3.1 Methodology

This chapter presents the details of the methodology that has been adopted to achieve the research objectives. The chapter also gives sections-wise details of the framework adopted. The first investigation is targeted to understand the influence of existing techniques for classification of anonymized data streams. The experiments are performed using appropriate data mining and anonymization tools.

The second investigation is aimed to analyze performance of classification algorithms on concept drifted data streams and detection of concept drifts. The fundamental data generation mechanism changes over the time, this is common in most real-world data streams, which introduces concept drift into the data. In the applications such as mobile, streaming, remote sensing etc. data stream encounter the problem of its size and also of the changes in concept. Hence, the issue of concept drift with real and synthetic data streams is addressed in the subsequent section of this chapter. In addition, comparison of performances of single and ensemble classifier has been checked to get the insight of the effects of concept drift. Classifiers have been applied on data stream with and without concept drift for analysis. This exercise resulted in performance analysis of the classifiers at different type of data whether it is categorical, numeric or alphanumeric. Increase in internet and communication technology led to generation of data streams of dynamic environment. Due to its dynamic nature, the traditional techniques are not sufficient for privacy preservation of data streams. Researchers have been actively exploring the alternative algorithms to achieve improved privacy of data streams. Hence, in an attempt to improve privacy with the use of an efficient method of reverse engineering, a hashing based technique is proposed. So, this chapter ends with the details of design and implementation of the proposed model of privacy preserving classification. Anonymization was chosen for the study and comparison for the proposed model, because anonymization technique of privacy preservation is the one which allows all types of data such as categorical, numerical, alphanumeric etc. In summary, the three main investigations conducted to achieve the objectives are as follows:
• Classification of anonymized data streams,
• Study of classification algorithms on concept drifted data streams,
• Develop a new privacy preservation technique using hashing and design an efficient model for privacy preserving classification of data streams.

The ARX 3.3.0 is used for privacy preservation using anonymization technique. The concept drift challenge of data streams is identified using Massive Online Analysis (MOA) tool. For the prediction of future behavior of data streams, classification is done using Weka -3.6 (Waikato Environment for Knowledge Analysis) tool. The details of the experiments performed using the mentioned tools is described in the subsequent section of this chapter.

3.2 Anonymization of Data Streams

The research begins with the performance evaluation of anonymization techniques. The aim of the experiment is to study the privacy preserving characteristics of data streams in terms of information loss, response time and privacy achieved. Section 3.2.1 explains the framework for the study.

3.2.1 Framework for Anonymized Data Stream Classification

A framework has been adopted to evaluate the performance of various classification algorithms on the data stream that has been privacy preserved using different anonymization techniques as mentioned in Figure.3.1. A data stream is generated using data stream generator or collected from the repository which is referred to as the original data set D. The classification algorithms are applied on this data set D. Thereafter, the privacy preserving anonymization techniques are applied to the data set D to obtain a privacy preserved data set D’. D’ is then classified by applying various classification algorithms. The results of classification of original data set D and that of privacy preserved data set D’ are then compared.
Figure 3.1 Framework for classification of anonymized data stream
3.2.1.1 Data collection

Experiments have been carried out on the standard adult database from UCI (University of California Irvine) machine learning repository with 30K instances and 15 attributes. 9 attributes have been chosen for experimental purpose. Table 3.1 shows the attributes of the adult data set and its values. The attributes were sex, age, race, marital-status, education, native-country, work class, occupation, and salary-class. The data set contained numerical and categorical attributes as well, which was suitable for generalization required. Salary was chosen as the class attribute. Sex, race, marital-status, education, native-country, work class, occupation were considered as quasi-identifier and age as sensitive attribute.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Attribute</th>
<th>Attribute Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>age</td>
<td>Continuous</td>
</tr>
<tr>
<td>3.</td>
<td>fnlwgt</td>
<td>Continuous</td>
</tr>
<tr>
<td>4.</td>
<td>education</td>
<td>Bachelors, Some-college, 11th, HS-grade, Prof-school, Assoc-acdm, Asso-voc,9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th</td>
</tr>
<tr>
<td>5.</td>
<td>education-num</td>
<td>Continuous</td>
</tr>
<tr>
<td>7.</td>
<td>occupation</td>
<td>Tech-support, Craft-repair, Other-services, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, machine-op-inspct, Adm-clerical</td>
</tr>
</tbody>
</table>
8. relationship | Wife, own-child, Husband, Not-in-family, Other-relative, Unmarried  
9. race | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black  
10. sex | Male, Female  
11. capital-Gain | Continuous  
12. capital-Loss | Continuous  
13. hours-per-week | Continuous  
14. native-country | United-states, Combodia, England, Puerto-raco, Canada, Germany, Outlying-US(guam-USVI), India, Japan, Greece, South, China  
15. income | Greater than 50K, less than equal to50 K

3.2.1.2 Anonymization of Data Stream using ARX 3.3.0

In this work, performance evaluation of classification and privacy preserving characteristics of data streams using anonymization technique has been conducted. The classification algorithms are applied on original data set and also to the privacy preserved data set using ARX 3.3.0 tool. Performance results by the various classification algorithms are then compared. K-anonymity, population uniqueness, average re-identification risk and ℓ-diversity techniques have been applied for anonymization.

ARX is a comprehensive open source software for anonymizing sensitive personal data. It supports risk-based anonymization using super-population models, strict-average risk and k-map, syntactic privacy models such as k-anonymity, ℓ-diversity, t-closeness, δ-disclosure privacy and δ-presence, semantic privacy models, such as (ɛ, δ)-differential privacy, data
transformation with generalization, suppression, micro aggregation and top/bottom-coding as well as global and local recoding methods for analyzing data utility and methods for analyzing re-identification risks. The software is able to handle very large datasets on commodity hardware and it features an intuitive cross-platform graphical user interface as shown in Figure.3.2.

![Graphical User Interface of ARX 3.3.0](image)

ARX allows to assign privacy risks and data types to attributes and to specify transformation rules in terms of generalization hierarchies and/or micro-aggregation functions. In terms of privacy risks, ARX distinguishes between the following four types of attributes:

- **Direct and Indirect Identifiers**: ARX can be used to remove direct identifiers (e.g., names) from datasets and to enforce further constraints on indirect identifiers.
- **Quasi-identifiers**: These are attributes that do not directly identify an individual but may together with other indirect identifiers form an identifier that can be used for linkage
attacks. It is typically assumed that information about indirect identifiers is available to the attacker (in some form of background knowledge) and that they cannot simply be removed from the dataset (e.g. because they are required for analyses).

- **Sensitive attributes:** These attributes are kept as is but may be required to meet some privacy guarantees. The encode properties with which individuals are not willing to be linked with. As such, they might be of interest to an attacker and, if disclosed, could cause harm to data subjects. They will be kept unmodified but may be subject to further constraints, such as t-closeness or l-diversity. Examples are diagnoses.

- **Insensitive attributes** are not associated with any privacy risks. They will be kept unmodified. Insensitive attributes are kept as they are.

### 3.2.2 Implementation of Methods in ARX

A broad spectrum of methods have been implemented in ARX, including (1) methods for analyzing re-identification risks, (2) methods for analyzing data utility, (3) syntactic privacy models and (4) methods for transforming data.

#### 3.2.2.1 Method for Analyzing Re-identification Risks

Re-identification risks has been analyzed based on sample characteristics or on the concept of uniqueness. Uniqueness can either be determined based on the sample itself or it may be estimated with super-population models. These statistical methods estimate characteristics of the overall population with probability distributions that are parameterized with sample characteristics.
3.2.2.2 Methods for Analyzing Utility

The utility of a dataset for a given usage scenario can be analyzed manually as well as automatically. For manual analysis, ARX implements methods from descriptive statistics. For automatic analysis of data utility, ARX employs so called utility measures, which measure the loss of information induced by transformations. These methods enable ARX to completely classify the solution space and automatically determine the transformation with optimal data utility. As it may not always be possible to automatically determine the solution that best fits a user’s requirements, the classified solution space can be explored and alternative transformations can be analyzed.

3.2.2.3 Syntactic Privacy Models

In the context of statistical disclosure control, three different types of privacy threats are commonly considered:

- **Membership disclosure**
  Membership disclosure means that data linkage allows an attacker to determine whether or not data about an individual is contained in a data set. While this does not directly disclose any information from the data set itself, it may allow an attacker to infer meta-information. While this deals with implicit sensitive attributes (meaning attributes of an individual that are not contained in the data set), other disclosure models deal with explicit sensitive attributes.

- **Attribute disclosure**
  Attribute disclosure may be achieved even without linking an individual to a specific item in a data set. It protects sensitive attributes, which are attributes from the data set with which individuals are not willing to be linked with. As such, they might be of interest to an attacker.
and, if disclosed, could cause harm to data subjects. As an example, linkage to a set of data entries allows inferring information if all items share a certain sensitive attribute value.

- **Identity disclosure (or re-identification)**
  Identity disclosure (or re-identification) means that an individual can be linked to a specific data entry. This is a very serious type of attack, as it has legal consequences for data owners according to many laws and regulations worldwide. From the definition it also follows that an attacker can learn all sensitive information contained in the data entry about the individual. Multiple privacy models have been proposed to prevent these types of disclosure, from which the following are currently implemented in the ARX anonymization tool.

- **K-Anonymity**
  This privacy model is well-known. It aims at protecting datasets from identity disclosure following the prosecutor attacker model. A dataset is k-anonymous if, regarding the quasi-identifiers, each data item cannot be distinguished from at least k – 1 other data items. The tuples with identical values for all quasi-identifiers form an equivalence class.

- **Population uniqueness**
  ARX also supports several relaxed privacy models for protecting datasets against re-identification attacks following the marketer model. For example, thresholds can be enforced of the proportion of records that are unique within the underlying population. If no explicit information about this population has been loaded into ARX, this information can be estimated with super-population models. These statistical methods estimate characteristics of the overall population with probability distributions that are parameterized with sample characteristics. We provide default settings for populations, such as the USA, UK, France or Germany, and support the methods by Pitman, Zayatz and the SNB model. ARX also implements the decision rule proposed and validated for clinical datasets.
- **k-Map**
  This privacy model is a variant of k-anonymity, which considers explicit information about the underlying population.

- **Strict-average risk**
  ARX also implements strict-average risk, which is a combination of a threshold on average re-identification risks combined with k-anonymity. It can be used to protect datasets from marketer attacks.

- **ℓ-Diversity**
  This privacy model protects a dataset against attribute disclosure. It ensures that the values of a set of predefined sensitive attributes are at least ℓ-diverse within each equivalence class. ℓ-Diversity also implies ℓ-anonymity. To fulfill the basic definition of ℓ-diversity, a sensitive attribute must have at least “well represented” distinct values in each equivalence class.

- **t-Closeness**
  This privacy model is an alternative for the protection against attribute disclosure. The basic idea is that equivalence classes are not allowed to stand out in the dataset. To achieve this, the distributions of the values of the sensitive attribute within each equivalence class must have a distance of less than t to the distribution of the attribute values in the original dataset. For measuring distances between distributions, the earth mover’s distance (EMD) is used.

- **δ-Disclosure privacy**
  This privacy model is a very strict measure for mitigating attribute disclosure.

- **δ-Presence**
  This model aims at protecting datasets against membership disclosure. The basic idea is to model the disclosed dataset as a subset of larger dataset that represents the attacker’s background knowledge. A dataset is (δmin, δmax)-present if the probability that an
individual from the global dataset is contained in the research subset lies between $\delta_{\text{min}}$ and $\delta_{\text{max}}$.

### 3.2.2.4 Method for Data Transformation

A combination of generalization, suppression and micro aggregation have been used in ARX. Transformation methods can be applied using global recoding and a local recoding scheme. For generalizing data items, the framework employs generalization hierarchies, which can easily be constructed by end-users to meet their requirements. The tool provides several methods for helping users with constructing such hierarchies. ARX implements combined support for two specific types of transformation methods: multi-dimensional global recording with full-domain attribute generalization and local recoding with tuple suppression. With tuple suppression, a subset of the data items is allowed to not adhere to the specified privacy models. During the anonymization process this subset is removed from the dataset, as long as the total number of suppressed tuples is lower than a user-defined threshold. This allows to further reduce information loss. This combination of methods is easy to understand by users and allows providing several advanced options for configuring and parameterizing the transformation process.

### 3.2.2.5 The Value of k

In k-anonymity, k has been calculated at different value. The value of k is depend on the accuracy parameter. For example, the value of k have chosen 5 because it is stable and round off values among all k’s value. The Table 3.2 accuracy at different values of k.
Table 3.2 Accuracy at different values of k

<table>
<thead>
<tr>
<th>k</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>81.16</td>
</tr>
<tr>
<td>3</td>
<td>82.26</td>
</tr>
<tr>
<td>4</td>
<td>83.28</td>
</tr>
<tr>
<td>5</td>
<td>83.30</td>
</tr>
<tr>
<td>6</td>
<td>83.32</td>
</tr>
<tr>
<td>7</td>
<td>83.29</td>
</tr>
<tr>
<td>8</td>
<td>83.21</td>
</tr>
</tbody>
</table>

### 3.2.3 Anonymization Workflow

The biggest challenge in data anonymization is to achieve a balance between data utility and privacy. ARX models many different aspects of this balancing process. Methods are combined into a multi-step workflow that allows users to iteratively adjust parameters, until the result matches their requirements. As depicted in the Figure 3.3, the basic steps consist of 1) configuring the transformation process, 2) exploring the solution space and 3) analyzing input and output data. In the configuration phase, input data is loaded, generalization hierarchies are created and all further parameters, such as privacy models, are specified.

![Workflow of anonymization technique](image)

**Figure. 3.3 Workflow of anonymization technique**
When the solution space has been characterized by executing the anonymization algorithm, the exploration phase allows searching the solution space for privacy-preserving data transformations that fulfill a user’s requirements. To assess suitability, the analysis phase allows comparing transformed datasets to the original input dataset. Moreover, datasets may be analyzed regarding re-identification risks. Based on these analyses, further solution candidates might be considered and analyzed, or the configuration of the anonymization process might be altered.

![Figure. 3.4 Loading dataset in the tool](image)

Figure. 3.4 shows the loading of dataset in ARX tool. For loading any data set the file is need to be imported using file menu of toolbar. Once the loading dialog box executes the data file input data is indicated at the top of the toolbar. In the input data window all attribute values
are displayed after loading as shown in Figure. 3.5. The attributes can be identified using attribute notations in ARX i.e. quasi-identifiers, sensitive, insensitive etc. Using data transformation attributes can also be generalized such as value of age = 64 is generalized as 60<age<65.

The output window opens up next to the execution of anonymization technique by the tool. The output is generalized and suppressed. The properties i.e. information loss, successors, predecessors, transformation, threshold and population models etc. are indicated in the results. These parameters serve as the evaluation parameters for different techniques of anonymizations e.g. k-anonymity, ℓ-diversity, t-closeness, population uniqueness and
average-reidentification-risk. The information loss is one of the major focus because the objective is related to privacy model.

### 3.3 Classification of Original Data Stream

For the classification of original data stream without any privacy preservation technique, Jrip, J48 and Naïve bayes classification algorithms have been used. Salary has been take as class attribute. 10 folds cross validation was taken as the test mode in WEKA-3.6. Performance of classification algorithm have been evaluated in terms of mean absolute error, kappa statistics, and accuracy and time parameters.

### 3.4 Classification of Anonymized Data Stream

For the classification of data stream that has been preserved using anonymization technique, Jrip, J48 and Naïve bayes classification algorithms have been used. l-diversity, k-anonymity, population uniqueness and average re-identification risk techniques of anonymization were used for anonymization. Salary has been take as class attribute. 10 folds cross validation was taken as the test mode in WEKA-3.6. Performance of classification algorithm have been evaluated in terms of mean absolute error, kappa statistics, accuracy and time parameters.
3.5 Evaluation of Classification algorithms on Concept Drifted Data Streams

This section presents the experiments aimed to overcome the concept drift challenge of data stream and classification performance analysis on concept drifted data. Concept drift primarily refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time. By categorizing existing strategies for handling concept and distinct and popular algorithms the experiment has been carried out. MOA (Massive Online Analysis) tool has been used for classification of data streams and detection of concept drift. Classification has been done using Weka-3.6, Waikato Environment for Knowledge Analysis, which is an open-source workbench containing implementations of a wide range of batch machine learning methods. The details of the framework followed is described in section 3.3.1.

3.5.1 Framework for Concept Drift Detection and Classification of Concept Drifted data streams

Figure 3.6 shows the framework for concept drift detection and classification of concept drifted data streams. Data streams are collected through data stream generator and repositories. Various inbuilt preprocessing learning algorithms make the data stream in the MOA executable format. Segments of data stream are selected for preprocessing. There are two types of approaches, namely evolve and triggered learner when concept drift occurs. With the help of these two methods concept drift in data stream can detect and overcome. Evolving learners employ change detection mechanism as a tool to reduce computational complexity. Evolving methods, on the contrary, do not maintain an explicit link between data progress and model construction, and usually do not detect changes. Evolving learners adapt at every step and it often corresponds to a forgetting mechanism. The change detection
methods such Drift Detection Method (DDM), Early Drift Detection Method (EDDM), CUSUMTEST, Geometry Moving Average Test detects if there is a concept change. After that classification is performed using different ensemble/single classifiers. Knowledge is attained on completion of classification process.

Figure. 3.6 Framework for concept drift detection and classification of concept drifted data streams
3.5.1.1 Collection of Data Streams

For experimental purpose synthetic and real data streams are used.

**Synthetic data stream:** Two synthetic data streams were selected to implement the experiments: LED data stream is composed of 24 categorical attributes. The aim is to predict the digit displayed on a seven-segment LED display, proposed by Breiman et al. in 1984 and Agarwal data stream generated through MOA tool. These data streams are usually used in the concept drift research area [Gama et al. (2013)]. Agarwal generator stream and LED generator Stream has Memory chunk frequency – 20,000; Position of drift – 50,000 and the length of change i.e. Window Size (W) – 5000.

**Real data stream:** Two real data streams are also taken for experiment: Airline, proposed by Elena Ikonomovska in 2009. It consists of 120 million records, containing flight arrival and departure details for all the commercial flights. Poker-Hand is taken from UCI repository and contain 1,000,000 instances. Airline data set and Poker hand data sets, are the two real data sets with memory chunk frequency 10,000 and 20000, respectively. Table 3.3 shows characteristics of data streams used in MOA.

<table>
<thead>
<tr>
<th>Data Stream</th>
<th>Type of Data</th>
<th>No. of Transaction</th>
<th>No. of attribute</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agarwal generator stream</td>
<td>Synthetic</td>
<td>10,0000</td>
<td>10</td>
<td>10MB</td>
</tr>
<tr>
<td>LED generator Stream</td>
<td>Synthetic</td>
<td>10,0000</td>
<td>24</td>
<td>15MB</td>
</tr>
<tr>
<td>Airline</td>
<td>Real</td>
<td>5,39,395</td>
<td>13</td>
<td>40MB</td>
</tr>
<tr>
<td>Poker-hand</td>
<td>Real</td>
<td>1,025,110</td>
<td>11</td>
<td>35MB</td>
</tr>
</tbody>
</table>
3.5.2 MOA for Concept Drift Detection

Massive Online Analysis (MOA) is the most popular open source framework for data stream mining. It includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, concept drift detection and recommender systems). MOA is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. MOA includes a collection of offline and online methods as well as tools for evaluation. In particular, it implements boosting, bagging, and Hoeffding Trees, all with and without Naive Bayes classifiers at the leaves.

A data stream environment has different requirements from the traditional batch learning setting. The most significant are the following:

Requirement 1: Process an example at a time, and inspect it only once (at most)
Requirement 2: Use a limited amount of memory
Requirement 3: Work in a limited amount of time
Requirement 4: Be ready to predict at any time

Figure.3.7 illustrates the typical use of a data stream classification algorithm, and how the requirements fit in a repeating cycle:

- The algorithm is passed the next available example from the stream (Requirement
- The algorithm processes the example, updating its data structures. It does so without exceeding the memory bounds set on it (requirement 2), and as quickly as possible (Requirement 3).
- The algorithm is ready to accept the next example. On request it is able to predict the class of unseen examples (Requirement 4).
MOA graphical user interface consists of menu bar, textual output, multiple task window and graphical representation parts as shown in Figure 3.8.
MOA allows multiple commands to execute simultaneously. This also allows the exporting of output data in *.txt file. The command execution and configuration of model is depicted in Figure 3.9.

In traditional batch learning the problem of limited data is overcome by analyzing and averaging multiple models produced with different random arrangements of training and test data. In the stream setting the problem of (effectively) unlimited data poses different challenges. One solution involves taking snapshots at different times during the induction of a model to see how much the model improves.

- The evaluation procedure of a learning algorithm determines which examples are used for training the algorithm, and which are used to test the model output by the algorithm.
When considering what procedure to use in the data stream setting, one of the unique concerns is how to build a picture of accuracy over time. Two main approaches arise:

- Holdout: When traditional batch learning reaches a scale where cross-validation is too time consuming, it is often accepted to instead measure performance on a single holdout set. This is most useful when the division between train and test sets has been pre-defined, so that results from different studies can be directly compared.

For change detection CUSUM test, statistical test, drift detection method, early drift detection method have been used in this work.

### 3.5.3 Experimental Setup

MOA is written in Java. The main benefits of Java are portability, where applications can be run on any platform with an appropriate Java virtual machine, and the strong and well-developed support libraries. Use of the language is widespread, and features such as automatic garbage collection help to reduce programmer burden and error. MOA contains stream generators, classifiers and evaluation methods.

### 3.5.4 Evolving Data Streams

MOA streams are build using generators, reading ARFF files, joining several streams, or filtering streams. MOA streams generators allow to simulate potentially infinite sequence of data. There are Random Tree, SEA Concepts, STAGGER Concepts, Rotating Hyperplane, Random RBF, LED Generator, Waveform, and Function Generator.

#### 3.5.4.1 generators.AgrawalGenerator

Generates one of ten different pre-defined loan functions. It was introduced by Agrawal et al. The generator produces a stream containing nine attributes, six numeric and three
categorical. Although not explicitly stated by the authors, a sensible conclusion is that these attributes describe hypothetical loan applications. There are ten functions defined for generating binary class labels from the attributes. Presumably these determine whether the loan should be approved.

Parameters:

f: Classification function used, as defined in the original paper.
i: Seed for random generation of instances.
p: The amount of perturbation (noise) introduced to numeric values
b: Balance the number of instances of each class.

3.5.4.2 generators.HyperplaneGenerator

Generates a problem of predicting class of a rotating hyperplane. (Ref Hulten G A hyperplane in d-dimensional space is the set of points x that satisfy

\[ \sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i \]

where \( x_i \) is the \( i^{th} \) coordinate of \( x \). Examples for which \( \sum_{i=1}^{d} w_i x_i \geq w_0 \) are labeled positive, and examples for which \( \sum_{i=1}^{d} w_i x_i < w_0 \) are labeled negative.

Parameters:
i : Seed for random generation of instances.
c : The number of classes to generate
a : The number of attributes to generate.
k : The number of attributes with drift.
t : Magnitude of the change for every example.
n : Percentage of noise to add to the data.
S : Percentage of probability that the direction of change is reversed.

3.5.4.3 generators.LEDGenerator

Generates a problem of predicting the digit displayed on a 7-segment LED display. This data source originates from the CART book. An implementation in C was donated to the UCI machine learning repository by David Aha. The goal is to predict the digit displayed
on a seven-segment LED display, where each attribute has a 10% chance of being inverted. It has an optimal Bayes classification rate of 74%. The particular configuration of the generator used for experiments (led) produces 24 binary attributes, 17 of which are irrelevant.

Parameters:
- i : Seed for random generation of instances.
- n : Percentage of noise to add to the data
- s : Reduce the data to only contain 7 relevant binary attributes

### 3.5.4.4 generators.LEDGeneratorDrift

Generates a problem of predicting the digit displayed on a 7-segment LED display with drift.

Parameters:
- i : Seed for random generation of instances.
- n : Percentage of noise to add to the data
- s : Reduce the data to only contain 7 relevant binary attributes
- d : Number of attributes with drift

### 3.5.4.5 generators.RandomRBFGenerator

Generates a random radial basis function stream. This generator was devised to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model. The RBF (Radial Basis Function) generator works as follows:

- A fixed number of random centroids are generated.
- Each center has a random position, a single standard deviation, class label and weight.
- New examples are generated by selecting a center at random, taking weights into consideration so that centers with higher weight are more likely to be chosen.
- A random direction is chosen to offset the attribute values from the central point.
• The length of the displacement is randomly drawn from a Gaussian distribution with standard deviation determined by the chosen centroid.

• The chosen centroid also determines the class label of the example.

• This effectively creates a normally distributed hyper sphere of examples surrounding each central point with varying densities. Only numeric attributes are generated.

Parameters:
- r : Seed for random generation of model
- i : Seed for random generation of instances
- c : The number of classes to generate
- a : The number of attributes to generate
- n : The number of centroids in the model

3.5.4.6 generators.RandomRBFGeneratorDrift

Generates a random radial basis function stream with drift. Drift is introduced by moving the centroids with constant speed.

Parameters:
- r : Seed for random generation of model
- i : Seed for random generation of instances
- c : The number of classes to generate
- a : The number of attributes to generate
- n : The number of centroids in the model

3.5.4.7 generators.RandomTreeGenerator

Generates a stream based on a randomly generated tree. This generator is based on that proposed in (REF).

It produces concepts that in theory should favour decision tree learners. It constructs a decision tree by choosing attributes at random to split, and assigning a random class label.
to each leaf. Once the tree is built, new examples are generated by assigning uniformly distributed random values to attributes which then determine the class label via the tree.

3.5.4.8 generators.SEAGenerator

Generates SEA concepts functions. This dataset contains abrupt concept drift, first introduced in paper.
It is generated using three attributes, where only the two first attributes are relevant. All three attributes have values between 0 and 10. The points of the dataset are divided into 4 blocks with different concepts. In each block, the classification is done using $f_1 + f_2 \leq \theta$ where $f_1$ and $f_2$ represent the first two attributes and $\theta$ is a threshold value. The most frequent values are 9, 8, 7 and 9.5 for the data blocks.
Parameters:
f: Classification function used, as defined in the original paper
i: Seed for random generation of instances
b: Balance the number of instances of each class
n: Percentage of noise to add to the data

3.5.4.9 generators.WaveformGenerator

Generates a problem of predicting one of three waveform types. It shares its origins with LED, and was also donated by David Aha to the UCI repository. The goal of the task is to differentiate between three different classes of waveform, each of which is generated from a combination of two or three base waves. The optimal Bayes classification rate is known to be 86%. There are two versions of the problem, wave 21 which has 21 numeric attributes, all of which include noise, and wave 40 which introduces an additional 19 irrelevant attributes.
Parameters:
i: Seed for random generation of instances
n: Adds noise, for a total of 40 attributes
3.6 WEKA for Classification of Data Streams

Weka is open source software under the GNU General Public License. System is developed at the University of Waikato in New Zealand. “Weka” stands for the Waikato Environment for Knowledge Analysis. The system is written using object oriented language Java. There are several different levels at which Weka can be used. Weka provides implementations of state-of-the-art data mining and machine learning algorithms. Weka contains modules for data preprocessing, classification, clustering and association rule extraction. Main features of Weka include:

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 15 attribute/subset evaluators + 10 search algorithms for feature selection.
- 3 algorithms for finding association rules
- 3 graphical user interface
  - The Explorer (exploratory data analysis)
  - The Experimenter (experimental environment)
  - The Knowledge Flow (new process model inspired interface)

**Weka Application Interfaces**

- Explorer – preprocessing, attribute selection, learning, visualization
- Experimenter – testing and evaluating machine learning algorithms
- Knowledge Flow – visual design of KDD process
- Simple Command-line – A simple interface for typing commands
3.6.1 WEKA data formats

Attribute Relation File Format (ARFF) is the default file type for data analysis in Weka but data can also be imported from various formats. ARFF (Attribute Relation File Format) has two sections:

- the Header information defines attribute name, type and relations
- the Data section lists the data records
- CSV: Comma Separated Values (text file)
- Data can also be read from a database using ODBC connectivity.

Attribute Relation File Format (arff): ARFF format of weather dataset from sample data in Weka is presented here. Attribute type is specified in the header tag. Nominal attribute have the distinct values of attribute in curly brackets along with attribute name. Numeric attribute is specified by the keyword real along with attribute name.

@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no

3.6.2 WEKA Explorer

Click the Explorer on Weka GUI Chooser

– On the Explorer window, click button “Open File” to open a data file from the folder where your data files stored.

– Then select the desired module (Preprocess, Classify, Cluster, Association etc.) from the upper tabs.

3.6.3 Data Preprocessing

Some attributes may not be required in the analysis, and then those attributes can be removed from the dataset before analysis. For example, attribute instance number of iris dataset is not required in analysis. This attribute can be removed by selecting it in the attributes check box, and clicking remove. Resulting dataset then can be stored in arff file format.
3.6.4 Weka Data Analysis

Mean absolute error (MAE): The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The equation is given in the library references. Expressed in words, the MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i| = \frac{1}{n} \sum_{i=1}^{n} |e_i|
\]  

Where, \( f_i \) = prediction value  
\( y_i \) = true value

Root mean squared error (RMSE): The RMSE is a quadratic scoring rule which measures the average magnitude of the error. The equation for the RMSE is given in both of the references. Expressing the formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable. The MAE and the RMSE can be used together to diagnose the variation in the errors in a set of forecasts. The RMSE will always be larger or equal to the MAE; the greater difference between them, the greater the variance in the individual errors in the sample. If the RMSE=MAE, then all the errors are of the same magnitude.

Accuracy: It is a measure to determine the utility of the dataset.

\[
\text{Accuracy} = \left( \frac{\text{Correctly classified instances}}{\text{Total number of Instance}} \right) \times 100
\]  

(3.3)
**Kappa statistic:** It measures the agreement of prediction with the true class, formulated as given below:

\[ k = \frac{p_0 - p_e}{1 - p_e} \quad (3.4) \]

where, \( p_0 \) is the relative observed agreement, \( p_e \) is the hypothetical probability of chance agreement.

**Precision:** which is defined as proportion of instances that are truly of a class divided by the total instances classified as that class.

\[ \text{Precision} = \frac{tp}{tp + fp} \quad (3.5) \]

where, \( tp \) = No. of examples predicted positive that are actually positive and \( fp \) = No. of examples predicted positive that are actually negative.

**Recall:** Recall is defined as proportion of instances classified as a given class divided by the actual total in that class. Recall means how complete the results are.

\[ \text{Recall} = \frac{tp}{tp + fn} \quad (3.6) \]

Where \( fn \) is No. of examples predicted negative that are actually positive.

**F-measures:** It is a measure that combine recall and precision which is given as below:

\[ \text{F-measure} = \frac{(2 \times \text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (3.7) \]

The data to be classified using this tool is preprocessed first by preprocess tab of menu bar. The adult dataset after anonymization has been preprocessed and imported using Open file option of the toolbar. Figure.3.11 shows the selection and import of anonymized data file in comma separated format as ‘outputaverage_reindentification.csv’ for its classification.
Many traditional algorithms do not exhibit good performance because of the dynamic nature of data. The researchers have been actively finding the newer model of privacy preservation for obtaining improved privacy characteristics. In an attempt to improve privacy along with an efficient method of reverse reengineering, a hashing based privacy preservation technique is proposed in this section.
3.7.1 Basics of Hashing

Conversion of a string of character in short fixed length key which represent original character is known as hashing. The values returned by a hash function are called hash values or hash codes. Hashing uses map data structure to store the data set in key-value. Hashing has two main operations, one is data insertion and the other is data retrieval. The procedure of data insertion in hashing is accomplished by using Put (key, value) method. The algorithm representing the put (key, value) method is as follows:

Algorithm for Put (key, value) method in hashing

Input: String of characters
Output: Data item insertion
Method:

int hashValue = key.hashCode();
int index = hashValue & (n-1);
if: bucket[index] == null;
   a) bucket[index] = put(key, value);
else:
   int i = 0;
   while (i<bucket.list.size()) {
      a) if: hashValue != bucket.list[i].hashValue; // move to next 'key
         - value' pair.
      b) else if: hashValue == bucket.list[i].hashValue &&
         key.equals(bucket.list[i].key);
      bucket.list[i] = put(key, value); // over write value for
         existing key.
      c) else: i++;
      bucket.list[i] = put(key, value); // adding new pair
   }


The concept of hashing Put() method is illustrated by an example as shown in Figure.3.12.

Suppose there is a player named Jones and he scores 99 in a game activity. Assume key-“WARD” and value is 99. Now to place the key value pair into the bucket (array) and to generate hash code, the hash code generator provide the hash value = 4 called hash mapping. So the key value should Put at the memory location 4 in the bucket. Now in the bucket at index value 4, the (key, value) i.e. (“WARD”, 99) will be put but at this location the key and value pair already exist. Hence this will traverse and search for NULL tail value in the link list. For the retrieval of data Get() method is used.
The concept of Get() method is shown in Figure 3.13.

![Figure 3.13 Concept of Hashing Get() Method](image)

The algorithm of Get (key) method is as follows:

Algorithm: Get (key)

---

Input: Key value
Output: Data item
Method:

```
int hashValue = key.hashCode();
int index = hashValue & (n-1);
Object value = null;
if: bucket[index] == null;
    a) return value; //no key value pair so returned value would be null.
```
else:
    inti = 0;
    while(i<bucket.list.size()) {
        a) if: hashValue != bucket.list.[i].hashValue; // move to next 'key -
        value' pair.
        b) else if: hashValue == bucket.list.[i].hashValue&&
        key.equals(bucket.list.[i].key);
        value = get(key); // value will be assigned from existing
        key.
        break; //breaking while loop.
        c) else: i++;
    }
return value; //If no match found value would be null else returned the
assigned value in step (b)

3.7.2 Framework of Privacy Preserving Classification using Hashing

The framework proposed for the implementation of hashing based privacy preserving
classification is illustrated in Figure.3.14. This is proposed that the generated data stream or
a collection of data stream should be applied to data stream mining system in two ways
simultaneously. One in which original data stream (D) is fed to the data stream mining
system and another when data stream mining system is fed after preserving privacy through
hashing based technique. The classification results of original data stream (R) and privacy
preserved DS (R') is then compared. The comparison of two different results observed in
this way will finally provide the performance parameters of the data stream.
Figure 3.14 Framework of privacy preserving classification using proposed hashing based technique
3.7.3 Hashing based Privacy Preservation

Hashing based Privacy Preservation provide simple and easy user interface. It uses fast and robust hashing for preservation which can lookup data association with best $O(1)$ and average $O(n)$ time complexity. A new hash based privacy preservation technique designed as shown Figure.3.15.

Figure. 3.15 Flowchart to design privacy preservation technique using Hashing.
3.7.4 Recuperate Original Data

Reverse reengineering is applied to recover the original data streams. Reverse engineering helps to recover the original data stream from a secured and filtered stream. The process of reverse reengineering is shown in Figure 3.16. The filtered data stream obtained in the output of hashing based privacy preservation serve as an input in the reverse reengineering process. The output of reverse reengineering is recovery of the original data stream.

![Flowchart to recuperate original data stream](image)

Figure 3.16 Flowchart to recuperate of original data stream
3.7.5 Software Developed for Hashing based Privacy Preservation

To implement the algorithm designed for hashing based privacy preservation a software was developed in JAVA-8. JAVA has been used because of its ease of implementation. As JAVA programming is used, the developed software is platform independent and portable. The software has been implemented on a system with Microsoft Windows 10 operating System, intel X64 bit quad core i5 processor having 8 GB RAM. Open Source Tomcat 7 web server has been used for collection of data stream from http server.

3.7.6 Data collection used in the Hashing based Model

The experiment has been carried out on the standard adult data stream obtained from UCI repository with total 32,564 records. The privacy preserved data stream obtained from the designed hashing based model is subsequently classified using massive online analysis tool.

3.7.7 Implementation of the Privacy Preservation using Hashing

There are three choices of selection of data stream HTTP, Data stream and Local file system. The sequential process for the implementation use of the hashing based technique is represented in the figures. Choice 1: The data stream is collected form HTTP Server. Figure.3.17 shows initial user interface of the software and selection of data stream from HTTP server.
Figure. 3.17 Initial user interface of the software and selection of data stream from HTTP server

Figure. 3.18 Fetching the data stream from HTTP server
Fetching the data stream from HTTP server is shown in Figure.3.18. After line by line fetching of data stream privacy preservation using hashing will be applied on selected attributes as shown in Figure.3.19.

Choice 2: Data stream has been imported from the data stream server. Hence a socket server has been developed for establishing connection with the software as shown in Figure.3.20.
Choice 3: Data stream has been imported from the local file system. Figure 3.21 depicts the Selection of data stream from local file system.
Figure. 3.21 Selection of data stream from local file system

3.7.8 Implementation of the Recovery of Original Data Streams

Recovery of original data stream i.e. reverse reengineering helps to recover the original data streams. The original data streams is recovered from secured and filtered data streams. Reverse reengineering is a reconstruction-based techniques where the original distribution of the data is reassembled from the privacy preserved data.
Recovery of original data stream by using reverse reengineering technique and process of reverse reengineering are signified in Figure.3.22 and Figure.3.23.