# 15SE376L – MINOR PROJECT II

## SEMESTER- VII

## ACADEMIC YEAR: 2019-20

**NAME: Paras Sibal**
**REGISTER NUMBER: RA1611020010055**
**NAME: Ankit Sahu**
**REGISTER NUMBER: RA1611020010095**

**PROGRAM: B. Tech – SWE**
**SUBMITTED TO: Dr. A. ALICE NITHYA**

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
*Deemed to be* **University** *u/s 3 of UGC Act, 1956*

**DEPARTMENT OF SOFTWARE ENGINEERING**
**Faculty of Engineering and Technology**

**SRM IST**
(Under section 3 of the UGC act, 1956)

**s**

# BONAFIDE CERTIFICATE

This is to certify that Minor Project titled “Brest Cancer Prediction system” is the bonafide of the student Paras Sibal (RA1611020010055) and Ankit Sahu (RA1611020010095) who carried out the project work under my supervision. Certified further, that to the best of my knowledge of the work reported here in does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion of this or any other candidate.

| | |
|---|---|
| **Signature of faculty** | **Signature of HOD** |
| **Dr. A. Alice Nithiya** | **Dr. C. Lakshmi** |
| **Ass. Professor** | **Professor and Head** |
| **Department of SWE** | **Department of SWE** |
| **SRMIST** | **SRMIST** |

## ABSTRACT

In this project, we will develop a system that can classify “Breast Cancer Disease” tumour using neural network with Back Propagation Algorithm to classify the tumour based on symptoms. The main aim of research is to develop more cost-effective and easy–to-use systems. The data is taken from the University of Wisconsin breast cancer dataset with 9 attributes and 699 records.

**Domain:** Machine Learning

## INTRODUCTION

Breast cancer is the second leading cause of cancer deaths in women worldwide and occurs in nearly one out of eight women. Currently there are three techniques to diagnose breast cancer: mammography, FNA (Fine Needle Aspirate) and surgical biopsy. The intention of this study is to design a prediction system that can predict the incidence of the breast cancer at early stage by analyzing smallest set of attributes and performing data cleaning.

## DOMAIN STUDY

Breast cancer is cancer that develops from breast tissue. Signs of breast cancer may include a lump in the breast, a change in breast shape, dimpling of the skin, fluid coming from the nipple, a newly-inverted nipple, or a red or scaly patch of skin. In those with distant spread of the disease, there may be bone pain, swollen lymph nodes, shortness of breath, or yellow skin.

Risk factors for developing breast cancer include being female, obesity, lack of physical exercise, drinking alcohol, hormone replacement therapy during menopause, ionizing radiation, early age at first menstruation, having children late or not at all, older age, prior history of breast cancer, and family history. About 5–10% of cases are due to genes inherited from a person's parents, including BRCA1 and BRCA2 among others. Breast cancer most commonly develops in cells from the lining of milk ducts and the lobules that supply the ducts with milk. Cancers developing from the ducts are

known as ductal carcinomas, while those developing from lobules are known as lobular carcinomas. In addition, there are more than 18 other sub-types of breast cancer. Some cancers, such as ductal carcinoma in situ, develop from pre-invasive lesions. The diagnosis of breast cancer is confirmed by taking a biopsy of the concerning lump. Once the diagnosis is made, further tests are done to determine if the cancer has spread beyond the breast and which treatments are most likely to be effective.

## REQUIREMENTS GATHERING

We have used various techniques of requirement gathering including:

1. Brainstorming
2. Focus Groups

## BRAINSTORMING

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members. In other words, brainstorming is a situation where a group of people meet to generate new ideas and solutions around a specific domain of interest by removing inhibitions. People are able to think more freely and they suggest as many spontaneous new ideas as possible. All the ideas are noted down without criticism and after the brainstorming session the ideas are evaluated. We can use brainstorming to find out what our system will do and what all functionalities be added that the user will like.

## FOCUS GROUP

A focus group is a gathering of deliberately selected people who participate in a planned discussion intended to elicit consumer perceptions about a particular topic or area of interest in an environment that is non-threatening and receptive. Focus groups are a collective on purpose. Unlike interviews, which usually occurs with an individual, the focus groups allow members of a group to interact and influence each other during the discussion and consideration of ideas and perspectives.

## FUNCTIONAL REQUIREMENTS OF BREAST CANCER PREDICTION SYSTEMS

Breast cancer prediction systems have a wide range of applications to help series of patients with cancer symptoms
The applications include:

1. Check tumour
2. Classify the tumour
3. Determine correct classification rate
4. Investigate the potential of neural network as outcome classifier
5. Determine hidden layer that corresponds to maximum accuracy

## NON FUNCTIONAL REQUIREMENTS OF BREAST CANCER PREDICTION SYSTEMS

The non functional requirements of a breast cancer prediction system include:

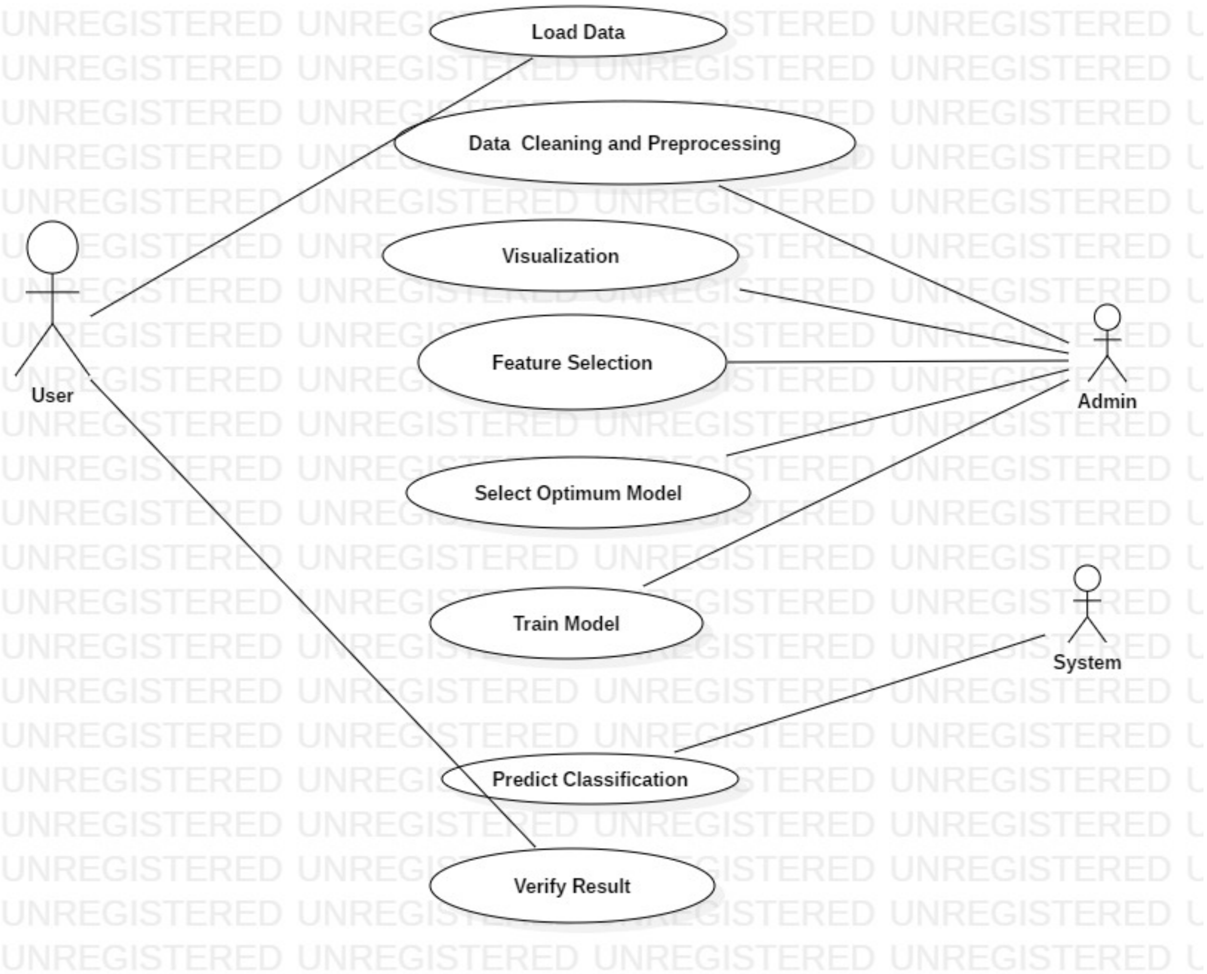
1. Performance
2. Reliability
3. Usability
4. Scalability

## LITERATURE REVIEW

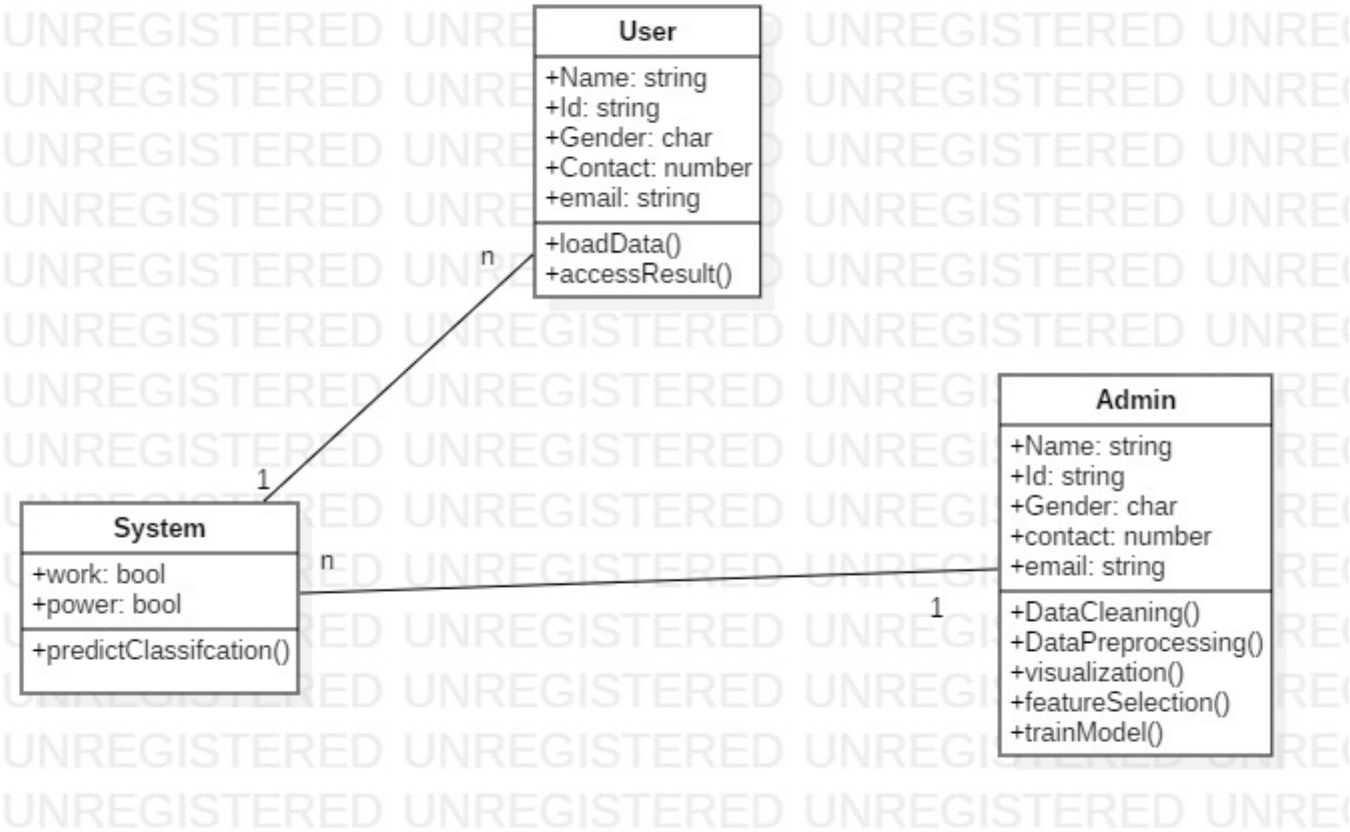
In [1], it notes that different classifiers have been used to conduct experiments on the standard WBCD. It is being observed KNN classifier yields the highest classification accuracies when used with most predictive variables. In [2], proposed a framework for breast cancer prognosis. The framework is named as extensible Breast Cancer Prognosis Framework (XBPF). As the prognosis refers to prediction of breast cancer susceptibility, survivability and recurrence, it is not advisable for our paper. In [3], Pre-classification method was not accurate in determining the records of "not survived" class as the cause of death and survivability rate were not taken into consideration. And Bellaachia took the study and proved the change in accuracy. In [4], Gradient Descent (GD) back propagation and Liebenberg Marquardt (LM) back propagation are applied to train MLP neural network. To evaluate the performance of neural network approach, accuracy, sensitivity and specificity of outputs are calculated. However, specificity obtained in this study cannot be accurate because the number of benign cases in the database was not relatively high. Specificity comparatively low as compared to accuracy and sensitivity. In [5], The analysis of the results signifies that the integration of multidimensional data along with different classification, feature selection and dimensionality reduction techniques can provide auspicious tools for inference in this domain.

# ARCHITECTURAL VIEWS

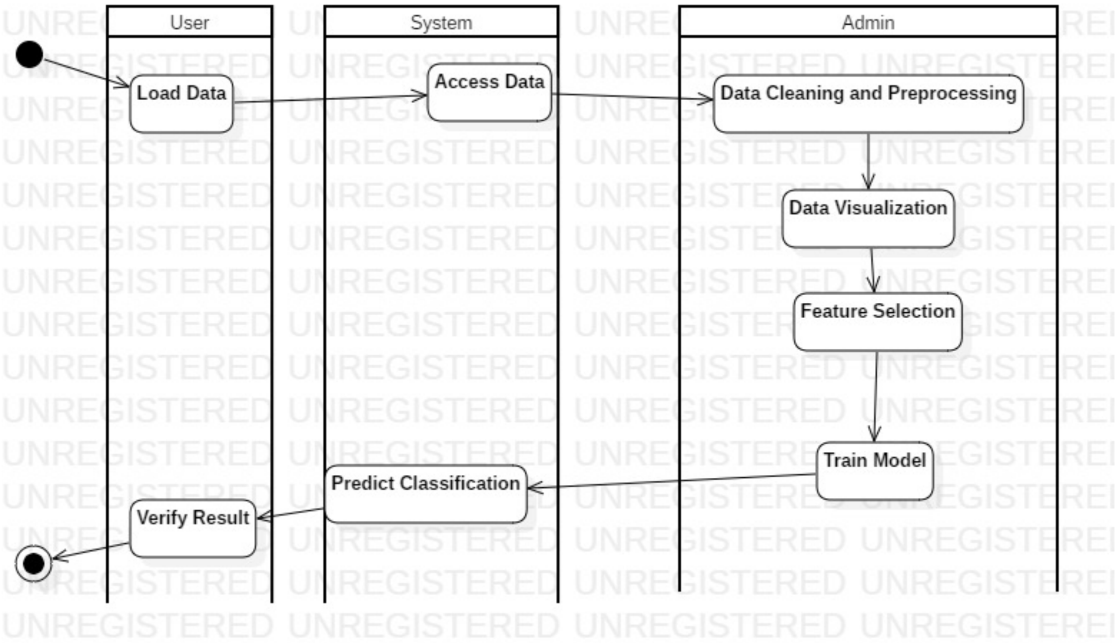
## USE CASE DIAGRAM



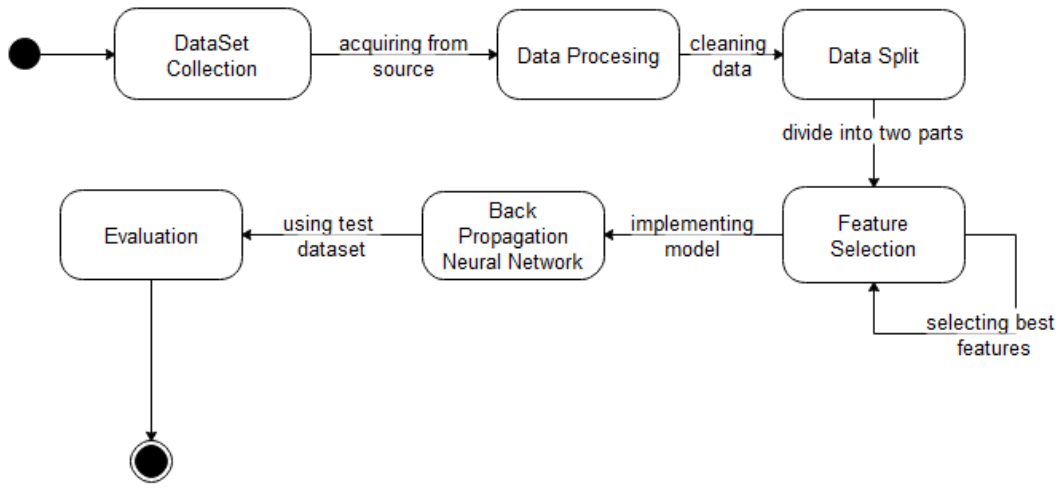
## CLASS DIAGRAM



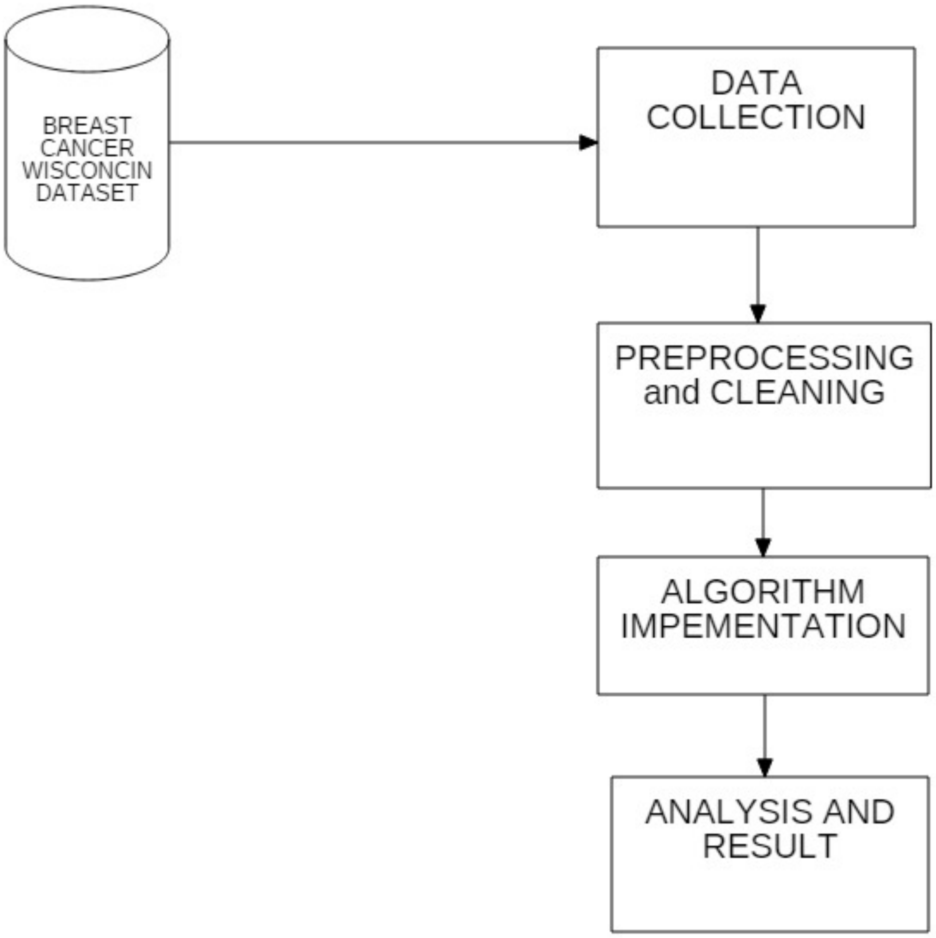
## ACTIVITY DIAGRAM



## STATE CHART DIAGRAM



## FLOWCHART



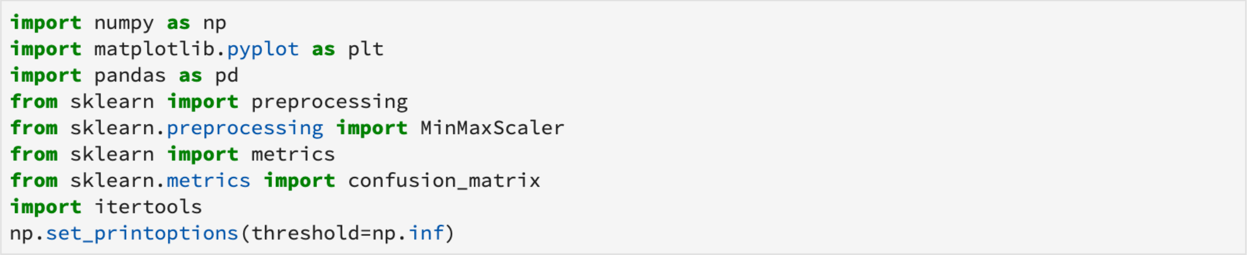
## RESULT AND DISCUSSIONS

The function learnt pretty well to adapt to both the training and validation sets. We can see even more clearly that our validation set has perfect accuracy on its 183 samples. We have set the learning rate to 0.07 and train for 65000 iterations, we obtain cost after iteration 64500: 0.16442 with accuracy of 99.45%

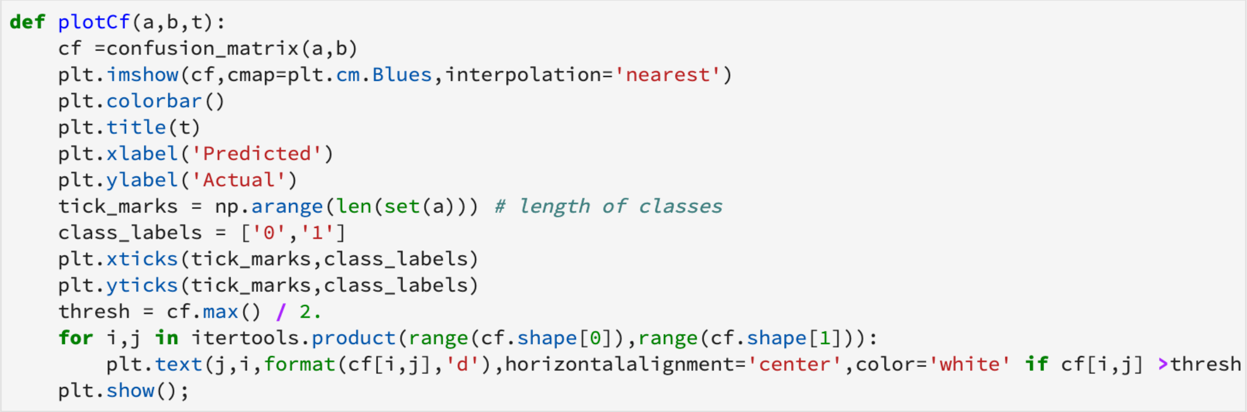
## FUTURE SCOPE

In the future, with more number of data entries, the model will tend to have better performance in deducing the cancer prediction and classification rate. More number of hidden layer neurons will increase the system performance. Due to less model cost and further enhancements, there can be a shift from traditional prediction system to a reliable and more efficient one.

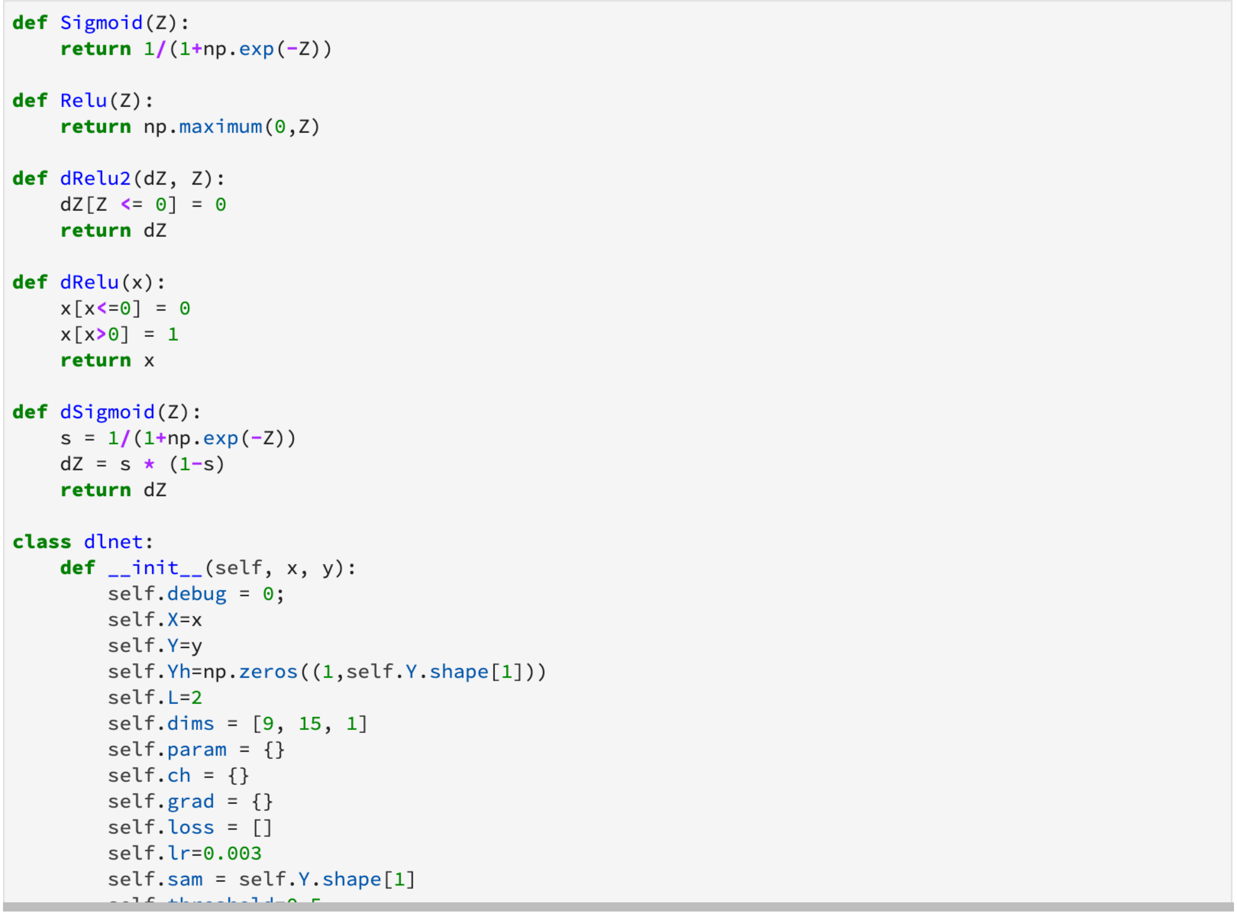
## SAMPLE SCREENSHOT



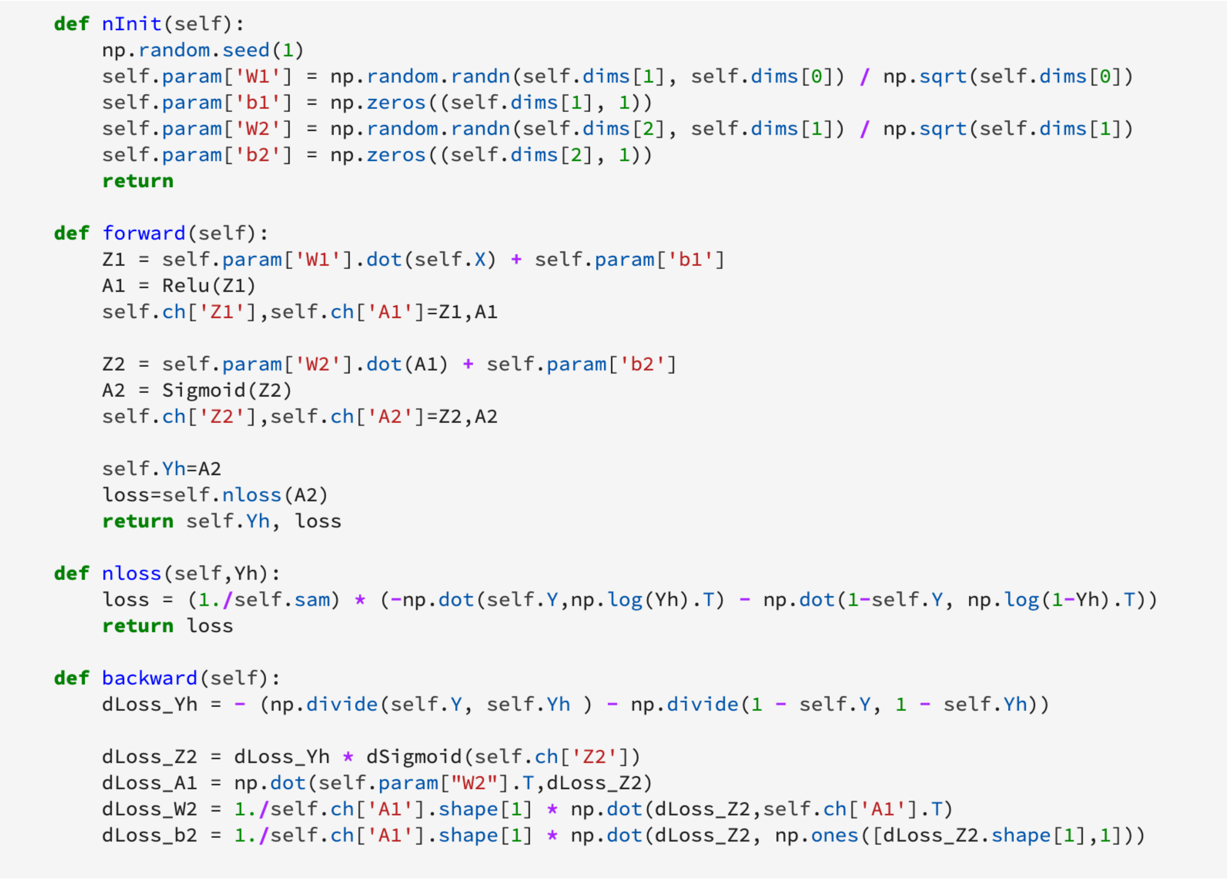
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.metrics import confusion_matrix
import itertools
np.set_printoptions(threshold=np.inf)
```



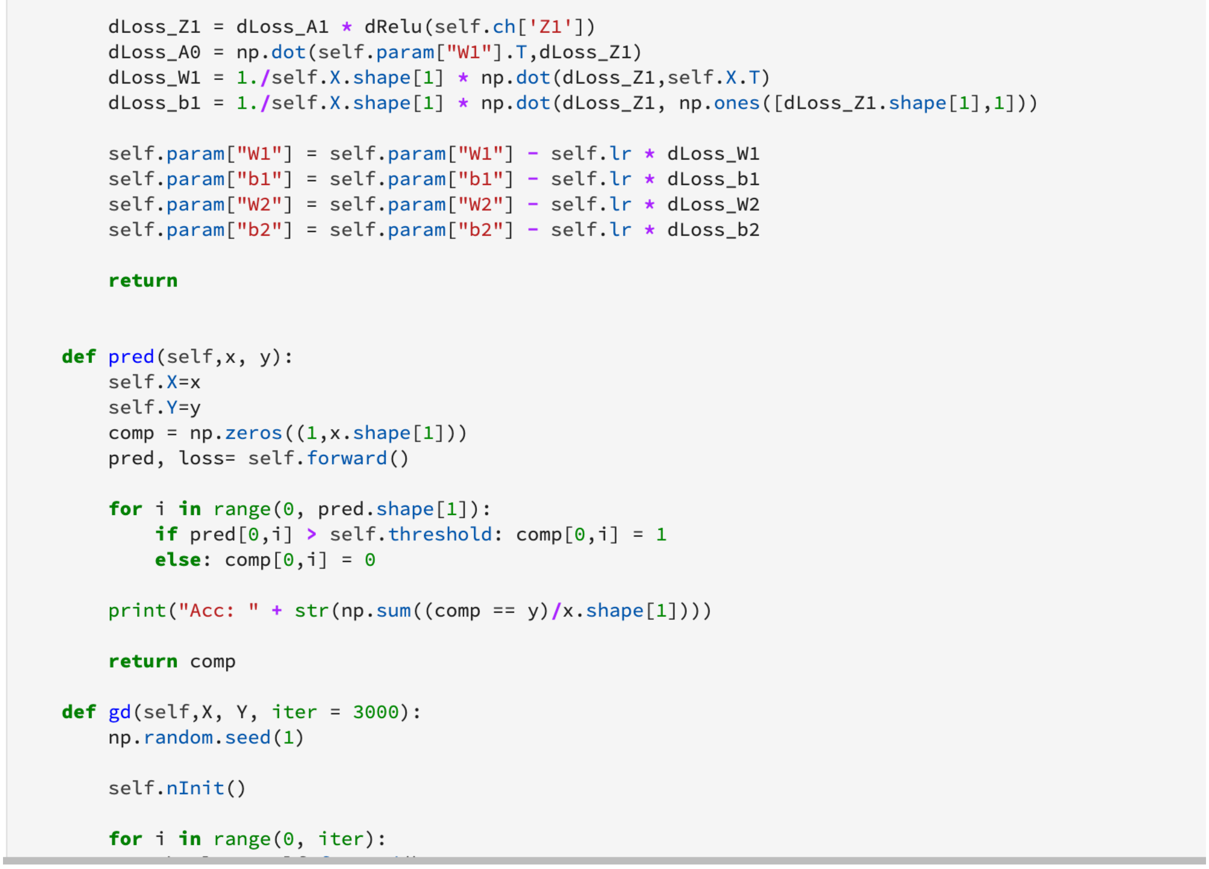
```
def plotCf(a,b,t):
    cf =confusion_matrix(a,b)
    plt.imshow(cf,cmap=plt.cm.Blues,interpolation='nearest')
    plt.colorbar()
    plt.title(t)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    tick_marks = np.arange(len(set(a))) # length of classes
    class_labels = ['0','1']
    plt.xticks(tick_marks,class_labels)
    plt.yticks(tick_marks,class_labels)
    thresh = cf.max() / 2.
    for i,j in itertools.product(range(cf.shape[0]),range(cf.shape[1])):
        plt.text(j,i,format(cf[i,j],'d'),horizontalalignment='center',color='white' if cf[i,j] >thresh
    plt.show();
```



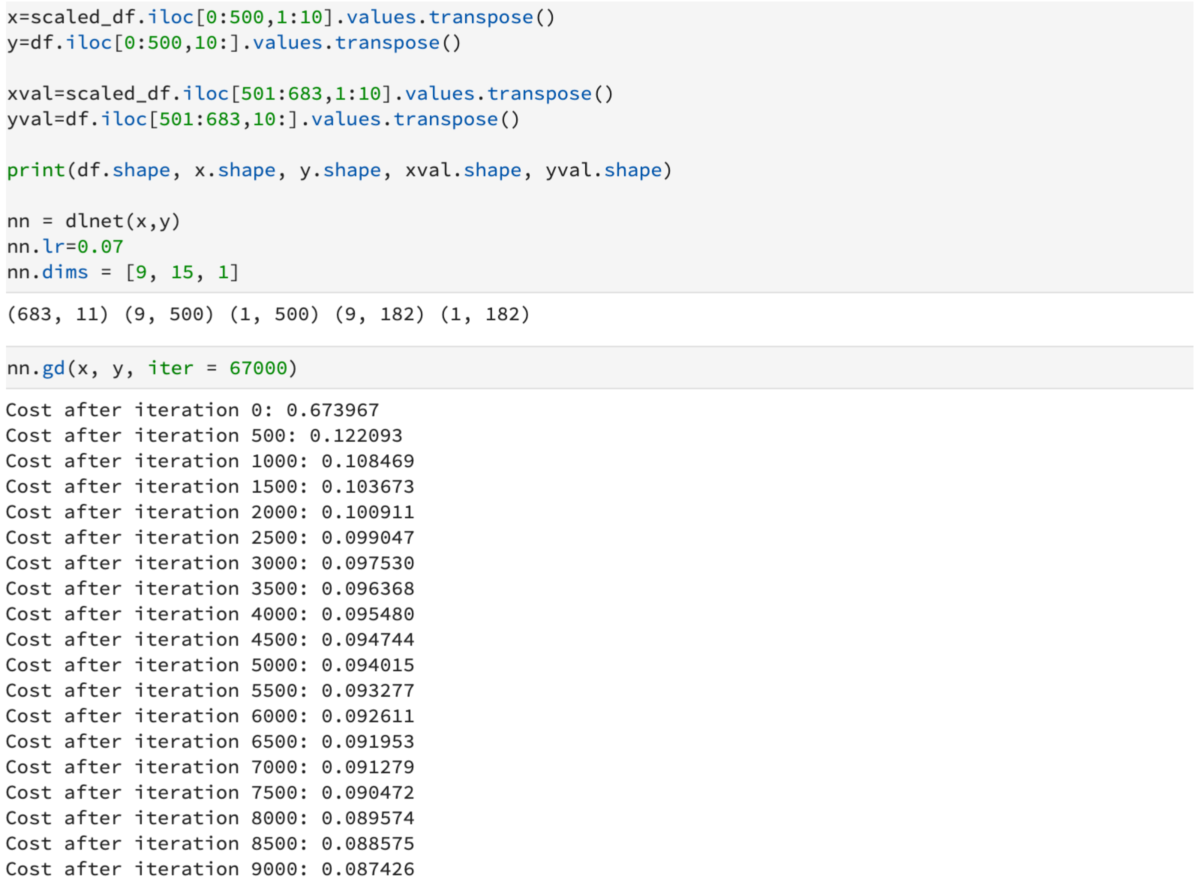
```
def Sigmoid(Z):
    return 1/(1+np.exp(-Z))

def Relu(Z):
    return np.maximum(0,Z)

def dRelu2(dZ, Z):
    dZ[Z <= 0] = 0
    return dZ

def dRelu(x):
    x[x<=0] = 0
    x[x>0] = 1
    return x

def dSigmoid(Z):
    s = 1/(1+np.exp(-Z))
    dZ = s * (1-s)
    return dZ

class dlnet:
    def __init__(self, x, y):
        self.debug = 0;
        self.X=x
        self.Y=y
        self.Yh=np.zeros((1,self.Y.shape[1]))
        self.L=2
        self.dims = [9, 15, 1]
        self.param = {}
        self.ch = {}
        self.grad = {}
        self.loss = []
        self.lr=0.003
        self.sam = self.Y.shape[1]
        self.threshold=0.5
```



```
    def nInit(self):
        np.random.seed(1)
        self.param['W1'] = np.random.randn(self.dims[1], self.dims[0]) / np.sqrt(self.dims[0])
        self.param['b1'] = np.zeros((self.dims[1], 1))
        self.param['W2'] = np.random.randn(self.dims[2], self.dims[1]) / np.sqrt(self.dims[1])
        self.param['b2'] = np.zeros((self.dims[2], 1))
        return

    def forward(self):
        Z1 = self.param['W1'].dot(self.X) + self.param['b1']
        A1 = Relu(Z1)
        self.ch['Z1'],self.ch['A1']=Z1,A1

        Z2 = self.param['W2'].dot(A1) + self.param['b2']
        A2 = Sigmoid(Z2)
        self.ch['Z2'],self.ch['A2']=Z2,A2

        self.Yh=A2
        loss=self.nloss(A2)
        return self.Yh, loss

    def nloss(self,Yh):
        loss = (1./self.sam) * (-np.dot(self.Y,np.log(Yh).T) - np.dot(1-self.Y, np.log(1-Yh).T))
        return loss

    def backward(self):
        dLoss_Yh = - (np.divide(self.Y, self.Yh ) - np.divide(1 - self.Y, 1 - self.Yh))

        dLoss_Z2 = dLoss_Yh * dSigmoid(self.ch['Z2'])
        dLoss_A1 = np.dot(self.param["W2"].T,dLoss_Z2)
        dLoss_W2 = 1./self.ch['A1'].shape[1] * np.dot(dLoss_Z2,self.ch['A1'].T)
        dLoss_b2 = 1./self.ch['A1'].shape[1] * np.dot(dLoss_Z2, np.ones([dLoss_Z2.shape[1],1]))
```



```
        dLoss_Z1 = dLoss_A1 * dRelu(self.ch['Z1'])
        dLoss_A0 = np.dot(self.param["W1"].T,dLoss_Z1)
        dLoss_W1 = 1./self.X.shape[1] * np.dot(dLoss_Z1,self.X.T)
        dLoss_b1 = 1./self.X.shape[1] * np.dot(dLoss_Z1, np.ones([dLoss_Z1.shape[1],1]))

        self.param["W1"] = self.param["W1"] - self.lr * dLoss_W1
        self.param["b1"] = self.param["b1"] - self.lr * dLoss_b1
        self.param["W2"] = self.param["W2"] - self.lr * dLoss_W2
        self.param["b2"] = self.param["b2"] - self.lr * dLoss_b2

        return


    def pred(self,x, y):
        self.X=x
        self.Y=y
        comp = np.zeros((1,x.shape[1]))
        pred, loss= self.forward()

        for i in range(0, pred.shape[1]):
            if pred[0,i] > self.threshold: comp[0,i] = 1
            else: comp[0,i] = 0

        print("Acc: " + str(np.sum((comp == y)/x.shape[1])))

        return comp

    def gd(self,X, Y, iter = 3000):
        np.random.seed(1)

        self.nInit()

        for i in range(0, iter):
```

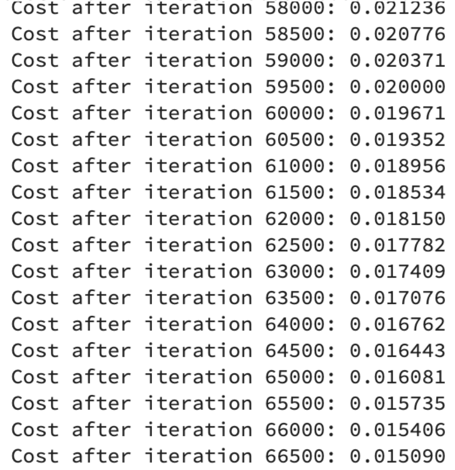


```
x=scaled_df.iloc[0:500,1:10].values.transpose()
y=df.iloc[0:500,10:].values.transpose()

xval=scaled_df.iloc[501:683,1:10].values.transpose()
yval=df.iloc[501:683,10:].values.transpose()

print(df.shape, x.shape, y.shape, xval.shape, yval.shape)

nn = dlnet(x,y)
nn.lr=0.07
nn.dims = [9, 15, 1]
```

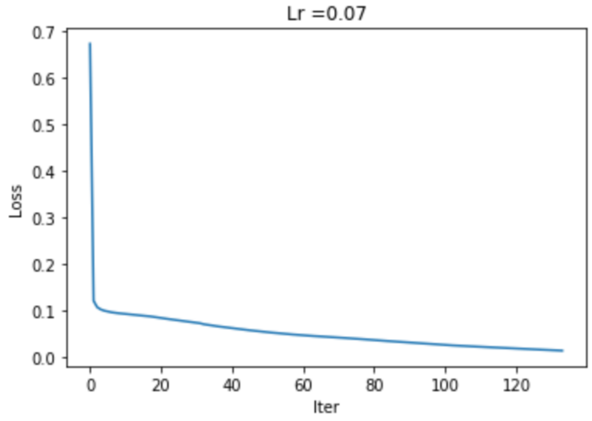
```
(683, 11) (9, 500) (1, 500) (9, 182) (1, 182)
```

```
nn.gd(x, y, iter = 67000)
```

```
Cost after iteration 0: 0.673967
Cost after iteration 500: 0.122093
Cost after iteration 1000: 0.108469
Cost after iteration 1500: 0.103673
Cost after iteration 2000: 0.100911
Cost after iteration 2500: 0.099047
Cost after iteration 3000: 0.097530
Cost after iteration 3500: 0.096368
Cost after iteration 4000: 0.095480
Cost after iteration 4500: 0.094744
Cost after iteration 5000: 0.094015
Cost after iteration 5500: 0.093277
Cost after iteration 6000: 0.092611
Cost after iteration 6500: 0.091953
Cost after iteration 7000: 0.091279
Cost after iteration 7500: 0.090472
Cost after iteration 8000: 0.089574
Cost after iteration 8500: 0.088575
Cost after iteration 9000: 0.087426
```

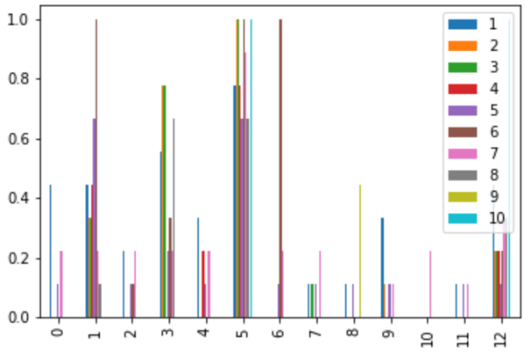


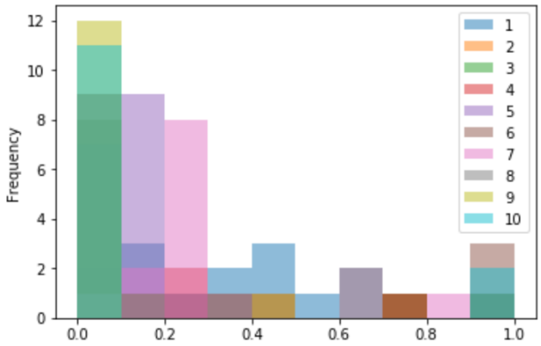
```
Cost after iteration 58000: 0.021236
Cost after iteration 58500: 0.020776
Cost after iteration 59000: 0.020371
Cost after iteration 59500: 0.020000
Cost after iteration 60000: 0.019671
Cost after iteration 60500: 0.019352
Cost after iteration 61000: 0.018956
Cost after iteration 61500: 0.018534
Cost after iteration 62000: 0.018150
Cost after iteration 62500: 0.017782
Cost after iteration 63000: 0.017409
Cost after iteration 63500: 0.017076
Cost after iteration 64000: 0.016762
Cost after iteration 64500: 0.016443
Cost after iteration 65000: 0.016081
Cost after iteration 65500: 0.015735
Cost after iteration 66000: 0.015406
Cost after iteration 66500: 0.015090
```



```
scaled_df[10]= df[10]
scaled_df.iloc[0:13,1:11].plot.bar();
scaled_df.iloc[0:13,1:11].plot.hist(alpha=0.5)
```

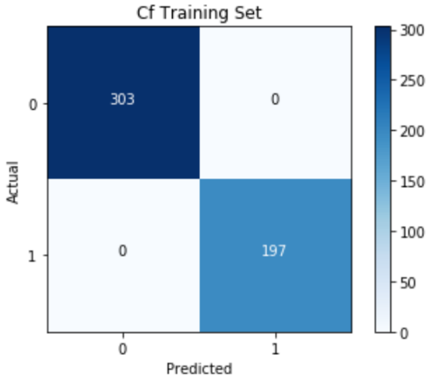
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a19b7e198>
```



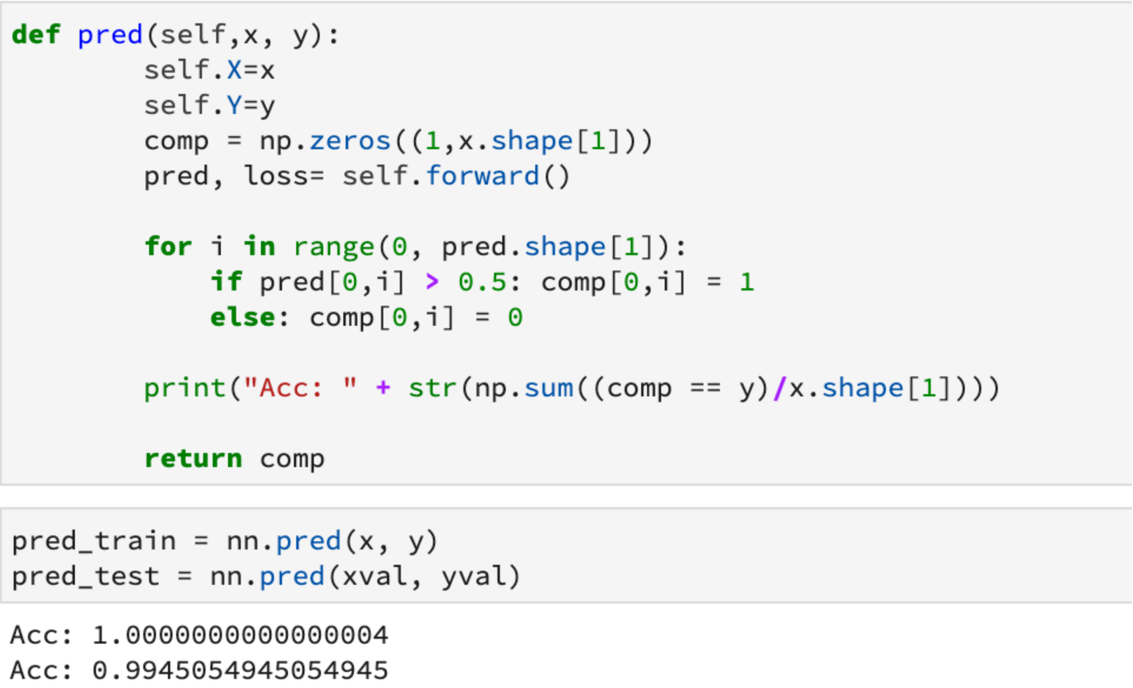


```
nn.X,nn.Y=x, y
target=np.around(np.squeeze(y), decimals=0).astype(np.int)
predicted=np.around(np.squeeze(nn.pred(x,y)), decimals=0).astype(np.int)
plotCf(target,predicted,'Cf Training Set')
nn.X,nn.Y=xval, yval
target=np.around(np.squeeze(yval), decimals=0).astype(np.int)
predicted=np.around(np.squeeze(nn.pred(xval,yval)), decimals=0).astype(np.int)
plotCf(target,predicted,'Cf Validation Set')
```

```
Acc: 1.0000000000000004
```



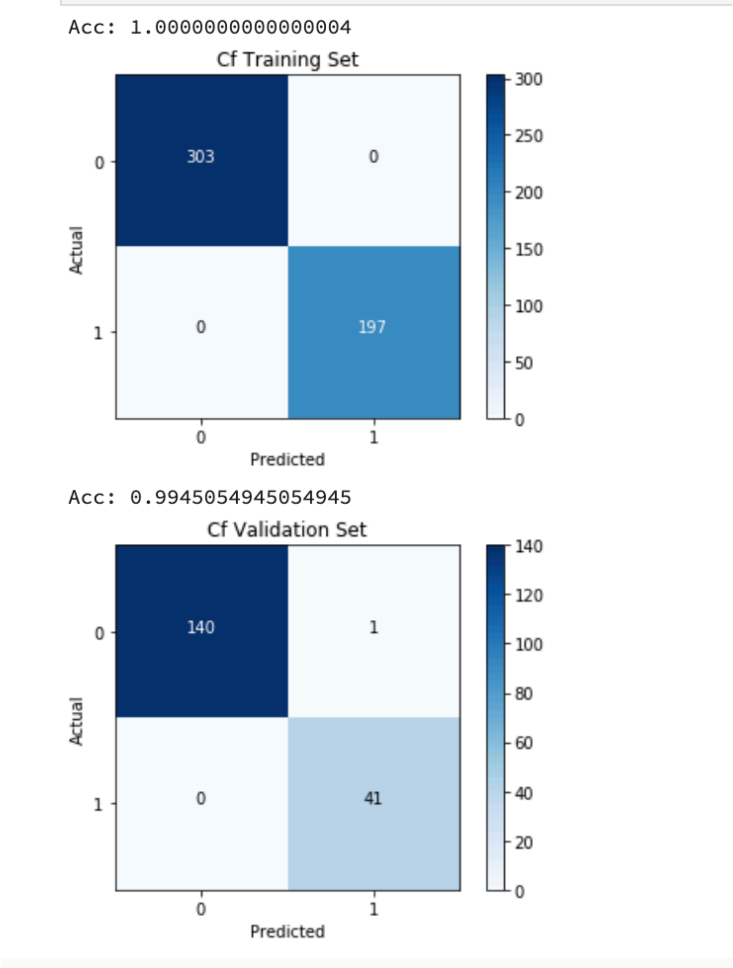
```
Acc: 0.9945054945054945
```



```
def pred(self,x, y):
        self.X=x
        self.Y=y
        comp = np.zeros((1,x.shape[1]))
        pred, loss= self.forward()

        for i in range(0, pred.shape[1]):
            if pred[0,i] > 0.5: comp[0,i] = 1
            else: comp[0,i] = 0

        print("Acc: " + str(np.sum((comp == y)/x.shape[1])))

        return comp
```

```
pred_train = nn.pred(x, y)
pred_test = nn.pred(xval, yval)
```

```
Acc: 1.0000000000000004
Acc: 0.9945054945054945
```



```
Acc: 1.0000000000000004
```

```
Acc: 0.9945054945054945
```

## REFERENCES

1. https://www.ijeat.org/wp-content/uploads/papers/v8i5/C5808028319.pdf
2. https://www.sciencedirect.com/science/article/pii/S1877050918309323
3. https://archive.siam.org/meetings/sdm06/workproceed/Scientific%20Datasets/bellaachia.pdf
4. https://www.ijitee.org/wp-content/uploads/papers/v8i6/F3384048619.pdf
5. https://www.sciencedirect.com/science/article/pii/S1877050918309323
6. https://www.researchgate.net/publication/332408487_Breast_Cancer_Prediction_using_SVM_with_PCA_Feature_Selection_Method