INDIRA GANDHI NATIONAL OPEN UNIVERSITY

LABORATORY RECORD

Month &Year : ……..….

Name : ……

Study Center : 1402, SH College, Thevara, Kochi-13

Course : ……..

Course Title : ……..……..…

……..……..

Course Code: ……..

Enrolment No: ……..……..

External Examiner Staff In-Charge

Computer Networks Lab

Session-1

1. Create a simple point to point network topology using two nodes.

Ans1:

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
// #include "ns3/simulator.h"

using namespace ns3;

int main(int argc, char *argv[]){
    NodeContainer nodes;
    nodes.Create(2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));

    NetDeviceContainer devices;
    devices =  pointToPoint.Install(nodes);

    InternetStackHelper stack;
    stack.Install(nodes);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0","255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign(devices);

    std::cout << "Node 0 - address - " << interfaces.GetAddress(0) << std::endl;
    std::cout << "Node 1 - address - " << interfaces.GetAddress(1) << std::endl;

    pointToPoint.EnablePcapAll("point-to-point-example");

    Simulator::Run();
    Simulator::Destroy();
    return 0;

}
```

2. Create a UdpClient and UdpServer nodes and communicate at a fixed data rate.

Ans2:

```
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/core-module.h"
```

```
#include "ns3/point-to-point-module.h"
#include "ns3/udp-client-server-helper.h"

using namespace ns3;
// NS_LOG_COMPONENT_DEFINE ("UdpTraceClientServerExample");
int main(int argc, char* argv[]){
   LogComponentEnable ("UdpTraceClient", LOG_LEVEL_INFO);
   LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
   // LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
   // LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);

   NodeContainer nodes;
   nodes.Create(2);

   PointToPointHelper pointToPoint;
   pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
   pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));

   NetDeviceContainer devices;
   devices = pointToPoint.Install(nodes);

   InternetStackHelper stack;
   stack.Install(nodes);

   Ipv4AddressHelper address;
   address.SetBase("10.1.1.0","255.255.255.0");
   Ipv4InterfaceContainer interfaces = address.Assign(devices);

   std::cout << "Node 0 address - " << interfaces.GetAddress(0) << std::endl;
   std::cout << "Node 1 address - " << interfaces.GetAddress(1) << std::endl;

   pointToPoint.EnablePcapAll("Udp client server app");
   UdpServerHelper server(3000);
   ApplicationContainer apps = server.Install(nodes.Get(0));

   apps.Start(Seconds(2));
   apps.Stop(Seconds(10));

   UdpClientHelper client(interfaces.GetAddress(0),3000);
   client.SetAttribute("MaxPackets", UintegerValue(3));
   client.SetAttribute("Interval", TimeValue(Seconds(1)));
   client.SetAttribute("PacketSize", UintegerValue(2048));

   ApplicationContainer clientApps;
   clientApps = client.Install(nodes.Get(1));

   clientApps.Start(Seconds(2));
   clientApps.Stop(Seconds(10));
```

```
    Simulator::Run();
    Simulator::Destroy();

    return 0;
}
```

Session - 2

3. Measure the throughput (end to end) while varying latency in the network created in Session -1.

Ans3:

```
#include "ns3/core-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/udp-client-server-helper.h"


using namespace ns3;

int main(int argc, char *argv[]){
    NodeContainer nodes;
    nodes.Create(2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate",StringValue("15Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("1ms"));
    NetDeviceContainer devices = pointToPoint.Install(nodes);

    InternetStackHelper stack;
    stack.Install(nodes);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0","255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign(devices);

    UdpServerHelper server(3000);
    ApplicationContainer app = server.Install(nodes.Get(0));
    app.Start(Seconds(1));
    app.Stop(Seconds(10));

    UdpClientHelper client(interfaces.GetAddress(0),3000);
    client.SetAttribute("MaxPackets",UintegerValue(3));
    client.SetAttribute("Interval", TimeValue(Seconds(2)));
    client.SetAttribute("PacketSize", UintegerValue(2048));

    ApplicationContainer clientApp = client.Install(nodes.Get(1));
    clientApp.Start(Seconds(2));
    clientApp.Stop(Seconds(10));

    FlowMonitorHelper flowmon;
```

```
   Ptr<FlowMonitor> monitor = flowmon.InstallAll();

   Simulator::Stop(Seconds(20));
   Simulator::Run();

   Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier());
   std::map<FlowId,FlowMonitor::FlowStats> stats = monitor->GetFlowStats();

   for(std::map<FlowId,FlowMonitor::FlowStats>::const_iterator
i=stats.begin();i!=stats.end();++i){
       std::cout << "RxBytes - " << i->second.rxBytes << std::endl;
       std::cout << "TxBytes - " << i->second.txBytes << std::endl;
       std::cout << "Throughput in Mbps - "<<
       i->second.rxBytes * 8.0 /(i->second.timeLastRxPacket.GetSeconds() -
i->second.timeFirstTxPacket.GetSeconds()) / 1024/1024 <<  " Mbps" << std::endl;
   }

   return 0;
}
```

4. Create a simple network topology having two client node on left side and two server nodes on the right side. Both clients are connected with another node n1. Similarly, both server node connecting to node n2. Also connect node n1 and n2 thus forming a dumbbell shape topology. Use point to point link only.

Ans4:

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/udp-client-server-helper.h"
#include "ns3/flow-monitor-module.h"

using namespace ns3;

int main(int argc, char* argv[]){
   NodeContainer clients;
   clients.Create(2);

   NodeContainer servers;
   servers.Create(2);

   NodeContainer n1n2;
   n1n2.Create(2);
```

```
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    // for clients
    NetDeviceContainer clientDevices1 = pointToPoint.Install(clients.Get(0),n1n2.Get(0));
    NetDeviceContainer clientDevices2 = pointToPoint.Install(clients.Get(1),n1n2.Get(0));

    // for servers
    NetDeviceContainer serverDevices1 = pointToPoint.Install(servers.Get(0),n1n2.Get(1));
    NetDeviceContainer serverDevices2 = pointToPoint.Install(servers.Get(1),n1n2.Get(1));

    // for n1 n2
    NetDeviceContainer n1n2Devices = pointToPoint.Install(n1n2.Get(0),n1n2.Get(1));

    InternetStackHelper stack;
    stack.Install(clients);
    stack.Install(servers);
    stack.Install(n1n2);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0","255.255.255.0");

    // client devices
    Ipv4InterfaceContainer clientInterfaces1 = address.Assign(clientDevices1);
    Ipv4InterfaceContainer clientInterfaces2 = address.Assign(clientDevices2);

    // server devices
    Ipv4InterfaceContainer serverInterfaces1 = address.Assign(serverDevices1);
    Ipv4InterfaceContainer serverInterfaces2 = address.Assign(serverDevices2);

    // n1n2 devices
    Ipv4InterfaceContainer n1n2Interfaces = address.Assign(n1n2Devices);

    pointToPoint.EnablePcapAll("dumbbell-topology");

    std::cout << "Client node 0 Ip Address - " << clientInterfaces1.GetAddress(0) <<
std::endl;
    std::cout << "Client node 1 Ip Address - " << clientInterfaces2.GetAddress(0) <<
std::endl;
    std::cout << "N1N2 node 0 Ip Address - " << n1n2Interfaces.GetAddress(0) << std::endl;
    std::cout << "N1N2 node 1 Ip Address - " << n1n2Interfaces.GetAddress(1) << std::endl;
    std::cout << "Server node 0 Ip Address - " << serverInterfaces1.GetAddress(0) <<
std::endl;
    std::cout << "Server node 1 Ip Address - " << serverInterfaces2.GetAddress(0) <<
std::endl;

    Simulator::Run();
    Simulator::Destroy();
```

```
    return 0;
}
```

Session - 3
5. Install a TCP socket instance connecting either of the client node with either of the server node in session 2's network topology.

Ans5:

```
#include "ns3/core-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"

NS_LOG_COMPONENT_DEFINE("TCP");

using namespace ns3;

int main(int argc, char* argv[]){
    LogComponentEnable("TCP",LOG_LEVEL_INFO);

    NodeContainer clients;
    clients.Create(2);

    NodeContainer servers;
    servers.Create(2);

    NodeContainer n1n2;
    n1n2.Create(2);

    PointToPointHelper p2p;
    p2p.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    p2p.SetChannelAttribute("Delay",StringValue("2ms"));

    NetDeviceContainer clientDevices1 = p2p.Install(clients.Get(0),n1n2.Get(0));
    NetDeviceContainer clientDevices2 = p2p.Install(clients.Get(1), n1n2.Get(0));

    NetDeviceContainer serverDevices1 = p2p.Install(n1n2.Get(1),servers.Get(0));
    NetDeviceContainer serverDevices2 = p2p.Install(n1n2.Get(1), servers.Get(1));

    NetDeviceContainer n1n2Devices = p2p.Install(n1n2);

    InternetStackHelper stack;
    stack.Install(clients);
    stack.Install(servers);
    stack.Install(n1n2);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0","255.255.255.0");
```

```
    Ipv4InterfaceContainer clientInterfaces1 = address.Assign(clientDevices1);
    Ipv4InterfaceContainer clientInterfaces2 = address.Assign(clientDevices2);

    Ipv4InterfaceContainer serverInterfaces1 = address.Assign(serverDevices1);
    Ipv4InterfaceContainer serverInterfaces2 = address.Assign(serverDevices2);

    Ipv4InterfaceContainer n1n2Interfaces = address.Assign(n1n2Devices);

    std::cout << "Client node 0 Ip Address - " << clientInterfaces1.GetAddress(0) <<
std::endl;
    std::cout << "Client node 1 Ip Address - " << clientInterfaces2.GetAddress(0) <<
std::endl;
    std::cout << "Server node 0 Ip Address - " << serverInterfaces1.GetAddress(1) <<
std::endl;
    std::cout << "Server node 0 Ip Address - " << serverInterfaces2.GetAddress(1) <<
std::endl;
    std::cout << "n1n2 node n1 Ip Address - " << n1n2Interfaces.GetAddress(1) <<
std::endl;
    std::cout << "n1n2 node n2 Ip Address - " << n1n2Interfaces.GetAddress(0) <<
std::endl;

    uint16_t port= 3000;

    Address serverAddress1(InetSocketAddress(serverInterfaces1.GetAddress(1),port));
    Address serverAddress2(InetSocketAddress(serverInterfaces2.GetAddress(1),port));

    PacketSinkHelper
packetSinkHelper("ns3::TcpSocketFactory",InetSocketAddress(Ipv4Address::GetAny(),port
));
    ApplicationContainer serverApps1 = packetSinkHelper.Install(servers.Get(0));
    ApplicationContainer serverApps2 = packetSinkHelper.Install(servers.Get(1));
    serverApps1.Start(Seconds(0));
    serverApps1.Stop(Seconds(10));

    serverApps2.Start(Seconds(0));
    serverApps2.Stop(Seconds(10));



    OnOffHelper onOffHelper1("ns3::TcpSocketFactory",serverAddress1);
    OnOffHelper onOffHelper2("ns3::TcpSocketFactory",serverAddress2);
    onOffHelper1.SetAttribute("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=1]"));

onOffHelper1.SetAttribute("OffTime",StringValue("ns3::ConstantRandomVariable[Constant
=0]"));
    onOffHelper1.SetAttribute("PacketSize",UintegerValue(1024));
    onOffHelper1.SetAttribute("MaxBytes",UintegerValue(1000000));
```

```
    onOffHelper2.SetAttribute("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=1]"));

onOffHelper2.SetAttribute("OffTime",StringValue("ns3::ConstantRandomVariable[Constant
=0]"));
    onOffHelper2.SetAttribute("PacketSize",UintegerValue(1024));
    onOffHelper2.SetAttribute("MaxBytes",UintegerValue(1000000));

    ApplicationContainer clientApps1 = onOffHelper1.Install(clients.Get(0));
    ApplicationContainer clientApps2 = onOffHelper2.Install(clients.Get(1));

    clientApps1.Start(Seconds(1));
    clientApps1.Stop(Seconds(10));

    clientApps2.Start(Seconds(1));
    clientApps2.Stop(Seconds(10));

    p2p.EnablePcapAll("tcp3-dumbbell-topology");
    Simulator::Run();
    Simulator::Destroy();


    return 0;
}
```

6. Install a TCP socket instance connecting other remaining client node with the remaining server node in session 2's network topology.

Ans6:

```
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include  "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

int main(int argc, char* argv[]){
    NodeContainer clients;
    clients.Create(2);

    NodeContainer servers;
    servers.Create(2);

    NodeContainer n1n2;
    n1n2.Create(2);

    PointToPointHelper p2p;
    p2p.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
```

```
    p2p.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer clientDevices1 = p2p.Install(clients.Get(0),n1n2.Get(0));
    NetDeviceContainer clientDevices2 = p2p.Install(clients.Get(1),n1n2.Get(0));

    NetDeviceContainer serverDevices1 = p2p.Install(n1n2.Get(1),servers.Get(0));
    NetDeviceContainer serverDevices2 = p2p.Install(n1n2.Get(1),servers.Get(1));

    NetDeviceContainer n1n2Devices = p2p.Install(n1n2.Get(0),n1n2.Get(1));

    InternetStackHelper stack;
    stack.Install(clients);
    stack.Install(servers);
    stack.Install(n1n2);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0","255.255.255.0");
    Ipv4InterfaceContainer clientInterfaces1 = address.Assign(clientDevices1);
    Ipv4InterfaceContainer clientInterfaces2 = address.Assign(clientDevices2);
    Ipv4InterfaceContainer serverInterfaces1 = address.Assign(serverDevices1);
    Ipv4InterfaceContainer serverInterfaces2 = address.Assign(serverDevices2);

    Ipv4InterfaceContainer n1n2Interfaces = address.Assign(n1n2Devices);
    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    uint16_t port = 3000;
    Address serverAddress1(InetSocketAddress(serverInterfaces1.GetAddress(1),port));
    Address serverAddress2(InetSocketAddress(serverInterfaces2.GetAddress(1),port));
    PacketSinkHelper
packetSinkHelper("ns3::TcpSocketFactory",InetSocketAddress(Ipv4Address::GetAny(),port
));

    ApplicationContainer serverApp1 = packetSinkHelper.Install(servers.Get(0));
    ApplicationContainer serverApp2 = packetSinkHelper.Install(servers.Get(1));

    serverApp1.Start(Seconds(1));
    serverApp1.Stop(Seconds(10));

    serverApp2.Start(Seconds(1));
    serverApp2.Stop(Seconds(10));

    OnOffHelper onOffHelper1("ns3::TcpSocketFactory",serverAddress1);

onOffHelper1.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant
=1]"));
    onOffHelper1.SetAttribute("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    onOffHelper1.SetAttribute("MaxBytes",UintegerValue(100000));
    onOffHelper1.SetAttribute("PacketSize",UintegerValue(1024));
```

```
    ApplicationContainer clientApp1 = onOffHelper1.Install(clients.Get(0));
    clientApp1.Start(Seconds(1));
    clientApp1.Stop(Seconds(10));

    OnOffHelper onOffHelper2("ns3::TcpSocketFactory",serverAddress2);

onOffHelper2.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant
=1]"));
    onOffHelper2.SetAttribute("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    onOffHelper2.SetAttribute("MaxBytes",UintegerValue(100000));
    onOffHelper2.SetAttribute("PacketSize",UintegerValue(1024));

    ApplicationContainer clientApp2 = onOffHelper2.Install(clients.Get(1));
    clientApp2.Start(Seconds(1));
    clientApp2.Stop(Seconds(10));
// hello
    OnOffHelper onOffHelper3("ns3::TcpSocketFactory",serverAddress1);

onOffHelper3.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant
=1]"));
    onOffHelper3.SetAttribute("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    onOffHelper3.SetAttribute("MaxBytes",UintegerValue(100000));
    onOffHelper3.SetAttribute("PacketSize",UintegerValue(1024));

    ApplicationContainer clientApp3 = onOffHelper3.Install(n1n2.Get(0));
    clientApp3.Start(Seconds(1));
    clientApp3.Stop(Seconds(10));

    OnOffHelper onOffHelper4("ns3::TcpSocketFactory",serverAddress2);

onOffHelper4.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariable[Constant
=1]"));
    onOffHelper4.SetAttribute("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    onOffHelper4.SetAttribute("MaxBytes",UintegerValue(100000));
    onOffHelper4.SetAttribute("PacketSize",UintegerValue(1024));

    ApplicationContainer clientApp4 = onOffHelper4.Install(n1n2.Get(1));
    clientApp4.Start(Seconds(1));
    clientApp4.Stop(Seconds(10));

    // p2p.EnablePcapAll("tcpdumb");
    p2p.EnablePcap("client0-server0",clientDevices1.Get(0),serverDevices1.Get(1));
    p2p.EnablePcap("client1-server1",clientDevices2.Get(0),serverDevices2.Get(1));

    Simulator::Run();
    Simulator::Destroy();
```

```
    return 0;
}
```

Session - 4

8. Take three nodes n1, n2 and n3 and create a wireless mobile ad-hoc network.

Ans 8:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-helper.h"
using namespace ns3;

int main (int argc, char *argv[])
{
  // Enable logging
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  // Create nodes
  NodeContainer nodes;
//   nodes.Create (2);
  nodes.Create(3);

  // Create mobility model
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
  positionAlloc->Add (Vector (0.0, 0.0, 0.0)); // Node 1 position
  positionAlloc->Add (Vector (5.0, 0.0, 0.0)); // Node 2 position
  positionAlloc->Add (Vector (8.0, 0.0, 0.0)); // Node 3 position
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);

  // Create wireless channel
  WifiHelper wifi;
  wifi.SetStandard (WIFI_STANDARD_80211b);

  // Set up physical layer
  YansWifiPhyHelper wifiPhy;
  YansWifiChannelHelper wifiChannel;
  wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
  wifiPhy.SetChannel (wifiChannel.Create ());
```

```
  // Set up MAC layer
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");

  // Install wifi on nodes
  NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, nodes);

  // Set up internet stack
  InternetStackHelper internet;
  internet.Install (nodes);

  // Assign IP addresses
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("192.168.1.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces = ipv4.Assign (devices);

  // Print IP addresses of nodes
  for (uint32_t i = 0; i < nodes.GetN (); ++i)
  {
    Ptr<Node> node = nodes.Get (i);
    Ptr<Ipv4> ipv4 = node->GetObject<Ipv4> ();
    Ipv4Address address = ipv4->GetAddress (1, 0).GetLocal ();
    std::cout << "Node " << i + 1 << " IP address: " << address << std::endl;
  }

  // Create applications (optional)
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  // Run simulation
  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
}
```

9. Install the optimized Link State Routing protocol on these nodes.

Ans9:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-helper.h"
using namespace ns3;

int main (int argc, char *argv[])
{
  // Enable logging
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  // Create nodes
  NodeContainer nodes;
//   nodes.Create (2);
  nodes.Create(3);

  // Create mobility model
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
  positionAlloc->Add (Vector (0.0, 0.0, 0.0)); // Node 1 position
  positionAlloc->Add (Vector (5.0, 0.0, 0.0)); // Node 2 position
  positionAlloc->Add (Vector (8.0, 0.0, 0.0)); // Node 3 position
  mobility.SetPositionAllocator (positionAlloc);
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);

  // Create wireless channel
  WifiHelper wifi;
  wifi.SetStandard (WIFI_STANDARD_80211b);

  // Set up physical layer
  YansWifiPhyHelper wifiPhy;
  YansWifiChannelHelper wifiChannel;
  wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
  wifiPhy.SetChannel (wifiChannel.Create ());

  // Set up MAC layer
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");

  // Install wifi on nodes
  NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, nodes);
```

```
  // Set up internet stack
  OlsrHelper olsr;

  Ipv4ListRoutingHelper list;
  list.Add (olsr, 10);

  InternetStackHelper internet;
  internet.SetRoutingHelper (list); // has effect on the next Install ()
  internet.Install (nodes);

  // Assign IP addresses
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("192.168.1.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces = ipv4.Assign (devices);

  // Print IP addresses of nodes
  for (uint32_t i = 0; i < nodes.GetN (); ++i)
  {
    Ptr<Node> node = nodes.Get (i);
    Ptr<Ipv4> ipv4 = node->GetObject<Ipv4> ();
    Ipv4Address address = ipv4->GetAddress (1, 0).GetLocal ();
    std::cout << "Node " << i + 1 << " IP address: " << address << std::endl;
  }

  // Create applications (optional)
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  // Run simulation
  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
}
```

Data Minning Lab

EXPERIMENT NO: 1

Aim:

Create an Employee Table with the help of Data Mining Tool WEKA.

Description:

We need to create an Employee Table with training data set which includes attributes like name, id, salary, experience, gender, phone number.

Procedure:

Steps:

1) Open Start → Programs → Accessories → Notepad
2) Type the following training data set with the help of Notepad for Employee Table.

```
@relation employee
@attribute name {x,y,z,a,b}
@attribute id numeric
@attribute salary {low,medium,high}
@attribute exp numeric
@attribute gender {male,female}
@attribute phone numeric

@data
x,101,low,2,male,250311
y,102,high,3,female,251665
z,103,medium,1,male,240238
a,104,low,5,female,200200
b,105,high,2,male,240240
```

3) After that the file is saved with **.arff** file format.
4) Minimize the arff file and then open Start → Programs → weka-3-4.
5) Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6) In that dialog box there are four modes, click on **explorer**.
7) Explorer shows many options. In that click on **'open file'** and select the arff file
8) Click on **edit button** which shows employee table on weka.

Viewer

Relation: employee

No.	1: name Nominal	2: id Numeric	3: salary Nominal	4: exp Numeric	5: gender Nominal	6: **phone** Numeric
1	x	101.0	low	2.0	male	250311.0
2	y	102.0	high	3.0	female	251665.0
3	z	103.0	medium	1.0	male	240238.0
4	a	104.0	low	5.0	female	200200.0
5	b	105.0	high	2.0	male	240240.0

Result:

This program has been successfully executed.

EXPERIMENT NO: 2

Aim:

Create a Weather Table with the help of Data Mining Tool WEKA.

Description:

We need to create a Weather table with training data set which includes attributes like outlook, temperature, humidity, windy, play.

Procedure:

Steps:

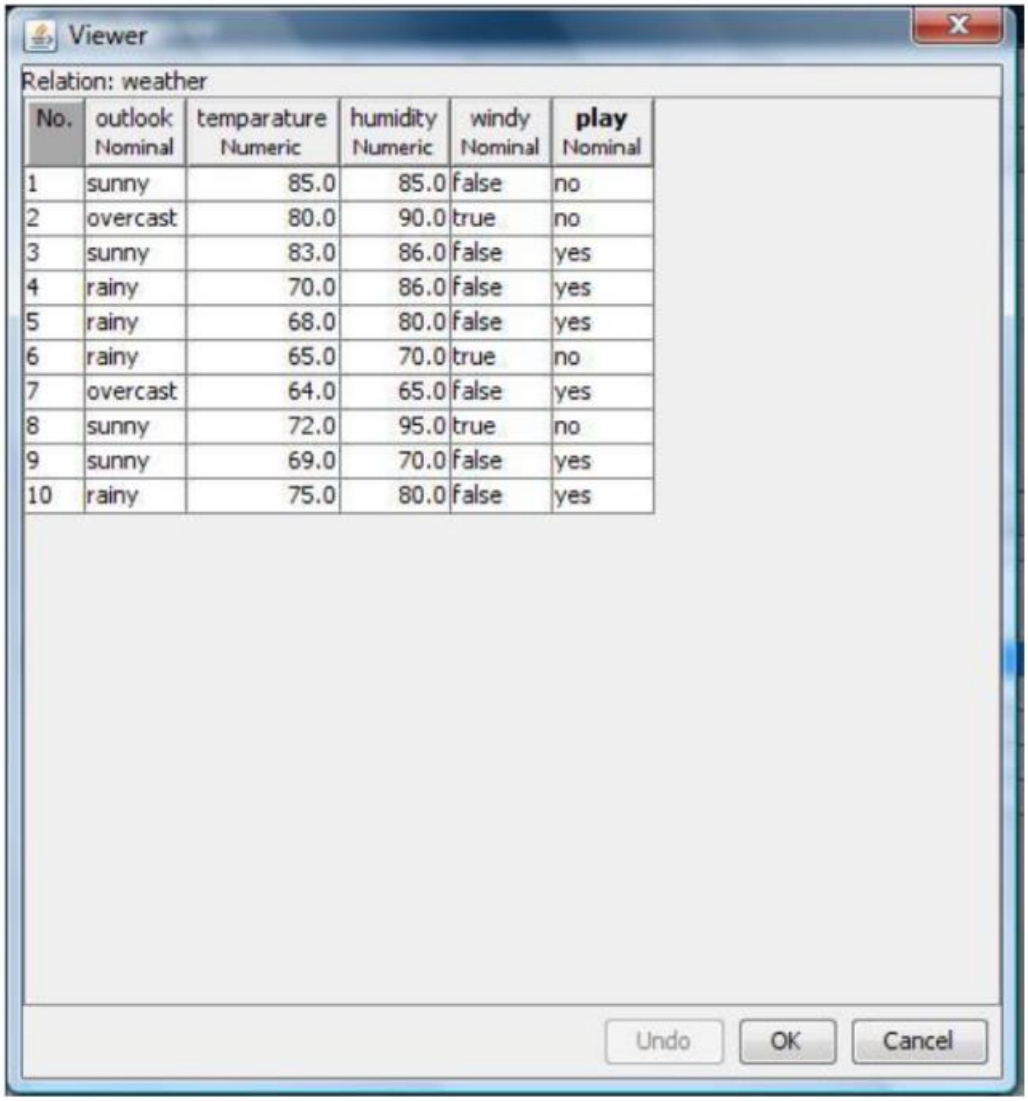
1) Open Start → Programs → Accessories → Notepad
2) Type the following training data set with the help of Notepad for Weather Table.

```
@relation weather
@attribute outlook {sunny,rainy,overcast}
@attribute temparature numeric
@attribute humidity numeric
@attribute windy {true,false}
@attribute play {yes,no}

@data
sunny,85.0,85.0,false,no
overcast,80.0,90.0,true,no
sunny,83.0,86.0,false,yes
rainy,70.0,86.0,false,yes
rainy,68.0,80.0,false,yes
rainy,65.0,70.0,true,no
overcast,64.0,65.0,false,yes
sunny,72.0,95.0,true,no
sunny,69.0,70.0,false,yes
rainy,75.0,80.0,false,yes
```

3) After that the file is saved with **.arff** file format.
4) Minimize the arff file and then open Start → Programs → weka-3-4.
5) Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6) In that dialog box there are four modes, click on **explorer**.
7) Explorer shows many options. In that click on **'open file'** and select the arff file
8) Click on **edit button** which shows weather table on weka.

Training Data Set → Weather Table



Result:

This program has been successfully executed.

EXPERIMENT NO: 3

Aim:

Apply Pre-Processing techniques to the training data set of Weather Table

Description:

Real world databases are highly influenced to noise, missing and inconsistency due to their queue size so the data can be pre-processed to improve the quality of data and missing results and it also improves the efficiency.

There are 3 pre-processing techniques they are:

1) Add
2) Remove
3) Normalization

Creation of Weather Table:

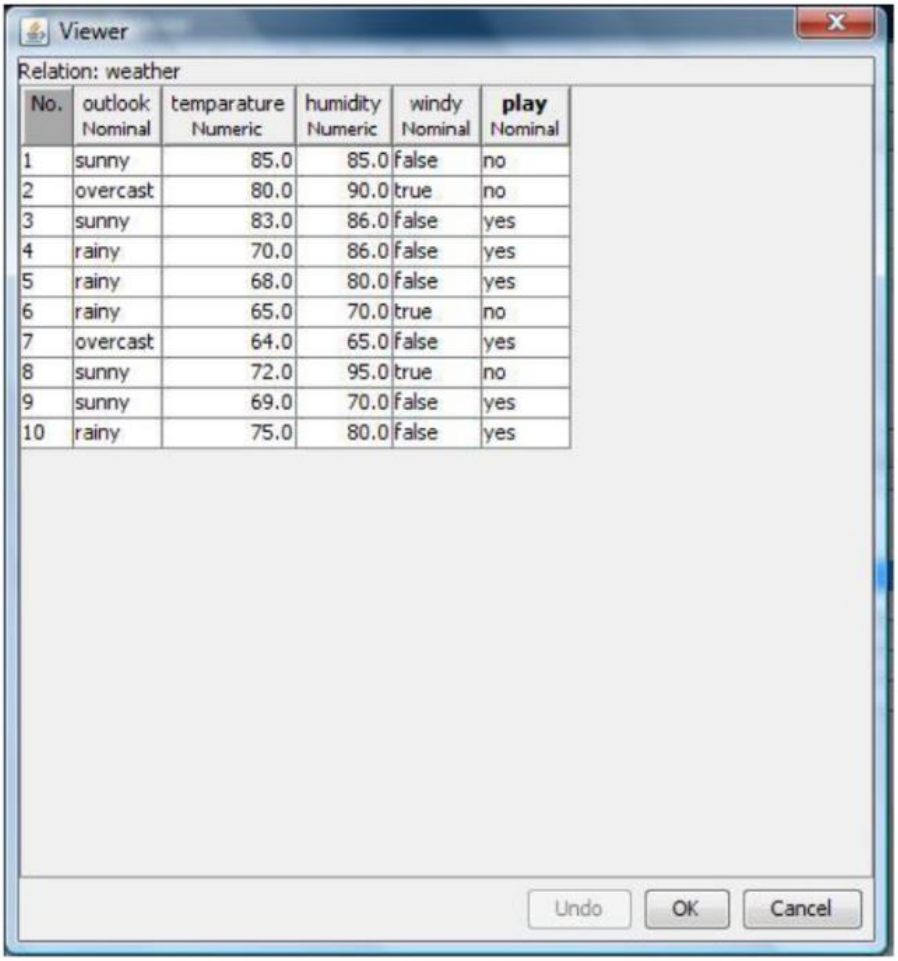
Procedure:

1) Open Start → Programs → Accessories → Notepad
2) Type the following training data set with the help of Notepad for Weather Table.

```
@relation weather
@attribute outlook {sunny,rainy,overcast}
@attribute temparature numeric
@attribute humidity numeric
@attribute windy {true,false}
@attribute play {yes,no}

@data
sunny,85.0,85.0,false,no
overcast,80.0,90.0,true,no
sunny,83.0,86.0,false,yes
rainy,70.0,86.0,false,yes
rainy,68.0,80.0,false,yes
rainy,65.0,70.0,true,no
overcast,64.0,65.0,false,yes
sunny,72.0,95.0,true,no
sunny,69.0,70.0,false,yes
rainy,75.0,80.0,false,yes
```

3) After that the file is saved with **.arff** file format.
4) Minimize the arff file and then open Start → Programs → weka-3-4.
5) Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6) In that dialog box there are four modes, click on **explorer**.
7) Explorer shows many options. In that click on **'open file'** and select the arff file
8) Click on **edit button** which shows weather table on weka.

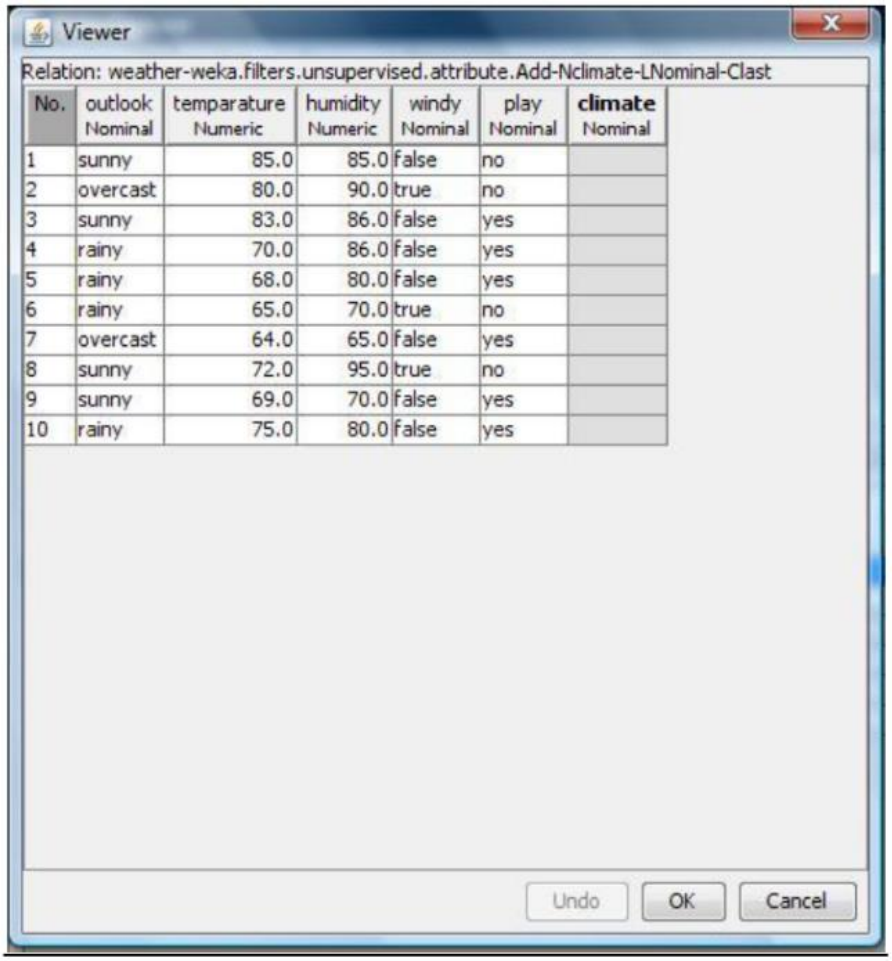


Add → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Weather.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Add**.
9) A new window is opened.
10) In that we enter attribute index, type, data format, nominal label values for **Climate**.
11) Click on **OK**.
12) Press the **Apply button**, then a new attribute is added to the Weather Table.
13) **Save** the file.
14) Click on the **Edit button**, it shows a new Weather Table on Weka.

Weather Table after adding new attribute CLIMATE:



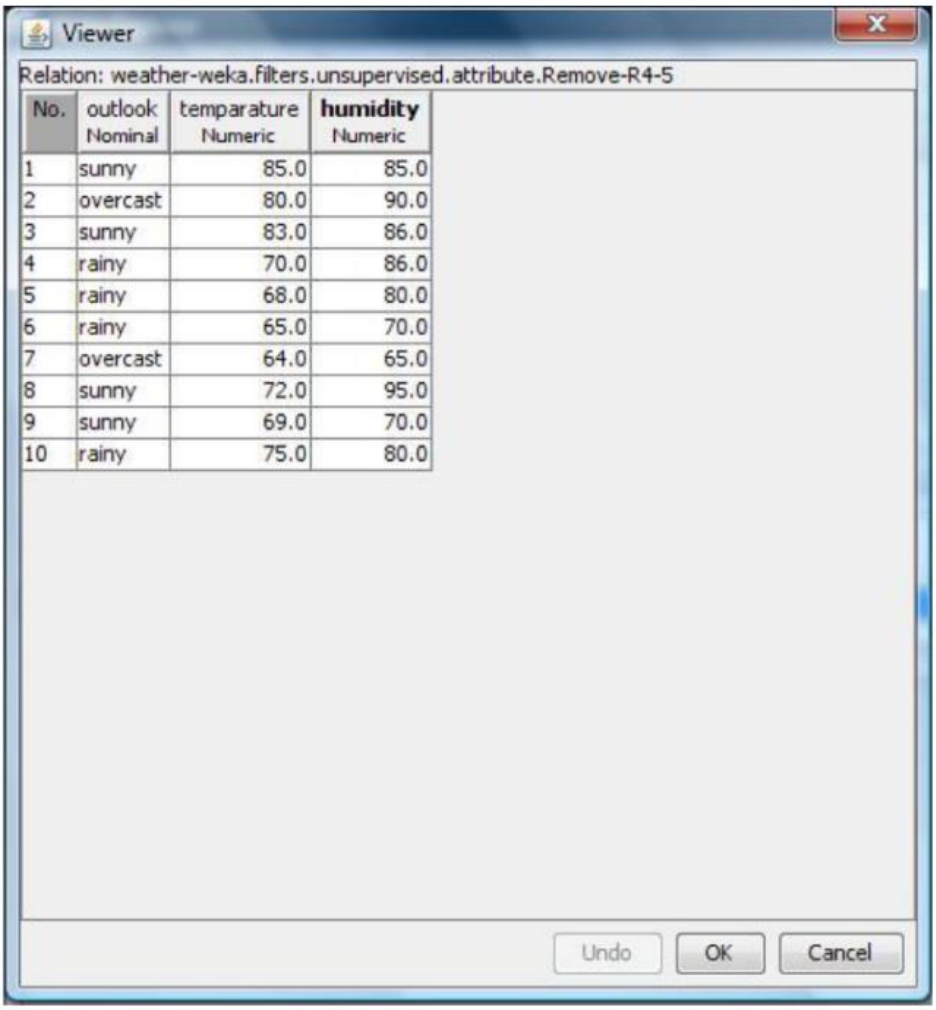
No.	outlook Nominal	temparature Numeric	humidity Numeric	windy Nominal	play Nominal	climate Nominal
1	sunny	85.0	85.0	false	no	
2	overcast	80.0	90.0	true	no	
3	sunny	83.0	86.0	false	yes	
4	rainy	70.0	86.0	false	yes	
5	rainy	68.0	80.0	false	yes	
6	rainy	65.0	70.0	true	no	
7	overcast	64.0	65.0	false	yes	
8	sunny	72.0	95.0	true	no	
9	sunny	69.0	70.0	false	yes	
10	rainy	75.0	80.0	false	yes	

Remove → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Weather.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Remove**.
9) Select the attributes **windy, play** to Remove.
10) Click **Remove button** and then **Save**.
11) Click on the **Edit button**, it shows a new Weather Table on Weka.

Weather Table after removing attributes WINDY, PLAY:



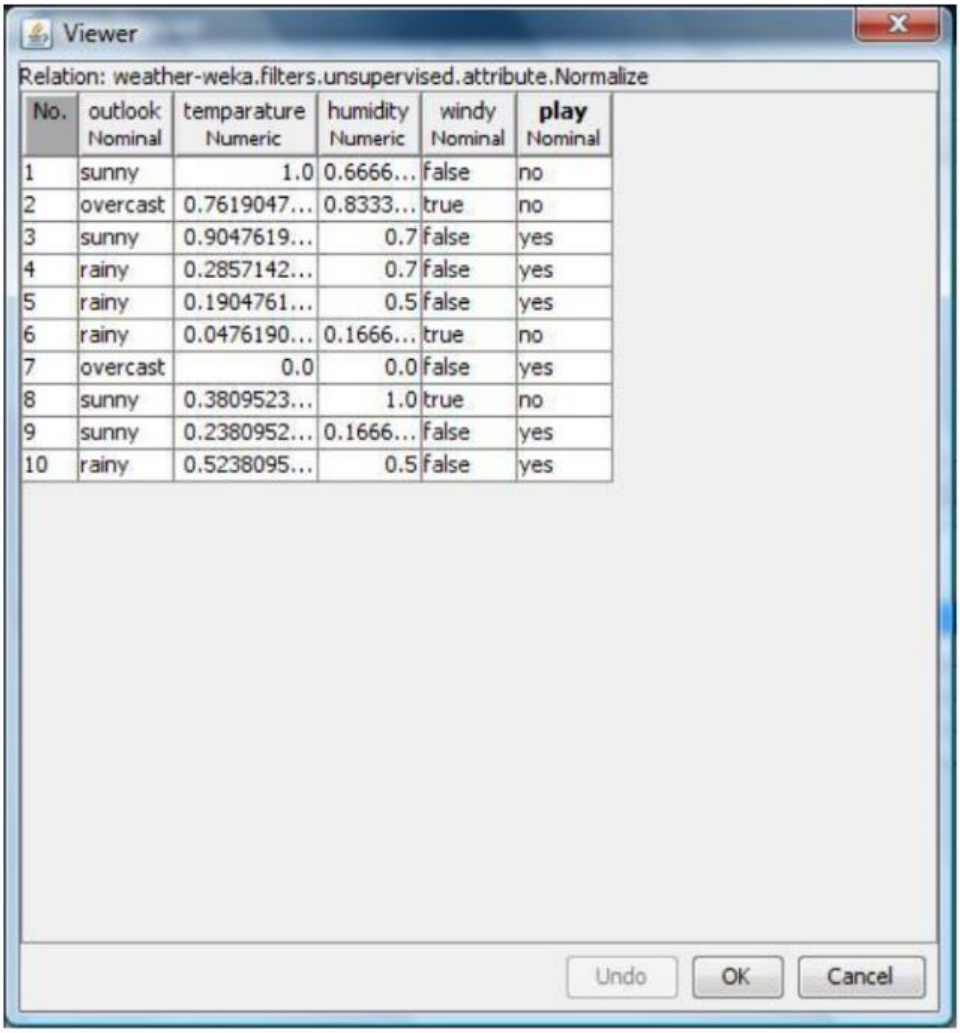
No.	outlook Nominal	temparature Numeric	**humidity** Numeric
1	sunny	85.0	85.0
2	overcast	80.0	90.0
3	sunny	83.0	86.0
4	rainy	70.0	86.0
5	rainy	68.0	80.0
6	rainy	65.0	70.0
7	overcast	64.0	65.0
8	sunny	72.0	95.0
9	sunny	69.0	70.0
10	rainy	75.0	80.0

Normalize → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Weather.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Normalize**.
9) Select the attributes **temparature, humidity** to Normalize.
10) Click on **Apply button** and then **Save**.
11) Click on the **Edit button**, it shows a new Weather Table with normalized values on Weka.

Weather Table after Normalizing TEMPARATURE, HUMIDITY:



No.	outlook Nominal	temparature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	1.0	0.6666...	false	no
2	overcast	0.7619047...	0.8333...	true	no
3	sunny	0.9047619...	0.7	false	yes
4	rainy	0.2857142...	0.7	false	yes
5	rainy	0.1904761...	0.5	false	yes
6	rainy	0.0476190...	0.1666...	true	no
7	overcast	0.0	0.0	false	yes
8	sunny	0.3809523...	1.0	true	no
9	sunny	0.2380952...	0.1666...	false	yes
10	rainy	0.5238095...	0.5	false	yes

Result:

This program has been successfully executed.

EXPERIMENT NO: 4

Aim:

Apply Pre-Processing techniques to the training data set of Employee Table

Description:

Real world databases are highly influenced to noise, missing and inconsistency due to their queue size so the data can be pre-processed to improve the quality of data and missing results and it also improves the efficiency.

There are 3 pre-processing techniques they are:

1) Add
2) Remove
3) Normalization

Creation of Employee Table:

Procedure:

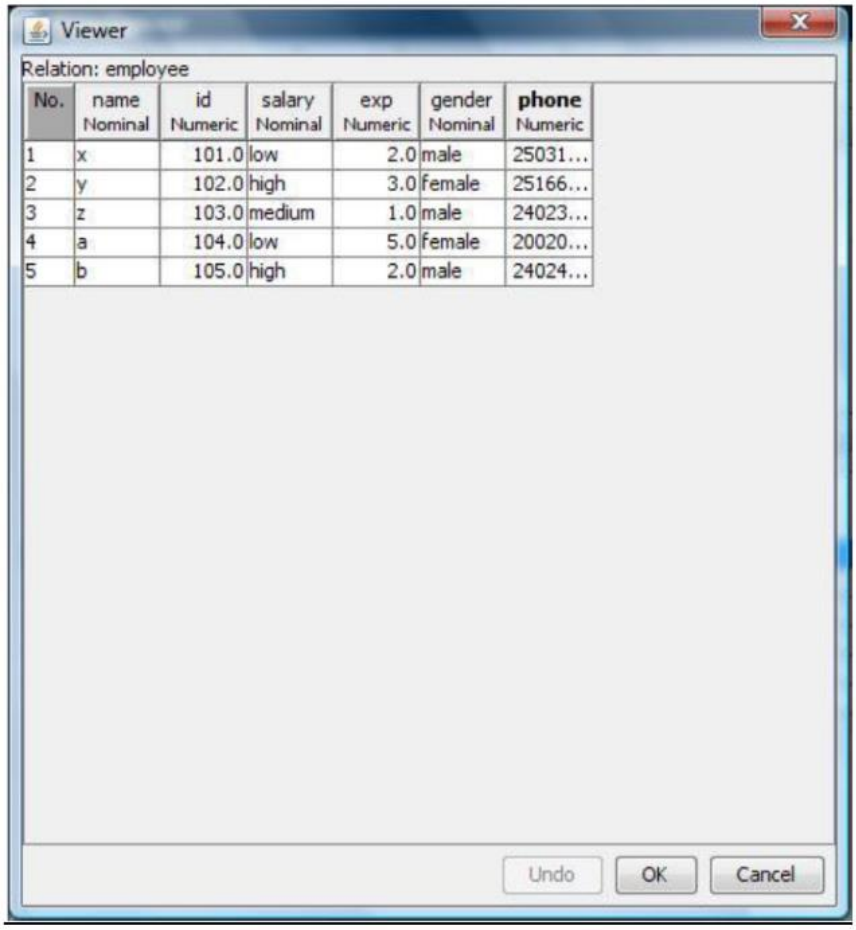
1) Open Start → Programs → Accessories → Notepad
2) Type the following training data set with the help of Notepad for Employee Table.

```
@relation employee
@attribute name {x,y,z,a,b}
@attribute id numeric
@attribute salary {low,medium,high}
@attribute exp numeric
@attribute gender {male,female}
@attribute phone numeric

@data
x,101,low,2,male,250311
y,102,high,3,female,251665
z,103,medium,1,male,240238
a,104,low,5,female,200200
b,105,high,2,male,240240
```

3) After that the file is saved with **.arff** file format.
4) Minimize the arff file and then open Start → Programs → weka-3-4.
5) Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6) In that dialog box there are four modes, click on **explorer**.
7) Explorer shows many options. In that click on **'open file'** and select the arff file
8) Click on **edit button** which shows employee table on weka.

Training Data Set → Employee Table

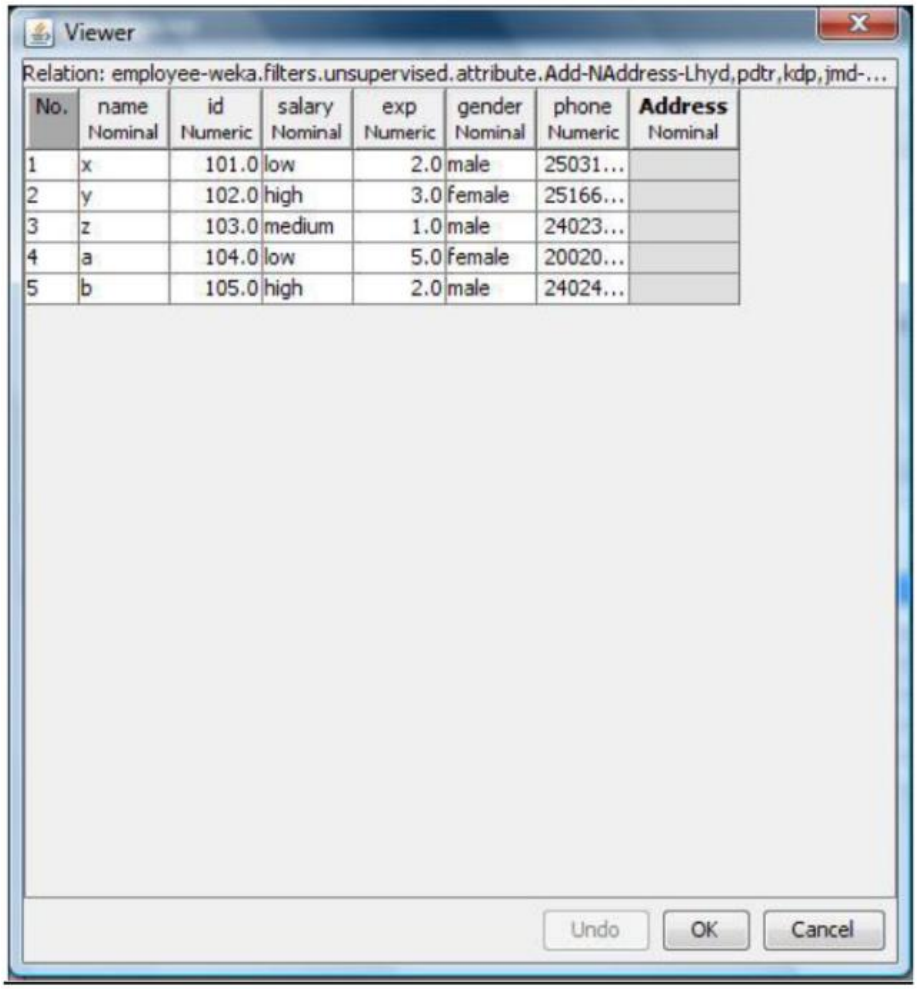


Add → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Employee.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Add**.
9) A new window is opened.
10) In that we enter attribute index, type, data format, nominal label values for **Address**.
11) Click on **OK**.
12) Press the **Apply button**, then a new attribute is added to the Employee Table.
13) **Save** the file.
14) Click on the **Edit button**, it shows a new Employee Table on Weka.

Employee Table after adding new attribute ADDRESS:

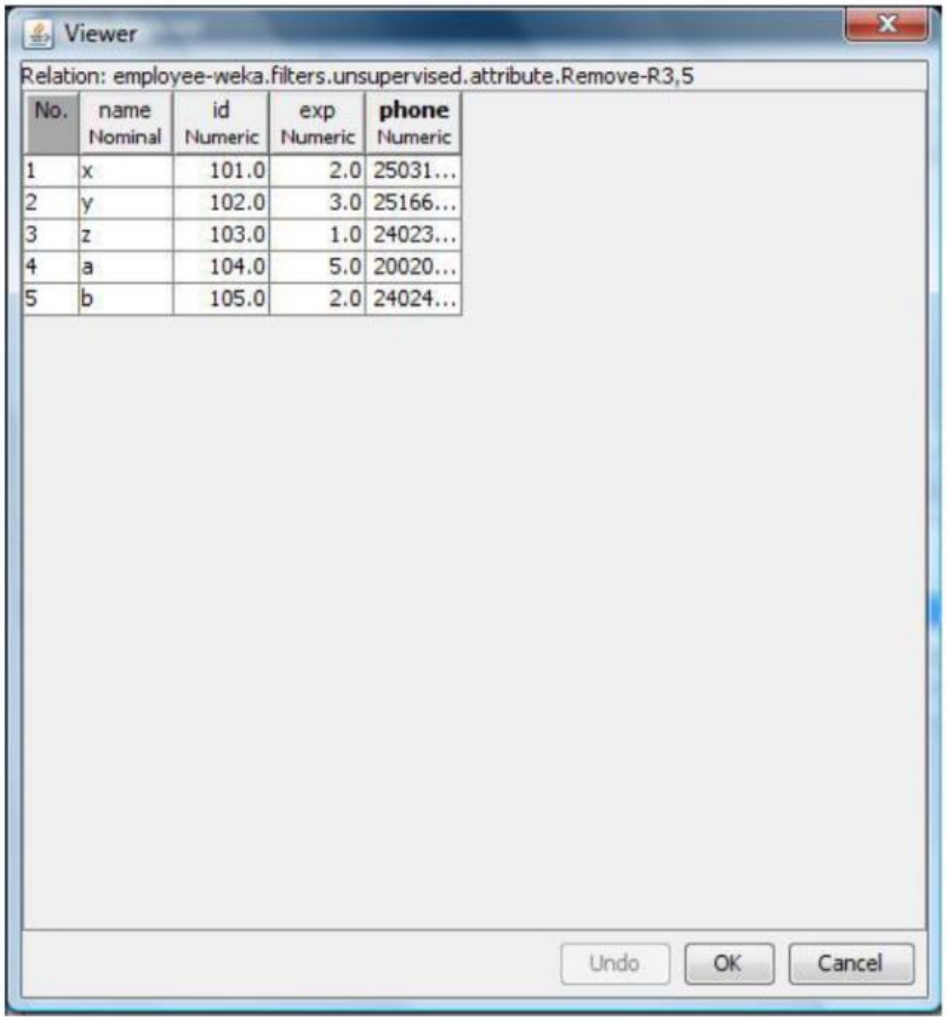


Remove → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Employee.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Remove**.
9) Select the attributes **salary, gender** to Remove.
10) Click **Remove button** and then **Save**.
11) Click on the **Edit button**, it shows a new Employee Table on Weka.

Employee Table after removing attributes SALARY, GENDER:

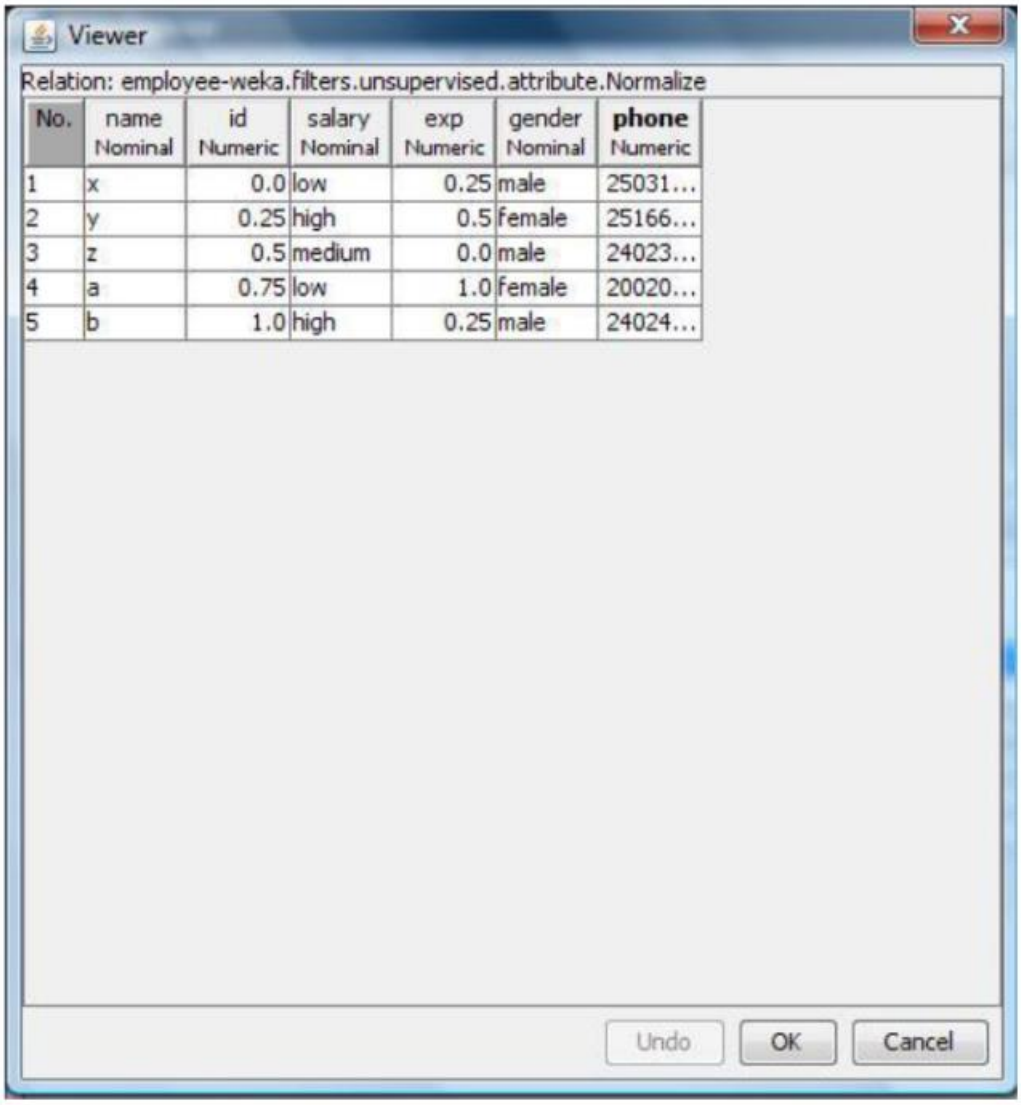


Normalize → Pre-Processing Technique:

Procedure:

1) Start → Programs → Weka-3-4 → Weka-3-4
2) Click on **explorer.**
3) Click on **open file.**
4) Select **Employee.arff** file and click on open.
5) Click on **Choose button** and select the **Filters option**.
6) In Filters, we have **Supervised** and **Unsupervised data**.
7) Click on **Unsupervised data**.
8) Select the attribute **Normalize**.
9) Select the attributes **id, experience, phone** to Normalize.
10) Click on **Apply button** and then **Save**.
11) Click on the **Edit button**, it shows a new Employee Table with normalized values on Weka.

Employee Table after Normalizing ID, EXP, PHONE:



No.	name Nominal	id Numeric	salary Nominal	exp Numeric	gender Nominal	**phone** Numeric
1	x	0.0	low	0.25	male	25031...
2	y	0.25	high	0.5	female	25166...
3	z	0.5	medium	0.0	male	24023...
4	a	0.75	low	1.0	female	20020...
5	b	1.0	high	0.25	male	24024...

Result:

This program has been successfully executed.