# Assignment-6    DEEKSHITH ATHMAKUR 700743388

## Neural Networks and Deep learning

## GitHub link: https://github.com/DEEKSHITH-ATHMAKUR/ICP6

```
In [2]:    #read the data
           import pandas as pd
           data = pd.read_csv('diabetes.csv')
```

```
In [3]:    path_to_csv = 'diabetes.csv'-
```

```
In [5]:    import keras
           import pandas
           from keras.models import Sequential
           from keras.layers 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(path_to_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(155)
           my_first_nn = Sequential() # create model
           my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
           my_first_nn.add(Dense(4, 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)
           print(my_first_nn.summary())
           print(my_first_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
18/18 [==============================] - 1s 2ms/step - loss: 42.5964 - acc: 0.3385
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 30.6527 - acc: 0.3403
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 17.5874 - acc: 0.3611
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 10.7208 - acc: 0.5295
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 6.6725 - acc: 0.6111
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 4.0025 - acc: 0.5556
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 2.9301 - acc: 0.5000
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 2.6375 - acc: 0.5191
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 2.3826 - acc: 0.5139
Epoch 10/100
```

```
In [6]:   #read the data
          data = pd.read_csv('breastcancer.csv')
```

```
In [7]:   path_to_csv = 'sample_data/breastcancer.csv'
```

```
In [8]:   import keras
          import pandas as pd
          import numpy as np
          from keras.models import Sequential
          from keras.layers import Dense, Activation
          from sklearn.datasets import load_breast_cancer
          from sklearn.model_selection import train_test_split

          # load dataset
          cancer_data = load_breast_cancer()
          X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                              test_size=0.25, random_state=87)
          np.random.seed(155)
          my_nn = Sequential() # create model
          my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
          my_nn.add(Dense(1, activation='sigmoid')) # output layer
          my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
          my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                                   initial_epoch=0)
          print(my_nn.summary())
          print(my_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
14/14 [==============================] - 1s 2ms/step - loss: 110.2545 - acc: 0.3803
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 76.1194 - acc: 0.3803
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 41.7353 - acc: 0.3803
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 11.9064 - acc: 0.3873
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 7.9976 - acc: 0.4225
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 6.2010 - acc: 0.4014
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: 5.3807 - acc: 0.4930
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: 4.5558 - acc: 0.4765
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 3.5828 - acc: 0.5516
Epoch 10/100
```

```
In [9]:   #read the data
          data = pd.read_csv('breastcancer.csv')
```

```
In [10]:  path_to_csv = 'breastcancer.csv'
```

```
In [11]:  from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()
```

```
In [12]:  import keras
          import pandas as pd
          import numpy as np
          from keras.models import Sequential
          from keras.layers import Dense, Activation
          from sklearn.datasets import load_breast_cancer
          from sklearn.model_selection import train_test_split

          # load dataset
          cancer_data = load_breast_cancer()
          X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                              test_size=0.25, random_state=87)
          np.random.seed(155)
          my_nn = Sequential() # create model
          my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
          my_nn.add(Dense(1, activation='sigmoid')) # output layer
          my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
          my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                                   initial_epoch=0)
          print(my_nn.summary())
          print(my_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
14/14 [==============================] - 1s 2ms/step - loss: 40.8152 - acc: 0.3803
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 12.8738 - acc: 0.3826
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 2.3873 - acc: 0.6385
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 1.0803 - acc: 0.6174
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5198 - acc: 0.8052
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4618 - acc: 0.8545
Epoch 7/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4190 - acc: 0.8404
Epoch 8/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3760 - acc: 0.8662
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3560 - acc: 0.8685
```

```python
In [13]:    import keras
            from keras.datasets import mnist
            from keras.models import Sequential
            from keras.layers import Dense, Dropout
            import matplotlib.pyplot as plt

            # load MNIST dataset
            (x_train, y_train), (x_test, y_test) = mnist.load_data()

            # normalize pixel values to range [0, 1]
            x_train = x_train.astype('float32') / 255
            x_test = x_test.astype('float32') / 255

            # convert class labels to binary class matrices
            num_classes = 10
            y_train = keras.utils.to_categorical(y_train, num_classes)
            y_test = keras.utils.to_categorical(y_test, num_classes)

            # create a simple neural network model
            model = Sequential()
            model.add(Dense(512, activation='relu', input_shape=(784,)))
            model.add(Dropout(0.2))
            model.add(Dense(512, activation='relu'))
            model.add(Dropout(0.2))
            model.add(Dense(num_classes, activation='softmax'))

            model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

            # train the model and record the training history
            history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                                epochs=20, batch_size=128)

            # plot the training and validation accuracy and loss curves
            plt.figure(figsize=(10, 5))
            plt.subplot(1, 2, 1)
            plt.plot(history.history['accuracy'])
            plt.plot(history.history['val_accuracy'])
            plt.title('Model Accuracy')
            plt.ylabel('Accuracy')
            plt.xlabel('Epoch')
            plt.legend(['Train', 'Validation'], loc='lower right')

            plt.subplot(1, 2, 2)
            plt.plot(history.history['loss'])
            plt.plot(history.history['val_loss'])
            plt.title('Model Loss')
            plt.ylabel('Loss')
            plt.xlabel('Epoch')
            plt.legend(['Train', 'Validation'], loc='upper right')

            plt.show()
```

0.9829



Model Accuracy — Train, Validation

Model Loss — Train, Validation