CSCE 633 MACHINE LEARNING Homework-1

Name: Sai Namith Garapati

UIN: 832001176

Question 1: Predicting patient post-surgery survival

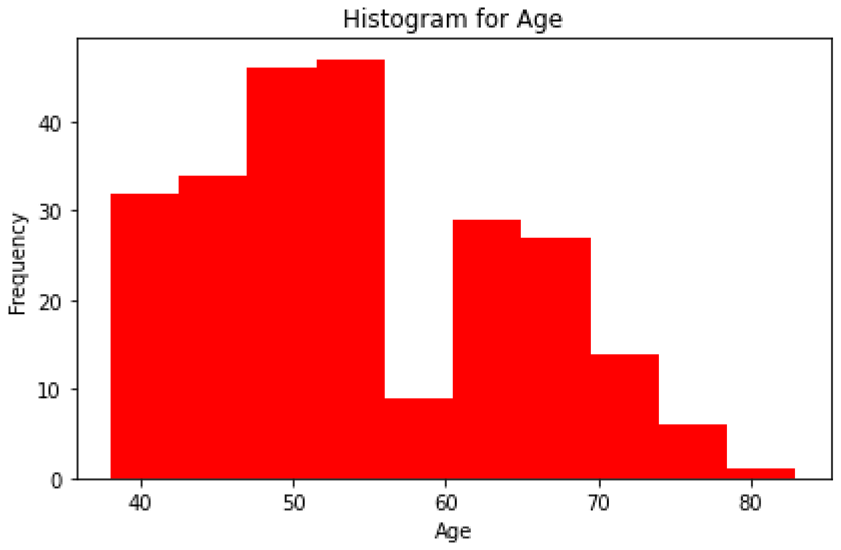
(a.i)

Ans: From the training data,173 samples belong to class 1 whereas 72 samples belong to class 2. The classes are not equally distributed since it can be observed that the samples are not equally divided between class 1 and class 2.

(a.ii)

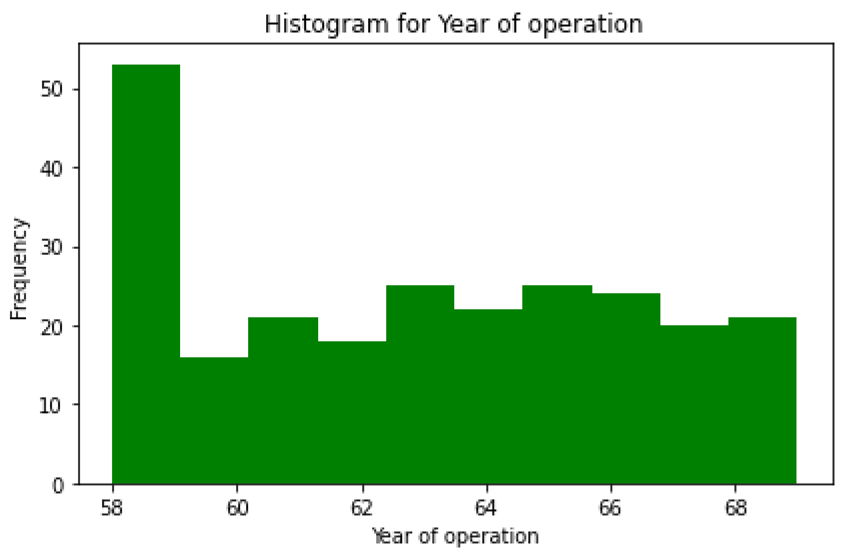
Ans: Three histograms are plotted for each of the three features using the training data.

Histogram for 'Age':



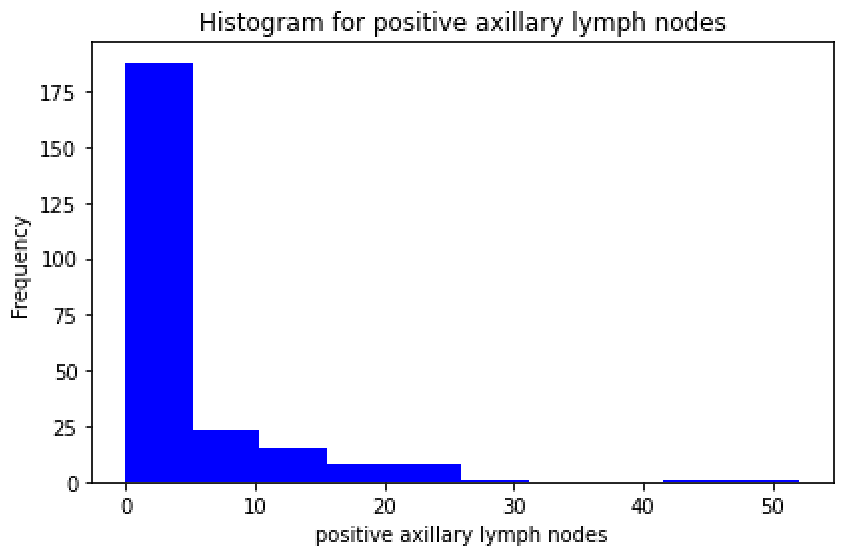
The histogram for age is a bimodal distribution.

Histogram for ‘Year of operation’:



Histogram for Year of operation is a multimodal distribution

Histogram for positive axillary lymph nodes:



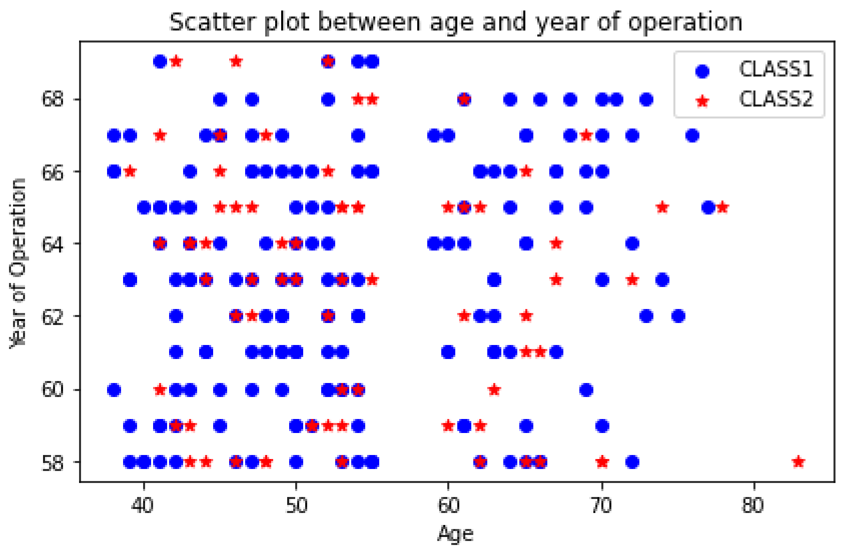
Histogram for positive axillary lymph nodes is a unimodal distribution

(a.iii)

Scatter plots for all pairs of features i.e., between Age and year of operation, year of operation and number of positive axillary lymph nodes, positive axillary lymph nodes and age have been plotted.

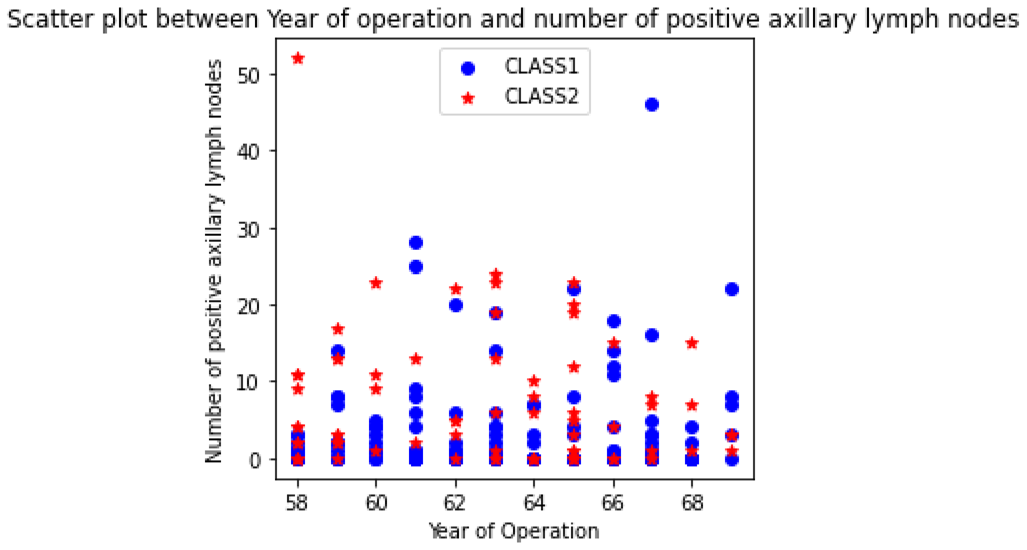
Color coding: Blue circle is used to indicate the samples that belong to class 1 whereas red star is used to indicate the samples that belong to class 2.

Scatter plot between Age and Year of operation:



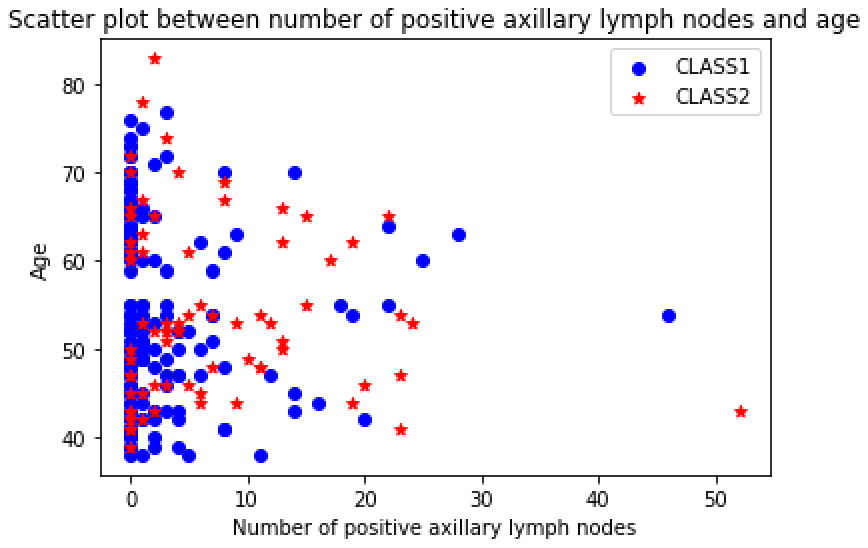
As observed from the scatter plot between age and year of operation, the classes are not separable.

Scatter plot between Year of operation and number of positive axillary lymph nodes:



As observed from the scatter plot between Year of operation and number of positive axillary nodes, the classes are not separable.

Scatter plot between number of positive axillary lymph nodes and Age:



As observed from the scatter plot between number of positive axillary nodes and Age, the classes are not separable.

There are no feature combinations for which the classes look separable, all three feature combinations are inseparable.

(b.i)

A K-Nearest neighbor Classifier(K-NN) has been implemented to classify between the three classes using the Euclidean distance (l2-norm)

Code for the implementation of KNN:

```python
#b.i, b.ii
#function to calculate the euclidean distance
def eucld(x,y):
  distance = 0
  for i in range(len(x)):
    distance = distance + (x[i]-y[i])**2
  return (distance)**(1/2)
trainingdist=[]
for i in range((trainingdata.shape[0])):
  trainingdist.append(list(trainingdata.iloc[i, :]))

devdist =[]
for i in range((devdata.shape[0])):
  devdist.append(list(devdata.iloc[i, :]))

testdist=[]
for i in range((testdata.shape[0])):
  testdist.append(list(testdata.iloc[i, :]))


#function that returns the nearest k neighbors

def obtain_neighbours(k,rnum):
  neighbours = {}
  dist = 0
  last_neighbours = []
  for i in range(len(trainingdist)):
    dist = eucld(trainingdist[i],devdist[rnum])
    neighbours[i] = dist
  sorted_dict = sorted(neighbours.items(), key=lambda item:item[1])
  for m in range(k):
    last_neighbours.append(sorted_dict[m])
  return last_neighbours

#implementation of KNN
knearest_neighbours = {}
temp = {}
```

```
k1 = [1,3,5,7,9,11,13]

accofdict = {}

for k in k1:
  for j in range(len(devdist)):
    knearest_neighbours[j] = obtain_neighbours(k,j)

  for i,j in enumerate(knearest_neighbours.values()):
    classlist = []
    for a,b in j :
      classlist.append(trainingdist[a][3])
    temp[i] = statistics.mode(classlist)

#Hyperparameter tuning is done by measuring accuracy and balanced accuracy
 by counting correctly classified and misclassified samples

  count_c1 = 0
  count_c2 = 0
  miscount_c1 = 0
  miscount_c2 = 0

  for i,j in enumerate(temp.values()):
    if((j == devdist[i][3] and j == 1)):
      count_c1 = count_c1 + 1
    elif ((j == devdist[i][3] and j == 2)):
      count_c2 = count_c2 + 1
    elif (devdist[i][3] == 1):
      miscount_c1 = miscount_c1 + 1
    elif (devdist[i][3] == 2):
      miscount_c2 = miscount_c2 + 1

  accofdev = (count_c1 + count_c2)/len(devdist)
  baccofdev = (count_c1/(count_c1+miscount_c1) + count_c2/(count_c2 + misc
ount_c2))/2
  accofdict[k] = (accofdev,baccofdev)

acclist = []
bacclist = []
for i,j in accofdict.values():
  acclist.append(i)
  bacclist.append(j)

#plot between the Accuracy and balanced accuracy for different values of k
plt.xlabel('k')
plt.ylabel('Acc,BAcc')
plt.plot(k1,acclist, label = 'Acc')
plt.plot(k1,bacclist,label = 'BAcc')
a = plt.gca()
a.legend()
```

The functioning of each part has been explained in the code itself by including comments. The explanation is elaborated below:

1) Implement a function for calculating the Euclidean distances between the sample with respect to all training data. In that function i has been iterated until the total number of features and the square root of sum of squares of distances gives us the Euclidian distances.
2) In the function that returns the nearest k neighbors, calculate the distance between the samples of training data and development sample and sort them. Make sure the distances are appended into a dictionary where we can also sort the indices along with the distances. After sorting store the k-nearest neighbors.
3) After getting the k-nearest neighbors, enumerate with respect to class 1 and class 2. Using mode, find out to which class does more samples are associated to and assign that class to it. In this way, KNN is successfully implemented
4) For hyperparameter tuning, we use the development set and find out the best hyperparameter with respect to classification accuracy and balanced classification accuracy. The formulas of accuracy and balanced accuracy are implemented by counting the correctly and incorrectly classified samples for each class. The hyperparameter that gives us the highest Acc and Bacc is the ideal hyper parameter to choose.
5) Evaluate the hyper parameter with the test data.

(b.ii)

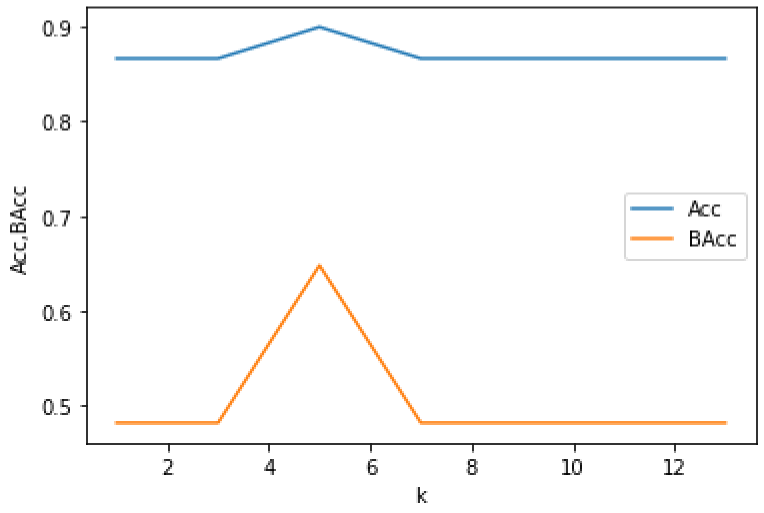
The model is trained for different values of K=1,3,5,7,9,11,13 using dev data, the classification accuracy (Acc) and the balanced classification accuracy (BAcc) have been evaluated for the model.

(Code and explanation for hyperparameter tuning has been given in b.i)

Acc at K = 1: 0.8666666666666667, Acc at K = 3: 0.8666666666666667, Acc at K = 5: 0.9, Acc at K = 7: 0.8666666666666667, Acc at K = 9: 0.8666666666666667, Acc at K = 11: 0.8666666666666667, Acc at K=13: 0.8666666666666667.

BAcc at K = 1 : 0.48148148148148145, BAcc at K = 3: 0.48148148148148145, BAcc at K = 5: 0.6481481481481481, BAcc at K = 7: 0.48148148148148145, BAcc at K = 9: 0.48148148148148145,BAcc at K = 11: 0.48148148148148145,Acc at K=13: 0.48148148148148145

Plot for Acc and BAcc metrics against different values of k:



Based on classification accuracy (Acc) metric, we can see that we get the highest accuracy at K = 5. So the best hyper-parameter K* based on Acc metric is K* = 5.

Based on Balanced classification accuracy (BAcc) metric, we can see that we get the highest accuracy at K = 5. So the best hyper-parameter K** based on BAcc metric is K** = 5.

(b.iii)

When the same analysis has been done on the test data with K* = 5 and K** =5, the values of Acc and BAcc are as follows

Acc at K* = 5 is 0.8387096774193549

BAcc at K** = 5 is 0.6466666666666666

(b.iv)

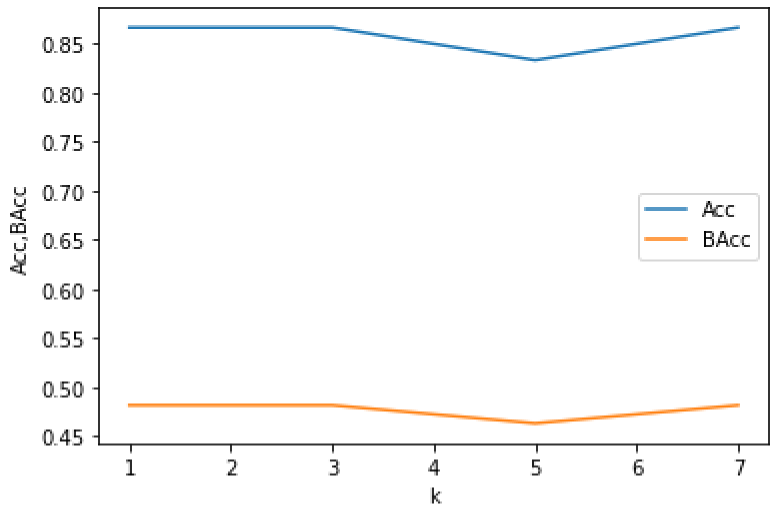
Instead of Euclidean distance, when KNN is implemented for l1-norm at K = 1,3,5,7.

Acc at K = 1: 0.8666666666666667, Acc at K = 3: 0.8666666666666667, Acc at K = 5: 0.8333333333333334, Acc at K = 7: 0.8666666666666667

BAcc at K = 1 : 0.48148148148148145, BAcc at K = 3: 0.48148148148148145, BAcc at K = 5: 0.46296296296296297, BAcc at K = 7: 0.48148148148148145

We find out that we get the least classification accuracy and balanced classification accuracy using l1 form at K = 5, for which we got the highest Acc and BAcc when we implemented with Euclidean distance

Plot for Acc and BAcc metrics against different values of k = 1,3,5,7 for l1 norm:



(c)

I feel that the model that we have deployed here is not so efficient model to implement in Memorial Merman Hospital in Houston. The following reasons make it a slightly inefficient model:

(i) For the test data, we are having only the accuracy of 83.8%. We can deploy much more powerful models with higher accuracy.

(ii) When we analyze the balanced classification accuracy for test data, we can see that it is having only 64.67% accuracy. The main point to observe here in BAcc is that the model is only efficiently classifying only class 1. For class 1, it is having an accuracy of 90% whereas for class 2 the model is having a horrific accuracy of around 30%. This makes physicians analyze the outcome in an incorrect way. When this model is deployed, although the patient will die within 5 years after surgery, it gives them wrong info 70% times that the patient would make it past 5 years. This makes this model skewed.

The training data provided to develop this model is very little which makes it less efficient. Much more data provided can enhance this model or newer methods can be deployed for better results.

Complete work has been done in Google colab. The Google colab notebook has been attached at the end of the homework. Earlier, code has only been mentioned in places where it was asked explicitly in the question to mention code.

QUESTION 3 : The survey has been completed by using email id : namith03@tamu.edu