

INTELIGENCIA ARTIFICIAL

Tercera Entrega

TIROTEOS EN USA

Proyecto realizado por:

Wilmer Andres Romero Cala - 2214102

Jhonalber David Rangel Medina - 2225510

Predecir el numero total de victimas

1

**Simulación y
normalización**

2

Selección de variables

3

**División en entrenamiento y
prueba**

4

Definición del modelo

5

Compilación

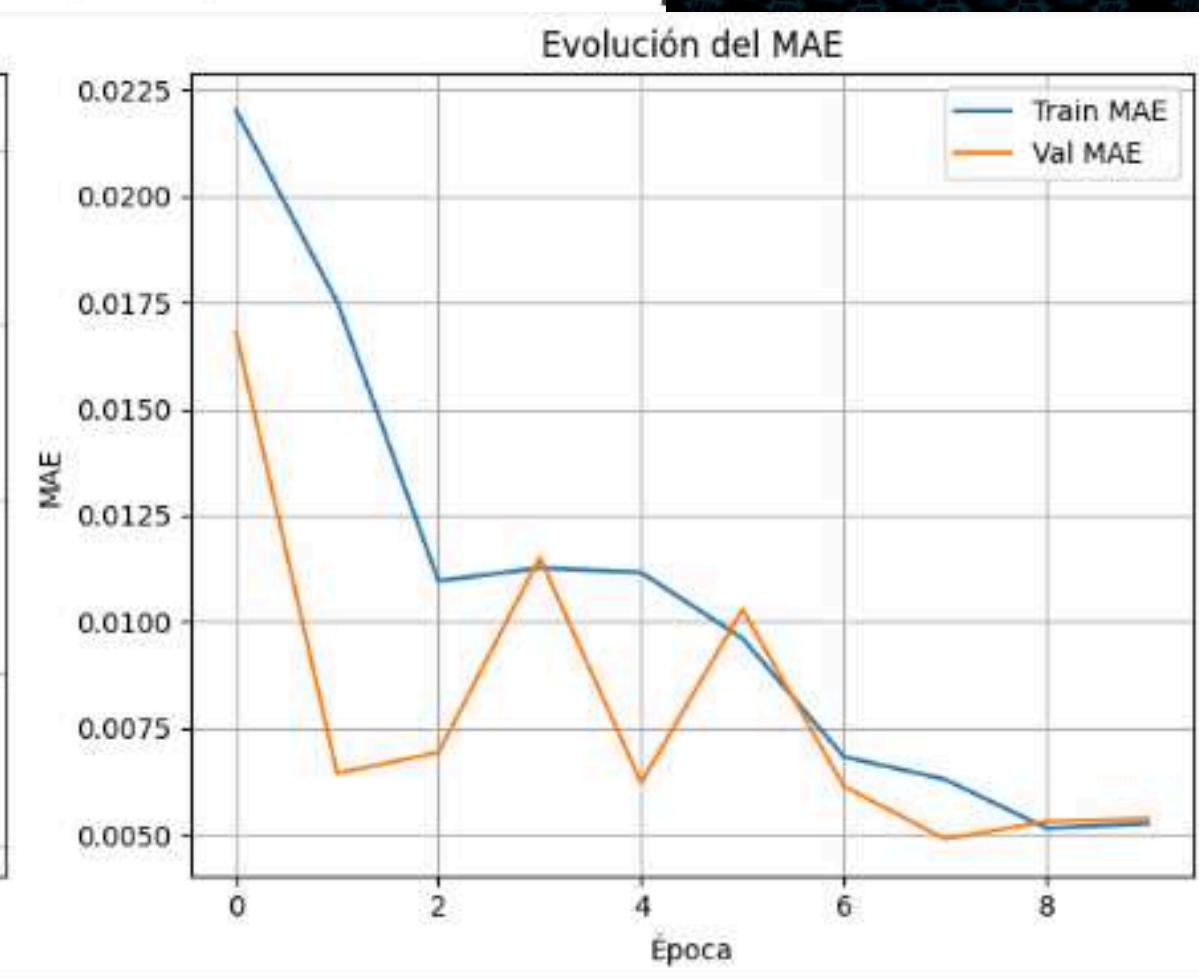
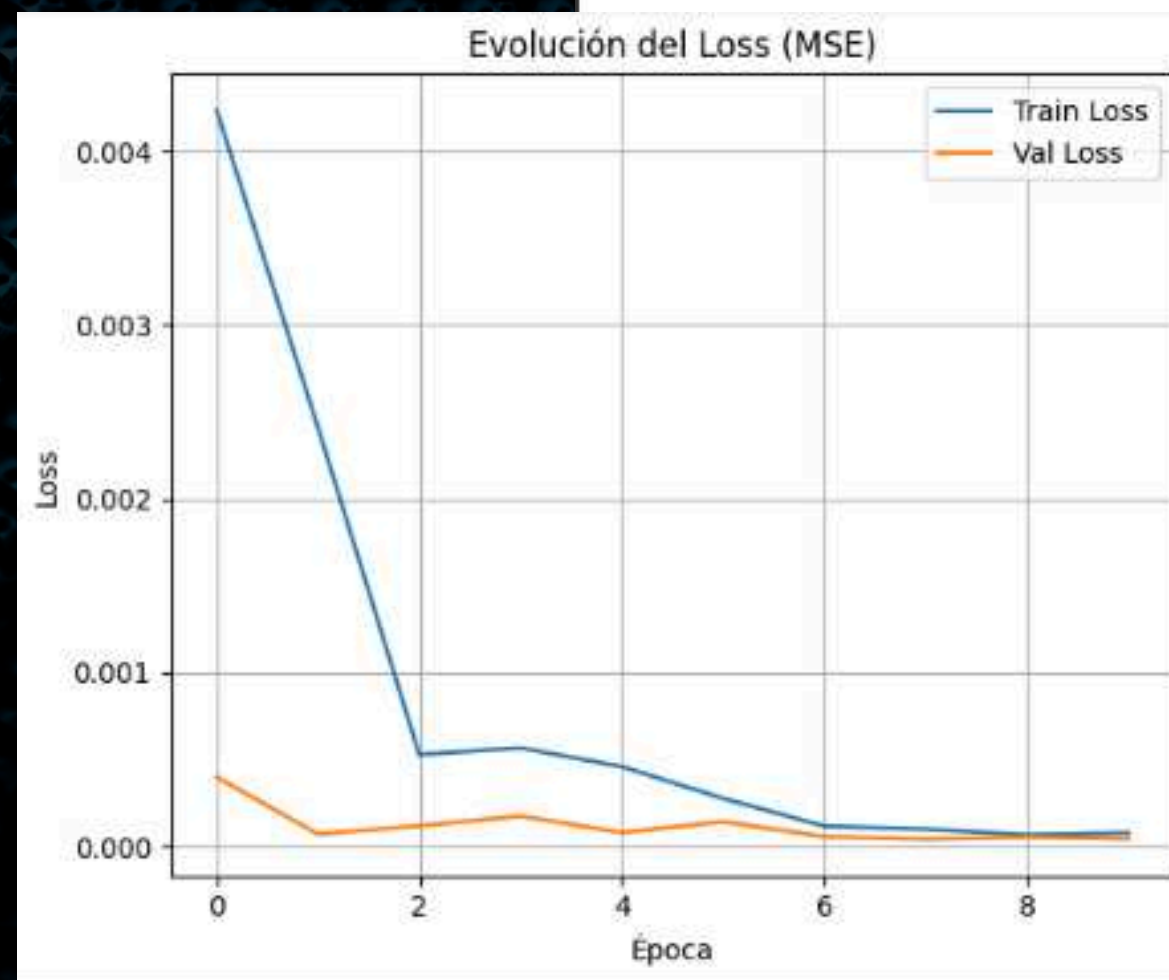
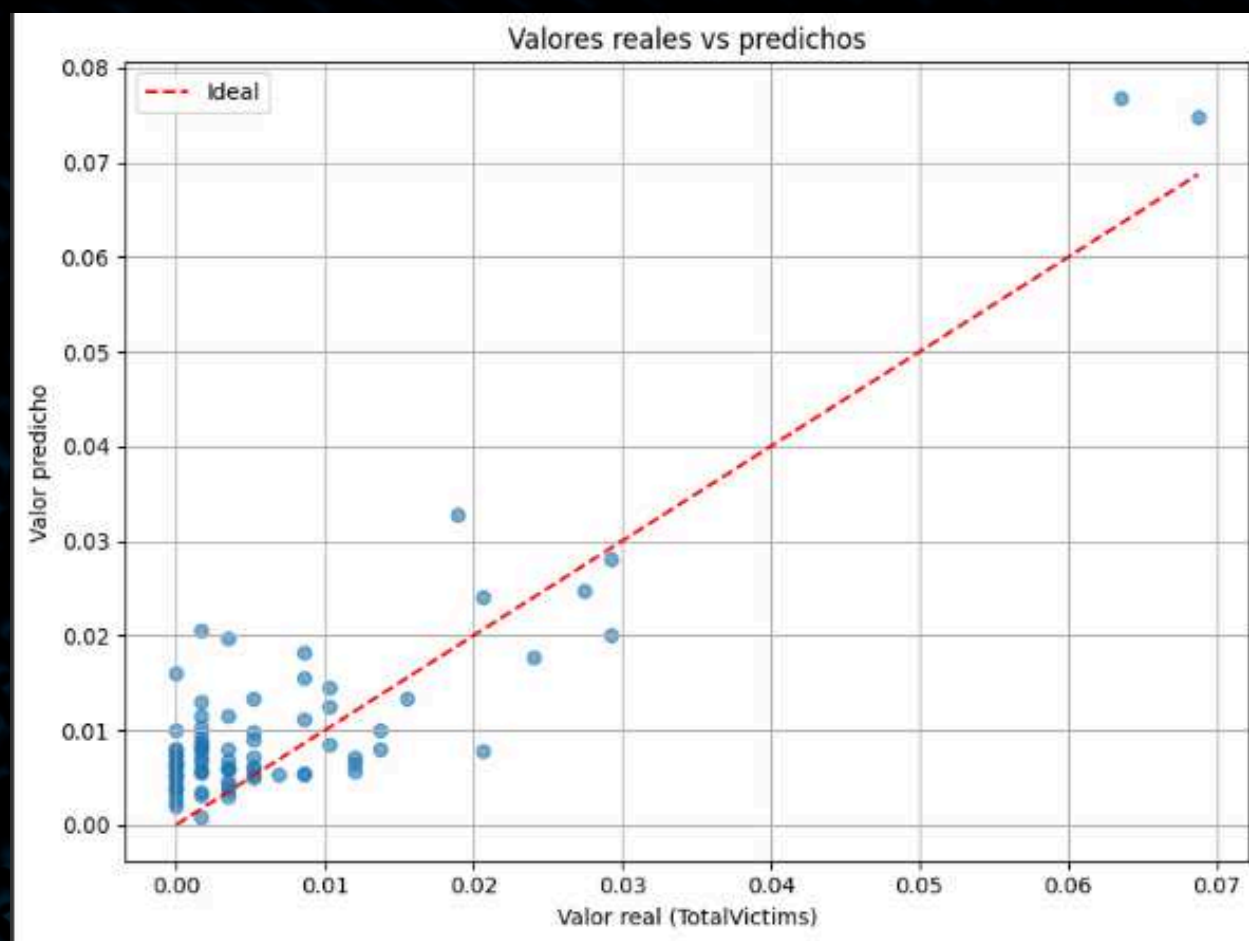
6

**Predicción y visualización de
resultados**

CODIGO

```
9 # Dataset educativo simulado de tiroteos en USA
10 np.random.seed(21)
11
12 # Normalización
13 scaler = MinMaxScaler()
14 scaled_data = scaler.fit_transform(df)
15 df_scaled = pd.DataFrame(scaled_data, columns=df.columns)
16
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Injured'])
19 y = df_scaled['Injured']
20
21 # Dividir en entrenamiento y prueba
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)
23
24 # Construcción del modelo
25 def build_model(input_dim, hidden_layers=6, units=128):
26     model = keras.Sequential()
27     model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
28     for _ in range(hidden_layers):
29         model.add(keras.layers.Dense(units, activation='relu'))
30     model.add(keras.layers.Dense(1)) # Salida única
31     return model
32
33 model = build_model(input_dim=X_train.shape[1], hidden_layers=6)
34
35 # Compilación
36 model.compile(optimizer=keras.optimizers.Adam(),
37               loss='mse',
38               metrics=['mse', 'mae'])
39
40 # Entrenamiento
41 history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)
42
43 # Predicciones
44 y_pred = model.predict(X_test).flatten()
```

```
46 # Gráfico: valores reales vs. predichos
47 plt.figure(figsize=(8, 6))
48 plt.scatter(y_test, y_pred, alpha=0.6)
49 plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
50 plt.xlabel("Valor real (TotalVictims)")
51 plt.ylabel("Valor predicho")
52 plt.title("Valores reales vs predichos")
53 plt.legend()
54 plt.grid(True)
55 plt.tight_layout()
56 plt.show()
57
58 # Gráficos de entrenamiento
59 plt.figure(figsize=(12, 5))
60
61 plt.subplot(1, 2, 1)
62 plt.plot(history.history['loss'], label='Train Loss')
63 plt.plot(history.history['val_loss'], label='Val Loss')
64 plt.title('Evolución del Loss (MSE)')
65 plt.xlabel('Época')
66 plt.ylabel('Loss')
67 plt.legend()
68 plt.grid(True)
69
70 plt.subplot(1, 2, 2)
71 plt.plot(history.history['mae'], label='Train MAE')
72 plt.plot(history.history['val_mae'], label='Val MAE')
73 plt.title('Evolución del MAE')
74 plt.xlabel('Época')
75 plt.ylabel('MAE')
76 plt.legend()
77 plt.grid(True)
78
79 plt.tight_layout()
80 plt.show()
```

Predecir el numero total de heridos

Este código es muy similar al anterior, pero con un pequeño cambio importante: ahora la variable objetivo (y) es Injured (heridos) en lugar de Total victims.

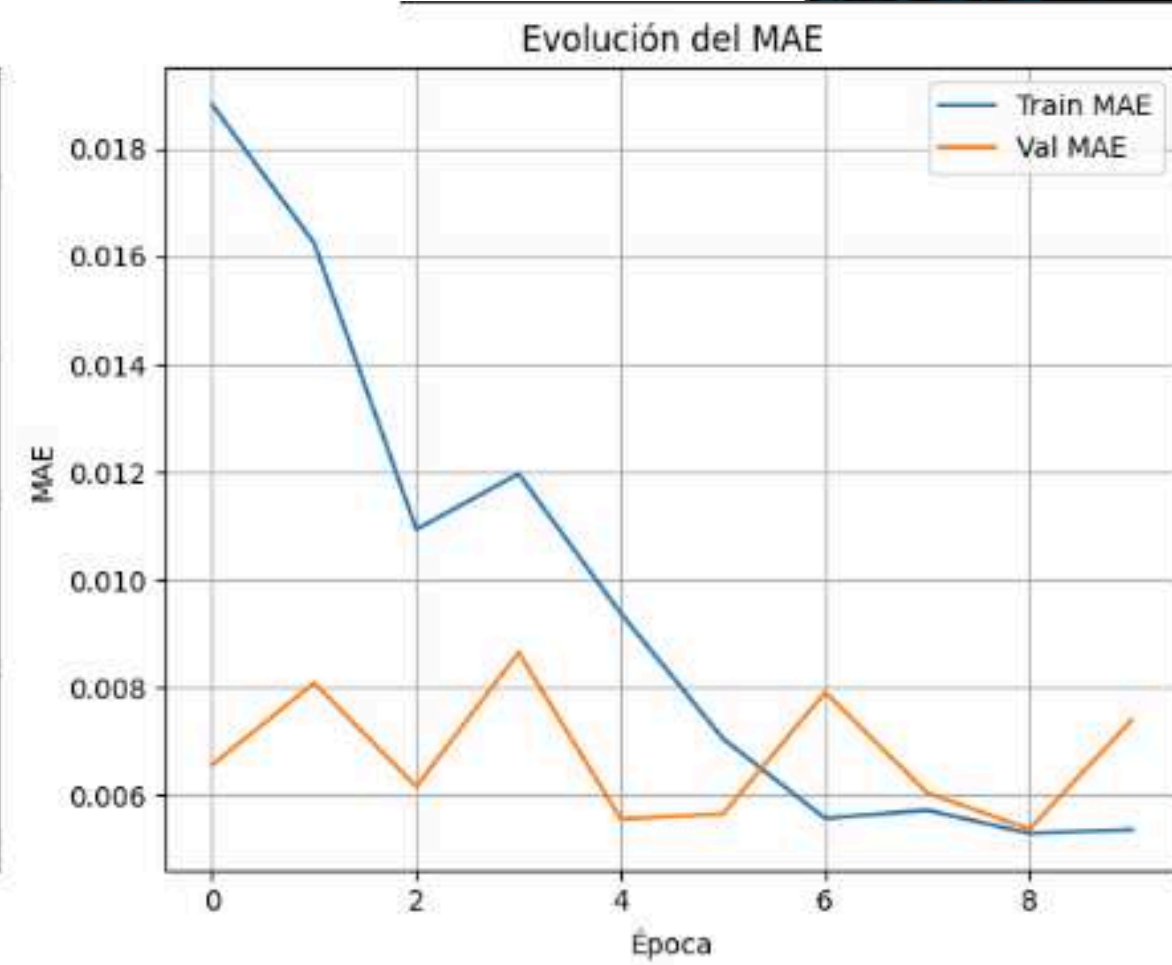
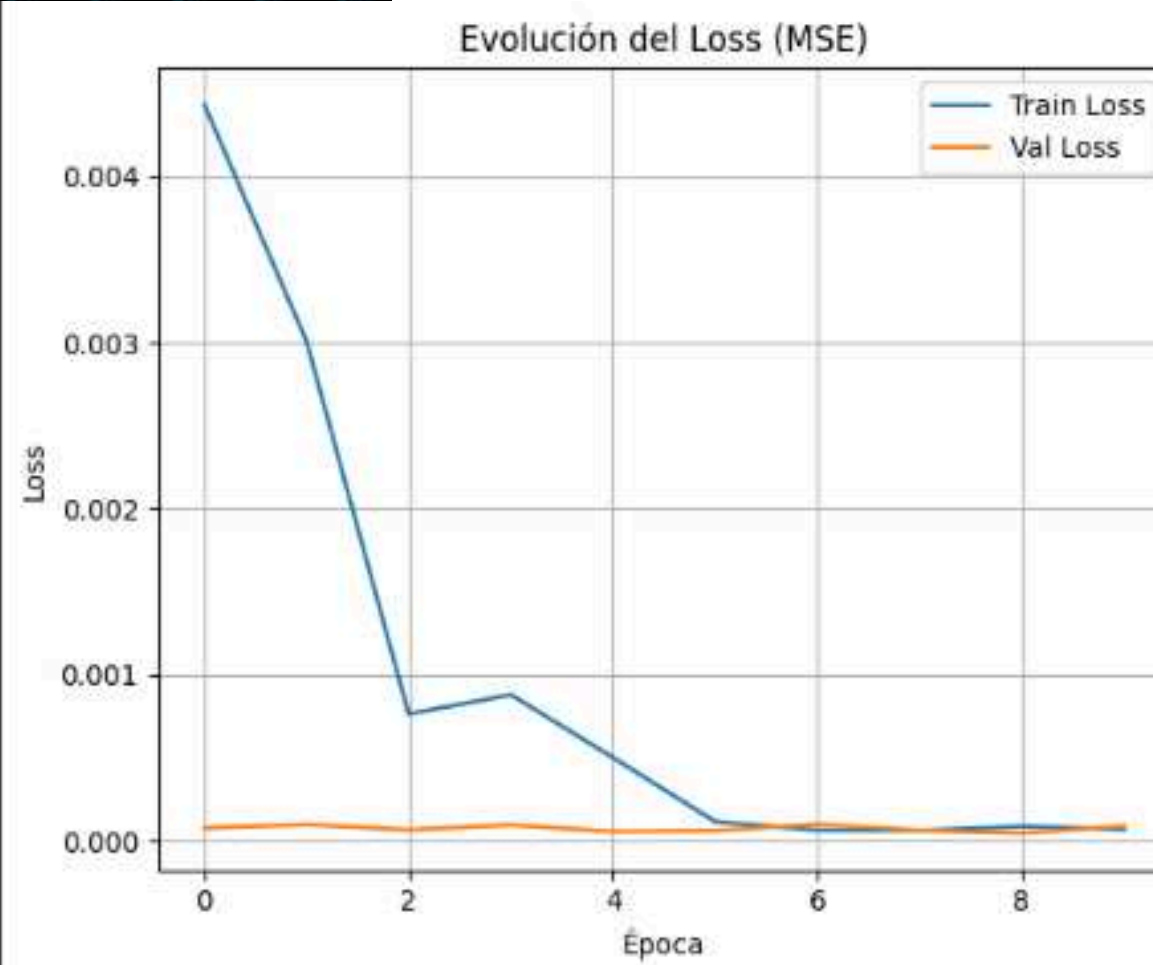
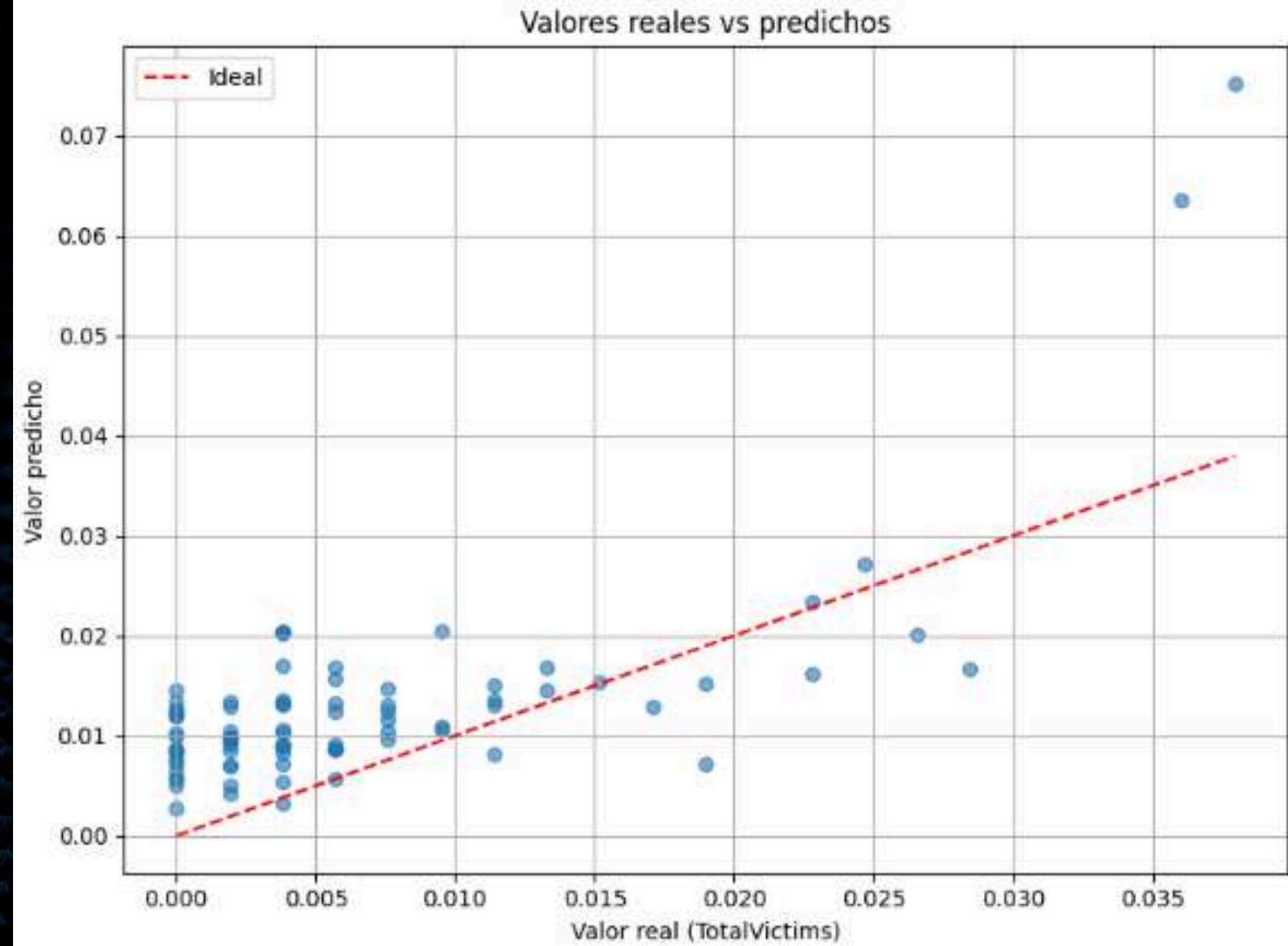
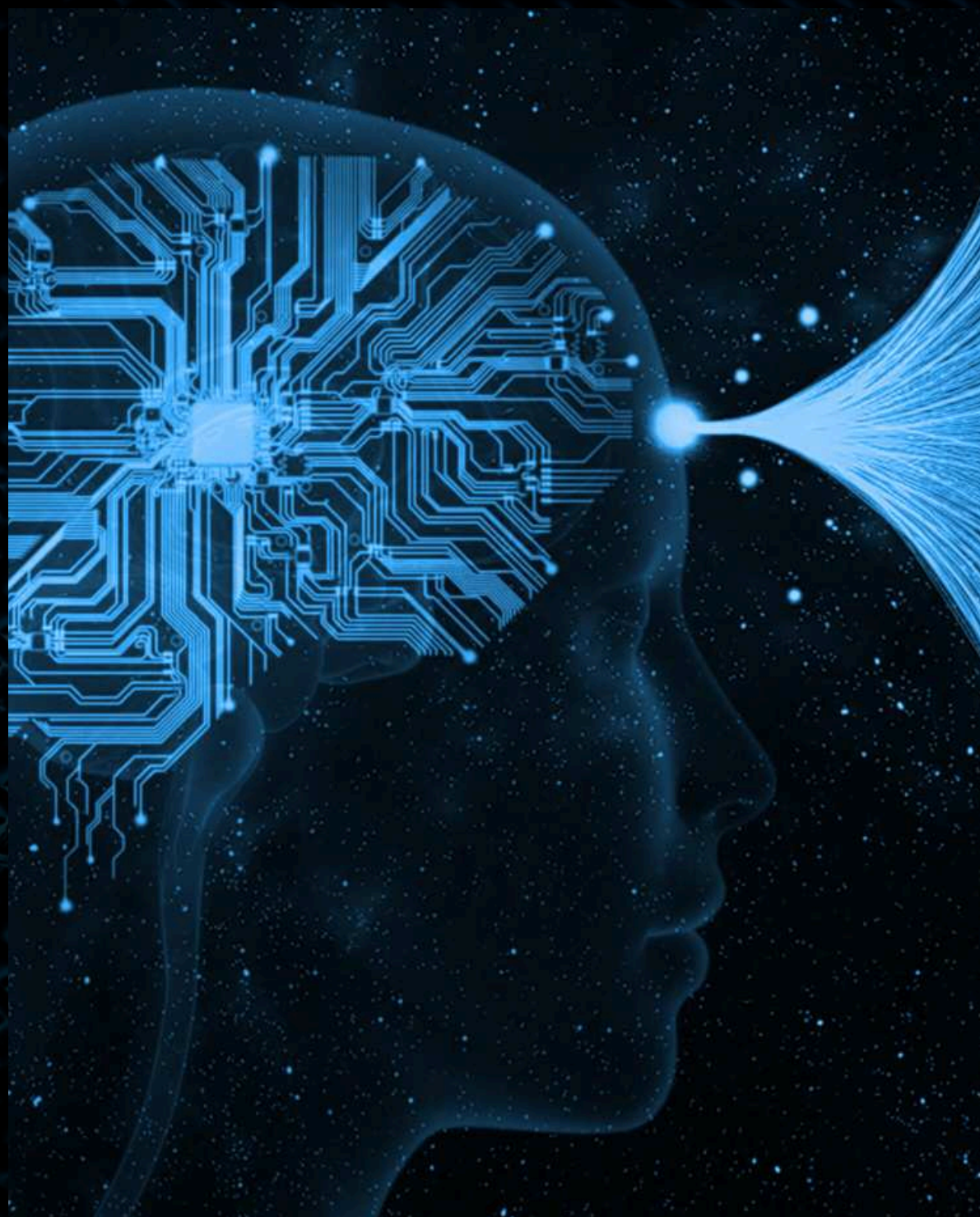
```
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Injured'])
19 y = df_scaled['Injured']
```


CODIGO

```
9 # Dataset educativo simulado de tiroteos en USA
10 np.random.seed(21)
11
12 # Normalización
13 scaler = MinMaxScaler()
14 scaled_data = scaler.fit_transform(df)
15 df_scaled = pd.DataFrame(scaled_data, columns=df.columns)
16
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Injured'])
19 y = df_scaled['Injured']
20
21 # Dividir en entrenamiento y prueba
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)
23
24 # Construcción del modelo
25 def build_model(input_dim, hidden_layers=6, units=128):
26     model = keras.Sequential()
27     model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
28     for _ in range(hidden_layers):
29         model.add(keras.layers.Dense(units, activation='relu'))
30     model.add(keras.layers.Dense(1)) # Salida única
31     return model
32
33 model = build_model(input_dim=X_train.shape[1], hidden_layers=6)
34
35 # Compilación
36 model.compile(optimizer=keras.optimizers.Adam(),
37               loss='mse',
38               metrics=['mse', 'mae'])
39
40 # Entrenamiento
41 history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)
42
43 # Predicciones
44 y_pred = model.predict(X_test).flatten()
```

```
46 # Gráfico: valores reales vs. predichos
47 plt.figure(figsize=(8, 6))
48 plt.scatter(y_test, y_pred, alpha=0.6)
49 plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
50 plt.xlabel("Valor real (TotalVictims)")
51 plt.ylabel("Valor predicho")
52 plt.title("Valores reales vs predichos")
53 plt.legend()
54 plt.grid(True)
55 plt.tight_layout()
56 plt.show()
57
58 # Gráficos de entrenamiento
59 plt.figure(figsize=(12, 5))
60
61 plt.subplot(1, 2, 1)
62 plt.plot(history.history['loss'], label='Train Loss')
63 plt.plot(history.history['val_loss'], label='Val Loss')
64 plt.title("Evolución del Loss (MSE)")
65 plt.xlabel('Época')
66 plt.ylabel('Loss')
67 plt.legend()
68 plt.grid(True)
69
70 plt.subplot(1, 2, 2)
71 plt.plot(history.history['mae'], label='Train MAE')
72 plt.plot(history.history['val_mae'], label='Val MAE')
73 plt.title("Evolución del MAE")
74 plt.xlabel('Época')
75 plt.ylabel('MAE')
76 plt.legend()
77 plt.grid(True)
78
79 plt.tight_layout()
80 plt.show()
```





Predecir el numero total de fallecidos

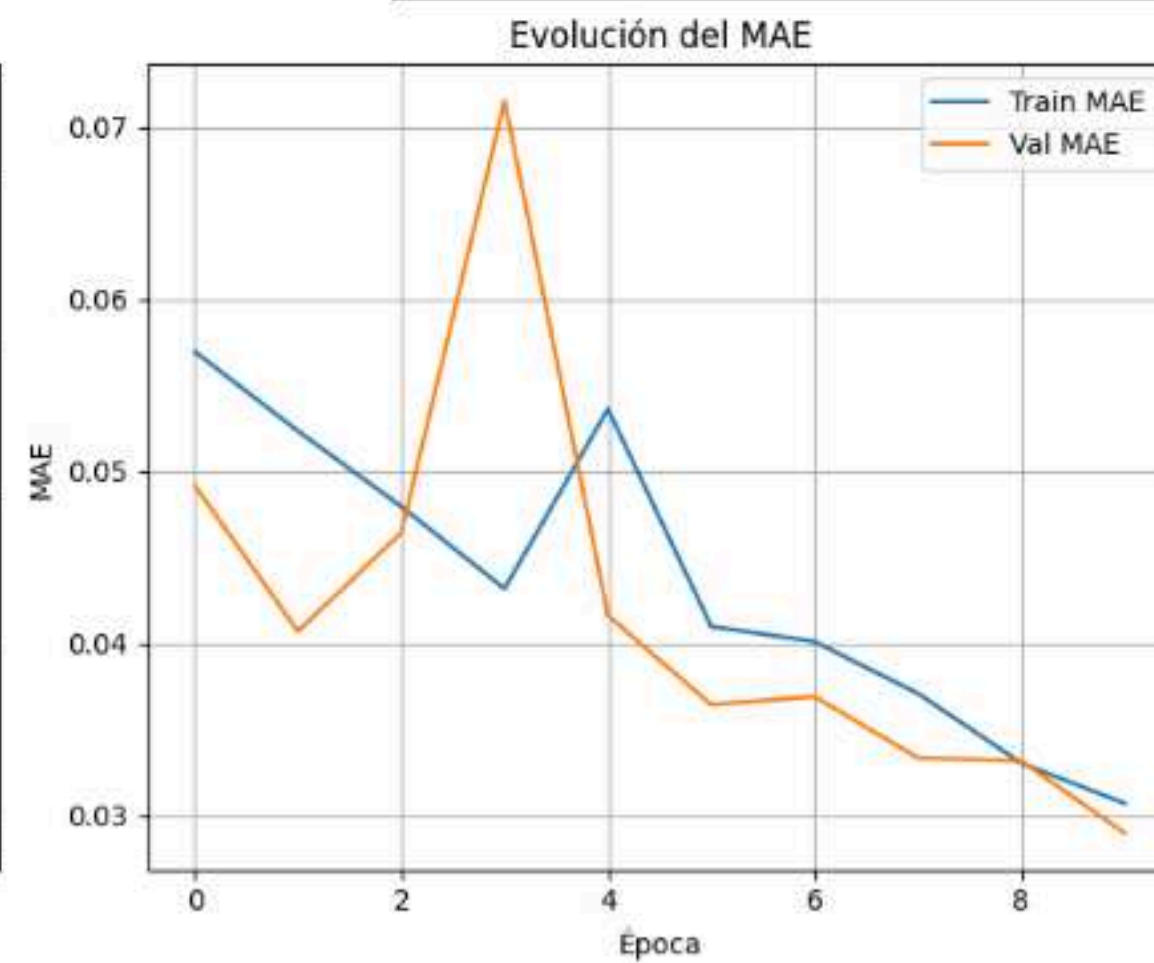
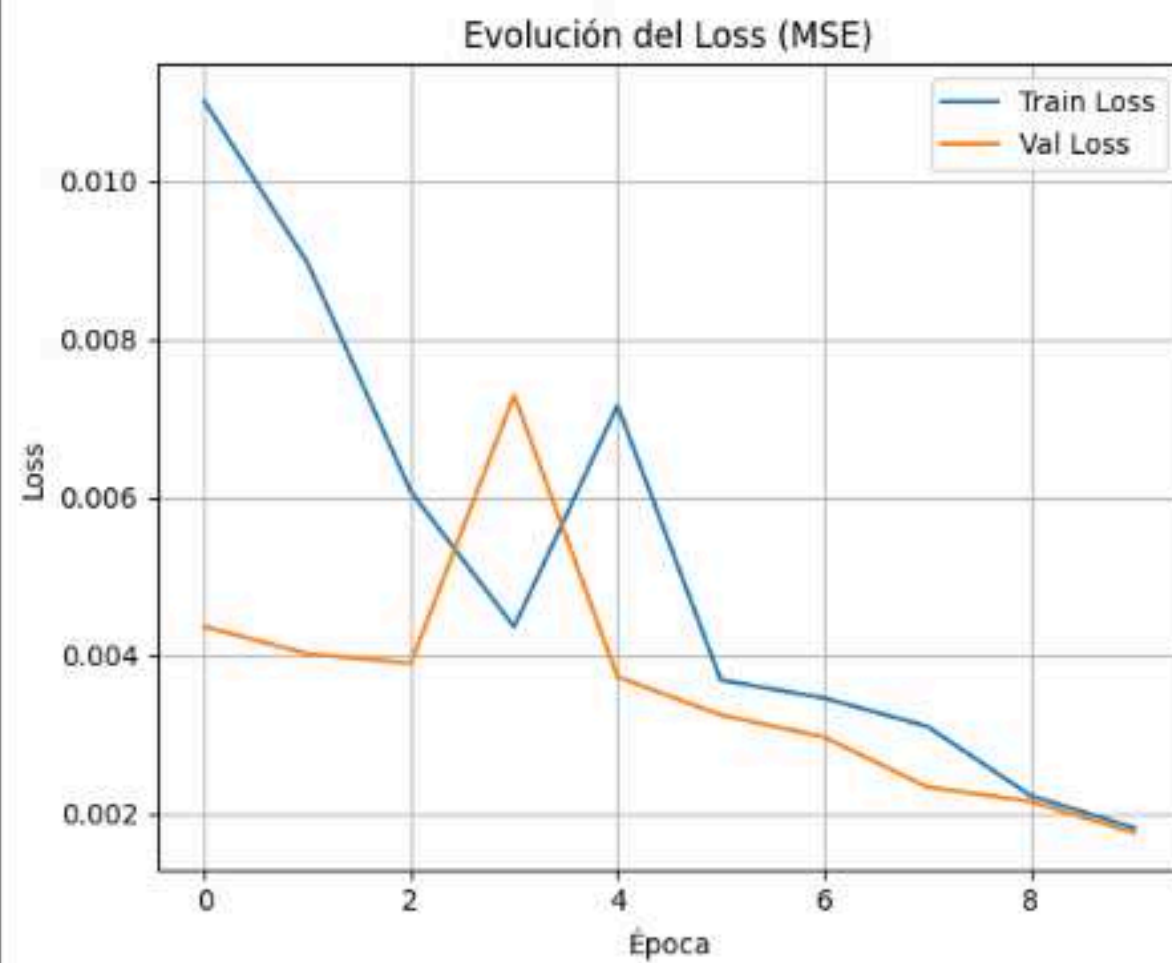
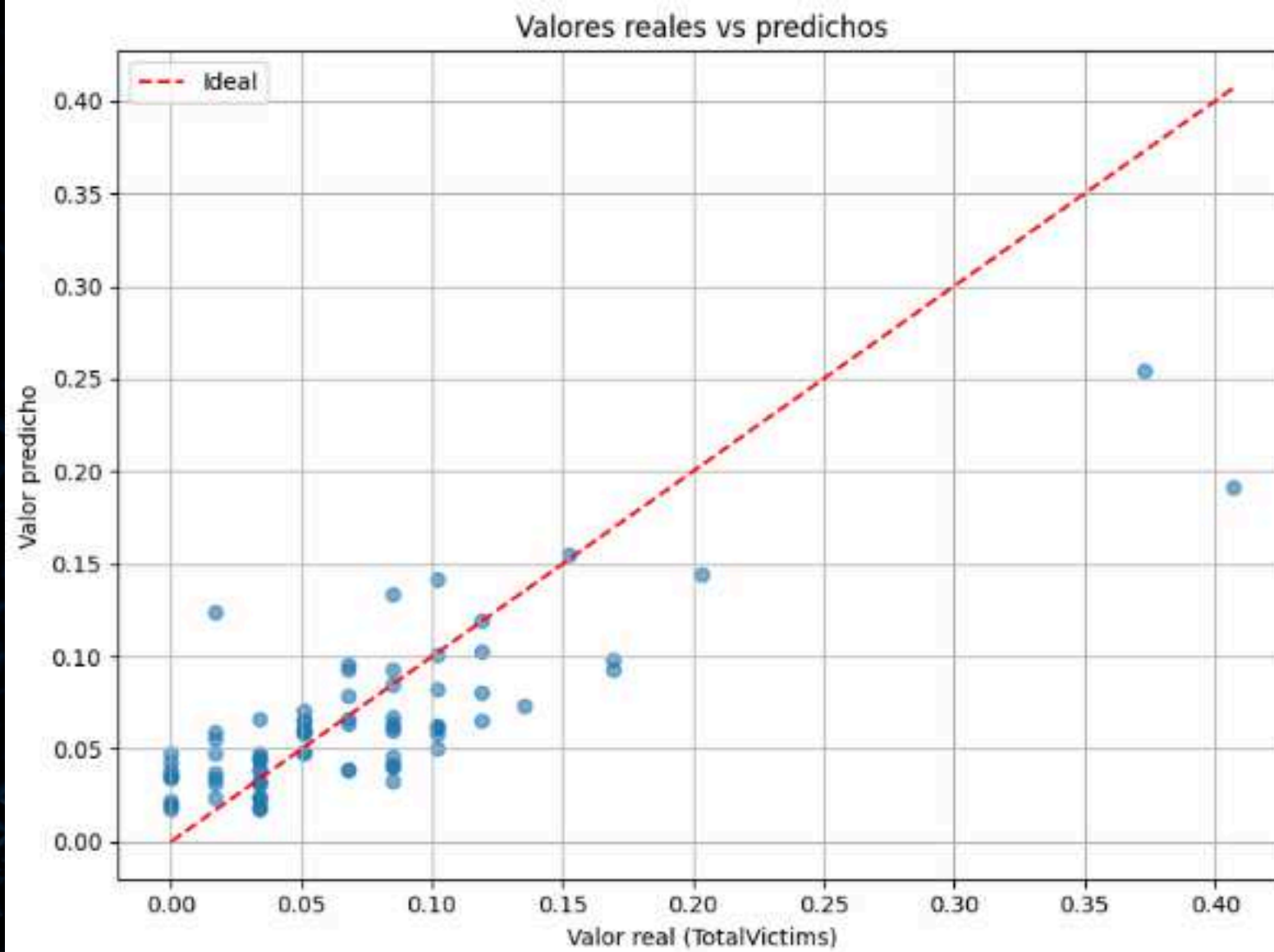
Este código es prácticamente igual a los anteriores, pero ahora estás entrenando una red neuronal para predecir la cantidad de fallecidos (Fatalities) en un conjunto de datos de tiroteos simulados en EE.UU.

```
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Fatalities'])
19 y = df_scaled['Fatalities']
20
```


CODIGO

```
9 # Dataset educativo simulado de tiroteos en USA
10 np.random.seed(21)
11
12 # Normalización
13 scaler = MinMaxScaler()
14 scaled_data = scaler.fit_transform(df)
15 df_scaled = pd.DataFrame(scaled_data, columns=df.columns)
16
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Fatalities'])
19 y = df_scaled['Fatalities']
20
21 # Dividir en entrenamiento y prueba
22 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)
23
24 # Construcción del modelo
25 def build_model(input_dim, hidden_layers=6, units=128):
26     model = keras.Sequential()
27     model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
28     for _ in range(hidden_layers):
29         model.add(keras.layers.Dense(units, activation='relu'))
30     model.add(keras.layers.Dense(1)) # Salida única
31     return model
32
33 model = build_model(input_dim=X_train.shape[1], hidden_layers=6)
34
35 # Compilación
36 model.compile(optimizer=keras.optimizers.Adam(),
37               loss='mse',
38               metrics=['mse', 'mae'])
39
40 # Entrenamiento
41 history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)
42
43 # Predicciones
44 y_pred = model.predict(X_test).flatten()
```

```
46 # Gráfico: valores reales vs. predichos
47 plt.figure(figsize=(8, 6))
48 plt.scatter(y_test, y_pred, alpha=0.6)
49 plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
50 plt.xlabel("Valor real (TotalVictims)")
51 plt.ylabel("Valor predicho")
52 plt.title("valores reales vs predichos")
53 plt.legend()
54 plt.grid(True)
55 plt.tight_layout()
56 plt.show()
57
58 # Gráficos de entrenamiento
59 plt.figure(figsize=(12, 5))
60
61 plt.subplot(1, 2, 1)
62 plt.plot(history.history['loss'], label='Train Loss')
63 plt.plot(history.history['val_loss'], label='Val Loss')
64 plt.title('Evolución del Loss (MSE)')
65 plt.xlabel('Época')
66 plt.ylabel('Loss')
67 plt.legend()
68 plt.grid(True)
69
70 plt.subplot(1, 2, 2)
71 plt.plot(history.history['mae'], label='Train MAE')
72 plt.plot(history.history['val_mae'], label='Val MAE')
73 plt.title('Evolución del MAE')
74 plt.xlabel('Época')
75 plt.ylabel('MAE')
76 plt.legend()
77 plt.grid(True)
78
79 plt.tight_layout()
80 plt.show()
81
```

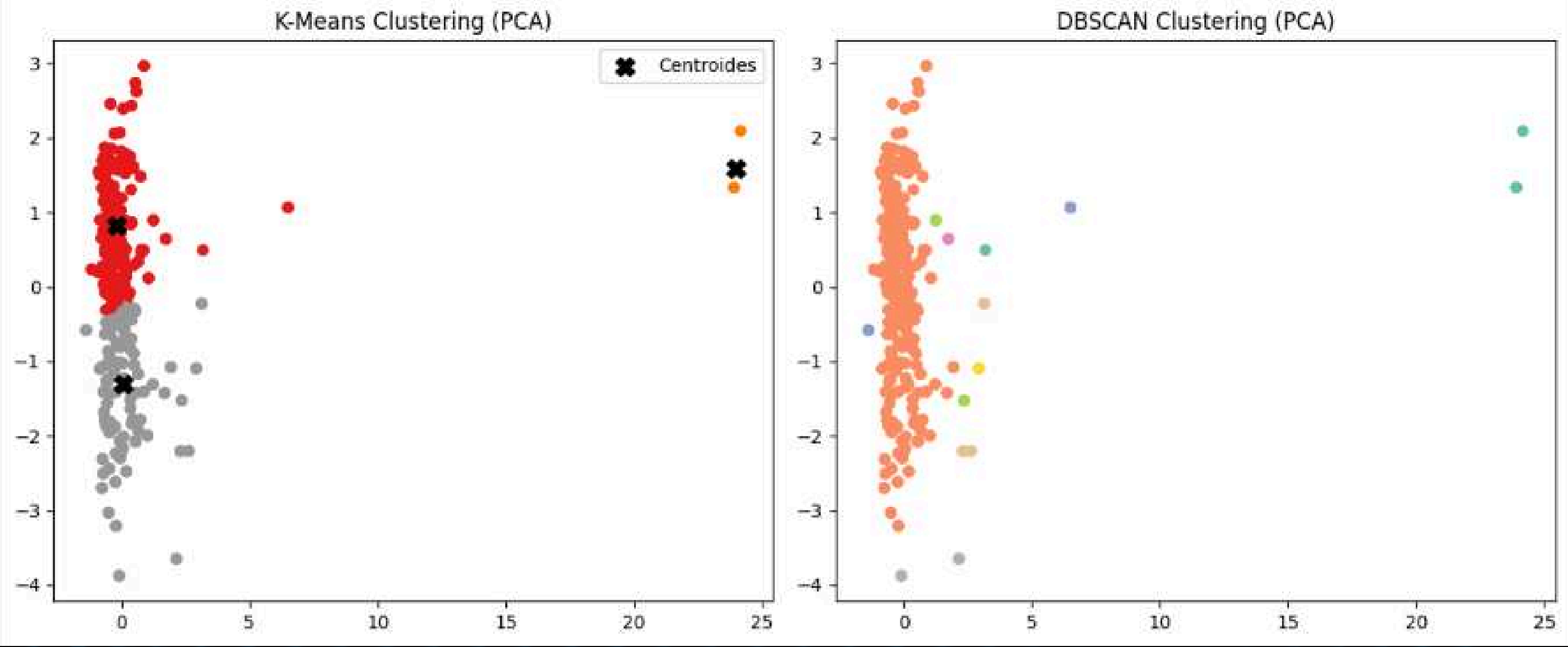
ANÁLISIS DE CLUSTERING

**Escalamiento
de datos**

**Reducción de
dimensiones
con PCA**

**Clustering con
K-Means**

**Clustering con
DBSCAN**





**THANK
YOU**