Prognozowanie szeregów czasowych

Marcin Wierzbiński

Szereg czasowy (ang. time series) to sekwencja danych numerycznych, które są zebrane w kolejnych punktach czasowych. Każdy punkt w szeregu czasowym reprezentuje wartość mierzonej lub obliczanej zmiennej w określonym momencie czasu.

Przykłady szeregów czasowych obejmują między innymi:

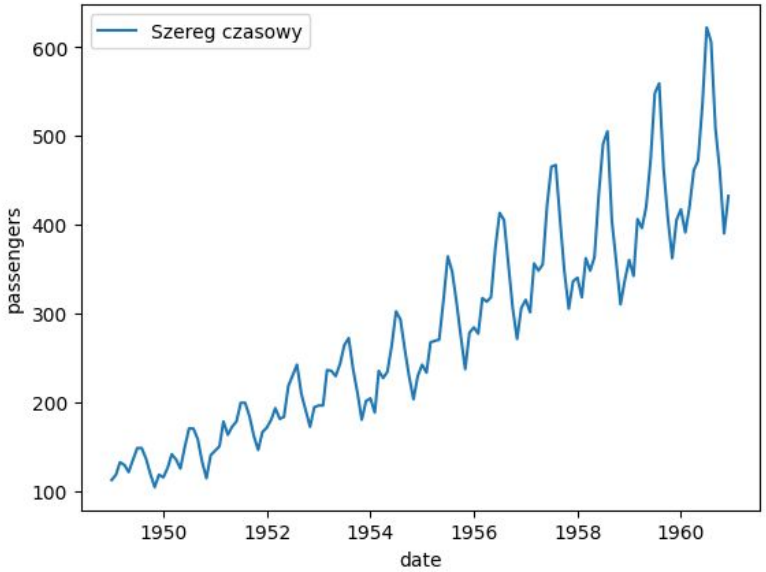
- wartości giełdowe w czasie
- liczba odwiedzin strony internetowej w ciągu dnia
- temperatura w określonym miejscu w różnych punktach czasowych
- poziom zanieczyszczenia powietrza w danym miejscu w określonym momencie czasu

Szeregi czasowe są często badane w celu identyfikacji wzorców lub trendów, które mogą pomóc w prognozowaniu przyszłych wartości.

Analiza szeregów czasowych wykorzystuje techniki statystyczne i uczenie maszynowe, aby znaleźć zależności między punktami czasowymi i przewidzieć przyszłe wartości szeregu czasowego.

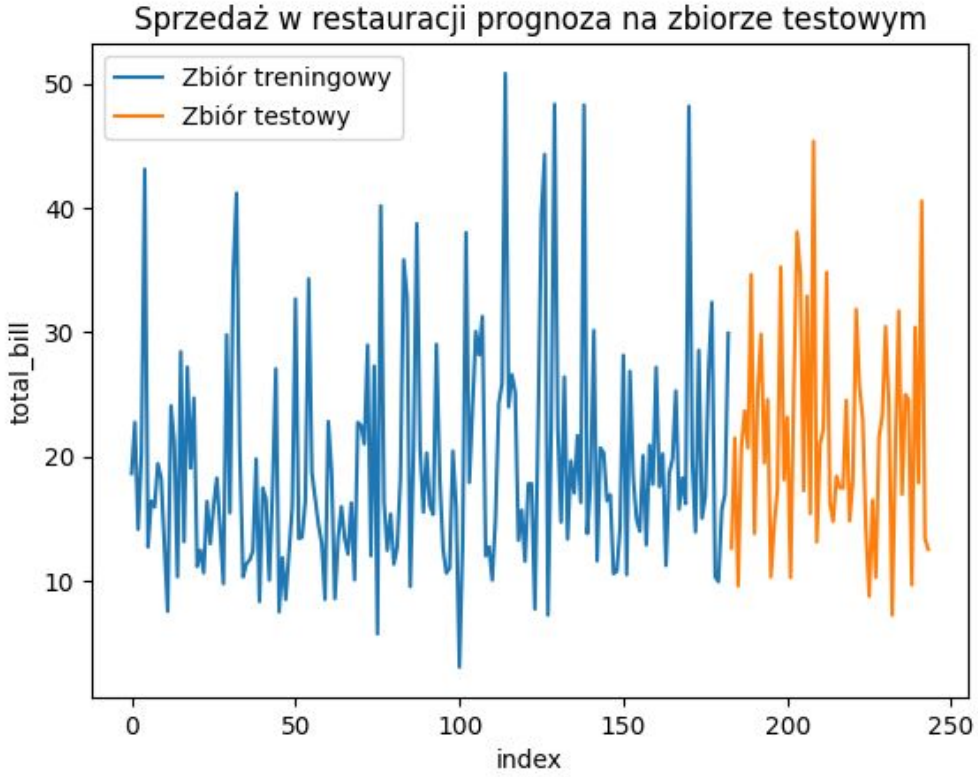
Przykład zbioru danych

- Zbiór danych zwany "Airline Passengers", który jest szeregiem czasowym reprezentującym miesięczną liczbę pasażerów przewoźnika lotniczego od stycznia 1949 do grudnia 1960 roku.
- Zbiór ten zawiera 144 obserwacje i jest często wykorzystywany do celów analizy i modelowania szeregów czasowych.

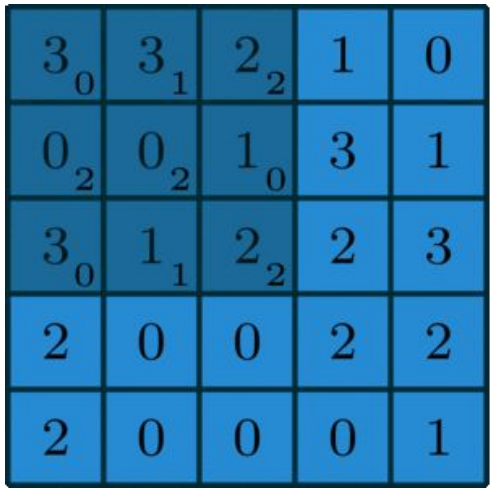


	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
5	1949	June	135
6	1949	July	148
7	1949	August	148
8	1949	September	136
9	1949	October	119

Toy example


Sprzedaż w restauracji prognoza na zbiorze testowym
Zbiór treningowy
Zbiór testowy
total_bill
index
50
40
30
20
10
0
50
100
150
200
250

Przypomnienie: Filtr dla konwolucji 2D -

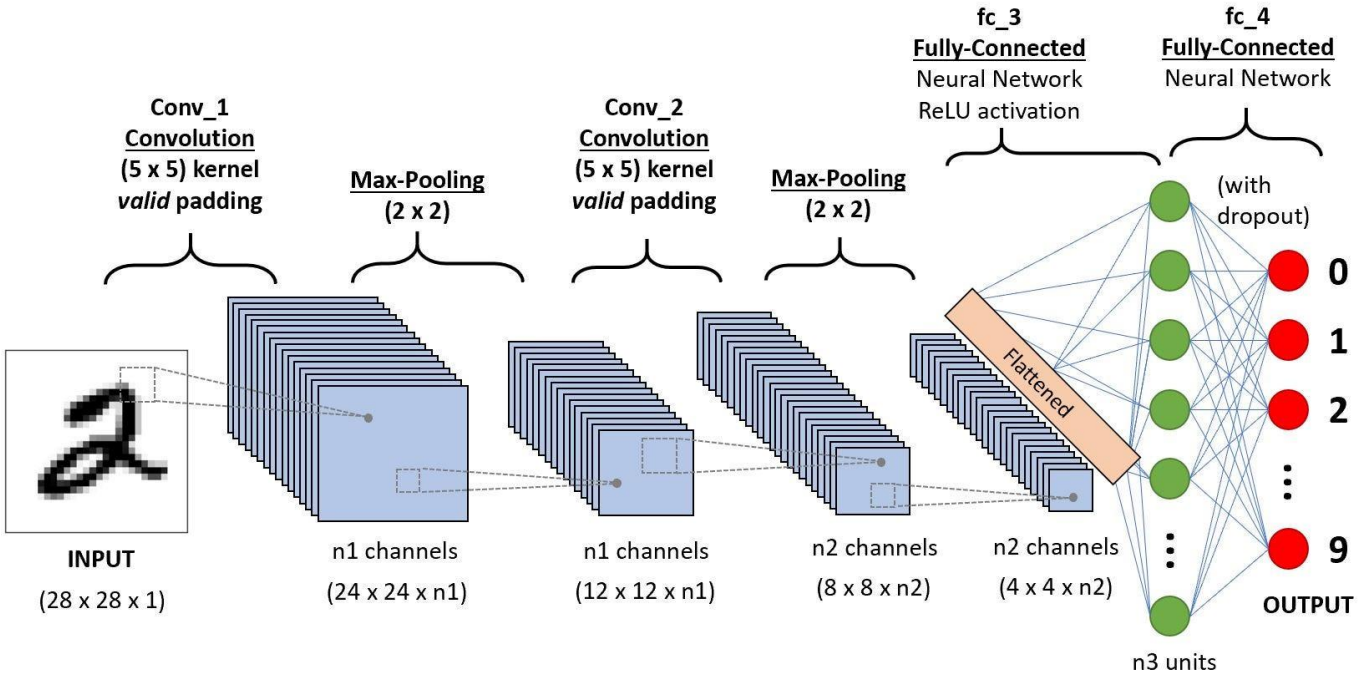


obraz

macierz wyjściowa

- Filtr w konwolucji 2D to macierz wag, która jest przesuwana po obrazie wejściowym, piksel po pikselu, w celu uzyskania wynikowego obrazu konwolucyjnego.
- Filtr służy do ekstrakcji cech z obrazów wejściowych.
- Filtr składa się z kwadratowej macierzy wag o rozmiarze n x n, gdzie n jest liczbą nieparzystą.

Konwolucja 2 wymiarowa - przypomnienie



- Kanał to zbiór filtrów.
- Każdy filtr w kanale wykrywa inny wzorzec w obrazie i wyodrębnia określoną cechę.

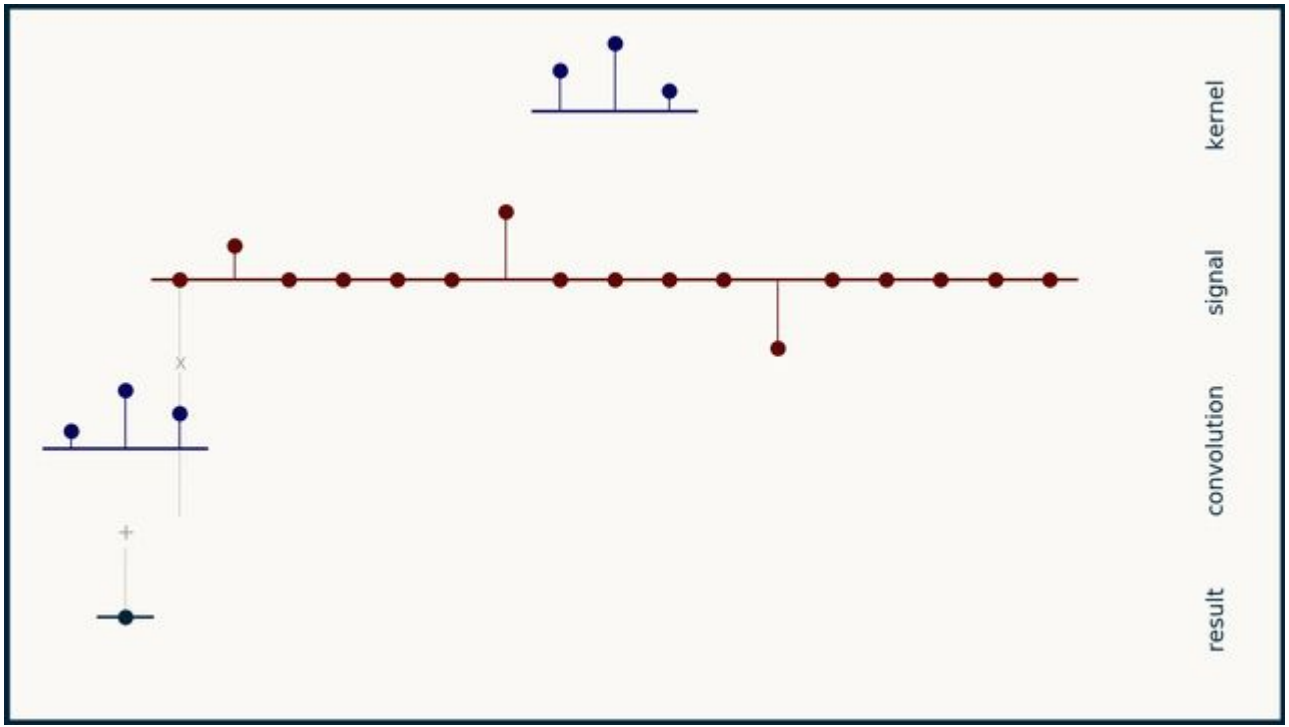
Konwolucja jednowymiarowa

Konwolucja jednowymiarowa to operacja matematyczna wykorzystywana w sieciach neuronowych, w której na podstawie dwóch wektorów (np. sygnału wejściowego i filtru) tworzony jest nowy wektor wynikowy.

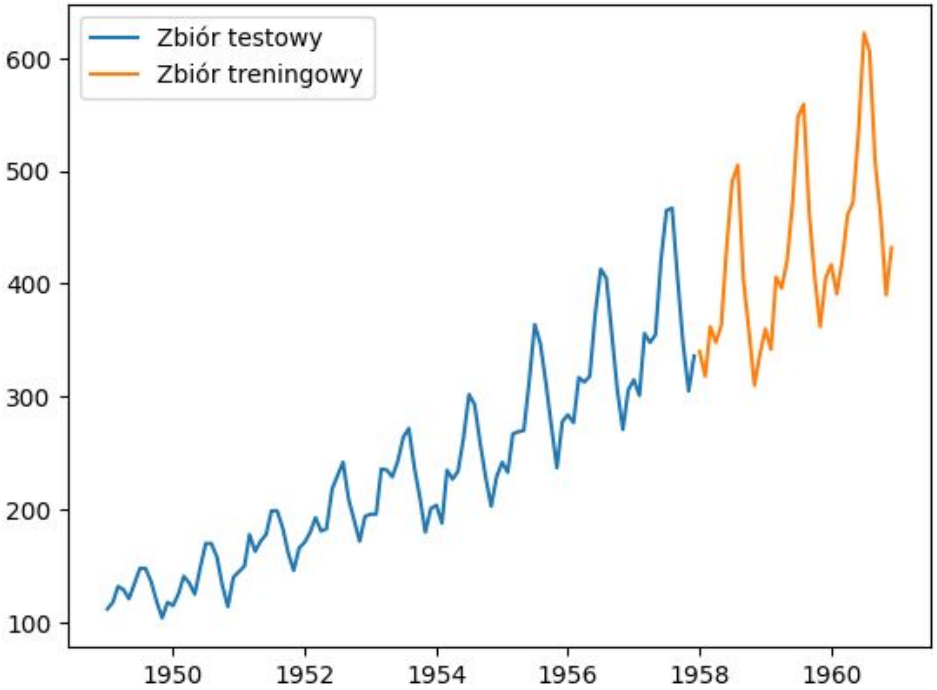
W przypadku sieci neuronowych, konwolucja jednowymiarowa jest stosowana w celu ekstrakcji cech z sygnału wejściowego.

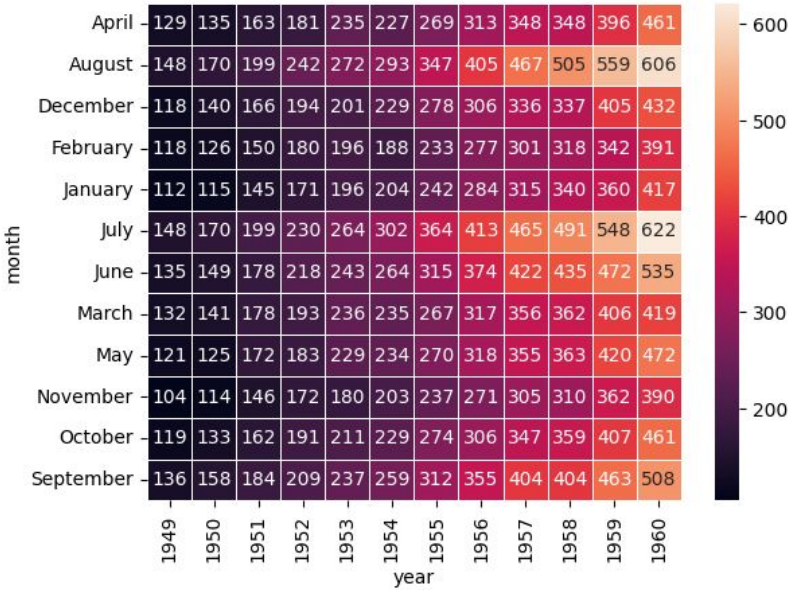
Konwolucja jednowymiarowa polega na przesuwaniu okna (filtru) po wejściowym sygnale i wykonaniu operacji mnożenia skalarowego pomiędzy wartościami sygnału a wagami filtra. Wartości te są następnie sumowane i zapisywane jako nowa wartość w wynikowym sygnale.

Conv 1D



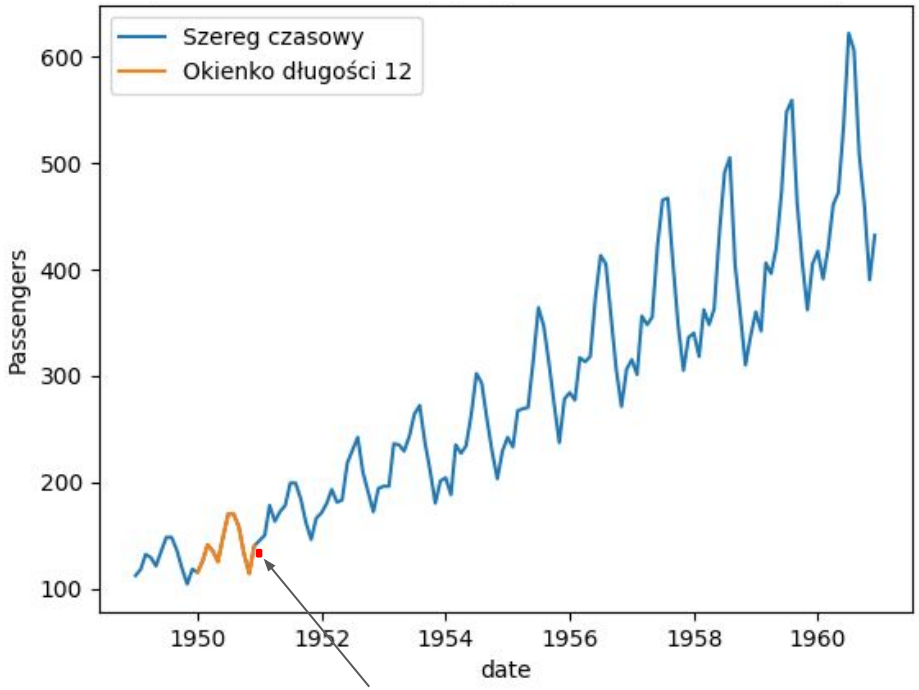
Model prognozy pasażerów w latach od 1958





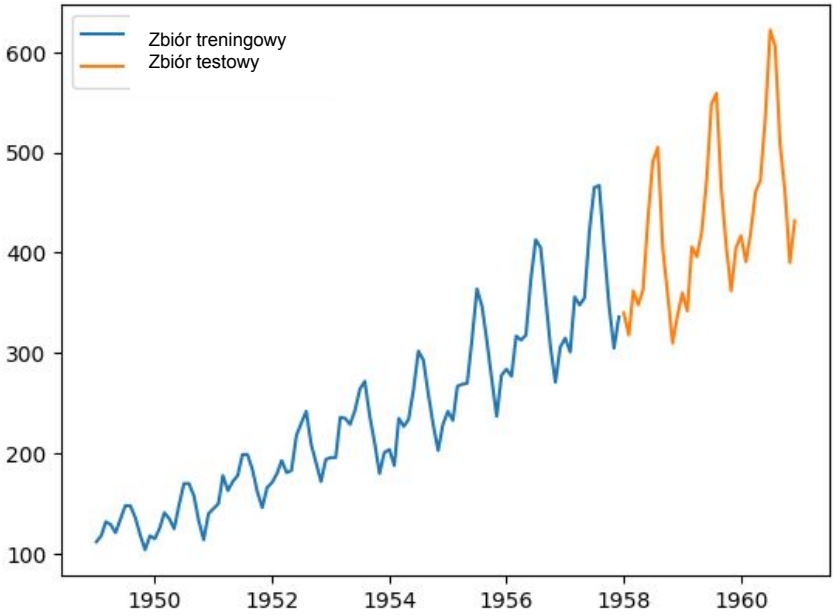
Tworzenie okienek czasowych

	year	month	passengers	date
12	1950	January	115	1950-01-01
13	1950	February	126	1950-02-01
14	1950	March	141	1950-03-01
15	1950	April	135	1950-04-01
16	1950	May	125	1950-05-01
17	1950	June	149	1950-06-01
18	1950	July	170	1950-07-01
19	1950	August	170	1950-08-01
20	1950	September	158	1950-09-01
21	1950	October	133	1950-10-01
22	1950	November	114	1950-11-01
23	1950	December	140	1950-12-01

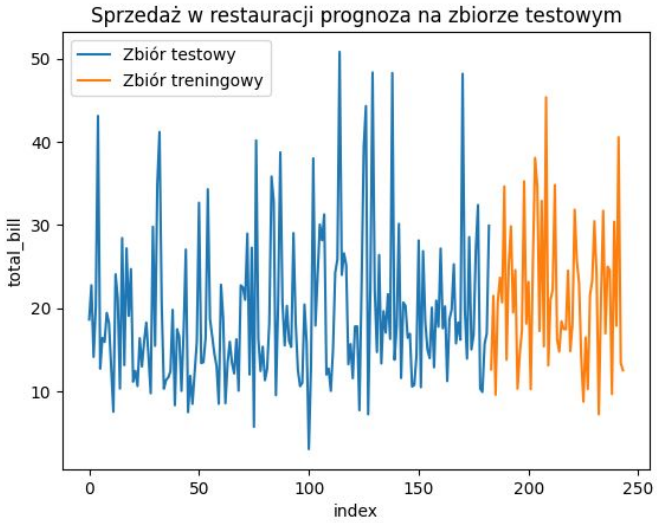


przewidywanie

Prognoza lotów wynik:



Case study: tips



	total_bill	tip	sex	smoker	day	time	size	date
0	18.64	1.36	Female	No	Thur	Lunch	3	2023-02-02 11:04:39
1	22.76	3.00	Male	No	Thur	Lunch	2	2023-02-02 11:05:09
2	14.15	2.00	Female	No	Thur	Lunch	2	2023-02-02 11:08:50
3	20.27	2.83	Female	No	Thur	Lunch	2	2023-02-02 11:10:11
4	43.11	5.00	Female	Yes	Thur	Lunch	4	2023-02-02 11:15:38

Wczytywanie danych

```
# importowanie potrzebnych bibliotek
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler

# wczytanie danych
df = pd.read_csv('https://raw.githubusercontent.com/marcin119a/PODSTAWY-UCZENIA-MASZYNOWEGO-W-PYTHONIE-/main/tips_date.csv' )

# konwersja daty na format datetime i ustawienie jej jako indeks
df['date'] = pd.to_datetime(df['date'])
# przygotowanie danych treningowych i testowych
data = df[['total_bill','date']]
data = data.set_index('date')

# Normalizacja danych
scaler = MinMaxScaler()
data = scaler.fit_transform(data)
```

Tworzenie okienek

```
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
from sklearn.metrics import mean_squared_error

size = int(len(data) * 0.75)

train_data, test_data = data[ 0:size], data[size: len(data)]

# Zmniejszenie wymiarowości danych
window_size = 4

def create_dataset(X, y, time_steps=1):
   Xs, ys = [], []
   for i in range(len(X) - time_steps):
       Xs.append(X[i:(i+time_steps)])
       ys.append(y[i+time_steps])
   return np.array(Xs), np.array(ys)

TIME_STEPS = 12
X_train, y_train = create_dataset(train_data, train_data, window_size)
X_test, y_test = create_dataset(test_data, test_data, window_size)
```

Prosty model konwolucji 1D

```
# tworzenie modelu sieci konwolucyjnej
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(window_size,1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(1))

# kompilacja modelu
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse', metrics='mse')

# trenowanie modelu
model.fit(X_train, y_train, epochs=100, batch_size=5, verbose=1)
```

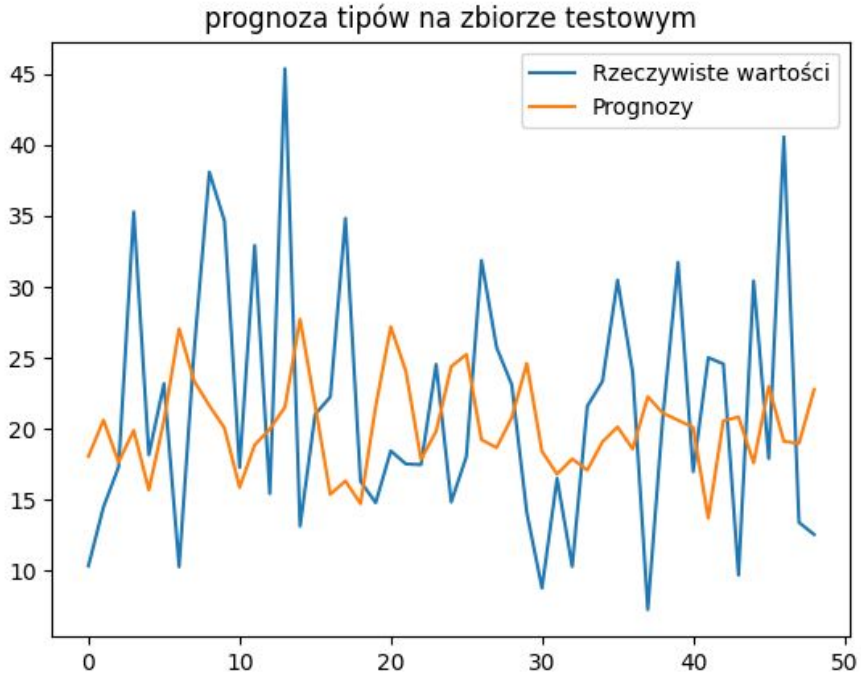
Predykcja i odwrotna normalizacja

```
y_pred = model.predict(X_test)
# Odwrotna normalizacja danych
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)

# Obliczenie błędu średniokwadratowego (MSE)
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print(f"MSE: {mse}")
```

Wizualizacja

```
# wizualizacja wyników predykcji
plt.plot(y_test, label='Rzeczywiste wartości')
plt.plot(y_pred, label='Prognozy')
plt.legend()
plt.title('prognoza tipów na zbiorze testowym')
```

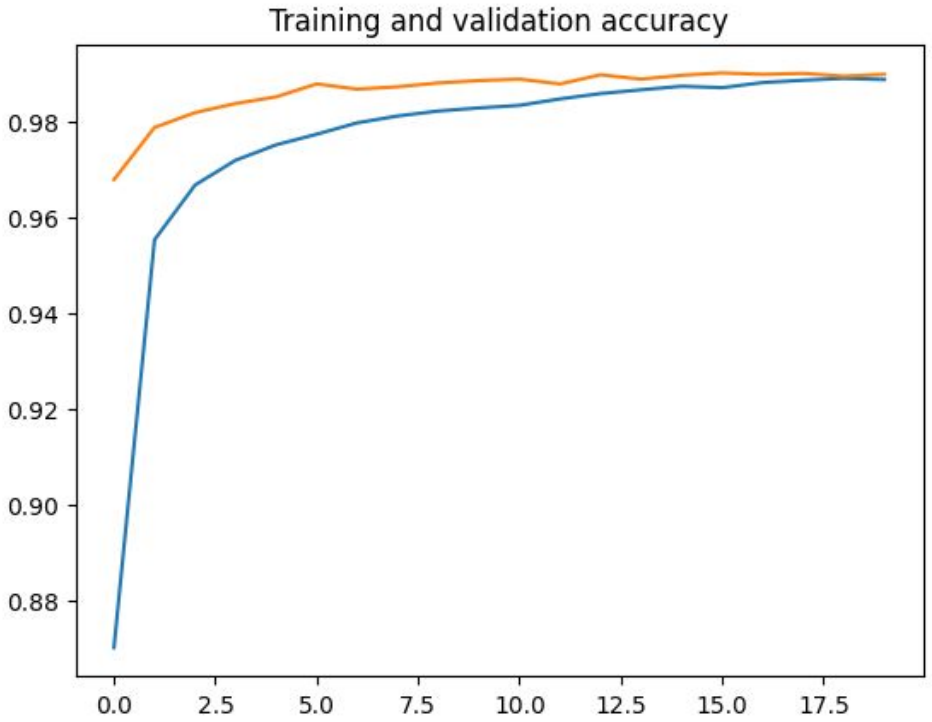


Zadanie dodatkowe MNIST conv1D

```
# Zdefiniuj model
model = Sequential()

# Dodaj warstwy konwolucyjne
model.add(Conv1D(filters=32, kernel_size=3,
activation='relu', input_shape=(28, 28)))

# TODO c.d
```



Linki:

- https://e2eml.school/convolution_one_d.html
- https://en.wikipedia.org/wiki/Time_series
- https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
-