Zillow Prize: Zillow's Home Value Prediction (Zestimate)

The goal for the notebook is predicting whether the house price is above or below median value using ann.

Description of notebook: The first part of the notebook is preparing the data for the model. The second part is regarding setting up the neural network with set hyperparameters, training the model and visualizing the results. And lastly, the third part is hypertuning the model to see if the model will perform better with different hyperparameters.

1. Processing the data

1.1. Importing needed imports

In [1]:
```
import pandas as pd
```

1.2. Retrieving the data

Data explanation can be found below for each column:

- Lot Area (in sq ft)
- Overall Quality (scale from 1 to 10)
- Overall Condition (scale from 1 to 10)
- Total Basement Area (in sq ft)
- Number of Full Bathrooms
- Number of Half Bathrooms
- Number of Bedrooms above ground
- Total Number of Rooms above ground
- Number of Fireplaces
- Garage Area (in sq ft)

The last columnn of the dataset is what we would like to predict: AboveMedianPrice. For example: Is the house price above the median or not? (1 for yes and 2 for no)

In [2]:
```
# Retrieving the data
```

```python
df = pd.read_csv('housepricedata.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageArea	AboveMedianPrice
0	8450	7	5	856	2	1	3	8	0	548	1
1	9600	6	8	1262	2	0	3	6	1	460	1
2	11250	7	5	920	2	1	3	6	1	608	1
3	9550	7	5	756	1	0	3	7	1	642	0
4	14260	8	5	1145	2	1	4	9	1	836	1

In [4]:
```python
df['AboveMedianPrice'].value_counts()
```

Out[4]:
```
0    732
1    728
Name: AboveMedianPrice, dtype: int64
```

1.3. Converting the data for our machine to process

we will be converting the dataframe to an array for the machine to be able to process the data.

In [5]:
```python
dataset = df.values
dataset
```

Out[5]:
```
array([[ 8450,     7,     5, ...,     0,   548,     1],
       [ 9600,     6,     8, ...,     1,   460,     1],
       [11250,     7,     5, ...,     1,   608,     1],
       ...,
       [ 9042,     7,     9, ...,     2,   252,     1],
       [ 9717,     5,     6, ...,     0,   240,     0],
       [ 9937,     5,     6, ...,     0,   276,     0]], dtype=int64)
```

1.4. Splitting the features and prediction data

In this section we will be splitting the data into the input and output data (X & Y).

In [6]:
```python
X = dataset[:, 0:10]
```

In [7]:
```python
Y = dataset[:, -1]
```

1.5. Scaling the feautres

In this section we will be scaling the input data to have similiar input value. For example not having one input being over 1000 and another input is in a range of 5. Due to the amount being so big, will make it difficault for the initialization of the neural network.

1.5.1. Importing needed imports

In [8]:
```
from sklearn import preprocessing
```

1.5.2. Scaling the data

We will be using the min-max scaler which scales the input data so that the input features are between the values 0 and 1. Scaling down to values between 0 & 1 will help aid the training of the neural network.

In [9]:
```
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
```

In [10]:
```
X_scale
```

Out[10]:
```
array([[0.0334198 , 0.66666667, 0.5       , ..., 0.5       , 0.        ,
        0.3864598 ],
       [0.03879502, 0.55555556, 0.875     , ..., 0.33333333, 0.33333333,
        0.32440056],
       [0.04650728, 0.66666667, 0.5       , ..., 0.33333333, 0.33333333,
        0.42877292],
       ...,
       [0.03618687, 0.66666667, 1.        , ..., 0.58333333, 0.66666667,
        0.17771509],
       [0.03934189, 0.44444444, 0.625     , ..., 0.25      , 0.        ,
        0.16925247],
       [0.04037019, 0.44444444, 0.625     , ..., 0.33333333, 0.        ,
        0.19464034]])
```

1.6. Splitting the data into a train, test & validation sets

1.6.1. Importing needed imports

In [11]:
```
from sklearn.model_selection import train_test_split
```

1.6.2. Splitting the data

In [12]:
```
# 70% will be training data
# 30% will be validation and testing data
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)
```

Now we will be splitting the test and validation data

In [13]:
```
# we will be splitting the data equally here between the test and validation data
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)
```

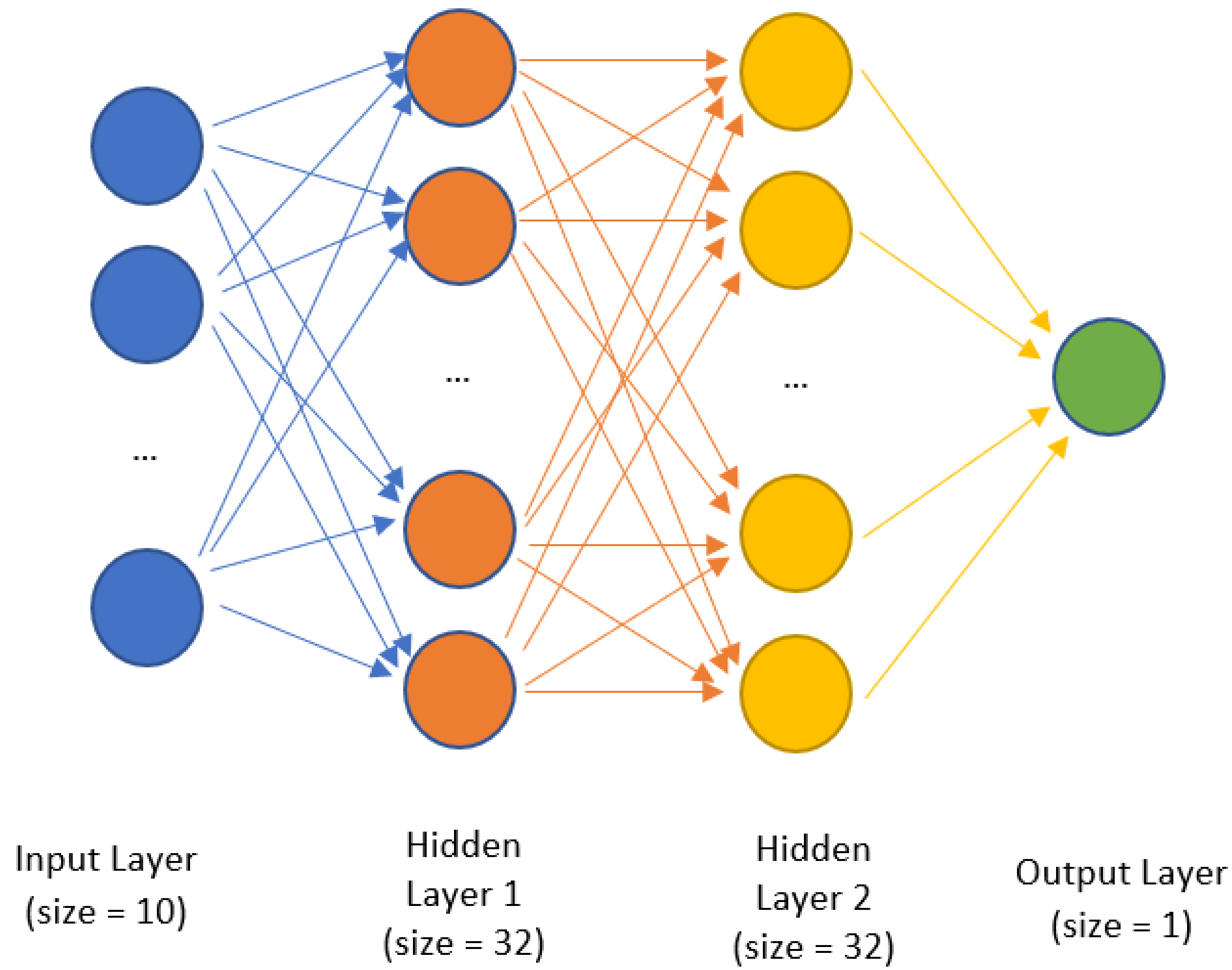
Below we can see the shapes for each data section

In [14]:
```
print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)
```

```
(1022, 10) (219, 10) (219, 10) (1022,) (219,) (219,)
```

2. Building & Training the Neural Network

2.1. Setting up the architecture



We will be setting the layers as followed:

- Hidden layer 1: 32 neurons, ReLU activation
- Hidden layer 2: 32 neurons, ReLU activation
- Output Layer: 1 neuron, Sigmoid activation

Different activation function that can be applied:

- Sigmoid Function (A function which ‘squeezes’ all the initial output to be between 0 and 1)
- tanh Function (A function which ‘squeezes’ all the initial output to be between -1 and 1)
- ReLU Function (If the initial output is negative, then output 0. If not, do nothing to the initial output)

In [15]:

```
# importing keras that will be used to set the architecture
from keras.models import Sequential
```

```
from keras.layers import Dense
```

We will be setting the model layers

In [16]:
```
model = Sequential([
    Dense(32, activation='relu', input_shape=(10,)), #32 is the size of the first layer, & 10 referes to the 10 input features
    Dense(32, activation='relu'), #32 is the size of the first layer
    Dense(1, activation='sigmoid'), #1 neuron as output layer for the prediction
])
```

2.2. Configuring & Training the Model

The architecture for the model has been set but we still need to configure the model. These configurations include:

- what optimizer will be used such as SGD & Adam.
- what lost function will be used? (first checking if it is for Probabilistic or regression losses) such as BinaryCrossentropy & CategoricalCressentropy.
- what other metrics will like to be tracker such as Accuray & MeanSquaredError.

In this section we will be conducting the followed steps:

1. Specify some hyper-parameters (the template)
2. Train on the training dataset (filling in the parameters)
3. Record the validation loss
4. Applying confusion matrix on the output
5. conclusion

In [17]:
```
model.compile(optimizer='sgd',
              loss='binary_crossentropy', #chosen for our binary output
              metrics=['accuracy'])
```

Time to train the model

In [18]:
```
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 11ms/step - loss: 0.6930 - accuracy: 0.4980 - val_loss: 0.6836 - val_accuracy: 0.5068
Epoch 2/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6814 - accuracy: 0.4980 - val_loss: 0.6743 - val_accuracy: 0.5068
Epoch 3/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6719 - accuracy: 0.5010 - val_loss: 0.6664 - val_accuracy: 0.5114
Epoch 4/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6640 - accuracy: 0.5098 - val_loss: 0.6605 - val_accuracy: 0.5571
Epoch 5/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6585 - accuracy: 0.5577 - val_loss: 0.6560 - val_accuracy: 0.6575
Epoch 6/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6537 - accuracy: 0.6526 - val_loss: 0.6517 - val_accuracy: 0.7306
Epoch 7/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6490 - accuracy: 0.7094 - val_loss: 0.6473 - val_accuracy: 0.7489
Epoch 8/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6442 - accuracy: 0.7319 - val_loss: 0.6428 - val_accuracy: 0.8037
Epoch 9/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6392 - accuracy: 0.7730 - val_loss: 0.6379 - val_accuracy: 0.8128
Epoch 10/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6339 - accuracy: 0.8072 - val_loss: 0.6328 - val_accuracy: 0.8082
Epoch 11/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6286 - accuracy: 0.7906 - val_loss: 0.6276 - val_accuracy: 0.8402
Epoch 12/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6230 - accuracy: 0.8151 - val_loss: 0.6222 - val_accuracy: 0.8447
Epoch 13/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6169 - accuracy: 0.8209 - val_loss: 0.6166 - val_accuracy: 0.8539
Epoch 14/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6108 - accuracy: 0.8493 - val_loss: 0.6105 - val_accuracy: 0.8539
Epoch 15/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6041 - accuracy: 0.8513 - val_loss: 0.6040 - val_accuracy: 0.8493
Epoch 16/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5973 - accuracy: 0.8503 - val_loss: 0.5973 - val_accuracy: 0.8447
Epoch 17/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5901 - accuracy: 0.8562 - val_loss: 0.5903 - val_accuracy: 0.8493
Epoch 18/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5825 - accuracy: 0.8532 - val_loss: 0.5827 - val_accuracy: 0.8402
Epoch 19/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5748 - accuracy: 0.8552 - val_loss: 0.5753 - val_accuracy: 0.8493
Epoch 20/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5664 - accuracy: 0.8532 - val_loss: 0.5674 - val_accuracy: 0.8539
Epoch 21/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5578 - accuracy: 0.8571 - val_loss: 0.5590 - val_accuracy: 0.8539
Epoch 22/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5491 - accuracy: 0.8611 - val_loss: 0.5505 - val_accuracy: 0.8539
Epoch 23/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5399 - accuracy: 0.8571 - val_loss: 0.5423 - val_accuracy: 0.8539
Epoch 24/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5306 - accuracy: 0.8611 - val_loss: 0.5336 - val_accuracy: 0.8539
```

```
Epoch 25/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5212 - accuracy: 0.8611 - val_loss: 0.5247 - val_accuracy: 0.8539
Epoch 26/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5116 - accuracy: 0.8581 - val_loss: 0.5154 - val_accuracy: 0.8584
Epoch 27/100
32/32 [==============================] - 0s 5ms/step - loss: 0.5022 - accuracy: 0.8630 - val_loss: 0.5065 - val_accuracy: 0.8539
Epoch 28/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4927 - accuracy: 0.8669 - val_loss: 0.4975 - val_accuracy: 0.8584
Epoch 29/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4830 - accuracy: 0.8728 - val_loss: 0.4883 - val_accuracy: 0.8630
Epoch 30/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4735 - accuracy: 0.8689 - val_loss: 0.4793 - val_accuracy: 0.8630
Epoch 31/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4642 - accuracy: 0.8708 - val_loss: 0.4707 - val_accuracy: 0.8630
Epoch 32/100
32/32 [==============================] - 0s 4ms/step - loss: 0.4553 - accuracy: 0.8650 - val_loss: 0.4631 - val_accuracy: 0.8493
Epoch 33/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4464 - accuracy: 0.8718 - val_loss: 0.4551 - val_accuracy: 0.8493
Epoch 34/100
32/32 [==============================] - 0s 4ms/step - loss: 0.4380 - accuracy: 0.8728 - val_loss: 0.4486 - val_accuracy: 0.8493
Epoch 35/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4296 - accuracy: 0.8689 - val_loss: 0.4395 - val_accuracy: 0.8584
Epoch 36/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4222 - accuracy: 0.8738 - val_loss: 0.4325 - val_accuracy: 0.8493
Epoch 37/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4145 - accuracy: 0.8738 - val_loss: 0.4263 - val_accuracy: 0.8447
Epoch 38/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4068 - accuracy: 0.8708 - val_loss: 0.4194 - val_accuracy: 0.8447
Epoch 39/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3998 - accuracy: 0.8718 - val_loss: 0.4142 - val_accuracy: 0.8493
Epoch 40/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3935 - accuracy: 0.8699 - val_loss: 0.4077 - val_accuracy: 0.8539
Epoch 41/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3871 - accuracy: 0.8689 - val_loss: 0.4017 - val_accuracy: 0.8539
Epoch 42/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3815 - accuracy: 0.8728 - val_loss: 0.3969 - val_accuracy: 0.8539
Epoch 43/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3750 - accuracy: 0.8767 - val_loss: 0.3952 - val_accuracy: 0.8539
Epoch 44/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3706 - accuracy: 0.8679 - val_loss: 0.3893 - val_accuracy: 0.8539
Epoch 45/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3649 - accuracy: 0.8689 - val_loss: 0.3821 - val_accuracy: 0.8539
Epoch 46/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3604 - accuracy: 0.8679 - val_loss: 0.3793 - val_accuracy: 0.8493
Epoch 47/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3560 - accuracy: 0.8669 - val_loss: 0.3740 - val_accuracy: 0.8539
Epoch 48/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3520 - accuracy: 0.8708 - val_loss: 0.3705 - val_accuracy: 0.8539
```

```
Epoch 49/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3474 - accuracy: 0.8748 - val_loss: 0.3690 - val_accuracy: 0.8584
Epoch 50/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3438 - accuracy: 0.8718 - val_loss: 0.3632 - val_accuracy: 0.8539
Epoch 51/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3400 - accuracy: 0.8669 - val_loss: 0.3605 - val_accuracy: 0.8539
Epoch 52/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3372 - accuracy: 0.8757 - val_loss: 0.3571 - val_accuracy: 0.8539
Epoch 53/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3336 - accuracy: 0.8748 - val_loss: 0.3557 - val_accuracy: 0.8630
Epoch 54/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3305 - accuracy: 0.8836 - val_loss: 0.3516 - val_accuracy: 0.8584
Epoch 55/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3275 - accuracy: 0.8757 - val_loss: 0.3478 - val_accuracy: 0.8584
Epoch 56/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3252 - accuracy: 0.8708 - val_loss: 0.3479 - val_accuracy: 0.8630
Epoch 57/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3231 - accuracy: 0.8816 - val_loss: 0.3447 - val_accuracy: 0.8630
Epoch 58/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3197 - accuracy: 0.8748 - val_loss: 0.3422 - val_accuracy: 0.8584
Epoch 59/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3181 - accuracy: 0.8757 - val_loss: 0.3405 - val_accuracy: 0.8630
Epoch 60/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3151 - accuracy: 0.8787 - val_loss: 0.3423 - val_accuracy: 0.8584
Epoch 61/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3139 - accuracy: 0.8816 - val_loss: 0.3395 - val_accuracy: 0.8630
Epoch 62/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3118 - accuracy: 0.8836 - val_loss: 0.3366 - val_accuracy: 0.8584
Epoch 63/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3096 - accuracy: 0.8826 - val_loss: 0.3354 - val_accuracy: 0.8630
Epoch 64/100
32/32 [==============================] - 0s 6ms/step - loss: 0.3082 - accuracy: 0.8806 - val_loss: 0.3307 - val_accuracy: 0.8539
Epoch 65/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3060 - accuracy: 0.8767 - val_loss: 0.3337 - val_accuracy: 0.8584
Epoch 66/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3044 - accuracy: 0.8845 - val_loss: 0.3274 - val_accuracy: 0.8539
Epoch 67/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3031 - accuracy: 0.8826 - val_loss: 0.3298 - val_accuracy: 0.8584
Epoch 68/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3019 - accuracy: 0.8816 - val_loss: 0.3271 - val_accuracy: 0.8630
Epoch 69/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3001 - accuracy: 0.8855 - val_loss: 0.3243 - val_accuracy: 0.8676
Epoch 70/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2991 - accuracy: 0.8836 - val_loss: 0.3258 - val_accuracy: 0.8630
Epoch 71/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2968 - accuracy: 0.8845 - val_loss: 0.3211 - val_accuracy: 0.8539
Epoch 72/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2966 - accuracy: 0.8836 - val_loss: 0.3231 - val_accuracy: 0.8630
```

```
Epoch 73/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2954 - accuracy: 0.8865 - val_loss: 0.3227 - val_accuracy: 0.8676
Epoch 74/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2942 - accuracy: 0.8894 - val_loss: 0.3190 - val_accuracy: 0.8676
Epoch 75/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2922 - accuracy: 0.8855 - val_loss: 0.3242 - val_accuracy: 0.8630
Epoch 76/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2922 - accuracy: 0.8865 - val_loss: 0.3182 - val_accuracy: 0.8676
Epoch 77/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2910 - accuracy: 0.8836 - val_loss: 0.3186 - val_accuracy: 0.8676
Epoch 78/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2902 - accuracy: 0.8855 - val_loss: 0.3157 - val_accuracy: 0.8676
Epoch 79/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2892 - accuracy: 0.8875 - val_loss: 0.3140 - val_accuracy: 0.8676
Epoch 80/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2881 - accuracy: 0.8855 - val_loss: 0.3135 - val_accuracy: 0.8630
Epoch 81/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2876 - accuracy: 0.8865 - val_loss: 0.3126 - val_accuracy: 0.8630
Epoch 82/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2867 - accuracy: 0.8855 - val_loss: 0.3122 - val_accuracy: 0.8676
Epoch 83/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2848 - accuracy: 0.8875 - val_loss: 0.3097 - val_accuracy: 0.8676
Epoch 84/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2844 - accuracy: 0.8914 - val_loss: 0.3092 - val_accuracy: 0.8721
Epoch 85/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2846 - accuracy: 0.8865 - val_loss: 0.3095 - val_accuracy: 0.8676
Epoch 86/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2835 - accuracy: 0.8885 - val_loss: 0.3094 - val_accuracy: 0.8676
Epoch 87/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2822 - accuracy: 0.8894 - val_loss: 0.3079 - val_accuracy: 0.8676
Epoch 88/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2816 - accuracy: 0.8855 - val_loss: 0.3070 - val_accuracy: 0.8676
Epoch 89/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2805 - accuracy: 0.8875 - val_loss: 0.3059 - val_accuracy: 0.8767
Epoch 90/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2804 - accuracy: 0.8894 - val_loss: 0.3063 - val_accuracy: 0.8676
Epoch 91/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2796 - accuracy: 0.8904 - val_loss: 0.3066 - val_accuracy: 0.8721
Epoch 92/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2785 - accuracy: 0.8943 - val_loss: 0.3068 - val_accuracy: 0.8676
Epoch 93/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2789 - accuracy: 0.8904 - val_loss: 0.3067 - val_accuracy: 0.8676
Epoch 94/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2772 - accuracy: 0.8924 - val_loss: 0.3029 - val_accuracy: 0.8721
Epoch 95/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2771 - accuracy: 0.8894 - val_loss: 0.3071 - val_accuracy: 0.8676
Epoch 96/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2751 - accuracy: 0.8894 - val_loss: 0.3013 - val_accuracy: 0.8721
```

```
Epoch 97/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2755 - accuracy: 0.8904 - val_loss: 0.3025 - val_accuracy: 0.8676
Epoch 98/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2751 - accuracy: 0.8924 - val_loss: 0.3028 - val_accuracy: 0.8721
Epoch 99/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2741 - accuracy: 0.8904 - val_loss: 0.2996 - val_accuracy: 0.8767
Epoch 100/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2723 - accuracy: 0.8885 - val_loss: 0.3065 - val_accuracy: 0.8721
```

Evaluating the model

In [19]:
```
model.evaluate(X_test, Y_test)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.2777 - accuracy: 0.9087
```

Out[19]:
```
[0.2777245044708252, 0.9086757898330688]
```

Summary of the model

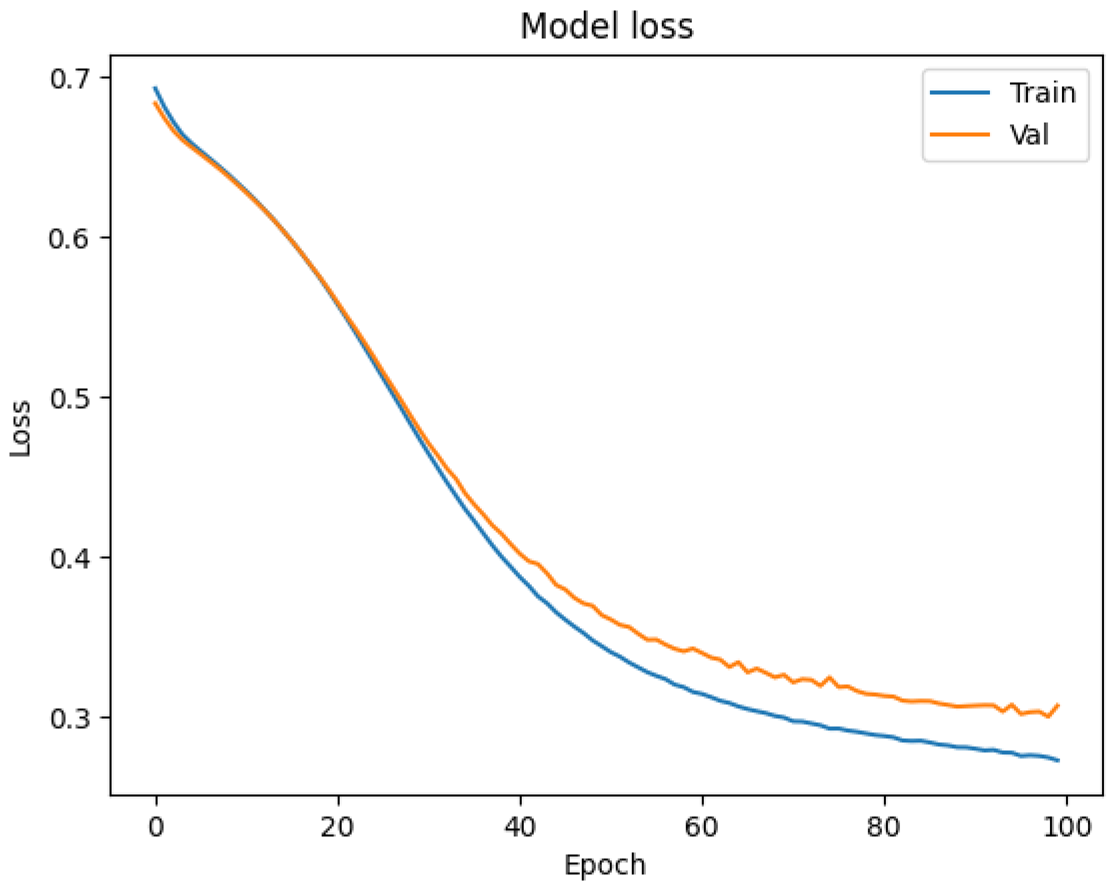
- setting up the architecture with keras. This included the input layer, 2 hidden layers and 1 output layer.
- compiling the model by seeting the settings of the model (optimizer, loss function & metrics)
- training the model and finding the best fit parameters with using the validation data.
- evaluating the nodel on the test dataset.

2.3. Visualizing Loss and Accuracy

THis chapter will be focussed on visualizing the training loss and validation loss to see how our model trained. We will be looking how well our model performs and if we have overfitted. Overfitting occurs when the model has fitted so well to the training dataset that it has failed to generalize to unseen examples. This is characterized by a high dev (validation) loss and a low train loss, and can be addressed with regularization techniques. These techniques can be see as adding regularization, early stopping or dropout.

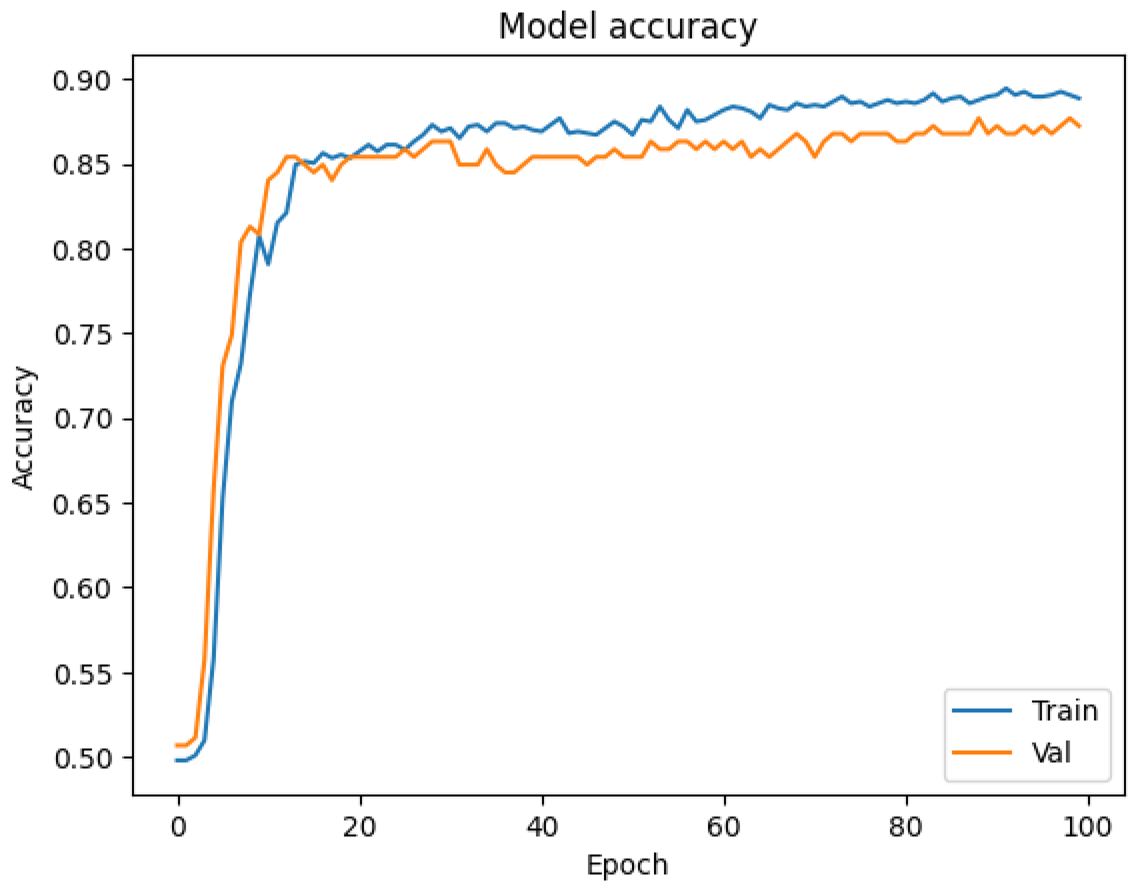
In [20]:
```
# Needed imports
import matplotlib.pyplot as plt
```

In [21]:
```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Next we will be plotting the training accuracy and validation accuracy

In [22]:
```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



the plot above shows the model's accuracy for the training and validation set

2.4 Confusion Matrix

In [23]:
```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

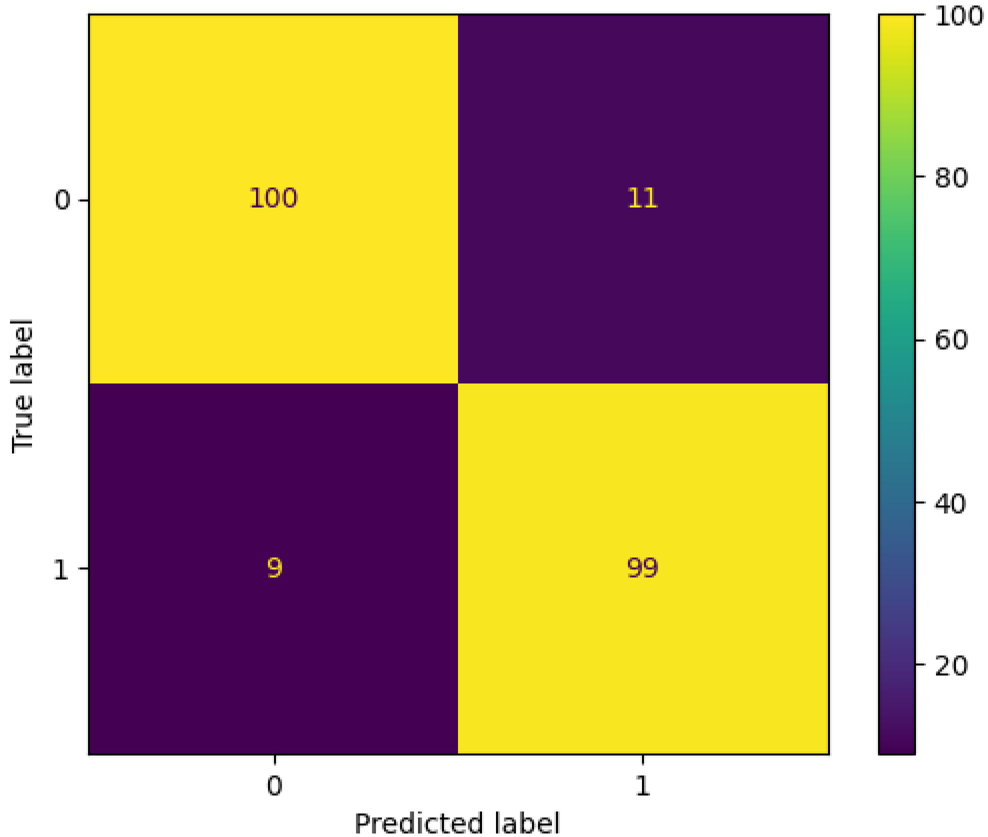
In [24]:
```
Y_pred = model.predict(X_test)
y_pred = np.round(Y_pred, 0).tolist()
```

```
7/7 [==============================] - 0s 3ms/step
```

In [25]:
```
confusion_matrix(Y_test, y_pred)
```

Out[25]:
```
array([[100,  11],
       [  9,  99]], dtype=int64)
```

In [26]:
```
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



In [27]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       111
           1       0.90      0.92      0.91       108

    accuracy                           0.91       219
   macro avg       0.91      0.91      0.91       219
weighted avg       0.91      0.91      0.91       219
```

3. Hypertuning the model

In this section we will be hypertuning the optimizer and loss function for the model to see if it will perform better or worse and why.

3.1. Hypertuning the model optimizers

In this chapter we will use the same steps for setting up the neural network but play around with the hyper paramters to see if we can get a better model by evaluating the loss and accuracy.

In this section we will be training and hypertuning the model to find the best results. The steps will go as followed:

1. Specify some hyper-parameters (the template)
2. Train on the training dataset (filling in the parameters)
3. Applying confusion matrix on the output
4. Record the validation loss
5. Repeat Steps 1 to 3 with a different set of hyper-parameters (many times)
6. Conclusion

3.1.1 Using Adam as optimizer

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Neural Network Architecture

In [28]:
```
model_2 = Sequential([
    Dense(32, activation='relu', input_shape=(10,)), #32 is the size of the first layer, & 10 referes to the 10 input features
    Dense(32, activation='relu'), #32 is the size of the first layer
    Dense(1, activation='sigmoid'), #1 neuron as output layer for the prediction
])
```

In [29]:
```
model_2.compile(optimizer='Adam',
              loss='binary_crossentropy', #chosen for our binary output
              metrics=['accuracy'])
```

Training the Model

In [30]:
```
hist_test_2 = model_2.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 9ms/step - loss: 0.6737 - accuracy: 0.5783 - val_loss: 0.6570 - val_accuracy: 0.5205
Epoch 2/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6428 - accuracy: 0.6135 - val_loss: 0.6221 - val_accuracy: 0.6941
Epoch 3/100
32/32 [==============================] - 0s 5ms/step - loss: 0.5958 - accuracy: 0.7524 - val_loss: 0.5658 - val_accuracy: 0.7717
Epoch 4/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5287 - accuracy: 0.8190 - val_loss: 0.4973 - val_accuracy: 0.8311
Epoch 5/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4565 - accuracy: 0.8454 - val_loss: 0.4354 - val_accuracy: 0.8493
Epoch 6/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3979 - accuracy: 0.8601 - val_loss: 0.3942 - val_accuracy: 0.8402
Epoch 7/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3601 - accuracy: 0.8611 - val_loss: 0.3702 - val_accuracy: 0.8402
Epoch 8/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3380 - accuracy: 0.8699 - val_loss: 0.3594 - val_accuracy: 0.8584
Epoch 9/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3232 - accuracy: 0.8767 - val_loss: 0.3484 - val_accuracy: 0.8584
Epoch 10/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3110 - accuracy: 0.8816 - val_loss: 0.3327 - val_accuracy: 0.8493
Epoch 11/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3028 - accuracy: 0.8865 - val_loss: 0.3293 - val_accuracy: 0.8721
Epoch 12/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2951 - accuracy: 0.8826 - val_loss: 0.3228 - val_accuracy: 0.8721
Epoch 13/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2912 - accuracy: 0.8904 - val_loss: 0.3149 - val_accuracy: 0.8676
Epoch 14/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2820 - accuracy: 0.8933 - val_loss: 0.3095 - val_accuracy: 0.8767
Epoch 15/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2792 - accuracy: 0.8894 - val_loss: 0.3064 - val_accuracy: 0.8447
Epoch 16/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2797 - accuracy: 0.8816 - val_loss: 0.3021 - val_accuracy: 0.8539
Epoch 17/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2656 - accuracy: 0.8924 - val_loss: 0.2957 - val_accuracy: 0.8630
Epoch 18/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2648 - accuracy: 0.8943 - val_loss: 0.2908 - val_accuracy: 0.8676
Epoch 19/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2599 - accuracy: 0.9012 - val_loss: 0.2891 - val_accuracy: 0.8721
Epoch 20/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2582 - accuracy: 0.9051 - val_loss: 0.2880 - val_accuracy: 0.8767
Epoch 21/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2512 - accuracy: 0.9012 - val_loss: 0.2815 - val_accuracy: 0.8676
Epoch 22/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2517 - accuracy: 0.9012 - val_loss: 0.2787 - val_accuracy: 0.8676
Epoch 23/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2516 - accuracy: 0.9061 - val_loss: 0.2761 - val_accuracy: 0.8630
Epoch 24/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2451 - accuracy: 0.9041 - val_loss: 0.2799 - val_accuracy: 0.8767
```

```
Epoch 25/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2484 - accuracy: 0.8904 - val_loss: 0.2706 - val_accuracy: 0.8721
Epoch 26/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2438 - accuracy: 0.9002 - val_loss: 0.2717 - val_accuracy: 0.8630
Epoch 27/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2393 - accuracy: 0.9051 - val_loss: 0.2685 - val_accuracy: 0.8676
Epoch 28/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2355 - accuracy: 0.9012 - val_loss: 0.2674 - val_accuracy: 0.8721
Epoch 29/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2373 - accuracy: 0.9041 - val_loss: 0.2756 - val_accuracy: 0.8767
Epoch 30/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2340 - accuracy: 0.9041 - val_loss: 0.2646 - val_accuracy: 0.8676
Epoch 31/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2329 - accuracy: 0.9051 - val_loss: 0.2643 - val_accuracy: 0.8676
Epoch 32/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2372 - accuracy: 0.8943 - val_loss: 0.2686 - val_accuracy: 0.8721
Epoch 33/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2283 - accuracy: 0.9031 - val_loss: 0.2601 - val_accuracy: 0.8721
Epoch 34/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2273 - accuracy: 0.9070 - val_loss: 0.2643 - val_accuracy: 0.8721
Epoch 35/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2241 - accuracy: 0.9041 - val_loss: 0.2587 - val_accuracy: 0.8721
Epoch 36/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2254 - accuracy: 0.9051 - val_loss: 0.2578 - val_accuracy: 0.8676
Epoch 37/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2284 - accuracy: 0.9012 - val_loss: 0.2636 - val_accuracy: 0.8721
Epoch 38/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2250 - accuracy: 0.9031 - val_loss: 0.2623 - val_accuracy: 0.8767
Epoch 39/100
32/32 [==============================] - 0s 7ms/step - loss: 0.2233 - accuracy: 0.9110 - val_loss: 0.2547 - val_accuracy: 0.8721
Epoch 40/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2198 - accuracy: 0.9070 - val_loss: 0.2583 - val_accuracy: 0.8721
Epoch 41/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2214 - accuracy: 0.9041 - val_loss: 0.2527 - val_accuracy: 0.8676
Epoch 42/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2261 - accuracy: 0.9051 - val_loss: 0.2809 - val_accuracy: 0.8813
Epoch 43/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2298 - accuracy: 0.9080 - val_loss: 0.2557 - val_accuracy: 0.8676
Epoch 44/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2198 - accuracy: 0.9002 - val_loss: 0.2522 - val_accuracy: 0.8767
Epoch 45/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2172 - accuracy: 0.9080 - val_loss: 0.2520 - val_accuracy: 0.8767
Epoch 46/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2159 - accuracy: 0.9110 - val_loss: 0.2661 - val_accuracy: 0.8813
Epoch 47/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2198 - accuracy: 0.9080 - val_loss: 0.2504 - val_accuracy: 0.8676
Epoch 48/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2235 - accuracy: 0.9061 - val_loss: 0.2500 - val_accuracy: 0.8813
```

```
Epoch 49/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2166 - accuracy: 0.9061 - val_loss: 0.2500 - val_accuracy: 0.8813
Epoch 50/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2158 - accuracy: 0.9061 - val_loss: 0.2555 - val_accuracy: 0.8676
Epoch 51/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2144 - accuracy: 0.9100 - val_loss: 0.2527 - val_accuracy: 0.8676
Epoch 52/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2142 - accuracy: 0.9129 - val_loss: 0.2493 - val_accuracy: 0.8676
Epoch 53/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2147 - accuracy: 0.9139 - val_loss: 0.2500 - val_accuracy: 0.8630
Epoch 54/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2117 - accuracy: 0.9070 - val_loss: 0.2572 - val_accuracy: 0.8721
Epoch 55/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2165 - accuracy: 0.9061 - val_loss: 0.2636 - val_accuracy: 0.8721
Epoch 56/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2254 - accuracy: 0.9070 - val_loss: 0.2520 - val_accuracy: 0.8721
Epoch 57/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2195 - accuracy: 0.9022 - val_loss: 0.2516 - val_accuracy: 0.8676
Epoch 58/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2175 - accuracy: 0.9090 - val_loss: 0.2496 - val_accuracy: 0.8858
Epoch 59/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2112 - accuracy: 0.9061 - val_loss: 0.2497 - val_accuracy: 0.8676
Epoch 60/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2109 - accuracy: 0.9090 - val_loss: 0.2506 - val_accuracy: 0.8721
Epoch 61/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2173 - accuracy: 0.9051 - val_loss: 0.2575 - val_accuracy: 0.8721
Epoch 62/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2137 - accuracy: 0.9070 - val_loss: 0.2561 - val_accuracy: 0.8767
Epoch 63/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2131 - accuracy: 0.9129 - val_loss: 0.2512 - val_accuracy: 0.8721
Epoch 64/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2192 - accuracy: 0.9051 - val_loss: 0.2500 - val_accuracy: 0.8721
Epoch 65/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2174 - accuracy: 0.9080 - val_loss: 0.2481 - val_accuracy: 0.8813
Epoch 66/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2088 - accuracy: 0.9090 - val_loss: 0.2503 - val_accuracy: 0.8721
Epoch 67/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2091 - accuracy: 0.9129 - val_loss: 0.2482 - val_accuracy: 0.8813
Epoch 68/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2068 - accuracy: 0.9051 - val_loss: 0.2812 - val_accuracy: 0.8721
Epoch 69/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2189 - accuracy: 0.9119 - val_loss: 0.2552 - val_accuracy: 0.8721
Epoch 70/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2076 - accuracy: 0.9119 - val_loss: 0.2500 - val_accuracy: 0.8721
Epoch 71/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2120 - accuracy: 0.9080 - val_loss: 0.2508 - val_accuracy: 0.8676
Epoch 72/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2065 - accuracy: 0.9100 - val_loss: 0.2488 - val_accuracy: 0.8858
```

```
Epoch 73/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2086 - accuracy: 0.9129 - val_loss: 0.2474 - val_accuracy: 0.8813
Epoch 74/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2110 - accuracy: 0.9070 - val_loss: 0.2487 - val_accuracy: 0.8721
Epoch 75/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2087 - accuracy: 0.9119 - val_loss: 0.2498 - val_accuracy: 0.8767
Epoch 76/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2121 - accuracy: 0.9129 - val_loss: 0.2486 - val_accuracy: 0.8813
Epoch 77/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2097 - accuracy: 0.9168 - val_loss: 0.2551 - val_accuracy: 0.8721
Epoch 78/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2166 - accuracy: 0.9110 - val_loss: 0.2490 - val_accuracy: 0.8813
Epoch 79/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2096 - accuracy: 0.9061 - val_loss: 0.2466 - val_accuracy: 0.8813
Epoch 80/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2102 - accuracy: 0.9119 - val_loss: 0.2517 - val_accuracy: 0.8767
Epoch 81/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2047 - accuracy: 0.9139 - val_loss: 0.2568 - val_accuracy: 0.8676
Epoch 82/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2059 - accuracy: 0.9129 - val_loss: 0.2471 - val_accuracy: 0.8767
Epoch 83/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2041 - accuracy: 0.9207 - val_loss: 0.2729 - val_accuracy: 0.8676
Epoch 84/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2152 - accuracy: 0.9139 - val_loss: 0.2562 - val_accuracy: 0.8676
Epoch 85/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2060 - accuracy: 0.9110 - val_loss: 0.2645 - val_accuracy: 0.8721
Epoch 86/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2058 - accuracy: 0.9129 - val_loss: 0.2477 - val_accuracy: 0.8721
Epoch 87/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2069 - accuracy: 0.9119 - val_loss: 0.2476 - val_accuracy: 0.8813
Epoch 88/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2066 - accuracy: 0.9149 - val_loss: 0.2471 - val_accuracy: 0.8813
Epoch 89/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2035 - accuracy: 0.9149 - val_loss: 0.2505 - val_accuracy: 0.8721
Epoch 90/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2035 - accuracy: 0.9100 - val_loss: 0.2475 - val_accuracy: 0.8813
Epoch 91/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2042 - accuracy: 0.9168 - val_loss: 0.2478 - val_accuracy: 0.8813
Epoch 92/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2039 - accuracy: 0.9139 - val_loss: 0.2462 - val_accuracy: 0.8813
Epoch 93/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2010 - accuracy: 0.9149 - val_loss: 0.2586 - val_accuracy: 0.8676
Epoch 94/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2032 - accuracy: 0.9149 - val_loss: 0.2485 - val_accuracy: 0.8721
Epoch 95/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2049 - accuracy: 0.9090 - val_loss: 0.2477 - val_accuracy: 0.8858
Epoch 96/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2065 - accuracy: 0.9119 - val_loss: 0.2474 - val_accuracy: 0.8767
```

```
Epoch 97/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2012 - accuracy: 0.9129 - val_loss: 0.2492 - val_accuracy: 0.8721
Epoch 98/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2018 - accuracy: 0.9139 - val_loss: 0.2486 - val_accuracy: 0.8813
Epoch 99/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2126 - accuracy: 0.9119 - val_loss: 0.2594 - val_accuracy: 0.8721
Epoch 100/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2018 - accuracy: 0.9168 - val_loss: 0.2489 - val_accuracy: 0.8767
```

Evaluating the Model

In [31]:
```
model_2.evaluate(X_test, Y_test)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.2356 - accuracy: 0.9132
```

Out[31]:
```
[0.23562783002853394, 0.913241982460022]
```

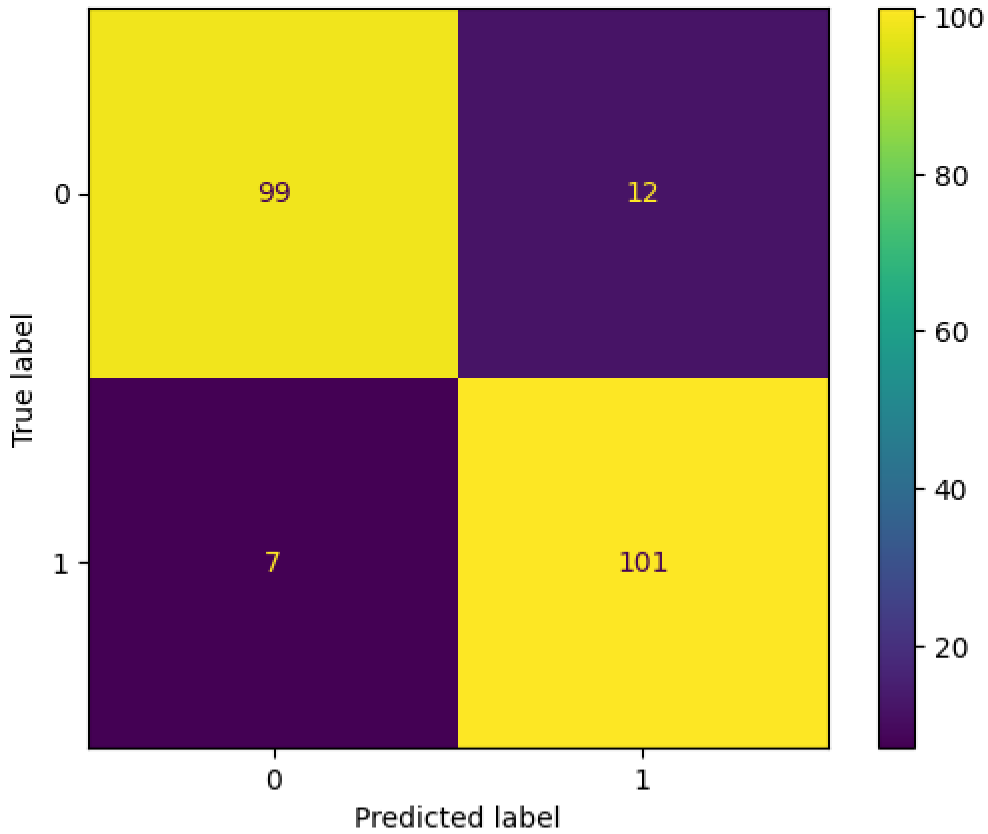
Confusion Matrix

In [32]:
```
Y_pred = model_2.predict(X_test).round()
y_pred = np.round(Y_pred, 0).tolist()
confusion_matrix(Y_test, y_pred)
```

```
7/7 [==============================] - 0s 2ms/step
```

Out[32]:
```
array([[ 99,  12],
       [  7, 101]], dtype=int64)
```

In [33]:
```
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

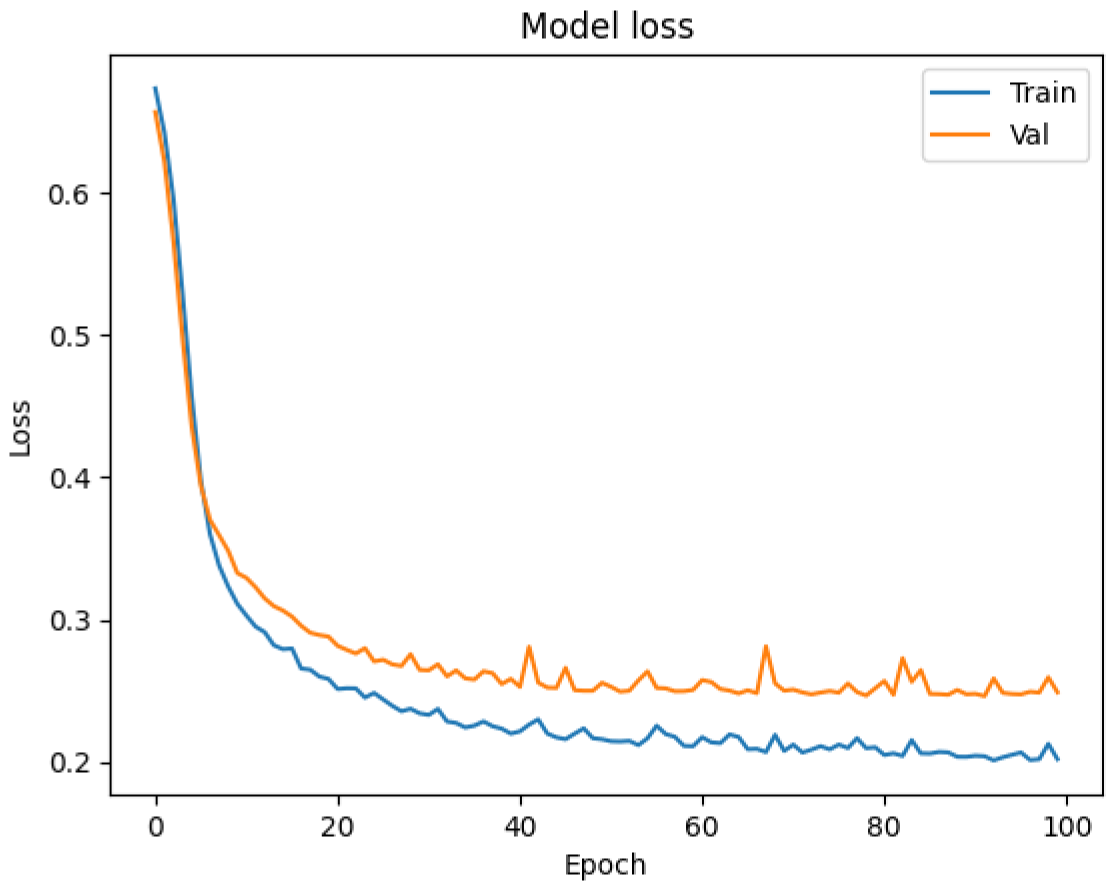


In [34]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       111
           1       0.89      0.94      0.91       108

    accuracy                           0.91       219
   macro avg       0.91      0.91      0.91       219
weighted avg       0.91      0.91      0.91       219
```

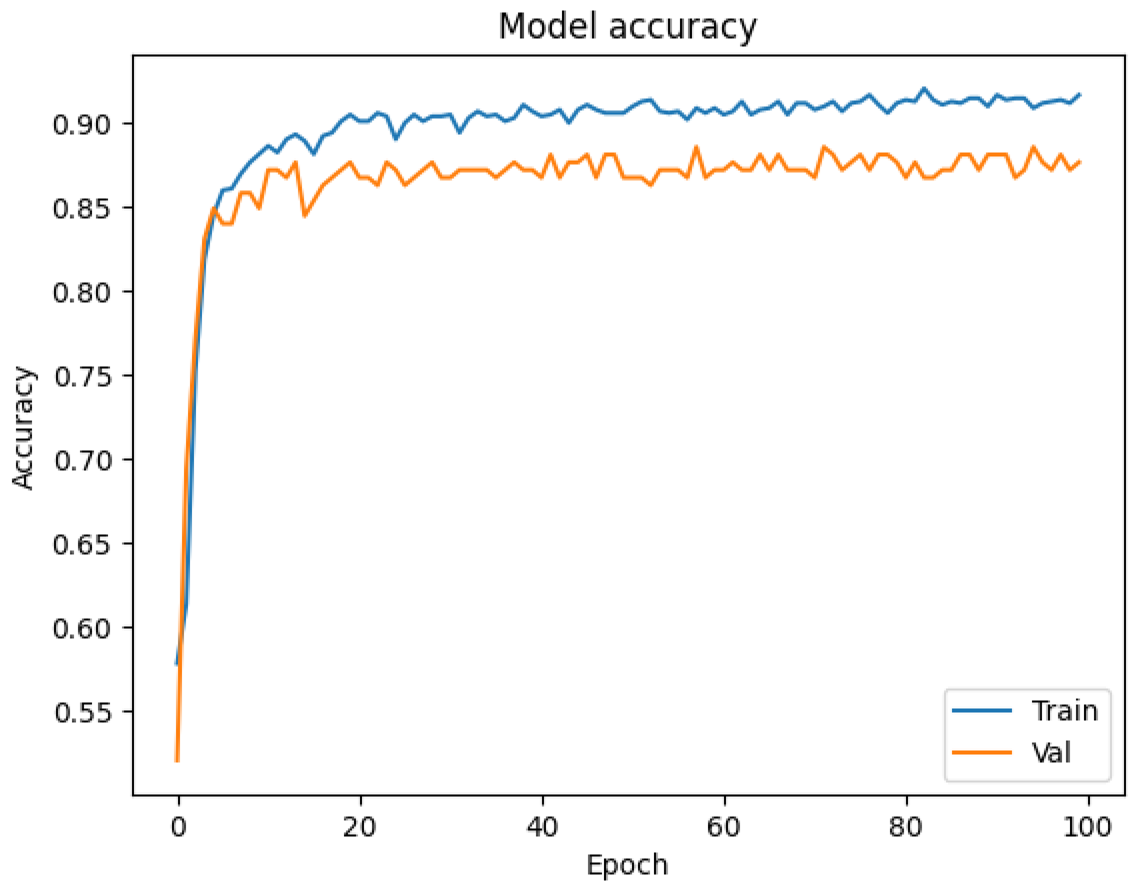
Visualizing Loss

In [35]:
```
plt.plot(hist_test_2.history['loss'])
plt.plot(hist_test_2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Visualizing Accuracy

In [36]:
```
plt.plot(hist_test_2.history['accuracy'])
plt.plot(hist_test_2.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



3.1.2 Using RMSprop as optimizer

The gist of RMSprop is to:

- Maintain a moving (discounted) average of the square of gradients
- Divide the gradient by the root of this average

Neural Network Architecture

In [37]:
```
model_3 = Sequential([
    Dense(32, activation='relu', input_shape=(10,)), #32 is the size of the first layer, & 10 referes to the 10 input features
    Dense(32, activation='relu'), #32 is the size of the first layer
    Dense(1, activation='sigmoid'), #1 neuron as output layer for the prediction
])
```

In [38]:
```
model_3.compile(optimizer='RMSprop',
              loss='binary_crossentropy', #chosen for our binary output
```

```
               metrics=['accuracy'])
```

Training the Model

In [39]:
```
hist_test_3 = model_3.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 9ms/step - loss: 0.6681 - accuracy: 0.5489 - val_loss: 0.6447 - val_accuracy: 0.7489
Epoch 2/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6225 - accuracy: 0.7779 - val_loss: 0.6033 - val_accuracy: 0.7534
Epoch 3/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5704 - accuracy: 0.8082 - val_loss: 0.5474 - val_accuracy: 0.8356
Epoch 4/100
32/32 [==============================] - 0s 4ms/step - loss: 0.5131 - accuracy: 0.8278 - val_loss: 0.4996 - val_accuracy: 0.8037
Epoch 5/100
32/32 [==============================] - 0s 4ms/step - loss: 0.4607 - accuracy: 0.8337 - val_loss: 0.4545 - val_accuracy: 0.8037
Epoch 6/100
32/32 [==============================] - 0s 4ms/step - loss: 0.4170 - accuracy: 0.8552 - val_loss: 0.4208 - val_accuracy: 0.8493
Epoch 7/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3876 - accuracy: 0.8552 - val_loss: 0.3974 - val_accuracy: 0.8447
Epoch 8/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3660 - accuracy: 0.8620 - val_loss: 0.3858 - val_accuracy: 0.8174
Epoch 9/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3505 - accuracy: 0.8650 - val_loss: 0.3728 - val_accuracy: 0.8265
Epoch 10/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3376 - accuracy: 0.8728 - val_loss: 0.3574 - val_accuracy: 0.8447
Epoch 11/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3270 - accuracy: 0.8708 - val_loss: 0.3609 - val_accuracy: 0.8311
Epoch 12/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3180 - accuracy: 0.8796 - val_loss: 0.3475 - val_accuracy: 0.8539
Epoch 13/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3141 - accuracy: 0.8738 - val_loss: 0.3342 - val_accuracy: 0.8493
Epoch 14/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3080 - accuracy: 0.8836 - val_loss: 0.3299 - val_accuracy: 0.8493
Epoch 15/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3029 - accuracy: 0.8816 - val_loss: 0.3325 - val_accuracy: 0.8584
Epoch 16/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2990 - accuracy: 0.8836 - val_loss: 0.3205 - val_accuracy: 0.8539
Epoch 17/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2953 - accuracy: 0.8826 - val_loss: 0.3285 - val_accuracy: 0.8630
Epoch 18/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2902 - accuracy: 0.8865 - val_loss: 0.3126 - val_accuracy: 0.8493
Epoch 19/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2839 - accuracy: 0.8885 - val_loss: 0.3114 - val_accuracy: 0.8584
Epoch 20/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2832 - accuracy: 0.8914 - val_loss: 0.3052 - val_accuracy: 0.8493
Epoch 21/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2797 - accuracy: 0.8875 - val_loss: 0.3024 - val_accuracy: 0.8539
Epoch 22/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2775 - accuracy: 0.8924 - val_loss: 0.3120 - val_accuracy: 0.8676
Epoch 23/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2742 - accuracy: 0.8973 - val_loss: 0.2954 - val_accuracy: 0.8584
Epoch 24/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2696 - accuracy: 0.8904 - val_loss: 0.3022 - val_accuracy: 0.8676
```

```
Epoch 25/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2699 - accuracy: 0.8943 - val_loss: 0.2900 - val_accuracy: 0.8584
Epoch 26/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2652 - accuracy: 0.8904 - val_loss: 0.2951 - val_accuracy: 0.8676
Epoch 27/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2623 - accuracy: 0.8933 - val_loss: 0.2860 - val_accuracy: 0.8676
Epoch 28/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2603 - accuracy: 0.8982 - val_loss: 0.2936 - val_accuracy: 0.8721
Epoch 29/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2601 - accuracy: 0.8973 - val_loss: 0.2794 - val_accuracy: 0.8584
Epoch 30/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2551 - accuracy: 0.9022 - val_loss: 0.2775 - val_accuracy: 0.8584
Epoch 31/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2537 - accuracy: 0.8973 - val_loss: 0.2838 - val_accuracy: 0.8721
Epoch 32/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2519 - accuracy: 0.8982 - val_loss: 0.2734 - val_accuracy: 0.8721
Epoch 33/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2484 - accuracy: 0.8973 - val_loss: 0.2692 - val_accuracy: 0.8676
Epoch 34/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2468 - accuracy: 0.9012 - val_loss: 0.2682 - val_accuracy: 0.8676
Epoch 35/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2463 - accuracy: 0.8992 - val_loss: 0.2701 - val_accuracy: 0.8767
Epoch 36/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2439 - accuracy: 0.9002 - val_loss: 0.2647 - val_accuracy: 0.8721
Epoch 37/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2425 - accuracy: 0.8982 - val_loss: 0.2639 - val_accuracy: 0.8676
Epoch 38/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2389 - accuracy: 0.9051 - val_loss: 0.2765 - val_accuracy: 0.8721
Epoch 39/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2388 - accuracy: 0.8963 - val_loss: 0.2707 - val_accuracy: 0.8767
Epoch 40/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2381 - accuracy: 0.9002 - val_loss: 0.2776 - val_accuracy: 0.8813
Epoch 41/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2376 - accuracy: 0.9051 - val_loss: 0.2586 - val_accuracy: 0.8721
Epoch 42/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2357 - accuracy: 0.9012 - val_loss: 0.2647 - val_accuracy: 0.8767
Epoch 43/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2356 - accuracy: 0.8992 - val_loss: 0.2672 - val_accuracy: 0.8767
Epoch 44/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2361 - accuracy: 0.9041 - val_loss: 0.2591 - val_accuracy: 0.8721
Epoch 45/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2336 - accuracy: 0.9070 - val_loss: 0.2761 - val_accuracy: 0.8767
Epoch 46/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2354 - accuracy: 0.9041 - val_loss: 0.2657 - val_accuracy: 0.8767
Epoch 47/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2319 - accuracy: 0.9022 - val_loss: 0.2791 - val_accuracy: 0.8767
Epoch 48/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2322 - accuracy: 0.9022 - val_loss: 0.2542 - val_accuracy: 0.8721
```

```
Epoch 49/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2303 - accuracy: 0.9002 - val_loss: 0.2509 - val_accuracy: 0.8721
Epoch 50/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2314 - accuracy: 0.9070 - val_loss: 0.2534 - val_accuracy: 0.8721
Epoch 51/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2311 - accuracy: 0.8992 - val_loss: 0.2503 - val_accuracy: 0.8721
Epoch 52/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2289 - accuracy: 0.9061 - val_loss: 0.2518 - val_accuracy: 0.8721
Epoch 53/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2277 - accuracy: 0.9022 - val_loss: 0.2581 - val_accuracy: 0.8767
Epoch 54/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2251 - accuracy: 0.9100 - val_loss: 0.2486 - val_accuracy: 0.8721
Epoch 55/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2278 - accuracy: 0.9061 - val_loss: 0.2540 - val_accuracy: 0.8721
Epoch 56/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2266 - accuracy: 0.9022 - val_loss: 0.2533 - val_accuracy: 0.8721
Epoch 57/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2252 - accuracy: 0.9002 - val_loss: 0.2478 - val_accuracy: 0.8721
Epoch 58/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2265 - accuracy: 0.9031 - val_loss: 0.2529 - val_accuracy: 0.8721
Epoch 59/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2245 - accuracy: 0.9022 - val_loss: 0.2492 - val_accuracy: 0.8721
Epoch 60/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2269 - accuracy: 0.9051 - val_loss: 0.2502 - val_accuracy: 0.8721
Epoch 61/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2196 - accuracy: 0.9090 - val_loss: 0.2494 - val_accuracy: 0.8721
Epoch 62/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2230 - accuracy: 0.9041 - val_loss: 0.2452 - val_accuracy: 0.8767
Epoch 63/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2225 - accuracy: 0.9100 - val_loss: 0.2459 - val_accuracy: 0.8904
Epoch 64/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2256 - accuracy: 0.9041 - val_loss: 0.2583 - val_accuracy: 0.8721
Epoch 65/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2216 - accuracy: 0.9012 - val_loss: 0.2450 - val_accuracy: 0.8767
Epoch 66/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2224 - accuracy: 0.9051 - val_loss: 0.2446 - val_accuracy: 0.8813
Epoch 67/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2217 - accuracy: 0.9041 - val_loss: 0.2448 - val_accuracy: 0.8858
Epoch 68/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2205 - accuracy: 0.9022 - val_loss: 0.2476 - val_accuracy: 0.8721
Epoch 69/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2203 - accuracy: 0.9041 - val_loss: 0.2450 - val_accuracy: 0.8721
Epoch 70/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2221 - accuracy: 0.9041 - val_loss: 0.2452 - val_accuracy: 0.8676
Epoch 71/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2230 - accuracy: 0.9041 - val_loss: 0.2448 - val_accuracy: 0.8676
Epoch 72/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2209 - accuracy: 0.9051 - val_loss: 0.2457 - val_accuracy: 0.8721
```

```
Epoch 73/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2224 - accuracy: 0.9031 - val_loss: 0.2435 - val_accuracy: 0.8767
Epoch 74/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2190 - accuracy: 0.9041 - val_loss: 0.2425 - val_accuracy: 0.8813
Epoch 75/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2186 - accuracy: 0.9022 - val_loss: 0.2518 - val_accuracy: 0.8676
Epoch 76/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2212 - accuracy: 0.9061 - val_loss: 0.2473 - val_accuracy: 0.8813
Epoch 77/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2183 - accuracy: 0.9080 - val_loss: 0.2491 - val_accuracy: 0.8721
Epoch 78/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2161 - accuracy: 0.9080 - val_loss: 0.2486 - val_accuracy: 0.8858
Epoch 79/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2223 - accuracy: 0.8963 - val_loss: 0.2435 - val_accuracy: 0.8721
Epoch 80/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2170 - accuracy: 0.9100 - val_loss: 0.2550 - val_accuracy: 0.8721
Epoch 81/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2195 - accuracy: 0.9090 - val_loss: 0.2514 - val_accuracy: 0.8721
Epoch 82/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2176 - accuracy: 0.9090 - val_loss: 0.2522 - val_accuracy: 0.8721
Epoch 83/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2208 - accuracy: 0.9022 - val_loss: 0.2448 - val_accuracy: 0.8721
Epoch 84/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2186 - accuracy: 0.9022 - val_loss: 0.2403 - val_accuracy: 0.8858
Epoch 85/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2175 - accuracy: 0.9061 - val_loss: 0.2412 - val_accuracy: 0.8813
Epoch 86/100
32/32 [==============================] - 0s 6ms/step - loss: 0.2162 - accuracy: 0.9041 - val_loss: 0.2413 - val_accuracy: 0.8813
Epoch 87/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2150 - accuracy: 0.9110 - val_loss: 0.2420 - val_accuracy: 0.8767
Epoch 88/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2173 - accuracy: 0.9041 - val_loss: 0.2418 - val_accuracy: 0.8767
Epoch 89/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2156 - accuracy: 0.9061 - val_loss: 0.2427 - val_accuracy: 0.8813
Epoch 90/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2155 - accuracy: 0.9100 - val_loss: 0.2439 - val_accuracy: 0.8767
Epoch 91/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2115 - accuracy: 0.9012 - val_loss: 0.2643 - val_accuracy: 0.8676
Epoch 92/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2150 - accuracy: 0.9080 - val_loss: 0.2422 - val_accuracy: 0.8767
Epoch 93/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2132 - accuracy: 0.9070 - val_loss: 0.2490 - val_accuracy: 0.8721
Epoch 94/100
32/32 [==============================] - 0s 5ms/step - loss: 0.2148 - accuracy: 0.9051 - val_loss: 0.2424 - val_accuracy: 0.8813
Epoch 95/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2112 - accuracy: 0.9080 - val_loss: 0.2508 - val_accuracy: 0.8858
Epoch 96/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2145 - accuracy: 0.9080 - val_loss: 0.2426 - val_accuracy: 0.8676
```

```
Epoch 97/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2075 - accuracy: 0.9129 - val_loss: 0.2568 - val_accuracy: 0.8676
Epoch 98/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2108 - accuracy: 0.9090 - val_loss: 0.2465 - val_accuracy: 0.8767
Epoch 99/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2116 - accuracy: 0.9100 - val_loss: 0.2440 - val_accuracy: 0.8721
Epoch 100/100
32/32 [==============================] - 0s 4ms/step - loss: 0.2098 - accuracy: 0.9110 - val_loss: 0.2482 - val_accuracy: 0.8858
```

Evaluating the Model

In [40]:
```
model_3.evaluate(X_test, Y_test)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.2582 - accuracy: 0.8950
```

Out[40]:
```
[0.25815895199775696, 0.8949771523475647]
```

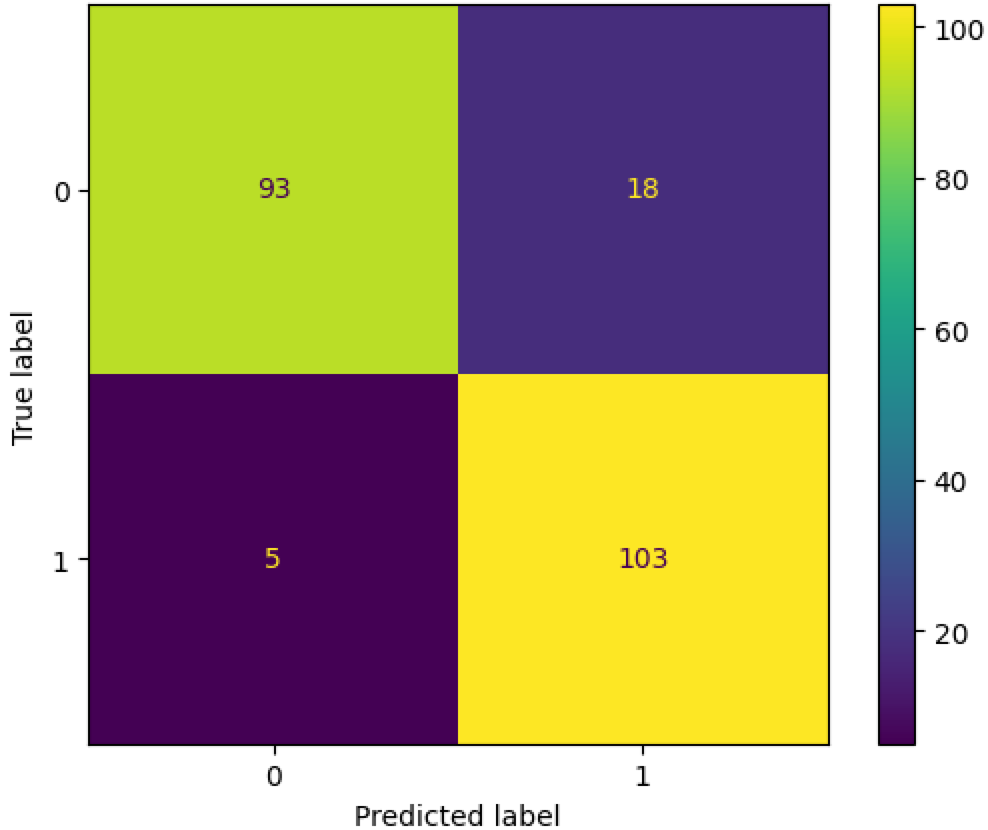
Confusion Matrix

In [41]:
```
Y_pred = model_3.predict(X_test).round()
y_pred = np.round(Y_pred, 0).tolist()
confusion_matrix(Y_test, y_pred)
```

```
7/7 [==============================] - 0s 2ms/step
```

Out[41]:
```
array([[ 93,  18],
       [  5, 103]], dtype=int64)
```

In [42]:
```
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

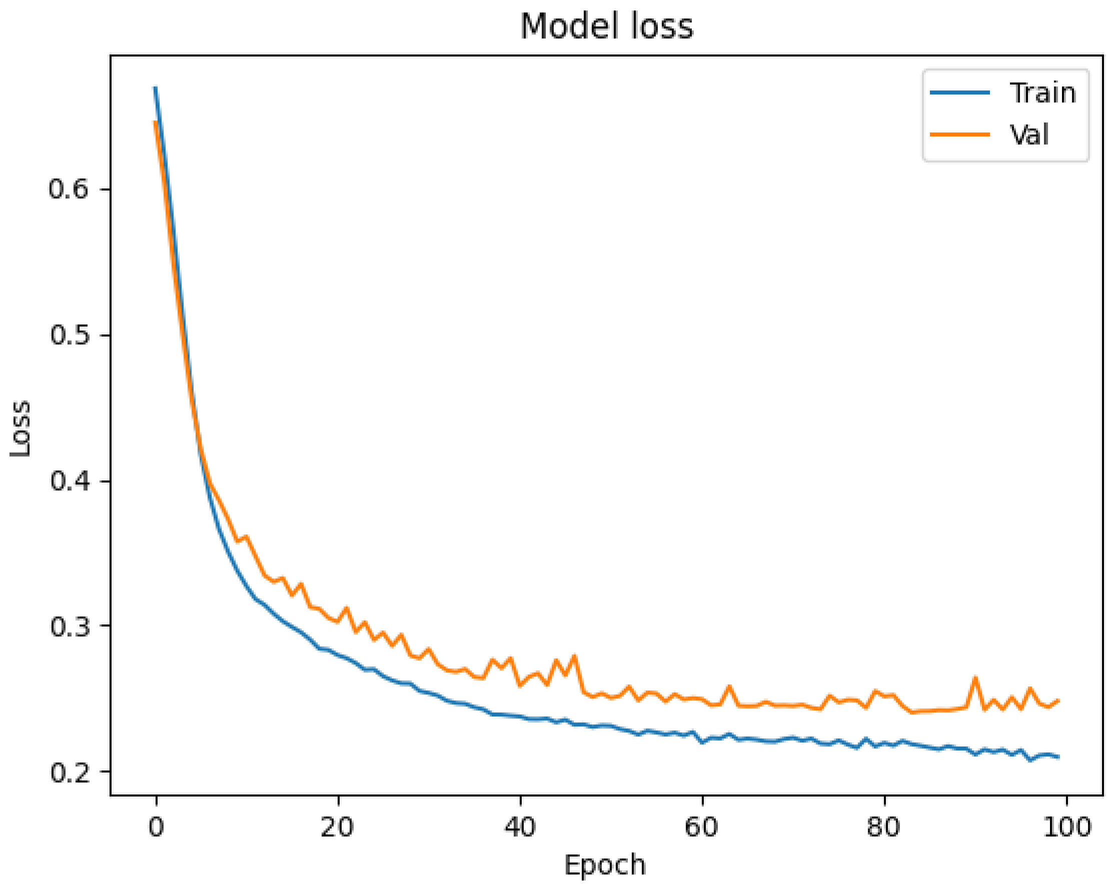


In [43]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.84      0.89       111
           1       0.85      0.95      0.90       108

    accuracy                           0.89       219
   macro avg       0.90      0.90      0.89       219
weighted avg       0.90      0.89      0.89       219
```

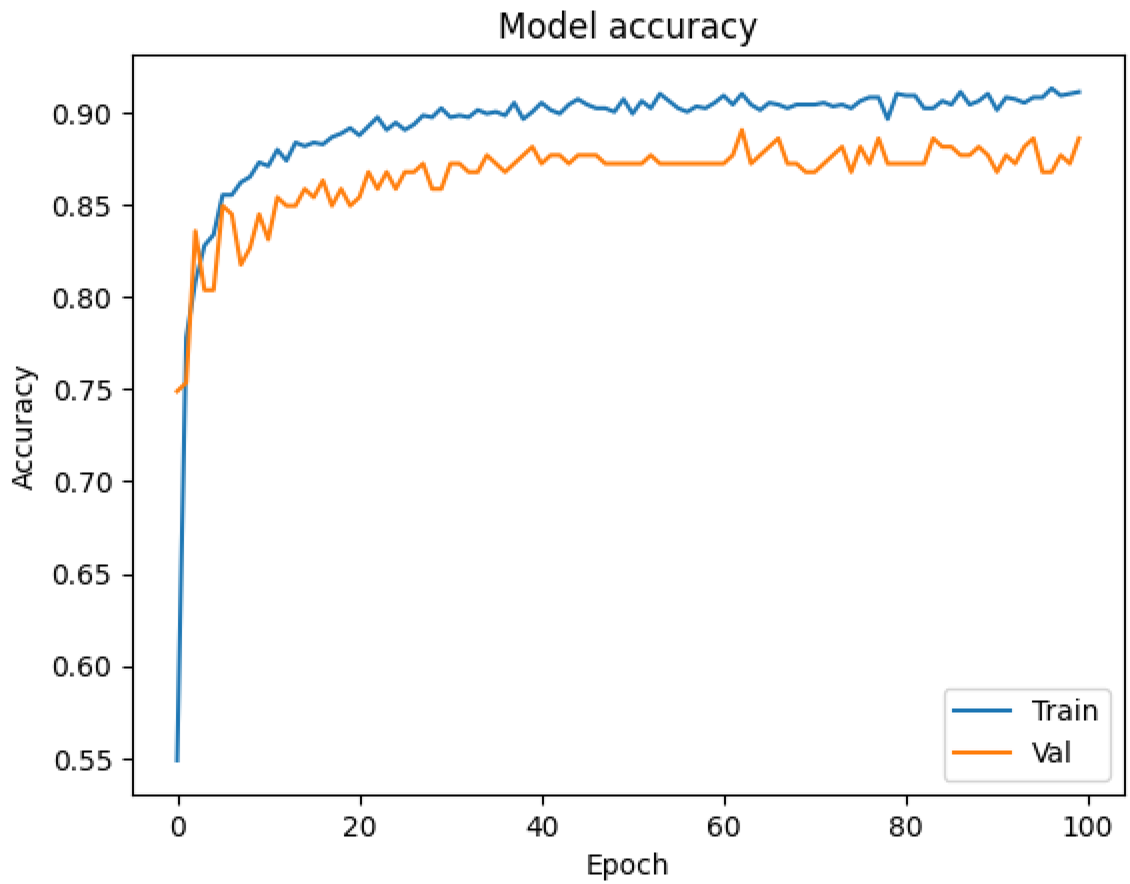
Visualizing Loss

In [44]:
```
plt.plot(hist_test_3.history['loss'])
plt.plot(hist_test_3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Visualizing Accuracy

In [45]:
```
plt.plot(hist_test_3.history['accuracy'])
plt.plot(hist_test_3.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



3.1.3. Conclusion

We have tried out 2 different optimizers (Adam & RMSprop) to compare to the SGD optimizer. Based on the finding, all three give a similiar overal accuracy with the RMSprop being the highest and for the loss section are they all again quite even with the Adam optimizer having the lowest loss. SGD is well known to perform better than Adam with also outperformances on the training data but in our case we had a slight better output with Adam for our dataset and how the model is set up. Lastly, Adam does perform better but if we visualize the loss on the plots compared to SGD, we see that SGD has a even distribution comparing both lines whereas Adam's plots differentiate for each epoch. Lastly, based on the confusion matrix the SGD did better where we would rather have more false positives than false negatives.

3.2. Hypertuning the model loss functions

In this chapter we will use the same steps for setting up the neural network but play around with the hyper paramters to see if we can get a better model by evaluating the loss and accuracy.

We will be following 4 steps:

1. Specify some hyper-parameters (the template)
2. Train on the training dataset (filling in the parameters)
3. Record the validation loss
4. Repeat Steps 1 to 3 with a different set of hyper-parameters (many times)

3.2.1 Using CategoricalHinge as loss

Computes the categorical hinge loss between y_true & y_pred.

Neural Network Architecture

In [46]:
```
model_4 = Sequential([
    Dense(32, activation='relu', input_shape=(10,)), #32 is the size of the first layer, & 10 referes to the 10 input features
    Dense(32, activation='relu'), #32 is the size of the first layer
    Dense(1, activation='sigmoid'), #1 neuron as output layer for the prediction
])
```

In [47]:
```
model_4.compile(optimizer='sgd',
              loss='CategoricalHinge', #chosen for our binary output
              metrics=['accuracy'])
```

Training the Model

In [48]:
```
hist_test_4 = model_4.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 12ms/step - loss: 1.0002 - accuracy: 0.4980 - val_loss: 0.9909 - val_accuracy: 0.5023
Epoch 2/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9991 - accuracy: 0.5059 - val_loss: 0.9898 - val_accuracy: 0.5068
Epoch 3/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9979 - accuracy: 0.5078 - val_loss: 0.9887 - val_accuracy: 0.5068
Epoch 4/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9968 - accuracy: 0.5059 - val_loss: 0.9876 - val_accuracy: 0.5068
Epoch 5/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9957 - accuracy: 0.5059 - val_loss: 0.9866 - val_accuracy: 0.5114
Epoch 6/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9945 - accuracy: 0.5127 - val_loss: 0.9855 - val_accuracy: 0.5205
Epoch 7/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9934 - accuracy: 0.5157 - val_loss: 0.9844 - val_accuracy: 0.5342
Epoch 8/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9922 - accuracy: 0.5157 - val_loss: 0.9832 - val_accuracy: 0.5571
Epoch 9/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9910 - accuracy: 0.5372 - val_loss: 0.9821 - val_accuracy: 0.5753
Epoch 10/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9898 - accuracy: 0.5509 - val_loss: 0.9809 - val_accuracy: 0.5845
Epoch 11/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9885 - accuracy: 0.5528 - val_loss: 0.9797 - val_accuracy: 0.6027
Epoch 12/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9872 - accuracy: 0.5714 - val_loss: 0.9784 - val_accuracy: 0.6119
Epoch 13/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9858 - accuracy: 0.6067 - val_loss: 0.9771 - val_accuracy: 0.6210
Epoch 14/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9844 - accuracy: 0.6204 - val_loss: 0.9757 - val_accuracy: 0.6347
Epoch 15/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9829 - accuracy: 0.6282 - val_loss: 0.9743 - val_accuracy: 0.6438
Epoch 16/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9814 - accuracy: 0.6262 - val_loss: 0.9728 - val_accuracy: 0.6575
Epoch 17/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9798 - accuracy: 0.6360 - val_loss: 0.9713 - val_accuracy: 0.6667
Epoch 18/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9782 - accuracy: 0.6517 - val_loss: 0.9697 - val_accuracy: 0.6804
Epoch 19/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9765 - accuracy: 0.6556 - val_loss: 0.9680 - val_accuracy: 0.6849
Epoch 20/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9747 - accuracy: 0.6614 - val_loss: 0.9663 - val_accuracy: 0.6895
Epoch 21/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9727 - accuracy: 0.6742 - val_loss: 0.9644 - val_accuracy: 0.6941
Epoch 22/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9707 - accuracy: 0.6869 - val_loss: 0.9625 - val_accuracy: 0.7032
Epoch 23/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9686 - accuracy: 0.6918 - val_loss: 0.9604 - val_accuracy: 0.7123
Epoch 24/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9665 - accuracy: 0.6928 - val_loss: 0.9584 - val_accuracy: 0.7169
```

```
Epoch 25/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9643 - accuracy: 0.6996 - val_loss: 0.9563 - val_accuracy: 0.7260
Epoch 26/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9620 - accuracy: 0.6947 - val_loss: 0.9541 - val_accuracy: 0.7306
Epoch 27/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9597 - accuracy: 0.7035 - val_loss: 0.9519 - val_accuracy: 0.7306
Epoch 28/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9573 - accuracy: 0.7065 - val_loss: 0.9496 - val_accuracy: 0.7306
Epoch 29/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9548 - accuracy: 0.7065 - val_loss: 0.9472 - val_accuracy: 0.7352
Epoch 30/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9522 - accuracy: 0.7084 - val_loss: 0.9447 - val_accuracy: 0.7443
Epoch 31/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9495 - accuracy: 0.7162 - val_loss: 0.9421 - val_accuracy: 0.7489
Epoch 32/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9467 - accuracy: 0.7182 - val_loss: 0.9394 - val_accuracy: 0.7580
Epoch 33/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9439 - accuracy: 0.7221 - val_loss: 0.9367 - val_accuracy: 0.7671
Epoch 34/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9409 - accuracy: 0.7221 - val_loss: 0.9338 - val_accuracy: 0.7671
Epoch 35/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9379 - accuracy: 0.7241 - val_loss: 0.9308 - val_accuracy: 0.7626
Epoch 36/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9347 - accuracy: 0.7309 - val_loss: 0.9277 - val_accuracy: 0.7626
Epoch 37/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9314 - accuracy: 0.7417 - val_loss: 0.9245 - val_accuracy: 0.7671
Epoch 38/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9279 - accuracy: 0.7427 - val_loss: 0.9212 - val_accuracy: 0.7717
Epoch 39/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9242 - accuracy: 0.7515 - val_loss: 0.9176 - val_accuracy: 0.7808
Epoch 40/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9205 - accuracy: 0.7524 - val_loss: 0.9140 - val_accuracy: 0.7854
Epoch 41/100
32/32 [==============================] - 0s 4ms/step - loss: 0.9165 - accuracy: 0.7544 - val_loss: 0.9101 - val_accuracy: 0.7854
Epoch 42/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9122 - accuracy: 0.7710 - val_loss: 0.9061 - val_accuracy: 0.7854
Epoch 43/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9079 - accuracy: 0.7759 - val_loss: 0.9019 - val_accuracy: 0.7854
Epoch 44/100
32/32 [==============================] - 0s 5ms/step - loss: 0.9034 - accuracy: 0.7769 - val_loss: 0.8975 - val_accuracy: 0.7854
Epoch 45/100
32/32 [==============================] - 0s 5ms/step - loss: 0.8987 - accuracy: 0.7789 - val_loss: 0.8931 - val_accuracy: 0.7945
Epoch 46/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8940 - accuracy: 0.7906 - val_loss: 0.8887 - val_accuracy: 0.7900
Epoch 47/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8894 - accuracy: 0.7916 - val_loss: 0.8842 - val_accuracy: 0.7900
Epoch 48/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8846 - accuracy: 0.7965 - val_loss: 0.8796 - val_accuracy: 0.7900
```

```
Epoch 49/100
32/32 [==============================] - 0s 5ms/step - loss: 0.8797 - accuracy: 0.8014 - val_loss: 0.8750 - val_accuracy: 0.7945
Epoch 50/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8748 - accuracy: 0.8053 - val_loss: 0.8703 - val_accuracy: 0.7945
Epoch 51/100
32/32 [==============================] - 0s 5ms/step - loss: 0.8697 - accuracy: 0.8131 - val_loss: 0.8656 - val_accuracy: 0.8082
Epoch 52/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8646 - accuracy: 0.8131 - val_loss: 0.8607 - val_accuracy: 0.8128
Epoch 53/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8594 - accuracy: 0.8219 - val_loss: 0.8559 - val_accuracy: 0.8082
Epoch 54/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8541 - accuracy: 0.8219 - val_loss: 0.8511 - val_accuracy: 0.8128
Epoch 55/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8489 - accuracy: 0.8249 - val_loss: 0.8462 - val_accuracy: 0.8082
Epoch 56/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8434 - accuracy: 0.8337 - val_loss: 0.8412 - val_accuracy: 0.8174
Epoch 57/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8381 - accuracy: 0.8405 - val_loss: 0.8362 - val_accuracy: 0.8174
Epoch 58/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8327 - accuracy: 0.8405 - val_loss: 0.8312 - val_accuracy: 0.8265
Epoch 59/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8272 - accuracy: 0.8474 - val_loss: 0.8261 - val_accuracy: 0.8265
Epoch 60/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8219 - accuracy: 0.8464 - val_loss: 0.8212 - val_accuracy: 0.8265
Epoch 61/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8164 - accuracy: 0.8513 - val_loss: 0.8163 - val_accuracy: 0.8265
Epoch 62/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8111 - accuracy: 0.8581 - val_loss: 0.8112 - val_accuracy: 0.8265
Epoch 63/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8060 - accuracy: 0.8493 - val_loss: 0.8067 - val_accuracy: 0.8265
Epoch 64/100
32/32 [==============================] - 0s 4ms/step - loss: 0.8006 - accuracy: 0.8650 - val_loss: 0.8018 - val_accuracy: 0.8265
Epoch 65/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7955 - accuracy: 0.8611 - val_loss: 0.7972 - val_accuracy: 0.8311
Epoch 66/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7906 - accuracy: 0.8581 - val_loss: 0.7928 - val_accuracy: 0.8311
Epoch 67/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7856 - accuracy: 0.8650 - val_loss: 0.7885 - val_accuracy: 0.8402
Epoch 68/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7808 - accuracy: 0.8659 - val_loss: 0.7842 - val_accuracy: 0.8402
Epoch 69/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7761 - accuracy: 0.8640 - val_loss: 0.7800 - val_accuracy: 0.8402
Epoch 70/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7715 - accuracy: 0.8659 - val_loss: 0.7760 - val_accuracy: 0.8402
Epoch 71/100
32/32 [==============================] - 0s 8ms/step - loss: 0.7671 - accuracy: 0.8650 - val_loss: 0.7720 - val_accuracy: 0.8356
Epoch 72/100
32/32 [==============================] - 0s 6ms/step - loss: 0.7629 - accuracy: 0.8650 - val_loss: 0.7686 - val_accuracy: 0.8356
```

```
Epoch 73/100
32/32 [==============================] - 0s 6ms/step - loss: 0.7587 - accuracy: 0.8659 - val_loss: 0.7645 - val_accuracy: 0.8356
Epoch 74/100
32/32 [==============================] - 0s 6ms/step - loss: 0.7547 - accuracy: 0.8659 - val_loss: 0.7609 - val_accuracy: 0.8356
Epoch 75/100
32/32 [==============================] - 0s 6ms/step - loss: 0.7508 - accuracy: 0.8669 - val_loss: 0.7576 - val_accuracy: 0.8356
Epoch 76/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7471 - accuracy: 0.8669 - val_loss: 0.7545 - val_accuracy: 0.8311
Epoch 77/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7435 - accuracy: 0.8659 - val_loss: 0.7511 - val_accuracy: 0.8311
Epoch 78/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7400 - accuracy: 0.8659 - val_loss: 0.7479 - val_accuracy: 0.8311
Epoch 79/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7367 - accuracy: 0.8669 - val_loss: 0.7452 - val_accuracy: 0.8311
Epoch 80/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7336 - accuracy: 0.8728 - val_loss: 0.7421 - val_accuracy: 0.8311
Epoch 81/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7304 - accuracy: 0.8679 - val_loss: 0.7398 - val_accuracy: 0.8356
Epoch 82/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7275 - accuracy: 0.8689 - val_loss: 0.7373 - val_accuracy: 0.8447
Epoch 83/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7246 - accuracy: 0.8699 - val_loss: 0.7349 - val_accuracy: 0.8447
Epoch 84/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7220 - accuracy: 0.8708 - val_loss: 0.7325 - val_accuracy: 0.8447
Epoch 85/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7193 - accuracy: 0.8699 - val_loss: 0.7304 - val_accuracy: 0.8447
Epoch 86/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7167 - accuracy: 0.8718 - val_loss: 0.7284 - val_accuracy: 0.8493
Epoch 87/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7144 - accuracy: 0.8718 - val_loss: 0.7260 - val_accuracy: 0.8493
Epoch 88/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7120 - accuracy: 0.8708 - val_loss: 0.7236 - val_accuracy: 0.8447
Epoch 89/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7099 - accuracy: 0.8718 - val_loss: 0.7214 - val_accuracy: 0.8447
Epoch 90/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7076 - accuracy: 0.8728 - val_loss: 0.7202 - val_accuracy: 0.8493
Epoch 91/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7056 - accuracy: 0.8718 - val_loss: 0.7182 - val_accuracy: 0.8493
Epoch 92/100
32/32 [==============================] - 0s 4ms/step - loss: 0.7036 - accuracy: 0.8728 - val_loss: 0.7163 - val_accuracy: 0.8493
Epoch 93/100
32/32 [==============================] - 0s 5ms/step - loss: 0.7018 - accuracy: 0.8748 - val_loss: 0.7143 - val_accuracy: 0.8493
Epoch 94/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6999 - accuracy: 0.8718 - val_loss: 0.7133 - val_accuracy: 0.8539
Epoch 95/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6981 - accuracy: 0.8738 - val_loss: 0.7113 - val_accuracy: 0.8493
Epoch 96/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6964 - accuracy: 0.8738 - val_loss: 0.7100 - val_accuracy: 0.8539
```

```
Epoch 97/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6948 - accuracy: 0.8748 - val_loss: 0.7084 - val_accuracy: 0.8539
Epoch 98/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.8738 - val_loss: 0.7069 - val_accuracy: 0.8539
Epoch 99/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6916 - accuracy: 0.8738 - val_loss: 0.7055 - val_accuracy: 0.8539
Epoch 100/100
32/32 [==============================] - 0s 5ms/step - loss: 0.6902 - accuracy: 0.8748 - val_loss: 0.7045 - val_accuracy: 0.8539
```

Evaluating the Model

In [49]:
```
model_4.evaluate(X_test, Y_test)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.6999 - accuracy: 0.8813
```

Out[49]:
```
[0.6999451518058777, 0.8812785148620605]
```

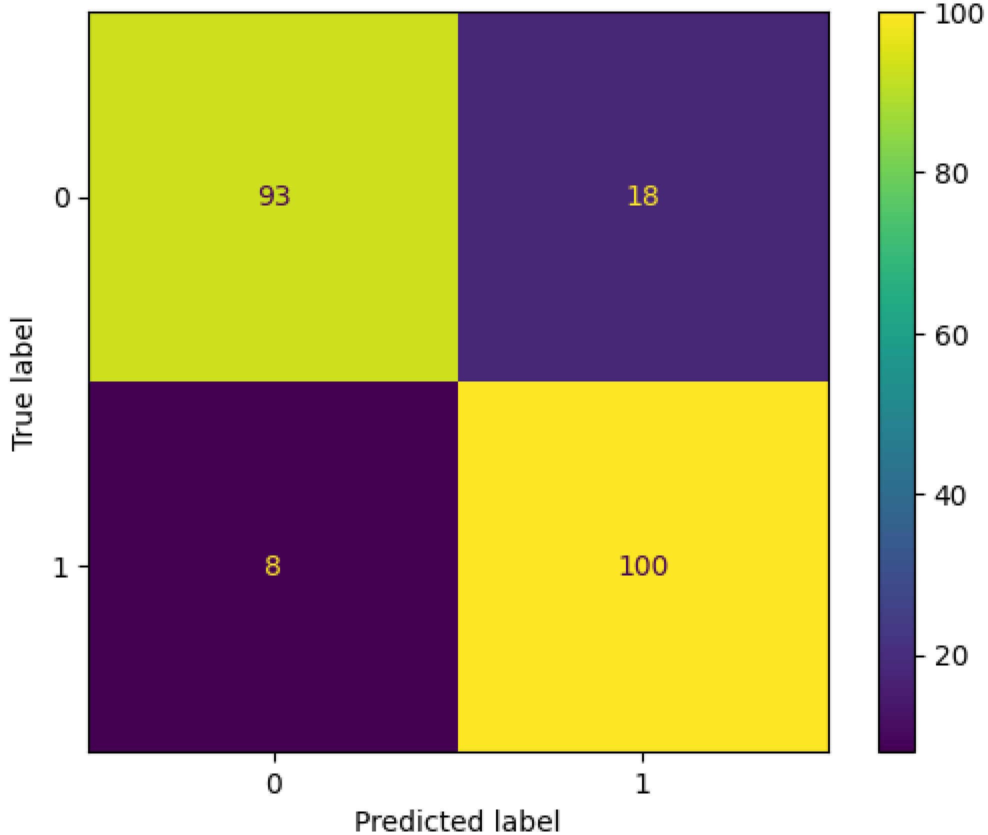
Confusion Matrix

In [50]:
```
Y_pred = model_4.predict(X_test).round()
y_pred = np.round(Y_pred, 0).tolist()
confusion_matrix(Y_test, y_pred)
```

```
7/7 [==============================] - 0s 3ms/step
```

Out[50]:
```
array([[ 93,  18],
       [  8, 100]], dtype=int64)
```

In [51]:
```
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



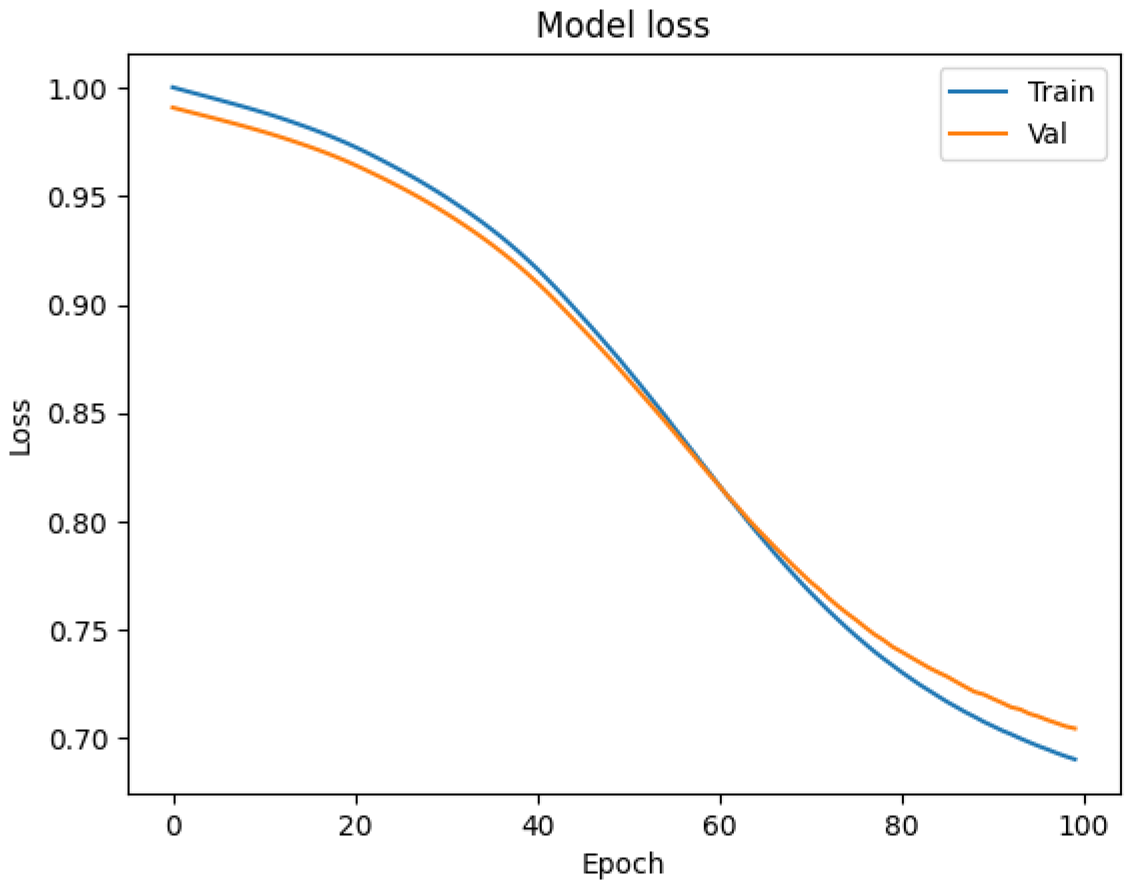
In [52]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.84      0.88       111
           1       0.85      0.93      0.88       108

    accuracy                           0.88       219
   macro avg       0.88      0.88      0.88       219
weighted avg       0.88      0.88      0.88       219
```

Visualizing Loss

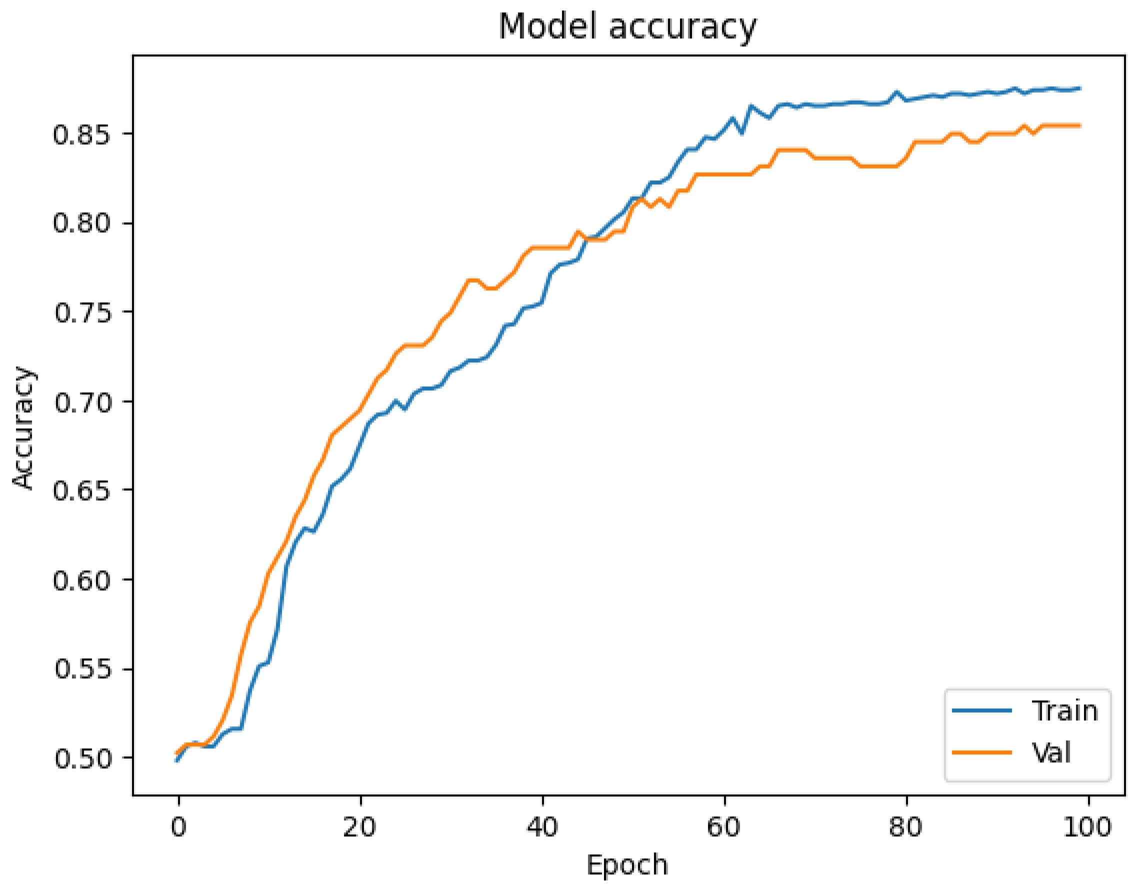
In [53]:
```
plt.plot(hist_test_4.history['loss'])
plt.plot(hist_test_4.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Visualizing Accuracy

In [54]:
```
plt.plot(hist_test_4.history['accuracy'])
plt.plot(hist_test_4.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



3.2.2 Using poisson as loss

Computes the Poisson loss between y_true and y_pred.

Neural Network Architecture

In [55]:
```
model_5 = Sequential([
    Dense(32, activation='relu', input_shape=(10,)), #32 is the size of the first layer, & 10 referes to the 10 input features
    Dense(32, activation='relu'), #32 is the size of the first layer
    Dense(1, activation='sigmoid'), #1 neuron as output layer for the prediction
])
```

In [56]:
```
model_5.compile(optimizer='sgd',
              loss='poisson', #chosen for our binary output
              metrics=['accuracy'])
```

Training the Model

In [57]:
```
hist_test_5 = model_5.fit(X_train, Y_train,
          batch_size=32, epochs=200,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/200
32/32 [==============================] - 1s 9ms/step - loss: 0.8460 - accuracy: 0.5538 - val_loss: 0.8523 - val_accuracy: 0.5297
Epoch 2/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8448 - accuracy: 0.5753 - val_loss: 0.8510 - val_accuracy: 0.5753
Epoch 3/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8435 - accuracy: 0.5812 - val_loss: 0.8498 - val_accuracy: 0.5753
Epoch 4/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8424 - accuracy: 0.5861 - val_loss: 0.8487 - val_accuracy: 0.5845
Epoch 5/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8414 - accuracy: 0.5959 - val_loss: 0.8478 - val_accuracy: 0.5890
Epoch 6/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8405 - accuracy: 0.6096 - val_loss: 0.8470 - val_accuracy: 0.6027
Epoch 7/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8399 - accuracy: 0.6370 - val_loss: 0.8463 - val_accuracy: 0.6210
Epoch 8/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8392 - accuracy: 0.6614 - val_loss: 0.8457 - val_accuracy: 0.6438
Epoch 9/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8386 - accuracy: 0.6781 - val_loss: 0.8451 - val_accuracy: 0.6530
Epoch 10/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8380 - accuracy: 0.6810 - val_loss: 0.8445 - val_accuracy: 0.6621
Epoch 11/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8374 - accuracy: 0.6977 - val_loss: 0.8439 - val_accuracy: 0.6758
Epoch 12/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8369 - accuracy: 0.7114 - val_loss: 0.8434 - val_accuracy: 0.6895
Epoch 13/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8363 - accuracy: 0.7231 - val_loss: 0.8428 - val_accuracy: 0.6941
Epoch 14/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8357 - accuracy: 0.7250 - val_loss: 0.8423 - val_accuracy: 0.7078
Epoch 15/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8352 - accuracy: 0.7387 - val_loss: 0.8417 - val_accuracy: 0.7169
Epoch 16/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8346 - accuracy: 0.7456 - val_loss: 0.8411 - val_accuracy: 0.7169
Epoch 17/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8340 - accuracy: 0.7495 - val_loss: 0.8405 - val_accuracy: 0.7260
Epoch 18/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8333 - accuracy: 0.7622 - val_loss: 0.8399 - val_accuracy: 0.7260
Epoch 19/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8327 - accuracy: 0.7671 - val_loss: 0.8393 - val_accuracy: 0.7397
Epoch 20/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8320 - accuracy: 0.7691 - val_loss: 0.8386 - val_accuracy: 0.7626
Epoch 21/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8313 - accuracy: 0.7750 - val_loss: 0.8380 - val_accuracy: 0.7671
Epoch 22/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8306 - accuracy: 0.7750 - val_loss: 0.8372 - val_accuracy: 0.7763
Epoch 23/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8298 - accuracy: 0.7798 - val_loss: 0.8365 - val_accuracy: 0.7900
Epoch 24/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8290 - accuracy: 0.7798 - val_loss: 0.8357 - val_accuracy: 0.7945
```

```
Epoch 25/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8281 - accuracy: 0.7808 - val_loss: 0.8349 - val_accuracy: 0.8037
Epoch 26/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8272 - accuracy: 0.7808 - val_loss: 0.8341 - val_accuracy: 0.7991
Epoch 27/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8263 - accuracy: 0.7838 - val_loss: 0.8332 - val_accuracy: 0.7991
Epoch 28/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8254 - accuracy: 0.7867 - val_loss: 0.8323 - val_accuracy: 0.7945
Epoch 29/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8244 - accuracy: 0.7808 - val_loss: 0.8314 - val_accuracy: 0.7854
Epoch 30/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8234 - accuracy: 0.7857 - val_loss: 0.8304 - val_accuracy: 0.7900
Epoch 31/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8224 - accuracy: 0.7935 - val_loss: 0.8294 - val_accuracy: 0.7945
Epoch 32/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8213 - accuracy: 0.7906 - val_loss: 0.8284 - val_accuracy: 0.7945
Epoch 33/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8202 - accuracy: 0.7896 - val_loss: 0.8273 - val_accuracy: 0.7945
Epoch 34/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8190 - accuracy: 0.8053 - val_loss: 0.8262 - val_accuracy: 0.7945
Epoch 35/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8178 - accuracy: 0.7994 - val_loss: 0.8251 - val_accuracy: 0.8082
Epoch 36/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8165 - accuracy: 0.7945 - val_loss: 0.8238 - val_accuracy: 0.8082
Epoch 37/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8152 - accuracy: 0.7984 - val_loss: 0.8226 - val_accuracy: 0.8128
Epoch 38/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8138 - accuracy: 0.8053 - val_loss: 0.8213 - val_accuracy: 0.8082
Epoch 39/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8124 - accuracy: 0.8092 - val_loss: 0.8200 - val_accuracy: 0.8082
Epoch 40/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8109 - accuracy: 0.8082 - val_loss: 0.8186 - val_accuracy: 0.8219
Epoch 41/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8094 - accuracy: 0.8131 - val_loss: 0.8172 - val_accuracy: 0.8219
Epoch 42/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8078 - accuracy: 0.8112 - val_loss: 0.8158 - val_accuracy: 0.8128
Epoch 43/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8062 - accuracy: 0.8121 - val_loss: 0.8142 - val_accuracy: 0.8128
Epoch 44/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8045 - accuracy: 0.8151 - val_loss: 0.8126 - val_accuracy: 0.8128
Epoch 45/200
32/32 [==============================] - 0s 4ms/step - loss: 0.8028 - accuracy: 0.8170 - val_loss: 0.8110 - val_accuracy: 0.8128
Epoch 46/200
32/32 [==============================] - 0s 5ms/step - loss: 0.8009 - accuracy: 0.8190 - val_loss: 0.8094 - val_accuracy: 0.8128
Epoch 47/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7990 - accuracy: 0.8200 - val_loss: 0.8077 - val_accuracy: 0.8128
Epoch 48/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7971 - accuracy: 0.8190 - val_loss: 0.8059 - val_accuracy: 0.8265
```

```
Epoch 49/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7951 - accuracy: 0.8200 - val_loss: 0.8041 - val_accuracy: 0.8265
Epoch 50/200
32/32 [==============================] - 0s 4ms/step - loss: 0.7930 - accuracy: 0.8190 - val_loss: 0.8022 - val_accuracy: 0.8219
Epoch 51/200
32/32 [==============================] - 0s 4ms/step - loss: 0.7909 - accuracy: 0.8170 - val_loss: 0.8003 - val_accuracy: 0.8219
Epoch 52/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7888 - accuracy: 0.8170 - val_loss: 0.7983 - val_accuracy: 0.8174
Epoch 53/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7866 - accuracy: 0.8200 - val_loss: 0.7963 - val_accuracy: 0.8219
Epoch 54/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7843 - accuracy: 0.8200 - val_loss: 0.7942 - val_accuracy: 0.8219
Epoch 55/200
32/32 [==============================] - 0s 8ms/step - loss: 0.7819 - accuracy: 0.8170 - val_loss: 0.7920 - val_accuracy: 0.8265
Epoch 56/200
32/32 [==============================] - 0s 7ms/step - loss: 0.7796 - accuracy: 0.8229 - val_loss: 0.7899 - val_accuracy: 0.8265
Epoch 57/200
32/32 [==============================] - 0s 7ms/step - loss: 0.7771 - accuracy: 0.8239 - val_loss: 0.7876 - val_accuracy: 0.8265
Epoch 58/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7746 - accuracy: 0.8278 - val_loss: 0.7854 - val_accuracy: 0.8265
Epoch 59/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7721 - accuracy: 0.8278 - val_loss: 0.7831 - val_accuracy: 0.8265
Epoch 60/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7695 - accuracy: 0.8346 - val_loss: 0.7808 - val_accuracy: 0.8265
Epoch 61/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7669 - accuracy: 0.8297 - val_loss: 0.7783 - val_accuracy: 0.8356
Epoch 62/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7642 - accuracy: 0.8366 - val_loss: 0.7760 - val_accuracy: 0.8356
Epoch 63/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7615 - accuracy: 0.8346 - val_loss: 0.7736 - val_accuracy: 0.8311
Epoch 64/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7588 - accuracy: 0.8356 - val_loss: 0.7711 - val_accuracy: 0.8356
Epoch 65/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7560 - accuracy: 0.8386 - val_loss: 0.7686 - val_accuracy: 0.8356
Epoch 66/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7532 - accuracy: 0.8395 - val_loss: 0.7661 - val_accuracy: 0.8356
Epoch 67/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7504 - accuracy: 0.8376 - val_loss: 0.7634 - val_accuracy: 0.8402
Epoch 68/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7476 - accuracy: 0.8405 - val_loss: 0.7609 - val_accuracy: 0.8356
Epoch 69/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7446 - accuracy: 0.8434 - val_loss: 0.7584 - val_accuracy: 0.8356
Epoch 70/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7418 - accuracy: 0.8415 - val_loss: 0.7558 - val_accuracy: 0.8356
Epoch 71/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7389 - accuracy: 0.8415 - val_loss: 0.7530 - val_accuracy: 0.8402
Epoch 72/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7362 - accuracy: 0.8454 - val_loss: 0.7505 - val_accuracy: 0.8402
```

```
Epoch 73/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7335 - accuracy: 0.8434 - val_loss: 0.7483 - val_accuracy: 0.8402
Epoch 74/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7308 - accuracy: 0.8425 - val_loss: 0.7460 - val_accuracy: 0.8356
Epoch 75/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7282 - accuracy: 0.8366 - val_loss: 0.7433 - val_accuracy: 0.8402
Epoch 76/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7255 - accuracy: 0.8474 - val_loss: 0.7413 - val_accuracy: 0.8356
Epoch 77/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7230 - accuracy: 0.8425 - val_loss: 0.7392 - val_accuracy: 0.8265
Epoch 78/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7205 - accuracy: 0.8464 - val_loss: 0.7370 - val_accuracy: 0.8311
Epoch 79/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7180 - accuracy: 0.8483 - val_loss: 0.7349 - val_accuracy: 0.8311
Epoch 80/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7157 - accuracy: 0.8523 - val_loss: 0.7329 - val_accuracy: 0.8265
Epoch 81/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7133 - accuracy: 0.8493 - val_loss: 0.7306 - val_accuracy: 0.8265
Epoch 82/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7111 - accuracy: 0.8513 - val_loss: 0.7283 - val_accuracy: 0.8356
Epoch 83/200
32/32 [==============================] - 0s 7ms/step - loss: 0.7088 - accuracy: 0.8532 - val_loss: 0.7267 - val_accuracy: 0.8265
Epoch 84/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7067 - accuracy: 0.8493 - val_loss: 0.7244 - val_accuracy: 0.8356
Epoch 85/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7045 - accuracy: 0.8542 - val_loss: 0.7224 - val_accuracy: 0.8356
Epoch 86/200
32/32 [==============================] - 0s 6ms/step - loss: 0.7025 - accuracy: 0.8552 - val_loss: 0.7208 - val_accuracy: 0.8356
Epoch 87/200
32/32 [==============================] - 0s 5ms/step - loss: 0.7005 - accuracy: 0.8581 - val_loss: 0.7196 - val_accuracy: 0.8265
Epoch 88/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6986 - accuracy: 0.8542 - val_loss: 0.7177 - val_accuracy: 0.8265
Epoch 89/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6968 - accuracy: 0.8562 - val_loss: 0.7159 - val_accuracy: 0.8265
Epoch 90/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6950 - accuracy: 0.8571 - val_loss: 0.7144 - val_accuracy: 0.8265
Epoch 91/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6933 - accuracy: 0.8581 - val_loss: 0.7129 - val_accuracy: 0.8219
Epoch 92/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6915 - accuracy: 0.8542 - val_loss: 0.7120 - val_accuracy: 0.8265
Epoch 93/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6899 - accuracy: 0.8562 - val_loss: 0.7103 - val_accuracy: 0.8219
Epoch 94/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6883 - accuracy: 0.8571 - val_loss: 0.7086 - val_accuracy: 0.8219
Epoch 95/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6869 - accuracy: 0.8591 - val_loss: 0.7079 - val_accuracy: 0.8219
Epoch 96/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6854 - accuracy: 0.8620 - val_loss: 0.7059 - val_accuracy: 0.8265
```

```
Epoch 97/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6839 - accuracy: 0.8591 - val_loss: 0.7044 - val_accuracy: 0.8402
Epoch 98/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6826 - accuracy: 0.8611 - val_loss: 0.7035 - val_accuracy: 0.8311
Epoch 99/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6813 - accuracy: 0.8591 - val_loss: 0.7025 - val_accuracy: 0.8356
Epoch 100/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6801 - accuracy: 0.8591 - val_loss: 0.7014 - val_accuracy: 0.8356
Epoch 101/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6787 - accuracy: 0.8611 - val_loss: 0.7007 - val_accuracy: 0.8356
Epoch 102/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6777 - accuracy: 0.8620 - val_loss: 0.6992 - val_accuracy: 0.8493
Epoch 103/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6764 - accuracy: 0.8601 - val_loss: 0.6976 - val_accuracy: 0.8447
Epoch 104/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6753 - accuracy: 0.8611 - val_loss: 0.6967 - val_accuracy: 0.8447
Epoch 105/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6742 - accuracy: 0.8620 - val_loss: 0.6955 - val_accuracy: 0.8447
Epoch 106/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6733 - accuracy: 0.8650 - val_loss: 0.6954 - val_accuracy: 0.8493
Epoch 107/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6722 - accuracy: 0.8650 - val_loss: 0.6945 - val_accuracy: 0.8493
Epoch 108/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6712 - accuracy: 0.8640 - val_loss: 0.6930 - val_accuracy: 0.8493
Epoch 109/200
32/32 [==============================] - 0s 7ms/step - loss: 0.6702 - accuracy: 0.8659 - val_loss: 0.6933 - val_accuracy: 0.8402
Epoch 110/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6694 - accuracy: 0.8669 - val_loss: 0.6918 - val_accuracy: 0.8493
Epoch 111/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6684 - accuracy: 0.8650 - val_loss: 0.6902 - val_accuracy: 0.8493
Epoch 112/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6677 - accuracy: 0.8669 - val_loss: 0.6901 - val_accuracy: 0.8539
Epoch 113/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6668 - accuracy: 0.8679 - val_loss: 0.6897 - val_accuracy: 0.8539
Epoch 114/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6661 - accuracy: 0.8708 - val_loss: 0.6894 - val_accuracy: 0.8539
Epoch 115/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6651 - accuracy: 0.8689 - val_loss: 0.6880 - val_accuracy: 0.8539
Epoch 116/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6645 - accuracy: 0.8689 - val_loss: 0.6879 - val_accuracy: 0.8539
Epoch 117/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6637 - accuracy: 0.8679 - val_loss: 0.6873 - val_accuracy: 0.8539
Epoch 118/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6630 - accuracy: 0.8659 - val_loss: 0.6863 - val_accuracy: 0.8539
Epoch 119/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6624 - accuracy: 0.8708 - val_loss: 0.6855 - val_accuracy: 0.8539
Epoch 120/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6616 - accuracy: 0.8699 - val_loss: 0.6857 - val_accuracy: 0.8539
```

```
Epoch 121/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6610 - accuracy: 0.8699 - val_loss: 0.6841 - val_accuracy: 0.8539
Epoch 122/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6603 - accuracy: 0.8679 - val_loss: 0.6840 - val_accuracy: 0.8539
Epoch 123/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6598 - accuracy: 0.8689 - val_loss: 0.6830 - val_accuracy: 0.8539
Epoch 124/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6592 - accuracy: 0.8679 - val_loss: 0.6825 - val_accuracy: 0.8539
Epoch 125/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6586 - accuracy: 0.8699 - val_loss: 0.6824 - val_accuracy: 0.8539
Epoch 126/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6580 - accuracy: 0.8699 - val_loss: 0.6823 - val_accuracy: 0.8539
Epoch 127/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6574 - accuracy: 0.8679 - val_loss: 0.6808 - val_accuracy: 0.8493
Epoch 128/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6568 - accuracy: 0.8718 - val_loss: 0.6815 - val_accuracy: 0.8539
Epoch 129/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6564 - accuracy: 0.8689 - val_loss: 0.6804 - val_accuracy: 0.8493
Epoch 130/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6559 - accuracy: 0.8708 - val_loss: 0.6803 - val_accuracy: 0.8539
Epoch 131/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6552 - accuracy: 0.8659 - val_loss: 0.6789 - val_accuracy: 0.8447
Epoch 132/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6549 - accuracy: 0.8728 - val_loss: 0.6796 - val_accuracy: 0.8493
Epoch 133/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6543 - accuracy: 0.8718 - val_loss: 0.6780 - val_accuracy: 0.8493
Epoch 134/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6537 - accuracy: 0.8699 - val_loss: 0.6788 - val_accuracy: 0.8493
Epoch 135/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6534 - accuracy: 0.8728 - val_loss: 0.6787 - val_accuracy: 0.8493
Epoch 136/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6529 - accuracy: 0.8718 - val_loss: 0.6778 - val_accuracy: 0.8493
Epoch 137/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6524 - accuracy: 0.8718 - val_loss: 0.6774 - val_accuracy: 0.8493
Epoch 138/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6520 - accuracy: 0.8728 - val_loss: 0.6769 - val_accuracy: 0.8539
Epoch 139/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6514 - accuracy: 0.8718 - val_loss: 0.6751 - val_accuracy: 0.8584
Epoch 140/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6513 - accuracy: 0.8699 - val_loss: 0.6751 - val_accuracy: 0.8584
Epoch 141/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6508 - accuracy: 0.8689 - val_loss: 0.6753 - val_accuracy: 0.8539
Epoch 142/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6503 - accuracy: 0.8728 - val_loss: 0.6749 - val_accuracy: 0.8539
Epoch 143/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6498 - accuracy: 0.8757 - val_loss: 0.6748 - val_accuracy: 0.8539
Epoch 144/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6496 - accuracy: 0.8718 - val_loss: 0.6744 - val_accuracy: 0.8539
```

```
Epoch 145/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6491 - accuracy: 0.8738 - val_loss: 0.6741 - val_accuracy: 0.8539
Epoch 146/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6488 - accuracy: 0.8738 - val_loss: 0.6729 - val_accuracy: 0.8584
Epoch 147/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6484 - accuracy: 0.8738 - val_loss: 0.6734 - val_accuracy: 0.8539
Epoch 148/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6479 - accuracy: 0.8718 - val_loss: 0.6733 - val_accuracy: 0.8539
Epoch 149/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6477 - accuracy: 0.8738 - val_loss: 0.6727 - val_accuracy: 0.8584
Epoch 150/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6473 - accuracy: 0.8728 - val_loss: 0.6729 - val_accuracy: 0.8539
Epoch 151/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6469 - accuracy: 0.8728 - val_loss: 0.6718 - val_accuracy: 0.8584
Epoch 152/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6467 - accuracy: 0.8738 - val_loss: 0.6719 - val_accuracy: 0.8584
Epoch 153/200
32/32 [==============================] - 0s 7ms/step - loss: 0.6464 - accuracy: 0.8718 - val_loss: 0.6713 - val_accuracy: 0.8584
Epoch 154/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6458 - accuracy: 0.8757 - val_loss: 0.6707 - val_accuracy: 0.8584
Epoch 155/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6456 - accuracy: 0.8777 - val_loss: 0.6708 - val_accuracy: 0.8584
Epoch 156/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6453 - accuracy: 0.8767 - val_loss: 0.6706 - val_accuracy: 0.8584
Epoch 157/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6450 - accuracy: 0.8748 - val_loss: 0.6703 - val_accuracy: 0.8584
Epoch 158/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6447 - accuracy: 0.8748 - val_loss: 0.6696 - val_accuracy: 0.8584
Epoch 159/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6443 - accuracy: 0.8777 - val_loss: 0.6700 - val_accuracy: 0.8584
Epoch 160/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6439 - accuracy: 0.8796 - val_loss: 0.6697 - val_accuracy: 0.8584
Epoch 161/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6437 - accuracy: 0.8757 - val_loss: 0.6693 - val_accuracy: 0.8584
Epoch 162/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6433 - accuracy: 0.8777 - val_loss: 0.6691 - val_accuracy: 0.8584
Epoch 163/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6431 - accuracy: 0.8767 - val_loss: 0.6687 - val_accuracy: 0.8584
Epoch 164/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6425 - accuracy: 0.8826 - val_loss: 0.6668 - val_accuracy: 0.8584
Epoch 165/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6426 - accuracy: 0.8806 - val_loss: 0.6673 - val_accuracy: 0.8584
Epoch 166/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6422 - accuracy: 0.8767 - val_loss: 0.6680 - val_accuracy: 0.8584
Epoch 167/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6418 - accuracy: 0.8796 - val_loss: 0.6671 - val_accuracy: 0.8584
Epoch 168/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6417 - accuracy: 0.8777 - val_loss: 0.6682 - val_accuracy: 0.8539
```

```
Epoch 169/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6413 - accuracy: 0.8796 - val_loss: 0.6664 - val_accuracy: 0.8584
Epoch 170/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6411 - accuracy: 0.8787 - val_loss: 0.6674 - val_accuracy: 0.8584
Epoch 171/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6409 - accuracy: 0.8816 - val_loss: 0.6660 - val_accuracy: 0.8584
Epoch 172/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6406 - accuracy: 0.8836 - val_loss: 0.6659 - val_accuracy: 0.8584
Epoch 173/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6403 - accuracy: 0.8806 - val_loss: 0.6659 - val_accuracy: 0.8584
Epoch 174/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6401 - accuracy: 0.8816 - val_loss: 0.6658 - val_accuracy: 0.8584
Epoch 175/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6397 - accuracy: 0.8816 - val_loss: 0.6651 - val_accuracy: 0.8584
Epoch 176/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6397 - accuracy: 0.8816 - val_loss: 0.6654 - val_accuracy: 0.8584
Epoch 177/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6391 - accuracy: 0.8836 - val_loss: 0.6659 - val_accuracy: 0.8584
Epoch 178/200
32/32 [==============================] - 0s 6ms/step - loss: 0.6390 - accuracy: 0.8836 - val_loss: 0.6660 - val_accuracy: 0.8584
Epoch 179/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6387 - accuracy: 0.8836 - val_loss: 0.6643 - val_accuracy: 0.8584
Epoch 180/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6386 - accuracy: 0.8826 - val_loss: 0.6649 - val_accuracy: 0.8584
Epoch 181/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6383 - accuracy: 0.8816 - val_loss: 0.6641 - val_accuracy: 0.8584
Epoch 182/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6382 - accuracy: 0.8845 - val_loss: 0.6640 - val_accuracy: 0.8584
Epoch 183/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6379 - accuracy: 0.8826 - val_loss: 0.6641 - val_accuracy: 0.8584
Epoch 184/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6376 - accuracy: 0.8826 - val_loss: 0.6632 - val_accuracy: 0.8584
Epoch 185/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6375 - accuracy: 0.8826 - val_loss: 0.6631 - val_accuracy: 0.8584
Epoch 186/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6371 - accuracy: 0.8845 - val_loss: 0.6626 - val_accuracy: 0.8584
Epoch 187/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6370 - accuracy: 0.8855 - val_loss: 0.6624 - val_accuracy: 0.8584
Epoch 188/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6368 - accuracy: 0.8806 - val_loss: 0.6628 - val_accuracy: 0.8584
Epoch 189/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6364 - accuracy: 0.8845 - val_loss: 0.6614 - val_accuracy: 0.8584
Epoch 190/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6365 - accuracy: 0.8855 - val_loss: 0.6618 - val_accuracy: 0.8584
Epoch 191/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6361 - accuracy: 0.8826 - val_loss: 0.6622 - val_accuracy: 0.8584
Epoch 192/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6358 - accuracy: 0.8845 - val_loss: 0.6615 - val_accuracy: 0.8584
```

```
Epoch 193/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6354 - accuracy: 0.8845 - val_loss: 0.6633 - val_accuracy: 0.8539
Epoch 194/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6356 - accuracy: 0.8836 - val_loss: 0.6619 - val_accuracy: 0.8584
Epoch 195/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6353 - accuracy: 0.8826 - val_loss: 0.6617 - val_accuracy: 0.8584
Epoch 196/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6351 - accuracy: 0.8836 - val_loss: 0.6618 - val_accuracy: 0.8584
Epoch 197/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6348 - accuracy: 0.8845 - val_loss: 0.6613 - val_accuracy: 0.8584
Epoch 198/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6348 - accuracy: 0.8845 - val_loss: 0.6603 - val_accuracy: 0.8584
Epoch 199/200
32/32 [==============================] - 0s 5ms/step - loss: 0.6346 - accuracy: 0.8855 - val_loss: 0.6606 - val_accuracy: 0.8584
Epoch 200/200
32/32 [==============================] - 0s 4ms/step - loss: 0.6343 - accuracy: 0.8836 - val_loss: 0.6616 - val_accuracy: 0.8584
```

Evaluating the Model

In [58]:
```
model_5.evaluate(X_test, Y_test)
```

```
7/7 [==============================] - 0s 3ms/step - loss: 0.6165 - accuracy: 0.8767
```

Out[58]:
```
[0.6165306568145752, 0.8767123222351074]
```

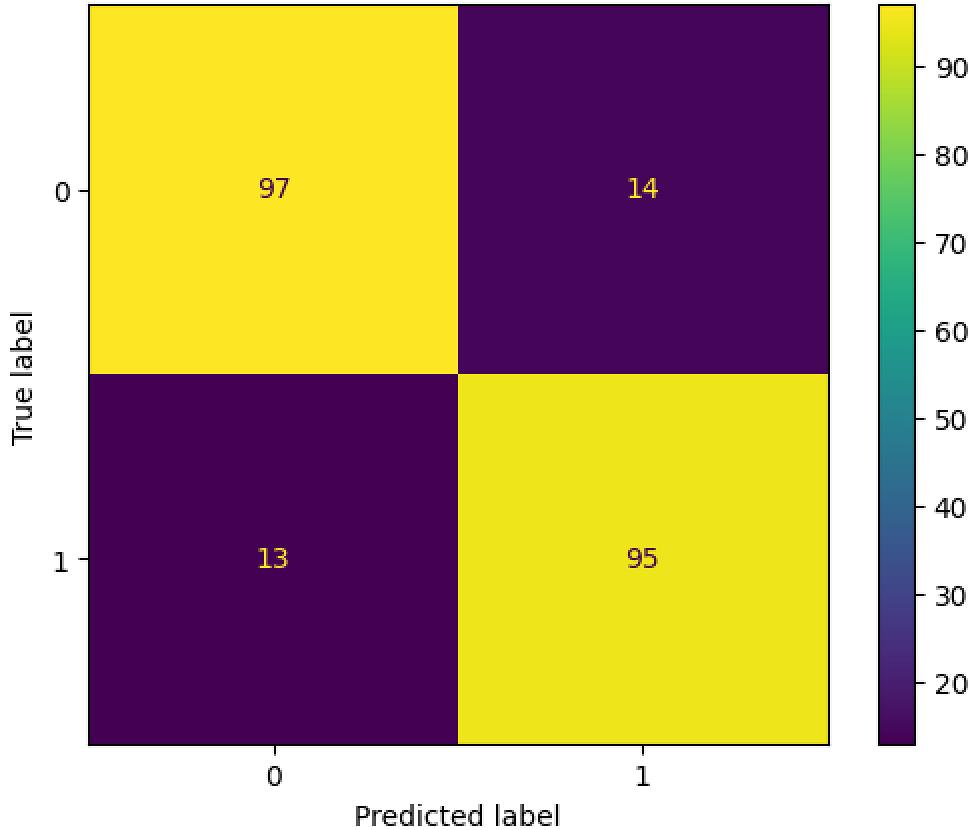
Confusion Matrix

In [59]:
```
Y_pred = model_5.predict(X_test).round()
y_pred = np.round(Y_pred, 0).tolist()
confusion_matrix(Y_test, y_pred)
```

```
7/7 [==============================] - 0s 2ms/step
```

Out[59]:
```
array([[97, 14],
       [13, 95]], dtype=int64)
```

In [60]:
```
cm = confusion_matrix(Y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

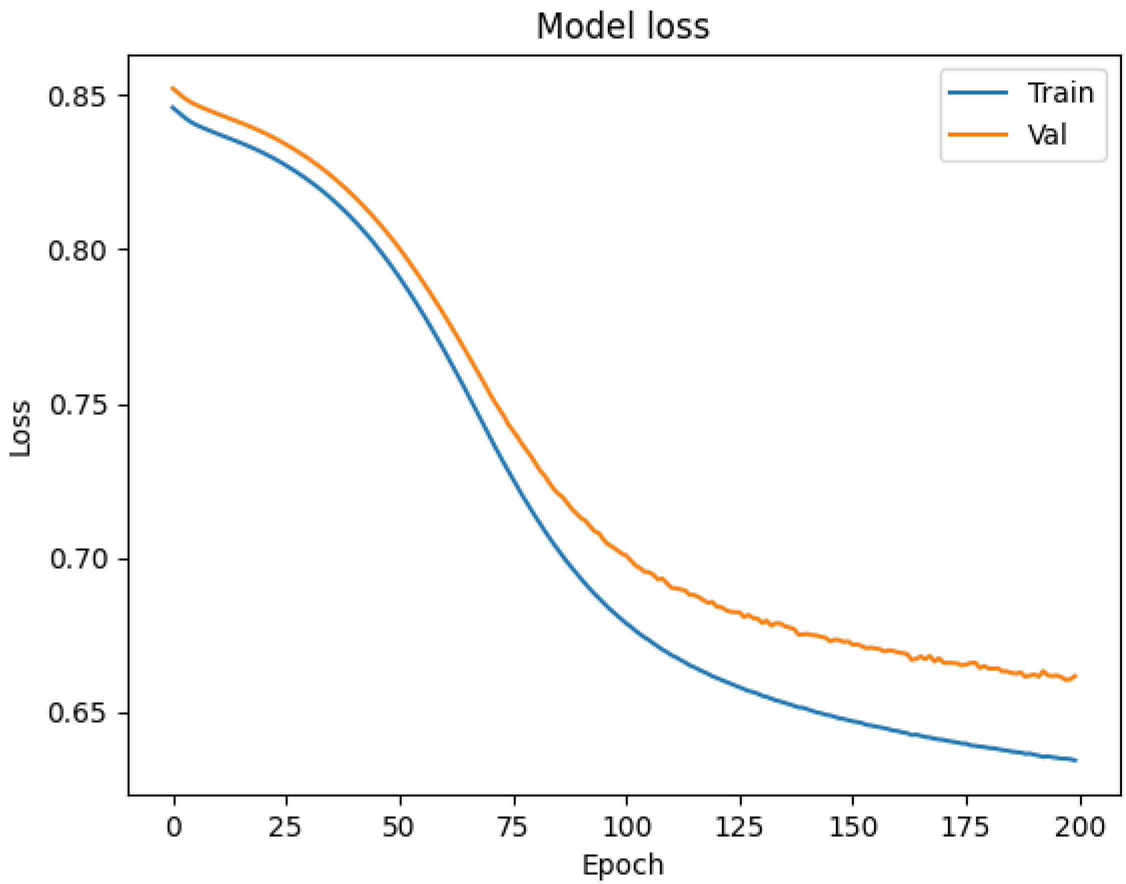


In [61]:
```
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       111
           1       0.87      0.88      0.88       108

    accuracy                           0.88       219
   macro avg       0.88      0.88      0.88       219
weighted avg       0.88      0.88      0.88       219
```

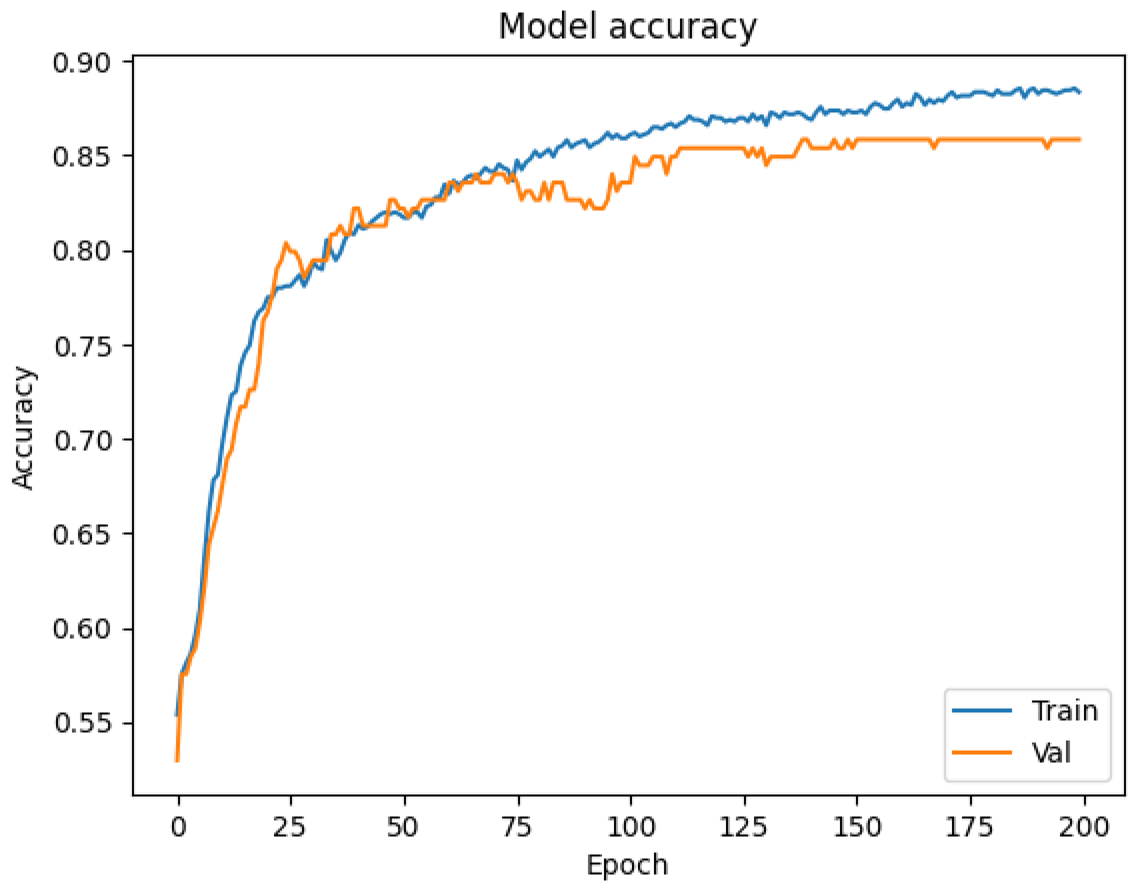
Visualizing Loss

In [62]:
```
plt.plot(hist_test_5.history['loss'])
plt.plot(hist_test_5.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```



Visualizing Accuracy

In [63]:
```
plt.plot(hist_test_5.history['accuracy'])
plt.plot(hist_test_5.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



3.2.3. Conclusion

We have tried out 2 different loss function to compare to the Binary Crossentropy function which are the poisson and categorical hinge functions. Based on the finding, all three give a similiar overal accuracy but comparing the losses which they are, the Binary Crossentropy function performs the best out of the 3 for this data.

3.3. Conclusion on Hypertuning the model

Based on hypertuning the loss function and optimizers we can conclude that the original set hyperparamaters used (Binary Crossentropy & SGD) were the best in comparison to the hyperparameters tried onto the model. The results overall with hypertuning the model came close but were the best ones being the orginal set hyper parameters which we have concluded based on looking at the loss, accuracy and confusion matrixes.

Reflection

To reflect on this exercise, I have learned how to apply my knowledge and skills in creating a neural network on a random dataset. Although I encountered some difficulties, such as applying the confusion matrix and optimizers, I persisted and eventually gained a better understanding of these concepts.

Additionally, I gained knowledge on new techniques such as Regularization, Early Stopping & Dropout to prevent overfitting of the neural network. Although my neural network in this exercise did not suffer from overfitting, I now have a better understanding of how to address this issue if it were to occur in the future.

Overall, this exercise has provided me with valuable experience and knowledge that I can apply in future projects involving neural networks and machine learning.

Sources

data source that the project is based on: https://www.kaggle.com/c/zillow-prize-1/data

reformed data source: https://drive.google.com/file/d/1GfvKA0qznNVknghV4botnNxyH-KvODOC/view

tutorial/code used to help: https://www.freecodecamp.org/news/how-to-build-your-first-neural-network-to-predict-house-prices-with-keras-f8db83049159/

Helped to set the prediction of the neural network to the right format tot use with also sigmoid included: https://stackoverflow.com/questions/73199505/confusion-matrix-for-binary-classification-with-nn

Steps used for applying the confusion matrix: https://www.jcchouinard.com/confusion-matrix-in-scikit-learn/