# Heart Disease prediction using ANN

## Code to upload files in colab

```
from google.colab import files
```

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

```
uploaded = files.upload()
```

Choose Files heart.csv

- **heart.csv**(application/vnd.ms-excel) - 11328 bytes, last modified: 8/15/2021 - 100% done

```
Saving heart.csv to heart.csv
```

```
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

```
User uploaded file "heart.csv" with length 11328 bytes
```

## Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

## Importing the dataset

```
dataset = pd.read_csv('/content/heart.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

```
print(X)
```

```
[[63.  1.  3. ...  0.  0.  1.]
 [37.  1.  2. ...  0.  0.  2.]
 [41.  0.  1. ...  2.  0.  2.]
 ...
 [68.  1.  0. ...  1.  2.  3.]
 [57.  1.  0. ...  1.  1.  3.]
 [57.  0.  1. ...  1.  1.  2.]]
```

```
print(y)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0]
```

## Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

## Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

## Importing the Keras libraries and packages

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

## Initialising the ANN

```
classifier = Sequential()
```

## Adding the input layer and the first hidden layer

```
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation='relu', input_dim=
```

## Adding the second hidden layer

```
classifier.add(Dense(units=6, kernel_initializer='uniform', activation='relu'))
```

## Adding the second hidden layer

```
classifier.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
```

## Compiling the ANN (Configuring the learning process)

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

## Fitting the ANN to the Training set

```
classifier.fit(X_train, y_train, batch_size = 10, epochs=50)
```

```
Epoch 1/50
25/25 [==============================] - 1s 1ms/step - loss: 0.6930 - accuracy: 0.52
Epoch 2/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6919 - accuracy: 0.54
Epoch 3/50
25/25 [==============================] - 0s 1ms/step - loss: 0.6876 - accuracy: 0.54
Epoch 4/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6748 - accuracy: 0.75
Epoch 5/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6482 - accuracy: 0.80
Epoch 6/50
25/25 [==============================] - 0s 2ms/step - loss: 0.6082 - accuracy: 0.81
Epoch 7/50
25/25 [==============================] - 0s 1ms/step - loss: 0.5593 - accuracy: 0.83
Epoch 8/50
25/25 [==============================] - 0s 1ms/step - loss: 0.5111 - accuracy: 0.83
Epoch 9/50
25/25 [==============================] - 0s 2ms/step - loss: 0.4715 - accuracy: 0.83
Epoch 10/50
25/25 [==============================] - 0s 2ms/step - loss: 0.4393 - accuracy: 0.83
Epoch 11/50
25/25 [==============================] - 0s 1ms/step - loss: 0.4113 - accuracy: 0.84
```

```
Epoch 12/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3922 - accuracy: 0.84
Epoch 13/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3802 - accuracy: 0.84
Epoch 14/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3712 - accuracy: 0.85
Epoch 15/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3641 - accuracy: 0.85
Epoch 16/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3568 - accuracy: 0.85
Epoch 17/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3526 - accuracy: 0.85
Epoch 18/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3494 - accuracy: 0.84
Epoch 19/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3458 - accuracy: 0.85
Epoch 20/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3445 - accuracy: 0.85
Epoch 21/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3422 - accuracy: 0.85
Epoch 22/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3399 - accuracy: 0.85
Epoch 23/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3390 - accuracy: 0.85
Epoch 24/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3367 - accuracy: 0.85
Epoch 25/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3356 - accuracy: 0.85
Epoch 26/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3347 - accuracy: 0.85
Epoch 27/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3334 - accuracy: 0.85
Epoch 28/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3330 - accuracy: 0.86
Epoch 29/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3318 - accuracy: 0.86
Epoch 30/50
```

## Predicting the Test set results

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

## Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
```

```
[[22  5]
```

```
[ 4 30]]
```

✓ 0s completed at 13:46