# =====================================================================

X\_train\_processed = preprocessor.fit\_transform(X\_train).toarray()

X\_test\_processed = preprocessor.transform(X\_test).toarray()

input\_dimension = X\_train\_processed.shape[1]

Y\_train\_array = Y\_train.values

Y\_test\_array = Y\_test.values

def build\_dl\_model(input\_dim):

"""Builds a Multi-Layer Perceptron (MLP) for Regression using Keras."""

model = Sequential([

Dense(128, input\_shape=(input\_dim,), activation='relu'),

Dense(64, activation='relu'),

Dense(1)

])

model.compile(optimizer=Adam(learning\_rate=0.001),

loss='mse',

metrics=['mae'])

return model

dl\_model = build\_dl\_model(input\_dimension)

print("\n--- Training Deep Learning Model (Keras MLP) ---")

dl\_history = dl\_model.fit(

X\_train\_processed, Y\_train\_array,

epochs=10,

batch\_size=32,

validation\_split=0.1,

verbose=0

)

print("Training Complete.")

Y\_dl\_pred = dl\_model.predict(X\_test\_processed).flatten()

dl\_r2 = r2\_score(Y\_test\_array, Y\_dl\_pred)

print("\n--- Deep Learning Performance ---")

print(f"DL R-squared Score (R2): {dl\_r2:.4f}")

plt.figure(figsize=(10, 6))

plt.plot(dl\_history.history['loss'], label='Training Loss')

plt.plot(dl\_history.history['val\_loss'], label='Validation Loss')

plt.title('DL Training Error Progress (Loss vs. Epochs)')

plt.xlabel('Number of Epochs')

plt.ylabel('Loss (Mean Squared Error)')

plt.legend()

plt.grid(True)

plt.show()