

# Práctica 3

Blanca Cano Camarero

Curso 2020-2021

## Índice

<b>Problema de clasificación</b>	<b>2</b>
Análisis del problema . . . . .	2
Descripción del tratamiento de los datos . . . . .	2
Lectura y tratamiento inicial de los datos . . . . .	2
Normalización . . . . .	4
<b>Fuentes</b>	<b>6</b>

## Problema de clasificación

### Análisis del problema

Estamos ante un problema de clasificación.

De la página web de la que se han obtenido los datos [Dataset for Sensorless Drive Diagnosis Data Set](#)

Abstract: Las características son extraídas de la corriente de un motos. El motor puede tener componentes intactos o defectuosos. Estos resultados se encuentran en 11 clases con diferentes condiciones.

Tenemos además la siguiente información:

- Las características del data set son multivariantes.
- Los datos son tipo números reales.
- Es una tarea de clasificación.
- El número de instancias total es 58509.
- El número de atributos es de 49.
- Si faltan datos: N/A . TO-DO (¿qué hacer si faltan datos?)

### Descripción del tratamiento de los datos

#### Lectura y tratamiento inicial de los datos

El fichero tiene extensión `.txt` de texto plano, para leerlo usaremos la función escrita `LeerDatos (nombre_fichero, separador)` que tiene como pilar básico la función `read_csv` de la biblioteca de `pandas`.

Nota: suponemos que la estructura de carpetas es:

```
.
|- clasificacion.py
|- datos
    |-Sensorless_drive_diagnosis.txt
```

Donde `clasificacion.py` es el nombre del ejecutable de nuestra práctica, `datos` es una carpeta y `Sensorless_drive_diagnosis.txt` es el fichero que contiene los datos.

## Selección de test y entrenamiento

Comprobaremos antes si los datos están balanceados, para ello contaré el número de distintas etiquetas.

Esto lo haremos viendo el número de veces que se repite cada etiqueta, el resultado es:

Etiqueta	Número apariciones
1.0	5319
2.0	5319
3.0	5319
4.0	5319
5.0	5319
6.0	5319
7.0	5319
8.0	5319
9.0	5319
10.0	5319
11.0	5319

Como podemos ver está perfectamente balanceado.

Debemos determinar ahora qué datos usaremos para test y cuáles para entrenamiento.

El porcentaje que voy a usar será un 20 de los datos reservados para test. La elección de esta se debe a heurísticas generales usadas y porque tenemos los suficientes datos para el entrenamiento.

En cuanto a las opciones de cómo separarlos estos deben ser seleccionados de manera aleatoria con una distribución uniforme. Desconozco si además el tamaño es suficientemente grande como para separarlos directamente sin tener que ir clase por clase tomando el mismo número, ya que al ser homogénea, si el tamaño es suficiente puedo suponer que la selección por clases será homogénea.

Para separarlos usaré la función `sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)` [source] de la biblioteca de sciklearn, concretamente con los siguientes parámetros:

```
ratio_test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(
    x, y,
    test_size= ratio_test_size,
    shuffle = True,
    random_state=1)
```

- `test_size` se corresponde a la proporción de los datos que usaremos para el test, está a 0.2 porque ya hemos comentado que trabajaremos con el 20.
- `shuffle` a `True` porque queremos coger los datos al azar.
- `random_state` es una semilla para la mezcla.

Los resultados han sido:

Etiqueta	Número apariciones
1.0	1138
2.0	1097
3.0	1056
4.0	1065
5.0	1055
6.0	1029
7.0	1072
8.0	1044
9.0	1044
10.0	1043
11.0	1059

Vemos que la mayor diferencia es de  $|1138 - 1029| = 109$  si recordamos que cada clase contaba con 5319 esto supone una diferencia de  $\frac{109}{5319}100 = 2.0493$  es decir que en el peor de los casos estamos entrenando con dos datos más por cada cien.

Esto no me parece del todo significativo, así que continuaré sin hacerlos por clases. (TODO Hay que justificar esto, ya sea por un paper o por ).

Nótese que desde ahora solo trabajaremos con los datos de entrenamiento, para no cometer ningún tipo de data snooping.

## Normalización

Diferencias muy grandes entre los datos podría perjudicar al modelo, luego comprobaremos antes si es necesario si es necesario normalizar los datos.

Para ello he diseñado la función `ExploracionInicial()` que muestra la media y la varianza de los datos.

```
-----
Resumen de las tablas
-----
```

Media

Valor mínimo de las medias -1.5019152989937367

Valor máximo de las medias 8.416765275493

Varianza

Valor mínimo de las varianzas 3.419960283480337e-09

Valor máximo de las varianzas 752.5259323408474

-----

La variabilidad entre las medias y datos es considerable, así que vamos a normalizar.

Para ello usaremos la función `class sklearn.preprocessing.StandardScaler(*, copy=True, with_mean=True, with_std=True)` (“StandardScaler Del Paquete sklearnPreprocessing” [n.d.](#)) Según la documentación oficial a fecha de hoy, esta función normaliza las características eliminando la media y escalando en función de la varianza, es calculado de la siguiente manera:

$$Z = \frac{X - U}{s}$$

Donde  $u$  es la media de los dtos de entrenamiento o cero si el parámetro `with_mean=False` y  $s$  es la desviación típica de los datos del ejemplo y 1 en caso de que `with_std=False`.

No es más que una normalización del estimado (Como se hace con una distribución de normal de varianza y media... TO-DO completar).

## Correlación de los datos

Veamos ahora si podemos encontrar alguna relación entre las características, para ello vamos a utilizar la matriz de correlación.

( TO-DO Añadir información sobre la correlación )

Para calcularla utilizaremos `corrcoef` de la biblioteca de numpy (“CorrcoefNumpy Del Paquete Numpy” [n.d.](#)) que devuelve el producto de los momentos de los coeficientes

## Fuentes

“CorrcoefNumpy Del Paquete Numpy.” n.d. Accessed May 21, 2021. <https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html>.

“StandardScaler Del Paquete sklearnPreprocessing.” n.d. Accessed May 21, 2021. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>.