IJSRM
INTERNATIONAL JOURNAL OF SCIENCE AND RESEARCH METHODOLOGY
An Official Publication of Human Journals

Human Journals
Research Article
August 2018 Vol.:10, Issue:2

Association Rule Mining Unsupervised Machine Learning Apriori Algorithms for Employee Loan Application

IJSRM
INTERNATIONAL JOURNAL OF SCIENCE AND RESEARCH METHODOLOGY
An Official Publication of Human Journals

Aliyu Sani Ahmad*, Mu'awuya Dalhatu, Anas Usman Inuwa, Yaknan John Gambo

Computer Science Department, Federal University Wukari, Taraba State

Submission: 20 July 2018
Accepted: 29 July 2018
Published: 30 August 2018

HUMAN JOURNALS
www.ijsrm.humanjournals.com

Keywords: Data Mining**,** Unsupervised Machine Learning, Association Rule, Loan, Employee, Application.

ABSTRACT

Digital age has reform our decision making through data mining which is an integral part of artificial intelligence. Data mining consist analysis of big data from various sources and retrieving useful information. This information can be converted to knowledge about future and past. Apriori algorithm is one of the technique used for data mining analysis, it is an influential algorithm for decision making. In this paper, the author had developed association rule mining based Apriori algorithm capable of unsupervised machine learning for finding the relationship between employee monthly loan repayment and performance in an organisation. ASP. Net was used to implement the algorithm while Waikato Environment for Knowledge Analysis (WEKA) was used to evaluate the implemented algorithms. Result shows that the algorithm implemented in ASP.NET work well with the highest accuracy of 80% while WEKA is 40% accuracy.

1. INTRODUCTION

Data Mining plays a vital role in problem solving. It is the newest answer to increase revenues and to reduce costs. It uses a variety of data analysis tools to discover patterns and relationships in data that may be used to make valid predictions. Also is a process of discovering fascinating designs, new instructions and information from large amount of data known as big data. It is an emerging technology which involves artificial intelligence, machine learning supervised or unsupervised, information retrieval and high-performance computing. It provides important and valuable knowledge to decision makers, resulting in incalculable economic benefits. The functions of data mining are association rule analysis, cluster analysis, outlier analysis, classification, prediction, and correlations analysis etc. Association rules provide the effective scientific base for decision making. Association rule have been used in many applications to find frequent patterns in datasets. Data mining can be applied in many field to find interesting insight that can be used to achieve accurate predictions and superb decision making. One of the key domains uses association rule is business field where it helps in a very effective and efficient decision making and marketing. Other field areas where association rule mining can be applied are market basket analysis, medical diagnosis, census data, fraud detection in web and DNA data analysis etc. For example, data mining can be applied in the field of oil and gas to find the impact of oil shocks in some macroeconomics variables such as GDP, exchange rate and inflation also in the field of electrical power engineering, data mining have been used for the condition monitoring of high voltage equipment. In medical field, ARM is used to find frequently occur diseases in area and to diagnose different diseases. It is also used to attain information about the navigational activities of users in webLog data. Recently it is discovered that there are various algorithms for finding the association rules. For frequent pattern mining, different frameworks have been defined. One of the best and most commonly used algorithm is Apriori. Other procedures are Equivalence Class Transformation (ÉCLAT), Tree Projection, FP-growth, direct hashing and pruning algorithm.

This research aim at finding the association between employee performances and loan monthly repayment so the study specifically uses Apriori Association rule technique and build an algorithm capable of unsupervised machine learning to scan the loan data in the database to find the frequent pattern in loan database to determine whether loan can be

granted or refused to subsequent employee that would apply for loan based on the previous existing data of employees that applied for a loan.

2. Association Rule Technique

This is one of the algorithms used in data mining in transaction (records) databases to find, frequent patterns, associations, relations and correlation etc. Tan et al. (2004). It is often used to do market basket analysis. In general sense, Association algorithms can be applied in wide range of situations to find association, frequent pattern from the sets of objects in databases. It expresses how object or item related to each other and how they tend to group together. It can be applied to large database of one thousand to one million records (dataset) as well as small database records Tan et al. (2004). The rules analyses and predict pattern behaviour from dataset.

The rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data while consequent is an item that is found in combination with the antecedent Munchi G. (2016). The algorithms also use two popular measurements which are Support and confidence

- **Support:** is the percentage of task-relevant to data transactions for which the pattern is true.

- **Confidence:** is the measure of certainty associated with each discovered pattern.

3. Problem Statement

If employees in a workplace are to be repaying monthly loan instalment, there is a probability that repayment would affect their performance.

4. Aim and Objectives of the Study

The aim of the study is to use association rule techniques to mine the frequent pattern and association within the loan data from the database and find the relationships between employee loan monthly repayment and his or her performance. And the objectives are:

1. To develop an algorithm capable of unsupervised machine learning that can scan database to find the frequent pattern and association within the dataset.

2. To uncover patterns from the loan data stored in the database and then used to build predictive models.

3. To discover the link between dataset and find the combinations of dataset that might affect employee performance.

4. Describe the promise and potential of data mining analytics for organisations.

5. Literature Review

Oluigbo I. *et al.* demonstrate how data mining is relevant and serve as tool for organisational growth and productivity, they show that data mining is important in a way that helps an organisation to identify products that are purchased concurrently, more so, it helps organisation to find the characteristics of consumers for certain product. In fraud detection, data mining helps organisation to identify which transaction are likely to be fraudulent. Predicting the likelihood of fraudulent behaviour may cause lots of savings for financial institutions especially banks for credit card fraud or telecommunication companies for telephone call fraud. For instance, online Data Miming Models running behind operational system can swiftly identify and monitor suspicious transactions. This shows that data mining can be applied in different organisation to achieve different goals. In retail marketing, data mining assist in identifying the buying patterns of customers, predicting response, finding associations among customer demographic characteristic and in market basket analysis.

Tan, et al. (2004) shows that methodology known as Association Rule is useful for discovering interesting relationships hidden in large datasets. The uncovered relationships can be presented in the form of association rules or set of frequent items. Business enterprise accumulate large amount of data from day-to-day operations. For instance, huge amount of customer purchase data are collected daily at the checkout counters of grocery stores. Such data is commonly known as market basket transactions. Each row in this table corresponds to a transaction, which contains a unique identifier labeled TID and a set of items bought by a given customer. Retailers are interested in analyzing the data to learn about the purchasing behaviour of their customers. Such valuable information can be used to support a variety of business-ralated applications such as marketing promotions, inventory management, and customer relationship management.

Rohit *et al.* (2014), used decision tree model and classification technique, they assert that "the prediction of employee's performance with high accuracy is more beneficial and to improve their performance. The main task of their association rule mining is to find set of binary variables that frequently occurs in the transaction database. "The goal of feature selection problem is to identify groups, which is correlated with each one of the target variable". In their finding, they realized that: "Staffs who have Heavy tasks and good quality, their performance is good, and most of them are actively cultivate their skills and their initiative is perfect. Staffs who have Easy tasks and bad quality, surely their performance is poor, and they mainly don't pay attention to cultivate their working skills, their imitativeness is ordinary". They found out that, comparing classification technique with the SQL query the classification algorithm performs efficiently.

Qasem *et al.* (2012*)* Data mining is a young and promising field of information and knowledge discovery it started to be an interest target for information industry, because of the existence of huge data containing large amounts of hidden knowledge. They assert with data mining techniques, such knowledge can be extracted and accessed transforming the databases tasks from storing and retrieval to learning and extracting knowledge.

Kahya (2007) studied certain features that affect the employee job performance. The researcher reviewed previous studies, describing the effect of experience, salary, education, working conditions and job satisfaction on the performance. As a result of his research work, he has found that several factors affected the employee's performance. The position or grade of the employee in the company was of high positive effect on his/her performance. Working conditions and environment, on the other hand, has shown both positive and negative relationship on performance. Highly educated and qualified employees showed dissatisfaction of bad working conditions and thus affected their performance negatively. Employees of low qualifications, on the other hand, showed high performance in spite of the bad conditions. In addition, experience showed positive relationship in most cases, while education did not yield clear relationship with the performance.

Jantan *et al.* (2010) have used decision tree C4.5 classification algorithm to predict human talent in Human Resource Management, they did that by generating classification rules for the historical human resource records, and testing them on unseen data to calculate accuracy. They intend to use these rules in creating a DSS system (decision support system) that can be used by managements to predict employees' performance and potential promotions. This

paper is an attempt to use data mining concepts, particularly classification, to help supporting the human resources directors and decision makers by evaluating employees' data to study the main attributes that may affect the employees' performance.

Roxanne A. *et al.* (2012) in their research they adopt classification and association rule emphasizing is the most appropriate data mining functionalities for training and performance predictions. Rule-based classification model is represented as a set of IF-THEN rules. These rules are generated either from a decision tree or directly from the training data using sequential covering algorithm (SCA). An if-then rule is an expression of the form IF condition THEN conclusion in which the "If" part (left side) is the rule antecedent or precondition and "Then" part (right side) is the rule consequent. In their study, they discover that most newly-hired teachers need professional training. "The result of this study intends not to replace the HR roles to design and plan the appropriate training needs of newly-hired faculty members but to provide scientific and sound bases for selecting more trainings associated to the identified weaknesses. Specifically, the rules generated as predictors for human resource development needs will be recommended as a supplementary tool for devising strategic plans for faculty empowerment programs".

Sangita G. *et al.* (2013) postulate that due to the swift transformation of technology, software industries owe to manage a large set of data having precious information hidden. Data mining technique enables one to effectively cope with this hidden information where it can be applied to code optimization, fault prediction and other domains which modulates the success nature of software projects. The position or objective of their work is to explore potentials of project personnel in terms of their competency and skill set and its influence on quality of project. They accomplished their goal using a Bayesian classifier in order to capture the pattern of human performance. "This mode of predictive study enables the project managers to reduce the failure ratio to a significant level and improve the performance of the project using the right choice of project personnel".

6. Methodology

Constructive research methodology was adopted for this study. The constructive research approach is a typical research procedure in field of computer science for designing and implementing or evaluating a software construction intended to solve domain problems faced in real world Liisa L. et al. (2016). This study also uses another common methodology known

as prototyping. A software prototype is different from a full-scale software system; it is just a partial implementation of a system with the purpose of exploring the problem to be solved Zhangmin L. (2009). For this study, a prototype system was design and implemented using ASP.net

The research delivers the following components as a prototype of the Employee Management System:

- **CRUD Operation**: This component is responsible for adding new employee to the system, updating existing employee data, deleting etc. The attribute as input that was used are: Title, Surname, FirstName, Other names, Gender, Address, Department, Qualification, Position, Date of Birth, HireDate, Telephone, Nationality, Email, Username, Role and Password.

- **Authentication**: This is responsible for the security and user authentication. It authenticates users and handles the user management activities. This component implements the "control access privilege matrix".

- **Employee Payment**: This component was implemented to enable employee's payment management; the payment processing considers employee basic salary, deductions, incentives and bonuses

- **Performance Evaluation:** This component implements employee performance self-evaluation. The interface provide means by which employee will use to evaluate his/her performance periodically. In this component, fifteen random questions were supplied, each question has five options, and score was assign to each option, having answer fifteen question, the system will evaluate the employee. The system stores the score in the database. The score was considered and used in the Association rule analysis.

- **Loan Application** this component was implemented to provide an interface for an employee to apply for loan. When employee sign-in, the system will capture his ID, the interface requires two input (1) Amount of money needed to borrow and (2) Duration to pay the money, the duration cannot exceed twelve months (one year), and 3.30 was used as interest rate. Once an employee applies for loan, after signing in, the system will show him total payment and monthly repayment. The monthly repayment was used in the association rule analysis.

7. Experimental Setting

The research capitalizes on Loan Repayment, Duration, BasicSalary, WorkExperience and Performance Result to build the association rule algorithm.

If Loan Monthly Repayment = **A**, Basic Salary = **B**, Hire Date = **C** and Performance Result = **D**

Support and confidence **A** => **B**, **A**=> **C**, **A**=>**D**, **B**=>**C**, **B**=>**D**, **C**=>**D**

- Support denotes probability that contains A, B, C, and D
- Confidence denotes probability that a record (transaction) containing

A also contains B,

A also contains C,

A also contains D,

B also contains C,

B also contains D,

C also contains D,

The major objective is to build the prediction model based on the dataset, to identify frequent item-sets, and extract strong association rules that may indicate whether

- If employee has low amount of loan to pay back in a month, is 90% likely to perform high or
- If employee has high amount of loan to pay back in a month, is 90% likely to perform low or
- If employee has low work experience and high amount of loan is 90% likely to perform low or
- If employee have low basic salary, low work experience and high amount of loan to repay is likely to perform low etc.

8. 0 Data Collection

A random record of 140 employee's data was recorded in the database, 100 random employee's performance self-evaluation tests were taken to get their score, basic salary of 100 employees was added to their payment and 100 employees was assumed to have applied for loan. This is done by log in each employee, using his username and password and his ID were taken by session. Table 1 below indicated Loan Table data

Table 1: Partial Datasets with nine attributes

LoanId	EmpId	Amount	Duration	BasicSalary	HireDate	AppraisalResult	MonthlyPayment	TotalPayment
123	47	12000	12	10000	12/04/2016 00:0...	68	1,018	12,216
124	48	10000	12	20000	01/11/2015 00:0...	80	848	10,180
125	49	17000	12	30000	01/01/2016 00:0...	50	1,442	17,305
126	50	20000	12	40000	14/02/2010 00:0...	50	1,697	20,359
127	51	15000	12	50000	14/05/2002 00:0...	49	1,272	15,269
128	52	5000	12	60000	11/02/2015 00:0...	73	424	5,090
129	53	10000	12	70000	01/01/2010 00:0...	61	848	10,180
130	55	20000	12	20000	01/02/2015 00:0...	74	1,697	20,359
131	57	5000	12	10000	14/10/2016 00:0...	62	424	5,090
132	58	15000	12	30000	01/01/2010 00:0...	78	1,272	15,269

8.1 Analysis

8.2 Data Pre-Processing

This involved arranging, categorizing and encoding data in a way that can communicate as algorithm. This process arranged the datasets which is used for further analysis.

8.3 Data Development

Five attributes were used from Loan Table as indicated in table 2 below. The duration of work experience was gotten by subtracting current date from employee hire-date thus the column that was used are: basic-salary, work-experience, performance-result, loan-monthly-repayment and duration.

Table 2: Partial Datasets with five (5) attributes

EMP ID	BASIC SALARY	WORK EXPERIENCE	PERFORMANCE RESULT	MONTHLY REPAYMENT	DURATION
47	10000	0 Year	38%	1,018	7 Months
48	30000	1 Year	50%	848	12 Months
49	20000	3 Years	70%	1,697	10 Months
50	30000	5 Years	40%	4,500	8 Months
51	40000	4 Years	56%	848	10 Months

Each row of the above table was considered as transaction, the employee-ID is the transaction-ID, data in each attribute was categorizes into ranges how it can be encoded as indicated in table 3 below.

Table 3: Attributes Ranges

Basic Salary (£)	
10000 - 20000	VeryLow
21000 - 30000	Low
31000 - 40000	Normal
41000 - 50000	High
51000 Above	VeryHigh
Work Experience (Hire Date)	
0 Year	VeryLow
1 Year	Low
2 Years	Normal
3 Years	High
4 Years Above	VeryHigh
Performance Result	
0 – 40%	VeryLow
41 – 50%	Low
51 – 60%	Normal
61 – 70%	High
71% Above	VeryHigh
Loan Monthly Repayment	
0 - 600	VeryLow
700 – 800	Low
900 - 1000	Normal
1100 – 1200	High
1300 Above	VeryHigh
Duration of Loan Repayment	
1 –2 Months	VeryLow
2– 4 Months	Low
5 – 6 Months	Normal
7 – 8 Months	High
9 Months Above	VeryHigh

To find frequent item-sets and generate association rule, data in table 3 above was transform as indicated in table 4 below. The data in table 4 below were further encoded into the data shown in table 6 below

Table 4: Partial Raw Transaction (String)

[0]	BasicSalaryVeryLow	WorkExperienceVeryLow	PerformanceResultVeryLow	LoanRepaymentVeryLow	DurationVeryLow
[1]	BasicSalaryLow	WorkExperienceLow	PerformanceResultLow	LoanRepaymentLow	DurationLow
[2]	BasicSalaryNormal	WorkExperienceNormal	PerformanceResultNormal	LoanRepaymentNormal	DurationNormal
[3]	BasicSalaryHigh	WorkExperienceHigh	PerformanceResultHigh	LoanRepaymentHigh	DurationHigh
[4]	BasicSalaryVeryHigh	WorkExperienceVeryHigh	PerformanceResultVeryHigh	LoanRepaymentVeryHigh	DurationVeryHigh

To encode the data, the raw transaction was assigned a value as indicated in table 5 below

Table 5: Value Assignment

BasicSalary VeryLow	0
BasicSalary Low	1
BasicSalary Normal	2
BasicSalary High	3
BasicSalary Very High	4
WorkExperience VeryLow	5
WorkExperience Low	6
WorkExperience Normal	7
WorkExperience High	8
WorkExperience Very High	9
PerformanceResult VeryLow	10
PerformanceResult Low	11
PerformanceResult Normal	12
PerformanceResult High	13
PerformanceResult Very High	14
LoanMonthlyRepayment VeryLow	15
LoanMonthlyRepayment Low	16
LoanMonthlyRepayment Normal	17
LoanMonthlyRepayment High	18
LoanMonthlyRepayment Very High	19
Duration VeryLow	20
Duration Low	21
Duration Normal	22
Duration High	23
Duration Very High	24

Table 6: Partial Encoded Transaction

[0]	0	5	10	19
[1]	1	6	11	18
[2]	2	5	12	16
[3]	3	9	10	19
[4]	0	5	11	16

9. Unsupervised Machine Learning Association Rule Algorithms Development

This process demonstrate how the association rule was develop, the algorithms works as machine learning in many phases, when employee apply for loan, the system calculate his monthly repayment and total payment interest inclusive, when he click submit button, the machine learning takes place by scanning the database thereby checking the previous records in the Loan Table from the database to determine whether to grant or refuse the loan based on his request, previous records, frequent item-ets in the algorithms and the association rules that the algorithms works on. The algorithms were developed in Loan Application class. Below showcases the component of the algorithms in steps

Step 1: Setting up Data Class Data Context, declaration of variables and calculation of monthly repayment as indicated in figure 1 below

```
public partial class Loan : System.Web.UI.Page
{
    EmployeeDataClassesDataContext dc = new EmployeeDataClassesDataContext();
    int N;
    string[][] rawTransactions;
    List<int[]> transactions;
    double tmp;
    double payment;
    double term;
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnApply_Click(object sender, EventArgs e)
    {
        double Rate = 3.3;
        try
        {
            double principal = Convert.ToDouble(txtAmount.Text);
            double rate = Convert.ToDouble(Rate) / 100;
            term = Convert.ToDouble(txtDuration.Text);
            tmp = System.Math.Pow(1 + (rate / 12), term);
            payment = principal * (((rate / 12) * tmp) / (tmp - 1));
            double totalpayment = (payment * term);

             LResponse.Text = " Your Monthly Payment = ";
            LTotalPayment.Text = " Your Total Payment = ";
            txtMonthlyPayment.Text = String.Format("{0:N0}", payment);
            txtTotalPayment.Text = String.Format("{0:N0}", totalpayment);


        }
        catch (Exception)
        {
            LResponse.Text = "Error";
        }


    }
```

Figure 1: Setting up a collection of all items

Step 2: The program takes employee ID by session when employee sign-in, take his basic-salary from salary table, take hire-date from employee table and subtract current date to get work-experience.

```
protected void btnSubmit_Click(object sender, EventArgs e)
  {
      //Take data of this employee from database
      int EmployeeID = Convert.ToInt16(Session["EmployeeID"]);

      Employee employee = dc.Employees.SingleOrDefault(x => x.EmpId == EmployeeID
      Salary eSalary = dc.Salaries.SingleOrDefault(x => x.EmpId == EmployeeID);
      double eBasic = eSalary.Basic;
      int eExperience = DateTime.Now.Year - employee.HireDate.Year;
      DateTime eHireDate = employee.HireDate;
```

Figure 2: Submit button click event

Step 3: Categorizing data based on very low, Low, Normal, High and very high

```
List<int> empItems = new List<int>();

        switch (eExperience)
        {
            case 0:
                empItems.Add(5);
                break;
            case 1:
                empItems.Add(6);
                break;
            case 2:
                empItems.Add(7);
                break;
            case 3:
                empItems.Add(8);
                break;
            default:
                empItems.Add(9);
                break;
        }

        if (eBasic < 20000)
            empItems.Add(0);
        else if (eBasic < 30000)
            empItems.Add(1);
        else if (eBasic < 40000)
            empItems.Add(2);
        else if (eBasic < 50000)
            empItems.Add(3);
        else
            empItems.Add(4);

        if (payment < 600)
            empItems.Add(15);
        else if (payment < 800)
            empItems.Add(16);
        else if (payment < 1000)
            empItems.Add(17);
        else if (payment < 1200)
            empItems.Add(18);
        else
            empItems.Add(19);

        if (term <= 3)
            empItems.Add(20);
        else if (term <= 5)
            empItems.Add(21);
        else if (term <= 7)
            empItems.Add(22);
        else if (term <= 9)
            empItems.Add(23);
        else
            empItems.Add(24);
```

Figure 3: Data categorization

Step 4: Setting up confidence and support values

```
double minSupportPct = 0.1;
double minConfident = 0.4;
int minItemSetLength = 2;
int maxItemSetLength = 4;
```

Figure 4: confidence and support values

Step 5: Here the program will take/check history of previous employee that has applied for loan

```
try
        {

            Response.Output.WriteLine("\nBegin frequent item-set extraction \n");

            string[] rawItems = new string[] { "BasicSalaryVeryLow", "BasicSalaryLow
", "BasicSalaryNormal ", "BasicSalaryHigh", "BasicSalaryVeryHigh",
"WorkExperienceVeryLow", "WorkExperienceLow ", "WorkExperienceNormal ",
"WorkExperienceHigh", "WorkExperienceVeryHigh", "PerformanceResultVeryLow",
"PerformanceResultLow ", "PerformanceResultNormal ", "PerformanceResultHigh",
"PerformanceResultVeryHigh", "LoanRepayementVeryLow", "LoanRepayementLow ",
"LoanRepayementNormal ", "LoanRepayementHigh", "LoanRepayementVeryHigh",
"DurationVeryLow", "DurationLow", "DurationNormal", "DurationHigh", "DurationVeryHigh"
};

            N = rawItems.Length; // total number of items to deal with ( [0..11] )

            var result = from L in dc.LoanTables
                         join P in dc.performanceAppraisals on L.EmpId equals P.EmpId
                         join S in dc.Salaries on L.EmpId equals S.EmpId
                         join E in dc.Employees on L.EmpId equals E.EmpId
                         select new { L.EmpId, L.Amount, L.Duration, L.MonthlyPayment,
P.Result, S.Basic, E.HireDate };

            int temp = 0;
            string Salary = null;
            string PResult = null;
            string MonthlyPayment = null;
            string WorkExperience = string.Empty;
            string Duration = null;
```

Figure 5: Checking History of Previous Data

Step 6: Categorizing data of other employees who were granted loan previously

```
foreach (var row in result)
            {
                switch (DateTime.Now.Year - row.HireDate.Year)
                {
                    case 0:
                        WorkExperience = "WorkExperienceVeryLow";
                        break;
                    case 1:
                        WorkExperience = "WorkExperienceLow";
                        break;
                    case 2:
                        WorkExperience = "WorkExperienceNormal";
                        break;
                    case 3:
                        WorkExperience = "WorkExperienceHigh";
                        break;
                    default:
                        WorkExperience = "WorkExperienceVeryHigh";
                        break;
                }

                //Salary
                temp = Convert.ToInt32(row.Basic);
                if (temp < 20000)
                    Salary = "BasicSalaryVeryLow";
                else if (temp < 30000)
                    Salary = "BasicSalaryLow";
                else if (temp < 40000)
                    Salary = "BasicSalaryNormal";
                else if (temp < 50000)
                    Salary = "BasicSalaryHigh";
                else
                    Salary = "BasicSalaryVeryHigh";
```

```
temp = Convert.ToInt32(row.Result);
                if (temp < 40)
                    PResult = "PerformanceResultVeryLow";
                else if (temp < 50)
                    PResult = "PerformanceResultLow";
                else if (temp < 60)
                    PResult = "PerformanceResultNormal";
                else if (temp < 70)
                    PResult = "PerformanceResultHigh";
                else
                    PResult = "PerformanceResultVeryHigh";

                temp = Convert.ToInt32(row.MonthlyPayment);
                if (temp < 600)
                    MonthlyPayment = "LoanRepayementVeryLow";
                else if (temp < 800)
                    MonthlyPayment = "LoanRepayementLow";
                else if (temp < 1000)
                    MonthlyPayment = "LoanRepayementNormal";
                else if (temp < 1200)
                    MonthlyPayment = "LoanRepayementHigh";
                else
                    MonthlyPayment = "LoanRepayementVeryHigh";

                temp = Convert.ToInt32(term);
                if (temp <= 3)
                    Duration = "DurationVeryLow";
                else if (temp <= 5)
                    Duration = "DurationLow";
                else if (temp <= 7)
                    Duration = "DurationNormal";
                else if (temp <= 9)
                    Duration = "DurationHigh";
                else
```

```
                    MonthlyPayment = "LoanRepayementVeryHigh";
                    rawTransactions[index] = new string[] { Salary,
  WorkExperience, PResult, MonthlyPayment, Duration };
                index++;
            }

            for (int i = 0; i < rawTransactions.Length; ++i)
            {
                Response.Output.WriteLine("[" + i + "] : ");
                for (int j = 0; j < rawTransactions[i].Length; ++j)
                    Response.Output.WriteLine(rawTransactions[i][j] + "    ");
                Response.Output.WriteLine("");
            }


            transactions = new List<int[]>();
            int[] TransInt;
            index = 0;
```

```
foreach (string[] Trans in rawTransactions)
{
    index = 0;
    TransInt = new int[5];
    foreach (string item in Trans)
    {
        switch (item)
        {
            case "BasicSalaryVeryLow":
                TransInt[index] = 0;
                break;
            case "BasicSalaryLow":
                TransInt[index] = 1;
                break;
            case "BasicSalaryNormal":
                TransInt[index] = 2;
                break;
            case "BasicSalaryHigh":
                TransInt[index] = 3;
                break;
            case "BasicSalaryVeryHigh":
                TransInt[index] = 4;
                break;

            case "WorkExperienceVeryLow":
                TransInt[index] = 5;
                break;
            case "WorkExperienceLow":
                TransInt[index] = 6;
                break;
            case "WorkExperienceNormal":
                TransInt[index] = 7;
                break;
            case "WorkExperienceHigh":
                TransInt[index] = 8;
                break;
            case "WorkExperienceVeryHigh":
                TransInt[index] = 9;
                break;
            case "PerformanceResultVeryLow":
                TransInt[index] = 10;
                break;
            case "PerformanceResultLow":
                TransInt[index] = 11;
                break;
            case "PerformanceResultNormal":
                TransInt[index] = 12;
                break;
            case "PerformanceResultHigh":
                TransInt[index] = 13;
                break;
            case "PerformanceResultVeryHigh":
                TransInt[index] = 14;
                break;
            case "LoanRepayementVeryLow":
                TransInt[index] = 15;
                break;
            case "LoanRepayementLow":
                TransInt[index] = 16;
                break;
            case "LoanRepayementNormal":
```

Figure 6: Previous Data categorization

Step 7: Find Item-Set that satisfy or pass support value by the call of GetFrequentItemSets method

```
    List<ItemSet> frequentItemSets = GetFrequentItemSets(N, transactions,
minSupportPct, minItemSetLength, maxItemSetLength);
```

Figure 7: GetFrequentItemSets method

Step 8: The Algorithm find the rule that satisfy confident value with call of GetRules method

```
List<Rule> rules = GetRules(frequentItemSets);
        List<Rule> refinedRules = RefineTheRule(rules, minConfident);
```

Figure 8: GetRules method

Step 9: Find interesting rule (find rules with items)

```
List<Rule> valuableRules = FindRulesWithParticularItems(refinedRules, new int[]
    { 10, 11, 12, 13, 14 });
   SortRulesFollowingConfident(valuableRules);
```

Figure 9: FindRulesWithParticularItems method

Step 10: Consider employee loan application based on the interesting rule

```
string statement = string.Empty;
            bool isGranted = true;
            double round;
            int reason = 0;
            foreach (Rule tempRule in valuableRules)
            {
                if (tempRule.AffectingSet.isBelong(empItems))
                {
                if ((tempRule.AffectedSet[0] == 10)||(tempRule.AffectedSet[0] == 11))
                    {
                        isGranted = false;
                        if (reason > 0)
                        {
                            statement += ", ";
                        }

                        for (int i = 0; i < tempRule.AffectingSet.quantity; i++)
                        {
                            if (i > 0)
                                statement += " and ";
                            statement += sConvert(tempRule.AffectingSet[i]);
                        }
                        statement += " ==> ";
                        statement += sConvert(tempRule.AffectedSet[0]);
                        statement += " : the likeliness: ";
                        round = Math.Round(tempRule.Confident, 2);
                        statement += round.ToString();
                        reason++;
                    }
                }
            }
            if (isGranted == true)
                MsgBox("Your loan is accepted", this.Page, this);
            else
                if (isGranted == false)
                    MsgBox(statement, this.Page, this);
        }


        catch (Exception ex)
        {
            Response.Output.WriteLine(ex.Message);
            Console.ReadLine();
        }
    }
```

Figure 10: Considering Loan

Step 11: Instead of specifying an input parameter for the support threshold as a percentage of transactions, an absolute count was used. so, three important collections are instantiated as indicated in figure 10 below

```
Dictionary<int, bool> frequentDict = new Dictionary<int, bool>();
List<ItemSet> frequentList = new List<ItemSet>();
List<int> validItems = new List<int>();
```

Figure 11: Instantiation of three important collection

Step 12: Get counts of all individual items, the individual item values in the transaction list are counted as shown in figure 11 below

```
int[] counts = new int[N]; // index is the item/value, cell content is the count
        for (int i = 0; i < transactions.Count; ++i)
        {
            for (int j = 0; j < transactions[i].Length; ++j)
            {
                int v = transactions[i][j];
                ++counts[v];
            }
        }
```

Figure 12: count individual item

Step 13: Item values that meet the minimum support count are used to create frequent ItemSet objects of size k = 1, which are then added to the list of frequent items as figure 12 below shown

```
        for (int i = 0; i < counts.Length; ++i)
            {
                if (counts[i] >= minSupportCount)
                {
                    validItems.Add(i);
                    int[] d = new int[1];
                    d[0] = i;
                    ItemSet ci = new ItemSet(N, d, 1);
                    frequentList.Add(ci);
                    frequentDict.Add(ci.hashValue, true);
                }
            }
```

Figure 13: frequent ItemSet objects of size k = 1

Step 14: The main processing loop is set up as in figure 13 below

```
bool done = false;
            for (int k = 2; k <= maxItemSetLength && done == false; ++k)
            {
                done = true;
                int numFreq = frequentList.Count;
```

Figure 14: Main processing loop

Step 15: The main loop will exit when all specified sizes have been examined, or when no new item-sets of the current size are found. Because all frequent item-sets are stored in a single list, the initial size of the list is stored for use by the inner loops, which are set up as shown in figure 14 below

```
for (int i = 0; i < numFreq; ++i)
                {
                    if (frequentList[i].k != k - 1) continue;

                    for (int j = 0; j < validItems.Count; ++j)
                    {
                        int[] newData = new int[k];
```

Figure 15: inner loops

Step 16: Two important features of the algorithm is the algorithm uses only frequent item-sets from the previous iteration to construct new candidate item-sets and it examines only valid item values to complete the candidates. The candidate frequent item-sets are created as shown in figure 15 below

```
for (int p = 0; p < k - 1; ++p)
                        newData[p] = frequentList[i].data[p];

                    if (validItems[j] <= newData[k - 2]) continue;

                    newData[k - 1] = validItems[j];
                    ItemSet ci = new ItemSet(N, newData, -1);
```

Figure 16: creation of candidate frequent item-sets

Step 17: After all candidates for the current size k have been constructed and examined, the list of valid items for the next iteration is updated, as shown in figure 16**.**

```
validItems.Clear();
                Dictionary<int, bool> validDict = new Dictionary<int, bool>();
                for (int idx = 0; idx < frequentList.Count; ++idx)
                {
                    if (frequentList[idx].k != k) continue;
                    for (int j = 0; j < frequentList[idx].data.Length; ++j)
                    {
                        int v = frequentList[idx].data[j];
                        if (validDict.ContainsKey(v) == false)
                        {
                         validItems.Add(v);
                            validDict.Add(v, true);
                        }
                    }
                }
                validItems.Sort();
            }
```

Figure 17: Updating the list of valid items

Step 18: Update process is time-consuming, and better performance can be gotten by skipping it and instead using the original list of valid items created for size k = 1 as defined in figure 17 below by filtering the results with minimum item-set length:

```
            List<ItemSet> result = new List<ItemSet>();
            for (int i = 0; i < frequentList.Count; ++i)
            {
                if (frequentList[i].k >= minItemSetLength)
                    result.Add(new ItemSet(frequentList[i].N, frequentList[i].data
frequentList[i].ct));
            }

            return result;
        }
```

Figure 18: Filtering the results with minimum item-set length:

Step 19: The model finds the rules that have Affected Item-set being sought

```
private List<Rule> FindRulesWithParticularAffectedItemset(List<Rule> rules, ItemSe
set)
    {
        List<Rule> result = new System.Collections.Generic.List<Rule>();
        for (int i = 0; i < rules.Count; i++)
            if (rules[i].AffectedSet.isEqual(set))
                result.Add(rules[i]);

        if (result.Count >= 1)
            return result;
        else return null;
    }
    private List<Rule> FindRulesWithParticularItems(List<Rule> rules, int[] items)
    {
        List<Rule> result = new System.Collections.Generic.List<Rule>();

        for (int i = 0; i < rules.Count; i++)
        {
            if (rules[i].AffectedSet.quantity > 1)
                continue;
            for (int j = 0; j < items.Length; j++)
                if (rules[i].AffectedSet.isContain(items[j]))
                {
                    result.Add(rules[i]);
                    break;
                }
        }
        if (result.Count >= 1)
            return result;
        else return null;
    }
```

Figure 19: AffectedItemset being sought

Step 20: Sorting rule following confident value

```
private void SortRulesFollowingConfident(List<Rule> rules)
    {
        Rule temp;
        for (int i = 0; i < rules.Count - 1; i++)
            for (int j = i + 1; j < rules.Count; j++)
                if (rules[i].Confident < rules[j].Confident)
                {
                    temp = rules[i];
                    rules[i] = rules[j];
                    rules[j] = temp;
                }

    }
```

Figure 20: Sorting rule

Step 21: Find rules from a list of ItemSet that satisfy the support value

```
private List<Rule> GetRules(List<ItemSet> itemset)
    {
        List<Rule> rules = new List<Rule>();
        for (int i = 0; i < itemset.Count; i++)
        {
            List<Rule> ruleTemp = FindRulesFromAnItemset(itemset[i]);
            for (int j = 0; j < ruleTemp.Count; j++)
                rules.Add(ruleTemp[j]);
        }
        return rules;

    }
```

Figure 21: Satisfy support value

Step 22: Find rules from an Itemset

```
private List<Rule> FindRulesFromAnItemset(ItemSet set)
    {
        List<Rule> result = new List<Rule>();
        List<ItemSet> subItemset = new List<ItemSet>();
        int[] items = new int[set.quantity];
        Rule newRule;
        int ct;
        //List<int[]> allItems = new List<int[]>();

        for (int i = 0; i < set.quantity; i++)
        {
            items[i] = set[i];
            ItemSet newItemset = new ItemSet(new int[] { set[i] });
            //allItems.Add(items);
            subItemset.Add(newItemset);
            ct = CountTimesInTransactions(newItemset, transactions);
            newItemset.ct = ct;
            // Insert some first rules
            newRule = CreateARule(set, newItemset);
            result.Add(newRule);
        }
        if (set.quantity > 2)
        {
            for (int k = 2; k < set.quantity; k++)
            {
                for (int numOfSub = 0; numOfSub < subItemset.Count; numOfSub++)
                {
                    if (subItemset[numOfSub].quantity != k - 1)
                        continue;

                    for (int numOfItems = 0; numOfItems < items.Length; numOfItems++)
                    {
                        if (items[numOfItems] <= subItemset[numOfSub][k - 1 - 1])
                            continue;

                        //now create new subItemset
                        int[] newData = new int[k];
                        for (int i = 0; i < k - 1; i++)
                            newData[i] = subItemset[numOfSub][i];
                        newData[k - 1] = items[numOfItems];
                        ItemSet newSubItemset = new ItemSet(newData);
                         //Add new subItemset just created into the subItemset
                     collection.
                        subItemset.Add(newSubItemset);

                        ct = CountTimesInTransactions(newSubItemset, transactions);
                        newSubItemset.ct = ct;

                        //Find a new rule
                        newRule = CreateARule(set, newSubItemset);
                        result.Add(newRule);
                    }
                }

            }
        }

        return result;
    }
```

Figure 22: Find rules from item set

Step 23: Creating rule

```
private Rule CreateARule(ItemSet WholeSet, ItemSet AffectingSet)
    {
        ItemSet affectedItemset = FindAffectedItemset(WholeSet, AffectingSet);
        double conf = (double)WholeSet.ct / (double)AffectingSet.ct;
        Rule newRule = new Rule(AffectingSet, affectedItemset, conf);
        return newRule;
    }
```

Figure 23: Create a rule

Step 24: Finding interesting rule

```
private List<Rule> RefineTheRule(List<Rule> rules, double conf)
    {
        List<Rule> refinedRules = new System.Collections.Generic.List<Rule>();

        for (int i = 0; i < rules.Count; i++)
            if (rules[i].Confident >= conf)
                refinedRules.Add(rules[i]);

        return refinedRules;
    }
```

Figure 24: find interesting rule

Step 25: Finding affected Itemset

```
private ItemSet FindAffectedItemset(ItemSet WholeSet, ItemSet AffectingSet)
    {
        int[] AffectedItems = new int[WholeSet.quantity - AffectingSet.quantity];
        int index = 0;
        for (int i = 0; i < WholeSet.quantity; i++)
            if (!AffectingSet.isContain(WholeSet[i]))
            {
                AffectedItems[index] = WholeSet[i];
                index++;
            }
        ItemSet AffectedItemset = new ItemSet(AffectedItems);
        int ct = CountTimesInTransactions(AffectedItemset, transactions);
        AffectedItemset.ct = ct;

        return AffectedItemset;
    }
```

Figure 25: Find the affected Itemset

Step 26: Count the time of an Itemset in every transaction

```
static int CountTimesInTransactions(ItemSet itemSet, List<int[]> transactions)
    {
        int ct = 0;
        for (int i = 0; i < transactions.Count; ++i)
        {
            if (itemSet.IsSubsetOf(transactions[i]) == true)
                ++ct;
        }
        return ct;       // number of times itemSet occurs in transactions
    }
```

Figure 26: counting itemset time in transaction

Last Step: Convert item from numeric value to string value.

```
private string sConvert(int item)
    {
        switch (item)
        {
            case 0:
                return "Basic Salary Very Low";
            case 1:
                return "Basic Salary Low";

            case 2:
                return "Basic Salary Normal";

            case 3:
                return "Basic Salary High";

            case 4:
                return "Basic Salary Very High";


            case 5:
                return "Work Experience Very Low";

            case 6:
                return "Work Experience Low";

            case 7:
                return "Work Experience Normal";

            case 8:
                return "Work Experience High";

            case 9:
                return "Work Experience Very High";

            case 10:
                return "PerformanceResult Very Low";
```

```
            case 11:
                return "Performance Result Low";

             case 12:
                return "Performance Result Normal";

             case 13:
                return "Performance Result High";

             case 14:
                return "Performanc eResult VeryHigh";

             case 15:
                return "Loan Repayement Very Low";

             case 16:
                return "LoanRepayementLow";

             case 17:
                return "LoanRepayementNormal";

             case 18:
                return "LoanRepayementHigh";

             case 19:
                return "LoanRepayementVeryHigh";

             case 20:
                return "Duration Very Short";

             case 21:
                return "Duration Short";

             case 22:
                return "Duration Normal";

             case 23:
                return "Duration Long";

             case 24:
                return "Duration Very Long";

             //-----------------------------------------
             default:
                return string.Empty;
        }
    }
    //----------------------------------------
    public void MsgBox(String ex, Page pg, Object obj)
    {
        string s = "<SCRIPT language='javascript'>alert('" + ex.Replace("\r\n",
"\\n").Replace("'", "") + "'); </SCRIPT>";
        Type cstype = obj.GetType();
        ClientScriptManager cs = pg.ClientScript;
        cs.RegisterClientScriptBlock(cstype, s, s.ToString());
    }
}
```

Figure 27: numeric to string conversion

Expected Results

The console application realised the output in both numeric and string form as shown in figure 28, 29 and Table 7 and 8 below

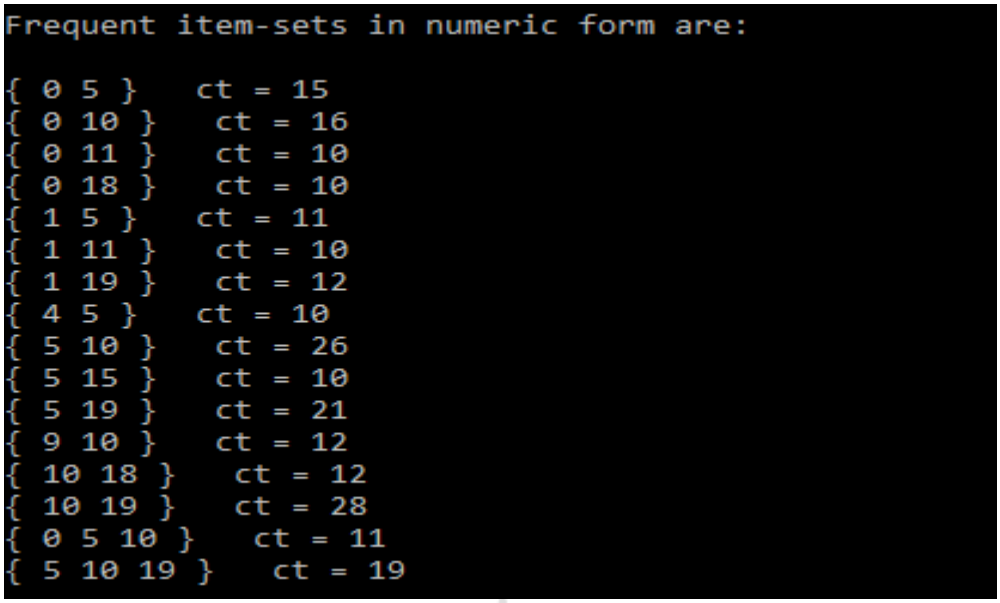


Figure 28 Frequent item-sets in numeric form

Table 7: Frequent item set in numeric form

{0 5}	ct = 15
{0 10}	ct = 16
{0 11}	ct = 10
{0 18}	ct = 10
{1 5}	ct = 11
{1 11}	ct = 10
{1 19}	ct = 12
{4 5}	ct = 10
{5 10}	ct = 26
{5 15}	ct = 10
{5 19}	ct = 21
{9 10}	ct = 12
{10 18}	ct = 12
{10 19}	ct = 28
{0 5 10}	ct = 11
{5 10 19}	ct = 19

```
Frequent item-sets in string form are:

BasicSalaryVeryLow   WorkExperienceVeryLow
BasicSalaryVeryLow   PerformanceResultVeryLow
BasicSalaryVeryLow   PerformanceResultLow
BasicSalaryVeryLow   LoanRepayementHigh
BasicSalaryLow    WorkExperienceVeryLow
BasicSalaryLow    PerformanceResultLow
BasicSalaryLow    LoanRepayementVeryHigh
BasicSalaryVeryHigh   WorkExperienceVeryLow
WorkExperienceVeryLow   PerformanceResultVeryLow
WorkExperienceVeryLow   LoanRepayementVeryLow
WorkExperienceVeryLow   LoanRepayementVeryHigh
WorkExperienceVeryHigh   PerformanceResultVeryLow
PerformanceResultVeryLow   LoanRepayementHigh
PerformanceResultVeryLow   LoanRepayementVeryHigh
BasicSalaryVeryLow   WorkExperienceVeryLow   PerformanceResultVeryLow
WorkExperienceVeryLow   PerformanceResultVeryLow   LoanRepayementVeryHigh
```

Figure 29 Frequent item-sets in string form

Table 8: Frequent item-sets in string form

Basic Salary Very Low	WorkExperience Very Low
Basic Salary Very Low	PerformanceResult Very Low
Basic Salary Very Low	PerformanceResult Low
Basic Salary Very Low	LoanRepayment High
Basic Salary Low	WorkExperience Very Low
Basic Salary Low	PerformanceResult Low
Basic Salary Low	LoanRepayment VeryHigh
Basic Salary VeryHigh	WorkExperience Very Low
Work Experience Very Low	PerformanceResult Very Low
Work Experience Very Low	LoanRepayment VeryLow
Work Experience Very Low	LoanRepayment Very High
Work Experience VeryHigh	PerformanceResult Very Low
PerformanceResult Very Low	LoanRepayment High
PerformanceResult Very Low	LoanRepayment VeryHigh
Basic Salary VeryHigh	Work Experience Very Low PerformanceResult Very Low
Work Experience Very Low	PerformanceResult Very Low LoanRepayment Very High

10. Generating Strong Association Rules from the Frequent Item-Sets

To generate strong association rule, the frequent item-sets generated in table 8 above was used to calculate the confidence (certainty associated with each discovered pattern.) of all the frequent item-sets as indicated in Table 9 below.

Table 9: Calculations of confidence of the frequent item-sets

Frequent Item-sets			Confidence	Remark
Basic Salary Very Low	WorkExperience Very Low		15/120*100 = 15%	Rejected
Basic Salary Very Low	PerformanceResult Very Low		16/100*100 = 16%	Accepted
Basic Salary Very Low	PerformanceResult Low		10/100*100 = 10%	Rejected
Basic Salary Very Low	LoanRepayment High		10/100*100 = 10%	Rejected
Basic Salary Low	WorkExperience Very Low		11/100*100 = 11%	Rejected
Basic Salary Low	PerformanceResult Low		10/100*100 = 10%	Rejected
Basic Salary Low	LoanRepayment VeryHigh		12/100*100 = 12%	Rejected
Basic Salary VeryHigh	WorkExperience Very Low		10/100*100 = 10%	Rejected
Work Experience Very Low	PerformanceResult Very Low		26/100*100 = 26%	Accepted
Work Experience Very Low	LoanRepayment VeryLow		10/100*100 = 10%	Rejected
Work Experience Very Low	LoanRepayment Very High		21/100*100 = 21%	Accepted
Work Experience VeryHigh	PerformanceResult Very Low		12/100*100 = 12%	Rejected
PerformanceResult Very Low	LoanRepayment High		12/100*100 = 12%	Rejected
PerformanceResult Very Low	LoanRepayment VeryHigh		28/100*100 = 28%	Accepted
Basic Salary VeryHigh	Work Experience Very Low	PerformanceResult Very Low	11/100*100 = 11%	Rejected
Work Experience Very Low	PerformanceResult Very Low	LoanRepayment Very High	19/100*100 = 19%	Accepted

The percentages of all the frequent item sets are shown in the confidence column, the item sets that did not pass minimum threshold of 15% is considered to have no strong confidence and so it is rejected, the frequent item-set that percentage pass 15% were accepted thus five (5) strong association rule with strong confidence were realized.

11. Results Analysis of the Five Strong Association Rules Realized

11.1 BasicSalary VeryLow PerformanceResult VeryLow

Is accepted because it passes the threshold with 16%. The certainty of their occurrences happened quite often. It is concluded that if employee's basic salary is very low it results to his performance to be very low because of loan monthly repayment as such, loan should be refuse to employee whose basic salary is very low because the loan will affect his performance.

11.2 Work Experience Very Low Performance Result Very Low

This pattern is accepted because it passes the threshold with 26% having strong confidence. The certainty of occurrences of this pattern happened quite often. That mean if employee's work experience is very low it will cause his performance to be very low, it is recommended that employee with very low work experience whose hire date is not up to a year should be refused loan if requested.

11.3 Work Experience Very Low Loan Repayment Very High

This pattern is accepted also since it passes the threshold with 21% having strong confidence. This pattern happens quite often. That mean if employee's work experience is very low and loan monthly repayment is very high is 90% likely that his performance will be very low therefore it is concluded that loan should be refused to such employee whose hire date is not up to a year when request large amount of loan that will lead to very high repayment.

11.4 Performance Result Very Low Loan Repayment Very High

This pattern is also accepted because it has passes the threshold with 28% having strong confidence. It happened quite frequent. Therefore, if employee's Performance Result is very low and loan monthly repayment is very high, it is recommended that employee whose previous performance is very low and requested to borrow large amount of loan should be refused because it is 95% likely that his performance will be very low since such situation frequently happened.

11.5 Work Experience Very Low Performance Result Very Low Loan Repayment Very High

This pattern is accepted as a strong association rule because it has strong confidence of 19% passing the minimum threshold. This pattern happened frequently employee whose work experience is very low his performance is 95% likely to be very low because of his loan monthly repayment is very high. It is concluded that employee whose requested large amount of loan and his hire date is not up to a year will lead to his performance to be very low because the high amount of monthly repayment will certainly affect is his performance since this pattern occur more frequent.

12. Created Rules

IF BasicSalary = VeryLow = AND PerformanceResult = VeryLow THEN Refuse

IF WorkExperience = VeryLow AND PerformanceResult = VeryLow THEN Refuse

IF WorkExperience =VeryLow AND LoanRepayment =VeryHigh THEN Refuse

IF PerformanceResult =VeryLow AND LoanRepayment =VeryHigh THEN Refuse

IF WorkExperience = VeryLow AND PerformanceResult = VeryLow AND LoanRepayment =VeryHigh THEN Refuse

Based on the five strong association rules generated by the algorithm, it is recommended that employee should be refuse loan on these conditions because the certainty (confidence) of their occurring together is frequent. Any other condition aside from these five rules, loan should be granted.

13. Real World Scenario Evaluation

When employee sign-in and apply for loan depending on the amount, the system will logically scan the record in the database to find the frequent item-set and interesting rules that will determine either to grant or refuse loan. For example, if he applies for 50,000 the algorithms refuse to grant him the loan as indicated in figure 29 below because IF BasicSalary is VeryLow and Loan Repayment (5,076) is Very High, it usually leads to Very Low Performance based on the frequent item-set and rules and the likeliness (probability) is 0.76 = 76%. Applying for Low amount does not lead to high or very high monthly repayment as such it will not lead to low or very low performance since not frequent pattern exist therefore the system will grant him the loan. For example, if he applied for 10,000 with monthly repayment of (1,015), the system grant the loan as indicated in figure 30 below.

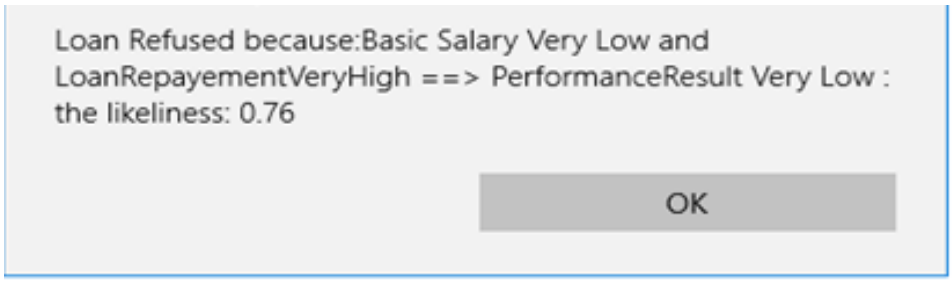


Figure 30 Loan Refuse

Figure 31 Loan Granted

14. Evaluation using WEKA Data Mining Tool

WEKA Data Mining Software, WEKA is an acronym aptly refers to Waikato Environment for Knowledge Analysis (WEKA) it is a software developed in University of Waikato and it is widely used in data mining. WEKA is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. The software contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes Mark H. et al. (2009)

14.1 Data Pre-processing

To evaluate the algorithms in another tool to see the results and compare, the data that was used in developing the algorithms was also used in WEKA. The data was clean, arrange as indicated in figure 31 below in a way that can be load and communicate in WEKA because it deals with relation, attribute and data also uses **ARFF** (Attribute-Relation File Format) file which is an ASCII text file that describes a list of instances sharing a set of attributes. **ARFF** files were developed by the Machine Learning Project at the Department of Computer Science of the University of Waikato for use with the WEKA machine learning software Mark H. et al. (2009).

```
@relation LoanApplication
@attribute BasicSalary {"VeryHighSalary", "HighSalary", "NormalSalary","LowSalary", "VeryLowSalary" }
@attribute WorkExperience {"VeryHighWorkExperience", "HighWorkExperience", "NormalWorkExperience","LowWorkExperience", "VeryLowWorkI
@attribute PerformanceResult {"VeryHighPerformance", "HighPerformance", "NormalPerformance","LowPerformance", "VeryLowPerformance" }
@attribute LoanRepayment {"VeryHighLoanRepayment", "HighLoanRepayment", "NormalLoanRepayment","LowLoanRepayment", "VeryLowLoanR
@attribute Duration {"DurationVeryHigh","DurationVeryLow" }
@attribute LoanStatus {"Granted", "Refuse"}
@data
VeryLowSalary,VeryLowWorkExperience,VeryLowPerformance,VeryHighLoanRepayment, DurationVeryHigh,Refuse
LowSalary,LowWorkExperience,LowPerformance,HighLoanRepayment,DurationVeryHigh, Refuse
NormalSalary,VeryLowWorkExperience,NormalPerformance,LowLoanRepayment, DurationVeryLow,Granted
HighSalary,VeryHighWorkExperience,VeryLowPerformance,VeryHighLoanRepayment,DurationVeryHigh,Refuse
VeryHighSalary,VeryHighWorkExperience,LowPerformance,LowLoanRepayment, DurationVeryHigh,Refuse
```

Figure 32 Loan Granted

14.2 Data Loading and Running

After loading the data, in the relation, there are 100 instances of records (transaction), six attributes were used these are Basic Salary, Loan Repaymet, Work Experience, Performance Result, Duration, and Loan Status, WEKA shows number of count and weight for each

attribute in text and chart, the chart uses two colours red and blue, red indicate successful grade whereas blue indicate unsuccessful grade.

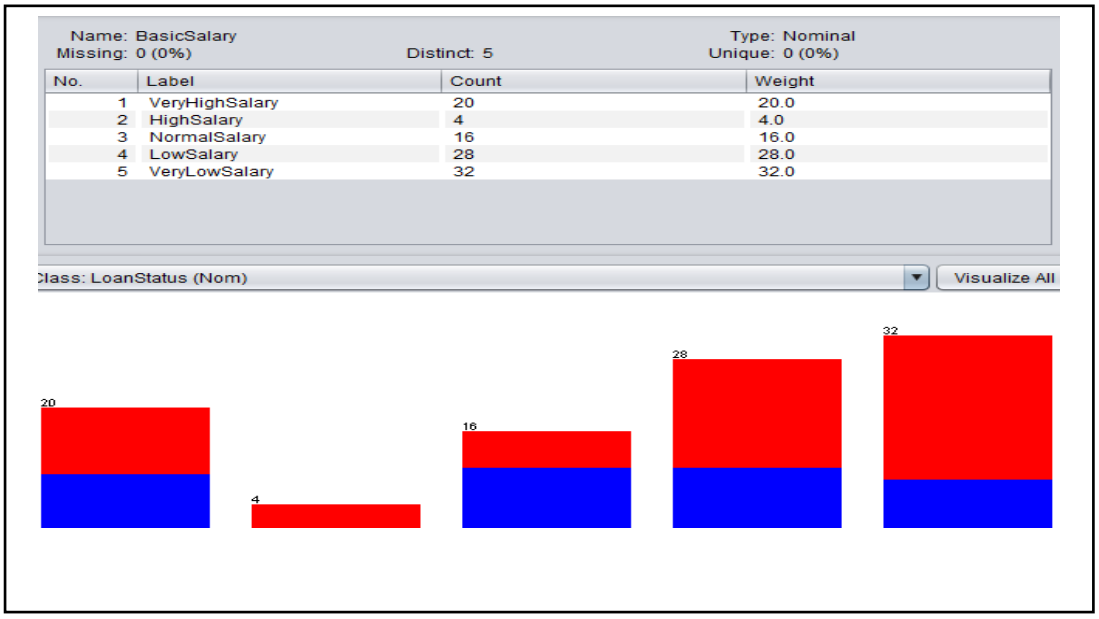


Figure 33 Basic Salary

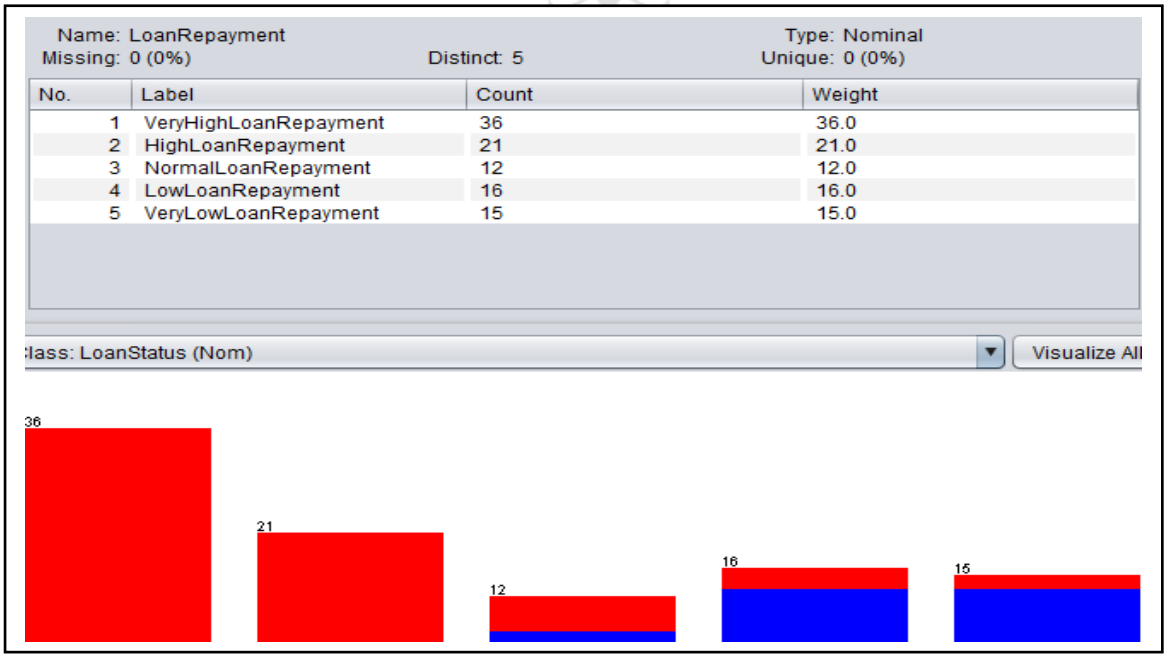
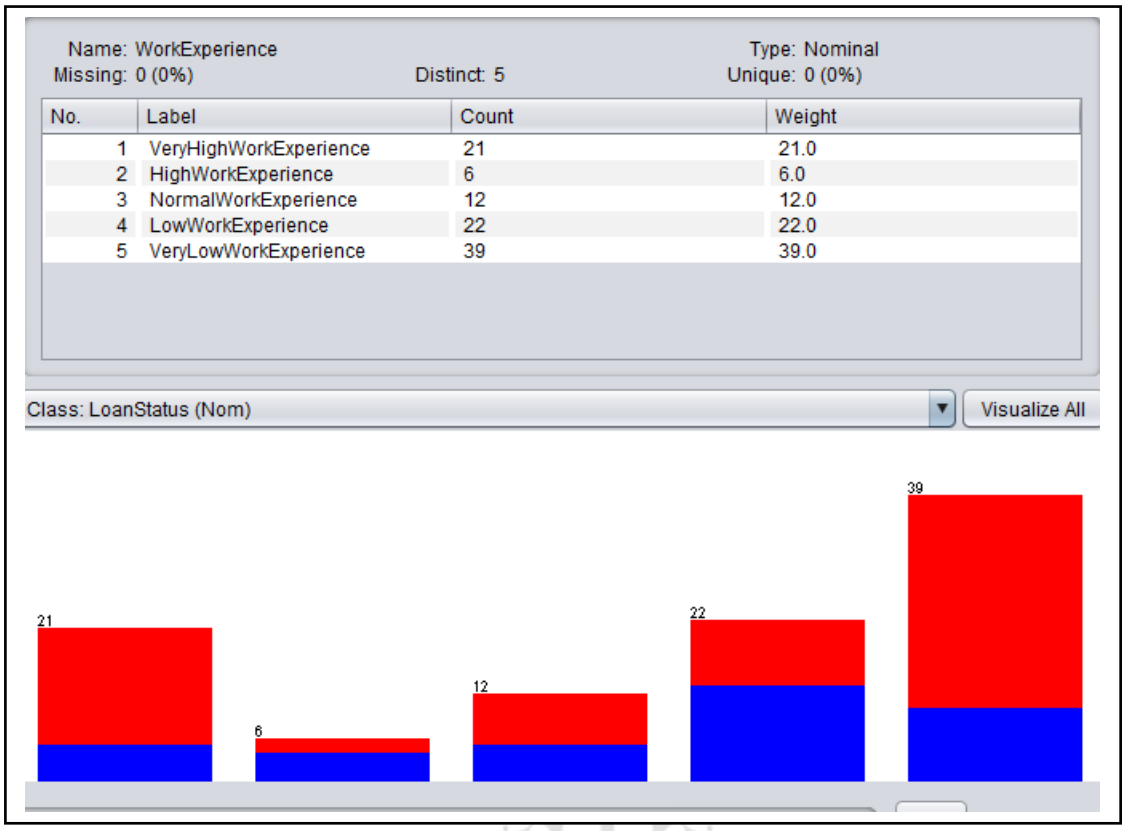


Figure 34 Loan Repaymet

Work Experience



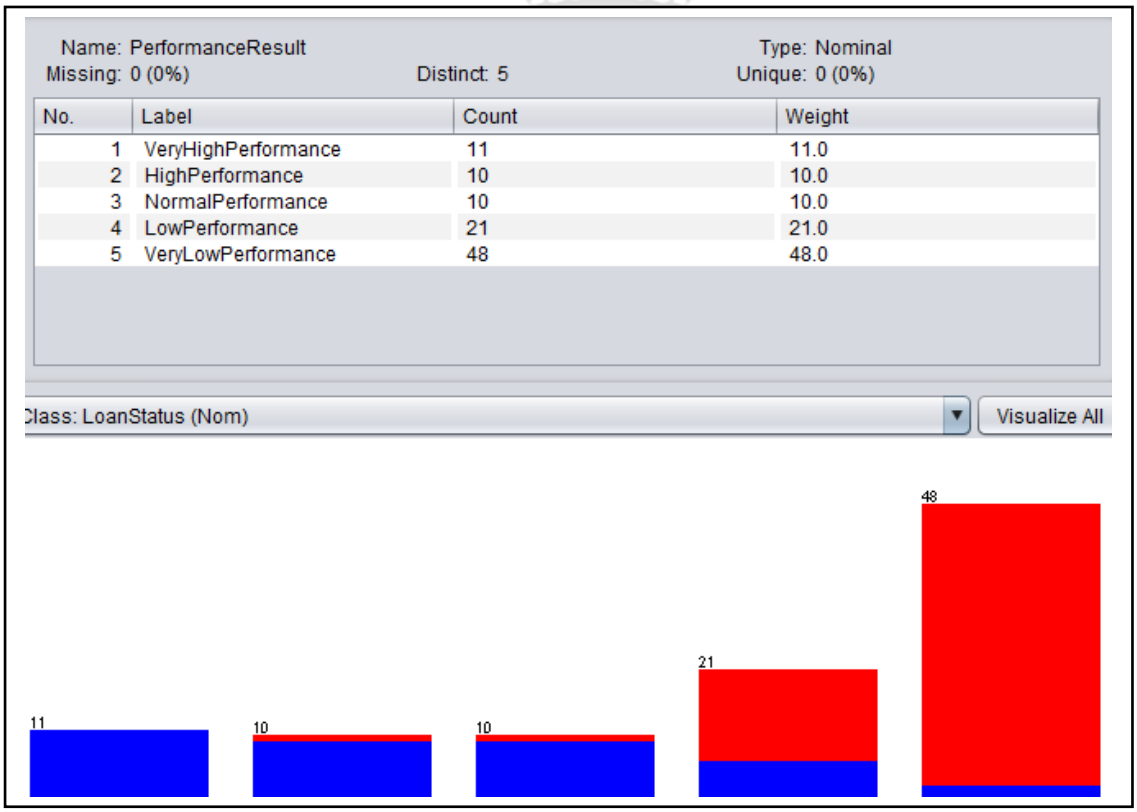


Figure 35 Work Experience

Duration

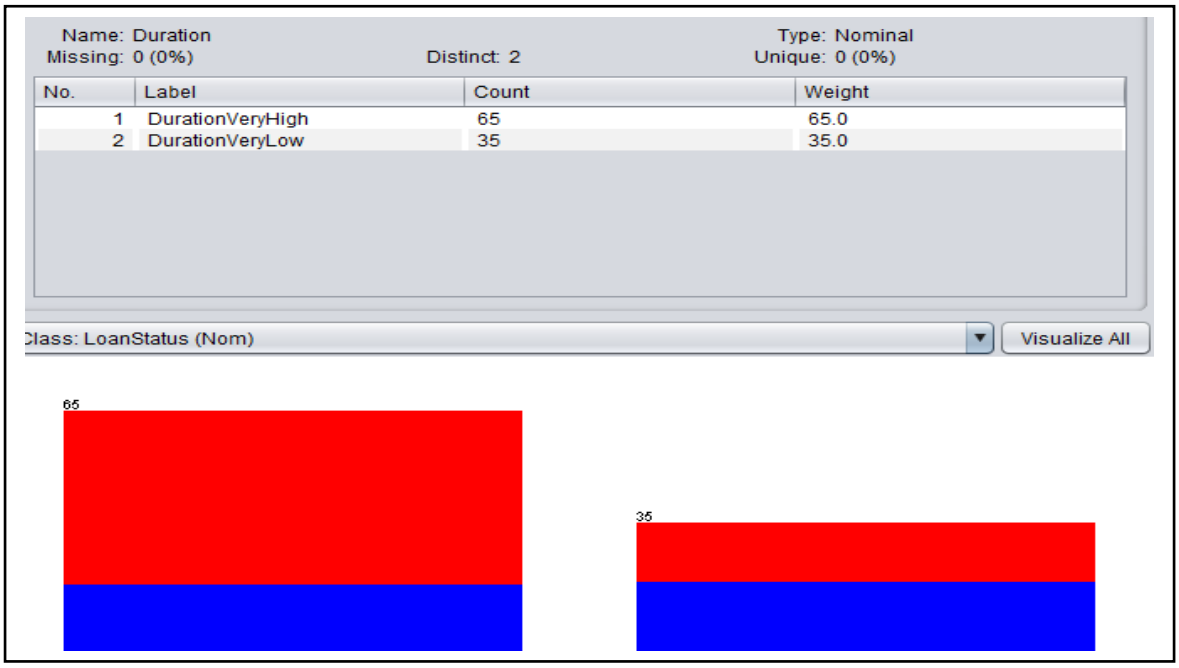


Figure 36 Duration

14.3 Association Rule Generated by WEKA

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     LoanApplication
Instances:    100
Attributes:   6
              BasicSalary
              WorkExperience
              PerformanceResult
              LoanRepayment
              Duration
              LoanStatus
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.2 (20 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 14

Size of set of large itemsets L(2): 12

Size of set of large itemsets L(3): 9

Size of set of large itemsets L(4): 3


Best rules found:

 1. PerformanceResult=VeryLowPerformance Duration=DurationVeryHigh 35 ==> LoanStatus=Refuse 35    <conf:(1)> lift:(1.59) lev:(0.13) [12] con
 2. PerformanceResult=VeryLowPerformance LoanRepayment=VeryHighLoanRepayment 30 ==> LoanStatus=Refuse 30    <conf:(1)> lift:(1.59) lev:(0.11
 3. WorkExperience=VeryLowWorkExperience PerformanceResult=VeryLowPerformance 27 ==> LoanStatus=Refuse 27    <conf:(1)> lift:(1.59) lev:(0.1
 4. LoanRepayment=VeryHighLoanRepayment Duration=DurationVeryHigh 27 ==> LoanStatus=Refuse 27    <conf:(1)> lift:(1.59) lev:(0.1) [9] conv:(
 5. WorkExperience=VeryLowWorkExperience LoanRepayment=VeryHighLoanRepayment 23 ==> LoanStatus=Refuse 23    <conf:(1)> lift:(1.59) lev:(0.09
 6. PerformanceResult=VeryLowPerformance LoanRepayment=VeryHighLoanRepayment Duration=DurationVeryHigh 22 ==> LoanStatus=Refuse 22    <conf:
 7. WorkExperience=VeryLowWorkExperience PerformanceResult=VeryLowPerformance LoanRepayment=VeryHighLoanRepayment 21 ==> LoanStatus=Refuse 2
 8. WorkExperience=VeryLowWorkExperience PerformanceResult=VeryLowPerformance Duration=DurationVeryHigh 20 ==> LoanStatus=Refuse 20    <conf
 9. LoanRepayment=VeryHighLoanRepayment 36 ==> LoanStatus=Refuse 35    <conf:(0.97)> lift:(1.54) lev:(0.12) [12] conv:(6.66)
10. PerformanceResult=VeryLowPerformance 48 ==> LoanStatus=Refuse 46    <conf:(0.96)> lift:(1.52) lev:(0.16) [15] conv:(5.92)
```

15. Result Analysis

Based on the 10 best rules generated by WEKA as indicated above, there are 10 instances that loan will be refused to employee any other instance, the loan will be granted. All the generated rules have confidence of <conf:(1)> which signifies the degree of trustworthiness or certainty of their occurrences hence the rules interpreted as shows below

- IF Performance Result=Very Low AND Duration=Very High THEN Loan Status =Refuse

- IF Performance Result=Very Low AND Loan Repayment=Very Low THE Loan Status = Refuse

- IF Work Experience=Very Low AND Performance Result=Very Low THE Loan Status =Refuse

- IF Loan Repayment =Very High AND Duration = Very High THE Loan Status=Refuse.

- IF Work Experience=Very Low AND Loan Repayment =Very High THEN Loan =Refused

- IF Performance Result=Very Low AND Loan Repayment=Very High AND Duration =Very High THEN Loan = Refuse

- IF Work Experience=Very Low AND Performance Result=Very Low AND Loan Repayment =Very High THEN Loan = Refuse

- IF Work Experience=Very Low AND Performance Result=Very Low AND Duration= Very High THEN Loan = Refuse

- IF Loan Repayment=Very High THEN Loan =Refuse

- IF Performance Result=Very Low THEN Loan = Refuse

16. RESULTS COMPARISON

Five Strong association rule were generated in the algorithms, in WEKA Ten best rules were found, there are four rules that existed in the algorithms and these are rules were also exist in the ten best rules found by WEKA these rules are:

IF Work Experience =Very Low AND Performance Result =Very Low THEN Loan = REFUSE

IF Performance Result= Very Low AND Loan Repayment =Very High THEN Loan = REFUSE

IF Work Experience=Very Low AND Loan Repayment =Very High THEN Loan = REFUSE

IF Work Experience Very Low AND Performance Result =Very Low AND Loan Repayment =Very High THEN Loan = REFUSE

16.1 Algorithms	4 Unique rules X 100 5 Generated rules	=	0.8*100=	80%
16.2 WEKA	4 Unique rules X 100 10 Best rules found	=	0.4*100=	40%

Apriori algorithms developed in ASP.NET from the prototype, achieved the highest accuracy of 80%. The accuracy achieved in WEKA was 40%.

17. CONCLUSION:

Data mining can be applied in different domain. However, privacy, security and misuse of information are the biggest challenges if they are not properly addressed. In this study, we adopt constructive methodology intended to solve domain problems faced in real world and developed an employee management prototype system using ASP.NET and. We then use the developed prototype and implemented association rule algorithms capable of unsupervised machine learning using Apriori algorithm approach. We have implemented the association rule on selected dataset from employee prototype system and generated set of association rules. We examined all the generated association rule and selected only 5 strong rules as most important rule to our work. The strong rules generated determined that loan should be refused to employee because their occurrences together tends to affect employee's performance negatively. We finally used WEKA to evaluate and compare the algorithms developed in ASP.NET and that of WEKA to find which one is having highest accuracy. We realised that the one implemented in ASP has highest accuracy of 80% while WEKA is 40%

18. Future Work

Our future work will involve using larger records of the above algorithms to generate many strong rules thereby enhancing the accuracy so that the algorithms can become more accurate.

REFERENCES

1. Jantan, H. et al. (2010). "Human Talent Prediction in HRM using C4.5 Classification Algorithm", International Journal on Computer Science and Engineering, 2(08-2010), pp. 2526-2534
2. Kayha, E. (2007). The Effects of Job Characteristics and Working Conditions on Job Performance", International Journal of Industrial Ergonomics, In Press.

3. Liisa L.et al The Constructive Research Approach: Problem Solving for Complex Projects available from http://www.gpmfirst.com/books/designs-methods-and-practices-research-project-management/constructive-research-approach
4. Mark H. et al. (2009) The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
5. Munchi G. (2016) Gartner Business Intelligence & Analytics Summit Available from http://searchbusinessanalytics.techtarget.com/definition/association-rules-in-data-mining
6. Oliggbo I. et al (2004) A Review of Data Mining as a Tool for Organisational Growth and Productivity International Journal of Computer Science and Mobile Computing, Vol.3 Issue.9, September- 2014, pg. 284-290
7. Qasem A. (2012) Using Data Mining Techniques to Build a Classification Model for Predicting Employees Performance Department of Computer Information Systems, Faculty of Information Technology and Computer Science Yarmouk University, Irbid 21163, Jordan. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 2, 2012
8. Rohit K. et al. (2014) Data Mining: Evaluating Performance of Employee's using Classification Algorithm Based on Decision Tree IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498
9. Vol.4, No. 2, April 2014
10. Roxanne A. et al. (2012) Predicting Faculty Development Trainings and Performance Using Rule-Based Classification Algorithm Department of Information Technology Education, Technological Institute of the Philippines, Quezon City, Philippines Asian Journal of Computer Science and Information Technology 2: 7 (2012) 203 – 209. http://www.innovativejournal.in/index.php/ajcsit
11. Sangita G. (2013) Enhancing Human aspect of Software Engineering using Bayesian classifier International Journal of Cognitive Science, Engineering, and Technology Volume 1, Issue 1, November-2013 ISSN 2347 – 8047
12. SCHOOL OF COMPUTER SCIENCE University of Manchester for the degree of MSc Software Engineering
13. Tan et al. (2004) Association Analysis: Basic Concept and Algorithms Available from
14. Zhangmin Lu (2009) Design and Implementation of a Web Service Application for SCM