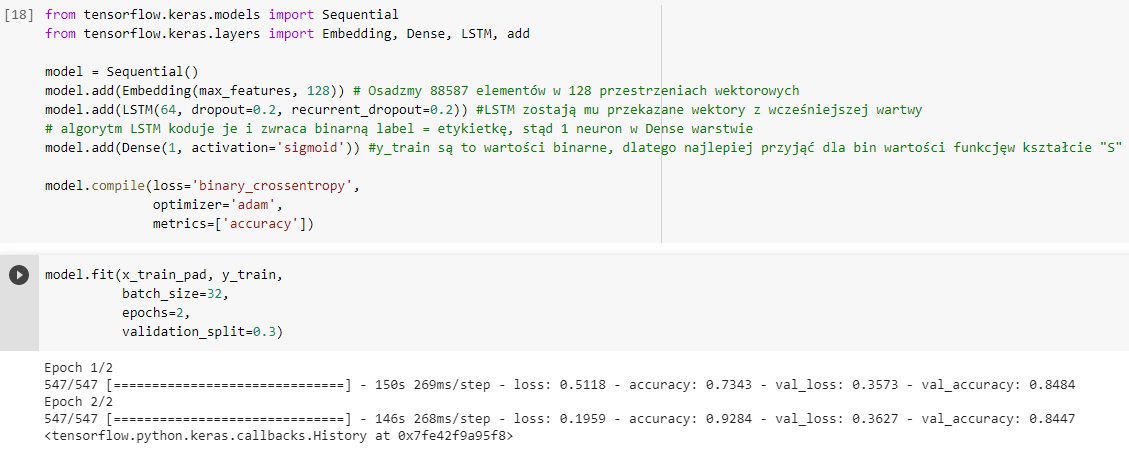
Zad. 2 Neuronowa sieć gęsta dla problemu klasyfikacji recenzji filmowych z bazy IMDB.

Mamy tutaj do czynienia z analizą sentymentalną:

Jest to proces obliczeniowego identyfikowania i kategoryzowania opinii z kawałków tekstu i określania czy nastawienie opiniodawcy wobec konkretnego zagadnienia produktu jest pozytywne, negatywne, czy neutralne.

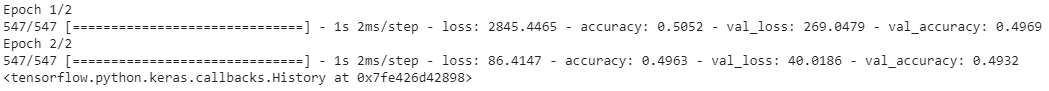


Pytania i eksperymenty:

Czy sprawność sieci zmieni się jeżeli:

* będzie tylko jedna warstwa gęsta?

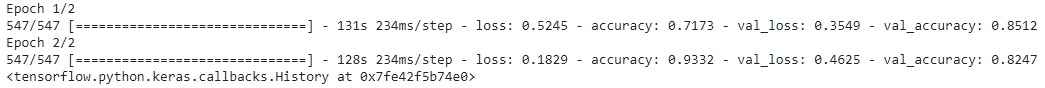
Zostawiłam warstwę Dense i nic zaskakującego się nie pojawiło. Wyniki są beznadziejne jak można było przewidzieć.



* będą trzy warstwy gęste?

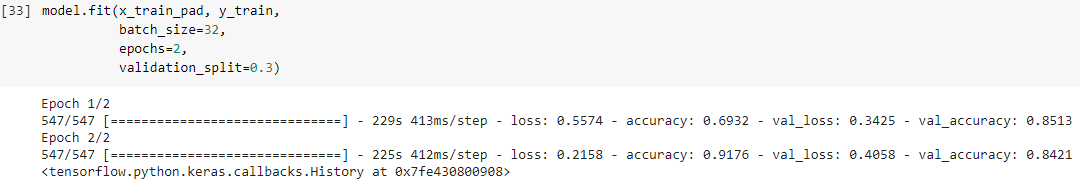
Jest to wyjściowa forma.

* liczba jednostek w warstwie/warstwach zmniejszy się?

W warstwie LSTM liczbę jednostek zmniejszyłam do 32 i oto wyniki:  


Najwyraźniej nie ma to znaczącego wpływu, ponieważ wyniki są bardzo zbliżone do tych uzyskanych z liczbą jednostek równą 64, a nawet precyzja dla zbioru uczącego się tutaj jest większa. Za to istotnie przyspieszyło proces z 270ms/step na 234ms/step.

* liczba jednostek w warstwie/warstwach zwiększy się?

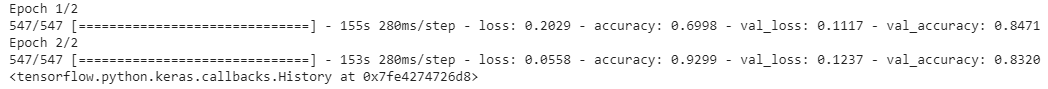


Wyniki są bardzo podobne do tych z mniejszą liczbą jednostek, lecz lekko gorsze. Za to warto zwrócić uwagę na to że czas uczenia jest krótszy.

Dodam od siebie, ze jestem świadoma, że gdybym dodała znacznie więcej neuronów to mogłoby dojść do overfittingu - przetrenowania modelu i wyniki były by bardzo niezadowalające.

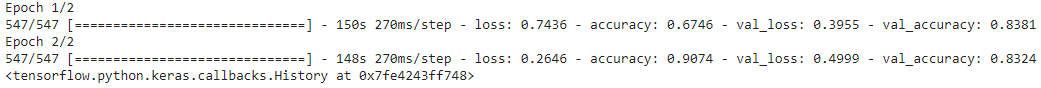
* zamiast entropii krzyżowej wykorzystamy inną funkcję straty, np. błąd średniokwadratowy?

Wykorzystałam proponowany ‘mse’



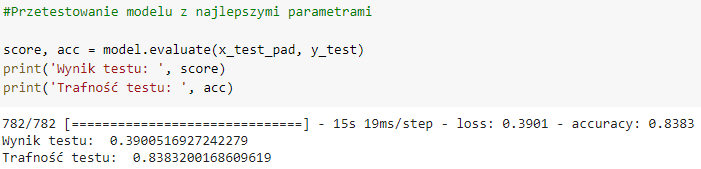
Wyniki są prawie takie jak oryginał. Finalnie zbiór testowy wypada lepiej pod względem trafności (accuacy) z funkcją straty binary\_crossentropy, natomiast dla zestawu uczącego się wynik poprawił się nieznacznie.

* w ostatniej warstwie funkcją aktywacji będzie tangens hiperboliczny?



Czas trwania epoki jest praktycznie taki sam. Trafność (accuracy) modelu jest gorsza w porównaniu z funkcją sigmoid, a strata (loss) znacznie wyższa. Dla danego zbioru i problemu lepsza okazała się funkcja sigmoid.

Na podstawie powyższych eksperymentów wybierz najlepsze wartości hiperparametrów sieci (liczba warstw, liczba jednostek w warstwie, funkcje aktywacji, rozmiar wsadu, liczba epok, itd.) Dla najlepszych wartości hiperparametrów wytrenuj sieć na całym zbiorze treningowym (25000 próbek). Sprawdź trafność klasyfikacji (accuracy) tak wytrenowanej sieci na zbiorze testowym.



Zad.3

Czy dla sieci neuronowej i zbioru „reuters” (https://keras.io/api/datasets/reuters/) sprawność zmieni się w stosunku do Rozdziału 3.6 [1] lub [2] jeżeli:

* będzie tylko jedna warstwa gęsta?
* będą trzy warstwy gęste?
* liczba jednostek w warstwie/warstwach zmniejszy się?
* liczba jednostek w warstwie/warstwach zwiększy się?
* rozmiar wsadu zmniejszy się?
* rozmiar wsadu zwiększy się?

Dla najlepszych wartości hiperparametrów znalezionych w trakcie powyższych eksperymentów, wytrenuj sieć na całym zbiorze treningowym. Sprawdź trafność klasyfikacji (accuracy) tak wytrenowanej sieci na zbiorze testowym.

Jest to prawie takie samo zadanie jak zad.1. dlatego pozwolę sobie je pominąć. Pewnie Pan na zajęciach omawiał poszczególne wariacje i było to bardzo kształcące. Ja sama przeklikując żadnych nowych wniosków nie wyciągnę.

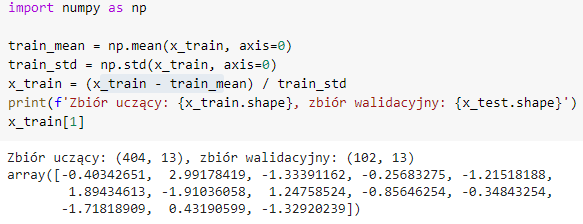
Zad 4

1. Eksperymentalnie dobierz najlepsze wartości hiperparametrów sieci neuronowej dla zadania regresji dla danych ze zbioru „Boston Housing” (https://keras.io/api/datasets/boston\_housing/).

Zadanie polega na przewidzeniu ceny domów. W zbiorze mamy 13 feutures, czyli cech z których oszacowywana będzie cena dla konkretnego domu, a wierszy jest 506.

Przeprowadziłam normalizację funkcji/cech. Normalizacja cech polega na odjęciu średniej cech od każdej cechy i podzieleniu każdego wyniku przez odchylenie standardowe.

Oczywiście taka normalizacja jest konieczna, ponieważ chcemy, aby dane były jakoś jednostkowe, a nie rozproszone.



W modelu zastosowałam rmsprop jako funkcję optymalizującą proces uczenia się. Wynika to z podstawowej wiedzy statystycznej, czym jest rmse:

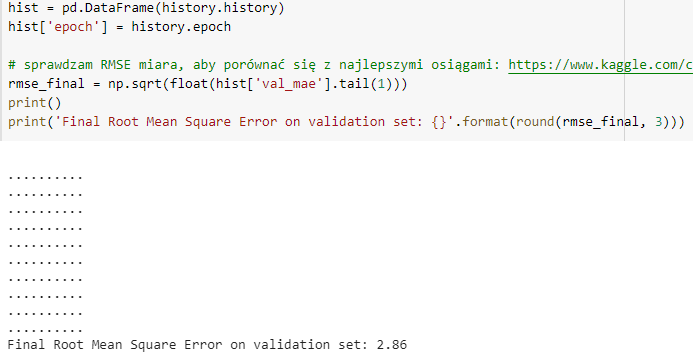
Podstawowy błąd średniokwadratowy (RMSE) to odchylenie standardowe reszt (błędów przewidywania). Reszty są miarą odległości od punktów danych linii regresji; RMSE jest miarą rozłożenia tych reszt. Innymi słowy, informuje, jak skoncentrowane są dane wokół linii najlepszego dopasowania. Podstawowy błąd średniokwadratowy jest powszechnie stosowany w klimatologii, prognozowaniu i analizie regresji w celu weryfikacji wyników eksperymentalnych.

Matryka ‘mae’ ukazuje nam mean absolute error, czyli średni błąd bezwzględny, a ‘mse’ to to co opisałam powyżej - jest to rónica między estymatorem, a wartością estymowaną.

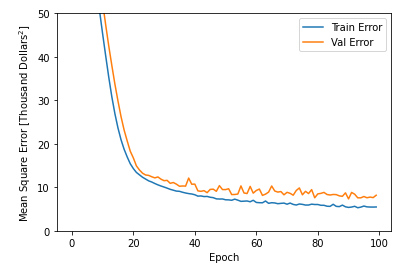
Model stopuje się po 50 epokach w których nie doszło do poprawy wyniku straty zbioru walidacyjnego ‘val\_loss’



Model i wyniki:

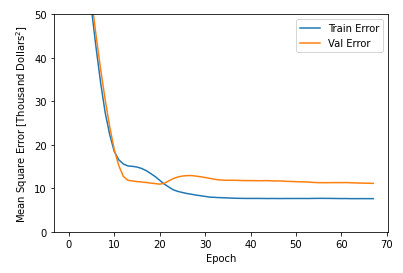


Wizualizacja loss:



Doszło do overfittingu, pomimo, że zmniejszyłam ilość warstw i neuronów.

Zmienię funkcję optymalizacji na adam:



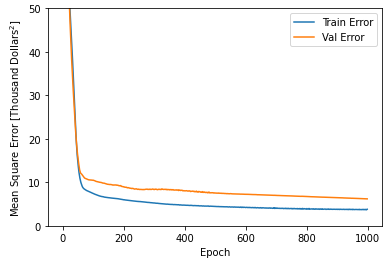
Final Root Mean Square Error on validation set: 3.337

Zdecydowana poprawa, teraz dodam ilość epok i odkomentuję warstw.

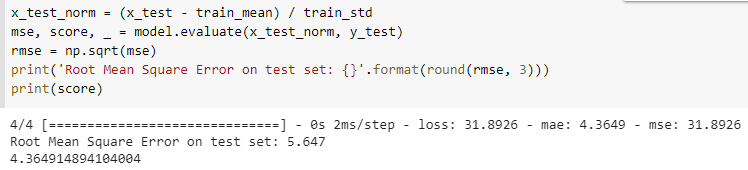
No niestety znowu doszło do overfittingu, lecz przy zachowanej ilości warstw i neuronów, a epochs=1000 i optimizer=’adam’ uzyskałam świetne rezultaty. Bardzo niski błąd:

Final Root Mean Square Error on validation set: 2.488

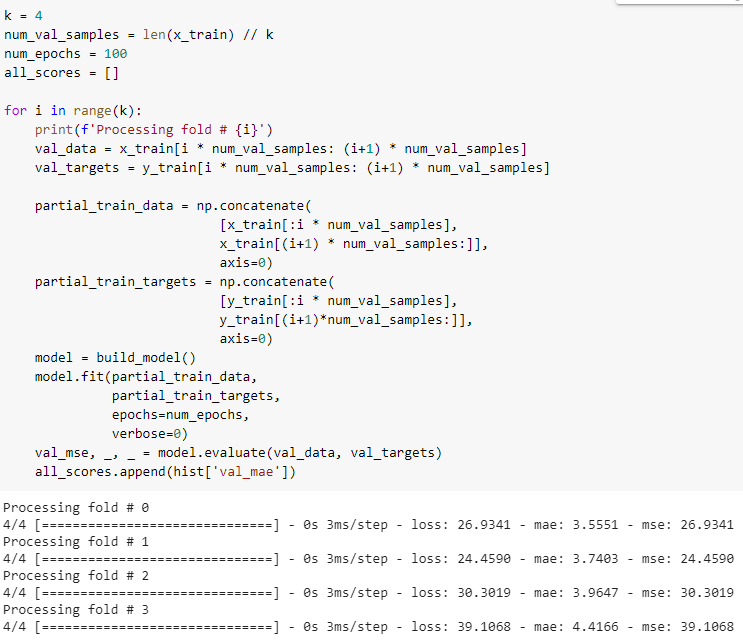
I brak overfittingu:



1. Sprawdzenie modelu na zbiorze testowym już nie pała takim optymizmem:



1. Czy zastosowanie K-składowej walidacji krzyżowej poprawia proces doboru wartości hiperparametrów?

Nie ma poprawy:

Dodatkowe wnioski: Pomimo, że posłużyłam się RMSE, aby ukazać błędy predykcji i jest to dość miarodajne tutaj to jednak warto pamiętać, że dla zbioru z odstającymi danymi wynik może być wyolbrzymiony. RMSE jest na to wrażliwe, bo wpływ każdego błędu na RMSE jest wprost proporcjonalne do kwadratu błędu.

Podobno w rzeczywistości i tak używa się kilku metryk na oszacowanie regresji, więc nie ma co się nadto przejmować powyższą uwagą.

Zad.5

Pobierz zbiór Iris (https://www.tensorflow.org/datasets/catalog/iris?hl=en) z modułu TensorFlow Datasets. Dobierz najlepsze wartości hiperparametrów sieci na podstawie:

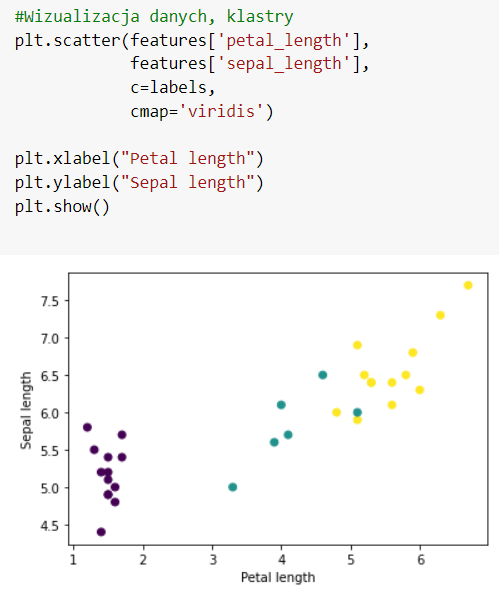
[na ocenę 3.0]: K-składowej walidacji krzyżowej.

K-składowa walidacja krzyżowa. Najpierw ze zbioru danych wydzielany jest zbiór testowy. Następnie pozostałe dane są dzielone na K składowych. Każda z tych składowych jest następnie zbiorem walidacyjnym, a pozostałe K-1 składowych składa się na zbiór treningowy. Walidacja następuje na podstawie średnich danych walidacyjnych z K treningów. Na podstawie walidacji wybieramy najlepsze wartości hiperparametrów, a następnie uczymy sieć na całym zbiorze danych oprócz części testowej. Ocena wyuczonej sieci następuje na zbiorze testowym.

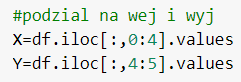
W sprawozdaniu zamieść i opisz:

* wykorzystany kod,
* sposób doboru hiperparametrów,
* wykresy funkcji straty i trafności dla walidacji,
* wyniki uzyskane dla najlepszej kombinacji parametrów oraz rysunek finalnej sieci wykonany przy wykorzystaniu keras.utils.plot\_model,
* Wnioski.

Do wizualizacji tych danych warto użyć scatter plot. Funkcja ta ukaże rozkład danych i ich relacje:

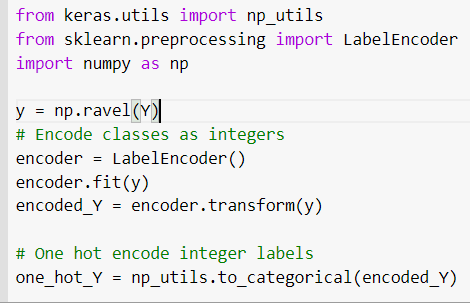


Teraz dane wystarczy podzielić na wejścia wyjścia:

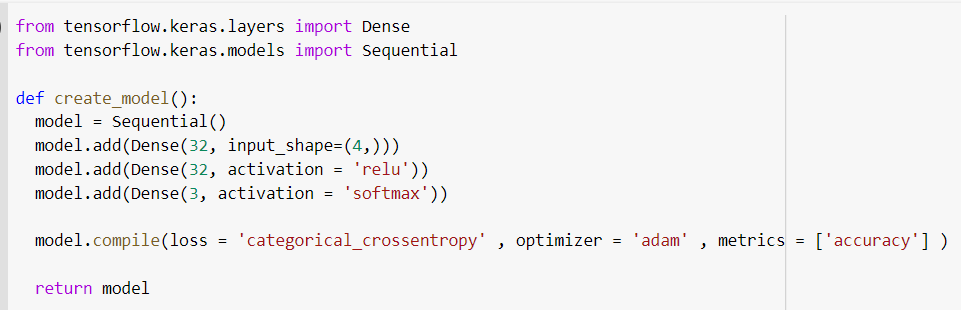


Dane ze zbioru iris z tensorflow są już znormalizowane. Należy jednak pamiętać o kodowaniu labels, po pierwsze ze słownych danych do danych matematycznych – tutaj już tak jest- a następnie najlepiej do one hot encode.

One hot różni się tym od innego kodowania, że długość tego wektora równa jest ilości klas do określenia, bo w miejscu danej kategorii mamy 1, a w całej reszcie zera. Dzięki takiemu kodowaniu danych nasz algorytm będzie lepiej działać w kontekście przewidywania.



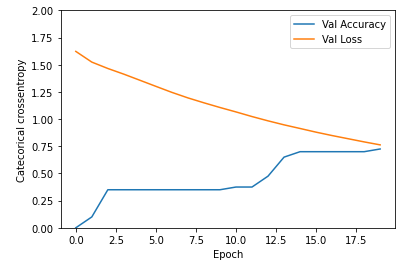
Model będziemy wywoływać w pętli, dlatego zaprogramowałam go zdefiniowałam oddzielnie:



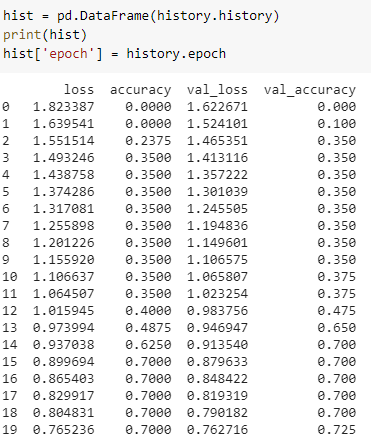
Hiperparametry dobierałam w oparciu o wiedzę teoretyczną, jak i manualnie opierając się o uzyskiwane wyniki z K-fold cross validation.

* Funkcję aktywacyjną wybrałam relu, aby pozbyć się wartości ujemnych, a następnie softmax - znormalizowana funkcja wykładnicza, najczęściej stosowana do klasyfikacji wielowymiarowej.
* Liczba neuronów w ostatniej warstwie oczywiście wynosi 3, tyle ile przewidujemy klas do wyszczególnienia.
* W powyższych warstwach ilość neuronów wynosi tyle ile zasugerowałam się z wyników k-fold cross validation. Tzn. Nie za dużo, aby nie doprowadzić do overfittingu, ani nie za mało, aby model był efektywny i najlepiej jego accuracy było około 0.9.

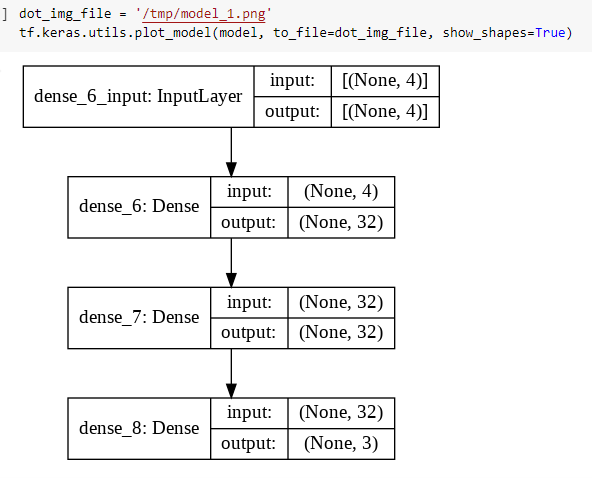
Wykresy funkcji straty i trafności dla walidacji:



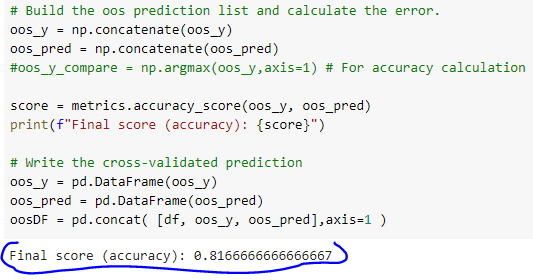
Wyniki uzyskane dla finalnej sieci:



Rysunek finalnej sieci:



Finalna wartość trafności (accuracy):



Wnioski:

Dzięki K-fold cross validation jestem w stanie uzyskać predictions na podstawie całego zbioru danych.

Analizując cały kod i jego wyniki można zaobserwować, że udało mi się osiągnąć całkiem pozytywne wyniki trafności. Wnioskować z tego można, że dla takiego zbioru jak iris wystarczy 20 epok i podział na 3 zestawy.

Im więcej zestawów tym dłuższy czas obliczania.

Zadanie 5. b)

K-składowa walidacja krzyżowa ze zbiorem walidacyjnym i testowym. Dane są dzielone na K składowych. Każda z tych składowych jest następnie zbiorem testowym, a z pozostałych K-1 składowych każda staje się zbiorem walidacyjnym, a pozostałe K-2 składowych składa się na zbiór treningowy. Walidacja następuje na podstawie średnich danych walidacyjnych z K\*(K-1) treningów. Na podstawie walidacji wybieramy najlepsze wartości hiperparametrów, a następnie uczymy sieć K-krotnie – za każdym razem na wszystkich K-1 składowych, które nie są w danym przebiegu zbiorem testowym. Ocena wyuczonej sieci jest średnim wynikiem dla K zbiorów testowych.