ABSTRACT

Classification is one of the most widely studied problems in the data mining and machine learning communities. Applications involving streaming data are ubiquitous. Typical examples include computer network traffic, phone conversations, ATM transactions, and so on. In traditional classification procedures, data is assumed to be stationary. However, in data streams, the properties of the dataset or the output variable tend to drift over time due to changes in the environment and hence it becomes difficult to classify data in a non stationary environment.

Significant research has been done and various methods have been proposed for both the problems individually i.e. classifying imbalanced datasets and to detect and adapt to concept drift. But relatively less work has been done on both the problems combined.

Therefore we propose a solution for classifying data in a non stationary environment (data streams) where we have an imbalanced dataset and the underlying concept that projects the attributes to the class labels is changing continuously viz concept drift.

A hierarchical architecture is proposed consisting of 2 layers. The first layer takes a guess or generates a hypothesis on whether a drift has occurred in the streaming data or not. This hypothesis is then passed on to the second layer along with the data stream. The second layer then tries and validates the hypothesis by performing additional computation. Now the output generated from this layer is sent as a feedback to the previous layer as well as used as the final output for class classification.

1. INTRODUCTION

1.1. Description

Since recent years the amount of data generated is increasing continuously and hence we need to collect this data in order to process it in an efficient manner. Most of the data being collected has a high volume and comes in at a very fast rate and might probably be of infinite length. Such data is referred to as data streams. A data stream is a sequence of relational tuples that is continuously generated as time goes on. Due to limited storage space data streams can be only read once in most of the cases.

Data Stream Mining is the process of extracting knowledge structures from continuous, rapid data records. In many data stream mining applications, the goal is to predict the class or value of new instances in the data stream given some knowledge about the class membership or values of previous instances in the data stream [1]. In most of the data mining applications of classifying datasets the data is assumed to be static but in non-stationary environments the underlying distribution of the data might change. This is known as concept drift. There is a need for algorithms which can detect this drift in concept and adapt to such changes.

Class imbalance is a serious problem and it occurs when there are far less instances of the minority class compared to other classes. In such situations, the classifier fails to classify the minority class accurately and the cost of misclassification is quite high in data stream applications. Class imbalance is a very common problem and it can be introduced due to the nature of the applications. As these challenges interfere with concept drifts in the context of data streams, it becomes an even more severe and compound problem [2]. We propose to design an algorithm for classifying skewed data streams which takes into account the challenges posed by imbalanced classes as well as adapts to concept drift.

1.2. Problem Formulation

Applications with data streams are considered an important domain of research. Concept drift in data streaming applications refers to the problem where the underlying distribution of the data which is used to train the classifier changes over time. In most of the real time data mining applications a large amount of skewed data is generated which means that there are far more instance of one class compared to the other class. Such a skewed distribution is known as imbalanced dataset.

Class Imbalance problems are seen in various applications such as medical diagnosis where we need to predict a rare but important disease compared to other diseases. Since this disease is rare we might have fewer instances of this class and hence it is known as minority class whereas we will mostly have a large number of instances of the other common diseases and hence they are known as the major classes. In such situations the classifier is biased towards the major class and hence it misclassifies the minor class[4]. The cost of such misclassification is very high. Hence a solution is required which can correctly classify the imbalanced data samples even in the presence of concept drift.

1.3. Motivation

There are several techniques[6] available for the classification of data streams such as Hoeffding tree algorithm, Streaming Random Forest, Concept-adapting Very Fast Decision Tree (CVFDT). Hoeffding tree algorithm is based on decision tree learning and it uses the Hoeffding bound for construction and analysis of the decision tree. Random forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. CVFDT continuously monitors the quality of new data and adjusts those that are no longer correct as compared to other existing systems. CVFDT handles concept drift by creating a new tree once a change is detected. The old subtrees are replaced by the new more accurate subtrees.

The problem with Hoeffding tree algorithm is that it assumes that the underlying distribution of the data does not change over time and a node once created cannot be changed. Hence it cannot be used to handle concept drift. Random Forests scale poorly with the size of the dataset which makes them impractical in streaming applications. In CVFDT if the splitting criteria for the root node is incorrectly chosen then the accuracy

will be very low and the entire method loses its meaning. This will lead to high misclassification errors and the cost for such misclassification is huge. Thus even though various algorithms have been proposed for concept drift and class imbalance they have various drawbacks. In non stationary environments the complexity of such classification problems increases and learning in such environments has not been explored thoroughly. In our solution we propose to classify datasets in a non-static environment experiencing class imbalance in the data, where the classifier will learn from a concept that is changing in time.

1.4. Proposed Solution

To solve the concept drift problem in presence of class imbalance we propose a combination of a hierarchical testing architecture used along with a neural network.

We use an easy to implement drift detection algorithm like LFR (Linear Four Rates) which is used initially to take a guess if drift has occurred or not. It uses the $ commonly used parameters - True Positive rate (TPR), True Negative Rate(TNR), False Positive rate(FPR), False Negative rate(FNR). The principle used here is that under a static context (the probability of distribution remain more or less the same) the values of the 4 parameters does not change appreciably.

A threshold value is set here and if the for the given streaming data the deviation in these 4 values crosses the threshold it indicates that a drift has occurred. This algorithm has a high precision rate of detecting concept drift.

In case the LFR algorithm detects any drift, the streaming data along with some digest information is sent to the Hypothesis testing framework. In this framework we use a ensemble classifier method to validate the hypothesis generated by the LFR algorithm. We use weight updating ensemble classifiers. All the component classifiers will be assigned weights reflecting their performance on the previous few examples of data. The better performing classifiers will be awarded the greatest weights and the classification output generated by these will be given the greater weightage.

We also propose using a feedback signal akin to a neural network setting that updates the LFR algorithm in the first stage depending on whether it made the right hypothesis or not. We also plan to remove the second ensemble classification method once the hypothesis

algorithm gains a sufficiently good accuracy in detecting concept drifts**.** Doing so will increase the speed of the drift detection technique as well as reduce the overall complexity of computation.

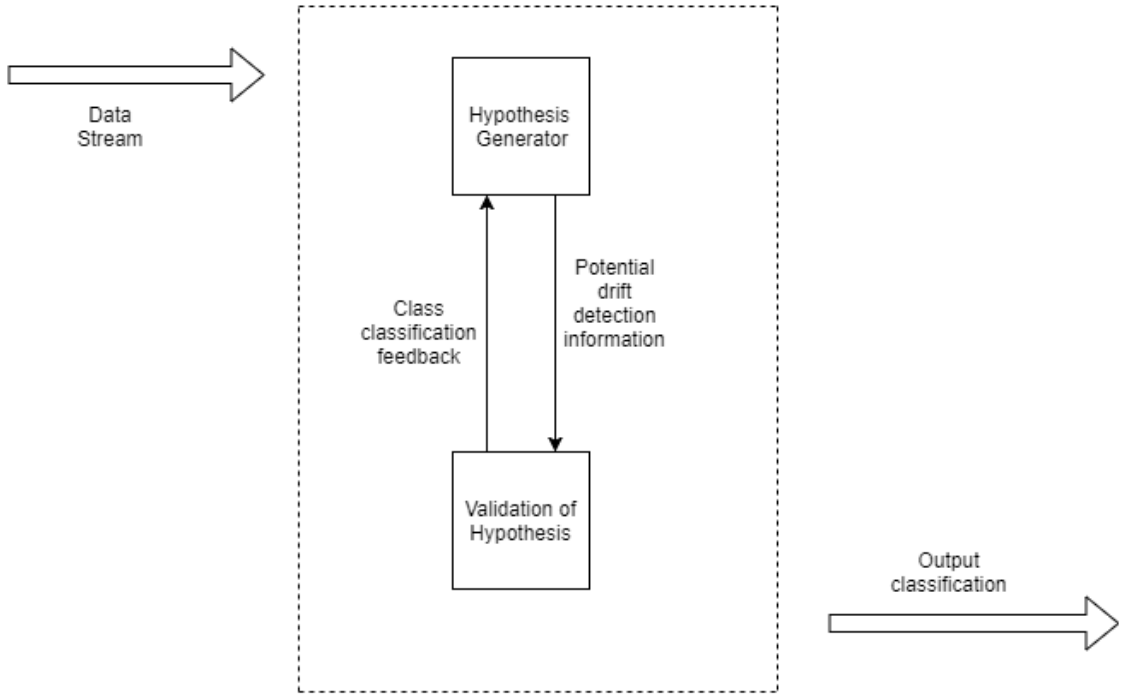


Fig 1.4.1 Architecture of the Proposed System

Objective of the project

- To develop a solution to detect concept drift in class imbalanced data streams.
- To adapt the classifier to concept drift with imbalanced datasets with very fast speed.
- To classify the imbalanced data in the presence of concept drift in a non stationary environment with high accuracy.

1.5. Scope of the project

- Our solution aims at handling concept drift in class imbalanced data streams. We need to build classifiers that can adapt to changes caused by concept drift.
- We will be using data mining techniques for classifying the data streams and later on we will use machine learning techniques to make the classifier adapt to changes in the underlying distribution of data.
- Since we will be working with streaming data, the classifier will be able to see the data only once so we will not be able to process all the data samples. Instead we will be working either with batches of data or data summaries. Hence our system will not be completely online as some processing has to be done offline.

2. REVIEW OF LITERATURE

2.1. Introduction

Most of the data mining applications with streaming data are affected by class imbalance and concept drift. This hinders the predictive performance of the classifiers to a great extent and the severity of the problem increases when they occur simultaneously. With the wide application of machine learning algorithms to the real world, class imbalance and concept drift have become crucial learning issues. Applications in various domains such as risk management, anomaly detection, fraud detection, software engineering, social media mining, and recommender systems are affected by both class imbalance and concept drift. (2)

2.2. Class Imbalance Problem

The class Imbalance problem refers to an issue in classification where the multiple classes of data in the training set are not represented equally. The minority class occur rarely but are very important (like the fraud class in a fraud detection system). This skewed data often leads to misclassification [1].

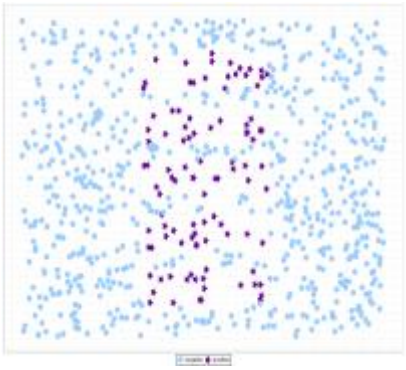


Fig 2.2.1 Representation of imbalanced data

For example there maybe a binary classification problem with 100 examples and 90 of them are labelled with Class-A and the remaining 10 examples are labelled with Class-B. This is an imbalanced classification problem and the ratio of Class-A to Class-B instances is 90:10 or more concisely 9:1. The figure shows a scatterplot for the above example. The Class Imbalance problem can be either two-class classification problems or multi-class

classification problems. In our solution we will be focussing on Binary Class (Two Class) Imbalance classification problem.

2.3. Concept Drift

The term concept drift refers to change in the underlying distribution of the data. In other words the properties that lead to change in the target variable (actual output class) changes over time. There are four different types of concept drift based on fundamental properties of the data [5]. They are abrupt/sudden, Incremental, gradual and recurring. In order to adapt to concept drift we first need to detect it. Concept drift detection is a widely researched field. Static classifiers do not work well data that has concept drift in it.(wiki)

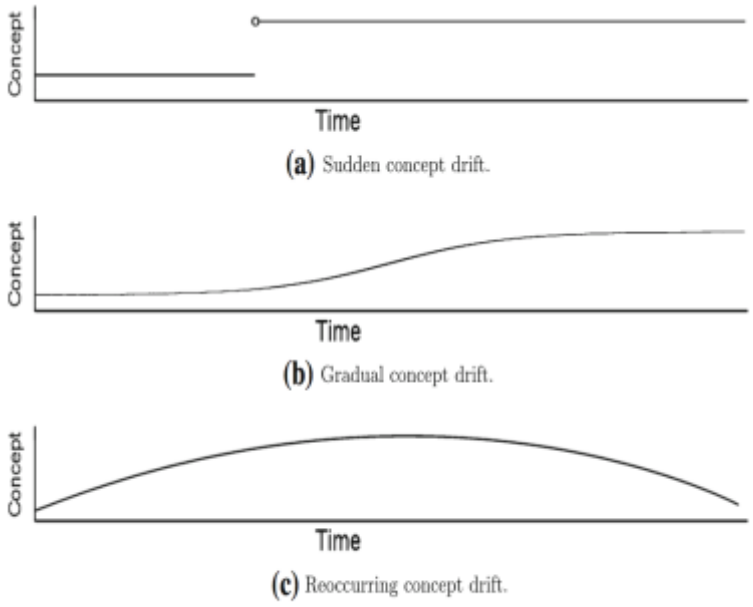


Fig 2.3.1 Different types of Concept Drift

Both these problems occurring, individually or simultaneously, significantly hinder the performance and accuracy of the learning algorithms. Several methods have been proposed that are able to adapt to these problems but when they occur concurrently each problem compounds the other. In such a case the algorithm may not be able to predict with the same accuracy.

2.4. Existing methods:

2.4.1 Incremental Algorithm (J. Gama, I. liobait, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation") [5]

The most commonly and famous method used to handle concept drift is using an incremental algorithm. The incremental algorithm processes the stream of examples one by one and stores some information (statistics) after each example is processed. The learning model is continually updated using the latest example. The learning model stores some summaries about the previous trainings. This helps the model to adapt to any new concept drift skewed data while keeping the original learning intact.

This method suffers from the stability-plasticity dilemma. This dilemma states that there should be a balance between learning from new datasets (as they could be noise) and being able to learn new patterns. Also common incremental algorithms like CVFDT might handle concept drift but do not work well with online data streams. It requires multiple passes over the data which is not feasible for real-time applications. (4)

2.4.2 Statistical Process Control Algorithm ("Concept Drift Detection and Adaptation with Hierarchical Hypothesis Testing", Shujian Yu, Zubin Abraham, Heng Wang, Mohak Shah and Jose C. Pr ´ ´ıncipe)[7]

The SPC(Statistical Process Control) algorithm one of the most first methods used to detect drifts in the data stream. This algorithm however works on the principle of static learning. Two parameters are defined for the classifier model i.e. the error rate and the standard deviation. Every time a new example is classified, a summary of the data element is stored offline. These 2 parameters are updated while learning and the algorithm uses a warning level to keep the context window to an optimal size. Whenever the warning level of either of the parameters is reached, a new decision model is created to replace the earlier one. The present examples in the context window are used for learning of the newly created decision model.It shows very high accuracy in recognizing concept drifts however it shows lesser sensitivity to gradual drifts. The major drawback is that a lot of information needs to be stored offline.

2.4.3 Genetic Process Algorithm ("Adapting to concept drift with genetic programming for classifying streaming data", Murray Smith, Vic Ciesielski)[8]

One of the latest methods used for classification of concept drift driven streaming data is Genetic Programming(GP). Two windows are utilized i.e the training window and the testing window. Learning from fast inline data streams is difficult because there is always some latency due to complex computations (learning process). This makes it difficult to learn from all the data points and some algorithms therefore only classify the points and may not learn from all of them. GP uses the testing window to perform the classification operation. The data sets from this window are then passed on to the training window which moves slower than the testing window. The examples in the training window are then used for the learning by the GP algorithm.

One limitation of this procedure is that if the window size is large there might be some time gap between when the classifier encounters the drift and the learning model learns from this new example. This might lead to inaccuracies.

3. SYSTEM ANALYSIS

3.1. Functional Requirements

3.1.1. Generate and accept data stream with concept drift and class imbalance

The system should be able to generate and take input binary data streams with class imbalance and concept drift present in it. The data stream can also be generated using software like Massive Online analysis.

3.1.2. Detect class imbalance and concept drift

After accepting the input, the system should be able to detect concept drift and class imbalance present in the non-stationary data stream.

3.1.3. Adapt to concept drift and class imbalance

The classifiers used in the system must be able to adapt to the problem of concept drift and class imbalance so that it can classify the data stream accurately.

3.2. Non-Functional Requirements

3.2.1. Accuracy Requirements

The major requirement for the classification algorithm is its accuracy in detecting whether concept drift has occurred or not. There are various metrics that are used to evaluate the performance of the classification algorithm.

3.2.2. Accuracy and Error Rate

It gives the percentage of the test set tuples that are correctly evaluated. The error rate gives the measure of the tuples that were not correctly labelled by the classifier..

Accuracy = (TP + TN) / (P + N)

Error rate = 1 - Accuracy

These metrics are poor when evaluating the performance of classifiers in class unbalanced streaming data hence we would not use them.

3.2.3. Sensitivity and Specificity

Sensitivity and Specificity are referred to as True Positive detection rate and True Negative Detection Rate. They measure the proportion of the positive or negative tuples that are correctly identified.

Sensitivity = TP / P

Specificity = TN / N

3.2.4 Precision and Recall

Precision and recall are used to measure how many of the tuples labeled positive are actually such and how many positive tuples were actually such respectively.

Precision = TP / (TP+FP)

Recall = TP / (TP+FN) = TP / P

These measures often have an inverse relation and increasing one metric leads to a decrease in the other.

3.2.5 F-Score and F_β

This score combines the values for Precision and Recall into a single formula.

F = (2 * Precision * Recall) / (Precision + Recall)

F_β = [$(1+\beta^2)$ * Precision * Recall] / [β^2 * (Precision + Recall)]

The F-score is the harmonic mean of Precision and Recall whereas the F_β represents the weighted measure of Precision and Recall.

3.3. Specific Requirements

3.3.1. Hardware Requirements

To run the project you will require a computer with the following minimum specs

- AMD Athlon (tm) II X3 435 @2.89 GHz Processor
- 4 GB RAM [8 GB Recommended]
- GPU

3.3.2. Software Requirements

A computer running on one of the following operating systems:

➔ Microsoft Windows 7 or later

➔ Ubuntu 14.04 or later

➔ Mac OS X 10.10 or later

● Python3

● Massive Online Analysis

● Gas Sensor Array Drift Dataset Data Set

3.3.4 Use-Case Diagrams and description

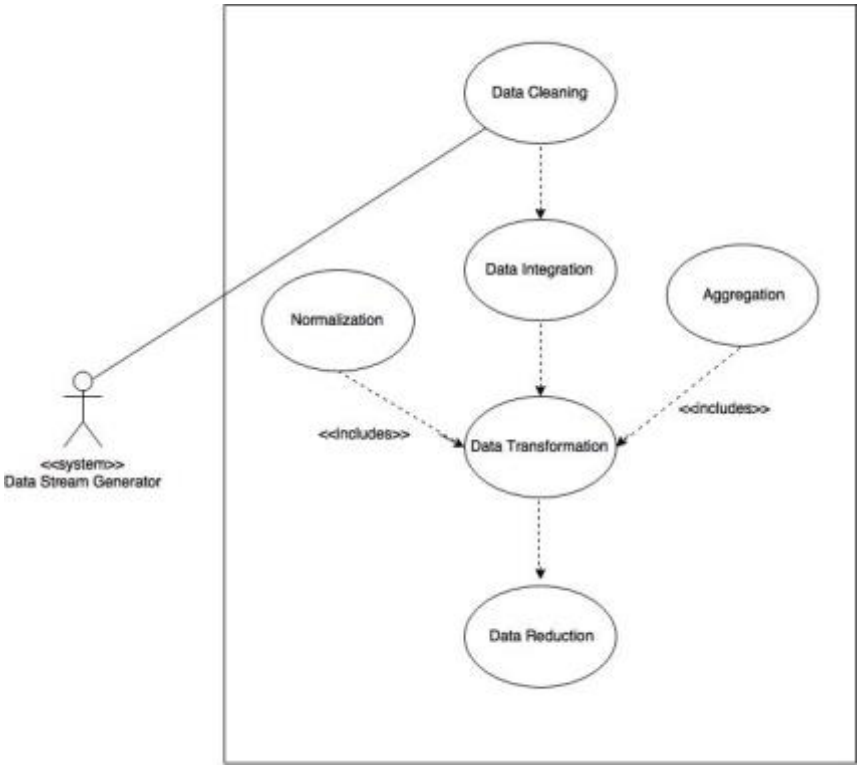


Fig 3.4.1 Use case of pre-processing the data stream

The use case diagram shows the data stream being pre-processed using various techniques like data cleaning which involves smoothing the noise data, data integration and data transformation and data reduction which involves reducing the volume but getting the same analytical results.

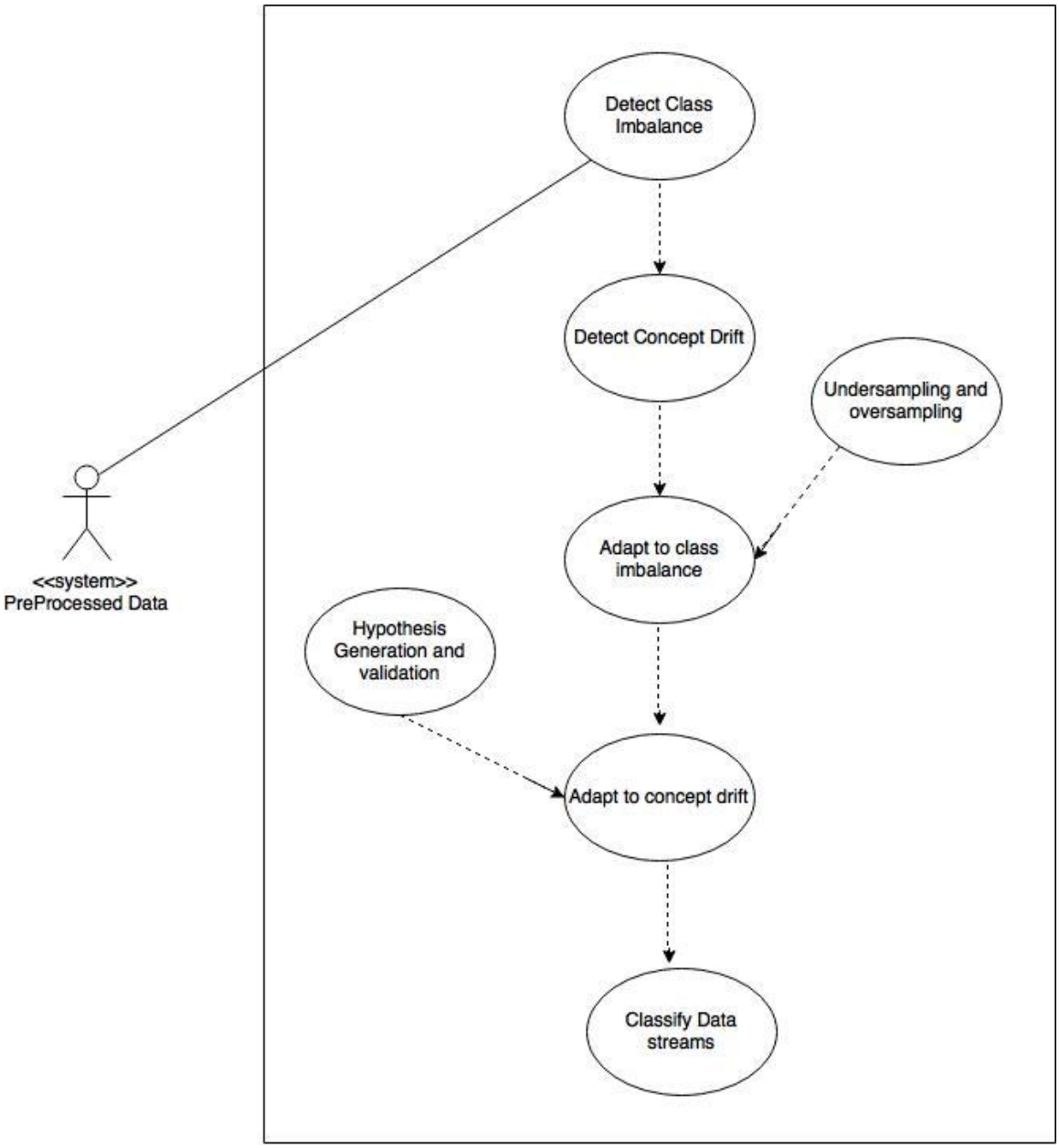


Fig 3.4.2 Detection and Adaptation to class imbalance and concept drift

4. ANALYSIS MODELING

4.1. Functional Modelling

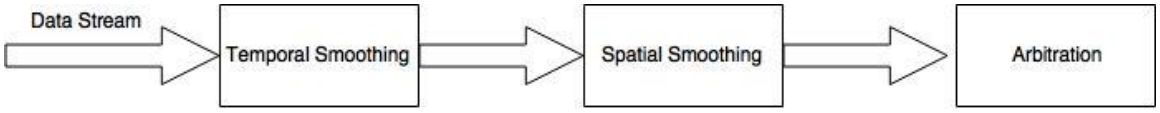


Fig 4.1.1 Function model of Data Cleaning

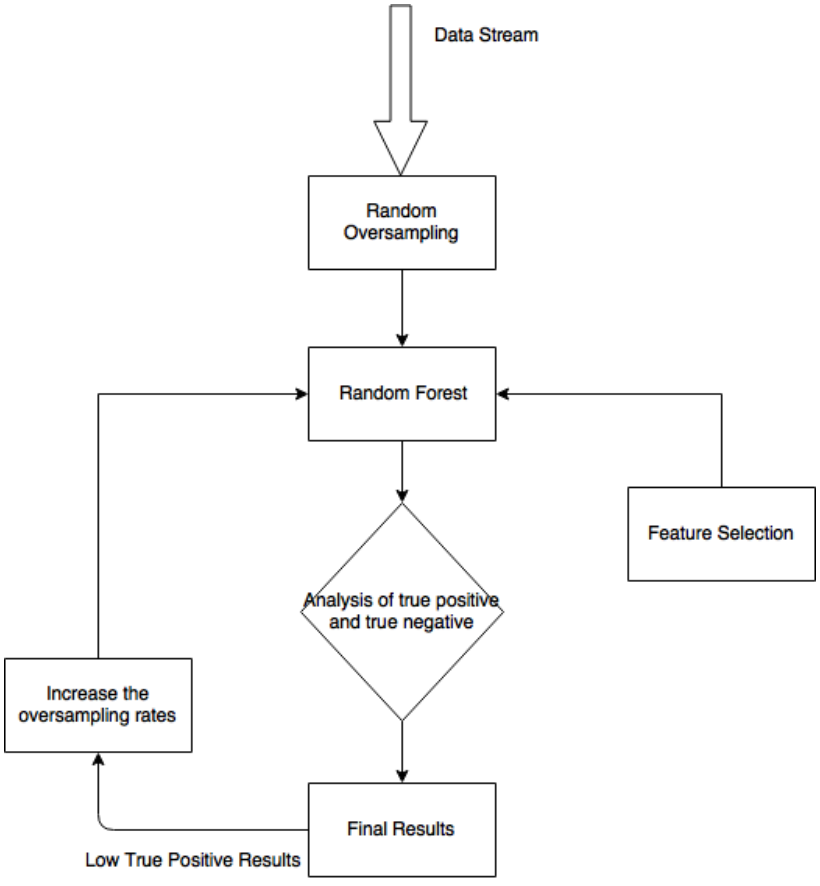
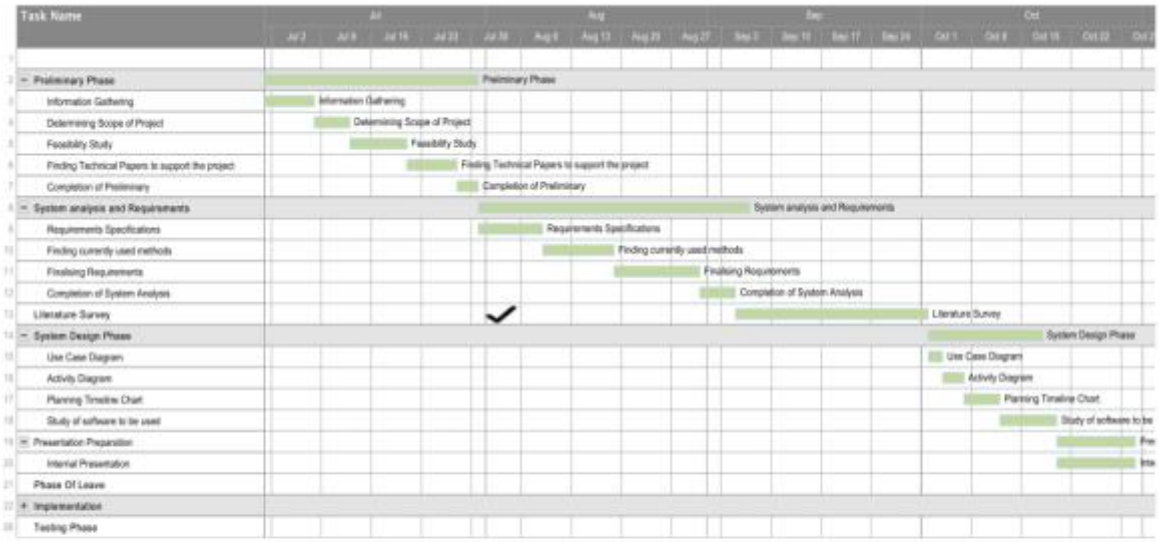
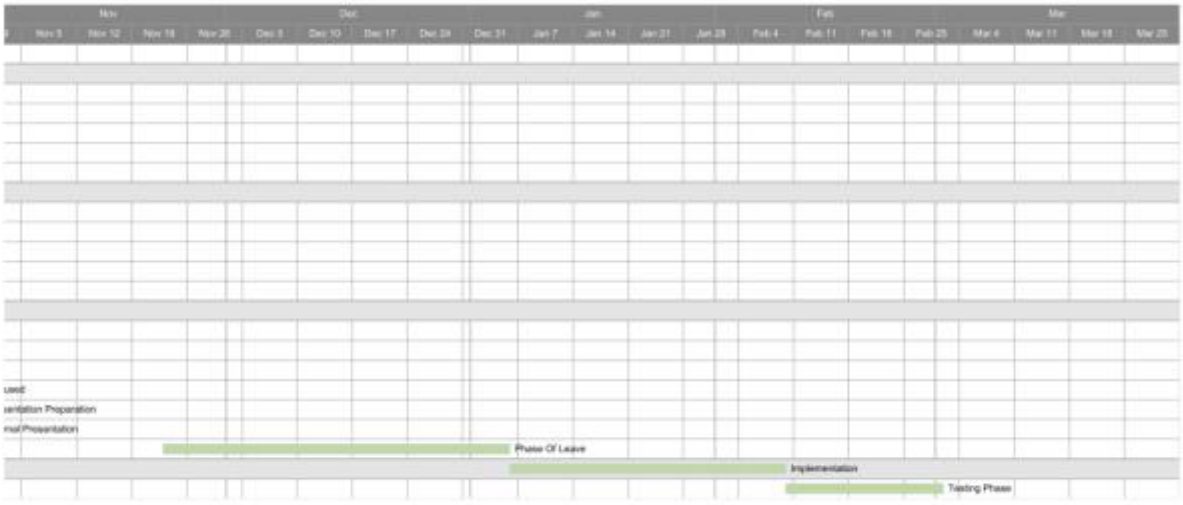


Fig 4.1.2 Functional model of data preprocessing

4.2. TimeLine Chart

	At Risk	Task Name	Start Date	End Date	Duration	Predecessors
1						
2		**Preliminary Phase**	**07/01/17**	**07/30/17**	**30d**	
3		Information Gathering	07/01/17	07/07/17	7d	
4		Determining Scope of Project	07/08/17	07/12/17	5d	
5		Feasibility Study	07/13/17	07/20/17	8d	
6		Finding Technical Papers to support the project	07/21/17	07/27/17	7d	
7		Completion of Preliminary	07/28/17	07/30/17	3d	
8		**System analysis and Requirements**	**07/31/17**	**09/06/17**	**36d**	
9		Requirements Specifications	07/31/17	08/08/17	9d	
10		Finding currently used methods	08/09/17	08/18/17	10d	
11		Finalising Requirements	08/19/17	08/30/17	12d	
12		Completion of System Analysis	08/31/17	09/04/17	5d	
13		**Literature Survey**	09/05/17	10/01/17	25d	
14		**System Design Phase**	**10/02/17**	**10/17/17**	**17d**	
15		Use Case Diagram	10/02/17	10/03/17	2d	
16		Activity Diagram	10/04/17	10/06/17	3d	
17		Planning Timeline Chart	10/07/17	10/11/17	5d	
18		Study of software to be used	10/12/17	10/19/17	5d	
19		Presentation Preparation	10/20/17	10/30/17	10d	
20		Internal Presentation	10/20/17	10/30/17	10d	
21		**Phase Of Leave**	11/23/17	01/05/18	32d	
22		**Implementation**	**01/06/18**	**02/09/18**	**40d**	
23		Find datastreams	01/06/18	01/15/18	10d	
24		Create Model	01/16/18	10/06/17	20d	
25		Train Model	10/07/17	10/11/17	10d	
26		**Testing Phase**	02/10/18	03/01/18	20d	


Task Name
Preliminary Phase
Information Gathering
Determining Scope of Project
Feasibility Study
Finding Technical Papers to support the project
Completion of Preliminary
System analysis and Requirements
Requirements Specifications
Finding currently used methods
Finalising Requirements
Completion of System Analysis
Literature Survey
System Design Phase
Use Case Diagram
Activity Diagram
Planning Timeline Chart
Study of software to be used
Presentation Preparation
Internal Presentation
Phase Of Leave
Implementation
Testing Phase


Phase Of Leave
Implementation
Testing Phase

5. DESIGN

5.1. Architectural Design

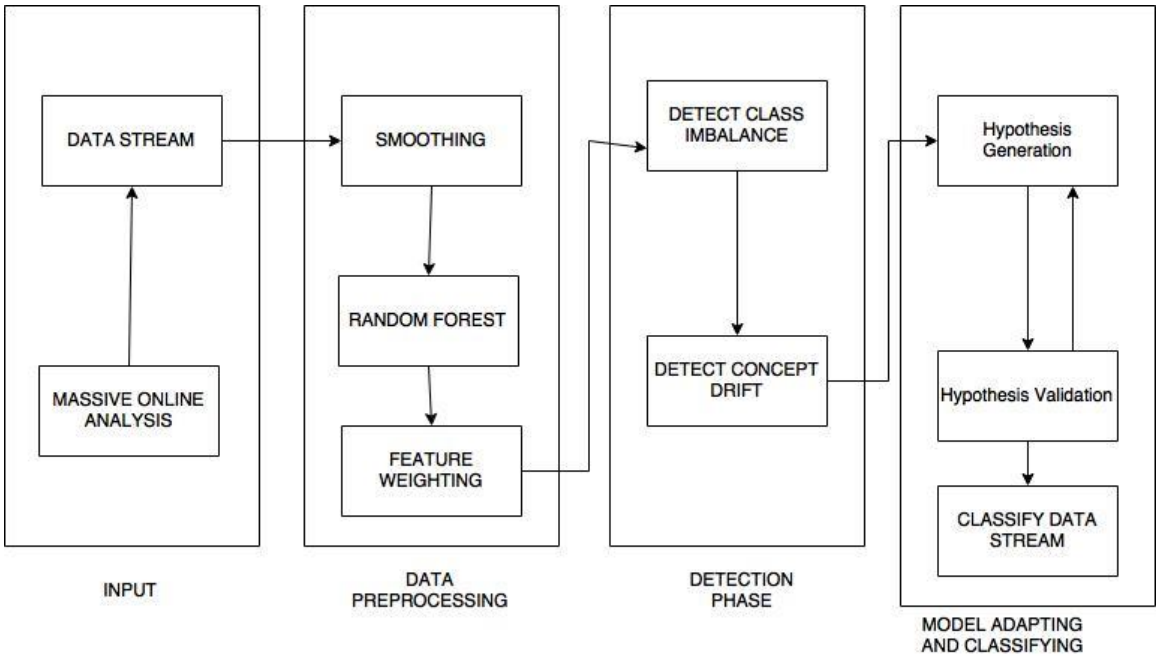


Fig 5.1.1 Architectural design of proposed system

The architectural design consists of 4 major blocks. Since there is no information or data that is stored offline no data stores are utilized. The data streams will be synthesized using the MOA (Massive Online Analysis Tool). Data from this continuous stream will be cleaned and preprocessed in the first block. Important features are extracted and weighed before they are sent to the detection phase. Class imbalance is detected over here. The last block performs the majority of the computation. The two layer hierarchical architecture for detecting and adapting to concept drift is implemented here. The final output then consists of the classification predicted by the algorithm in the last block.

6. CONCLUSION

In recent years, classification of streaming data has gained a lot of attention. The Class Imbalance and Concept drift problems hinder the performance of the classifier to a large extent. Several algorithms have been proposed for concept drift and class imbalance but they have various drawbacks. In non-stationary environments the complexity of such classification problems increases and learning in such environments has not been explored thoroughly. In our solution we propose to classify datasets in a non-static environment experiencing class imbalance in the data, where the classifier will learn from a concept that is changing in time.

REFERENCES

[1]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4283472/

[2]Data Mining Concepts and Techniques by Jiawei Han and Micheline Kamber

[3]https://pdfs.semanticscholar.org/f43e/f3428bda0177fee78bd5a522bbc5872e28b4.pdf , Mr.Rushi Longadge, Ms. Snehlata S. Dongre, Dr. Latesh Malik - Class Imbalance Problem in Data Mining: Review

[4]J. Gama, I. liobait, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys (CSUR), vol. 46, no. 4, p. 44, 2014.

[5] http://www.cpuh.in/academics/pdf/6-Arvind.pdf A Survey on Hoeffding Tree Stream Data Classification Algorithms Arvind Kumar , Parminder Kaur and Pratibha Sharma

[6] Concept Drift Detection and Adaptation with

Hierarchical Hypothesis Testing

Shujian Yu, Zubin Abraham, Heng Wang, Mohak Shah and Jose C. Pr ´ ´ıncipe

[7] http://ieeexplore.ieee.org/document/7748327/, Murray Smith, Vic Ciesielski - Adapting to concept drift with genetic programming for classifying streaming data

[8] Concept Drift Detection and Adaptation with

Hierarchical Hypothesis Testing

Shujian Yu, Zubin Abraham, Heng Wang, Mohak Shah and Jose C. Pr ´ ´ıncipe

[9] A Systematic Study of Online Class Imbalance Learning with Concept Drift Shuo Wang, Member, IEEE, Leandro L. Minku, Member, IEEE, and Xin Yao, Fellow, IEEE

[10] https://is.muni.cz/th/143255/fi_r/thesis_streams_and_concept_drift.pdf , Petr Kosina Data Stream Mining and Concept Drift: Novel Approaches and Applications

[11] Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams Adel Ghazikhani • Reza Monsefi • Hadi Sadoghi Yazdi

[12] Learning from streaming data with concept drift and imbalance: an overview T. Ryan Hoens · Robi Polikar · Nitesh V. Chawla

Acknowledgement

A task of such a magnitude always requires support and mentorship from those supervising it, along with the effort put in by the students. We acknowledge the same with great pleasure to our Principal Dr. Hari Vasudevan and our Head of Department Dr. Narendra M. Shekhokar for providing us with all the faculties that has made working on our project much easier than what it would have been otherwise.

We would also like to thank our guide, Prof. Kiran Bhowmick for giving us this opportunity to work with her on this project. Her knowledge and experience gave us the perfect platform to build on our interests and lastly this would not have been possible without the constant support of our families and friends.

We also thank the Mumbai University for giving us an opportunity to understand the practical aspects of engineering. We would thank all the teaching and non-teaching staff for supporting us through the report in every possible manner.