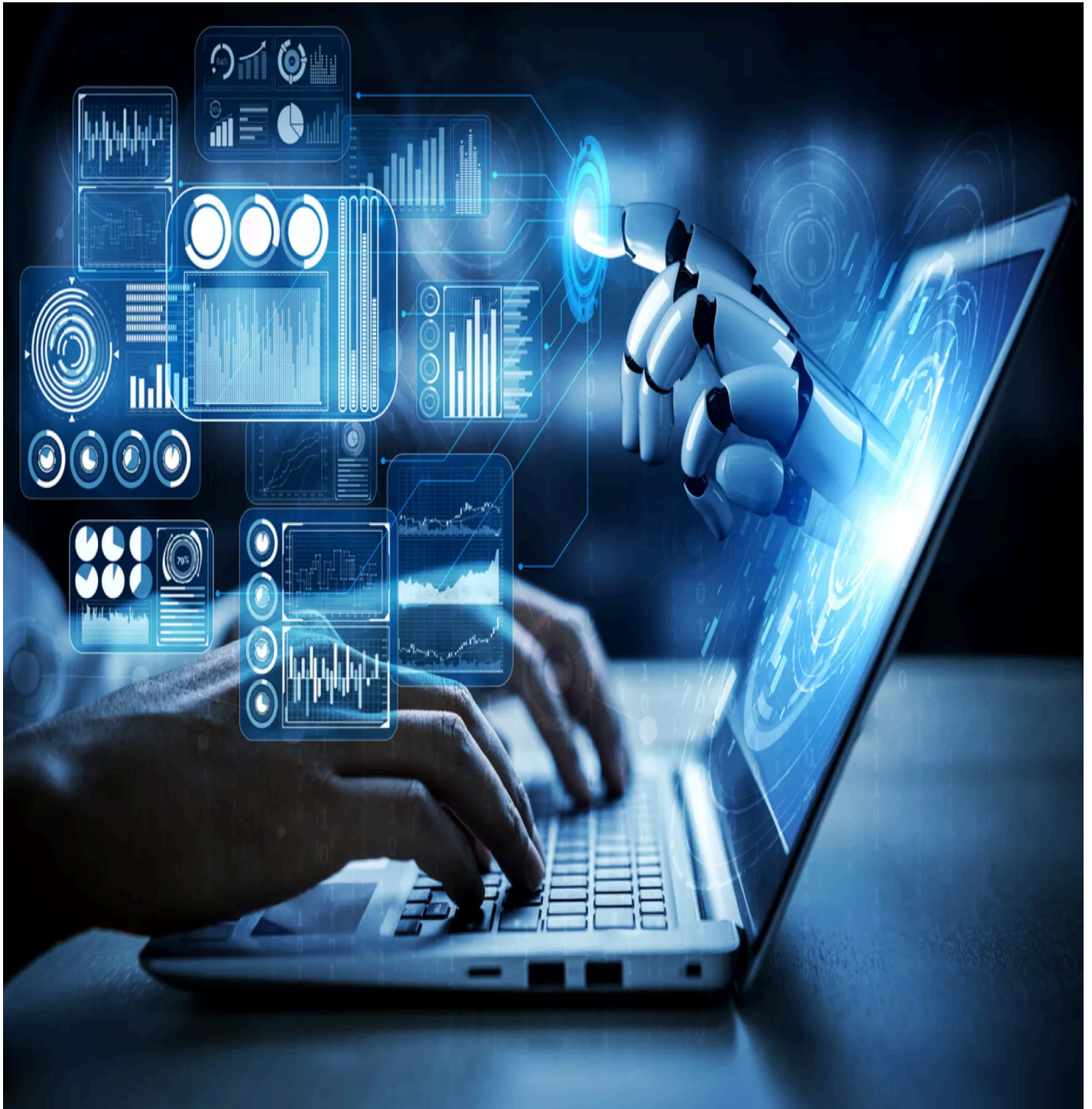


Programación de Inteligencia Artificial



Nombre: Victoria Jiménez Martín

Módulo: Programación de Inteligencia Artificial

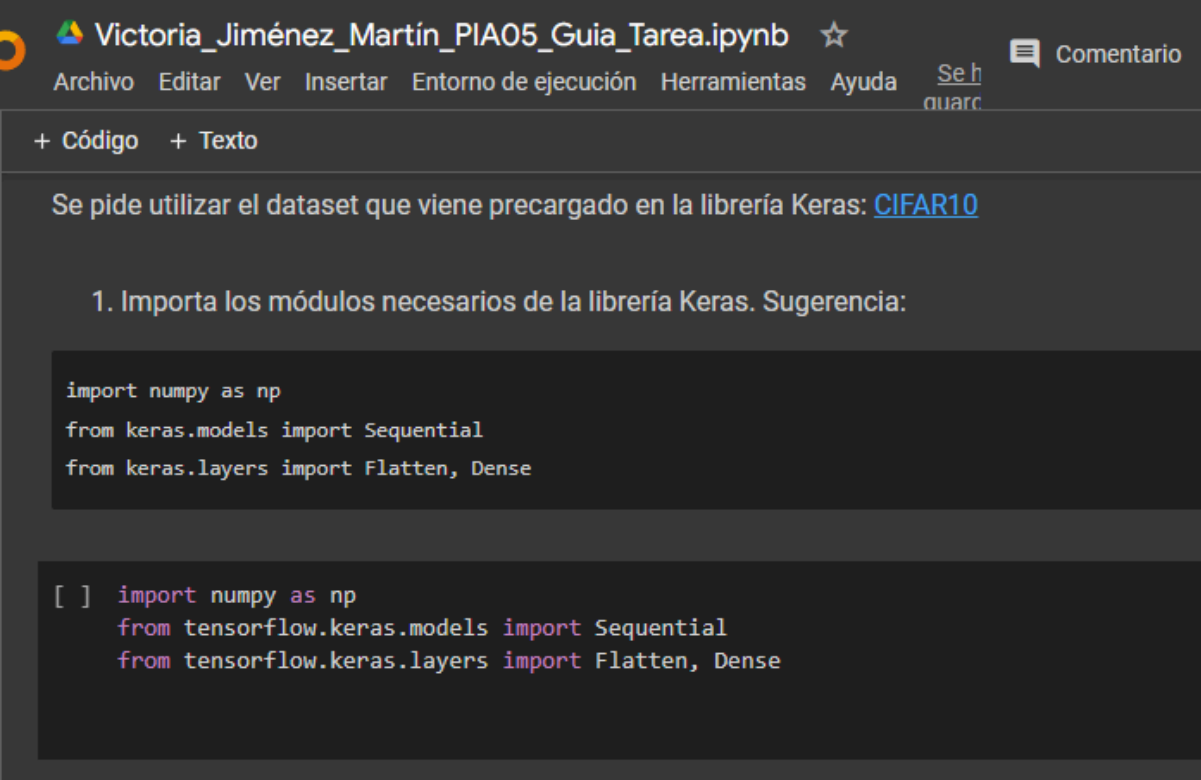
Curso: Especialización de Inteligencia Artificial y Big Data

Índice

Apartado 1: Carga y explora el dataset CIFAR10	3
Apartado 2: Importa el dataset CIFAR10 de Keras, en un conjunto de datos de entrenamiento y un conjunto de datos para test.	4
Apartado 3: Explora los datos.	5
Apartado 4: Crea el modelo.	6
Apartado 5: Entrena el modelo.	7
Apartado 6: Mejora el modelo.	8
○ ¿Has conseguido mejorar la precisión? haz varias pruebas y quédate con el modelo que mejores resultados da.	8
Apartado 7: Evalúa el nuevo modelo.	9
○ ¿Es muy diferente a la precisión alcanzada en el entrenamiento?	9
Apartado 8: Visualiza las predicciones.	10

Apartado 1: Carga y explora el dataset CIFAR10

- Inicia un nuevo notebook, preferiblemente en Google Colab. Para guiarte en el proceso, puedes utilizar este [cuaderno-guía](#) con algunas sugerencias de fragmentos de código indicados en las celdas de texto, pero tendrás que escribir el código en la celda de código correspondiente y ejecutarlo.
- Importa la librerías **Numpy**.
- Importa los módulos necesarios para construir una red neuronal profunda: Sequential, Dense y Flatten.



Victoria_Jiménez_Martín_PIA05_Guía_Tarea.ipynb ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda [Se h](#)
[quarc](#) Comentario

+ Código + Texto

Se pide utilizar el dataset que viene precargado en la librería Keras: [CIFAR10](#)

1. Importa los módulos necesarios de la librería Keras. Sugerencia:

```
import numpy as np
from keras.models import Sequential
from keras.layers import Flatten, Dense
```

```
[ ] import numpy as np
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Flatten, Dense
```

Apartado 2: Importa el dataset CIFAR10 de Keras, en un conjunto de datos de entrenamiento y un conjunto de datos para test.

- Consulta la [documentación de Keras relativa a este dataset](#) para entender cómo están organizados los datos y saber importarlos.



The screenshot shows a Jupyter Notebook titled "Victoria_Jiménez_Martín_PIA05_Guía_Tarea.ipynb". The interface includes a top menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and "Se h...". Below the menu, there are tabs for "+ Código" and "+ Texto", and a "Conectar" button. The notebook content displays a text cell with the instruction: "2. Importa el dataset CIFAR10 de Keras, en un conjunto de datos de entrenamiento y un conjunto de datos para test. Sugerencia:". Below this is a code cell containing the following Python code:

```
from keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

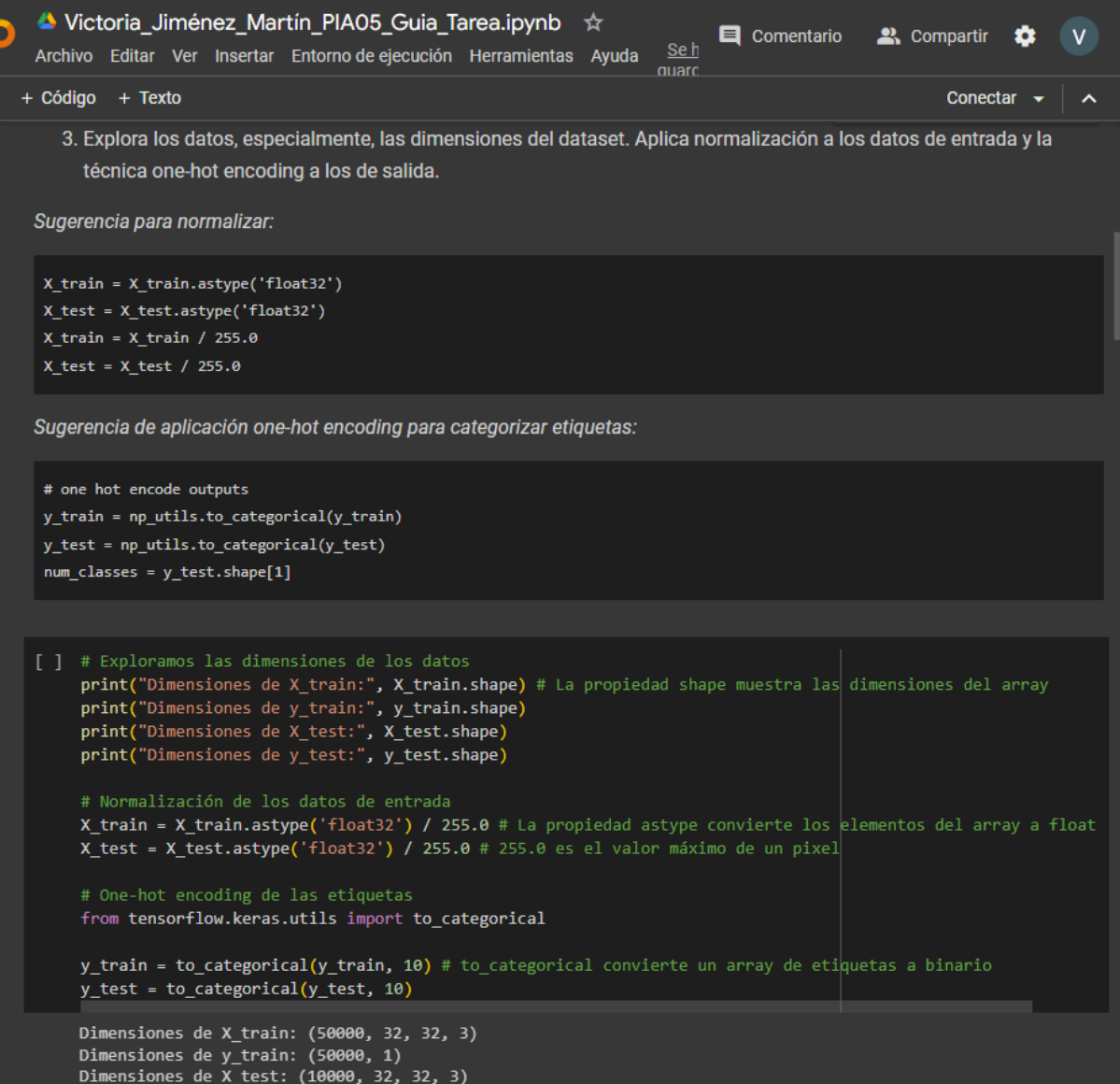
Below the code cell, there is a toolbar with icons for undo, redo, copy, paste, and other actions. The code cell is followed by another code cell that contains the same code, but with a comment added:

```
from tensorflow.keras.datasets import cifar10

# Cargamos los datos
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Apartado 3: Explora los datos.

- Explora los datos, especialmente, las dimensiones del dataset.
- Aplica normalización a los datos de entrada.
- Aplica la técnica one-hot encoding al conjunto de datos de salida.
- En general, aplica las funciones necesarias para entender cómo son los datos para poder crear el modelo de forma adecuada y entender también los resultados del entrenamiento.



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the notebook title "Victoria_Jiménez_Martín_PIA05_Guia_Tarea.ipynb" and various icons for file operations, execution, and sharing. Below the top bar, there are tabs for "Código" and "Texto". The main content area displays a numbered instruction: "3. Explora los datos, especialmente, las dimensiones del dataset. Aplica normalización a los datos de entrada y la técnica one-hot encoding a los de salida." This is followed by two sections of code. The first section, titled "Sugerencia para normalizar:", contains code to convert training and testing data to float32 and normalize them by dividing by 255.0. The second section, titled "Sugerencia de aplicación one-hot encoding para categorizar etiquetas:", contains code to use np_utils.to_categorical for training and testing labels, and tensorflow.keras.utils.to_categorical for the training labels. A large code block follows, which includes print statements to check the dimensions of the data arrays, comments explaining the shape and astype properties, and code for one-hot encoding the training labels. At the bottom, the output of the dimension checks is displayed.

```
Victoria_Jiménez_Martín_PIA05_Guia_Tarea.ipynb ☆
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se h
+ Código + Texto Conectar ^

3. Explora los datos, especialmente, las dimensiones del dataset. Aplica normalización a los datos de entrada y la
técnica one-hot encoding a los de salida.

Sugerencia para normalizar:

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0

Sugerencia de aplicación one-hot encoding para categorizar etiquetas:

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]

[ ] # Exploramos las dimensiones de los datos
print("Dimensiones de X_train:", X_train.shape) # La propiedad shape muestra las dimensiones del array
print("Dimensiones de y_train:", y_train.shape)
print("Dimensiones de X_test:", X_test.shape)
print("Dimensiones de y_test:", y_test.shape)

# Normalización de los datos de entrada
X_train = X_train.astype('float32') / 255.0 # La propiedad astype convierte los elementos del array a float
X_test = X_test.astype('float32') / 255.0 # 255.0 es el valor máximo de un pixel

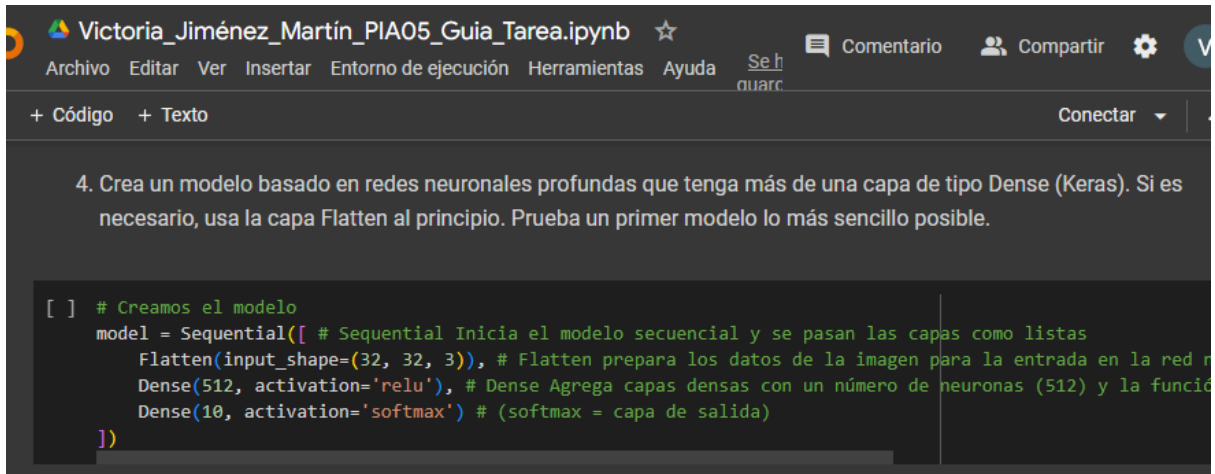
# One-hot encoding de las etiquetas
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train, 10) # to_categorical convierte un array de etiquetas a binario
y_test = to_categorical(y_test, 10)

Dimensiones de X_train: (50000, 32, 32, 3)
Dimensiones de y_train: (50000, 1)
Dimensiones de X_test: (10000, 32, 32, 3)
```

Apartado 4: Crea el modelo.

- Genera un modelo con la clase Sequential.
- Añade el menor número de capas posible, utilizando las clases Dense y Flatten.

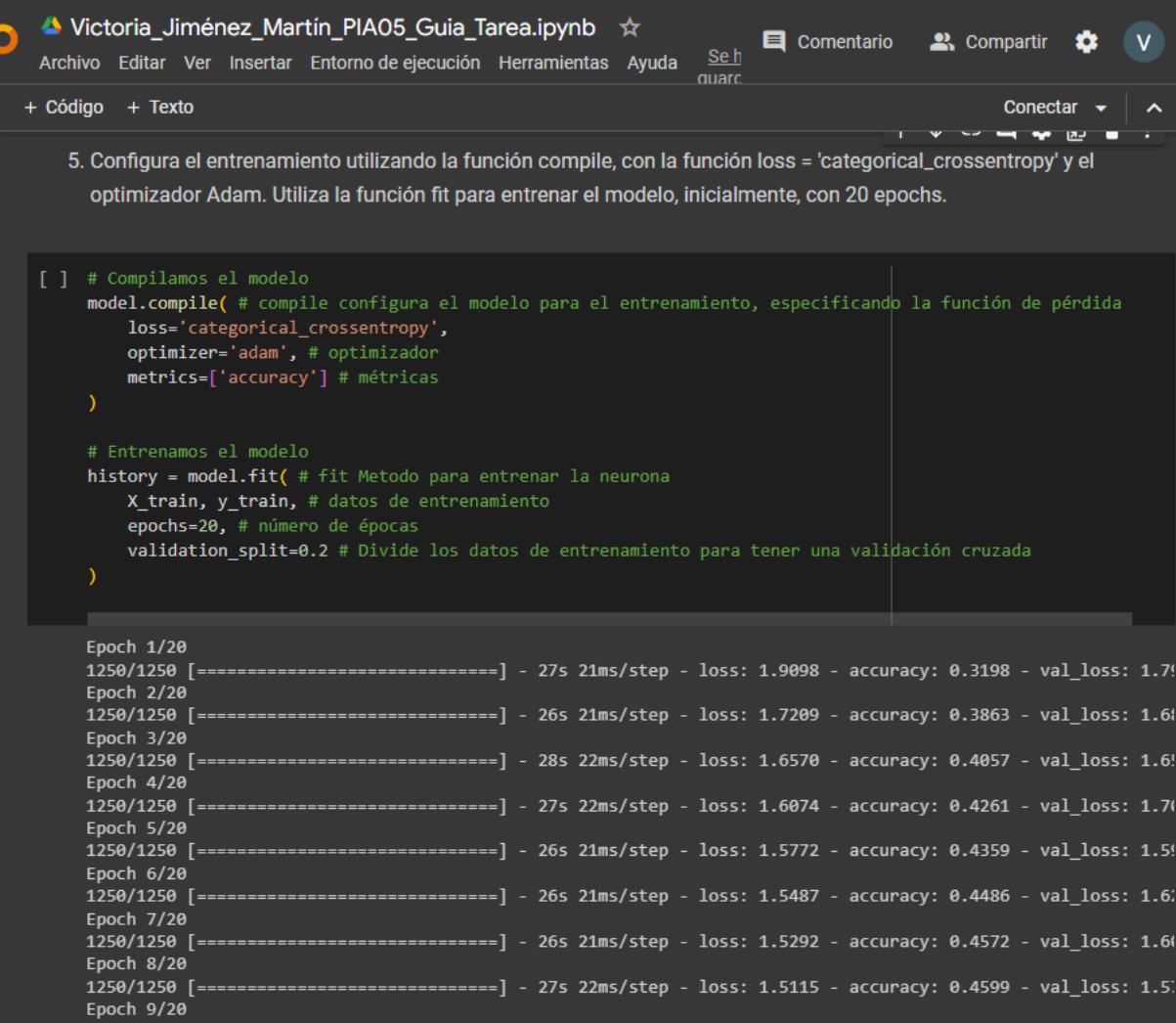


The screenshot shows a Jupyter Notebook interface. The top bar includes the file name 'Victoria_Jiménez_Martín_PIA05_Guia_Tarea.ipynb' and various icons for file operations, comments, sharing, and settings. Below the top bar is a menu bar with options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', 'Ayuda', and 'Se guardar'. The main area of the notebook contains a text cell with the instruction: '4. Crea un modelo basado en redes neuronales profundas que tenga más de una capa de tipo Dense (Keras). Si es necesario, usa la capa Flatten al principio. Prueba un primer modelo lo más sencillo posible.' Below this is a code cell with the following Python code:

```
[ ] # Creamos el modelo
model = Sequential([ # Sequential Inicia el modelo secuencial y se pasan las capas como listas
    Flatten(input_shape=(32, 32, 3)), # Flatten prepara los datos de la imagen para la entrada en la red n
    Dense(512, activation='relu'), # Dense Agrega capas densas con un número de neuronas (512) y la funci
    Dense(10, activation='softmax') # (softmax = capa de salida)
])
```

Apartado 5: Entrena el modelo.

- Configura el modo de entrenamiento con el método compile.
- Utiliza la función loss = 'categorical_crossentropy'.
- Selecciona el optimizador Adam.
- Utiliza la función fit para entrenar el modelo, con un máximo de 20 epochs.



The screenshot shows a Jupyter Notebook titled "Victoria_Jiménez_Martin_PIA05_Guia_Tarea.ipynb". The interface includes a top bar with navigation options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and "Se guardar". Below the top bar, there are tabs for "+ Código" and "+ Texto". The main area contains a code cell with the following Python code:

```
[ ] # Compilamos el modelo
model.compile( # compile configura el modelo para el entrenamiento, especificando la función de pérdida
    loss='categorical_crossentropy',
    optimizer='adam', # optimizador
    metrics=['accuracy'] # métricas
)

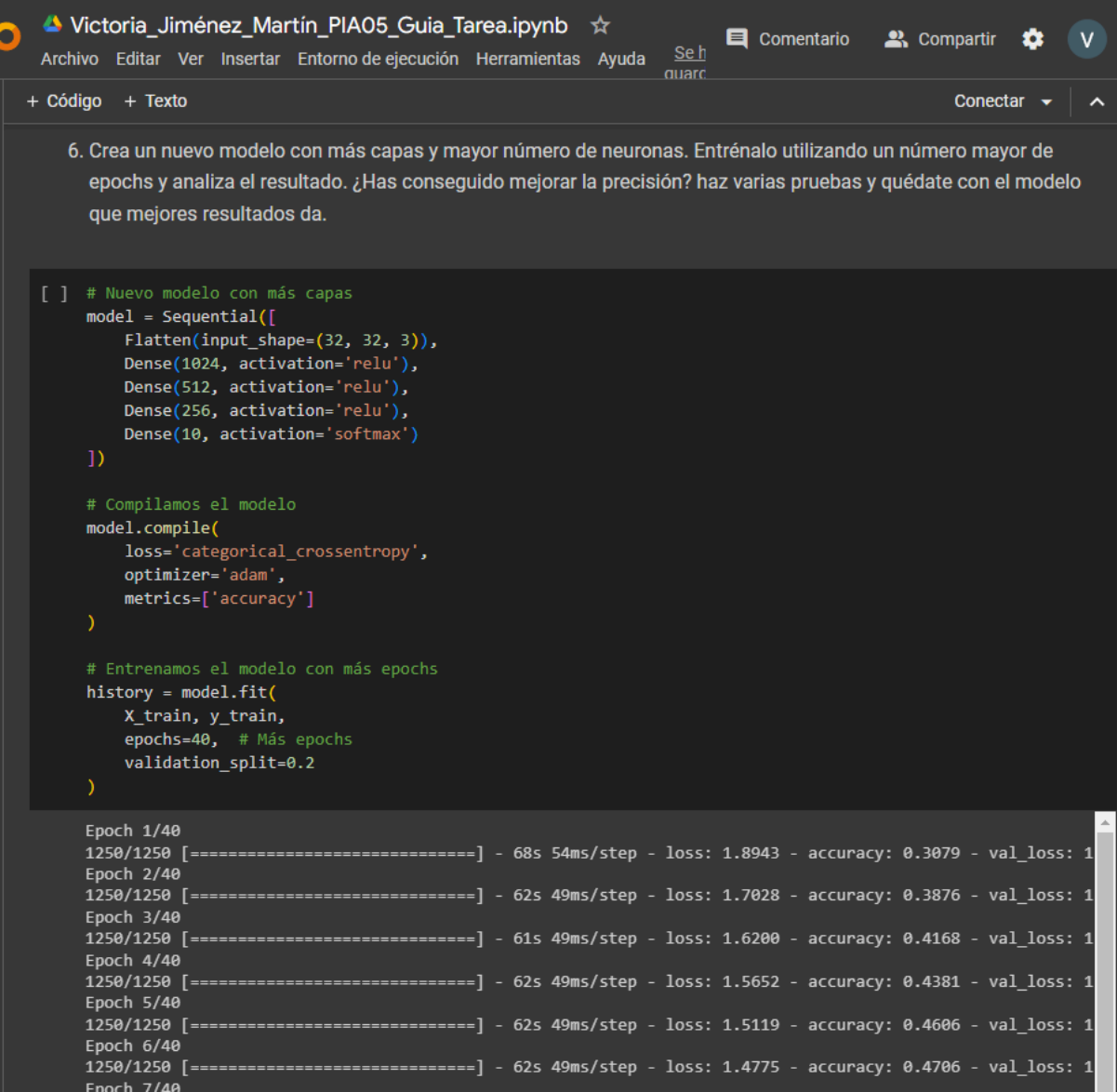
# Entrenamos el modelo
history = model.fit( # fit Metodo para entrenar la neurona
    X_train, y_train, # datos de entrenamiento
    epochs=20, # número de épocas
    validation_split=0.2 # Divide los datos de entrenamiento para tener una validación cruzada
)
```

Below the code cell, the output of the training process is displayed, showing the progress for each epoch from 1 to 9. The output includes the number of samples processed (1250/1250), a progress bar, and the training and validation loss and accuracy for each epoch.

```
Epoch 1/20
1250/1250 [=====] - 27s 21ms/step - loss: 1.9098 - accuracy: 0.3198 - val_loss: 1.71
Epoch 2/20
1250/1250 [=====] - 26s 21ms/step - loss: 1.7209 - accuracy: 0.3863 - val_loss: 1.61
Epoch 3/20
1250/1250 [=====] - 28s 22ms/step - loss: 1.6570 - accuracy: 0.4057 - val_loss: 1.61
Epoch 4/20
1250/1250 [=====] - 27s 22ms/step - loss: 1.6074 - accuracy: 0.4261 - val_loss: 1.71
Epoch 5/20
1250/1250 [=====] - 26s 21ms/step - loss: 1.5772 - accuracy: 0.4359 - val_loss: 1.51
Epoch 6/20
1250/1250 [=====] - 26s 21ms/step - loss: 1.5487 - accuracy: 0.4486 - val_loss: 1.61
Epoch 7/20
1250/1250 [=====] - 26s 21ms/step - loss: 1.5292 - accuracy: 0.4572 - val_loss: 1.61
Epoch 8/20
1250/1250 [=====] - 27s 22ms/step - loss: 1.5115 - accuracy: 0.4599 - val_loss: 1.51
Epoch 9/20
1250/1250 [=====] - 27s 22ms/step - loss: 1.4958 - accuracy: 0.4653 - val_loss: 1.4958
```

Apartado 6: Mejora el modelo.

- Crea un nuevo modelo con más capas y mayor número de neuronas.
- Entrénalo utilizando un número mayor de epochs y analiza el resultado.



The screenshot shows a Jupyter Notebook titled "Victoria_Jiménez_Martin_PIA05_Guia_Tarea.ipynb". The notebook contains a code cell with the following Python code:

```
[ ] # Nuevo modelo con más capas
model = Sequential([
    Flatten(input_shape=(32, 32, 3)),
    Dense(1024, activation='relu'),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(10, activation='softmax')
])

# Compilamos el modelo
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

# Entrenamos el modelo con más epochs
history = model.fit(
    X_train, y_train,
    epochs=40, # Más epochs
    validation_split=0.2
)
```

Below the code, the output of the training process is displayed, showing progress for epochs 1/40 through 7/40. The output includes training steps (1250/1250), loss, accuracy, and validation loss for each epoch.

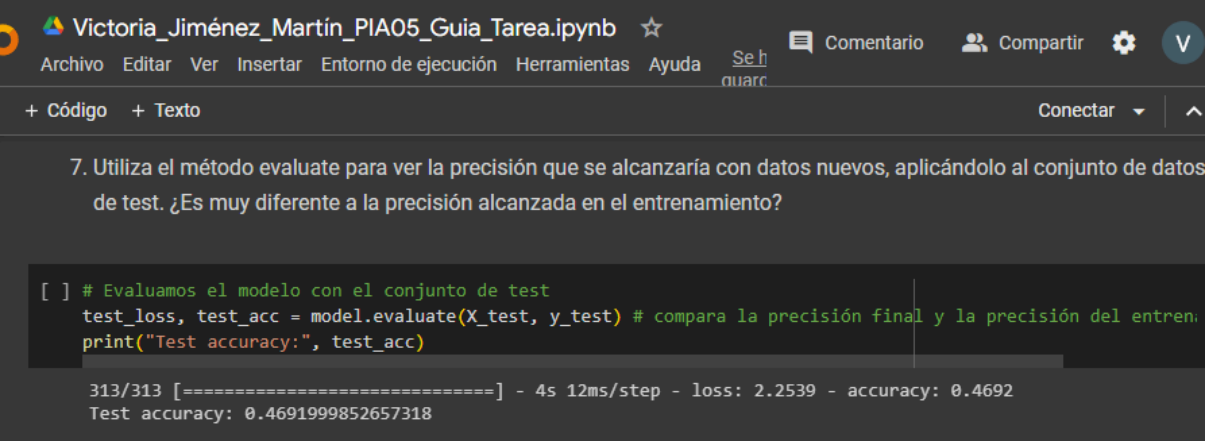
Epoch	Steps	Loss	Accuracy	Val_Loss
Epoch 1/40	1250/1250	1.8943	0.3079	1.7028
Epoch 2/40	1250/1250	1.7028	0.3876	1.6200
Epoch 3/40	1250/1250	1.6200	0.4168	1.5652
Epoch 4/40	1250/1250	1.5652	0.4381	1.5119
Epoch 5/40	1250/1250	1.5119	0.4606	1.4775
Epoch 6/40	1250/1250	1.4775	0.4706	1.4775
Epoch 7/40	1250/1250	1.4775	0.4706	1.4775

- ¿Has conseguido mejorar la precisión? haz varias pruebas y quédate con el modelo que mejores resultados da.

El modelo mejorado muestra una mejora de precisión en comparación con el modelo inicial. Esto indica que aumenta la complejidad del modelo y el número de epochs ha tenido un efecto positivo en la capacidad del modelo para capturar patrones de datos, eso sí, con el costo de un mayor sobreajuste, debido a la brecha entre la precisión de entrenamiento y la precisión de validación hacia las últimas epochs.

Apartado 7: Evalúa el nuevo modelo.

- Utiliza el método `evaluate` para ver la precisión que se alcanzaría con datos nuevos, aplicándolo al conjunto de datos de test.



The screenshot shows a Jupyter Notebook interface. The top bar includes the file name 'Victoria_Jiménez_Martín_PIA05_Guia_Tarea.ipynb' and various icons for saving, commenting, and sharing. Below the top bar is a menu with options like 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', 'Ayuda', and 'Se h...'. The main area contains a text cell with the following text: '7. Utiliza el método `evaluate` para ver la precisión que se alcanzaría con datos nuevos, aplicándolo al conjunto de datos de test. ¿Es muy diferente a la precisión alcanzada en el entrenamiento?'. Below the text cell is a code cell with the following code:

```
[ ] # Evaluamos el modelo con el conjunto de test
test_loss, test_acc = model.evaluate(X_test, y_test) # compara la precisión final y la precisión del entren.
print("Test accuracy:", test_acc)
```

 The output of the code cell is:

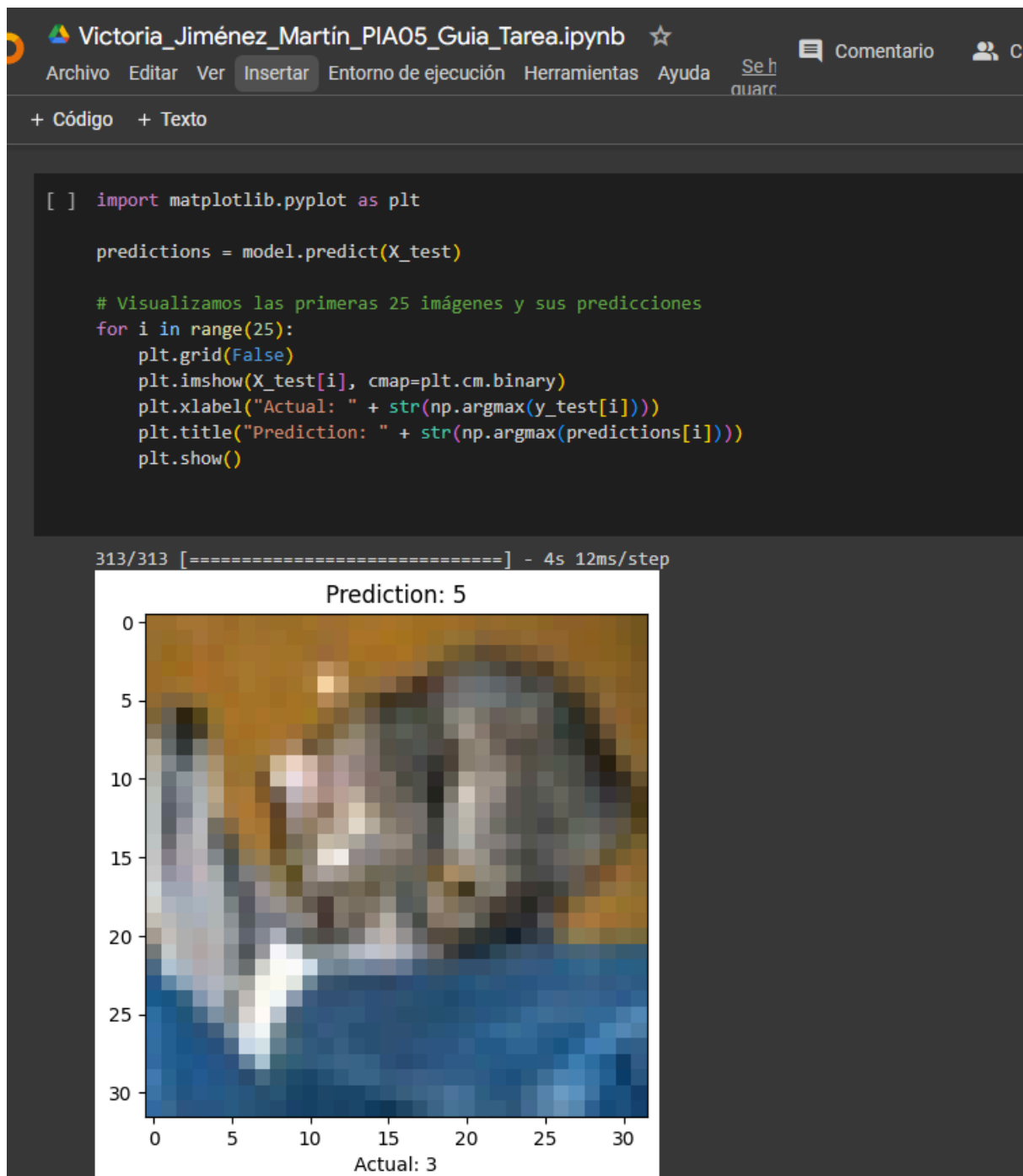
```
313/313 [=====] - 4s 12ms/step - loss: 2.2539 - accuracy: 0.4692
Test accuracy: 0.4691999852657318
```

- ¿Es muy diferente a la precisión alcanzada en el entrenamiento?

Hay una pequeña diferencia entre la precisión de entrenamiento y la precisión en el conjunto de la prueba, lo que indica que el modelo generaliza bien a datos nuevos.

Apartado 8: Visualiza las predicciones.

- Explora de forma visual la precisión que se consigue, representando las primeras 25 imágenes del conjunto de datos de test, y comparando la etiqueta real con la de la predicción.
- En la guía tienes un script sugerido para ayudarte con el código.



<https://colab.research.google.com/drive/1Boxm8iDThWx9Z3xzImAKjsmWV0QRHu40?usp=sharing>

