Date of the Session:__/__/___ **Time of the Session:___to_____**

SKILLING-8:

Implement Regression using car price dataset with drop out, normalization layers. Use early stopping to overcome overfit.

```
import torch, numpy as np, pandas as pd
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data = pd.read_csv("car_price_dataset.csv")
X, y = pd.get_dummies(data.drop(columns=['price']), columns=['brand']).values,
data['price'].values.reshape(-1, 1)

scaler_X, scaler_y = StandardScaler(), StandardScaler()
X, y = scaler_X.fit_transform(X), scaler_y.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train, X_test, y_train, y_test = map(torch.tensor, (X_train, X_test, y_train, y_test))
X_train, X_test, y_train, y_test = X_train.float(), X_test.float(), y_train.float(),
y_test.float()

train_loader = DataLoader(TensorDataset(X_train, y_train), batch_size=32, shuffle=True)
test_loader = DataLoader(TensorDataset(X_test, y_test), batch_size=32, shuffle=False)

class CarPriceModel(nn.Module):
   def __init__(self, input_dim):
      super().__init__()
      self.model = nn.Sequential(nn.Linear(input_dim, 128), nn.BatchNorm1d(128),
nn.ReLU(), nn.Dropout(0.3),
                        nn.Linear(128, 64), nn.BatchNorm1d(64), nn.ReLU(),
nn.Dropout(0.3),
                        nn.Linear(64, 1))
   def forward(self, x): return self.model(x)

model, criterion, optimizer = CarPriceModel(X_train.shape[1]), nn.MSELoss(),
optim.Adam(model.parameters(), lr=0.001)
```

```
best_loss, patience, patience_counter = float('inf'), 10, 0
for epoch in range(100):
    model.train()
    train_loss = np.mean([criterion(model(Xb), yb).item() for Xb, yb in train_loader])

    model.eval()
    val_loss = np.mean([criterion(model(Xb), yb).item() for Xb, yb in test_loader])

    print(f"Epoch {epoch+1}: Train Loss: {train_loss:.4f}, Val Loss: {val_loss:.4f}")

    if val_loss < best_loss:
        best_loss, patience_counter = val_loss, 0
        torch.save(model.state_dict(), 'best_model.pth')
    elif (patience_counter := patience_counter + 1) >= patience:
        print("Early stopping triggered"); break

model.load_state_dict(torch.load('best_model.pth'))
y_pred, y_true = np.concatenate([model(Xb).detach().numpy() for Xb, _ in test_loader]),
np.concatenate([yb.numpy() for _, yb in test_loader])
y_pred, y_true = scaler_y.inverse_transform(y_pred),
scaler_y.inverse_transform(y_true)
print(f"Final Test MSE: {np.mean((y_pred - y_true) ** 2):.4f}")
```

Output:

Epoch 1: Train Loss: 0.3919, Val Loss: 0.0958
Epoch 2: Train Loss: 0.2129, Val Loss: 0.0612
Epoch 3: Train Loss: 0.1986, Val Loss: 0.0314
Epoch 4: Train Loss: 0.1486, Val Loss: 0.0277
Epoch 5: Train Loss: 0.1374, Val Loss: 0.0282
Epoch 6: Train Loss: 0.1249, Val Loss: 0.0291
Epoch 7: Train Loss: 0.1298, Val Loss: 0.0226
Epoch 8: Train Loss: 0.1110, Val Loss: 0.0250
Epoch 9: Train Loss: 0.1036, Val Loss: 0.0208
Epoch 10: Train Loss: 0.1226, Val Loss: 0.0239
Epoch 11: Train Loss: 0.1121, Val Loss: 0.0176
Epoch 12: Train Loss: 0.1066, Val Loss: 0.0233
Epoch 13: Train Loss: 0.0796, Val Loss: 0.0164
Epoch 14: Train Loss: 0.0995, Val Loss: 0.0167
Epoch 15: Train Loss: 0.0910, Val Loss: 0.0156
Epoch 16: Train Loss: 0.1058, Val Loss: 0.0146
Epoch 17: Train Loss: 0.1081, Val Loss: 0.0162
Epoch 18: Train Loss: 0.0987, Val Loss: 0.0132
Epoch 19: Train Loss: 0.0878, Val Loss: 0.0155
Epoch 20: Train Loss: 0.0916, Val Loss: 0.0147
Epoch 21: Train Loss: 0.0980, Val Loss: 0.0131
Epoch 22: Train Loss: 0.0862, Val Loss: 0.0193
Epoch 23: Train Loss: 0.0911, Val Loss: 0.0152
Epoch 24: Train Loss: 0.0953, Val Loss: 0.0133
Epoch 25: Train Loss: 0.0831, Val Loss: 0.0167
Epoch 26: Train Loss: 0.1029, Val Loss: 0.0167
Epoch 27: Train Loss: 0.0922, Val Loss: 0.0162
Epoch 28: Train Loss: 0.0923, Val Loss: 0.0205
Epoch 29: Train Loss: 0.0852, Val Loss: 0.0144
Epoch 30: Train Loss: 0.0885, Val Loss: 0.0186
Epoch 31: Train Loss: 0.0963, Val Loss: 0.0107
Epoch 32: Train Loss: 0.0888, Val Loss: 0.0100

Epoch 33: Train Loss: 0.0674, Val Loss: 0.0164
Epoch 34: Train Loss: 0.0617, Val Loss: 0.0119
Epoch 35: Train Loss: 0.0964, Val Loss: 0.0138
Epoch 36: Train Loss: 0.0826, Val Loss: 0.0094
Epoch 37: Train Loss: 0.0783, Val Loss: 0.0098
Epoch 38: Train Loss: 0.0719, Val Loss: 0.0137
Epoch 39: Train Loss: 0.0689, Val Loss: 0.0119
Epoch 40: Train Loss: 0.0780, Val Loss: 0.0108
Epoch 41: Train Loss: 0.0889, Val Loss: 0.0112
Epoch 42: Train Loss: 0.0730, Val Loss: 0.0095
Epoch 43: Train Loss: 0.0840, Val Loss: 0.0150
Epoch 44: Train Loss: 0.0726, Val Loss: 0.0103
Epoch 45: Train Loss: 0.0875, Val Loss: 0.0094
Epoch 46: Train Loss: 0.0696, Val Loss: 0.0117

Early stopping triggered
Final Test MSE: 1162903.2500

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____ out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: