Concordia University
Engineering and
Computer Science

Binary search tree Assignment

Course:	COEN 352	**Section:**	
Assignment N◦	3	**Due Date:**	2023-06-16

Professor:

Members:

Name:	Harout Kayabalian	**ID:**	40209920
Name:	Usman Usman	**ID:**	40174744

“We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality”,

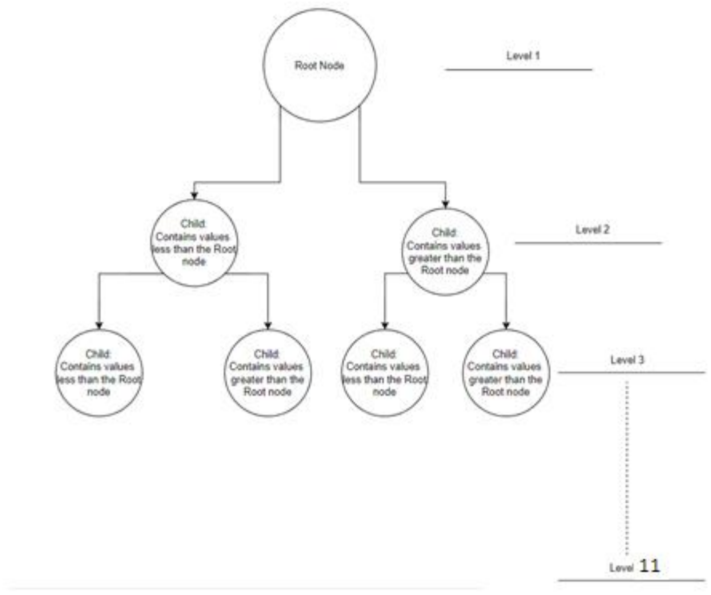
Abstract:

From the diagnostic data set provided by The Breast Cancer Wisconsin (Diagnostic) data set available at https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data?resource=download, we designed, implemented as well as trained a machine learning model to accurately predict the breast cancer diagnosis of a patient based on a set of relevant input features provided in the dataset. To accomplish this, the AI model is trained using a large data set consisting of the 10 most relevant attributes in the dataset for diagnosis namely radius mean, texture mean, perimeter mean, area mean, smoothness mean, compactness mean, concavity mean, concave points mean and symmetry mean. Therefore, Small to medium-sized datasets and the Multi-class classification we encountered while trying to build the AI model we decided to use K-NN model. An accuracy test as well as computational efficiency and scalability tests are provided to evaluate our machine learning model.

Description:

The main data structure used in this code is binary tree for storing the training data set and makes searching for values more efficient in both time and number of operations. In figure 2.1 we have an annotated diagram that visually represents the annotated diagram of the binary tree below:

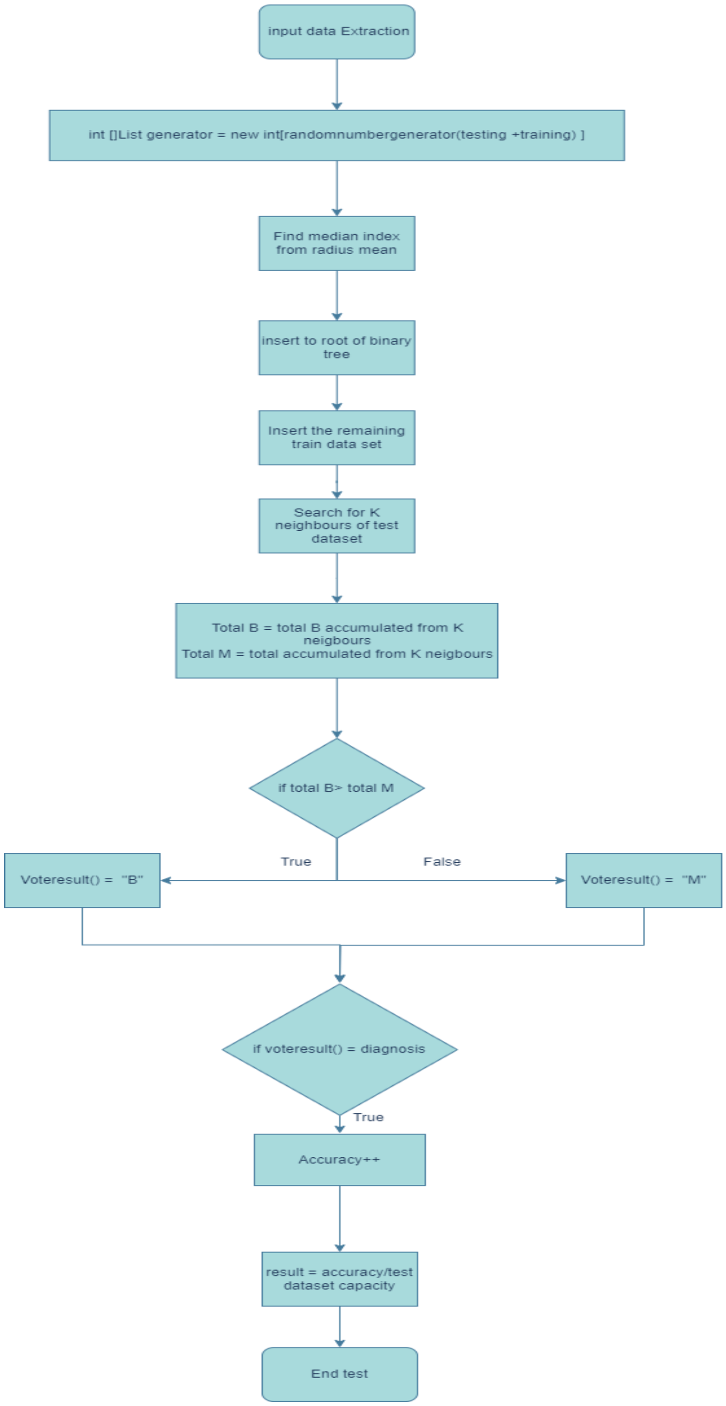
Figure 2.1: Annotated diagram of the Binary tree implementation


Root Node
Level 1
Child: Contains values less than the Root node
Child: Contains values greater than the Root node
Level 2
Child: Contains values less than the Root node
Child: Contains values greater than the Root node
Child: Contains values less than the Root node
Child: Contains values greater than the Root node
Level 3
Level 11

The root node contains the values that corresponds to the median of the radius mean. Each children node in the binary tree contains ten values corresponding to the ten relevant attributes in the data set.

The level of the binary tree was essential in determining the data set attribute used to find the Euclidean distance of the test data at each node it parses through. Therefore, since we are using 10 attributes from the data set sections the number of levels is also 10.

Fig 2.2: Highlevel Flowchart


input data Extraction
int []List generator = new int[randomnumbergenerator(testing +training)]
Find median index from radius mean
insert to root of binary tree
Insert the remaining train data set
Search for K neighbours of test dataset
Total B = total B accumulated from K neigbours
Total M = total accumulated from K neigbours
if total B> total M
True
False
Voteresult() = "B"
Voteresult() = "M"
if voteresult() = diagnosis
True
Accuracy++
result = accuracy/test dataset capacity
End test

Pseudocode:

Insert():

```
function insert(p: 2D array of strings, index: integer)
    currentlevel = root.level
    nodetobeinserted = new Node(p, index)
    i = root
    j = i
    k = 0

    while i is not null do
        if i.level == 2 then
            if i.radius_mean > nodetobeinserted.radius_mean then
                j = i
                i = i.left
                currentlevel = currentlevel + 1
                k = 0
            else
                j = i
                i = i.right
                currentlevel = currentlevel + 1
                  k = 1
            end if
else if i.level == 3 then do the same use (texture_mean)
else if i.level == 4 then do the same use (perimeter_mean)
else if i.level == 5 then do the same use (area_mean)
else if i.level == 6 then do the same use (smoothness_mean)
else if i.level == 7 then do the same use (compactness_mean)
else if i.level == 8 then do the same use (concavity_mean)
else if i.level == 9 then do the same use (concave points_mean)
else if i.level == 10 then do the same use (symmetry_mean)
else if i.level == 11 then do the same use (fractal_dimension_mean)
```

Findk_neighbours():

```
function findK_neighbours(p: 2D array of strings, index: integer, K: integer) -> array of integers
    nodetobeinserted = createNode(p, index)  // Create a new node to be inserted
    i = root  // Start traversal from the root
    j = i
    direction = 0
    alldistances = empty array of doubles
    allindexes = empty array of integers

    while i is not null:
        if i.level == 2:
            if i.radius_mean > nodetobeinserted.radius_mean:
                j = i
                i = i.left
                direction = 0
            else:
                j = i
                i = i.right
                direction = 1Results:
else if i.level == 3 then do the same use (texture_mean)
else if i.level == 4 then do the same use (perimeter_mean)
else if i.level == 5 then do the same use (area_mean)
else if i.level == 6 then do the same use (smoothness_mean)
else if i.level == 7 then do the same use (compactness_mean)
else if i.level == 8 then do the same use (concavity_mean)
else if i.level == 9 then do the same use (concave points_mean)
else if i.level == 10 then do the same use (symmetry_mean)
else if i.level == 11 then do the same use (fractal_dimension_mea
```

Fig 4.1: Running Time graph

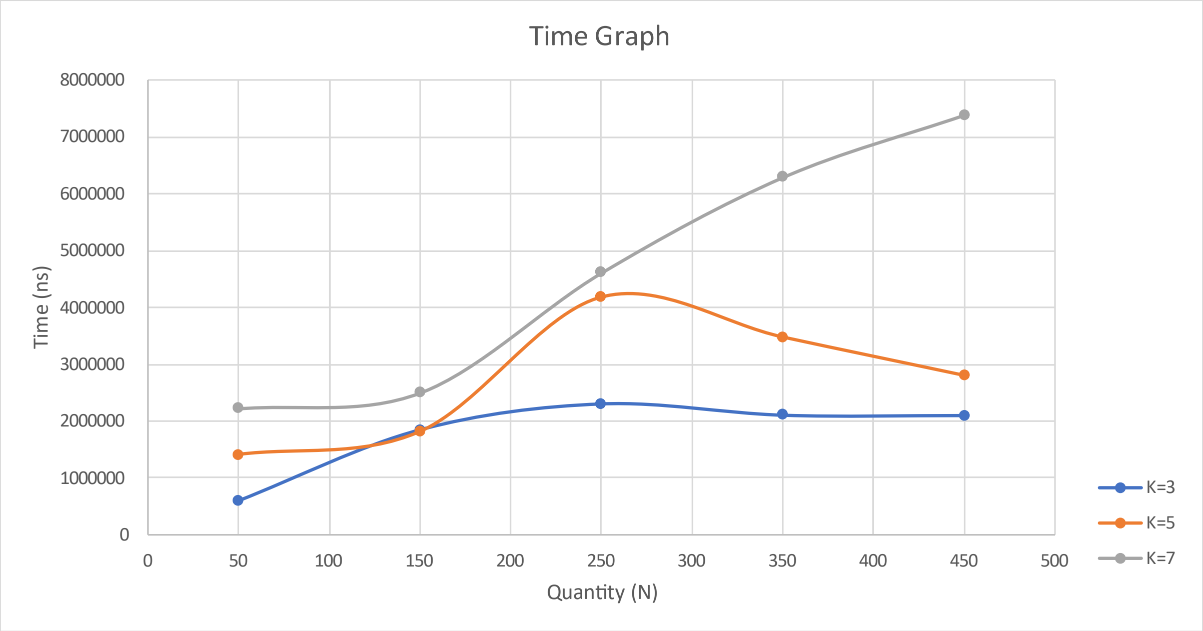
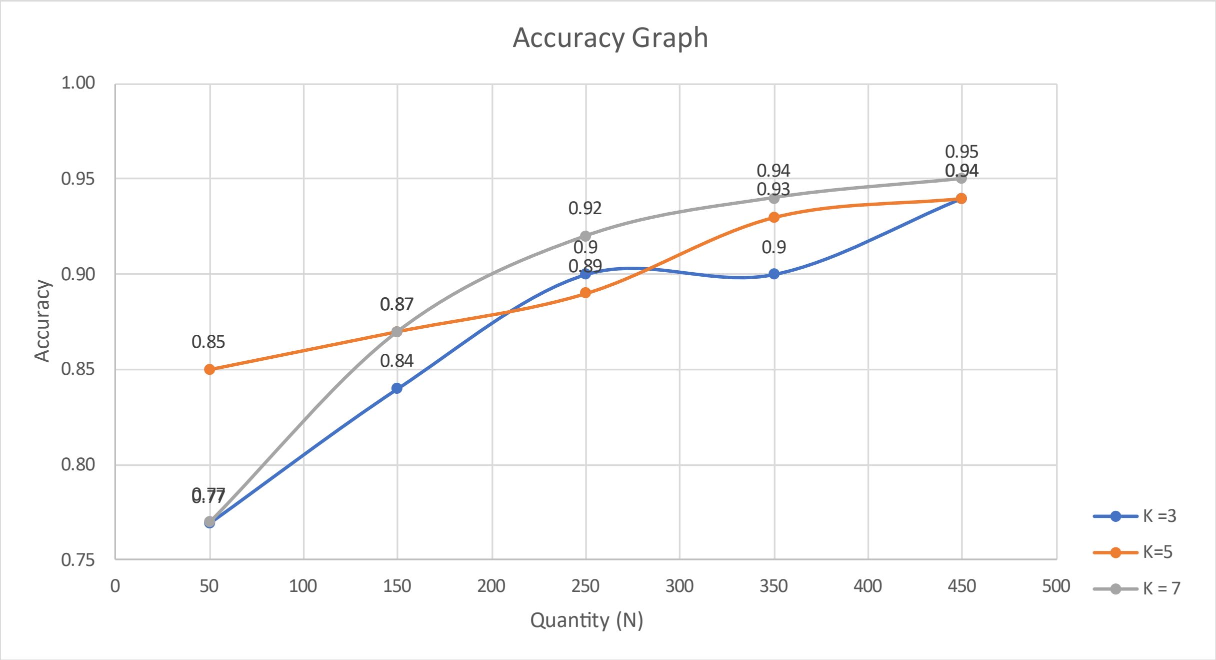


Fig 4.2: Testing Accuracy graph


Accuracy Graph
Accuracy
Quantity (N)
K =3
K=5
K = 7