Importing Libraries

In [150]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [110]:
```
# reading test and labels csv
df_test=pd.read_csv("test.csv", header=None)
df_test_label=pd.read_csv("test_label.csv", header=None)

# reading smap.test and labels csv
df_smap_test=pd.read_csv("smap_test.csv", header=None)
df_smap_test_label=pd.read_csv("smap_test_label.csv", header=None)

# reading msl.test and labels csv
df_msl_test=pd.read_csv("msl_test.csv", header=None)
df_msl_test_label=pd.read_csv("msl_test_label.csv", header=None)

# reading psm.test and labels csv
df_psm_test=pd.read_csv("psm_test.csv")
df_psm_test_label=pd.read_csv("psm_test_label.csv")
```

If File Extraction Is Not Recommended We Can Use The Below Code To Load The Data

In [59]:
```

# Define the path to the ZIP file
#zip_path = "files.zip"

# Use zipfile to read files within the ZIP archive
#with zipfile.ZipFile(zip_path, 'r') as z:
    # Read CSV files into DataFrames
#     df_test = pd.read_csv(z.open("test.csv"))
#     df_test_label = pd.read_csv(z.open("test_label.csv"))

#    df_smap_test = pd.read_csv(z.open("smap_test.csv"))
#    df_smap_test_label = pd.read_csv(z.open("smap_test_labels.csv"))

#    df_msl_test = pd.read_csv(z.open("msl_test.csv"))
#    df_msl_test_label = pd.read_csv(z.open("msl_test_labels.csv"))

#    df_psm_test = pd.read_csv(z.open("psm_test.csv"))
#    df_psm_test_label = pd.read_csv(z.open("psm_test_labels.csv"))

```

Plotting the time Series dataset

In [73]:

```
def plot_anomalies(data, labels, title):
    plt.figure(figsize=(12, 6))
    sns.set(style="whitegrid")

    for i in range(1, data.shape[1]):  # Plot all columns, skipping the first (assumed index)
        plt.plot(data.index, data[i], alpha=0.3)  # Reduced opacity to handle overlaps


    anomaly_indices = labels[labels[0] == 1].index.tolist()  # Indices where label is 1
    plt.scatter(anomaly_indices, [data.loc[i, 1] for i in anomaly_indices],
                color='red', s=100, marker='x', alpha=0.8, label='Anomaly')  # Larger marker

    plt.ylim(-2, 2)  # Set y-axis range
    plt.gca().yaxis.set_major_locator(ticker.MultipleLocator(0.25))  # Major ticks every 0.25

    # Customize the plot
    plt.title(title)
    plt.xlabel("Time")
    plt.ylabel("Values")
    plt.grid(True)


    plt.legend()

    plt.show()
```

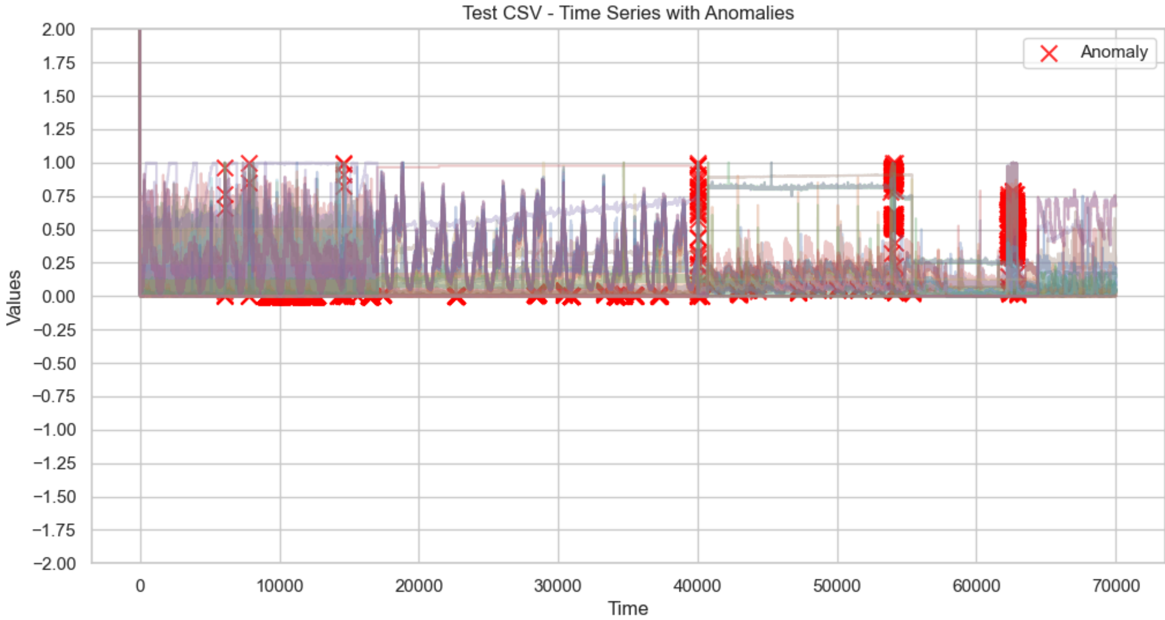
In [74]:
```python
print("Test Data Columns:", test_data.columns)  # Should be a list of column indices/names
print("Test Label Columns:", test_labels.columns)  # Should also have correct columns
print("First few rows of Test Data:")
print(test_data.head())
print("First few rows of Test Labels:")
print(test_labels.head())
```

```
Test Data Columns: Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 1
6, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
       36, 37],
      dtype='int64')
Test Label Columns: Index([0], dtype='int64')
First few rows of Test Data:
    0        1         2    3    4         5         6    7         8    9  \
0  0.0  1.00000  2.000000  3.0  4.0  5.000000  6.000000  7.0  8.000000  9.0
1  0.0  0.00034  0.000432  0.0  0.0  0.694290  0.038316  0.0  0.000000  0.0
2  0.0  0.00051  0.000576  0.0  0.0  0.694702  0.038856  0.0  0.427536  0.0
3  0.0  0.00051  0.000576  0.0  0.0  0.694908  0.038856  0.0  0.000000  0.0
4  0.0  0.00017  0.000432  0.0  0.0  0.695114  0.038856  0.0  0.007246  0.0

   ...    28     29         30         31    32         33         34  \
0  ...  28.0  29.00  30.000000  31.000000  32.0  33.000000  34.000000
1  ...   0.0   0.50   0.036442   0.000000   0.0   0.023256   0.055147
2  ...   0.0   0.25   0.025862   0.000000   0.0   0.028623   0.040441
3  ...   0.0   0.25   0.307994   0.013699   0.0   0.026834   0.183824
4  ...   0.0   0.25   0.026254   0.000000   0.0   0.030411   0.047794

          35    36    37
0  35.000000  36.0  37.0
1   0.055147   0.0   0.0
2   0.040441   0.0   0.0
3   0.180147   0.0   0.0
4   0.047794   0.0   0.0

[5 rows x 38 columns]
First few rows of Test Labels:
   0
0  0
1  0
2  0
3  0
4  0
```

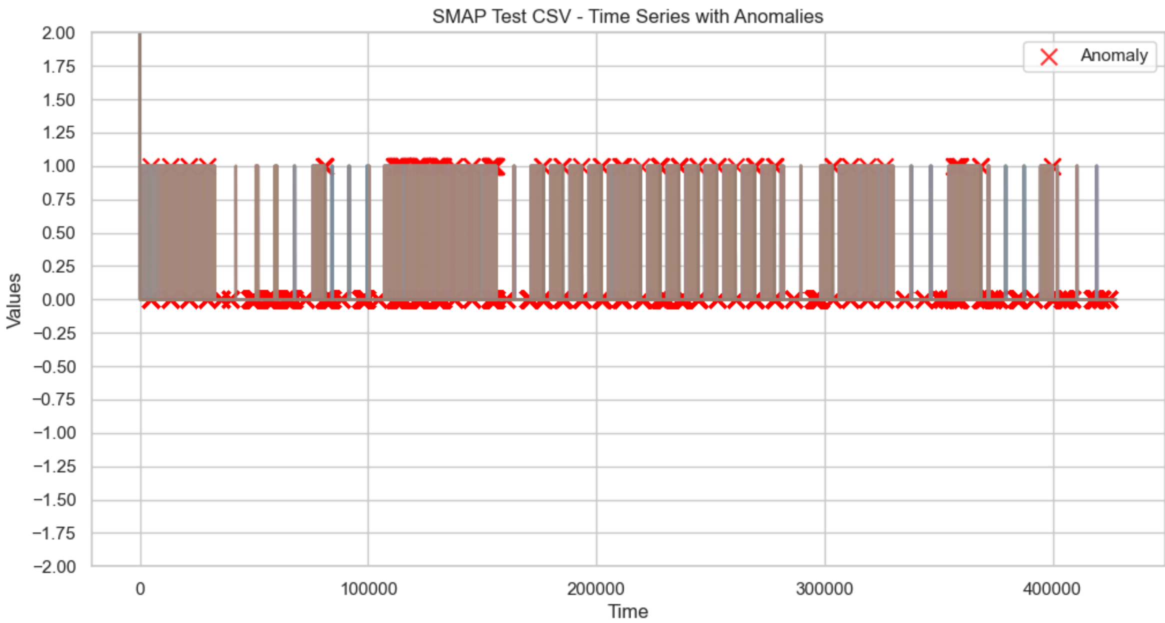
In [75]:
```python
test_data[1]
```

Out[75]:
```
0        1.000000
1        0.000340
2        0.000510
3        0.000510
4        0.000170
           ...
69997    0.047847
69998    0.031100
69999    0.021531
70000    0.039474
70001    0.061005
Name: 1, Length: 70002, dtype: float64
```

In [76]:
```
plot_anomalies(test_data, test_labels, "Test CSV - Time Series with Anomalies")
```

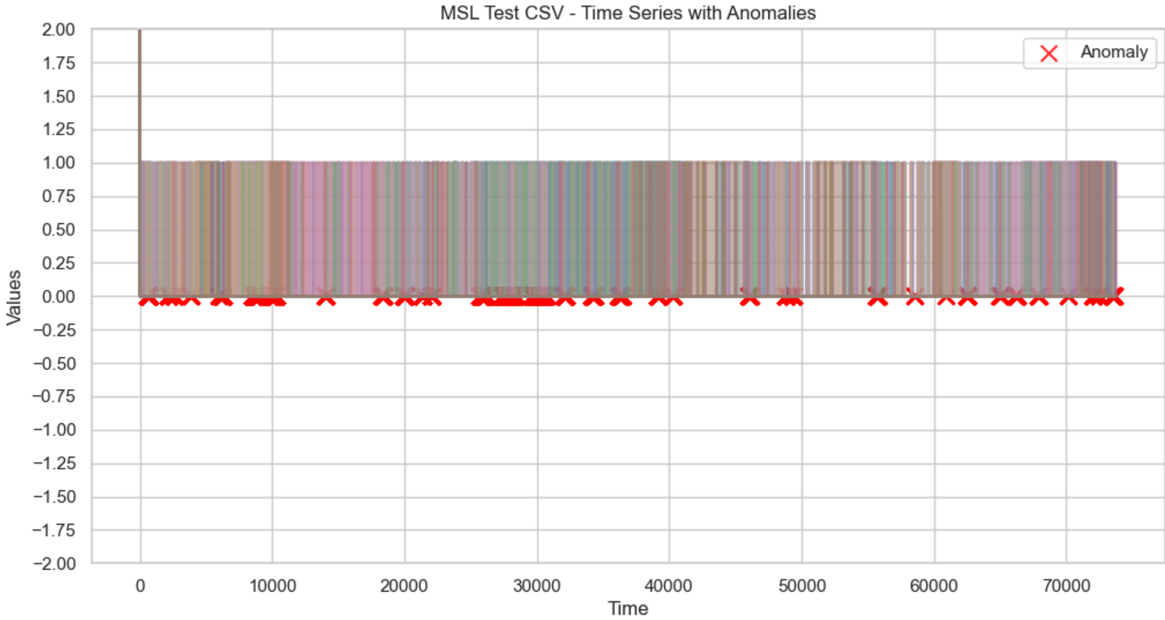


In [77]:
```
plot_anomalies(df_smap_test, df_smap_test_label, "SMAP Test CSV - Time Series with Anomalies"
```



In [78]:

```
plot_anomalies(df_msl_test, df_msl_test_label, "MSL Test CSV - Time Series with Anomalies")
```

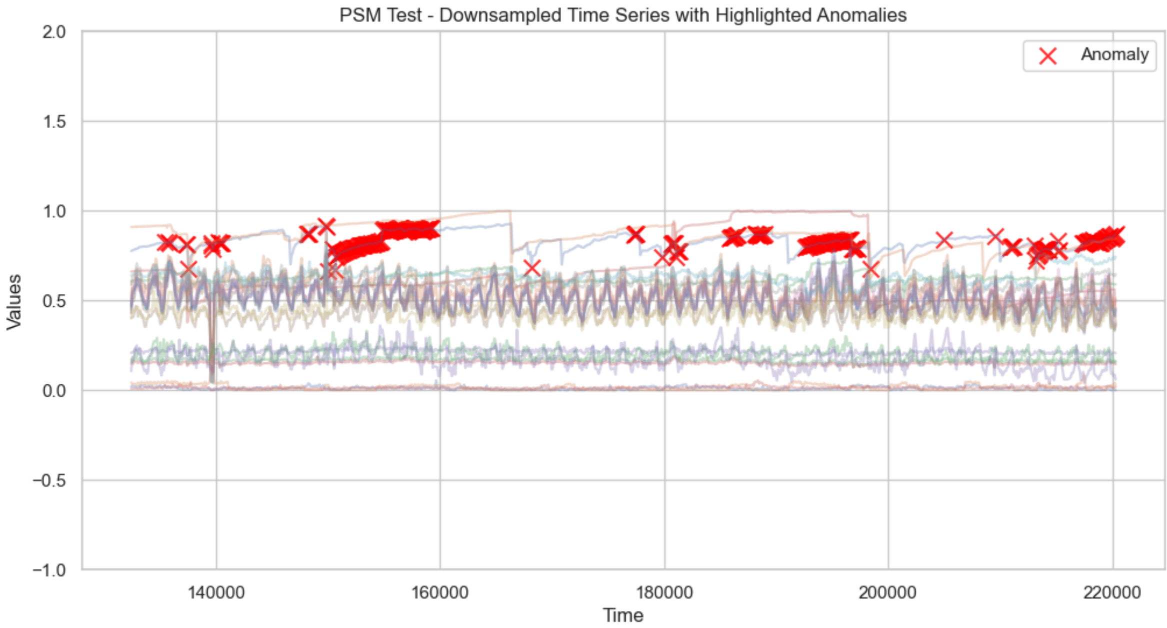


In [79]:

```
# Downsample the data by selecting every 100th row
downsampled_data = df_psm_test.iloc[::100]  # Select every 10th row
downsampled_labels = df_psm_test_label.iloc[::100]  # Downsample labels similarly

# Function to plot time series with highlighted anomalies
def plot_downsampled_anomalies1(data, labels, title="Downsampled Time Series with Anomalies")
    plt.figure(figsize=(12, 6))
    sns.set(style="whitegrid")

    # Plot the downsampled time series data
    for column in data.columns[1:]:  # Skip the first column if it's an index/timestamp
        plt.plot(data["timestamp_(min)"], data[column], alpha=0.3)  # Plot all data columns

    anomaly_indices = labels[labels["label"] == 1]["timestamp_(min)"].tolist()  # Get the dow
    plt.scatter(anomaly_indices, [data.loc[data["timestamp_(min)"] == i, data.columns[1]] for
                color='red', s=100, marker='x', alpha=0.8, label='Anomaly')  # Highlight anom

    # Customizing the y-axis for better vizuvalization
    plt.gca().yaxis.set_major_locator(ticker.MultipleLocator(0.5))  # Major ticks every 0.5 u
    plt.gca().yaxis.set_minor_locator(ticker.MultipleLocator(0.25))  # Minor ticks for extra
    plt.ylim(-1, 2)  # Limit the y-axis range

    # Customize the plot
    plt.title(title)
    plt.xlabel("Time")
    plt.ylabel("Values")
    plt.grid(True)

    plt.legend(loc="upper right")

    plt.show()  # Display the plot
```

In [80]:
```
plot_downsampled_anomalies1(downsampled_data, downsampled_labels, "PSM Test - Downsampled Tim
```

```
C:\Users\NANI\anaconda3\envs\notebook\lib\site-packages\matplotlib\cbook.py:1699: FutureWarn
ing: Calling float on a single element Series is deprecated and will raise a TypeError in th
e future. Use float(ser.iloc[0]) instead
  return math.isfinite(val)
```



Performing EDA to find out root cause

In [82]:

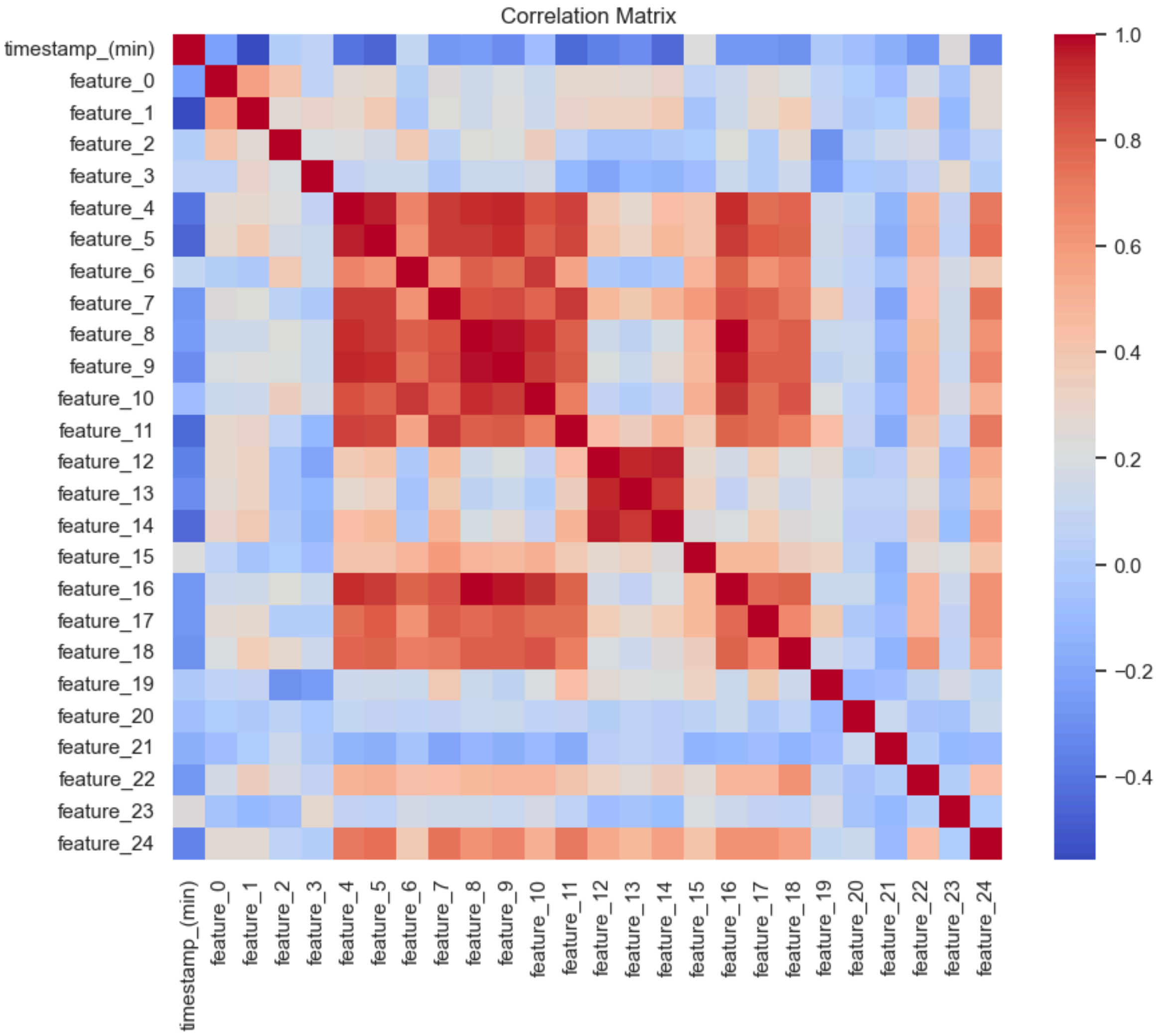
```
# Descriptive statistics
import pandas as pd

# Display basic statistics for the dataset
summary_stats = df_psm_test.describe()
print("Summary Statistics:")
print(summary_stats)

# Check for missing values
missing_values = psm_test_data.isnull().sum()
print("Missing Values:")
print(missing_values)

# Display counts of labels
label_counts = df_psm_test_label["label"].value_counts()
print("Label Distribution:")
print(label_counts)
```

```
Summary Statistics:
       timestamp_(min)     feature_0     feature_1     feature_2  \
count     87841.000000  87841.000000  87841.000000  87841.000000
mean     176400.000000      0.829105      0.857500      0.622801
std       25357.656835      0.047640      0.073858      0.031606
min      132480.000000      0.521701      0.387415      0.453511
25%      154440.000000      0.795859      0.809958      0.604902
50%      176400.000000      0.825835      0.867525      0.616689
75%      198360.000000      0.861815      0.914453      0.636475
max      220320.000000      0.928893      1.000000      0.720898

          feature_3     feature_4     feature_5     feature_6     feature_7  \
count  87841.000000  87841.000000  87841.000000  87841.000000  87841.000000
mean       0.652205      0.516833      0.482637      0.539184      0.520756
std        0.171261      0.077901      0.069440      0.054200      0.069604
min        0.331163      0.073765      0.117442      0.193182      0.080438
25%        0.548472      0.463117      0.433682      0.502273      0.474320
50%        0.585220      0.519564      0.480848      0.534091      0.519637
75%        0.671173      0.573124      0.529915      0.568182      0.567221
max        1.000000      1.000000      1.000000      0.880682      1.000000

          feature_8  ...    feature_15    feature_16    feature_17  \
count  87841.000000  ...  87841.000000  87841.000000  87841.000000
mean       0.528672  ...      0.430095      0.530528      0.611334
std        0.072979  ...      0.042759      0.071854      0.043456
min        0.036741  ...      0.158811      0.077798      0.393533
25%        0.481604  ...      0.401386      0.483288      0.581527
50%        0.531086  ...      0.430216      0.533210      0.605214
75%        0.575146  ...      0.457492      0.576874      0.640682
max        1.000000  ...      0.903567      1.000000      1.000000

         feature_18    feature_19    feature_20    feature_21    feature_22  \
count  87841.000000  87841.000000  87841.000000  87841.000000  87841.000000
mean       0.426030      0.640173      0.010739      0.014477      0.209236
std        0.049188      0.043308      0.010034      0.017572      0.033675
min        0.117788      0.424242      0.000000      0.000000      0.132879
25%        0.394623      0.615479      0.005059      0.000000      0.183885
50%        0.422243      0.640186      0.010118      0.007117      0.205642
75%        0.453484      0.661070      0.015177      0.024911      0.230517
max        0.761031      0.895987      0.994941      1.000000      0.554052

         feature_23    feature_24
count  87841.000000  87841.000000
mean       0.013991      0.174961
std        0.006031      0.057975
min        0.000000      0.023041
25%        0.010893      0.133641
50%        0.013072      0.170507
75%        0.015251      0.211982
max        0.091503      0.990783

[8 rows x 26 columns]
Missing Values:
timestamp_(min)    0
feature_0          0
feature_1          0
feature_2          0
feature_3          0
feature_4          0
feature_5          0
feature_6          0
feature_7          0
feature_8          0
feature_9          0
feature_10         0
feature_11         0
feature_12         0
feature_13         0
feature_14         0
feature_15         0
feature_16         0
feature_17         0
feature_18         0
```

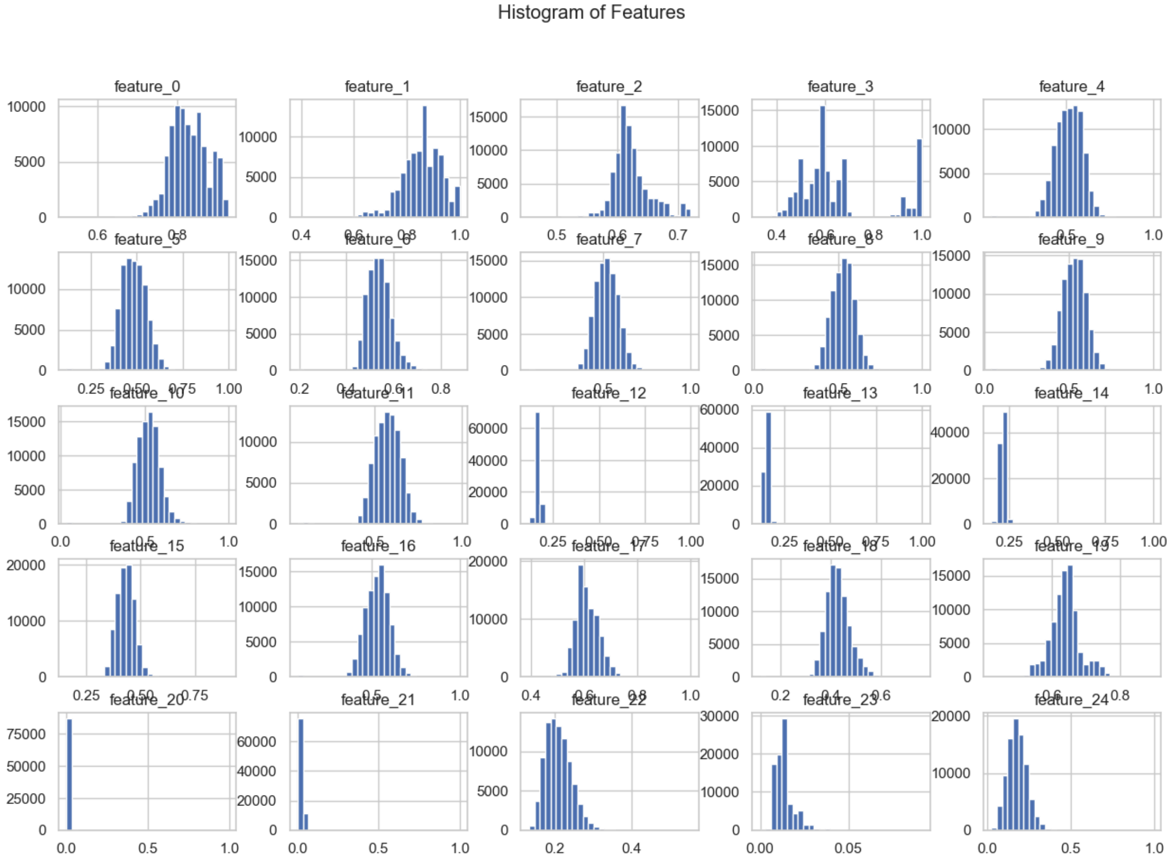
```
feature_19        0
feature_20        0
feature_21        0
feature_22        0
feature_23        0
feature_24        0
dtype: int64
Label Distribution:
label
0    63460
1    24381
Name: count, dtype: int64
```

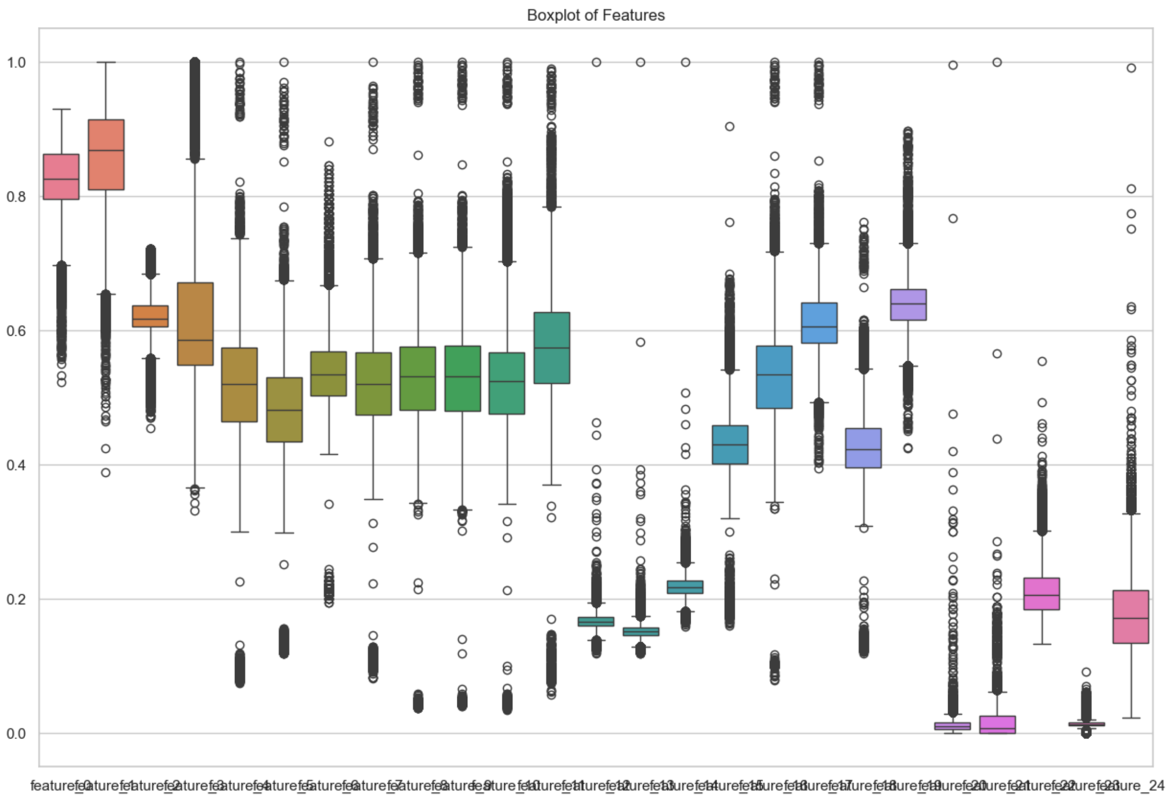
Plotting to Show Anomalies for psm dataset

In [84]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Generate a correlation matrix
correlation_matrix = df_psm_test.corr()

#display corr matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=False, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```

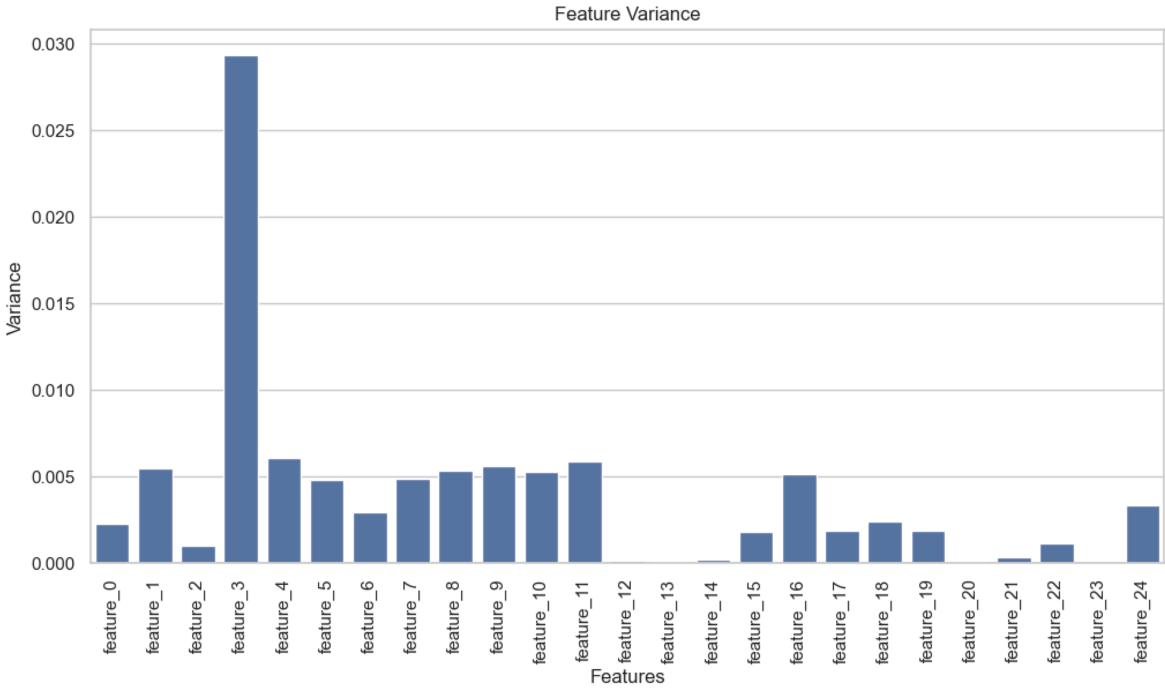


In [101]:
```
# Histogram of the features to visualize distributions
df_psm_test.iloc[:,1:].hist(figsize=(15, 10), bins=30)
plt.suptitle("Histogram of Features")
plt.show()

# Boxplots to identify outliers in the features
plt.figure(figsize=(15, 10))
sns.boxplot(data=df_psm_test.iloc[:, 1:])
plt.title("Boxplot of Features")
plt.show()
```





In [90]:
```
feature_variance = psm_test_data.iloc[:, 1:].var()
plt.figure(figsize=(12, 6))
sns.barplot(x=feature_variance.index, y=feature_variance.values)
plt.title("Feature Variance")
plt.xlabel("Features")
plt.ylabel("Variance")
plt.xticks(rotation=90)  # Rotate feature names for better visibility
plt.show()
```



In [126]:
```
def identify_anomalies(data, labels):

    anomaly_indices = labels[labels["anomaly"] == 1].index  # Index with 'anomaly' = 1

    anomaly_data = data.iloc[anomaly_indices]

    return anomaly_data
def analyze_features_with_anomalies(data, labels, title="Features with Anomalies"):

    #Analyze and visualize features with anomalies. Create a boxplot and print features with 
    anomaly_data = identify_anomalies(data, labels)

    plt.figure(figsize=(12, 8))
    sns.boxplot(data=data.iloc[:, 1:])  # Exclude the first column (timestamp)
    plt.title(title)
    plt.xlabel("Features")
    plt.ylabel("Values")
    plt.xticks(rotation=90)
    plt.show()

    print(f"Anomaly Features for {title}:")
    print(anomaly_data)
```

In [119]:
```
df_msl_test_label.columns = ["anomaly"]  # Second column is the label
df_smap_test_label.columns = ["anomaly"]
df_test_label.columns = ["anomaly"]
```

In [120]:
```
num_features = df_smap_test.shape[1] - 1  # Excluding the timestamp
feature_names = ["timestamp_(min)"] + [f"feature_{i}" for i in range(1, num_features + 1)]

df_smap_test.columns = feature_names

num_features = df_msl_test.shape[1] - 1  # Excluding the timestamp
feature_names = ["timestamp_(min)"] + [f"feature_{i}" for i in range(1, num_features + 1)]

df_msl_test.columns = feature_names

num_features = df_test.shape[1] - 1  # Excluding the timestamp
feature_names = ["timestamp_(min)"] + [f"feature_{i}" for i in range(1, num_features + 1)]

df_test.columns = feature_names
```

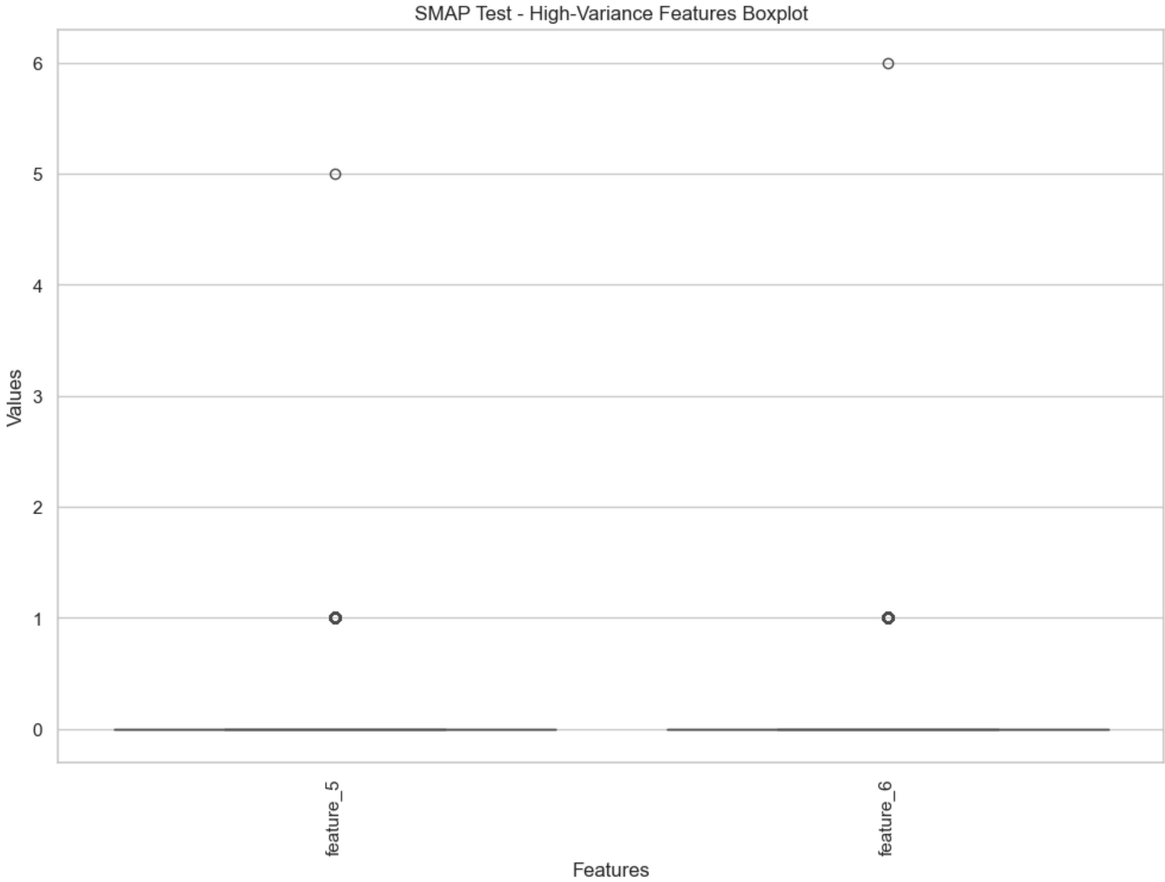
In [131]:
```
smap_variance = df_smap_test.iloc[:, 1:].var()  # Exclude first column
msl_variance = df_msl_test.iloc[:, 1:].var()  # Exclude first column
test_variance = df_test.iloc[:, 1:].var()  # Exclude first column

print("SMAP Test Feature Variance:")
print(smap_variance[:5])

print("MSL Test Feature Variance:")
print(msl_variance[:5])

print("Test Feature Variance:")
print(test_variance[:5])
```

```
SMAP Test Feature Variance:
feature_1    0.018808
feature_2    0.003440
feature_3    0.015176
feature_4    0.000351
feature_5    0.082076
dtype: float64
MSL Test Feature Variance:
feature_1    0.000014
feature_2    0.000054
feature_3    0.000122
feature_4    0.000217
feature_5    0.072040
dtype: float64
Test Feature Variance:
feature_1    0.005253
feature_2    0.007910
feature_3    0.008871
feature_4    0.208798
feature_5    0.120712
dtype: float64
```

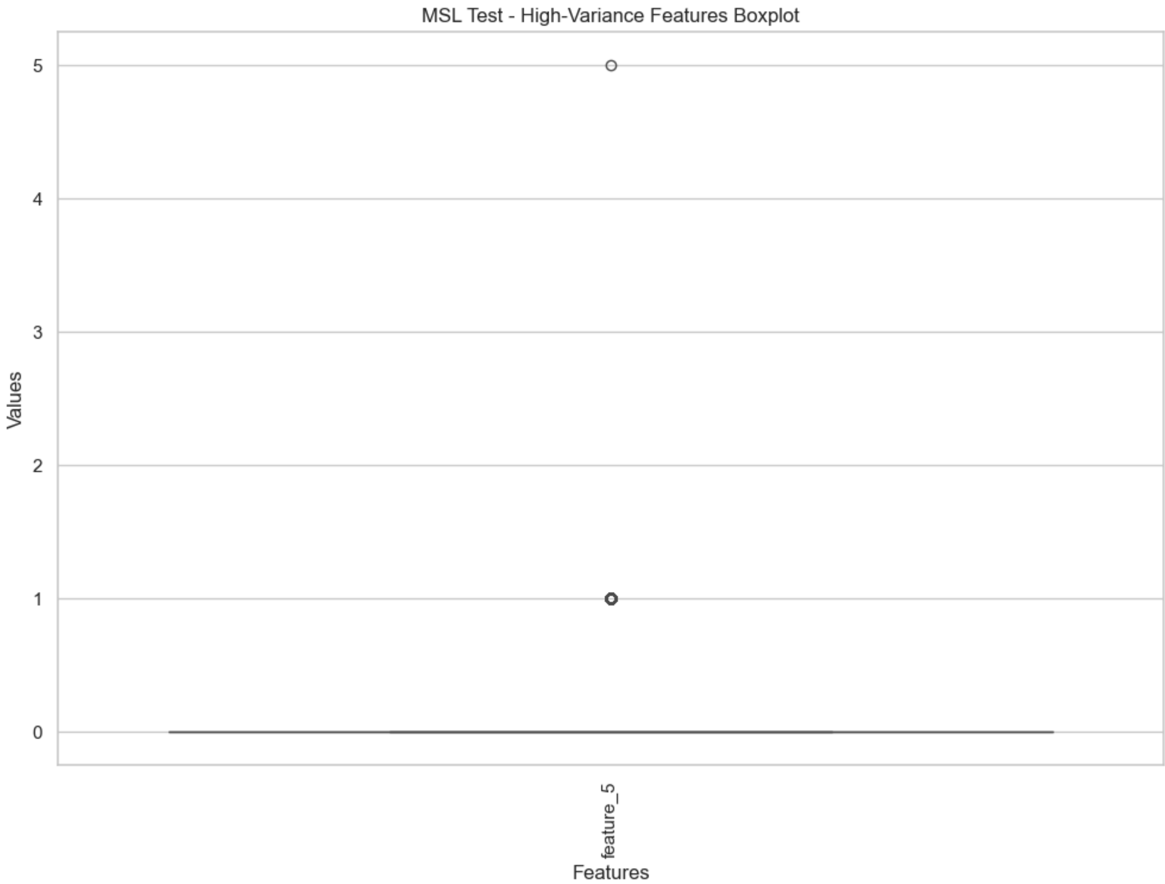
Plotting to show anomalies using Box plot for test, msl, smap datasets

In [140]:
```
def create_boxplot_with_points(data, title="Boxplot with Individual Data Points"):
    plt.figure(figsize=(12, 8))

    # Create a boxplot
    sns.boxplot(data=data.iloc[:, 1:])  # Exclude the first column (timestamp)

    # Add individual data points with swarmplot to show variation
    sns.swarmplot(data=data.iloc[:, 1:], color='black', alpha=0.6)  # Black dots for individu

    plt.title(title)
    plt.xlabel("Features")
    plt.ylabel("Values")
    plt.xticks(rotation=90)  # Rotate feature names for better visibility
    plt.show()
```

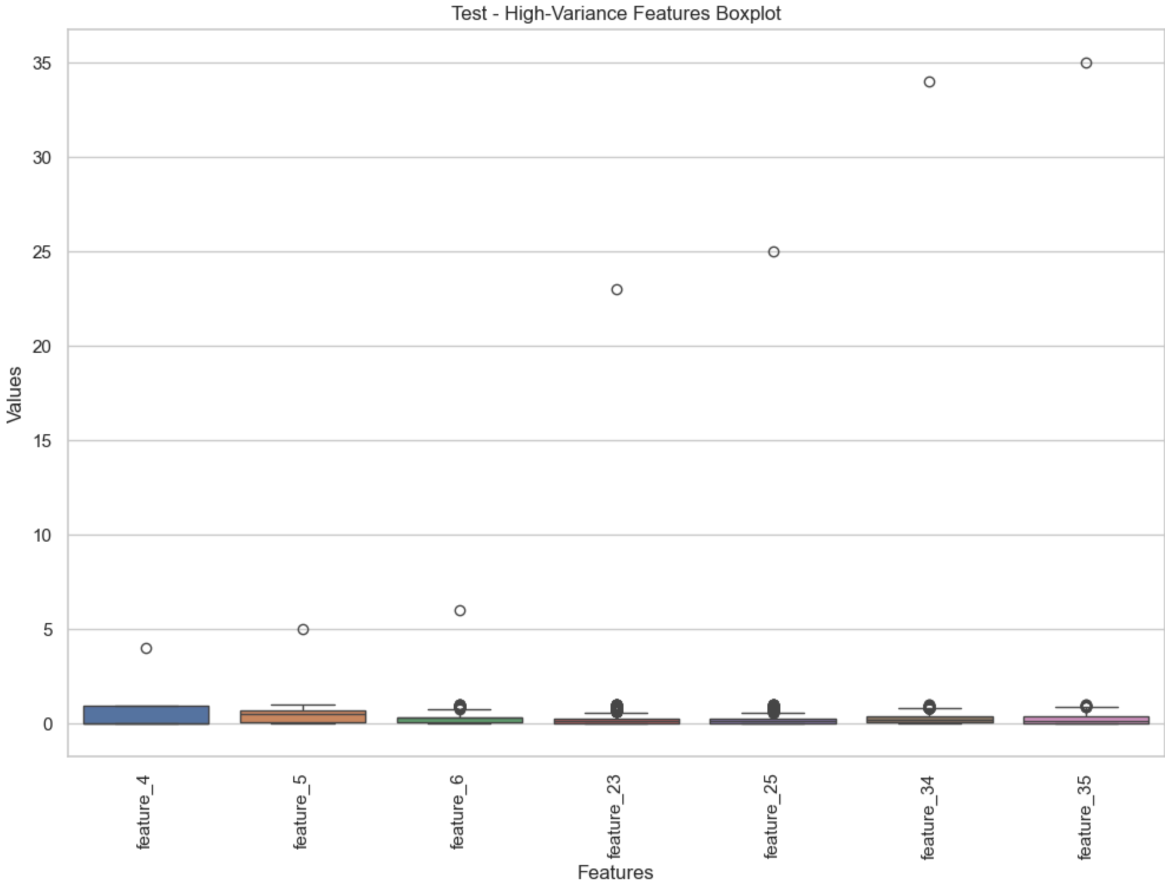
In [141]:
```
create_boxplot_high_variance(df_smap_test, smap_high_variance_features, "SMAP Test - High-Var
```



In [142]:
```
create_boxplot_high_variance(df_msl_test, msl_high_variance_features, "MSL Test - High-Varian
```



In [143]:
```
create_boxplot_high_variance(df_test, test_high_variance_features, "Test - High-Variance Featu
```



Features that are causing anomalies

In [146]:
```
# Set a variance threshold to determine high-variance features
variance_threshold = 0.05

# Find high-variance features for each dataset
smap_high_variance_features = smap_variance[smap_variance > variance_threshold]
msl_high_variance_features = msl_variance[msl_variance > variance_threshold]
test_high_variance_features = test_variance[test_variance > variance_threshold]
```

In [147]:
```
print("SMAP Test - High-Variance Features:")
print(smap_high_variance_features)
```

```
SMAP Test - High-Variance Features:
feature_5    0.082076
feature_6    0.064284
dtype: float64
```

In [148]:
```
print("MSL Test - High-Variance Features:")
print(msl_high_variance_features)
```

```
MSL Test - High-Variance Features:
feature_5    0.07204
dtype: float64
```

In [149]:
```
print("Test - High-Variance Features:")
print(test_high_variance_features)
```

```
Test - High-Variance Features:
feature_4     0.208798
feature_5     0.120712
feature_6     0.098124
feature_23    0.089665
feature_25    0.092598
feature_34    0.061988
feature_35    0.070602
dtype: float64
```

In [145]:
```
variance_threshold = 0.02


high_variance_features = feature_variance[feature_variance > variance_threshold]

print("Features with High Variance:")
print(high_variance_features)
```

```
Features with High Variance:
feature_3    0.02933
dtype: float64
```

In [ ]: