Breast Cancer Classification

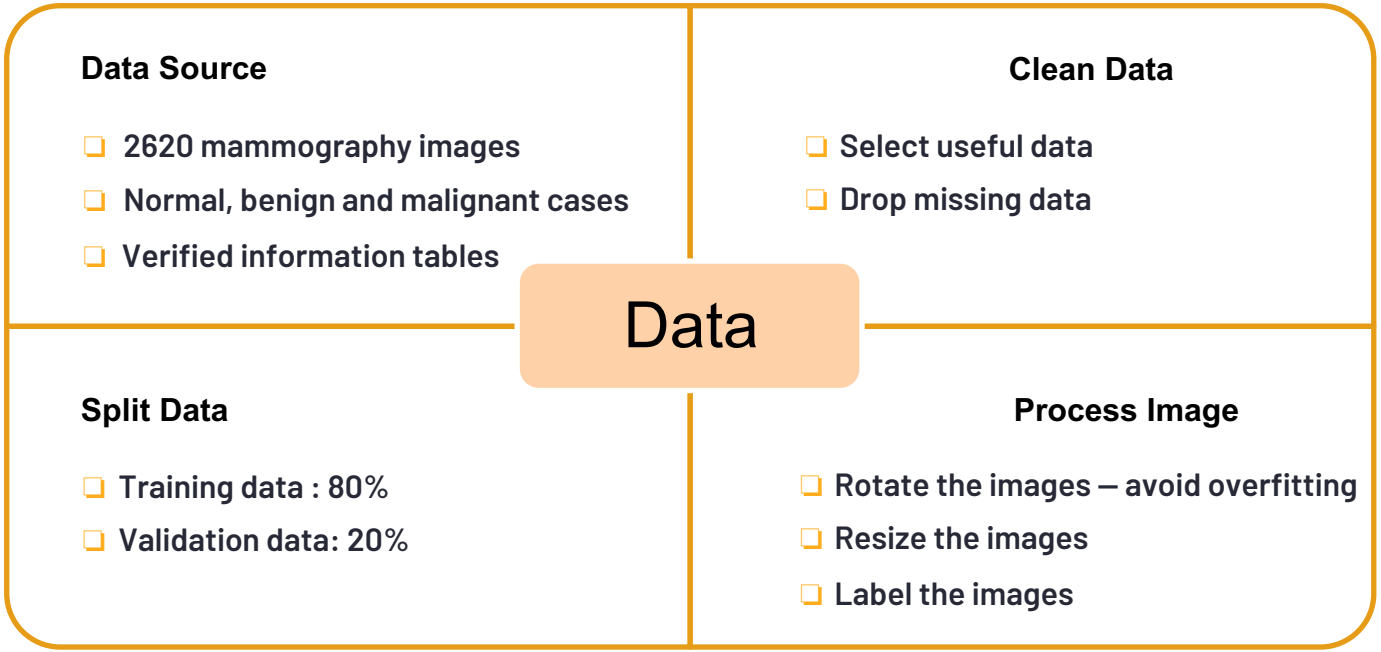
Team Name: Prophet

Team Members: Boyu Lu, Shihui Chen, Meng Zhou

Background

- Depend on digital Biomedical Photography Analysis----Histopathological images
- Classify images between Benign, Benign without call back and Malignant
- Convolutional Neural Network

Data Pre-process



https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM

Model & Core Algorithm

- **Model : Convolutional Neural Network**
  - Automatically detecting the important features without any human supervision
- **Core Algorithm :**
  - Create the convolutional base
  - Add Dense layers on top
  - Compile and train model
  - Evaluate the model

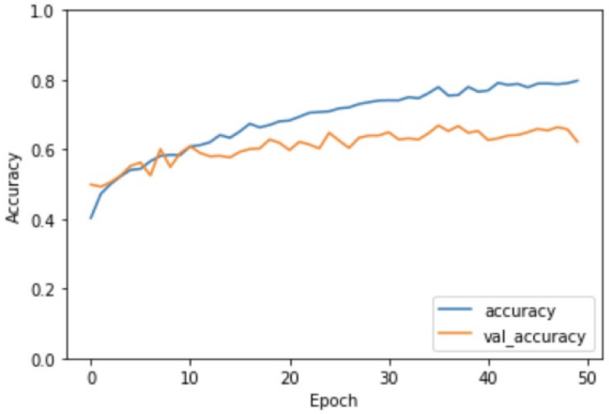
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu', input_dim=X_train.shape[1],
                       kernel_regularizer=regularizers.l2(0.01)))
model.add(layers.Dropout(0.5))
# 3 classes with softmax activation for multi-class
model.add(layers.Dense(3, activation='softmax'))
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```

Result Analysis

- Training Accuracy: **79.74%**
- Validation Accuracy: **62.20%**
- Test Accuracy: **54.14%**

```
20/20 - 9s - loss: 1.5724 - accuracy: 0.6220
```



Former Model Analysis 1

Problem: **Overfitting**

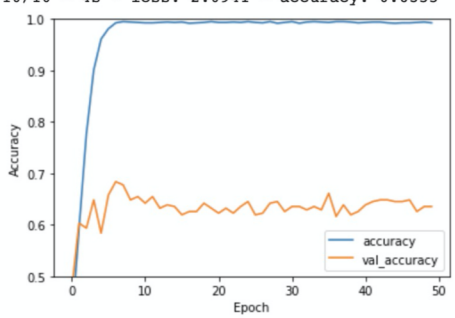
Solution:

- **Expand data volume**
  - **Rotate original images**
- **Add regularization**
- **Dropout features**

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
```

```
10/10 - 4s - loss: 2.0941 - accuracy: 0.6355
```



Former Model Analysis 2

Problem: **Accuracy fluctuating**

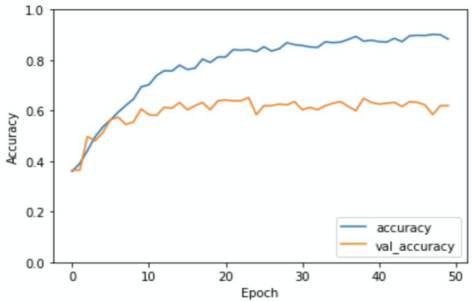
Solution:

- **Increase batch size**

```
[83] plt.plot(history.history['accuracy'], label='accuracy')
     plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
     plt.xlabel('Epoch')
     plt.ylabel('Accuracy')
     plt.ylim([0, 1])
     plt.legend(loc='lower right')

     test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
```

```
10/10 - 5s - loss: 1.7174 - accuracy: 0.6194
```



Further Optimization

- Limited volume of training data
  - Find larger dataset
  - Add more rotated images
- Poor accuracy
  - Create another prediction model
  - Add boost process
- Not suitable for complex condition
  - Add different layers (10 scale or even more)