In [87]:
```
!pip install numpy==2.0.2
!pip install pandas==2.2.2
!pip install tensorflow_cpu==2.18.0
!pip install scikit-learn
```

```
Requirement already satisfied: numpy==2.0.2 in /opt/conda/lib/python3.11/site-pac
kages (2.0.2)
Requirement already satisfied: pandas==2.2.2 in /opt/conda/lib/python3.11/site-pa
ckages (2.2.2)
Requirement already satisfied: numpy>=1.23.2 in /opt/conda/lib/python3.11/site-pa
ckages (from pandas==2.2.2) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.1
1/site-packages (from pandas==2.2.2) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-pac
kages (from pandas==2.2.2) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-p
ackages (from pandas==2.2.2) (2024.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-package
s (from python-dateutil>=2.8.2->pandas==2.2.2) (1.16.0)
Requirement already satisfied: tensorflow_cpu==2.18.0 in /opt/conda/lib/python3.1
1/site-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /opt/conda/lib/python3.11/site-p
ackages (from tensorflow_cpu==2.18.0) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /opt/conda/lib/python3.11/sit
e-packages (from tensorflow_cpu==2.18.0) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /opt/conda/lib/python3.11/
site-packages (from tensorflow_cpu==2.18.0) (24.12.23)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /opt/conda/
lib/python3.11/site-packages (from tensorflow_cpu==2.18.0) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /opt/conda/lib/python3.11/s
ite-packages (from tensorflow_cpu==2.18.0) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /opt/conda/lib/python3.11/site
-packages (from tensorflow_cpu==2.18.0) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /opt/conda/lib/python3.11/sit
e-packages (from tensorflow_cpu==2.18.0) (3.4.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.11/site-packag
es (from tensorflow_cpu==2.18.0) (24.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.2
1.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /opt/conda/lib/python3.11/site-packages (from
tensorflow_cpu==2.18.0) (5.29.2)
Requirement already satisfied: requests<3,>=2.21.0 in /opt/conda/lib/python3.11/s
ite-packages (from tensorflow_cpu==2.18.0) (2.31.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.11/site-packa
ges (from tensorflow_cpu==2.18.0) (69.5.1)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.11/site-pack
ages (from tensorflow_cpu==2.18.0) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/lib/python3.11/site
-packages (from tensorflow_cpu==2.18.0) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /opt/conda/lib/python
3.11/site-packages (from tensorflow_cpu==2.18.0) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /opt/conda/lib/python3.11/site-pa
ckages (from tensorflow_cpu==2.18.0) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /opt/conda/lib/python3.11/s
ite-packages (from tensorflow_cpu==2.18.0) (1.68.1)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /opt/conda/lib/python3.
11/site-packages (from tensorflow_cpu==2.18.0) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /opt/conda/lib/python3.11/site-pac
kages (from tensorflow_cpu==2.18.0) (3.7.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /opt/conda/lib/python3.11/
site-packages (from tensorflow_cpu==2.18.0) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /opt/conda/lib/python3.11/site-pac
kages (from tensorflow_cpu==2.18.0) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /opt/conda/lib/python3.
11/site-packages (from tensorflow_cpu==2.18.0) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /opt/cond
```

```
a/lib/python3.11/site-packages (from tensorflow_cpu==2.18.0) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /opt/conda/lib/python3.11/si
te-packages (from astunparse>=1.6.0->tensorflow_cpu==2.18.0) (0.43.0)
Requirement already satisfied: rich in /opt/conda/lib/python3.11/site-packages (f
rom keras>=3.5.0->tensorflow_cpu==2.18.0) (13.9.4)
Requirement already satisfied: namex in /opt/conda/lib/python3.11/site-packages
(from keras>=3.5.0->tensorflow_cpu==2.18.0) (0.0.8)
Requirement already satisfied: optree in /opt/conda/lib/python3.11/site-packages
(from keras>=3.5.0->tensorflow_cpu==2.18.0) (0.13.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python
3.11/site-packages (from requests<3,>=2.21.0->tensorflow_cpu==2.18.0) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-pac
kages (from requests<3,>=2.21.0->tensorflow_cpu==2.18.0) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.11/si
te-packages (from requests<3,>=2.21.0->tensorflow_cpu==2.18.0) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.11/si
te-packages (from requests<3,>=2.21.0->tensorflow_cpu==2.18.0) (2024.12.14)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.11/site-
packages (from tensorboard<2.19,>=2.18->tensorflow_cpu==2.18.0) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /opt/cond
a/lib/python3.11/site-packages (from tensorboard<2.19,>=2.18->tensorflow_cpu==2.1
8.0) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /opt/conda/lib/python3.11/site-
packages (from tensorboard<2.19,>=2.18->tensorflow_cpu==2.18.0) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /opt/conda/lib/python3.11/sit
e-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow_cpu==2.18.
0) (2.1.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in /opt/conda/lib/python3.1
1/site-packages (from rich->keras>=3.5.0->tensorflow_cpu==2.18.0) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /opt/conda/lib/python3.
11/site-packages (from rich->keras>=3.5.0->tensorflow_cpu==2.18.0) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /opt/conda/lib/python3.11/site-packa
ges (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow_cpu==2.18.0) (0.
1.2)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.11/site-pac
kages (1.6.0)
Requirement already satisfied: numpy>=1.19.5 in /opt/conda/lib/python3.11/site-pa
ckages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /opt/conda/lib/python3.11/site-pac
kages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /opt/conda/lib/python3.11/site-pa
ckages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /opt/conda/lib/python3.11/
site-packages (from scikit-learn) (3.5.0)
```

In [67]:

```python
#Importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from sklearn.metrics import mean_squared_error
```

In [88]:

```python
# Reading the data
url = 'https://cocl.us/concrete_data'
data = pd.read_csv(url)
```

In [96]:
```
# To get the count of rows and columns of the dataset
data.shape
```

Out[96]:
```
(1030, 9)
```

In [97]:
```
# To display top 5 rows of the dataset
data.head()
```

Out[97]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Streng
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.!
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.:
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.:
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.(
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.:

In [98]:
```
# To get the details of datatype for each column and memory required
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Cement              1030 non-null   float64
 1   Blast Furnace Slag  1030 non-null   float64
 2   Fly Ash             1030 non-null   float64
 3   Water               1030 non-null   float64
 4   Superplasticizer    1030 non-null   float64
 5   Coarse Aggregate    1030 non-null   float64
 6   Fine Aggregate      1030 non-null   float64
 7   Age                 1030 non-null   int64  
 8   Strength            1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.6 KB
```

In [99]:
```
# To get overall statistics of the dataset
data.describe()
```

Out[99]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.00000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.91893
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.75395
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.00000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.00000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.00000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.40000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.00000

In [100…

```
# To display null values in each column
data.isnull().sum()
```

Out[100…

```
Cement                0
Blast Furnace Slag    0
Fly Ash               0
Water                 0
Superplasticizer      0
Coarse Aggregate      0
Fine Aggregate        0
Age                   0
Strength              0
dtype: int64
```

In [101…

```
#To display the columns of the dataset
concrete_data_columns = data.columns
data.columns
```

Out[101…

```
Index(['Cement', 'Blast Furnace Slag', 'Fly Ash', 'Water', 'Superplasticizer',
       'Coarse Aggregate', 'Fine Aggregate', 'Age', 'Strength'],
      dtype='object')
```

In [102…

```
#Define the predictors and the Target
predictors = data[concrete_data_columns[concrete_data_columns != 'Strength']]
target = data['Strength']
print(predictors.head())
```

```
   Cement  Blast Furnace Slag  Fly Ash  Water  Superplasticizer  \
0   540.0                 0.0      0.0  162.0               2.5
1   540.0                 0.0      0.0  162.0               2.5
2   332.5               142.5      0.0  228.0               0.0
3   332.5               142.5      0.0  228.0               0.0
4   198.6               132.4      0.0  192.0               0.0

   Coarse Aggregate  Fine Aggregate  Age
0            1040.0           676.0   28
1            1055.0           676.0   28
2             932.0           594.0  270
3             932.0           594.0  365
4             978.4           825.5  360
```

In [103…

```
print(target.head())
```

```
0    79.99
1    61.89
2    40.27
3    41.05
4    44.30
Name: Strength, dtype: float64
```

In [ ]:

```
n_cols = predictors_norm.shape[1]  # Number of predictors
```

In [104…

```
# Step A: Without Normalization and Epoch = 50, Evaluate 50 Times
# Define the regression model with 1 hidden layer
def regression_model():
    model = Sequential()
    model.add(Input(shape=(n_cols,)))  # Specify the input shape
    model.add(Dense(10, activation='relu'))  # Hidden layer with 10 nodes and Re
    model.add(Dense(1))  # Output layer
    model.compile(optimizer='adam', loss='mean_squared_error')  # Compile the mo
    return model

# Initialize an empty list to store the mean squared errors
mse_list = []

# Loop to train and evaluate the model 50 times
for i in range(50):
    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test

    # Build the model
    model = regression_model()

    # Train the model on the training data
    model.fit(X_train, y_train, epochs=50, verbose=0)

    # Evaluate the model on the test data
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    mse_list.append(mse)

mean_mse_A = np.mean(mse_list)
std_mse_A = np.std(mse_list)

print(f"Mean of Mean Squared Errors (Step A - No Normalization, 50 epochs): {mea
print(f"Standard Deviation of Mean Squared Errors (Step A - No Normalization, 50
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Mean of Mean Squared Errors (Step A - No Normalization, 50 epochs): 303.630609049
2363
Standard Deviation of Mean Squared Errors (Step A - No Normalization, 50 epochs):
233.7104445832209
```

In [105…

```python
# Step B: With Normalization and Epoch = 50, Evaluate 50 Times, Compare with Ste

# Normalize the predictors
predictors_norm = (predictors - predictors.mean()) / predictors.std()
```

```
# Initialize an empty list to store the mean squared errors
mse_list = []

# Loop to train and evaluate the model 50 times
for i in range(50):
    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target,

    # Build the model
    model = regression_model()

    # Train the model on the training data
    model.fit(X_train, y_train, epochs=50, verbose=0)

    # Evaluate the model on the test data
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    mse_list.append(mse)

mean_mse_B = np.mean(mse_list)
std_mse_B = np.std(mse_list)

print(f"Mean of Mean Squared Errors (Step B - With Normalization, 50 epochs): {m
print(f"Standard Deviation of Mean Squared Errors (Step B - With Normalization,
print(f"Difference in Mean Squared Errors between Step A and Step B: {mean_mse_A
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Mean of Mean Squared Errors (Step B - With Normalization, 50 epochs): 370.2345658
49912
Standard Deviation of Mean Squared Errors (Step B - With Normalization, 50 epoch
s): 107.75069223008683
Difference in Mean Squared Errors between Step A and Step B: -66.60395680067569
```

In [106…

```python
# Step C: With Normalization and Epoch = 100, Compare with Step B

# Initialize an empty list to store the mean squared errors
mse_list = []
```

```
# Loop to train and evaluate the model 50 times
for i in range(50):
    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target,

    # Build the model
    model = regression_model()

    # Train the model on the training data
    model.fit(X_train, y_train, epochs=100, verbose=0)

    # Evaluate the model on the test data
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    mse_list.append(mse)

mean_mse_C = np.mean(mse_list)
std_mse_C = np.std(mse_list)

print(f"Mean of Mean Squared Errors (Step C - With Normalization, 100 epochs): {
print(f"Standard Deviation of Mean Squared Errors (Step C - With Normalization,
print(f"Difference in Mean Squared Errors between Step B and Step C: {mean_mse_B
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
Mean of Mean Squared Errors (Step C - With Normalization, 100 epochs): 165.212853
79928682
Standard Deviation of Mean Squared Errors (Step C - With Normalization, 100 epoch
s): 18.51426082827909
Difference in Mean Squared Errors between Step B and Step C: 205.02171205062515
```

In [107…

```python
# Define the regression model with three hidden layers
def regression_model_increased_layers():
    model = Sequential()
    model.add(Input(shape=(predictors_norm.shape[1],)))  # Specify the input sha
```

```
    model.add(Dense(10, activation='relu'))  # First hidden layer with 10 nodes
    model.add(Dense(10, activation='relu'))  # Second hidden layer with 10 nodes
    model.add(Dense(10, activation='relu'))  # Third hidden layer with 10 nodes
    model.add(Dense(1))  # Output layer
    model.compile(optimizer='adam', loss='mean_squared_error')  # Compile the mo
    return model

# Initialize an empty list to store the mean squared errors
mse_list = []

# Loop to train and evaluate the model 50 times
for i in range(50):
    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target,

    # Build the model
    model = regression_model_increased_layers()

    # Train the model on the training data
    model.fit(X_train, y_train, epochs=50, verbose=0)

    # Evaluate the model on the test data
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    mse_list.append(mse)

mean_mse_D = np.mean(mse_list)
std_mse_D = np.std(mse_list)

print(f"Mean of Mean Squared Errors (Step D - Increased Layers, With Normalizati
print(f"Standard Deviation of Mean Squared Errors (Step D - Increased Layers, Wi
print(f"Difference in Mean Squared Errors between Step B and Step D: {mean_mse_B
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Mean of Mean Squared Errors (Step D - Increased Layers, With Normalization, 50 epochs): 125.87031561843013
Standard Deviation of Mean Squared Errors (Step D - Increased Layers, With Normalization, 50 epochs): 16.765422916874382
Difference in Mean Squared Errors between Step B and Step D: 244.36425023148183
```