Software Project Lab-1

Machine Learning Algorithms

Name : Abir Ashab Niloy

ID : BSSE-1315

Supervisor:

Kishan Kumar Ganguly

Assistant Professor,

Institute of Information Technology,

University of Dhaka

Machine Learning

- Machine learning is one of the most sought after technologies in the modern world.
- It is used to make predictions about a data. It generally takes a training dataset to train itself and then check the accuracy on test dataset.
- Finally it takes the input to predict about it.

OBJECTIVES

There are several algorithms regarding machine learning in which some of them are widely used.Here I will construct some introductory and widely used ML algorithms using C,C++ which will make my road of learning ML easier and will build a strong base for doing something bigger in future.The algorithms I will construct are given below:

-Logistic regression

-Decision tree

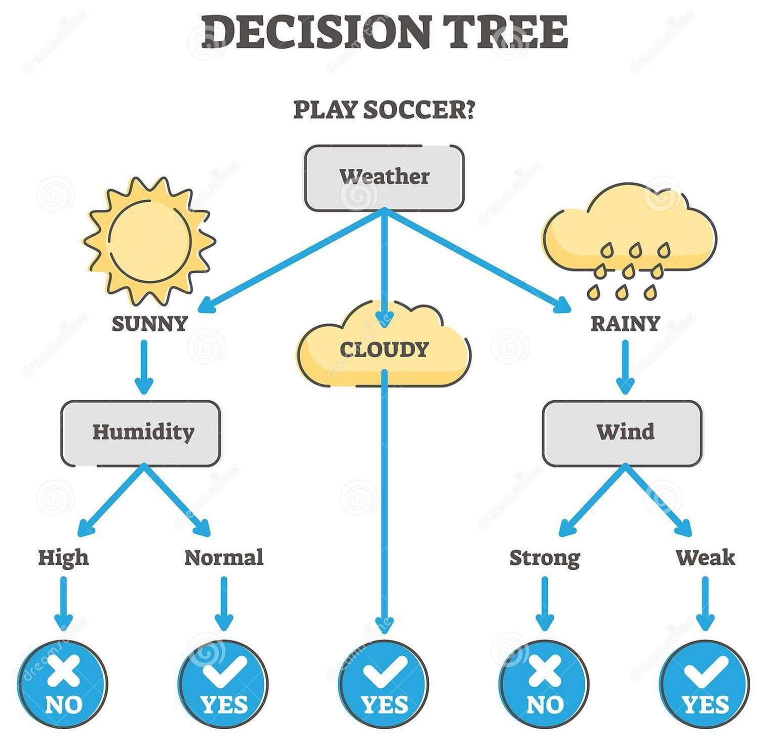
-Random forest

-K-Nearest Neighbor(KNN)

-K-means

Decision Tree

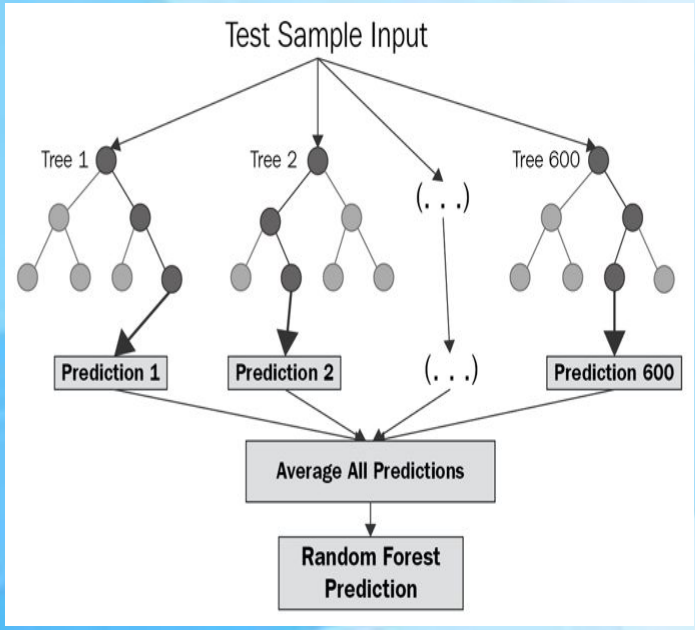
- Decision Tree Algorithm classifies a dataset based on some splitting criteria.



- It decreases the randomness of the data in each stage and specifies the categories to classify the data.

Random Forest

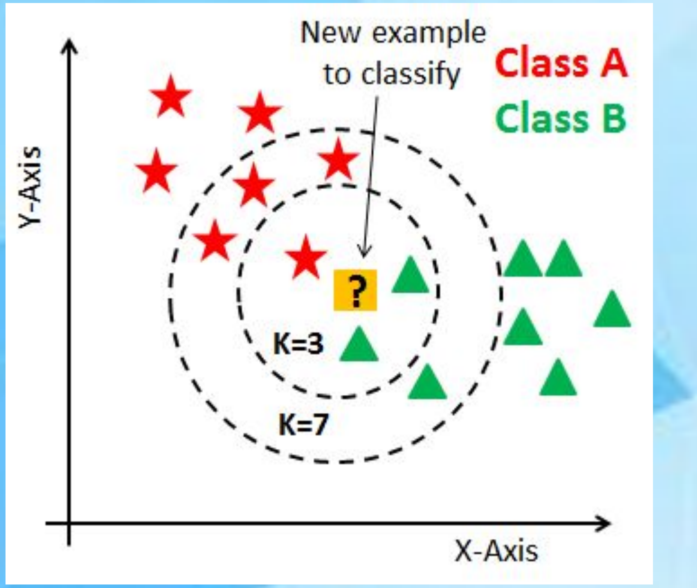
- Random forest is a combination of multiple decision tree. This is used to solve the problem of over specification/overfitting.



- The program makes several decision trees, each time with different criteria and then gives the average result of all the trees.

K-Nearest Neighbours

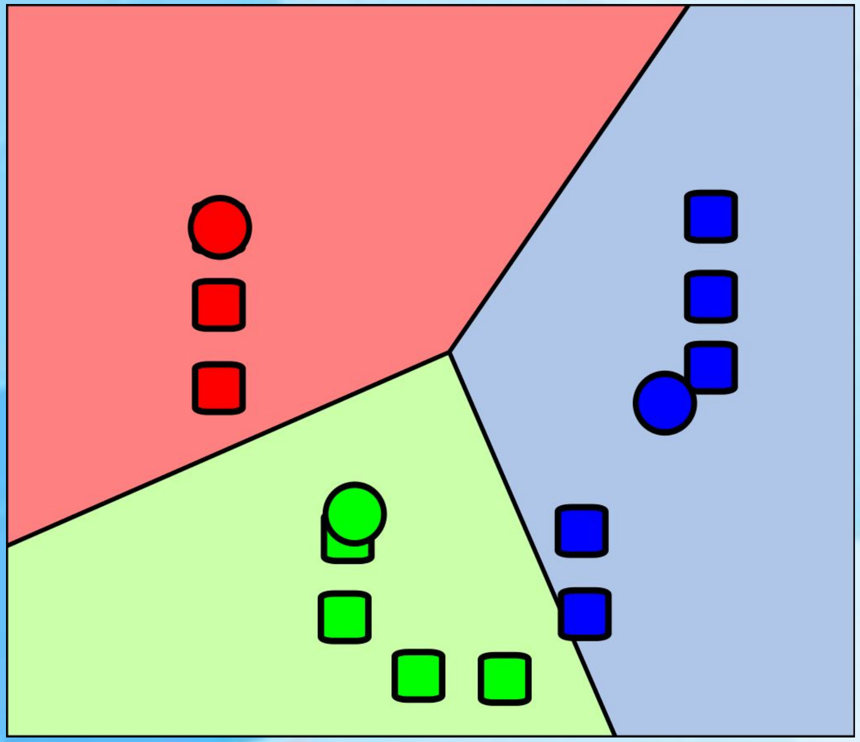
- Here the K-nearest neighbours algorithm work by calculating distance of all the data point from the unknown data point And then sort them to find k number of neighbours to the test data and then returns the class with highest frequency among those k neighbours.



Here the algorithm will return B for k=3 and A for k=7.

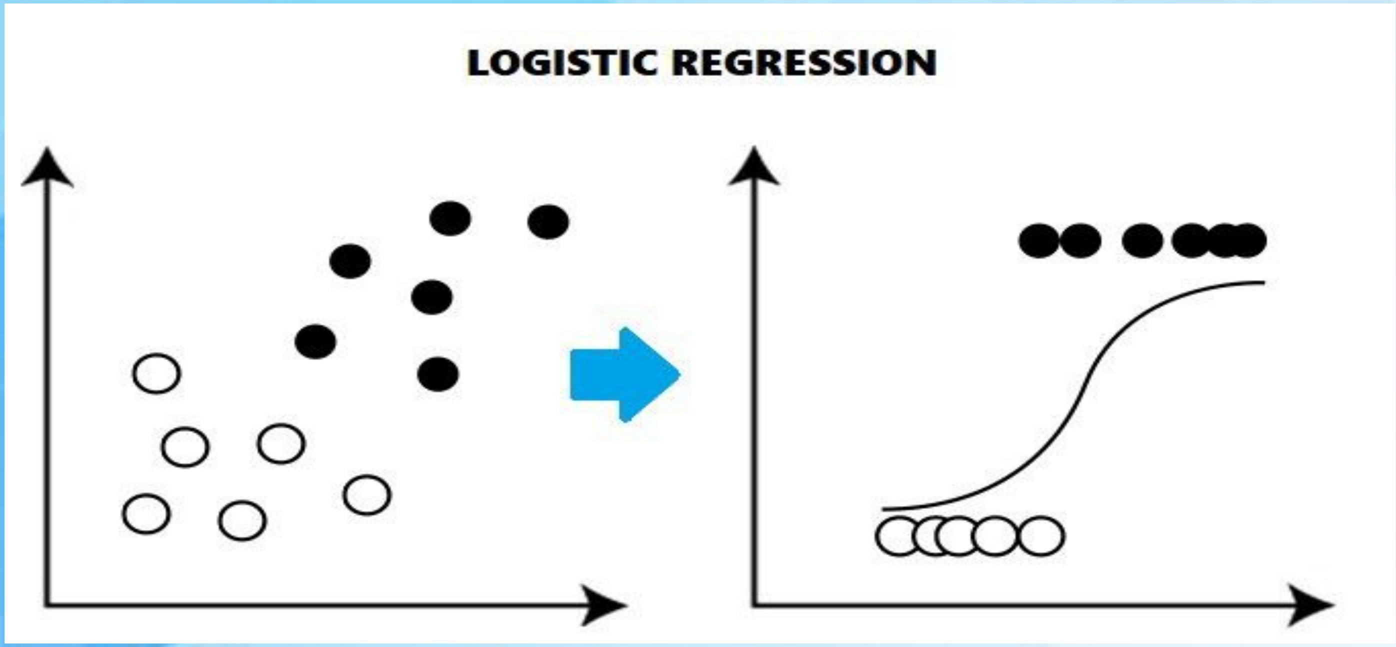
K-Means

It divides all the data points in k different clusters.Here k = 3.



Logistic Regression

Logistic regression is used in various fields including ML,mostly in medical fields and social science.

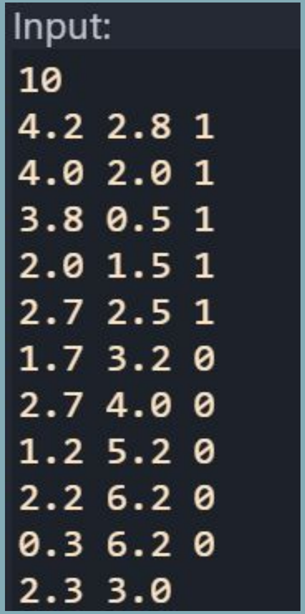


Progress till now….

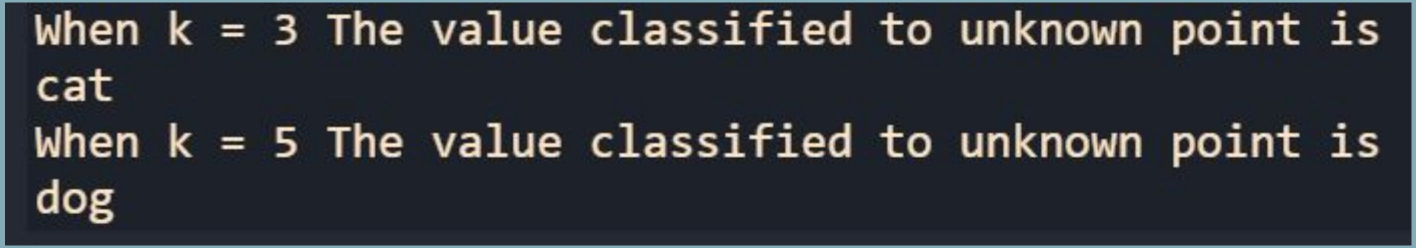
I have worked on **KNN, K-Means and Decision Tree** algorithms.

KNN : Here I took some data point as well as their class as input (i take the class as 1, 0 where 1 means dog and 0 means cat).After taking input I put an unknown point(2.3, 3.0) for knowing the class of this point(0 or 1).

The number of times I take different k,the class of the unknown data point might differ also.Here is the input and output for my KNN Algorithm -



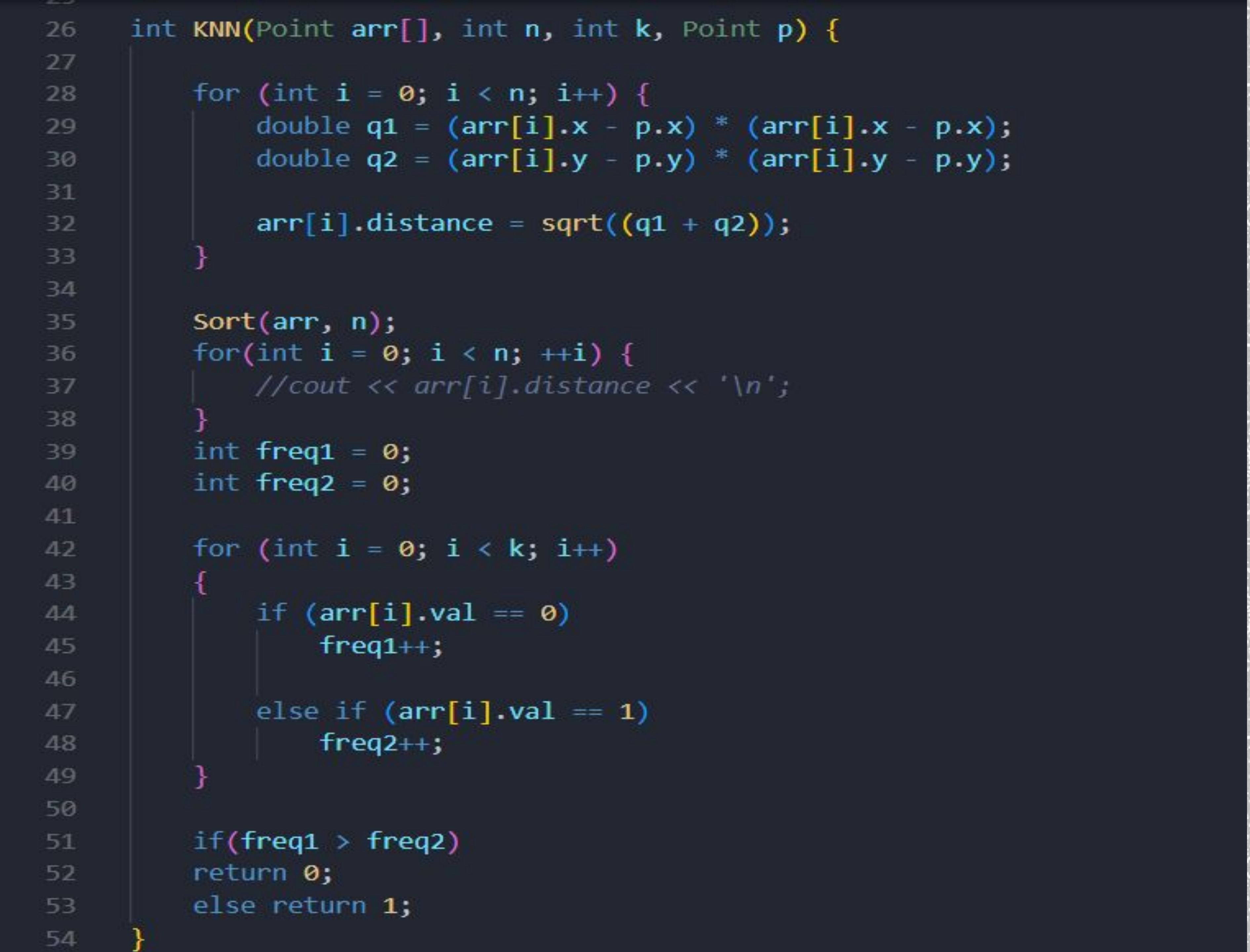
```
Input:
10
4.2 2.8 1
4.0 2.0 1
3.8 0.5 1
2.0 1.5 1
2.7 2.5 1
1.7 3.2 0
2.7 4.0 0
1.2 5.2 0
2.2 6.2 0
0.3 6.2 0
2.3 3.0
```



```
When k = 3 The value classified to unknown point is
cat
When k = 5 The value classified to unknown point is
dog
```

kNN function

Knn function



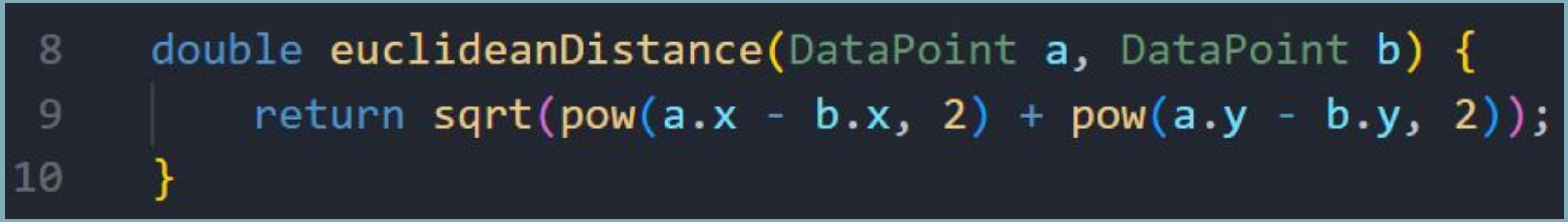
```
int KNN(Point arr[], int n, int k, Point p) {

    for (int i = 0; i < n; i++) {
        double q1 = (arr[i].x - p.x) * (arr[i].x - p.x);
        double q2 = (arr[i].y - p.y) * (arr[i].y - p.y);

        arr[i].distance = sqrt((q1 + q2));
    }

    Sort(arr, n);
    for(int i = 0; i < n; ++i) {
        //cout << arr[i].distance << '\n';
    }
    int freq1 = 0;
    int freq2 = 0;

    for (int i = 0; i < k; i++)
    {
        if (arr[i].val == 0)
            freq1++;

        else if (arr[i].val == 1)
            freq2++;
    }

    if(freq1 > freq2)
    return 0;
    else return 1;
}
```

Progress till now….

K-Means : Here i take the data point as input

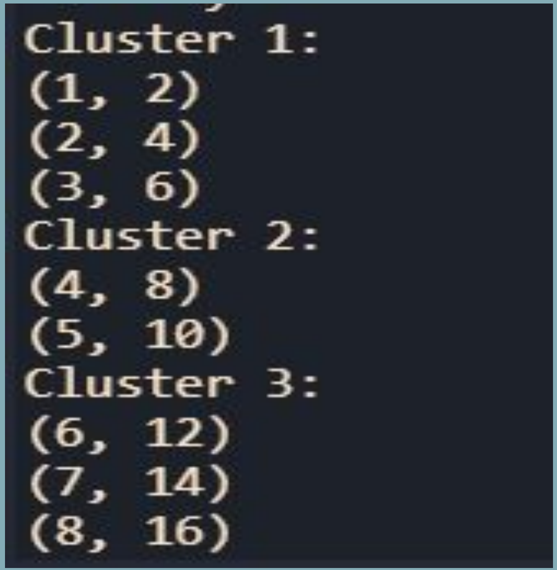
8 point -> (1,2),(2,4),(3,6),(4,8),(5,10,)(6,12),(7,14),(8,16)

And take pass the values along with k = 3 to the function K-means.Here i used euclidean equation for getting distance between two point.

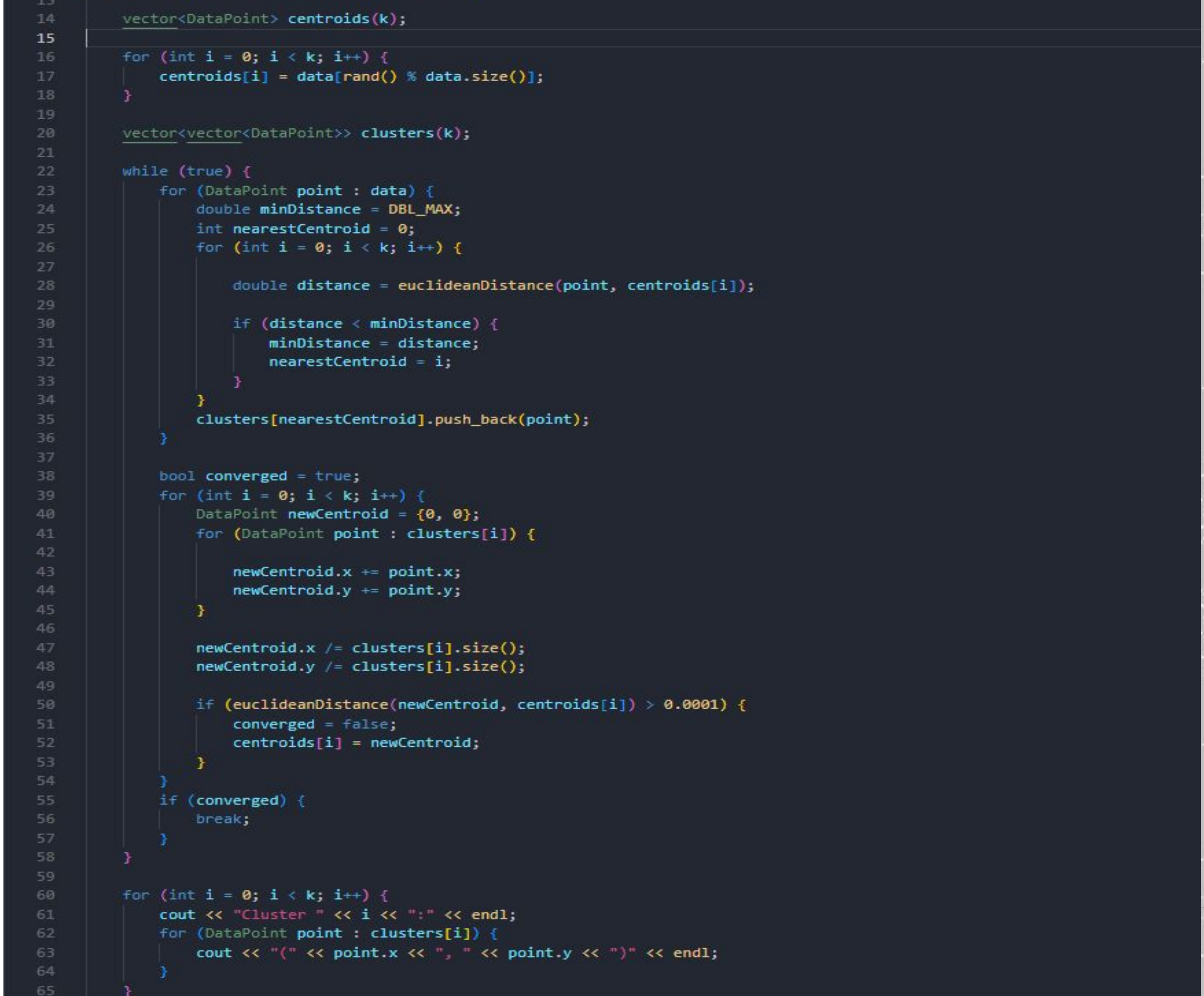


```
double euclideanDistance(DataPoint a, DataPoint b) {
    return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}
```

Output is :



```
Cluster 1:
(1, 2)
(2, 4)
(3, 6)
Cluster 2:
(4, 8)
(5, 10)
Cluster 3:
(6, 12)
(7, 14)
(8, 16)
```

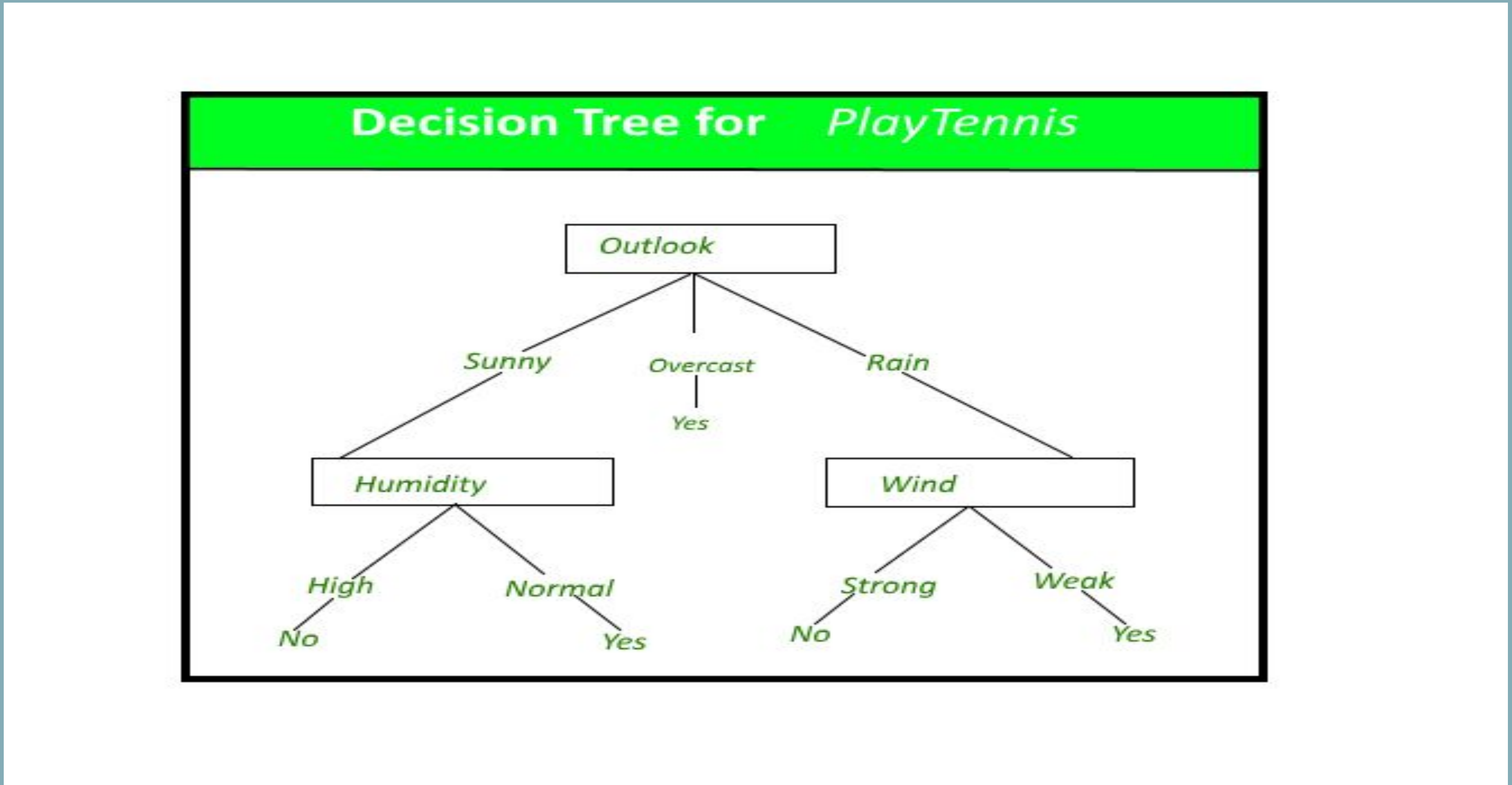


```cpp
    vector<DataPoint> centroids(k);

    for (int i = 0; i < k; i++) {
        centroids[i] = data[rand() % data.size()];
    }

    vector<vector<DataPoint>> clusters(k);

    while (true) {
        for (DataPoint point : data) {
            double minDistance = DBL_MAX;
            int nearestCentroid = 0;
            for (int i = 0; i < k; i++) {

                double distance = euclideanDistance(point, centroids[i]);

                if (distance < minDistance) {
                    minDistance = distance;
                    nearestCentroid = i;
                }
            }
            clusters[nearestCentroid].push_back(point);
        }

        bool converged = true;
        for (int i = 0; i < k; i++) {
            DataPoint newCentroid = {0, 0};
            for (DataPoint point : clusters[i]) {

                newCentroid.x += point.x;
                newCentroid.y += point.y;
            }

            newCentroid.x /= clusters[i].size();
            newCentroid.y /= clusters[i].size();

            if (euclideanDistance(newCentroid, centroids[i]) > 0.0001) {
                converged = false;
                centroids[i] = newCentroid;
            }
        }
        if (converged) {
            break;
        }
    }

    for (int i = 0; i < k; i++) {
        cout << "Cluster " << i << ":" << endl;
        for (DataPoint point : clusters[i]) {
            cout << "(" << point.x << ", " << point.y << ")" << endl;
        }
    }
```

K-Means Function

Progress till now….

Decision tree : Here i used ID3 (Iterative Dichotomiser 3) algorithm for building a decision tree.The tree I used as sample is :



Entropy & Gain

Entropy formula

$$-\sum_{i=1}^{c} P(x_i) log_b P(x_i)$$

Here ‘Pi’ is simply the frequentist probability of an element/class ‘i’ in our data.

Gain formula

$$Gain(T, a) = Entropy(T) - \sum_{i=1}^{|a|} \frac{|a_i|}{|T|} Entropy(a_i)$$

To be continued...

- My future goals regarding this project is to implement rest of the algorithms (Random forest, Logistic regression)
- Increasing productivity of each of these algorithms.
- Here is my Repository link for spl-1: https://github.com/Abir-Ashab/SPL-1

THANK YOU!!!