INTRODUCTION TO DATA SCIENCE

Group 14 -- Project 1 -- Phase 1

Name	Student ID
Pham, Quoc Huy	2299356
Hussain, Zakiuddin	2338350
Lee, Daeul Haven	2308018
Preetham	2288949
Jayanth	2288552
Srikavya	2311351

Link to colab: https://colab.research.google.com/drive/1zeUCdIEciMGKir-OK0m8sxObEmXK_QMF?usp=sharing

Link to the dataset: https://archive.ics.uci.edu/dataset/791/metropt+3+dataset

```
#Define ENV
USE_GG_DRIVE = 0
```

```
#Seting google Drive
if USE_GG_DRIVE:
  from goolge.colab import drive
  drive.mount('content\drive')
```

```
import pandas as pd
import seaborn as sns
import numpy as np

data = pd.read_csv("metro_data.csv")
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1516948 entries, 0 to 1516947
Data columns (total 17 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   Unnamed: 0       1516948 non-null  int64
 1   timestamp        1516948 non-null  object
 2   TP2              1516948 non-null  float64
 3   TP3              1516948 non-null  float64
 4   H1               1516948 non-null  float64
 5   DV_pressure      1516948 non-null  float64
 6   Reservoirs       1516948 non-null  float64
 7   Oil_temperature  1516948 non-null  float64
 8   Motor_current    1516948 non-null  float64
 9   COMP             1516948 non-null  float64
 10  DV_eletric       1516948 non-null  float64
 11  Towers           1516948 non-null  float64
 12  MPG              1516948 non-null  float64
 13  LPS              1516948 non-null  float64
 14  Pressure_switch  1516948 non-null  float64
 15  Oil_level        1516948 non-null  float64
 16  Caudal_impulses  1516948 non-null  float64
dtypes: float64(15), int64(1), object(1)
memory usage: 196.7+ MB
None
```

I. Data Introduction

The MetroPT-3 dataset was created to support the development of predictive maintenance, anomaly detection, and remaining useful life (RUL) prediction models for compressors using deep learning and machine learning methods. It consists of multivariate time series data from several analogue and digital sensors installed on a train compressor. The data spans between February and August 2020 and includes 15 signals, such as pressures, motor current, oil temperature, and electrical signals of air intake valves. The dataset is eligible for incremental training and does not contain sensitive data. Data preprocessing includes data segmentation, normalization, and feature extraction. The dataset is unlabeled, but failure reports provided by the company are available for evaluating the effectiveness of anomaly detection, failure prediction, and RUL estimation algorithms. The dataset does not have missing values.

This dataset was chosen because it contains information that was obtained from actual events that accurately reflect real-world situations. This makes it a valuable resource for developing and testing algorithms in the field of anomaly detection, failure prediction, and remaining useful life estimation. Additionally, the availability of failure reports from the company allows us to validate the performance of our algorithms against real-world incidents. Also, this dataset is also represent time series attribute, which is crucial for analyzing trends and patterns over a specific period. The inclusion of time series data enables researchers to observe the progression of anomalies and failures, providing deeper insights into their causes and potential mitigation strategies.

II. Data Cleaning and Pre-precessing

1) Data Overview

```
print(data.describe().round(2))
print(data.columns)
```

```
        Unnamed: 0         TP2         TP3          H1  DV_pressure  \
count   1516948.00  1516948.00  1516948.00  1516948.00   1516948.00
mean    7584735.00        1.37        8.98        7.57         0.06
std     4379053.12        3.25        0.64        3.33         0.38
min           0.00       -0.03        0.73       -0.04        -0.03
25%     3792367.50       -0.01        8.49        8.25        -0.02
50%     7584735.00       -0.01        8.96        8.78        -0.02
75%    11377102.50       -0.01        9.49        9.37        -0.02
max    15169470.00       10.68       10.30       10.29         9.84

       Reservoirs  Oil_temperature  Motor_current        COMP  DV_eletric  \
count  1516948.00       1516948.00     1516948.00  1516948.00  1516948.00
mean         8.99            62.64           2.05        0.84        0.16
std          0.64             6.52           2.30        0.37        0.37
min          0.71            15.40           0.02        0.00        0.00
25%          8.49            57.78           0.04        1.00        0.00
50%          8.96            62.70           0.04        1.00        0.00
75%          9.49            67.25           3.81        1.00        0.00
max         10.30            89.05           9.30        1.00        1.00

           Towers         MPG         LPS  Pressure_switch   Oil_level  \
count  1516948.00  1516948.00  1516948.00       1516948.00  1516948.00
mean         0.92        0.83        0.00             0.99        0.90
std          0.27        0.37        0.06             0.09        0.29
min          0.00        0.00        0.00             0.00        0.00
25%          1.00        1.00        0.00             1.00        1.00
50%          1.00        1.00        0.00             1.00        1.00
75%          1.00        1.00        0.00             1.00        1.00
max          1.00        1.00        1.00             1.00        1.00

       Caudal_impulses
count       1516948.00
mean              0.94
std               0.24
min               0.00
25%               1.00
50%               1.00
75%               1.00
max               1.00
Index(['Unnamed: 0', 'timestamp', 'TP2', 'TP3', 'H1', 'DV_pressure',
       'Reservoirs', 'Oil_temperature', 'Motor_current', 'COMP', 'DV_eletric',
       'Towers', 'MPG', 'LPS', 'Pressure_switch', 'Oil_level',
       'Caudal_impulses'],
      dtype='object')
```

```
print(data.head(10))
```

```
   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
5          50  2020-02-01 00:00:49 -0.012  9.306  9.290       -0.024
6          60  2020-02-01 00:00:59 -0.012  9.296  9.280       -0.024
7          70  2020-02-01 00:01:09 -0.014  9.286  9.270       -0.024
8          80  2020-02-01 00:01:19 -0.012  9.276  9.258       -0.022
9          90  2020-02-01 00:01:29 -0.012  9.264  9.248       -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
5       9.308           53.500         0.0400   1.0         0.0     1.0  1.0
6       9.298           53.375         0.0400   1.0         0.0     1.0  1.0
7       9.286           53.550         0.0400   1.0         0.0     1.0  1.0
8       9.276           53.425         0.0400   1.0         0.0     1.0  1.0
9       9.264           53.375         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
5  0.0              1.0        1.0              1.0
6  0.0              1.0        1.0              1.0
7  0.0              1.0        1.0              1.0
8  0.0              1.0        1.0              1.0
9  0.0              1.0        1.0              1.0
```

2) Drop unnecessary columns

```
print(data.columns)
data = data.drop('Unnamed: 0', axis= 1)
```

```
Index(['Unnamed: 0', 'timestamp', 'TP2', 'TP3', 'H1', 'DV_pressure',
       'Reservoirs', 'Oil_temperature', 'Motor_current', 'COMP', 'DV_eletric',
       'Towers', 'MPG', 'LPS', 'Pressure_switch', 'Oil_level',
       'Caudal_impulses'],
      dtype='object')
```

3) Convert the timestamp collumn into pandas.DateTime data type

```
import datetime

#Check the current type of timestamp
print(f"Current type of timestamp is {type(data.timestamp[0])}")

#Convert timestamp to pandas.DateTime
data['timestamp'] = data['timestamp'].apply(pd.to_datetime, format = "%Y-%m-%d %H:%M:%S")

#Re-check the type
print(f"Current type of timestamp is {type(data.timestamp[0])}")
```

```
Current type of timestamp is <class 'str'>
Current type of timestamp is <class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

```
print(data.head(10))
```

```
            timestamp    TP2    TP3     H1  DV_pressure  Reservoirs  \
0 2020-02-01 00:00:00 -0.012  9.358  9.340       -0.024       9.358
1 2020-02-01 00:00:10 -0.014  9.348  9.332       -0.022       9.348
2 2020-02-01 00:00:19 -0.012  9.338  9.322       -0.022       9.338
3 2020-02-01 00:00:29 -0.012  9.328  9.312       -0.022       9.328
4 2020-02-01 00:00:39 -0.012  9.318  9.302       -0.022       9.318
5 2020-02-01 00:00:49 -0.012  9.306  9.290       -0.024       9.308
6 2020-02-01 00:00:59 -0.012  9.296  9.280       -0.024       9.298
7 2020-02-01 00:01:09 -0.014  9.286  9.270       -0.024       9.286
8 2020-02-01 00:01:19 -0.012  9.276  9.258       -0.022       9.276
9 2020-02-01 00:01:29 -0.012  9.264  9.248       -0.022       9.264

   Oil_temperature  Motor_current  COMP  DV_eletric  Towers  MPG  LPS  \
0           53.600         0.0400   1.0         0.0     1.0  1.0  0.0
1           53.675         0.0400   1.0         0.0     1.0  1.0  0.0
2           53.600         0.0425   1.0         0.0     1.0  1.0  0.0
3           53.425         0.0400   1.0         0.0     1.0  1.0  0.0
4           53.475         0.0400   1.0         0.0     1.0  1.0  0.0
5           53.500         0.0400   1.0         0.0     1.0  1.0  0.0
6           53.375         0.0400   1.0         0.0     1.0  1.0  0.0
7           53.550         0.0400   1.0         0.0     1.0  1.0  0.0
8           53.425         0.0400   1.0         0.0     1.0  1.0  0.0
9           53.375         0.0400   1.0         0.0     1.0  1.0  0.0
```

```
   Pressure_switch  Oil_level  Caudal_impulses
0              1.0        1.0              1.0
1              1.0        1.0              1.0
2              1.0        1.0              1.0
3              1.0        1.0              1.0
4              1.0        1.0              1.0
5              1.0        1.0              1.0
6              1.0        1.0              1.0
7              1.0        1.0              1.0
8              1.0        1.0              1.0
9              1.0        1.0              1.0
```

4) Add a label column

```
#Create a new column for target variable called status, indicate the equipment has deficiencies and need to be maintained
# status = 0; system ups and running
# status = 1; system downs and needs recovering
labeled_data = data.copy()
labeled_data['status'] = 0
print(labeled_data.head(5))
```

```
             timestamp    TP2    TP3     H1  DV_pressure  Reservoirs  \
0 2020-02-01 00:00:00 -0.012  9.358  9.340       -0.024       9.358
1 2020-02-01 00:00:10 -0.014  9.348  9.332       -0.022       9.348
2 2020-02-01 00:00:19 -0.012  9.338  9.322       -0.022       9.338
3 2020-02-01 00:00:29 -0.012  9.328  9.312       -0.022       9.328
4 2020-02-01 00:00:39 -0.012  9.318  9.302       -0.022       9.318

   Oil_temperature  Motor_current  COMP  DV_eletric  Towers  MPG  LPS  \
0           53.600         0.0400   1.0         0.0     1.0  1.0  0.0
1           53.675         0.0400   1.0         0.0     1.0  1.0  0.0
2           53.600         0.0425   1.0         0.0     1.0  1.0  0.0
3           53.425         0.0400   1.0         0.0     1.0  1.0  0.0
4           53.475         0.0400   1.0         0.0     1.0  1.0  0.0

   Pressure_switch  Oil_level  Caudal_impulses  status
0              1.0        1.0              1.0       0
1              1.0        1.0              1.0       0
2              1.0        1.0              1.0       0
3              1.0        1.0              1.0       0
4              1.0        1.0              1.0       0
```

Next, set the status of the machine to 1 based on the time from the table below

Nr.	Start Time	End Time	Failure	Severity	Report
#1	4/18/2020 0:00	4/18/2020 23:59	**Air leak**	High stress	
#1	5/29/2020 23:30	5/30/2020 6:00	**Air Leak**	High stress	Maintenance on 30Apr at 12:00
#3	6/5/2020 10:00	6/7/2020 14:30	**Air Leak**	High stress	Maintenance on 8Jun at 16:00
#4	7/15/2020 14:30	7/15/2020 19:00	**Air Leak**	High stress	Maintenance on 16Jul at 00:00

```
def to_datetime(xs):
  result = []
  format =  "%Y-%m-%d %H:%M:%S"
  for x in xs:
```

```
    result.append(pd.to_datetime(x, format = format))
  return result


failure_start_time = to_datetime(["2020-04-18 00:00:00", "2020-05-29 23:30:00", "2020-06-05 10:00:00", "2020-07-15 14:30:00"] )
failure_end_time   = to_datetime(["2020-04-18 23:59:00", "2020-05-30 06:00:00", "2020-06-07 14:30:00", "2020-07-15 19:00:00"] )

print(failure_start_time,"\n", failure_end_time[0].minute)
```

```
[Timestamp('2020-04-18 00:00:00'), Timestamp('2020-05-29 23:30:00'), Timestamp('2020-06-05 10:00:00'), Timestamp('2020-07-15 14:30:00')]
 59
```

```
def in_between(x, start, end):

  start_con = x >= start
  end_con = x<= end

  inbetween_con = start_con and end_con
  if inbetween_con:
    return 1
  else:
    return 0


failure_indx = []
import numpy as np
for i, (start_time, end_time) in enumerate(zip(failure_start_time, failure_end_time)):
  mask = labeled_data['timestamp'].apply(in_between, start = start_time, end = end_time)
  indx = labeled_data.index[mask == True].tolist()
  failure_indx += indx


print(f" Found {len(failure_indx)} samples representing failure state")
```

```
 Found 29954 samples representing failure state
```

```
#Set the sample with the timestamp falled between the failure time to 1
labeled_data['status'].iloc[failure_indx] = 1
```

```
<ipython-input-16-502c3a3e2569>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  labeled_data['status'].iloc[failure_indx] = 1
```

```
#Check the number of possitive samples
print(f"We have {labeled_data['status'][labeled_data['status']==1].count()} positve samples" )
```

```
We have 29954 positve samples
```

```
print(f"Example of Failure state \n {labeled_data[labeled_data['status']==1].head()}")
```

```
Example of Failure state
                  timestamp    TP2    TP3     H1  DV_pressure  Reservoirs  \
562564 2020-04-18 00:00:01 -0.018  8.248  8.238       -0.024       8.248
562565 2020-04-18 00:00:13 -0.018  8.248  8.238       -0.024       8.248
562566 2020-04-18 00:00:24 -0.018  8.248  8.238       -0.024       8.248
562567 2020-04-18 00:00:36 -0.018  8.248  8.238       -0.024       8.248
562568 2020-04-18 00:00:49 -0.018  8.248  8.238       -0.024       8.248

        Oil_temperature  Motor_current  COMP  DV_eletric  Towers  MPG  LPS  \
562564            49.45           0.04   1.0         0.0     1.0  1.0  0.0
562565            49.45           0.04   1.0         0.0     1.0  1.0  0.0
562566            49.45           0.04   1.0         0.0     1.0  1.0  0.0
562567            49.45           0.04   0.0         0.0     0.0  0.0  0.0
562568            49.45           0.04   1.0         0.0     1.0  1.0  0.0

        Pressure_switch  Oil_level  Caudal_impulses  status
562564              1.0        1.0              1.0       1
562565              1.0        1.0              1.0       1
562566              1.0        1.0              1.0       1
562567              0.0        0.0              0.0       1
562568              1.0        1.0              1.0       1
```

5) Subsample the dataset

```
#Seperate Positive samples and Negative sample
pos_data = labeled_data[labeled_data['status'] == 1]
neg_data = labeled_data[labeled_data['status'] == 0]

#Print out the info of 2 dataset
print(f"Positive dataset\n {pos_data.info()}\n")
print(f"Negative dataset\n {neg_data.info()}\n")
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29954 entries, 562564 to 1172714
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   timestamp        29954 non-null  datetime64[ns]
 1   TP2              29954 non-null  float64
 2   TP3              29954 non-null  float64
 3   H1               29954 non-null  float64
 4   DV_pressure      29954 non-null  float64
 5   Reservoirs       29954 non-null  float64
 6   Oil_temperature  29954 non-null  float64
 7   Motor_current    29954 non-null  float64
 8   COMP             29954 non-null  float64
 9   DV_eletric       29954 non-null  float64
 10  Towers           29954 non-null  float64
 11  MPG              29954 non-null  float64
 12  LPS              29954 non-null  float64
 13  Pressure_switch  29954 non-null  float64
 14  Oil_level        29954 non-null  float64
 15  Caudal_impulses  29954 non-null  float64
 16  status           29954 non-null  int64
dtypes: datetime64[ns](1), float64(15), int64(1)
memory usage: 4.1 MB
Positive dataset
 None

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1486994 entries, 0 to 1516947
Data columns (total 17 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   timestamp        1486994 non-null  datetime64[ns]
 1   TP2              1486994 non-null  float64
 2   TP3              1486994 non-null  float64
 3   H1               1486994 non-null  float64
 4   DV_pressure      1486994 non-null  float64
 5   Reservoirs       1486994 non-null  float64
 6   Oil_temperature  1486994 non-null  float64
 7   Motor_current    1486994 non-null  float64
 8   COMP             1486994 non-null  float64
 9   DV_eletric       1486994 non-null  float64
 10  Towers           1486994 non-null  float64
 11  MPG              1486994 non-null  float64
 12  LPS              1486994 non-null  float64
 13  Pressure_switch  1486994 non-null  float64
 14  Oil_level        1486994 non-null  float64
 15  Caudal_impulses  1486994 non-null  float64
 16  status           1486994 non-null  int64
dtypes: datetime64[ns](1), float64(15), int64(1)
memory usage: 204.2 MB
Negative dataset
 None
```

As we can see, we have around 30K postive samples and 1500K negative sample. This indicates highly imbalanced dataset. Thus, we have to subsample the negative class to balance the training data. To achive this, we will randomly sample 30K negative sample from the set of 1500K sample.

```
n_positives = int(pos_data['status'].count())
sub_neg_data = neg_data.sample(n_positives, random_state = 42)
print(f"Negative dataset after subsampling {sub_neg_data.info()}")
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29954 entries, 1306134 to 1276653
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   timestamp        29954 non-null  datetime64[ns]
 1   TP2              29954 non-null  float64
 2   TP3              29954 non-null  float64
 3   H1               29954 non-null  float64
 4   DV_pressure      29954 non-null  float64
 5   Reservoirs       29954 non-null  float64
 6   Oil_temperature  29954 non-null  float64
 7   Motor_current    29954 non-null  float64
 8   COMP             29954 non-null  float64
 9   DV_eletric       29954 non-null  float64
 10  Towers           29954 non-null  float64
 11  MPG              29954 non-null  float64
 12  LPS              29954 non-null  float64
 13  Pressure_switch  29954 non-null  float64
 14  Oil_level        29954 non-null  float64
 15  Caudal_impulses  29954 non-null  float64
 16  status           29954 non-null  int64
dtypes: datetime64[ns](1), float64(15), int64(1)
memory usage: 4.1 MB
Negative dataset after subsampling None
```

Now, we merge the postive set and negative set into one

```
merged_data = pd.concat([pos_data, sub_neg_data], axis = 0)
print(f"Merged dataset\n")
merged_data.info()
```

6) Find and drop ouliers

```
def investigate_outliers(data, c):
    q1 = data[c].quantile(0.25)
    q3 = data[c].quantile(0.75)
    iqr = q3 - q1
    ll = q1 - 1.5*iqr
    ul = q3 + 1.5*iqr

    num_outliers = data[data[c] < ll][c].count()  + data[data[c] > ul][c].count()
    if num_outliers>0:
        print(f"Found {num_outliers} oulier(s) for feature {c}")
    return {'col': c, 'n_outliers': num_outliers, 'll': ll, 'ul': ul, 'q1': q1, 'q3':q3}

print("\nDropping outliers ...\n")
clean_data = merged_data.copy()
for i in range(5):
  for c in clean_data.columns:
      if c not in ["Unnamed: 0","timestamp"]:
          cue = investigate_outliers(clean_data, c)
          if cue["n_outliers"] > 0 and (cue["q1"]!= cue["q3"]):
              print(f"Droping {cue['n_outliers']} from column {c}")
              clean_data = clean_data[clean_data[c]> cue["ll"]]
              clean_data = clean_data[clean_data[c]< cue["ul"]]
              print(f"{clean_data.shape[0]} samples left\n")
          elif (cue["q1"]== cue["q3"]):
              print("Skipping .. data has Q1 equals to Q3")
              print(f"{clean_data.shape[0]} rows left\n")


print("\nDropping Completed ...\n")
#Recheck data
for c in clean_data.columns:
    if c not in ["Unnamed: 0","timestamp","COMP", 'status']:
        cue = investigate_outliers(clean_data, c)
```

```
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 2897 oulier(s) for feature Oil_level
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 1947 oulier(s) for feature Caudal_impulses
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 395 oulier(s) for feature LPS
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 402 oulier(s) for feature Pressure_switch
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 2897 oulier(s) for feature Oil_level
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 1947 oulier(s) for feature Caudal_impulses
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 395 oulier(s) for feature LPS
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 402 oulier(s) for feature Pressure_switch
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 2897 oulier(s) for feature Oil_level
Skipping .. data has Q1 equals to Q3
59445 rows left

Found 1947 oulier(s) for feature Caudal_impulses
Skipping .. data has Q1 equals to Q3
59445 rows left


Dropping Completed ...
```

```
#Investigate the columns with the binary values
binary_cols = ['LPS', 'Pressure_switch', 'Oil_level', 'Caudal_impulses']
#Ensure the the binary data is binary
clean_data[binary_cols] = clean_data[binary_cols].apply(np.round)
```

7) Summary

In summary, prior to undergoing preprocessing, the dataset:

- has unnecessary collumn
- has columns with wrong format
- is unlabeled
- is highly imbalanced
- has outliers

Thus, throughout the preprocessing and data cleaning phase, we performed the following tasks:

- remove unnecessary column
- format the timestamp column
- add column for target variable
- subsample to balance the dataset
- find and drop all outliers

According to its documentation, the following preprocessing steps have been conducted before publishing the data, so we do not apply it in our work:

- Data segmentation
- Normalization
- Feature Extraction

III. Exploratory Data Analysis

1) Correlation

Describing the correlation between the features, the values closer to 1 or -1 represent a stronger relation.

```
clean_data.corr().round(2)
```

```
<ipython-input-24-ef7828e8fbfe>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  clean_data.corr().round(2)
```

	TP2	TP3	H1	DV_pressure	Reservoirs	Oil_temperature	Motor_current	COMP	DV_eletric	Towers	MPG	LPS	Pressure
TP2	1.00	-0.42	-0.98	0.81	-0.42	0.74	0.87	-0.98	0.98	-0.52	-0.97	0.06	
TP3	-0.42	1.00	0.54	-0.55	1.00	-0.28	-0.17	0.49	-0.48	0.27	0.48	-0.18	
H1	-0.98	0.54	1.00	-0.83	0.54	-0.73	-0.84	0.99	-0.99	0.54	0.99	-0.07	
DV_pressure	0.81	-0.55	-0.83	1.00	-0.55	0.73	0.72	-0.83	0.83	-0.45	-0.83	-0.05	
Reservoirs	-0.42	1.00	0.54	-0.55	1.00	-0.28	-0.17	0.49	-0.48	0.27	0.48	-0.18	
Oil_temperature	0.74	-0.28	-0.73	0.73	-0.28	1.00	0.79	-0.75	0.76	-0.41	-0.75	0.10	
Motor_current	0.87	-0.17	-0.84	0.72	-0.17	0.79	1.00	-0.87	0.87	-0.47	-0.87	0.06	
COMP	-0.98	0.49	0.99	-0.83	0.49	-0.75	-0.87	1.00	-0.99	0.55	1.00	-0.07	
DV_eletric	0.98	-0.48	-0.99	0.83	-0.48	0.76	0.87	-0.99	1.00	-0.53	-0.99	0.07	
Towers	-0.52	0.27	0.54	-0.45	0.27	-0.41	-0.47	0.55	-0.53	1.00	0.55	-0.04	
MPG	-0.97	0.48	0.99	-0.83	0.48	-0.75	-0.87	1.00	-0.99	0.55	1.00	-0.07	
LPS	0.06	-0.18	-0.07	-0.05	-0.18	0.10	0.06	-0.07	0.07	-0.04	-0.07	1.00	
Pressure_switch	0.00	0.01	0.01	0.02	0.01	0.02	0.00	0.07	0.03	0.09	0.06	0.01	
Oil_level	0.19	-0.14	-0.20	0.21	-0.14	0.10	0.14	-0.17	0.19	-0.07	-0.15	0.02	

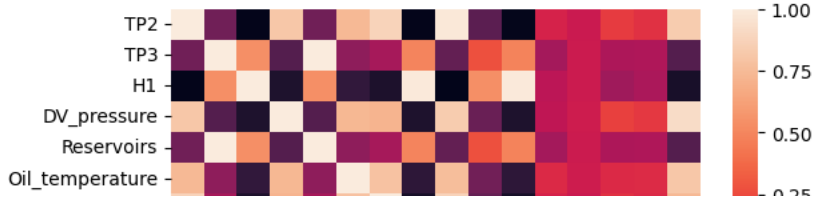
We can see that our target variable "status" has high correlation with TP2, H1, DV_pressure, Oil_temparature, Motor_current, COMP, DV_electric and MPG.

2) Visualize Correlation

Below shows a Heat map,which can be used to analyse trends, from the below heat map you can see the trends in correlation of data.

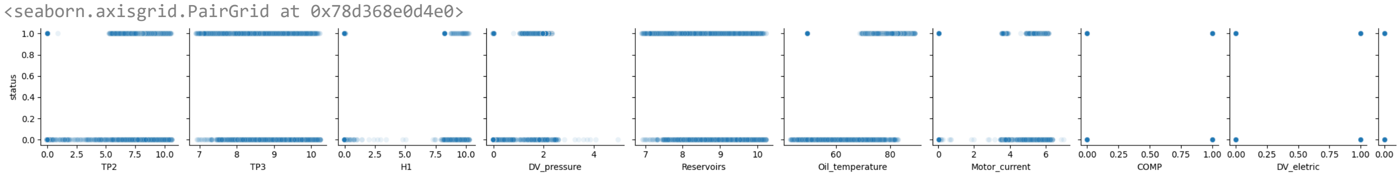
```
sns.heatmap(clean_data.corr().round(2),annot=False )
```

```
<ipython-input-25-c75868ede4c3>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  sns.heatmap(clean_data.corr().round(2),annot=False )
<Axes: >
```




```
sns.pairplot(clean_data,  y_vars = ['status'] , plot_kws=  {'alpha' : 0.1})
```

```
<seaborn.axisgrid.PairGrid at 0x78d368e0d4e0>
```



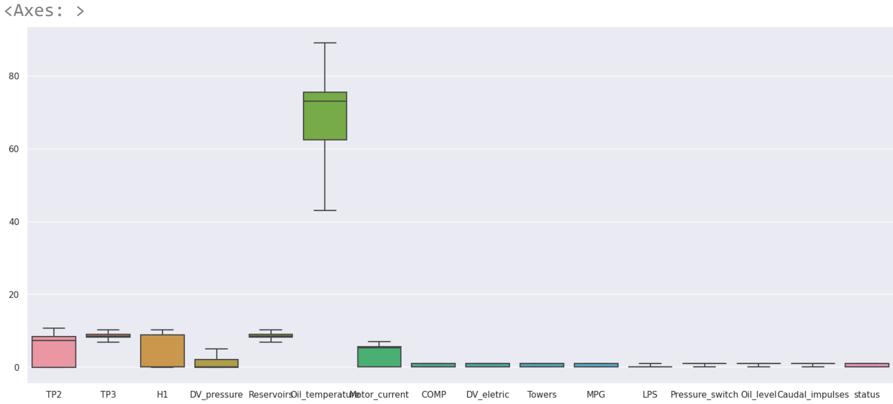
3) Visualize Outliers

Drawing box plot to find outliers, I plot it on scale data so it is easier to visualize different features' range. As we can see our preprocessing function work perfectly that leaves no outliers

```
sns.set(rc={'figure.figsize':(20,8.27)})
```

```
sns.boxplot(clean_data, autorange = True)
```

```
<Axes: >
```



IV. Save Data

```
clean_data.to_csv('Group_14_Clean_Data.csv')
np.savez("Group_14_Clean_Data.npz", clean_data.to_numpy())
```