VALLEYHACK

Analiza danych maszyn

wozy odstawcze – WOS

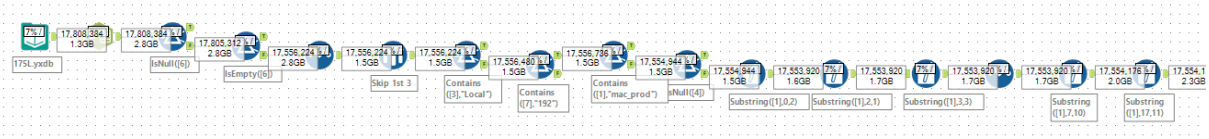
Team: NatLuk

11-13 czerwca 2021 r.

Analizę danych rozpoczęliśmy od wczytania i uporządkowania danych. Etap ten przeprowadziliśmy w programie Alteryx oraz równolegle przy wykorzystaniu skryptu w języku Python. Z uwagi na duży rozmiar danych oraz ograniczone moce obliczeniowe, którymi dysponowaliśmy dokonaliśmy wyboru rodzaju maszyny oraz awarii, na której się skupimy podczas pracy.

W pliku Awarie SMG wybraliśmy awarię nr 2. Dotyczyła ona wozu odstawczego WOS 175/L i awarii silnika wysokoprężnego QSB6,7-C240 CUMMINS. W silniku wystąpiły przedmuchy do skrzyni korbowej - przytarte pierścienie tłokowe. W materiałach nie został podany dokładny czas awarii. Urządzenie pracowało od 21.05.2019 do 11.02.2021.

Opracowaliśmy workflow w Alteryxie, który pobiera dane z plików zip, rozpakowuje je i wczytuje dane do pamięci a następnie przefiltrowuje i czyści.



Rozkodowaliśmy nazwy sygnałów w plikach csv i wyodrębniliśmy z nich elementy niezbędne do analizy.

Analogiczne przekształcenia dokonywaliśmy w Pythonie. Na kilka sposobów.

Sposób 1

```
import glob
import zipfile
import pandas as pd
import gc
```

```
columns = ['LP1', 'Data', 'Czas', 'LP2','Wartosc','Jakosc_pomiaru']
df=pd.DataFrame(columns=columns)
print(df)

#pętla wczytująca wszystkie csv z zipów z jednego katalogu

for zip_file in glob.glob("C:/Users/E7240/Documents/cuvalley/MonitoringSMG/WOS 175L/2020/05/*.zip"):
    zf = zipfile.ZipFile(zip_file)
    dfs = [pd.read_csv(zf.open(f), sep='|', skiprows=4, names=columns) for f in zf.namelist()]
    df = df.append(dfs)
#    df = pd.concat(dfs,ignore_index=False)
    print(zip_file)

# resetowanie indeksu
df=df.reset_index()

# wyodrębnianie informacji z pierwszej kolumny
df['param']=df['index'].str.slice(start=17, stop=28)
df['Analog/Digital']=df['index'].str.slice(start=28, stop=29)
df['jednostka']=df['index'].str.slice(start=29, stop=32)
df['param+jednostka']=df['param'] + df['jednostka']

#przekształcenie daty i czasu na typ datetime
df['Data_time']=df.Data+' '+df.Czas
df['Data_time']=pd.to_datetime(df['Data_time'], format='%Y/%m/%d %H:%M:%S.%f')

#usunięcie zbędnych kolumn
df.drop(labels=['param','jednostka','Data','Czas'], axis=1, inplace=True)

print('Skończono transformację danych')
#odfiltrowanie prawidłowych pomiarów
df=df[df.Jakosc_pomiaru == 192]

df_pivot=pd.pivot_table(df, index='Data_time',columns='param+jednostka', values='Wartosc', aggfunc='mean')

print('skończono pivot')

df_pivot.to_csv('C:/Users/E7240/OneDrive/cuvalley/pivot2020_05.csv')
print('csv ready')
```

Sposób 2

```
import pandas as pd
import numpy as np
import re
from datetime import datetime

jedyna = pd.read_csv("KLDSMG_WOS___175L_20200501_00-01.csv")
```

```
def oddzial(value):
    if len(value)<3:
        return np.nan
    else:
        return value[:2]
def dane_blad(value):
    if len(value)<3:
        return np.nan
    else:
        return value[2]
def machine(value):
    if len(value)<6:
        return np.nan
    else:
        return value[3:6]
def new_1(value):
    if len(value)<10:
        return np.nan
    else:
        return value[7:17]
def new_1_2(value):
    if len(value)<10:
        return np.nan
    else:
        value = re.sub(r"[^A-Za-z]+", '', value[17:28])
        return value
def new_1_4(value):
    if len(value)<27:
        return np.nan
    else:
        return value[28]
def new_1_3(value):
    if len(value)<28:
        return np.nan
    else:
        return value[29:]
```

```
def czas(col):
    if type(col[0]) == 'NoneType' or col[0] == None or len(col[1])<4:
        return np.nan
    else:
        value = col[0] + ' ' + col[1]
        return datetime.strptime(value, '%Y/%m/%d %H:%M:%S.%f')

def preprocessing(jedyna):
    jedyna = jedyna['ASCII'].str.split('|', expand=True)
    jedyna.columns=['test', '1', 'Data', 'Czas', '4', '5', '6']
    jedyna['oddzial'] = jedyna['test'].apply(oddzial)
    jedyna['blad'] = jedyna['test'].apply(dane_blad)
    jedyna['machine'] = jedyna['test'].apply(machine)
    jedyna['New_1'] = jedyna['test'].apply(new_1)
    jedyna['New_1_2'] = jedyna['test'].apply(new_1_2)
    jedyna['New_1_3'] = jedyna['test'].apply(new_1_3)
    jedyna['New_1_4'] = jedyna['test'].apply(new_1_4)
    jedyna.dropna(inplace=True)
    jedyna['Datetime'] = jedyna[['Data', 'Czas']].apply(czas, axis=1)
    return jedyna[['oddzial', 'blad', 'machine', 'New_1', 'New_1_2', 'New_1_3', 'New_1_4', 'Datetime']]
```

```
result = preprocessing(jedyna)
result
```

Sposób 3

```
import pandas as pd
import numpy as np
jedyna = pd.read_csv("KLDSMG_WOS___175L_20200501_00-01.csv")
```

```
# Przygotowuje nazwy kolumn i tablice
columns = ['Numer_2', 'Data', 'Czas', 'Numer_5','Numer_6','Channel','Units','Oddzial', 'Dane_Blad','Machine','Serial','Analog_Di
Oddzial = []
Dane_Blad = []
Machine = []
Serial = []
Channel = []
Units = []
Analog_Digital = []

# Druga czesc elementu tablicy np. pierwszego wiersza
Numer_2 = []
Data = []
Czas = []
Numer_5 = []
Numer_6 = []
Numer_7 = []
```

```
# Zbieram dane do listy
jedyna_tablica = jedyna.values.tolist()

#Oddzielam początkowe wiersze
jedyna_tablica_bez_poczatkowych_wierszy = jedyna_tablica[3::]
```

```
# Iterujemy przez cały tablice wierszy
# Przykładowe dane są dla pierwszego wierszu

for i in jedyna_tablica_bez_poczatkowych_wierszy:
    caly_tablica_segment = i[0].split('|')
    print(caly_tablica_segment)
    pierwsza_czesc = caly_tablica_segment[0]
    print("Piersza czesc : ",pierwsza_czesc)
    druga_czesc = caly_tablica_segment[1:]
    print("Druga czesc : ", druga_czesc)
```

```
    # Pracujemy nd pierwsza czescia

    pierwsza_czesc_na_dwa_elementy = pierwsza_czesc.split('_____')
    print("Pierwsza czesc na dwa elementy: ",pierwsza_czesc_na_dwa_elementy)
    if len(pierwsza_czesc_na_dwa_elementy) > 1:
        print("Jesteśmy w środku: ",pierwsza_czesc_na_dwa_elementy[0]) #### <-----Pierwszy z dwoch
        oddzial_data_machine = pierwsza_czesc_na_dwa_elementy[0]
        oddzial = oddzial_data_machine[:2]
        print("Oddzial : ",oddzial)
        dane_blad = oddzial_data_machine[2]
        print("Dane i blad : ",dane_blad)
        machine = oddzial_data_machine[3:6]
        print("Maszyna : ",machine)
        serial = oddzial_data_machine[7:17]
        print("Serial : ", serial)
        channel = oddzial_data_machine[17:]
        print("Channel : ",channel)
        print("Teraz popatrzmy na ten drugi element : ",pierwsza_czesc_na_dwa_elementy[1].lstrip("_"))
        signal = pierwsza_czesc_na_dwa_elementy[1][0]
        print("Sygnal: ",signal)
        units = pierwsza_czesc_na_dwa_elementy[1][1:]
        print("Jednostki : ",units)

    else:
        print("Tutaj trzeba popracowacPIBPAISDGFPUABSDPGBSPDJGHPSDNFGPNSDGPSDGNPSDOGNPSODKGNSPDJOGNSPDG")
        print("Jazda z tym !!!", pierwsza_czesc_na_dwa_elementy[0])
        oddzial_data_machine = pierwsza_czesc_na_dwa_elementy[0]
        oddzial = oddzial_data_machine[:2]
        print("Oddzial : ",oddzial)
        dane_blad = oddzial_data_machine[2]
        print("Dane i blad : ",dane_blad)
        machine = oddzial_data_machine[3:6]
        print("Maszyna : ",machine)
        serial = oddzial_data_machine[7:17]
        print("Serial : ", serial)
        channel= oddzial_data_machine[17:26].rstrip("_")
        print("Channel : ",channel)
        signal_units = oddzial_data_machine[26:].lstrip("_")
        signal = signal_units[0]
        print("Sygnal: ",signal)
        units = signal_units[1:]
        print("Jednostki : ",units)
```

```
    # Pracujemy nad druga i dodajemy do tablic

    Numer_2.append(druga_czesc[0])
    Data.append(druga_czesc[1])
    Czas.append(druga_czesc[2])
    Numer_5.append(druga_czesc[3])
    Numer_6.append(druga_czesc[4])
    Numer_7.append(druga_czesc[5])

    Oddzial.append(oddzial)
    Dane_Blad.append(dane_blad)
    Machine.append(machine)
    Serial.append(serial)
    Channel.append(channel)
    Units.append(units)
    Analog_Digital.append(signal)
```

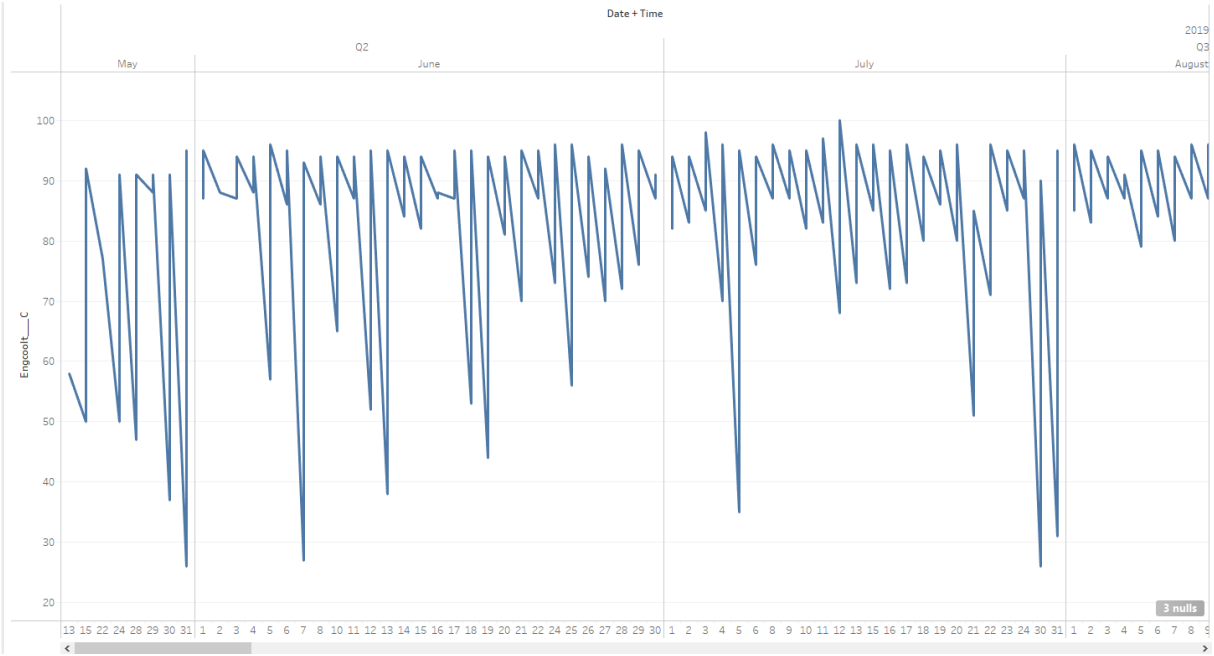
```
df = pd.DataFrame(
    {
        "Numer_2": Numer_2,
        "Data": Data,
        "Czas": Czas,
        "Numer_5": Numer_5,
        "Numer_6": Numer_6,
        "Numer_7": Numer_7,
        "Oddzial": Oddzial,
        "Dane_Blad": Dane_Blad,
        "Machine": Machine,
        "Serial": Serial,
        "Channel": Channel,
        "Units": Units,
        "Analog_Digital": Analog_Digital,
        }
#     index= np.arange(1,len(jedyna_tablica_bez_poczatkowych_wierszy))
)
```

Kolejnym etapem była analiza danych w celu wyszukania anomalii, które mogą świadczyć o zbliżającej się awarii.

Głównym wyznacznikiem powinny być analizy błędów, jednakże wobec braku informacji na temat kodów błędów skupiliśmy się na analizie dostępnych danych.

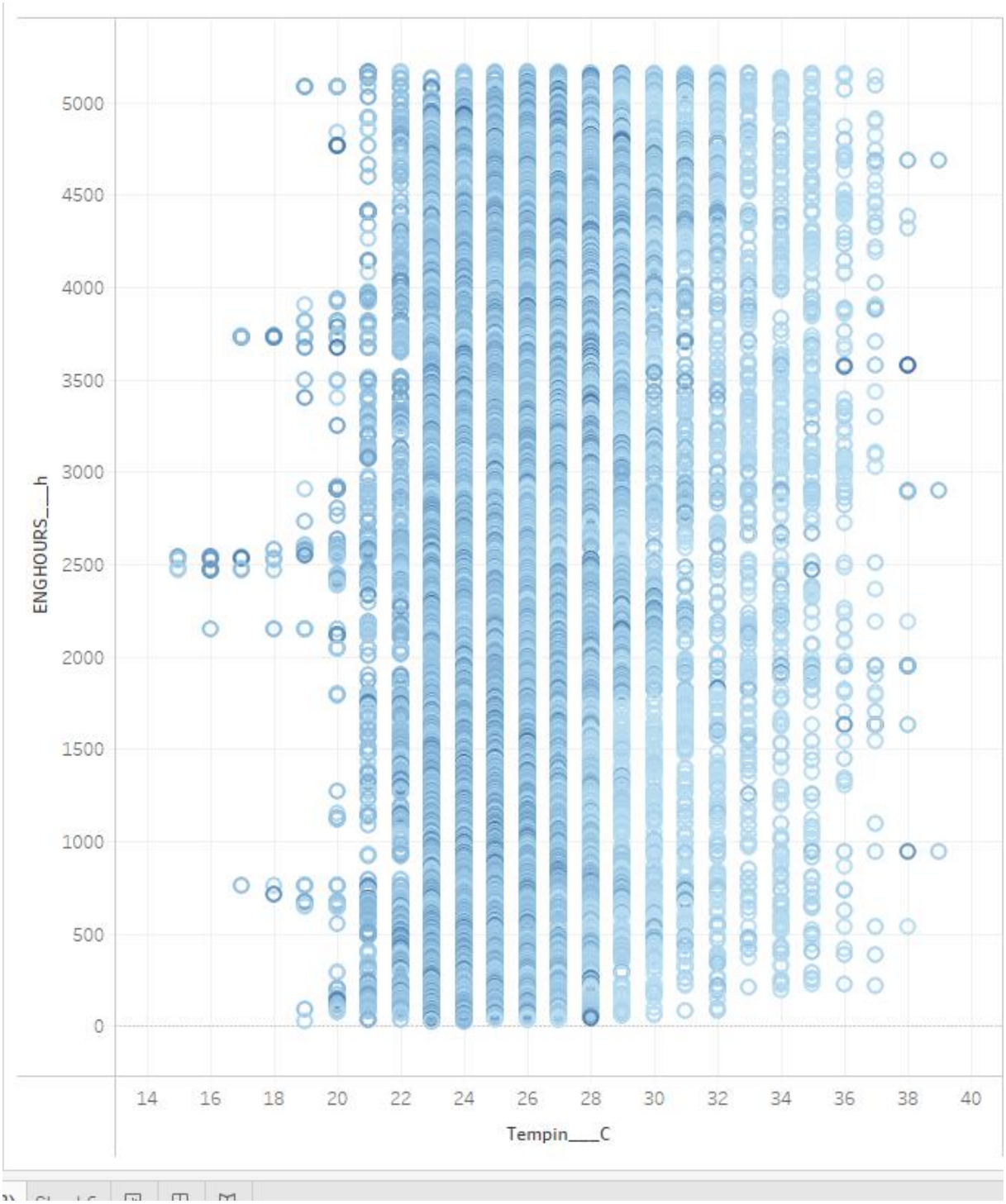
Wykonaliśmy szereg wizualizacji w programie Tableau. Poniżej przykładowe diagramy.

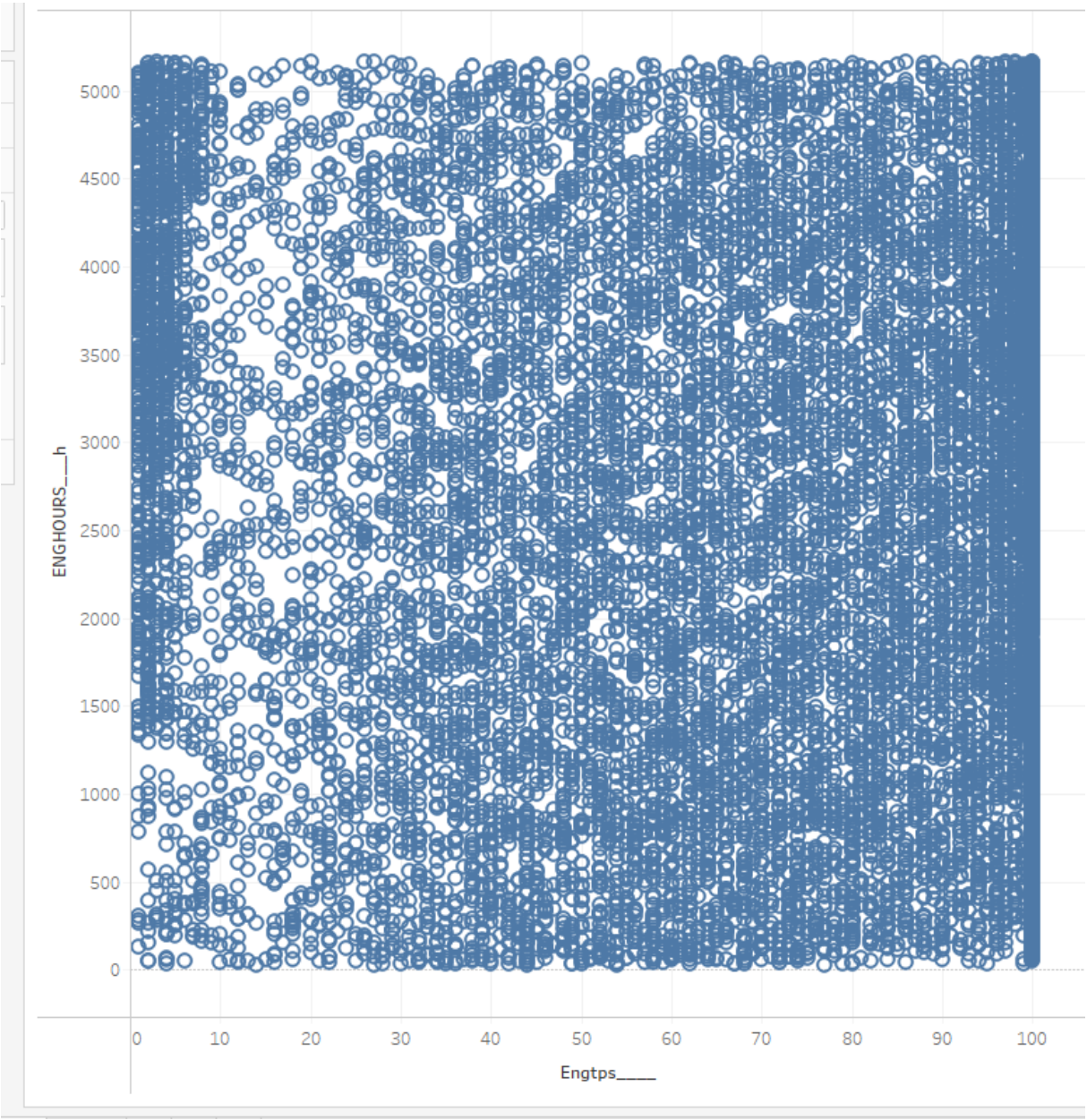
Rysunek 1. Wykres temeratury



Rysunek 2. Wykres obrotów silnika

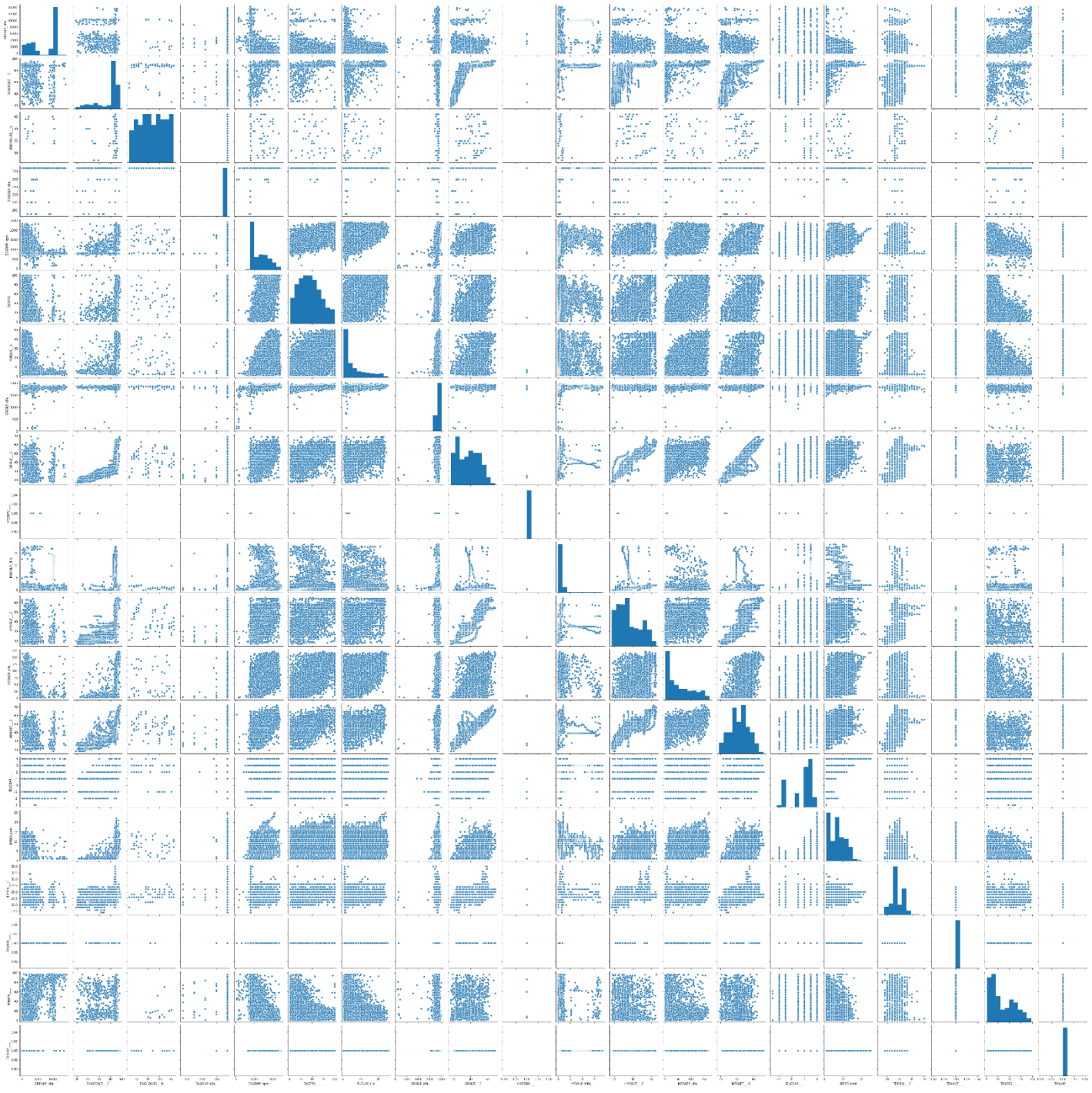
Szukając wydarzeń szczególnych znaleźliśmy kilka maszyn które wychodzą po za swój zasięg obrotów.


ENGHOURS___h
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
14
16
18
20
22
24
26
28
30
32
34
36
38
40
Tempin___C

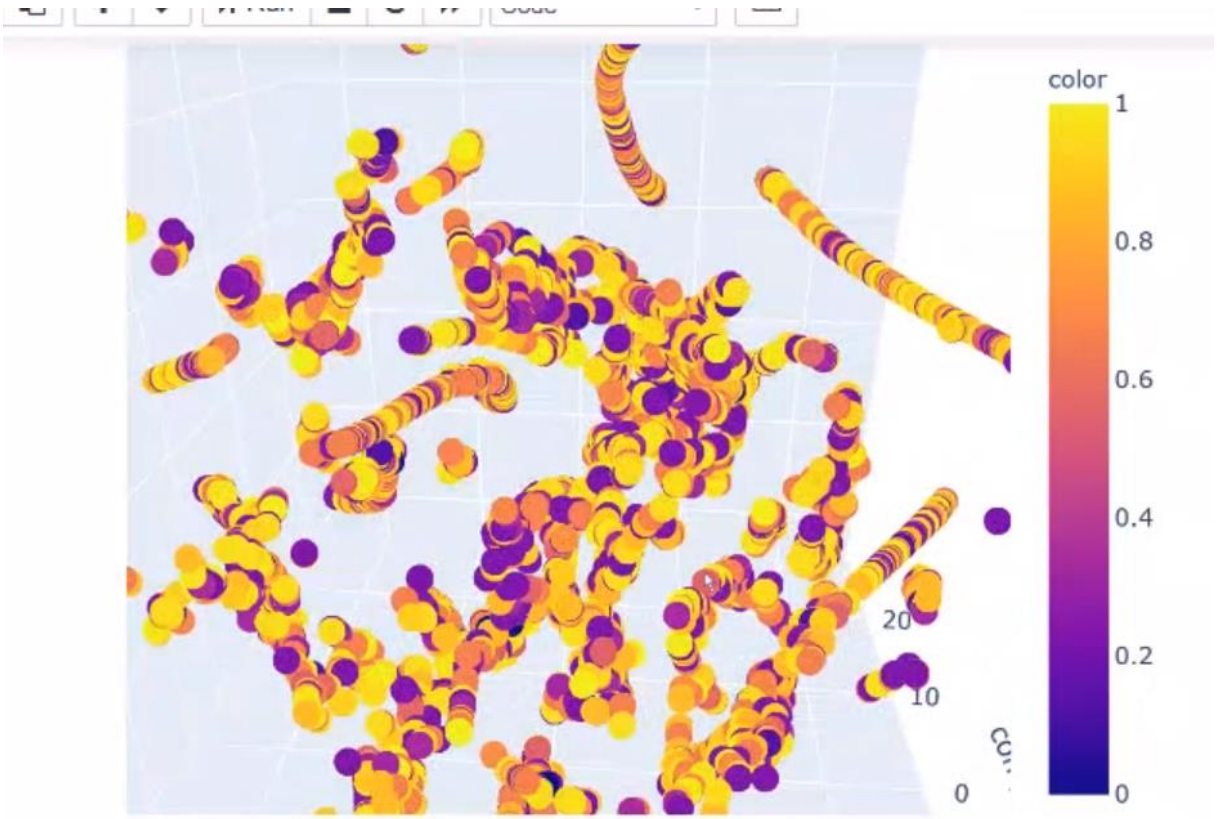

ENGHOURS___h
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90
100
Engtps____

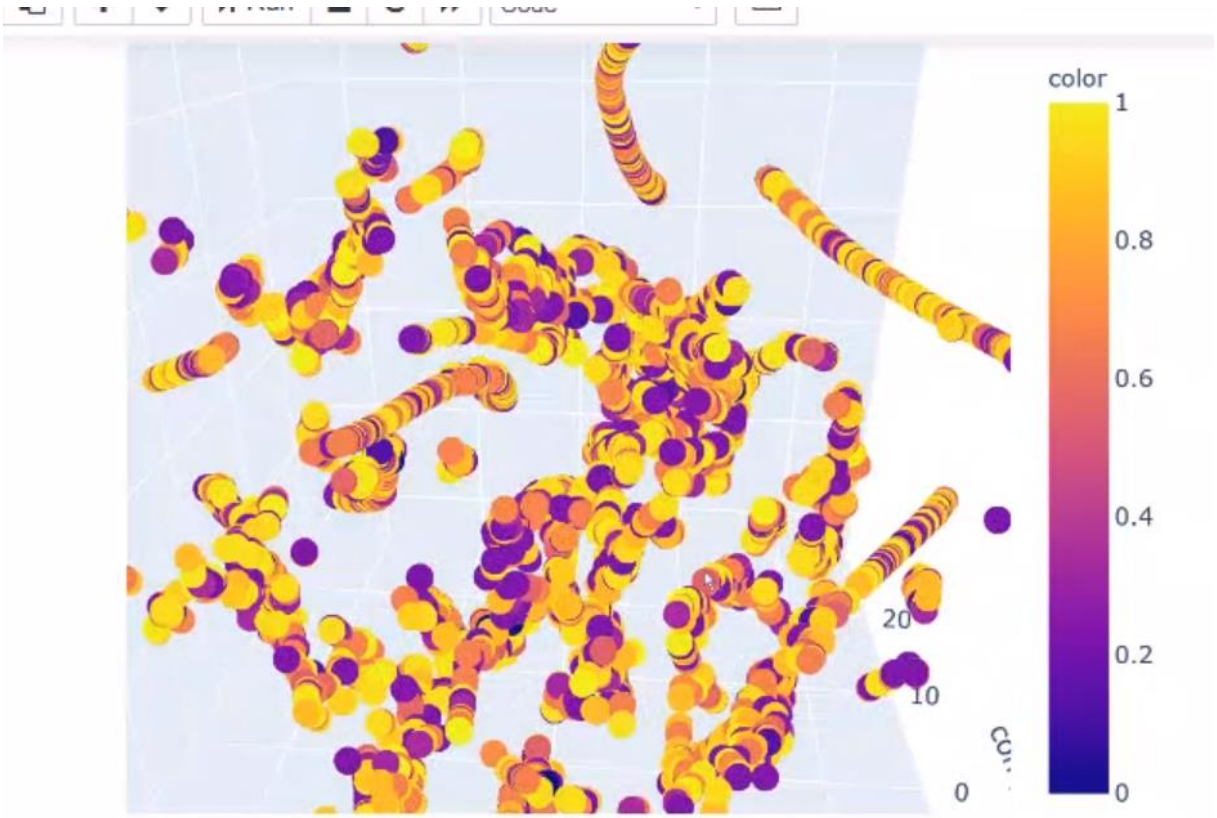
Badaliśmy również korelację między zmiennymi.

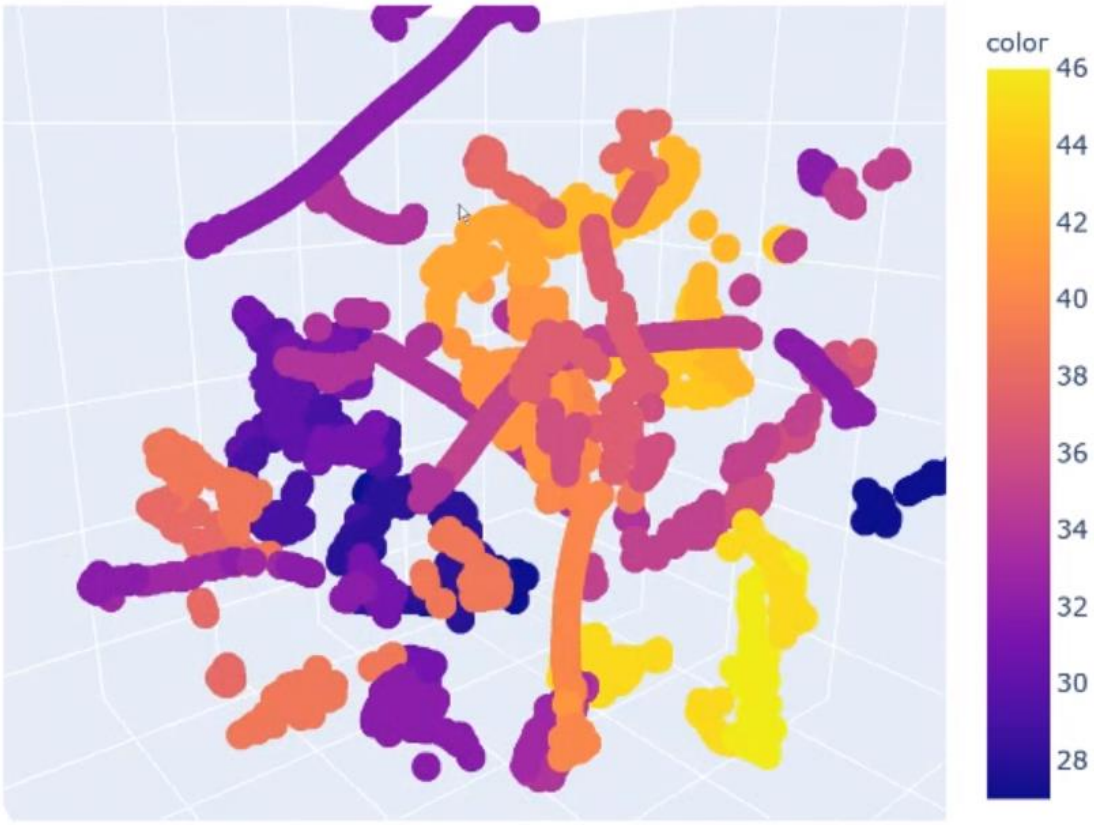
Rysunek 3. Wykres korelacji par zmiennych

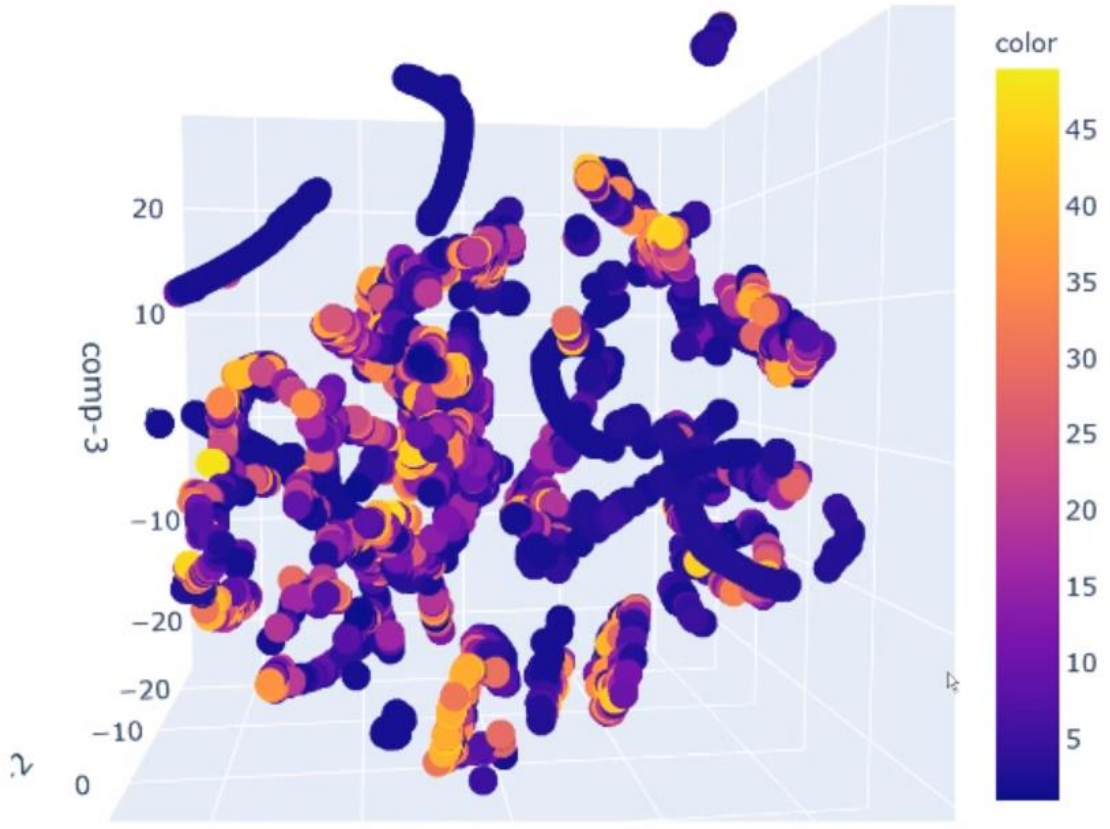


Pozostałe wizualizacje


color
1
0.8
0.6
0.4
0.2
0
20
10
0


color
1
0.8
0.6
0.4
0.2
0
20
10
0


color
46
44
42
40
38
36
34
32
30
28



Podsumowanie

1. Napotkaliśmy problemy dotyczące niewystarczającej mocy obliczeniowej. Rozmiar danych przekraczał wielkość dostępnej pamięci RAM. Z tego względu skupiliśmy się na analizie jednej maszyny.
2. Niewystarczające informacje o awariach, trudność w określeniu jaki zestaw parametrów wystąpił w chwili awarii, uniemożliwiły opracowanie modelu predykcji, jednakże zidentyfikowaliśmy czynniki ryzyka wystąpienia awarii.
3. Z uwagi na ograniczenia czasowe nie udało nam się zbudować kompletnego rozwiązania. Mamy wiele pomysłów co można by dalej zrobić, m.in. obliczenie Load Factor, czyli obciążenia silnika, wykorzystanie modelu regresji logistycznej lub XGBoost.
4. Hackaton CuValley to wspaniała inicjatywa 😊 Jesteśmy zadowoleni że mogliśmy wziąć w nim udział. Sprawdziliśmy swoją zdolność do pracy w grupie i podnieśliśmy swoje umiejętności. Panowała serdeczna atmosfera zdrowej rywalizacji. Gratulacje dla Organizatorów i życzymy więcej takich eventów 😊

NatLuk Team

1. Lider - Łukasz Barwicki – Discord: Xarabek#8165
2. Krzysztof Michalski - KrzysiekDev#9704
3. Agnieszka Dąbrowska - Agnieszka#0218
4. Patryk K - Patryk_K#8812
5. Hiszoan#7753