Rozpoznawanie awarii urządzenia

Michał Kacprzak
Magdalena Górska
Dariusz Biela

8 czerwca 2022

Spis treści

1 Opis projektu **2**

2 Kod źródłowy **2**

3 Podział pracy **2**

4 Wykorzystane technologie **2**

5 Sposób uruchomienia programu **3**
5.1 Wymagania 3
5.2 Instrukcja uruchomienia 3

6 Analiza projektu **4**
6.1 Struktura projektu 4
6.2 Zbiór danych 4
6.3 Statystyczna analiza danych 5
6.4 Sztuczne sieci neuronowe 12

7 Podsumowanie **26**

1 Opis projektu

Projekt zakłada dokonanie analizy możliwości sztucznych sieci neuronowych typu LSTM (ang. Long short-term memory) do rozpoznawania awarii urządzenia.

2 Kod źródłowy

Kod źródłowy projektu został udostępniony jako repozytorium na Githubie i jest dostępny pod poniższym linkiem:
`https://github.com/Darkosz1012/artificial_neural_network_project`

3 Podział pracy

Podział pracy między członków zespołów przedstawia się następująco:

- Michał Kacprzak - statystyczna analiza danych na całym zbiorze danych, stworzenie dokumentacji projektu;
- Dariusz Biela - przygotowanie i przeprowadzenie analizy możliwości sztucznych sieci neuronowych typu LSTM i GRU w wykrywaniu awarii urządzenia;
- Magdalena Górska - statystyczna analiza danych na całym zbiorze danych z uwzględnieniem podziału na poszczególne czasy pomiarów, przygotowanie materiałów graficznych użytych w dokumentacji projektu;

4 Wykorzystane technologie

Projekt został napisany w całości w Python 3 a dokładniej w wersji 3.7, więc oczywiście taka wersja języka Python jest wymagana do uruchomienia programu. Dodatkowo zostały wykorzystane następujące biblioteki:

- tenserflow - biblioteka została użyta w celu stworzenia modeli sztucznych sieci neuronowych typu LSTM;
- scikit-learn - została użyta ze względu na wbudowany moduł pozwalający w trywialny sposób przeprowadzić cross-walidacje danych;
- pandas - biblioteka została użyta w procesie wczytywania danych oraz do statycznej analizy danych. Została wybrana ze względu na mnogość wbudowanych funkcji oraz doskonałą dokumentacje
- numpy - podstawowa biblioteka w przypadku pracy z większą ilością danych liczbowych
- matplotlib oraz seaborn - biblioteki odpowiadająca za wizualizacje statystycznej analizy danych oraz rezultatów uczenia sztucznej sieci neuronowej;
- notebook - biblioteka jest wymagana w przypadku korzystania z środowiska Jupyter Notebook

5 Sposób uruchomienia programu

5.1 Wymagania

Do uruchomienia projektu wymagane jest posiadanie zainstalowanych następujących narzędzi:

- język Python wersja 3.7

5.2 Instrukcja uruchomienia

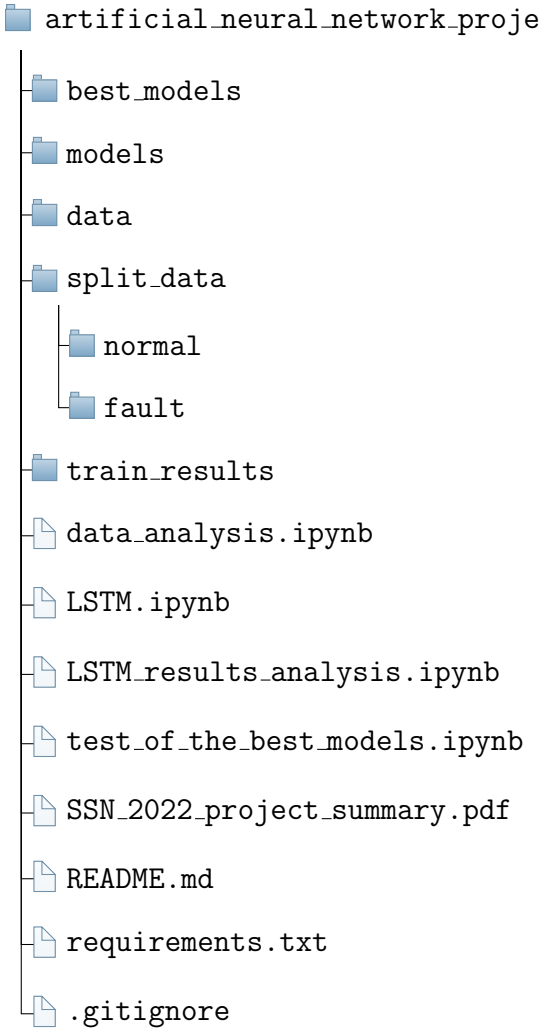
Najprostszym sposobem na uruchomienie projektu jest sklonowanie repozytorium. Następnie należy zainstalować odpowiednie biblioteki, za pomocą pliku *requirements.txt*. Ostatnim etapem jest uruchomienie serwera Jupyter Notebook. Poniższe komendy pozwalają na zrealizowanie wyżej opisanych kroków.

```
$ git clone https://github.com/Darkosz1012/artificial_neural_network_project.git
$ cd artificial_neural_network_project/
$ pip install -r requirements.txt
$ jupyter notebook
```

Uruchomienie serwera spowoduje otworzenie nowego okna lub karty przeglądarki internetowej. Po wykonaniu powyższych komend użytkownik zyskuje możliwość swobodnego uruchamiania plików z rozszerzeniem IPYNB.

6 Analiza projektu

6.1 Struktura projektu



6.2 Zbiór danych

Zbiór danych wykorzystanych do analizy oraz treningu został przekazany w postaci plików CSV. Pliki z wyrazem „fault" w nazwie zawierają dane dla których doszło do awarii urządzenia(w dalszej części sprawozdania będą one określane jako „błędne"). Natomiast jeśli nazwa pliku zawiera człon „normal" znaczy to, że praca przebiegła bez przeszkód, w dalszej części sprawozdania będą one określane jako „poprawne". Dane znajdują się one w dwóch folderach - `data/` oraz `split_data/`. W subfolderach `split_data/normal/` oraz `split_data/fault/` dane zostały pogrupowane ze względu na awarie lub jej brak. Każdy plik CSV wygląda tak samo. Mianowicie pierwsza kolumna to czas liczony w minutach. Natomiast pozostałe kolumny stanowią wartości pomiarowe. W celu ułatwienia analizy i wizualizacji danych pierwsza kolumna została oznaczona nazwą „TIME" natomiast pozostałe zostały nazwane kolejnymi literami alfabetu angielskiego (1).

TIME	A	B	C	D	E	F	G	H	I
1	29.9	14	21.6	24	660	6.3	66.8	38.1	16.4
2	30.2	14	21.6	24	600	5.9	69.3	38.2	15.6
3	30.5	14	21.6	24	300	5.6	68	36.3	15
4	30.6	14	21.6	24	60	5.7	58.1	35.2	14.9
5	30.4	14	21.6	24	120	5.9	49.7	34.9	15.4

Tablica 1: 5 przykładowych elementów z zbioru danych

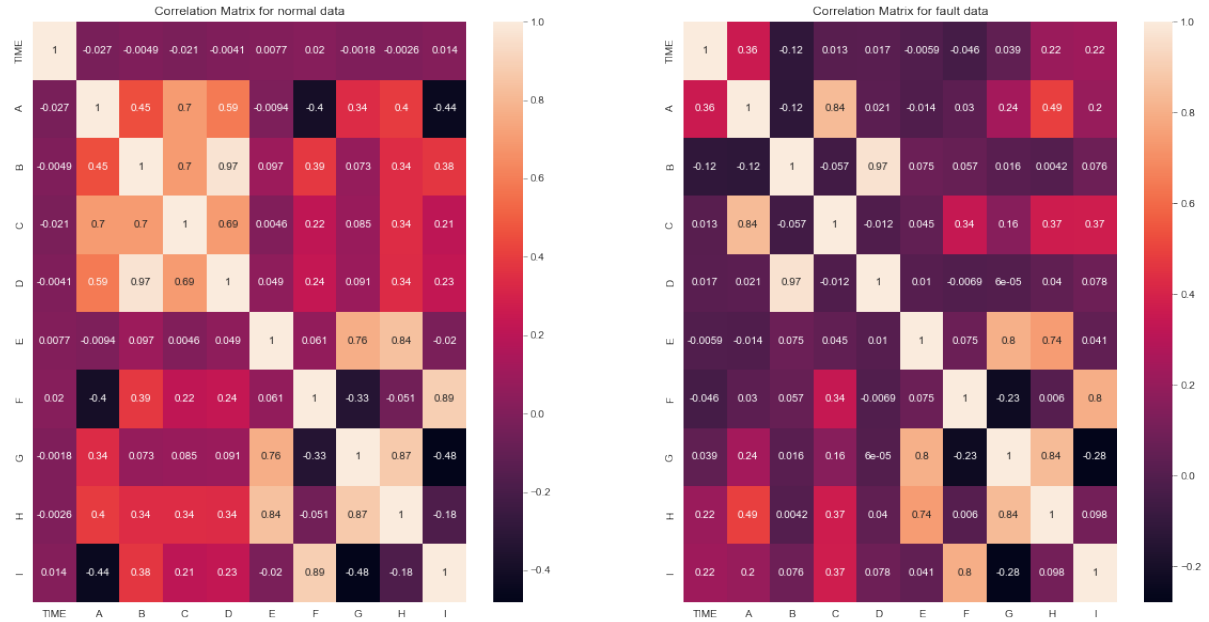
6.3 Statystyczna analiza danych

Statystyczna analiza danych została przeprowadzona na dwa sposoby:

- Dane były podzielone na dwie grupy, ze względu na fakt awarii lub jej braku
- Dane były podzielone tak jak w przypadku pierwszego sposobu ale dodatkowo były pogrupowane w 31 podgrupy ze względu na czas pomiaru

Analiza danych została przeprowadzona w pliku `data_analysis.ipynb`.

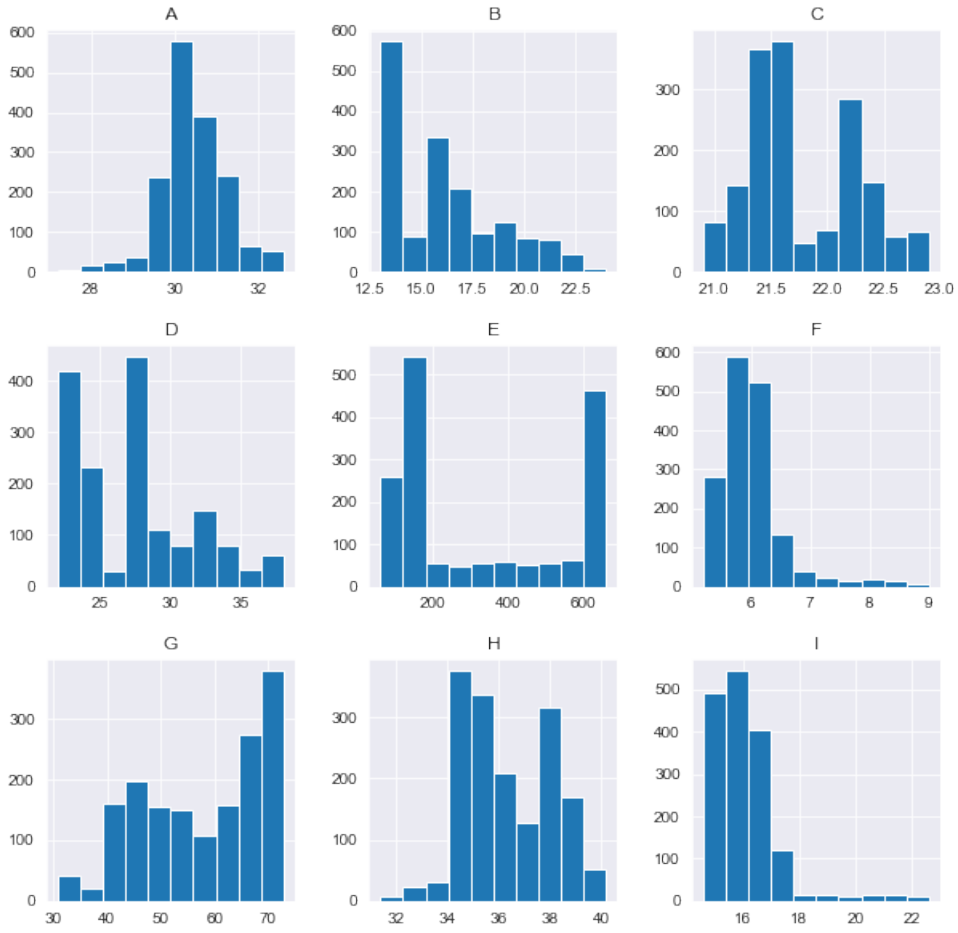
Pierwszym etapem było stworzenie macierzy korelacji dla danych poprawnych jak i błędnych (tab. 1). Macierze zostały stworzone za pomocą wbudowanej metody `corr` dla obiektu `Dataframe`, czyli najbardziej podstawowego obiektu, który przechowuje dane z biblioteki *Pandas*. Wyraźnie widoczne są znaczne różnice w korelacjach między poszczególnymi zmiennymi dla danych poprawnych oraz błędnych.



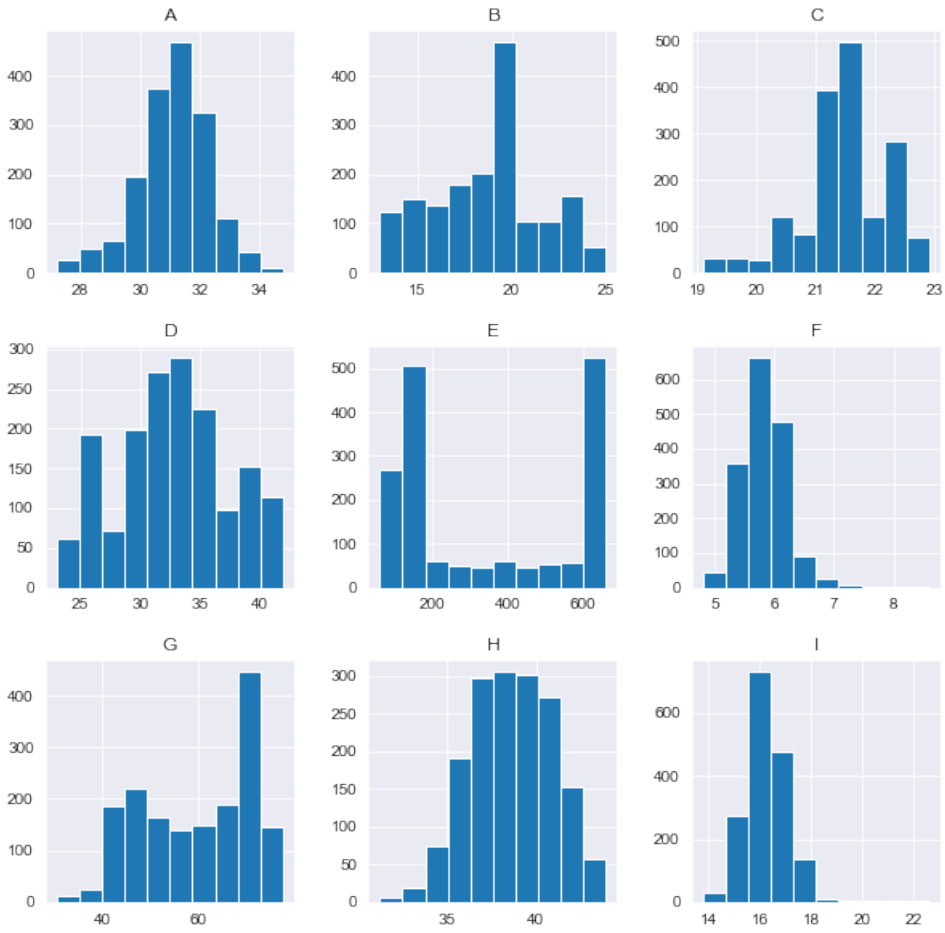
Rysunek 1: Macierze korelacji dla danych poprawnych oraz błędnych

Kolejnym etapem było przeanalizowanie histogramów poszczególnych zmiennych, z wyłączeniem czasu(rys. 2 i rys. 3. Histogramy zostały stworzone za pomocą wbudowanej metody `hist` dla obiektu `Dataframe`. Analizując powstałe histogramy można zauważyć że widoczne zmiany w rozkładzie wartości kolumn są bardzo różne. Dla niektórych kolumn np. A, E zmiany są bardzo małe, natomiast dla kolumn B, C, D różnice są drastyczne. Pozwala to

wysnuć hipotezę, że niektóre zmienne będą bardziej istotne od innych w procesie wykrywania awarii urządzenia.

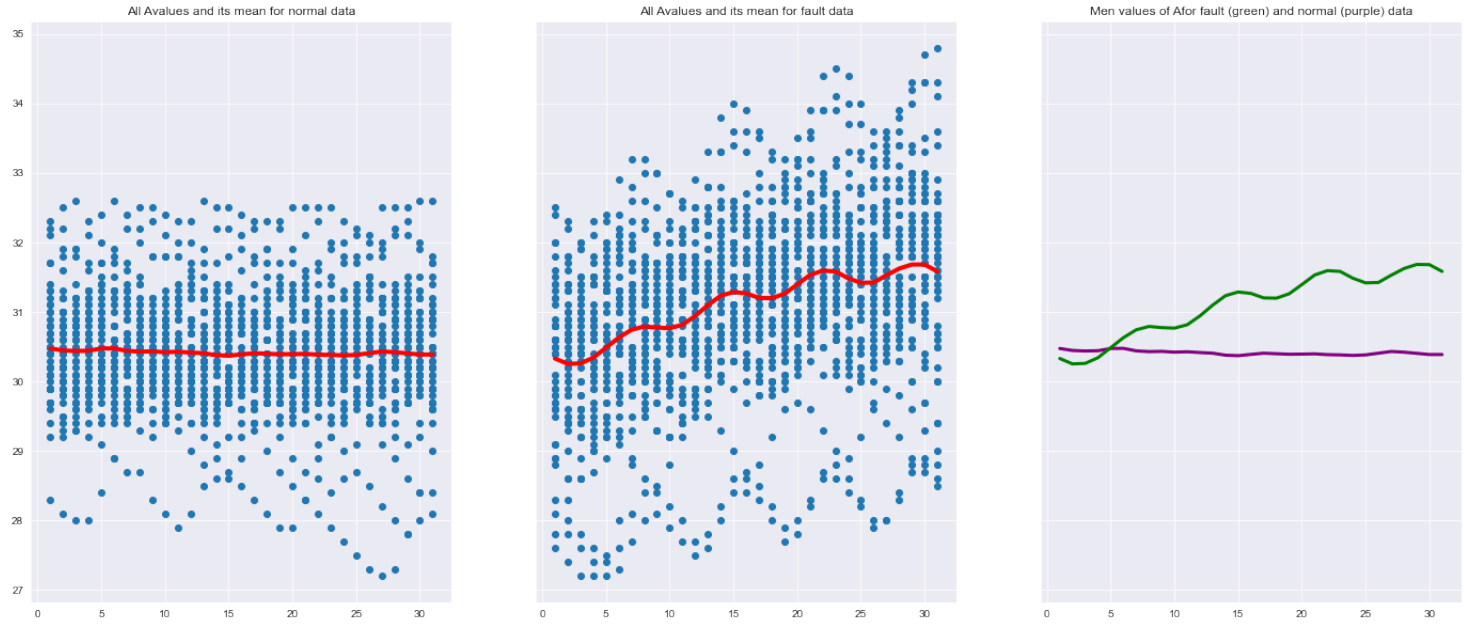


Rysunek 2: Histogramy wszystkich zmiennych dla danych poprawnych



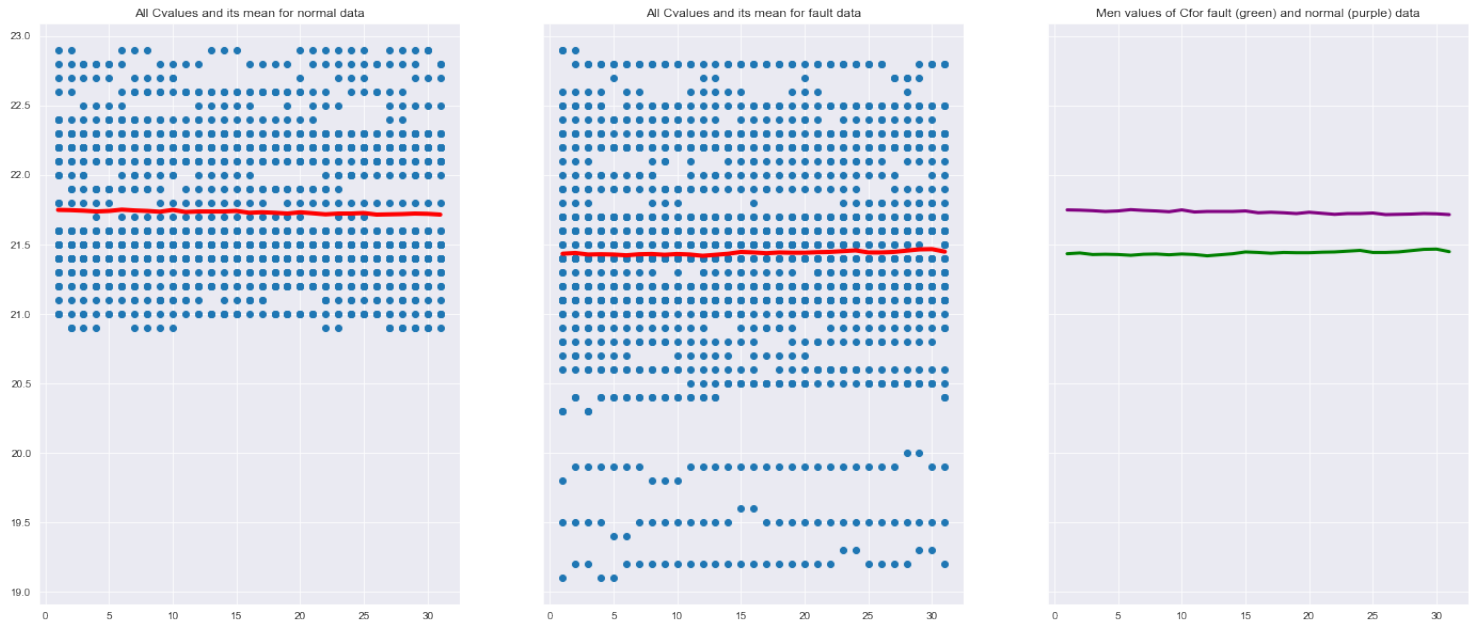
Rysunek 3: Histogramy wszystkich zmiennych dla danych fałszywych

W pliku `data_analysis.ipynb` po uruchomieniu komórek 9 oraz 10 zostaną wyświetlone tabele, które zawierają podstawowe parametry statystyczne takie jak średnia, odchylenie standardowe, wartość minimalna i maksymalne dla poszczególnych kolumn. Zgodnie z przewidywaniami wartości te są różne w praktycznie każdym przypadku dla danych poprawnych oraz błędnych. Drugim etapem statystycznej analizy danych była analiza danych podzielonych dodatkowo na 31 podgrup ze względu na czas pomiaru. Uzyskane wyniki prezentują się bardzo ciekawie. Każda zmienna była analizowana oddzielnie. Na wykresach (od rys. 4 do rys. 12) można zobaczyć zbiorcze wykresy wartości poszczególnych kolumn dla danych poprawnych jak i błędnych wraz z porównaniem średnich wartości w danych minutach. Na uzyskanych wykresach można zauważyć że naniesione wartości tworzą pewne wzorce, trendy. Na uwagę zasługuje np. rys. 4. Dotyczy on analizy wartości kolumny A. Wartości tej zmiennej dla danych poprawnych są skupione wokół jednej wartości - 30.5, bez względu na minutę pomiaru. Natomiast wartości błędnych wartości kolumny charakteryzują się znacznym trendem wzrostowym co wyraźnie widać przy analizie porównawczej średnich wartości dla poszczególnych minut dla danych poprawnych i błędnych. Jednakże znaczne różnice widoczne na wykresach przedstawiających wartości dla poszczególnych minut nie koniecznie muszą świadczyć o znacznej zmianie średniej wartości. Przykład takiej iluzorycznej różnicy znajduje się na rys. 6. Punkty na wykresie dla danych błędnych są znacznie bardziej rozproszone jednakże średnie wartości dla poszczególnych minut są bardzo podobne do tych uzyskanych dla danych poprawnych. Jest to spowodowane faktem, że niektóre punkty na wykresie nakładają się na siebie co nie zostało uwzględnione w wizualizacji.



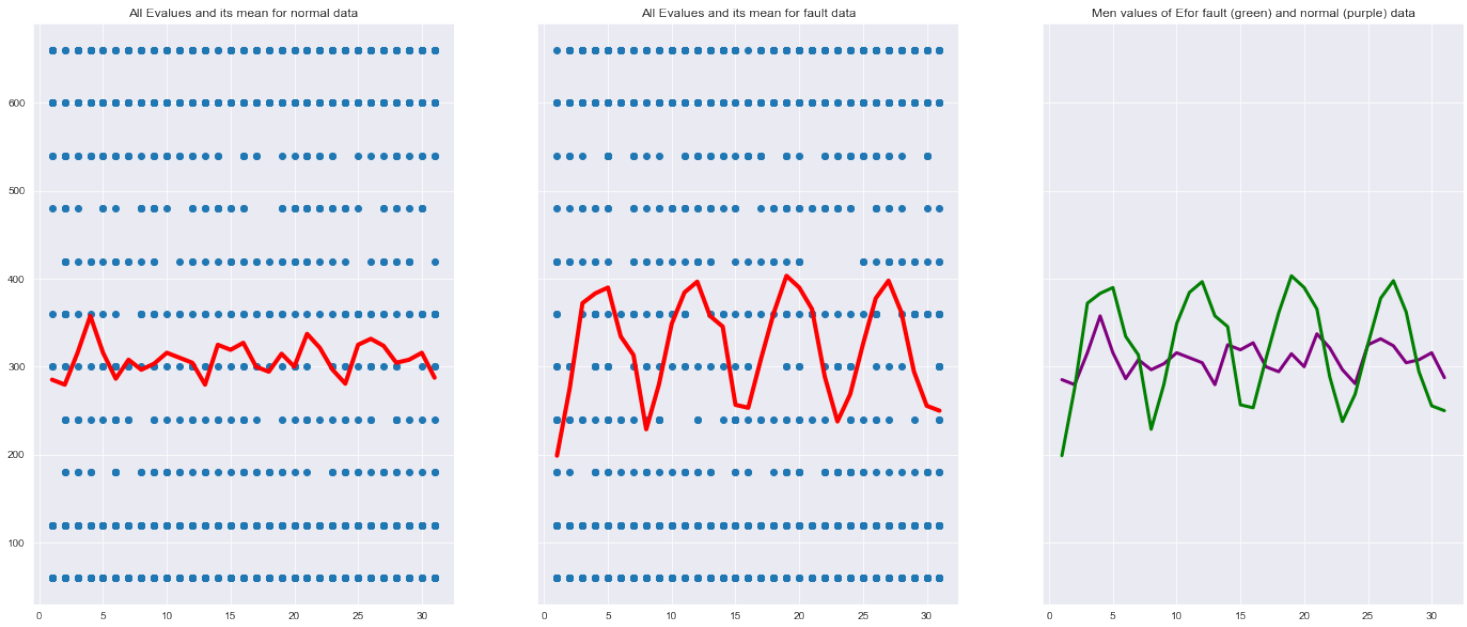
Rysunek 4: Zbiorczy wykres wartości kolumny A dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut

Rysunek 5: Zbiorczy wykres wartości kolumny B dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut

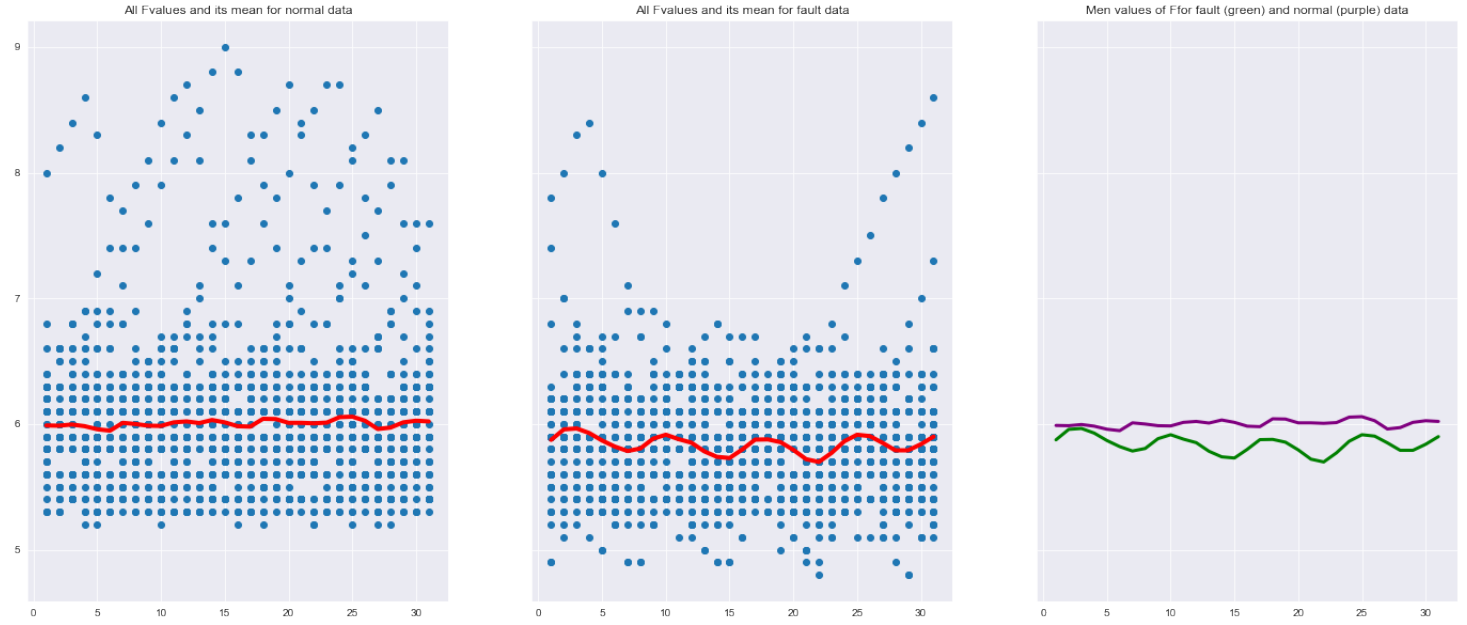


Rysunek 6: Zbiorczy wykres wartości kolumny C dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut

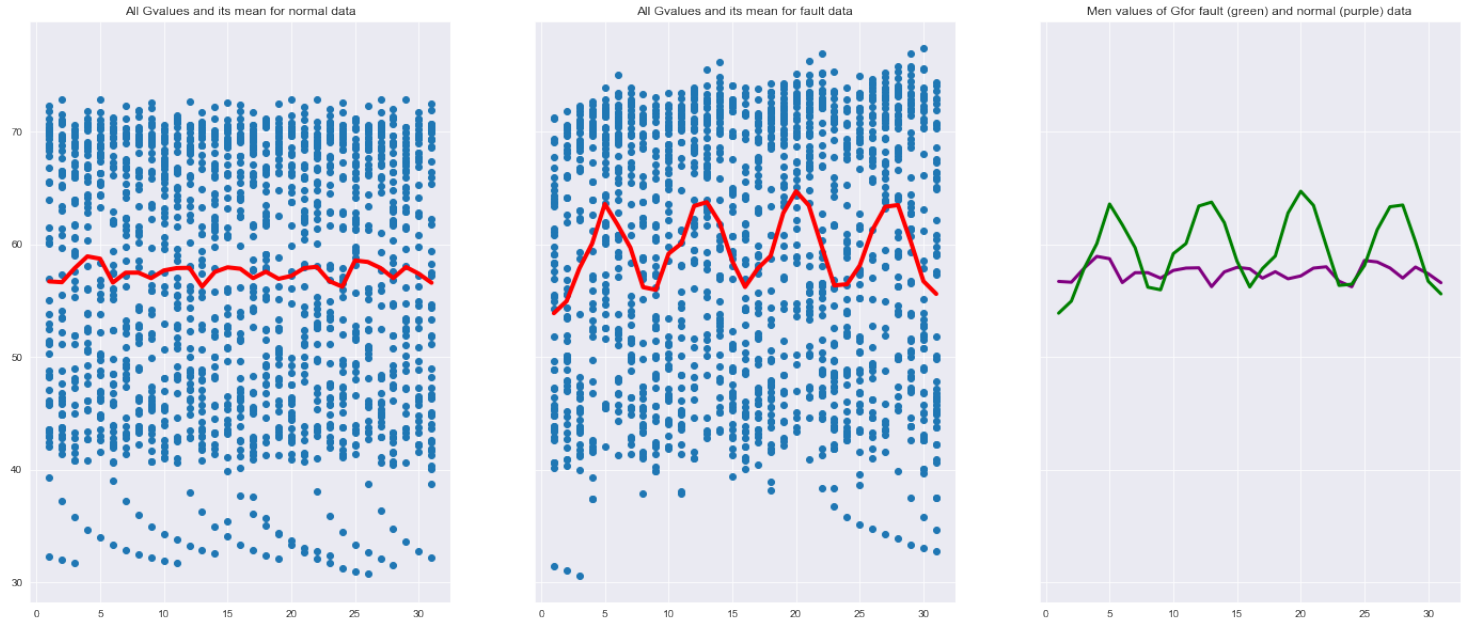
Rysunek 7: Zbiorczy wykres wartości kolumny D dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut



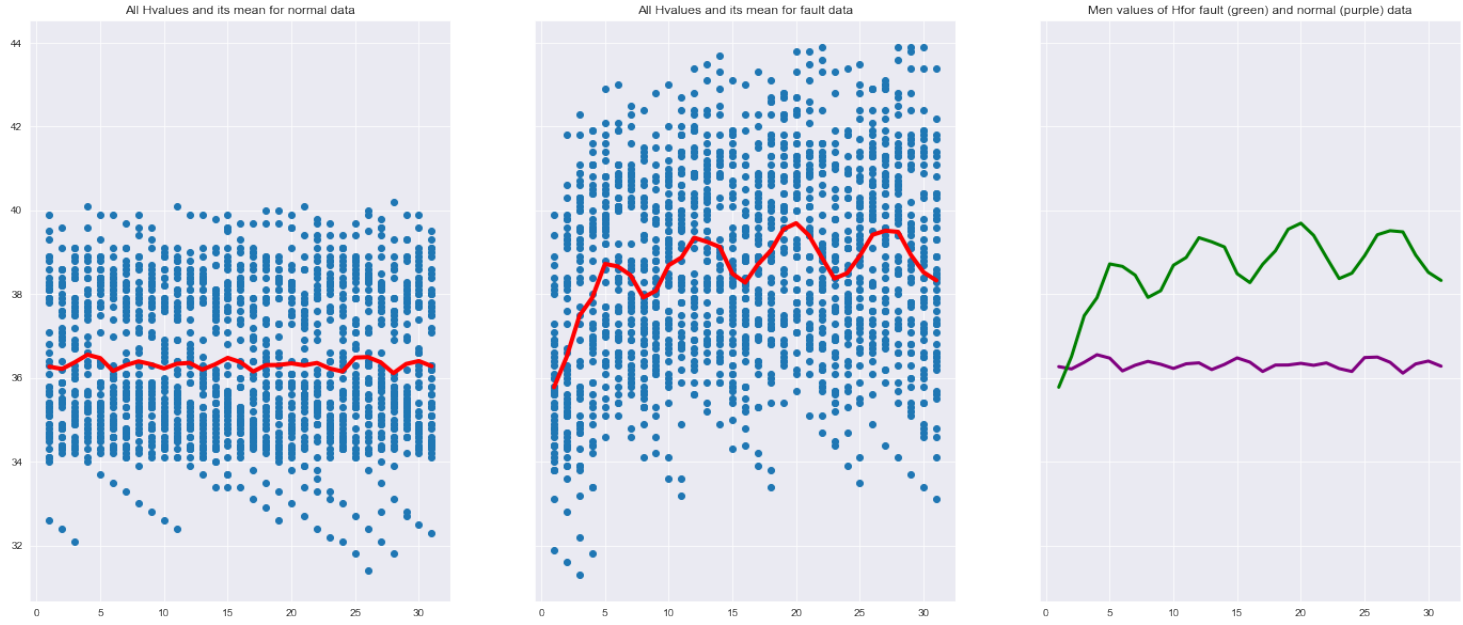
Rysunek 8: Zbiorczy wykres wartości kolumny E dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut



Rysunek 9: Zbiorczy wykres wartości kolumny F dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut



Rysunek 10: Zbiorczy wykres wartości kolumny G dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut



Rysunek 11: Zbiorczy wykres wartości kolumny H dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut

Rysunek 12: Zbiorczy wykres wartości kolumny I dla poszczególnych minut dla danych poprawnych jak i błędnych oraz porównanie średnich wartości dla poszczególnych minut

6.4 Sztuczne sieci neuronowe

6.4.1 Architektura

W ramach naszego projektu zaimplementowaliśmy cztery różne architektury sieci neuronowych:

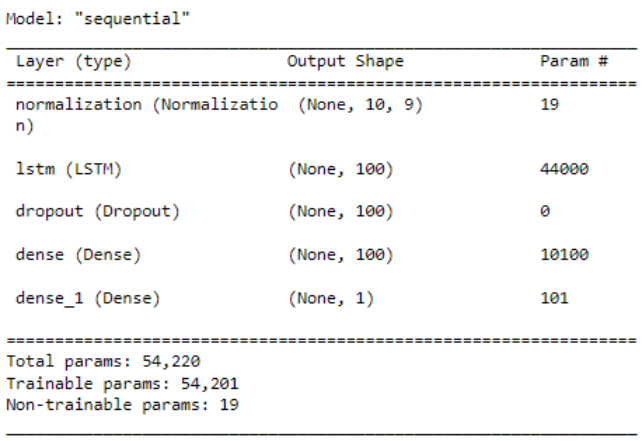
- prostą jednowarstwową sieć LSTM (nr 0)
- wielowarstwową sieć LSTM (nr 1)
- prostą jednowarstwową sieć GRU (nr 2)
- wielowarstwową sieć GRU (nr 3)

6.4.1.1 Cechy wspólne wszystkich architektur sieci

Pierwszą warstwą każdej z nich jest warstwa normalizacyjna, która jest przed uczeniem sieci dopasowywana do całego zbioru danych. Ostatnią warstwą każdego modelu jest warstwa gęsta o rozmiarze 1. Dodatkowo sieć nr 0 i 2 oraz 1 i 3 mają dokładnie tą samą strukturę i różnią się tylko zmianą typu warstw z LSTM na GRU.

6.4.1.2 Jednowarstwowa sieć LSTM nr 0

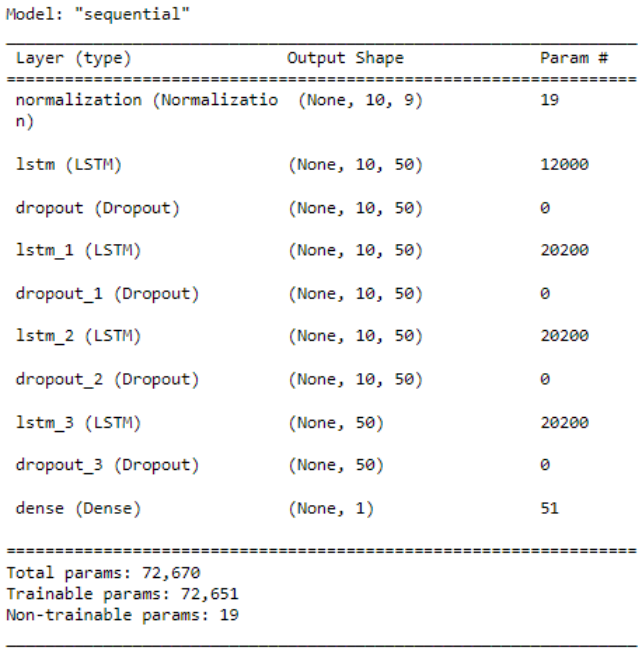
Model nr 0 jest to pierwsza stworzona przez nas architektura, na której testowaliśmy wstępne możliwość LSTM. Architektura tego modelu składa się z pięciu warstw przedstawionych na rysunku nr 13. Poza warstwą normalizacyjną model posiada jedną warstwę LSTM o rozmiarze 100, po której podczas uczenia występuje 50% dropout. W celu przyspieszenia uczenia oraz poprawienia wyników dodaliśmy, również warstwę gęstą o rozmiarze 100.



Rysunek 13: Podsumowanie architektury modelu nr 0.

6.4.1.3 Wielowarstwowa sieć LSTM nr 1

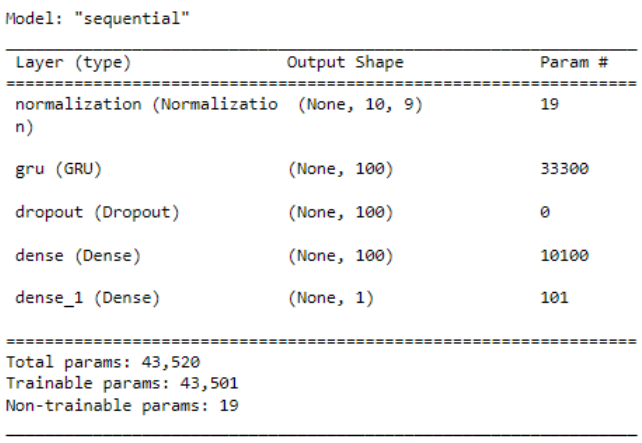
Model nr 1 jest naszą kolejną architekturą opartą o warstwy LSTM. Architektura tego modelu jest przedstawiona na rysunku nr 14. Jego struktura jest stworzona na podstawie 4 warstw LSTM o rozmiarze 50 z dropout 40%. Ponadto wszystkie warstwy LSTM poza ostatnią zwracają sekwencję, żeby zapewnić poprawne działanie następnej warstwy rekurencyjnej.



Rysunek 14: Podsumowanie architektury modelu nr 1.

6.4.1.4 Jednowarstwowa sieć GRU nr 2

Model nr 2 jest architekturą opartą o warstwę GRU. Architektura tego modelu jest przedstawiona na rysunku nr 15. Jego struktura jest stworzona na podstawie modelu nr 0 z zamienioną warstwą LSTM na GRU o dokładnie tym samym rozmiarze.

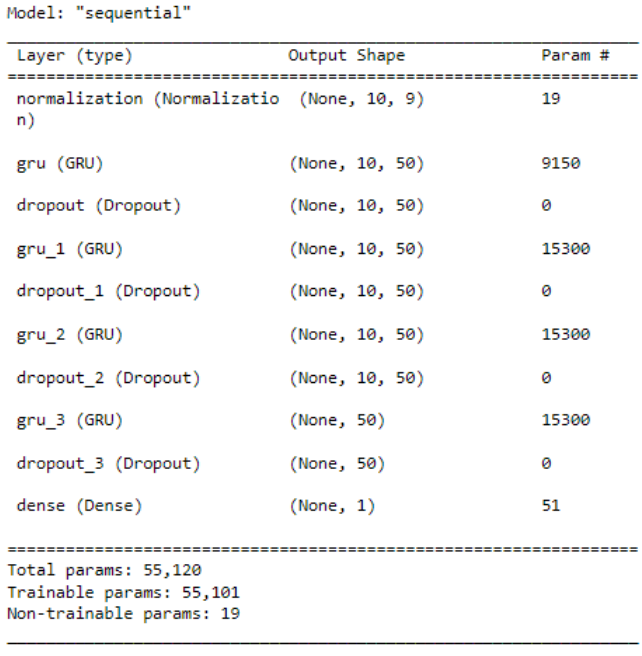


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizatio  (None, 10, 9)            19
 n)

 gru (GRU)                   (None, 100)               33300

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 43,520
Trainable params: 43,501
Non-trainable params: 19
_________________________________________________________________
```

Rysunek 15: Podsumowanie architektury modelu nr 2.

6.4.1.5 Wielowarstwowa sieć GRU nr 3

Model nr 3 jest kolejną architekturą opartą o warstwę GRU. Architektura tego modelu jest przedstawiona na rysunku nr 16. Jego struktura jest stworzona z wykorzystaniem modelu nr 1. Z zamienionymi warstwami LSTM na GRU.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizatio  (None, 10, 9)            19
 n)

 gru (GRU)                   (None, 10, 50)            9150

 dropout (Dropout)           (None, 10, 50)            0

 gru_1 (GRU)                 (None, 10, 50)            15300

 dropout_1 (Dropout)         (None, 10, 50)            0

 gru_2 (GRU)                 (None, 10, 50)            15300

 dropout_2 (Dropout)         (None, 10, 50)            0

 gru_3 (GRU)                 (None, 50)                15300

 dropout_3 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 55,120
Trainable params: 55,101
Non-trainable params: 19
_________________________________________________________________
```

Rysunek 16: Podsumowanie architektury modelu nr 3.

6.4.2 Przygotowanie danych

Pliki z danymi podzieliśmy na dwa foldery normal i fault, żeby ułatwić pobieranie danych dla konkretnej klasy.

Następnie utworzyliśmy funkcję do wczytywania zbioru danych i tworzenia dla niego etykiet. Funkcja ta pozwala również na wybór rozmiaru serii, czyli układa dane w szereg czasowy o podanej długości.

6.4.3 Trening

W celu sprawdzenia, która z powyższych sieci jest najlepsza, zastosowaliśmy hiperparametryzację dwóch parametrów:

- numer modelu - [0, 1, 2, 3]
- długość serii danych - [2, 5, 10, 31]

Natomiast reszta parametrów została dobrana manualnie poprzez kilkukrotne uczenie modelu nr 0 z przyjętą serią 5:

- epochs = 30
- batch size = 32
- optimizer = Adam
- learning rate = 0.0005
- funkcja strat - binary crossentropy
- normalizacja danych wejściowych za pomocą warstwy normalizacyjnej
- funkcja aktywacji wyjścia - sigmoid

Ponadto, żeby uzyskać dokładniejsze wyniki z uczenia sieci zastosowaliśmy cross validation 5, dla każdego układu parametrów. Dokonaliśmy tego za pomocą funkcji StratifiedKFold z biblioteki Sklearn.

Podczas uczenia zbieraliśmy dane oraz zapisywaliśmy je do plików za pomocą biblioteki "pickle":

- title - nazwy sieci używane przy tworzeniu wykresów
- avg - średnie metryk obliczone z cross validation danego modelu
- desc - opis każdego uczenia w cross validation
- history - zapis historii uczenia modelu
- conf_mat - zapis z macierzy pomyłek dla zbioru walidującego
- params - parametry hiperparametryzacji użyte w danej iteracji

6.4.4 Analiza wyników

Podczas treningu modeli sztucznych sieci neuronowych dla każdej epoki były zbierane następujące parametry:

- Wartość funkcji straty - loss
- Dokładność modelu - accuracy
- Precyzja modelu - precision
- Czułość modelu - recall

- Pole pod wykresem krzywej ROC (AUC) - auc

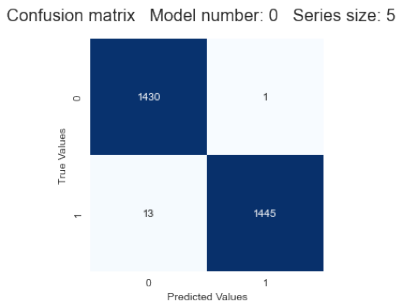
W dalszej części sprawozdania dla wyżej wymienionych parametrów używane będą nazwy znajdujące się po myślniku np. `loss` dla "Wartość funkcji straty". Dodatkowo prefix: `val_` będzie dotyczył wartości parametru uzyskanej dla zbioru walidującego. Parametry były mierzone zarówno dla predykcji na zbiorze treningowym jak i walidującym. Analiza oraz wizualizacja wszystkich wyników otrzymanych podczas treningu modeli sztucznych sieci neuronowych znajduje się w pliku `LSTM_results_analysis.ipynb`. Poniżej zostały przedstawione jedynie główne wnioski. W tabeli nr. 2 wyraźnie widać, że najlepsze wyniki modele osiągały dla długości serii równej 10, ponieważ te modele zajęły 4 pierwsze miejsca pod względem minimalnej wartości średniej funkcji straty, natomiast najgorszą długością serii danych okazała się 2, ponieważ modele trenowane na danych o takiej serii zajęły 4 ostatnie miejsca. Wśród modeli trenowanych na danych zgrupowanych w serie o długości 10 najlepszy okazał się model nr 1, czyli wielowarstwowa sieć LSTM.

Model	Seria	Min. średnia wartość funkcji straty na zbiorze walidującym	Dokładność dla min. średniej wartości funkcji straty	Max. dokładność
1	10	0.000948	0.999574	0.999574
2	10	0.001428	0.999574	0.999574
3	10	0.002218	0.999574	0.999574
0	10	0.002263	0.999575	0.999575
3	31	0.006285	1.000000	1.000000
2	5	0.011854	0.997230	0.997230
0	5	0.012597	0.995155	0.995501
2	31	0.013282	0.990476	1.000000
1	31	0.015495	0.990476	0.990476
3	5	0.016136	0.994809	0.995848
1	5	0.018092	0.994117	0.994809
0	31	0.049023	0.981385	0.990909
0	2	0.059974	0.981620	0.982866
2	2	0.060750	0.980997	0.982866
1	2	0.064003	0.979751	0.980997
3	2	0.069187	0.977259	0.979128

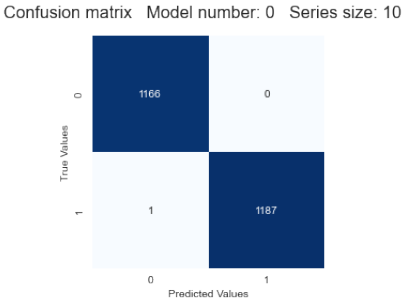
Tablica 2: Wytrenowane modele posortowane ze względu na minimalną średnią wartości funkcji straty

Fakt, że seria o długości 10 jest najlepszym wyborem bez względu na architekturę znajduje także potwierdzenie na rysunkach od 17 do 20. Przedstawiają one macierze pomyłek dla danej długości serii dla każdego modelu. W przypadku każdego modelu wyraźnie widać, że najlepsza macierz pomyłek dla poszczególnych modeli została osiągnięta dla serii o długości 10.

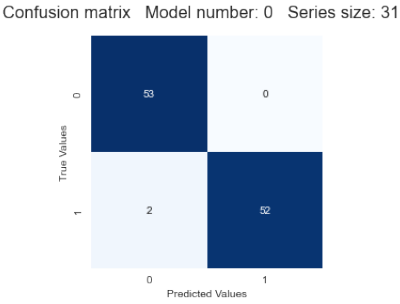
(a) Macierz pomyłek - seria 2



(b) Macierz pomyłek - seria 5



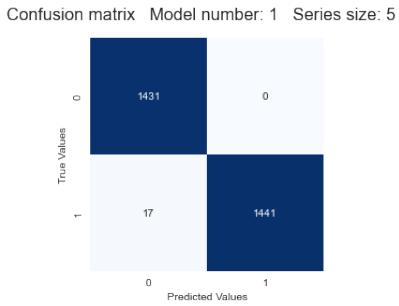
(c) Macierz pomyłek - seria 10



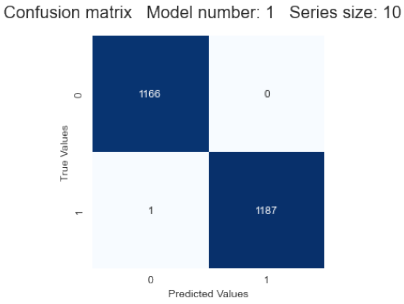
(d) Macierz pomyłek- seria 31

Rysunek 17: Macierze pomyłek dla modelu nr 0

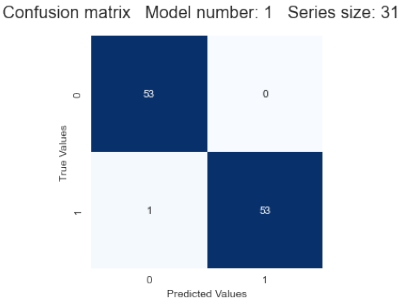
(a) Macierz pomyłek - seria 2



(b) Macierz pomyłek - seria 5

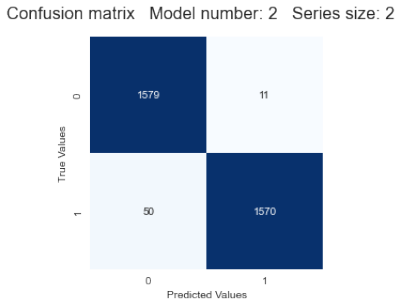


(c) Macierz pomyłek - seria 10

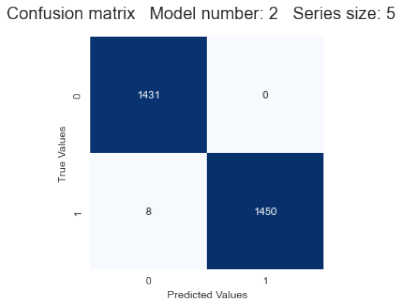


(d) Macierz pomyłek- seria 31

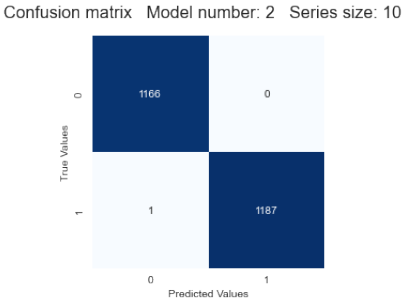
Rysunek 18: Macierze pomyłek dla modelu nr 1



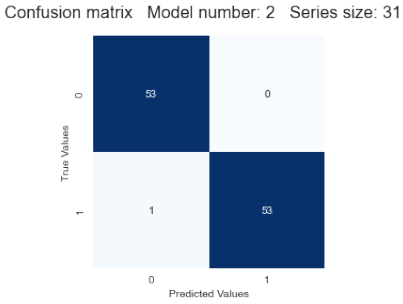
(a) Macierz pomyłek - seria 2



(b) Macierz pomyłek - seria 5

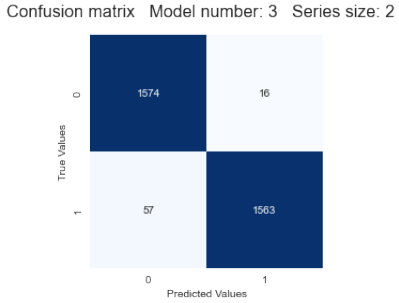


(c) Macierz pomyłek - seria 10

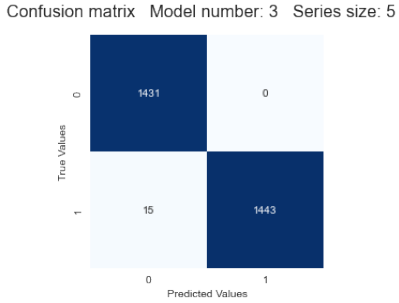


(d) Macierz pomyłek- seria 31

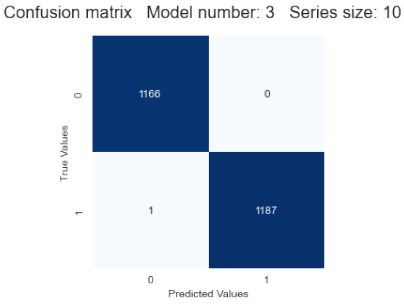
Rysunek 19: Macierze pomyłek dla modelu nr 2



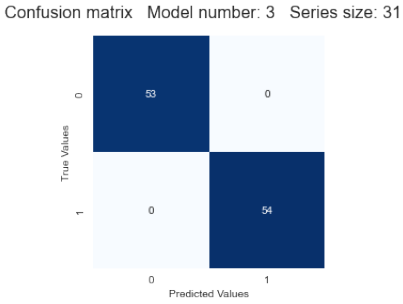
(a) Macierz pomyłek - seria 2



(b) Macierz pomyłek - seria 5



(c) Macierz pomyłek - seria 10

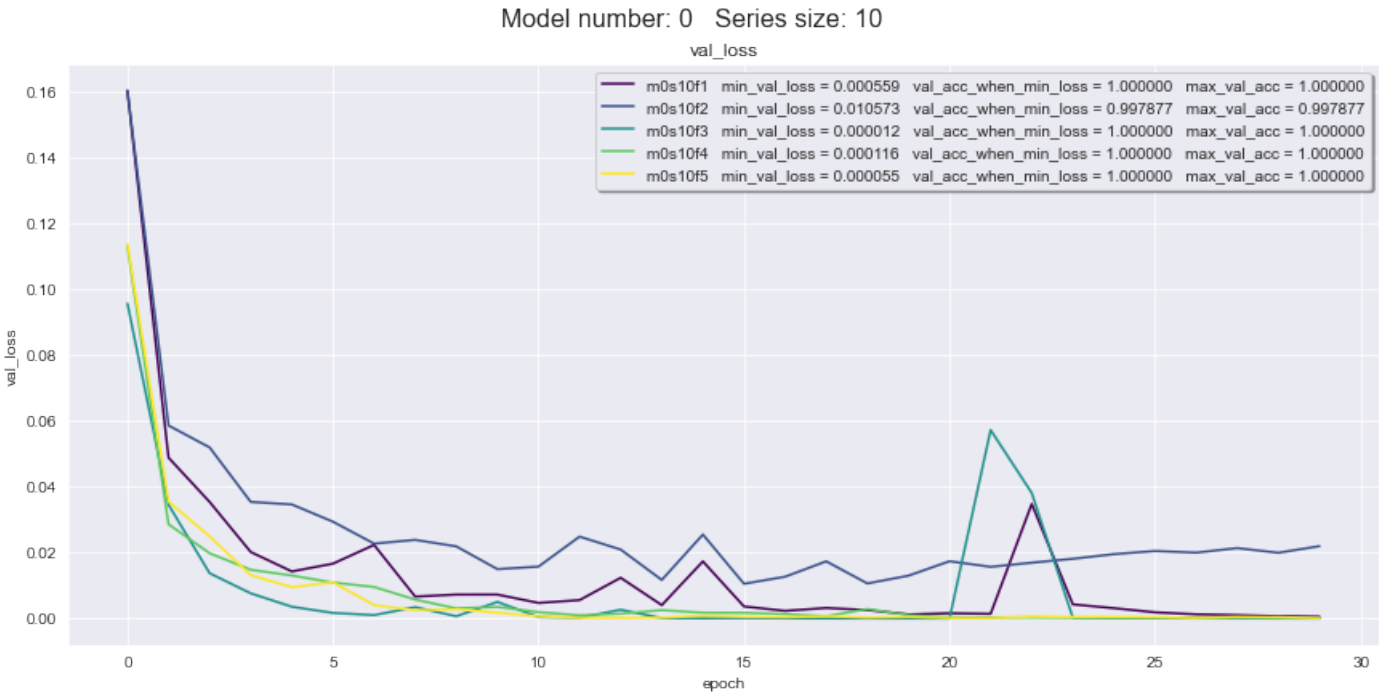


(d) Macierz pomyłek- seria 31

Rysunek 20: Macierze pomyłek dla modelu nr 3

Powodem gorszych rezultatów osiąganych przez modele trenowane na danych w seriach o długości 2 jest fakt, że takie serie są za krótkie, żeby uchwycić trend „awarii". Potwierdzenie tego faktu znajduje się w macierzach pomyłek widocznych na rysunkach od 17 do 20. Dla

każdego modelu na macierzy pomyłek dla serii danych o długości 2 widać że ilość popełnionych błędów typu TN jest znacznie większa niż FP co potwierdza hipotezę o problemach modelu w wykrywania trendu ąwarii"dla tak krótkiej serii danych. Kontynuując analizę tabeli nr. 2 oraz wspomnianych macierzy pomyłek łatwo dostrzec, że modele osiągają podobne rezultaty podczas treningu na danych zgrupowanych w seriach o długości 5 oraz 31. Jednakże podczas dokładniejszej analizy uzyskanych rezultatów zauważono, że dowolny model wytrenowany na seriach danych o długości 31 jest zdecydowanie bardziej niestabilny od modelu trenowanego na danych o długości serii równej 5. Prawdopodobnym powodem tego zjawiska jest zdecydowana mniejsza ilość danych co sprawia, że `batch_size` o rozmiarze 32 jest zdecydowanie za duży i powoduje niestabilność uczenia modelu. Na rys od nr. 21 od nr. 24 znajdują się wykresy, które przedstawiają częściowe wyniki, mianowicie dokładność oraz wartość funkcji straty, uzyskane podczas cross walidacji modelu nr 0 dla serii danych o długości 10 oraz 31. Porównanie tych wykresów pokazuje, że dla danych zgrupowanych w seriach o długości danych trening jest stabilny i model stopniowo się poprawia. Przeciwnie zachowuje się model trenowany na danych o długości 31, co szczególnie dobrze widać na rys. 24. Uzyskana skuteczność modelu zmienia się w zakresie nawet kilkunastu procent bez względu na zbiór walidujący wylosowany w procesie cross-walidacji.



Rysunek 21: Wartość funkcji straty w zależności od epoki podczas cross-walidacji modelu nr 0 dla serii danych o długości 10

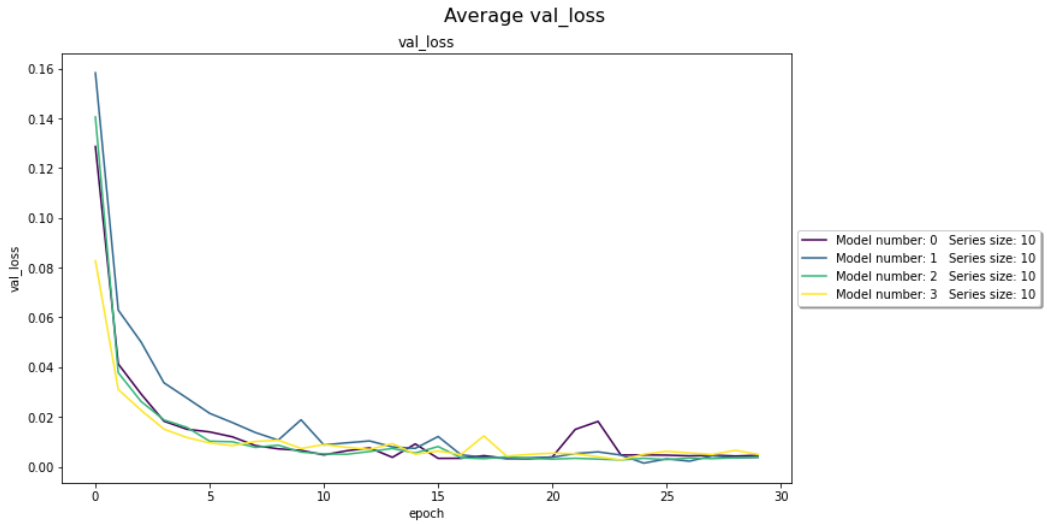
Rysunek 22: Wartość funkcji straty w zależności od epoki podczas cross-walidacji modelu nr 0 dla serii danych o długości 31

Rysunek 23: Dokładność modelu w zależności od epoki podczas cross-walidacji modelu nr 0 dla serii danych o długości 10

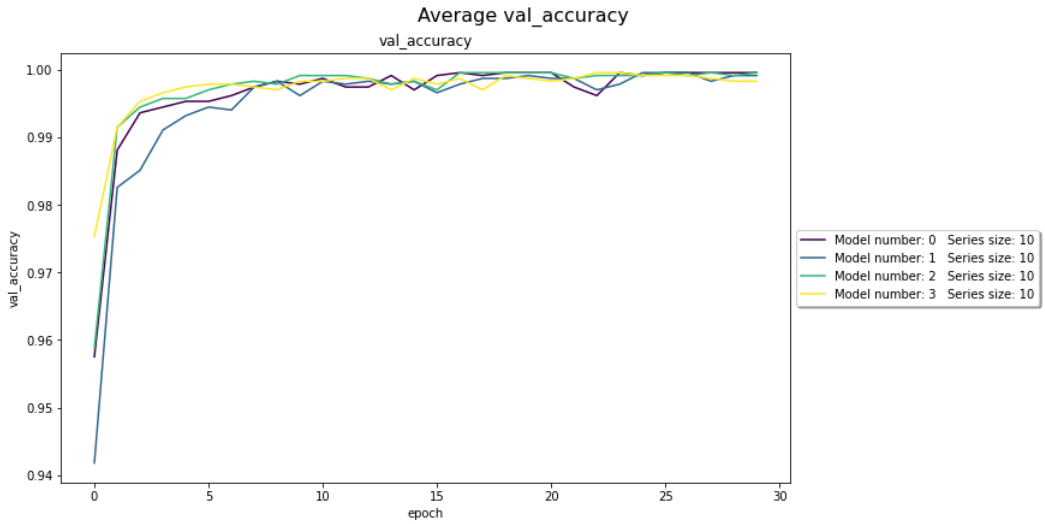
Rysunek 24: Dokładność modelu w zależności od epoki podczas cross-walidacji modelu nr 0 dla serii danych o długości 31

6.4.5 Analiza wyników dla najlepszej serii o długości 10

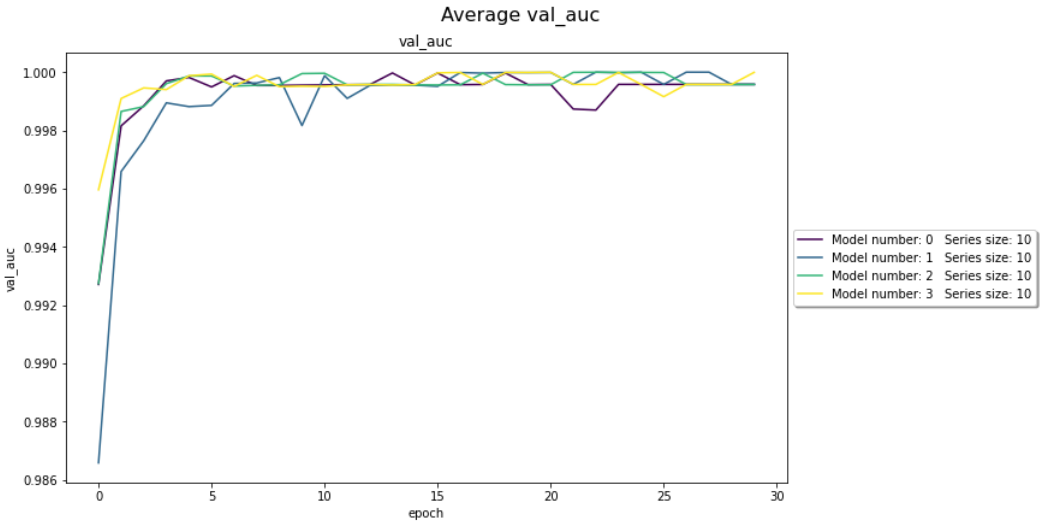
Długość serii ustawiona na wartość 10 była dla nas zdecydowanie najlepsza. Mimo to, że najlepsze wyniki dla tej długości patrząc na `val_loss` (Rysunek nr 25) dawał nam model nr 1, wszystkie z nich osiągnęły to samo `max_val_accuracy` (Rysunek nr 26). Wykres `val_auc` (Rysunek nr 27) bardzo mocno przypomina swoim wyglądem `val_accuracy` (Rysunek nr 26) ze względu na podobną ilość danych w każdej z klas. Dodatkowo dzięki wykresom `val_recall` (Rysunek nr 28) i `val_precision` (Rysunek nr 29) możemy potwierdzić, że zdecydowana większa ilość błędów polega na błędnie sklasyfikowanym stanie awarii, niż błędnie sklasyfikowanym stanie normalnym.



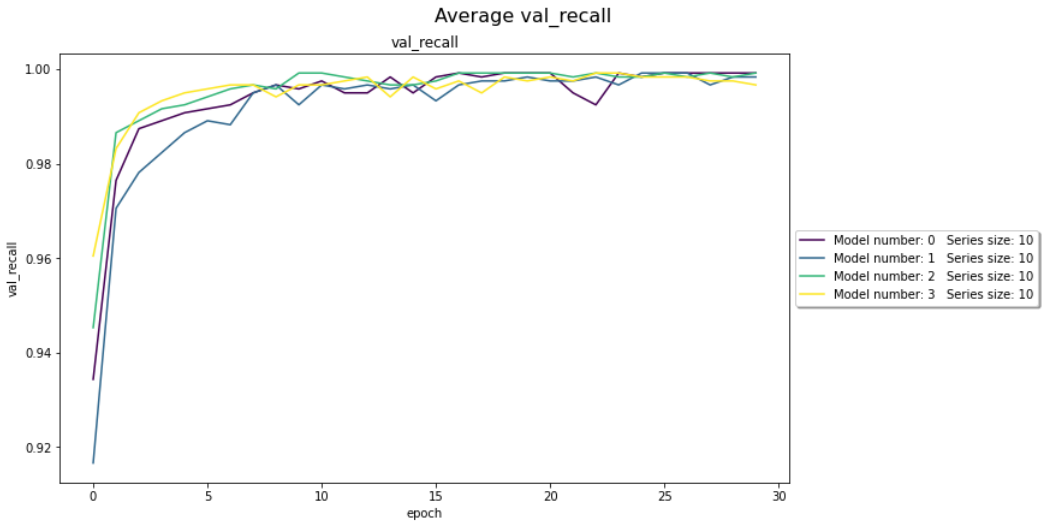
Rysunek 25: Uśredniona wartość val_loss z cross-walidacji dla wszystkich modeli uczonych długością serii 10.



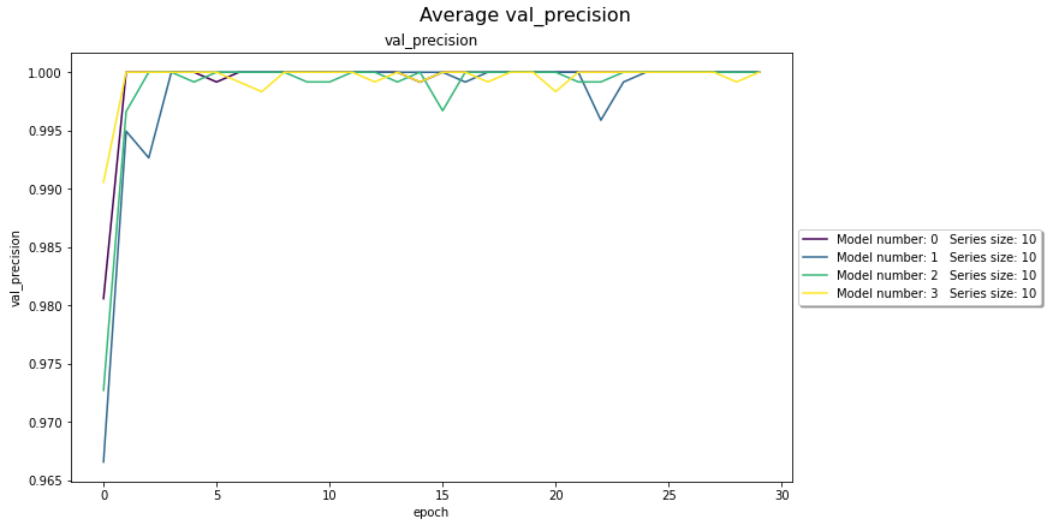
Rysunek 26: Uśredniona wartość val_accuracy z cross-walidacji dla wszystkich modeli uczonych długością serii 10.



Rysunek 27: Uśredniona wartość val_auc z cross-walidacji dla wszystkich modeli uczonych długością serii 10.



Rysunek 28: Uśredniona wartość val_recall z cross-walidacji dla wszystkich modeli uczonych długością serii 10.

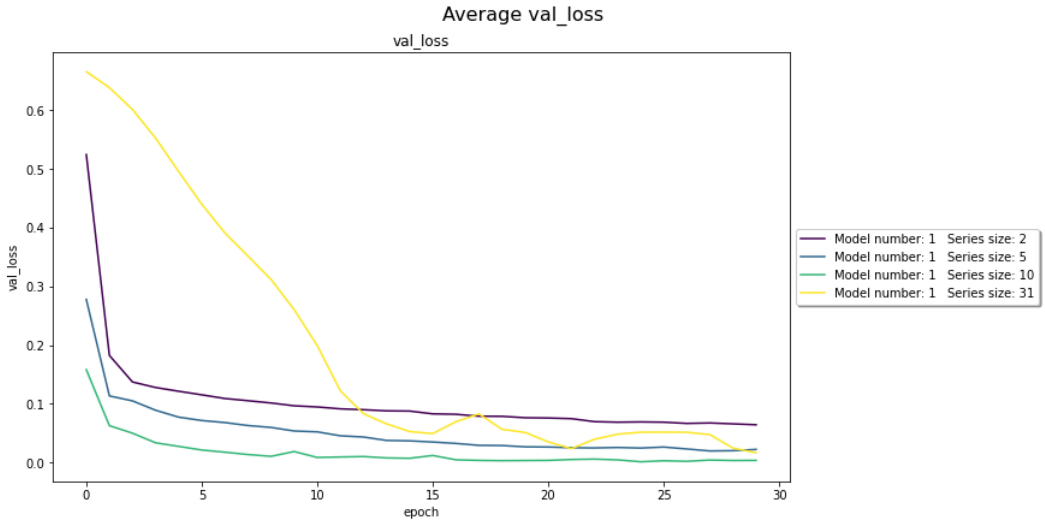


Rysunek 29: Uśredniona wartość val_precision z cross-walidacji dla wszystkich modeli uczonych długością serii 10.

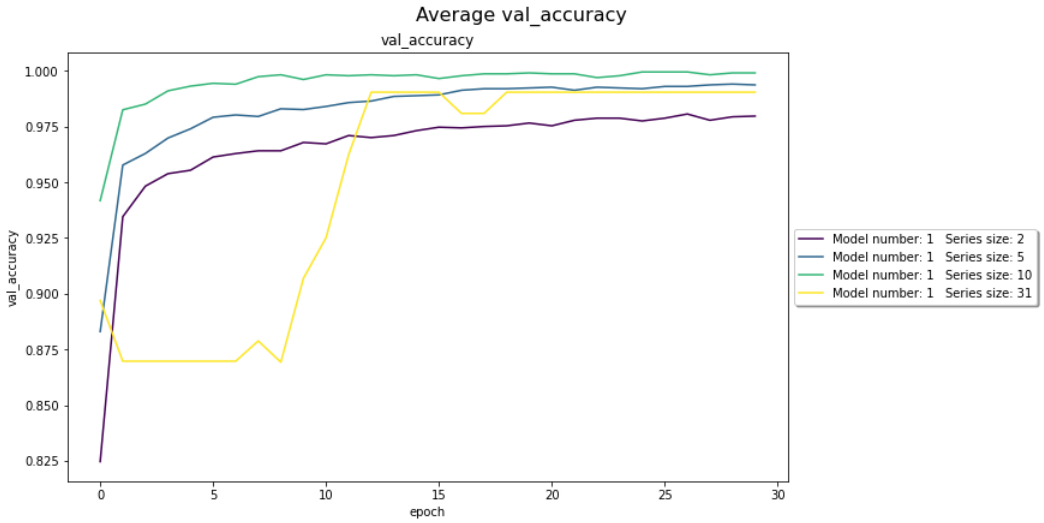
6.4.6 Analiza wyników dla najlepszego modelu nr 1

Model nr 1 (wielowarstwowa sieci LSTM) okazała się najlepsza dla długości serii 10 i przez połączenie tych dwóch zmiennych otrzymaliśmy najniższe `val_loss` w całym naszym projekcie.

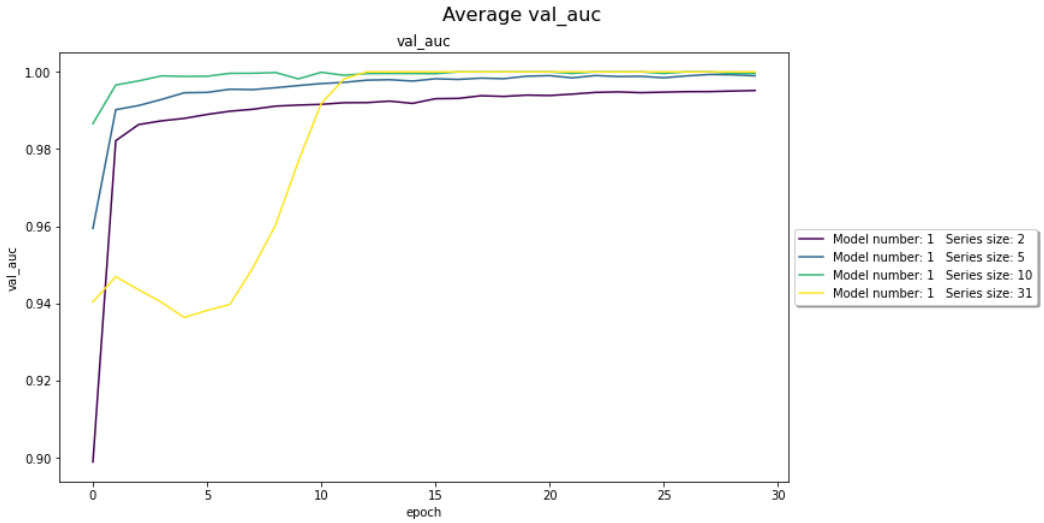
Poniżej przedstawione są wykresy przedstawiające przebieg uczenia dla tego modelu, które jeszcze mocniej nas utwierdzają we wszystkich wyżej wyciągniętych wnioskach. Serie o długości 10 działają zdecydowanie najlepiej, następnie długość 5 i 10 wykazuje podobne wyniki, natomiast seria o długości 2 radzi sobie najgorzej, możemy to zauważyć na wykresie `val_loss` (Rysunek nr 30), `val_accuracy` (Rysunek nr 31) oraz `val_auc` (Rysunek nr 32). Dodatkowo kolejny raz możemy zobaczyć rozbieżność pomiędzy `val_recall` (Rysunek nr 33) i `val_precision` (Rysunek nr 34) tylko tym razem dla różnych rozmiarów serii.



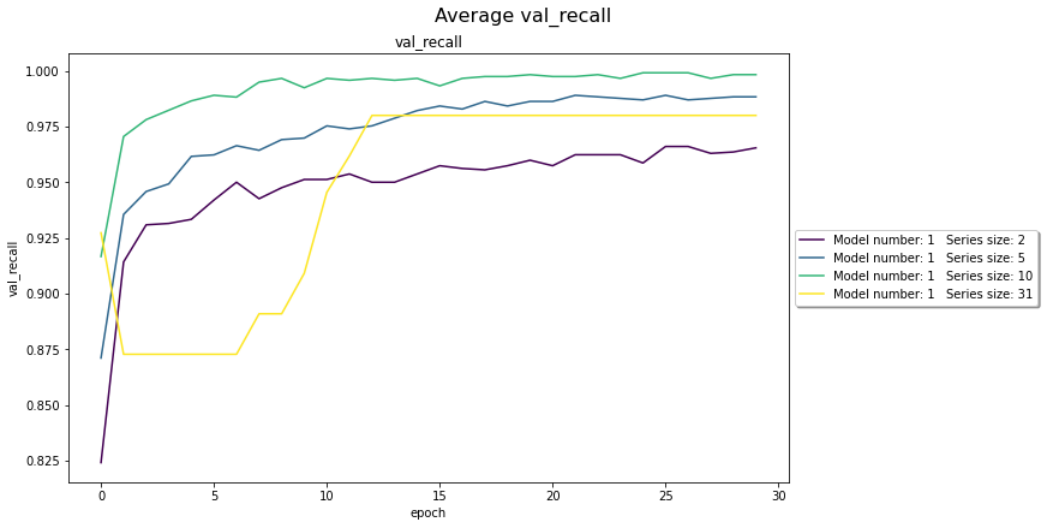
Rysunek 30: Uśredniona wartość val_loss z cross-walidacji dla modelu nr 1 (wielowarstwowy LSTM) z ukazaniem wyników dla poszczególnych długości serii.



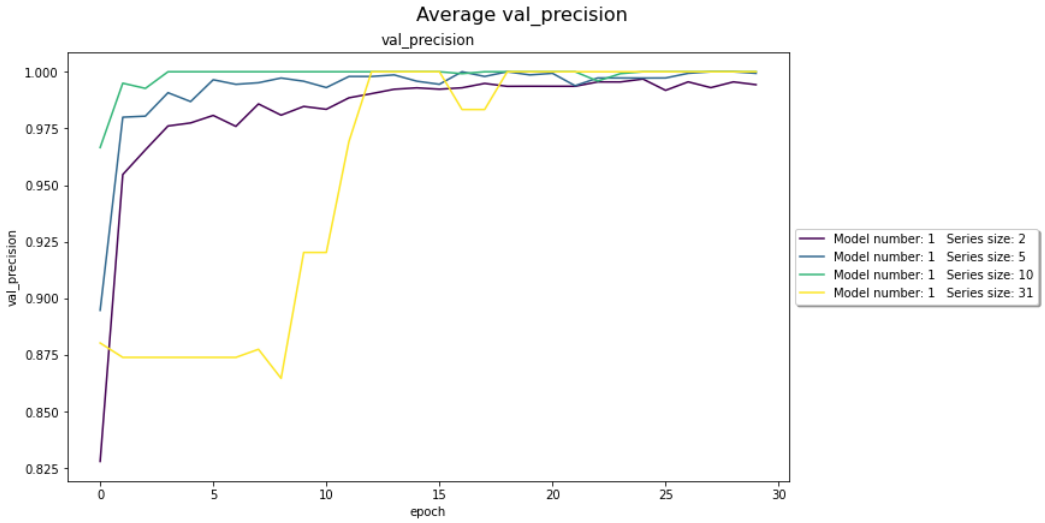
Rysunek 31: Uśredniona wartość val_accuracy z cross-walidacji dla modelu nr 1 (wielowarstwowy LSTM) z ukazaniem wyników dla poszczególnych długości serii.



Rysunek 32: Uśredniona wartość val_auc z cross-walidacji dla modelu nr 1 (wielowarstwowy LSTM) z ukazaniem wyników dla poszczególnych długości serii.



Rysunek 33: Uśredniona wartość val_recall z cross-walidacji dla modelu nr 1 (wielowarstwowy LSTM) z ukazaniem wyników dla poszczególnych długości serii.



Rysunek 34: Uśredniona wartość val_precision z cross-walidacji dla modelu nr 1 (wielowarstwowy LSTM) z ukazaniem wyników dla poszczególnych długości serii.

7 Podsumowanie

Efektem końcowym projektu jest dogłębna analiza porównawcza 4 architektur sztucznych sieci neuronowych typu LSTM i GRU. Model każdej z architektur jest w stanie bez problemu zrealizować założony problem, mianowicie nauczyć się rozpoznawać czy w trakcie pracy urządzenia doszło do awarii. W celu ułatwienia kontynuacji pracy z projektem osobom trzecim został stworzony plik `test_of_the_best_models.ipynb`. Umożliwia on użytkownikowi wczytanie dowolnego wcześniej wytrenowanego modelu np. z folderu `models/` lub `best_models/` i wykorzystanie go do dalszej analizy porównawczej.