Curso : MACHINE LEARNING CON R

Tema : Modelos de Predicción/ Regresión.

Docente : Victor Henostroza

**GUIA DE LABORATORIO 3**

**Objetivos del laboratorio:**

1. **Desarrollar un Modelo de Regresión Lineal Múltiple.**
2. **Desarrollar un Modelo de Regresión con Máquinas de Soporte Vectorial.**
3. **Desarrollar un Modelo de Regresión con Árboles de Decisión.**
4. **Desarrollar un Modelo de Regresión con Bosques Aleatorios.**
5. **Desarrollar un Modelo de Regresión con Red Neuronal Artificial.**
6. **Desarrollar un Modelo de Regresión con K-Nearest Neighbors.**

**------------------------------------------------------------------------------------------------------**

**Objetivo 1:**

Para nuestro ejemplo utilizaremos el dataset “auto-mpg.csv” que tenemos en la carpeta del laboratorio 2, por lo tanto, hacemos lo siguiente

**#Regresión Lineal Múltiple**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8] #Excluimos las columnas No y car\_name por no ser significativas

**# Generando variables dummy para las variables categóricas**

# La variable cylinders la tomamos como cualitativa ordinal por ser una característica ordenable

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Escalando las variables independientes entre 0 y 1 (0 valor mín y 1 valor máx)**

# Esta es otra forma de escalar utilizada para cuando ya tienes variables dummy

# install.packages("caret")

library(caret)

# install.packages("scales")

library(scales)

dataset$displacement = rescale(dataset$displacement)

dataset$horsepower = rescale(dataset$horsepower)

dataset$weight = rescale(dataset$weight)

dataset$acceleration = rescale(dataset$acceleration)

dataset$model\_year = rescale(dataset$model\_year)

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(123)

split = sample.split(dataset$mpg, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE) # 80% de los datos para entrenar

testing\_set = subset(dataset, split == FALSE) # 20% de los datos para validar

**# Ajustar el modelo de Regresión Lineal Múltiple con el Conjunto de Entrenamiento**

regression = lm(formula = mpg ~ ., data = training\_set)

Analizamos por consola nuestro modelo creado:

> summary(regression)

Call:

lm(formula = mpg ~ ., data = training\_set)

Residuals:

Min 1Q Median 3Q Max

-9.2936 -2.1808 -0.0855 1.7960 13.7491

Coefficients: (1 not defined because of singularities)

Estimate Std. Error t value Pr(>|t|)

(Intercept) 27.1373 1.9833 13.683 <2e-16 \*\*\*

cylinders.3 -5.2870 2.5333 -2.087 0.0377 \*

cylinders.4 0.5978 1.4060 0.425 0.6710

cylinders.5 2.3138 2.2933 1.009 0.3137

cylinders.6 -2.4157 0.9341 -2.586 0.0101 \*

cylinders.8 NA NA NA NA

displacement 5.1386 3.1497 1.631 0.1037

horsepower -2.1054 2.7853 -0.756 0.4502

weight -25.4039 2.7564 -9.216 <2e-16 \*\*\*

acceleration 2.3084 1.8209 1.268 0.2058

model\_year 8.7763 0.6547 13.405 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.318 on 328 degrees of freedom

Multiple R-squared: 0.8299, Adjusted R-squared: 0.8253

F-statistic: 177.8 on 9 and 328 DF, p-value: < 2.2e-16

Apreciamos los diversos niveles de aporte que tienen las variables (mientras más estrellitas presenta es más significativa la variable para el modelo). Este modelo nos está arrojando un R cuadrado ajustado de 0.8253 y se valida con su p-valor < 2.2e-16 que es menor al 5% de valor de significación.

**Observación:** Apreciamos que las variables “cylinders” (3 cilindros y 6 cilindros), “weight” y “model\_year” son significativas por tener un p\_valor menor al nivel de significación del 5%. Las demás variables, al tener un p\_valor superior al nivel de significación no aportan significativamente al modelo y posiblemente deberían ser excluidas del mismo.

**# Predecir los resultados con el conjunto de testing**

y\_pred = predict(regression, newdata = testing\_set)

> y\_pred

2 6 9 13 15 19 21 32 40

18.34738 24.63101 15.11165 18.37669 11.74657 11.40530 20.21601 13.73804 29.67320

45 51 58 61 64 65 68 72 81

24.80892 26.37652 27.43861 34.04100 12.46350 14.37007 11.31846 20.94944 29.39047

Ahora, aplicaremos la eliminación de variables hacia atrás con una función “step” de R para poder generar un modelo final de regresión con las variables más significativas y dejando de lado a las no significativas.

**# Hallando el mejor modelo de regresión con eliminación hacia atrás**

SL = 0.05

library(MASS)

modelo\_final = stepAIC(regression, direction = "forward")

regression = modelo\_final

Con las sentencias anteriormente escritas obtendremos la visualización de como el modelo ha generado la eliminación hacia atrás, obteniéndose en consola lo siguiente:

Start: AIC=820.59

mpg ~ cylinders.3 + cylinders.4 + cylinders.5 + cylinders.6 +

cylinders.8 + displacement + horsepower + weight + acceleration +

model\_year

Step: **AIC=820.59**

**# Elimina cylinders.8 por tener AIC menor al global y ser el más bajo**

mpg ~ cylinders.3 + cylinders.4 + cylinders.5 + cylinders.6 +

displacement + horsepower + weight + acceleration + model\_year

Df Sum of Sq RSS AIC

- **cylinders.4 1 1.99 3612.7 818.77**

- horsepower 1 6.29 3617.0 819.18

- cylinders.5 1 11.21 3621.9 819.64

- acceleration 1 17.69 3628.4 820.24

<none> 3610.7 820.59

- displacement 1 29.30 3640.0 821.32

- cylinders.3 1 47.95 3658.6 823.05

- cylinders.6 1 73.62 3684.3 825.41

- weight 1 935.03 4545.7 896.43

- model\_year 1 1978.03 5588.7 966.24

Step: **AIC=818.77**

**# Elimina horspower por tener AIC menor al global y ser el más bajo**

mpg ~ cylinders.3 + cylinders.5 + cylinders.6 + displacement +

horsepower + weight + acceleration + model\_year

Df Sum of Sq RSS AIC

- **horsepower 1 6.32 3619.0 817.37**

- cylinders.5 1 9.35 3622.0 817.65

- acceleration 1 17.91 3630.6 818.45

<none> 3612.7 818.77

- displacement 1 35.15 3647.8 820.05

- cylinders.3 1 96.81 3709.5 825.71

- cylinders.6 1 358.19 3970.9 848.73

- weight 1 934.83 4547.5 894.56

- model\_year 1 1976.65 5589.3 964.28

Step: **AIC=817.37**

**# Elimina cylinders.5 por tener AIC menor al global y ser el más bajo**

mpg ~ cylinders.3 + cylinders.5 + cylinders.6 + displacement +

weight + acceleration + model\_year

Df Sum of Sq RSS AIC

- **cylinders.5 1 10.48 3629.5 816.34**

<none> 3619.0 817.37

- displacement 1 31.37 3650.4 818.28

- acceleration 1 47.59 3666.6 819.78

- cylinders.3 1 99.50 3718.5 824.53

- cylinders.6 1 360.06 3979.1 847.43

- weight 1 1347.52 4966.5 922.35

- model\_year 1 2171.24 5790.2 974.22

Step: **AIC=816.34**

**# Elimina displacemente porque a pesar de no ser menor el AICes casi el mismo que el anterior y es mejor tener menos variables cuando casi no se mueve la predicción.**

mpg ~ cylinders.3 + cylinders.6 + displacement + weight + acceleration +

model\_year

Df Sum of Sq RSS AIC

<none> 3629.5 816.34

- **displacement 1 27.70 3657.2 816.91**

- acceleration 1 50.35 3679.8 819.00

- cylinders.3 1 100.99 3730.5 823.62

- cylinders.6 1 370.19 3999.7 847.17

- weight 1 1341.18 4970.7 920.63

- model\_year 1 2182.20 5811.7 973.47

**Finalmente, apreciamos en consola los resultados del modelo final:**

> summary(regression)

Call:

lm(formula = mpg ~ cylinders.3 + cylinders.6 +

weight + acceleration + model\_year, data = training\_set)

Residuals:

Min 1Q Median 3Q Max

-9.3294 -2.1535 -0.0846 1.8140 13.6672

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 27.1275 0.8897 30.489 < 2e-16 \*\*\*

cylinders.3 -6.0636 1.9981 -3.035 0.0026 \*\*

cylinders.6 -2.6846 0.4620 -5.810 1.46e-08 \*\*\*

weight -26.1086 2.3607 -11.060 < 2e-16 \*\*\*

acceleration 3.1932 1.4901 2.143 0.0328 \*

model\_year 8.9159 0.6320 14.107 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.311 on 331 degrees of freedom

Multiple R-squared: 0.829, Adjusted R-squared: 0.8259

F-statistic: 267.5 on 6 and 331 DF, p-value: < 2.2e-16

Se ha eliminado la variable “displacement” porque tiene un p-valor menor al nivel de significación (más adelante se usará el criterio de AIC para justificar también). Comparado con el modelo anterior que incluía todas las variables, apreciamos que el R cuadrado ajustado a mejorado y tenemos la ventaja de ahora utilizar menos variables.

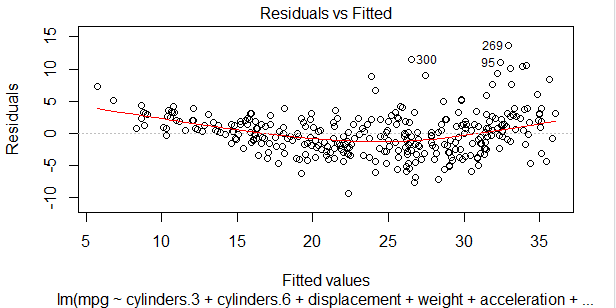
Nuestra ecuación de regresión final quedaría de la siguiente manera:

**Mpg = 27.1275 - 6.0636(cilinders.3) - 2.6846(cilinders.6) - 26.1086(weight) + 3.1932(acceleration) + 8.9159(model\_year)**

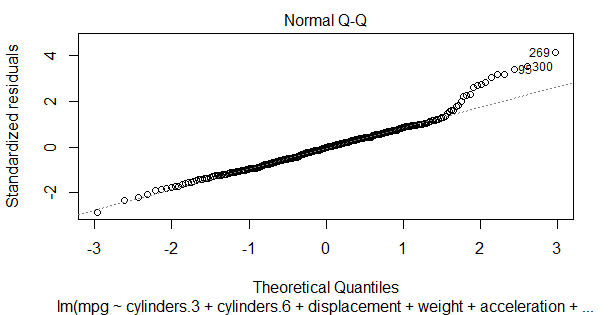
**Observación:** Vemos que el modelo final solo depende de 4 variables “cilindros” (3 y 6), “weight”, “acceleration” y “model\_year” (se ha excluido las variables “displacement” y “horsepower”). Además, el valor de R cuadrado ajustado a mejorado porque ahora es de 0.8259 (más que antes y es un valor excelente) manteniendo el mismo p\_valor menor al 5% de significación.

Ahora analizaremos la validación de los supuestos en la consola de la siguiente forma:

plot(modelo\_final)



Del gráfico, observamos que existe homocedasticidad de las varianzas ya que aproximadamente los valores por encima y por debajo de la recta roja se autocancelarìan (forma de embudo) para la mayoría de puntos.

****

Del gráfico, observamos que existe normalidad de los residuos para la gran mayoría de los datos, solo para cuando los cuantiles teóricos del modelo pasan de 2 allì deja de ser ajustado (pero es menoría respecto de toda la data).

Para analizar que no haya autocorrelación de los errores escribimos en consola lo siguiente:

> require(lmtest)

> dwtest(modelo\_final)

Durbin-Watson test

data: modelo\_final

DW = 1.9856, p-value = 0.45

alternative hypothesis: true autocorrelation is greater than 0

Como el DW está dentro del rango de [1.5 : 2.5] concluimos que no existe autocorrelación de los errores.

Para analizar que no exista multicolinealidad de variables escribimos en consola lo siguiente:

> car::vif(modelo\_final)

cylinders.3 cylinders.6 displacement weight acceleration model\_year

1.082566 1.057556 12.370799 9.907822 1.807297 1.222165

Apreciamos que el estadístico VIF es menor a 10 para todas las variables predictoras con excepción de “displacement”, pero esta variable fue retirada por el criterio AIC, por lo tanto, no existe multicolinealidad entre las variables independientes del modelo y con esto quedan validados todos los supuestos de regresión.

Para comparar modelos de machine learning, al no tener una ecuación matemática y no usar el R cuadrado ajustado, se utiliza la raíz cuadrada del error cuadrático medio.

**# Hallando la raiz cuadrada del error cuadrático medio para determinar**

**# el ajuste del modelo (mientras más cercano a cero es mejor)**

error\_del\_modelo = sqrt(mean(testing\_set$mpg - y\_pred)^2)

plot(testing\_set$mpg, y\_pred, main = "Valores Reales vs Valores Predichos",

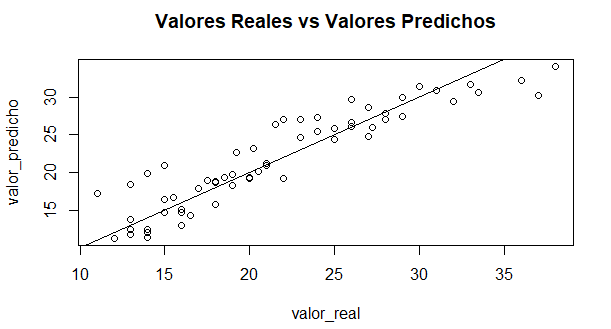
xlab = "valor\_real", ylab = "valor\_predicho")

abline(0,1) #pinta la recta que forma 45 grados con el eje x

En consola nos aparecerá el error del modelo de la siguiente manera:

> error\_del\_modelo

[1] 0.2795762 #Este error es bastante bajo para el tipo de variable que es mpg

****

Vemos que los puntos de la gráfica se ajustan alrededor de la recta con pendiente de 45 grados respecto del eje x, por lo tanto el modelo es muy bueno.

**Objetivo 2:**

Desarrollaremos un modelo de Regresión con Máquinas de Soporte Vectorial con el mismo dataset que usamos en la clase pasada para el modelo de Regresión Lineal Múltiple.

**# Máquina de Soporte Vectorial para Regresión**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8]

**# Generando variables dummy para las variables categóricas**

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Escalando las variables independientes entre 0 y 1 (0 valor mín y 1 valor máx)**

# Esta es otra forma de escalar utilizada para cuando ya tienes variables dummy

#install.packages("caret")

library(caret)

#install.packages("scales")

library(scales)

dataset$displacement = rescale(dataset$displacement)

dataset$horsepower = rescale(dataset$horsepower)

dataset$weight = rescale(dataset$weight)

dataset$acceleration = rescale(dataset$acceleration)

dataset$model\_year = rescale(dataset$model\_year)

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(123)

split = sample.split(dataset$mpg, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Ajustar SVR con el Conjunto de Datos de Entrenamiento**

#install.packages("e1071")

library(e1071)

regression = svm(formula = mpg ~ .,

data = training\_set,

type = "eps-regression",

kernel = "radial")

**# Predicción de nuevos resultados con SVR**

y\_pred = predict(regression, newdata = testing\_set)

> y\_pred

2 6 9 13 15 19 21 32 40

18.67863 23.43175 16.54372 16.22941 11.93825 13.76211 19.34353 15.05149 28.40324

45 51 58 61 64 65 68 72 81

26.34751 25.79777 26.12969 35.43765 13.67502 14.09526 12.57279 18.79365 28.96278

**# Hallando la raíz cuadrada del error cuadrático medio para determinar**

**# el ajuste del modelo (mientras más cercano a cero es mejor)**

error\_del\_modelo = sqrt(mean(testing\_set$mpg - y\_pred)^2)

plot(testing\_set$mpg, y\_pred, main = "Valores Reales vs Valores Predichos",

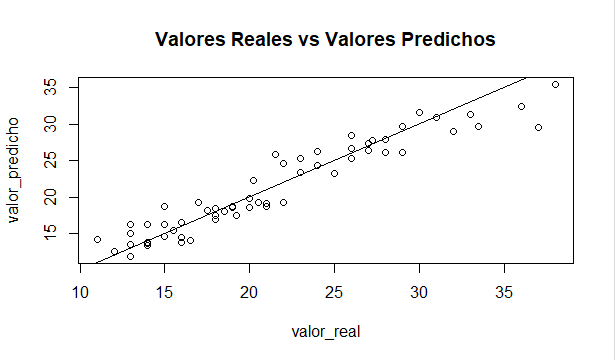
xlab = "valor\_real", ylab = "valor\_predicho")

abline(0,1) #pinta la recta que forma 45 grados con el eje x

> error\_del\_modelo

[1] 0.2128288

El error del modelo es aún más pequeño que el obtenido en el modelo de regresión lineal múltiple, con eso apreciamos que este método basado en distancias es más potente que el conocido método estadístico.

****

Del gráfico, apreciamos que los datos crecen alrededor de la recta con pendiente de 45 grados tanto para los datos reales como para los datos predichos, por lo tanto, el modelo está bien ajustado.

**Objetivo 3:**

Siguiendo con los fines comparativos entre modelos, haremos un modelo de regresión con árbol de decisión con el dataset auto.

**# Árbol de Decisión para Regresión**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8]

**# Generando variables dummy para las variables categóricas**

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(1234)

split = sample.split(dataset$mpg, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Ajustar Modelo de Árbol de Regresión con el Conjunto de Datos**

# install.packages("rpart")

library(rpart)

regression = rpart(formula = mpg ~ .,

data = training\_set)

**# Predicción de nuevos resultados con Árbol de Regresión**

y\_pred = predict(regression, newdata = testing\_set)

> y\_pred

5 10 11 29 35 53 75 77 90

19.38413 14.46184 14.46184 19.38413 23.18361 25.36667 19.38413 19.38413 23.18361

91 100 101 108 112 122 132 139 143

14.46184 14.46184 30.21739 14.46184 14.46184 14.46184 36.03600 14.46184 14.46184

**# Hallando la raíz cuadrada del error cuadrático medio para determinar**

**# el ajuste del modelo (mientras más cercano a cero es mejor)**

error\_del\_modelo = sqrt(mean(testing\_set$mpg - y\_pred)^2)

plot(testing\_set$mpg, y\_pred, main = "Valores Reales vs Valores Predichos",

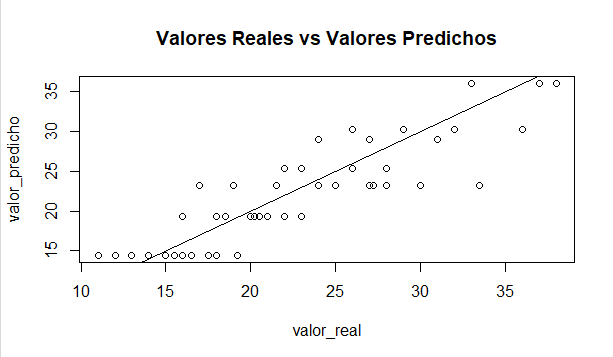
xlab = "valor\_real", ylab = "valor\_predicho")

abline(0,1) #pinta la recta que forma 45 grados con el eje x

> error\_del\_modelo

[1] 0.304907

El error del modelo es de 0.304907, este error es un poco más elevado que en el caso de regresión lineal, con eso apreciamos que este método basado en agrupaciones para predecir con las medias es menos potente para nuestro caso de estudio.



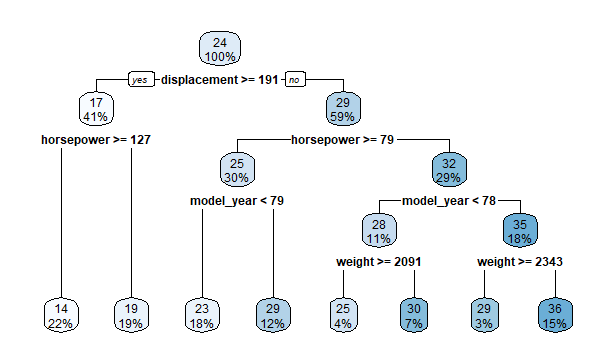
Del gráfico, apreciamos que los datos crecen alrededor de la recta con pendiente de 45 grados tanto para los datos reales como para los datos predichos, por lo tanto, el modelo está bien ajustado. Apreciamos como es el efecto de aplicar el promedio por grupos, ya que para varios valores reales en eje x, obtenemos un único valor en los valores predichos dando saltos en el eje y.

Si deseamos ver la construcción de nuestro árbol de regresión escribimos en consola lo siguiente:

|  |
| --- |
| > regression  n= 338  node), split, n, deviance, yval  \* denotes terminal node  1) root 338 21229.70000 23.85414  2) displacement>=190.5 139 1910.39300 16.69281  4) horsepower>=127 76 374.89930 14.46184 \*  5) horsepower< 127 63 700.90410 19.38413 \*  3) displacement< 190.5 199 7211.47000 28.85628  6) horsepower>=78.5 100 1952.71000 25.43600  12) model\_year< 78.5 61 581.44360 23.18361 \*  13) model\_year>=78.5 39 577.75440 28.95897 \*  7) horsepower< 78.5 99 2907.27800 32.31111  14) model\_year< 77.5 38 441.26970 28.30263  28) weight>=2091 15 89.73333 25.36667 \*  29) weight< 2091 23 137.91300 30.21739 \*  15) model\_year>=77.5 61 1475.06600 34.80820  30) weight>=2342.5 11 163.86180 29.22727 \*  31) weight< 2342.5 50 893.21520 36.03600 \* |

Apreciamos como se van generando las agrupaciones de datos de afuera hacia adentro con las sangrías empezando por la variable displacement, la cual a su vez para se dividen en horsepower y weight, así sucesivamente hasta llegar al siguiente gráfico final.

> rpart.plot(regression)



En estos gráficos, cada uno de los rectángulos representa un **nodo** de nuestro árbol, con su regla de clasificación (esto nos será más útil cuando usemos los árboles de decisión para clasificar).

Dentro del rectángulo de cada nodo se nos muestra qué proporción de casos pertenecen a cada categoría y la proporción del total de datos que han sido agrupados allí.

**Objetivo 4:**

Siguiendo con los fines comparativos entre modelos, haremos un modelo de regresión con bosques aleatorios con el dataset auto.

**# Random Forest Regression**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8]

**# Generando variables dummy para las variables categóricas**

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(1234)

split = sample.split(dataset$mpg, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Ajustar Modelo de Random Forest con el Conjunto de Datos de Entrenamiento**

# install.packages("randomForest")

library(randomForest)

set.seed(1234)

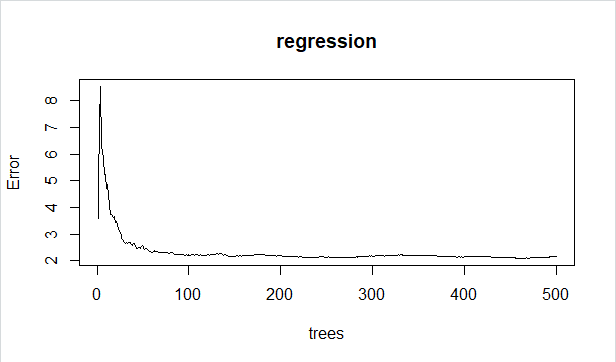
regression = randomForest(x = training\_set,

y = training\_set$mpg,

ntree = 500)

**# Analizando el número de árboles elegidos vs el error cuadrático medio**

> plot(regression)



Del gráfico, podemos ver como el error cuadrático medio va disminuyendo a medida que se va aumentando el número de árboles y queda aproximadamente estable con 100 árboles. Si el conjunto de datos hubiese sido más grande, con fines computacionales hubiese sido suficiente quedarnos con 100.

**# Predicción de nuevos resultados con Bosque Aleatorio**

y\_pred = predict(regression, newdata = testing\_set)

> y\_pred

5 10 11 29 35 53 75 77 90

20.98458 13.89345 12.14271 19.13128 25.16767 22.94417 17.81041 20.34862 24.27606

91 100 101 108 112 122 132 139 143

15.30757 11.99359 27.58647 14.82264 16.95875 13.99756 33.03121 14.07384 12.69439

**# Hallando la raíz cuadrada del error cuadrático medio para determinar**

**# el ajuste del modelo (mientras más cercano a cero es mejor)**

error\_del\_modelo = sqrt(mean(testing\_set$mpg - y\_pred)^2)

plot(testing\_set$mpg, y\_pred, main = "Valores Reales vs Valores Predichos",

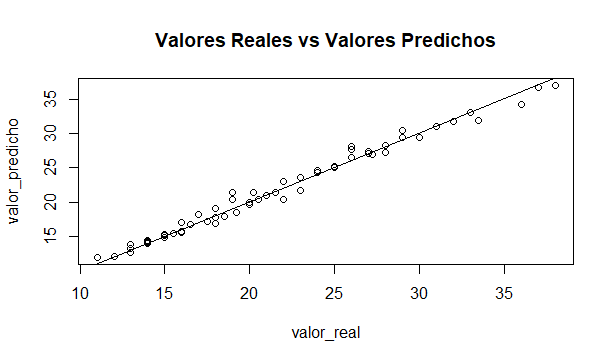
xlab = "valor\_real", ylab = "valor\_predicho")

abline(0,1) #pinta la recta que forma 45 grados con el eje x

> error\_del\_modelo

[1] 0.1339277

El error es de tan solo 0.1339277, hasta el momento el más bajo de todos los modelos realizados.



Del gráfico, vemos cómo cambia el valor predicho en relación al cambio del valor real, ambos se ajustan en crecimiento lineal respecto a la recta de 45 grados con el eje X. Estos puntos tan cercanos a la recta coinciden con el error tan bajo hallado anteriormente.

**Objetivo 5:**

Ahora, nos toca realizar una red neuronal con nuestro conjunto de datos de autos, las redes neuronales suelen ser modelos muy robustos y precisos, veamos a continuación la sintaxis:

**# Neural Network Regression**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8]

**# Generando variables dummy para las variables categóricas**

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(1234)

split = sample.split(dataset$mpg, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Ajustar Modelo de Red Neurnal con el Conjunto de Datos de Entrenamiento**

library(nnet)

library(caret)

library(devtools)

set.seed(1234)

regression = nnet(mpg ~., data=training\_set,

size = 4, decay = 0.1,

maxit = 1000, linout=T)

**# Predicción de nuevos resultados con la Red Neuronal**

y\_pred = predict(regression, newdata = testing\_set)

> y\_pred

[,1]

5 19.36894

10 15.11456

11 13.36591

29 18.91454

35 21.83104

53 22.89265

**# Hallando la raíz cuadrada del error cuadrático medio para determinar**

**# el ajuste del modelo (mientras más cercano a cero es mejor)**

error\_del\_modelo = sqrt(mean(testing\_set$mpg - y\_pred)^2)

plot(testing\_set$mpg, y\_pred, main = "Valores Reales vs Valores Predichos",

xlab = "valor\_real", ylab = "valor\_predicho")

abline(0,1) #pinta la recta que forma 45 grados con el eje x

> error\_del\_modelo

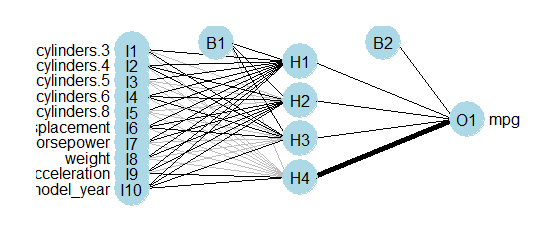
[1] 0.222797

Al analizar el error del modelo (0.222797), vemos que nuestra red neuronal tiene un error bajo pero intermedio entre los ya obtenidos siendo aún menor el del modelo de Bosque Aleatorio.

**# Visualización de la Red Neuronal con una librería externa al Cran**

source\_url("https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r")

plot(regression, max.sp = T)



Del gráfico, vemos como son representadas todas las variables independientes, en este caso se usan absolutamente todas para la predicción. Los valores H (6 en total) muestran la capa que tiene los 6 nodos como máximo, finalmente, B1 y B2 son representaciones de nodos internos. Nuestra salida 01 tiene una línea muy gruesa que conecta con ella proveniente de H1, lo cual indica que dicho nodo es muy importante en cuanto a la generación de pesos en la red neuronal.

**Objetivo 6:**

Finalmente, haremos nuestro modelo de K- Nereast Neighbors con nuestro mismo dataset de autos, en este caso es necesario generar un conjunto intermedio para validar además de los conjuntos de entrenamiento y de testing, esto es con fines de generar una mayor precisión en el modelo.

**# K-Nearest Neighbors para Regression**

**# Importar el dataset**

dataset = read.csv('auto-mpg.csv')

dataset = dataset[, 2:8]

**# Generando variables dummy para las variables categorícas**

# install.packages("dummies")

library(dummies)

dataset = dummy.data.frame(dataset, names = c("cylinders"), sep = ".")

**# Escalando las variables independientes entre 0 y 1 (0 valor mín y 1 valor máx)**

# Esta es otra forma de escalar utilizada para cuando ya tienes variables dummy

#install.packages("caret")

library(caret)

#install.packages("scales")

library(scales)

dataset$displacement = rescale(dataset$displacement)

dataset$horsepower = rescale(dataset$horsepower)

dataset$weight = rescale(dataset$weight)

dataset$acceleration = rescale(dataset$acceleration)

dataset$model\_year = rescale(dataset$model\_year)

**# Dividir los datos en conjunto de entrenamiento , validación y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(1234)

set.seed(2018)

t.id <- createDataPartition(dataset$mpg, p=0.6, list = F)

tr <- dataset[t.id, ]

temp <- dataset[-t.id, ]

v.id <- createDataPartition(temp$mpg, p=0.5, list = F)

val <- temp[v.id,]

test <- temp[-v.id,]

**# Ajustar Modelo de KNN con el Conjunto de Datos de Entrenamiento y validación**

**# para k=3, k=5 y K=7**

# install.packages("randomForest")

library(FNN)

regression1 = knn.reg(tr[,-1], val[,-1], tr$mpg, k=1, algorithm = "brute")

regression2 = knn.reg(tr[,-1], val[,-1], tr$mpg, k=3, algorithm = "brute")

regression3 = knn.reg(tr[,-1], val[,-1], tr$mpg, k=5, algorithm = "brute")

regression4 = knn.reg(tr[,-1], val[,-1], tr$mpg, k=7, algorithm = "brute")

**# Hallando la raiz cuadrada del error cuadràtico medio para k=3, k=5 y k=7**

rmse1 = sqrt(mean((regression1$pred-val$mpg)^2))

rmse2 = sqrt(mean((regression2$pred-val$mpg)^2))

rmse3 = sqrt(mean((regression3$pred-val$mpg)^2))

rmse4 = sqrt(mean((regression4$pred-val$mpg)^2))

**# Generando un vector de los rmse encontrados**

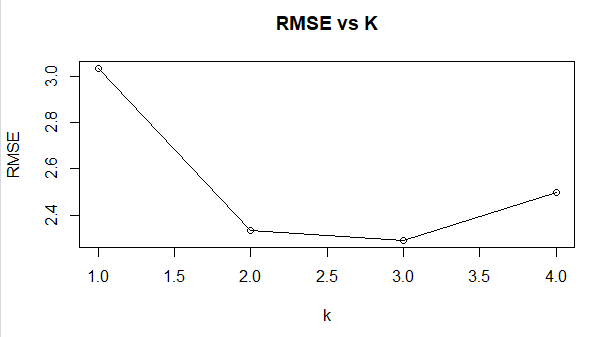
errors = c(rmse1, rmse2, rmse3, rmse4)

> errors

[1] 3.033755 2.332477 2.291931 2.498000

**# Visualizando los errores en relación al número k de items a agrupar**

plot(errors, type = 'o', xlab = "k", ylab = "RMSE", main ="RMSE vs K")



Del gráfico, apreciamos que para k = 5 (3er punto) el RMSE ha convergido a un menor valor, por lo tanto, nos quedaremos con k = 5, para el cual tenemos un RMSE = 2.291931, este es un valor muy alto, lo cual nos indica que no es el mejor modelo para nuestro caso.

**Conclusión Final:**

Recordemos que valores obtuvimos de los RMSE con nuestros modelos de regresión.

> error\_del\_modelo de Regresión múltiple

[1] 0.2795762

> error\_del\_modelo de Máquinas de Soporte Vectorial

[1] 0.2128288

> error\_del\_modelo de Árbol de Decisión

[1] 0.304907

> error\_del\_modelo de Bosques Aleatorios

[1] 0.1339277

> error\_del\_modelo de Red Neuronal Artificial

[1] 0.222797

> error\_del\_modelo de Red Neuronal Artificial

[1] 2.291931

**Por lo tanto, concluimos que el mejor modelo para nuestro caso fue el de Bosques Aleatorios, puesto que tienen el RMSE más pequeño de todos.**