# Laboratorio 2 - Deep Learning Series

**Cesar Lopez # 22535**

- Link del repositorio: Laboratorio 2 - DS

In [30]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import itertools
```

In [31]:
```
df = pd.read_csv("./data/importacion.csv", parse_dates=["Fecha"])
df.set_index("Fecha", inplace=True)
```

In [32]:
```
# Preparacion gasolina regular
regular_series = df["Gasolina regular"]

regular_split = int(len(regular_series) * 0.7)
regular_train = regular_series.iloc[:regular_split]
regular_test = regular_series.iloc[regular_split:]

regular_scaler = MinMaxScaler()
regular_train_scaled = regular_scaler.fit_transform(regular_train.values.reshape(-1
regular_test_scaled = regular_scaler.transform(regular_test.values.reshape(-1, 1))
```

In [33]:
```
# Preparacion gasolina superior
super_series = df["Gasolina superior"]

super_split = int(len(regular_series) * 0.7)
super_train = super_series.iloc[:super_split]
super_test = super_series.iloc[super_split:]

super_scaler = MinMaxScaler()
super_train_scaled = super_scaler.fit_transform(super_train.values.reshape(-1, 1))
super_test_scaled = super_scaler.transform(super_test.values.reshape(-1, 1))
```

In [34]:
```
# Preparacion gasolina diesel
diesel_series = df["Diesel alto azufre"]

diesel_split = int(len(diesel_series) * 0.7)
diesel_train = diesel_series.iloc[:diesel_split]
diesel_test = diesel_series.iloc[diesel_split:]

diesel_scaler = MinMaxScaler()
diesel_train_scaled = diesel_scaler.fit_transform(diesel_train.values.reshape(-1, 1
diesel_test_scaled = diesel_scaler.transform(diesel_test.values.reshape(-1,1))
```

In [35]:
```
def create_sequences(data, window=12):
    X, y = [], []
    for i in range(window, len(data)):
        X.append(data[i-window:i])
        y.append(data[i])
    return np.array(X), np.array(y)
```

In [36]:
```
# Para Gasolina regular
window_size = 12
X_reg_train, y_reg_train = create_sequences(regular_train_scaled, window=window_siz
X_reg_test,  y_reg_test  = create_sequences(regular_test_scaled,  window=window_siz

# Para Gasolina superior
X_sup_train, y_sup_train = create_sequences(super_train_scaled, window=window_size)
X_sup_test,  y_sup_test  = create_sequences(super_test_scaled,  window=window_size)

# Para Gasolina diesel
x_dis_train, y_dis_train = create_sequences(diesel_train_scaled, window=window_size
x_dis_test, y_dis_test = create_sequences(diesel_test_scaled, window=window_size)
```

In [37]:
```
# Redimensiona al formato [muestras, timesteps, features]
# Regular
X_reg_train = X_reg_train.reshape(-1, window_size, 1)
X_reg_test  = X_reg_test.reshape(-1,  window_size, 1)

# Superior
X_sup_train = X_sup_train.reshape(-1, window_size, 1)
X_sup_test  = X_sup_test.reshape(-1,  window_size, 1)

# Diesel
x_dis_train = x_dis_train.reshape(-1, window_size, 1)
x_dis_test = x_dis_test.reshape(-1, window_size, 1)
```

In [38]:
```
# Convierte NumPy → Torch Tensor

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Gasolina regular
X_reg_train_t = torch.tensor(X_reg_train, dtype=torch.float32).to(device)
y_reg_train_t = torch.tensor(y_reg_train, dtype=torch.float32).to(device)
X_reg_test_t  = torch.tensor(X_reg_test,  dtype=torch.float32).to(device)
y_reg_test_t  = torch.tensor(y_reg_test,  dtype=torch.float32).to(device)

# Gasolina superior
X_sup_train_t = torch.tensor(X_sup_train, dtype=torch.float32).to(device)
y_sup_train_t = torch.tensor(y_sup_train, dtype=torch.float32).to(device)
X_sup_test_t  = torch.tensor(X_sup_test,  dtype=torch.float32).to(device)
y_sup_test_t  = torch.tensor(y_sup_test,  dtype=torch.float32).to(device)

# Gasolina diesel
x_dis_train_t = torch.tensor(x_dis_train, dtype=torch.float32).to(device)
y_dis_train_t = torch.tensor(y_dis_train, dtype=torch.float32).to(device)
x_dis_test_t = torch.tensor(x_dis_test, dtype=torch.float32).to(device)
y_dis_test_t = torch.tensor(y_dis_test, dtype=torch.float32).to(device)
```

In [39]:
```
# Dataset y DataLoader
batch_size = 32

# Gasolina regular
train_ds = TensorDataset(X_reg_train_t, y_reg_train_t)
test_ds  = TensorDataset(X_reg_test_t,  y_reg_test_t)

train_loader = DataLoader(train_ds, batch_size=batch_size, shuffle=True)
test_loader  = DataLoader(test_ds,  batch_size=batch_size)

# Gasolina superior
train_ds_sup = TensorDataset(X_sup_train_t, y_sup_train_t)
test_ds_sup  = TensorDataset(X_sup_test_t,  y_sup_test_t)

train_loader_sup = DataLoader(train_ds_sup, batch_size=64, shuffle=True)
test_loader_sup  = DataLoader(test_ds_sup,  batch_size=64)

# Gasolina diesel
train_ds_dis = TensorDataset(x_dis_train_t, y_dis_train_t)
test_ds_dis = TensorDataset(x_dis_test_t, y_dis_test_t)

train_loader_dis = DataLoader(train_ds_dis, batch_size=64, shuffle=True)
test_loader_dis = DataLoader(test_ds_dis, batch_size=64)
```

In [40]:
```
# Define dos arquitecturas de LSTM
class LSTMModelA(nn.Module):
    def __init__(self, input_size=1, hidden_size=50, dropout_rate=0.2):
        super().__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out, _ = self.lstm(x)
        last = out[:, -1, :]
        y = self.dropout(last)
        return self.fc(y)
```

In [41]:
```
class LSTMModelB(nn.Module):
    def __init__(self, input_size=1, hidden_sizes=(100,50)):
        super().__init__()
        self.lstm1 = nn.LSTM(input_size,  hidden_sizes[0], batch_first=True)
        self.dropout1 = nn.Dropout(0.3)
        self.lstm2 = nn.LSTM(hidden_sizes[0], hidden_sizes[1], batch_first=True)
        self.fc     = nn.Linear(hidden_sizes[1], 1)

    def forward(self, x):
        out, _ = self.lstm1(x)
        out     = self.dropout1(out)
        out, _ = self.lstm2(out)
        last    = out[:, -1, :]
        return self.fc(last)
```

In [42]:
```
# Bucle de entrenamiento y validación
def train_model(model, train_loader, test_loader, epochs=50, lr=1e-3):
    model.to(device)
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)
    criterion = nn.MSELoss()
    best_loss = float('inf')
    patience, counter = 5, 0

    for epoch in range(1, epochs+1):
        # Training
        model.train()
        running = 0.0
        for xb, yb in train_loader:
            optimizer.zero_grad()
            preds = model(xb).squeeze()
            loss = criterion(preds, yb)
            loss.backward()
            optimizer.step()
            running += loss.item() * xb.size(0)
        train_loss = running / len(train_loader.dataset)

        # Validation
        model.eval()
        running = 0.0
        with torch.no_grad():
            for xb, yb in test_loader:
                preds = model(xb).squeeze()
                running += criterion(preds, yb).item() * xb.size(0)
        val_loss = running / len(test_loader.dataset)

        print(f"Epoch {epoch} — Train MSE: {train_loss:.4f} — Val MSE: {val_loss:.4

        if val_loss < best_loss:
            best_loss = val_loss
            counter = 0
            torch.save(model.state_dict(), "best_modelA.pth")
        else:
            counter += 1
            if counter >= patience:
                print("Early stopping.")
                break

    model.load_state_dict(torch.load("best_modelA.pth"))
    return model, best_loss
```

In [43]:
```
param_grid = {
    'hidden_size': [50, 100],
    'dropout_rate': [0.2, 0.3],
    'lr': [1e-3, 1e-4],
    'batch_size': [32, 64],
}
results = []
```

In [44]:
```
for hs, dr, lr, bs in itertools.product(
        param_grid['hidden_size'],
```

```
        param_grid['dropout_rate'],
        param_grid['lr'],
        param_grid['batch_size']
    ):
    print(f"Probando hs={hs}, dr={dr}, lr={lr}, bs={bs}…")

    # Redefine modelo con estos hiperparámetros
    model = LSTMModelA(input_size=1, hidden_size=hs, dropout_rate=dr)

    # DataLoaders con nuevo batch_size
    train_loader = DataLoader(train_ds, batch_size=bs, shuffle=True)
    test_loader  = DataLoader(test_ds,  batch_size=bs)

    # Entrena y captura el val_loss
    _, val_loss = train_model(model, train_loader, test_loader, epochs=30, lr=lr)
    results.append({
        'hidden_size': hs,
        'dropout_rate': dr,
        'lr': lr,
        'batch_size': bs,
        'val_loss': val_loss
    })
```

```
Probando hs=50, dr=0.2, lr=0.001, bs=32…
Epoch 1 — Train MSE: 0.1633 — Val MSE: 1.1606
Epoch 2 — Train MSE: 0.1167 — Val MSE: 0.9447
Epoch 3 — Train MSE: 0.0741 — Val MSE: 0.6761
Epoch 4 — Train MSE: 0.0527 — Val MSE: 0.4197
Epoch 5 — Train MSE: 0.0575 — Val MSE: 0.4568
```

```
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([32, 1])) that is different to the input size (torch.Size([32])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([1, 1])) that is different to the input size (torch.Size([])). This will likely lea
d to incorrect results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([12, 1])) that is different to the input size (torch.Size([12])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
Epoch 6 — Train MSE: 0.0511 — Val MSE: 0.6214
Epoch 7 — Train MSE: 0.0533 — Val MSE: 0.6681
Epoch 8 — Train MSE: 0.0542 — Val MSE: 0.6310
Epoch 9 — Train MSE: 0.0510 — Val MSE: 0.5377
Early stopping.
Probando hs=50, dr=0.2, lr=0.001, bs=64…
Epoch 1 — Train MSE: 0.1708 — Val MSE: 1.1474
Epoch 2 — Train MSE: 0.1378 — Val MSE: 1.0041
Epoch 3 — Train MSE: 0.1053 — Val MSE: 0.8574
Epoch 4 — Train MSE: 0.0777 — Val MSE: 0.7009
Epoch 5 — Train MSE: 0.0601 — Val MSE: 0.5539
```

```
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([64, 1])) that is different to the input size (torch.Size([64])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
Epoch 6 — Train MSE: 0.0525 — Val MSE: 0.4404
Epoch 7 — Train MSE: 0.0534 — Val MSE: 0.3808
Epoch 8 — Train MSE: 0.0555 — Val MSE: 0.3969
Epoch 9 — Train MSE: 0.0547 — Val MSE: 0.4390
Epoch 10 — Train MSE: 0.0516 — Val MSE: 0.4962
Epoch 11 — Train MSE: 0.0505 — Val MSE: 0.5005
Epoch 12 — Train MSE: 0.0503 — Val MSE: 0.5217
Early stopping.
Probando hs=50, dr=0.2, lr=0.0001, bs=32…
Epoch 1 — Train MSE: 0.1087 — Val MSE: 1.1120
Epoch 2 — Train MSE: 0.1059 — Val MSE: 1.0865
Epoch 3 — Train MSE: 0.1003 — Val MSE: 1.0617
Epoch 4 — Train MSE: 0.0958 — Val MSE: 1.0383
Epoch 5 — Train MSE: 0.0912 — Val MSE: 1.0171
Epoch 6 — Train MSE: 0.0878 — Val MSE: 0.9966
Epoch 7 — Train MSE: 0.0849 — Val MSE: 0.9755
Epoch 8 — Train MSE: 0.0814 — Val MSE: 0.9558
Epoch 9 — Train MSE: 0.0782 — Val MSE: 0.9348
Epoch 10 — Train MSE: 0.0756 — Val MSE: 0.9149
Epoch 11 — Train MSE: 0.0728 — Val MSE: 0.8964
Epoch 12 — Train MSE: 0.0710 — Val MSE: 0.8774
Epoch 13 — Train MSE: 0.0676 — Val MSE: 0.8583
Epoch 14 — Train MSE: 0.0650 — Val MSE: 0.8380
Epoch 15 — Train MSE: 0.0624 — Val MSE: 0.8150
Epoch 16 — Train MSE: 0.0604 — Val MSE: 0.7909
Epoch 17 — Train MSE: 0.0581 — Val MSE: 0.7714
Epoch 18 — Train MSE: 0.0564 — Val MSE: 0.7515
Epoch 19 — Train MSE: 0.0549 — Val MSE: 0.7321
Epoch 20 — Train MSE: 0.0537 — Val MSE: 0.7152
Epoch 21 — Train MSE: 0.0528 — Val MSE: 0.6962
Epoch 22 — Train MSE: 0.0515 — Val MSE: 0.6838
Epoch 23 — Train MSE: 0.0512 — Val MSE: 0.6740
Epoch 24 — Train MSE: 0.0513 — Val MSE: 0.6607
Epoch 25 — Train MSE: 0.0502 — Val MSE: 0.6507
Epoch 26 — Train MSE: 0.0500 — Val MSE: 0.6427
Epoch 27 — Train MSE: 0.0495 — Val MSE: 0.6400
Epoch 28 — Train MSE: 0.0496 — Val MSE: 0.6381
Epoch 29 — Train MSE: 0.0500 — Val MSE: 0.6411
Epoch 30 — Train MSE: 0.0499 — Val MSE: 0.6349
Probando hs=50, dr=0.2, lr=0.0001, bs=64…
Epoch 1 — Train MSE: 0.1681 — Val MSE: 1.2633
Epoch 2 — Train MSE: 0.1647 — Val MSE: 1.2506
Epoch 3 — Train MSE: 0.1601 — Val MSE: 1.2378
Epoch 4 — Train MSE: 0.1584 — Val MSE: 1.2248
Epoch 5 — Train MSE: 0.1558 — Val MSE: 1.2119
Epoch 6 — Train MSE: 0.1509 — Val MSE: 1.1992
Epoch 7 — Train MSE: 0.1468 — Val MSE: 1.1864
Epoch 8 — Train MSE: 0.1462 — Val MSE: 1.1740
Epoch 9 — Train MSE: 0.1429 — Val MSE: 1.1608
Epoch 10 — Train MSE: 0.1390 — Val MSE: 1.1478
Epoch 11 — Train MSE: 0.1360 — Val MSE: 1.1354
Epoch 12 — Train MSE: 0.1337 — Val MSE: 1.1230
Epoch 13 — Train MSE: 0.1309 — Val MSE: 1.1114
Epoch 14 — Train MSE: 0.1275 — Val MSE: 1.0993
Epoch 15 — Train MSE: 0.1256 — Val MSE: 1.0863
Epoch 16 — Train MSE: 0.1207 — Val MSE: 1.0735
```

```
Epoch 17 — Train MSE: 0.1185 — Val MSE: 1.0612
Epoch 18 — Train MSE: 0.1157 — Val MSE: 1.0490
Epoch 19 — Train MSE: 0.1140 — Val MSE: 1.0360
Epoch 20 — Train MSE: 0.1098 — Val MSE: 1.0205
Epoch 21 — Train MSE: 0.1065 — Val MSE: 1.0051
Epoch 22 — Train MSE: 0.1039 — Val MSE: 0.9901
Epoch 23 — Train MSE: 0.1000 — Val MSE: 0.9732
Epoch 24 — Train MSE: 0.0969 — Val MSE: 0.9566
Epoch 25 — Train MSE: 0.0948 — Val MSE: 0.9405
Epoch 26 — Train MSE: 0.0894 — Val MSE: 0.9238
Epoch 27 — Train MSE: 0.0883 — Val MSE: 0.9073
Epoch 28 — Train MSE: 0.0859 — Val MSE: 0.8904
Epoch 29 — Train MSE: 0.0827 — Val MSE: 0.8744
Epoch 30 — Train MSE: 0.0789 — Val MSE: 0.8582
Probando hs=50, dr=0.3, lr=0.001, bs=32…
Epoch 1 — Train MSE: 0.2533 — Val MSE: 1.3796
Epoch 2 — Train MSE: 0.1810 — Val MSE: 1.0898
Epoch 3 — Train MSE: 0.1036 — Val MSE: 0.6767
Epoch 4 — Train MSE: 0.0578 — Val MSE: 0.3246
Epoch 5 — Train MSE: 0.0597 — Val MSE: 0.4936
Epoch 6 — Train MSE: 0.0561 — Val MSE: 0.6641
Epoch 7 — Train MSE: 0.0612 — Val MSE: 0.6199
Epoch 8 — Train MSE: 0.0553 — Val MSE: 0.4487
Epoch 9 — Train MSE: 0.0551 — Val MSE: 0.3984
Early stopping.
Probando hs=50, dr=0.3, lr=0.001, bs=64…
Epoch 1 — Train MSE: 0.0804 — Val MSE: 0.8502
Epoch 2 — Train MSE: 0.0648 — Val MSE: 0.7611
Epoch 3 — Train MSE: 0.0572 — Val MSE: 0.7014
Epoch 4 — Train MSE: 0.0543 — Val MSE: 0.6591
Epoch 5 — Train MSE: 0.0519 — Val MSE: 0.6158
Epoch 6 — Train MSE: 0.0507 — Val MSE: 0.5702
Epoch 7 — Train MSE: 0.0504 — Val MSE: 0.5551
Epoch 8 — Train MSE: 0.0501 — Val MSE: 0.5592
Epoch 9 — Train MSE: 0.0501 — Val MSE: 0.5689
Epoch 10 — Train MSE: 0.0506 — Val MSE: 0.5876
Epoch 11 — Train MSE: 0.0503 — Val MSE: 0.5755
Epoch 12 — Train MSE: 0.0502 — Val MSE: 0.5704
Early stopping.
Probando hs=50, dr=0.3, lr=0.0001, bs=32…
Epoch 1 — Train MSE: 0.1686 — Val MSE: 1.2857
Epoch 2 — Train MSE: 0.1661 — Val MSE: 1.2647
Epoch 3 — Train MSE: 0.1594 — Val MSE: 1.2437
Epoch 4 — Train MSE: 0.1560 — Val MSE: 1.2232
Epoch 5 — Train MSE: 0.1488 — Val MSE: 1.2030
Epoch 6 — Train MSE: 0.1440 — Val MSE: 1.1831
Epoch 7 — Train MSE: 0.1378 — Val MSE: 1.1635
Epoch 8 — Train MSE: 0.1335 — Val MSE: 1.1433
Epoch 9 — Train MSE: 0.1313 — Val MSE: 1.1219
Epoch 10 — Train MSE: 0.1255 — Val MSE: 1.1000
Epoch 11 — Train MSE: 0.1219 — Val MSE: 1.0780
Epoch 12 — Train MSE: 0.1137 — Val MSE: 1.0548
Epoch 13 — Train MSE: 0.1121 — Val MSE: 1.0310
Epoch 14 — Train MSE: 0.1063 — Val MSE: 1.0063
Epoch 15 — Train MSE: 0.1028 — Val MSE: 0.9812
Epoch 16 — Train MSE: 0.0982 — Val MSE: 0.9558
```

```
Epoch 17 — Train MSE: 0.0902 — Val MSE: 0.9289
Epoch 18 — Train MSE: 0.0866 — Val MSE: 0.9003
Epoch 19 — Train MSE: 0.0836 — Val MSE: 0.8715
Epoch 20 — Train MSE: 0.0791 — Val MSE: 0.8374
Epoch 21 — Train MSE: 0.0725 — Val MSE: 0.8034
Epoch 22 — Train MSE: 0.0693 — Val MSE: 0.7744
Epoch 23 — Train MSE: 0.0670 — Val MSE: 0.7441
Epoch 24 — Train MSE: 0.0624 — Val MSE: 0.7193
Epoch 25 — Train MSE: 0.0610 — Val MSE: 0.6901
Epoch 26 — Train MSE: 0.0586 — Val MSE: 0.6567
Epoch 27 — Train MSE: 0.0553 — Val MSE: 0.6295
Epoch 28 — Train MSE: 0.0547 — Val MSE: 0.6067
Epoch 29 — Train MSE: 0.0527 — Val MSE: 0.5893
Epoch 30 — Train MSE: 0.0517 — Val MSE: 0.5740
Probando hs=50, dr=0.3, lr=0.0001, bs=64…
Epoch 1 — Train MSE: 0.1610 — Val MSE: 1.2992
Epoch 2 — Train MSE: 0.1567 — Val MSE: 1.2853
Epoch 3 — Train MSE: 0.1539 — Val MSE: 1.2711
Epoch 4 — Train MSE: 0.1513 — Val MSE: 1.2573
Epoch 5 — Train MSE: 0.1459 — Val MSE: 1.2439
Epoch 6 — Train MSE: 0.1423 — Val MSE: 1.2306
Epoch 7 — Train MSE: 0.1402 — Val MSE: 1.2171
Epoch 8 — Train MSE: 0.1351 — Val MSE: 1.2034
Epoch 9 — Train MSE: 0.1337 — Val MSE: 1.1895
Epoch 10 — Train MSE: 0.1310 — Val MSE: 1.1759
Epoch 11 — Train MSE: 0.1271 — Val MSE: 1.1618
Epoch 12 — Train MSE: 0.1252 — Val MSE: 1.1479
Epoch 13 — Train MSE: 0.1216 — Val MSE: 1.1344
Epoch 14 — Train MSE: 0.1203 — Val MSE: 1.1213
Epoch 15 — Train MSE: 0.1127 — Val MSE: 1.1083
Epoch 16 — Train MSE: 0.1137 — Val MSE: 1.0949
Epoch 17 — Train MSE: 0.1109 — Val MSE: 1.0813
Epoch 18 — Train MSE: 0.1083 — Val MSE: 1.0678
Epoch 19 — Train MSE: 0.1068 — Val MSE: 1.0538
Epoch 20 — Train MSE: 0.1021 — Val MSE: 1.0387
Epoch 21 — Train MSE: 0.0985 — Val MSE: 1.0241
Epoch 22 — Train MSE: 0.0974 — Val MSE: 1.0094
Epoch 23 — Train MSE: 0.0936 — Val MSE: 0.9936
Epoch 24 — Train MSE: 0.0924 — Val MSE: 0.9784
Epoch 25 — Train MSE: 0.0901 — Val MSE: 0.9628
Epoch 26 — Train MSE: 0.0865 — Val MSE: 0.9491
Epoch 27 — Train MSE: 0.0843 — Val MSE: 0.9358
Epoch 28 — Train MSE: 0.0828 — Val MSE: 0.9211
Epoch 29 — Train MSE: 0.0783 — Val MSE: 0.9072
Epoch 30 — Train MSE: 0.0765 — Val MSE: 0.8913
Probando hs=100, dr=0.2, lr=0.001, bs=32…
Epoch 1 — Train MSE: 0.0949 — Val MSE: 0.7878
Epoch 2 — Train MSE: 0.0543 — Val MSE: 0.5030
Epoch 3 — Train MSE: 0.0559 — Val MSE: 0.4986
Epoch 4 — Train MSE: 0.0508 — Val MSE: 0.6345
Epoch 5 — Train MSE: 0.0510 — Val MSE: 0.6724
Epoch 6 — Train MSE: 0.0511 — Val MSE: 0.6293
Epoch 7 — Train MSE: 0.0493 — Val MSE: 0.5862
Epoch 8 — Train MSE: 0.0491 — Val MSE: 0.5845
Early stopping.
Probando hs=100, dr=0.2, lr=0.001, bs=64…
```

```
Epoch 1 — Train MSE: 0.2295 — Val MSE: 1.3375
Epoch 2 — Train MSE: 0.1653 — Val MSE: 1.0797
Epoch 3 — Train MSE: 0.1046 — Val MSE: 0.7390
Epoch 4 — Train MSE: 0.0532 — Val MSE: 0.3239
Epoch 5 — Train MSE: 0.0783 — Val MSE: 0.2129
Epoch 6 — Train MSE: 0.0849 — Val MSE: 0.3685
Epoch 7 — Train MSE: 0.0526 — Val MSE: 0.5821
Epoch 8 — Train MSE: 0.0540 — Val MSE: 0.6881
Epoch 9 — Train MSE: 0.0620 — Val MSE: 0.7097
Epoch 10 — Train MSE: 0.0611 — Val MSE: 0.6515
Early stopping.
Probando hs=100, dr=0.2, lr=0.0001, bs=32…
Epoch 1 — Train MSE: 0.2704 — Val MSE: 1.5991
Epoch 2 — Train MSE: 0.2546 — Val MSE: 1.5533
Epoch 3 — Train MSE: 0.2411 — Val MSE: 1.5089
Epoch 4 — Train MSE: 0.2296 — Val MSE: 1.4653
Epoch 5 — Train MSE: 0.2155 — Val MSE: 1.4220
Epoch 6 — Train MSE: 0.2043 — Val MSE: 1.3775
Epoch 7 — Train MSE: 0.1920 — Val MSE: 1.3303
Epoch 8 — Train MSE: 0.1806 — Val MSE: 1.2809
Epoch 9 — Train MSE: 0.1673 — Val MSE: 1.2258
Epoch 10 — Train MSE: 0.1535 — Val MSE: 1.1687
Epoch 11 — Train MSE: 0.1400 — Val MSE: 1.1034
Epoch 12 — Train MSE: 0.1244 — Val MSE: 1.0328
Epoch 13 — Train MSE: 0.1106 — Val MSE: 0.9526
Epoch 14 — Train MSE: 0.0957 — Val MSE: 0.8716
Epoch 15 — Train MSE: 0.0830 — Val MSE: 0.7875
Epoch 16 — Train MSE: 0.0714 — Val MSE: 0.6996
Epoch 17 — Train MSE: 0.0619 — Val MSE: 0.6142
Epoch 18 — Train MSE: 0.0546 — Val MSE: 0.5446
Epoch 19 — Train MSE: 0.0512 — Val MSE: 0.4856
Epoch 20 — Train MSE: 0.0500 — Val MSE: 0.4398
Epoch 21 — Train MSE: 0.0500 — Val MSE: 0.4199
Epoch 22 — Train MSE: 0.0510 — Val MSE: 0.3910
Epoch 23 — Train MSE: 0.0518 — Val MSE: 0.3846
Epoch 24 — Train MSE: 0.0530 — Val MSE: 0.3843
Epoch 25 — Train MSE: 0.0517 — Val MSE: 0.4326
Epoch 26 — Train MSE: 0.0501 — Val MSE: 0.4773
Epoch 27 — Train MSE: 0.0505 — Val MSE: 0.4904
Epoch 28 — Train MSE: 0.0505 — Val MSE: 0.4912
Epoch 29 — Train MSE: 0.0501 — Val MSE: 0.4728
Early stopping.
Probando hs=100, dr=0.2, lr=0.0001, bs=64…
Epoch 1 — Train MSE: 0.2023 — Val MSE: 1.3878
Epoch 2 — Train MSE: 0.1963 — Val MSE: 1.3665
Epoch 3 — Train MSE: 0.1905 — Val MSE: 1.3453
Epoch 4 — Train MSE: 0.1855 — Val MSE: 1.3240
Epoch 5 — Train MSE: 0.1787 — Val MSE: 1.3028
Epoch 6 — Train MSE: 0.1736 — Val MSE: 1.2816
Epoch 7 — Train MSE: 0.1683 — Val MSE: 1.2602
Epoch 8 — Train MSE: 0.1629 — Val MSE: 1.2381
Epoch 9 — Train MSE: 0.1574 — Val MSE: 1.2158
Epoch 10 — Train MSE: 0.1511 — Val MSE: 1.1940
Epoch 11 — Train MSE: 0.1465 — Val MSE: 1.1710
Epoch 12 — Train MSE: 0.1413 — Val MSE: 1.1470
Epoch 13 — Train MSE: 0.1360 — Val MSE: 1.1218
```

```
Epoch 14 — Train MSE: 0.1294 — Val MSE: 1.0953
Epoch 15 — Train MSE: 0.1243 — Val MSE: 1.0664
Epoch 16 — Train MSE: 0.1172 — Val MSE: 1.0353
Epoch 17 — Train MSE: 0.1111 — Val MSE: 1.0006
Epoch 18 — Train MSE: 0.1050 — Val MSE: 0.9629
Epoch 19 — Train MSE: 0.0973 — Val MSE: 0.9242
Epoch 20 — Train MSE: 0.0896 — Val MSE: 0.8836
Epoch 21 — Train MSE: 0.0831 — Val MSE: 0.8379
Epoch 22 — Train MSE: 0.0755 — Val MSE: 0.7922
Epoch 23 — Train MSE: 0.0694 — Val MSE: 0.7431
Epoch 24 — Train MSE: 0.0637 — Val MSE: 0.6930
Epoch 25 — Train MSE: 0.0595 — Val MSE: 0.6514
Epoch 26 — Train MSE: 0.0547 — Val MSE: 0.6098
Epoch 27 — Train MSE: 0.0525 — Val MSE: 0.5641
Epoch 28 — Train MSE: 0.0508 — Val MSE: 0.5291
Epoch 29 — Train MSE: 0.0498 — Val MSE: 0.5154
Epoch 30 — Train MSE: 0.0498 — Val MSE: 0.5108
Probando hs=100, dr=0.3, lr=0.001, bs=32…
Epoch 1 — Train MSE: 0.0928 — Val MSE: 0.7195
Epoch 2 — Train MSE: 0.0541 — Val MSE: 0.4132
Epoch 3 — Train MSE: 0.0575 — Val MSE: 0.4269
Epoch 4 — Train MSE: 0.0500 — Val MSE: 0.5875
Epoch 5 — Train MSE: 0.0516 — Val MSE: 0.6011
Epoch 6 — Train MSE: 0.0509 — Val MSE: 0.5297
Epoch 7 — Train MSE: 0.0496 — Val MSE: 0.5219
Early stopping.
Probando hs=100, dr=0.3, lr=0.001, bs=64…
Epoch 1 — Train MSE: 0.1446 — Val MSE: 1.0906
Epoch 2 — Train MSE: 0.1053 — Val MSE: 0.9013
Epoch 3 — Train MSE: 0.0728 — Val MSE: 0.6717
Epoch 4 — Train MSE: 0.0517 — Val MSE: 0.4203
Epoch 5 — Train MSE: 0.0546 — Val MSE: 0.3821
Epoch 6 — Train MSE: 0.0614 — Val MSE: 0.3568
Epoch 7 — Train MSE: 0.0582 — Val MSE: 0.4663
Epoch 8 — Train MSE: 0.0497 — Val MSE: 0.5760
Epoch 9 — Train MSE: 0.0517 — Val MSE: 0.6419
Epoch 10 — Train MSE: 0.0540 — Val MSE: 0.6636
Epoch 11 — Train MSE: 0.0546 — Val MSE: 0.6448
Early stopping.
Probando hs=100, dr=0.3, lr=0.0001, bs=32…
Epoch 1 — Train MSE: 0.1031 — Val MSE: 0.9906
Epoch 2 — Train MSE: 0.0977 — Val MSE: 0.9636
Epoch 3 — Train MSE: 0.0916 — Val MSE: 0.9379
Epoch 4 — Train MSE: 0.0871 — Val MSE: 0.9115
Epoch 5 — Train MSE: 0.0804 — Val MSE: 0.8845
Epoch 6 — Train MSE: 0.0762 — Val MSE: 0.8567
Epoch 7 — Train MSE: 0.0722 — Val MSE: 0.8287
Epoch 8 — Train MSE: 0.0684 — Val MSE: 0.8009
Epoch 9 — Train MSE: 0.0644 — Val MSE: 0.7741
Epoch 10 — Train MSE: 0.0611 — Val MSE: 0.7445
Epoch 11 — Train MSE: 0.0576 — Val MSE: 0.7144
Epoch 12 — Train MSE: 0.0559 — Val MSE: 0.6899
Epoch 13 — Train MSE: 0.0535 — Val MSE: 0.6684
Epoch 14 — Train MSE: 0.0521 — Val MSE: 0.6466
Epoch 15 — Train MSE: 0.0503 — Val MSE: 0.6230
Epoch 16 — Train MSE: 0.0500 — Val MSE: 0.6029
```

```
Epoch 17 — Train MSE: 0.0495 — Val MSE: 0.5875
Epoch 18 — Train MSE: 0.0494 — Val MSE: 0.5852
Epoch 19 — Train MSE: 0.0495 — Val MSE: 0.5887
Epoch 20 — Train MSE: 0.0492 — Val MSE: 0.5863
Epoch 21 — Train MSE: 0.0495 — Val MSE: 0.5840
Epoch 22 — Train MSE: 0.0494 — Val MSE: 0.5895
Epoch 23 — Train MSE: 0.0494 — Val MSE: 0.5916
Epoch 24 — Train MSE: 0.0493 — Val MSE: 0.5732
Epoch 25 — Train MSE: 0.0492 — Val MSE: 0.5503
Epoch 26 — Train MSE: 0.0494 — Val MSE: 0.5479
Epoch 27 — Train MSE: 0.0492 — Val MSE: 0.5588
Epoch 28 — Train MSE: 0.0490 — Val MSE: 0.5726
Epoch 29 — Train MSE: 0.0493 — Val MSE: 0.5787
Epoch 30 — Train MSE: 0.0495 — Val MSE: 0.5858
Probando hs=100, dr=0.3, lr=0.0001, bs=64…
Epoch 1 — Train MSE: 0.1911 — Val MSE: 1.4051
Epoch 2 — Train MSE: 0.1850 — Val MSE: 1.3827
Epoch 3 — Train MSE: 0.1772 — Val MSE: 1.3611
Epoch 4 — Train MSE: 0.1732 — Val MSE: 1.3401
Epoch 5 — Train MSE: 0.1659 — Val MSE: 1.3192
Epoch 6 — Train MSE: 0.1627 — Val MSE: 1.2990
Epoch 7 — Train MSE: 0.1584 — Val MSE: 1.2799
Epoch 8 — Train MSE: 0.1536 — Val MSE: 1.2611
Epoch 9 — Train MSE: 0.1480 — Val MSE: 1.2422
Epoch 10 — Train MSE: 0.1428 — Val MSE: 1.2236
Epoch 11 — Train MSE: 0.1386 — Val MSE: 1.2049
Epoch 12 — Train MSE: 0.1338 — Val MSE: 1.1865
Epoch 13 — Train MSE: 0.1280 — Val MSE: 1.1674
Epoch 14 — Train MSE: 0.1252 — Val MSE: 1.1473
Epoch 15 — Train MSE: 0.1200 — Val MSE: 1.1264
Epoch 16 — Train MSE: 0.1157 — Val MSE: 1.1044
Epoch 17 — Train MSE: 0.1108 — Val MSE: 1.0820
Epoch 18 — Train MSE: 0.1064 — Val MSE: 1.0596
Epoch 19 — Train MSE: 0.1018 — Val MSE: 1.0367
Epoch 20 — Train MSE: 0.0969 — Val MSE: 1.0131
Epoch 21 — Train MSE: 0.0916 — Val MSE: 0.9876
Epoch 22 — Train MSE: 0.0884 — Val MSE: 0.9593
Epoch 23 — Train MSE: 0.0832 — Val MSE: 0.9306
Epoch 24 — Train MSE: 0.0782 — Val MSE: 0.9022
Epoch 25 — Train MSE: 0.0751 — Val MSE: 0.8736
Epoch 26 — Train MSE: 0.0700 — Val MSE: 0.8444
Epoch 27 — Train MSE: 0.0663 — Val MSE: 0.8111
Epoch 28 — Train MSE: 0.0631 — Val MSE: 0.7793
Epoch 29 — Train MSE: 0.0592 — Val MSE: 0.7420
Epoch 30 — Train MSE: 0.0563 — Val MSE: 0.7124
```

In [45]:
```
df_results = pd.DataFrame(results)
print(df_results.sort_values('val_loss').head())
```

```
    hidden_size  dropout_rate      lr  batch_size  val_loss
9           100           0.2  0.0010          64  0.212905
4            50           0.3  0.0010          32  0.324617
13          100           0.3  0.0010          64  0.356820
1            50           0.2  0.0010          64  0.380771
10          100           0.2  0.0001          32  0.384292
```

In [46]:
```
# Mejores hiperparametros
best_params_reg = {
    'hidden_size': 100,
    'dropout_rate': 0.3,
    'lr': 1e-3,
    'batch_size': 64
}
```

## Mejor modelo para gasolina regular

In [47]:
```
# Definir modelo con estos hieperparametros
model_reg = LSTMModelA(
    input_size=1,
    hidden_size=best_params_reg['hidden_size'],
    dropout_rate=best_params_reg['dropout_rate']
)

train_loader_reg = DataLoader(train_ds, batch_size=best_params_reg['batch_size'], s
test_loader_reg  = DataLoader(test_ds,  batch_size=best_params_reg['batch_size'])


# Entrena y captura la mejor pérdida de validación
model_reg, best_loss_reg = train_model(
    model_reg,
    train_loader_reg,
    test_loader_reg,
    epochs=50,
    lr=best_params_reg['lr']
)
print(f"Mejor Val MSE (regular): {best_loss_reg:.4f}")
```

```
Epoch 1 — Train MSE: 0.1642 — Val MSE: 1.1269
Epoch 2 — Train MSE: 0.1171 — Val MSE: 0.9335
Epoch 3 — Train MSE: 0.0806 — Val MSE: 0.7179
Epoch 4 — Train MSE: 0.0559 — Val MSE: 0.4862
Epoch 5 — Train MSE: 0.0531 — Val MSE: 0.3736
Epoch 6 — Train MSE: 0.0584 — Val MSE: 0.4330
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([64, 1])) that is different to the input size (torch.Size([64])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([1, 1])) that is different to the input size (torch.Size([])). This will likely lea
d to incorrect results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([12, 1])) that is different to the input size (torch.Size([12])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
```

```
Epoch 7 — Train MSE: 0.0514 — Val MSE: 0.5512
Epoch 8 — Train MSE: 0.0504 — Val MSE: 0.6294
Epoch 9 — Train MSE: 0.0538 — Val MSE: 0.6631
Epoch 10 — Train MSE: 0.0555 — Val MSE: 0.6551
Early stopping.
Mejor Val MSE (regular): 0.3736
```

In [48]:
```python
# Predicciones
model_reg.eval()
preds_reg = []
with torch.no_grad():
    for xb, _ in test_loader_reg:
        preds_reg.extend(model_reg(xb).squeeze().cpu().numpy())
preds_reg = np.array(preds_reg).reshape(-1,1)

# Inversión de escala
preds_reg_inv = regular_scaler.inverse_transform(preds_reg)
y_reg_true    = regular_test.values[window_size:]

# Metricas
mae_reg  = mean_absolute_error(y_reg_true, preds_reg_inv)
rmse_reg = np.sqrt(mean_squared_error(y_reg_true, preds_reg_inv))
print(f"Gasolina Regular: \n\nMAE: {mae_reg:.4f} \nRMSE: {rmse_reg:.4f}")
```

```
Gasolina Regular:

MAE: 342398.5285
RMSE: 364936.5787
```

## Mejor modelo para gasolina Superior

In [49]:
```python
model_sup = LSTMModelA(input_size=1, hidden_size=100, dropout_rate=0.3)

model_sup, best_loss_sup = train_model(
    model_sup,
    train_loader_sup,
    test_loader_sup,
    epochs=50,
    lr=1e-3
)
print(f"Mejor Val MSE (superior): {best_loss_sup:.4f}")
```

```
Epoch 1 — Train MSE: 0.1540 — Val MSE: 0.3782
Epoch 2 — Train MSE: 0.1023 — Val MSE: 0.2804
Epoch 3 — Train MSE: 0.0644 — Val MSE: 0.1830
Epoch 4 — Train MSE: 0.0456 — Val MSE: 0.1199
Epoch 5 — Train MSE: 0.0446 — Val MSE: 0.1366
Epoch 6 — Train MSE: 0.0439 — Val MSE: 0.1611
Epoch 7 — Train MSE: 0.0456 — Val MSE: 0.1759
```

```
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([64, 1])) that is different to the input size (torch.Size([64])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([1, 1])) that is different to the input size (torch.Size([])). This will likely lea
d to incorrect results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([12, 1])) that is different to the input size (torch.Size([12])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
Epoch 8 — Train MSE: 0.0458 — Val MSE: 0.1675
Epoch 9 — Train MSE: 0.0452 — Val MSE: 0.1612
Early stopping.
Mejor Val MSE (superior): 0.1199
```

In [50]:
```python
# Predicciones
model_sup.eval()
preds_sup = []
with torch.no_grad():
    for xb, _ in test_loader_sup:
        preds_sup.extend(model_sup(xb).squeeze().cpu().numpy())
preds_sup = np.array(preds_sup).reshape(-1,1)

# Inversión de escala
preds_sup_inv = super_scaler.inverse_transform(preds_sup)
y_sup_true    = super_test.values[window_size:]

# Métricas
mae_sup  = mean_absolute_error(y_sup_true, preds_sup_inv)
rmse_sup = np.sqrt(mean_squared_error(y_sup_true, preds_sup_inv))
print(f"Gasolina Superior:  \n\nMAE: {mae_sup:.4f} \nRMSE: {rmse_sup:.4f}")
```

```
Gasolina Superior:

MAE: 179196.5457
RMSE: 209866.8716
```

## Mejor modelo para Diesel

In [52]:
```python
model_dis = LSTMModelA(input_size=1, hidden_size=100, dropout_rate=0.3)

model_dis, best_loss_dis = train_model(
    model_dis,
    train_loader_dis,
    test_loader_dis,
    epochs=50,
    lr=1e-3
```

```
)
print(f"Mejor Val MSE (diesel): {best_loss_dis:.4f}")
```

```
Epoch 1 — Train MSE: 0.2631 — Val MSE: 0.0033
Epoch 2 — Train MSE: 0.1783 — Val MSE: 0.0208
Epoch 3 — Train MSE: 0.1031 — Val MSE: 0.0745
Epoch 4 — Train MSE: 0.0342 — Val MSE: 0.2312
Epoch 5 — Train MSE: 0.0443 — Val MSE: 0.2057
Epoch 6 — Train MSE: 0.0263 — Val MSE: 0.1285
Early stopping.
Mejor Val MSE (diesel): 0.0033
```

```
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([64, 1])) that is different to the input size (torch.Size([64])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([1, 1])) that is different to the input size (torch.Size([])). This will likely lea
d to incorrect results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
c:\Users\MSI\Desktop\info\Escritorio\UVG\8vo Semestre\Data Science\DSvenv\Lib\site-p
ackages\torch\nn\modules\loss.py:610: UserWarning: Using a target size (torch.Size
([12, 1])) that is different to the input size (torch.Size([12])). This will likely
lead to incorrect results due to broadcasting. Please ensure they have the same siz
e.
  return F.mse_loss(input, target, reduction=self.reduction)
```

In [53]:
```
# Predicciones
model_dis.eval()
preds_dis = []
with torch.no_grad():
    for xb, _ in test_loader_dis:
        preds_dis.extend(model_dis(xb).squeeze().cpu().numpy())
preds_dis = np.array(preds_dis).reshape(-1, 1)

# Inversion de escala
preds_dis_inv = diesel_scaler.inverse_transform(preds_dis)
y_dis_true = diesel_test.values[window_size:]

# Metricas
mae_dis = mean_squared_error(y_dis_true, preds_dis_inv)
rmse_dis = np.sqrt(mean_squared_error(y_dis_true, preds_dis_inv))
print(f"Gasolina Diesel: \n\nMae: {mae_dis:.4f} \nRMSE: {rmse_dis:.4f}")
```

```
Gasolina Diesel:

Mae: 8460750124.5191
RMSE: 91982.3359
```

# Comparacion modelos Laboratorio 1 vs Laboratorio 2

### Laboratorio 1 – Modelos ARIMA

- Gasolina Regular

  Modelo usado para predicción de los últimos 36 meses: ARIMA(45,1,20)

  MAE: 102 164.24

  RMSE: 124 108.98

- Gasolina Superior

  Modelo usado para predicción de los últimos 36 meses: ARIMA(45,1,20)

  MAE: 93 291.85

  RMSE: 130 298.21

- Gasolina Diesel

  Sobre la ultima serie filtrada hasta diciembre 2017: ARIMA(45,1,20)

  MAE: 182 448.53

  RMSE: 228 476.56

**Laboratorio 2 – Modelos LSTM**

- Gasolina Regular

  MAE: 361 909.9981

  RMSE: 383 606.3430

- Gasolina Superior

  MAE: 116 476.7082

  RMSE: 155 397.9649

- Gasolina Diesel:

  MAE: 8460750124.5191

  RMSE: 91982.3359

Para comparar los modelos usamos dos criterios claros sobre la serie original:

- **MAE** (error absoluto medio), que nos dice cuánto, en promedio, se equivocan las predicciones.
- **RMSE** (raíz del error cuadrático medio), que penaliza más los errores grandes.

En **Gasolina Regular**, el LSTM obtuvo un **MAE = 361 910** y un **RMSE = 383 606**, mientras que el ARIMA(45,1,20) del laboratorio 1 logró un **MAE = 102 164** y **RMSE = 124 109** . Es decir, el ARIMA se quedó mucho más cerca de los valores reales, con errores alrededor de un tercio de los del LSTM. Por eso, para esta serie el ARIMA sigue siendo el ganador claro.

En **Gasolina Superior**, el LSTM bajó el error respecto al caso anterior (MAE = 116 477, RMSE = 155 398), pero el ARIMA(45,1,20) todavía lo superó con un **MAE = 93 292** y **RMSE = 130 298** . Aunque la diferencia es menor que en la serie regular, de nuevo el ARIMA entrega predicciones más ajustadas.

Para **Gasolina Diésel** vemos un comportamiento curioso: el LSTM reportó un **RMSE = 91 982**, que técnicamente es mejor que el **228 477** del ARIMA, pero su **MAE = 8 460 750 124** es colosal, lo que indica que hubo predicciones completamente fuera de escala. En cambio, el ARIMA(45,1,20) mantuvo ambos errores en rangos razonables (MAE = 182 449, RMSE = 228 477) . Esto sugiere que el LSTM, al no capturar bien la dinámica de la serie diésel, produjo picos de error muy grandes que no reflejan la realidad.

**Conclusión**:

- **ARIMA(45,1,20)** es el mejor modelo para las tres series cuando se mide con MAE y RMSE en la escala original.
- El **LSTM**, aunque prometedor en otros contextos, necesita más ajustes (arquitectura, datos de entrada o exógenas) para alcanzar a los modelos clásicos en estas series de gasolina.