In [6]:
```
import math
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import Model
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
```

In [7]:
```
df = np.loadtxt('https://raw.githubusercontent.com/jbrownlee/Datasets/master/pim
```

In [8]:
```
df
```

Out[8]:
```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

In [9]:
```
df.shape
```

Out[9]:
```
(768, 9)
```

In [10]:
```
x = df[:,:8]
y = df[:,8]
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(x, y, test_size=0.2, random_

X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp, test_size=0.5, r
```

In [12]:
```
print(f"x train shape{X_train.shape}")
print(f"y train shape{y_train.shape}")
print(f"x test shape{X_test.shape}")
print(f"y test shape{y_test.shape}")
print(f"x val shape{X_val.shape}")
print(f"y val shape{y_val.shape}")
```

```
x train shape(614, 8)
y train shape(614,)
x test shape(77, 8)
y test shape(77,)
x val shape(77, 8)
y val shape(77,)
```

In [13]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_val = scaler.transform(X_val)
```

In [14]:
```
from collections import Counter
Counter(y)
```

Out[14]:
```
Counter({0.0: 500, 1.0: 268})
```

In [15]:
```
import seaborn as sns
```

In [13]:
```
from tensorflow.keras.models import Sequential
```

In [16]:
```
model = Sequential([

tf.keras.layers.InputLayer(8,),
Dense(50,activation='relu'),

Dense(50,activation='relu'),
Dense(50,activation='relu'),
Dense(50,activation='relu'),

Dense(1,activation='sigmoid')
])
```

In [17]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                450

 dense_1 (Dense)             (None, 50)                2550

 dense_2 (Dense)             (None, 50)                2550

 dense_3 (Dense)             (None, 50)                2550

 dense_4 (Dense)             (None, 1)                 51

=================================================================
Total params: 8151 (31.84 KB)
Trainable params: 8151 (31.84 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [18]:
```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
```

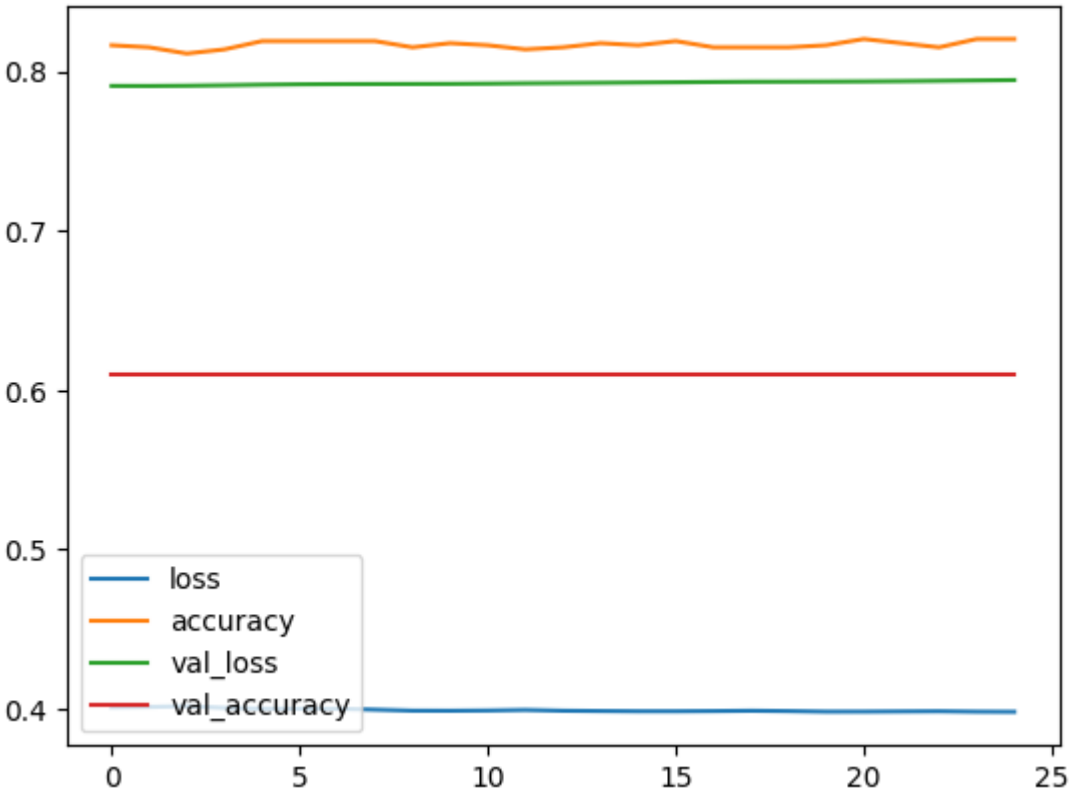
In [30]:
```
history = model.fit(x=x,y=y,epochs=25, batch_size=500,validation_data=(X_val,y_v
```

```
Epoch 1/25
2/2 [==============================] - 0s 123ms/step - loss: 0.4009 - accuracy:
0.8164 - val_loss: 0.7909 - val_accuracy: 0.6104
Epoch 2/25
2/2 [==============================] - 0s 66ms/step - loss: 0.4012 - accuracy: 0.
8151 - val_loss: 0.7909 - val_accuracy: 0.6104
Epoch 3/25
2/2 [==============================] - 0s 69ms/step - loss: 0.4016 - accuracy: 0.
8112 - val_loss: 0.7910 - val_accuracy: 0.6104
Epoch 4/25
2/2 [==============================] - 0s 62ms/step - loss: 0.4006 - accuracy: 0.
8138 - val_loss: 0.7913 - val_accuracy: 0.6104
Epoch 5/25
2/2 [==============================] - 0s 67ms/step - loss: 0.4001 - accuracy: 0.
8190 - val_loss: 0.7916 - val_accuracy: 0.6104
Epoch 6/25
2/2 [==============================] - 0s 55ms/step - loss: 0.4001 - accuracy: 0.
8190 - val_loss: 0.7919 - val_accuracy: 0.6104
Epoch 7/25
2/2 [==============================] - 0s 64ms/step - loss: 0.4001 - accuracy: 0.
8190 - val_loss: 0.7920 - val_accuracy: 0.6104
Epoch 8/25
2/2 [==============================] - 0s 78ms/step - loss: 0.3995 - accuracy: 0.
8190 - val_loss: 0.7921 - val_accuracy: 0.6104
Epoch 9/25
2/2 [==============================] - 0s 41ms/step - loss: 0.3988 - accuracy: 0.
8151 - val_loss: 0.7921 - val_accuracy: 0.6104
Epoch 10/25
2/2 [==============================] - 0s 43ms/step - loss: 0.3988 - accuracy: 0.
8177 - val_loss: 0.7922 - val_accuracy: 0.6104
Epoch 11/25
2/2 [==============================] - 0s 68ms/step - loss: 0.3989 - accuracy: 0.
8164 - val_loss: 0.7923 - val_accuracy: 0.6104
Epoch 12/25
2/2 [==============================] - 0s 60ms/step - loss: 0.3993 - accuracy: 0.
8138 - val_loss: 0.7925 - val_accuracy: 0.6104
Epoch 13/25
2/2 [==============================] - 0s 75ms/step - loss: 0.3988 - accuracy: 0.
8151 - val_loss: 0.7927 - val_accuracy: 0.6104
Epoch 14/25
2/2 [==============================] - 0s 51ms/step - loss: 0.3986 - accuracy: 0.
8177 - val_loss: 0.7928 - val_accuracy: 0.6104
Epoch 15/25
2/2 [==============================] - 0s 66ms/step - loss: 0.3985 - accuracy: 0.
8164 - val_loss: 0.7930 - val_accuracy: 0.6104
Epoch 16/25
2/2 [==============================] - 0s 106ms/step - loss: 0.3985 - accuracy:
0.8190 - val_loss: 0.7932 - val_accuracy: 0.6104
Epoch 17/25
2/2 [==============================] - 0s 82ms/step - loss: 0.3986 - accuracy: 0.
8151 - val_loss: 0.7933 - val_accuracy: 0.6104
Epoch 18/25
2/2 [==============================] - 0s 85ms/step - loss: 0.3988 - accuracy: 0.
8151 - val_loss: 0.7935 - val_accuracy: 0.6104
Epoch 19/25
2/2 [==============================] - 0s 64ms/step - loss: 0.3986 - accuracy: 0.
8151 - val_loss: 0.7935 - val_accuracy: 0.6104
Epoch 20/25
2/2 [==============================] - 0s 48ms/step - loss: 0.3982 - accuracy: 0.
8164 - val_loss: 0.7936 - val_accuracy: 0.6104
```

```
Epoch 21/25
2/2 [==============================] - 0s 64ms/step - loss: 0.3982 - accuracy: 0.
8203 - val_loss: 0.7937 - val_accuracy: 0.6104
Epoch 22/25
2/2 [==============================] - 0s 68ms/step - loss: 0.3984 - accuracy: 0.
8177 - val_loss: 0.7939 - val_accuracy: 0.6104
Epoch 23/25
2/2 [==============================] - 0s 91ms/step - loss: 0.3985 - accuracy: 0.
8151 - val_loss: 0.7941 - val_accuracy: 0.6104
Epoch 24/25
2/2 [==============================] - 0s 98ms/step - loss: 0.3982 - accuracy: 0.
8203 - val_loss: 0.7943 - val_accuracy: 0.6104
Epoch 25/25
2/2 [==============================] - 0s 64ms/step - loss: 0.3981 - accuracy: 0.
8203 - val_loss: 0.7945 - val_accuracy: 0.6104
```

In [31]:
```
losses = pd.DataFrame(model.history.history)
losses.plot()
```

Out[31]: <AxesSubplot: >



In [32]:
```
model.evaluate(x,y)
```

```
24/24 [==============================] - 0s 7ms/step - loss: 0.3981 - accuracy:
0.8190
```

Out[32]: [0.3981264531612396, 0.8190104365348816]

In [33]:
```
y_pred = model.predict(X_test)
y_pred
```

```
3/3 [==============================] - 0s 6ms/step
```

Out[33]:
```
array([[0.19067593],
       [0.19199562],
       [0.12028762],
       [0.12056991],
       [0.08439234],
       [0.17099527],
       [0.12366695],
       [0.20651293],
       [0.20941985],
       [0.15472724],
       [0.13664874],
       [0.21422127],
       [0.15540652],
       [0.17577767],
       [0.18489483],
       [0.20770562],
       [0.18772168],
       [0.22056726],
       [0.21485648],
       [0.21828829],
       [0.12115736],
       [0.1883155 ],
       [0.24208197],
       [0.14715748],
       [0.15659195],
       [0.12092391],
       [0.20264044],
       [0.1654087 ],
       [0.16633132],
       [0.18862365],
       [0.21634044],
       [0.18640423],
       [0.2316734 ],
       [0.17272425],
       [0.18025881],
       [0.2274827 ],
       [0.25268403],
       [0.16944598],
       [0.14875412],
       [0.13752207],
       [0.1651227 ],
       [0.14246757],
       [0.20300068],
       [0.16977538],
       [0.1331932 ],
       [0.20911501],
       [0.18446396],
       [0.07179271],
       [0.17416352],
       [0.16832711],
       [0.19544922],
       [0.25303903],
       [0.17899173],
       [0.2198722 ],
       [0.21194766],
       [0.16202568],
       [0.0634077 ],
       [0.20405914],
       [0.18574911],
       [0.2142827 ],
```

```
[0.1272719 ],
[0.15324473],
[0.21843497],
[0.13171278],
[0.17464535],
[0.2314195 ],
[0.1949308 ],
[0.20401546],
[0.19573765],
[0.17276601],
[0.21602567],
[0.22562845],
[0.20881379],
[0.16267908],
[0.15343472],
[0.0625412 ],
[0.23691072]], dtype=float32)
```