Currency Model LSTM

Aby zainstalować Tensorflow, potrzebujemy pip dla Pythona. Zwykle pip jest już zainstalowany, jeśli wersja Pythona to 3 lub wyższa.

1. Tworzenie wirtualnego środowiska Tensorflow: pip install --user virtualenv
2. Po skonfigurowaniu środowiska wirtualnego użyj następującego polecenia, aby zainstalować pakiet Tensorflow pip: pip install --upgrade tensorflow

Gdybyśmy użyli na początku pip install tensorflow, to wyskoczyłby nam błąd. Dlaczego ?

Ponieważ u nas Python nie ma dostępu do plików serwera. Dlatego trzeba to zrobić z uwagi na konkretnego, danego użytkownika. Dopisujemy zatem –user, by zainstalować biblioteki tylko dla danego użytkownika.

Istnieje wiele metod, których można użyć do zainstalowania TensorFlow, takich jak użycie pip do zainstalowania kół dostępnych w PyPI. Instalacja TensorFlow przy użyciu pakietów conda oferuje szereg korzyści, w tym kompletny system zarządzania pakietami, szerszą obsługę platform, bardziej usprawnione działanie GPU i lepszą wydajność procesora. Pakiety te są dostępne za pośrednictwem repozytorium Anaconda, a ich instalacja jest tak prosta, jak uruchomienie „conda install tensorflow” lub „conda install tensorflow-gpu” z interfejsu wiersza poleceń.

Jedną z kluczowych korzyści z instalacji TensorFlow przy użyciu conda zamiast pip jest wynik systemu zarządzania pakietami conda. Gdy TensorFlow jest instalowany przy użyciu conda, conda instaluje również wszystkie niezbędne i kompatybilne zależności dla pakietów. Odbywa się to automatycznie; użytkownicy nie muszą instalować żadnego dodatkowego oprogramowania za pośrednictwem menedżerów pakietów systemowych lub w inny sposób.

KERAS Czy mogę używać keras bez TensorFlow?

Nie, Keras to wysokopoziomowe API do budowania i trenowania modeli sieci neuronowych. **Keras nie jest zależny od TensorFlow** i na odwrót. Keras może używać TensorFlow jako swojego zaplecza. Może również korzystać z innych zapleczy, takich jak Theano (Theano to biblioteka Pythona i kompilator optymalizujący do manipulowania i oceny wyrażeń matematycznych, zwłaszcza tych o wartościach macierzowych. W Theano obliczenia są wyrażane przy użyciu składni NumPy-esque i kompilowane w celu wydajnego działania na architekturze CPU ((jednostka centralna) to uogólniony procesor przeznaczony do wykonywania różnorodnych zadań), lub GPU ((jednostka przetwarzania grafiki) to wyspecjalizowana jednostka przetwarzania z rozszerzonymi możliwościami obliczeń matematycznych, idealna do zadań związanych z grafiką komputerową i uczeniem maszynowym)) i CNTK (Microsoft Cognitive Toolkit, wcześniej znany jako CNTK, a czasem stylizowany jako Microsoft Cognitive Toolkit, to przestarzała platforma do głębokiego uczenia się opracowana przez Microsoft Research. Microsoft Cognitive Toolkit opisuje sieci neuronowe jako serię kroków obliczeniowych za pomocą ukierunkowanego wykresu).

DENSE – GĘSTY- Gęstą warstwę można dodać do modelu sekwencyjnego **za pomocą metody „dodaj” i określając typ warstwy jako „Gęsta”** . Warstwa gęsta to **regularna, głęboko połączona warstwa sieci neuronowej** . Jest to najpowszechniejsza i najczęściej stosowana warstwa.

FLATTEN- SPŁASZCZYĆ- Spłaszcza dane wejściowe. Nie wpływa na wielkość partii. Dlaczego jej używamy? Musi mieć postać jednowymiarowego wektora liniowego. Kształty prostokątne lub sześcienne nie mogą być danymi wejściowymi bezpośrednimi. I dlatego potrzebujemy spłaszczania i w pełni połączonych warstw. Spłaszczanie to **przekształcanie danych w jednowymiarową tablicę w celu wprowadzenia ich do następnej warstwy** .

INPUT- służy do tworzenia instancji tensora Keras.

Tensor Keras to obiekt tensor z bazowego backendu (Theano lub TensorFlow), który uzupełniamy o pewne atrybuty, które pozwalają nam zbudować model Keras po prostu znając dane wejściowe i wyjściowe modelu.

BatchNormalization- NORMALIZACJA WSADOWA- Warstwa, która normalizuje swoje dane wejściowe. Normalizacja wsadowa to **technika uczenia bardzo głębokich sieci neuronowych, która standaryzuje dane wejściowe do warstwy dla każdej minipartii** . Wpływa to na stabilizację procesu uczenia się i radykalne zmniejszenie liczby epok treningowych wymaganych do trenowania głębokich sieci.

Dropout- Stosuje Dropout do wejścia. Warstwa Dropout **losowo ustawia jednostki wejściowe na 0 z częstotliwością na każdym kroku w czasie treningu, co pomaga zapobiegać nadmiernemu dopasowaniu.** Termin „dropout” jest używany w odniesieniu **do techniki, która usuwa niektóre węzły sieci** . Wypadanie może być postrzegane jako chwilowa dezaktywacja lub ignorowanie neuronów sieci. Technikę tę stosuje się w fazie treningu, aby zredukować efekty nadmiernego dopasowania.

LSTM- Long Short-Term Memory warstwa. Long-Short-Term Memory Network lub LSTM to **odmiana rekurencyjnej sieci neuronowej (RNN), która jest dość skuteczna w przewidywaniu długich sekwencji danych, takich jak zdania i ceny akcji w określonym czasie** . Różni się od normalnej sieci ze sprzężeniem zwrotnym, ponieważ w jej architekturze występuje pętla sprzężenia zwrotnego.

Sequential- Kiedy używać modelu sekwencyjnego?

**Model ma wiele wejść lub wiele wyjść** . Każda z twoich warstw ma wiele wejść lub wiele wyjść. Musisz udostępnić warstwy. Potrzebujesz nieliniowej topologii (np. połączenie resztkowe, model wielorozgałęziony)

load\_model- Wczytuje model zapisany przez model.save(). **Ładuje model zapisany przez model. zapisz()** . Należy zauważyć, że po załadowaniu wagi modelu mogą mieć różne nazwy w zakresie. Nazwy w zakresie obejmują nazwy modelu/warstwy, takie jak „dense\_1/kernel:0” .

Matplotlib to **kompleksowa biblioteka do tworzenia statycznych, animowanych i interaktywnych wizualizacji w Pythonie** . Matplotlib czyni łatwe rzeczy łatwymi i trudnymi możliwymi.

Seaborn- to biblioteka wizualizacji danych Pythona zbudowana na bazie Matplotlib. Część kodu dotycząca importu seaborn mówi Pythonowi, aby wprowadził bibliotekę Seaborn do bieżącego środowiska.

Pandas- Pandas to pakiet Pythona typu open source, który jest **najczęściej używany do zadań związanych z nauką o danych/analizą danych i uczeniem maszynowym** . Jest zbudowany na innym pakiecie o nazwie Numpy, który zapewnia obsługę tablic wielowymiarowych.

NumPy- to biblioteka Pythona używana do pracy z tablicami. Posiada również funkcje do pracy w dziedzinie algebry liniowej, transformaty Fouriera i macierzy. NumPy został stworzony w 2005 roku przez Travisa Oliphanta. Jest to projekt open source i możesz z niego swobodnie korzystać. NumPy to skrót od **Numerical Python**.

#data.shape- wymiary tablicy z danymi, ilość wierszy i kolumn

**Różne sposoby na znalezienie kolumn, które zawierają Nan w DataFrame**

1. df.isna().any() -> Użyj isna() do znalezienia wszystkich kolumn z Nan
2. df.isnull().any()
3. df[df.columns[df.isna().any()]] -> Użyj isna() , aby wybrać wszystkie kolumny z wartościami NaN
4. df[df.columns[df.isnull().any()]]

1. jeżeli kolumny posiadają tylko wartości puste i 1 to uzupełniamy puste wartości o 0 fillna ()

df = fill\_0(df)

1. uzupelnienie o wartość średnią PROWIZJA\_mean = df['Admin'].mean()

<https://ichi.pro/pl/jak-radzic-sobie-z-brakujacymi-danymi-za-pomoca-pand-155823730532226>

data.columns[data.isna().any()] .tolist() -> tolist() -> używany do konwersji elementów danych tablicy na listę, w naszym przypadku plik wygląda przejrzyściej (w liście a nie w tabeli).

#len() jest jedną z wbudowanych funkcji Pythona. Zwraca długość obiektu. Na przykład może zwrócić liczbę pozycji na liście.

#isnull() Wykryj brakujące wartości dla obiektu podobnego do tablicy.

#sum() zwraca liczbę, sumę wszystkich elementów w iteracji.

30% u nas umownie. Jeżeli chcemy dokładniejsze usuwanie kolumn, to możemy użyć 10%. Myślę, że 30% pozwoli nam usunąć te najbardziej „szkodliwe” kolumny. Gdy kolumna ma powyżej 30% NaN to zepsuje nam wyniki.

#info() ; entries- wpisy

#drop() Usuń określone etykiety z wierszy lub kolumn. Usuń wiersze lub kolumny, określając nazwy etykiet i odpowiednią oś lub określając bezpośrednio nazwy indeksów lub kolumn.

data = data.drop('Wolumen\_ftse250\_d', axis=1)

**DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')**

Parameters: labels: single label or list-like

Index or column labels to drop. A tuple will be used as a single label and not treated as a list-like.

**-axis**{0 or ‘index’, 1 or ‘columns’}, default 0

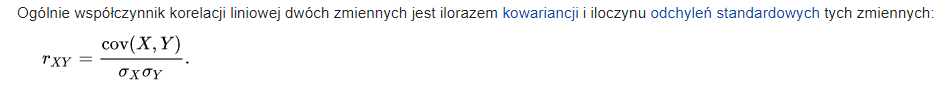
Whether to drop labels from the index (0 or ‘index’) or columns (1 or ‘columns’).

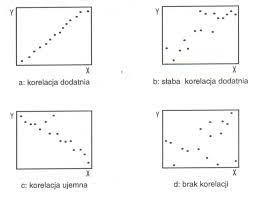
#shift(-5) Aby łatwo przesunąć na przykład o 5 wartości; Przesuń indeks o żądaną liczbę okresów z opcjonalną częstotliwością czasu. SESJE OTWIERAJĄ SIĘ CODZIENNIE, WIĘC CODZIENNIE DANA AKCJA MA INNY KURS. W KOLUMNIE ZAKOŃCZENIA NA DANY DZIEŃ- CHCEMY JĄ PRZESUNĄĆ O 1. Przesuwając zamknięcie o 1 dzień, otwarcie będzie prognozować nam o 1 dzień do przodu- nauka co się może zdarzyć następnego dnia.

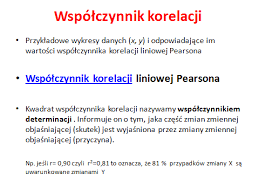
#head() Zwraca pierwsze n wierszy. n domyślnie = 5

Float- zmiennoprzecinkowe wartości rzeczywiste

Int- typ numeryczny







1. Głównym celem budowy modeli predykcyjnych jest jego późniejsze wykorzystanie. W związku z tym należy zbadać pomiar jego dokładności, np. za pomocą MSE (ang. Mean Square Error). W modelach do klasyfikacji do badania jakości modelu wykorzystuje się np. krzywą ROC (zależność czułości od swoistości) -> AUC (area under the curve), macierz błędów/pomyłek.

Uczenie modelu i wnioskowanie jedynie na podstawie wyniku na zbiorze danych, na którym uczymy jest krótkowzroczne. Oczywiście, model uruchomiony na dokładnie tych danych będzie działać wzorowo; jednakże szybko okaże się, że nie ma zastosowania w praktyce, gdyż jego zdolność generalizacji będzie mała. Należy przyjąć strategię jak testować postęp nauki, żeby wyciągać właściwe wnioski na temat jakości modelu ([źródło](https://aivision.pl/train-test-val/)).

1. Grupowanie danych na zbiory:

Dane dzielimy na zbiory, aby być pewnym, że nie uczy się na pamięć, tylko wyciąga z danych ogólną wiedzę.

- uczącą (70-80 %) - 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. Musi być to zbiór duży i z danymi różnorodnymi, aby model mógł generalizować zebraną wiedzę.

W skrócie zbiór uczący służy do nauki modelu.

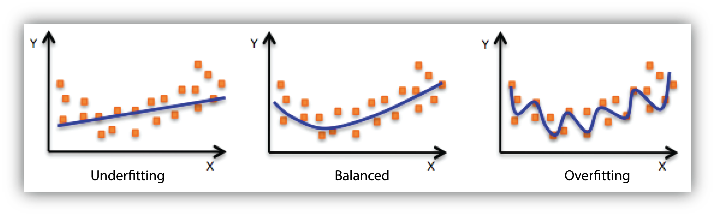
- walidacyjną (10%) - jest to taki zbiór danych, którego używamy do przeprowadzenia nieobciążonego testu modelu, który przeszkoliliśmy na danych treningowych (uczących). Test ten przeprowadzamy podczas wyboru modelu albo dobierając zestaw hiperparametrów.

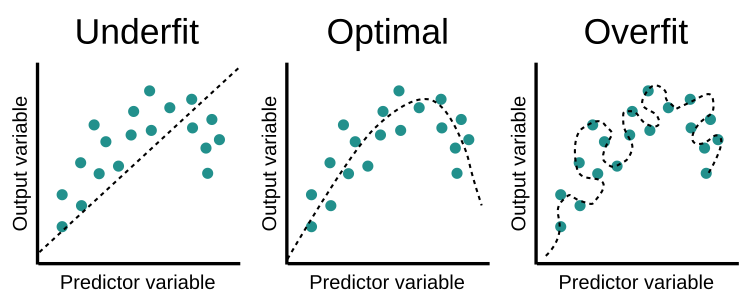
Test nieobciążony – test, którego moc (1-β) przewyższa poziom istotności. Ważnym jest, żeby dane zawarte w zbiorze walidacyjnym nie były używane wcześniej do nauki modelu, ponieważ nie będą wtedy nadawać się do obiektywnego, nieobciążonego testowania.

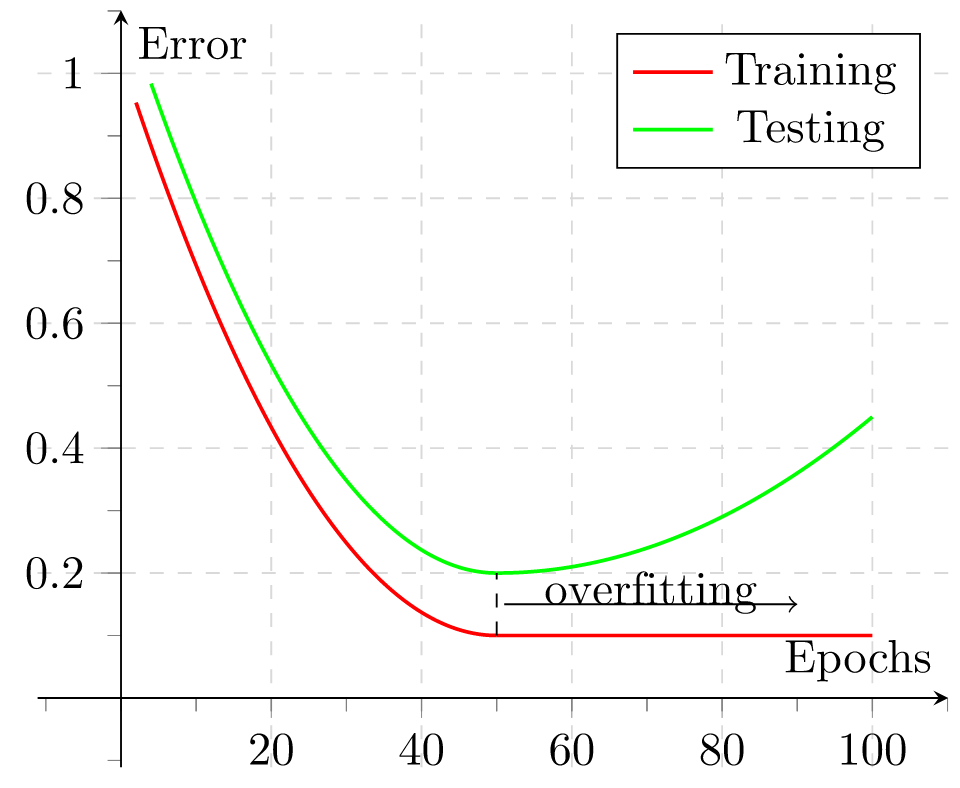
Zbiór walidacyjny nazywany jest też jako „development set” lub „dev set”. Podkreśla to, że zestaw danych walidacyjnych służy do testowania wyników podczas doboru modelu czy hiperparametrów na etapie rozwoju wybranego algorytmu.

Zbiór walidacyjny stosujemy, aby uniknąć przeuczenia ze zbioru uczącego.

- testową (20-30 %) – bardzo ważne -> powinien spełniać złoty standard: Dane idealnie przygotowane używamy tylko raz. Zbiór testowy dobieramy po to, aby uniknąć przeuczenia.





Wektory części testowej zawierają informację o faktycznym wyniku, a nauczony model dostarcza przewidywania.

Zdarzają się modele, które mają bardzo mało hiperparametrów do optymalizacji. Zdarzają się sytuacje, kiedy z góry wiemy, jakiego modelu będziemy używać. Wtedy można sobie pozwolić na wykorzystanie tylko dwóch zestawów danych – uczącego i testującego. Przy czym należy zrobić to świadomie, mając na uwadze wszelkie możliwe konsekwencje wyboru.

Najprostszym możliwym podejściem jest prosty, losowy podział zbioru na dwie części z zachowaniem jakiejś proporcji pomiędzy nimi. Służy do tego metoda **sklearn.model\_selection.train\_test\_split**.

Cross walidacja – metoda statystyczna polegająca na podziale próby statystycznej na podzbiory, a następnie przeprowadzaniu wszelkich analiz na niektórych z nich, tzw. zbiór uczący, podczas gdy pozostałe służą do potwierdzenia wiarygodności jej wyników, tzw. zbiór testowy (branż. zbiór walidacyjny).

Sprawdzian prosty

Jest to najbardziej typowy rodzaj sprawdzianu, w którym próbę dzieli się losowo na rozłączne zbiory: uczący i testowy. Zwykle zbiór testowy stanowi mniej niż 1/3 próby. Niektórzy nie zaliczają tego typu sprawdzenia do metody sprawdzianu krzyżowego.

Sprawdzian k-krotny

W tej metodzie oryginalna próba jest dzielona na k podzbiorów. Następnie kolejno każdy z nich bierze się jako zbiór testowy, a pozostałe razem jako zbiór uczący, po czym wykonuje się analizę (analiza jest więc wykonywana k razy). Uzyskane k rezultaty łączy się (np. uśrednia) w celu uzyskania jednego wyniku. k-krotna walidacja k-podzbiorów. Każdy z nich działa jako zbiór walidacyjny. K-1 to zbiory uczące. Średnia ze wszystkich.

K-krotna walidacja krzyżowa poprawia model poprzez walidację danych. Technika ta zapewnia, że wynik modelu nie jest związany z techniką, której używamy do wyboru testowego lub treningowego zbioru danych. Metoda K-krotnej walidacji krzyżowej dzieli zbiór danych na podzbiory o liczbie K. W związku z tym powtarza ona metodę holdout k razy.

Zestaw danych dzielimy na trzy grupy, gdy mamy wiele parametrów oraz hiperparametrów.

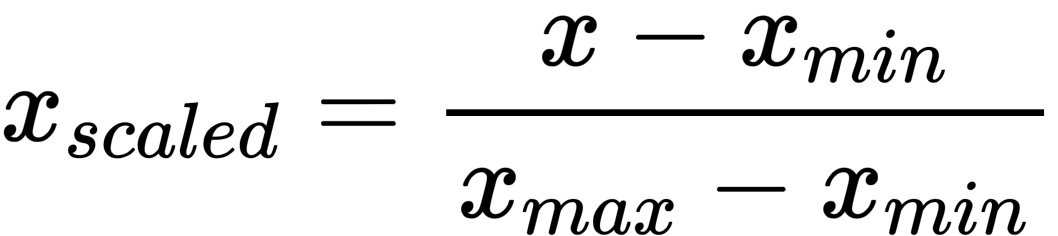
Jeśli parametr wyliczany jest samodzielnie przez algorytm podczas uczenia nazywamy go po prostu parametrem. Przykładem mogą być wagi w sieciach neuronowych.

Hiperparametry – są podawane przez użytkownika, który buduje model, pisze algorytm. To one kontrolują proces uczenia. Przykładami hiperparametrów są: ilość warstw, SVM, liczba drzew i iteracji w lesie decyzyjnym.

Kod:

MinMaxScaler - służy do normalizacji danych

Główna idea normalizacji/standaryzacji jest zawsze taka sama. Zmienne, które są mierzone w różnych skalach, nie wnoszą równego wkładu do funkcji dopasowania i uczenia modelu i mogą powodować zakłócenia. Dlatego, aby rozwiązać ten potencjalny problem, przed dopasowaniem modelu zwykle stosuje się normalizację według cech, taką jak skalowanie MinMax.



MinMaxScaler**(**feature\_range**=(0,** **1),** **\*,** copy**=True,** clip**=False)**

**Docstring:**

Transform features by scaling each feature to a given range.

This estimator scales and translates each feature individually such

that it is in the given range on the training set, e.g., between

zero and one.

The transformation is given by:

X\_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X\_scaled = X\_std \* (max - min) + min

where min, max = feature\_range.

This transformation is often used as an alternative to zero mean,

unit variance scaling.

Init – wpisujemy za każdym razem przy tworzeniu nowej klasy

Konwencje Python:

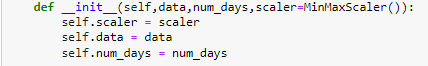
\_\_init\_\_() to specjalna funkcja która wywoływana jest przy tworzeniu instancji klasy - można sobie w niej stworzyć potrzebne w programie zmienne, dodać jakiś kod który ma być wykonany za każdym razem gdy tworzymy nowy obiekt klasy. To metoda zarezerwowana. Jeśli chcesz przykładowo zdefiniować sobie na starcie jakieś zmienne z których będą korzystać metody tej klasy to okazuje się dość przydatne.

self - to taka "zmienna globalna" która (gdy zostanie zdefiniowana) staje się "widoczna" w każdej metodzie klasy (głównie za sprawą tego że jest pierwszym argumentem każdej funkcji/metody w ramach klasy).

Klasa – przepis na obiekt 😊

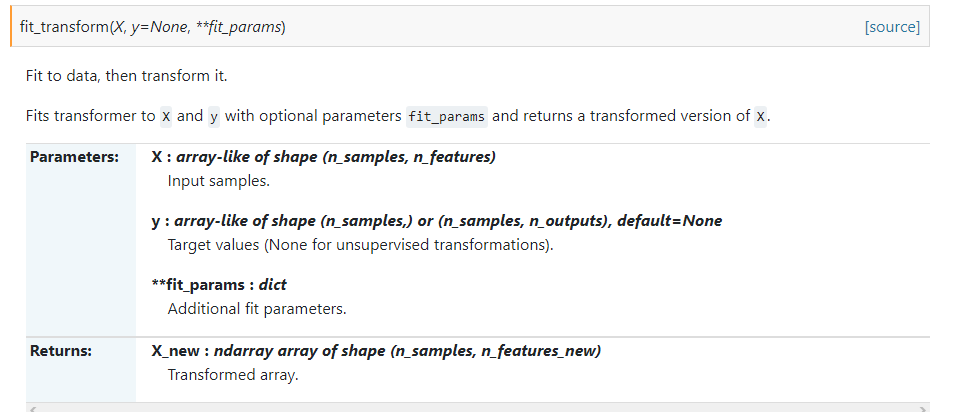
Metoda – służy do konstruowania instancji (egzemplarza) klasy

Jak wygląda przekazywanie argumentów i definiowanie parametrów? Identycznie jak w przypadku metod w klasie – wystarczy dopisać parametry w nawiasie 😉 W poniższym przykładzie dodajemy parametry data, num\_days, scaler a następnie przypisujemy go do zmiennej z prefiksem self co sprawi, że wszystkie metody w klasie będą miały do niej dostęp. Skorzystamy z tego i dodamy tę zmienną do tekstu wypisywanego na konsolę:





fit\_transform – to połączenie fit i transform (stosujemy dla zbioru uczącego jak i testującego)



Metoda ta wykonuje dopasowanie i przekształcenie na danych wejściowych w jednym czasie i przekształca punkty danych. Jeśli użyjemy metod fit i transform oddzielnie, gdy potrzebujemy obu, zmniejszy to wydajność modelu, więc użyjemy fit\_transform(), która wykona obie czynności.

Paczki danych zbieramy losowo.

Załóżmy, że utworzymy obiekt StandarScaler, a następnie wykonamy funkcję .fit\_transform(), która obliczy średnią(μ) i odchylenie standardowe(σ) cechy F, a następnie przekształci punkty danych cechy F (pochodna x’).

Self Y – doprowadzi do tego, że wartość maksymalna będzie mieć wartość 1.

randint(low, high=None, size=None, dtype=int)

Return random integers from `low` (inclusive) to `high` (exclusive).

Return random integers from the "discrete uniform" distribution of

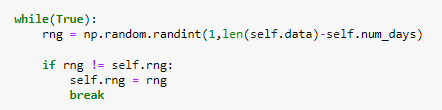
the specified dtype in the "half-open" interval [`low`, `high`). If

`high` is None (the default), then results are from [0, `low`).

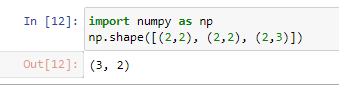
self.x\_batch

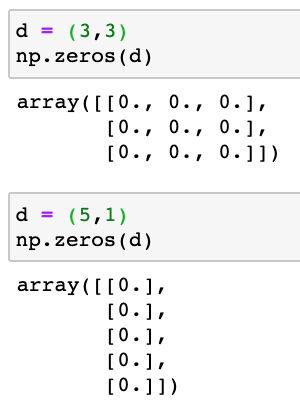
self.y\_batch

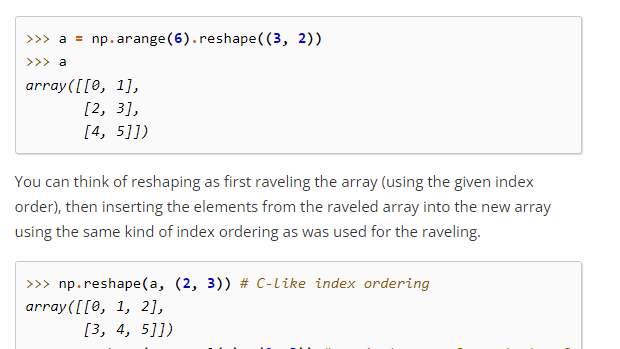
To są już zdefiniowane paczki danych.



Pętla zabiezpieczająca, aby nie trafić na ten sam zbiór danych.







**Sequential** – odpowiada ze przekazywanie danych między warstwami.

**LSTM** (ang. Long short term memory) – bardzo mocne, sieci neuronowe rekurencyjne ze sprzężeniem zwrotnym czasami wyciąga za dużo z danych, podają do tylu RNN (ang. Recurrent Neural Networks), do danych zmiennych w czasie

Możliwe jest uzyskanie dostępu do wyjścia stanu ukrytego dla każdego wejściowego kroku czasowego. Można to zrobić, ustawiając atrybut return\_sequences na True podczas definiowania warstwy **LSTM**.

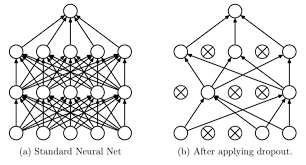
input\_shape=[num\_days,data\_size] – na początku musimy mieć macierz

**Dropout** - Stosuje usuwanie do danych wejściowych warstwę. Rate -> wartość zmienna z zakresu od 0 do 1. Ułamek jednostek wejściowych do usunięcia.

Warstwa Dropout losowo ustawia jednostki wejściowe na 0 z częstotliwością `rate w każdym kroku podczas treningu, co pomaga uniknąć przepełnienia. Wejścia nie ustawione na 0 są skalowane przez 1/(1 - rate) tak, że suma wszystkich wszystkich wejść pozostaje niezmieniona.

Zauważ, że warstwa Dropout jest stosowana tylko wtedy, gdy `training` jest ustawione na True

co oznacza, że żadne wartości nie są odrzucane podczas wnioskowania. Gdy używamy `model.fit`, `training` będzie automatycznie ustawiony na True, a w innych w innych kontekstach można ustawić kwarg na True podczas wywoływania warstwy. (Jest to przeciwieństwo do ustawienia `trainable=False` dla warstwy Dropout. Ustawienie `trainable` nie wpływa na zachowanie warstwy, ponieważ warstwa Dropout nie posiada żadnych zmiennych/wag, które mogłyby zostać zamrożone podczas treningu).



**Flatten** - Spłaszcza dane wejściowe. Nie ma wpływu na rozmiar partii.

Uwaga: Jeśli dane wejściowe mają kształt `(batch,)` bez osi charakterystycznej, wówczas

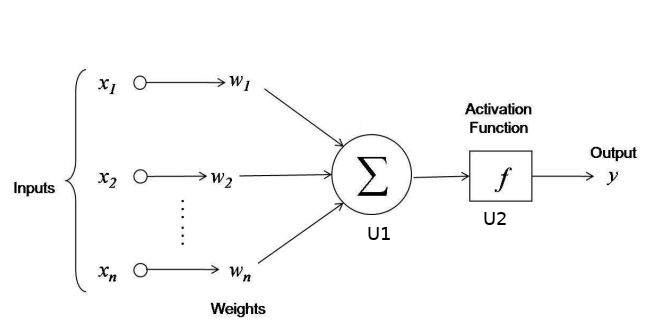
spłaszczenie dodaje dodatkowy wymiar kanału, a kształtem wyjściowym jest `(batch, 1)`.

Flatten - rozciąga do wektora.

**Dense** - Po prostu zwykła, gęsto połączona warstwa NN. Dense ma funkcję aktywacji, to prosta warstwa, która oblicza swoje i podaje dalej.

Warstwa `Dense` implementuje operację: `output = activation(dot(input, kernel) + bias)`.

gdzie `aktywacja` jest funkcją aktywacji w całym elemencie przekazaną jako argument `activation`, `kernel` jest macierzą wag utworzoną przez warstwę, a `bias` jest wektorem uprzedzeń utworzonym przez warstwę(ma zastosowanie tylko wtedy, gdy `use\_bias` jest `True`). Wszystkie te atrybuty są atrybutami `Dense`.

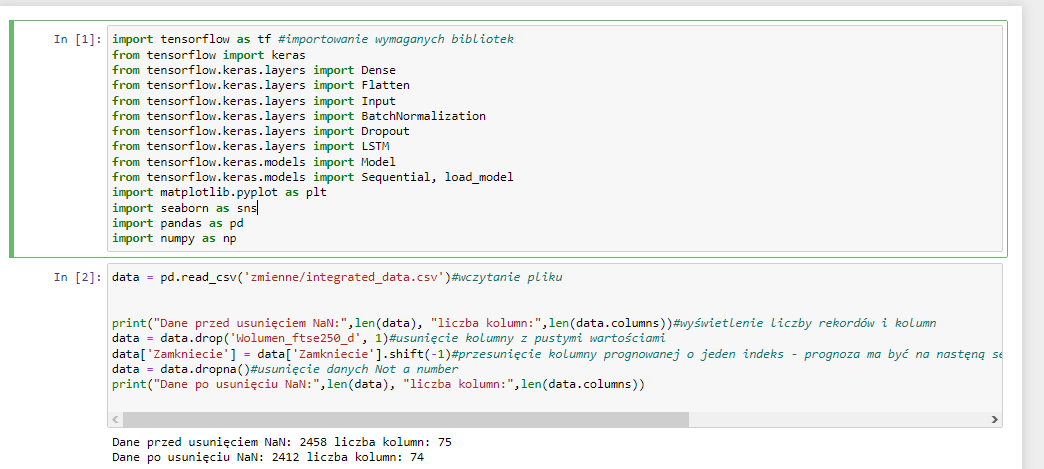


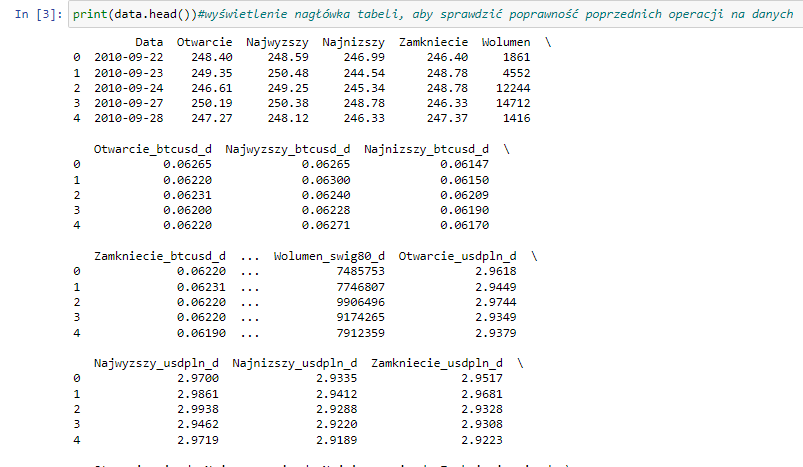
Funkcja aktywacji

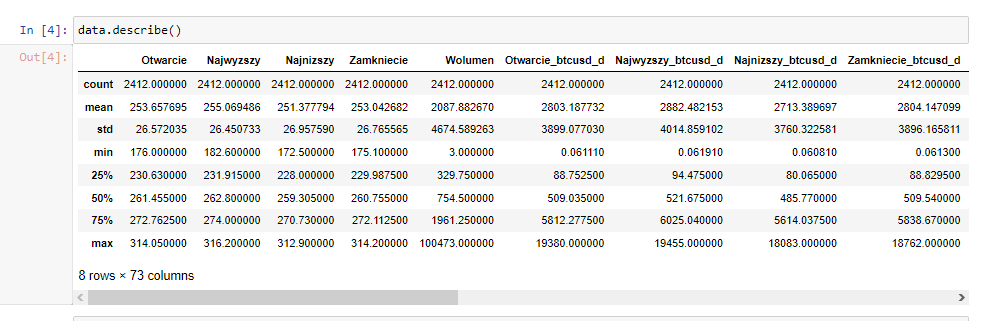
według funkcji aktywacji obliczana jest wartość wyjścia neuronów sieci neuronowej. Po agregacji danych wejściowych z uwzględnieniem wag powstaje sygnał sumarycznego pobudzenia. Rola funkcji aktywacji polega na tym, że musi ona określić sposób obliczania wartości sygnału wyjściowego neuronu na podstawie wartości tego sumarycznego pobudzenia. Wyniki są przepuszczane przez funkcję aktywacji, która określa wartość wyjściową. Jeśli wartość wyjściowa przekroczy dany próg, to „wyzwoli” (lub aktywuje) węzeł, który przekaże dane do następnej warstwy w sieci. W ten sposób dane wyjściowe jednego węzła stają się danymi wejściowymi kolejnego węzła. Ten proces przekazywania danych z jednej warstwy do następnej definiuje sieć neuronową jako sieć jednokierunkową (feedforward).

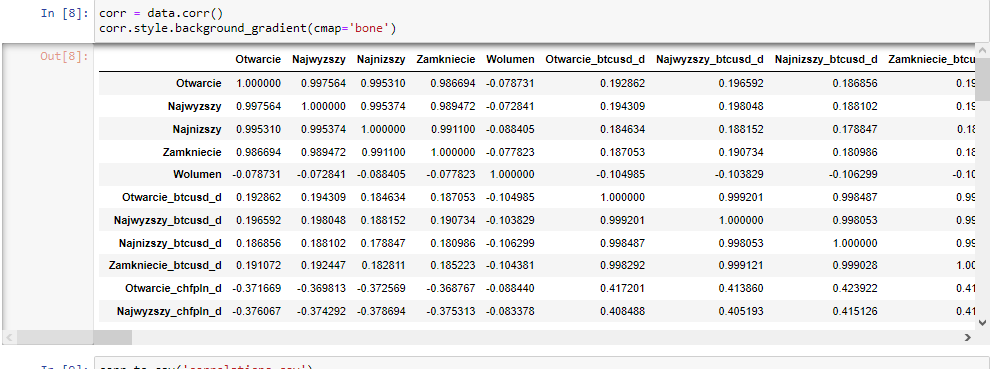
Sigmoid – przez to, że skalowaliśmy w zakresie 0-1.

1. Tworzenie wirtualnego środowiska Tensorflow: pip install --user virtualenv
2. Po skonfigurowaniu środowiska wirtualnego użyj następującego polecenia, aby zainstalować pakiet Tensorflow pip: pip install --upgrade tensorflow
3. Pip install sklearn

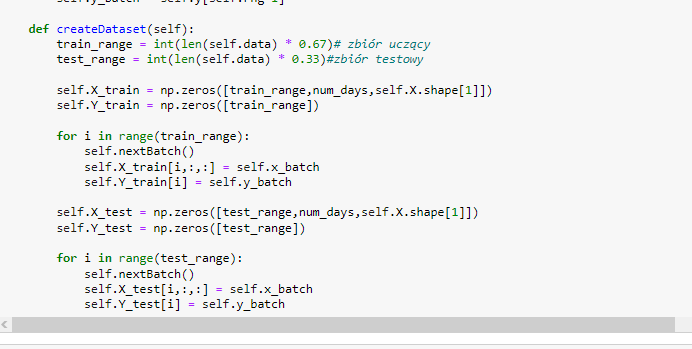
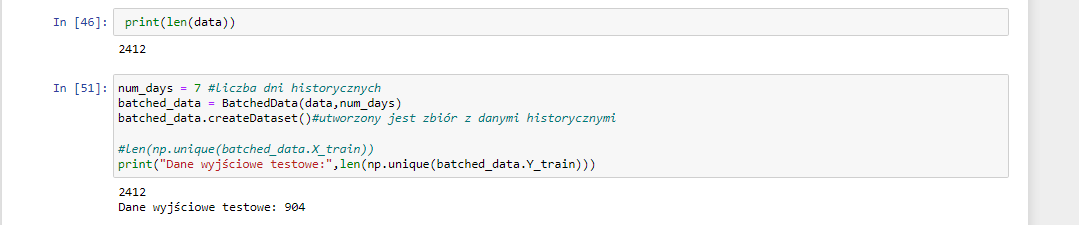








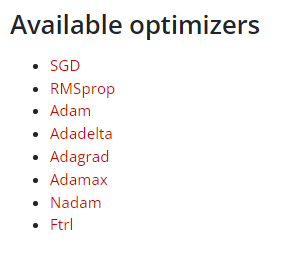




**model.compile** (kompilowanie modelu)- definiuje funkcję straty, optymalizator i metryki. Do trenowania potrzebny jest skompilowany model (ponieważ trening wykorzystuje funkcję straty i optymalizator).

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.

**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)

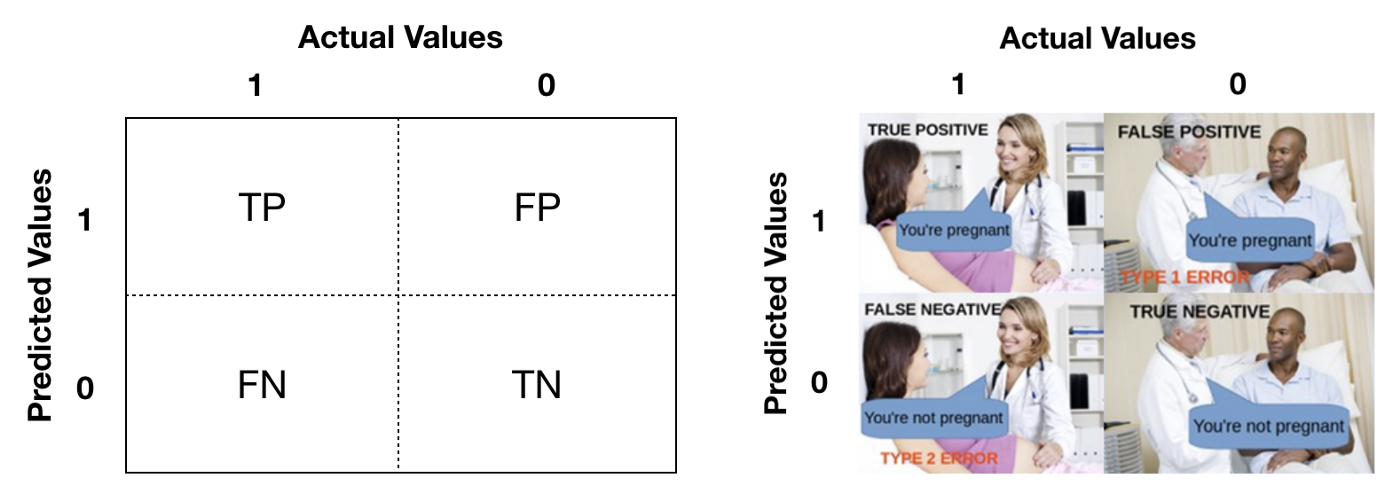
Swoistość = TN/(TN+FP)

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)) ).
* 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.

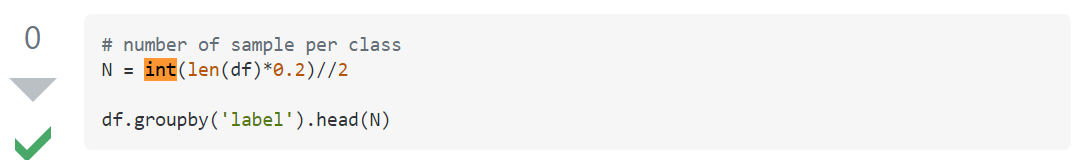
Czyli jeśli za pomocą jakiegoś modelu otrzymamy prognozę równą „44” a po obliczeniu MAE dla okresu testowego wyniesie ono Mae = 2,456, wtedy nasza prognoza = 44 może odbiegać od rzeczywistej wartości przyszłej o 2,456 czyli może być z przedziału od 41,544 do 46,456.

* 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.

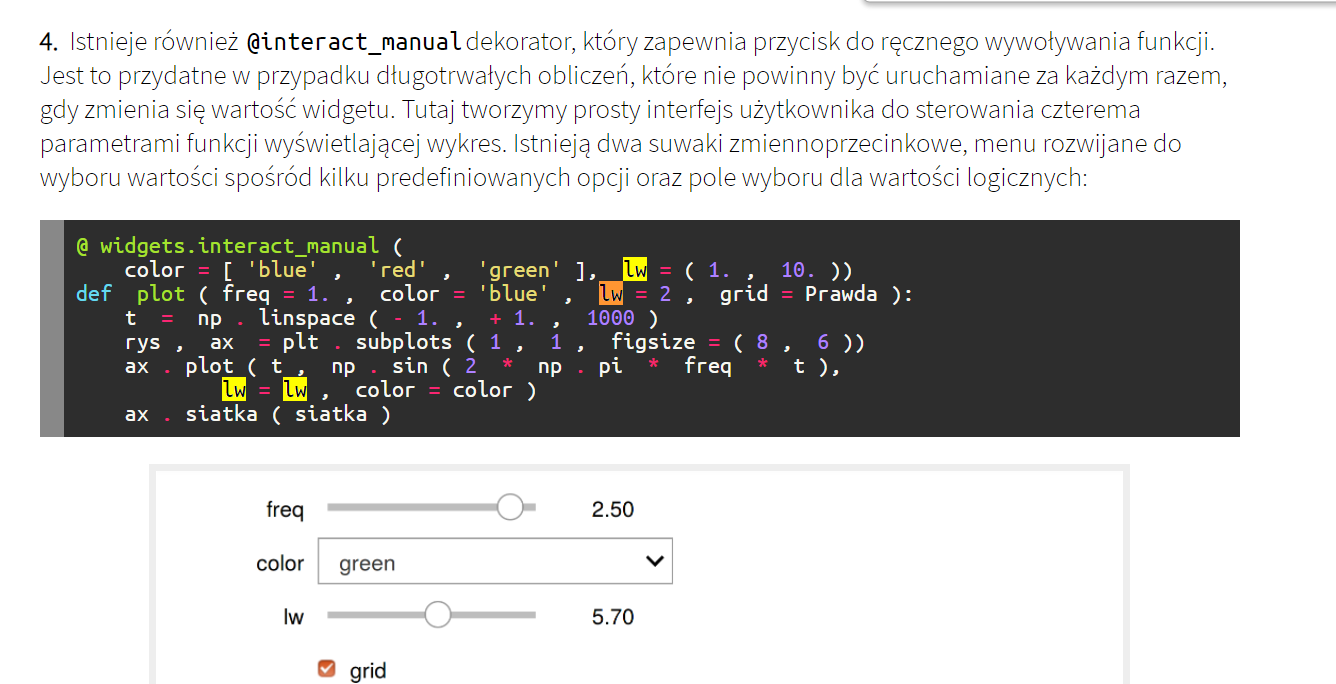
Jeśli przekazana jest podklasa BaseIndexer, oblicza granice okna na podstawie zdefiniowanej get\_window\_boundsmetody. Dodatkowe argumenty słów kluczowych, mianowicie *min\_periods* , *center* i *closed,* zostaną przekazane do *get\_window\_bounds* .

**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#numpy.array), [**zeros**](https://numpy.org/doc/stable/reference/generated/numpy.zeros.html#numpy.zeros)lub [**empty**](https://numpy.org/doc/stable/reference/generated/numpy.empty.html#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)

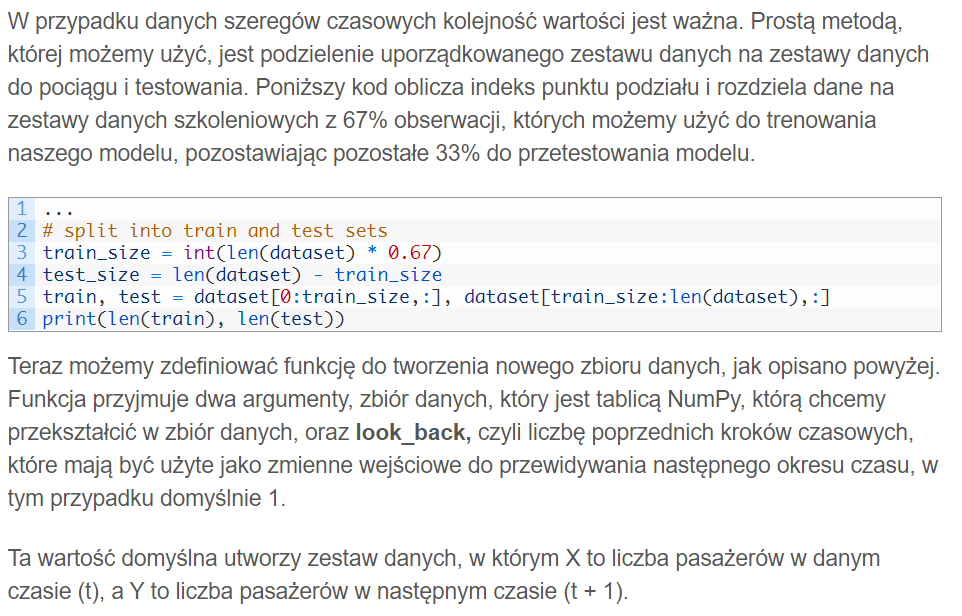
[https://www.w3schools.com ›numpy\_array\_reshape](Zmiana kształtu tablicy NumPy - W3Schoolshttps://www.w3schools.com ›numpy_array_reshape)

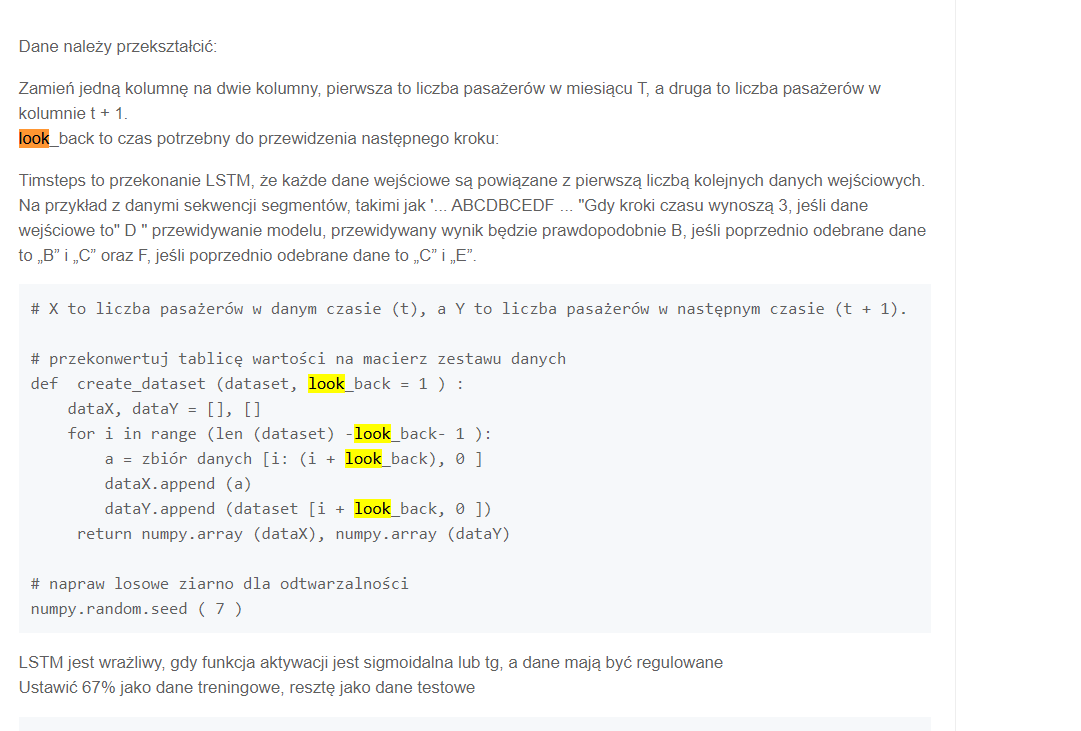
Importowanie i używanie *MinMaxScaler* działa - tak jak wszystkie następne skalery - w dokładnie taki sam sposób jak *StandardScaler* . Jedyna różnica polega na parametrach przy inicjowaniu nowej instancji.

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?

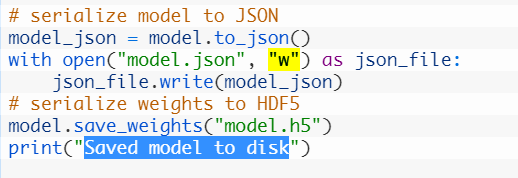
**Adam** jest zastępczym algorytmem **optymalizacji** dla stochastycznego zejścia w gradiencie do trenowania modeli uczenia głębokiego. **Adam** łączy najlepsze właściwości algorytmów AdaGrad i RMSProp, aby zapewnić algorytm **optymalizacji** , który radzi sobie z rzadkimi gradientami w przypadku hałaśliwych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