Máquinas de vector de soporte y redes de neuronas

Grupo de laboratorio:

- Carlos García Peral
- Carlos Fernández de la Torre
- Israel Laguna García

ÍNDICE

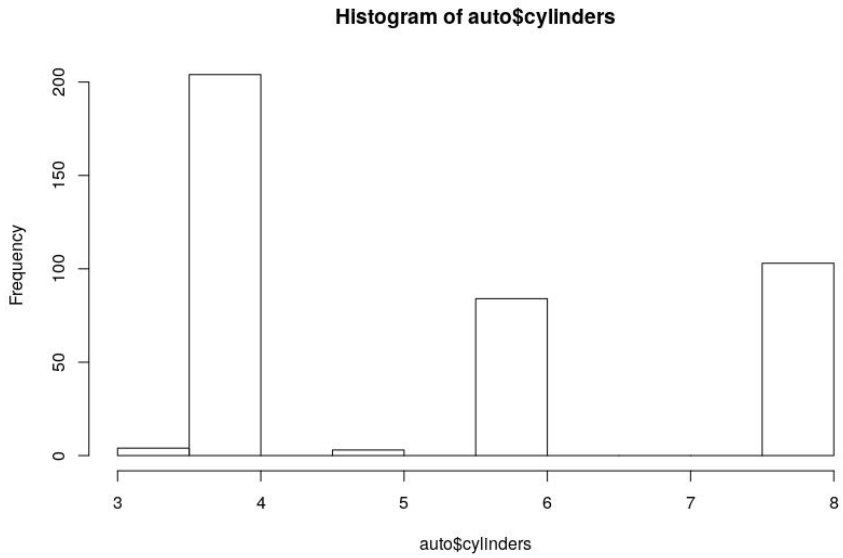
Introducción

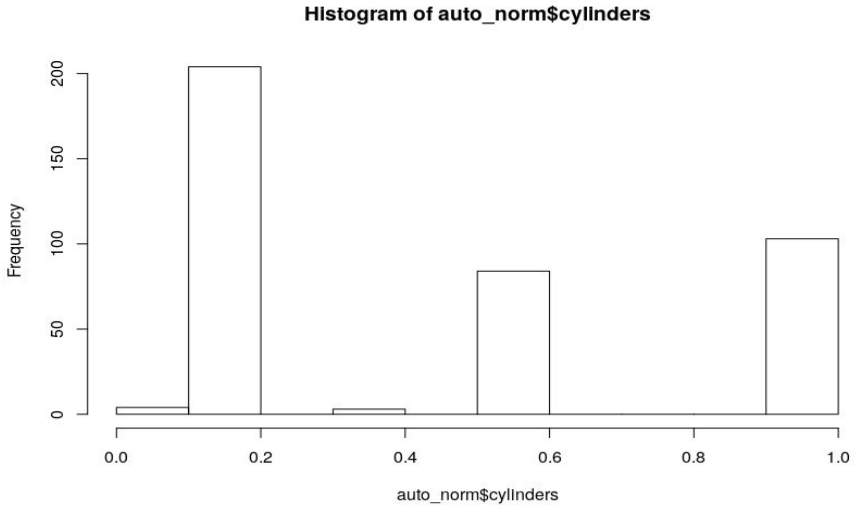
En el presente informe se muestran los resultados obtenidos al emplear dos modelos de aprendizaje automático basados en las técnicas de máquinas de vectores de soporte y de redes de neuronas, utilizando diferentes metodologías, con el fin de abordar el problema objeto de estudio del Laboratorio.

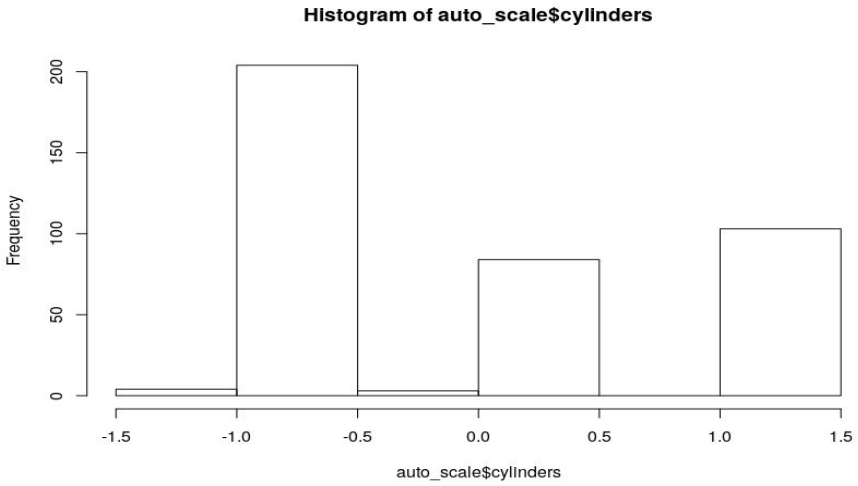
El objetivo es explicar el proceso seguido, desglosando cada fase, haciendo énfasis en las partes esenciales en el desarrollo de un modelo predictivo, y comparando y analizando las distintas métricas que ofrece cada técnica utilizada, para finalizar exponiendo las conclusiones y posibles mejoras aplicables.

Preparación de los datos

- En primer lugar leemos los datos del fichero de entrada (csv) y realizamos la normalización y tratamiento de valores "*missing*" detectados en la variable "*horsepower*":
    a. Primero asignamos la media de la variable aquellos valores "*missing*" detectados, forzando además que esos valores no se tengan en cuenta en el cálculo de dicha media
    b. Excluimos del set de datos la variable "*car_name*", ya que es de tipo string y no requiere normalización, y la variable de respuesta "*mpg*".
    c. Normalizamos utilizando máximos y mínimos
    d. Escalamos los datos
- Realizando la comparación entre los datos normalizados y escalados, vemos que las variables normalizadas se asemejan más al comportamiento de las variables originales, que las variables escaladas. A continuación mostramos como ejemplo la gráfica para la variable "cylinders" de los datos de entrada (para el resto puede verse ejecutando el código).


Histogram of auto$cylinders
Frequency
0
50
100
150
200
3
4
5
6
7
8
auto$cylinders


Histogram of auto_norm$cylinders
Frequency
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0
auto_norm$cylinders



- Con los datos ya normalizados añadimos al dataset la variable de respuesta "*mpg*", instanciamos una semilla para la generación aleatoria de los números y creamos los conjuntos de datos de entrenamiento y test, asignando un 80% y 20% de los datos respectivamente.

Entrenamiento y predicción con SVM

- Entrenamos el modelo con svm, utilizando un kernel lineal, indicando que no se escalen los datos, y aplicando un valor de 1 al parámetro de coste C.
  Los resultados de este primer entrenamiento son:

  SVM-Type: **nu-regression**

  SVM-Kernel: **linear**

  Cost: **1**

  Gamma: **0.1428571**

  nu: **0.5**

  Number of Support Vectors: **161**
- Para poder realizar una comparativa más exhaustiva y ajustar el modelo, utilizamos la función "*tune*" del paquete e1071, que nos permite probar diferentes valores del parámetro de coste C, y aplicamos cross-validation con un valor de k = 5.

Los resultados obtenidos se muestran a continuación:

Parameter tuning of 'svm':

- Sampling method: 5-fold cross validation

- Best parameters:

  - Cost: 100

- Best performance: 11.85734

Coste	**Error**	**Dispersión**
0.001	22.54886	4.727181
0.010	13.46166	3.706722
0.100	12.15930	3.465686
1.000	11.86091	3.169872
10.000	11.85817	3.120138
100.000	11.85734	3.117280
150.000	11.85880	3.117075

- Obtenemos el mejor modelo y mostramos los resultados:

  SVM-Type: **eps-regression**

  SVM-Kernel: **linear**

  Cost: **100**

  Gamma: **0.1428571**

  Epsilon: **0.1**

  Number of Support Vectors: **258**

- Ahora entrenamos el modelo con svm, utilizando un kernel de base radial, con diferentes valores del parámetro de coste C, y aplicamos cross-validation con un valor de k = 5.

Los resultados de este primer entrenamiento son:

Parameter tuning of 'svm':

- Sampling method: 5-fold cross validation

- Best parameters:
  - Cost: 10
- Best performance: 6.913938

Coste	**Error**	**Dispersión**
0.001	57.907329	8.852838
0.010	40.742771	7.764466
0.100	11.695132	2.527644
1.000	7.316121	1.837912
10.000	6.913938	2.688906
100.000	9.300175	4.090927
150.000	10.355352	4.742473

- Obtenemos el mejor modelo y mostramos los resultados:

  SVM-Type: **nu-regression**

  SVM-Kernel: **radial**

  Cost: **10**

  Gamma: **0.1428571**

  nu: **0.5**

  Number of Support Vectors: **216**

- Los resultados muestran que el mejor modelo es el que usa kernel lineal, así pues lo aplicamos en la predicción.

Aplicamos las métricas RMSE y MSE, para obtener el error cuadrático medio, y la raíz cuadrada del error cuadrático medio respectivamente:

RMSE: **3.561055**

MSE: **12.68111**

Calculamos también la correlación lineal entre la variable de respuesta "*mpg*" predicha y real:

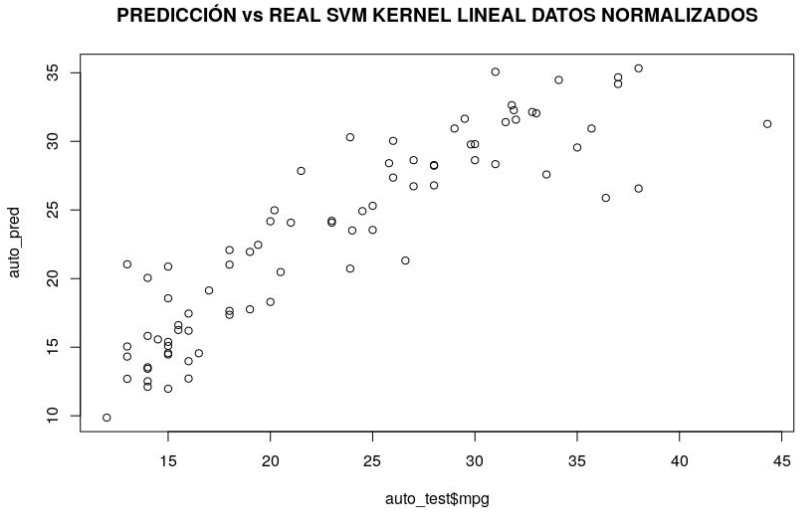
Correlación: **0.8958071**

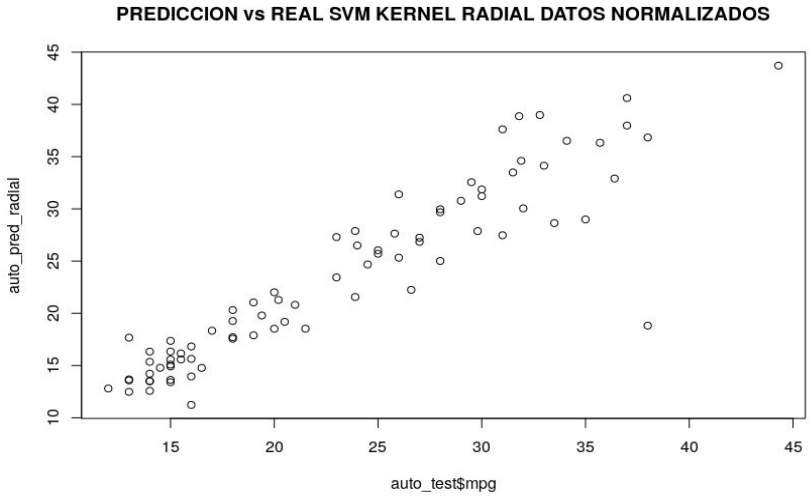
- A continuación obtenemos las métricas para el modelo que utiliza kernel de base radial, para realizar la comparativa:

  RMSE: **3.294431**

  Correlación: **0.9197275**

- Por último mostramos los gráficos de dispersión de ambos modelos (kernel lineal y radial)

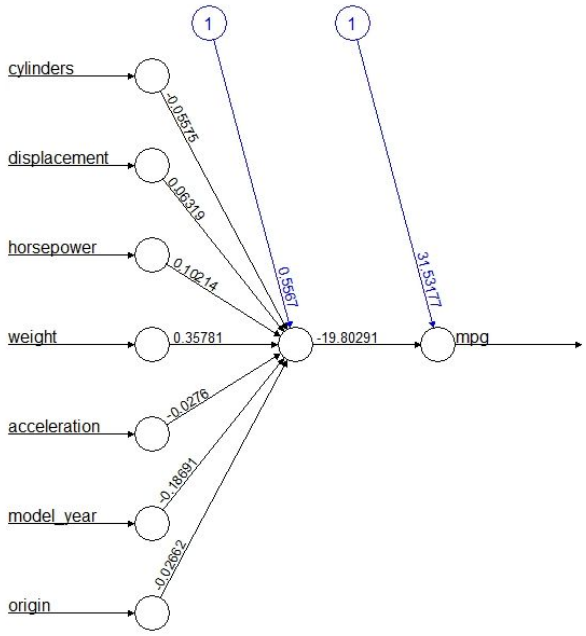




- Como conclusión podemos observar que el modelo de svm basado en kernel radial muestra un mejor gráfico de dispersión, a diferencia de lo que parecía inicialmente

Entrenamiento y predicción con redes de neuronas (NN)

- Para crear el modelo con redes neuronales utilizamos los datos escalados, ya que las funciones de activación tienen márgenes muy estrechos
- Con los datos escalados, añadimos al dataset la variable de respuesta "*mpg*", instanciamos una semilla para la generación aleatoria de los números y creamos los conjuntos de datos de entrenamiento y test, asignando un 80% y 20% de los datos respectivamente.
- Seleccionamos como función de activación la función "*tangente hiperbólica*", ya que sur márgenes de activación se ajustan mejor a los de las variables escaladas.
  La siguiente gráfica muestra el resultado generado al entrenar el modelo:



Error: 1205.172705 Steps: 8762

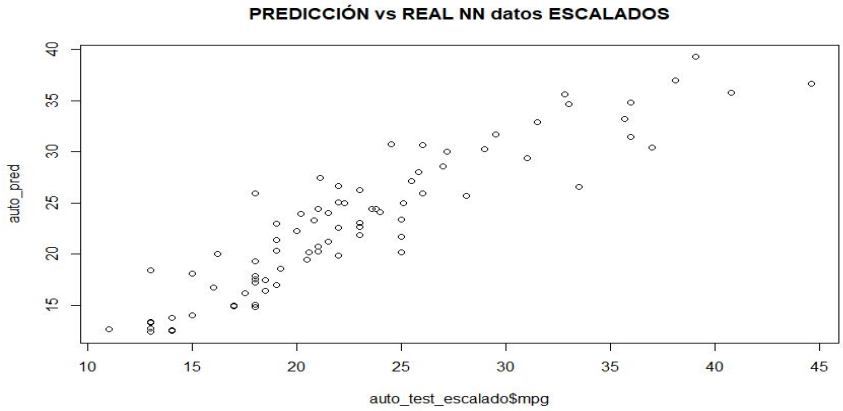
- A continuación realizamos la predicción con el modelo entrenado, y evaluamos los resultados:

  Correlación: **0.914992**

  RMSE: **2.924584**

  MSE: **8.553194**

  El siguiente gráfico nos muestra que las predicciones no se alejan demasiado de la diagonal, pero están por lo general un poco sobreestimadas.



Conclusiones

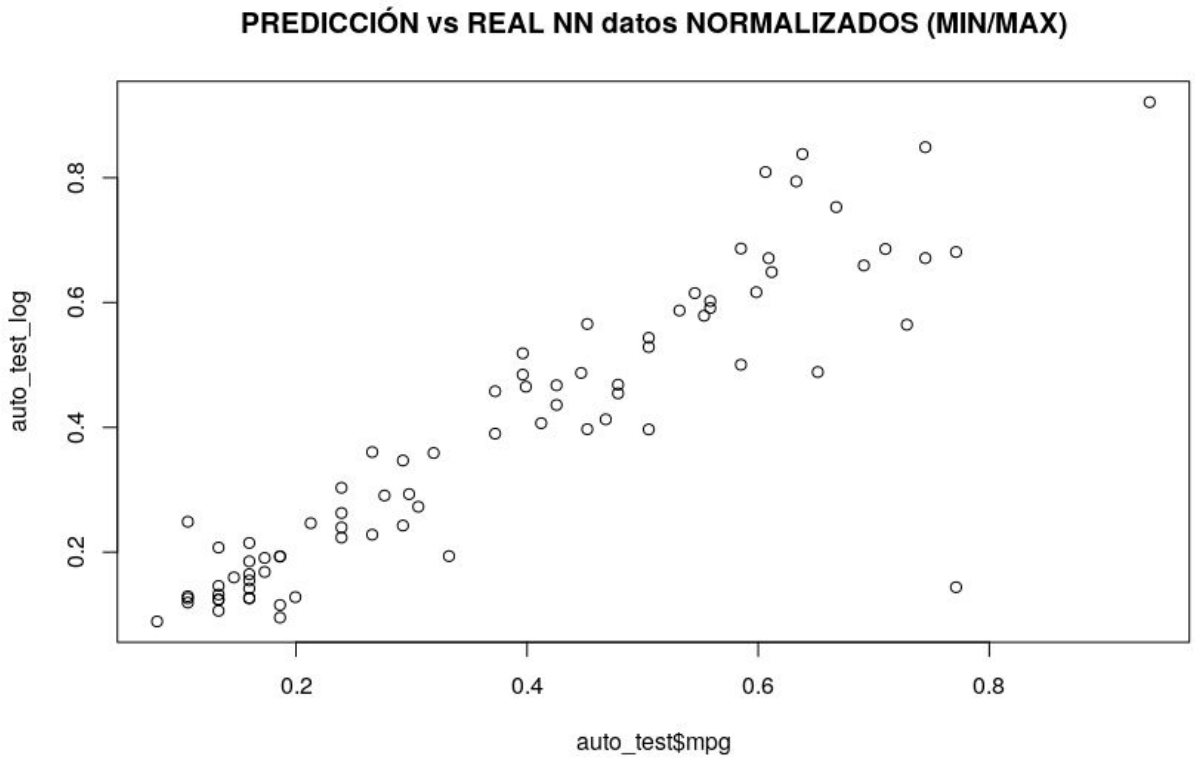
Después de realizar diversas pruebas con los siguientes modelos y aproximaciones:

- Redes Neuronales logística y tangencial con diferentes capas
- Redes Neuronales con una función de activación elegida por nosotros, como por ejemplo la función Sigmoide.
- Redes Neuronales con algoritmo backpropagation

Comprobamos que el mejor resultado obtenido revisando las métricas de todos los modelos, se corresponde con :

- La utilización de una función de activación logística
- Uso de datos normalizados mediante mínimos y máximos, incluyendo la variable independiente
- Uso de una red neuronal con 3 capas ocultas

La siguiente gráfica muestra el resultado generado al entrenar el modelo:



A continuación realizamos la predicción con el modelo entrenado, y evaluamos los resultados:

Correlación: **0.9603399**

RMSE: **0.05588081**

MSE: **0.003122665**