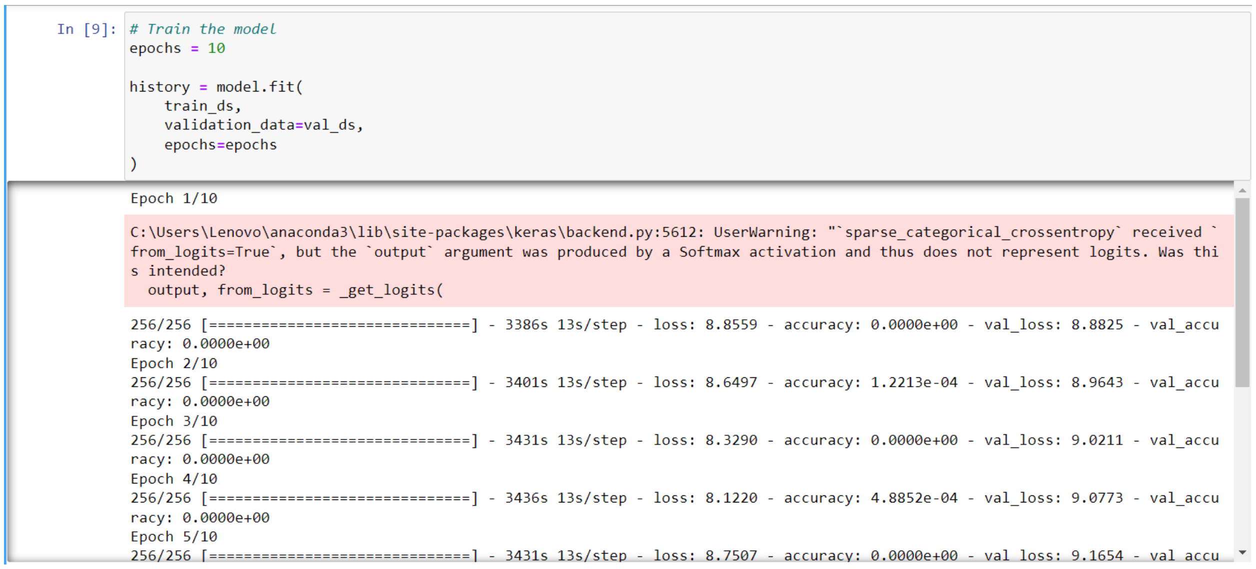


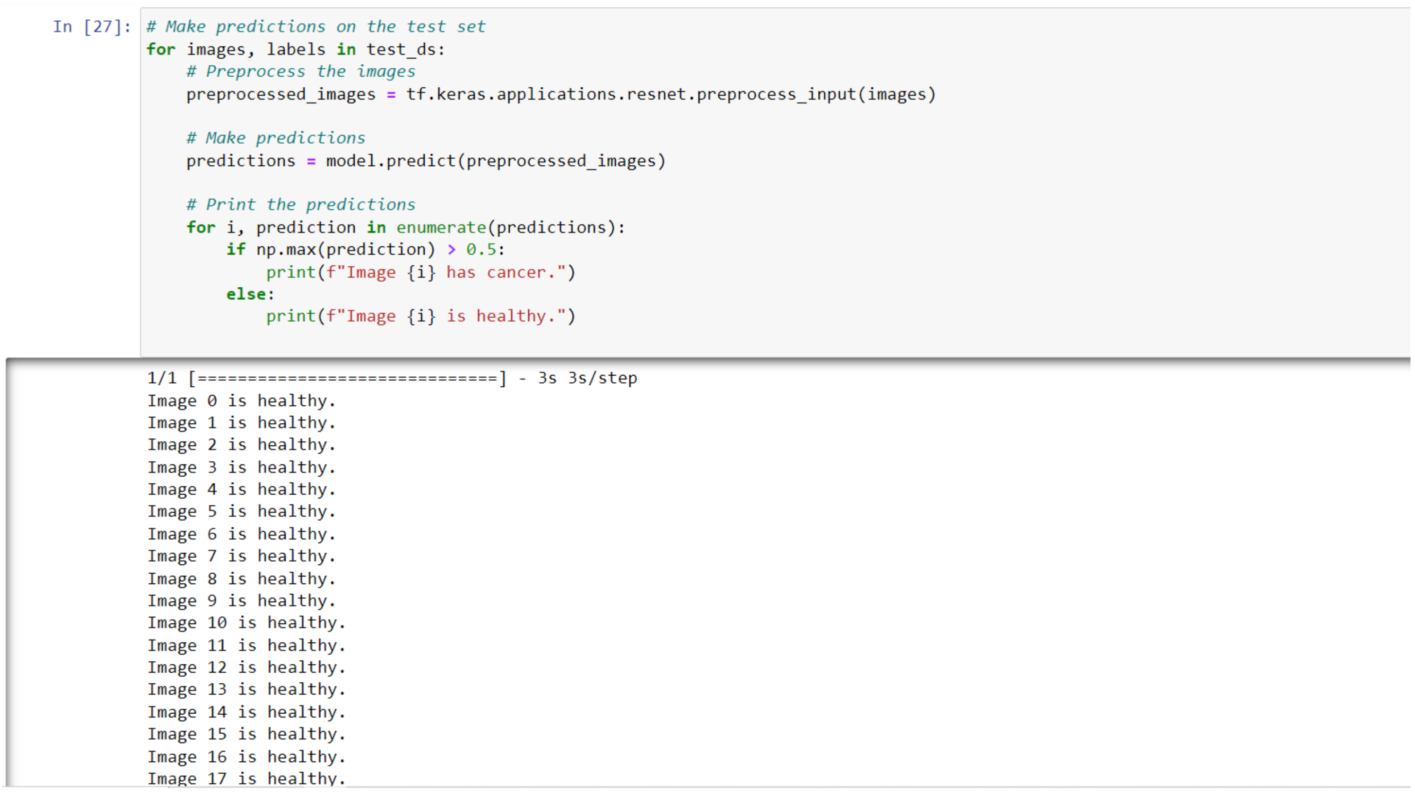
```
In [9]: # Train the model
        epochs = 10

        history = model.fit(
            train_ds,
            validation_data=val_ds,
            epochs=epochs
        )
```

```
Epoch 1/10
C:\Users\Lenovo\anaconda3\lib\site-packages\keras\backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` received `
from_logits=True`, but the `output` argument was produced by a Softmax activation and thus does not represent logits. Was thi
s intended?
  output, from_logits = _get_logits(
256/256 [==============================] - 3386s 13s/step - loss: 8.8559 - accuracy: 0.0000e+00 - val_loss: 8.8825 - val_accu
racy: 0.0000e+00
Epoch 2/10
256/256 [==============================] - 3401s 13s/step - loss: 8.6497 - accuracy: 1.2213e-04 - val_loss: 8.9643 - val_accu
racy: 0.0000e+00
Epoch 3/10
256/256 [==============================] - 3431s 13s/step - loss: 8.3290 - accuracy: 0.0000e+00 - val_loss: 9.0211 - val_accu
racy: 0.0000e+00
Epoch 4/10
256/256 [==============================] - 3436s 13s/step - loss: 8.1220 - accuracy: 4.8852e-04 - val_loss: 9.0773 - val_accu
racy: 0.0000e+00
Epoch 5/10
256/256 [==============================] - 3431s 13s/step - loss: 8.7507 - accuracy: 0.0000e+00 - val_loss: 9.1654 - val_accu
```

This is set to be the output of a neural network training process. The model have been trained for 10 epochs. The training loss starts at 8.8559 and decreases gradually to 8.7075 over the course of the 10 epochs. The training accuracy is very low, at or near 0, throughout the training process. The validation loss starts at 8.8825 and increases to 9.5237 over the course of the 10 epochs. The validation accuracy is also very low, at or near 0, throughout the training process. This suggests that the model is not performing well on the given task and may require further tuning or adjustment.

So the final output predicted to be all healthy which is logically not true as it shown in the next figure:



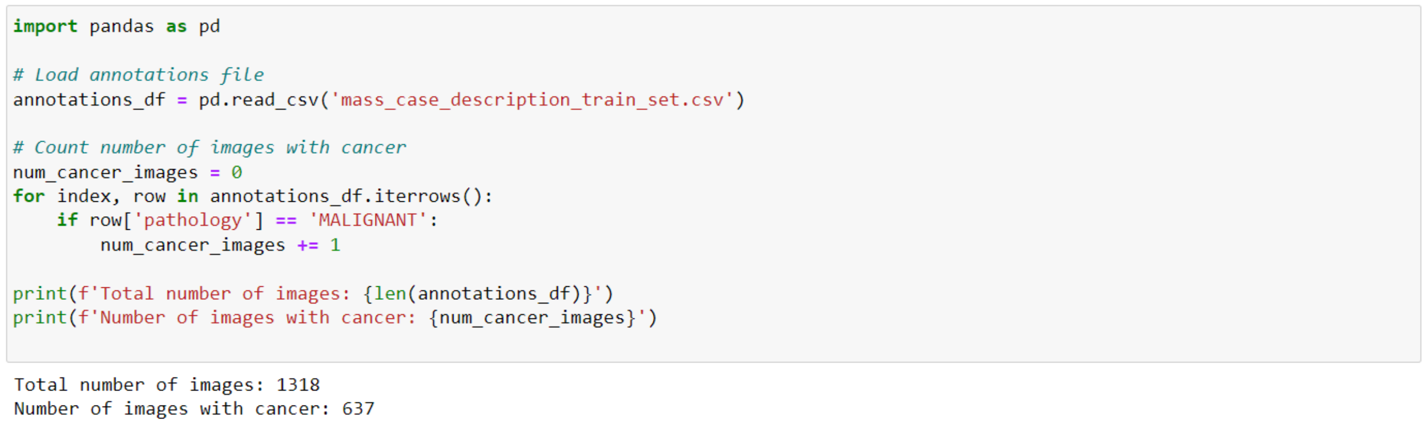
```
In [27]: # Make predictions on the test set
         for images, labels in test_ds:
             # Preprocess the images
             preprocessed_images = tf.keras.applications.resnet.preprocess_input(images)

             # Make predictions
             predictions = model.predict(preprocessed_images)

             # Print the predictions
             for i, prediction in enumerate(predictions):
                 if np.max(prediction) > 0.5:
                     print(f"Image {i} has cancer.")
                 else:
                     print(f"Image {i} is healthy.")
```

```
1/1 [==============================] - 3s 3s/step
Image 0 is healthy.
Image 1 is healthy.
Image 2 is healthy.
Image 3 is healthy.
Image 4 is healthy.
Image 5 is healthy.
Image 6 is healthy.
Image 7 is healthy.
Image 8 is healthy.
Image 9 is healthy.
Image 10 is healthy.
Image 11 is healthy.
Image 12 is healthy.
Image 13 is healthy.
Image 14 is healthy.
Image 15 is healthy.
Image 16 is healthy.
Image 17 is healthy.
```

After that I created a new notebook to test on how many images have cancer, and the output was:



```
import pandas as pd

# Load annotations file
annotations_df = pd.read_csv('mass_case_description_train_set.csv')

# Count number of images with cancer
num_cancer_images = 0
for index, row in annotations_df.iterrows():
    if row['pathology'] == 'MALIGNANT':
        num_cancer_images += 1

print(f'Total number of images: {len(annotations_df)}')
print(f'Number of images with cancer: {num_cancer_images}')
```

```
Total number of images: 1318
Number of images with cancer: 637
```

As it shown there are 637 images have cancer so I rewrote a more accurate code which trained the model well for prediction.

It is located in a notepad file called “Trained Well.txt”.