Simple Ann:

```
import numpy as np

import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import SGD
from sklearn.metrics import accuracy_score
```

```
df=pd.read_csv('/content/drive/MyDrive/Deep Learning/Breast_cancer_data.csv')
df
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	(
1	20.57	17.77	132.90	1326.0	0.08474	(
2	19.69	21.25	130.00	1203.0	0.10960	(
3	11.42	20.38	77.58	386.1	0.14250	(
4	20.29	14.34	135.10	1297.0	0.10030	(
...	...	...	...	...	...	..
564	21.56	22.39	142.00	1479.0	0.11100	(
565	20.13	28.25	131.20	1261.0	0.09780	(
566	16.60	28.08	108.30	858.1	0.08455	(
567	20.60	29.33	140.10	1265.0	0.11780	(
568	7.76	24.54	47.92	181.0	0.05263	

569 rows × 6 columns

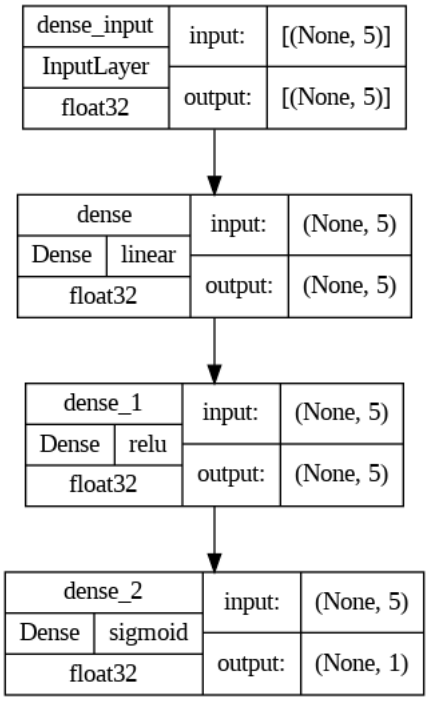
Next steps: View recommended plots

```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
print(x.shape,y.shape)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
(569, 5) (569,)
```

```
# shalllow neural network
model = Sequential()
model.add(Dense(5,input_dim=5))
model.add(Dense(5,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

```
keras.utils.plot_model(
    model,
    show_shapes = True,
    show_dtype = True,
    show_layer_activations = True
)
```




```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.fit(x_train,y_train,epochs=100)
```

```
Epoch 73/100
13/13 [==============================] - 0s 2ms/step - loss: 0.5809 - accuracy: 0.7739
Epoch 74/100
13/13 [==============================] - 0s 2ms/step - loss: 0.5830 - accuracy: 0.7965
```

```
y_pred=model.predict(x_test)
y_pred=y_pred>0.5
print(accuracy_score(y_pred,y_test))
```

```
6/6 [==============================] - 0s 4ms/step
0.8070175438596491
```

Autoencoder:

```
encoding_dim = 4
input_img = keras.Input(shape=(5))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(5, activation='sigmoid')(encoded)
```

```
autoencoder = keras.Model(input_img, decoded)
```

```
df=pd.read_csv('/content/drive/MyDrive/Deep Learning/Breast_cancer_data.csv')
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
print(x.shape,y.shape)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
(569, 5) (569,)
```

```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

```
autoencoder.fit(x_train,x_train,
                epochs=50,
                batch_size=256,
                validation_data=(x_test,x_test))
```

```
Epoch 1/50
2/2 [==============================] - 1s 214ms/step - loss: 9783.4902 - val_loss: 8735.9053
Epoch 2/50
2/2 [==============================] - 0s 30ms/step - loss: 9571.1670 - val_loss: 8539.3711
Epoch 3/50
2/2 [==============================] - 0s 31ms/step - loss: 9354.4014 - val_loss: 8340.8828
Epoch 4/50
2/2 [==============================] - 0s 28ms/step - loss: 9140.4834 - val_loss: 8140.6792
Epoch 5/50
2/2 [==============================] - 0s 30ms/step - loss: 8918.7900 - val_loss: 7940.3555
Epoch 6/50
2/2 [==============================] - 0s 29ms/step - loss: 8701.1025 - val_loss: 7737.8398
Epoch 7/50
2/2 [==============================] - 0s 30ms/step - loss: 8471.5361 - val_loss: 7534.0483
Epoch 8/50
2/2 [==============================] - 0s 32ms/step - loss: 8253.6162 - val_loss: 7329.3340
Epoch 9/50
2/2 [==============================] - 0s 32ms/step - loss: 8027.2500 - val_loss: 7124.7705
Epoch 10/50
2/2 [==============================] - 0s 33ms/step - loss: 7799.0752 - val_loss: 6920.1113
Epoch 11/50
2/2 [==============================] - 0s 34ms/step - loss: 7578.4644 - val_loss: 6713.2822
Epoch 12/50
2/2 [==============================] - 0s 34ms/step - loss: 7345.1382 - val_loss: 6506.2842
Epoch 13/50
2/2 [==============================] - 0s 35ms/step - loss: 7114.8022 - val_loss: 6298.7041
Epoch 14/50
2/2 [==============================] - 0s 32ms/step - loss: 6890.4492 - val_loss: 6090.1729
Epoch 15/50
2/2 [==============================] - 0s 33ms/step - loss: 6654.1289 - val_loss: 5881.4800
Epoch 16/50
2/2 [==============================] - 0s 31ms/step - loss: 6429.2036 - val_loss: 5671.3862
Epoch 17/50
2/2 [==============================] - 0s 35ms/step - loss: 6202.1191 - val_loss: 5460.8896
Epoch 18/50
2/2 [==============================] - 0s 32ms/step - loss: 5969.4131 - val_loss: 5250.6157
Epoch 19/50
2/2 [==============================] - 0s 33ms/step - loss: 5734.5469 - val_loss: 5040.1362
Epoch 20/50
2/2 [==============================] - 0s 35ms/step - loss: 5506.2915 - val_loss: 4828.6992
Epoch 21/50
2/2 [==============================] - 0s 35ms/step - loss: 5271.1494 - val_loss: 4616.9355
Epoch 22/50
2/2 [==============================] - 0s 34ms/step - loss: 5038.3525 - val_loss: 4404.2661
Epoch 23/50
```

```
2/2 [==============================] - 0s 47ms/step - loss: 4801.1094 - val_loss: 4190.7690
Epoch 24/50
2/2 [==============================] - 0s 32ms/step - loss: 4568.7300 - val_loss: 3975.8101
Epoch 25/50
2/2 [==============================] - 0s 34ms/step - loss: 4333.0967 - val_loss: 3759.9697
Epoch 26/50
2/2 [==============================] - 0s 35ms/step - loss: 4094.8875 - val_loss: 3543.3884
Epoch 27/50
2/2 [==============================] - 0s 32ms/step - loss: 3856.9536 - val_loss: 3325.8796
Epoch 28/50
2/2 [==============================] - 0s 32ms/step - loss: 3621.8401 - val_loss: 3107.5200
Epoch 29/50
2/2 [==============================] - 0s 34ms/step - loss: 3372.9333 - val_loss: 2889.1646
```

```
encoder = keras.Model(input_img, encoded)
encoded_input = keras.Input(shape=(encoding_dim,))
decoder_layer = autoencoder.layers[-1]
decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
```

```
encoded_inp = encoder.predict(x_test)
print(encoded_inp)
decoded_inp = decoder.predict(encoded_inp)
print(decoded_inp)
```

```
 2.41535426e-14]
[1.81601426e-25 9.99875844e-01 1.00000000e+00 9.98299241e-01
 2.30189889e-19]
[7.15302399e-26 9.99926746e-01 1.00000000e+00 9.97221053e-01
 1.53390421e-19]
[2.48551596e-23 9.99775350e-01 1.00000000e+00 9.96183753e-01
 1.05920329e-17]
[5.79516315e-21 9.99623060e-01 1.00000000e+00 9.88439143e-01
 8.21881204e-16]
[7.82349326e-18 9.99299407e-01 1.00000000e+00 9.46508169e-01
 2.72134528e-13]
[8.51992234e-21 9.99655366e-01 1.00000000e+00 9.84830618e-01
 1.23194632e-15]
[1.11794066e-15 9.98528004e-01 1.00000000e+00 9.07264411e-01
 1.15864063e-11]
[1.36165561e-21 9.99615312e-01 1.00000000e+00 9.93245542e-01
 2.29065502e-16]
[4.46516226e-26 9.99882340e-01 1.00000000e+00 9.98871505e-01
 7.06685213e-20]
[2.24704809e-20 9.99459922e-01 1.00000000e+00 9.89342332e-01
 2.02910702e-15]
[0.00000000e+00 9.99998748e-01 1.00000000e+00 9.99999762e-01
 3.85836620e-35]
[4.22170980e-20 9.99561369e-01 1.00000000e+00 9.81739342e-01
 4.12576535e-15]
[1.66507568e-26 9.99931633e-01 1.00000000e+00 9.98146713e-01
 4.54349276e-20]
[3.62444878e-27 9.99925792e-01 1.00000000e+00 9.99045849e-01
 1.13003325e-20]
[1.68223509e-36 9.99983370e-01 1.00000000e+00 9.99994993e-01
 2.64071910e-28]
[5.30383834e-29 9.99964297e-01 1.00000000e+00 9.99328077e-01
 5.00142257e-22]
[3.62007116e-18 9.99519646e-01 1.00000000e+00 9.28055584e-01
 1.85965501e-13]
[0.00000000e+00 9.99991119e-01 1.00000000e+00 9.99997735e-01
 5.16402342e-30]
[0.00000000e+00 9.99999583e-01 1.00000000e+00 1.00000000e+00
 0.00000000e+00]
[1.44944200e-28 9.99947906e-01 1.00000000e+00 9.99467492e-01
 8.98250190e-22]
[1.87472629e-32 9.99971807e-01 1.00000000e+00 9.99940932e-01
 5.90402040e-25]
[3.73780668e-18 9.99312997e-01 1.00000000e+00 9.57015157e-01
 1.45170239e-13]
[0.00000000e+00 9.99997020e-01 1.00000000e+00 9.99999821e-01
 2.43306185e-34]
[5.70136636e-18 9.99152124e-01 1.00000000e+00 9.63798642e-01
 1.78341554e-13]
[2.28939168e-28 9.99938011e-01 1.00000000e+00 9.99522924e-01
 1.16982867e-21]
[3.12026881e-25 9.99852717e-01 1.00000000e+00 9.98422921e-01
 3.23349270e-19]
[5.67497099e-14 9.97107506e-01 1.00000000e+00 8.75428379e-01
 2.10690659e-10]
[2.87820508e-23 9.99799490e-01 1.00000000e+00 9.95208144e-01
 1.31368681e-17]
[0.00000000e+00 9.99994159e-01 1.00000000e+00 9.99996305e-01
 4.94996496e-30]
```

```
encoded_train = encoder.predict(x_train)
decoded_train = decoder.predict(encoded_train)
```

```
13/13 [==============================] - 0s 2ms/step
13/13 [==============================] - 0s 2ms/step
```

```
from sklearn.metrics import mean_squared_error
print(mean_squared_error(x_test,decoded_inp))
```

```
103704.05916784401
```

```
model = Sequential()
model.add(Dense(5,input_dim=5))
model.add(Dense(5,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
optimizer=SGD(learning_rate=0.01,momentum=0.1)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accuracy'])
```

```
model.fit(decoded_train,y_train,epochs=30)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 0.6675 - accuracy: 0.6181
Epoch 3/30
13/13 [==============================] - 0s 3ms/step - loss: 0.6661 - accuracy: 0.6181
Epoch 4/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6655 - accuracy: 0.6181
Epoch 5/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6655 - accuracy: 0.6181
Epoch 6/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6654 - accuracy: 0.6181
Epoch 7/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6655 - accuracy: 0.6181
Epoch 8/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6654 - accuracy: 0.6181
Epoch 9/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6655 - accuracy: 0.6181
Epoch 10/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6656 - accuracy: 0.6181
Epoch 11/30
13/13 [==============================] - 0s 3ms/step - loss: 0.6652 - accuracy: 0.6181
Epoch 12/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6655 - accuracy: 0.6181
Epoch 13/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6653 - accuracy: 0.6181
Epoch 14/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6652 - accuracy: 0.6181
Epoch 15/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6652 - accuracy: 0.6181
Epoch 16/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6652 - accuracy: 0.6181
Epoch 17/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6654 - accuracy: 0.6181
Epoch 18/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6653 - accuracy: 0.6181
Epoch 19/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6650 - accuracy: 0.6181
Epoch 20/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6652 - accuracy: 0.6181
Epoch 21/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6650 - accuracy: 0.6181
Epoch 22/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6650 - accuracy: 0.6181
Epoch 23/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6653 - accuracy: 0.6181
Epoch 24/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6648 - accuracy: 0.6181
Epoch 25/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6651 - accuracy: 0.6181
Epoch 26/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6653 - accuracy: 0.6181
Epoch 27/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6650 - accuracy: 0.6181
Epoch 28/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6651 - accuracy: 0.6181
Epoch 29/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6649 - accuracy: 0.6181
Epoch 30/30
13/13 [==============================] - 0s 2ms/step - loss: 0.6651 - accuracy: 0.6181
<keras.src.callbacks.History at 0x78fabc3a6d10>
```

```
pred_auto=model.predict(decoded_inp)
pred_auto=(pred_auto>0.5)
from sklearn.metrics import accuracy_score
print(accuracy_score(pred_auto,y_test))
```

```
6/6 [==============================] - 0s 4ms/step
0.6491228070175439
```