LAPORAN RESMI
PEKAN 11
MACHINE LEARNING

DISUSUN OLEH :
FABYAN KINDARYA
2110191024

NEURAL NETWORK

NODE.JAVA

```
public class Node {
    protected float value;
    protected float weight;

    public Node (float weight){
        this.weight = weight;
    }

    public void setValue(float value) {
        this.value = value;
    }

    public void setWeight(float weight) {
        this.weight = weight;
    }

    public float getValue() {
        return value;
    }

    public float getWeight() {
        return weight;
    }

    public float getWeightValue(){
        return (this.weight * this.value);
    }
}
```

DATA.JAVA

```
import java.util.List;

public class Data {
    private List<Float> elementList;
    private float output;

    public Data(List<Float> elementList, float output){
        this.setElementList(elementList);
        this.setOutput(output);
    }

    public List<Float> getElementList() {
        return elementList;
    }

    public void setElementList(List<Float> elementList) {
        this.elementList = elementList;
    }

    public float getOutput() {
        return output;
    }

    public void setOutput(float output) {
        this.output = output;
    }
}
```

NEURALNETWORK.JAVA

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class NeuralNetwork {

    public static final float YA = 2f;
    public static final float TIDAK = 1f;

    public static final float BIAS = 1f;
    public static final float MAX_NUM = 1f;
    public static final float MIN_NUM = -1f;

    public static final float LEARNING_RATE = 0.1f;

    List<Node> nodeList;
    List<Data> dataList;

    public NeuralNetwork(){
        nodeList = new ArrayList<>();
        dataList = new ArrayList<>();
        createDataset();
        initNode();
    }

    public void initNode(){
        for(int i=0; i<dataList.get(0).getElementList().size(); i++){
            Random random = new Random();
            float weight  = random.nextFloat(MAX_NUM - MIN_NUM) + MIN_NUM;
            nodeList.add(new Node(weight));
        }
    }
```

NEURALNETWORK.JAVA

```
public void createDataset(){
        String line = "";
        String splitBy = ",";
        try{
            BufferedReader br = new BufferedReader(new FileReader("src\\heart.csv"));
            while ((line = br.readLine()) != null){
                String[] datafile = line.split(splitBy);
                List<Float> floatList = new ArrayList<>();
                floatList.add(BIAS);
                for(int i=0; i<datafile.length-1; i++){
                    floatList.add(Float.parseFloat(datafile[i]));
                }
                float target = Float.parseFloat(datafile[datafile.length-1]);
                dataList.add(new Data(floatList, target));
            }
        }
        catch (IOException e){
            e.printStackTrace();
        }
    }

    public float getSummationInput(List<Float> floatList){
        float sum = 0;
        for(int i=0; i<nodeList.size(); i++){
            nodeList.get(i).setValue(floatList.get(i));
        }
        for(Node node : nodeList){
            sum += node.getWeightValue();
        }
        return sum;
    }
```

NEURALNETWORK.JAVA

```
public boolean isUpdateWeight(float output, Data data){
        boolean isUpdate = false;
        if(output != data.getOutput()){
            isUpdate = true;
            float error = (float) (data.getOutput() - output );
            for(Node node : nodeList){
                float weight = node.getWeight() + (LEARNING_RATE * node.getValue() * error);
                node.setWeight(weight);
            }
        }
        return isUpdate;
    }

public void compute(){
        boolean next = true;
        int iteration = 0;
        float error = 0;
        while (next) {
            int counterNext = 0;
            for (int i = 0; i < dataList.size(); i++) {
                List<Float> floatList = dataList.get(i).getElementList();
                float sum = getSummationInput(floatList);
                float output = getOutput(sum);

                if (isUpdateWeight(output, dataList.get(i))) {
                    counterNext++;
                }
            }
            iteration++;
            if(counterNext == 0 || (iteration >= 100)){
                error = ((float)counterNext / (float) dataList.size() * 100);
                next = false;
            }
        }
        printWeight();
        System.out.println("DONE in "+iteration+" iteration");
        System.out.println("\nError: "+error+" %");
    }
```

NEURALNETWORK.JAVA

```
    public float getOutput(float sum){
        float output;
        if(sum <= 0){
            output = TIDAK;
        }
        else {
            output = YA;
        }
        return output;
    }

    void printWeight(){
        int i =0;
        for(Node node : nodeList){
            System.out.println("BEST WEIGHT "+i+": "+node.getWeight());
            i++;
        }
    }

}
```

MAIN.JAVA

```
public class Main {
    public static void main(String[] args){
        NeuralNetwork neuralNetwork = new NeuralNetwork();
        neuralNetwork.compute();
    }
}
```

OUTPUT

```
C:\Users\user\.jdks\openjdk-17\bin\java.exe
BEST WEIGHT 0: 8.419757
BEST WEIGHT 1: 2.8008294
BEST WEIGHT 2: 148.02536
BEST WEIGHT 3: 289.9845
BEST WEIGHT 4: 77.888794
BEST WEIGHT 5: 22.217922
BEST WEIGHT 6: -32.091885
BEST WEIGHT 7: 160.92639
BEST WEIGHT 8: -127.328735
BEST WEIGHT 9: 139.63254
BEST WEIGHT 10: 286.1517
BEST WEIGHT 11: 168.3198
BEST WEIGHT 12: 337.02045
BEST WEIGHT 13: 778.8639
DONE in 100 iteration

Error: 28.518518 %
```