MODEL no.2: Regression Model w/ bigger dataset and noise

Initialize Notebook

```
import datetime
print(f"Notebook last run (end-to-end): {datetime.datetime.now()}")

Notebook last run (end-to-end): 2025-09-12 00:58:05.375442
```

1. Import libraries

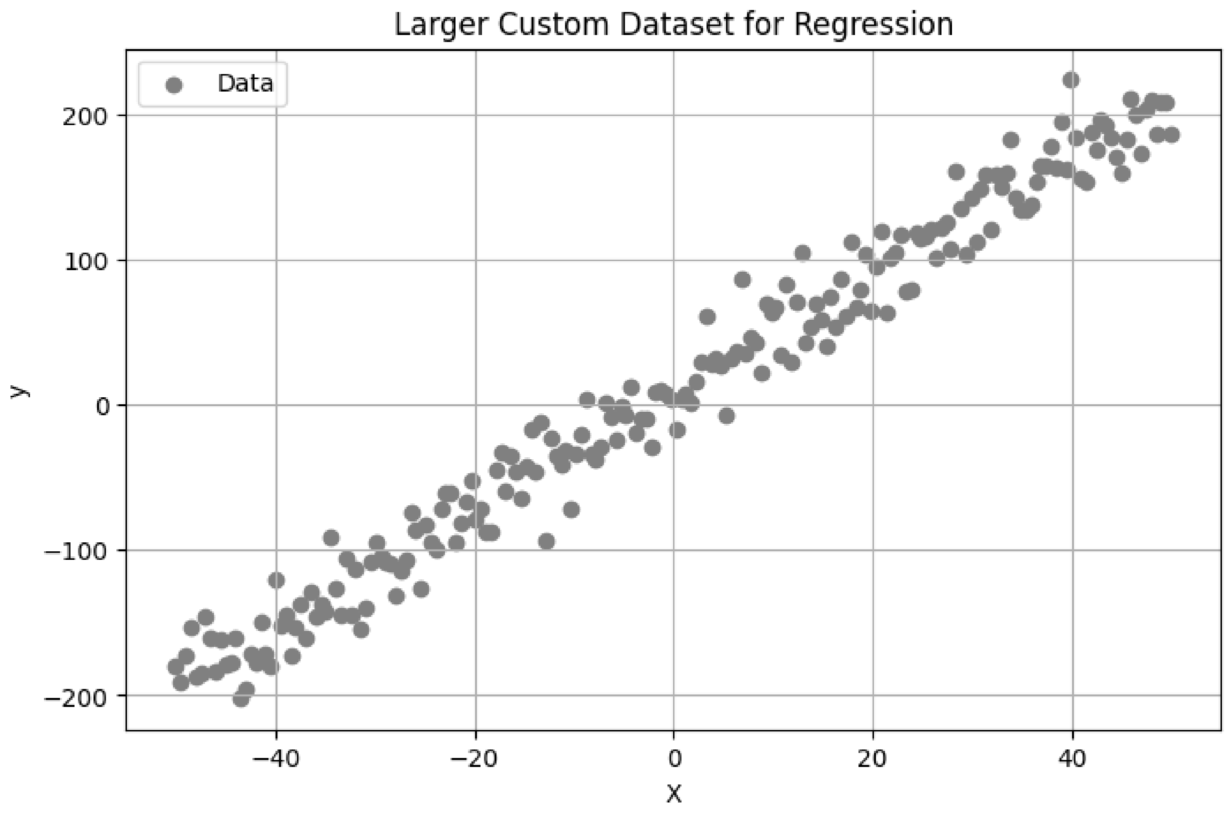
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print(tf.__version__)

2.20.0
```

1. Generate a dataset (using np.arange instead of np.array)

```
np.random.seed(42)
X = np.linspace(-50, 50, 200, dtype=float)
y = 4 * X + 10 + np.random.normal(0, 20, size=len(X))  # Using
np.random to add noise
```

2.1 dataset Visualization using matplotlib

```
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color="gray", label="Data")
plt.title("Larger Custom Dataset for Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.grid(True)
plt.legend()
plt.show()
```



1. Building the model

```
tf.random.set_seed(42)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=[1]),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

```
/opt/anaconda3/envs/MLenv/lib/python3.11/site-packages/keras/src/
layers/core/dense.py:92: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

1. Compile the model

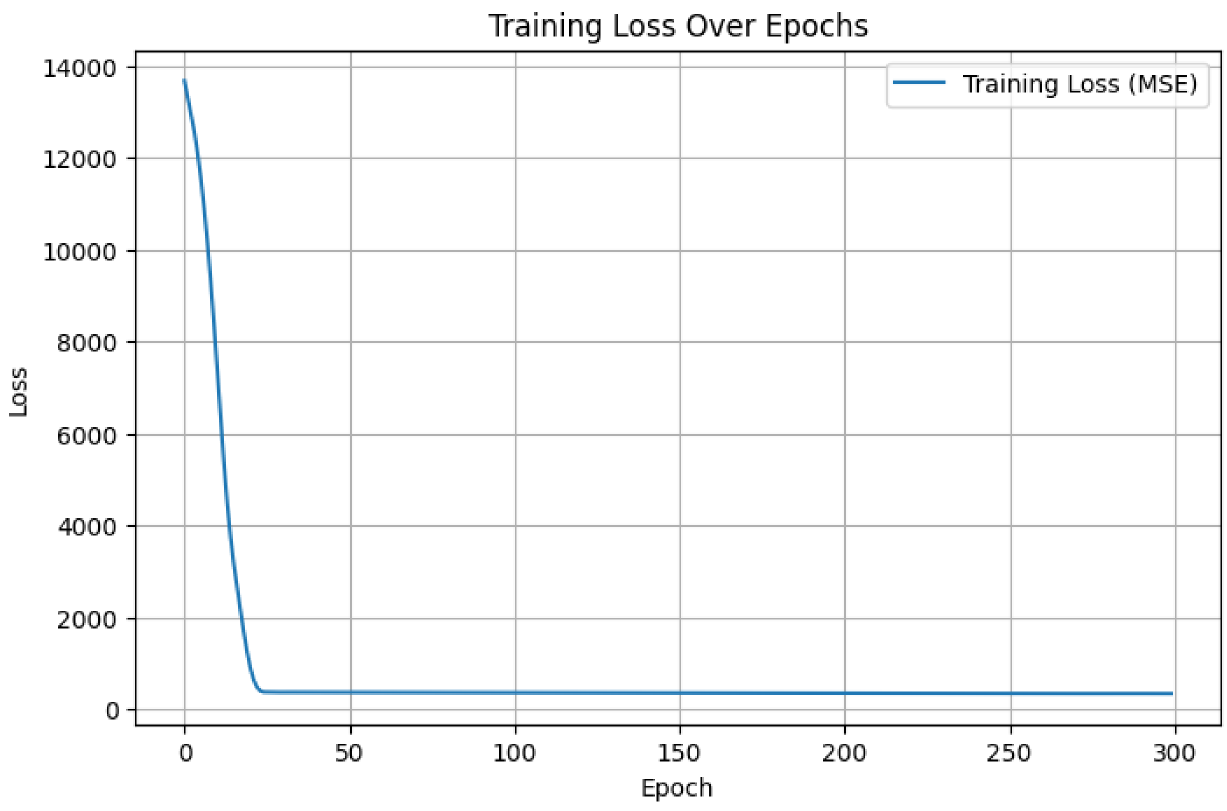
```
model.compile(
    loss=tf.keras.losses.MeanSquaredError(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=["mae"]
)
```

1. Training the model

```
history = model.fit(
    X, y,
    epochs=300,
    batch_size=32,
    verbose=0
)
```

5.1 Visualize training data loss

```
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label="Training Loss (MSE)")
plt.title("Training Loss Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```



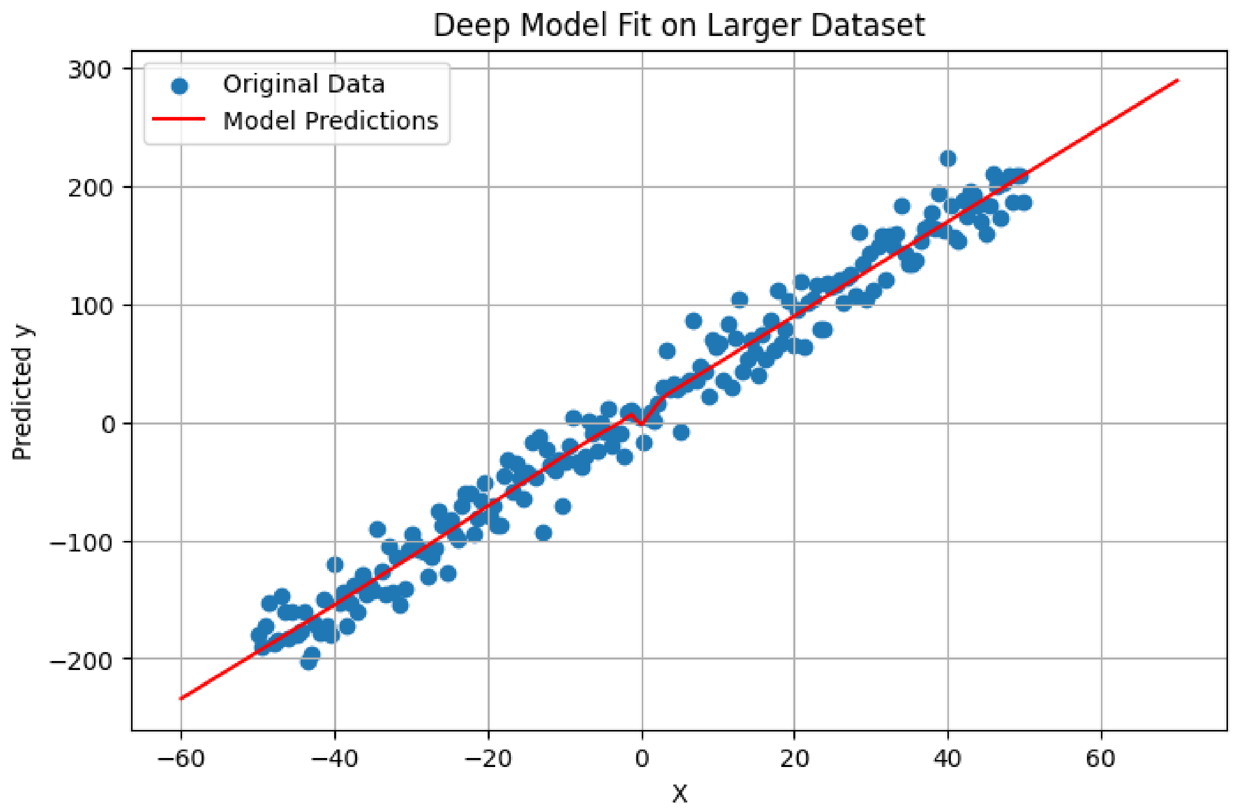
1. Generate predictions

```
# Formula: y ≈ 4(X) + 10
model.predict(np.array([50.0]))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step

array([[209.6662]], dtype=float32)
```

6.1 Visualize predictions

```
X_test = np.linspace(-60, 70, 200)
y_pred = model.predict(X_test)

plt.figure(figsize=(8, 5))
plt.scatter(X, y, label="Original Data")
plt.plot(X_test, y_pred, color="red", label="Model Predictions")
plt.title("Deep Model Fit on Larger Dataset")
plt.xlabel("X")
plt.ylabel("Predicted y")
plt.grid(True)
plt.legend()
plt.show()

7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```


Deep Model Fit on Larger Dataset
Original Data
Model Predictions
Predicted y
X