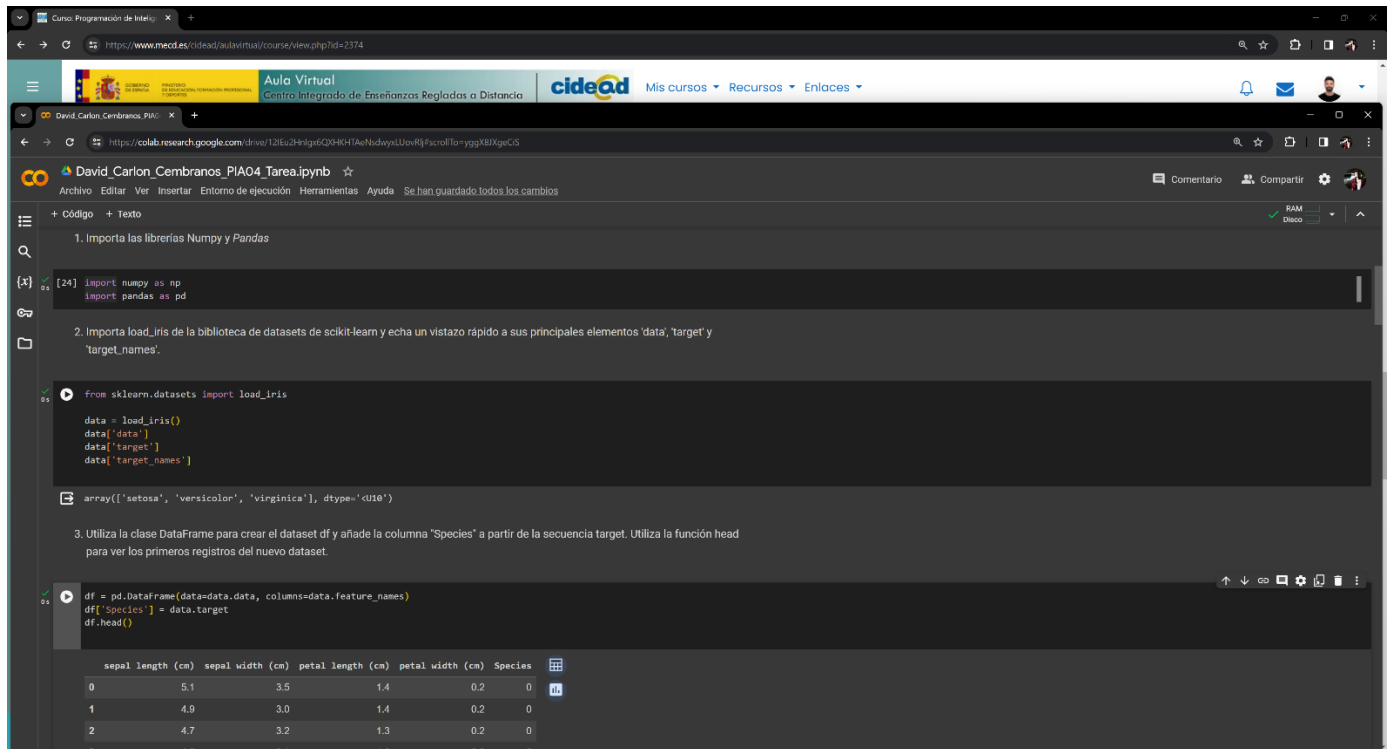


TAREA DE PROGRAMACION DE INTELIGENCIA ARTIFICIAL 04

Apartado 1: Se crea un notebook en Colab, con su título y las celdas de código indicadas de la librería Pandas



The screenshot shows a Google Colab notebook titled "David_Carlon_Cembranos_PIA04_Tarea.ipynb". The notebook is open in a web browser with the URL <https://colab.research.google.com/drive/12Eu2Hnrg6QXKH7AeNbdwytUJovR9fscroITo-yggXBJXgcG5>. The notebook interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". The main area shows three code cells:

1. Importa las librerías Numpy y Pandas
2. Importa load_iris de la biblioteca de datasets de scikit-learn y echa un vistazo rápido a sus principales elementos 'data', 'target' y 'target_names'.
3. Utiliza la clase DataFrame para crear el dataset df y añade la columna 'Species' a partir de la secuencia target. Utiliza la función head para ver los primeros registros del nuevo dataset.

```
[24] import numpy as np
import pandas as pd

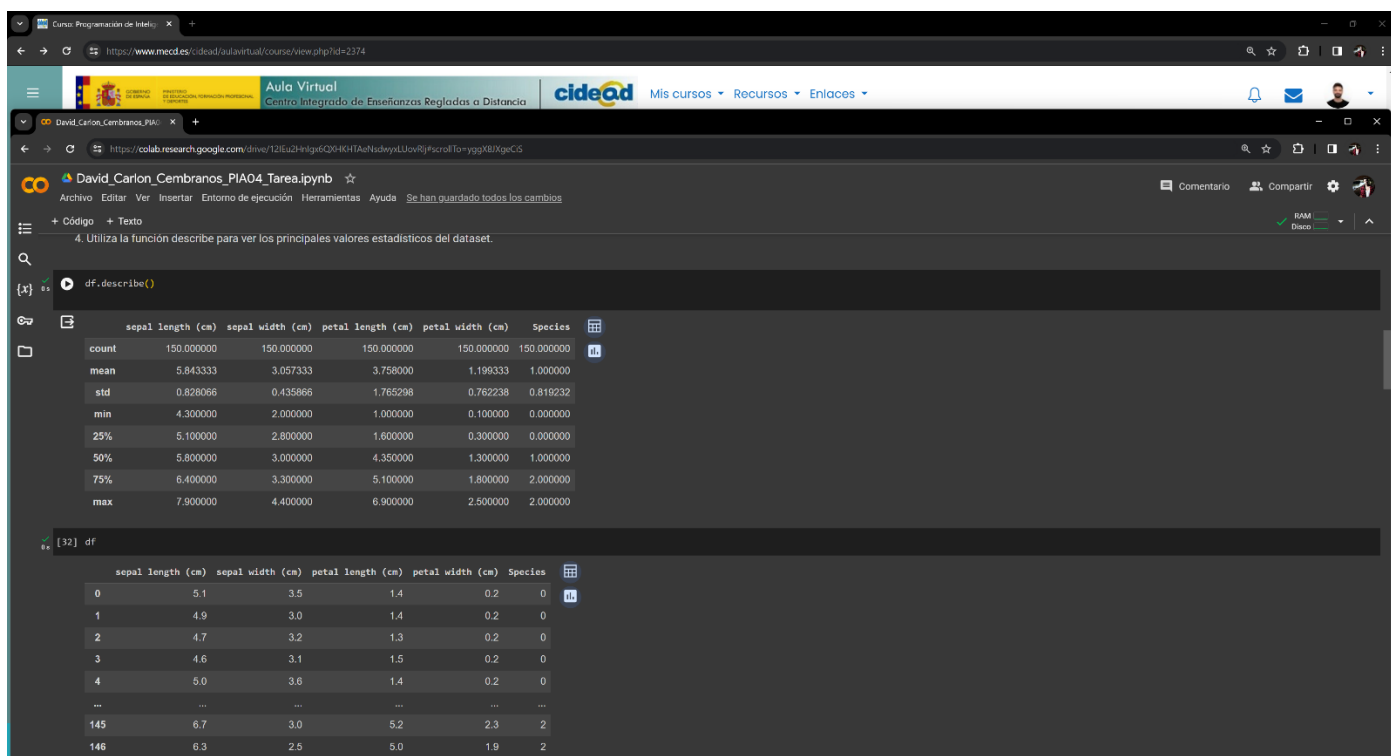
from sklearn.datasets import load_iris

data = load_iris()
data['data']
data['target']
data['target_names']

array(['setosa', 'versicolor', 'virginica'], dtype=object)

df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['Species'] = data.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0



The screenshot shows the same Google Colab notebook, now with a fourth code cell added:

4. Utiliza la función describe para ver los principales valores estadísticos del dataset.

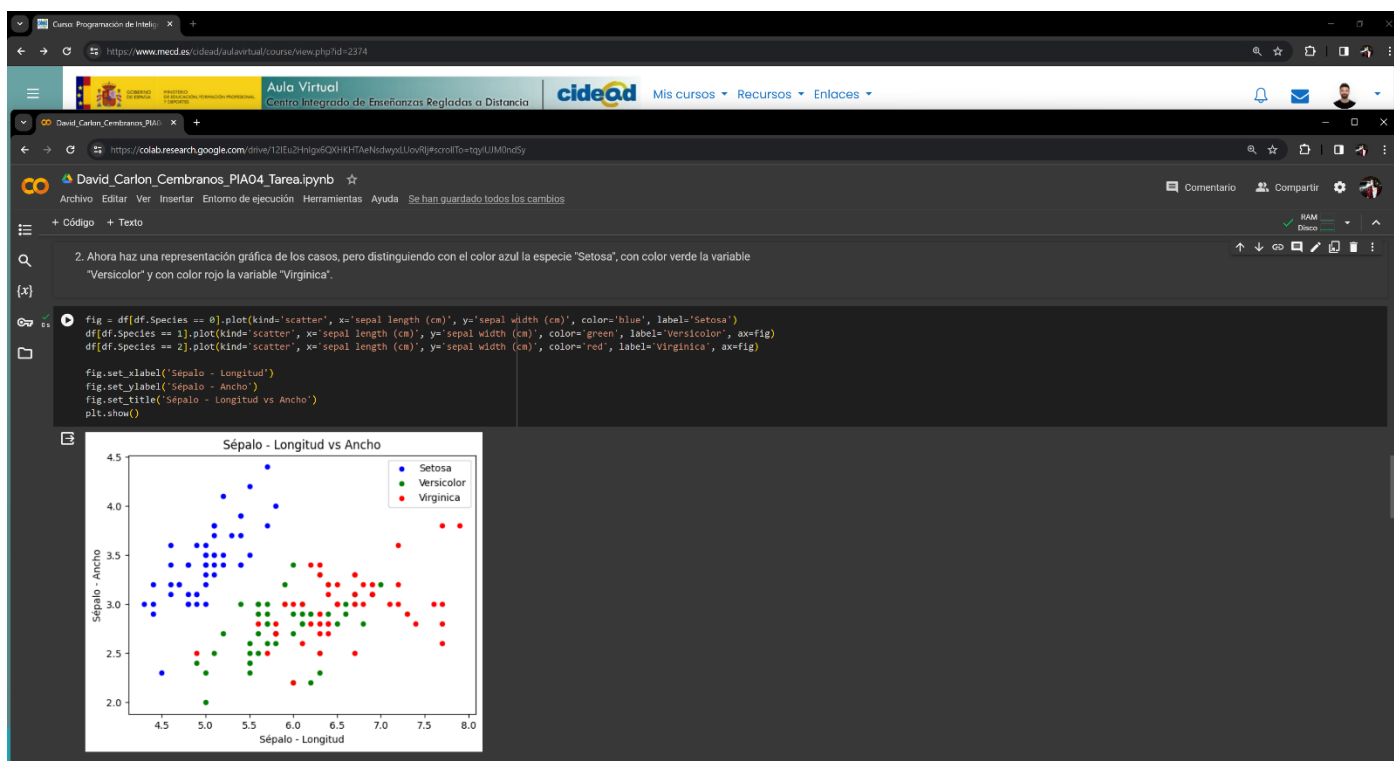
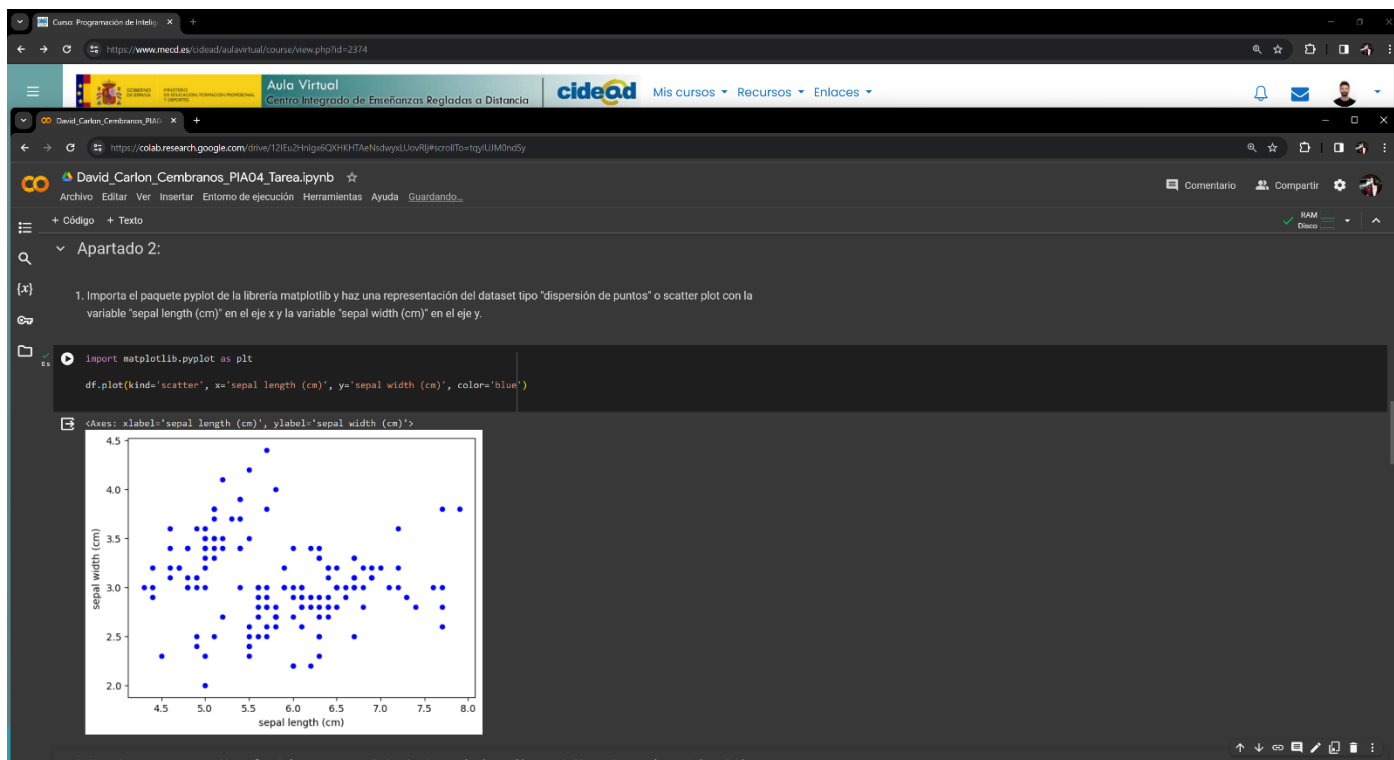
```
df.describe()
```

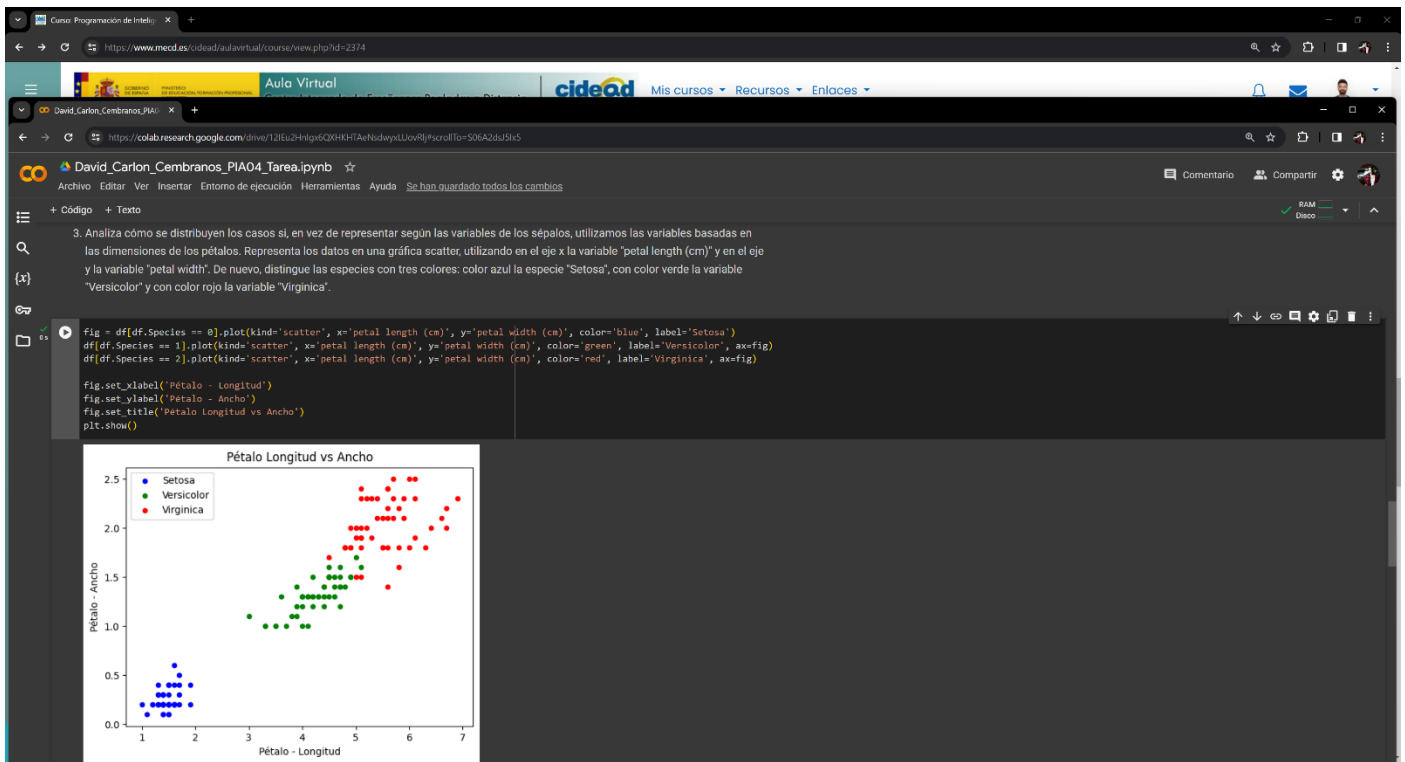
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.818232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

```
[32] df
```

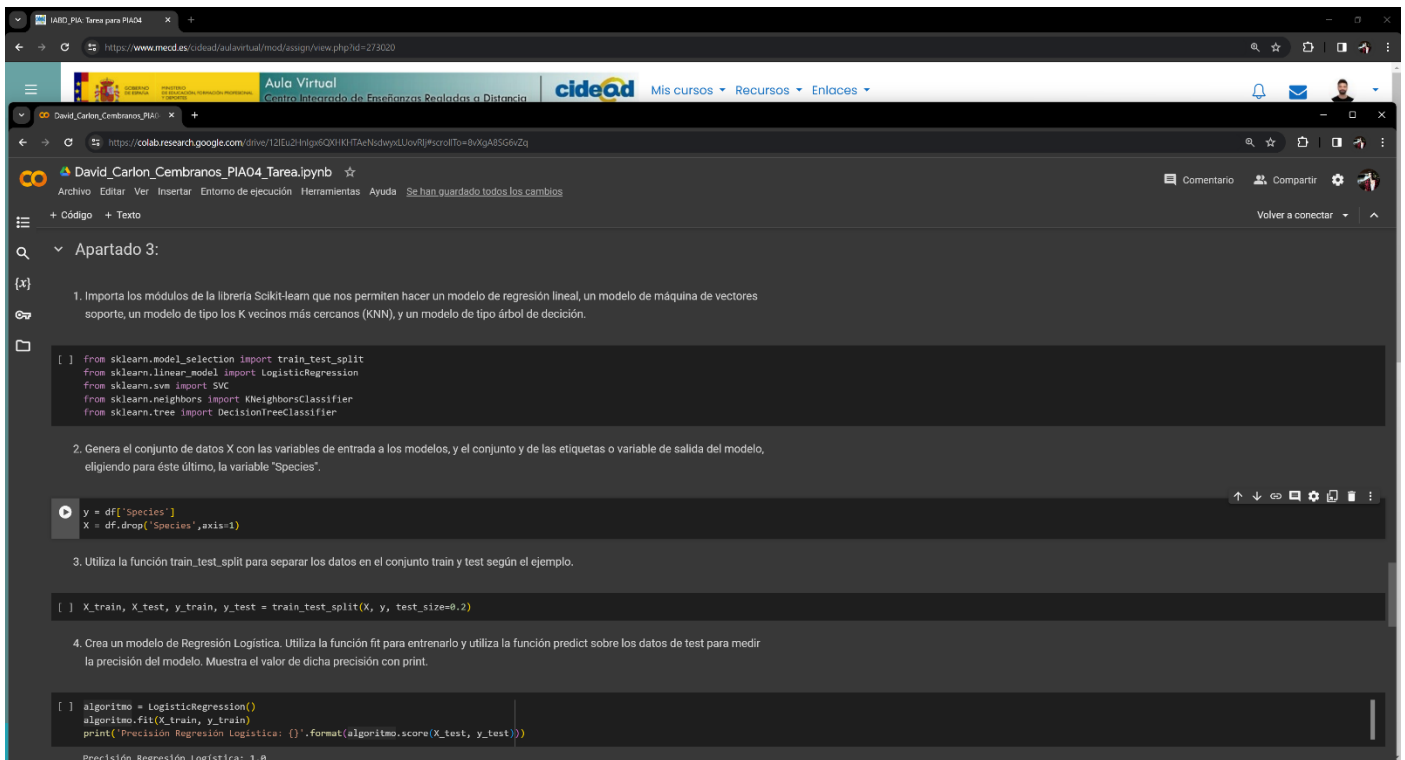
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2

Apartado 2: Se incluyen las celdas de código indicadas de la librería Matplotlib/Pyplot.

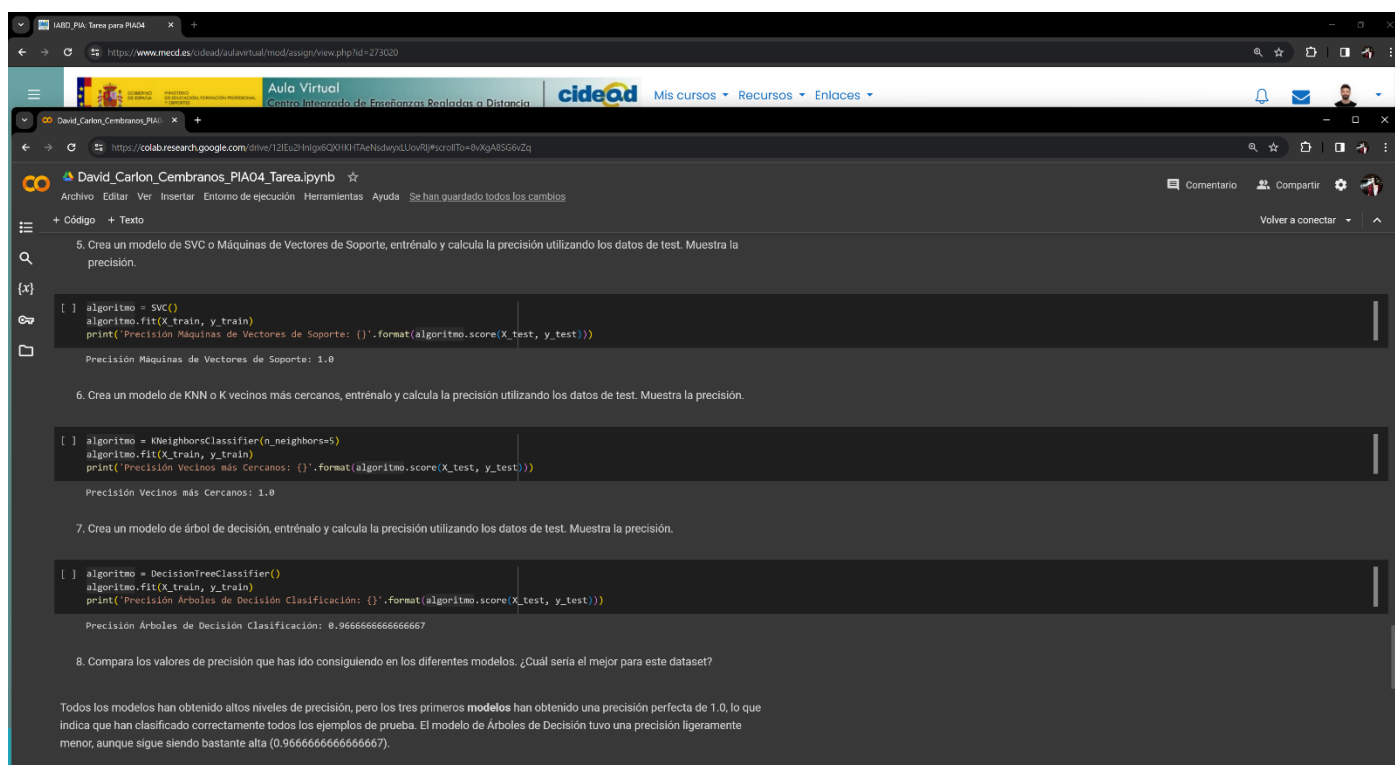




Apartado 3: Se incluyen las celdas de código necesarias para, al menos, crear y entrenar un modelo de aprendizaje automático con Scikit-learn.



Apartado 3: Se incluyen todas las celdas de código para entrenar varios modelos de aprendizaje automático basados en diferentes tipos de algoritmos. Se comparan los índices de precisión de éstos.



```
5. Crea un modelo de SVC o Máquinas de Vectores de Soporte, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.

[ ] algoritmo = SVC()
algoritmo.fit(X_train, y_train)
print('Precisión Máquinas de Vectores de Soporte: {}'.format(algoritmo.score(X_test, y_test)))

Precisión Máquinas de Vectores de Soporte: 1.0

6. Crea un modelo de KNN o K vecinos más cercanos, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.

[ ] algoritmo = KNeighborsClassifier(n_neighbors=5)
algoritmo.fit(X_train, y_train)
print('Precisión Vecinos más Cercanos: {}'.format(algoritmo.score(X_test, y_test)))

Precisión Vecinos más Cercanos: 1.0

7. Crea un modelo de árbol de decisión, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.

[ ] algoritmo = DecisionTreeClassifier()
algoritmo.fit(X_train, y_train)
print('Precisión Árboles de Decisión Clasificación: {}'.format(algoritmo.score(X_test, y_test)))

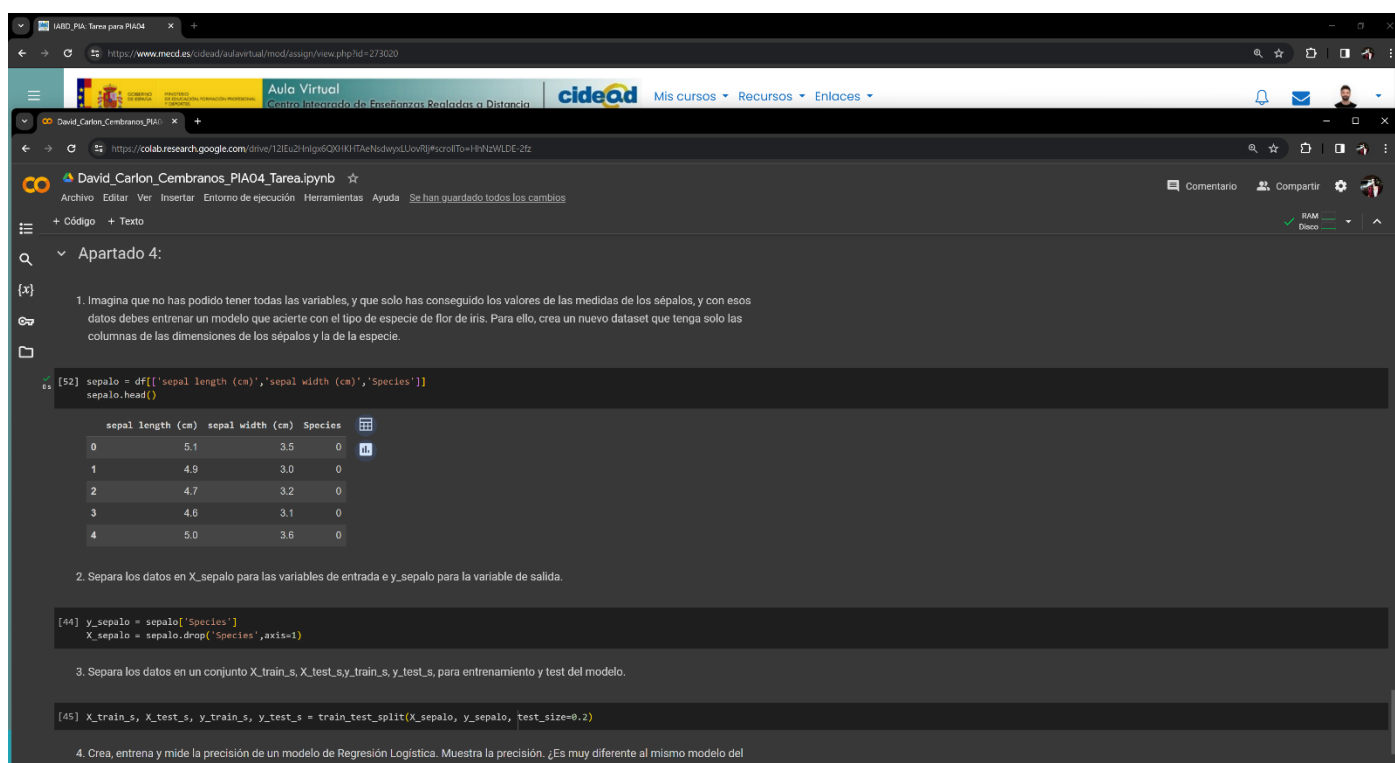
Precisión Árboles de Decisión Clasificación: 0.9666666666666667

8. Compara los valores de precisión que has ido consiguiendo en los diferentes modelos. ¿Cuál sería el mejor para este dataset?

Todos los modelos han obtenido altos niveles de precisión, pero los tres primeros modelos han obtenido una precisión perfecta de 1.0, lo que indica que han clasificado correctamente todos los ejemplos de prueba. El modelo de Árboles de Decisión tuvo una precisión ligeramente menor, aunque sigue siendo bastante alta (0.9666666666666667).
```

Todos los modelos han obtenido altos niveles de precisión, pero los tres primeros **modelos** han obtenido una precisión perfecta de 1.0, lo que indica que han clasificado correctamente todos los ejemplos de prueba. El modelo de Árboles de Decisión tuvo una precisión ligeramente menor, aunque sigue siendo bastante alta (0.9666666666666667).

Apartado 4: Se modifica el dataset de trabajo a solo dos variables (dimensiones sépalos) y se entrenan varios modelos de aprendizaje automático disponibles en Scikit-learn. Se comparan los índices de precisión con los del apartado anterior.



```
Apartado 4:

1. Imagina que no has podido tener todas las variables, y que solo has conseguido los valores de las medidas de los sépalos, y con esos datos debes entrenar un modelo que acierte con el tipo de especie de flor de iris. Para ello, crea un nuevo dataset que tenga solo las columnas de las dimensiones de los sépalos y la de la especie.

[52] sepalo = df[['sepal length (cm)', 'sepal width (cm)', 'Species']]
sepalo.head()

   sepal length (cm)  sepal width (cm)  Species
0                5.1                3.5        0
1                4.9                3.0        0
2                4.7                3.2        0
3                4.6                3.1        0
4                5.0                3.6        0

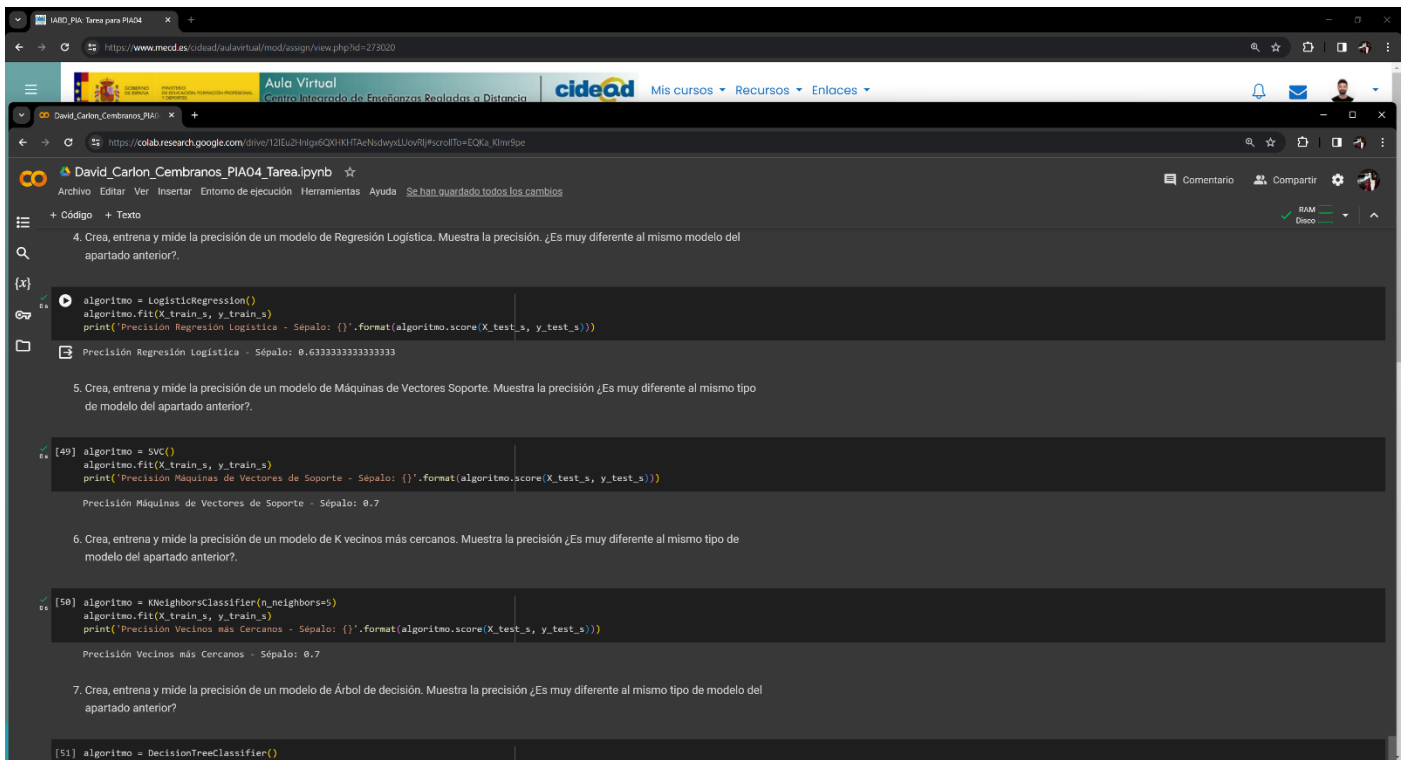
2. Separa los datos en X_sepalo para las variables de entrada y y_sepalo para la variable de salida.

[44] y_sepalo = sepalo['Species']
X_sepalo = sepalo.drop('Species', axis=1)

3. Separa los datos en un conjunto X_train_s, X_test_s, y_train_s, y_test_s, para entrenamiento y test del modelo.

[45] X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_sepalo, y_sepalo, test_size=0.2)

4. Crea, entrena y mide la precisión de un modelo de Regresión Logística. Muestra la precisión. ¿Es muy diferente al mismo modelo del
```



4. Crea, entrena y mide la precisión de un modelo de Regresión Logística. Muestra la precisión. ¿Es muy diferente al mismo modelo del apartado anterior?

```
algoritmo = LogisticRegression()  
algoritmo.fit(X_train_s, y_train_s)  
print('Precisión Regresión Logística - Sépalo: {}'.format(algoritmo.score(X_test_s, y_test_s)))
```

Precisión Regresión Logística - Sépalo: 0.6333333333333333

5. Crea, entrena y mide la precisión de un modelo de Máquinas de Vectores Soporte. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?

```
[49] algoritmo = SVC()  
algoritmo.fit(X_train_s, y_train_s)  
print('Precisión Máquinas de Vectores de Soporte - Sépalo: {}'.format(algoritmo.score(X_test_s, y_test_s)))
```

Precisión Máquinas de Vectores de Soporte - Sépalo: 0.7

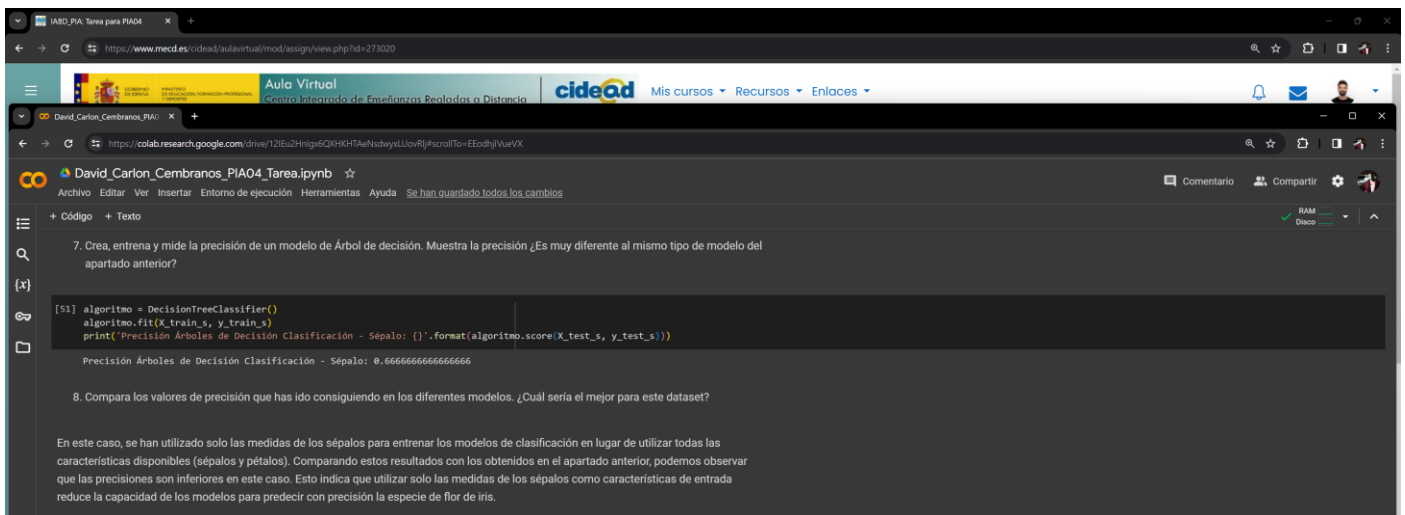
6. Crea, entrena y mide la precisión de un modelo de K vecinos más cercanos. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?

```
[50] algoritmo = KNeighborsClassifier(n_neighbors=5)  
algoritmo.fit(X_train_s, y_train_s)  
print('Precisión Vecinos más Cercanos - Sépalo: {}'.format(algoritmo.score(X_test_s, y_test_s)))
```

Precisión Vecinos más Cercanos - Sépalo: 0.7

7. Crea, entrena y mide la precisión de un modelo de Árbol de decisión. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?

```
[51] algoritmo = DecisionTreeClassifier()
```



7. Crea, entrena y mide la precisión de un modelo de Árbol de decisión. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?

```
[51] algoritmo = DecisionTreeClassifier()  
algoritmo.fit(X_train_s, y_train_s)  
print('Precisión Árboles de Decisión Clasificación - Sépalo: {}'.format(algoritmo.score(X_test_s, y_test_s)))
```

Precisión Árboles de Decisión Clasificación - Sépalo: 0.6666666666666666

8. Compara los valores de precisión que has ido consiguiendo en los diferentes modelos. ¿Cuál sería el mejor para este dataset?

En este caso, se han utilizado solo las medidas de los sépalos para entrenar los modelos de clasificación en lugar de utilizar todas las características disponibles (sépalos y pétalos). Comparando estos resultados con los obtenidos en el apartado anterior, podemos observar que las precisiones son inferiores en este caso. Esto indica que utilizar solo las medidas de los sépalos como características de entrada reduce la capacidad de los modelos para predecir con precisión la especie de flor de iris.

En este caso, se han utilizado solo las medidas de los sépalos para entrenar los modelos de clasificación en lugar de utilizar todas las características disponibles (sépalos y pétalos). Comparando estos resultados con los obtenidos en el apartado anterior, podemos observar que las precisiones son inferiores en este caso. Esto indica que utilizar solo las medidas de los sépalos como características de entrada reduce la capacidad de los modelos para predecir con precisión la especie de flor de iris.

Enlace al notebook de Google Colab:

<https://colab.research.google.com/drive/12IEu2Hnlgx6QXHKHTAeNsdwyxLUovRlj?usp=sharing>