LEARNING TO CLASSIFY IMBALANCED DATA STREAMS FOR BINARY CLASSIFICATION

Submitted in partial fulfillment of the requirements
of the degree of

B. E. Computer Engineering

By

Megha Jakhotia	**60004140033**
Yash Kapadia	**60004140040**
Nikita Luthra	**60004140047**

Guide(s):

Prof. Kiran Bhowmick
Assistant Professor

Department of Computer Engineering
D. J. Sanghvi College of Engineering
Mumbai – 400 056

University of Mumbai
2017-2018

CERTIFICATE

This is to certify that the project entitled **“Learning to classify Imbalanced data streams for binary classification”** is a bonafide work of **“Megha Jakhotia” (60004140033), “Yash Kapadia” (60004140040) and “Nikita Luthra” (60004140047)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering

Prof. Kiran Bhowmick
Internal Guide

Dr. Narendra Shekokar
Head of Department

Dr. Hari Vasudevan
Principal

Project Report Approval for B.E.

This project report entitled ***“Learning to classify Imbalanced data streams for binary classification”*** by ***“Megha Jakhotia” (60004140033), “Yash Kapadia” (60004140040) and “Nikita Luthra” (60004140047)*** is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.______________________________

2.______________________________

Date:

Place:

Declaration

We declare that this written submission represents my/our ideas in my/our own words and where others' ideas or words have been included, I/We have adequately cited and referenced the original sources. I/We also declare that I/We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my/our submission. I/We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

Megha Jakhotia

60004140033

____

Yash Kapadia

60004140040

____

Nikita Luthra

60004140047

Date:

Abstract

Most of the Classifiers in supervised learning predict accurately for balanced data sets. But when dealt with imbalanced data set, it intends to be biased towards majority class thus ignoring the minority class. Learning from imbalanced data has conventionally been conducted on stationary data sets. Recently, there have been several methods proposed for mining imbalanced data streams. Training data is read in consecutive chunks, each of which is considered as a conventional imbalanced data set called data streams. We will be working on the problem of classifying imbalanced data streams in binary classifiers. Dealing with imbalanced datasets entails strategies such as improving classification algorithms or balancing classes in the training data (data pre-processing) before providing the data as input to the machine learning algorithm.

Many practical classification tasks involve dealing with imbalanced class distributions, such as detection of fraudulent credit card transactions and diagnosis of rare diseases. Therefore, sampling methods can easily be performed to make data chunks class-balanced, such as over-sampling by generating synthetic minority class instances, oversampling by reusing historical training instances, and under-sampling by clustering the majority class.

The issues in imbalanced data stream contain the non-availability of class labels. We will be finding a method via which we will classify the data so that it belongs to the appropriate class by not being biased towards the majority class and not considering the minority class as noise.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Description

Classification is the problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known. The training set on which a classification algorithm is trained might not contain a balance between the number of examples from the different classes and this raises the issue of Imbalanced data in classification problems.

Canonical machine learning algorithms assume that the number of objects in considered classes is roughly similar. However, in many real-life situations the distribution of examples is skewed since representatives of some of classes appear much more frequently. This poses a difficulty for learning algorithms, as they will be biased towards the majority group. At the same time usually the minority class is the one more important from the machine learning perspective, as despite its rareness it may carry important and useful knowledge.

With data being generated continuously due to the advent of technology, analysis of this data has become prime interest of many business applications. Data streams that are generally characterized by data that has high speed, volume and infinite length also suffer from skewed distribution. The classifier in such streams is biased towards the majority class and tends to misclassify the instances of the minority class. Classification cost for misclassifying minority class instances can be expensive for many of the real time applications like financial fraud detection, medical fraud detection, oil spillage detection etc. Learning to classify from these imbalanced data streams is challenging because unlike static data the data cannot be stored and is seen at most once.

Data streams concern many applications involving large and temporal data sets such as telephone records, banking, multimedia, network traffic monitoring, and sensor network data processing, etc. In these applications, the data patterns may evolve continuously and depend on time or depend on the events. In stream models, the data elements can only be accessed in the order in which they arrive; random access to the data is not allowed; and memory is limited in relative to the number of data elements. Due to the limited memory availability, it is impossible to load the entire data set into memory. Mining data streams poses great challenges.

1.2 Problem Formulation

Rare events, unusual patterns and abnormal behavior are difficult to detect, but often require responses from various management functions in a timely manner. By definition, rare events refer to events that occur much less frequently than commonly occurring events. Examples of rare events include software defects, natural disasters, cancer gene expressions, fraudulent credit card transactions, and telecommunications fraud.

In the field of data mining, detecting events is a prediction problem, or, typically, a data classification problem. Rare events are difficult to detect because of their infrequency and casualness; however, misclassifying rare events can result in heavy costs. This model aims to classify the data accurately by increasing various performance measures such as recall, precision and not only rely on the confusion matrix and accuracy of the algorithm.

Most of the research papers assume the class labels are always available for classification and hence most have them are supervised in nature. These algorithms suffer as the actual labels of the class are available only for a limited amount of data. Semi supervised models are tried for this reason, but to further reduce the cost of any labelling active learning are advantageous. In addition to the imbalanced data streams, finding an algorithm superior to the existing algorithms by refining them that will also implement active learning by taking care of proper designation labels is the overall goal of the proposed project.

1.3 Motivation

There is currently a great deal of interest in utilizing automated methods—particularly data mining and machine learning methods—to analyze the enormous amount of data that is routinely being collected. An important class of problems involves predicting future events based on past events. Event prediction often involves predicting rare events. Rare events lurk in many shapes, including natural disaster (e.g. earthquakes, solar flares, tornado), anthropogenic hazards (e.g. financial fraud, industrial accidents, violent conflict) and diseases. For financial fraud detection, invalid transactions may only emerge out of hundreds of thousands of transaction records, but failing to identify a serious fraudulent transaction would cause enormous losses. The scarce occurrences of rare events impair the detection task to imbalanced data classification problem.

Numerous papers and articles have been published, wherein classification algorithm taking imbalanced data into consideration have been discussed in various real-life applications like Diseases, Information Technology, Business and Environmental management [1]. This project aims to develop an algorithm that solves these real-time issues by considering their original occurrence frequency and develop a generalized algorithm to work on any type of data stream.

1.4 Proposed Solution

The data set has been divided into two parts. The first part will be used to train the model and the second part will be used as real-time streams input to the model for prediction. The first part of the data is split into three sets: training, cross-validation and test. The 3 sampled data sets are transformed to their respective data streams using the stream package in R.

The train stream is then used as input to the clustering algorithm from the streamMOA package in R. The labels of the input data arrive with some delay and these labels are then compared with the labels assigned by the clustering algorithm to find the actual ones. These assigned classes are then given to the classifier along with the training stream to train the classifier. To handle the issue of imbalance, the algorithmic level approach of ensemble classifier with boosting methodology is used. The classifier predictions are evaluated with the help of true labels on the basis of confusion matrix and recall.

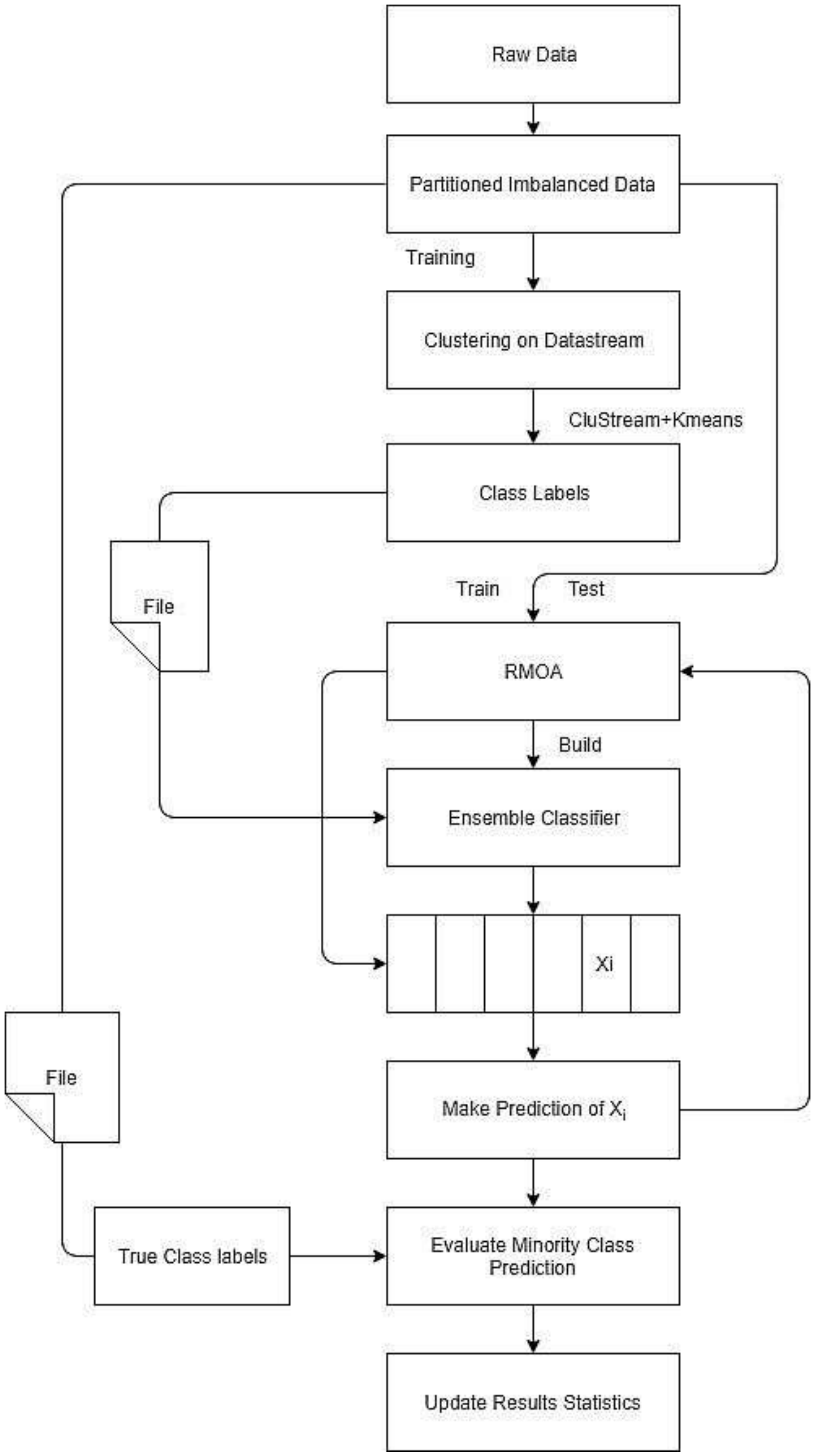


Fig. 1.1 Proposed Solution

1.5 Scope of the project

In the healthcare industry, the rare infectious diseases could be treated successfully when detected at the initial stages. This has been one of the biggest problems, the correct prediction of diseases – rare diseases in particular like cancer malignancy. Healthcare datasets dealing with rare diseases are often imbalanced: hence the difficulty in accurately predicting them. Along with being imbalanced, the sample sets are enormous and also varied in respect to the individual populations of different countries, which all-together increases the problems for correct classification.

Even when we divert from the healthcare field similar problems occur in other fields for example the banking industry, where the correct classification of fraudulent credit card transactions are critical. Also, the detection of the oil spills is vital for the protection of the ocean ecosystems. So in this project, keeping the underlying problem common to various fields in mind and finding a solution that would be generalized and would be useful to all such problems.

The three insights of the problem considered are: class imbalance, data streams and absence of labels. The classifier deals with imbalanced data streams and classifies the data accurately. The ensemble classifier algorithm is used to handle the imbalance data issue. Its performance is judged on the performance metrics like Recall and Confusion matrix. Also, the datasets would actually be data streams that the classifier would view only once and thus even without viewing the data previously the predictions would be error-free.

Chapter 2

Review of Literature

2.1 Imbalanced Data Classification

The fundamental issue with the imbalanced learning problem is the ability of imbalanced data to significantly compromise the performance of most standard learning algorithms. Most standard algorithms assume or expect balanced class distributions or equal misclassification costs. Therefore, when presented with complex imbalanced data sets, these algorithms fail to properly represent the distributive characteristics of the data and resultantly provide unfavorable accuracies across the classes of the data.

Imbalance data sets degrades the performance of data mining and machine learning techniques as the overall accuracy and decision making be biased to the majority class, which lead to misclassifying the minority class samples or furthermore treated them as noise.

We may distinguish three main approaches to learning from imbalanced data[2]:

1. Data-level methods that modify the collection of examples to balance distributions and/or remove difficult samples. Some of the resampling techniques are Random under-sampling, Random over-sampling, cluster based over sampling, informed over sampling, modified synthetic minority over sampling technique (MSMOTE).

2. Algorithm-level methods that directly modify existing learning algorithms to alleviate the bias towards majority objects and adapt them to mining data with skewed distributions. Some of the techniques are Bagging based, Boosting based, Adaptive boosting, Gradient tree boosting, XG boost.

3. Hybrid methods that combine the advantages of two previous groups. This approach combines ensemble learning with data preprocessing techniques which is commonly

found in traditional ensembles like boosting and bagging. They usually perform preprocessing before ensemble learning.

Several methods proposed for solution of imbalance class problems include re-sampling and feature selection at the data level and other ones at the algorithm level such as cost sensitive and single class learning.

2.1.1 Sampling Methods

Sampling methods is a pre-processing of data which handle the imbalance problem by constructing balanced training data set and adjusting the prior distribution for minority and majority class. Sampling methods include under sampling and over sampling methods. Under sampling balance the data by removing samples from majority class and over sampling balance the data by create copies of the existing samples or adding more samples to the minority class. Significant differences between majority and minority class can be handled by oversampling when data is highly imbalanced.

2.1.2 Cost sensitive learning

In many imbalance problems, not only the data distribution is skewed but also the misclassification error cost is uneven. The cost learning techniques take the misclassification cost in its account by assigning higher cost of misclassification to the positive class (minority class) and generate the model with lowest cost. However, this misclassification errors costs are often unknown and furthermore, cost sensitive learning may lead to over fitting [3]. Another cost sensitive learning approach used in unbalance dataset is adjusting the decision threshold of the standard machine learning algorithms, wherever the selection of threshold is an effective factor on the performance of learning algorithms.

2.1.3 Ensemble based methods

Ensemble is a combination of multiple classifiers so as to improve the generalization ability and increase the prediction accuracy. The most popular combining techniques are boosting and bagging. In boosting, each classifier is dependent on the previous one, and focuses on the previous one’s errors. Examples that are misclassified in previous classifiers are chosen more often or weighted more heavily. Whereas, in bagging, each model in the ensemble votes with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set.

2.2 Data Streams

Data streams are continuous flows of data in which the training data is read in consecutive chunks. Examples of data streams include network traffic, sensor data, call center records and so on. Data Stream Mining is the process of extracting knowledge structures from continuous, rapid data records. The sheer volume and speed of data streams pose a great challenge for the data mining community to mine them. Data streams are an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities. Data arrives in a stream or streams, and if it is not processed immediately or stored, then it can be lost forever. Data arrives so rapidly that it is not feasible to store it all in active storage (i.e., in a conventional database), and then interact with it at the time of our choosing.

2.3 Labelling

The data streams in many applications are characterized by imbalanced class distribution. The absence of labels at the start of the data stream has caused serious issues. It is required to learn the class labels that are represented by the data stream to train the classifier and classify the data stream points that come later for prediction. Most of the papers have assumed the presence of labels from the start or at least some data points with their class labels are given to train the classifier.

Chapter 3

System Analysis

3.1 Functional Requirements

3.1.1 Clustering

In this module, the purpose is to identify the class labels that are represented by the data stream since they are not already known. Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters. The clustering algorithm must be accurate to cluster similar points together thus the points with same labels are in a single cluster and the points in a single cluster represent a single class.

3.1.2 Classification

In this module, the purpose is to identify the class to which a particular data sample belongs to. Classification is the most important task as it would signify the success or failure of this algorithm. The imbalanced data streams are given as the input which the classifier would view only once. The classifier has to be well-trained and accurate enough to correctly classify the input data streams into the majority and minority classes. It should not consider the incoming data sets of minority classes as noise. The goal of classification is to find a balance in the classes among the imbalanced input data. This accuracy would be the most critical analytic measurement of the implemented classifier.

3.1.3 Labelling

When data occurs without the class labels, it is required to identify the class label for that point without any previous knowledge. For this purpose clustering needs to be done on data stream and find the appropriate clusters and class labels.

3.2 Non-functional Requirements

3.2.1 Accuracy

It is extremely important that the output predictions produced by this system are accurate. As the real-time implementations of these problems are critical, the system must generate optimal results. We will not only be using the confusion matrix but also other performance measures that include recall, precision, G-mean and F-score.

Evaluation metrics are most important aspect for a classifier that acts as an indicator of the performance of the model. Most data mining algorithms do not account for the underlying class-imbalance. For a two class problem, confusion matrix is calculated as shown below:

	Predicted as positive	Predicted as negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Table 3.1: Confusion Matrix

The confusion matrix also helps us to measure other important performance measures related to the classifier such as accuracy, sensitivity, specificity, precision and recall.

Accuracy represents how often the classifier predicts correct outcome. It is defined as:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Sensitivity and Specificity are used when the performance of the classifier is tested for their recognition of positive and negative instances. Sensitivity represents how often does the classifier predict 0 (Non-Risk) when it is actually 0 (Non-Risk).

$$Sensitivity = \frac{TP}{(TP+FP)}$$

Specificity represents how often the classifier predicts 1 (Risk) when it is actually 1 (Risk).

$$Specificity = \frac{TN}{(TN+FP)}$$

Precision is a measure of exactness. It determines the total percentage of instances that the classifier labeled as minority actually belongs to the minority class.

$$\Pr ecision = \frac{TP}{(TP+FP)}$$

Recall is a measure of completeness. It determines the percentage of minority class instances that are labeled correctly as minority.

$$\text{Re}\,call = \frac{TP}{(TP+FN)}$$

The learning objective of class imbalance learning is to improve recall without hurting precision. But, recall and precision can be conflicting.

F-measure is defined to show the trade-off between them. While precision is used in problems interested in high performance on only one class, F measure and G-mean are used when the performance on both classes – majority and minority classes - needed to be high.

F-Measure: It is a harmonic mean of Precision & Recall and reflects the goodness of the classifier in presence of the minority class.

$$F - measure = \frac{(1+\beta^2) * precision * recall}{\beta^2 * precision + recall}$$

Where β is usually set 1 and corresponds to the relative importance of recall and precision.

G-Mean: G-Mean is a metric that measures the balanced performance of a learning algorithm between the classes. A classifier with high value for G-mean have high accuracies for both the classes.

$$G - Mean = \sqrt{recall * specificity}$$

$$G - Mean = \sqrt{\frac{TP}{(TP+FN)} * \frac{TN}{(TN+FP)}}$$

3.2.2 Performance Requirements

In the mentioned system, the functionality plays a very important role. It tells us how the system predicts the output of the set of inputs. The criticality of these functions tells us the performance and success of the system. The system would be subject to vast streams of input data and at a very fast rate. Under these conditions the performance of the classifier should not hinder. The performance of the system completely depends upon how accurately the system will be able to identify the classes of the data, which in turn will depend on the balance of the system. Also, it is important to determine the effects of absence of class labels which leads to the changes in the model, on overall performance and behaviour. The system's ability to transform its algorithm according to the changes in the input data streams is an important aspect for accurate predictions.

3.3 Specific Requirements

3.3.1 Datasets

Kaggle.com and DataCamp will be used to obtain the datasets. The data sets taken would belong to different fields that face the problems similar to the ones we have considered. The datasets obtained from these would be converted to data streams which would be available for a short amount of time to the classifier. The datasets used in this project are the following:

Mammography dataset:

The original Mammography (Woods et al., 1993) data set[16] was made available by the courtesy of Aleksandar Lazarevic. This dataset is publicly available in openML. It has 11,183 samples with 260 calcifications. If we look at predictive accuracy as a measure of goodness of the classifier for this case, the default accuracy would be 97.68% when every sample is labeled non-calcification. But, it is desirable for the classifier to predict most of the calcifications correctly. For outlier detection, the minority class of calcification is considered as outlier class and the non-calcification class as inliers.

3.3.2 Language

Empirical evaluation remains the most used approach for the algorithm assessment, although ML algorithms can be evaluated through empirical assessment or theory or both. The programming language R is used. R is an open source programming language and software environment for statistical computing and graphics that is 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.

Chapter 4

Analysis Modeling

4.1 Data Modeling

In analogy to a database-management system, we can view a stream processor as a kind of data-management system, the high-level organization of which is suggested in figure.

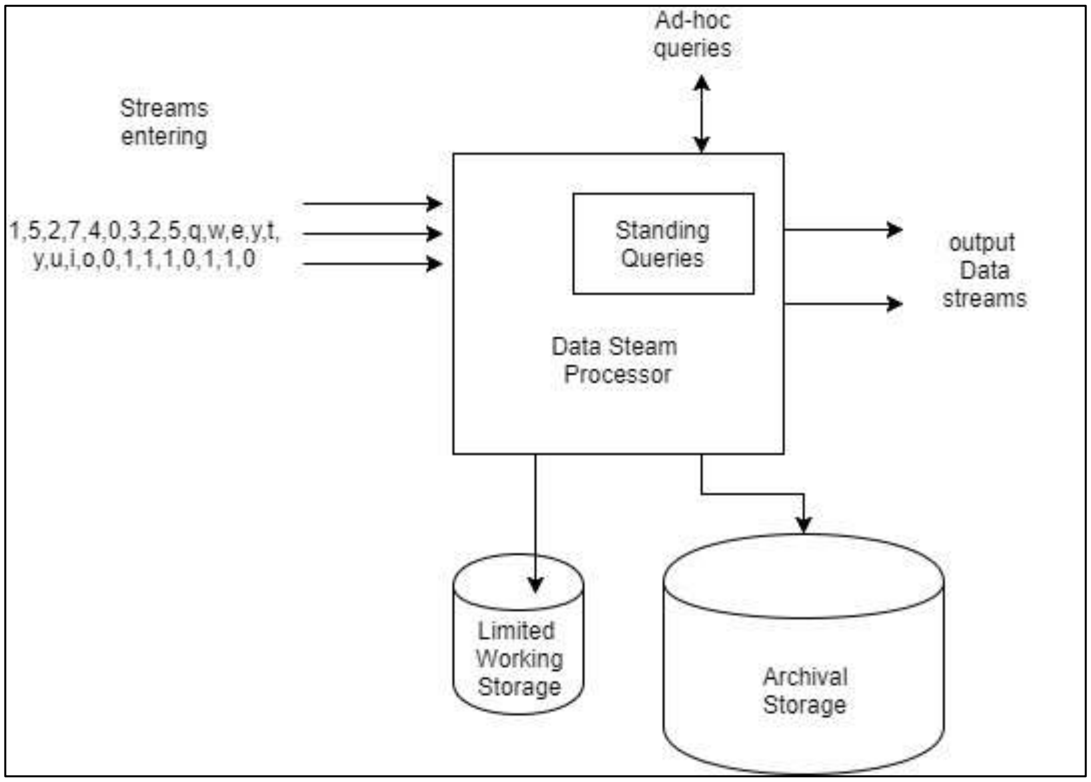


Fig 4.1: Data Model

- Any number of streams can enter the system. Each stream can provide elements at its own schedule; they need not have the same data rates or data types, and the time between elements of one stream need not be uniform.

- The fact that the rate at which the data is read from the disk is not under the control of the system distinguishes stream processing from the processing of data that goes on within a database-management system.
- Streams may be archived in a large archival store, but it is assumed that it is not possible to answer queries from the archival store. It could be examined only under special circumstances using time-consuming retrieval processes.
- There is also a working store, into which summaries or parts of streams may be placed, and which can be used for answering queries.
- The working store might be disk, or it might be main memory, depending on how fast we need to process queries. But either way, it is of sufficiently limited capacity that it cannot store all the data from all the streams.

4.2 Activity Diagram

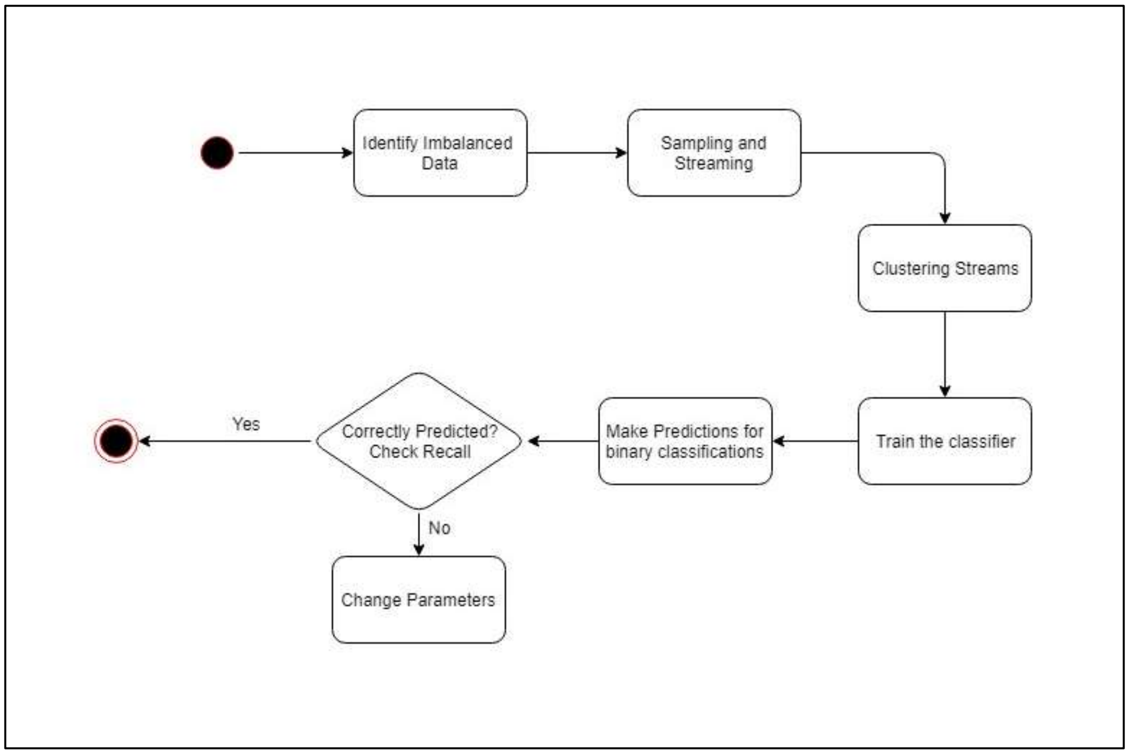


Fig 4.2: Activity Diagram

The activity diagram has various blocks which define the different blocks (or activities) which take place one after the other. Identify the training data on which we need to train the algorithm. The data is split into two batches as training data and testing data. The training data set is sampled to balance the majority and minority classes and this is done in the pre-processing stage. Clustering algorithms are applied to the training data sets to form the micro and macro clusters. These labels assigned by the clustering algorithms are given with the input data to the classifier. The performance of the classifier algorithms are evaluated using the recall measure values.

4.3 Functional Modeling

4.3.1 Overview of Process

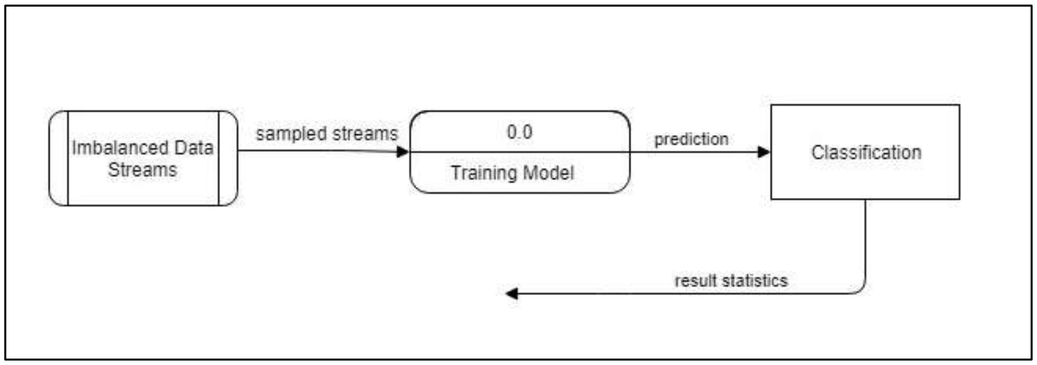


Fig 4.3: Overview of Process

The above diagram gives the fundamental context model of the entire system. Imbalanced Data Streams are processed first and given as input to the supervised algorithms. An Optimal Classifier Algorithm is chosen which predicts the results. These result statistics are evaluated which helps to further improve the classifier model.

4.3.2 Overall process

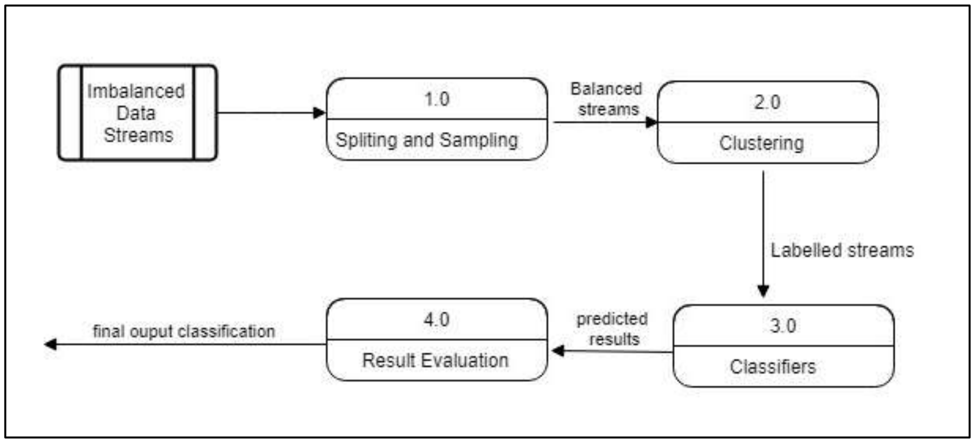
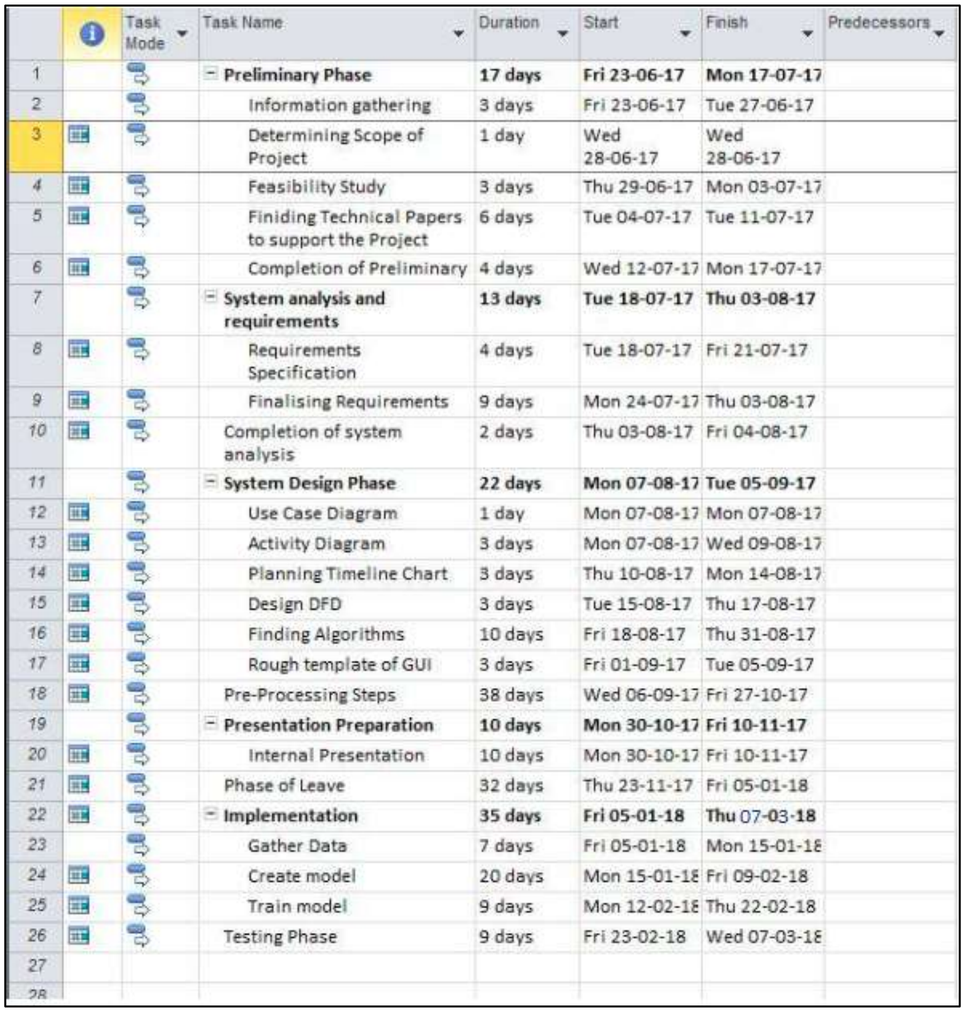


Fig 4.4: Overall Process

The overall process diagram gives us a little insight into the working of the system. Imbalanced Data Sets are fed into the pre-processor which uses packages in R to process this data to generate data streams. Pre-processing is done since the data sets are generally incomplete, noisy and inconsistent. Tasks to be performed in data preprocessing are Data cleaning, Data Sampling and Converting the data to streams. The algorithms are trained on these data streams, according to which the best algorithm is selected for prediction. The Classification predictor predicts the results of classification which are further evaluated. After evaluation, the characteristics of the classifier model can be changed and desired output classification of the data streams can be attained.

4.4 TimeLine Chart



	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		⇨	**Preliminary Phase**	**17 days**	**Fri 23-06-17**	**Mon 17-07-17**	
2		⇨	Information gathering	3 days	Fri 23-06-17	Tue 27-06-17	
3	▦	⇨	Determining Scope of Project	1 day	Wed 28-06-17	Wed 28-06-17	
4	▦	⇨	Feasibility Study	3 days	Thu 29-06-17	Mon 03-07-17	
5	▦	⇨	Finiding Technical Papers to support the Project	6 days	Tue 04-07-17	Tue 11-07-17	
6	▦	⇨	Completion of Preliminary	4 days	Wed 12-07-17	Mon 17-07-17	
7		⇨	**System analysis and requirements**	**13 days**	**Tue 18-07-17**	**Thu 03-08-17**	
8	▦	⇨	Requirements Specification	4 days	Tue 18-07-17	Fri 21-07-17	
9	▦	⇨	Finalising Requirements	9 days	Mon 24-07-17	Thu 03-08-17	
10	▦	⇨	Completion of system analysis	2 days	Thu 03-08-17	Fri 04-08-17	
11		⇨	**System Design Phase**	**22 days**	**Mon 07-08-17**	**Tue 05-09-17**	
12	▦	⇨	Use Case Diagram	1 day	Mon 07-08-17	Mon 07-08-17	
13	▦	⇨	Activity Diagram	3 days	Mon 07-08-17	Wed 09-08-17	
14	▦	⇨	Planning Timeline Chart	3 days	Thu 10-08-17	Mon 14-08-17	
15	▦	⇨	Design DFD	3 days	Tue 15-08-17	Thu 17-08-17	
16	▦	⇨	Finding Algorithms	10 days	Fri 18-08-17	Thu 31-08-17	
17	▦	⇨	Rough template of GUI	3 days	Fri 01-09-17	Tue 05-09-17	
18	▦	⇨	Pre-Processing Steps	38 days	Wed 06-09-17	Fri 27-10-17	
19		⇨	**Presentation Preparation**	**10 days**	**Mon 30-10-17**	**Fri 10-11-17**	
20	▦	⇨	Internal Presentation	10 days	Mon 30-10-17	Fri 10-11-17	
21	▦	⇨	Phase of Leave	32 days	Thu 23-11-17	Fri 05-01-18	
22	▦	⇨	**Implementation**	**35 days**	**Fri 05-01-18**	**Thu 07-03-18**	
23		⇨	Gather Data	7 days	Fri 05-01-18	Mon 15-01-18	
24	▦	⇨	Create model	20 days	Mon 15-01-18	Fri 09-02-18	
25	▦	⇨	Train model	9 days	Mon 12-02-18	Thu 22-02-18	
26	▦	⇨	Testing Phase	9 days	Fri 23-02-18	Wed 07-03-18	
27							
28							

Fig 4.5: Timeline Chart Diagram

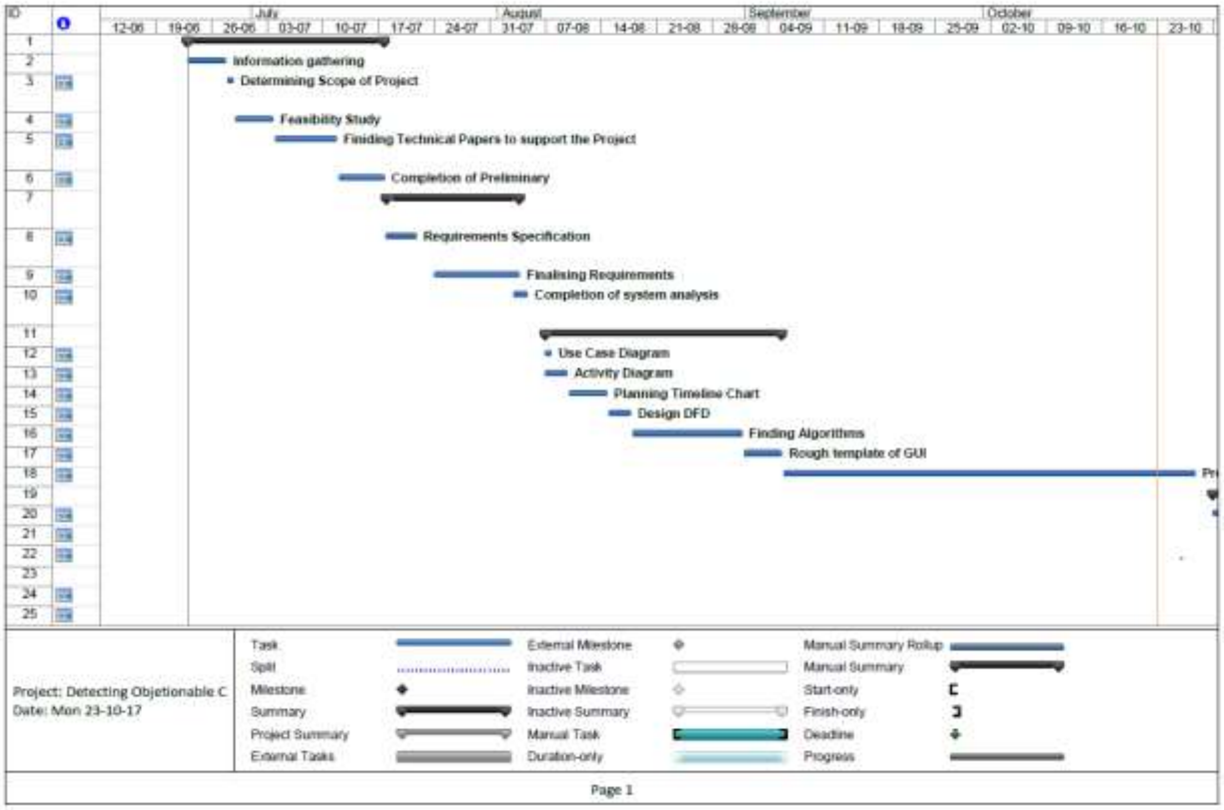


Fig. 4.6: Gantt chart

Chapter 5

Design

5.1 Architectural Design

The data sets are collected from various sources and they are given to the pre-processing unit of the system. In the pre-processing unit, splitting of data is performed to create training and test sets. These sets undergo sampling to handle the imbalanced data in which synthetic samples of the minority classes are created to increase the number of data points.

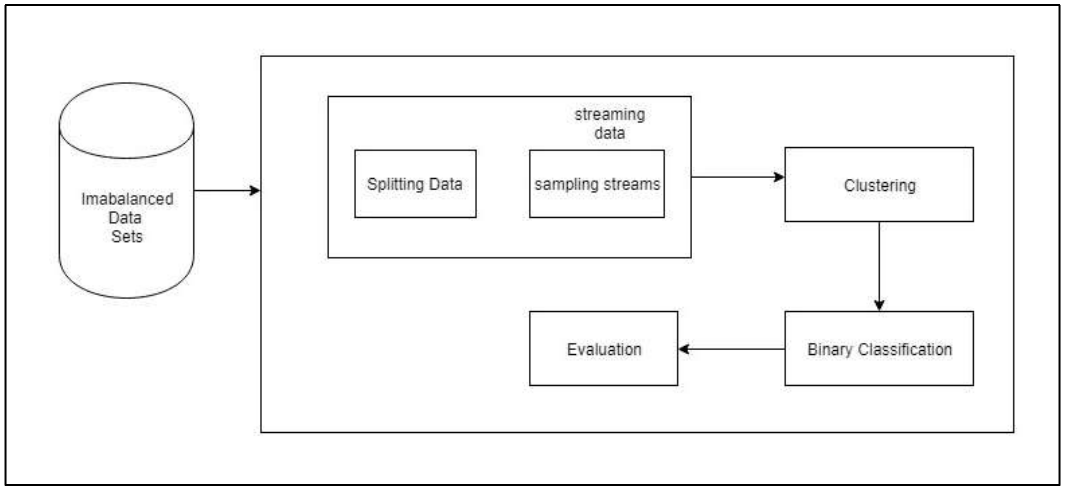


Fig 5.1: Architectural Design

These sampled data sets are then converted to data streams using various packages in R. Various algorithms are then performed on these data streams and thus the classifier is trained. The prediction evaluator unit is then used to check the performance of the classifier based on the recall performance metric. If the predictions are not accurate and balanced, necessary changes and updates are made to the classifier to improve the performance.

Chapter 6

Implementation

6.1 Algorithms/Methods Used

6.1.1 SMOTE

Data-level methods to handle imbalance data modify the collection of examples to balance distributions and/or remove difficult samples. Sampling methods include undersampling and oversampling methods. Undersampling balance the data by removing samples from majority class and oversampling balance the data by create copies of the existing samples or adding more samples to the minority class. However, under sampling may cause loss of useful information by removing significant patterns and over sampling may cause over fitting and may introduce additional computational task. To tackle the over fitting problem, Chawla et al[5] proposed a synthetic minority oversampling technique (SMOTE) by generating a synthetic examples rather than replacement with replication. This technique identifies more specific regions in the feature space for the minority class. The synthetic minority oversampling technique (SMOTE) is an important approach by oversampling the positive class or the minority class.

6.1.2 Clustering Algorithms

6.1.2.1 Sliding window + k-means

Data stream clustering with sliding windows generates clusters for every window movement. In the sliding windows model, data elements arrive continually, and only the most recent elements are considered. In this context, at the period t - called window t, we consider N recent elements that contain some elements of window $t - 1$. In most data stream applications, the most recent tuples are considered to be more decisive and influential. Clustering algorithms are blocking since they require having all the data points to compute the clusters. Therefore, windowing is needed for data stream clustering. A sliding window is specified by defining its range and stride.[11] The range R of a sliding window specifies the length of the window while the stride S specifies the portion of the range that is evicted from the window when the window moves forward.

We have used the clustering algorithms in the "stream" and "streamMOA" package in R.

For our implementation, we use a two-stage clustering algorithm in which first the micro-clusters are generated by the input data points. Then the micro-clusters are re-clustered to generate the desired number of macro-clusters. The DSC_TwoStage function of the "stream" package is used which can include two different algorithms for micro and macro clustering. We have used the DSC_Window function to implement the sliding window clustering algorithm for micro-clusters and the DSC_Kmeans function to implement the k-means clustering algorithm for macro-clusters.

DSC_TwoStage: micro-DSC_Window, macro-DSC_Kmeans

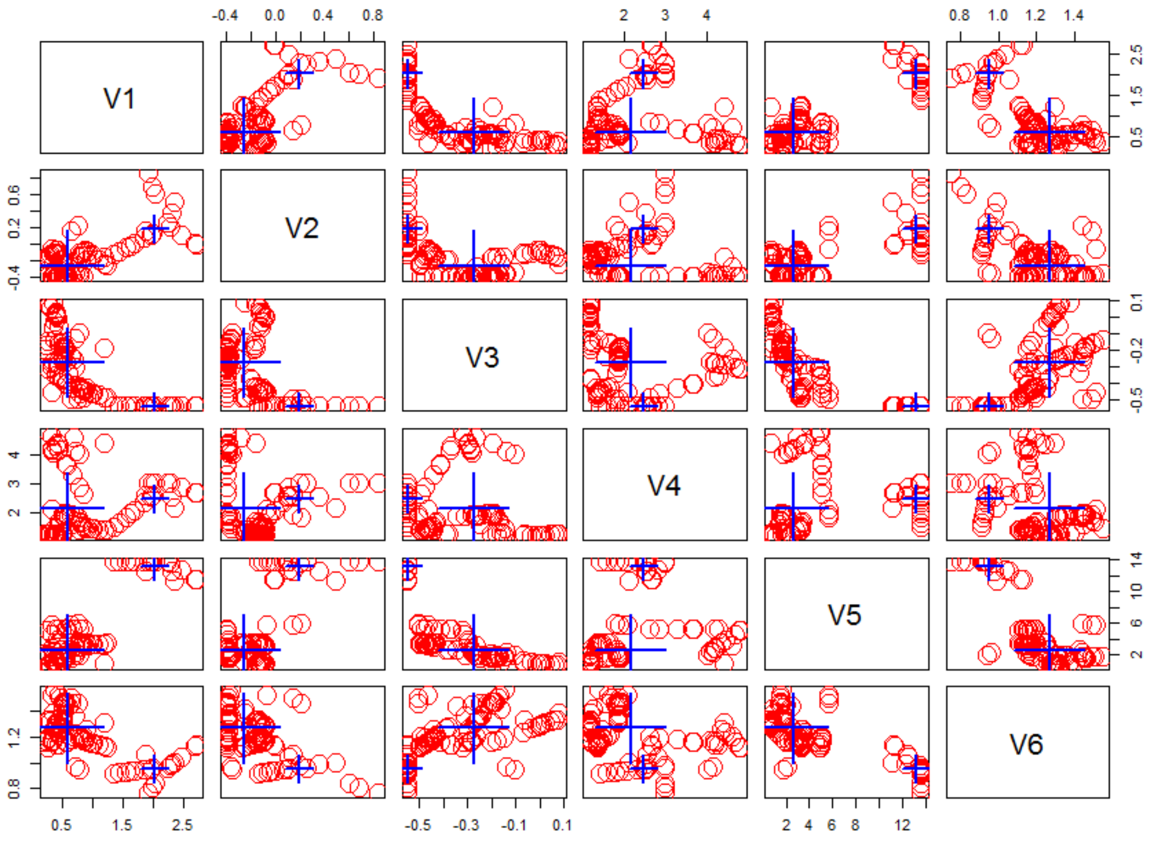


Fig. 6.1: Sliding window + k-means algorithm plot

6.1.2.2 Clustream + k-means

CluStream is an algorithm for the clustering of evolving data streams based on user-specified, on-line clustering queries. It divides the clustering process into on-line and off-line components. The on-line component computes and stores summary statistics about the data stream using microclusters, and perform incremental on-line computation and maintenance of the microclusters. The off-line component does macroclustering and maintenance and answers various user questions using the stored summary statistics.

The on-line microcluster processing is divided into two phases: (1) statistical data collection and (2) updating of microclusters. In the first phase, a total of q microclusters are maintained, where q is usually significantly larger than the number of natural clusters. Each new data point is added to either an existing cluster or a new one. To decide whether a new cluster is required, a maximum boundary for each cluster is defined. If the new data point falls within the boundary, it is added to the cluster; otherwise, it is the first data point in a new cluster.

The off-line component can perform user-directed macroclustering or cluster evolution analysis. Macroclustering allows a user to explore the stream clusters over different horizons. CluStream offers rich functionality to the user because it registers the essential historical information with respect to cluster evolution. The titled time frame along with the microclustering structure allow for better accuracy and efficiency on real data. It also maintains scalability in terms of stream size, dimensionality, and the number of clusters.

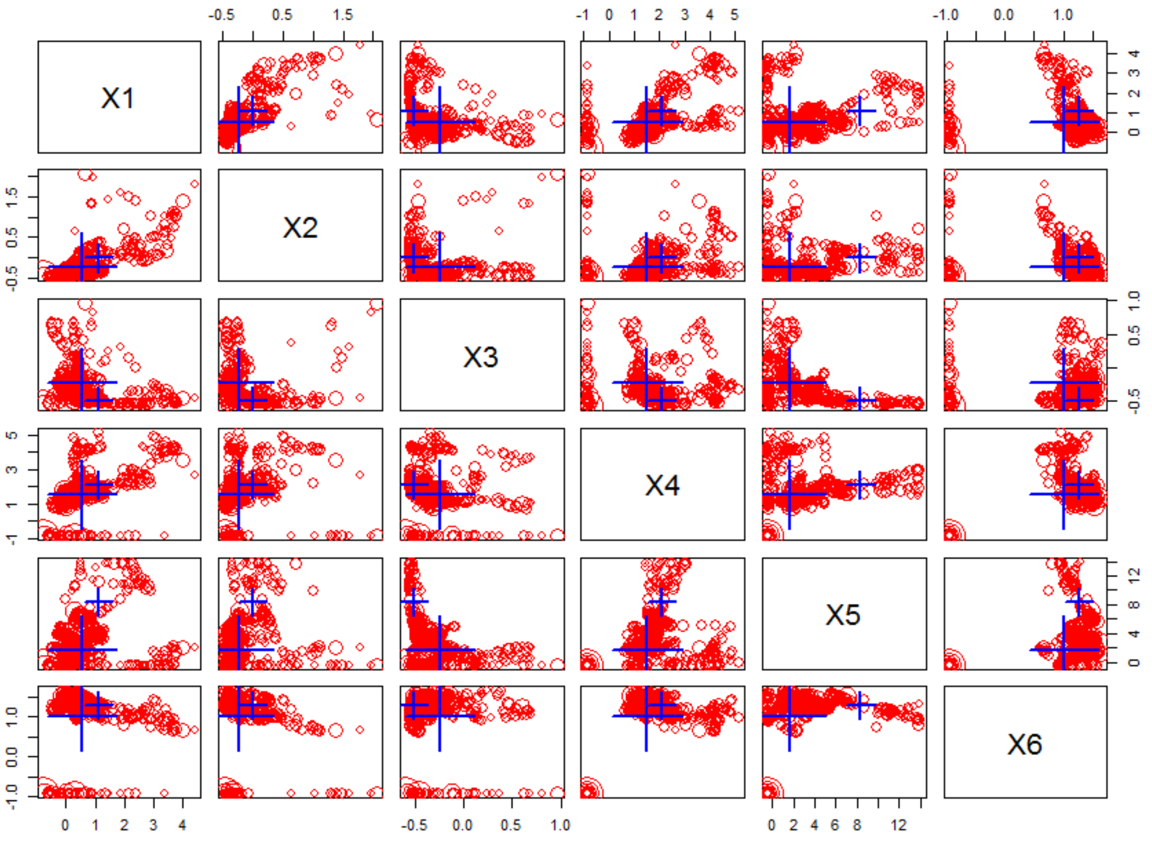


Fig. 6.2: Clustream + k-means algorithm plot

6.1.3 Classification Algorithms

6.1.3.1 Naïve Bayes

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Usage:

NaiveBayes(control = NULL, ...)

NaiveBayesMultinomial(control = NULL, ...)

Value:

An object of class MOA_classifier which sets up an untrained MOA model, which can be trained using trainMOA

6.1.3.2 OzaBoost

For the boosting method, Oza and Russell note that the weighting procedure of AdaBoost actually divides the total example weight into two halves – half of the weight is assigned to the correctly classified examples, and the other half goes to the misclassified examples. They use the Poisson distribution for deciding the random probability that an example is used for training, only this time the parameter changes according to the boosting weight of the example as it is passed through each model in sequence.

6.2 Working of the project

1. Libraries Used: caret, e1071, stream, streamMOA, DMwR, RMOA

2. Preprocessing of input data:
```
Inp <- read.csv("Input.txt",col.names = c("V1","V2", "V3", "V4", "V5", "V6","V7"),
sep=",")
index <- createDataPartition(Inp$V7, p=0.5, list=FALSE)
write.table(Inp[index,],"TrainF.txt",sep=",",row.names=FALSE,col.names = TRUE)
write.table(Inp[-index,],"TestF.txt",sep=",",row.names=FALSE,col.names = TRUE)
```

3. Converting the data set to data streams:
```
train_strm <- DSD_ReadCSV("TrainF..txt",k=2,take=c(1:6),sep = ",",header=TRUE)
```

4. Sliding Window clustering algorithm using the DSC_TwoStage of "streamMOA" package:
```
Windo <- DSC_TwoStage(micro=DSC_Window(100),macro=DSC_Kmeans(k=2))
update(Windo,train_strm,5000)
plot(Windo, assignment= TRUE, type="both")
```

5. Read the Class Labels that come with Delay:
```
classlabel_stream<- DSD_ReadCSV("SampleClass.txt",k=2,sep = ",",header=TRUE)
```

6. Write data from clustering to provide as input to classifier:
```
write_stream(train_strm, file="InputforClassifier.txt", n=5000,class=FALSE, append =
FALSE, sep=",", header=TRUE, row.names=FALSE)
```

7. Processing the input data to be given to the classifier:
```
sea_data <- read.csv("InputforClassifier.txt", sep= ",")
sea_data<-data.frame(sea_data,a)
sea_datastream <- datastream_dataframe(data=sea_data)
trainset <- sea_datastream$get_points(sea_datastream, n = 5592, outofpoints = c("stop",
"warn", "ignore"))
trainset <- datastream_dataframe(data=trainset)
```

8. Classifier Model-Naive Bayes:
```
ctrl<-MOAoptions(model="NaiveBayes")
mymodel<-NaiveBayes(control=ctrl)
naive_model <- trainMOA(model = mymodel,a ~ ., data = trainset)
sea_test_data <- read.csv("TestF.txt", sep= ",")
sea_test_datastream <- datastream_dataframe(data=sea_test_data)
testset <- sea_test_datastream$get_points(sea_test_datastream, n = 4473, outofpoints =
c("stop", "warn", "ignore"))
scores <- predict(naive_model, newdata=testset, type="response")
Confuse<-confusionMatrix(data=scores,testset[,7], positive="1")
```

9. Ensemble model – OzaBoost:
```
emodel1<-OzaBoost(baseLearner="trees.HoeffdingTree",ensembleSize=30)
ozaboost_model <- trainMOA(model = emodel1,a ~ ., data = trainset)
testset2 <- sea_test_datastream$get_points(sea_test_datastream, n = 5591, outofpoints =
c("stop", "warn", "ignore"))
scores2 <- predict(ozaboost_model, newdata=testset2, type="response")
Confuse_oza<-confusionMatrix(data=as.numeric(scores2>0.5),testset2[,7], positive="1")
```

Chapter 7

Testing

7.1 Experimental Setup

7.1.1 DBSTREAM

This algorithm[12] is supposed to be the first micro-cluster-based online clustering component that explicitly captures the density between micro-clusters via a shared density graph. The density information in this graph is then exploited for reclustering based on actual density between adjacent micro-clusters.

7.1.2 DenStream

The “dense” micro-cluster (named core-micro-cluster)[13] is introduced to summarize the clusters with arbitrary shape, while the potential core-micro-cluster and outlier micro-cluster structures are proposed to maintain and distinguish the potential clusters and outliers. A novel pruning strategy is designed based on these concepts, which guarantees the precision of the weights of the micro-clusters with limited memory. The method can discover clusters of arbitrary shape in data streams, and it is insensitive to noise. This algorithm can handle arbitrary shaped clusters, but due to the numerous time vector calculations, it has high time complexity.

7.1.3 DStream

The D-Stream algorithm [15] has also been proposed, which is capable of making automatic and dynamic adjustments to the data clusters without user specification with regard to the target time horizon and number of clusters. This algorithm creates separated grids to map new incoming data. A decay factor is used with the density of each data point in order to determine which data are recent and which are less important (old). The D-Stream algorithm is incapable of processing very high-dimensional data; however, the DenStream algorithm has no difficulty in processing such data.

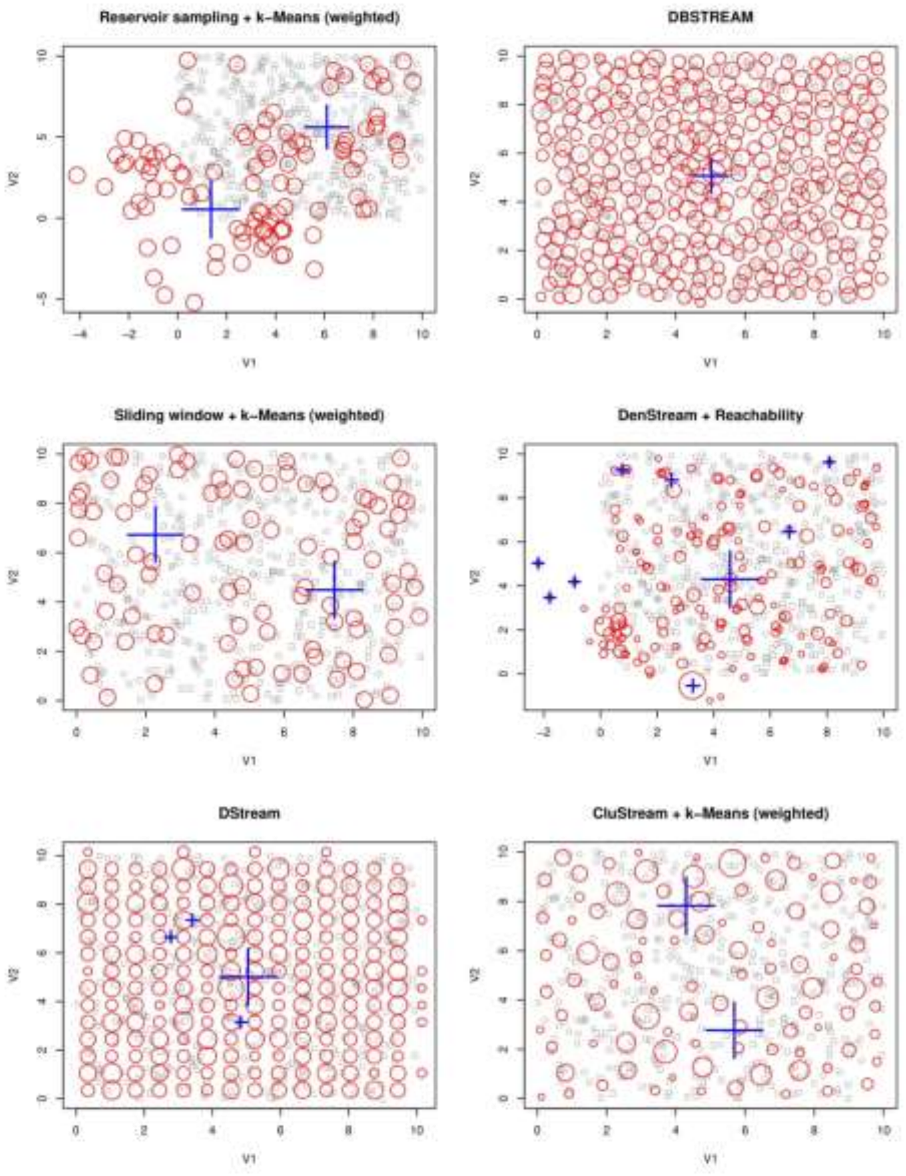


Fig. 7.1: Comparison of Clustering algorithms

7.1.4 Reservoir Sampling + k-means

A reservoir-sampling algorithm[14] is a simple, unbiased random sampling algorithm for drawing a sample of size n without replacement from a population of size N. Given a data set containing n objects, k-means partitions these objects into k groups. Each group is represented by the centroid of the cluster. Once cluster representatives are selected, data objects are assigned to the nearest centers. After the initial step of dividing the data space into bins of equal size, the information of the first n groups are put into the n reservoirs residing in main memory. The collected information includes the number of points in each group and the id of the group. The algorithm performs a single scan on a data set in a random manner. The density biasing is achieved through the consideration of two consecutive data groups. If the original data is assumed to be much varied in cluster sizes, density-biased sampling is an appropriate method to preserve the density. The inherent advantage of efficient memory usage in the reservoir scheme is adopted and extended with the additional capability of dealing with data that are much different in density distribution.

7.2 Experimental Results

7.2.1 Classification using true labels

Dataset used: Mammography

```
Naive Bayes:
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5239   29
         1  235   88

               Accuracy : 0.9528
                 95% CI : (0.9469, 0.9582)
    No Information Rate : 0.9791
    P-Value [Acc > NIR] : 1

                  Kappa : 0.381
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.75214
            Specificity : 0.95707
         Pos Pred Value : 0.27245
         Neg Pred Value : 0.99450
             Prevalence : 0.02093
         Detection Rate : 0.01574
   Detection Prevalence : 0.05777
      Balanced Accuracy : 0.85460

       'Positive' Class : 1
```

```
Ozaboost:
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5431   55
         1   43   61

               Accuracy : 0.9825
                 95% CI : (0.9787, 0.9857)
    No Information Rate : 0.9792
    P-Value [Acc > NIR] : 0.04749

                  Kappa : 0.5456
 Mcnemar's Test P-Value : 0.26650

            Sensitivity : 0.52586
            Specificity : 0.99214
         Pos Pred Value : 0.58654
         Neg Pred Value : 0.98997
             Prevalence : 0.02075
         Detection Rate : 0.01091
   Detection Prevalence : 0.01860
      Balanced Accuracy : 0.75900

       'Positive' Class : 1
```

7.2.2 Clustream + k-means Clustering and Classification with SMOTE

Dataset used: Mammography

```
Naive Bayes:
Confusion Matrix and Statistics

      Reference
```

```
Ozaboost:
Confusion Matrix and Statistics

      Reference
```

```
Prediction    0    1
         0 2059 1355
         1    8  842

              Accuracy : 0.6803
                95% CI : (0.6661, 0.6943)
    No Information Rate : 0.5152
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.3722
 Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.3832
           Specificity : 0.9961
        Pos Pred Value : 0.9906
        Neg Pred Value : 0.6031
            Prevalence : 0.5152
        Detection Rate : 0.1975
   Detection Prevalence : 0.1993
      Balanced Accuracy : 0.6897

       'Positive' Class : 1
```

```
Prediction    0    1
         0 2065 1743
         1    2  454

              Accuracy : 0.5908
                95% CI : (0.5758, 0.6056)
    No Information Rate : 0.5152
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.2007
 Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.2066
           Specificity : 0.9990
        Pos Pred Value : 0.9956
        Neg Pred Value : 0.5423
            Prevalence : 0.5152
        Detection Rate : 0.1065
   Detection Prevalence : 0.1069
      Balanced Accuracy : 0.6028

       'Positive' Class : 1
```

7.2.3 Sliding window(horizon size = 100) + k-means Clustering and Classification with SMOTE

Dataset used: Mammography

```
Naive Bayes:
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1964  589
         1  103 1608

              Accuracy : 0.8377
                95% CI : (0.8263, 0.8487)
    No Information Rate : 0.5152
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.6774
 Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.7319
           Specificity : 0.9502
        Pos Pred Value : 0.9398
        Neg Pred Value : 0.7693
            Prevalence : 0.5152
        Detection Rate : 0.3771
   Detection Prevalence : 0.4013
      Balanced Accuracy : 0.8410
       'Positive' Class : 1
```

```
Ozaboost:
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2037 1072
         1   30 1125

              Accuracy : 0.7416
                95% CI : (0.7281, 0.7546)
    No Information Rate : 0.5152
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.4902
 Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.5121
           Specificity : 0.9855
        Pos Pred Value : 0.9740
        Neg Pred Value : 0.6552
            Prevalence : 0.5152
        Detection Rate : 0.2638
   Detection Prevalence : 0.2709
      Balanced Accuracy : 0.7488
       'Positive' Class : 1
```

7.2.4 Sliding window(horizon size = 10) + k-means Clustering and Classification with SMOTE

Dataset used: Mammography

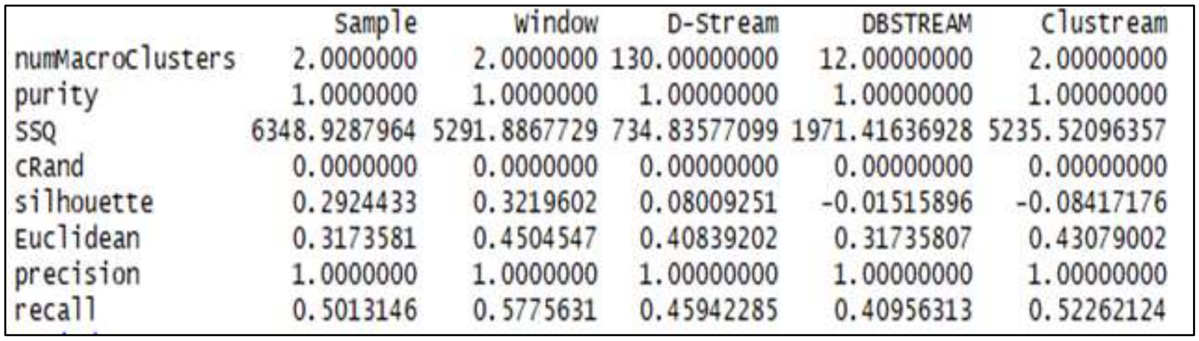
The performance of the Naïve Bayes classifier was not affected at all whereas the performance of Ensemble classifier improved.

```
OzaBoost
Confusion Matrix and Statistics

          Reference
Prediction       0         1
          0      1897  830
          1      170 1367

             Accuracy : 0.7655
               95% CI : (0.7525, 0.7781)
          No Information Rate : 0.5152
    P-Value [Acc > NIR] : < 2.2e-16

                Kappa : 0.5349
 Mcnemar's Test P-Value : < 2.2e-16

           Sensitivity : 0.6222
           Specificity : 0.9178
        Pos Pred Value : 0.8894
        Neg Pred Value : 0.6956
            Prevalence : 0.5152
        Detection Rate : 0.3206
   Detection Prevalence : 0.3605
      Balanced Accuracy : 0.7700

       'Positive' Class : 1
```

Chapter 8

Results and Discussions

- We have implemented various clustering algorithm on the datasets from which the best one is chosen based on the evaluation measures. The evaluation measures used are purity, SSQ, cRand, silhouette, Euclidean distance, precision and recall. The Sliding Window + k-means clustering algorithm has the best evaluation measures and recall value amongst them.



```
                       Sample        Window      D-Stream       DBSTREAM      Clustream
numMacroClusters    2.0000000     2.0000000  130.00000000    12.00000000     2.00000000
purity              1.0000000     1.0000000    1.00000000     1.00000000     1.00000000
SSQ              6348.9287964  5291.8867729  734.83577099  1971.41636928  5235.52096357
cRand               0.0000000     0.0000000    0.00000000     0.00000000     0.00000000
silhouette          0.2924433     0.3219602    0.08009251    -0.01515896    -0.08417176
Euclidean           0.3173581     0.4504547    0.40839202     0.31735807     0.43079002
precision           1.0000000     1.0000000    1.00000000     1.00000000     1.00000000
recall              0.5013146     0.5775631    0.45942285     0.40956313     0.52262124
```

Fig. 8.1: Evaluation measures of Clustering Algorithms on Mammography Data set

- The proposed model mainly focuses on improving the sensitivity(recall) so that maximum number of minority instances are correctly classified. According to the Confusion Matrix and Statistics, Sliding Window + k-means Clustering Algorithm along with Naive bayes classifier performs the best providing sensitivity value as 0.73. When true labels were used for Binary classification, the sensitivity obtained is 0.75. Hence, the two results obtained are very similar.
- In Sliding Window + k - means Clustering Algorithm, horizon determines the number of stream points taken in a window. The Ensemble Ozaboost classifier with base learner as Naive Bayes improves its performance when the horizon size of the sliding window is reduced to 10. But, the performance of Naive bayes classifier remains almost unchanged when size of the horizon is changed. The number of streaming points in a batch taken can sometimes play an important role as it can change the imbalance ratio of the data streams.

Chapter 9

Conclusion

In the field of machine learning and data mining, detecting events is a prediction problem, or, typically, a data classification problem. Rare events are difficult to detect because of their infrequency and casualness; however, misclassifying rare events can result in heavy costs. This was the main motivation to select this topic. While learning about the various existing algorithms used to handle imbalanced data streams, various problems were found unresolved.

The three insights of the problem considered are: class imbalance, data streams and absence of labels. The classifier deals with imbalanced data streams and classifies the data accurately. The ensemble classifier algorithm is used to handle the imbalance data issue. Its performance is judged on the performance metrics like Sensitivity and Confusion matrix. This implementation of the algorithm of the binary classifier would not only deal with time- critical problems like banking transactions but also health-critical problems like medical diagnosis of cancer malignancy.

This solution is useful to predict rare cases and events not in a particular field but all the various domains that face similar problems of imbalanced data classification in data streams that are not only enormous but also available only once.

Literature Cited

[1] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, Gong Bing, “Learning from class-imbalanced data: Review of methods and applications”, International Journal of Expert systems with applications, Elsevier, Dec 2016.

[2] B. Krawczyk, “Learning from imbalanced data: open challenges and future directions”, Progress in Artificial Intelligence, vol. 5, no. 4, pp. 221–232, 2016.

[3] Abraham Jacob Frandsen, "Machine learning for disease prediction"

[4] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In International Conference on Machine Learning, pp. 49-56, Montreal, Canada, June 2009

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, et al., “Smote: synthetic minority over-sampling technique,” Journal of Artificial Intelligence Research, Vol. 16, pp. 321–357, 2002.

[6] S. Chen and H. He, “Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach,” Evolving Systems, Vol. 2, 2011.

[7] H. He and E. Garcia, “Learning from imbalanced data,” Knowledge and Data Engineering, IEEE Transactions on, Vol. 21, no. 9, 2009

[8] JuiHsi Fu and SingLing Lee, “Certainity-Enhanced Active Learning for Improving Imbalanced Data Classification”

[9] Using Random Forest to Learn Imbalanced Data Chao Chen, Department of Statistics, UC Berkeley Andy Liaw, Biometrics Research, Merck Research Labs Leo Breiman.

[10]L. Tu and Y. Chen, "Stream data clustering based on grid density and attraction," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 3, p. 12, 2009.

[11] Partition-Based Clustering with Sliding Windows for Data Streams

[12] Clustering Algorithms for Data Stream Karishma Nadhe1, Prof. P. M. Chawan

[13] Density-Based Clustering over an Evolving Data Stream with Noise Feng Cao ∗ Martin Ester Weining Qian Aoying Zhou

[14] Density-Biased Clustering Based On Reservoir Sampling Kittisak Kerdprasop, Nittaya Kerdprasop and Pairote Sattayatham

[15] Data Stream Clustering Algorithms: A Review Maryam Mousavi1 , Azuraliza Abu Bakar1 , and Mohammadmahdi Vakilian.

[16] http://odds.cs.stonybrook.edu/mammography-dataset/

Acknowledgements

We would like to express our sincere gratitude to our guide, Prof. Kiran Bhowmick, for her guidance, encouragement and gracious support throughout the course of our work, for her expertise in the field that motivated us to work in this area and for her faith in us in every stage of research.

We are grateful to Dr. N. M. Shekokar, Head of the Department of Computer Engineering, for letting us use the department resources, labs, and books.

We are highly indebted to our Principal, Dr. Hari Vasudevan, for availing us with proper and resourceful journals like IEEE and other online material that helped us finish our project with ease and perfection.

We would also like to thank all our fellow students and the staff of the Department of Computer Engineering for their help in the process leading to the conceptualization of this project.

Megha Jakhotia

Yash Kapadia

Nikita Luthra