```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv1D,
MaxPool1D,Flatten,Dense,Dropout,BatchNormalization
from tensorflow.keras.optimizers import Adam

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn import datasets,metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

cancerData = datasets.load_breast_cancer()

X = pd.DataFrame(data = cancerData.data,
columns=cancerData.feature_names )
X.head()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030

   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst radius  worst texture  worst
perimeter  \
0                 0.07871  ...         25.38          17.33
```

```
184.60
1                  0.05667  ...         24.99          23.41
158.80
2                  0.05999  ...         23.57          25.53
152.50
3                  0.09744  ...         14.91          26.50
98.87
4                  0.05883  ...         22.54          16.67
152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

```
y = cancerData.target
```

```
X.shape
```

```
(569, 30)
```

```
X_train,X_test,y_train,y_test=
train_test_split(X,y,test_size=0.2,stratify=y)
```

```
X_train.shape
```

```
(455, 30)
```

```
y_test.shape
```

```
(114,)
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train = X_train.reshape(455,30,1)
X_test = X_test.reshape(114,30,1)
```

```
model = Sequential()
model.add(Conv1D(filters=32,kernel_size=2,activation='relu',input_shap
e=(30,1)))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv1D(64,2,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))
```

```
/opt/anaconda3/lib/python3.12/site-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
model.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv1d (Conv1D)                 │ (None, 29, 32)         │       
96 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 29, 32)         │       
128 │
│ (BatchNormalization)            │                        │               
│
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 29, 32)         │       
0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv1d_1 (Conv1D)               │ (None, 28, 64)         │       
4,160 │
```

```
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ batch_normalization_1            ┃ (None, 28, 64)          ┃
256 ┃
┃ (BatchNormalization)             ┃                         ┃
    ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ dropout_1 (Dropout)              ┃ (None, 28, 64)          ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ flatten (Flatten)                ┃ (None, 1792)            ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ dense (Dense)                    ┃ (None, 64)              ┃
114,752 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ dropout_2 (Dropout)              ┃ (None, 64)              ┃
0 ┃
┣━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┫━━━━━━━━━━━━━
━━━━┫
┃ dense_1 (Dense)                  ┃ (None, 1)               ┃
65 ┃
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┻━━━━━━━━━━━━━
━━━━┛

 Total params: 119,457 (466.63 KB)

 Trainable params: 119,265 (465.88 KB)

 Non-trainable params: 192 (768.00 B)
```

```
model.compile(optimizer=Adam(learning_rate=0.0005),loss='binary_crosse
ntropy',metrics=['accuracy'])

history =
model.fit(X_train,y_train,epochs=50,verbose=1,validation_data=(X_test,
y_test))
```

```
Epoch 1/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6288 - loss:
0.8921 - val_accuracy: 0.9298 - val_loss: 0.5021
Epoch 2/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9149 - loss:
0.1983 - val_accuracy: 0.9123 - val_loss: 0.4375
Epoch 3/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9492 - loss:
0.1252 - val_accuracy: 0.8772 - val_loss: 0.4025
```

```
Epoch 4/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9377 - loss:
0.1751 - val_accuracy: 0.8509 - val_loss: 0.3693
Epoch 5/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9519 - loss:
0.1343 - val_accuracy: 0.8158 - val_loss: 0.3682
Epoch 6/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9453 - loss:
0.1261 - val_accuracy: 0.8333 - val_loss: 0.3411
Epoch 7/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.9672 - loss:
0.1145 - val_accuracy: 0.8772 - val_loss: 0.3197
Epoch 8/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9676 - loss:
0.0966 - val_accuracy: 0.8772 - val_loss: 0.2989
Epoch 9/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9798 - loss:
0.0860 - val_accuracy: 0.8509 - val_loss: 0.2911
Epoch 10/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9715 - loss:
0.0742 - val_accuracy: 0.8684 - val_loss: 0.2809
Epoch 11/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9690 - loss:
0.0883 - val_accuracy: 0.8684 - val_loss: 0.2708
Epoch 12/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9769 - loss:
0.0533 - val_accuracy: 0.8860 - val_loss: 0.2556
Epoch 13/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9806 - loss:
0.0538 - val_accuracy: 0.8860 - val_loss: 0.2518
Epoch 14/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9723 - loss:
0.0588 - val_accuracy: 0.8947 - val_loss: 0.2383
Epoch 15/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9812 - loss:
0.0480 - val_accuracy: 0.9035 - val_loss: 0.2174
Epoch 16/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9761 - loss:
0.0620 - val_accuracy: 0.9386 - val_loss: 0.1649
Epoch 17/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9795 - loss:
0.0678 - val_accuracy: 0.9474 - val_loss: 0.1431
Epoch 18/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9728 - loss:
0.0552 - val_accuracy: 0.9474 - val_loss: 0.1464
Epoch 19/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9843 - loss:
0.0609 - val_accuracy: 0.9474 - val_loss: 0.1444
Epoch 20/50
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9777 - loss:
0.0612 - val_accuracy: 0.9561 - val_loss: 0.1213
Epoch 21/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9877 - loss:
0.0519 - val_accuracy: 0.9561 - val_loss: 0.1270
Epoch 22/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9777 - loss:
0.0482 - val_accuracy: 0.9298 - val_loss: 0.1844
Epoch 23/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9808 - loss:
0.0467 - val_accuracy: 0.9561 - val_loss: 0.1368
Epoch 24/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9621 - loss:
0.0786 - val_accuracy: 0.9649 - val_loss: 0.1196
Epoch 25/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9773 - loss:
0.0746 - val_accuracy: 0.9649 - val_loss: 0.1102
Epoch 26/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9815 - loss:
0.0367 - val_accuracy: 0.9649 - val_loss: 0.1041
Epoch 27/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9809 - loss:
0.0507 - val_accuracy: 0.9649 - val_loss: 0.1077
Epoch 28/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9820 - loss:
0.0512 - val_accuracy: 0.9649 - val_loss: 0.1142
Epoch 29/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9889 - loss:
0.0453 - val_accuracy: 0.9649 - val_loss: 0.1047
Epoch 30/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9788 - loss:
0.0732 - val_accuracy: 0.9561 - val_loss: 0.1068
Epoch 31/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9700 - loss:
0.0513 - val_accuracy: 0.9649 - val_loss: 0.1048
Epoch 32/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9873 - loss:
0.0349 - val_accuracy: 0.9561 - val_loss: 0.1095
Epoch 33/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9830 - loss:
0.0441 - val_accuracy: 0.9561 - val_loss: 0.1183
Epoch 34/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9867 - loss:
0.0351 - val_accuracy: 0.9474 - val_loss: 0.1190
Epoch 35/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9845 - loss:
0.0308 - val_accuracy: 0.9561 - val_loss: 0.1143
Epoch 36/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9629 - loss:
```

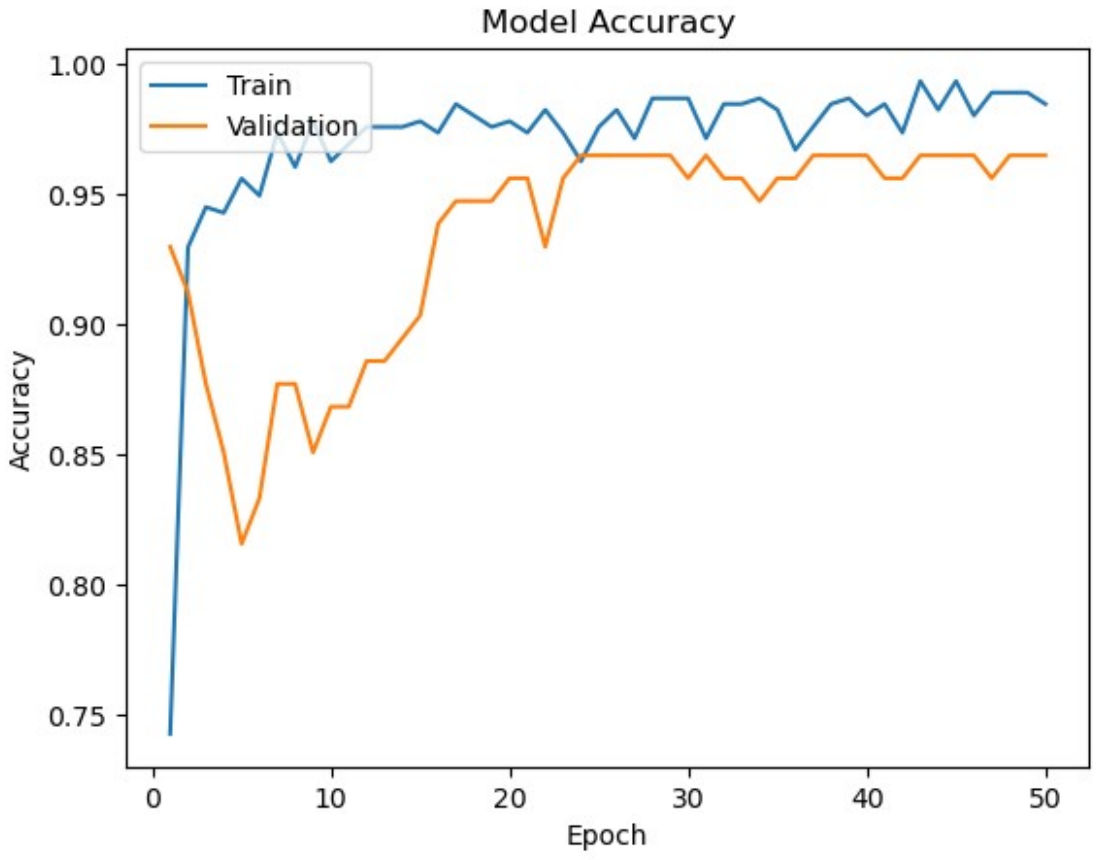
```
0.0847 - val_accuracy: 0.9561 - val_loss: 0.1117
Epoch 37/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9753 - loss:
0.0723 - val_accuracy: 0.9649 - val_loss: 0.1130
Epoch 38/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9797 - loss:
0.0853 - val_accuracy: 0.9649 - val_loss: 0.1149
Epoch 39/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9826 - loss:
0.0342 - val_accuracy: 0.9649 - val_loss: 0.1097
Epoch 40/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9869 - loss:
0.0309 - val_accuracy: 0.9649 - val_loss: 0.1118
Epoch 41/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9860 - loss:
0.0415 - val_accuracy: 0.9561 - val_loss: 0.1141
Epoch 42/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9664 - loss:
0.0718 - val_accuracy: 0.9561 - val_loss: 0.1185
Epoch 43/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9922 - loss:
0.0227 - val_accuracy: 0.9649 - val_loss: 0.1179
Epoch 44/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9830 - loss:
0.0371 - val_accuracy: 0.9649 - val_loss: 0.1271
Epoch 45/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9922 - loss:
0.0223 - val_accuracy: 0.9649 - val_loss: 0.1343
Epoch 46/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9866 - loss:
0.0319 - val_accuracy: 0.9649 - val_loss: 0.1321
Epoch 47/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9879 - loss:
0.0363 - val_accuracy: 0.9561 - val_loss: 0.1345
Epoch 48/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9878 - loss:
0.0258 - val_accuracy: 0.9649 - val_loss: 0.1256
Epoch 49/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9885 - loss:
0.0285 - val_accuracy: 0.9649 - val_loss: 0.1167
Epoch 50/50
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9803 - loss:
0.0473 - val_accuracy: 0.9649 - val_loss: 0.1007
```

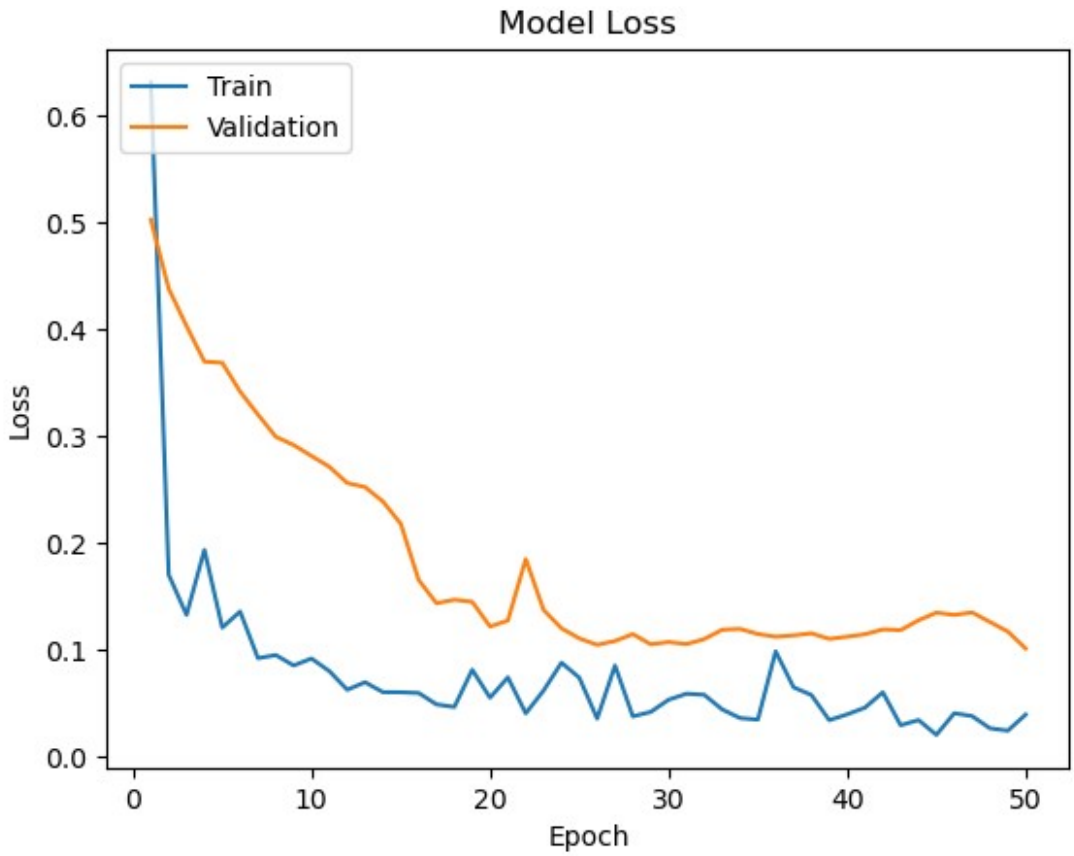
```python
def plotLearningCurve(history,epochs):
  epochRange = range(1,epochs+1)
  plt.plot(epochRange,history.history['accuracy'])
  plt.plot(epochRange,history.history['val_accuracy'])
  plt.title('Model Accuracy')
  plt.xlabel('Epoch')
```

```
  plt.ylabel('Accuracy')
  plt.legend(['Train','Validation'],loc='upper left')
  plt.show()

  plt.plot(epochRange,history.history['loss'])
  plt.plot(epochRange,history.history['val_loss'])
  plt.title('Model Loss')
  plt.xlabel('Epoch')
  plt.ylabel('Loss')
  plt.legend(['Train','Validation'],loc='upper left')
  plt.show()


plotLearningCurve(history,50)
```




Model Loss
Train
Validation
Loss
Epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
10
20
30
40
50