Evaluating Various Learning Techniques for Efficiency

Mehnaz Khan, S.M.K. Quadri

Abstract— Machine learning is a vast field and has a broad range of applications including natural language processing, medical diagnosis, search engines, speech recognition, game playing and a lot more. A number of machine learning algorithms have been developed for different applications. However no single machine learning algorithm can be used appropriately for all learning problems. It is not possible to create a general learner for all problems because there are varied types of real world datasets that cannot be handled by a single learner. In this paper we present an evaluation of various state-of-the-art machine learning algorithms using WEKA (Waikato Environment for Knowledge Analysis) for a real world learning problem- credit approval used in banks. First we provide a brief description about WEKA. After that we describe the learning problem and the dataset that we have used in our experiments. Later we explain the machine learning methods that we have evaluated. Finally we provide description about our experimental setup and procedure and discuss the conclusion and the result.

Index Terms—credit approval, machine learning, test sets and training sets.

I. INTRODUCTION

WEKA (http://www.cs.waikato.ac.nz/ml/weka/) is an open source software which consists of a collection of state-of-the-art machine learning algorithms and data preprocessing tools. It has been developed at the University of Waikato in New Zealand. It is designed in such a way that allows users to try all machine learning algorithms on new datasets easily. The WEKA system is written in Java. It can be used for a variety of tasks. It provides an implementation of state-of-the-art machine learning algorithms that we can apply to our datasets for extracting information about the data or we can apply several algorithms to our dataset for comparing their performance and choosing one for prediction. It also provides a number of tools for data preprocessing i.e. transforming datasets and analyzing the resulting classifier. Such tools are called filters. Thus the main focus of WEKA is on the learning methods and the filters. There are two ways in which we can invoke these methods: either by using command line options or by using the interactive graphical user interface. In our experiments we have used graphical user interface of WEKA because it is much more convenient. We have used WEKA 3.7.7.

Manuscript received on December, 2012.

Mehnaz Khan, Department of Computer Science, University of Kashmir, Srinagar, India.

Dr. S.M.K. Quadri, Department of Computer Science, University of Kashmir, Srinagar, India.

II. LEARNING PROBLEM AND THE DATASET USED IN OUR EXPERIMENTS

For evaluating the efficiency of the state-of-the-art machine learning algorithms, we used credit approval problem which is used in banks.

A. Understanding the problem

A financial institution, e.g. a bank, gives its customers an amount of money and expects them to pay it back in installments along with interest. This amount of money is called credit. However, before approving any credit application, it is necessary for the bank to make sure that the customer will pay the whole amount back. The bank should be able to predict in advance the risk associated with a loan. It is done for making sure that the bank will make a profit and that the customers get a loan within his or her financial capacity. This calls for a need to find out efficient methods for automatic credit approval that can help the authorities in assessing credit applications effectively.

B. Risk involved in credit approval

Here the risk involved refers to the risk of loss to the financial institution if they lend the money to the customers who fail to pay the amount back [3]. The reason for this default can be anything like inability, unwillingness, etc. The bank should be able to predict in advance the risk associated with a loan. It is necessary for the lenders to calculate the probability of risk involved so that they can make correct decisions regarding the approval of the credit.

C. Credit Evaluation Method

Credit evaluation or credit scoring [1] is an evaluation system that is used for improving or increasing the abilities of the credit lenders in deciding the probability of the credit risk of a customer. In this method, risk is calculated by the bank on the basis of the amount of credit and the information about the customer. The information about the customer includes data that the bank has access to and is relevant in calculating financial capacity of the customer. This data consists of income, savings, collaterals, profession, age, past financial history, and so forth. The bank has a record of past loans containing such customer data and whether the loan was paid back or not. From this data of particular applications, the aim is to infer a general rule coding the association between a customer's attributes and his risk. That is, the machine learning system fits a model to the past data to be able to calculate the risk for a new application and then decides to accept or refuse it accordingly.

This process can be carried out in two ways. The first is called deductive credit scoring in which points are assigned to relevant customer attributes. These points are then used to

form a credit score. The experience of the credit professionals is used to select the relevant attributes, determine the points and calculate the credit scores. Another type of credit scoring is empirical credit scoring in which the past data about the customers is analyzed and used to construct the scoring models. This is done by using appropriate algorithms for identifying characteristics relating to the credit risk of customers. These scoring models are then used to calculate the credit risk of new customers [7].

Bank professionals then use these credit scores to indicate the level of the credit risk and then decide accordingly whether to approve the credit to the customers or not and at what interest rate the credit should be approved. For the low risk customers, the chances of getting the credit at lower interest rates and on longer repayment terms are higher. However, if the risk of the customers is high but lower than the cut-off credit risk, the customer is not disqualified from getting the credit but in this case the bank professionals review the customer application more carefully before deciding whether to approve or deny the credit request. If the credit in approved in case of such customers, it is given on higher interest rates and shorter repayment terms as compared to the low-risk customers.

D. Automating the process

The above stated processes i.e. credit scoring and approval can be carried out more efficiently if they are done automatically using computers. Automatic scoring and approval helps in gathering the necessary information quickly and speeds up the process of evaluation and determining whether to approve or deny credit applications. Automating this process does not mean that it can take place of the credit professional but it can help in making rapid decisions. The credit applications that are identified as good credit risk and those as bad credit risk may be automatically approved, or denied, while those of intermediate risk may still be passed to credit analysts for more detailed review before deciding whether to approve or deny credit. This can reduce the number of credit applications that need more detailed review and reduce the wastage of time, thus allowing credit analysts to concentrate only on those credit applications that are difficult or important.

III. DESCRIPTION OF THE DATASET USED

The dataset that we used for our experiments for evaluating the learning algorithms was provided originally by Quinlan in his studies of ID3 and C4.5 system in 1987 and 1992, to induce decision trees for assessing credit card applications. It is the Australian Credit Approval dataset from UCI Repository of Machine Learning Databases and Domain theories. The dataset consists of 15 attributes and a class label attribute. Before being made available to use, the attributes of the dataset have been modified to keep the data confidential. Their names and values are converted to some symbols that don't have any meaning. The class attribute can have two values, either + or -, and the type of other attributes can be either continuous or nominal. The dataset consists of 490 instances with 44.5% being positive (credit approved), 55.5% being negative (credit denied) and 5% having missing values. Table I shows the attribute names and attribute types of the dataset and Table II shows distributions of "+" and "-" values.

Table I: Australian Credit Approval Dataset

Attribute	Type
A1	nominal
A2	continuous
A3	continuous
A4	nominal
A5	nominal
A6	nominal
A7	nominal
A8	continuous
A9	nominal
A10	nominal
A11	continuous
A12	nominal
A13	nominal
A14	continuous
A15	continuous
Class	nominal

Table II: Class distribution

Class	No. of Instances
+	218
-	272

The "class" attribute can take two values i.e. "+" and "-" as stated earlier. The two values represent the low-risk and high-risk customers here. For low-risk customers, "class" attribute takes "+" value meaning credit can be approved for such customers and vice-versa for high-risk customers. This makes our learning problem a classification problem.

WEKA provides a number of classification algorithms. Hence our experiments use this dataset for evaluation of various classification learning algorithms. For our experiments we divided our dataset into training and test sets by the same procedure as described in Section 5.1.

IV. SPLITTING THE DATASET

WEKA allows us to use the dataset in a number of ways in our experiments. We can perform cross-validation, percentage split or we can use the supplied test set option. For using the "supplied test set" option we need to split our dataset into appropriate quantities of training and test sets. In the "Explorer" interface, we first load our dataset in the "Preprocess" panel. This is done either by loading an ARFF file or CSV file. We can also load our dataset directly from a URL or database. In our example, we have loaded the dataset using a URL as shown in Fig. 1. Next we have to split our dataset into two sets, 30% testing and 70% training. To do this we first randomize the dataset by choosing Randomize filter. This creates a random permutation. Next we apply RemovePercentage filter on our dataset keeping percentage as 30 and we save the dataset as a training set. This is shown in Fig. 2.

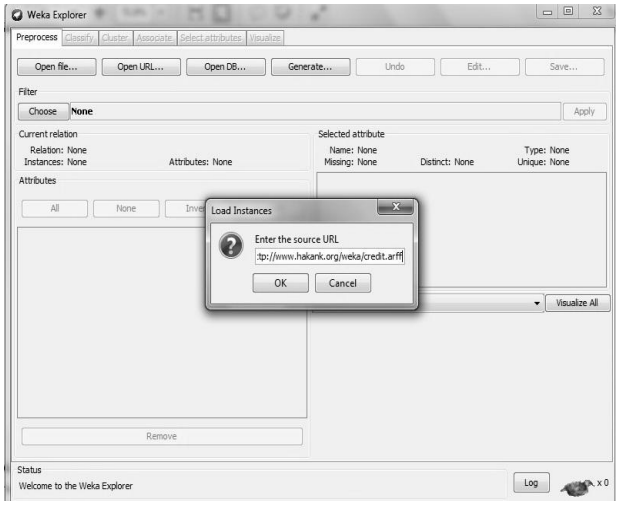

Fig. 1: Loading Dataset from URL

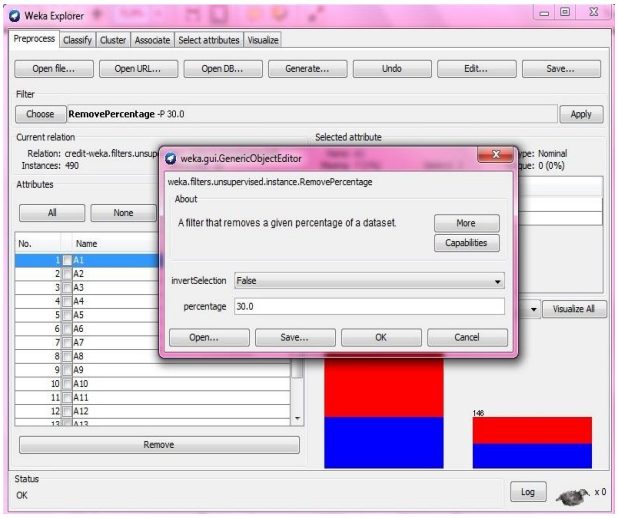

Fig. 2: Using RemovePercentage filter.

Next we undo the change and again apply the same filter but changing the invertSelection option to "True". This picks the rest of the dataset i.e. 30% and is saved as a testing set.

This way our dataset gets divided into 30% testing and 70% training set. Next to use our sets in the experiments we choose the training set and move to the "Classify" panel and choose the procedure that we have to use and start the experiment. After that we apply the same procedure on our testing set to check what it predicts on the unseen data. For that, we select "supplied test set" and choose the testing dataset that we created. We run the algorithm again and we notice the differences in the confusion matrix and the accuracy.

V. LEARNING METHODS CHOSEN FOR EVALUATION

As discussed above, the learning problem that we have used in our experiments is a classification problem. Therefore, we have used WEKA's classification algorithms for evaluation of the chosen dataset. In our experiments, we have chosen 10 learning algorithms from 6 different types. These are given below:

- Rule based
  - Zero R
  - One R
- Bayes Rule
  - NaiveBayes
  - NaiveBayesUpdateable
- Functions
  - Multilayer Perceptron
- Lazy Learners
  - KStar (K*)
- Tree Based
  - J48
  - RandomForest
- Meta-Algorithm
  - AdaBoostM1
  - Bagging

Zero R and One R are rule-based algorithms. A rule-based algorithm uses rules to make deductions or choices. The classification method uses an algorithm to generate rules from the sample data. These rules are then applied to new data. OneR (One Rule) is a simple classification algorithm that generates a one-level decision tree. OneR selects the rule with the lowest error rate. In the event that two or more rules have the same error rate, the rule is chosen at random. In the implementation of WEKA, the OneR algorithm picks the rule with the highest number of correct instances, not lowest error rate, and does not randomly select a rule when error rates are identical. Zero Regression (ZeroR) is a pseudo-regression method that always builds models with cross-validation coefficient Q2=0. The Naive Bayes [2] algorithm is a mchine learning algorithm that works on conditional probabilities. It is based on Bayes' Theorem that computes probability by counting the frequency combinations of values in the data. It is used to calculate the probability of the occurring of an event provided the probability of an event that has occurred before is given. NaiveBayesUpdateable is the updateable version of NaiveBayes which uses estimator classes. It uses 0.1 precision in case of numeric attributes. For classifying instances, Multilayer Perceptron classifier makes use of back propagation. It is a feed forward neural network which is used to map the inputs to the outputs. It is a directed graph which has many layers of nodes. Each layer is fully connected to the next layer. MLP is used for solving linearly inseparable problems [4]. J48 and Random forest are the machine learning algorithms that are based on decision tree. In WEKA C4.5 learner is implemented as J4.8. C4.5 is used for generating a decision tree developed by Ross Quinlan [6]. Random forest is a new learning algorithm that implements Breiman's random forest algorithm. It is used for classification and regression. K* is one of the lazy learning approaches. These are also called memory-based methods. These methods work by learning the structure of a domain by storing examples with their classification [8]. Bootstrap aggregating (bagging) and boosting are the learning techniques that are used for improving the performance of prediction of trees. AdaBoost [5] is a boosting algorithm developed by Freund and Schapire. It is used for minimizing the error of learning algorithms that produces classifiers having slightly better performance than random guessing. AdaBoostM1 is a version of AdaBoost algorithm. Bagging is based on making a number of samples of the training set.

VI. EXPERIMENTAL SETUP

This section shows the performance evaluation of all the learning algorithms mentioned above. The evaluation is shown on the dataset chosen i.e. Credit Dataset. As already stated, we carried our experiments using WEKA. It provides a number of measures of evaluation that can be used to check the performance of the algorithms. When an experiment is

run, results are displayed on "Classifier Output" area. This area has several sections showing different results. First is run information. It is a list of information that shows various options of learning, attributes, name of the relation and mode of testing used in the process. After this information about classifier is displayed in a textual form that depicts the classification model generated on training data. Then information about the accuracy of the classifiers is displayed. It shows the accuracy of the classifier in predicting the true class of the instances under the chosen test mode. After that detailed information about the accuracy of a classifier in predicting the class is shown. Finally, confusion matrix is displayed that shows the number of instances assigned to each class. The evaluation measures that we used to compare the learners are number of correctly classified instances, time taken to build the model, F-Measure.

Before using our dataset in the experiments, we used the method discussed in Section IV for splitting it into 70% training set and 30% test set. First we loaded the actual dataset into the WEKA from URL (http://www.hakank.org/weka/credit.arff). Then after applying the splitting procedure, we saved both these sets as separate files, trainingcredit.arff and testingcredit.arff. For all experiments we used these two files. Fig. 3 shows the actual dataset, Fig. 4 shows trainingcredit.arff file and Fig. 5 shows testingcredit.arff file.

A. Experimental Procedure

- In our experiments, for each learner, we first load trainingcredit.arff file into WEKA through "Preprocess panel.
- Then in the " Classify " panel we choose the classification algorithm to be implemented and start the analysis using 10-fold cross validation.
- After that we load the file trainingcredit.arff using the Supplied test set option" and then start the analysis again.
- Finally, we analyze the results on the basis of the evaluation measures discussed above.
- The same process is repeated for all the classification algorithms that are to be evaluated.

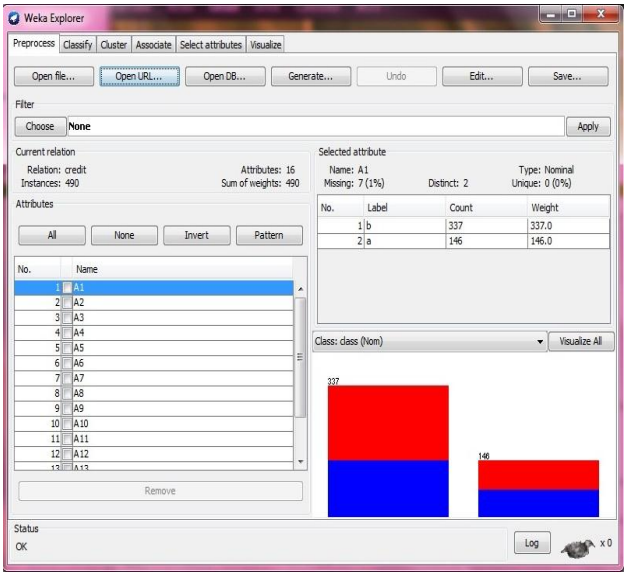

Fig. 3: Credit Dataset

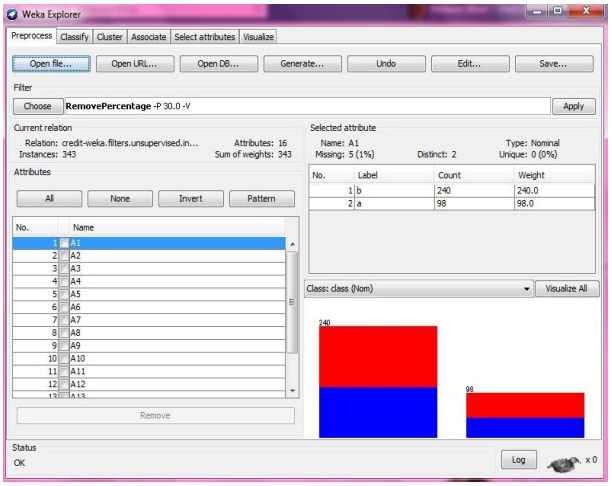

Fig. 4: trainingcredit.arff file

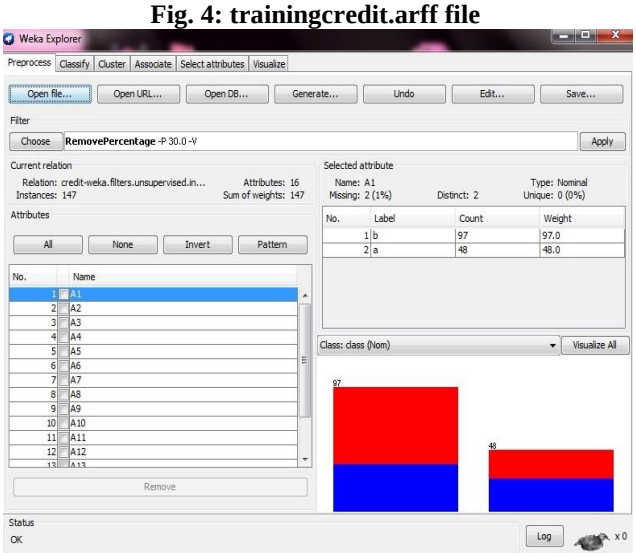

Fig. 5: testingcredit.arff file

B. Experimental Results

In this Section we show the results of the learning algorithms on our dataset.

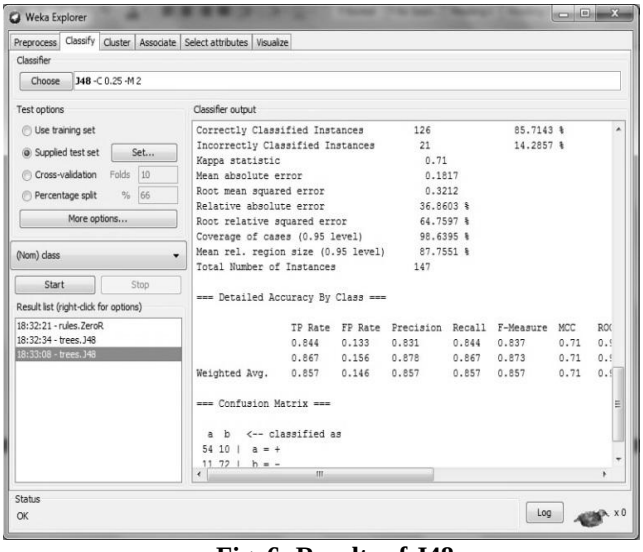

Fig. 6: Results of J48

Correctly Classified Instances (%) = 85.7143
Incorrectly Classified Instances (%) = 14.2857
Kappa Statistic = 0.71
Mean Absolute Error = 0.1817
F-Measure = 0.837
Time taken to build the Model = 0.01 seconds

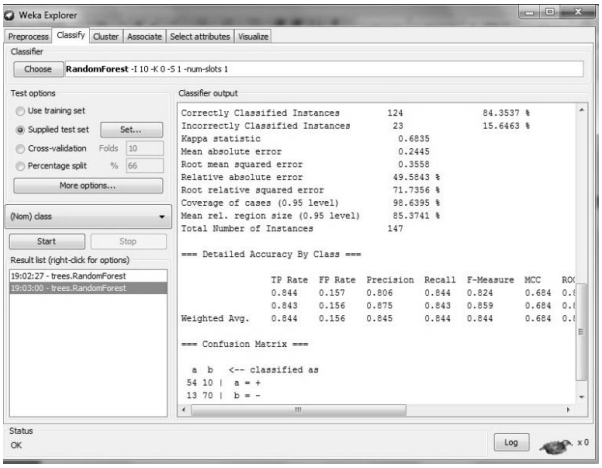

Fig. 7: Results of RandomForest

Correctly Classified Instances (%) = 84.3537
Incorrectly Classified Instances (%) = 15.6463
Kappa Statistic = 0.6835
Mean Absolute Error = 0.2445
F-Measure = 0.824
Time Taken to build the Model = 0.05 seconds

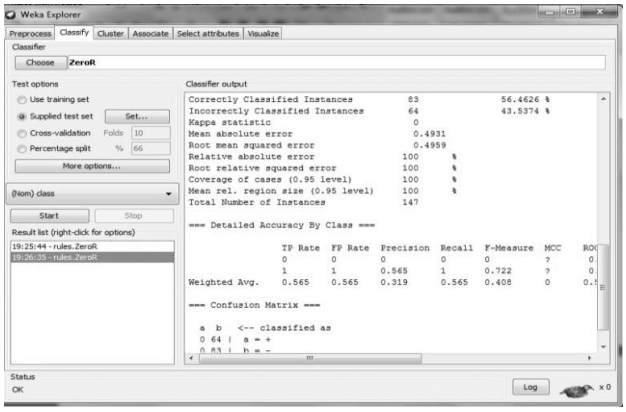

Fig. 8: Results of ZeroR

Correctly Classified Instances (%) = 56.4626
Incorrectly Classified Instances (%) = 43.5374
Kappa Statistic = 0
Mean Absolute Error = 0.4931
F-Measure = 0
Time Taken to build the Model = 0 seconds

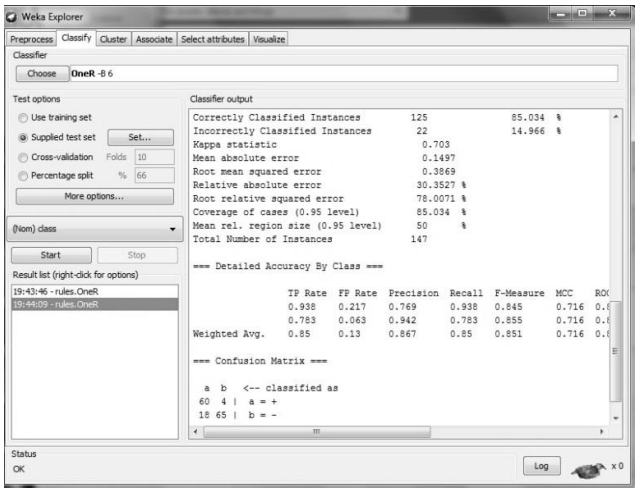

Fig. 9: Results of OneR

Correctly Classified Instances (%) = 85.034
Incorrectly Classified Instances (%) = 14.966
Kappa Statistic = 0.703
Mean Absolute Error = 0.1497
F-Measure = 0.845
Time Taken to build the Model = 0 seconds

For rest of the algorithms we have shown only the results except the figures.

Results of NaiveBayes

Correctly Classified Instances (%) = 79.5918
Incorrectly Classified Instances (%) = 20.4082
Kappa Statistic = 0.5727
Mean Absolute Error = 0.21
F-Measure = 0.732
Time Taken to build the Model = 0.01 seconds

Results of NaiveBayesUpdateable

Correctly Classified Instances (%) = 79.5918
Incorrectly Classified Instances (%) = 20.4082
Kappa Statistic = 0.5727
Mean Absolute Error = 0.21
F-Measure = 0.732
Time Taken to build the Model = 0.01 seconds

Results of AdaBoostM1

Correctly Classified Instances (%) = 84.3537
Incorrectly Classified Instances (%) = 15.6463
Kappa Statistic = 0.689
Mean Absolute Error = 0.211
F-Measure = 0.837
Time Taken to build the Model = 0.04 seconds

Results of Bagging

Correctly Classified Instances (%) = 84.3537
Incorrectly Classified Instances (%) = 15.6463
Kappa Statistic = 0.6879
Mean Absolute Error = 0.2196
F-Measure = 0.835
Time Taken to build the Model = 0.05 seconds

Results of MultiLayerPerceptron

Correctly Classified Instances (%) = 86.3946
Incorrectly Classified Instances (%) = 13.6054
Kappa Statistic = 0.7252
Mean Absolute Error = 0.1508
F-Measure = 0.848
Time Taken to build the Model = 0.01 seconds

Results of KStar

Correctly Classified Instances (%) = 71.4286
Incorrectly Classified Instances (%) = 28.5714
Kappa Statistic = 0.4017
Mean Absolute Error = 0.2896
F-Measure = 0.625
Time Taken to build the Model = 0 seconds

VII. CONCLUSION

In this section we show the results in the form of charts and tables. Fig. 10 shows the comparison of all the algorithms with respect to the time taken to build the model.

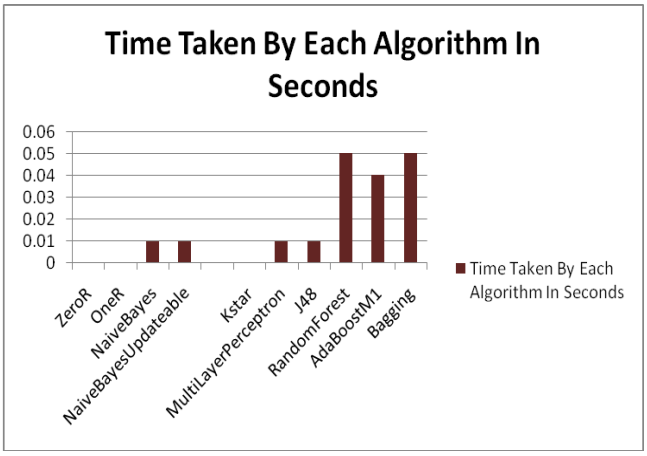

Fig. 10: Time chart of Algorithms

Fig. 11 shows the comparison based about the number of correctly classified instances by each learning algorithm.

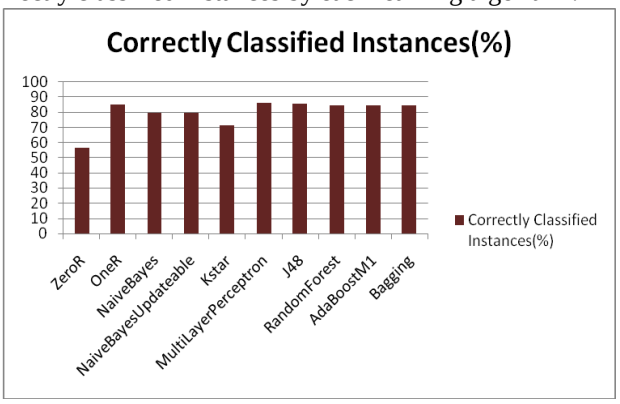

Fig. 11: Comparison of Algorithms By Percentage Of Correct Instances

In Table III we have summarized three main measures of evaluation for each algorithm i.e. time taken to build the model, number of correctly classified instances, and F-Measure.

Table III: Comparison of Algorithms

Algorithms	**Time Taken to Build Models (sec)**	**Correctly Classified Instances(%)**	**F-Measure**
J48	0.01	85.7143	0.837
RandomForest	0.05	84.3537	0.824
ZeroR	0	56.4626	0
OneR	0	85.034	0.845
NaiveBayes	0.01	79.5918	0.732
NaiveBayesUpdateable	0.01	79.5918	0.732
KStar	0	71.4286	0.625
MutilayerPerceptron	0.01	86.3946	0.848
AdaBoostM1	0.04	84.3537	0.837
Bagging	0.05	84.3537	0.835

It shows that RandomForest and Bagging take maximum amount of time to build the model i.e. 0.05 seconds. Next highest is 0.04 taken by AdaBoostM1. NaiveBayes, NaiveBayesUpdateable and MultiLayerPerceptron take 0.01 seconds and the remaining take 0 seconds to build the model. In terms of second measure of evaluation, MultiLayerPerceptron has the highest percentage of correctly labeled instances and has the best F-Measure among all.

Hence, we conclude that MultiLayerPerceptron has performed better than all the other classifiers in the analysis of our dataset.

ACKNOWLEDGMENT

The authors are thankful to the faculty of the Department of Computer Science, University of Kashmir for their constant support.

REFERENCES

[1] Hand, David J. (1998): "Reject inference in credit operations," in Credit Risk Modeling: Design and Application (ed. E. Mays), 181-190, AMACOM.
[2] Murphy, K.P.(2006) Naïve Bayes Classifiers.
[3] K.H. Ng, Commercial Banking in Singapore. Singapore: Addison Wesley, 1996, pp. 252-253.
[4] Steinwender, J. and Bitzer, S. Multilayer Perceptrons, A discussion of The Algebraic Mind 2003, University of Osnabrueck, (2003).
[5] Freund,Y.and Schapire, R. Experiments with a new Boosting Algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, 148-156. (1996).
[6] Quinlan, J. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, 1993.
[7] Liu, Y. New Issues in Credit Scoring Applications (2001).
[8] Van den Bosch, A., Daelemans, W. and Weijters, A. 1996. Morphological analysis as classification: an inductive-learning approach. In K. Ofiazer and H. Somers, editors, Proceedings of the Second International Conference on New Methods in Natural Language Processing.