Programación con fastai

Autor.

- Arturo Cortés Sánchez

Índice

Introducción

fast.ai es un grupo de investigación sin ánimo de lucro centrado en el aprendizaje profundo y la inteligencia artificial. Fue fundado en 2016 por Jeremy Howard y Rachel Thomas con el objetivo de democratizar el aprendizaje profundo. Para ello, imparten un curso online masivo y abierto (MOOC) llamado "Practical Deep Learning for Coders", que no tiene más requisitos previos que el conocimiento del lenguaje de programación Python.

En otoño de 2018, fast.ai lanzó la v1.0 de su biblioteca gratuita de código abierto para el aprendizaje profundo llamada fastai (sin punto).

fastai es una biblioteca de aprendizaje profundo que proporciona componentes de alto nivel que pueden proporcionar rápida y fácilmente resultados de vanguardia en dominios estándar de aprendizaje profundo, y proporciona a sus usuarios componentes de bajo nivel que pueden mezclarse y combinarse para construir nuevos enfoques.
Su objetivo es hacer ambas cosas sin comprometer sustancialmente la facilidad de uso, la flexibilidad o el rendimiento. Esto es posible gracias a una arquitectura cuidadosamente estratificada, que expresa los patrones subyacentes comunes de muchas técnicas de aprendizaje profundo y procesamiento de datos en términos de abstracciones desacopladas. Estas abstracciones pueden expresarse de forma concisa y clara aprovechando el dinamismo del lenguaje Python y la flexibilidad de la biblioteca PyTorch.

PyTorch es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca Torch, utilizada para aplicaciones como visión por ordenador y procesamiento del lenguaje natural, es desarrollada principalmente por el laboratorio de investigación de IA de Facebook.

fastai incluye entre otras cosas:

- Un nuevo sistema de tipos para Python junto con una jerarquía de tipos semántica para tensores.
- Una biblioteca de visión por ordenador optimizada para la GPU que puede ampliarse en Python.
- Un optimizador que refactoriza la funcionalidad común de los optimizadores modernos en dos piezas básicas, permitiendo implementar algoritmos de optimización en 4-5 líneas de código.
- Un novedoso sistema de callbacks de dos vías que puede acceder a cualquier parte de los datos, el modelo o el optimizador y cambiarlo en cualquier momento durante el entrenamiento.
- Una API de bloques de datos.

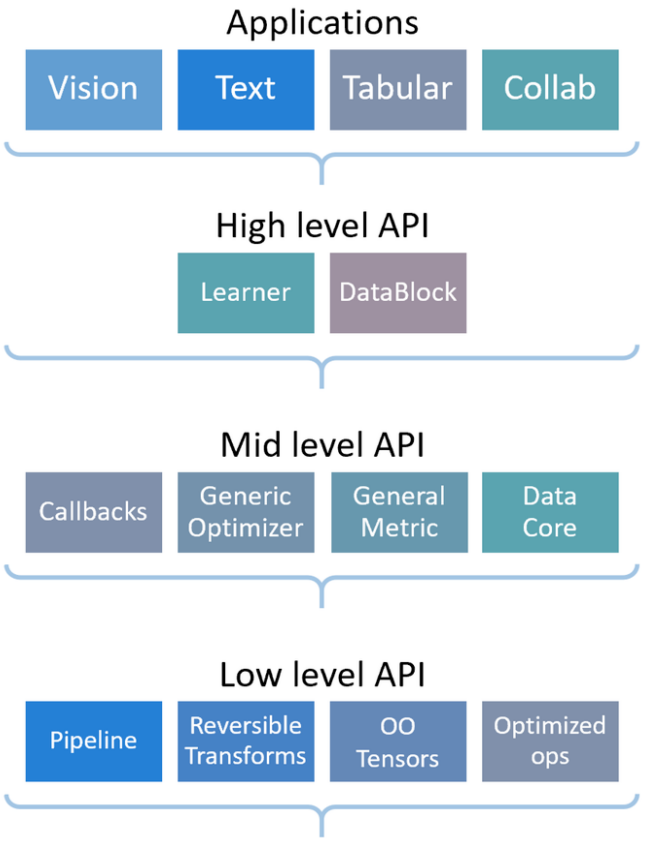
fastai está organizado en torno a dos objetivos principales de diseño: ser accesible y rápidamente productivo, mientras que también es profundamente hackeable y configurable. Está construido sobre una jerarquía de APIs de nivel inferior que proporcionan bloques de

construcción componibles. De este modo, un usuario que quiera reescribir parte de la API de alto nivel o añadir un comportamiento concreto que se adapte a sus necesidades no tiene que aprender a utilizar el nivel más bajo.

Flujo de trabajo con fastai

Todas las aplicaciones de fastai utilizan los mismos pasos básicos y el mismo código:

Crear los DataLoaders adecuados
Crear un Learner
Llamar a un método fit
Hacer predicciones o ver los resultados.



DataLoaders

La clase DataLoaders es una envoltura básica de varios DataLoaders, mientras que la clase DataLoader de fastai es una extensión de la clase DataLoader de PyTorch que añade muchos más callbacks y flexibilidad. Las primitivas DataLoader y Dataset de PyTorch permiten utilizar conjuntos de datos precargados así como datos propios. Dataset almacena las muestras y sus correspondientes etiquetas, y DataLoader la envuelve en un objeto iterable para permitir un fácil acceso a las muestras.

El constructor de DataLoader (fastai) admite los siguientes argumentos:

- dataset: conjunto de datos desde el que se cargan los datos. Puede ser un conjunto de datos de tipo map o iterable.
- bs (int): el número de muestras por lote que se va a cargar (si se proporciona batch_size, batch_size sustituirá a bs).
- num_workers (int): cuántos subprocesos se utilizarán para la carga de datos. 0 significa que los datos se cargarán en el proceso principal.
- pin_memory (bool): Si es True, el cargador de datos copiará los Tensores en la memoria anclada de CUDA antes de devolverlos.
- timeout (float>0): el valor del tiempo de espera en segundos para reunir un lote de los trabajadores.
- batch_size (int): Sólo se proporciona por compatibilidad con PyTorch. Se recomienda utilizar bs.
- shuffle (bool): Si es True, los datos se barajan cada vez que el dataloader es leído/iterado completamente.
- drop_last (bool): Si es True, entonces el último lote incompleto es descartado.
- index (bool): El DataLoader hará una conjetura sobre si el conjunto de datos puede ser indexado (o es iterable), pero se puede anular con este parámetro. Por defecto es True.
- n (int): Por defecto es len(dataset). Si utiliza un conjunto de datos de tipo iterable, puede especificar el tamaño con n.
- device (torch.device): Por defecto es default_device() que es CUDA por defecto. Puede especificar el dispositivo como torch.device('cpu').

Learner

La clase learner agrupa un modelo, unos DataLoaders y una función de pérdida. Con estos parámetros es suficiente para entrenar la red neuronal. Un modelo en el contexto de fastai se refiere a uno de los múltiples modelos pre entrenados que esta biblioteca incorpora.

Clasificación de imágenes: Razas de perros y gatos

Para esta tarea, utilizaremos el conjunto de datos de mascotas Oxford-IIIT que contiene imágenes de gatos y perros de 37 razas diferentes.

Lo primero es importar las funciones de procesamiento y clasificación de imagenes de fastai

```
from fastai.vision.all import *
```

El conjunto de datos se puede descargar y descomprimir con esta línea de código:

```
path = untar_data(URLs.PETS)
path.ls()
```

Sólo hará esta descarga una vez, y devolverá la ubicación del archivo descomprimido. Podemos comprobar lo que hay dentro con el método .ls().

get_image_files es una función fastai que nos ayuda a coger todos los archivos de imágenes (recursivamente) en una carpeta.

```
files = get_image_files(path/"images")
```

Entonces podemos crear un Learner, que es un objeto fastai que combina los datos y un modelo para el entrenamiento, y utiliza el aprendizaje de transferencia para afinar un modelo preentrenado en sólo dos líneas de código:

Para tener nuestros datos listos para un modelo, necesitamos ponerlos en un objeto DataLoaders. Para etiquetar nuestros datos con el nombre de la raza, utilizaremos una expresión regular para extraerla del nombre del archivo. Mirando un nombre de archivo, tenemos:

```
files[0].name
```

'great_pyrenees_173.jpg'

Podemos ver que la clase es todo lo que está antes del último _ seguido de algunos dígitos. Una expresión regular que atrapará el nombre es así:

```
pat = r'^(.*)_\d+.jpg'
```

Dado que es bastante común utilizar expresiones regulares para etiquetar los datos (a menudo, las etiquetas están ocultas en los nombres de los archivos), existe un método factoría para hacer precisamente eso:

```
dls = ImageDataLoaders.from_name_re(path, files, pat,
item_tfms=Resize(224))
```

Hemos pasado a esta función el directorio en el que estamos trabajando, los archivos que hemos cogido, la expresión regular y un último argumento item_tfms: se trata de una transformación aplicada a todos los elementos de nuestro conjunto de datos que redimensionará cada imagen a 224 por 224, utilizando un recorte aleatorio en la dimensión

mayor para convertirla en un cuadrado, y luego redimensionando a 224 por 224. Si no hiciésemos esto, obtendríamos un error más tarde, ya que sería imposible agrupar los elementos.

A continuación, podemos comprobar si todo está bien con el método show_batch

```
dls.show_batch()
```

Dado que clasificar la raza exacta de gatos o perros entre 37 razas diferentes es un problema relativamente difícil, cambiaremos ligeramente la definición del DataLoaders para utilizar data augmentation:

```
dls = ImageDataLoaders.from_name_re(path, files, pat,
item_tfms=Resize(460), batch_tfms=aug_transforms(size=224))
```

Esta vez cambiamos el tamaño de las imágenes antes de la agrupación por lotes, y añadimos batch_tfms. aug_transforms es una función que proporciona una colección de transformaciones para data augmentation con valores predeterminados que funcionan bien en muchos conjuntos de datos. Estas transformaciones pueden ser personalizadas pasando los argumentos adecuados a aug_transnforms.

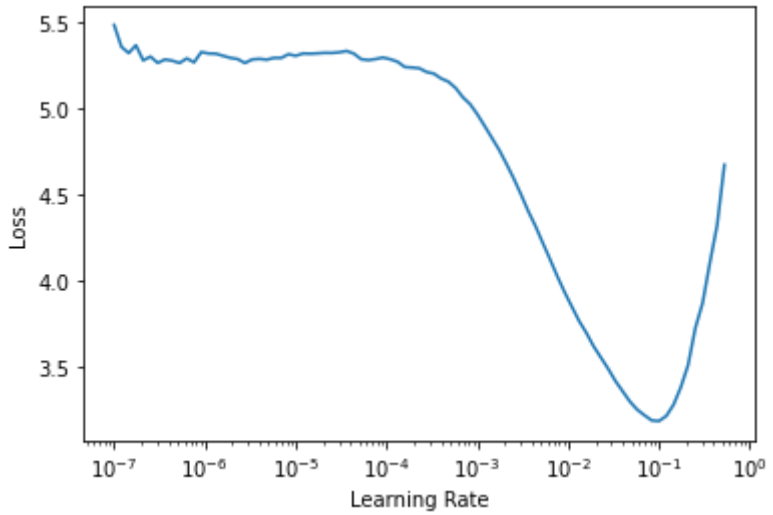
Entonces podemos crear un Learner, que es un objeto fastai que combina los datos y un modelo para el entrenamiento, y utiliza transferencia de aprendizaje para afinar un modelo pre entrenado en sólo dos líneas de código:

```
learn = cnn_learner(dls, resnet34, metrics=error_rate)
```

La primera línea descarga un modelo llamado ResNet34, preentrenado en ImageNet, y lo adapta a este problema específico.

En lugar de utilizar el learning rate por defecto, podemos tratar de encontrar la mejor posible. Para ello, podemos utilizar el buscador de learning rates:

```
learn.lr_find()
SuggestedLRs(lr_min=0.010000000149011612,
lr_steep=0.0063095735386013985)
```



Esta función dibuja la gráfica de la learning rates y nos da dos sugerencias (mínimo dividido por 10 y gradiente más pronunciado). Vamos a usar un learning rate de 3e-3 y dos epochs:

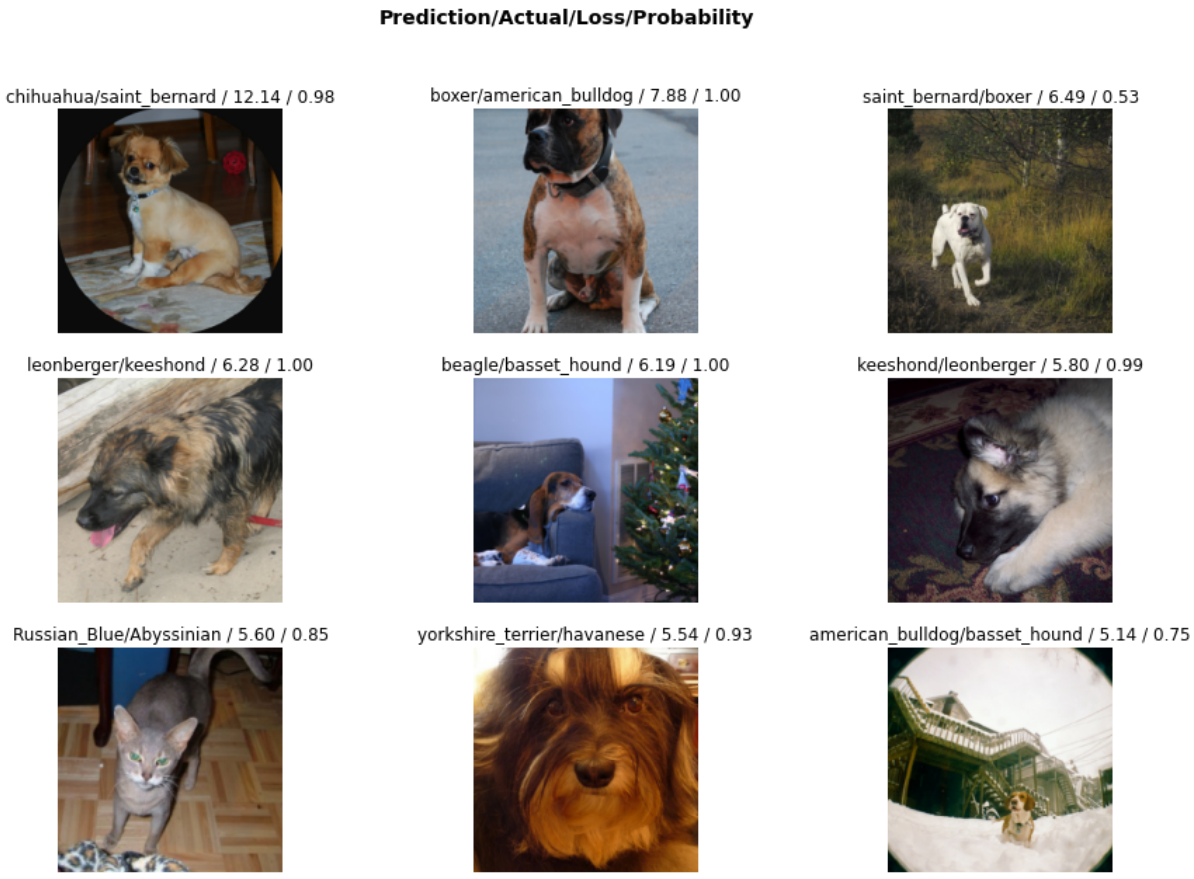
```
learn.fine_tune(2, 3e-3)
```

podemos echar un vistazo a algunas predicciones con show_results:

```
learn.show_results()
```

Otra funcionalidad útil son los objetos de interpretación, pueden mostrarnos dónde el modelo ha hecho las peores predicciones:

```
interp = Interpretation.from_learner(learn)
interp.plot_top_losses(9, figsize=(15,10))
```



Clasificación de texto: Críticas de IMDb

En este ejemplo, veremos cómo se puede entrenar un modelo para clasificar texto, basado en el sentimiento de su autor (análisis de sentimientos). Utilizaremos el conjunto de datos de IMDb del artículo Learning Word Vectors for Sentiment Analysis, que contiene unos cuantos miles de críticas de películas.

Igual que en el ejemplo anterior comenzamos importando fastai, pero en este caso la funcionalidad de análisis de texto:

```
from fastai.text.all import *
```

Descargamos los datos y los descomprimimos con el siguiente comando:

```
path = untar_data(URLs.IMDB)
```

Los datos siguen una organización al estilo de ImageNet, en la carpeta train, tenemos dos subcarpetas, pos y neg (para las reseñas positivas y las negativas). Podemos agruparlos utilizando el método TextDataLoaders.from_folder. Lo único que tenemos que especificar es el nombre de la carpeta de validación, que es "test" (y no el predeterminado "valid").

```
dls = TextDataLoaders.from_folder(untar_data(URLs.IMDB), valid='test')
```

A continuación, podemos echar un vistazo a los datos con el método show_batch:

```
dls.show_batch()
```

	text	category
0	xxbos xxmaj match 1 : xxmaj tag xxmaj team xxmaj table xxmaj match xxmaj bubba xxmaj ray and xxmaj spike xxmaj dudley vs xxmaj eddie xxmaj guerrero and xxmaj chris xxmaj benoit xxmaj bubba xxmaj ray and xxmaj spike xxmaj dudley started things off with a xxmaj tag xxmaj team xxmaj table xxmaj match against xxmaj eddie xxmaj guerrero and xxmaj chris xxmaj benoit . xxmaj according to the rules of the match , both opponents have to go through tables in order to get the win . xxmaj benoit and xxmaj guerrero heated up early on by taking turns hammering first xxmaj spike and then xxmaj bubba xxmaj ray . a xxmaj german xxunk by xxmaj benoit to xxmaj bubba took the wind out of the xxmaj dudley brother . xxmaj spike tried to help his brother , but the referee restrained him while xxmaj benoit and xxmaj guerrero	pos
1	xxbos xxmaj warning : xxmaj does contain spoilers . \n\n xxmaj open xxmaj your xxmaj eyes \n\n xxmaj if you have not seen this film and plan on doing so , just stop reading here and take my word for it . xxmaj you have to see this film . i have seen it four times so far and i still have n't made up my mind as to what exactly happened in the film . xxmaj that is all i am going to say because if you have not seen this film , then stop reading right now . \n\n xxmaj if you are still reading then i am going to pose some questions to you and maybe if anyone has any answers you can email me and let me know what you think . \n\n i remember my xxmaj grade 11 xxmaj english teacher quite well . xxmaj	pos

Se han recortado los resultados para ahorrar espacio

Podemos ver que la biblioteca procesó automáticamente todos los textos para dividirlos en tokens, añadiendo algunos tokens especiales como

- xxbos para indicar el comienzo de un texto
- xxmaj para indicar que la siguiente palabra está en mayúsculas

A continuación, podemos definir un Learner adecuado para la clasificación de textos en una línea:

```
learn = text_classifier_learner(dls, AWD_LSTM, drop_mult=0.5,
metrics=accuracy)
```

Utilizamos la arquitectura AWD LSTM, drop_mult es un parámetro que controla la magnitud de los dropouts en ese modelo, y utilizamos accuracy para saber como de bien lo estamos haciendo. Así podemos ajustar nuestro modelo preentrenado:

```
learn.fine_tune(4, 1e-2)
```

Para ver lo bien que va nuestro modelo, podemos utilizar el método show_results:

```
learn.show_results()
```

	text	category	category_
0	xxbos xxmaj there 's a sign on xxmaj the xxmaj lost xxmaj highway that says : \n\n * major xxup spoilers xxup ahead * \n\n (but you already knew that , did n't you ?) \n\n xxmaj since there 's a great deal of people that apparently did not get the point of this movie , xxmaj i 'd like to contribute my interpretation of why the plot makes perfect sense . xxmaj as others have pointed out , one single viewing of this movie is not sufficient . xxmaj if you have the xxup dvd of xxup md , you can " cheat " by looking at xxmaj david xxmaj lynch 's " top 10 xxmaj hints to xxmaj unlocking xxup md " (but only upon second or third viewing , please .) ;) \n\n xxmaj first of all , xxmaj mulholland xxmaj drive is	pos	pos
1	xxbos (some spoilers included :) \n\n xxmaj although , many commentators have called this film surreal , the term fits poorly here . xxmaj to quote from xxmaj encyclopedia xxmaj xxunk 's , surreal means : \n\n " fantastic or incongruous imagery " : xxmaj one need n't explain to the unimaginative how many ways a plucky ten - year - old boy at large and seeking his fortune in the driver 's seat of a red xxmaj mustang could be fantastic : those curious might read xxmaj james xxmaj kincaid ; but if you asked said lad how he were incongruous behind the wheel of a sports car , he 'd surely protest , " no way ! " xxmaj what fantasies and incongruities the film offers mostly appear within the first fifteen minutes . xxmaj thereafter we get more iterations of the same , in an	pos	neg

2	xxbos xxmaj hearkening back to those " good xxmaj old xxmaj days " of 1971 , we can vividly recall when we were treated with a whole xxmaj season of xxmaj charles xxmaj chaplin at the xxmaj cinema . xxmaj that 's what the promotional guy called it when we saw him on somebody 's old talk show . (we ca n't recall just whose it was ; either xxup merv xxup griffin or xxup woody xxup woodbury , one or the other !) xxmaj the guest talked about xxmaj sir xxmaj charles ' career and how his films had been out of circulation ever since the 1952 exclusion of the former " little xxmaj tramp ' from xxmaj los xxmaj xxunk xxmaj xxunk on the grounds of his being an " undesirable xxmaj alien " . (no xxmaj schultz , he 's xxup not from another	pos	pos
3	xxbos " buffalo xxmaj bill , xxmaj hero of the xxmaj far xxmaj west " director xxmaj mario xxmaj costa 's unsavory xxmaj spaghetti western " the xxmaj beast " with xxmaj klaus xxmaj kinski could only have been produced in xxmaj europe . xxmaj hollywood would never dared to have made a western about a sexual predator on the prowl as the protagonist of a movie . xxmaj never mind that xxmaj kinski is ideally suited to the role of ' crazy ' xxmaj johnny . xxmaj he plays an individual entirely without sympathy who is ironically dressed from head to toe in a white suit , pants , and hat . xxmaj this low - budget oater has nothing appetizing about it . xxmaj the typically breathtaking xxmaj spanish scenery around xxmaj almeria is nowhere in evidence . xxmaj instead , xxmaj costa and his director of photography	pos	pos
4	xxbos xxmaj if you 've seen the trailer for this movie , you pretty much know what to expect , because what you see here is what you get . xxmaj and even if you have n't seen the previews , it wo n't take you long to pick up on what you 're in for-- specifically , a good time and plenty of xxunk from this clever satire of ` reality xxup tv ' shows and ` buddy xxmaj cop ' movies , ` showtime , ' directed by xxmaj tom xxmaj dey , starring xxmaj robert xxmaj de xxmaj niro and xxmaj eddie xxmaj murphy . \n\n\t xxmaj mitch xxmaj preston (de xxmaj niro) is a detective with the xxup l.a.p.d . , and he 's good at what he does ; but working a case one night , things suddenly go south when another cop	pos	pos

5	xxbos * xxmaj some spoilers * \n\n xxmaj this movie is sometimes subtitled " life xxmaj everlasting . " xxmaj that 's often taken as reference to the final scene , but more accurately describes how dead and buried this once - estimable series is after this sloppy and illogical send - off . \n\n xxmaj there 's a " hey kids , let 's put on a show air " about this telemovie , which can be endearing in spots . xxmaj some fans will feel like insiders as they enjoy picking out all the various cameo appearances . xxmaj co - writer , co - producer xxmaj tom xxmaj fontana and his pals pack the goings - on with friends and favorites from other shows , as well as real xxmaj baltimore personages . \n\n xxmaj that 's on top of the returns of virtually all the members	neg	neg

Se ha recortado la salida

Y podemos predecir sobre nuevos textos con bastante facilidad:

```
learn.predict("I really liked that movie!")
```

('pos', tensor(1), tensor([0.0092, 0.9908]))

El método predict devuelve tres cosas: la predicción decodificada, el índice de la clase predicha y el tensor de probabilidades de todas las clases en el orden de sus etiquetas indexadas. Aquí podemos ver que el modelo ha considerado la reseña como positiva. La segunda parte del resultado es el índice de "pos" en nuestro vocabulario de datos y la última parte son las probabilidades atribuidas a cada clase (99,1% para "pos" y 0,9% para "neg").

Entrenamiento Tabular: Predicción de salarios

Para ilustrar la aplicación tabular, utilizaremos el ejemplo del conjunto de datos de Adultos en el que tenemos que predecir si una persona gana más o menos de 50.000 dólares al año utilizando algunos datos generales.

Como siempre comenzamos importando la funcionalidad requerida de fastai

```
from fastai.tabular.all import *
```

Podemos descargar una muestra de este conjunto de datos con el comando habitual untar_data:

```
path = untar_data(URLs.ADULT_SAMPLE)
```

A continuación, podemos echar un vistazo a la estructura de los datos:

```
df = pd.read_csv(path/'adult.csv')
df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	49	Private	101320	Assoc-acdm	12.0	Married-civ-spouse	NaN	Wife	White	Female	0	1902	40	United-States	>=50k
1	44	Private	236746	Masters	14.0	Divorced	Exec-managerial	Not-in-family	White	Male	10520	0	45	United-States	>=50k
2	38	Private	96185	HS-grad	NaN	Divorced	NaN	Unmarried	Black	Female	0	0	32	United-States	<50k
3	38	Self-emp-inc	112847	Prof-school	15.0	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	United-States	>=50k
4	42	Self-emp-not-inc	82297	7th-8th	NaN	Married-civ-spouse	Other-service	Wife	Black	Female	0	0	50	United-States	<50k

Algunas de las columnas son continuas (como la edad) y las trataremos como números flotantes que podemos alimentar directamente a nuestro modelo. Otras son categóricas (como la clase de trabajo o la educación) y las convertiremos en un índice único que introduciremos en las capas de incrustación. Podemos especificar los nombres de nuestras columnas categóricas y continuas, así como el nombre de la variable dependiente en los métodos de fábrica de TabularDataLoaders:

```
dls = TabularDataLoaders.from_csv(path/'adult.csv', path=path,
y_names="salary",
    cat_names = ['workclass', 'education', 'marital-status',
'occupation', 'relationship', 'race'],
    cont_names = ['age', 'fnlwgt', 'education-num'],
    procs = [Categorify, FillMissing, Normalize])
```

La última parte es la lista de preprocesadores que aplicamos a nuestros datos:

- Categorify va a tomar cada variable categórica y hacer un mapa de enteros a categorías únicas, luego reemplazará los valores por el índice correspondiente.
- FillMissing rellenará los valores perdidos en las variables continuas por la mediana de los valores existentes (puede elegir un valor específico si lo prefiere)
- Normalize normalizará las variables continuas (resta la media y divide por la std)

Para exponer aún más lo que sucede bajo la superficie, vamos a reescribir esto utilizando la clase TabularPandas de fastai. Tendremos que definir cómo queremos dividir nuestros datos. Por defecto, el método factoría anterior utiliza una división aleatoria 80/20, así que haremos lo mismo:

```
splits = RandomSplitter(valid_pct=0.2)(range_of(df))
```

```
to = TabularPandas(df, procs=[Categorify, FillMissing,Normalize],
                   cat_names = ['workclass', 'education',
'marital-status', 'occupation', 'relationship', 'race'],
                   cont_names = ['age', 'fnlwgt', 'education-num'],
                   y_names='salary',
                   splits=splits)
```

Una vez que construimos nuestro objeto TabularPandas, nuestros datos están completamente preprocesados como se ve a continuación:

```
to.xs.iloc[:2]
```

	workclass	education	marital-status	occupation	relationship	race	education-num_na	age	fnlwgt	education-num
15780	2	16	1	5	2	5	1	0.984037	2.210372	-0.033692
17442	5	12	5	8	2	5	1	-1.509555	-0.319624	-0.425324

Ahora podemos volver a construir nuestros DataLoaders:

```
dls = to.dataloaders(bs=64)
```

Utilizamos ell método show_batch como en los ejemplos anteriores:

```
dls.show_batch()
```

	workclass	education	marital-status	occupation	relationship	race	education-num_na	age	fnlwgt	education-num	salary
0	State-gov	Bachelors	Married-civ-spouse	Prof-specialty	Wife	White	False	41.000000	75409.001182	13.0	>=50k
1	Private	Some-college	Never-married	Craft-repair	Not-in-family	White	False	24.000000	38455.005013	10.0	<50k
2	Private	Assoc-acdm	Married-civ-spouse	Prof-specialty	Husband	White	False	48.000000	101299.003093	12.0	<50k
3	Private	HS-grad	Never-married	Other-service	Other-relative	Black	False	42.000000	227465.999281	9.0	<50k
4	State-gov	Some-college	Never-married	Prof-specialty	Not-in-family	White	False	20.999999	258489.997130	10.0	<50k
5	Local-gov	12th	Married-civ-spouse	Tech-support	Husband	White	False	39.000000	207853.000067	8.0	<50k

Podemos definir un modelo utilizando el método tabular_learner. Cuando definimos nuestro modelo, fastai intentará inferir la función de pérdida basándose en nuestros y_names anteriores.

```
learn = tabular_learner(dls, metrics=accuracy)
```

Y podemos entrenar ese modelo con el método fit_one_cycle (el método fine_tune no será útil aquí ya que no tenemos un modelo preentrenado).

```
learn.fit_one_cycle(1)
```

A continuación, podemos hacer algunas predicciones:

```
row, clas, probs = learn.predict(df.iloc[0])
row.show()
```

	workclass	education	marital-status	occupation	relationship	race	education-num_na	age	fnlwgt	education-num	salary
0	Private	Assoc-acdm	Married-civ-spouse	#na#	Wife	White	False	49.0	101319.99788	12.0	>=50k

Bibliografía

https://en.wikipedia.org/wiki/Fast.ai
https://en.wikipedia.org/wiki/PyTorch
https://github.com/fastai/fastai
https://docs.fast.ai/tutorial.text.html
https://docs.fast.ai/tutorial.vision.html
https://docs.fast.ai/tutorial.tabular.html
https://pytorch.org/tutorials/beginner/basics/data_tutorial.html
https://docs.fast.ai/dev/performance.html