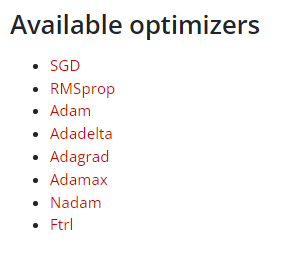
**Uzupełnienie zajęć (Budowa modelu (na podstawie istniejącego modelu) z 2 kwietnia**

**model.compile** (kompilowanie modelu)- definiuje funkcję straty, optymalizator i metryki. Do trenowania potrzebny jest skompilowany model (ponieważ trening wykorzystuje funkcję straty i optymalizator).

Optymalizacja Adam to stochastyczna metoda zstępowania gradientu, która opiera się na adaptacyjnej estymacji momentów pierwszego i drugiego rzędu.



Mając zdefiniowany model, potrzebujemy funkcji oceny naszego modelu. Najczęściej nazywa się ją funkcją straty (loss function). Jest to funkcja dzięki, dzięki której wiemy czy model jest dobry czy zły, na jej podstawie dokonujemy optymalizacji. Informuje ona o ilości popełnianych błędów przez nasz model, jeżeli wartość jest wysoka to znaczy, że nasz model wymaga jeszcze uczenia, czyli dostosowania wag z macierzy W.

**model.fit** (dopasowanie modelu)- mierzy, jak dobrze model uczenia maszynowego uogólnia dane podobne do tych, na których został przeszkolony. fit(): dopasowanie danych treningowych. W przypadku uczenia nadzorowanego przyjmuje dwa argumenty: dane x i etykiety y (np. model.fit(x,y).

**„i”** to zmienna tymczasowa, której używa się do przechowywania wartości całkowitej bieżącej pozycji w zakresie pętli for, której zakres obejmuje tylko pętlę for.

W napisach Pythona ukośnik odwrotny „/” to znak specjalny. **„\t”** oznacza tabulator -> do określenia przerw między naszymi wynikowymi wartościami.

**model.predict** Generuje prognozy wyjściowe dla próbek wejściowych.

Obliczenia są wykonywane w partiach. Ta metoda została zaprojektowana do przetwarzania wsadowego dużej liczby danych wejściowych. Nie jest ona przeznaczona do stosowania wewnątrz pętli, które iterują po danych i przetwarzają małe liczby danych wejściowych naraz.

**02.04.2023 (grupa 2) & 23.04.2023 (grupa 1)**

**Miary dokładności uczenia dla problemów klasyfikacyjnych**

* Macierz konfuzji

Fałsz

Prawda

TP

FP

Pozytywne

Oszacowane

TN

FN

Negatywne

* parametr (ang. true negative) TN
* parametr (ang. false negative) FN
* parametr (ang. positive) Pos=TP+EP
* parametr (ang. negative) Neg= TN+FN
* parametr (ang. true) True= TP+TN
* parametr (ang. false) False= FP+FN
* parametr (ang. true positive) TP
* parametr (ang. false positive) FP

Czułość = TP/(TP+FN)

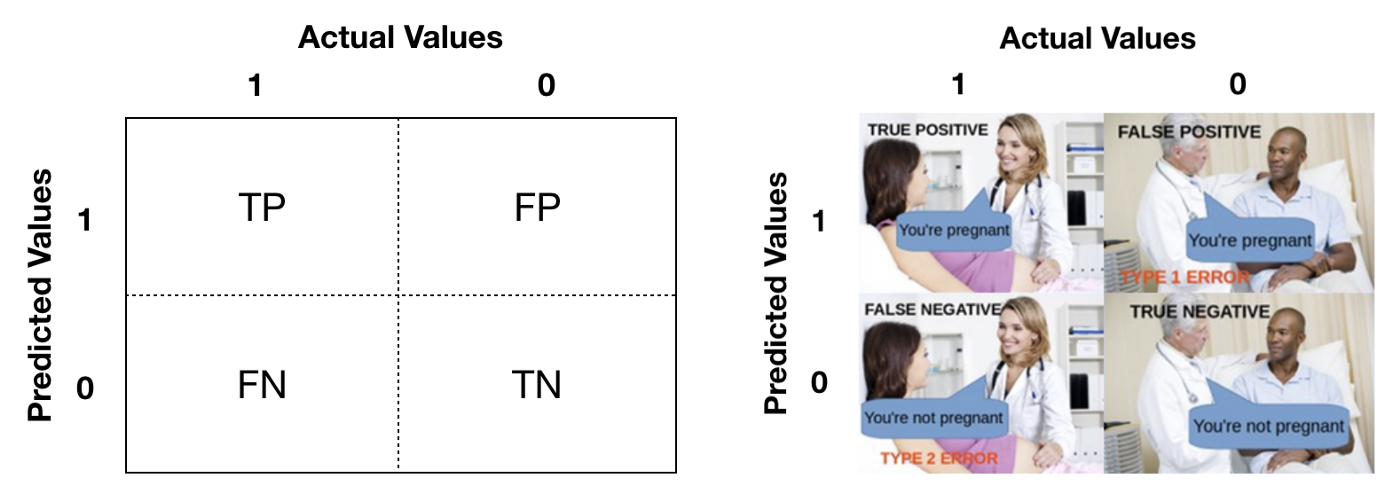
Swoistość = TN/(TN+FP)

Precyzja (PPV Positive Predictive Value) = TP/(TP+FP)

NPV (Negative Predictive Value) = TN/(TN+FN)

Dokładność = TP+TN/Suma

**sklearn.metrics.confusion\_matrix(*y\_true*, *y\_pred*, *\**, *labels=None*, *sample\_weight=None*, *normalize=None*)**

**Miary dokładności uczenia dla problemów klasyfikacyjnych cd.**

* Dokładność (ang. accuracy) Err=

Acc= 1- Err

* Dokładność zbalansowana (ang. Balanced accuracy) BErr=

BAcc= 1- BErr

* Czułość lub wrażliwość (ang. sensitivity, recall) Se=
* Znamienność (ang. specificity) Sp=
* Precyzja (ang. precision) P=
* Miara (ang. measure która dla β=1 przyjmuje postać

* Dokładność Acc=
* Dokładność zbalansowana BAcc=

**Miary dokładności uczenia dla problemów regresyjnych**

* Błąd średniokwadratowy (MSE)

e=

* (RMSE) e=
* Błąd średni (ME) e=
* Znormalizowany błąd średniokwadratowy (NMSE) e=

**Partia danych (Batch)**

Wielkość partii jest hiperparametrem określającym liczbę próbek, przez które należy przejść przed aktualizacją wewnętrznych parametrów modelu.

Partię należy traktować jako pętlę for iterującą po jednej lub kilku próbkach i dokonującą przewidywań. Na końcu serii przewidywania są porównywane z oczekiwanymi zmiennymi wyjściowymi i obliczany jest błąd. Na podstawie tego błędu algorytm aktualizacji jest używany do ulepszania modelu, np. przesuwania się w dół wzdłuż gradientu błędu.

Zbiór danych treningowych można podzielić na jedną lub więcej partii.

Gdy wszystkie próbki treningowe są wykorzystywane do utworzenia jednej partii, algorytm uczenia nazywa się zstępowaniem gradientowym partii. Gdy wielkość partii jest równa wielkości jednej próbki, algorytm uczenia nazywa się stochastycznym zstępowaniem gradientowym. Gdy wielkość partii jest większa niż jedna próbka i mniejsza niż wielkość zbioru danych treningowych, algorytm uczenia nazywa się zstępowaniem gradientowym przez minizbiorowisko.

Zstępowanie gradientowe dla partii. Wielkość partii = wielkość zbioru szkoleniowego

Stochastyczne zstępowanie gradientowe. Wielkość partii = 1

Zstępowanie gradientowe miniwzrostowe. 1 < wielkość partii < wielkość zbioru treningowego

W przypadku zstępowania gradientowego typu mini-batch popularne wielkości partii to 32, 64 i 128 próbek. Wartości te można spotkać w modelach w literaturze i w samouczkach.

Co zrobić, jeśli zbiór danych nie dzieli się równomiernie przez rozmiar partii?

Może się to zdarzyć i zdarza się często podczas trenowania modelu. Oznacza to po prostu, że ostatnia partia ma mniej próbek niż pozostałe partie.

Alternatywnie można usunąć niektóre próbki ze zbioru danych lub zmienić wielkość partii tak, aby liczba próbek w zbiorze danych dzieliła się równomiernie przez wielkość partii.

Łagodne wprowadzenie do zstępowania gradientowego w mini partiach i jak skonfigurować wielkość partii

Jak kontrolować szybkość i stabilność szkolenia sieci neuronowych Wielkość partii

Partia polega na aktualizacji modelu za pomocą próbek; następnie przyjrzyjmy się epoce.

**Epoka**

Liczba epok to hiperparametr określający, ile razy algorytm uczenia będzie pracował na całym zbiorze danych treningowych.

Jedna epoka oznacza, że każda próbka w zbiorze danych treningowych miała możliwość uaktualnienia wewnętrznych parametrów modelu. Epoka składa się z jednej lub kilku serii. Na przykład, jak powyżej, epoka składająca się z jednej serii jest nazywana algorytmem uczenia zstępującego gradientu.

Można pomyśleć o pętli for nad liczbą epok, w której każda pętla przechodzi przez zbiór danych treningowych. Wewnątrz tej pętli for znajduje się kolejna zagnieżdżona pętla for, która iteruje nad każdą partią próbek, przy czym jedna partia zawiera określoną liczbę próbek o "wielkości partii".

Liczba epok jest tradycyjnie duża, często wynosi setki lub tysiące, co pozwala algorytmowi uczącemu działać tak długo, aż błąd modelu zostanie wystarczająco zminimalizowany. W literaturze i w tutorialach można znaleźć przykłady liczby epok ustawionej na 10, 100, 500, 1000 i więcej.

Powszechne jest tworzenie wykresów liniowych, które przedstawiają epoki wzdłuż osi x jako czas, a błąd lub umiejętności modelu na osi y. Wykresy te są czasem nazywane krzywymi uczenia. Wykresy te są czasem nazywane krzywymi uczenia. Wykresy te mogą pomóc w ustaleniu, czy model nie nauczył się zbyt wiele, nie nauczył się wystarczająco dużo lub czy jest odpowiednio dopasowany do zbioru danych treningowych.

**Różnica między partią a epoką (Batch vs Epoch)**

Rozmiar partii to liczba próbek przetwarzanych przed aktualizacją modelu.

Liczba epok to liczba pełnych przejść przez zbiór danych treningowych.

Rozmiar partii musi być większy lub równy jeden i mniejszy lub równy liczbie próbek w zbiorze danych treningowych.

Liczbę epok można określić jako liczbę całkowitą z przedziału od jednego do nieskończoności. Można uruchamiać algorytm tak długo, jak się chce, a nawet zatrzymywać go, korzystając z innych kryteriów poza ustaloną liczbą epok, takich jak zmiana (lub brak zmiany) błędu modelu w czasie.

Obie wartości są całkowite i obie są hiperparametrami algorytmu uczącego, tzn. parametrami procesu uczenia, a nie wewnętrznymi parametrami modelu znalezionymi przez proces uczenia.

W przypadku algorytmu uczenia należy określić wielkość partii i liczbę epok.

Nie ma magicznych reguł, jak skonfigurować te parametry. Należy wypróbować różne wartości i sprawdzić, które z nich najlepiej sprawdzają się w danym problemie.

Załóżmy, że masz zbiór danych zawierający 200 próbek (wierszy danych) i wybrałeś wielkość serii 5 oraz 1000 epok.

Oznacza to, że zbiór danych zostanie podzielony na 40 partii, z których każda będzie miała pięć próbek. Wagi modelu będą aktualizowane po każdej serii pięciu próbek.

Oznacza to również, że jedna epoka będzie obejmowała 40 partii lub 40 aktualizacji modelu.

W przypadku 1000 epok model zostanie poddany działaniu lub przejdzie przez cały zbiór danych 1000 razy. Oznacza to łącznie 40 000 partii podczas całego procesu szkolenia.

**IPython** (Interactive Python) to [powłoka poleceń](https://en.wikipedia.org/wiki/Shell_(computing)) do obliczeń interaktywnych w wielu językach programowania. Pierwotnie opracowana dla [języka programowania Python](https://en.wikipedia.org/wiki/Python_(programming_language)) , która oferuje [introspekcję](https://en.wikipedia.org/wiki/Introspection_(computer_science)) , [multimedia](https://en.wikipedia.org/wiki/Rich_media) , składnię powłoki, [uzupełnianie tabulatorami](https://en.wikipedia.org/wiki/Tab_completion) i historię. IPython zapewnia następujące funkcje:

* Powłoki interaktywne (na bazie terminala i [Qt](https://en.wikipedia.org/wiki/Qt_(framework)" \o "Qt (framework)) ).
* Oparty na przeglądarce [interfejs notebooka](https://en.wikipedia.org/wiki/Notebook_interface) z obsługą kodu, tekstu, wyrażeń matematycznych, wykresów wbudowanych i innych mediów.
* Wsparcie dla interaktywnej wizualizacji danych i korzystania z zestawów narzędzi GUI.
* Elastyczne, wbudowalne tłumacze do załadowania do własnych projektów.
* Narzędzia do [obliczeń równoległych](https://en.wikipedia.org/wiki/Parallel_computing).
* sklearn.metrics.**r2\_score**( *y\_true* , *y\_pred* , *\** , *sample\_weight = brak* , *multioutput = 'uniform\_average'* )[[źródło]](https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/metrics/_regression.py#L587)

R2 (współczynnik determinacji) funkcja oceny regresji.

* Najlepszy możliwy wynik to 1,0 i może być ujemny (ponieważ model może być arbitralnie gorszy). Stały model, który zawsze przewiduje oczekiwaną wartość y, pomijając cechy wejściowe, otrzyma wartośćR2 wynik 0,0.
* sklearn.metrics.**mean\_absolute\_error**( *y\_true* , *y\_pred* , *\** , *sample\_weight = brak* , *multioutput = 'uniform\_average'* )[[źródło]](https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/metrics/_regression.py#L125)[¶](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html#sklearn.metrics.mean_absolute_error)
* Średnia strata wynikająca z regresji błędu bezwzględnego.

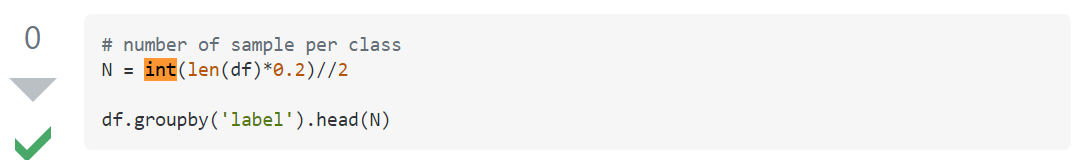
Czyli jeśli za pomocą jakiegoś modelu otrzymamy prognozę równą „44” a po obliczeniu MAE dla okresu testowego wyniesie ono Mae = 2,456, wtedy nasza prognoza = 44 może odbiegać od rzeczywistej wartości przyszłej o 2,456 czyli może być z przedziału od 41,544 do 46,456.

* sklearn.metrics.**mean\_squared\_error**( *y\_true* , *y\_pred* , *\** , *sample\_weight = None* , *multioutput = 'uniform\_average'* , *squared = True* )[[źródło]](https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/metrics/_regression.py#L274)[¶](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error)
* Utrata regresji błędu średniokwadratowego.

MSE jest miarą jakości estymatora - jest zawsze nieujemna, a wartości bliższe zeru są lepsze.

ratio = 0.6

N = int(len(data)\*ratio)



Liczba próbek na klasę

**DataFrame.rolling( *window* , *min\_periods = None* , *center = False* , *win\_type = None* , *on = None* , *axis = 0* , *closed = None* )**

Zapewnij obliczenia w oknie kroczącym.

**Parametry**

**window *int, offset lub podklasa BaseIndexer***

Rozmiar ruchomego okna. Jest to liczba obserwacji użytych do obliczenia statystyki. Każde okno będzie miało stały rozmiar.

Jeśli jest to przesunięcie, będzie to okres czasu każdego okna. Każde okno będzie miało zmienną wielkość w oparciu o obserwacje zawarte w okresie. Dotyczy to tylko indeksów podobnych do danych.

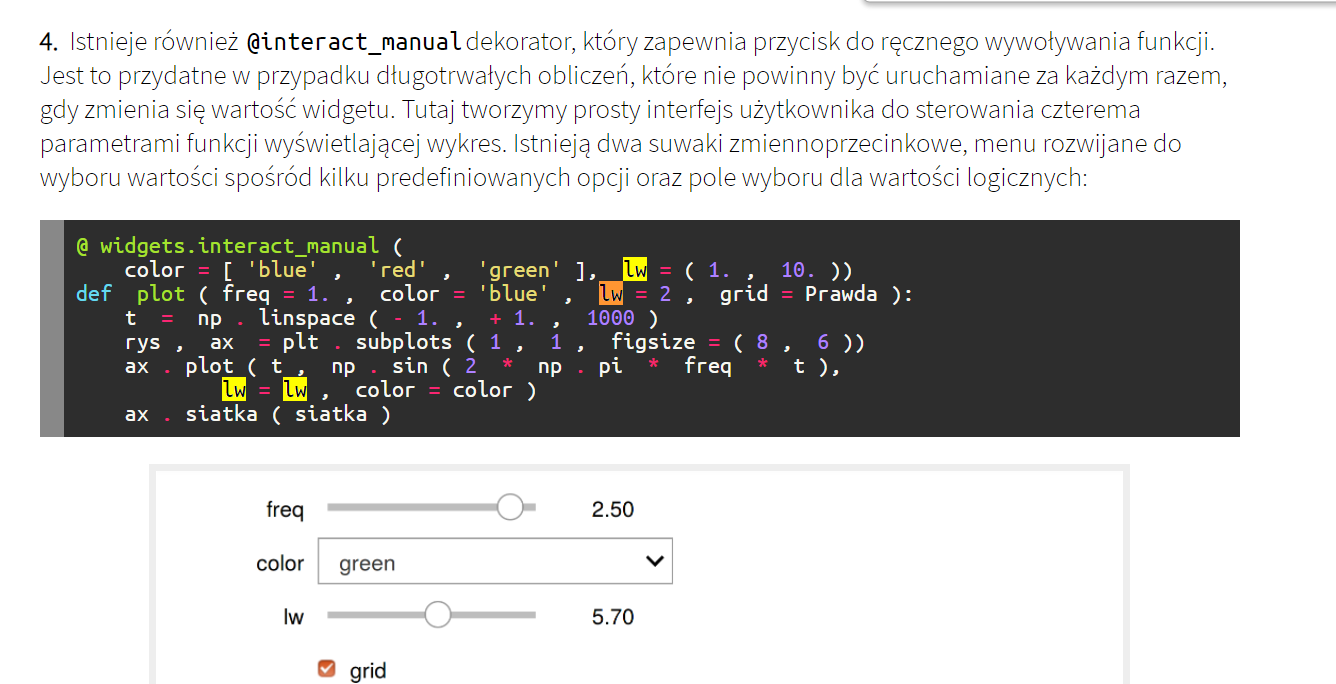
Jeśli przekazana jest podklasa BaseIndexer, oblicza granice okna na podstawie zdefiniowanej get\_window\_boundsmetody. Dodatkowe argumenty słów kluczowych, mianowicie *min\_periods* , *center* i *closed,* zostaną przekazane do *get\_window\_bounds* .

**center *bool, domyślnie False***

Umieść etykiety na środku okna.

Domyślnie wynik jest ustawiany na prawej krawędzi okna. Można to zmienić na środek okna, ustawiając center=True.

<https://ipython-books.github.io/33-mastering-widgets-in-the-jupyter-notebook/>



Co to jest Keras API?

**Keras** to **interfejs API** przeznaczony dla ludzi, a nie maszyn. **Keras** postępuje zgodnie   
z najlepszymi praktykami w zakresie zmniejszania obciążenia poznawczego: oferuje spójne   
i proste **interfejsy API** , minimalizuje liczbę działań użytkownika wymaganych w typowych przypadkach użycia oraz zapewnia jasne i przydatne komunikaty o błędach.

[Dokumentacja API Keras](https://keras.io/api) / [API](https://keras.io/api/callbacks) wywołań zwrotnych / EarlyStopping

### EarlyStopping klasa

Przerwij trenowanie, gdy monitorowana metryka przestanie się poprawiać.

Zakładając, że celem treningu jest zminimalizowanie straty. Dzięki temu monitorowana metryka byłaby 'loss', a tryb byłby 'min'. model.fit(). Pętla szkolenie będzie sprawdzić na końcu każdej epoki czy strata nie jest już maleje, biorąc pod uwagę min\_delta, a patience jeśli ma to zastosowanie. Gdy okaże się, że już się nie zmniejsza, model.stop\_training jest oznaczany jako Prawda i szkolenie się kończy.

Ilość do monitorowania musi być podana w logs dykt. Aby tak się stało, podaj stratę lub wskaźniki na model.compile().

[**sklearn.preprocessing**](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing).MinMaxScaler

*class*sklearn.preprocessing.**MinMaxScaler**( *feature\_range = 0, 1* , *\** , *copy = True* , *clip = False* )[[źródło]](https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/preprocessing/_data.py#L220)

Przekształć cechy, skalując każdą cechę do określonego zakresu.

Estymator ten skaluje i tłumaczy każdą cechę indywidualnie tak, że znajduje się ona w zadanym zakresie na zbiorze uczącym, np. Między zerem a jedynką.

#### **Podział na grupę uczącą, walidacyną i testową**

**Grupa ucząca** – to taki zestaw danych, który używamy do nauki algorytmu. Na podstawie tych danych model uczy się odpowiednio klasyfikować, buduje wszelkie zależności. Można powiedzieć, że przewiduje możliwe wyniki i podejmuje decyzje na podstawie przekazanych mu danych.

**Grupa walidacyjna** jest to taki zbiór danych, którego używamy do przeprowadzenia nieobciążonego testu modelu, który przeszkoliliśmy na danych treningowych (uczących). Test ten przeprowadzamy podczas wyboru modelu albo dobierając zestaw hiperparametrów. Ważnym jest, żeby dane zawarte w zbiorze walidacyjnym nie były używane wcześniej do nauki modelu, ponieważ nie będą wtedy nadawać się do obiektywnego, nieobciążonego testowania.

**Grupa testowa**– kiedy już wybraliśmy model, wybraliśmy hiperparametry, to nadchodzi czas na przetestowanie wszystkiego danymi z grupy testowej. Jest bardzo ważne, by dane te nie były wcześniej używane do uczenia czy walidacji modelu, ponieważ chcemy wiedzieć, jak wybrany algorytm sprawdza się na danych, z którymi nigdy wcześniej nie miał do czynienia.

Zestaw testowy powinien spełniać tak zwany „złoty standard”, czyli być idealnie przygotowanym, bez jakichkolwiek błędów. Jest używany tylko raz, na samym końcu, kiedy chcemy sprawdzić, jak zadziałały nasze powyżej wspomniane kroki – polegające na uczeniu i walidacji.

**numpy.ndarray**

***class*numpy.ndarray( *shape* , *dtype = float* , *buffer = None* , *offset = 0* , *strides = None* , *order = None*)**

Obiekt tablicy reprezentuje wielowymiarową, jednorodną tablicę elementów o stałym rozmiarze. Powiązany obiekt typu danych opisuje format każdego elementu w tablicy (jego kolejność bajtów, ile bajtów zajmuje w pamięci, czy jest to liczba całkowita, liczba zmiennoprzecinkowa, czy coś innego itp.)

Tablice powinny być konstruowane z użyciem **[array](https://numpy.org/doc/stable/reference/generated/numpy.array.html" \l "numpy.array" \o "numpy.array)**, **[zeros](https://numpy.org/doc/stable/reference/generated/numpy.zeros.html" \l "numpy.zeros" \o "numpy.zeros)**lub **[empty](https://numpy.org/doc/stable/reference/generated/numpy.empty.html" \l "numpy.empty" \o "numpy.empty)**(patrz Patrz również rozdział poniżej). Podane tutaj parametry odnoszą się do niskopoziomowej metody ( *ndarray (…)* ) *służącej* do tworzenia instancji tablicy.

Jakie jest znaczenie przekształcenia (- 1 1)?

Na przykład, jeśli masz tablicę kształtu (2,4), a następnie **przekształcasz** ją za pomocą **(-1** , **1** ), to tablica zostanie **przekształcona** w taki sposób, że wynikowa tablica ma tylko **1** kolumnę i jest to możliwe tylko poprzez 8 rzędów, a więc (8, **1** ).

### [Co oznacza zmiana kształtu (-1,1) - Uczenie maszynowe - Kodowanie ...](Co oznacza zmiana kształtu (-1,1) - Uczenie maszynowe - Kodowanie ...https://discuss.codingblocks.com ›what-is-mean-of-res ...)

[https://discuss.codingblocks.com ›what-is-mean-of-res ...](Co oznacza zmiana kształtu (-1,1) - Uczenie maszynowe - Kodowanie ...https://discuss.codingblocks.com ›what-is-mean-of-res ...)

[Jakie jest znaczenie przekształcenia (- 1 1)?](https://www.google.com/search?sxsrf=ALeKk01rcFOJ1VUmVWp_elQCer4rdqknSA:1621112702106&q=What+is+the+meaning+of+reshape+(-+1+1%3F&sa=X&ved=2ahUKEwjyv_-1y8zwAhWxCRAIHRGUCEEQzmd6BAgDEAU)

Jaki jest pożytek z reshape (- 1 1) w Pythonie?

Tablica spłaszczania oznacza konwersję tablicy wielowymiarowej na tablicę 1D. W tym celu możemy **użyć reshape (-1** ).

[Zmiana kształtu tablicy NumPy - W3Schools](Zmiana kształtu tablicy NumPy - W3Schoolshttps://www.w3schools.com ›numpy_array_reshape)

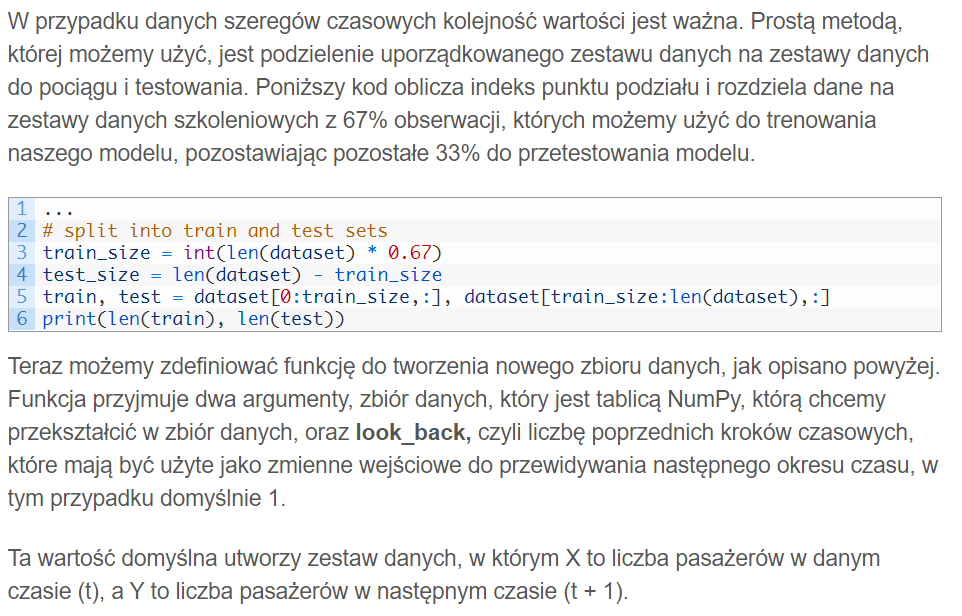
[https://www.w3schools.com ›numpy\_array\_reshape](Zmiana kształtu tablicy NumPy - W3Schoolshttps://www.w3schools.com ›numpy_array_reshape)

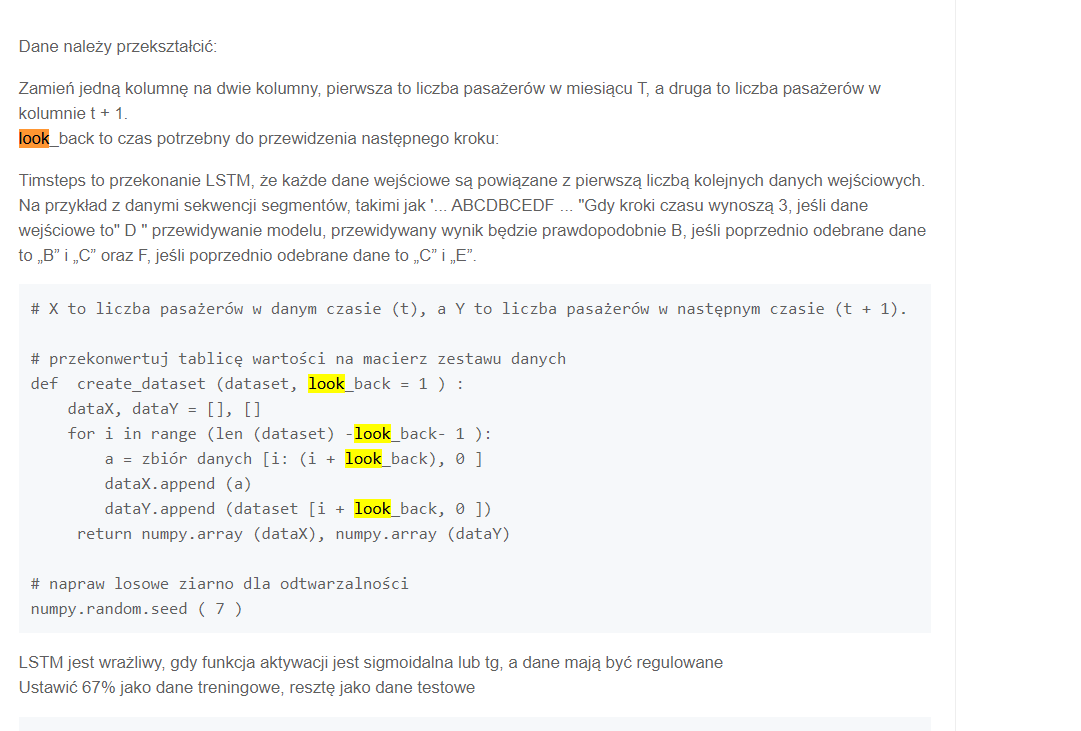
Importowanie i używanie *MinMaxScaler* działa - tak jak wszystkie następne skalery - w dokładnie taki sam sposób jak *StandardScaler* . Jedyna różnica polega na parametrach przy inicjowaniu nowej instancji.

Tutaj skalujemy *cechę 3* ( *f3* ) do skali od -3 do 3. Zgodnie z oczekiwaniami nasza wartość maksymalna ( *25* ) jest przekształcana do 3, a nasza wartość minimalna (- *1* ) jest przekształcana do -3. Wszystkie inne wartości są liniowo skalowane między tymi wartościami.

from sklearn.preprocessing import MinMaxScalerscaler = MinMaxScaler (feature\_range = (- 3,3))scaler.fit\_transform (X.f3.values.reshape (-1, 1))

Cecha 3 przed i po zastosowaniu MinMaxScaler





Jakie jest znaczenie model sequential ()? model = Sequential()

Z **definicji** dokumentacji Keras model sekwencyjny jest liniowym stosem warstw.Model sekwencyjny można utworzyć, przekazując konstruktorowi listę instancji warstw: z keras.models import Sequential from keras.layers import Dense, Activation model = Sequential ([Gęsty (32, kształt\_wejścia = (784,))…

Co to jest Adam Optimizer?

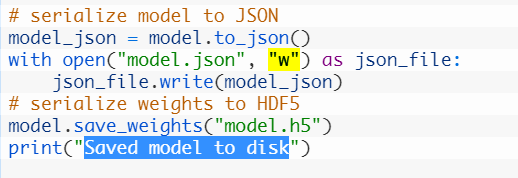
**Adam** jest zastępczym algorytmem **optymalizacji** dla stochastycznego zejścia w gradiencie do trenowania modeli uczenia głębokiego. **Adam** łączy najlepsze właściwości algorytmów AdaGrad i RMSProp, aby zapewnić algorytm **optymalizacji** , który radzi sobie z rzadkimi gradientami w przypadku hałaśliwych problemów.

W większości przypadków **Adam** jest **najlepszy** spośród adaptacyjnych **optymalizatorów** . **Dobre w** przypadku rzadkich danych: adaptacyjna szybkość uczenia się jest **idealna** dla tego typu zestawów danych.

Verbose (polski: gadatliwy) -> stąd poniższe tłumaczenia (chciałam, by mieli Państwo wszystko po polsku). 😊



Zapisz swój model sieci neuronowej w formacie JSON



JSON to prosty format pliku do hierarchicznego opisywania danych.

Keras zapewnia możliwość opisania dowolnego modelu przy użyciu formatu JSON z funkcją to\_json () . Można to zapisać do pliku, a następnie załadować za pomocą funkcji model\_from\_json () , która utworzy nowy model ze specyfikacji JSON.

Wagi są zapisywane bezpośrednio z modelu za pomocą funkcji save\_weights (), a później ładowane za pomocą symetrycznej funkcji load\_weights () .

Poniższy przykład szkoli i ocenia prosty model ze zbioru danych Indian Pima. Model jest następnie konwertowany do formatu JSON i zapisywany do model.json w katalogu lokalnym. Wagi sieci są zapisywane w pliku model.h5 w katalogu lokalnym.

Dane modelu i wagi są ładowane z zapisanych plików i tworzony jest nowy model. Ważne jest, aby skompilować załadowany model przed jego użyciem. Dzieje się tak, aby prognozy wykonane przy użyciu modelu mogły korzystać z odpowiednich wydajnych obliczeń z zaplecza Keras.

<https://blog.eduonix.com/software-development/building-logistic-regression-python/>

*n = len (X\_train [0])  
k = len (Y\_train [0])*

Tutaj „n” oznacza liczbę cech w punkcie danych, a „k” oznacza całkowitą liczbę klas (co jest równe 3 w przypadku zbioru danych Iris).

Jeśli używasz Matplotlib z poziomu skryptu, funkcja **plt** . **show** () jest Twoim przyjacielem. **plt** . **show** () uruchamia pętlę zdarzeń, wyszukuje wszystkie aktualnie aktywne obiekty figur i otwiera jedno lub więcej interaktywnych okien, które wyświetlają twoją figurę lub figury.

# matplotlib.pyplot.subplots\_adjust

matplotlib.pyplot.subplots\_adjust( *left = None* , *bottom = None* , *right = None* , *top = None* , *wspace = None* , *hspace = None* )

Dostosuj parametry układu wydruku pomocniczego.

Nieustawione parametry pozostają niezmienione; początkowe wartości są podane przez [rcParams["figure.subplot.[name]"]](https://matplotlib.org/stable/tutorials/introductory/customizing.html?highlight=figure.subplot.%5bname%5d" \l "a-sample-matplotlibrc-file).

|  |  |
| --- | --- |
| **Parametry:** | **left**float, opcjonalny  Położenie lewej krawędzi wykresów podrzędnych jako ułamek szerokości figury.  **right**float, opcjonalny Położenie prawej krawędzi wykresów podrzędnych jako ułamek szerokości figury.  bottom:  Położenie dolnej krawędzi wykresów podrzędnych jako ułamek wysokości figury.  top:  Położenie górnej krawędzi wykresów podrzędnych jako ułamek wysokości figury.  wspace:  Szerokość wypełnienia między wykresami podrzędnymi jako ułamek średniej  szerokości osi.  hspace:  Wysokość wypełnienia między polami pomocniczymi jako ułamek średniej  wysokości osi. |

# seaborn.despine

**seaborn.despine( *fig = brak* , *ax = brak* , *top = True* , *right = True* , *left = false* , *bottom = False* , *offset = None* , *trim = False* )**

Usuń górne i prawe grzbiety z działek.

**top, right, left, bottom *boolean, opcjonalnie***

Jeśli to prawda, usuń ten „kręgosłup” (spine).

# matplotlib.pyplot.tight\_layout

matplotlib.pyplot.tight\_layout( *\** , *pad = 1.08* , *h\_pad = brak* , *w\_pad = brak* , *rect = brak* )

Dostosuj dopełnienie między polami pomocniczymi i wokół nich.

Przeglądając powyższy obraz, możemy domyślnie obserwować znaczniki zaznaczeń Matplotlib na osi x i y.

**Przypadek 1.1: Gdy chcemy usunąć tiki na pojedynczej osi (tutaj oś y):**

Aby usunąć zaznaczenia na osi y, metoda tick\_params () ma atrybut o nazwie **left** i możemy ustawić jego wartość na **False** i przekazać go jako parametr wewnątrz funkcji tick\_params (). Usuwa zaznaczenie na osi Y.

* Pyton

<https://www.geeksforgeeks.org/how-to-remove-ticks-from-matplotlib-plots/>

**Przypadek 3: Gdy chcemy usunąć zarówno znaczniki, jak i etykiety z obu osi**

Aby jednocześnie usunąć znaczniki i etykiety z obu osi, oprócz ustawienia atrybutu **lewego** i **prawego** na **Fałsz** , użyjemy również dwóch dodatkowych atrybutów dla etykiet, którymi są: **labelleft** i **labelbottom,** i ustawimy jego wartość na **False** i przekażemy ją do środka funkcja tick\_params () . Spowoduje to również usunięcie etykiet z obu osi.