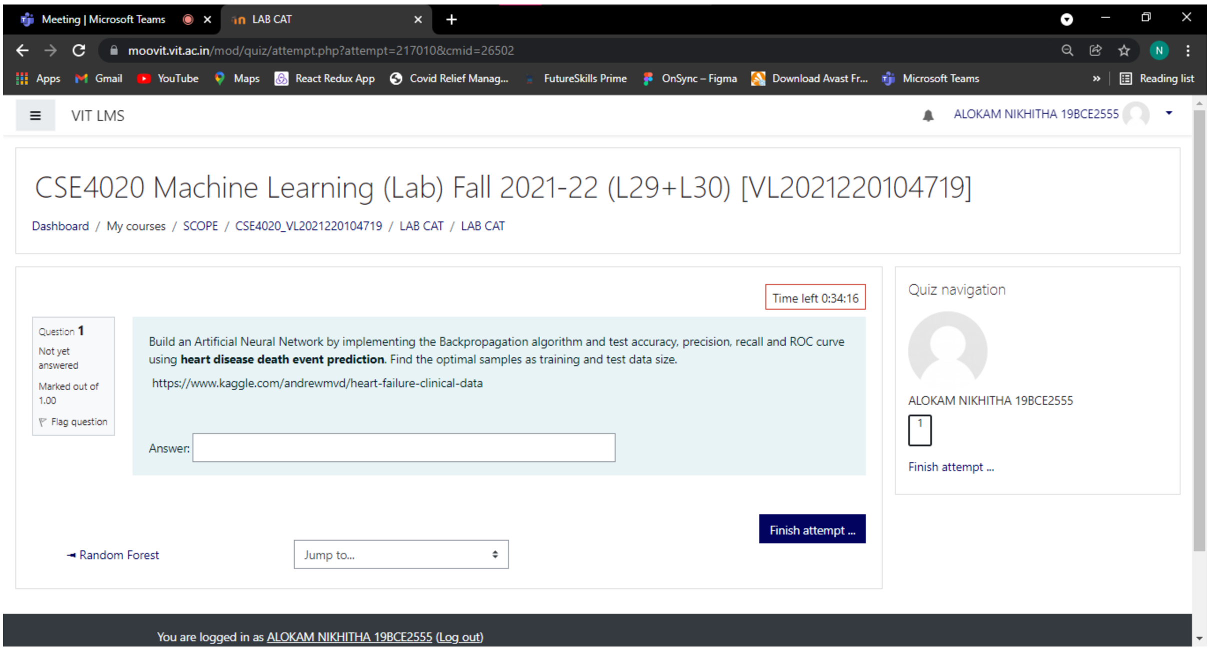
CSE 4020 - MACHINE LEARNING

Lab 29+30

Lab CAT

Submitted by: Alokam Nikhitha(19BCE2555)

Question:



CSE4020 Machine Learning (Lab) Fall 2021-22 (L29+L30) [VL2021220104719]

Dashboard / My courses / SCOPE / CSE4020_VL2021220104719 / LAB CAT / LAB CAT

Time left 0:34:16

Question 1
Not yet answered
Marked out of 1.00
Flag question

Build an Artificial Neural Network by implementing the Backpropagation algorithm and test accuracy, precision, recall and ROC curve using **heart disease death event prediction**. Find the optimal samples as training and test data size.

https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

Answer:

You are logged in as ALOKAM NIKHITHA 19BCE2555 (Log out)

Build an Artificial Neural Network by implementing the Backpropagation algorithm and test accuracy, precision, recall and ROC curve using **heart disease death event prediction**. Find the optimal samples as training and test data size.

https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

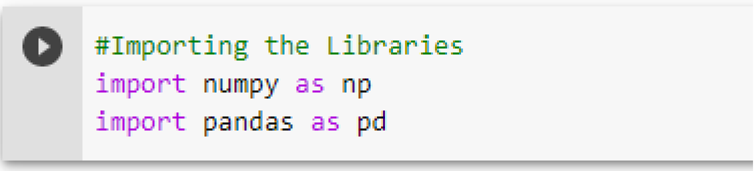
Dataset Used:

https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

Procedure:

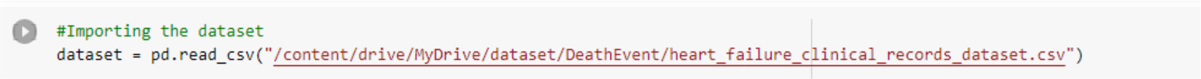
- **Firstly we are importing the Libraries**
- **We are importing the dataset using pandas**
- **Here we displayed the first 10 rows of the dataset.**
- **We identified Dependent and Independent variables in the dataset.**
- **Splitting the dataset in to Testing and Training sets.**
- **Feature Scalling the data**
- **Later, We initialized th ANN**
- **We displayed the Accuracy, Precision**
- **We plotted the ROC graph using Matplot Library**

Code Snippets and Explanation:



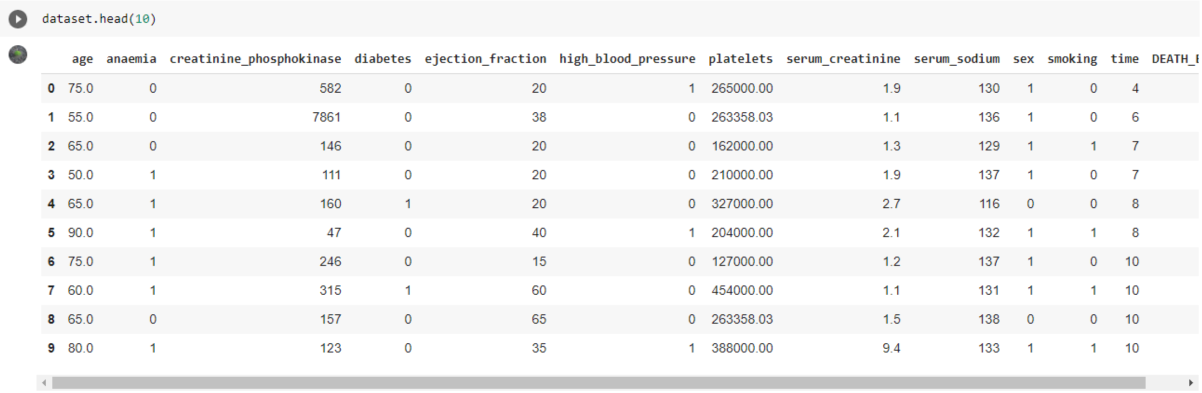
```
#Importing the Libraries
import numpy as np
import pandas as pd
```

Here we are importing the libraries.



```
#Importing the dataset
dataset = pd.read_csv("/content/drive/MyDrive/dataset/DeathEvent/heart_failure_clinical_records_dataset.csv")
```

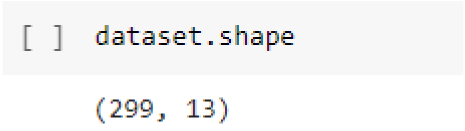
Here we are importing the dataset using pandas.



```
dataset.head(10)
```

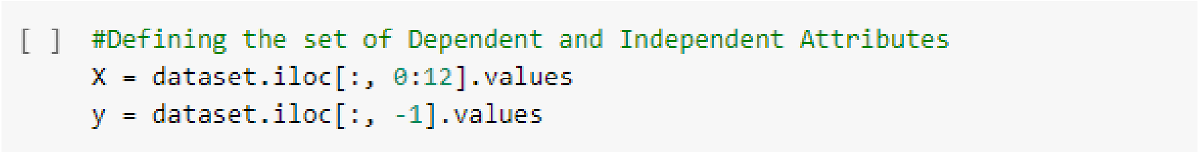
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_E
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	
5	90.0	1	47	0	40	1	204000.00	2.1	132	1	1	8	
6	75.0	1	246	0	15	0	127000.00	1.2	137	1	0	10	
7	60.0	1	315	1	60	0	454000.00	1.1	131	1	1	10	
8	65.0	0	157	0	65	0	263358.03	1.5	138	0	0	10	
9	80.0	1	123	0	35	1	388000.00	9.4	133	1	1	10	

Here we displayed the first 10 rows of the dataset.



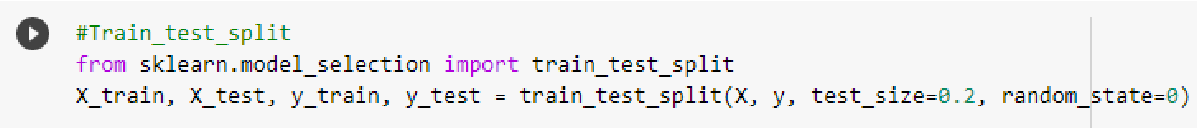
```
[ ]  dataset.shape

     (299, 13)
```

Displaying the size of Dataset



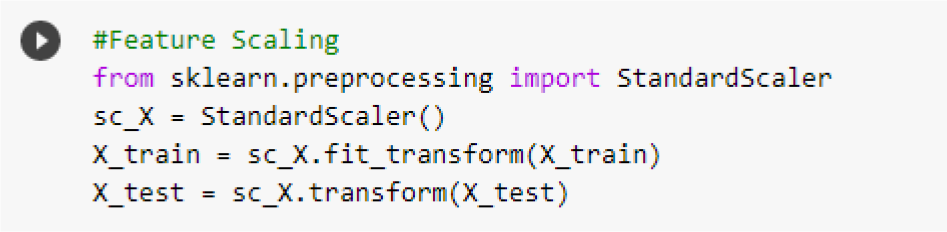
```
[ ]  #Defining the set of Dependent and Independent Attributes
     X = dataset.iloc[:, 0:12].values
     y = dataset.iloc[:, -1].values
```

Identifying Dependent and Independent variables in the dataset.



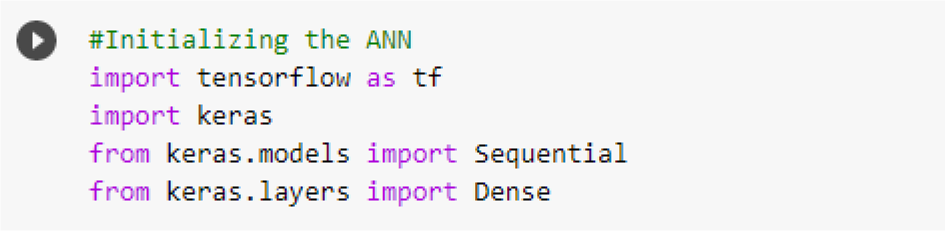
```
#Train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Splitting the Testing and Training sets.

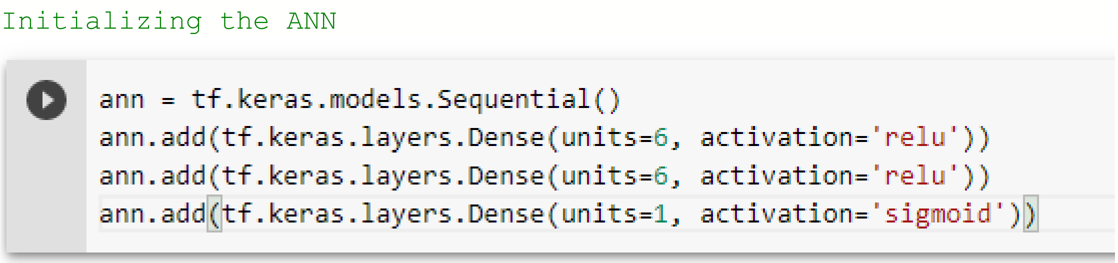


```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Feature Scalling the data

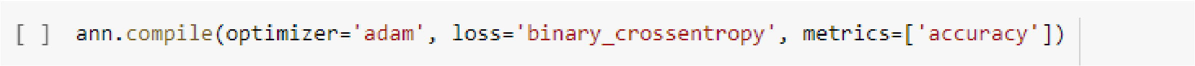


```
#Initializing the ANN
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense
```

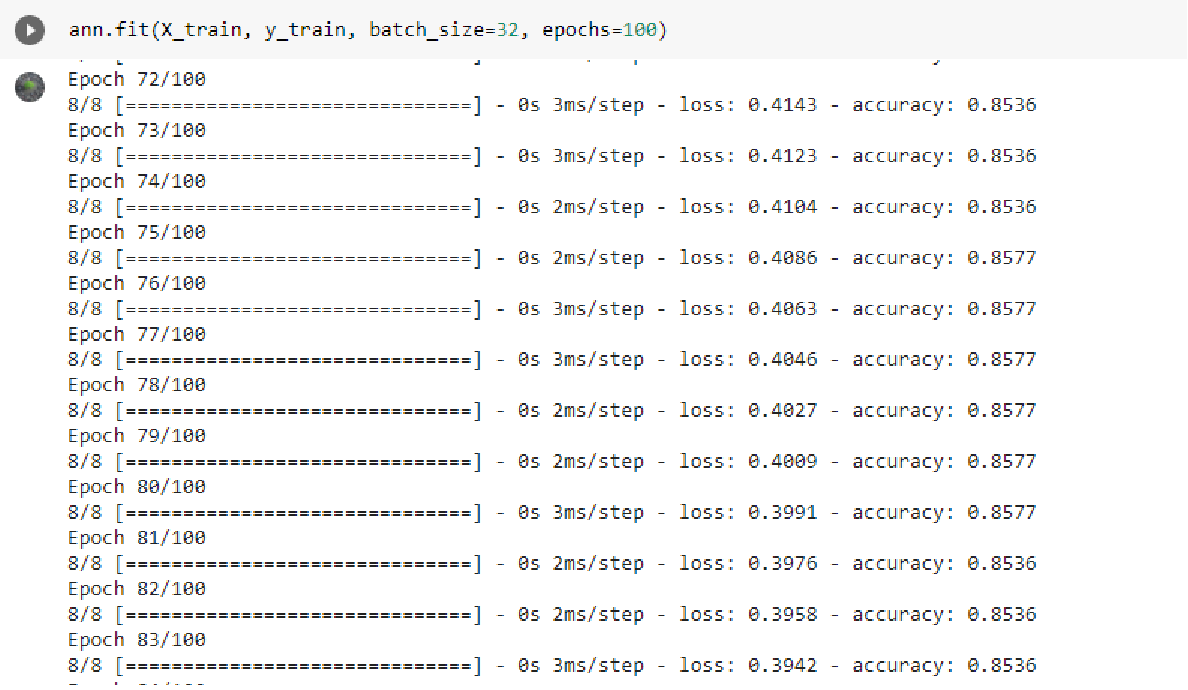


Initializing the ANN

```
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'))
```

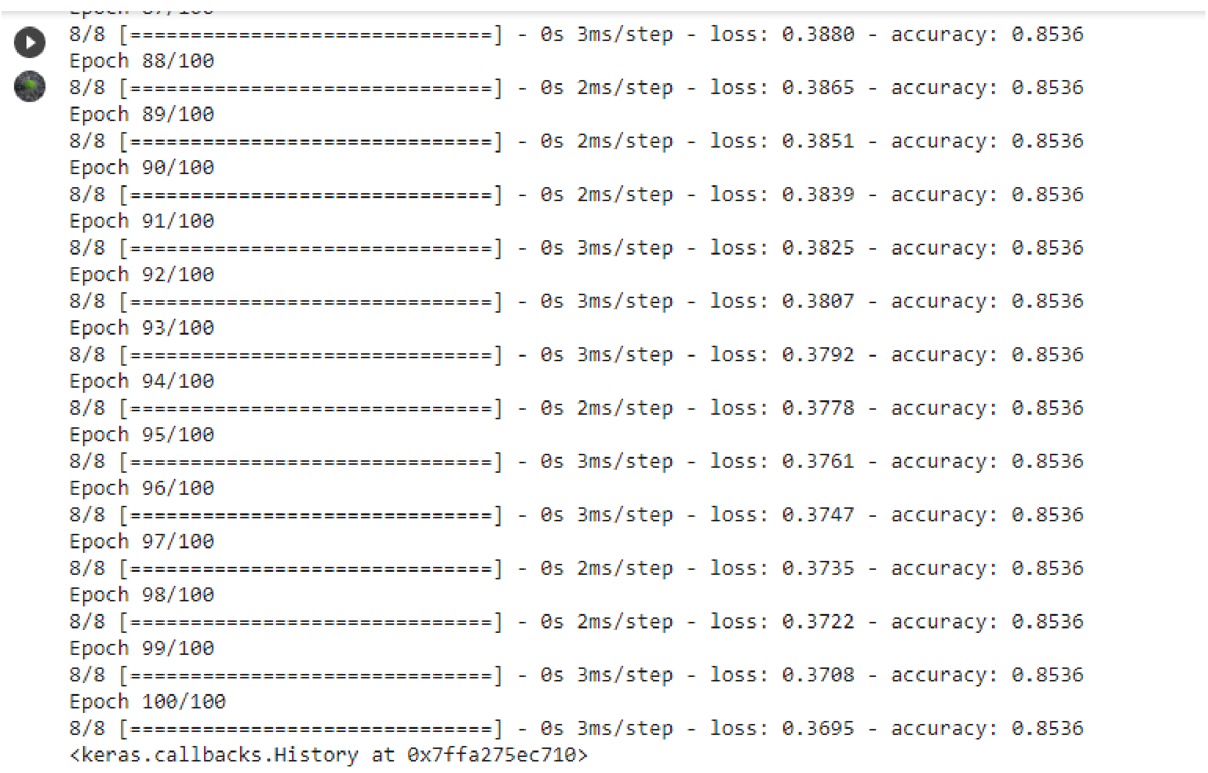


```
ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

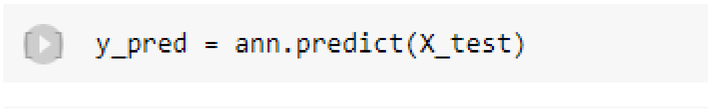


```
ann.fit(X_train, y_train, batch_size=32, epochs=100)
```

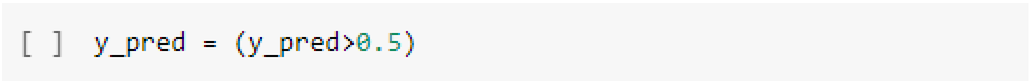
```
Epoch 72/100
8/8 [==============================] - 0s 3ms/step - loss: 0.4143 - accuracy: 0.8536
Epoch 73/100
8/8 [==============================] - 0s 3ms/step - loss: 0.4123 - accuracy: 0.8536
Epoch 74/100
8/8 [==============================] - 0s 2ms/step - loss: 0.4104 - accuracy: 0.8536
Epoch 75/100
8/8 [==============================] - 0s 2ms/step - loss: 0.4086 - accuracy: 0.8577
Epoch 76/100
8/8 [==============================] - 0s 3ms/step - loss: 0.4063 - accuracy: 0.8577
Epoch 77/100
8/8 [==============================] - 0s 3ms/step - loss: 0.4046 - accuracy: 0.8577
Epoch 78/100
8/8 [==============================] - 0s 2ms/step - loss: 0.4027 - accuracy: 0.8577
Epoch 79/100
8/8 [==============================] - 0s 2ms/step - loss: 0.4009 - accuracy: 0.8577
Epoch 80/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3991 - accuracy: 0.8577
Epoch 81/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3976 - accuracy: 0.8536
Epoch 82/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3958 - accuracy: 0.8536
Epoch 83/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3942 - accuracy: 0.8536
```



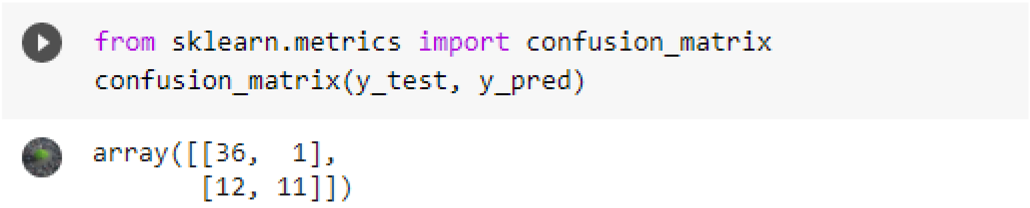
```
Epoch 87/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3880 - accuracy: 0.8536
Epoch 88/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3865 - accuracy: 0.8536
Epoch 89/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3851 - accuracy: 0.8536
Epoch 90/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3839 - accuracy: 0.8536
Epoch 91/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3825 - accuracy: 0.8536
Epoch 92/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3807 - accuracy: 0.8536
Epoch 93/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3792 - accuracy: 0.8536
Epoch 94/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3778 - accuracy: 0.8536
Epoch 95/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3761 - accuracy: 0.8536
Epoch 96/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3747 - accuracy: 0.8536
Epoch 97/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3735 - accuracy: 0.8536
Epoch 98/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3722 - accuracy: 0.8536
Epoch 99/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3708 - accuracy: 0.8536
Epoch 100/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3695 - accuracy: 0.8536
<keras.callbacks.History at 0x7ffa275ec710>
```



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



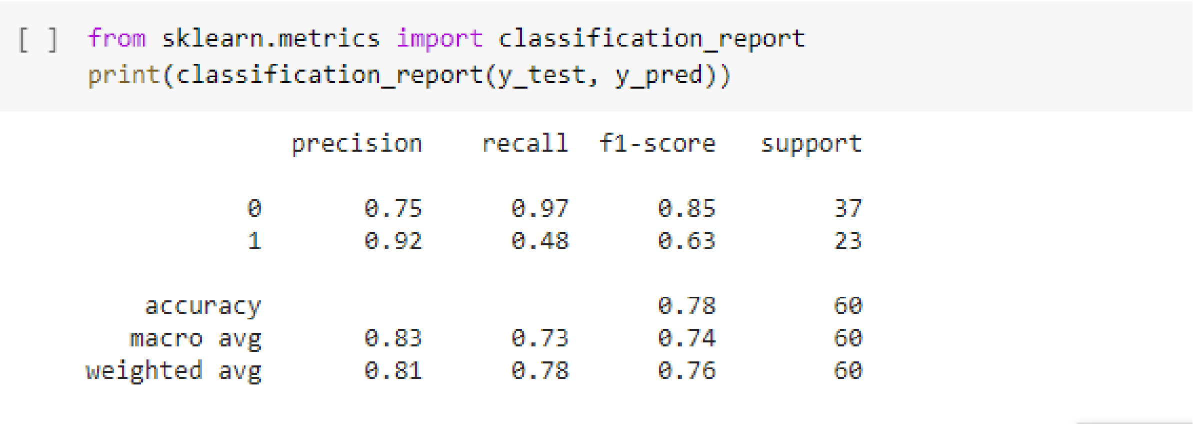
```
[ ]  y_pred = (y_pred>0.5)
```



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

```
array([[36,  1],
       [12, 11]])
```

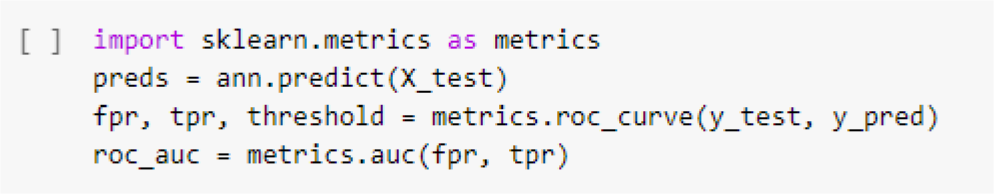
Confusion Matrix



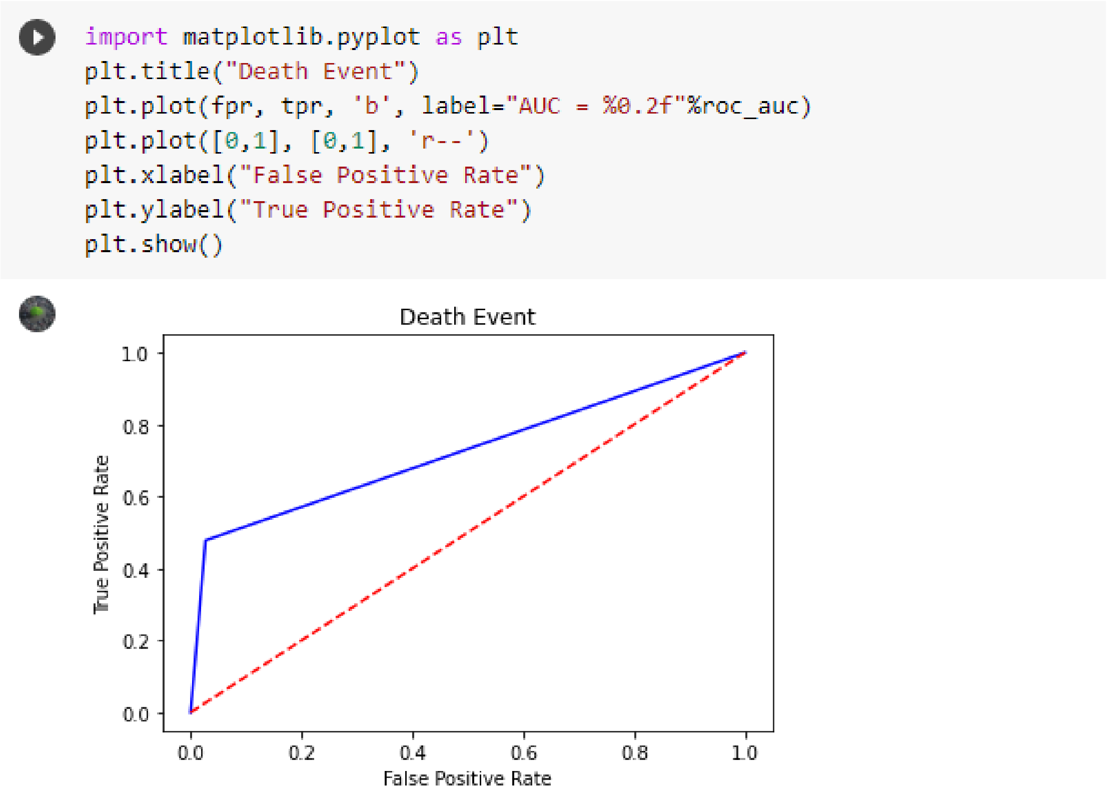
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.97      0.85        37
           1       0.92      0.48      0.63        23

    accuracy                           0.78        60
   macro avg       0.83      0.73      0.74        60
weighted avg       0.81      0.78      0.76        60
```

Precision of the Given Dataset



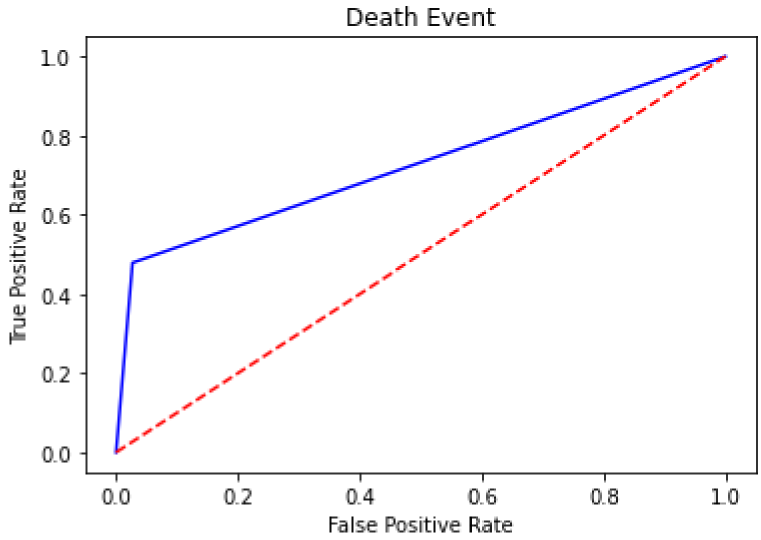
```
import sklearn.metrics as metrics
preds = ann.predict(X_test)
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred)
roc_auc = metrics.auc(fpr, tpr)
```



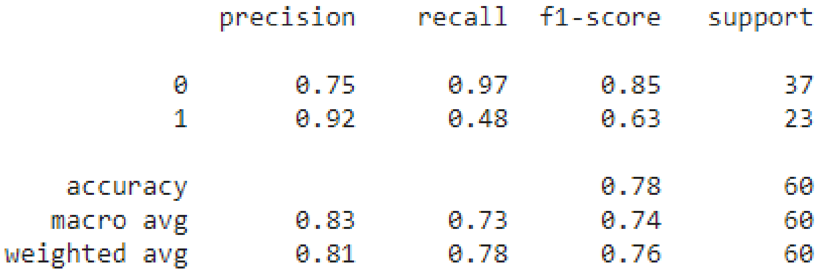
```
import matplotlib.pyplot as plt
plt.title("Death Event")
plt.plot(fpr, tpr, 'b', label="AUC = %0.2f"%roc_auc)
plt.plot([0,1], [0,1], 'r--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

ROC Graph

Results and Conclusion:

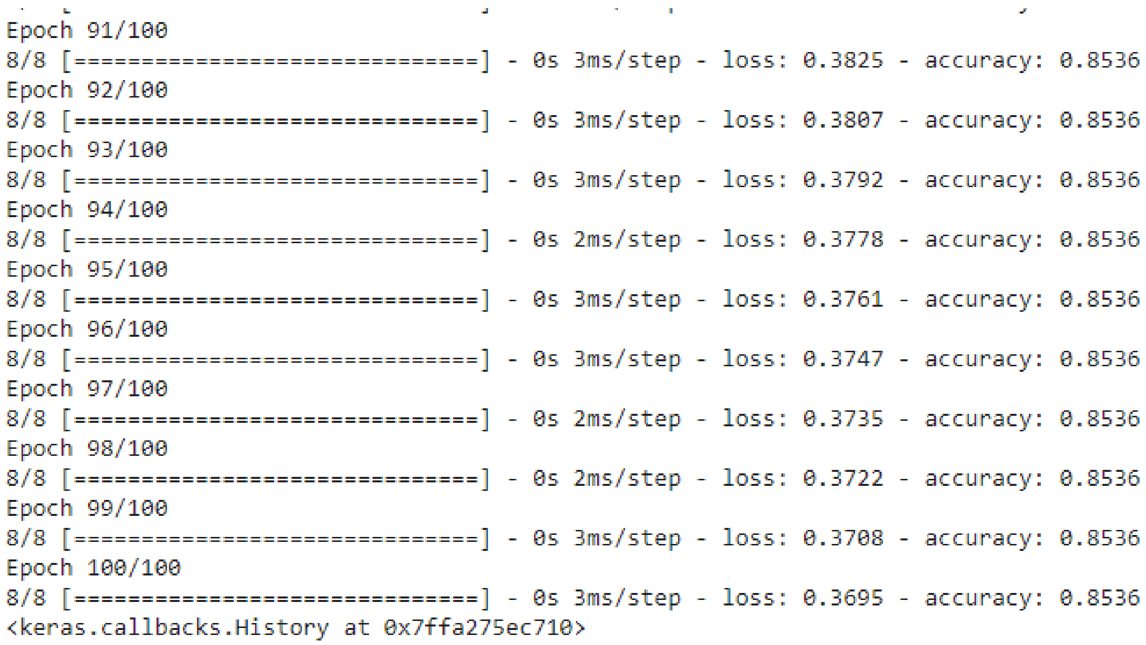


ROC Graph



```
              precision    recall  f1-score   support

           0       0.75      0.97      0.85        37
           1       0.92      0.48      0.63        23

    accuracy                           0.78        60
   macro avg       0.83      0.73      0.74        60
weighted avg       0.81      0.78      0.76        60
```

Precision



```
Epoch 91/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3825 - accuracy: 0.8536
Epoch 92/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3807 - accuracy: 0.8536
Epoch 93/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3792 - accuracy: 0.8536
Epoch 94/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3778 - accuracy: 0.8536
Epoch 95/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3761 - accuracy: 0.8536
Epoch 96/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3747 - accuracy: 0.8536
Epoch 97/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3735 - accuracy: 0.8536
Epoch 98/100
8/8 [==============================] - 0s 2ms/step - loss: 0.3722 - accuracy: 0.8536
Epoch 99/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3708 - accuracy: 0.8536
Epoch 100/100
8/8 [==============================] - 0s 3ms/step - loss: 0.3695 - accuracy: 0.8536
<keras.callbacks.History at 0x7ffa275ec710>
```

Accuracy is 85.36%