PARAMETRIC DATA EXPLORATION

Aim: The aim of this parametric data exploration is to analyze data to understand trends and factors influencing outcomes.

1. DATA COLLECTION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm

data = pd.read_csv('diabetes.csv')
data
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

PARAMETRIC DATA EXPLORATION

2. DESCRIPTIVE STATISTICS

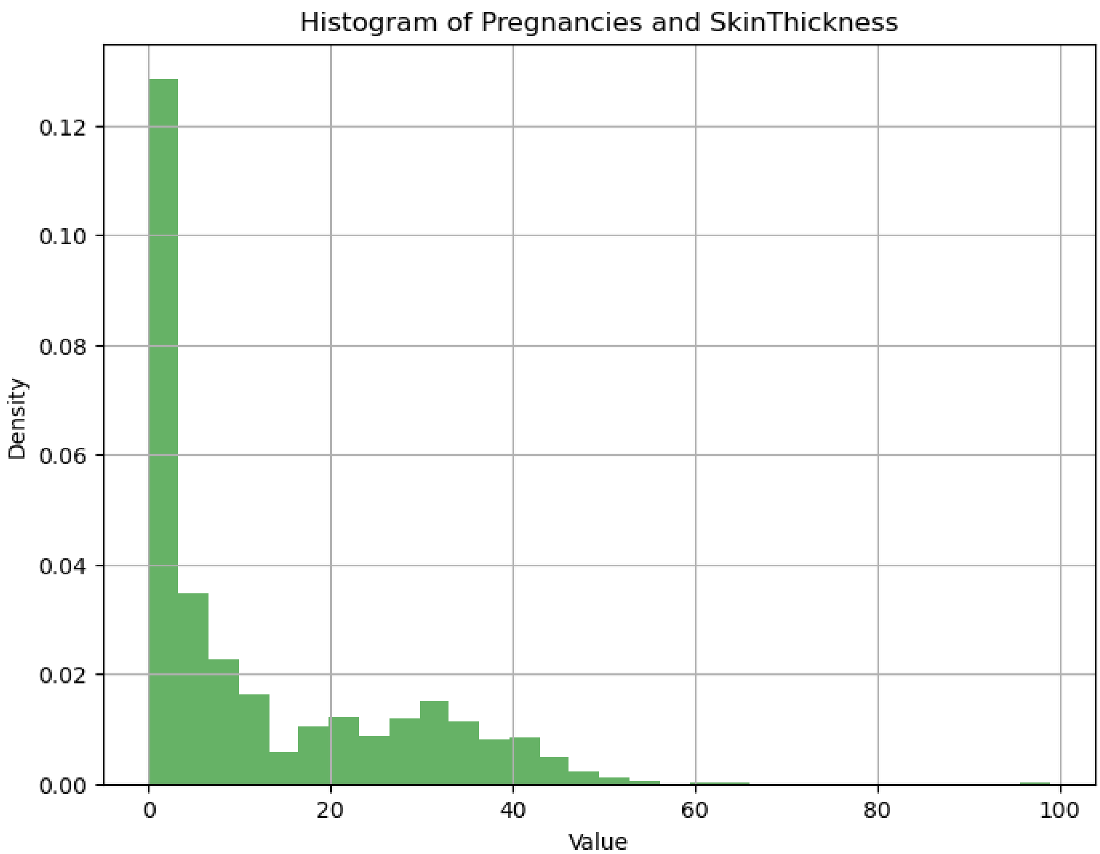
```
Outcome = data['Outcome']
mean = np.mean(Outcome)
std_dev = np.std(Outcome)
skewness = np.mean((Outcome - mean) ** 3) / (std_dev ** 3)
kurtosis = np.mean((Outcome - mean) ** 4) / (std_dev ** 4) - 3

print("Descriptive Statistics:")
print(f"Mean: {mean:.2f}")
print(f"Standard Deviation: {std_dev:.2f}")
print(f"Skewness: {skewness:.2f}")
print(f"Kurtosis: {kurtosis:.2f}")
```

```
Descriptive Statistics:
Mean: 0.35
Standard Deviation: 0.48
Skewness: 0.63
Kurtosis: -1.60
```

PARAMETRIC DATA EXPLORATION

3. HISTOGRAMS AND DENSITY PLOTS ON THE DATA

```
combined_data = pd.concat([data['Pregnancies'], data['SkinThickness']])

plt.figure(figsize=(8, 6))
plt.hist(combined_data, bins=30, density=True, alpha=0.6, color='g')
plt.title('Histogram of Pregnancies and SkinThickness')
plt.xlabel('Value')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```



PARAMETRIC DATA EXPLORATION

4.PARAMETRIC DISTRIBUTION FITTING ON THE DATA

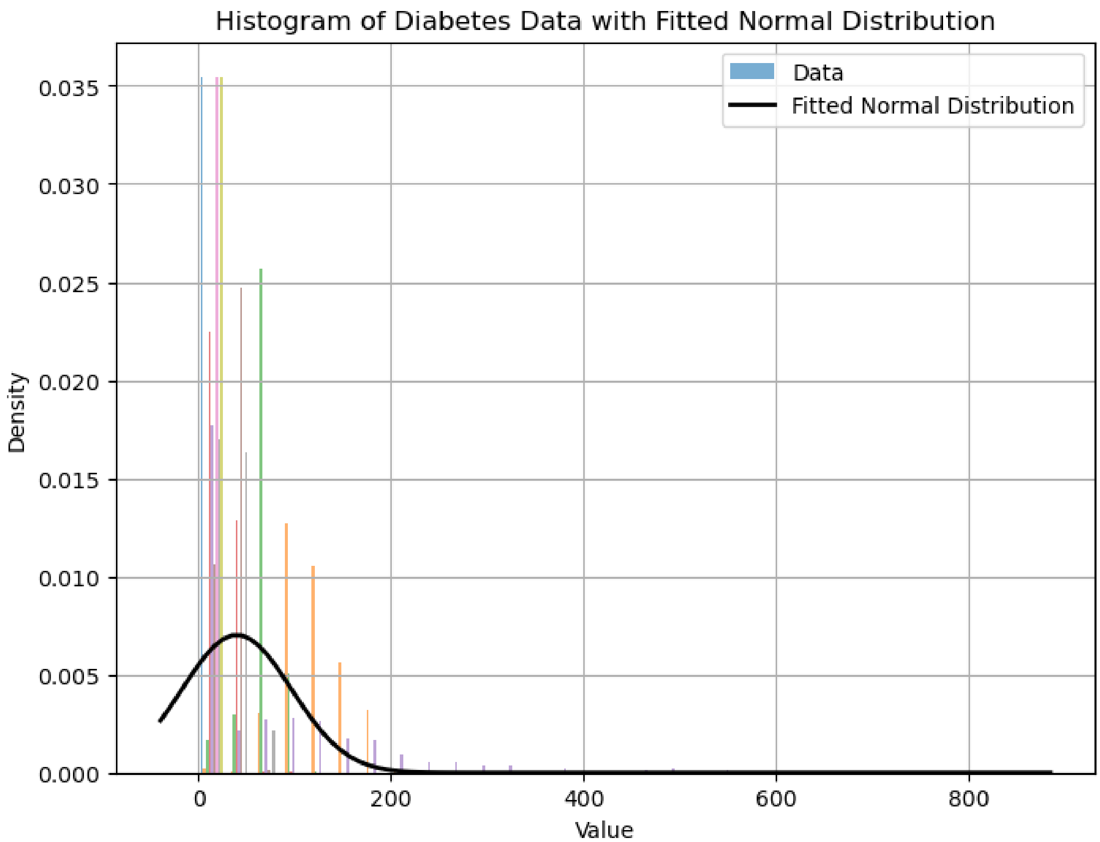
```python
from scipy.stats import shapiro

# Shapiro-Wilk Test
statistic, p_value = shapiro(data)

print("Shapiro-Wilk Test:")
print(f"Statistic: {statistic:.4f}")
print(f"P-value: {p_value:.4f}")

alpha = 0.05
if p_value > alpha:
    print("Accept the null hypothesis")
    print("The data appears to be normally distributed")
else:
    print("Reject the null hypothesis")
    print("The data does not appear to be normally distributed")
```

```
Shapiro-Wilk Test:
Statistic: 0.6918
P-value: 0.0000
Reject the null hypothesis
The data does not appear to be normally distributed
```

5. GOODNESS-OF-FIT TEST

```python
from scipy.stats import norm

# Fitting a normal distribution to the data
mu, sigma = norm.fit(data)

#histogram of the data
plt.figure(figsize=(8, 6))
plt.hist(data, bins=30, density=True, alpha=0.6, label='Data')

#PDF of the fitted normal distribution
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, sigma)
plt.plot(x, p, 'k', linewidth=2, label='Fitted Normal Distribution')
plt.title('Histogram of Diabetes Data with Fitted Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()

#parameters of the fitted normal distribution
print("Parameters of the Fitted Normal Distribution:")
print(f"Mean: {mu:.2f}")
print(f"Standard Deviation: {sigma:.2f}")
```

PARAMETRIC DATA EXPLORATION




```
Parameters of the Fitted Normal Distribution:
Mean: 40.03
Standard Deviation: 56.79
```

6. PARAMETER ESTIMATION & CALCULATING THE CONFIDENCE INTERVALS

```
from scipy.stats import norm
# Fit a normal distribution to the data
mu, sigma = norm.fit(data)

# Parameter Estimation
print("Parameter Estimation:")
print(f"Estimated Mean (mu): {mu:.2f}")
print(f"Estimated Standard Deviation (sigma): {sigma:.2f}")

n = len(data)
standard_error_mean = sigma / np.sqrt(n)
standard_error_std_dev = sigma / np.sqrt(2 * (n - 1))
confidence_interval_mean = norm.interval(0.95, loc=mu,
scale=standard_error_mean)
confidence_interval_std_dev = norm.interval(0.95, loc=sigma,
scale=standard_error_std_dev)

print("\nConfidence Intervals:")
print(f"95% Confidence Interval for Mean (mu): {confidence_interval_mean}")
print(f"95% Confidence Interval for Standard Deviation (sigma):
{confidence_interval_std_dev}")
```

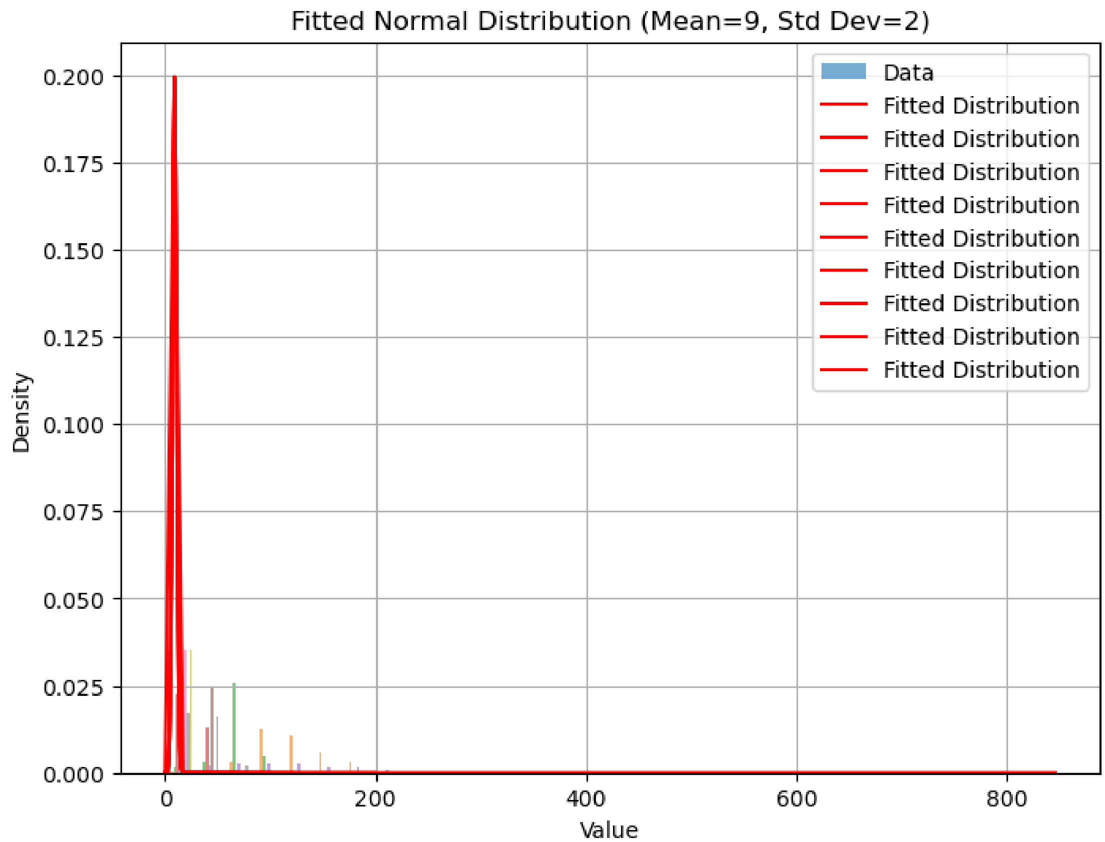
```
Parameter Estimation:
Estimated Mean (mu): 40.03
Estimated Standard Deviation (sigma): 56.79

Confidence Intervals:
95% Confidence Interval for Mean (mu): (36.0094038026056, 44.042882366375885)
95% Confidence Interval for Standard Deviation (sigma): (53.952370044773446,
59.636599089113005)
```

7. HYPOTHESIS TESTING AND SENSITIVITY ANALYSIS

```
from scipy import stats

t_statistic, p_value = stats.ttest_1samp(data, 10)
print("\nHypothesis Testing:")
print(f"T-Statistic: {t_statistic[0]:.4f}, p-value: {p_value[0]:.4f}")

# The t-statistic measures how-many-standard-errors the sample mean is from
the null hypothesis mean (in this case, 10).
# Compare the p-value to a significance level (such as 0.05) to determine
whether the null hypothesis can be rejected.
# If the p-value is less than the significance level (such as 0.05), the null
hypothesis is rejected, suggesting that
# the sample mean is significantly different from 10.

# Sensitivity Analysis
varying_parameters = [(9, 2), (10, 3), (10, 2), (11, 2)]  # Vary mean and
standard deviation

for params in varying_parameters:
    mu_variation, sigma_variation = params
    fitted_distribution = norm(mu_variation, sigma_variation)

    plt.figure(figsize=(8, 6))
    plt.hist(data, bins=30, density=True, alpha=0.6, label='Data')
    x = np.linspace(data.min(), data.max(), 100)  # Modify this line
    plt.plot(x, fitted_distribution.pdf(x), 'r-', label='Fitted
Distribution')
    plt.title(f'Fitted Normal Distribution (Mean={mu_variation}, Std
Dev={sigma_variation})')
    plt.xlabel('Value')
    plt.ylabel('Density')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(f"\nParameters: Mean={mu_variation}, Std Dev={sigma_variation}")
    print(f"Descriptive Statistics:")
    print(f"Mean: {mu_variation:.2f}")
    print(f"Standard Deviation: {sigma_variation:.2f}")
```

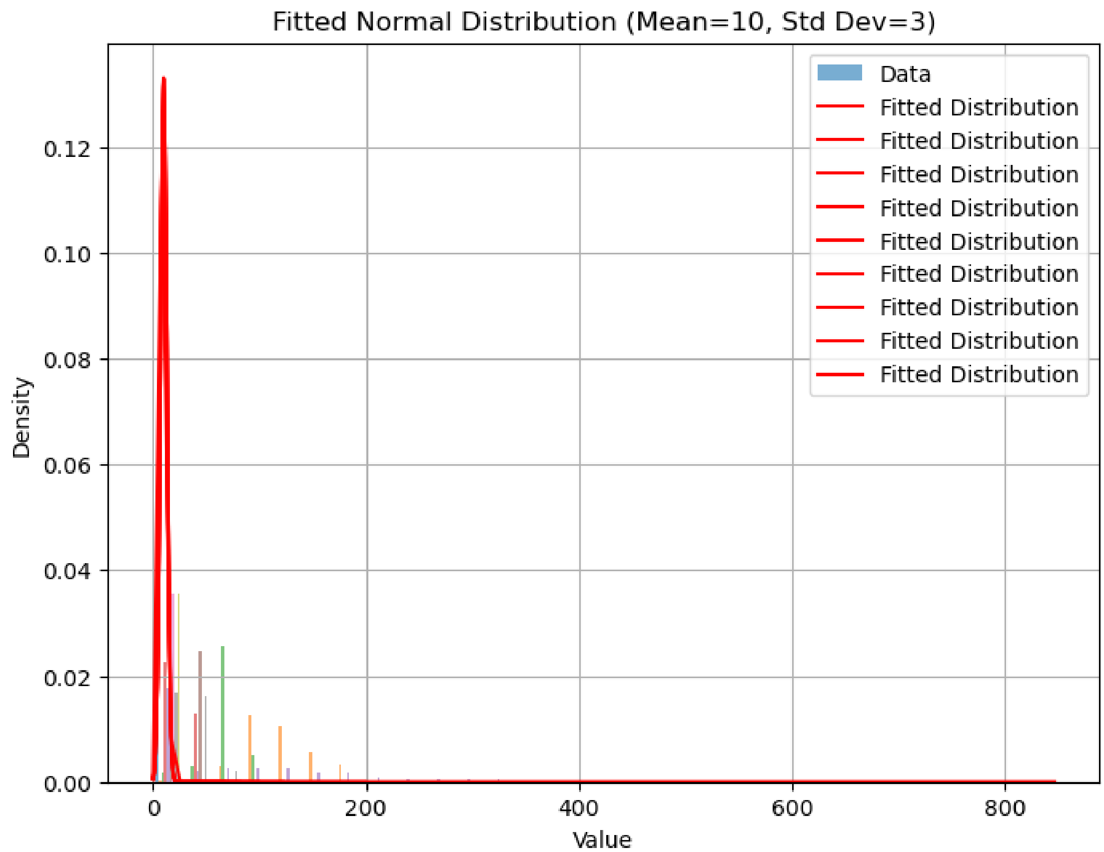
```
Hypothesis Testing:
T-Statistic: -50.6209, p-value: 0.0000
```

PARAMETRIC DATA EXPLORATION



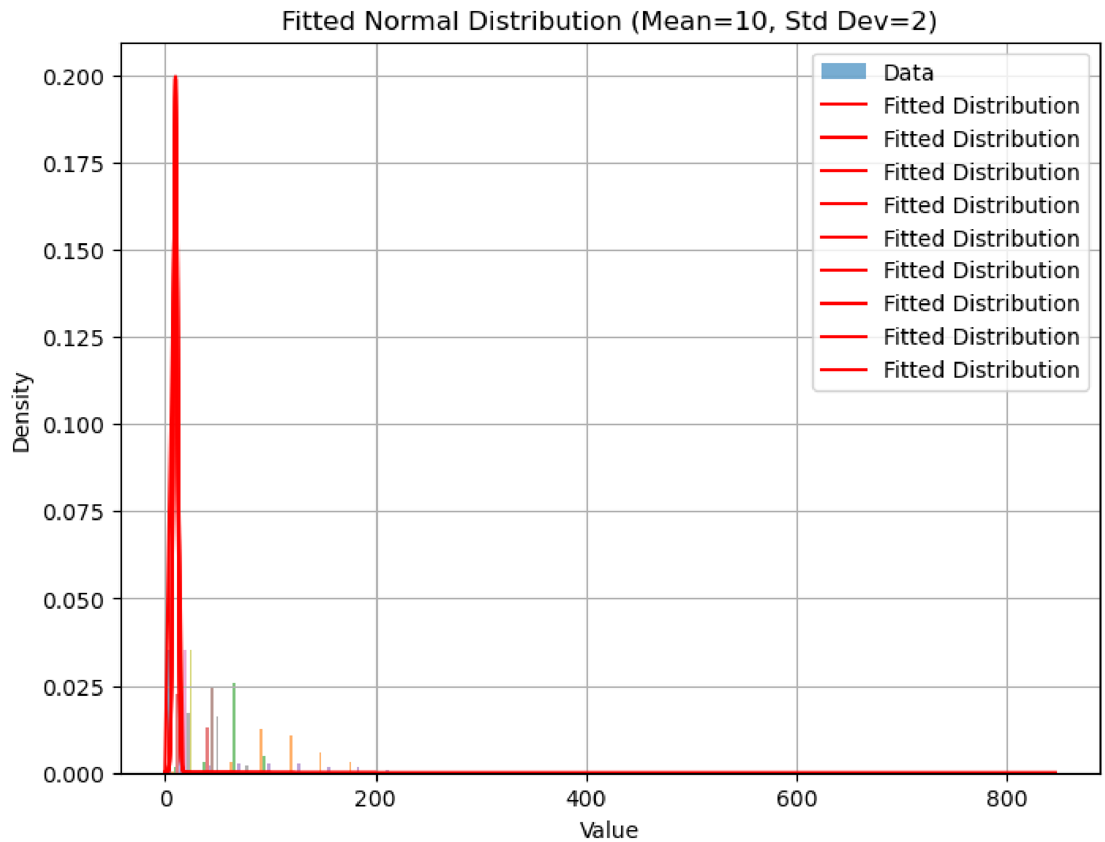

```
Parameters: Mean=9, Std Dev=2
Descriptive Statistics:
Mean: 9.00
Standard Deviation: 2.00
```

PARAMETRIC DATA EXPLORATION



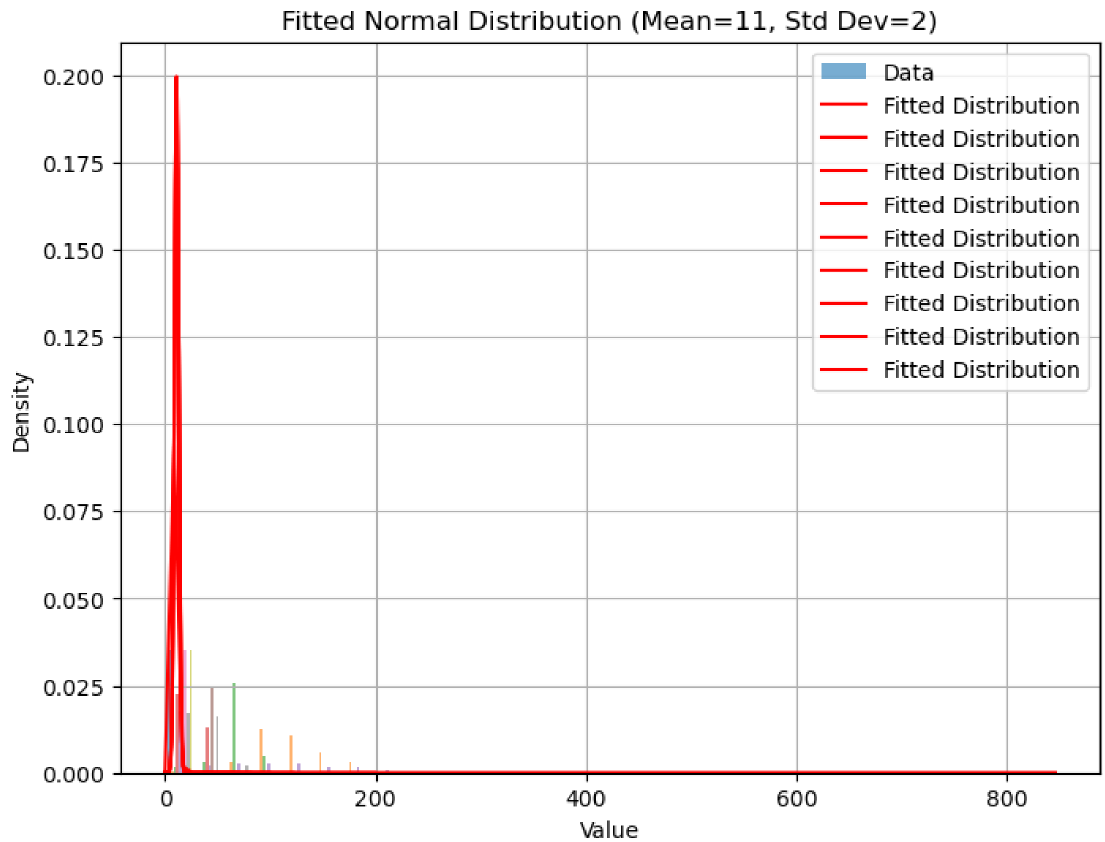
Parameters: Mean=10, Std Dev=3
Descriptive Statistics:
Mean: 10.00
Standard Deviation: 3.00

PARAMETRIC DATA EXPLORATION



Parameters: Mean=10, Std Dev=2
Descriptive Statistics:
Mean: 10.00
Standard Deviation: 2.00

PARAMETRIC DATA EXPLORATION



Parameters: Mean=11, Std Dev=2
Descriptive Statistics:
Mean: 11.00
Standard Deviation: 2.00

Conclusion: The p-value of 0 for the Outcome feature rejects the null hypothesis, indicating a non-normal distribution. Even after altering parameters (9, 2), (10, 3), (10, 2), and (11, 2), the rejection persists at a 5% significance level. This robust rejection across varied parameterizations underscores the deviation from normality in the Outcome distribution, suggesting non-normal characteristics that persist regardless of adjustments made to distribution parameters.