Introducción

El presente documento tiene como principal objetivo construir una red neuronal de clasificación binaria para predecir el cancer de seno usando los datos Breast Cancer Wisconsin. En este documento se presenta inicialmente una exploración de los datos y analisis de los mismos, al final se desarrollara el módelo de red neuronal de clasificación binaria para predecir el cancer de seno.

Inicialmente se lee el dataset.

In [1]:

```
import pandas as pd
data = pd.read_csv('./data.csv')
```

Ahora se observa superficialmente el contenido del dataset, se observa que el archivo contiene 33 columnas (33 posibles variables). La columna `id` no sigue una secuencia por lo cual no es muy util para el modelo final. La columna `diagnosis` es la columna que nos da la información de si esa fila corresponde al diagnóstico donde 'B' significa tumor benigno y 'M' tumor maligno. La última columna `Unnamed: 32` solo tiene valores "NaN" por lo cual no es útil.

In [2]:

```
data.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 33 columns

In [3]:

```
col = data.columns
print(col)
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

Se eliminan las columnas `id` `y` `diagnosis`

In [4]:

```
data = data.drop(['id', 'Unnamed: 32'],axis = 1 )
```

```
data.head()
```

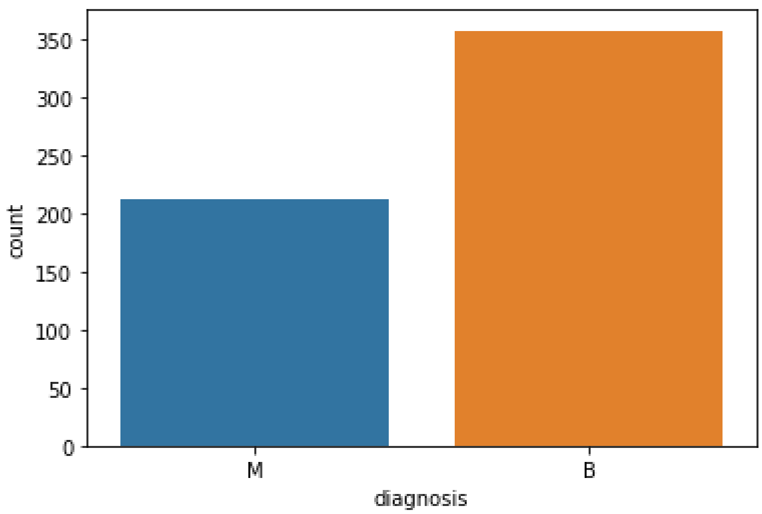
Out[4]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

Se observa la cantidad de diagnósticos para tumores malignos y benignos.

In [5]:
```
import seaborn as sns
ax = sns.countplot(x = data['diagnosis'],label="Count")
```



In [6]:
```
benigno, maligno = data['diagnosis'].value_counts()
print('Número de diagnósticos con tumor maligno: ', maligno)
print('Número de diagnósticos con tumor benigno: ', benigno)
```

```
Número de diagnósticos con tumor maligno:  212
Número de diagnósticos con tumor benigno:  357
```

Cambiar valores `M` y `B` por 1 y 0 respectivamente.

In [7]:
```
data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})
data.head()
```

Out[7]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

Descripción general de las variables:

In [8]:

```
data.describe()
```

Out[8]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	con
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 31 columns

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   diagnosis                569 non-null    int64  
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
```

```
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

Ahora se van a gráficar los histogramas de los valores que representan una media, y serán estas variables las que se tendrán en cuenta ya que las otras columnas (se - standard error y worst - valor mas alto de las medias) no son útiles en este caso. Estos histogramas se contrastaran respecto a los valores correspondientes de tumores malignos y benignos.

Columnas con valores que representan medias.

In [10]:

```
features_mean=list(data.columns[1:11])
features_mean
```

Out[10]:

```
['radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean']
```

In [11]:

```
import matplotlib.pyplot as plt
import numpy as np

malign = data[data['diagnosis'] == 1]
benign = data[data['diagnosis'] == 0]
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(16,20))
axes = axes.ravel()
for index,axe in enumerate(axes):
    axe.hist([malign[features_mean[index]],benign[features_mean[index]]],bins=5(
    axe.legend(loc='upper right')
    axe.set_title(features_mean[index])
plt.tight_layout()
plt.show()
```

De acuerdo a los histogramas anteriores, se pueden observar los siguientes comportamientos:

1. Las columnas de `radius` , `perimeter` , `area` , `compactness` , `concavity y concave points` presentan valores altos en los datos correspondientes a tumores malignos, por lo cual son útiles para el modelo.

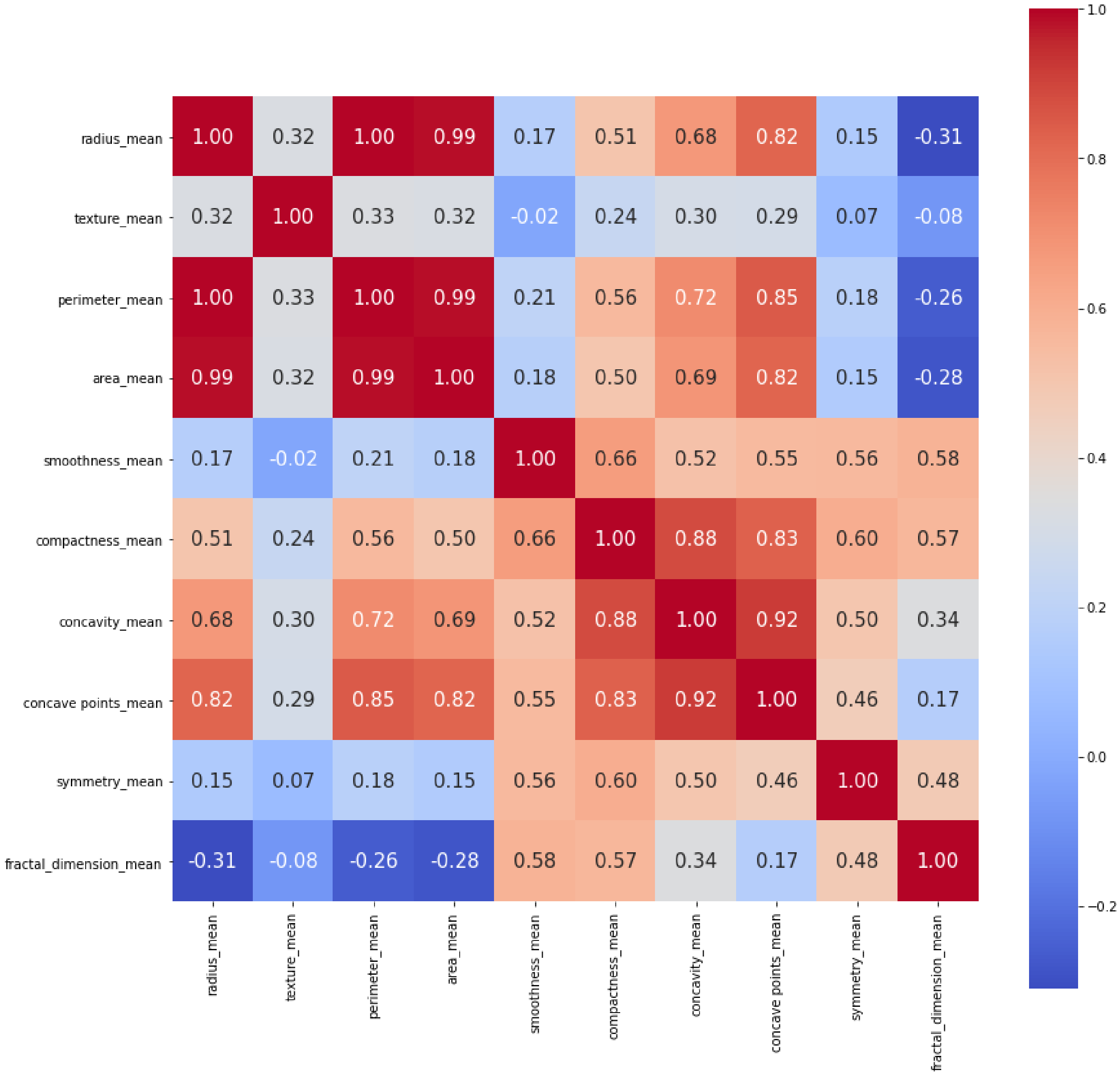
2. Las columnas de `texture` , `smoothness` , `symmetry y fractal_dimension` presentan datos muy parecidos tanto para tumores malignos como para tumores benignos por

lo cual no resultan muy útiles para el modelo.

Ahora se va analizar la correlación entre las variables mediante un mapa de calor

In [12]:
```
correlation = data[features_mean].corr()
plt.figure(figsize=(14,14))
sns.heatmap(correlation, cbar = True,  square = True, annot=True, fmt= '.2f',anr
            xticklabels= features_mean, yticklabels= features_mean,
            cmap= 'coolwarm')
```

Out[12]: <AxesSubplot:>



Con relación a este mapa de calor se observa una alta correlación entre algunas variables por lo cual esas correlaciones se dejará solamente una de las variables de la siguiente manera:

1. Las columnas de `radius` , `perimeter y area` presentan una alta correlación.
2. Las columnas de `compactness` , `concavity y concave points` presentan una alta correlación.

3. Se seleccionan las columnas de `radius y compactness` dado que de los dos grupos de variables correlacionadas, estas 2 variables son las que menos correlación tienen.

In [13]:
```
prediction_var = ['area_mean','compactness_mean']
```

Ahora se dividen los datos para entrenamiento y pruebas.

In [16]:
```
from sklearn.model_selection import train_test_split

train, test = train_test_split(data, test_size = 0.3)

train_x = train[prediction_var]
train_y = train.diagnosis

test_x = test[prediction_var]
test_y = test.diagnosis
```

Normalización de los datos para mejorar los resultados del modelo.

In [15]:
```
from sklearn.preprocessing import MinMaxScaler, Normalizer

train_x = MinMaxScaler().fit_transform(train_x)
print("Datos entrenamiento:", train_x.shape)

test_x = MinMaxScaler().fit_transform(test_x)
print("Datos pruebas:", test_x.shape)
```

```
Datos entrenamiento: (398, 2)
Datos pruebas: (171, 2)
```

El modelo que se va plantear a continuación se desarrollo usando como base otros cuadernos de Kaggle y los cuadernos de clase, adicionalmente los parámetros se ajustaron mediante ensayo y error.

In [37]:
```
import keras
import tensorflow as tf
from keras.models import Model
from keras.layers import Dense, Dropout, Input, Activation
from tensorflow.keras.utils import plot_model
```

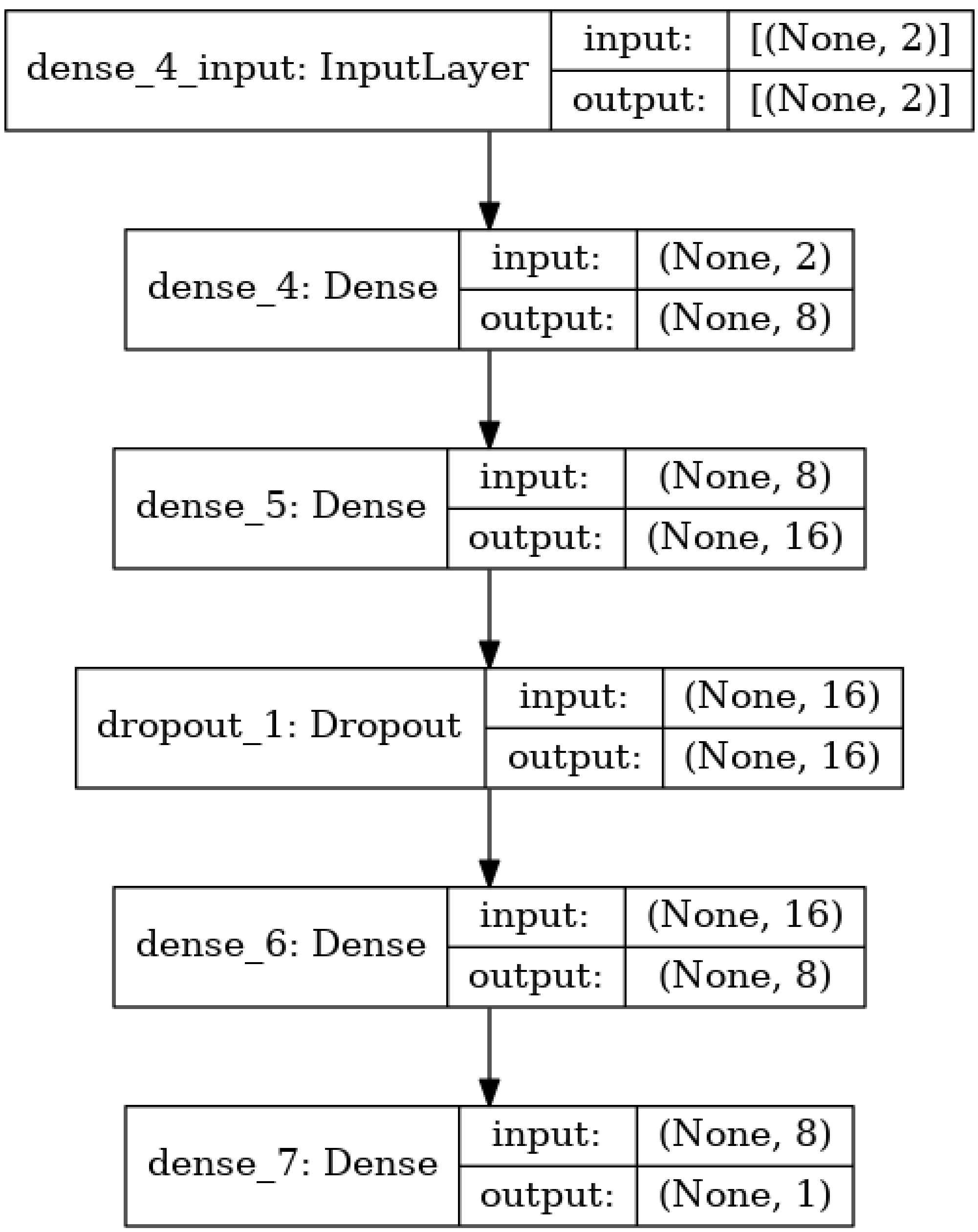
In [38]:
```
model = keras.Sequential([
    Dense(8, activation='relu', input_shape=(train_x.shape[1],)),
    Dense(16, activation='relu'),
    Dropout(0.2),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

In [39]:

```python
model.summary()
plot_model(model, to_file='./img/cancer.png',
           show_shapes=True)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 8)                 24
_________________________________________________________________
dense_5 (Dense)              (None, 16)                144
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0
_________________________________________________________________
dense_6 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 9
=================================================================
Total params: 313
Trainable params: 313
Non-trainable params: 0
_________________________________________________________________
```

Out[39]:



In [40]:

```
from tensorflow.keras. metrics import Precision, Recall
model.compile(optimizer='adam', loss="binary_crossentropy", metrics=[Precision(r
```

In [41]:

```
history = model.fit(
    train_x,
    train_y,
    batch_size=128,
    epochs=150,
    verbose=2,
    validation_data=(test_x, test_y)
)
```

```
Epoch 1/150
4/4 - 2s - loss: 0.6932 - precision: 0.4717 - accuracy: 0.6281 - val_loss: 0.694
5 - val_precision: 0.0000e+00 - val_accuracy: 0.6082
Epoch 2/150
4/4 - 0s - loss: 0.6909 - precision: 0.7241 - accuracy: 0.6683 - val_loss: 0.692
6 - val_precision: 0.0000e+00 - val_accuracy: 0.6082
Epoch 3/150
4/4 - 0s - loss: 0.6894 - precision: 0.8824 - accuracy: 0.6683 - val_loss: 0.690
9 - val_precision: 1.0000 - val_accuracy: 0.6140
Epoch 4/150
4/4 - 0s - loss: 0.6883 - precision: 1.0000 - accuracy: 0.6683 - val_loss: 0.689
2 - val_precision: 1.0000 - val_accuracy: 0.6140
Epoch 5/150
4/4 - 0s - loss: 0.6867 - precision: 1.0000 - accuracy: 0.6583 - val_loss: 0.687
4 - val_precision: 1.0000 - val_accuracy: 0.6140
Epoch 6/150
4/4 - 0s - loss: 0.6848 - precision: 1.0000 - accuracy: 0.6583 - val_loss: 0.685
6 - val_precision: 1.0000 - val_accuracy: 0.6199
Epoch 7/150
4/4 - 0s - loss: 0.6819 - precision: 0.9333 - accuracy: 0.6683 - val_loss: 0.683
8 - val_precision: 1.0000 - val_accuracy: 0.6257
Epoch 8/150
4/4 - 0s - loss: 0.6793 - precision: 1.0000 - accuracy: 0.6834 - val_loss: 0.681
8 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 9/150
4/4 - 0s - loss: 0.6786 - precision: 1.0000 - accuracy: 0.6734 - val_loss: 0.679
8 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 10/150
4/4 - 0s - loss: 0.6767 - precision: 1.0000 - accuracy: 0.6784 - val_loss: 0.677
7 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 11/150
4/4 - 0s - loss: 0.6747 - precision: 1.0000 - accuracy: 0.6709 - val_loss: 0.675
6 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 12/150
4/4 - 0s - loss: 0.6723 - precision: 1.0000 - accuracy: 0.6734 - val_loss: 0.673
4 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 13/150
4/4 - 0s - loss: 0.6693 - precision: 1.0000 - accuracy: 0.6809 - val_loss: 0.671
1 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 14/150
4/4 - 0s - loss: 0.6660 - precision: 1.0000 - accuracy: 0.6859 - val_loss: 0.668
4 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 15/150
4/4 - 0s - loss: 0.6633 - precision: 1.0000 - accuracy: 0.6935 - val_loss: 0.665
4 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 16/150
4/4 - 0s - loss: 0.6595 - precision: 1.0000 - accuracy: 0.6834 - val_loss: 0.662
3 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 17/150
```

```
4/4 - 0s - loss: 0.6574 - precision: 1.0000 - accuracy: 0.6834 - val_loss: 0.658
7 - val_precision: 1.0000 - val_accuracy: 0.6316
Epoch 18/150
4/4 - 0s - loss: 0.6526 - precision: 1.0000 - accuracy: 0.6960 - val_loss: 0.654
6 - val_precision: 1.0000 - val_accuracy: 0.6491
Epoch 19/150
4/4 - 0s - loss: 0.6484 - precision: 0.9643 - accuracy: 0.7010 - val_loss: 0.650
1 - val_precision: 1.0000 - val_accuracy: 0.6491
Epoch 20/150
4/4 - 0s - loss: 0.6424 - precision: 1.0000 - accuracy: 0.7236 - val_loss: 0.645
1 - val_precision: 1.0000 - val_accuracy: 0.6667
Epoch 21/150
4/4 - 0s - loss: 0.6394 - precision: 1.0000 - accuracy: 0.7111 - val_loss: 0.640
0 - val_precision: 1.0000 - val_accuracy: 0.6725
Epoch 22/150
4/4 - 0s - loss: 0.6310 - precision: 1.0000 - accuracy: 0.7211 - val_loss: 0.634
6 - val_precision: 1.0000 - val_accuracy: 0.6842
Epoch 23/150
4/4 - 0s - loss: 0.6276 - precision: 1.0000 - accuracy: 0.7211 - val_loss: 0.629
1 - val_precision: 1.0000 - val_accuracy: 0.6842
Epoch 24/150
4/4 - 0s - loss: 0.6225 - precision: 1.0000 - accuracy: 0.7286 - val_loss: 0.623
4 - val_precision: 1.0000 - val_accuracy: 0.6901
Epoch 25/150
4/4 - 0s - loss: 0.6145 - precision: 1.0000 - accuracy: 0.7412 - val_loss: 0.617
6 - val_precision: 1.0000 - val_accuracy: 0.6959
Epoch 26/150
4/4 - 0s - loss: 0.6064 - precision: 1.0000 - accuracy: 0.7613 - val_loss: 0.611
4 - val_precision: 1.0000 - val_accuracy: 0.7018
Epoch 27/150
4/4 - 0s - loss: 0.5986 - precision: 1.0000 - accuracy: 0.7513 - val_loss: 0.605
1 - val_precision: 1.0000 - val_accuracy: 0.7076
Epoch 28/150
4/4 - 0s - loss: 0.5960 - precision: 1.0000 - accuracy: 0.7387 - val_loss: 0.598
5 - val_precision: 1.0000 - val_accuracy: 0.7076
Epoch 29/150
4/4 - 0s - loss: 0.5890 - precision: 0.9767 - accuracy: 0.7387 - val_loss: 0.591
7 - val_precision: 1.0000 - val_accuracy: 0.7193
Epoch 30/150
4/4 - 0s - loss: 0.5837 - precision: 1.0000 - accuracy: 0.7613 - val_loss: 0.585
0 - val_precision: 1.0000 - val_accuracy: 0.7310
Epoch 31/150
4/4 - 0s - loss: 0.5723 - precision: 1.0000 - accuracy: 0.7538 - val_loss: 0.577
6 - val_precision: 1.0000 - val_accuracy: 0.7427
Epoch 32/150
4/4 - 0s - loss: 0.5703 - precision: 1.0000 - accuracy: 0.7613 - val_loss: 0.569
6 - val_precision: 1.0000 - val_accuracy: 0.7485
Epoch 33/150
4/4 - 0s - loss: 0.5613 - precision: 1.0000 - accuracy: 0.7864 - val_loss: 0.561
5 - val_precision: 1.0000 - val_accuracy: 0.7953
Epoch 34/150
4/4 - 0s - loss: 0.5548 - precision: 1.0000 - accuracy: 0.7839 - val_loss: 0.553
4 - val_precision: 1.0000 - val_accuracy: 0.7953
Epoch 35/150
4/4 - 0s - loss: 0.5360 - precision: 1.0000 - accuracy: 0.8191 - val_loss: 0.545
1 - val_precision: 1.0000 - val_accuracy: 0.7953
Epoch 36/150
4/4 - 0s - loss: 0.5347 - precision: 1.0000 - accuracy: 0.8216 - val_loss: 0.536
4 - val_precision: 1.0000 - val_accuracy: 0.8012
Epoch 37/150
4/4 - 0s - loss: 0.5303 - precision: 1.0000 - accuracy: 0.8015 - val_loss: 0.527
3 - val_precision: 0.9706 - val_accuracy: 0.7953
Epoch 38/150
4/4 - 0s - loss: 0.5222 - precision: 1.0000 - accuracy: 0.8065 - val_loss: 0.518
2 - val_precision: 0.9706 - val_accuracy: 0.7953
```

```
Epoch 39/150
4/4 - 0s - loss: 0.5160 - precision: 0.9740 - accuracy: 0.8191 - val_loss: 0.509
1 - val_precision: 0.9714 - val_accuracy: 0.8012
Epoch 40/150
4/4 - 0s - loss: 0.5065 - precision: 0.9863 - accuracy: 0.8141 - val_loss: 0.499
8 - val_precision: 0.9722 - val_accuracy: 0.8070
Epoch 41/150
4/4 - 0s - loss: 0.5001 - precision: 0.9878 - accuracy: 0.8367 - val_loss: 0.490
5 - val_precision: 0.9730 - val_accuracy: 0.8129
Epoch 42/150
4/4 - 0s - loss: 0.4837 - precision: 0.9765 - accuracy: 0.8392 - val_loss: 0.481
5 - val_precision: 0.9730 - val_accuracy: 0.8129
Epoch 43/150
4/4 - 0s - loss: 0.4787 - precision: 0.9878 - accuracy: 0.8367 - val_loss: 0.472
5 - val_precision: 0.9730 - val_accuracy: 0.8129
Epoch 44/150
4/4 - 0s - loss: 0.4727 - precision: 0.9639 - accuracy: 0.8291 - val_loss: 0.462
5 - val_precision: 0.9730 - val_accuracy: 0.8129
Epoch 45/150
4/4 - 0s - loss: 0.4542 - precision: 1.0000 - accuracy: 0.8593 - val_loss: 0.451
8 - val_precision: 0.9756 - val_accuracy: 0.8363
Epoch 46/150
4/4 - 0s - loss: 0.4604 - precision: 0.9778 - accuracy: 0.8518 - val_loss: 0.441
2 - val_precision: 0.9767 - val_accuracy: 0.8480
Epoch 47/150
4/4 - 0s - loss: 0.4454 - precision: 0.9588 - accuracy: 0.8593 - val_loss: 0.431
1 - val_precision: 0.9787 - val_accuracy: 0.8713
Epoch 48/150
4/4 - 0s - loss: 0.4330 - precision: 0.9806 - accuracy: 0.8844 - val_loss: 0.421
2 - val_precision: 0.9600 - val_accuracy: 0.8772
Epoch 49/150
4/4 - 0s - loss: 0.4228 - precision: 0.9423 - accuracy: 0.8668 - val_loss: 0.411
2 - val_precision: 0.9608 - val_accuracy: 0.8830
Epoch 50/150
4/4 - 0s - loss: 0.4199 - precision: 0.9327 - accuracy: 0.8618 - val_loss: 0.401
4 - val_precision: 0.9608 - val_accuracy: 0.8830
Epoch 51/150
4/4 - 0s - loss: 0.4141 - precision: 0.9684 - accuracy: 0.8593 - val_loss: 0.391
7 - val_precision: 0.9608 - val_accuracy: 0.8830
Epoch 52/150
4/4 - 0s - loss: 0.4119 - precision: 0.9208 - accuracy: 0.8492 - val_loss: 0.382
7 - val_precision: 0.9608 - val_accuracy: 0.8830
Epoch 53/150
4/4 - 0s - loss: 0.3966 - precision: 0.9720 - accuracy: 0.8894 - val_loss: 0.373
4 - val_precision: 0.9630 - val_accuracy: 0.9006
Epoch 54/150
4/4 - 0s - loss: 0.4032 - precision: 0.9608 - accuracy: 0.8719 - val_loss: 0.364
6 - val_precision: 0.9464 - val_accuracy: 0.9006
Epoch 55/150
4/4 - 0s - loss: 0.3791 - precision: 0.9444 - accuracy: 0.8769 - val_loss: 0.356
3 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 56/150
4/4 - 0s - loss: 0.3779 - precision: 0.9259 - accuracy: 0.8668 - val_loss: 0.348
3 - val_precision: 0.9474 - val_accuracy: 0.9064
Epoch 57/150
4/4 - 0s - loss: 0.3657 - precision: 0.9099 - accuracy: 0.8643 - val_loss: 0.340
8 - val_precision: 0.9474 - val_accuracy: 0.9064
Epoch 58/150
4/4 - 0s - loss: 0.3447 - precision: 0.9375 - accuracy: 0.8819 - val_loss: 0.333
4 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 59/150
4/4 - 0s - loss: 0.3578 - precision: 0.9174 - accuracy: 0.8643 - val_loss: 0.326
1 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 60/150
4/4 - 0s - loss: 0.3399 - precision: 0.9322 - accuracy: 0.8920 - val_loss: 0.319
```

```
2 - val_precision: 0.9322 - val_accuracy: 0.9064
Epoch 61/150
4/4 - 0s - loss: 0.3310 - precision: 0.9333 - accuracy: 0.8970 - val_loss: 0.312
1 - val_precision: 0.9322 - val_accuracy: 0.9064
Epoch 62/150
4/4 - 0s - loss: 0.3406 - precision: 0.9076 - accuracy: 0.8794 - val_loss: 0.305
4 - val_precision: 0.9322 - val_accuracy: 0.9064
Epoch 63/150
4/4 - 0s - loss: 0.3268 - precision: 0.9180 - accuracy: 0.8920 - val_loss: 0.298
7 - val_precision: 0.9322 - val_accuracy: 0.9064
Epoch 64/150
4/4 - 0s - loss: 0.3331 - precision: 0.9130 - accuracy: 0.8744 - val_loss: 0.292
9 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 65/150
4/4 - 0s - loss: 0.3273 - precision: 0.9310 - accuracy: 0.8869 - val_loss: 0.287
6 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 66/150
4/4 - 0s - loss: 0.3086 - precision: 0.9250 - accuracy: 0.8920 - val_loss: 0.282
8 - val_precision: 0.9483 - val_accuracy: 0.9123
Epoch 67/150
4/4 - 0s - loss: 0.3101 - precision: 0.9098 - accuracy: 0.8869 - val_loss: 0.278
3 - val_precision: 0.9322 - val_accuracy: 0.9064
Epoch 68/150
4/4 - 0s - loss: 0.3153 - precision: 0.9167 - accuracy: 0.8869 - val_loss: 0.274
2 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 69/150
4/4 - 0s - loss: 0.3132 - precision: 0.9091 - accuracy: 0.8844 - val_loss: 0.269
8 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 70/150
4/4 - 0s - loss: 0.3020 - precision: 0.8880 - accuracy: 0.8794 - val_loss: 0.265
1 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 71/150
4/4 - 0s - loss: 0.3143 - precision: 0.8926 - accuracy: 0.8744 - val_loss: 0.260
7 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 72/150
4/4 - 0s - loss: 0.3028 - precision: 0.9316 - accuracy: 0.8894 - val_loss: 0.257
0 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 73/150
4/4 - 0s - loss: 0.2919 - precision: 0.8974 - accuracy: 0.8693 - val_loss: 0.253
3 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 74/150
4/4 - 0s - loss: 0.3046 - precision: 0.8852 - accuracy: 0.8719 - val_loss: 0.250
0 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 75/150
4/4 - 0s - loss: 0.2978 - precision: 0.8750 - accuracy: 0.8769 - val_loss: 0.246
8 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 76/150
4/4 - 0s - loss: 0.2806 - precision: 0.9091 - accuracy: 0.8844 - val_loss: 0.243
7 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 77/150
4/4 - 0s - loss: 0.2935 - precision: 0.8898 - accuracy: 0.8668 - val_loss: 0.240
9 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 78/150
4/4 - 0s - loss: 0.2943 - precision: 0.8750 - accuracy: 0.8769 - val_loss: 0.239
3 - val_precision: 0.9180 - val_accuracy: 0.9064
Epoch 79/150
4/4 - 0s - loss: 0.2981 - precision: 0.8943 - accuracy: 0.8794 - val_loss: 0.238
7 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 80/150
4/4 - 0s - loss: 0.2821 - precision: 0.8828 - accuracy: 0.8819 - val_loss: 0.238
9 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 81/150
4/4 - 0s - loss: 0.2714 - precision: 0.9030 - accuracy: 0.9070 - val_loss: 0.236
7 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 82/150
```

```
4/4 - 0s - loss: 0.2680 - precision: 0.8976 - accuracy: 0.8894 - val_loss: 0.232
5 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 83/150
4/4 - 0s - loss: 0.2910 - precision: 0.8800 - accuracy: 0.8744 - val_loss: 0.228
8 - val_precision: 0.9180 - val_accuracy: 0.9064
Epoch 84/150
4/4 - 0s - loss: 0.2681 - precision: 0.9322 - accuracy: 0.8920 - val_loss: 0.226
7 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 85/150
4/4 - 0s - loss: 0.2768 - precision: 0.9174 - accuracy: 0.8894 - val_loss: 0.224
9 - val_precision: 0.9333 - val_accuracy: 0.9123
Epoch 86/150
4/4 - 0s - loss: 0.2874 - precision: 0.8828 - accuracy: 0.8819 - val_loss: 0.223
5 - val_precision: 0.9180 - val_accuracy: 0.9064
Epoch 87/150
4/4 - 0s - loss: 0.2820 - precision: 0.8740 - accuracy: 0.8744 - val_loss: 0.225
1 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 88/150
4/4 - 0s - loss: 0.2723 - precision: 0.8788 - accuracy: 0.8869 - val_loss: 0.228
1 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 89/150
4/4 - 0s - loss: 0.2701 - precision: 0.8963 - accuracy: 0.9045 - val_loss: 0.227
8 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 90/150
4/4 - 0s - loss: 0.2604 - precision: 0.8815 - accuracy: 0.8945 - val_loss: 0.223
8 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 91/150
4/4 - 0s - loss: 0.2705 - precision: 0.9015 - accuracy: 0.9020 - val_loss: 0.220
9 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 92/150
4/4 - 0s - loss: 0.2576 - precision: 0.8889 - accuracy: 0.8995 - val_loss: 0.218
5 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 93/150
4/4 - 0s - loss: 0.2591 - precision: 0.9048 - accuracy: 0.8920 - val_loss: 0.216
3 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 94/150
4/4 - 0s - loss: 0.2530 - precision: 0.8984 - accuracy: 0.8920 - val_loss: 0.214
8 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 95/150
4/4 - 0s - loss: 0.2548 - precision: 0.9048 - accuracy: 0.8920 - val_loss: 0.214
5 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 96/150
4/4 - 0s - loss: 0.2607 - precision: 0.8797 - accuracy: 0.8894 - val_loss: 0.214
7 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 97/150
4/4 - 0s - loss: 0.2695 - precision: 0.8686 - accuracy: 0.8894 - val_loss: 0.214
0 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 98/150
4/4 - 0s - loss: 0.2640 - precision: 0.9077 - accuracy: 0.9020 - val_loss: 0.212
6 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 99/150
4/4 - 0s - loss: 0.2634 - precision: 0.9048 - accuracy: 0.8920 - val_loss: 0.211
3 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 100/150
4/4 - 0s - loss: 0.2771 - precision: 0.8712 - accuracy: 0.8819 - val_loss: 0.210
7 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 101/150
4/4 - 0s - loss: 0.2485 - precision: 0.9154 - accuracy: 0.9070 - val_loss: 0.212
5 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 102/150
4/4 - 0s - loss: 0.2505 - precision: 0.8777 - accuracy: 0.8995 - val_loss: 0.210
5 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 103/150
4/4 - 0s - loss: 0.2539 - precision: 0.8955 - accuracy: 0.9020 - val_loss: 0.208
6 - val_precision: 0.9032 - val_accuracy: 0.9006
```

```
Epoch 104/150
4/4 - 0s - loss: 0.2569 - precision: 0.8897 - accuracy: 0.9020 - val_loss: 0.206
5 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 105/150
4/4 - 0s - loss: 0.2532 - precision: 0.9062 - accuracy: 0.8970 - val_loss: 0.205
7 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 106/150
4/4 - 0s - loss: 0.2497 - precision: 0.9040 - accuracy: 0.8894 - val_loss: 0.205
8 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 107/150
4/4 - 0s - loss: 0.2575 - precision: 0.9070 - accuracy: 0.8995 - val_loss: 0.206
7 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 108/150
4/4 - 0s - loss: 0.2540 - precision: 0.9030 - accuracy: 0.9070 - val_loss: 0.208
8 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 109/150
4/4 - 0s - loss: 0.2585 - precision: 0.8955 - accuracy: 0.9020 - val_loss: 0.211
2 - val_precision: 0.9104 - val_accuracy: 0.9298
Epoch 110/150
4/4 - 0s - loss: 0.2596 - precision: 0.8786 - accuracy: 0.9020 - val_loss: 0.210
1 - val_precision: 0.9104 - val_accuracy: 0.9298
Epoch 111/150
4/4 - 0s - loss: 0.2393 - precision: 0.8929 - accuracy: 0.9121 - val_loss: 0.206
6 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 112/150
4/4 - 0s - loss: 0.2405 - precision: 0.8947 - accuracy: 0.8995 - val_loss: 0.204
4 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 113/150
4/4 - 0s - loss: 0.2592 - precision: 0.8889 - accuracy: 0.8995 - val_loss: 0.203
9 - val_precision: 0.9062 - val_accuracy: 0.9123
Epoch 114/150
4/4 - 0s - loss: 0.2487 - precision: 0.9077 - accuracy: 0.9020 - val_loss: 0.201
9 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 115/150
4/4 - 0s - loss: 0.2545 - precision: 0.8855 - accuracy: 0.8894 - val_loss: 0.200
5 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 116/150
4/4 - 0s - loss: 0.2487 - precision: 0.8984 - accuracy: 0.8920 - val_loss: 0.199
8 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 117/150
4/4 - 0s - loss: 0.2486 - precision: 0.8806 - accuracy: 0.8920 - val_loss: 0.200
6 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 118/150
4/4 - 0s - loss: 0.2465 - precision: 0.8872 - accuracy: 0.8945 - val_loss: 0.200
1 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 119/150
4/4 - 0s - loss: 0.2547 - precision: 0.8889 - accuracy: 0.8995 - val_loss: 0.198
2 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 120/150
4/4 - 0s - loss: 0.2632 - precision: 0.8682 - accuracy: 0.8744 - val_loss: 0.196
6 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 121/150
4/4 - 0s - loss: 0.2667 - precision: 0.9024 - accuracy: 0.8844 - val_loss: 0.196
1 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 122/150
4/4 - 0s - loss: 0.2552 - precision: 0.9062 - accuracy: 0.8970 - val_loss: 0.196
3 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 123/150
4/4 - 0s - loss: 0.2367 - precision: 0.9219 - accuracy: 0.9070 - val_loss: 0.197
8 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 124/150
4/4 - 0s - loss: 0.2425 - precision: 0.8947 - accuracy: 0.8995 - val_loss: 0.199
2 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 125/150
4/4 - 0s - loss: 0.2486 - precision: 0.9008 - accuracy: 0.8995 - val_loss: 0.200
```

```
1 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 126/150
4/4 - 0s - loss: 0.2555 - precision: 0.8815 - accuracy: 0.8945 - val_loss: 0.199
8 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 127/150
4/4 - 0s - loss: 0.2333 - precision: 0.8849 - accuracy: 0.9045 - val_loss: 0.199
7 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 128/150
4/4 - 0s - loss: 0.2467 - precision: 0.8643 - accuracy: 0.8920 - val_loss: 0.199
2 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 129/150
4/4 - 0s - loss: 0.2431 - precision: 0.8889 - accuracy: 0.8995 - val_loss: 0.198
2 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 130/150
4/4 - 0s - loss: 0.2688 - precision: 0.8769 - accuracy: 0.8819 - val_loss: 0.196
7 - val_precision: 0.9091 - val_accuracy: 0.9240
Epoch 131/150
4/4 - 0s - loss: 0.2656 - precision: 0.8846 - accuracy: 0.8869 - val_loss: 0.196
5 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 132/150
4/4 - 0s - loss: 0.2471 - precision: 0.9055 - accuracy: 0.8945 - val_loss: 0.195
2 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 133/150
4/4 - 0s - loss: 0.2499 - precision: 0.8832 - accuracy: 0.8995 - val_loss: 0.194
8 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 134/150
4/4 - 0s - loss: 0.2435 - precision: 0.9000 - accuracy: 0.8970 - val_loss: 0.194
0 - val_precision: 0.9032 - val_accuracy: 0.9006
Epoch 135/150
4/4 - 0s - loss: 0.2630 - precision: 0.9055 - accuracy: 0.8945 - val_loss: 0.194
2 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 136/150
4/4 - 0s - loss: 0.2533 - precision: 0.9024 - accuracy: 0.8844 - val_loss: 0.194
6 - val_precision: 0.9048 - val_accuracy: 0.9064
Epoch 137/150
4/4 - 0s - loss: 0.2398 - precision: 0.9160 - accuracy: 0.9095 - val_loss: 0.196
0 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 138/150
4/4 - 0s - loss: 0.2341 - precision: 0.8955 - accuracy: 0.9020 - val_loss: 0.197
2 - val_precision: 0.8955 - val_accuracy: 0.9181
Epoch 139/150
4/4 - 0s - loss: 0.2485 - precision: 0.8864 - accuracy: 0.8920 - val_loss: 0.200
4 - val_precision: 0.8857 - val_accuracy: 0.9240
Epoch 140/150
4/4 - 0s - loss: 0.2433 - precision: 0.8582 - accuracy: 0.8894 - val_loss: 0.204
3 - val_precision: 0.8889 - val_accuracy: 0.9357
Epoch 141/150
4/4 - 0s - loss: 0.2512 - precision: 0.9051 - accuracy: 0.9146 - val_loss: 0.206
6 - val_precision: 0.8767 - val_accuracy: 0.9298
Epoch 142/150
4/4 - 0s - loss: 0.2443 - precision: 0.8794 - accuracy: 0.9045 - val_loss: 0.205
2 - val_precision: 0.8767 - val_accuracy: 0.9298
Epoch 143/150
4/4 - 0s - loss: 0.2562 - precision: 0.8493 - accuracy: 0.8920 - val_loss: 0.200
0 - val_precision: 0.8857 - val_accuracy: 0.9240
Epoch 144/150
4/4 - 0s - loss: 0.2482 - precision: 0.8978 - accuracy: 0.9095 - val_loss: 0.196
9 - val_precision: 0.8955 - val_accuracy: 0.9181
Epoch 145/150
4/4 - 0s - loss: 0.2252 - precision: 0.9037 - accuracy: 0.9095 - val_loss: 0.193
7 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 146/150
4/4 - 0s - loss: 0.2400 - precision: 0.8779 - accuracy: 0.8844 - val_loss: 0.193
3 - val_precision: 0.9077 - val_accuracy: 0.9181
Epoch 147/150
```

```
4/4 - 0s - loss: 0.2260 - precision: 0.9000 - accuracy: 0.8970 - val_loss: 0.194
9 - val_precision: 0.8955 - val_accuracy: 0.9181
Epoch 148/150
4/4 - 0s - loss: 0.2257 - precision: 0.9051 - accuracy: 0.9146 - val_loss: 0.199
2 - val_precision: 0.8857 - val_accuracy: 0.9240
Epoch 149/150
4/4 - 0s - loss: 0.2407 - precision: 0.8777 - accuracy: 0.8995 - val_loss: 0.204
6 - val_precision: 0.8767 - val_accuracy: 0.9298
Epoch 150/150
4/4 - 0s - loss: 0.2331 - precision: 0.8705 - accuracy: 0.8945 - val_loss: 0.207
7 - val_precision: 0.8784 - val_accuracy: 0.9357
```

In [42]:

```
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist
```

Out[42]:

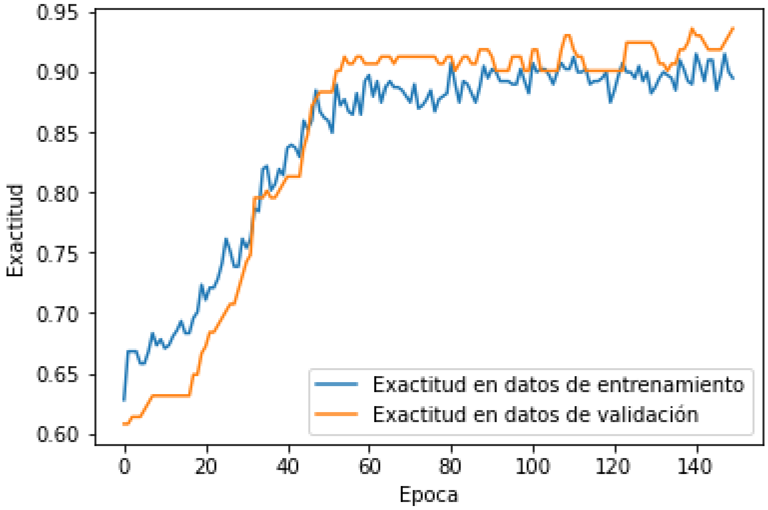
	loss	precision	accuracy	val_loss	val_precision	val_accuracy	epoch
0	0.693163	0.471698	0.628141	0.694515	0.000000	0.608187	0
1	0.690860	0.724138	0.668342	0.692641	0.000000	0.608187	1
2	0.689422	0.882353	0.668342	0.690940	1.000000	0.614035	2
3	0.688251	1.000000	0.668342	0.689189	1.000000	0.614035	3
4	0.686662	1.000000	0.658291	0.687443	1.000000	0.614035	4
...	...	...	...	...	...	...	...
145	0.240020	0.877863	0.884422	0.193304	0.907692	0.918129	145
146	0.226005	0.900000	0.896985	0.194879	0.895522	0.918129	146
147	0.225738	0.905109	0.914573	0.199151	0.885714	0.923977	147
148	0.240670	0.877698	0.899498	0.204625	0.876712	0.929825	148
149	0.233106	0.870504	0.894472	0.207704	0.878378	0.935673	149

150 rows × 7 columns

El parámetro `epoch` se ajustó mediante ensayo y error tomando como referencia las siguientes gráficas, especialmente la gráfica de error en el entrenamiento que después de `epoch=100` , se observó que no habia mas perdida de error.

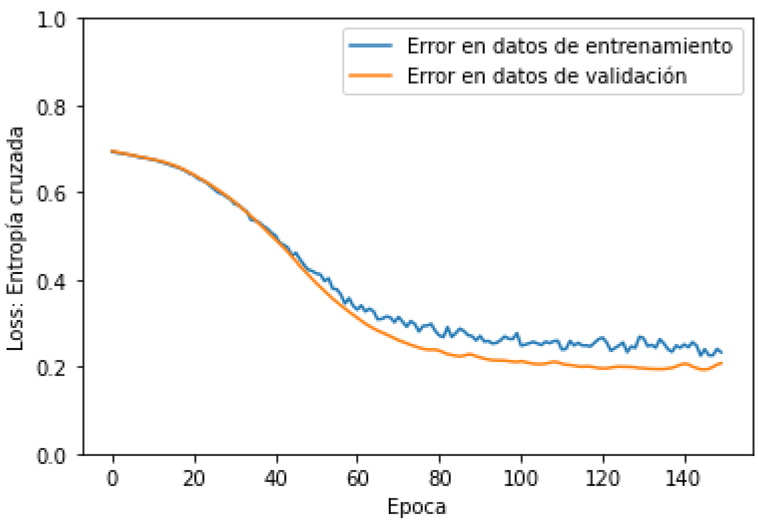
In [44]:

```
plt.figure()
plt.xlabel('Epoca')
plt.ylabel('Exactitud')
plt.plot(hist['epoch'], hist['accuracy'],
           label='Exactitud en datos de entrenamiento')
plt.plot(hist['epoch'], hist['val_accuracy'],
           label='Exactitud en datos de validación')
# plt.ylim([0.9,1])
plt.legend()
plt.show()
```



In [47]:

```
plt.figure()
plt.xlabel('Epoca')
plt.ylabel('Loss: Entropía cruzada')
plt.plot(hist['epoch'], hist['loss'],
           label='Error en datos de entrenamiento')
plt.plot(hist['epoch'], hist['val_loss'],
           label='Error en datos de validación')
plt.ylim([0,1])
plt.legend()
plt.show()
```



In [48]:

```
pred_y = model.predict(test_x)
pred_y = (pred_y > 0.5)
pred_test_y = model.predict(train_x)
pred_test_y = (pred_test_y > 0.5)
```

In [49]:

```
from sklearn import metrics

print(metrics.classification_report(test_y, pred_y))
```

```
              precision    recall  f1-score   support

           0       0.98      0.91      0.95       104
```

```
           1       0.88      0.97      0.92        67

    accuracy                           0.94       171
   macro avg       0.93      0.94      0.93       171
weighted avg       0.94      0.94      0.94       171
```

In [51]:

```
print("Exactitud del modelo sobre los datos de prueba: {0:0.3f}".format(metrics.
print("Exactitud del modelo sobre los datos de entrenamiento: {0:0.3f}".format(m
```

```
Exactitud del modelo sobre los datos de prueba: 0.936
Exactitud del modelo sobre los datos de entrenamiento: 0.910
```

In [52]:

```
print("Precisión del modelo sobre los datos de prueba: {0:0.3f}".format(metrics.
print("Precisión del modelo sobre los datos de entrenamiento: {0:0.3f}".format(m
```

```
Precisión del modelo sobre los datos de prueba: 0.878
Precisión del modelo sobre los datos de entrenamiento: 0.892
```