EDA REPORT

TRANSFORMER HEALTH ANALYSIS

Assessing transformer health: Reflecting on past performance, driving future reliability.

Prepared by:
Mansoor ul Hassan

Course
Artificial Intelligence

Project Description: Exploratory Data Analysis (EDA) for Measuring Transformer Health

Overview

This project focuses on performing Exploratory Data Analysis (EDA) on a dataset related to the health assessment of measurement transformers. The dataset comprises various gas concentrations and electrical properties critical for evaluating the condition and performance of these transformers. The columns in the dataset are:

- **Hydrogen**: Concentration of hydrogen gas, often used to detect insulation degradation.
- **Oxygen**: Concentration of oxygen gas, which can be relevant in evaluating chemical reactions in the transformer's environment.
- **Nitrogen**: Concentration of nitrogen gas, which might be involved in insulation or environmental conditions.
- **Methane**: Concentration of methane gas, a potential indicator of insulation breakdown.
- **CO (Carbon Monoxide)**: Concentration of carbon monoxide gas, which can signify insulation or operational issues.
- **CO2 (Carbon Dioxide)**: Concentration of carbon dioxide gas, which may reflect chemical reactions or degradation processes.
- **Ethylene**: Concentration of ethylene gas, often linked to insulation degradation.
- **Ethane**: Concentration of ethane gas, another indicator of insulation degradation.
- **Acetylene**: Concentration of acetylene gas, associated with severe insulation failure.
- **DBDS (Dibenzyl Disulfide)**: Concentration of dibenzyl disulfide, a key indicator of the health of insulation materials.
- **Power Factor**: A measure of the efficiency of electrical power usage, reflecting the health of the electrical system.
- **Interfacial V**: Voltage between different materials or phases, indicative of insulation integrity.
- **Dielectric Rigidity**: Ability of the insulation material to withstand electrical stress, crucial for transformers health.
- **Water Content**: Amount of water present, affecting insulation performance and overall health of the transformers.
- **Health Index**: A composite score or metric representing the overall health and condition of the transformers.

Objectives

1. **Data Cleaning**: Assess and handle missing values, outliers, and inconsistencies to ensure data quality for reliable analysis.
2. **Descriptive Statistics**: Compute and analyze basic statistics (mean, median, mode, standard deviation) for each column to understand distributions and central tendencies.

3. **Data Visualization**: Generate visualizations (e.g., histograms, scatter plots, box plots) to explore distributions, relationships between variables, and trends.
4. **Correlation Analysis**: Examine correlations between gas concentrations, electrical properties, and the health index to identify key indicators of transformer health.
5. **Feature Importance**: Evaluate the importance of each feature in predicting the health index or assessing transformer condition using statistical methods or machine learning techniques.
6. **Anomaly Detection**: Identify any anomalies or unusual patterns that could indicate potential issues with the transformer's health or performance.

Expected Outcomes

- Insightful understanding of how different gas concentrations and electrical properties relate to the overall health of the measurement transformer.
- Visualizations and statistical summaries that reveal important patterns, trends, and relationships in the data.
- Identification of critical features and indicators for transformer health assessment, leading to more informed maintenance and operational decisions.
- Recommendations for further data collection or analysis based on observed findings and potential anomalies.

This EDA will provide valuable insights into the health and performance of measurement transformers, supporting better maintenance strategies and operational efficiency.

About Me

Syed Mansoor ul Hassan Bukhari

- I am a 6th-semester student pursuing a Bachelor of Science in Artificial Intelligence (BS(AI)) at the University of Azad Jammu & Kashmir. My academic focus is on data analysis, machine learning, and the practical applications of artificial intelligence.
- In addition to my university studies, I am also enrolled in an Artificial Intelligence course at Corvit Systems under the guidance of **Sir Muhammad Rizwan**. You can connect with him on LinkedIn.
- Feel free to explore my projects on GitHub or reach out if you have any questions.

Load Libraries

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import f_regression
```

```
from sklearn.ensemble import RandomForestRegressor, IsolationForest,
GradientBoostingRegressor
from sklearn.svm import OneClassSVM
import shap # SHAP (SHapley Additive exPlanations) values provide a
detailed and interpretable view of feature importance.
from scipy.stats import zscore
```

Load DataSet

```
df = pd.read_csv('Health index2.csv')

# Columns In Data Set
df.columns
```

```
Index(['Hydrogen', 'Oxygen', 'Nitrogen', 'Methane', 'CO', 'CO2',
'Ethylene',
       'Ethane', 'Acethylene', 'DBDS', 'Power factor', 'Interfacial
V',
       'Dielectric rigidity', 'Water content', 'Health index'],
      dtype='object')
```

Shape of dataset

```
df.shape
```

```
(470, 15)
```

Head

```
df.head()
```

```
   Hydrogen  Oxygen  Nitrogen  Methane   CO   CO2  Ethylene  Ethane  \
0      2845    5860     27842     7406   32  1344     16684    5467
1     12886      61     25041      877   83   864         4     305
2      2820   16400     56300      144  257  1080       206      11
3      1099      70     37520      545  184  1402         6     230
4      3210    3570     47900      160  360  2130         4      43

   Acethylene  DBDS  Power factor  Interfacial V  Dielectric rigidity
\
0           7  19.0          1.00             45                   55

1           0  45.0          1.00             45                   55

2        2190   1.0          1.00             39                   52

3           0  87.0          4.58             33                   49

4           4   1.0          0.77             44                   55
```

```
   Water content  Health index
0              0          95.2
1              0          85.5
2             11          85.3
3              5          85.3
4              3          85.2
```

1. Data Cleaning

1.1. Check For Missing Values

```
df.isna().sum()
```

```
Hydrogen               0
Oxygen                 0
Nitrogen               0
Methane                0
CO                     0
CO2                    0
Ethylene               0
Ethane                 0
Acethylene             0
DBDS                   0
Power factor           0
Interfacial V          0
Dielectric rigidity    0
Water content          0
Health index           0
dtype: int64
```

Note: This dataset contains `no missing value,` so we `skip step two` which is `handle missing values`.

1.2. Check For Duplicate Values

```
# Check for duplicate rows
duplicates = df.duplicated().sum()

duplicates
```

```
0
```

Note: This dataset contains `no duplicate value`, so we skip step to `drop duplicates`

1.3. Check for Incorrect Data

Incorrect data can include out-of-range values, unexpected data types, and logically inconsistent values. Here's how to identify and handle these:

1.3.1. Out-of-Range Values

Power Factor

```
# Check if 'Power factor' is within the range [0, 1]
out_of_range_power_factor = df[(df['Power factor'] < 0) | (df['Power
factor'] > 1)]

print(f"Number of rows with 'Power factor' > 1:
{len(out_of_range_power_factor)}")
```

```
Number of rows with 'Power factor' > 1: 113
```

```
print("\nOut-of-range 'Power factor' values:\n",
out_of_range_power_factor[['Power factor']].head(20))
```

```
Out-of-range 'Power factor' values:
     Power factor
3            4.58
5            4.93
6            3.53
10           1.32
19          42.10
22           1.27
26           1.24
29           1.16
34           1.71
36           4.45
38          71.75
39          67.67
40          73.20
42           1.90
50           4.40
61           1.68
64           1.15
71           1.82
81           1.02
83           1.55
```

Normalize Power Factor

```
# Define a function to normalize 'Power factor' values
def normalize_power_factor(value):
    if value > 10:
        return min(1, value / 100) # Scaling by 100
    elif value > 1:
        return min(1, value / 10) # Scaling by 10
    return value

# Apply the normalization function
df['Power factor'] = df['Power factor'].apply(normalize_power_factor)

# Verify that all values are within the correct range now
print("\nUnique values in 'Power factor' after normalization:\n\n",
df['Power factor'].unique())
```

```
Unique values in 'Power factor' after normalization:

 [1.     0.458  0.77   0.493  0.353  0.58   0.29   0.27   0.132  0.65
 0.44   0.421  0.25   0.127  0.16   0.23   0.124  0.45   0.116  0.19
 0.171  0.445  0.48   0.7175 0.6767 0.732  0.63   0.12   0.44   0.37
 0.57   0.82   0.15   0.17   0.13   0.21   0.32   0.168  0.115  0.9
 0.14   0.38   0.182  0.3    0.52   0.102  0.155  0.205  0.149  0.89
 0.335  0.28   0.8    0.41   0.86   0.55   0.35   0.08   0.943  0.62
 0.147  0.844  0.36   0.138  0.06   0.95   0.09   0.2    0.24   0.11
 0.158  0.26   0.72   0.141  0.71   0.22   0.54   0.39   0.1    0.159
 0.78   0.17   0.59   0.105  0.46   0.07   0.393  0.193  0.31   0.53
 0.5    0.556  0.293  0.107  0.237  0.323  0.295  0.254  0.101  0.616
 0.234  0.186  0.208  0.249  0.296  0.287  0.278  0.535  0.256  0.304
 0.422  0.338  0.324  0.262  0.429  0.43   0.259  0.275  0.325  0.297
 0.283  0.317  0.93   0.937  0.255  0.6    0.33   0.81   0.64   0.398
 0.67   0.137  0.261  0.218  0.314  0.214  0.201  0.111  0.383  0.197
 0.49   0.152  0.84   0.112  0.145  0.347  0.11   0.75   0.83   0.05
 0.47   0.133  0.76   0.169  0.175  0.439  0.397  0.253  0.446  0.418
 0.215  0.183  0.322  0.73   0.121 ]
```

Verify Data Cleaning on Power Factor

```
# Check if there are any remaining values greater than 1
print("Number of rows with 'Power factor' > 1 after normalization:",
df[df['Power factor'] > 1].shape[0])
```

```
Number of rows with 'Power factor' > 1 after normalization: 0
```

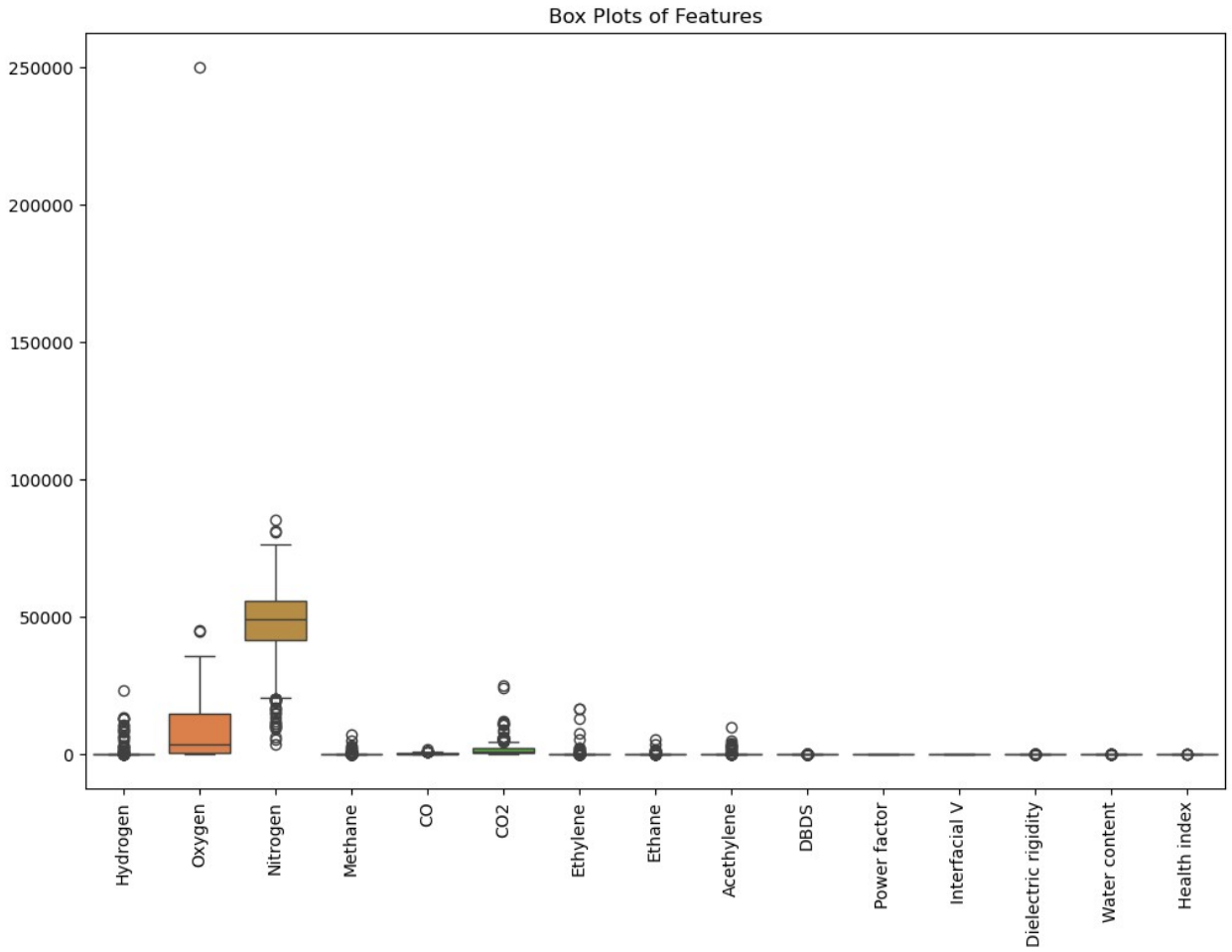
```
print("Number of rows with 'Power factor' < 0 after normalization:",
df[df['Power factor'] < 0].shape[0])
```

```
Number of rows with 'Power factor' < 0 after normalization: 0
```

1.3.2. Value Ranges and Distributions

Visualize distributions to identify unusual patterns or outliers:

```
# Plot box plots to identify outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[['Hydrogen', 'Oxygen', 'Nitrogen', 'Methane',
'CO', 'CO2', 'Ethylene', 'Ethane', 'Acethylene', 'DBDS', 'Power
factor', 'Interfacial V', 'Dielectric rigidity', 'Water content',
'Health index']])
plt.xticks(rotation=90)
plt.title('Box Plots of Features')
plt.show()
```



Looks like Some Columns have outliers, Let's analyze outliers in the Oxygen, Hydrogen, CO2, Nitrogen, and Ethylene columns one by one. The analysis will involve:

- Filtering Values: Identifying values that are outliers.
- Finding Extremes: Determining maximum and minimum values.
- Top 5 Maximum Values: Listing the top 5 highest values.

Oxygen

```
# Filter outliers (values significantly above the mean)
oxygen_mean = df['Oxygen'].mean()
oxygen_std = df['Oxygen'].std()
oxygen_outliers = df[df['Oxygen'] > (oxygen_mean + 3 * oxygen_std)]

# Maximum and minimum values
oxygen_max = df['Oxygen'].max()
oxygen_min = df['Oxygen'].min()

# Top 5 maximum values
top_5_oxygen_max = df.nlargest(5, 'Oxygen')[['Oxygen']]

print("Oxygen Analysis:")
print("-----------------------------------------------")
print("Maximum value:", oxygen_max)
print("Minimum value:", oxygen_min)

print("\n-----------------------------------------\nTop 5 maximum
values:\n-----------------------------------------\n",
top_5_oxygen_max)

print("-----------------------------------------------")
print("\n-----------------------------------------\nOutliers:\
n-----------------------------------------\n",
oxygen_outliers[['Oxygen']])
print("-----------------------------------------------")
```

```
Oxygen Analysis:
-----------------------------------------------
Maximum value: 249900
Minimum value: 57

-----------------------------------------
Top 5 maximum values:
-----------------------------------------
      Oxygen
45   249900
395   45100
250   44788
121   35700
100   30800
-----------------------------------------------

-----------------------------------------
Outliers:
-----------------------------------------
     Oxygen
45  249900
-----------------------------------------------
```

```
df.shape
```

```
(470, 15)
```

```
df = df[df['Oxygen'] < 249900] # Drop Row Where Outlier Occurs
```

```
df.shape
```

```
(469, 15)
```

Hydrogen

```
# Filter outliers (values significantly above the mean)
hydrogen_mean = df['Hydrogen'].mean()
hydrogen_std = df['Hydrogen'].std()

# Maximum and minimum values
hydrogen_max = df['Hydrogen'].max()
hydrogen_min = df['Hydrogen'].min()

# Top 5 maximum values
top_5_hydrogen_max = df.nlargest(5, 'Hydrogen')[['Hydrogen']]

print("\nHydrogen Analysis:")
print("Maximum value:", hydrogen_max)
print("Minimum value:", hydrogen_min)
print("\nTop 5 maximum values:\n", top_5_hydrogen_max)
```

```
Hydrogen Analysis:
Maximum value: 23349
Minimum value: 0

Top 5 maximum values:
     Hydrogen
13     23349
5      13500
15     13200
1      12886
35     12880
```

```
df.shape
```

```
(469, 15)
```

```
df = df[df['Hydrogen'] < 23349] # Drop Row Where Outlier Occurs
```

```
df.shape
```

```
(468, 15)
```

CO2

```
# Filter outliers (values significantly above the mean)
co2_mean = df['CO2'].mean()
co2_std = df['CO2'].std()
co2_outliers = df[df['CO2'] > (co2_mean + 3 * co2_std)]


# Maximum and minimum values
co2_max = df['CO2'].max()
co2_min = df['CO2'].min()


# Top 5 maximum values
top_5_co2_max = df.nlargest(5, 'CO2')[['CO2']]



print("\nCO2 Analysis:")
print("Maximum value:", co2_max)
print("Minimum value:", co2_min)
print("\nTop 5 maximum values:\n", top_5_co2_max)
print("\nOutliers:\n", co2_outliers[['CO2']])
```

```
CO2 Analysis:
Maximum value: 24900
Minimum value: 51

Top 5 maximum values:
        CO2
19   24900
361  24200
456  12000
185  11700
117  11300

Outliers:
        CO2
19   24900
39   10600
40    8930
117  11300
176  11100
185  11700
333  10600
```

```
361  24200
456  12000
```

```
df.shape
```

```
(468, 15)
```

```
df = df[df['CO2'] < 24200] # Drop Row Where Outlier Occurs
```

```
df.shape
```

```
(466, 15)
```

Nitrogen

```
# Filter outliers (values significantly above the mean)
nitrogen_mean = df['Nitrogen'].mean()
nitrogen_std = df['Nitrogen'].std()
nitrogen_outliers = df[df['Nitrogen'] > (nitrogen_mean + 3 *
nitrogen_std)]

# Maximum and minimum values
nitrogen_max = df['Nitrogen'].max()
nitrogen_min = df['Nitrogen'].min()

# Top 5 maximum values
top_5_nitrogen_max = df.nlargest(5, 'Nitrogen')[['Nitrogen']]


print("\nNitrogen Analysis:")
print("Maximum value:", nitrogen_max)
print("Minimum value:", nitrogen_min)
print("\nTop 5 maximum values:\n", top_5_nitrogen_max)
print("\nOutliers:\n", nitrogen_outliers[['Nitrogen']])
```

```
Nitrogen Analysis:
Maximum value: 85300
Minimum value: 3600

Top 5 maximum values:
      Nitrogen
121     85300
63      81300
24      80800
156     76582
59      76500

Outliers:
 Empty DataFrame
```

```
Columns: [Nitrogen]
Index: []
```

Note: `No Outliers Detected`

Ethylene

```
# Filter outliers (values significantly above the mean)
ethylene_mean = df['Ethylene'].mean()
ethylene_std = df['Ethylene'].std()
ethylene_outliers = df[df['Ethylene'] > (ethylene_mean + 3 *
ethylene_std)]


# Maximum and minimum values
ethylene_max = df['Ethylene'].max()
ethylene_min = df['Ethylene'].min()


# Top 5 maximum values
top_5_ethylene_max = df.nlargest(5, 'Ethylene')[['Ethylene']]


print("\nEthylene Analysis:")
print("Maximum value:", ethylene_max)
print("Minimum value:", ethylene_min)
print("\nTop 5 maximum values:\n", top_5_ethylene_max)
print("\nOutliers:\n", ethylene_outliers[['Ethylene']])


Ethylene Analysis:
Maximum value: 16684
Minimum value: 0

Top 5 maximum values:
     Ethylene
0       16684
15      16400
16      13100
14       7820
21       2520

Outliers:
     Ethylene
0       16684
14       7820
15      16400
16      13100
```

```
df.shape
```

```
(466, 15)
```

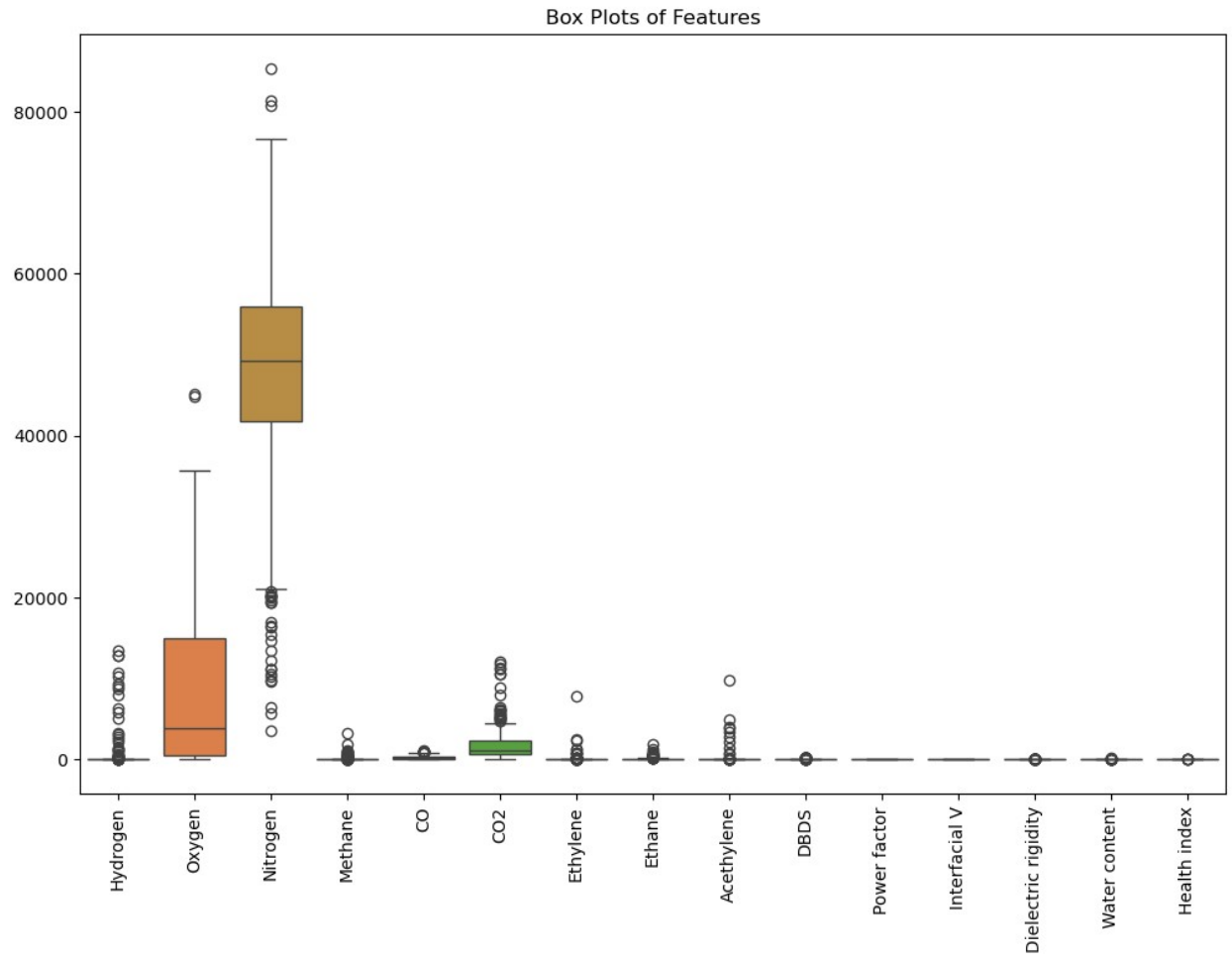
```
df = df[df['Ethylene'] < 13100] # Drop Row Where Outlier Occurs
```

```
df.shape
```

```
(463, 15)
```

Visculize Again

```
# Plot box plots to identify outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=df[['Hydrogen', 'Oxygen', 'Nitrogen', 'Methane',
'CO', 'CO2', 'Ethylene', 'Ethane', 'Acethylene', 'DBDS', 'Power
factor', 'Interfacial V', 'Dielectric rigidity', 'Water content',
'Health index']])
plt.xticks(rotation=90)
plt.title('Box Plots of Features')
plt.show()
```



2. Descriptive Statistics

Generate basic statistics to understand the distribution and central tendencies of the data.

```
# Basic statistics
print("Basic statistics:")
df.describe()
```

```
Basic statistics:

           Hydrogen        Oxygen     Nitrogen      Methane
CO  \
count    463.000000    463.000000    463.00000   463.000000
463.000000
```

```
mean     308.889849   7895.293737  47816.50324    39.444924
240.933045
std     1571.323608   8763.738435  13759.11443   204.424122
227.250548
min        0.000000     57.000000   3600.00000     0.000000
10.000000
25%        3.500000    492.500000  41800.00000     2.000000
66.000000
50%        9.000000   3760.000000  49200.00000     3.000000
148.000000
75%       31.000000  14904.000000  55850.00000     7.000000
361.500000
max    13500.000000  45100.000000  85300.00000  3150.000000
1070.000000

                CO2     Ethylene       Ethane   Acethylene        DBDS
\
count    463.000000   463.000000   463.000000   463.000000  463.000000

mean    1720.978402    47.021598    55.388769    73.069114   17.253132

std     1707.384683   409.438571   141.724750   612.406480   47.047149

min       51.000000     0.000000     0.000000     0.000000    0.000000

25%      638.500000     0.000000     0.000000     0.000000    0.000000

50%     1110.000000     3.000000     4.000000     0.000000    0.000000

75%     2250.000000     6.000000    67.500000     0.000000    2.000000

max    12000.000000  7820.000000  1850.000000  9740.000000  227.000000


       Power factor  Interfacial V  Dielectric rigidity  Water content
\
count    463.000000     463.000000           463.000000     463.000000

mean       0.619454      38.397408            53.473002      16.222462

std        0.364686       6.126530             6.430012      16.973246

min        0.050000      21.000000            27.000000       0.000000

25%        0.260000      32.000000            51.000000       5.000000

50%        0.600000      39.000000            54.000000      12.000000

75%        1.000000      44.000000            56.000000      21.000000

max        1.000000      57.000000            75.000000     183.000000
```

```
       Health index
count    463.000000
mean      27.053348
std       17.274086
min       13.400000
25%       13.400000
50%       13.400000
75%       38.300000
max       85.500000
```

Compute median for each column

```
median_values = df.median(numeric_only=True)

print("\nMedian values:\n", median_values)
```

```
Median values:
 Hydrogen                   9.0
Oxygen                   3760.0
Nitrogen                49200.0
Methane                    3.0
CO                       148.0
CO2                     1110.0
Ethylene                   3.0
Ethane                     4.0
Acethylene                 0.0
DBDS                       0.0
Power factor               0.6
Interfacial V             39.0
Dielectric rigidity       54.0
Water content             12.0
Health index              13.4
dtype: float64
```

Compute mode for each column

- Mode might return multiple values; take the first mode for each column

```
mode_values = df.mode(numeric_only=True).iloc[0]

print("\nMode values:\n", mode_values)
```

```
Mode values:
 Hydrogen                   0.0
Oxygen                  15700.0
Nitrogen                56700.0
```

```
Methane                      2.0
CO                          54.0
CO2                       2750.0
Ethylene                     0.0
Ethane                       0.0
Acethylene                   0.0
DBDS                         0.0
Power factor                 1.0
Interfacial V               32.0
Dielectric rigidity         55.0
Water content                4.0
Health index                13.4
Name: 0, dtype: float64
```

Compute standard deviation for each column

```
std_deviation = df.std(numeric_only=True)
print("\nStandard Deviation:\n", std_deviation)
```

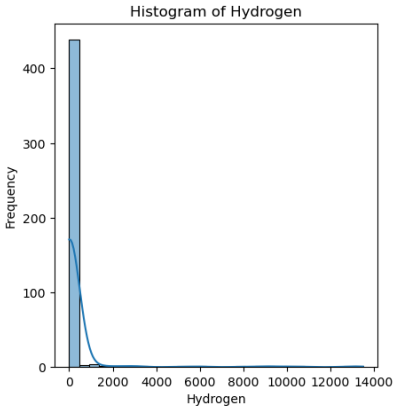
```
Standard Deviation:
 Hydrogen                 1571.323608
Oxygen                    8763.738435
Nitrogen                 13759.114430
Methane                    204.424122
CO                         227.250548
CO2                       1707.384683
Ethylene                   409.438571
Ethane                     141.724750
Acethylene                 612.406480
DBDS                        47.047149
Power factor                 0.364686
Interfacial V                6.126530
Dielectric rigidity          6.430012
Water content               16.973246
Health index                17.274086
dtype: float64
```

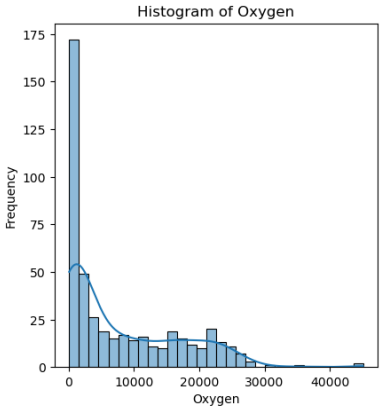
3. Data Visculization

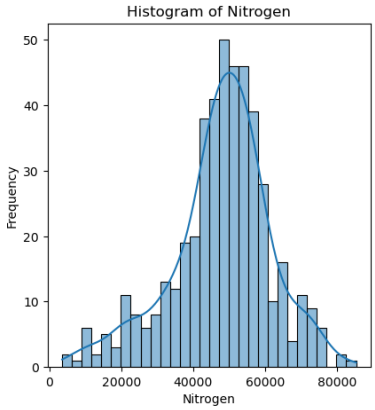
1. Generate Histograms

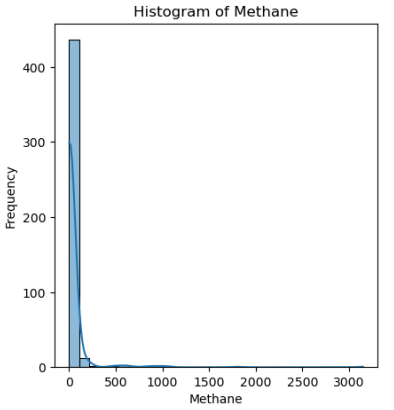
Histograms show the distribution of a single variable. They are useful for understanding the frequency distribution of data.

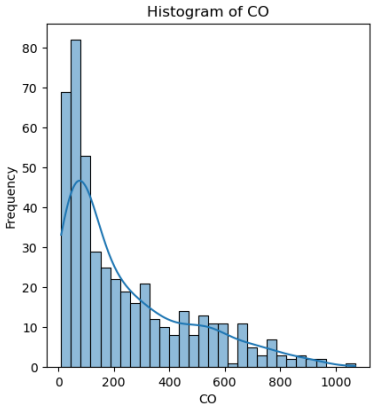
```
# Define the number of rows and columns for the subplot grid
num_cols = len(df.select_dtypes(include='number').columns)
num_rows = (num_cols // 3) + (num_cols % 3 > 0)  # Calculate number of
rows needed

# Create a figure with subplots
fig, axes = plt.subplots(num_rows, 3, figsize=(15, num_rows * 5))
fig.tight_layout(pad=5.0)
axes = axes.flatten()  # Flatten the 2D array of axes for easy
iteration

# Plot histograms
numeric_cols = df.select_dtypes(include='number').columns
for i, col in enumerate(numeric_cols):
    sns.histplot(df[col], kde=True, bins=30, ax=axes[i])
    axes[i].set_title(f'Histogram of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Frequency')

# Hide any unused subplots
for j in range(len(numeric_cols), len(axes)):
    axes[j].set_visible(False)

plt.show()
```

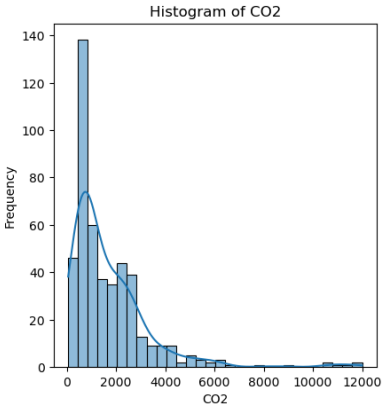

Histogram of Hydrogen
Frequency
Hydrogen

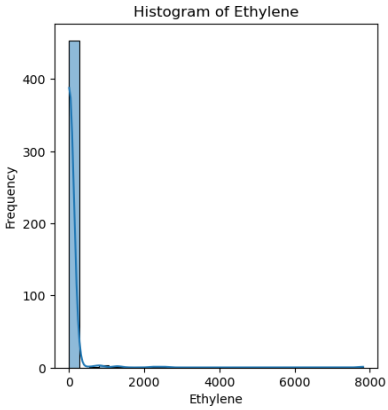

Histogram of Oxygen
Frequency
Oxygen

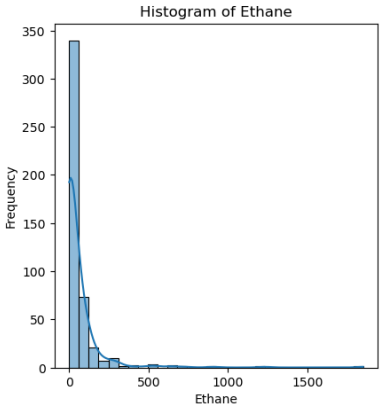

Histogram of Nitrogen
Frequency
Nitrogen

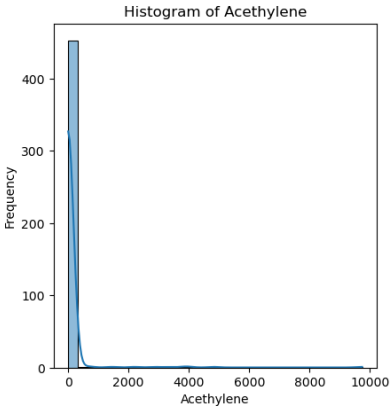

Histogram of Methane
Frequency
Methane

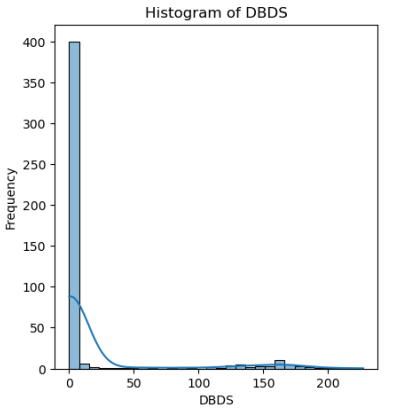

Histogram of CO
Frequency
CO

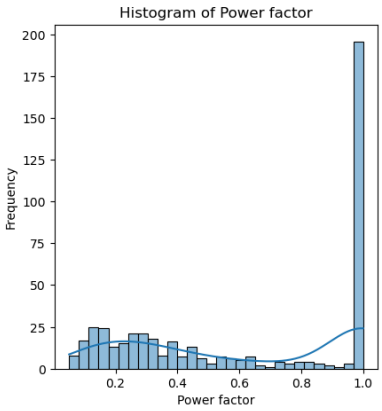

Histogram of CO2
Frequency
CO2

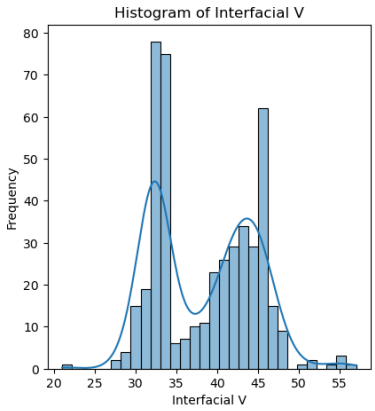

Histogram of Ethylene
Frequency
Ethylene


Histogram of Ethane
Frequency
Ethane


Histogram of Acethylene
Frequency
Acethylene


Histogram of DBDS
Frequency
DBDS

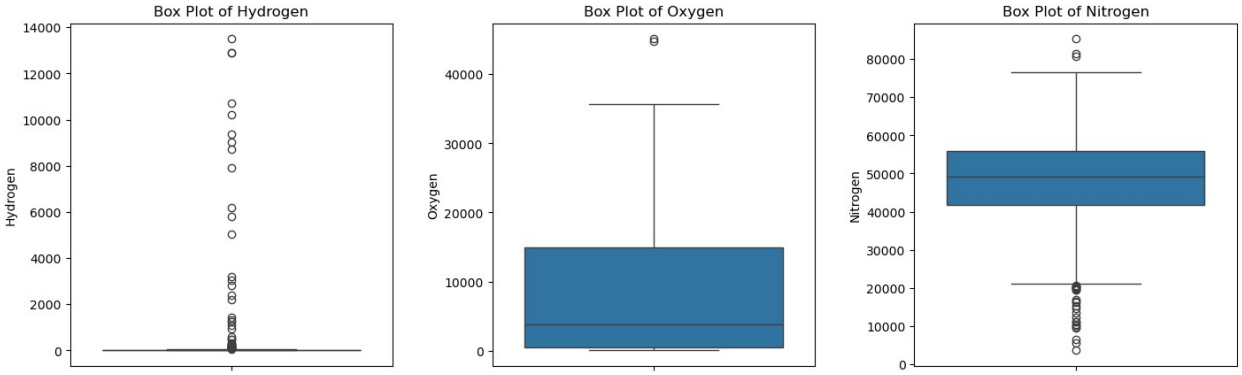

Histogram of Power factor
Frequency
Power factor

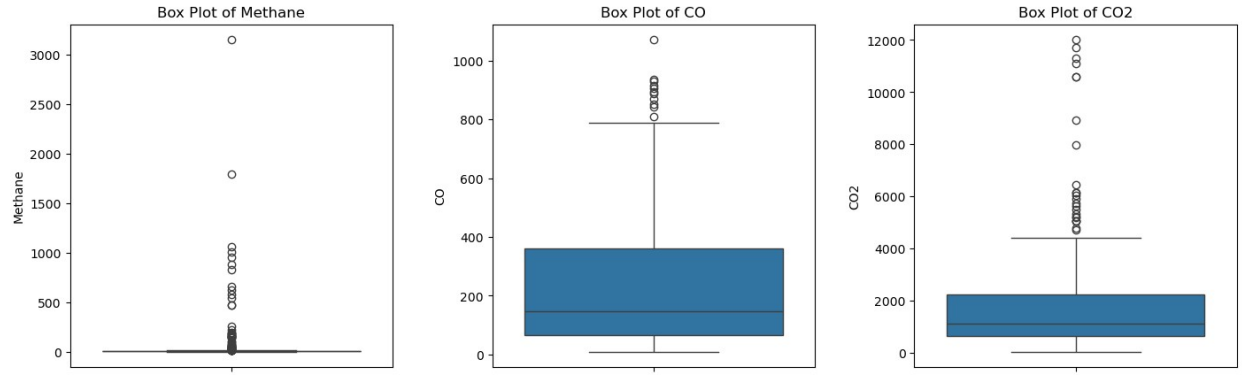

Histogram of Interfacial V
Frequency
Interfacial V

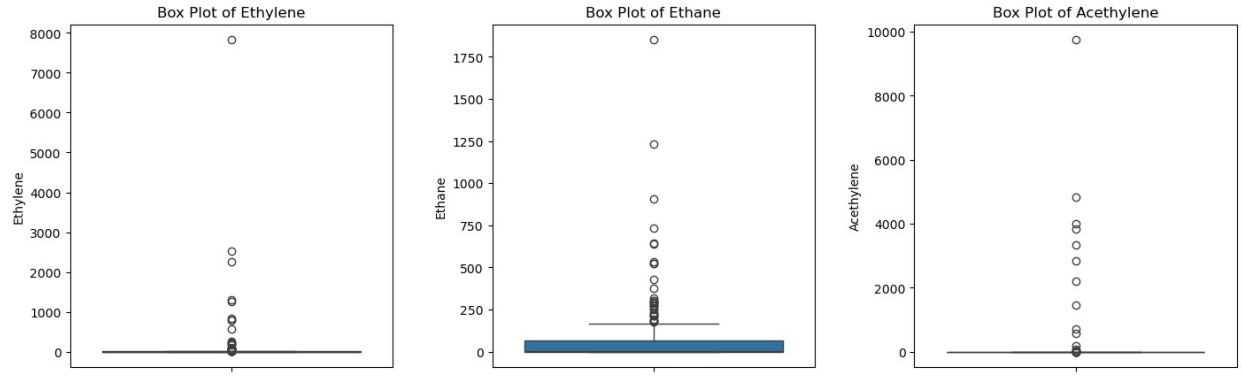
2. Box Plot

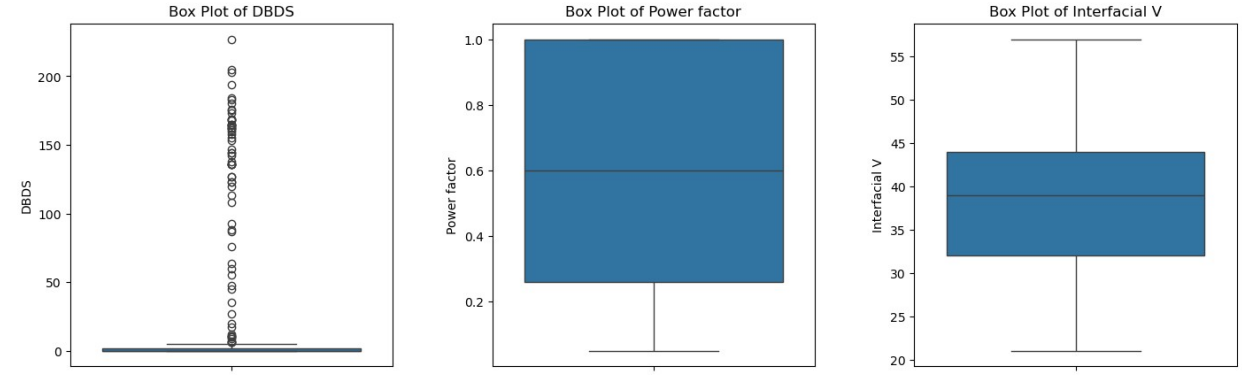
To visualize distributions and detect outliers:

```
# Create a figure with subplots for box plots
fig, axes = plt.subplots(num_rows, 3, figsize=(15, num_rows * 5))
fig.tight_layout(pad=5.0)
axes = axes.flatten()

# Plot box plots
for i, col in enumerate(numeric_cols):
    sns.boxplot(y=df[col], ax=axes[i])
    axes[i].set_title(f'Box Plot of {col}')
    axes[i].set_ylabel(col)

# Hide any unused subplots
for j in range(len(numeric_cols), len(axes)):
    axes[j].set_visible(False)

plt.show()
```


Box Plot of Hydrogen
Hydrogen
Box Plot of Oxygen
Oxygen
Box Plot of Nitrogen
Nitrogen


Box Plot of Methane
Methane
Box Plot of CO
CO
Box Plot of CO2
CO2

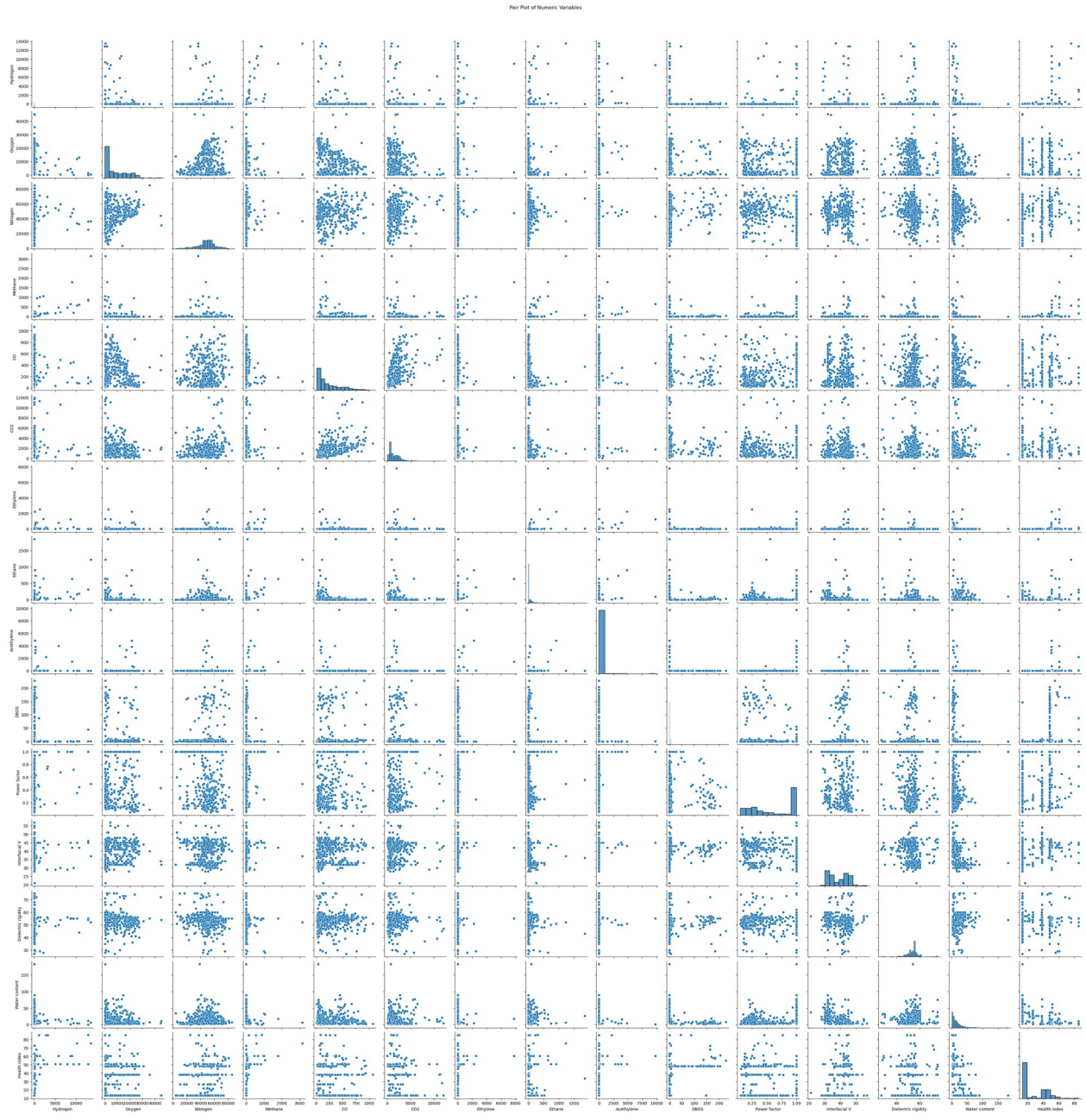

Box Plot of Ethylene
Ethylene
Box Plot of Ethane
Ethane
Box Plot of Acethylene
Acethylene


Box Plot of DBDS
DBDS
Box Plot of Power factor
Power factor
Box Plot of Interfacial V
Interfacial V

3. Pair Plot

For a comprehensive view of pairwise relationships:

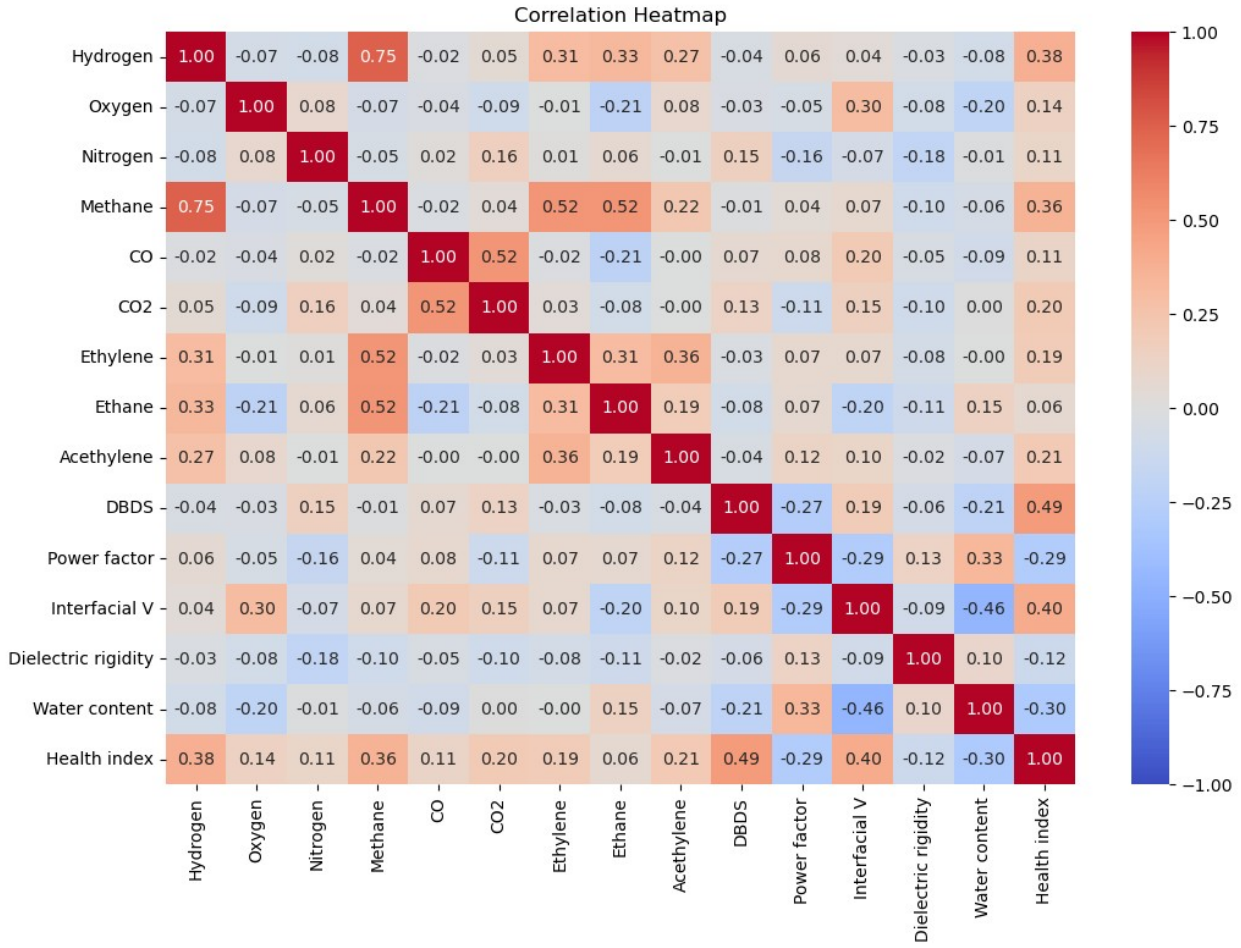
```
# Create a pair plot
sns.pairplot(df[numeric_cols])
plt.suptitle('Pair Plot of Numeric Variables', y=1.02)
plt.show()
```



4. Correlation Heatmap

To visualize the correlations between variables:

```
# Compute the correlation matrix
correlation_matrix = df[numeric_cols].corr()

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f', vmin=-1, vmax=1)
plt.title('Correlation Heatmap')
plt.show()
```



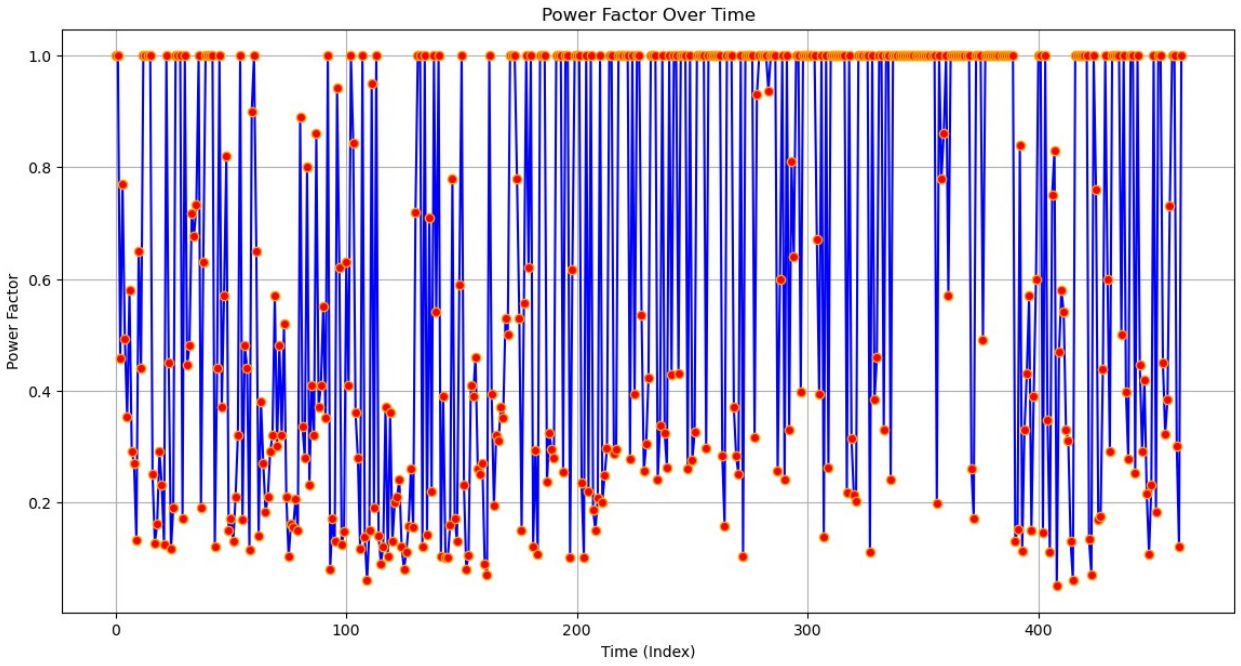
Correlation with Health Index

```
# Compute and display correlation with 'Health index'
health_index_col = 'Health index'
correlations = df.corr()[health_index_col]

print("\nCorrelation with Health Index:")
print(correlations)
```

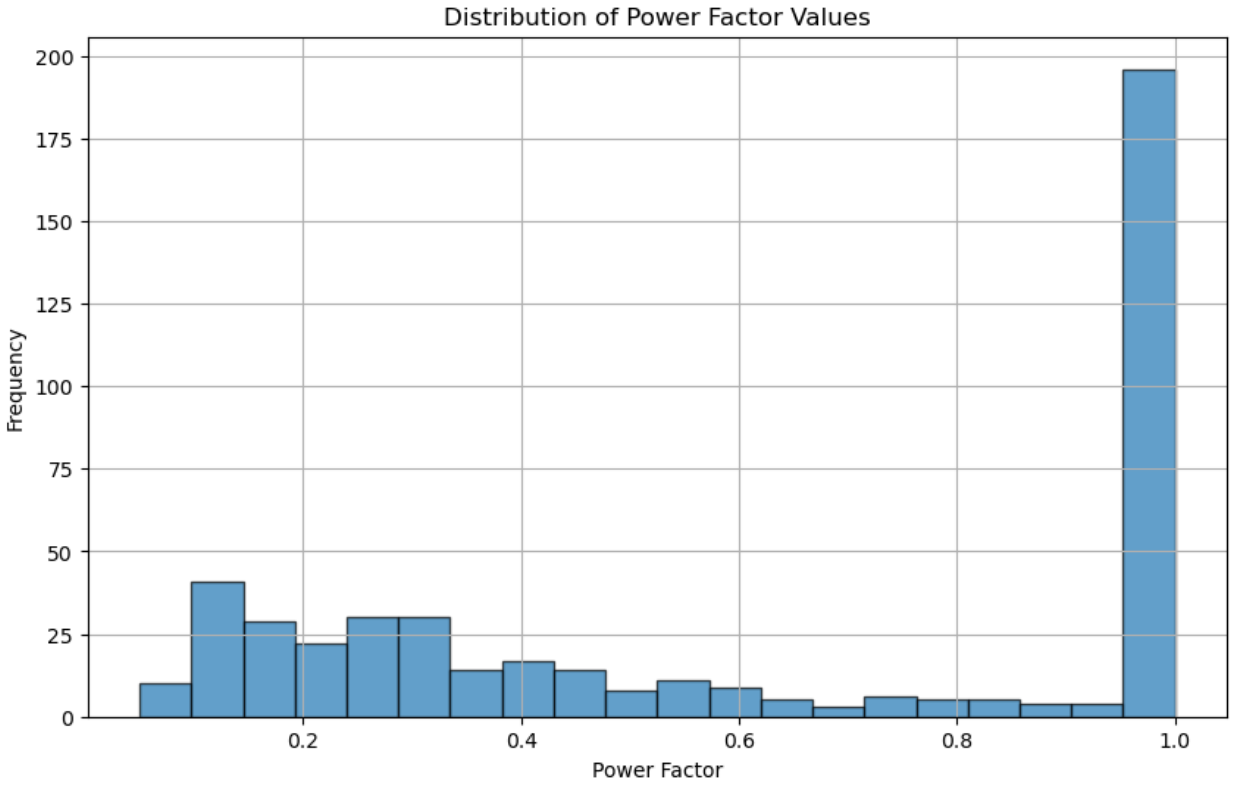
```
Correlation with Health Index:
Hydrogen               0.375171
Oxygen                 0.136489
Nitrogen               0.108785
Methane                0.363219
CO                     0.112956
CO2                    0.203058
Ethylene               0.189952
Ethane                 0.064073
Acethylene             0.210743
DBDS                   0.491709
Power factor          -0.289037
Interfacial V          0.400666
Dielectric rigidity   -0.117358
Water content         -0.300724
Health index           1.000000
Name: Health index, dtype: float64
```

Analysis On Power Factor

```
# Add an index to simulate time
df['Time'] = range(len(df))

# Plotting Time Series Line Chart
plt.figure(figsize=(14, 7))
plt.plot(df['Time'], df['Power factor'], color='b', marker='o',
linestyle='-', mfc='r', mec='orange')

plt.title('Power Factor Over Time')
plt.xlabel('Time (Index)')
plt.ylabel('Power Factor')
plt.grid(True)
plt.show()
```

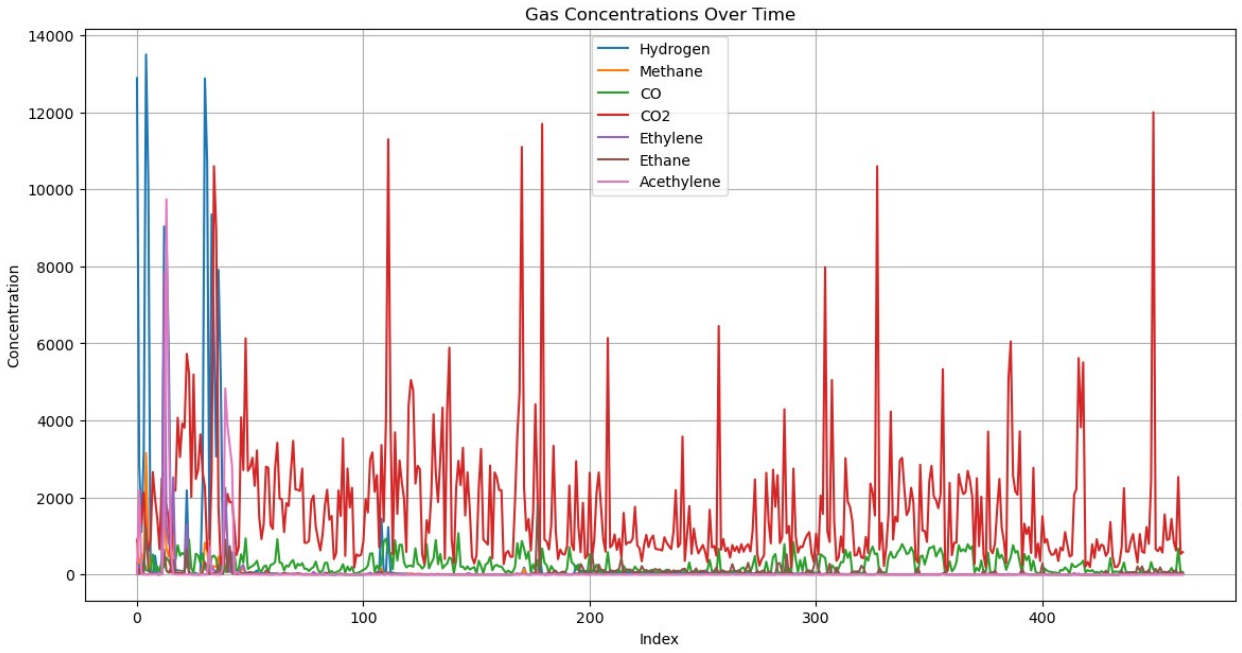



```
# Plotting Histogram
plt.figure(figsize=(10, 6))
plt.hist(df['Power factor'], bins=20, edgecolor='k', alpha=0.7)

plt.title('Distribution of Power Factor Values')
plt.xlabel('Power Factor')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



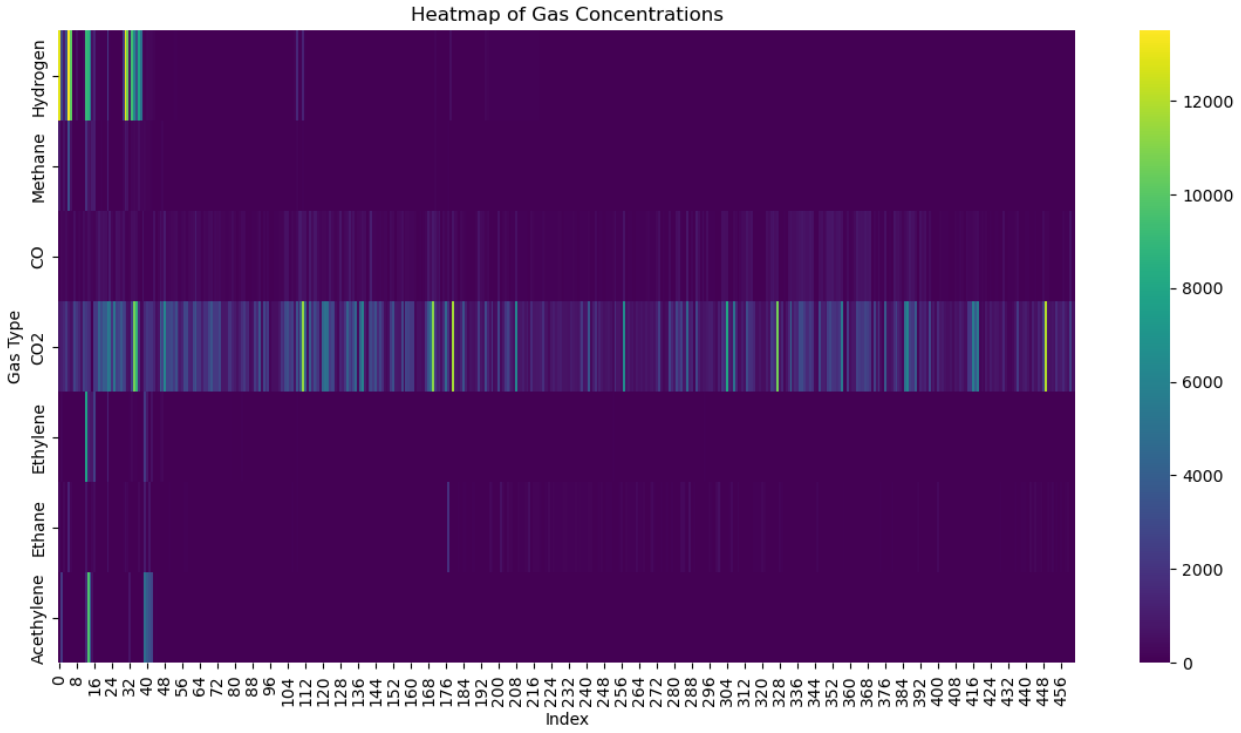
Analysis On Gases

```
# Plotting Time Series Line Charts
plt.figure(figsize=(14, 7))
for column in ['Hydrogen', 'Methane', 'CO', 'CO2', 'Ethylene',
'Ethane', 'Acethylene']:
    plt.plot(df['Time'], df[column], label=column)

plt.title('Gas Concentrations Over Time')
plt.xlabel('Index')
plt.ylabel('Concentration')
plt.legend()
plt.grid(True)
plt.show()
```



Heatmap

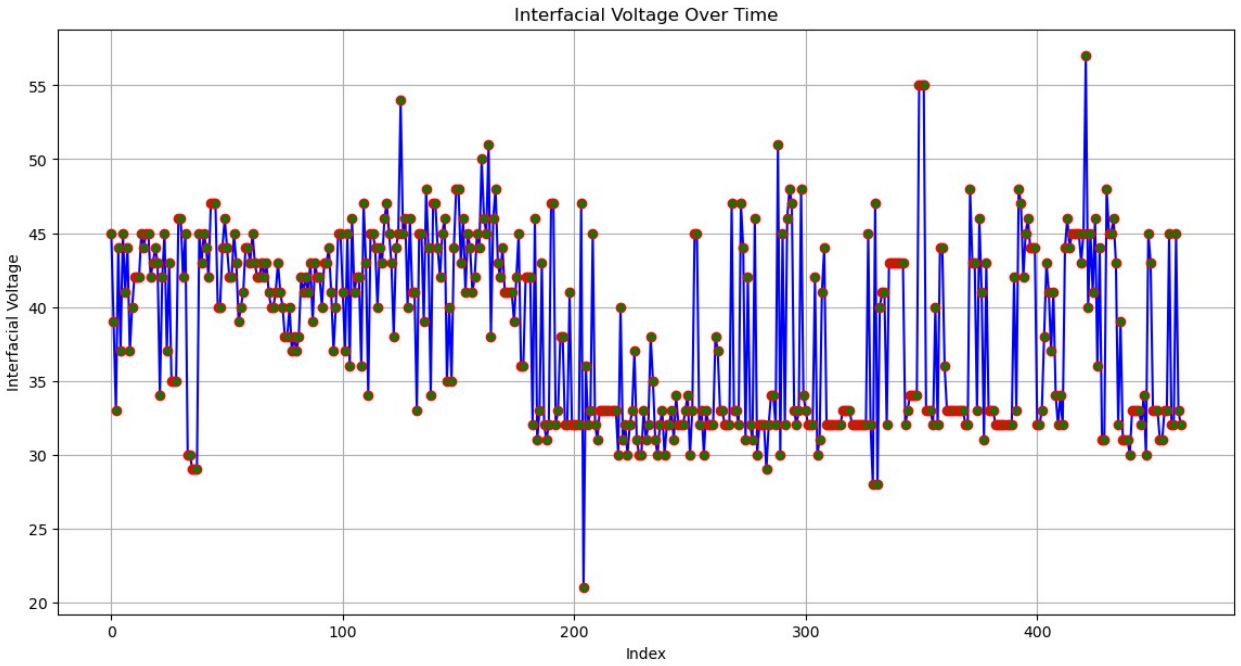
```
# Set the index for the heatmap
heatmap_data = df.set_index('Time')

# Plotting Heatmap
plt.figure(figsize=(14, 7))
sns.heatmap(heatmap_data[['Hydrogen', 'Methane', 'CO', 'CO2',
'Ethylene', 'Ethane', 'Acethylene']].T, cmap='viridis', annot=False)

plt.title('Heatmap of Gas Concentrations')
plt.xlabel('Index')
plt.ylabel('Gas Type')
plt.show()
```



3. Interfacial Voltage

Purpose: Assess the integrity of insulation by measuring voltage between different materials or phases.

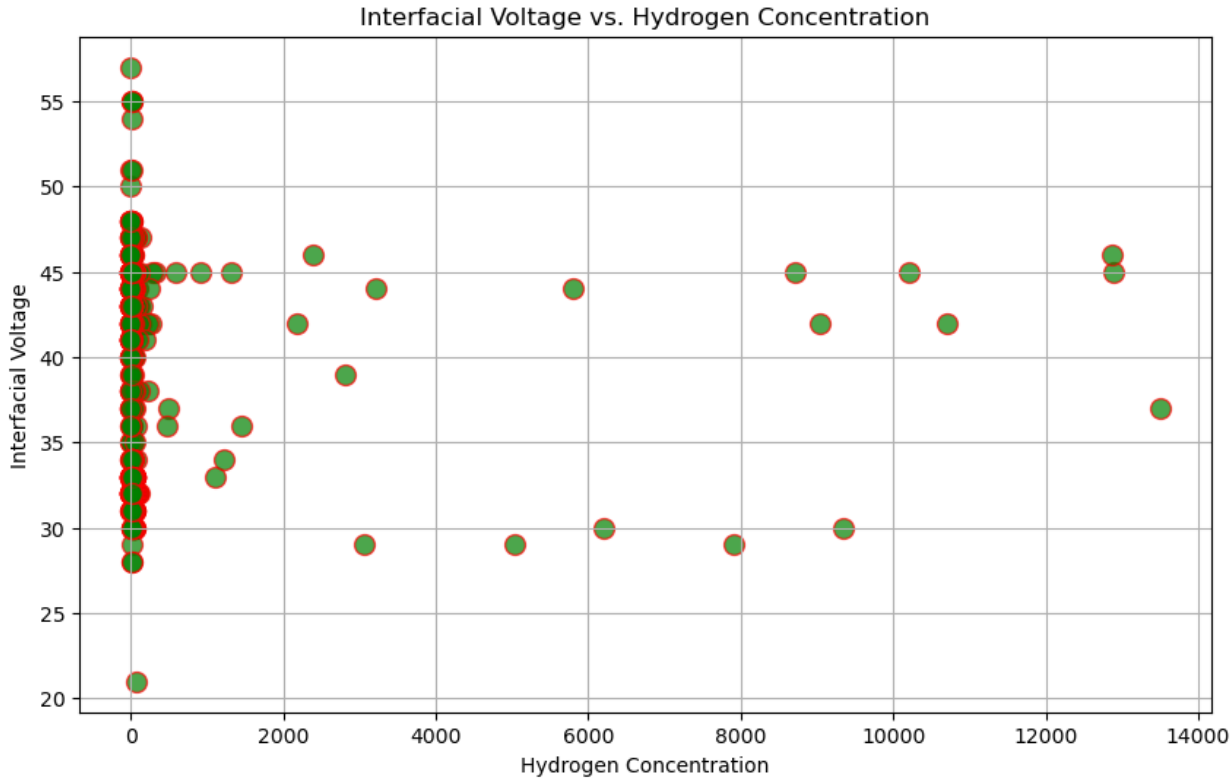
```
# Plotting Line Chart
plt.figure(figsize=(14, 7))
plt.plot(df['Time'], df['Interfacial V'], color='b', marker='o',
linestyle='-', mec='r', mfc='g')

plt.title('Interfacial Voltage Over Time')
plt.xlabel('Index')
plt.ylabel('Interfacial Voltage')
plt.grid(True)
plt.show()
```



Scatter Plot

A scatter plot can be used to show the relationship between interfacial voltage and another variable, such as gas concentrations or time.

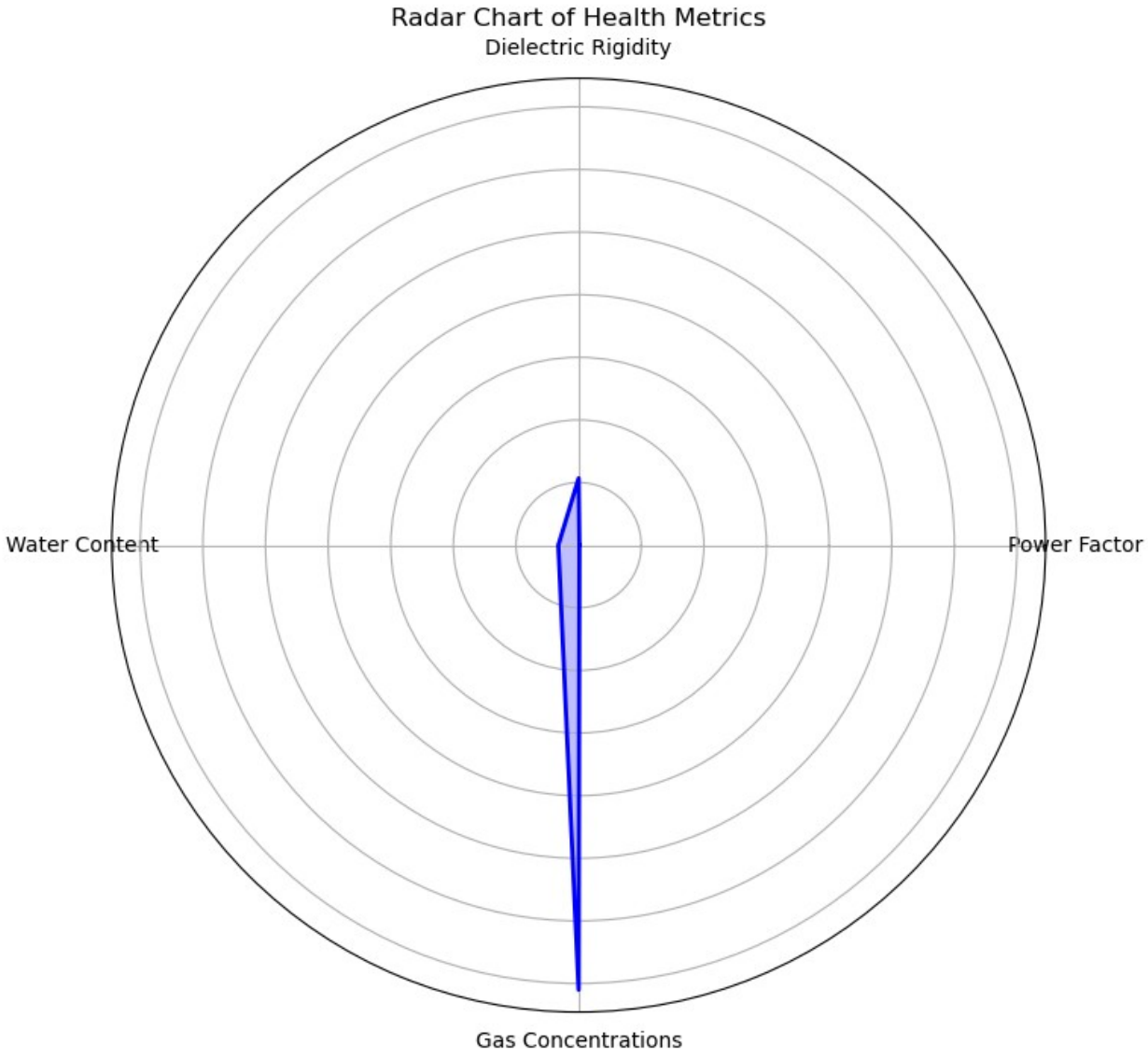
```
# Scatter Plot of Interfacial Voltage vs. Hydrogen Concentration
plt.figure(figsize=(10, 6))
plt.scatter(df['Hydrogen'], df['Interfacial V'], alpha=0.7,
edgecolors='red', s=100, color='g')

plt.title('Interfacial Voltage vs. Hydrogen Concentration')
plt.xlabel('Hydrogen Concentration')
plt.ylabel('Interfacial Voltage')
plt.grid(True)
plt.show()
```



Radar Chart

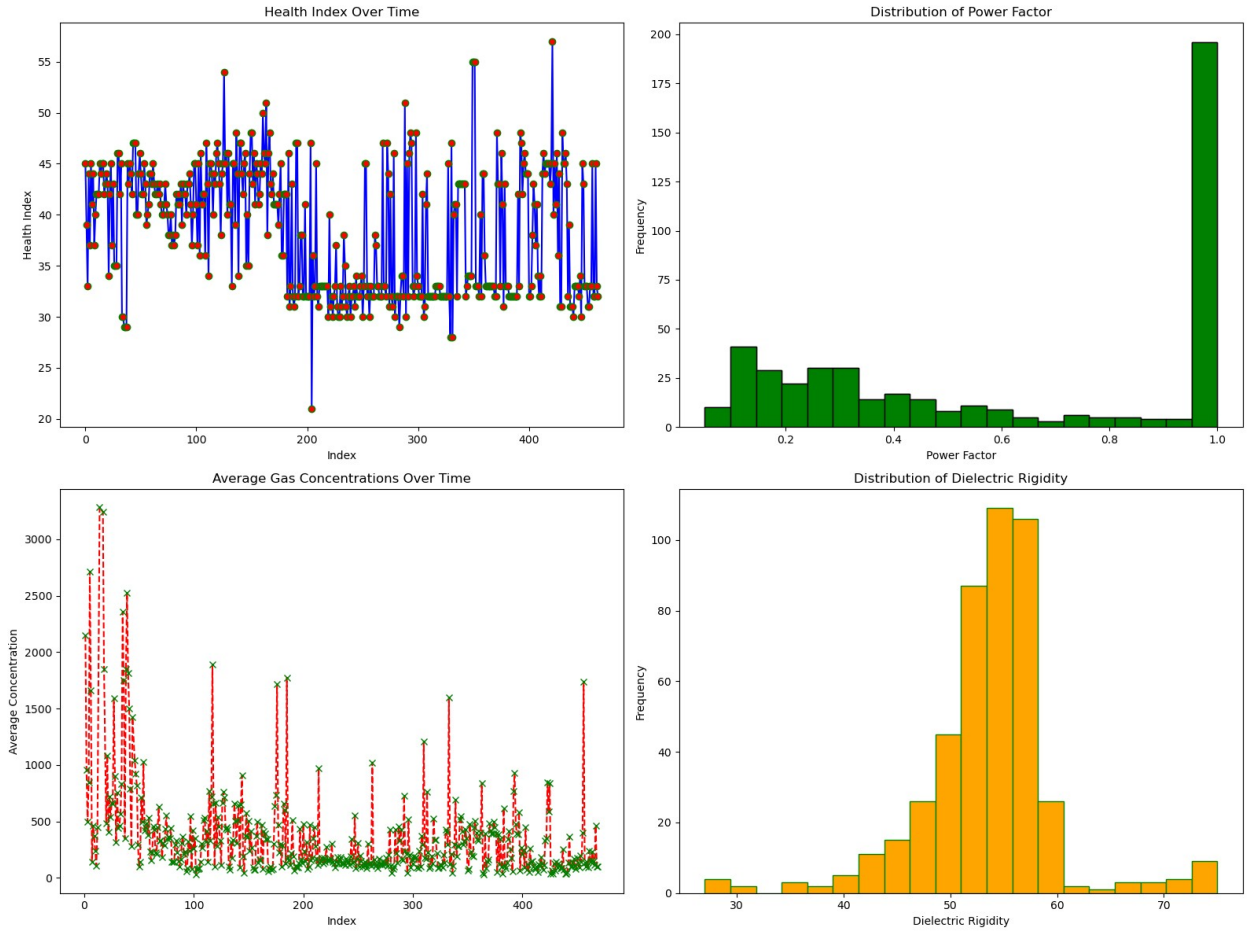
```
metrics = {
    'Power Factor': df['Power factor'].mean(),
    'Dielectric Rigidity': df['Dielectric rigidity'].mean(),
    'Water Content': df['Water content'].mean(),
    'Gas Concentrations': df[['Hydrogen', 'Methane', 'CO', 'CO2',
'Ethylene', 'Ethane', 'Acethylene']].mean().mean()
}

# Create radar chart
labels = list(metrics.keys())
values = list(metrics.values())

# Number of variables
num_vars = len(labels)

# Compute angle for each axis
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
values += values[:1]
angles += angles[:1]

# Plot
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
```

```
ax.fill(angles, values, color='blue', alpha=0.25)
ax.plot(angles, values, color='blue', linewidth=2)

# Labels
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

plt.title('Radar Chart of Health Metrics')
plt.show()
```



Composite Visculization

```
# Create subplots
fig, axs = plt.subplots(2, 2, figsize=(16, 12))

# Plot Health Index Over Time
axs[0, 0].plot(df.Time, df['Interfacial V'], color='b', marker='o',
linestyle='-', mec='g', mfc='r')
axs[0, 0].set_title('Health Index Over Time')
axs[0, 0].set_xlabel('Index')
axs[0, 0].set_ylabel('Health Index')

# Plot Power Factor Distribution
axs[0, 1].hist(df['Power factor'], bins=20, color='g',
edgecolor='black')
axs[0, 1].set_title('Distribution of Power Factor')
axs[0, 1].set_xlabel('Power Factor')
axs[0, 1].set_ylabel('Frequency')

# Plot Gas Concentrations (Time Series)
axs[1, 0].plot(df.index, df[['Hydrogen', 'Methane', 'CO', 'CO2',
'Ethylene', 'Ethane', 'Acethylene']].mean(axis=1), color='r',
marker='x', linestyle='--',  mec='g', mfc='y')
axs[1, 0].set_title('Average Gas Concentrations Over Time')
axs[1, 0].set_xlabel('Index')
axs[1, 0].set_ylabel('Average Concentration')


# Plot Power Factor Distribution
axs[0, 1].hist(df['Power factor'], bins=20, color='g',
edgecolor='black')
axs[0, 1].set_title('Distribution of Power Factor')
axs[0, 1].set_xlabel('Power Factor')
axs[0, 1].set_ylabel('Frequency')

# Plot Dielectric Rigidity Distribution
axs[1, 1].hist(df['Dielectric rigidity'], bins=20, color='orange',
edgecolor='green')
axs[1, 1].set_title('Distribution of Dielectric Rigidity')
axs[1, 1].set_xlabel('Dielectric Rigidity')
axs[1, 1].set_ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.show()
```



4. Coorelation Analysis

Calculate the correlation matrix

```
correlation_matrix = df.corr()
correlation_matrix
```

	Hydrogen	Oxygen	Nitrogen	Methane	CO
Hydrogen	1.000000	-0.070666	-0.080532	0.748477	-0.018310
Oxygen	-0.070666	1.000000	0.084253	-0.066420	-0.036966
Nitrogen	-0.080532	0.084253	1.000000	-0.047046	0.015651
Methane	0.748477	-0.066420	-0.047046	1.000000	-0.023716
CO	-0.018310	-0.036966	0.015651	-0.023716	1.000000

```
CO2                    0.054322 -0.093978  0.156402  0.038255  0.522475
Ethylene               0.306348 -0.007308  0.012322  0.519412 -0.020173
Ethane                 0.326903 -0.214976  0.063460  0.519188 -0.211410
Acethylene             0.272670  0.082992 -0.005095  0.224547 -0.001666
DBDS                  -0.041495 -0.034141  0.153792 -0.014614  0.068180
Power factor           0.058620 -0.052476 -0.158950  0.040296  0.076231
Interfacial V          0.035000  0.300856 -0.066980  0.071919  0.198140
Dielectric rigidity -0.027914 -0.083509 -0.179902 -0.100654 -0.045053
Water content       -0.080184 -0.202987 -0.009357 -0.056009 -0.092964
Health index           0.375171  0.136489  0.108785  0.363219  0.112956
Time                  -0.294169 -0.002765 -0.151205 -0.276770 -0.093551


                            CO2  Ethylene    Ethane  Acethylene
DBDS  \
Hydrogen               0.054322  0.306348  0.326903    0.272670 -
0.041495
Oxygen                -0.093978 -0.007308 -0.214976    0.082992 -
0.034141
Nitrogen               0.156402  0.012322  0.063460   -0.005095
0.153792
Methane                0.038255  0.519412  0.519188    0.224547 -
0.014614
CO                     0.522475 -0.020173 -0.211410   -0.001666
0.068180
CO2                    1.000000  0.033419 -0.083847   -0.004338
0.127624
Ethylene               0.033419  1.000000  0.307136    0.362919 -
0.028445
Ethane                -0.083847  0.307136  1.000000    0.186270 -
0.080207
Acethylene            -0.004338  0.362919  0.186270    1.000000 -
0.042619
DBDS                   0.127624 -0.028445 -0.080207   -0.042619
1.000000
Power factor          -0.111914  0.070803  0.067379    0.119832 -
0.269423
Interfacial V          0.148753  0.074667 -0.197281    0.104032
0.187818
```

```
Dielectric rigidity -0.096483 -0.077143 -0.105402   -0.018223 -
0.064454
Water content        0.004457 -0.001735  0.145577   -0.069651 -
0.208736
Health index         0.203058  0.189952  0.064073    0.210743
0.491709
Time                -0.194906 -0.167780 -0.097785   -0.184144 -
0.437329

                     Power factor  Interfacial V  Dielectric rigidity
\
Hydrogen                 0.058620       0.035000            -0.027914

Oxygen                  -0.052476       0.300856            -0.083509

Nitrogen                -0.158950      -0.066980            -0.179902

Methane                  0.040296       0.071919            -0.100654

CO                       0.076231       0.198140            -0.045053

CO2                     -0.111914       0.148753            -0.096483

Ethylene                 0.070803       0.074667            -0.077143

Ethane                   0.067379      -0.197281            -0.105402

Acethylene               0.119832       0.104032            -0.018223

DBDS                    -0.269423       0.187818            -0.064454

Power factor             1.000000      -0.287076             0.128064

Interfacial V           -0.287076       1.000000            -0.091980

Dielectric rigidity      0.128064      -0.091980             1.000000

Water content            0.333518      -0.456099             0.101549

Health index            -0.289037       0.400666            -0.117358

Time                     0.260432      -0.286377             0.087210


                    Water content  Health index      Time
Hydrogen                -0.080184      0.375171 -0.294169
Oxygen                  -0.202987      0.136489 -0.002765
Nitrogen                -0.009357      0.108785 -0.151205
Methane                 -0.056009      0.363219 -0.276770
CO                      -0.092964      0.112956 -0.093551
CO2                      0.004457      0.203058 -0.194906
```

```
Ethylene                -0.001735      0.189952 -0.167780
Ethane                   0.145577      0.064073 -0.097785
Acethylene              -0.069651      0.210743 -0.184144
DBDS                    -0.208736      0.491709 -0.437329
Power factor             0.333518     -0.289037  0.260432
Interfacial V           -0.456099      0.400666 -0.286377
Dielectric rigidity      0.101549     -0.117358  0.087210
Water content            1.000000     -0.300724  0.224829
Health index            -0.300724      1.000000 -0.886173
Time                     0.224829     -0.886173  1.000000
```

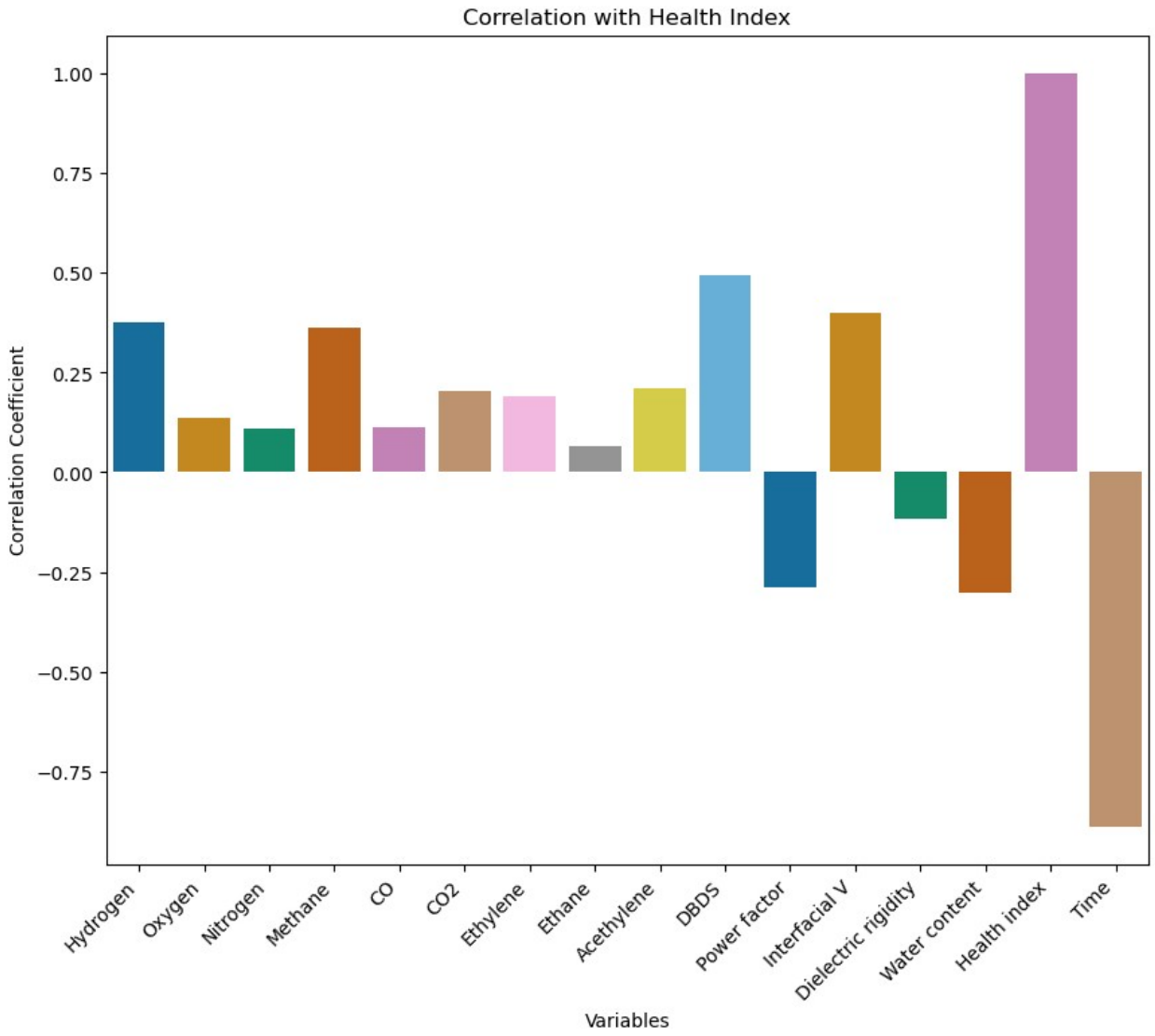
Extract correlation with `Health index`

```
health_index_corr = correlation_matrix['Health index']
health_index_corr
```

```
Hydrogen                0.375171
Oxygen                  0.136489
Nitrogen                0.108785
Methane                 0.363219
CO                      0.112956
CO2                     0.203058
Ethylene                0.189952
Ethane                  0.064073
Acethylene              0.210743
DBDS                    0.491709
Power factor           -0.289037
Interfacial V           0.400666
Dielectric rigidity    -0.117358
Water content          -0.300724
Health index            1.000000
Time                   -0.886173
Name: Health index, dtype: float64
```

Plot

```
# Set up the figure for correlation with Health index
plt.figure(figsize=(10, 8))

# Create a bar plot for correlations with Health index
sns.barplot(x=health_index_corr.index, y=health_index_corr.values,
hue=health_index_corr.index,palette='colorblind', legend=False)
plt.xticks(rotation=45, ha='right')
plt.title('Correlation with Health Index')
plt.xlabel('Variables')
plt.ylabel('Correlation Coefficient')
plt.show()
```




```
df.head()
```

	Hydrogen	Oxygen	Nitrogen	Methane	CO	CO2	Ethylene	Ethane	\
1	12886	61	25041	877	83	864	4	305	
2	2820	16400	56300	144	257	1080	206	11	
3	1099	70	37520	545	184	1402	6	230	
4	3210	3570	47900	160	360	2130	4	43	
5	13500	343	36500	3150	113	984	5	1230	

	Acethylene	DBDS	Power factor	Interfacial V	Dielectric rigidity	\
1	0	45.0	1.000	45	55	
2	2190	1.0	1.000	39	52	

```
3           0  87.0         0.458              33                      49

4           4   1.0         0.770              44                      55

5           1   1.0         0.493              37                      52


   Water content  Health index  Time
1              0          85.5     0
2             11          85.3     1
3              5          85.3     2
4              3          85.2     3
5              6          75.6     4
```

As Time is not from our dataset, so remove it permanantly from df

```
df.drop(columns=['Time'], errors='ignore', inplace=True)  # Safely
drop the column if it exists

df.columns
```

```
Index(['Hydrogen', 'Oxygen', 'Nitrogen', 'Methane', 'CO', 'CO2',
'Ethylene',
       'Ethane', 'Acethylene', 'DBDS', 'Power factor', 'Interfacial
V',
       'Dielectric rigidity', 'Water content', 'Health index'],
      dtype='object')
```

5. Feature Importance

a. Univariate Feature Selection

```
# Define features and target variable
X = df.drop(columns=['Health index'])
y = df['Health index']



# Perform univariate feature selection using F-statistic
f_values, p_values = f_regression(X, y)



# Create a DataFrame to view the results
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'F-Statistic': f_values,
    'p-Value': p_values
}).sort_values(by='F-Statistic', ascending=False)
```

```
feature_importance
```

	Feature	F-Statistic	p-Value
9	DBDS	147.001318	1.473409e-29
11	Interfacial V	88.158134	2.786795e-19
0	Hydrogen	75.516320	6.371945e-17
3	Methane	70.061859	6.931598e-16
13	Water content	45.835581	3.925658e-11
10	Power factor	42.023960	2.322899e-10
8	Acethylene	21.425748	4.788333e-06
5	CO2	19.825685	1.065497e-05
6	Ethylene	17.256315	3.892963e-05
1	Oxygen	8.751062	3.253327e-03
12	Dielectric rigidity	6.437966	1.149913e-02
4	CO	5.957915	1.502671e-02
2	Nitrogen	5.520912	1.921138e-02
7	Ethane	1.900382	1.687042e-01

b. Machine Learning Techniques

```
# Train a Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X, y)
```

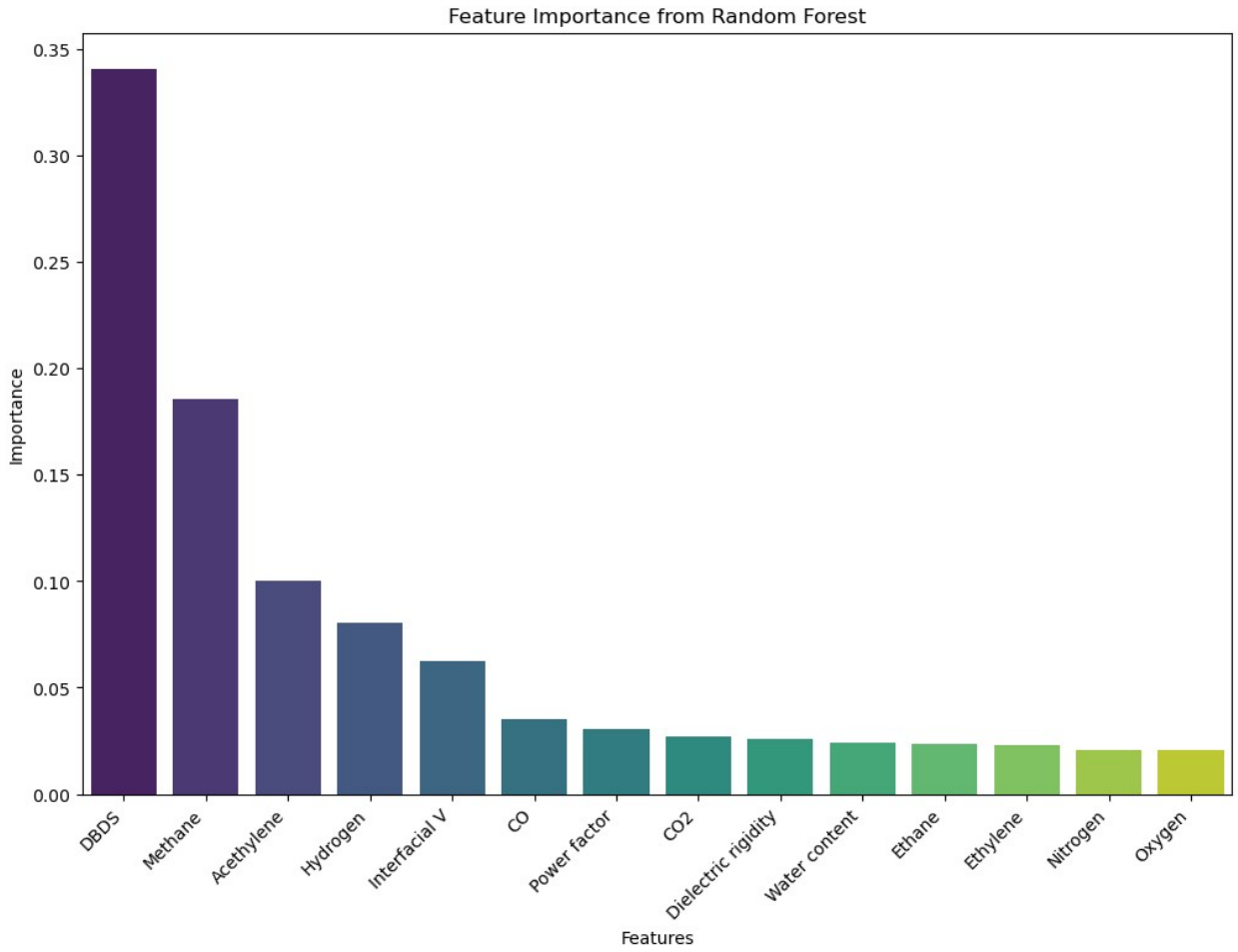
```
RandomForestRegressor(random_state=42)
```

```
# Extract feature importances
importances = model.feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

importances
```

```
array([0.08068513, 0.02057668, 0.02090363, 0.18538283, 0.03528976,
       0.02695304, 0.02291341, 0.02360264, 0.09990571, 0.34057451,
       0.03070734, 0.06231217, 0.02580481, 0.02438835])
```

```
# Plot feature importances
plt.figure(figsize=(12, 8))
sns.barplot(x=feature_importance_df['Feature'],
y=feature_importance_df['Importance'], palette='viridis',
hue=feature_importance_df['Feature'])
plt.xticks(rotation=45, ha='right')
plt.title('Feature Importance from Random Forest')
```

```
plt.xlabel('Features')
plt.ylabel('Importance')
plt.show()
```



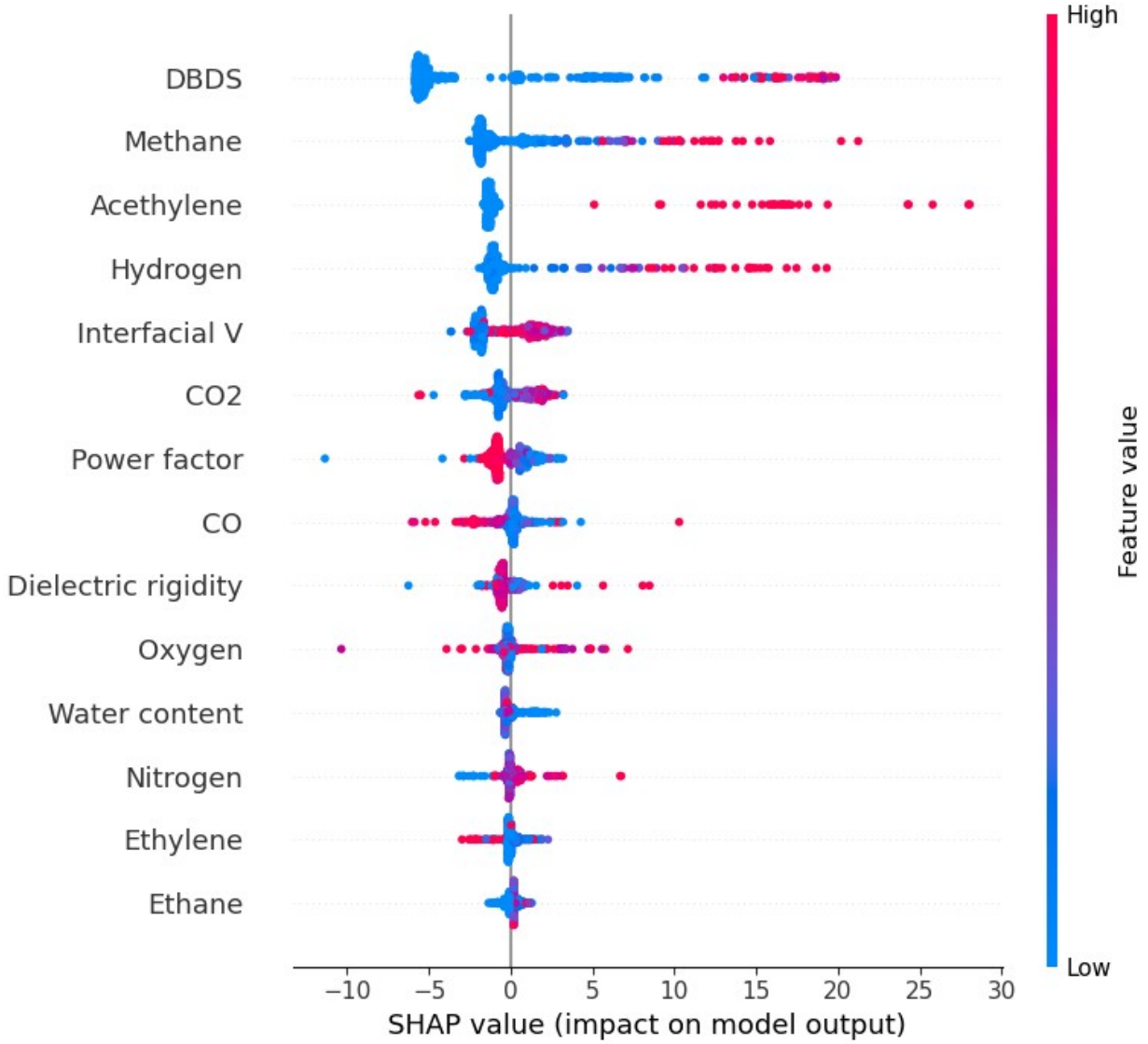
c. Feature Importance with SHAP Values

```
# Train a Gradient Boosting Regressor
model = GradientBoostingRegressor(n_estimators=100, random_state=42)
model.fit(X, y)
```

```
GradientBoostingRegressor(random_state=42)
```

```
# Compute SHAP values
explainer = shap.Explainer(model, X)
shap_values = explainer(X)
```

```
# Plot SHAP values for feature importance
shap.summary_plot(shap_values, X)
```



6. Anomoly Detection

1. Statististical Method

Z-Score

```
# Calculate Z-scores for each feature
z_scores = df.apply(zscore)

# Set a threshold for identifying anomalies
threshold = 3

# Identify anomalies
anomalies = (z_scores.abs() > threshold).any(axis=1)

df[anomalies]
```

```
     Hydrogen  Oxygen  Nitrogen  Methane   CO    CO2  Ethylene  Ethane
\
1       12886      61     25041      877   83    864         4     305

2        2820   16400     56300      144  257   1080       206      11

3        1099      70     37520      545  184   1402         6     230

4        3210    3570     47900      160  360   2130         4      43

5       13500     343     36500     3150  113    984         5    1230

..        ...     ...       ...      ...  ...    ...       ...     ...

395         3   45100     31200        5  568   2160         3       2

414         0   16443     43308        1   10    348         0       0

425         0    9444     35331        3  358   5507         6       0

428         0    4330     11200        0   81    192         0       0

456        16    1050     51600       15  122  12000         0      21


     Acethylene  DBDS  Power factor  Interfacial V  Dielectric
rigidity  \
1             0  45.0         1.000             45
55
2          2190   1.0         1.000             39
52
3             0  87.0         0.458             33
49
4             4   1.0         0.770             44
55
5             1   1.0         0.493             37
52
..          ...   ...           ...            ...                  ..
.
395           0   0.0         1.000             32
```

```
54
414           0   0.0         0.830              41
73
425           0   0.0         1.000              45
75
428           0   0.0         1.000              57
57
456           0   0.0         0.230              43
47

     Water content  Health index
1                0          85.5
2               11          85.3
3                5          85.3
4                3          85.2
5                6          75.6
..             ...           ...
395             11          13.4
414              5          13.4
425             75          13.4
428              4          13.4
456              5          13.4

[87 rows x 15 columns]
```

b. Machine Learning Methods

i. Isolation Forest:

- The Isolation Forest algorithm is effective for anomaly detection in high-dimensional datasets.

```
# Train Isolation Forest
iso_forest = IsolationForest(contamination=0.01, random_state=42)  #
Adjust contamination rate if needed

y_pred = iso_forest.fit_predict(X)

# Identify anomalies
anomalies_if = y_pred == -1
df[anomalies_if]

    Hydrogen  Oxygen  Nitrogen  Methane   CO   CO2  Ethylene
Ethane  \
5      13500     343     36500     3150  113   984         5     1230

14      9040    1870     47500     1790  183  2060      7820      638

17      8710    4530     42800      658  437  1780      1260       97
```

```
27      2183     192     43380     1061  183  5730      1308     646
44       152   21400     49300      254   64  1510      2250     908

    Acethylene  DBDS  Power factor  Interfacial V  Dielectric rigidity
\
5            1   1.0         0.493             37                   52
14        1450   0.0         1.000             42                   55
17        9740   1.0         1.000             45                   55
27           2   0.0         1.000             42                   28
44        4830   0.0         1.000             43                   51

    Water content  Health index
5               6          75.6
14             18          60.5
17              1          60.5
27             31          55.8
44              5          50.7
```

ii. One-Class SVM

- The One-Class SVM method is also used for anomaly detection in datasets.

```
# Train One-Class SVM
oc_svm = OneClassSVM(gamma='auto', nu=0.01)  # Adjust nu parameter
based on expected anomaly rate

y_pred = oc_svm.fit_predict(X)

# Identify anomalies
anomalies_svm = y_pred == -1

df[anomalies_svm]
```

```
     Hydrogen  Oxygen  Nitrogen  Methane   CO    CO2  Ethylene  Ethane
\
5       13500     343     36500     3150  113    984         5    1230
6       10200   11900     33700      573   87    611         0     162
7           3   15459     41347        5   68    902        12       2
8          16    2470     59600        8  520   2660         5       8
```

```
9          488   11861     48353        13   85   1957          29       23
..         ...     ...       ...       ...  ...    ...         ...      ...
437          0   16000     48000         2  426   1360           0        0
438          0   23900     45900         0   49    466           0        0
455         17     564     14600         5  320   2450           0        0
456         16    1050     51600        15  122  12000           0       21
465         15     227     52900         3   60    853           3       84

     Acethylene   DBDS  Power factor  Interfacial V  Dielectric
rigidity  \
5              1    1.0         0.493             37
52
6              0    1.0         0.353             45
55
7             13    5.0         0.580             41
71
8              2  164.0         0.290             44
56
9              0  164.0         0.270             37
72
..           ...    ...           ...            ...                   .
..
437            0    5.0         0.600             48
54
438            0    5.0         0.290             45
53
455            0    1.0         0.107             45
52
456            0    0.0         0.230             43
47
465            0    0.0         1.000             32
56

     Water content  Health index
5                6          75.6
6                5          75.6
7                6          73.2
8                4          72.8
9               10          68.0
..             ...           ...
437              4          13.4
438              4          13.4
```

```
455               6           13.4
456               5           13.4
465              28           13.4

[346 rows x 15 columns]
```