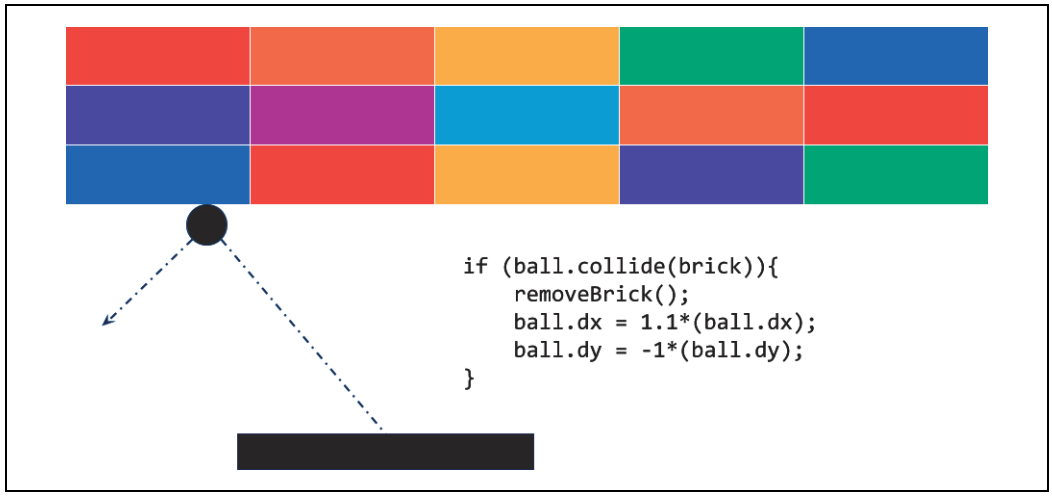
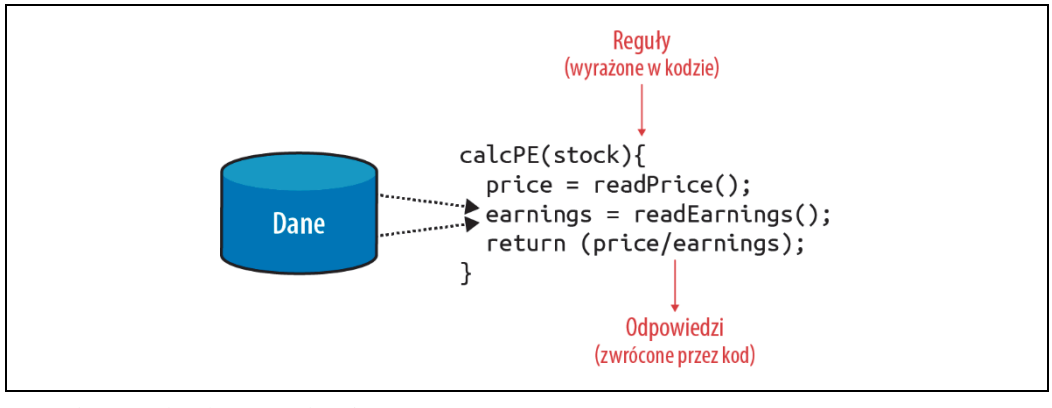
Sztuczna inteligencja i uczenie maszynowe dla programistów

Praktyczny przewodnik po sztucznej inteligencji

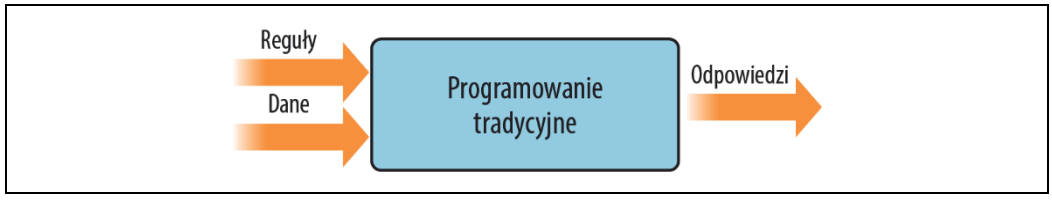
RYSUNKI



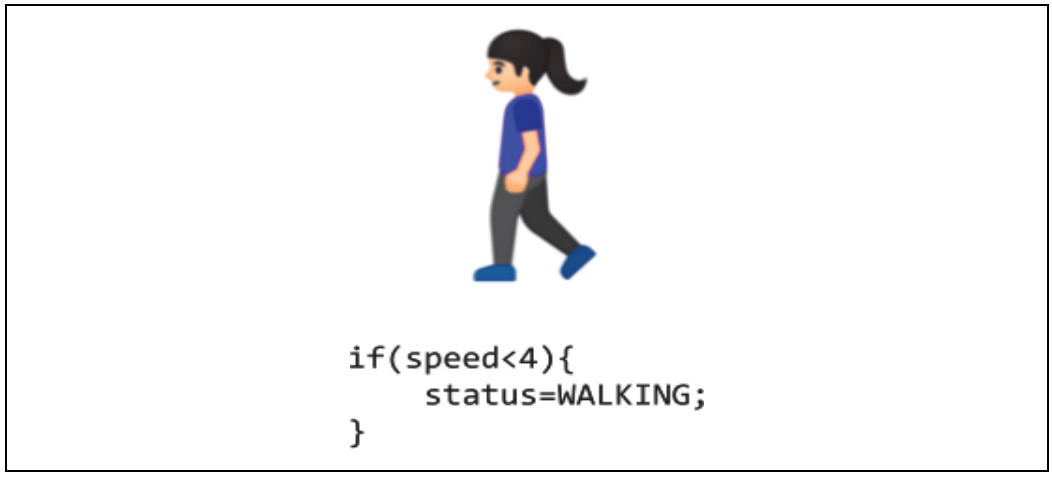
Rysunek 1.1. Kod w grze Breakout



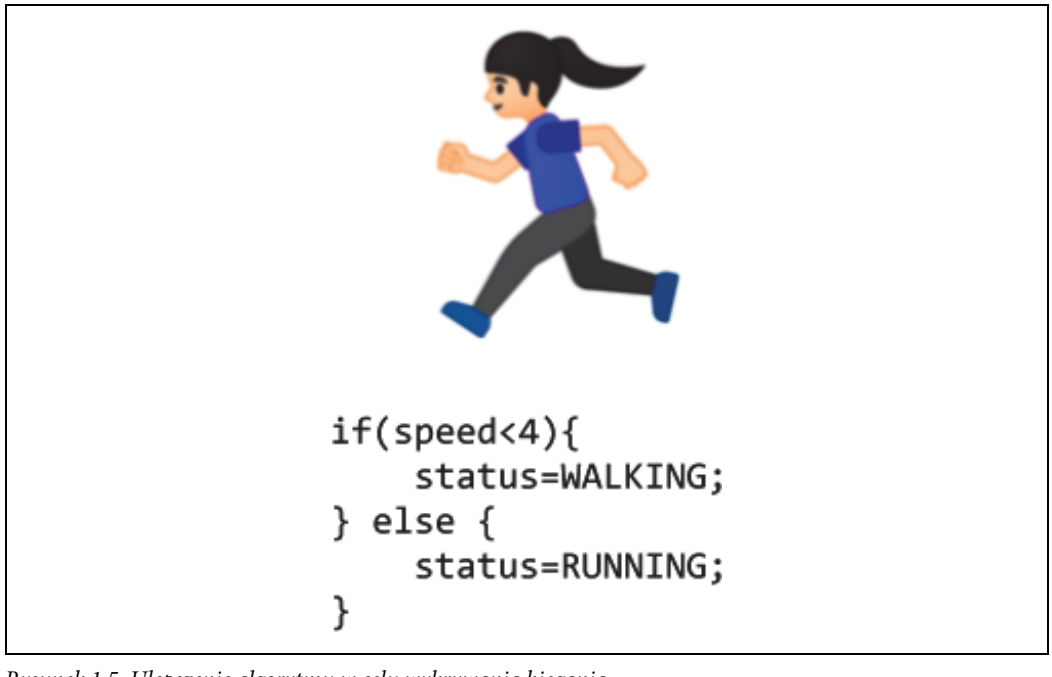
Rysunek 1.2. Kod realizujący usługi finansowe



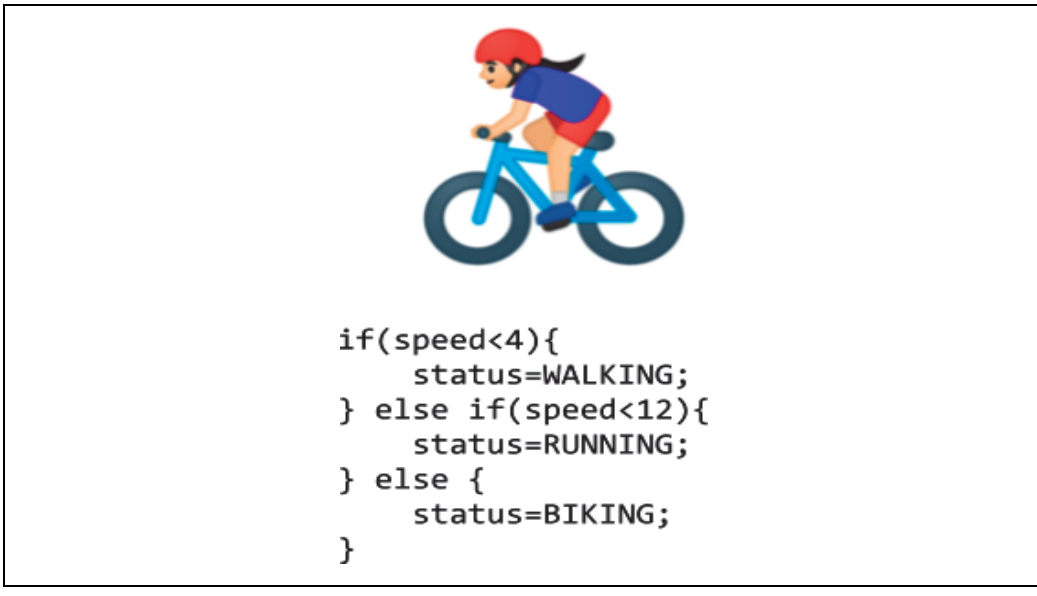
Rysunek 1.3. Wysokopoziomowa struktura programowania tradycyjnego



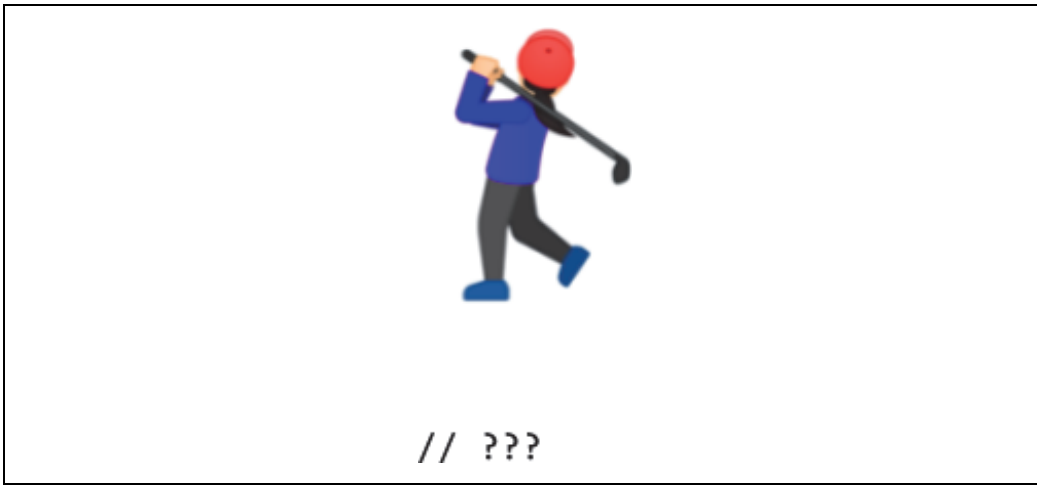
Rysunek 1.4. Algorytm wykrywający aktywność



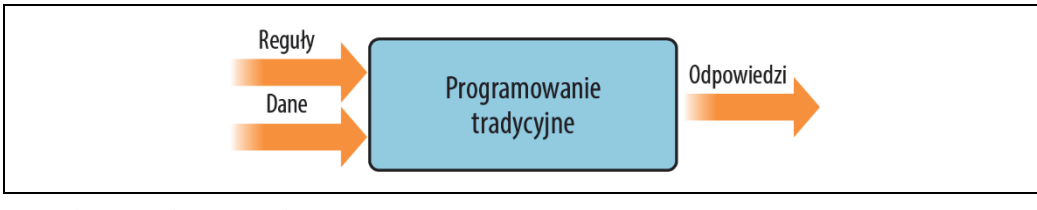
Rysunek 1.5. Ulepszenie algorytmu w celu wykrywania biegania



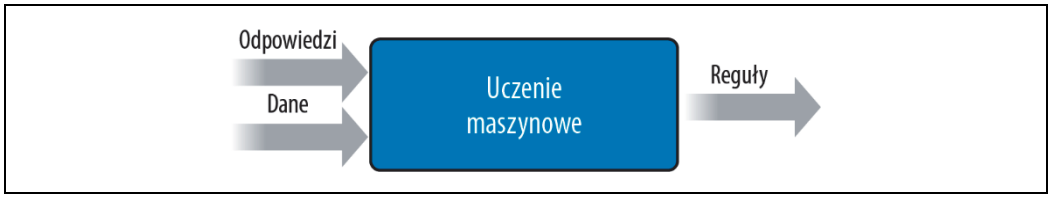
Rysunek 1.6. Ulepszenie algorytmu w celu wykrywania jazdy na rowerze



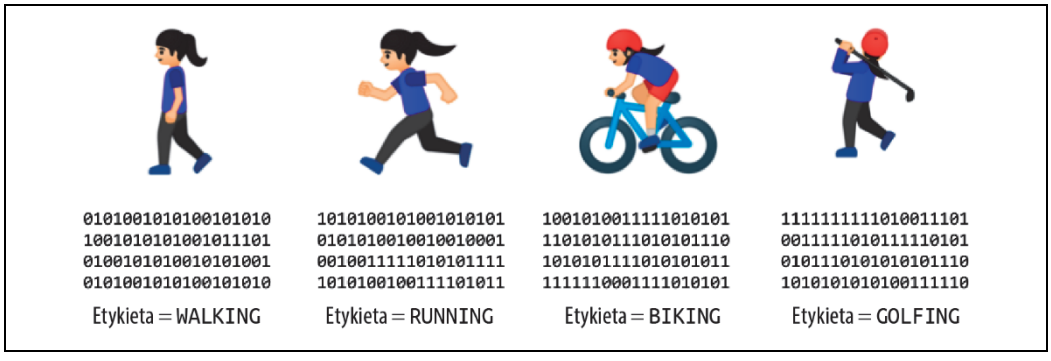
Rysunek 1.7. W jaki sposób zdefiniujemy algorytm wykrywający granie w golfa?



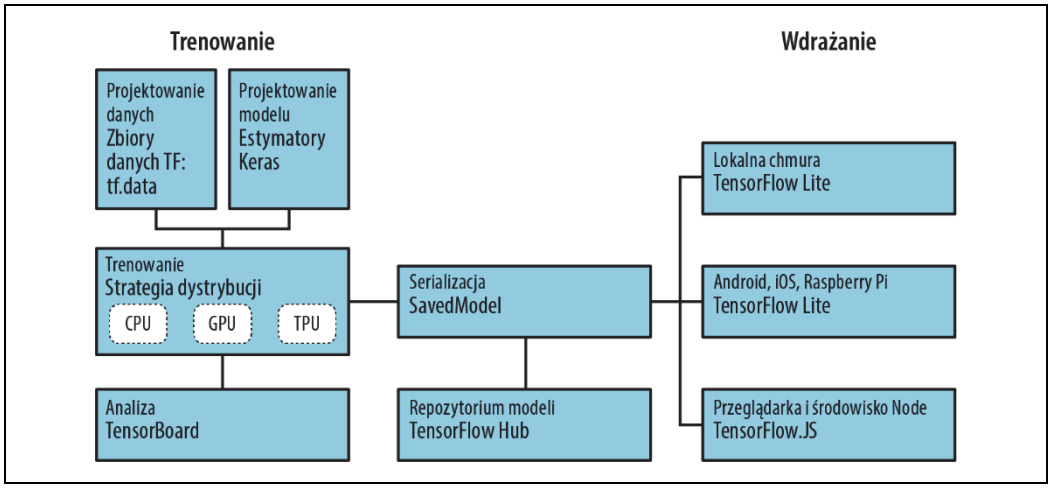
Rysunek 1.8. Tradycyjny przebieg programu



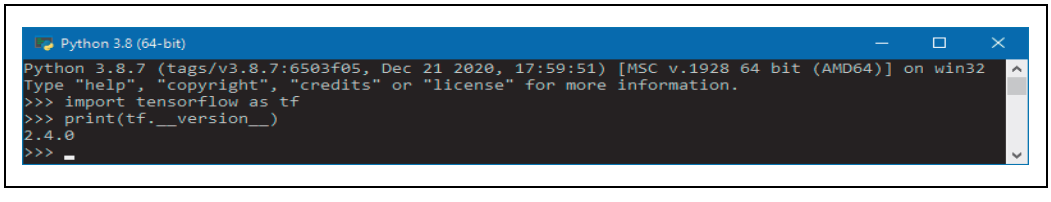
Rysunek 1.9. Zmiana kolejności działań spowodowała, że otrzymaliśmy schemat uczenia maszynowego



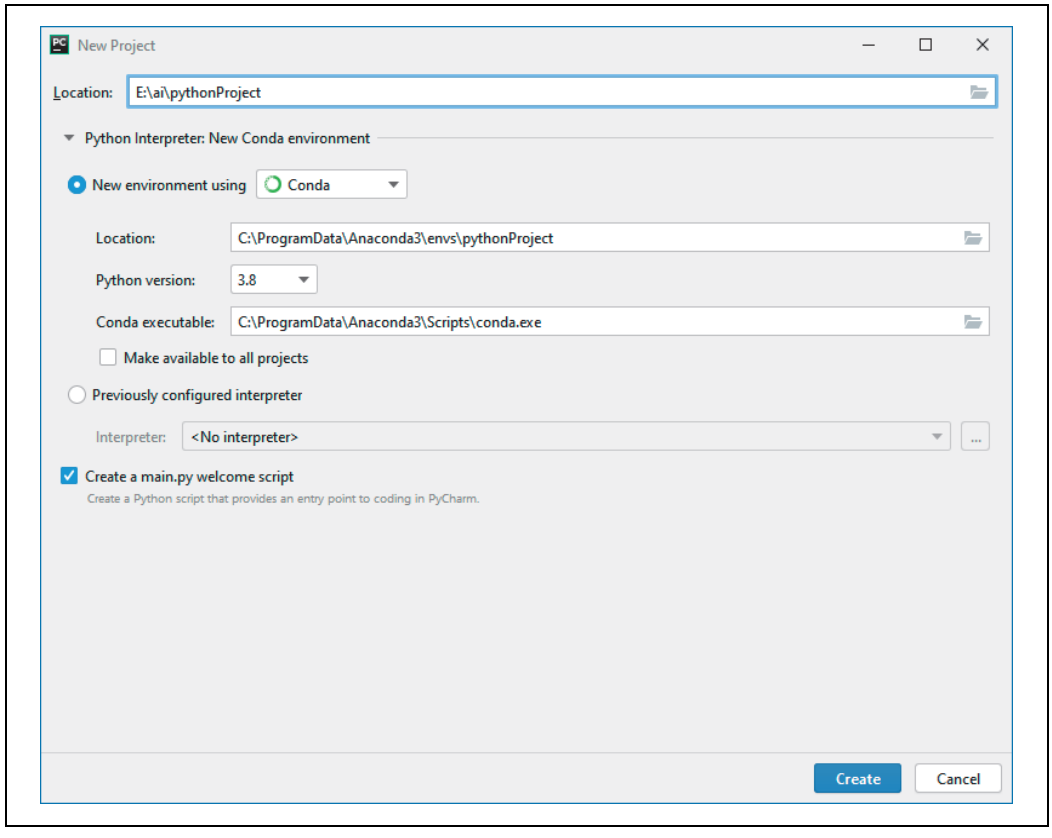
Rysunek 1.10. Od kodowania do uczenia maszynowego: gromadzenie i etykietowanie danych



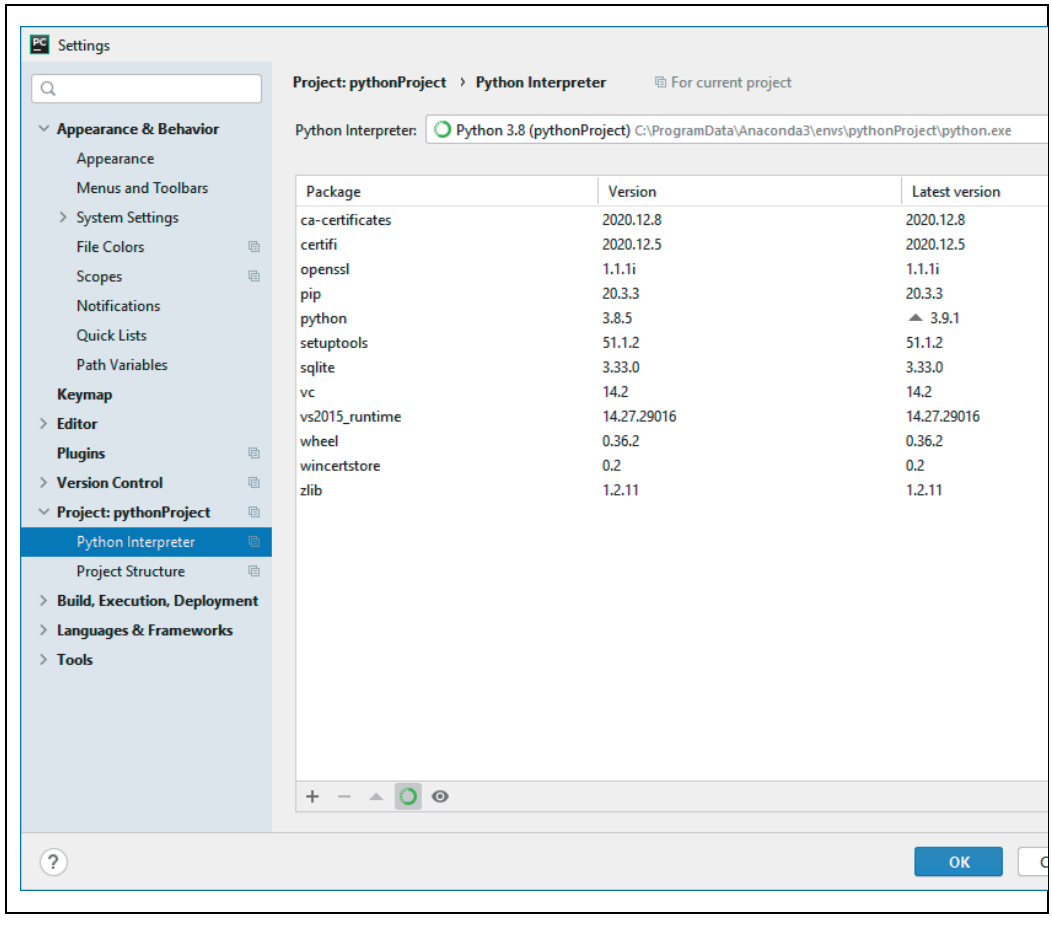
Rysunek 1.11. Wysokopoziomowa architektura platformy TensorFlow



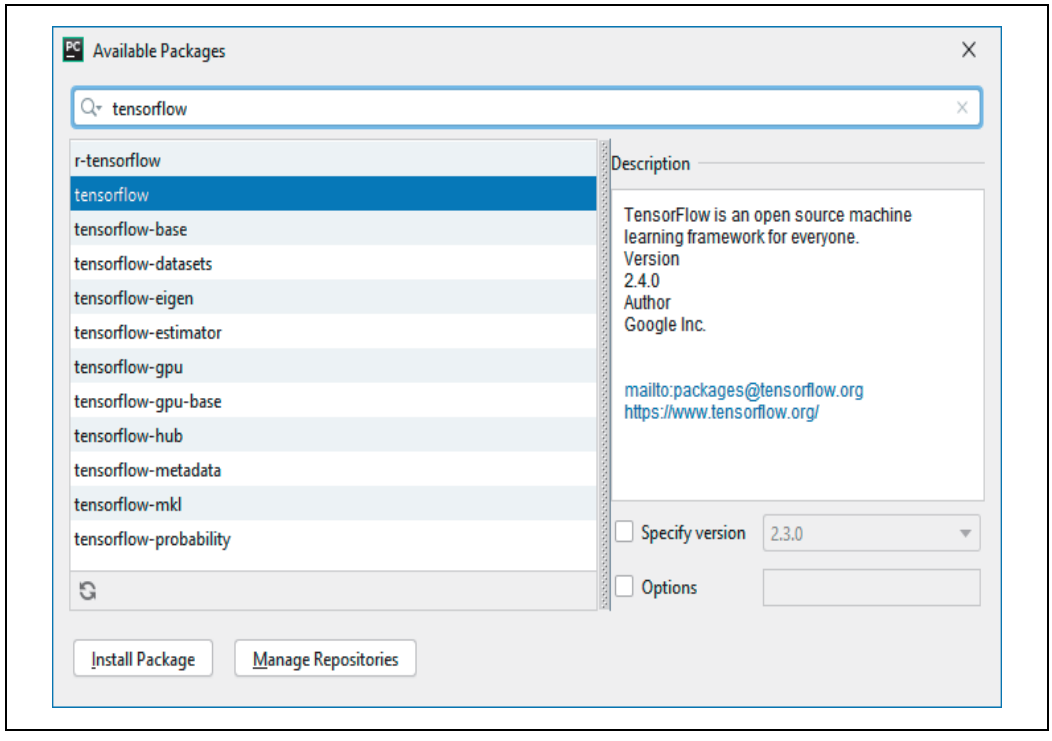
Rysunek 1.12. Uruchomienie biblioteki TensorFlow w środowisku języka Python



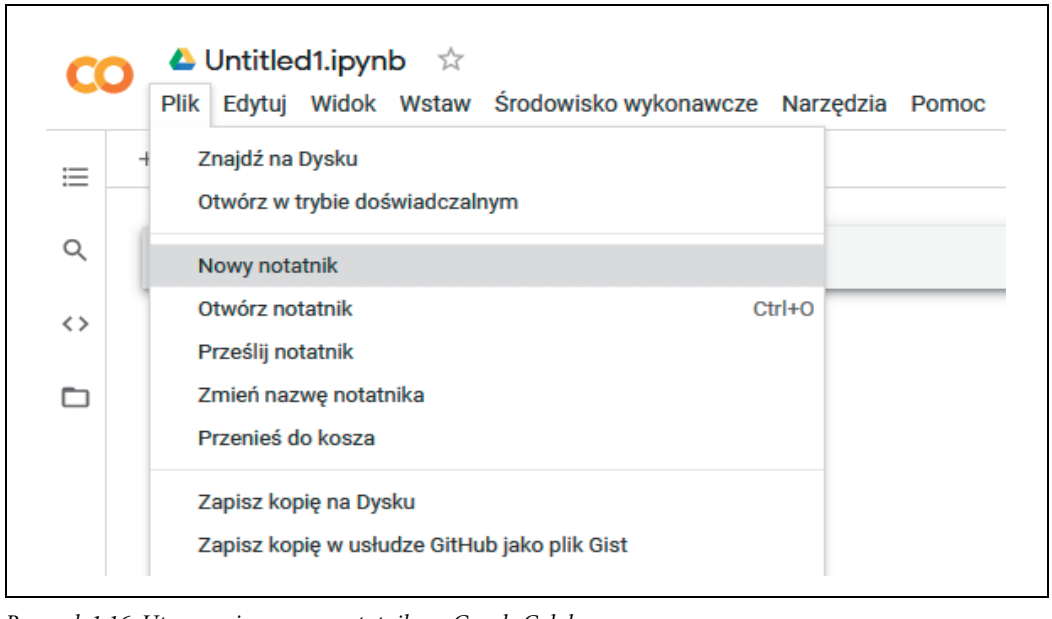
Rysunek 1.13. Tworzenie nowego wirtualnego środowiska z użyciem oprogramowania PyCharm



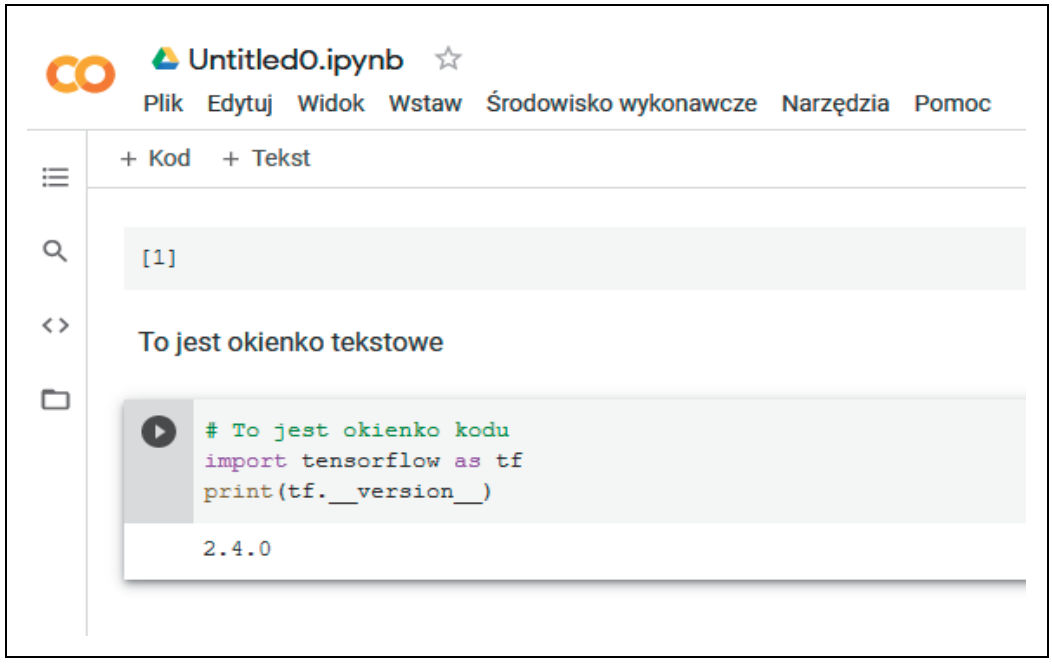
Rysunek 1.14. Dodawanie pakietów do środowiska wirtualnego



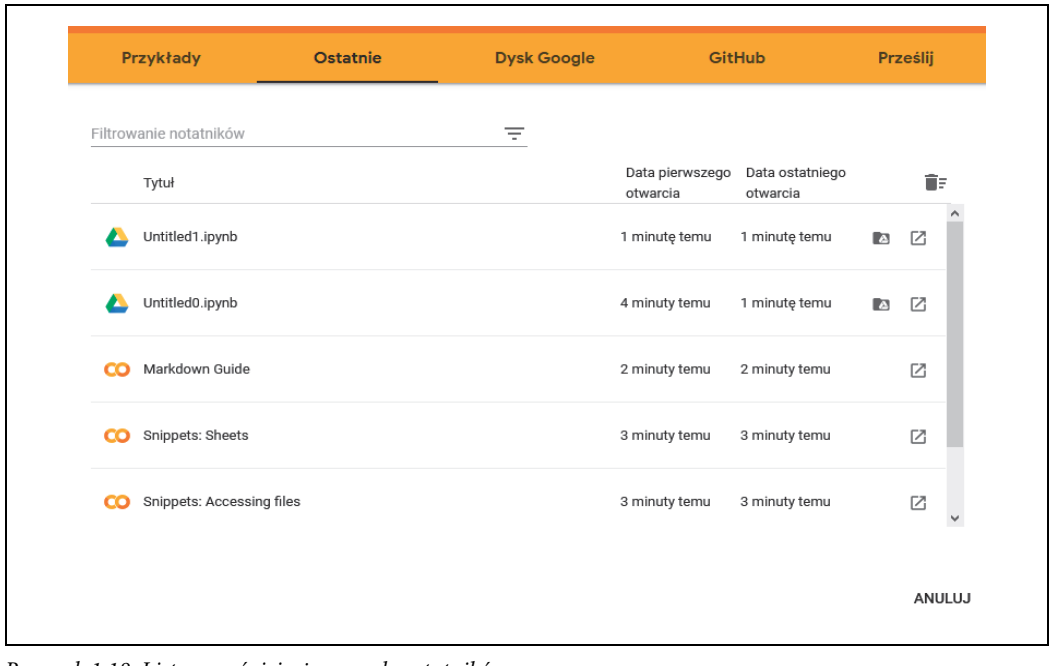
Rysunek 1.15. Instalowanie platformy TensorFlow za pomocą środowiska PyCharm



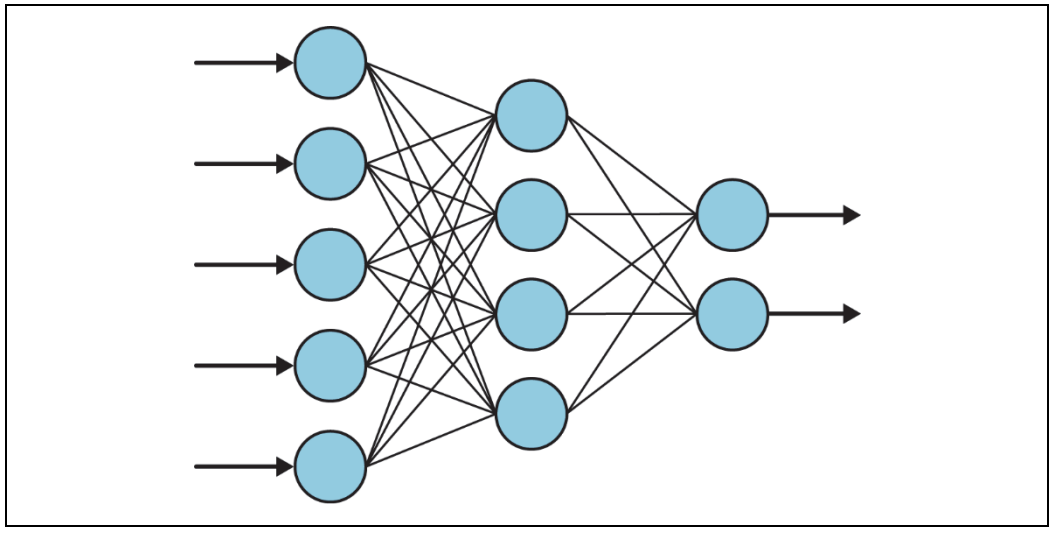
Rysunek 1.16. Utworzenie nowego notatnika w Google Colab



Rysunek 1.17. Uruchamianie kodu TensorFlow w Google Colab



Rysunek 1.18. Lista wcześniej używanych notatników



Rysunek 1.19. Typowa sieć neuronowa

```
Epoch 1/500
6/6 [==============================] - 9s 2s/sample - loss: 3.2868
Epoch 2/500
6/6 [==============================] - 0s 652us/sample - loss: 2.7447
Epoch 3/500
6/6 [==============================] - 0s 323us/sample - loss: 2.3150
Epoch 4/500
6/6 [==============================] - 0s 411us/sample - loss: 1.9737
Epoch 5/500
6/6 [==============================] - 0s 306us/sample - loss: 1.7021
Epoch 6/500
6/6 [==============================] - 0s 496us/sample - loss: 1.4853
Epoch 7/500
6/6 [==============================] - 0s 470us/sample - loss: 1.3117
Epoch 8/500
6/6 [==============================] - 0s 405us/sample - loss: 1.1723
Epoch 9/500
6/6 [==============================] - 0s 616us/sample - loss: 1.0596
Epoch 10/500
6/6 [==============================] - 0s 669us/sample - loss: 0.9682
```

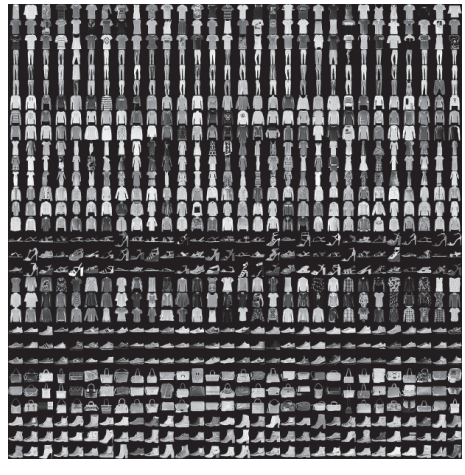
Rysunek 1.20. Trenowanie sieci neuronowej

```
Epoch 495/500
6/6 [==============================] - 0s 374us/sample - loss: 2.9063e-05
Epoch 496/500
6/6 [==============================] - 0s 540us/sample - loss: 2.8466e-05
Epoch 497/500
6/6 [==============================] - 0s 382us/sample - loss: 2.7882e-05
Epoch 498/500
6/6 [==============================] - 0s 397us/sample - loss: 2.7309e-05
Epoch 499/500
6/6 [==============================] - 0s 367us/sample - loss: 2.6748e-05
Epoch 500/500
6/6 [==============================] - 0s 363us/sample - loss: 2.6199e-05
```

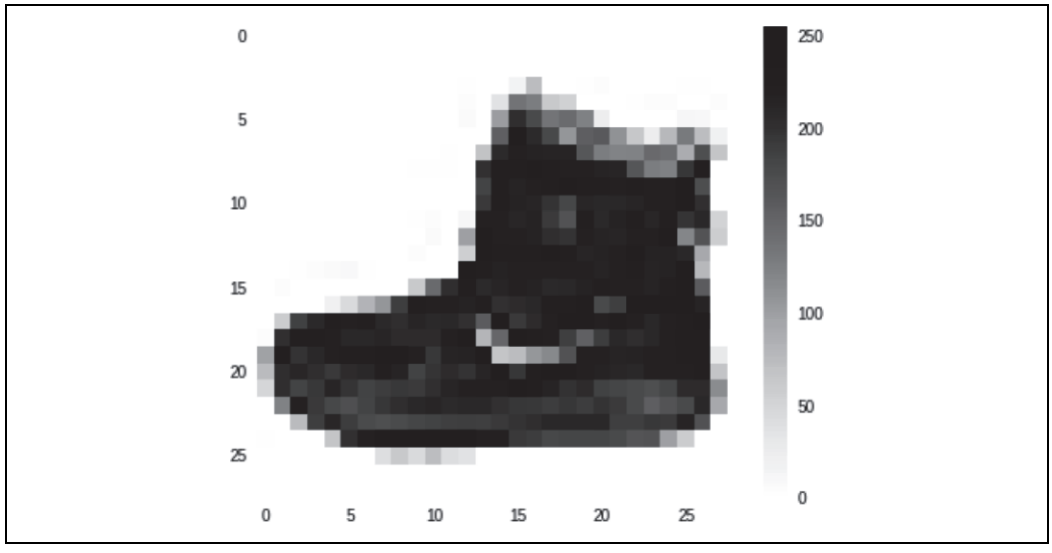
Rysunek 1.21. Trenowanie sieci neuronowej

-==========

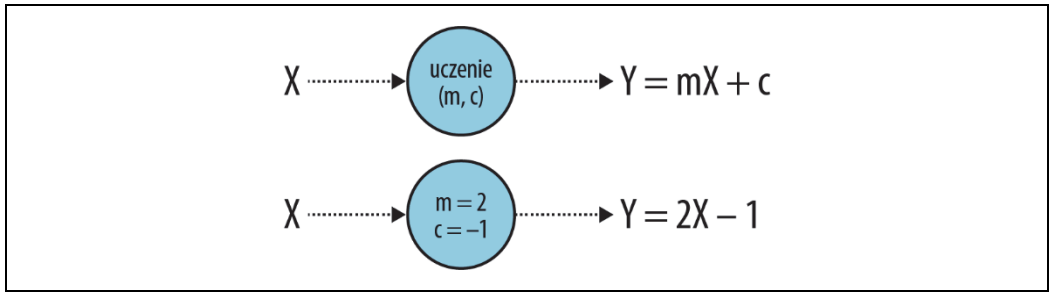
Rysunek 2.1. Przykłady ubrań, butów i dodatków



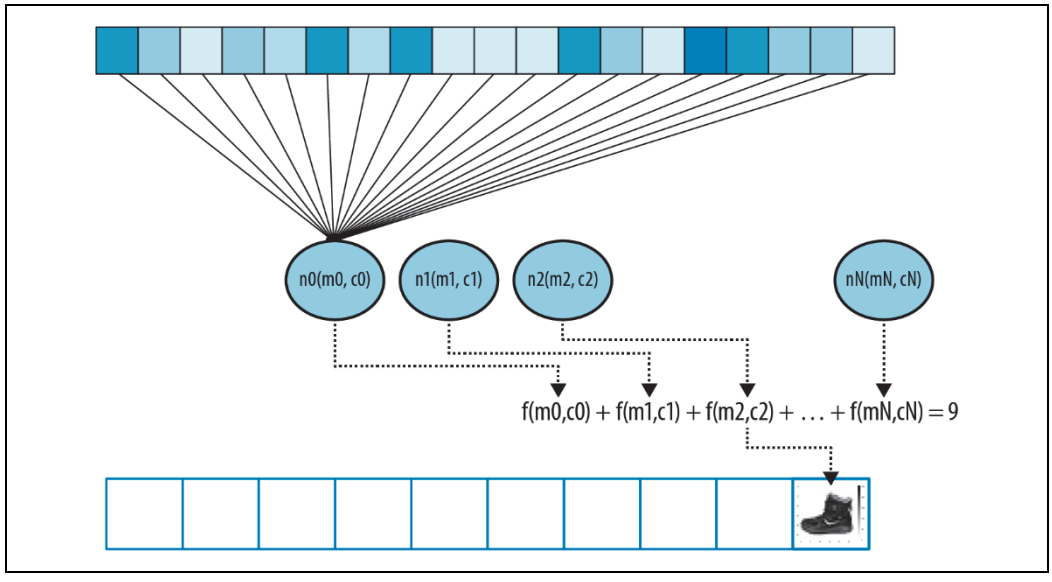
Rysunek 2.2. Przeglądanie zbioru Fashion MNIST



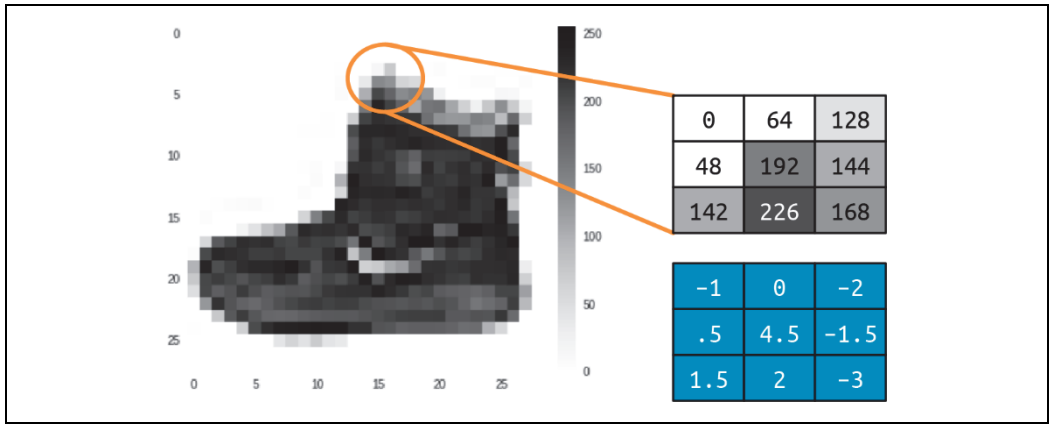
Rysunek 2.3. Powiększenie jednego z obrazów ze zbioru danych Fashion MNIST



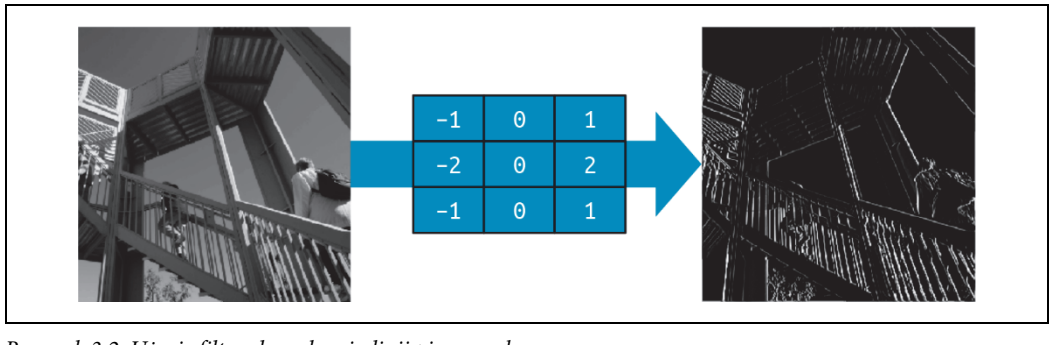
Rysunek 2.4. Pojedynczy neuron uczący się liniowej zależności



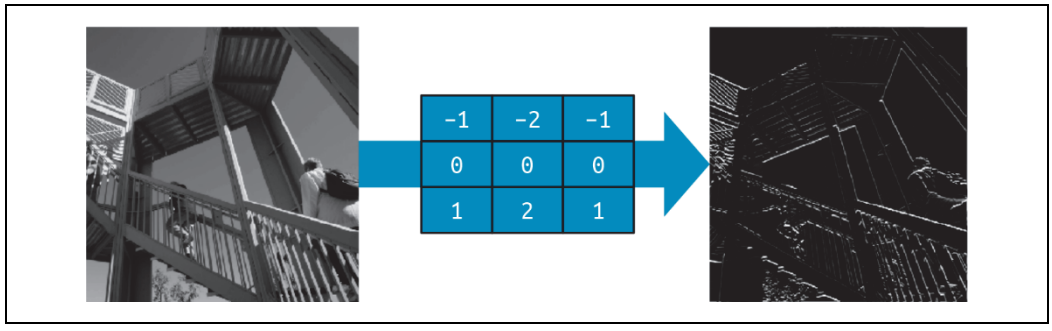
Rysunek 2.5. Wykorzystanie prostego wzorca w bardziej złożonym przykładzie



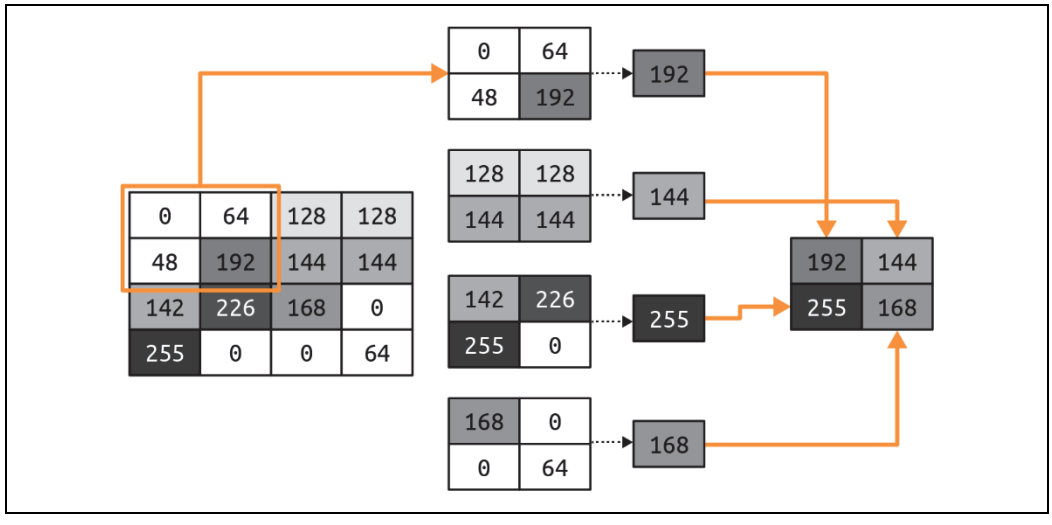
Rysunek 3.1. Obraz buta za kostkę z filtrem konwolucyjnym



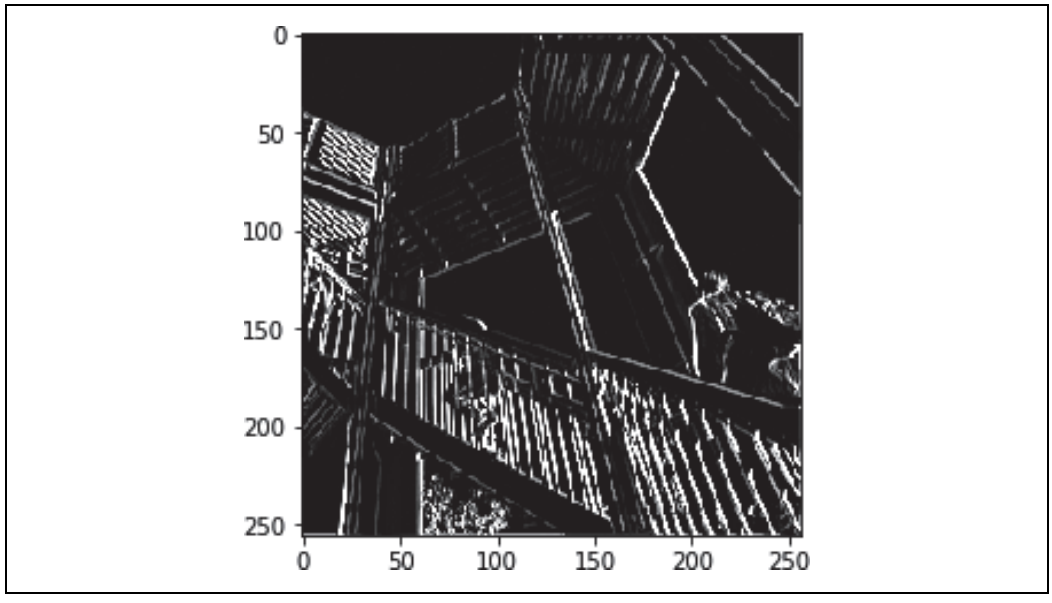
Rysunek 3.2. Użycie filtru do wykrycia linii pionowych



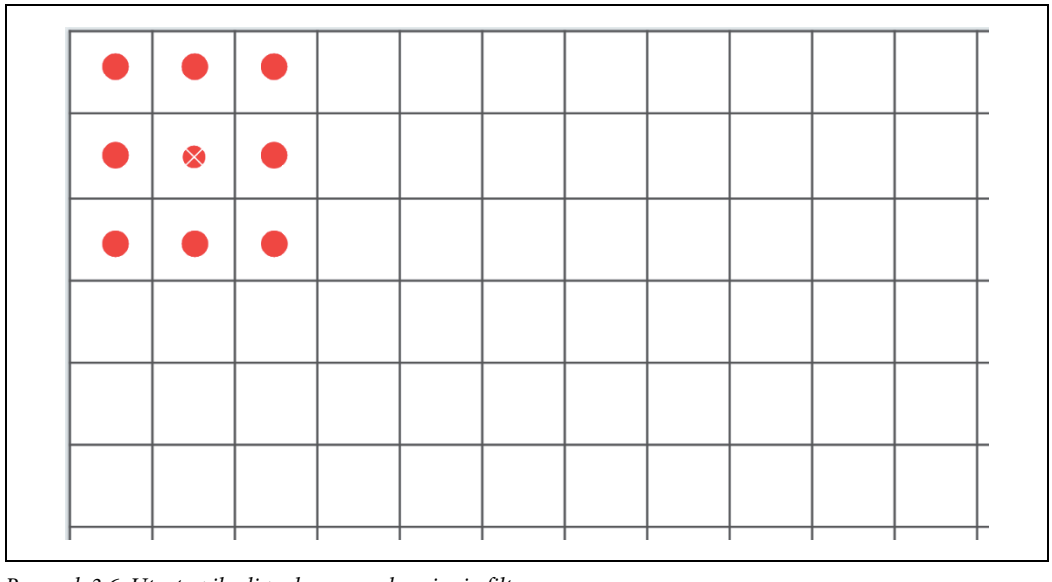
Rysunek 3.3. Użycie filtru do wykrycia linii poziomych



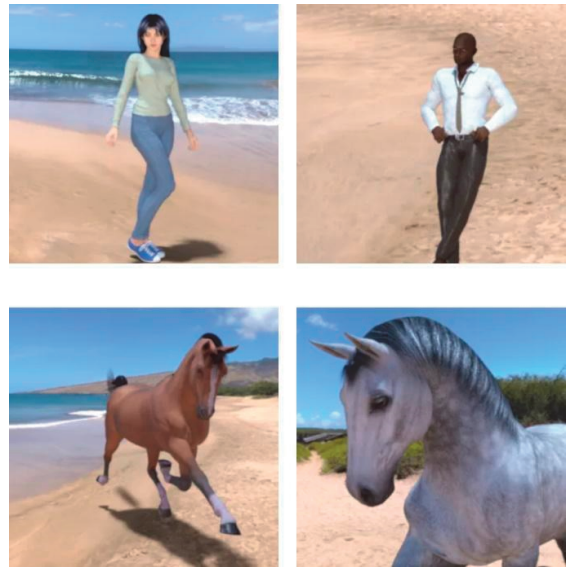
Rysunek 3.4. Działanie operacji max pooling



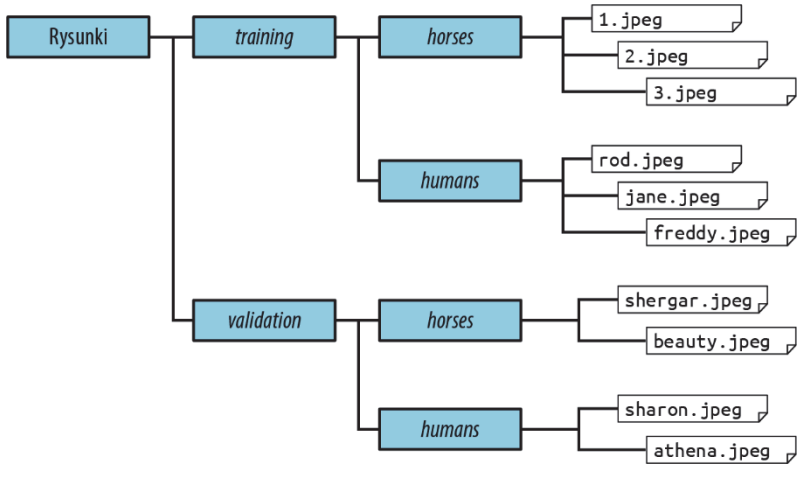
Rysunek 3.5. Obraz (przedstawiający wchodzenie po schodach) po zastosowaniu filtru linii pionowych oraz operacji max pooling



Rysunek 3.6. Utrata pikseli podczas uruchamiania filtru



Rysunek 3.7. Ludzie i konie



Rysunek 3.8. Obrazy muszą się znaleźć w nazwanych podkatalogach

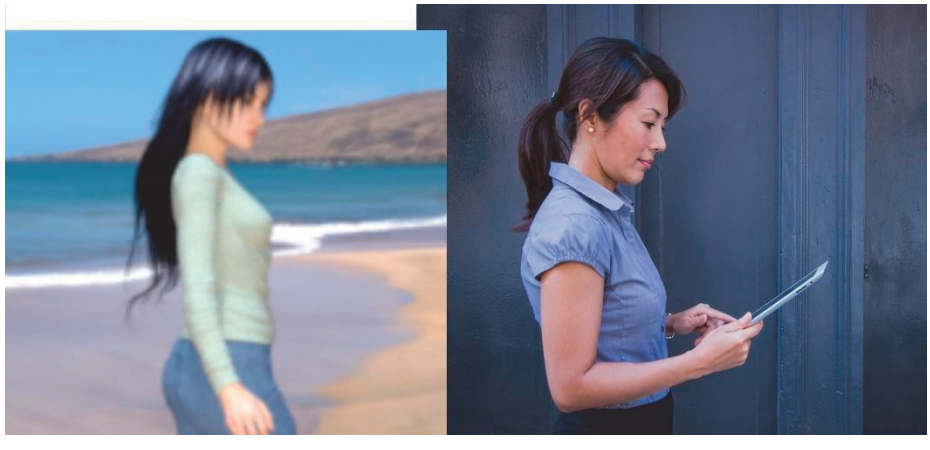
Rysunek 3.9. Obrazy testowe

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

  # prognozowanie obrazów
  path = '/content/' + fn
  img = image.load_img(path, target_size=(300, 300))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)
  print(classes)
  print(classes[0])
  if classes[0]>0.5:
    print(fn + " przedstawia człowieka")
  else:
    print(fn + " przedstawia konia")
```

```
Przeglądaj...  Wybranych plików: 3.
beautiful-1274056_640.jpg(image/jpeg) - 97343 bytes, last modified: n/a - 100% done
horse-1330690_640.jpg(image/jpeg) - 225580 bytes, last modified: n/a - 100% done
model-993907_640.jpg(image/jpeg) - 637884 bytes, last modified: n/a - 100% done
Saving beautiful-1274056_640.jpg to beautiful-1274056_640 (2).jpg
Saving horse-1330690_640.jpg to horse-1330690_640 (2).jpg
Saving model-993907_640.jpg to model-993907_640 (2).jpg
[[0.]]
[0.]
beautiful-1274056_640.jpg przedstawia konia
[[0.]]
[0.]
horse-1330690_640.jpg przedstawia konia
[[1.]]
[1.]
model-993907_640.jpg przedstawia człowieka
```

Rysunek 3.10. Uruchomienie modelu

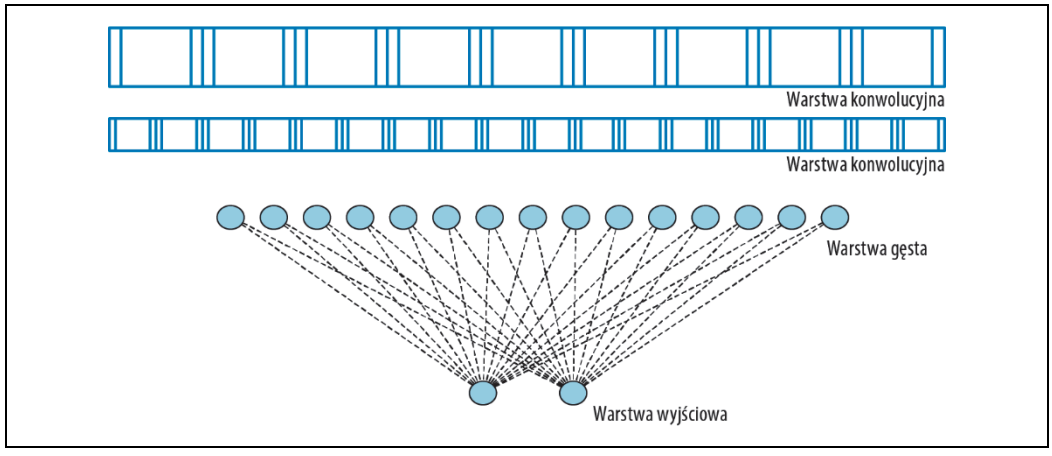
Rysunek 3.11. Podobieństwo do obrazów ze zbioru danych



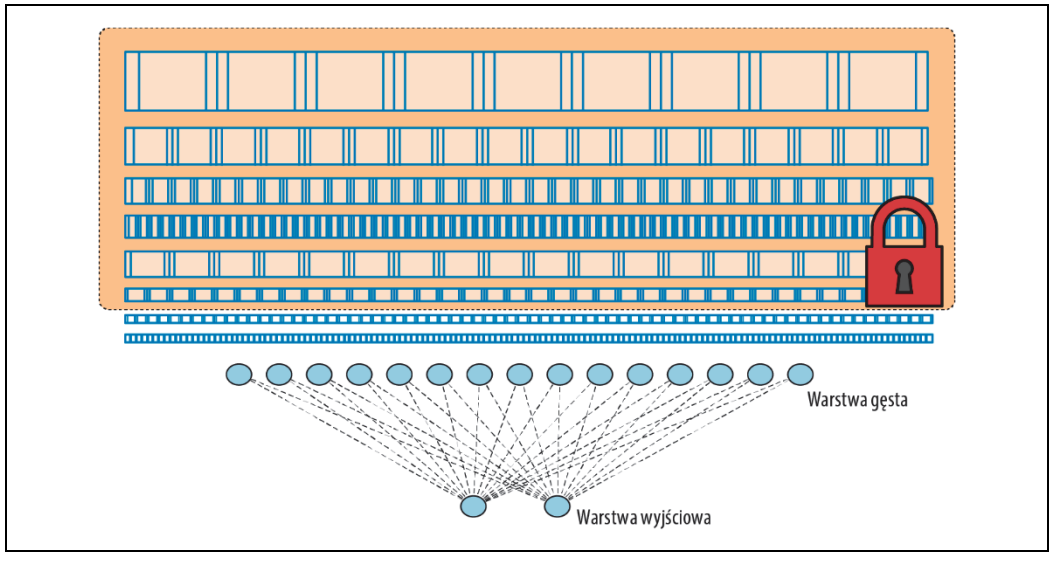
Rysunek 3.12. Powiększenie obrazu ze zbioru treningowego

```
Przeglądaj... beautiful-1274056_640.jpg
beautiful-1274056_640.jpg(image/jpeg) - 97343 bytes, last modified: n/a - 100% done
Saving beautiful-1274056_640.jpg to beautiful-1274056_640 (3).jpg
[[1.]]
[1.]
beautiful-1274056_640.jpg przedstawia człowieka
```

Rysunek 3.13. Zdjęcie kobiety zostało tym razem poprawnie sklasyfikowane



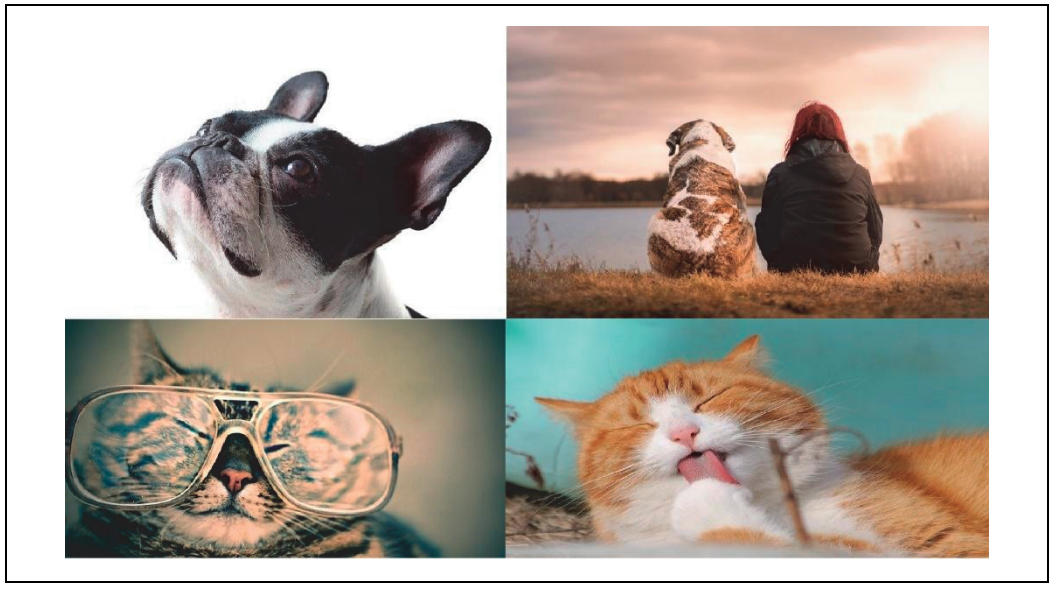
Rysunek 3.14. Architektura konwolucyjnej sieci neuronowej



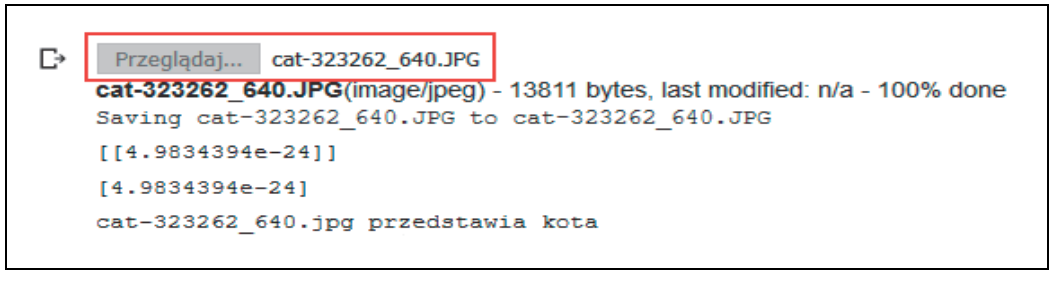
Rysunek 3.15. Pobieranie warstw z innej architektury poprzez uczenie transferowe

```
52/52 [==============================] - 11s 210ms/step - loss: 0.0149 - acc: 0.9961 - val_loss: 0.3838 - val_acc: 0.9669
Epoch 38/40
52/52 [==============================] - 11s 211ms/step - loss: 0.0088 - acc: 0.9981 - val_loss: 0.2245 - val_acc: 0.9727
Epoch 39/40
52/52 [==============================] - 11s 210ms/step - loss: 0.0088 - acc: 0.9971 - val_loss: 0.2072 - val_acc: 0.9727
Epoch 40/40
52/52 [==============================] - 11s 219ms/step - loss: 0.0118 - acc: 0.9971 - val_loss: 0.6150 - val_acc: 0.9609
```

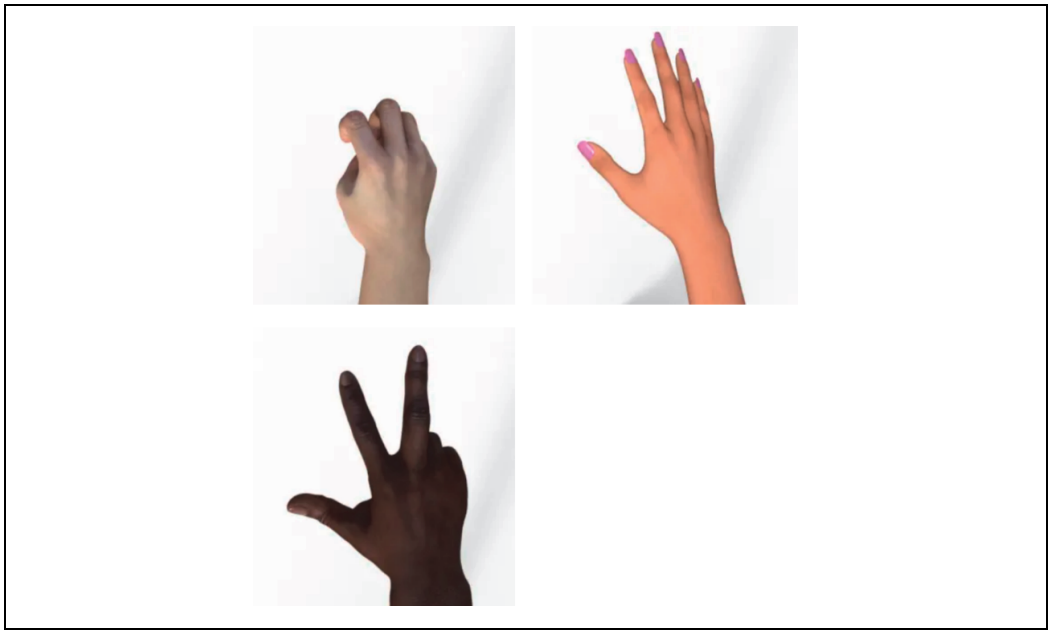
Rysunek 3.16. Trenowanie za pomocą uczenia transferowego klasyfikatora rozróżniającego konie i ludzi



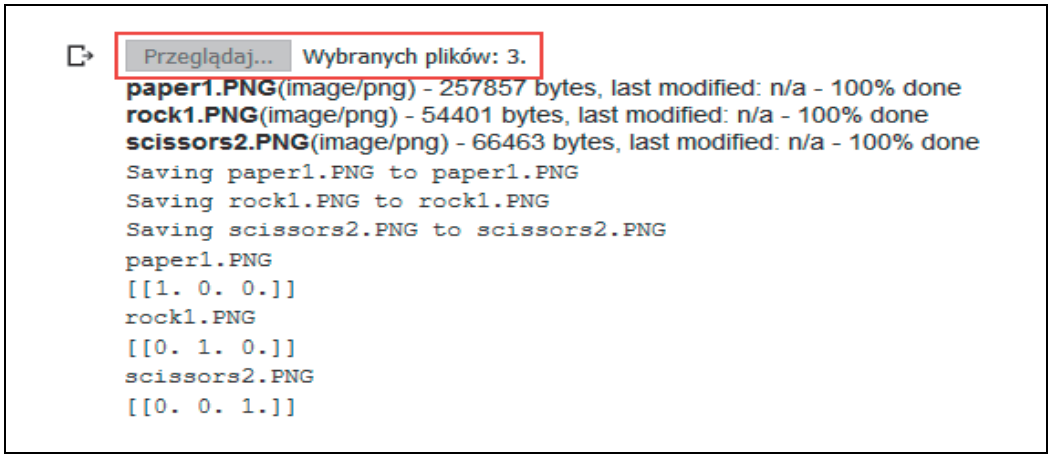
Rysunek 3.17. Nietypowe zdjęcia psów i kotów, które zostały poprawnie zaklasyfikowane



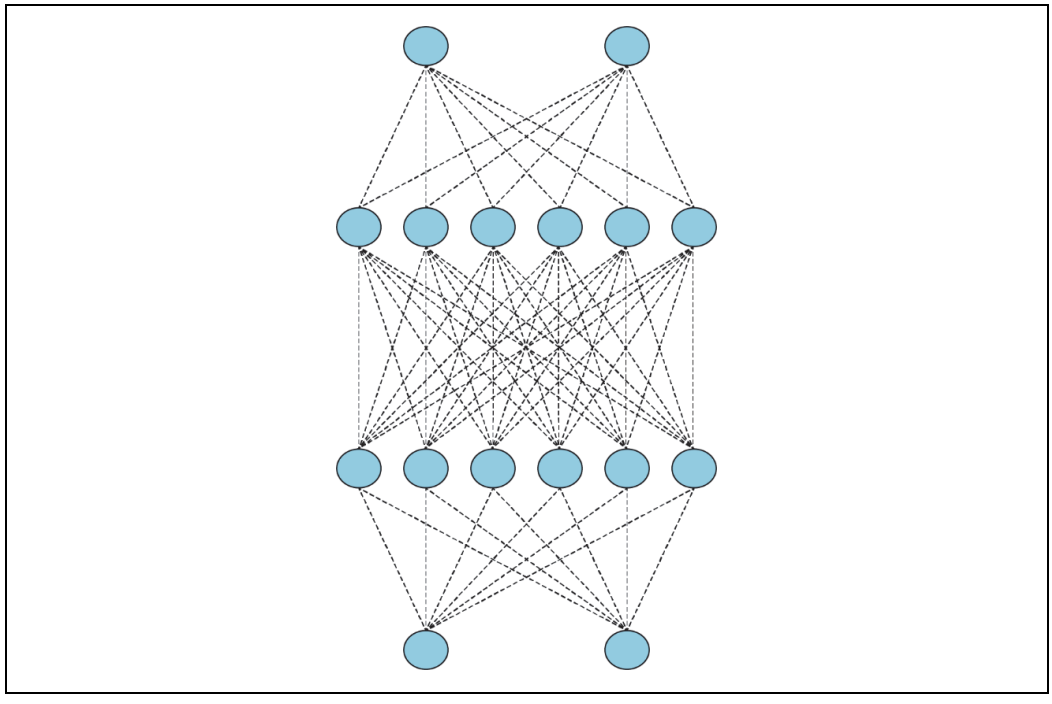
Rysunek 3.18. Klasyfikowanie kota liżącego łapę



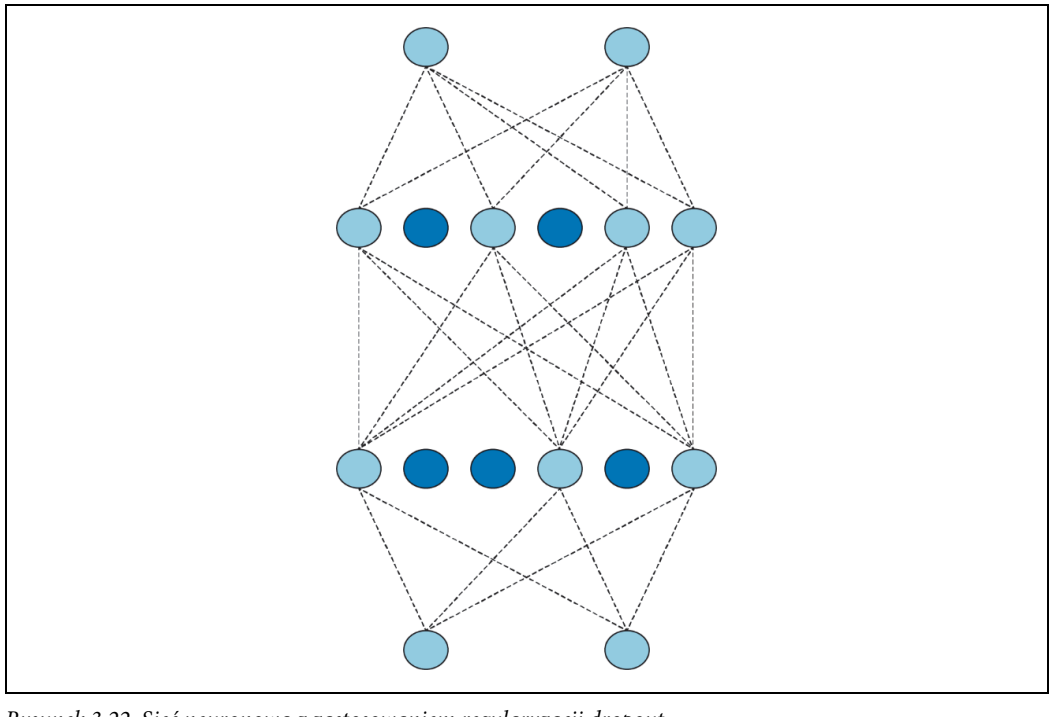
Rysunek 3.19. Przykłady gestów stosowanych w grze „kamień, papier, nożyce”



Rysunek 3.20. Testowanie klasyfikatora dla gry „kamień, papier, nożyce”



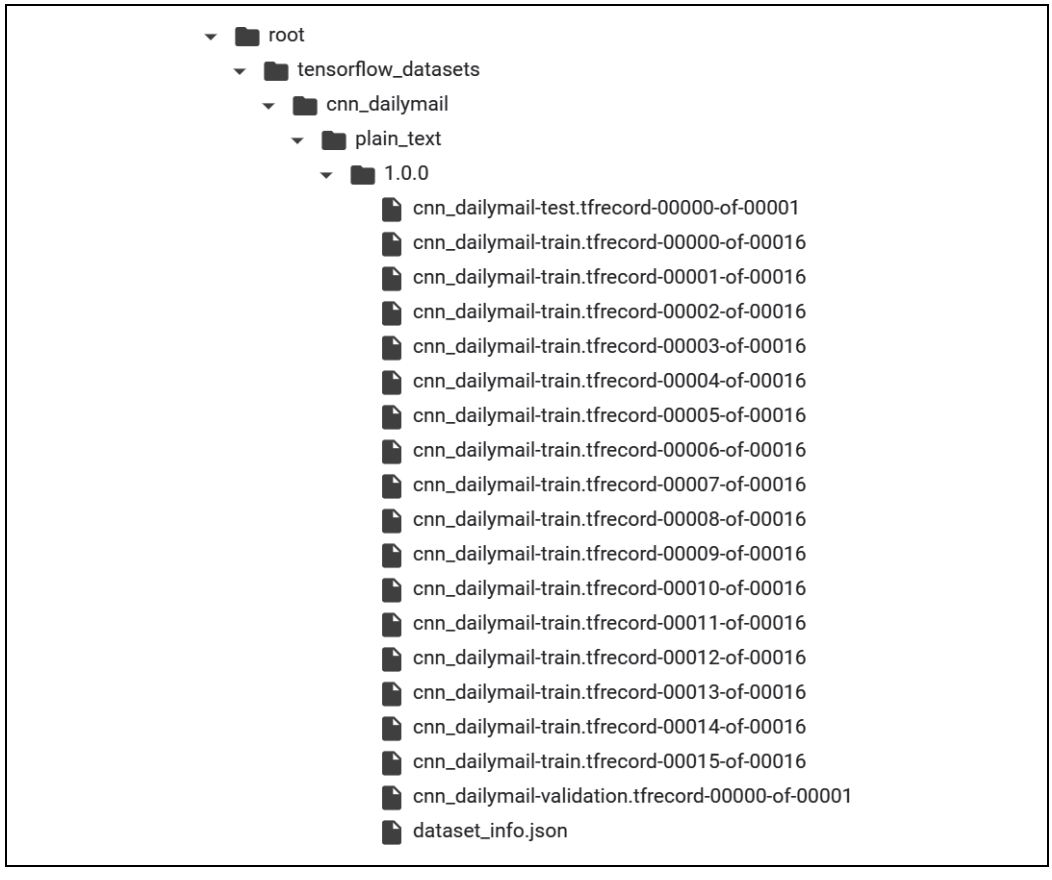
Rysunek 3.21. Prosta sieć neuronowa



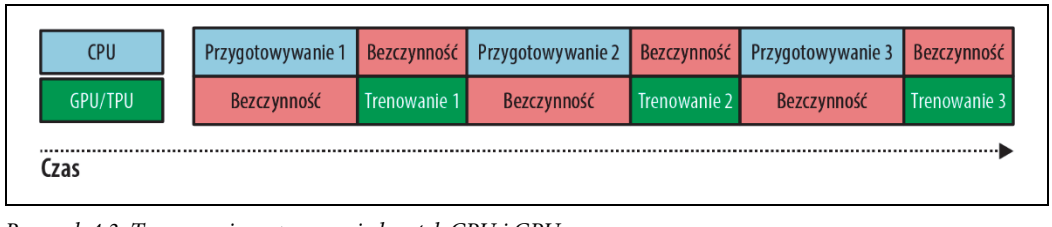
Rysunek 3.22. Sieć neuronowa z zastosowaniem regularyzacji dropout

```
/tensorflow_datasets/cnn_dailymail/plain_text/1.0.0.incompleteFFCKFG/cnn_dailymail-train.tfrecord
99% 282918/287113 [00:06<00:00, 48503.77 examples/s]
10153 examples [00:07, 1475.51 examples/s]
```

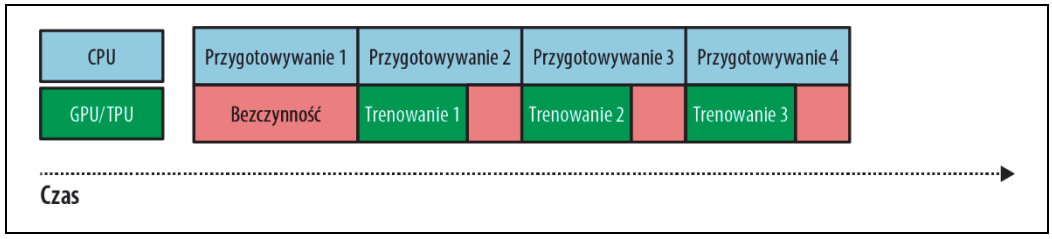
Rysunek 4.1. Pobieranie zbioru danych jako pliku TFRecord



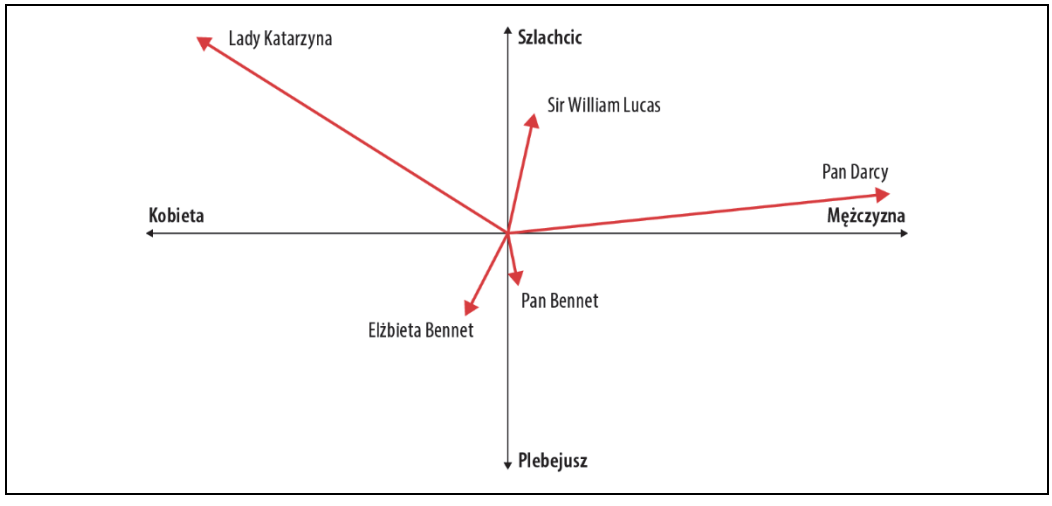
Rysunek 4.2. Analiza plików TFRecord w przypadku zbioru danych cnn_dailymail



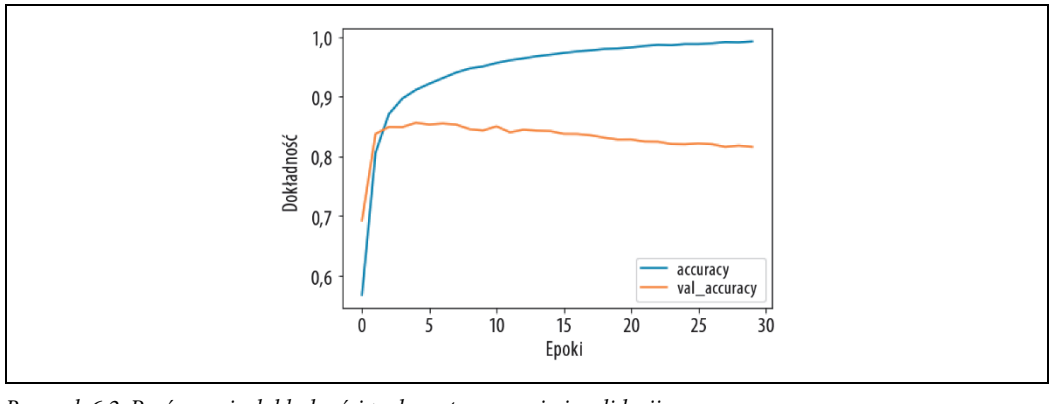
Rysunek 4.3. Trenowanie za pomocą jednostek CPU i GPU



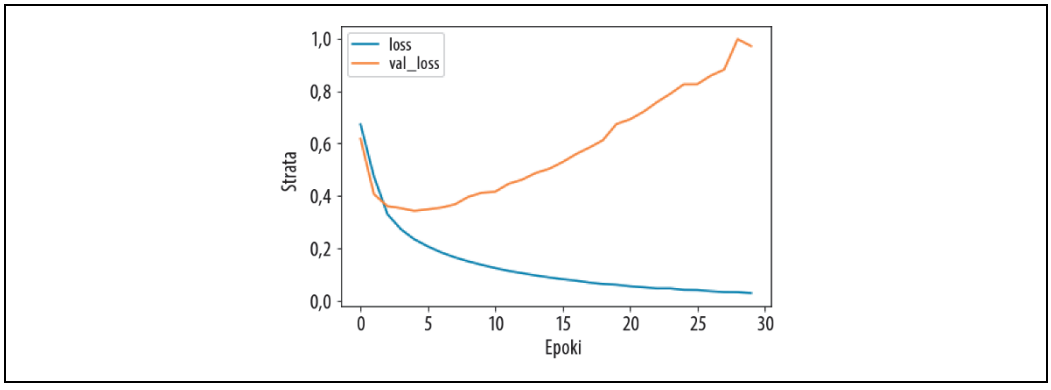
Rysunek 4.4. Potokowość



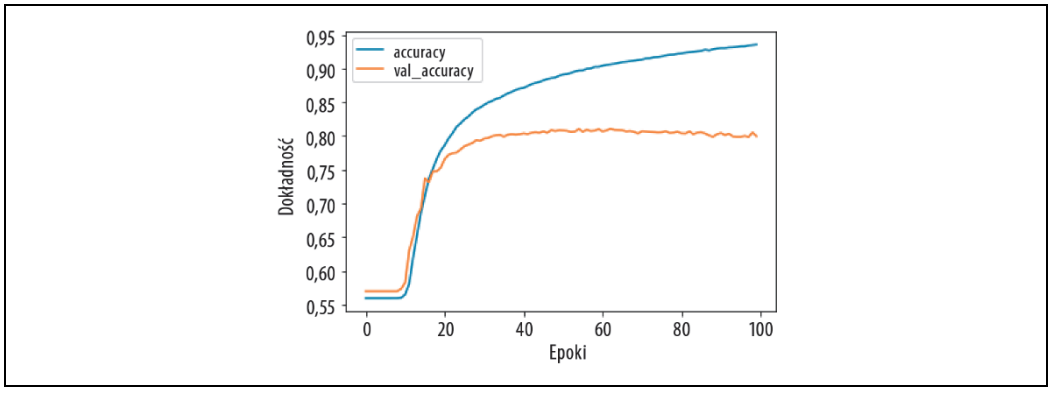
Rysunek 6.1. Postacie z książki Duma i uprzedzenie odwzorowane za pomocą wektorów



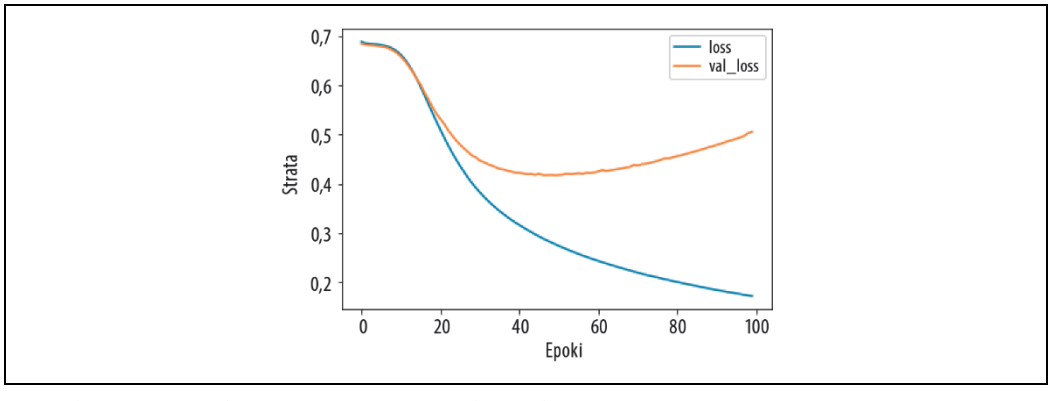
Rysunek 6.2. Porównanie dokładności podczas trenowania i walidacji



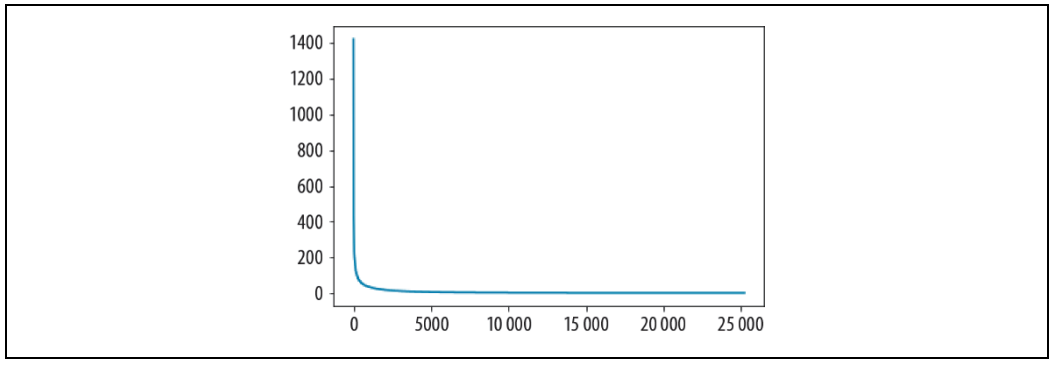
Rysunek 6.3. Porównanie straty podczas trenowania i walidacji



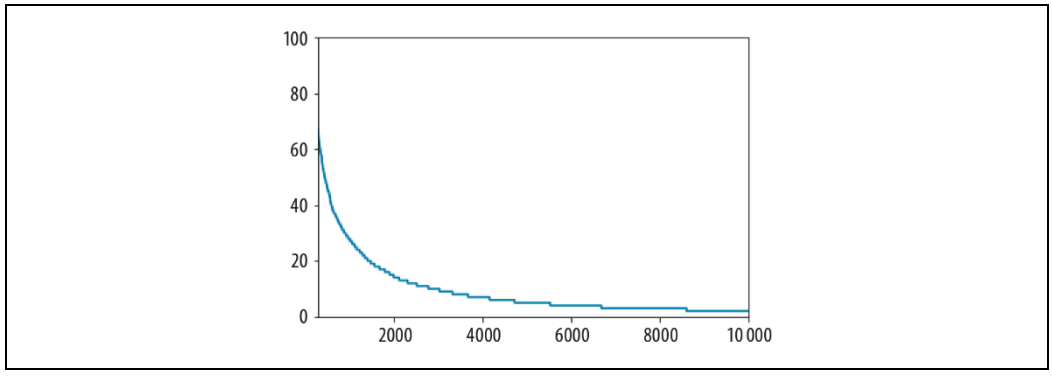
Rysunek 6.4. Dokładność podczas użycia niższego współczynnika uczenia



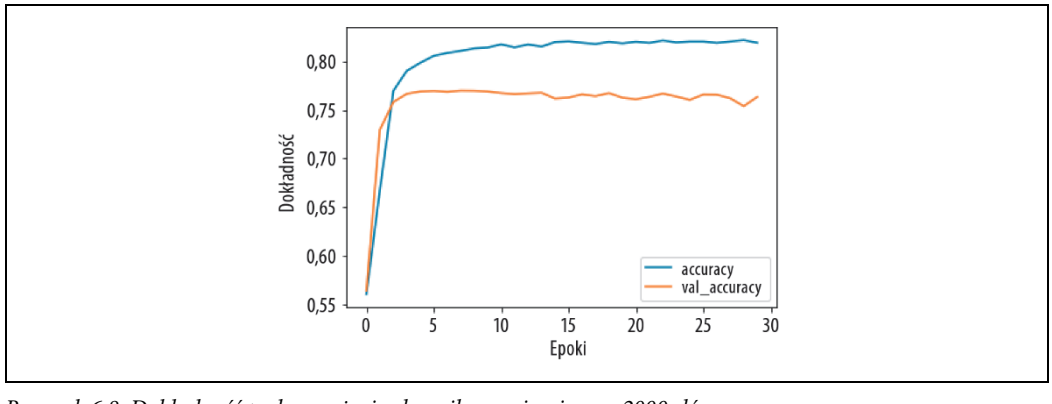
Rysunek 6.5. Strata podczas użycia niższego współczynnika uczenia



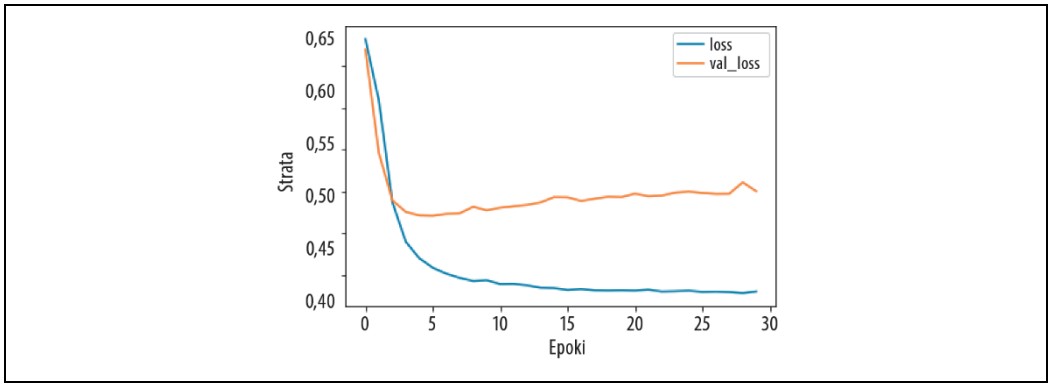
Rysunek 6.6. Wykres częstości występowania słów



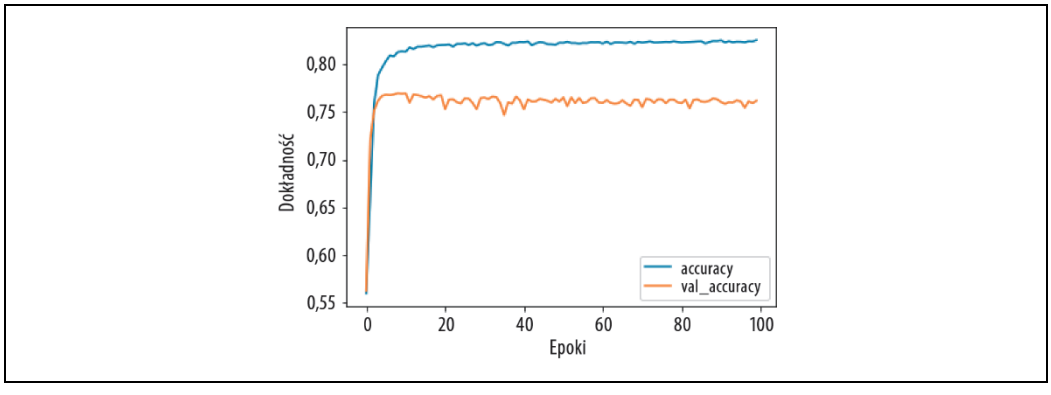
Rysunek 6.7. Wykres częstości występowania słów z zakresu od 300 do 10 000



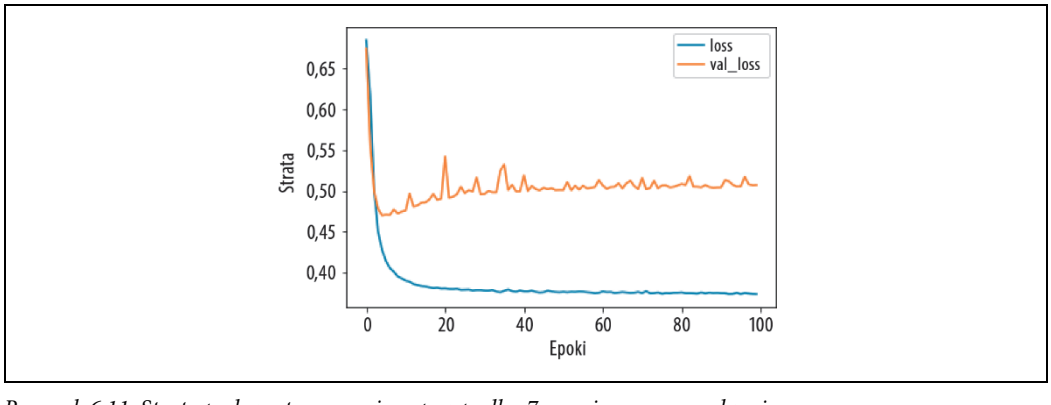
Rysunek 6.8. Dokładność podczas użycia słownika zawierającego 2000 słów



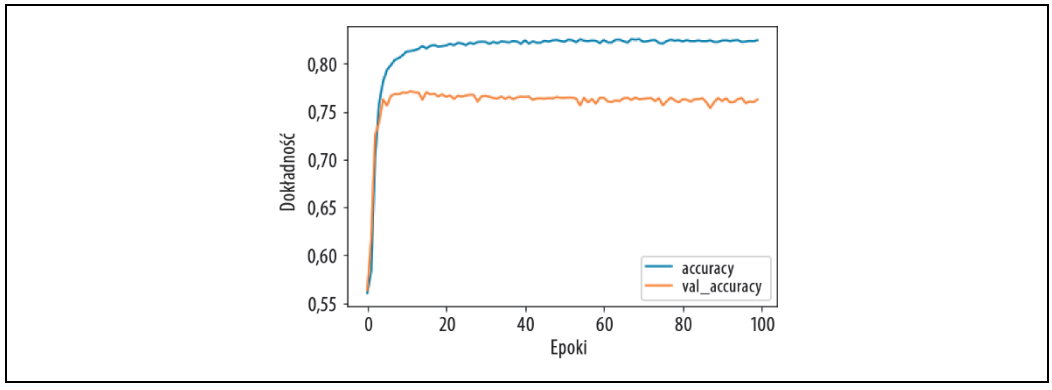
Rysunek 6.9. Strata podczas użycia słownika zawierającego 2000 słów



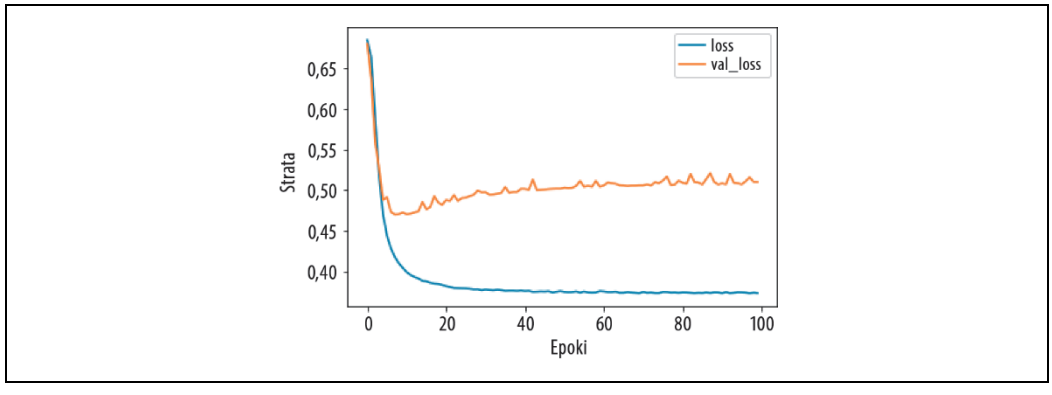
Rysunek 6.10. Dokładność podczas trenowania w przypadku 7-wymiarowego osadzania



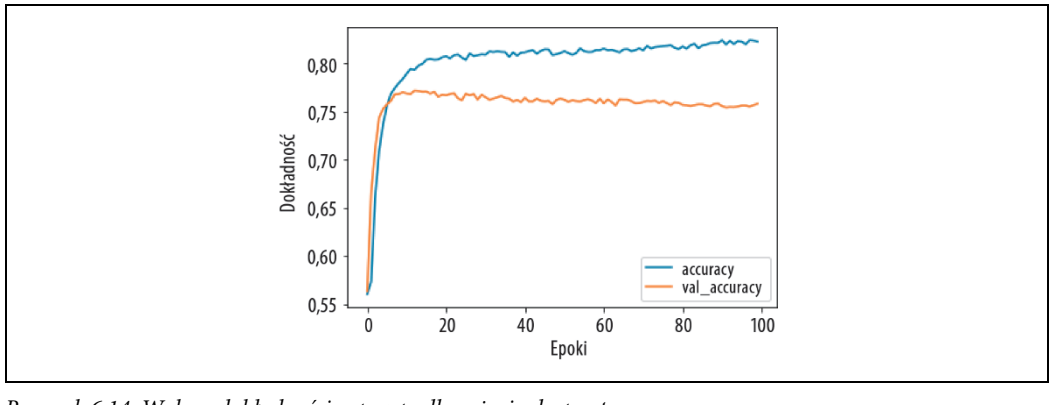
Rysunek 6.11. Strata podczas trenowania w przypadku 7-wymiarowego osadzania



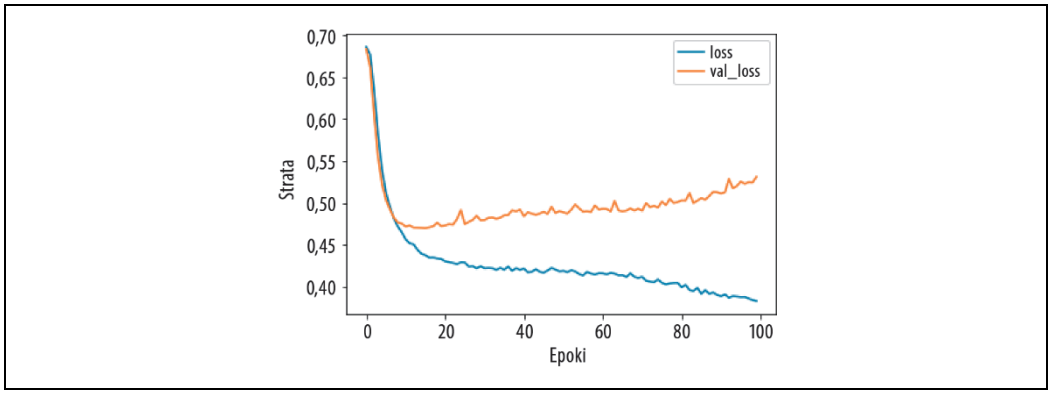
Rysunek 6.12. Wykres dokładności dla modelu, w którym zmniejszono liczbę neuronów



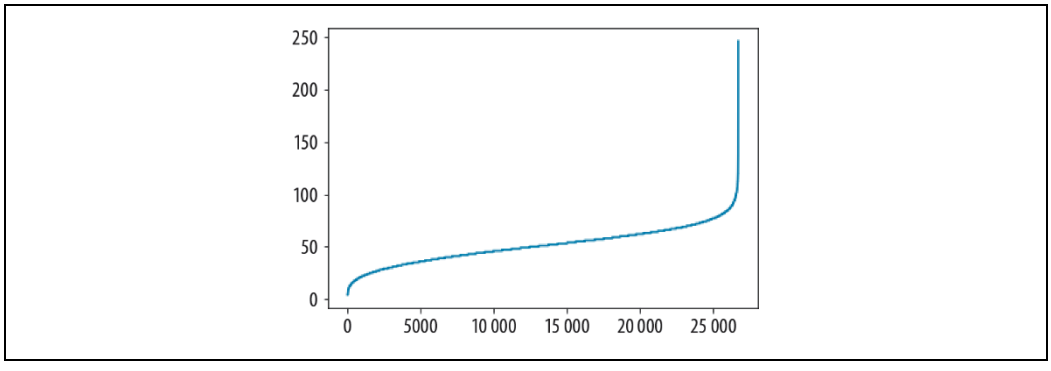
Rysunek 6.13. Wykres straty dla modelu, w którym zmniejszono liczbę neuronów



Rysunek 6.14. Wykres dokładności w przypadku użycia dropoutu

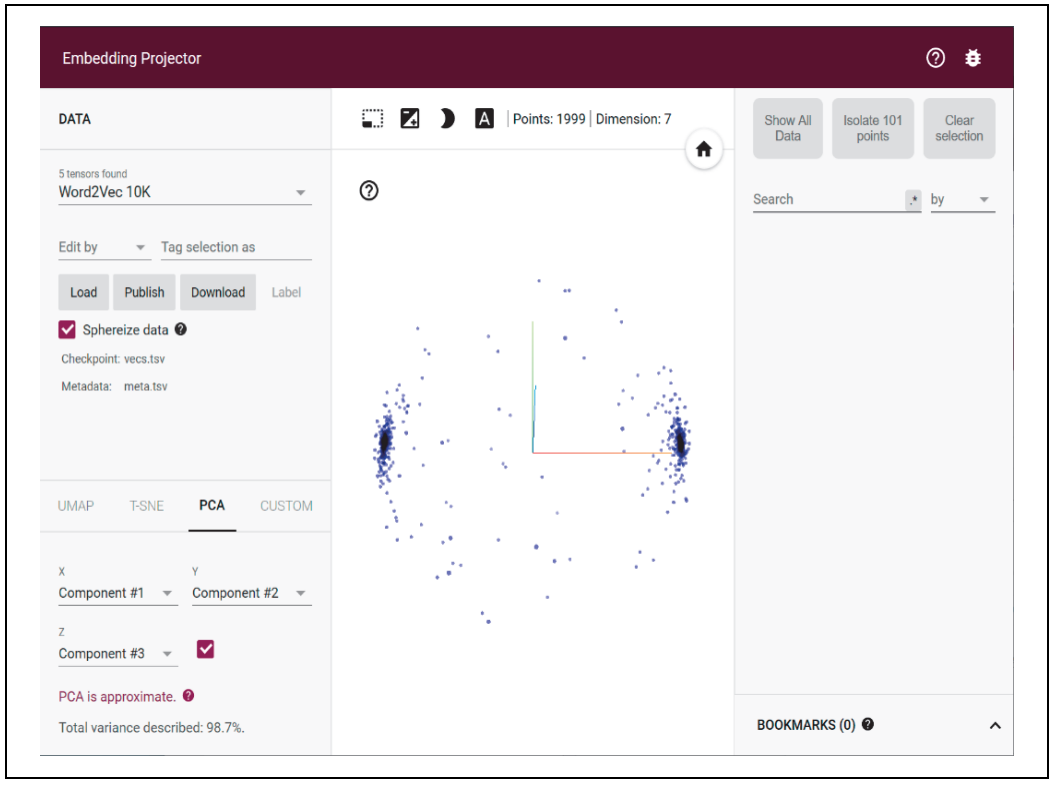


Rysunek 6.15. Wykres straty w przypadku użycia dropoutu

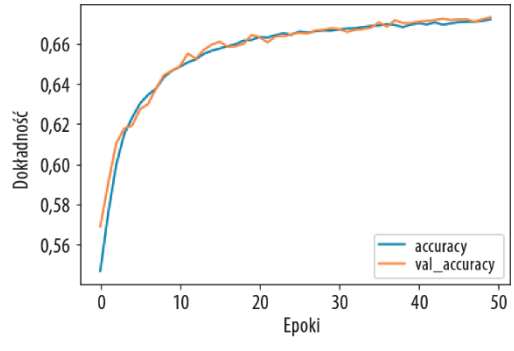


Rysunek 6.16. Wykres długości zdań

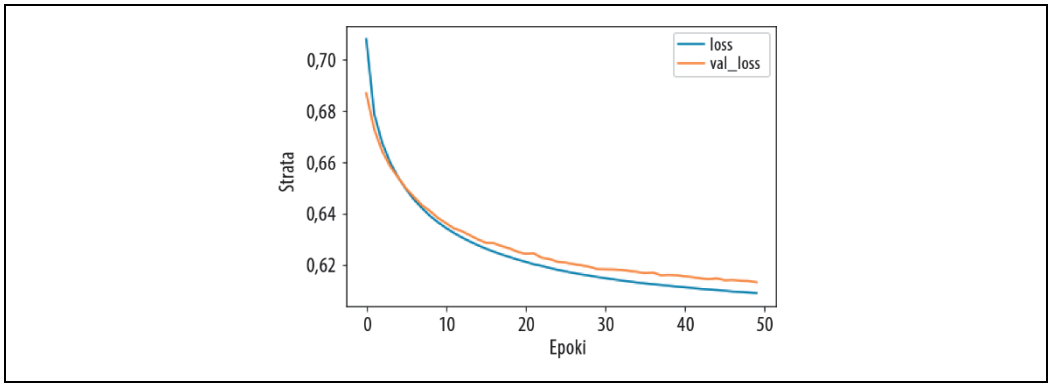
Rysunek 6.17. Użycie narzędzia Embedding Projector



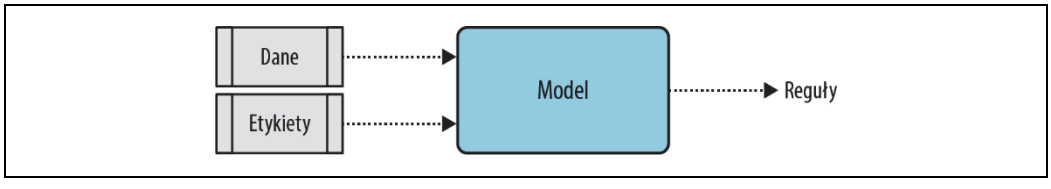
Rysunek 6.18. Wizualizacja osadzeń dla zdań sarkastycznych i normalnych



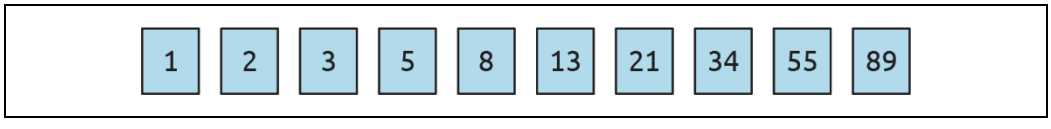
Rysunek 6.19. Dokładność podczas użycia osadzeń ze zbioru Swivel



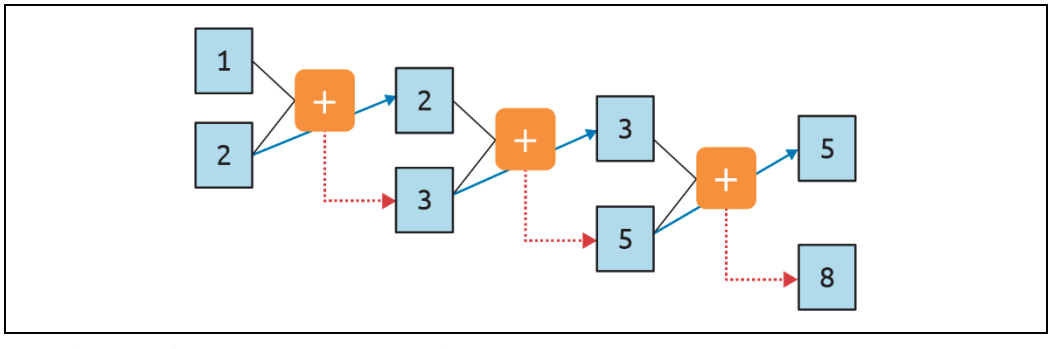
Rysunek 6.20. Strata podczas użycia osadzeń ze zbioru Swivel



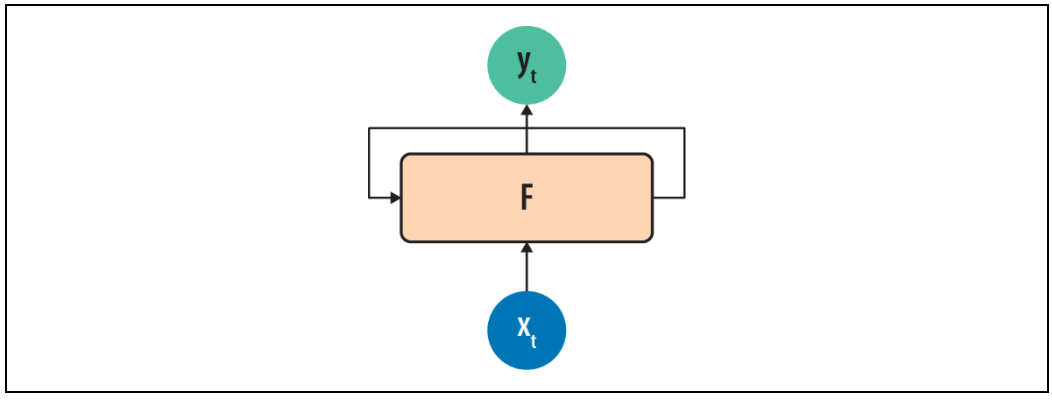
Rysunek 7.1. Wysokopoziomowy schemat tworzenia modelu



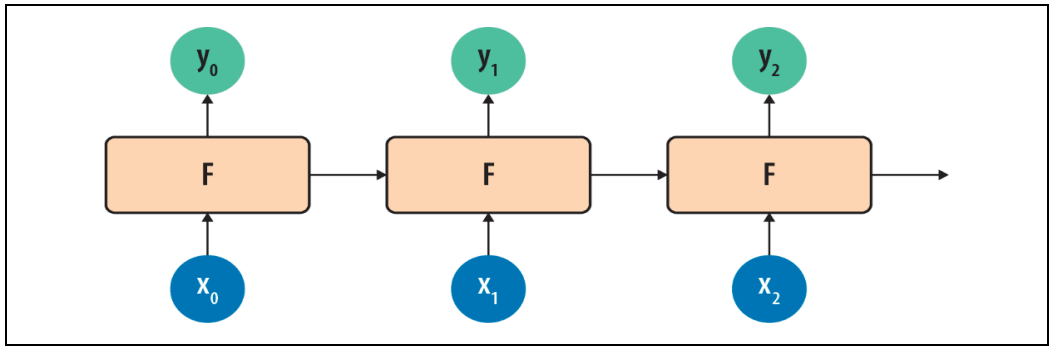
Rysunek 7.2. Kilka pierwszych liczb ciągu Fibonacciego



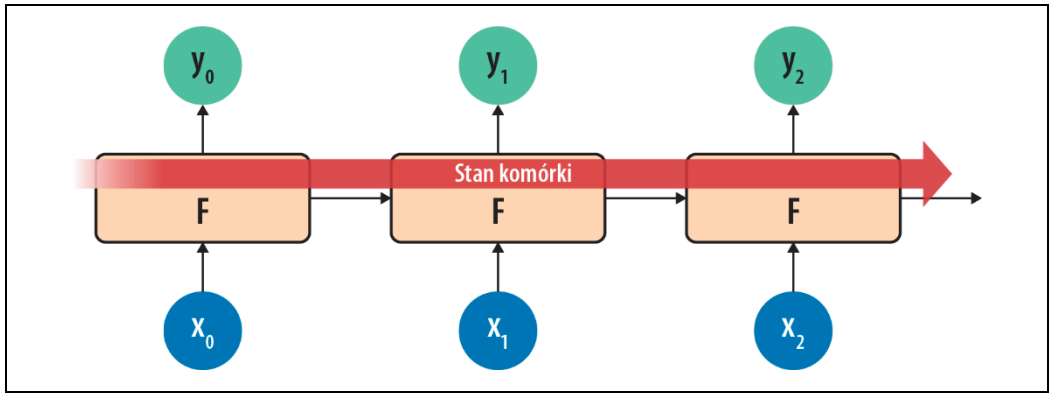
Rysunek 7.3. Graficzna reprezentacja ciągu Fibonacciego



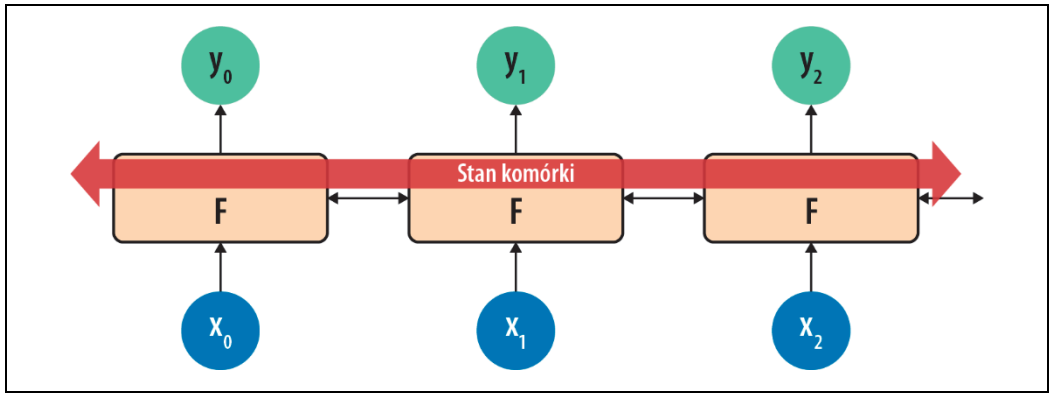
Rysunek 7.4. Neuron rekurencyjny



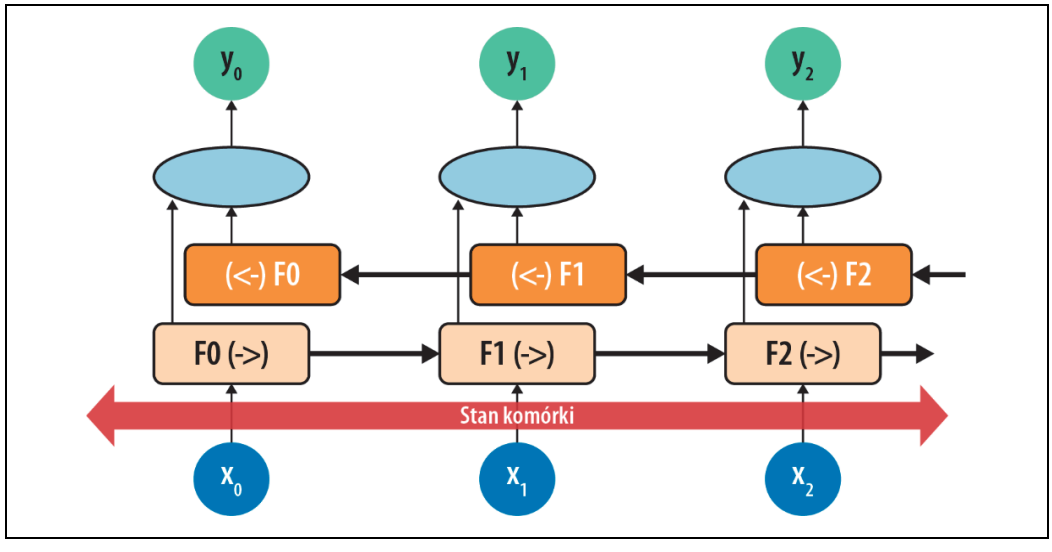
Rysunek 7.5. Działanie neuronów rekurencyjnych w kolejnych przedziałach czasowych



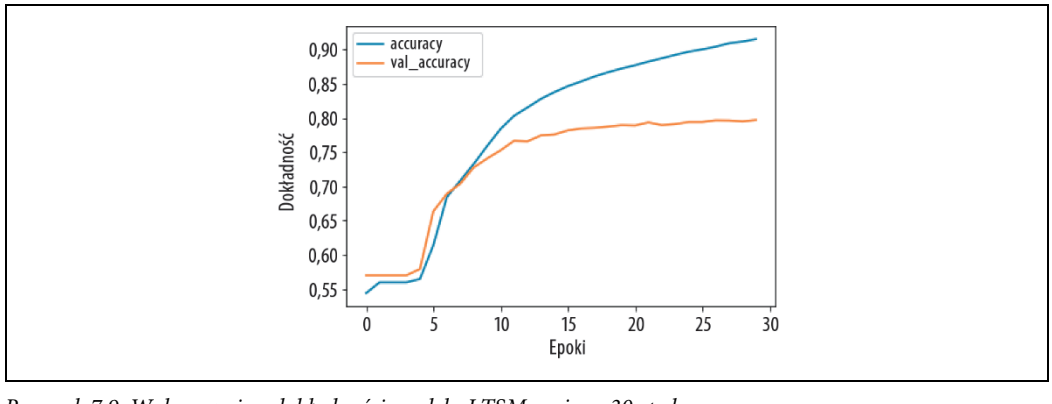
Rysunek 7.6. Wysokopoziomowy schemat architektury LSTM



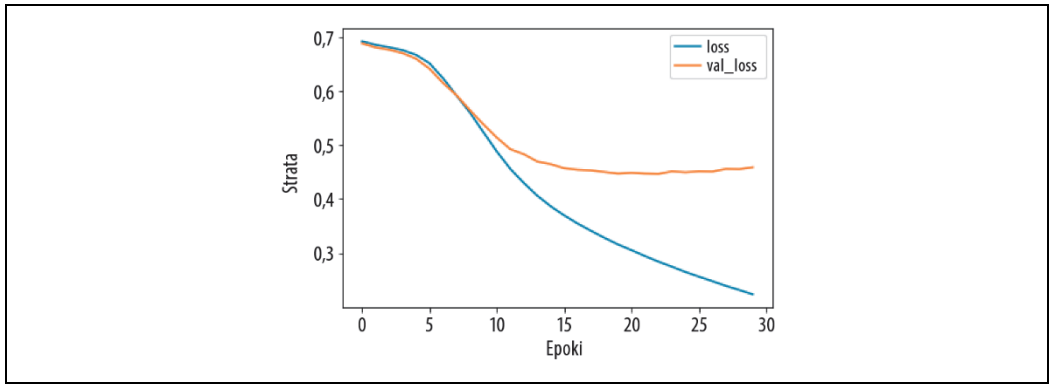
Rysunek 7.7. Wysokopoziomowy schemat dwukierunkowej architektury LSTM



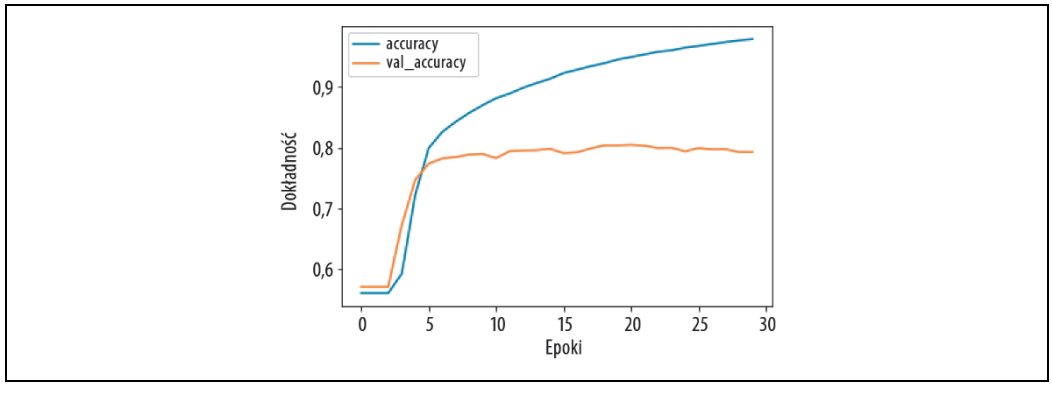
Rysunek 7.8. Działanie dwukierunkowej architektury LSTM



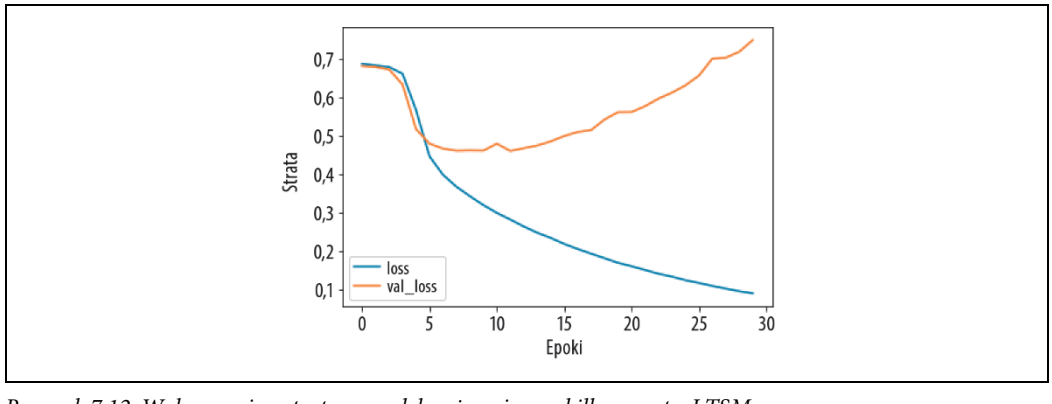
Rysunek 7.9. Wykres zmian dokładności modelu LTSM w ciągu 30 epok



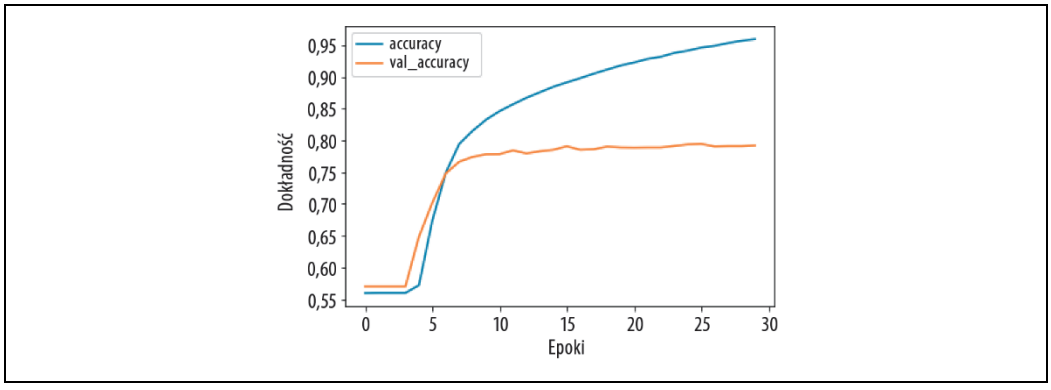
Rysunek 7.10. Wykres zmian straty modelu LTSM w ciągu 30 epok



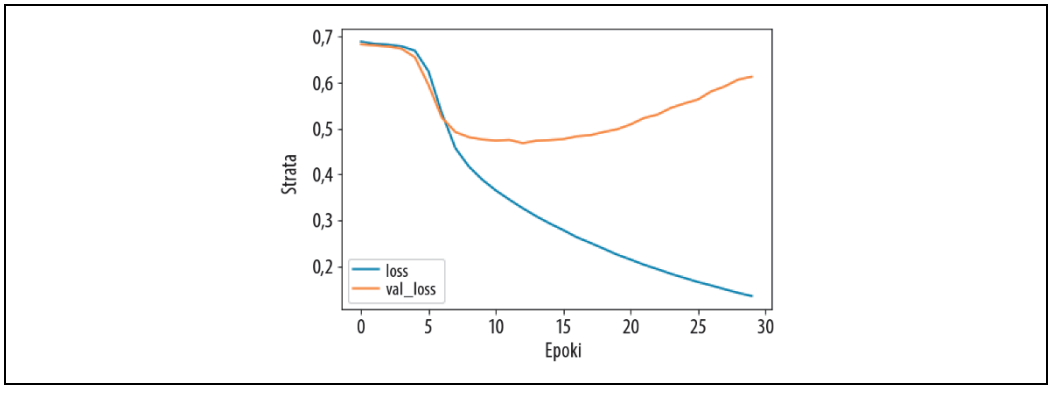
Rysunek 7.11. Wykres zmian dokładności w modelu używającym kilku warstw LTSM



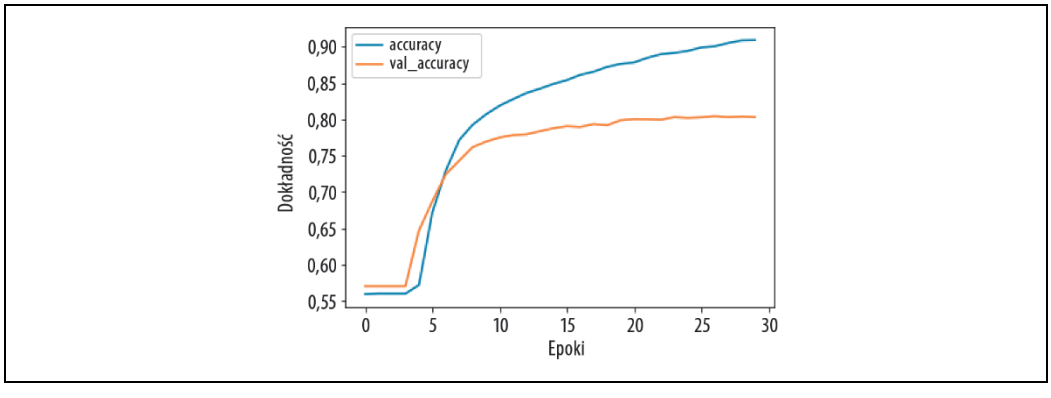
Rysunek 7.12. Wykres zmian straty w modelu używającym kilku warstw LTSM



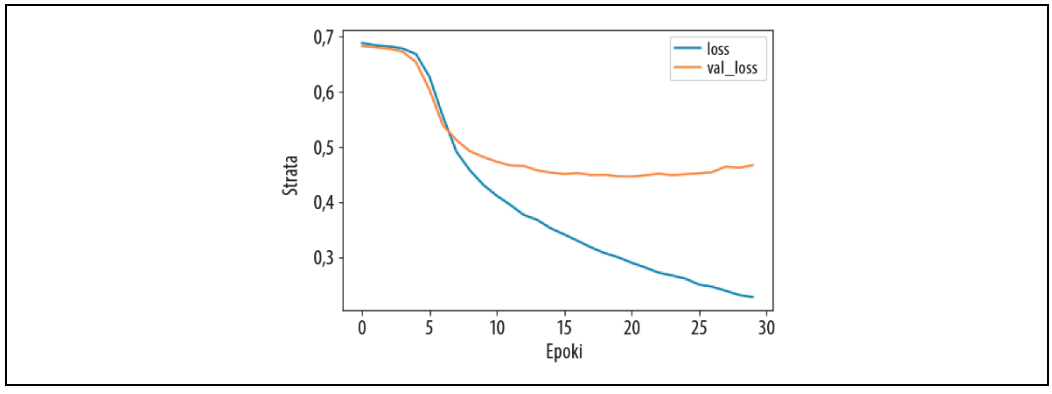
Rysunek 7.13. Wpływ zmniejszonej wartości współczynnika uczenia na dokładność modelu z wieloma warstwami LTSM



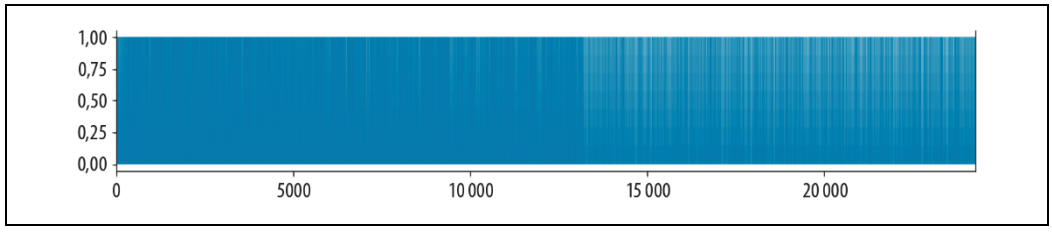
Rysunek 7.14. Wpływ zmniejszonej wartości współczynnika uczenia na stratę modelu z wieloma warstwami LTSM



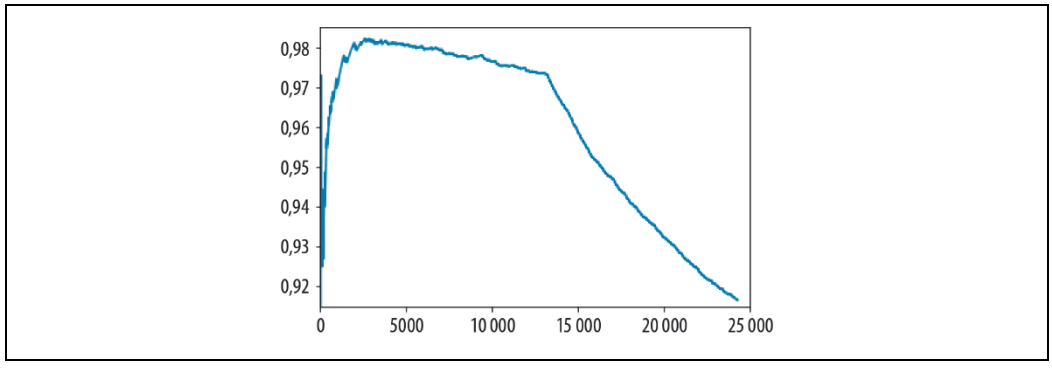
Rysunek 7.15. Wykres zmian dokładności w modelu LTSM wykorzystującym dropout



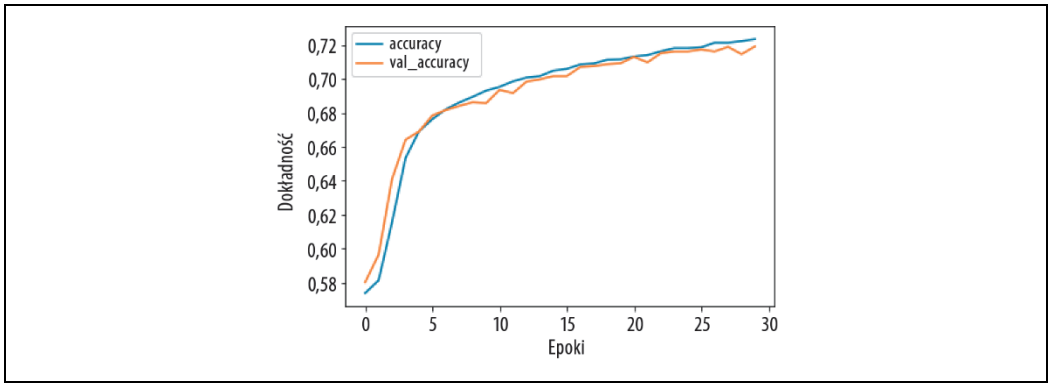
Rysunek 7.16. Wykres zmian straty w modelu LTSM wykorzystującym dropout



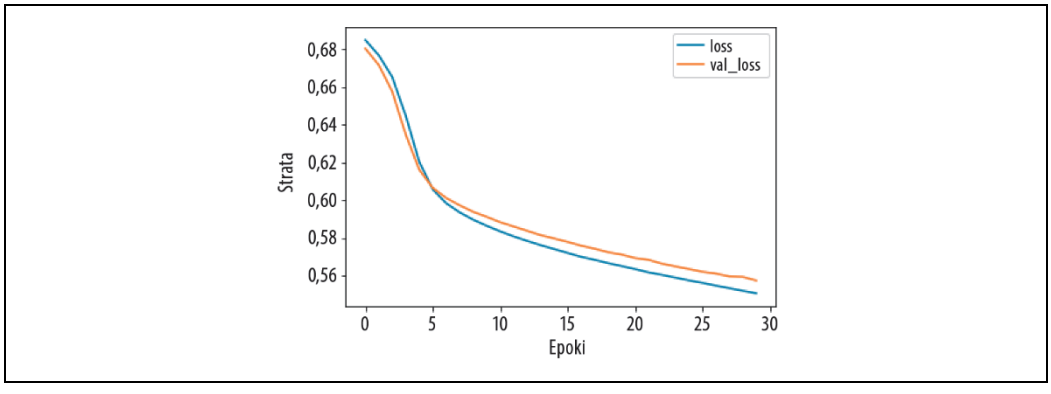
Rysunek 7.17. Wykres częstości występowania słów



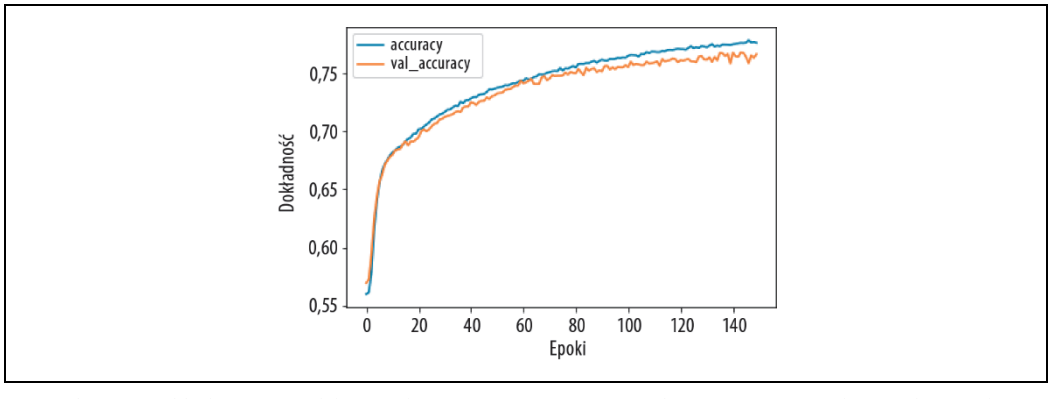
Rysunek 7.18. Częstość występowania słów w bazie GloVe



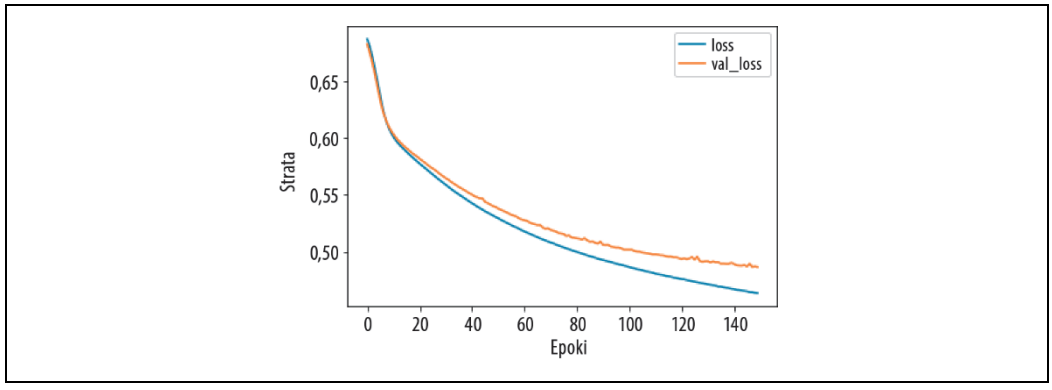
Rysunek 7.19. Dokładność w modelu z wieloma warstwami LTSM wykorzystującym osadzenia zbioru GloVe



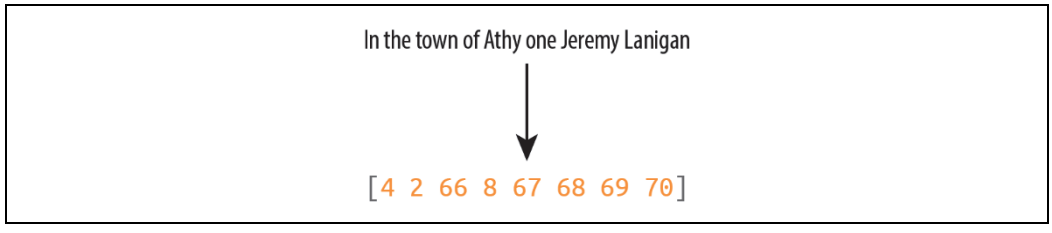
Rysunek 7.20. Strata w modelu z wieloma warstwami LTSM wykorzystującym osadzenia zbioru GloVe



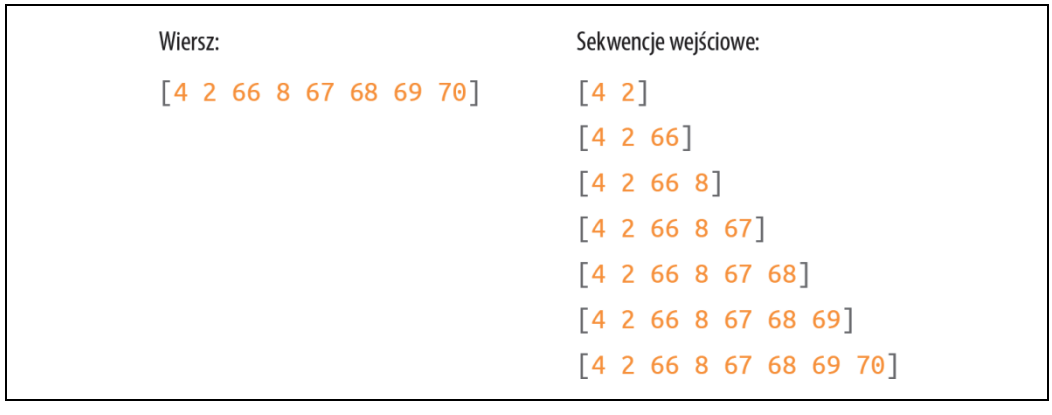
Rysunek 7.21. Dokładność w modelu z wieloma warstwami LTSM wykorzystującym osadzenia zbioru GloVe i trenowanym przez 150 epok



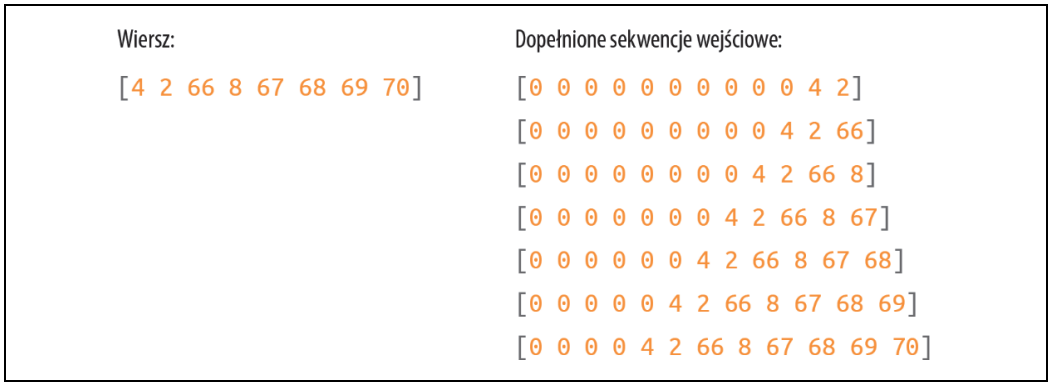
Rysunek 7.22. Strata w modelu z wieloma warstwami LTSM wykorzystującym osadzenia zbioru GloVe i trenowanym przez 150 epok



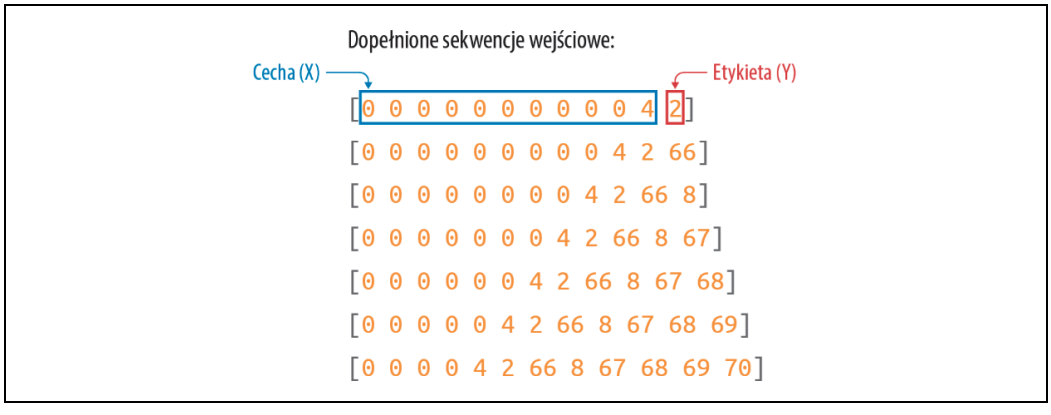
Rysunek 8.1. Tokenizacja zdania



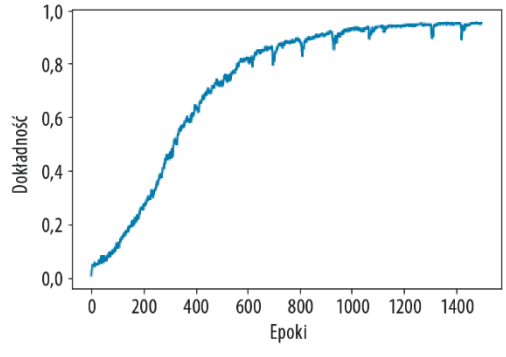
Rysunek 8.2. Zamiana sekwencji na zestaw sekwencji wejściowych



Rysunek 8.3. Dopełnianie sekwencji wejściowych



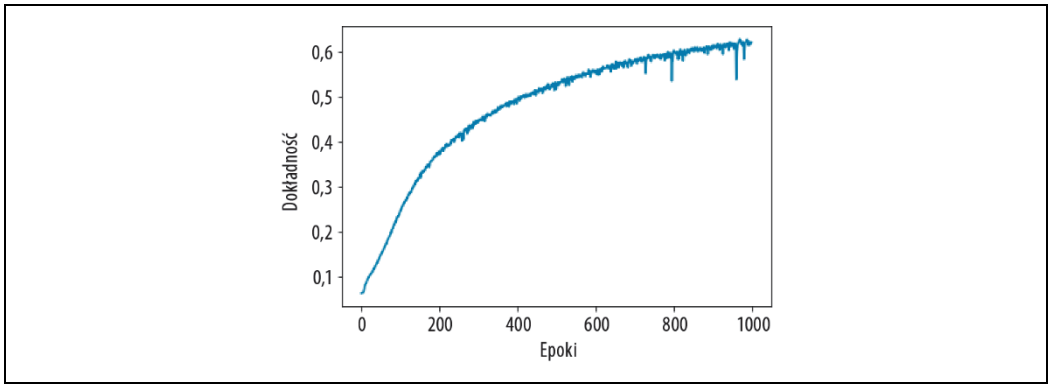
Rysunek 8.4. Przekształcanie dopełnionych sekwencji w cechy (X) i etykiety (Y)



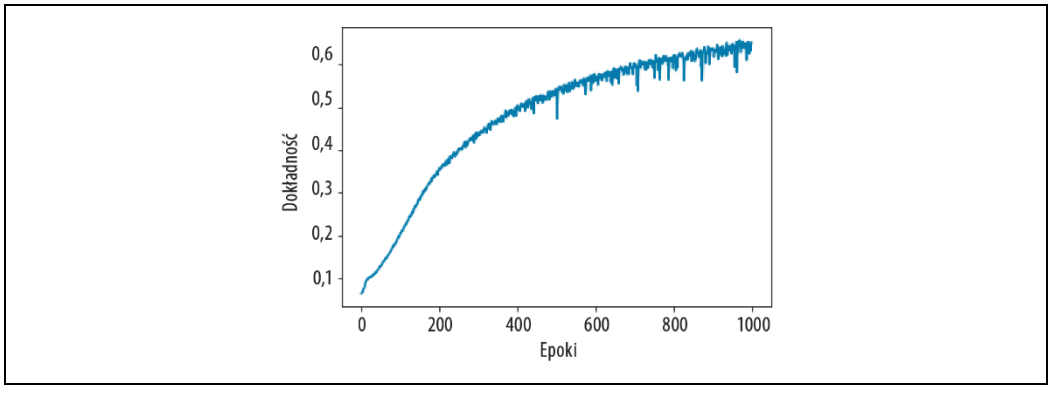
```
   Zdanie: [0 0 0 0 4 2 66 8 67 68 69 70]

        X: [0 0 0 0 0 4 2 66 8 67 68 69]

 Etykieta: [70]

        Y: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
            0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Rysunek 8.5. Etykiety kodowane z gorącą jedynką

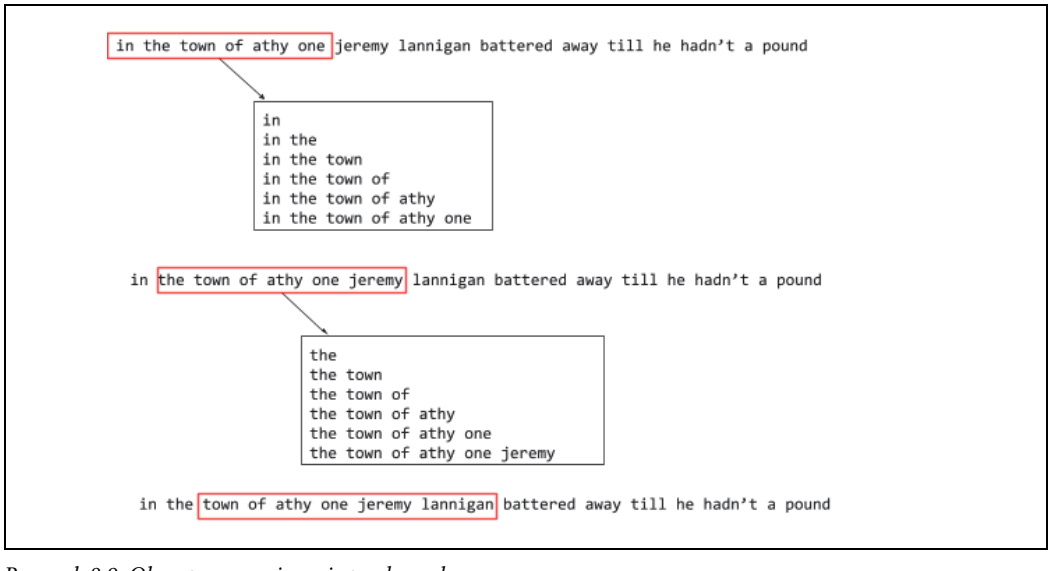
Rysunek 8.6. Wykres zmian dokładności podczas trenowania



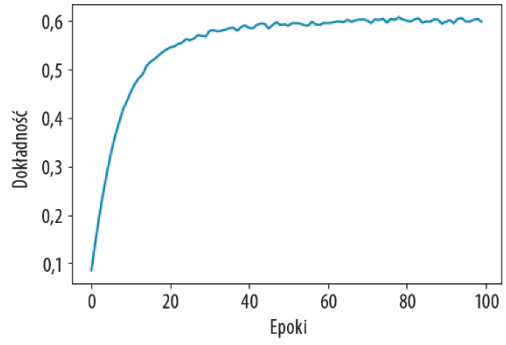
Rysunek 8.7. Trenowanie z użyciem większego zbioru danych



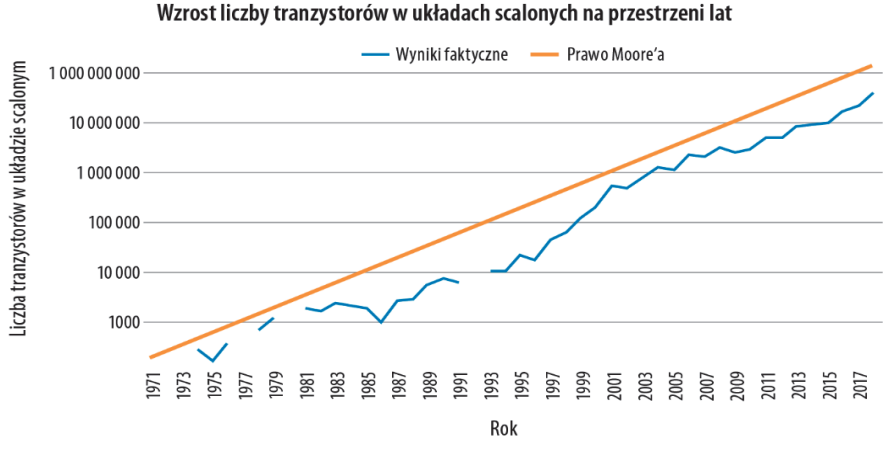
Rysunek 8.8. Dodanie drugiej warstwy LSTM



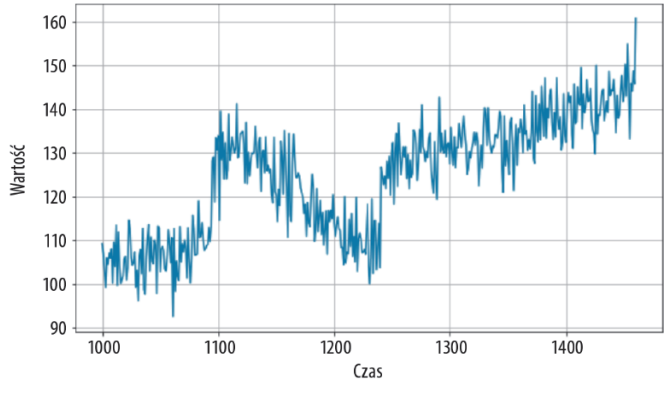
Rysunek 8.9. Okno przesuwające się po słowach



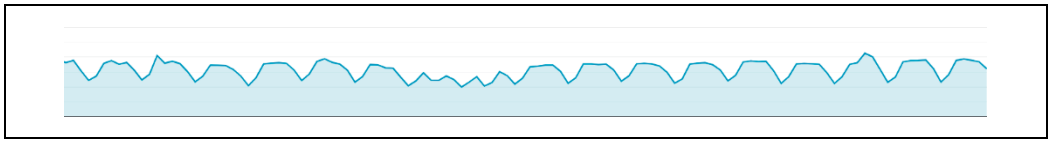
Rysunek 8.10. Krzywa uczenia po zmodyfikowaniu hiperparametrów



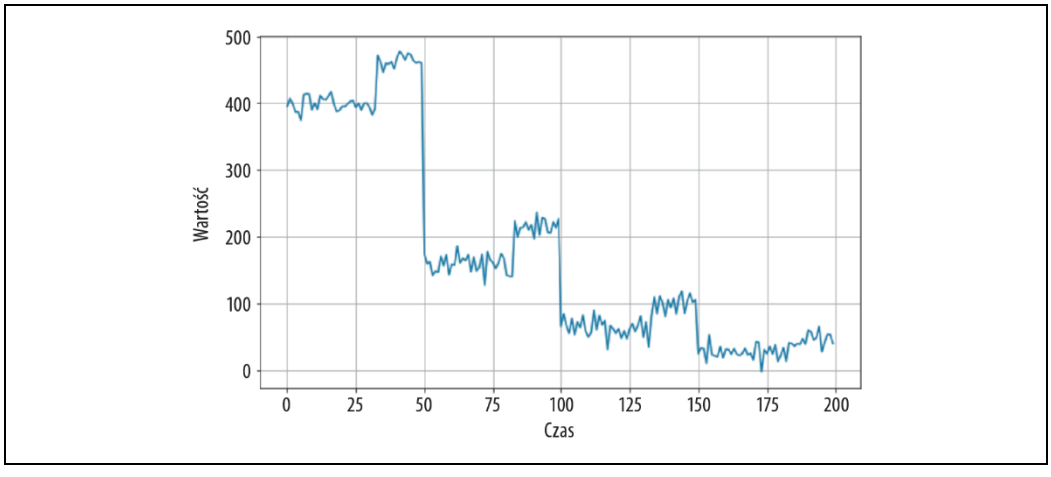
Rysunek 9.1. Wykres prawa Moore'a



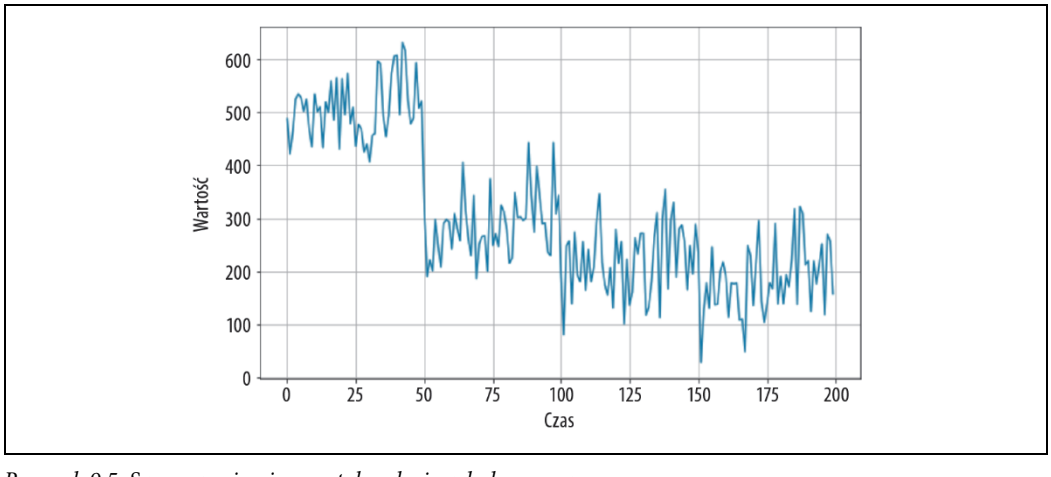
Rysunek 9.2. Szereg czasowy ze świata rzeczywistego



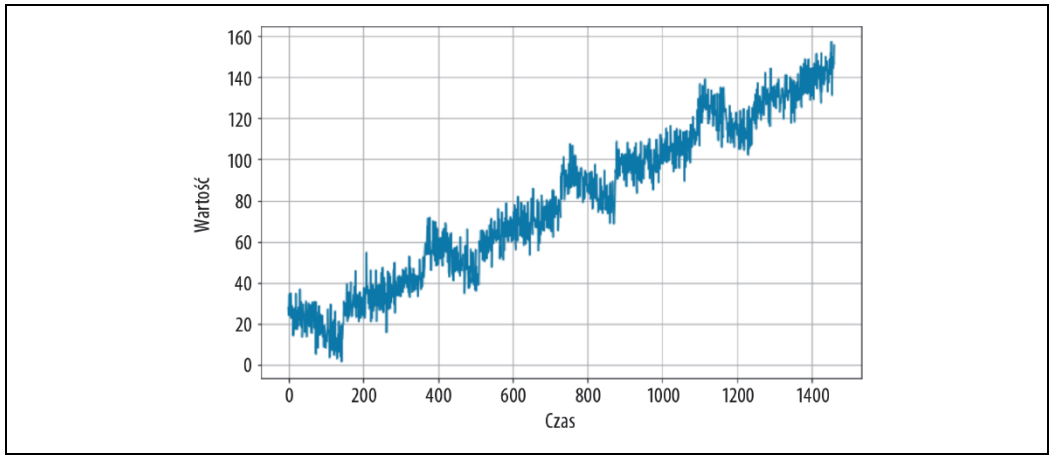
Rysunek 9.3. Natężenie ruchu na stronie internetowej



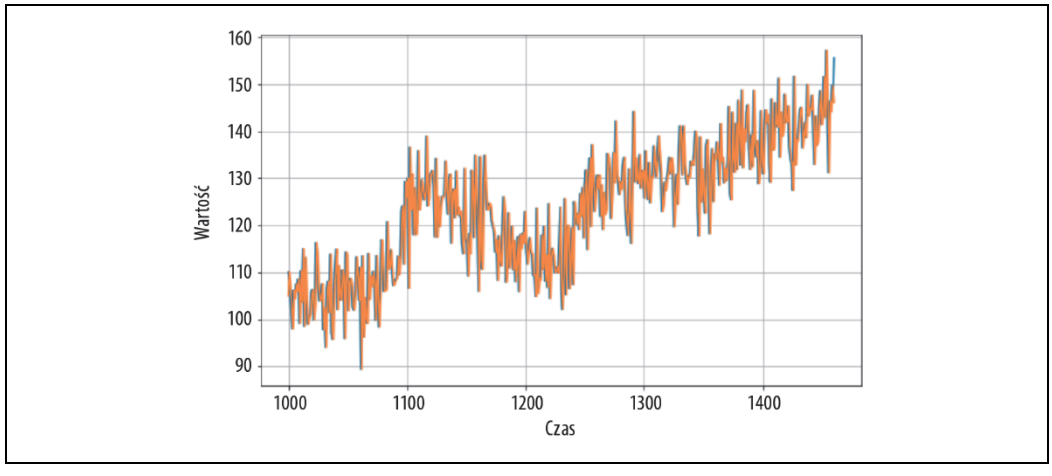
Rysunek 9.4. Autokorelacja



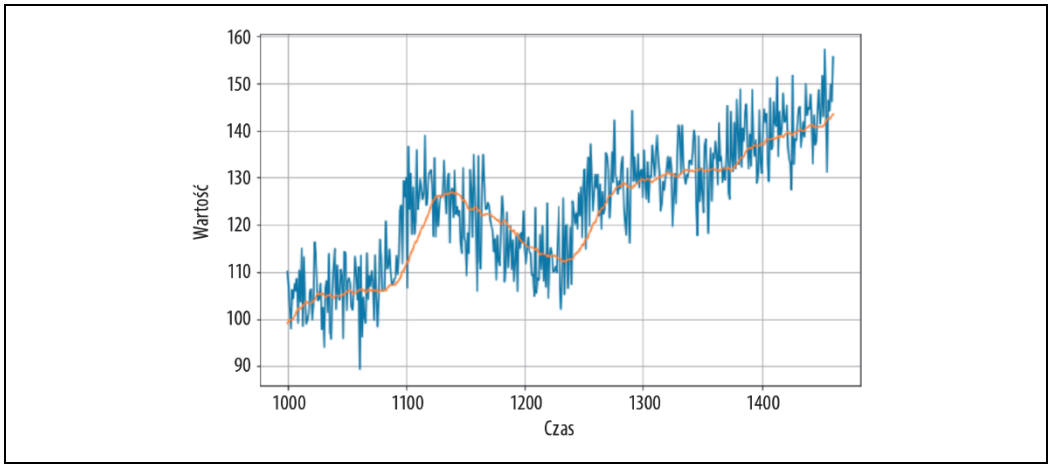
Rysunek 9.5. Szereg zawierający autokorelację z dodanym szumem



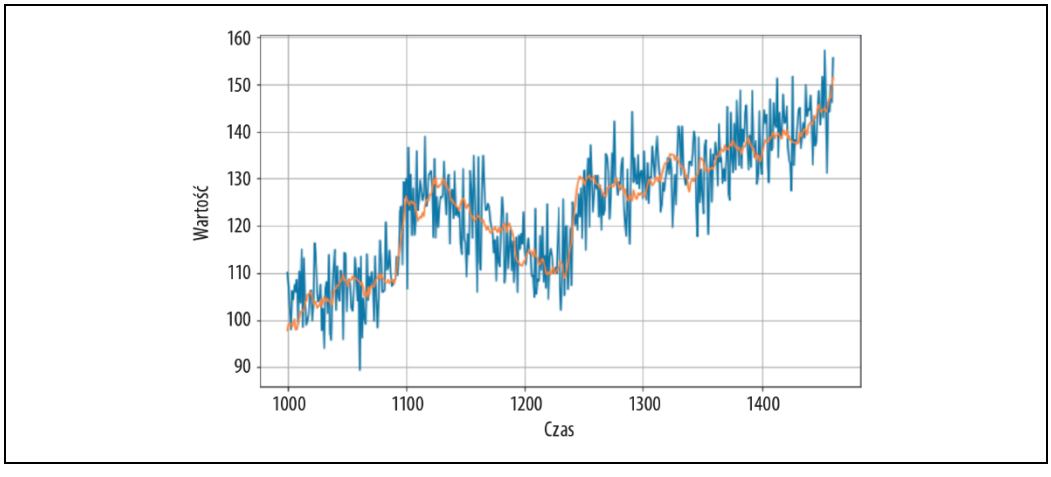
Rysunek 9.6. Szereg czasowy zawierający tendencję, sezonowość i szum



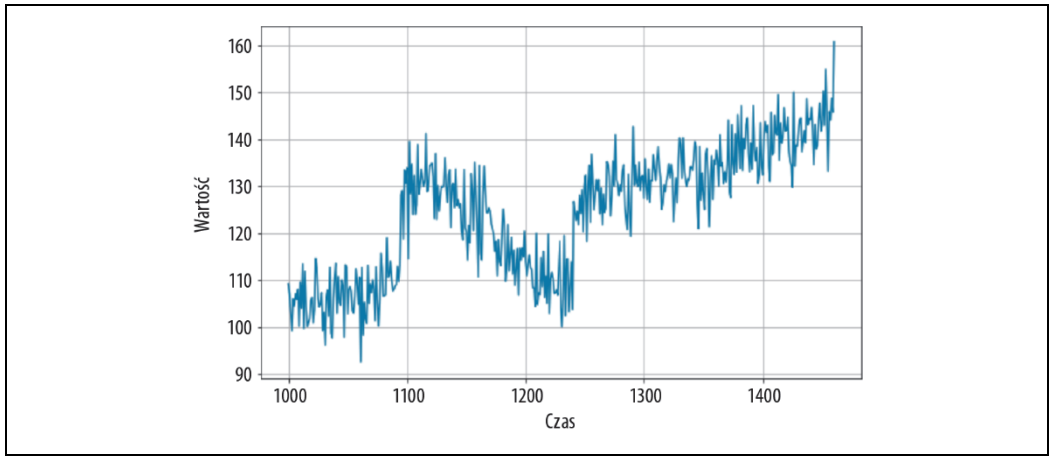
Rysunek 9.7. Prosta prognoza szeregu czasowego



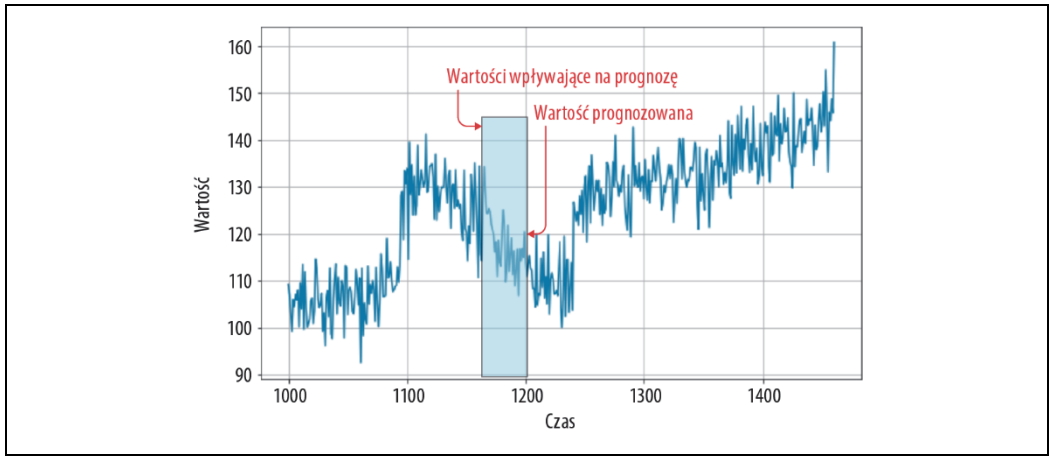
Rysunek 9.8. Wykres średniej ruchomej



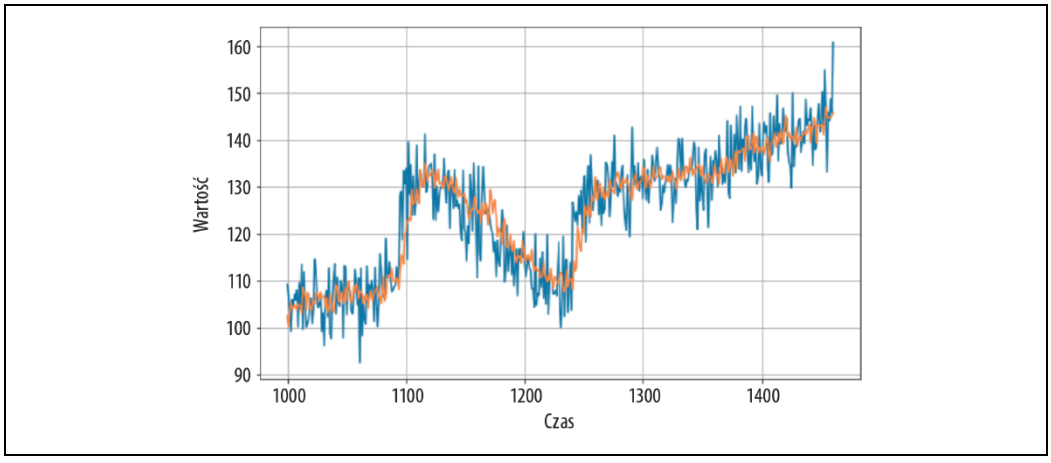
Rysunek 9.9. Ulepszenie metody wykorzystującej średnią ruchomą



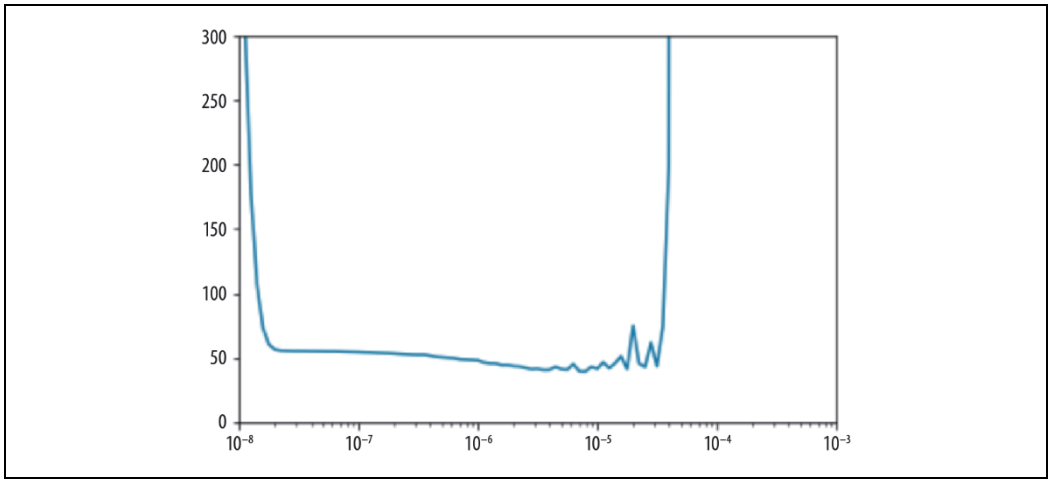
Rysunek 10.1. Syntetyczny szereg czasowy



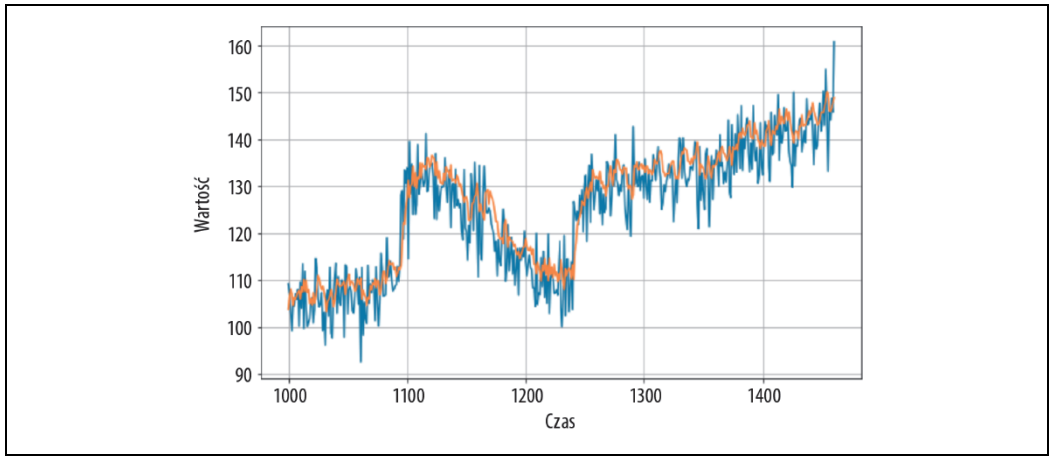
Rysunek 10.2. Wartości wcześniejsze wpływają na prognozę



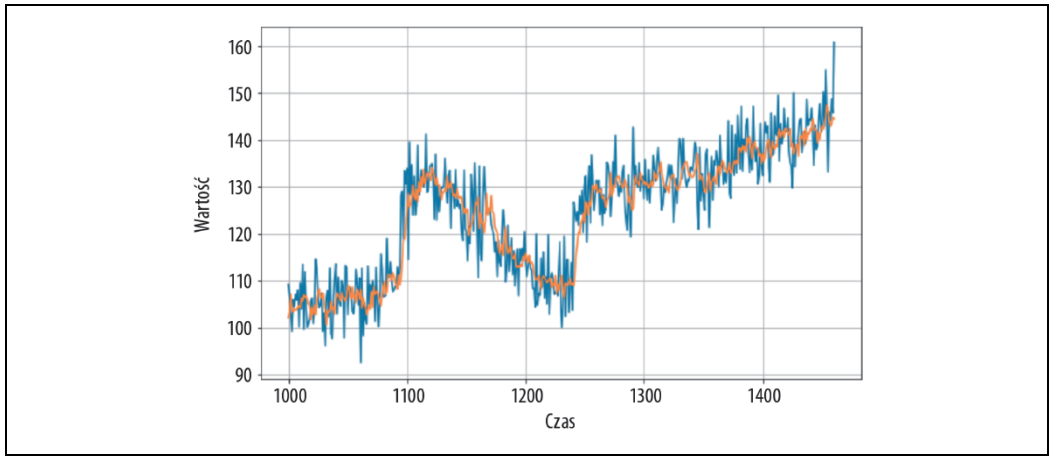
Rysunek 10.3. Prognozy i wartości rzeczywiste



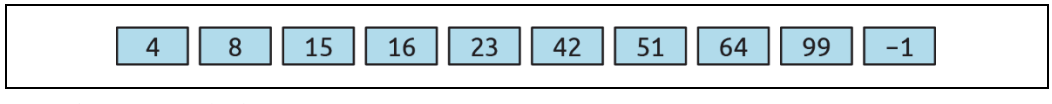
Rysunek 10.4. Wykres zależności współczynnika uczenia od straty



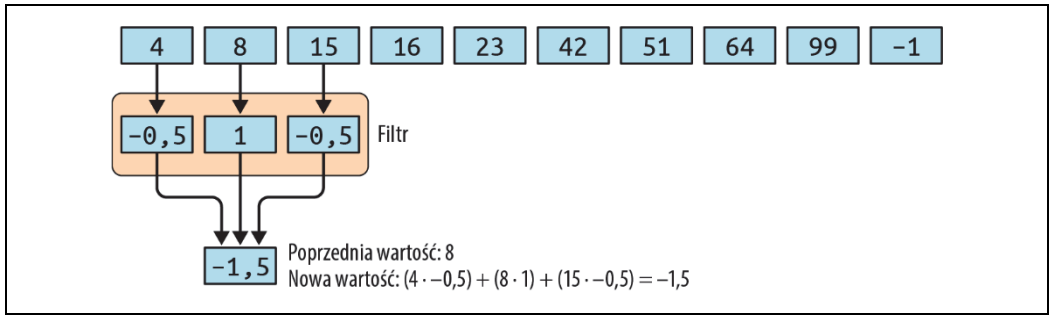
Rysunek 10.5. Wykres w przypadku zmodyfikowanej wartości współczynnika uczenia



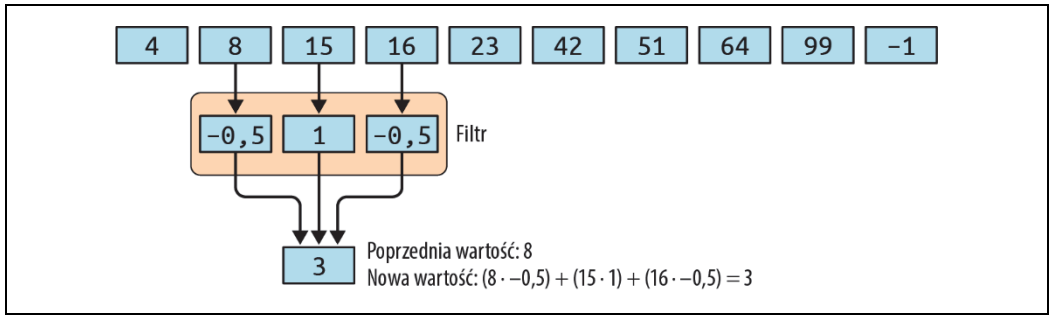
Rysunek 10.6. Wykres prognoz dla modelu ze zoptymalizowanymi hiperparametrami



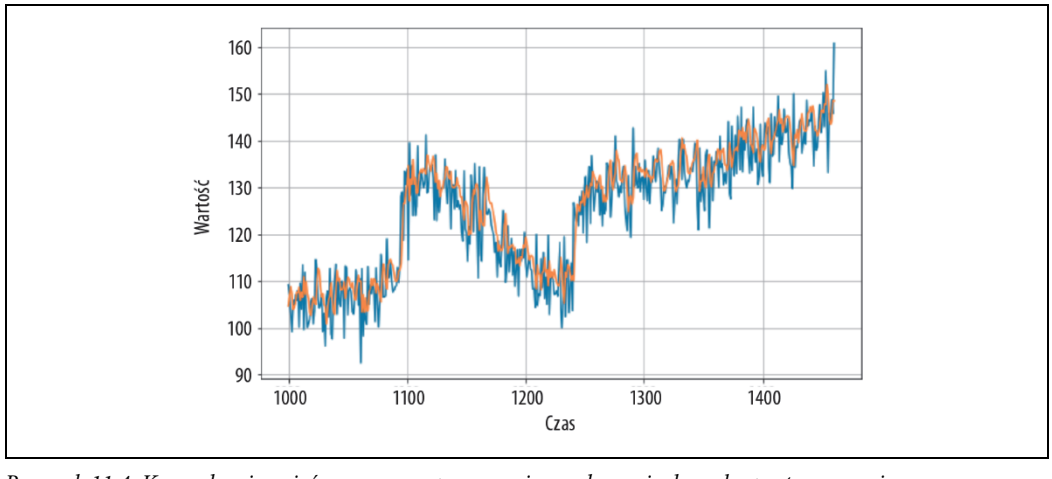
Rysunek 11.1. Szereg liczbowy



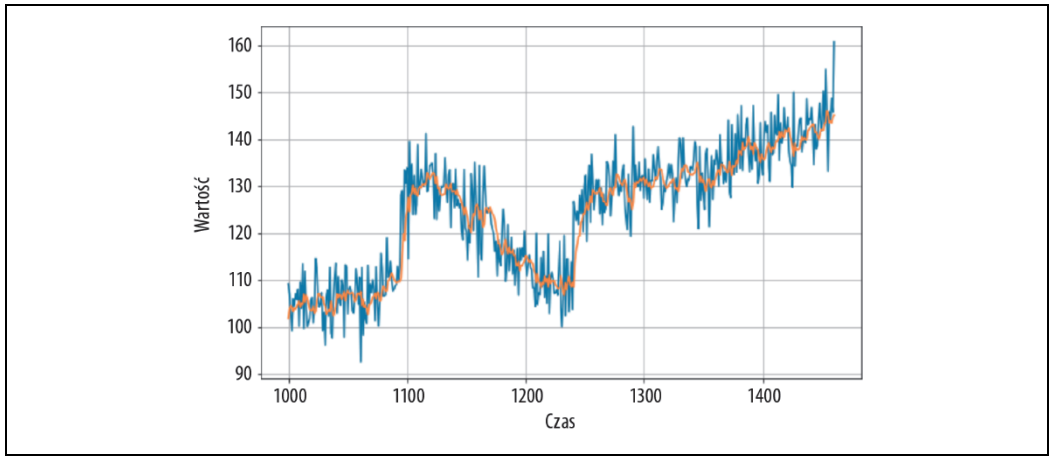
Rysunek 11.2. Użycie konwolucji z szeregiem liczbowym



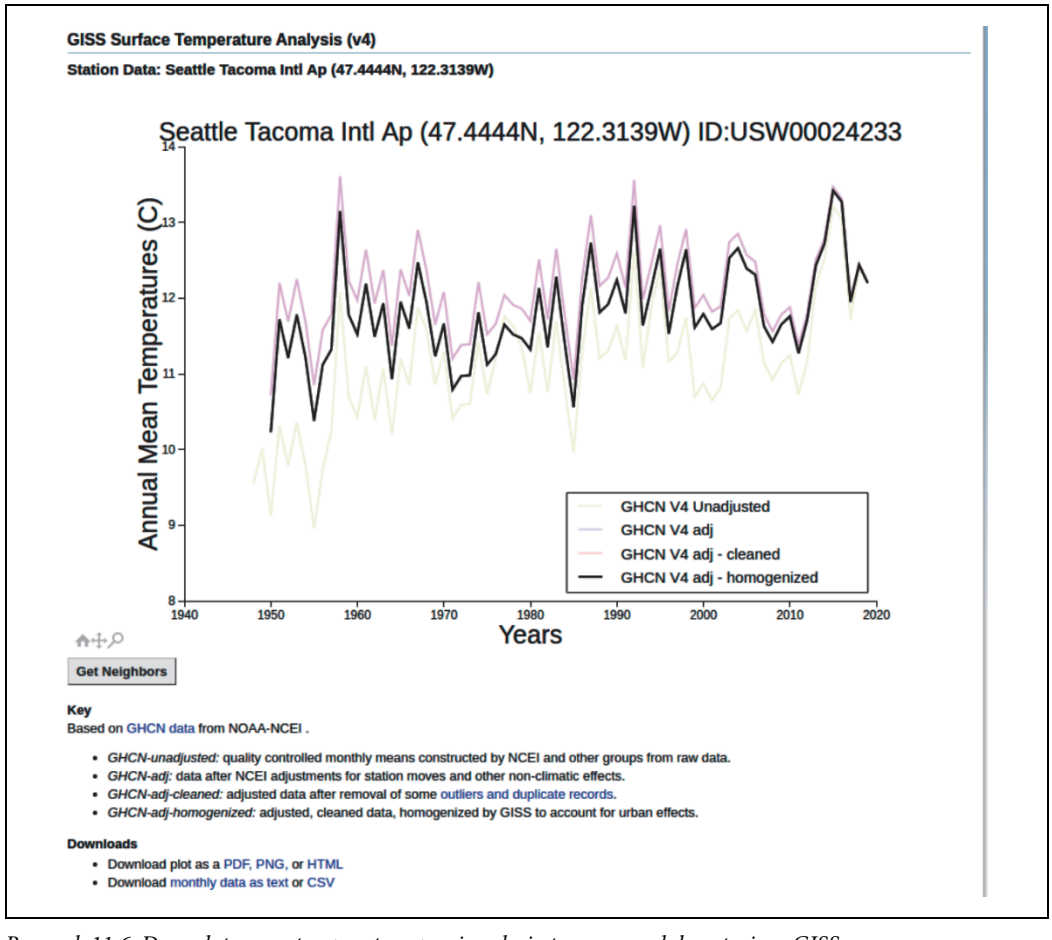
Rysunek 11.3. Kolejny krok konwolucji jednowymiarowej



Rysunek 11.4. Konwolucyjna sieć neuronowa prognozująca sekwencję danych opartą na czasie



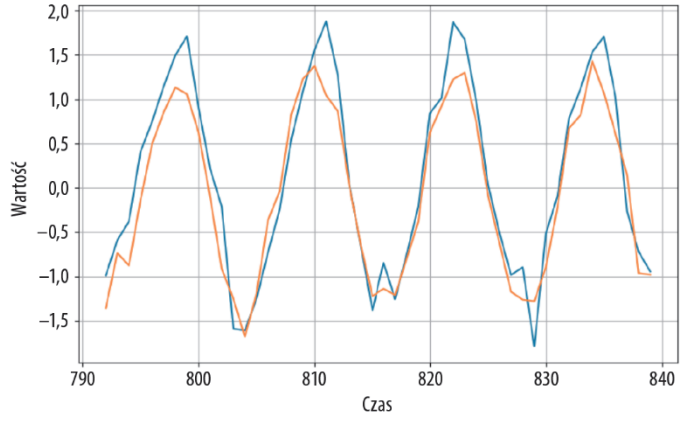
Rysunek 11.5. Zoptymalizowane prognozy sieci CNN



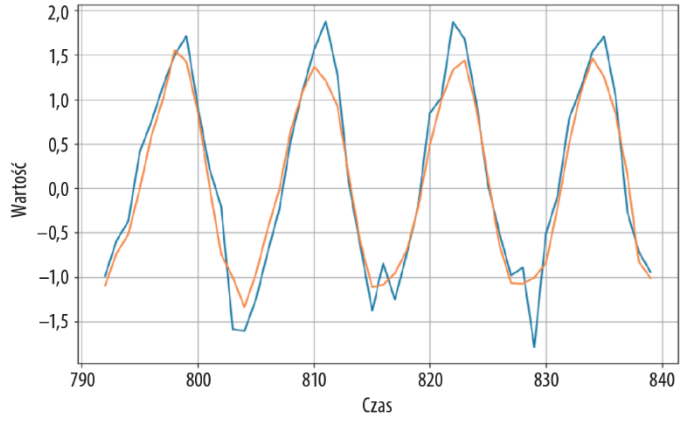
Rysunek 11.6. Dane dotyczące temperatury powierzchni otrzymane z laboratorium GISS

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
2	1950	-2.54	5.85	6.99	9.37	12.31	17.04	18.99	19.05	15.74	10.95	7.84	8.5
3	1951	4.14	6.19	5.49	11.15	13.73	17.57	19.38	17.86	16.43	11.85	8.38	3.87
4	1952	3.86	6.21	7.13	10.55	13.63	15.01	18.86	18.63	16.51	13.77	6.43	6.76
5	1953	8.25	5.95	7.49	9.63	12.85	14.45	18	18.49	16.77	13.17	9.56	7.12
6	1954	3.67	7.08	6.24	8.94	13.57	14.91	17.02	17.3	16.24	11.73	10.83	6.26
7	1955	5.38	4.86	5.37	8.39	11.78	15.83	17	17.49	15.3	11.62	5.23	4.96
8	1956	5.3	3.45	6.32	11.2	15.2	15.25	19.54	18.61	15.81	10.77	6.98	5.64

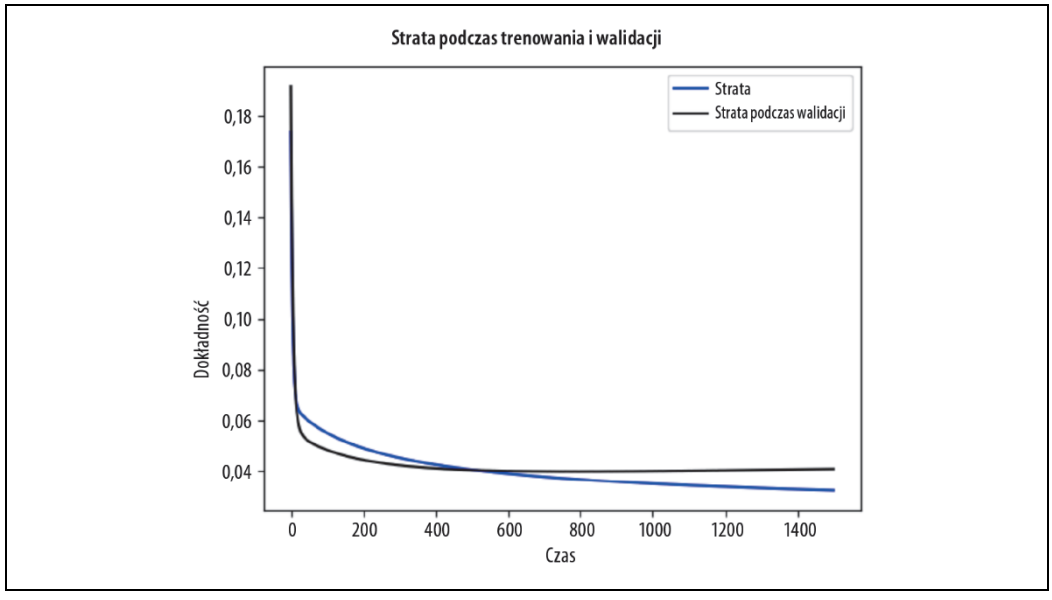
Rysunek 11.7. Podgląd danych



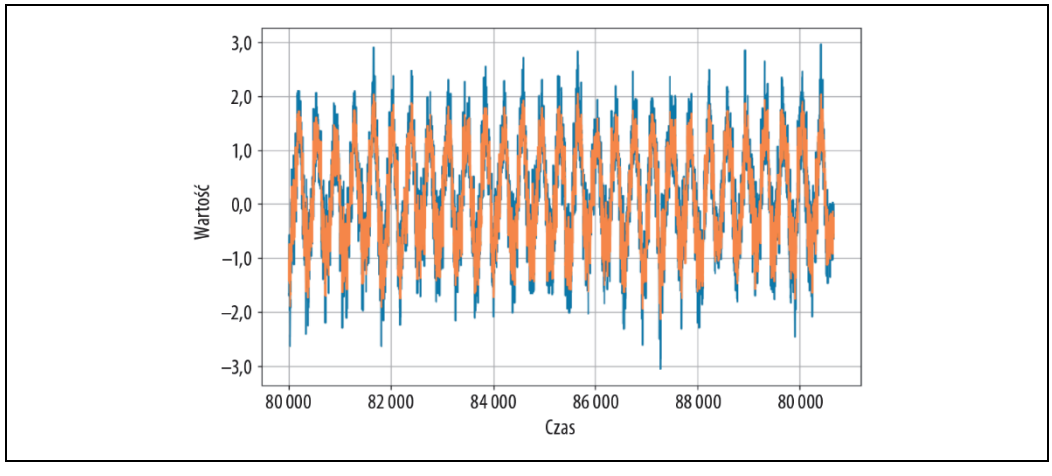
Rysunek 11.8. Wynik działania modelu z warstwą SimpleRNN



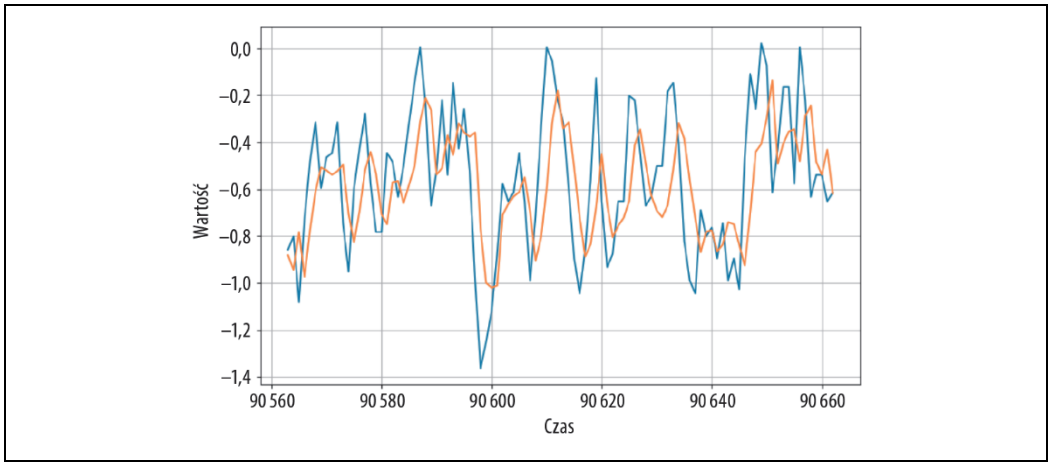
Rysunek 11.9. Sieć RNN trenowana przez 1500 epok



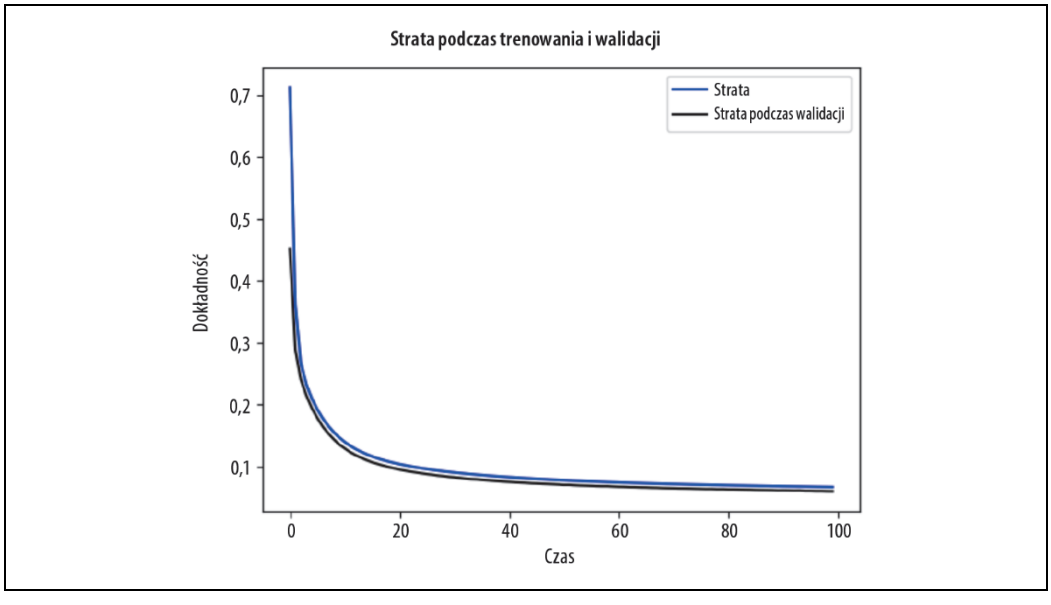
Rysunek 11.10. Strata podczas trenowania i walidacji dla modelu z warstwą SimpleRNN



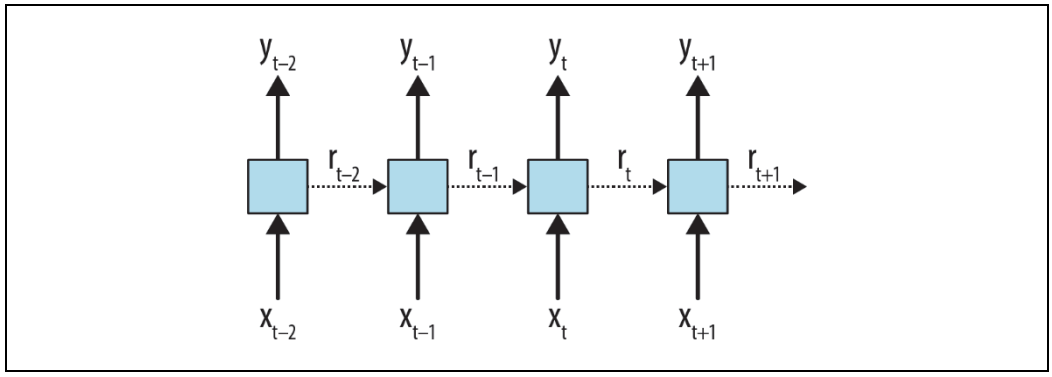
Rysunek 11.11. Prognozy w porównaniu z danymi realnymi



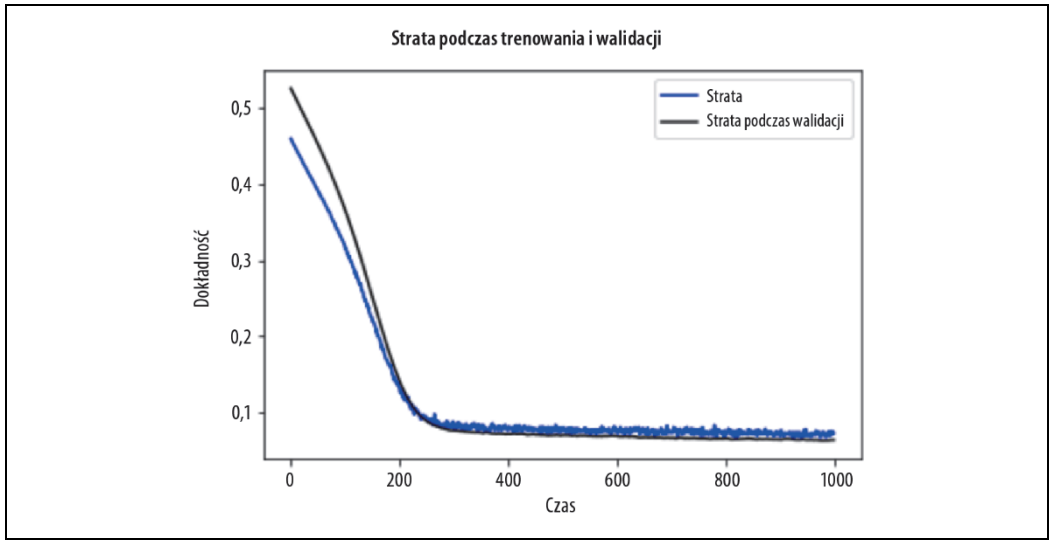
Rysunek 11.12. Wyniki dla danych pochodzących ze 100 dni



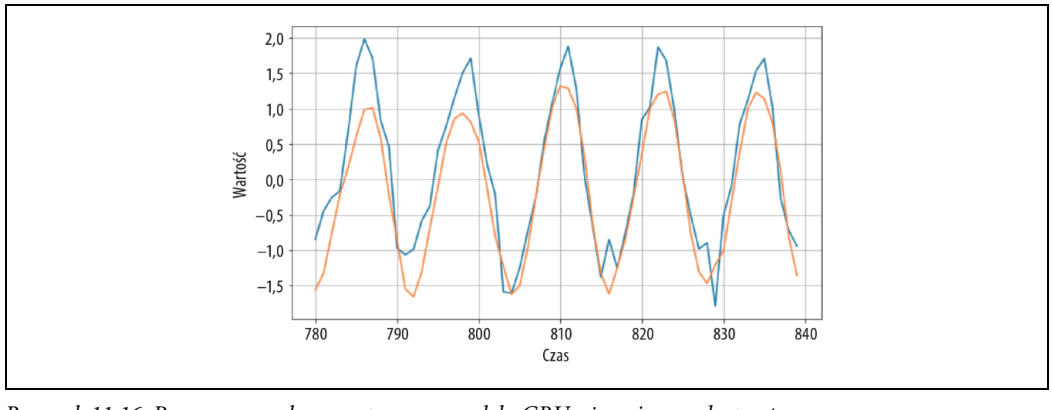
Rysunek 11.13. Strata podczas trenowania i walidacji w przypadku większego zbioru danych



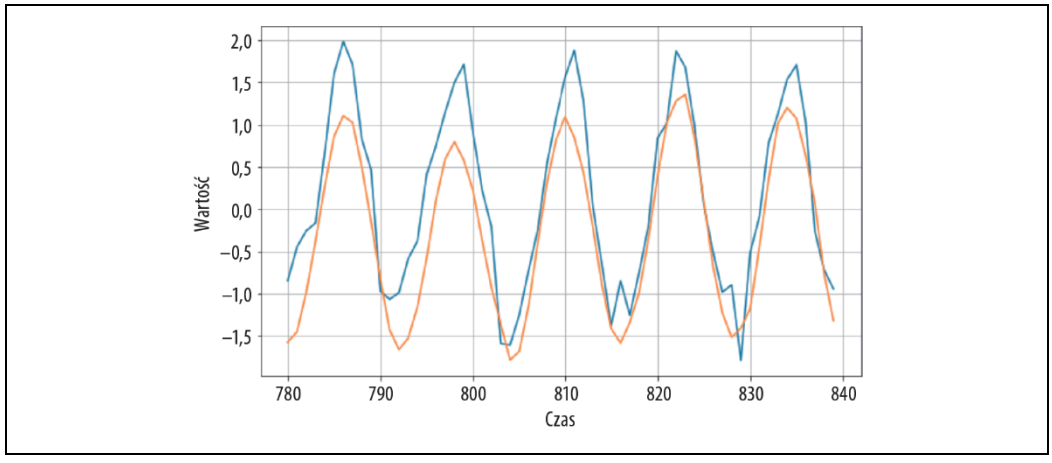
Rysunek 11.14. Rekurencyjna sieć neuronowa



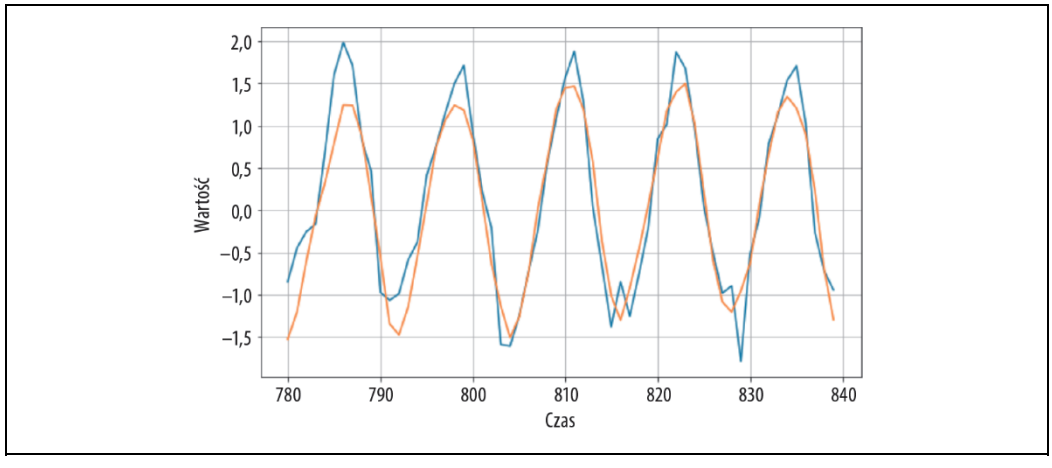
Rysunek 11.15. Trenowanie modelu GRU używającego dropoutu



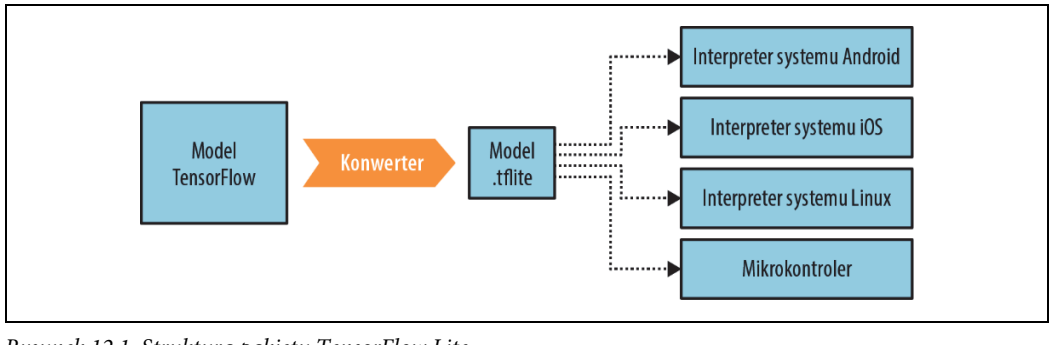
Rysunek 11.16. Prognozy uzyskane za pomocą modelu GRU używającego dropoutu



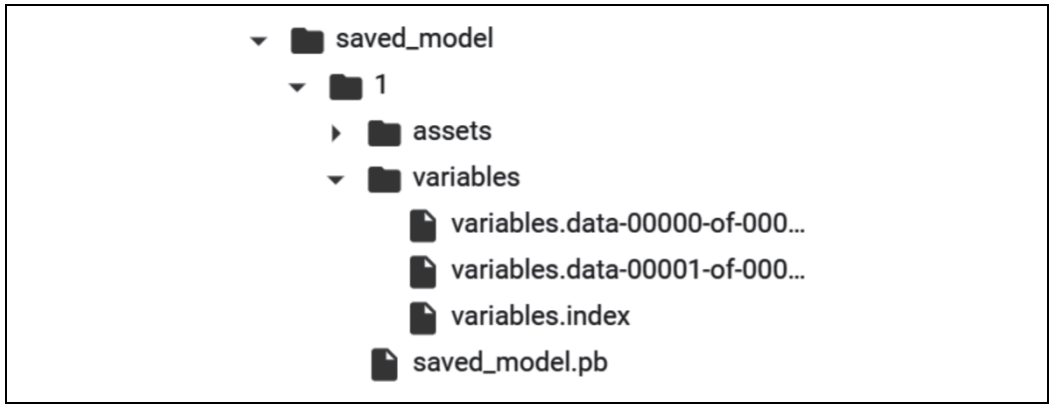
Rysunek 11.17. Trenowanie za pomocą dwukierunkowego modelu GRU



Rysunek 11.18. Wyniki uzyskane przy użyciu dwukierunkowego modelu GRU wykorzystującego większy rozmiar okna

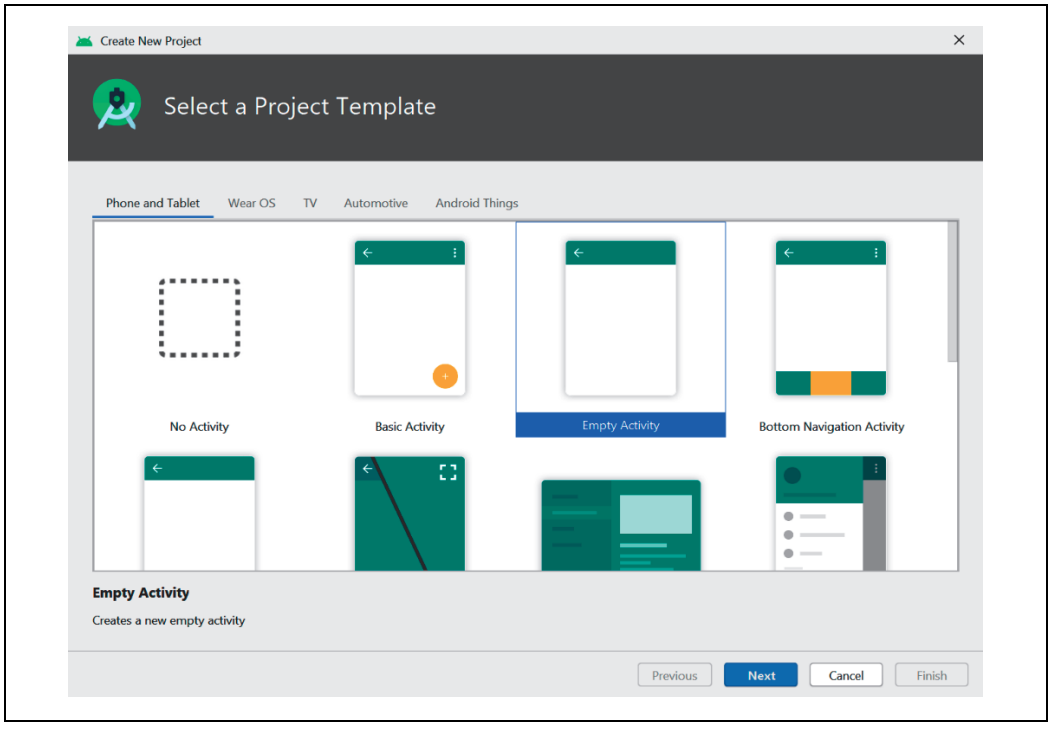


Rysunek 12.1. Struktura pakietu TensorFlow Lite

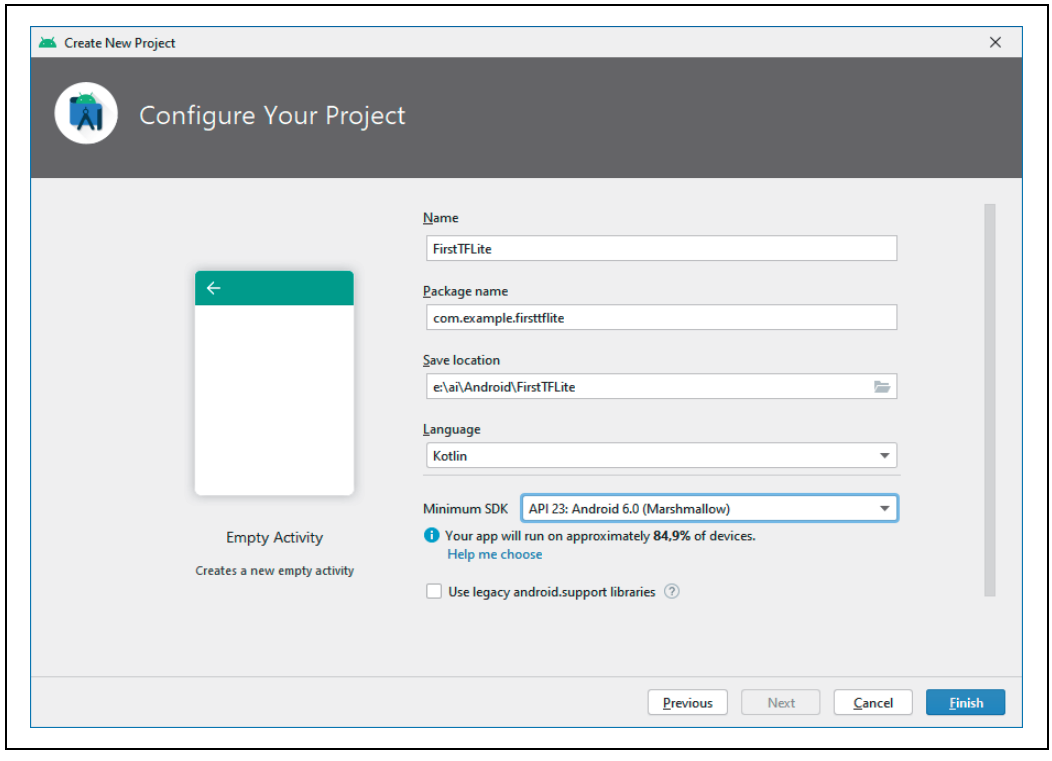


Rysunek 12.2. Struktura formatu SavedModel

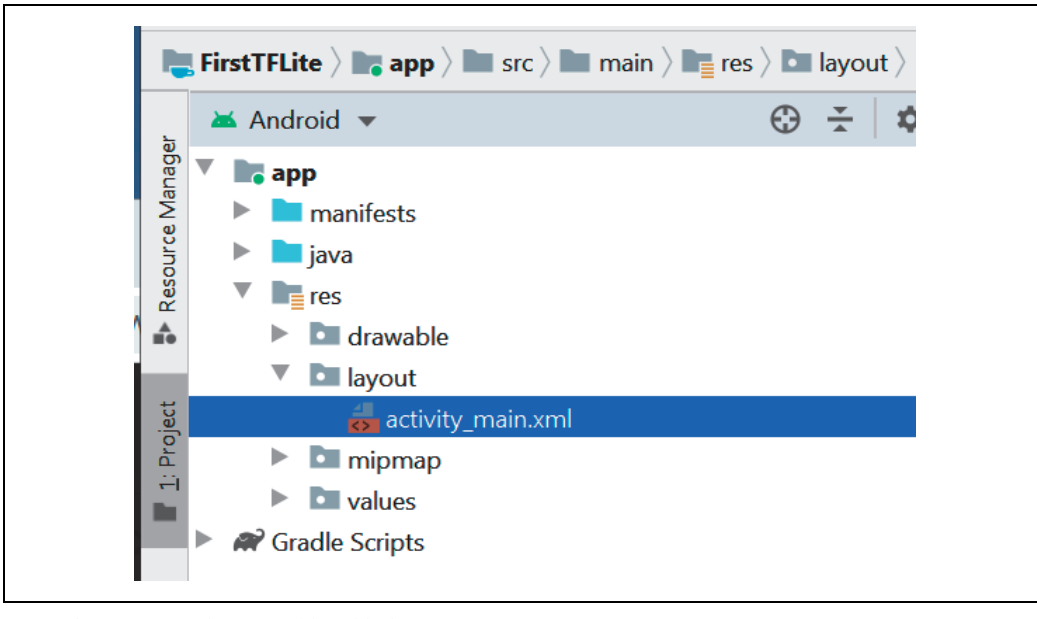
Rysunek 12.3. Wyniki wnioskowania



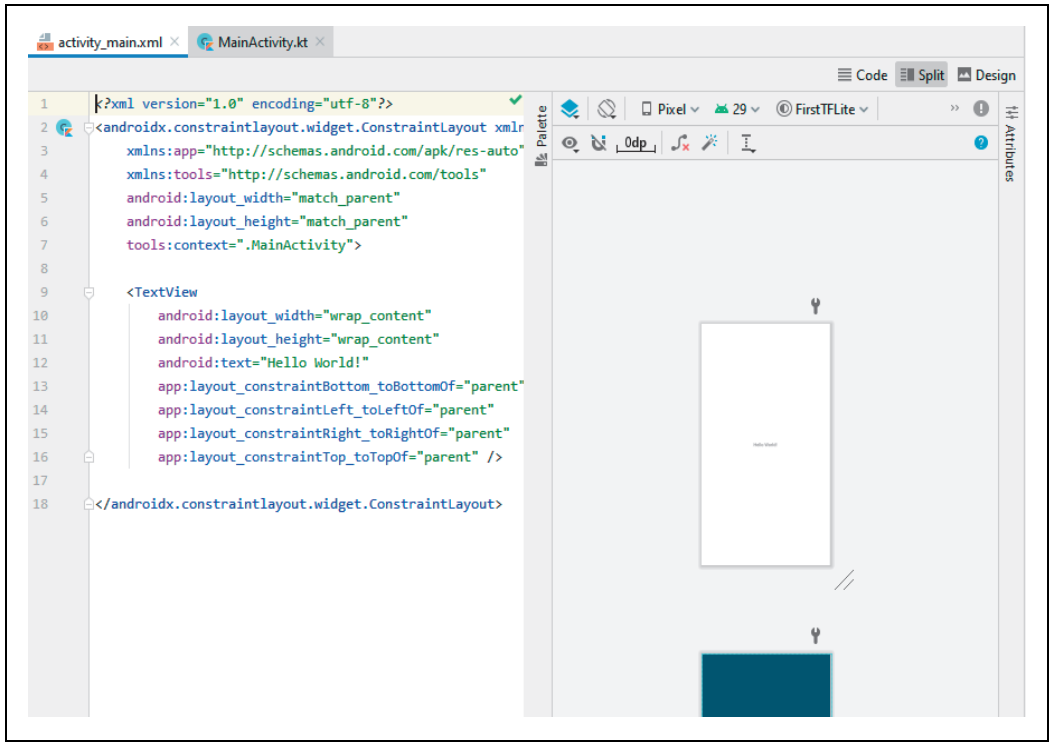
Rysunek 13.1. Tworzenie nowego projektu w środowisku Android Studio



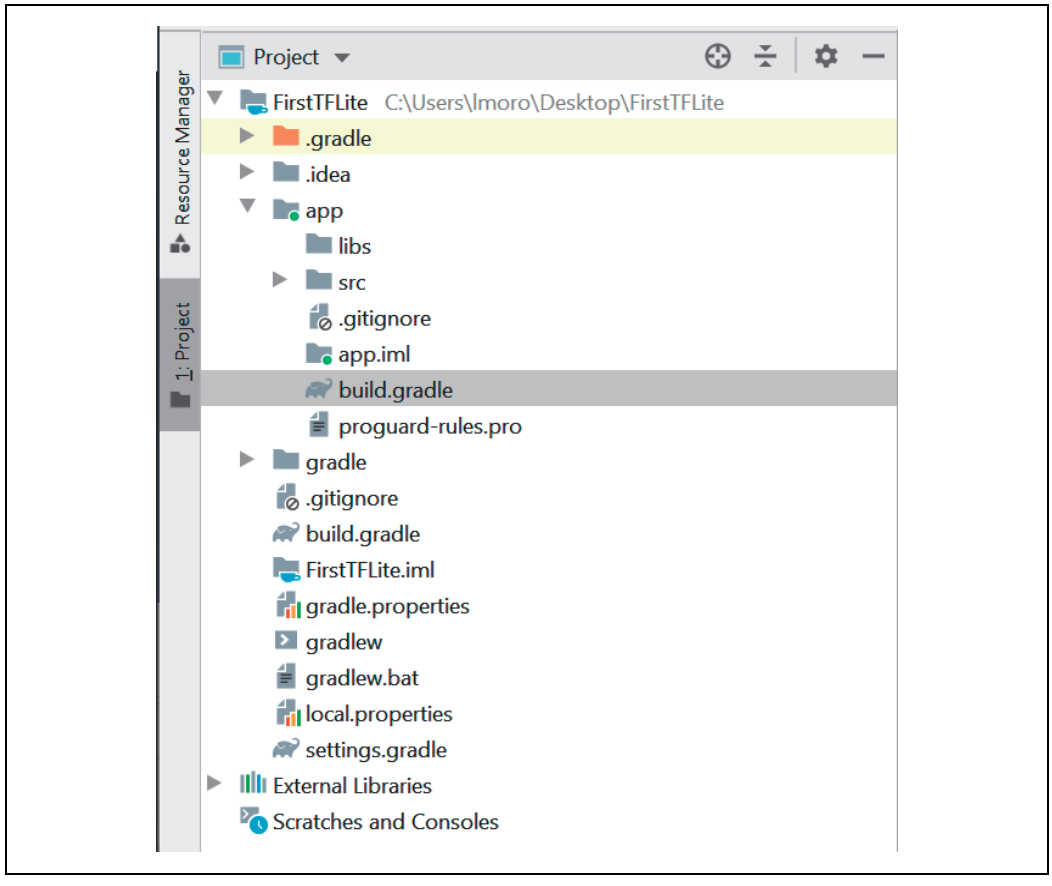
Rysunek 13.2. Konfigurowanie projektu



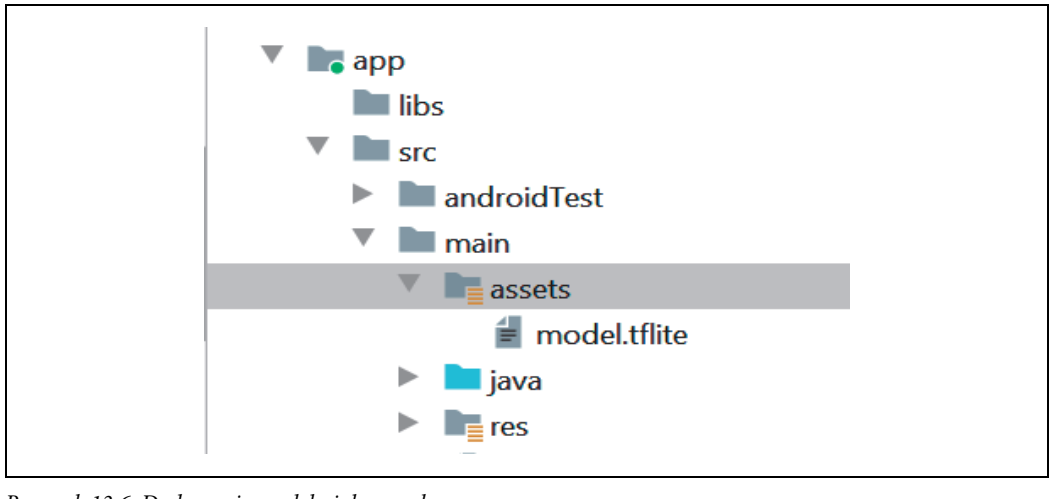
Rysunek 13.3. Wyszukiwanie pliku układu



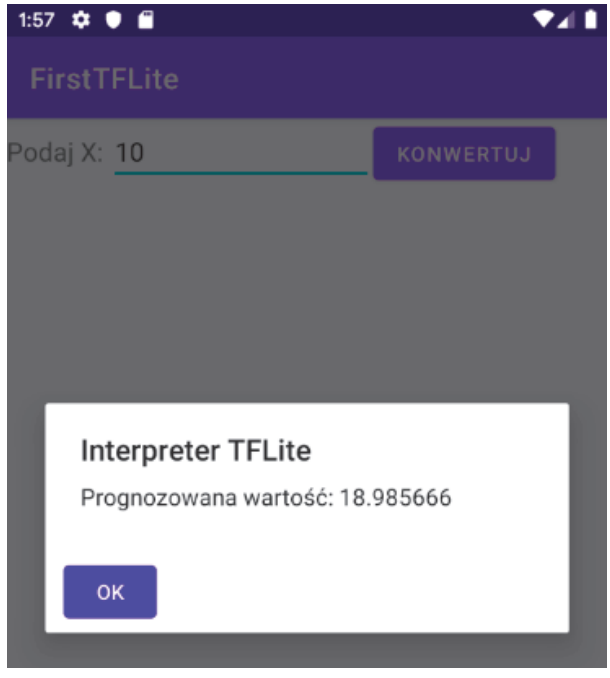
Rysunek 13.4. Użycie edytora układu w Android Studio



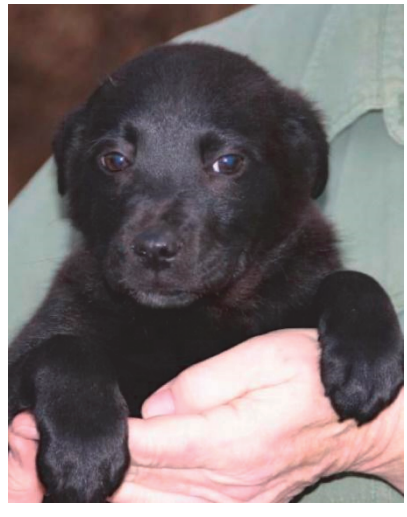
Rysunek 13.5. Wybór pliku build.gradle



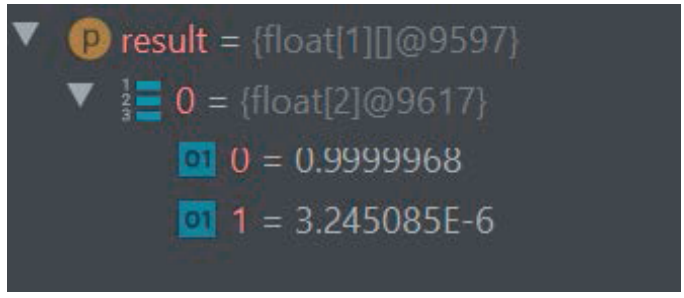
Rysunek 13.6. Dodawanie modelu jako zasobu



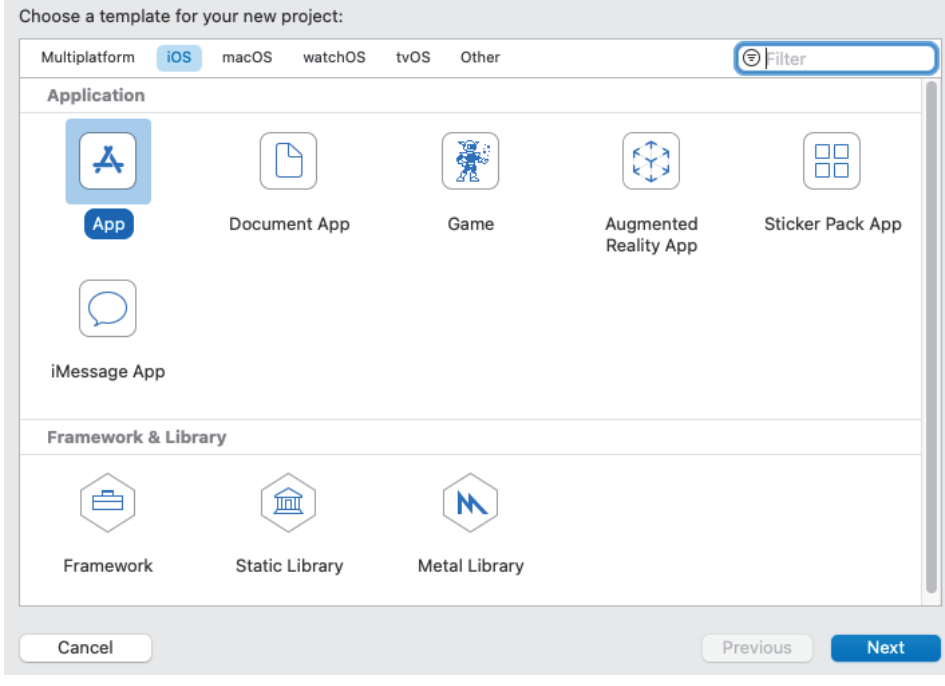
Rysunek 13.7. Uruchomienie interpretera w emulatorze



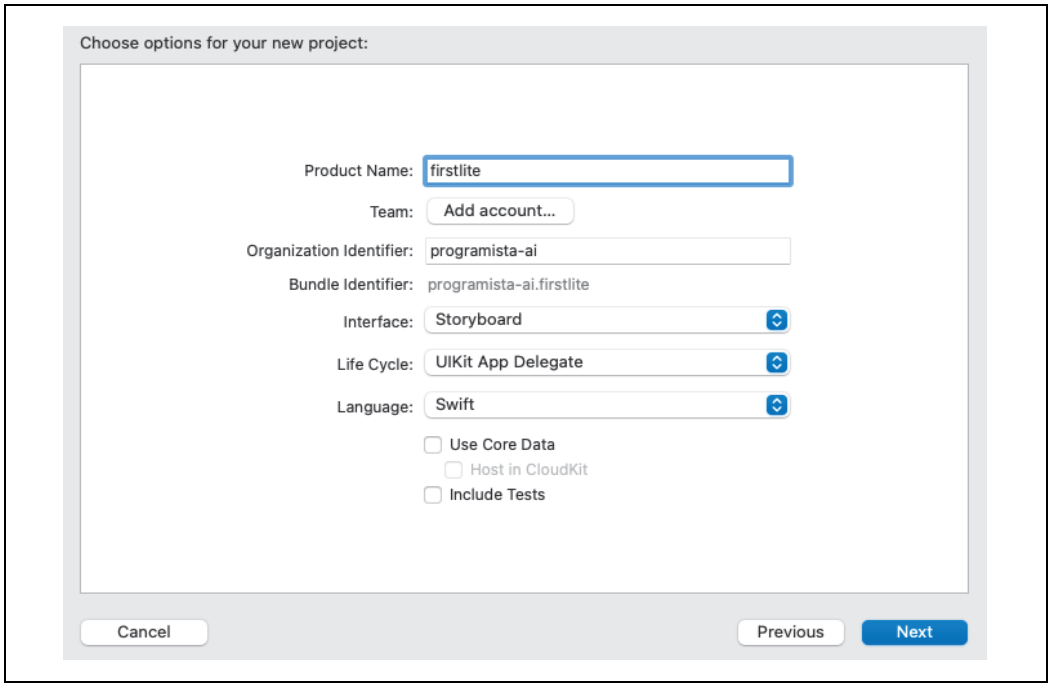
Rysunek 13.8. Obraz, na którym model powinien rozpoznać psa



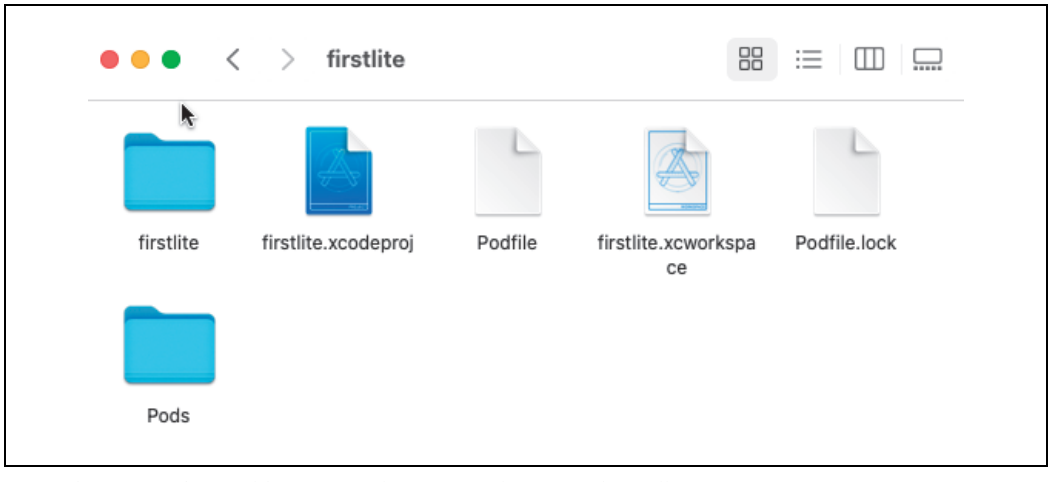
Rysunek 13.9. Wyświetlanie wartości wyjściowych



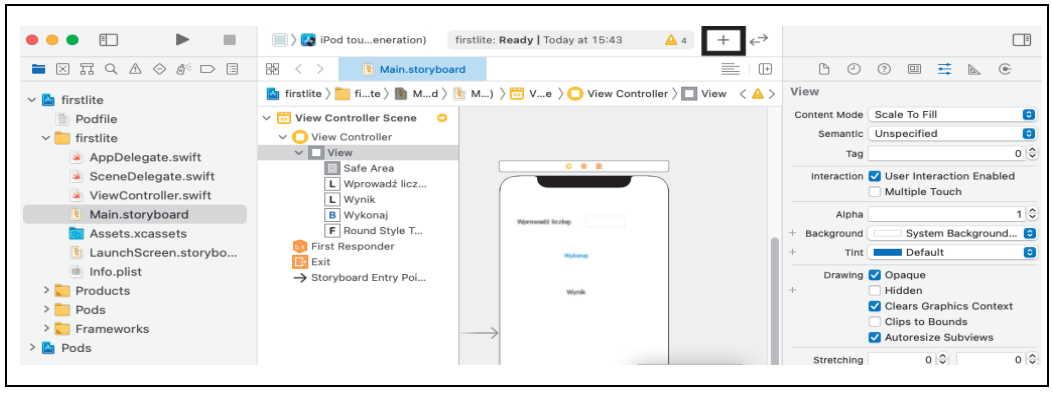
Rysunek 14.1. Tworzenie aplikacji iOS w środowisku Xcode



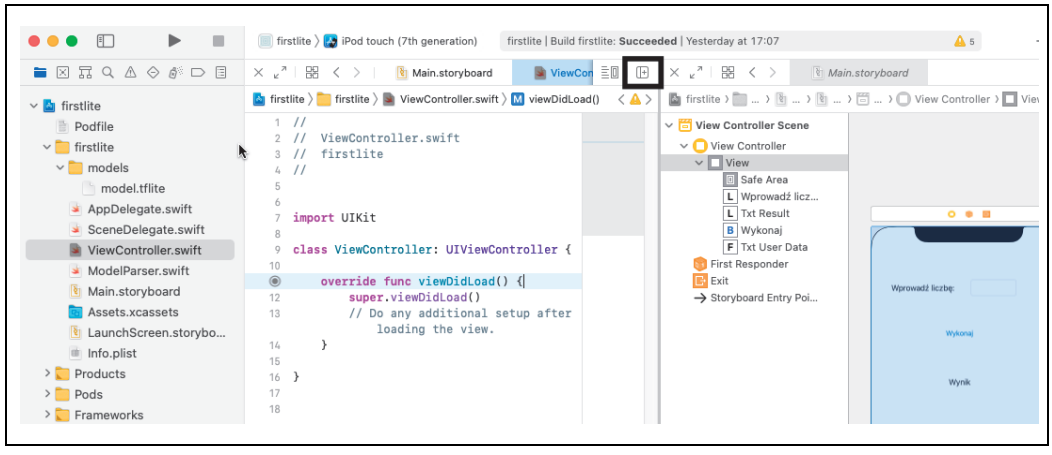
Rysunek 14.2. Opcje dla nowo tworzonego projektu



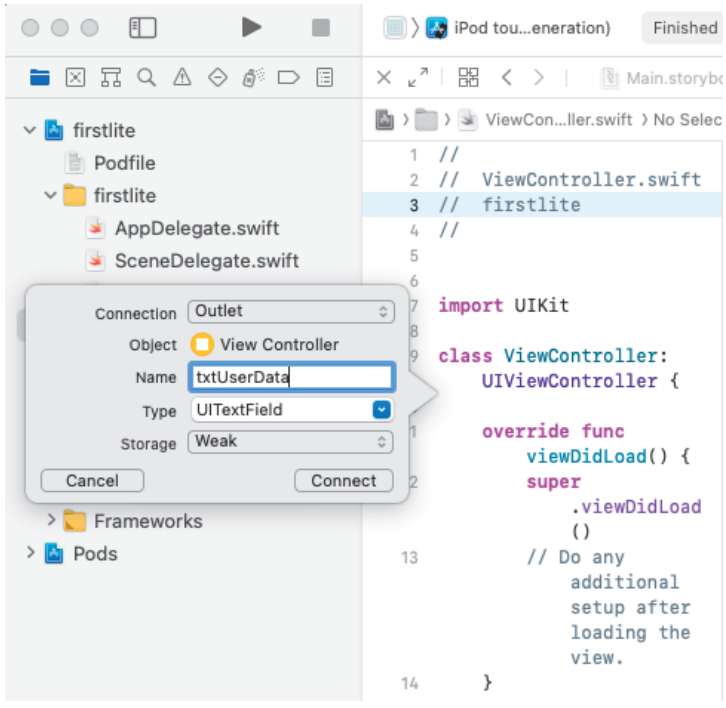
Rysunek 14.3. Struktura plików po uruchomieniu polecenia pod install



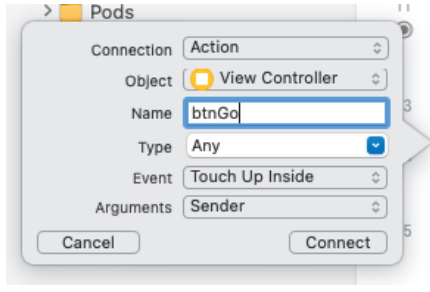
Rysunek 14.4. Dodawanie kontrolek do interfejsu graficznego



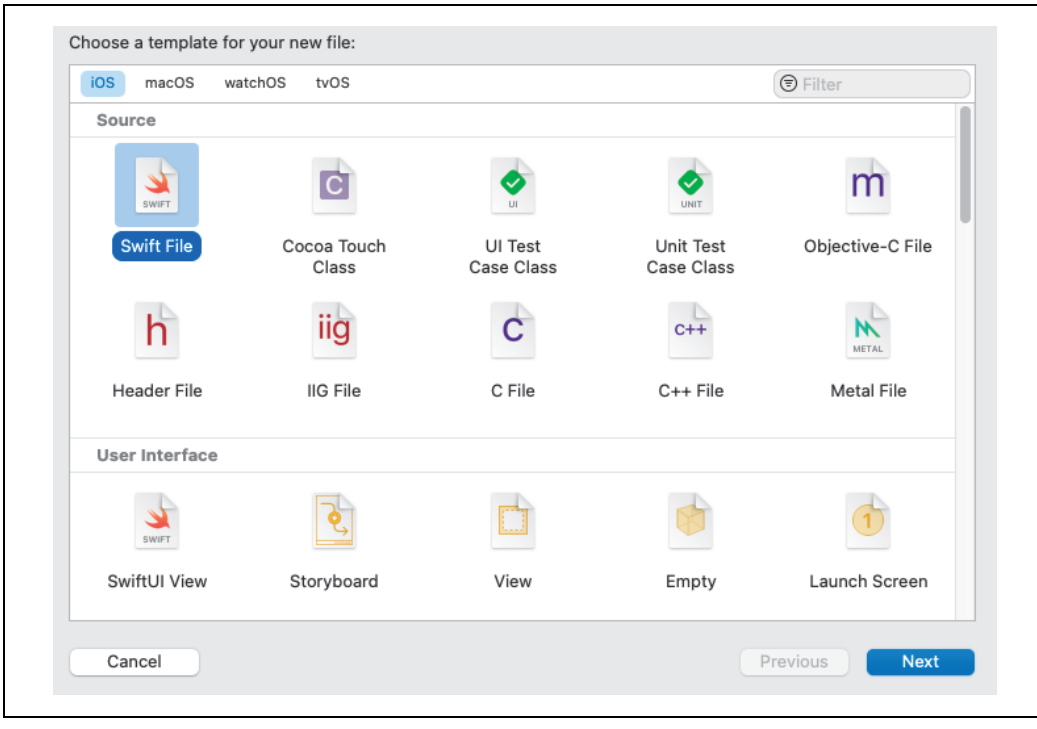
Rysunek 14.5. Podział ekranu



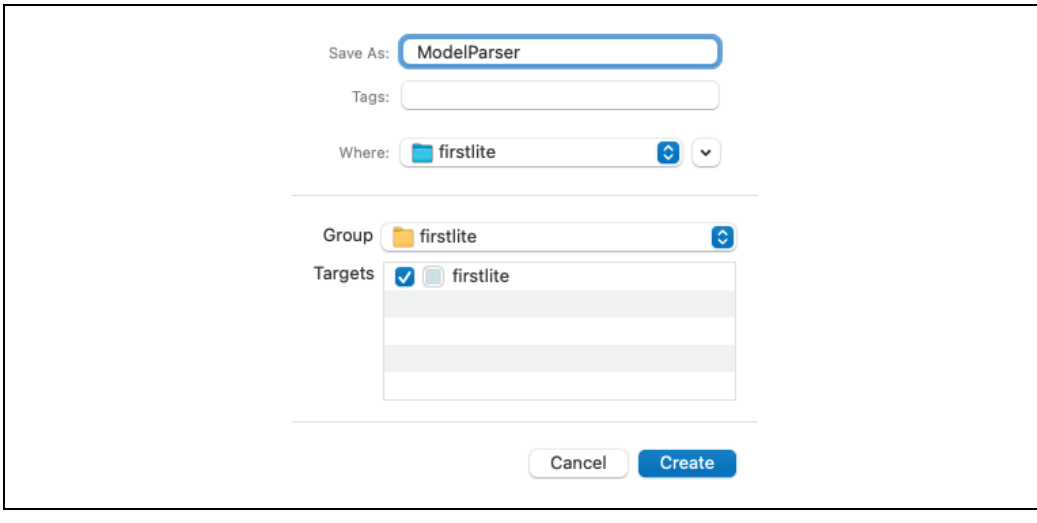
Rysunek 14.6. Tworzenie gniazda



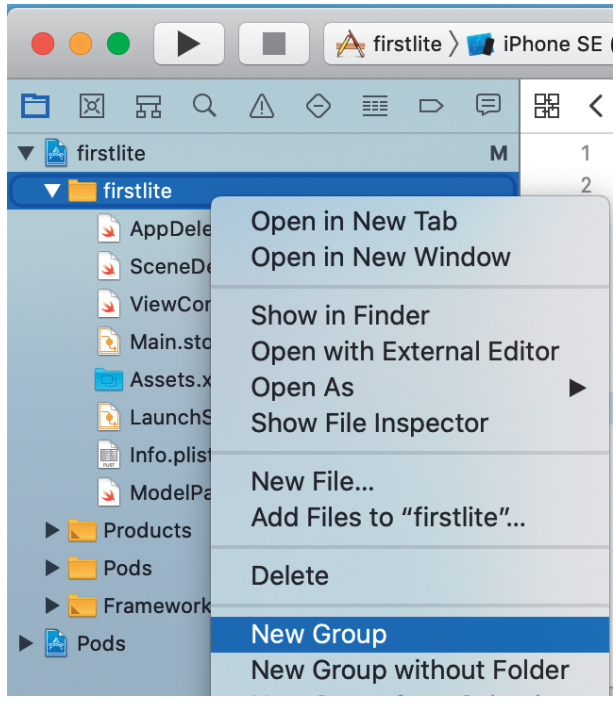
Rysunek 14.7. Dodawanie akcji



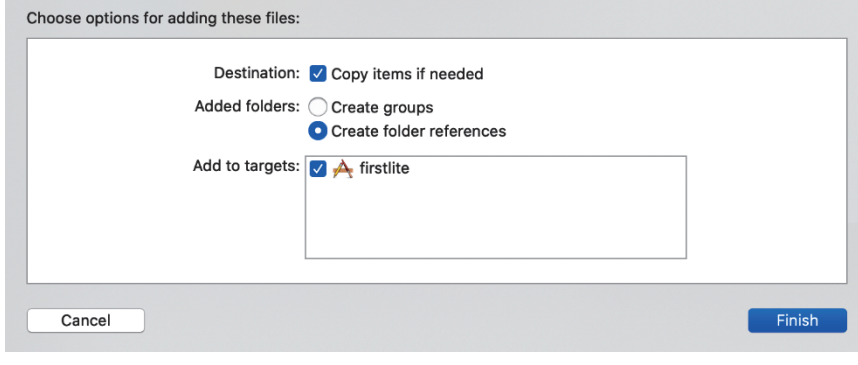
Rysunek 14.8. Dodawanie pliku języka Swift



Rysunek 14.9. Dodawanie pliku ModelParser.swift do projektu

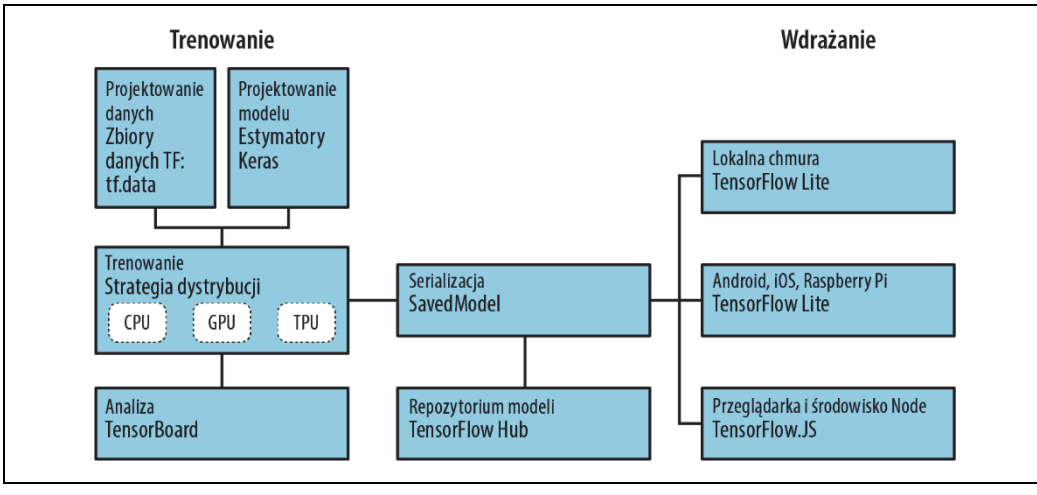


Rysunek 14.10. Dodawanie grupy do aplikacji

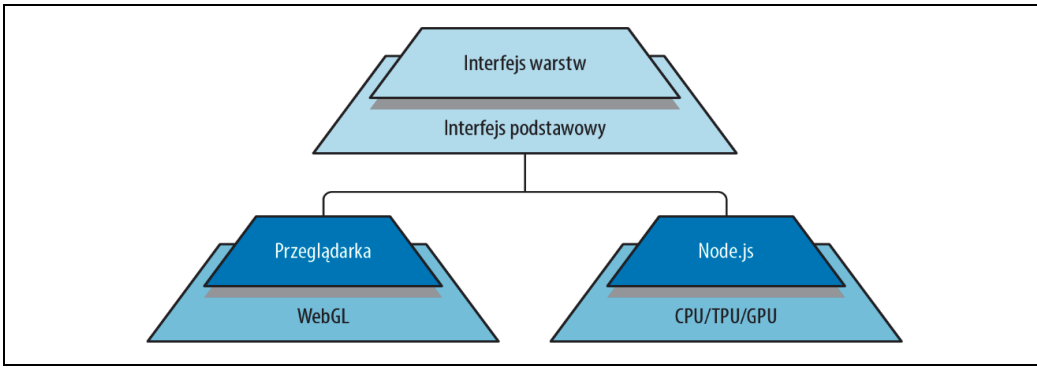


Rysunek 14.11. Dodawanie modelu do projektu

Rysunek 14.12. Uruchomienie aplikacji w symulatorze telefonu iPhone

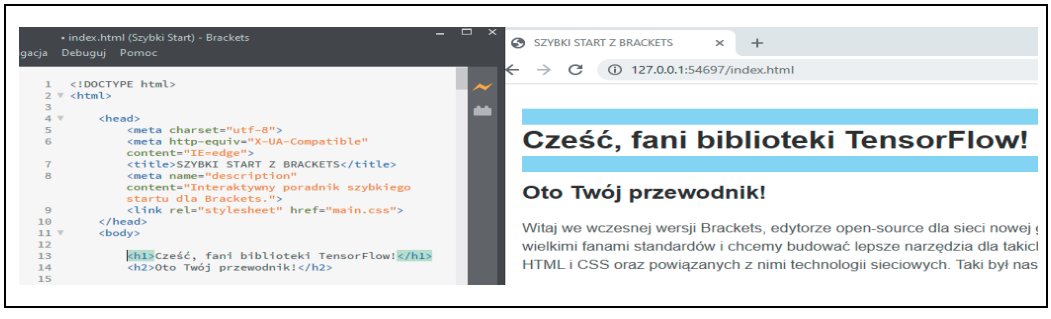


Rysunek 15.1. Środowisko TensorFlow

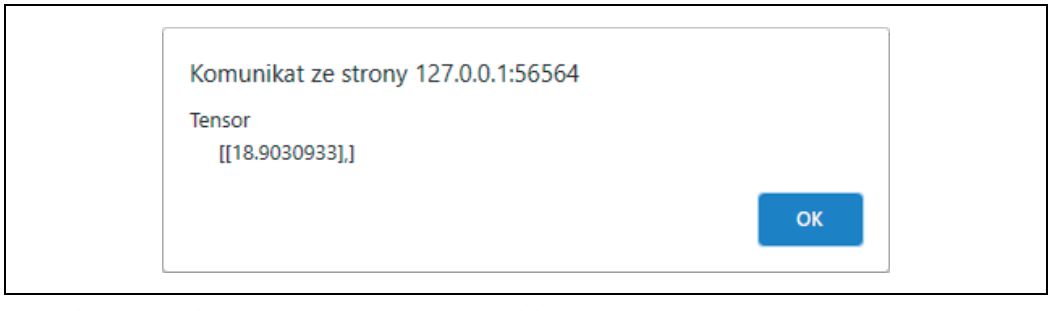


Rysunek 15.2. Wysokopoziomowa architektura biblioteki TensorFlow.js

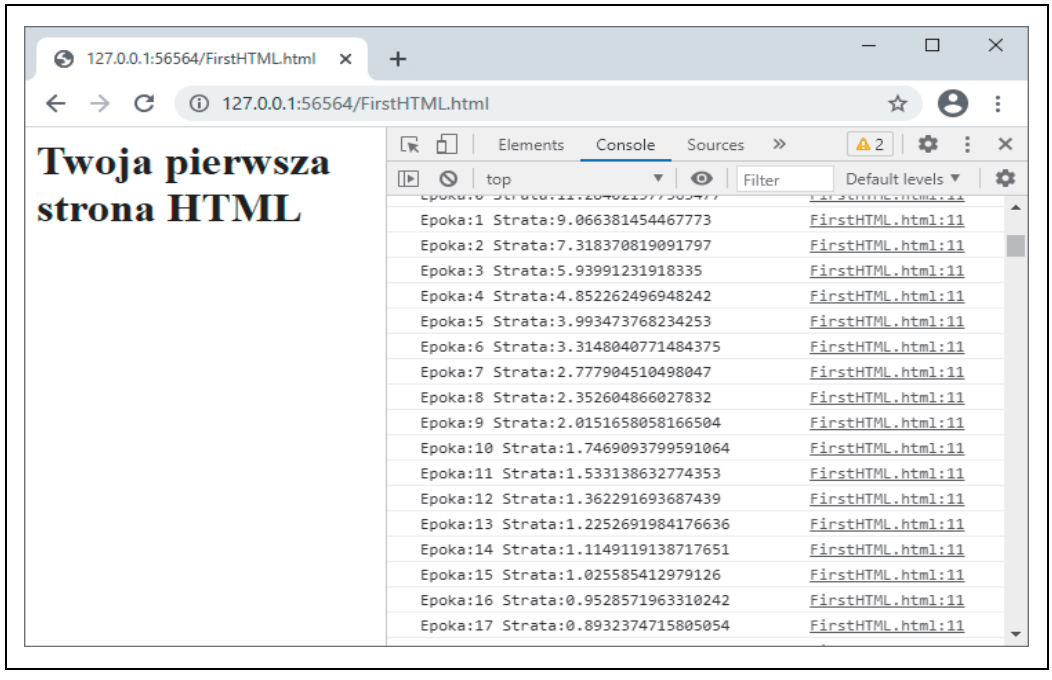
Rysunek 15.3. Powitalna strona środowiska Brackets



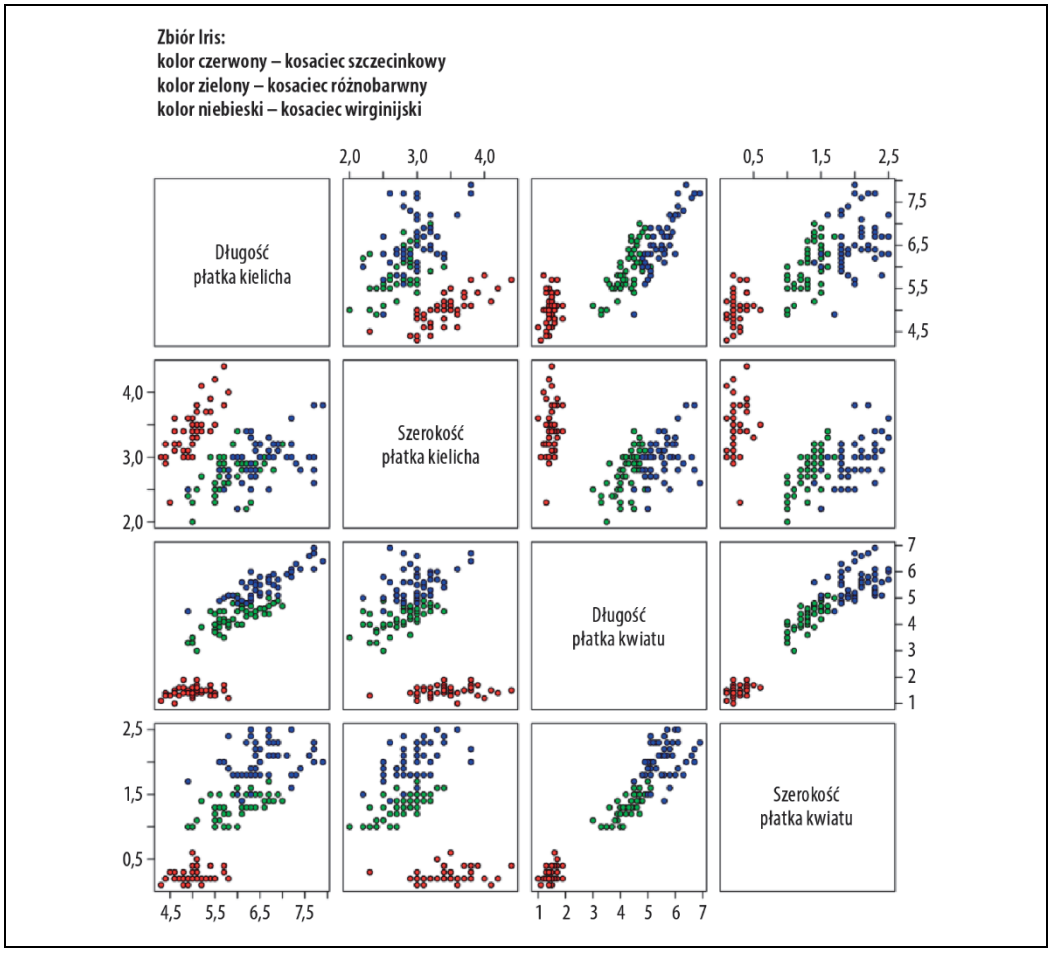
Rysunek 15.4. Aktualizowanie okna przeglądarki w czasie rzeczywistym



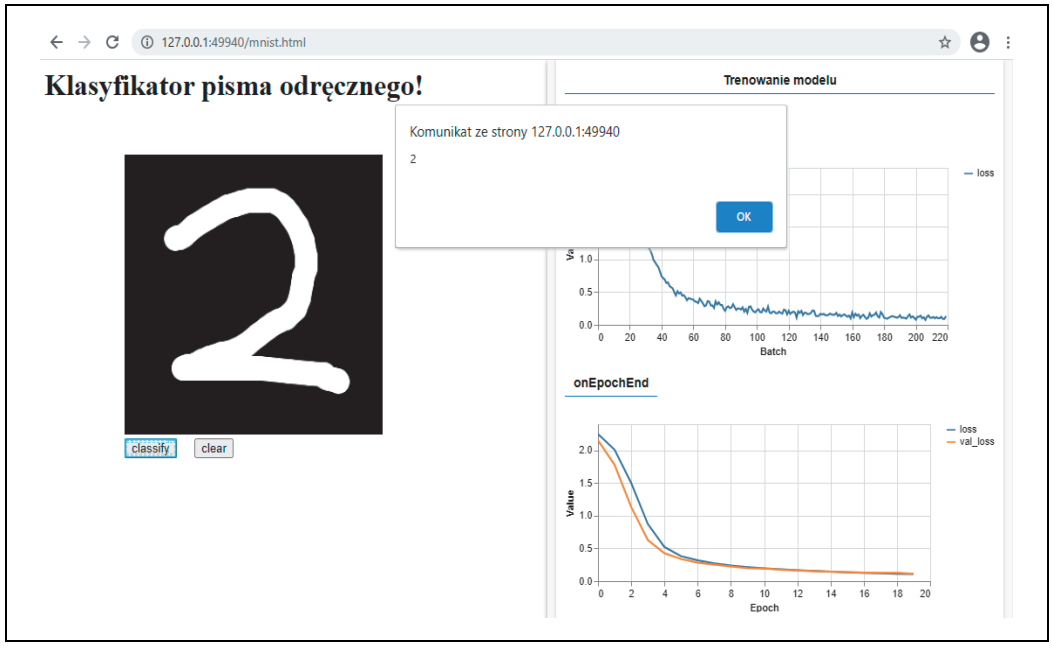
Rysunek 15.5. Wyniki prognozy po wytrenowaniu modelu



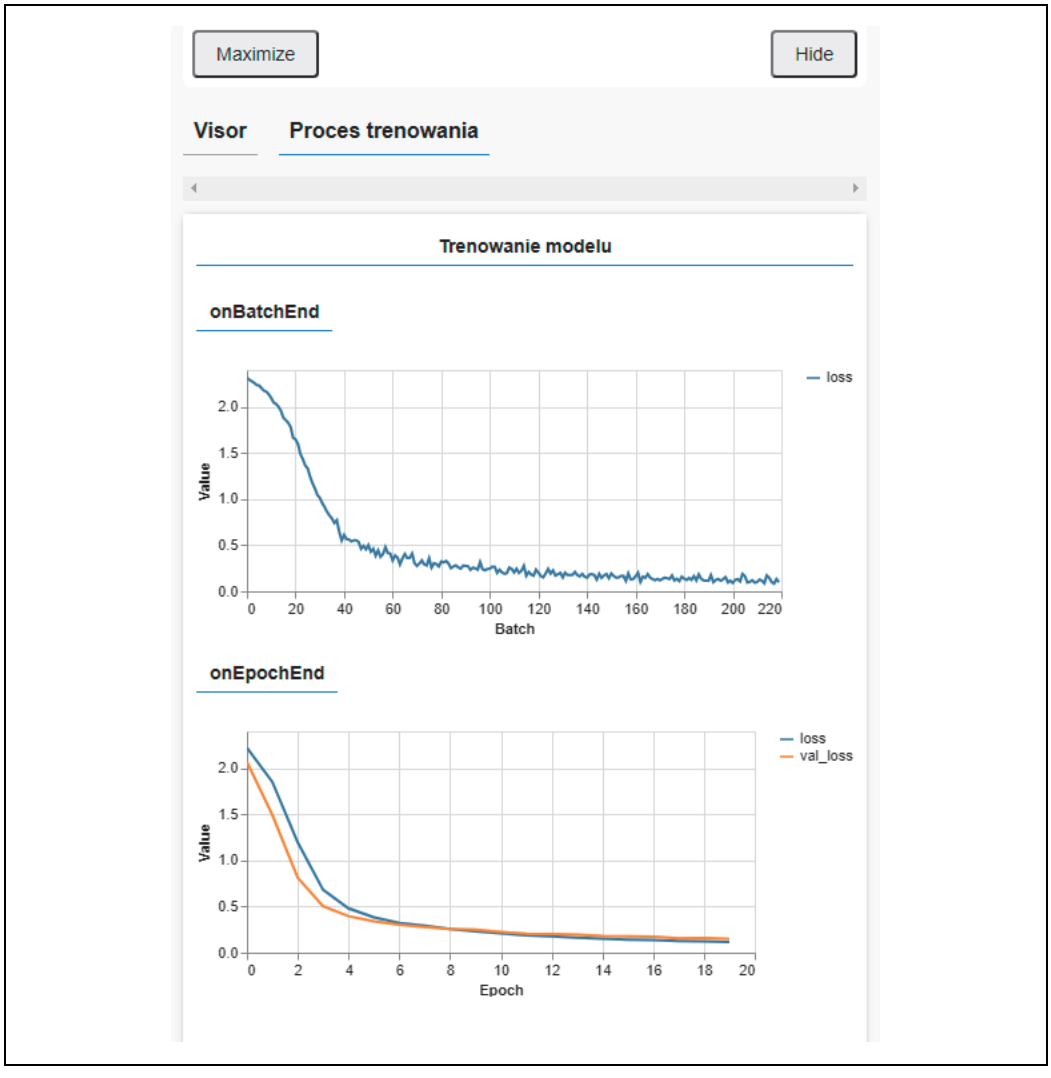
Rysunek 15.6. Dziennik zawierający wartości straty dla poszczególnych epok



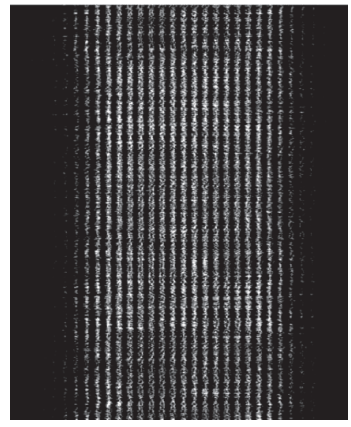
Rysunek 15.7. Wykres cech zbioru danych Iris (autor: Nicoguaro, obraz dostępny w Wikimedia Commons (https://oreil.ly/zgf7c))



Rysunek 16.1. Klasyfikator pisma odręcznego działający w przeglądarce



Rysunek 16.2. Użycie narzędzi wizualizacyjnych



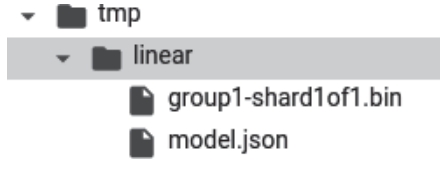
Rysunek 16.3. Fragment arkusza sprajtów mnist_images.png

Lister - [e:\ai\mnist\mnist_labels_uint8]

Plik Edytuj Opcje Kodowanie Pomoc 0 %

```
00000000: 00 00 00 00 00 00 00 01|00 00 00 00 00 01 00 00 | ................
00000010: 00 00 00 00 00 00 00 00|01 00 00 00 00 00 00 00 | ................
00000020: 00 00 00 00 01 00 00 00|00 01 00 00 00 00 00 00 | ................
00000030: 00 00 00 00 00 00 00 00|00 00 01 00 00 01 00 00 | ................
00000040: 00 00 00 00 00 00 01 00|00 00 00 00 00 00 00 00 | ................
00000050: 00 00 00 00 00 00 00 00|00 01 00 00 00 00 00 00 | ................
00000060: 00 00 01 00 01 00 00 00|00 00 00 00 00 00 00 00 | ................
00000070: 00 01 00 00 00 00 00 00|00 01 00 00 00 00 00 00 | ................
00000080: 00 00 00 00 01 00 00 00|00 00 00 00 00 00 00 00 | ................
00000090: 00 00 00 01 00 00 01 00|00 00 00 00 00 00 00 00 | ................
000000A0: 00 00 01 00 00 00 00 00|00 00 00 00 00 00 00 00 | ................
000000B0: 00 00 00 01 00 00 00 00|00 00 01 00 00 00 01 00 | ................
000000C0: 00 00 00 00 00 00 00 00|00 01 00 00 00 00 00 00 | ................
```

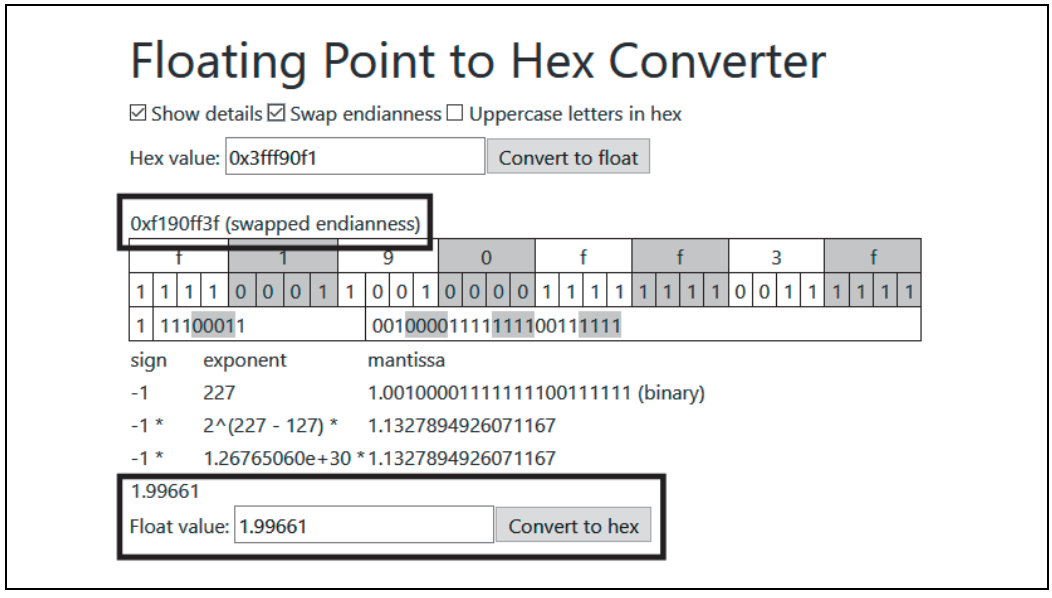
Rysunek 16.4. Zawartość pliku z etykietami



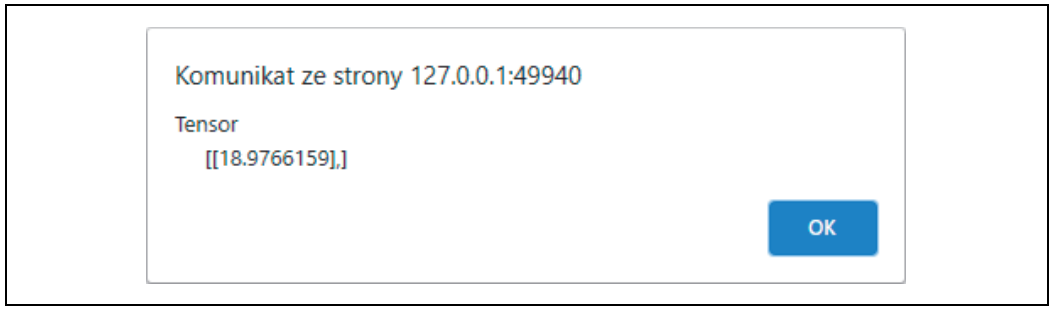
Rysunek 17.1. Wynikowe pliki uzyskane po uruchomieniu konwertera JS

```
00000000F1 90 FF 3F 5A 4F 7D BF
```

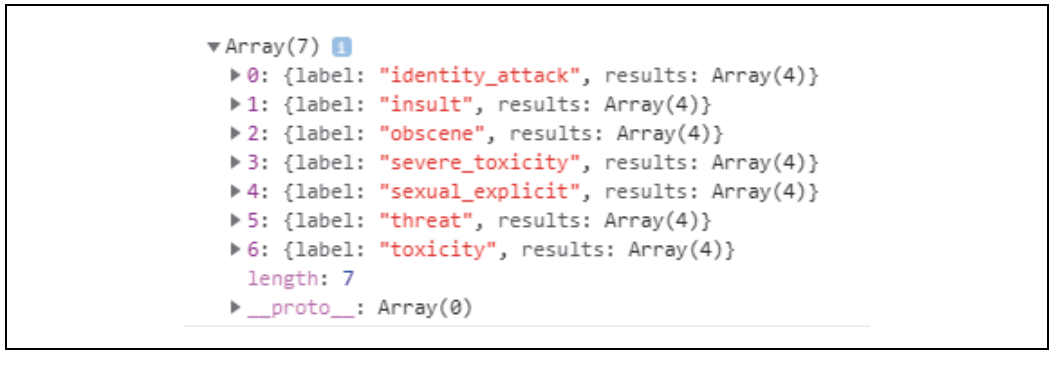
Rysunek 17.2. Zawartość pliku .bin



Rysunek 17.3. Konwersja wartości zmiennoprzecinkowej na szesnastkową

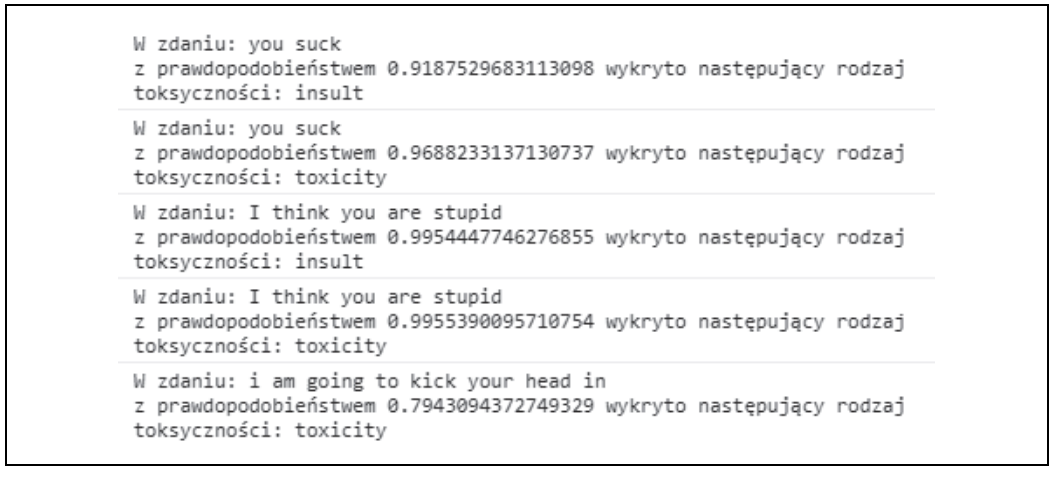


Rysunek 17.4. Wyniki wnioskowania



Rysunek 17.5. Wyniki prognozowania toksyczności

Rysunek 17.6. Przykładowe prawdopodobieństwa wystąpienia toksyczności



Rysunek 17.7. Wyniki działania klasyfikatora toksyczności dla przykładowych danych wejściowych

```
▼Array(3) i
  ▼0:
      className: "vase"
      probability: 0.6312612295150757
    ▶__proto__: Object
  ▼1:
      className: "pot, flowerpot"
      probability: 0.20017513632774353
    ▶__proto__: Object
  ▼2:
      className: "cup"
      probability: 0.11374199390411377
    ▶__proto__: Object
    length: 3
  ▶__proto__: Array(0)
```


Rysunek 17.8. Wyniki działania modelu wykorzystującego bibliotekę MobileNet

```
vase : 0.6312612295150757
pot, flowerpot : 0.20017513632774353
cup : 0.11374199390411377
```

Rysunek 17.9. Klasyfikowanie obrazu

```
▼Object
  ▼keypoints: Array(17)
    ▼0:
       part: "nose"
      ▶position: {x: 338.951882447714, y: 147.63976189876809}
       score: 0.9993178844451904
      ▶__proto__: Object
    ▶1: {score: 0.9989699125289917, part: "leftEye", position: {…}}
    ▶2: {score: 0.9965984225273132, part: "rightEye", position: {…}}
    ▶3: {score: 0.9560711979866028, part: "leftEar", position: {…}}
    ▶4: {score: 0.736717164516449, part: "rightEar", position: {…}}
    ▶5: {score: 0.9932397603988647, part: "leftShoulder", position: {…}}
    ▶6: {score: 0.9985883831977844, part: "rightShoulder", position: {…}}
    ▶7: {score: 0.9734750390052795, part: "leftElbow", position: {…}}
    ▶8: {score: 0.9767395853996277, part: "rightElbow", position: {…}}
    ▶9: {score: 0.9655238389968872, part: "leftWrist", position: {…}}
    ▶10: {score: 0.7352950572967529, part: "rightWrist", position: {…}}
    ▶11: {score: 0.9939306974411011, part: "leftHip", position: {…}}
    ▶12: {score: 0.9954432249069214, part: "rightHip", position: {…}}
    ▶13: {score: 0.9826021790504456, part: "leftKnee", position: {…}}
    ▶14: {score: 0.93722003698349, part: "rightKnee", position: {…}}
    ▶15: {score: 0.9435631632804871, part: "leftAnkle", position: {…}}
    ▶16: {score: 0.860307514667511, part: "rightAnkle", position: {…}}
     length: 17
```

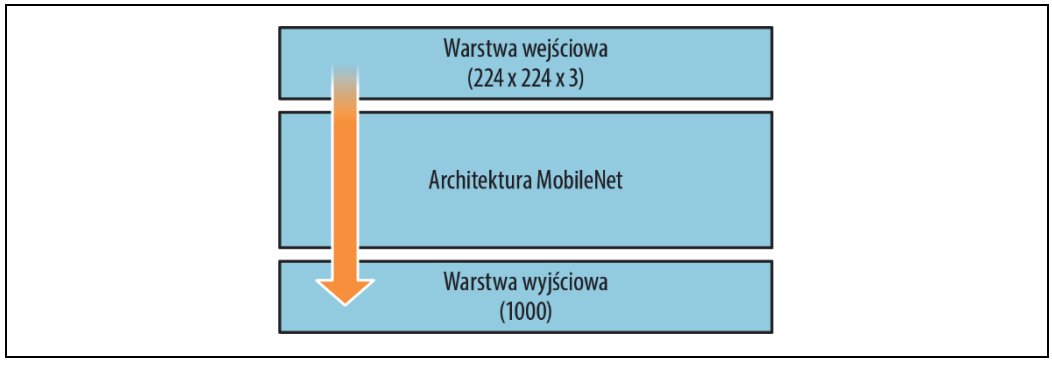

Rysunek 17.10. Zwrócone punkty charakteryzujące części ciała

Rysunek 17.11. Wyznaczanie i opisywanie części ciała

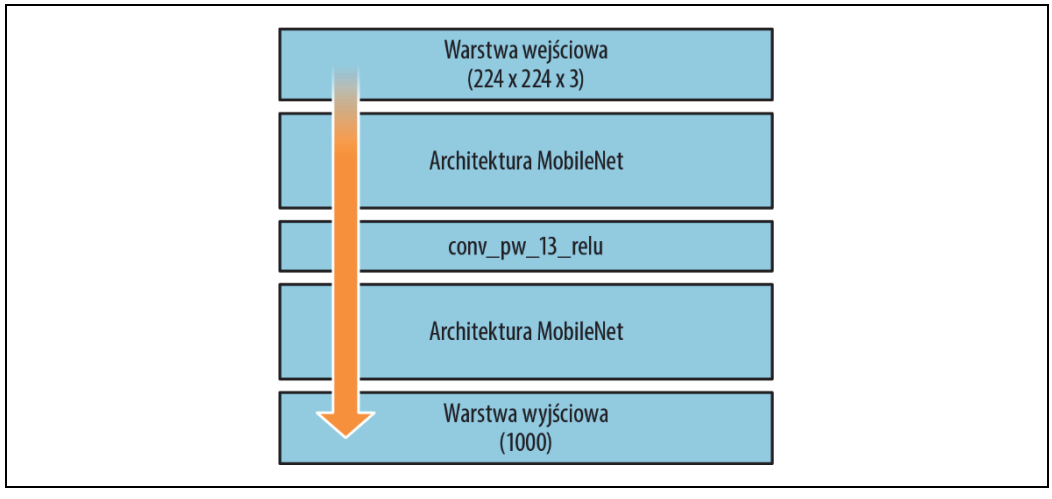
Rysunek 17.12. Użycie biblioteki PoseNet z obrazem przedstawiającym twarz

```
_________________________________________________________________ tfjs@latest:2
_____
conv_pw_13_bn (BatchNormaliz [null,7,7,256]           1024 tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
conv_pw_13_relu (Activation) [null,7,7,256]           0    tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
global_average_pooling2d_1 ( [null,256]               0    tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
reshape_1 (Reshape)          [null,1,1,256]           0    tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
dropout (Dropout)            [null,1,1,256]           0    tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
conv_preds (Conv2D)          [null,1,1,1000]               tfjs@latest:2
257000
_________________________________________________________________ tfjs@latest:2
_____
act_softmax (Activation)     [null,1,1,1000]          0    tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
_____
reshape_2 (Reshape)          [null,1000]              0    tfjs@latest:2
================================================================= tfjs@latest:2
======
Total params: 475544                                       tfjs@latest:2
Trainable params: 470072                                   tfjs@latest:2
Non-trainable params: 5472                                 tfjs@latest:2
_________________________________________________________________ tfjs@latest:2
```

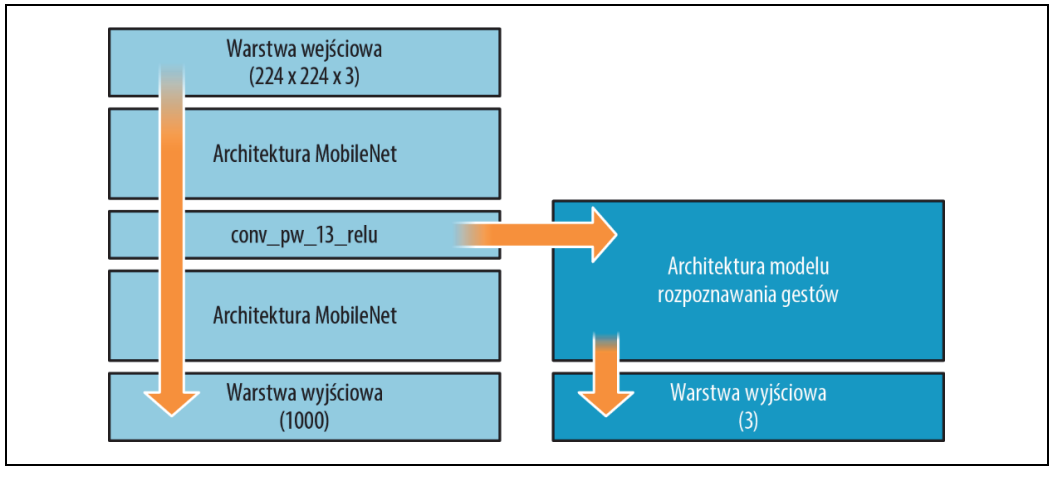
Rysunek 18.1. Wynik działania metody mobilenet.summary dla modelu JSON z architekturą MobileNet



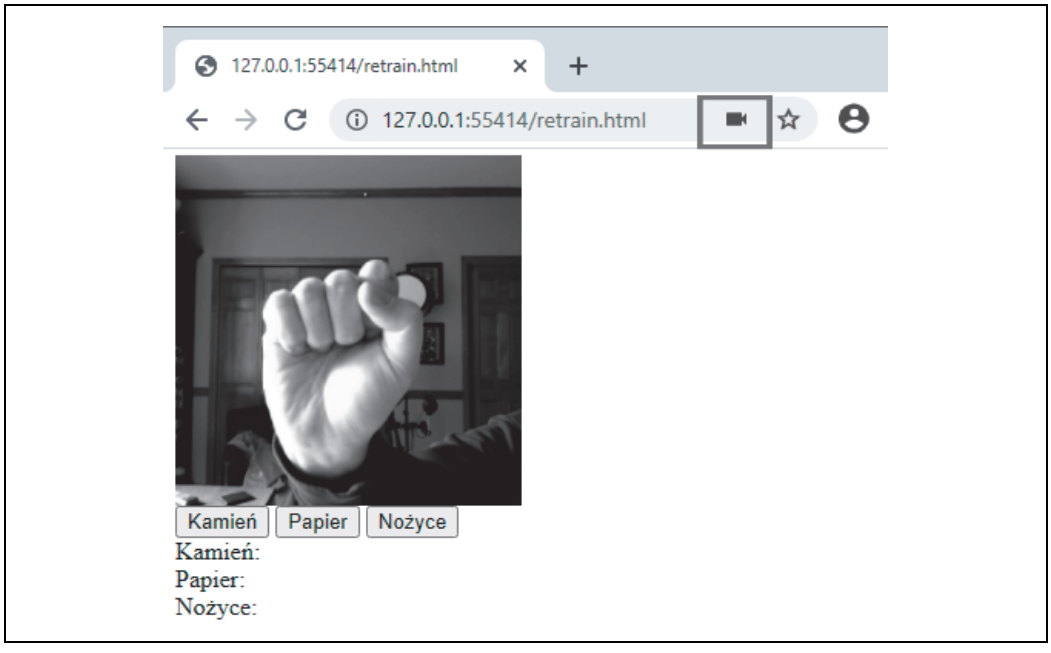
Rysunek 18.2. Wysokopoziomowy schemat architektury MobileNet



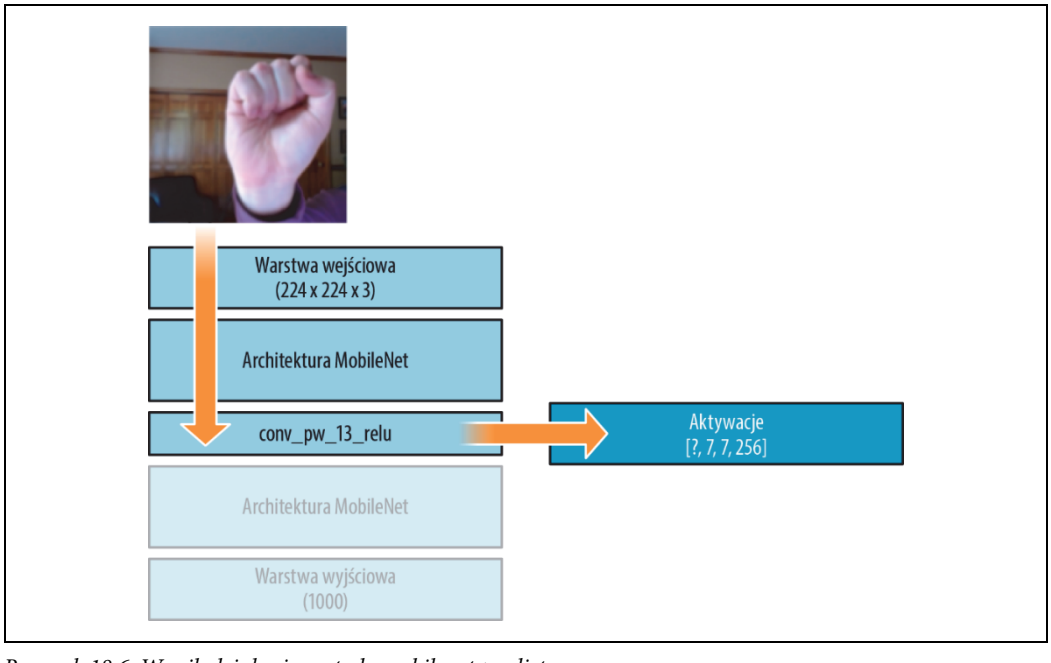
Rysunek 18.3. Wysokopoziomowy schemat architektury MobileNet z uwzględnieniem warstwy conv_pw_13_relu



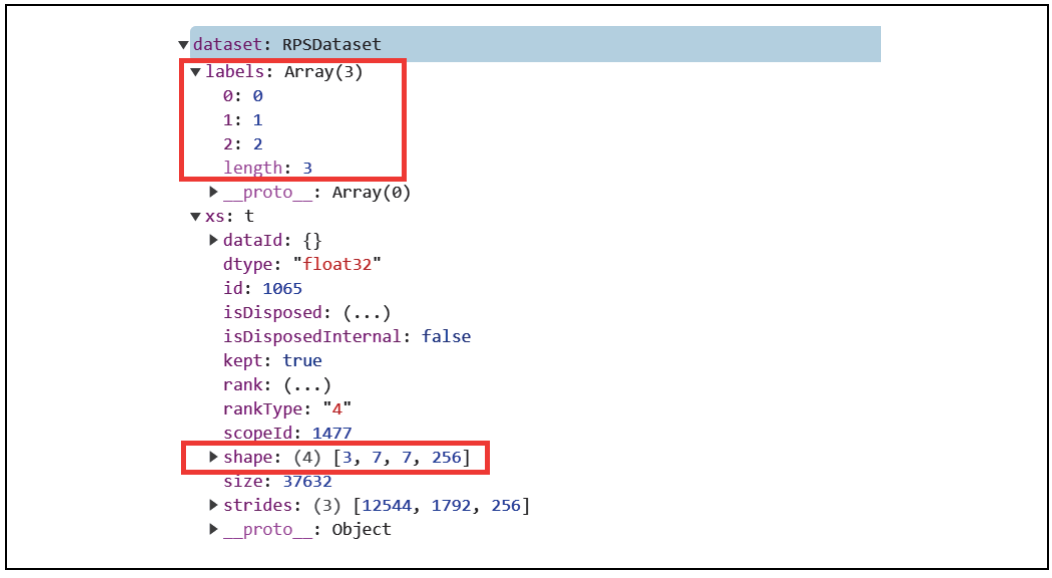
Rysunek 18.4. Przykład uczenia transferowego: przekazanie danych z warstwy conv_pw_13_relu do nowego modelu



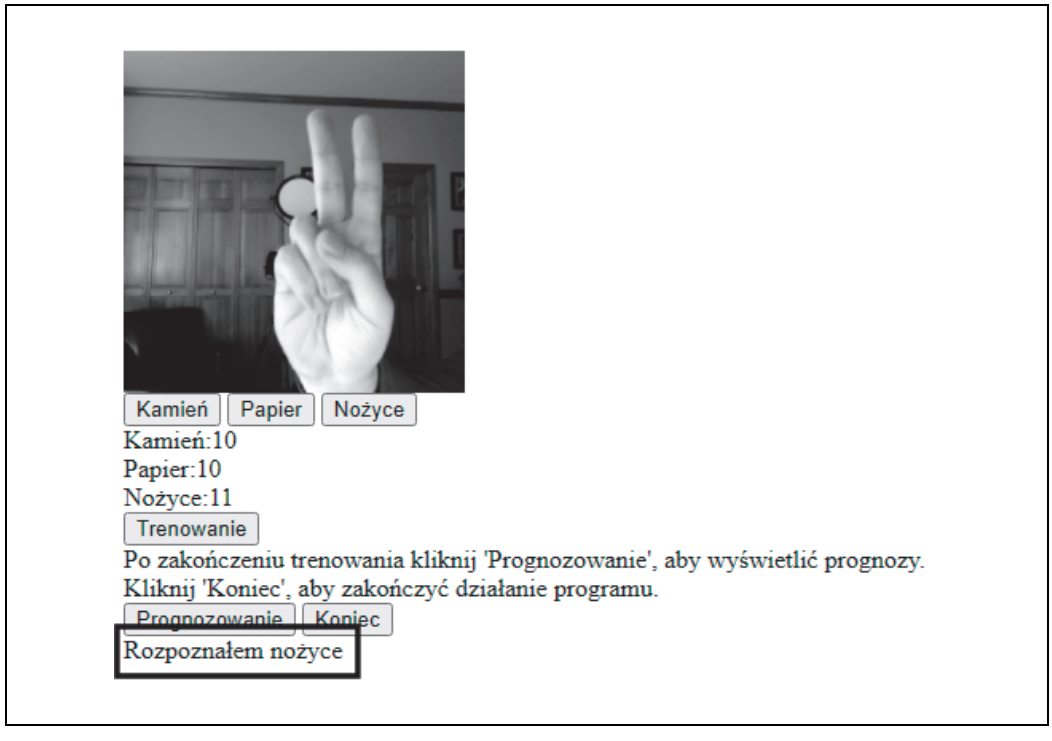
Rysunek 18.5. Podgląd z kamery internetowej



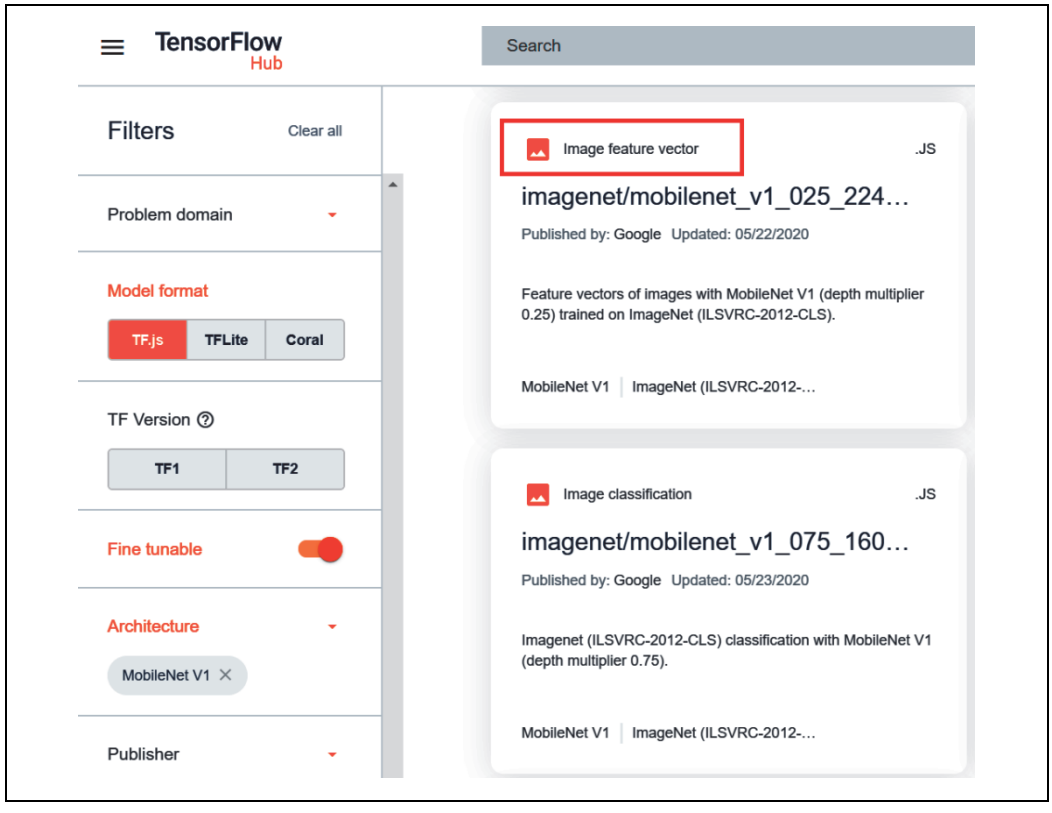
Rysunek 18.6. Wynik działania metody mobilenet.predict



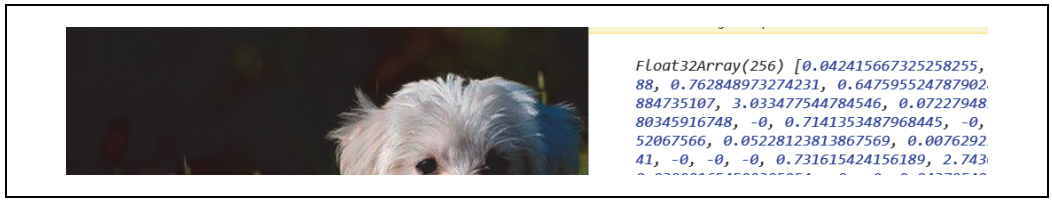
Rysunek 18.7. Analiza zbioru danych



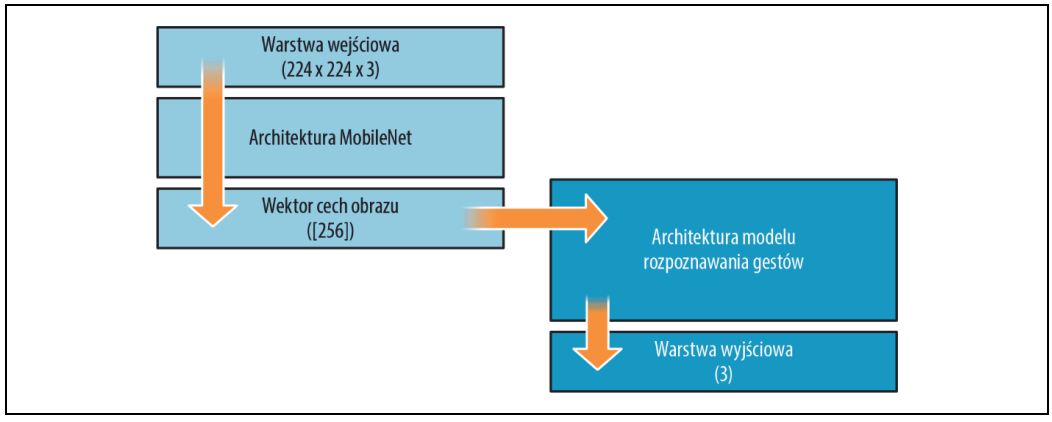
Rysunek 18.8. Przeprowadzanie wnioskowania w przeglądarce przy użyciu wytrenowanego modelu



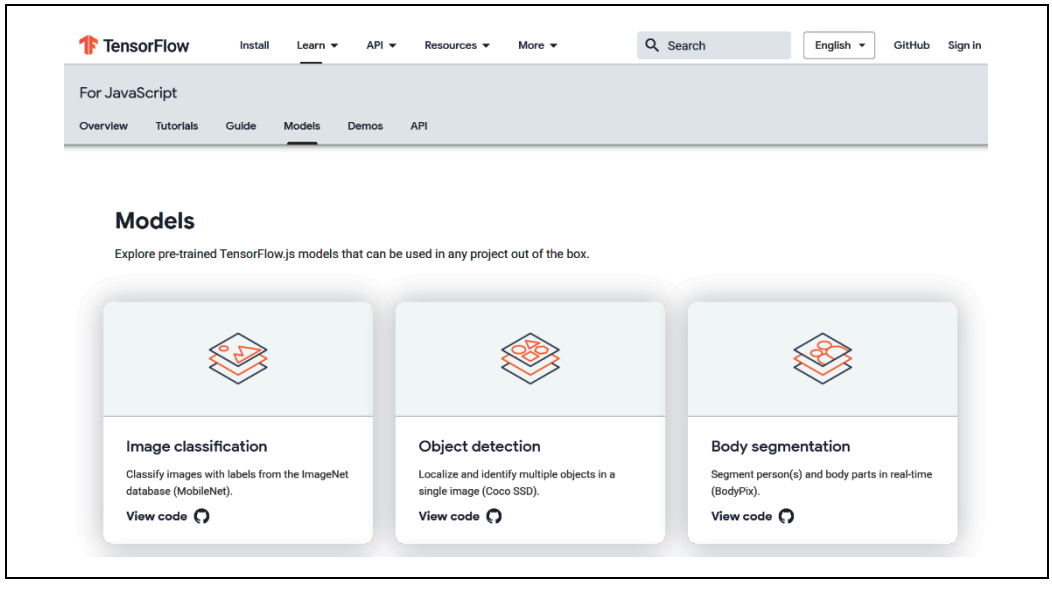
Rysunek 18.9. Wykorzystanie strony https://tfhub.dev *do wyszukania modeli działających w języku JavaScript*



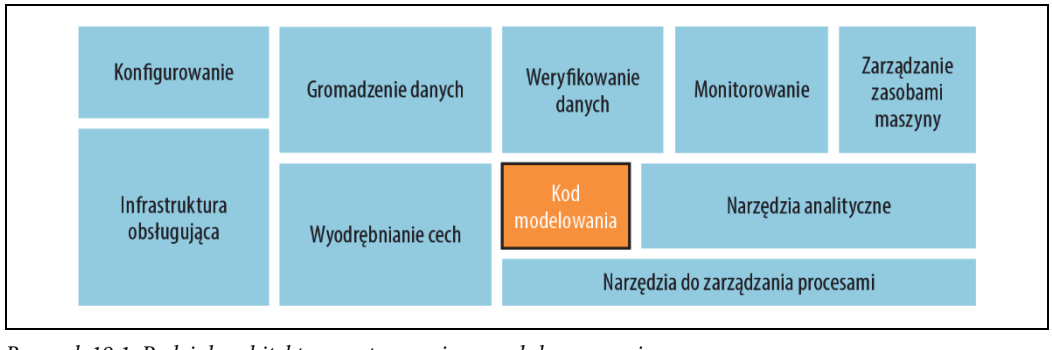
Rysunek 18.10. Wyniki wyświetlone w konsoli



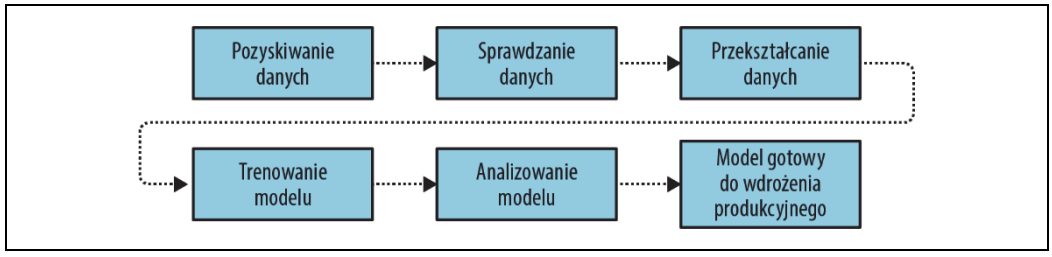
Rysunek 18.11. Uczenie transferowe przy użyciu wektorów cech obrazu



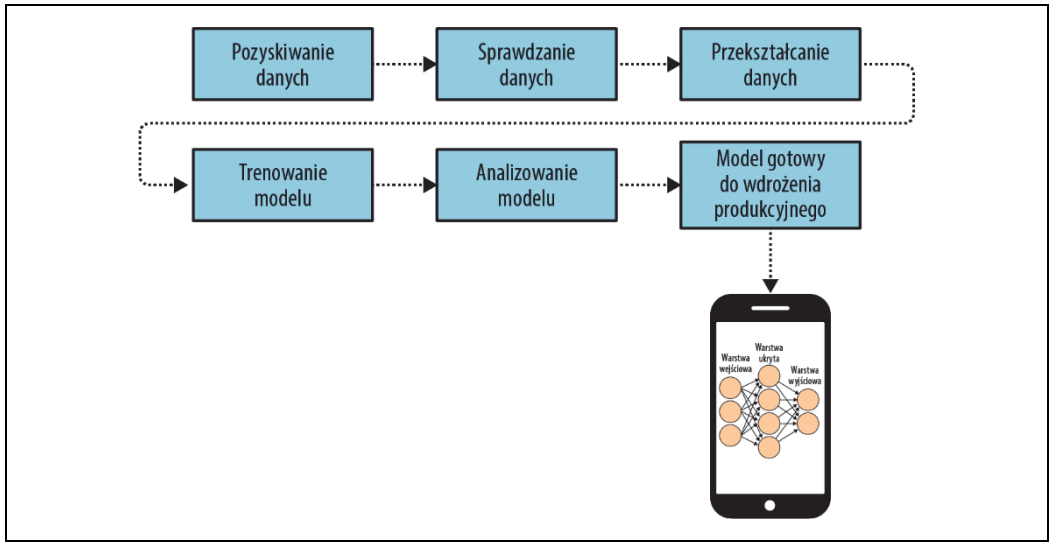
Rysunek 18.12. Przeglądanie modeli na stronie TensorFlow.org



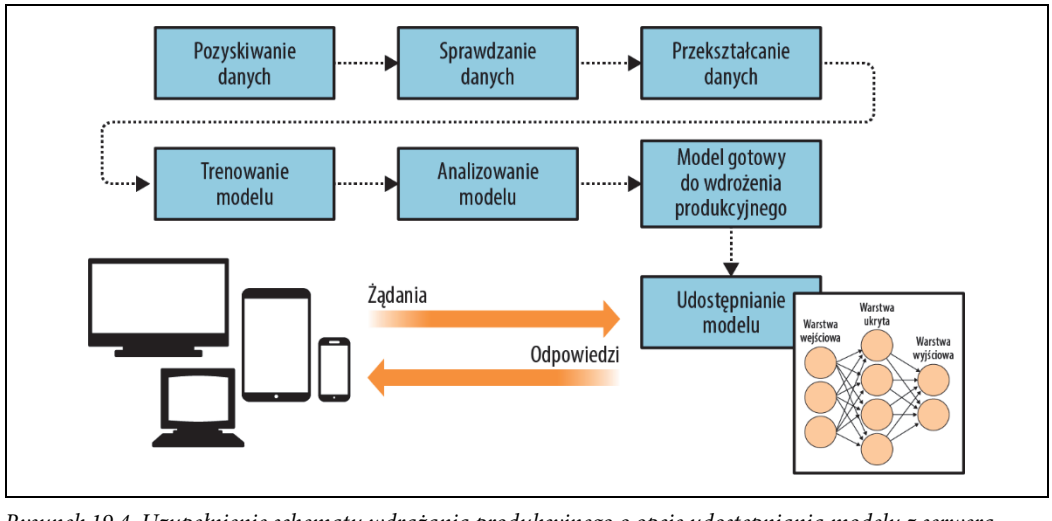
Rysunek 19.1. Podział architektury systemowej na moduły w uczeniu maszynowym



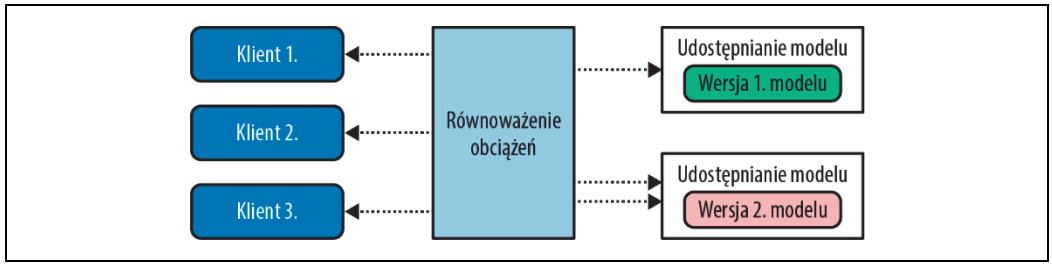
Rysunek 19.2. Schemat produkcyjnego wdrażania modeli uczenia maszynowego



Rysunek 19.3. Wdrażanie modelu produkcyjnego w urządzeniu mobilnym



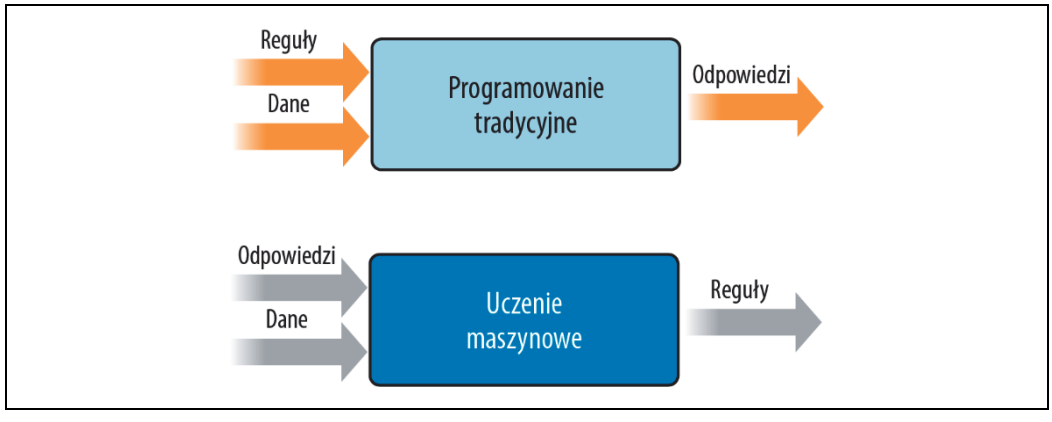
Rysunek 19.4. Uzupełnienie schematu wdrażania produkcyjnego o opcję udostępniania modelu z serwera



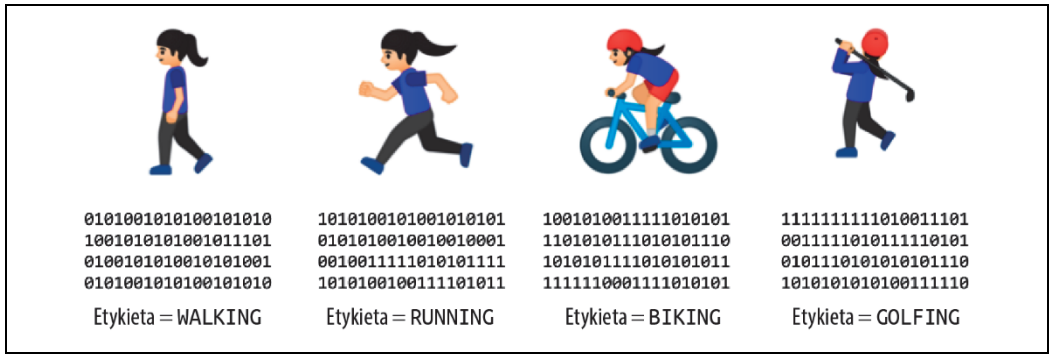
Rysunek 19.5. Użycie usługi TensorFlow Serving do udostępniania wielu wersji modelu

```
lmoroney@lmoroney-ASM100:~/serving/tensorflow
instances": [1.0, 2.0, 5.0]}'     -X POST htt
{
    "predictions": [2.5, 3.0, 4.5
    ]
}lmoroney@lmoroney-ASM100:~/serving/tensorflo
```

Rysunek 19.6. Wyniki działania modelu udostępnianego przez usługę TensorFlow Serving

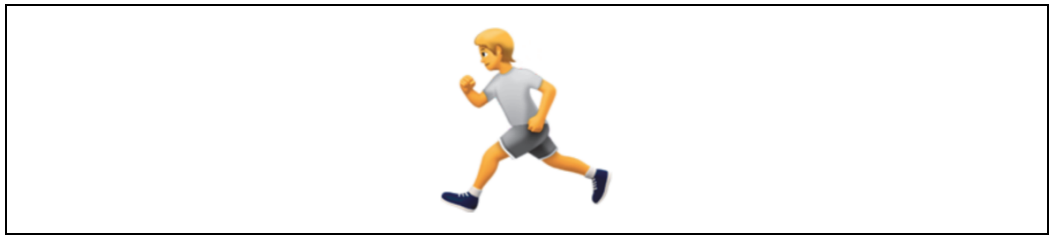


Rysunek 20.1. Porównanie programowania tradycyjnego i uczenia maszynowego

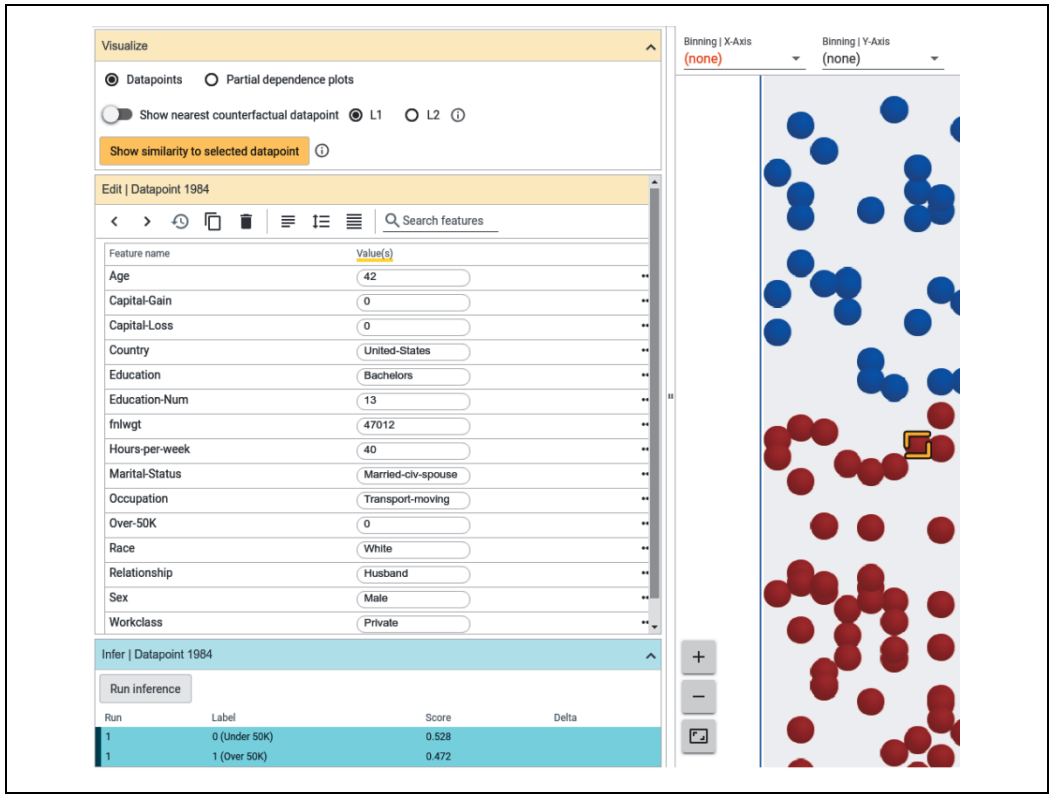


Rysunek 20.2. Ikony wykorzystywane podczas prezentacji sposobu działania uczenia maszynowego

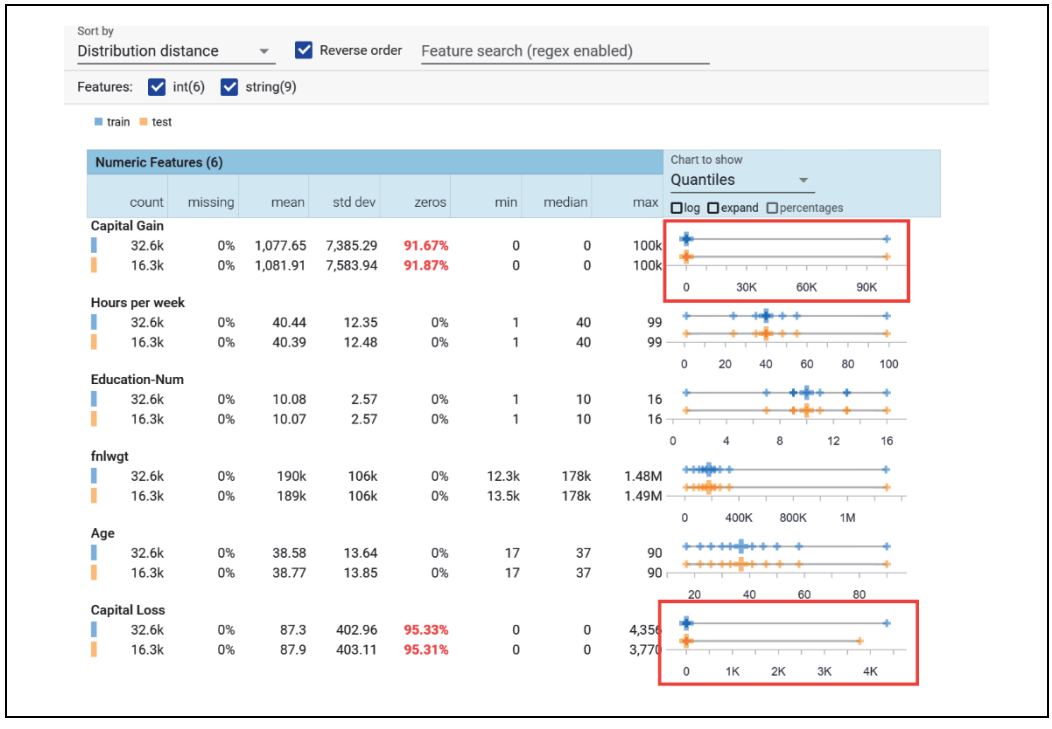
Rysunek 20.3. Tekst zawierający emotikony



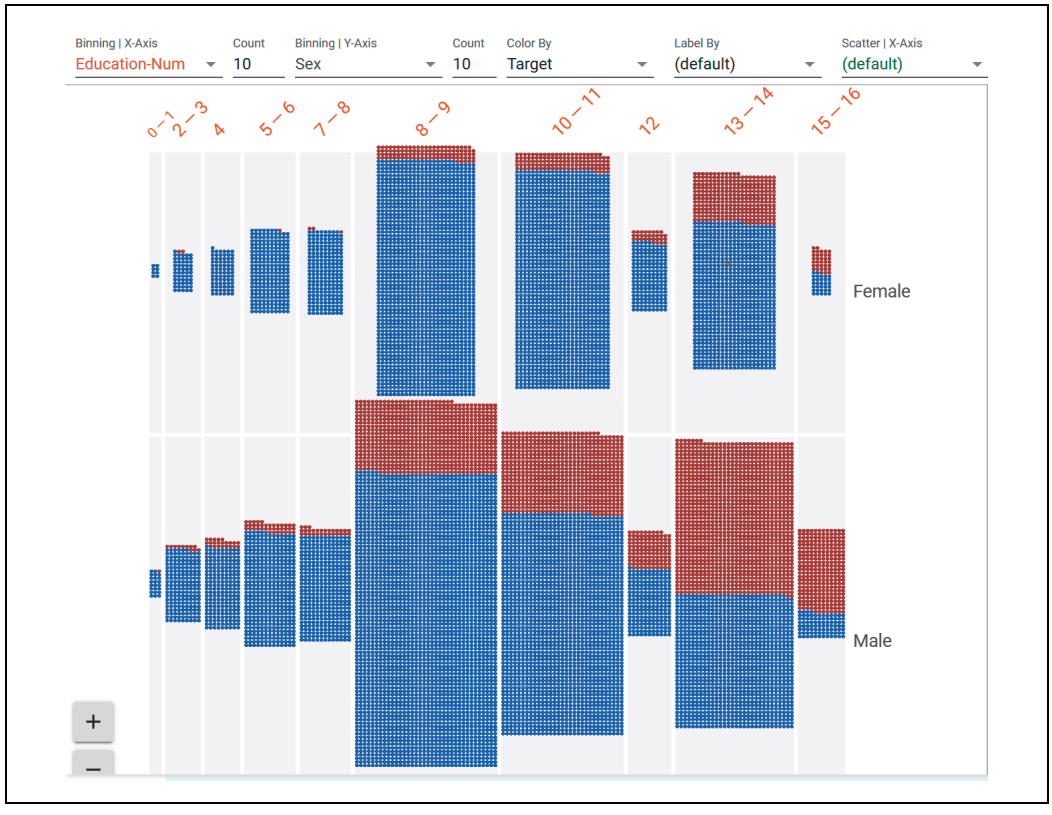
Rysunek 20.4. Emotikon „biegnącej osoby" z systemu iOS 13.3



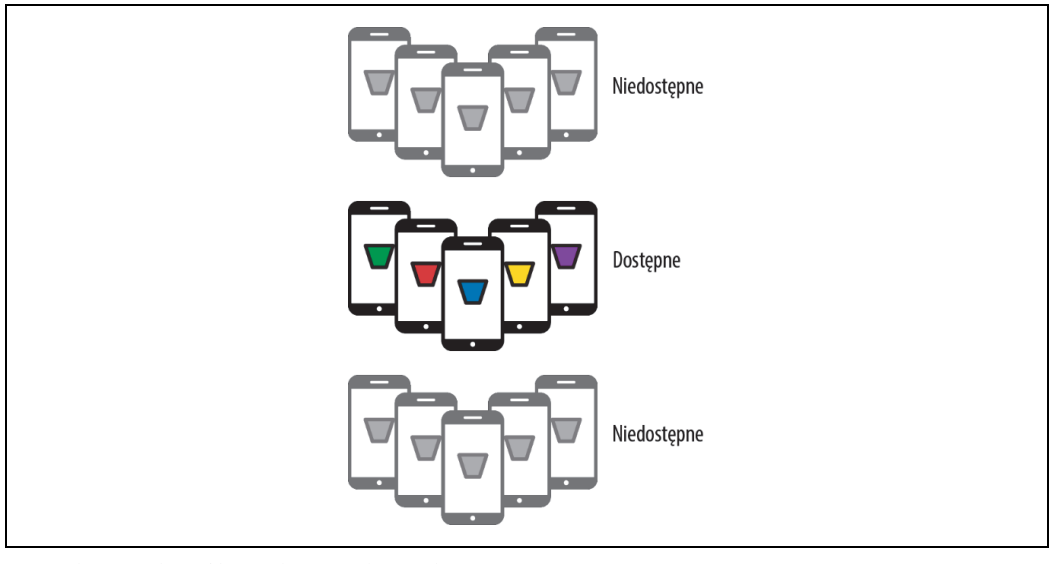
Rysunek 20.5. Użycie narzędzia What-If



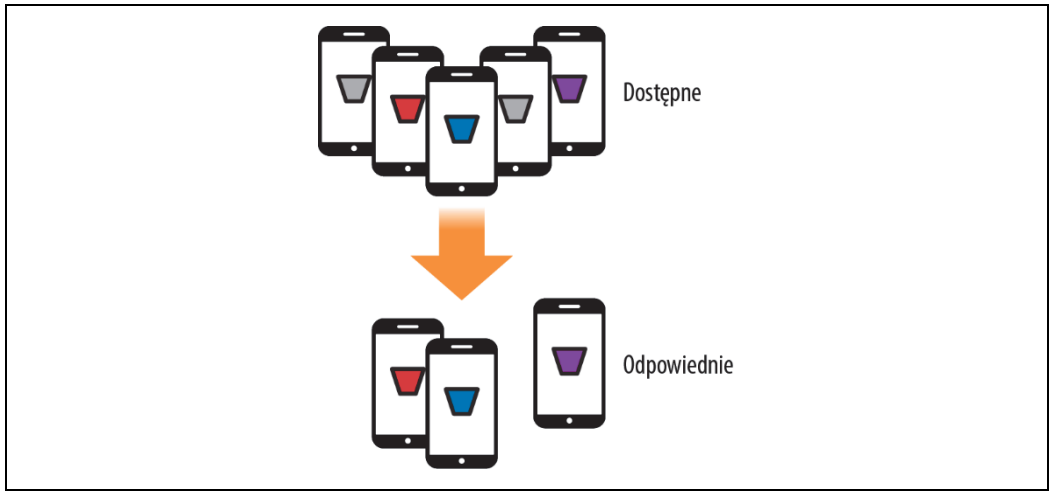
Rysunek 20.6. Użycie narzędzia Facets do analizy zbioru danych



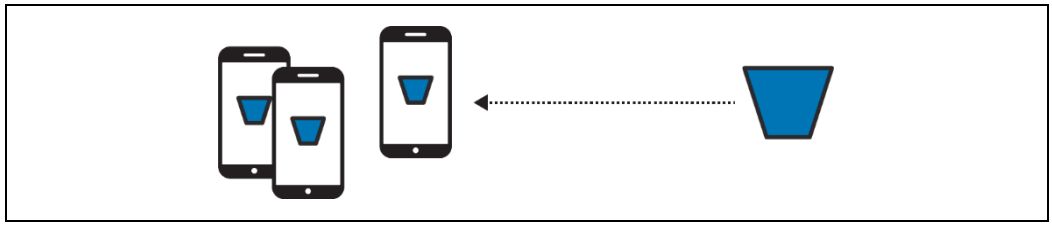
Rysunek 20.7. Szczegółowa analiza danych za pomocą narzędzia Facets



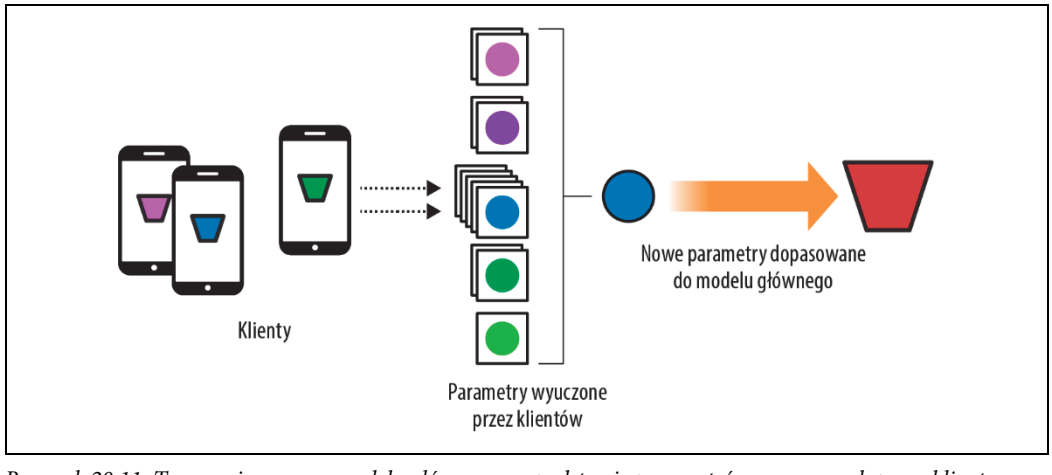
Rysunek 20.8. Identyfikacja dostępnych urządzeń



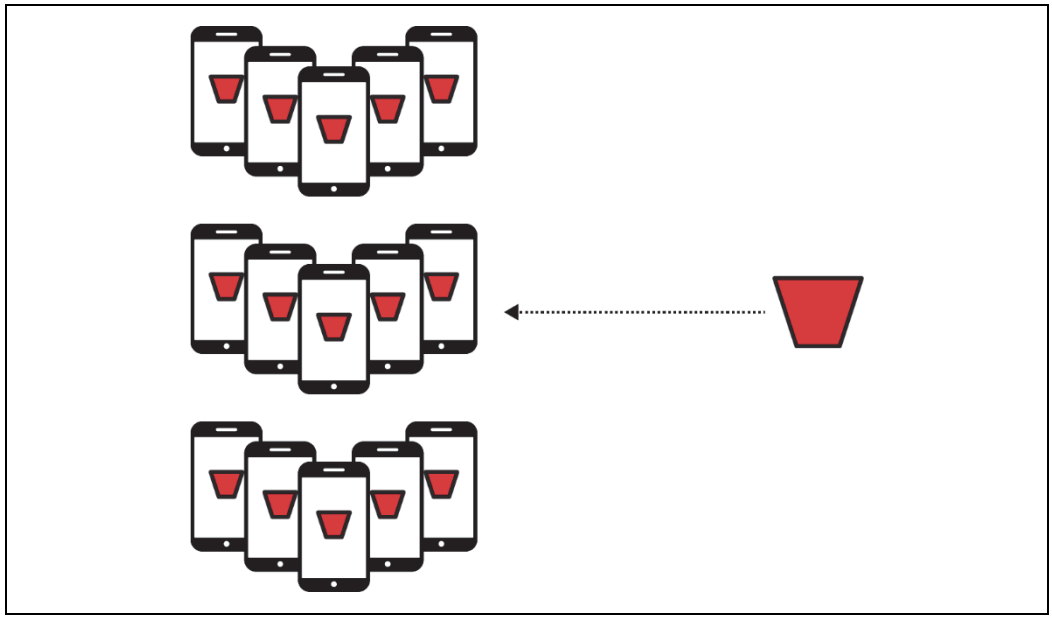
Rysunek 20.9. Wybór odpowiednich urządzeń spośród tych, które są dostępne



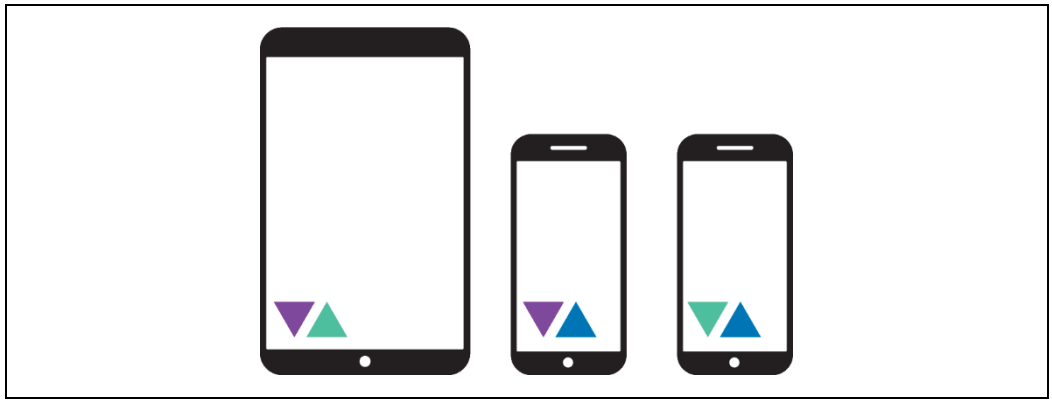
Rysunek 20.10. Instalowanie modelu w urządzeniach



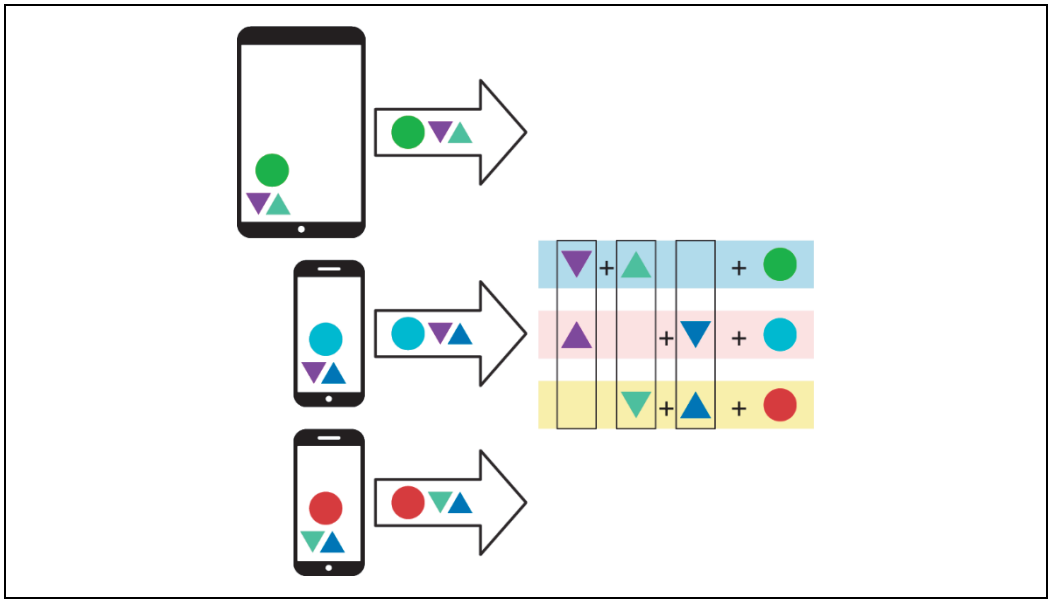
Rysunek 20.11. Tworzenie nowego modelu głównego na podstawie parametrów wyuczonych przez klienty



Rysunek 20.12. Model główny zostaje zainstalowany we wszystkich urządzeniach klienckich



Rysunek 20.13. Urządzenia połączone w systemy partnerskie



Rysunek 20.14. Wysyłanie wartości na serwer z użyciem bezpiecznej agregacji