In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as Tnr
#Read the training data
train_data = pd.read_csv('breast-cancer.csv')
```

In [2]:
```
#Print the data
print(train_data.head())
```

```
   diagnosis  radius_mean  texture_mean  smoothness_mean  compactness_mean  \
0          1        17.99         10.38          0.11840           0.27760
1          1        20.57         17.77          0.08474           0.07864
2          1        19.69         21.25          0.10960           0.15990
3          1        11.42         20.38          0.14250           0.28390
4          1        20.29         14.34          0.10030           0.13280

   concavity_mean  concave points_mean
0          0.3001              0.14710
1          0.0869              0.07017
2          0.1974              0.12790
3          0.2414              0.10520
4          0.1980              0.10430
```

In [3]:
```
#Print the dimesnion of the data
train_data.shape
```

Out[3]:
```
(569, 7)
```

In [4]:
```
#separating X train and Y train
X_train = train_data[['radius_mean','texture_mean','smoothness_mean','compactness_mean','concavity_mean','concav
Y_train = train_data[['diagnosis']]
```

In [5]:
```
# importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
```

In [6]:
```
# splitting the data
X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size = 0.3, random_state = 0)
```

In [7]:
```
#print the shape of train and test data after splting
print (X_train.shape)
print (Y_train.shape)
print (X_test.shape)
print (Y_test.shape)
```

```
(398, 6)
(398, 1)
(171, 6)
(171, 1)
```

In [8]:
```
from keras.layers import Dense
from keras.models import Sequential
```

In [9]:
```
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='sigmoid'))
model.add(Dense(32, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
```

In [10]:
```
model.compile(optimizer=Tnr.keras.optimizers.Adam(learning_rate=0.001),
                loss=Tnr.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=200, batch_size=32, verbose=1)

# Evaluate the model on the test set
test_loss, test_acc1 = model.evaluate(X_test, Y_test, verbose=0)
```

```
Epoch 1/200
13/13 [==============================] - 1s 2ms/step - loss: 1.2747 - accuracy: 0.3744
Epoch 2/200
13/13 [==============================] - 0s 1ms/step - loss: 0.9064 - accuracy: 0.3744
Epoch 3/200
13/13 [==============================] - 0s 1ms/step - loss: 0.7188 - accuracy: 0.4171
Epoch 4/200
13/13 [==============================] - 0s 1ms/step - loss: 0.6703 - accuracy: 0.6256
Epoch 5/200
13/13 [==============================] - 0s 1ms/step - loss: 0.6626 - accuracy: 0.6256
Epoch 6/200
13/13 [==============================] - 0s 1ms/step - loss: 0.6569 - accuracy: 0.6256
Epoch 7/200
13/13 [==============================] - 0s 3ms/step - loss: 0.6524 - accuracy: 0.6256
Epoch 8/200
13/13 [==============================] - 0s 2ms/step - loss: 0.6460 - accuracy: 0.6256
Epoch 9/200
13/13 [==============================] - 0s 1ms/step - loss: 0.6395 - accuracy: 0.6256
Epoch 10/200
```

In [11]:
```

# Build the model with ReLU activation function
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='relu'))
```

In [12]:
```
# Compile the model
model.compile(optimizer=Tnr.keras.optimizers.Adam(learning_rate=0.001),
                loss=Tnr.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

In [13]:
```
# Train the model
model.fit(X_train, Y_train, epochs=200, batch_size=32, verbose=1)
```

```
Epoch 1/200
13/13 [==============================] - 1s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 2/200
13/13 [==============================] - 0s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 3/200
13/13 [==============================] - 0s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 4/200
13/13 [==============================] - 0s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 5/200
13/13 [==============================] - 0s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 6/200
13/13 [==============================] - 0s 2ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 7/200
13/13 [==============================] - 0s 1ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 8/200
13/13 [==============================] - 0s 1ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 9/200
13/13 [==============================] - 0s 1ms/step - loss: 5.7747 - accuracy: 0.6256
Epoch 10/200
```

In [14]:
```
# Evaluate the model on the test set
test_loss, test_acc2 = model.evaluate(X_test, Y_test, verbose=0)
print("Test accuracy with sigmoid activation:", test_acc1)
print("Test accuracy with ReLU activation:", test_acc2)
```

```
Test accuracy with sigmoid activation: 0.9298245906829834
Test accuracy with ReLU activation: 0.6315789222717285
```