🚄 Metro Train Dataset

It describe the in depth EDA of very famous Metro Train Dataset.


METRO
TESTED
METRO

!

Metro Engineering Equipment Company, Coimbatore, Tamil Nadu, India is an established company that was founded in the year of 1970. We offer avant-garde quality Air compressors and Spares for industries and Borewell Compressors for Agriculture Purposes.

It describe Air Pressure in the Metro Engine EDA of very famous Metro Train dataset

Metro Train dataset describe the Air Pressure status of individual compressor in Engine.

EDA is performed by using following Python Libraries

- Numpy
- Pandas
- Matplotlib
- Seaborn

Installing Necessary Libraries

```
# pip install numpy

# pip install pandas

# pip install matplotlib

# pip install seaborn

# pip install pandas-profiling
```

Importing Necessary Python Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# Ignore warnings
warnings.filterwarnings("ignore")
```

```
# Magic command to display plots inline in Jupyter Notebook
%matplotlib inline

# inline is a arguments

df = "/kaggle/input/metro-train-dataset/MetroPT3(AirCompressor).csv"
df = pd.read_csv(df)

df.head()
```

```
   Unnamed: 0            timestamp    TP2    TP3     H1
DV_pressure  \
0           0  2020-02-01 00:00:00 -0.012  9.358  9.340       -0.024

1          10  2020-02-01 00:00:10 -0.014  9.348  9.332       -0.022

2          20  2020-02-01 00:00:19 -0.012  9.338  9.322       -0.022

3          30  2020-02-01 00:00:29 -0.012  9.328  9.312       -0.022

4          40  2020-02-01 00:00:39 -0.012  9.318  9.302       -0.022


   Reservoirs  Oil_temperature  Motor_current  COMP  DV_eletric
Towers  MPG  \
0       9.358           53.600         0.0400   1.0         0.0
1.0  1.0
1       9.348           53.675         0.0400   1.0         0.0
1.0  1.0
2       9.338           53.600         0.0425   1.0         0.0
1.0  1.0
3       9.328           53.425         0.0400   1.0         0.0
1.0  1.0
4       9.318           53.475         0.0400   1.0         0.0
1.0  1.0

   LPS  Pressure_switch  Oil_level  Caudal_impulses
0  0.0              1.0        1.0              1.0
1  0.0              1.0        1.0              1.0
2  0.0              1.0        1.0              1.0
3  0.0              1.0        1.0              1.0
4  0.0              1.0        1.0              1.0
```

Showing Data Type is Present in Data_Set :

```
# Get the data types of the columns

df.dtypes
```

```
Unnamed: 0           int64
timestamp           object
```

```
TP2                 float64
TP3                 float64
H1                  float64
DV_pressure         float64
Reservoirs          float64
Oil_temperature     float64
Motor_current       float64
COMP                float64
DV_eletric          float64
Towers              float64
MPG                 float64
LPS                 float64
Pressure_switch     float64
Oil_level           float64
Caudal_impulses     float64
dtype: object
```

Converting string dates to 'Date_Time' and set the date column as index :

```
# Import data, convert string dates to 'datetime64' and set the date
column as index:

df =
pd.read_csv('/kaggle/input/metro-train-dataset/MetroPT3(AirCompressor)
.csv',
                 parse_dates=['timestamp'],
                 infer_datetime_format=True,
                 index_col='timestamp',
                 thousands=',',
                 decimal='.'
                )
```

Review of Missing Values and Dtypes :

```
#  Review the general info on data, paying attention to missing values
and dtypes

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1516948 entries, 2020-02-01 00:00:00 to 2020-09-01
03:59:50
Data columns (total 16 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   Unnamed: 0       1516948 non-null  int64
 1   TP2              1516948 non-null  float64
```

```
 2   TP3              1516948 non-null  float64
 3   H1               1516948 non-null  float64
 4   DV_pressure      1516948 non-null  float64
 5   Reservoirs       1516948 non-null  float64
 6   Oil_temperature  1516948 non-null  float64
 7   Motor_current    1516948 non-null  float64
 8   COMP             1516948 non-null  float64
 9   DV_eletric       1516948 non-null  float64
 10  Towers           1516948 non-null  float64
 11  MPG              1516948 non-null  float64
 12  LPS              1516948 non-null  float64
 13  Pressure_switch  1516948 non-null  float64
 14  Oil_level        1516948 non-null  float64
 15  Caudal_impulses  1516948 non-null  float64
dtypes: float64(15), int64(1)
memory usage: 196.7 MB

df.describe()

         Unnamed: 0           TP2           TP3            H1
DV_pressure  \
count  1.516948e+06  1.516948e+06  1.516948e+06  1.516948e+06
1.516948e+06
mean   7.584735e+06  1.367826e+00  8.984611e+00  7.568155e+00
5.595619e-02
std    4.379053e+06  3.250930e+00  6.390951e-01  3.333200e+00
3.824015e-01
min    0.000000e+00 -3.200000e-02  7.300000e-01 -3.600000e-02 -
3.200000e-02
25%    3.792368e+06 -1.400000e-02  8.492000e+00  8.254000e+00 -
2.200000e-02
50%    7.584735e+06 -1.200000e-02  8.960000e+00  8.784000e+00 -
2.000000e-02
75%    1.137710e+07 -1.000000e-02  9.492000e+00  9.374000e+00 -
1.800000e-02
max    1.516947e+07  1.067600e+01  1.030200e+01  1.028800e+01
9.844000e+00

         Reservoirs  Oil_temperature  Motor_current          COMP  \
count  1.516948e+06     1.516948e+06   1.516948e+06  1.516948e+06
mean   8.985233e+00     6.264418e+01   2.050171e+00  8.369568e-01
std    6.383070e-01     6.516261e+00   2.302053e+00  3.694052e-01
min    7.120000e-01     1.540000e+01   2.000000e-02  0.000000e+00
25%    8.494000e+00     5.777500e+01   4.000000e-02  1.000000e+00
50%    8.960000e+00     6.270000e+01   4.500000e-02  1.000000e+00
75%    9.492000e+00     6.725000e+01   3.807500e+00  1.000000e+00
max    1.030000e+01     8.905000e+01   9.295000e+00  1.000000e+00

         DV_eletric        Towers           MPG           LPS  \
count  1.516948e+06  1.516948e+06  1.516948e+06  1.516948e+06
```

```
mean   1.606106e-01  9.198483e-01  8.326640e-01  3.420025e-03
std    3.671716e-01  2.715280e-01  3.732757e-01  5.838091e-02
min    0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
25%    0.000000e+00  1.000000e+00  1.000000e+00  0.000000e+00
50%    0.000000e+00  1.000000e+00  1.000000e+00  0.000000e+00
75%    0.000000e+00  1.000000e+00  1.000000e+00  0.000000e+00
max    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

       Pressure_switch     Oil_level  Caudal_impulses
count     1.516948e+06  1.516948e+06     1.516948e+06
mean      9.914368e-01  9.041556e-01     9.371066e-01
std       9.214078e-02  2.943779e-01     2.427712e-01
min       0.000000e+00  0.000000e+00     0.000000e+00
25%       1.000000e+00  1.000000e+00     1.000000e+00
50%       1.000000e+00  1.000000e+00     1.000000e+00
75%       1.000000e+00  1.000000e+00     1.000000e+00
max       1.000000e+00  1.000000e+00     1.000000e+00
```

Checking whether there is missing values :

```python
# Check for missing values
df.isnull().sum()
```

```
Unnamed: 0          0
TP2                 0
TP3                 0
H1                  0
DV_pressure         0
Reservoirs          0
Oil_temperature     0
Motor_current       0
COMP                0
DV_eletric          0
Towers              0
MPG                 0
LPS                 0
Pressure_switch     0
Oil_level           0
Caudal_impulses     0
dtype: int64
```

Removing the empty Columns :

```python
# Let's remove the empty column and look at some examples of data:
df = df.drop(columns='Unnamed: 0')
print(f'data shape = {df.shape}')
df.head()
```

```
data shape = (1516948, 15)
```

```
                        TP2    TP3     H1  DV_pressure  Reservoirs  \
timestamp
2020-02-01 00:00:00 -0.012  9.358  9.340       -0.024       9.358
2020-02-01 00:00:10 -0.014  9.348  9.332       -0.022       9.348
2020-02-01 00:00:19 -0.012  9.338  9.322       -0.022       9.338
2020-02-01 00:00:29 -0.012  9.328  9.312       -0.022       9.328
2020-02-01 00:00:39 -0.012  9.318  9.302       -0.022       9.318

                     Oil_temperature  Motor_current  COMP  DV_eletric
Towers  \
timestamp

2020-02-01 00:00:00           53.600         0.0400   1.0         0.0
1.0
2020-02-01 00:00:10           53.675         0.0400   1.0         0.0
1.0
2020-02-01 00:00:19           53.600         0.0425   1.0         0.0
1.0
2020-02-01 00:00:29           53.425         0.0400   1.0         0.0
1.0
2020-02-01 00:00:39           53.475         0.0400   1.0         0.0
1.0

                     MPG  LPS  Pressure_switch  Oil_level
Caudal_impulses
timestamp

2020-02-01 00:00:00  1.0  0.0              1.0        1.0
1.0
2020-02-01 00:00:10  1.0  0.0              1.0        1.0
1.0
2020-02-01 00:00:19  1.0  0.0              1.0        1.0
1.0
2020-02-01 00:00:29  1.0  0.0              1.0        1.0
1.0
2020-02-01 00:00:39  1.0  0.0              1.0        1.0
1.0

# df.describe().transpose()

# The transpose() method is used to swap the rows and columns of the
DataFrame

df.describe().transpose()

                     count       mean       std     min     25%
50%  \
TP2              1516948.0   1.367826  3.250930  -0.032  -0.014  -
0.012
TP3              1516948.0   8.984611  0.639095   0.730   8.492
```

```
8.960
H1                1516948.0   7.568155  3.333200  -0.036   8.254
8.784
DV_pressure       1516948.0   0.055956  0.382402  -0.032  -0.022  -
0.020
Reservoirs        1516948.0   8.985233  0.638307   0.712   8.494
8.960
Oil_temperature   1516948.0  62.644182  6.516261  15.400  57.775
62.700
Motor_current     1516948.0   2.050171  2.302053   0.020   0.040
0.045
COMP              1516948.0   0.836957  0.369405   0.000   1.000
1.000
DV_eletric        1516948.0   0.160611  0.367172   0.000   0.000
0.000
Towers            1516948.0   0.919848  0.271528   0.000   1.000
1.000
MPG               1516948.0   0.832664  0.373276   0.000   1.000
1.000
LPS               1516948.0   0.003420  0.058381   0.000   0.000
0.000
Pressure_switch   1516948.0   0.991437  0.092141   0.000   1.000
1.000
Oil_level         1516948.0   0.904156  0.294378   0.000   1.000
1.000
Caudal_impulses   1516948.0   0.937107  0.242771   0.000   1.000
1.000

                       75%     max
TP2                -0.0100  10.676
TP3                 9.4920  10.302
H1                  9.3740  10.288
DV_pressure        -0.0180   9.844
Reservoirs          9.4920  10.300
Oil_temperature    67.2500  89.050
Motor_current       3.8075   9.295
COMP                1.0000   1.000
DV_eletric          0.0000   1.000
Towers              1.0000   1.000
MPG                 1.0000   1.000
LPS                 0.0000   1.000
Pressure_switch     1.0000   1.000
Oil_level           1.0000   1.000
Caudal_impulses     1.0000   1.000
```

Correlations of Matrix's :

```
# Compute the correlation matrix
```

```
df.corr()

# Display the correlation matrix_corr
```

```
                      TP2       TP3        H1  DV_pressure  Reservoirs
\
TP2              1.000000 -0.011161 -0.961269     0.415025   -0.012403
TP3             -0.011161  1.000000  0.224867    -0.153074    0.999993
H1              -0.961269  0.224867  1.000000    -0.425513    0.226037
DV_pressure      0.415025 -0.153074 -0.425513     1.000000   -0.153080
Reservoirs      -0.012403  0.999993  0.226037    -0.153080    1.000000
Oil_temperature  0.250710  0.401616 -0.161810     0.339697    0.401647
Motor_current    0.697480  0.413756 -0.600178     0.302160    0.412691
COMP            -0.955521  0.103295  0.971419    -0.423992    0.104509
DV_eletric       0.947396 -0.078428 -0.958663     0.427813   -0.079640
Towers          -0.616405  0.064937  0.628964    -0.285256    0.065729
MPG             -0.941250  0.088343  0.954307    -0.417284    0.089555
LPS              0.057651 -0.324556 -0.133178     0.011928   -0.325235
Pressure_switch -0.069532  0.025240  0.064769    -0.088285    0.025352
Oil_level        0.013033 -0.032829 -0.020691     0.058520   -0.032832
Caudal_impulses -0.010397 -0.053179 -0.001900     0.042388   -0.052300

                 Oil_temperature  Motor_current      COMP  DV_eletric
\
TP2                     0.250710       0.697480 -0.955521    0.947396
TP3                     0.401616       0.413756  0.103295   -0.078428
H1                     -0.161810      -0.600178  0.971419   -0.958663
DV_pressure             0.339697       0.302160 -0.423992    0.427813
Reservoirs              0.401647       0.412691  0.104509   -0.079640
Oil_temperature         1.000000       0.528739 -0.233677    0.241678
Motor_current           0.528739       1.000000 -0.681326    0.689828
```

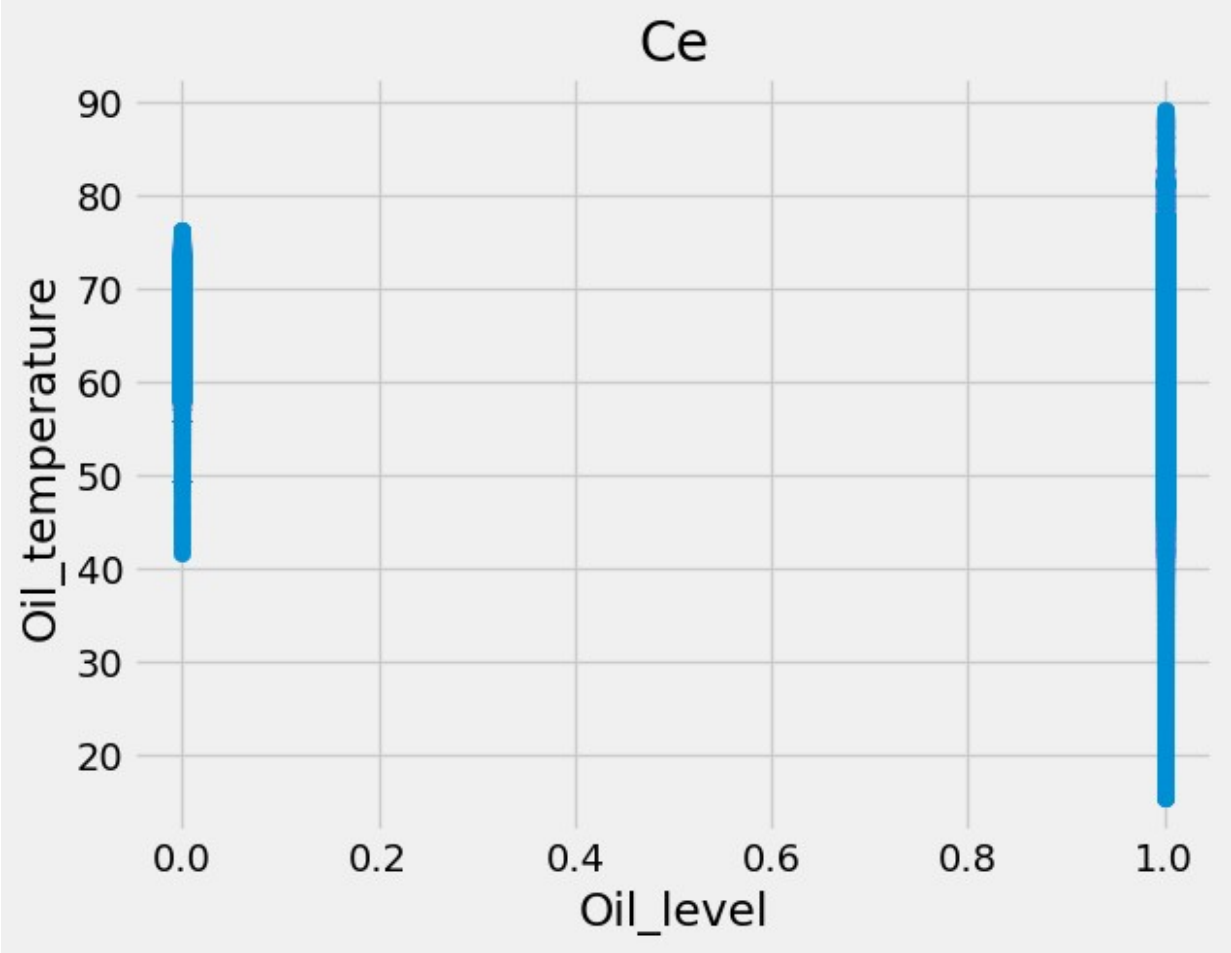
```
COMP                -0.233677      -0.681326  1.000000   -0.959307
DV_eletric           0.241678       0.689828 -0.959307    1.000000
Towers              -0.154531      -0.439421  0.668804   -0.607211
MPG                 -0.239154      -0.683543  0.984555   -0.975641
LPS                 -0.057989       0.053664 -0.132727    0.133922
Pressure_switch     -0.019560      -0.047003  0.183160    0.014289
Oil_level           -0.145239      -0.032793  0.017974    0.002478
Caudal_impulses     -0.068120      -0.051145  0.050725   -0.025202

                    Towers       MPG       LPS  Pressure_switch
Oil_level  \
TP2              -0.616405 -0.941250  0.057651        -0.069532
0.013033
TP3               0.064937  0.088343 -0.324556         0.025240  -
0.032829
H1                0.628964  0.954307 -0.133178         0.064769  -
0.020691
DV_pressure      -0.285256 -0.417284  0.011928        -0.088285
0.058520
Reservoirs        0.065729  0.089555 -0.325235         0.025352  -
0.032832
Oil_temperature -0.154531 -0.239154 -0.057989        -0.019560  -
0.145239
Motor_current    -0.439421 -0.683543  0.053664        -0.047003  -
0.032793
COMP              0.668804  0.984555 -0.132727         0.183160
0.017974
DV_eletric       -0.607211 -0.975641  0.133922         0.014289
0.002478
Towers            1.000000  0.658474 -0.092452         0.262536
0.056029
MPG               0.658474  1.000000 -0.130677         0.180250
0.053032
LPS              -0.092452 -0.130677  1.000000         0.003729
0.016465
Pressure_switch   0.262536  0.180250  0.003729         1.000000
0.223812
Oil_level         0.056029  0.053032  0.016465         0.223812
1.000000
Caudal_impulses   0.087214  0.094497  0.015176         0.291253
0.070259
```

```
                  Caudal_impulses
TP2                     -0.010397
TP3                     -0.053179
H1                      -0.001900
DV_pressure              0.042388
Reservoirs              -0.052300
Oil_temperature         -0.068120
Motor_current           -0.051145
COMP                     0.050725
DV_eletric              -0.025202
Towers                   0.087214
MPG                      0.094497
LPS                      0.015176
Pressure_switch          0.291253
Oil_level                0.070259
Caudal_impulses          1.000000
```

" Visualization into different Plots "

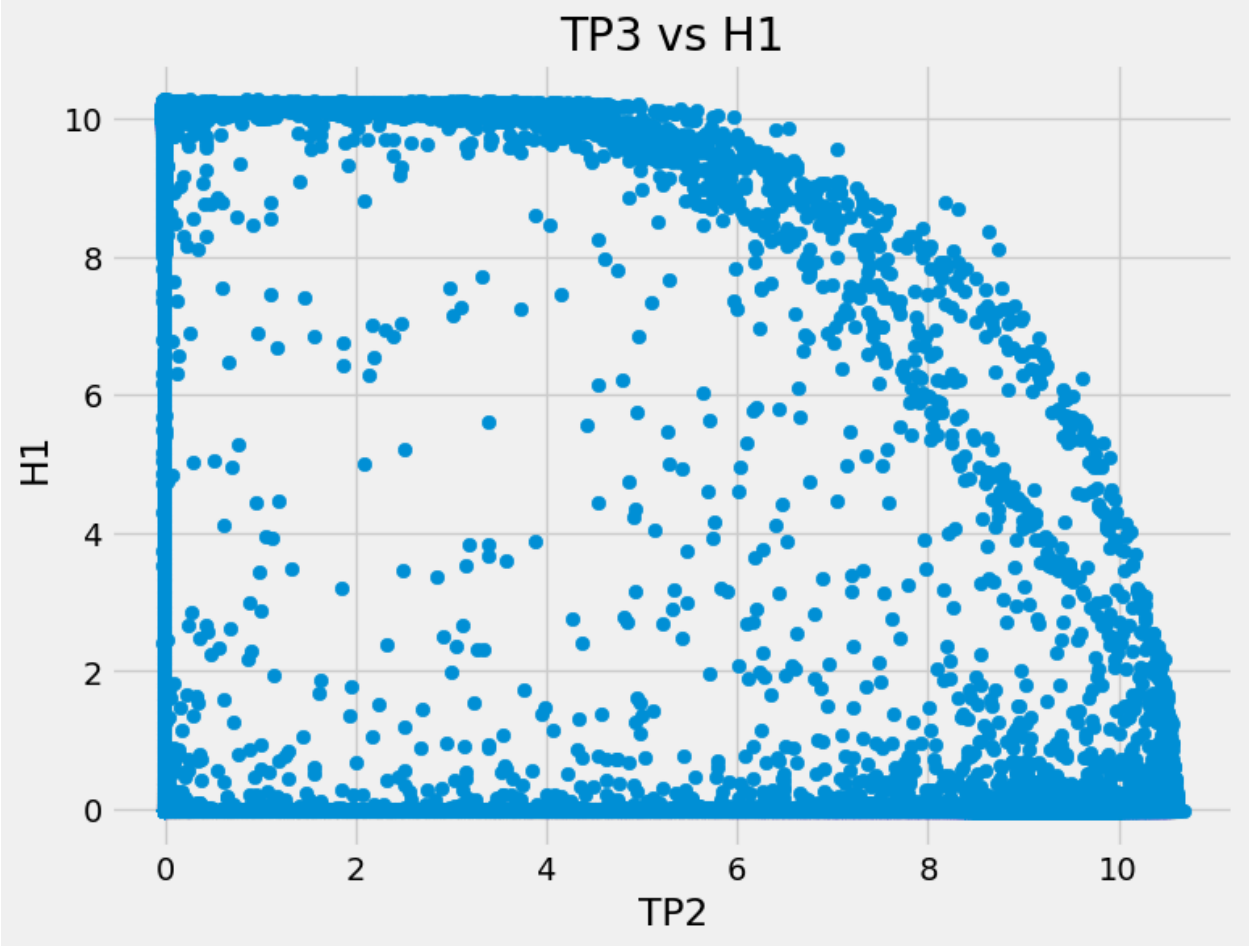
Scatter Plot : " Oil Level vs Oil Temperature "

```
# Scatter plot

plt.scatter(df['Oil_level'], df['Oil_temperature'])
plt.xlabel('Oil_level')
plt.ylabel('Oil_temperature')
plt.title('Ce')
plt.show()
```



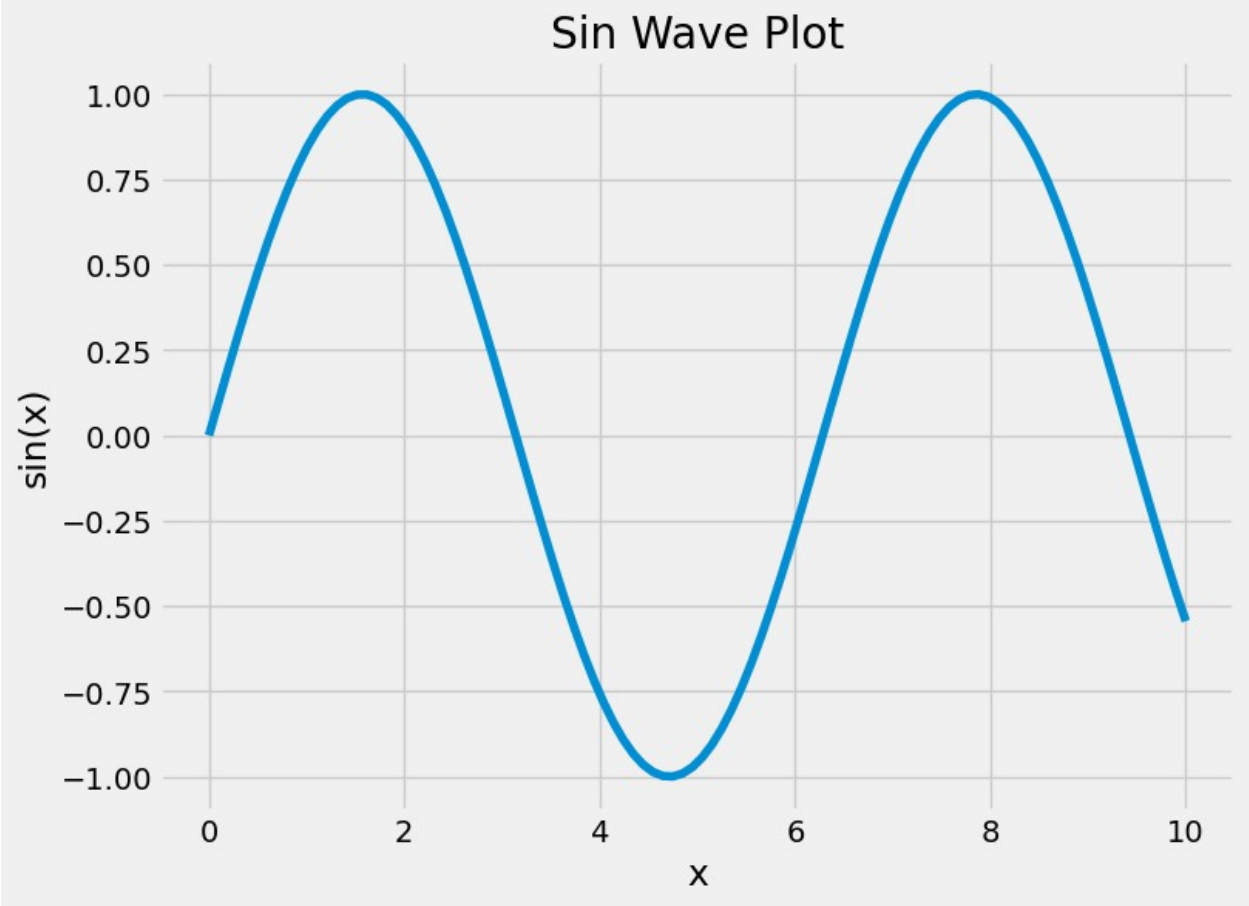
Scatter PLot Analysis TP3 Vs H1 :

```
# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(df['TP2'], df['H1'])
plt.xlabel('TP2')
plt.ylabel('H1')
plt.title('TP3 vs H1')
plt.show()
```



Visualization in form of Sine_Wave_Form :

```
# Sin Wave Plot
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.figure(figsize=(8, 6))
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('Sin Wave Plot')
plt.show()
```



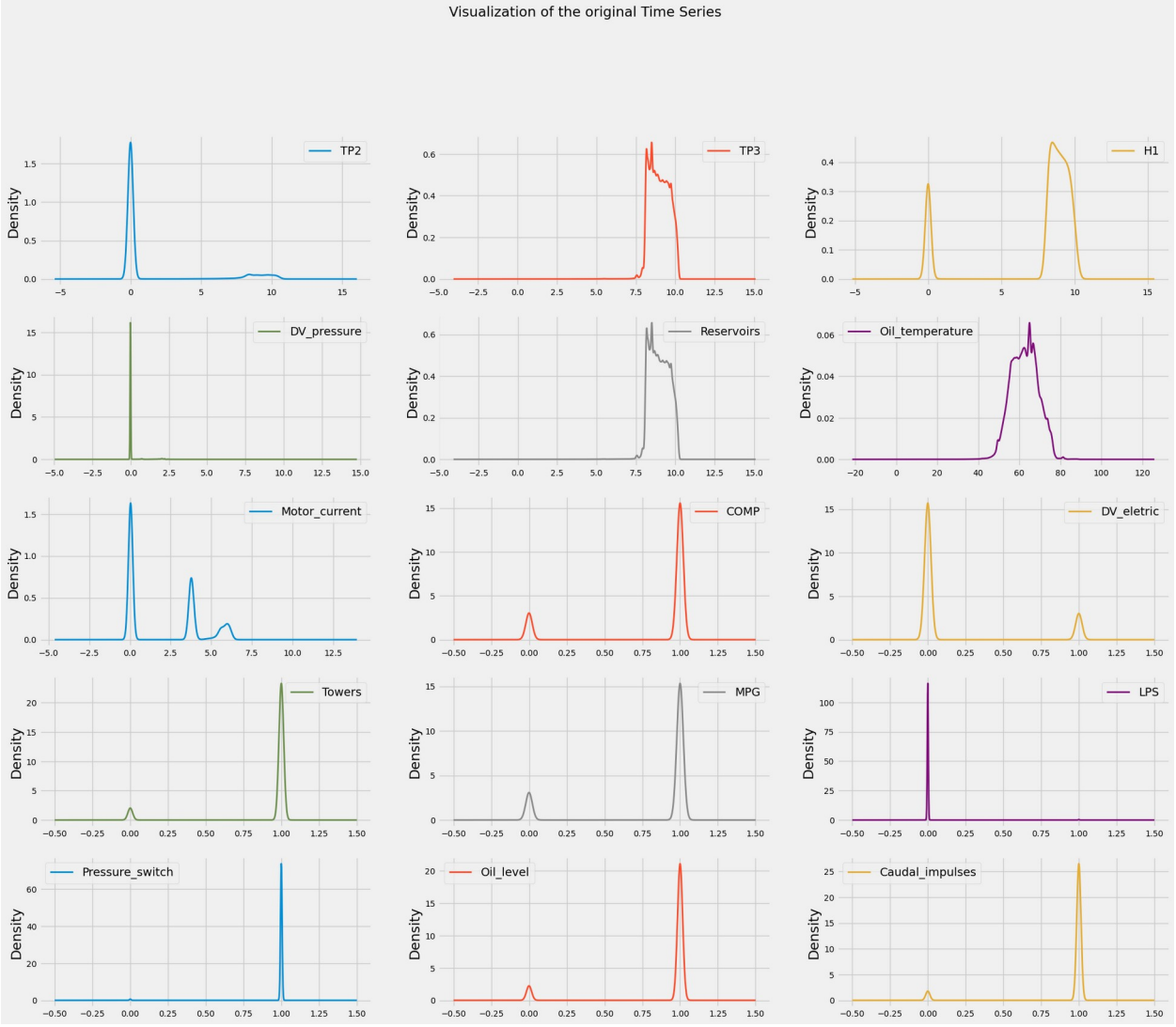
Visualization of the original Time Series :

```
# Plot the time series
plt.style.use('fivethirtyeight')
df.plot(subplots=True,
        layout=(6, 3),
        figsize=(22,22),
        fontsize=10,
        linewidth=1,
        sharex=False,
        title='Visualization of the original Time Series')
plt.show()

# the sharex parameter is used to control the sharing of the x-axis
between subplots.
# It determines whether the subplots in a grid share the same x-axis
scale or
# have independent x-axis scales.
```

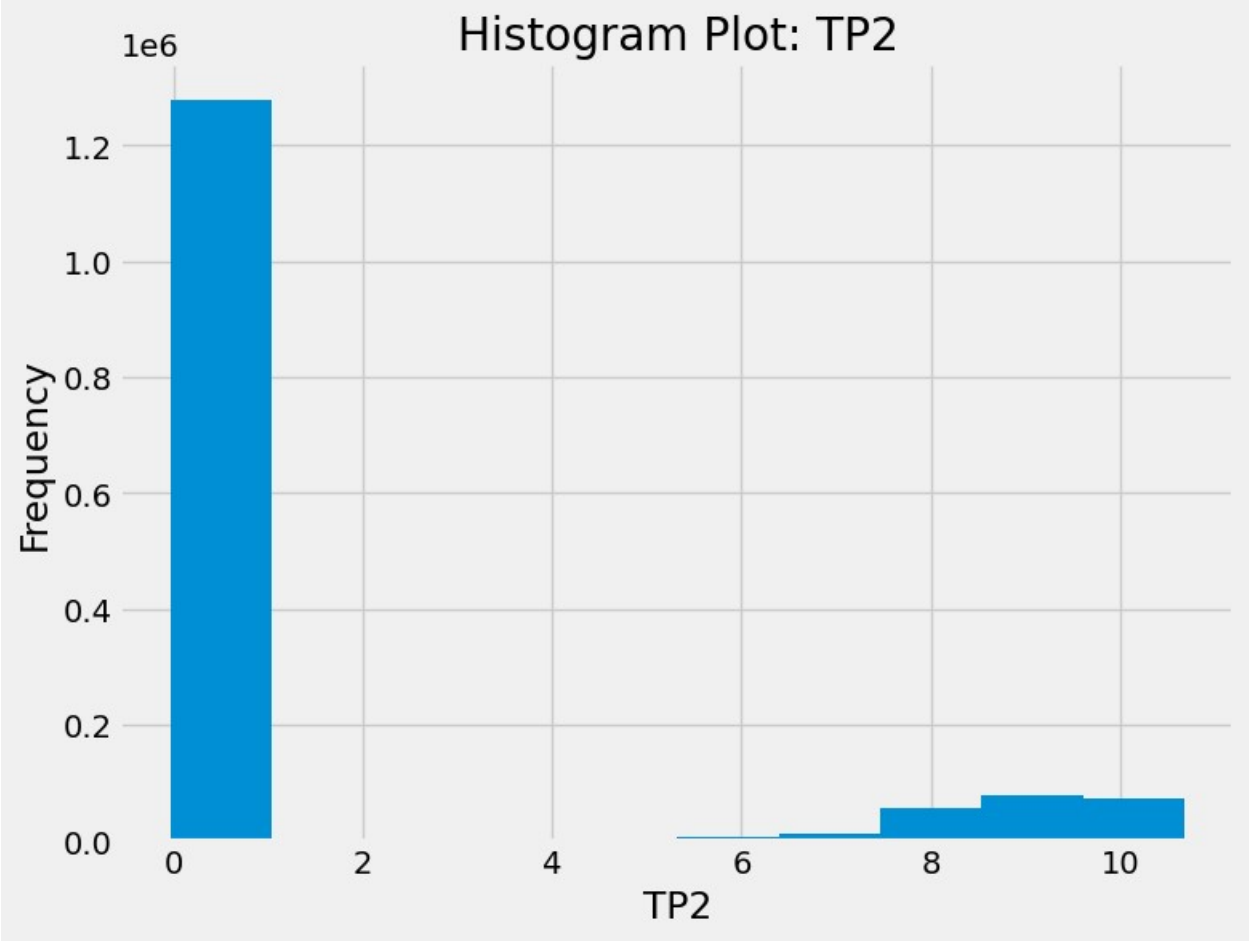
Visualization of the original Time Series :

```
# Plot the time series

plt.style.use('fivethirtyeight')
df.plot(kind='kde',
        #  This specifies the type of plot to create, which is a
kernel density plot.
        # Kernel density plots are used to estimate the probability
density function of a continuous random variable.

        subplots=True,
        layout=(6, 3),
        figsize=(22,22),
        fontsize=10,
        linewidth=2,
```

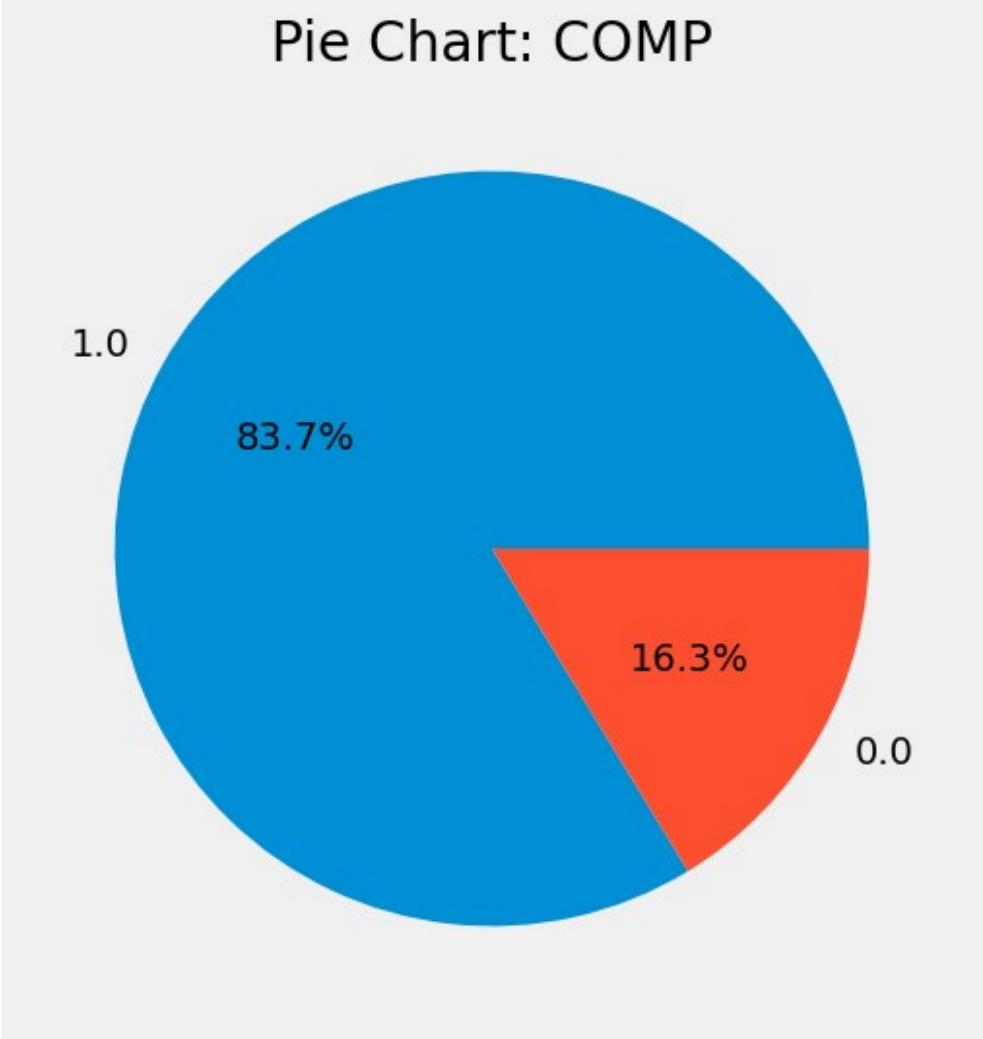
```
        sharex=False,

        # the sharex parameter is used to control the sharing of the
x-axis between subplots

        title='Visualization of the original Time Series')
plt.show()
```




```
# Histogram Plot
plt.figure(figsize=(8, 6))
plt.hist(df['TP2'], bins=10)
plt.xlabel('TP2')
plt.ylabel('Frequency')
plt.title('Histogram Plot: TP2')
plt.show()
```

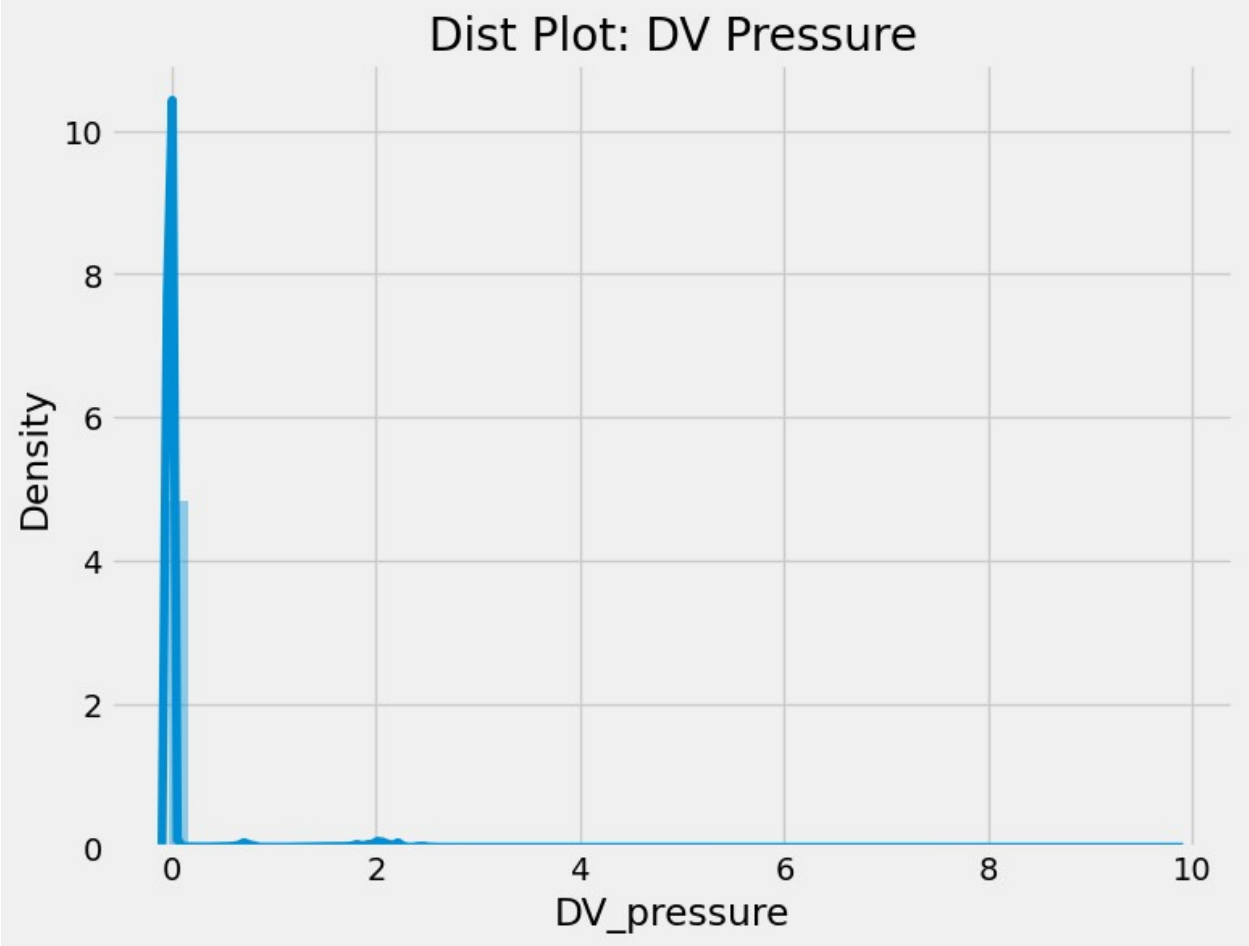



```
# Pie Chart
plt.figure(figsize=(6, 6))
pie_data = df['COMP'].value_counts()
plt.pie(pie_data, labels=pie_data.index, autopct='%1.1f%%')
plt.title('Pie Chart: COMP')
plt.show()
```

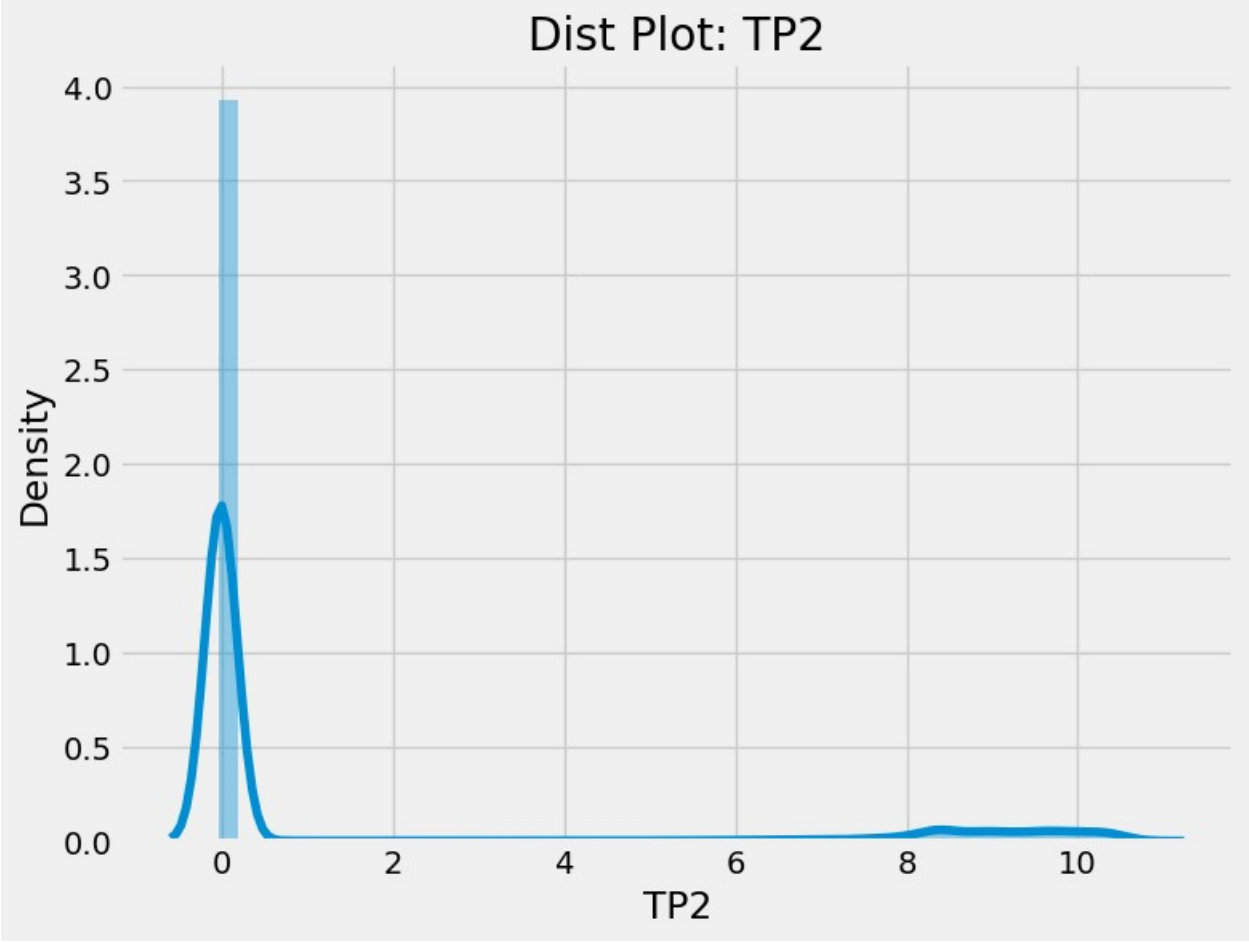


Analysis the Pressure in the form of Dist Plot :

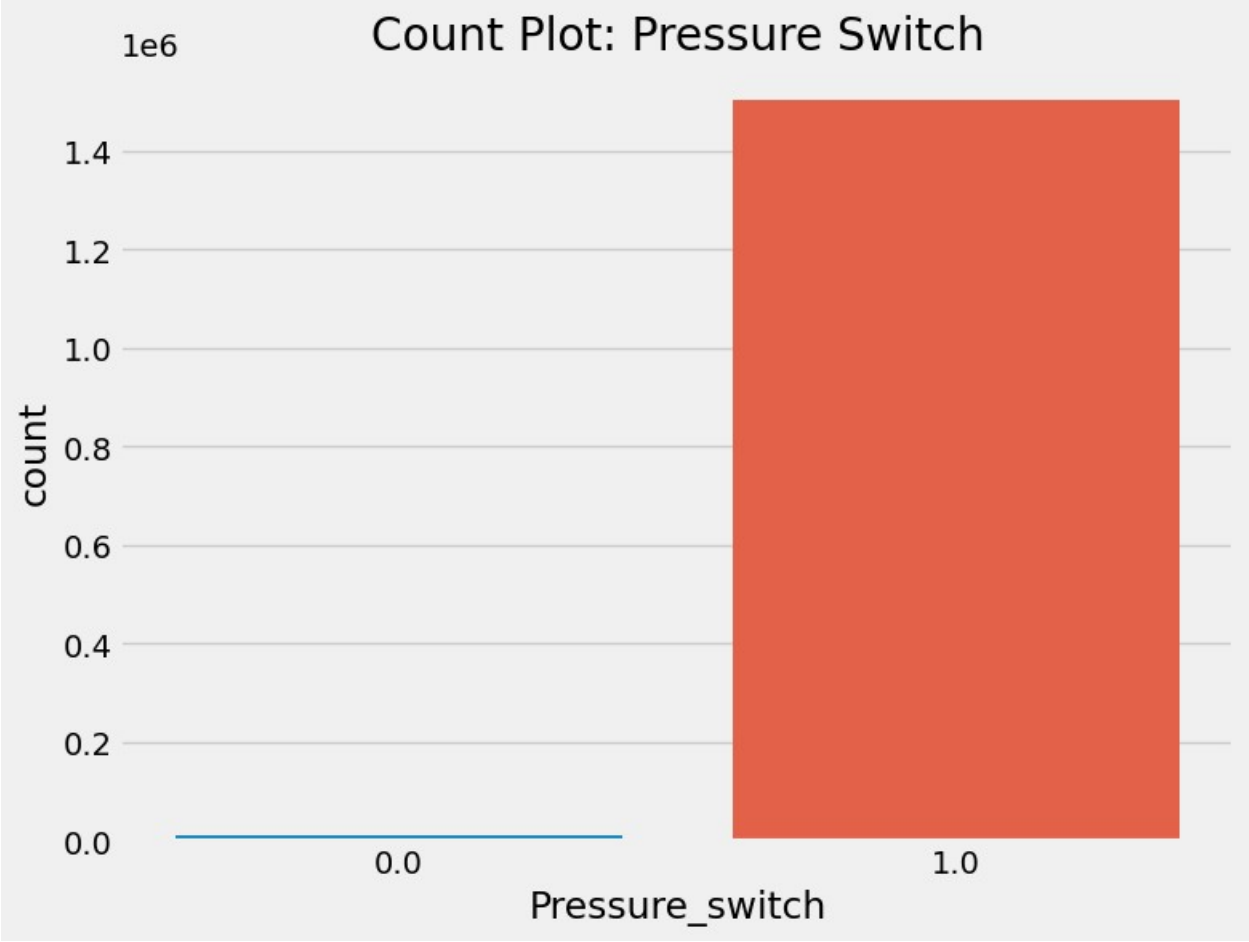
```
# Dist Plot
plt.figure(figsize=(8, 6))
sns.distplot(df['DV_pressure'])
plt.title('Dist Plot: DV Pressure')
plt.show()
```



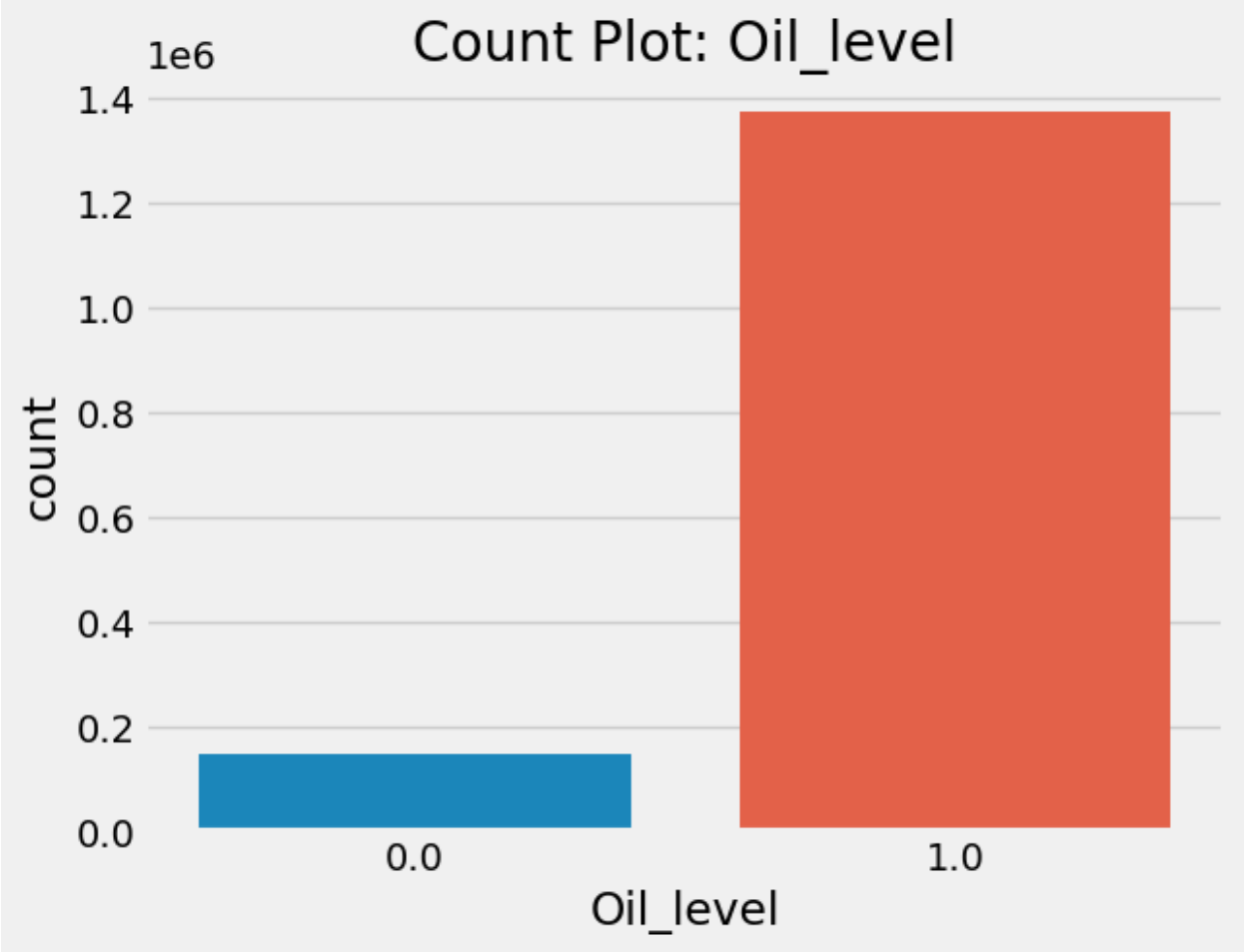

```
# Dist Plot
plt.figure(figsize=(8, 6))
sns.distplot(df['TP2'])
plt.title('Dist Plot: TP2')
plt.show()
```




```
# Count Plot
plt.figure(figsize=(8, 6))
sns.countplot(x='Pressure_switch', data=df)
plt.title('Count Plot: Pressure Switch')
plt.show()
```

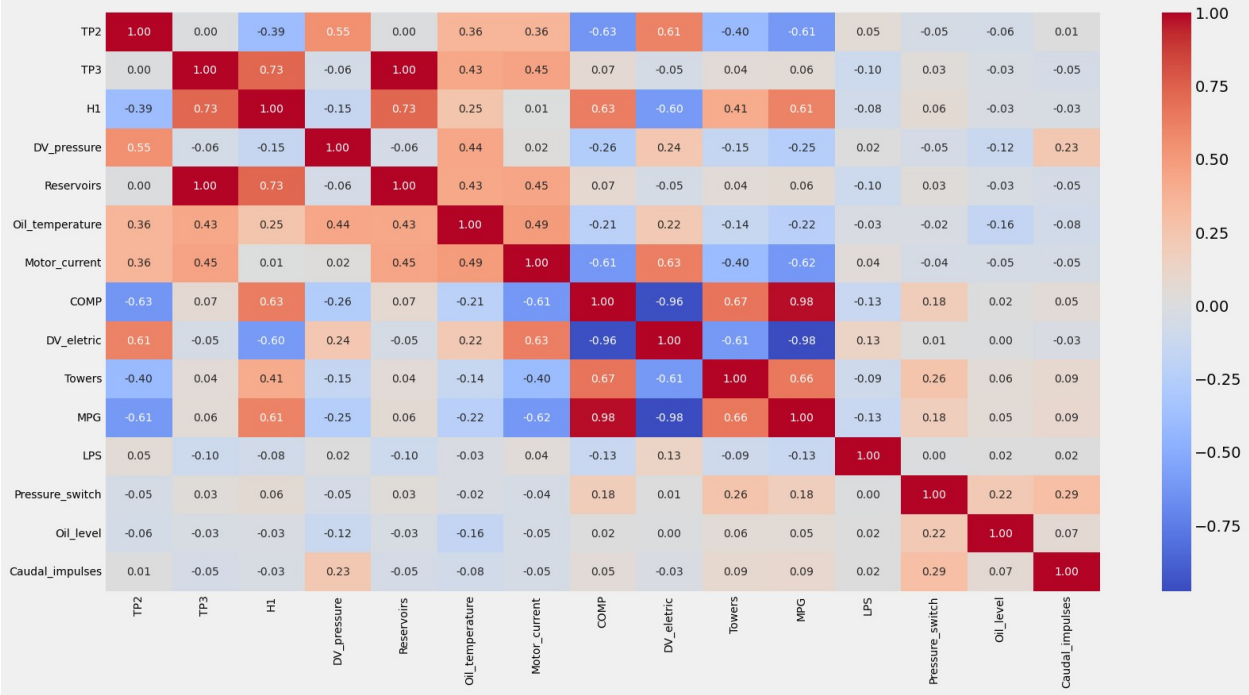



```
# Count Plot
# plt.figure(figsize=(8, 6))
sns.countplot(x='Oil_level', data=df)
plt.title('Count Plot: Oil_level')
plt.show()
```

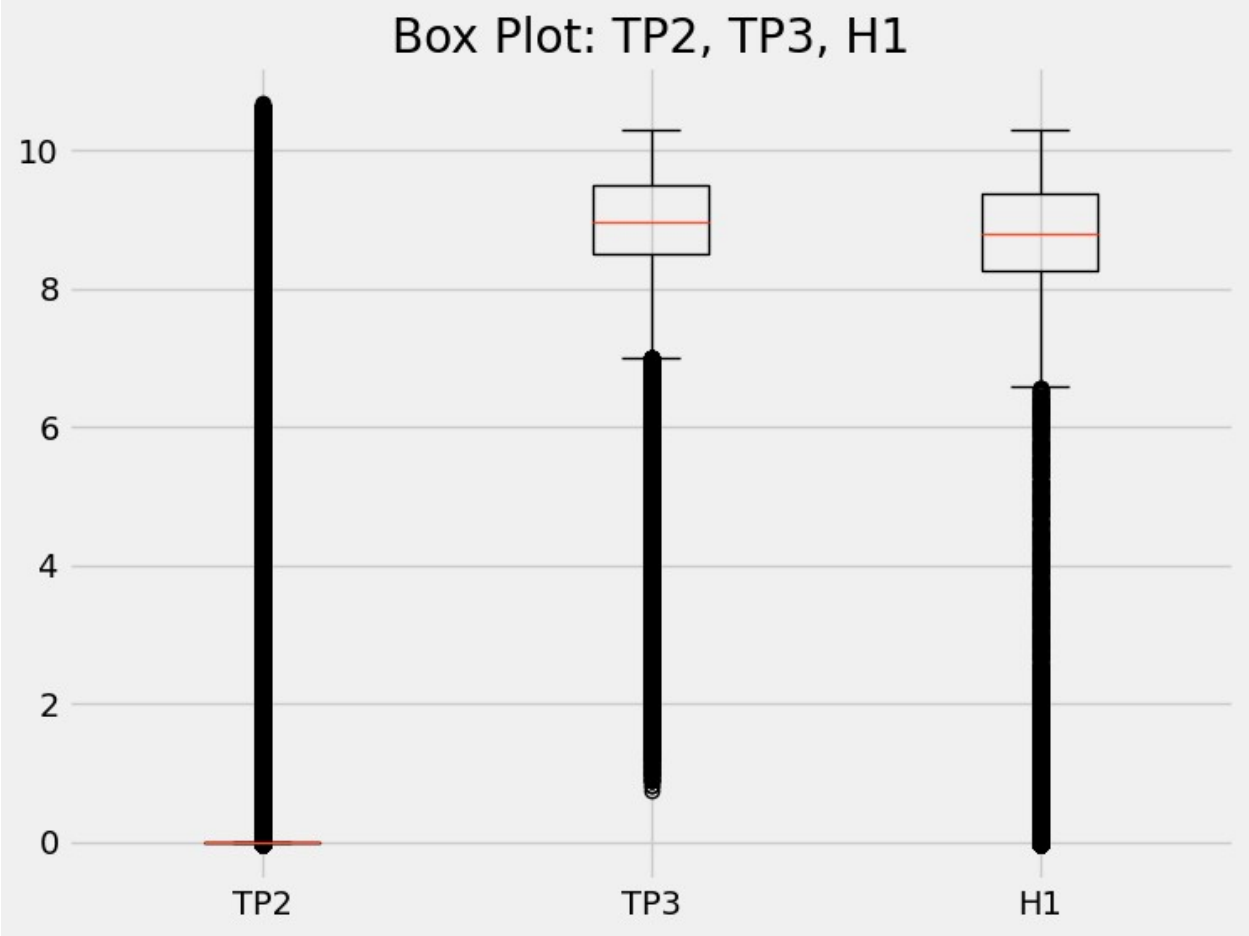


Heatmap visualization of the correlation matrix :

```
# Let's also draw a heatmap visualization of the correlation matrix

corr_matrix = df.corr(method='spearman')
f, ax = plt.subplots(figsize=(18,9))
sns.heatmap(corr_matrix,
            annot=True,
            fmt='.2f',
            linewidth=0.,
            annot_kws={"size": 10},
            cmap='coolwarm', ax=ax)

plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```




```
# Box Plot
plt.figure(figsize=(8, 6))
plt.boxplot(df[['TP2', 'TP3', 'H1']].values,
            labels=['TP2', 'TP3', 'H1'])

plt.title('Box Plot: TP2, TP3, H1')
plt.show()
```




```
# from pandas_profiling import ProfileReport
# report = ProfileReport(df)
# report.to_file("Kartik_eda.html")
```


THANK
YOU!