IN_SYS SW10, HS2025 - Work 1

Data Standardization vs. Normalization

2025-11-17, Eugen Rodel

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

In [2]:

```
# Generating a new dataset with Gaussian distribution
np.random.seed(42)
data_gaussian = {
    'Feature 1': np.random.normal(loc=500, scale=100, size=100),
    'Feature 2': np.random.normal(loc=300, scale=50, size=100)
}
```

The random function for Feature 2 above generates an array of 100 random numbers drawn from a normal (Gaussian) distribution with the following parameters:

- **loc=300:** This is the mean (center) of the distribution). The generated numbers will be centered around 300.
- **scale=50:** This is the standard deviation (spread or width) of the distribution). It determines how spread out the values are around the mean. A -higher value results in a wider distribution.
- **size=100:** This indicates the number of random numbers to generate, which in this case is 100.

The result is an array of 100 random values that are distributed around a mean of 300, with a standard deviation of 50, simulating a normal distribution.

In [3]:

```
# create a Pandas Data Frame from the numpy array
df_gaussian = pd.DataFrame(data_gaussian)

df_gaussian.head()
```

Out[3]:

	Feature 1	Feature 2
0	549.671415	229.231463
1	486.173570	278.967734
2	564.768854	282.864274
3	652.302986	259.886137
4	476.584663	291.935714

In [4]:

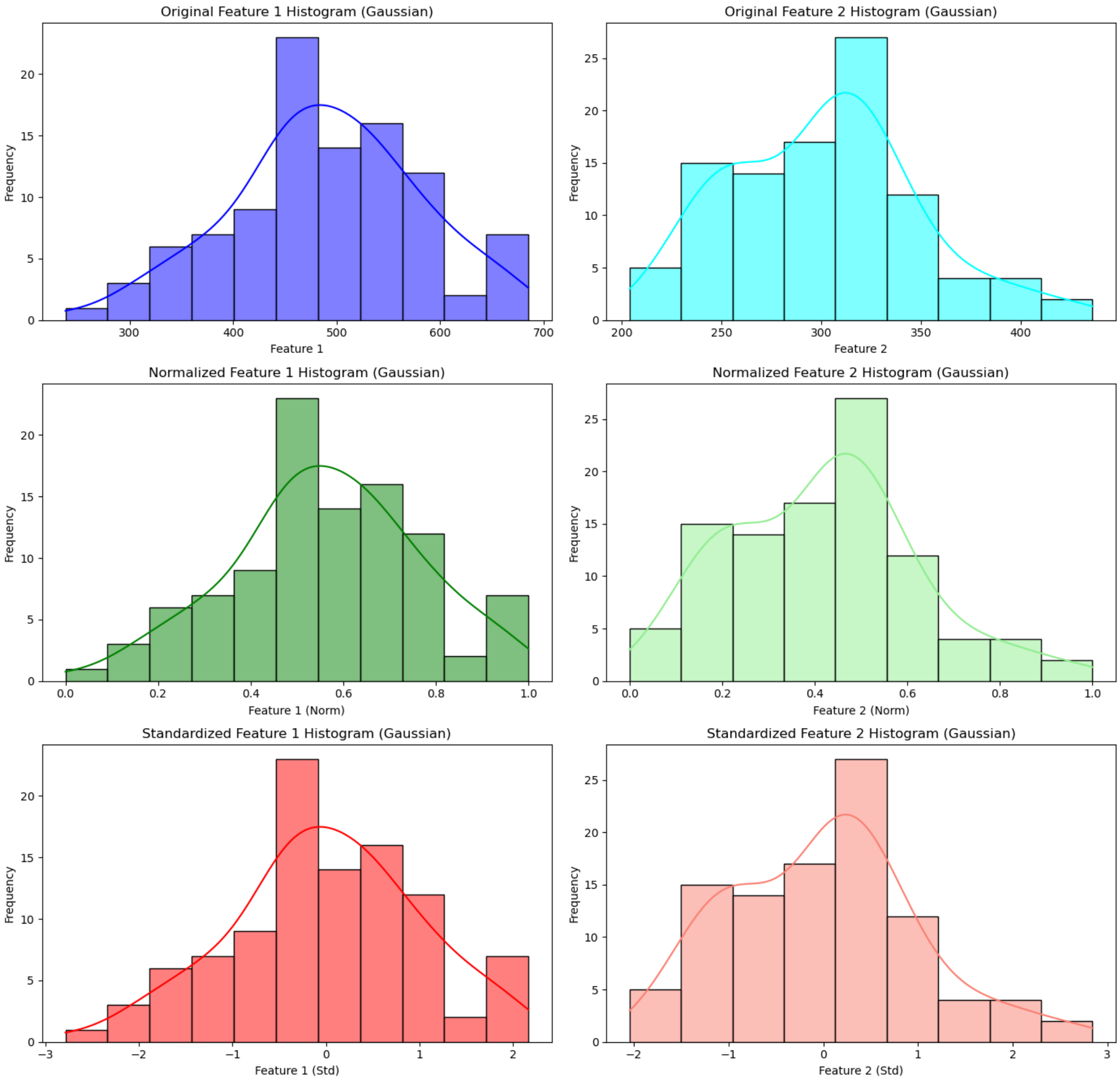
```
# Initializing the scalers
```

```
scaler_minmax = MinMaxScaler()
scaler_standard = StandardScaler()

# Applying Normalization and Standardization
normalized_data_gaussian = scaler_minmax.fit_transform(df_gaussian)
standardized_data_gaussian = scaler_standard.fit_transform(df_gaussian)
```

In [5]:

```
# Creating DataFrames for visualization
df_gaussian_normalized = pd.DataFrame(normalized_data_gaussian, columns=['Featur
df_gaussian_standardized = pd.DataFrame(standardized_data_gaussian, columns=['Fe

# Visualizing histograms for Feature 1 and Feature 2 independently
fig, axes = plt.subplots(3, 2, figsize=(14, 15))
fig.suptitle('Histograms for Gaussian Distributed Data: Original, Normalized, an

# Original Feature 1 Histogram
sns.histplot(df_gaussian['Feature 1'], ax=axes[0, 0], color='blue', kde=True)
axes[0, 0].set_title('Original Feature 1 Histogram (Gaussian)')
axes[0, 0].set_xlabel('Feature 1')
axes[0, 0].set_ylabel('Frequency')

# Original Feature 2 Histogram
sns.histplot(df_gaussian['Feature 2'], ax=axes[0, 1], color='cyan', kde=True)
axes[0, 1].set_title('Original Feature 2 Histogram (Gaussian)')
axes[0, 1].set_xlabel('Feature 2')
axes[0, 1].set_ylabel('Frequency')

# Normalized Feature 1 Histogram
sns.histplot(df_gaussian_normalized['Feature 1 (Norm)'], ax=axes[1, 0], color='g
axes[1, 0].set_title('Normalized Feature 1 Histogram (Gaussian)')
axes[1, 0].set_xlabel('Feature 1 (Norm)')
axes[1, 0].set_ylabel('Frequency')

# Normalized Feature 2 Histogram
sns.histplot(df_gaussian_normalized['Feature 2 (Norm)'], ax=axes[1, 1], color='l
axes[1, 1].set_title('Normalized Feature 2 Histogram (Gaussian)')
axes[1, 1].set_xlabel('Feature 2 (Norm)')
axes[1, 1].set_ylabel('Frequency')

# Standardized Feature 1 Histogram
sns.histplot(df_gaussian_standardized['Feature 1 (Std)'], ax=axes[2, 0], color='
axes[2, 0].set_title('Standardized Feature 1 Histogram (Gaussian)')
axes[2, 0].set_xlabel('Feature 1 (Std)')
axes[2, 0].set_ylabel('Frequency')

# Standardized Feature 2 Histogram
sns.histplot(df_gaussian_standardized['Feature 2 (Std)'], ax=axes[2, 1], color='
axes[2, 1].set_title('Standardized Feature 2 Histogram (Gaussian)')
axes[2, 1].set_xlabel('Feature 2 (Std)')
axes[2, 1].set_ylabel('Frequency')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Histograms for Gaussian Distributed Data: Original, Normalized, and Standardized



In [ ]: