```
from google.colab import files


uploaded = files.upload()
```

- **heart.csv**(application/vnd.ms-excel) - 11328 bytes, last modified: 10/26/2021 - 100% done

```
Saving heart.csv to heart (1).csv


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

    User uploaded file "heart.csv" with 11328 bytes


import numpy as np
import pandas as pd


dataset = pd.read_csv('/content/heart.csv')
dataset.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

```
X = dataset.iloc[:,:-1].values
Y = dataset.iloc[:,-1].values


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)


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


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

```
classifier = Sequential()

classifier.add(Dense(units=6,kernel_initializer='uniform',activation='relu',input_dim=13))
classifier.add(Dense(units=6,kernel_initializer='uniform',activation='relu'))
classifier.add(Dense(units=1,kernel_initializer='uniform',activation='sigmoid'))


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

classifier.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 6)                 84
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 7
=================================================================
Total params: 133
Trainable params: 133
Non-trainable params: 0
_________________________________________________________________
```

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

```
Epoch 22/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3461 - accuracy: 0.843
Epoch 23/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3454 - accuracy: 0.843
Epoch 24/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3447 - accuracy: 0.838
Epoch 25/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3439 - accuracy: 0.838
Epoch 26/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3431 - accuracy: 0.843
Epoch 27/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3423 - accuracy: 0.838
Epoch 28/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3419 - accuracy: 0.843
Epoch 29/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3404 - accuracy: 0.843
Epoch 30/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3393 - accuracy: 0.843
Epoch 31/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3386 - accuracy: 0.843
Epoch 32/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3377 - accuracy: 0.843
Epoch 33/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3372 - accuracy: 0.847
Epoch 34/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3368 - accuracy: 0.847
Epoch 35/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3357 - accuracy: 0.847
```

```
Epoch 36/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3353 - accuracy: 0.851
Epoch 37/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3350 - accuracy: 0.851
Epoch 38/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3346 - accuracy: 0.847
Epoch 39/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3340 - accuracy: 0.855
Epoch 40/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3332 - accuracy: 0.859
Epoch 41/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3325 - accuracy: 0.863
Epoch 42/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3320 - accuracy: 0.859
Epoch 43/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3315 - accuracy: 0.859
Epoch 44/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3309 - accuracy: 0.855
Epoch 45/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3295 - accuracy: 0.859
Epoch 46/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3294 - accuracy: 0.863
Epoch 47/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3296 - accuracy: 0.859
Epoch 48/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3282 - accuracy: 0.859
Epoch 49/50
25/25 [==============================] - 0s 2ms/step - loss: 0.3273 - accuracy: 0.863
Epoch 50/50
25/25 [==============================] - 0s 1ms/step - loss: 0.3269 - accuracy: 0.863
<keras.callbacks.History at 0x7f4f4d585a10>
```

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

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,Y_pred)


print(cm)
```

```
[[22  5]
 [ 4 30]]
```

```
tn,fp,fn,tp = cm.ravel()
(tn,fp,fn,tp)
```

```
(22, 5, 4, 30)
```

```
acc = (((tp+tn)/(tn+fp+fn+tp))*100)
print(acc)
```

```
85.24590163934425
```

✓ 0s completed at 4:58 PM