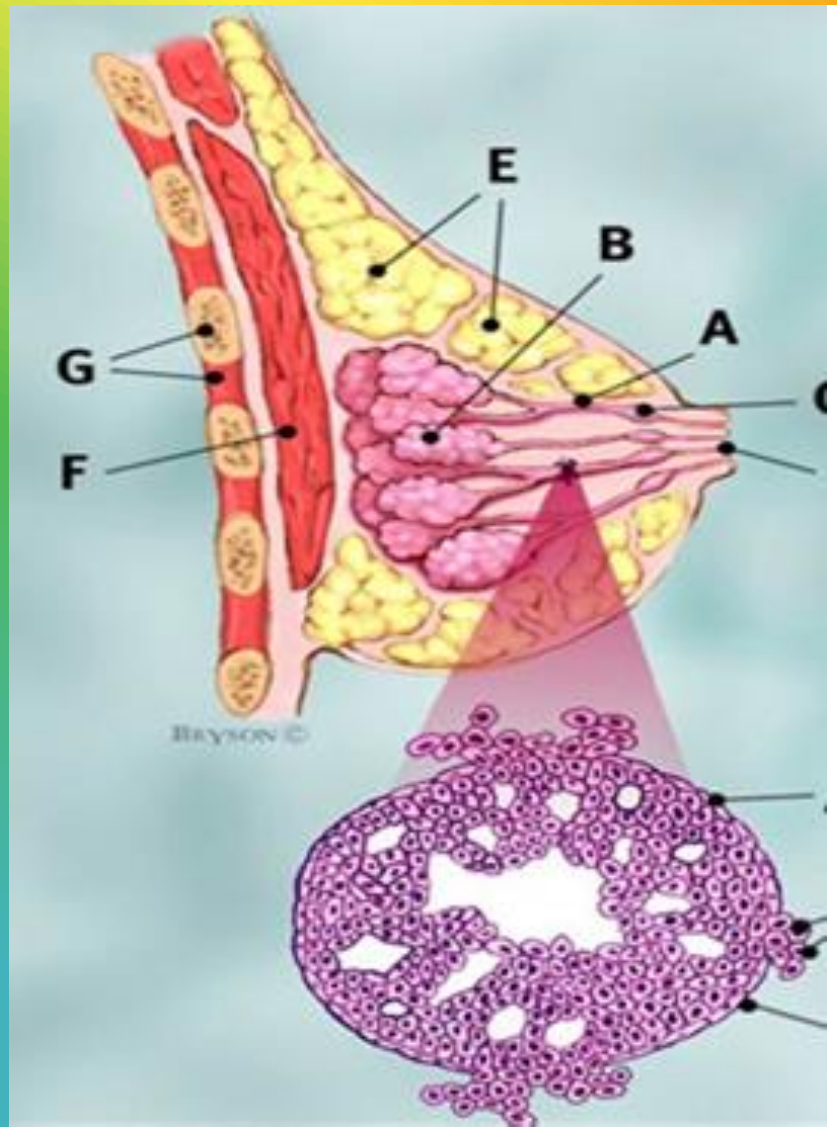


Invasive Ductal Carcinoma (IDC)-A Type of Breast Cancer Prediction

Team 8
Jinning Yang 001029162
Shuyao Zhang001522915

Background

Cancer occurs when some of the body's cells behave abnormally-changing, growing and reproducing more than usual. For breast cancer, the cells in the breast tissue are the ones with abnormal growth. Breast cancer is the kind of cancers that starts in different part of the breasts.

A breast cancer diagnosis comes with a lot of information. It is normal for patient with breast cancer to feel overwhelmed. It is also one of the most common types of cancers for women, sometimes for men as well. There are more than 4700 women were diagnosed with breast cancer in 2016 and 710 of them were dead.

Key Concept

- Invasive Ductal Carcinoma (IDC) is the most common type of all breast cancers. Pathologists normally focus on the regions which contain the Invasive Ductal Carcinoma to evaluate how aggressive the mount sample could be. Typically, they would give an aggressiveness grade to the sample as a result.

Motivation

- In the past, clinicians have used basic software programs, such as Microsoft Excel, to analyze breast cancer. However, it is not adaptable in identifying new variables as well as generating creative and integrative visualizations. It is important to find a better solution for breast cancer prediction.
- Machine Learning is one of the best solutions for this problem. Various machine learning approaches like decision tree, random forest, neural networks, extreme boost, logistic regression could be the answer of predicting breast cancer currently.

Goal

- Giving a patient a slice of tissue to predict if it contains IDC - three possibilities: healthy tissue, IDC, another subtype of breast cancer.
- So far, predictions have been made manually by pathologists, and the predictions vary from expert to expert. Our goal is to help automatically detect tumors (independent of experts).
- We will use K-Fold this method to train and test a model, and we combine K-Fold with Decision Tree, Random Forest Classifier, SVM, Logistic Regression, K Neighbors, or Naive Bayesian. Then we compare the accuracy between different models and try to find the best one.
- We also used data to train and test the CNN (Central Neural Networks) model. We spited the data into 3 parts-test data, train data, and dev data. CNN is expected to give us a precise result after training.

Methodology

- K-Fold, Decision Tree, Random Forest, SVM, Logistic Regression, K-Neighbors, Naive Bayesian, CNN

Jinning Yang Part

Data Description

(277524, 3)

	patient_id	diagnosis	path
0	10295	0	../input/breast-histopathology-images/IDC_regu...
1	10295	0	../input/breast-histopathology-images/IDC_regu...
2	10295	0	../input/breast-histopathology-images/IDC_regu...
3	10295	0	../input/breast-histopathology-images/IDC_regu...
4	10295	0	../input/breast-histopathology-images/IDC_regu...
...	...	...	...
277519	12873	1	../input/breast-histopathology-images/IDC_regu...
277520	12873	1	../input/breast-histopathology-images/IDC_regu...
277521	12873	1	../input/breast-histopathology-images/IDC_regu...
277522	12873	1	../input/breast-histopathology-images/IDC_regu...
277523	12873	1	../input/breast-histopathology-images/IDC_regu...

277524 rows × 3 columns

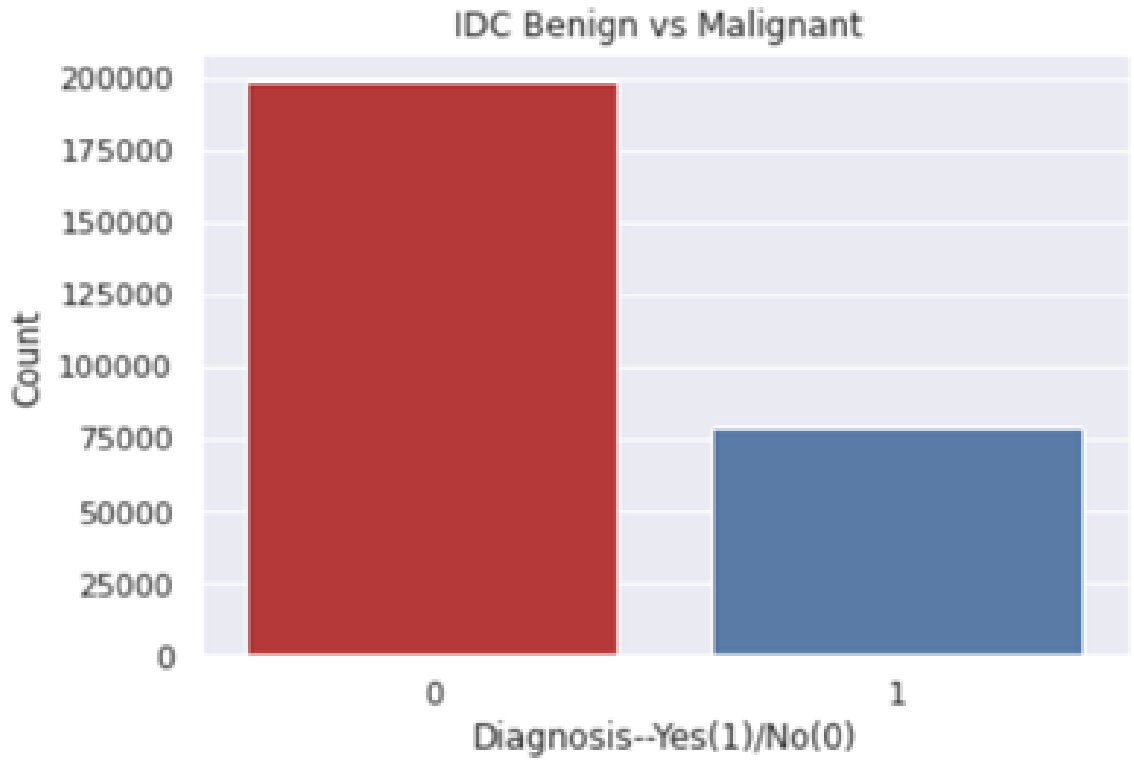
```
data.describe()
```

	diagnosis
count	277524.000000
mean	0.283889
std	0.450884
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 277524 entries, 0 to 277523
Data columns (total 3 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   patient_id  277524 non-null  object
 1   diagnosis   277524 non-null  int64 
 2   path        277524 non-null  object
dtypes: int64(1), object(2)
memory usage: 6.4+ MB
```

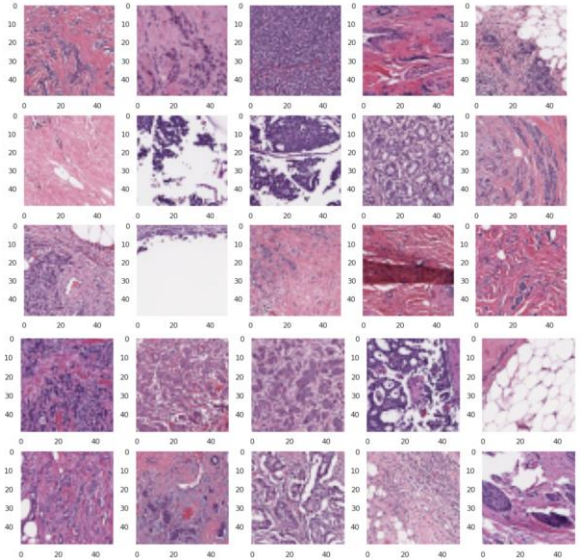
```
0    198738
1     78786
Name: diagnosis, dtype: int64
```



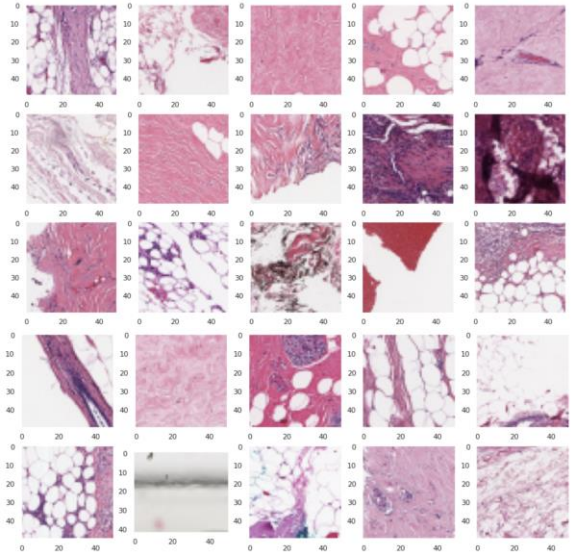

```
The difference between benign and malignant : 119952
```

Comparison of Tissue Samples

Cancer



Healthy



```
(X):
a matrix X, assumes 0-centered
= X.shape
t np.allclose(X.mean(axis=0), np.z
pute covariance matrix
p.dot(X.T, X) / (n-1)
en decomposition
_vals, eigen_vecs = np.linalg.eig(
ject X onto PC space
 = np.dot(X, eigen_vecs)
n X_pca
```

- **Data Standardization: Before applying PCA, we have to bring our data to a common format through standardization. The purpose of doing this is to make sure that variables are internally consistent with each other regardless of their type.**
- **PCA Transformation**
- **Eigenvalues and Vectors Computation: In this step, data compression and dimensionality reduction come into the picture.**

Image Pre-Processing

Apply PCA into our application:

RGB->Grayscale

```
#all
all_image_paths=data['path']
all_rgb_pca=rgb_to_grayscale(all_image_paths)
```

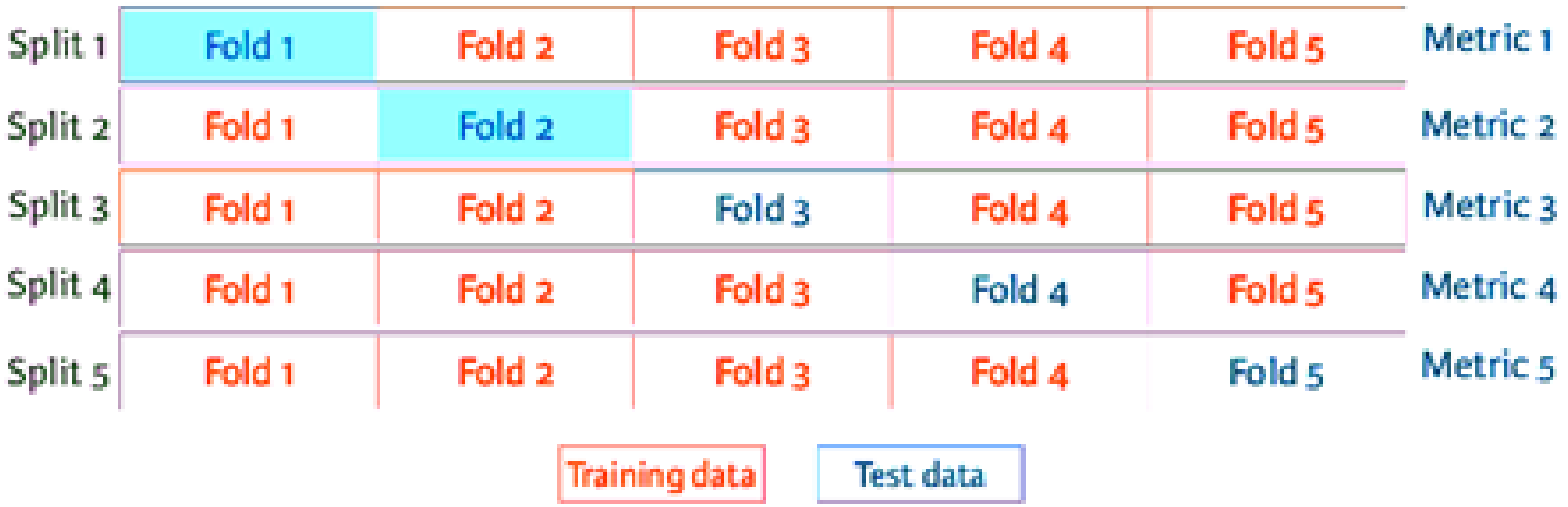
```
def apply_rgb_to_pca(img,pca_object):

    img = img.reshape(-1, 3)

    img = pca_object.transform(img)

    img = img.flatten()
    return img
```

```
#Transorming RGB image paths to single scale image array using PCA
#all
all_image_paths = data['path']
all_image_targets = data['diagnosis']
```

initialize an array containing zeros to place the images in it

```
all_img_array = np.zeros((all_image_paths.shape[0],2500))

for i,all_image_paths in tqdm(enumerate(all_image_paths),total = all_image_paths.shape[0]):
    img = io.imread(all_image_paths)
    try:
        all_img_array[i:img.shape[0]] = apply_rgb_to_pca(img,all_rgb_pca)

    except:
        pass
```

```
#all
all_pca=fit_pixel_pca(all_img_array)
print(all_pca)
```



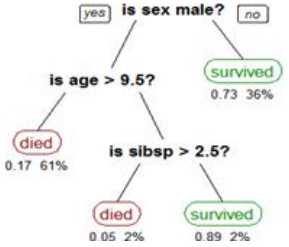
Cross validation (more accurate in calculating accuracy)--K-Fold

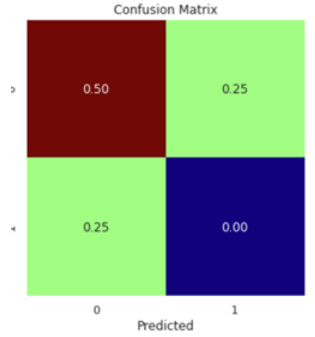
```
def classification_model(model,prediction_input,output):
    kf = KFold(n_splits=5,shuffle=True, random_state=1)
    error = []

    for train, test in kf.split(data):
        train_X = (all_data_train_X[train,:])
        train_y = data['diagnosis'].iloc[train]
        x = train_X[~np.all(train_X == 0, axis=1)]
        y = train_y[~np.all(train_X == 0, axis=1)]
        model.fit(x, y)
        test_X=all_data_train_X[test,:]
        test_y=data['diagnosis'].iloc[test]
        x = test_X[~np.all(test_X == 0, axis=1)]
        y = test_y[~np.all(test_X == 0, axis=1)]
        error.append(model.score(x,y))
        print("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))
    fig, ax = plt.subplots(figsize=(5,5))
    conf_matrix = confusion_matrix(model.predict(x),y)
    conf_matrix = conf_matrix/np.sum(conf_matrix)
    sns.heatmap(conf_matrix, annot=True, fmt=".2f", square=True, cbar=False, cmap=plt.cm.jet,ax=ax)
    ax.set_ylabel('Actual')
    ax.set_xlabel('Predicted')
    ax.set_title('Confusion Matrix')
    plt.show(block=False)
    print("The Accuracy Is : %s" % "{0:.3%}".format(np.mean(error)))
    print(error)
    return error
```

Decision Tree

```
_results=[[],[]]
m sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
assification_model(dt_model,all_data_train_X,
_results[0] += ["Decision Tree"]
_results[1] += [classification_model(dt_model
```

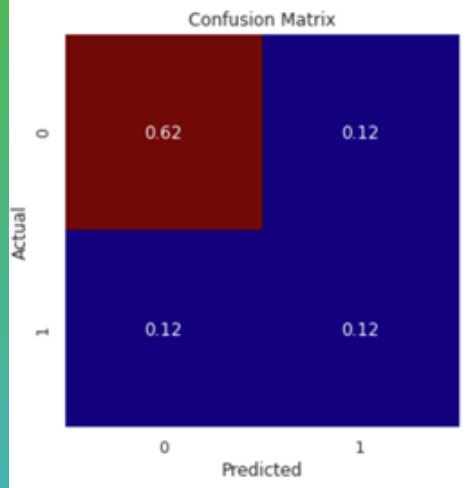
```
ss-Validation Score : 60.000%
ss-Validation Score : 59.167%
ss-Validation Score : 69.074%
ss-Validation Score : 65.442%
ss-Validation Score : 62.354%
```






```
Accuracy Is : 62.354%
, 0.5833333333333334, 0.8888888888888888, 0.5454545
```

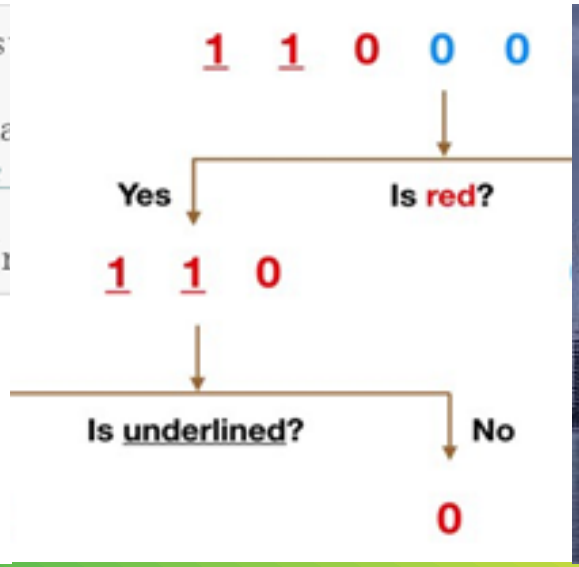
Random Forest Classifier

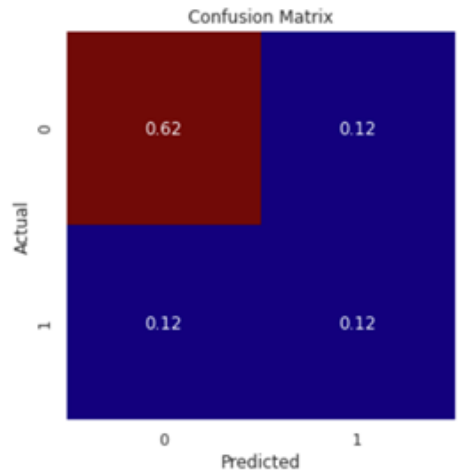



```
arn.ensemble import RandomFores

=RandomForestClassifier(n_estima
cation_model(rfc_model,all_data_
ts[0] += ["Random Forest"]
ts[1] += [classification_model(r
```

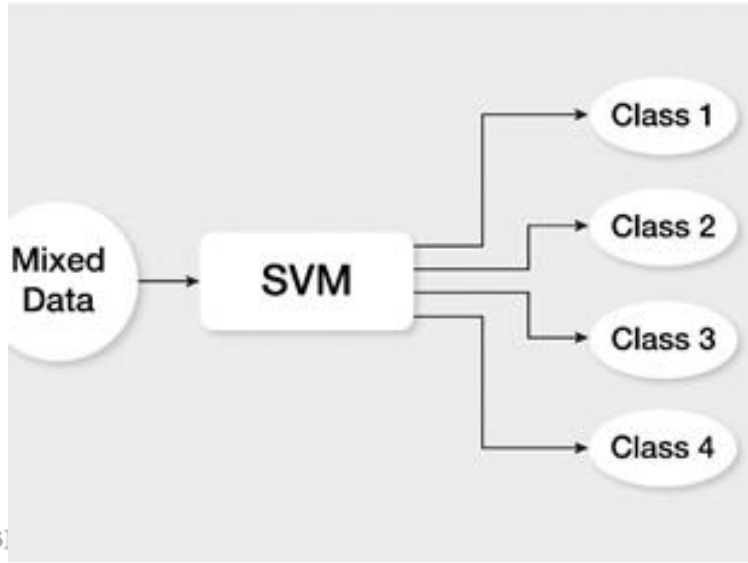
```
idation Score : 80.000%
idation Score : 81.667%
idation Score : 84.074%
idation Score : 81.237%
idation Score : 79.990%
```

```
he Accuracy Is : 79.990%
0.8, 0.8333333333333334, 0.8888888888888888, 0.72727272
```






```
The Accuracy Is : 79.990%
[0.8, 0.8333333333333334, 0.8888888888888888, 0.7272727272727273, 0.75]
```




```python
from sklearn.svm import SVC
svc = SVC(kernel = 'rbf', gamma = 'auto' )
svc_n = SVC(gamma = 'auto')
```

```python
svm_model = svm.SVC()
#classification_model(svm_model,all_data_train_X,
acc_results[0] += ["SVM"]
acc_results[1] += [classification_model(svm_model
```

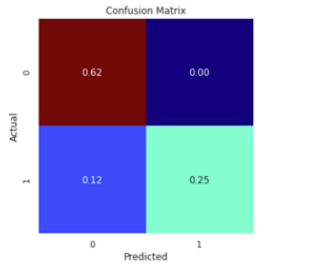
```
Cross-Validation Score : 80.000%
Cross-Validation Score : 81.667%
Cross-Validation Score : 84.074%
Cross-Validation Score : 81.237%
Cross-Validation Score : 79.990%
```

SVM

Logistic Regression

```python
from sklearn.linear_model import LogisticRegression
lr_model=LogisticRegression()
#classification_model(lr_model,all_data_train_X,data['diagnosis'])
acc_results[0] += ['Logistic Regression']
acc_results[1] += [classification_model(lr_model,all_data_train_X,data['diagnosis'])]
```

```
Cross-Validation Score : 80.000%
Cross-Validation Score : 77.500%
Cross-Validation Score : 81.296%
Cross-Validation Score : 79.154%
Cross-Validation Score : 80.823%
```

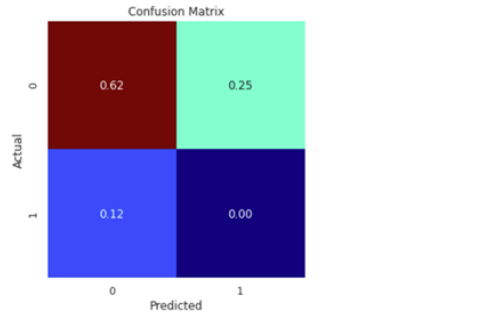



```
The Accuracy Is : 80.823%
[0.8, 0.75, 0.8888888888888888, 0.7272727272727273, 0.875]
```

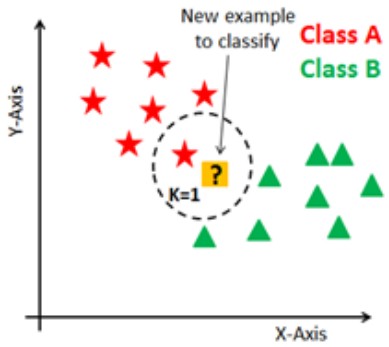
```
: from sklearn.neighbors import KNeighborsClassifier
kn_model = KNeighborsClassifier()
#classification_model(kn_model,all_data_train_X,data['diagnosis'])
acc_results[0] += ["K Neighbors"]
acc_results[1] += [classification_model(kn_model,all_data_train_X,data['diagnosis'])]
```

```
Cross-Validation Score : 70.000%
Cross-Validation Score : 76.667%
Cross-Validation Score : 80.741%
Cross-Validation Score : 76.465%
Cross-Validation Score : 73.672%
```

K-Neighbors




```
The Accuracy Is : 73.672%
[0.7, 0.8333333333333334, 0.8888888888888888, 0.6363636363636364, 0.625]
```

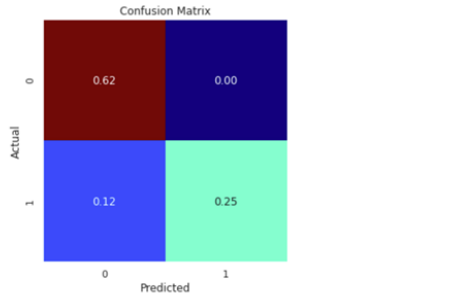


Naïve Bayesian

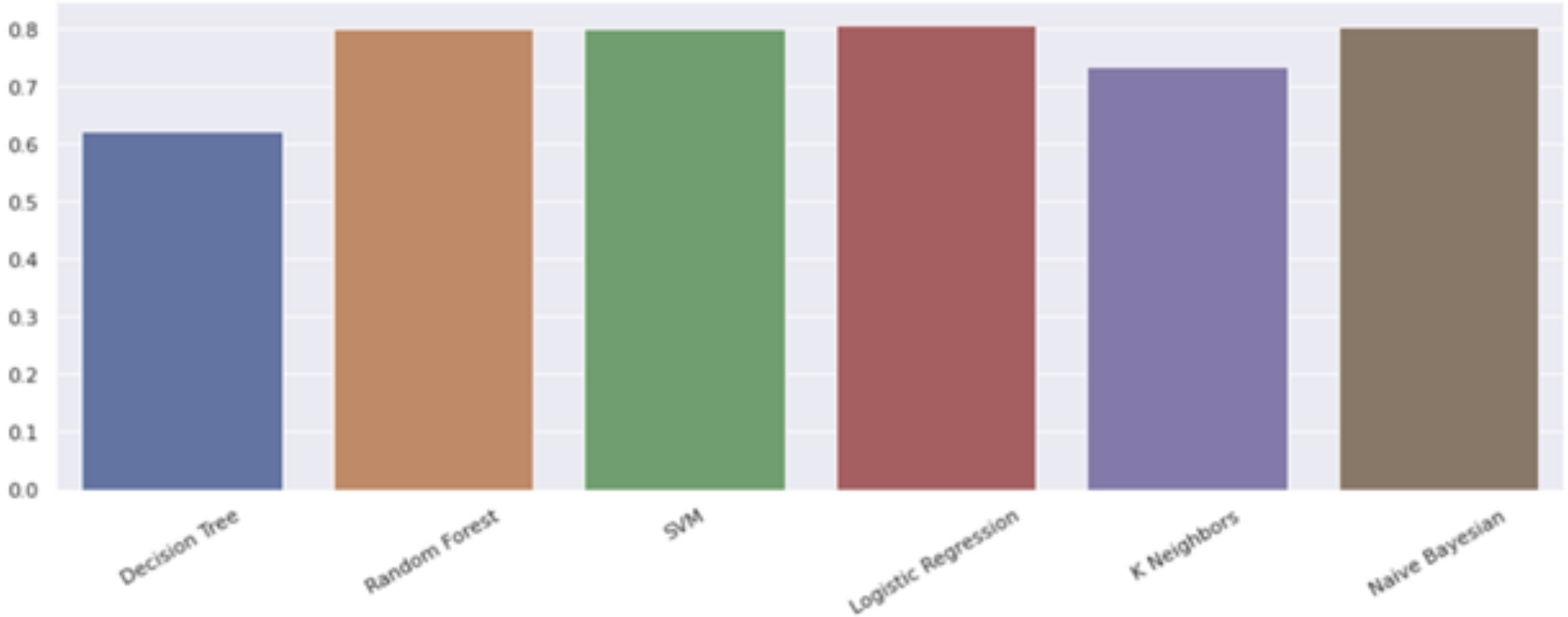
```
from sklearn.naive_bayes import GaussianNB
nb_model=GaussianNB()
#classification_model(nb_model,all_data_train_X,data['diagnosis'])
acc_results[0] += ["Naive Bayesian"]
acc_results[1] += [classification_model(nb_model,all_data_train_X,data['diagnosis'])]
```

```
Cross-Validation Score : 80.000%
Cross-Validation Score : 85.833%
Cross-Validation Score : 86.852%
Cross-Validation Score : 78.775%
Cross-Validation Score : 80.520%
```

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$




```
The Accuracy Is : 80.520%
[0.8, 0.9166666666666666, 0.8888888888888888, 0.5454545454545454, 0.875]
```



Accuracy Comparison

Conclusion

With the development of the times, huge technical and scientific made great advances, and the living standard of human beings has been improved enormously. However, woman's rights have not been promoted. Big mental pressure, big labor intensity and lack of feeling safety are easy to cause breast cancer's occurrence. Our project is going to help the expertise of the pathologist automatically detect and locate tumor tissue cells and to speed up the detecting process.

Deep learning is helpful to the automatic detection and localization of tumor cells, and accelerate the detection and localization process of tumor cells. In order to fully tap this potential, people can use a large number of tissue image data from different hospitals evaluated by different experts to construct pipelines. This can overcome the dependence on pathologists, which is especially useful in areas where there are no experts.

We can make some improvements in the future :

Because of the storage limitation of the kaggle cloud, we can't train more data sets. As we all know, the more training data sets, the more accurate the training results.

Because the classes of IDC versus no IDC are imbalanced, when training data, we need to shuffle and reorganize the data.

Before building the model, we should carefully analyze the data and select the appropriate training model according to the characteristics of the data.

Reference:

[1] https://www.cancer.org/cancer/breast-cancer.html
[2] https://www.breastcancer.org/symptoms/types/idc
[3] McLachlan, Geoffrey J.; Do, Kim-Anh; Ambroise, Christophe (2004). Analyzing microarray gene expression data. Wiley.
[4] Deng,H.; Runger, G.; Tuv, E. (2011). Bias of importance measures for multi-valued attributes and solutions. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN).
[5] https://en.wikipedia.org/wiki/Decision_tree#cite_note-7
[6] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Archived from the original (PDF) on 17 April 2016. Retrieved 5 June 2016.
[7] Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844. doi:10.1109/34.709601.
[8] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (PDF). Machine Learning. 20 (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478.

Shuyao Zhang Part

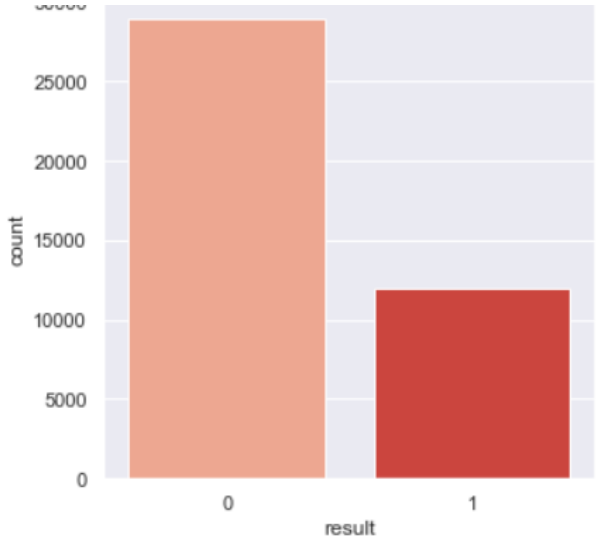
```
0]: def resize_image(key="train", plot=False):
    train_sequence = [transforms.Resize((50,50)),
                      transforms.RandomHorizontalFlip(),
                      transforms.RandomVerticalFlip()]
    val_sequence = [transforms.Resize((50,50))]
    if plot==False:
        train_sequence.extend([
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])])
        val_sequence.extend([
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])])

    data_transforms = {'train': transforms.Compose(train_sequence),'val': transforms.Compose(val_sequence)}
    return data_transforms[key]
```

Image Pre-Processing

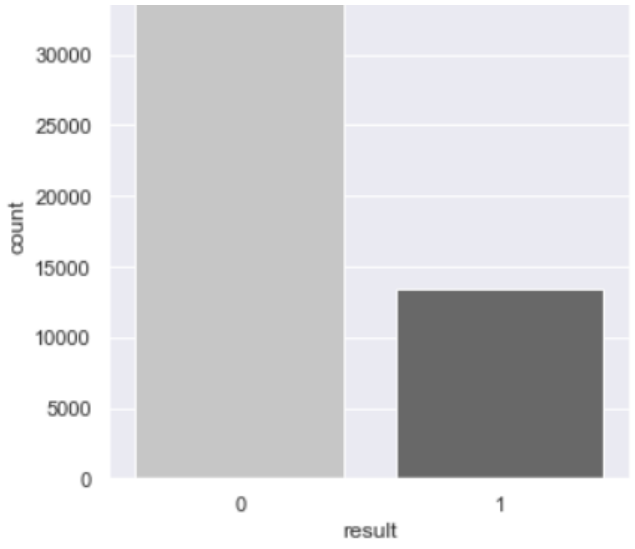
+ Instead of splitting data into two parts-30% test data and 70% training data, we categorized the data into 3 parts-15% test data, 15% validation data and 70% training data.

Data Distribution

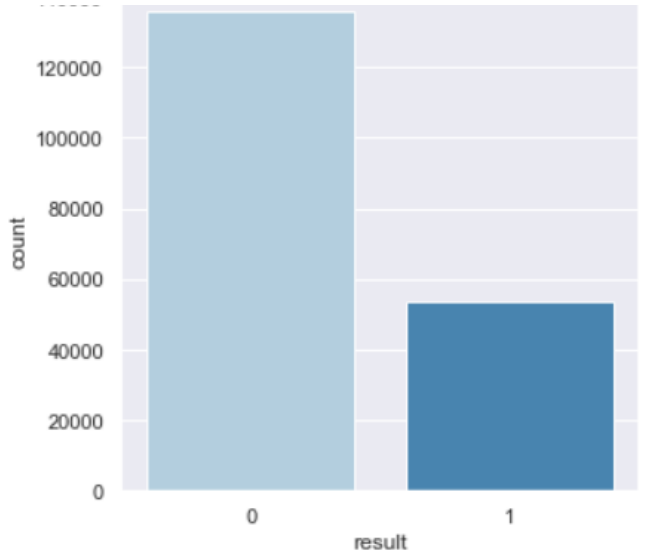
Dev



Train



Test



Accuracy of CNN

In [64]:
```
answer = test_predictions
length = answer.shape[0]
correct_predicted = 0

for i in range(length):
    if answer.true[i] == answer.predicted[i]:
        correct_predicted += 1

print("Accuracy of test set: ", correct_predicted/length)
```

Accuracy of test set: 0.83355636914531

In [65]:
```
answer = dev_predictions
length = answer.shape[0]
correct_predicted = 0

for i in range(length):
    if answer.true[i] == answer.predicted[i]:
        correct_predicted += 1

print("Accuracy of validation set: ", correct_predicted/length)
```

Accuracy of validation set: 0.8304516065830722

In [62]:
```
dev_predictions
```

Out[62]:

	true	predicted	proba	x	y	patient_id
0	0	0	0.003411	501.0	2001.0	9135
1	0	0	0.111403	1251.0	501.0	9135
2	0	0	0.234970	351.0	351.0	9135
3	0	0	0.599228	351.0	1051.0	9135
4	0	0	0.096663	1401.0	151.0	9135
...	...	...	...	...	...	...
40827	1	1	0.958453	2901.0	501.0	9255
40828	1	1	0.965054	1351.0	1101.0	9255
40829	1	1	0.845481	2751.0	1101.0	9255
40830	1	1	0.988100	2801.0	501.0	9255
40831	1	1	0.820634	2851.0	751.0	9255

40832 rows × 6 columns

In [63]:
```
test_predictions
```

Out[63]:

	true	predicted	proba	x	y	patient_id
0	0	0	0.128713	1201.0	2401.0	12906
1	0	0	0.347586	151.0	1001.0	12906
2	0	0	0.175564	1301.0	1251.0	12906
3	0	0	0.103502	1351.0	551.0	12906
4	0	0	0.224504	601.0	1551.0	12906
...	...	...	...	...	...	...
39600	1	1	0.680131	1651.0	701.0	14304
39601	1	1	0.814124	1201.0	851.0	14304
39602	1	0	0.551968	1201.0	901.0	14304
39603	1	1	0.894965	851.0	101.0	14304
39604	1	1	0.948164	1401.0	1151.0	14304

39605 rows × 6 columns

Conclusion

- CNN has shared convolution sum, which would increase the efficiency of dealing with high dimensional datasets.
- Also, we need to select characteristics manually to get the weight of different characteristic. This would make the classification of characteristics more precise.