

Video Link: <https://www.loom.com/share/3921c34d9d9f429aa0af1df8f7e94505>

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
dataset = pd.read_csv("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(100)

my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden input layer
my_first_nn.add(Dense(15, activation='relu')) #adding dense layer
my_first_nn.add(Dense(8, activation='relu')) ##adding dense layer
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, verbose=0, initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test, verbose=0))
```

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 | | |
| [0.5882100164890289, 0.6458333134651184] | | |

No additional
dense layer

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-------------------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 180 |
| dense_2 (Dense) | (None, 15) | 315 |
| dense_3 (Dense) | (None, 8) | 128 |
| dense_4 (Dense) | (None, 1) | 9 |
| Total params: 632 | | |
| Trainable params: 632 | | |
| Non-trainable params: 0 | | |

None

[0.6109174291292826, 0.6510416865348816]

After adding
dense layer

2. Change the data source to Breast Cancer dataset * available in the source folder and make required changes

```
# Importing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense

# Reading data
breastCancer = pd.read_csv('BreastCancer.csv')

# Converting non-numerical data into numerical
breastCancer["diagnosis"] = pd.Categorical(breastCancer["diagnosis"])
breastCancer["diagnosis"] = breastCancer["diagnosis"].cat.codes
cancerData = breastCancer.values

# Split the data set into training and test sets
x_train, x_test, y_train, y_test = train_test_split(cancerData[:, 2:32], cancerData[:, 1], test_size=0.2, random_state=45)

nnCancer = Sequential() #Creating model
nnCancer.add(Dense(20, input_dim=30, activation='relu')) # first hidden input dense layer
nnCancer.add(Dense(1, activation='sigmoid')) #Define the output neuron

# Fitting the neural network model on the training data set
nnCancer.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
nnCancerModel = nnCancer.fit(x_train, y_train, epochs=100, verbose=0, initial_epoch=0)

# Display the neural network identified
print('The Summary of the Neural Model is', nnCancer.summary())
print(nnCancer.evaluate(x_test, y_test, verbose=0))
```

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-------------------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 620 |
| dense_2 (Dense) | (None, 1) | 21 |
| Total params: 641 | | |
| Trainable params: 641 | | |
| Non-trainable params: 0 | | |

No Normalization

The Summary of the Neural Model is None

[0.15573108196258545, 0.9298245906829834]

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

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
# Importing Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense

# Reading data
breastCancer = pd.read_csv('BreastCancer.csv')

# Converting non-numerical data into numerical
breastCancer["diagnosis"] = pd.Categorical(breastCancer["diagnosis"])
breastCancer["diagnosis"] = breastCancer["diagnosis"].cat.codes
cancerData = breastCancer.values

# Split the data set into training and test sets
x_train, x_test, y_train, y_test = train_test_split(cancerData[:, 2:32], cancerData[:, 1], test_size=0.2, random_state=45)

# Data is normalize here
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

nnCancer = Sequential() #Creating model
nnCancer.add(Dense(20, input_dim=30, activation='relu')) # first hidden input dense layer
nnCancer.add(Dense(1, activation='sigmoid')) #Define the output neuron

# Fitting the neural network model on the training data set
nnCancer.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
nnCancerModel = nnCancer.fit(x_train, y_train, epochs=100, verbose=0, initial_epoch=0)

# Display the neural network identified
print('The Summary of the Neural Model is', nnCancer.summary())
print(nnCancer.evaluate(x_test, y_test, verbose=0))
```

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|---|--------------|---------|
| dense_1 (Dense) | (None, 20) | 620 |
| dense_2 (Dense) | (None, 1) | 21 |
| Total params: 641 | | |
| Trainable params: 641 | | |
| Non-trainable params: 0 | | |
| The Summary of the Neural Model is None | | |
| [0.11361751602472443, 0.9824561476707458] | | |

with Normalization

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

Diabetes with rmsprop

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-------------------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 180 |
| dense_2 (Dense) | (None, 15) | 315 |
| dense_3 (Dense) | (None, 8) | 128 |
| dense_4 (Dense) | (None, 1) | 9 |
| Total params: 632 | | |
| Trainable params: 632 | | |
| Non-trainable params: 0 | | |

With Adam

None

[0.610171357790629, 0.65625]

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-------------------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 180 |
| dense_2 (Dense) | (None, 15) | 315 |
| dense_3 (Dense) | (None, 8) | 128 |
| dense_4 (Dense) | (None, 1) | 9 |
| Total params: 632 | | |
| Trainable params: 632 | | |
| Non-trainable params: 0 | | |

with rmsprop

None

[0.5722076793511709, 0.7083333134651184]

Breast cancer Normalized data with rmsprop optimizer:

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-----------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 620 |
| dense_2 (Dense) | (None, 1) | 21 |

Total params: 641

Trainable params: 641

Non-trainable params: 0

With Adam

The Summary of the Neural Model is None

[0.12394108947595223, 0.9824561476707458]

Process finished with exit code 0

Model: "sequential_1"

| Layer (type) | Output Shape | Param # |
|-----------------|--------------|---------|
| dense_1 (Dense) | (None, 20) | 620 |
| dense_2 (Dense) | (None, 1) | 21 |

Total params: 641

Trainable params: 641

Non-trainable params: 0

With rmsprop

The Summary of the Neural Model is None

[0.211771922479353, 0.9824561476707458]
