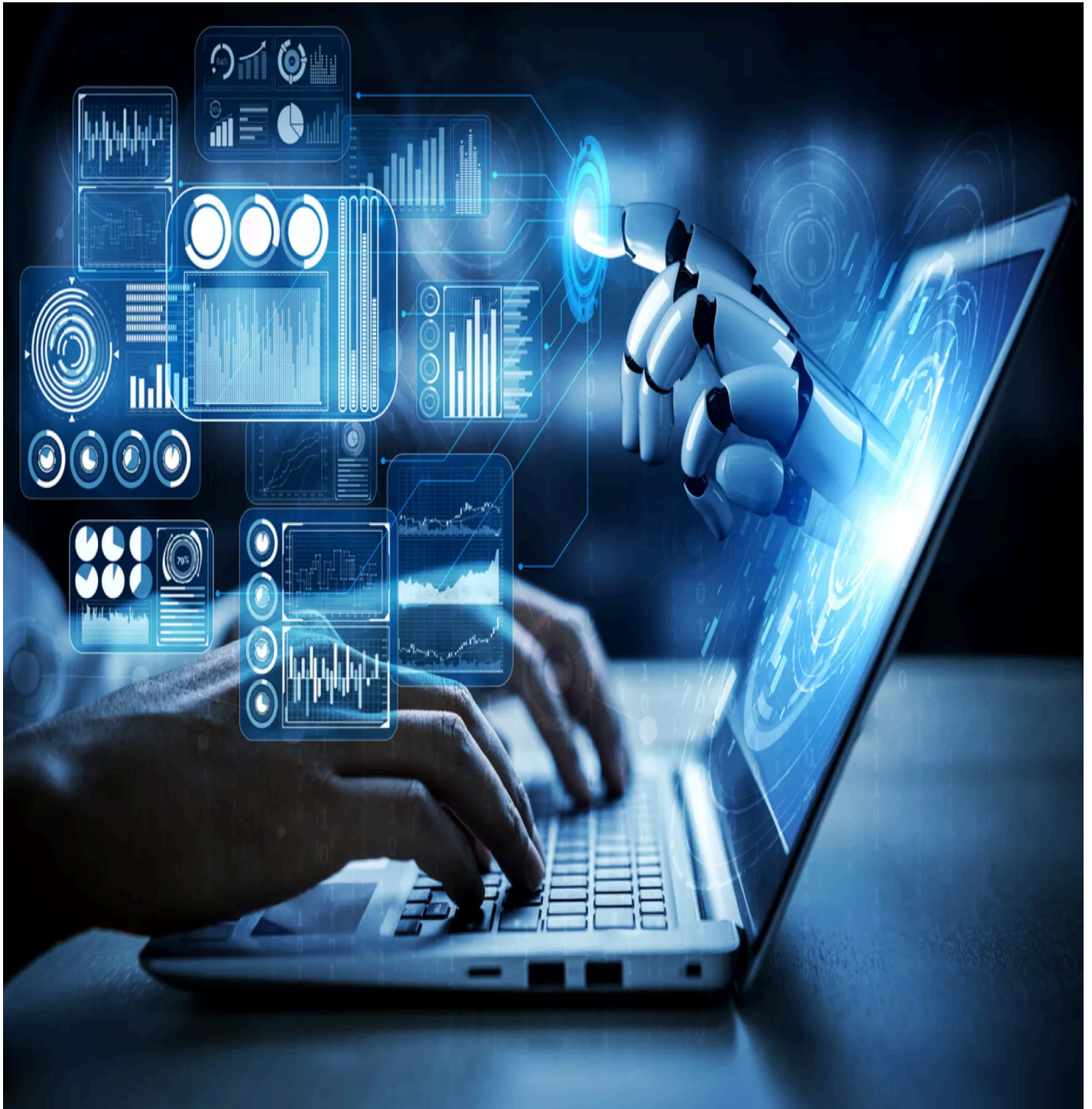


Programación de Inteligencia Artificial



Nombre: Victoria Jiménez Martín

Módulo: Programación de Inteligencia Artificial

Curso: Especialización de Inteligencia Artificial y Big Data

Índice

Apartado 1: Explora los datos con Pandas	4
Inicia un nuevo notebook, preferiblemente en Google Colab. Para guiarte en el proceso, puedes utilizar este cuaderno-guía con los fragmentos de código indicados en las celdas de texto, pero tendrás que escribir el código en la celda de código correspondiente y ejecutarlo.	4
Importa las librerías Numpy y Pandas.	4
Importa la función load_iris de la biblioteca de datasets de Scikit-learn y echa un vistazo rápido a sus principales elementos 'data', 'target' y 'target_names'.	5
Utiliza la clase DataFrame de Pandas para crear el dataset df y añade la columna "Species" a partir de la secuencia de datos target. Utiliza la función head para ver los primeros registros del nuevo dataset.	6
Utiliza la función describe para ver los principales valores estadísticos del dataset.	7
Apartado 2: Visualiza los datos con Pyplot.	8
Utiliza el paquete Pyplot para hacer representaciones gráficas de los datos. Importa Pyplot de la librería Matplotlib y crea una figura tipo "dispersión de puntos" (Scatter plot) con la variable sepal length (cm) en el eje x y la variable sepal width (cm) en el eje y.	8
Crea otra figura distinguiendo con el color azul la especie "Setosa", con el color verde la especie "Versicolor" y con color rojo la especie "Virginica".	9
Analiza cómo se distribuyen los casos si, en vez de representar según las variables de las dimensiones de los sépalos, utilizas las variables basadas en las dimensiones de los pétalos. Representa los datos en una figura tipo scatter, utilizando en el eje x la variable "petal length (cm)" y en el eje y la variable "petal width". De nuevo, distingue las especies con tres colores: color azul la especie "Setosa", con color verde la especie "Versicolor" y con color rojo la especie "Virginica".	10
Apartado 3: Entrena modelos de aprendizaje automático con Scikit-learn.	12
Importa los módulos de la librería Scikit-learn que nos permiten hacer un modelo de regresión lineal, un modelo de máquina de vectores soporte, un modelo de tipo los K vecinos más cercanos (KNN), y un modelo de tipo árbol de decisión.	12
Genera el conjunto de datos X con las variables de entrada a los modelos, y el conjunto y de las etiquetas o variable de salida del modelo, eligiendo para éste último, la variable "Species".	12
Utiliza la función train_test_split para separar los datos en el conjunto train y test según el ejemplo.	13
Crea un modelo de Regresión Logística. Utiliza la función fit para entrenarlo y utiliza la función predict sobre los datos de test para medir la precisión del modelo. Muestra el valor de dicha precisión con print.	13
Crea un modelo de SVC o Máquinas de Vectores de Soporte, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.	14
Crea un modelo de KNN o K vecinos más cercanos, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.	14
Crea un modelo de árbol de decisión, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión.	15
Compara los valores de precisión que has ido consiguiendo en los diferentes modelos. ¿Cuál sería el mejor para este dataset?	15
Apartado 4: Entrena modelos de aprendizaje automático con pocas variables.	16
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. 16

Separa los datos en X_sepalo para las variables de entrada e y_sepalo para la variable de salida. 17

Separa los datos en un conjunto X_train_s, X_test_s, y_train_s, y_test_s, para entrenamiento y test del modelo. 17

Crea, entrena y mide la precisión de un modelo de Regresión Logística. Muestra la precisión. ¿Es muy diferente al mismo modelo del apartado anterior?. 18

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?. 18

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?. 19

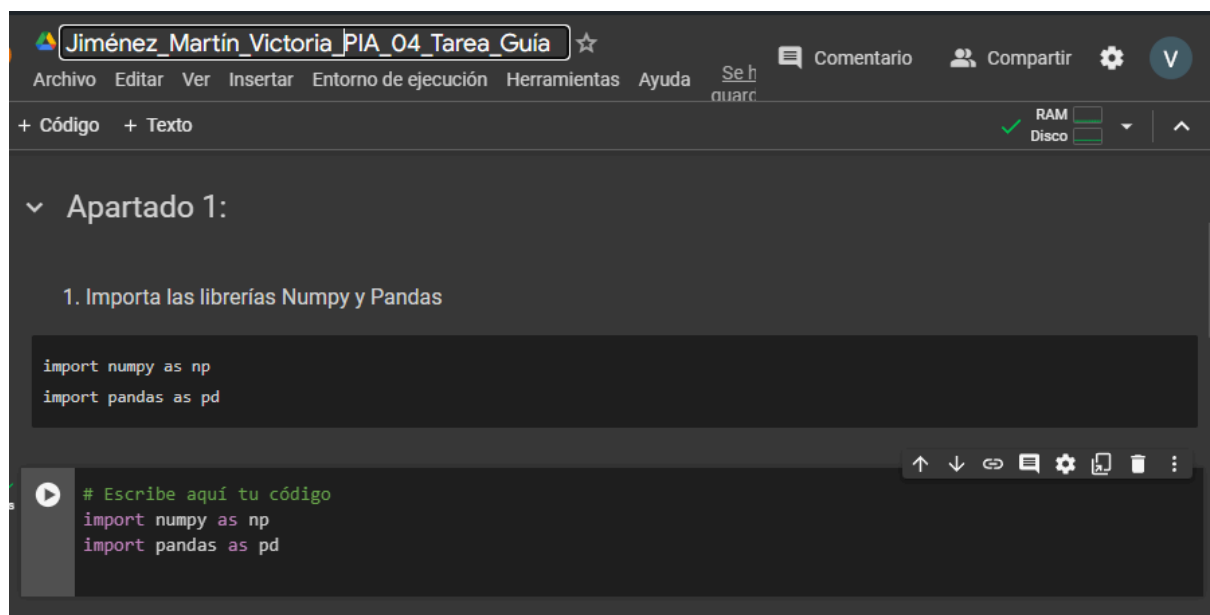
Crea, entrena y mide la precisión de un modelo de Árbol de decisión. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?. 19

Compara los valores de precisión que has ido consiguiendo en los diferentes modelos. ¿Cuál sería el mejor para este dataset?. 19

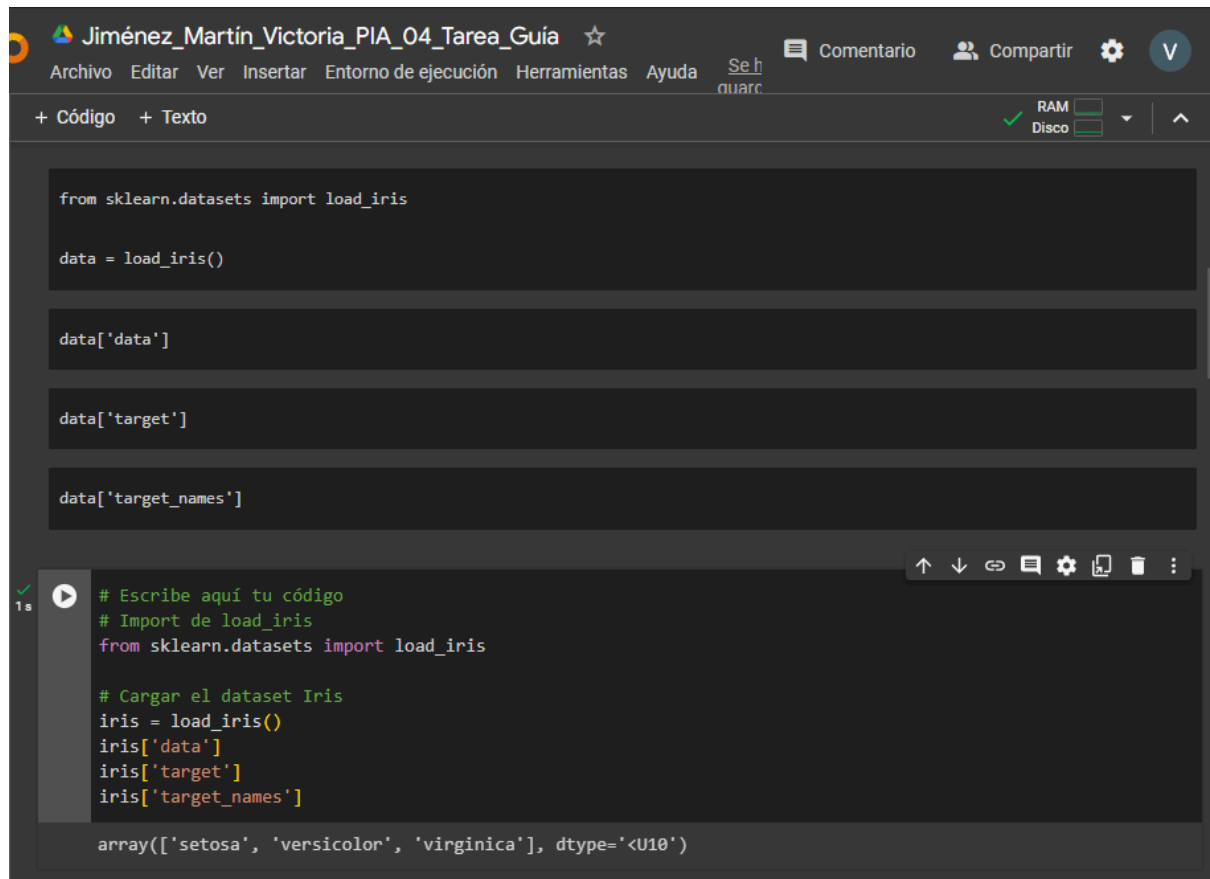
Apartado 1: Explora los datos con Pandas

Inicia un nuevo notebook, preferiblemente en Google Colab. Para guiarte en el proceso, puedes utilizar este cuaderno-guía con los fragmentos de código indicados en las celdas de texto, pero tendrás que escribir el código en la celda de código correspondiente y ejecutarlo.

Importa las librerías **Numpy** y **Pandas**.



Importa la función `load_iris` de la biblioteca de datasets de **Scikit-learn** y echa un vistazo rápido a sus principales elementos `'data'`, `'target'` y `'target_names'`.



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes the username 'Jiménez_Martín_Victoria_PIA_04_Tarea_Guía' and various icons for file management, execution, and sharing. The notebook has two cells. The first cell contains the following code:

```
from sklearn.datasets import load_iris

data = load_iris()

data['data']

data['target']

data['target_names']
```

The second cell shows the execution of the code, with a green checkmark and a play button icon. The output of the code is displayed below the cell:



```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Utiliza la clase *DataFrame* de Pandas para crear el dataset *df* y añade la columna "Species" a partir de la secuencia de datos *target*. Utiliza la función *head* para ver los primeros registros del nuevo dataset.

Jiménez_Martín_Victoria_PIA_04_Tarea_Guía ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto

RAM  Disco 

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.

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

Escribe aquí tu código

```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['Species'] = iris.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
4	5.0	3.6	1.4	0.2	0

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

Jiménez_Martín_Victoria_PIA_04_Tarea_Guía

ComentarioCompartir

ArchivoEditarVerInsertarEntorno de ejecuciónHerramientasAyudaGuar

+ Código+ Texto

RAMDisco

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

```
df.describe()
```

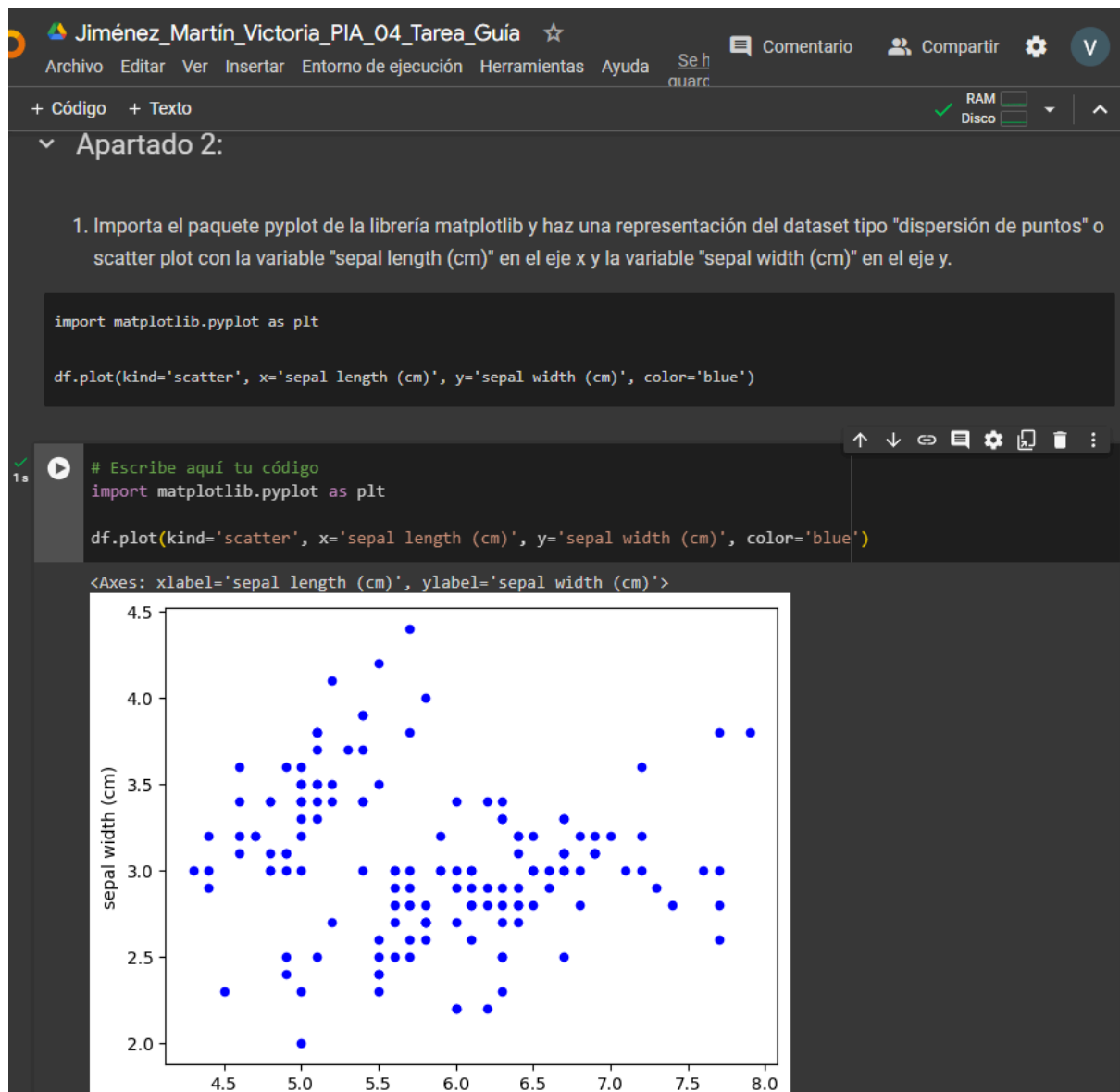
Escribe aquí tu código

```
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	0.819232
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

Apartado 2: Visualiza los datos con Pyplot.

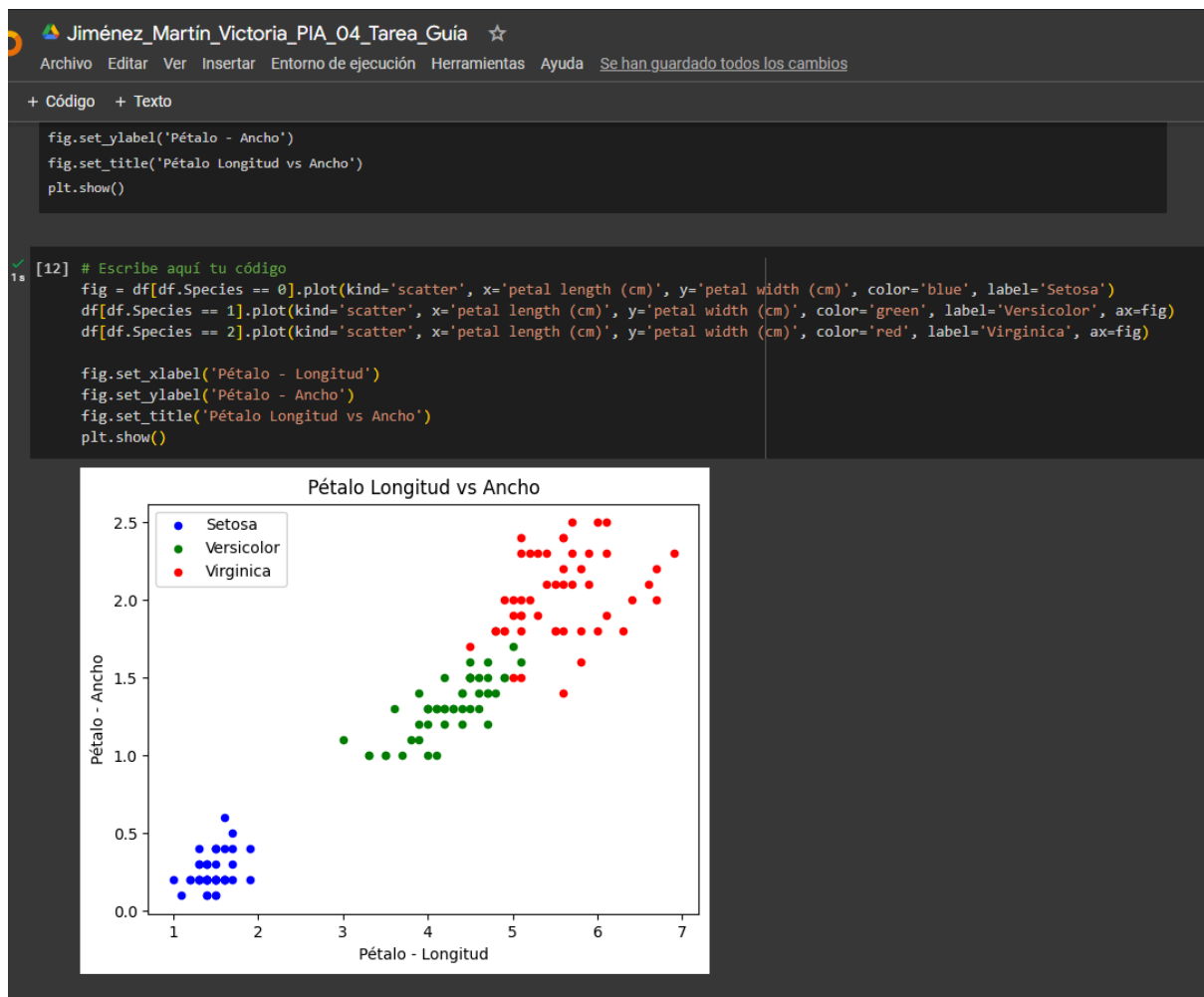
Utiliza el paquete **Pyplot** para hacer representaciones gráficas de los datos. Importa **Pyplot** de la librería **Matplotlib** y crea una figura tipo "dispersión de puntos" (Scatter plot) con la variable *sepal length (cm)* en el eje x y la variable *sepal width (cm)* en el eje y.



Crea otra figura distinguiendo con el color azul la especie "Setosa", con el color verde la especie "Versicolor" y con color rojo la especie "Virginica".



Analiza cómo se distribuyen los casos si, en vez de representar según las variables de las dimensiones de los sépalos, utilizas las variables basadas en las dimensiones de los pétalos. Representa los datos en una figura tipo scatter, utilizando en el eje x la variable "petal length (cm)" y en el eje y la variable "petal width". De nuevo, distingue las especies con tres colores: color azul la especie "Setosa", con color verde la especie "Versicolor" y con color rojo la especie "Virginica".



Podemos realizar el siguiente análisis:

Iris Setosa se distingue fácilmente de Iris Versicolor e Iris Virginica en ambos tipos de gráficos debido a sus características tanto en sépalos como en pétalos.

La clasificación entre Versicolor y Virginica es más retadora al usar solo las dimensiones de los sépalos, ya que estas especies muestran superposición en sus medidas. Sin embargo, las dimensiones de los pétalos ofrecen una distinción más clara y son más efectivas para diferenciar entre estas dos especies.

Las medidas de los pétalos son, por lo tanto, indicadores más útiles para clasificar las tres especies de Iris, proporcionando una separación más definida en el análisis visual.

Apartado 3: Entrena modelos de aprendizaje automático con Scikit-learn.

Importa los módulos de la librería **Scikit-learn** que nos permiten hacer un modelo de **regresión lineal**, un modelo de **máquina de vectores soporte**, un modelo de tipo los **K vecinos más cercanos (KNN)**, y un modelo de tipo **árbol de decisión**.



Jiménez_Martín_Victoria_PIA_04_Tarea_Guía ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto

▼ Apartado 3:

1. Importa los módulos de la librería Scikit-learn que nos permiten hacer un modelo de regresión lineal, un modelo de máquina de vectores soporte, un modelo de tipo los K vecinos más cercanos (KNN), y un modelo de tipo árbol de decisión.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

0s # Escribe aquí tu código

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

Genera el conjunto de datos X con las variables de entrada a los modelos, y el conjunto y de las etiquetas o variable de salida del modelo, eligiendo para éste último, la variable "Species".



Jiménez_Martín_Victoria_PIA_04_Tarea_Guía ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

+ Código + Texto

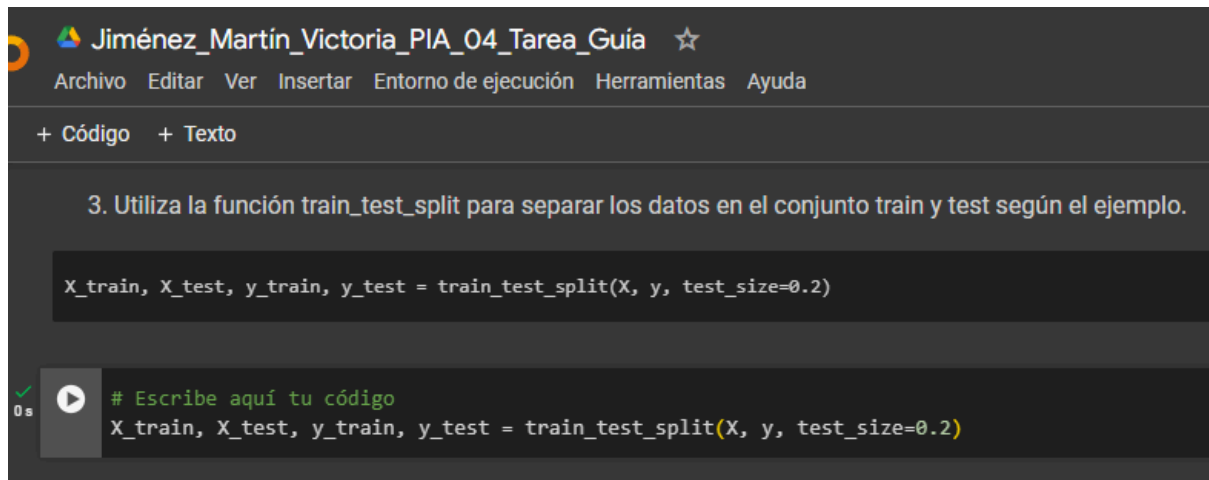
2. Genera el conjunto de datos X con las variables de entrada a los modelos, y el conjunto y de las etiquetas o variable de salida del modelo, eligiendo para éste último, la variable "Species".

```
y = df['Species']
X = df.drop('Species',axis=1)
```

0s # Escribe aquí tu código

```
X = df.drop('Species',axis=1)
y = df['Species']
```

Utiliza la función *train_test_split* para separar los datos en el conjunto train y test según el ejemplo.



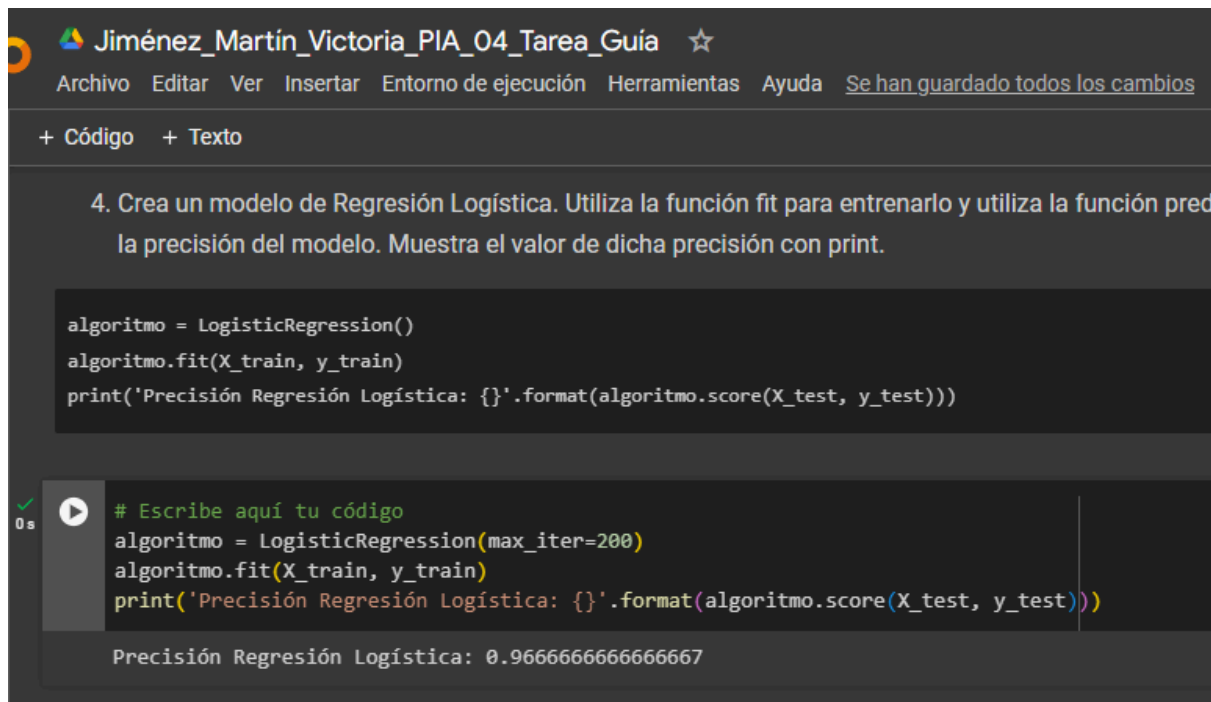
The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu bar, there are tabs for "+ Código" and "+ Texto". The main area displays a code cell with the following text:

```
3. Utiliza la función train_test_split para separar los datos en el conjunto train y test según el ejemplo.
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

At the bottom, there is a run button (a play icon) and a status bar showing "0s".

Crea un modelo de Regresión Logística. Utiliza la función *fit* para entrenarlo y utiliza la función *predict* sobre los datos de test para medir la precisión del modelo. Muestra el valor de dicha precisión con *print*.



The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and "Se han guardado todos los cambios". Below the menu bar, there are tabs for "+ Código" and "+ Texto". The main area displays a code cell with the following text:

```
4. Crea un modelo de Regresión Logística. Utiliza la función fit para entrenarlo y utiliza la función predict para medir la precisión del modelo. Muestra el valor de dicha precisión con print.
```

```
algoritmo = LogisticRegression()  
algoritmo.fit(X_train, y_train)  
print('Precisión Regresión Logística: {}'.format(algoritmo.score(X_test, y_test)))
```

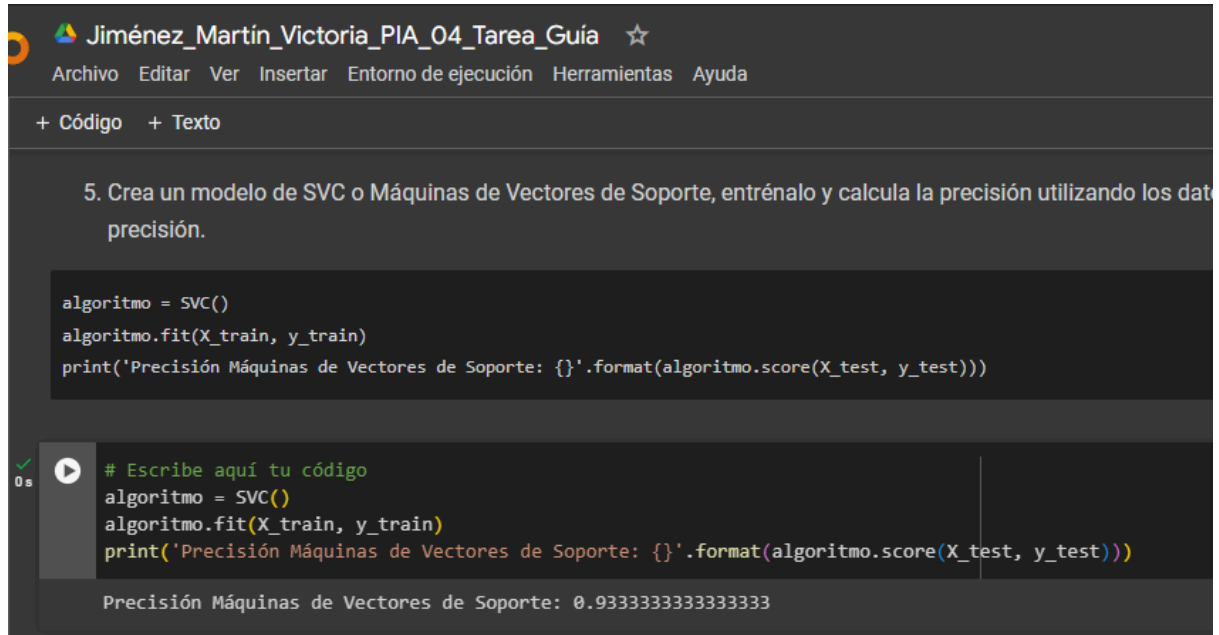
At the bottom, there is a run button (a play icon) and a status bar showing "0s".

```
algoritmo = LogisticRegression(max_iter=200)  
algoritmo.fit(X_train, y_train)  
print('Precisión Regresión Logística: {}'.format(algoritmo.score(X_test, y_test)))
```

The output of the code is displayed below the cell:

```
Precisión Regresión Logística: 0.9666666666666667
```

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



The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar reads "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía" with a star icon. The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu, there are tabs for "+ Código" and "+ Texto". The main cell contains the following text: "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." Below this, a code block contains the following Python code:

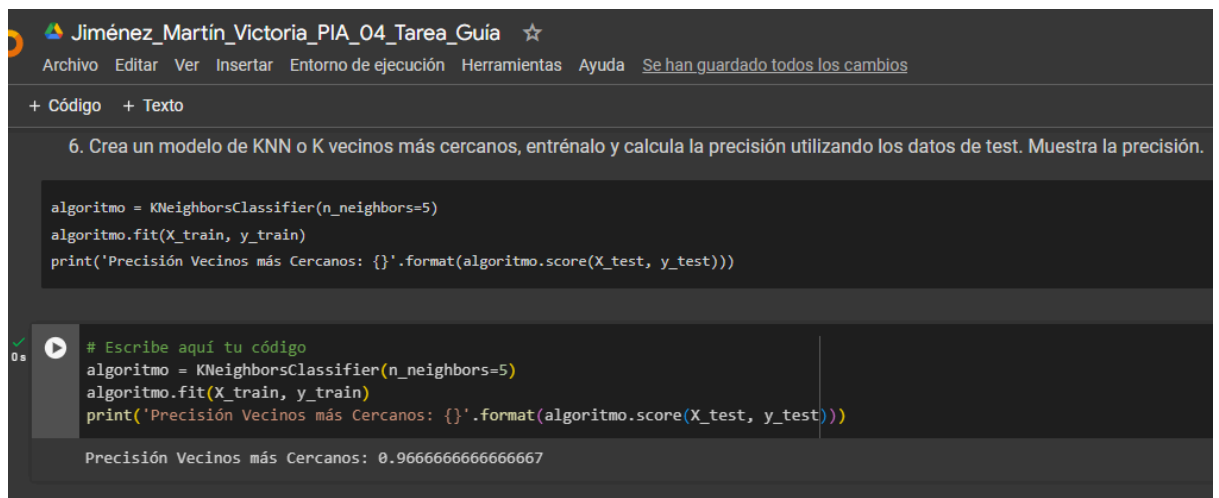
```
algoritmo = SVC()
algoritmo.fit(X_train, y_train)
print('Precisión Máquinas de Vectores de Soporte: {}'.format(algoritmo.score(X_test, y_test)))
```

 The code is executed, and the output is displayed below the code block: "Precisión Máquinas de Vectores de Soporte: 0.9333333333333333".

```
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: 0.9333333333333333

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



The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar reads "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía" with a star icon. The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and "Se han guardado todos los cambios". Below the menu, there are tabs for "+ Código" and "+ Texto". The main cell contains the following text: "6. Crea un modelo de KNN o K vecinos más cercanos, entrénalo y calcula la precisión utilizando los datos de test. Muestra la precisión." Below this, a code block contains the following Python code:

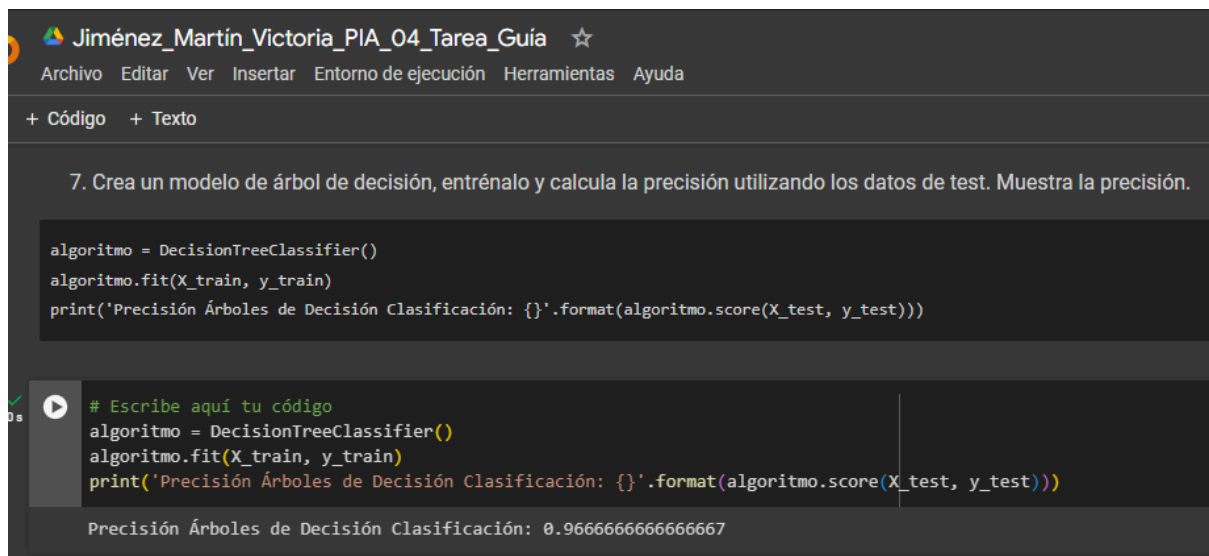
```
algoritmo = KNeighborsClassifier(n_neighbors=5)
algoritmo.fit(X_train, y_train)
print('Precisión Vecinos más Cercanos: {}'.format(algoritmo.score(X_test, y_test)))
```

 The code is executed, and the output is displayed below the code block: "Precisión Vecinos más Cercanos: 0.9666666666666667".

```
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: 0.9666666666666667

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



The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu is a toolbar with "+ Código" and "+ Texto". The notebook contains a cell with the following Python code:

```
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)))
```

Below the code cell is an execution bar with a play button icon and the text "# Escribe aquí tu código". The output of the code is displayed below the execution bar:

```
Precisión Árboles de Decisión Clasificación: 0.9666666666666667
```

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

Regresión Logística es efectiva para clases linealmente separables, lo cual es relativamente adecuado para el dataset Iris.

SVC funciona bien con una clara separación de clases y es potente en espacios de alta dimensión, esperando un alto rendimiento para Iris.

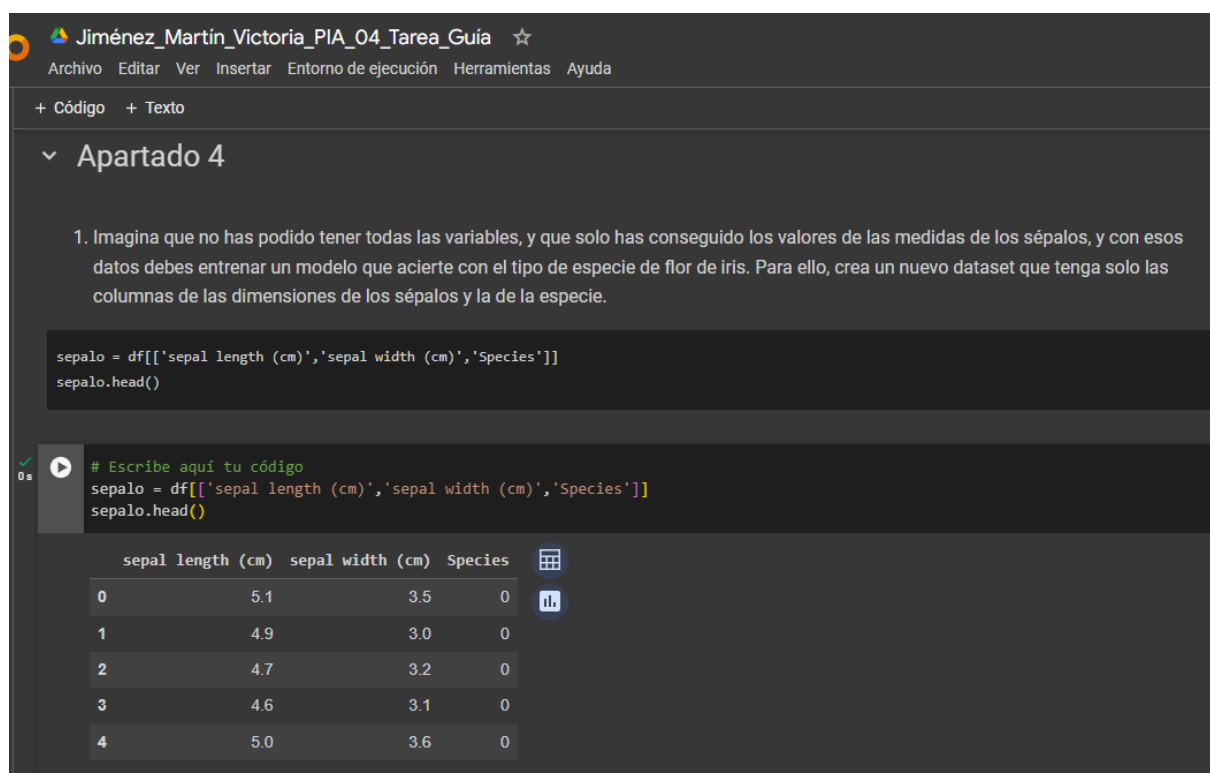
KNN se basa en la proximidad de las características para clasificar y puede ser muy efectivo si se elige correctamente el número de vecinos.

Árbol de Decisión es intuitivo y captura patrones complejos, pero puede ser propenso al sobreajuste si no se gestiona cuidadosamente.

Para este caso, la mejor opción dependería de la precisión obtenida en el conjunto de pruebas entre otros factores. Dado que Iris es un dataset sencillo, todos los modelos pueden tener un buen desempeño. SVC y Árboles de Decisión podrían ser una mejor opción debido a que estos destacan por su habilidad para manejar separaciones no lineales y estructuras complejas.

Apartado 4: Entrena modelos de aprendizaje automático con pocas variables.

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.



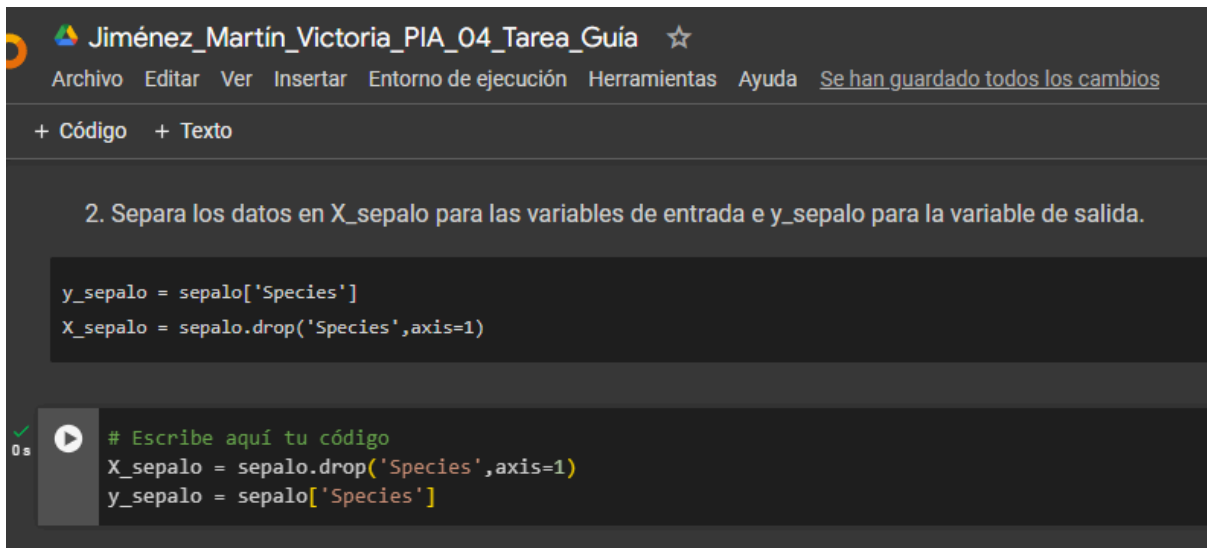
The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The notebook is on "Apartado 4" and contains a single code cell. The code in the cell is:

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

The output of the code cell is a DataFrame with 5 rows and 4 columns. The columns are "sepal length (cm)", "sepal width (cm)", "Species", and an unnamed column with values 0. The data is as follows:

	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	

Separa los datos en *X_sepalo* para las variables de entrada e *y_sepalo* para la variable de salida.



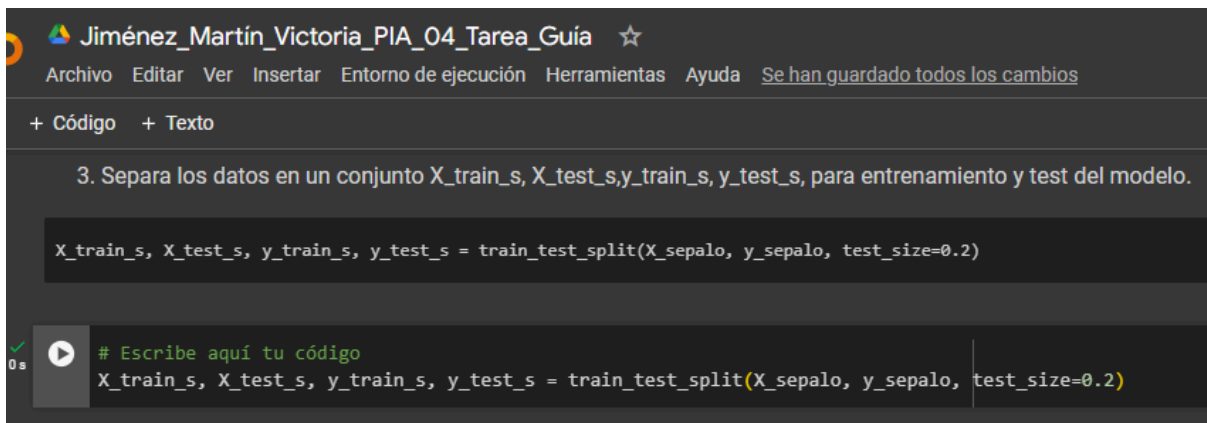
The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and a status message "Se han guardado todos los cambios". Below the menu, there are tabs for "+ Código" and "+ Texto". The main cell contains the following text: "2. Separa los datos en X_sepalo para las variables de entrada e y_sepalo para la variable de salida." Below this, a code cell is shown with the following Python code:

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

The code cell is executed, showing a green checkmark and a play button icon. The output of the code is displayed below the code cell:

```
# Escribe aquí tu código
X_sepalo = sepalo.drop('Species',axis=1)
y_sepalo = sepalo['Species']
```

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



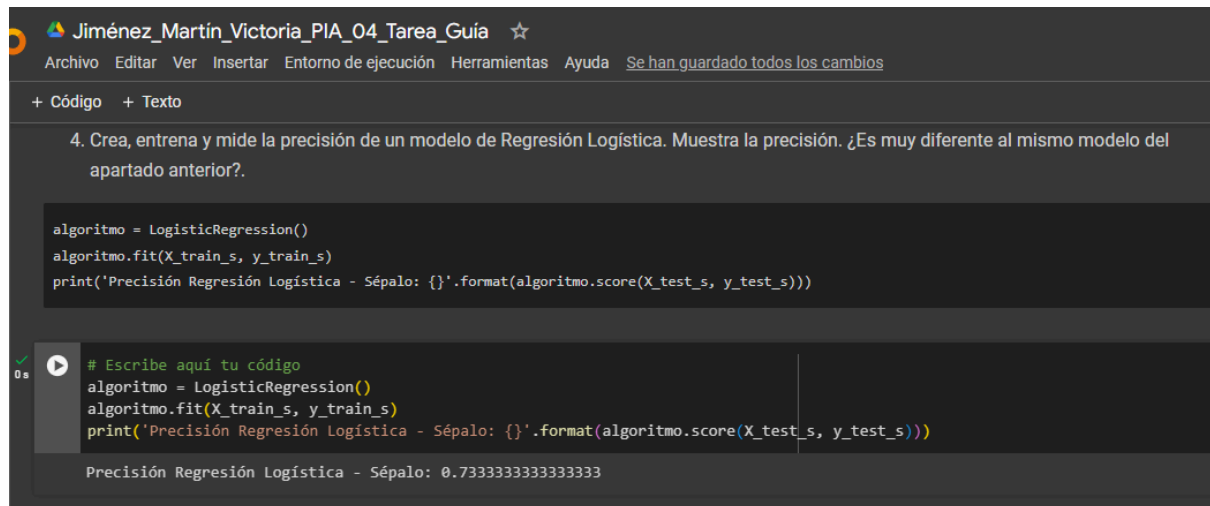
The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The menu bar includes "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and a status message "Se han guardado todos los cambios". Below the menu, there are tabs for "+ Código" and "+ Texto". The main cell contains the following text: "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." Below this, a code cell is shown with the following Python code:

```
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_sepalo, y_sepalo, test_size=0.2)
```

The code cell is executed, showing a green checkmark and a play button icon. The output of the code is displayed below the code cell:

```
# Escribe aquí tu código
X_train_s, X_test_s, y_train_s, y_test_s = train_test_split(X_sepalo, y_sepalo, test_size=0.2)
```

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



The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu, there are tabs for "+ Código" and "+ Texto". The main content area displays a code cell with the following Python code:

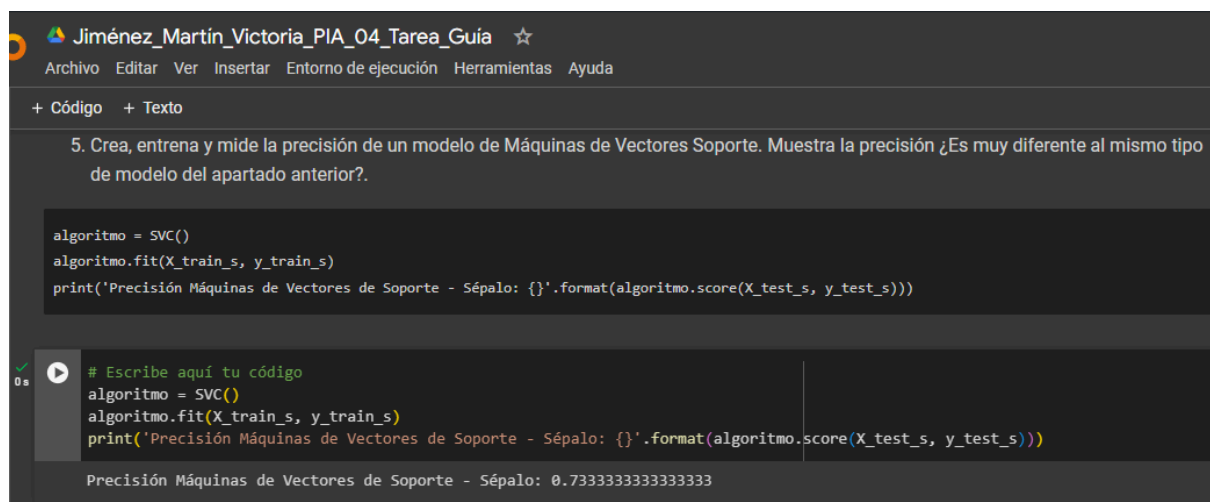
```
4. Crea, entrena y mide la precisión de un modelo de Regresión Logística. Muestra la precisión. ¿Es muy diferente al mismo modelo del apartado anterior?.
```

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

Below the code cell, there is a run button (a play icon) and a status bar indicating "0 s". The output of the code is displayed in a separate cell:

```
Precisión Regresión Logística - Sépalo: 0.7333333333333333
```

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?.



The screenshot shows a Jupyter Notebook window titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu, there are tabs for "+ Código" and "+ Texto". The main content area displays a code cell with the following Python code:

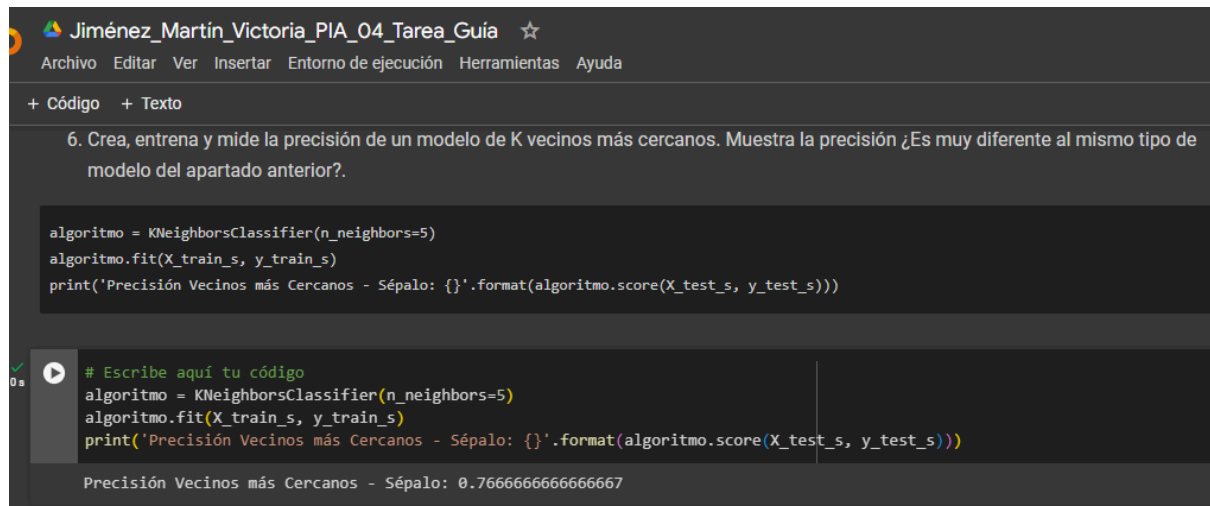
```
5. Crea, entrena y mide la precisión de un modelo de Máquinas de Vectores Soporte. Muestra la precisión ¿Es muy diferente al mismo tipo de modelo del apartado anterior?.
```

```
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)))
```

Below the code cell, there is a run button (a play icon) and a status bar indicating "0 s". The output of the code is displayed in a separate cell:

```
Precisión Máquinas de Vectores de Soporte - Sépalo: 0.7333333333333333
```

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?.



The screenshot shows a Jupyter Notebook titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". Below the menu, there are tabs for "+ Código" and "+ Texto". The main content area displays the following text:

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?.

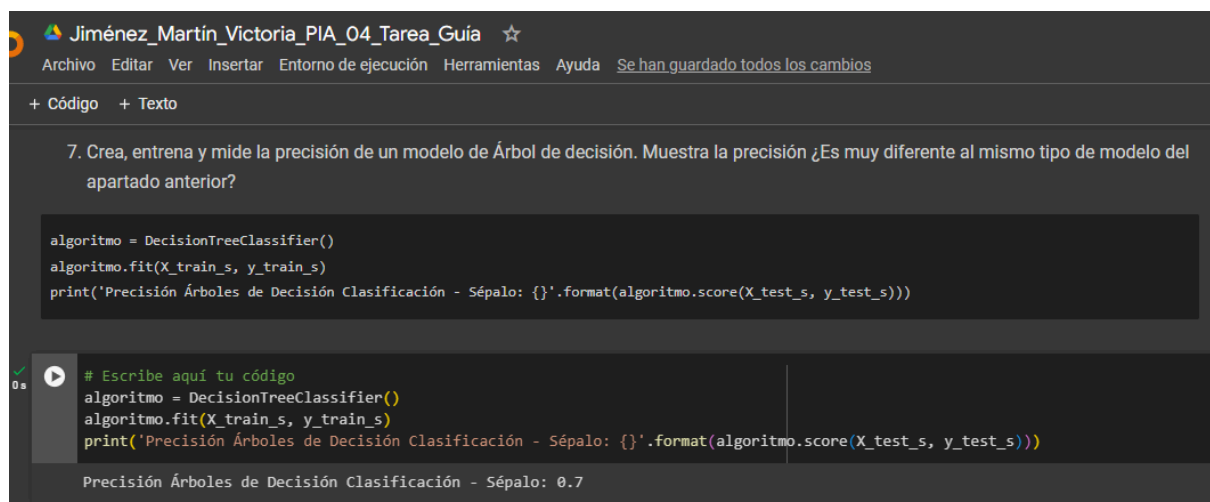
```
algoritmo = KNeighborsClassifier(n_neighbors=5)
algoritmo.fit(X_train_s, y_train_s)
print('Precisión Vecinos más Cercanos - Sépalo: {}'.format(algoritmo.score(X_test_s, y_test_s)))
```

Below the code, there is a play button icon and a comment "# Escribe aquí tu código". The code is repeated:

```
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)))
```

The output of the code is displayed at the bottom: "Precisión Vecinos más Cercanos - Sépalo: 0.7666666666666667".

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?.



The screenshot shows a Jupyter Notebook titled "Jiménez_Martín_Victoria_PIA_04_Tarea_Guía". The interface includes a menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", "Ayuda", and "Se han guardado todos los cambios". Below the menu, there are tabs for "+ Código" and "+ Texto". The main content area displays the following text:

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?

```
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)))
```

Below the code, there is a play button icon and a comment "# Escribe aquí tu código". The code is repeated:

```
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)))
```

The output of the code is displayed at the bottom: "Precisión Árboles de Decisión Clasificación - Sépalo: 0.7".

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

Podemos comprobar que se pierde la precisión cuando se utilizan únicamente las medidas de los sépalos para entrenar modelos de aprendizaje automático con el dataset Iris, comparado con el uso completo de las características (incluyendo los pétalos). Esto se debe a que las medidas de los pétalos ofrecen información crucial para distinguir entre las especies de Iris, especialmente entre Versicolor y Virginica. Los modelos, como la Regresión Logística, SVC, KNN y Árboles de Decisión, pueden verse afectados de diferentes maneras:

- Regresión Logística y SVC podrían enfrentar más dificultades para clasificar las especies correctamente debido a la menor separabilidad lineal con solo las medidas de los sépalos.
- KNN puede ser menos afectado gracias a su enfoque en la proximidad de los datos, pero la elección de k se vuelve crítica.
- Árboles de Decisión pueden manejar bien el conjunto de datos reducido por su habilidad para enfocarse en las características más importantes, aunque el riesgo de sobreajuste puede aumentar.

<https://colab.research.google.com/drive/1gwvgkjbV0yn0l1gwWLJBmIfn609dppLf?usp=sharing>