Aprendizaje Automático

Laboratorio 1
Redes neuronales y máquinas de soporte vectorial

unir
LA UNIVERSIDAD
EN INTERNET

Alberto González Isorna
Federico Damián Estébanez
José Felipe López Guerra

Introducción

Se nos pide predecir el consumo (millas por galón, *mpg*) de unos vehículos en función de los datos proporcionados en el dataset *'auto-mpg'*. Dicha regresión se hará utilizando máquinas de vector de soporte y redes neuronales.

Análisis y tratamiento previo de los datos

Según comprobamos en la descripción del dataset, las columnas proporcionadas corresponden a valores de (por fila): mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, car name.

Nuestra variable dependiente serán las mpg. Dado que las mpg son una medida de la distancia recorrida por galón de combustible, tenemos que cuanto mayor sea el valor de mpg, menor es el consumo del vehículo (recorre más distancia con la misma cantidad de combustible).

Por otro lado tenemos como variables independientes: cylinders, horsepower, displacement, weight, acceleration, model year, origin y car name.

Con este planteamiento, realizamos un análisis de la información contenida en el dataset para: encontrar posibles *valores missing* y tratarlos adecuadamente, analizar la distribución de los datos para detectar correlaciones entre las diferentes variables y, seleccionar las variables que utilizaremos para nuestras posteriores predicciones.

Cargamos los datos en un dataframe con la librería *pandas,* especificando los nombres de columnas de cada dato.

```
#Read CSV
car_data = pd.read_csv(
    'auto-mpg.data',
    header=None,
    encoding='utf-8',
    delim_whitespace=True,
    names=['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year', 'origin', 'car name'],

)
```

Una vez los tenemos cargados, hacemos una primera comprobación de posibles valores nulos y vemos que aparentemente no encontramos ninguno.

```
#Comprobación de datos nulos
car_data.isnull().any() #Aparentemente ningún dato nulo
```

```
mpg             False
cylinders       False
displacement    False
horsepower      False
weight          False
acceleration    False
model year      False
origin          False
car name        False
dtype: bool
```

Continuamos con el tratamiento y hacemos una comprobación de los tipos de datos de los que disponemos. A priori, todos los valores deben ser numéricos excepto la columna con el nombre del vehículo.

```
#Chequeo del tipo de los datos
car_data.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight          float64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

La columna de horsepower debe ser un tipo numérico en lugar de un objeto. Analizamos los datos para ver qué ocurre. En un primer vistazo, listamos los valores únicos que tenemos en la columna, y vemos que hay valores tipo string con el símbolo '?'. Comprobamos que se trata de 6 filas con ese valor y decidimos eliminarlas de nuestro dataframe. Realizamos una conversión a tipo numérico de los valores restantes en la columna horsepower. También transformamos a tipo numérico la columna que almacena el nombre del vehículo.

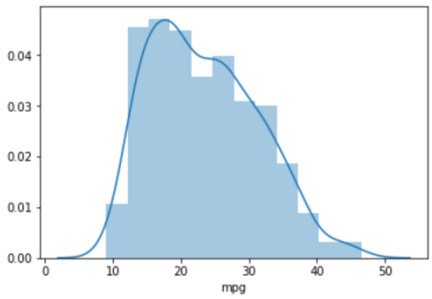
```
car_data.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight          float64
acceleration    float64
model year        int64
origin            int64
car name          int64
dtype: object
```

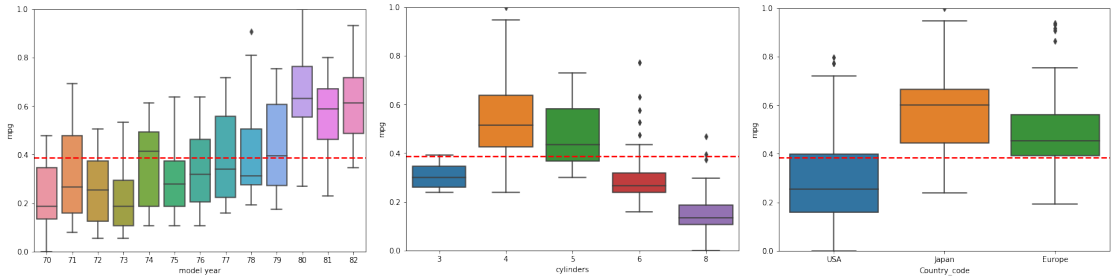
Si analizamos la distribución global de mpg, tenemos que el valor mínimo es de 9 y el máximo de 46,6, con una media de 23,44. Con coeficiente de asimetría 0,457 y curtosis -0,515.

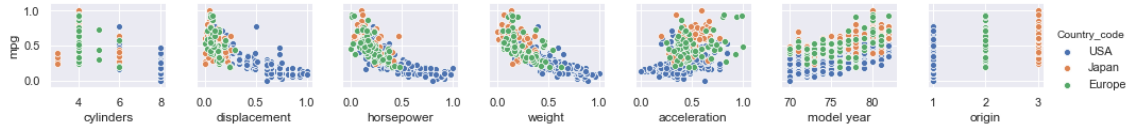
```
# Análisis de la distribución de mpg
sns.distplot(car_data['mpg'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x120c9b4e0>
```

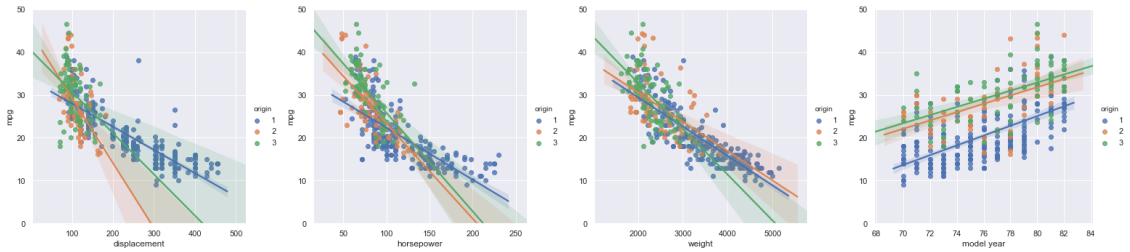


Una vez realizamos una copia de nuestro dataframe pero con los datos escalados para facilitar su representación, generamos diferentes gráficas para poder analizar la distribución de mpg con las diferentes variables.





Según los datos observados, tenemos que casi el 75% de los coches USA tienen un mpg por debajo de la media y, la mayoría de los coches de origen europeo o asiático un mpg por encima de la media. También vemos que displacement, horsepower y weight están relacionadas inversamente con mpg y el año directamente.



Preparación de datos de training y test

Tras hacer varias pruebas y analizar las gráficas, hemos elegido los factores que influyen realmente al consumo para llevar a cabo nuestra regresión, estos son:

- Desplazamiento (kilómetros recorridos): relación inversa.
- Potencia: relación inversa.
- Peso: relación inversa.
- País de origen: menos consumo en Japón, seguido de Europa y por último USA.
- Año del modelo: relación directa

Relación Inversa: a mayor valor de la variable X menos millas por galón de gasolina podemos hacer con el coche.

Relación Directa: a menor valor de la variable X menos millas por galón de gasolina podemos hacer con el coche.

Hemos decidido también utilizar un 80% de los datos para training y un 20% para test.

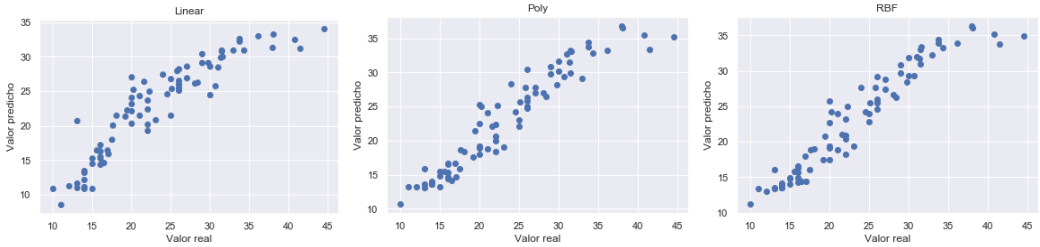
```
# Preparación de los datos de training

factors = ['displacement', 'horsepower', 'weight', 'origin', 'model year']
X = pd.DataFrame(car_data[factors].copy())
y = car_data['mpg'].copy()

X = StandardScaler().fit_transform(X) #escalamos los datos de 0 a 1

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

```

Predicción: Máquina de vector de soporte

Una vez hemos separado los datos de entrenamiento y test, procedemos a hacer la predicciones. Hemos procedido con una estrategia de regresión ya que nuestro objetivo es predecir que valor que tendrá un coche con unas determinadas caracteristicas no introducirlo en una categoría.

Dentro del Suport Vector Regresion hemos elegido tres estrategias: Nucleo Radial (Rbf), Nucleo Lineal (Linear) y Predicción polinomial (Poly).

Comprobando los valores reales con los predichos, vemos como en general obtenemos buenos resultados con los tres algoritmos.



Además, comprobamos la precisión de nuestros modelos y los valores MSE y RMSE de nuestros resultados, obteniendo para cada algoritmo:

```
RBF: 0.9124853741870398
RBF r2 score: 0.9124853741870398
RBF MSE: 5.4394991216810595
RBF RMSE: 2.3322733805626346
Linear: 0.8332201820138039
Linear r2 score: 0.8332201820138039
Linear MSE: 10.36625209812291
Linear RMSE: 3.219666457588877
Poly: 0.9048044580165465
Poly r2 score: 0.9048044580165466
Poly MSE: 5.9169088846205415
Poly RMSE: 2.432469708880368
```

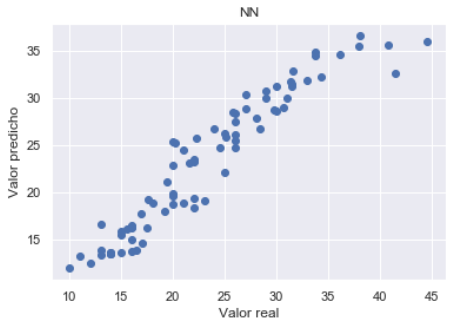
Vemos como en general obtenemos buenos resultados con los tres algoritmos. Dado que los datos no son totalmente lineales si no que algunos tienen una ligera tendencia exponencial el ajuste polinómico o el radial nos aportan los mejores resultados con un RMSE de entorno al 2.5 y precisiones del 90%.

Predicción: Red neuronal

El procedimiento de la red neuronal respecto al la SVM es totalmente distinto. La filosofía en este caso es primero crear la estructura de la red neuronal y posteriormente gracias a la propagación inversa ajustar los pesos que obtienen los mejores resultados.

Hemos probado distintas estructuras y posibilidades, y hemos optado por utilizar el MLPRegressor.

Comprobando los valores reales con los predichos, vemos también que obtenemos buenos resultados.



También chequeamos la precisión de nuestro modelo y los valores MSE y RMSE de nuestros resultados, obteniendo para cada algoritmo:

```
MLPRegressor: 0.9071646795188888
MLPRegressor r2 score: 0.9071646795188887
MLPRegressor MSE: 5.770208574018737
MLPRegressor RMSE: 2.40212584475059
```

Vemos como obtenemos resultados muy similares a los mejores datos obtenidos con el SVM, precisión del 90% y RSME de 2.4. Como ventaja la red neuronal requiere menos ajuste que los métodos utilizados previamente, ya que iteración tras iteración, aprende y mejora sus propios resultados.