ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
GAZIPUR, BANGLADESH

Department of Computer Science and Engineering

THESIS THEME

Concept Drift Detection on financial data Using Naïve Bayes Classifier

By

Hassan Amadu	154450
Ousmanou Mamoudou Bouba	154451

Mr. Ashikur Rahman	Supervisor
Mr. Hamjajul Ashmafee	Co-supervisor

18 November, 2019

Table of Contents

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Hassan Amadu** and **Ousmanou Mamoudou Bouba** under the supervision of Mr. **Mr. Ashikur Rahman** and **Mr. Hamjajul Ashmafee,** lecturers at the Islamic University of Technology (IUT), Dhaka, Bangladesh. It also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Hassan Amadu

ID: 154450

Ousmanou Mamoudou Bouba

ID: 154451

Mr. Ashikur Rahman Supervisor

Mr. Hamjajul Ashmafee   Co-Supervisor

Abstract

Concept drift detection has been a very active field of research in the domain of finance due to streaming of data which sometimes caused a significant lost to the organizations. As a result, we collected dataset for an organization and applied different machine learning algorithm to analyze the best reliable and computational comfort.

Introduction

Overview

Concept drift detection has been a very active field of research since the 90s in the intertwined domains of machine learning and data mining. Data mining streams are used frequently in domains such as sensor networks, network analysis, real time surveillance systems and financial transactions. There is very high intensity of data generated in these systems and the data might change over time. This is referred to as concept drift problem and it is considered to be the root cause of performance degradation of online machine learning models. To tackle this problem, a reliable and fast drift detection method is required to achieve real time responsiveness to the drifts. These data streams consist of millions or billions of updates and must be processed to extract the useful information. Because of the high speed and huge size of data set in data streams, the traditional classification technologies are no longer applicable.

In recent years a great deal of research has been done on this problem, most intends to efficiently solve the data streams mining problem with concept drift. In the following paper we focus on detecting concept drift in financial data using a variation of the Naïve Bayes classifier.

What is Concept drift?

Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes.

> The term concept refers to the quantity to be predicted. More generally, it can also refer to other phenomena of interest besides the target concept, such as an input, but, in the context of concept drift, the term commonly refers to the target variable.

Benefits of Concept Drift Detection

The following are the benefits of Concept drift detection

- It helps prevent deterioration in prediction accuracy.
- It also enables data scientist to realize the change in the statistics of the data gathering process.
- In stationary conditions, any fresh information made available can be integrated to improve the model.
- It equally enables us determine the dependency between the various variables.

Problem Statement

The main focus here is on financial data which consist of credit, expenses and income. The issue of concern is, to find how income varies with respect to credit and expenses. The variation for a few consecutive months is observed and analyzed. This is done by the Naïve Bayes Algorithm method which is a method of classification. The data set used in this paper is from an investment company and the goal is to determine whether at the end of the month the customer is satisfied or unsatisfied with her income.

Thesis Objectives

- To obtain the relationship between income and (expense, credit).
- To visualize the variation of income over the various months
- To establish a method of predicting income in the future
- To ensure credit and expense scrutiny to maintain a viable income.
- To manage the drift over years and even for different organizations.

Related Works

Financial Data Analysis with PGMs using AMIDST

Previously a proposed demonstration of PGMs framework with a particular focus on concept drift for performing real-world financial data set from a Spanish bank was conducted. Because PGMs framework was implemented in AMIDST toolbox, it provides an overview of AMIDST toolbox and illustration set up to perform this particular analysis. [1]

- PGMs was composed on a principled modelling framework for learning and reasoning under uncertainty [2].
- PGMs defined two parts; qualitative and quantitative components.
- Qualitative component independence relationship between the variables in the domain already modelled.
- Quantitative component consists of a set of probability distributions quantifying the relations specified in the graph [3].
- Updating frequency can produce huge volumes of data, because of the streaming nature of data.

- PGMs only consider the most recently generated data or store the data at a much lower frequency than with which it is generated.

The main procedure used in the paper was as follows:

In particular Hybrid Bayesian Network was considered, in which the qualitative part is a direct acyclic graph (DAG) and the quantitative part is a set of conditional probability distributions and dynamic Bayesian networks (DBNs) in the form of 2-Timeslice.

The main features of the AMIDST Toolbox are as follows:

- Probabilistic graphical models
- Multi-core and distributed processing
- Concept drift detection
- Interoperability

The advantages of this procedure are as follows:

AMIDST Toolbox is the first system to directly support scalable mining and analysis of data streams based on PGMs.

- AMIDST Toolbox has the possibility of handling large data streams
- The AMIDST software has been built around the Maven3 automation tool for managing software projects.
- Application of AMIDST to concept drift detections with code examples

The main criticism is that it does not support some distributed network processing frameworks like apache spark.

Large-Scale Data-Driven Financial Risk Modelling using Big Data Technology

Analyzing real-world data is challenging due to heterogeneous data across difference firms and legal entities, current financial risk algorithms are typically inconsistent and non-scalable. So, the paper focused on how to implement a use case scalable algorithm that can perform large-scale financial risk analysis maximizing big data technology, and the performance demonstration was linear scalable. [4] Basel Committee on Banking Supervision (BCBS) identified the major shortcoming (of Lehman Brothers) to be the banks' poor capability to quickly and accurately aggregate risk exposures and identify concentrations of risk at the bank group level, across business lines and between legal entities

To provide efficiency, reliability and transparency for performing large-scale risk analysis data for contract and cash flow generating algorithms must be well concise and defined.

The data input for ACTUS framework needs two steps:

- Contract data defined in the legal agreements
- Risk factors scenarios determine the state of the financial and economic environment under which the cash flows of a contract should be evaluated.

The benefits of this procedure are as follows:

- All experiments were executed on Amazon Web Services running Spark 2.3 using the Java interface of Spark and ACTUS.

- Up to 32 instances were used, each with 30 GB RAM and 16 vCPU cores running at 2.5 GHz [5].
- The test case contains up to 96 million *financial contracts*
- Scalability was studied by increasing the number of CPU cores

Some drawbacks of this model are as follows.

- Cannot work above 96millions financial contracts on 512 vCPUs cores.
- Performs linear scalability.

  Another paper presents a fast and accurate drift detection method, namely *KS-SVD test — KSSVD*, to monitor the distribution changes of the data stream. The method employs the SVD technique to first check the direction changes of the data, followed by a KS test on each direction to detect the univariate distribution changes. [6]

  Another paper proposes a novel method of corporate financial risk prediction (FRP) modeling called the adaptive and dynamic ensemble (ADE) of support vector machine (SVM). (ADE-SVM), which integrates the inflow of new data batches for FRP with the process of time. Namely, the characteristic change of corporate financial distress hidden in the data flow is considered as the concept drift of financial distress, and it is handled by ADE-SVM that keeps updating in time. Using the criteria of predictive ability and classifier diversity, the SVM ensemble is dynamically constructed by adaptively selecting the current base SVMs from candidate ones [7]. The candidate SVMs are incrementally updated by considering the newest data batch at each new current time point. The results of the base SVMs are dynamically weighted by their validation accuracies on the latest data batch to generate the final prediction.

Background Knowledge

Mathematical background

By definition of a concept,
Concept = P(x, y)
In the context of data streams, we need to recognize that concepts may change over time.

To this end we define the concept at a particular time t as,
$p_t\ (x, y)$

And at a specific time period [t, u] as $p_{[t,u]}\ (x, y)$

Concept drift occurs between times t and u when the distributions change.
p_t (x, y) != p_u(x, y)

We define concept drift mapping as taking as input a data stream and generating as output useful descriptions of the drift in the process that generates the data.
Now measuring total drift magnitude using Total variation distance:
$\sigma_{t,u}(z) = 1/2\sum_{z\in dom(z)}[p_t(z) - p_u(z)]$

Now measuring marginal drift magnitude

$\sigma_{t,u}(y) = 1/2\sum_{y\in Y}[p_t(y) - p_u(y)]$

Posterior drift

$\sigma_{t,u}(x) = 1/2\sum_{x\in X}[p_t(x) - p_u(x)]$

Algorithms

The first algorithm used in this thesis is the Naïve Bayes Classifier.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Naive Bayes is a probabilistic technique for constructing classifiers. The characteristic assumption of the naive Bayes classifier is to consider that the value of a particular feature is independent of the value of any other feature, given the class variable.

Despite the oversimplified assumptions mentioned previously, naive Bayes classifiers have good results in complex real-world situations. An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification and that the classifier can be trained incrementally.

Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector **x** = (x_1,x_n) representing some n features (independent variables), it assigns to this instance probabilities for each of K possible outcomes or classes.

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it simpler. Using Bayes theorem, the conditional probability can be decomposed as −

p(Ck|x)=p(Ck)p(x|Ck)p(x)p(Ck|x)=p(Ck)p(x|Ck)p(x)

This means that under the above independence assumptions, the conditional distribution over the class variable C is −

p(Ck|x1,.....,xn)=1Zp(Ck)∏i=1np(xi|Ck)p(Ck|x1,.....,xn)=1Zp(Ck)∏i=1np(xi|Ck)

where the evidence Z = p(**x**) is a scaling factor dependent only on x_1, …, x_n, that is a constant if the values of the feature variables are known. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule.

The pseudocode for the Naïve Bayes algorithm is as follows:

```
Input:
        Training dataset T,
        F = (f1, f2, f3, … fn) value of the predictor variable in
        Testing dataset.
Output.
        A class of testing dataset.
Step
    1.  Read the testing dataset
    2.  Calculate the mean and standard deviation of each
        Variable in each class
    3.  Repeat
            Calculate the probability of fi using the gauss density function in
            each class;

    Until the probability of all predictor
    Variables (f1,f2,f3…fn) has been calculated
```

4. Calculate the likelihood for each class.
5. Calculate the greatest likelihood

Theory on Naïve Bayes

The Naïve Bayes classifier models produces probability estimates rather than hard classifications. For each class value, they estimate the probability that a given instance belongs to that class. Most other types of classifiers can be coerced into yielding this kind of information if necessary. For example, probabilities can be obtained from a decision by computing the relative frequency of each class in a leaf and from a decision list by examining the instances that a particular rule cover.

Probability estimates are often more useful than plain predictions. They allow predictions to be ranked and their expected cost to be minimized

In fact, there is a strong argument for treating classification learning as the task of learning class probability estimates from data. What is being estimated is the conditional probability distribution of the values of the class attribute given the values of the other attributes. Ideally, the classification model represents this conditional distribution in a concise and easily comprehensible form.

Viewed in this way, Naïve Bayes classifiers, logistic regression models, decision trees, and so on, are just alternative ways of representing a conditional probability distribution. Of course, they differ in representational power. Naïve Bayes classifiers and logistic regression models can only represent simple distributions, whereas decision trees can represent—or at least approximate—arbitrary distributions. However, decision trees have their drawbacks: They fragment the training set into smaller and smaller pieces, which inevitably yields less reliable probability estimates, and they suffer from the replicated subtree problem. Rule sets go some way toward addressing these shortcomings, but the design of a good rule learner is guided by heuristics with scant theoretical justification.

Does this mean that we have to accept our fate and live with these shortcomings?

No! There is a statistically based alternative: a theoretically well-founded way of representing probability distributions concisely and comprehensibly in a graphical manner; the structures are called Bayesian networks. They are drawn as a network of nodes, one for each attribute, connected by directed edges in such a way that there are no cycles—a directed acyclic graph. In our explanation of how to interpret Bayesian networks and how to learn them from data, we will make some simplifying assumptions. We assume that all attributes are nominal and that there are no missing values. Some advanced learning algorithms can create new attributes in addition to the ones present in the data—so-called hidden attributes with values that cannot be observed. These can support better models if they represent salient features of the underlying problem, and Bayesian networks provide a good way of using them at prediction time. However, they make both learning and prediction far more complex and time consuming, so we will not consider them here.

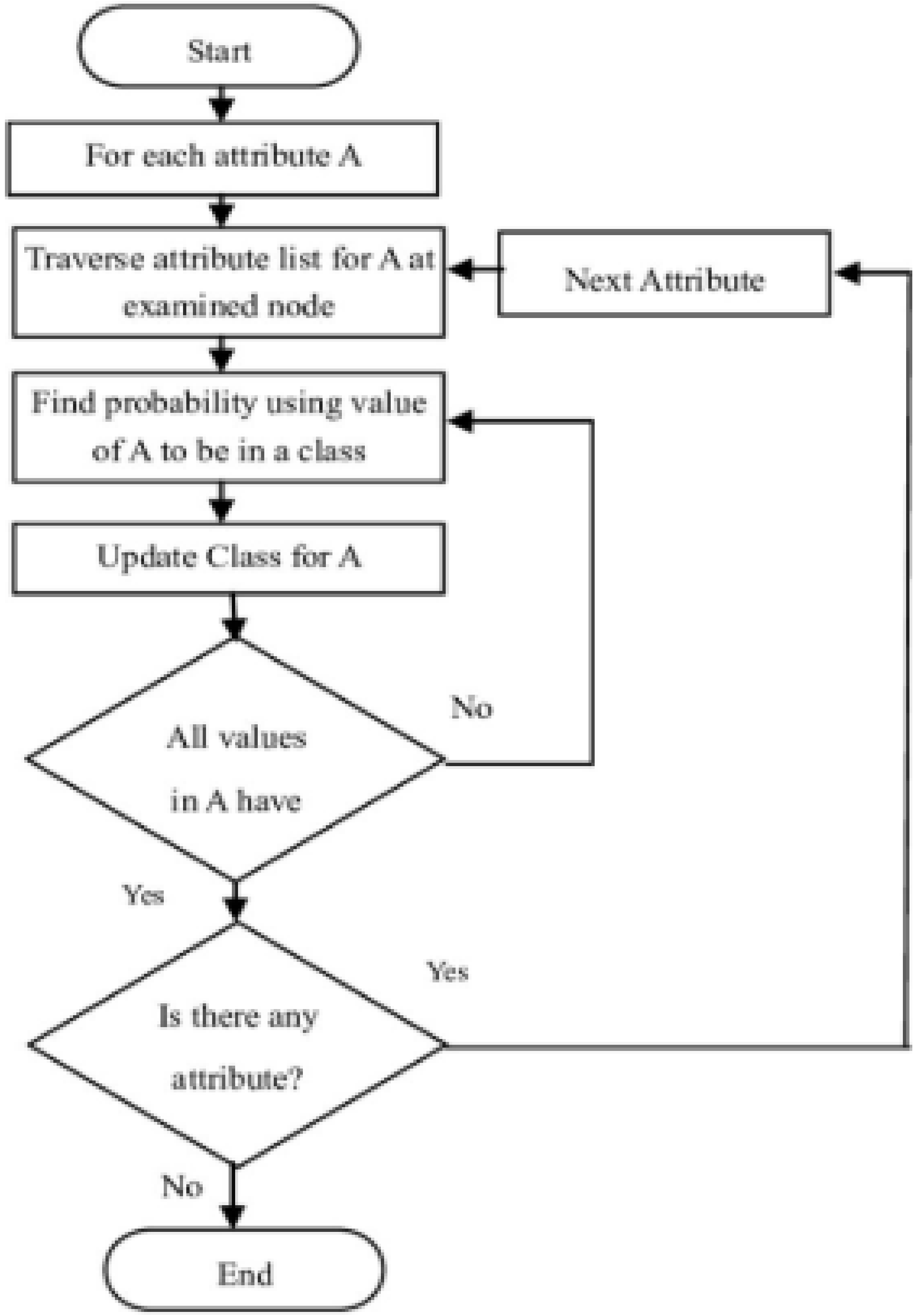


Figure 1: Flowchart for Naïve Bayes Algorithm

Materials

The following materials were used in the thesis experiment.

Weka

Waikato Environment for Knowledge Analysis (**Weka**), developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License, and the companion software to the book "Data Mining: Practical Machine Learning Tools and Techniques". Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and a Makefile-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:

- Free availability under the GNU General Public License.
- Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.
- A comprehensive collection of data preprocessing and modeling techniques.
- Ease of use due to its graphical user interfaces.

Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as one flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query. Weka provides access to deep learning with Deeplearning4j· It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka. Another important area that is currently not covered by the algorithms included in the Weka distribution is sequence modeling.

R

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity; as of November 2019, R ranks 16th in the TIOBE index, a measure of popularity of programming languages.

A GNU package, source code for the R software environment is written primarily in C, Fortran, and R itself and is freely available under the GNU General Public License. Pre-compiled binary versions are provided for various operating systems. Although R has a command line interface, there are several graphical user interfaces, such as RStudio, an integrated development environment.

MATLAB

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

As of 2018, MATLAB has more than 3 million users worldwide. MATLAB users come from various backgrounds of engineering, science, and economics.

AMIDST Toolbox

AMIDST is an open source Java toolbox for scalable probabilistic machine learning with a special focus on (massive) streaming data. The toolbox allows specifying probabilistic graphical models with latent variables and temporal dependencies.

The main features of the toolbox are listed below:

Probabilistic Graphical Models: Specify your model using probabilistic graphical models with latent variables and temporal dependencies. AMIDST contains a large list of predefined latent variable models:

Scalable inference: Perform inference on your probabilistic models with powerful approximate and scalable algorithms.

Data Streams: Update your models when new data is available. This makes our toolbox appropriate for learning from (massive) data streams.

Large-scale Data: Use your defined models to process massive data sets in a distributed computer cluster using Apache Flink or (soon) Apache Spark.

Extensible: Code your models or algorithms within AMIDST and expand the toolbox functionalities. Flexible toolbox for researchers performing their experimentation in machine learning.

Interoperability: Leverage existing functionalities and algorithms by interfacing to other software tools such as Hugin, MOA, Weka, R, [8] etc.

Multi-Core Scalability using Java 8 Streams

Scalability is a main concern for the AMIDST toolbox. Java 8 streams are used to provide parallel implementations of our learning algorithms. If more computation capacity is needed to process data, AMIDST users can also use more CPU cores. As an example, the following figure shows how the data processing capacity of our toolbox increases given the number of CPU cores when learning and a probabilistic model (including a class variable C, two latent variables (LM, LG), multinomial (M1,…,M50) and Gaussian (G1,…,G50) observable variables) using the AMIDST's learning engine. As can be seen, using our variation learning engine, AMIDST toolbox is able to process data in the order of gigabytes (GB) per hour depending on the number of available CPU cores with large and complex PGMs with latent variables. Note that, these experiments were carried out on an Ubuntu Linux server with a x86_64 architecture and 32 cores. The size of the processed data set was measured according to the Weka's ARFF format.

Results in a multicore CPU
Distributed Scalability using Apache Flink

If your data is really big and cannot be stored in a single laptop, you can also learn your probabilistic model on it by using the AMIDST distributed learning engine based on a novel and state-of-the-art distributed message passing scheme implemented on top of Apache Flink. We were able to perform inference in a billion node (i.e. 109) probabilistic model in an Amazon's cluster with 2, 4, 8 and 16 nodes, each node containing 8 processing units. The following figure shows the scalability of our approach under these settings.

Procedures

Preprocessing

Data preprocessing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

Often, data preprocessing is the most important phase of a machine learning project, especially in computational biology.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data preprocessing includes cleaning, Instance selection, normalization, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training set.

Data pre-processing may affect the way in which outcomes of the final data processing can be interpreted. This aspect should be carefully considered when interpretation of the results is a key point, such in the multivariate processing of chemical data.

Data Cleansing

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

The actual process of data cleansing may involve removing typographical errors or validating and correcting values against a known list of entities. The validation may be strict (such as rejecting any address that does not have a valid postal code) or fuzzy (such as correcting records that partially match existing, known records). Some data cleansing solutions will clean data by cross checking with a validated data set. A common data cleansing practice is data enhancement, where data is made more complete by adding related information. For example, appending addresses with any phone numbers related to that address. Data cleansing may also involve harmonization (or normalization) of data, which is the process of bringing together data of "varying file formats, naming conventions, and columns", and transforming it into one cohesive data set; a simple example is the expansion of abbreviations.

Data editing

Data editing is defined as the process involving the review and adjustment of collected survey data. The purpose is to control the quality of the collected data. Data editing can be performed manually, with the assistance of a computer or a combination of both.

Interactive editing

The term interactive editing is commonly used for modern computer-assisted manual editing. Most interactive data editing tools applied at National Statistical Institutes (NSIs) allow one to check the specified edits during or after data entry, and if necessary, to correct erroneous data immediately. Several approaches can be followed to correct erroneous data:

- Re-contact the respondent
- Compare the respondent's data to his data from previous year
- Compare the respondent's data to data from similar respondents
- Use the subject matter knowledge of the human editor

Interactive editing is a standard way to edit data. It can be used to edit both categorical and continuous data. Interactive editing reduces the time frame needed to complete the cyclical process of review and adjustment.

Selective editing is an umbrella term for several methods to identify the influential errors, and outliers. Selective editing techniques aim to apply interactive editing to a well-chosen subset of the records, such that the limited time and resources available for interactive editing are allocated to those records where it has the most effect on the quality of the final estimates of publication figures. In selective editing, data is split into two streams:

- The critical stream
- The non-critical stream

The critical stream consists of records that are more likely to contain influential errors. These critical records are edited in a traditional interactive manner. The records in the non-critical stream which are unlikely to contain influential errors are not edited in a computer assisted manner.

Macro editing
There are two methods of macro editing:
Aggregation method

This method is followed in almost every statistical agency before publication: verifying whether figures to be published seem plausible. This is accomplished by comparing quantities in publication tables with same quantities in previous publications. If an unusual value is observed, a micro-editing procedure is applied to the individual records and fields contributing to the suspicious quantity.

Distribution method

Data available is used to characterize the distribution of the variables. Then all individual values are compared with the distribution. Records containing values that could be considered uncommon (given the distribution) are candidates for further inspection and possibly for editing.

Automatic editing

In automatic editing records are edited by a computer without human intervention. Prior knowledge on the values of a single variable or a combination of variables can be formulated as a set of edit rules which specify or constrain the admissible values.

Data Reduction

Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. The basic concept is the reduction of multitudinous amounts of data down to the meaningful parts.

When information is derived from instrument readings there may also be a transformation from analog to digital form. When the data are already in digital form the 'reduction' of the data typically involves some editing, scaling, encoding, sorting, collating, and producing tabular summaries. When the observations are discrete but the underlying phenomenon is continuous then smoothing and interpolation are often needed. Often the data reduction is undertaken in the presence of reading or measurement errors. Some idea of the nature of these errors is needed before the most likely value may be determined.

An example in astronomy is the data reduction in the Kepler satellite. This satellite records 95-megapixel images once every six seconds, generating tens of megabytes of data per second, which is orders of magnitudes more than the downlink bandwidth of 550 KBPS. The on-board data reduction encompasses co-adding the raw frames for thirty minutes, reducing the bandwidth by a factor of 300. Furthermore, interesting targets are pre-selected and only the relevant pixels are processed, which is 6% of the total. This reduced data is then sent to Earth where it is processed further.

Research has also been carried out on the use of data reduction in wearable (wireless) devices for health monitoring and diagnosis applications. For example, in the context of epilepsy diagnosis, data reduction has been used to increase the battery lifetime of a wearable EEG device by selecting, and only transmitting, EEG data that is relevant for diagnosis and discarding background activity.

Data Transformation

Data transformation is the process of converting data from one format or structure into another format or structure. It is a fundamental aspect of most data integration and data management tasks such as data wrangling, data warehousing, data integration and application integration.

Data transformation can be simple or complex based on the required changes to the data between the source (initial) data and the target (final) data. Data transformation is typically performed via a mixture of manual and automated steps. Tools and technologies used for data transformation can vary widely based on the format, structure, complexity, and volume of the data being transformed.

A master data recast is another form of data transformation where the entire database of data values is transformed or recast without extracting the data from the database. All data in a well-designed database is directly or indirectly related to a limited set of master database tables by a network of foreign key constraints. Each foreign key constraint is dependent upon a unique database index from the parent database table. Therefore, when the proper master database table is recast with a different unique index, the directly and indirectly related data are also recast or restated. The directly and indirectly related data may also still be viewed in the original form since the original unique index still exists with the master data. Also, the database recast must be done in such a way as to not impact the applications architecture software.

Experimental work

In this paper, we have used WEKA (Waikato environment for knowledge analysis) tool for computation of Naive Bayes calculating efficiency based on accuracy regarding correct and incorrect instances generated with confusion matrix. We have used bank data from Chambre de Commerce. The dataset consists of 3 months each having 1000 instances. The attributes consist of income, expense, credit and satisfaction.

The flow chart of the experiment is as follows:

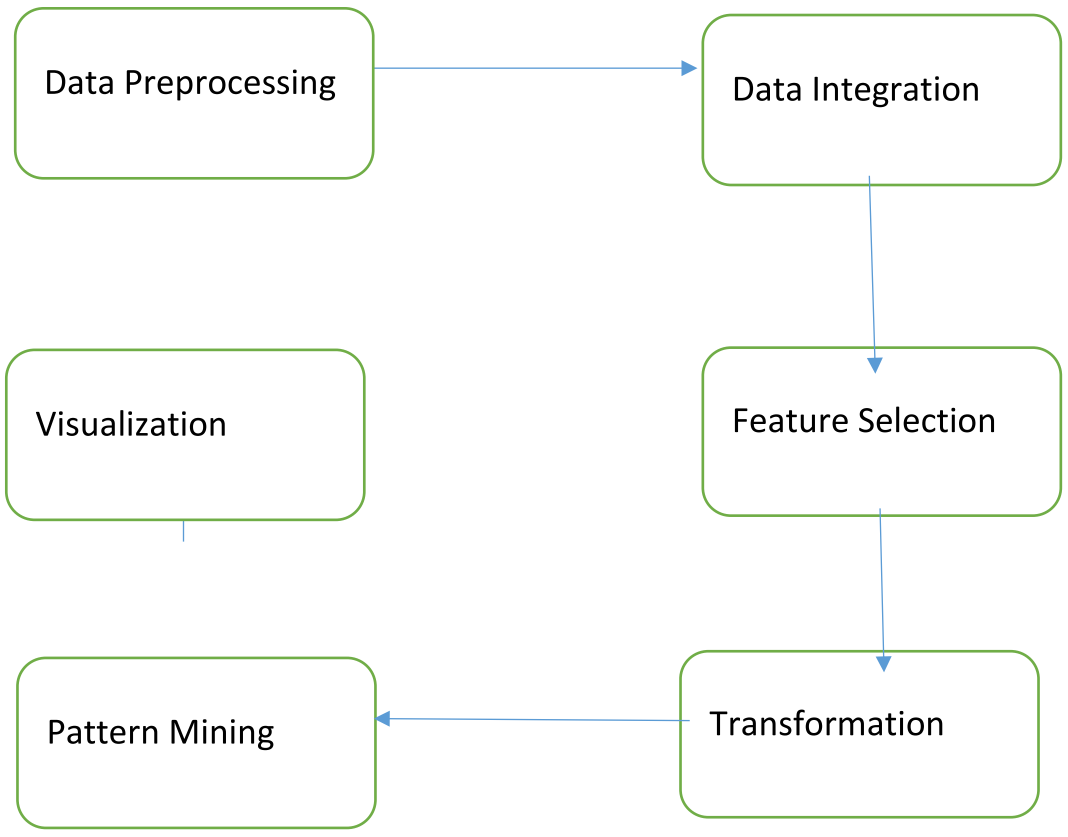


Figure 2: Flow Chart of experiment

Experimental Results

The data for three months is observed and the out is as shown below.

The various weighted sums and standard deviations are as shown below for each month.

Key points to note include:

Confusion Matrix

A confusion matrix illustrates the accuracy of the solution to a classification problem. Given n classes a confusion matrix is a m x n matrix, where Ci, j indicates the number of tuples from D that were assign to class Ci,j but where the correct class is Ci.

Obviously, the best solution will have only zero values outside the diagonal.

A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

The entries in the confusion matrix have the following meaning in the context of our study:

- a is the number of correct predictions that an instance is negative,
- b is the number of incorrect predictions that an instance is positive,
- c is the number of incorrect of predictions that an instance negative
- d is the number of correct predictions that

True positive rate = diagonal element/ sum of relevant row

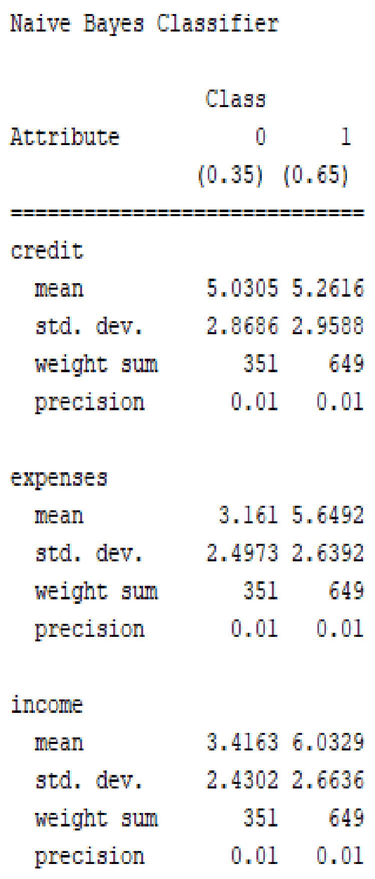
False positive rate = non-diagonal element/ sum of relevant row

Some standards and terms:

1. True positive (TP): If the outcome from a prediction is p and the actual value is also p, then it is called a true positive.

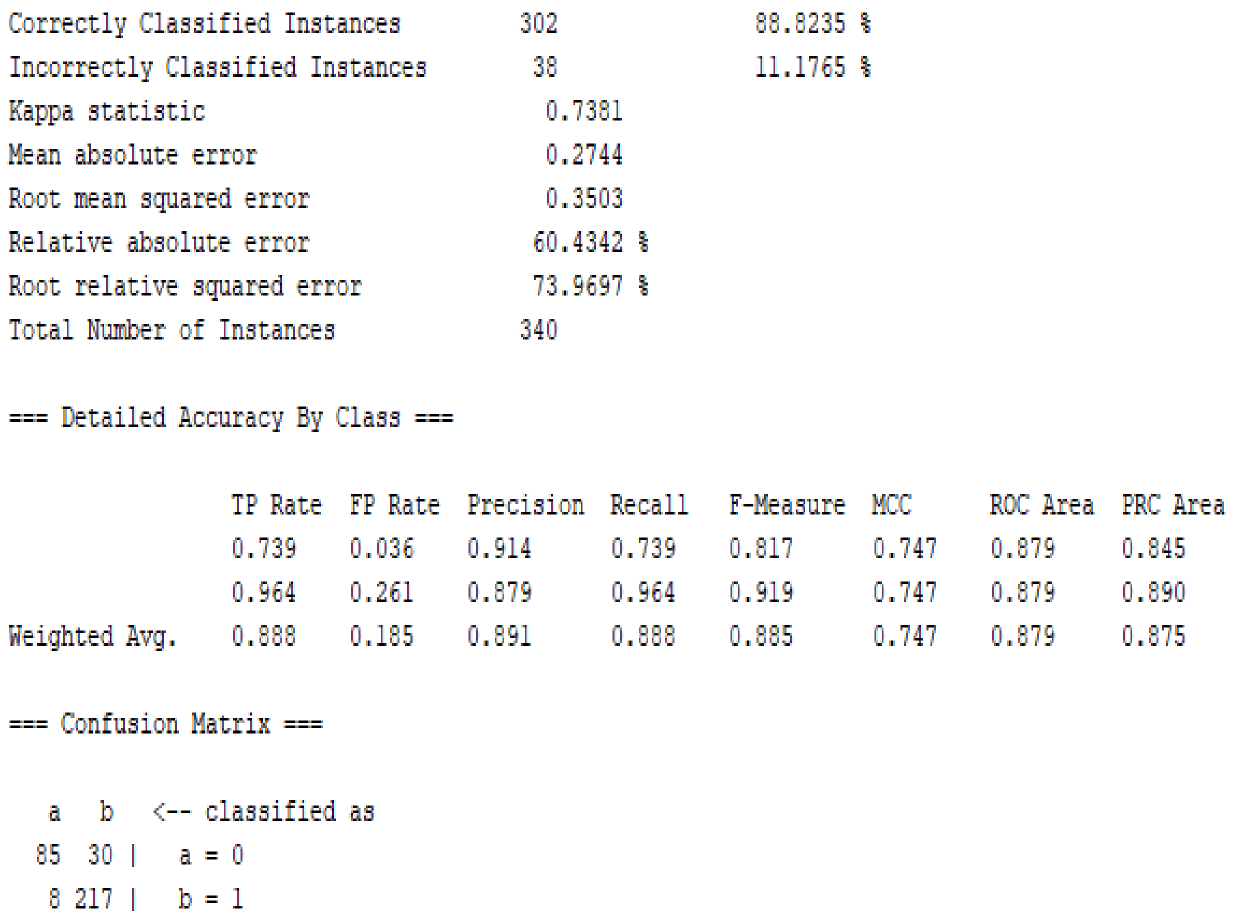
2. False positive (FP): However, if the actual value is n then it is said to be a false positive.

3. Precision and recall: Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. Precision can be seen as a measure of exactness or quality, whereas recall is a measure of completeness or quantity. Recall is nothing but the true positive rate for the class



```
Naive Bayes Classifier

                Class
Attribute           0      1
               (0.35) (0.65)
==============================
credit
  mean          5.0305 5.2616
  std. dev.     2.8686 2.9588
  weight sum       351    649
  precision       0.01   0.01

expenses
  mean           3.161 5.6492
  std. dev.     2.4973 2.6392
  weight sum       351    649
  precision       0.01   0.01

income
  mean          3.4163 6.0329
  std. dev.     2.4302 2.6636
  weight sum       351    649
  precision       0.01   0.01
```

Figure 3: Naive Bayes for month1



```
Correctly Classified Instances         302               88.8235 %
Incorrectly Classified Instances        38               11.1765 %
Kappa statistic                          0.7381
Mean absolute error                      0.2744
Root mean squared error                  0.3503
Relative absolute error                 60.4342 %
Root relative squared error             73.9697 %
Total Number of Instances              340

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.739    0.036    0.914      0.739    0.817      0.747    0.879     0.845
                 0.964    0.261    0.879      0.964    0.919      0.747    0.879     0.890
Weighted Avg.    0.888    0.185    0.891      0.888    0.885      0.747    0.879     0.875

=== Confusion Matrix ===

   a   b   <-- classified as
  85  30 |   a = 0
   8 217 |   b = 1
```

Figure 4: classifiers differences

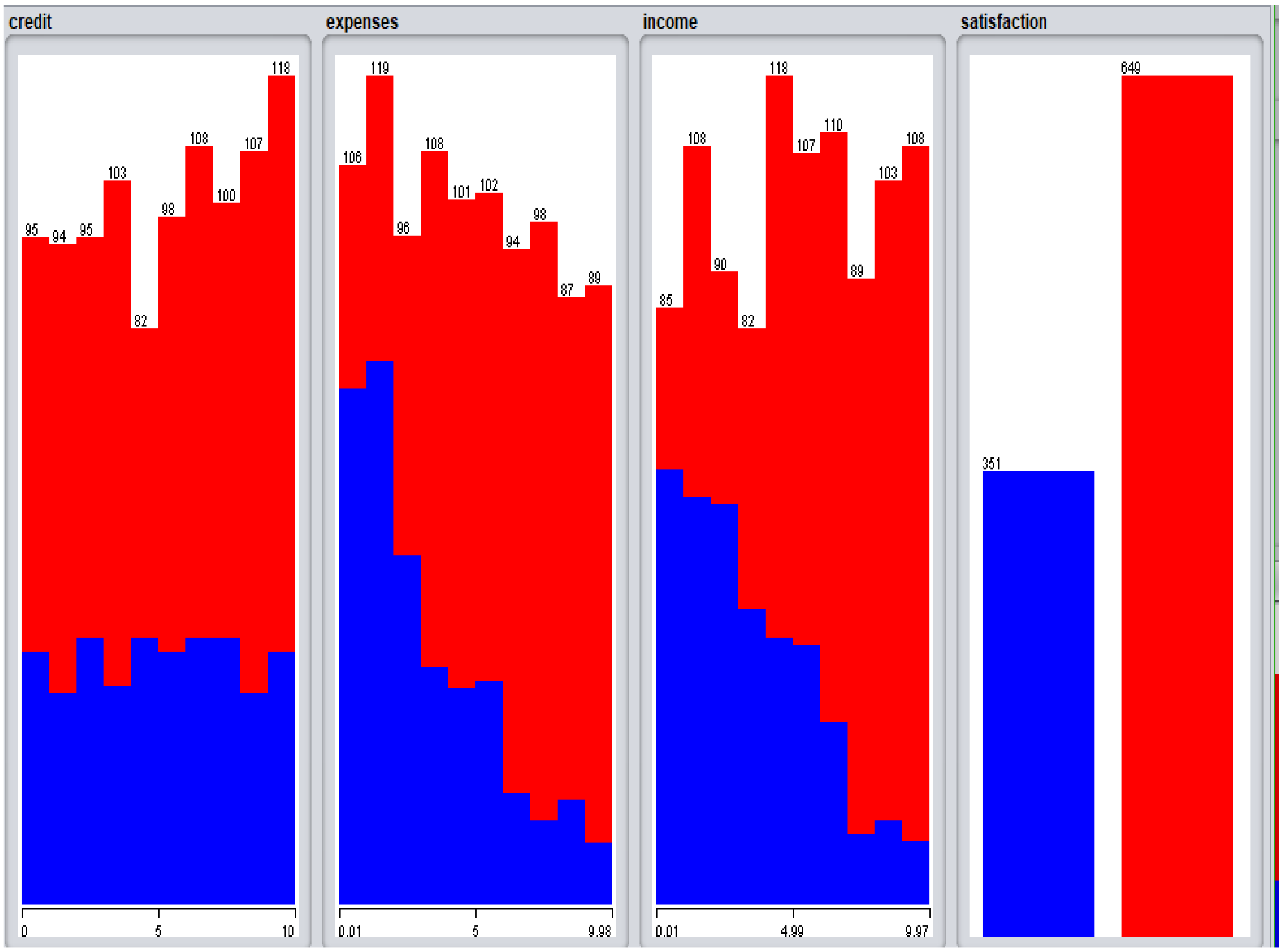


Figure 5: general view month1

The plot margin for naïve Bayes classifier on Month 1 is given below.

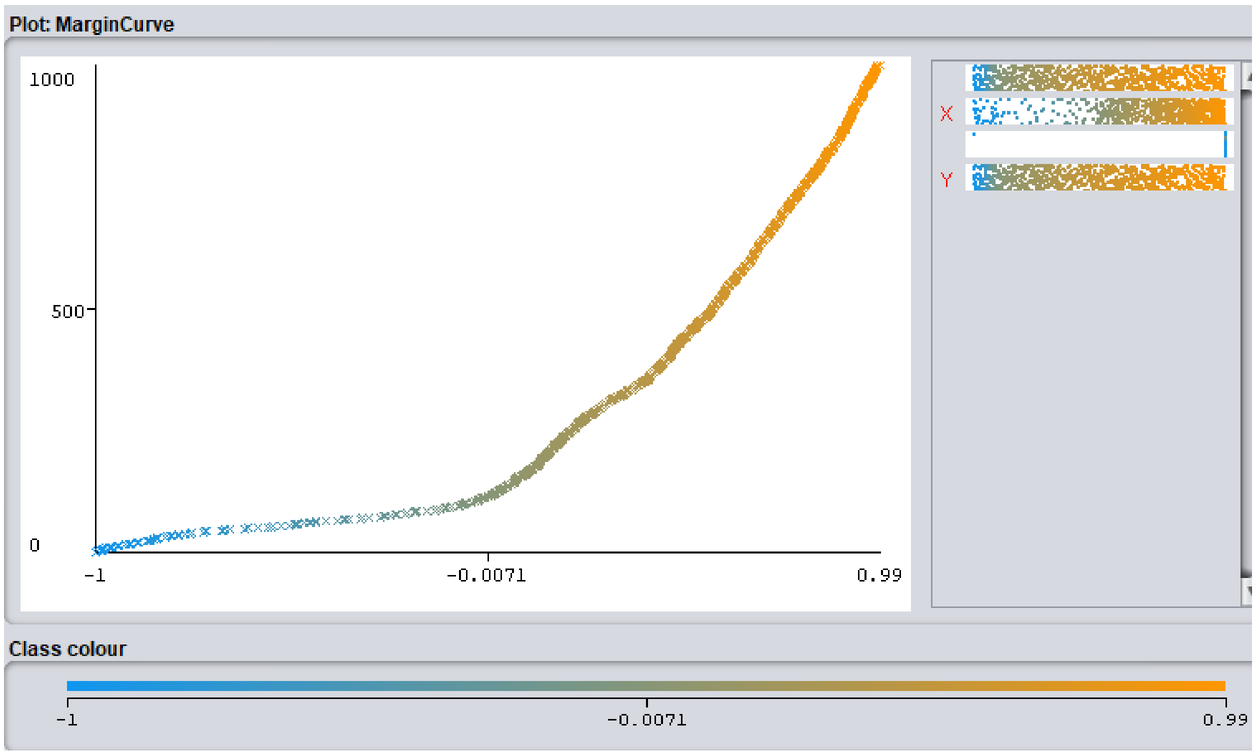
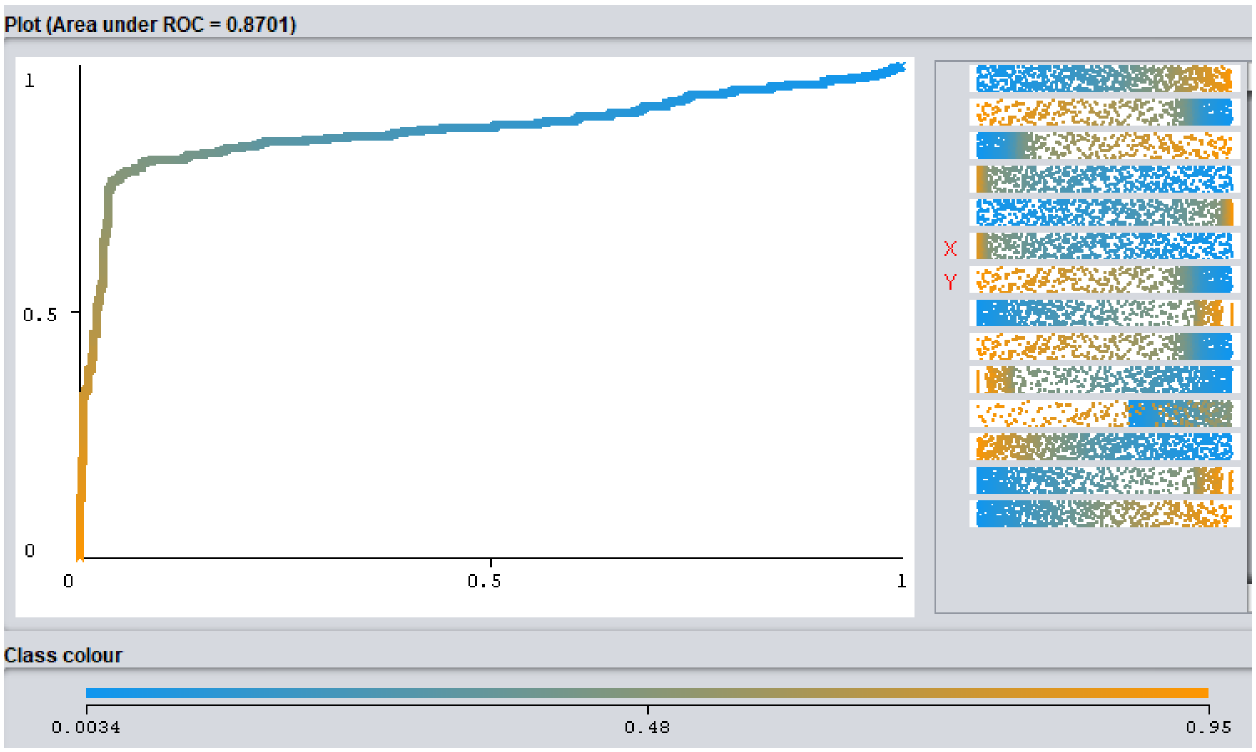


Figure 6: The plot margin for naïve Bayes classifier on Month1

A margin curve generates points illustrating the prediction margin. The margin is defined as the difference between the probability predicted for the actual class and the highest probability predicted for the other classes. One hypothesis as to the good performance of boosting algorithms is that they increases the margins on the training data and this gives better performance on test data.



Generates points illustrating prediction tradeoffs that can be obtained by varying the threshold value between classes. For example, the typical threshold value of 0.5 means the predicted probability of "positive" must be higher than 0.5 for the instance to be predicted as "positive". The resulting dataset can be used to visualize precision/recall tradeoff, or for ROC curve analysis (true positive rate vs false positive rate). Weka just varies the threshold on the class probability estimates in each case. The Mann Whitney statistic is used to calculate the AUC.

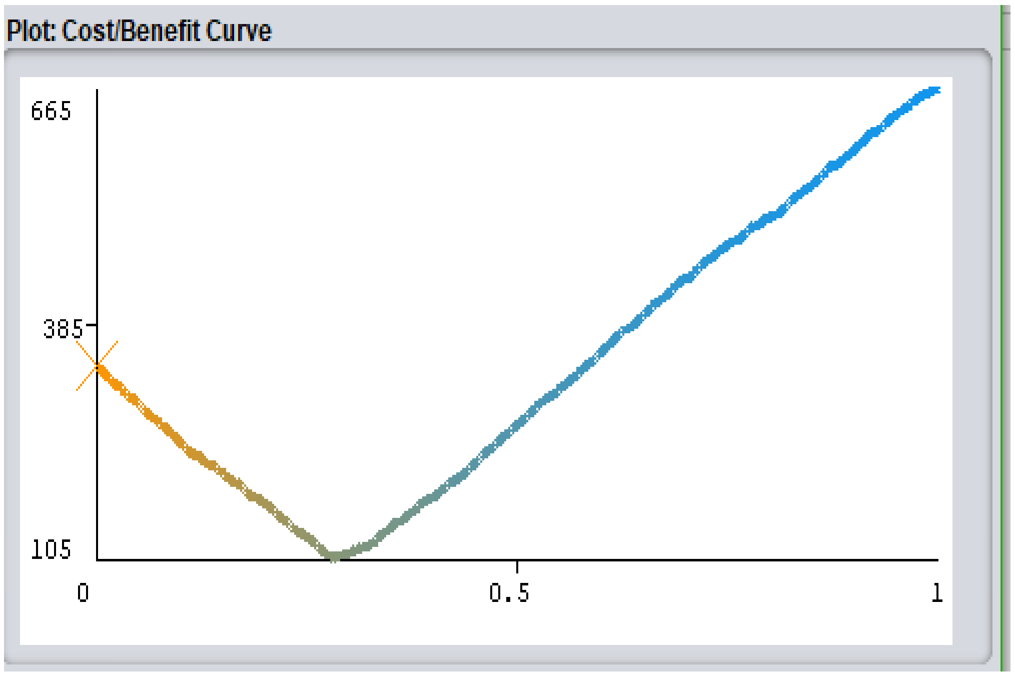


Figure 7: Cost benefit curve

A cost benefit analysis (also known as a benefit cost analysis) is a process by which organizations can analyze decisions, systems or projects, or determine a value for intangibles. The model is built by identifying the benefits of an action as well as the associated costs, and subtracting the costs from benefits. When completed, a cost benefit analysis will yield concrete results that can be used to develop reasonable conclusions around the feasibility and/or advisability of a decision or situation.

The cost/benefit analysis tool is particularly useful for the analysis of predictive analytic outcomes for direct mail campaigns (or any ranking application where costs are involved). It allows the user to explore various cost/benefit tradeoffs by interactively selecting different population sizes from the ranked list of prospects or by varying the threshold on the predicted probability of the positive class.

Cost–benefit analysis (CBA), sometimes called benefit costs analysis (BCA), is a systematic approach to estimating the strengths and weaknesses of alternatives used to determine options which provide the best approach to achieving benefits while preserving savings (for example, in transactions, activities, and functional business requirements). A CBA may be used to compare completed or potential courses of actions, or to estimate (or evaluate) the value against the cost of a decision, project, or policy. It is commonly used in commercial transactions, business or policy decisions (particularly public policy), and project investments.

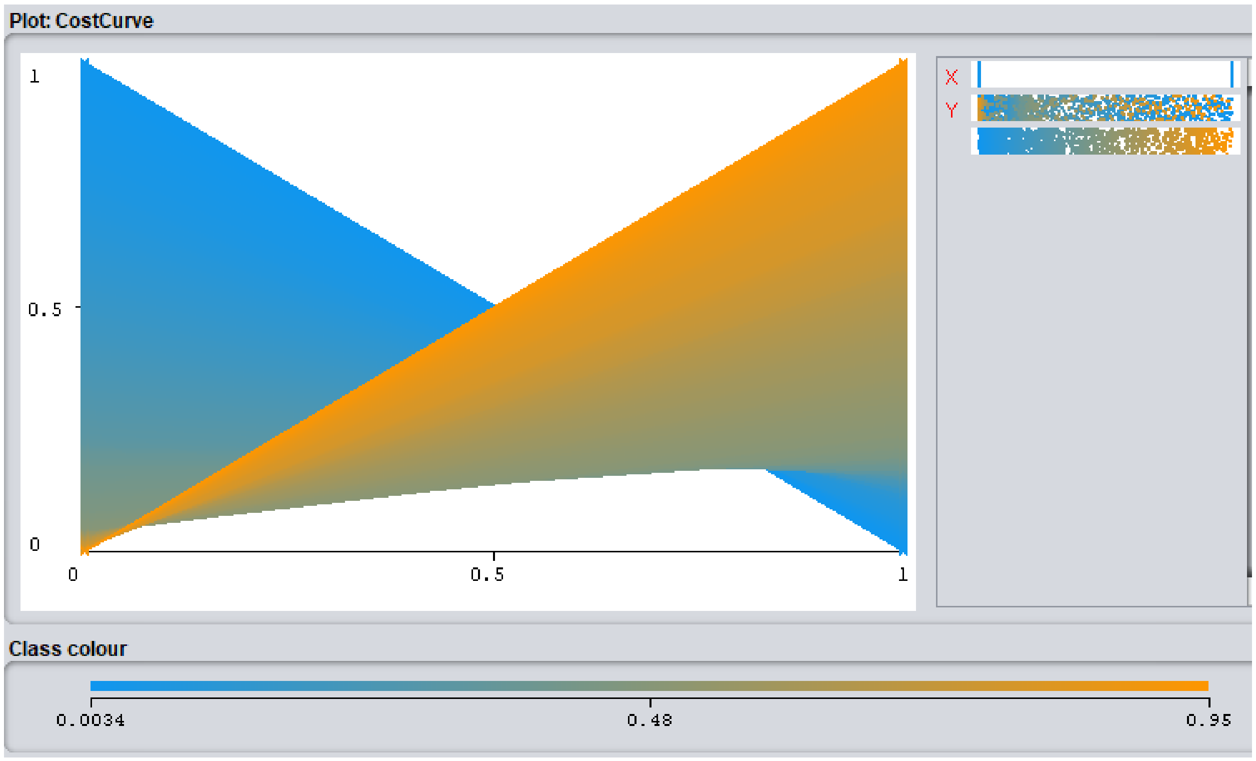


Figure 8: cost curve

A cost curve represents many classifiers and thresholds in a simple informative chart. Each software system is unique in a sense that it is developed following specific processes, quality assurance techniques, and it operates in environments in which the occurrence of failures implies different consequences. The assessments of failure criticality or fault resolution priority have been a part of software V& V activities for a long time. Software development teams typically have a good understanding of the range of pertinent cost implications of failures, making cost curve evaluations practical.

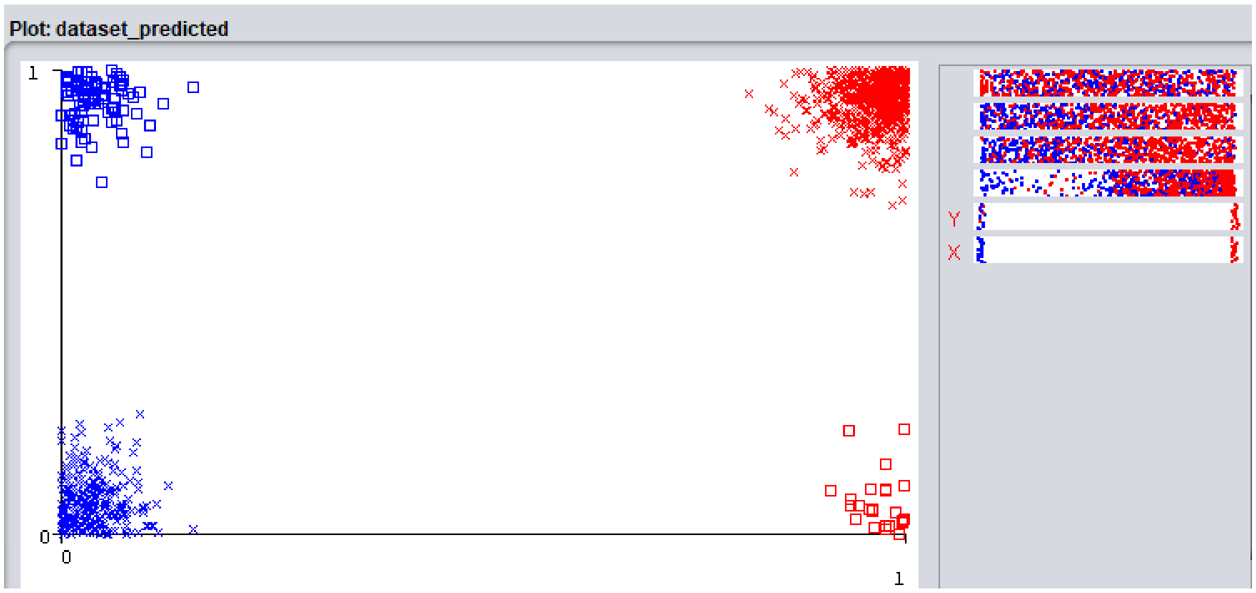


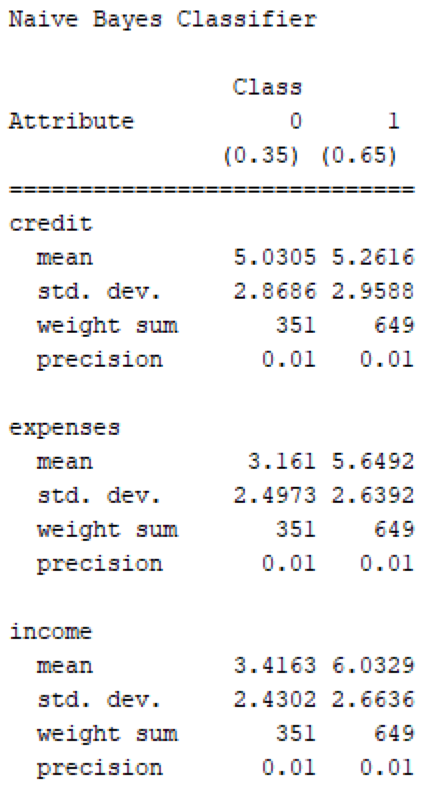
Figure 9: dataset prediction

Prediction curve showing

The contingency table can derive several evaluation "metrics". To draw an ROC curve, only the true positive rate (TPR) and false positive rate (FPR) are needed (as functions of some classifier parameter). The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

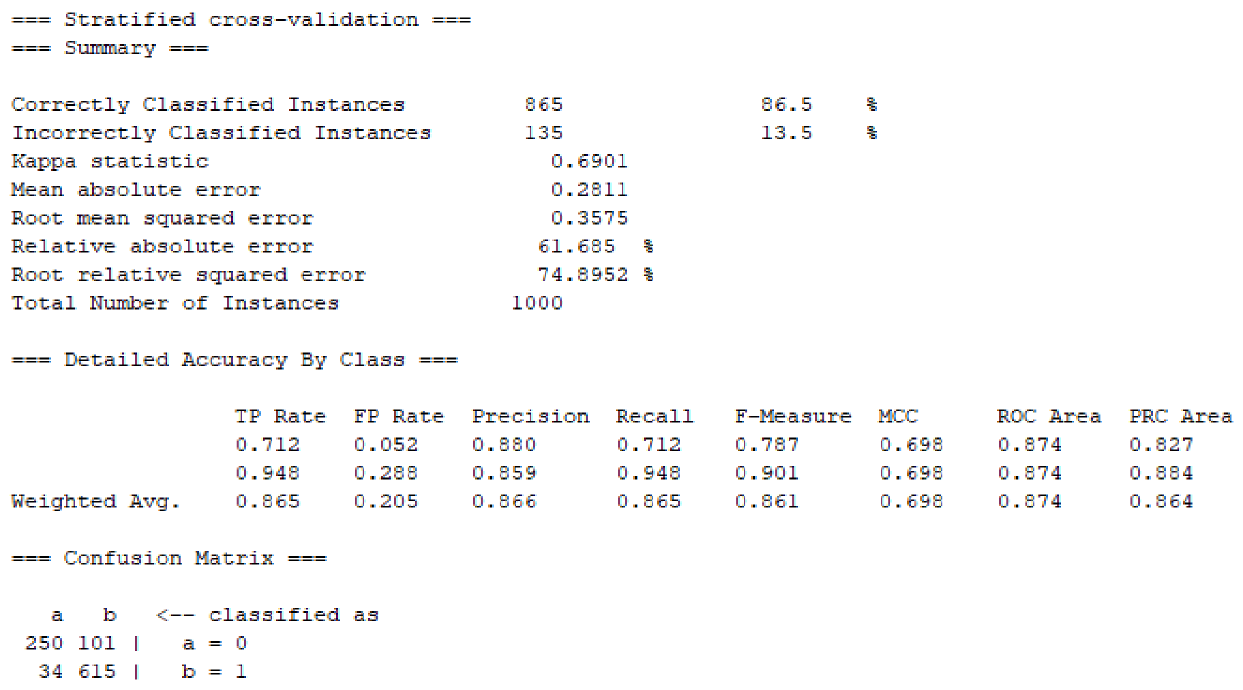
An ROC space is defined by FPR and TPR as x and y axes, respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). Since TPR is equivalent to sensitivity and FPR is equal to 1 − specificity, the ROC graph is sometimes called the sensitivity vs (1 − specificity) plot. Each prediction result or instance of a confusion matrix represents one point in the ROC space.

The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification. A random guess would give a point along a diagonal line (the so-called line of no-discrimination) from the left bottom to the top right corners (regardless of the positive and negative base rates). An intuitive example of random guessing is a decision by flipping coins. As the size of the sample increases, a random classifier's ROC point tends towards the diagonal line. In the case of a balanced coin, it will tend to the point (0.5, 0.5).



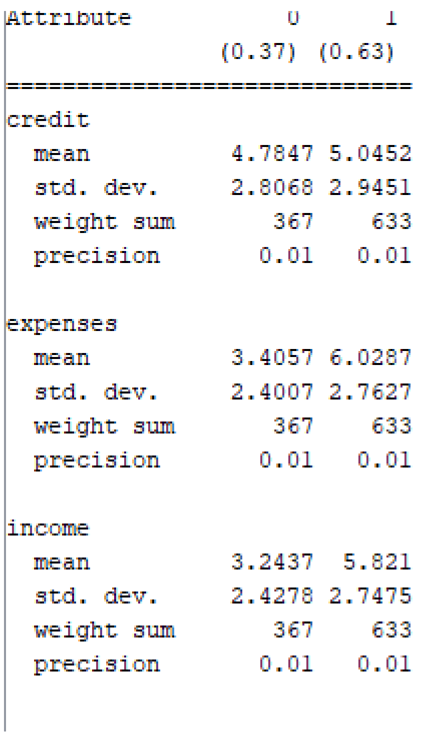
```
Naive Bayes Classifier

                 Class
Attribute            0      1
                (0.35) (0.65)
==============================
credit
  mean           5.0305 5.2616
  std. dev.      2.8686 2.9588
  weight sum        351    649
  precision        0.01   0.01

expenses
  mean            3.161 5.6492
  std. dev.      2.4973 2.6392
  weight sum        351    649
  precision        0.01   0.01

income
  mean           3.4163 6.0329
  std. dev.      2.4302 2.6636
  weight sum        351    649
  precision        0.01   0.01
```

Figure 10: Analysis for month 2



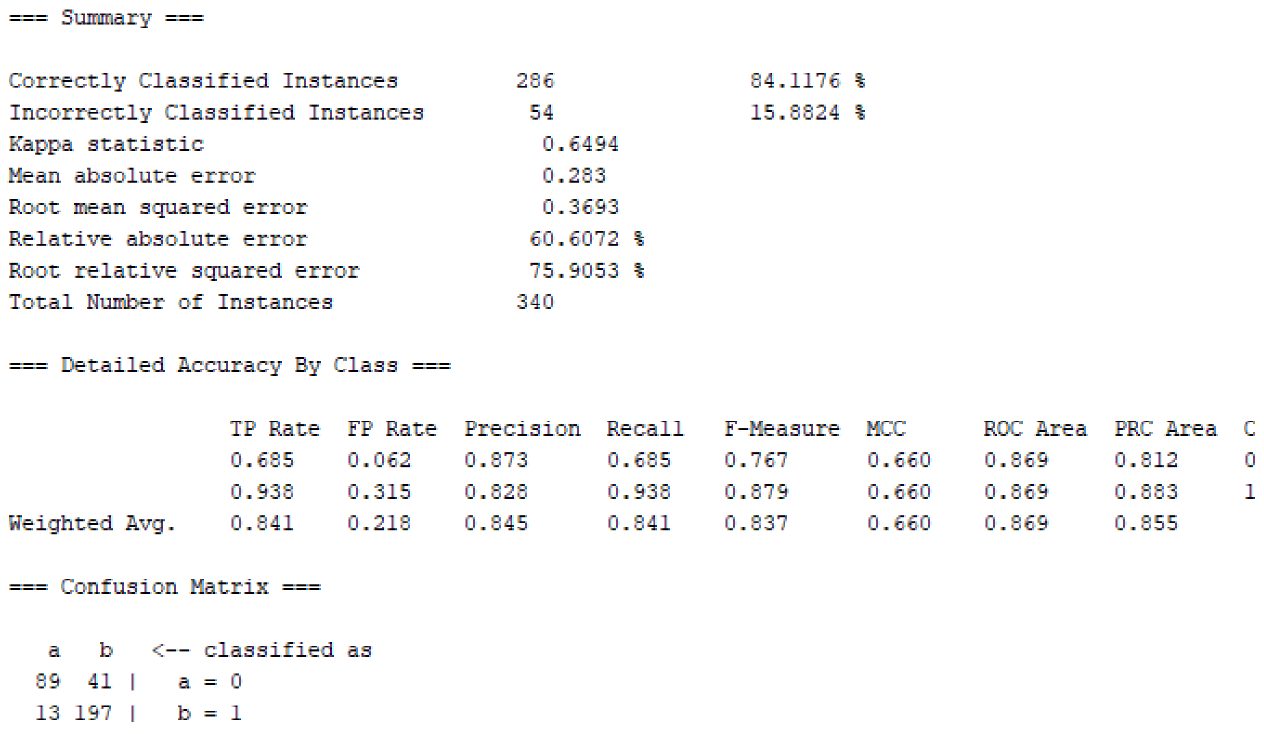
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         865               86.5    %
Incorrectly Classified Instances       135               13.5    %
Kappa statistic                          0.6901
Mean absolute error                      0.2811
Root mean squared error                  0.3575
Relative absolute error                 61.685  %
Root relative squared error             74.8952 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.712    0.052    0.880      0.712    0.787      0.698    0.874     0.827
                 0.948    0.288    0.859      0.948    0.901      0.698    0.874     0.884
Weighted Avg.    0.865    0.205    0.866      0.865    0.861      0.698    0.874     0.864

=== Confusion Matrix ===

   a   b   <-- classified as
 250 101 |   a = 0
  34 615 |   b = 1
```

Figure 11: classifier month2



```
Attribute           0      1
               (0.37) (0.63)
==============================
credit
  mean          4.7847 5.0452
  std. dev.     2.8068 2.9451
  weight sum       367    633
  precision       0.01   0.01

expenses
  mean          3.4057 6.0287
  std. dev.     2.4007 2.7627
  weight sum       367    633
  precision       0.01   0.01

income
  mean          3.2437  5.821
  std. dev.     2.4278 2.7475
  weight sum       367    633
  precision       0.01   0.01
```

Figure 12: Analysis for Month 3



```
=== Summary ===

Correctly Classified Instances         286               84.1176 %
Incorrectly Classified Instances        54               15.8824 %
Kappa statistic                          0.6494
Mean absolute error                      0.283
Root mean squared error                  0.3693
Relative absolute error                 60.6072 %
Root relative squared error             75.9053 %
Total Number of Instances              340

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  C
                 0.685    0.062    0.873      0.685    0.767      0.660    0.869     0.812     0
                 0.938    0.315    0.828      0.938    0.879      0.660    0.869     0.883     1
Weighted Avg.    0.841    0.218    0.845      0.841    0.837      0.660    0.869     0.855

=== Confusion Matrix ===

   a   b   <-- classified as
  89  41 |   a = 0
  13 197 |   b = 1
```

Figure 13: classifier month3

Discussions

It can be observed from the results that the percentage of performance decreases as we move from month 1 to month 3.

This implies the drift of data.

The change to the data could take any form. It is conceptually easier to consider the case where there is some temporal consistency to the change such that data collected within a specific time period show the same relationship and that this relationship changes smoothly over time.

Note that this is not always the case and this assumption should be challenged. Some other types of changes may include:

A gradual change over time.
A recurring or cyclical change.
A sudden or abrupt change.

Different concept drift detection and handling schemes may be required for each situation. Often, recurring change and long-term trends are considered systematic and can be explicitly identified and handled.

Concept drift may be present on supervised learning problems where predictions are made and data is collected over time. These are traditionally called online learning problems, given the change expected in the data over time.

There are domains where predictions are ordered by time, such as time series forecasting and predictions on streaming data where the problem of concept drift is more likely and should be explicitly tested for and addressed.

A common challenge when mining data streams is that the data streams are not always strictly stationary, i.e., the concept of data (underlying distribution of incoming data) unpredictably drifts over time. This has encouraged the need to detect these concept drifts in the data streams in a timely manner

Conclusion and future work

In this work we have used dataset from one organization and apply various algorithms to scrutinize which perform better on financial data analysis for concept drift detection. In the future work, we aim to work on the dataset of multiple enterprise then at the national level and continent

References.

[1] F. Wang and Y. Xu, "What Determines Chinese Stock Returns?"

Financial Analysts Journal, vol. 60, pp. 65-77, 2004.

[2] S. Thawornwong and D. Enke, "The adaptive selection of financial

and economic variables for use with artificial neural networks,"

Neurocomputing, vol. 56, pp. 205-232, 2004.

[3] S. Chun and Y. Park, "Dynamic adaptive ensemble case-based

reasoning: application to stock market prediction," Expert Systems

with Applications, vol. 28, pp. 435-443, 2005.

[4] J. O, J. Lee, J. W. Lee, and B. Zhang, "Adaptive stock trading with

dynamic asset allocation using reinforcement learning," Information

Sciences, vol. 176, pp. 2121-2147, 2006.

[5] G. Armano, A. Murru and F. Roli, "Stock Market Prediction by a

Mixture of Genetic-Neural Experts," International Journal of Pattern Recognition and Artificial Intelligence, vol. 16, pp. 501-526, 2002-08-01 2002.

[6] A. Chen, W. Lin and Y. Chen, "An Intelligent Model for Stock Investment with Buffett Strategy, Classifier System, Neural Network and Linear Programming," The fourth international conference on electronic business, pp. 255-260, 2004.

[7] N. Ren, M. Zargham and S. Rahimi, "A Decision Tree-Based Classification Approach To Rule Extraction For Security Analysis," International Journal of Information Technology & Decision Making, vol. 1, pp. 227-240, 2006.

[8] I. Zliobaite, "Learning under Concept Drift: an Overview," CoRR, vol. abs/1010.4784, 2010.

[9] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A casebased technique for tracking concept drift in spam filtering," Knowledge-Based Systems, vol. 18, pp. 187-195, 2005.

[10] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C., 2003, pp. 226-235.