In [4]:

```
import sys
import pandas as pd
import numpy as np
import sklearn
import matplotlib
import keras
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
```

In [2]:

```
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.c
```

In [6]:

```
names = ['age',
        'sex',
        'cp',
        'trestbps',
        'chol',
        'fbs',

        'restecg',
        'thalach',
        'exang',
        'oldpeak',
        'slope',
        'ca',
        'thal',
        'class']

# read the csv
cleveland = pd.read_csv(url, names=names)
```

In [7]:

```
print ('format(cleveland.shape')
print (cleveland.loc[1])
```

```
format(cleveland.shape
age          67.0
sex           1.0
cp            4.0
trestbps    160.0
chol        286.0
fbs           0.0
restecg       2.0
thalach     108.0
exang         1.0
oldpeak       1.5
slope         2.0
ca            3.0
thal          3.0
class           2
Name: 1, dtype: object
```

In [8]:

```
# print the last twenty or so data points
cleveland.loc[280:]
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	class
280	57.0	1.0	4.0	110.0	335.0	0.0	0.0	143.0	1.0	3.0	2.0	1.0	7.0	2
281	47.0	1.0	3.0	130.0	253.0	0.0	0.0	179.0	0.0	0.0	1.0	0.0	3.0	0
282	55.0	0.0	4.0	128.0	205.0	0.0	1.0	130.0	1.0	2.0	2.0	1.0	7.0	3
283	35.0	1.0	2.0	122.0	192.0	0.0	0.0	174.0	0.0	0.0	1.0	0.0	3.0	0
284	61.0	1.0	4.0	148.0	203.0	0.0	0.0	161.0	0.0	0.0	1.0	1.0	7.0	2
285	58.0	1.0	4.0	114.0	318.0	0.0	1.0	140.0	0.0	4.4	3.0	3.0	6.0	4
286	58.0	0.0	4.0	170.0	225.0	1.0	2.0	146.0	1.0	2.8	2.0	2.0	6.0	2
287	58.0	1.0	2.0	125.0	220.0	0.0	0.0	144.0	0.0	0.4	2.0	?	7.0	0
288	56.0	1.0	2.0	130.0	221.0	0.0	2.0	163.0	0.0	0.0	1.0	0.0	7.0	0
289	56.0	1.0	2.0	120.0	240.0	0.0	0.0	169.0	0.0	0.0	3.0	0.0	3.0	0
290	67.0	1.0	3.0	152.0	212.0	0.0	2.0	150.0	0.0	0.8	2.0	0.0	7.0	1
291	55.0	0.0	2.0	132.0	342.0	0.0	0.0	166.0	0.0	1.2	1.0	0.0	3.0	0
292	44.0	1.0	4.0	120.0	169.0	0.0	0.0	144.0	1.0	2.8	3.0	0.0	6.0	2
293	63.0	1.0	4.0	140.0	187.0	0.0	2.0	144.0	1.0	4.0	1.0	2.0	7.0	2
294	63.0	0.0	4.0	124.0	197.0	0.0	0.0	136.0	1.0	0.0	2.0	0.0	3.0	1
295	41.0	1.0	2.0	120.0	157.0	0.0	0.0	182.0	0.0	0.0	1.0	0.0	3.0	0
296	59.0	1.0	4.0	164.0	176.0	1.0	2.0	90.0	0.0	1.0	2.0	2.0	6.0	3
297	57.0	0.0	4.0	140.0	241.0	0.0	0.0	123.0	1.0	0.2	2.0	0.0	7.0	1
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	0

In [9]:

```
data = cleveland[~cleveland.isin(['?'])]
data.loc[280:]
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	class
280	57.0	1.0	4.0	110.0	335.0	0.0	0.0	143.0	1.0	3.0	2.0	1.0	7.0	2
281	47.0	1.0	3.0	130.0	253.0	0.0	0.0	179.0	0.0	0.0	1.0	0.0	3.0	0
282	55.0	0.0	4.0	128.0	205.0	0.0	1.0	130.0	1.0	2.0	2.0	1.0	7.0	3
283	35.0	1.0	2.0	122.0	192.0	0.0	0.0	174.0	0.0	0.0	1.0	0.0	3.0	0
284	61.0	1.0	4.0	148.0	203.0	0.0	0.0	161.0	0.0	0.0	1.0	1.0	7.0	2
285	58.0	1.0	4.0	114.0	318.0	0.0	1.0	140.0	0.0	4.4	3.0	3.0	6.0	4
286	58.0	0.0	4.0	170.0	225.0	1.0	2.0	146.0	1.0	2.8	2.0	2.0	6.0	2
287	58.0	1.0	2.0	125.0	220.0	0.0	0.0	144.0	0.0	0.4	2.0	NaN	7.0	0
288	56.0	1.0	2.0	130.0	221.0	0.0	2.0	163.0	0.0	0.0	1.0	0.0	7.0	0
289	56.0	1.0	2.0	120.0	240.0	0.0	0.0	169.0	0.0	0.0	3.0	0.0	3.0	0
290	67.0	1.0	3.0	152.0	212.0	0.0	2.0	150.0	0.0	0.8	2.0	0.0	7.0	1
291	55.0	0.0	2.0	132.0	342.0	0.0	0.0	166.0	0.0	1.2	1.0	0.0	3.0	0
292	44.0	1.0	4.0	120.0	169.0	0.0	0.0	144.0	1.0	2.8	3.0	0.0	6.0	2
293	63.0	1.0	4.0	140.0	187.0	0.0	2.0	144.0	1.0	4.0	1.0	2.0	7.0	2
294	63.0	0.0	4.0	124.0	197.0	0.0	0.0	136.0	1.0	0.0	2.0	0.0	3.0	1
295	41.0	1.0	2.0	120.0	157.0	0.0	0.0	182.0	0.0	0.0	1.0	0.0	3.0	0
296	59.0	1.0	4.0	164.0	176.0	1.0	2.0	90.0	0.0	1.0	2.0	2.0	6.0	3
297	57.0	0.0	4.0	140.0	241.0	0.0	0.0	123.0	1.0	0.2	2.0	0.0	7.0	1
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	NaN	3.0	0

In [10]:

```
# drop rows with NaN values from DataFrame
data = data.dropna(axis=0)
data.loc[280:]
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	class
280	57.0	1.0	4.0	110.0	335.0	0.0	0.0	143.0	1.0	3.0	2.0	1.0	7.0	2
281	47.0	1.0	3.0	130.0	253.0	0.0	0.0	179.0	0.0	0.0	1.0	0.0	3.0	0
282	55.0	0.0	4.0	128.0	205.0	0.0	1.0	130.0	1.0	2.0	2.0	1.0	7.0	3
283	35.0	1.0	2.0	122.0	192.0	0.0	0.0	174.0	0.0	0.0	1.0	0.0	3.0	0
284	61.0	1.0	4.0	148.0	203.0	0.0	0.0	161.0	0.0	0.0	1.0	1.0	7.0	2
285	58.0	1.0	4.0	114.0	318.0	0.0	1.0	140.0	0.0	4.4	3.0	3.0	6.0	4
286	58.0	0.0	4.0	170.0	225.0	1.0	2.0	146.0	1.0	2.8	2.0	2.0	6.0	2
288	56.0	1.0	2.0	130.0	221.0	0.0	2.0	163.0	0.0	0.0	1.0	0.0	7.0	0
289	56.0	1.0	2.0	120.0	240.0	0.0	0.0	169.0	0.0	0.0	3.0	0.0	3.0	0
290	67.0	1.0	3.0	152.0	212.0	0.0	2.0	150.0	0.0	0.8	2.0	0.0	7.0	1
291	55.0	0.0	2.0	132.0	342.0	0.0	0.0	166.0	0.0	1.2	1.0	0.0	3.0	0
292	44.0	1.0	4.0	120.0	169.0	0.0	0.0	144.0	1.0	2.8	3.0	0.0	6.0	2
293	63.0	1.0	4.0	140.0	187.0	0.0	2.0	144.0	1.0	4.0	1.0	2.0	7.0	2
294	63.0	0.0	4.0	124.0	197.0	0.0	0.0	136.0	1.0	0.0	2.0	0.0	3.0	1
295	41.0	1.0	2.0	120.0	157.0	0.0	0.0	182.0	0.0	0.0	1.0	0.0	3.0	0
296	59.0	1.0	4.0	164.0	176.0	1.0	2.0	90.0	0.0	1.0	2.0	2.0	6.0	3
297	57.0	0.0	4.0	140.0	241.0	0.0	0.0	123.0	1.0	0.2	2.0	0.0	7.0	1
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1

In [11]:

```
# print the shape and data type of the dataframe
print (data.shape)
print (data.dtypes)
```

```
(297, 14)
age           float64
sex           float64
cp            float64
trestbps      float64
chol          float64
fbs           float64
restecg       float64
thalach       float64
exang         float64
oldpeak       float64
slope         float64
ca             object
thal           object
class           int64
dtype: object
```

In [12]:
```python
# transform data to numeric to enable further analysis
data = data.apply(pd.to_numeric)
data.dtypes
```

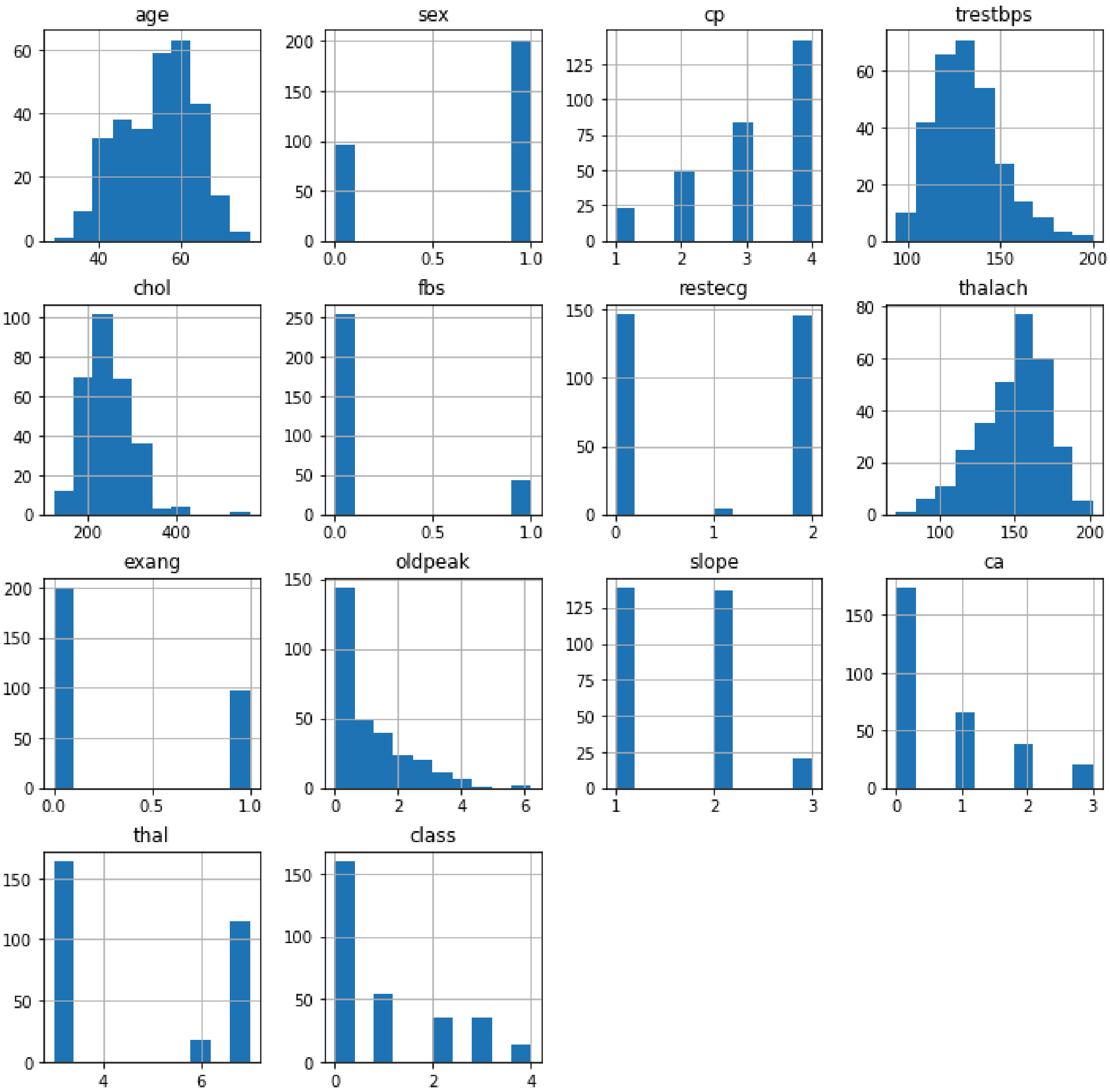
Out[12]:
```
age         float64
sex         float64
cp          float64
trestbps    float64
chol        float64
fbs         float64
restecg     float64
thalach     float64
exang       float64
oldpeak     float64
slope       float64
ca          float64
thal        float64
class         int64
dtype: object
```

In [13]:
```python
# print data characteristics, usings pandas built-in describe() function
data.describe()
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	e
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.00
mean	54.542088	0.676768	3.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.32
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.46
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.00
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.00
50%	56.000000	1.000000	3.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.00
75%	61.000000	1.000000	4.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.00
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.00

```
In [14]: # plot histograms for each variable
         data.hist(figsize = (12, 12))
         plt.show()
```



```
In [15]: # create X and Y datasets for training
         from sklearn import model_selection

         X = np.array(data.drop(['class'], 1))
         y = np.array(data['class'])

         X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.2
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_32748\1346931819.py:4: FutureWarning: In a futu
re version of pandas all arguments of DataFrame.drop except for the argument 'labels' wi
ll be keyword-only.
  X = np.array(data.drop(['class'], 1))
```

In [16]:

```
# convert the data to categorical labels
from keras.utils import to_categorical

Y_train = to_categorical(y_train, num_classes=None)
Y_test = to_categorical(y_test, num_classes=None)
print (Y_train.shape)
print (Y_train[:10])
```

```
(237, 5)
[[1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
```

In [17]:

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

# define a function to build the keras model
def create_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=13, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(5, activation='softmax'))

    # compile model
    adam = Adam(lr=0.001)
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    return model

model = create_model()

print(model.summary())
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\keras\src\backend.py:87
3: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_grap
h instead.


WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use th
e legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 112

 dense_1 (Dense)             (None, 4)                 36

 dense_2 (Dense)             (None, 5)                 25

=================================================================
Total params: 173 (692.00 Byte)
Trainable params: 173 (692.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

```
In [18]: # fit the model to the training data
         model.fit(X_train, Y_train, epochs=100, batch_size=10, verbose = 1)
```

```
Epoch 40/100
24/24 [==============================] - 0s 3ms/step - loss: 1.0766 - accuracy: 0.569
6
Epoch 41/100
24/24 [==============================] - 0s 5ms/step - loss: 1.0692 - accuracy: 0.561
2
Epoch 42/100
24/24 [==============================] - 0s 11ms/step - loss: 1.0631 - accuracy: 0.57
38
Epoch 43/100
24/24 [==============================] - 0s 7ms/step - loss: 1.0630 - accuracy: 0.573
8
Epoch 44/100
24/24 [==============================] - 0s 5ms/step - loss: 1.0554 - accuracy: 0.565
4
Epoch 45/100
24/24 [==============================] - 0s 3ms/step - loss: 1.0478 - accuracy: 0.573
8
Epoch 46/100
24/24 [==============================] - 0s 8ms/step - loss: 1.0523 - accuracy: 0.565
```

```
In [19]: # convert into binary classification problem - heart disease or no heart disease
         Y_train_binary = y_train.copy()
         Y_test_binary = y_test.copy()

         Y_train_binary[Y_train_binary > 0] = 1
         Y_test_binary[Y_test_binary > 0] = 1

         print (Y_train_binary[:20])
```

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

In [20]:
```python
# define a new keras model for binary classification
def create_binary_model():
    # create model
    model = Sequential()
    model.add(Dense(8, input_dim=13, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model
    adam = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    return model

binary_model = create_binary_model()

print(binary_model.summary())
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use th
e legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_3 (Dense)             (None, 8)                 112       
                                                                 
 dense_4 (Dense)             (None, 4)                 36        
                                                                 
 dense_5 (Dense)             (None, 1)                 5         
                                                                 
=================================================================
Total params: 153 (612.00 Byte)
Trainable params: 153 (612.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

In [21]:
```python
# fit the binary model on the training data
binary_model.fit(X_train, Y_train_binary, epochs=100, batch_size=10, verbose = 1)
```

```
2
Epoch 11/100
24/24 [==============================] - 0s 3ms/step - loss: 0.6928 - accuracy: 0.523
2
Epoch 12/100
24/24 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.523
2
Epoch 13/100
24/24 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.523
2
Epoch 14/100
24/24 [==============================] - 0s 3ms/step - loss: 0.6926 - accuracy: 0.523
2
Epoch 15/100
24/24 [==============================] - 0s 3ms/step - loss: 0.6926 - accuracy: 0.523
2
Epoch 16/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6925 - accuracy: 0.523
2
Epoch 17/100
```

In [22]:

```
# generate classification report using predictions for categorical model
from sklearn.metrics import classification_report, accuracy_score

categorical_pred = np.argmax(model.predict(X_test), axis=1)

print('Results for Categorical Model')
print(accuracy_score(y_test, categorical_pred))
print(classification_report(y_test, categorical_pred))
```

```
2/2 [==============================] - 0s 0s/step
Results for Categorical Model
0.6333333333333333
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        36
           1       0.25      0.10      0.14        10
           2       0.00      0.00      0.00         6
           3       0.29      0.83      0.43         6
           4       0.00      0.00      0.00         2

    accuracy                           0.63        60
   macro avg       0.27      0.36      0.29        60
weighted avg       0.56      0.63      0.58        60


C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [23]:

```
# generate classification report using predictions for binary model
binary_pred = np.round(binary_model.predict(X_test)).astype(int)

print('Results for Binary Model')
print(accuracy_score(Y_test_binary, binary_pred))
print(classification_report(Y_test_binary, binary_pred))
```

```
2/2 [==============================] - 0s 0s/step
Results for Binary Model
0.6
              precision    recall  f1-score   support

           0       0.60      1.00      0.75        36
           1       0.00      0.00      0.00        24

    accuracy                           0.60        60
   macro avg       0.30      0.50      0.37        60
weighted avg       0.36      0.60      0.45        60


C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\hp\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]: