Capítulo 3
Utilizar las funciones PHP

1. Preámbulo

El objetivo de este capítulo es presentar las funciones más útiles para el desarrollo de un sitio web.

PHP ofrece numerosas funciones; la descripción de cada función está disponible en línea en el sitio www.php.net.

Versión 8

Desde la **versión 8**, es posible pasar parámetros a una función utilizando el nombre del parámetro en lugar de su posición. Esta funcionalidad se presenta en el capítulo Escribir funciones y clases PHP, pero puede utilizarse para las funciones propias del lenguaje PHP y, por tanto, para las funciones que se presentan en este capítulo. Sin embargo, en este capítulo, los nombres reales de los parámetros de las funciones no se presentan (están traducidos); para conocerlos, consulte la documentación en línea de las funciones…

Desde la **version 8.1**, pasar el valor `NULL` en un parámetro que no es explícitamente opcional es obsoleto, y por lo tanto, genera una alerta `E_DEPRECATED`.

Ejemplo

```
<?php
$x = null;
$n = strlen($x);
?>
```

Resultado

```
Deprecated: strlen(): Passing null to parameter #1 ($string) of type string
is deprecated in /app/scripts/index.php on line 3
```

2. Manipular las constantes, las variables y los tipos de datos

2.1 Constantes

PHP ofrece una serie de funciones útiles sobre las constantes:

Nombre	Función
`defined`	Indica si una constante está definida o no.
`constant`	Devuelve el valor de una constante.

defined

La función `defined` permite saber si una constante está definida o no.

Sintaxis

booleano `defined(`*cadena* `nombre)`

`nombre` Nombre de la constante.

La función `defined` devuelve `TRUE` si la constante está definida y `FALSE` en caso contrario.

Ejemplo

```
<?php
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE está definida.<br />';
} else {
  echo 'CONSTANTE no está definida.<br />';
};
// Definir la constante CONSTANTE
define('CONSTANTE','valor de la CONSTANTE');
// Probar si la constante CONSTANTE está definida.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE está definida.<br />';
} else {
```

```
  echo 'CONSTANTE no está definida.<br />';
};
?>
```

Resultado

```
CONSTANTE no está definida.
CONSTANTE está definida.
```

constant

La función `constant` devuelve el valor de una constante cuyo nombre se pasa como parámetro.

Sintaxis

mixto constant(*cadena* nombre)

Donde

`nombre` Nombre de la constante.

Esta función es útil para recuperar el valor de una constante cuyo nombre no se conoce a priori.

Ejemplo

```
<?php
// definir el nombre de la constante en una variable
$nombreConstante = 'OTRA CONSTANTE';
// definir el valor de la constante
define($nombreConstante,'valor de la OTRA CONSTANTE');
// mostrar el valor de la constante
echo $nombreConstante,' = ',constant($nombreConstante);
?>
```

Resultado

```
OTRA CONSTANTE = valor de la OTRA CONSTANTE
```

Otras funciones permiten conocer el tipo de una constante (véase la sección Manipular las constantes, las variables y los tipos de datos - Tipos de datos).

2.2 Variables

PHP ofrece una serie de funciones útiles en las variables:

Nombre	Función
`empty`	Indica si una variable está vacía o no.
`isset`	Indica si una o varias variables están definidas o no.

Nombre	Función
unset	Elimina una o varias variables.
var_dump	Muestra la información sobre una o varias variables (tipo y valor).

empty

La función `empty` permite probar si una variable está vacía o no.

Sintaxis

```
booleano empty(mixto variable)
```

`variable` Variable que se va a probar.

`empty` devuelve TRUE si la variable está definida y FALSE en caso contrario.

Una variable se considera vacía si no ha sido asignada o si contiene una cadena vacía (""), una cadena igual a 0 ("0"), un 0, NULL, FALSE o una tabla vacía.

La función `empty` también se puede utilizar para probar si una expresión está vacía o no.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_vacía = empty($variable);
echo '$variable no inicializada<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_vacía = empty($variable);
echo '$variable = \'\'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_vacía = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
```

```
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_vacía = empty($variable);
echo '$variable = ',$variable,'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
$está_vacía = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_vacía) {
  echo '=> $variable está vacía.<br />';
} else {
  echo '=> $variable no está vacía.<br />';
}
?>
```

Resultado

```
$variable no inicializada
=> $variable está vacía.
$variable = ''
=> $variable está vacía.
$variable = '0'
=> $variable está vacía.
$variable = 0
=> $variable está vacía.
$variable = 'x'
=> $variable no está vacía.
```

isset

La función `isset` permite probar si una o varias variables están definidas o no.

Sintaxis

`booleano` `isset(`*`mixto`* `variable[,...])`

`variable` Variable que se va a probar; pueden ser varias, separadas por una coma.

`isset` devuelve **TRUE** si la variable está definida y **FALSE** en caso contrario.

Si se facilitan varios parámetros, la función devuelve TRUE únicamente si se definen todas las variables.

Una variable se considera como no definida si no se ha visto asignada o si contiene NULL. A diferencia de la función empty, una variable que contiene una cadena vacía (""), una cadena igual a 0 ("0"), un 0, un FALSE o una tabla vacía, no se considera como no definida.

Ejemplo

```
<?php
// Prueba de una variable no inicializada.
$está_definida = isset($variable);
echo '$variable no inicializada<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena vacía.
$variable = '';
$está_definida= isset($variable);
echo '$variable = \'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena igual a 0.
$variable = '0';
$está_definida = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene 0.
$variable = 0;
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
// Prueba de una variable que contiene una cadena no vacía.
$variable = 'x';
```

```
$está_definida = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable no inicializada
=> $variable no está definida.
$variable = ''
=> $variable está definida.
$variable = '0'
=> $variable está definida.
$variable = 0
=> $variable está definida.
$variable = 'x'
=> $variable está definida.
```

unset

La función `unset` permite eliminar una o varias variables.

Sintaxis

```
unset(mixto variable)
```

`variable` Variable que se va a eliminar (para eliminar varias, deben estar separadas por una coma).

Después de la eliminación, la variable se encuentra en el mismo estado que si no hubiera sido asignada. El uso de la función `isset` en una variable eliminada devuelve FALSE.

Ejemplo

```
<?php
// Definir una variable.
$variable = 1;
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
```

```
// Eliminar la variable.
unset($variable);
// Mostrar la variable y probar si está definida.
$está_definida = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($está_definida) {
  echo '=> $variable está definida.<br />';
} else {
  echo '=> $variable no está definida.<br />';
}
?>
```

Resultado

```
$variable = 1
=> $variable está definida.
$variable =
=> $variable no está definida.
```

Observación

Al asignar un 0 o una cadena vacía a una variable, no se borra.

var_dump

La función `var_dump` muestra información sobre una o varias variables (tipo y contenido).

Sintaxis

```
var_dump(mixto variable, [,...])
```

`variable` Variable que se va a mostrar (pueden ser varias, separadas por una coma).

La función `var_dump` es especialmente interesante en las fases de desarrollo.

Ejemplo

```
<?php
// mostrar la información sobre una variable no inicializada
$variable = NULL;
var_dump($variable);
// inicializar la variable con un número entero
$variable = 10;
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 3.14; // número decimal
```

```
// mostrar la información sobre una variable
echo '<br />';
var_dump($variable);
// modificar el valor (y el tipo) de la variable
$variable = 'abc'; // cadena de caracteres
// mostrar la información sobre la variable
echo '<br />';
var_dump($variable);
?>
```

Resultado

```
NULL
int(10)
float(3.14)
string(3) "abc"
```

Para una variable no inicializada, `var_dump` devuelve `NULL`. Para un número, `var_dump` indica el tipo (`int` = entero, `float` = número decimal), seguido por el valor entre paréntesis. Para una cadena, `var_dump` indica el tipo (`string`), seguido de la longitud entre paréntesis, seguido por el valor entre comillas.

PHP también ofrece las funciones `print_r` y `var_export`, que son similares a la función `var_dump`. La función `print_r` muestra o devuelve el contenido de la variable en una forma más legible, sin mencionar el tipo de datos. La función `var_export` muestra o devuelve una cadena que ofrece un código PHP de definición de la variable.

Observación

En la sección Tipos de datos de este capítulo, estudiaremos otras funciones que permiten determinar el tipo de una variable y realizar conversiones de tipos (de número a cadena, de cadena a número...).

2.3 Tipos de datos

2.3.1 Conversiones

PHP es capaz de realizar las conversiones automáticas implícitas de tipo.

Cuando un valor/expresión se asigna a una variable, la variable pasa a ser el tipo de valor/expresión.

Capítulo 3
Entender y preparar los datos

1. Introducción

Antes de empezar a crear nuestros propios algoritmos que nos permitirán obtener modelos eficaces, hay un paso fundamental que nunca debemos subestimar: **la preparación de los datos**.

De hecho, cualquier proyecto de IA requerirá recopilar datos, explorarlos visualizándolos para comprenderlos mejor, limpiarlos y ponerlos a disposición para poder crear modelos adecuados y eficaces. En función de estos y de los resultados deseados, tendremos que elegir algunos algoritmos más adecuados que otros. La calidad de los datos es la clave del éxito de cualquier proyecto de IA. Sin datos limpios, relevantes y bien estructurados, incluso los algoritmos más sofisticados fracasarán.

Descubramos esta etapa fundamental explorando los siguientes temas:

– tipos de datos,
– tipos de aprendizaje,
– soluciones de visualización de datos,
– recopilación, limpieza e imputación de datos,
– exploración y análisis de datos aplicados a PHP,
– preprocesamiento de datos,

– soluciones de reducción de dimensiones.

Al final de este capítulo, conoceremos los diferentes tipos de datos que podemos encontrar tras su recopilación y dispondremos de herramientas para visualizarlos, limpiarlos y analizarlos con el fin de comprenderlos mejor y prepararlos para su explotación.

2. Tipos de datos

En un enunciado de problema que se va a tratar con IA, el primer paso es recopilar los datos que nos permitirán educar y entrenar nuestros modelos, cuya función será resolver nuestro problema. En los conjuntos que tendremos, será posible encontrar una gran variedad de tipos de datos. A continuación, se muestra una lista de los tipos más comunes.

2.1 Datos numéricos

Los datos numéricos pueden ser **continuos** o **discretos**. Son **discretos** si solo pueden tomar una cantidad limitada de valores en un intervalo dado (por ejemplo, el número de personas presentes en una sala solo puede tomar los valores 0, 1, 2... 30). Son **continuos** si pueden tomar una infinidad de valores en un intervalo definido (por ejemplo, la temperatura de una sala de estar).

2.2 Datos categóricos

Los datos **categóricos** toman valores que representan categorías distintas en un panel determinado.

Entre estos datos categóricos, podríamos encontrar, por ejemplo, los tipos de productos (electrónica, ropa, alimentación), los géneros (masculino, femenino, otro) o incluso los colores (rojo, azul, verde).

2.3 Datos ordinales

Los datos **ordinales** son datos categóricos con un orden intrínseco.

Por ejemplo, datos como los niveles de educación (primaria, secundaria, universitaria), las escalas de satisfacción (insatisfecho, neutral, satisfecho) o las clasificaciones de rendimiento (malo, medio, bueno) son ordinales.

2.4 Datos textuales

Los datos **textuales** son datos en forma de texto libre.

Por ejemplo, los comentarios de los clientes, los artículos de blog o el cuerpo de un correo electrónico son datos textuales.

2.5 Datos temporales

Los datos **temporales** son datos indexados en el tiempo.

Las series cronológicas de ventas diarias, los datos de sensores recopilados cada minuto o los registros de temperaturas horarias son datos temporales que podríamos encontrar en nuestros proyectos.

2.6 Datos geoespaciales

Los datos **geoespaciales** son datos asociados a coordenadas geográficas. Por ejemplo, la localización GPS de vehículos, las coordenadas de tiendas y las densidades de población forman parte de este tipo de datos.

2.7 Datos multimedia

Los datos **multimedia** pueden ser datos de audio, imágenes o vídeo.

Las grabaciones de voz para asistentes vocales, los sonidos de animales para proyectos de reconocimiento de fauna, los vídeos deportivos para el análisis del rendimiento, las radiografías médicas, las imágenes satelitales y las fotos de productos para sitios de comercio electrónico, son buenos ejemplos de este tipo de datos.

2.8 Datos lógicos/binarios

Los datos **binarios** son datos que representan dos estados (verdadero/falso, sí/no), como los indicadores de activación de funciones (activado/desactivado), las respuestas a preguntas de sí/no o la presencia/ausencia de características específicas.

Estos diferentes tipos de datos se pueden utilizar solos o combinados para resolver diversos problemas de IA, ya que cada tipo de datos tiene sus propios métodos y tratamientos. De hecho, vamos a aprender a comprender estos datos para que un algoritmo de IA pueda utilizarlos. No olvidemos que una máquina no tiene intrínsecamente la capacidad de comprender el simbolismo de un dato, por lo que será necesario transformar los datos digitalmente conservando sus principales significados y características. Antes de analizar los datos, es importante situar nuestro problema para identificar el modo de aprendizaje que tendremos que utilizar para resolver nuestra necesidad.

3. Tipos de aprendizaje

En Machine Learning, nos encontraremos principalmente con dos enfoques para educar y entrenar nuestros modelos:

- **Aprendizaje supervisado**: este tipo de aprendizaje utiliza datos etiquetados (a menudo el resultado esperado), en los que cada ejemplo de entrenamiento se asocia a una respuesta o salida conocida.
  - **Si las etiquetas son datos numéricos continuos**, nos encontramos ante un problema de **regresión**.
  - **Si las etiquetas son datos categóricos**, nos encontramos ante un problema de **clasificación**.

Ejemplo concreto

Quiere enseñar a su hijo los nombres de las capitales de todos los países del mundo. Le hace preguntas y espera a que le dé una respuesta. A continuación, le da la capital correcta. De este modo, él corrige su comportamiento en función de la respuesta (corrección o confirmación).

- **Aprendizaje no supervisado**: este tipo de aprendizaje utiliza datos sin etiquetar. El modelo busca estructuras o patrones inherentes a los datos sin supervisión humana.

Ejemplo concreto

Tiene una bolsa con múltiples objetos y debe clasificarlos en varios grupos. Depende de usted buscar criterios para definir estos grupos y crearlos. Algunos los clasificarán por tipos de uso, otros por colores, etc. Dependerá del algoritmo implementado.

Así, en función del conjunto de datos que se nos proporcione, ya podremos definir nuestro modo de aprendizaje. Esto nos permitirá posteriormente identificar los algoritmos que nos permitirán resolver nuestro problema. De hecho, dispondremos de todo un arsenal de algoritmos para llevar a cabo estos aprendizajes y algunos serán más adecuados que otros para determinados tipos de datos y también en función de su número.

Al visualizar y analizar los datos con más detalle, podremos precisar nuestra elección de algoritmos.

4. Soluciones de visualización de datos

Para identificar bien nuestro problema y poder abordarlo mejor posteriormente, observemos nuestros datos para comprenderlos mejor. El objetivo de esta fase de análisis es obtener una estimación de la distribución de los datos para hacernos una idea de su representatividad. ¿Están los datos bien distribuidos o nuestra muestra solo representa determinados tipos o intervalos en particular? Esto nos permitirá conocer las posibles limitaciones de nuestro modelo.

Dado que PHP no es un lenguaje destinado a la creación visual, utilizaremos una librería JavaScript optimizada para la visualización de datos. Nuestra elección recaerá en **Plotly.js**, una capa superior del famoso **D3.js**.

Para evitar perder demasiado tiempo y poder centrarnos en nuestras necesidades reales, utilizaremos una librería PHP que permite generar el código para los diagramas directamente desde PHP en unas pocas líneas sencillas, disponible en esta dirección: https://github.com/LouisAUTHIE/Php2Plotly/

Lo más sencillo será delegar su instalación a **Composer**, mediante el siguiente comando:

```
composer require louisauthie/php2plotly
```

Lo único que hay que hacer antes de utilizarla es incluir la librería Plotly.js en nuestro código HTML, dentro de la etiqueta `<head>` de nuestra página. Por ejemplo, podríamos utilizar el siguiente código:

```
<script src="js/plotly-2.32.0.min.js" charset="utf-8"></script>
```

Presentemos rápidamente los tipos de diagramas que tenemos a nuestra disposición. Aprovecharemos para ofrecer un ejemplo de implementación para cada uno de los diagramas.

4.1 Diagrama de barras

Este tipo de diagrama permite visualizar el número de personas en las ordenadas en función de las categorías en las abscisas. A continuación, se muestra uno que ilustra el número de personas por grupos de edad en una población definida.

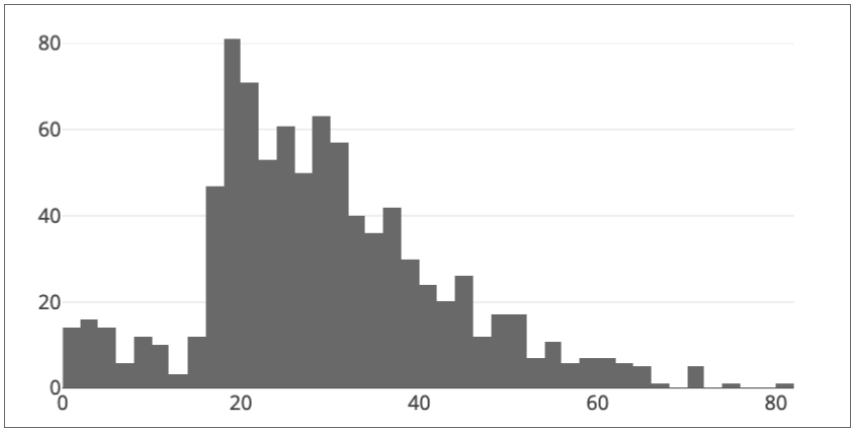

Figura 3.1 - Ejemplo de diagrama de barras

Este tipo de diagrama se utilizará mucho en nuestros estudios de datos para datos numéricos discretos (sin necesidad de reelaboración) y numéricos continuos (con la creación de intervalos de datos para definir categorías en las coordenadas horizontales).

Utilizando nuestra librería de visualización desde PHP, podemos incluir un diagrama de barras con el siguiente código:

```
<div id="bar" style="width:600px;height:400px;"></div>

<?php
       $bar = new BarChart('bar', ['x' => ["Football", "Rugby",
"Handball", "BasketBall"], 'y' => [34, 15, 18, 26]]);
       echo '<script>'.$bar->render().'</script>';
?>
```

Este es el resultado de este ejemplo:

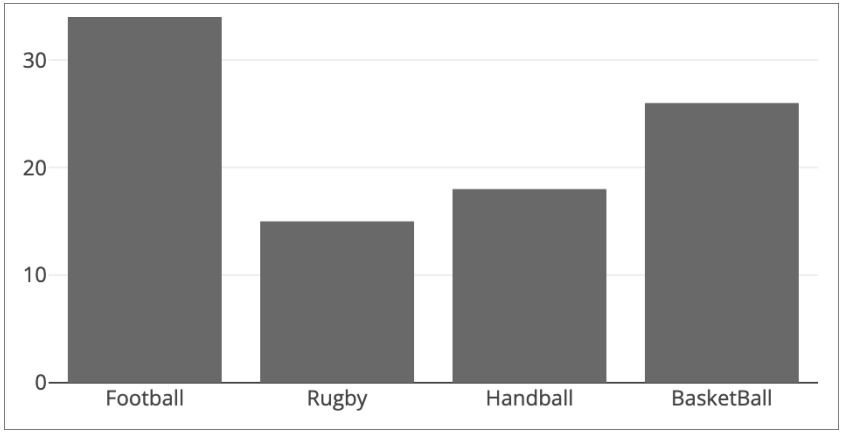

Figura 3.2: resultado de nuestro ejemplo de diagrama de barras

Simplemente instanciamos el objeto `BarChart`, cuyo constructor toma como argumentos el identificador (id) del contenedor en el que se mostrará el diagrama y una tabla asociativa con la clave `x` para las abscisas y la clave `y` para las ordenadas.

4.2 Diagrama de tarta

Este tipo de diagrama permite visualizar las proporciones de los efectivos de cada categoría dentro de todas las demás categorías de un mismo conjunto de datos.

He aquí un ejemplo sencillo que muestra las proporciones de personas de sexo masculino y femenino en la población estudiada.

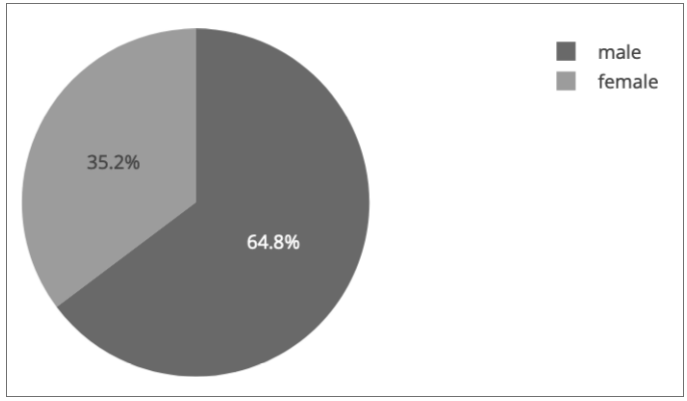

Figura 3.3 - Ejemplo de diagrama de tarta

Este tipo de diagrama es especialmente adecuado para visualizar datos categóricos, ordinales y binarios.

Un ejemplo de código para crear un diagrama de este tipo podría ser el siguiente:

```
<div id="pie" style="width:600px;height:400px;"></div>
<?php
   $pie = new PieChart('pie', ['values' => [10, 15, 13, 17],
'labels' => ["Gatos", "Perros", "Pájaros", "Ositos"]],
['height' => 400, 'width' => 600]);
   echo '<script>'.$pie->render().'</script>';
?>
```