Coding Companion for Intuitive Deep Learning Part 1 (Annotated)

Code Author: Joseph Lee Wei En

Additional Annotations in Italics (or additional Python comments) by Kayla Bandy

Date Annotated: 11/25/22

In this notebook, we'll go through the code for the coding companion for Intuitive Deep Learning Part 1 (Part 1a, Part 1b) to create your very first neural network to predict whether the house price is below or above median value. We will go through the following in this notebook:

- Exploring and Processing the Data
- Building and Training our Neural Network
- Visualizing Loss and Accuracy
- Adding Regularization to our Neural Network

The code is annotated throughout the notebook and you simply need to download the dataset here, put the dataset in the same folder as this notebook and run the code cells below. Note that the results you get might differ slightly from the blogpost as there is a degree of randomness in the way we split our dataset as well as the initialization of our neural network.

This Jupyter Notebook will take us through 3 neural network examples, with the first one being relatively simple. The 2nd example shows exagerrated overfitting while the 3rd example takes the 2nd example and implements some common strategies to reduce over-fitting.

Exploring and Processing the Data

We first have to read in the CSV file that we've been given. We'll use a package called pandas for that:

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

In [2]:
```
df = pd.read_csv('housepricedata.csv')
```

In [3]:
```
#View some of the dataset to see the structure of the data
#Measurements are in sq ft
df
```

Out[3]:

	LotArea	OverallQual	OverallCond	TotalBsmtSF	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Firep
0	8450	7	5	856	2	1	3	8	
1	9600	6	8	1262	2	0	3	6	
2	11250	7	5	920	2	1	3	6	
3	9550	7	5	756	1	0	3	7	
4	14260	8	5	1145	2	1	4	9	
...	...	...	...	...	...	...	...	...	
1455	7917	6	5	953	2	1	3	7	
1456	13175	6	6	1542	2	0	3	7	
1457	9042	7	9	1152	2	0	4	9	
1458	9717	5	6	1078	1	0	2	5	
1459	9937	5	6	1256	1	1	3	6	

1460 rows × 11 columns

The dataset that we have now is in what we call a pandas dataframe. To convert it to an array, simply access its values:

In [4]:
```
#Create an array from the DataFrame values
dataset = df.values
```

In [5]:
```
#View some of the dataset to see the structure of the data
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)
```

Now, we split the dataset into our input features and the label we wish to predict.

In [6]:
```
#Select the first 10 columns into input features (X)
#Select the 11th column into the predictor label (Y)
X = dataset[:,0:10]
Y = dataset[:,10]
```

Normalizing our data is very important, as we want the input features to be on the same order of magnitude to make our training easier. We'll use a min-max scaler from scikit-learn which scales our data to be between 0 and 1.

Rescaling variables is conceptually similar to Z-scores from statistics, where a value is displayed in relation to the mean of a group of values.

In [7]:
```
from sklearn import preprocessing
```

In [8]:
```
#This function will scale or convert the dataset's features
#to values between 0 and 1
min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)
```

In [9]:
```
#View how the data changed after scaling
X_scale
```

Out[9]:
```
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]])
```

Lastly, we wish to set aside some parts of our dataset for a validation set and a test set. We use the function train_test_split from scikit-learn to do that.

We use the train_test_split function twice since we want to sub-split the validation and test set.

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
#Split total data into 70% training and 30% validation & test
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)
```

In [12]:
```
#Split validation & test dataset into 50% validation and test each
#of the original 30% dataset (15% each)
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)
```

In [13]:
```
#View the dimensions of the resulting datasets
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,)
```

Building and Training Our First Neural Network

We will be using Keras to build our architecture. Let's import the code from Keras that we will need to use:

In [14]:
```
from keras.models import Sequential
from keras.layers import Dense
```

We will be using the Sequential model, which means that we merely need to describe the layers above in sequence. Our neural network has three layers:

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

In [15]:
```
#This model will have 2 hidden layers with 32 neurons each
#ReLU Function: If output is negative, then output 0. Otherwise, do nothing
#Sigmoid Function: Put the initial output between 0 and 1
model = Sequential([
    Dense(32, activation='relu', input_shape=(10,)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid'),
])
```

Now that we've got our architecture specified, we need to find the best numbers for it. Before we start our training, we have to configure the model by

- Telling it what algorithm you want to use to do the optimization (we'll use stochastic gradient descent)
- Telling it what loss function to use (for binary classification, we will use binary cross entropy)
- Telling it what other metrics you want to track apart from the loss function (we want to track accuracy as well)

Loss functions numerically punish bad predictions and optimization algorithms are meant to minimize losses.

We do so below:

In [16]:
```
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['acc'])
```

Training on the data is pretty straightforward and requires us to write one line of code. The function is called 'fit' as we are fitting the parameters to the data. We specify:

- what data we are training on, which is X_train and Y_train
- the size of our mini-batch
- how long we want to train it for (epochs)
- what our validation data is so that the model will tell us how we are doing on the validation data at each point.

This function will output a history, which we save under the variable hist. We'll use this variable a little later.

In [17]:
```
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 1s 10ms/step - loss: 0.7114 - acc: 0.5176 - val_loss:
0.7186 - val_acc: 0.4612
Epoch 2/100
32/32 [==============================] - 0s 3ms/step - loss: 0.7011 - acc: 0.5176 - val_loss: 0.
7063 - val_acc: 0.4612
Epoch 3/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6947 - acc: 0.5176 - val_loss: 0.
6982 - val_acc: 0.4612
Epoch 4/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6900 - acc: 0.5166 - val_loss: 0.
6918 - val_acc: 0.4612
Epoch 5/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6862 - acc: 0.5157 - val_loss: 0.
6864 - val_acc: 0.4521
Epoch 6/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6827 - acc: 0.5372 - val_loss: 0.
6819 - val_acc: 0.5205
Epoch 7/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6796 - acc: 0.6037 - val_loss: 0.
6776 - val_acc: 0.6484
Epoch 8/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6764 - acc: 0.6556 - val_loss: 0.
6735 - val_acc: 0.6986
Epoch 9/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6733 - acc: 0.6898 - val_loss: 0.
6692 - val_acc: 0.7352
Epoch 10/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6699 - acc: 0.7094 - val_loss: 0.
6643 - val_acc: 0.7717
Epoch 11/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6660 - acc: 0.7329 - val_loss: 0.
6594 - val_acc: 0.7808
Epoch 12/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6622 - acc: 0.7436 - val_loss: 0.
6544 - val_acc: 0.7808
Epoch 13/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6584 - acc: 0.7485 - val_loss: 0.
6497 - val_acc: 0.7854
Epoch 14/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6544 - acc: 0.7593 - val_loss: 0.
6450 - val_acc: 0.7991
Epoch 15/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6504 - acc: 0.7603 - val_loss: 0.
6403 - val_acc: 0.7991
Epoch 16/100
32/32 [==============================] - 0s 4ms/step - loss: 0.6463 - acc: 0.7593 - val_loss: 0.
6352 - val_acc: 0.8082
Epoch 17/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6421 - acc: 0.7701 - val_loss: 0.
6304 - val_acc: 0.7900
Epoch 18/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6376 - acc: 0.7652 - val_loss: 0.
6253 - val_acc: 0.7945
Epoch 19/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6331 - acc: 0.7691 - val_loss: 0.
6196 - val_acc: 0.8174
Epoch 20/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6284 - acc: 0.7740 - val_loss: 0.
6140 - val_acc: 0.8219
Epoch 21/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6233 - acc: 0.7750 - val_loss: 0.
```

```
6078 - val_acc: 0.8265
Epoch 22/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6182 - acc: 0.7818 - val_loss: 0.
6017 - val_acc: 0.8311
Epoch 23/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6127 - acc: 0.7847 - val_loss: 0.
5957 - val_acc: 0.8311
Epoch 24/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6071 - acc: 0.7828 - val_loss: 0.
5893 - val_acc: 0.8265
Epoch 25/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6012 - acc: 0.7886 - val_loss: 0.
5828 - val_acc: 0.8174
Epoch 26/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5949 - acc: 0.7867 - val_loss: 0.
5748 - val_acc: 0.8402
Epoch 27/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5883 - acc: 0.7896 - val_loss: 0.
5672 - val_acc: 0.8447
Epoch 28/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5816 - acc: 0.7896 - val_loss: 0.
5593 - val_acc: 0.8447
Epoch 29/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5746 - acc: 0.7886 - val_loss: 0.
5511 - val_acc: 0.8447
Epoch 30/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5676 - acc: 0.7945 - val_loss: 0.
5441 - val_acc: 0.8402
Epoch 31/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5604 - acc: 0.7945 - val_loss: 0.
5344 - val_acc: 0.8447
Epoch 32/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5528 - acc: 0.7945 - val_loss: 0.
5257 - val_acc: 0.8493
Epoch 33/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5451 - acc: 0.7955 - val_loss: 0.
5173 - val_acc: 0.8539
Epoch 34/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5371 - acc: 0.7945 - val_loss: 0.
5071 - val_acc: 0.8584
Epoch 35/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5294 - acc: 0.8043 - val_loss: 0.
4992 - val_acc: 0.8584
Epoch 36/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5213 - acc: 0.8004 - val_loss: 0.
4900 - val_acc: 0.8584
Epoch 37/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5130 - acc: 0.8092 - val_loss: 0.
4820 - val_acc: 0.8493
Epoch 38/100
32/32 [==============================] - 0s 3ms/step - loss: 0.5050 - acc: 0.8014 - val_loss: 0.
4716 - val_acc: 0.8676
Epoch 39/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4972 - acc: 0.8131 - val_loss: 0.
4631 - val_acc: 0.8676
Epoch 40/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4891 - acc: 0.8229 - val_loss: 0.
4544 - val_acc: 0.8630
Epoch 41/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4811 - acc: 0.8141 - val_loss: 0.
4442 - val_acc: 0.8767
Epoch 42/100
```

```
32/32 [==============================] - 0s 3ms/step - loss: 0.4734 - acc: 0.8229 - val_loss: 0.
4362 - val_acc: 0.8767
Epoch 43/100
32/32 [==============================] - 0s 2ms/step - loss: 0.4660 - acc: 0.8288 - val_loss: 0.
4285 - val_acc: 0.8767
Epoch 44/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4584 - acc: 0.8317 - val_loss: 0.
4206 - val_acc: 0.8767
Epoch 45/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4512 - acc: 0.8327 - val_loss: 0.
4120 - val_acc: 0.8767
Epoch 46/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4437 - acc: 0.8356 - val_loss: 0.
4031 - val_acc: 0.8767
Epoch 47/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4371 - acc: 0.8376 - val_loss: 0.
3947 - val_acc: 0.8904
Epoch 48/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4301 - acc: 0.8493 - val_loss: 0.
3920 - val_acc: 0.8721
Epoch 49/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4241 - acc: 0.8376 - val_loss: 0.
3828 - val_acc: 0.8904
Epoch 50/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4180 - acc: 0.8434 - val_loss: 0.
3760 - val_acc: 0.8904
Epoch 51/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4123 - acc: 0.8464 - val_loss: 0.
3699 - val_acc: 0.8904
Epoch 52/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4064 - acc: 0.8493 - val_loss: 0.
3649 - val_acc: 0.8904
Epoch 53/100
32/32 [==============================] - 0s 3ms/step - loss: 0.4013 - acc: 0.8454 - val_loss: 0.
3589 - val_acc: 0.8950
Epoch 54/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3961 - acc: 0.8434 - val_loss: 0.
3524 - val_acc: 0.8950
Epoch 55/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3916 - acc: 0.8503 - val_loss: 0.
3502 - val_acc: 0.8904
Epoch 56/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3868 - acc: 0.8513 - val_loss: 0.
3467 - val_acc: 0.8858
Epoch 57/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3824 - acc: 0.8513 - val_loss: 0.
3408 - val_acc: 0.8904
Epoch 58/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3779 - acc: 0.8532 - val_loss: 0.
3384 - val_acc: 0.8813
Epoch 59/100
32/32 [==============================] - 0s 4ms/step - loss: 0.3744 - acc: 0.8542 - val_loss: 0.
3287 - val_acc: 0.8950
Epoch 60/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3707 - acc: 0.8571 - val_loss: 0.
3283 - val_acc: 0.8904
Epoch 61/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3672 - acc: 0.8581 - val_loss: 0.
3248 - val_acc: 0.8858
Epoch 62/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3641 - acc: 0.8542 - val_loss: 0.
3209 - val_acc: 0.8904
```

```
Epoch 63/100
32/32 [==============================] - 0s 2ms/step - loss: 0.3605 - acc: 0.8571 - val_loss: 0.
3157 - val_acc: 0.8995
Epoch 64/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3572 - acc: 0.8591 - val_loss: 0.
3130 - val_acc: 0.8950
Epoch 65/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3546 - acc: 0.8620 - val_loss: 0.
3101 - val_acc: 0.8904
Epoch 66/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3518 - acc: 0.8611 - val_loss: 0.
3089 - val_acc: 0.8858
Epoch 67/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3488 - acc: 0.8630 - val_loss: 0.
3055 - val_acc: 0.8858
Epoch 68/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3460 - acc: 0.8620 - val_loss: 0.
3064 - val_acc: 0.8858
Epoch 69/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3440 - acc: 0.8669 - val_loss: 0.
2996 - val_acc: 0.8858
Epoch 70/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3415 - acc: 0.8659 - val_loss: 0.
2982 - val_acc: 0.8858
Epoch 71/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3398 - acc: 0.8679 - val_loss: 0.
2958 - val_acc: 0.8858
Epoch 72/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3371 - acc: 0.8679 - val_loss: 0.
2942 - val_acc: 0.8858
Epoch 73/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3354 - acc: 0.8650 - val_loss: 0.
2928 - val_acc: 0.8858
Epoch 74/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3328 - acc: 0.8650 - val_loss: 0.
2873 - val_acc: 0.8904
Epoch 75/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3312 - acc: 0.8708 - val_loss: 0.
2889 - val_acc: 0.8904
Epoch 76/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3294 - acc: 0.8669 - val_loss: 0.
2847 - val_acc: 0.8858
Epoch 77/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3276 - acc: 0.8689 - val_loss: 0.
2847 - val_acc: 0.8858
Epoch 78/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3256 - acc: 0.8708 - val_loss: 0.
2853 - val_acc: 0.8858
Epoch 79/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3248 - acc: 0.8708 - val_loss: 0.
2822 - val_acc: 0.8904
Epoch 80/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3228 - acc: 0.8718 - val_loss: 0.
2811 - val_acc: 0.8904
Epoch 81/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3208 - acc: 0.8699 - val_loss: 0.
2778 - val_acc: 0.8904
Epoch 82/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3194 - acc: 0.8708 - val_loss: 0.
2755 - val_acc: 0.8858
Epoch 83/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3175 - acc: 0.8728 - val_loss: 0.
```

```
2729 - val_acc: 0.8904
Epoch 84/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3174 - acc: 0.8708 - val_loss: 0.
2747 - val_acc: 0.8904
Epoch 85/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3156 - acc: 0.8689 - val_loss: 0.
2729 - val_acc: 0.8904
Epoch 86/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3143 - acc: 0.8748 - val_loss: 0.
2728 - val_acc: 0.8904
Epoch 87/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3133 - acc: 0.8718 - val_loss: 0.
2735 - val_acc: 0.8904
Epoch 88/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3115 - acc: 0.8718 - val_loss: 0.
2747 - val_acc: 0.8950
Epoch 89/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3104 - acc: 0.8738 - val_loss: 0.
2689 - val_acc: 0.8950
Epoch 90/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3092 - acc: 0.8718 - val_loss: 0.
2675 - val_acc: 0.8950
Epoch 91/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3078 - acc: 0.8708 - val_loss: 0.
2711 - val_acc: 0.8950
Epoch 92/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3068 - acc: 0.8689 - val_loss: 0.
2669 - val_acc: 0.8950
Epoch 93/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3056 - acc: 0.8767 - val_loss: 0.
2622 - val_acc: 0.8904
Epoch 94/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3043 - acc: 0.8708 - val_loss: 0.
2652 - val_acc: 0.8950
Epoch 95/100
32/32 [==============================] - 0s 5ms/step - loss: 0.3023 - acc: 0.8748 - val_loss: 0.
2692 - val_acc: 0.8995
Epoch 96/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3030 - acc: 0.8748 - val_loss: 0.
2618 - val_acc: 0.8995
Epoch 97/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3015 - acc: 0.8757 - val_loss: 0.
2669 - val_acc: 0.8950
Epoch 98/100
32/32 [==============================] - 0s 3ms/step - loss: 0.3005 - acc: 0.8767 - val_loss: 0.
2598 - val_acc: 0.8995
Epoch 99/100
32/32 [==============================] - 0s 3ms/step - loss: 0.2998 - acc: 0.8708 - val_loss: 0.
2602 - val_acc: 0.8995
Epoch 100/100
32/32 [==============================] - 0s 3ms/step - loss: 0.2976 - acc: 0.8816 - val_loss: 0.
2566 - val_acc: 0.8904
```

Evaluating our data on the test set:

In [18]:
```
#Get the model accuracy
model.evaluate(X_test, Y_test)[1]
```

```
7/7 [==============================] - 0s 2ms/step - loss: 0.2890 - acc: 0.8767
```

Out[18]:
```
0.8767123222351074
```

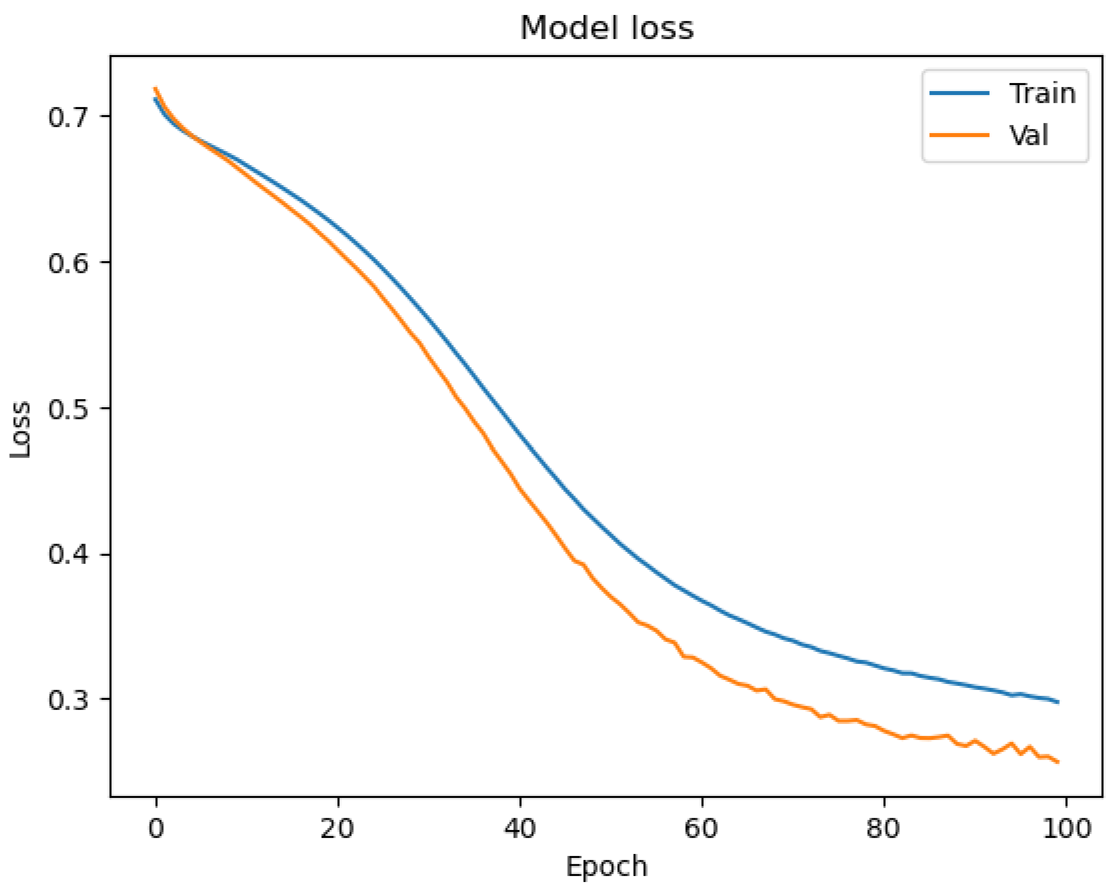
Visualizing Loss and Accuracy

Import the relevant package we need to do the visualization

In [19]:
```
import matplotlib.pyplot as plt
```

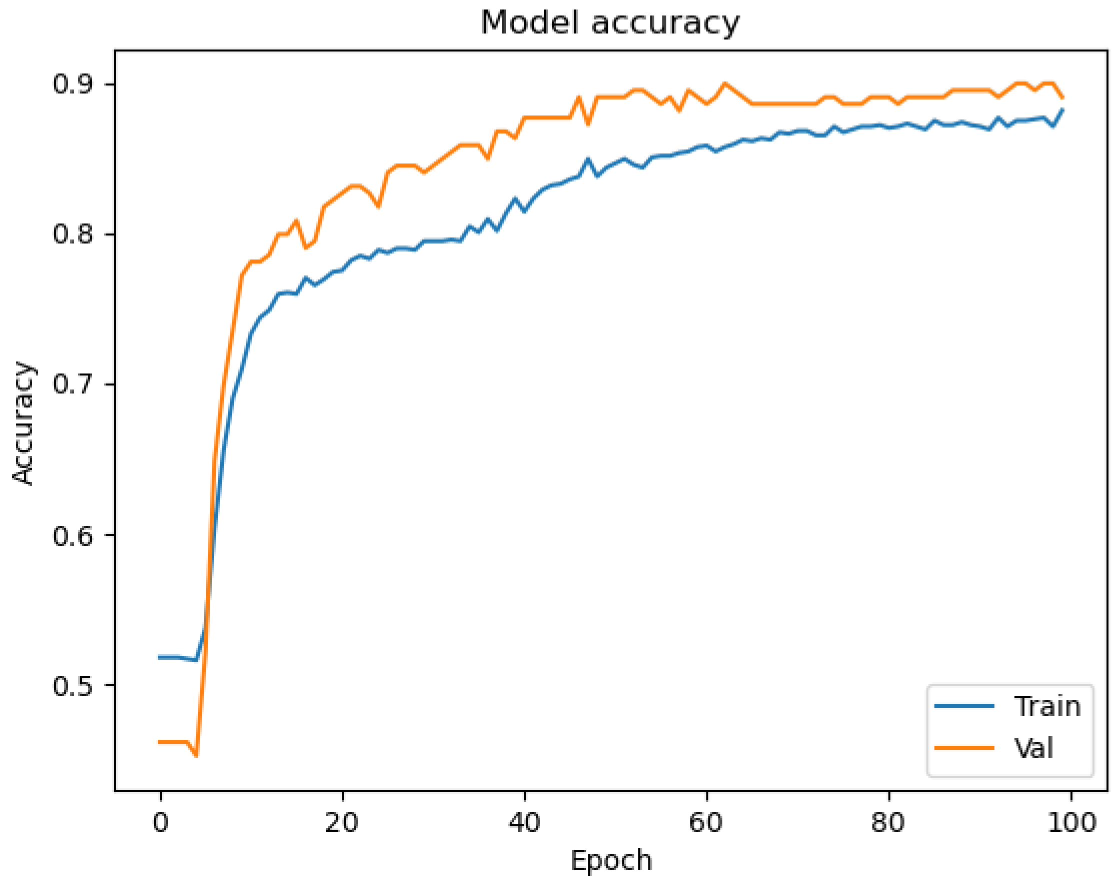
We want to visualize the training loss and the validation loss like this:

In [20]:
```
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()
```



We can also visualize the training accuracy and the validation accuracy like this:

In [21]:
```
plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



Adding Regularization to our Neural Network

We'll train a model which will overfit, which we call Model 2. This might take a few minutes.

There are 2 additional hidden layers in this model, all layers have 1000 neurons, and Adam is a common optimizer that reaches the lower loss function faster.

In [22]:

```
model_2 = Sequential([
    Dense(1000, activation='relu', input_shape=(10,)),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1, activation='sigmoid'),
])
model_2.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])
hist_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 26ms/step - loss: 0.5294 - acc: 0.7476 - val_loss:
0.3373 - val_acc: 0.9041
Epoch 2/100
32/32 [==============================] - 1s 20ms/step - loss: 0.3616 - acc: 0.8532 - val_loss:
0.3039 - val_acc: 0.9087
Epoch 3/100
32/32 [==============================] - 1s 20ms/step - loss: 0.3144 - acc: 0.8601 - val_loss:
0.3093 - val_acc: 0.8858
Epoch 4/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2811 - acc: 0.8816 - val_loss:
0.2830 - val_acc: 0.9041
Epoch 5/100
32/32 [==============================] - 1s 20ms/step - loss: 0.3284 - acc: 0.8415 - val_loss:
0.2947 - val_acc: 0.9087
Epoch 6/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2550 - acc: 0.8865 - val_loss:
0.3111 - val_acc: 0.9132
Epoch 7/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2432 - acc: 0.8865 - val_loss:
0.3269 - val_acc: 0.8995
Epoch 8/100
32/32 [==============================] - 1s 23ms/step - loss: 0.2297 - acc: 0.9022 - val_loss:
0.2939 - val_acc: 0.9132
Epoch 9/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2350 - acc: 0.8992 - val_loss:
0.3749 - val_acc: 0.8904
Epoch 10/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2535 - acc: 0.8914 - val_loss:
0.2871 - val_acc: 0.9041
Epoch 11/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2114 - acc: 0.9090 - val_loss:
0.4917 - val_acc: 0.8721
Epoch 12/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2333 - acc: 0.8992 - val_loss:
0.3536 - val_acc: 0.8995
Epoch 13/100
32/32 [==============================] - 1s 24ms/step - loss: 0.2314 - acc: 0.9022 - val_loss:
0.2979 - val_acc: 0.9132
Epoch 14/100
32/32 [==============================] - 1s 22ms/step - loss: 0.2190 - acc: 0.8973 - val_loss:
0.2991 - val_acc: 0.9087
Epoch 15/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2083 - acc: 0.8992 - val_loss:
0.3205 - val_acc: 0.9041
Epoch 16/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2155 - acc: 0.9080 - val_loss:
0.3263 - val_acc: 0.9224
Epoch 17/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2136 - acc: 0.9110 - val_loss:
0.3027 - val_acc: 0.8950
Epoch 18/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2384 - acc: 0.9100 - val_loss:
0.3553 - val_acc: 0.8995
Epoch 19/100
32/32 [==============================] - 1s 19ms/step - loss: 0.2167 - acc: 0.9080 - val_loss:
0.3415 - val_acc: 0.9041
Epoch 20/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2009 - acc: 0.9080 - val_loss:
0.3329 - val_acc: 0.9087
Epoch 21/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1990 - acc: 0.9159 - val_loss:
```

```
0.3145 - val_acc: 0.9087
Epoch 22/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2094 - acc: 0.9070 - val_loss:
0.3045 - val_acc: 0.9041
Epoch 23/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2149 - acc: 0.9041 - val_loss:
0.2879 - val_acc: 0.9041
Epoch 24/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2093 - acc: 0.9129 - val_loss:
0.3155 - val_acc: 0.9041
Epoch 25/100
32/32 [==============================] - 1s 19ms/step - loss: 0.2134 - acc: 0.9061 - val_loss:
0.3559 - val_acc: 0.8995
Epoch 26/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2538 - acc: 0.8865 - val_loss:
0.3146 - val_acc: 0.8995
Epoch 27/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2192 - acc: 0.9100 - val_loss:
0.3336 - val_acc: 0.9041
Epoch 28/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2006 - acc: 0.9119 - val_loss:
0.3198 - val_acc: 0.9087
Epoch 29/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1807 - acc: 0.9178 - val_loss:
0.3383 - val_acc: 0.9041
Epoch 30/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1907 - acc: 0.9207 - val_loss:
0.3349 - val_acc: 0.9041
Epoch 31/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1819 - acc: 0.9188 - val_loss:
0.3411 - val_acc: 0.9087
Epoch 32/100
32/32 [==============================] - 1s 25ms/step - loss: 0.1745 - acc: 0.9266 - val_loss:
0.3748 - val_acc: 0.9087
Epoch 33/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1822 - acc: 0.9198 - val_loss:
0.3658 - val_acc: 0.9041
Epoch 34/100
32/32 [==============================] - 1s 24ms/step - loss: 0.1796 - acc: 0.9256 - val_loss:
0.3255 - val_acc: 0.9132
Epoch 35/100
32/32 [==============================] - 1s 24ms/step - loss: 0.2068 - acc: 0.9159 - val_loss:
0.3620 - val_acc: 0.9087
Epoch 36/100
32/32 [==============================] - 1s 24ms/step - loss: 0.1926 - acc: 0.9227 - val_loss:
0.3941 - val_acc: 0.9087
Epoch 37/100
32/32 [==============================] - 1s 23ms/step - loss: 0.1767 - acc: 0.9237 - val_loss:
0.3963 - val_acc: 0.8904
Epoch 38/100
32/32 [==============================] - 1s 22ms/step - loss: 0.2040 - acc: 0.9100 - val_loss:
0.2992 - val_acc: 0.8904
Epoch 39/100
32/32 [==============================] - 1s 21ms/step - loss: 0.2122 - acc: 0.9090 - val_loss:
0.3165 - val_acc: 0.8950
Epoch 40/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1826 - acc: 0.9286 - val_loss:
0.4209 - val_acc: 0.8995
Epoch 41/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1729 - acc: 0.9256 - val_loss:
0.3817 - val_acc: 0.9224
Epoch 42/100
```

```
32/32 [==============================] - 1s 21ms/step - loss: 0.1744 - acc: 0.9305 - val_loss:
0.3944 - val_acc: 0.9178
Epoch 43/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1997 - acc: 0.9129 - val_loss:
0.4712 - val_acc: 0.8904
Epoch 44/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1752 - acc: 0.9266 - val_loss:
0.4340 - val_acc: 0.8995
Epoch 45/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1799 - acc: 0.9295 - val_loss:
0.3568 - val_acc: 0.9087
Epoch 46/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1757 - acc: 0.9178 - val_loss:
0.3704 - val_acc: 0.9041
Epoch 47/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2018 - acc: 0.9080 - val_loss:
0.3627 - val_acc: 0.9087
Epoch 48/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1656 - acc: 0.9354 - val_loss:
0.4203 - val_acc: 0.9087
Epoch 49/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1715 - acc: 0.9198 - val_loss:
0.3763 - val_acc: 0.9041
Epoch 50/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1525 - acc: 0.9315 - val_loss:
0.4688 - val_acc: 0.9087
Epoch 51/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1810 - acc: 0.9188 - val_loss:
0.4263 - val_acc: 0.8904
Epoch 52/100
32/32 [==============================] - 1s 20ms/step - loss: 0.2162 - acc: 0.9041 - val_loss:
0.3500 - val_acc: 0.8950
Epoch 53/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1852 - acc: 0.9237 - val_loss:
0.3489 - val_acc: 0.8995
Epoch 54/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1643 - acc: 0.9325 - val_loss:
0.4327 - val_acc: 0.9041
Epoch 55/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1675 - acc: 0.9256 - val_loss:
0.4887 - val_acc: 0.8995
Epoch 56/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1590 - acc: 0.9384 - val_loss:
0.4765 - val_acc: 0.9087
Epoch 57/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1687 - acc: 0.9266 - val_loss:
0.4386 - val_acc: 0.9041
Epoch 58/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1707 - acc: 0.9237 - val_loss:
0.5437 - val_acc: 0.8858
Epoch 59/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1682 - acc: 0.9247 - val_loss:
0.4130 - val_acc: 0.9178
Epoch 60/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1592 - acc: 0.9374 - val_loss:
0.4360 - val_acc: 0.8858
Epoch 61/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1598 - acc: 0.9335 - val_loss:
0.4758 - val_acc: 0.9087
Epoch 62/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1536 - acc: 0.9325 - val_loss:
0.5729 - val_acc: 0.9132
```

```
Epoch 63/100
32/32 [==============================] - 1s 19ms/step - loss: 0.1513 - acc: 0.9335 - val_loss:
0.4354 - val_acc: 0.8904
Epoch 64/100
32/32 [==============================] - 1s 19ms/step - loss: 0.1492 - acc: 0.9374 - val_loss:
0.5650 - val_acc: 0.8995
Epoch 65/100
32/32 [==============================] - 1s 19ms/step - loss: 0.1549 - acc: 0.9325 - val_loss:
0.4152 - val_acc: 0.9132
Epoch 66/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1694 - acc: 0.9217 - val_loss:
0.3672 - val_acc: 0.8995
Epoch 67/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1561 - acc: 0.9344 - val_loss:
0.5215 - val_acc: 0.9132
Epoch 68/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1578 - acc: 0.9295 - val_loss:
0.5394 - val_acc: 0.8950
Epoch 69/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1519 - acc: 0.9413 - val_loss:
0.3956 - val_acc: 0.9087
Epoch 70/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1415 - acc: 0.9423 - val_loss:
0.5476 - val_acc: 0.8995
Epoch 71/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1534 - acc: 0.9335 - val_loss:
0.4615 - val_acc: 0.8904
Epoch 72/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1397 - acc: 0.9393 - val_loss:
0.5581 - val_acc: 0.9132
Epoch 73/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1693 - acc: 0.9344 - val_loss:
0.4443 - val_acc: 0.8995
Epoch 74/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1677 - acc: 0.9217 - val_loss:
0.3535 - val_acc: 0.9087
Epoch 75/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1525 - acc: 0.9374 - val_loss:
0.4147 - val_acc: 0.9041
Epoch 76/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1301 - acc: 0.9403 - val_loss:
0.4984 - val_acc: 0.9132
Epoch 77/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1304 - acc: 0.9472 - val_loss:
0.4668 - val_acc: 0.9178
Epoch 78/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1548 - acc: 0.9354 - val_loss:
0.5246 - val_acc: 0.9087
Epoch 79/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1431 - acc: 0.9413 - val_loss:
0.4890 - val_acc: 0.9041
Epoch 80/100
32/32 [==============================] - 1s 23ms/step - loss: 0.1361 - acc: 0.9462 - val_loss:
0.5592 - val_acc: 0.9087
Epoch 81/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1288 - acc: 0.9403 - val_loss:
0.6431 - val_acc: 0.8995
Epoch 82/100
32/32 [==============================] - 1s 23ms/step - loss: 0.1429 - acc: 0.9374 - val_loss:
0.6051 - val_acc: 0.9087
Epoch 83/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1137 - acc: 0.9521 - val_loss:
```

```
0.5751 - val_acc: 0.8995
Epoch 84/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1193 - acc: 0.9540 - val_loss:
0.6031 - val_acc: 0.9132
Epoch 85/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1378 - acc: 0.9403 - val_loss:
0.5880 - val_acc: 0.9087
Epoch 86/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1141 - acc: 0.9521 - val_loss:
0.5663 - val_acc: 0.8995
Epoch 87/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1278 - acc: 0.9521 - val_loss:
0.7668 - val_acc: 0.8950
Epoch 88/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1901 - acc: 0.9237 - val_loss:
0.4090 - val_acc: 0.9041
Epoch 89/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1449 - acc: 0.9472 - val_loss:
0.5495 - val_acc: 0.8995
Epoch 90/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1178 - acc: 0.9599 - val_loss:
0.5126 - val_acc: 0.9041
Epoch 91/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1144 - acc: 0.9550 - val_loss:
0.4947 - val_acc: 0.9132
Epoch 92/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1066 - acc: 0.9589 - val_loss:
0.7052 - val_acc: 0.8950
Epoch 93/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1136 - acc: 0.9569 - val_loss:
0.5853 - val_acc: 0.9269
Epoch 94/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1130 - acc: 0.9569 - val_loss:
0.5815 - val_acc: 0.8995
Epoch 95/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1332 - acc: 0.9501 - val_loss:
0.4411 - val_acc: 0.9041
Epoch 96/100
32/32 [==============================] - 1s 21ms/step - loss: 0.1340 - acc: 0.9452 - val_loss:
0.4967 - val_acc: 0.9087
Epoch 97/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1480 - acc: 0.9393 - val_loss:
0.5045 - val_acc: 0.9087
Epoch 98/100
32/32 [==============================] - 1s 22ms/step - loss: 0.1082 - acc: 0.9569 - val_loss:
0.5534 - val_acc: 0.9132
Epoch 99/100
32/32 [==============================] - 1s 20ms/step - loss: 0.1006 - acc: 0.9667 - val_loss:
0.4671 - val_acc: 0.9041
Epoch 100/100
32/32 [==============================] - 1s 21ms/step - loss: 0.0911 - acc: 0.9648 - val_loss:
0.6131 - val_acc: 0.9087
```

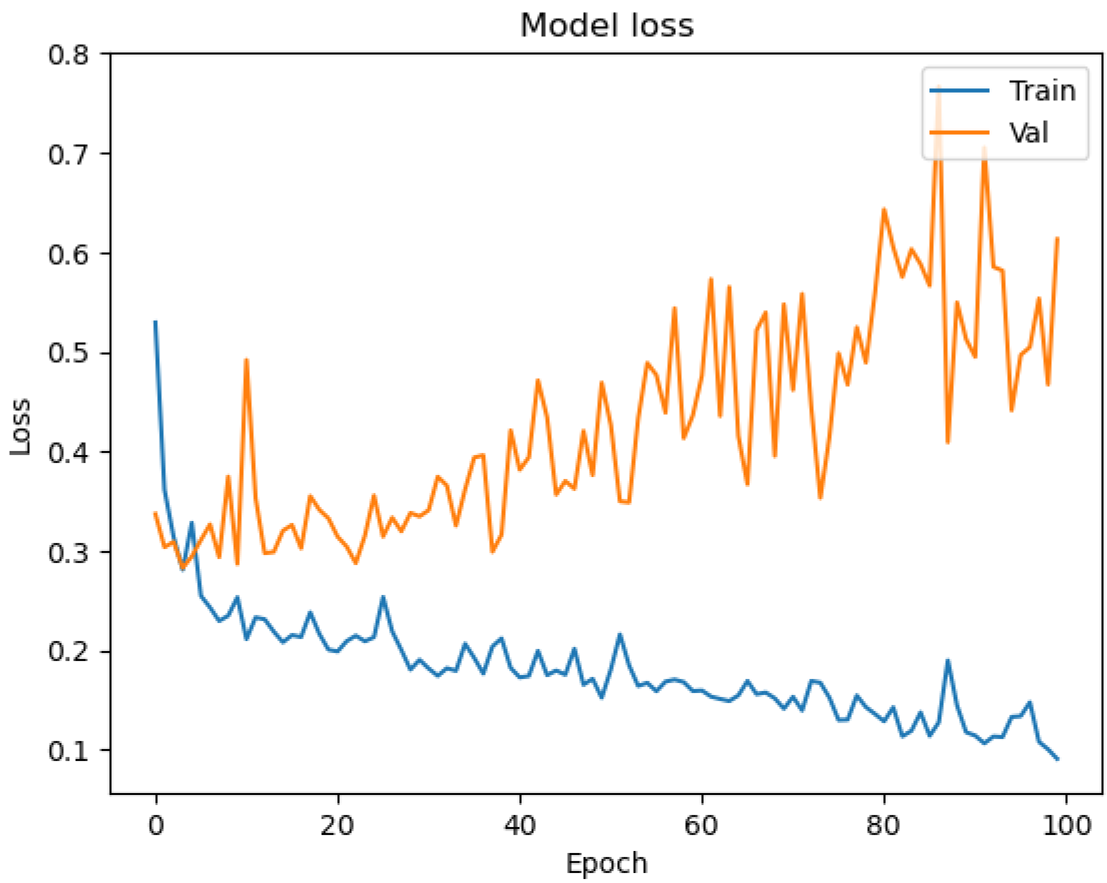
Let's do the same visualization to see what overfitting looks like in terms of the loss and accuracy.

The loss for the Validation data continues to increase while the loss for the Training data reduces, which is a sign of overfitting.

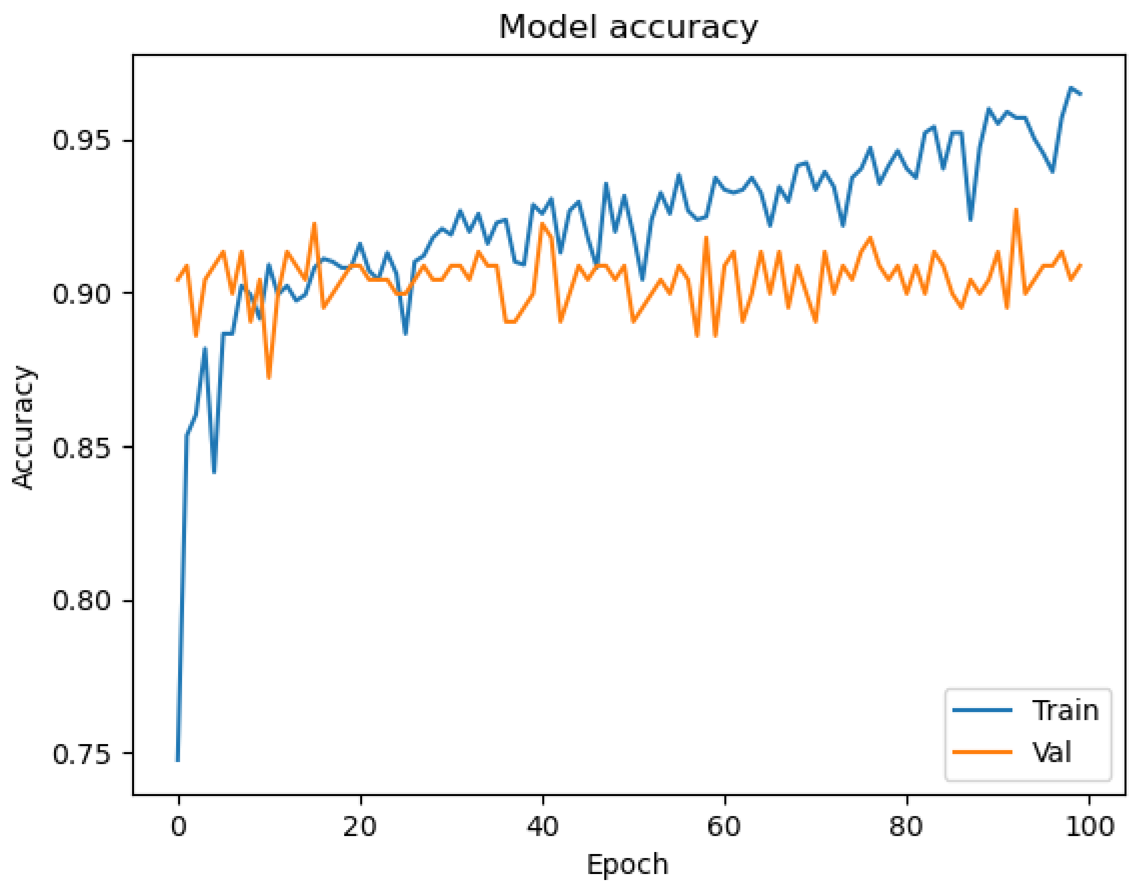
In [23]:

```
plt.plot(hist_2.history['loss'])
plt.plot(hist_2.history['val_loss'])
plt.title('Model loss')
```

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



In [24]:
```
plt.plot(hist_2.history['acc'])
plt.plot(hist_2.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



To address the overfitting we see in Model 2, we'll incorporate L2 regularization and dropout in our third model here (Model 3).

In [25]:
```
from keras.layers import Dropout
from keras import regularizers
```

In [26]:
```
#Change the model code to include Dropout and regularization to
#include the the squared values of the parameters into the loss function
#Dropout of 0.3 means a %30 probability of dropping out during training
model_3 = Sequential([
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01), input_shape=(10,)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1000, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(1, activation='sigmoid', kernel_regularizer=regularizers.l2(0.01)),
])
```

In [27]:
```
model_3.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])
hist_3 = model_3.fit(X_train, Y_train,
          batch_size=32, epochs=100,
          validation_data=(X_val, Y_val))
```

```
Epoch 1/100
32/32 [==============================] - 2s 43ms/step - loss: 14.2309 - acc: 0.6311 - val_loss:
3.8815 - val_acc: 0.8265
Epoch 2/100
32/32 [==============================] - 1s 35ms/step - loss: 1.6598 - acc: 0.8395 - val_loss:
0.5977 - val_acc: 0.9087
Epoch 3/100
32/32 [==============================] - 1s 35ms/step - loss: 0.5772 - acc: 0.8483 - val_loss:
0.4699 - val_acc: 0.9041
Epoch 4/100
32/32 [==============================] - 1s 35ms/step - loss: 0.5342 - acc: 0.8513 - val_loss:
0.4638 - val_acc: 0.8995
Epoch 5/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4841 - acc: 0.8757 - val_loss:
0.4374 - val_acc: 0.8950
Epoch 6/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4845 - acc: 0.8708 - val_loss:
0.4274 - val_acc: 0.9087
Epoch 7/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4741 - acc: 0.8777 - val_loss:
0.4397 - val_acc: 0.8904
Epoch 8/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4857 - acc: 0.8620 - val_loss:
0.4230 - val_acc: 0.8995
Epoch 9/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4877 - acc: 0.8611 - val_loss:
0.4284 - val_acc: 0.8950
Epoch 10/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4836 - acc: 0.8679 - val_loss:
0.4210 - val_acc: 0.9041
Epoch 11/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4576 - acc: 0.8777 - val_loss:
0.4129 - val_acc: 0.9087
Epoch 12/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4896 - acc: 0.8581 - val_loss:
0.4341 - val_acc: 0.8904
Epoch 13/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4683 - acc: 0.8630 - val_loss:
0.4096 - val_acc: 0.9041
Epoch 14/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4566 - acc: 0.8708 - val_loss:
0.4140 - val_acc: 0.8950
Epoch 15/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4566 - acc: 0.8708 - val_loss:
0.4188 - val_acc: 0.9041
Epoch 16/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4476 - acc: 0.8767 - val_loss:
0.4058 - val_acc: 0.9087
Epoch 17/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4398 - acc: 0.8757 - val_loss:
0.4249 - val_acc: 0.8904
Epoch 18/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4564 - acc: 0.8796 - val_loss:
0.4115 - val_acc: 0.8858
Epoch 19/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4543 - acc: 0.8669 - val_loss:
0.4569 - val_acc: 0.8858
Epoch 20/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4495 - acc: 0.8806 - val_loss:
0.4610 - val_acc: 0.8813
Epoch 21/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4690 - acc: 0.8601 - val_loss:
```

```
0.4646 - val_acc: 0.8813
Epoch 22/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4707 - acc: 0.8630 - val_loss:
0.4409 - val_acc: 0.9087
Epoch 23/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4510 - acc: 0.8708 - val_loss:
0.4104 - val_acc: 0.9087
Epoch 24/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4537 - acc: 0.8767 - val_loss:
0.4560 - val_acc: 0.8813
Epoch 25/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4375 - acc: 0.8845 - val_loss:
0.4281 - val_acc: 0.9087
Epoch 26/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4829 - acc: 0.8620 - val_loss:
0.4389 - val_acc: 0.8858
Epoch 27/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4499 - acc: 0.8796 - val_loss:
0.4016 - val_acc: 0.9041
Epoch 28/100
32/32 [==============================] - 1s 31ms/step - loss: 0.4399 - acc: 0.8718 - val_loss:
0.4014 - val_acc: 0.9087
Epoch 29/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4529 - acc: 0.8767 - val_loss:
0.4197 - val_acc: 0.8995
Epoch 30/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4439 - acc: 0.8679 - val_loss:
0.4011 - val_acc: 0.9087
Epoch 31/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4423 - acc: 0.8816 - val_loss:
0.4140 - val_acc: 0.8858
Epoch 32/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4429 - acc: 0.8787 - val_loss:
0.4132 - val_acc: 0.8995
Epoch 33/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4447 - acc: 0.8708 - val_loss:
0.3995 - val_acc: 0.9087
Epoch 34/100
32/32 [==============================] - 1s 40ms/step - loss: 0.4389 - acc: 0.8748 - val_loss:
0.4197 - val_acc: 0.8858
Epoch 35/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4375 - acc: 0.8816 - val_loss:
0.4396 - val_acc: 0.8904
Epoch 36/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4434 - acc: 0.8708 - val_loss:
0.4157 - val_acc: 0.8904
Epoch 37/100
32/32 [==============================] - 1s 40ms/step - loss: 0.4669 - acc: 0.8679 - val_loss:
0.4062 - val_acc: 0.8950
Epoch 38/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4455 - acc: 0.8699 - val_loss:
0.4051 - val_acc: 0.9132
Epoch 39/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4375 - acc: 0.8767 - val_loss:
0.4302 - val_acc: 0.8995
Epoch 40/100
32/32 [==============================] - 1s 39ms/step - loss: 0.4491 - acc: 0.8826 - val_loss:
0.4018 - val_acc: 0.8995
Epoch 41/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4494 - acc: 0.8767 - val_loss:
0.4135 - val_acc: 0.8995
Epoch 42/100
```

```
32/32 [==============================] - 1s 38ms/step - loss: 0.4337 - acc: 0.8787 - val_loss:
0.4457 - val_acc: 0.8813
Epoch 43/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4519 - acc: 0.8640 - val_loss:
0.4093 - val_acc: 0.9087
Epoch 44/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4700 - acc: 0.8699 - val_loss:
0.4321 - val_acc: 0.8858
Epoch 45/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4425 - acc: 0.8708 - val_loss:
0.4108 - val_acc: 0.8904
Epoch 46/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4415 - acc: 0.8806 - val_loss:
0.4005 - val_acc: 0.9041
Epoch 47/100
32/32 [==============================] - 1s 40ms/step - loss: 0.4441 - acc: 0.8787 - val_loss:
0.4071 - val_acc: 0.9132
Epoch 48/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4451 - acc: 0.8826 - val_loss:
0.4176 - val_acc: 0.8858
Epoch 49/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4470 - acc: 0.8767 - val_loss:
0.4132 - val_acc: 0.8858
Epoch 50/100
32/32 [==============================] - 1s 41ms/step - loss: 0.4515 - acc: 0.8718 - val_loss:
0.4007 - val_acc: 0.9132
Epoch 51/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4356 - acc: 0.8777 - val_loss:
0.4028 - val_acc: 0.9041
Epoch 52/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4369 - acc: 0.8826 - val_loss:
0.4029 - val_acc: 0.8858
Epoch 53/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4539 - acc: 0.8728 - val_loss:
0.4120 - val_acc: 0.8858
Epoch 54/100
32/32 [==============================] - 1s 38ms/step - loss: 0.4421 - acc: 0.8718 - val_loss:
0.4021 - val_acc: 0.9041
Epoch 55/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4406 - acc: 0.8796 - val_loss:
0.4024 - val_acc: 0.8950
Epoch 56/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4729 - acc: 0.8552 - val_loss:
0.4209 - val_acc: 0.8995
Epoch 57/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4495 - acc: 0.8738 - val_loss:
0.4057 - val_acc: 0.8950
Epoch 58/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4364 - acc: 0.8738 - val_loss:
0.4103 - val_acc: 0.9041
Epoch 59/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4474 - acc: 0.8806 - val_loss:
0.4201 - val_acc: 0.8995
Epoch 60/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4390 - acc: 0.8796 - val_loss:
0.4006 - val_acc: 0.9041
Epoch 61/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4417 - acc: 0.8679 - val_loss:
0.4103 - val_acc: 0.8858
Epoch 62/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4369 - acc: 0.8826 - val_loss:
0.4490 - val_acc: 0.8950
```

```
Epoch 63/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4370 - acc: 0.8796 - val_loss:
0.4044 - val_acc: 0.9132
Epoch 64/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4373 - acc: 0.8767 - val_loss:
0.4322 - val_acc: 0.9087
Epoch 65/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4548 - acc: 0.8748 - val_loss:
0.4583 - val_acc: 0.8813
Epoch 66/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4433 - acc: 0.8806 - val_loss:
0.4005 - val_acc: 0.9041
Epoch 67/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4320 - acc: 0.8875 - val_loss:
0.4629 - val_acc: 0.8767
Epoch 68/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4681 - acc: 0.8640 - val_loss:
0.4255 - val_acc: 0.8950
Epoch 69/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4410 - acc: 0.8855 - val_loss:
0.4092 - val_acc: 0.8904
Epoch 70/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4491 - acc: 0.8699 - val_loss:
0.4062 - val_acc: 0.8858
Epoch 71/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4365 - acc: 0.8836 - val_loss:
0.4255 - val_acc: 0.8950
Epoch 72/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4479 - acc: 0.8757 - val_loss:
0.4006 - val_acc: 0.9041
Epoch 73/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4367 - acc: 0.8855 - val_loss:
0.4015 - val_acc: 0.9041
Epoch 74/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4501 - acc: 0.8748 - val_loss:
0.4137 - val_acc: 0.8904
Epoch 75/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4315 - acc: 0.8806 - val_loss:
0.4017 - val_acc: 0.8904
Epoch 76/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4370 - acc: 0.8777 - val_loss:
0.4138 - val_acc: 0.8858
Epoch 77/100
32/32 [==============================] - 1s 39ms/step - loss: 0.4377 - acc: 0.8689 - val_loss:
0.4402 - val_acc: 0.8904
Epoch 78/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4523 - acc: 0.8650 - val_loss:
0.4253 - val_acc: 0.8904
Epoch 79/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4266 - acc: 0.8865 - val_loss:
0.4118 - val_acc: 0.8904
Epoch 80/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4554 - acc: 0.8601 - val_loss:
0.4442 - val_acc: 0.8813
Epoch 81/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4386 - acc: 0.8826 - val_loss:
0.4245 - val_acc: 0.9041
Epoch 82/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4420 - acc: 0.8777 - val_loss:
0.4241 - val_acc: 0.8904
Epoch 83/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4267 - acc: 0.8845 - val_loss:
```

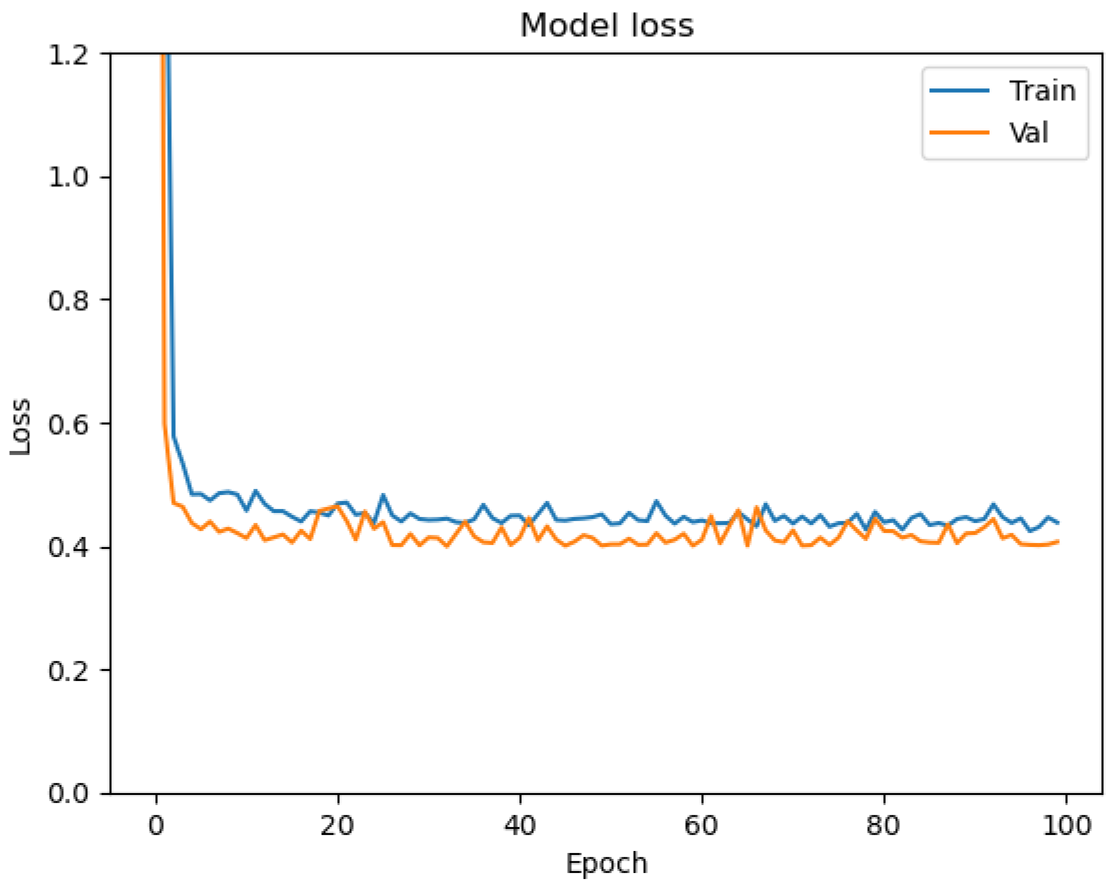
```
0.4136 - val_acc: 0.8813
Epoch 84/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4456 - acc: 0.8796 - val_loss:
0.4179 - val_acc: 0.8995
Epoch 85/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4520 - acc: 0.8767 - val_loss:
0.4077 - val_acc: 0.8904
Epoch 86/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4342 - acc: 0.8914 - val_loss:
0.4059 - val_acc: 0.8904
Epoch 87/100
32/32 [==============================] - 1s 37ms/step - loss: 0.4374 - acc: 0.8826 - val_loss:
0.4054 - val_acc: 0.9132
Epoch 88/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4331 - acc: 0.8699 - val_loss:
0.4354 - val_acc: 0.8904
Epoch 89/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4439 - acc: 0.8708 - val_loss:
0.4048 - val_acc: 0.8904
Epoch 90/100
32/32 [==============================] - 1s 40ms/step - loss: 0.4469 - acc: 0.8620 - val_loss:
0.4204 - val_acc: 0.8995
Epoch 91/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4405 - acc: 0.8757 - val_loss:
0.4212 - val_acc: 0.8858
Epoch 92/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4440 - acc: 0.8787 - val_loss:
0.4305 - val_acc: 0.9087
Epoch 93/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4679 - acc: 0.8640 - val_loss:
0.4439 - val_acc: 0.8858
Epoch 94/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4471 - acc: 0.8738 - val_loss:
0.4126 - val_acc: 0.8858
Epoch 95/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4377 - acc: 0.8826 - val_loss:
0.4181 - val_acc: 0.8858
Epoch 96/100
32/32 [==============================] - 1s 33ms/step - loss: 0.4450 - acc: 0.8777 - val_loss:
0.4032 - val_acc: 0.9178
Epoch 97/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4243 - acc: 0.8855 - val_loss:
0.4019 - val_acc: 0.9132
Epoch 98/100
32/32 [==============================] - 1s 34ms/step - loss: 0.4309 - acc: 0.8885 - val_loss:
0.4014 - val_acc: 0.9132
Epoch 99/100
32/32 [==============================] - 1s 36ms/step - loss: 0.4465 - acc: 0.8855 - val_loss:
0.4026 - val_acc: 0.9041
Epoch 100/100
32/32 [==============================] - 1s 35ms/step - loss: 0.4382 - acc: 0.8855 - val_loss:
0.4068 - val_acc: 0.8904
```

We'll now plot the loss and accuracy graphs for Model 3. You'll notice that the loss is a lot higher at the start, and that's because we've changed our loss function. To plot such that the window is zoomed in between 0 and 1.2 for the loss, we add an additional line of code (plt.ylim) when plotting

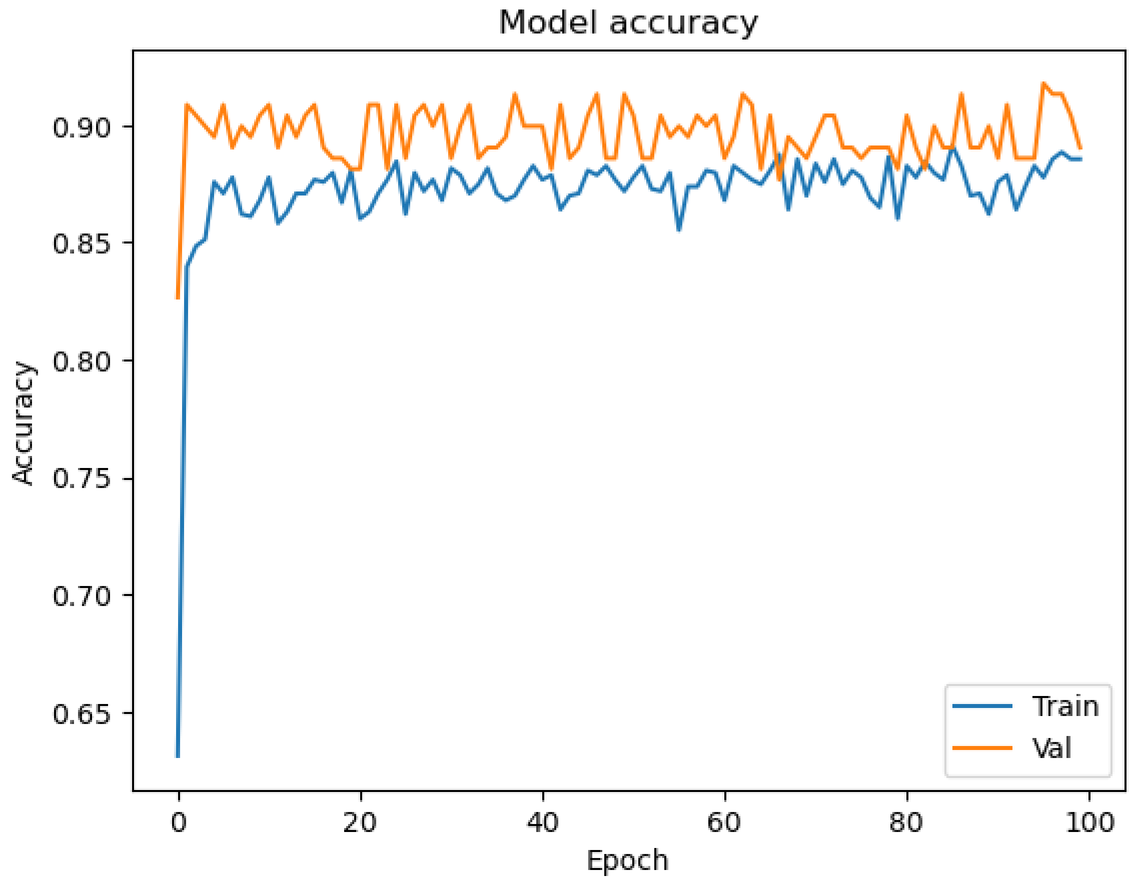
In [28]:

```
plt.plot(hist_3.history['loss'])
plt.plot(hist_3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.ylim(top=1.2, bottom=0)
plt.show()
```



In [29]:
```
plt.plot(hist_3.history['acc'])
plt.plot(hist_3.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



As compared to Model 2, you should see that there's less overfitting!