## iter  10 value 3510.259152
```

```
## iter  20 value 2859.835748
## iter  30 value 2365.490460
## iter  40 value 1917.430556
## iter  50 value 1865.648585
## iter  60 value 1816.939006
## iter  70 value 1774.518478
## iter  80 value 1725.429308
## iter  90 value 1711.993517
## iter 100 value 1704.368871
## iter 110 value 1692.793870
## iter 120 value 1684.370303
## iter 130 value 1672.709359
## iter 140 value 1648.343213
## iter 150 value 1635.567829
## iter 160 value 1631.187237
## iter 170 value 1625.037627
## iter 180 value 1619.464177
## iter 190 value 1613.703954
## iter 200 value 1607.558772
## iter 210 value 1600.698688
## iter 220 value 1595.496540
## iter 230 value 1589.251243
## iter 240 value 1582.972231
## iter 250 value 1576.738562
## iter 260 value 1569.997144
## iter 270 value 1561.726110
## iter 280 value 1556.082256
## iter 290 value 1552.163552
## iter 300 value 1549.586836
## iter 310 value 1546.723657
## iter 320 value 1544.704648
## iter 330 value 1542.628656
## iter 340 value 1540.398095
## iter 350 value 1538.958908
## iter 360 value 1537.814466
## iter 370 value 1536.714898
## iter 380 value 1535.623447
## iter 390 value 1534.579331
## iter 400 value 1533.609708
## iter 410 value 1532.187911
## iter 420 value 1530.661662
## iter 430 value 1529.297443
## iter 440 value 1528.455470
## iter 450 value 1527.433837
## iter 460 value 1526.594095
## iter 470 value 1525.874801
## iter 480 value 1525.195132
## iter 490 value 1524.444730
## iter 500 value 1523.791318
## final  value 1523.791318
## stopped after 500 iterations
```

```
nn1.pred <- predict(nn1, newdata = ca2_test)
```

```r
mse_reg_nn <- mean((nn1.pred-ca2_test$ln_median_house_value)^2)
mse_reg_nn
```

```
## [1] 0.1223209
```

Observamos que el error de la red es de 0.1223209, que es ligeramente mayor al mejor modelo utilizando SVR.

4 Conclusiones

Para finaliar este trabajo, decimos que el mejor modelo de todos los que se han considerado es el modelo de nu-SVR, utilizando kernel radial y con un coste de 5. Otros modelos de SVR y el modelo de bosques aleatorios también obtienen un error muy parecido al que se obtiene con este. Veamos esto en la siguiente tabla.

```r
resumen <- data.frame("MSE" = df[, ncol(df)])

resumen <- rbind(resumen, mse_reg_tree, mse_reg, mse_reg_rf, mse_reg_nn)

rownames(resumen) <- c(rownames(df), "Arbol_decision",
                       "Regresion_lineal", "Bosques_Aleatorios", "Red_Neuronal")

kable( resumen , caption = "Tabla resumen de los errores de los modelos"
       , row.names = TRUE
      )
```

Table 1: Tabla resumen de los errores de los modelos

	MSE
eps_lineal_0.001	0.1321158
eps_lineal_0.01	0.1244349
eps_lineal_0.05	0.1241798
eps_lineal_0.1	0.1242272
eps_lineal_1	0.1243773
eps_lineal_5	0.1243867
eps_lineal_10	0.1244017
nu_lineal_0.001	0.1343531
nu_lineal_0.01	0.1246274
nu_lineal_0.05	0.1243413
nu_lineal_0.1	0.1243504
nu_lineal_1	0.1244719
nu_lineal_5	0.1244893
nu_lineal_10	0.1244927
eps_radial_0.001	0.1879357
eps_radial_0.01	0.1221816
eps_radial_0.05	0.1082407
eps_radial_0.1	0.1056513
eps_radial_1	0.1020321
eps_radial_5	0.1022869
eps_radial_10	0.1034360

	MSE
nu_radial_0.001	0.1969699
nu_radial_0.01	0.1256445
nu_radial_0.05	0.1093485
nu_radial_0.1	0.1064517
nu_radial_1	0.1021670
nu_radial_5	0.1016856
nu_radial_10	0.1024811
eps_polinomial_0.001	0.2923887
eps_polinomial_0.01	0.2812626
eps_polinomial_0.05	0.2891082
eps_polinomial_0.1	0.2910190
eps_polinomial_1	0.2499569
eps_polinomial_5	0.1854983
eps_polinomial_10	0.1639668
nu_polinomial_0.001	0.6557654
nu_polinomial_0.01	1.3705859
nu_polinomial_0.05	1.4627173
nu_polinomial_0.1	1.3136952
nu_polinomial_1	1.3852566
nu_polinomial_5	1.1172810
nu_polinomial_10	0.9443754
Arbol_decision	0.1419695
Regresion_lineal	0.1234448
Bosques_Aleatorios	0.1062535
Red_Neuronal	0.1223209

El método que mejor predice el precio medio de la vivienda es el modelo de nu-SVR, utilizando kernel radial y con un coste de 5.