Final Report

Course code: INT248

Advance Machine Learning

Bachelor of Computer Science Engineering

Submitted to

LOVELY PROFESSIONAL UNIVERSITY
PHAGWARA, PUNJAB

Submitted By:
Name of student: Chenna Krishna Reddy
Registration Number: 11914197
Roll Number: 60
Section: KM032

Submitted to:

Niharika Thakur

Title: Breast Cancer Classification Using Machine Learning Techniques

Abstract: - Automated breast cancer multi-classification from histopathological images plays a key role in computer-aided breast cancer diagnosis or prognosis. Breast cancer multi-classification is to identify subordinate classes of breast cancer (Ductal carcinoma, Fibroadenoma, Lobular carcinoma, etc.). However, breast cancer multi-classification from histopathological images faces two main challenges from: (1) the great difficulties in breast cancer multi-classification methods contrasting with the classification of binary classes (benign and malignant), and (2) the subtle differences in multiple classes due to the broad variability of high-resolution image appearances, high coherency of cancerous cells, and extensive inhomogeneity of color distribution. Therefore, automated breast cancer multi-classification from histopathological images is of great clinical significance yet has never been explored. Existing works in literature only focus on the binary classification but do not support further breast cancer quantitative assessment. In this study, we propose a breast cancer multi-classification method using a newly proposed deep learning model. The structured deep learning model has achieved remarkable performance (average 93.2% accuracy) on a large-scale dataset, which demonstrates the strength of our method in providing an efficient tool for breast cancer multi-classification in clinical settings.

Introduction: -

Automated breast cancer multi-classification from histopathological images is significant for clinical diagnosis and prognosis with the launch of the precision medicine initiative. According to the World Cancer Report from the World Health Organization (WHO), breast cancer is the most common cancer with high morbidity and mortality among women worldwide. Breast cancer patients account for 25.2%, which is ranked first place among women patients, and morbidity is 14.7%, which is ranked second place following lung cancer in the survey about cancer mortality in recent years. About half a million breast cancer patients are dead and nearly 1.7 million new cases arise per year. These statistics are expected to increase significantly. Furthermore, the histopathological image is a gold standard for identifying breast cancer compared with other medical imaging, e.g., mammography, magnetic resonance (MR), and computed tomography (CT). Noticeably, the decision of an optimal therapeutic schedule of breast cancer rests upon refined multi-classification. One main reason is that doctors who know the subordinate classes of breast cancer can control the metastasis of tumour cells early, and make substantial therapeutic schedules according to special clinical performance and prognosis result of multiple breast cancers.

Recently, many scholars have mentioned that the mortality rate has raised in women due to breast cancer. According to the World Health Organization (WHO), the number of females that died in 2018 is about 627,000. Also, this organization predicts that the number may reach to 2.7 million in 2030 globally. The late discovery of this disease and complex procedures are the main reasons for the low survival rate. Therefore, detection of breast cancer earlier is vital to decrease the risk of developing cancer in other tissue cells and carry out a proper treatment. Cancer is a creation of

abnormal cells that come from a modification in these cells genetically and spreads into the body, a late in diagnosis and treatment leads to death. There are two types of breast cancer, invasive and non-invasive. The former is harmful, malignant, ability to infect other organs, and classified as cancerous. The latter is non-invasive, not harmful, and not spread to other organs. This disease infects the women's chest and specifically glands and milk ducts, the spread of breast cancer to other organs is frequent and could be through the bloodstream [3]. Different techniques are used to capture breast cancer such as Ultrasound Sonography (ULS), Computerized Thermography (CT), Biopsy (Histological images), Magnetic-Resonance-Imaging (MRI), and Digital Mammography breast X-ray images (DMG). CT is a computerized x-ray imaging procedure that uses a narrow beam of x-ray focusing on a patient with rotation. This procedure produces signals that are dealing with computer to generate cross-sectional images. These images are called tomographic and include rich information from traditional x-ray

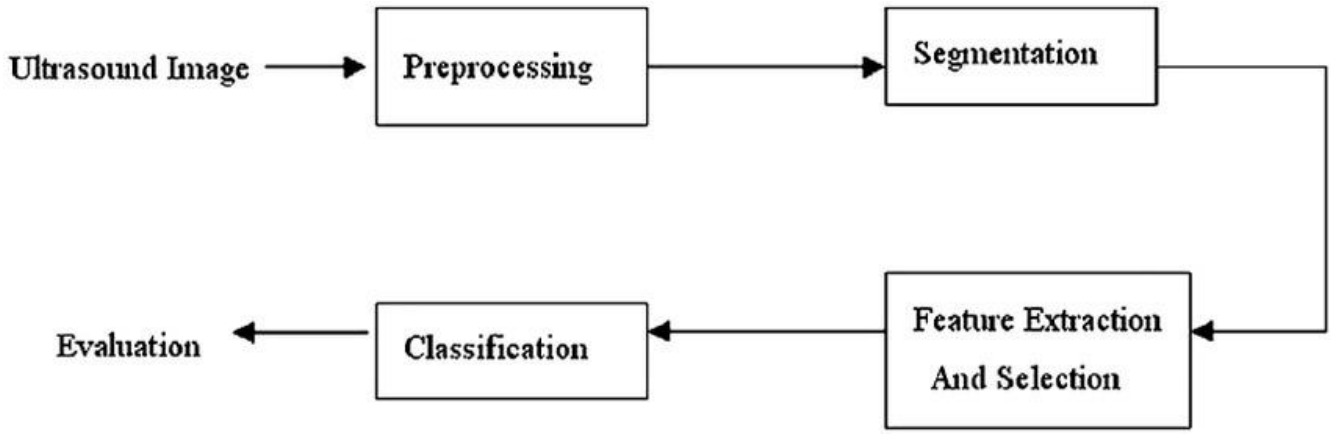
On Breast Cancer Detection: An Application of Machine Learning Algorithms on the Wisconsin Diagnostic Dataset by the Abien Fred M. Agarap. In this paper, six machine learning algorithms are used for detection of cancer. GRUSVM model is used for the diagnosis of breast cancer GRUSVM, Linear Regression, Multilayer Perceptron (MLP), Nearest Neighbor (NN) search, SoftMax Regression, and Support Vector Ma- chine (SVM) on the Wisconsin Diagnostic Breast Cancer (WDBC) dataset by measuring their classification test accuracy, and their sensitivity and specificity values. The said dataset consists of features which were computed from digitized images of FNA tests on a breast mass. For the implementation of the ML algorithms, the dataset was partitioned in the following fashion 70 percent for training phase, and 30 percent for the testing phase. Their results were that all presented ML algorithms exhibited high performance on the binary classification of carcinoma, i.e., determining whether benign tumor or malignant tumor. Therefore, the statistical measures on the classification problem were also satisfactory. To further corroborate the results of this study, a CV technique such as k-fold cross-validation should be used. The appliance of such a way won't only provide a more accurate measure of model prediction performance, but it'll also assist in determining the foremost optimal hyper-parameters for the ML algorithms.

Nevertheless, manual multi-classification for breast cancer histopathological images is a big challenge. There are three main reasons: (1) professional background and rich experience of pathologists are so difficult to inherit or innovate that primary-level hospitals and clinics suffer from the absence of skilled pathologists, (2) the tedious task is expensive and time-consuming, and (3) over fatigue of pathologists might lead to misdiagnosis. Hence, it is extremely urgent and important for the use of computer-aided breast cancer multi-classification, which can reduce the heavy workloads of pathologists and help avoid misdiagnosis.

However, automated breast cancer multi-classification still faces serious obstacles. The first obstacle is that the supervised feature engineering is inefficient and laborious with great computational burden. The initialization and processing steps of supervised feature engineering are also tedious and time-consuming. Meaningful and representative features lie at the heart of its success to multi-classify breast cancer. Nevertheless, feature engineering is an independent domain, task-related features are mostly designed by medical specialists who use their knowledge for histopathological image processing. E.g., Zhang *et al*. applied a one class kernel principal

component analysis (PCA) method based on hand-crafted features to classify benign and malignant of breast cancer histopathological images, the accuracy reached 92%. Recent years, general feature descriptors used for feature extraction have been invented, e.g., scale-invariant feature transform (SIFT), Gray-level co-occurrence matrix (GLCM), histogram of oriented gradient (HOG), etc. However, feature descriptors extract merely insufficient features for describing histopathological images, such as low-level and unrepresentative surface features, which are not suitable for classifiers with discriminant analysis ability. There are several applications that use general feature descriptors on binary classification for histopathological images of breast cancer. Pinhole *al*. used a breast cancer histopathological images dataset (Break His), then provided a baseline of binary classification recognition rates by means of different feature descriptors and different traditional machine learning classifiers, the range of the accuracy is 80% to 85%. Based on four shape and 138 textual feature descriptors, Wang *et al*. realized accurate binary classification using a support vector machine (SVM) classifier. The second obstacle is that breast cancer histopathological images have huge limitations. Eight classes histopathological images of breast cancer are presented. These are fine-grained high-resolution images from breast tissue biopsy slides stained with haematoxylin and eosin (H&E). Noticeably, different classes have subtle differences and cancerous cells have high coherency. The differences of same class images' resolution, contrast, and appearances are always in greater compared to different classes. In addition, histopathological fine-grained images have large variations which always result in difficulties for distinguishing breast cancers. Finally, despite such effective performance in the medical imaging analysis domain by deep learning, existing related methods only studied on binary classification for breast cancer; however, multi-classification has more clinical values.

Fig 1: A general structure of CAD system for breast cancer diagnosis.



CODE:

Importing the Dependencies

In [1]:
```
import numpy as np
import pandas as pd
import sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection & Processing

In [2]:
```
# loading the data from sklearn
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()
```

In [3]:
```
print(breast_cancer_dataset)
```

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
```

```
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]), 'frame': N
one, 'target_names': array(['malignant', 'benign'], dtype='
```

In [4]:
```
# loading the data to a data frame
data_frame = pd.DataFrame(breast_cancer_dataset.data, columns = breast_cance
```

In [5]:
```
# print the first 5 rows of the dataframe
data_frame.head()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 30 columns

In [6]:
```
# adding the 'target' column to the data frame
data_frame['label'] = breast_cancer_dataset.target
```

In [7]:
```
# print last 5 rows of the dataframe
data_frame.tail()
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	

5 rows × 31 columns

In [8]:
```
# number of rows and columns in the dataset
data_frame.shape
```

Out[8]: (569, 31)

In [9]:
```
# getting some information about the data
data_frame.info()
```

```
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  label                    569 non-null    int32  
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

In [10]:
```
# statistical measures about the data
data_frame.describe()
```

Out[10]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	n conca
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.08
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.00
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.02

50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.06
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.13
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.42

8 rows × 31 columns

In [11]:
```
# checking the distribution of Target Varibale
data_frame['label'].value_counts()
```

Out[11]:
```
1    357
0    212
Name: label, dtype: int64
```

1 --> Benign

0 --> Malignant

In [12]:
```
data_frame.groupby('label').mean()
```

Out[12]:

label	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity
0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775
1	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058

2 rows × 30 columns

Separating the features and target

In [13]:
```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']
```

In [14]:
```
print(X)
```

```
     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
..           ...           ...             ...        ...              ...
564        21.56         22.39          142.00     1479.0          0.11100
```

```
565        20.13         28.25          131.20      1261.0          0.09780
566        16.60         28.08          108.30       858.1          0.08455
567        20.60         29.33          140.10      1265.0          0.11780
568         7.76         24.54           47.92       181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
0             0.27760         0.30010              0.14710         0.2419
1             0.07864         0.08690              0.07017         0.1812
2             0.15990         0.19740              0.12790         0.2069
3             0.28390         0.24140              0.10520         0.2597
4             0.13280         0.19800              0.10430         0.1809
..                ...             ...                  ...            ...
564           0.11590         0.24390              0.13890         0.1726
565           0.10340         0.14400              0.09791         0.1752
566           0.10230         0.09251              0.05302         0.1590
567           0.27700         0.35140              0.15200         0.2397
568           0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  ...  worst radius  worst texture  \
0                   0.07871  ...        25.380          17.33
1                   0.05667  ...        24.990          23.41
2                   0.05999  ...        23.570          25.53
3                   0.09744  ...        14.910          26.50
4                   0.05883  ...        22.540          16.67
..                      ...  ...           ...            ...
564                 0.05623  ...        25.450          26.40
565                 0.05533  ...        23.690          38.25
566                 0.05648  ...        18.980          34.12
567                 0.07016  ...        25.740          39.42
568                 0.05884  ...         9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness  \
0             184.60      2019.0           0.16220            0.66560
1             158.80      1956.0           0.12380            0.18660
2             152.50      1709.0           0.14440            0.42450
3              98.87       567.7           0.20980            0.86630
4             152.20      1575.0           0.13740            0.20500
..               ...         ...               ...                ...
564           166.10      2027.0           0.14100            0.21130
565           155.00      1731.0           0.11660            0.19220
566           126.70      1124.0           0.11390            0.30940
567           184.60      1821.0           0.16500            0.86810
568            59.16       268.6           0.08996            0.06444

     worst concavity  worst concave points  worst symmetry  \
0             0.7119                0.2654          0.4601
1             0.2416                0.1860          0.2750
2             0.4504                0.2430          0.3613
3             0.6869                0.2575          0.6638
4             0.4000                0.1625          0.2364
..               ...                   ...             ...
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
568           0.0000                0.0000          0.2871

     worst fractal dimension
0                    0.11890
1                    0.08902
```

```
2                   0.08758
3                   0.17300
4                   0.07678
..                      ...
564                 0.07115
565                 0.06637
566                 0.07820
567                 0.12400
568                 0.07039

[569 rows x 30 columns]
```

In [15]:
```
print(Y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: label, Length: 569, dtype: int32
```

Splitting the data into training data & Testing data

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, rar
```

In [17]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

Model Training

Logistic Regression

In [18]:
```
model = LogisticRegression()
```

In [19]:
```
# training the Logistic Regression model using Training data
model.fit(X_train, Y_train)
```

```
C:\Users\sathy\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

Out[19]:
```
LogisticRegression()
```

Model Evaluation

Accuracy Score

In [20]:
```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

In [21]:
```
print('Accuracy on training data = ', training_data_accuracy)
```

```
Accuracy on training data =  0.9384615384615385
```

In [22]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

In [23]:
```
print('Accuracy on test data = ', test_data_accuracy)
```

```
Accuracy on test data =  0.9298245614035088
```

Building a Predictive System

In [24]:
```
input_data = (13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885

# change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array as we are predicting for one datapoint
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
  print('The Breast cancer is Malignant')

else:
  print('The Breast Cancer is Benign')
```

Conclusion:

Breast cancer if found at an early stage will help save lives of thousands of women or even men. These projects help the real-world patients and doctors to gather as much information as they can. The research on nine papers has helped us gather the data for the project proposed by us. By using machine learning algorithms, we will be able to classify and predict the cancer into being or malignant. Machine learning algorithms can be used for medical oriented research, it advances the system, reduces human errors and lowers manual mistakes. The primary purpose of this study is to create and execute a novel computation for interpreting and orchestrating chest disease data obtained from mammography and pathology results of patient scans that is obtained and UCI repositories' cloud. Python programming, a convolutional neural association model is utilized to accomplish this, and the results are confirmed. According to the findings, the suggested CNN outperforms estimations in recognizing and requesting breast cancer for image datasets. And SVM has proven to outperforms CART, NB and KNN in analysis and prediction of cancer with numerical dataset.

GitHub Link:

https://github.com/bckr2549/Breast-Cancer-Prediction