
# Sztuczne sieci neuronowe

## Rozpoznawanie awarii urządzenia
## Modele oparte o CNN

Weronika Ciurej
Aleksandra Rolka
Patryk Śledź

Informatyka Stosowana
Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza w Krakowie

# Spis treści

# 1. Cel projektu

Celem projektu jest zastosowanie modelu opartego o CNN do rozpoznawania awarii urządzenia.
Projekt zakłada następujące etapy implementowania rozwiązania:

1. Przygotowanie danych:
    a. Statystyczna analiza danych
    b. Normalizacja
    c. Grupowanie danych poszczególnych minut - tworzenie szeregów czasowych o różnych długościach
    d. Przygotowanie danych do walidacji CV5
2. Przygotowanie własnej architektury SSN do klasyfikacji
3. Uczenie i testowanie sieci neuronowej
4. Porównanie wyników w kontekście zmian w topologii.

# 2. Wykorzystane technologie

Projekt został wykonany w Python3 z wykorzystaniem środowiska Jupyter Notebook, udostępnionym w ramach usługi cloudowej Google Colabolatory.
Dodatkowe biblioteki, które zostały wykorzystane:

- **keras** - biblioteka umożliwiła zaprojektowanie modelu sieci konwolucyjnej CNN,
- **pandas** - wykorzystywana do operacji na zbiorach danych - wczytywanie, analiza,
- **sklearn** - biblioteka posiadająca wiele przydatnych funkcji, pozwoliła przeprowadzić cross-walidację CV5, standaryzację danych oraz umożliwiła analizę poprawności modeli zwracając parametry takie jak precyzja, recall, f1 score,
- **seaborn** - biblioteka pozwalająca tworzyć przejrzyste wykresy - w naszym przypadku heatmape korelacji,
- **numpy**,
- **matplotlib**.

# 3. Uzyskane wyniki

## 3.1. Statystyczna analiza danych

Dane wejściowe zostały umieszczone w plikach CSV i podzielono je na dwie grupy:

- dataN_fault.csv - pliki zawierające informację o awarii urządzenia,
- dataM_normal.csv - pliki zawierające informację o poprawnym działaniu urządzenia,

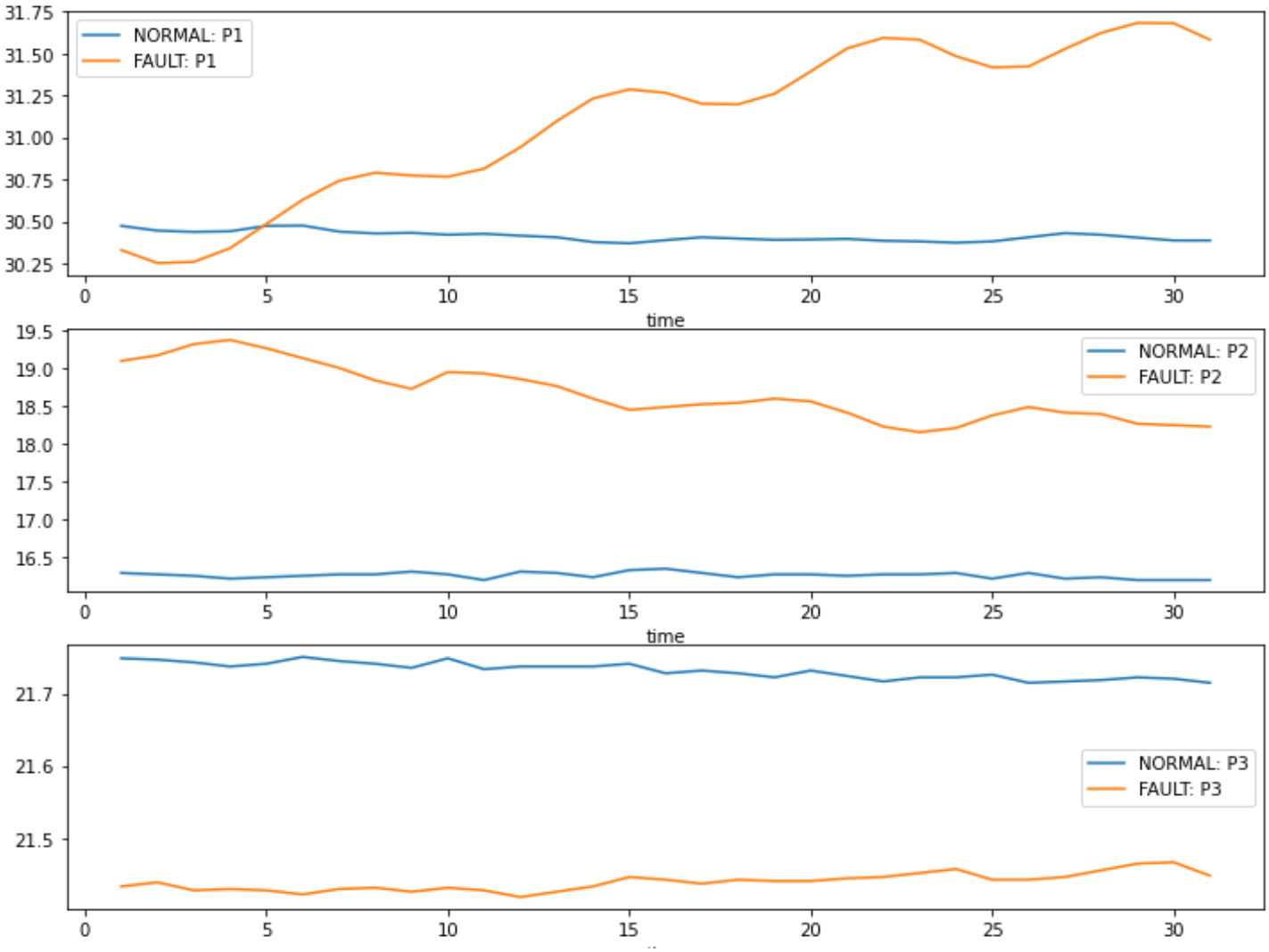
gdzie N jest numerem porządkowym z przedziału [1, 54], a M z przedziału [1, 53].

Każdy z wyżej wymienionych plików posiada ten sam schemat kolumn przedstawiony w tabeli poniżej:

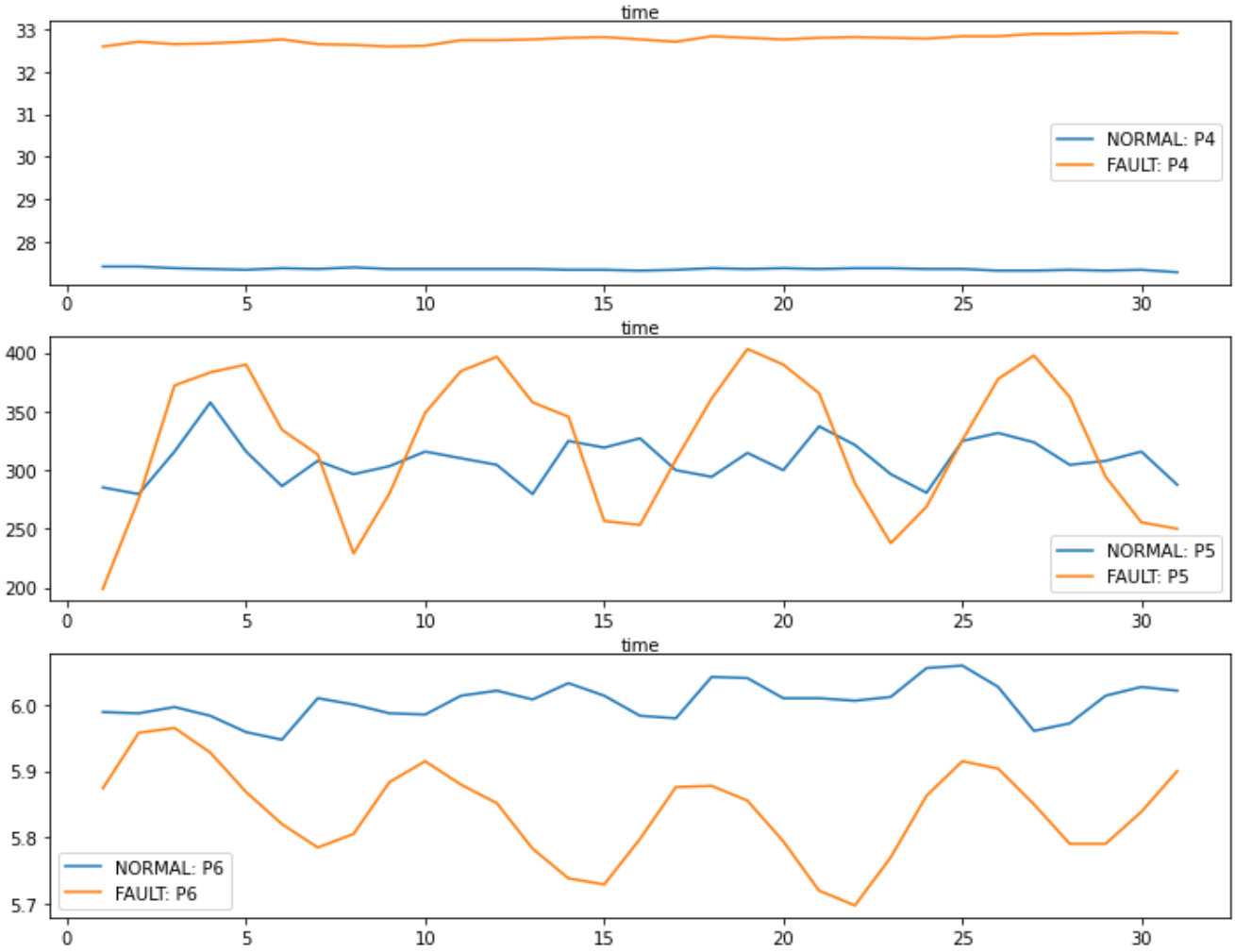
| Time | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 30.4 | 20 | 21.7 | 34 | 60 | 5.9 | 42.6 | 34.9 | 16.9 |
| 2 | 30.2 | 21 | 21.6 | 34 | 660 | 6.2 | 62 | 38.3 | 17 |
| 3 | 30.4 | 21 | 21.6 | 34 | 600 | 6.1 | 70.4 | 40.6 | 16.2 |
| 4 | 30.9 | 20 | 21.6 | 34 | 660 | 5.8 | 71.9 | 40.6 | 15.8 |
| 5 | 31.3 | 20 | 21.7 | 34 | 180 | 5.4 | 69.3 | 38.8 | 15.4 |

*Tabela 1 Poglądowe wartości parametrów urządzeń dla 5 minut*

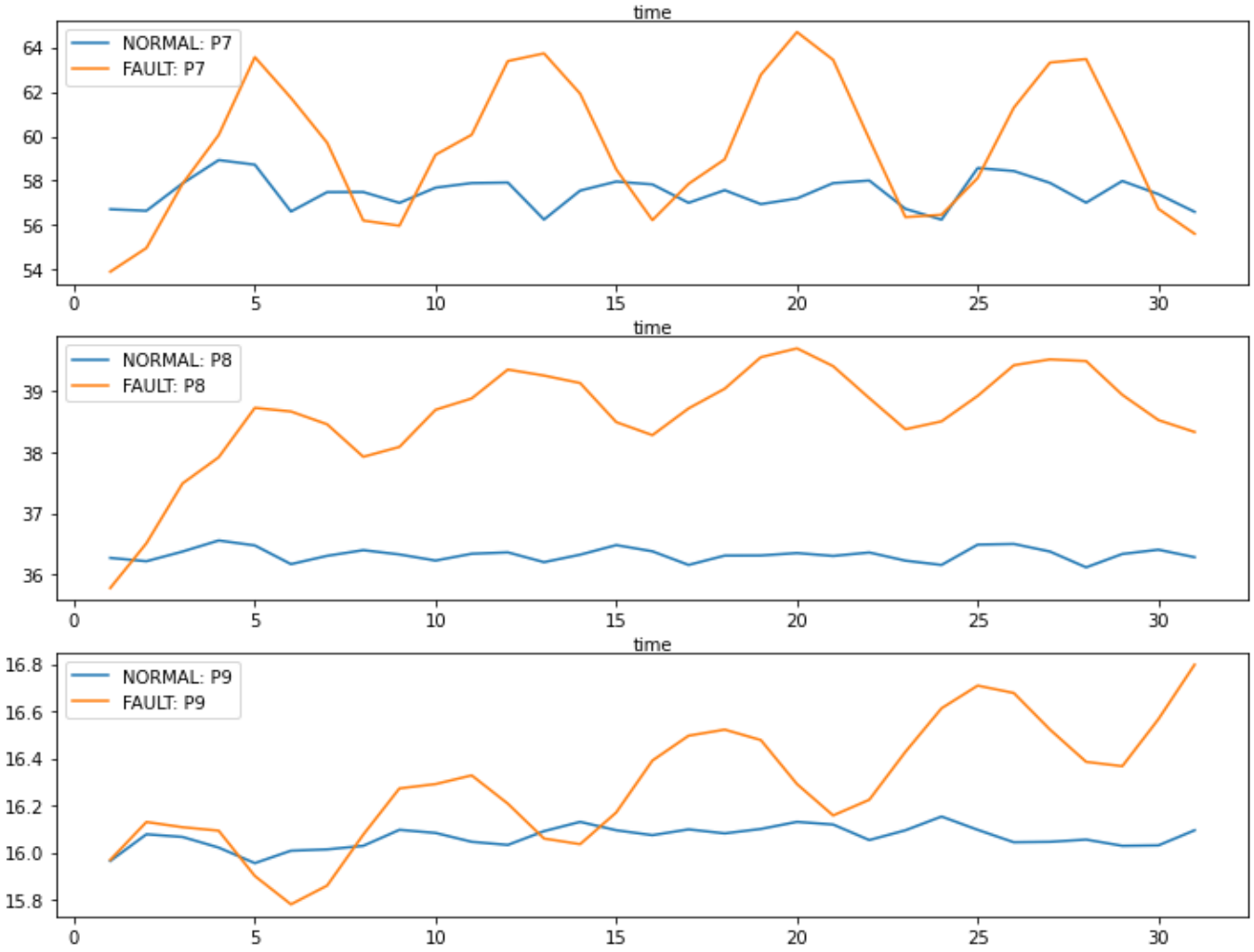
Pierwszym etapem analizy było sprawdzenie zachowania się poszczególnych parametrów od P1 do P9 w czasie t. Postanowiono uśrednić wartości godzinowe ze wszystkich dni oddzielnie dla pomiarów z awarią i sprawnym urządzeniem. Wyniki przedstawiono na poniższych wykresach.




*Rysunek 3.1 Średnie wartości parametrów P1, P2, P3 dla 31 minut*




*Rysunek 3.2 Średnie wartości parametrów P4, P5, P6 dla 31 minut*




*Rysunek 3.3 Średnie wartości parametrów P7, P8, P9 dla 31 minut*

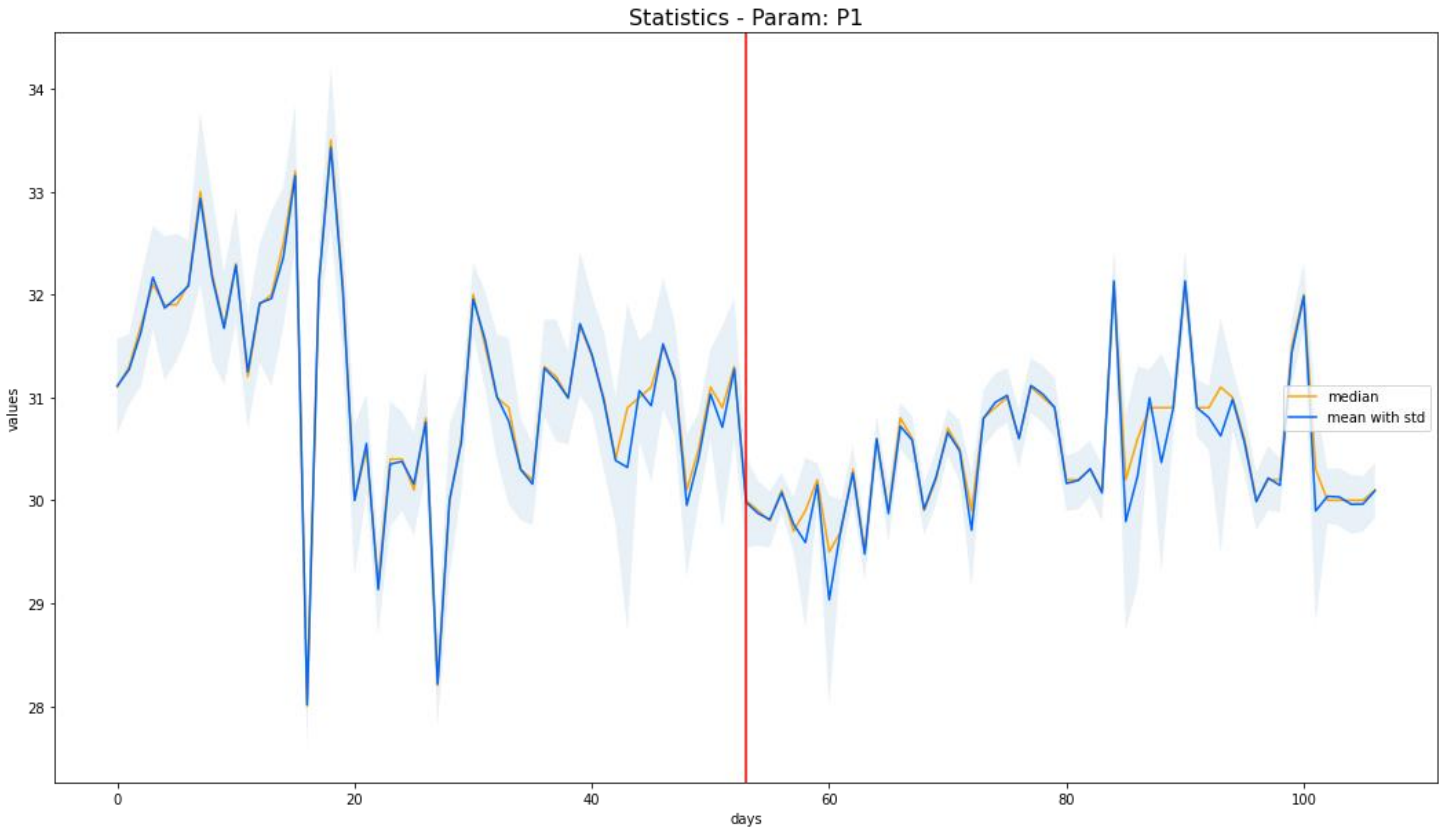
Jak można zauważyć na powyższych wykresach dla niektórych parametrów widać znaczną różnicę sygnalizującą awarię urządzenia. Przykładem może być wykres dla parametru P3, gdzie dla urządzenia pracującego poprawnie wartość parametru oscyluje w okolicy 21.8, natomiast dla awarii wartość ta spadła do okolic 21.4.

Dalszą analizę danych podzielono na dwie części:

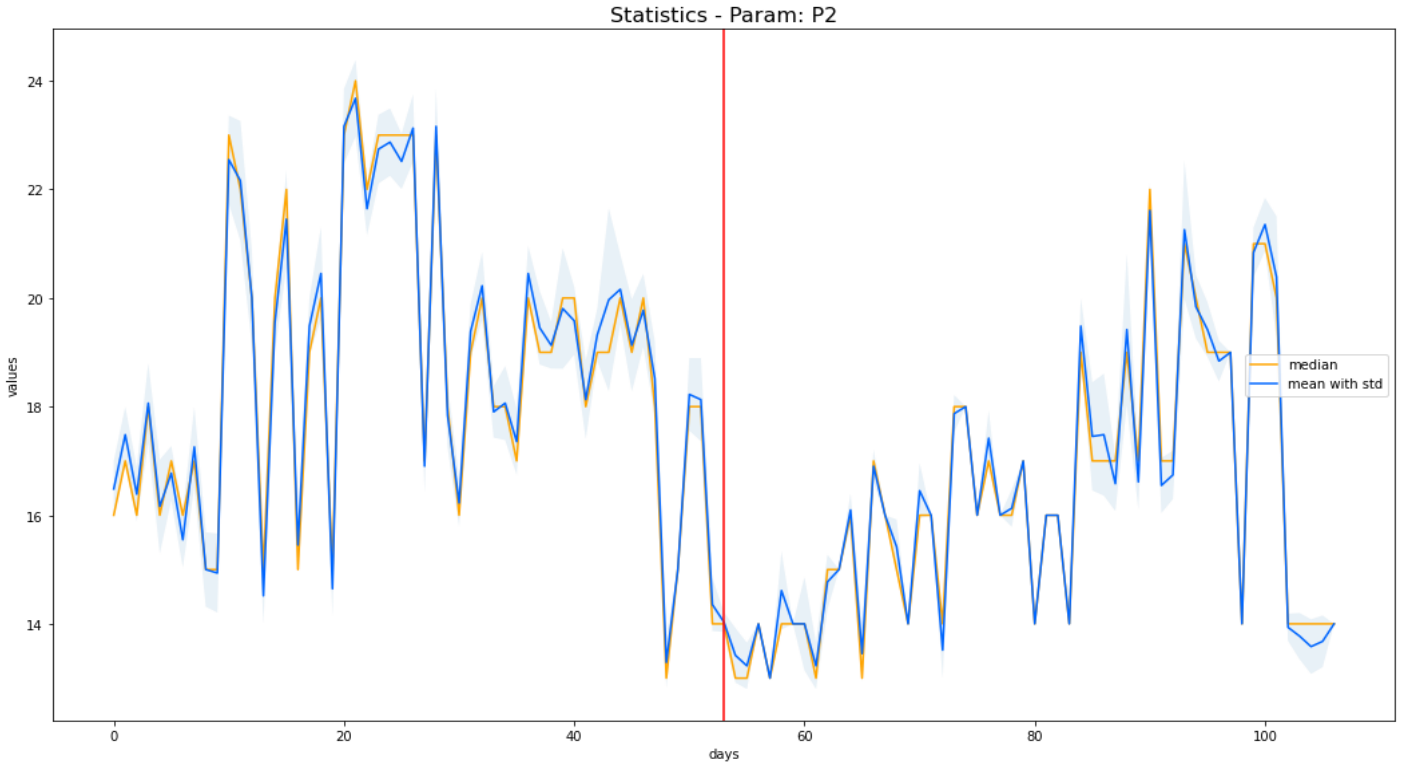
- analizę danych bez standaryzacji,
- analizę danych ustandaryzowanych.

### 3.1.1. Dane bez standaryzacji - średnia, mediana, odchylenie

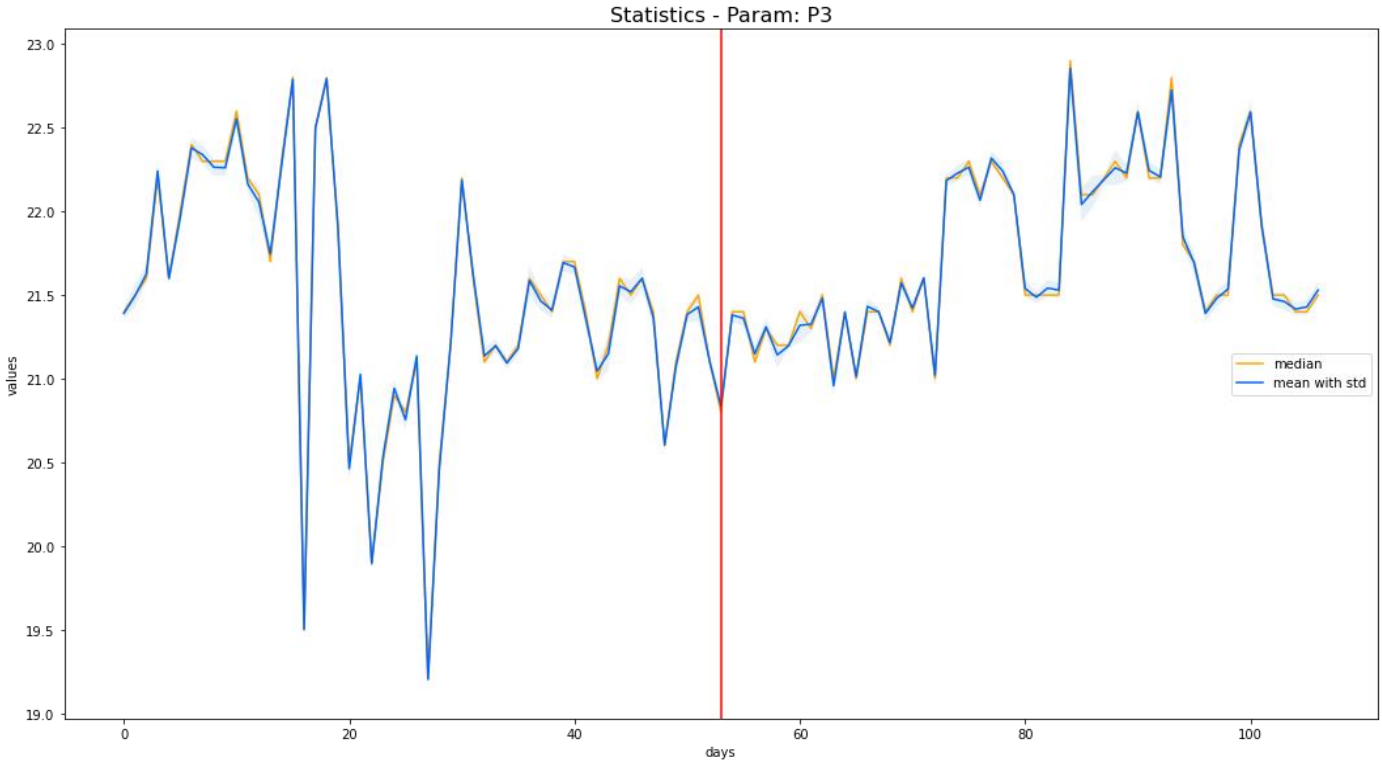
Wykorzystując wbudowaną funkcjonalność obiektu DataFrame biblioteki Pandas – describe jesteśmy w stanie zwrócić wiele interesujących parametrów wczytanych danych. Wybrano średnią, medianę oraz odchylenie standardowe w celu wizualizacji statystycznej wczytanych danych. W celu lepszego przedstawienia różnic na jeden wykres naniesiono wartości dla pomiarów z urządzeń działających poprawnie (wykres po lewej) oraz tych z awarią (wykres po prawej).



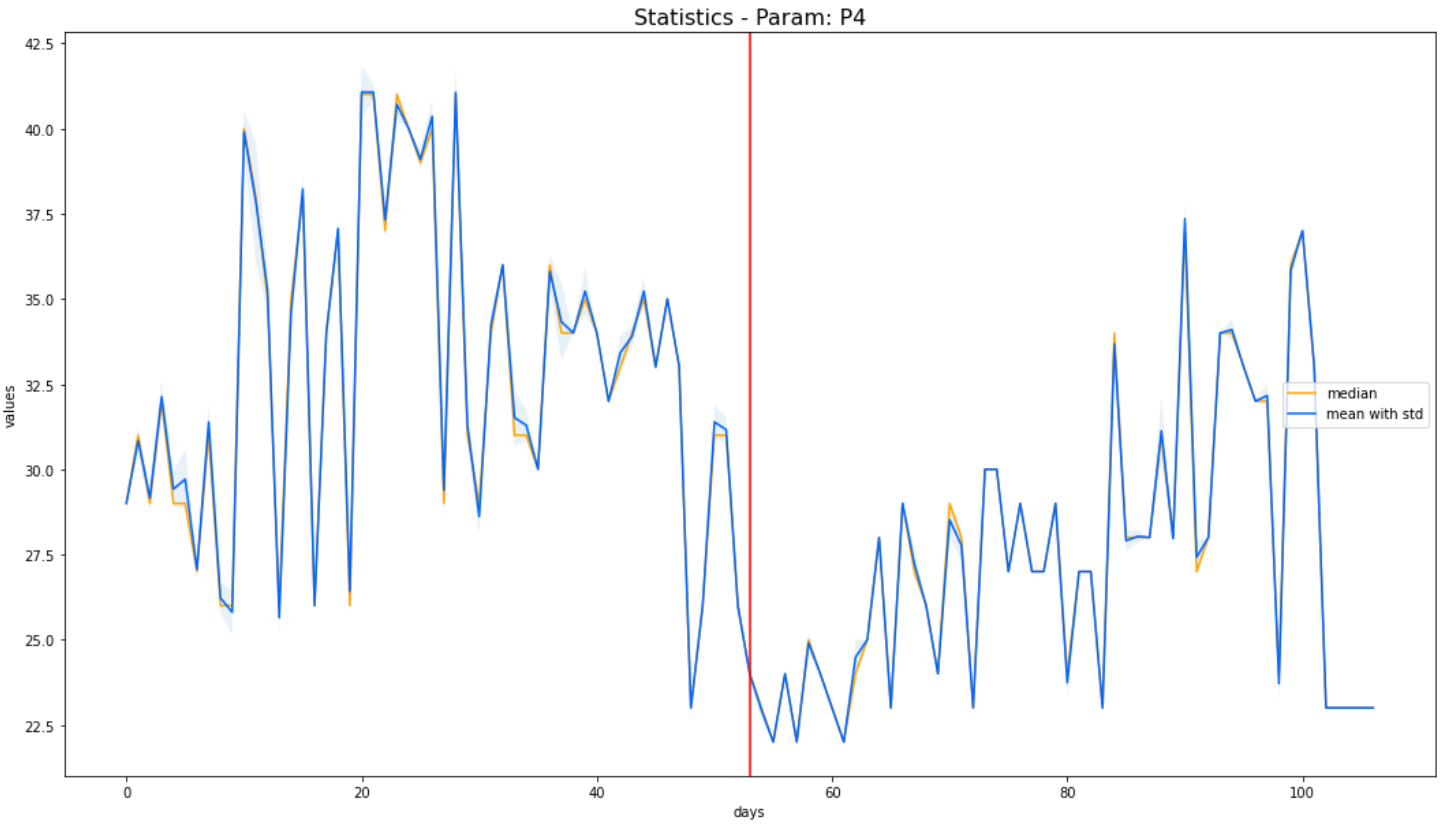

*Rysunek 3.4 Średnia, mediana, odchylenie dla parametru P1. Po lewej brak awarii, po prawej awaria*



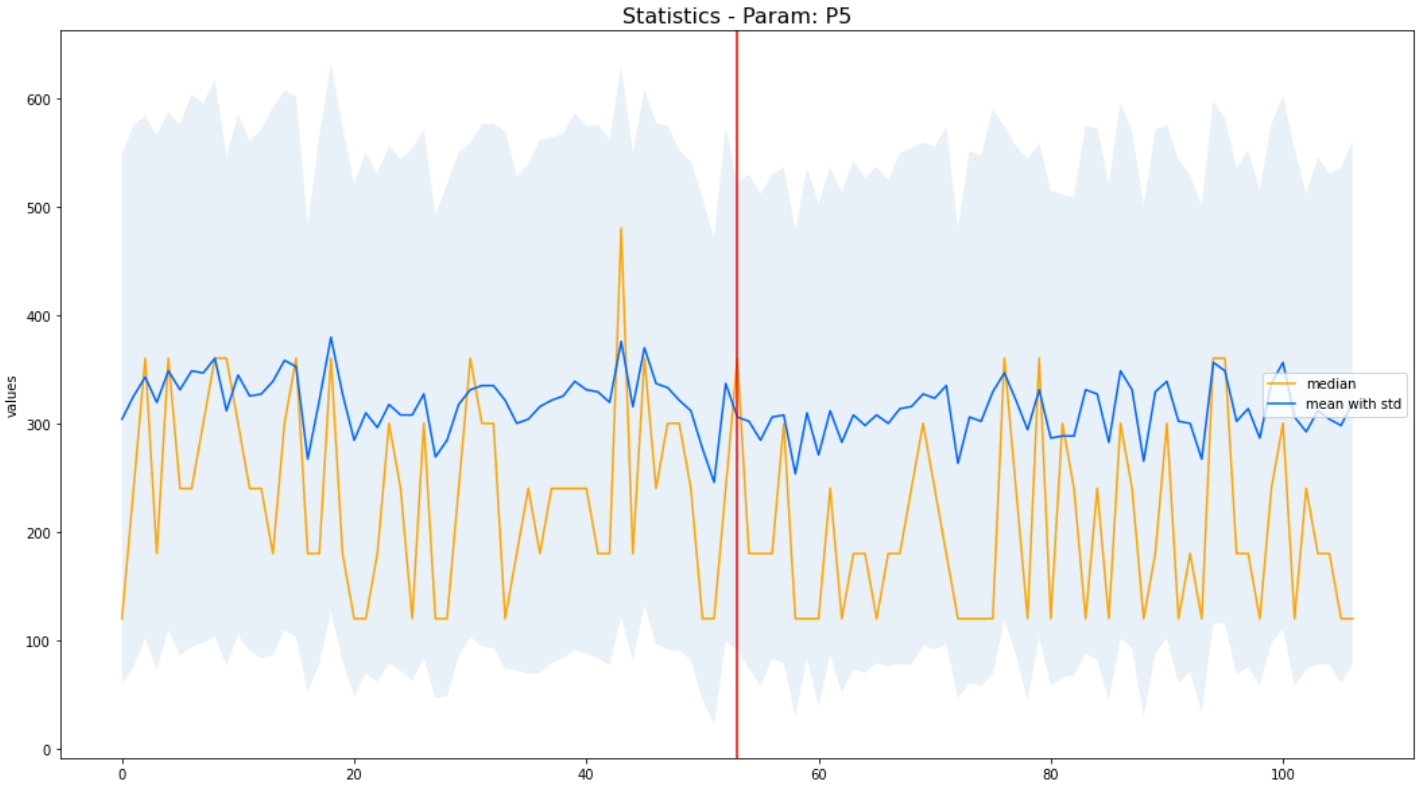

*Rysunek 3.5 Średnia, mediana, odchylenie dla parametru P2. Po lewej brak awarii, po prawej awaria*



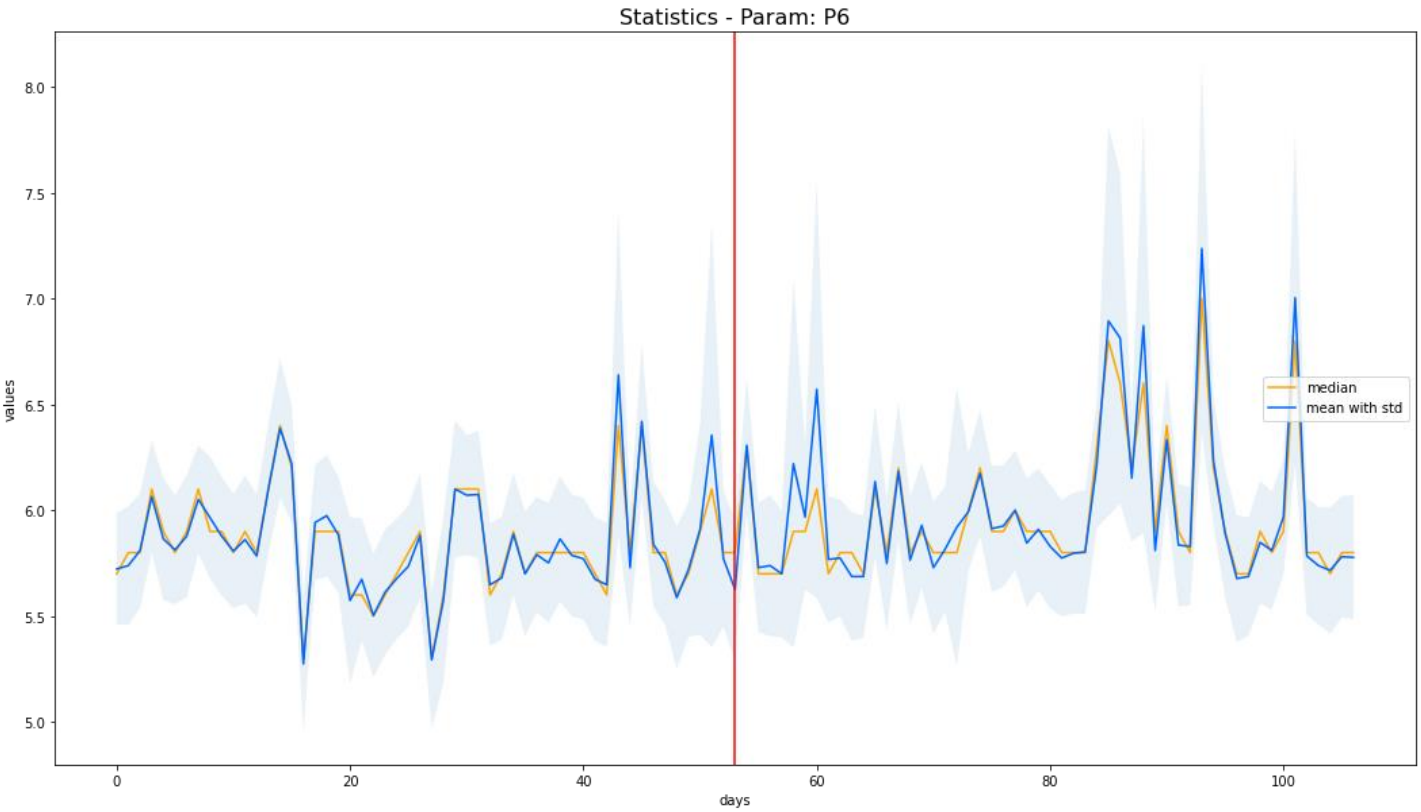

*Rysunek 3.6 Średnia, mediana, odchylenie dla parametru P3. Po lewej brak awarii, po prawej awaria*



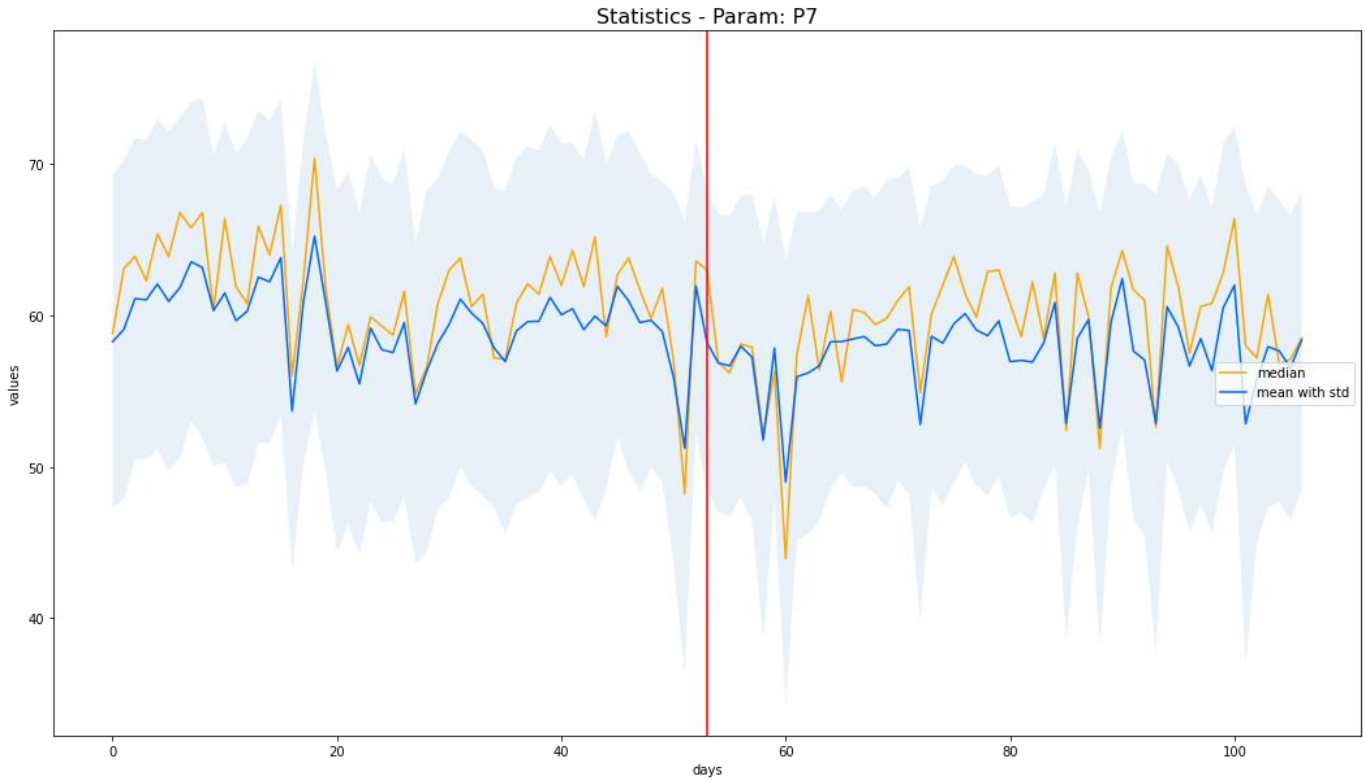

*Rysunek 3.7 Średnia, mediana, odchylenie dla parametru P4. Po lewej brak awarii, po prawej awaria*



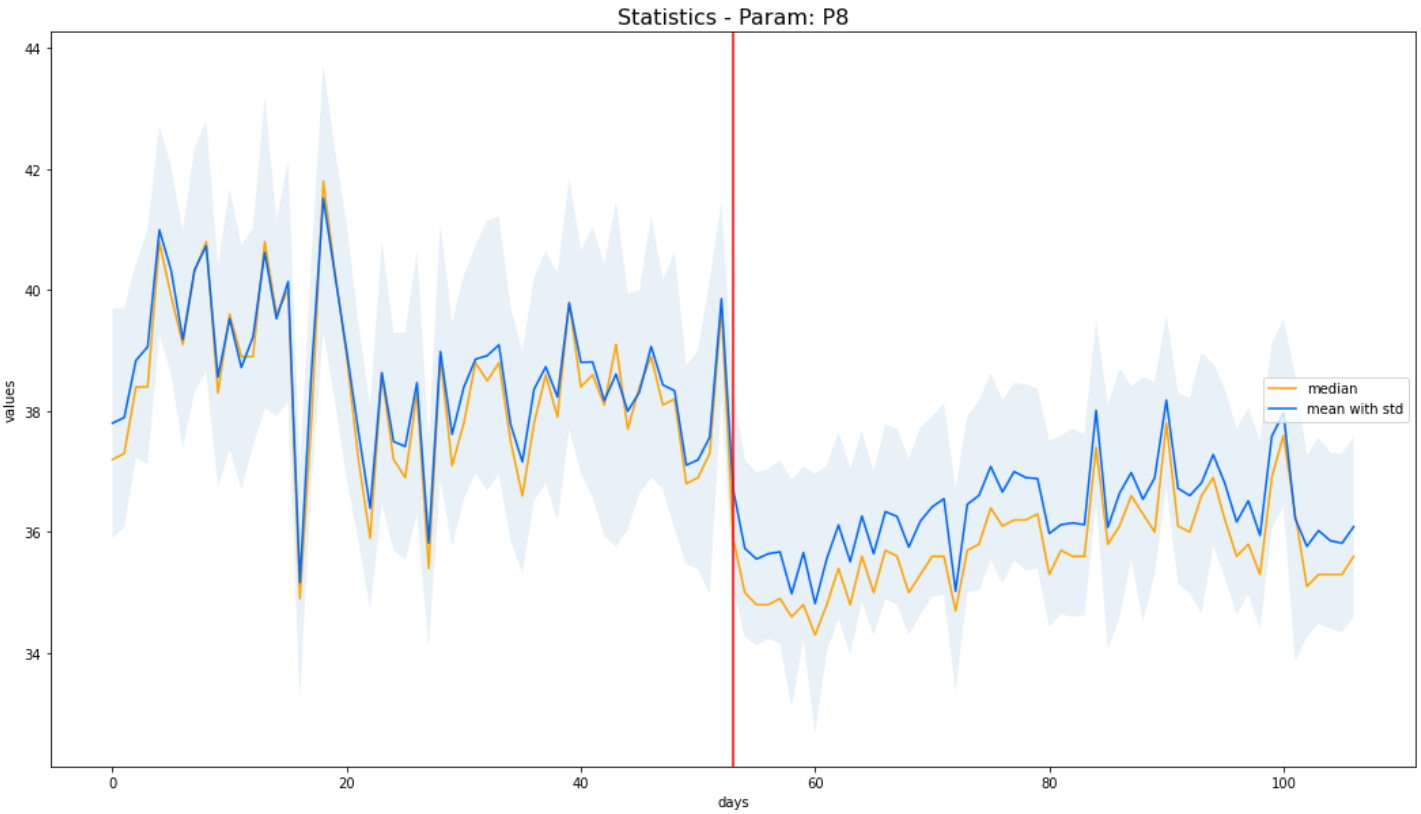

*Rysunek 3.8 Średnia, mediana, odchylenie dla parametru P5. Po lewej brak awarii, po prawej awaria*



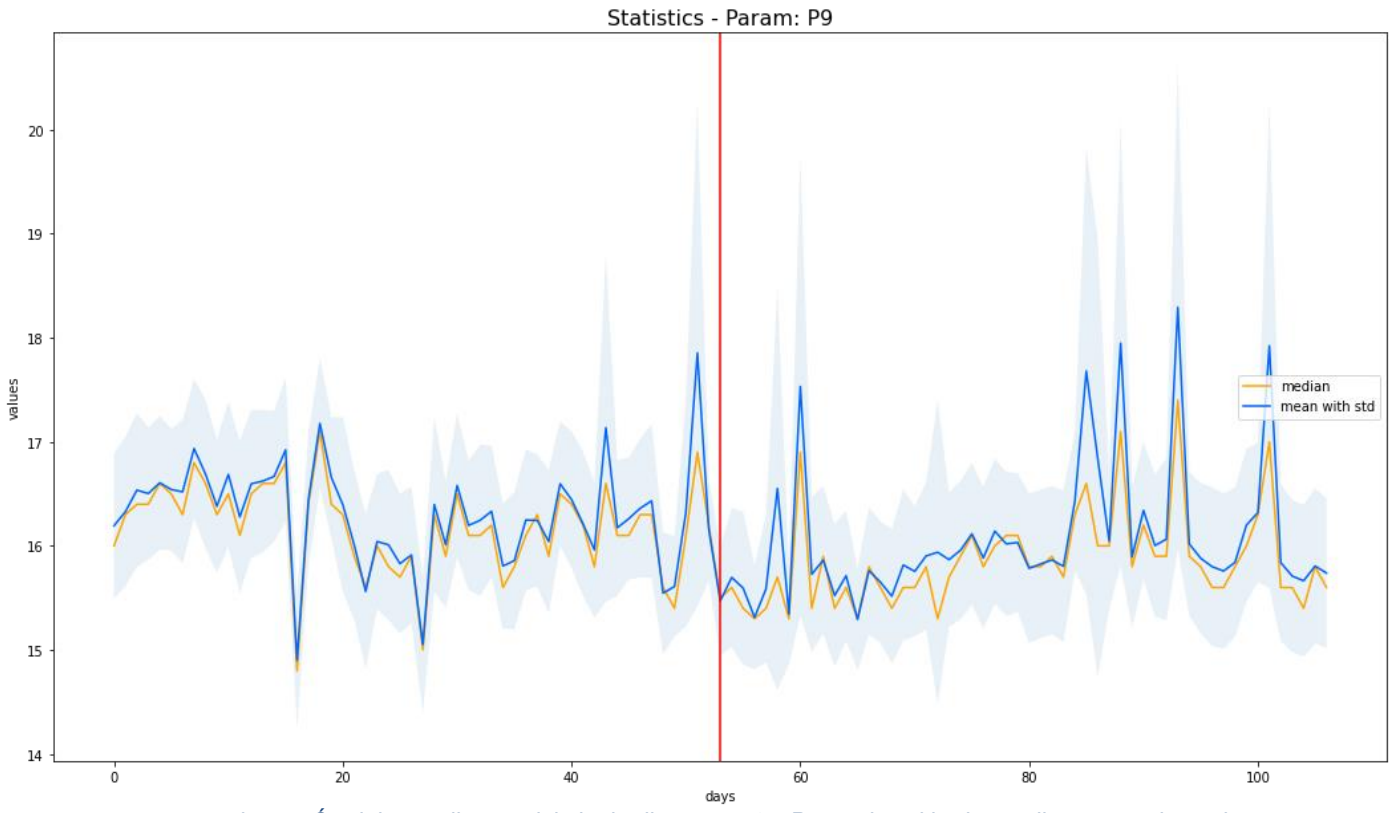

*Rysunek 3.9 Średnia, mediana, odchylenie dla parametru P6. Po lewej brak awarii, po prawej awaria*




*Rysunek 3.10 Średnia, mediana, odchylenie dla parametru P7. Po lewej brak awarii, po prawej awaria*




*Rysunek 3.11 Średnia, mediana, odchylenie dla parametru P8. Po lewej brak awarii, po prawej awaria*

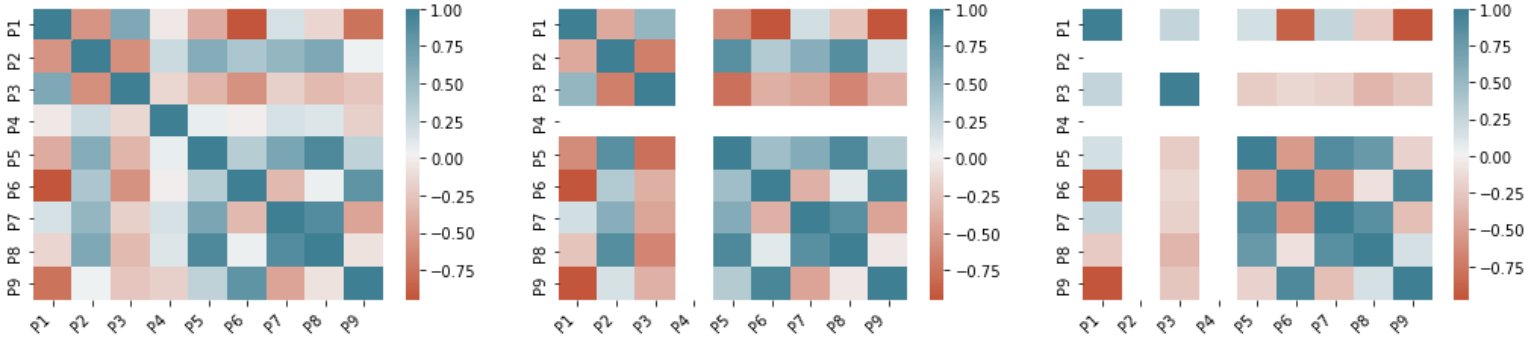



*Rysunek 3.12 Średnia, mediana, odchylenie dla parametru P9. Po lewej brak awarii, po prawej awaria*

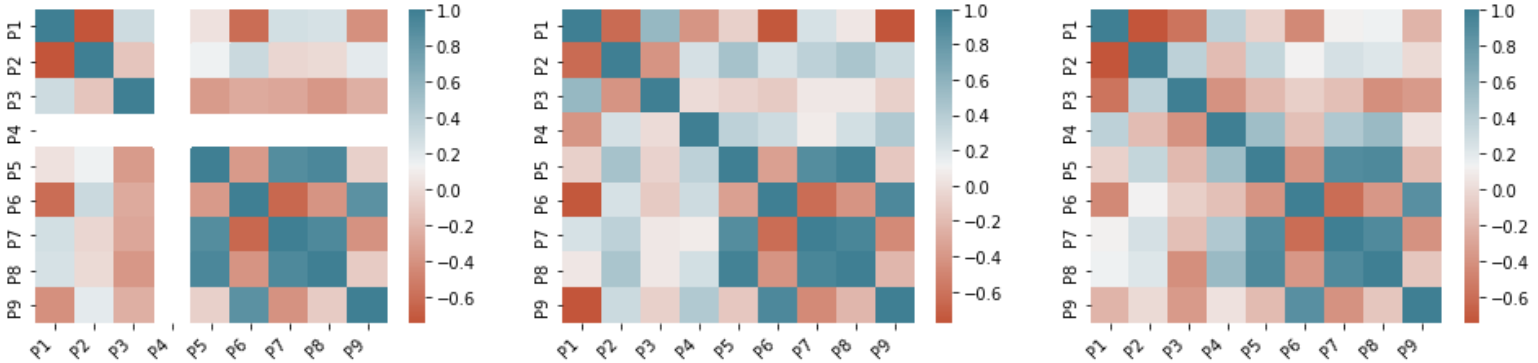
### 3.1.2. Dane bez standaryzacji - korelacja parametrów

Używając biblioteki Seaborn wykonano szereg wykresów reprezentujących korelację pomiędzy parametrami. Na poniższych rysunkach przedstawiono wyniki dla trzech kolejnych dni osobno dla pomiarów z poprawnym działaniem oraz awarią. Można zauważyć znaczne różnice w wartości korelacji dla poszczególnych parametrów.

Otrzymane wykresy nie pozwoliły jednoznacznie stwierdzić, które parametry można by było pominąć. Mimo wysokiej korelacji parametrów w pewnych dniach, wielkość tej korelacji w kolejnych dniach była inna. Dlatego zdecydowaliśmy się przy trenowaniu i testowaniu modelu, wziąć pod uwagę wszystkie parametry.

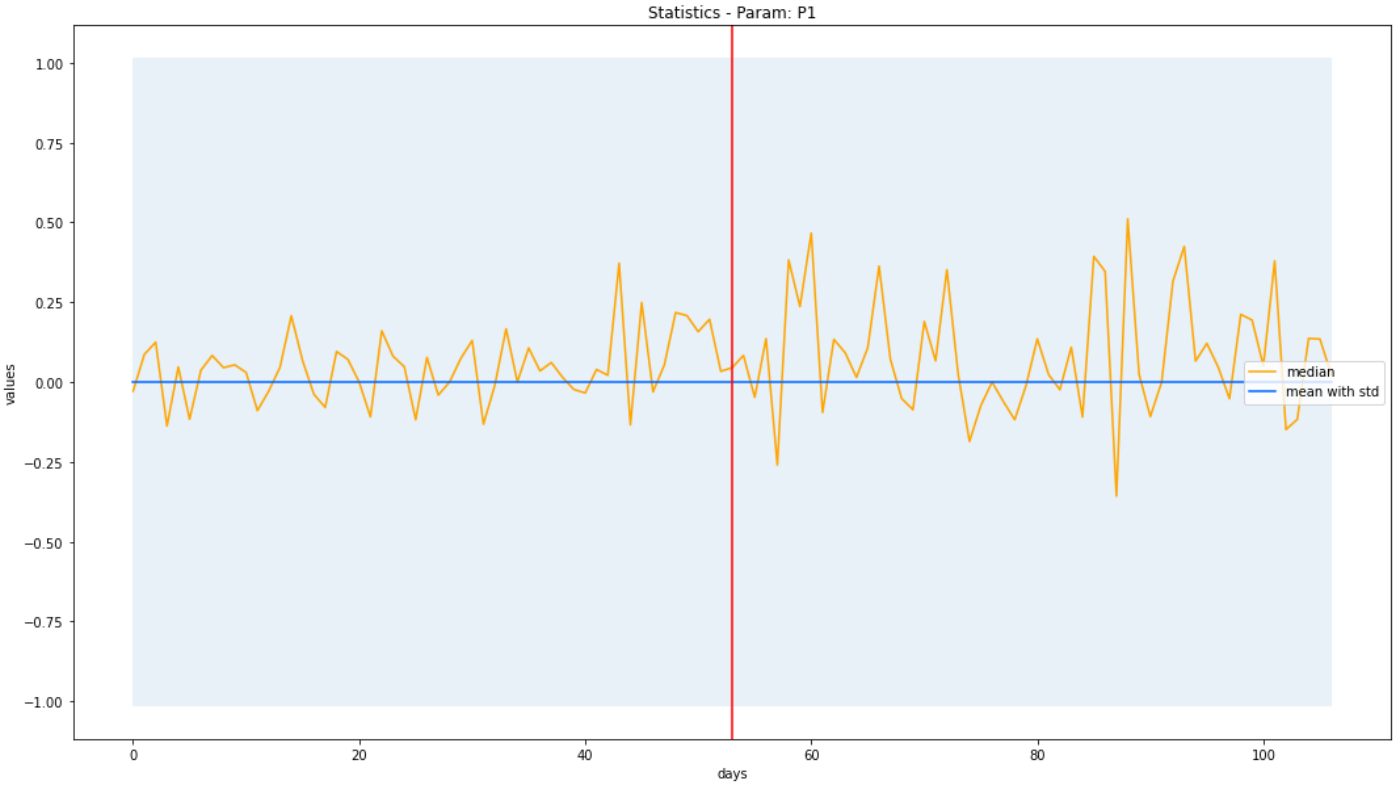



*Rysunek 3.13 Korelacja parametrów dla wybranych 3 dni - poprawne działanie*

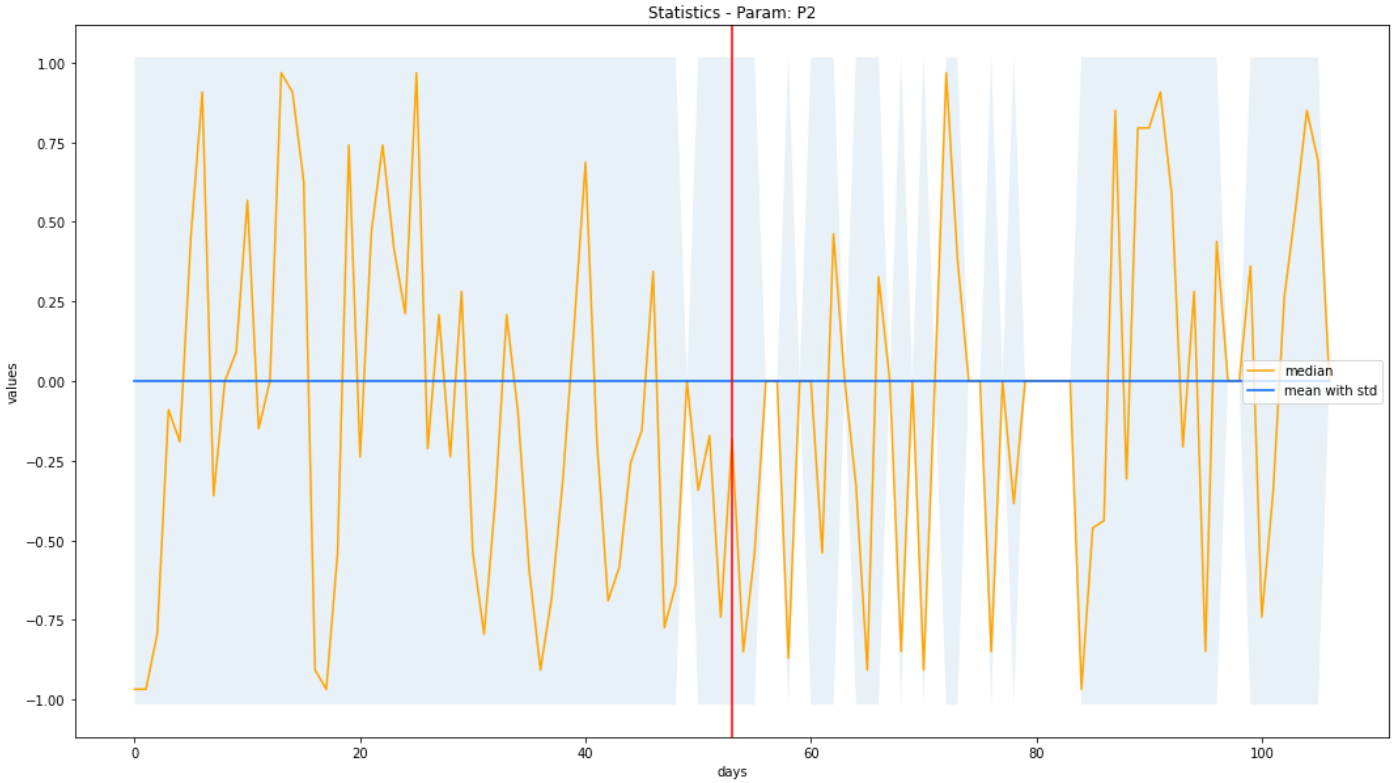



*Rysunek 3.14 Korelacja parametrów dla wybranych 3 dni - awaria urządzenia*

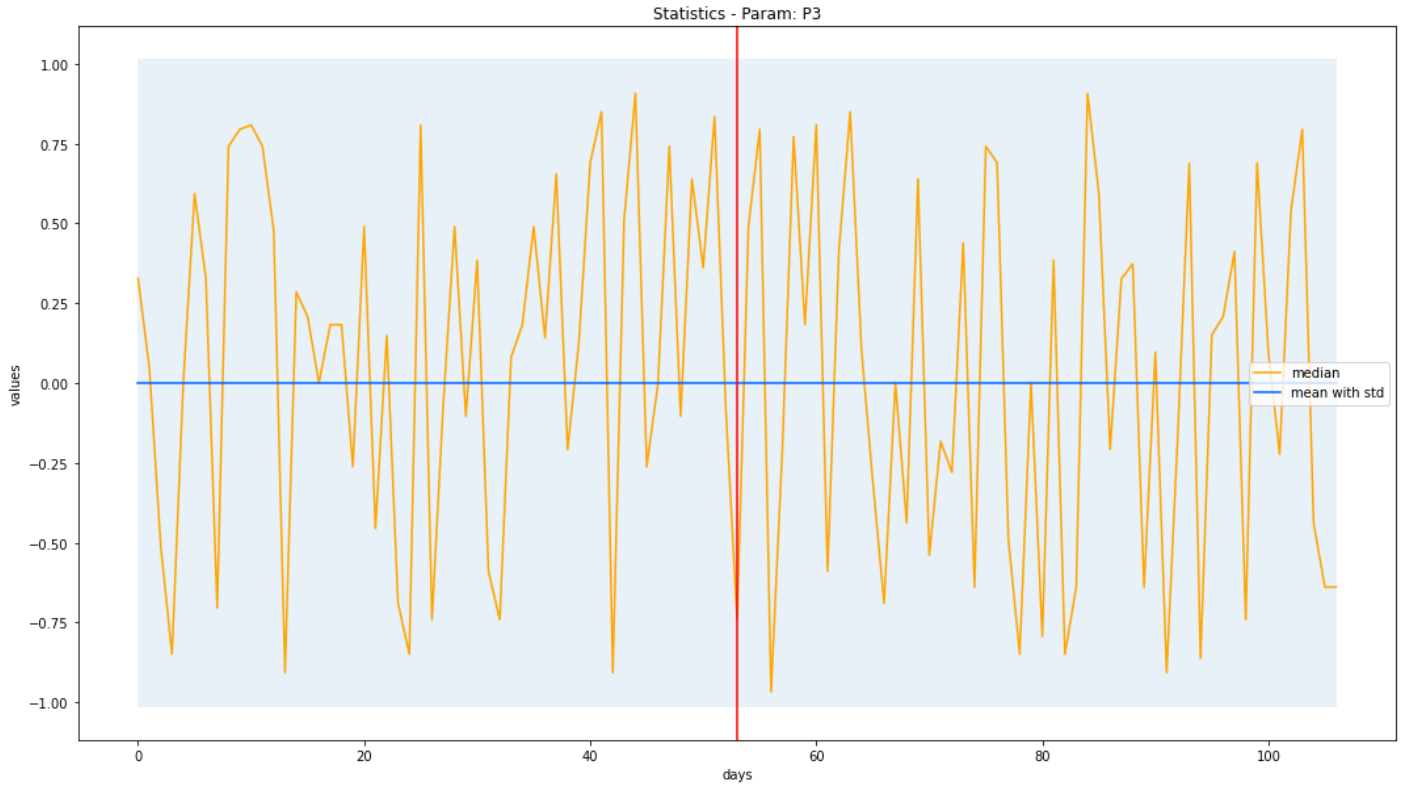
### 3.1.3. Dane ustandaryzowane - średnia, mediana, odchylenie



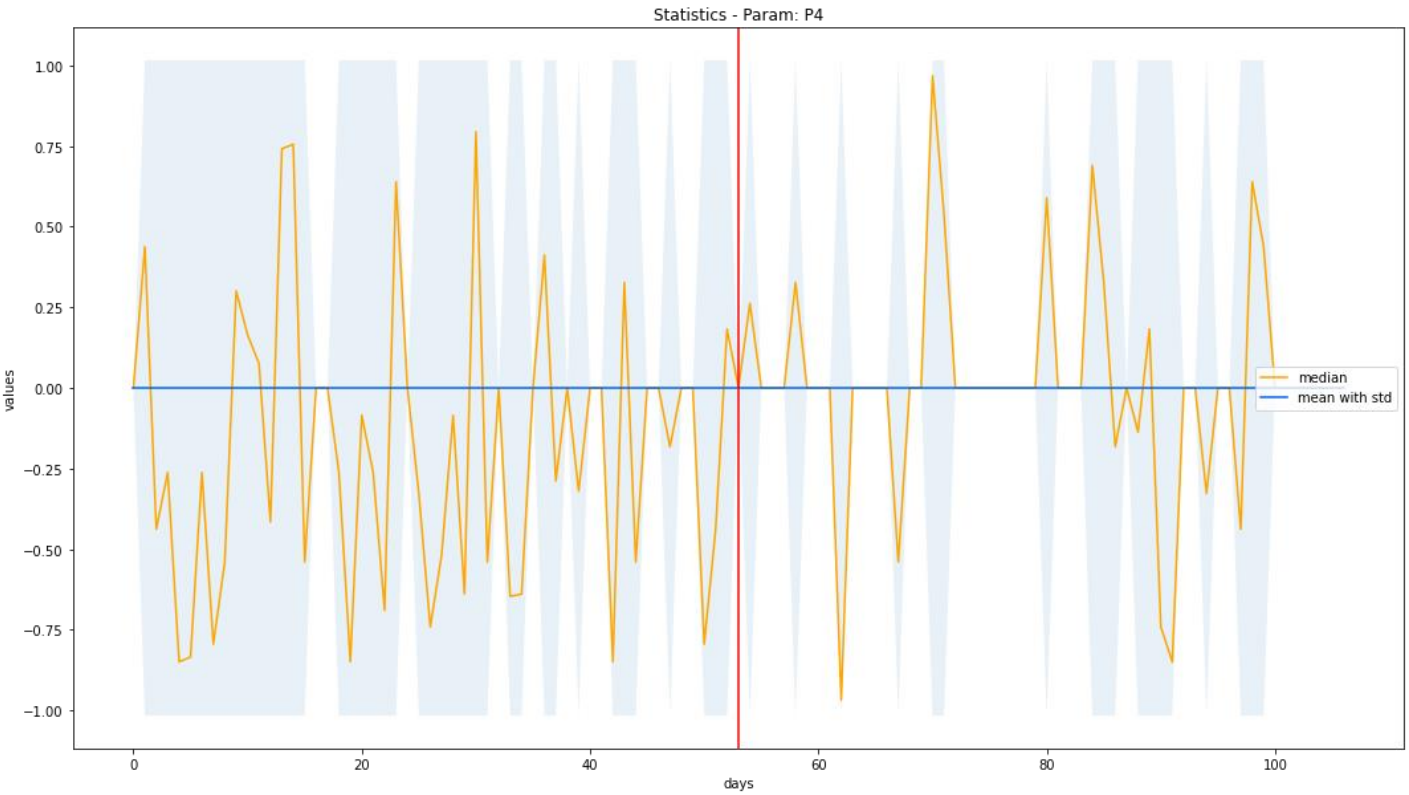

*Rysunek 3.15 Średnia, mediana, odchylenie dla parametru P1 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*



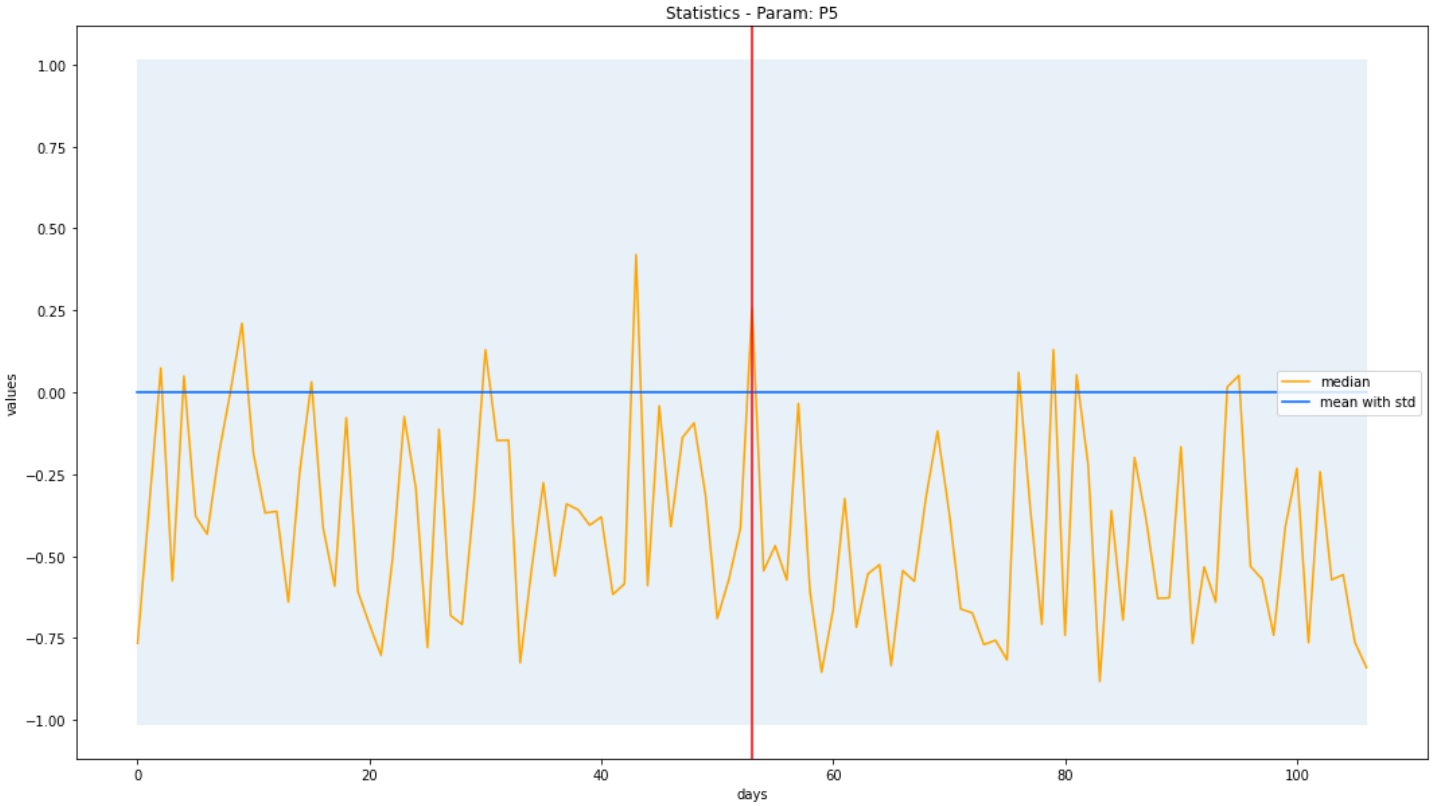

*Rysunek 3.16 Średnia, mediana, odchylenie dla parametru P2 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*



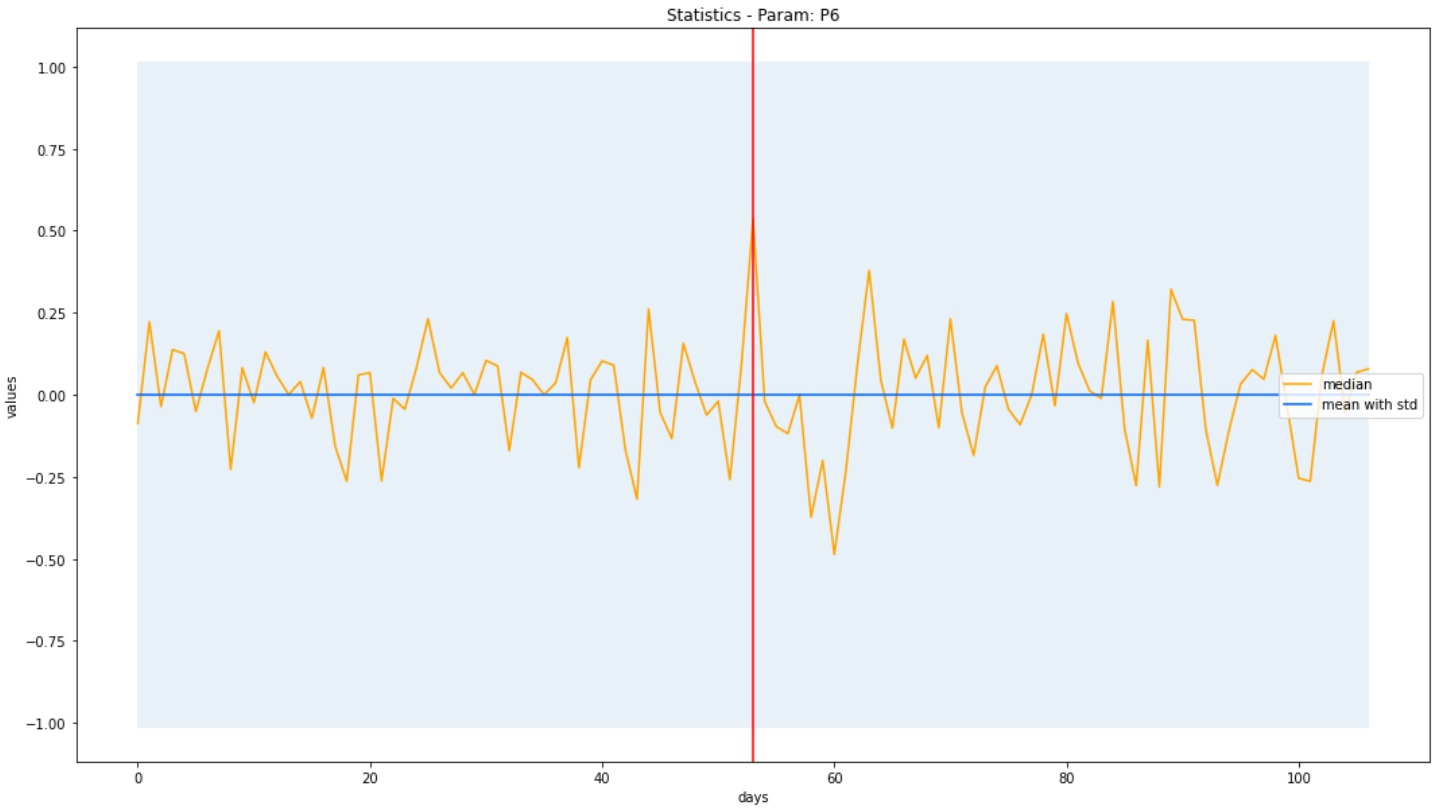

*Rysunek 3.17 Średnia, mediana, odchylenie dla parametru P3 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*



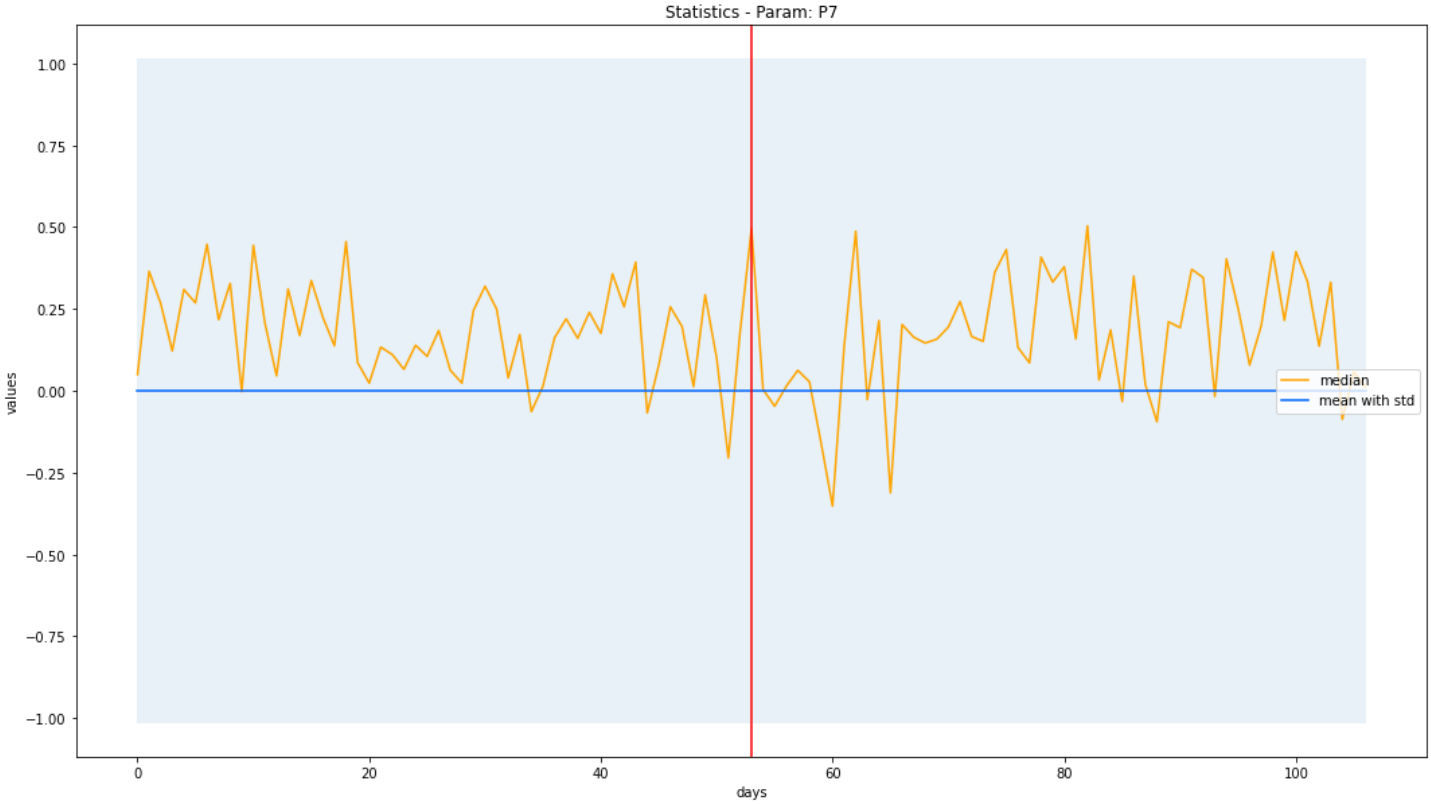

*Rysunek 3.18 Średnia, mediana, odchylenie dla parametru P4 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*



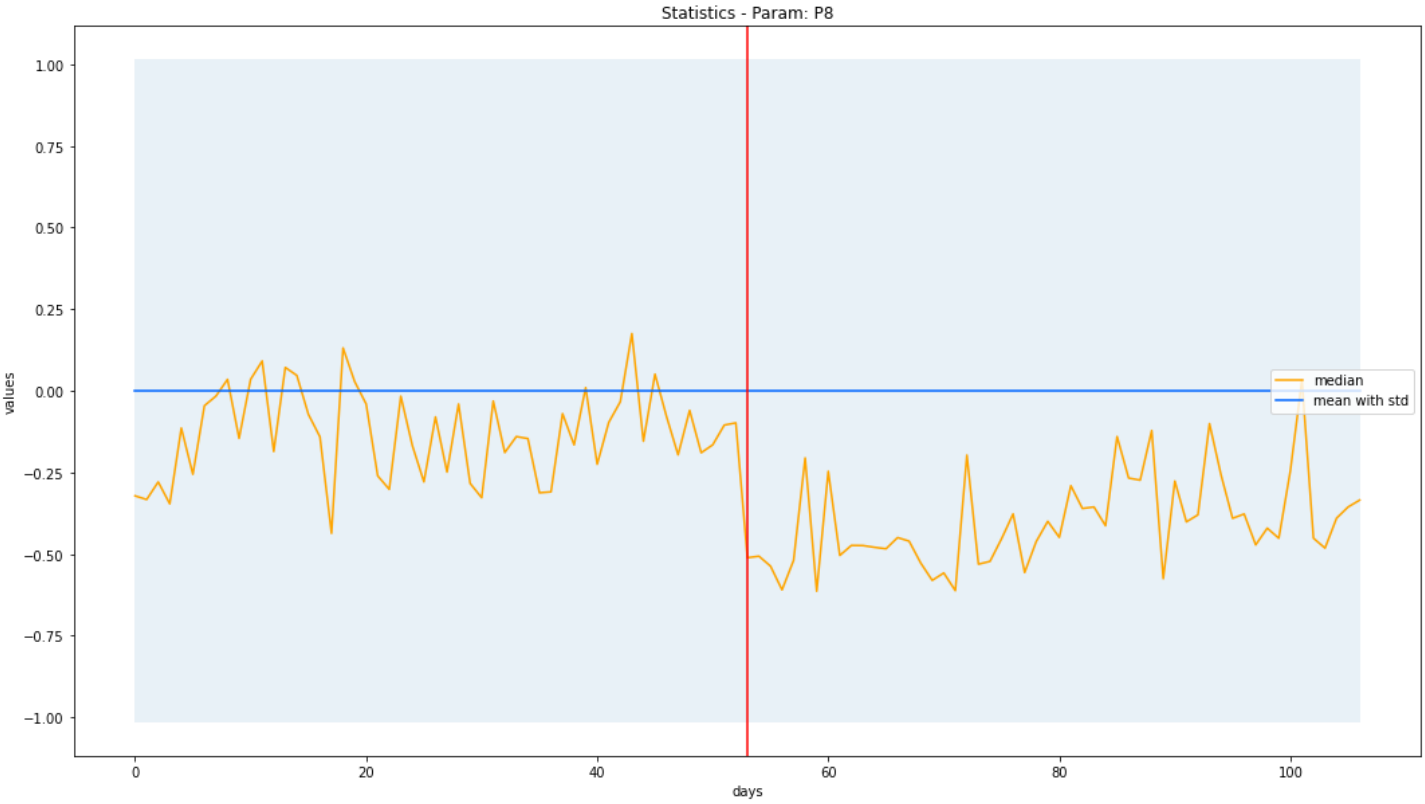

*Rysunek 3.19 Średnia, mediana, odchylenie dla parametru P5 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*



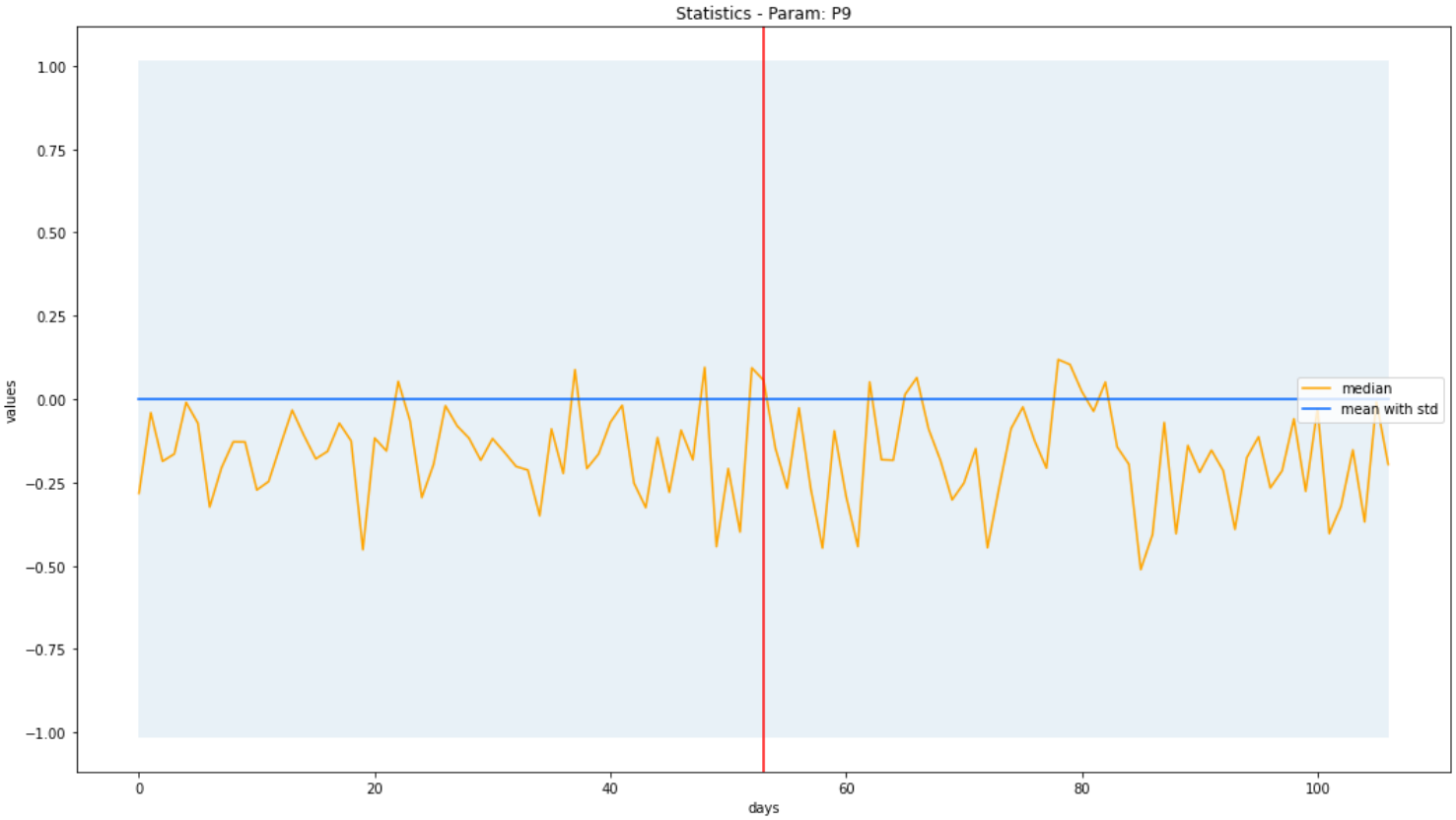

*Rysunek 3.20 Średnia, mediana, odchylenie dla parametru P6 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*




*Rysunek 3.21 Średnia, mediana, odchylenie dla parametru P7 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*




*Rysunek 3.22 Średnia, mediana, odchylenie dla parametru P8 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*

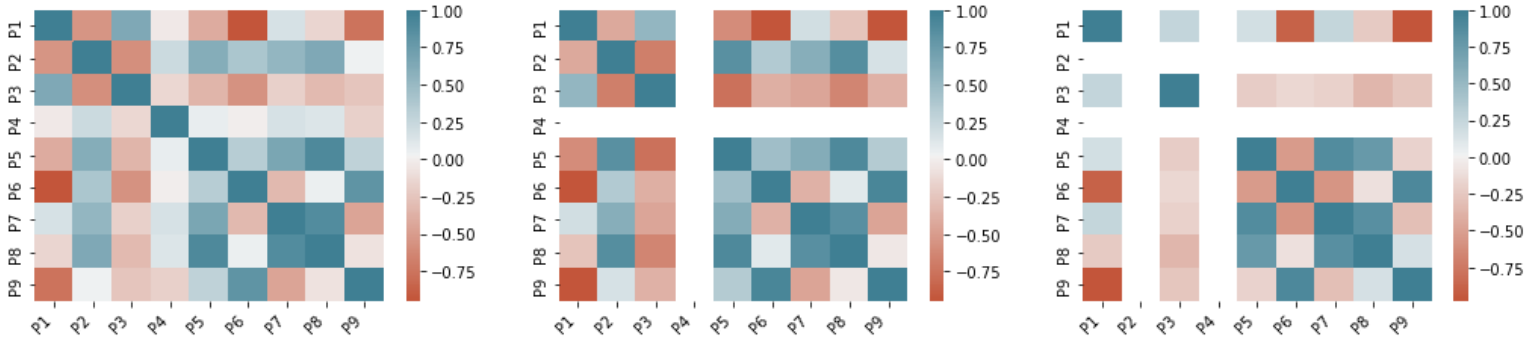



*Rysunek 3.23 Średnia, mediana, odchylenie dla parametru P9 - dane ustandaryzowane. Po lewej brak awarii, po prawej awaria*

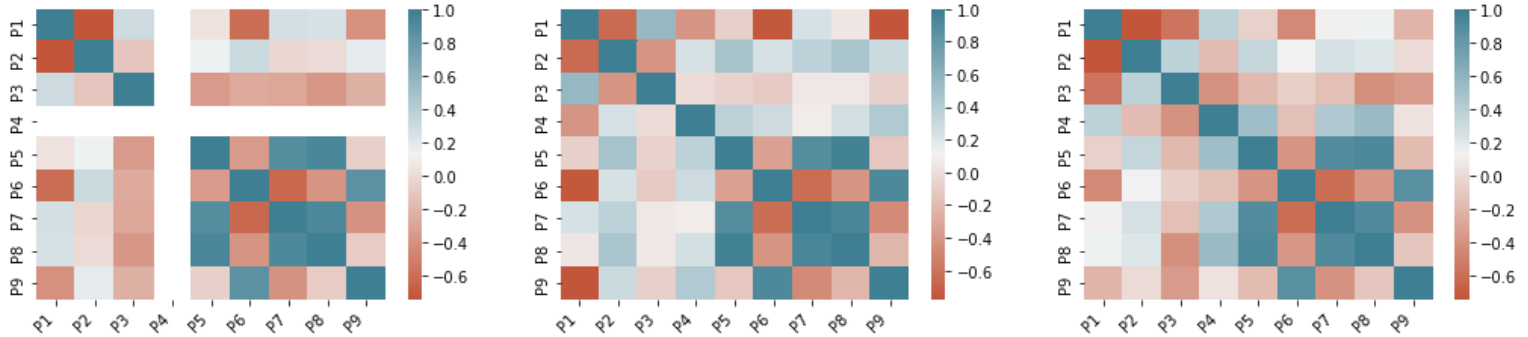
Zgodnie z założeniami standaryzacji, otrzymaliśmy średnią bliską 0, a odchylenie standardowe równe 1.

### 3.1.4 Dane ustandaryzowane - korelacja parametrów

Zgodnie z przewidywaniami, korelacja parametrów jest identyczna jak w przypadku danych nieustandaryzowanych. Standaryzacja nie wpływa na korelację danych.




*Rysunek 3.24 Korelacja parametrów ustandaryzowanych dla wybranych 3 dni - poprawne działanie*




*Rysunek 3.25 Korelacja parametrów ustandaryzowanych dla wybranych 3 dni - awaria*

## 3.2. Model sieci neuronowej

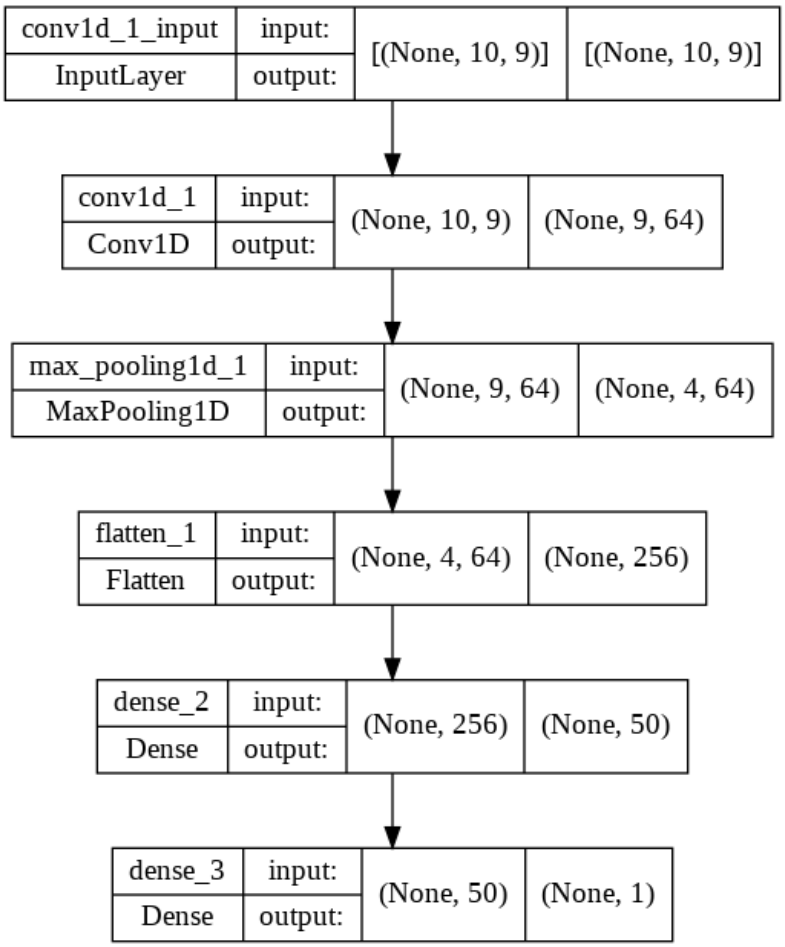
Tematyka naszego projektu związana była z implementacją rozwiązania przy użyciu sieci konwolucyjnych. Zdecydowaliśmy się stworzyć jeden model na którym testowaliśmy wpływ parametrów modelu na rezultat końcowy. Architektura omawianego modelu wygląda następująco:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d_2 (Conv1D)            (None, 30, 64)            1216

 max_pooling1d_2 (MaxPooling  (None, 15, 64)            0
 1D)

 flatten_2 (Flatten)          (None, 960)               0

 dense_4 (Dense)              (None, 50)                48050

 dense_5 (Dense)              (None, 1)                 51

=================================================================
Total params: 49,317
Trainable params: 49,317
Non-trainable params: 0
_________________________________________________________________
```

```python
def create_model(shape=(10,9), hidden_neuron = 50, pool_size = 2, filters = 64):
  model = Sequential()
  model.add(Conv1D(filters=filters, kernel_size=2, activation='relu', input_shape=shape))
  model.add(MaxPooling1D(pool_size=pool_size))
  model.add(Flatten())
  model.add(Dense(hidden_neuron, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(optimizer='adam', loss=keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
  return model
```

```python
model.fit(train_X, train_Y, batch_size=64, epochs=50,
                  validation_data=(vald_X, vald_Y),
                  callbacks=[
                            PlotLossesKeras(),
                            early_stop,
                            ModelCheckpoint(filepath=models_path + '/model0.hdf5',save_best_only=True)
                            ])
```

Poniżej ten sam model dla szeregu czasowego o długości 10 min:




Zapobiegając przetrenowaniu sieci skorzystaliśmy z **EarlyStoppingu**, który został zdefiniowany następującymi parametrami:

- monitor: loss,
- patience: 5,
- restore best weights,
- mode: minimum.

Do wizualizacji procesu uczenia użyliśmy rozszerzenie **LiveLossPlot**, a zapisywanie modelu odbywa się przy użyciu **ModelCheckpoint**, po każdej epoce i zachowuje jedynie najlepiej wytrenowaną sieć.

Pozostałe parametry modelu to:

- epochs = 50,
- batch_size = 64,
- optymalizator = Adam,
- funkcja straty = Binary Crossentropy,
- funkcja aktywacji warstwy wyjściowej = sigmoid.

### 3.2.1. Podział danych

Dane pobrane z plików CSV zostały wczytane do obiektów typu DataFrame z biblioteki Pandas. Znacznie ułatwiło to dalsze przypisanie danych do odpowiedniej klasy. Dodatkowo wykorzystując funkcję KFold z biblioteki sklearn przygotowaliśmy nasze dane do cross walidacji CV5.

Zgodnie z założeniami dane podzielono na 3 części:

- dane trenujące (ok. 80%),
- dane walidujące (ok. 10%),
- dane testujące (ok. 10%).

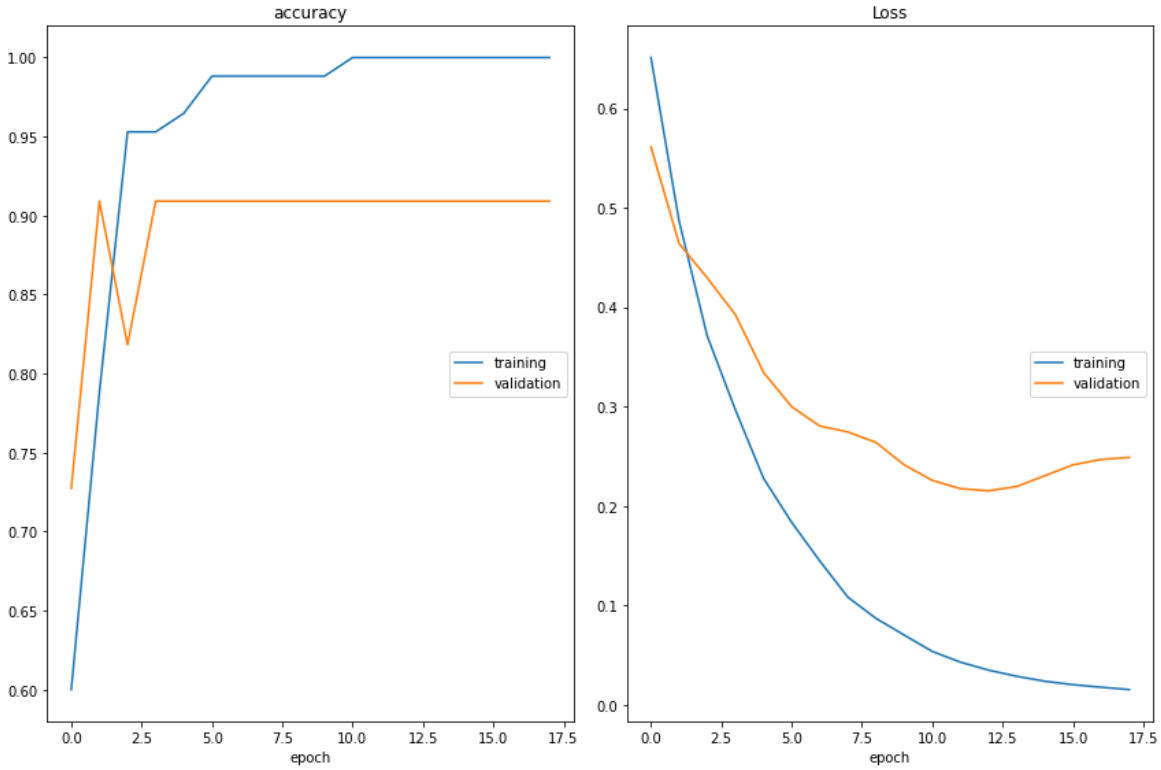
```
# cross validation CV5
kf = KFold(n_splits = 5, shuffle = True, random_state = 9)

train, test = next(kf.split(X_std, Y_std), None)

test_l = len(test)
train_X = np.asarray([X_std[x] for x in train])
train_Y = np.asarray([Y_std[y] for y in train])
vald_X =  np.asarray([X_std[x] for x in test[0:test_l:2]])
vald_Y =  np.asarray([Y_std[y] for y in test[0:test_l:2]])
test_X =  np.asarray([X_std[x] for x in test[1:test_l:2]])
test_Y =  np.asarray([Y_std[y] for y in test[1:test_l:2]])
```

### 3.2.2. Podział danych – szereg czasowy

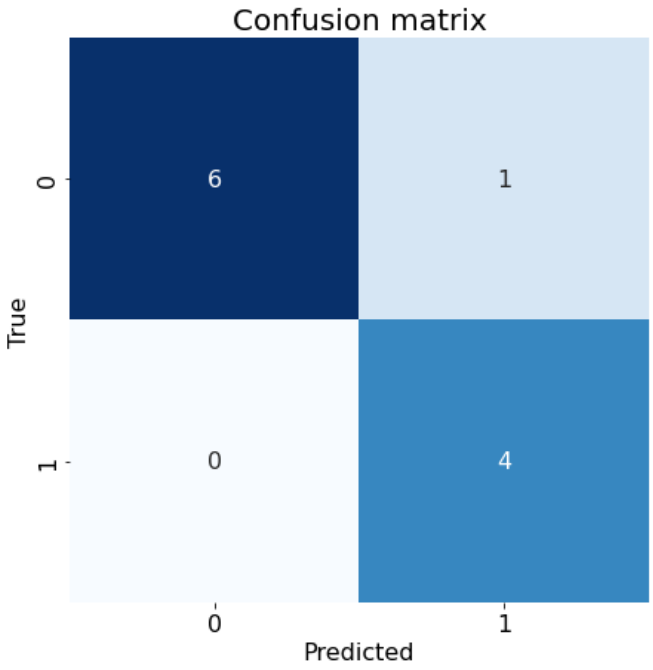
Oprócz trenowania modelu na danych z całego okresu czasu pracy urządzenia (31 min), zdecydowaliśmy się podzielić rekordy na szeregi czasowe różnej długości. Pozwoliło nam to przeanalizować ile minimalnie potrzebujemy danych na wejściu, aby sieć poprawnie określała czy urządzenie uległo awarii. Stworzono więc funkcję, która dzieli przekazane dane na sekwencję o podanej długości.

## 3.3. Analiza uzyskanych modeli – metryki

### 3.3.1. Model bez grupowania w szeregi czasowe

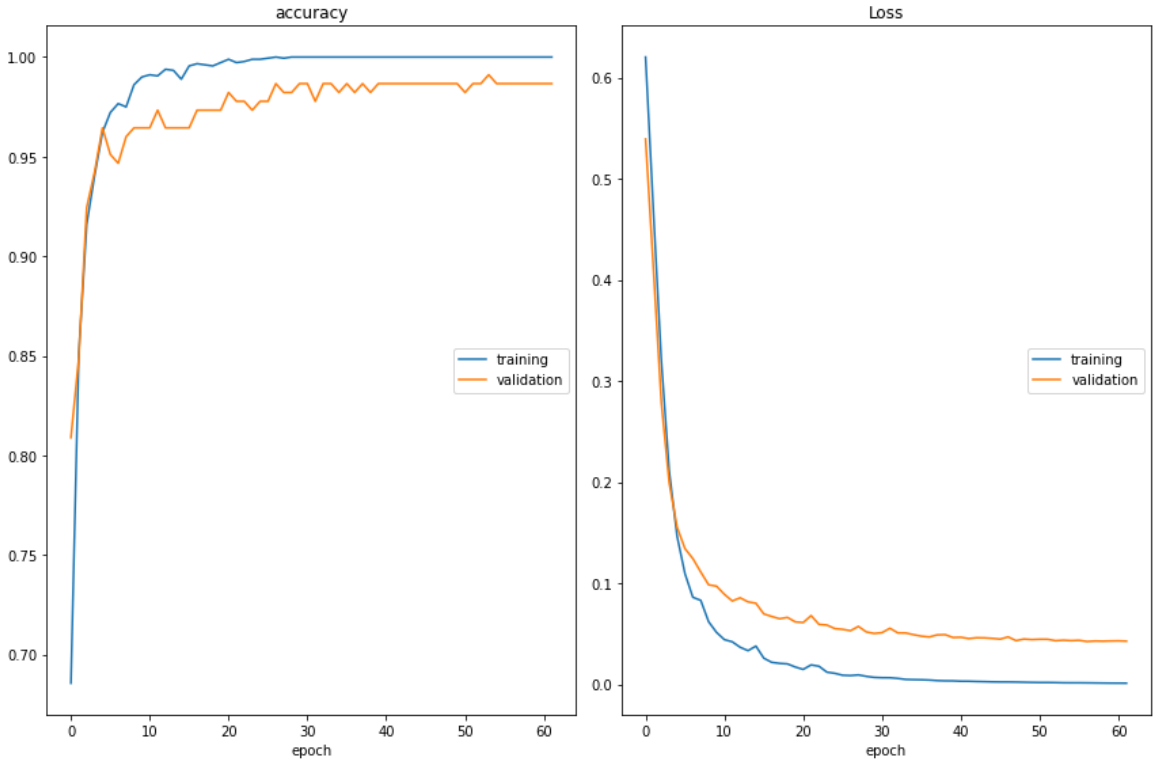



*Rysunek 3.26 Wykres dokładności modelu (po lewej) oraz wartość funkcji straty (po prawej). Brak grupowania*

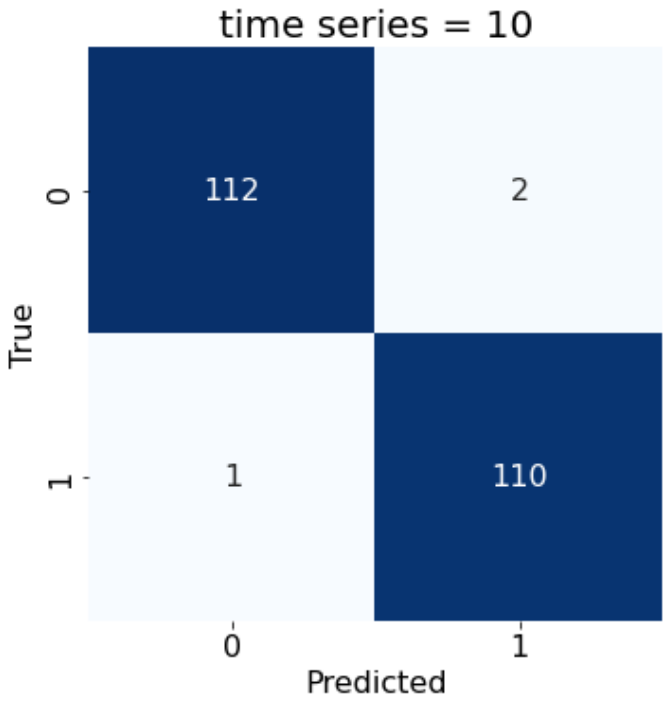



*Rysunek 3.27 Macierz pomyłek, model bez grupowania*

### 3.3.2. Model z szeregiem czasowym 10min

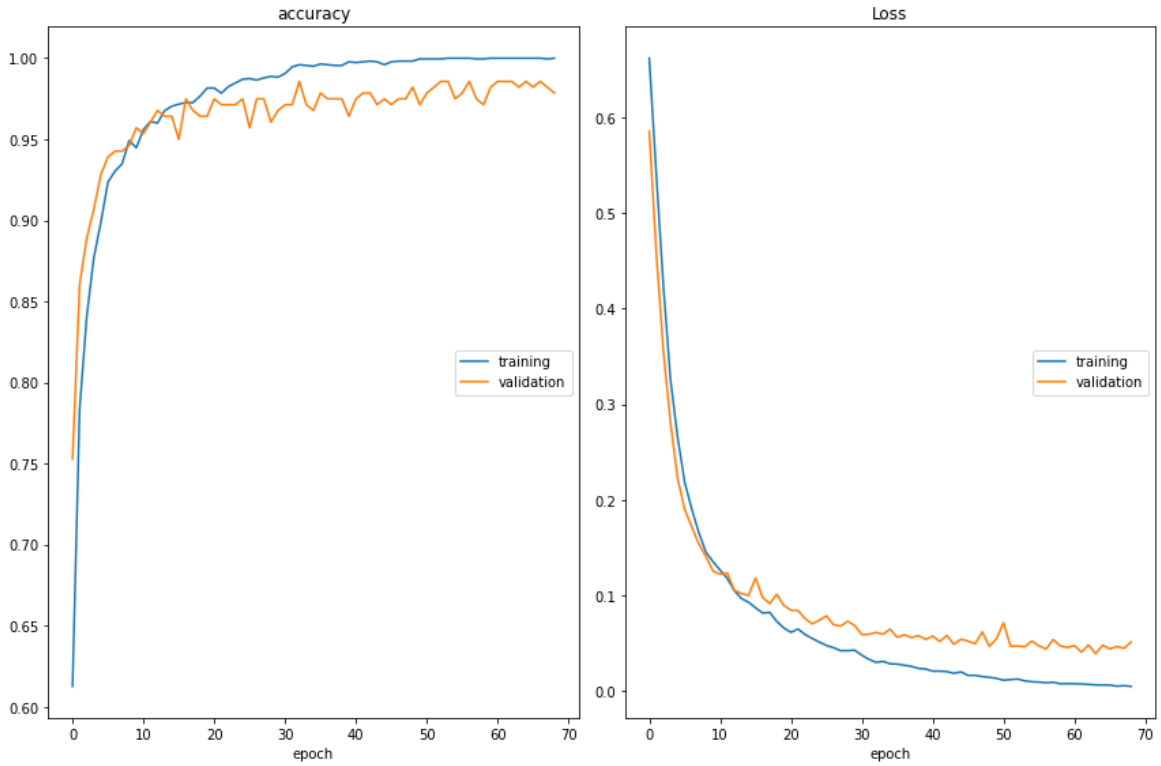



*Rysunek 3.28 Wykres dokładności modelu (po lewej) oraz wartość funkcji straty (po prawej). Szereg czasowy 10min*

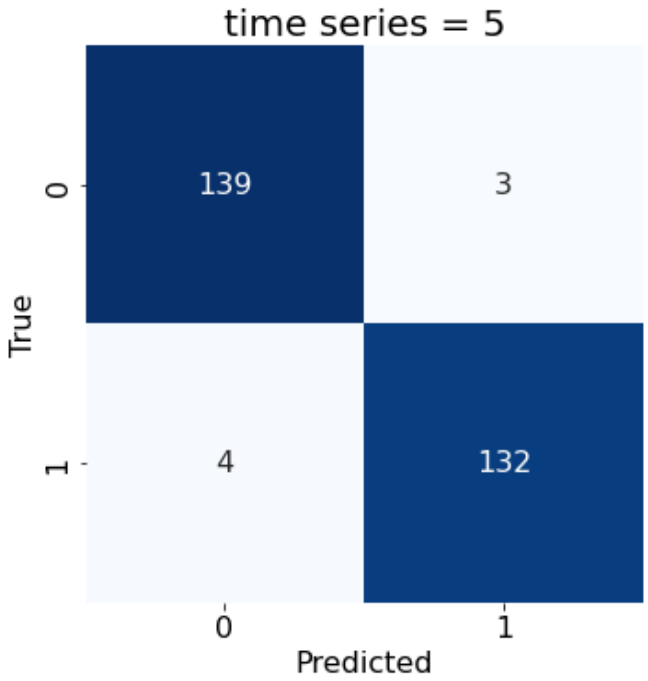



*Rysunek 3.29 Macierz pomyłek, szereg czasowy 10min*

### 3.3.3. Model z szeregiem czasowym 5min

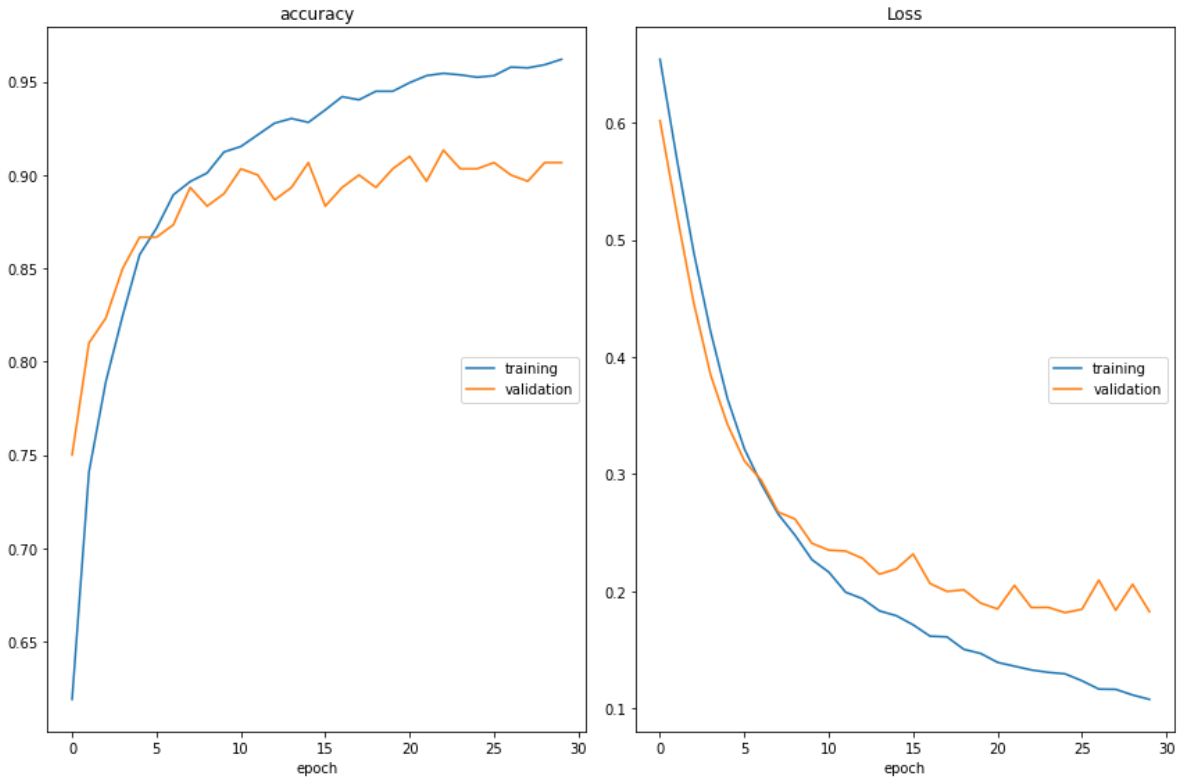



*Rysunek 3.30 Wykres dokładności modelu (po lewej) oraz wartość funkcji straty (po prawej). Szereg czasowy 5min*

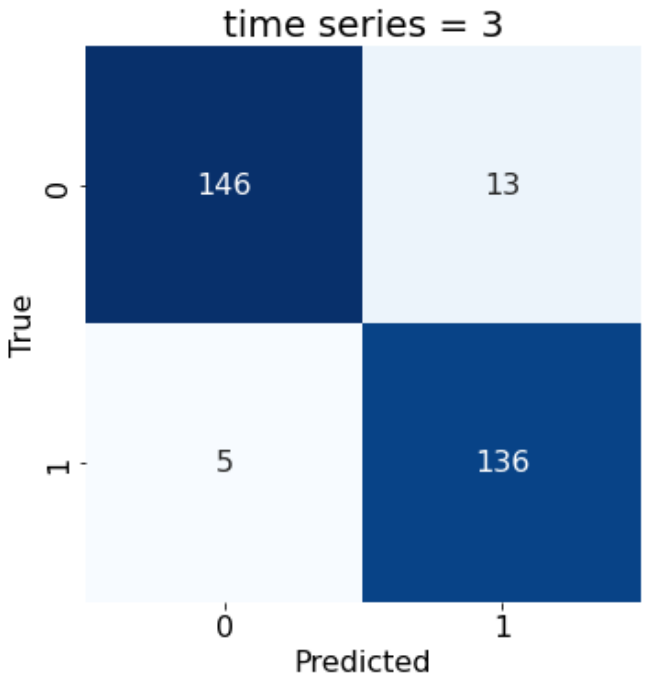



*Rysunek 3.31 Macierz pomyłek, szereg czasowy 5min*

### 3.3.4. Model z szeregiem czasowym 3min

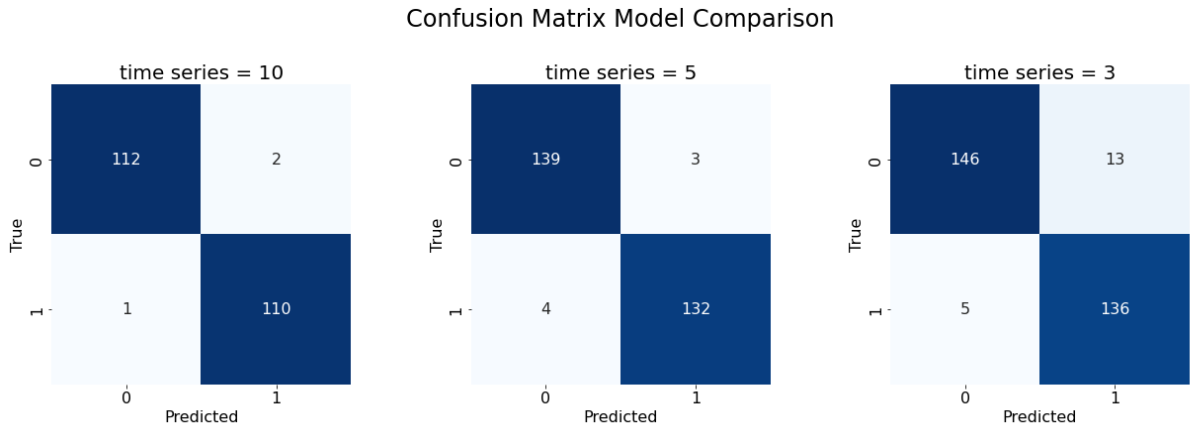



*Rysunek 3.32 Wykres dokładności modelu (po lewej) oraz wartość funkcji straty (po prawej). Szereg czasowy 3min*




*Rysunek 3.33 Macierz pomyłek, szereg czasowy 3min*

Poniżej zestawienie macierzy pomyłek dla wybranych szeregów czasowych:




*Rysunek 3.34 Porównanie macierzy pomyłek szeregów czasowych 10min, 5min, 3min*

### 3.3.5. Zestawienie uzyskanych wartości *accuracy* poszczególnych modeli

| Typ modelu | Accuracy w fazie walidacyjnej | Accuracy w fazie testowej |
|---|---|---|
| **Bez grupowania** | 0.909 | 0.909 |
| **Szereg czasowy 10 min** | **0.991** | **0.987** |
| **Szereg czasowy 5 min** | 0.986 | 0.975 |
| **Szereg czasowy 3 min** | 0.962 | 0.940 |

### 3.3.6. Porównanie metryk modeli

Aby zbadać, który model sprawuje się najlepiej skorzystano z szeregu metryk takich jak:

- accuracy,
- precision PPV,
- precision NPV,
- recall,
- F1 score,
- ROC AUC

| Typ modelu | Accuracy | Precision (PPV) | Precision (NPV) | Recall | F1 score | ROC AUC |
|---|---|---|---|---|---|---|
| **Bez grupowania** | 0.909 | 1.0 | 0.8 | 1.0 | 0.888 | 0.928 |
| **Szereg czasowy 10 min** | 0.987 | 0.983 | 1.0 | 0.981 | 0.990 | 0.990 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Szereg czasowy 5 min** | 0.975 | 0.972 | 0.978 | 0.970 | 0.974 | 0.974 |
| **Szereg czasowy 3 min** | 0.940 | 0.967 | 0.913 | 0.965 | 0.938 | 0.941 |

*Tabela 2 Wartości metryk uzyskanych modeli*

W tabeli powyżej przedstawiono porównanie statystyk uzyskanych predykcji modeli na danych testowych. Wraz ze zmniejszaniem długości szeregu czasowego, dokładność predykcji maleje. W przypadku szeregu czasowego 3min uzyskano jedne z gorszych wyników dla niemal wszystkich parametrów. Również dla modelu bez grupowania danych wyniki są zdecydowanie gorsze niż dla szeregu czasowego 10 min.

Najlepsze wyniki uzyskano dla szeregu czasowego o długości 10min.

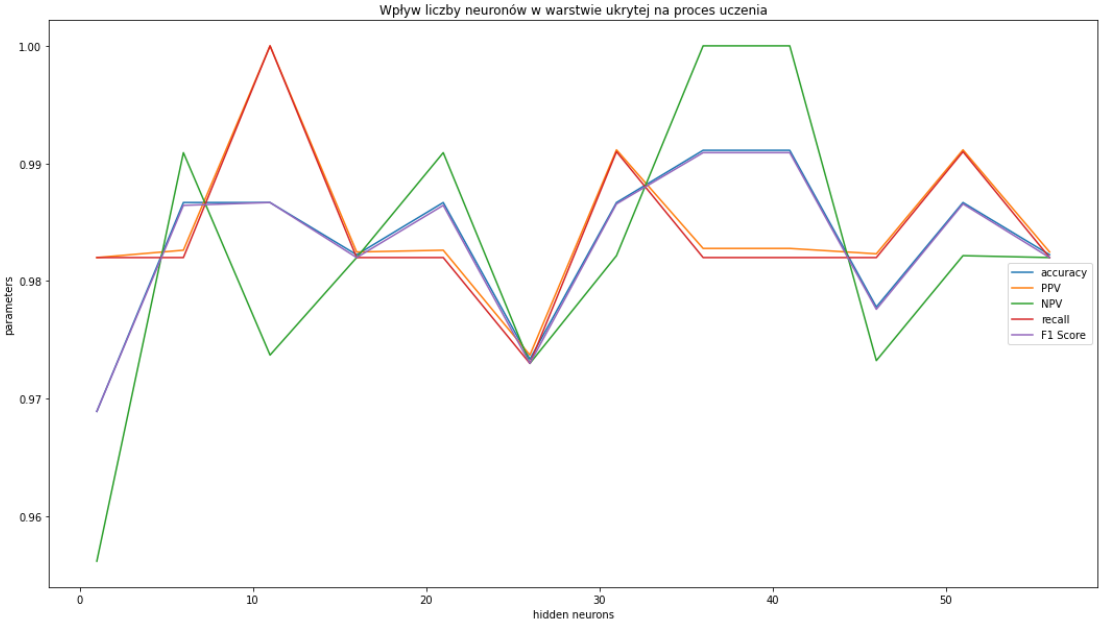
## 3.4. Wpływ parametrów sieci na proces uczenia

Biorąc pod uwagę model z szeregiem czasowym 10min zbadano wpływ parametrów sieci na proces uczenia.

- Zmiana ilości **neuronów** w warstwie ukrytej- pozostałe parametry sieci pozostały niezmienione

| Neurons | Accuracy | PPV | NPV | RECALL | F1 Score | ROC |
|---|---|---|---|---|---|---|
| 1 | 0.969 | 0.982 | 0.956 | 0.982 | 0.969 | 0.969 |
| 6 | 0.987 | 0.983 | 0.991 | 0.982 | 0.986 | 0.987 |
| 11 | 0.987 | 1.000 | 0.974 | 1.000 | 0.987 | 0.987 |
| 16 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 |
| 21 | 0.987 | 0.983 | 0.991 | 0.982 | 0.986 | 0.987 |
| 26 | 0.973 | 0.974 | 0.973 | 0.973 | 0.973 | 0.973 |
| 31 | 0.987 | 0.991 | 0.982 | 0.991 | 0.987 | 0.987 |
| 36 | 0.991 | 0.983 | 1.000 | 0.982 | 0.991 | 0.991 |
| 41 | 0.991 | 0.983 | 1.000 | 0.982 | 0.991 | 0.991 |
| 46 | 0.978 | 0.982 | 0.973 | 0.982 | 0.978 | 0.978 |
| 51 | 0.987 | 0.991 | 0.982 | 0.991 | 0.987 | 0.987 |
| 56 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 |

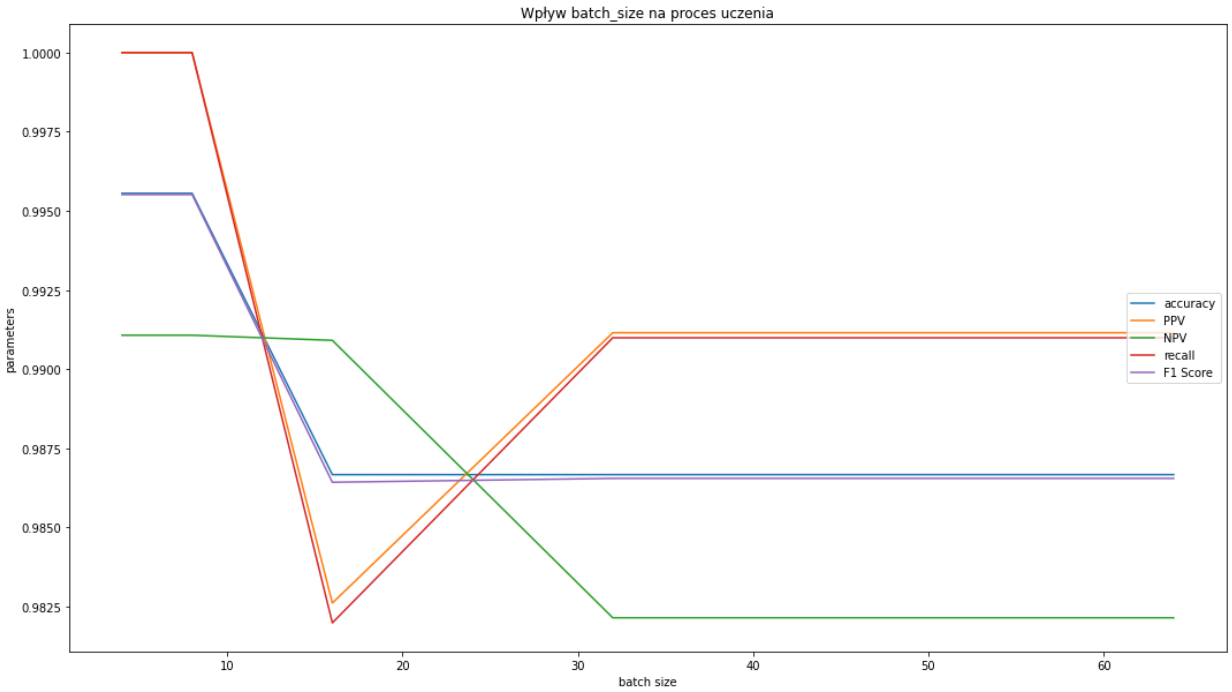
*Rysunek 3.35 Wpływ liczby neuronów w warstwie ukrytej na proces uczenia*


Wpływ liczby neuronów w warstwie ukrytej na proces uczenia
parameters
hidden neurons
accuracy
PPV
NPV
recall
F1 Score
1.00
0.99
0.98
0.97
0.96
0
10
20
30
40
50

- Zmiana wielkości parametru **batch_size** – pozostałe parametry sieci pozostały niezmienione

| Batch size | Accuracy | PPV | NPV | RECALL | F1 Score | ROC |
|---|---|---|---|---|---|---|
| 4 | 0.996 | 1.000 | 0.991 | 1.000 | 0.996 | 0.996 |
| 8 | 0.996 | 1.000 | 0.991 | 1.000 | 0.996 | 0.996 |
| 16 | 0.987 | 0.983 | 0.991 | 0.982 | 0.986 | 0.987 |
| 32 | 0.987 | 0.991 | 0.982 | 0.991 | 0.987 | 0.987 |
| 64 | 0.987 | 0.991 | 0.982 | 0.991 | 0.987 | 0.987 |

*Rysunek 3.36 Wpływ parametru batch_size na proces uczenia*

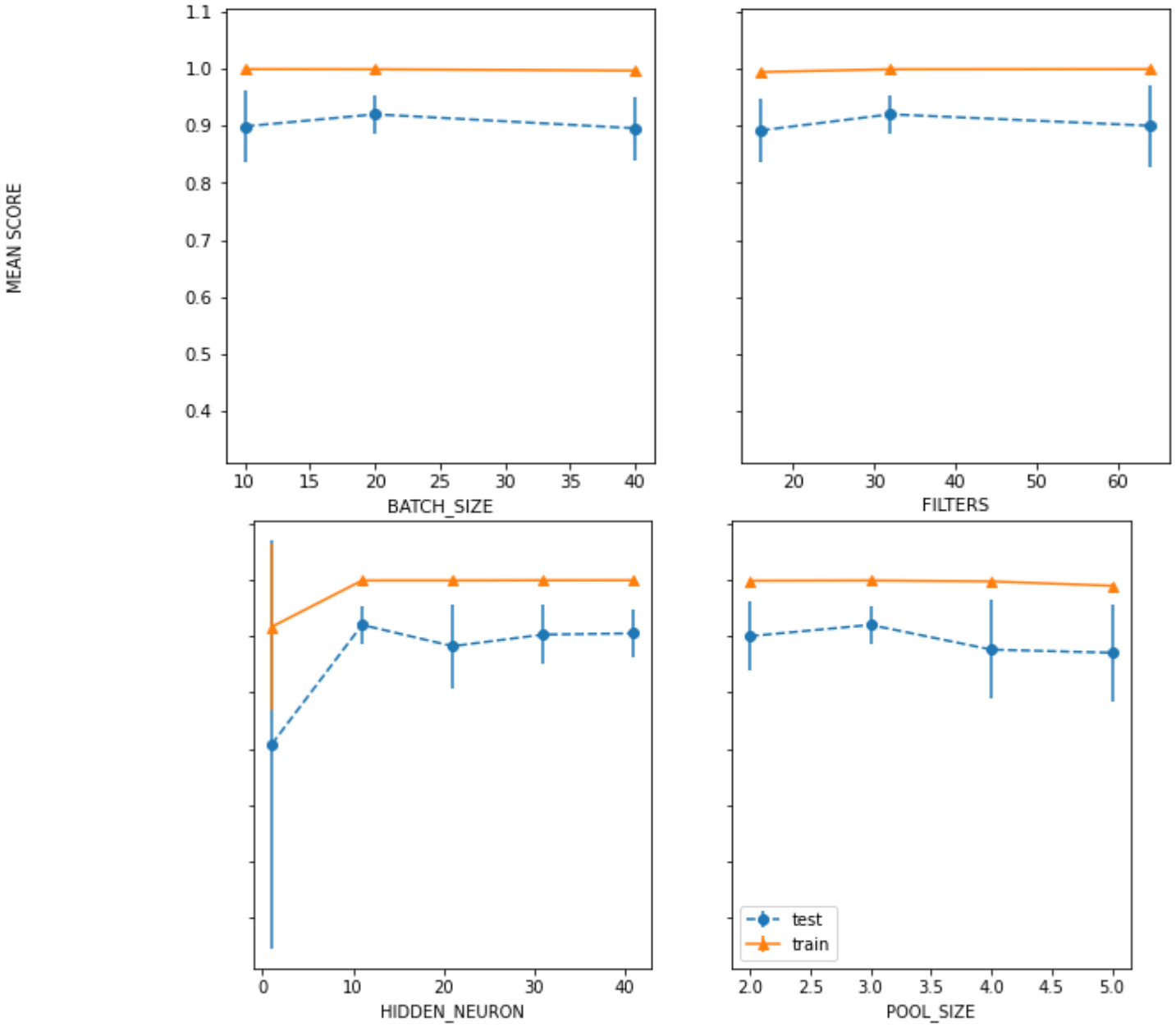



## 3.4.1. Poszukiwanie najlepszych parametrów

W celu zbadania najlepszych parametrów modelu wykorzystano funkcję **GridSearchCV** dostępną w pakiecie **sklearn**. Pozwala ona na zdefiniowanie parametrów, które chcemy przetestować. Funkcja automatycznie stworzy modele, dokona uczenia oraz zwróci parametry uzyskanych sieci.

Zdefiniowane parametry testowe:

- Szereg czasowy: **10min**
- Liczba neuronów w warstwie ukrytej: **[1, 11, 21, 31, 41]**,
- Wielkość warstwy pooling'u: **[2, 3, 4]**,
- Ilość filtrów: **[16, 32, 64]**,
- Wielkość batch_size: **[10, 20, 40]**,
- Ilość epok: **20**




*Rysunek 3.37 Wynik score dla badanych parametrów sieci*

Najlepszy wynik (score = 0.92) uzyskano dla parametrów:

- batch_size: 20,
- filters: 32,
- hidden_neuron: 11,
- pool_size: 3

## 4. Kod źródłowy

Kod źródłowy projektu został udostępniony w serwisie GitHub, dostępny jest pod poniższym linkiem: https://github.com/patryk0504/SSN-projekt

## 5. Instrukcja uruchomienia aplikacji

Aplikację można uruchomić wykorzystując darmowe środowisko cloudowe Google Colab manualnie przekazując do niego kod źródłowy z repozytorium lub otwierając poniższy link: https://drive.google.com/file/d/1jJRWKUX4tkfiTcsqlLIi92yHqDjR6R3t/view?usp=sharing

Ewentualnie można skorzystać z środowiska lokalnego:

- klonujemy repozytorium:
  - git clone https://github.com/patryk0504/SSN-projekt.git
- przechodzimy do głównego katalogu projektu
- instalujemy wymagania z pliku requirements.txt poleceniem:
  - pip install -r requirements.txt
- uruchamiamy środowisko Jupyter Notebook poleceniem:
  - jupyter notebook
- środowisko jupyter notebook powinno zostać automatycznie otwarte w nowym oknie przeglądarki

## 6. Podsumowanie

Podsumowując, założenia projektowe zostały spełnione. Wykonano statystyczną analizę danych na podstawie, której wykonano standaryzację danych. Zaprojektowano model sieci konwolucyjnej i przeprowadzono analizę wpływu jego parametrów na wyniki końcowe.
Dodatkowo wykonano podział danych wejściowych na szeregi czasowe:

- 10 minutowe,
- 5 minutowe,
- 3 minutowe.

Zbadano czy podział danych wejściowych daje lepsze rezultaty.

Mimo, że najlepszym wyborem dla problemów predykcji wykorzystujących szeregi czasowe jest RNN i LSTM to jak można wnioskować po uzyskanych przez nas wynikach, dobrym wyborem również może okazać się sieć CNN. Sieć ta przy doborze już kilku warstw daje bardzo wysokie wyniki predykcji.