Gaussian Distribution Transformation Demo

This notebook demonstrates the transformation of Gaussian distribution data through linear and non-linear transformations.

Gaussian Distribution

We start with a Gaussian distribution with mean (\mu = 0) and standard deviation (\sigma = 1).

```python

Parameters for the original Gaussian distribution

mu_original, sigma_original = 0, 1

Generate Gaussian data

original_data = generate_gaussian_data(mu_original, sigma_original)

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Function to generate Gaussian data
def generate_gaussian_data(mu, sigma, size=1000):
    return np.random.normal(mu, sigma, size)

# Function to perform linear transformation
def linear_transformation(data, slope, intercept):
    return slope * data + intercept

# Function to perform non-linear transformation (quadratic)
def nonlinear_transformation(data):
    poly = PolynomialFeatures(degree=2, include_bias=False)
    transformed_data = poly.fit_transform(data.reshape(-1, 1))
    return transformed_data[:, 1]  # Use the quadratic term

# Parameters for the original Gaussian distribution
mu_original, sigma_original = 0, 1

# Generate Gaussian data
original_data = generate_gaussian_data(mu_original, sigma_original)

# Linear transformation parameters
slope = 2
intercept = 3

# Apply linear transformation
linear_transformed_data = linear_transformation(original_data, slope, intercept)

# Apply non-linear transformation (quadratic)
nonlinear_transformed_data = nonlinear_transformation(original_data)

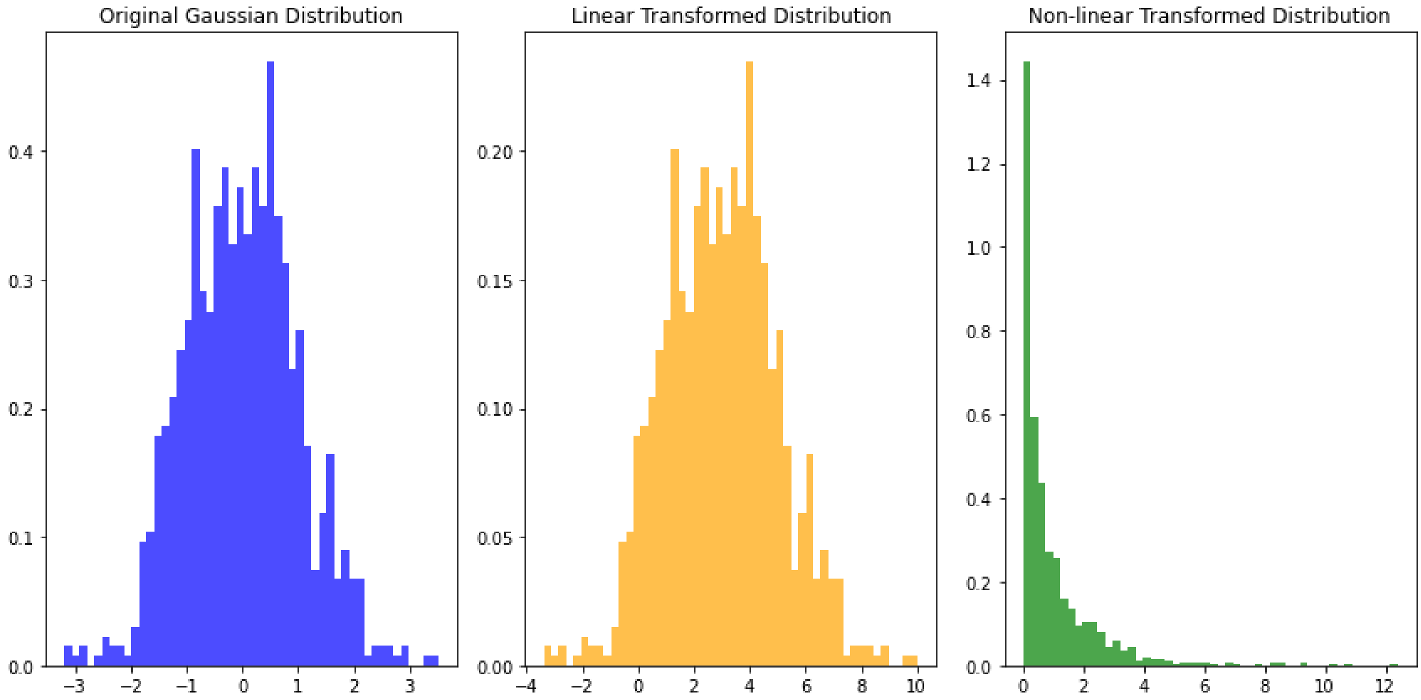
# Plot the original, linear-transformed, and non-linear-transformed data
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.hist(original_data, bins=50, density=True, alpha=0.7, color='blue')
plt.title('Original Gaussian Distribution')
```

```
plt.subplot(1, 3, 2)
plt.hist(linear_transformed_data, bins=50, density=True, alpha=0.7, color='orange')
plt.title('Linear Transformed Distribution')

plt.subplot(1, 3, 3)
plt.hist(nonlinear_transformed_data, bins=50, density=True, alpha=0.7, color='green')
plt.title('Non-linear Transformed Distribution')

plt.tight_layout()
plt.show()
```



Linear Transformation

Next, we perform a linear transformation on the Gaussian data using the formula (y = mx + b), where (m) is the slope and (b) is the intercept.

```python

Linear transformation parameters

slope = 2 intercept = 3

Apply linear transformation

linear_transformed_data = linear_transformation(original_data, slope, intercept)

In [2]:

```
# Linear transformation parameters
slope = 2
intercept = 3

# Apply linear transformation
linear_transformed_data = linear_transformation(original_data, slope, intercept)
```

Non-linear Transformation

We also apply a non-linear transformation using a quadratic function (y = ax^2 + bx + c).

```python

Apply non-linear transformation (quadratic)

nonlinear_transformed_data = nonlinear_transformation(original_data)

In [3]:

```
# Apply non-linear transformation (quadratic)
nonlinear_transformed_data = nonlinear_transformation(original_data)
```