INTELIGENCIA ARTIFICIAL

Tercera Entrega

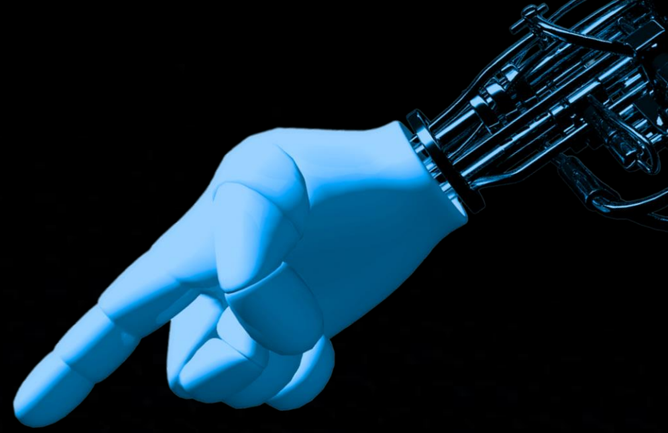
TIROTEOS EN USA

Proyecto realizado por:
Wilmer Andres Romero Cala - 2214102
Jhonalber David Rangel Medina - 2225510

Predecir el numero total de victimas

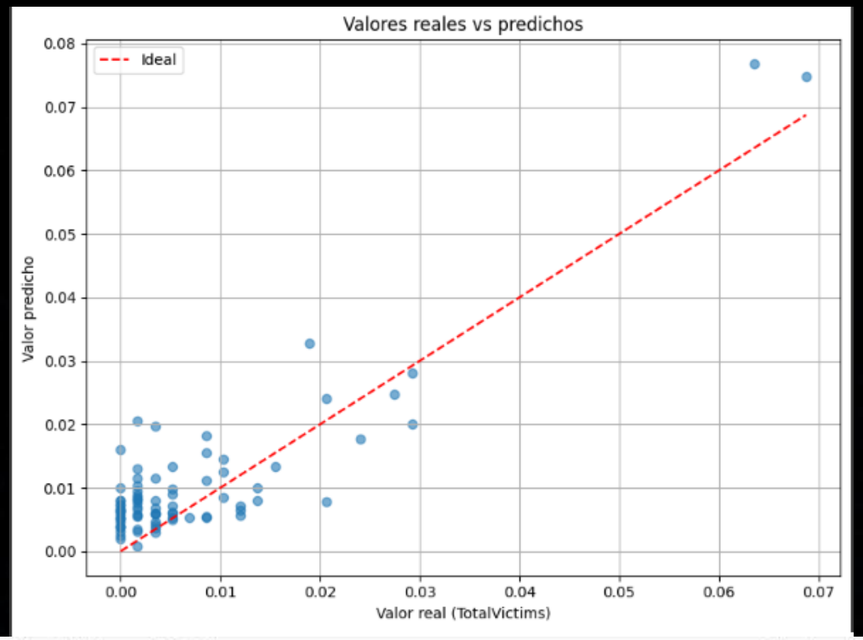
1. Sim ulación y no m alización
2. Selección de variables
3. División en entrenamiento y prueba
4. Definición del modelo
5. Compilación
6. Predicción y visualización de resultados

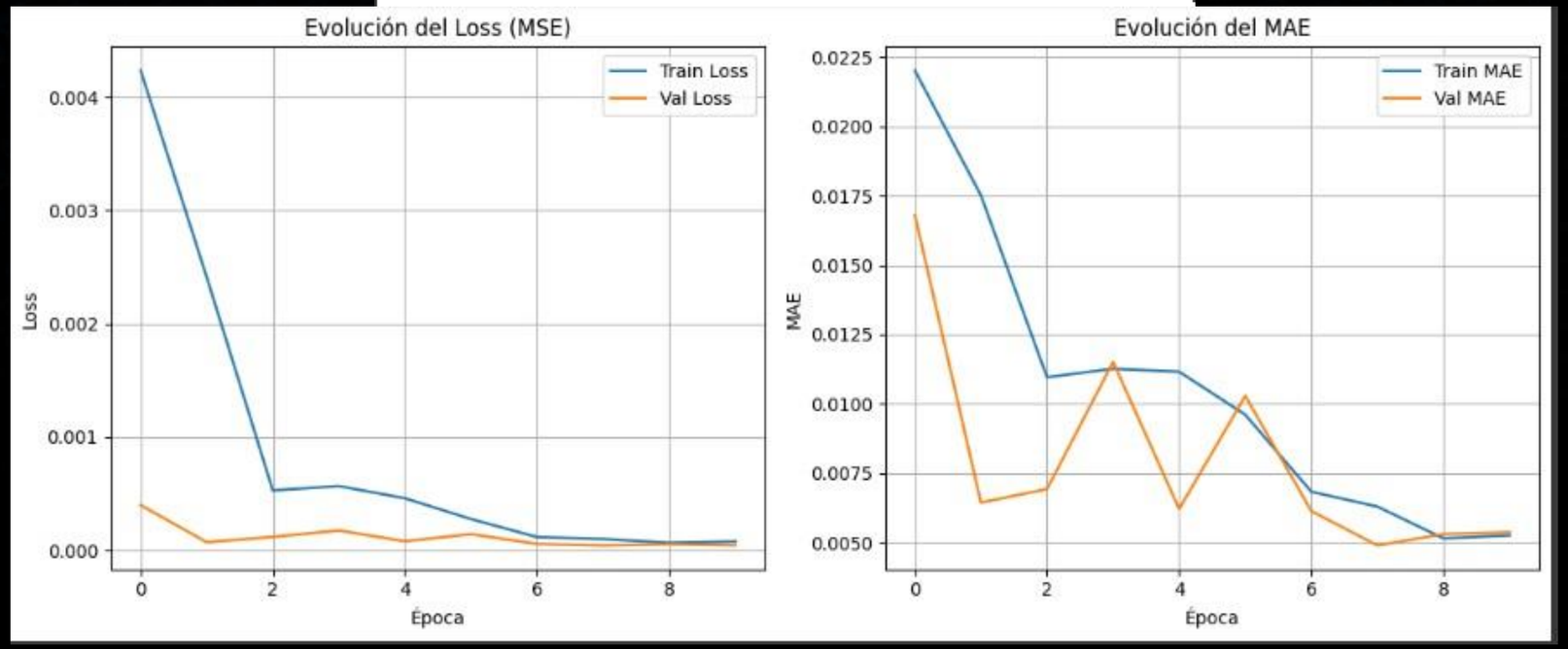
CODIGO



```python
# Dataset educativo simulado de tiroteos en USA
np.random.seed(21)

# Normalización
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)

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

# Dividir en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)

# Construcción del modelo
def build_model(input_dim, hidden_layers=6, units=128):
    model = keras.Sequential()
    model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
    for _ in range(hidden_layers):
        model.add(keras.layers.Dense(units, activation='relu'))
    model.add(keras.layers.Dense(1))  # Salida única
    return model

model = build_model(input_dim=X_train.shape[1], hidden_layers=6)

# Compilación
model.compile(optimizer=keras.optimizers.Adam(),
              loss='mse',
              metrics=['mse', 'mae'])

# Entrenamiento
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)

# Predicciones
y_pred = model.predict(X_test).flatten()
```

```python
# Gráfico: valores reales vs. predichos
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
plt.xlabel("Valor real (TotalVictims)")
plt.ylabel("Valor predicho")
plt.title("Valores reales vs predichos")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Gráficos de entrenamiento
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Evolución del Loss (MSE)')
plt.xlabel('Época')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Val MAE')
plt.title('Evolución del MAE')
plt.xlabel('Época')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```


Valores reales vs predichos
Ideal
Valor predicho
Valor real (TotalVictims)
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08


Evolución del Loss (MSE)
Train Loss
Val Loss
Loss
Época
Evolución del MAE
Train MAE
Val MAE
MAE
Época

Predecir el numero total de heridos

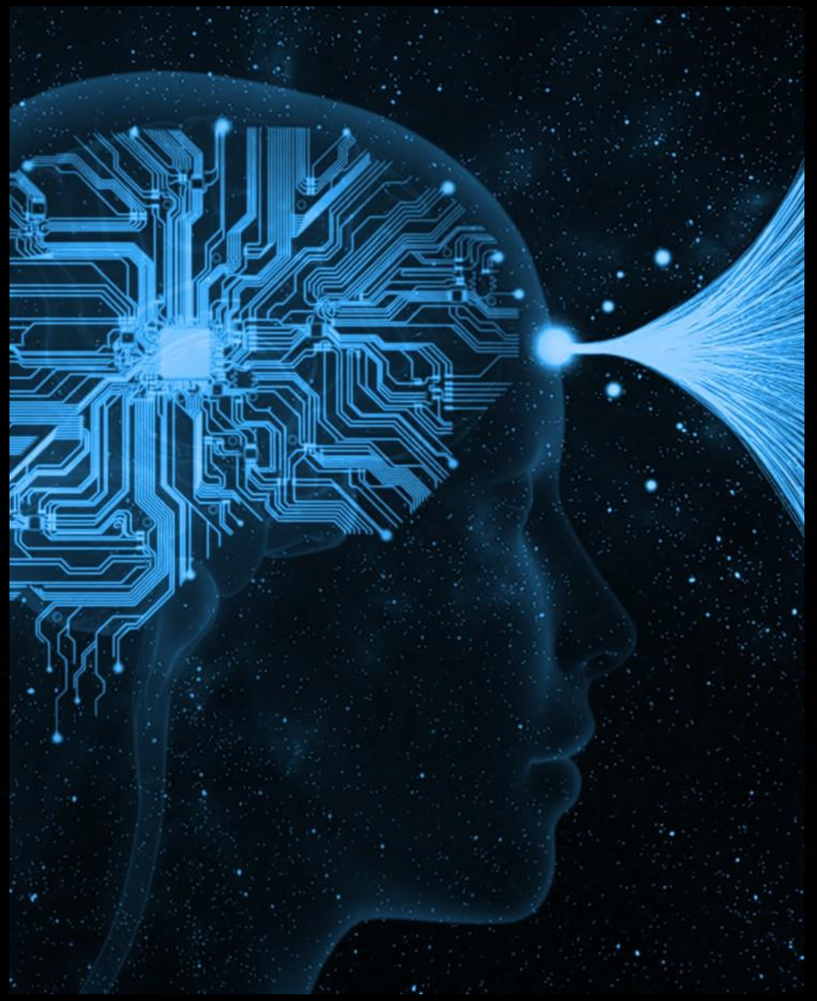
Este código es muy similar al anterior, pero con un pequeño cambio importante: ahora la variable objetivo (y) es Injured (heridos) en lugar de Total victims.

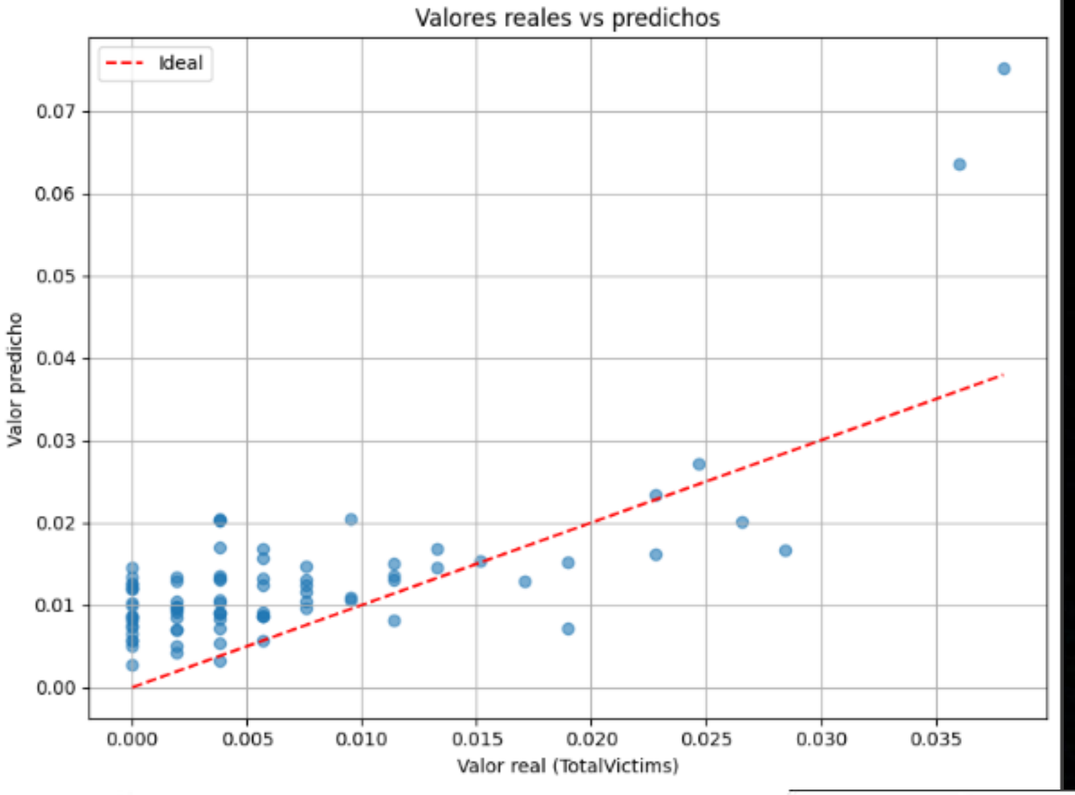
```
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Injured'])
19 y = df_scaled['Injured']
```

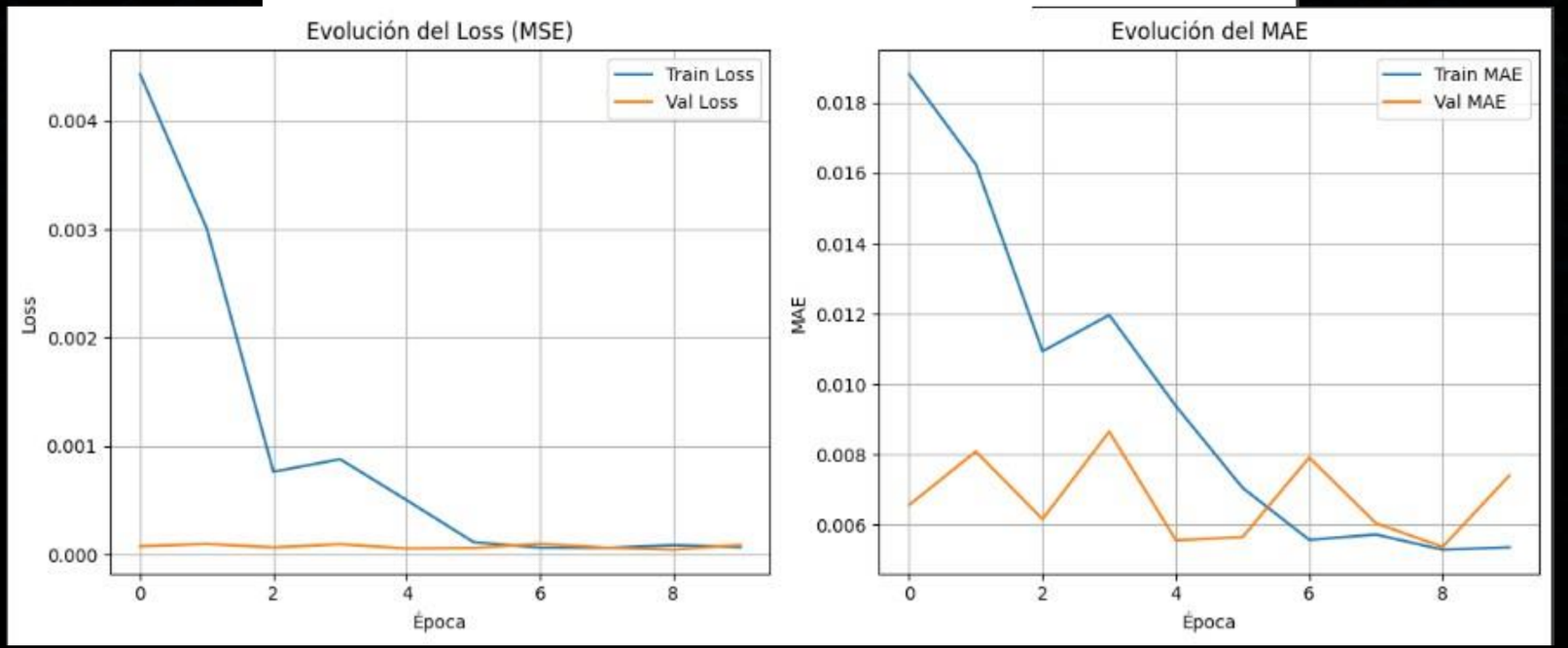
CODIGO

```python
# Dataset educativo simulado de tiroteos en USA
np.random.seed(21)

# Normalización
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)

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

# Dividir en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)

# Construcción del modelo
def build_model(input_dim, hidden_layers=6, units=128):
    model = keras.Sequential()
    model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
    for _ in range(hidden_layers):
        model.add(keras.layers.Dense(units, activation='relu'))
    model.add(keras.layers.Dense(1))  # Salida única
    return model

model = build_model(input_dim=X_train.shape[1], hidden_layers=6)

# Compilación
model.compile(optimizer=keras.optimizers.Adam(),
              loss='mse',
              metrics=['mse', 'mae'])

# Entrenamiento
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)

# Predicciones
y_pred = model.predict(X_test).flatten()
```

```python
# Gráfico: valores reales vs. predichos
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
plt.xlabel("Valor real (TotalVictims)")
plt.ylabel("Valor predicho")
plt.title("Valores reales vs predichos")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Gráficos de entrenamiento
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Evolución del Loss (MSE)')
plt.xlabel('Época')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Val MAE')
plt.title('Evolución del MAE')
plt.xlabel('Época')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```




Valores reales vs predichos
Ideal
Valor predicho
Valor real (TotalVictims)
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035


Evolución del Loss (MSE)
Train Loss
Val Loss
Loss
Época
0.000
0.001
0.002
0.003
0.004
0
2
4
6
8
Evolución del MAE
Train MAE
Val MAE
MAE
Época
0.006
0.008
0.010
0.012
0.014
0.016
0.018

Predecir el numero total de fallecidos

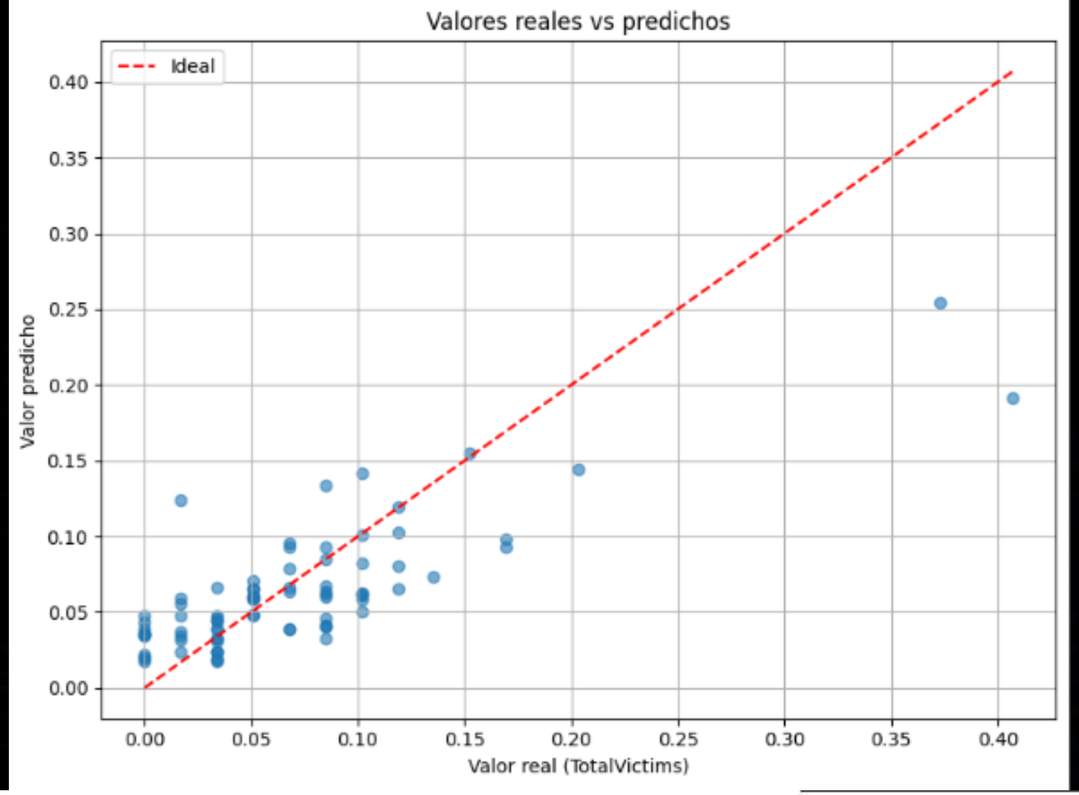
Este código es prácticamente igual a los anteriores, pero ahora estás entrenando una red neuronal para predecir la cantidad de fallecidos (Fatalities) en un conjunto de datos de tiroteos simulados en EE.UU.

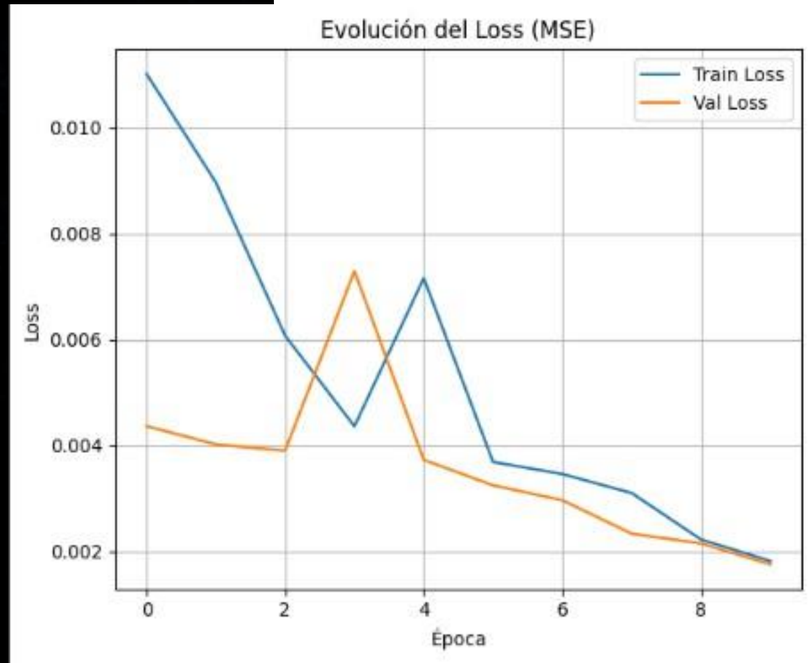
```
17 # Separar variables predictoras y target
18 X = df_scaled.drop(columns=['Fatalities'])
19 y = df_scaled['Fatalities']
```

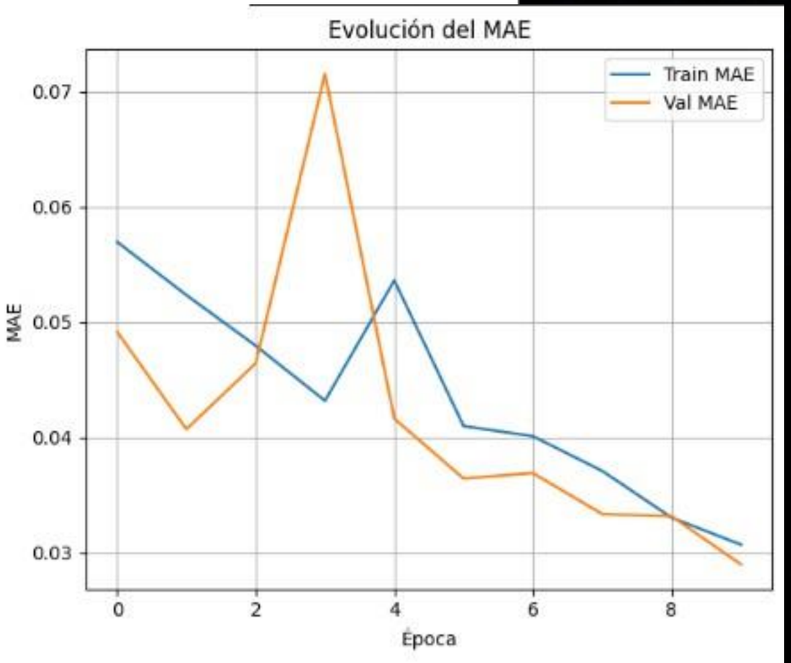
CODIGO

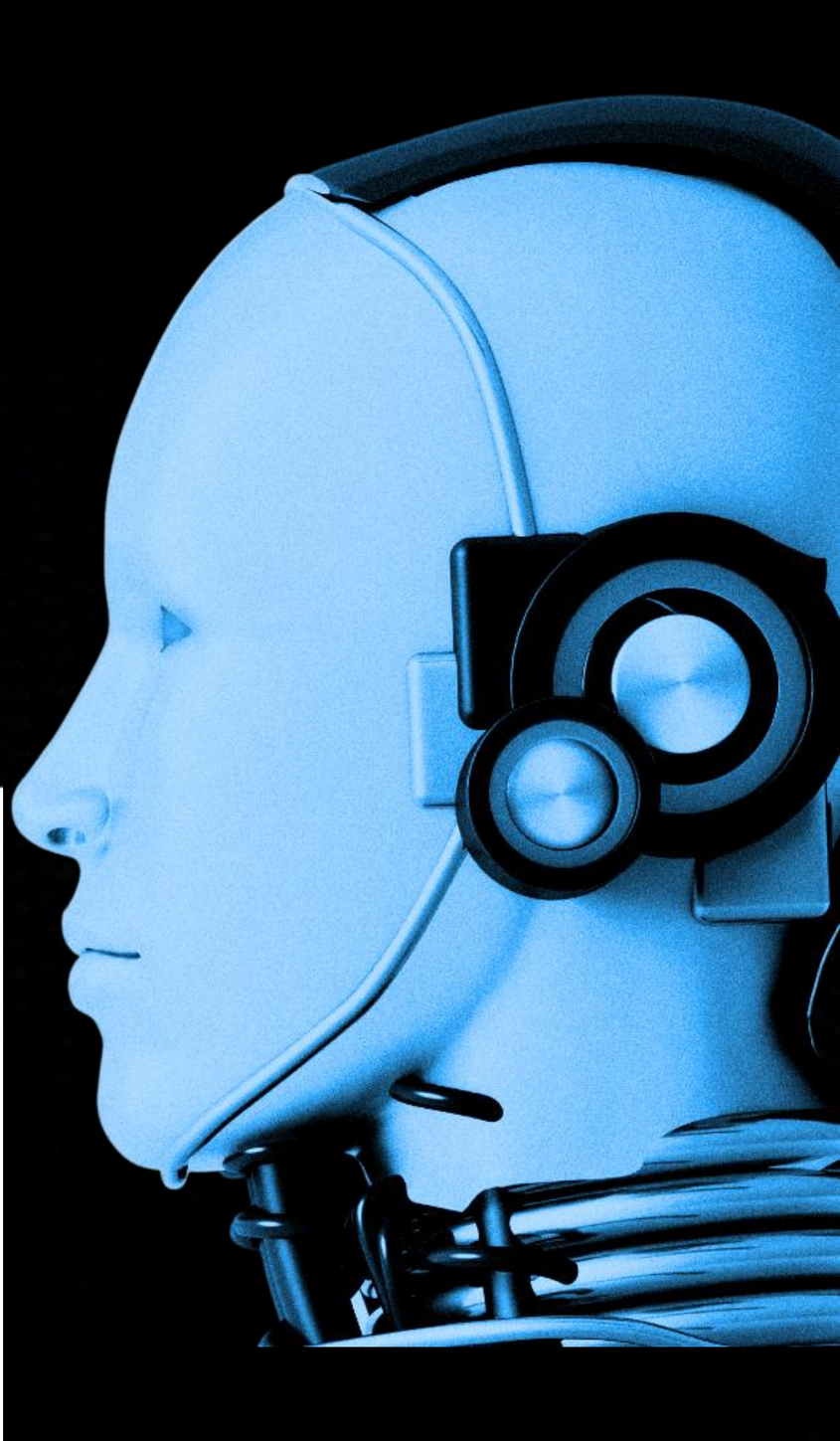
```python
# Dataset educativo simulado de tiroteos en USA
np.random.seed(21)

# Normalización
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
df_scaled = pd.DataFrame(scaled_data, columns=df.columns)

# Separar variables predictoras y target
X = df_scaled.drop(columns=['Fatalities'])
y = df_scaled['Fatalities']

# Dividir en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=21)

# Construcción del modelo
def build_model(input_dim, hidden_layers=6, units=128):
    model = keras.Sequential()
    model.add(keras.layers.InputLayer(input_shape=(input_dim,)))
    for _ in range(hidden_layers):
        model.add(keras.layers.Dense(units, activation='relu'))
    model.add(keras.layers.Dense(1))  # Salida única
    return model

model = build_model(input_dim=X_train.shape[1], hidden_layers=6)

# Compilación
model.compile(optimizer=keras.optimizers.Adam(),
              loss='mse',
              metrics=['mse', 'mae'])

# Entrenamiento
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test), verbose=1)

# Predicciones
y_pred = model.predict(X_test).flatten()
```

```python
# Gráfico: valores reales vs. predichos
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label="Ideal")
plt.xlabel("Valor real (TotalVictims)")
plt.ylabel("Valor predicho")
plt.title("Valores reales vs predichos")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Gráficos de entrenamiento
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Evolución del Loss (MSE)')
plt.xlabel('Época')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Val MAE')
plt.title('Evolución del MAE')
plt.xlabel('Época')
plt.ylabel('MAE')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```


Valores reales vs predichos
Ideal
Valor predicho
Valor real (TotalVictims)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40


Evolución del Loss (MSE)
Train Loss
Val Loss
Loss
Época
0.002
0.004
0.006
0.008
0.010
0
2
4
6
8


Evolución del MAE
Train MAE
Val MAE
MAE
Época
0.03
0.04
0.05
0.06
0.07
0
2
4
6
8



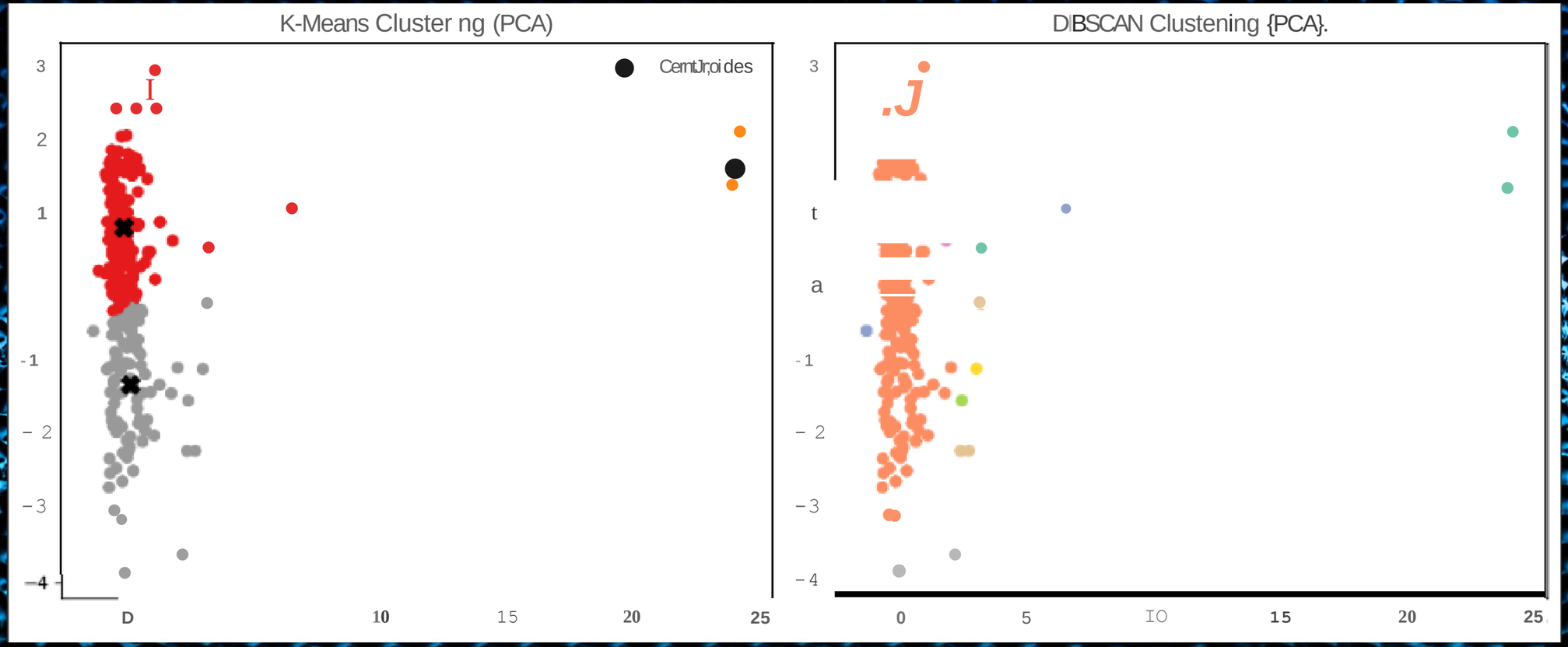
ANÁLISIS DE CLUSTERING

Escalamiento de datos

Reducción de dimensiones con PCA

Clustering con K-Means

Clustering con DBSCAN


K-Means Cluster ng (PCA)
CemtJr;oi des
3
2
1
-1
- 2
-3
-4
D
10
15
20
25
DIBSCAN Clustening {PCA}.
3
t
a
-1
- 2
-3
- 4
0
5
IO
15
20
25


THANK
YOU