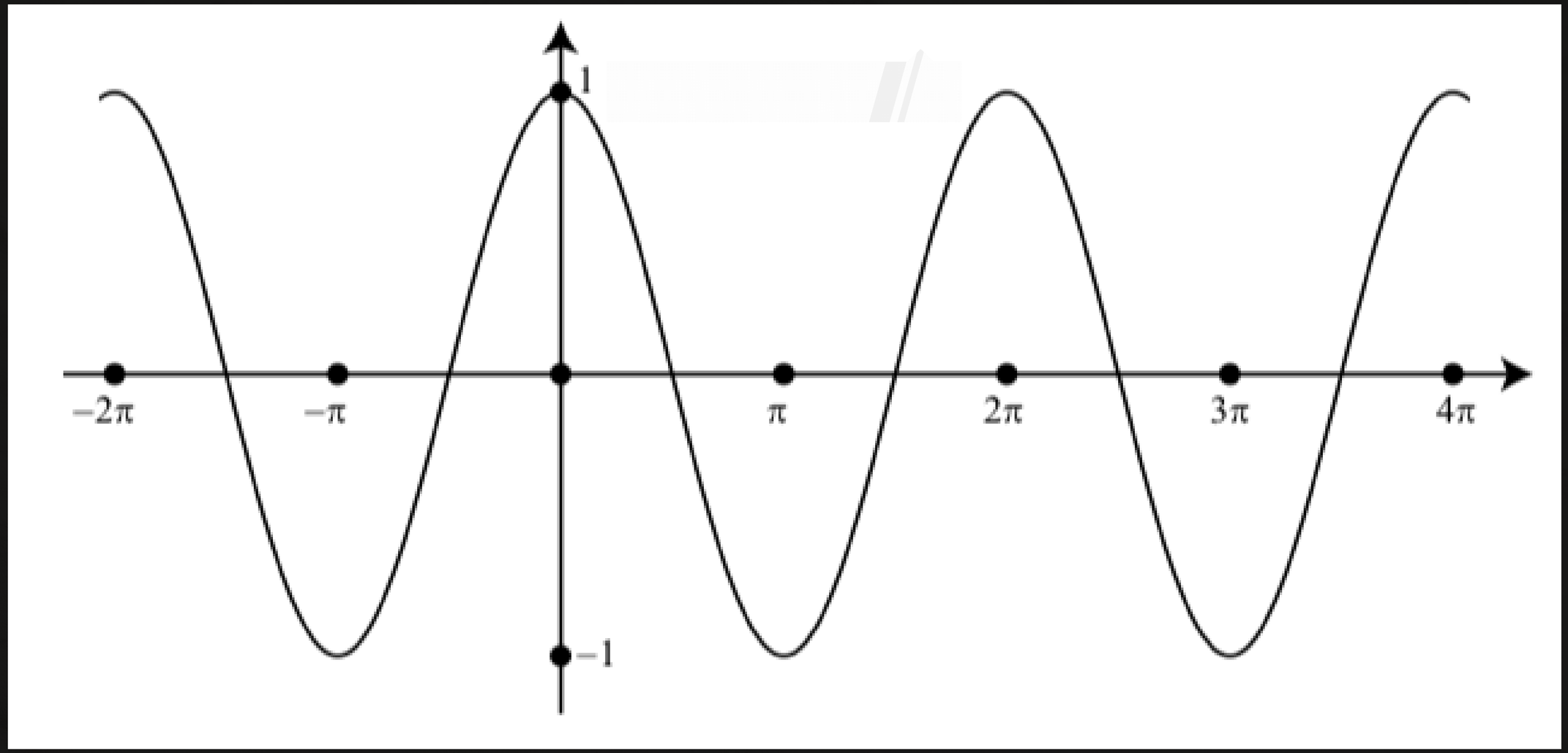
ARTIFICIAL NEURAL NETWORKS

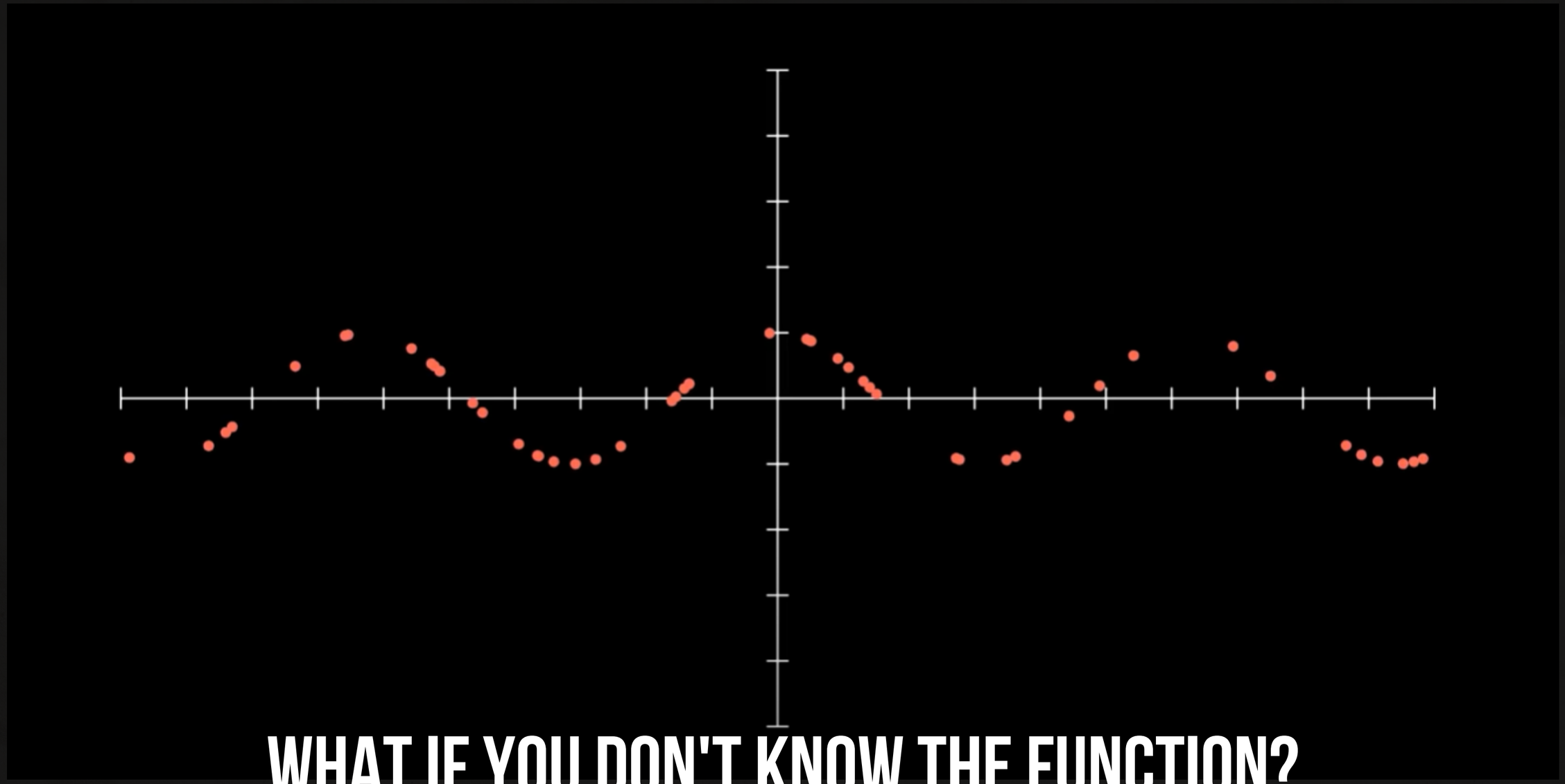
AND PREDICTING HEART ATTACKS;

COMPUTATIONAL BIOLOGY PROJECT;
PHASE 1 AND 2

Abhay Bhat
PES1UG19BT002

WHAT IS A FUNCTION?


1
−1
−2π
−π
π
2π
3π
4π



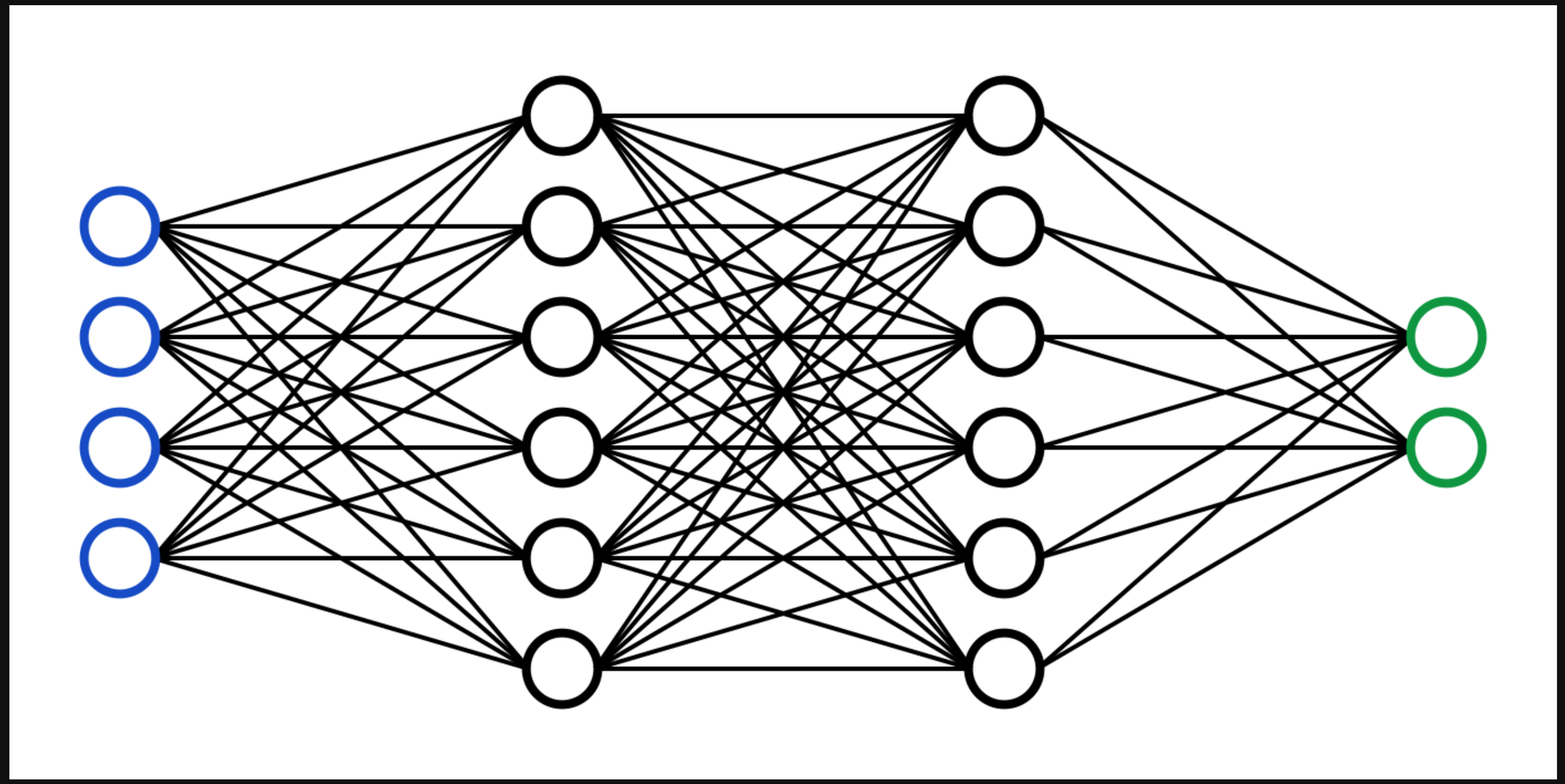
WHAT IF YOU DON'T KNOW THE FUNCTION?

$$f(x) \approx T(x)$$

~~ARTIFICIAL~~ NEURAL NETWORK

WHAT ARE NEURONS?

HOW ARE THEY CONNECTED?



NEURON;

HOLDS A NUMBER FROM 0 TO 1

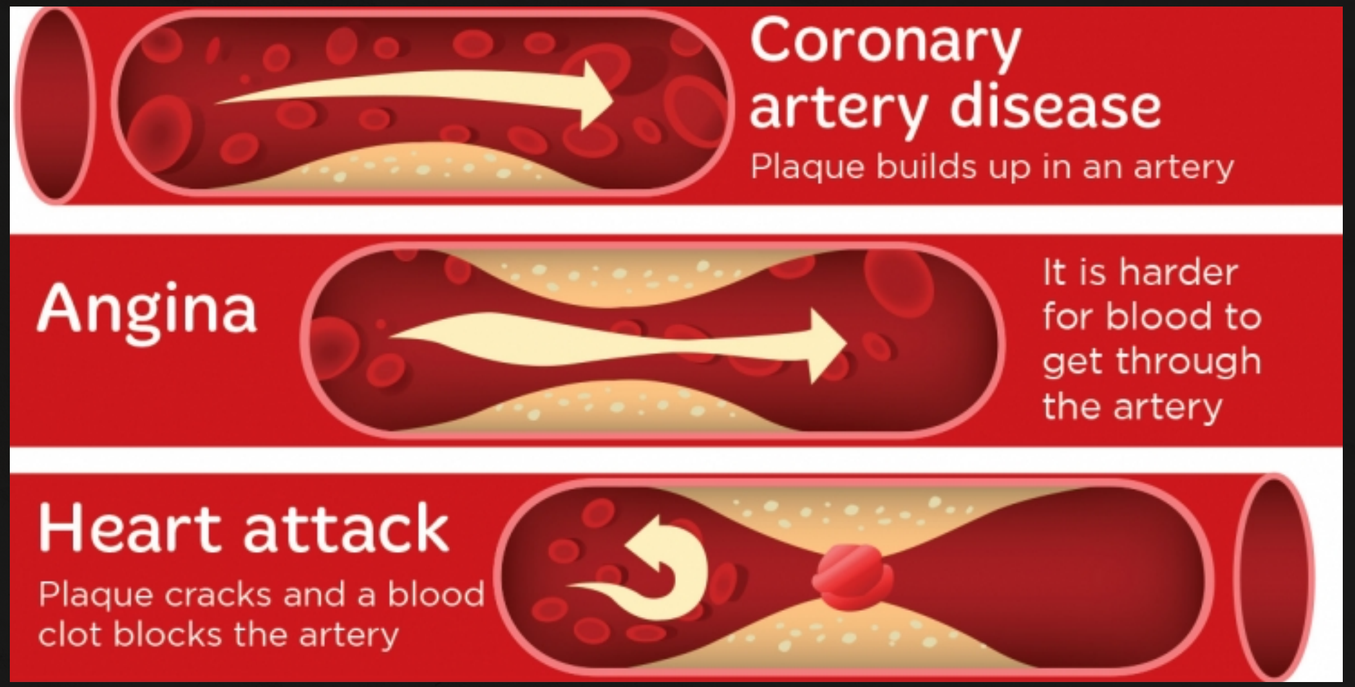
PRODUCES A RESULT BASED ON THE OUTPUT FROM THE PREVIOUS NEURON;

WHEN DOES A HEART ATTACK OCCUR?

The flow of blood to the heart is blocked;

Buildup of fat, cholesterol and other substances which form a plaque in the arteries.

Sometimes, a plaque can rupture and form a clot that blocks blood flow. The interrupted blood flow can damage or destroy part of the heart muscle.



Age : Age of the patient

Sex : Sex of the patient

exang: exercise induced angina (1 = yes; 0 = no)

ca: number of major vessels (0-3)
cp : Chest Pain type chest pain type

Value 1: typical angina
Value 2: atypical angina
Value 3: non-anginal pain
Value 4: asymptomatic
trtbps : resting blood pressure (in mm Hg)

chol : cholestoral in mg/dl fetched via BMI sensor

fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

rest_ecg : resting electrocardiographic results

DATA USED;

HEART ATTACK ANALYSIS & PREDICTION DATASET: KAGGLE

303

PATIENTS CONSIDERED.

Value 0 : normal
Value 1 : having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
Value 2 : showing probable or definite left ventricular hypertrophy by Estes' criteria

thalach : maximum heart rate achieved

Target : 0 = less chance of heart attack
1 = more chance of heart attack

```
     age  sex  cp  trtbps  chol  fbs  ...  exng  oldpeak  slp  caa  thall  output
0     63    1   3     145   233    1  ...     0      2.3    0    0      1       1
1     37    1   2     130   250    0  ...     0      3.5    0    0      2       1
2     41    0   1     130   204    0  ...     0      1.4    2    0      2       1
3     56    1   1     120   236    0  ...     0      0.8    2    0      2       1
4     57    0   0     120   354    0  ...     1      0.6    2    0      2       1
..   ...  ...  ..     ...   ...  ...  ...   ...      ...  ...  ...    ...     ...
298   57    0   0     140   241    0  ...     1      0.2    1    0      3       0
299   45    1   3     110   264    0  ...     0      1.2    1    0      3       0
300   68    1   0     144   193    1  ...     0      3.4    1    2      3       0
301   57    1   0     130   131    0  ...     1      1.2    1    1      3       0
302   57    0   1     130   236    0  ...     0      0.0    1    1      2       0

[303 rows x 14 columns]

Process finished with exit code 0
```

IMPORTING LIBRARIES ON PYTHON;

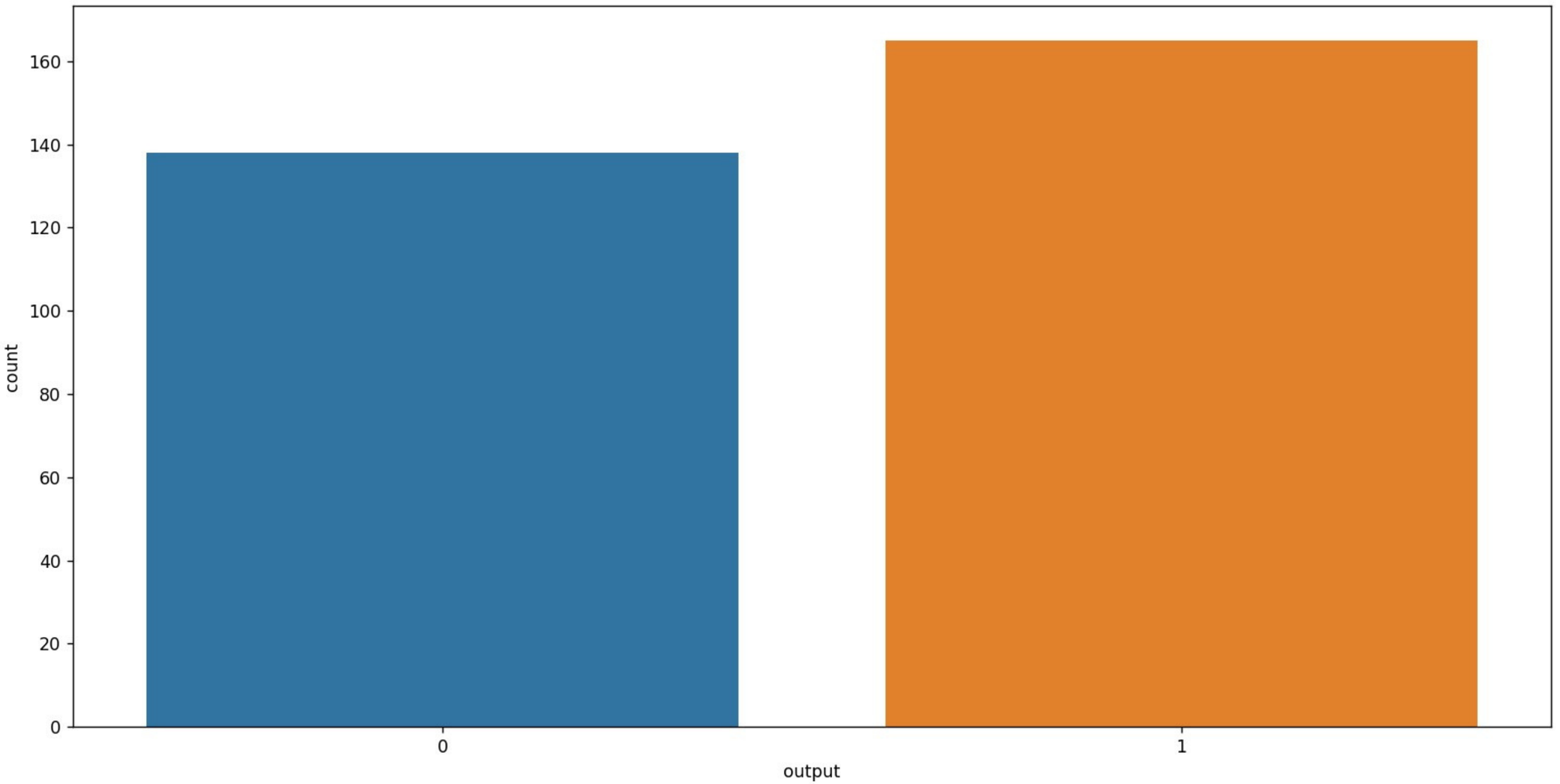
Test1.py

```
import tensorflow as tf
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

heart=pd.read_csv("/Users/Abhay Bhat/Desktop/heart.csv")
print(heart)
```

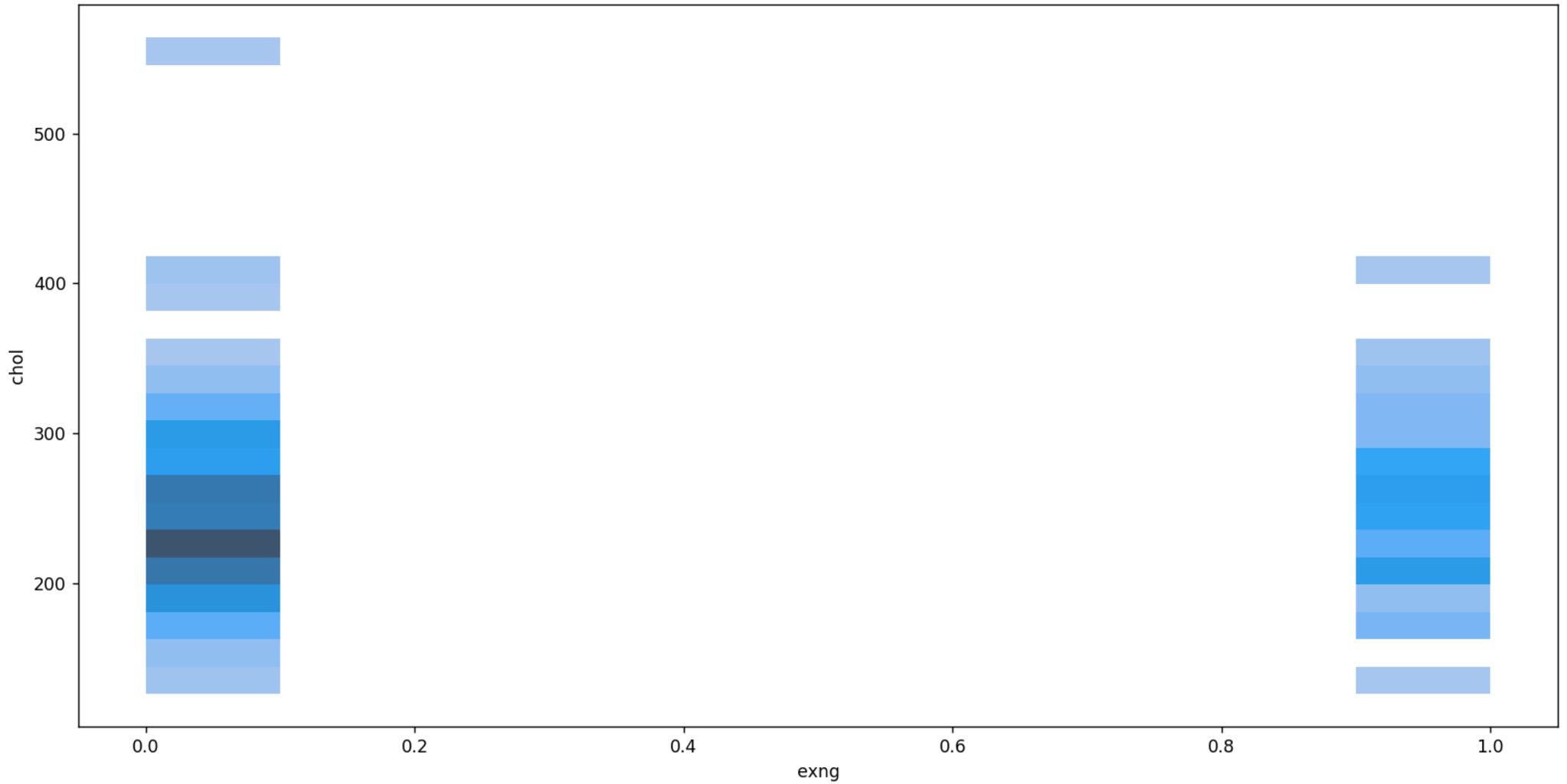
SIMPLE CODE; SEABORN

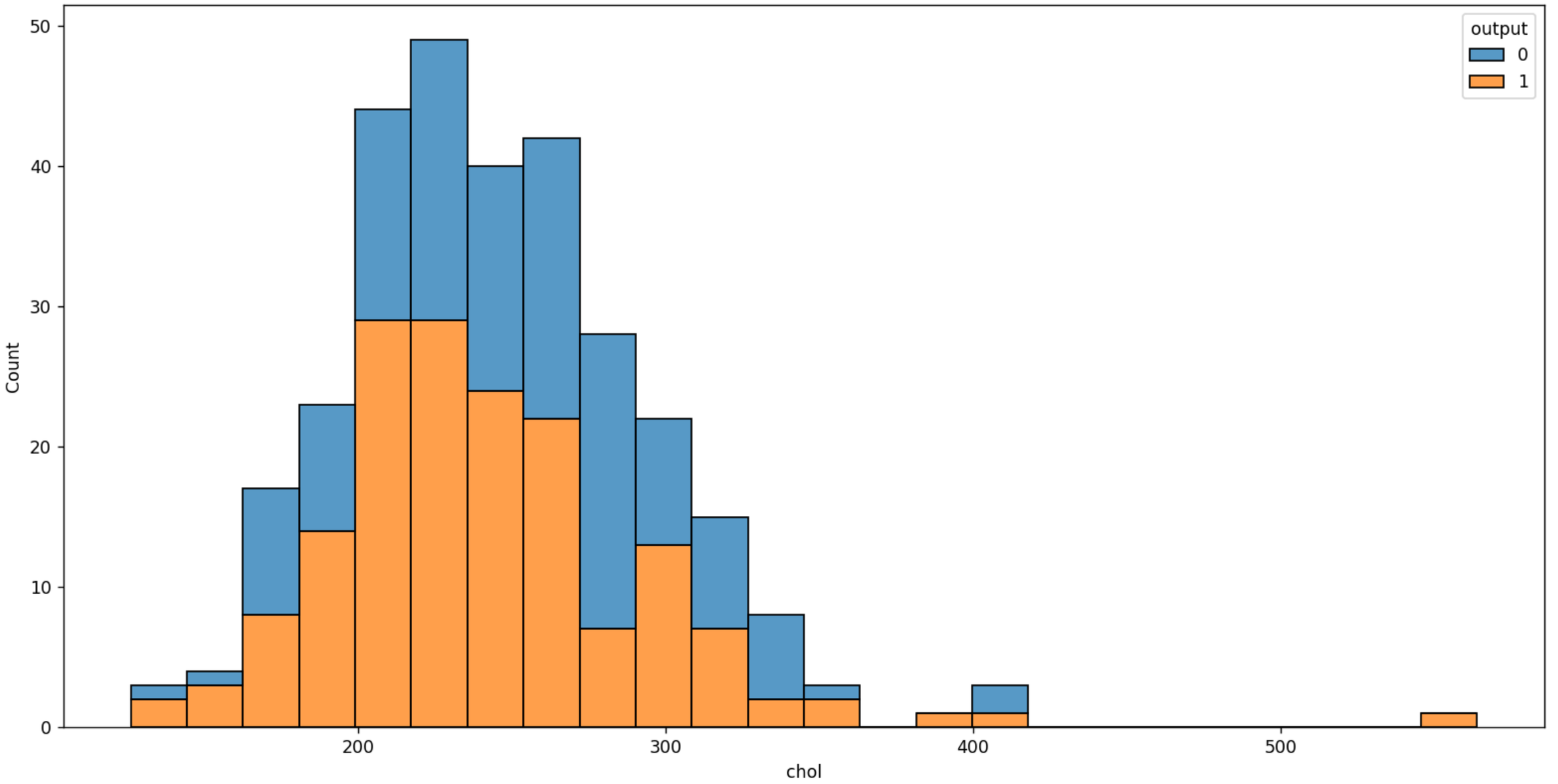
```
heart=pd.read_csv("/Users/Abhay Bhat/Desktop/heart.csv")

sns.histplot(data=heart, hue="species")
sns.histplot(data=heart, x="exng",y='chol')

plt.show()
```

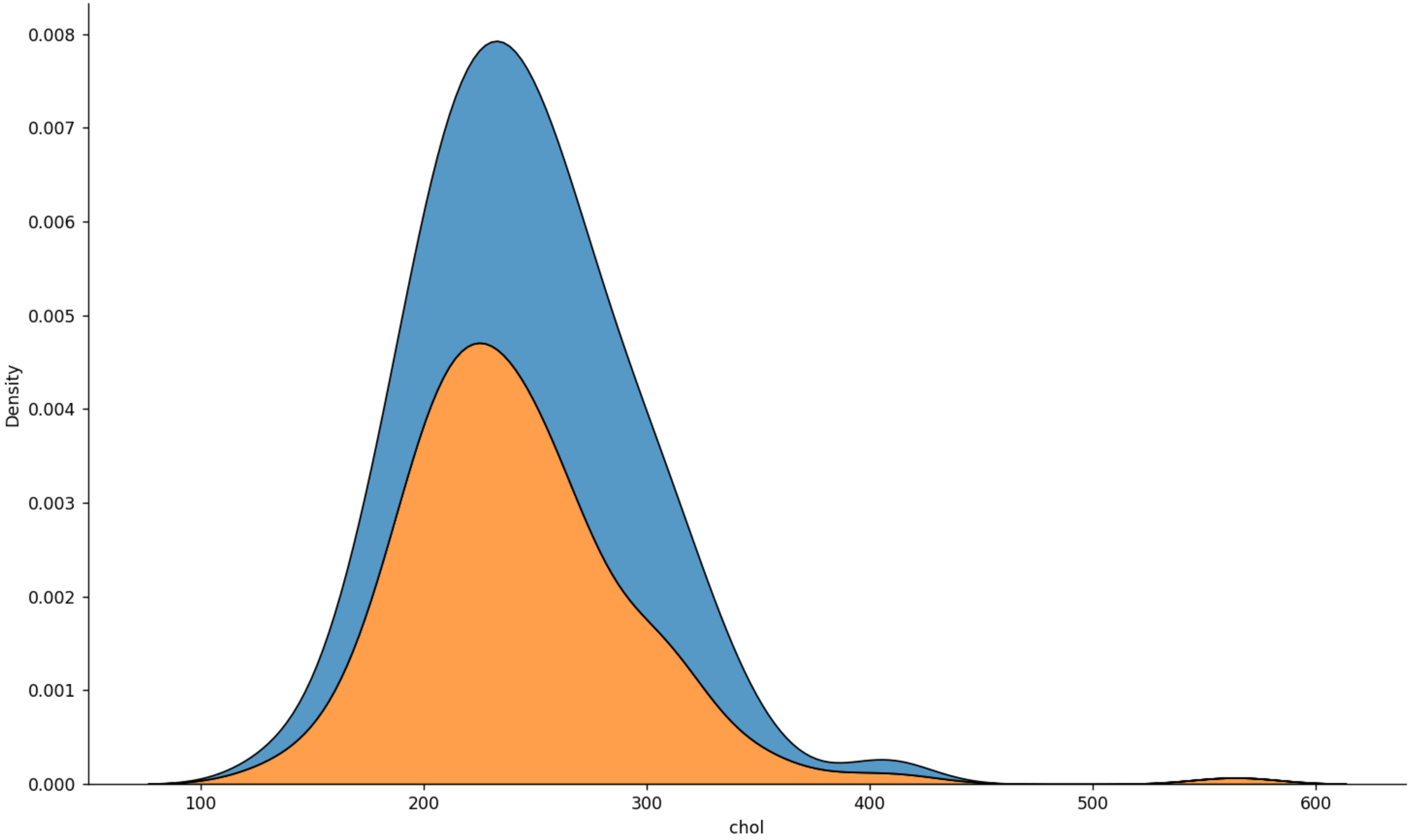
OUTPUT COUNT



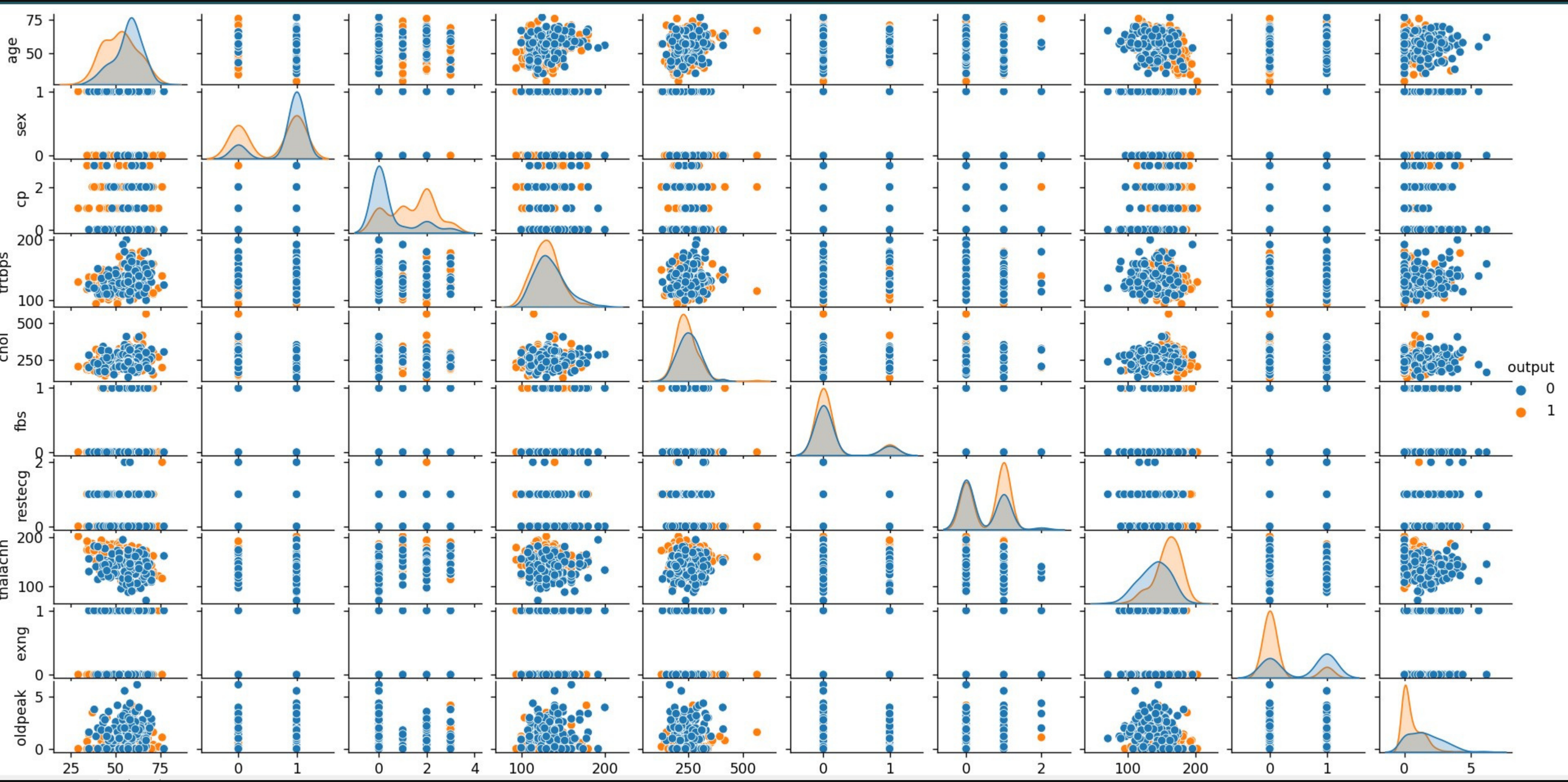
EXERCISE DIRECTLY CORRELATES TO LESSER CHOLESTEROL

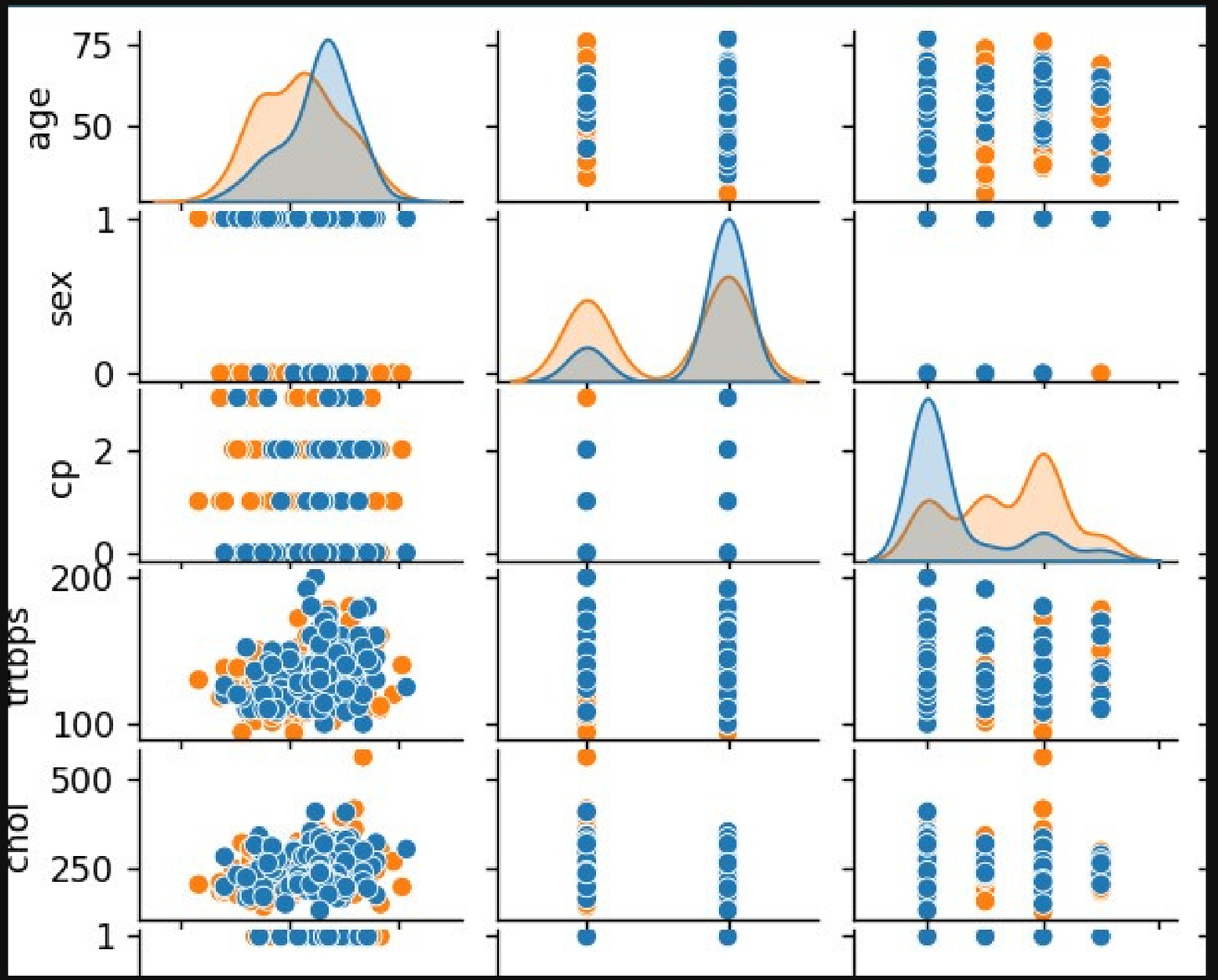



output
0
1
Count
chol
0
10
20
30
40
50
200
300
400
500


0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
Density
100
200
300
400
500
600
chol

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

heart=pd.read_csv("/Users/Abhay Bhat/Desktop/heart.csv")

sns.pairplot(heart, hue = 'output', vars = ['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
plt.show()
```


age
sex
cp
trtbps
chol
fbs
restecg
thalachh
exng
oldpeak
output
0
1


age
sex
cp
trtbps
chol
75
50
1
0
2
0
200
100
500
250
1

WHAT ARE THE SYMPTOMS?

- DISCOMFORT, PRESSURE, HEAVINESS, TIGHTNESS, SQUEEZING, OR PAIN IN YOUR CHEST OR ARM OR BELOW YOUR BREASTBONE
- DISCOMFORT THAT GOES INTO YOUR BACK, JAW, THROAT, OR ARM
- FULLNESS, INDIGESTION, OR A CHOKING FEELING (IT MAY FEEL LIKE HEARTBURN)
- SWEATING, UPSET STOMACH, VOMITING, OR DIZZINESS
- SEVERE WEAKNESS, ANXIETY, FATIGUE, OR SHORTNESS OF BREATH
- FAST OR UNEVEN HEARTBEAT

WHAT TO DO IF YOU THINK SOMEONE IS HAVING A HEART ATTACK?

FIRST CALL FOR EMERGENCY MEDICAL HELP.

THEN CHECK IF THE PERSON IS BREATHING AND HAS A PULSE.

IF THE PERSON ISN'T BREATHING OR YOU DON'T FIND A PULSE,
ONLY THEN SHOULD YOU BEGIN CPR.

CPR;

PUSH HARD AND FAST ON THE PERSON'S CHEST IN A FAIRLY RAPID RHYTHM — ABOUT 100 TO 120 COMPRESSIONS A MINUTE.

DATASET;

```
      age  sex  cp  trtbps  chol  fbs  ...  exng  oldpeak  slp  caa  thall  output
0      63    1   3     145   233    1  ...     0      2.3    0    0      1       1
1      37    1   2     130   250    0  ...     0      3.5    0    0      2       1
2      41    0   1     130   204    0  ...     0      1.4    2    0      2       1
3      56    1   1     120   236    0  ...     0      0.8    2    0      2       1
4      57    0   0     120   354    0  ...     1      0.6    2    0      2       1
..    ...  ...  ..     ...   ...  ...  ...   ...      ...  ...  ...    ...     ...
298    57    0   0     140   241    0  ...     1      0.2    1    0      3       0
299    45    1   3     110   264    0  ...     0      1.2    1    0      3       0
300    68    1   0     144   193    1  ...     0      3.4    1    2      3       0
301    57    1   0     130   131    0  ...     1      1.2    1    1      3       0
302    57    0   1     130   236    0  ...     0      0.0    1    1      2       0

[303 rows x 14 columns]

Process finished with exit code 0
```

SPLITTING THE DATA FOR TRAINING AND TESTING

```
X = heart.iloc[:, 0:13].values
y = heart.iloc[:, 13].values

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

[ROW,COLUMNS]

SCALING THE DATA AND SEPARATING INTO TESTING AND TRAINING SET;

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)

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)
print(X_train.shape)
print(X_test.shape)
```

DONE SUCH THAT ALL THE DATA IS TREATED EQUALLY

DO NOT WANT SOME FEATURES TO DOMINATE OVER THE OTHERS;

SCALED DATA

```
[[ 1.06248543  0.68100522 -0.93851463 ... -0.64911323  0.26508221
   1.12302895]
 [-1.47415758  0.68100522  0.03203122 ...  0.97635214 -0.71442887
  -0.51292188]
 [-0.48155814 -1.46841752  1.00257707 ... -0.64911323 -0.71442887
  -0.51292188]
 ...
 [-0.26098049  0.68100522  1.00257707 ...  0.97635214  3.20361543
  -0.51292188]
 [ 0.84190778 -1.46841752 -0.93851463 ... -2.27457861  2.22410436
   1.12302895]
 [ 0.40075247  0.68100522  1.00257707 ... -0.64911323 -0.71442887
   1.12302895]]
[1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 1
 1 1 1 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0
 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1
 1 1 0 0 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 0 0 1
 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 0 1
 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 1 0
 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 1]
```

TRAINING SET

TESTING SET

```
(242, 13)
(61, 13)

Process finished with exit code 0
```

GENERALIZE, NOT MEMORIZE

BUILDING NEURAL NETWORK USING KERAS

```
import tensorflow as tf

ANN_model = tf.keras.models.Sequential()
ANN_model.add(tf.keras.layers.Dense(units=400, activation='relu', input_shape=(13, )))
ANN_model.add(tf.keras.layers.Dropout(0.2))

ANN_model.add(tf.keras.layers.Dense(units=400, activation='relu'))
ANN_model.add(tf.keras.layers.Dropout(0.2))

ANN_model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ANN_model.summary()
```

RELU : RECTIFIED LINEAR OUTPUTS

IN ORDER TO INTRODUCE NON-LINEARITY IN THE NETWORKS

OUTPUT:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 400)               5600

 dropout (Dropout)           (None, 400)               0

 dense_1 (Dense)             (None, 400)               160400

 dropout_1 (Dropout)         (None, 400)               0

 dense_2 (Dense)             (None, 1)                 401

=================================================================
Total params: 166,401
Trainable params: 166,401
Non-trainable params: 0
```

166 TRAINABLE PARAMETERS;

IE. 166 WEIGHTS OR NEURONS TO OPTIMIZE;

OPTIMISING THE MODEL AND SEEING THE EPOCHS

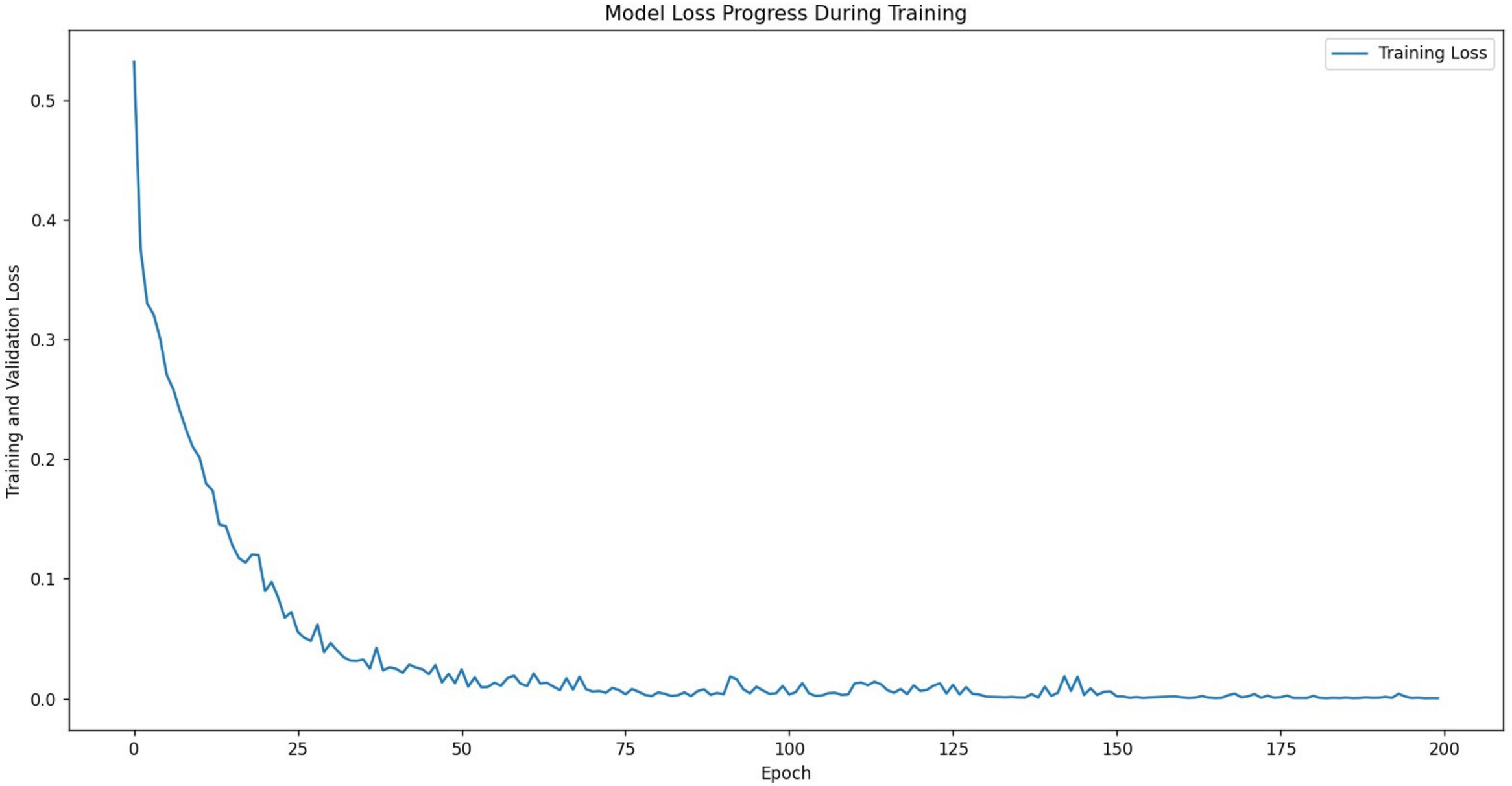
```
import tensorflow as tf

ANN_model = tf.keras.models.Sequential()
ANN_model.add(tf.keras.layers.Dense(units=400, activation='relu', input_shape=(13, )))
ANN_model.add(tf.keras.layers.Dropout(0.2))

ANN_model.add(tf.keras.layers.Dense(units=400, activation='relu'))
ANN_model.add(tf.keras.layers.Dropout(0.2))

ANN_model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ANN_model.compile(optimizer='Adam', loss='binary_crossentropy', metrics = ['accuracy'])

epochs_hist = ANN_model.fit(X_train, y_train, epochs = 200)
print(epochs_hist)
```

```
Epoch 1/200
8/8 [==============================] - 1s 4ms/step - loss: 0.5560 - accuracy: 0.7438
Epoch 2/200
8/8 [==============================] - 0s 4ms/step - loss: 0.3573 - accuracy: 0.8636
Epoch 3/200
8/8 [==============================] - 0s 2ms/step - loss: 0.3224 - accuracy: 0.8678
Epoch 4/200
8/8 [==============================] - 0s 4ms/step - loss: 0.2970 - accuracy: 0.8884
Epoch 5/200
8/8 [==============================] - 0s 2ms/step - loss: 0.2833 - accuracy: 0.8884
Epoch 6/200
8/8 [==============================] - 0s 5ms/step - loss: 0.2681 - accuracy: 0.8926
Epoch 7/200
8/8 [==============================] - 0s 1ms/step - loss: 0.2510 - accuracy: 0.9050
Epoch 8/200
8/8 [==============================] - 0s 2ms/step - loss: 0.2355 - accuracy: 0.9215
Epoch 9/200
8/8 [==============================] - 0s 4ms/step - loss: 0.2325 - accuracy: 0.9091
Epoch 10/200
8/8 [==============================] - 0s 4ms/step - loss: 0.2162 - accuracy: 0.9215
```

```
Epoch 190/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0026 - accuracy: 1.0000
Epoch 191/200
8/8 [==============================] - 0s 5ms/step - loss: 9.9855e-04 - accuracy: 1.000
Epoch 192/200
8/8 [==============================] - 0s 3ms/step - loss: 0.0019 - accuracy: 1.0000
Epoch 193/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0028 - accuracy: 1.0000
Epoch 194/200
8/8 [==============================] - 0s 4ms/step - loss: 0.0011 - accuracy: 1.0000
Epoch 195/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0030 - accuracy: 1.0000
Epoch 196/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0078 - accuracy: 0.9959
Epoch 197/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0293 - accuracy: 0.9917
Epoch 198/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0125 - accuracy: 0.9917
Epoch 199/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0255 - accuracy: 0.9917
Epoch 200/200
8/8 [==============================] - 0s 2ms/step - loss: 0.0202 - accuracy: 0.9917
<keras.callbacks.History object at 0x000001559B0C5760>
```

VISUALISING THE EPOCH HISTORY

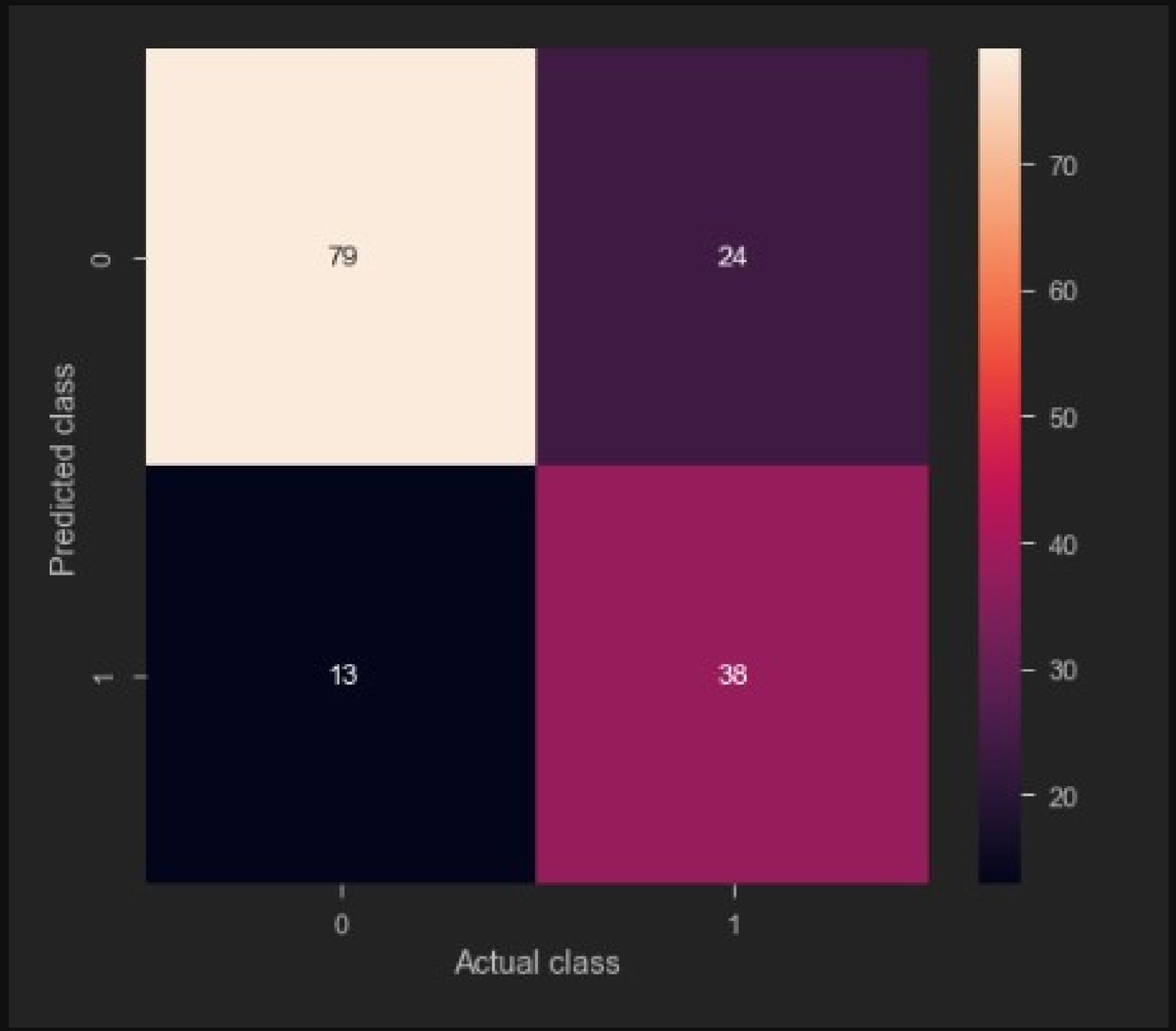
```
epochs_hist.history.keys()


plt.plot(epochs_hist.history['loss'])
plt.title('Model Loss Progress During Training')
plt.xlabel('Epoch')
plt.ylabel('Training and Validation Loss')
plt.legend(['Training Loss'])
plt.show()
```


Model Loss Progress During Training
Training Loss
Training and Validation Loss
0.0
0.1
0.2
0.3
0.4
0.5
0
25
50
75
100
125
150
175
200
Epoch

CONFUSION MATRIX:

GROUND TRUTH VS. PREDICTED MODEL

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot = True)
```


79
24
13
38
0
1
Predicted class
0
1
Actual class
70
60
50
40
30
20

THANK YOU!

ABHAY BHAT
PES1UG19BT002