1. Introduction

This report details the process of anomaly detection in the provided dataset using the Isolation Forest algorithm. The dataset, which doesn't have a specific name and contains 61 features (x1 to x60, plus an unnamed column) and a binary response variable (y), has been analyzed for anomalies. The primary objective was to identify unusual data points that deviate significantly from the majority of the data.

2. Data Exploration and Preprocessing

Initial exploration of the data involved examining the first few rows and the data types of each column. The dataset was checked for missing values, but none were found. Subsequently, a descriptive analysis was performed, providing statistical summaries like mean, standard deviation, minimum, and maximum for each column. Additionally, the number of unique values for each column was determined.

	y	x1	x2	x3	x4	x5
count	18398	18398	18398	18398	18398	18398
mean	0.00673986	0.0118235	0.157986	0.5693	-9.95835	0.006
std	0.0818218	0.742875	4.93976	5.93718	131.034	0.634
min	0	-3.78728	-17.3165	-18.1985	-322.782	-1.623
25%	0	-0.405681	-2.15823	-3.53705	-111.378	-0.446
50%	0	0.128245	-0.0755048	-0.190683	-14.8816	-0.120

Number of duplicate rows: 0

Descriptive Statistics:

	time	y	x1 \
count	18398	18398.000000	18398.000000
mean	1999-05-15 01:20:42.728557312	0.006740	0.011824
min	1999-05-01 00:00:00	0.000000	-3.787279
25%	1999-05-08 03:36:30	0.000000	-0.405681
50%	1999-05-14 18:39:00	0.000000	0.128245
75%	1999-05-22 06:01:30	0.000000	0.421222
max	1999-05-29 00:06:00	1.000000	3.054156

std	NaN	0.081822	0.742875

	x2	x3	x4	x5	x6 \
count	18398.000000	18398.000000	18398.000000	18398.000000	18398.000000
mean	0.157986	0.569300	-9.958345	0.006518	2.387533
min	-17.316550	-18.198509	-322.781610	-1.623988	-279.408440
25%	-2.158235	-3.537054	-111.378372	-0.446787	-24.345268
50%	-0.075505	-0.190683	-14.881585	-0.120745	10.528435
75%	2.319297	3.421223	92.199134	0.325152	32.172974
max	16.742105	15.900116	334.694098	4.239385	96.060768
std	4.939762	5.937178	131.033712	0.634054	37.104012

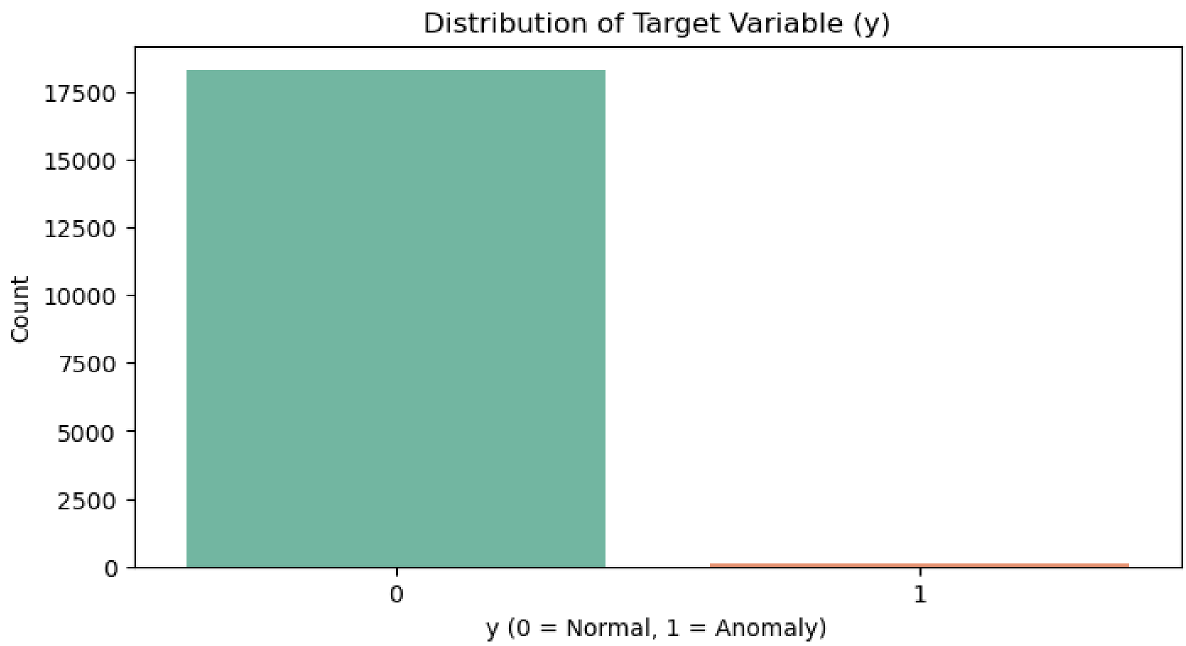
	x7	x8	...	x51	x52 \
count	18398.000000	18398.000000	...	18398.000000	18398.000000
mean	0.001647	-0.004125	...	-3.357339	0.380519
min	-0.429273	-0.451141	...	-3652.989000	-187.943440
25%	-0.058520	-0.051043	...	29.984624	-3.672684
50%	-0.009338	-0.000993	...	29.984624	0.294846
75%	0.060515	0.038986	...	29.984624	5.109543
max	1.705590	0.788826	...	40.152348	14.180588
std	0.108870	0.075460	...	348.256716	6.211598

	x54	x55	x56	x57	x58 \
count	18398.000000	18398.000000	18398.000000	18398.000000	18398.000000
mean	0.173708	2.379154	9.234953	0.233493	-0.001861
min	-8.210370	-230.574030	-269.039500	-12.640370	-0.149790
25%	0.487780	-40.050046	-45.519149	-1.598804	0.000470
50%	0.702299	17.471317	1.438806	0.085826	0.012888
75%	2.675751	44.093387	63.209681	2.222118	0.020991

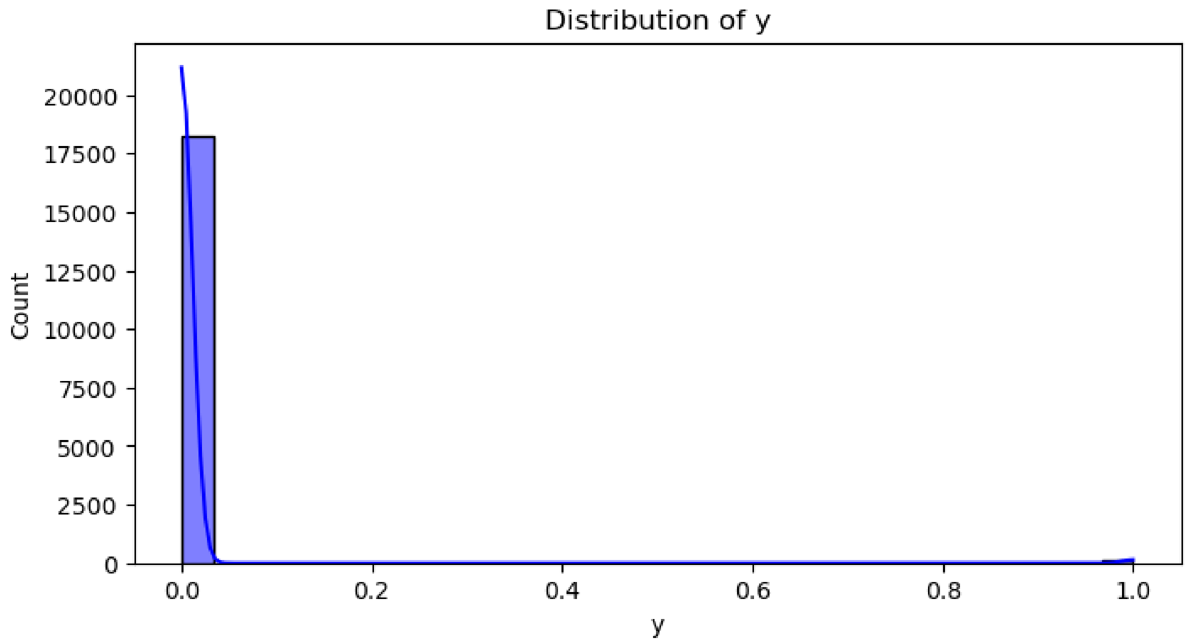
max	6.637265	287.252017	252.147455	6.922008	0.067249
std	3.029516	67.940694	81.274103	2.326838	0.048732

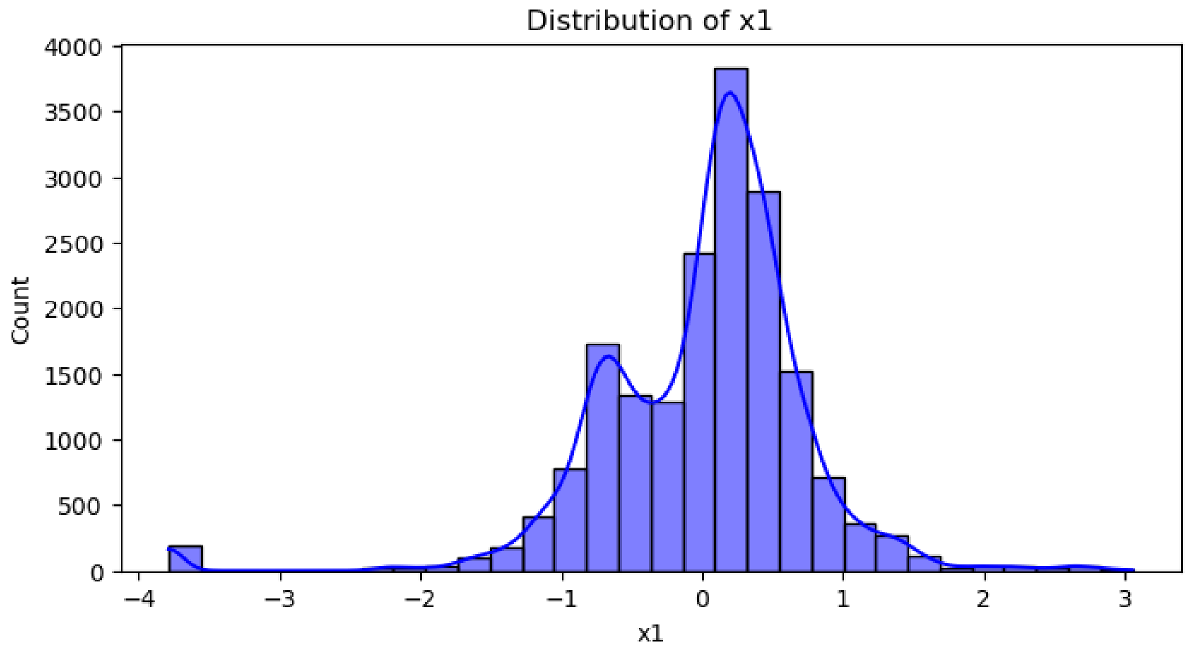
	x59	x60	y.1
count	18398.000000	18398.000000	18398.000000
mean	-0.061522	0.001258	0.001033
min	-100.810500	-0.012229	0.000000
25%	0.295023	-0.001805	0.000000
50%	0.734591	0.000710	0.000000
75%	1.266506	0.004087	0.000000
max	6.985460	0.020510	1.000000
std	10.394085	0.004721	0.032120

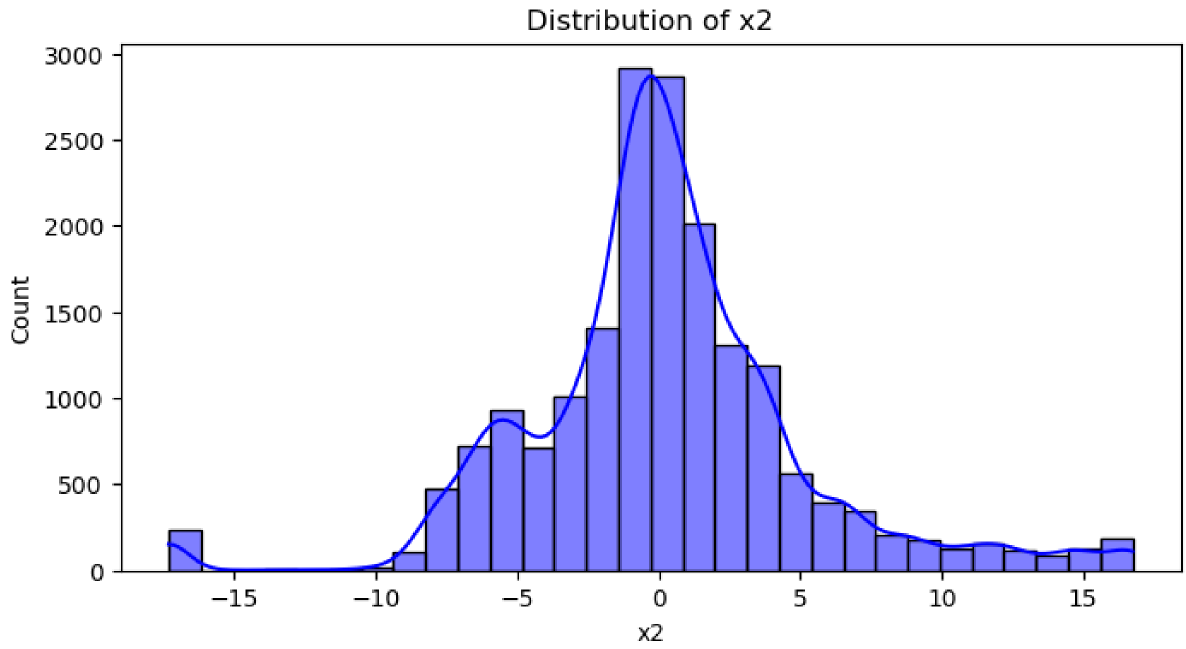
[8 rows x 62 columns]

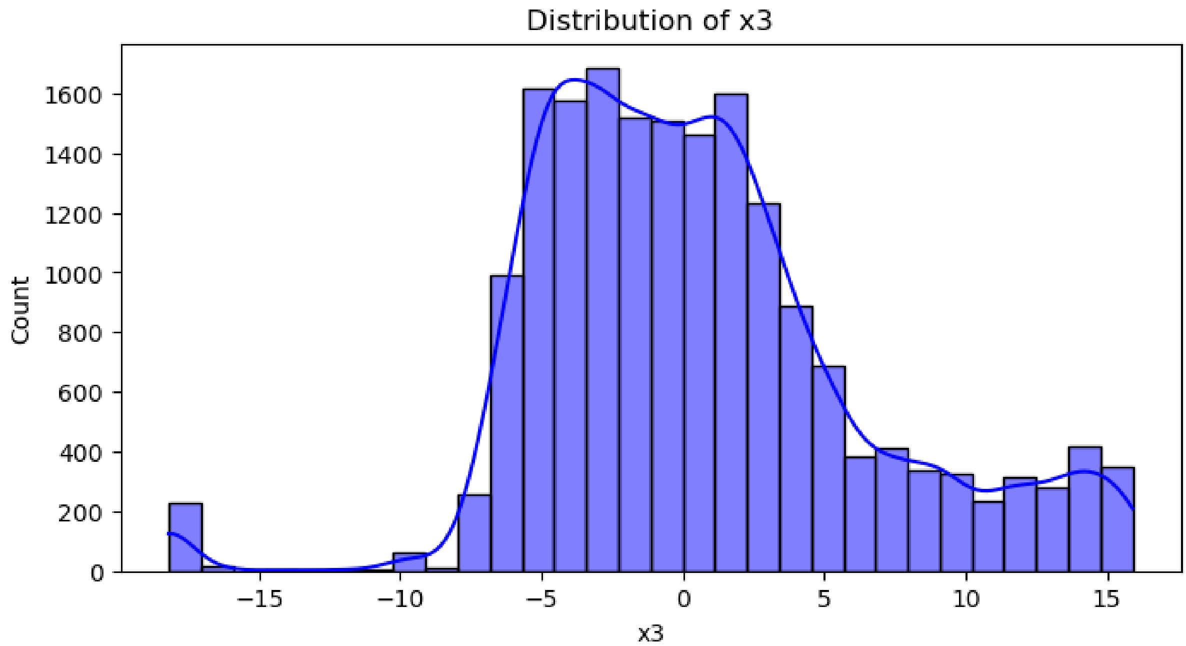


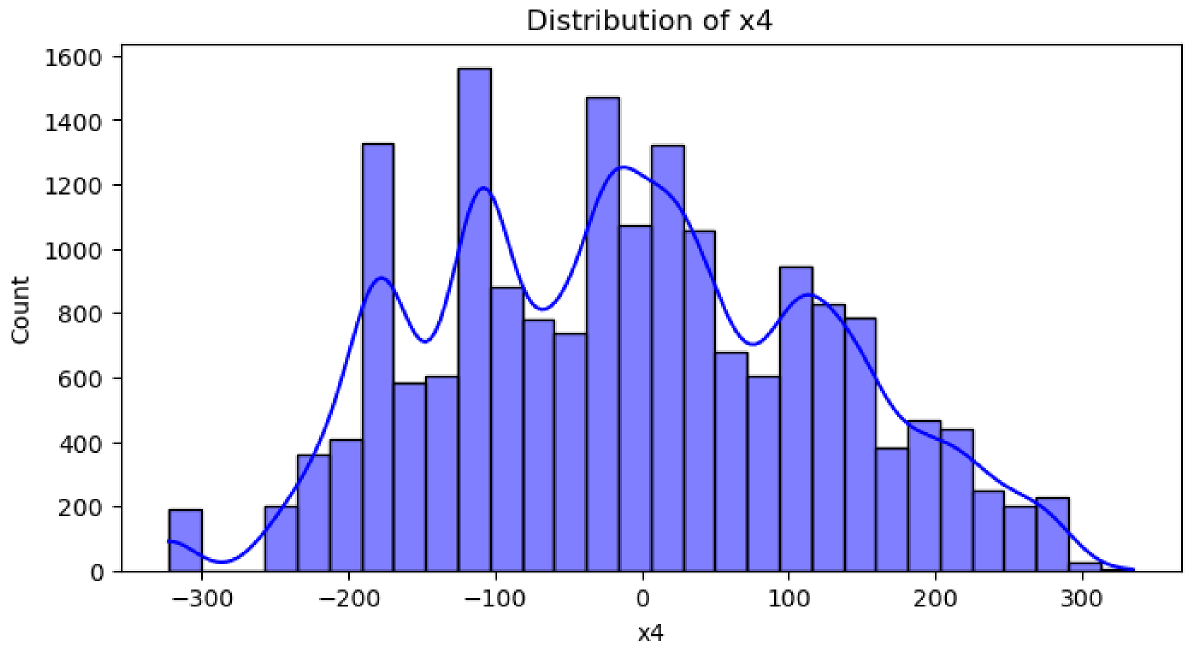
Generating histograms for numeric features:

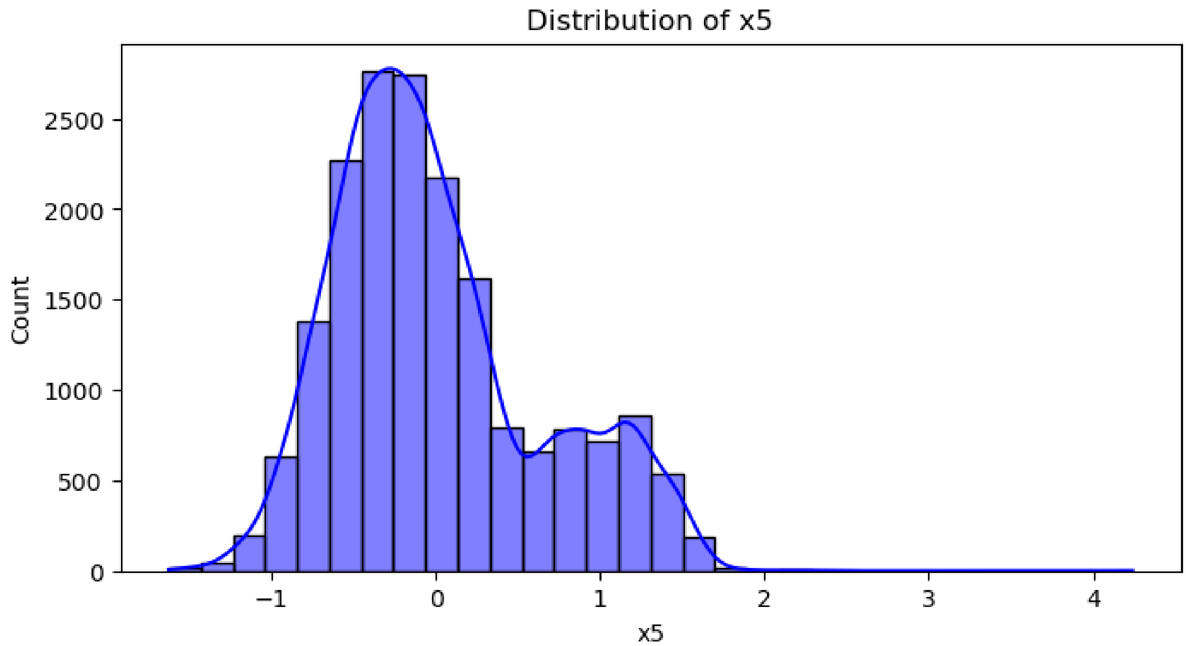

Distribution of y
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0.0
0.2
0.4
0.6
0.8
1.0
y

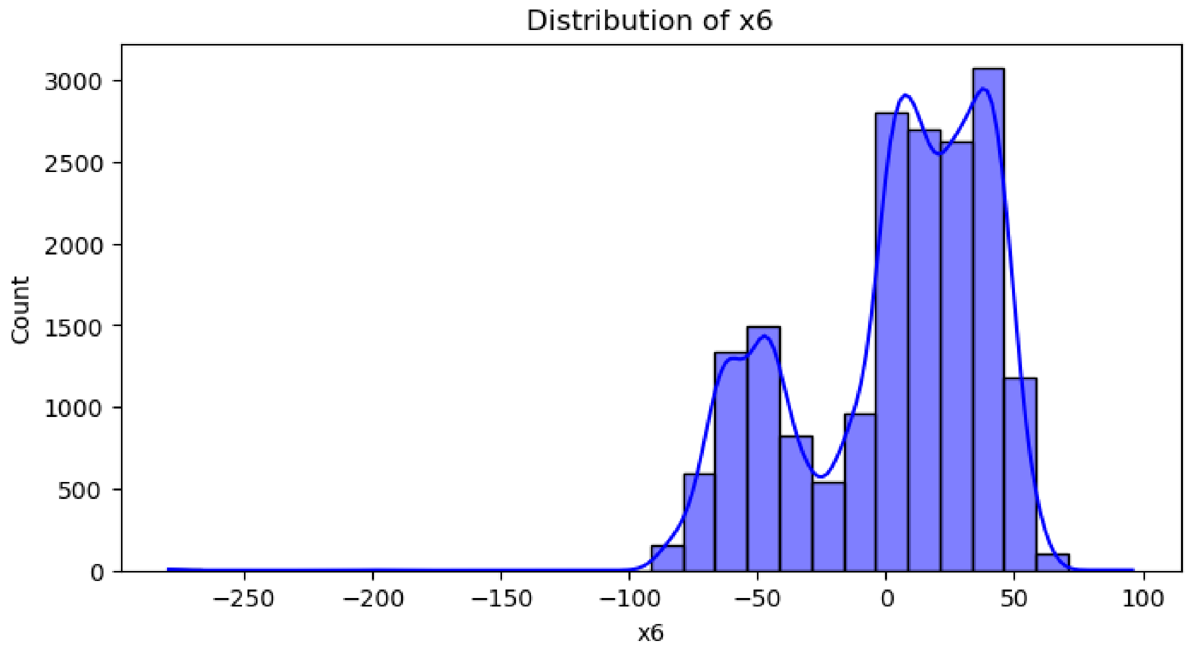

Distribution of x1
Count
4000
3500
3000
2500
2000
1500
1000
500
0
−4
−3
−2
−1
0
1
2
3
x1


Distribution of x2
Count
3000
2500
2000
1500
1000
500
0
−15
−10
−5
0
5
10
15
x2

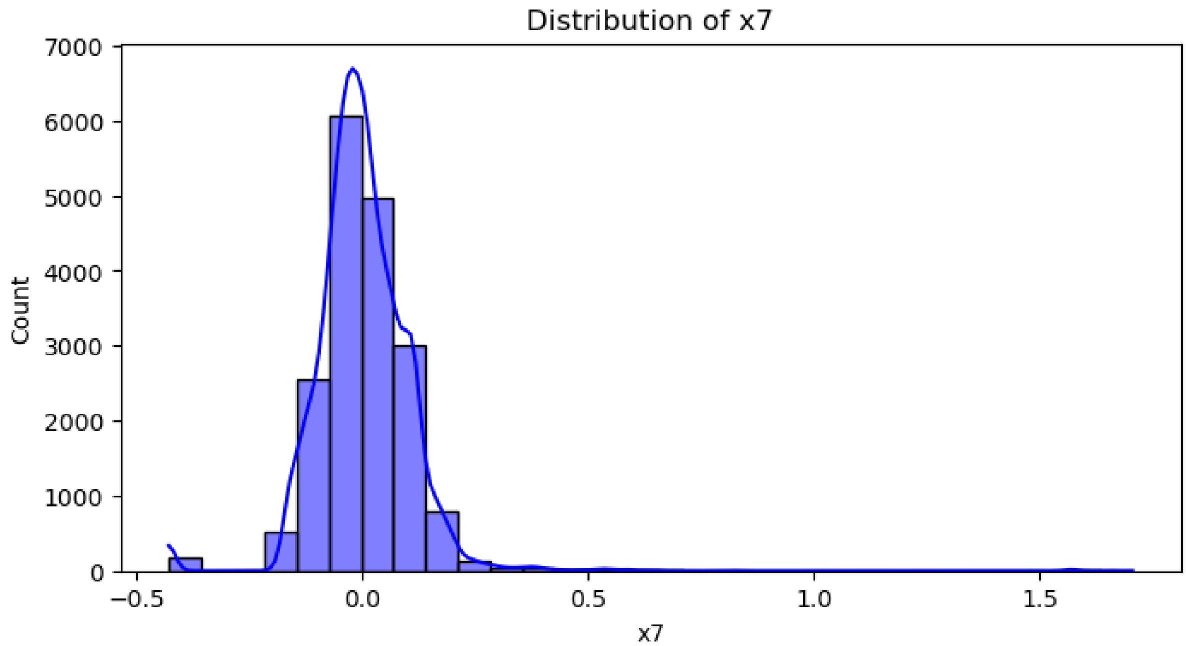

Distribution of x3
Count
1600
1400
1200
1000
800
600
400
200
0
−15
−10
−5
0
5
10
15
x3


Distribution of x4
Count
x4
1600
1400
1200
1000
800
600
400
200
0
−300
−200
−100
0
100
200
300

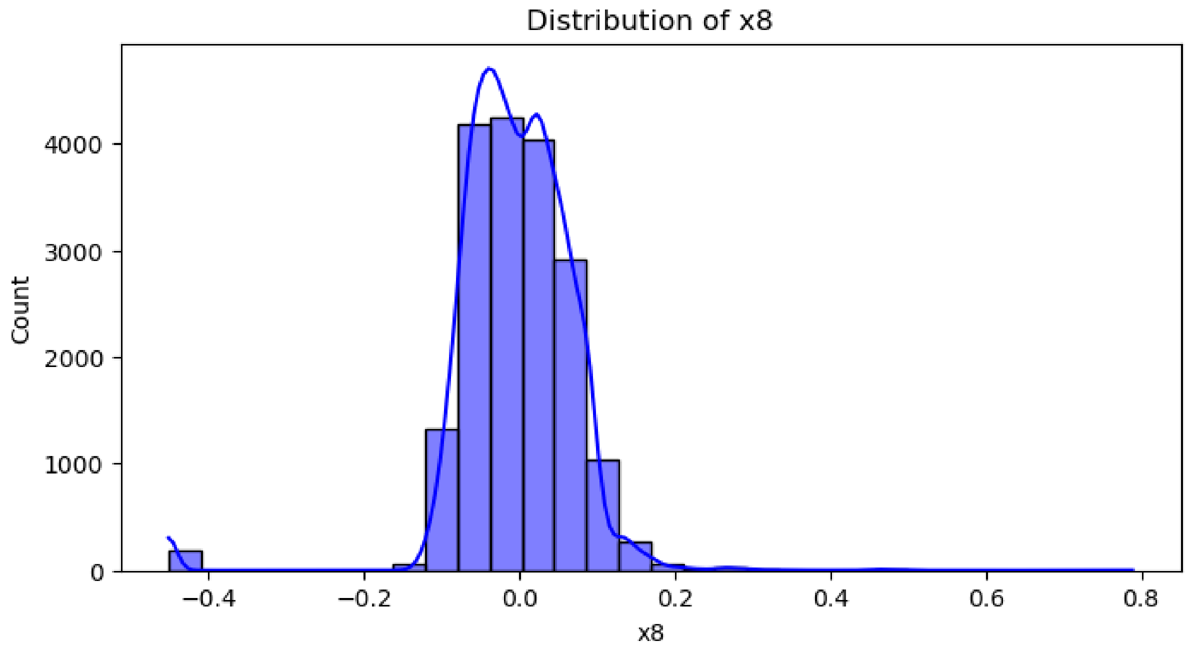

Distribution of x5
Count
x5
2500
2000
1500
1000
500
0
−1
0
1
2
3
4

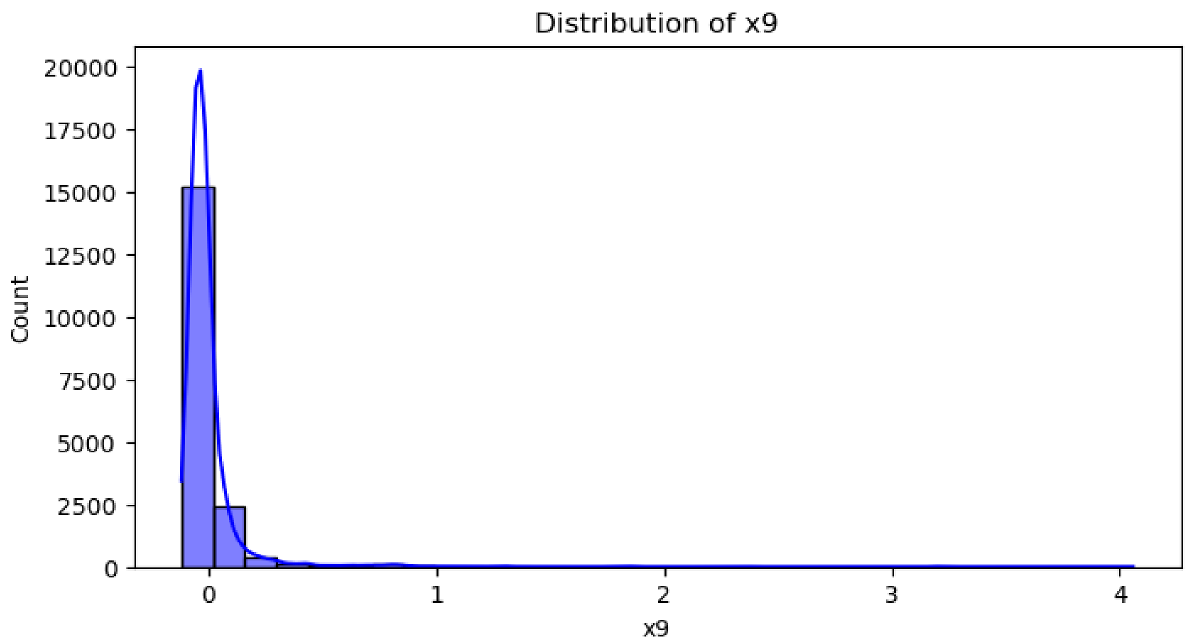


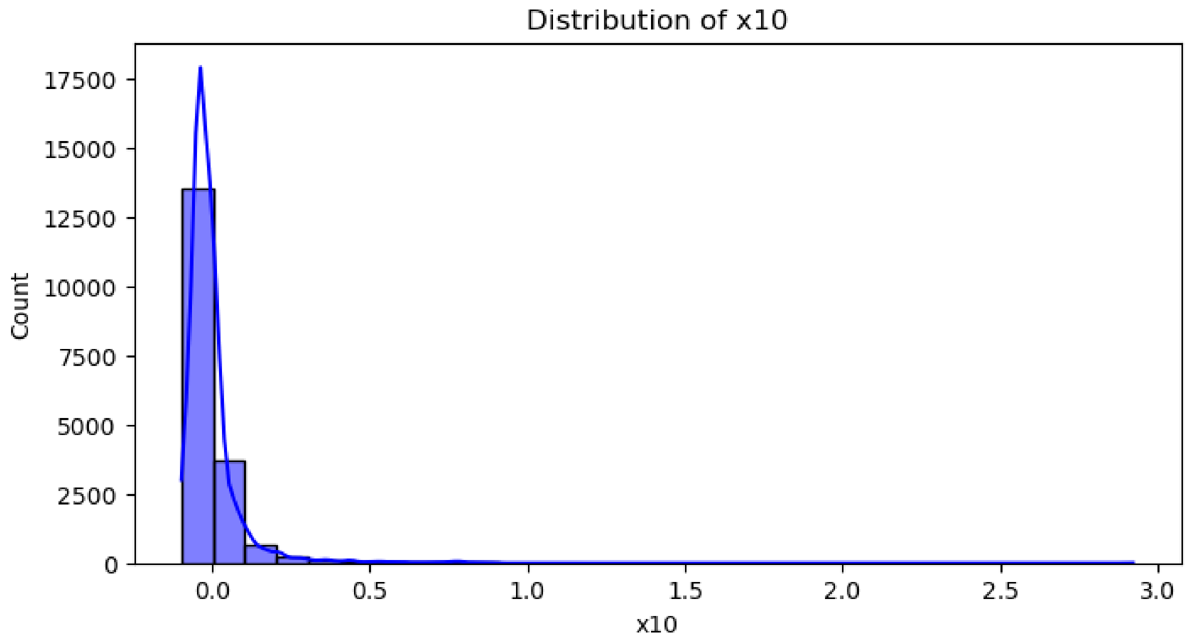
Distribution of x6
Count
3000
2500
2000
1500
1000
500
0
−250
−200
−150
−100
−50
0
50
100
x6

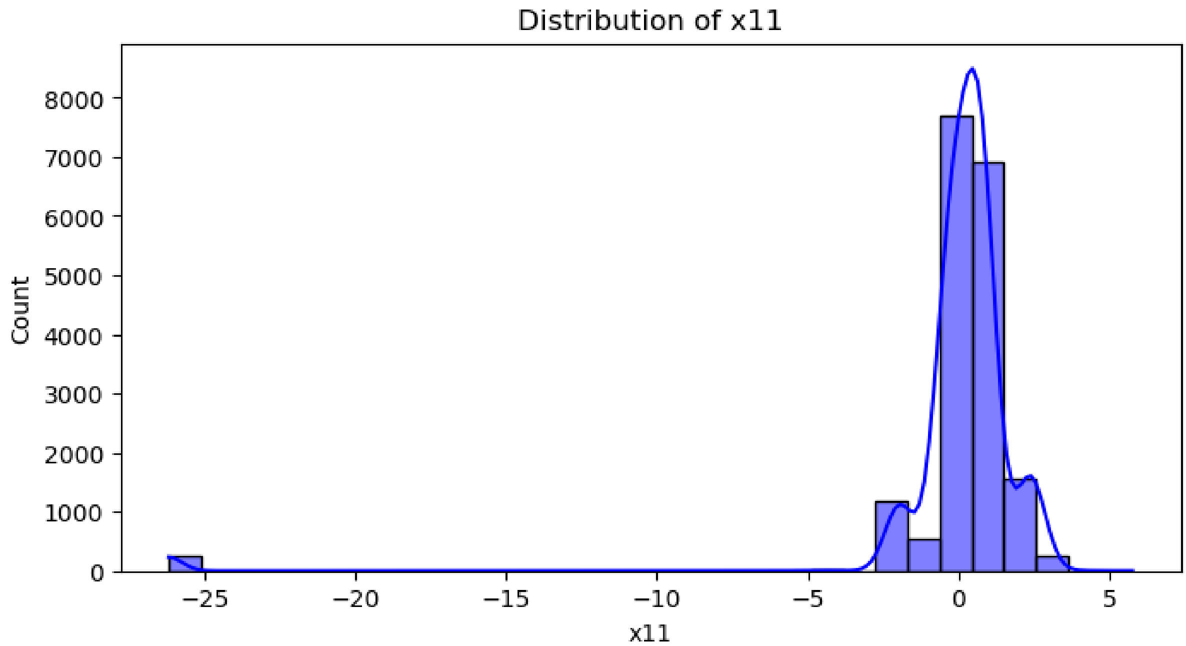


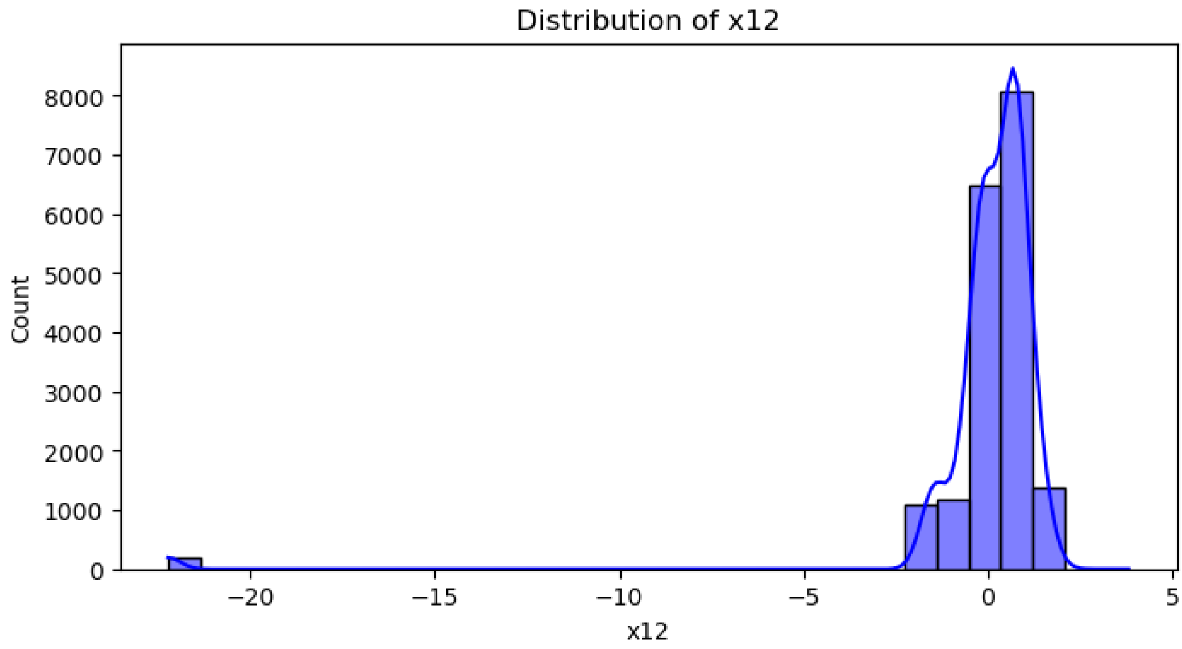
Distribution of x7
Count
7000
6000
5000
4000
3000
2000
1000
0
−0.5
0.0
0.5
1.0
1.5
x7


Distribution of x8
Count
4000
3000
2000
1000
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
x8

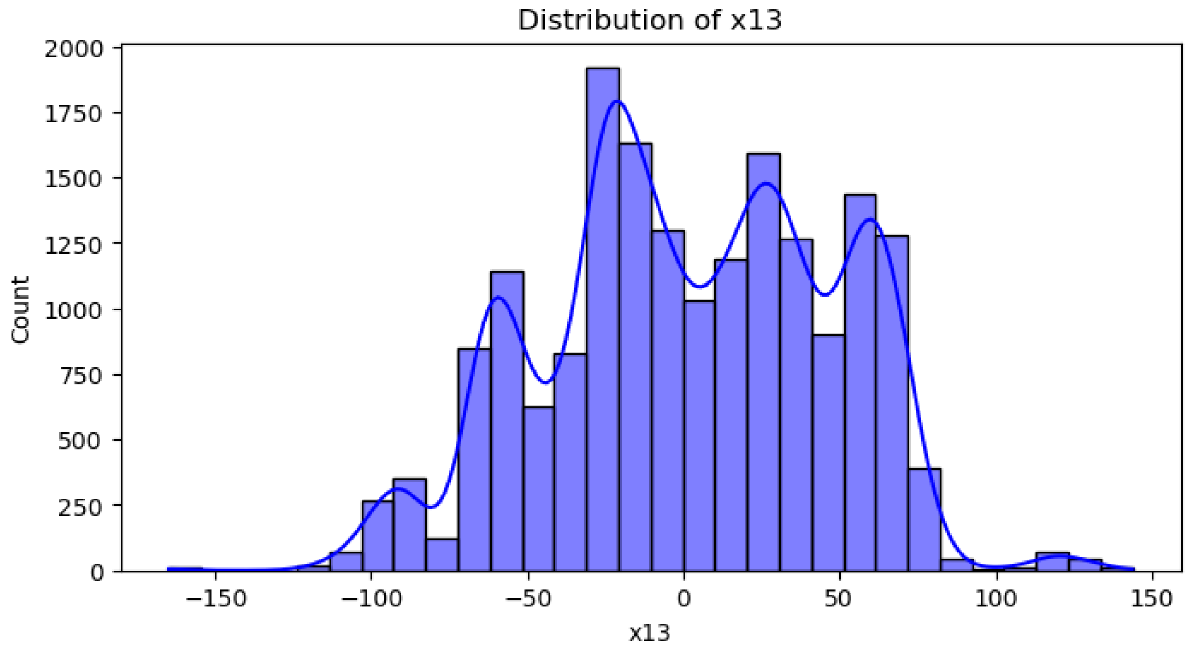

Distribution of x9
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
x9


Distribution of x10
17500
15000
12500
10000
7500
5000
2500
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x10

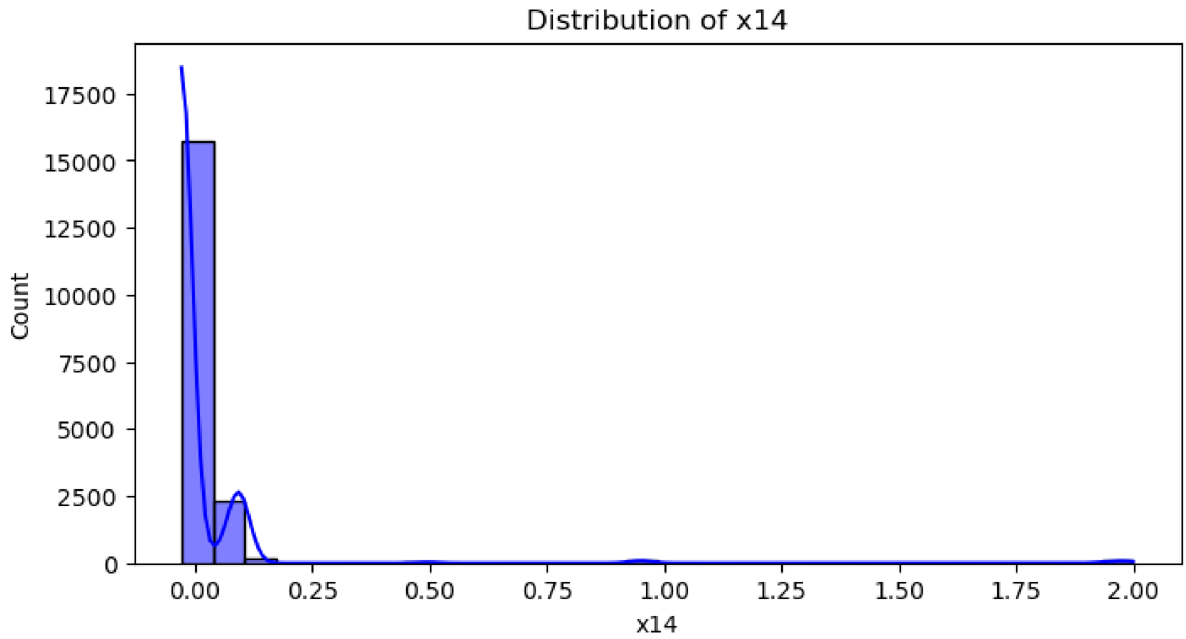

Distribution of x11
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
−25
−20
−15
−10
−5
0
5
x11

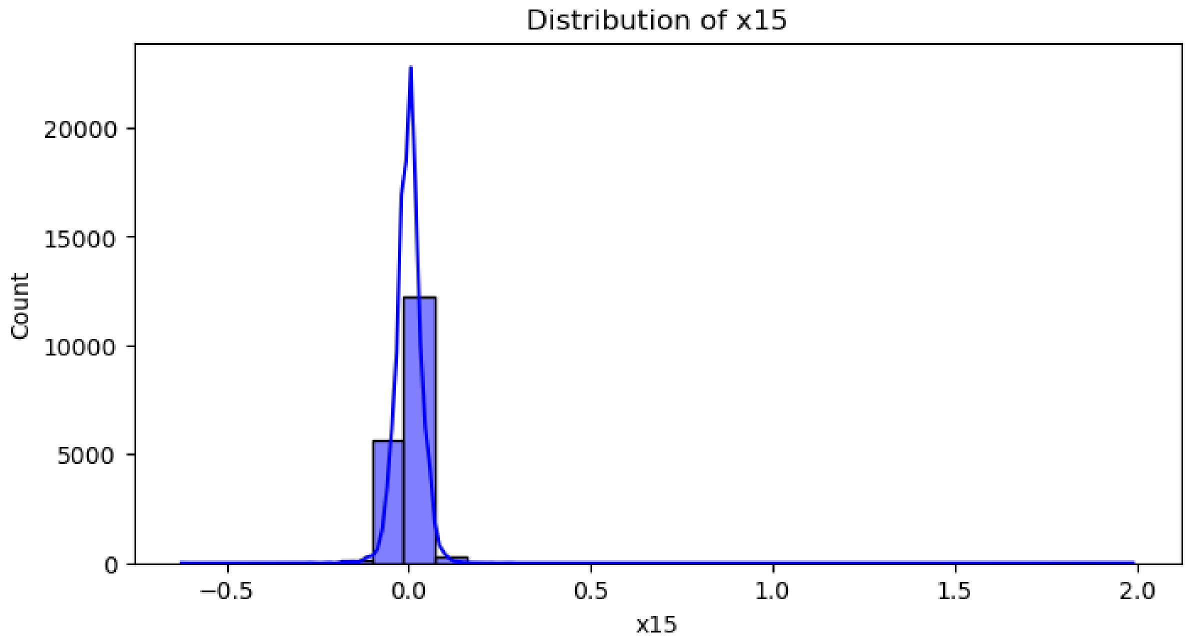


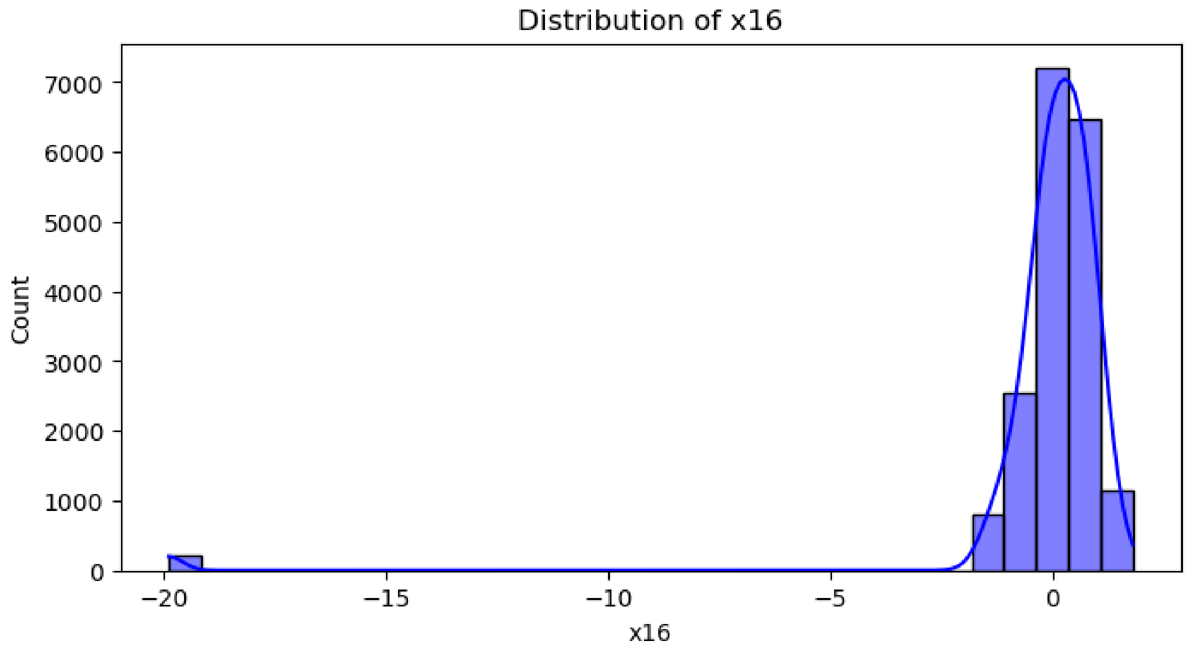
Distribution of x12
Count
x12

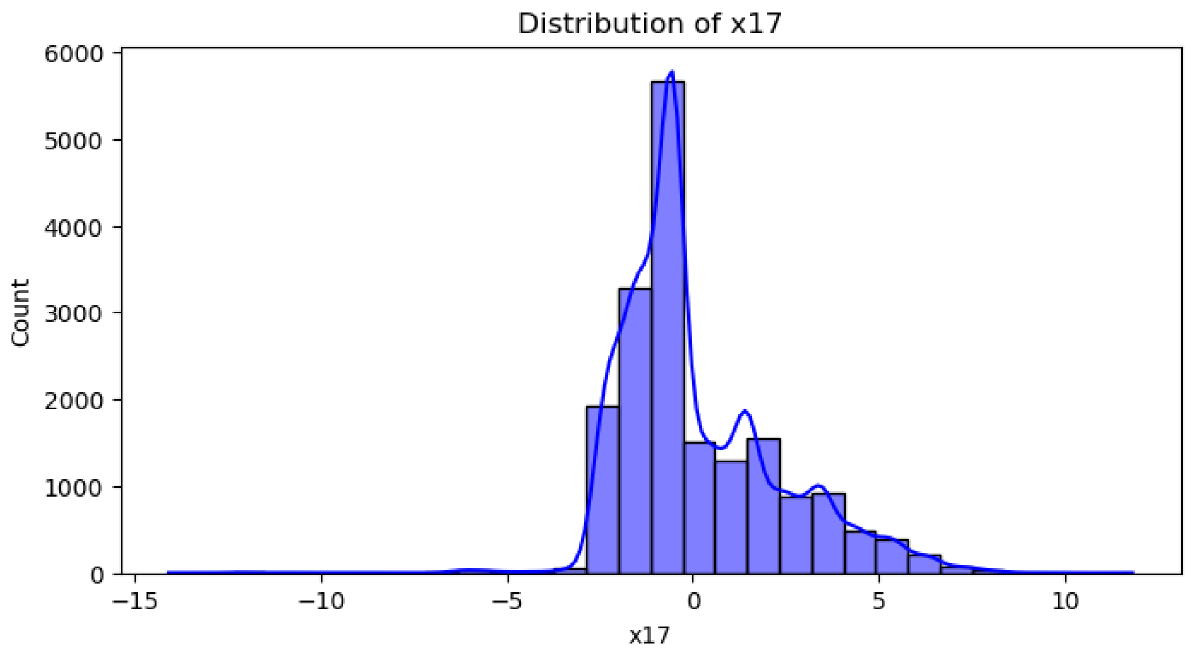


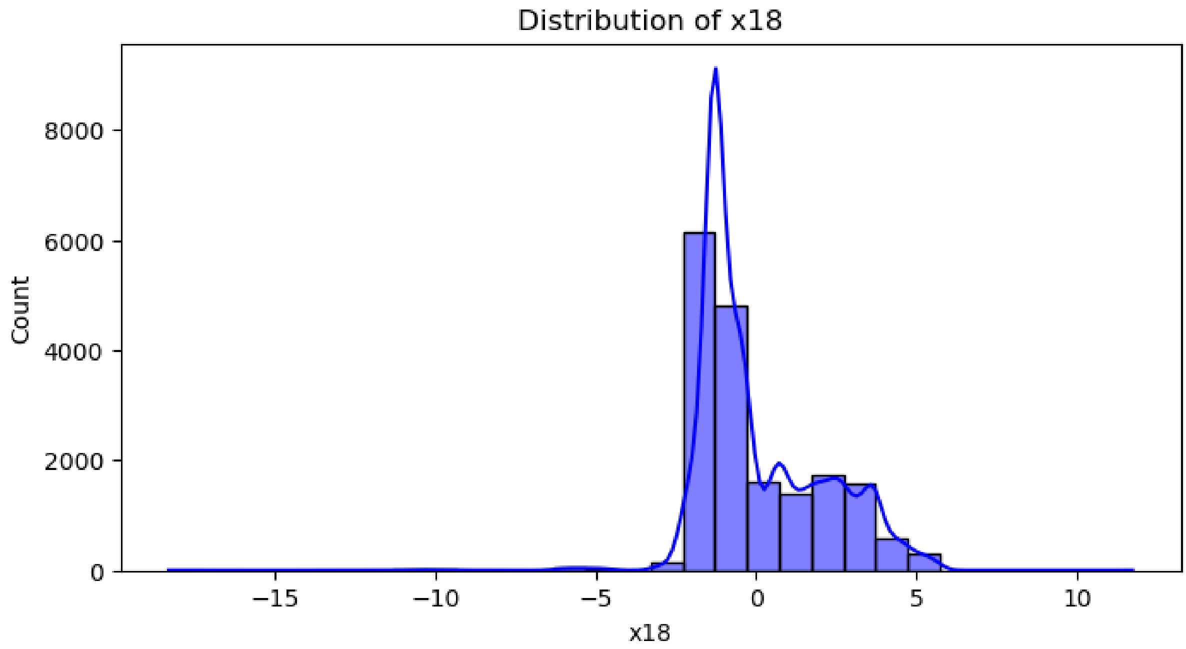
Distribution of x13
Count
x13


Distribution of x14
17500
15000
12500
10000
7500
5000
2500
0
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x14

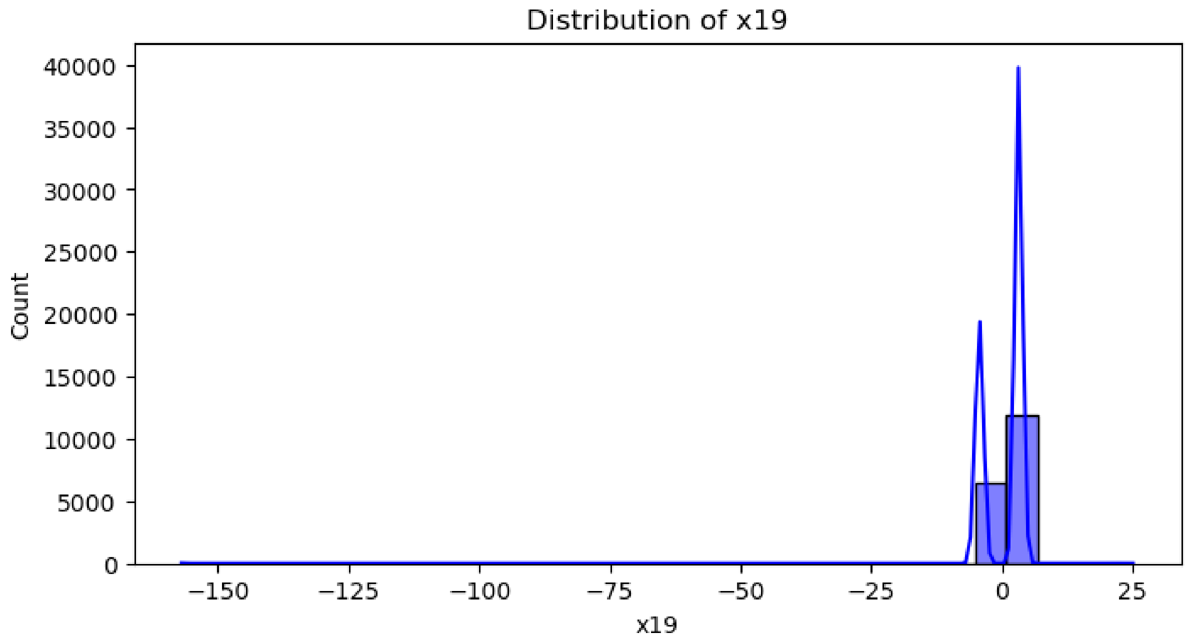

Distribution of x15
20000
15000
10000
5000
0
Count
−0.5
0.0
0.5
1.0
1.5
2.0
x15


Distribution of x16
Count
7000
6000
5000
4000
3000
2000
1000
0
−20
−15
−10
−5
0
x16

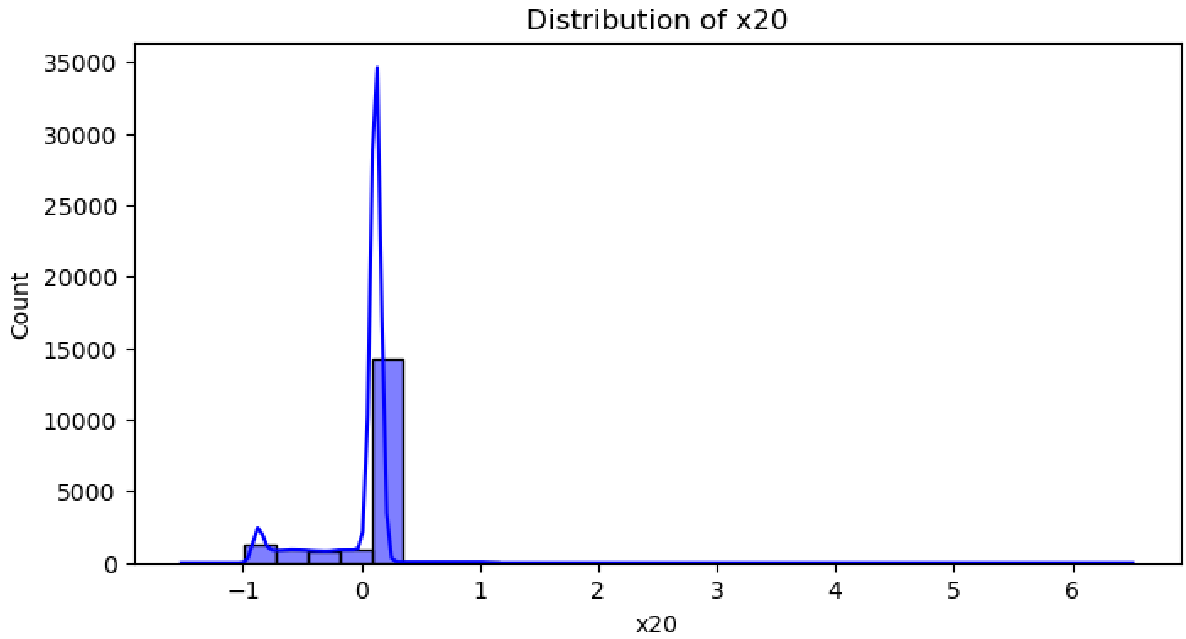

Distribution of x17
Count
6000
5000
4000
3000
2000
1000
0
−15
−10
−5
0
5
10
x17

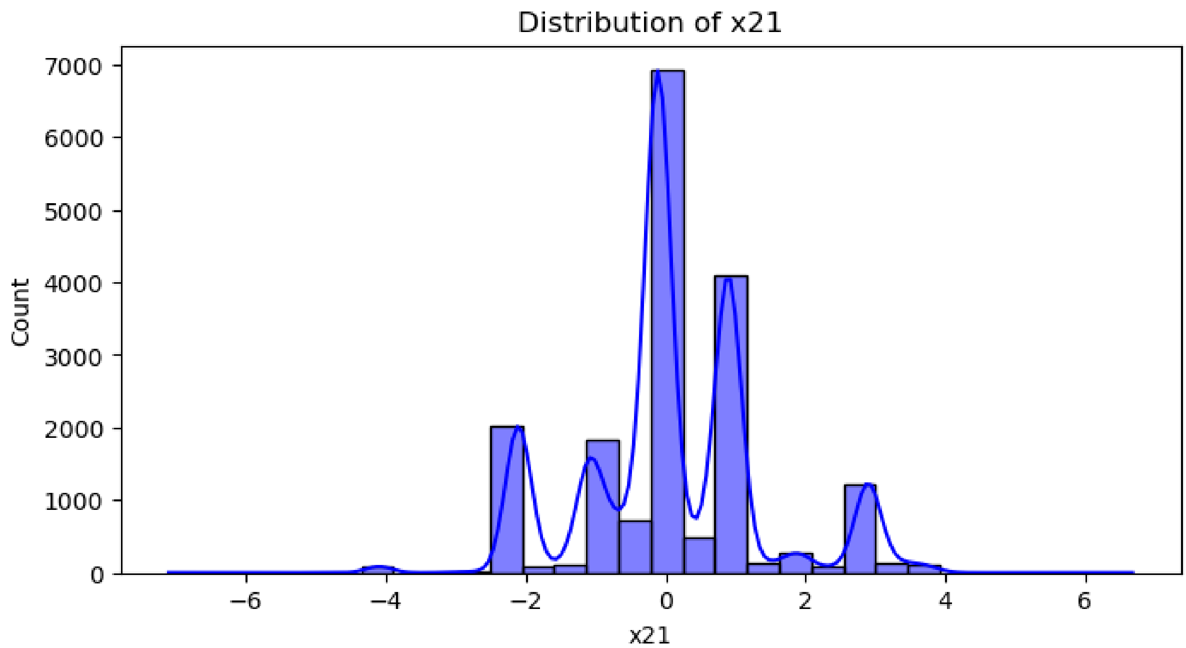


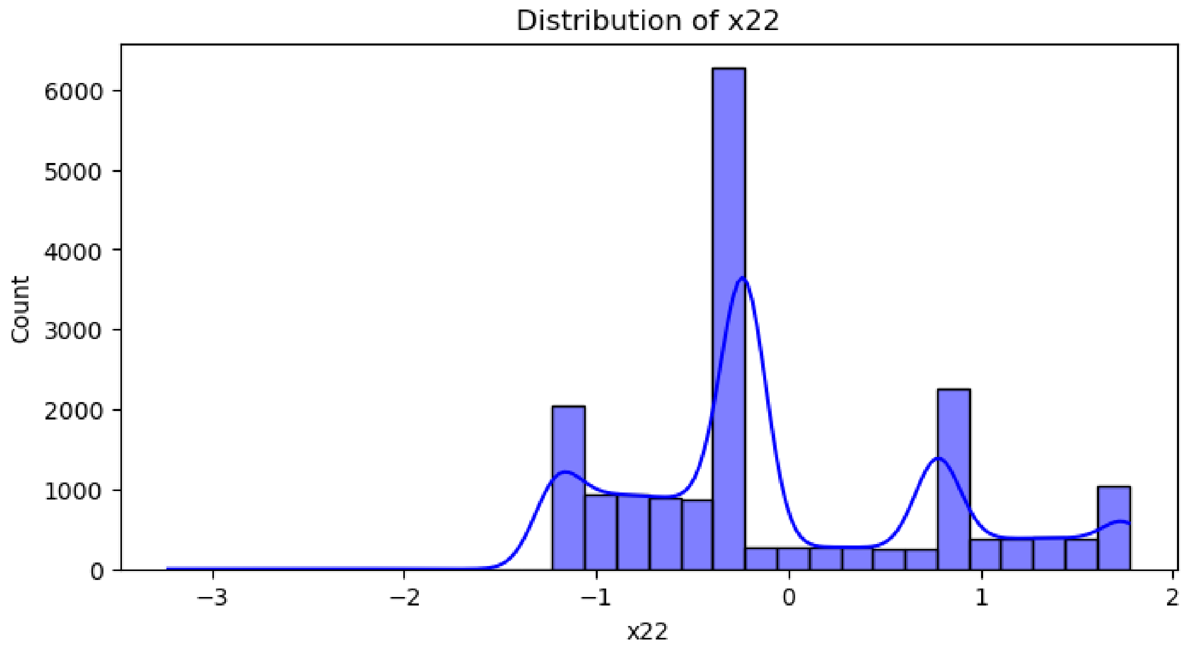
Distribution of x18
Count
8000
6000
4000
2000
0
−15
−10
−5
0
5
10
x18

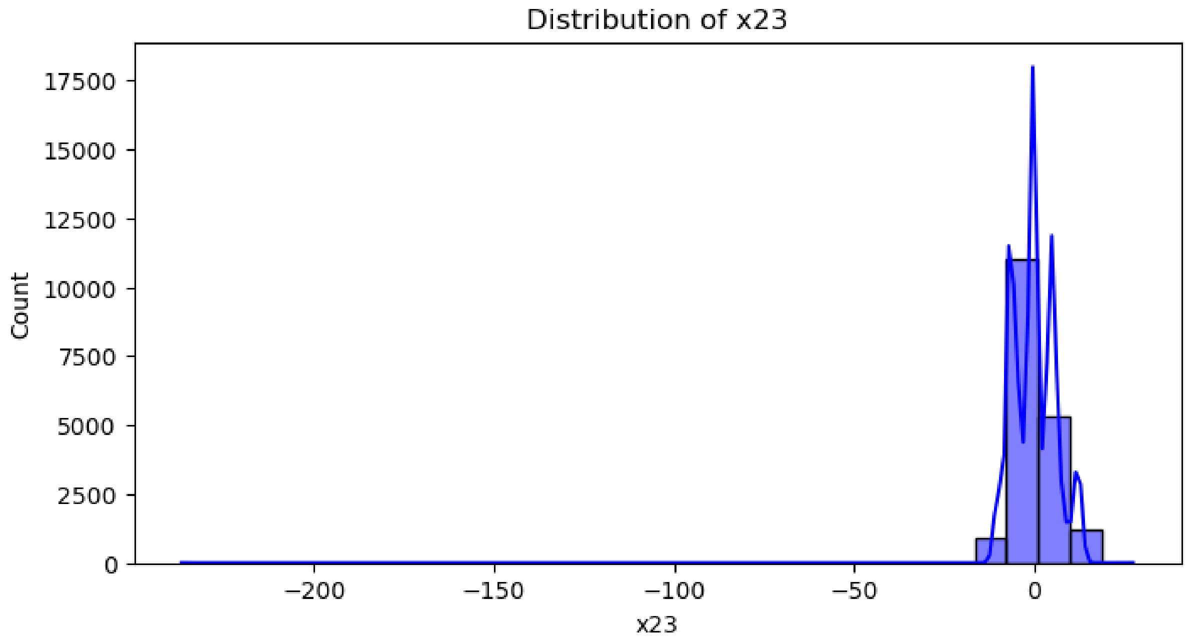


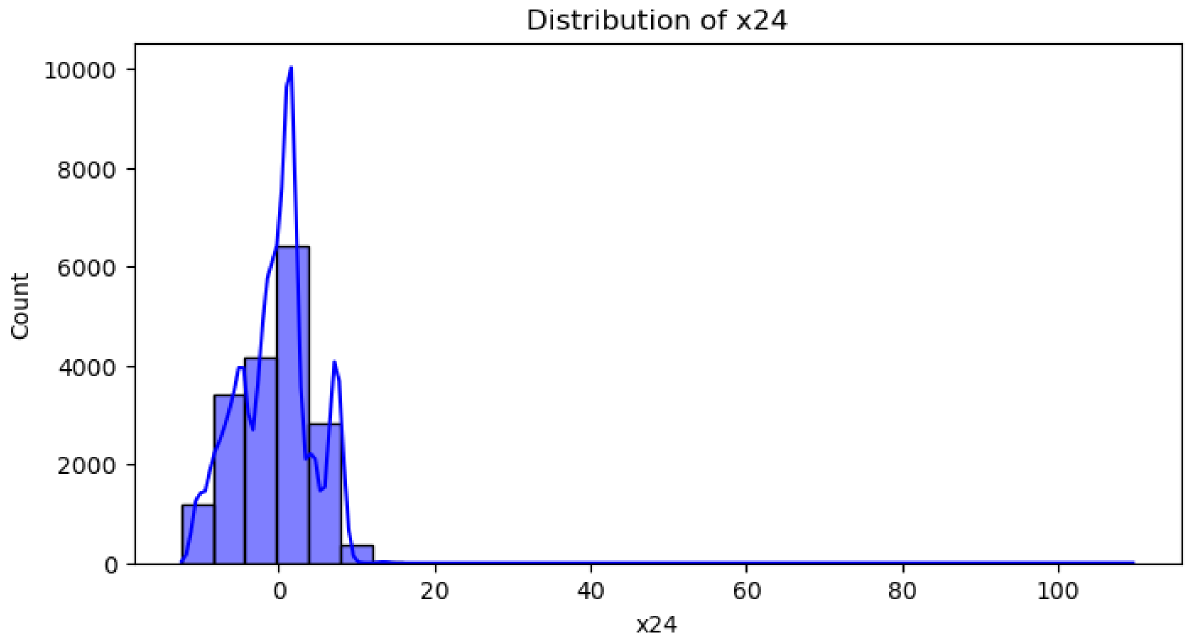
Distribution of x19
Count
40000
35000
30000
25000
20000
15000
10000
5000
0
−150
−125
−100
−75
−50
−25
0
25
x19

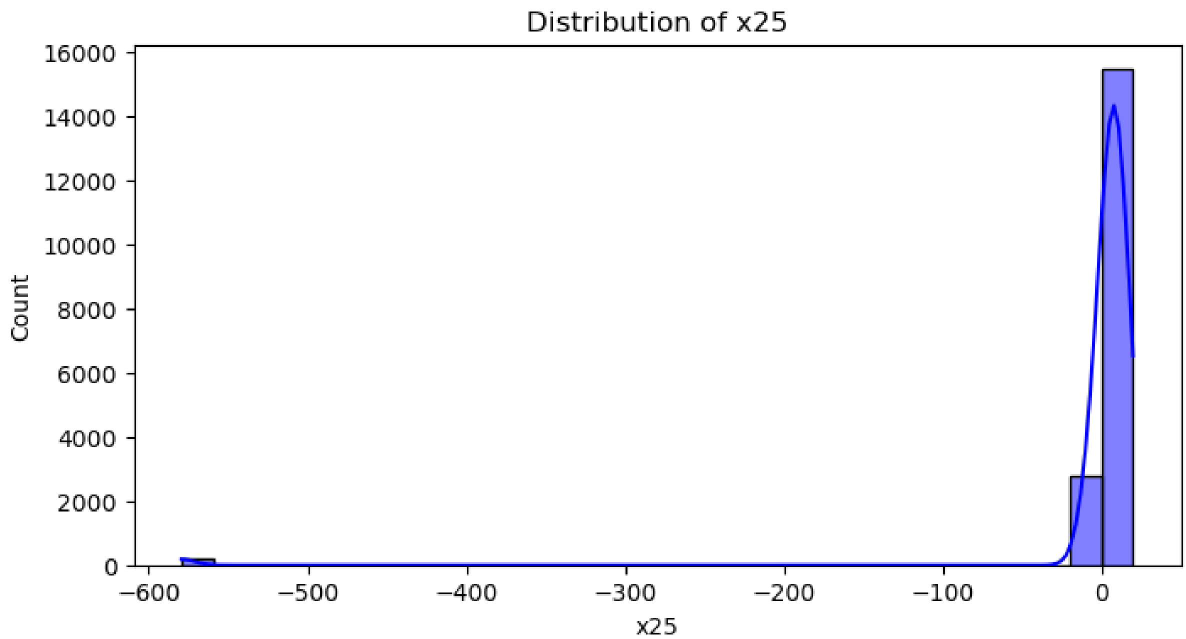

Distribution of x20
35000
30000
25000
20000
15000
10000
5000
0
Count
−1
0
1
2
3
4
5
6
x20

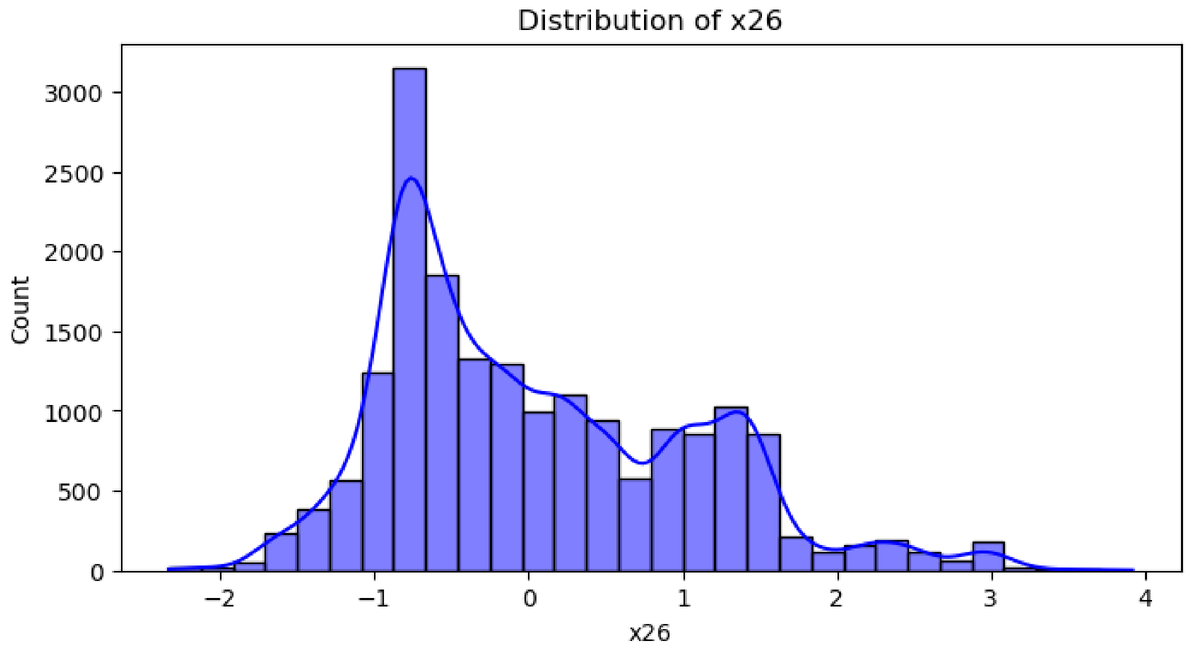

Distribution of x21
7000
6000
5000
4000
3000
2000
1000
0
Count
−6
−4
−2
0
2
4
6
x21

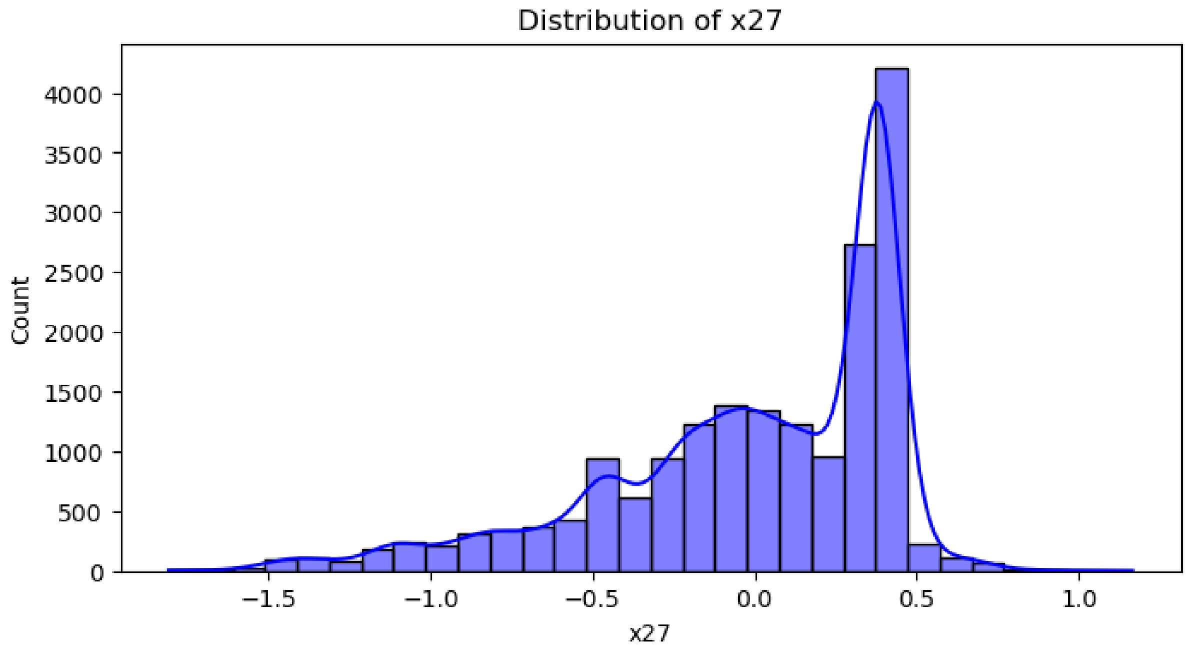

Distribution of x22
Count
x22
6000
5000
4000
3000
2000
1000
0
−3
−2
−1
0
1
2

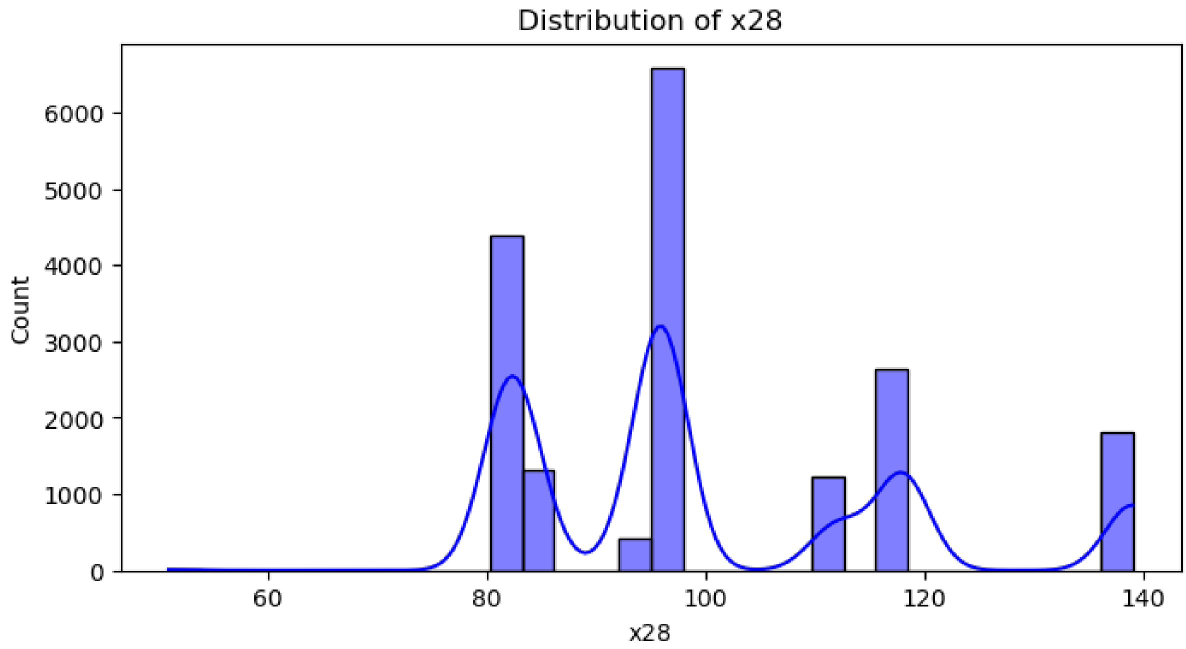

Distribution of x23
Count
x23
17500
15000
12500
10000
7500
5000
2500
0
−200
−150
−100
−50
0

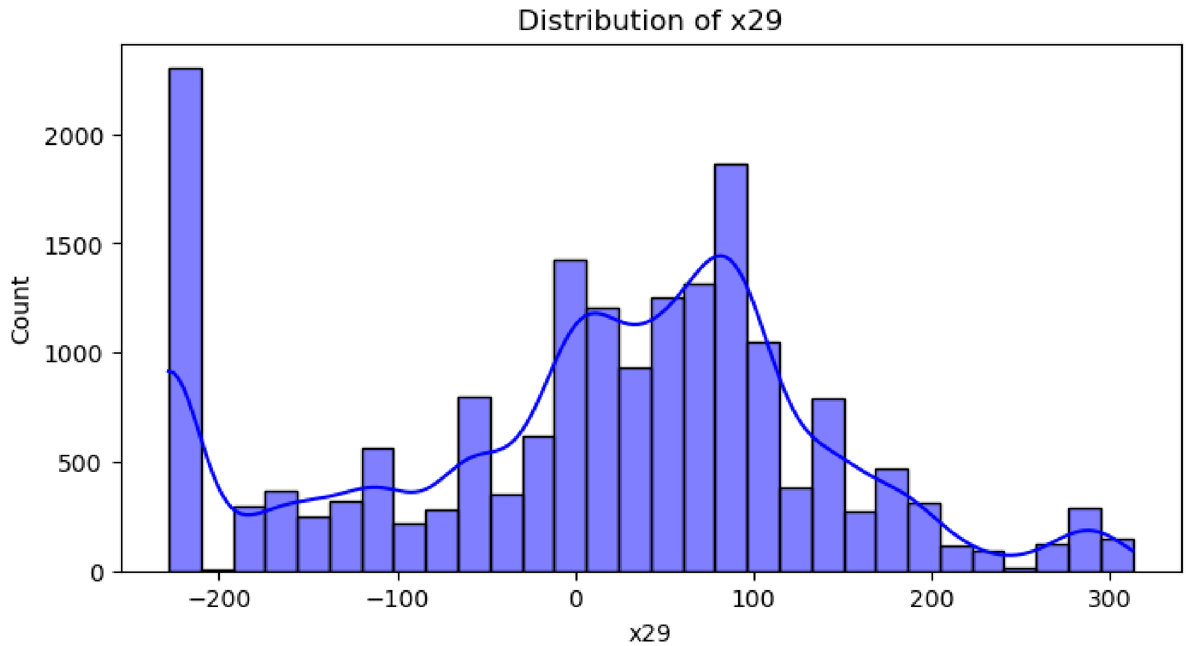

Distribution of x24
Count
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
x24

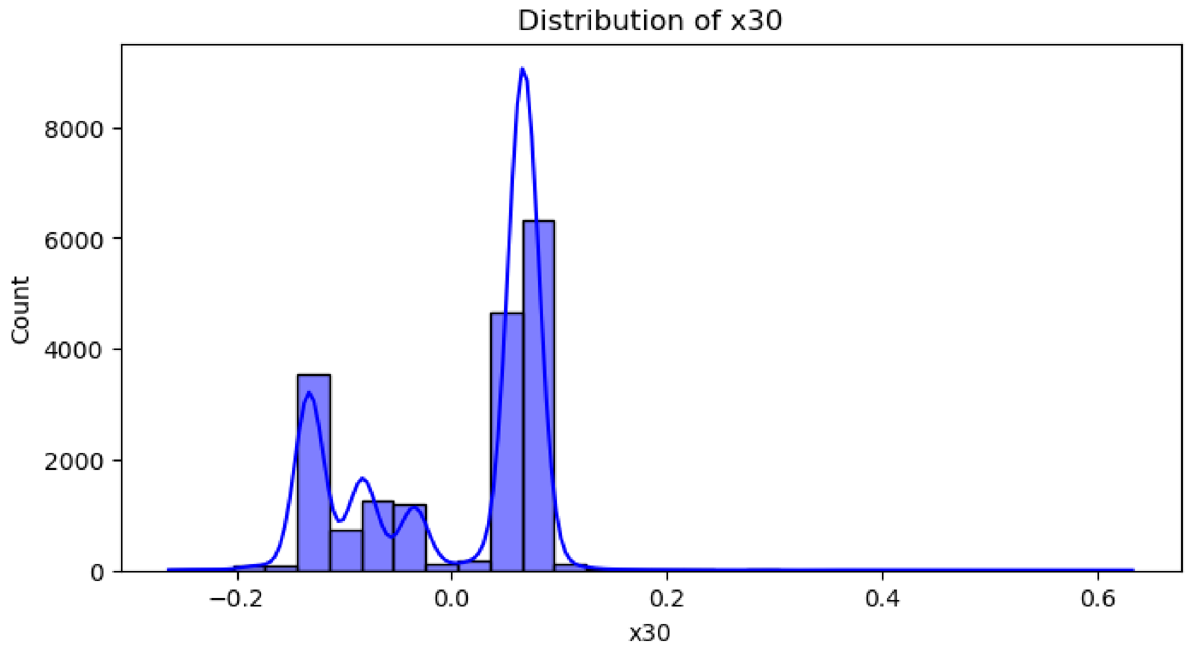

Distribution of x25
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
−600
−500
−400
−300
−200
−100
0
x25

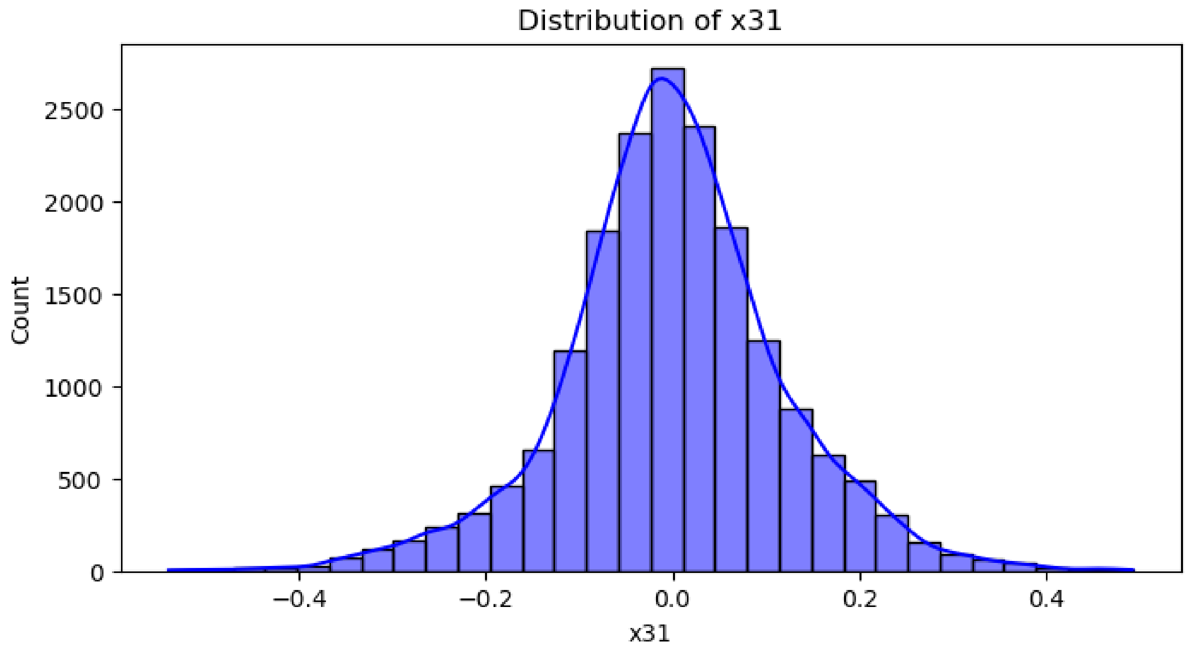

Distribution of x26
Count
3000
2500
2000
1500
1000
500
0
−2
−1
0
1
2
3
4
x26

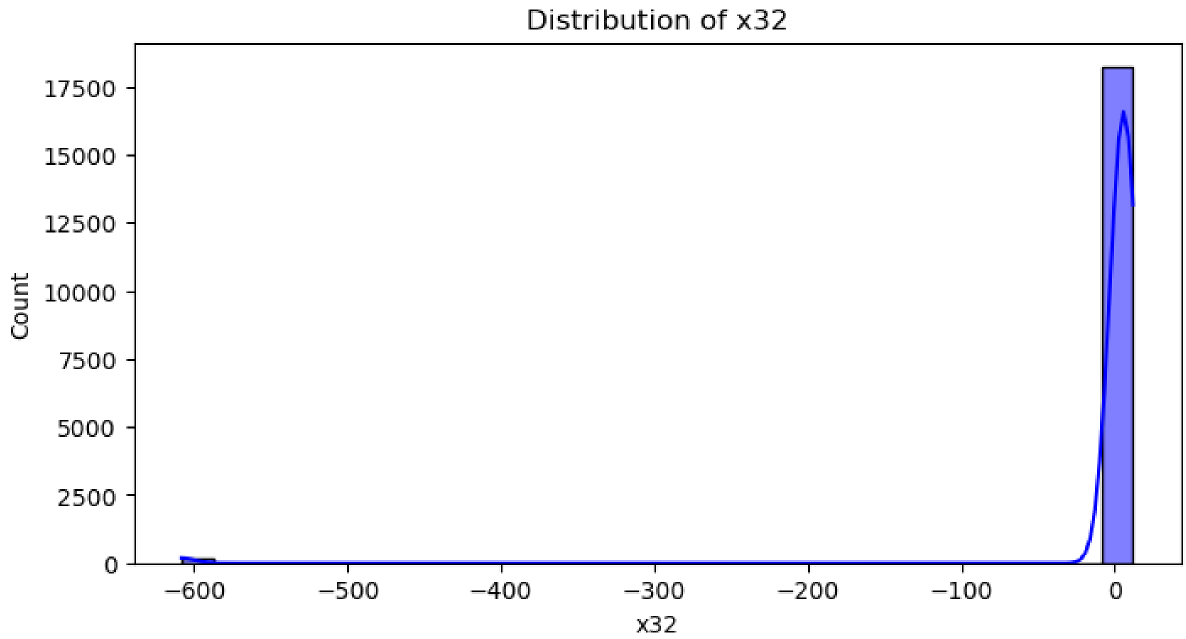

Distribution of x27
Count
4000
3500
3000
2500
2000
1500
1000
500
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
x27


Distribution of x28
Count
0
1000
2000
3000
4000
5000
6000
60
80
100
120
140
x28

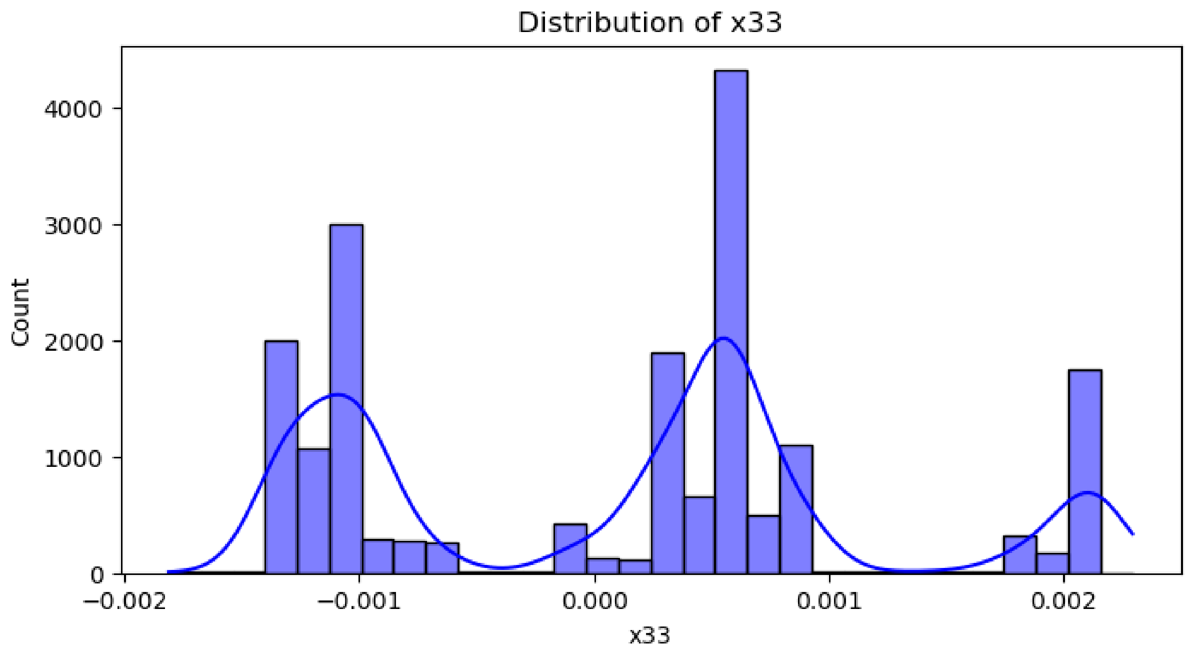

Distribution of x29
Count
0
500
1000
1500
2000
−200
−100
0
100
200
300
x29


Distribution of x30
Count
8000
6000
4000
2000
0
−0.2
0.0
0.2
0.4
0.6
x30

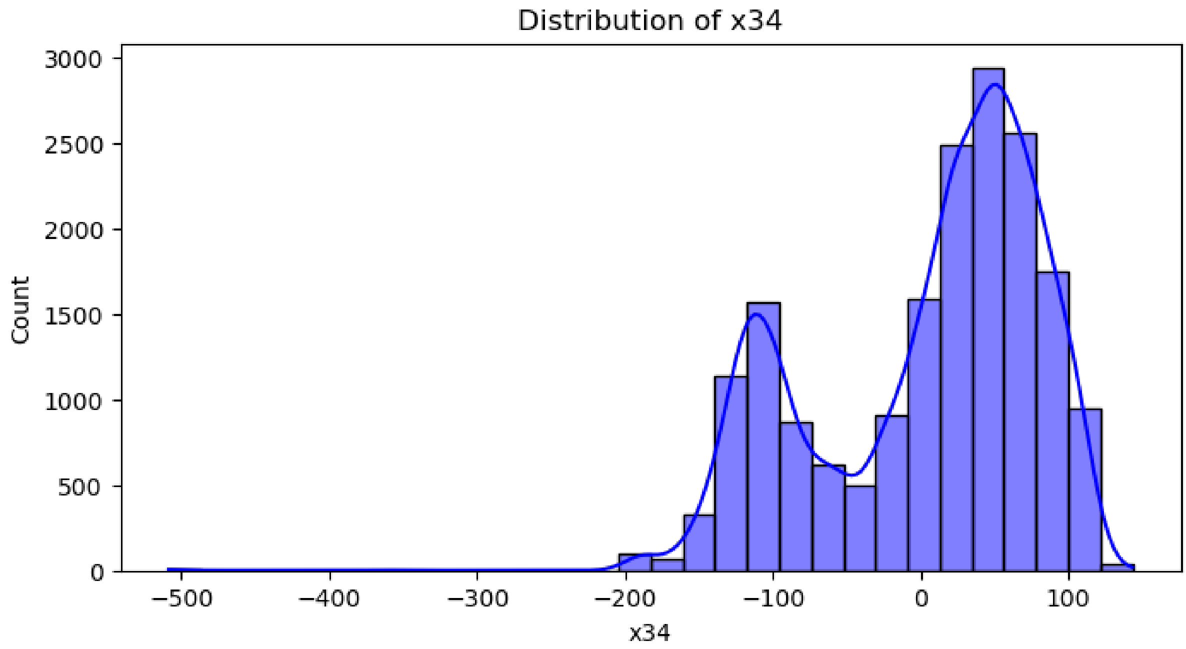

Distribution of x31
Count
2500
2000
1500
1000
500
0
−0.4
−0.2
0.0
0.2
0.4
x31

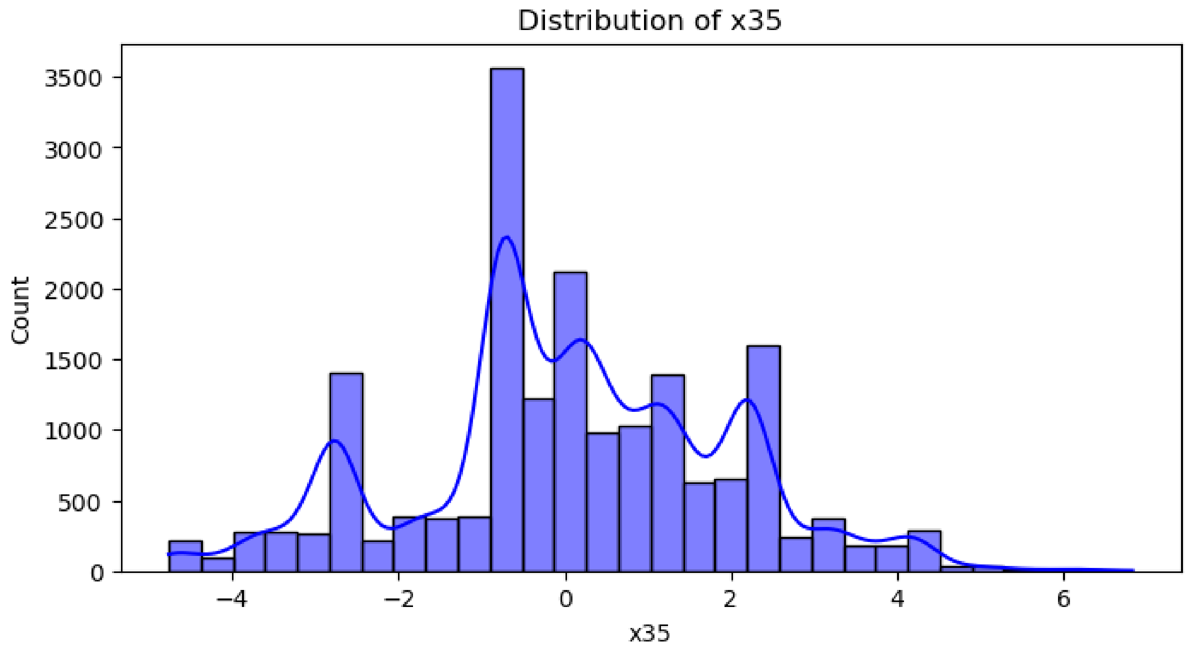


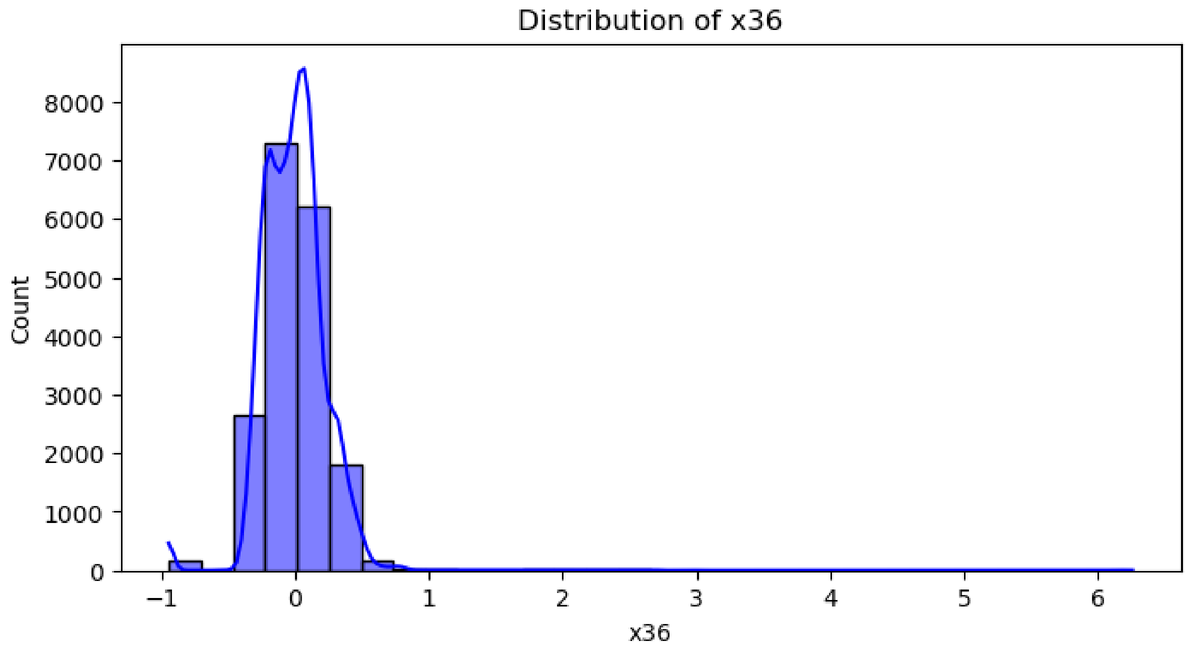
Distribution of x32
17500
15000
12500
10000
7500
5000
2500
0
Count
−600
−500
−400
−300
−200
−100
0
x32

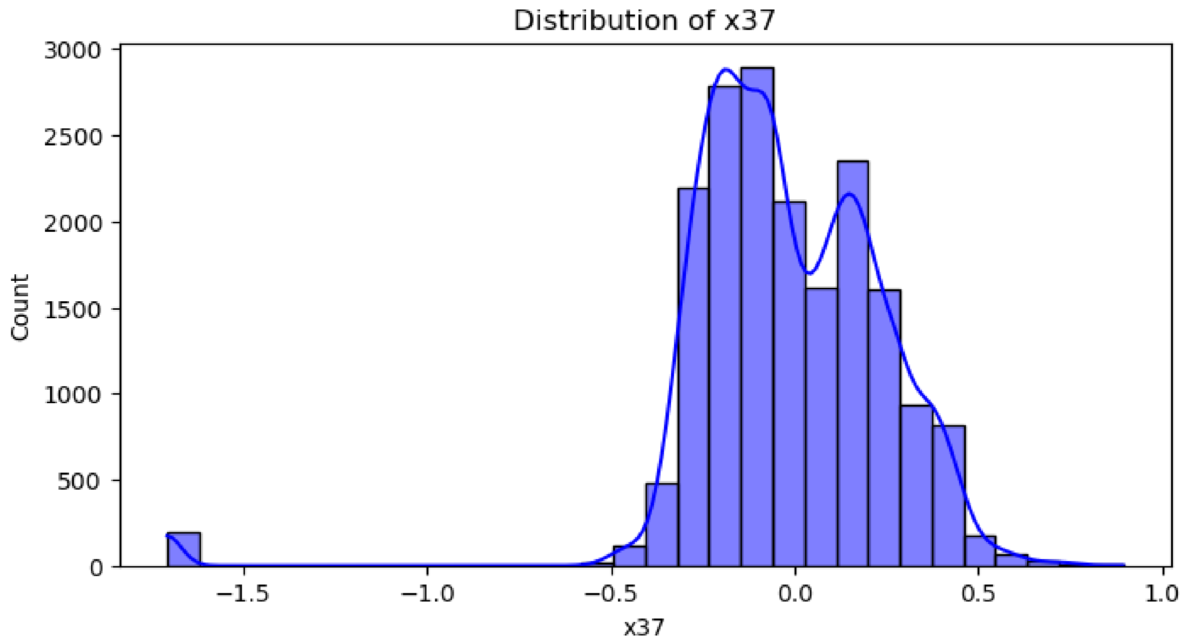


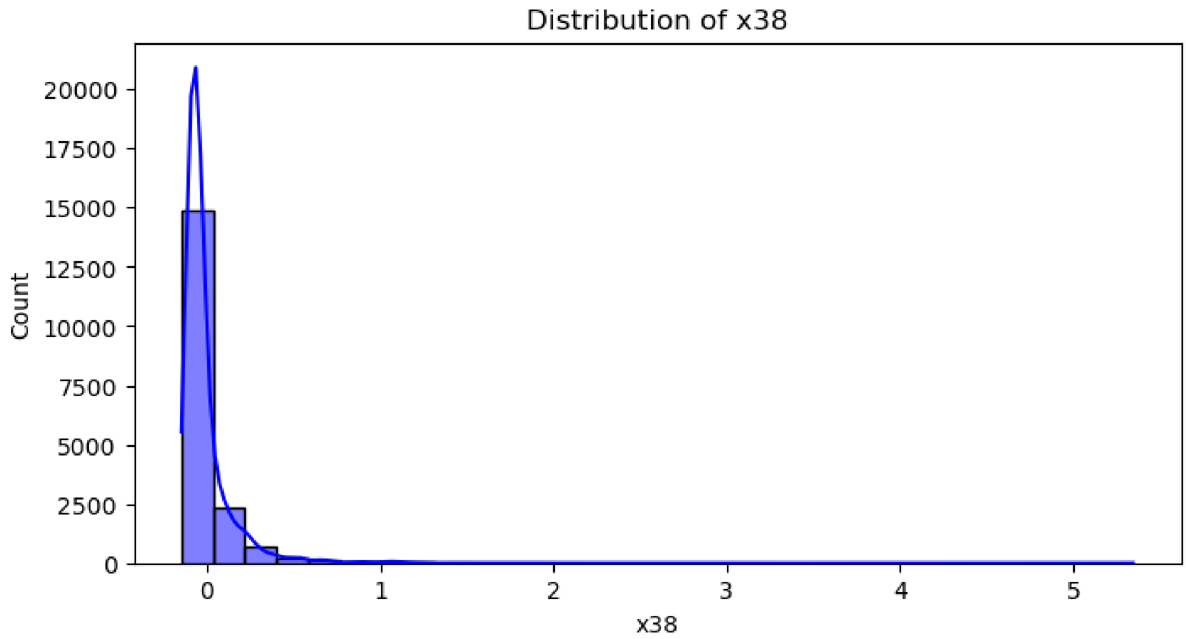
Distribution of x33
4000
3000
2000
1000
0
Count
−0.002
−0.001
0.000
0.001
0.002
x33

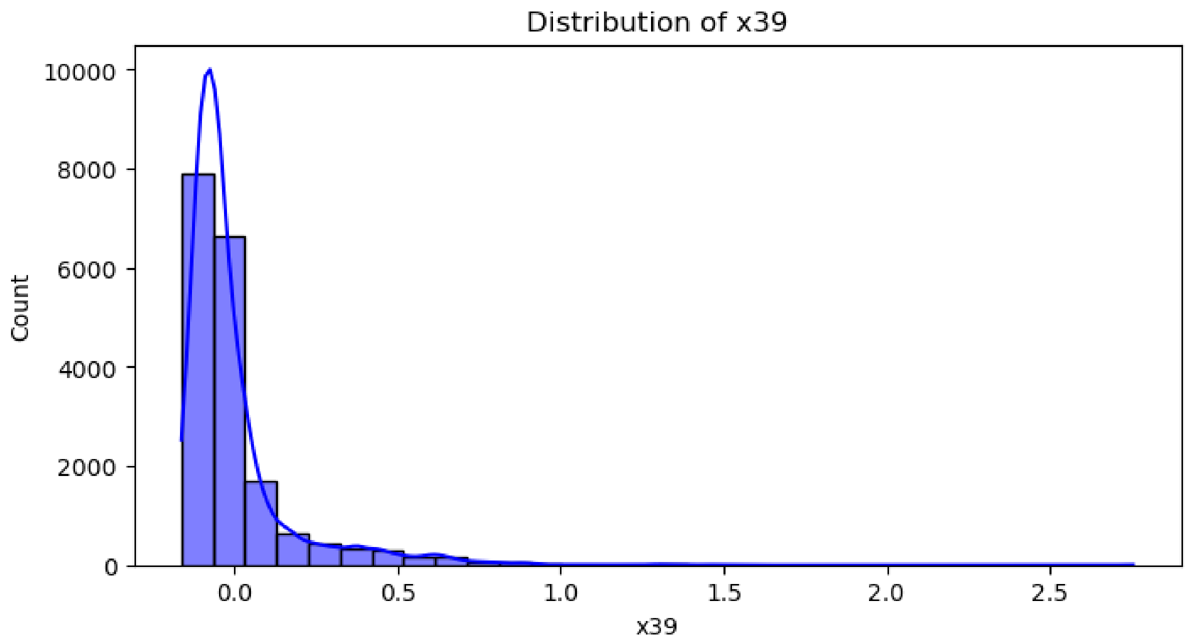

Distribution of x34
Count
x34
3000
2500
2000
1500
1000
500
0
−500
−400
−300
−200
−100
0
100

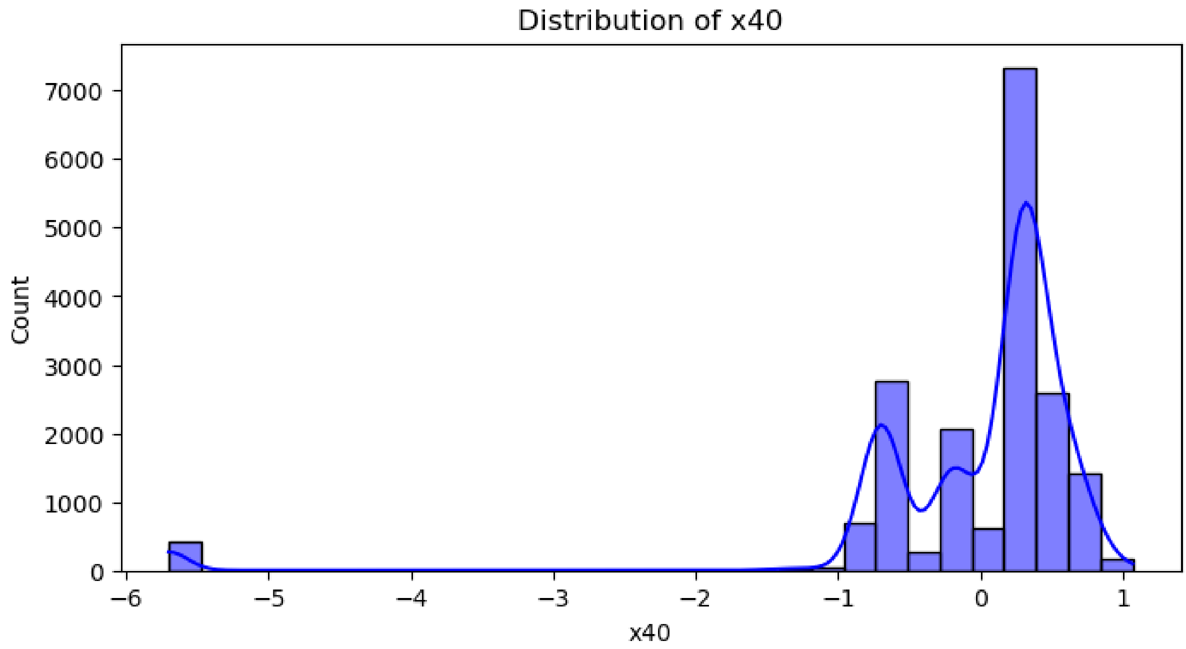

Distribution of x35
Count
x35
3500
3000
2500
2000
1500
1000
500
0
−4
−2
0
2
4
6


Distribution of x36
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
−1
0
1
2
3
4
5
6
x36

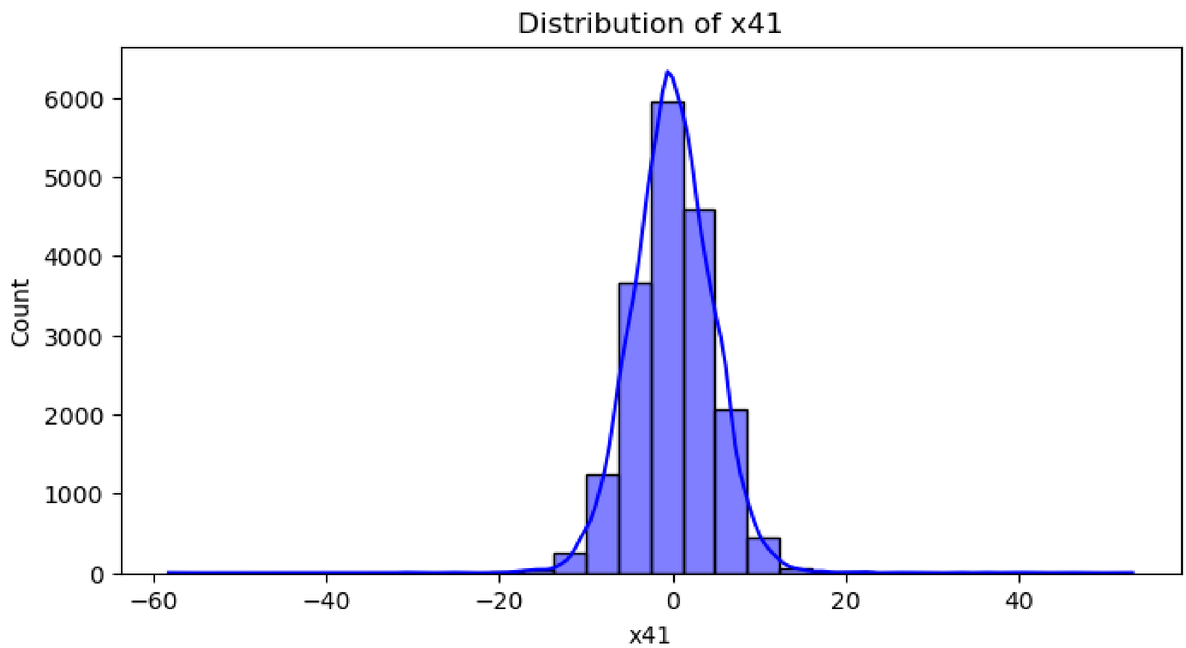

Distribution of x37
Count
3000
2500
2000
1500
1000
500
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
x37


Distribution of x38
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5
x38

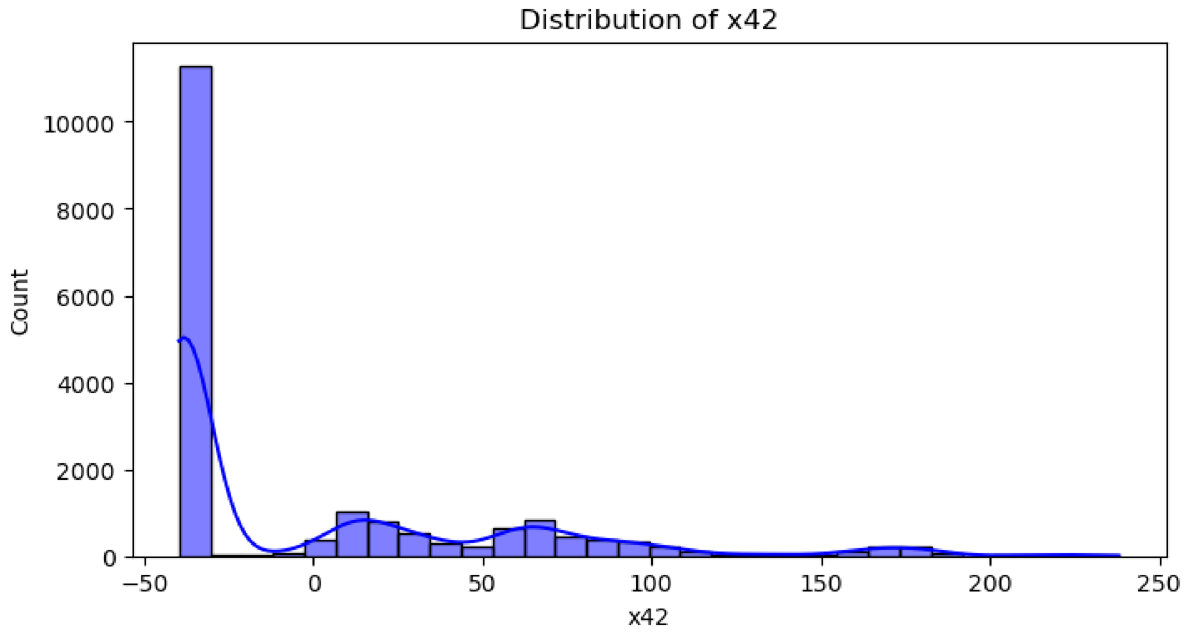

Distribution of x39
Count
10000
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
x39

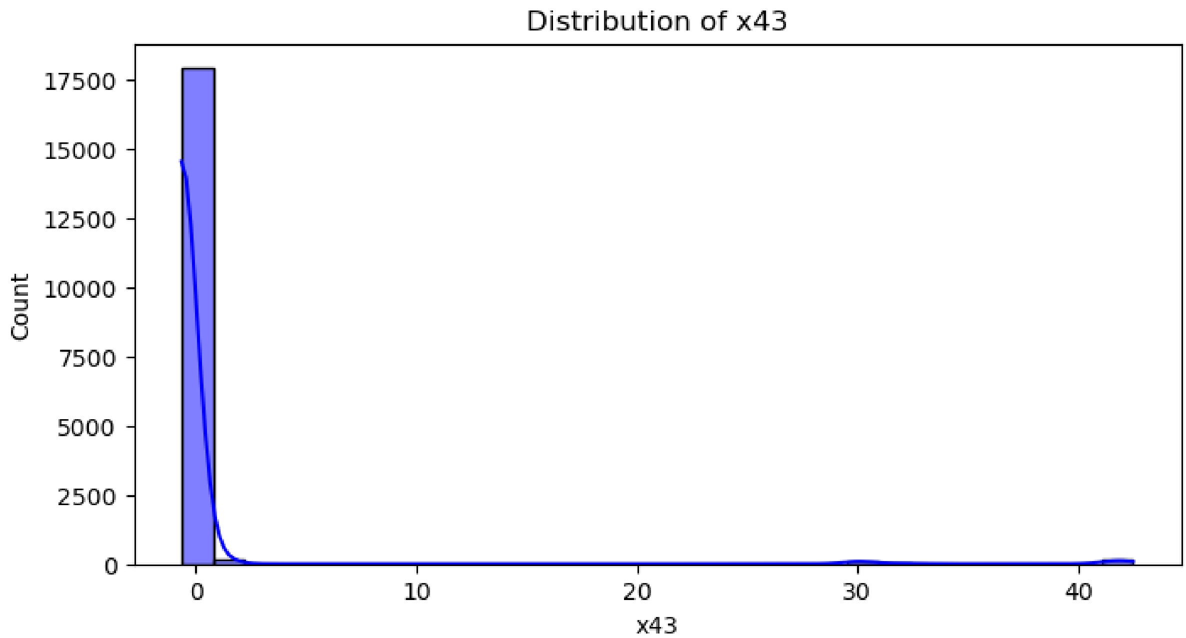


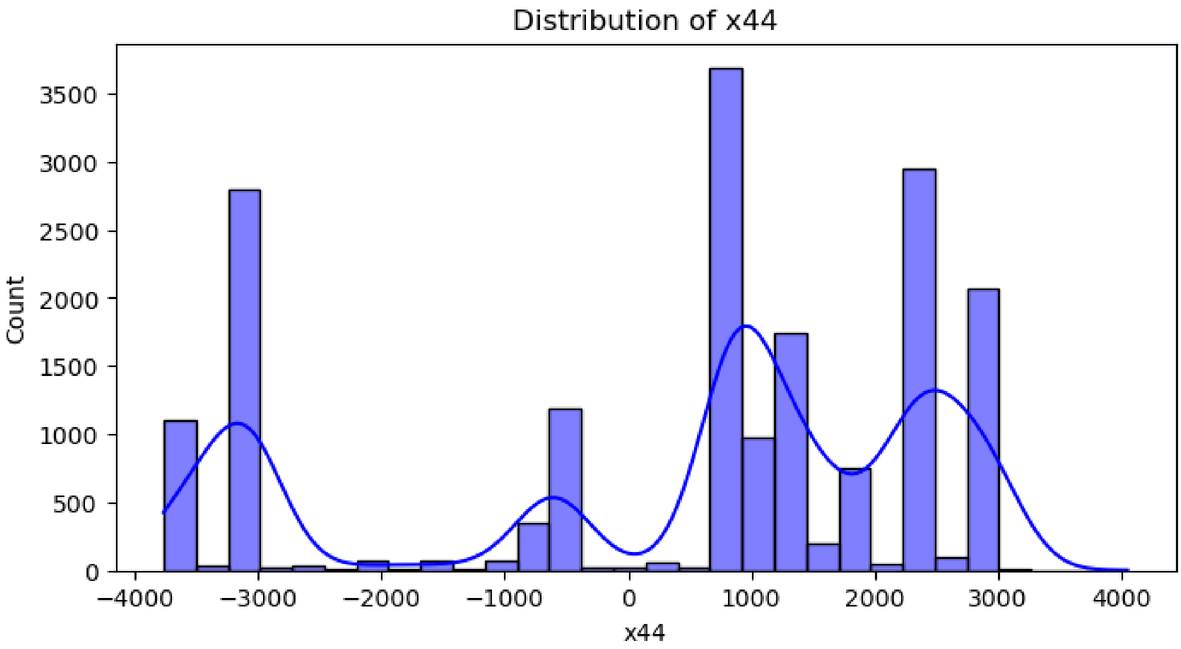
Distribution of x40
7000
6000
5000
4000
3000
2000
1000
0
Count
−6
−5
−4
−3
−2
−1
0
1
x40

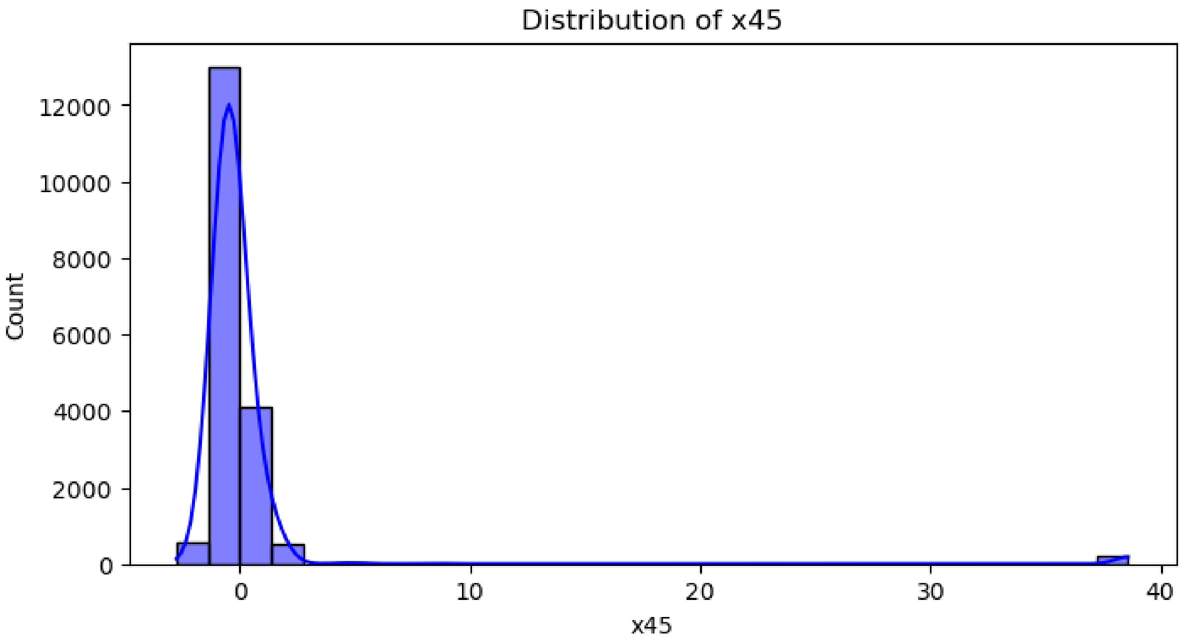


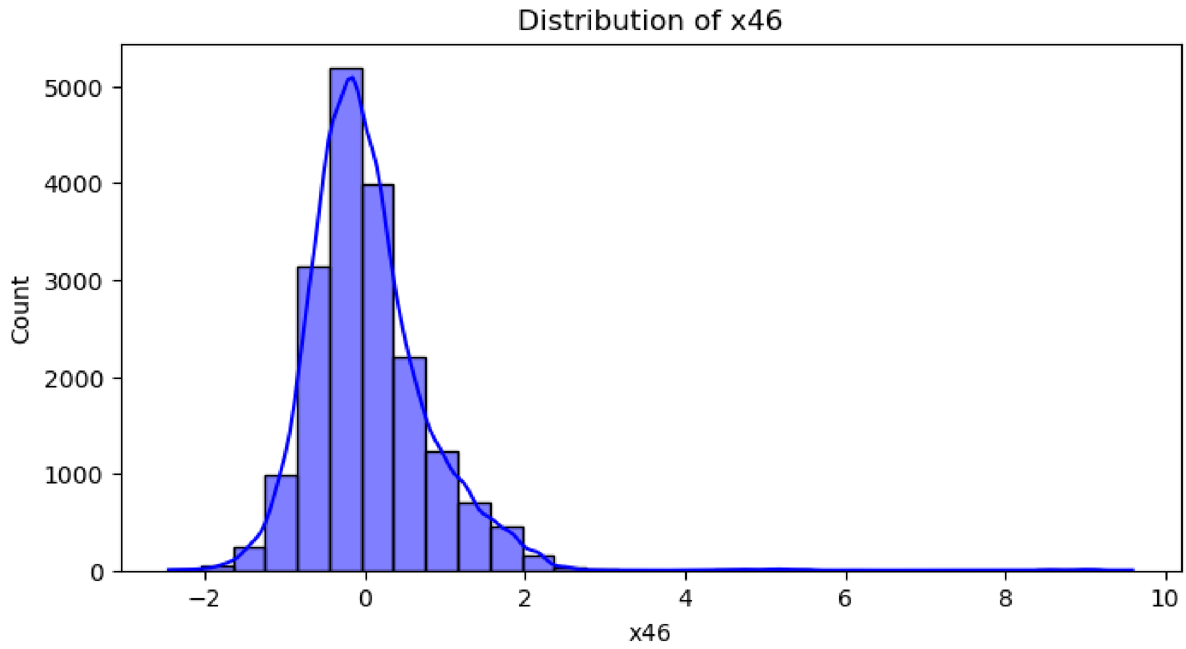
Distribution of x41
6000
5000
4000
3000
2000
1000
0
Count
−60
−40
−20
0
20
40
x41


Distribution of x42
Count
x42
0
2000
4000
6000
8000
10000
−50
0
50
100
150
200
250

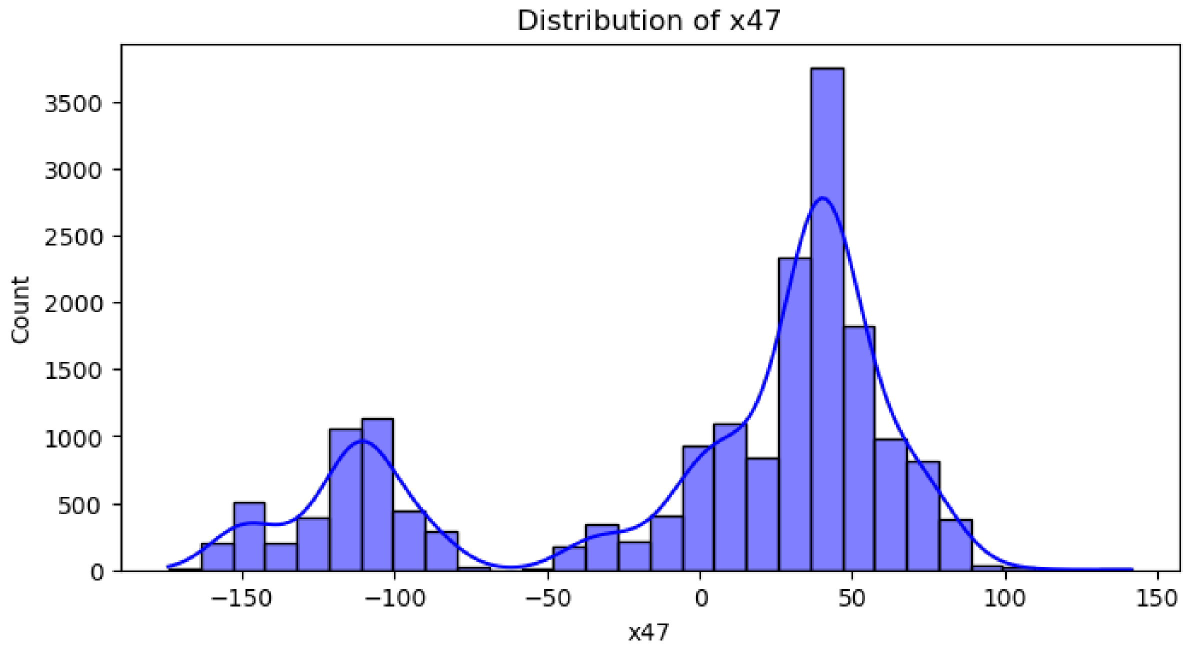

Distribution of x43
Count
x43
0
2500
5000
7500
10000
12500
15000
17500
0
10
20
30
40


Distribution of x44
Count
3500
3000
2500
2000
1500
1000
500
0
−4000
−3000
−2000
−1000
0
1000
2000
3000
4000
x44

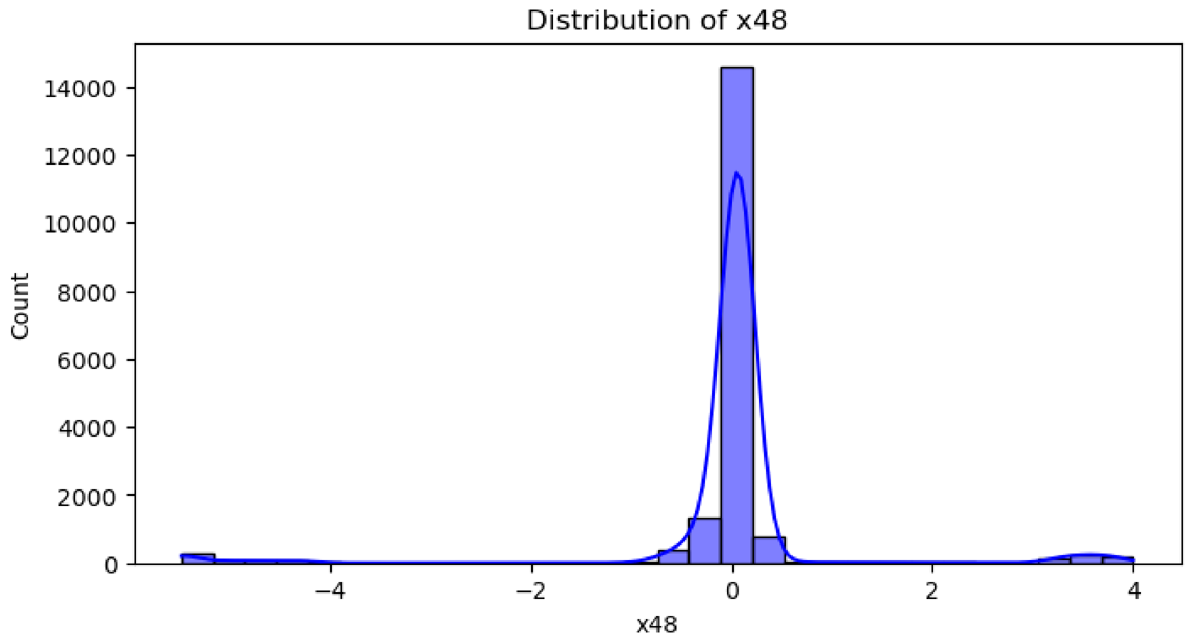

Distribution of x45
Count
12000
10000
8000
6000
4000
2000
0
0
10
20
30
40
x45



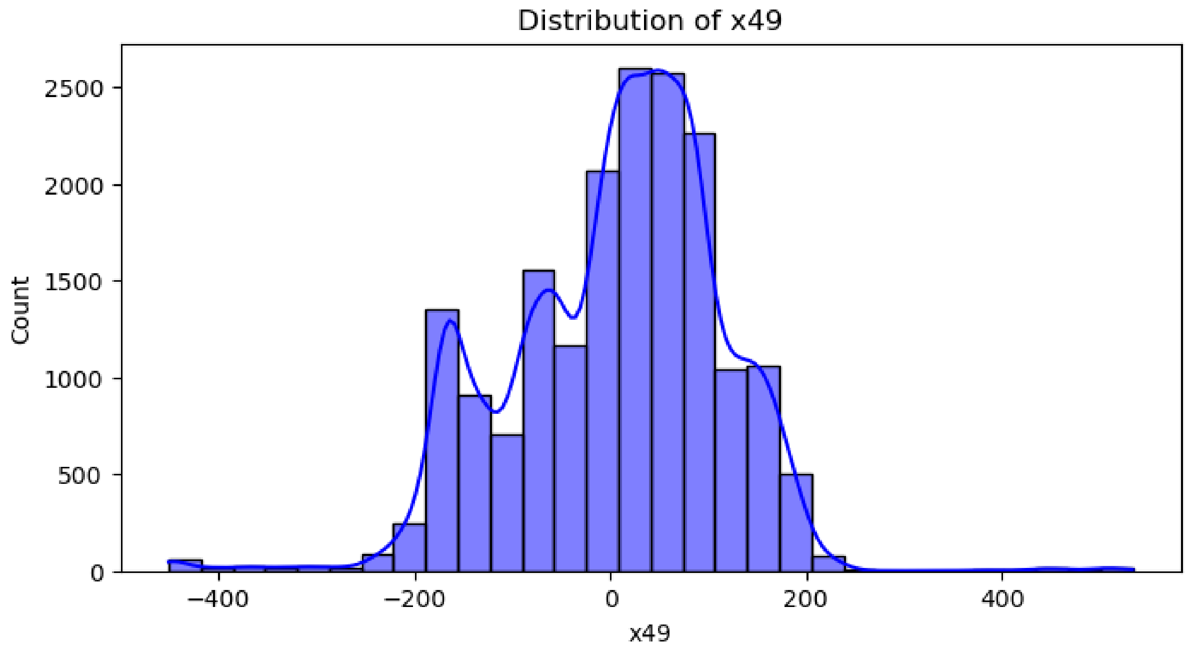
Distribution of x46
Count
x46
0
1000
2000
3000
4000
5000
−2
0
2
4
6
8
10



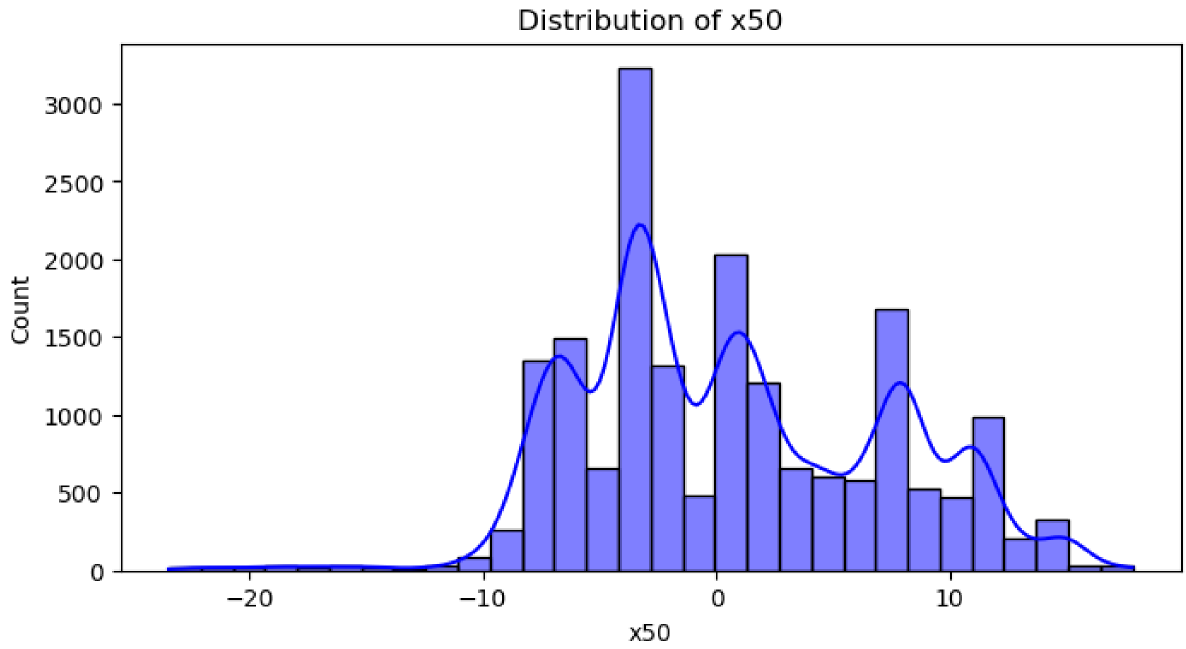
Distribution of x47
Count
x47
0
500
1000
1500
2000
2500
3000
3500
−150
−100
−50
0
50
100
150

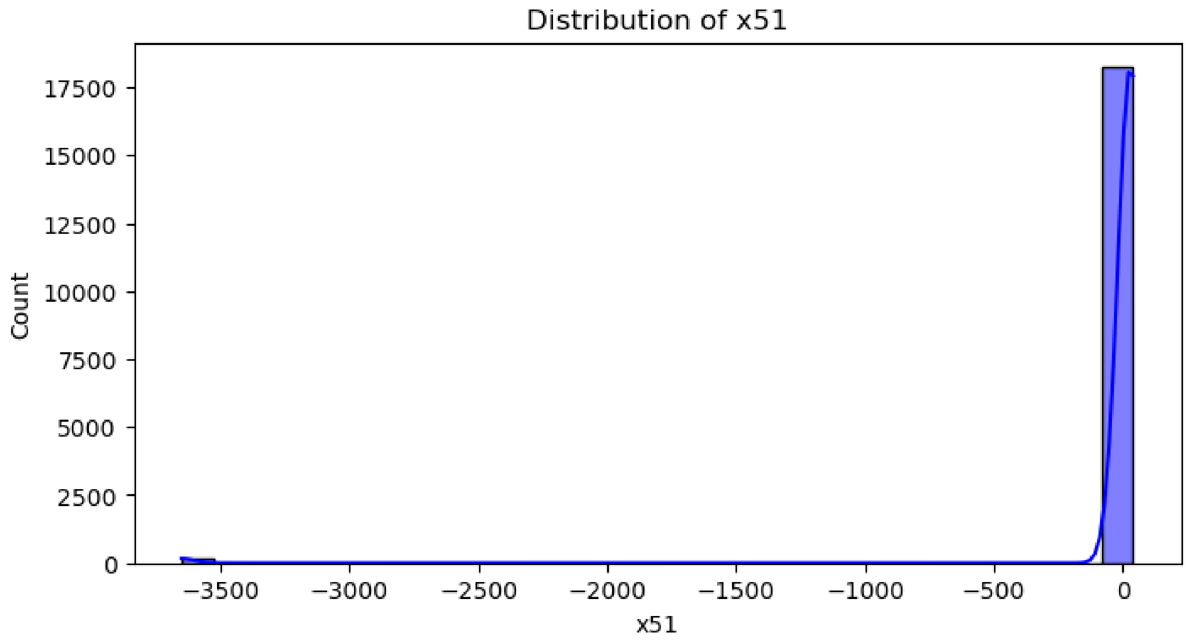


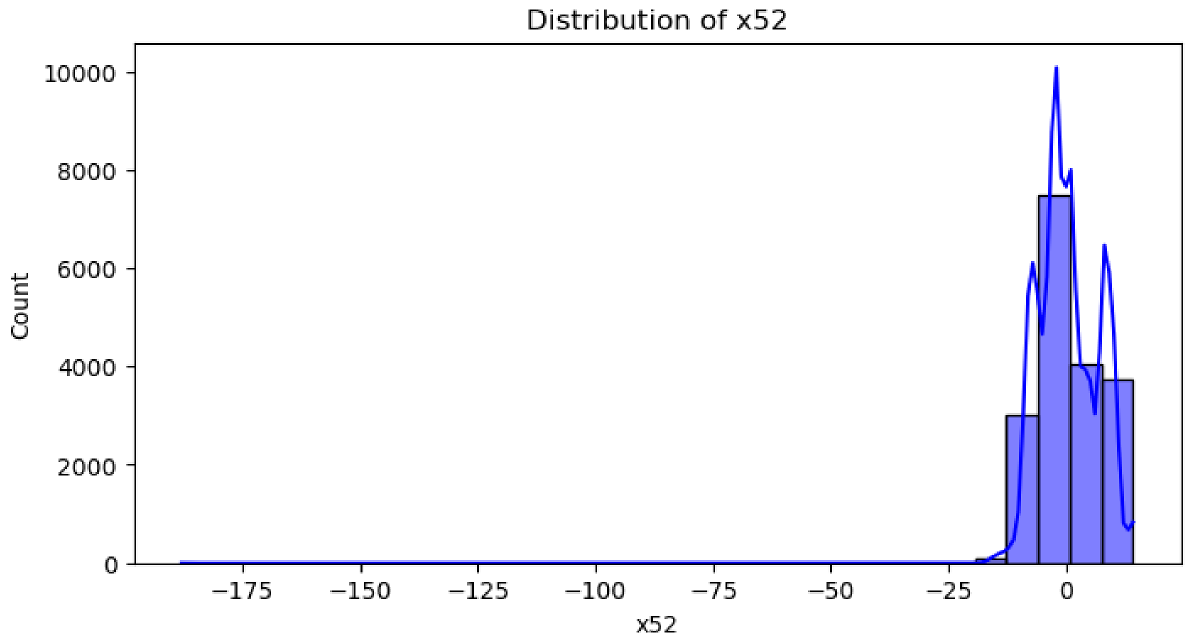
Distribution of x48
Count
x48
0
2000
4000
6000
8000
10000
12000
14000
−4
−2
0
2
4

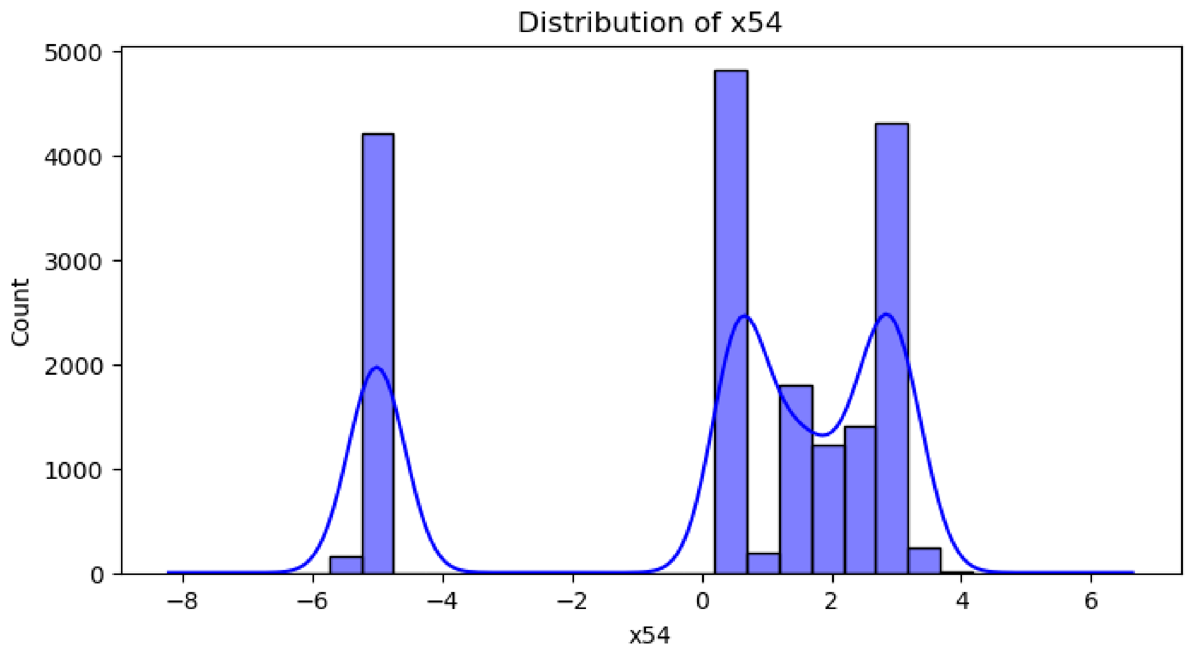


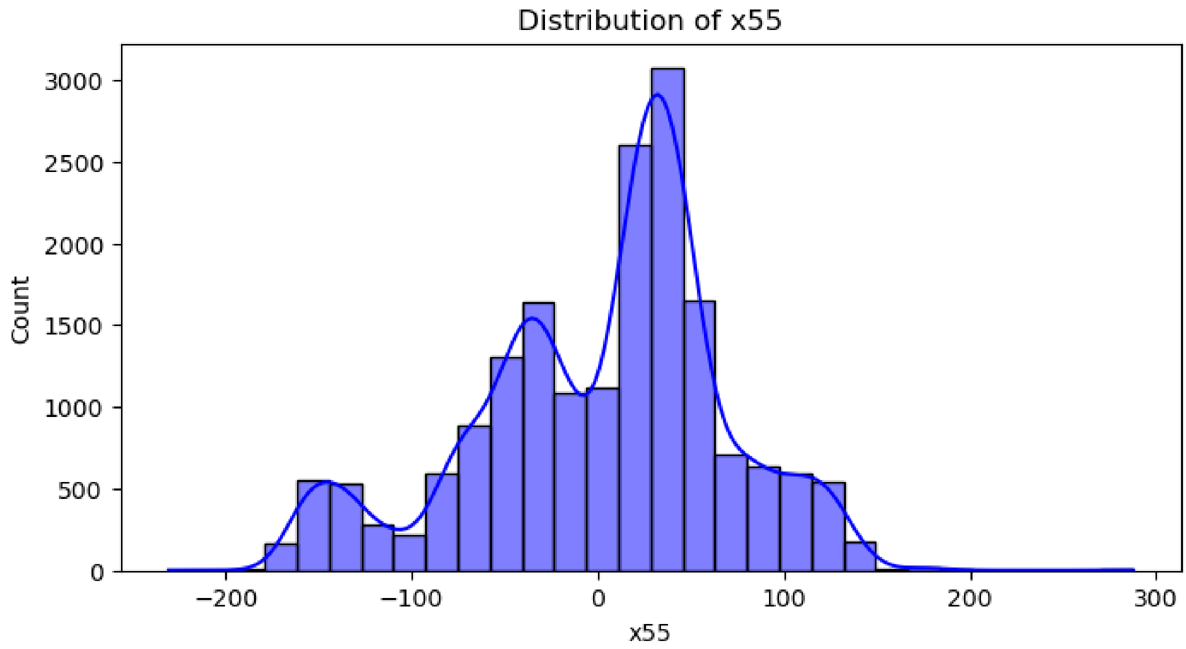
Distribution of x49
Count
x49
0
500
1000
1500
2000
2500
−400
−200
0
200
400


Distribution of x50
3000
2500
2000
1500
1000
500
0
Count
−20
−10
0
10
x50

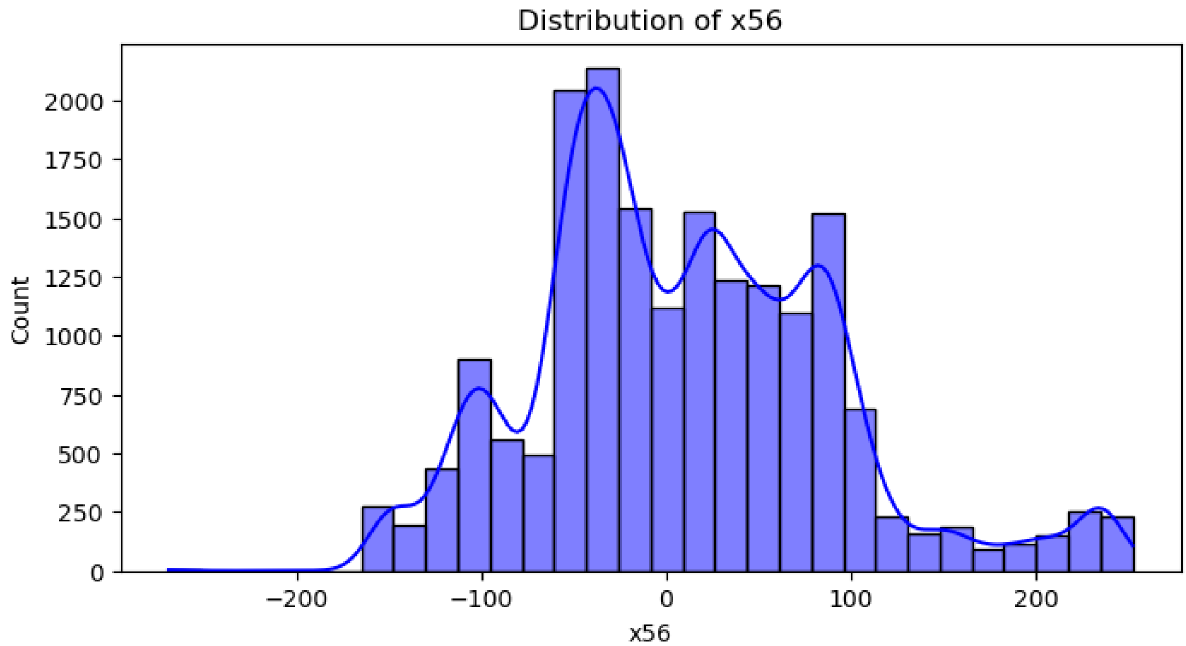

Distribution of x51
17500
15000
12500
10000
7500
5000
2500
0
Count
−3500
−3000
−2500
−2000
−1500
−1000
−500
0
x51


Distribution of x52
Count
x52
10000
8000
6000
4000
2000
0
−175
−150
−125
−100
−75
−50
−25
0

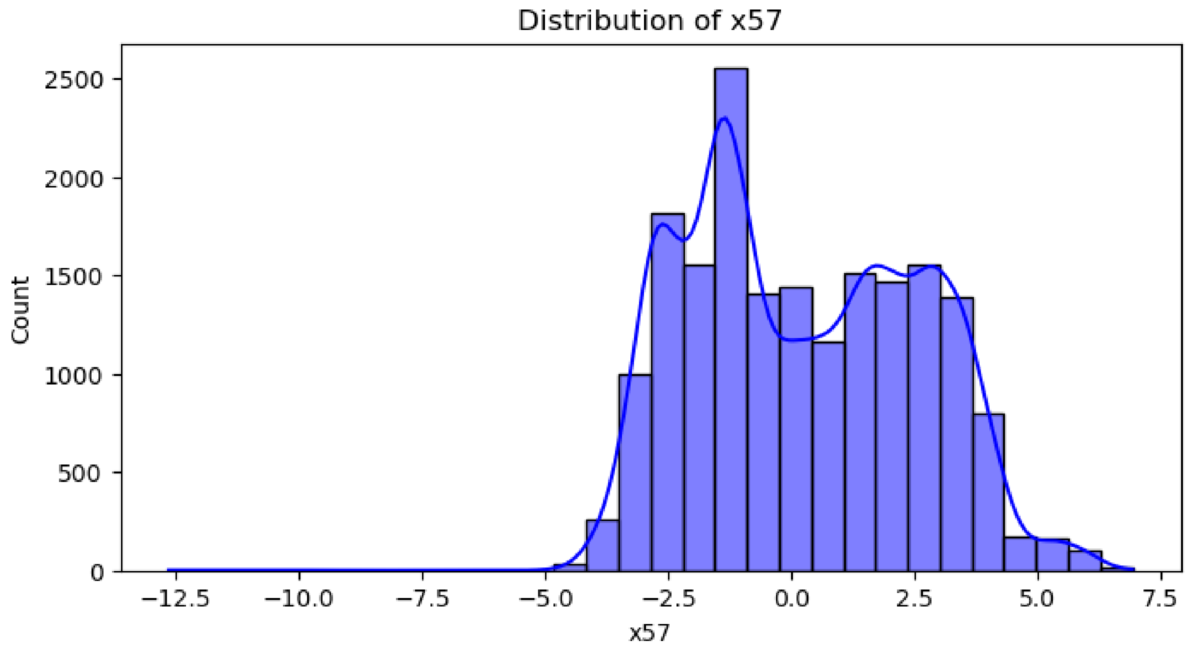

Distribution of x54
Count
x54
5000
4000
3000
2000
1000
0
−8
−6
−4
−2
0
2
4
6



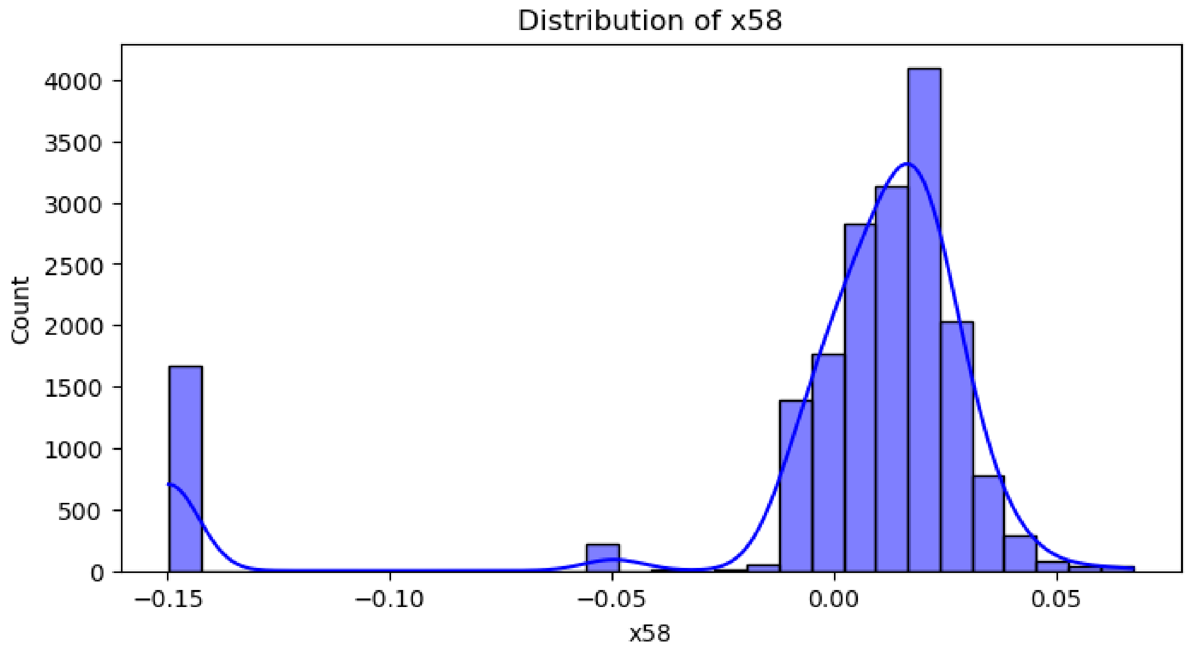
Distribution of x55
Count
x55
0
500
1000
1500
2000
2500
3000
−200
−100
0
100
200
300



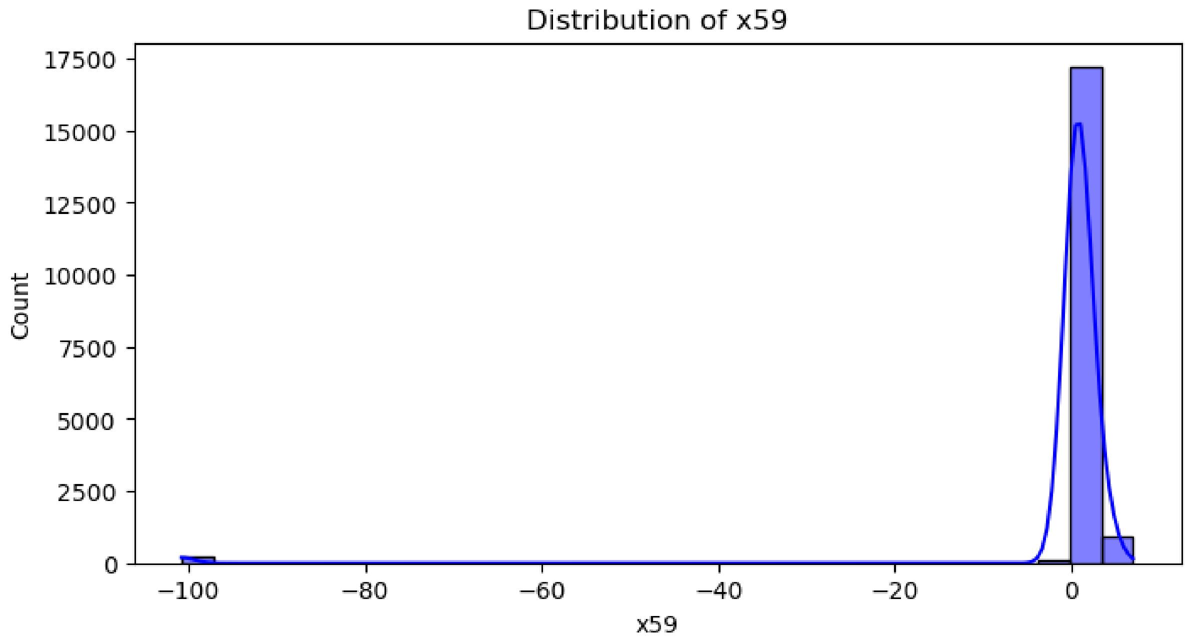
Distribution of x56
Count
x56
0
250
500
750
1000
1250
1500
1750
2000
−200
−100
0
100
200



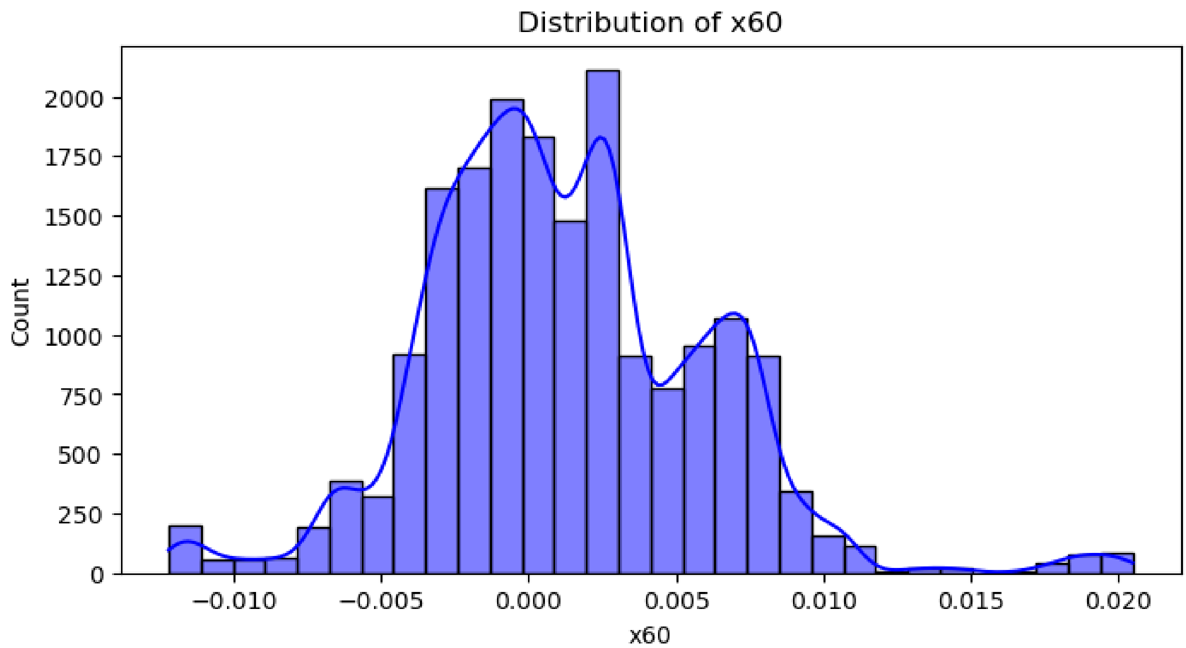
Distribution of x57
Count
x57



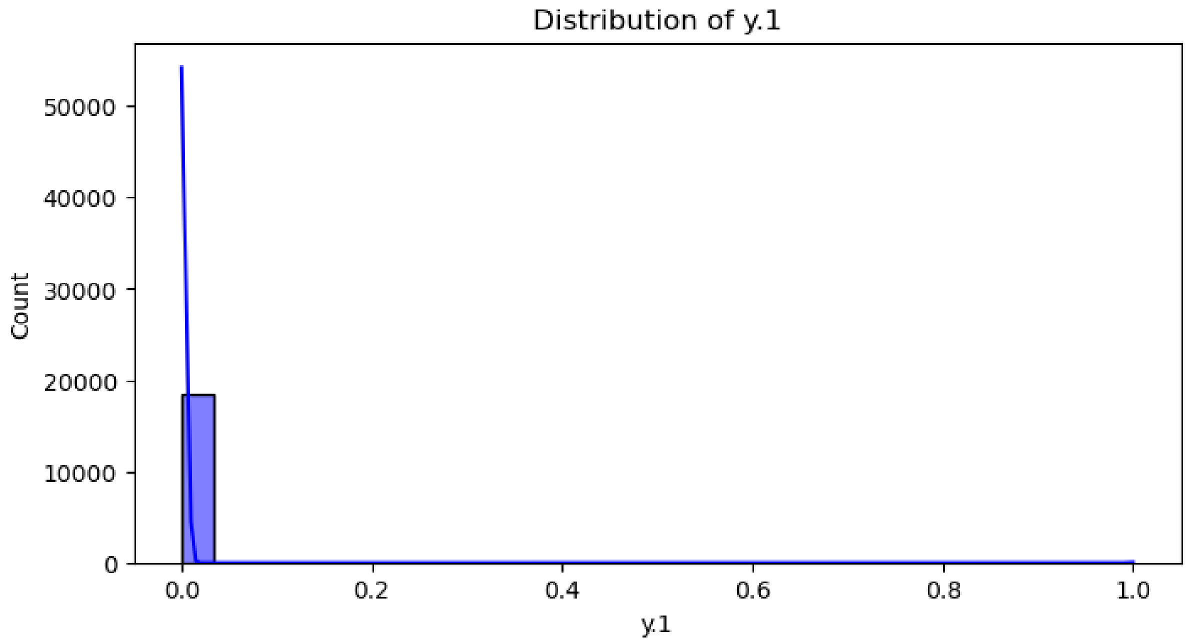
Distribution of x58
Count
x58



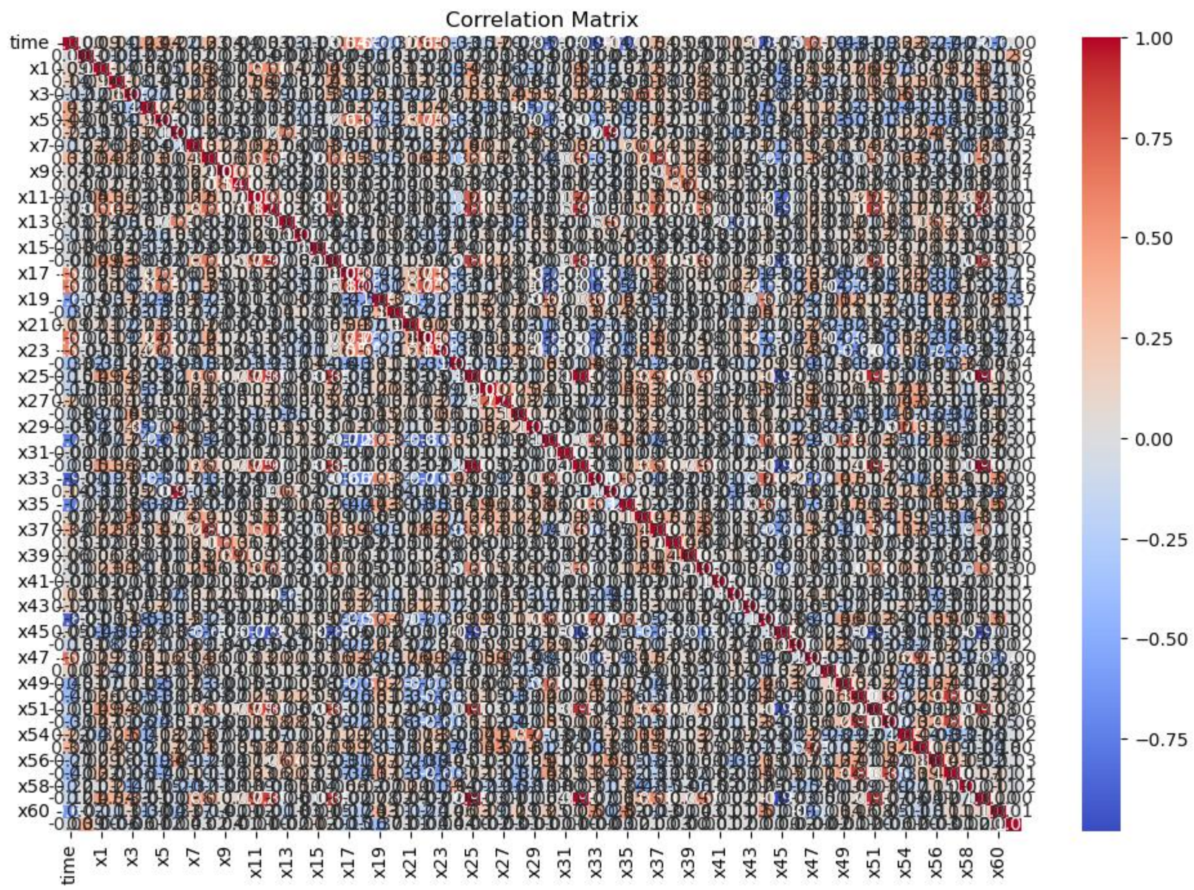
Distribution of x59
Count
x59



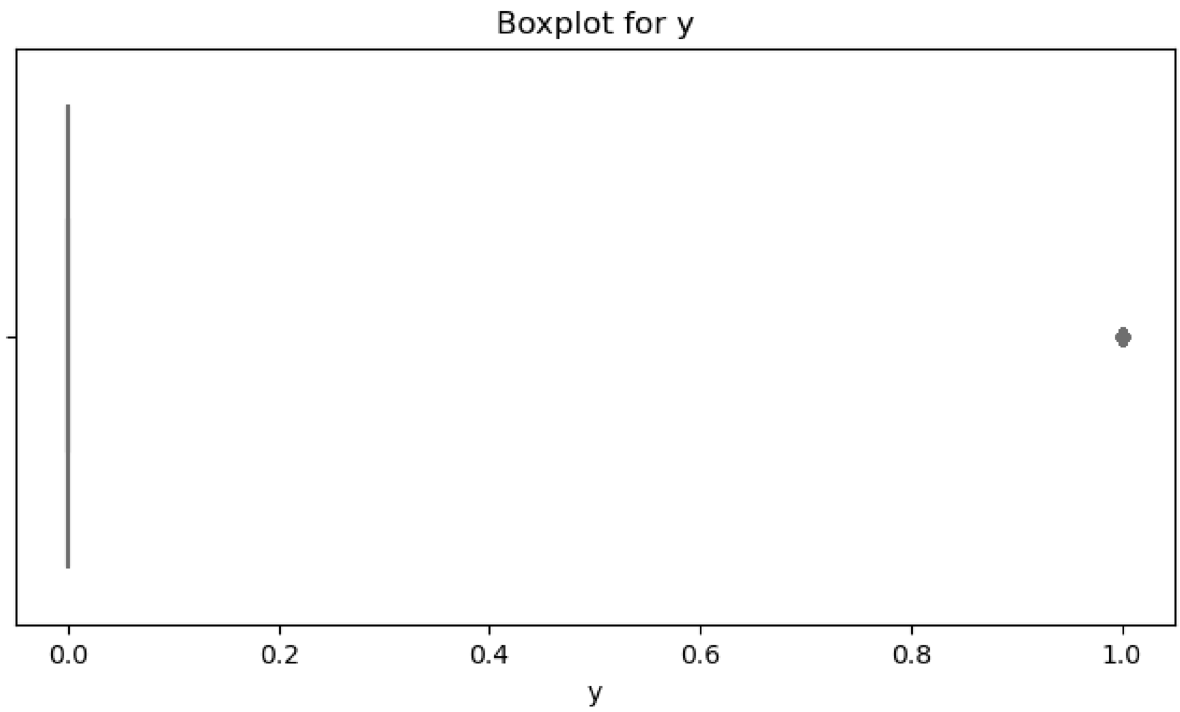
Distribution of x60
Count
x60

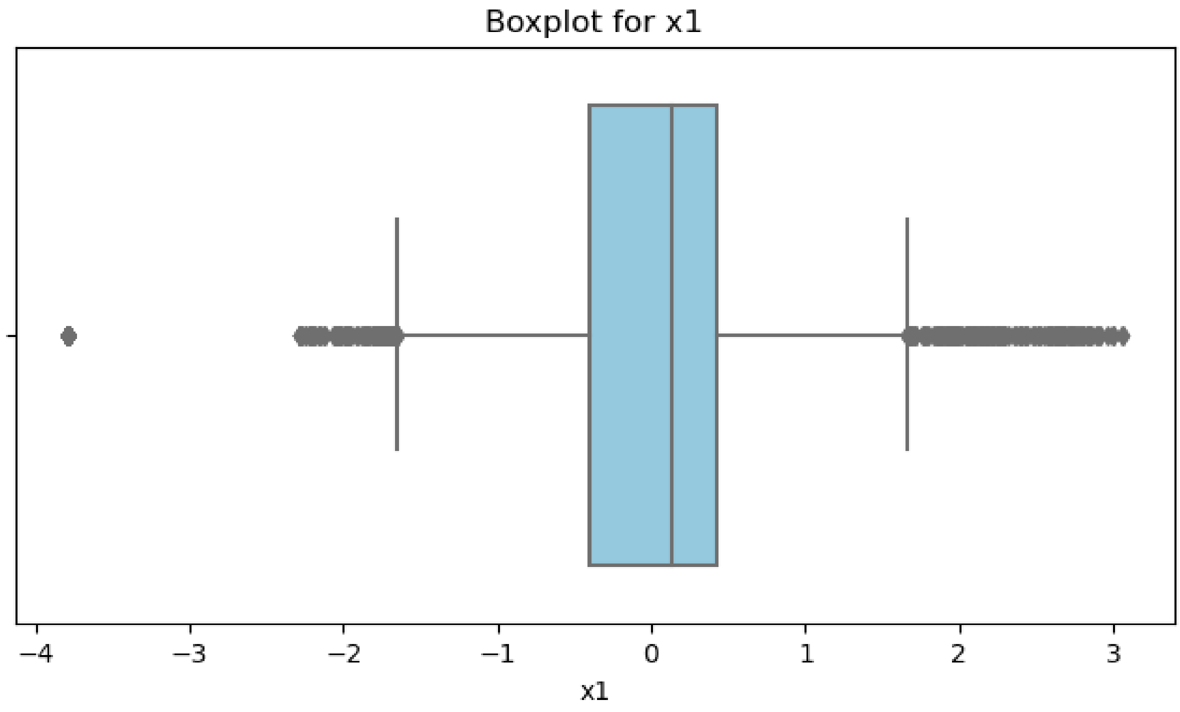


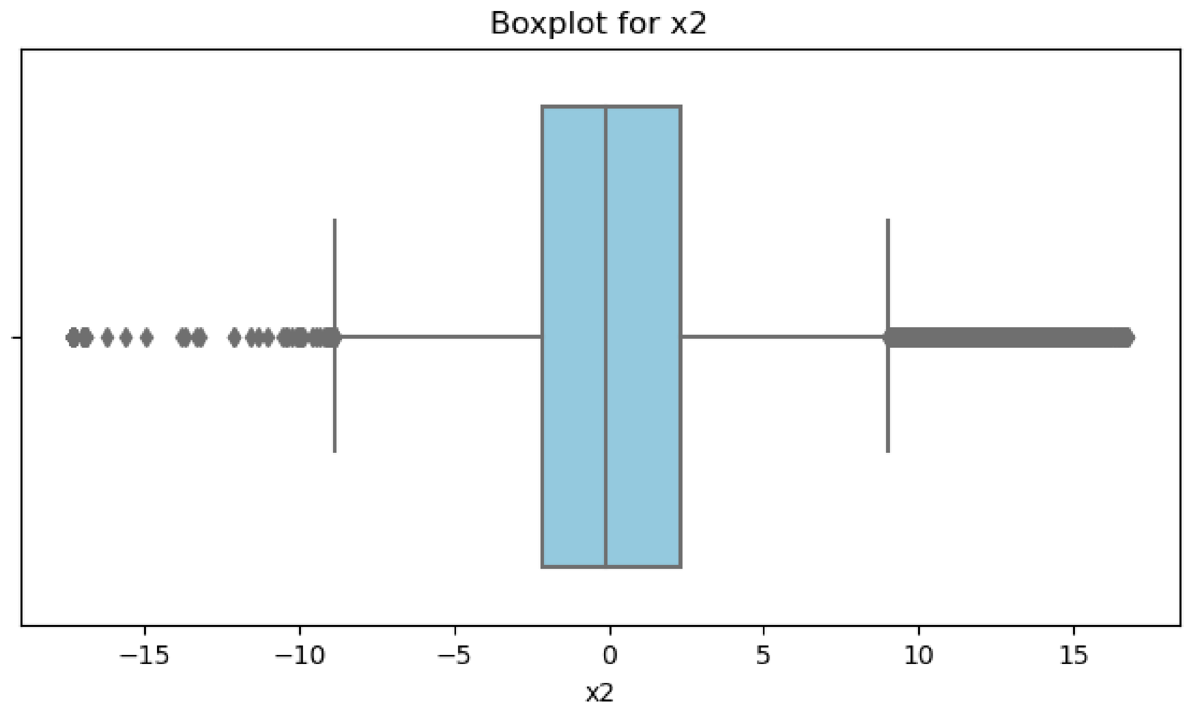
Correlation Heatmap:

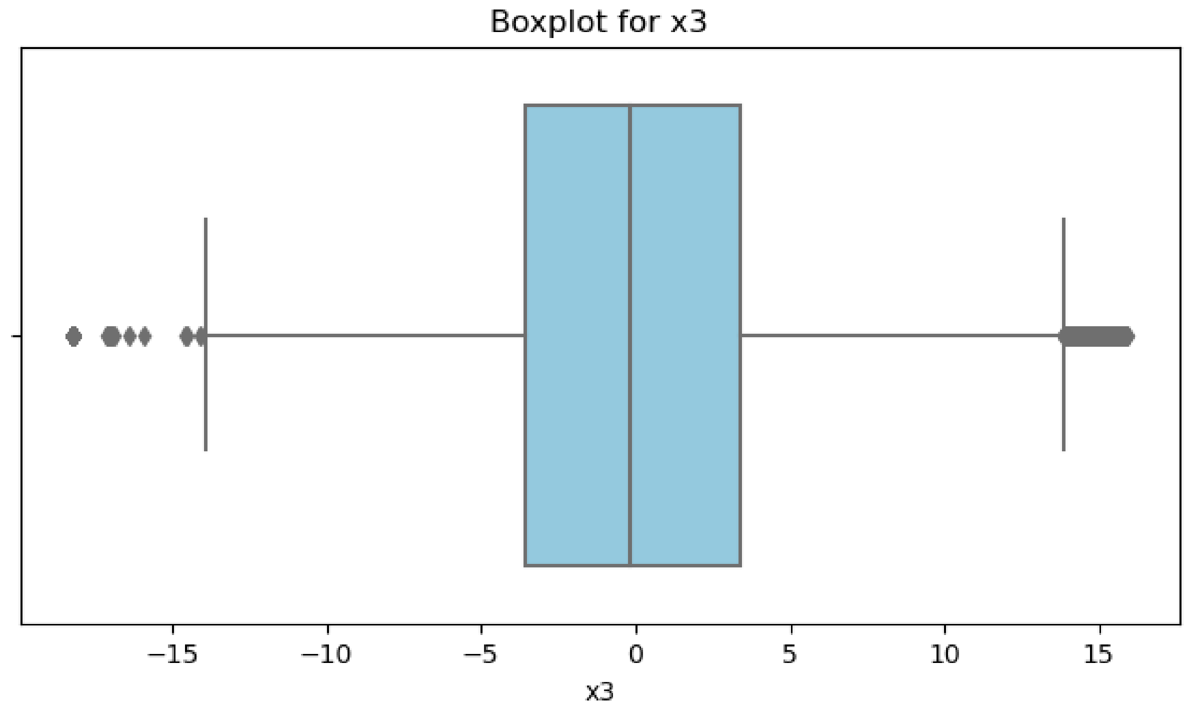


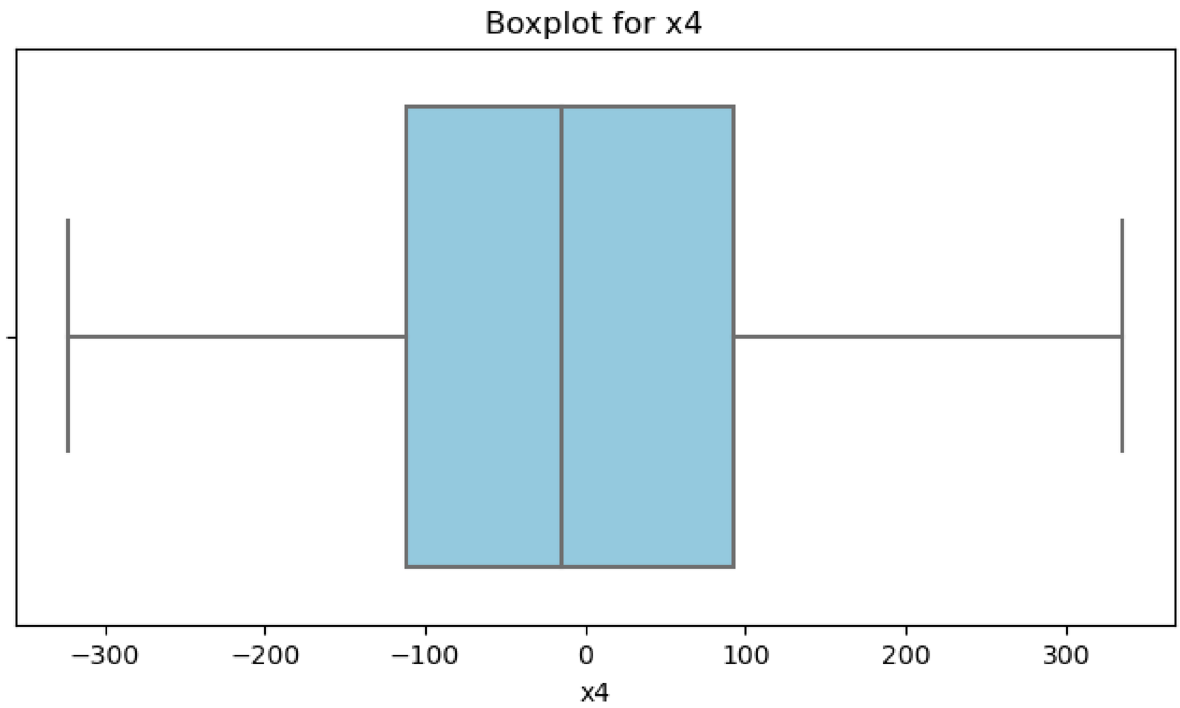
Generating boxplots for numerical features:


Boxplot for y
0.0
0.2
0.4
0.6
0.8
1.0
y

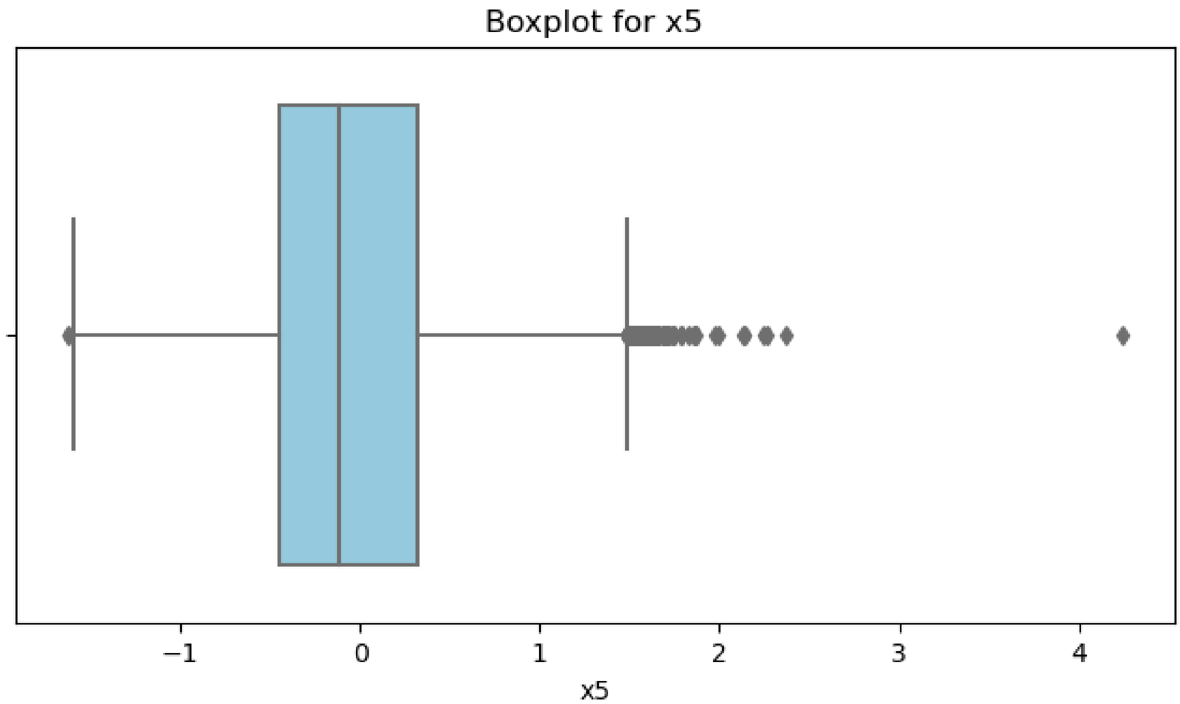

Boxplot for x1
−4
−3
−2
−1
0
1
2
3
x1


Boxplot for x2
−15
−10
−5
0
5
10
15
x2

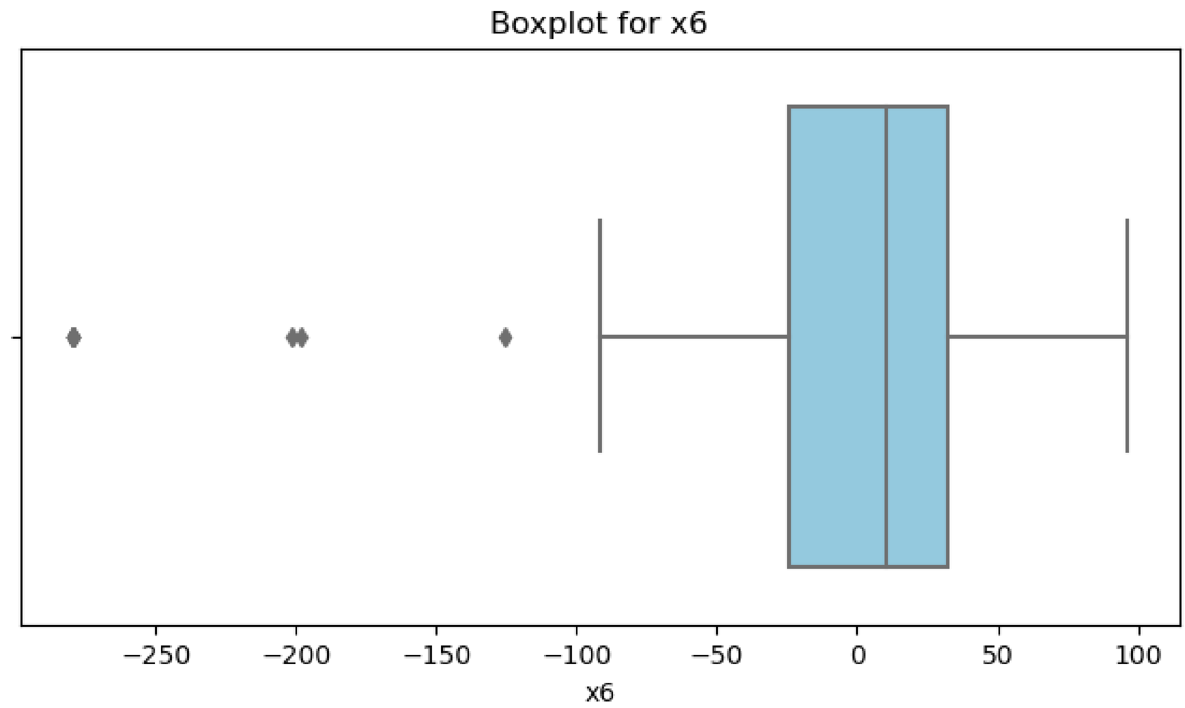

Boxplot for x3
−15
−10
−5
0
5
10
15
x3

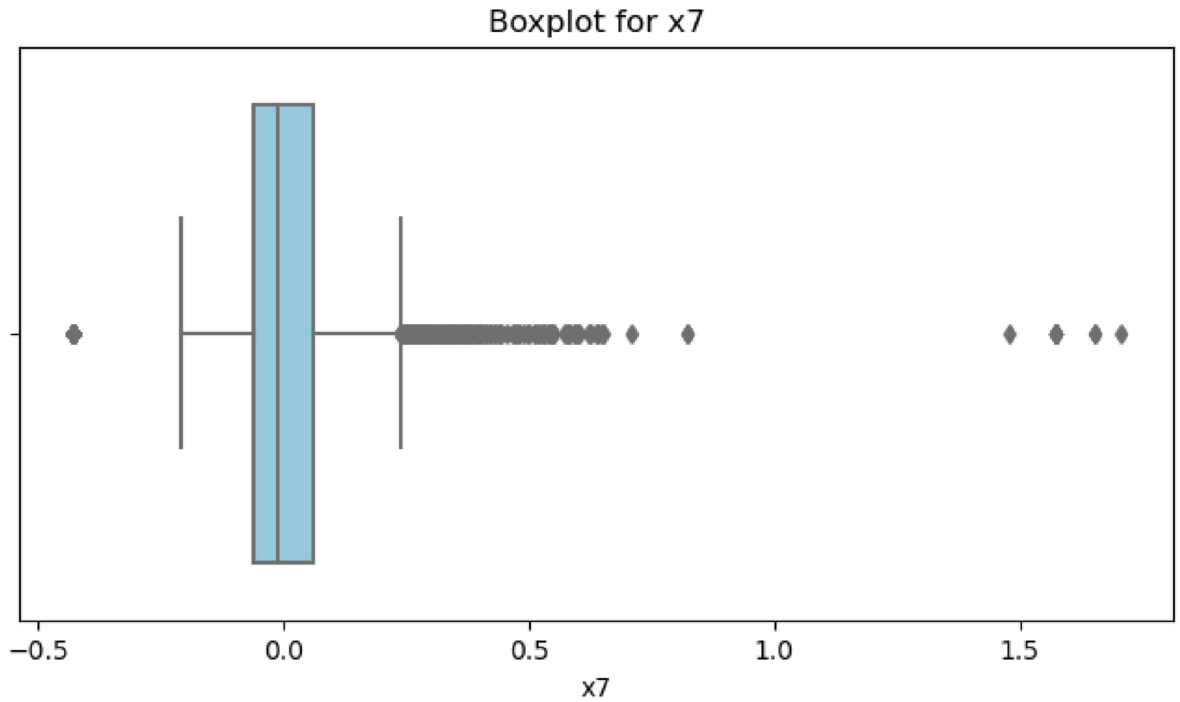


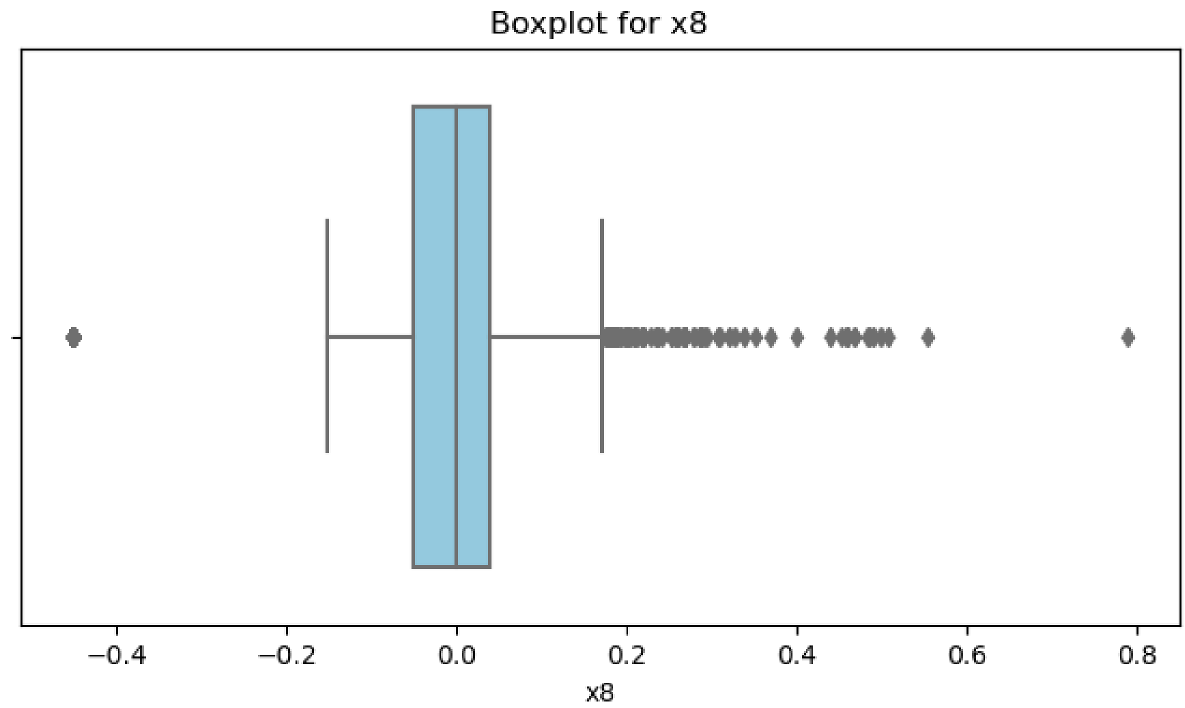
Boxplot for x4
−300
−200
−100
0
100
200
300
x4

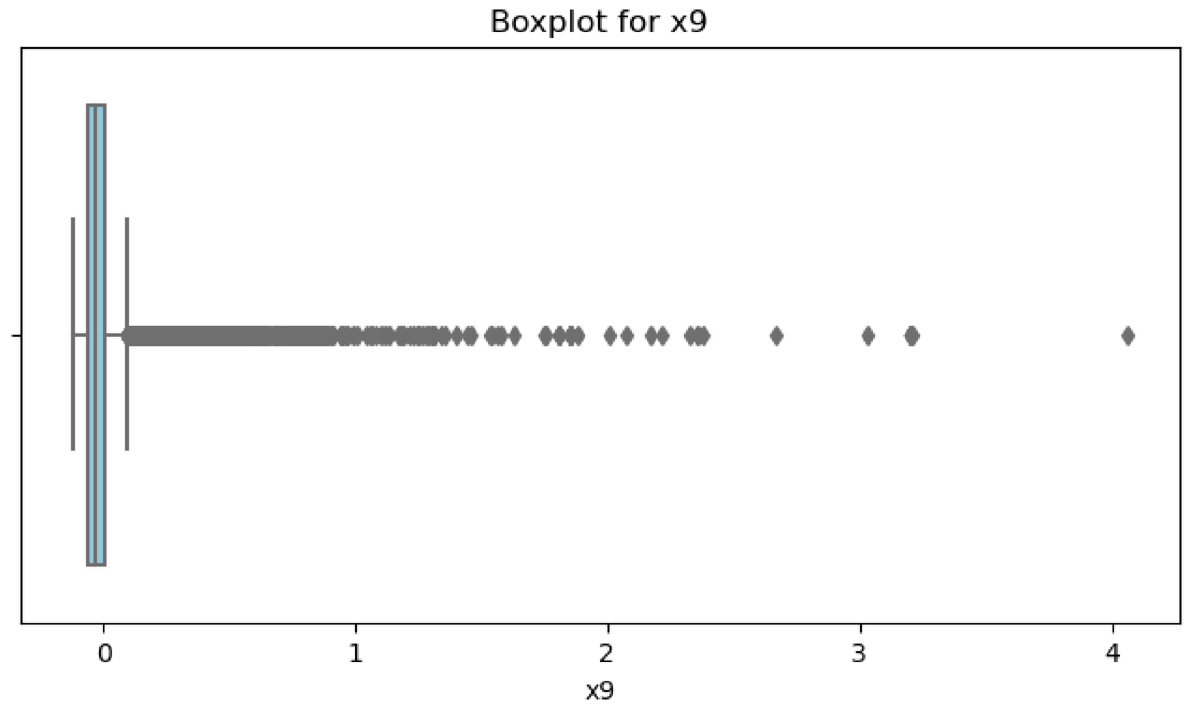


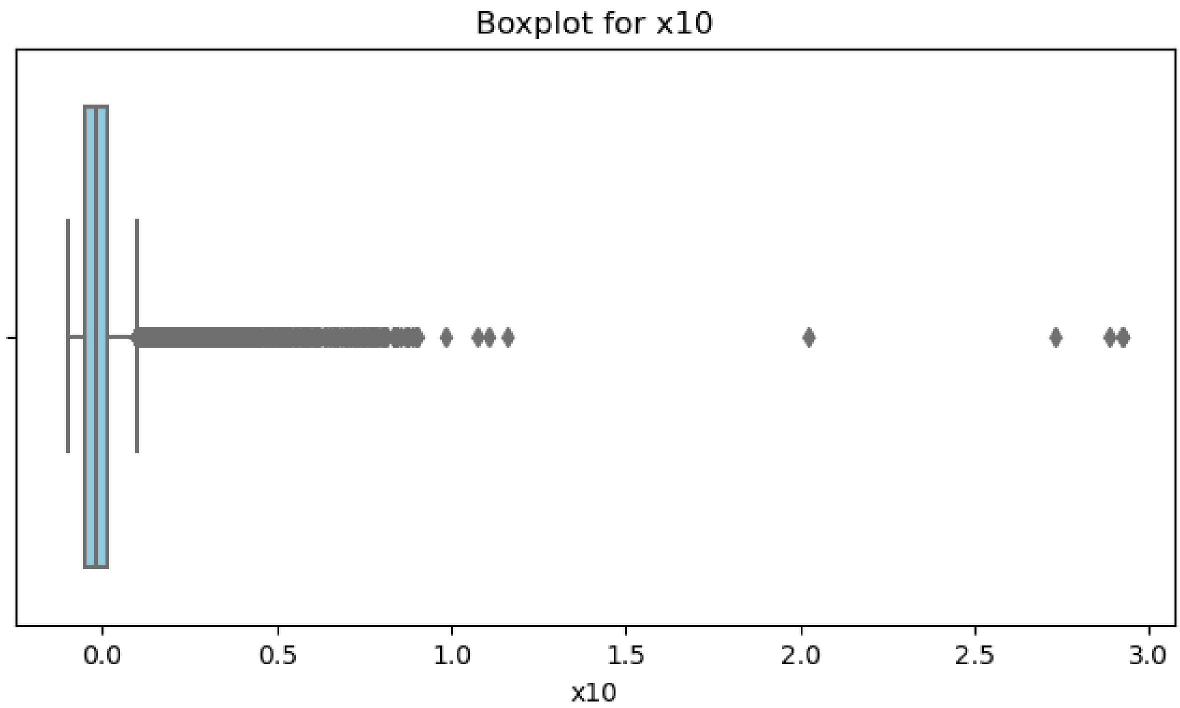
Boxplot for x5
−1
0
1
2
3
4
x5

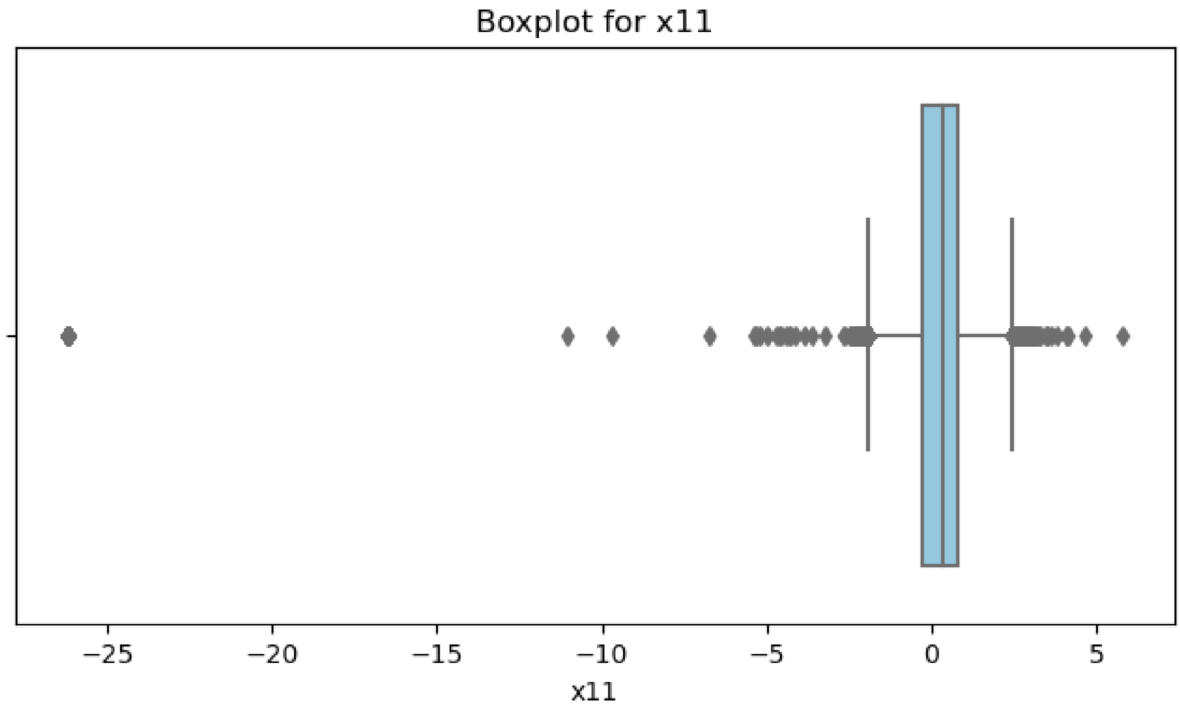

Boxplot for x6
−250
−200
−150
−100
−50
0
50
100
x6

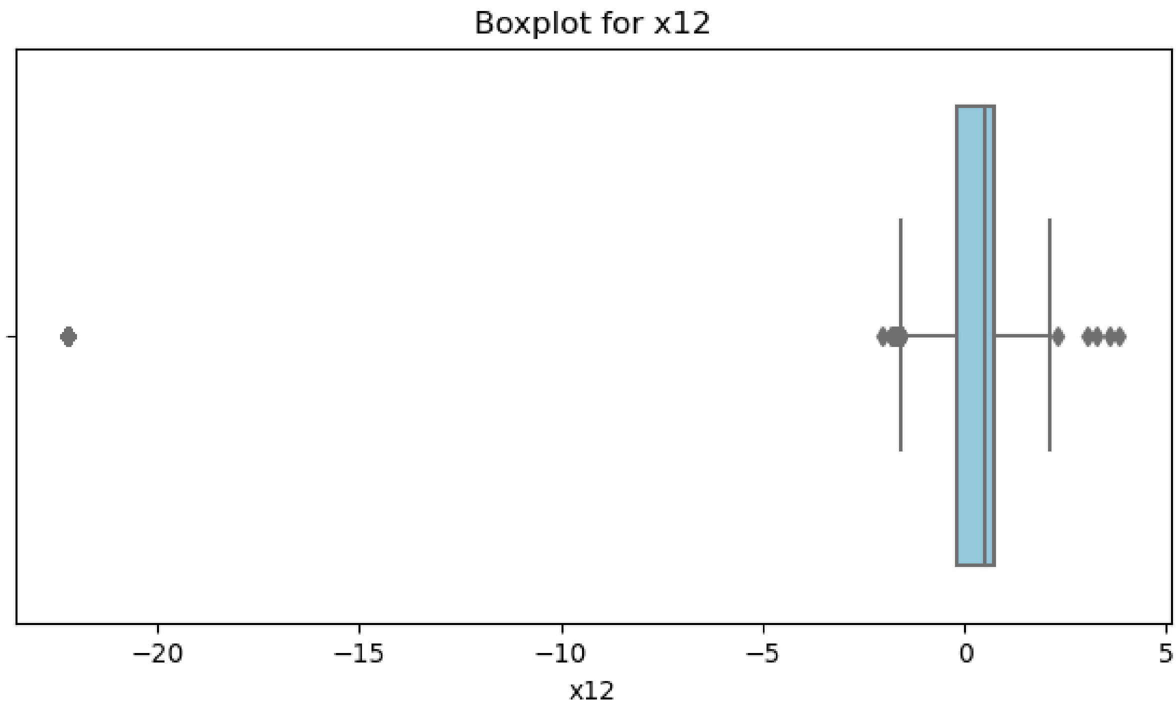

Boxplot for x7
−0.5
0.0
0.5
1.0
1.5
x7


Boxplot for x8
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
x8

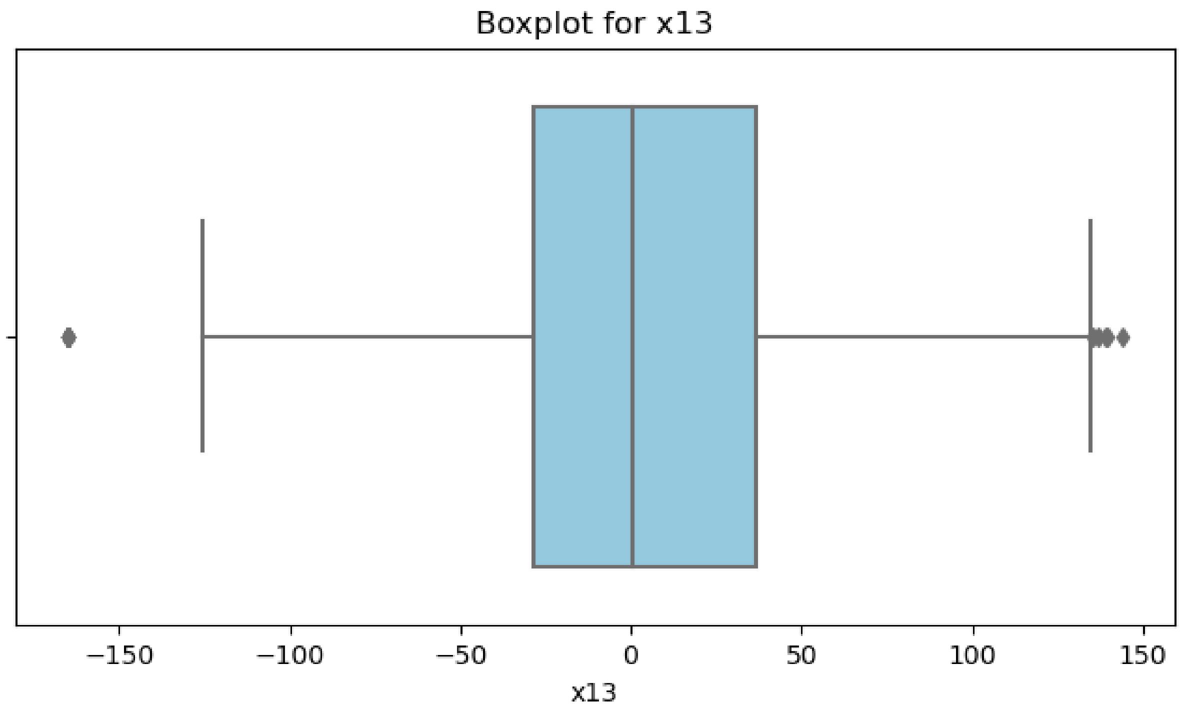

Boxplot for x9
0
1
2
3
4
x9


Boxplot for x10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x10

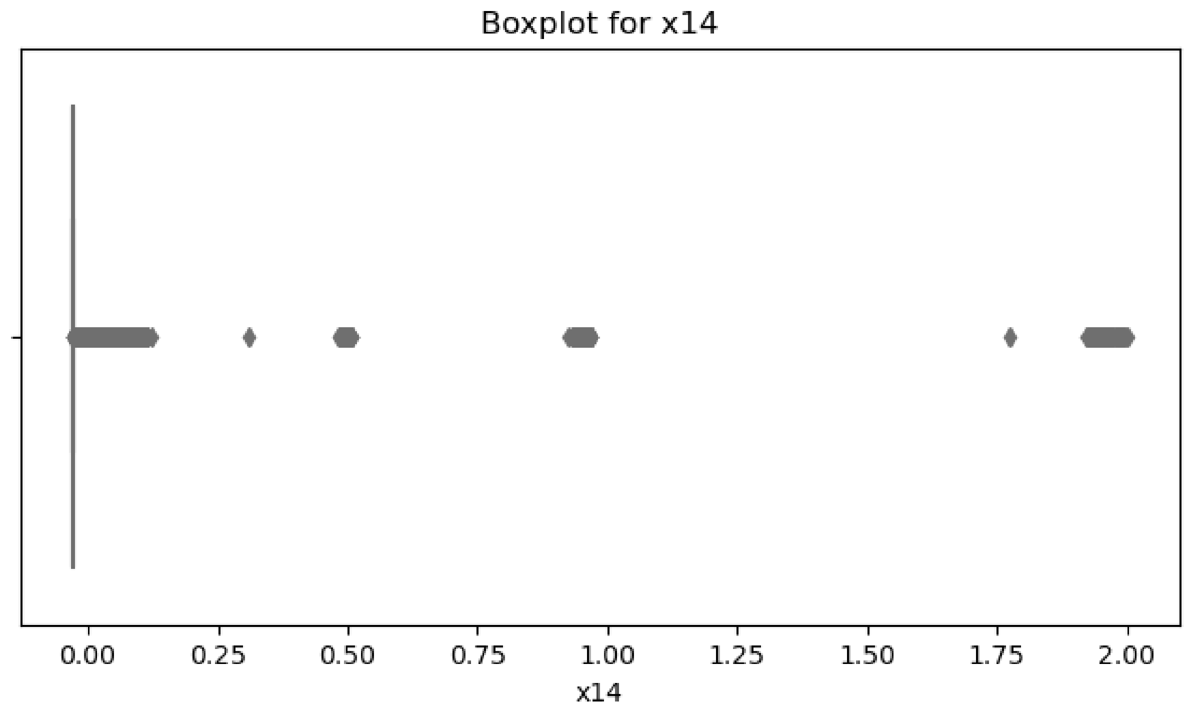

Boxplot for x11
−25
−20
−15
−10
−5
0
5
x11

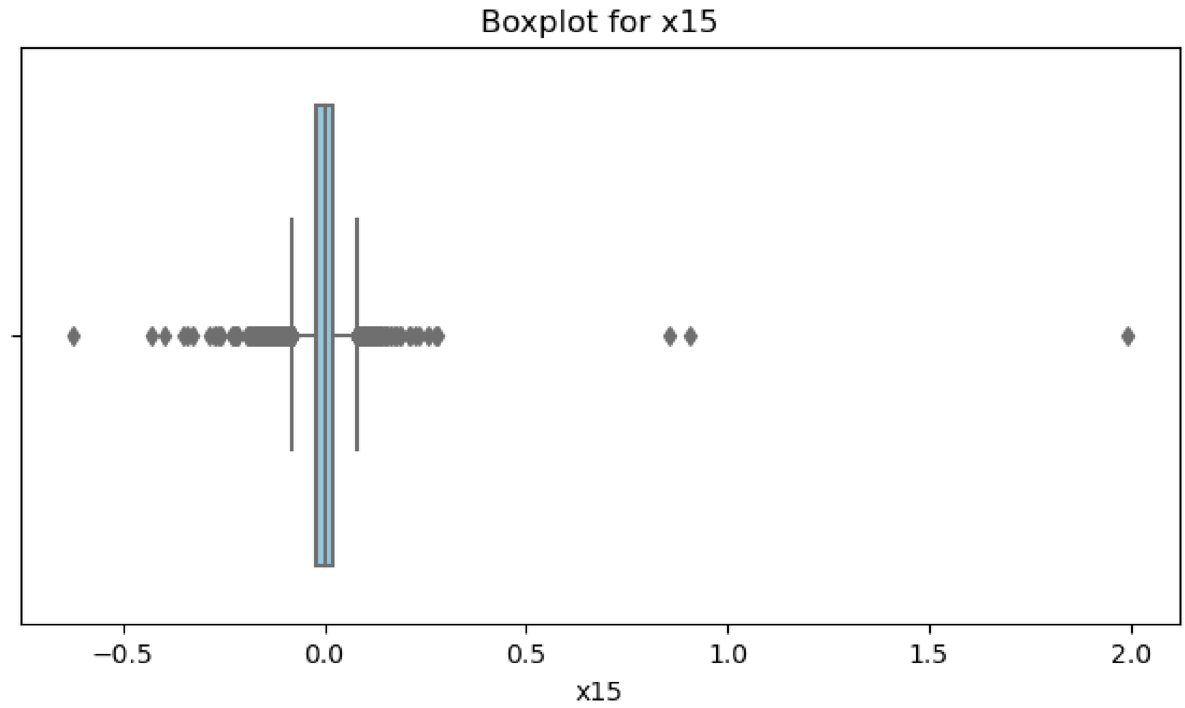


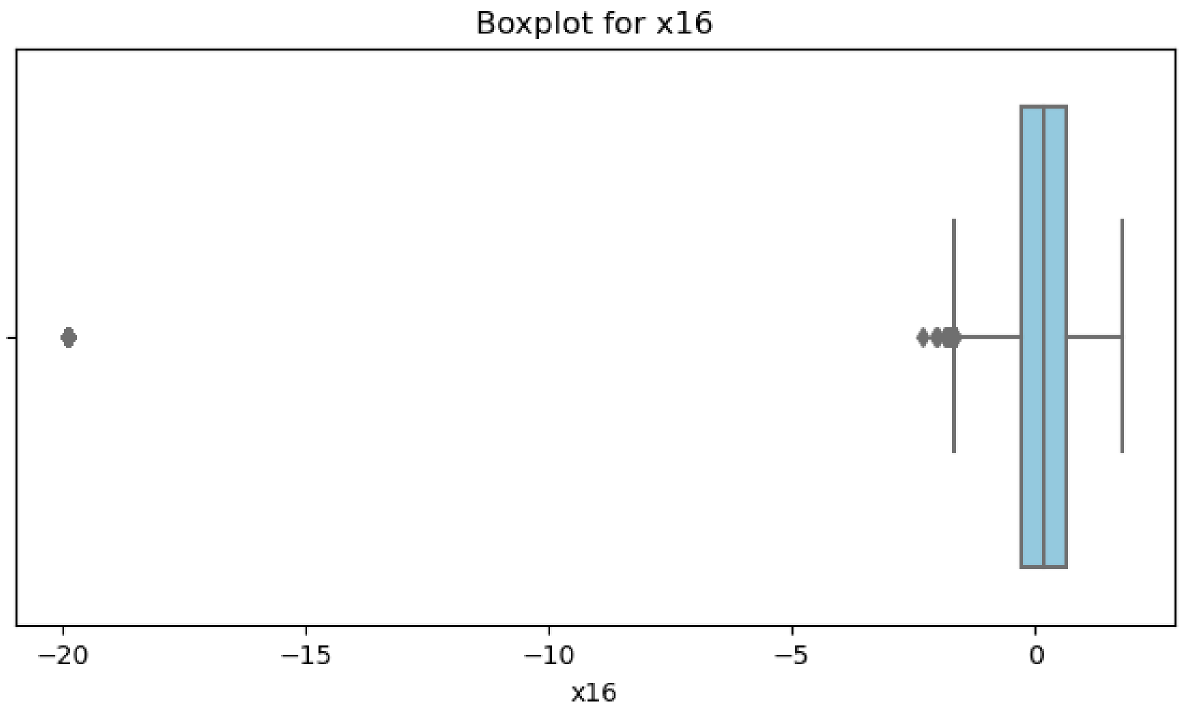
Boxplot for x12
−20
−15
−10
−5
0
5
x12

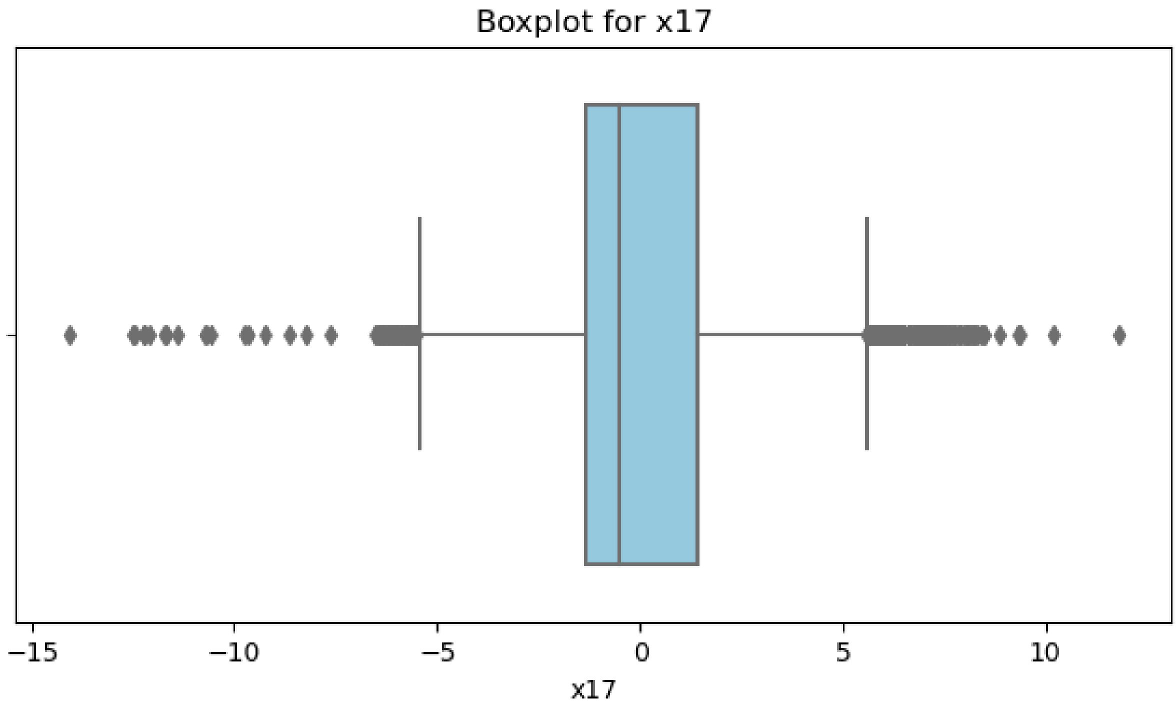


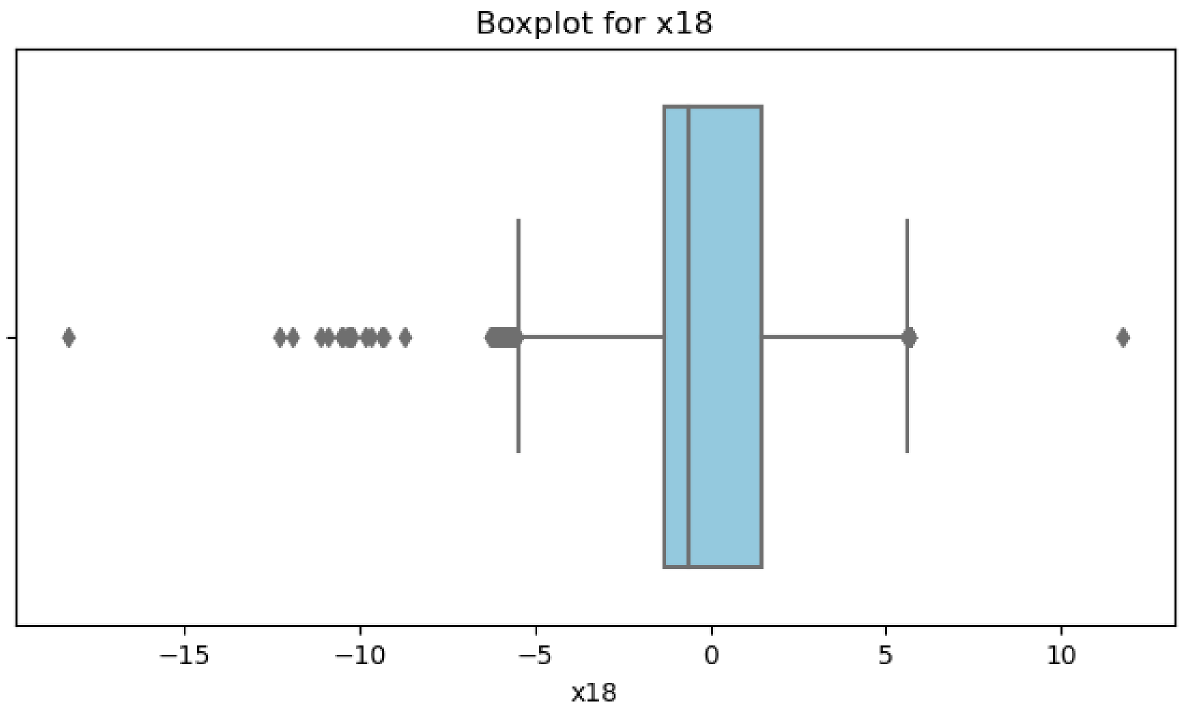
Boxplot for x13
−150
−100
−50
0
50
100
150
x13

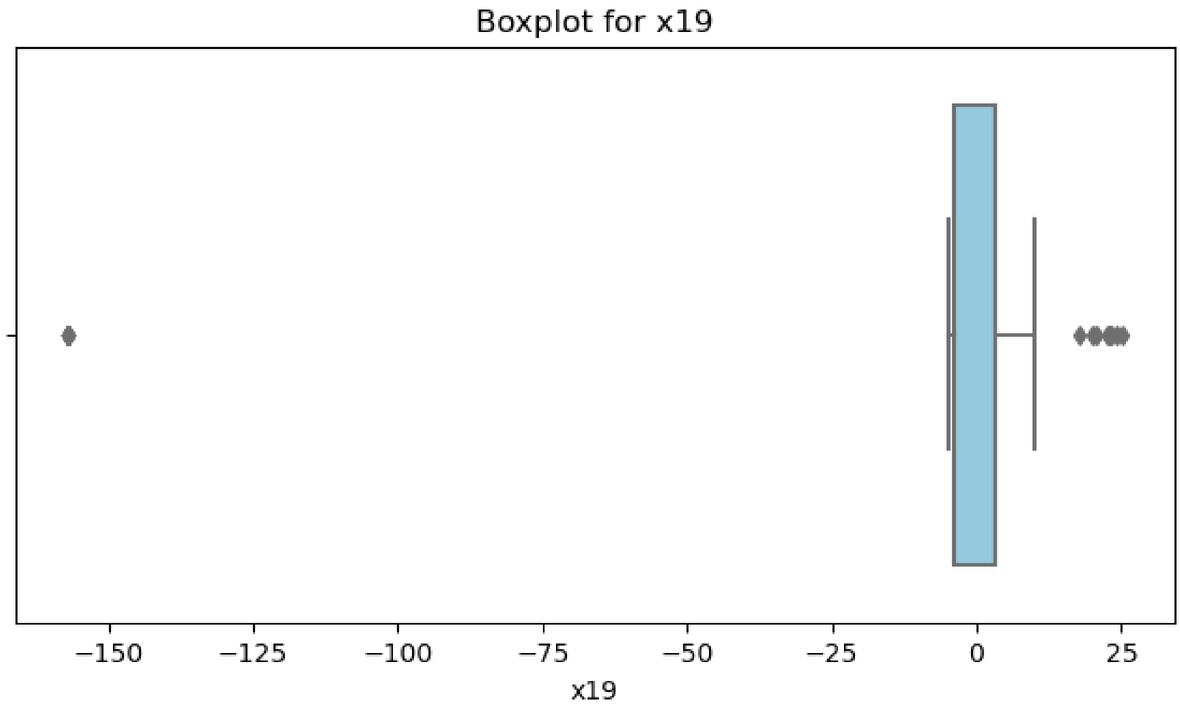

Boxplot for x14
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x14

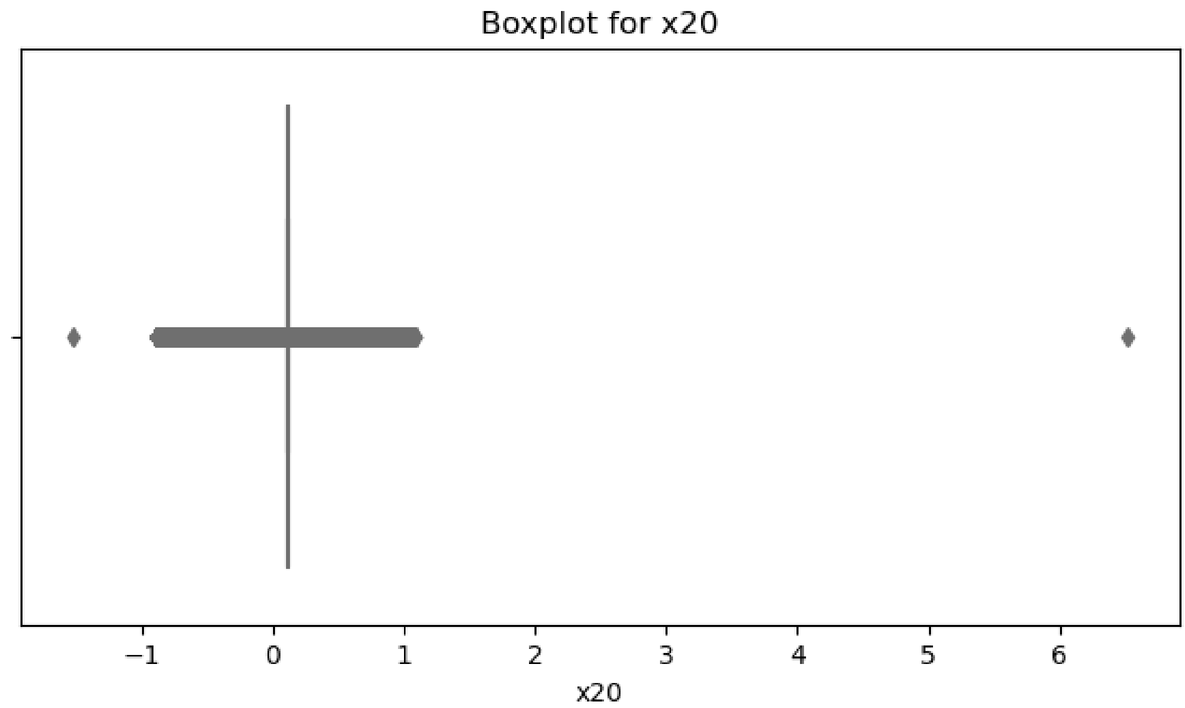

Boxplot for x15
−0.5
0.0
0.5
1.0
1.5
2.0
x15

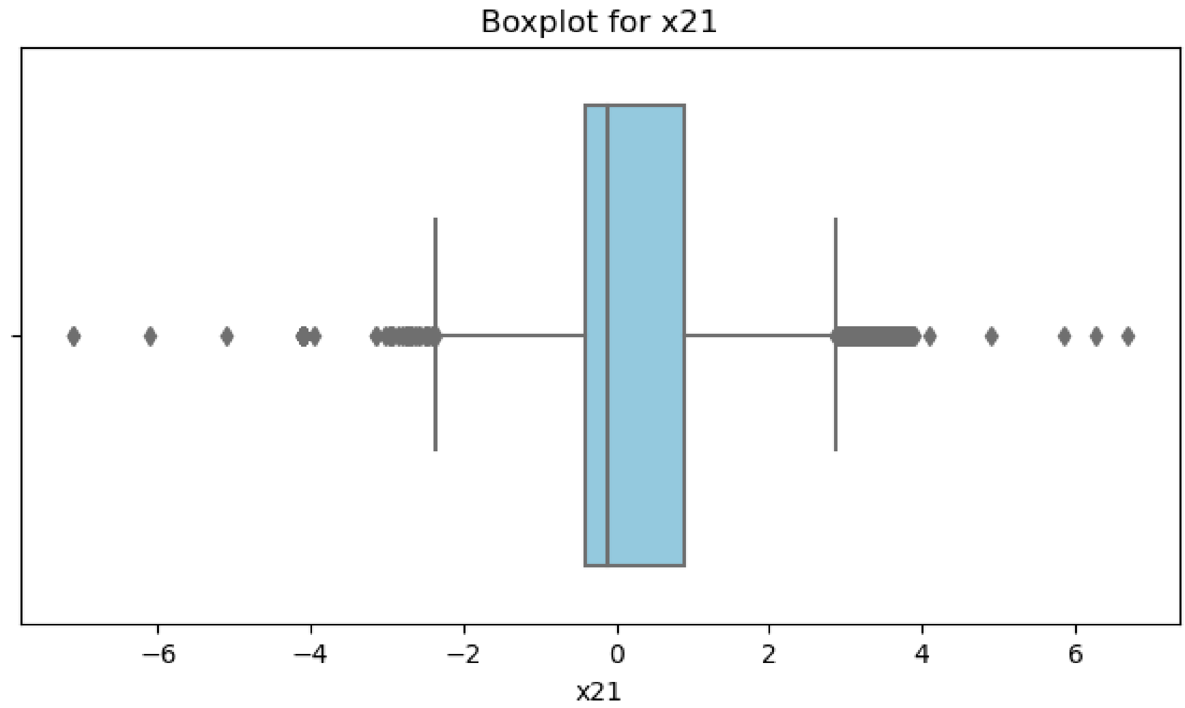

Boxplot for x16
−20
−15
−10
−5
0
x16

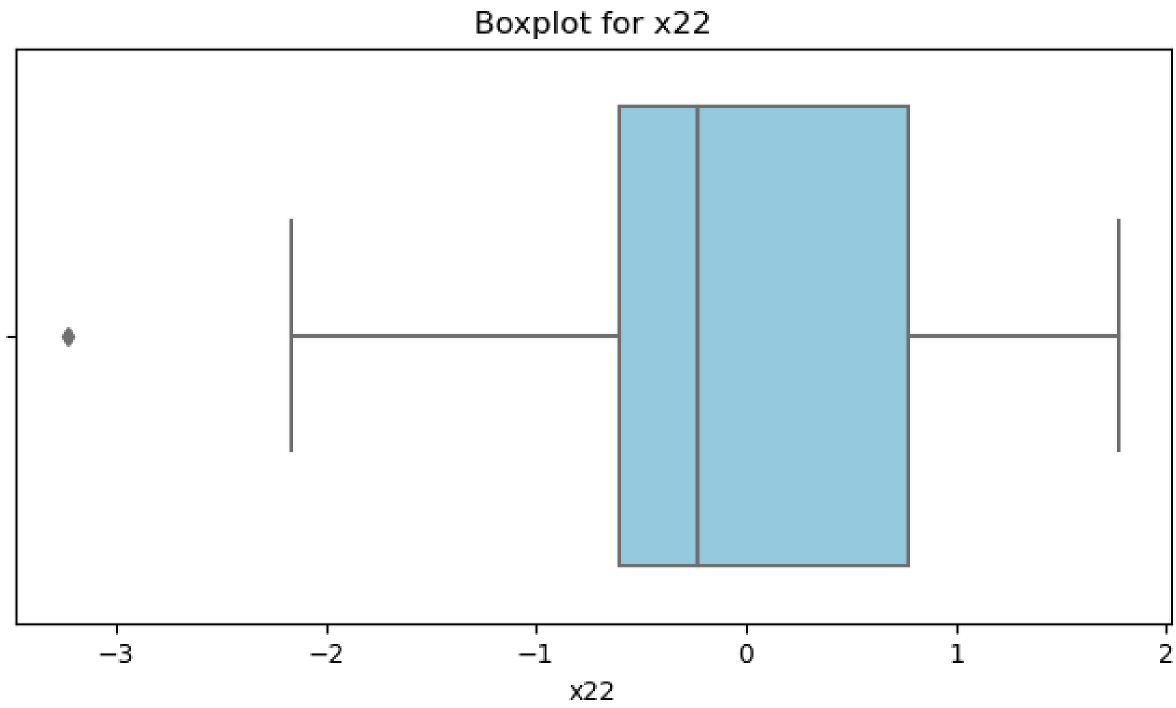

Boxplot for x17
−15
−10
−5
0
5
10
x17

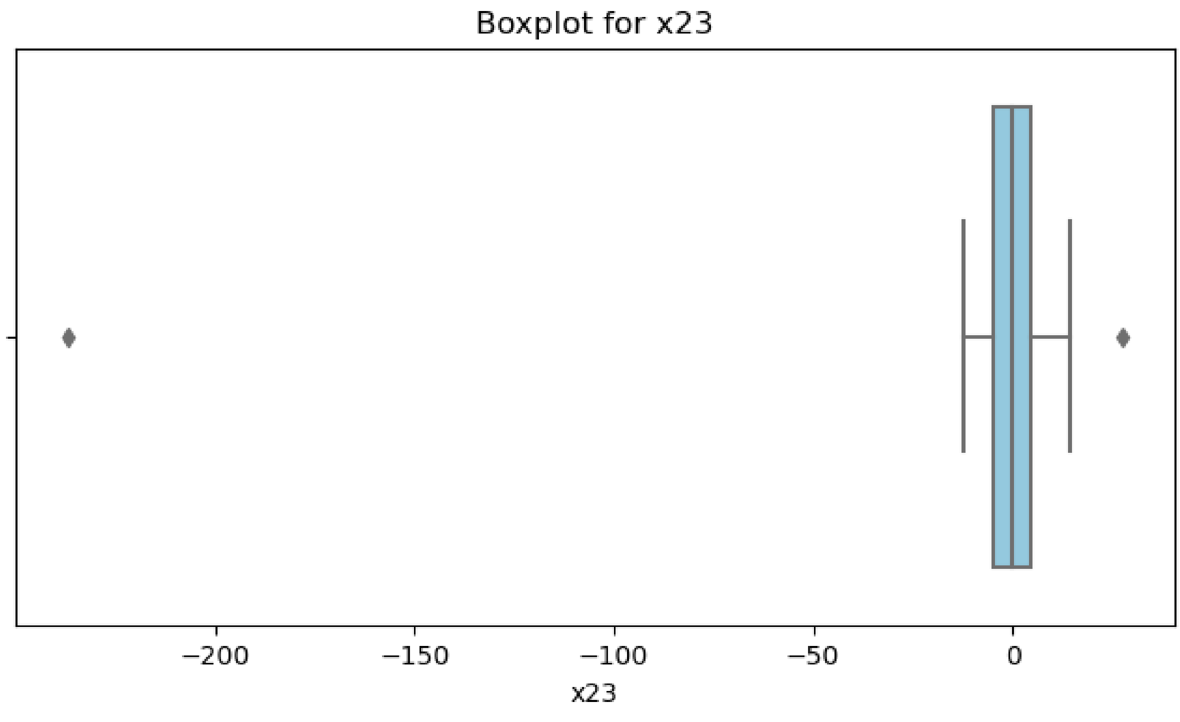

Boxplot for x18
−15
−10
−5
0
5
10
x18

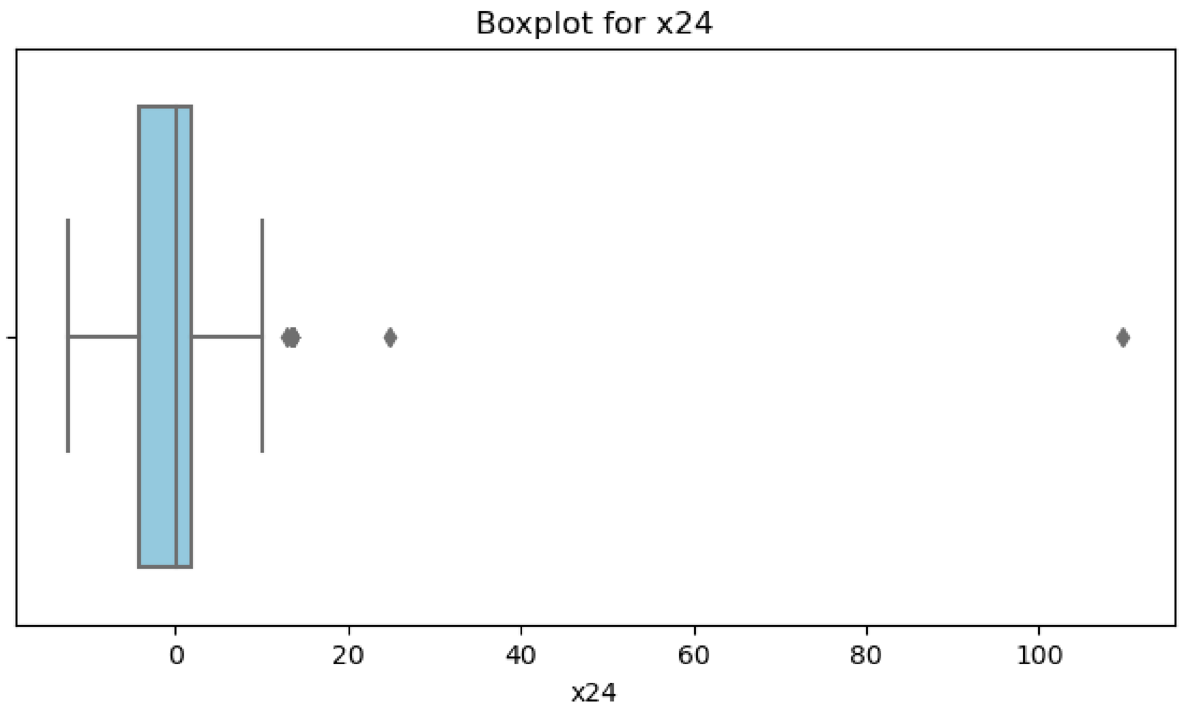

Boxplot for x19
−150
−125
−100
−75
−50
−25
0
25
x19

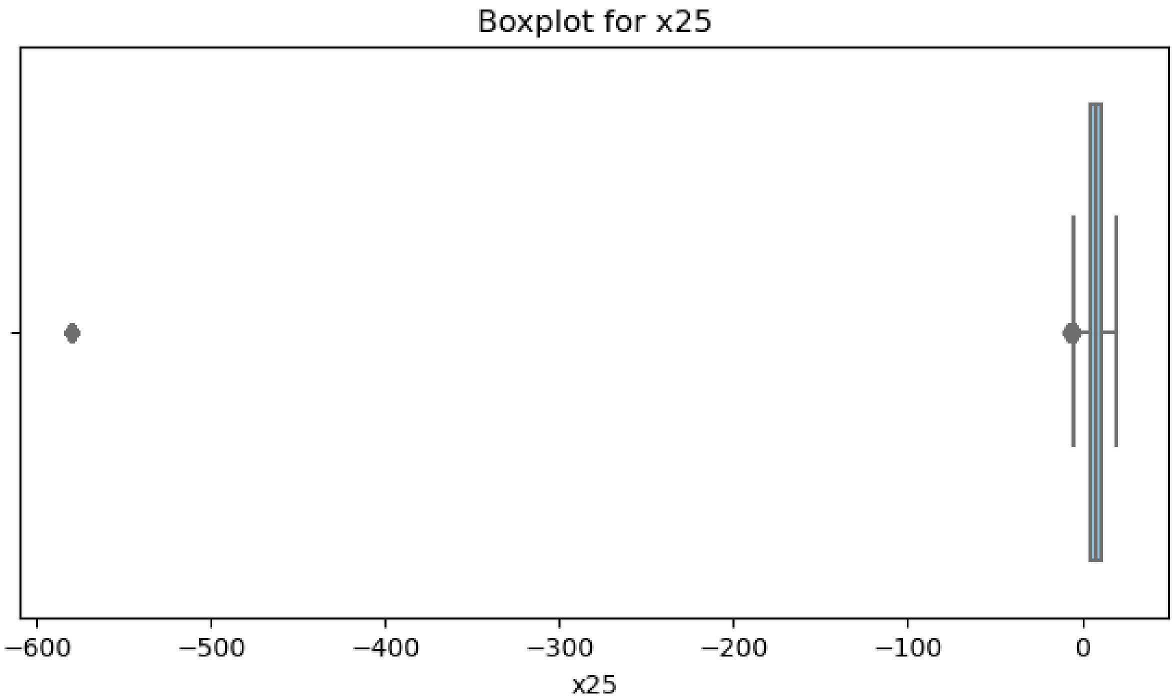

Boxplot for x20
−1
0
1
2
3
4
5
6
x20

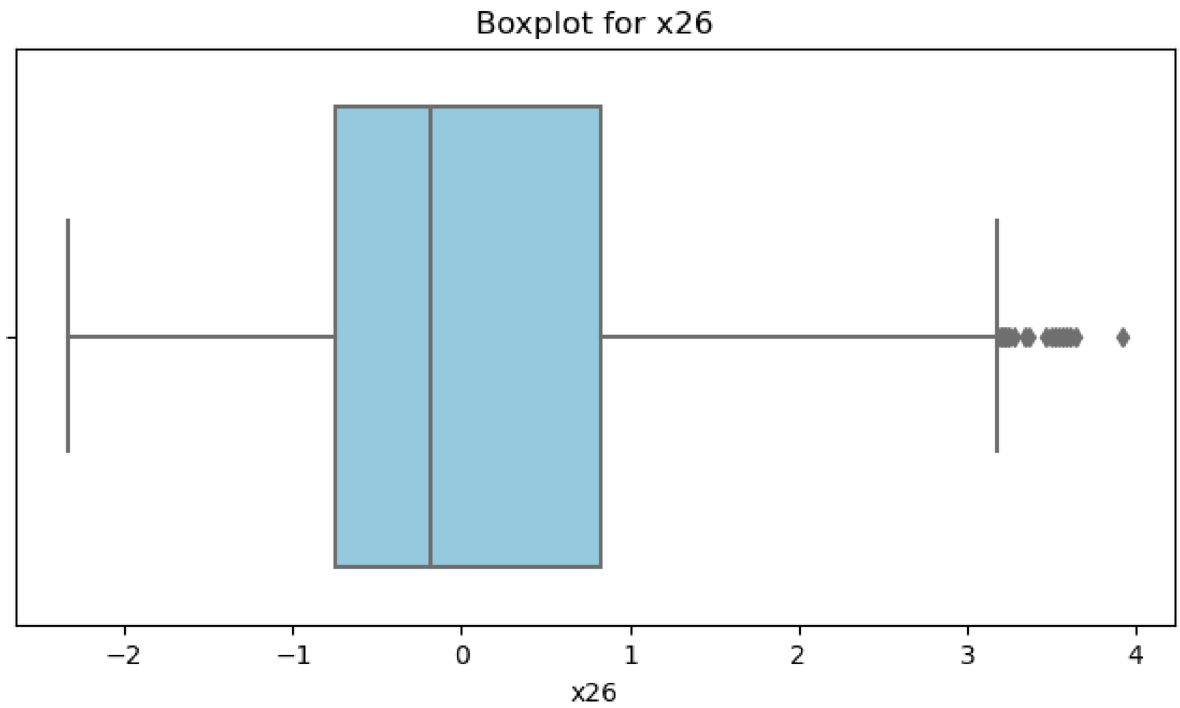

Boxplot for x21
−6
−4
−2
0
2
4
6
x21

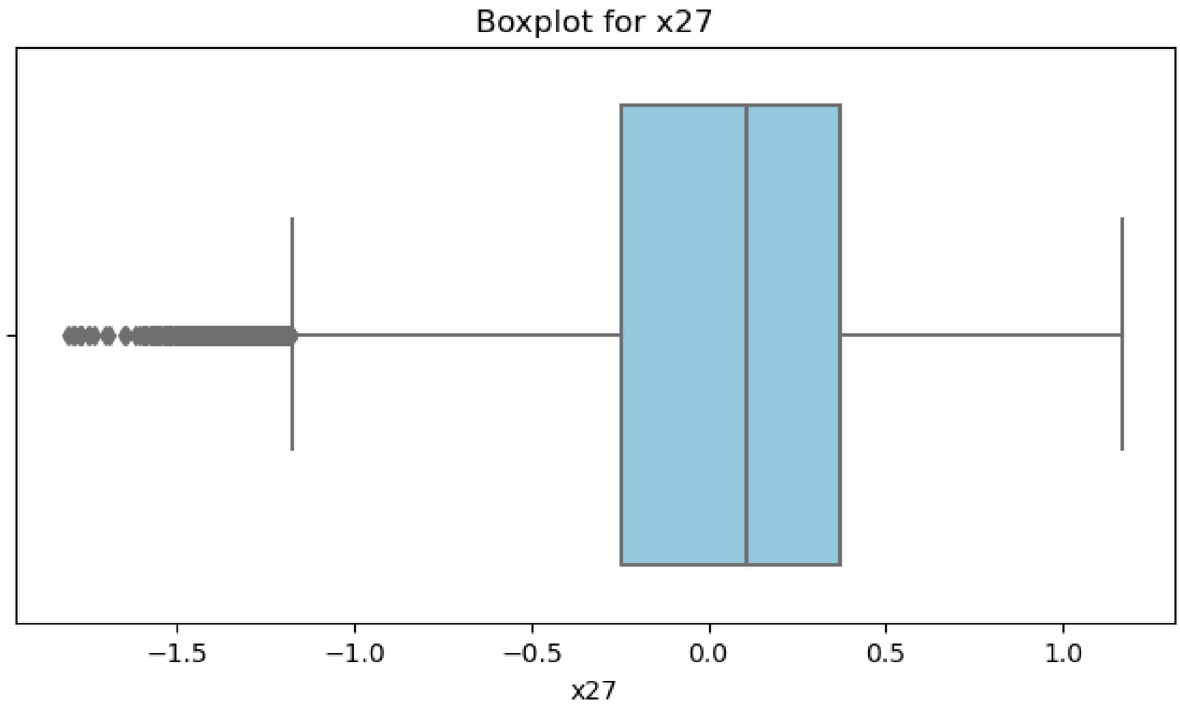

Boxplot for x22
−3
−2
−1
0
1
2
x22

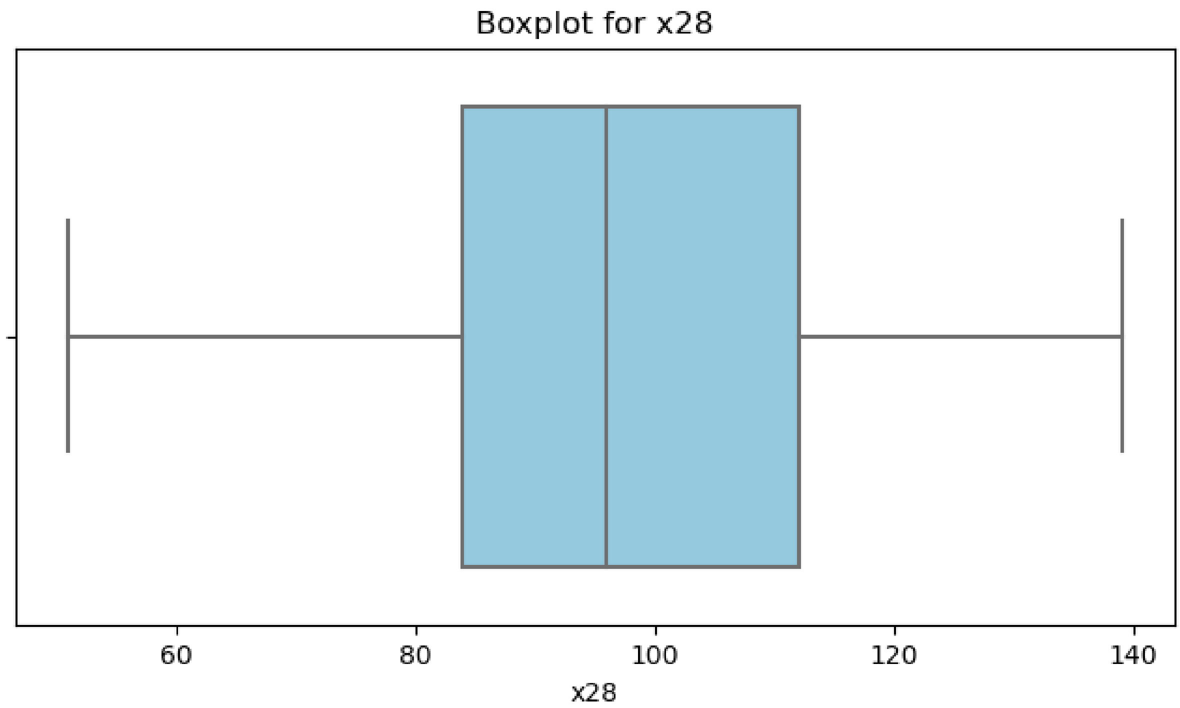

Boxplot for x23
−200
−150
−100
−50
0
x23


Boxplot for x24
0
20
40
60
80
100
x24

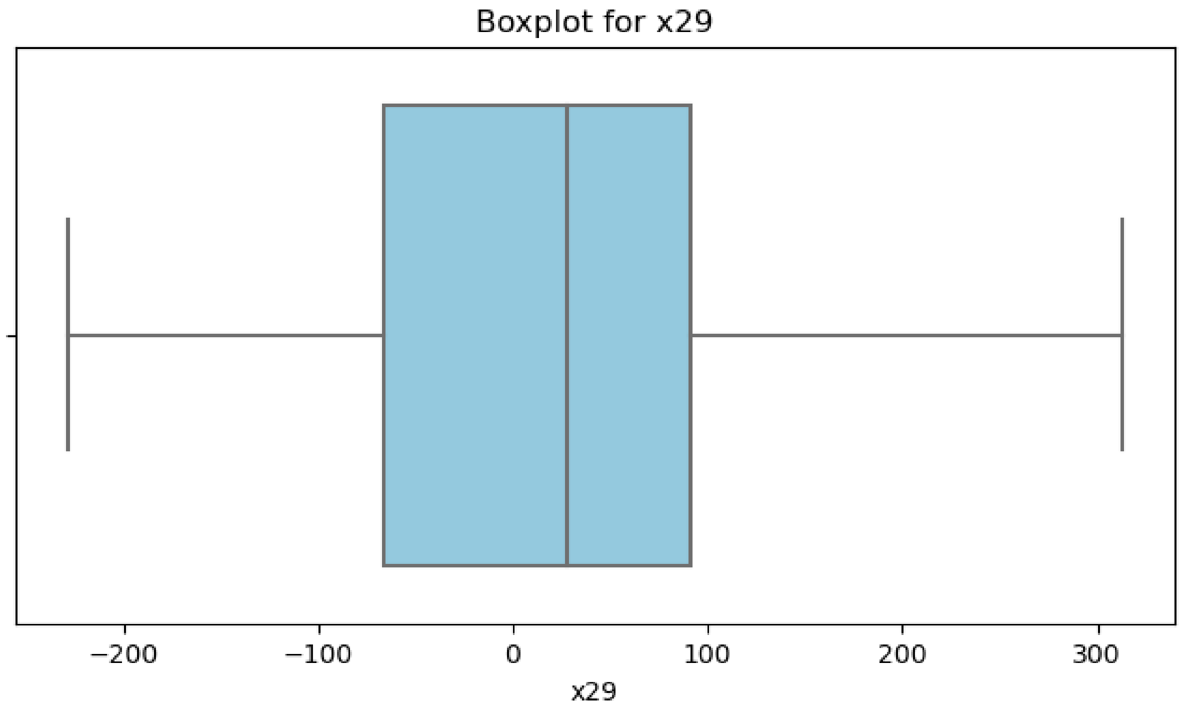

Boxplot for x25
−600
−500
−400
−300
−200
−100
0
x25


Boxplot for x26
−2
−1
0
1
2
3
4
x26

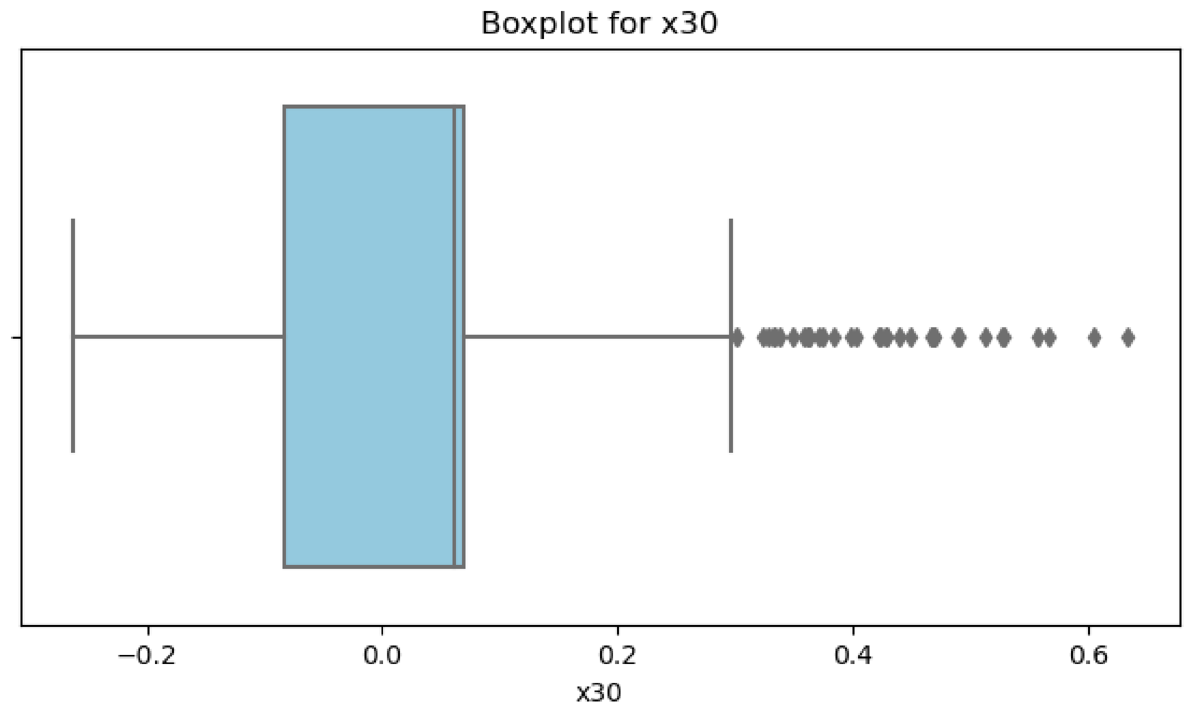

Boxplot for x27
−1.5
−1.0
−0.5
0.0
0.5
1.0
x27

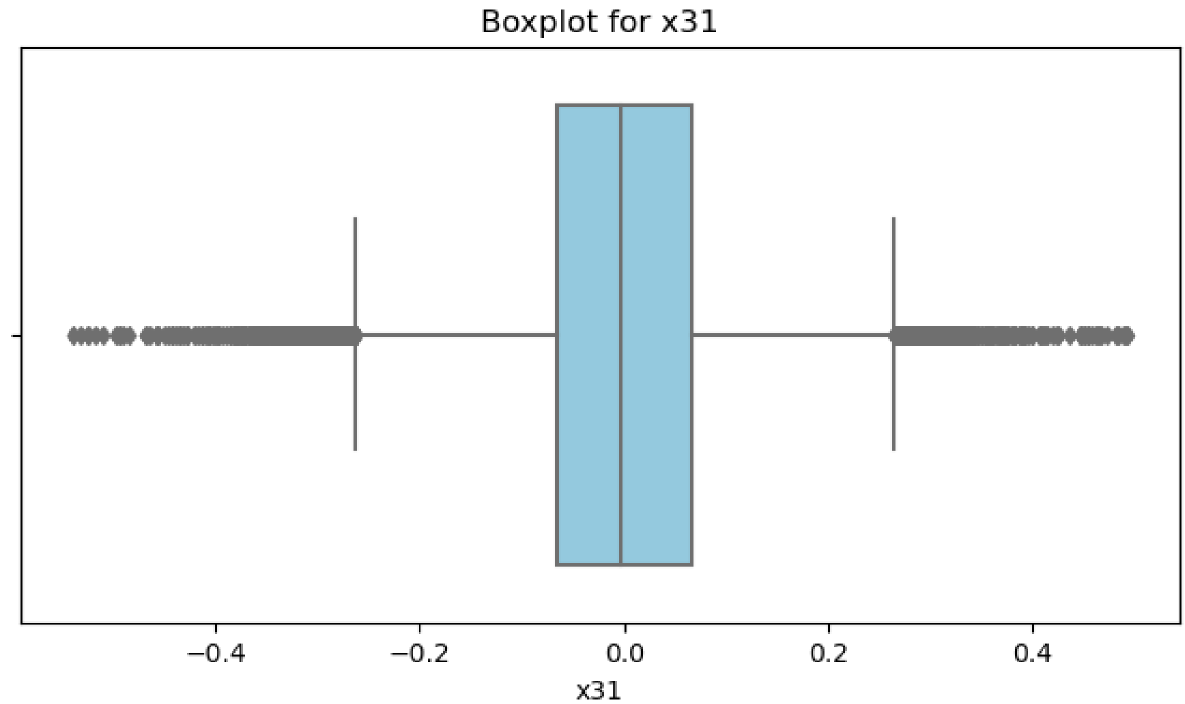


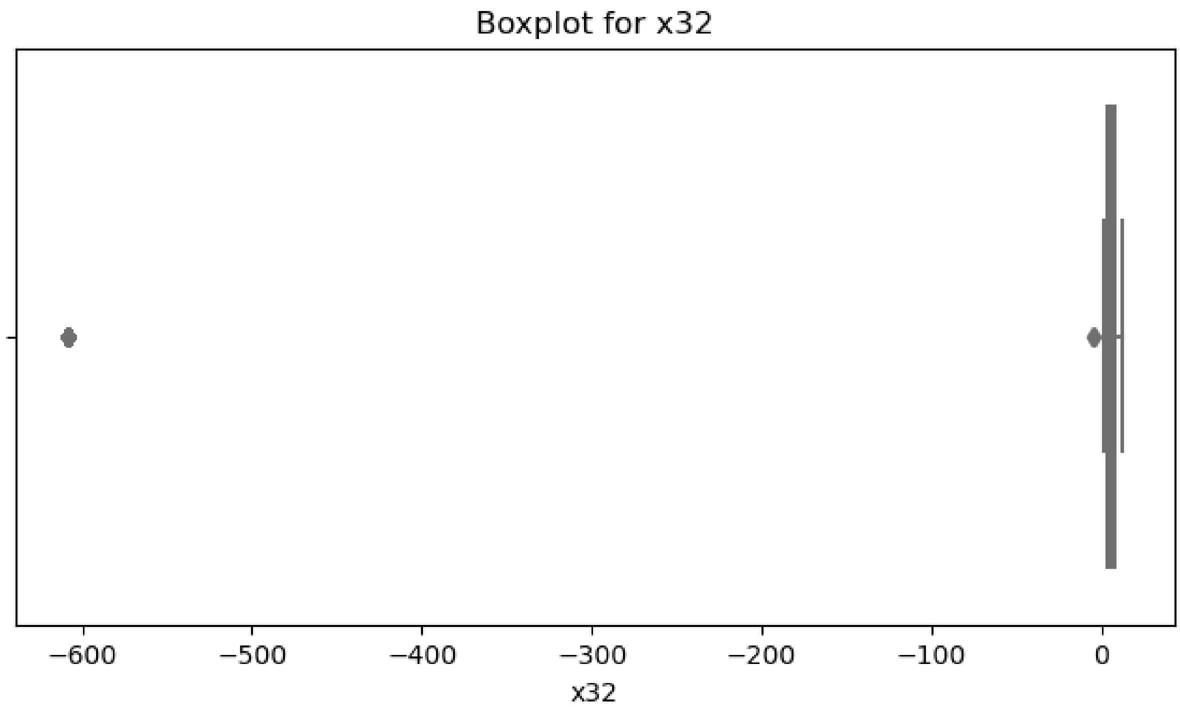
Boxplot for x28
60
80
100
120
140
x28

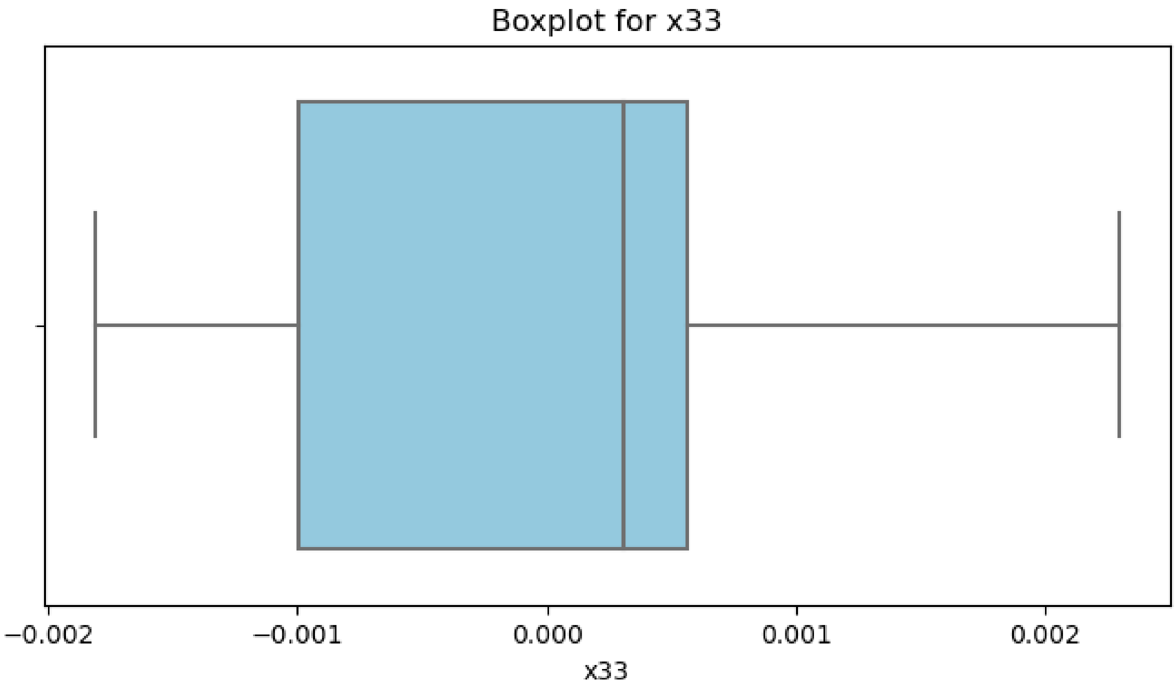


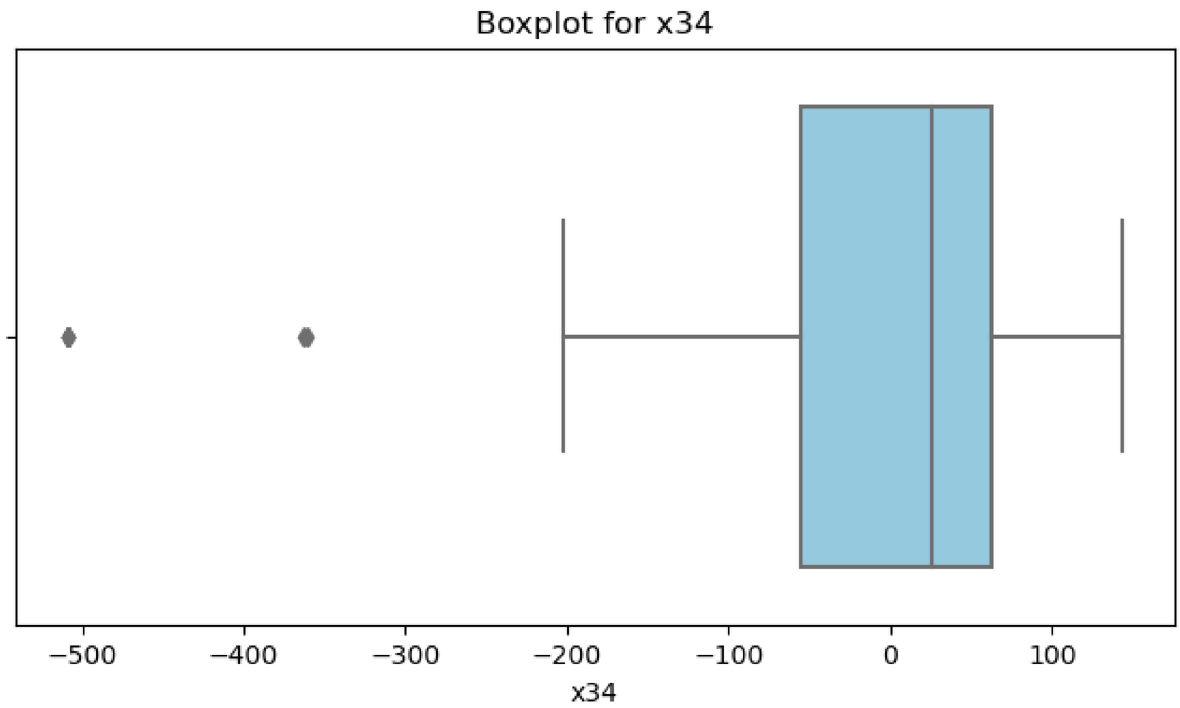
Boxplot for x29
−200
−100
0
100
200
300
x29

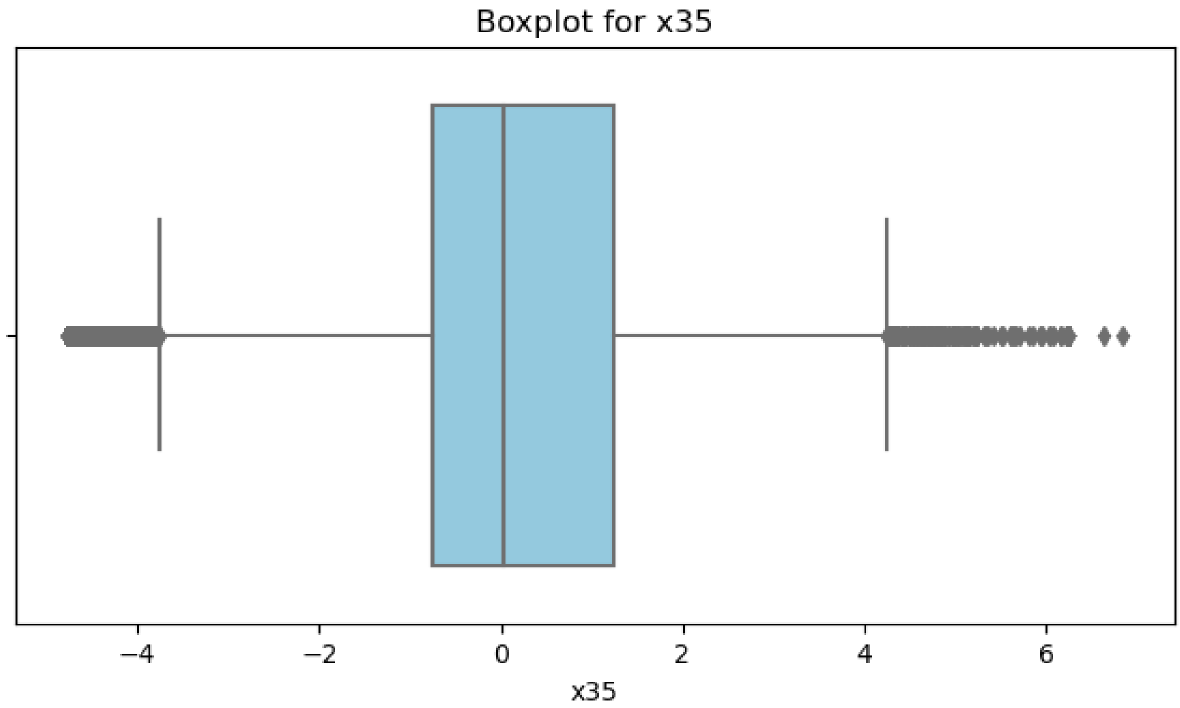

Boxplot for x30
−0.2
0.0
0.2
0.4
0.6
x30

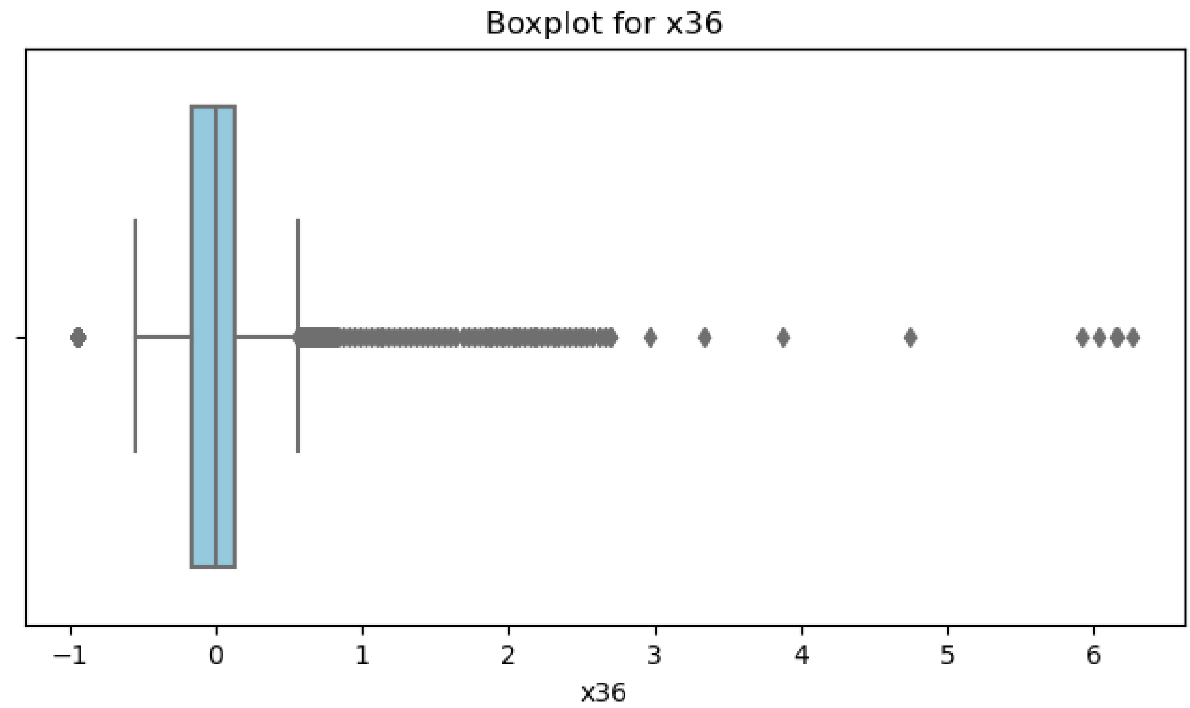

Boxplot for x31
−0.4
−0.2
0.0
0.2
0.4
x31

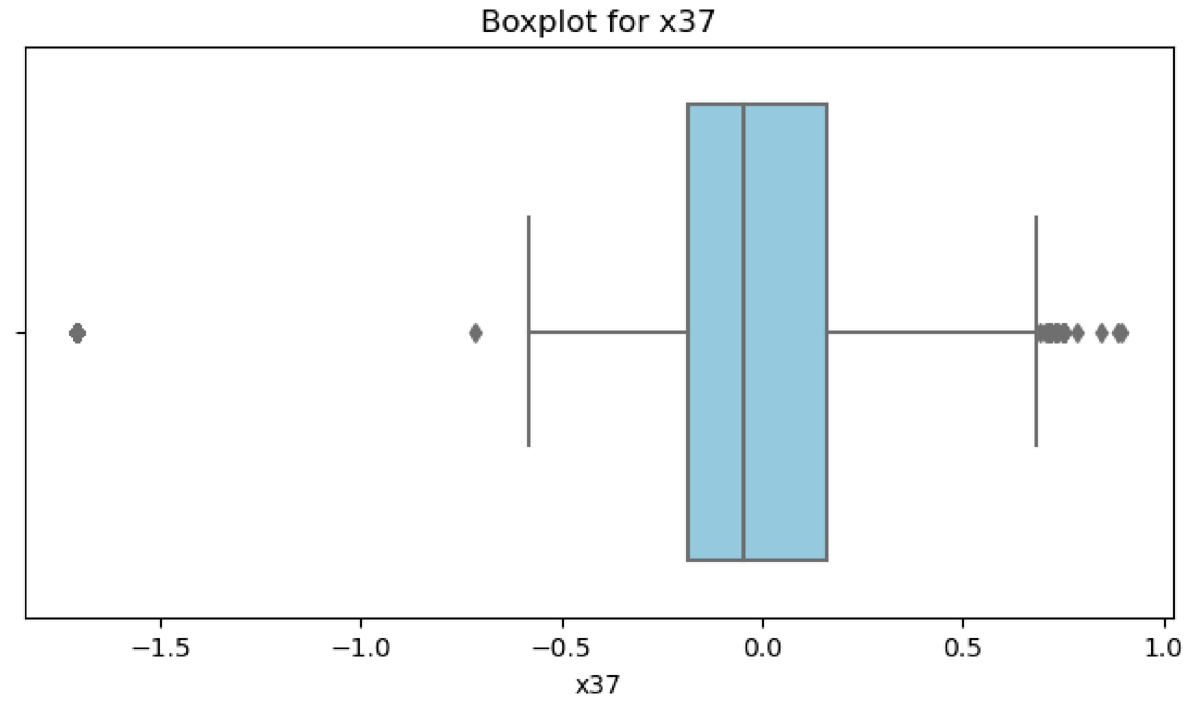

Boxplot for x32
−600
−500
−400
−300
−200
−100
0
x32

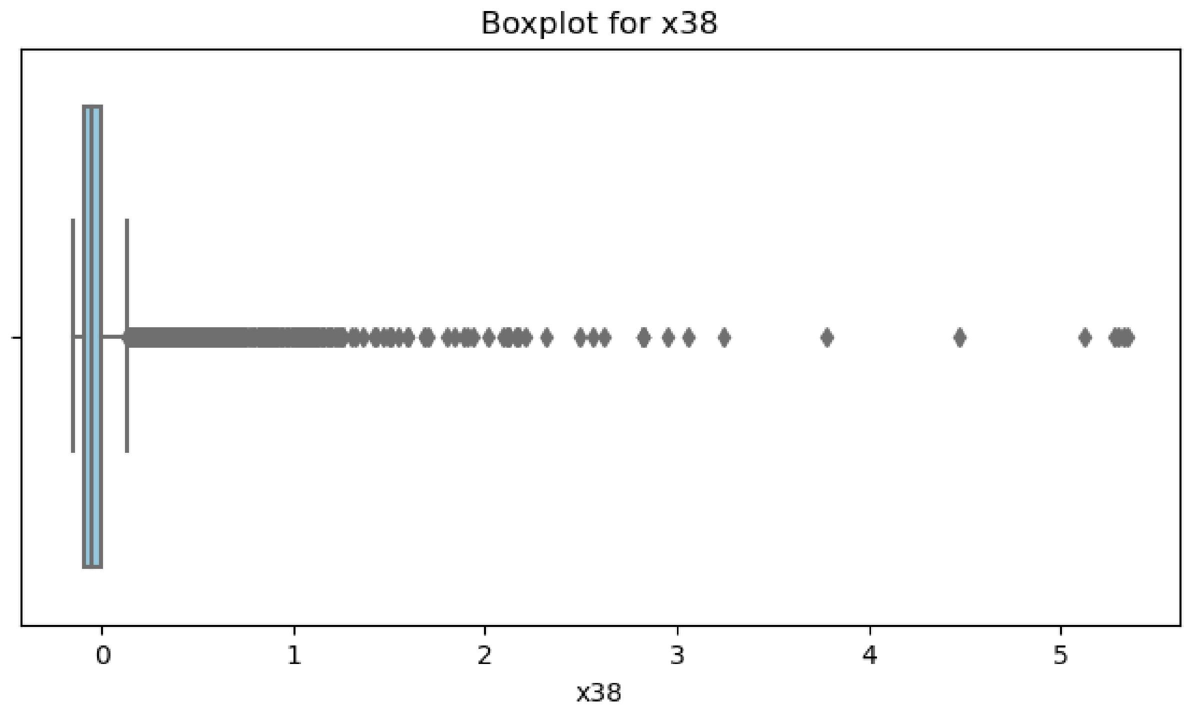

Boxplot for x33
−0.002
−0.001
0.000
0.001
0.002
x33


Boxplot for x34
−500
−400
−300
−200
−100
0
100
x34

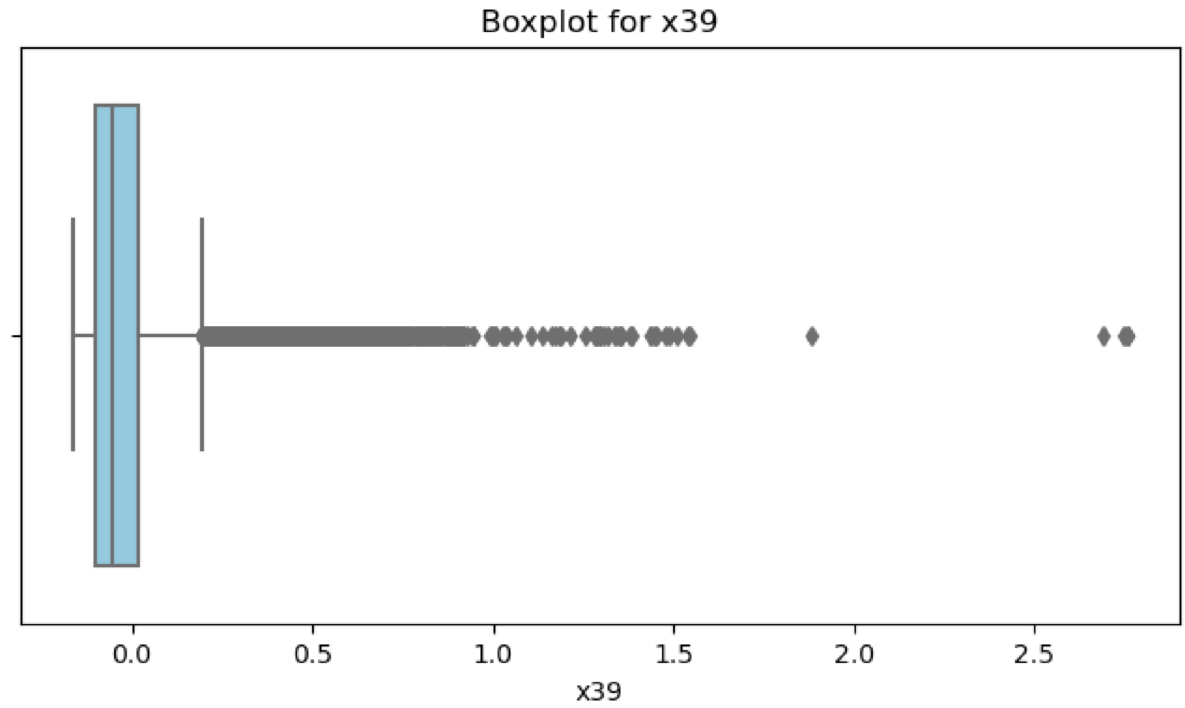

Boxplot for x35
−4
−2
0
2
4
6
x35


Boxplot for x36
−1
0
1
2
3
4
5
6
x36

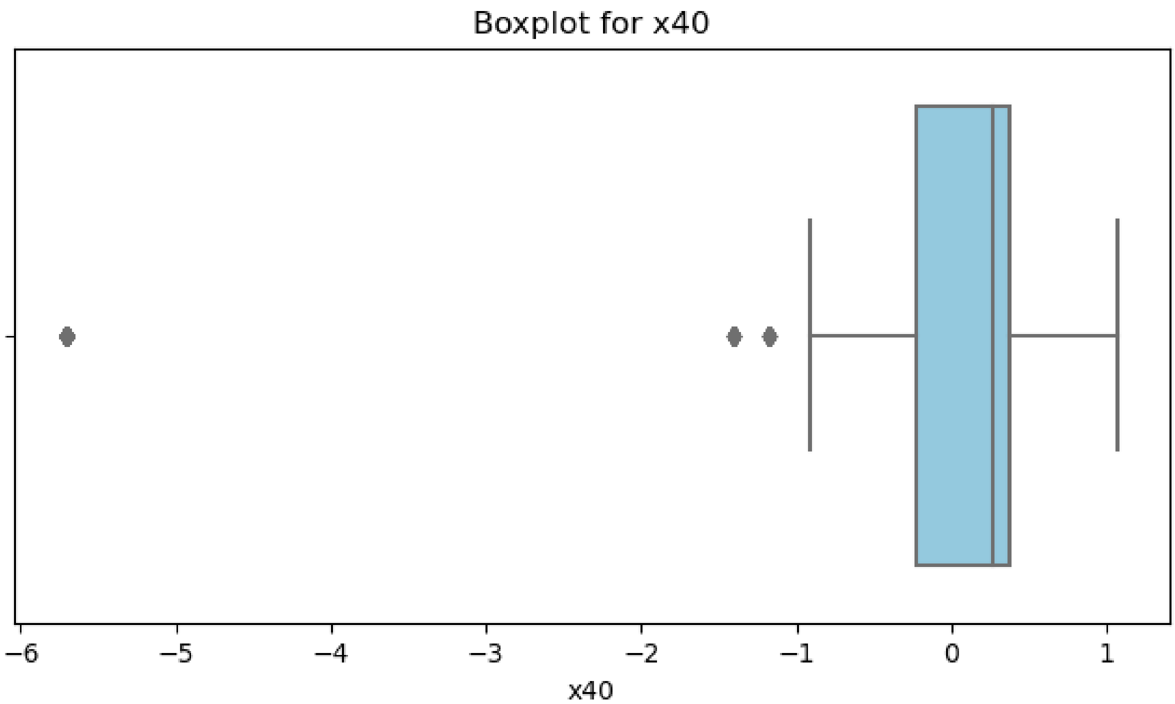

Boxplot for x37
−1.5
−1.0
−0.5
0.0
0.5
1.0
x37

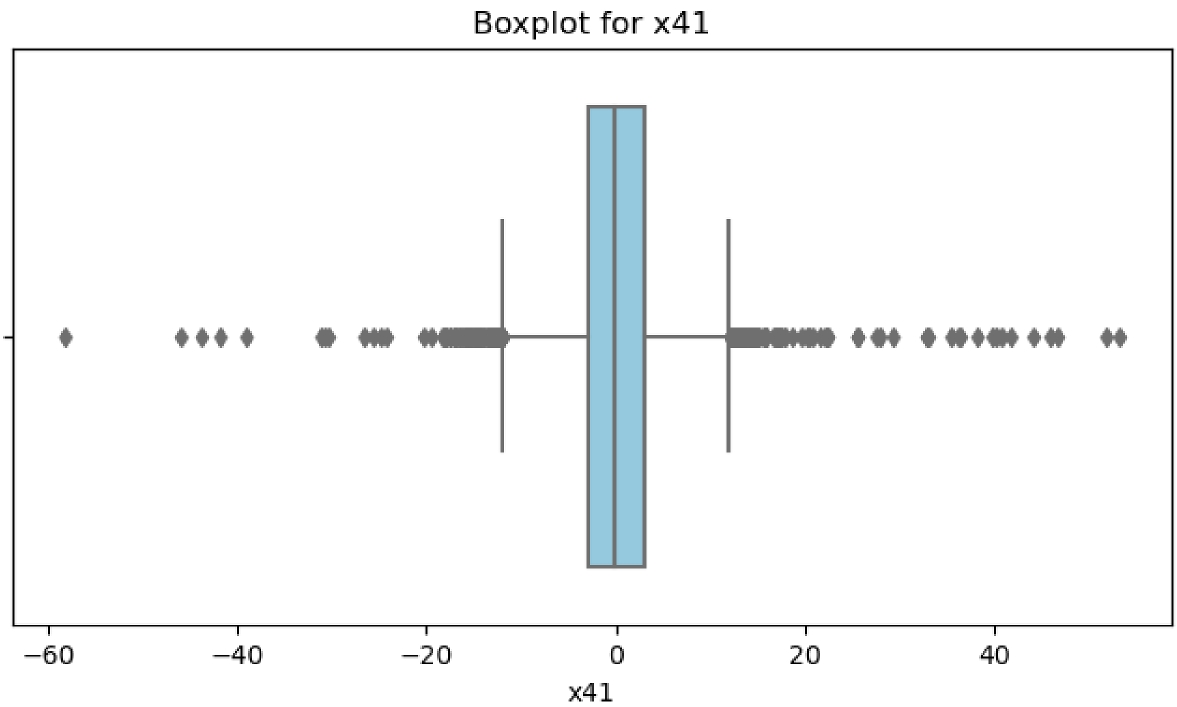


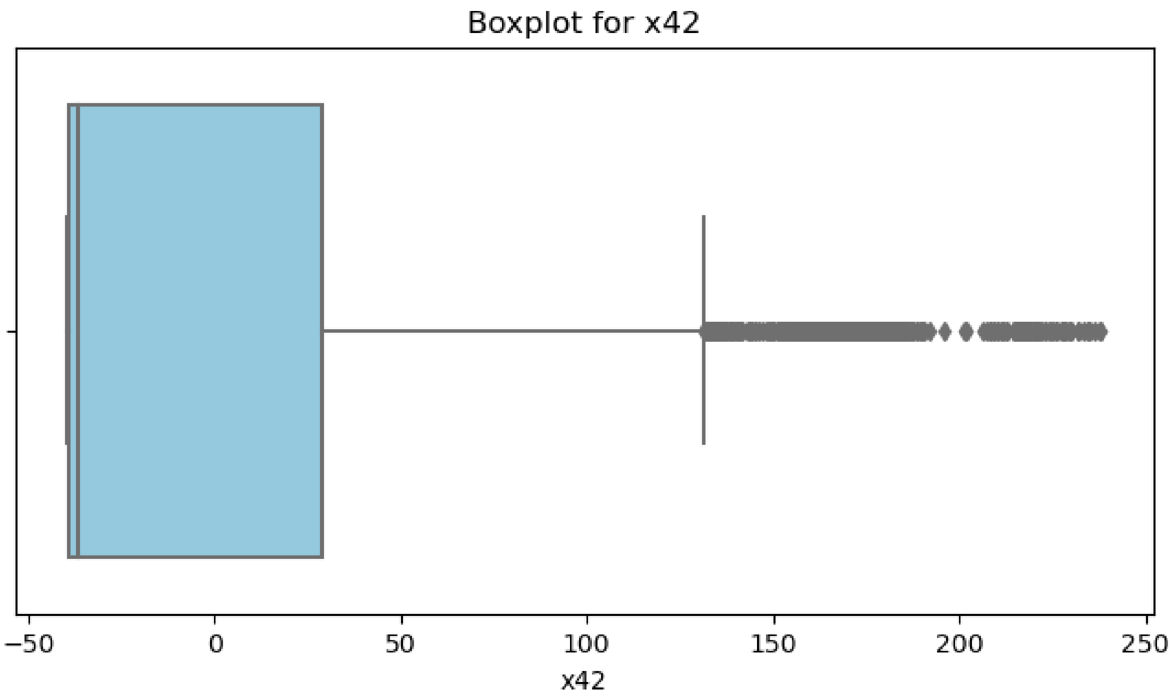
Boxplot for x38
0
1
2
3
4
5
x38

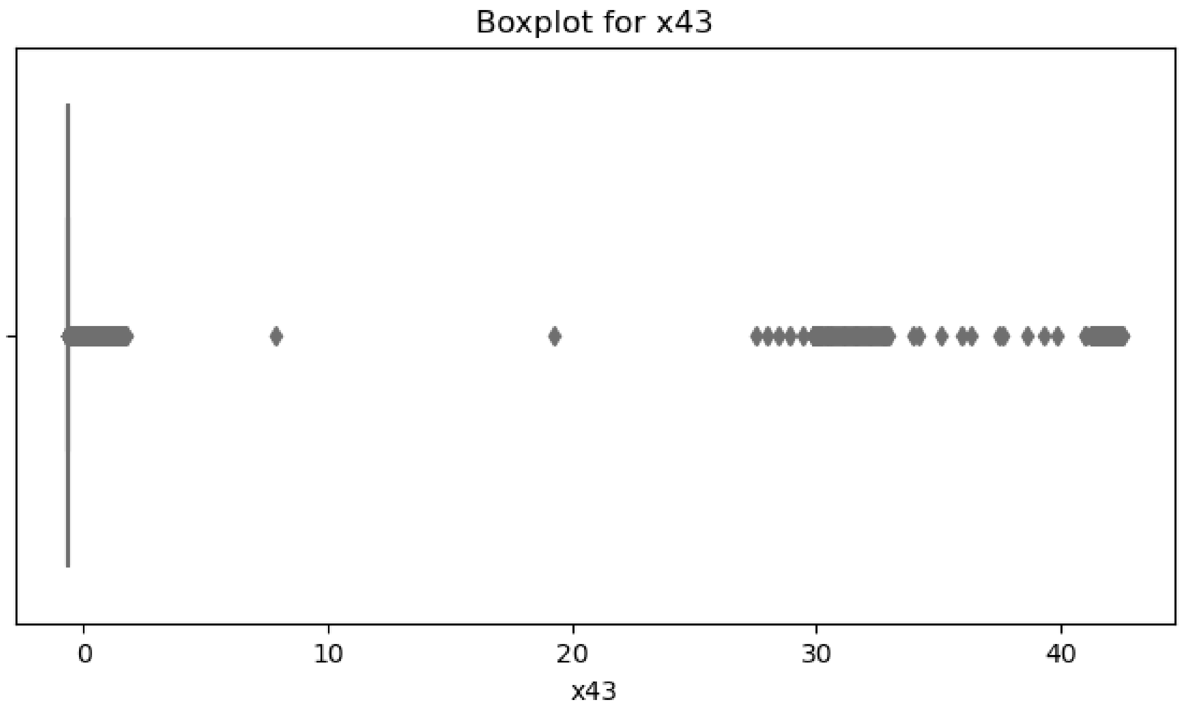


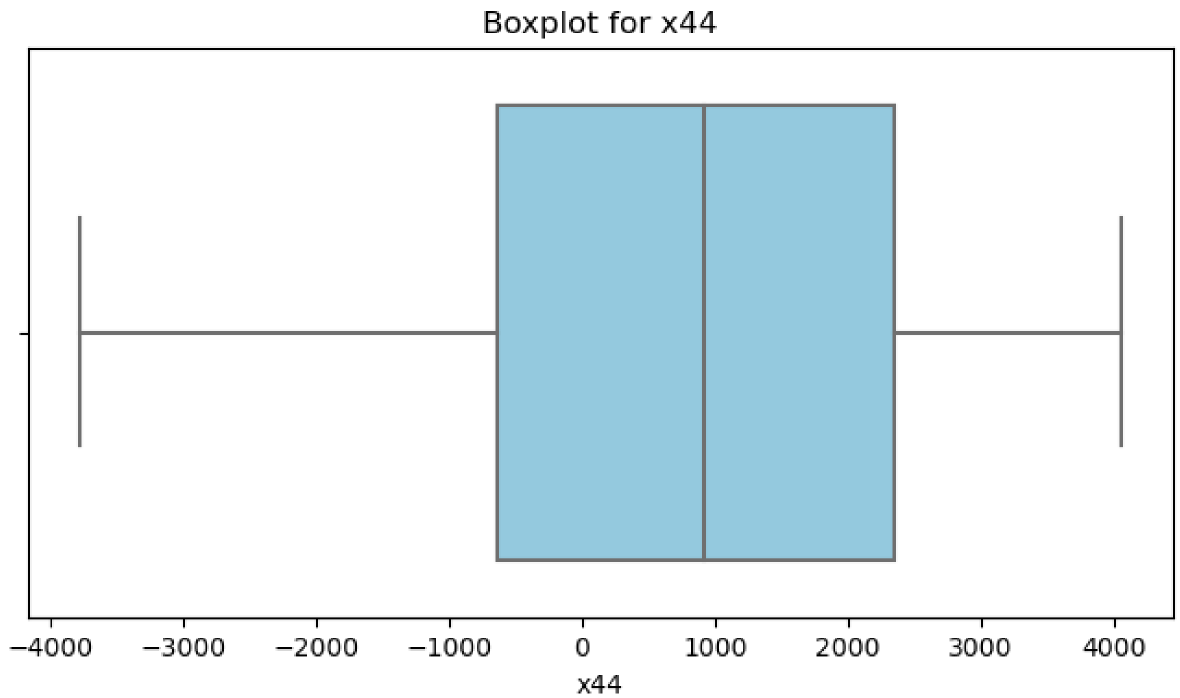
Boxplot for x39
0.0
0.5
1.0
1.5
2.0
2.5
x39


Boxplot for x40
−6
−5
−4
−3
−2
−1
0
1
x40

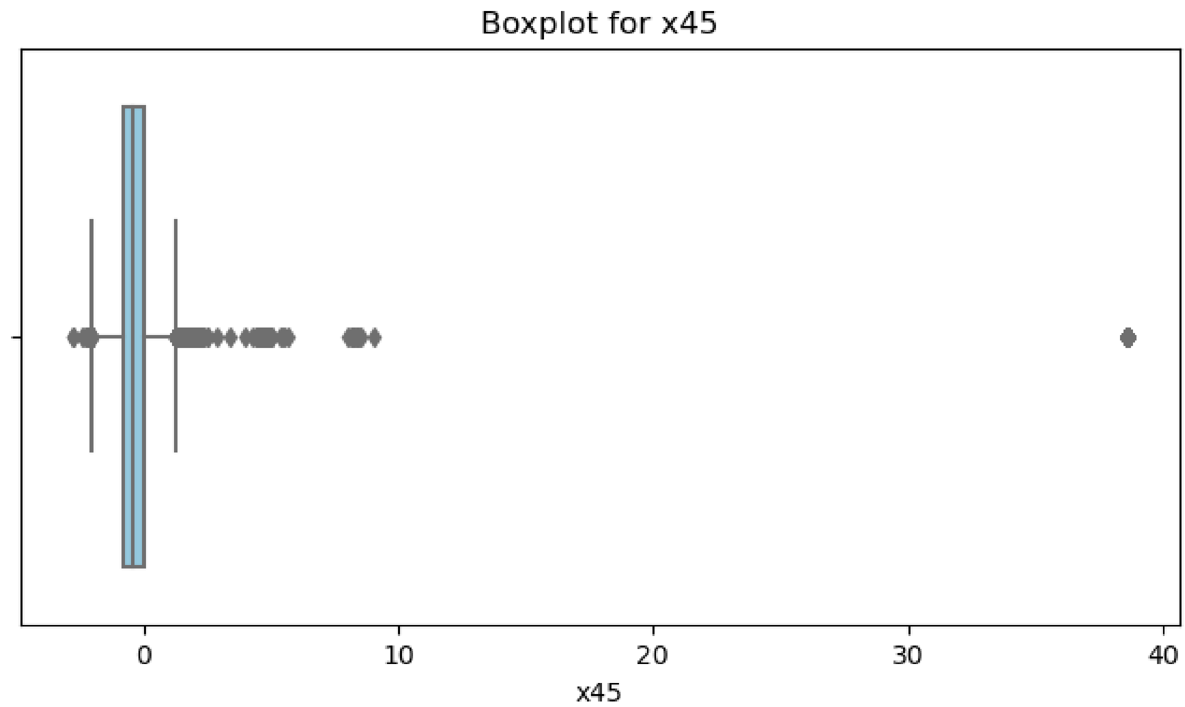

Boxplot for x41
−60
−40
−20
0
20
40
x41


Boxplot for x42
−50
0
50
100
150
200
250
x42

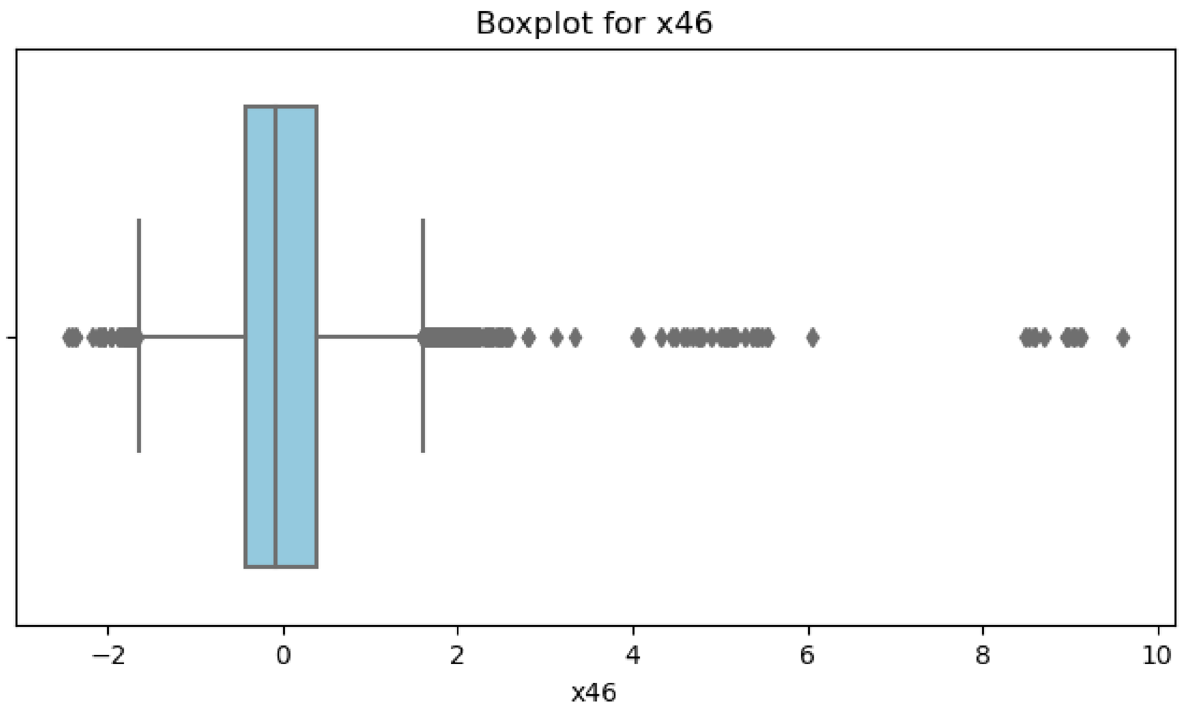

Boxplot for x43
0
10
20
30
40
x43

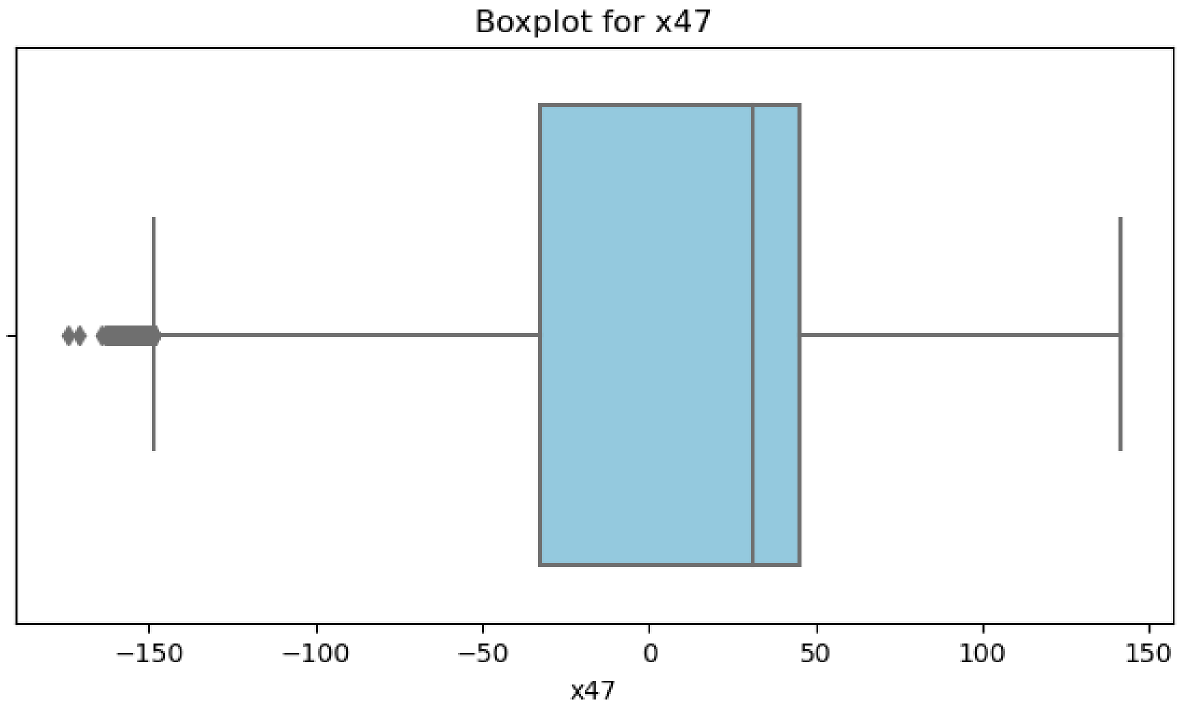


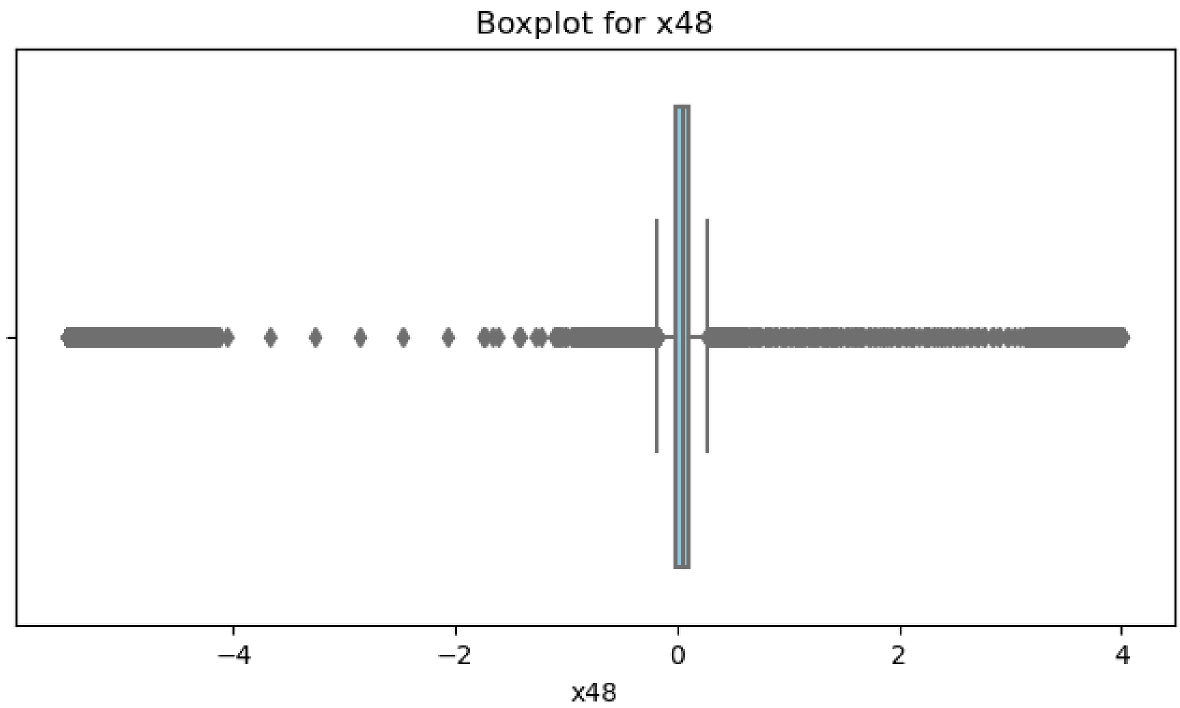
Boxplot for x44
−4000
−3000
−2000
−1000
0
1000
2000
3000
4000
x44

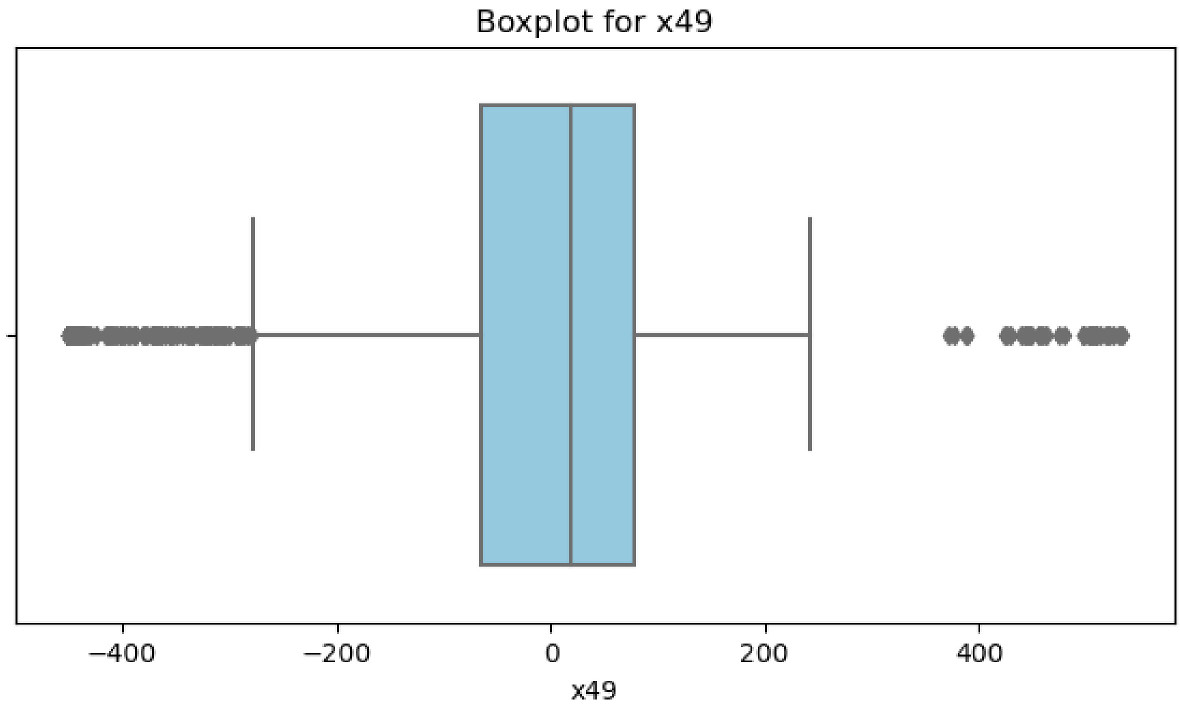


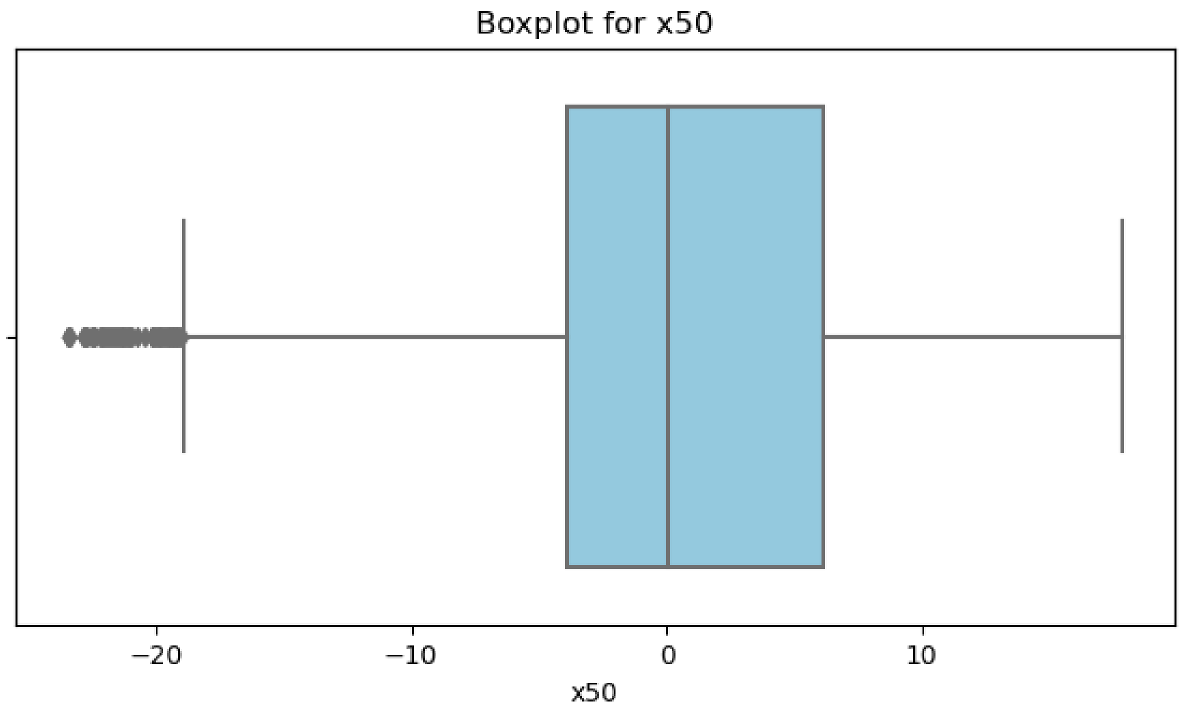
Boxplot for x45
0
10
20
30
40
x45

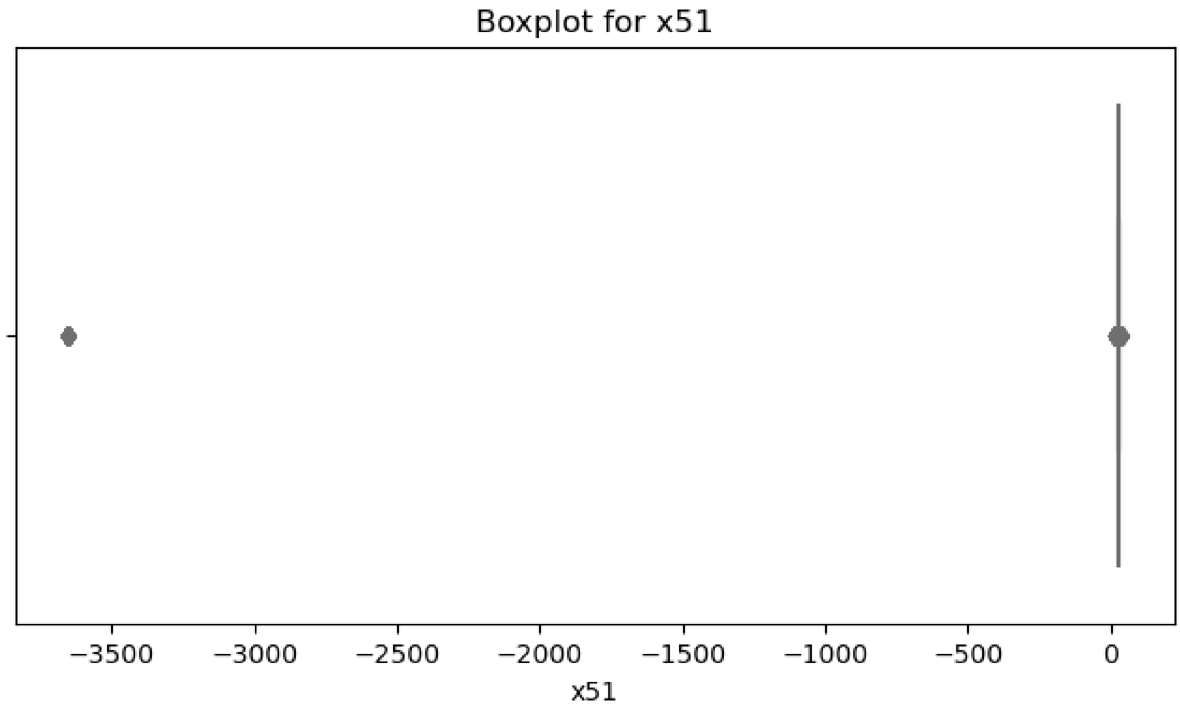

Boxplot for x46
−2
0
2
4
6
8
10
x46

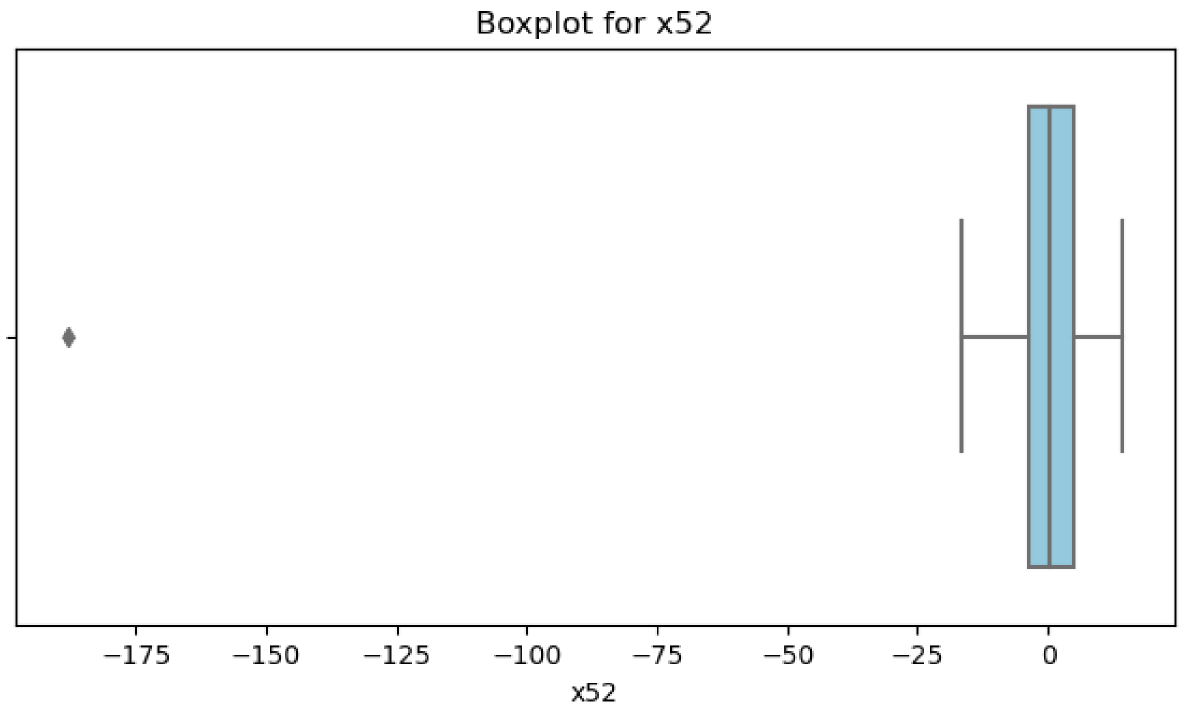

Boxplot for x47
−150
−100
−50
0
50
100
150
x47

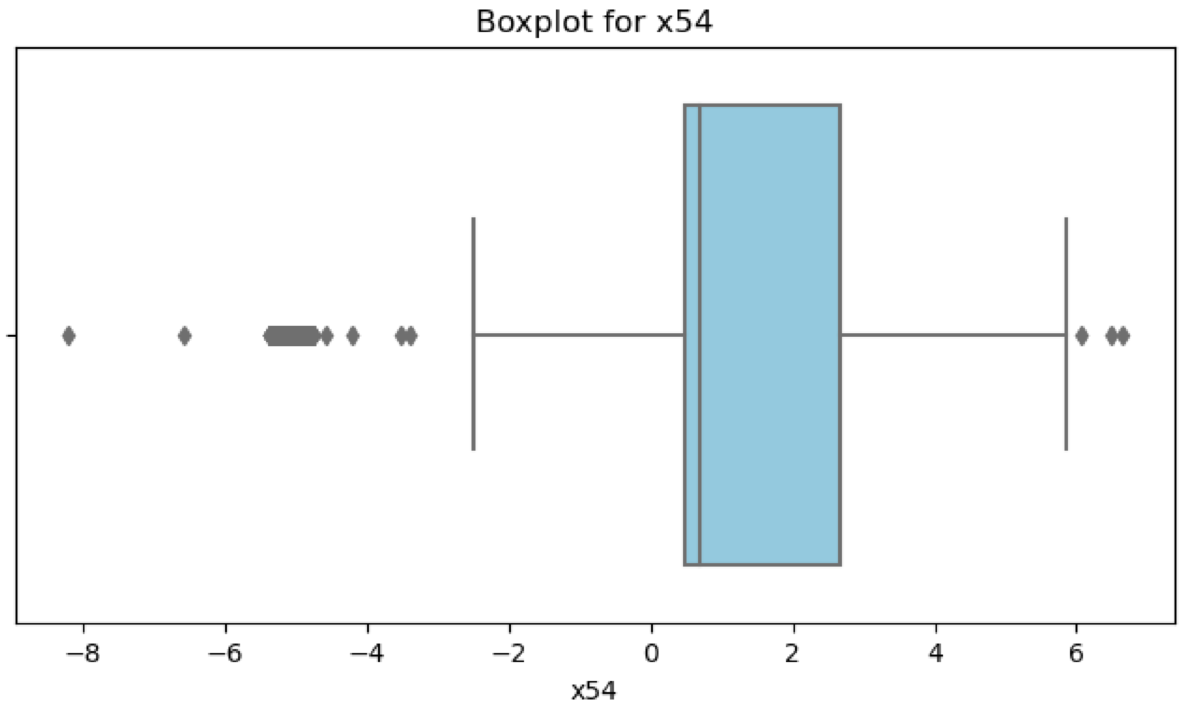

Boxplot for x48
−4
−2
0
2
4
x48


Boxplot for x49
−400
−200
0
200
400
x49

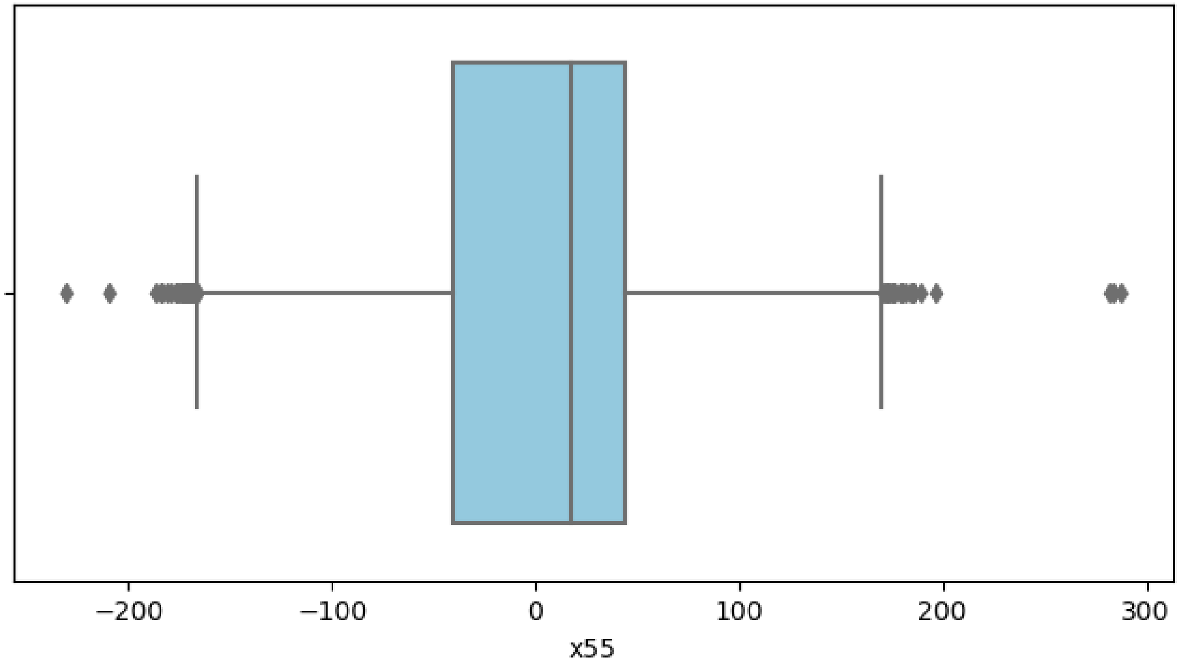

Boxplot for x50
−20
−10
0
10
x50


Boxplot for x51
−3500
−3000
−2500
−2000
−1500
−1000
−500
0
x51

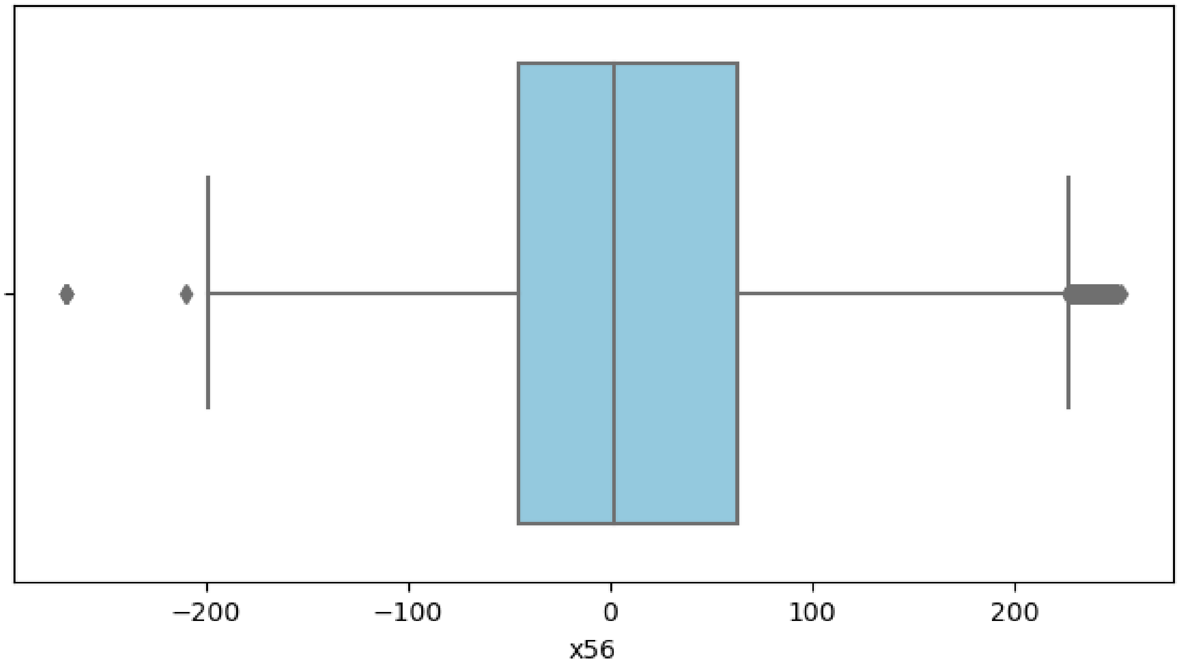

Boxplot for x52
−175
−150
−125
−100
−75
−50
−25
0
x52

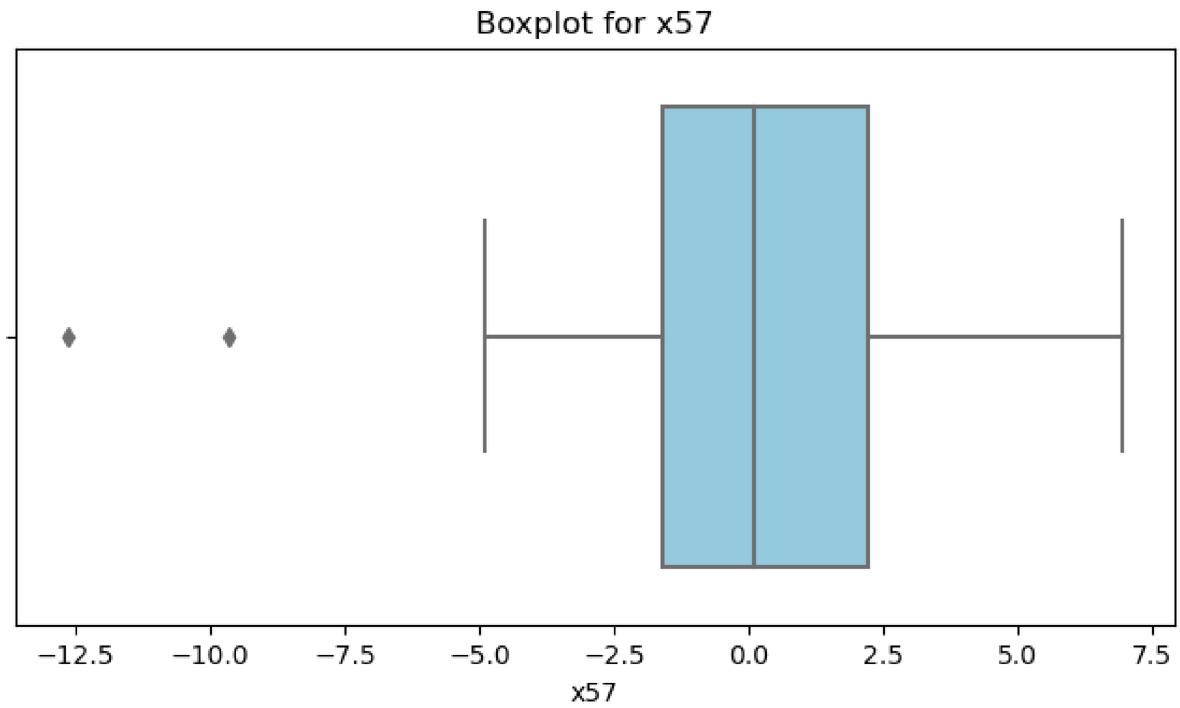

Boxplot for x54
−8
−6
−4
−2
0
2
4
6
x54

Boxplot for x55

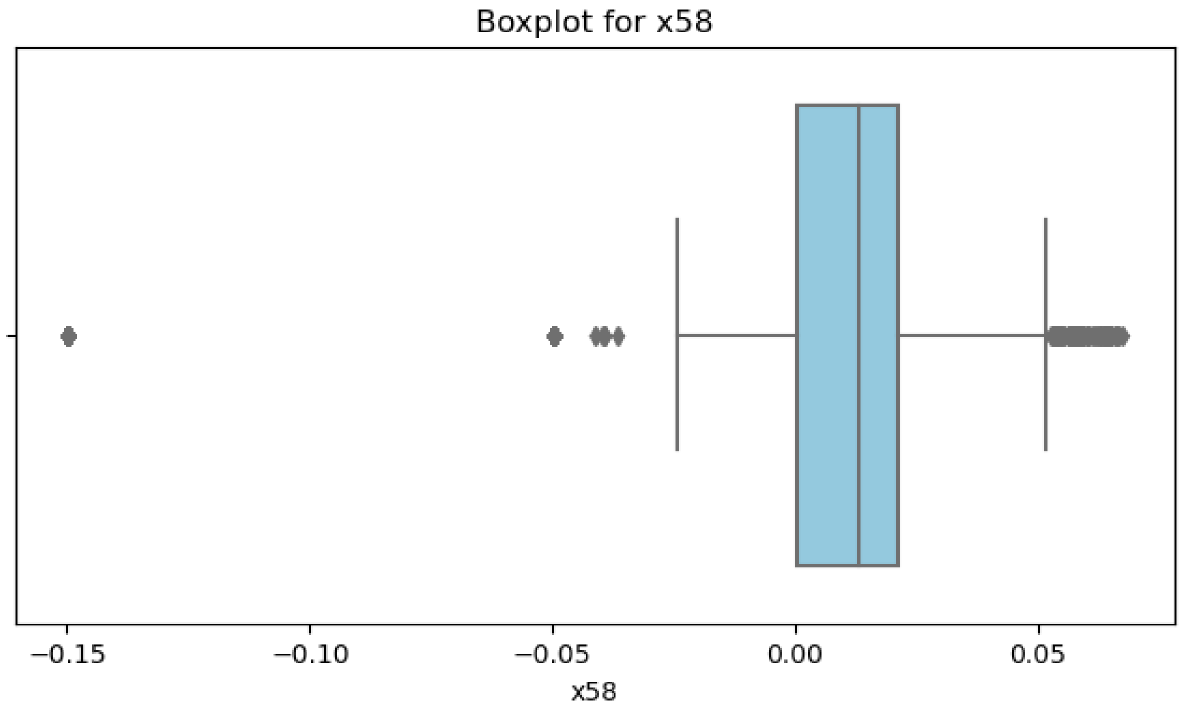


Boxplot for x56

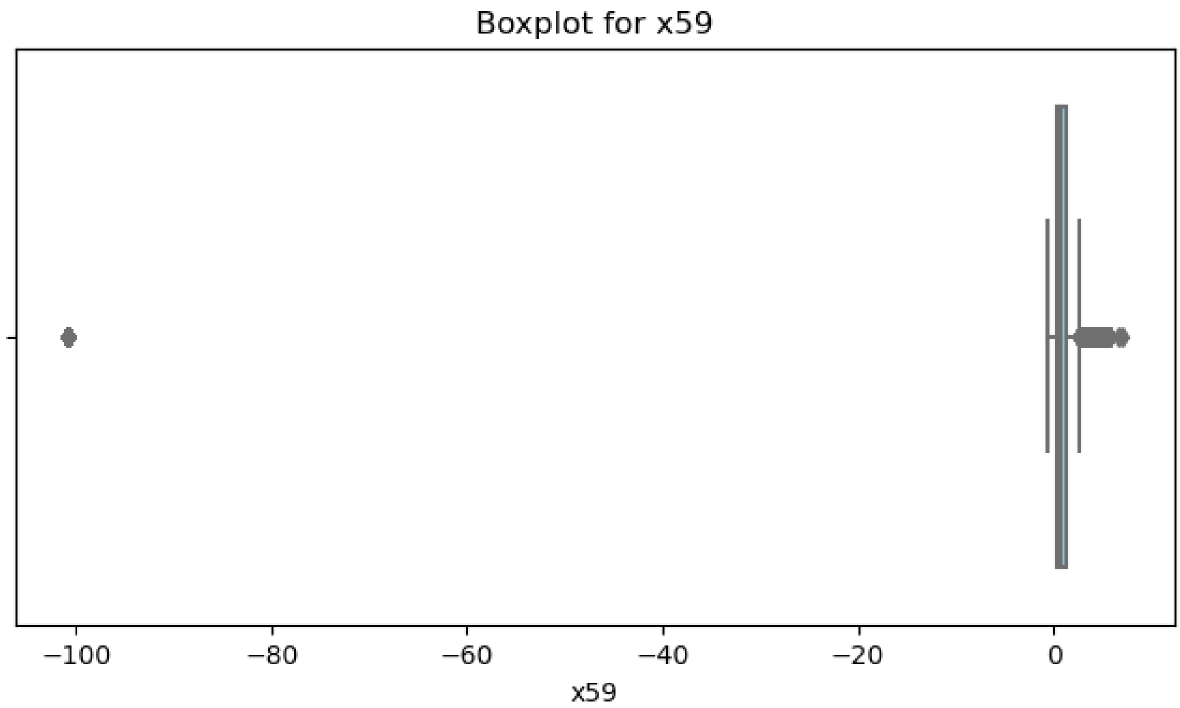




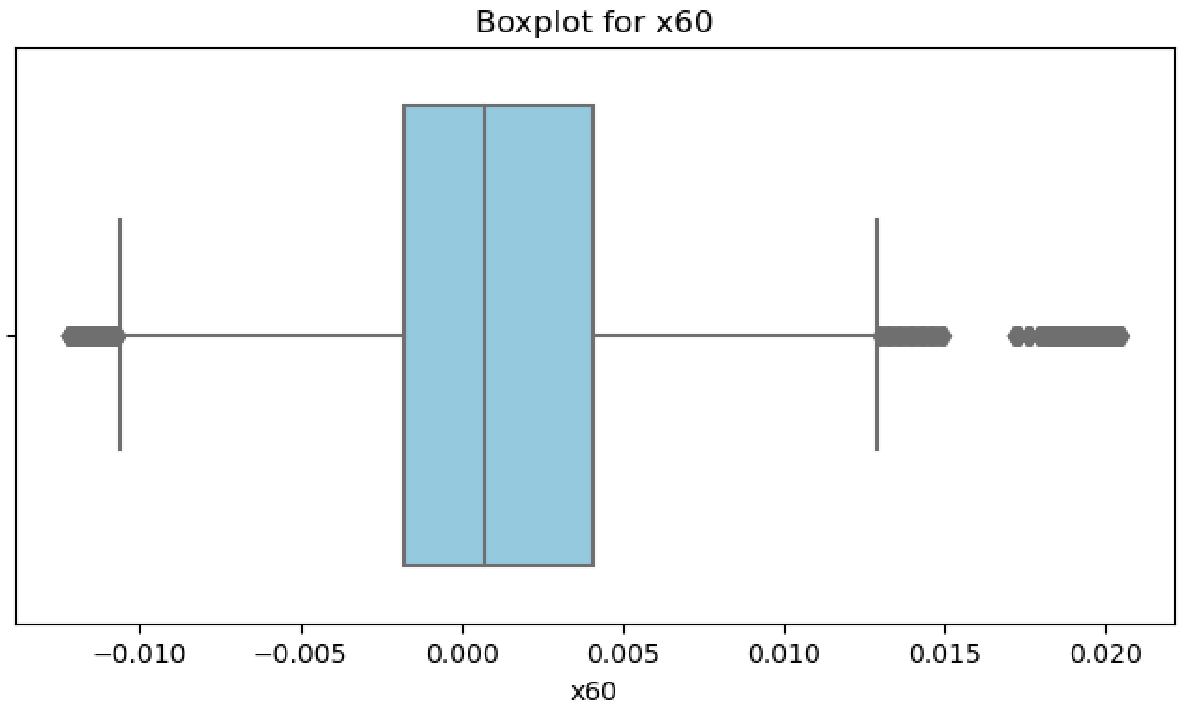
Boxplot for x57
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
x57



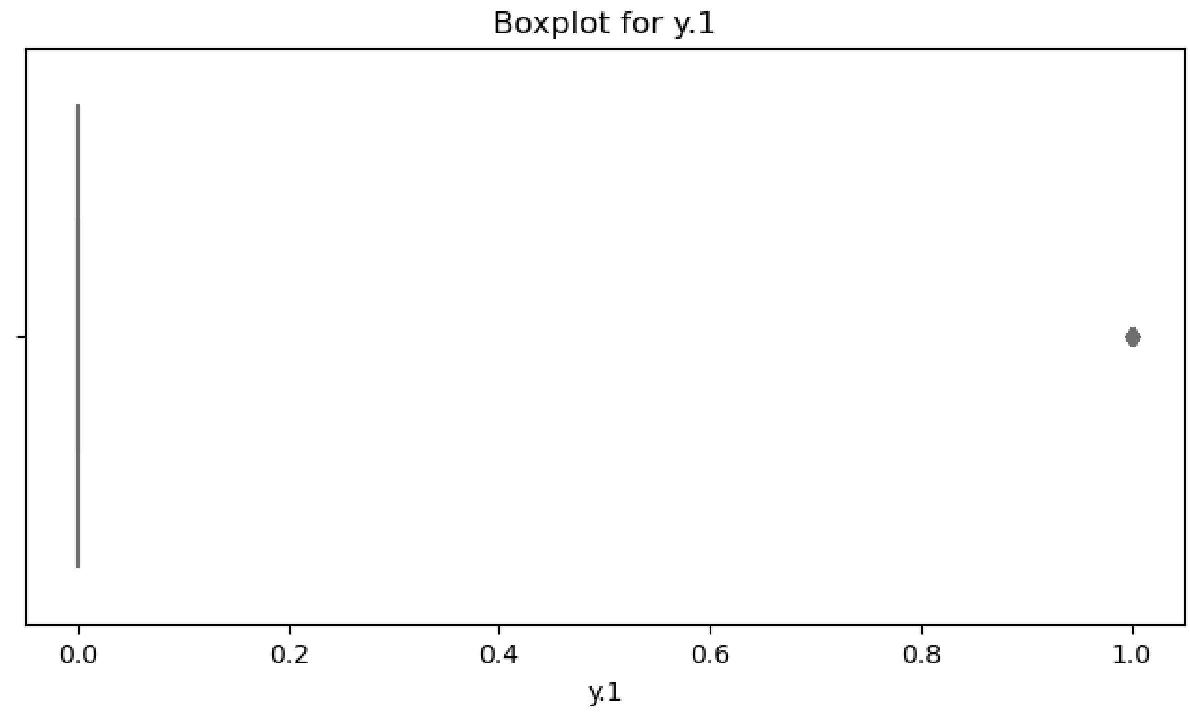
Boxplot for x58
−0.15
−0.10
−0.05
0.00
0.05
x58



Boxplot for x59
−100
−80
−60
−40
−20
0
x59

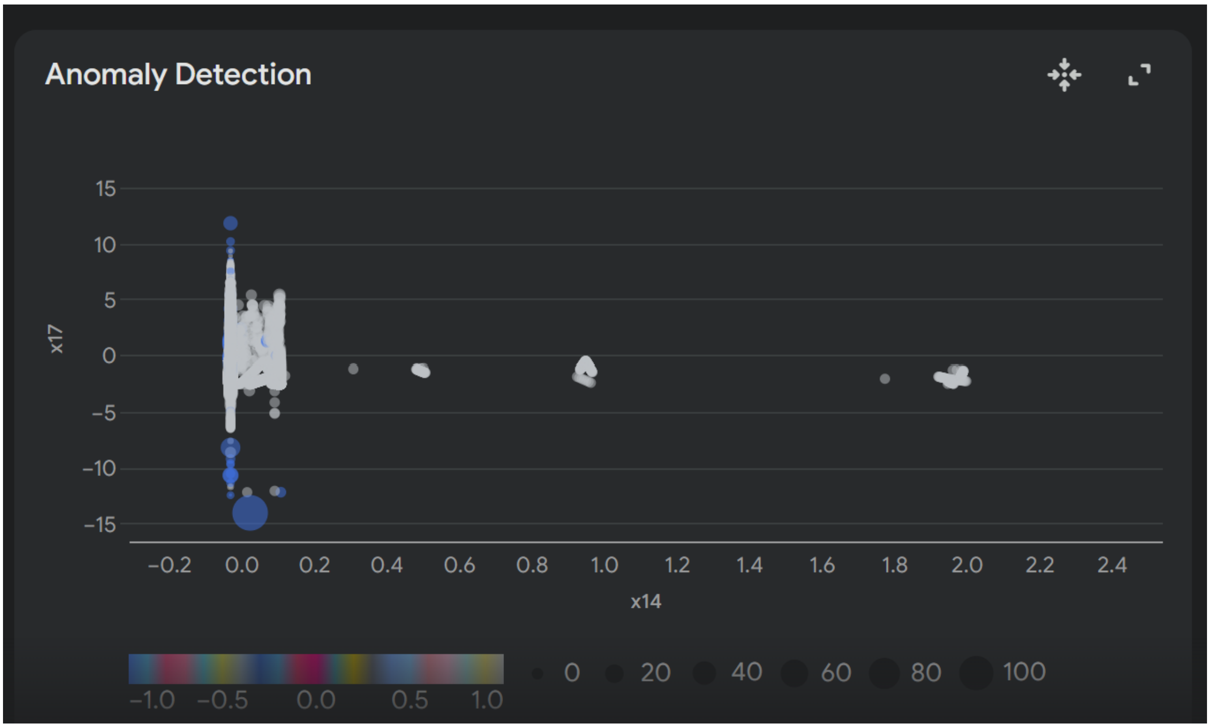


Boxplot for x60
−0.010
−0.005
0.000
0.005
0.010
0.015
0.020
x60



3. Anomaly Detection with Isolation Forest

The Isolation Forest algorithm was chosen for anomaly detection due to its effectiveness in identifying anomalies in high-dimensional datasets. This algorithm isolates anomalies by randomly selecting a feature and then randomly creating a split value between the maximum and minimum values of the selected feature. The process is repeated recursively until all data points are isolated. Anomalies are identified as the points that are isolated first, requiring fewer splits.



4. Model Implementation and Evaluation

An Isolation Forest model was initialized with a random_state set to 42 for reproducibility. The model was trained on the entire dataset. Predictions were made on the same dataset to identify anomalies within it. The algorithm assigned a value of '1' to data points classified as normal and '-1' to those identified as anomalies.

5. Results and Discussion

The results of the anomaly detection process are stored in a new column named 'Predictions'. A '1' in this column indicates that the corresponding data point is not an anomaly, while a '-1' suggests it is an anomaly. The analysis successfully identified anomalies within the dataset.

6. Future Work

Future work may involve exploring other anomaly detection algorithms, such as One-Class SVM or Local Outlier Factor, and comparing their performance with the Isolation Forest. Additionally, fine-tuning the hyperparameters of the Isolation Forest, including the number of estimators and contamination rate, could potentially improve the accuracy of anomaly detection. Further investigation into the characteristics of the identified anomalies could provide valuable insights into the data and potential underlying issues.

7. Conclusion

The anomaly detection process using the Isolation Forest algorithm effectively identified unusual data points within the provided dataset. This information can be crucial for various data analysis tasks, such as data cleaning, fraud detection, and predictive modelling, by highlighting potential areas of concern or interest within the data.

This scatter plot visualizes the results of an Isolation Forest anomaly detection model. The model has identified two distinct groups within the dataset: 'normal' data points (represented by blue dots) and anomalies (represented by orange dots).

The majority of the data points have been classified as normal, forming a dense cluster towards the center of the visualization. The anomalies, on the other hand, appear scattered and isolated from the main cluster, indicating their deviation from the typical data patterns.

The size of the data points represents the magnitude of the values in the x24 column. Larger data points may indicate stronger deviations from normal patterns, further highlighting the outliers identified by the model.

Source Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest

# Load the data
file_path = "AnomaData.xlsx"  # Update with your file path if needed
data = pd.read_excel(file_path)
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

Cleaning The Dataset and Exploratory Data Analysis
# Display first 5 rows of the data
print("First 5 rows of the dataset:")
print(data.head())


# Basic info about dataset
print("\nDataset Information:")
print(data.info())
```

```
# Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Load the data
file_path = "AnomaData.xlsx"  # Update with your file path if needed
data = pd.read_excel(file_path)

# Display first 5 rows of the data
print("First 5 rows of the dataset:")
print(data.head())

# Basic information about the dataset
print("\nDataset Information:")
data.info()

# Checking for missing values
print("\nMissing values in the dataset:")
print(data.isnull().sum())

# Drop columns with too many missing values (threshold = 50%)
missing_threshold = len(data) * 0.5
data = data.dropna(axis=1, thresh=missing_threshold)

# Fill remaining missing values with column mean for numerical columns
numerical_columns = data.select_dtypes(include=[np.number]).columns
data[numerical_columns] =
data[numerical_columns].fillna(data[numerical_columns].mean())

# Verify missing values are handled
print("\nMissing values after cleaning:")
print(data.isnull().sum())

# Descriptive statistics for numerical columns
print("\nDescriptive Statistics:")
print(data.describe())

# Check for duplicate rows
print("\nNumber of duplicate rows before cleaning:",
data.duplicated().sum())
data = data.drop_duplicates()
print("Number of duplicate rows after cleaning:", data.duplicated().sum())

# Distribution of target variable 'y'
if 'y' in data.columns:
    plt.figure(figsize=(8, 4))
    sns.countplot(x='y', data=data, palette='Set2')
```

```
    plt.title("Distribution of Target Variable (y)")
    plt.xlabel("y (1 = Anomaly, 0 = Normal)")
    plt.ylabel("Count")
    plt.show()

# Pairplot for understanding feature relationships
print("\nGenerating pairplot for numerical features:")
sns.pairplot(data, hue='y', palette='husl')
plt.show()

# Correlation matrix to understand relationships
print("\nCorrelation Matrix:")
plt.figure(figsize=(12, 8))
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Matrix")
plt.show()

# Histograms for numerical columns
print("\nGenerating histograms for numerical features:")
numerical_columns = data.select_dtypes(include=[np.number]).columns
for col in numerical_columns:
    plt.figure(figsize=(8, 4))
    sns.histplot(data[col], kde=True, bins=30, color='blue')
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.show()

# Boxplots to identify outliers
print("\nGenerating boxplots to identify outliers:")
for col in numerical_columns:
    plt.figure(figsize=(8, 4))
    sns.boxplot(data[col], color='skyblue')
    plt.title(f"Boxplot of {col}")
    plt.show()

# Scatterplots for numerical features vs target
print("\nGenerating scatterplots for features against target variable
'y':")
if 'y' in data.columns:
    for col in numerical_columns:
        if col != 'y':
            plt.figure(figsize=(8, 4))
            sns.scatterplot(x=col, y='y', data=data, color='green')
            plt.title(f"Scatterplot of {col} vs y")
            plt.xlabel(col)
            plt.ylabel("y")
            plt.show()

# Time Series Check if 'timestamp' exists
print("\nChecking for time series analysis:")
if 'timestamp' in data.columns:
    data['timestamp'] = pd.to_datetime(data['timestamp'])
    data = data.set_index('timestamp')
    plt.figure(figsize=(12, 6))
    plt.plot(data.index, data['y'], color='tab:blue')
    plt.title("Time Series of Target Variable (y)")
```

```
    plt.xlabel("Timestamp")
    plt.ylabel("y")
    plt.show()
else:
    print("No 'timestamp' column found for time series analysis.")

# Isolation Forest for anomaly detection
print("\nApplying Isolation Forest for Anomaly Detection:")
features = data.drop(columns=['y']) if 'y' in data.columns else data
iso_forest = IsolationForest(n_estimators=100, contamination=0.05,
random_state=42)
data['anomaly_score'] = iso_forest.fit_predict(features)
data['isolation_forest_anomaly'] = data['anomaly_score'].map({1: 0, -1: 1})

# Plot Isolation Forest results
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['isolation_forest_anomaly'], color='red',
label="Anomalies")
plt.title("Isolation Forest Anomaly Detection")
plt.xlabel("Index / Timestamp")
plt.ylabel("Anomaly (1=Anomaly, 0=Normal)")
plt.legend()
plt.show()

# Summary of anomalies detected
print("\nSummary of Anomalies Detected by Isolation Forest:")
print(data['isolation_forest_anomaly'].value_counts())
```

Isolation Forest pipeline :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report, confusion_matrix
import joblib

# 1. Load and Preprocess the Data
def load_and_preprocess_data(file_path, target_column=None):
    data = pd.read_csv(file_path)
    data = data.dropna()

    if target_column:
        X = data.drop(columns=[target_column])
    else:
        X = data  # Unsupervised case with no target

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    return X_scaled, scaler

# 2. Train Isolation Forest Model
def train_isolation_forest(X_train, contamination=0.1):
    model = IsolationForest(contamination=contamination, random_state=42)
    model.fit(X_train)
    return model
```

```
# 3. Evaluate the Model (if true labels exist)
def evaluate_model(model, X_val, y_true):
    y_pred = model.predict(X_val)
    y_pred = [1 if x == 1 else 0 for x in y_pred]  # Convert -1 to 0
(outliers)
    print("Classification Report:")
    print(classification_report(y_true, y_pred))
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))

# 4. Save the Model and Scaler
def save_model(model, scaler, model_path='isolation_forest_model.pkl',
scaler_path='scaler.pkl'):
    joblib.dump(model, model_path)
    joblib.dump(scaler, scaler_path)

# 5. Load the Model and Make Predictions
def load_and_predict(model_path, scaler_path, new_data_path):
    model = joblib.load(model_path)
    scaler = joblib.load(scaler_path)

    new_data = pd.read_csv(new_data_path)
    new_data_scaled = scaler.transform(new_data)

    predictions = model.predict(new_data_scaled)
    predictions = [1 if x == 1 else 0 for x in predictions]
    return predictions

# 6. Exploratory Data Analysis for Anomalies
def exploratory_data_analysis(file_path):
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Load the dataset
    data = pd.read_csv(file_path)

    print("Data Overview:")
    print(data.head())
    print("\nData Description:")
    print(data.describe())
    print("\nMissing Values:")
    print(data.isnull().sum())

    # Pair plot for understanding relationships
    print("\nGenerating pair plot...")
    sns.pairplot(data)
    plt.show()

    # Boxplots to detect outliers
    print("\nGenerating boxplots for each feature...")
    for column in data.columns:
        plt.figure(figsize=(10, 6))
        sns.boxplot(data[column])
        plt.title(f"Boxplot for {column}")
        plt.show()

    # Correlation heatmap
```

```
    print("\nGenerating correlation heatmap...")
    plt.figure(figsize=(12, 8))
    sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
    plt.title("Feature Correlation Heatmap")
    plt.show()

# Main Pipeline
if __name__ == "__main__":
    # File paths
    data_path = 'your_dataset.csv'
    new_data_path = 'new_data.csv'
    target_column = None  # Set to your target column if labeled data
exists

    # Load and preprocess data
    X, scaler = load_and_preprocess_data(data_path, target_column)
    X_train, X_val = train_test_split(X, test_size=0.2, random_state=42)

    # Train model
    model = train_isolation_forest(X_train, contamination=0.1)

    # If labeled data is available for evaluation
    # y_true = ... # Provide true labels here
    # evaluate_model(model, X_val, y_true)

    # Save model and scaler
    save_model(model, scaler)

    # Predict on new data
    predictions = load_and_predict('isolation_forest_model.pkl',
'scaler.pkl', new_data_path)
    print("Predictions on new data:", predictions)

    # Perform Exploratory Data Analysis
    print("\nPerforming Exploratory Data Analysis...")
    exploratory_data_analysis(data_path)
```