Statistical Learning and Data Mining

November 22, 2019

1 Abstract

These days, skin cancer is categorized as one of the most dangerous forms of cancers found in humans. The early detection of skin cancer, especially malignant type, can be tremendously advantageous as it may increase the survival rate of the patients. In order to provide precise detection, other than being done by dermatologists, computers can be helpful in the medical field by providing support in the diagnosis. In this paper, we present computer-aided methodology for the detection of skin cancer by using neural network and image processing tools. The input to the system includes the skin lesion images which then analyzed to conclude and classify the presence of skin cancer. The most prominent result of our computer investigation provides comparable result with human analysis.

2 Literature Review

Skin Cancer has been commended as one of the top ten most common cancer in Singapore, according to the Singapore Cancer Registry report [1]. There were recorded 1822 male sufferers in the 2011-2015 period and 1404 female sufferers at the same time. Like all types of cancers, skin cancer is classified as dangerous, especially if it is categorized as malignant cancer [2]. The early stage of this cancer is treatable, nonetheless, the late-stage can be deadly. Aggressive cancer cells may potentially spread deeper into the skin or onto other parts of the body. Thus far, specialists in [3] believe that early and accurate diagnosis is seen as important to reduce morbidity and increase patient's survival.

The most common way of identifying skin cancers, dermoscopy, detects the skin moles using the in-vivo technique with fluid applied to the skin lesion to render the substructure of the skin [3]. This method, however, is highly dependent on the ability of the dermatologists which in turn results in lower accuracy in the hands of in-experience dermatologists. Even under experienced dermatologist supervision, this analysis will incur an accuracy between 75% to 84% only[4]. Noticing the importance of accurate diagnosis, therefore, computer interpretation is seen as crucial in aiding skin cancer detection.

Compared to human eye analysis which depends on the ABCDE (asymmetry, border, color, diameter, and evolution) features of the lesion [5], computer analysis on the skin moles will be based on feature extraction and feature classification of the images. The algorithms which will be used in this project are Support Vector Machine (SVM), Random Forest, and Convolutional Neural Network (CNN). The outcome of these algorithms will be compared to detemine which one produces the

best accuracy. In order to provide significant information to alleviate human analysis, the model is targeted to achieve more than 84% accuracy.

3 Data Summary

The data used consists of 3301 images, where 1802 images are benign and 1499 images are malignant. The test set consists of 661 images with 361 benign images and 301 malignant images. Subsequently, the train set consists of 2639 images with 1441 benign images and 1198 malignant images. The data is sourced from the International Skin Imaging Collaboration (ISIC) archive. All of the images are standardized to $224 \times 224 \times 3$ size. Below are the examples of benign and malignant skin cancer mole.

4 Feature Extraction

In this problem, we are interested in the set of coloured pictures that consist of both benign and malignant cancers. Since the pictures are in color, we can extract the pictures as the combination of 3 RGB matrices of the same size (The 3 matrices represent the RGB colour scheme). We are going to use two different methods of images feature extractions.

Firstly, recall that all the pictures have the same resolution - $224 \times 224 \times 3$. In the modeling process, we will flatten the 3-dimensional array into 1 vector and we are going to use this vector (Every pixel of the images) as our feature for the model training. In other words, we will have 150,528 many features for each image.

Apart from that, we are going to make use of the so-called "Edge Features" where we could identify and determine any sharp changes in colour from the given RGB matrices. This is possible by using the Prewitt kernel.

Prewitt Kernel are two 3×3 matrices which are convolved with the original image to calculate approximates of the derivatives - one for horizontal changes, and one for the vertical.

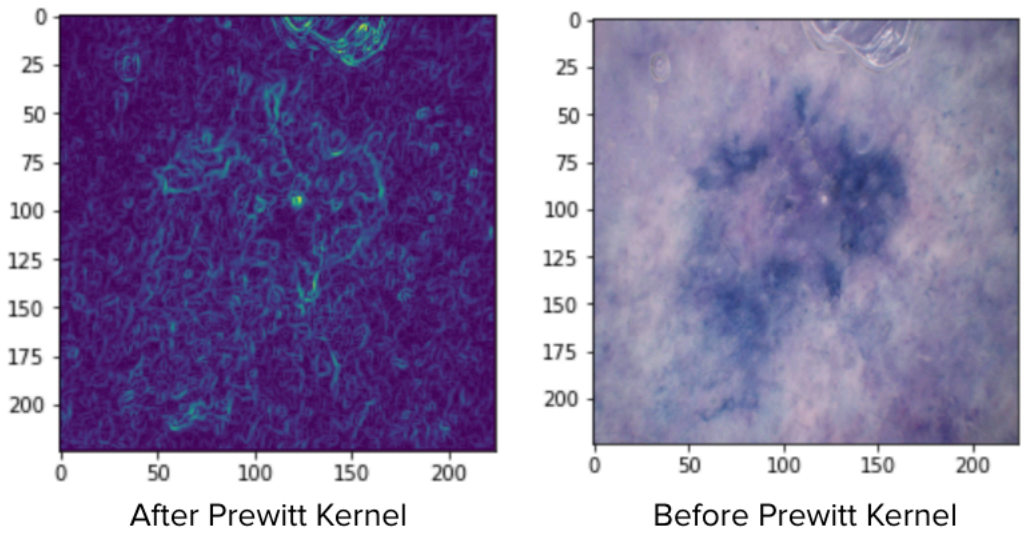
To extract the edges from an image, the two Prewitt Kernel matrices will be convolved with the images (Exactly the same process as in the convolutional layer). The convolution from the first and second matrix will be called G_x and G_y and they represent the horizontal and vertical derivative approximations respectively as illustrated below

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Eventually, in the final step, at each point in the image the resulting gradient approximation can be combined to give the gradient magnitude using :

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

The resulted image will be similar to of silhouette of the shapes from the original image. By doing this, the emphasis is more on the shape or the geometry of the images where the reader can refer to the images below. We are going to make use of Prewitt Kernel only in the SVM and Random Forest models as the CNN model already has its feature extraction from the maxpooling layer. In the end, we will compare these two feature extraction methods and decide which one suits our model the best.

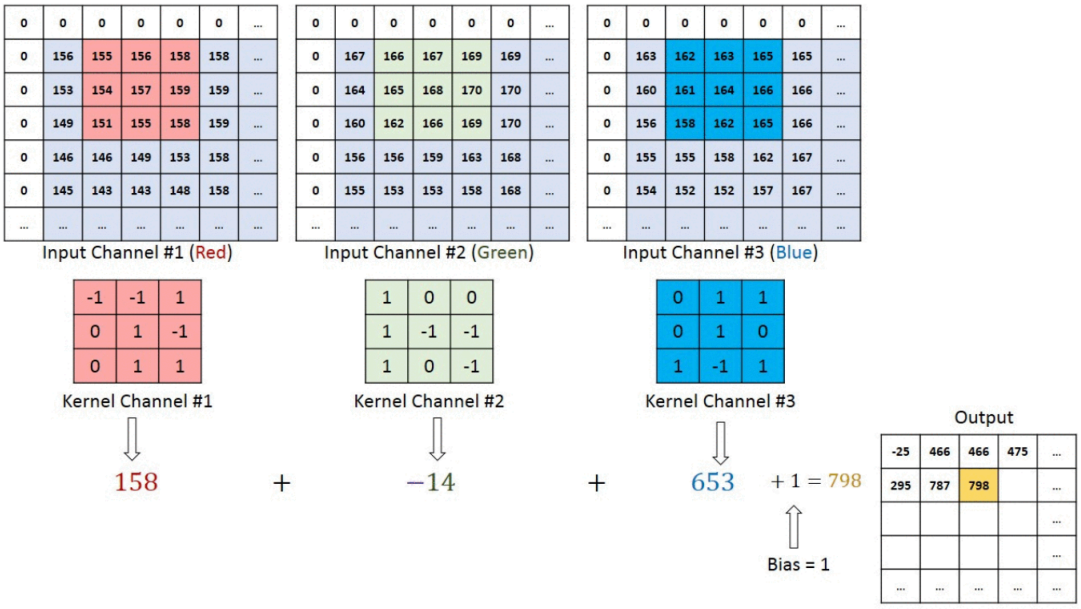


5 Convolutional Neural Network

Convolutional neural network (CNN) is one type of Artificial Neural Network that is compatible in doing image classification. The reason for that is CNN is able to preserve 3 dimensional input and extract features out of it. For example, suppose the input image with dimension (224,224,3), which means that the image is coloured or RGB, CNN will be able to take this as its input and extract features out of it. In the subsequent parts, we will denote the dimension of an image as (width, length, channel). CNN has some major components when creating its architecture, as the following.

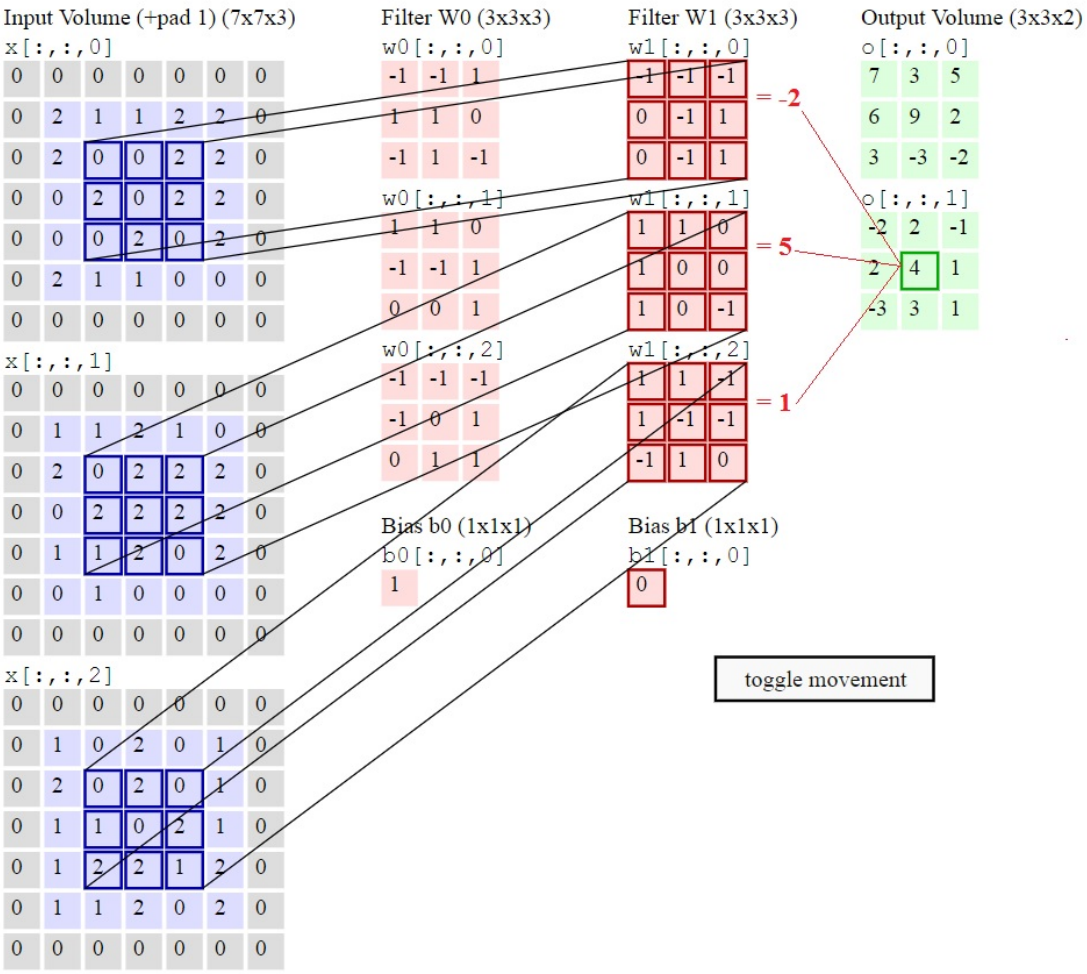
5.1 Convolutional Layer

This layer is the crucial building block in CNN architecture. This layer will filter each entry from each channel from the image by doing dot product with a square matrix (filter matrix) and result in another matrix called feature map. These filter matrix's entries and the bias are parameters to be optimized when training the model. Reader may also refer to the following figure to better understand the idea of convolutional layer.



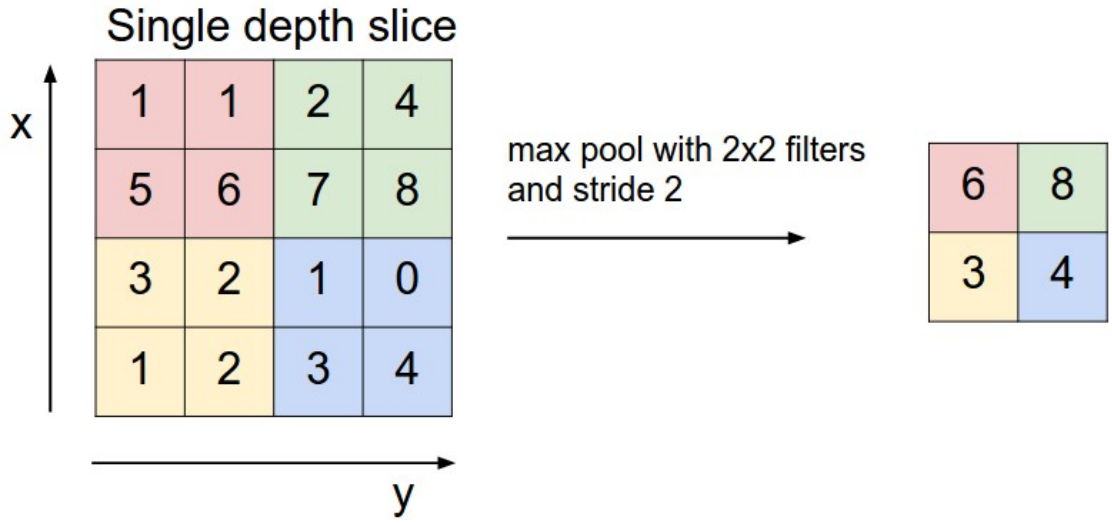
Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

To better summarize this part, we will use the figure above. The input of convolutional layer consists of 3 input channels (red, green, and blue), and will output a matrix called feature map. In the case where we have 3 channels or colored images, the number of filter matrices corresponds to the number of channel desired in the feature map. For example, if the output image will have 1 output channel, then we will have 3 kernel channel (filter matrix) corresponding to each channel, see figure above. If one is expecting output with 2 channels, then each channel have 2 filter (kernel) matrices, see the figure below. Then convolutional layer will take the sum for each dot product between each channel with the filter to produce a feature map, which is the final output. The figure below will summarize the whole process of convolutional layer with $6 \times 3 \times 3 + 2 = 56$ parameters.



5.2 Pooling

This layer will reduce the size of a matrix by the size of pool input by user. Suppose that the input matrix has size (32,32,3) and pool size has size (2,2) then the output will be matrix with size (16,16,3). Generally, pooling will reduce the size of the input matrix by factor k where k is the size of pool. There are two types of pooling, average pooling and max pooling. Observe the figure below, the input matrix of size (8,8,1) will have output matrix of size (4,4,1). Moreover, (2,2) Max pooling will take the maximum value for each (2,2) sub-matrix of the input. Notice that these sub-matrices must not overlap with each other. While (2,2) average pooling will take the average value of each (2,2) sub-matrix. Notice that there is no trainable parameter in the pooling layer. It is used for dimension reduction of an image

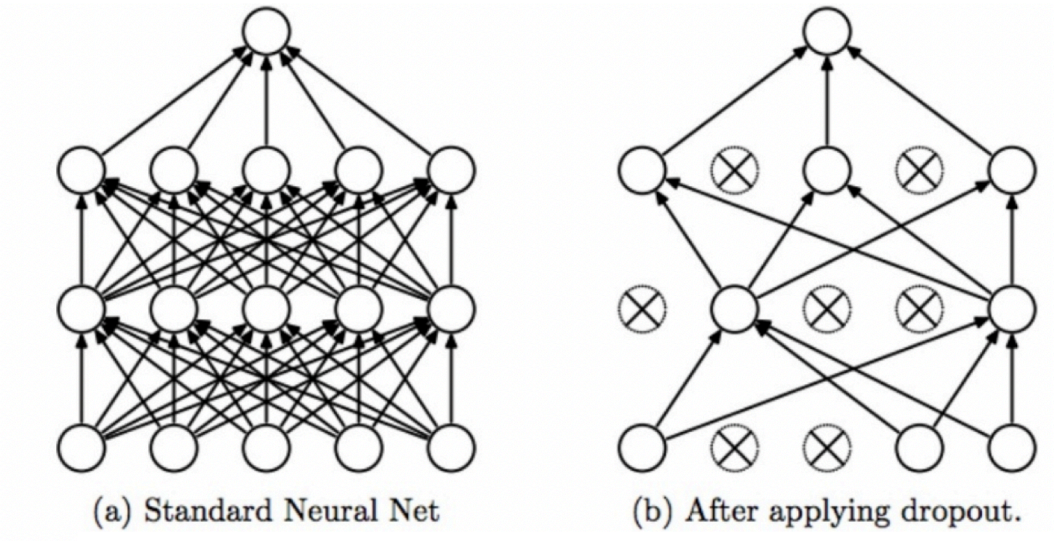


5.3 Dropout rate

This rate can be perceived as a regularization method to avoid overfitting. Dropout rate can be understood as a rate/probability that the link of a node from the previous layer and the node from the next layer will be terminated. In other words, this will reduce some parameters to be estimated.

5.4 Dense (Fully Connected Layer)

This is a standard neural network layer where one node will connect to every node in the next layer. However, notice that a standard neural network layer can only take a vector as input, not a matrix with 3 dimension. Therefore, one should flatten the matrix before connect them using fully connected layer. In CNN, we will apply dropout rate to fully connected layer to avoid overfitting. The figure below illustrates the effect of drop out rate in a fully connected layer.



(a) Standard Neural Net (b) After applying dropout.

In this project, we have also implemented our own CNN to do skin cancer image classification. Our model consist of the following layers:

1. Convolutional layer
2. MaxPooling with size (2,2)
3. Dropout rate of 0.2
4. Convolutional layer
5. MaxPooling with size (2,2)
6. Dropout rate of 0.2
7. Fully Connected Layer
8. Drop out rate of 0.5
9. Fully connected layer to predict the output.

Notice that dropout rate for convolution layer is 0.2, it is because CNN model tends to perform better when dropout rate for the convolution layer is 0.2 (Park, S. et. al [8]). While for the fully connected layer, a rule of thumb is to set dropout rate as 0.5. Since in the first paper that introduce dropout rate used 0.5 and it has been shown that it performs very well [9].

6 Model

Our dataset consists of matrices of images with the size of $224 \times 224 \times 3$. And notice that we are going to use every pixel from our images as the feature. Therefore as previously mentioned, for every image there will be 150,528 features. Here we have the case where the number of observations is much smaller than the number of features. Both Random Forest and Support Vector Machine model provide better accuracy in the case of data with large dimension (Features). Thus, we are going to use Support Vector Machine and Random Forest model for this problem.

Notice that in our case, we do care about high false negative rate. Since it is more costly when we wrongly detect someone without malignant mole compared to when we wrongly detect someone with malignant mole. Therefore, we will calculate false negative rate for each model and make our conclusion.

6.1 Random Forest Without Prewitt Kernel

Here the model creates a large number of individual decision trees where each one of them will generate a prediction. It is a very good model for classification problem which has large data set and high dimensionality (Which are the case in our project). However by using random forest, we can easily achieve an overfitting results, and it is also difficult to interpret the model.

The accuracy of our Random Forest model is 78.78% and the false negative rate is 24.52%

6.2 Random Forest With Prewitt Kernel

In this case, we emphasize more on the shape of the pictures. In a sense, we are using the edge features purely to train this model.

The accuracy of our Random Forest model (Using Prewitt Kernel) is 69.09% and the false negative rate is 30.16%

6.3 Support Vector Machine Without Prewitt Kernel

A model that based on the idea of finding a hyperplane that best divides the dataset into two classes (In our case benign and malignant). It is one of the models that works very well in high dimensional spaces and it is memory efficient. However this model works really well on smaller dataset and most of the times is not good for large data sets, which is the case in our project.

The accuracy of our Support Vector Machine model is 78.33% and the false negative rate is 21.66%

6.4 Support Vector Machine With Prewitt Kernel

We fit the SVM model by using edge features, and here are the results

The accuracy of our Support Vector Machine model (Using Prewitt Kernel) is 71.82% and the false negative rate is 30.34%

6.5 CNN

This model is created specifically for images as its input. The model works by assigning importance (learnable weights and biases) to various aspects/objects in the image and later differentiating one from the others. Therefore we believe that CNN model will be the best model for our image datasets.

The accuracy of our CNN model is 84.47% and the false negative rate is 24.01%

The figure below are the confusion matrix of our 5 models.

SVM	Predicted 0	Predicted 1
Actual 0	300	83
Actual 1	60	217

Accuracy : 78.33%

Random Forest	Predicted 0	Predicted 1
Actual 0	283	63
Actual 1	77	237

Accuracy : 78.78%

Without Prewitt Kernel

SVM	Predicted 0	Predicted 1
Actual 0	272	98
Actual 1	88	202

Accuracy : 71.82%

Random Forest	Predicted 0	Predicted 1
Actual 0	287	131
Actual 1	73	169

Accuracy : 69.09%

With Prewitt Kernel

CNN	Predicted 0	Predicted 1
Actual 0	269	12
Actual 1	91	288

Accuracy : 84.47%

From the result above, we can clearly see that the usage of Prewitt Kernel decreased the accuracy and increased the false negative rate of our models. In this case we conclude that Prewitt Kernel does not perform better than using all the image pixels as the feature. The main reason why Prewitt Kernel does not work well is the nature of the images processed by this method. Images that are processed by Prewitt Kernel will have more emphasis on the shape and pattern rather than the color. However in the case of skin cancer prediction, the color intensity and gradient are really important to determine which one is the benign case and which one is not.

From all the models, we can clearly see that the CNN model has the highest accuracy. However it still has a quite high false negative rate, thus in further studies a reduction in the threshold need to be done.

In conclusion, the CNN model is the best model to be used in our study -predicting the skin cancer type based on the given picture.

7 Codes

[1]:
```
# Import and define function for data pre-processing
```

[3]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
```

[4]:
```
import cv2
import os

def load_images_from_folder(folder):
    images = []
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder,filename))
        if img is not None:
            images.append(img)
    return images
```

[5]:
```
benign=load_images_from_folder('/Users/gilbert/Documents/Statistical Leaning/
↪skin-cancer-malignant-vs-benign/train/benign')
```

[6]:
```
malignant = load_images_from_folder('/Users/gilbert/Documents/Statistical␣
↪Leaning/skin-cancer-malignant-vs-benign/train/malignant')
```

DATASET FOR CNN, CNN model do not require feature selection

[16]:
```
X_train_cnn = benign+malignant
Y_train_cnn = [0]*len(benign)+[1]*len(malignant)
X_train_cnn,Y_train_cnn = shuffle(X_train_cnn,Y_train_cnn)
```

```
X_train_cnn = np.array(X_train_cnn)
Y_train_cnn = np.array(Y_train_cnn)
```

[17]:
```
benign_test =load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/benign')
malignant_test = load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/malignant')
X_test_cnn = benign_test+malignant_test
Y_test_cnn = [0]*len(benign_test)+[1]*len(malignant_test)
X_test_cnn,Y_test_cnn = shuffle(X_test_cnn,Y_test_cnn)
X_test_cnn = np.array(X_test_cnn)
Y_test_cnn = np.array(Y_test_cnn)
```

THIS IS THE DATASET FOR MACHINE LEARNING Model, feature selection is being done here. We are extracting each pixels as the feature

[31]:
```
benign_data = []
for i in range(len(benign)):
    benign_data += [np.array(benign[i]).flatten()]
```

[32]:
```
malignant_data = []
for i in range(len(malignant)):
    malignant_data += [np.array(malignant[i]).flatten()]
```

[7]:
```
#preparing training set
X_train = benign_data + malignant_data
```

[8]:
```
y_train = [0]*len(benign_data)+[1]*len(malignant_data)
```

[9]:
```
from sklearn.utils import shuffle
X_train,y_train = shuffle(X_train,y_train)
```

[33]:
```
#preparing test set
X_test1 = []
X_pretest = load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/malignant')
for i in range(len(X_pretest)):
    X_test1 += [np.array(X_pretest[i].flatten())]
X_test0 = []
X_pretest = load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/benign')
for i in range(len(X_pretest)):
    X_test0 += [np.array(X_pretest[i].flatten())]
X_test = X_test0+X_test1
y_test = [0]*len(X_test0)+[1]*len(X_test1)
```

[ ]:
```
# To print out K cross validation result
```

[16]:
```python
from sklearn.model_selection import GridSearchCV
def print_training(model):
    print('BEST PARAMETERS: {}\n'.format(model.best_params_))
    means = model.cv_results_['mean_test_score']
    stds = model.cv_results_['std_test_score']
    for mean, std, params in zip(means,stds,model.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean,3),round(std*2,3),params))
```

SVM K-Cross Validation and Model Fitting

[14]:
```python
from sklearn.svm import SVC
```

[15]:
```python
model_2 = SVC()
params_svc = {
    'kernel':['rbf','linear']
}
svc_cv = GridSearchCV(model_2,params_svc,cv=5)
svc_cv.fit(X_train[:500],y_train[:500])
```

```
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[15]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
```

```
                            decision_function_shape='ovr', degree=3,
                            gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                            probability=False, random_state=None, shrinking=True,
                            tol=0.001, verbose=False),
                 iid='warn', n_jobs=None, param_grid={'kernel': ['rbf', 'linear']},
                 pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                 scoring=None, verbose=0)
```

[16]:
```
print_training(svc_cv)
```

```
BEST PARAMETERS: {'kernel': 'linear'}

0.588 (+/-0.005) for {'kernel': 'rbf'}
0.762 (+/-0.099) for {'kernel': 'linear'}
```

[17]:
```
#fitting svm based on the suggested kernel in K cross validation
model_svm = SVC(kernel='linear')
model_svm.fit(X_train,y_train)
```

[17]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

Random Forest (bench mark model)

[20]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(X_train,y_train)
```

```
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

[20]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Convolutional Neural Network (CNN)

[92]:
```
#CNN model
from keras.datasets import cifar10
from keras.models import Sequential
```

```
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils

#prepare dataset for training and test (X variable)
X_train_final = X_train_cnn.astype('float32')
X_test_final = X_test_cnn.astype('float32')
X_train_final = X_train_final / 255.0
X_test_final = X_test_final / 255.0
#prepare dataset for training and test (Y variable)
y_train_final = np_utils.to_categorical(Y_train_cnn)
y_test_final = np_utils.to_categorical(Y_test_cnn)
num_classes = y_test_final.shape[1]
# Create the model
## We did try out 2 times convolution layer stick together, but it does not␣
 ↪work well in our case.
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(224, 224, 3), padding='same',␣
 ↪activation='relu', kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D())
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same',␣
 ↪kernel_constraint=maxnorm(3)))
model.add(MaxPooling2D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='sigmoid'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
model.summary()
# Fit the model
model.fit(X_train_final, y_train_final, validation_data=(X_test_final,␣
 ↪y_test_final), epochs=epochs, batch_size=32)
# Final evaluation of the model
scores = model.evaluate(X_test_final, y_test_final, verbose=0)
```

```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Model: "sequential_25"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_71 (Conv2D)           (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d_43 (MaxPooling (None, 112, 112, 32)      0
_________________________________________________________________
dropout_54 (Dropout)         (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_72 (Conv2D)           (None, 112, 112, 64)      18496
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 56, 56, 64)        0
_________________________________________________________________
dropout_55 (Dropout)         (None, 56, 56, 64)        0
_________________________________________________________________
flatten_24 (Flatten)         (None, 200704)            0
_________________________________________________________________
dense_47 (Dense)             (None, 512)               102760960
_________________________________________________________________
dropout_56 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_48 (Dense)             (None, 2)                 1026
=================================================================
Total params: 102,781,378
Trainable params: 102,781,378
Non-trainable params: 0
_________________________________________________________________
Train on 2637 samples, validate on 660 samples
Epoch 1/25
2637/2637 [==============================] - 227s 86ms/step - loss: 0.6196 -
accuracy: 0.6399 - val_loss: 0.4874 - val_accuracy: 0.7364
Epoch 2/25
2637/2637 [==============================] - 187s 71ms/step - loss: 0.5455 -
accuracy: 0.7226 - val_loss: 0.5723 - val_accuracy: 0.7561
Epoch 3/25
2637/2637 [==============================] - 175s 66ms/step - loss: 0.4910 -
accuracy: 0.7613 - val_loss: 0.4638 - val_accuracy: 0.7826
Epoch 4/25
2637/2637 [==============================] - 204s 77ms/step - loss: 0.4414 -
accuracy: 0.7854 - val_loss: 0.3937 - val_accuracy: 0.7712
Epoch 5/25
2637/2637 [==============================] - 170s 64ms/step - loss: 0.4806 -
accuracy: 0.7757 - val_loss: 0.5597 - val_accuracy: 0.7189
Epoch 6/25
```

```
2637/2637 [==============================] - 176s 67ms/step - loss: 0.4373 -
accuracy: 0.7778 - val_loss: 0.5048 - val_accuracy: 0.7492
Epoch 7/25
2637/2637 [==============================] - 171s 65ms/step - loss: 0.4130 -
accuracy: 0.7971 - val_loss: 0.4135 - val_accuracy: 0.8091
Epoch 8/25
2637/2637 [==============================] - 172s 65ms/step - loss: 0.4024 -
accuracy: 0.8036 - val_loss: 0.3730 - val_accuracy: 0.8083
Epoch 9/25
2637/2637 [==============================] - 173s 66ms/step - loss: 0.3885 -
accuracy: 0.8108 - val_loss: 0.3631 - val_accuracy: 0.8258
Epoch 10/25
2637/2637 [==============================] - 176s 67ms/step - loss: 0.3685 -
accuracy: 0.8214 - val_loss: 0.3984 - val_accuracy: 0.8121
Epoch 11/25
2637/2637 [==============================] - 174s 66ms/step - loss: 0.3525 -
accuracy: 0.8248 - val_loss: 0.3715 - val_accuracy: 0.8326
Epoch 12/25
2637/2637 [==============================] - 175s 66ms/step - loss: 0.3433 -
accuracy: 0.8303 - val_loss: 0.3486 - val_accuracy: 0.8205
Epoch 13/25
2637/2637 [==============================] - 183s 69ms/step - loss: 0.3300 -
accuracy: 0.8419 - val_loss: 0.3355 - val_accuracy: 0.8235
Epoch 14/25
2637/2637 [==============================] - 174s 66ms/step - loss: 0.3616 -
accuracy: 0.8276 - val_loss: 0.3579 - val_accuracy: 0.8364
Epoch 15/25
2637/2637 [==============================] - 176s 67ms/step - loss: 0.3490 -
accuracy: 0.8318 - val_loss: 0.3987 - val_accuracy: 0.8212
Epoch 16/25
2637/2637 [==============================] - 171s 65ms/step - loss: 0.3227 -
accuracy: 0.8396 - val_loss: 0.3611 - val_accuracy: 0.8318
Epoch 17/25
2637/2637 [==============================] - 167s 63ms/step - loss: 0.3123 -
accuracy: 0.8481 - val_loss: 0.4186 - val_accuracy: 0.7856
Epoch 18/25
2637/2637 [==============================] - 172s 65ms/step - loss: 0.3170 -
accuracy: 0.8470 - val_loss: 0.3442 - val_accuracy: 0.8356
Epoch 19/25
2637/2637 [==============================] - 213s 81ms/step - loss: 0.3011 -
accuracy: 0.8563 - val_loss: 0.3295 - val_accuracy: 0.8417
Epoch 20/25
2637/2637 [==============================] - 191s 72ms/step - loss: 0.2878 -
accuracy: 0.8701 - val_loss: 0.3263 - val_accuracy: 0.8386
Epoch 21/25
2637/2637 [==============================] - 177s 67ms/step - loss: 0.2807 -
accuracy: 0.8673 - val_loss: 0.3323 - val_accuracy: 0.8432
Epoch 22/25
```

```
2637/2637 [==============================] - 184s 70ms/step - loss: 0.2758 -
accuracy: 0.8741 - val_loss: 0.3382 - val_accuracy: 0.8227
Epoch 23/25
2637/2637 [==============================] - 201s 76ms/step - loss: 0.2776 -
accuracy: 0.8661 - val_loss: 0.3419 - val_accuracy: 0.8492
Epoch 24/25
2637/2637 [==============================] - 167s 64ms/step - loss: 0.2692 -
accuracy: 0.8790 - val_loss: 0.3334 - val_accuracy: 0.8386
Epoch 25/25
2637/2637 [==============================] - 164s 62ms/step - loss: 0.2505 -
accuracy: 0.8838 - val_loss: 0.3351 - val_accuracy: 0.8447
Accuracy: 84.47%
```

Model Evaluation

[35]:
```
import joblib
rf_model = joblib.load('../../gilbert/Documents/Statistical Leaning/rf_model.
 ↪joblib')
model_svm = joblib.load('../../gilbert/Documents/Statistical Leaning/model_svm.
 ↪joblib')
```

[34]:
```
#Random Forrest
confusion_matrix(rf_model.predict(X_test),y_test)
```

[34]:
```
array([[283,  63],
       [ 77, 237]])
```

[36]:
```
#SVM
confusion_matrix(model_svm.predict(X_test),y_test)
```

[36]:
```
array([[300,  83],
       [ 60, 217]])
```

[11]:
```
#CNN model
confusion_matrix(cnn_model.predict_classes(X_test.astype('float32')/255.
 ↪0),Y_test)
```

[11]:
```
array([[269,  12],
       [ 91, 288]])
```

Trying different feature selection (Prewitt Kernel)

[1]:
```
from skimage.filters import prewitt
import matplotlib.pyplot as plt
```

[7]:
```
dataset = benign+malignant
prewit_dataset = []
for i in dataset:
```

```python
    result = prewitt(i[:,:,0])
    prewit_dataset += [np.array(result[i]).flatten()]
```

[8]:
```python
label = [0]*len(benign)+[1]*len(malignant)
```

[9]:
```python
from sklearn.utils import shuffle
prewit_dataset,prewit_label = shuffle(prewit_dataset,label)
```

[10]:
```python
benign_test =load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/benign')
malignant_test = load_images_from_folder('/Users/gilbert/Documents/Statistical␣
 ↪Leaning/skin-cancer-malignant-vs-benign/test/malignant')
testset = benign_test+malignant_test
prewit_testset=[]
for i in testset:
    result = prewitt(i[:,:,0])
    prewit_testset += [np.array(result[i]).flatten()]
```

[11]:
```python
prewit_testlabel = [0]*len(benign_test)+[1]*len(malignant_test)
```

[12]:
```python
from sklearn.utils import shuffle
prewit_testset,prewit_testlabel = shuffle(prewit_testset,prewit_testlabel)
```

[17]:
```python
#svm model cross validation for rbf or linear
model_2 = SVC()
params_svc = {
    'kernel':['rbf','linear']
}
svc_cv = GridSearchCV(model_2,params_svc,cv=5)
svc_cv.fit(prewit_dataset[:500],prewit_label[:500])
```

```
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[17]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='warn', n_jobs=None, param_grid={'kernel': ['rbf', 'linear']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

[18]:
```
print_training(svc_cv)
```

```
BEST PARAMETERS: {'kernel': 'linear'}

0.576 (+/-0.004) for {'kernel': 'rbf'}
0.742 (+/-0.033) for {'kernel': 'linear'}
```

[26]:
```
model_svm_prewitt = SVC(kernel="linear")
model_svm_prewitt.fit(prewit_dataset,prewit_label)
```

[26]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

[27]:
```
confusion_matrix(model_svm_prewitt.predict(prewit_testset),prewit_testlabel)
```

[27]:
```
array([[272,  98],
       [ 88, 202]])
```

[24]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model_prewitt = RandomForestClassifier()
rf_model_prewitt.fit(prewit_dataset,prewit_label)
```

```
/Users/gilbert/opt/anaconda3/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
```

```
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

[24]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[25]:
```
confusion_matrix(rf_model_prewitt.predict(prewit_testset),prewit_testlabel)
```

[25]:
```
array([[287, 131],
       [ 73, 169]])
```

8 References

[1] Natural Registry of Diseases Office. (2015). Singapore Cancer Registry Annual Registry Report 2015, 12 -13.

[2] Jain, S., Jagtap, V., & Pise, N. (2015). Computer Aided Melanoma Skin Cancer Detection Using Image Processing. Procedia Computer Science, 48, 735–740. doi: 10.1016/j.procs.2015.04.209

[3] Bafounta, M.L., Beauche, A., Aegerter, P., Saiag P. (2001). Is Dermoscopy (Epiluminescence Microscopy) Useful for the Diagnosis of Melanoma?, 1.

[4] Jaina, S., Jagtap, V., Pise, N. (2015). Computer aided Melanoma skin cancer detection using Image Processing. International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014). doi: 10.1016/j.procs.2015.04.209

[5] Kalouche, S.Vision-Based Classification of Skin Cancer using Deep Learning. Stanford University.

[6] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011a). Flexible, high performance convolutional neural networks for image classification. In Intl. Joint Conference on Artificial Intelligence IJCAI, pages 1237–1242.

[7] Introduction to Convolutional Neural Network. (2018, February 23). Retrieved November 13, 2019, from https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf.

[8] Park, S., & Kwak, N. (2016, November). Analysis on the dropout effect in convolutional neural networks. In Asian Conference on Computer Vision (pp. 189-204). Springer, Cham.

[9] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.