The International Journal Of Engineering And Science (IJES)
|| Volume || 5 || Issue || 8 || Pages || PP 74-89|| 2016 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Predicting Software Defects Using Bayesian Network Approach

E. E. Ogheneovo*, N. E. Udofia
[1]*Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria*
[2]*Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria*

ABSTRACT
Software defects are very predominant especially in large software these days because programmers do not take the pain to properly debug and test their software before releasing them to customers and users. When these software are put into use, the defects manifest themselves especially if they are dormant or inactive at the time the software is being developed. As a result, they do cause errors and eventually failure if not quickly handled thereby causing serious damages in terms of human and material loss. This paper proposed the Bayesian network model for predicting software defects. In the model, certain software inputs are tested to determine the number of defects in them. Therefore, our program which is developed in Java programming language is able to count the number of inputs and outputs using certain parameters and expressions. This is possible when we input query into the network. The program is able to determine that a query has been supplied, the query type, the various events that takes place at each stage of the query determination, the choices made and the probability of defects being found or not being found in the software.

Keywords: Software, software defects, Bayesian network, defect prediction model, NASA metric, software failure

Date of Submission: 07 August 2016 Date of Accepted: 30 August 2016

I. INTRODUCTION

There has been a huge growth in the demand for quality and reliable software in recent times. As a result, issues relating to software testing, software quality and software reliability are at the forefront when discussing issues relating to software [1]. These issues are increasingly becoming more critical among software engineers, developers, and users of the software. The ability to determine and measure software defects are extremely important to ensure cost minimization and also help in improving the overall quality and reliability of such software. As noted in Schull et al. [2], finding and fixing a severe software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase. Therefore, it is always easier and less expensive to be able to detect and/or predict software defects at the early stage of software development [3].

Software defects [4] -- [6] are very predominant especially in large software these days because programmers do not take the pain to properly debug and test their software before releasing them to customers and users. When these software are put into use, the defects manifest themselves especially if they are dormant or inactive at the time the software is being developed. As a result, they often cause errors and eventually failure if not quickly handled thereby causing serious damages in terms of human and material loss. Software defects are faults or bugs introduced into software intentionally or unintentionally by the software developer(s) when the software is being developed. Quinlan and Panas [7] define software defects as faults that are introduced unintentionally into computer programs, preventing them from behaving correctly. Software defects pose serious problems during development and after delivery to users [8] [9]. There is hardly any piece of software that is free of errors or defects no matter how small it is especially when it is first developed.

Software complexity is a major cause of software defects [10]. As software complexity increases, so does the likelihood of more defects or bugs in the software. Software complexity [11] -- [13] causes more errors to be introduced to the software. Large software projects are more likely to have more defects due to their complexities. Therefore, it is necessary to do everything humanly possible to minimize them drastically at each stage of the development process instead of trying to fix the software after development or during post-delivery stage. It is often better to find and fix defects during development than after delivery. Better still, finding and fixing bugs at each stage of the developmental process is even for cheaper than after development or after delivery. Therefore, early verification and validation of software is to ensure that software defects are found and fixed earlier in software development life cycle.

This paper proposed the Bayesian network model for predicting software defects. In the model, certain software inputs are tested to determine the number of defects in them. Therefore, our program which is developed in Java programming language is able to count the number of inputs and outputs using certain parameters and expressions. This is possible when we input query into the network. The program is able to determine that a query has been supplied, the query type, the various events that takes place at each stage of the query determination, the choices made and the probability of defects being found or not being found in the software. It paper also determines the probability of avoiding defects during software development which depends on the defect in given total potential defects which represents the number of defects before testing that are in the new code that has been implemented. This number is used to determine the probability of finding defect in the code which therefore determines the number of defects found in the software.

II. RELATED WORK

Liu et al. [14] proposed a model that builds Bayesian Network that represents a probability distribution of each factor and how they affect defects, considering strong or weak correlations exists between individual metric attributes. The model was compared with other models and it produces statically significant estimations. Fenton et al. [15] proposed a Bayesian network approach for predicting the number of residual defects that are likely to be found during independent testing or operational usage.

Radhinski and Hoffmann [16] carried out a comprehensive study on software development prediction by comparing 23 classifiers in WEKA over four public datasets. They analyzes the accuracy of predictions for software. Development effort using different machine learning techniques based on the stability of the predictions. They tried to find out if particular techniques achieve similar level of accuracy using different datasets. Two assumptions were drawn from the work: (1) predictions are performed using local empirical data, and (2) predictions are performed using local empirical data and very little expert input is required. They used 23 machine learning techniques with four publicly available datasets: COCOMO, Desharnais, Maxwell, and QQDefects.

Mendes and Mosley [17] compared several Bayesian network models for Web effort estimation using a cross-company dataset. They developed eight Bayesian networks, which were divided into two groups. Four of them were built using Hugin and PowerSoft tools with two training sets, each with 130 Web projects while the other four were built using a causal graph based on domain expert. Using a benchmark, the BN-based estimates were also compared to estimates obtained using manual stepwise regression (MSWR), case-based reasoning (CBR), mean-and-median-based effort models. The result shows that better performance are obtained when simpler models such as median effort as opposed to complex models such as MSWR.

III. METHODOLOGY

In the model, certain software inputs are tested to determine the number of defects in them. Therefore, our program which is developed in Java programming language is able to count the number of inputs and outputs using certain parameters and expressions. This is possible when we input query into the network. The program is able to determine that a query has been supplied, the query type, the various events that takes place at each stage of the query determination, the choices made and the probability of defects being found or not being found in the software. It paper also determines the probability of avoiding defects during software development which depends on the defect in given total potential defects which represents the number of defects before testing that are in the new code that has been implemented. This number is used to determine the probability of finding defect in the code which therefore determines the number of defects found in the software. Figure 1 shows a proposed framework for our model.

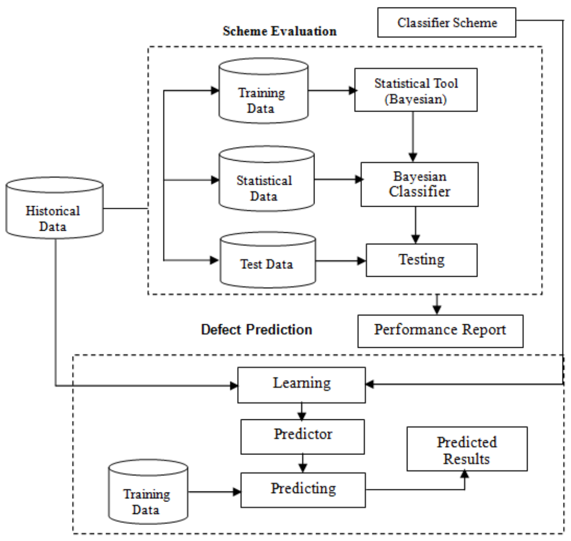


Fig. 1 Proposed framework for software defect prediction

3.1 Overview of the Proposed Model

The framework is made up of three parts: scheme evaluation, defect prediction, and verification. The scheme evaluation helps to analyze the prediction function of various competing schemes with all required statistics. The defect predictor builds techniques with respect to the estimated learning scheme. Apart from this, the defect predictor also helps to predict the software defects with all the data based on the constructed model. The verification step helps to verify that the classification is properly done and that the model is properly built. The various parts of the model are now explained in detail.

3.2 Scheme Evaluation

The scheme evaluation is perhaps the most prominent part of this prediction framework. It comprises the following: training data, statistical data, test data, learning tool (Artificial Neural Network), Statistical tool (Bayesian Networks), and Testing tool. This scheme helps to analyze the prediction function of various competing schemes with all required statistics. The defect predictor builds techniques with respect to the estimated learning scheme. At this level, many learning schemes are estimated by construction and measuring learners. However, the major function of this scheme is to categorize the training and test data by using historic data. The historic data contain all the data that have been used in the past to predict data. It stores both the dataset and information pertaining them in a repository and ensure that these information are available when requested for by the scheme evaluation. Thus the test data are usually autonomous when building the learner. Also, it contains the precondition to be evaluated in order to find the functioning of a learner over fresh data. This way, a cross validation is used to compute the precision of a predictive model in real life situation. Thus the cross-validation in a single round helps in partitioning the dataset into required complementary subsets. It then helps to analyze each of the subsets and then validate the analysis on the subsets. Therefore, to ensure variability, cross validation should be carried out on the partitioned datasets after which the evaluation results are computed

Training Data: The training data are the datasets that are used in the experiment for implementing the research. Usually, they are trained by the training tool such as the artificial neural network before they are used to test the program.

Test Data: These are the trained data which are used to test the source code after they have been trained. The real-time defect data sets used in this research. The data sets used in this study are obtained from NASA projects such as NASA MDP, and other space exploration related projects such as PC1, PC4, CM1, and ground orbiting satellite such as KC1 and KC3. There are twelve data sets are used to validate our technique. These datasets were obtained from NASA's Metric Data Program (MDP) data repository, available online at http//mdp.ivy.nasa.gov. The CM1 datasets is obtained from a spacecraft instrument, written in C programming language. It contains approximately 506 modules. The JM1 dataset is obtained from a predictive system project, written in C++. It contains approximately 10879 modules. The KC1 data is obtained from a science data processing project, written in C++. It contains approximately 2108 modules. The PC1 data is obtained from a science data processing project coded in C++. It contains 1108 modules. As seen in Table 1, these datasets varied in the percentage of defects modules, with CM1 dataset having the least percentage of defects in its modules and KC1having the largest percentage of defects.

Learning Tool: The learning tool used is the artificial neural networks. A neural networks model is a multi-layer perception model that produces a value between 0 and 1. Usually, the predictors are in one layer, with each predictor as one neuron, and the output is in one layer. A non-linear function is used to combine values to connect layers and to produce the output. For a new observation, the predictors' values are placed on the outer layer and the predicted value between 0 and 1 is produced at the output neuron. Each artificial neuron is a simple processor with the ability to add all weighted inputs and then apply a mathematical transformation to generate an output.

Statistical Tool: Bayesian Networks is used as the statistical tool for performing statistical data. A Bayesian network is a graph together with an associated set of probability tables. The nodes represent uncertain variables and the arcs represent the causal/relevance relationships between the variables. The variable 'effective KLOC implemented' represents the complexity-adjusted size of the functionality implemented. As the amount of functionality increases the number of potential defects rises. This is due to increase in the size and complexity of the source code.

The 'probability of avoiding defect in development' depends on 'defects in' given total potential defects. This number represents the number of defects (before testing) that are in the new code that has been implemented. This number 'defect in' the code determines the probability of finding defect in the code which therefore determines the number of defects found in the software.

Testing: Testing is carried out after the data sets have been trained to determine whether they have defects or not. After testing the software projects, the number of defects found in each module is recorded and the total number of defects found in each project is also recorded and some values are the computed based on these findings.

3.3 Defect Predictor

The defect predictor is built to ensure that it can accurately predict the defects by ensuring that the false positive (fp) and false negatives (fn) are reduced to small fractions. When building the predictor, the following should be taken to consideration.

1). A learning scheme is selected based on the performance results. This scheme is then used to predictor the defects by ensuring that the defects are properly classified.

2). A predictor is constructed with the required relevant learning scheme as well as the entire statistics. The learning scheme contains the neural network which is used to train the datasets to ensure that they are well utilized.

3). The datasets must be preprocessed in a similar way as earlier performed by historical data and the purpose of constructed predictor seems to predict software defects using all the preprocessed data.

3.4 Bayesian Network Classifier

The Bayesian classifier is a simple probabilistic classifier based on applying Bayesian theorem with strong independence assumption. The underlying probability model is an independent feature model. The Bayesian classifier is efficiently trained in supervised learning due to the precise nature of the probability model. Maximum likelihood methodology is used to find the parameter estimates in the Bayesian model. When the machine learning techniques are used to create functions from the training data, it is called supervised learning. The statistical parameters in a dataset is derived and this is called Maximum Likelihood Estimation (MLE). Based on interpretation of Bayesian probability and in accordance with Bayesian theory, a Bayesian classifier states that the occurrence or nonoccurrence of a feature is non- linked in any way to the occurrence or nonoccurrence of any other feature.

Thus in a supervised learning problem, if an unknown target function

$f : X \rightarrow Y$ or equivalently $P(Y|X)$ is to be approximated, assume Y is a Boolean-valued random variable, and X is a vector containing n Boolean attributes. In other words, $X = (X_1, X_2, \ldots, X_n)$, where X_i is the Boolean random variable denoting the ith attributes of X.

Applying Bayes rule, it is learnt that $P(Y = y_i|X)$ can be represented as:

$$P(Y = y_i|X = x_k) = \frac{P(x=xk|Y=yi)P(Y=y=i)}{\Sigma i\, P(X=Ixi|Y=yi)P(Y=yj)} \qquad (3.1)$$

Where y_m denotes the mth possible value for Y, x_k denotes the kth possible vector value for X.

The Bayesian classification algorithm assumes the attributes $(X_1, X_2, \ldots, X_n)$ are all conditionally independent of one another, given Y. the value of this supposition is that it radically simplifies the representation of P(X|Y), and the problem of approximating it from the training data.

Thus we write $P(X|Y)$ to denote the probability of event X (an hypothesis) conditional on the occurrence of some event Y (evidence). If we are counting sample points, we are interested in the fraction of events Y for which X is also true. From this it should be clear that (with the comma denoting the conjunction of events), we have

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} \qquad (3.2)$$

This is often written in the form

$$P(X, Y) = p(X \mid Y)p(y) \qquad (3.3)$$

and referred to as the "product rule." It is important to realize that this form of the rule is not, as often stated, a definition. Rather, it is a theorem derivable from simpler assumption.

The Bayesian theorem can be used to tell us how to obtain a posterior probability of a hypothesis X after observation of some evidence Y, given the *prior* probability of X and the likelihood of observing Y were X to be the case:

$$P(X|Y) = \frac{P(X\,|,Y)p(A)}{P(Y)} \qquad (3.4)$$

This simple formula has immense practical importance on a domain such as diagnosis. It is often easier to elicit the probability, for example, of observing a symptom given a disease than that of a disease given symptom. Yet, operationally, it is usually the latter which is required. Thus in its general form, the Bayesian Theorem is as shown in equation (3.3).

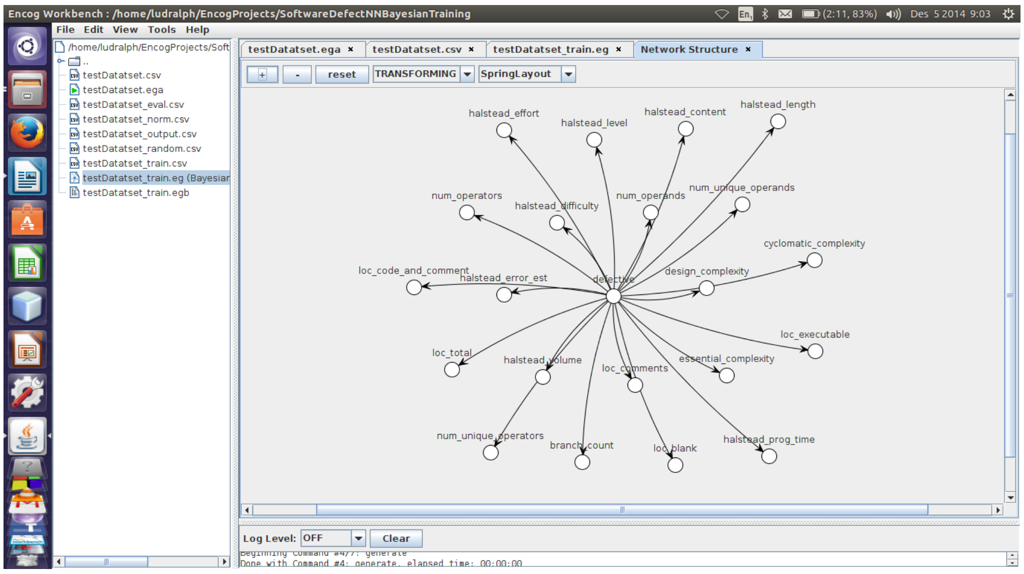
IV. RESULTS AND DISCUSSION

The real-time defect data sets used in this research shown in table 1. The data sets used in this study are obtained from NASA projects such as NASA MDP, and other space exploration related projects such as PC1, PC4, CM1, and ground orbiting satellite such as KC1 and KC3.

Table 1 Data sets used for this study

Data Set	No. of Attributes	No. of Modules	Programming Language
CM1	39	506	C
JM1	21	10879	C++
KC1	21	2108	C++
KC3	39	429	Java
KC4	39	125	C
MC1	39	4621	C
MC2	39	161	C
MW1	39	403	C
PC1	39	1108	C
PC2	39	4505	C
PC3	39	1511	C
PC4	39	1347	C

This NASA database is a repository that stores problems, products, and metrics data. The primary goal of this repository is to provide project data to the software community. Thus the Metrics Data Program collects artifacts from a large NASA datasets, generates metrics on the artifacts reports made available to the public. These datasets contains modules such as functions, subroutine, or methods containing lines of code (LOC) based metrics, Halstead metrics, and McCabes's Complexity measures. The number of defective modules is indicated in each of the projects.



```
Beginning Command #4/7: generate
Done with Command #4: generate, elapsed time: 00:00:00
```

Fig. 2 A screenshot of Bayesian network after running the Java program

Figure 2 shows a screenshot of the Bayesian network showing the different datasets such as lines of code (LOC) based metrics, Halstead metrics, and McCabes's Complexity measures, etc.

V. CONCLUSIONS

In this paper, we proposed the Bayesian network model for predicting software defects. In the model, certain software inputs are tested to determine the number of defects in them. Therefore, our program which is developed in Java programming language is able to count the number of inputs and outputs using certain parameters and expressions. This is possible when we input query into the network. The program is able to determine that a query has been supplied, the query type, the various events that takes place at each stage of the query determination, the choices made and the probability of defects being found or not being found in the software.

REFERENCES

[1]. E. E. Ogheneovo, "Software Dysfunctions: Why Do Software Fails? *Journal of Computer and Communications*," vol. 2, pp. 25-35, April, 2014. Available: http://dx.doi.org/10.4235/jcc.2014.

[2]. F. Schull, F. Basili, V. Boehm, W. Brown, P. Costa, M. Lindvall , Port, I. Rus, R. Tesoriero and M. Zelkowitz, " What We Have Learned About Fitting Defects,*" METRICS'02, IEEE*, 2002.

[3]. J. E. Gathey, Jr. (1984) "Estimating the Number of Faults in Code," *IEEE Transactions on Software Engineering*, vol. SE10, pp. 459-464, 1984.

[4]. F. Akiyama, "An example of software system debugging. *Proceedings of the International Federation for Information Processing Congress*," Ljubljana, Yugoslavia, pp. 353–379.

[5]. S. Lessmann, B. Baesens, C. Mues and S. Pietsch, " Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Transactions on Software Engineering*, vol. 34, Issue 4, pp. 485-496,2008, DOI: 10.1109/TSE.2008.35.

[6]. T. M. Khoshgoflaar and J. C. Munson, "Predicting Software Development errors Using Software Complexity Metrics, IEEE Journal on Selected Areas in Communications, Vol. 8, no. 2, 253-264, 1990.

[7]. D. Quinlan and T. Panas, "Source code and Binary Analysis of Software Defects," CSIIRW, April 13-15, OAK Ridge, Tennessee, USA.

[8]. A. G. Koru and h. Liu, "An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures," PROMISE'05, May 15 2005, St. Louis Missouri, USA.

[9]. N. Katiyar and R. Singh, "Prediction of Software Development Faults Using Neural Network," *VSRD Int'l Journal of Computer Science and Information Technology*, vol. 1, no. 8, 2011, pp. 556-566, 2.011

[10]. E. E. Ogheneovo, "On the Relationship between Software Complexity and Maintenance Costs," Journal of Computer and Communications, vol. 2, pp. 1-16. Available: http://dx.doi.org/10.4236/jcc.2014.

[11]. M. H. Halstead," *Elements of Software Science*," Amsterdam: Elsevier North-Holland, Inc. ISBN 0-444-00205-7, 1976.

[12]. T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, December 1976.

[13]. S. C. Narula and J. F. Wellington, "Prediction, Linear Regression and the Minimum Sum of Relative Errors," Technometrics, vol. 19, pp. 185-190, 1977.

[14]. Y. Liu, W. P. Cheah, B.-K. Kim and H. Park, "Predict Software Failure-prone by Learning Bayesian Network," *Int'l Journal of Advanced Science and Technology*, pp. 35-42, 2005.

[15]. N. Fenton, M. Neil, W. Marsh, and P. Hearty, "On the Effectiveness of Early Lifecycle Defect Prediction with Bayesian Networks," Empir. Software Engineering, vol. 13, pp. 499-537, June 2008. DOI. 10.1007/s10664-008-9072-x.

[16]. L. Radlinski and W. Hoffmann, "On Predicting Software Development Effort Using Machine Learning Technique and Local Data," *Int'l Journal of Software engineering and Computing*, vol. 2, no. 2, pp. 123-136, 2010.

[17]. E. Mendes and N. Mosley, "Bayesian Network Models for Web Effort Prediction: A Comparative Study," *Software Engineering Journal, IEEE Transactions on Software Engineering*, vol. 34, no. 6, pp. 723-737, 2008.

Appendix: Bayesian Network Implementation

Input Count	22
Output Count	1
Parameter Count	425
Expression	P(loc_blank|defective) P(branch_count|defective) P(loc_code_and_comment|defective) P(loc_comments|defective) P(cyclomatic_complexity|defective) P(design_complexity|defective) P(essential_complexity|defective) P(loc_executable|defective) P(halstead_content|defective) P(halstead_difficulty|defective) P(halstead_effort|defective) P(halstead_error_est|defective) P(halstead_length|defective) P(halstead_level|defective) P(halstead_prog_time|defective) P(halstead_volume|defective) P(num_operands|defective) P(num_operators|defective) P(num_unique_operands|defective) P(num_unique_operators|defective) P(loc_total|defective) P(defective)
Query Type	EnumerationQuery
Query	P(+defective|loc_blank=0,branch_count=0,loc_code_and_comment=0,loc_comments=0,cyclomatic_complexity=0,design_complexity=0,essential_complexity=0,loc_executable=0,halstead_content=0,halstead_difficulty=0,halstead_effort=0,halstead_error_est=0,halstead_length=0,halstead_level=0,halstead_prog_time=0,halstead_volume=0,num_operands=0,num_operators=0,num_unique_operands=0,num_unique_operators=0,loc_total=0)

Events

Event	Choices	Probability
loc_blank	0, 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24, 3, 35, 4, 5, 58, 6, 7, 8, 9	P(loc_blank|defective)
branch_count	1, 10, 11, 12, 13, 14, 15, 16, 17, 19, 21, 23, 25, 26, 27, 29, 3, 31, 33, 35, 37, 39, 4, 42, 49, 5, 51, 53, 54, 6, 67, 7, 8, 89, 9	P(branch_count|defective)

loc_code_and_comment	0, 1, 11, 2, 3, 4, 5, 6, 7, 8	P(loc_code_and_comment|defective)
loc_comments	0, 1, 10, 11, 12, 14, 16, 17, 19, 2, 20, 26, 3, 35, 4, 44, 5, 6, 7, 8, 9	P(loc_comments|defective)
cyclomatic_complexity	1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 22, 25, 26, 27, 28, 3, 34, 4, 45, 5, 6, 7, 8, 9	P(cyclomatic_complexity|defective)
design_complexity	1, 10, 11, 12, 13, 14, 15, 16, 18, 19, 2, 22, 25, 27, 29, 3, 4, 45, 5, 6, 7, 8, 9	P(design_complexity|defective)
essential_complexity	1, 10, 11, 12, 14, 15, 16, 18, 19, 21, 22, 26, 3, 4, 5, 6, 7, 8, 9	P(essential_complexity|defective)
loc_executable	Type0, Type1, Type2	P(loc_executable|defective)
halstead_content	Type0, Type1, Type2	P(halstead_content|defective)
halstead_difficulty	Type0, Type1, Type2	P(halstead_difficulty|defective)
halstead_effort	Type0, Type1, Type2	P(halstead_effort|defective)
halstead_error_est	Type0, Type1, Type2	P(halstead_error_est|defective)
halstead_length	Type0, Type1, Type2	P(halstead_length|defective)
halstead_level	Type0, Type1, Type2	P(halstead_level|defective)
halstead_prog_time	Type0, Type1, Type2	P(halstead_prog_time|defective)
halstead_volume	Type0, Type1, Type2	P(halstead_volume|defective)
num_operands	Type0, Type1, Type2	P(num_operands|defective)
num_operators	Type0, Type1, Type2	P(num_operators|defective)
num_unique_operands	Type0, Type1, Type2	P(num_unique_operands|defective)
num_unique_operators	0, 1, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 3, 30, 31, 4, 5, 6, 7, 8, 9	P(num_unique_operators|defective)
loc_total	Type0, Type1, Type2	P(loc_total|defective)
defective	N, Y	P(defective)

Probability Tables

loc_blank	
P(loc_blank=0|-defective)	87.441860%
P(loc_blank=1|-defective)	57.251908%
P(loc_blank=2|-defective)	3.333333%
P(loc_blank=3|-defective)	3.703704%
P(loc_blank=4|-defective)	63.709677%
P(loc_blank=5|-defective)	7.142857%
P(loc_blank=6|-defective)	3.703704%
P(loc_blank=7|-defective)	3.703704%
P(loc_blank=8|-defective)	6.896552%
P(loc_blank=9|-defective)	3.703704%
P(loc_blank=10|-defective)	48.351648%
P(loc_blank=11|-defective)	3.703704%
P(loc_blank=12|-defective)	6.060606%
P(loc_blank=13|-defective)	40.384615%
P(loc_blank=14|-defective)	22.727273%
P(loc_blank=15|-defective)	3.571429%

P(loc_blank=16|-defective)	28.888889%
P(loc_blank=17|-defective)	15.789474%
P(loc_blank=18|-defective)	6.250000%
P(loc_blank=19|-defective)	0.000000%
P(loc_blank=20|-defective)	9.375000%
P(loc_blank=21|-defective)	9.677419%
P(loc_blank=22|-defective)	0.000000%
P(loc_blank=23|-defective)	3.571429%
P(loc_blank=24|-defective)	3.448276%
P(loc_blank=25|-defective)	8.823529%
P(loc_blank=26|-defective)	20.588235%
P(loc_blank=0|+defective)	6.744186%
P(loc_blank=1|+defective)	23.664122%
P(loc_blank=2|+defective)	13.333333%
P(loc_blank=3|+defective)	3.703704%
P(loc_blank=4|+defective)	16.129032%
P(loc_blank=5|+defective)	3.571429%
P(loc_blank=6|+defective)	3.703704%
P(loc_blank=7|+defective)	3.703704%
P(loc_blank=8|+defective)	6.896552%
P(loc_blank=9|+defective)	3.703704%
P(loc_blank=10|+defective)	24.175824%
P(loc_blank=11|+defective)	3.703704%
P(loc_blank=12|+defective)	18.181818%
P(loc_blank=13|+defective)	11.538462%
P(loc_blank=14|+defective)	20.454545%
P(loc_blank=15|+defective)	7.142857%
P(loc_blank=16|+defective)	15.555556%
P(loc_blank=17|+defective)	18.421053%
P(loc_blank=18|+defective)	15.625000%
P(loc_blank=19|+defective)	0.000000%
P(loc_blank=20|+defective)	12.500000%
P(loc_blank=21|+defective)	9.677419%
P(loc_blank=22|+defective)	0.000000%
P(loc_blank=23|+defective)	7.142857%
P(loc_blank=24|+defective)	10.344828%
P(loc_blank=25|+defective)	17.647059%
P(loc_blank=26|+defective)	5.882353%
branch_count	
P(branch_count=0|-defective)	5.555556%
P(branch_count=1|-defective)	10.869565%
P(branch_count=2|-defective)	7.317073%
P(branch_count=3|-defective)	11.627907%
P(branch_count=4|-defective)	2.857143%
P(branch_count=5|-defective)	5.263158%
P(branch_count=6|-defective)	5.128205%
P(branch_count=7|-defective)	68.387097%
P(branch_count=8|-defective)	2.702703%
P(branch_count=9|-defective)	5.263158%
P(branch_count=10|-defective)	5.555556%
P(branch_count=11|-defective)	2.777778%
P(branch_count=12|-defective)	42.268041%
P(branch_count=13|-defective)	2.857143%
P(branch_count=14|-defective)	2.777778%
P(branch_count=15|-defective)	8.108108%
P(branch_count=16|-defective)	7.894737%
P(branch_count=17|-defective)	33.333333%

P(branch_count=18|-defective)	2.777778%
P(branch_count=19|-defective)	0.000000%
P(branch_count=20|-defective)	7.894737%
P(branch_count=21|-defective)	0.000000%
P(branch_count=22|-defective)	5.555556%
P(branch_count=23|-defective)	0.000000%
P(branch_count=24|-defective)	10.256410%
P(branch_count=25|-defective)	8.108108%
P(branch_count=26|-defective)	0.000000%
P(branch_count=27|-defective)	0.000000%
P(branch_count=28|-defective)	2.777778%
P(branch_count=29|-defective)	8.333333%
P(branch_count=30|-defective)	5.555556%
P(branch_count=31|-defective)	8.108108%
P(branch_count=32|-defective)	15.555556%
P(branch_count=33|-defective)	0.000000%
P(branch_count=34|-defective)	19.642857%
P(branch_count=0|+defective)	2.777778%
P(branch_count=1|+defective)	17.391304%
P(branch_count=2|+defective)	12.195122%
P(branch_count=3|+defective)	11.627907%
P(branch_count=4|+defective)	2.857143%
P(branch_count=5|+defective)	7.894737%
P(branch_count=6|+defective)	10.256410%
P(branch_count=7|+defective)	10.322581%
P(branch_count=8|+defective)	8.108108%
P(branch_count=9|+defective)	7.894737%
P(branch_count=10|+defective)	2.777778%
P(branch_count=11|+defective)	5.555556%
P(branch_count=12|+defective)	23.711340%
P(branch_count=13|+defective)	2.857143%
P(branch_count=14|+defective)	5.555556%
P(branch_count=15|+defective)	2.702703%
P(branch_count=16|+defective)	5.263158%
P(branch_count=17|+defective)	18.840580%
P(branch_count=18|+defective)	5.555556%
P(branch_count=19|+defective)	0.000000%
P(branch_count=20|+defective)	5.263158%
P(branch_count=21|+defective)	0.000000%
P(branch_count=22|+defective)	2.777778%
P(branch_count=23|+defective)	0.000000%
P(branch_count=24|+defective)	5.128205%
P(branch_count=25|+defective)	2.702703%
P(branch_count=26|+defective)	0.000000%
P(branch_count=27|+defective)	0.000000%
P(branch_count=28|+defective)	5.555556%
P(branch_count=29|+defective)	22.916667%
P(branch_count=30|+defective)	2.777778%
P(branch_count=31|+defective)	2.702703%
P(branch_count=32|+defective)	11.111111%
P(branch_count=33|+defective)	0.000000%
P(branch_count=34|+defective)	21.428571%
loc_code_and_comment	
P(loc_code_and_comment=0|-defective)	79.423329%
P(loc_code_and_comment=1|-defective)	57.142857%
P(loc_code_and_comment=2|-defective)	0.000000%
P(loc_code_and_comment=3|-defective)	18.181818%

P(loc_code_and_comment=4|-defective)	33.333333%
P(loc_code_and_comment=5|-defective)	37.500000%
P(loc_code_and_comment=6|-defective)	10.000000%
P(loc_code_and_comment=7|-defective)	25.000000%
P(loc_code_and_comment=8|-defective)	18.181818%
P(loc_code_and_comment=9|-defective)	10.000000%
P(loc_code_and_comment=0|+defective)	19.528178%
P(loc_code_and_comment=1|+defective)	14.285714%
P(loc_code_and_comment=2|+defective)	0.000000%
P(loc_code_and_comment=3|+defective)	9.090909%
P(loc_code_and_comment=4|+defective)	13.333333%
P(loc_code_and_comment=5|+defective)	12.500000%
P(loc_code_and_comment=6|+defective)	10.000000%
P(loc_code_and_comment=7|+defective)	8.333333%
P(loc_code_and_comment=8|+defective)	9.090909%
P(loc_code_and_comment=9|+defective)	10.000000%
loc_comments	
P(loc_comments=0|-defective)	82.380952%
P(loc_comments=1|-defective)	61.458333%
P(loc_comments=2|-defective)	13.043478%
P(loc_comments=3|-defective)	4.545455%
P(loc_comments=4|-defective)	31.707317%
P(loc_comments=5|-defective)	27.272727%
P(loc_comments=6|-defective)	18.518519%
P(loc_comments=7|-defective)	11.538462%
P(loc_comments=8|-defective)	4.761905%
P(loc_comments=9|-defective)	4.000000%
P(loc_comments=10|-defective)	12.500000%
P(loc_comments=11|-defective)	0.000000%
P(loc_comments=12|-defective)	9.090909%
P(loc_comments=13|-defective)	4.545455%
P(loc_comments=14|-defective)	9.090909%
P(loc_comments=15|-defective)	4.761905%
P(loc_comments=16|-defective)	4.545455%
P(loc_comments=17|-defective)	9.090909%
P(loc_comments=18|-defective)	15.384615%
P(loc_comments=19|-defective)	4.761905%
P(loc_comments=20|-defective)	37.288136%
P(loc_comments=0|+defective)	14.603175%
P(loc_comments=1|+defective)	18.750000%
P(loc_comments=2|+defective)	4.347826%
P(loc_comments=3|+defective)	9.090909%
P(loc_comments=4|+defective)	21.951220%
P(loc_comments=5|+defective)	15.151515%
P(loc_comments=6|+defective)	11.111111%
P(loc_comments=7|+defective)	15.384615%
P(loc_comments=8|+defective)	4.761905%
P(loc_comments=9|+defective)	20.000000%
P(loc_comments=10|+defective)	8.333333%
P(loc_comments=11|+defective)	0.000000%
P(loc_comments=12|+defective)	4.545455%
P(loc_comments=13|+defective)	9.090909%
P(loc_comments=14|+defective)	4.545455%
P(loc_comments=15|+defective)	4.761905%
P(loc_comments=16|+defective)	9.090909%
P(loc_comments=17|+defective)	4.545455%
P(loc_comments=18|+defective)	11.538462%

P(loc_comments=19|+defective)	4.761905%
P(loc_comments=20|+defective)	30.508475%
cyclomatic_complexity	
P(cyclomatic_complexity=0|-defective)	16.129032%
P(cyclomatic_complexity=1|-defective)	10.000000%
P(cyclomatic_complexity=2|-defective)	72.108844%
P(cyclomatic_complexity=3|-defective)	7.142857%
P(cyclomatic_complexity=4|-defective)	3.571429%
P(cyclomatic_complexity=5|-defective)	12.903226%
P(cyclomatic_complexity=6|-defective)	3.703704%
P(cyclomatic_complexity=7|-defective)	3.571429%
P(cyclomatic_complexity=8|-defective)	3.571429%
P(cyclomatic_complexity=9|-defective)	45.977011%
P(cyclomatic_complexity=10|-defective)	7.142857%
P(cyclomatic_complexity=11|-defective)	13.513514%
P(cyclomatic_complexity=12|-defective)	40.625000%
P(cyclomatic_complexity=13|-defective)	28.571429%
P(cyclomatic_complexity=14|-defective)	12.195122%
P(cyclomatic_complexity=15|-defective)	21.621622%
P(cyclomatic_complexity=16|-defective)	0.000000%
P(cyclomatic_complexity=17|-defective)	0.000000%
P(cyclomatic_complexity=18|-defective)	8.823529%
P(cyclomatic_complexity=19|-defective)	0.000000%
P(cyclomatic_complexity=20|-defective)	14.285714%
P(cyclomatic_complexity=21|-defective)	3.571429%
P(cyclomatic_complexity=22|-defective)	6.666667%
P(cyclomatic_complexity=23|-defective)	26.666667%
P(cyclomatic_complexity=24|-defective)	29.787234%
P(cyclomatic_complexity=25|-defective)	10.000000%
P(cyclomatic_complexity=26|-defective)	6.666667%
P(cyclomatic_complexity=0|+defective)	3.225806%
P(cyclomatic_complexity=1|+defective)	6.666667%
P(cyclomatic_complexity=2|+defective)	10.884354%
P(cyclomatic_complexity=3|+defective)	3.571429%
P(cyclomatic_complexity=4|+defective)	7.142857%
P(cyclomatic_complexity=5|+defective)	6.451613%
P(cyclomatic_complexity=6|+defective)	3.703704%
P(cyclomatic_complexity=7|+defective)	7.142857%
P(cyclomatic_complexity=8|+defective)	7.142857%
P(cyclomatic_complexity=9|+defective)	25.287356%
P(cyclomatic_complexity=10|+defective)	3.571429%
P(cyclomatic_complexity=11|+defective)	18.918919%
P(cyclomatic_complexity=12|+defective)	20.312500%
P(cyclomatic_complexity=13|+defective)	26.785714%
P(cyclomatic_complexity=14|+defective)	26.829268%
P(cyclomatic_complexity=15|+defective)	10.810811%
P(cyclomatic_complexity=16|+defective)	0.000000%
P(cyclomatic_complexity=17|+defective)	0.000000%
P(cyclomatic_complexity=18|+defective)	17.647059%
P(cyclomatic_complexity=19|+defective)	0.000000%
P(cyclomatic_complexity=20|+defective)	14.285714%
P(cyclomatic_complexity=21|+defective)	7.142857%
P(cyclomatic_complexity=22|+defective)	10.000000%
P(cyclomatic_complexity=23|+defective)	17.777778%
P(cyclomatic_complexity=24|+defective)	17.021277%
P(cyclomatic_complexity=25|+defective)	6.666667%
P(cyclomatic_complexity=26|+defective)	10.000000%

design_complexity	
P(design_complexity=0|-defective)	17.241379%
P(design_complexity=1|-defective)	71.612903%
P(design_complexity=2|-defective)	8.333333%
P(design_complexity=3|-defective)	8.333333%
P(design_complexity=4|-defective)	4.166667%
P(design_complexity=5|-defective)	8.333333%
P(design_complexity=6|-defective)	48.148148%
P(design_complexity=7|-defective)	31.034483%
P(design_complexity=8|-defective)	30.612245%
P(design_complexity=9|-defective)	36.585366%
P(design_complexity=10|-defective)	15.151515%
P(design_complexity=11|-defective)	19.354839%
P(design_complexity=12|-defective)	0.000000%
P(design_complexity=13|-defective)	0.000000%
P(design_complexity=14|-defective)	0.000000%
P(design_complexity=15|-defective)	13.333333%
P(design_complexity=16|-defective)	11.111111%
P(design_complexity=17|-defective)	4.166667%
P(design_complexity=18|-defective)	7.407407%
P(design_complexity=19|-defective)	7.407407%
P(design_complexity=20|-defective)	27.027027%
P(design_complexity=21|-defective)	6.666667%
P(design_complexity=22|-defective)	8.000000%
P(design_complexity=0|+defective)	10.344828%
P(design_complexity=1|+defective)	14.838710%
P(design_complexity=2|+defective)	4.166667%
P(design_complexity=3|+defective)	4.166667%
P(design_complexity=4|+defective)	8.333333%
P(design_complexity=5|+defective)	4.166667%
P(design_complexity=6|+defective)	25.925926%
P(design_complexity=7|+defective)	32.758621%
P(design_complexity=8|+defective)	26.530612%
P(design_complexity=9|+defective)	12.195122%
P(design_complexity=10|+defective)	21.212121%
P(design_complexity=11|+defective)	12.903226%
P(design_complexity=12|+defective)	0.000000%
P(design_complexity=13|+defective)	0.000000%
P(design_complexity=14|+defective)	0.000000%
P(design_complexity=15|+defective)	16.666667%
P(design_complexity=16|+defective)	11.111111%
P(design_complexity=17|+defective)	8.333333%
P(design_complexity=18|+defective)	14.814815%
P(design_complexity=19|+defective)	14.814815%
P(design_complexity=20|+defective)	16.216216%
P(design_complexity=21|+defective)	23.333333%
P(design_complexity=22|+defective)	8.000000%
essential_complexity	
P(essential_complexity=0|-defective)	16.666667%
P(essential_complexity=1|-defective)	5.263158%
P(essential_complexity=2|-defective)	10.000000%
P(essential_complexity=3|-defective)	51.724138%
P(essential_complexity=4|-defective)	28.125000%
P(essential_complexity=5|-defective)	13.793103%
P(essential_complexity=6|-defective)	30.000000%
P(essential_complexity=7|-defective)	13.043478%
P(essential_complexity=8|-defective)	0.000000%

P(essential_complexity=9|-defective)	0.000000%
P(essential_complexity=10|-defective)	0.000000%
P(essential_complexity=11|-defective)	0.000000%
P(essential_complexity=12|-defective)	4.761905%
P(essential_complexity=13|-defective)	17.857143%
P(essential_complexity=14|-defective)	9.523810%
P(essential_complexity=15|-defective)	5.263158%
P(essential_complexity=16|-defective)	5.000000%
P(essential_complexity=17|-defective)	21.428571%
P(essential_complexity=18|-defective)	5.000000%
P(essential_complexity=0|+defective)	12.500000%
P(essential_complexity=1|+defective)	5.263158%
P(essential_complexity=2|+defective)	5.000000%
P(essential_complexity=3|+defective)	18.965517%
P(essential_complexity=4|+defective)	18.750000%
P(essential_complexity=5|+defective)	27.586207%
P(essential_complexity=6|+defective)	13.333333%
P(essential_complexity=7|+defective)	13.043478%
P(essential_complexity=8|+defective)	0.000000%
P(essential_complexity=9|+defective)	0.000000%
P(essential_complexity=10|+defective)	0.000000%
P(essential_complexity=11|+defective)	0.000000%
P(essential_complexity=12|+defective)	14.285714%
P(essential_complexity=13|+defective)	21.428571%
P(essential_complexity=14|+defective)	9.523810%
P(essential_complexity=15|+defective)	5.263158%
P(essential_complexity=16|+defective)	10.000000%
P(essential_complexity=17|+defective)	17.857143%
P(essential_complexity=18|+defective)	10.000000%
loc_executable	
P(loc_executable=0|-defective)	81.322957%
P(loc_executable=1|-defective)	33.333333%
P(loc_executable=2|-defective)	50.000000%
P(loc_executable=0|+defective)	18.547341%
P(loc_executable=1|+defective)	61.111111%
P(loc_executable=2|+defective)	33.333333%
halstead_content	
P(halstead_content=0|-defective)	81.432361%
P(halstead_content=1|-defective)	57.575758%
P(halstead_content=2|-defective)	37.500000%
P(halstead_content=0|+defective)	18.435013%
P(halstead_content=1|+defective)	39.393939%
P(halstead_content=2|+defective)	50.000000%
halstead_difficulty	
P(halstead_difficulty=0|-defective)	85.446686%
P(halstead_difficulty=1|-defective)	43.529412%
P(halstead_difficulty=2|-defective)	37.500000%
P(halstead_difficulty=0|+defective)	14.409222%
P(halstead_difficulty=1|+defective)	55.294118%
P(halstead_difficulty=2|+defective)	56.250000%
halstead_effort	
P(halstead_effort=0|-defective)	80.690537%
P(halstead_effort=1|-defective)	37.500000%
P(halstead_effort=2|-defective)	40.000000%
P(halstead_effort=0|+defective)	19.181586%
P(halstead_effort=1|+defective)	50.000000%
P(halstead_effort=2|+defective)	40.000000%

halstead_error_est	
P(halstead_error_est=0|-defective)	81.001284%
P(halstead_error_est=1|-defective)	25.000000%
P(halstead_error_est=2|-defective)	50.000000%
P(halstead_error_est=0|+defective)	18.870347%
P(halstead_error_est=1|+defective)	66.666667%
P(halstead_error_est=2|+defective)	25.000000%
halstead_length	
P(halstead_length=0|-defective)	81.056701%
P(halstead_length=1|-defective)	35.714286%
P(halstead_length=2|-defective)	40.000000%
P(halstead_length=0|+defective)	18.814433%
P(halstead_length=1|+defective)	57.142857%
P(halstead_length=2|+defective)	40.000000%
halstead_level	
P(halstead_level=0|-defective)	75.981162%
P(halstead_level=1|-defective)	97.260274%
P(halstead_level=2|-defective)	83.333333%
P(halstead_level=0|+defective)	23.861852%
P(halstead_level=1|+defective)	2.054795%
P(halstead_level=2|+defective)	8.333333%
halstead_prog_time	
P(halstead_prog_time=0|-defective)	80.690537%
P(halstead_prog_time=1|-defective)	37.500000%
P(halstead_prog_time=2|-defective)	40.000000%
P(halstead_prog_time=0|+defective)	19.181586%
P(halstead_prog_time=1|+defective)	50.000000%
P(halstead_prog_time=2|+defective)	40.000000%
halstead_volume	
P(halstead_volume=0|-defective)	80.897436%
P(halstead_volume=1|-defective)	27.272727%
P(halstead_volume=2|-defective)	50.000000%
P(halstead_volume=0|+defective)	18.974359%
P(halstead_volume=1|+defective)	63.636364%
P(halstead_volume=2|+defective)	25.000000%
num_operands	
P(num_operands=0|-defective)	80.976864%
P(num_operands=1|-defective)	30.769231%
P(num_operands=2|-defective)	50.000000%
P(num_operands=0|+defective)	18.894602%
P(num_operands=1|+defective)	61.538462%
P(num_operands=2|+defective)	25.000000%
num_operators	
P(num_operators=0|-defective)	81.161290%
P(num_operators=1|-defective)	33.333333%
P(num_operators=2|-defective)	40.000000%
P(num_operators=0|+defective)	18.709677%
P(num_operators=1|+defective)	60.000000%
P(num_operators=2|+defective)	40.000000%
num_unique_operands	
P(num_unique_operands=0|-defective)	81.973684%
P(num_unique_operands=1|-defective)	35.483871%
P(num_unique_operands=2|-defective)	50.000000%
P(num_unique_operands=0|+defective)	17.894737%
P(num_unique_operands=1|+defective)	61.290323%
P(num_unique_operands=2|+defective)	25.000000%
num_unique_operators	

P(num_unique_operators=0|-defective)	44.067797%
P(num_unique_operators=1|-defective)	58.974359%
P(num_unique_operators=2|-defective)	16.666667%
P(num_unique_operators=3|-defective)	12.195122%
P(num_unique_operators=4|-defective)	34.693878%
P(num_unique_operators=5|-defective)	17.073171%
P(num_unique_operators=6|-defective)	9.523810%
P(num_unique_operators=7|-defective)	2.631579%
P(num_unique_operators=8|-defective)	8.108108%
P(num_unique_operators=9|-defective)	8.571429%
P(num_unique_operators=10|-defective)	2.857143%
P(num_unique_operators=11|-defective)	3.030303%
P(num_unique_operators=12|-defective)	40.845070%
P(num_unique_operators=13|-defective)	5.714286%
P(num_unique_operators=14|-defective)	2.941176%
P(num_unique_operators=15|-defective)	3.030303%
P(num_unique_operators=16|-defective)	79.878049%
P(num_unique_operators=17|-defective)	3.125000%
P(num_unique_operators=18|-defective)	2.941176%
P(num_unique_operators=19|-defective)	42.105263%
P(num_unique_operators=20|-defective)	60.439560%
P(num_unique_operators=21|-defective)	58.426966%
P(num_unique_operators=22|-defective)	48.648649%
P(num_unique_operators=23|-defective)	35.616438%
P(num_unique_operators=24|-defective)	58.947368%
P(num_unique_operators=25|-defective)	47.945205%
P(num_unique_operators=26|-defective)	35.714286%
P(num_unique_operators=27|-defective)	37.704918%
P(num_unique_operators=28|-defective)	32.075472%
P(num_unique_operators=29|-defective)	26.000000%
P(num_unique_operators=30|-defective)	21.739130%
P(num_unique_operators=31|-defective)	12.820513%
P(num_unique_operators=0|+defective)	5.084746%
P(num_unique_operators=1|+defective)	2.564103%
P(num_unique_operators=2|+defective)	20.833333%
P(num_unique_operators=3|+defective)	14.634146%
P(num_unique_operators=4|+defective)	4.081633%
P(num_unique_operators=5|+defective)	9.756098%
P(num_unique_operators=6|+defective)	19.047619%
P(num_unique_operators=7|+defective)	18.421053%
P(num_unique_operators=8|+defective)	10.810811%
P(num_unique_operators=9|+defective)	5.714286%
P(num_unique_operators=10|+defective)	11.428571%
P(num_unique_operators=11|+defective)	6.060606%
P(num_unique_operators=12|+defective)	16.901408%
P(num_unique_operators=13|+defective)	8.571429%
P(num_unique_operators=14|+defective)	8.823529%
P(num_unique_operators=15|+defective)	6.060606%
P(num_unique_operators=16|+defective)	1.829268%
P(num_unique_operators=17|+defective)	3.125000%
P(num_unique_operators=18|+defective)	8.823529%
P(num_unique_operators=19|+defective)	5.263158%
P(num_unique_operators=20|+defective)	6.593407%
P(num_unique_operators=21|+defective)	7.865169%
P(num_unique_operators=22|+defective)	10.810811%
P(num_unique_operators=23|+defective)	23.287671%
P(num_unique_operators=24|+defective)	9.473684%

P(num_unique_operators=25|+defective)	10.958904%
P(num_unique_operators=26|+defective)	21.428571%
P(num_unique_operators=27|+defective)	13.114754%
P(num_unique_operators=28|+defective)	11.320755%
P(num_unique_operators=29|+defective)	14.000000%
P(num_unique_operators=30|+defective)	13.043478%
P(num_unique_operators=31|+defective)	10.256410%
loc_total	
P(loc_total=0|-defective)	81.592689%
P(loc_total=1|-defective)	36.363636%
P(loc_total=2|-defective)	42.857143%
P(loc_total=0|+defective)	18.276762%
P(loc_total=1|+defective)	59.090909%
P(loc_total=2|+defective)	42.857143%
defective	
P(+defective)	80.456853%
P(-defective)	19.543147%