Abhishek Sinha CSCE 633

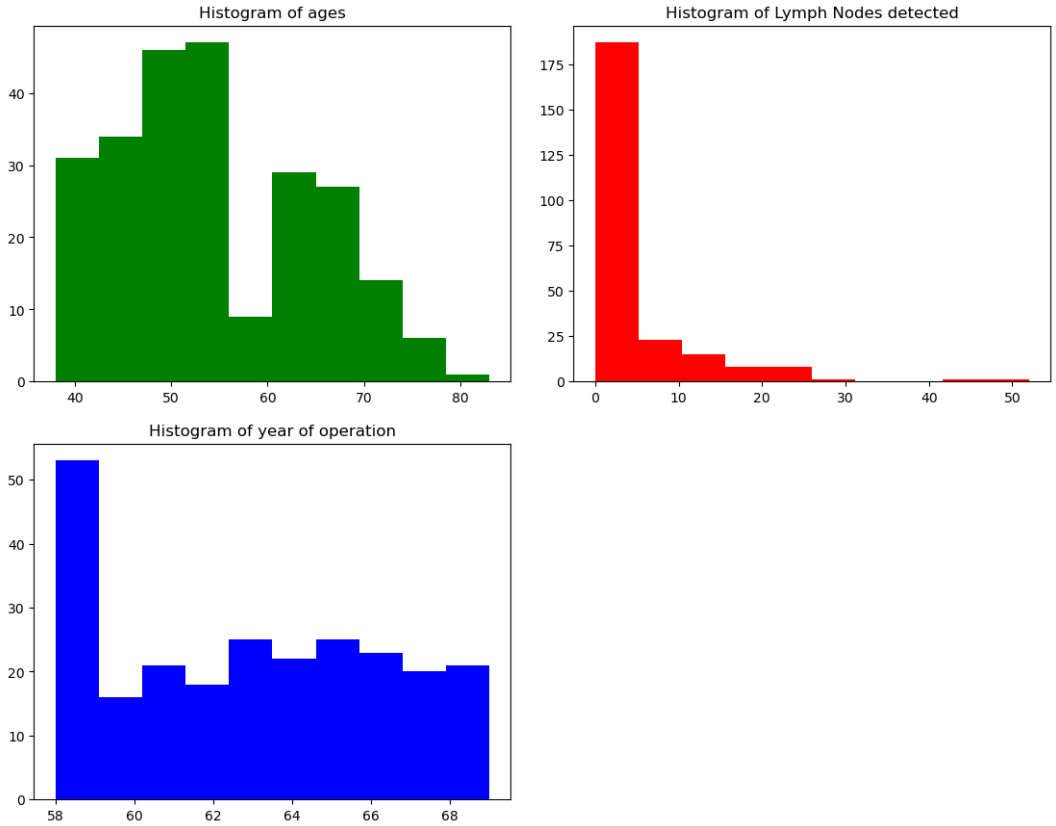
Homework Assignment 1

Question 1

a i): Samples belonging to class 1 is 172 while samples belonging to class 2 is just 72. The distribution shows that the classes are not equally distributed.

```
In [4]: data1['class'].value_counts()

Out[4]: 1    172
        2     72
        Name: class, dtype: int64
```

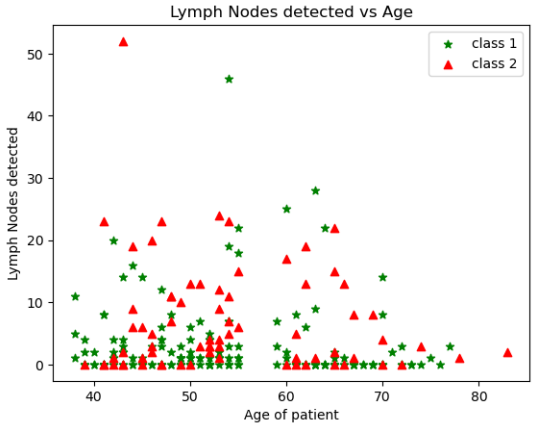
a ii): Below are the histograms of the features

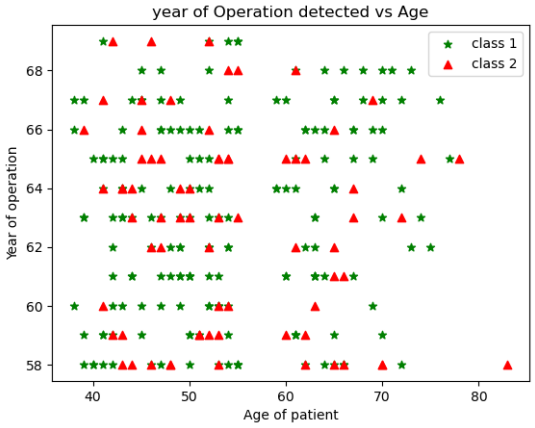


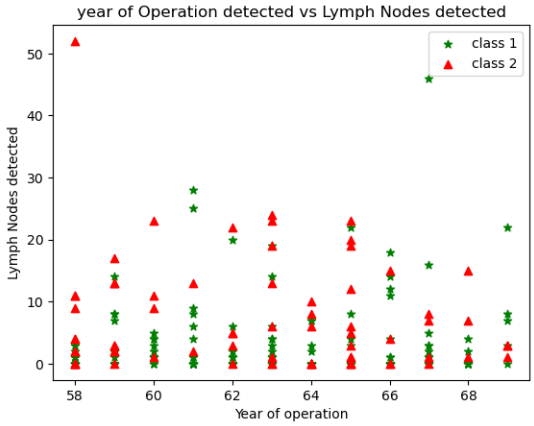
From the histogram, it looks that ages have a bimodel distribution, Lymph nodes detected and year of operation have a unimodel distribution.

Homework Assignment 1

a iii): Below are the scatter plots







We can observe from the scatter plots that there are no two features that can be used since the distributions show no definite pattern. If one would have to take a guess, then the best pair of features would be **Lymph nodes detected vs Age of patients**.

b i): Below is the code for KNN

```
import math
from collections import Counter
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np

#Normalize the data points
def normalize(X_train):
    for column in X_train.columns:
        X_train[column] = (X_train[column] - X_train[column].min()) /
   (X_train[column].max() - X_train[column].min())
    return X_train

#Define KNN classifier
def knn_Classifier(X_train, X_test, Y_train, k, p):
```

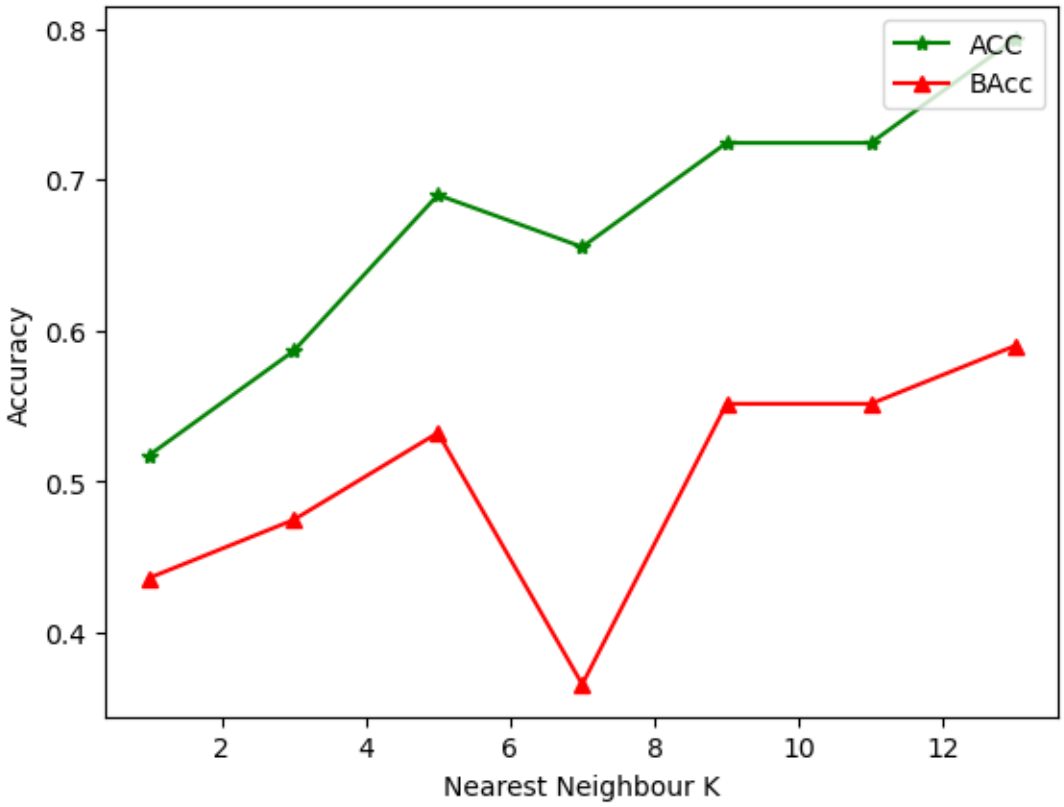
```
    Y_predicted = []
    for index, row in X_test.iterrows():
        #find the distance. Here P differentiates between L2 and L1 norm
        euclid_dist = []
        euclid_dist.append(((abs([row['age'], row['op_year'], row['lymph_detect']] - X_train)**p).sum(axis=1))**(1/p))
        df_dists = pd.DataFrame(euclid_dist).transpose()
        df_dists.set_index(Y_train.index)
        df_dists.columns = ['dist']
        df_nn = df_dists.sort_values(by=['dist'], axis=0)[:k]
        # Find the most common class around the test point
        counts = Counter(Y_train.iloc[df_nn.index])
        prediction = counts.most_common()[0][0]
        #Append the Prediction vector with the predicted class
        Y_predicted.append(prediction)
    return Y_predicted

#Read Test data
cols = ['age', 'op_year', 'lymph_detect', 'class']
data_test = pd.read_csv("data_test.csv")
data_test.columns = cols
for col in cols:
    data_test[col] = data_test[col].astype(float)

#Read Train data
cols = ['age', 'op_year', 'lymph_detect', 'class']
data1 = pd.read_csv("data_train.csv")
data1.columns = cols
for col in cols:
    data1[col] = data1[col].astype(float)

cols = ['age', 'op_year', 'lymph_detect', 'class']
data_dev = pd.read_csv("data_dev.csv")
data_dev.columns = cols
for col in cols:
    data_dev[col] = data_dev[col].astype(float)

Y_train = data1['class']
Y_test = data_dev['class']
X_train = normalize(data1.loc[:, data1.columns != "class"])
X_test = normalize(data_dev.loc[:, data_test.columns != "class"])


#Below is the exploration for L2 norm
def explore_KNN():
    total_sample = len(X_test)
    acc = []
    Bacc = []
    for i in range(1,15,2):
```

```
63          Y_pred = knn_Classifier(X_train, X_test, Y_train, i, 2)
64          myPrediction = Counter(Y_pred)
65          answers = Counter(Y_test)
66          pf = pd.DataFrame(Y_pred)
67          lm = pf[Y_test==1].subtract(Y_test[Y_test==1], axis=0).abs().
    sum()
68          lp = pf[Y_test==2].subtract(Y_test[Y_test==2], axis=0).abs().
    sum()
69          class1_correct = 0.5*(abs(answers[1] - lm)/answers[1])
70          class2_correct = 0.5*(abs(answers[2] - lp)/answers[2])
71          wrong_predict = (Y_test - Y_pred).abs().sum()
72          acc.append((total_sample-wrong_predict)/total_sample)
73          Bacc.append(class1_correct+class2_correct)
74 #      print(acc)
75      plt.plot(range(1,15,2), acc, label = 'ACC', marker = '*', color =
    'green')
76      plt.plot(range(1,15,2), Bacc, label = 'BAcc', marker = '^', color
    = 'red')
77      plt.xlabel('Nearest Neighbour K')
78      plt.ylabel('Accuracy')
79      plt.legend(loc="upper right")
80      plt.show()
81
82 explore_KNN()
83
84 #Below Snippet is for the exploration of L1 norm
85 def explore_KNN():
86      total_sample = len(X_test)
87      acc = []
88      Bacc = []
89      for i in range(1,15,2):
90          Y_pred = knn_Classifier(X_train, X_test, Y_train, i, 1)
91          myPrediction = Counter(Y_pred)
92          answers = Counter(Y_test)
93          pf = pd.DataFrame(Y_pred)
94          lm = pf[Y_test==1].subtract(Y_test[Y_test==1], axis=0).abs().
    sum()
95          lp = pf[Y_test==2].subtract(Y_test[Y_test==2], axis=0).abs().
    sum()
96          class1_correct = 0.5*(abs(answers[1] - lm)/answers[1])
97          class2_correct = 0.5*(abs(answers[2] - lp)/answers[2])
98          wrong_predict = (Y_test - Y_pred).abs().sum()
99          acc.append((total_sample-wrong_predict)/total_sample)
100         Bacc.append(class1_correct+class2_correct)
101 #      print(acc)
102     plt.plot(range(1,15,2), acc, label = 'ACC', marker = '*', color =
    'green')
103     plt.plot(range(1,15,2), Bacc, label = 'BAcc', marker = '^', color
    = 'red')
104     plt.xlabel('Nearest Neighbour K')
```

Homework Assignment 1

```
105     plt.ylabel('Accuracy')
106     plt.legend(loc="upper right")
107     plt.show()
108
109 explore_KNN()
```

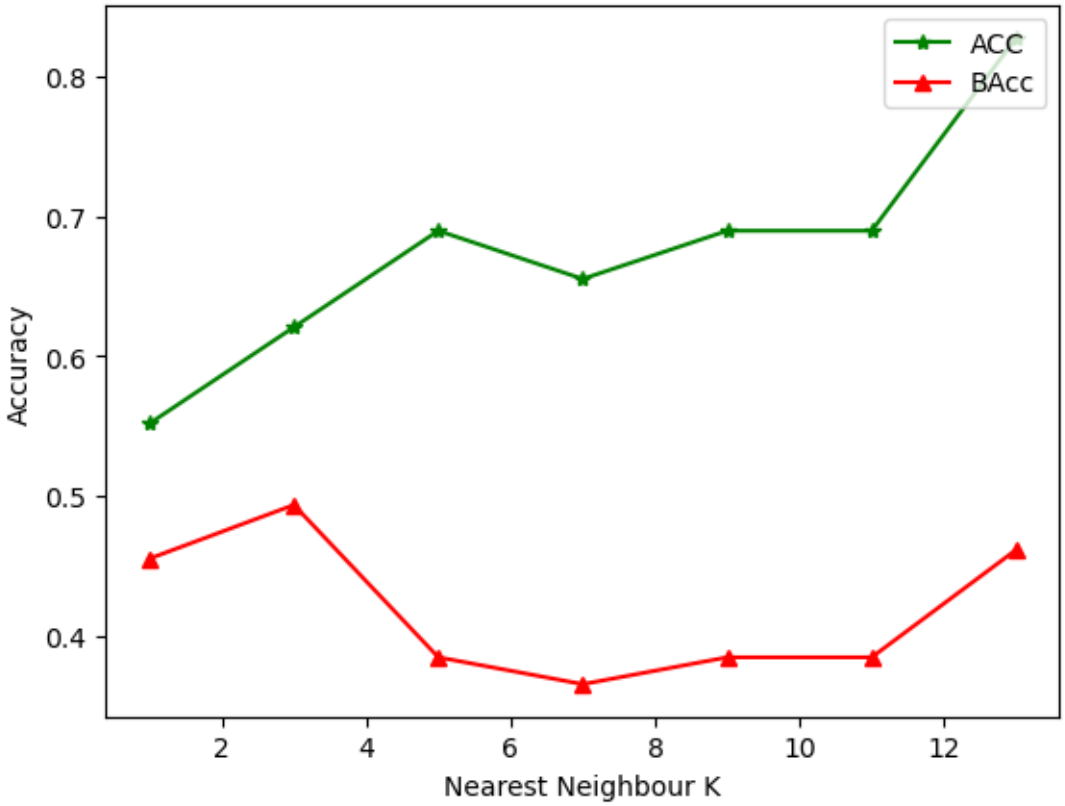
b ii): Below are the plots for different values of K on the development set



We can see that the best classification accuracy is when the nearest neighbours $K = 13$. The Balanced classification accuracy (BAcc) is best when number of nearest neighbours is also $K = 13$. However, there is a peak at K=5, where the BAcc is is 58.97%,while Acc is 79.31%.$On the test set, ACC with hyperparameter$K=13$ is$66.67%, $while BAcc is$86.20%.

b iii): Below plot shows the variation of ACC and BAcc with hyperparameter K with L1 norm on the development set

Homework Assignment 1



As it can be observed, Acc is maximum (about 49.35%) when hyper parameter is $K = 3$. Acc is maximum (about 82.79%) when hyper parameter is $K = 13$. For the test set, when K=13, Bacc is 56.25% while acc is 79.31%. With K=3, Bacc is 41.67% and acc is 65.52%

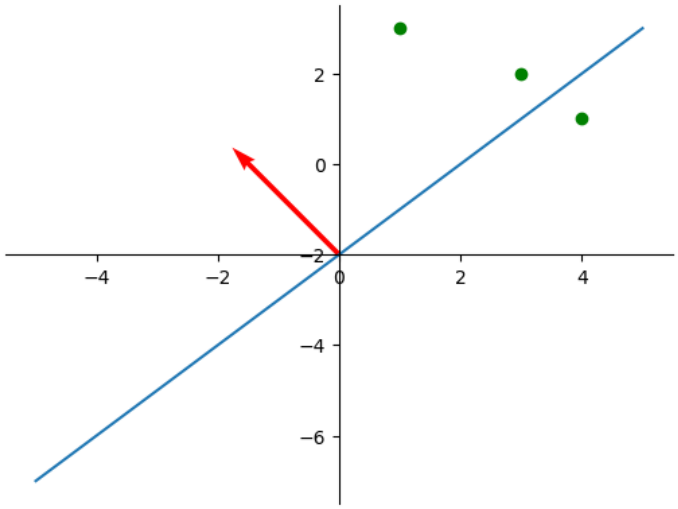
c): If Memorial Herman Hospital in Houston, TX is planning to deploy this system over the next months in order to predict patient post-surgery mortality rate, there are many things that are to be taken care off:

- The accuracy of the model could be better if more data points were available.
- The current model uses just 3 features for its prediction. Perhaps better features and rigourous feature engineering would better help predicting even with limited data set
- Just employing KNN to predict post surgery mortality would be not a good idea, since there are other models (Deep neural Networks) already present which might be able to learn more about the inter-dependencies than a simple KNN.
- To improve the accuracy of the model, it would be a better idea to use ensemble of models, rather than stick to a single model.
- Though not used in this model, it would be better to train the model on negative examples or various sets of examples to improve prediction.

Homework Assignment 1

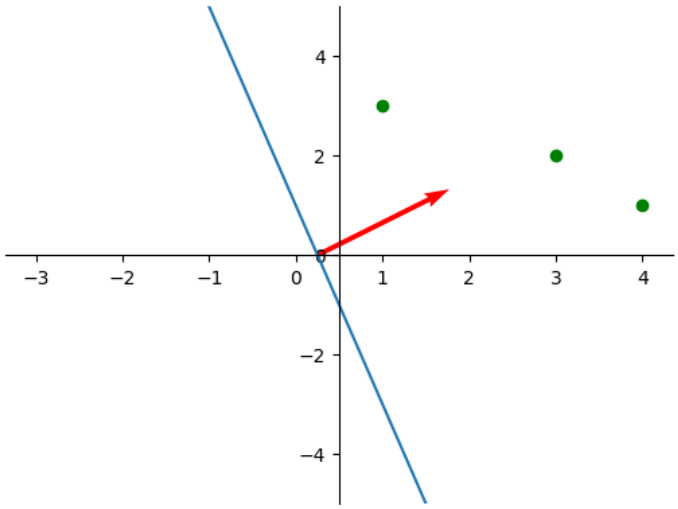
Question 2

(i) Plot is as follows:



(ii) Based on the $sign(\mathbf{w(t)}^T x_n)$, calculations are as follows:
$[2 -1\ 1]^T[1,3] = 4$ thus class 1
$[2 -1\ 1]^T[3,2] = 1$ thus class 1
$[2 -1\ 1]^T[4,1] = -1$ thus class 2
based on this rule, point 2 (3,2) is mis-classified.

(iii) Based on the update rule, $\mathbf{w(t+1) = w(t) + yX}$ we get:
$[2 -1\ 1]^T + (-1)[1\ 3\ 2]^T = [1 -4\ -1]^T$
This line classifies point 1 (1,3) wrong. Below is the image (eq of line is: $1 - 4x - y = 0$):



Homework Assignment 1

(iv) Using the update rule again on the new line, with point 1 we get:
$[1 - 4\ -1]^T + (1)[1\ 1\ 3]^T = [2 - 3\ 2]^T$
Below is the final classification line, which classifies the points correctly (eq of line is $2 - 3x + 2y = 0$):