



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ
KATEDRA ODDZIAŁYWANIA I DETEKCJI CZĄSTEK

Projekt dyplomowy

Opracowanie surowych danych medycznych zakończone ich zastosowaniem jako dane treningowe sieci neuronowej

Raw medical data processing culminated in their use as neural network training data

Autor:

Zuzanna Weronika Szczelina

Kierunek studiów:

Mikro- i Nanotechnologie w Biofizyce

Opiekun pracy:

prof. dr hab. inż. Tomasz Szumlak

Kraków, 2025

Spis Treści

1	WSTĘP	3
1.1	Pochodzenie Danych	4
1.2	Opis wybranych parametrów medycznych	5
1.3	Wymagania formalne przy pracy z danymi medycznymi	7
1.4	Konsultacja Medyczna.....	7
1.5	Rekurencyjne Sieci Neuronowe (RNN) i sieci LSTM	7
2	OBRÓBKA DANYCH.....	9
2.1	Wyodrębnienie informacji z kart	10
2.2	Utworzenie początkowego zestawu danych.....	10
2.3	Wizualizacja	11
2.4	Wybór parametrów.....	13
2.5	Wartości brakujące	15
2.6	Interpolacja Liniowa	17
2.7	Interpolacja z użyciem dekompozycji sezonowej	18
2.8	Korelacje	23
2.9	Augmentacja danych.....	24
3	PREDYKCJA ANOMALII.....	26
3.1	Prognozowanie bez dostrajania	27
3.2	Prognozowanie z dostrajaniem	30
4	PORÓWNANIE WYNIKÓW.....	32
4.1	Dane rzeczywiste	33
4.1.1	Dostrajanie vs bez dostrajania	33
4.1.2	Porównanie modeli.....	37
4.1.3	Porównanie długości sekwencji	39
4.2	Optymalizacja.....	39
4.3	Dane generowane	39
5	ZAKOŃCZENIE.....	41
5.1	PODSUMOWANIE	41
5.2	GŁÓWNE NARZĘDZIA.....	41
5.3	Konsultacja Medyczna.....	42
6	Bibliografia	42

1 WSTĘP

1.1 Pochodzenie Danych

Dane z którymi pracowałem w projekcie pochodząły z Odcinka Kardiochirurgii i Intensywnej Opieki Kardiochirurgicznej Oddziału Anestezjologii i Intensywnej Terapii Uniwersyteckiego Szpitala Dziecięcego w Krakowie.

Część Intensywnej Terapii Kardiochirurgicznej ww. oddziału to miejsce gdzie leczone są dzieci z wrodzonymi i nabitymi wadami serca oraz chorobami kardiologicznymi zagrażającymi życiu. Są to pacjenci w różnym wieku, począwszy od noworodków (także wcześniaków) aż do 18 roku życia. Najczęściej na oddział trafiają dzieci po operacjach serca, a ich obserwacja i odpowiednio szybkie reagowanie w pierwszych dobach po zabiegu są kluczowe dla zachowania ich życia.



Rysunek 1: Zdjęcie stanowisk obserwacyjnych na oddziale z którego pochodzą dane. Zdjęcie własne.

Podstawowym narzędziem służącym do rejestracji obserwacji pacjentów na ww. oddziale są tzw. *Karty obserwacji i pielęgnacji*. Są to papierowe karty formatu A3, na których zapisuje się m.in. wartości parametrów życiowych, rozłożone w czasie ilości podawanych leków czy płynów przyjęte oraz wydalone przez pacjenta. Wśród obserwowanych parametrów życiowych znajdują się m.in. puls, temperatura ciała, ciśnienie tętnicze, ośrodkowe ciśnienie żyłne czy saturacja krwi. Informacja o płynach praz lekach konieczna jest do obliczania tzw. *bilansu płynów*. Jest to różnica między objętością płynów podanych oraz wydalonych przez pacjenta, która jest stale monitorowana. [K.M]

Zdjęcie przykładowej karty przedstawiam na Rysunku 2.

Rysunek 2: Przykładowa wypełniona karta obserwacji pacjenta z zaznaczonymi sekcjami: zielony - informacje o pacjencie, pomarańczowy - parametry monitorowane, niebieski - płyty przyjęte (w tym leki), żółty - płyty wydalone. Jedna kolumna tabeli odpowiada jednej porannej godzinie. Jedna karta to obserwacja z jednego dnia (24h).

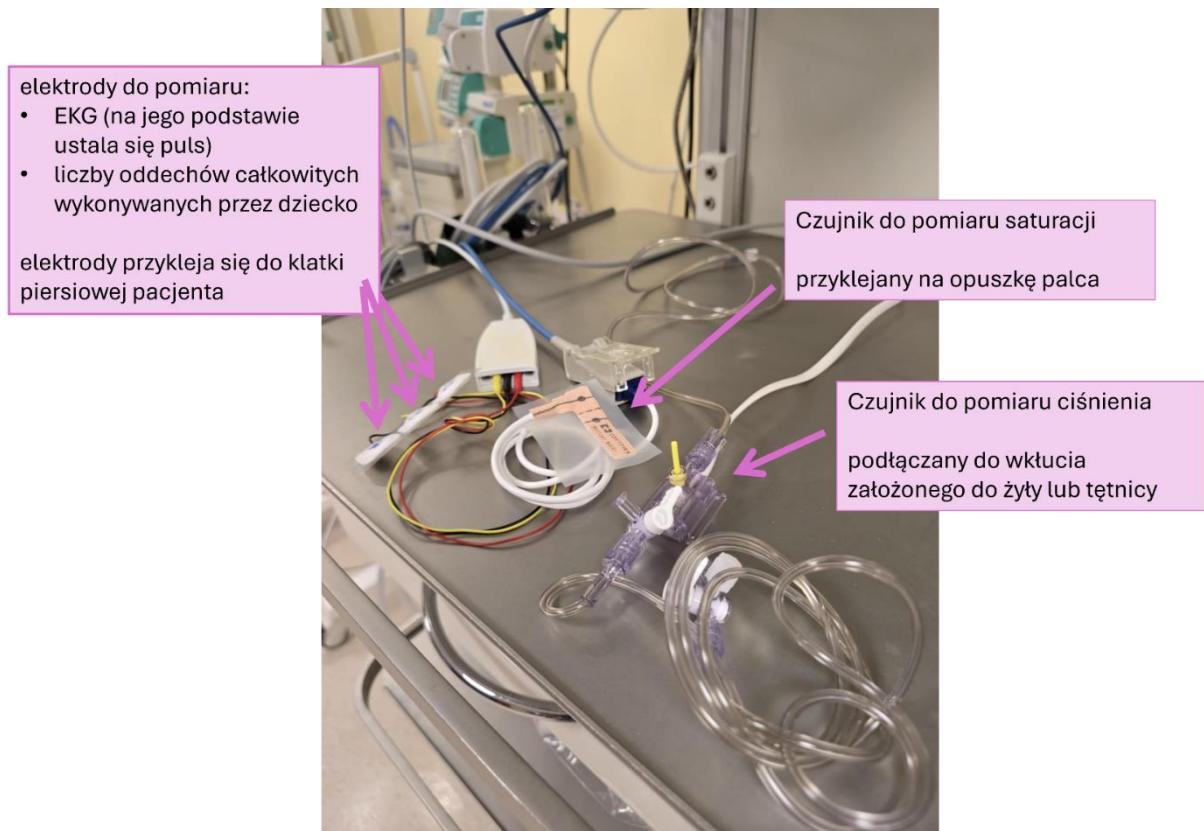
1.2 Opis wybranych parametrów medycznych

Puls – tzn. liczba uderzeń serca na minutę. Zapisywany co godzinę (w nagłych sytuacjach częściej). Zapisywana jest wartość liczbową z dokładnością do 5 uderzeń serca na minutę. Jego zbyt wysoka wartość może oznaczać np. ból, odwodnienie, gorączkę lub krwawienie u pacjenta, a zbyt niska np. hipotermię lub nagłe zatrzymanie krażenia.

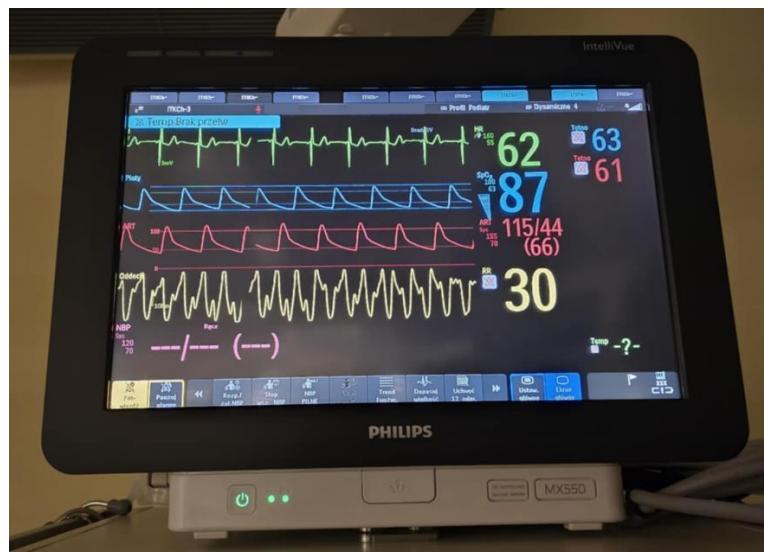
Saturacja – oznacza procent wysycenia krwi tlenem. Dopuszczalna minimalna wartość saturacji jest różna w zależności od wady serca. Zbyt niska wartość wymaga zwiększenia ilości podawanego tlenu lub modyfikacji procesu wspomagania oddechu.

Oddechy – zapisywana jest całkowita liczba oddechów wykonywanych przez pacjenta oraz liczba oddechów wymuszanych czyli wykonywanych dzięki obecności urządzenia wspomagającego oddychanie.

Na Rysunkach 3 i 4 przedstawiam przykładowe czujniki wykorzystywane na oddziale służące m.in. do pomiaru ww. parametrów oraz zdjęcie kardiomonitora. Z kardiomonitora pielęgniarki odczytują część parametrów, które zapisują na karcie obserwacyjnej. [K.M]



Rysunek 3: Czujniki kardiomonitora. Zdjęcie własne.



Rysunek 4: Kardiomonitor, z którego odczytywane są niektóre parametry zapisywane na karcie obserwacyjnej. Zdjęcie własne.

1.3 Wymagania formalne przy pracy z danymi medycznymi

Dokumentacja medyczna, z którą pracowałam w trakcie projektu nie jest powszechnie dostępna. Przed jej pozyskaniem wymagana była wcześniejsza anonimizacja kard, a prace wykonane przy procesie pozyskiwania dokumentacji medycznej choć nie są widoczne w późniejszych analizach stanowią istotną część tego projektu.

1.4 Konsultacja Medyczna

W czasie całego projektu współpracowałam z pielęgniarką doświadczoną w pracy na oddziale OIOM, z którego pochodziły dane. Pielęgniarka zgodziła się na czas trwania projektu zostać Konsultantką i jej udział był bardzo znaczący dla przebiegu wszystkich działań [K.M].

1.5 Rekurencyjne Sieci Neuronowe (RNN) i sieci LSTM

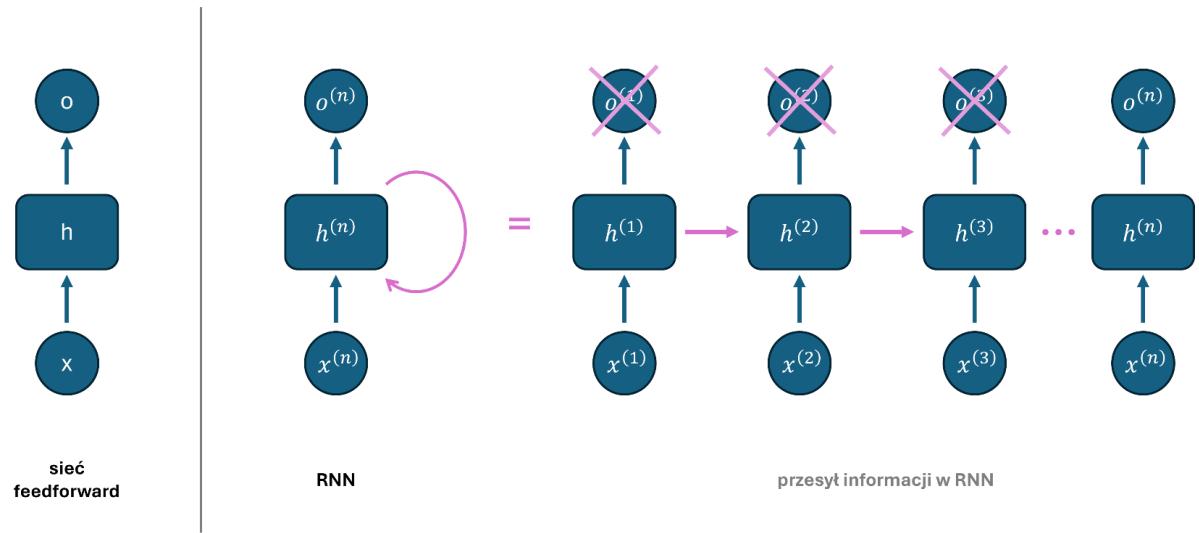
Dane sekwencyjne to takie dane, w których poszczególne elementy mają określoną kolejność. Przykładem są szeregi czasowe (np. informacja o tym jak zmieniał się puls pacjenta medycznego w czasie, przedstawiona w formie wartości przypisanych konkretnym godzinom).

Rekurencyjne Sieci Neuronowe to podstawowy rodzaj sieci nadający się do modelowania danych sekwencyjnych.

Najprostsza architektura RNN wygląda jak zwykła sieć typu *feedforward* z jedną warstwą ukrytą, z tą różnicą że istnieje w niej jedno dodatkowe połączenie: pomiędzy wyjściem warstwy ukrytej, a jej wejściem.

Właśnie to połączenie odpowiada za zdolność sieci do uwzględniania sekwencyjności danych. Mechanizm działa następująco: kiedy w sieci wykonywane są operacje kroku czasowego $n-1$ informacja o otrzymanym stanie warstwy ukrytej $h^{(n-1)}$ jest zapamiętywana. W kolejnym kroku czasowym (tj. n) do ustalenia stanu $h^{(n)}$ wykorzystywane są wartości z aktualnego stanu warstwy wejściowej $x^{(n)}$ oraz (dzięki ww. połączeniu) z zapamiętanego stanu $h^{(n-1)}$. Oznacza to, że w kroku n uwzględniana jest informacja o kroku $n-1$. Graficznie przedstawiłam to na Rysunku 5.

Wyżej opisana zasada działania sprawia, że gdy na wejście sieci podawany jest drugi element sekwencji w obliczeniach uwzględniana jest dodatkowo informacja o pierwszym, gdy podawany jest trzeci uwzględniana jest dodatkowo informacja o pierwszym i drugim itd. Innymi słowy, przy podawaniu na wejście sieci informacji o każdym elemencie sekwencji uwzględniana jest dodatkowo informacja o wszystkich jej poprzednich elementach. Ostateczną wartością wyjściową jest wartość uzyskiwana na wyjściu sieci w kroku, w którym na jej wejście podawany jest ostatni element sekwencji.



Rysunek 5: Porównanie sieci typu fastforward i Rekurencyjnej Sieci Neuronowej. x - wejście, o - wyjście, h - warstwa ukryta. (Grafika własna wykonana na podstawie ilustracji z książki [9].)

Z powodu ponownego podawania na wejście sieci informacji o wcześniejszych elementach, dla długich sekwencji wartość liczbową elementów jest wielokrotnie przetwarzana używając tej samej macierzy współczynników. W zależności od współczynników może to powodować nadmierne zwiększenie, lub przeciwnie, wygaszanie wartości gradientów obliczanych w trakcie treningu sieci z wykorzystaniem propagacji wstecznej. Jest to tzw. *problem znikającego lub eksplodującego gradientu*. Jednym ze sposobów na rozwiązywanie tego problemu jest zastąpienie warstwy ukrytej podstawowej RNN czymś co nazywamy komórką *Long Short-Term Memory* (LSTM), która ma bardziej skomplikowaną architekturę niż najprostsza warstwa ukryta bazująca tylko na macierzy współczynników i wyrazów wolnych. Sieć rekurencyjną której jednostkami budulcowymi są komórki LSTM nazywamy Siecią Rekurencyjną typu LSTM lub po prostu LSTM. [9] [10]

2 OBRÓBKA DANYCH

2.1 Wyodrębnienie informacji z kart

Na tym etapie dysponowałem surowymi papierowymi kartami zawierającymi wiele informacji. Konieczne było ustalenie znaczenia i sposobu zapisu poszczególnych parametrów. Znaczna część zapisywanych danych nie była przedstawiana jako liczby lecz symbole zaznaczane na wspólnej skali. Nieregularność oraz struktura zapisu uniemożliwiała użycie systemu Optycznego Rozpoznawania Znaków (OCR). Do zrozumienia części zapisów konieczne było doświadczenie w pracy na oddziale i umiejętność rozpoznania stanu dziecka, który występował w odpowiadającej chwili. Odczytanie i przedstawienie wybranych Informacji z kart w formie numerycznego zestawu danych było możliwe dzięki konsultacjom z współpracującą Pielęgniarką. Do zapisu danych korzystałem z programu Excel.

2.2 Utworzenie początkowego zestawu danych

Do dalszej pracy używałam języka programowania Python. Pierwszą rzeczą którą zrobiłam po przeniesieniu danych do osobnych arkuszy programu Excel było utworzenie zestawu danych składającego się ze wszystkich pozyskanych parametrów. Żeby to zrobić wczytałam wszystkie arkusze programu Excel jako *pandas dataframe* (podstawowa struktura danych biblioteki *pandas*) i połączylam w jeden *dataframe*. Ostateczny zestaw danych został zapisany jako plik .csv w celu dalszego użytku. Analogicznych struktur danych używałam w czasie dalszej pracy.

```
na_values = ['NA']
respirator = pd.read_excel('../data/raw/karty.xlsx', sheet_name='respirator', na_values=na_values).drop('info', axis='columns')
param_zyciowe = pd.read_excel('../data/raw/karty.xlsx', sheet_name='param_zyciowe', na_values=na_values)
saturacja = pd.read_excel('../data/raw/karty.xlsx', sheet_name='saturacja', na_values=na_values)
plyny_przyjete = pd.read_excel('../data/raw/karty.xlsx', sheet_name='plyny_przyjete', na_values=na_values)

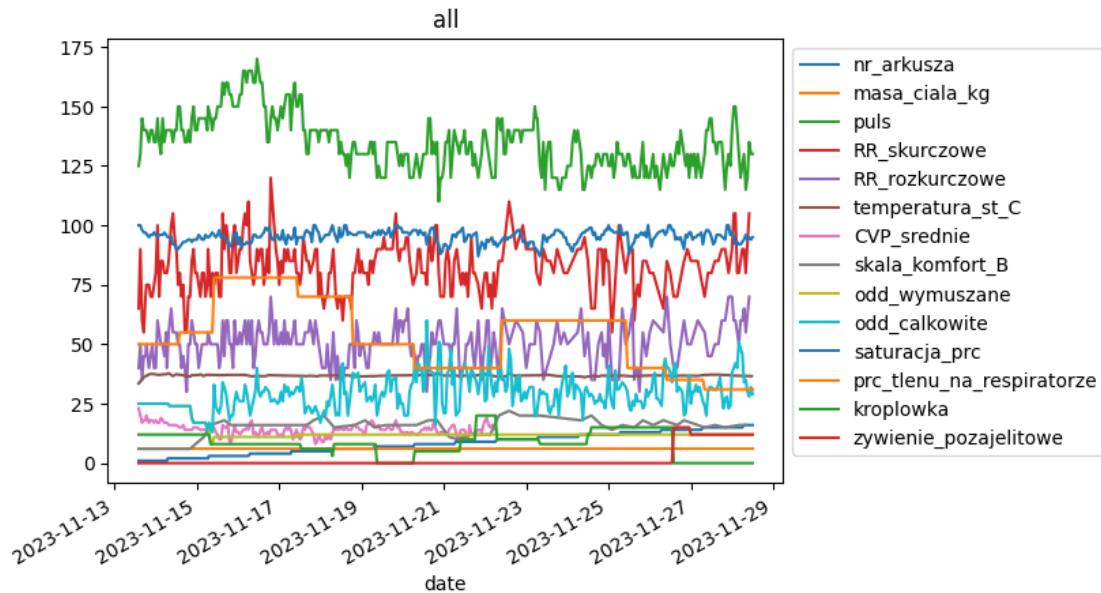
dfs_to_merge = [param_zyciowe, respirator, saturacja, plyny_przyjete]
df = reduce(lambda left, right: pd.merge(left, right, how='outer', on=['nr_arkusza', 'date']), dfs_to_merge)

df.to_csv('../data/df_all.csv', index=False)
```

2.3 Wizualizacja

W następnej kolejności konieczne było przeprowadzenie procesu eksploracji danych. Zaczęłam od wizualizacji wszystkich uzyskanych szeregów czasowych. Wszystkie szeregi zawierały brakujące wartości, ponadto parametry nie były zapisywane ze stałym krokiem czasowym. Momenty kiedy zapisywano dane nie zawsze były jednakowe dla wszystkich parametrów.

Na Rysunku 6 przedstawiam wszystkie szeregi na wspólnym wykresie. Przebiegi zostały interpolowane przez połączenie znanych wartości liniami prostymi.

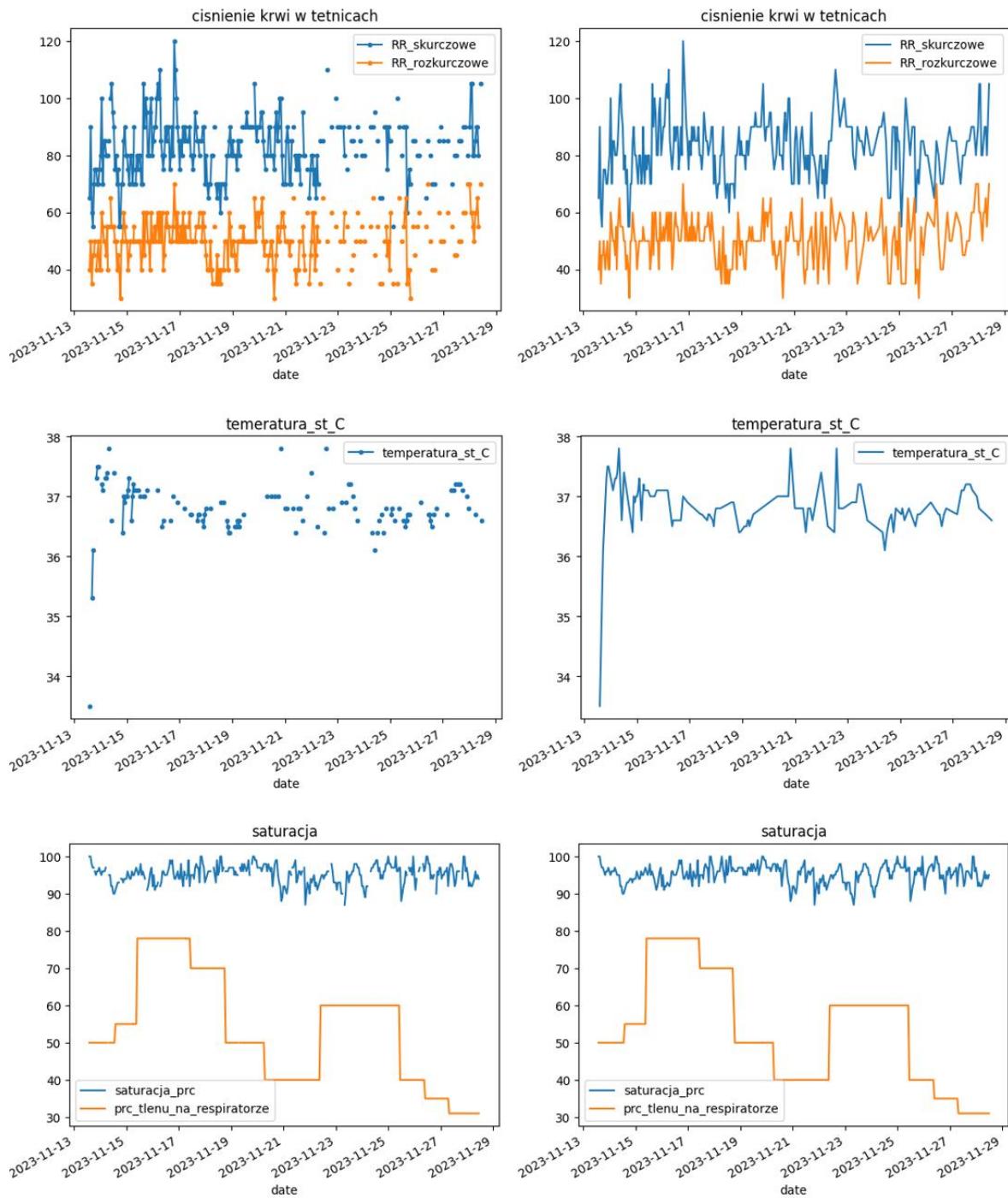


Rysunek 6: Wszystkie uzyskane szeregi czasowe umieszczone na jednym wykresie w celu poglądowej wizualizacji.
Skala na osi pionowej jest wspólna dla wszystkich szeregów.

O przedstawienia poszczególnych parametrów stosowałam 2 sposoby wizualizacji:

1. Używając punktów połączonych tylko w miejscach gdzie nie występowały brakujące wartości
2. Łącząc wszystkie znane wartości liniami prostymi

Na Rysunku 7 przedstawiam przykłady wykresów.



Rysunek 7: Wizualizacja przykładowych parametrów. Na wykresach po lewej w miejscach gdzie brakuje wartości obecne są nieciągłości, na wykresach po prawej znane wartości zostały połączone liniami prostymi.

2.4 Wybór parametrów

Określiłam dwa główne podejścia do wykorzystania danych które potencjalnie mogłyby pomóc pielęgniarkom w opiece nad pacjentem:

1. Użycie kombinacji kilku parametrów jako danych wejściowych Głębokiej Sieci Neuronowej (ang: Deep Neural Network - DNN). Sieć byłaby użyta w celu rozróżniania stabilnego i niestabilnego stanu pacjenta. Ostrzeżenia byłyby wywoływane w momencie wykrycia niestabilnego stanu co zwiększałoby prawdopodobieństwo, że zostanie on dostrzeżony wcześniej.
2. Użycie pojedynczego szeregu czasowego (jednego parametru) jako dane wejściowe Rekurencyjnej Sieci Neuronowej (ang: Recurrent Neural Network - RNN). Sieć byłaby użyta w celu przewidywania wartości monitorowanego parametru w kolejnym kroku czasowym. Ostrzeżenia byłyby wywoływane gdy przewidywana wartość parametru oznaczałaby pojawienie się niebezpiecznego stanu medycznego u pacjenta.

Oba te podejścia musiały zostać skonsultowane z Pielęgniarką Konsultantką aby określić ich zasadność.

Dodatkowo na podstawie poprzednich działań mogłam zauważyć, że jakość rejestru poszczególnych parametrów różniła się znacznie. Konieczne było wybranie parametrów najbardziej nadających się do przyszłego zastosowania w metodach uczenia maszynowego. Istniało jednak ryzyko, że wybór tylko kilku parametrów będzie skutkował brakiem możliwości przewidywania na ich podstawie stanu pacjenta.

Wyżej wymienione podejścia oraz wątpliwości przedyskutowałam z współpracującą Pielęgniarką:

1. Podejście z użyciem DNN: aby ustalić czy kombinacja tylko części parametrów niesie informację na temat stanu pacjenta zapytałam Pielęgniarkę czy byłaby w stanie określić czy stan pacjenta jest alarmujący na podstawie kilku wybranych parametrów. Otrzymałam odpowiedź twierdzącą. Na tej podstawie założyłam, że jest szansa, na odtworzenie tego samego procesu rozróżniania przy użyciu DNN.
2. Podejście z użyciem RNN: na podstawie konsultacji ustaliłam, że przewidywanie wartości poszczególnych parametrów w kolejnych krokach czasowych może być użyteczne dla pracy pielęgniarki ponieważ może umożliwić szybszą reakcję na zmiany stanu pacjenta.

Biorąc pod uwagę jakość danych oraz wiedzę Eksperta na temat ich znaczenia medycznego do kolejnych etapów projektu wybrałam kilka spośród zebranych parametrów. Były to: puls, ciśnienie skurczowe oraz rozkurczowe, saturacja, ośrodkowe ciśnienie żylne (CVP), oddechy całkowite oraz oddechy wymuszane (tzn. oddechy dostarczane pacjentowi z powodu mechanicznego wspomagania oddechu).

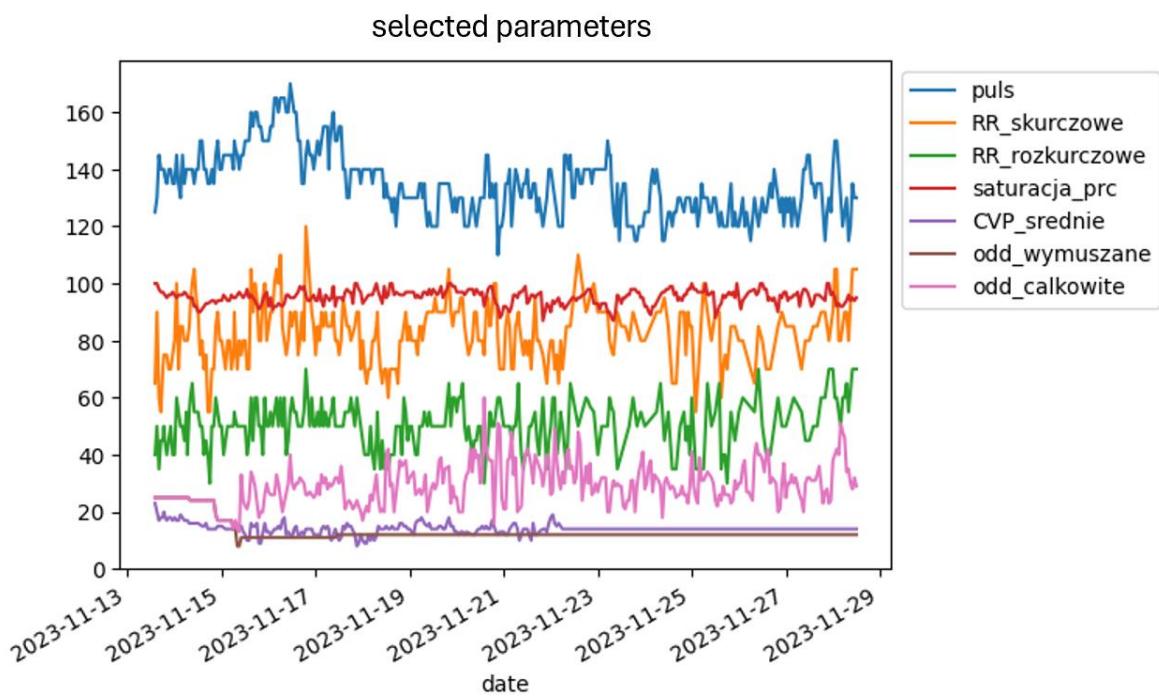
Wymienione parametry wyodrębniam z początkowego zestawu danych:

```
selected_parameters = ['puls', 'RR_skurczowe', 'RR_rozkurczowe',
'saturacja_prc', 'CVP_srednie', 'odd_wymuszane', 'odd_calkowite']

df = df_all.loc[:, ['date', *selected_parameters]]

df.to_csv('../data/df.csv', index=False)
```

Ww. parametry są ukazane na wykresie na Rysunku 8:



Rysunek 8: Parametry wybrane do dalszej pracy. Wykres wykonany w celu poglądowej wizualizacji. Oś pionowa jest wspólna dla wszystkich parametrów.

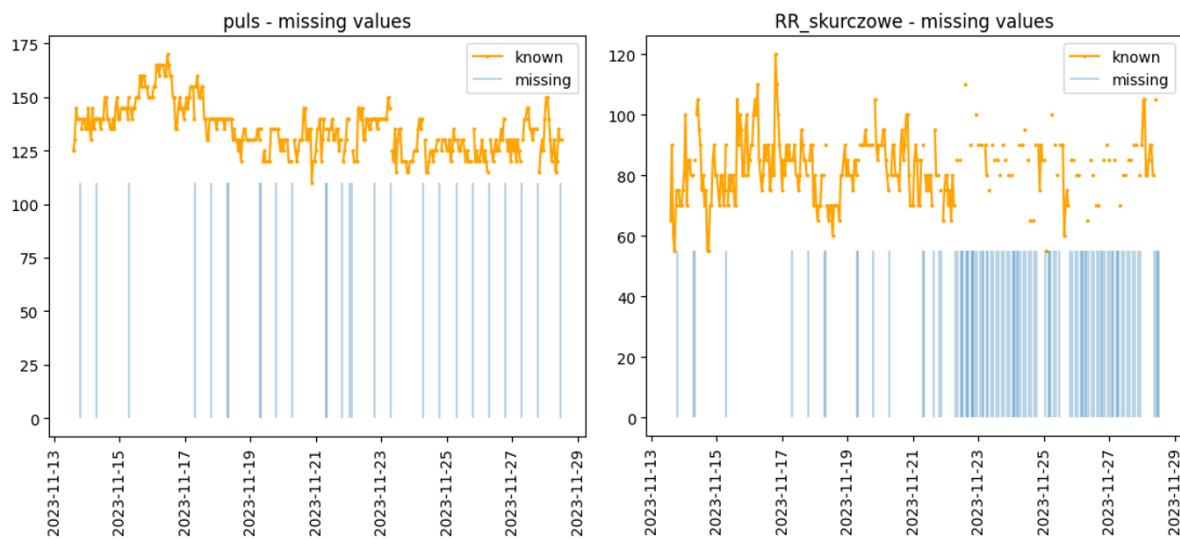
2.5 Wartości brakujące

Kolejnym krokiem była eksploracja brakujących wartości. Sprawdziłem najpierw jak wiele brakujących wartości jest obecnych w każdym szeregu czasowym. Sprawdziłem ich liczbę (*na_count*) oraz to jaki procent wszystkich wartości stanowiły (*na_prc*):

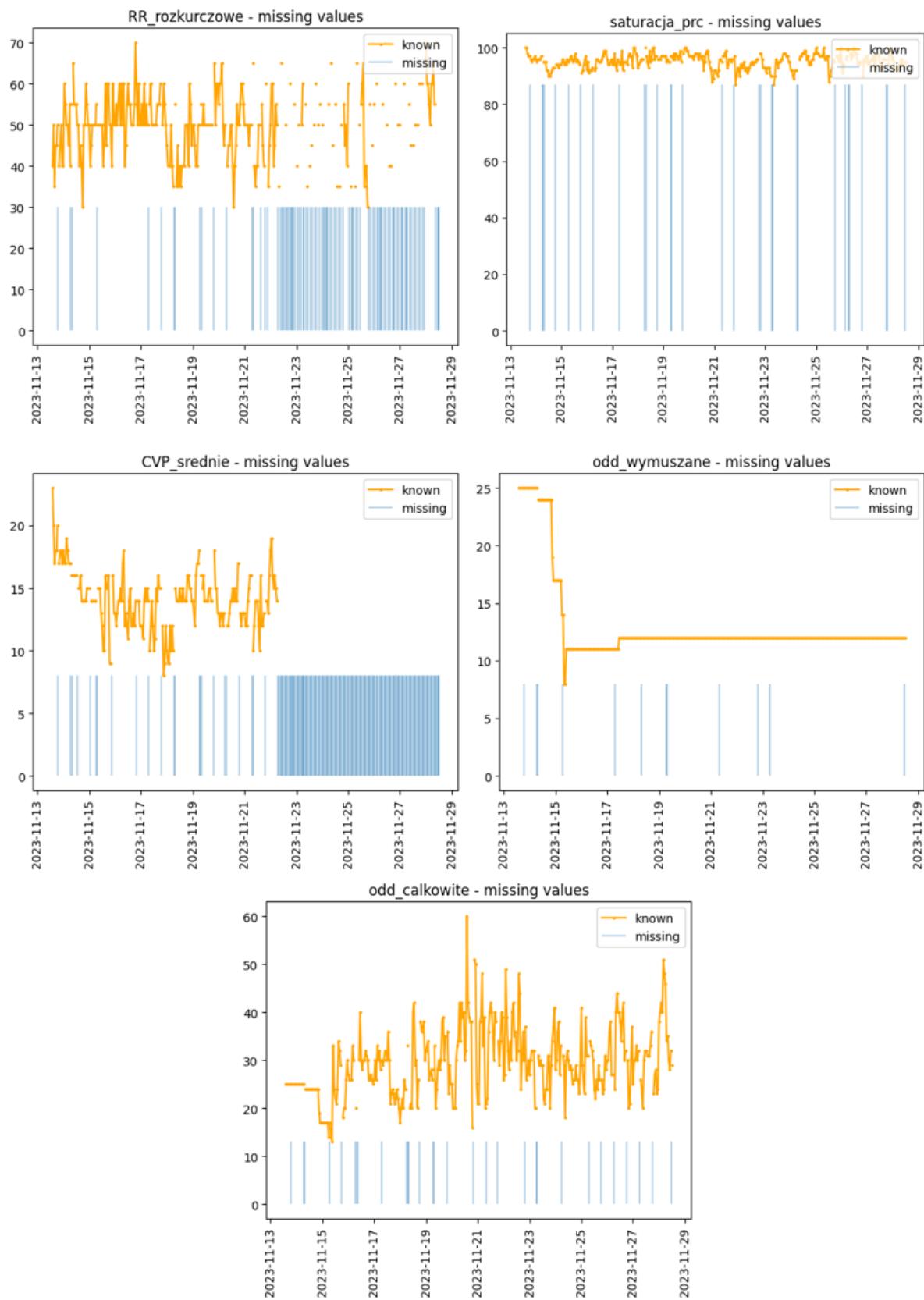
	na_count	na_prc
date	0	0.00
puls	27	7.28
RR_skurczowe	98	26.42
RR_rozkurczowe	98	26.42
saturacja_prc	31	8.36
CVP_srednie	176	47.44
odd_wymuszane	12	3.23
odd_calkowite	30	8.09

Rysunek 9: Zliczone wartości brakujące obecne w opracowywanych danych. *na_count* – liczba wartości brakujących, *na_prc* – procent wszystkich wartości, który stanowią.

Na tym etapie wiedziałem jak wiele brakujących wartości jest obecnych w danych. Nie wiedziałem jednak nadal jaki był sposób ich rozmieszczenia w poszczególnych szeregach. Żeby zeksplorować rozmieszczenie przestrzenne brakujących wartości skonstruowałem wykresy przedstawiające wszystkie szeregi z lokalizacjami brakujących wartości zaznaczonymi jako pionowe linie (Rysunki 10 i 11). Tworząc wykresy użyłem metody wizualizacji wartości nieznanych opisanej w artykule [2].



Rysunek 10: Wizualizacja brakujących danych. Lokalizacje brakujących wartości zostały zaznaczone pionowymi niebieskimi liniami.

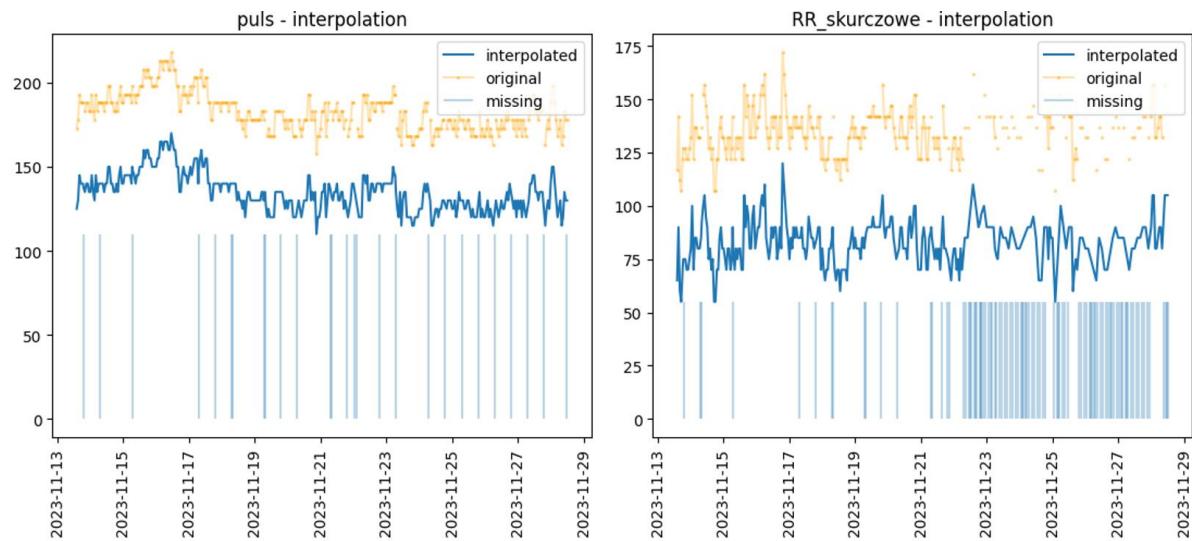


Rysunek 11: Wizualizacja brakujących danych. Lokalizacje brakujących wartości zostały zaznaczone pionowymi niebiskimi liniami.

Wykonane działania pozwoliły mi np. jasno ukazać, że wysoki procent wartości brakujących w szeregu opisującym CVP wynikał z faktu, że w pewnym momencie całkowicie zaprzestano obserwacji tego parametru oraz, że gęstość rozmieszczenia wartości brakujących w szeregach opisujących puls i saturację była w przybliżeniu stała.

2.6 Interpolacja Liniowa

Aby użyć szeregów czasowych jako dane treningowe np. Rekurencyjnej Sieci Neuronowej konieczne jest rozwiązywanie problemu brakujących wartości. Jednym ze sposobów na poradzenie sobie z ich obecnością jest interpolacja danych. Pierwszą metodą której spróbowałam była interpolacja liniowa. W kolejnych krokach utworzyłam zestaw interpolowanych liniowo danych i zwizualizowałam efekty interpolacji na wykresach. Na Rysunku 12 przedstawiam przykładowe wykresy.



Rysunek 12: Przykładowe wyniki uzupełniania brakujących wartości przy pomocy interpolacji liniowej

2.7 Interpolacja z użyciem dekompozycji sezonowej

Jako, że zasada działania interpolacji liniowej polega po prostu na łączeniu znanych wartości przy pomocy linii prostych postanowiłem spróbować jeszcze jednego podejścia do interpolacji. Dane, z którymi pracowałem, to szeregi czasowe opisujące parametry życiowe pacjenta obserwowane przez okres kilkunastu dni i zapisywane mniej więcej co godzinę. Z tego powodu mogłem spodziewać się wystąpienia w danych jakichś okresowych trendów (ich obecność mogłaby wynikać np. z tendencji niektórych parametrów do zmiany wartości w zależności od pory dnia lub ze względu na okresowe zmiany personelu oddziału, które następowały co 12 godzin (o 7 rano i 7 wieczorem) i mogły wpływać na pobudzenie dziecka).

Aby to uwzględnić postanowiłem dla każdego z szeregów:

1. po pierwsze wykonać dekompozycję sezonową
2. odjąć ustalony komponent sezonowy
3. wykonać interpolację liniową szeregu pozbawionego trendu sezonowego
4. odtworzyć szereg dodając usunięty wcześniej komponent sezonowy

Wzorowałem się na przykładzie opisanym w artykule [1]

Nim mogłem przeprowadzić dekompozycję sezonową musiałam ustalić stały krok czasowy dla każdego szeregu. Zdecydowana większość wartości była przypisana pełnym godzinom. W całym zestawie danych była tylko jedna obserwacja która została przypisana punktowi czasowemu innemu niż pełna godzina lub połowa pełnej godziny. Zdecydowałem się pominąć tą jedną obserwację, a do pozostałym danych dodać odpowiednią liczbę pustych wierszy tak aby uzyskać stały krok czasowy wynoszący 30 min. Na tak przygotowanym zestawie wykonałem dekompozycję sezonową. Przy dekompozycji sezonowej określiłem parametr *period* (okres) jako 48 próbek (co odpowiadało 24 godzinom). Po przeprowadzeniu interpolacji zredukowałem zestaw danych tak aby zawierał zapis parametrów jedynie o pełnych godzinach (stały jednogodzinny krok czasowy).

Do wykonania dekompozycji użyłem funkcji *STL* z biblioteki *statmodels*. Poniżej zamieszczam fragment klasy którą zdefiniowałem aby wykonać opisane operacje:

```
class DecompositionMonster:  
    def __init__(self, param, df_to_interpolate):  
        self.param = param  
        self.series_decomposed = []  
        self.missing_values_indieces = []  
        self.df_stl_int = df_to_interpolate.copy()  
        self.df_lin_int = df_to_interpolate.interpolate(method='linear')  
  
    def decompose(self, period=24):  
        self.missing_values_indieces =
```

```

df_to_interpolate[df_to_interpolate[self.param].isnull()].index

    stl = STL(self.df_lin_int[self.param], period=period, seasonal=7)
    self.series_decomposed = stl.fit()
    self.series_decomposed.plot()
def stl_interpolate(self):
    seasonal_component = self.series_decomposed.seasonal
    series_deseasonalised = df_to_interpolate[self.param] -
seasonal_component
        series_deseasonalised_interpolated =
series_deseasonalised.interpolate(method="linear")
        series_interpolated = series_deseasonalised_interpolated +
seasonal_component

    self.df_stl_int[self.param] = series_interpolated

```

Wyniki dekompozycji sezonowej każdego z szeregów przedstawione są na wykresach. Wykonałam także porównanie efektów interpolacji uzyskanych przy zastosowaniu metody liniowej i tej z użyciem dekompozycji sezonowej. Na wykresach zaznaczyłam punktami miejsca

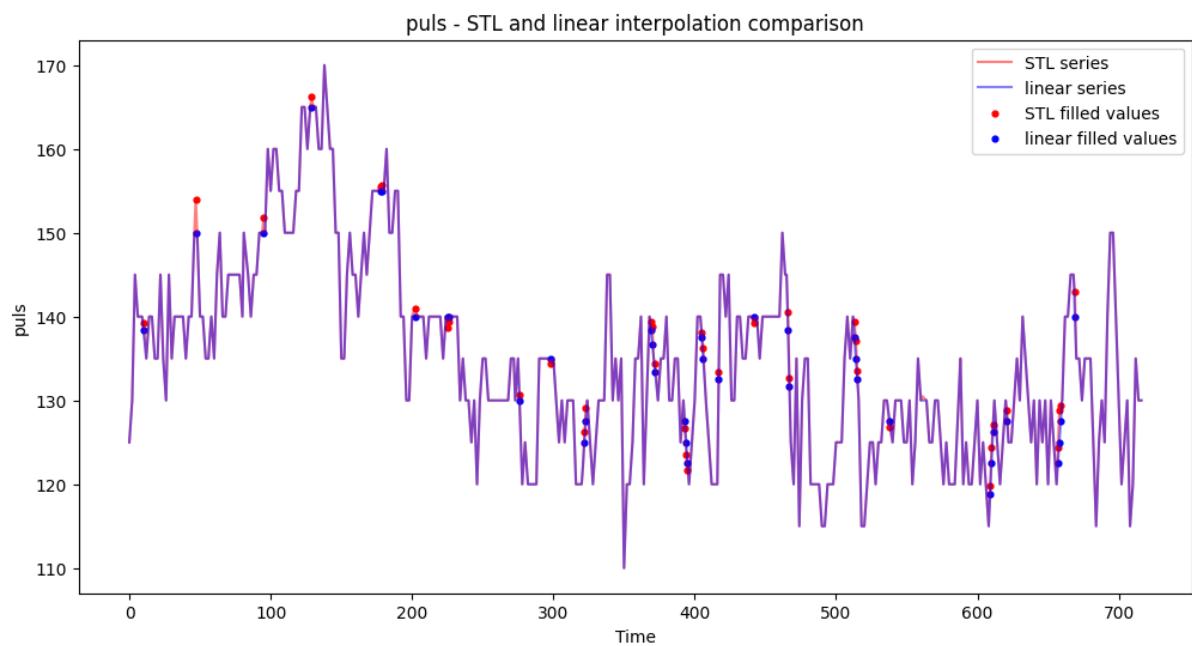
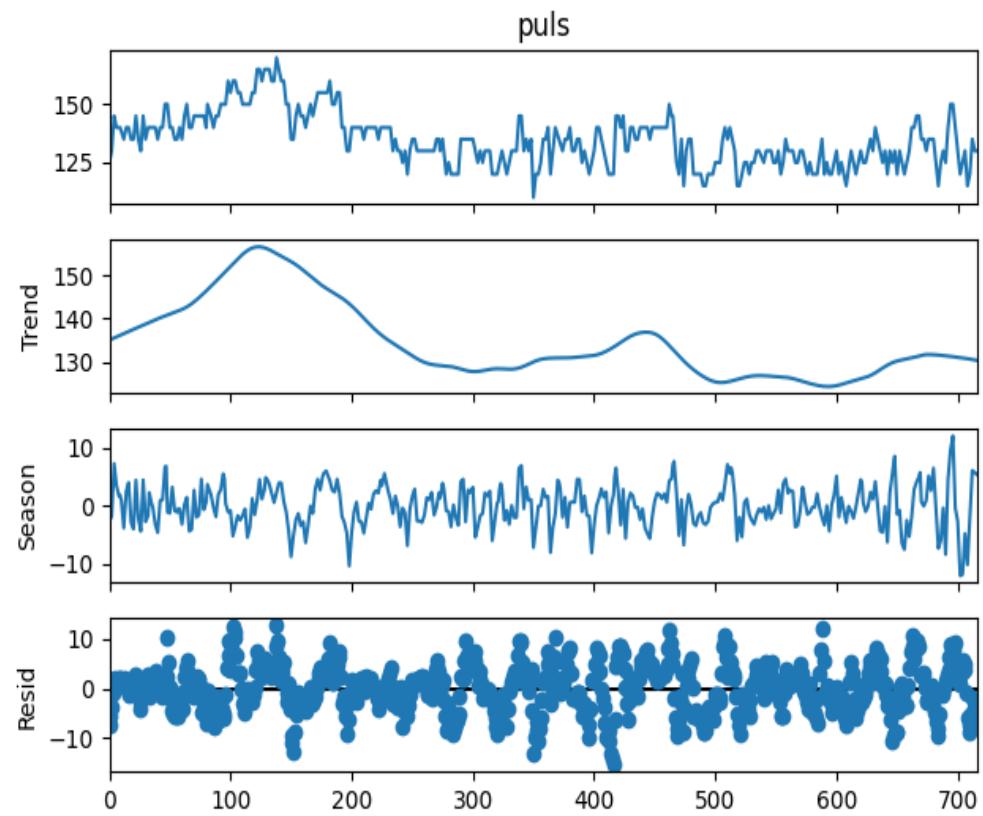
gdzie dane wypełnione obiema metodami różniły się w większym stopniu. Poniżej przedstawiam kolejne wykonane kroki:

```

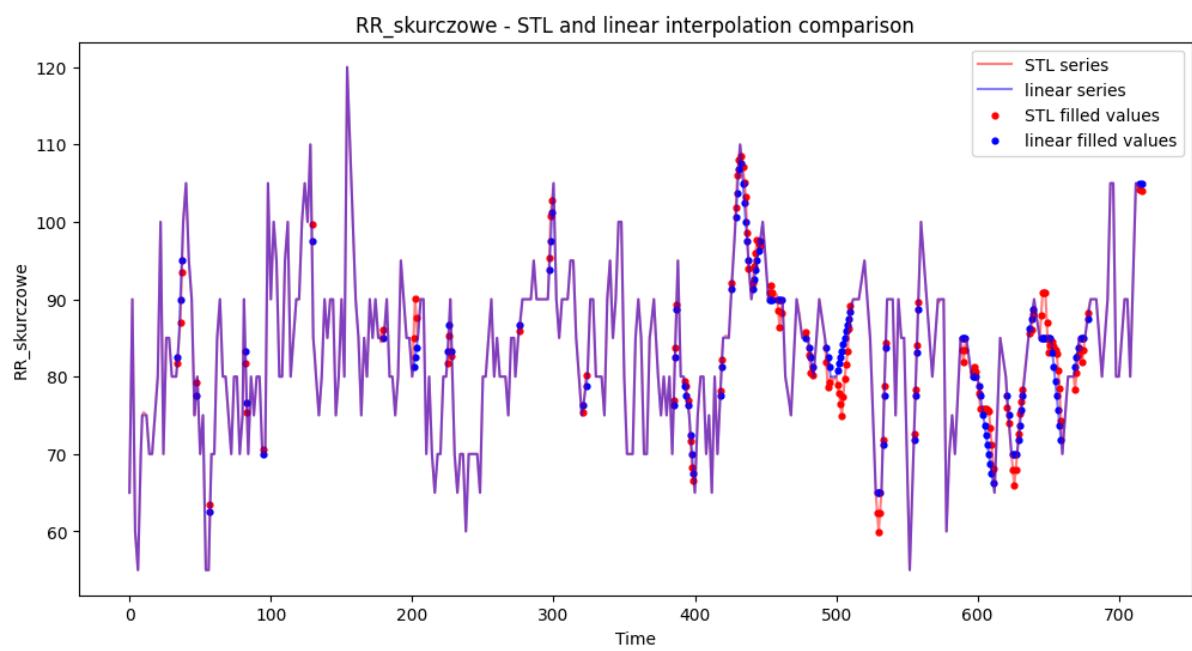
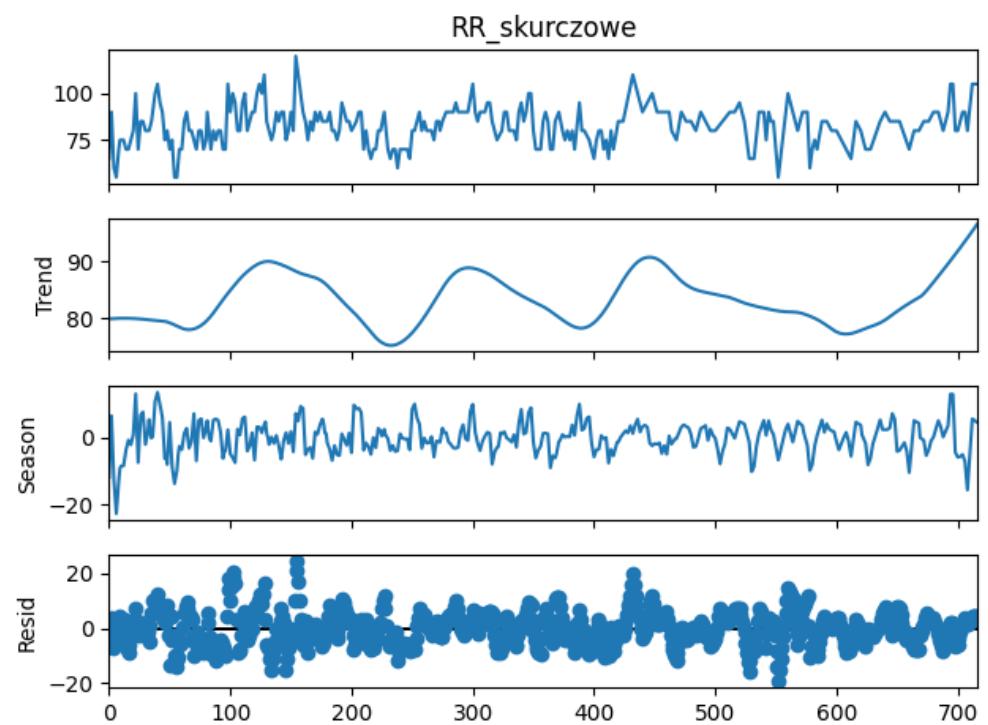
for param in columns:
    dm = DecompositionMonster(param, df_to_interpolate=df_to_interpolate)
    dm.decompose(period=48)
    dm.stl_interpolate()
    dm.comparison_plot_both()
    df_to_interpolate_updated = dm.df_stl_int
    df_to_interpolate = df_to_interpolate_updated

```

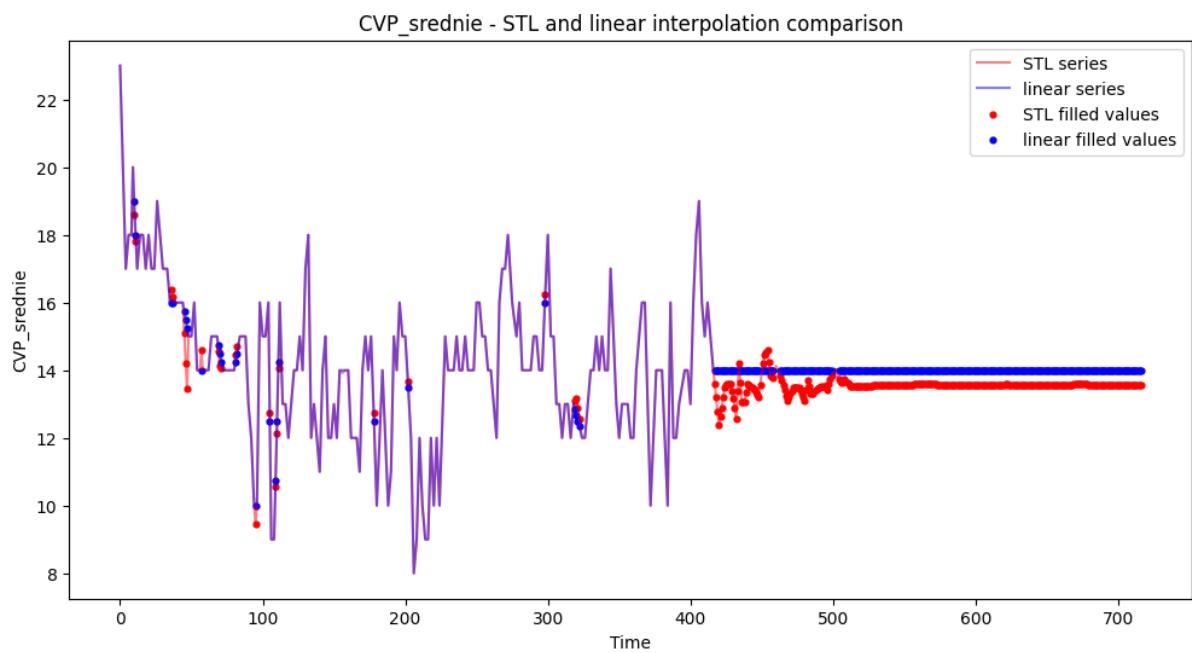
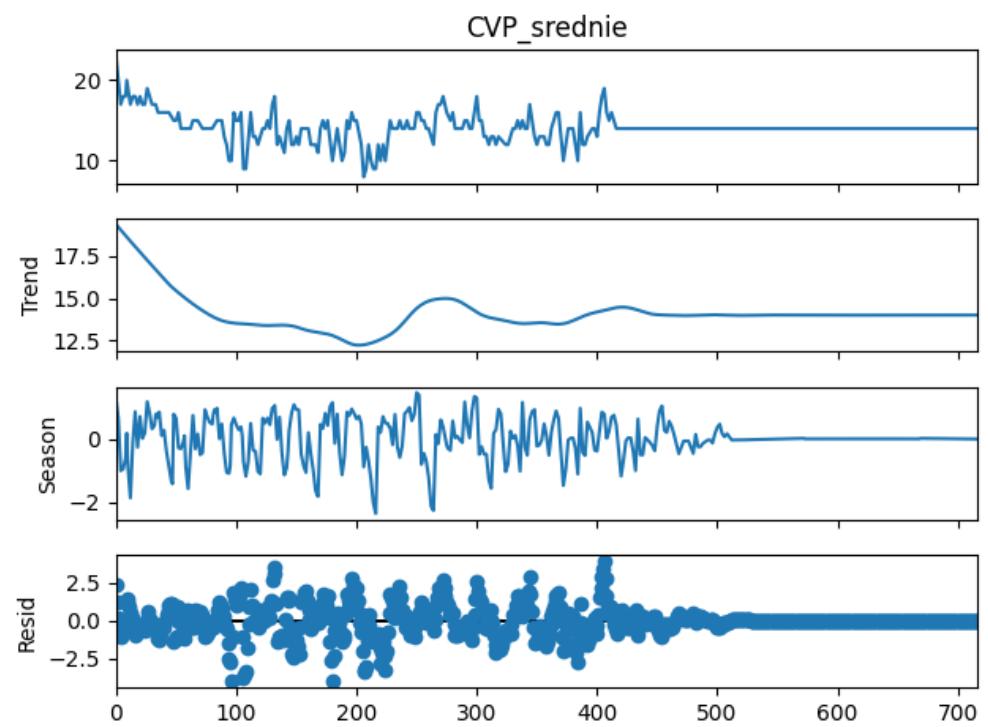
Przykłady wizualizacji wyników znajdują się na kolejnych stronach na Rysunkach 13, 14 i 15.



Rysunek 13: Puls - dekompozycja sezonowa (wykres górnny) i porównanie 2 metod interpolacji (wykres dolny)



Rysunek 14: Ciśnienie skurczowe - dekompozycja sezonowa (wykres górnny) i porównanie 2 metod interpolacji (wykres dolny)



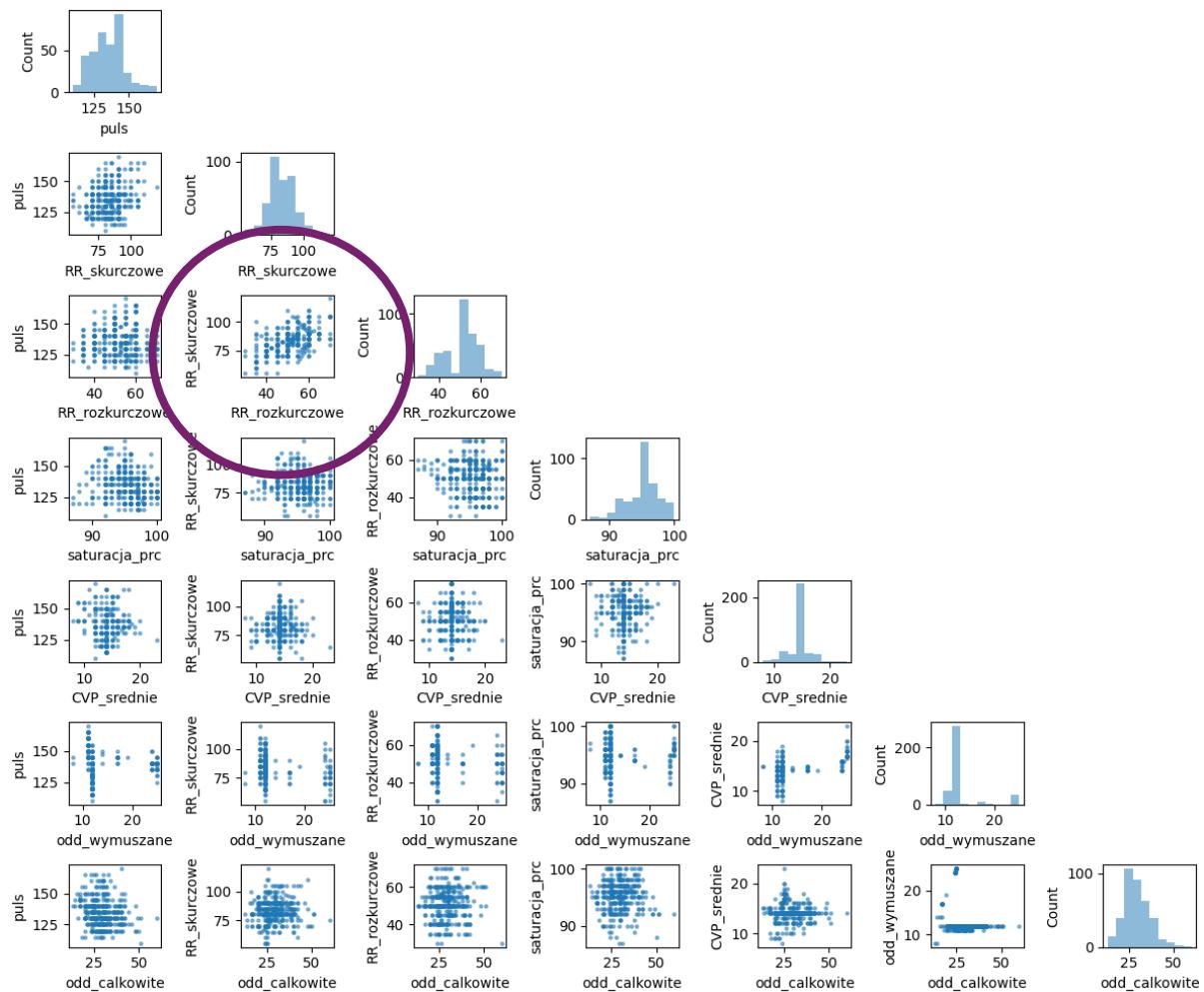
Rysunek 15: CVP - dekompozycja sezonowa (wykres górnny) i porównanie 2 metod interpolacji (wykres dolny)

Podejście do interpolacji danych powinno zostać zmodyfikowane gdybym zdecydowała się w dalszej pracy korzystać z szeregu pisującego parametr CVP (który całkowicie przestał być zapisywany na pewnym etapie obserwacji pacjenta). Było jednak interesujące porównać jak wyglądała ekstrapolacja szeregu na tak długim odcinku czasowym uzyskana przy stosowaniu metod interpolacji.

Po interpolacji zaokrągliałam uzyskane wartości tak aby odpowiadały dokładności, z którą zbierane były oryginalne próbki (np. dokładność zapisu wartości pulsu wynosiła 5 jednostek). Do dalszej pracy używałam danych interpolowanych przy wykorzystaniu dekompozycji sezonowej.

2.8 Korelacje

Spróbowałam krótko zbadać korelację między parametrami. W tym celu przedstawiłam parametry na wykresie *scatterplotmatrix* (z biblioteki *mlxtend.plotting*). Można było przypuszczać istnienie korelacji między ciśnieniem skurczowym i rozkurczowym (co nie jest zaskakujące ze względu na znaczenie biologiczne parametrów).



Rysunek 16: Wykres 'scaterplotmatrix' z zaznaczonym wykresem zestawiającym ciśnienia skurczowe i rozkurczowe, którego kształt wskazuje na możliwą korelację.

2.9 Augmentacja danych

Końcowym celem projektu było użycie opracowanych danych w technikach uczenia maszynowego. Na tym etapie pracy nadal nie było wiadomo czy uda nam się uzyskać ze szpitala dokumentację medyczną większej liczby pacjentów (dotychczas dysponowałam danymi jednego pacjenta obserwowanego przez ok. 15 dni). Z tego powodu wykonałam próbę augmentacji danych. Wygenerowanie wielu szeregów czasowych na bazie tylko jednego szeregu oryginalnego z pewnością nie mogło zastąpić rzeczywistej różnorodności zmian parametrów życiowych. Dodatkowe rzeczywiste dane najpewniej zawierałyby inne schematy występujących w nich anomalii. Metoda augmentacji użyta w tym kroku bazuje na modyfikacji kształtu oryginalnego szeregu. Modyfikacje zostały wykonane przez wykonanie Szybkiej Transformaty Fouriera szeregu, dodanie szumu losowego do uzyskanego widma częstotliwościowego, a następnie wykonanie Transformacji Odwrotnej w celu uzyskania zmodyfikowanego szeregu.

Metoda augmentacji została zaproponowana przez Promotora mojego projektu inżynierskiego. Był on autorem wyjściowego skryptu, który modyfikowałam aby przeprowadzić augmentację. Poniżej przedstawiam funkcję opisującą metodę augmentacji:

```
def fourier_augment_dc(time_series: np.ndarray | pd.core.series.Series,
                        noise_factor: float = 0.1) -> np.ndarray:

    freq_spectrum = np.fft.fft(time_series)
    original_dc_component = freq_spectrum[0]

    # Apply scaling and noise to all components
    noise = np.random.normal(size=freq_spectrum.shape) * noise_factor
    modified_freq_spectrum = freq_spectrum + noise

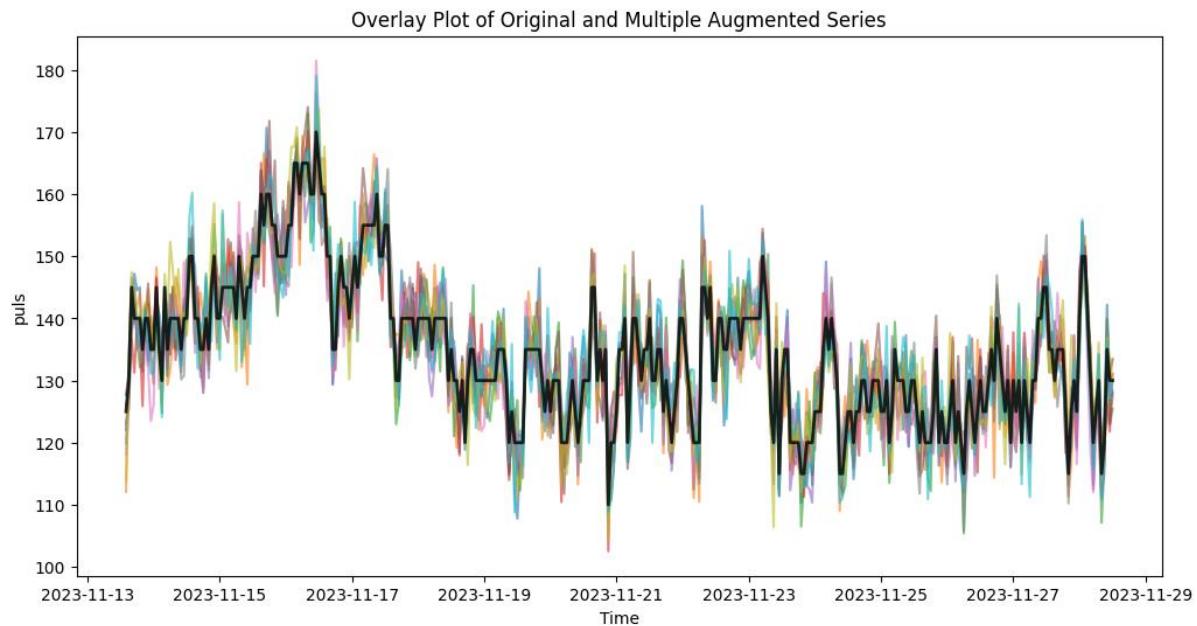
    # Restore unscaled DC component
    modified_freq_spectrum[0] = original_dc_component

    augmented_time_series = np.fft.ifft(modified_freq_spectrum).real
    return augmented_time_series
```

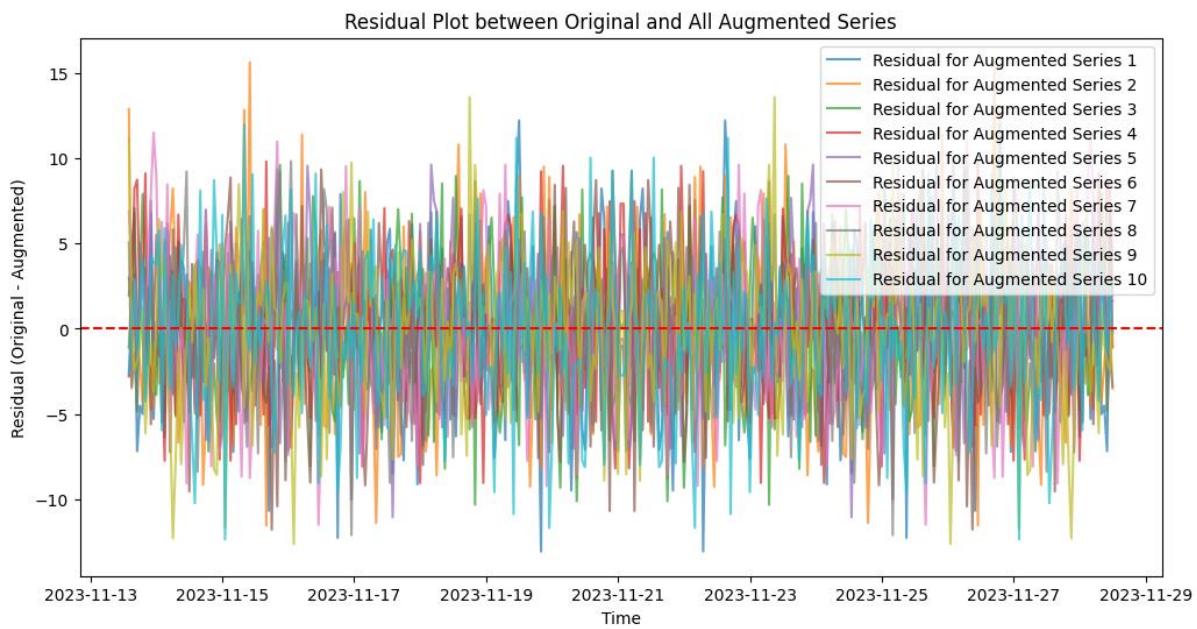
Kolejne etapy pracy przedstawiają augmentację szeregu opisującego puls. Użyłam parametrów:

```
blur_rate = 2
noise_factor= (max(original_series) - min(original_series)) * blur_rate
```

Na Rysunku 17 przedstawiam wykres ukazujący szereg wyjściowy i 10 szeregów wygenerowanych oraz a na Rysunku 18 wykres przedstawiający różnicę między otrzymanymi szeregami wygenerowanymi, a szeregiem oryginalnym.



Rysunek 17: Szereg oryginalny (linia czarna) oraz 10 wygenerowanych szeregów (linie kolorowe)



Rysunek 18: Różnice pomiędzy szeregiem wygenerowanym i szeregiem oryginalnym

Wynikiem tego etapu pracy był zestaw danych składający się z zadanej liczby argumentowanych szeregów czasowych. Celem było użycie sztucznie wytworzonych danych jako placeholder rzeczywistych danych pochodzących od większej liczby pacjentów, do których nie mieliśmy dostępu. W kolejnych etapach pracy jeden wytworzony szereg został użyty do zbadania zachowania sieci neuronowej (próby sprawdzenia stosowności tej metody augmentacji).

3 PREDYKCJA ANOMALII

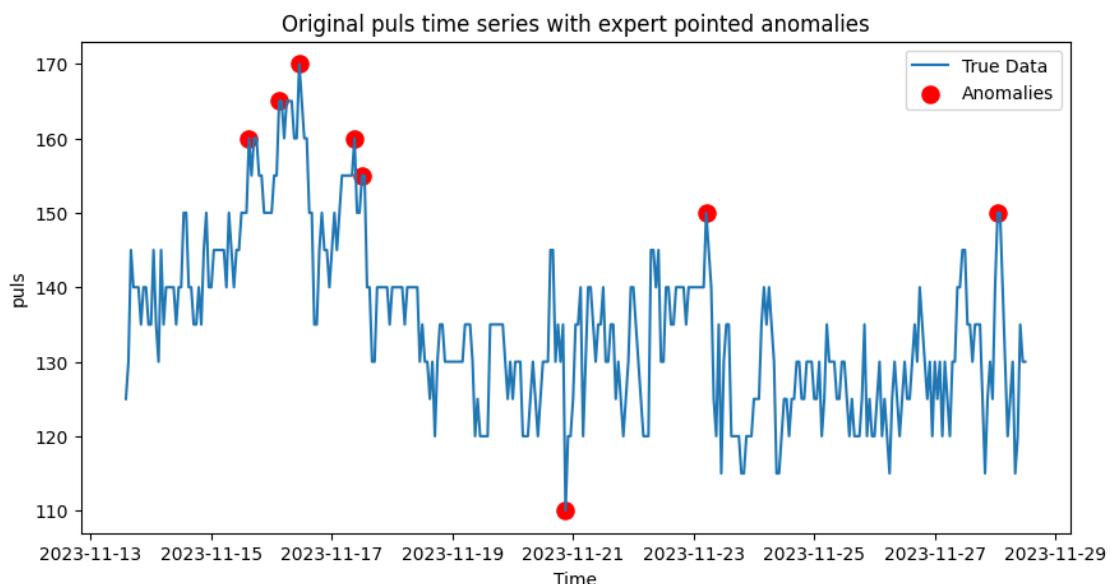
3.1 Prognozowanie bez dostrajania

Wyjściowym pomysłem stojącym za wykonaniem projektu było użycie technik uczenia maszynowego w celu przewidywania alarmujących stanów medycznych obserwowanych u pacjentów. Jak opisałam wcześniej, rozważałam użycie Rekurencyjnej Sieci Neuronowej do prognozowania wartości pojedynczych szeregów czasowych i wykrywania anomalii oraz Sieci Głębokiej do determinowania czy stan pacjenta jest stabilny. Ostatnim etapem projektu było zbadanie przykładu pierwszego z wymienionych podejść.

Workflow opisuję poniżej:

Na tym etapie wykonywałam operacje na pojedynczym szeregu czasowym. Wybrałam początkowo puls, saturację i oddechy całkowite jako najbardziej odpowiednie. Przy wyborze kierowałam się początkową jakością surowych danych (ilością wartości nieznanych) oraz ich kształtem (liczba oddechów wymuszonych mogła być użyteczna w połączeniu z innymi parametrami jednak nie nadawała się do zastosowania w predykcji z użyciem pojedynczego parametru ponieważ wykazywała mają zmienność, a jej wartość zależała od ustawień urządzenia wspomagającego oddech). Na drodze konsultacji z współpracującą Pielęgniarką ustaliłam, że w przypadku wszystkich trzech parametrów anomalie przybierają formę pików na wykresie. Konsultująca Pielęgniarka określiła i zaznaczyła lokalizacje anomalii które występowały w danych opisujących 3 ww. wybrane na tym etapie parametry.

Dalsze kroki wykonałam używając danych opisujących puls. Wykres przedstawiający wskazane przez Ekspertkę anomalie przedstawiam na Rysunku 19.



Rysunek 19: Szereg czasowy opisujący puls wraz z zaznaczonymi przez Ekspertkę anomaliami

Ideą dalszej pracy była symulacja sytuacji kiedy:

1. Początkowo, przez określoną liczbę godzin zbierane są dane obserwacji pulsu
2. Następnie Sieć Neuronowa (NN) jest trenowana na zebranych danych

3. W kolejnych krokach czasowych NN przewiduje oczekiwana wartość pulsu, a występująca wartość rzeczywista jest równocześnie rejestrowana

4. Jeżeli zarejestrowana wartość różni się znaczco od wartości przewidywanej przez NN alert jest wywoływany.

Jako, że opisywane dane są przykładem szeregów czasowych użyłam tzw. Long Short-Term Memory (LSTM) Recurrent Neural Network. Sieć Rekurencyjna w procesie prognozowania wykorzystuje sekwencję poprzednich kroków czasowych - uwzględnia kolejność i wartość każdego z nich.

Wykorzystywałam 2 architektury Sieci, obie zostały zaproponowane przez Promotora mojego projektu inżynierskiego. Poniżej przedstawiam użyte modele:

- i. LSTMForecaster – to prosty model składający się z jednej warstwy LSTM i jednej warstwy Liniowej. Posiadał 64 parametry stanu ukrytego. Wartością wyjściową była wartość przewidywana dla następnego kroku czasowego w postaci skalara

```
class LSTMForecaster(nn.Module):
    def __init__(self, input_dim=1, hidden_dim=64, output_dim=1):
        super(LSTMForecaster, self).__init__()
        self.lstm = nn.LSTM(input_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        return self.fc(lstm_out)
```

- ii. LSTMAttentionForecaster – to modyfikacja pierwszego modelu. 2 dodatkowe warstwy liniowe, mechanizm Dropout oraz funkcja aktywacyjna ReLu zostały wkomponowane modyfikując wykonywane obliczenia.

```
class LSTMAttentionForecaster(nn.Module):
    def __init__(self, input_dim=1, hidden_dim=128, num_layers=2,
                 output_dim=1, dropout=0.2):
        super(LSTMAttentionForecaster, self).__init__()
        # LSTM Layers
        self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers=num_layers,
                            batch_first=True, dropout=dropout)
        # Attention Layer
        self.attention = nn.Linear(hidden_dim, 1)

        # Fully Connected Layers for Output
        self.fc1 = nn.Linear(hidden_dim, hidden_dim // 2)
        self.fc2 = nn.Linear(hidden_dim // 2, output_dim)

        # Regularization
        self.dropout = nn.Dropout(dropout)
        self.relu = nn.ReLU()
```

```

def forward(self, x):
    lstm_out, _ = self.lstm(x)

    # Attention Mechanism
    attention_weights = torch.softmax(self.attention(lstm_out), dim=1)
    context_vector = torch.sum(attention_weights * lstm_out, dim=1)

    # Fully Connected Layers
    x = self.dropout(self.relu(self.fc1(context_vector)))
    output = self.fc2(x)
    return output

```

Porównałam wyniki eksperymentu dla:

1. 2 szeregów czasowych opisujących puls (danych rzeczywistych oraz jednego szeregu wygenerowanego na drodze augmentacji)
2. 2 architektur modeli sieci neuronowej
3. Kilku długości sekwencji danych używanych do uczenia sieci oraz późniejszej predykcji (w poszczególnych przypadkach sekwencje 2, 3, 5, 8, 12 i 20 wartości były podawane na wejście sieci, zanim uzyskiwana ostateczna była predykcja wartości kolejnej)

Przed użyciem dane znormalizowałam tak aby wartości pulsu zawierały się w przedziale [-1;1].

Sieć była uczona na pierwszych 100 próbkach przy użyciu tak zdefiniowanej funkcji:

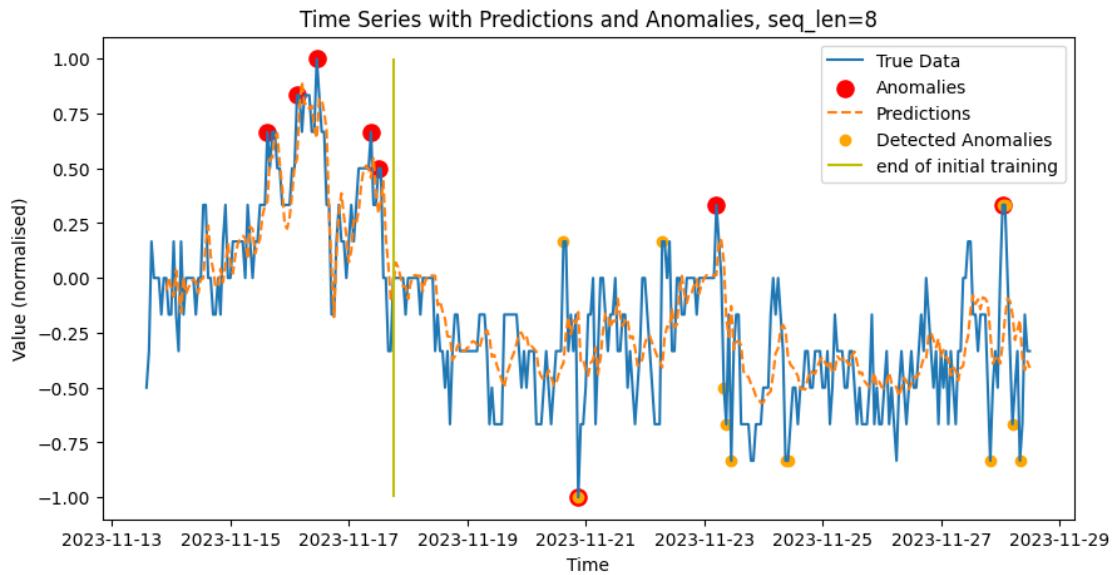
```

def train_model(model, X, Y, training_series_length=None, epochs=100):
    criterion = nn.MSELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()
        pred = model(X[0:training_series_length])
        if type(model) == LSTMForecaster:
            loss = criterion(pred[:, -1, :], Y[0:training_series_length])
        elif type(model) == LSTMAttentionForecaster:
            loss = criterion(pred, Y[0:training_series_length])
        loss.backward()
        optimizer.step()

```

Pożądanym rezultatem byłoby wskazanie jako anomalii trzech punktów występujących po zakończeniu wstępnie zbierania danych, na których sieć była uczona (na wykresach wskazałam ten moment pionowa zielona linią) Przykładowy rezultat otrzymany dla danych oryginalnych przedstawiam na Rysunku 20. Wyniki zostaną dokładniej omówione w sekcji „PORÓWNANIE WYNIKÓW”



Rysunek 20: Wyniki eksperymentu dla długości sekwencji = 8 wartości, model = LSTMForecaster

3.2 Prognozowanie z dostrajaniem

Jako finalny etap zmodyfikowałem wcześniejszy eksperiment dodając dostrajanie, czyli doszkalanie wstępnie przeszkolonego modelu na nowych danych. Po upływie wybranej liczby kroków czasowych wykonywałem dostrajanie modelu aby uwzględnić dodatkowe dane zebrane podczas dalszej obserwacji pacjenta.

Kod źródłowy opisujący mechanizm wykonywania dostrajania przedstawiam poniżej:

```

for i in range(0, len(X)):
    if i == training_series_length:
        train_model(model,
                    X,
                    Y,
                    training_series_length,
                    epochs=training_series_length)
    initial_pred = model_predict(
        model, X[0:training_series_length + finetuning_step])
    pred = np.append(pred, initial_pred)
    # print(f'len(pred)={len(pred)}')
    if all([i > training_series_length,
           (i - training_series_length) % finetuning_step == 0]):
        # print(f'i={i}')

        training_start = i - finetuning_step
        training_end = i
        train_model(model,
                    X[training_start:training_end],
                    Y[training_start:training_end],
                    epochs=finetuning_step)

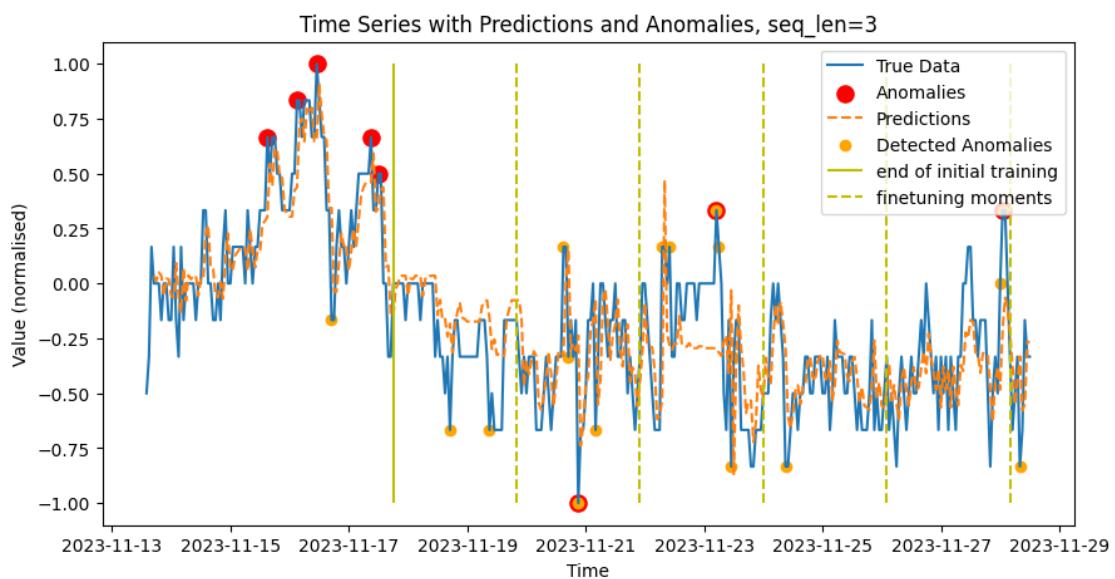
```

```

if training_end + finetuning_step <= len(X):
    new_pred = model_predict(
        model, X[training_end:training_end + finetuning_step])
elif training_end + finetuning_step > len(X):
    new_pred = model_predict(model,
                             X[training_end:])
pred = np.append(pred, new_pred)

```

Wystąpiła widoczna różnica względem wcześniejszej metody, w której sieć otrzymywała informacje tylko o stanie pacjenta podczas pierwszych 100 godzin. Dla części konfiguracji parametrów modele były w stanie wskazać wszystkie 3 anomalie. Przykładowy rezultat otrzymany dla danych oryginalnych przedstawiam na Rysunku 21. Wyniki tej jak i poprzedniej wersji eksperymentu zostaną dokładniej omówione w sekcji „PORÓWNANIE WYNIKÓW”



Rysunek 21: Wyniki eksperymentu dla długości sekwencji = 3 wartości, model = LSTMForecaster

4 PORÓWNANIE WYNIKÓW

4.1 Dane rzeczywiste

4.1.1 Dostrajanie vs bez dostrajania

Na Rysunkach 22 i 23 przedstawiam wykresy, na których zestawiłem wyniki predykcji uzyskane dla różnych długości sekwencji (długość sekwencji oznacza tutaj długość sekwencji zastosowaną przy uczeniu modelu oraz przy wykonywaniu predykcji). Poniżej fragment kodu służącego do wykonania ww. wykresów:

```
if anomalies != []:
    plt.scatter(x[anomalies], y[anomalies], s=200/(fig_scaler**2),
                color='red', alpha=1, label="True Anomalies")

plt.plot(x, y, label="True Data", linewidth=line_width)
for i, seq_len in enumerate(seq_lengths):
    pred = pred_dict[seq_len]
    detected_anomalies = detected_anomalies_dict[seq_len]
    pred_color = colors[i]
    anomalies_tilt = i*.05

    plt.scatter(
        x[detected_anomalies],
        y[detected_anomalies] + anomalies_tilt * np.sign(y[detected_anomalies]),
        color=pred_color,
        label="seq_len=" + str(seq_len) + ", Detected Anomalies",
        s=40/(fig_scaler**2))

if finetuning:
    finetuning_ends = list(range(training_series_length + finetuning_step,
                                 len(x), finetuning_step))
    plt.vlines(x[finetuning_ends], -1, 1, colors='y', linestyles='dashed',
               linewidth=line_width, label='finetuning moments')
    plt.vlines(x[training_series_length], -1, 1, color='y', linewidth=line_width,
               label='end of initial training')
else:
    plt.vlines(x[training_series_length], -1, 1, color='y',
               linewidth=line_width, label='end of training series')

plt.legend(loc='upper left', bbox_to_anchor=(0, -0.1), ncol=3)
plt.title('Zestawienie predykcji anomalii dla wszystkich testowanych długości sekwencji',
          fontsize = 15/fig_scaler)
plt.suptitle(model_class.__name__ + ', ' + series_name + ', without finetuning')
plt.xlabel('date')
plt.ylabel(param + ' (normalised)')
```

Anomalie wskazane dla różnych długości sekwencji zaznaczylem odrębnymi kolorami. Pierwsze 2 wykresy przedstawiają wyniki uzyskane dla obu modeli bez użycia dostrajania, kolejne 2 z jego zastosowaniem.

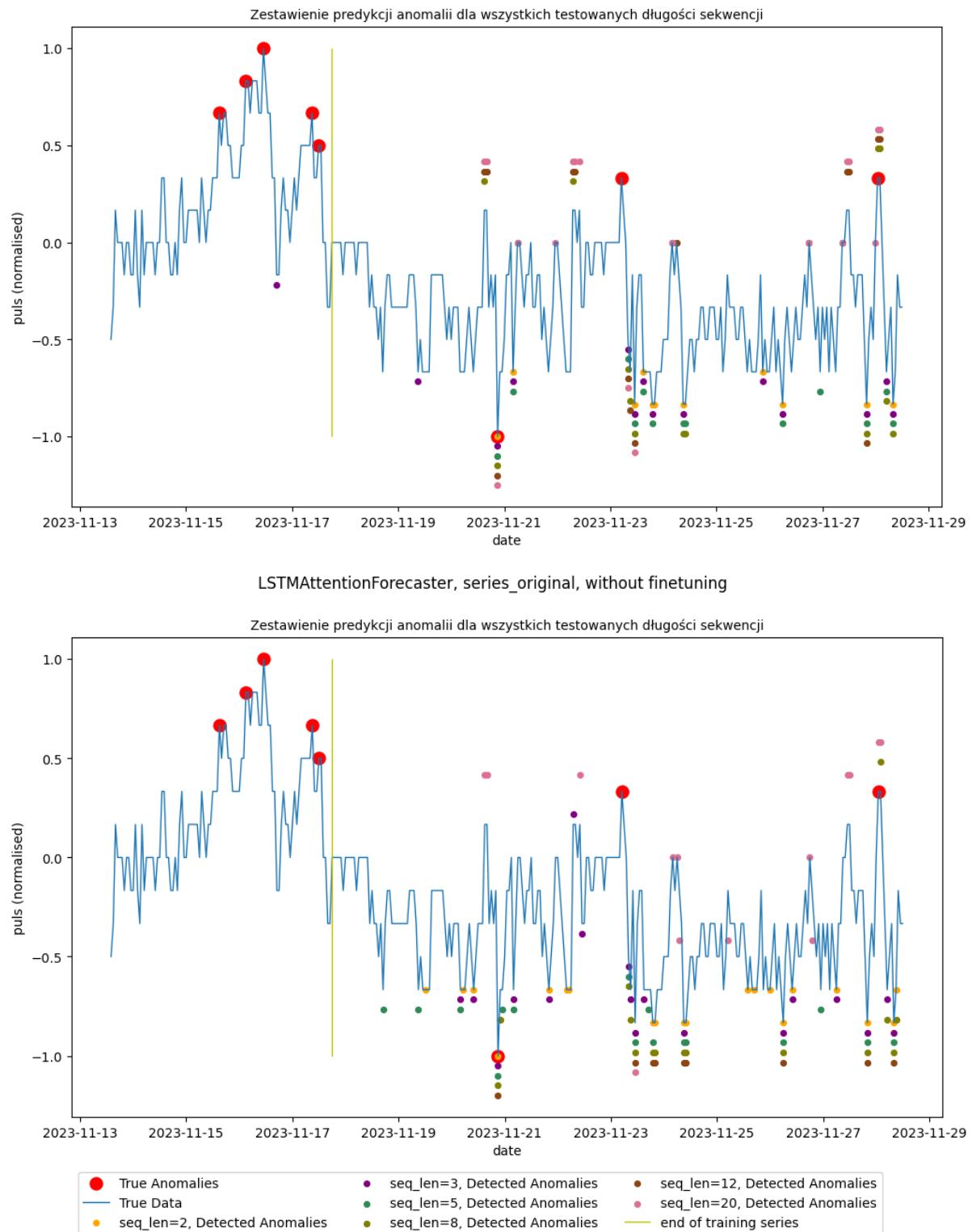
Pierwsze 5 rzeczywistych anomalii wystąpiło w pierwszych 100 godzinach obserwacji pacjenta. Tak jak opisywałam wyżej pierwsze 100 godzin był to okres, w którym zbierane były dane do początkowego uczenia modelu.

Dopiero dla kolejnych godzin model miał być stosowany go predykcji anomalii. Z tego powodu interesuje nas jak poszczególne przypadki radziły sobie ze wskazywaniem kolejnych 3 anomalii, które wystąpiły w późniejszym czasie.

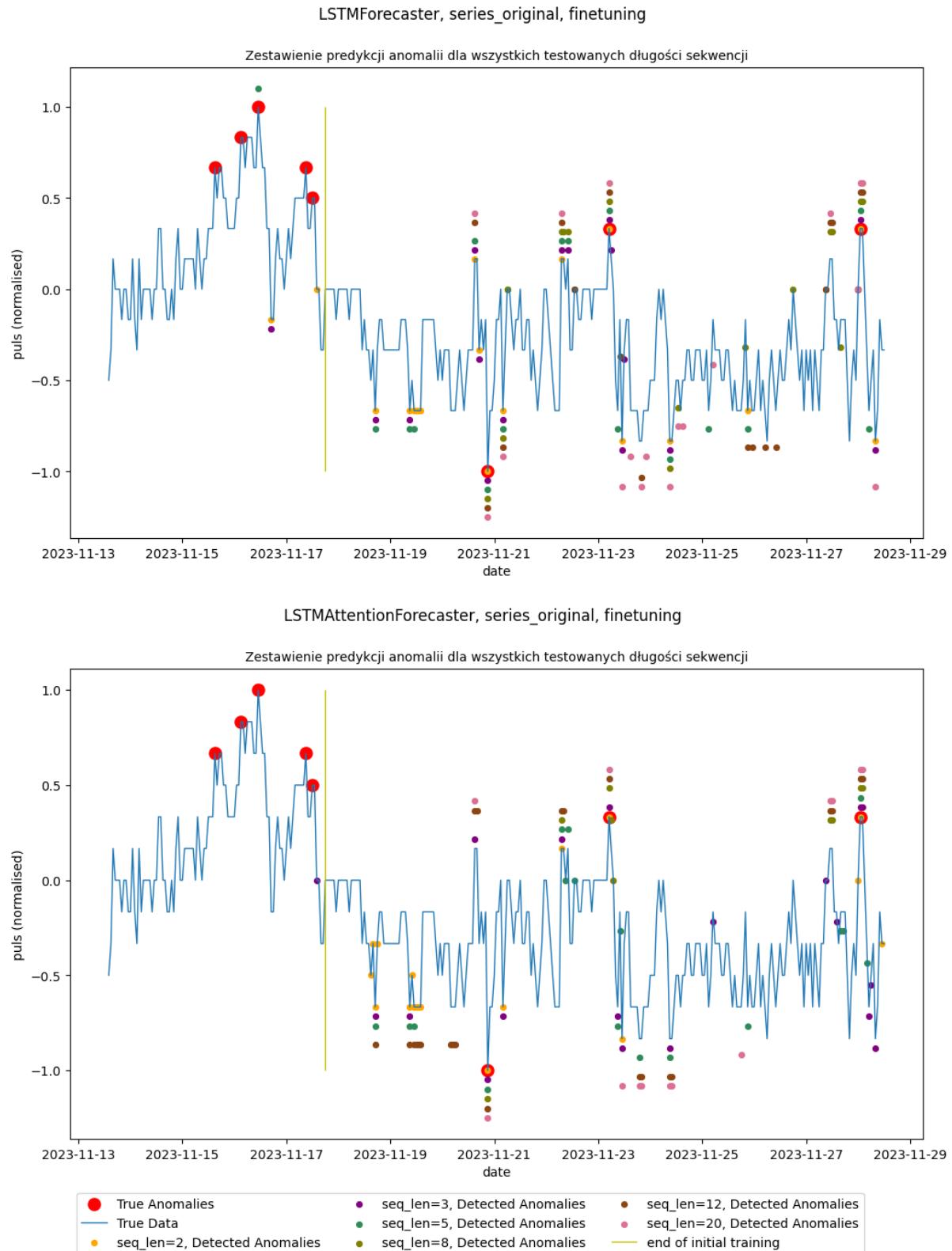
W obu przypadkach jako potencjalne anomalie wskazanych zostało znacznie więcej momentów niż 3 rzeczywiste określone przez specjalistę. Można jednak zauważyć, że po zastosowaniu dostrajania wykrywalność rzeczywistych anomalii znacznie wzrosła. Jest to szczególnie widoczne w przypadku drugiej spośród 3 interesujących nas anomalii, która bez stosowania dostrajania w ogóle nie została wykryta, a po jego zastosowaniu została wskazana przy większości stosowanych długości sekwencji.

Po zastosowaniu dostrajania widocznie zmniejszyła się także liczba niepotrzebnie wskazywanych punktów.

LSTMForecaster, series_original, without finetuning

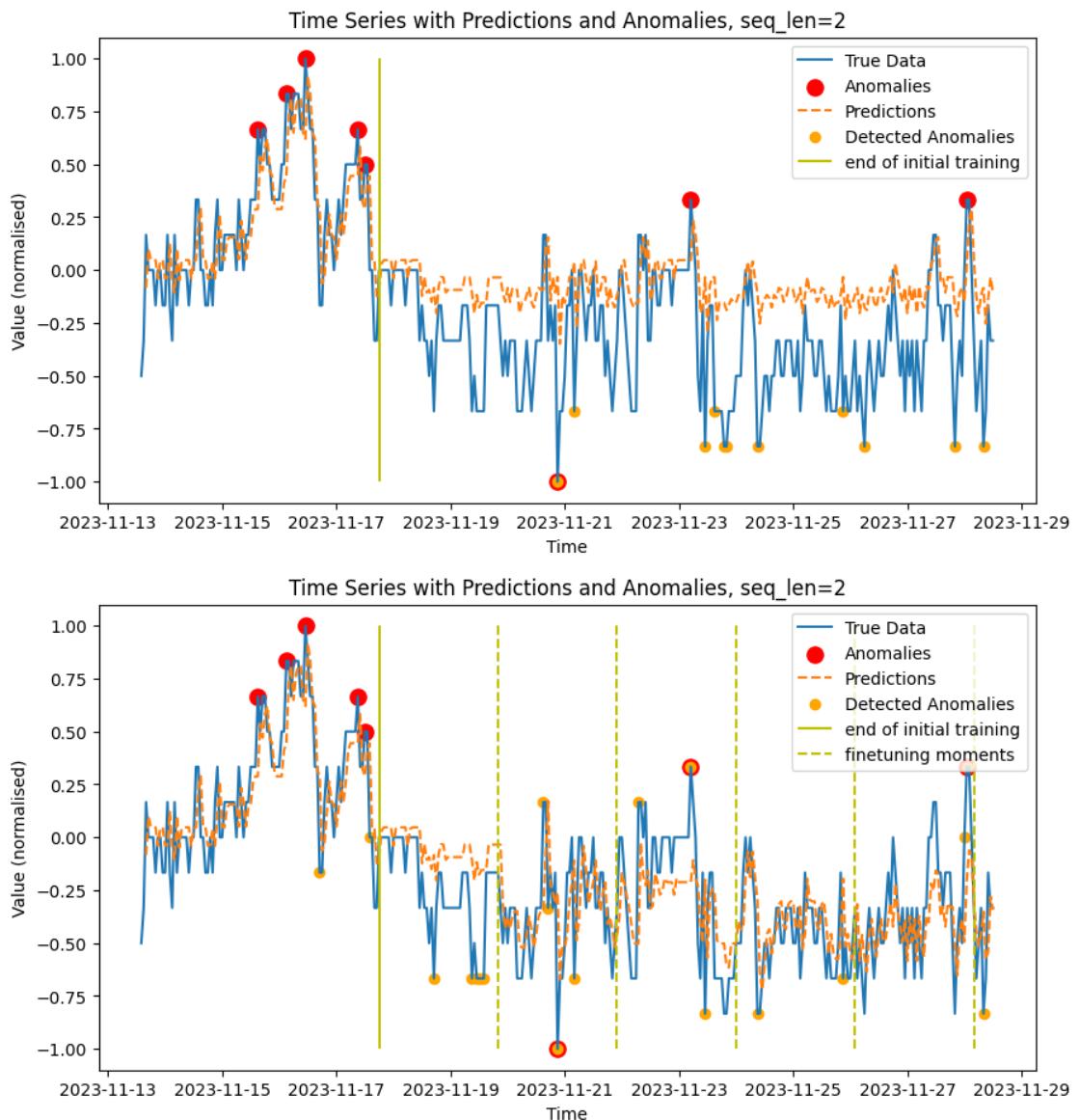


Rysunek 22: Porównanie predykcji anomalii wykonanych przez 2 modele, przypadek bez dostrajania



Rysunek 23: Porównanie predykcji anomalii wykonanych przez 2 modele, przypadek z dostrajaniem

Wpływ dostrajania jest także widoczny kiedy porównana się przebieg predykcji wartości pulsu dla tej samej długości sekwencji bez oraz z jego zastosowaniem. Przykładowo, dla długości sekwencji = 2 wartości, w drugim przypadku predykcje były znacznie bardziej zbliżone do danych rzeczywistych (Rysunek 24)



Rysunek 24: Porównanie wyników bez dostrajania (wykres górnny) oraz z zastosowaniem dostrajania (wykres dolny) dla długości sekwencji = 2 wartości. Użyty model to LSTMForecaster. Dla przypadku z dostrajaniem predykcje rzeczywistych wartości pulsu (kolor pomarańczowy) były znacznie bardziej zbliżone do danych rzeczywistych (kolor niebieski).

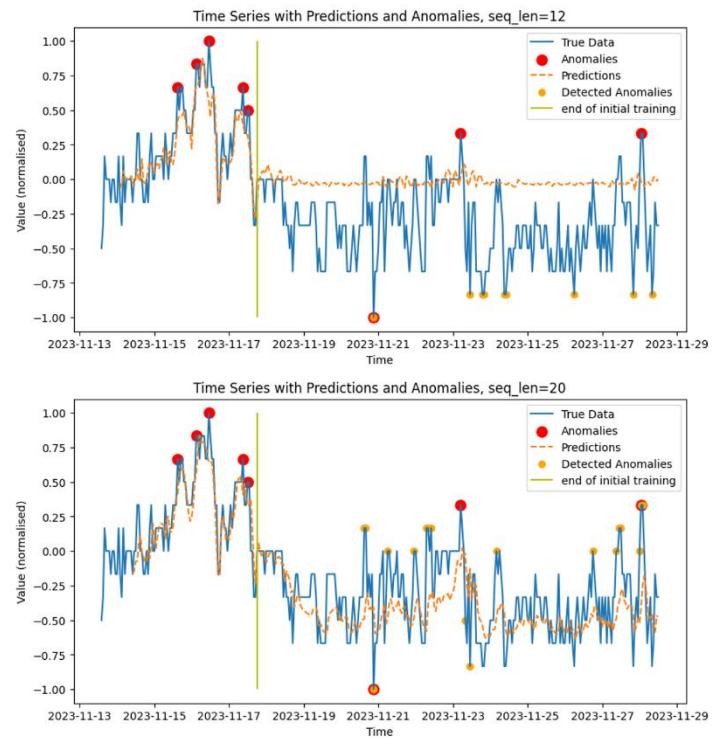
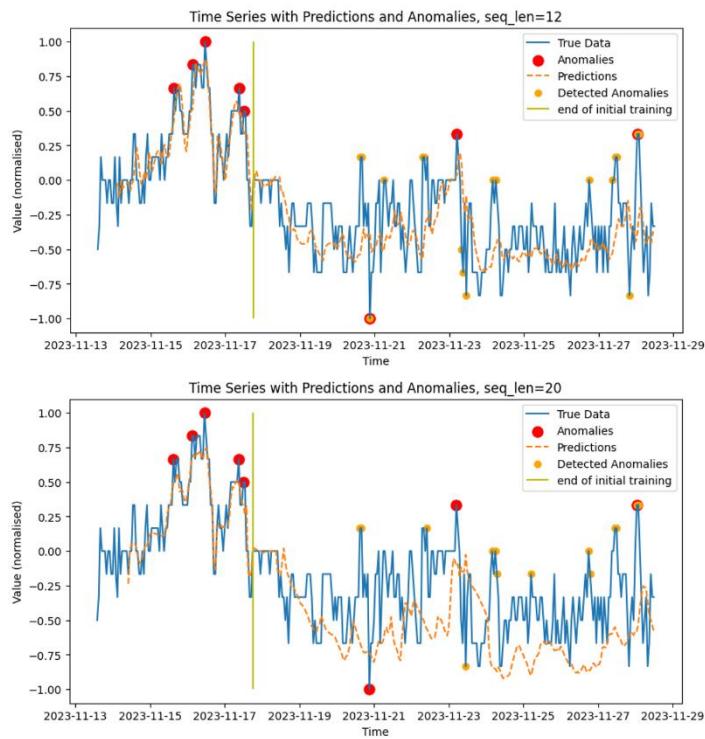
4.1.2 Porównanie modeli

Patrząc na wykresy zamieszone w poprzednim punkcie różnice w jakości predykcji wykonywanych przez oba modele nie są bardzo wyraźne. Na podstawie wizualnego porównania rezultatów uzyskanych dla poszczególnych długości sekwencji wniosuję, że w przypadku danych tego konkretnego pacjenta dla dłuższych sekwencji pierwszy model (LSTMForecaster) lepiej niż drugi (LSTMAttentionForecaster) przewidywał kształt danych rzeczywistych. Na Rysunku 25 przedstawiam porównanie. Przewidywane wartości zaznaczone są linią pomarańcową, rzeczywiste niebieską:

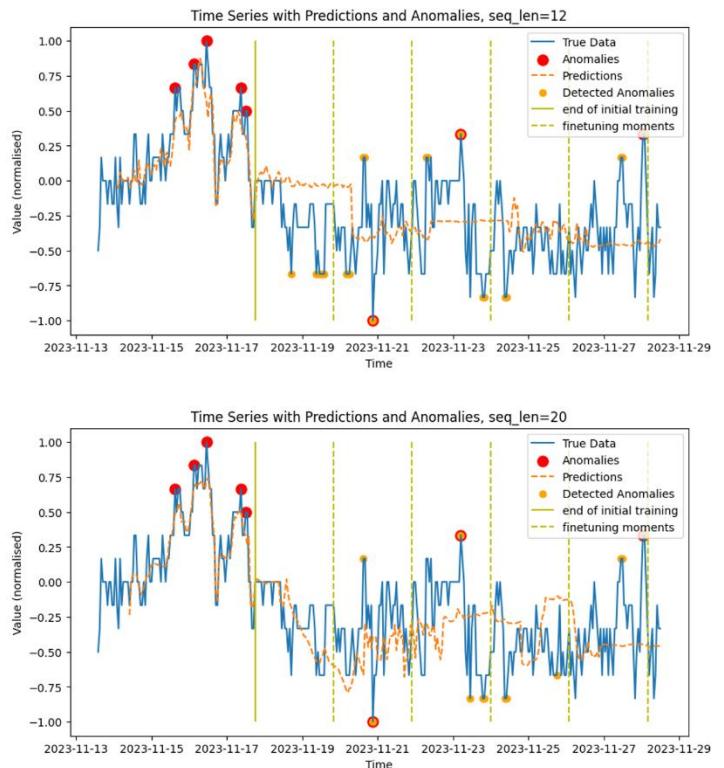
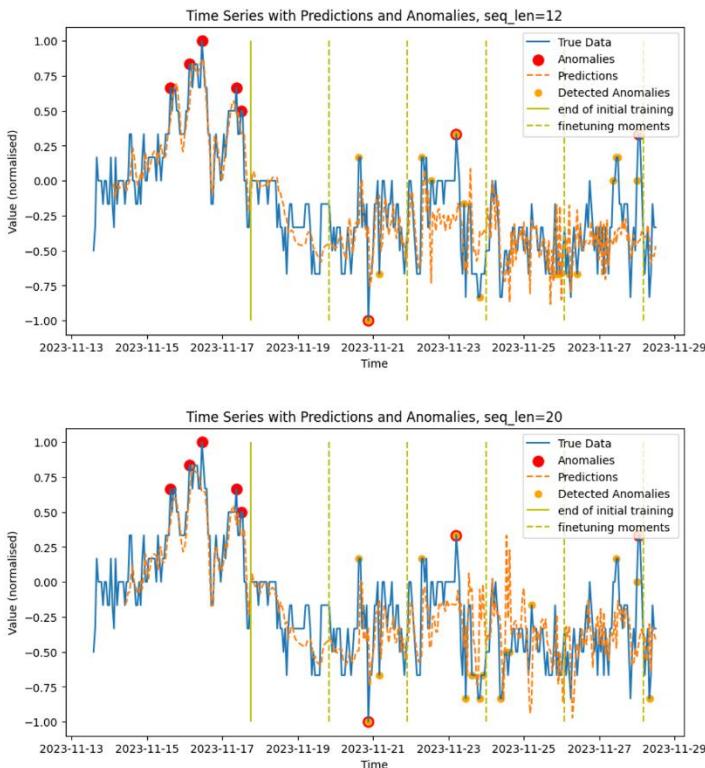
LSTMForecaster

LSTMAttentionForecaster

bez dostrajania:



z dostrajaniem:



Rysunek 25: Porównanie predykcji rzeczywistych wartości pulsu wykonywanych przez oba modele dla dłuższych sekwencji (tj. 12 i 20 wartości). Po lewej przedstawione są wyniki dla modelu LSTMForecaster, po prawej wyniki dla modelu LSTMAttentionForecaster uzyskane przy tych samych wartościach parametrów. Góra cześć Rysunku przedstawia wyniki dla przypadku bez zastosowania dostrajania, dolna z jego zastosowaniem.

4.1.3 Porównanie długości sekwencji

Porównując otrzymane przebiegi wydaje się, że dla ostatecznego przypadku z zastosowaniem dostrajania lepsze odwzorowanie rzeczywistego przebiegu danych połączone w lepszym wykrywaniem anomalii uzyskiwane było dla krótszych sekwencji (2, 3, 5)

4.2 Optymalizacja

Parametry które, poza długością sekwencji, mogłyby zostać zoptymalizowane to np.:

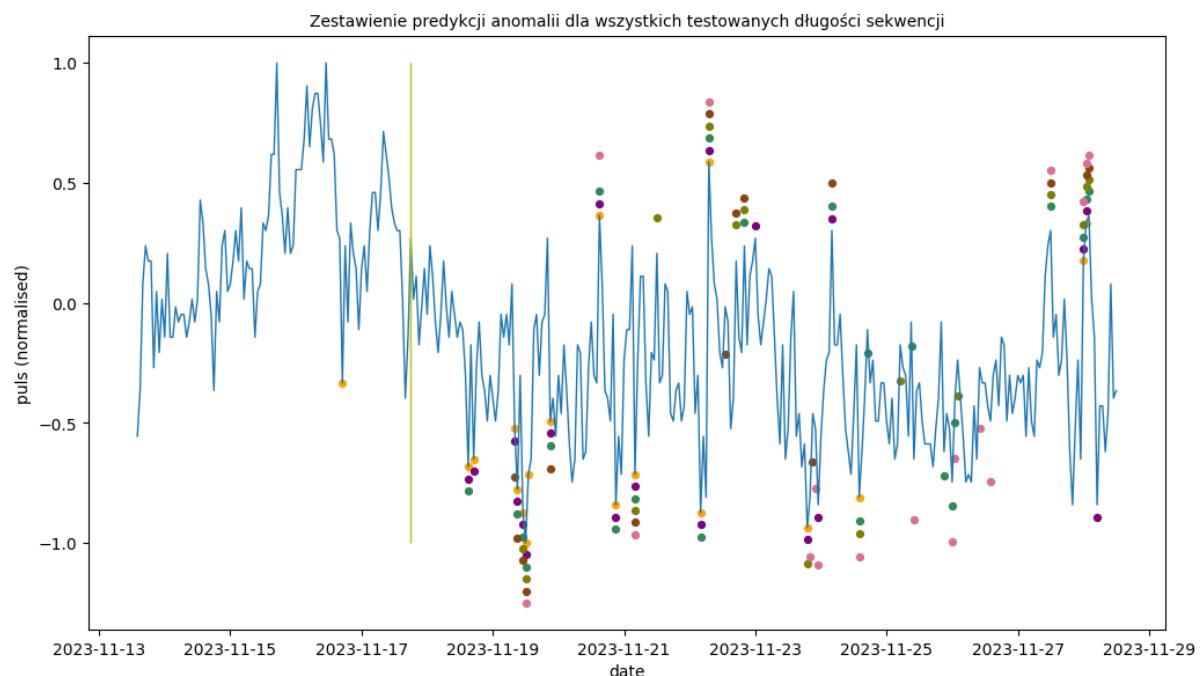
- współczynnik uczenia optymalizatora,
- próg wykrywania anomalii (różnica wartości przewidywanej i rzeczywistej która skutkowała wskazaniem anomalii)
- *finetuning_step* (liczba kroków czasowych po których wykonywany był dostrajanie)

Celem optymalizacji byłaby maksymalizacja specyficzności oraz czułości wykrywania anomalii, przy czułości uznawanej z nadzczną (jako, że jest bardziej wskazane wskazać alert niepotrzebnie niż pominąć stan zagrażający życiu). Warto jednak zwrócić uwagę, że ostatecznie posiadałam dane pochodzące tylko od jednego pacjenta co zgodnie z moją wiedzą jest zbyt mało aby przeprowadzić istotną optymalizację parametrów. Nie jest to także wystarczająco aby porównać sposób przewidywania prawidłowego przebiegu danych dla przypadku danych generowanych i danych rzeczywistych.

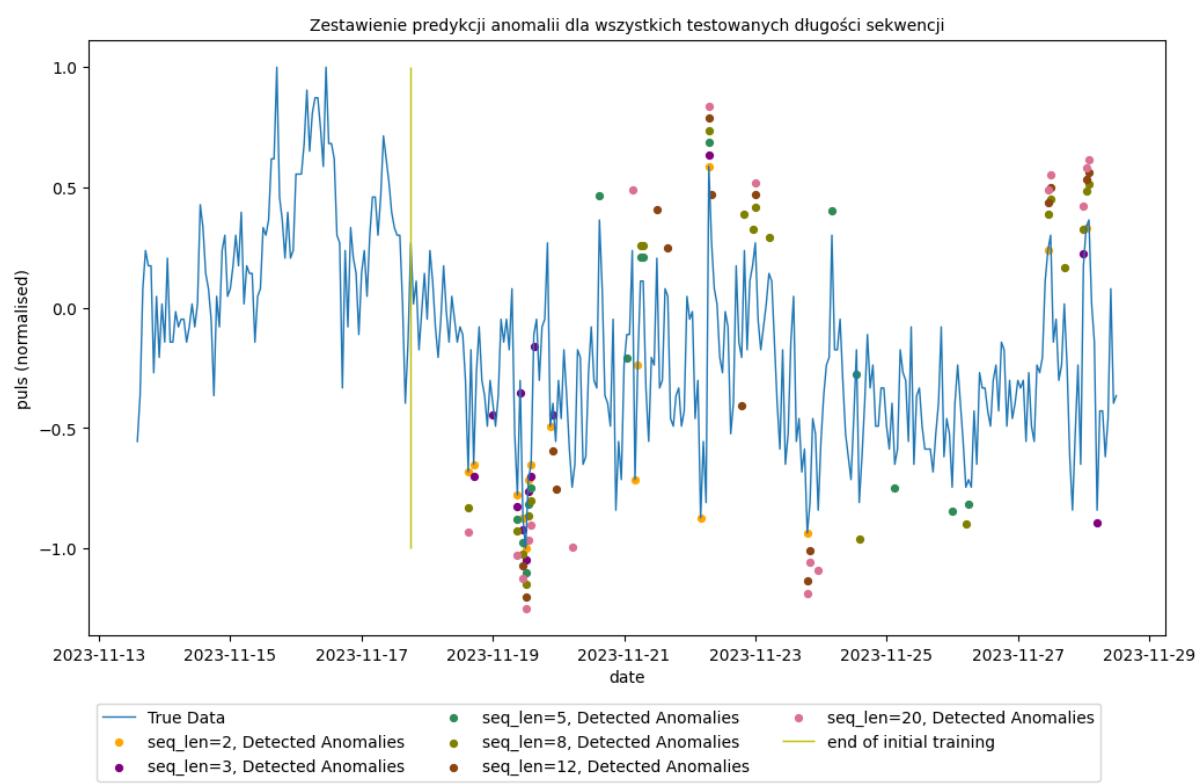
4.3 Dane generowane

Poniżej przedstawiam jak wskazywane były anomalie dla przykładu danych generowanych sztucznie metodą opisaną wcześniej w sekcji „Augmentacja danych”. Widoczna jest powtarzalność lokalizacji anomalii wskazywanych przez modele dla różnych długości sekwencji.

LSTMForecaster, series_aug, finetuning



LSTMAttentionForecaster, series_aug, finetuning



Rysunek 26: Porównanie predykcji anomalii z zastosowaniem dostrajania dla danych generowanych sztucznie

5 ZAKOŃCZENIE

5.1 PODSUMOWANIE

W niniejszej pracy opracowałem surowe dane medyczne pochodzące z Oddziału Intensywnej Terapii Kardiologicznej. Obróbka danych doprowadziła do utworzenia zestawu danych przystosowanego do zastosowania w technikach uczenia maszynowego. Uzyskane dane wykorzystałem do przeprowadzenia symulacji wykrywania stanów anomalnych u pacjenta medycznego. W eksperymencie wykorzystałem Rekurencyjne Sieci Neuronowe typu LSTM. Udało mi się uzyskać poprawę wyników stosując dostrajanie (po jego zastosowaniu zdolność sieci do wskazywania rzeczywistych anomalii znacznie wzrosła). Została wykonana praca łącząca wiedzę środowiska pielęgniarskiego oraz wiedzę dotyczącą obróbki oraz zastosowania danych. Wynikiem pracy nie jest system zdolny w sposób wiarygodny przewidywać stany anomalne u pacjentów ww. oddziału OIOM. Zapewnia ona jednak przekrój kolejnych etapów obróbki surowych danych oraz przykład zastosowania danych medycznych w technikach uczenia maszynowego. Powyższa praca może służyć jako przykład podejścia do wykorzystania zaawansowanych technik analizy danych do celów medycznych i stanowić podstawę do dalszej pracy z analogicznymi przykładami danych.

5.2 GŁÓWNE NARZĘDZIA

- język programowania Python (ważniejsze biblioteki: torch, numpy, pandas, matplotlib.pyplot)
- środowisko JupyterLab

5.3 Konsultacja Medyczna

[K.M] Konsultantka, źródło wiedzy medycznej oraz informacji na temat oddziału, z którego pochodziły dane:

Anna Szczelina, magister pielęgniarstwa, specjalista pielęgniarstwa pediatrycznego

6 Bibliografia

- [1] Abulkhair, Ahmed. "Data Imputation Demystified | Time Series Data (section: Seasonal Trend Decomposition using Loess (STL) Imputation)." (2023). <<https://medium.com/@aaabulkhair/data-imputation-demystified-time-series-data-69bc9c798cb7>>.
- [2] Data-Science-Wizards. *Preprocessing and Data Exploration for Time Series — Handling Missing Values*. 2023. <<https://medium.com/@datasciencewizards/preprocessing-and-data-exploration-for-time-series-handling-missing-values-e5c507f6c71c>>.
- [3] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, 2023.
- [4] Kosowski, Piotr. "Introduction to Data Analysis with R." *Exploratory Data Analysis, Data Cleaning and Visualisation*. AGH University Course, 2024.
- [5] *Matplotlib 3.10.0 documentation*. 2024. <<https://matplotlib.org/stable/>>.
- [6] *PyTorch documentation*. 2024. <<https://pytorch.org/docs/stable/index.html>>.
- [7] Schafer, Corey. *Matplotlib Tutorials*. n.d. <https://youtube.com/playlist?list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB_&si=k5lFE1VepzgQtU65>.
- [8] —. *Pandas Tutorials*. n.d. <<https://youtube.com/playlist?list=PL-osiE80TeTsWmV9i9c58mdDCSskIFdDS&si=wOilnsitsMXVD3HK>>.
- [9] Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-Learn*. Packt Publishing, 2022.
- [10] Starmer, Josh. *Neural Networks / Deep Learning*. n.d. <https://youtube.com/playlist?list=PLblh5JKOoLUlxGDQs4LFFD--41Vzf-ME1&si=M6Y_slr3DI5vd5I_>.