**PROJECT REPORT ON**

**MACHINE LEARNING ON RAW USER SMS FOR LOAN RISK PREDICTION**

Under the guidance of **Dr. Manish Kumar**



**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY**

**ALLAHABAD**

Jan – May, 2016

Submitted by:

Amit Bhardwaj IIT2013056

Abhishek Jaiswal IIT2013129

Shubham Chaudhary IIT2013149

**Certificate from Supervisor**

I do hereby recommend that the mini project report prepared under my supervision, titled “Machine Learning for Loan Risk Prediction” be accepted for the completion sixth semester.

Date: 06.05.2016 **Guide’s name**

Place: Allahabad

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Manish Kumar

**Acknowledgement**

We owe special debt of gratitude to **Dr. Manish Kumar**, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only because of his cognizant efforts that our project is gradually seeing the light of the day.

**Abstract**

Traditional loan taking process involves checking credit worthiness by pulling out one’s credit report and score from various sources and thus giving you credit score.

But, if you do not have regular income, proper documentation to show income flow or have never taken a loan before then getting a loan from a commercial bank may be difficult.

To tackle this issue financial institutions are considering tapping into non-traditional data sources to build a credit report, by using individual’s personal information or any data available to measure their credit worthiness. For instance, a person’s SMS, emails, online transactions etc.

Financial state of a person can be judged from the SMS messages that you receive for your each transaction from telecom companies, banks and e-retailers etc.

This project involves creating a useful data set by mining users SMS messages and predicting the credit worthiness and hence the loan risk factor by using Data Mining and Machine Learning approach.

**Table of Content**

1. Problem Definition
2. Introduction
3. Literature Survey
4. Software Requirement
5. Methodology
   1. Data Collection and Preprocessing
   2. Attribute Extraction using Data Mining
   3. ANN(Aritificial Neural Networks)
   4. Decision Tree
   5. KNN(K nearest neighbours)
   6. SVM(Support Vector Machine)
6. Activity Time Chart
7. Conclusions
8. Reference

**1.** **Problem Definition**

Targeting specific messages such as recharge SMSes, Bank transactions (NEFT), ATM transactions containing amount debited or credited, EMI messages to know whether user has taken previous loans etc. out of 3 lac SMS to make our sampled database.

Extraction of amount, company name, vendors associated with each messages we are targeting and resolving various ambiguities faced in the process such as:

* Missing or Null values in amount field.
* Blank Messages.
* Redundant fields.

Automating the process of discovering the deserving applicant and rejecting the risky ones by application of various machine learning approach on the training data set we generate by the SMS mining processes in above steps.

**2. Introduction**

While other media may be used more than SMS to communicate, you still receive messages from various service providers such as telecom companies, banks and e-retailers. Companies will look at non-traditional data to assess consumer risk and determine the creditworthiness

User SMS are mined to create a dataset for loan risk prediction. Various token and amounts are extracted to create attributes for dataset such as: total amount credited, debited, total amount recharged, Emi amount paid if loan has been taken by user etc.

Loan Prediction System is based on machine learning approach in which system is trained on the basis of some training data that we have created by SMS mining over 3 lac SMS messages.

Model with best accuracy is chosen for training and testing purpose.

Selected model will be used for prediction of loan risk.

**3. Literature Survey**

* In August 2011 an important paper was published on “Credit Scoring and Loan Default” by Research Division (Federal Reserve Bank of St. Louis). This paper introduced a measure of credit score performance that abstracts from the influence of situational factors. The paper demonstrates an increasing trend of reliance on credit scoring not only as a measure of credit risk but also as a means to offset other riskier attributes of the origination. This reliance led to deterioration in loan performance even though average credit quality as measured in terms of credit scores actually improved over the years.
* Another Important paper on “New Credit Risk Models for the unbanked” by Tony Goland was published on March 2012 which listed challenges faced in Lending to lower-income households and small informal enterprises many of which have limited familiarity with formal financial services which inhibits their ability to make good decisions about the responsible and appropriate use of credit. In this new data form six sources were taken such as (Telecommunication providers, wholesale suppliers and government services).

**4. Software Requirements**

To develop his project the following tools are required to be installed on the developer’s Personal Device:

* Mongo Chef
* Web storm
* Node.js
* Mongoose libraries
* Sublime Text editor
* Python
* Scikit-learn, mathplotlib, pandas, numpy etc libraries

**5. Methodology**

Four major phases into which this project has been divided are as follows:

Phase 1: Data Collection and Preprocessing

* Collect data from various sources
* Data Sampling and Reduction

Phase 2: Attribute Extraction using Data Mining

* Make rules to identify different attributes from User's SMS.
* Processing of extracted data and resolving ambiguities.

Phase 3: Analyzing accuracy of different ML algorithms on extracted

Data set.

Phase 4: Implementing and improvising ML algorithm with best

Accuracy results.

**5.1 Data Collection and Preprocessing**

* We collected raw data in the form of SMS messages (~ 3lac) from companies which was provided for open source usage.
* The types of messages of our interest are :
* Recharge Transactions
* E – Retail orders.
* Bank Credit and Debit Transactions through various sources such as ATM, NEFT etc.
* EMI transactions if loan taken.
* But our raw dataset also had some unnecessary messages like :
* Advertisements such as

“Book 2/3 BHK Flat near HEBBAL JUNCTION starts from Rs.38Lacs.”

“111 Dhamaka offer Book & win 1 BHK flat.”

“Get Rs.50 off on all Taxi For Sure rides between 11am - 5pm in Bangalore till 13th.”

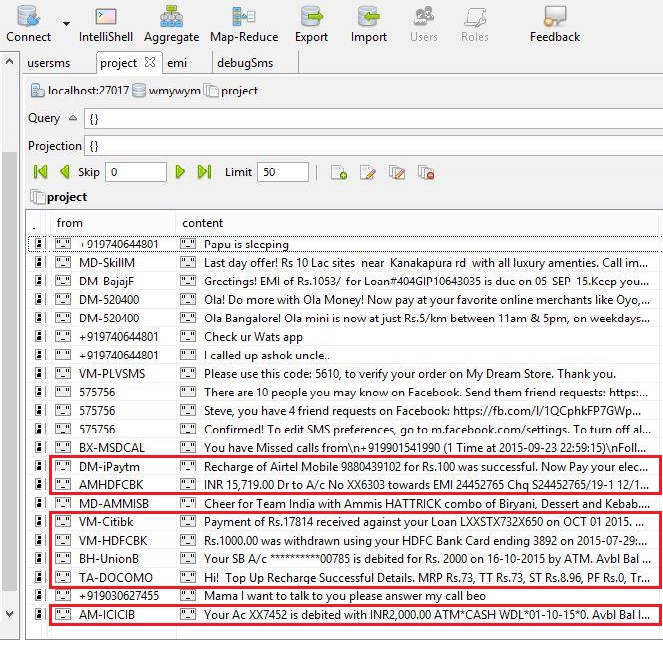
* Personal Messages such as

“Busy right now, call u later”

“Happy Anniversary Uncle and Aunty”

* Social Media messages such as

“Hi user, Facebook update! You have 2 notifications and one friend request pending.” Etc.

Screenshot of random text messages from our database.

Messages of our interest are marked **Red**.

**5.2 Attribute Extraction using Data Mining**

**5.2.1 Rules to identify different Attributes**

We are interested in some specific types of messages as mentioned earlier and the amount associated with all transactions involved in those messages.

Extraction of those transactions is done by defining rules which we have done by using “regex” (regular expressions) as follows:

* **Recharge Transaction Analysis:** We have following type of recharge messages in our database:

**FROM: AX-AIRMTA “**Recharge with Rs18 from any airtel Retailer & enjoy 150 Local SMS for 14 days.”

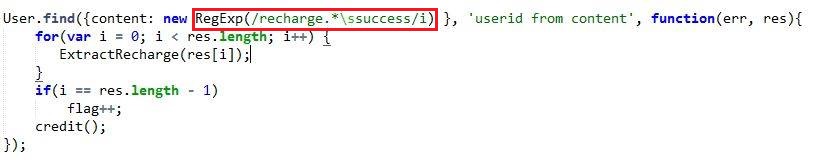
**From: DM-iPAYTM “**Recharge successful for Rs 200.00 .Talktime Credited: Rs 200.00, Processing Fee: Rs -22.00, Service Tax: Rs 22.00. Current Balance: Rs 200.01. Validity : 29-Dec-37”

**From: AM-iPAYTM “**Hi There! Recharge of Airtel Mobile 9880439102 has been submitted to AirTel. However, AirTel is taking time to process your recharge. You will either receive recharge in next 2-3 hours, or your amount will be refunded in your Paytm wallet.”

It can be observed that all the successful recharge messages have word **“Recharge”** followed by **“successful”** in common.

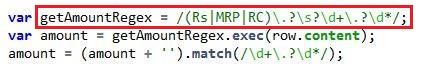
So, we used the regex : new RegExp(/recharge.\*\ssuccess/i)

case insensitive for targeting the specific messages.



And for parsing the amount being recharged we used t he

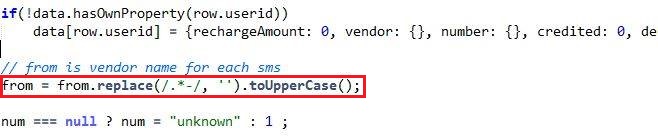
regex : /(Rs|MRP|RC)\.?\s?\d+\.?\d\*/ as shown below:



Vendor for each recharge could be found out by the “from” field of every SMS.

Ex: AM-iPAYTM, DM-iPAYTM.

Here we observed that the text after the “-” remains the same for a particular vendor while two character before the “-” can change so, we remove the text before “-” as shown below



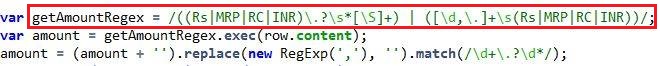
All the recharge successful sms also had the phone number that was recharged, So, we also parsed out the phone numbers for each user id.

C:\Users\Amit\Desktop\12834576_983511548394064_1876125035_n.jpg

Then we summed up the amount being recharged on each phone number and thus we got all the phone numbers with amount recharged on them for a particular user id.

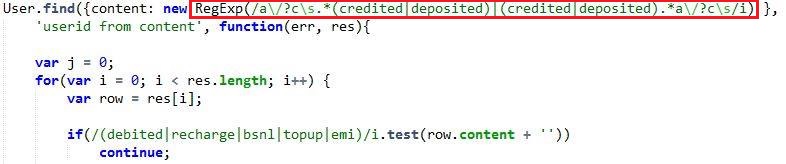
* **Debited Amount Analysis:** All the messages containing the word “debited” were first selected and then out of those messages having “recharge” or “EMI” word were ignored for processing.

Debited amount was parsed using the regex as shown in screenshot below:

****

Then for each userid we summed up the amount debited.

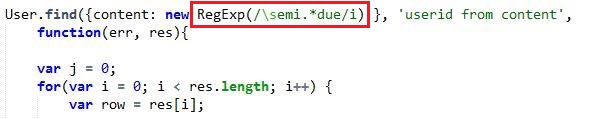
* **Credited Amount Analysis:** Amount were parsed in the same way as in debit analysis.



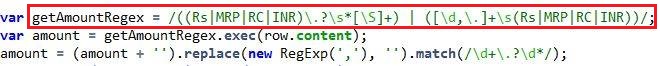
All the messages containing the word “created” were first targeted and then out of them, the messages having (debited | recharge | bsnl | topup | emi) word in their content were ignored for processing.

* **EMI Due Analysis:** The messages related to the loans taken on EMI were of form that “Your EMI of amount xxx is due on date xxxxxxx.”

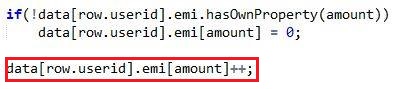
Regex used: /\semi.\*due/i to select those messages.

****

andAmount was parsed using :

****

Then using the emi amount as key I counted the number of time due message for each emi amount received



**5.2.2 Processing of extracted data and resolving ambiguities.**

Ambiguities are settled as follows:

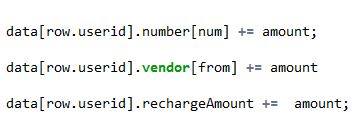
* Missing or Null values in amount field is handled by assigning zero to the null or missing field.

C:\Users\Amit\Desktop\ambi1.jpg

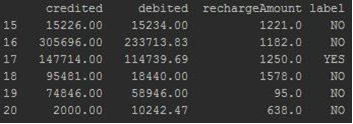
* Missing or Null values in number field are handled by assigning “unknown” to the null or missing field.

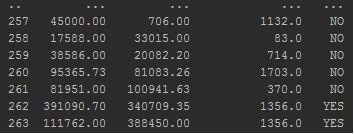
C:\Users\Amit\Desktop\ambi2.jpg

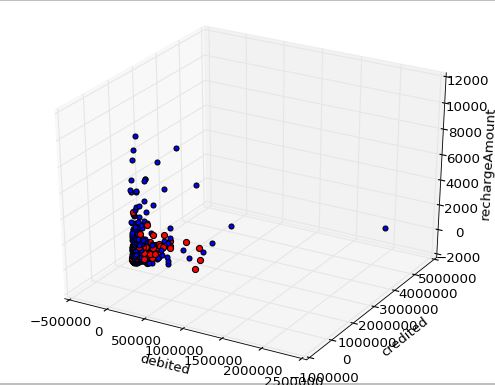
* For each user id the different recharge amount values are summed up and stored in sub categories of vendors and phone numbers.



**DATASET**





**

**5.3 Implementing various ML and Data Mining Algorithms for Classification:**

**5.3.1 Artificial Neural Network (ANN):**

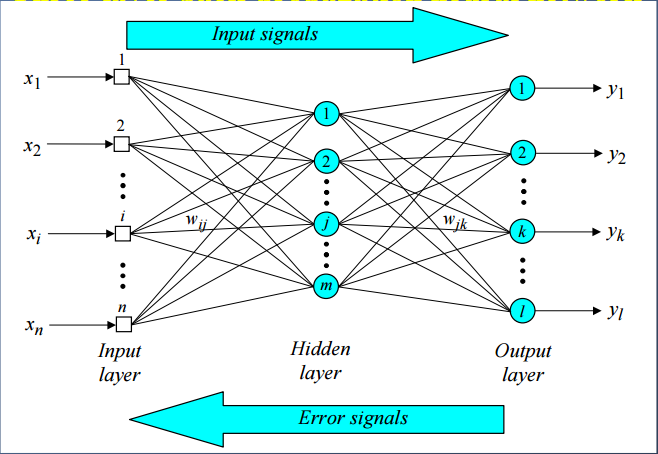
The network is first trained to achieve some required accuracy rate. Redundant connections of the network are then removed by a network pruning algorithm.

Data mining using pruned artificial neural network tree (ANNT): ANNT pruning approach consists of three phases: training, pruning and rule extraction.

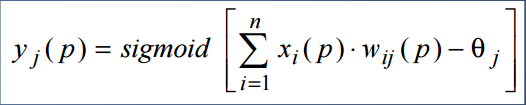
**Back Propagation Algorithm:**

* Learning in a multilayer network proceeds the same way as for a perceptron.
* A training set of input patterns is presented to the network.
* The network computes its output pattern, and if there is an error − or in other words a difference between actual and desired output patterns − the weights are adjusted to reduce this error.
* In a back in a back-propagation neural network, the learning propagation neural network, the learning algorithm has two phases.
* First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer.
* If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated.

**Three-layer back layer back-propagation neural network:**

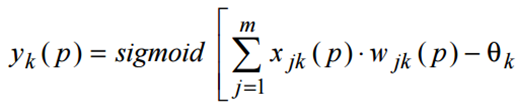


* **Step 1 (Initialization**): Set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range. The weight initialization is done on a neuron on a neuron-by-neuron basis.
* **Step 2 (Activation):** Activate the back Activate the back-propagation neural network by propagation neural network by applying inputs x1(p), x2(p),…, xn(p) and desired outputs yd,1(p), yd,2(p),…, yd,n(p). (a) Calculate the actual outputs of the neurons in the hidden layer:



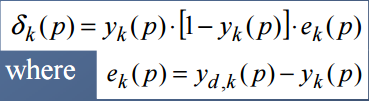
Where, n is the number of inputs of neuron j in the hidden layer, and sigmoid is the sigmoid activation function.

(b) Calculate the actual outputs of the neurons in the output layer:



where m is the number of inputs of neuron kin the output layer.

* **Step 3 (Weight training):** Update the weights in the back Update the weights in the back-propagation network propagation network propagating backward the errors associated with output neurons. (a) Calculate the error gradient for the neurons in the output layer:



Calculate the weight corrections:

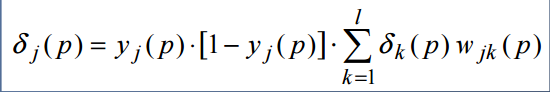


Update the weights at the output neurons:



* **Step 3: Weight training (continued**)

(b) Calculate the error gradient for the neurons in the hidden layer:



Calculate the weight corrections:



Update the weights at the hidden neurons:



* **Step 4 (Iteration):** Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.

**Dynamic Node Creation:**

One of the problems with the traditional Back propagation algorithm is the **predetermination of the number of neurons in the hidden layer** within a network. To overcome this problem the construction algorithm for feed forward networks may be used, which constructs the network during training. Thus we can have an optimal number of neurons in the hidden layer to attain a satisfactory level of efficiency for a particular problem. Besides applying the early **stopping method of training using cross-validation** we can also train the network in a relatively short estimation period (training period).

This is why we used a new algorithm for pattern classification that defines the stopping condition by the acceptance of efficiency level. Another consideration we have made that the desired or acceptable efficiency on the test sets may not be achieved even though the mean square error on training set is minimum. These considerations encouraged us to propose an algorithm that will **combine the learning rule of backpropagation algorithm to update weights of the network and the construction algorithm to construct the network dynamically** and also consider the efficiency factor as a determinant of the training process.

**Weight Freezing Based Constructive Algorithm:**

The training time is an important issue in designing ANNs. One approach for reducing the number of weights to be trained is to train few weights rather than all weights in a network and keep remaining weights fixed, commonly known as weight freezing. The idea behind the weight freezing-based constructive algorithm is to freeze input weights of a hidden node when its output does not change much in the successive few training epochs.

Theoretical and experimental studies reveal that some hidden nodes of an ANN maintain almost constant output after some training epochs, while others continuously change during the whole training period.

In our algorithm, it has been proposed that the output of a hidden node can be frozen when its output does not change much in the successive training epochs. This weight freezing method can be considered as combination of the two extremes: for training all the weights of ANNs and for training the weights of only the newly added hidden node of ANNs. The major steps of our weight freezing based constructive algorithm are explained further as follows:

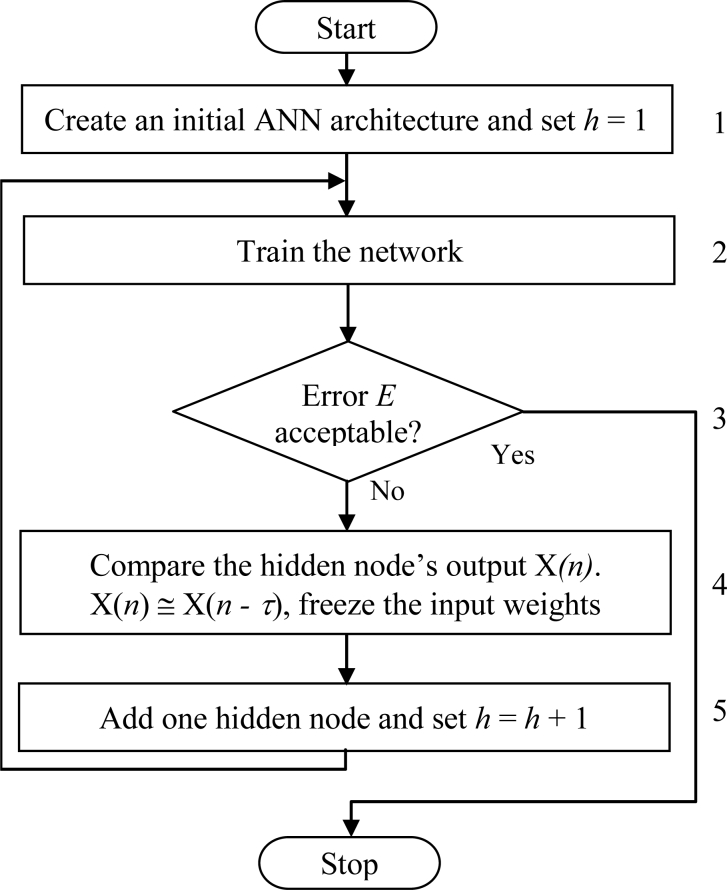
**Step 1** Create an initial ANN consisting of three layers, i.e., an input, an output, and a hidden layer. The number of nodes in the input and output layers is the same as the number of inputs and outputs of the problem. Initially, the hidden layer contains only one node i.e., h = 1, where h is the number of hidden nodes in the network. Randomly initialize all connection weights within a certain range.

**Step 2** Train the network on the training set by using backpropagation algorithm until the error E is almost constant for a certain number of training epochs, τ, is specified by the user.

**Step 3** Compute the ANN error E. If E is found unacceptable (i.e., too large), then assume that the ANN has inappropriate architecture, and go to the next step. Otherwise stop the training process.

**Step 4** Compare each hidden node’s output X (n) at training epoch n with its previous value X (n − τ). If X (n) ≅ X (n − τ), freeze the input weights of that node.

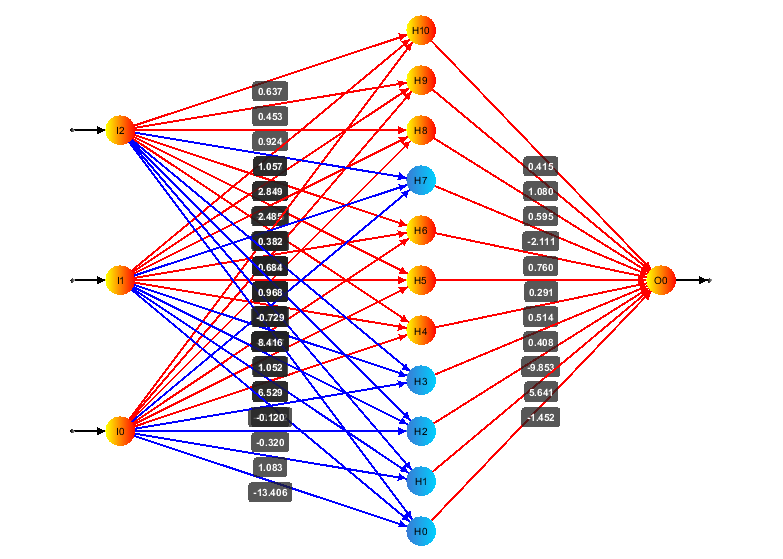
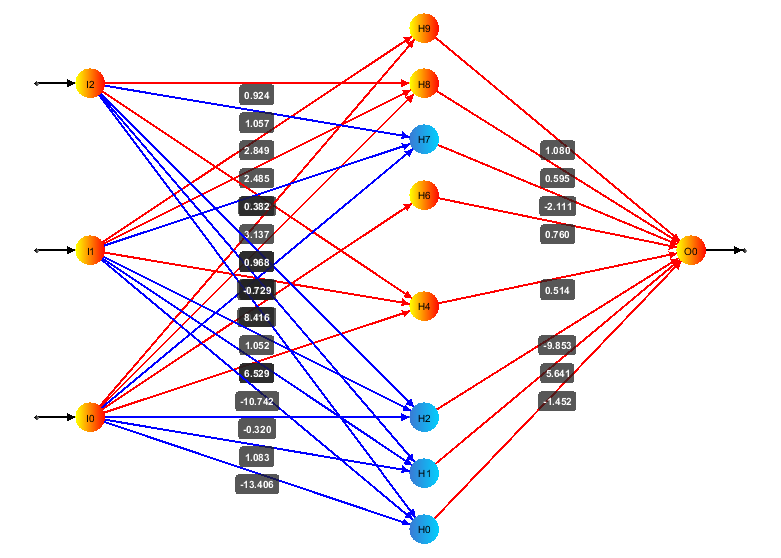
**Step 5** Add one hidden node to the hidden layer. Randomly initialize the weights of the arcs connecting this new hidden node with input nodes and output nodes. Set h = h + 1 and go to **Step 2**.

****

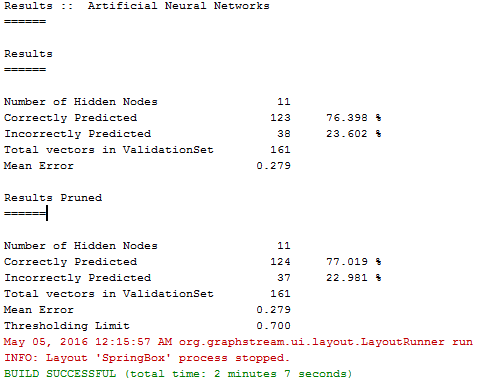
**Network Pruning:**

* Pruning offers an approach for dynamically determining an appropriate network topology. The technique begin by training a larger than necessary network and then eliminate weights and nodes that are deemed redundant.
* If classification accuracy of the network falls below an acceptable level, then stop and use the previous setting of the network weights.
* Remove all weights below a threshold

**Neural Network after Completion of Classification:**

**Results of ANN Classification:**



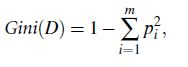
**5.4 DECISION TREE**

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.

Decision trees can handle high dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. In general, decision tree classifiers have good accuracy. This procedure employs an attribute selection measure, such as information gain or the gini index.

* Information gain
* Gini index.
* Gain ratio

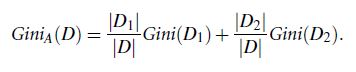
We have used gini index method for decision tree attribute selection.



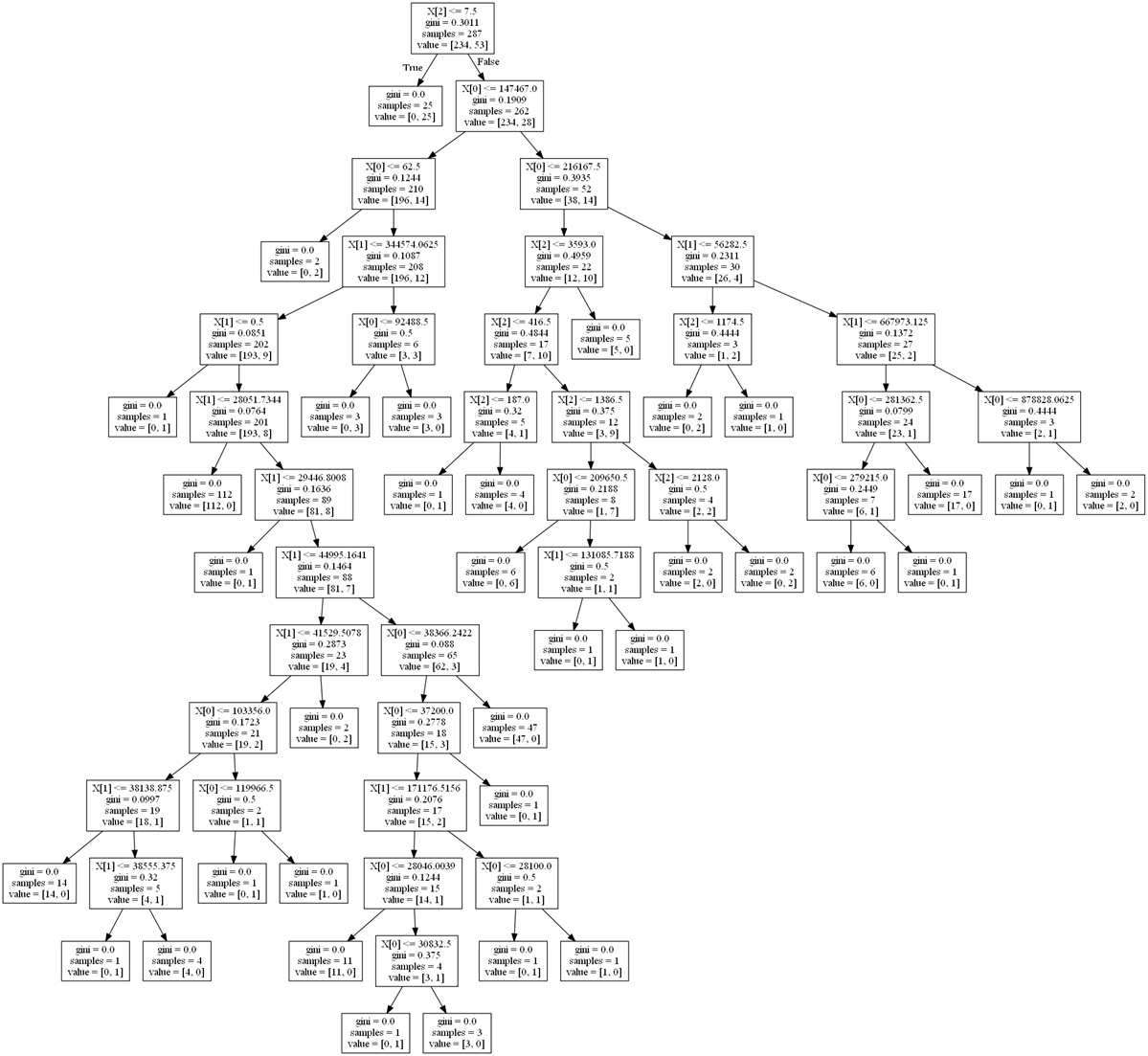
*pi* is the probability that a tuple in *D* belongs to class *Ci.* The sum is computed over *m* classes. *D*.

To determine the best binary split on *A*, we examine all of the possible subsets that can be formed using known values of *A*. Given a tuple, this test is satisfied if the value of *A* for the tuple is among the values listed in SA.

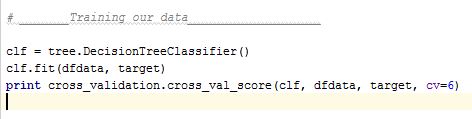
We compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on *A* partitions *D* into *D*1 and *D*2, the gini index of *D* given that partitioning is.



**DECISION TREE ON OUR DATASET.**



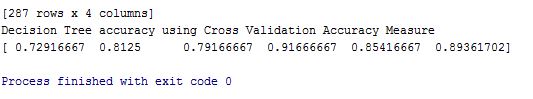
Code Snippet



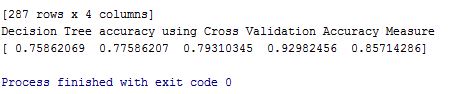
**Cross Validation Accuracy:**

We have take k = 6 here, i.e. we are diving our data randomly into random chunks of 6 partition and cross validating output by taking five partition as Training data and 1 partition as testing data.

K = 6 Average percentage = 82.26 %



K = 5 Average percentage = 81.35 %



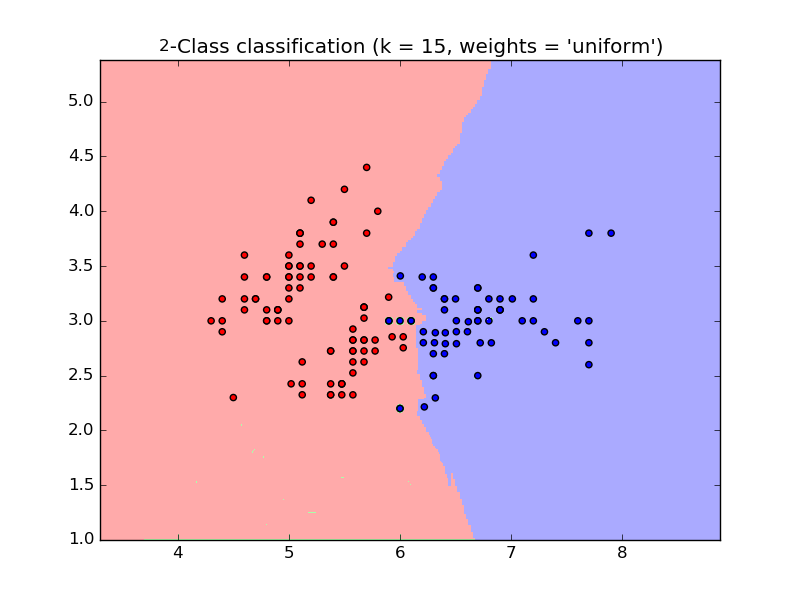
**5.5 K-Nearest Neighbors(K-NN)**

* *k*-NN is a type of instance based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.
* Classification is computed from a simple majority vote of the nearest neighbors of each point
* A query point is assigned the data class which has the most representatives within the nearest neighbors of the point.
* We used KNeighborsClassifier which implements learning based on the k nearest neighbors of each query point , where k is an integer value specified by the user.

.

* The k-neighbors classification in KNeighborsClassifier is the more commonly used of the two techniques.
* The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.
* The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors.
* This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns uniform weights to each neighbor.
* weights = 'distance' assigns weights proportional to the inverse of the distance from the query point.

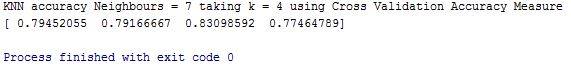
**2 CLASS CLASSIFICATION BY K-NN:**



Above is the KNN Classification of two classes with K = 15 and weights = ‘uniform

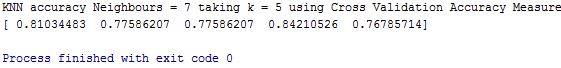
**Accuracy results of KNN with different values of k**:

**For k = 4:**



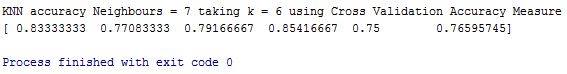
**Average accuracy: 79.5 %**

**For k = 5:**



**Average accuracy:79 %**

**For k = 6:**



**Average accuracy:79.16 %**

**5.6 SVM(Support Vector Machine):**

* Support vector machines are supervised learning with associated learning models with associated learning algorithms that analyze data used for classification and regression analysis.
* It uses a non linear mapping to transform the original training data into a higher dimension.
* With that non linear mapping , data from 2 classes can always be separated by hyperplane.
* SVM finds this hyperplane using support vectors.

****

****

**RBF SVM(Radial Basis Function SVM):**

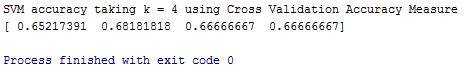
* SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
* For non linear classification, RBF SVM is quite adaptive.
* It uses RBF Kernel , which on two samples x and x’, represented as feature vectors in some input space is defined as:



* where gamma is a parameter 

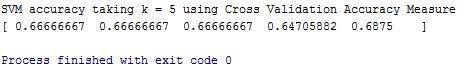
**RBF SVM Accuracy with different values of k:**

**K = 4:**



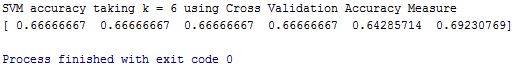
**Average accuracy: 66.25 %**

**K = 5:**



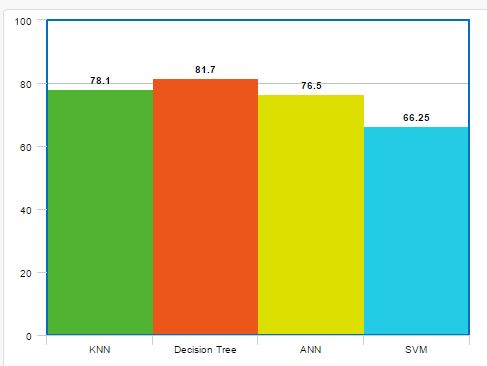
**Average accuracy : 66 %**

**K = 6:**



**Average accuracy: 66.16 %**

**6. COMPARATIVE RESULT**



**6. Activity Time Chart**

* Before Mid Semester
* Data Collection and Preprocessing
* Attribute Extraction and resolving ambiguities.
* After Mid Semester
* Analyzing accuracies of various Machine Learning Algorithms on Extracted data set.
* Implementing and improvising ML algorithm with best accuracy results.

**7. Conclusions**

* We have successfully gathered Real Time user SMSes from various sources and preprocessed this raw data.
* We have successfully extracted Key components 3Lac user SMSes and processed these attributes to create a useful data set.
* We used this data set as our training data to train different Machine Learning Algorithms and will analyze their accuracies.
* Comparative study of accuracies of different machine learning algorithms.

**8. References**

1. <https://www.bostonfed.org/economic/wp/wp2015/wp1502.pdf>
2. http://www.ijarcsms.com/docs/paper/volume3/issue3/V3I3-0006.pdf
3. <http://epub.lib.aalto.fi/en/ethesis/pdf/12299/hse_ethesis_12299.pdf>
4. <http://scikit-learn.org/stable/modules/svm.html>
5. <http://scikit-learn.org/stable/>
6. http://scikit-learn.org/stable/modules/svm.html

(Last accessed on 5/5/2016)