Sonal Shukla (CS23176)

Lab-6 Assignment-1 : To design and implement an Artificial Neural Network for classification problem and evaluate its learning performance.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
path ="/content/drive/MyDrive/ML_DATASETS/Heart_Failure_Prediction_dataset - Heart_Failure_Prediction_dataset.csv"
df=pd.read_csv(path)
df.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDis
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	

```
df.shape
```

```
(918, 12)
```

```
df.columns
```

```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

```
df.isnull().any().any()
```

```
np.False_
```

```
X = df.drop(columns=['HeartDisease'])
y = df['HeartDisease']
```

```
X.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up

```
X = pd.get_dummies(X,dtype = int)
```

```
X.head()
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Sex_F	Sex_M	ChestPainType_ASY	ChestPainType_ATA	ChestPainType_
0	40	140	289	0	172	0.0	0	1	0	1	
1	49	160	180	0	156	1.0	1	0	0	0	
2	37	130	283	0	98	0.0	0	1	0	1	
3	48	138	214	0	108	1.5	1	0	1	0	
4	54	150	195	0	122	0.0	0	1	0	0	

```
X = df.iloc[:,3:-1]
X = pd.get_dummies(X, dtype=int)
X
```

	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	RestingECG_LVH	RestingECG_Normal	RestingECG_ST	ExerciseAngina_N	
0	140	289	0	172	0.0	0	1	0	1	
1	160	180	0	156	1.0	0	1	0	1	
2	130	283	0	98	0.0	0	0	1	1	
3	138	214	0	108	1.5	0	1	0	0	
4	150	195	0	122	0.0	0	1	0	1	
...	...	...	...	...	...	...	...	...	...	
913	110	264	0	132	1.2	0	1	0	1	
914	144	193	1	141	3.4	0	1	0	1	
915	130	131	0	115	1.2	0	1	0	0	
916	130	236	0	174	0.0	1	0	0	1	
917	138	175	0	173	0.0	0	1	0	1	

918 rows × 13 columns

```
y=df.iloc[:,-1]
print(y)
```

```
0      0
1      1
2      0
3      1
4      0
      ..
913    1
914    1
915    1
916    1
917    0
Name: HeartDisease, Length: 918, dtype: int64
```

```
X.shape
```

```
(918, 13)
```

```
X.columns
```

```
Index(['RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak',
       'RestingECG_LVH', 'RestingECG_Normal', 'RestingECG_ST',
       'ExerciseAngina_N', 'ExerciseAngina_Y', 'ST_Slope_Down',
       'ST_Slope_Flat', 'ST_Slope_Up'],
      dtype='object')
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7,test_size=0.3,random_state=42)
X_train.shape,X_test.shape
```

```
((642, 13), (276, 13))
```

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

```
X_train
```

```
array([[-1.77231378,  0.2949892 , -0.52568236, ..., -0.25431598,
        -1.02524056,  1.15890711],
       [-1.23486848, -1.90539005,  1.90228942, ...,  3.93211628,
        -1.02524056, -0.86288193],
       [ 1.45235803, -1.90539005,  1.90228942, ..., -0.25431598,
         0.97538084, -0.86288193],
       ...,
       [-0.15997788,  0.47365248, -0.52568236, ..., -0.25431598,
        -1.02524056,  1.15890711],
       [ 1.02240179, -1.90539005, -0.52568236, ..., -0.25431598,
        -1.02524056,  1.15890711],
       [ 0.91491273,  1.78071537, -0.52568236, ..., -0.25431598,
         0.97538084, -0.86288193]])
```

```
y_train
```

	HeartDisease
712	1
477	1
409	1
448	1
838	1
...	...
106	0
270	0
860	1
435	0
102	1

642 rows × 1 columns

dtype: int64

```
import tensorflow as tf
```

```
ann = tf.keras.models.Sequential()
```

```
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
```

```
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
```

```
ann.add(tf.keras.layers.Dense(units=1,activation="sigmoid"))
```

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

```
ann.fit(X_train,y_train,batch_size=32,epochs=100)
```

```
Epoch 1/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.5646 - loss: 0.6722
Epoch 2/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6750 - loss: 0.6226
Epoch 3/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7570 - loss: 0.5915
Epoch 4/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7546 - loss: 0.5442
Epoch 5/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7730 - loss: 0.5212
Epoch 6/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8039 - loss: 0.4911
Epoch 7/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8170 - loss: 0.4743
Epoch 8/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8100 - loss: 0.4640
Epoch 9/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7984 - loss: 0.4758
Epoch 10/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8487 - loss: 0.3878
Epoch 11/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8333 - loss: 0.4191
Epoch 12/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8456 - loss: 0.3829
```

```
Epoch 13/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8404 - loss: 0.3809
Epoch 14/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8237 - loss: 0.4226
Epoch 15/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8409 - loss: 0.3780
Epoch 16/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8525 - loss: 0.3695
Epoch 17/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8249 - loss: 0.4054
Epoch 18/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8297 - loss: 0.4044
Epoch 19/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8362 - loss: 0.3722
Epoch 20/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8348 - loss: 0.3793
Epoch 21/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8287 - loss: 0.3912
Epoch 22/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8595 - loss: 0.3621
Epoch 23/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8438 - loss: 0.3586
Epoch 24/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8363 - loss: 0.3860
Epoch 25/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8348 - loss: 0.3950
Epoch 26/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8496 - loss: 0.3686
Epoch 27/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8633 - loss: 0.3319
Epoch 28/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8454 - loss: 0.3641
Epoch 29/100
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8475 - loss: 0.3512
```

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

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
array([[False],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [False],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [False],
       [ True],
       [False],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
```

```
       [False],
       [False],
       [ True],
       [ True],
       [False],
       [False],
       [ True]
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

```
print('Confusion Matrix')
cm=confusion_matrix(y_pred,y_test)
print(confusion_matrix(y_pred,y_test))
```

```
Confusion Matrix
[[ 90  21]
 [ 22 143]]
```

```
print('Accuracy Score')
print(accuracy_score(y_pred,y_test))
```

```
Accuracy Score
0.8442028985507246
```