Final Project: Extra Credit
Group 4: Gaurav Goyal

Summary:

The provided code in “Appendix A: Code” performs data imputation using two methods in the TensorFlow environment: Multiple Imputation by Chained Equations (MICE) and a Neural Network-based approach. Here's a summary of the code and the results:

A. Data Preprocessing:

- The dataset containing parts and dimensions is loaded.
- Missing values are removed from columns 'Length', 'Width', and 'Height'.
- Outliers are detected and removed using the Isolation Forest algorithm.
- The data is normalized using Min-Max Scaling to ensure consistent ranges across features.

B. Data Imputation:

- **MICE Imputation:**
  - The IterativeImputer from scikit-learn is used to impute missing values based on other available features.
  - The imputed results are displayed in tabular form.

- **Neural Network Imputation:**
  - A neural network model is trained using available data to predict missing 'Height' values based on 'Length' and 'Width'.
  - The trained model is used to predict missing 'Height' values.
  - The imputed results are displayed in tabular form.

C. Results Summary:

- **MICE Imputation Results:**
  - The imputed 'Height' values are shown in “Table 1: MICE Imputation Results” for several data points along with their corresponding 'Length' and 'Width' values.

Final Project: Extra Credit
Group 4: Gaurav Goyal

Table 1: MICE Imputation Results

MICE Imputation Results:

	Length	Width	Height
280	0.435308	0.448095	0.509155
440	0.181983	0.505913	0.503330
59	0.187424	0.494087	0.503824
290	0.233374	0.691196	0.497630
162	0.537485	0.705650	0.501695

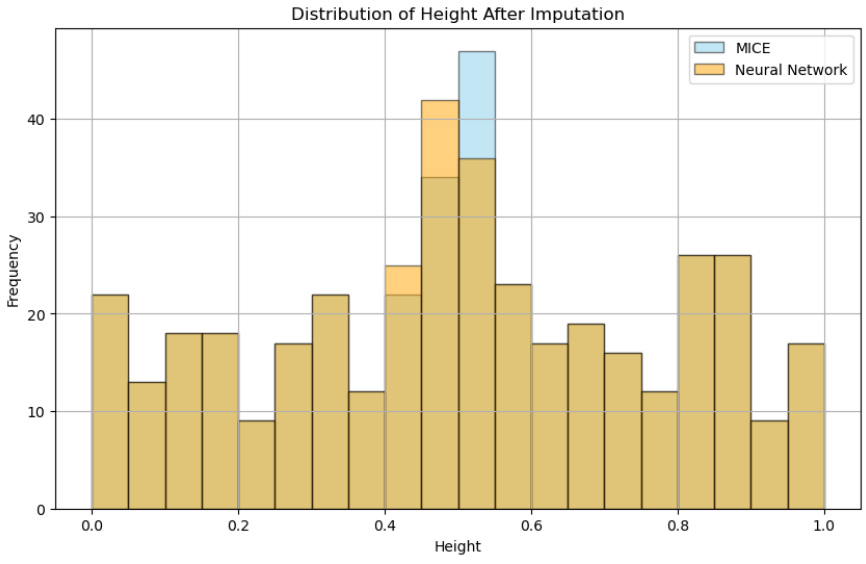
- **Neural Network Imputation Results:**
  - The imputed 'Height' values are shown in Table 2: Neural Network Imputation Results.

Table 2: Neural Network Imputation Results

Neural Network Imputation Results:

	Item_No	Length	Width	Height
280	281	0.435308	0.448095	0.518871
440	441	0.181983	0.505913	0.454741
59	60	0.187424	0.494087	0.456336
290	291	0.233374	0.691196	0.460492
162	163	0.537485	0.705650	0.479646

Both MICE and neural network-based imputation methods effectively imputed missing 'Height' values.



Final Project: Extra Credit
Group 4: Gaurav Goyal

Appendix A: Code

In [2]:
```
import os

# Define the path to the directory you want to switch to
new_directory = "C:\\Users\gaurav.goyal\Downloads"

# Change to the new directory
os.chdir(new_directory)

# Check the current working directory
print("Current Working Directory:", os.getcwd())
```

```
Current Working Directory: C:\Users\gaurav.goyal\Downloads
```

In [3]:
```
from sklearn.experimental import enable_iterative_imputer
```

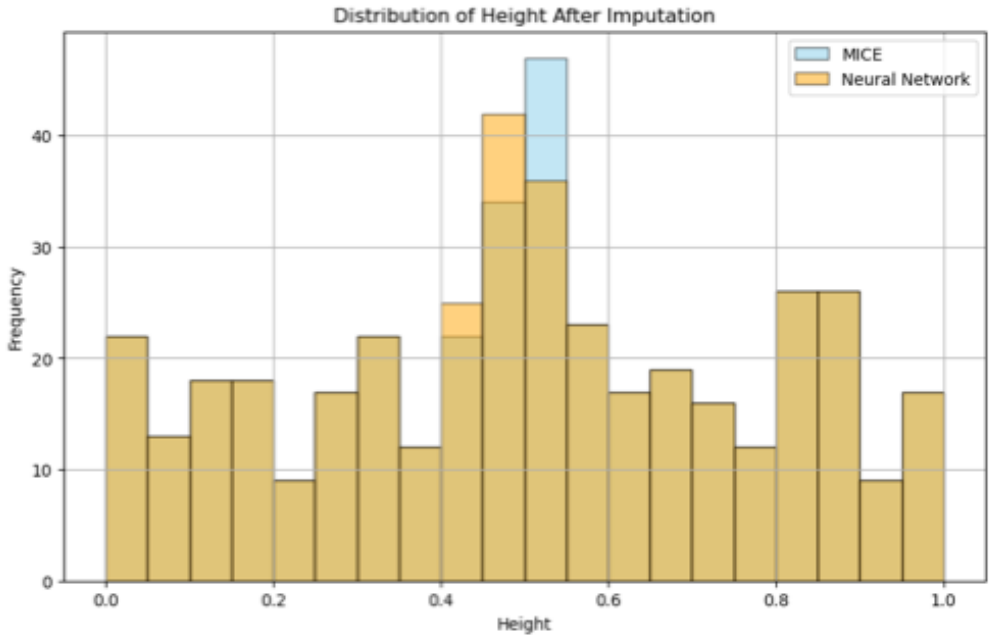
In [5]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import MinMaxScaler
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
import tensorflow as tf
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_excel('Parts and Dimesions.xlsx')

# Step 1: Remove Missing Data
data_cleaned = data.dropna(subset=['Length', 'Width', 'Height'])

# Step 2: Remove Outliers using Isolation Forest
isolation_forest = IsolationForest(random_state=42)
outliers = isolation_forest.fit_predict(data_cleaned[['Length', 'Width', 'Hei
data_no_outliers = data_cleaned[outliers == 1]

# Step 3: Normalize Data using Min-Max Scaling
scaler = MinMaxScaler()
data_no_outliers.loc[:, ['Length', 'Width', 'Height']] = scaler.fit_transform

# Simulate missing data in 'Height' for demonstration
np.random.seed(42)
missing_indices = np.random.choice(data_no_outliers.index, size=int(0.1 * len
data_no_outliers.loc[missing_indices, 'Height'] = np.nan

# Data Imputation using MICE
imputer_mice = IterativeImputer(random_state=42)
data_mice_imputed = imputer_mice.fit_transform(data_no_outliers[['Length', 'W

# Convert imputed data back to DataFrame
```

```
data_mice_imputed_df = pd.DataFrame(data_mice_imputed, columns=['Length', 'Wi

# Data Imputation using Neural Network
# Prepare the dataset for training the imputation model
train_data = data_no_outliers.dropna(subset=['Height'])
X_train = train_data[['Length', 'Width']]
y_train = train_data['Height']

# Define the neural network model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(2,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Set up early stopping to prevent overfitting
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=5)

# Train the model
model.fit(X_train, y_train, epochs=100, callbacks=[early_stopping], verbose=1

# Predicting the missing 'Height' values
predicted_heights = model.predict(data_no_outliers.loc[missing_indices, ['Len

# Filling in the missing 'Height' values in the original dataset
data_nn_imputed = data_no_outliers.copy()
data_nn_imputed.loc[missing_indices, 'Height'] = predicted_heights.ravel()

# Display the imputation results in table form
print("MICE Imputation Results:")
print(data_mice_imputed_df.loc[missing_indices].head())
print("\nNeural Network Imputation Results:")
print(data_nn_imputed.loc[missing_indices].head())

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

# Histogram for MICE-imputed data
plt.hist(data_mice_imputed_df['Height'], bins=20, color='skyblue', edgecolor=

# Histogram for neural network-imputed data
plt.hist(data_nn_imputed['Height'], bins=20, color='orange', edgecolor='black

plt.title('Distribution of Height After Imputation')
plt.xlabel('Height')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.show()
```

```
Epoch 1/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - loss: 0.5000
Epoch 2/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2244
Epoch 3/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1091
Epoch 4/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0887
Epoch 5/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0933
Epoch 6/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0886
Epoch 7/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0878
Epoch 8/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0862
Epoch 9/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0765
Epoch 10/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0883
Epoch 11/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0827
Epoch 12/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0869
Epoch 13/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0784
Epoch 14/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0888
Epoch 15/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0831
Epoch 16/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0837
Epoch 17/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0798
Epoch 18/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0762
Epoch 19/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0818
Epoch 20/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0830
Epoch 21/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0803
Epoch 22/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0858
Epoch 23/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0796
Epoch 24/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0819
Epoch 25/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0737
Epoch 26/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0746
```

```
Epoch 27/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0823
Epoch 28/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0770
Epoch 29/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0769
Epoch 30/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0777
Epoch 31/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0829
Epoch 32/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0774
Epoch 33/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0787
Epoch 34/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0789
Epoch 35/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0804
Epoch 36/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0726
Epoch 37/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0843
Epoch 38/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0822
Epoch 39/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0766
Epoch 40/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0784
Epoch 41/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0751
Epoch 42/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0732
Epoch 43/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0859
Epoch 44/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0840
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
MICE Imputation Results:
       Length     Width    Height
280  0.435308  0.448095  0.509155
440  0.181983  0.505913  0.503330
59   0.187424  0.494087  0.503824
290  0.233374  0.691196  0.497630
162  0.537485  0.705650  0.501695

Neural Network Imputation Results:
     Item_No    Length     Width    Height Operator
280      281  0.435308  0.448095  0.518871    Op-11
440      441  0.181983  0.505913  0.454741    Op-18
59        60  0.187424  0.494087  0.456336     Op-3
290      291  0.233374  0.691196  0.460492    Op-12
162      163  0.537485  0.705650  0.479646     Op-7
```

Final Project: Extra Credit
Group 4: Gaurav Goyal



In [7]:
```
print("NaNs in MICE-imputed data:", data_mice_imputed_df.loc[missing_indices,
print("NaNs in Neural Network-imputed data:", np.isnan(predicted_heights).sum
```

```
NaNs in MICE-imputed data: 0
NaNs in Neural Network-imputed data: 0
```

In [ ]: