

and betal width. Output neuron O1, O2 and O3 represent the predicted class (This setosa, Iris versicolor or Iris virginica) Activation Function: (i) Relu (ii) Sigmoident) softmax i) Relu-used in hidden layers of the model formula! $\phi(x_1) = \max\{x_1, 0\}, \phi$ Joinula: $\phi(x_i) = e^{x_i}$, $\phi(x_i) = e^{x_2}$, $\phi(x_3) = e^{x_3}$ $= e^{x_4}$ $= e^{x_4$ such that, $\phi(x_1) + \phi(x_2) + \phi(x_3) = 1$ Loss Junction (for Multinomial Logistic Regression) c) Categorial crossentropy- It is a multiclass extension/version of binary cross entropy. formula: -log(ýc(i)) which is the class of true class (i One hot encoding! -) It is a machine learning technique which converts given categorical information into a format such that all other inputs are zeros Exifor ivis uselosa oc, = [100] and ivis uregenica = [00] Codes import Sklearn os amonglutos import pandas as pd Prairie of Paramet 18 import setboom as sns import matphotlib. pyplot as plt Non-trainable pari import numpy as hp df = pd. read_csv(Tris.csv) x = dj. loc [:,:5], Value y = dj. loc [:,-1], Value

Code: # no hidden layer model = lequential() model. add(normalizes) model. add (layers Dense (Units = 3, activation = (softmax')) model·compil(loss = "(alegorical-crossintiopy") optimizer = "adam", metrics = ["accuracy"]) model·summary() and project brighten in the

history = model. fit(x-train, y-train, epachs = 10, batch-size = 2, Validation -data = (x-test, y-test))

loss, accuracy = model evaluate (x-test, y-test)

Test-loss append (loss)

Test-accuracy, append (accuracy)

loss, acturacy = model. evaluate (xtrain, y-train)
Train-dold. append (loss)

Train-accuracy append (accuracy)

Output: Model: "sequential

Layer (dybe)	Outfut shape	Param#	
Inormalization [Normalization]	(None,G)	11	
dense	(None, 3)	20 2180 x	

byen solven traffel

202 80 modelle the god

Harman Jugar

populated tom tom

Total params 129

Trainable params: 18

Non-trainable params: 11

```
code: # with 1 hidden layer
                              ( ) with many by
   model = sequential ()
   model. add (normalized)
                              resimmer who resums
   model. add (layer. Dense (8, activation = "relu", in put dim= 5))
   model · add (layer. Dense (Units = 3, activation= 'softmax'))
   model-compile (loss = "Categorical-crossentropy", optimizer = "adam
            metrices = ["accuracy"])
  model summary ()
  history = model. fit (x-train, y-train, epochs=10, botch_size=2,
      Validation-data = (x-test, y-test).
  loss, accuracy = model evaluate(x-test, y-text)
  Test_accuracy; append (accuracy)
  Test, accuracy = model evaluate (x-train, y-train)
  Train loss. append (loss)
  Train-accuracy append (accuracy)
Output: Model: "sequential 1"
      Layer(type) Output shape Param#
                  (None, 5) 11
      normalization
      (Normalization)
   dense
              (None, 8) 48
      dense-1 (None, 3)
                        8 = HONV DRIEG Blow of books
```

Total params: 86

Trainable Paramy: 75

non-trainable Parand: 11

```
and while higher
 Code: # 2 hidden layers
    model = sequential ()
                                        Carolina of the A. Com
    model·add (normalizer)
    model add (layers Dense (8, activation = "relu", input_dimes))
    model add (Dense (unite = 8, Kernel initializer = 'uniform', activation
    model add (dayors. Dense (units=3, activation = 'softmax'))
   model compile (loss="(ategorical-crossentiopy", optimizer= "adam",
  model. summary (1.
history = model (1)
  history = model. fit(r-train, y-train, epoch = 10, batch-size=2, Validada,
-data=(x. test, y-test))
  Test-loss. append (loss)
  Test-accuracy: append (accuracy)
Loss, accuracy = model. evaluate(x-train) y-train)

Train - loss append (loss)

Train - accuracy, append (loss)

Code: # 150
       # with 3 hidden layers (3, mili)
       model-sequential()
                                              (mitus alumnos)
       model add (normalizer)
      model. add (layers. Dense (8) activation = "relu", input_dim=5))
      model. add (Dense (Units=8, Kernel-initializer = luniform, activation)
     model add [Dense (Units = 8, Kernel _initializer = "Uniform, activation
                                            = relu)
```

model add (layers. Dense (Units=3, activation="softmax")) model and poense filmitsee her model admipile (loss = " Categorical - crossentropy", optimizer: "adam", metric = [accuracy"]) model summary ()
hustory = model-fit (x-train, y-train, epoch = 10, batch-size = 2, validation -data = (x - test, y-test)) loss, accuracy = model. evaluate (x-test, y-test) loss, accuracy = model. evaluate (x train, y-train) Outputs: (2 hidden layers) (3 hidden layer) model: "sequential?" mo del: "sequential 3" Layer (type) Output shape Param# Layer (type) Output Shape Param # normalization (None,5) 11 normalization (None, s) 11 dense (Dense) (None, 8) dense (Dense) (None, 8) 48 48 dense-1 (Pense) (None,8) 72 dense 1 (Dense) (None, 8) dense_2 (Dense) (None, 8) 72 dense 2 (Densé) (None, 3) dense-3 (Dense) (None,3) 27 Total param: 230 To tal params: 158 Trainable Params: 2:19 Trainable Paramy 147 Non-trainable params:11 Non-trainable params: 11

(ode: # 4 hidden layers

model = sequential()

model. add(normalizer)

model. add(layers. Dense (8, activation = "relu", in put_din = s))

model. add (Dense (Units = 8, Kernel-initializer = 'uniform')

activation = 'relu'))

model "uniform" activations
add (Dense [Unites, Kernel-initializer 2
model add (Dense (Unite & Kernel - initializer = 'uniform' activations redu')) model add (Dense (Unite & Kernel - initializer = 'uniform' model add (Layers Dense (Units = 3, activalizer = 'softmax')) model add (Layers Dense (Units = 3, activalizer = "addense")
moderate (Vint de 86 Kernet Cinitratizat
De la add (Layers: Dense (Units = 3, activations soft add
model. Compile (loss = "Categorical-vorsentropy") optimizer= "adam
metrices = [a crus a cumit
model. Summaris
model. summary() history = model. fit(x-train, y-train, epoch = 10, batch= stze=2,
oder fit (x-Hain, y-tain, epoch = 10) bar onte
van nat m = data= 1 v fest u-test).
loss, accuracy = model. evaluate (x-tst, y-test)
"Todel . evaluate & train yetrain)
Outfut: Model: "Sequential 4"
sequential 4 minus
Layer(type) Output shape Param#
normalization (None, 5) 11
dense (dense) (None, 8) 48
dense-1 (dense) (None, 8) 72
dense_2 (dense) (None, 8) 72
dense-3 (dense) (None, 8) +2
dense-4 (dense) (None, 3) 27
Total Params: 302
Trainable Params: 291

Non-trainable Params: 11

((2=mils ded (interior) = cottonition (3) 2000 cond) lobe closes model odd (Bong (Unide & Kernel instalier - Unidan).

(milenron) bbs . 13bom

Result/observation:

	Test loss	Test actionally	Train Loss	Train always	Model	
	0.44	0.80	0.38	0.88	Ohidden layer	
	0.43	D. 77	0.29	0.93	1 hidden layer	
T	0.16	0.97	0.12	0.98	2 hidden layer	→ best model
7	0.22	0.97	0.18	0.98	3 hidden layer	
	0.42	0-60	0.35	0-74	4 hidden layer)-) worst model

Best Model!

- model with 2 hidden layers had the best test and train accuracy along with least loss in terms of testing and training, that is, only 16% and 15%
- -) 3 hidden layers is the next best performer with testloss of mly 2019 and train loss of 18%, along with train accuracy of 98%

Worst Model!

-) upon adding 4 hidden layers, the model's accuracy is observered to be the lowest, that is, test accuracy is only 60% and train accuracy is 74%.

Conclusion!

.: Models with about 2-3 hidden layers perform well.