

Machine Learning

K Vecinos Próximos (KNN)

Christian Oliva Moya
Luis Fernando Lago Fernández

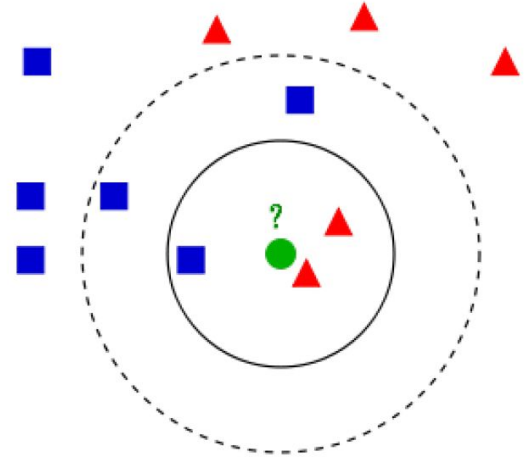
Introducción - K Vecinos Próximos (KNN)

K Vecinos Próximos (K-Nearest Neighbors, KNN) es un algoritmo de ML:

- Supervisado
- De clasificación
- No paramétrico

Además, es realmente intuitivo

Utiliza la **proximidad a los vecinos** para hacer una predicción



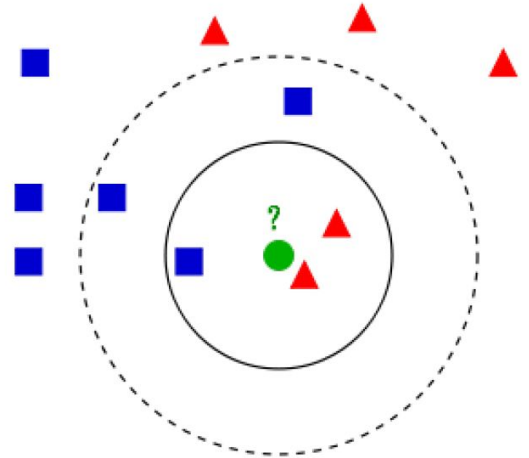
Introducción - K Vecinos Próximos (KNN)

- Para clasificar un nuevo dato x_i hay que observar los K puntos más cercanos y contar cuántos son de la clase a predecir.

$$P(C_j|x_i) = \frac{K_{C_j}}{K}$$

En el ejemplo, el punto verde se clasificaría como ROJO

si $K = 3$, pero se clasificaría como AZUL si $K = 5$

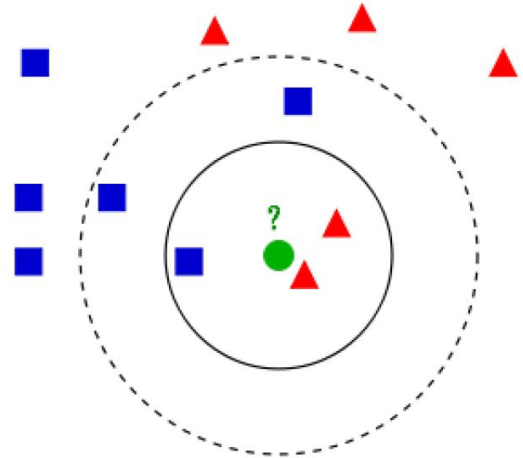


Introducción - K Vecinos Próximos (KNN)

- Métrica de similitud típica: Distancia Euclídea

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

- Donde p y q son dos puntos en un espacio de dimensión n



Introducción - K Vecinos Próximos (KNN)

- Consideraciones:
 - ¿Qué pasa si los atributos tienen diferente rango?
 - ¿Qué pasa si los atributos no son numéricos?
 - Es obligatorio definir el hiperparámetro K.

Observación

La mayoría de los algoritmos de ML necesitan una fase de pre-procesado que prepare las características para que sean útiles para discriminar entre clases

Scikit-Learn - Introducción

- **Sklearn** es una librería de ML con una amplia variedad de herramientas y algoritmos.

Es realmente popular ya que:

- Tiene una interfaz sencilla
- Es directamente compatible con Numpy
- Tiene una amplia documentación, incluyendo tutoriales y ejemplos
- Tiene datasets de introducción realmente interesantes



Scikit-Learn - Introducción

- Si estás buscando un algoritmo de Machine Learning, busca en google:



sklearn knn



Example

Regressor

Imágenes

Videos

Noticias

Libros

Maps

Vuelos

Finance

Aproximadamente 2.810.000 resultados (0,28 segundos)

Sugerencia: Limita esta búsqueda a resultados en **español**. Más información sobre cómo filtrar por idioma



scikit-learn

<http://scikit-learn.org> · generated

sklearn.neighbors.KNeighborsClassifier

Examples using **sklearn.neighbors.KNeighborsClassifier**: Release Highlights for **scikit-learn**

0.24 Classifier comparison Plot the decision boundaries of a ...

[Nearest Neighbors Classification](#) · [Sklearn.metrics.pairwise...](#) · [BallTree](#) · [KDTree](#)



Scikit-Learn - Introducción

- ¿Cómo es la documentación? `from sklearn.neighbors import KNeighborsClassifier`

`sklearn.neighbors.KNeighborsClassifier`

¿Cómo importar la clase?

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2,
metric='minkowski', metric_params=None, n_jobs=None)
```

[\[source\]](#)

Classifier implementing the k-nearest neighbors vote.

Read more in the [User Guide](#).

Parameters:

n_neighbors : int, default=5

Number of neighbors to use by default for `kneighbors` queries.

weights : {'uniform', 'distance'}, callable or None, default='uniform'

Weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.



Scikit-Learn - Introducción

- ¿Cómo es la documentación?

`sklearn.neighbors.KNeighborsClassifier`

Cabecera del constructor

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None) \[source\]
```

Classifier implementing the k-nearest neighbors vote.

Read more in the [User Guide](#).

Parameters:

`n_neighbors` : int, default=5

Number of neighbors to use by default for `kneighbors` queries.

`weights` : {'uniform', 'distance'}, callable or None, default='uniform'

Weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.



Scikit-Learn - Introducción

- ¿Cómo es la documentación?

`sklearn.neighbors.KNeighborsClassifier`

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
```

[\[source\]](#)

Classifier implementing the k-nearest neighbors vote.

Definición de todos los parámetros

[Read more in the User Guide](#)

Parameters:

`n_neighbors` : int, default=5

Number of neighbors to use by default for `kneighbors` queries.

`weights` : {'uniform', 'distance'}, callable or None, default='uniform'

Weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.



Scikit-Learn - Introducción

- Siempre nos dan algún ejemplo y la definición de cada uno de los métodos

Examples

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.666... 0.333...]]
```

Methods

<code>fit(X, y)</code>	Fit the k-nearest neighbors classifier from the training dataset.
<code>get_metadata_routing()</code>	Get metadata routing of this object.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>kneighbors([X, n_neighbors, return_distance])</code>	Find the K-neighbors of a point.
<code>kneighbors_graph([X, n_neighbors, mode])</code>	Compute the (weighted) graph of k-Neighbors for points in X.
<code>predict(X)</code>	Predict the class labels for the provided data.
<code>predict_proba(X)</code>	Return probability estimates for the test data X.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(**params)</code>	Set the parameters of this estimator.
<code>set_score_request(*[, sample_weight])</code>	Request metadata passed to the <code>score</code> method.



Scikit-Learn - Introducción

- Siempre nos dan algún ejemplo y la definición de cada uno de los métodos

Examples

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.666... 0.333...]]
```

fit entrena el modelo

Methods

<code>fit(X, y)</code>	Fit the k-nearest neighbors classifier from the training dataset.
<code>get_metadata_routing()</code>	Get metadata routing of this object.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>kneighbors([X, n_neighbors, return_distance])</code>	Find the K-neighbors of a point.
<code>kneighbors_graph([X, n_neighbors, mode])</code>	Compute the (weighted) graph of k-Neighbors for points in X.
<code>predict(X)</code>	Predict the class labels for the provided data.
<code>predict_proba(X)</code>	Return probability estimates for the test data X.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(**params)</code>	Set the parameters of this estimator.
<code>set_score_request(*[, sample_weight])</code>	Request metadata passed to the <code>score</code> method.



Scikit-Learn - Introducción

- Siempre nos dan algún ejemplo y la definición de cada uno de los métodos

Examples

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.666... 0.333...]]
```

predict devuelve la opción más probable
predict_proba devuelve las probabilidades

Methods

<code>fit(X, y)</code>	Fit the k-nearest neighbors classifier from the training dataset.
<code>get_metadata_routing()</code>	Get metadata routing of this object.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>kneighbors([X, n_neighbors, return_distance])</code>	Find the K-neighbors of a point.
<code>kneighbors_graph([X, n_neighbors, model])</code>	Compute the (weighted) graph of k-Neighbors for points in X.
<code>predict(X)</code>	Predict the class labels for the provided data.
<code>predict_proba(X)</code>	Return probability estimates for the test data X.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(**params)</code>	Set the parameters of this estimator.
<code>set_score_request(*[, sample_weight])</code>	Request metadata passed to the <code>score</code> method.



Scikit-Learn - K Vecinos Próximos (KNN)

Recordatorio: K Vecinos Próximos (K-Nearest Neighbors, KNN) es un algoritmo de ML:

- Supervisado
- De clasificación
- **No paramétrico**

¿Cómo es que tiene un fit?

Methods

<code>fit(X, y)</code>	Fit the k-nearest neighbors classifier from the training dataset.
<code>get_metadata_routing()</code>	Get metadata routing of this object.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>kneighbors([X, n_neighbors, return_distance])</code>	Find the K-neighbors of a point.
<code>kneighbors_graph([X, n_neighbors, mode])</code>	Compute the (weighted) graph of k-Neighbors.
<code>predict(X)</code>	Predict the class labels for the provided data.
<code>predict_proba(X)</code>	Return probability estimates for the test data.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on a given test set.
<code>set_params(**params)</code>	Set the parameters of this estimator.
<code>set_score_request(*[, sample_weight])</code>	Request metadata for the score method.

<



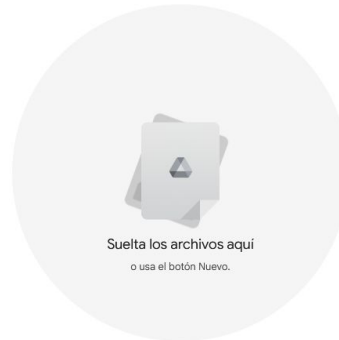
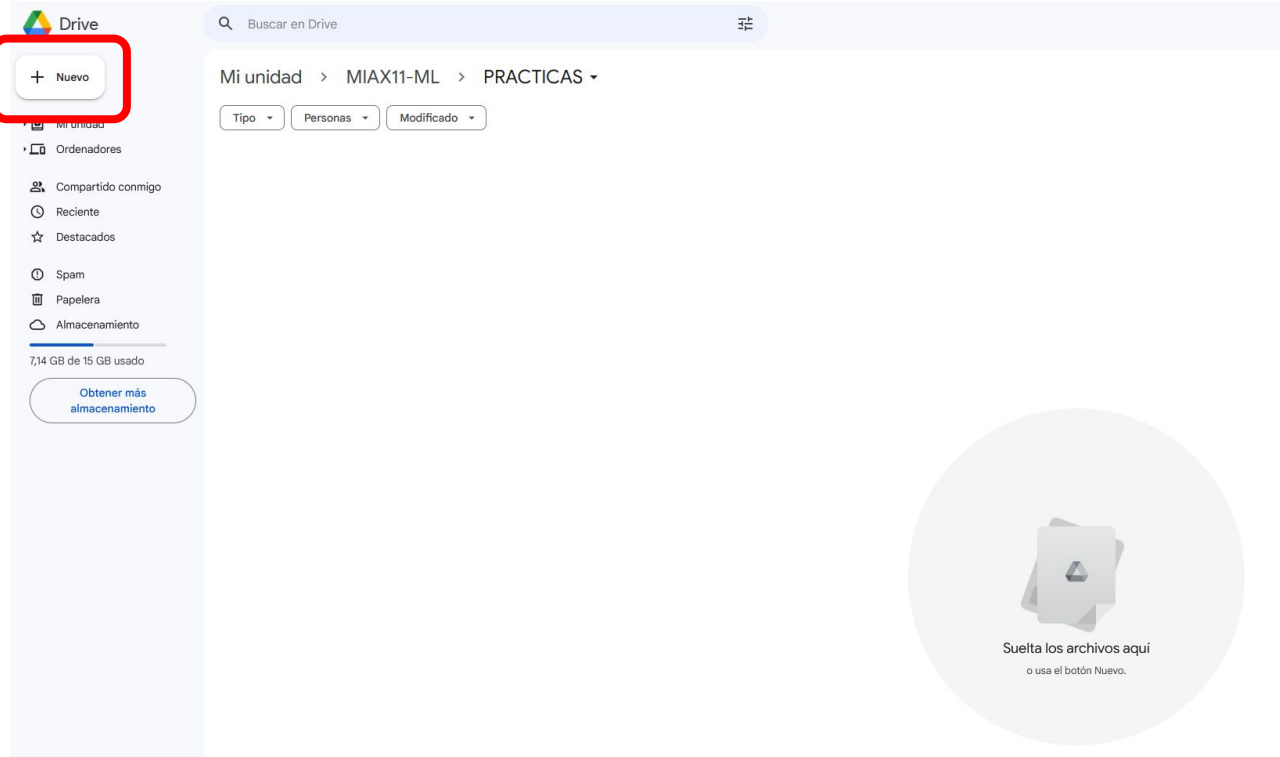
Google Colaboratory

- Plataforma en línea de Google para ejecutar código Python
- Diseñada para ser una herramienta colaborativa que permite compartir proyectos de ML en un entorno basado en el navegador.
- Tienes acceso a potentes recursos en la nube: GPUs y TPUs para Deep Learning
- No requiere instalación más allá de instalar la herramienta en Google Drive
- Se basa en los cuadernos Jupyter



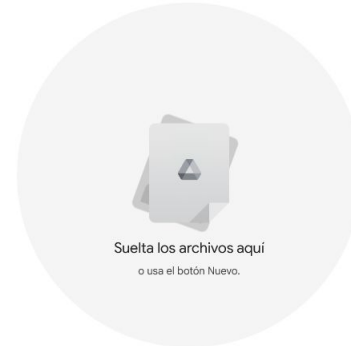
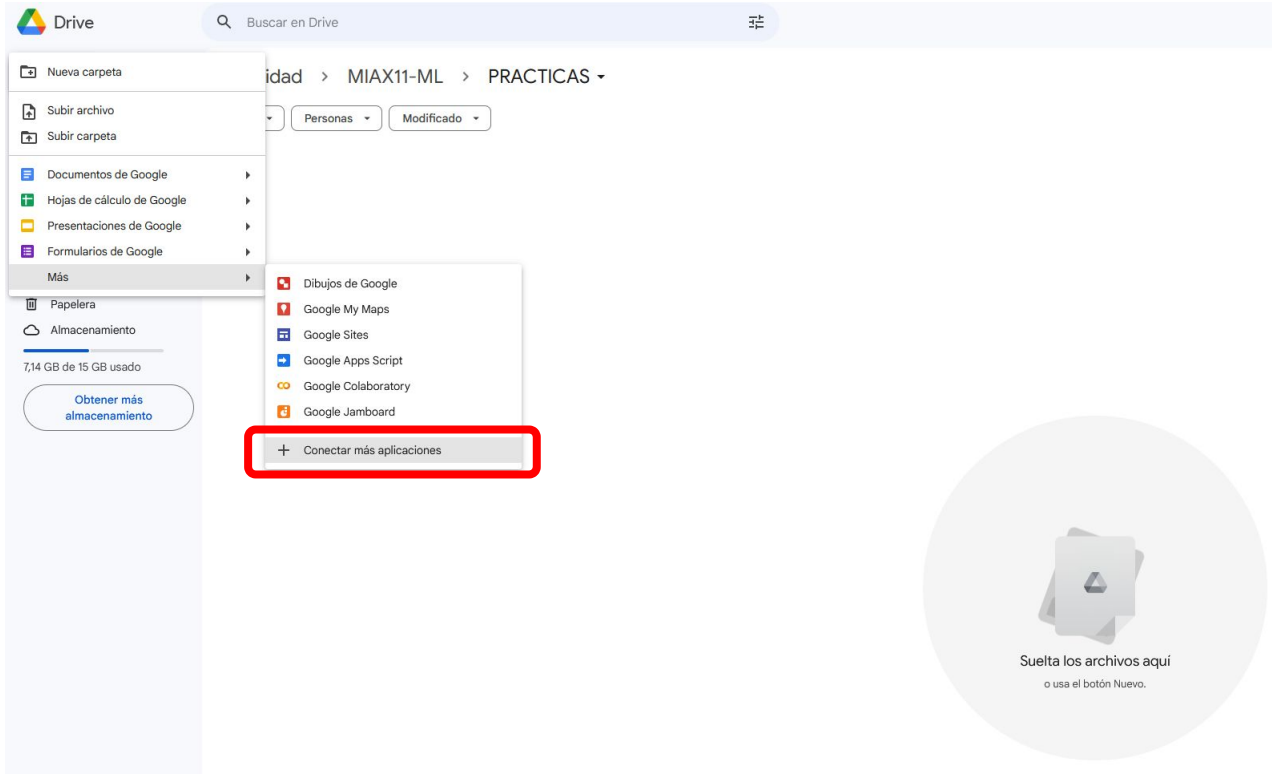
Google Colaboratory

- ¿Cómo instalarlo?



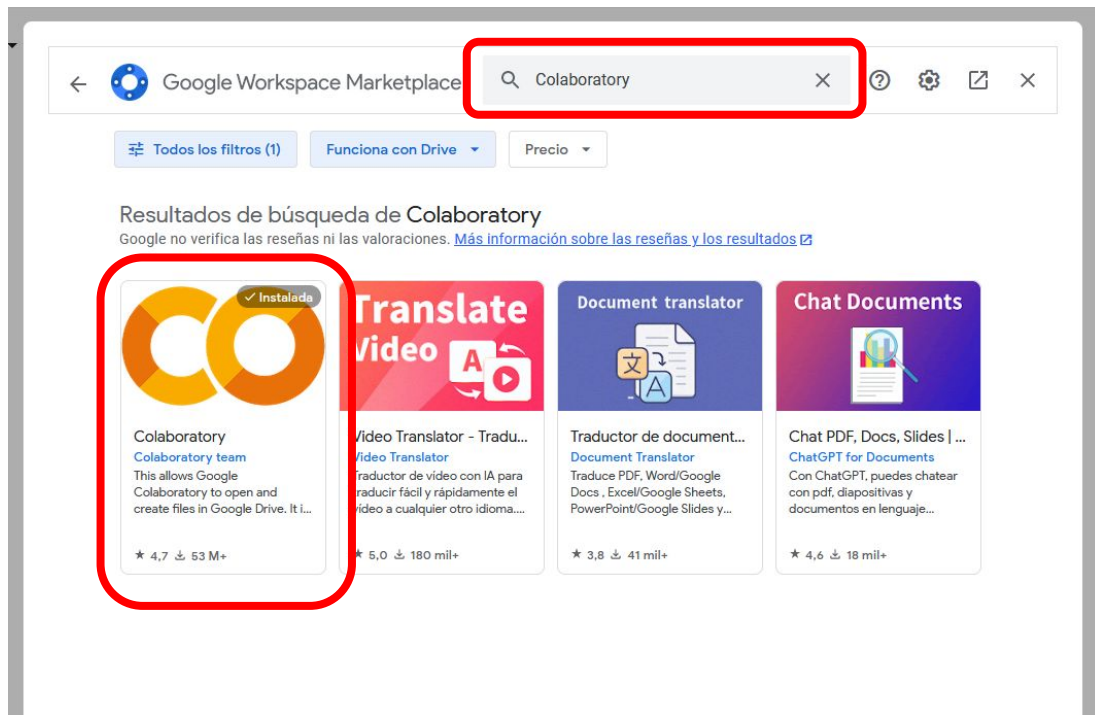
Google Colaboratory

- ¿Cómo instalarlo?



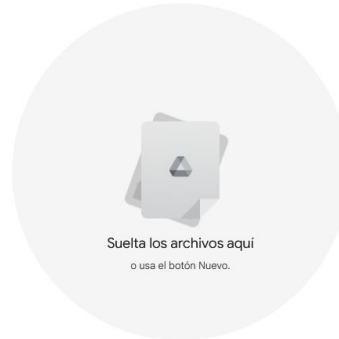
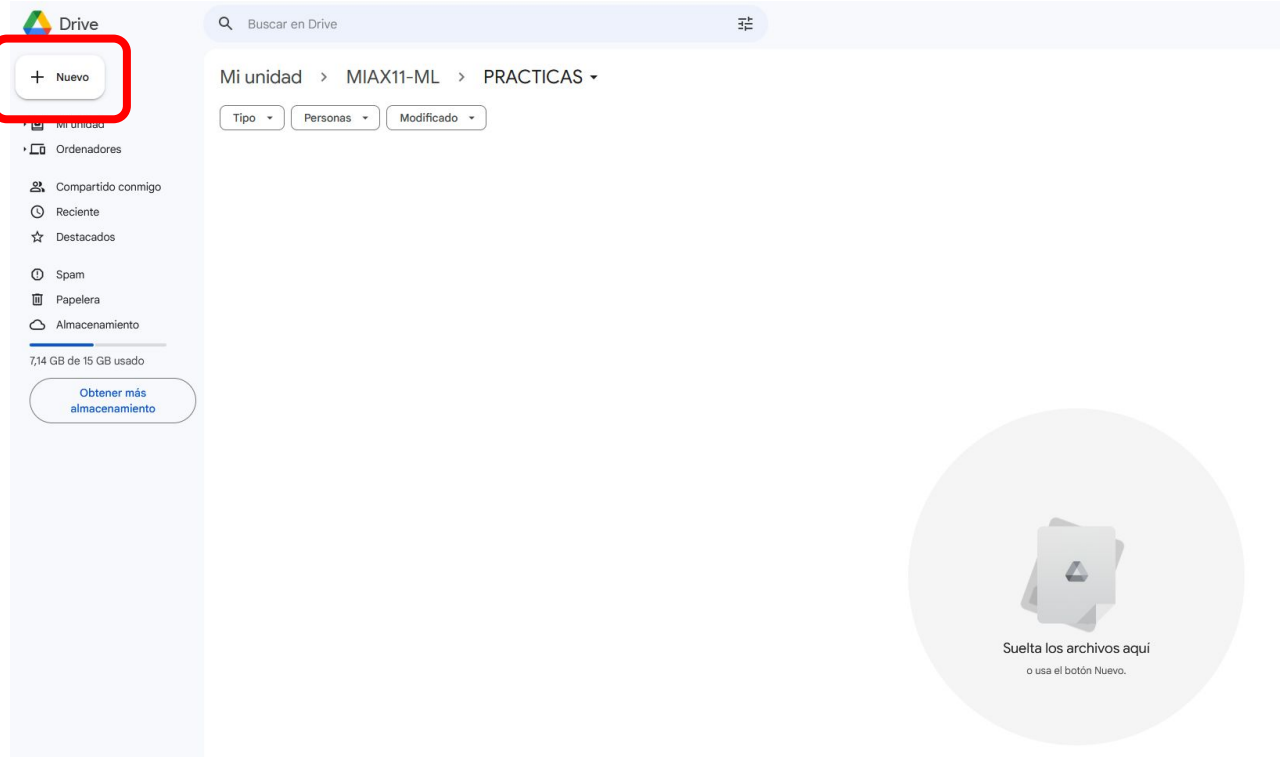
Google Colaboratory

- ¿Cómo instalarlo?



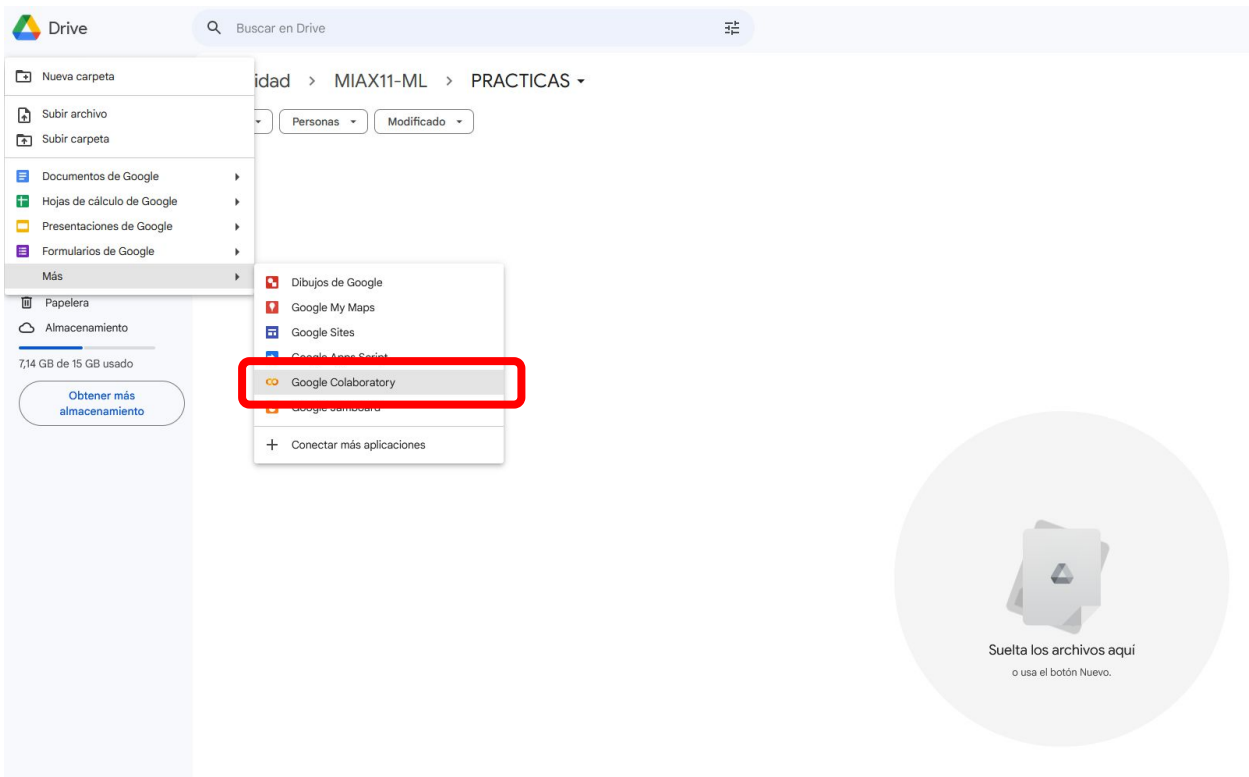
Google Colaboratory

- ¿Cómo crear un nuevo notebook?



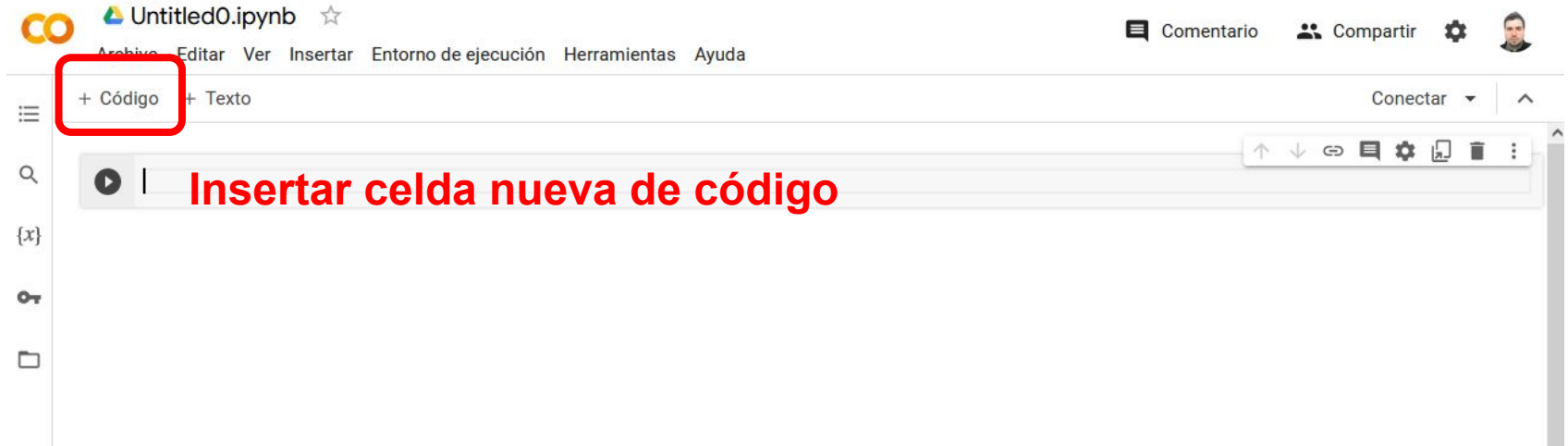
Google Colaboratory

- ¿Cómo crear un nuevo notebook?



Google Colaboratory

- ¿Cómo es Google Colaboratory?



Google Colaboratory

- ¿Cómo es Google Colaboratory?



The screenshot displays the Google Colaboratory web interface. At the top, the Google Colaboratory logo is visible on the left, and navigation links for 'Comentario', 'Compartir', and user settings are on the right. The main workspace shows two code cells. The first cell, labeled '[2]', contains Python code that imports numpy and prints a random array. The second cell, labeled '[3]', contains the code `print(x+5)` and shows the output '15'. A red text overlay is positioned over the second cell, stating: **Podemos utilizar variables en distintas celdas. Las mantiene**. The interface also includes a top menu bar with options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', 'Ayuda', and 'Guardando...'. On the right side of the workspace, there are indicators for RAM and Disco usage.

```
[2] # Aquí podemos añadir el código en python que queramos

import numpy as np

print(np.random.randn(10))

x = 10

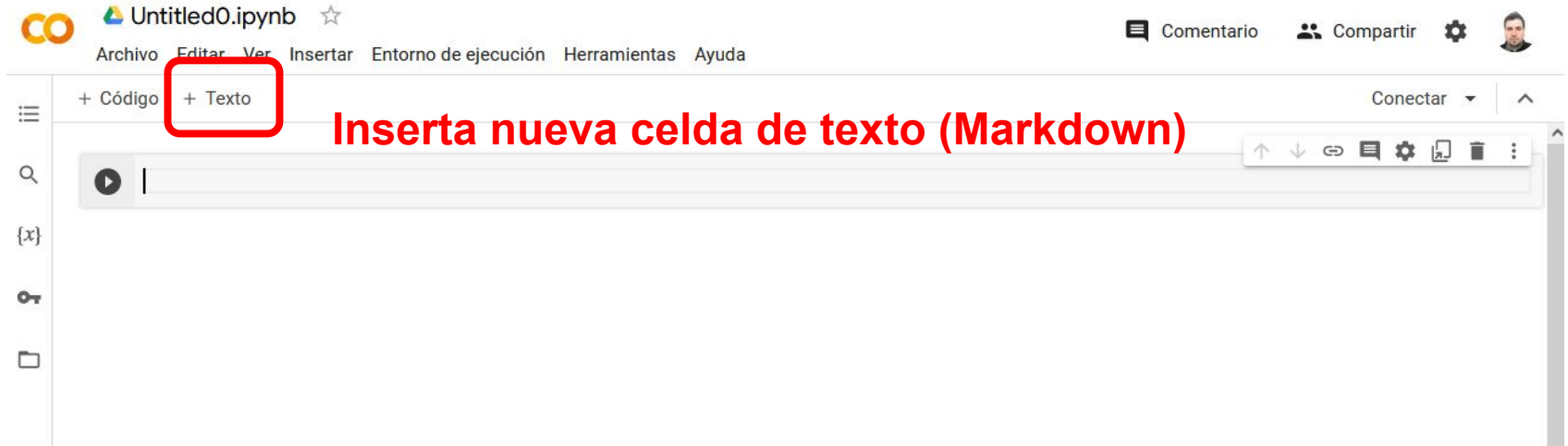
[ 2.29957827  1.28524583  0.08256021  1.20581105 -0.17149717 -0.78319253
 -0.7800814   0.35801726  0.44478018 -1.43448062]

[3] print(x+5)

15
```

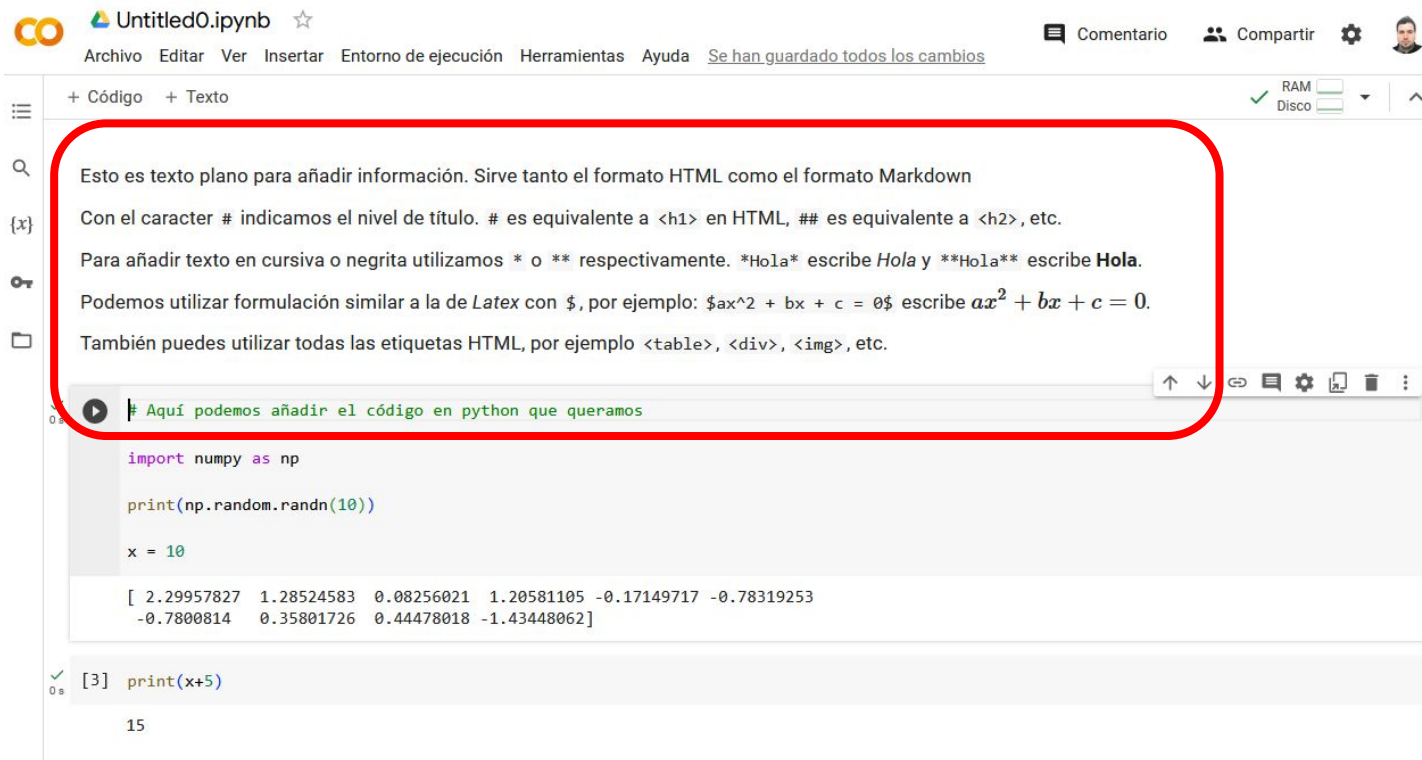
Google Colaboratory

- ¿Cómo es Google Colaboratory?



Google Colaboratory

- ¿Cómo es Google Colaboratory?



co Untitled0.ipynb ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda [Se han guardado todos los cambios](#)

+ Código + Texto

✓ RAM
Disco

Esto es texto plano para añadir información. Sirve tanto el formato HTML como el formato Markdown

Con el caracter # indicamos el nivel de título. # es equivalente a <h1> en HTML, ## es equivalente a <h2>, etc.

Para añadir texto en cursiva o negrita utilizamos * o ** respectivamente. *Hola* escribe *Hola* y **Hola** escribe **Hola**.

Podemos utilizar formulación similar a la de *Latex* con \$, por ejemplo: $ax^2 + bx + c = 0$ escribe $ax^2 + bx + c = 0$.

También puedes utilizar todas las etiquetas HTML, por ejemplo <table>, <div>, , etc.

▶ Aquí podemos añadir el código en python que queramos

```
import numpy as np

print(np.random.randn(10))

x = 10
```

```
[ 2.29957827  1.28524583  0.08256021  1.20581105 -0.17149717 -0.78319253
 -0.7800814   0.35801726  0.44478018 -1.43448062]
```

✓ [3] print(x+5)

15



Introducción - K Vecinos Próximos (KNN)

- Vamos al notebook para probar el KNN
1. Descargad el notebook del material de clase que os damos.
 2. Subid el notebook a vuestro Drive
 3. Abrirlo con Google Colaboratory:

*El notebook es: **2_1_knn.ipynb***