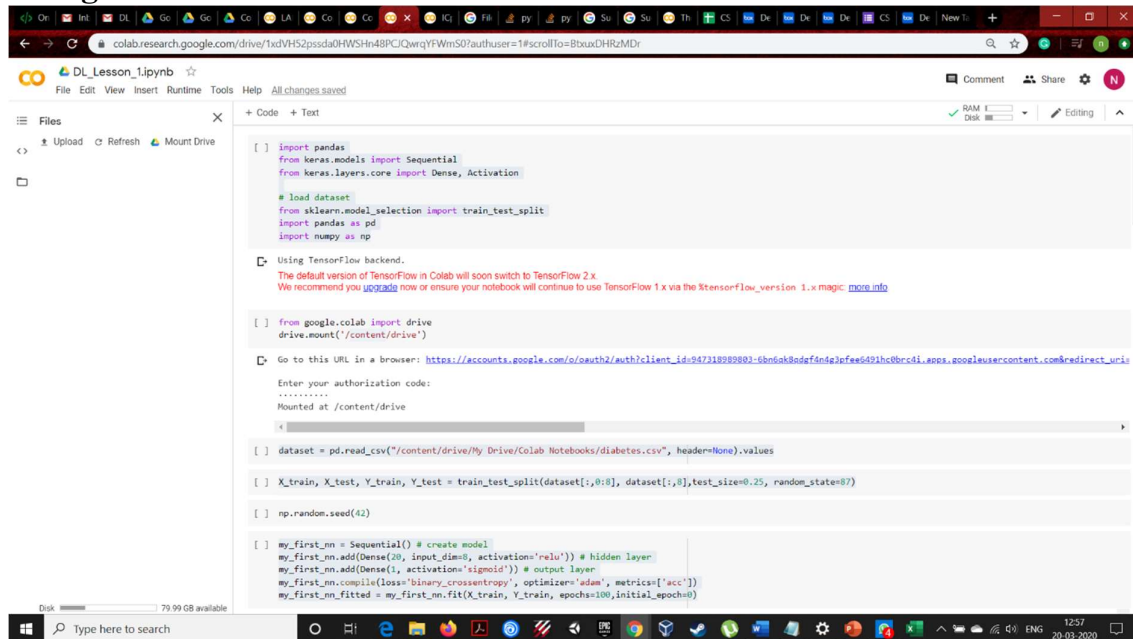


# ICP1\_MODULE2\_DL

## 1. Use the use case in the class:

a. Add more Dense layers to the existing code and check how the accuracy changes.



```
[ ] import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# load dataset
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

[ ] from google.colab import drive
drive.mount('/content/drive')

[ ] dataset = pd.read_csv("/content/drive/My Drive/Colab Notebooks/diabetes.csv", header=None).values

[ ] X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8], test_size=0.25, random_state=87)

[ ] np.random.seed(42)

[ ] my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0)
```

```
colab.research.google.com/drive/1xdVH52psda0HWSHn48PCjQwrqYFwM50?authuser=1#scrollTo=BtuxDHRzMDr

DL_Lesson_1.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[ ] 576/576 [=====] - 0s 49us/step - loss: 0.5182 - acc: 0.7517
Epoch 98/100
576/576 [=====] - 0s 46us/step - loss: 0.5362 - acc: 0.7535
Epoch 99/100
576/576 [=====] - 0s 44us/step - loss: 0.5184 - acc: 0.7500
Epoch 100/100
576/576 [=====] - 0s 48us/step - loss: 0.5076 - acc: 0.7795

[ ] print(my_first_nn.summary())

Model: "sequential_1"
Layer (type) Output Shape Param #
-----
dense_1 (Dense) (None, 20) 180
-----
dense_2 (Dense) (None, 1) 21
-----
Total params: 201
Trainable params: 201
Non-trainable params: 0
None

[ ] print(my_first_nn.evaluate(X_test, Y_test))
192/192 [=====] - 0s 246us/step
[0.6724422723054886, 0.6666666666666666]
```

```
colab.research.google.com/drive/1xdVH52psda0HWSHn48PCjQwrqYFwM50?authuser=1#scrollTo=BtuxDHRzMDr

DL_Lesson_1.ipynb
File Edit View Insert Runtime Tools Help All changes saved

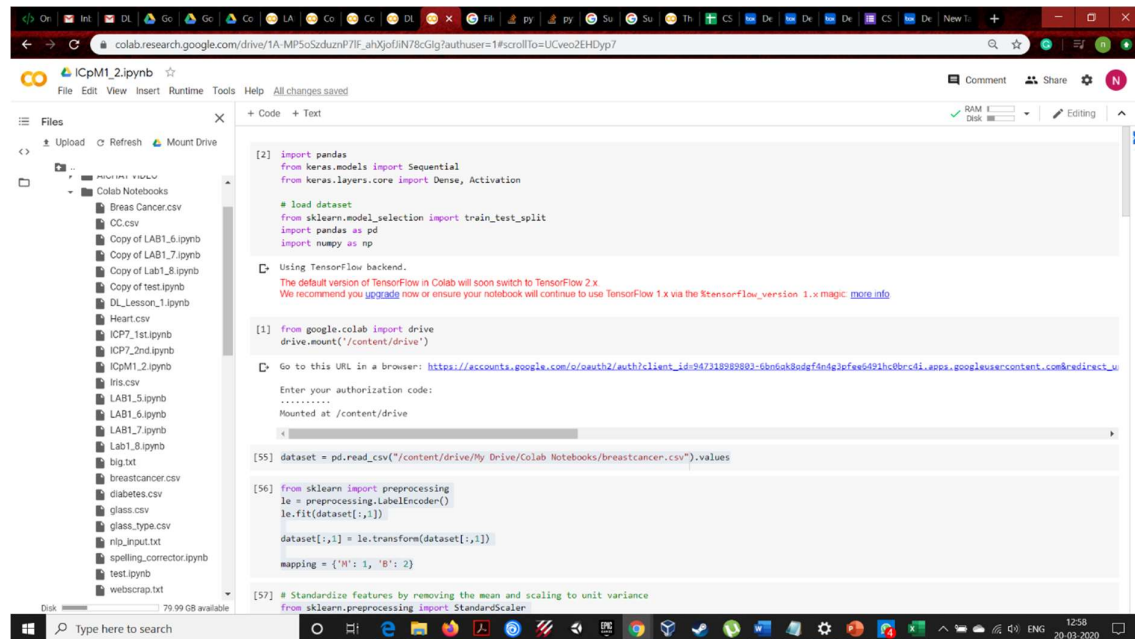
+ Code + Text
[ ] Epoch 97/100
576/576 [=====] - 0s 46us/step - loss: 0.5455 - acc: 0.7413
Epoch 98/100
576/576 [=====] - 0s 54us/step - loss: 0.5425 - acc: 0.7188
Epoch 99/100
576/576 [=====] - 0s 45us/step - loss: 0.5394 - acc: 0.7344
Epoch 100/100
576/576 [=====] - 0s 48us/step - loss: 0.5461 - acc: 0.7344

[ ] my_first_nn = Sequential() # create model
my_first_nn.add(Dense(40, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=97)

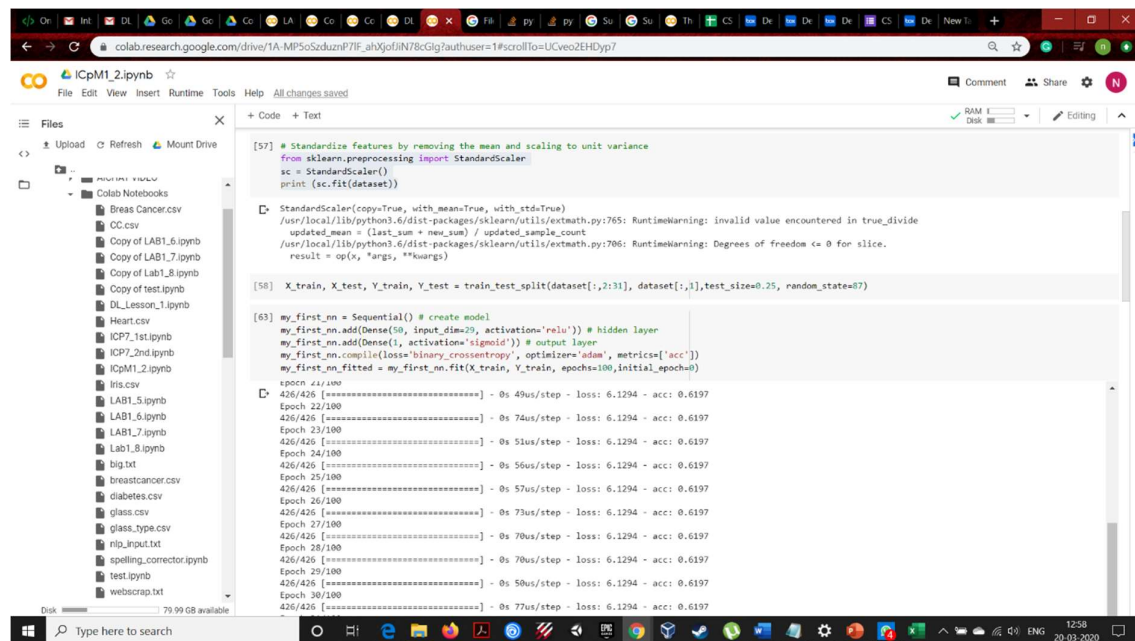
Epoch 1/100
576/576 [=====] - 0s 596us/step - loss: 3.1621 - acc: 0.5938
Epoch 2/100
576/576 [=====] - 0s 36us/step - loss: 2.3460 - acc: 0.6233
Epoch 3/100
576/576 [=====] - 0s 45us/step - loss: 2.0439 - acc: 0.6094
Epoch 4/100
576/576 [=====] - 0s 43us/step - loss: 1.7227 - acc: 0.5868
Epoch 5/100
576/576 [=====] - 0s 59us/step - loss: 1.5028 - acc: 0.5990
Epoch 6/100
576/576 [=====] - 0s 54us/step - loss: 1.2667 - acc: 0.6146
Epoch 7/100
576/576 [=====] - 0s 53us/step - loss: 1.0901 - acc: 0.6163
Epoch 8/100
576/576 [=====] - 0s 46us/step - loss: 0.9755 - acc: 0.6458
Epoch 9/100
576/576 [=====] - 0s 47us/step - loss: 0.9029 - acc: 0.6319
Epoch 10/100
576/576 [=====] - 0s 41us/step - loss: 0.8510 - acc: 0.6493
Epoch 11/100
576/576 [=====] - 0s 49us/step - loss: 0.7846 - acc: 0.6510
Epoch 12/100
576/576 [=====] - 0s 40us/step - loss: 0.7846 - acc: 0.6545
Epoch 13/100
576/576 [=====] - 0s 41us/step - loss: 0.7073 - acc: 0.6633
```

## 2. Change the data source to Breast Cancer dataset

**\* available in the source folder and make required changes**



**3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).**



#### 4. Try new different optimizers and report the accuracy for each one.

```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(50, input_dim=29, activation='relu')) # hidden layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, initial_epoch=0)
```

Epoch 1/100  
426/426 [=====] - 0s 679us/step - loss: 9.8798 - acc: 0.3803  
Epoch 2/100  
426/426 [=====] - 0s 48us/step - loss: 9.8798 - acc: 0.3803  
Epoch 3/100  
426/426 [=====] - 0s 56us/step - loss: 9.8798 - acc: 0.3803  
Epoch 4/100  
426/426 [=====] - 0s 53us/step - loss: 9.8798 - acc: 0.3803  
Epoch 5/100  
426/426 [=====] - 0s 61us/step - loss: 9.8798 - acc: 0.3803  
Epoch 6/100  
426/426 [=====] - 0s 50us/step - loss: 9.8798 - acc: 0.3803  
Epoch 7/100  
426/426 [=====] - 0s 66us/step - loss: 9.8798 - acc: 0.3803  
Epoch 8/100  
426/426 [=====] - 0s 50us/step - loss: 9.8798 - acc: 0.3803  
Epoch 9/100  
426/426 [=====] - 0s 64us/step - loss: 9.8798 - acc: 0.3803  
Epoch 10/100  
426/426 [=====] - 0s 49us/step - loss: 9.8798 - acc: 0.3803  
Epoch 11/100  
426/426 [=====] - 0s 56us/step - loss: 9.8798 - acc: 0.3803  
Epoch 12/100  
426/426 [=====] - 0s 61us/step - loss: 9.8798 - acc: 0.3803  
Epoch 13/100  
426/426 [=====] - 0s 62us/step - loss: 9.8798 - acc: 0.3803  
Epoch 14/100  
426/426 [=====] - 0s 66us/step - loss: 9.8798 - acc: 0.3803  
Epoch 15/100  
426/426 [=====] - 0s 69us/step - loss: 9.8798 - acc: 0.3803  
Epoch 16/100  
426/426 [=====] - 0s 62us/step - loss: 9.8798 - acc: 0.3803  
Epoch 17/100

```
[61] print(my_first_nn.summary())
```

Model: "sequential\_5"

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 50)	1500
dense_10 (Dense)	(None, 1)	51

Total params: 1,551  
Trainable params: 1,551  
Non-trainable params: 0

```
[62] print(my_first_nn.evaluate(X_test, Y_test))
```

143/143 [=====] - 0s 388us/step  
[10.3681242028507772, 0.3496503496587575]

BY  
DUKKIPATI, SRI SAI NITHIN CHOWDARY  
CLASS ID: 4