```
from google.colab import drive
drive.mount('/content/drive')
!pwd
```

```
Mounted at /content/drive
/content
```

```
%cd "MyDrive"
!ls
```

```
/content/drive/MyDrive
'1.2.1 SE Ejercicio de Planeación de Base de Datos.gsheet'
'1.2.2Ejercicio de Modelacion.drawio'
'2021-11-25 (2).png'
'2021-11-25 (3).png'
'2.2.6 Actividad Ejercicio Clase Modelo relacional a Modelo relacional.gsheet'
'2.4.2 Actividad Ejercicio 2 Entidad relación a Modelo relacional.gdoc'
 2.5.1_Práctica_SQL-1.docx
 2.6.1_Práctica_Normalización.docx
 A01236390_1erPasaporte
 A01236390_LINUX.docx
 A3D1F470-7683-4F51-8826-FBD35A4EA835.jpeg
 Act2-8.ipynb
'Actividad 2 Usabilidad y Jugabilidad.gdoc'
 Actividad41InterseccionConvexHull.gdoc
'Actividad 5.3 Hill Climber, ILS, SA.gdoc'
'Actividad de aprendizaje 4 (1).gdoc'
'Actividad de aprendizaje 4.gdoc'
 Activity1.mp4
 ag_new_csv
'Analisis del Contexto y la Normatividad .gdoc'
 AnalisisyReporte.ipynb
'Base de datos distribuidas.gdoc'
 C2AA989C-3C86-44BD-8F83-A1AC5E868F90.jpeg
 Cancion.gdoc
'clase 18 (1).gdoc'
'clase 18.gdoc'
'clase 22.gdoc'
 climate_data_2009_2012.csv
 CloudSecurity_Equipo1.gdoc
'Colab Notebooks'
'Copia_de_Sports_Analytics_y_EDA (3).ipynb'
'CURP_PEGC030808HCLRLLA6 (1).pdf'
 CURP_PEGC030808HCLRLLA6.pdf
 DataModifications.ipynb
'¿Debemos preocuparnos por la IA?.gdoc'
'Discurso individual Guion.gdoc'
'Diseño de Prueba.gsheet'
'Diseño inicial web del proyecto integrador.gslides'
'DocumentoFinalClubAmanecer (1).pdf'
'Documento sin título (1).gdoc'
'Documento sin título (2).gdoc'
'Documento sin título (3).gdoc'
'Documento sin título (4).gdoc'
'Documento sin título (5).gdoc'
'Documento sin título (6).gdoc'
'Documento sin título (7).gdoc'
'Documento sin título.gdoc'
'Ecuacion en geogebra.mp4'
'Ejercicio MongoDB.gdoc'
'Entrega 0: ChatBot.gdoc'
 EntregaProyecto.unitypackage
 EntregaProyecto.zip
'evidencia 1 2.docx'
'Evidencia entrega proyecto final Construccion de software'
'Evidencia Integradora final: Plan de desarrollo personal.gslides'
'Experimento1 FJ22.docx'
 Exposicion
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
```

```
data = pd.read_csv("Student_performance_data _.csv")
dataset = data.drop(columns=['StudentID', 'GradeClass', 'Ethnicity'])
```

```
dataset['Activities'] = dataset['Extracurricular'] + dataset['Sports'] + dataset['Music'] + dataset['Volunteering']
dataset = dataset.drop(columns=['Extracurricular', 'Sports', 'Music', 'Volunteering'])
dataset
```

	Age	Gender	ParentalEducation	StudyTimeWeekly	Absences	Tutoring	ParentalSupport	GPA	Activities
0	17	1	2	19.833723	7	1	2	2.929196	1
1	18	0	1	15.408756	0	0	1	3.042915	0
2	15	0	3	4.210570	26	0	2	0.112602	0
3	17	1	3	10.028829	14	0	3	2.054218	1
4	17	1	2	4.672495	17	1	3	1.288061	0
...	...	...	...	...	...	...	...	...	...
2387	18	1	3	10.680555	2	0	4	3.455509	1
2388	17	0	1	7.583217	4	1	4	3.279150	1
2389	16	1	2	6.805500	20	0	2	1.142333	1
2390	16	1	0	12.416653	17	0	2	1.803297	2
2391	16	1	2	17.819907	13	0	2	2.140014	1

2392 rows × 9 columns

Pasos siguientes: Generar código con dataset | Ver gráficos recomendados | New interactive sheet

```
X = dataset.drop(columns=['GPA'])
y = dataset['GPA']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
def build_and_evaluate(model):
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
    history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_size=32, callbacks=[early_stop], verbose=0)
    results = model.evaluate(X_test, y_test, verbose=0)
    return results
```

```
model1 = Sequential()
model1.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model1.add(Dense(1))

model2 = Sequential()
model2.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model2.add(Dense(64, activation='relu'))
model2.add(Dense(1))

model3 = Sequential()
model3.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model3.add(Dropout(0.3))
model3.add(Dense(64, activation='relu'))
model3.add(Dropout(0.3))
model3.add(Dense(1))

model4 = Sequential()
model4.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model4.add(Dropout(0.3))
model4.add(BatchNormalization())
model4.add(Dense(64, activation='relu'))
model4.add(Dropout(0.3))
model4.add(BatchNormalization())
model4.add(Dense(1))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
results = []
```

```
models = [model1, model2, model3, model4]
for i, model in enumerate(models, start=1):
    mse, mae = build_and_evaluate(model)
    results.append([f"Experiment {i}", mse, mae])

results_df = pd.DataFrame(results, columns=['Experiment', 'MSE', 'MAE'])
print(results_df)
```

```
       Experiment       MSE       MAE
0  Experiment 1  0.050003  0.174853
1  Experiment 2  0.053644  0.181894
2  Experiment 3  0.049275  0.174656
3  Experiment 4  0.047154  0.171172
```