TÉCNICO LISBOA

Detection of Fraud, Abuse and Cost Inefficiencies in the National Healthcare System

Francisco Guerreiro Gomes Pedreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Rui Fuentecilla Maia Ferreira Neves

Prof. Nuno Cavaco Gomes Horta

Examination Committee

Chairperson: Prof. José Carlos Martins Delgado

Supervisor: Prof. Rui Fuentecilla Maia Ferreira Neves

Members of the Committee: Prof. Alexandre P. Francisco

October 2016

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible.

I would also like to thank Joana Pedreira for helping improving the quality of the pictures presented in this work.

I would also like to acknowledge my dissertation supervisors Prof. Rui Neves and Prof. Nuno Horta for their insight and sharing of knowledge that has made this Thesis possible.

Abstract

In today's world healthcare fraud and abuse has taken enormous proportions. Every year millions of dollars are lost in the healthcare system due to fraud. It is therefore important to develop systems that can efficiently combat and prevent this behaviour. This dissertation proposes a solution for the problem of detecting fraud and abuse in the healthcare system. The chosen approach can be divided into three main phases. In the first phase, a data generator is used to create a dataset composed by prescriptions and geographical distributions. In the second phase, two different algorithms are used to detect fraudulent behaviour, namely a geo-location based and a rule based algorithm. These algorithms use a supervised learning approach and are evaluated using a variety of metrics, specifically accuracy, confusion matrix, precision and recall, and K-fold cross validation. In the last phase, the output of both algorithms is analysed and the fraudulent claims are sent for further investigation. Using the prescriptions dataset the obtained value for the recall, when using 10-fold cross validation, is of 0.33 which is not a very high value. However using the geographical distributions dataset a recall of 0.69, again using 10-fold, is obtained which is a good result for this measure. A novel approach is taken in this work by designing and implementing a data generator that can create data for a plethora of algorithms. Using this approach allows for cost reduction since only a few cases, the ones identified by the algorithms as fraudulent, will be sent for further analysis, resulting in a cheaper and more efficient service.

Keywords

Healthcare Fraud, Fraud, Abuse, National Healthcare System, Detection of Fraud

Resumo

No mundo actual a fraude e abuso na área da saúde atingiu proporções enormes. A cada ano que passa milhões de dólares são perdidos no sistema nacional de saúde devido à fraude. É portanto importante desenvolver sistemas que possam combater e prevenir de forma eficiente este comportamento. Esta dissertação propõe uma solução para o problema da detecção de fraude e abuso no sistema nacional de saúde. A abordagem escolhida pode ser dividida em três fases principais. Na primeira fase, é utilizado um gerador de dados de modo a criar um *dataset* constituído por prescrições e distribuições geográficas. Na segunda fase, são usados dois algoritmos distintos para detectar comportamento fraudulento, nomeadamente um algoritmo baseado em geolocalização e um baseado em regras médicas e farmacéuticas. Estes algoritmos usam *supervised learning* e são avaliados utilizando várias métricas especificamente *accuracy*, *confusion matrix*, *precision* e *recall*, e *K Fold cross validation*. Na última fase, o *output* de ambos os algoritmos é analisado e os casos fraudulentos são enviados a especialistas, para que seja realizada uma investigação mais aprofundada. Usando o *dataset* das prescrições, o valor obtido para o recall, utilizando *10-fold cross validation*, é de 0.33 cujo valor não é muito alto para esta métrica. No entanto, utilizando o *dataset* das distribuições geográficas obtém-se um *recall* de 0.69, mais uma vez com *10-fold cross validation*, o que é um bom resultado. Uma nova abordagem é tomada neste trabalho, projectando e implementando um gerador de dados que tem a capacidade de criar dados para um grande número de algoritmos. Esta abordagem permite que haja uma redução de custos visto que apenas uma porção dos casos, os que forem detectados como fraudulentos, serão enviados para investigação futura, tornando o serviço mais barato e eficiente.

Palavras Chave

Fraude no Sistema de Saúde, Fraude, Abuso, Sistema Nacional de Saúde, Detecção de Fraude

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

NN	Neural Network
SSB	Social Security Body
SVM	Support Vector Machine
ECM	Evolving Clustering Method
EFD	Electronic Fraud Detection
HIC	Health Insurance Commission
HICN	Health Insurance Claim Number
SDLE	Sequentially Discounting Laplace Estimation
NMF	Nonnegative Matrix Factorization
DG	Data Generator
NPI	National Provider Identifier
AWS	Amazon Web Services
GB	Gigabyte
SSD	Solid State Drive
GiB	Gibibyte
AMI	Amazon Machine Image
SSH	Secure Shell
RAM	Random Access Memory
CPU	Central Processing Unit

IP Internet Protocol

OS Operating System

1

Introduction

Contents

In the modern world, fraud is present in many fields such as banking, finances, business practice and healthcare. One example is identity theft [1], in which criminals use stolen identities to steal money from bank accounts, claim eligibility for services or hack into networks without previous authorization. Another example is financial fraud. Millions were lost to this kind of fraud as shown in [2]. A specific example is when criminals hold money or other property from the company, causing loss and damage [3]. Credit card is also a very popular target for criminals as shown in [4]. The credit card industry is estimated to lose about $2 billion a year globally due to fraud [5]. Here, credit card information is stolen and used to pay for services or goods in behalf of the original owner. In 2004 alone, in the U.S.A, 800 million dollars were lost in fraud.

Inside companies there are also other types of fraud committed which involve business practices. In [6] it was reported that in two years, 8.3 million dollars were overcharged by a large number of taxi drivers who deceived their costumers by arbitrarily modifying their taximeter.

In the healthcare system, fraud generates a loss of over $30 billion annually to healthcare insurance frauds [7].

As has been seen, there are several kinds of fraud, mainly credit card, financial, business practice and, last but not least, healthcare fraud.

The last will be the matter of this work.

1.1 General Goals

The main goal of this work is to implement an application that can detect fraud in the national healthcare system. In order to accomplish this goal, the following objectives must be fulfilled:

- Design and implement a data generator.
- Define two algorithms to be used in the detection of fraud.
- Analyse the output of the algorithms and decide which claims to send to further investigation.

1.2 Thesis Structure

This thesis is organized as follows:

- Chapter 2 presents and discusses the existing literature;
- Chapter 3 details and analyses the architecture of the developed solution;
- Chapter 4 presents the metrics used to test the proposed solution and the results obtained from applying the chosen metrics

- Chapter 5 presents the conclusion of this work along with some improvements that can be added in a future work.

2

Related Work

Contents

Health care produces massive amounts of information in today's world. And in order to process this data automated algorithms must be used, since humans can't process it timely. Some examples of developed algorithms can be found in [8], [9] and [10].

Having this in mind, and knowing that the problem at hands is detecting fraud, statistical methods were developed to help tackling the problem. They are divided in two main methods: supervised and unsupervised.

The supervised methods are used to give the system the ability to recognize which cases are legitimate and which are fraudulent. In order to achieve this, these methods require prior knowledge, that is, they require all the cases in a training dataset to be labelled, as either legitimate or fraudulent, by experts. This way, the system is trained to understand how to classify a new case and so it will know to which class this new case belongs when it is introduced in the system.

The unsupervised methods are used to cluster similar cases into groups. This allows the system to find outliers in the data since they will be aggregated in the same group. Unlike supervised methods, unsupervised do not require previous data, but because of this they cannot detect known cases of fraud. Hence it is important to use this type of methods along with the supervised ones to ensure that the largest number of fraudulent cases are detected.

An important aspect in fraud detection is understanding who might be involved in the criminal activities. According to the authors of [11] there are three main participants: service providers, insurance subscribers and insurance carriers. Service providers can be doctors, hospitals, ambulance companies or laboratories. Insurance subscribers are patients and patients' employers. Insurance carriers include governmental health departments and private insurance companies.

Among all the mentioned participants, service providers account for the majority of fraud committed.

Before introducing the main types of fraud and their perpetrators, a brief explanation of machine learning will be presented.

[11] was used as a reference to write a big part of this section since it was one of the most cited papers in the literature, containing very good information on the topic of this thesis.

2.1 Machine Learning in Healthcare

Healthcare fraud detection is a complex task. It comprises the analysis of large datasets with the ultimate goal of discovering cases with similar features or detecting similar patterns and aggregate them in groups. This can be achieved by using Classification and Clustering techniques. A Clustering technique is used to group cases with similar characteristics, which makes it easier to find outliers in the dataset. A Classification technique is applied to separate cases into different classes which can then be used to classify the data (for example, in healthcare fraud detection, two possible classes may be

legitimate and fraudulent). Both techniques are useful but both have flaws that the other bridges. In the Clustering technique we have the evident problem of not being able to separate cases into classes and in the Classification one we can not detect new types of fraud, i.e., classes that are not present are not recognized. This problem is solved by the Clustering technique which finds all the outliers and groups them, thus, effectively finding new types of fraud whenever they are present in the dataset. Therefore, as was seen above, a hybrid solution is necessary in order to find the maximum number of frauds committed, whether it is a new or a known type of fraud.

Machine Learning is a subfield within Artificial Intelligence that studies the creation of algorithms that allow a computer to learn how to perform tasks from data instead of it being explicitly programmed. There are two main groups of algorithms: supervised and unsupervised.

The supervised algorithms, as mentioned above, use labelled data to determine if a case resides within a certain class, the classes being in this case fraudulent or legitimate. This is a specific type of Classification technique since it uses labelled data to classify a case accordingly.

The unsupervised algorithms do not use labelled data. Instead they aggregate cases with similar features, creating groups where every element is related. This is a Clustering technique since clusters are created where each element has the same set of features.

As demonstrated above, Machine Learning algorithms are very important to solve the problem of detecting fraud in the healthcare system. These techniques are used extensively in fraud detection and, in the following sections, the most common will be discussed more in depth.

2.2 Types of Fraud

In this section the main types of fraud will be introduced. There are three major entities involved in the criminal activities, namely, service providers, insurance subscribers and insurance carriers. However, service providers account for the vast majority of the fraud committed.

Bellow, a list of the various types of fraud, as described in [11], will be presented.

- Service providers' committed types of fraud:

1. **Billing unperformed services**: billing services, which were not performed, to the insurance company, in order to obtain profit;
2. **Performing unnecessary medical services**: performing medical services which are not required by the patient, in order to generate insurance payments for those services;
3. **Falsifying non-covered treatment**: falsifying a non-covered treatment as a medically necessary covered treatment in the interest of obtaining insurance payments;
4. **Unbundling**: billing each stage of a procedure as if it were a separate treatment;

5. **Upcoding**: billing more costly services than the ones actually performed (for example, classifying a patients' illness into the highest possible treatment category in order to claim a higher reimbursement);

6. **Tamper with the diagnosis**: tampering with the patients' diagnosis and/or treatment histories in the interest of justifying tests, surgeries, or other procedures that are not medically necessary;

- Insurance subscribers' committed types of fraud:

1. **Forging records of eligibility**: forging records of employment and/or eligibility in order to obtain a lower cost insurance;

2. **Filing claims for unreceived services**: filing claims for medical services which are not actually received;

3. **Illegal use of cards**: using other person's coverage or insurance card to illegally claim the insurance benefits;

- Insurance carriers' committed types of fraud:

1. **Falsification**: falsifying reimbursements;

2. **Forgery**: forging benefit/service statements;

In addition to the types described above, there is another newly emerging type called conspiracy [11], which involves more than one party. An example is a patient and a physician fabricating medical service and transition records in order to defraud the insurance company to whom the patient subscribes.

Bellow a graphical representation of some types of fraud will be presented to allow for a better understanding.

In Figure 2.1 we have an example of unperformed services being billed to the insurance company. This happens when a service provider introduces new services in the bill, which were not actually performed on the patient, in order to obtain extra income from the insurance company. In Figure 2.2 we have a scenario in which tests that are not necessary to the patient are conducted and billed with the purpose of, once again, obtain extra revenue.

Another type of fraud is the one where a non-covered treatment is falsified as a medically covered and necessary treatment in the interest of obtaining profit. An example can be seen in Figure 2.3.

Unbundling is a type of fraud in which every step of a medical procedure is billed as a separate treatment, as seen in Figure 2.4.

Upcoding is a type of fraud where more costly services are billed, that is, the service provider bills services which are more expensive than the ones actually performed. Figure 2.5 shows an example.

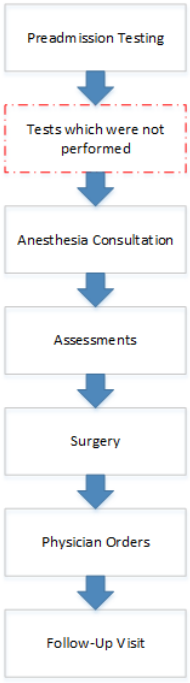

Figure 2.1: Billing unperformed services

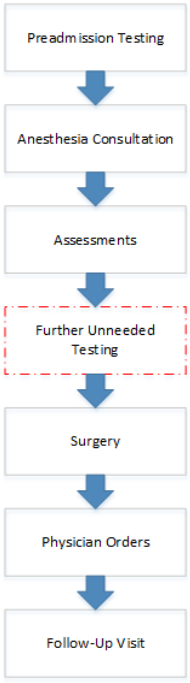

Figure 2.2: Performing unnecessary medical services

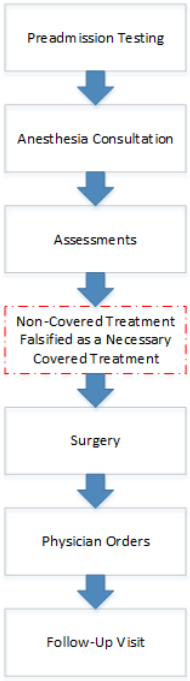

Figure 2.3: Falsifying non-covered treatment

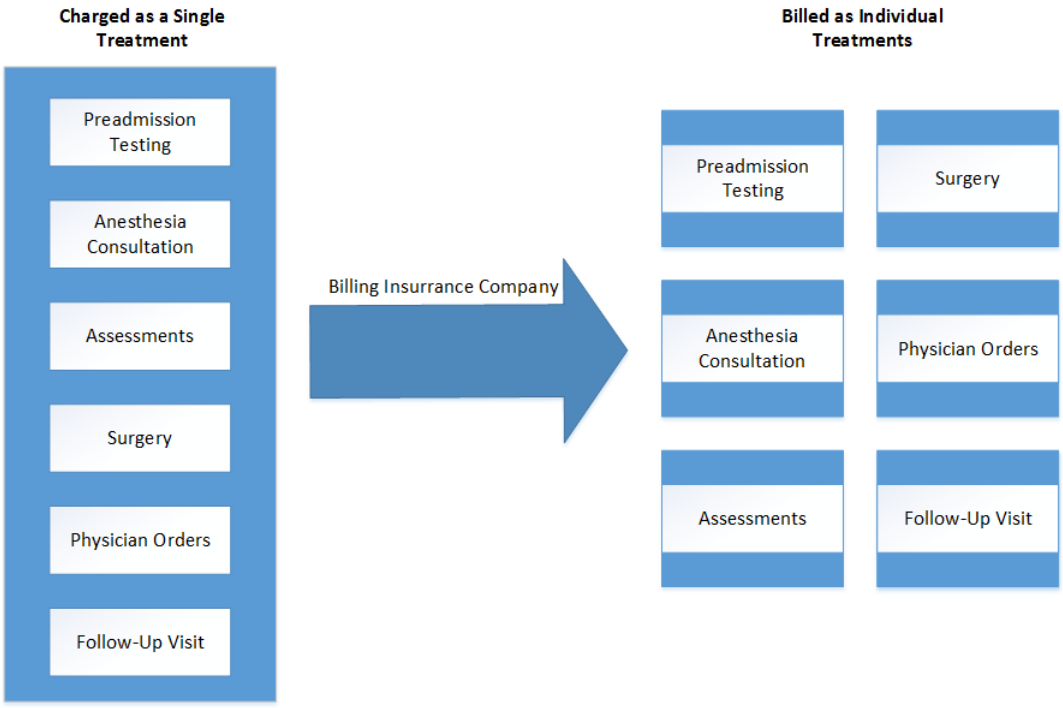

Figure 2.4: Unbundling

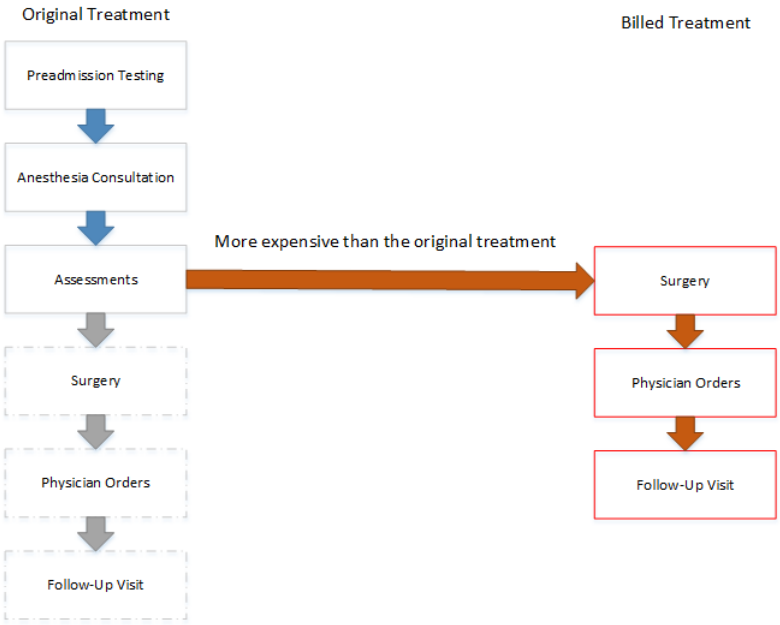

Figure 2.5: Upcoding

2.3 Data Preprocessing

Raw data is mostly composed by insurance claims. These involve the participation of an insurance subscriber and a service provider. Claim data has two characteristics that make it valuable.

Firstly, it contains a wide set of attributes that describe the behaviour of the involved insurance subscriber and service provider, thus proving useful in detecting the types of fraud committed by both parties.

Secondly, each claim usually contains unique identifiers for the involved service provider and insurance subscriber. By using the identifiers to associate different claims, a general view of a service provider's behaviour over time and across different subscribers can be obtained, as well as a general view of an insurance subscriber's behaviour over time and across different service providers. These views can then be used to detect and identify fraud committed by service providers and insurance subscribers therefore making the process more straightforward.

Since there is a need to use statistical methods, the raw data must be transformed into a format that is both recognizable and usable by the methods. This process is called preprocessing. Preprocessing takes roughly 80% of the total time in fraud detection. Below, the key aspects involved in preprocessing are presented:

- **Goal setting**: this step focuses on identifying and prioritizing the types of fraud on which the detection should centre on.
- **Data cleaning**: there are numerous ways to represent the same data. For example, a physician's

unique identifier can be represented by different digit formats [11], a disease may be represented by a text description but also as a diagnostic code [11]. This is why there must be a way to standardize the data, that is, clean it.

- **Handling missing values**: missing values are normal in health care data. These values are not collected due to a variety of causes, some of them being omission, irrelevance, excess risk, or inapplicability in a specific clinical context [11]. But it has been proven, as stated in [11], that a better performance is achieved if the missing values are properly handled instead of removing all the cases with missing values from the dataset. With the intention of solving this problem, two methods were created - hot-deck imputation and regression imputation.

  Hot-deck imputation fills in the missing values in an incomplete case using values from the most similar (have more equal variables with non-missing values) and more complete cases of the same dataset.

  In regression imputation, a regression model is fitted for each variable with missing values, with other variables without missing values as covariates. Then, the missing values are replaced by the predicted values from the regression.

- **Data transformation**: the raw claim data must be transformed into new and more relevant views. A flattened table format of data is required by most statistical methods, thus it is used in this step. In the new table, each row represents a case for training or testing a statistical model, and each column contains the values for an attribute across cases. For example, if we want to detect the fraud committed by insurance subscribers, patients for instance, then each row in the new dataset should correspond to a different patient, and each column to an attribute describing either a demographic characteristic (sex, date of birth) or an aspect of a patient's usage of the health care system (total number of top service codes and the average charge per service, for a certain time period). Since we need a time period to measure the patient's usage of the health care system, we need to establish it before we can create the new dataset.

- **Feature selection**: delineate new features from the original attributes in order to maximize the discrimination power of the statistical method in separating fraudulent and legitimate cases.

2.4 Supervised Methods

As was shown before, supervised methods are used to effectively detect fraudulent behaviour in the healthcare system. The authors of [11] mentioned neural networks, decision trees, fuzzy logic and Bayesian Networks as the most common supervised methods.

A **Neural Network (NN)** is a model composed by neurons (computational units). In this model, a neuron receives several inputs which have a strength (x_i) and a weight (w_i). An activation function is then calculated by the following Formula 2.1.

$$\sum_{n=1}^{k} x_i * w_i \tag{2.1}$$

Then we check to see if the result is greater than or equal to a theta. If it is, the output will be 1, otherwise it will be 0.

One example of a NN is a multi-layer perceptron (MLP) which was used by [12] to detect fraud. The NN initially had four classifications and was trained using the backpropagation algorithm. This algorithm starts by getting a training example and calculating the outputs. Since it is using historical data, meaning the expected results are known, it uses this information to calculate the errors (difference between the calculated and known results). After obtaining the errors, the algorithm goes back to the inputs and adjusts the weights in order to minimize the error.

This algorithm has also been used by [11] Hall, which used a NN to detect service providers' fraud based on discriminating features identified by domain experts and Cooper, which used a NN to identify fraudulent medical claims submitted to a Chilean health insurance company.

A **decision tree** is a tree flowchart structure in which each internal node represents a decision to be made based on an attribute (if a coin flip comes up heads or tails for example), each branch represents one possible outcome of the decision and each leaf node represents a class label which defines the final decision after all the attributes have been computed. The paths from root to leaf represent classification rules. Decision trees were extensively used for health care fraud detection [13], [14], [15].

Decision trees are used in this sort of problem due to the easy interpretation of the results and ability to handle missing values which, as seen before, is a major problem in the health care system.

Fuzzy logic is an approach in which, instead of using only 0 and 1 (true or false as in standard logic), it uses a degree of truth that includes any real number between 0 and 1, i.e., it can go from 0.0 (absolute false) to 1.0 (absolute true). So instead of saying something belongs to the "legitimate" group with 100% certainty, we say that it belongs with a 0.8 degree of truth. This is useful since in the field of fraud we can never be certain that something is legitimate or fraudulent.

Bayesian Networks are graphical models that allow reasoning under an uncertain domain. In this model, the nodes represent a set of random variables from the domain (which may be discrete or continuous) and a set of directed arcs connect pairs of nodes representing the direct dependencies between variables. These are useful for solving our problem for the same reason stated in the above paragraph.

In [16] the authors used **similarity learning** to automatically find a measure of how similar two entities were in relation to an aspect of interest such as their fraud status. In other words, the algorithm

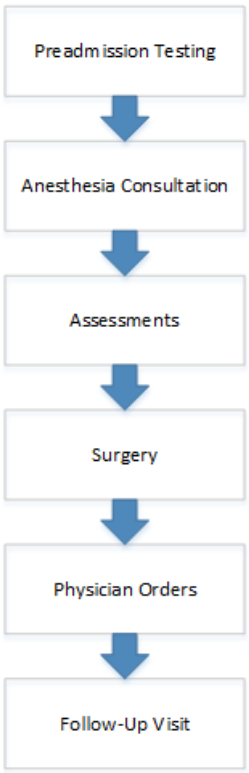

Figure 2.6: Clinical pathway of cholecystectomy.

Figure 2.7: A pathway that indicates suspicious activity since it does not follow the normal order and it has several follow-up visits.

tries to find similarities between known fraudulent cases and new ones. This approach deals with the fact that very often it is not possible to have enough cases of known fraud to extract a valid rule.

As seen in [17] a **clinical pathway** has been used as a supervised method. Here, a number of tasks is performed sequentially in a specific order. The primary objective of this method is to enable best practice by having the medical staff performing the tasks in the right order and thus without repeating work or wasting valuable resources. A way of detecting fraud using this technique is, for example, to see if the standard order is not followed or if a certain task is performed more than once. An example can be seen in Figures 2.6 and 2.7, where in Figure 2.6 we have the normal clinical pathway for treating cholecystectomy and in Figure 2.7 we have an abnormal situation in which the normal order is not followed and a task is repeated several times, which might indicate a fraudulent behaviour.

The authors of [18] implemented a **system based on rules** that checks for missing or invalid data and also the medical validity of the prescription. The system was developed to address a problem that existed at the time which was that the prescriptions were hand-checked, very few of them were checked and they mostly just checked the validity of the financial data. There were other problems like human errors, incomplete prescriptions and the fact that a prescription could only be identified as 'suspicious' if

it was checked against others prescribed by the same doctor, submitted to the same pharmacy by the same patient.

So in order to solve this problem, a set of rules was implemented. It was organized into two groups, namely administrative and medical rules.

Administrative rules were used to detect missing or invalid data in the initial phase, that is, when the data was introduced into the system, while medical rules were used to check for the medical validity of the prescriptions. In order to construct these rules, ATC (Anatomical Therapeutic Chemical) coding rules based on medical and pharmaceutical sciences were applied. So for example an administrative rule [18] checked whether the sum of the healthcare provider's contribution for each medicine was less than the total sum entered in the prescription. However, a medical rule was, for example, one that checked whether there was a correspondence between prescribed drugs and the diagnoses that appeared in the prescription.

The following rules were used in the application:

— Administrative rules:

1. **Missing diagnoses:** each prescription must include at least one diagnosis for the prescribed drugs;
2. **Violation of the 5-day execution period:** the execution date of the prescription should not exceed its issue date by more than five days;
3. **Missing insured person's code, pharmacy code or doctor's code:** the prescription must include the insured person's code, pharmacy code and doctor's code;
4. **Lack of membership of the SSB:** doctor, patient or pharmacy are not members of the Social Security Body (SSB);
5. **Execution date precedes issue date** the execution date of the prescription cannot precede the issue date;
6. **False participation percentage:** each diagnosis has a predefined participation percentage for the SSB in the price of drugs. This rule checks if a prescribed drug has less participation percentage than the corresponding predefined participation percentage of the diagnoses that appear in the prescription;
7. **Prescribed drugs are overpriced:** this rule checks the drug prices in the prescription against the official price list;
8. **Provider contribution less than in the prescription:** the sum of the healthcare provider's contribution for each medicine is less than the total sum entered in the prescription;

— Medical rules:

1. **No correspondence between diagnosis and drug:** the correspondence of each prescribed drug is checked against the prescription's diagnosis;
2. **No correspondence between diagnosis and doctor speciality:** the correspondence of each diagnosis in the prescription is checked against doctor speciality;

Violation of these rules did not automatically imply that a prescription was suspicious, but it was a strong indication that the prescription should be investigated by a third party.

There was a second degree of classification in which the testing rules were categorized as obligatory and warning. Obligatory rules could not be violated for the prescription to be definitively entered. In the warning category the user was warned before definitively entering the prescription.

Before the prescription was committed to the database, the following checks were performed:

1. If the date of issue and the date of dispensing have been entered;
2. If the prescription's total value, the patient's and the SSB's contribution have been entered;
3. If the prescription has at least one prescribed drug with all its corresponding details (items prescribed, items collected, patient's contribution and price);
4. In case that there exists more than one drug in the prescription, whether their corresponding details have been entered;

This kind of system is very useful in healthcare fraud detection since a problem that may arise when trying to detect fraud is that labelled data may not exist. Since the system uses rules and rules only, it is not dependent on experts to label the data and therefore allows for greater flexibility.

Support Vector Machine (SVM) was also used in the literature, as seen in [19]. Support Vector Machine (SVM) is a classification technique in which individual records are identified as normal or fraudulent. The system is trained so that it can detect a boundary between normal and fraudulent records. After obtaining a boundary, whenever a new record is processed, it will be compared with the boundary and thus it will be identified as being in one of two bounds: normal or fraudulent.

SVM has two main phases: training and classification. In the training phase, also called preprocessing phase, two class labels are defined, one for the normal case and another for the exception, i.e., the fraudulent case. The second step consists in classifying the dataset claims into the two defined classes. In the final step, the support vectors are chosen (the vectors that define the margin of the maximum marginal hyperplane) [19]) and the maximum marginal hyperplane (a hyperplane which separates two groups of points and is at equal distance from the two) is found. This will allow for the separation of the claims into one of the two classes.

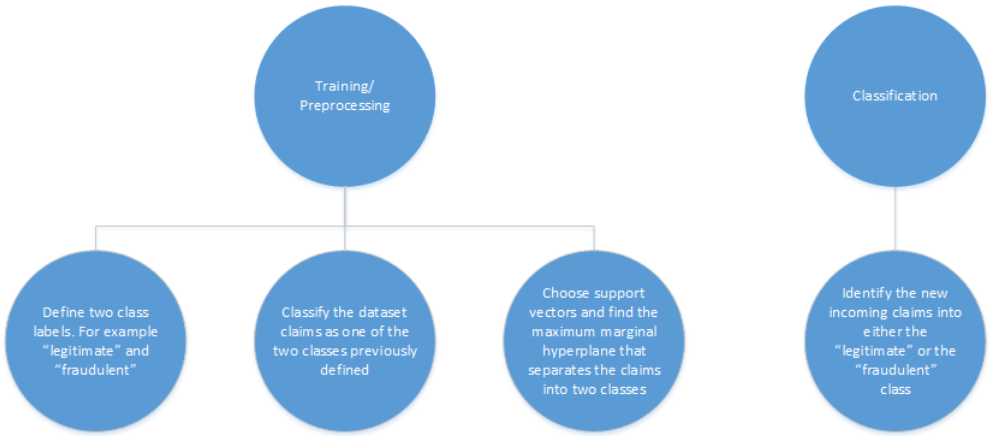

Figure 2.8: SVM components.

In the classification phase, that is, after the training, the system will be ready to determine to which class a particular claim belongs to. In Figure 2.8 an overview of the discussed phases of SVM is presented.

2.5 Unsupervised Methods

Unsupervised methods are used to cluster data into groups. Each element of the group shares similarities with all the other elements in that same group. Different groups represent different characteristics. This is a good way of organizing the data into subsets to facilitate prior processing and is also crucial to identify new groups that have never been found (which in the case of this thesis represents a new type of fraud).

Several methods have been applied in the healthcare fraud field. In this section some methods will be discussed.

One example is given in [7] in which **Evolving Clustering Method (ECM)** is used as a clustering technique.

Evolving Clustering Method (ECM) is used to cluster dynamic data. Dynamic data is one that is in constant change as time goes by. When a new data point is fed to the algorithm, ECM clusters it by changing the position and adapting the size of the cluster. Each cluster has an associated parameter, known as radius, which determines the boundaries of that cluster. Initially it is set to zero. As more data points are added, the radius is incremented. An additional parameter exists, called distance threshold (Dthr). Depending on the value given, several small clusters may be created (small Dthr) or, if the value is large, a small number of large clusters will be created.

This algorithm is useful in the healthcare fraud detection field since in this scenario data is always being created and is in constant change, thus creating a continuous flow of information and therefore

the need to cluster dynamic data.

In [11] the authors mentioned the **Electronic Fraud Detection (EFD)** system. Electronic Fraud Detection (EFD) is an expert system, developed to assist in the detection of service providers' fraud. It has several phases. In an initial phase, discriminating features, named "behavioural heuristics", are defined by experts. Then the information gain for a provider is calculated, measuring how different the distribution of the provider is from that of all the peers taken together. Using that information, the system plots a graphic where the providers are points in a 2D space with one axis representing the information gain and the other the total dollars paid to the providers. As a result, the points in the frontier will represent the ones with the biggest difference in relation to their peers and also the ones with more dollars received, making them suspicious. The system then proceeds to highlight these points and the corresponding providers are considered to be suspicious.

SmartSifter is also referred in the literature [11]. It was used to detect outliers in the pathology dataset provided by the Health Insurance Commission (HIC) of Australia. The system uses a probabilistic model to represent the underlying mechanism. This model uses a histogram to represent the probability distribution of categorical variables (variables that can take one value of a limited, and usually fixed, number of possible values). For each bin of the histogram a finite mixture model (a probabilistic model used to represent the presence of sub-populations within a bigger population) is used to represent the probability distribution of continuous variables.

When a new case is introduced in the system, SmartSifter updates the probabilistic model by employing a Sequentially Discounting Laplace Estimation (SDLE) [20] to calculate the probability distribution of the categorical variables. SDLE is a variant of the Laplace Law, obtained by modifying the law so that it is run on-line and it can discount the effect of past examples gradually. In the next step an SDEM (Sequentially Discounting Expectation and Maximizing) [20] algorithm is used to learn the probability distribution of the continuous variables. A score is given to the new case, measuring how much the probabilistic model has changed since the last update. A high score indicates that it may be an outlier.

In [21] a different approach is used. The authors use **geo-location** to identify fraud. There are many ways this information can be used to commit fraudulent acts. As shown in [21], for example if a Medicare/Medicaid beneficiary's identification is stolen, the thief can use this identification in a service provider close to himself but far from the beneficiary. This can be seen as a type of fraud since the majority of Medicare/Medicaid subscribers are poor or nearly poor and thus cannot afford to travel a long distance, usually choosing the closest service provider available. Another example is that a service provider and a beneficiary make an agreement that the beneficiary travels a long distance to get a kickback (an illicit payment made to someone in return for facilitating a transaction or appointment), and the service provider can bill unnecessary services to cover the expense and earn extra money.

The developed geo-location based algorithm has three main phases: preparation, data preprocess-

ing, and analysis. In the preparation phase, since the original dataset only contained the beneficiaries' living county and the service providers' locating state, in order to calculate the distance between beneficiaries and service providers, their latitude and longitude had to be known. This information was collected from the US census website. Next, their location was mapped to the used Medicaid dataset according to the SSA code (state code from claim) of each county and state. After the conclusion of this step, every claim had both the beneficiary's and the service provider's latitude and longitude information.

In the data preprocessing phase, the Euclidean distance between beneficiaries and service providers was calculated using the information obtained in the previous phase. Since the availability of the service and the payment amount differed with the type of disease, the diagnose was used as the control variable. Thus the claims were classified based on their principal diagnoses. The three most common diagnoses were attained, namely Pneumonia, Rehabilitation procedure and Septicemia. Claims from the three diagnoses were extracted in order to form three separate datasets where each dataset contains claims with the same principal diagnose. Payment amount and distance were used in the clustering model.

In the analysis phase, the payment amount and the distance are analysed separately. In the final step, the model is used to cluster the three datasets and the analytic results of all the experiments are compared. In [22] the authors used **Nonnegative Matrix Factorization** (NMF) to cluster "medical treatment items". These items may represent a medicine or a medical measurement. Nonnegative Matrix Factorization (NMF) was chosen since it respects the property of non-negativity, which is inherent to the healthcare data. This allows for a better understanding of the obtained results, since negative values would be hard to interpret for the reason stated previously.

A set of patients P can be expressed as an N x M matrix called Mat, where N is the number of patients in P and M is the number of "medical treatment items" in the healthcare dictionary. Each row, Mat_j, of Mat is an encoding of a patient in P and each entry, mat_ij, of the vector Mat_j, represents the significance of the "medical treatment item" i in Mat_j, where i ranges across the "medical treatment items" in the dictionary.

The problem to be solved is the one of finding a low rank approximation of the Mat matrix in terms of some metrics by factoring Mat into the product of lower dimensional matrices W and H (WH). Each column of W is a basis vector containing an encoding of a type of "medical treatment item" from Mat and each column of H contains an encoding of the linear combination of the basis vectors that approximates the corresponding column of Mat. The dimensions of W and H are m x k and k x n respectively, where k is the reduced rank or the selected number of types. Usually k is chosen to be much smaller than n, but the way it is actually calculated is k $<<$ min(m, n). The appropriate value of k depends on the application and also on the nature of the data itself.

In order to obtain an approximation of Mat, the original matrix containing all the measurements, a pair (W, H) is computed as to minimize the Frobenius norm (defined as the square root of the sum of the

absolute squares of its elements [23]) of the difference Mat-WH. Then, the minimization problem can be stated as

$$min_{W,H} ||Mat - WH||_F^2 \tag{2.2}$$

with $W_{ij} > 0$ and $H_{ij} > 0$ for each i and j.

The matrices W and H are not unique. Usually H is initialized to zero and W to a randomly generated matrix where each $W_{ij} > 0$ and the initial estimates are improved or updated with the following iterations of the NMF algorithm.

The way NMF is used to detect fraud can be seen in an example described in [22]. In this paper, according to the knowledge of medical experts, in one month, a patient would not use two medicines with the same effect. So each cluster element was complementary to the other in terms of medical effectiveness. All the medicines are checked per cluster and two medicines, "Yunnan baiyao powder" and "Xinhuang pian", with similar medical effectiveness are found in one cluster. Further experiments with different k values are conducted but the medicines always appear together in one cluster. It was considered that the patients who bought the two medicines in one month were "fraud-suspicious". When consulting experts with these results, it was proven that indeed some of the patients were, in fact, committing fraud.

2.6 Chapter Summary

In this chapter the most relevant literature was presented and discussed. Table 2.1 displays a list of the more important papers presented throughout this chapter. These introduced critical concepts and innovative systems, such as neural networks, bayesian networks, a system that uses pharmaceutical and medical rules and another based on geo-location. The two last mentioned systems are used in the architecture of the solution since they were well explained in the literature and did not need historical data in order to function. All the other mentioned algorithms could have been chosen, but a decision had to be made, since there was a limited amount of resources that could be dedicated to the implementation of the algorithms. For these reasons only some papers were included, as to highlight the ones that have made a greater contribution to the overall chapter.

Table 2.1: A review of the most important papers used in this thesis

Chapter	Reference	Date	Period of simulation	Market Tested	Used Algorithms	Supervised	Unsupervised	Survey	Best Results
Related Work	[11]	2008	-	-	-	No	No	Yes	-
	[17]	2006	July 2001 to June 2002	Bureau of National Health Insurance	Clinical Pathway	Yes	No	No	Detection rate of 69%
	[18]	2009	Year 2006	T.S.A.Y	Pharmaceutical and Medical Rules	Yes	No	No	-
	[7]	2015	-	-	SVM with ECM	Yes	Yes	Yes	-
	[12]	1997	-	HIC of Australia	Neural Networks	Yes	No	No	80.9% agreement rate
	[20]	2004	-	-	SmartSifter	No	Yes	No	-
	[22]	2011	-	Health Insurance Management Center of Xiamen	Nonnegative Matrix Factorization	No	Yes	No	-
	[21]	2013	-	Medicare	Geo-location	No	Yes	No	-
	[8]	2014	-	eHealth Exchange	-	Yes	Yes	No	-
	[9]	2014	-	-	-	Yes	Yes	No	-
	[10]	2013	-	HIC of Australia	-	Yes	Yes	No	-

3

Architecture

Contents

The main goal of this section is to define how the architecture is structured and organized. In order to achieve this objective, an overview of the architecture will be presented, followed by a comprehensive description of each module. In this architecture a data generator will generate a dataset, that will then be analyzed by two algorithms, present in the Data Analysis module, as to detect fraudulent behavior and finally the output of those same algorithms is shown to the user so that it can then be sent to experts for further analysis.

The architecture consists of three main modules: Data Generator, Data Analysis and Output Analysis. The Data Generator is used to generate all the necessary data to use in the Data Analysis module. The Data Analysis module is responsible for the detection of fraudulent patterns and behaviours. And finally the Output Analysis module gathers all the information regarding the fraudulent cases, groups them by type of fraud and presents the result to the user. This result would then be sent to experts for further investigation. A simple graphical representation of the architecture can be seen in Figure 3.1.

Figure 3.1: System Architecture

In the following sections an extended overview will be presented as well as a detailed explanation of each module's functionality.

3.1 General Overview

The presented architecture aims to help solve the topic discussed in this thesis: how to effectively detect fraudulent behaviour in the healthcare system. To do so it is important to define the type of fraud the system is to detect. Two algorithms were considered in the designing of the architecture. They are a medical and pharmaceutical rules based system and a geo-location based system. These algorithms were chosen since they were very well detailed in the literature and presented interesting statistical results that proved their effectiveness.

The architecture is composed by, as stated above, three main modules: Data Generator, Data Analysis and Output Analysis.

- **Data Generator**: a central module in the architecture, it generates all the necessary data to be used in the Data Analysis module. This is where all the prescriptions are generated, including each of the necessary fields. Geo-location data is also generated and all the information is stored

in the database. Both Data Analysis and Output Analysis use the database to get and process the necessary information.

- **Data Analysis**: in this module the data is analysed for possible fraudulent behaviour. As stated above, two algorithms are used, rules based and geo-location based, and here is where they process and analyse the data as to find the fraudulent cases. One analyses prescriptions, the rules based one, and the other analyses patterns in geographical distributions, the geo-location one, and together they identify different types of fraud, making it possible to detect a greater amount of fraud.
- **Output Analysis**: this module is used to select and present the result of the algorithms. This result consists of a set of prescriptions and a set of geographical distributions which are to be sent for further investigation.

Further investigation is necessary in the Ouput Analysis since there may be cases where fraudulent behaviour was detected but it turned out to be a legitimate case. For example, if a patient resides in Lisbon, Portugal and goes to a medical appointment in Porto, Portugal, that may be an indication of fraud since these locations are very far away from each other. But there can be a situation where a patient goes on vacation to Porto and decides to take an appointment there. This is a case where it seemed like the patient was committing a fraudulent behaviour but it turned out to be a legitimate situation.

Since a data generator was used, this architecture provides a wider range of possibilities than traditional detection systems, where the algorithms must rely on the data to be used within the system, i.e., the algorithms are chosen according to the dataset. This is an important aspect of this architecture that innovates in the sense that it is possible to virtually use any desired algorithm by extending the functionality of both the Data Generator and the Data Analysis modules.

To better understand the system, a module view is presented in Figure 3.2.

A more simplified view of the solution is presented in Figure 3.3 where a layered architecture shows a more structured view of the developed solution.

3.2 Data Flow

In this section the data flow of the developed application is presented. Summarily it starts by generating a dataset, which includes a set of prescriptions and a set of geographical distributions, analyses this data for fraudulent behaviour and finally it presents all the detected fraudulent cases to the user. A more detailed view of the data flow is as follows:

- The Data Generator creates a dataset which includes, as stated above, prescriptions and geographical distributions;

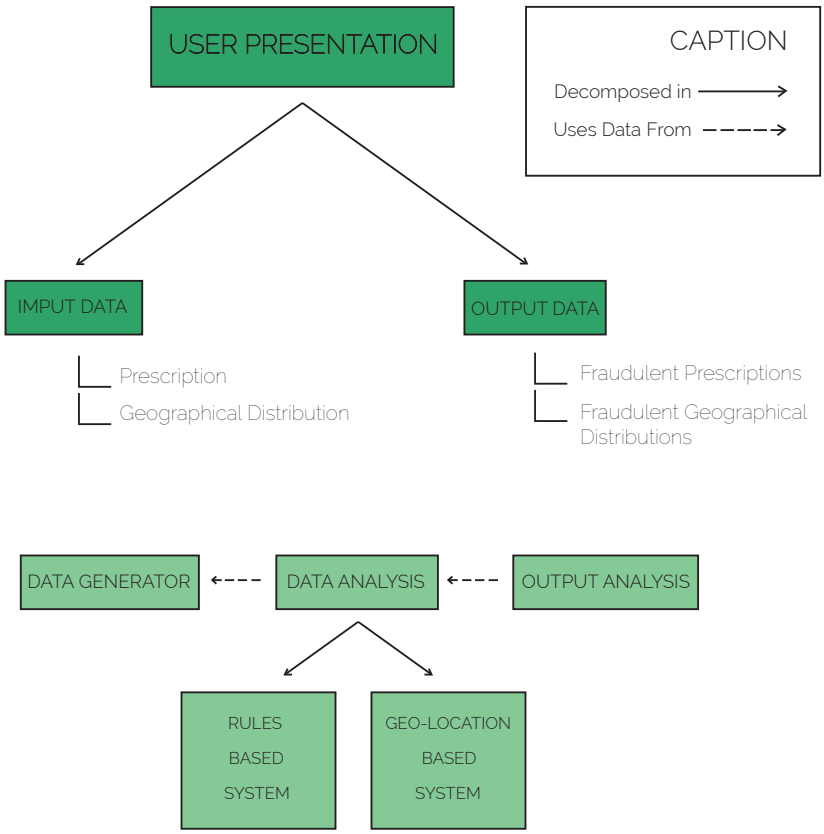

Figure 3.2: Module view of the developed solution

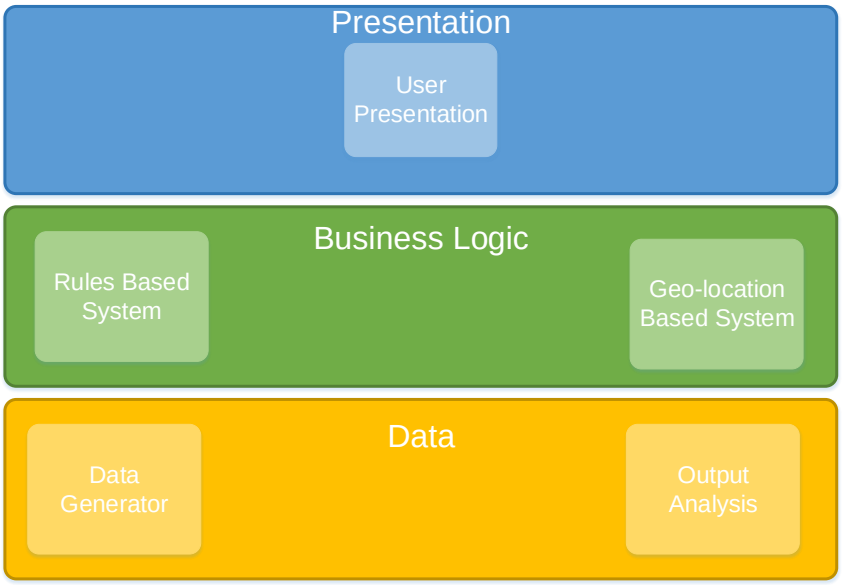

Figure 3.3: Layered architecture of the developed solution

- Next the data is analysed by two algorithms, rules based and geo-location based. The rules based uses a set of rules to check the validity of the prescriptions while the geo-location based checks every geographical distribution to see if there are suspicious cases where many patients go to the furthest pharmacy.
- In the final step the fraudulent cases are collected and presented to the user. Later the results are to be sent to experts for further investigation.

In Figure 3.4 a data flow diagram of the above described process can be seen.

In the following sections the modules from the architecture will be analysed in further detail. Section 3.3 describes the Data Generator and all its composing classes. In section 3.4 the Data Analysis module is explained and finally in section 3.5 the Output Analysis module is discussed.

3.3 Data Generator

A data generator was designed and implemented in order to generate a plethora of information. This information is then used by the detection algorithms to discover fraudulent cases. This approach is interesting, since it allows different algorithms to be tested using a single dataset that can be changed simply by executing the Data Generator program again.

The Data Generator (DG) module was designed to generate all the necessary information to be used by the Data Analysis module. This includes a database containing all the possible values for each field

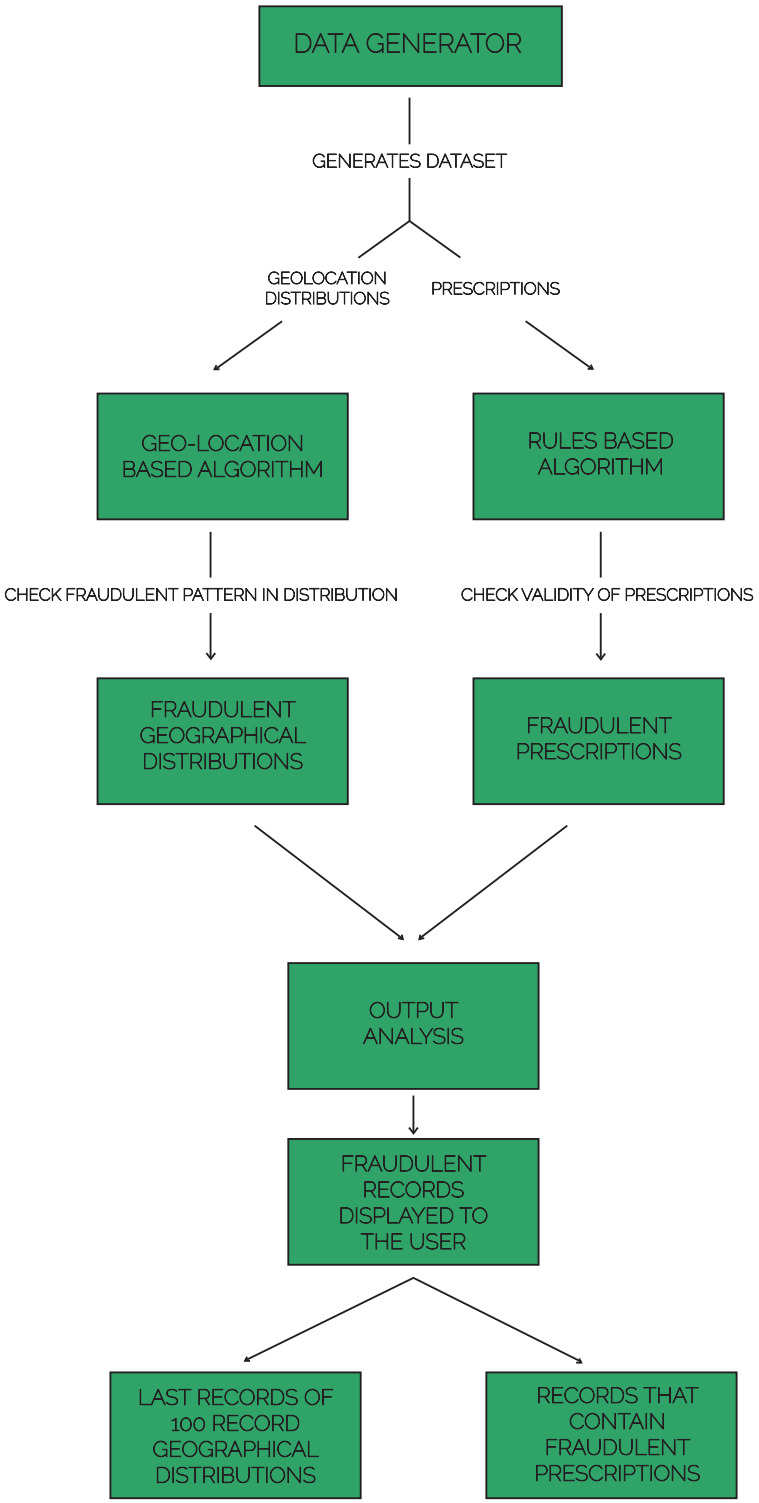

Figure 3.4: Data flow of the implemented solution

and several methods to manipulate this data in order to create a new dataset every time the algorithm is executed. It is composed by several smaller components. These components are represented in the solution as classes.

A brief description of each of the composing classes is shown below.

1. **DataGenerator** - uses all the other classes to create the dataset;
2. **CANRangeBetween** - generates a Claim Account Number (CAN), one of the two numbers that constitute the HICN (Health Insurance Claim Number);
3. **BICGenerator** - generates a Beneficiary Identification Code, used in conjunction with the CAN to create the HICN;
4. **Diagnosis** - generates a random diagnosis;
5. **Doctor** - generates a random doctor with an associated NPI (National Provider Identifier);
6. **Drug** - generates a random drug which can have a normal price or a dubious price depending on the odds;
7. **GenerateMapDrugDiagnosis** - generates a HashMap were each drug corresponds to one specific diagnosis, where there can be a valid or invalid diagnosis depending on the odds;
8. **Location** - generates a random location in the Pennsylvania state;
9. **LocationGenerator** - used to generate locations within a certain distance from an origin point;
10. **NPIGenerator** - used to generate an NPI (National Provider Identifier);
11. **OfficialDrugPriceListGenerator** - used to generate the official drug price list to which every drug price will be checked against;
12. **Patient** - generates a patient;
13. **RandomDate** - used to generate random dates for the prescription;
14. **TextFileReader** - used to read information from text files that will be used to create the final dataset;

3.3.1 CANRangeBetween

This class generates a CAN number, Claim Account Number, which is used in conjunction with the BIC number (Beneficiary Identification Code) to create the HICN (Health Insurance Claim Number). It is a complementary class that creates a random CAN to be concatenated with a BIC to generate the HICN,

which is used as a unique identifier in the Medicare federal health insurance program. It generates a number between 0 and 999999999 which is the maximum number of digits this number can have, 9 digits total.

3.3.2 BICGenerator

This class generates a BIC number, Beneficiary Identification Code, which, in conjunction with the CAN number, creates the HICN. This is also a complementary class that generates a one or two letter string that identifies the relationship a cardholder has to the qualifying wage earner. For example, if a wife of a wage earner becomes a widow, the letter code will change from “B” to “D” to reflect the change in status.

After generating both the CAN and the BIC they are concatenated together, in this order, to create the HICN, which is stored in the person HICN variable. This variable in turn is stored later in the database where it is associated to a patient in the prescription being created at the moment.

3.3.3 Diagnosis

This class is used to collect all the possible diagnoses, aggregate them in an ArrayList and send them to the GenerateMapDrugDiagnosis class, where they will be matched with their respective drug. To do this, a method is used: getIllnessList(). This method reads a text file which contains a list of the top 200 drugs in 2016 [24] and it’s corresponding purpose, which are used as a diagnostic in the solution. For example Xanax, also known as Alprazolam, has the purpose of being an anti-anxiety drug. The file is then parsed to create a list of all the diagnoses found in that file.

3.3.4 Drug

This class is, similarly to the Diagnois class, used to collect all the possible drugs in order to send them to the GenerateMapDrugDiagnosis. The method described above to collect the information is the same in this case. Using the data in [24] a list of all the drugs is collected. All the drugs are paired with the corresponding diagnosis.

3.3.5 GenerateMapDrugDiagnosis

In this class the HashMap that contains all the drugs and their correspondent diagnoses is created. There are three possibilities when using this class. One is to generate a normal case, where a random drug is selected along with it’s corresponding diagnosis. These values are used as the drug and the diagnosis in the prescription being generated at the moment. The second case scenario is when a drug is

selected randomly but, instead of selecting the corresponding diagnosis, another diagnosis is randomly chosen. Finally the last method creates the HashMap where all the drugs are correctly associated with their respective diagnostic.

This is necessary for the medical and pharmaceutical rules based algorithm since there is a rule that checks whether the prescribed drug corresponds to the prescription's diagnosis.

3.3.6 Doctor

This class represents a doctor and contains the NPI (National Provider Identifier) and data indicating whether the doctor belongs, or not, to the SSB. This data is sufficient to accurately identify a doctor, since the NPI is a unique identifier in the entire Medicare & Medicaid Services. More attributes could have been added but a decision was made to only include this data mainly for simplification reasons.

3.3.7 Location

This class is responsible for generating all the locations in the dataset. The first decision to be made was to choose from which country to collect information. The United States of America was chosen since it was simpler to get geographical information related to that country. Following the decision of the country, a state had to be chosen to collect the locations from. At first the New York state was considered, but soon one problem arose: the state has many geographical positions where there are watercourses. This is a problem because, as shown in the next class, there is a need to generate locations to a certain distance from an origin point. And it is important that they are not watercourses because they represent important geographical points, such as a residence or a pharmacy.

Therefore it was decided that the Pennsylvania state would be a better fit. Besides having a greater area it has close to no watercourses. The difference can be seen in Figures 3.5 and 3.6. It was decided that the locations would be randomly selected from the state of Pennsylvania. In order to select a random location in that state, a text file containing every location in Pennsylvania is read and parsed. Using this information, whenever a location is generated, except when using another location as a starting point, one of the coordinates from the file is randomly selected.

There are some constraints regarding the generated locations. One patient can not have the same location as a residence and as a workplace. But, as will be shown in 3.3.11, the residence and the workplace can never be the same since the workplace is dependent on the residence. A location that is a hospital however can be at the same coordinates as a patient's residence or workplace, since it is feasible that someone can live or work near a hospital.

Figure 3.5: Google Maps' representation of the New York State

Figure 3.6: Google Maps' representation of the Pennsylvania State

3.3.8 LocationGenerator

This class uses a method to generate a location to a certain distance from an origin point. In order to achieve this, it uses 5 variables: R, brng, distance, originLatitude, originLongitude. The R variable represents the radius of the earth, brng represents the wanted bearing, which in this case is 90 degrees, distance represents the distance from the origin point and finally originLatitude and originLongitude represent the origin point's coordinates. The bearing in this scenario refers to the degrees away from North of the generated location relative to the origin point. So having a 90 degrees bearing means the new location is generated to the East of the original point. Both the brng and the distance are converted to radians when calculating the new latitude and longitude along with the coordinates of the origin point.

The method receives three inputs: latitude and longitude of the origin point and the distance to which the point is to be generated. Adding to these three inputs are the radius of the earth and the bearing as mentioned above. After having all the necessary data, the formula is applied and, in the end, a new latitude and longitude are obtained, which have to be converted to degrees in the last step after applying the formula. In the final step, a location is created with the new coordinates and is then returned. The final distance, when calculated from the origin point to the new location point, is given in meters. The code is as follows:

Listing 3.1: Method that generates a location to a desired distance.

```
1
2 public Location generateLocationToXKilometers(String latitude, String longitude,
      double distance){
3
4     final double R = 6371; // Radius of the earth
5     double brng = 90; //1.57
```

```
6       double brngRad = Math.toRadians(brng); //Bearing is 90 degrees converted to
            radians.
7       this.distance = distance; //Distance in km;
8       double distanceRad = this.distance / R;
9       double lat1, lat2, lon1, lon2, lat2Degrees, lon2Degrees;
10
11      lat1 = Math.toRadians(Double.parseDouble(latitude)); //Current lat point
            converted to radians
12      lon1 = Math.toRadians(Double.parseDouble(longitude)); //Current long point
            converted to radians
13
14      lat2 = Math.asin( Math.sin(lat1) * Math.cos(distanceRad) +
15      Math.cos(lat1) * Math.sin(distanceRad)  * Math.cos(brngRad));
16
17      double aux = Math.sin(lat1) * Math.cos(distanceRad) + Math.cos(lat1) * Math.
            sin(distanceRad) * Math.cos(brngRad);
18
19      lon2 = lon1 + Math.atan2(Math.sin(brngRad) * Math.sin(distanceRad) * Math.cos
            (lat1),
20      Math.cos(distanceRad) - Math.sin(lat1) * Math.sin(lat2));
21
22      lat2Degrees = Math.toDegrees(lat2);
23      lon2Degrees = Math.toDegrees(lon2);
24
25      //generate Location object and return it here
26      Location result = new Location();
27      result.setLatitude(String.valueOf(lat2Degrees));
28      result.setLongitude(String.valueOf(lon2Degrees));
29      return result;
30
31  }
```

3.3.9 NPIGenerator

This class, as the name suggests, is the one responsible for managing the NPIs that are generated. It has two methods: generateNPI() and getNPI(). The main method is getNPI() which in turn uses generateNPI().

It starts by generating an NPI using the generateNPI() method. Then it proceeds to checking it

already exists. If not, then the NPI is added to the global list of NPIs and returned to the invoking class. If it does exist however, it will continue to generate NPIs until a new non-existent number is found.

This is a complementary class created to help manage these numbers since it makes sense to have a specific entity dedicated to this effect. This way the management is isolated from the rest of the program, making it easy to alter and maintain if necessary.

3.3.10 OfficialDrugPriceListGenerator

This class was created to simulate the existence of an official price list for the drugs used in the dataset. This is interesting because it helps solving two problems: privacy issues and flexibility. The data regarding the official prices of the drug was very difficult to find, mainly due to the sensitive nature of the information, thus resulting in no data with specific prices per drug being found. On the other hand, this approach enables not only flexibility when creating the dataset but also experimenting with several values which gives the user the ability to truly test their algorithms with as much different case scenarios as possible.

Even though it was difficult to find information regarding the price of the specific drugs used in the dataset, a document was found that gives some insight into the market, specifically the Portuguese Pharmaceutical Market in January 2016 [25]. The Portuguese Market was chosen since the only available document that provides real data regarding the prices of drugs uses this market.

Utilizing this information, this class generates the prices of drugs based on their price, i.e., the market share is organized by price. The table from where the information was taken can be seen in Figure 3.7.

PVP	January 2015	%MS	January 2016	%MS	Δ pp	MAT January 2015	%MS	MAT January 2016	%MS	Δ pp
0 € - 6,68 €	4 817	*49,4%*	4 831	*48,6%*	-0,8	6 204	*49,6%*	6 218	*50,1%*	0,5
6,69 € - 9,97 €	1 560	*16,0%*	1 558	*15,7%*	-0,3	2 007	*16,1%*	1 906	*15,4%*	-0,7
9,98 € - 14,1 €	1 090	*11,2%*	1 099	*11,1%*	-0,1	1 406	*11,2%*	1 321	*10,6%*	-0,6
14,11 € - 26,96 €	1 296	*13,3%*	1 328	*13,4%*	0,1	1 633	*13,1%*	1 632	*13,2%*	0,1
26,97 € - 64,58 €	767	*7,9%*	889	*8,9%*	1,1	984	*7,9%*	1 056	*8,5%*	0,6
> 64,58 €	220	*2,3%*	237	*2,4%*	0,1	269	*2,2%*	276	*2,2%*	0,1

Figure 3.7: Market share of drugs based on their price.

The percentages used in the solution are highlighted with a red rectangle in Figure 3.7. All the values generated in a certain interval are always randomly selected within that interval. For example, if a value is in the interval 6.69 - 9.97, one example of a price is 7.67.

This data allows for a more accurate approach of the real prices in the market and thus gives the generated datasets more authentic values.

3.3.11 Patient

This class represents the patient in the solution. It has associated with it a patientID and two locations: residence and workplace. The residence location is set in the DataGenerator class, when a patient is created. The workplace is generated using the residence location as an origin point. A decision was made to generate the workplace using the residence as a starting point, since it is the most common way of measuring the distance to work. It was also decided that the distance to the workplace would be 17Km based on two surveys, one of which was made by the author and the second one was made online with 238 participants [26]. Both of the surveys were directed at residents in Portugal since these were the only options available. The method used to generate the workplace was the same used in 3.3.8. Finally the patient has associated an attribute that stores the status of the patient in the SSB, that is, if the patient belongs or not to the SSB.

3.3.12 RandomDate

This class was used to generate the dates in the prescriptions. It receives an interval as an input in the constructor and it used that interval to create two dates separated by that value. There are two intervals used in the creation of the dates: 4 to 5 and 6 to 7.

There is a rule in the medical and pharmaceutical rules based algorithm [18], that states that there can not be a bigger interval than 5 days between the issue and the execution dates. This class generates both cases which are represented by the two intervals: the first interval refers to the normal interval since it can have 4 or 5 days in between; the second case however represents the fraudulent case where the number of days is above the limit imposed by the rule.

To generate this data, a GregorianCalendar was used. For the year, a function is used that selects a number randomly between 1992 and 2016. The day is chosen randomly between the first and last day of the year, where it is later decomposed into day of the month, month of the year and year being that the year is set to be the number previously generated.

If an interval of days happens to change the month, for example from 30/05/2015 to 03/06/2015, the appropriate change is made. This is checked by summing the current day to the defined interval and checking if the number is bigger than 32 in which case the day of the month is calculated by subtracting 32 to the initial day. For example, in the example above, since the month is May, it means it has 31 days. So we sum 30 with the interval, in this case 5, and it equals 35, which is bigger than 32. Therefore we subtract 35 to 32 and we get the correct number for the day of the month: 3. To the value of the month one unit is added. The same logic applies to the month and year.

3.3.13 TextFileReader

Finally the last class is, as the name suggests, used to read text files that contain important data to be used in the generation of the dataset. There are several classes where the TextFileReader's method, readFromFile(), is used, mainly the Drug, Location, Diagnosis and BICGenerator classes. These read text files containing the information necessary to create the dataset both for creating prescriptions (Drug, Diagnosis and BICGenerator), as explained above, and geographical distributions (Location).

One example of the use of this class is when the text file containing all the drugs and correspondent diagnoses is read and stored in the database so that the information can be used by the fraud detection algorithms to check whether the drug present in the prescription matches or not the prescribed diagnosis.

3.3.14 DataGenerator

This class is the one where all the other classes are used to create all the necessary pieces that form the dataset, i.e., where all the prescriptions and geographical distributions are generated. In order to achieve this, several methods were incorporated in this class. Also a set of attributes was created for each algorithm according to each algorithm's need. Since this is the more relevant class among this module, a longer description will be made both of the attributes and methods included in it. Following is a description of the main attributes relevant to each type of fraud detection algorithm and, subsequently, a more in depth explanation is presented of all the methods that comprise this class.

A – Medical and pharmaceutical rules based algorithm For the medical and pharmaceutical rules based algorithm the following attributes are used:

- **numberOfRecords** - the number of records, that is, prescriptions to be generated when the program is executed; for every prescription one distribution of 100 records is created where the starting point of the distribution may be a hospital, the patient's residence or it's workplace;
- **diagnosis** - the diagnosis to be attributed to a certain prescription, which changes for every prescription;
- **drugName** - the drug to be attributed to a certain prescription, which also changes for every prescription;
- **drugPrice** - the price of the attributed drug;
- **drug** - an object of class Drug that is used to collect the list of all the drugs; this list is then used to, during the execution of the program, dynamically select one drug per prescription (represented in this set of attributes by drugName, as mentioned above);

- **ssbParticipationPerc** - the monetary contribution of the SSB, whose percentage can be 70%, the normal case, or other value for a fraudulent case (the percentage is calculated later in the Data Analysis phase to check wether it is 70% or a different value);
- **issueDate** - date in which the patient receives a prescription from the doctor;
- **execDate** - date in which a patient executes the prescription;
- **personHICN** - unique identifier of a patient;
- **pharmacyNPI** - unique identifier of a pharmacy;
- **doctorNPI** - unique identifier of a doctor;
- **patient** - object of the class Patient that has information about the patient, namely, the patient's unique identifier (personHICN), the patient's residence location, workplace location and information about the patient's status in the SSB, i.e., whether he belongs or not to the SSB;
- **doctor** - object of the class Doctor, carrying information about the doctor, namely, the doctor's NPI (doctorNPI) and information relating to the doctor's status in the SSB, i.e., whether he belongs ot not to the SSB;

B – Geo-location based algorithm In the geo-location algorithm the attributes as follows:

- **startLatitude** - latitude of origin point;
- **startLongitude** - longitude of origin point;
- **homeLatitude** - patient's residence latitude;
- **homeLongitude** - patient's residence longitude;
- **pharmacyLatitude** - pharmacy's latitude;
- **pharmacyLongitude** - pharmacy's longitude;

The origin point represents the location from which the patient departs when executing a prescription. It can be a hospital, the patient's residence or the patient's workplace. The functioning of this algorithm is explained in further detail in 3.4.

After describing the attributes of this class, the methods used will now be explained in further detail.

The DataGenerator class comprises several methods. The main method is called generateData(). It is important to point out that in all cases mentioned from now on there is a 2% chance of a fraudulent

case being generated. This percentage is used based on [27] where the authors obtained values between 1% and 3% of fraud detection, therefore making 2% the average of these values and a realistic value. Being that most results are between 1% and 3%, 2% seems to be a good value for the topic of this work.

It starts by generating the HashMap with all the drugs and correspondent correct diagnoses and storing it in the database.

The next step consists in the creation of the dataset itself. A for loop is initiated, where the number of iterations is dictated by the numberOfRecords variable.

The first piece of data created are the issue and execution dates. These can be generated in two ways, or better, using two intervals: 0 to 5 or 6 to 7. The first interval will randomly generate dates which have 0 to 5 days of interval between them. Whereas if the interval is between 6 and 7, the generated dates will have those intervals, one or the other.

Next the drug and the diagnosis are created. There are two possible instances: either the drug and the diagnosis match or not. The first case is the legitimate one since the rule in the rules based algorithm states that there can not be a greater interval than 5 days in between dates. The second is the fraudulent one since it violates the rule.

The drug price follows but in this case there are three possible outcomes: normal case, fraudulent case and dubious case. The dubious cases were implemented in order to test the accuracy of the algorithms so that it is possible to have fairly realistic results when testing. The fraudulent case occurs when an incorrect price is obtained. The incorrect price is calculated by adding 5 plus a random number between 5 and 10 to the original price. For example, if the original price is 4.75 then the incorrect price could be, assuming the random number is 7, the fraudulent price will be 16.75. This method of calculation was used to try and emulate a real case scenario since in the real world the difference can not be big to avoid detection. The dubious price was a special case generated by the reasons aforementioned and a decision was made to add a value of 2 to the original price since it was lower than the minimum value created for the fraudulent price and it is was still a very low difference to the original price. Since the detection algorithm can not simply detect if the price is the same as the original price, an assumption had to be made that, in order for a price to be considered fraudulent, it had to be in between an interval which was the original price plus minAdd and the original price plus maxAdd, where minAdd and maxAdd are the inferior and superior limits that delineate the range of a fraudulent price. The following Formula 3.2 demonstrates the assumption:

$$originalPrice + minAdd <= fraudulentPrice <= originalPrice + maxAdd \tag{3.1}$$

The minAdd variable may contain values between 3 and 10 inclusive, since 2 or less will affect the dubious case detection rate. The maxAdd variable can range between 11 and 15 inclusive.

This formula allows for the dubious cases to not be detected and thus the system can be tested in a more accurate and real world like scenario, where there would be data points in the dataset that would not fit either the category of legitimate or fraudulent while also allowing the testing of the accuracy of the fraud detection algorithms. When discussing the results obtained during the evaluation process, in Chapter 4, there will be a more in depth explanation of how the dubious and fraudulent cases are managed.

Finally the normal case occurs when the drug price matches the official price, i.e., the one present in the official price list.

After the price is obtained, the patient's contribution is calculated as being 30% of the total cost of the drug, meaning the remaining 70% is contributed by the SSB.

The SSB has a normal participation percentage of 70%. However this is not always the case. There are four possible cases.

The first one is when the total cost of the drug is zero for the patient, in which case, since there is no real dataset with the actual total value of a specific drug, a value must be assumed for this type of scenarios. The assumed value is represented by the variable assumedPrice, which can take different values and is very easily changeable within the solution. The assumedPrice variable can range between 5 and 15. The second case is the fraudulent one and here the SSB's participation percentage is reduced from 70% to 50% giving the SSB 20% of the money. This is done by multiplying the total drug cost by 50%. The third case is the dubious one and here the participation percentage is reduced from 70% to 60% giving 10% to the SSB. As the previous one, this value is represented as 60% of the total drug cost. The fourth and final case is the normal case, where the percentage is 70%.

In the following lines three variables are created: personHICN, pharmacyNPI and doctorNPI. Here there are also four different outcomes. A fraudulent case is represented as one of the three variables missing in the prescription. So, for every fraudulent outcome, one of the variables is missing. If personHICN is not in the dataset, that is one case, if pharmacyNPI is 0 another case and finally if doctorNPI is 0 that is the third fraudulent case. The fourth case is when all the variables are present, therefore the legitimate one. The pharmacyNPI and doctorNPI are created using the NPIGenerator, as can be seen in 3.3.9. As for the personHICN, it used a CAN and a BIC number, which are generated, respectively, as seen in 3.3.1 and 3.3.2. By combining both CAN and BIC, in that order, the personHICN is formed.

Next the doctor is created. When creating the doctor, an NPI is associated with it and a variable with his SSB's status is set with the value true. Every doctor is created belonging to the SSB initially although there are cases, as will be seen later, where the status changes.

The next cases are related to the issue and execution date. In this case, a fraudulent set of dates is one where the execution date precedes the issue date. For example, if there is an issue date 08/08/2016 and an execution date of 03/08/2016, the execution day precedes the issue dates, meaning that when

the prescription was executed it had not been issued yet. If there is no fraudulent case, the dates remain the same. In order to create this case, the issue and execution dates are swapped, i.e., if the issue date was 03/08/2016 and the execution date was 08/08/2016, the new issue date will be 08/08/2016 and the new execution date will be 03/08/2016. The method used was chosen since it is simple and fast, thus making the creation of this data more efficient.

After the dates are created, the patient is generated. Upon the creation, one HICN is attributed to it. After this, the patient's residence is generated along with it's workplace. After both locations being generated they are attributed to the patient, i.e., they are set as attributes of the patient object being manipulated at the moment. The residence location is generated as seen in 3.3.7 and the workplace is generated using the method described in 3.3.11.

The next set of attributes are the latitude and longitude of the starting point to the pharmacy, i.e., they represent the starting point of the geographical distribution that is to be generated. As mentioned before the starting point can be one of three locations: patient's residence, patient's workplace or a hospital. These were chosen since they are believed to represent the three most common situations a patient encounters himself in. If a patient goes to a hospital and there is a pharmacy nearby, the tendency is to get the prescribed drugs right after the medical appointment. Likewise, if the patient went on an appointment and there was no nearby pharmacy, he chooses to go home and then find a pharmacy that is located closer to this place of residence. Finally if the patient is at work and needs to get a prescribed drug every month or so, he may decide to go to a pharmacy nearby in his lunch break or right after he leaves work. Having three distinct places to choose from, it is necessary to decide which places will be more likely to be chosen, since it does not make sense that the workplace has the same probability as the hospital to be selected. Therefore it was decided that the workplace had a very low chance of getting selected, represented by the chanceWorkplaceStartPoint variable, the residence had a bigger chance but still low of being selected, represented by the chanceResidenceStartPoint variabe and finally the hospital had the remaining chance, represented by the chanceHospitalStartPoint. The chanceWorkplaceStartPoint can range between 1% and 19%, the chanceResidenceStartPoint can range between 20% and 69% and finally the chanceHospitalStartPoint can range between 12% and 79%.

If desired it is easy to set the presented variables to different values within the solution. The locations are generated as seen in 3.3.7.

Following the selection of the starting point come the cases where the type of geographical distribution is decided. There are three outcomes: fraudulent, dubious and normal case. The normal distribution occurs when most patients go to the pharmacy which is closer to the starting point, some go the second closer and very few go to the furthest. The fraudulent case however inverts this tendency, by having a greater number of patients going to the furthest pharmacy. The third scenario is a dubious one where

25% go to the second closer and furthest pharmacies and the remaining 50% go to the closest one. This case is dubious since there is an even amount of patients going to the more far away pharmacies and a majority of them goes to the, expected, closer pharmacy. Once again this case was created to test the accuracy of the algorithms.

The distributions are generated by using the starting point and creating 100 different locations to a certain distance from it. The distance can be of 3, 5 or 10 Kilometres. Depending on the type of distribution, as mentioned, there will be different scenarios. In the legitimate case, the following distribution is generated:

- 3 Km normal3KmPerc
- 5 Km normal5KmPerc
- 10 Km normal10KmPerc

In this distribution the normal3KmPerc variable can range between 60% and 70%, normal5KmPerc can range from 20% to 30% and finally normal10KmPerc can range from 0% and 10%. The normal10KmPerc can not take a 20% value since that case is considered to be a dubious case, like will be seen in the following lines.

This distribution was chosen since it is an approximation to the normal case scenario where most of the patients would choose the closest pharmacy, whereas only a small percentage would go to a far away pharmacy.

The fraudulent distribution, however, generates a different pattern:

- 3 Km fraudulent3KmPerc
- 5 Km fraudulent5KmPerc
- 10 Km fraudulent10KmPerc

In this distribution the fraudulent3KmPerc variable can range between 40% and 50%, fraudulent5KmPerc can range from 20% to 30% and finally fraudulent10KmPerc can range from 30% to 40%.

In this case the distribution is clearly unbalanced as there is a greater percentage of patients going to the furthest pharmacy, clearly indicating a suspicious behaviour that could involve fraudulent activity.

A visual representation of both the normal and fraudulent distributions can be seen in Figure 3.8 and Figure 3.9 respectively.

At last there is the dubious distribution. The distribution is as follows:

- 3 Km dubious3KmPerc
- 5 Km dubious5KmPerc

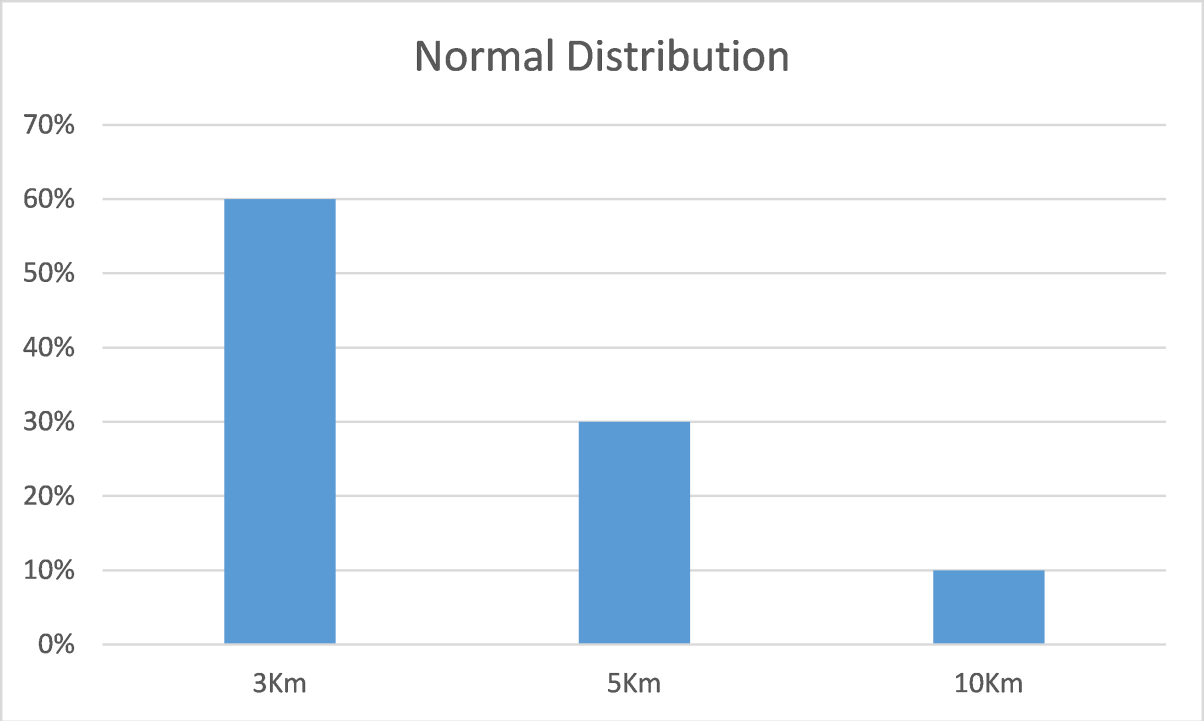

Figure 3.8: Normal Distribution

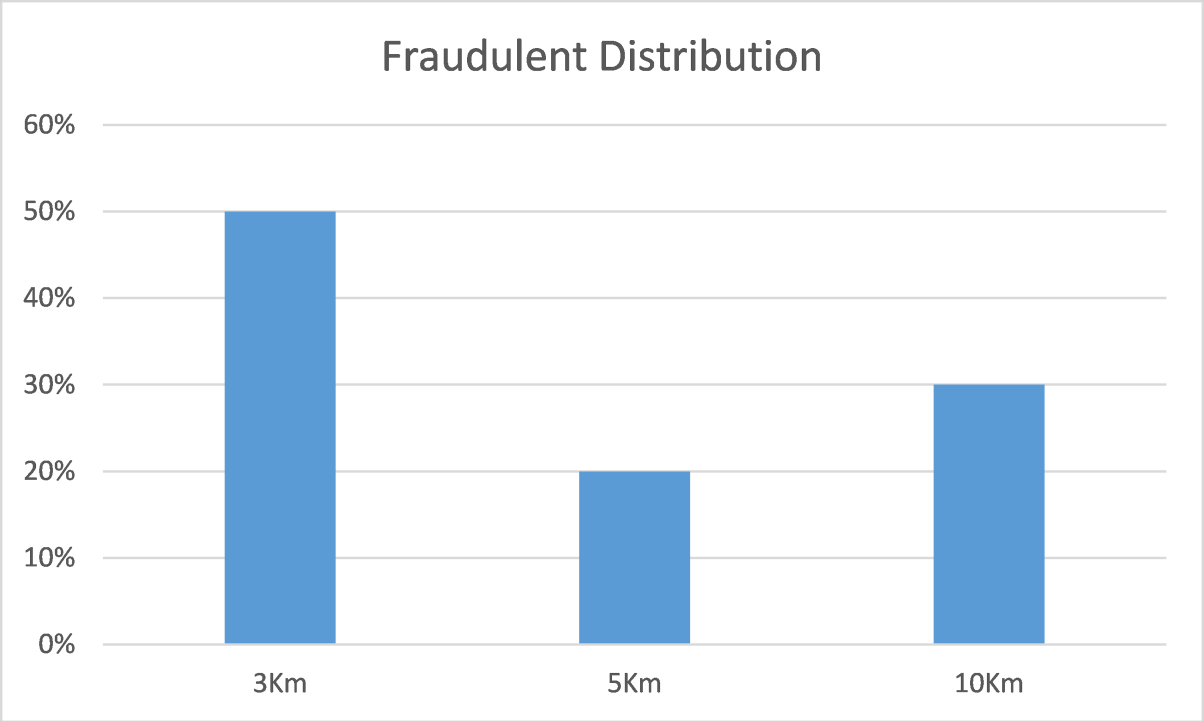

Figure 3.9: Fraudulent Distribution

- 10 Km dubious10KmPerc

Finally in this distribution the dubious3KmPerc variable can range between 50% and 70% and both dubious5KmPerc and dubious10KmPerc can range from 25% to 15%.

As mentioned above, this case is difficult to classify into either legitimate or fraudulent since there is an even distribution of the patients for the two more far away pharmacies, making it harder to assess whether there exists fraudulent activity. In chapter 4 the effectiveness of the fraud detection algorithms will be tested using this and the two dubious cases mentioned above.

The final data to be generated is the SSB's status of the entities represented in the dataset, which are the patient, the doctor and the pharmacy. As in previous data generation, there is a legitimate and a fraudulent case. The fraudulent case occurs when one of the three mentioned entities does not belong to the SSB. In the solution the doctor, the patient and the pharmacy have a boolean attribute that represents this status.

In the end all the data is stored in the database. This process is explained in greater detail in the following lines.

C – Database storage The information is stored in the database in three main phases: data collection, temporary data storage and data writing. In the first phase all the queries to the database are collected; the second phase stores these queries in text files in order to optimize the performance of the system at runtime and finally the data is read from the text files and executed in the end using only one connection to the database, which saves immense time due to the delays present whenever a connection is opened and closed.

In the data collection phase, every query to be executed is stored in a string. This happens every time the main method, generateData(), executes a cycle of the for loop mentioned above. So every time the loop runs, there are several strings that store the queries necessary to write the generated data to the database. For each of the mentioned classes there is one specific string that stores the information described below:

- **Diagnosis** - HashMap information containing all the drugs and respective diagnosis in the correct order;
- **Doctor** - doctor's NPI and SSB status;
- **Hospital** - hospital's NPI and respective coordinates, i.e., latitude and longitude;
- **LocationDistribution** - patient's HICN, origin point coordinates and pharmacy location, stores all the generated geographical distributions;

- **Prescription** - patient's HICN, pharmacy's NPI, doctor's NPI, issue date, execution date, drug's name, drug's price, diagnosis, starting latitude, starting longitude, patientId, pharmacy's latitude, pharmacy's longitude, SSB's drug participation percentage, patient's drug contribution;
- **Patient** - patient HICN, residence and workplace coordinates and SSB status;
- **Pharmacy** - SSB status and pharmacy coordinates;

In order to make the writing to the database efficient, the queries were joined together in one big query. For example, if we have a query that inserts two integers x and y in table Table1 and want to insert 2 rows using only one query, the query would be

```
INSERT INTO TABLE1 VALUES (1,2),(3,4);
```

instead of

```
INSERT INTO TABLE1 VALUES (1,2);
```

and

```
INSERT INTO TABLE1 VALUES (3,4);
```

. This method allows for a significant improvement in performance since it is much faster to execute the first insert than several separate ones. So using this method, the strings are initialized with the value

```
INSERT INTO TABLE_NAME VALUES
```

and then, with each new iteration, a new set of values is concatenated to the string. For example, using the last insert, after the first iteration of the for loop, a possible value of the string could be

```
INSERT INTO TABLE_NAME VALUES (1,2),
```

.

Based on this approach, during the collection phase, every 100 cycles performed by the for loop, the strings are copied to a text file, where each string has its own text file associated. This measure was implemented to improve performance since the program, during execution, was using a great amount of RAM memory and so, to reduce that usage, the strings are written into a text file and set to a null string so that the program does not have to concatenate bigger and bigger strings as it runs. Concatenating big strings consumes more and more RAM because in order to concatenate a string, Java creates a new string with the new substring at the end of the old string. This information is stored in RAM when it is being created, making it more and more resource hungry as time goes on.

When the loop ends and all the information is generated the data is written to the database. This is achieved by reading all the text files into their respective strings and afterwards opening a connection to

the database and executing all the queries sequentially. This method increases the writing's speed in about 97.5% versus opening and closing a connection whenever an insert or update is executed in the database and concatenating bigger and bigger strings.

The function responsible for writing to the database is called writeDataToDB(). It works as described above with an extra step before executing the queries, which is replacing the last comma with a semi-colon. This step is necessary since with every concatenation of a new substring, it always ends with a comma. As such, to be able to execute the query correctly a replacement is necessary. This replacement is executed by the fixComma() method.

The remaining methods are responsible for creating the strings that contain all the information to be written in the database in the last step.

In the following section the fraud detection algorithms will be explained in detail.

3.4 Data Analysis

Analysing data for fraudulent behaviour is not a simple process since there may be cases where the data indicates a suspicious pattern but in reality it was just an abnormal case where one patient used a far away pharmacy simply because he was on vacations and bought his prescribed drugs at a pharmacy near his holiday location. In other words, there may be a potential scenario where a patient goes on holidays in a place that is faraway from his residence or local area and needs to buy a drug. In this scenario he will go to the nearest pharmacy in his holiday area and make the purchase. According to a geo-location algorithm that detects distances from a residence, or other relevant starting point in the local area of residence, to a pharmacy this would be considered a fraudulent behaviour since the measured distance would be considerably larger than the normal case scenario where a patient goes to the nearest pharmacy available. This case would then be reviewed by an expert and the conclusion would be that, in fact, it was an anomalous case however a legitimate one. An example of this behaviour can be seen in Figure 3.10.

In order to solve this problem, algorithms were developed to more accurately detect such patterns and behaviours. Of the many existing algorithms in the literature two were chosen: one based on medical and pharmaceutical rules and another based on geo-location. The first one analyses prescriptions for missing or incorrect data. The second one was adapted from the original algorithm present on the paper and instead of checking only the distance, it checks for patients' distributions along the various nearby pharmacies.

In the following subsections both algorithms will be analysed and explained in greater detail.

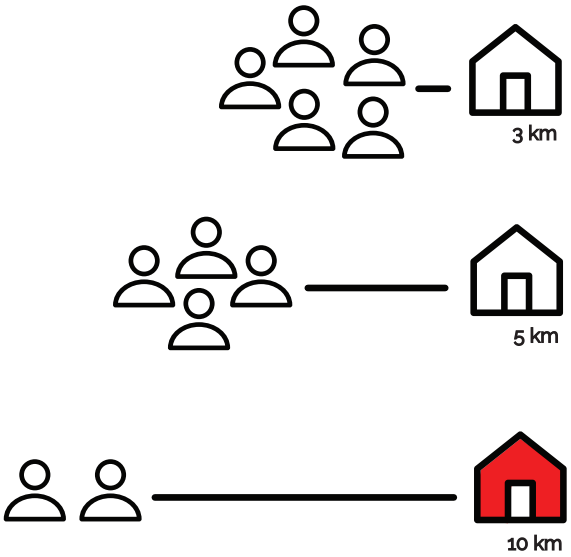

Figure 3.10: Example of a possible fraudulent behaviour where a few patients go to a faraway pharmacy.

3.4.1 Medical and Pharmaceutical Rules Based Algorithm

The algorithm starts by retrieving the official drug price list and the list of drugs and corresponding diagnosis and stores them in two ArrayLists to facilitate future access.

The first method, checkExecDatePrecedesIssueDate(), corresponds to the rule that states the execution date may not precede the issue date present in the prescription. There are three case scenarios: the days are on the same month and year, the days are on different months and same year and the dates are in different months and years. In the first case the only necessary step is to subtract the execution date day from the issue date day and check the result. If the result is negative then it means there is a fraudulent case. For example 01/08/2016 is the execution date and 04/08/2016 is the issue date. If we subtract 1 to 4 we get negative 3 which is below 0 and thus a fraudulent case. In the second case the same principle applies except this time to the month. For example 31/03/2016 being the execution date and 03/04/2016 being the issue date, if we subtract the months, i.e., 3 minus 4, we get negative 1, which is lower than 0 and thus a fraudulent case. Lastly there is the scenario where the month and the year is different. The same logic can be applied to this case. For example 31/12/2015 as the execution date and 03/01/2016 as the issue date, if we subtract the years we get a negative 1 which is lower than 0 thus indicating that this is, in fact, a case of fraudulent behaviour.

The next method, checkFiveDayExecutionPeriod(), as the name suggests, coincides with the rule that states there can not be a bigger interval than 5 days in between the issue date and the execution date of the prescription. In order to validate this rule, a similar approach to the first method was taken with the difference being that here it is checked if the difference between the days is greater than 5, that is, 6 or more. Like the previous method there are also several cases. The first one checks whether the dates are from the same month and year, in which case two additional verifications are made to assess whether there is a fraudulent case: if the difference between the execution date day and the issue date day is bigger than 0 and if the difference between execution date day and issue date day is greater then 5. The first verification is used to guarantee that the cases being tested do not have the execution date preceding the issue date. The second case is one where the month is different but the year is the same. Here there are two options depending on whether the month of the execution date has 31 days or not. If there are 31 days in the month, then 31 is subtracted to the issue date day, the execution date day remains the same and then both these numbers are added and a check is made to see whether the sum is greater than 5. An example to clarify: having an issue date of 29/08/2016 and an execution of 04/08/2016, the first value would be 31 minus 29 which is 2; so the sum of both values would be, since the execution date remains the same, 2 plus 4 which is 6. The number 6 in then compared to see if it is greater than 5, which in this case, it is and as such it is considered to be a fraudulent case. The same logic applies to the 30 days months and to February (here it was assumed the year was never a leap year, having only 28 days). When the years are different, the case is similar to when the months

are different and same year, with 31 days since this case only happens in December which is a 31 day month.

The third method is checkMissingCode(). It corresponds to the rule that states a prescription must include the insured person's, pharmacy's and doctor's codes. In this work the person's code is represented by the Health Insurance Claim Number (HICN) while the pharmacy's and doctor's codes are represented by an National Provider Identifier (NPI). As was seen above, there are cases where one of these will not be present. What this method does is check whether a prescription has one of them missing. The way this is implemented is all the three field are checked: if the personHICN is null or the pharmacyNPI is 0 or the doctorNPI is 0 there is a fraudulent case. Otherwise it is a legitimate one.

The checkSSN() method corresponds to the rule that states the doctors, patients and pharmacies must be members of the SSB. As such it checks each of these elements' SSB status. If they do not belong they are suspicious of fraud, otherwise they are not.

The checkFalseParticipationPercentage() method matches the rule where each diagnosis has a predefined participation percentage for the SSB in the price of drugs in which the normal case is 70%, the fraudulent is 50% and the dubious one is 60%. In order to verify the percentage, in the implementation, the SSB´s participation value is obtained and then this value is divided by the total price of the drug and if the value is 0.5 then a suspicious fraudulent case exists.

The checkDrugsOverPrice() method verifies that a drug's price is the official one. To do so, a rule is verified:

$$originalPrice + minAdd <= fraudulentPrice <= originalPrice + maxAdd \tag{3.2}$$

This rule was created, as is mentioned above, to make possible the creation of the dubious case which consists in adding 2 to the original price. As such, if the price is in between this interval of values (the original value plus a number between 5 and 10) then it is considered a fraudulent case.

The checkCorrespondenceBetweenDrugDiagnosis() method, as the name implies, corresponds to the rule stating that the correspondence of each prescribed drug must be checked against the prescription's diagnosis. To do this, the Hash Map generated in the Data Generator is used to check every drug and their respective diagnosis. In this method the only verification made is whether the diagnosis in the prescription is identical to the one in the Hash Map. If not, there is a fraudulent case.

These methods comprise the medical and pharmaceutical rules based algorithm. In the next subsection the geo-location based algorithm will be described.

3.4.2 Geo-location Based Algorithm

This algorithm is composed of one method: checkDistribution(). The objective is to calculate all the distances and for every 100 records check the distribution. As mentioned above there are three

distributions: normal, dubious and fraudulent. This method checks whether the number of patients that go to a pharmacy located 5 Kilometres from the origin point, is lower than the number of patients that go to a pharmacy located 10 Kilometres away. A visual example of this distribution can be seen in 3.9. This way the fraudulent case is always detected whereas the dubious case is never identified.

3.5 Output Analysis

This module is used to collect and present to the user all the detected fraudulent cases. After the Data Analysis phase is completed and all the cases are detected they are presented to the user so that they can be analysed by experts and a more extensive verification can be made.

In this module the output of all the algorithms is collected and presented to the user. There are two sets of outputs and those are the ones from the medical rules based algorithm and the geo-location based algorithm. The medical rules based algorithm are categorized by type of rule, i.e., each medical or pharmaceutical rule has a subset of fraudulent cases. The geo-location algorithm is different since there is only one case scenario which consists in having a fraudulent distribution and as such there is only one subset for the geo-location fraudulent cases.

For each subset of the medical rules based algorithm two pieces of information are presented: the table from which the fraudulent case was obtained and the fraudulent record containing the information of the suspicious fraudulent case. Knowing the table is important since several tables are used when storing the data and so, by identifying the correct table, the experts can more easily access the original record. Besides giving this information, the total number of records in every subset is also displayed in the first line before presenting all the records.

In the geo-location case, as mentioned above, there is only one set of fraudulent cases. As in the previous case the total number of fraudulent cases is presented in the beginning. However the records are displayed in a slightly different way: even though the table is also presented, the record presents the final record of the 100 record distribution. It was decided that the final record would be used, in order to make the consulting process of the records easier.

In the end all the detected records are presented to the user. The final purpose of this solution is to reduce the amount of records to analyse from several millions to a few hundreds. This process allows the experts to concentrate on the cases with a high probability of being fraudulent and thus significantly reduces the amount of time and energy necessary to complete this task.

In the following section the process of creating a server in Amazon Web Services (AWS) and running the data generator is discussed.

3.6 Amazon Web Services

In order to adequately test the solution it is necessary to create a big volume of data. To do so requires both great processing power and memory. Unfortunately acquiring such a system involves a high price which is not within reach of all the population. However there is an alternative solution that is both cheap and scalable. That solution is found in the Amazon platform, more specifically, in Amazon Web Services or, as it is more commonly known,AWS.

Amazon Web Services is a platform that offers services related to cloud computing on demand. The price of usage is very cheap varying from $0.006 an hour for the simplest machine available, the t2.nano, to $13.338 an hour for the most powerful machine available with 4 Central Processing Unit (CPU)s and 2 Terabytes of Random Access Memory (RAM).

In this section the creation and usage of AWS servers will be discussed as well as the cost per server and hour, specifically the c3 family since this was the chosen one due to it being compute optimized.

The first goal is to discover what the requirements are to be able to generate the desired volume of data. To accomplish this, the data generator was executed and the usage of the machine was analysed. Initially it was ran on an Asus K55VM-SX082V which has an Intel Core i5 3210M CPU and 4 Gigabyte (GB) of RAM. From the analysis it was possible to conclude that the resource the solution was using the most of was the CPU. As such, it was necessary to use machines that were optimized for heavy computation. Amazon offers such instances and they belong to the c3 group which means every c3 instance is optimized for computationally demanding, i.e., CPU intensive applications. The differences between them are in the number of CPUs and amount of RAM.

There are five different machines in the c3 group. All use the same CPU in variable amounts, and Intel Xeon E5-2680 v2 (Ivy Bridge) Processor and use Solid State Drive (SSD) storage. The CPU number varies from 2 up to 32, the memory (RAM) starts at 3.75 Gibibyte (GiB) and ends in 60, finally the storage starts with a two 16 GB SSDs, adding to 32 GB and finishes with two 320 GB SSDs which sum a total of 640 GB. In Figure 3.11 a table is presented where all the information related to the c3 instance types is summarized.

Model	vCPU	Mem (GiB)	SSD Storage (GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Figure 3.11: Available c3 instances in Amazon Web Services.

The first server, c3.large, could not be used since there was not enough RAM for the solution to run. However the second one, c3.xlarge, had enough memory and as such was a good candidate. Both the c3.xlarge and c3.2xlarge were tested being that the two took about 1 hour and 30 minutes to generate 100 thousand prescriptions plus 10 million records of geographical distributions. Since both presented the same performance, the c3.xlarge was chosen considering it's price corresponded to half the one from c3.2xlarge without noticeable improvements.

In Table 3.1 the prices for the different instances aforementioned are presented.

Table 3.1: Price of each instance in the c3 group.

		Price per Hour
Instance Type	c3.large	$0.105
	c3.xlarge	$0.210
	c3.2xlarge	$0.420
	c3.4xlarge	$0.840
	c3.8xlarge	$1.680

3.6.1 Server Deployment

In order to use a server it is necessary to deploy it first. As mentioned above the chosen instance type was the c3.xlarge, therefore the steps to create a server will be the ones used for c3.xlarge but they are easily transferable to any type of instance as will be seen in the following lines.

The first step is, after logging in to AWS's website [28], going to the S̈ervicesẗab and select C̈omputeẗfollowed by ËC2.̈ This will lead the user to the Ämazon Elastic Compute Cloudservice which provides scalable compute capacity. In this page it is possible to see a variety of information, for example the number of running instances a user has at a given moment, how many dedicated hosts or elastic Internet Protocol (IP)s are currently associated to the user. Next the R̈unning Instancesöption is selected, which leads to a page where all active instances are listed. Since there are no running instances the first time the user enters the page, it will be empty, presenting a message with guidance and a blue button with the

words L̈aunch Instance.̈ By clicking this button a new page appears where a list of all possible Amazon Machine Image (AMI)s is presented. The c3 group only supports HVM type AMIs which leaves only one option: the Amazon Linux AMI 2016.09.0 (HVM). An example of the AMI selection process can be seen in Figure 3.12.

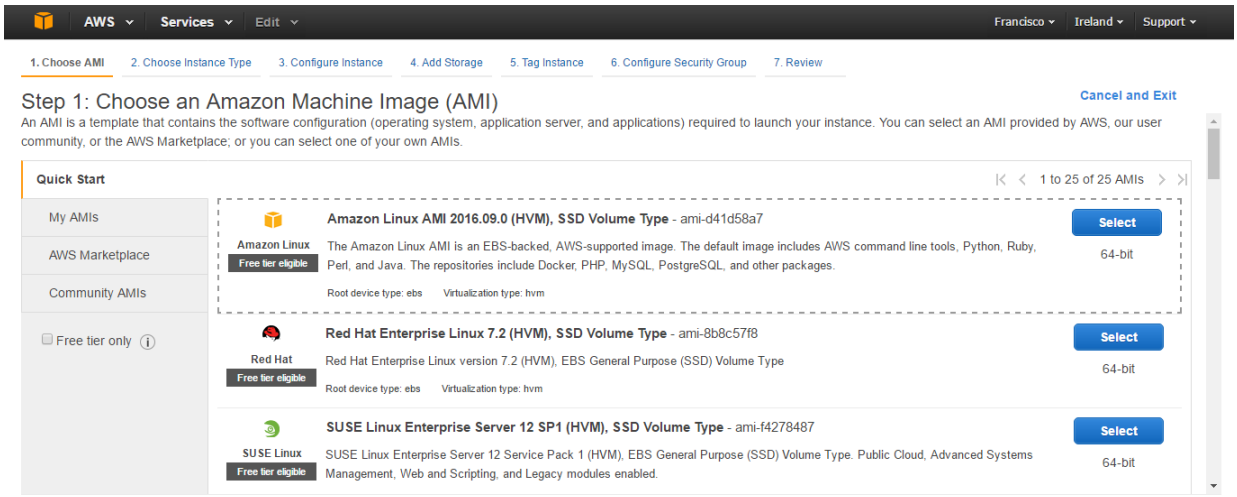

Figure 3.12: Menu for selecting the desired AMI.

Selecting this option, leads to a new menu where the user must choose the instance type. This is where the process differs according to the desired type. In this particular case, the c3.xlarge is selected by clicking the button R̈eview and Launch,̈ as can be seen in Figure 3.13.

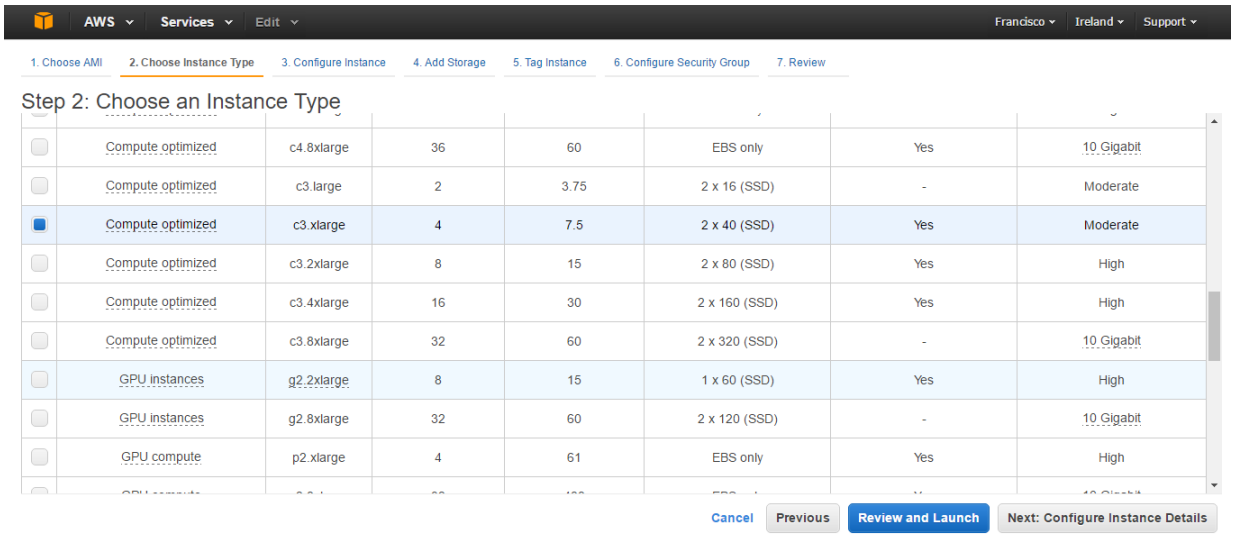

Figure 3.13: Menu for selecting the instance type.

A new page appears with a summary of the selected instance and configurations associated with it, as seen in Figure 3.14.

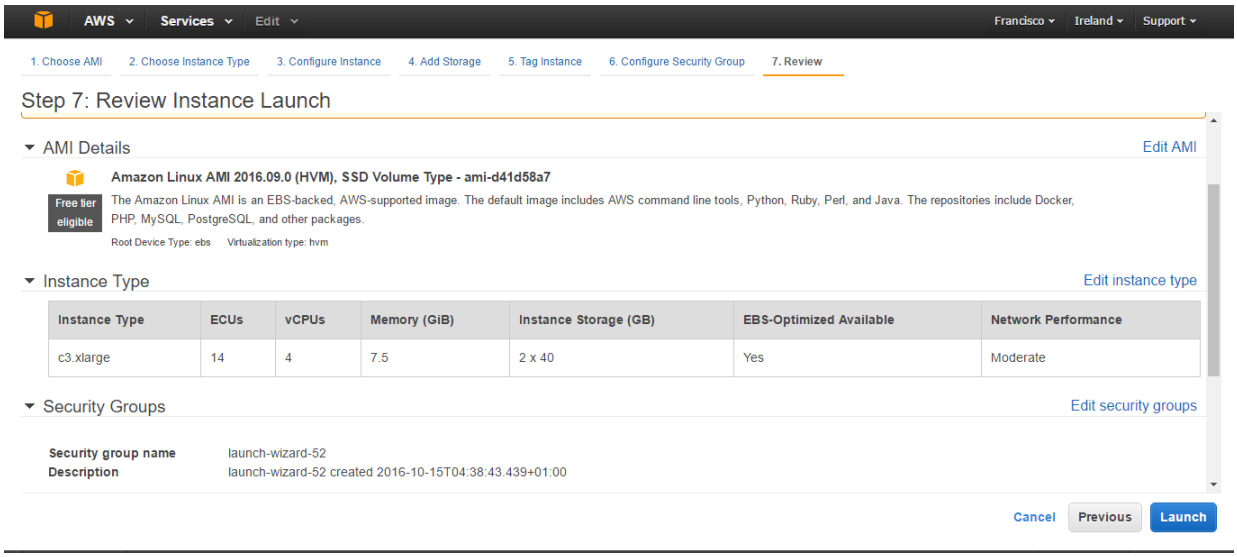

Figure 3.14: Review the selected instance type before launching.

Nothing was changed from the default settings. When clicking the L̈aunchb̈utton a pop up window appears so that the user can select the private key pair to use in this instance, as shows Figure 3.15.

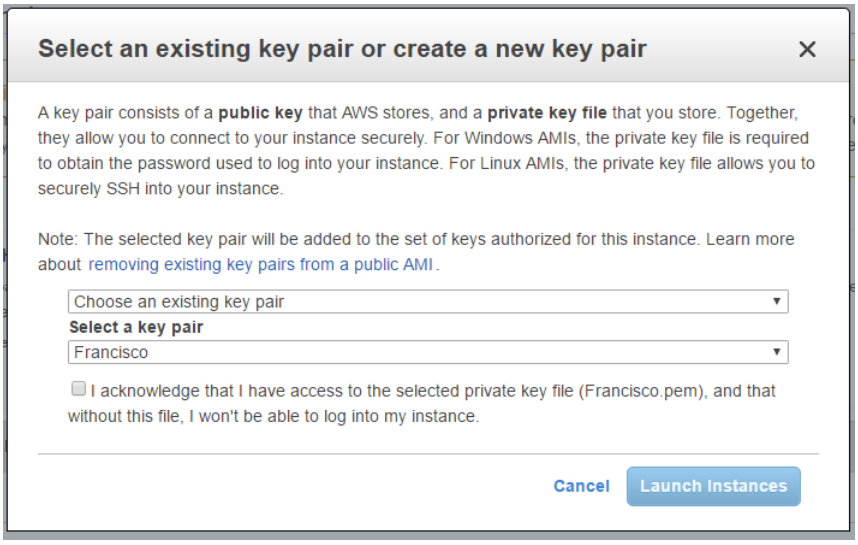

Figure 3.15: Selection of the key pair to be used to secure the instance when accessing it remotely, for example with SSH.

As soon as the key is chosen and the user clicks the L̈aunch Instancesb̈utton the instance begins the preparation process and the clock starts ticking. It takes a few minutes, approximately 5, to set up. As soon as it is ready it can start to be used. From then on, for every hour it costs $0.21 per hour.

3.6.2 Code Execution

After setting up the server, it is necessary to install all the needed software and upload the Java executable to the server. In order to achieve this, a bash script was created. This script allowed not only to automate the entire process but also to do it in a much faster way than installing and running everything manually, which is very important since, when a server is shut down everything is wiped from the hard drive meaning every time one creates a new instance everything has to be newly installed. The code for the bash script it as follows:

Listing 3.2: Bash script that automates the process of installing all the software and running the data generator.

```
1
2  read -p "Insert ip address of remote server: " ipaddress
3
4  constring="ssh -i /home/francisco/.ssh/TeseHadoop.pem ec2-user@${ipaddress} /bin/
       bash"
5
6  sudo scp -i /home/francisco/.ssh/TeseHadoop.pem -r /home/francisco/Desktop/
       DataGenerator  ec2-user@${ipaddress}:/home/ec2-user/
7
8  # ` is used to evaluate the statement and store the result in the variable
9  mysqlexists=`echo "which mysql" | ${constring}`
10 javaexists=`echo "which java" | ${constring}`
11
12 #-z if it is empty
13 if ! [[ $mysqlexists =~ */usr/bin/mysql* ]]
14 then
15 echo "sudo yum -y install mysql mysql-server" | ${constring}
16 echo "cd /etc && sudo sed -i '/pid-file=\/var\/run\/mysqld\/mysqld.pid/a bind-
       address=127.0.0.1' /etc/my.cnf" | ${constring}
17 echo "cd /etc && sudo sed -i '/socket=\/var\/lib\/mysql\/mysql.sock/a
       max_allowed_packet=500M' /etc/my.cnf" | ${constring}
18 echo "sudo service mysqld restart" | ${constring}
19 echo "mysql --protocol=TCP -u root < /home/ec2-user/DataGenerator/changeRootPass.
       sql" | ${constring}
20
21 else
22 echo "mysql --protocol=TCP -u root -ptoor < /home/ec2-user/DataGenerator/
       truncateTables.sql" | ${constring}
```

```
23 echo "mysql --protocol=TCP -u root -ptoor < /home/ec2-user/DataGenerator/
       resetAutoIncrement.sql" | ${constring}
24 echo "mysql --protocol=TCP -u root -ptoor < /home/ec2-user/DataGenerator/
       resetDatabase.sql" | ${constring}
25 fi
26
27 echo "sudo yum -y remove java-1.7.0-openjdk.x86_64" | ${constring}
28
29 if ! [[ $javaexists =~ */usr/bin/java* ]]
30 then
31 echo 'sudo wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=
       http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "
       http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-x64.rpm"'
        | ${constring}
32 echo "sudo rpm -Uvh jdk-8u45-linux-x64.rpm" | ${constring}
33 fi
34
35 echo "mysql --protocol=TCP -u root -ptoor < /home/ec2-user/DataGenerator/
       createHealthcareDatabase.sql" | ${constring}
36
37 echo "mysql --protocol=TCP -u root -ptoor healthcare < /home/ec2-user/
       DataGenerator/healthcareDump.sql" | ${constring}
38
39 echo "RESETING AUTO INCREMENT"
40 echo "mysql --protocol=TCP -u root -ptoor < /home/ec2-user/DataGenerator/
       resetAutoIncrement.sql" | ${constring}
41
42 echo "Starting DataGenerator..."
43 echo "cd /home/ec2-user/DataGenerator && java -Xmx5000m -cp GeradorDeDados.jar pt
       .ist.utl.OfficialDrugPriceListGenerator" | ${constring}
44
45 echo "cd /home/ec2-user/DataGenerator && java -Xmx5000m -jar GeradorDeDados.jar"
       | ${constring}
46
47 echo "mysqldump --protocol=TCP -u root -ptoor healthcare > /home/ec2-user/
       DataGenerator/healthcareResult.sql" | ${constring}
48
49 sudo scp -i /home/francisco/.ssh/TeseHadoop.pem -r ec2-user@${ipaddress}:/home/
```

```
ec2-user/DataGenerator/healthcareResult.sql /home/francisco/Desktop/
DataGenerator/
```

Using this script the only input necessary is the public IP address of the server. Using that input the script first checks if the necessary software is already installed in the server, which for this application is the MySQL Server and the Java JDK. If not, which is usually the case when creating a new Amazon Linux AMI 2016.09.0 (HVM) instance, the script proceeds to installing both MySQL Server and the Java SDK. After the installation is complete, the database schema is inserted into MySQL as to create all the necessary tables. Next the data generator is executed and all the generated data is stored in the database during the final step of the program. When the program finished its execution, all the data in the database is exported to a file, healthcareResult.sql in this case, which is then copied back from the Amazon server to the local machine where the file is then imported to the local MySQL running server. The detection algorithms are executed in the local machine since they do not require heavy computation in order to run correctly.

This script was created and executed using a virtual machine running the Red Hat operating system since it is the closest one to the distribution used in the Amazon Linux AMI 2016.09.0 (HVM). It was important to select the closest possible Operating System (OS) in order to test the script commands before using the Amazon server since not all distributions use the same commands to execute the same tasks.

3.7 Chapter Summary

In this chapter the architecture was presented and all its core components were analyzed and discussed. The Data Generator module was presented as being the one responsible for creating all the data necessary to test the detection algorithms. After generating all the necessary information, the Data Analysis module analyses this data and finds suspicious fraudulent cases based on medical rules and geographical distributions. In the end the Output Analysis module collects and presents to the user all the detected cases, organizing them in different subsets based on the type of fraud.

4

Evaluation Results

Contents

In this chapter the main goal is to test the developed algorithms and check their effectiveness. In order to accomplish this objective, performance measures will be used. They are confusion matrix, accuracy, precision-recall curve, and K-fold cross-validation. By using all of these metrics it is possible to have a good general idea of how well the system performs since five different measures are applied thus making the evaluation method fairly comprehensive.

Before describing the metrics in greater detail, a table is presented with all the variables used in Chapter 3, Table 4.1, where all of them are given values which are discussed below.

Table 4.1: Variables presented in the architecture with the respective values.

		Selected Values
Variables Presented in the Architecture	minAdd	5
	maxAdd	10
	assumedPrice	10
	chanceWorkplaceStartPoint	10%
	chanceResidenceStartPoint	20%
	chanceHospitalStartPoint	70%
	normal3KmPerc	60%
	normal5KmPerc	30%
	normal10KmPerc	10%
	fraudulent3KmPerc	50%
	fraudulent5KmPerc	20%
	fraudulent10KmPerc	30%
	dubious3KmPerc	50%
	dubious5KmPerc	25%
	dubious10KmPerc	25%

These values were assumed since no real data was found to support them. In the following paragraphs all the values are discussed and explained.

The minAdd and maxAdd variables are used to represent the range in which the algorithm detects overpriced drugs. These values were chosen since they represent a slight increase but nonetheless they produce values with a reasonable difference. It was important to use values that would be both plausible and different enough to be considered a fraudulent case in the real world.

The assumedPrice variable is used when a drug has a cost of 0 to the patient, in which case the SSB's contribution is total, i.e., the SSB pays for the total price of the drug. A value of 10 was assumed for this variable since it is close to the centre of the price table in which the values were based on, thus making it a good average value.

The chanceWorkplaceStartPoint, chanceResidenceStartPoint and chanceHospitalStartPoint have the values 10%, 20% and 70% respectively. These values represent the probability of a patient going to

a pharmacy from his residence, workplace or a hospital. Since the most common case for patients is to go to the pharmacy right after getting a prescription from a medic, the hospital has a larger percentage than the other locations. The next location with a higher percentage is the residence since, after the hospital, it is the more likely place from which the patients would visit a pharmacy. Lastly the workplace with the lower percentage since it is the less likely place to go directly from to a pharmacy.

The next three variables, normal3KmPerc, normal5KmPerc and normal10KmPerc represent the normal distribution of patients throughout pharmacies located at three different distances. The values are 60%, 30% and 10% respectively. These were chosen since the majority of patients chooses to go to the closest pharmacy available, and so the percentage is higher in the closest pharmacy and lower in the furthest one.

Following the normal distribution comes the fraudulent one with the variables fraudulent3KmPerc, fraudulent5KmPerc and fraudulent10KmPerc. The chosen values were 50%, 20% and 30%. In this distribution the objective was to create a scenario were more patients go to the furthest pharmacy since this is an indication of a fraudulent behaviour. So, the percentage of patients going to the second closest pharmacy is lower than that of the ones going to the furthest one.

At last a dubious distribution was created as to test the performance of the developed algorithm. For this distribution the three remaining variables are used, i.e., dubious3KmPerc, dubious5KmPerc and dubious10KmPerc. For this particular case it was necessary to create a distribution that did not fall under any of the two previously described categories. In order to do this the following values were used: 50%, 25% and 25% for the dubious3KmPerc, dubious5KmPerc and dubious10KmPerc respectively. As can be seen there is an equal distribution to the second closest and furthest pharmacies, which is an inconclusive result. It is not possible to detect with certainty that a fraudulent behaviour is occurring in this scenario.

In the following sections the metrics are discussed and two case studies are presented with the obtained results for the rule based algorithm and the geo-location based algorithm respectively.

The dataset used is composed of approximately 800 thousand prescriptions and roughly 82 million records of geo-location data which is equivalent to about 820 thousand geographical distributions.

4.1 Confusion Matrix

A confusion matrix is a table often used to describe the performance of a classification model, or classifier, on a set of labelled test data. An example can be seen in Table 4.2.

Using the values contained in this matrix it is possible to calculate all sorts of metrics including accuracy, precision and recall.

Table 4.2: Structure of a confusion matrix

		Predicted Class	
		0	1
True Class	0	True Negatives	False Positives
	1	False Negatives	True Positives

4.2 Accuracy

Accuracy can be defined as how close a measured value is to the actual, i.e., true value. To calculate the accuracy, true positives and true negatives are summed and then divided by the total amount of cases. The result gives the percentage of correctly identified cases, meaning the higher the value the better the detection. The formula for calculating the accuracy is presented in Formula 4.1, where TP, TN, FP and FN represent, respectively, True Positives, True Negatives, False Positives and False Negatives.

$$(TP + TN)/(TP + TN + FP + FN) \tag{4.1}$$

However this measure can sometimes be deceiving. For example, if a model that only predicted legitimate cases was used, then the accuracy would be low but a high number of fraudulent patients would not be caught (a high number of false negatives would occur). On the other hand, if a model that only predicts fraudulent cases is used, then the accuracy is high but a lot of legitimate patients are detected as being fraudulent (a high number of false positive results would occur).

For this reason precision and recall are used, since they give a more certain result than accuracy. They are explained in the next section.

4.3 Precision and Recall

There are two measures that give a more correct result than the accuracy alone. They are precision and recall. Precision gives the amount of positive cases correctly labelled and recall provides the amount of positive cases that were collected from the existing total.

The precision is given by Formula 4.2:

$$TP/(TP + FP) \tag{4.2}$$

While the recall is given by Formula 4.3:

$$TP/(TP + FN) \tag{4.3}$$

Where TP, FP and FN have the same meaning as the ones present in the accuracy section.

Given that the topic of this work is related to fraud detection, the most relevant metric is recall since it is crucial that most of the fraudulent cases are identified rather than having many cases labelled as fraudulent correctly. It is more desirable to have a greater number of cases identified, even if falsely labelled as true, than having many correctly labelled cases but missing many correct ones as well.

4.4 K-Fold Cross-Validation

This method is used to obtain a better approximation of the performance achieved by the system. The data is split in K equal parts where k-1 are used to train the system and the last set is used for testing. All sets are used for testing meaning that every set is used in the training stage as well as the testing phase. In the end all the results are gathered and the mean value is obtained.

This method has a great advantage against the previously mentioned ones since it prevents over-fitting. By using every subset as training and test, the model is never adapted to a specific unchanged dataset which allows for more realistic results.

Since the most relevant metric for the topic of this work is recall, as explained in the previous section, the K-Fold Cross Validation will focus on testing this measure.

4.5 Case Study 1

In this case study the prescriptions are analysed. The metrics described above are calculated and discussed.

The first performance metric is the confusion matrix, that can be seen in Figure 4.1.

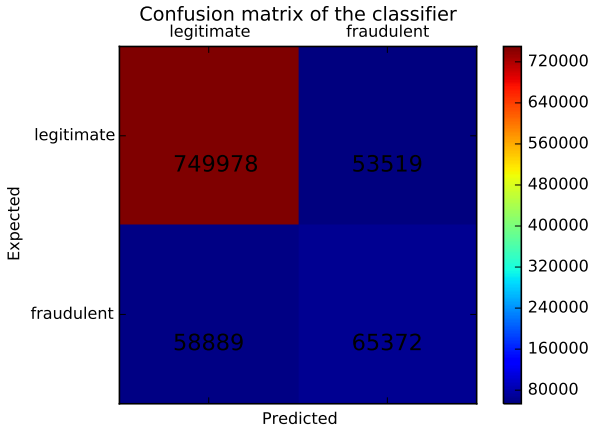

Figure 4.1: Confusion matrix of the prescription dataset.

The accuracy using this dataset was of approximately 0.88, or 88% which is a good result. But as seen above, this measure can not be completely trusted.

In terms of precision the obtained result was of approximately 0.55 which is an average result and the recall was roughly 0.53 which is also an average result. As mentioned previously the goal is to maximize the recall since, in the topic of this work it is more important to obtain the maximum number of positive results, even if false positives, because it is less costly to falsely identify a patient as fraudulent than the opposite.

As can be seen the recall is not very high and this is due to the number of dubious cases which were identified as legitimate (negatives) when they should have been identified as positives (which means they were false negatives). This implies that the recall only identified roughly half of all the positive cases, since the other half is composed by the false negatives.

In order to improve the recall, one can reduce the percentage of dubious cases from 2% to 1% or 0.5%.

If 1% of dubious cases are used, a precision of 0.38 and a recall of 0.69 is obtained, meaning that a better recall is obtained. However the precision changed as well. This happens because precision and recall are inversely related, i.e., if one gets a higher result the other will necessarily have a lower result. Using 0.5% of dubious cases increases the recall even more, obtaining a value of 0.23 for the precision and 0.82 for the recall.

As the number of dubious cases diminishes the recall gets higher. Values lower than 0.5% would not be significant so they will not be calculated.

A representation of the above results can be seen in Table 4.3.

Lastly a K Fold cross validation method was applied to the dataset with K = 10. This value of K was chosen since, as can be seen in [29], using a higher number of folders would make the sets overlap, thus not adding new information.

Table 4.3: Variation of precision and recall with dubious case percentage

		Precision	Recall
Dubious	2%	0.55	0.53
Cases	1%	0.38	0.69
Percentage	0.5%	0.23	0.82

So, using a K = 10 the obtained result is, for the recall, 0.33. The recall was used since, as mentioned before, the primary goal is to maximize the recall for the topic of this work. As can be seen this result is not very high, implying that the implemented algorithm would not work very well with new data. However, if the value of K is changed for example to 15 or 20, a different recall is obtained, closer to the one seen above, with the original dataset. So for K = 15 the recall is 0.53, roughly the same as seen above. Using K = 20 a recall of 0.53 is attained. So, as can be seen, K values higher than 10 result in the same recall.

The different values of K and the obtained recall can be seen in Table 4.4.

Table 4.4: Recall for different values of K

		Recall
K	10	0.33
	15	0.53
	20	0.53

An example of a detected case generated by the data generator is presented below:

Table : prescription Record $->$ 19 | 722122524WJ | 1514686183 | 4305876060 | 7-10-2009 | 13-10-2009 | levofloxacin | 8.69 | Anti-Depressant | 40.9002 | -76.0054 | '722122524WJ' | 40.90019449756683 | -75.96970561033231 | 6.08 | 2.61

As can be seen the record comes from the table prescription and in this case it was detected since it broke the 5 day execution period rule, that states that a patient can not take more than 5 days between the issue and the execution date of a prescription. In this case the issue date was 7-10-2009 and the execution date was 13-10-2009, thus 6 days have passed and the prescription is potentially fraudulent, i.e., this case is a True Positive.

An example of a True Negative generated by the data generator is as follows:

Table : prescription Record $->$ 41 | 665134526BA | 1514461347 | 4305875032 | 5-08-2010 | 9-08-2010 | Metformin | 8.69 | Anti-Diabetic | 40.9002 | -76.0054 | '532145532WJ' | 40.90019449756683 | -75.96970561033231 | 6.08 | 2.61

In this example there are no rules being broken, therefore it is considered to be a True Negative by the algorithm and as such it is not detected as fraudulent.

An example of a False Negative generated by the data generator is as follows:

Table : prescription Record $->$ 235 | 665125726BA | 1514461632 | 4305875126 | 4-06-2010 | 7-06-

2010 | Trazodone | 8.69 | Anti-Depressant | 40.9002 | -76.0054 | '532145532WJ' | 40.90019449756683 | -75.96970561033231 | 6.08 | 3.48

This case was predicted by the data generator as legitimate when in fact it was a dubious case. The reason it was detected was because the patient contribution is 3.48 which, if the total value of the drug, 8.69, is subtracted to it and then that value is divided by 8.69, the obtained result is 0.60, i.e., (8.69 - 3.48)/8.69 = 0.60. This is an indication that this case is a dubious case where instead od the SSB contributing with 70% or 50%, the fraudulent case, it contributes with 60% and therefore the algorithm did not detect any fraud when in fact the contribution percentage was not the expected.

4.6 Case Study 2

In this case study the geo-location dataset is analysed. The same method used in the first case study is applied in this one as well.

The first performance metric is the confusion matrix, that is represented in Figure 4.2.

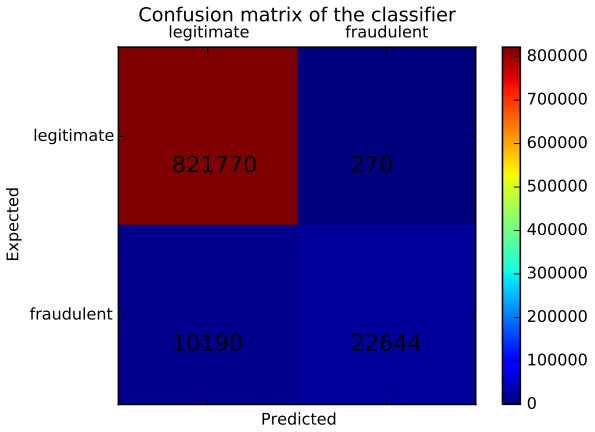

Figure 4.2: Confusion matrix of the geo-location dataset.

The accuracy using this dataset was of approximately 0.99, i.e. 99%, which is a very good result. But, once again, this measure is not enough to accurately measure the system's performance.

In terms of precision the obtained result was of approximately 0.99 which is a very high result and the recall was roughly 0.69 which is a good result.

As before the goal is to maximize the recall. Using the same method as before, if the dubious cases are reduced in half, the obtained precision is of approximately 0.98 while the recall increases to roughly 0.82. Using a 0.5% dubious case rate, the precision and recall would be 0.95 and 0.90 respectively. As can be seen using this dataset it is also possible to increase the recall by changing the percentage of dubious cases. The presented values can be seen in a more structured way in Table 4.5.

Table 4.5: Precision and recall values with distinct dubious case percentages

		Precision	Recall
Dubious	2%	0.99	0.69
Cases	1%	0.98	0.82
Percentage	0.5%	0.95	0.90

The last measure to test is the K Fold cross validation. In this dataset the value K = 10 is also used, the reason being the same as stated in the previous case study.

Using K = 10 the attained recall is 0.69 which is a good value. This value is also roughly the same as the observed recall in the original dataset. As can be seen, this algorithm is more likely to perform very well with new data than the one used in the previous case study.

If different values of K are used, for example 15 and 20, the recall values are both 0.69 which implies that, once again, the algorithm should work very well with new data.

The results can be seen in Table 4.6.

Table 4.6: Recall values for variable K

		Recall
K	10	0.69
	15	0.69
	20	0.69

A few examples are given below for cases generated by the data generator. Since this dataset is composed of sets of 100 records each, representing one single distribution, it was decided that the output of this algorithm would only include the last record as to facilitate the reading and analysis. Also, presenting all the records would not make it possible to visually understand if a distribution is normal or fraudulent due to all the calculations of the distances being performed by the algorithm it self, thus the extra information would not be useful for the user. When sent to experts, it is easy to locate the record in the database and analyse the complete distribution. An example of the output given by the algorithm when a fraudulent case is detected is presented below:

Table : locationDistribution Final Record (100 record distribution) $->$ 23402000 | 41.4181 | -76.046 | 41.41809439626989 | -76.01002244255723 | 99

An example of a True Negative is when a distribution follows what was defined in the architecture as a normal distribution, that is, when the 60% of the patients go to the pharmacy located in a 3 Kilometre radius, the closest one to their location, 30% go to the one in a 5 Kilometre radius, the second closest and finally 10% go to the one located in a 10 Kilometre radius, the furthest one.

A True Positive, on the contrary, is a case where a fraudulent distribution is present and the data generator detects it as such.

A False Negative, in this dataset, is a distribution that is perceived as legitimate by the algorithm but it is, in fact, a fraudulent one. A case generated by the data generator that is classified as a False Negative is a dubious distribution, where 50% of the patients go to the nearest pharmacy, at 3 Kilometres, 25% of the patients go to the second closest pharmacy, located at 5 Kilometres, and finally 25% of the patients go to the furthest pharmacy. Since this distribution is different from the fraudulent one, the algorithm does not detect this case as a fraudulent one and, therefore, it is considered legitimate when in fact it is fraudulent.

A False Positive occurs when a case is predicted as being fraudulent when in fact it was legitimate. In this dataset such a case occurs due to the following reason. When a dubious case is generated, the two more far away pharmacies have a 25% chance of being visited by the patients in the distribution. However, since the probabilities are calculated with a random number, there are cases in which the distribution will not be perfectly divided where 25% of patients go to each. So, as a result, some cases end up having a distribution where 24% goes to the second closest pharmacy and 26% goes to the furthest pharmacy, thus transforming a case that was originally created as being a dubious, i.e. undetectable by the algorithm case, into a fraudulent one and detecting it as such.

4.7 Chapter Summary

In this chapter the implemented algorithms were tested using a variety of metrics, such as accuracy, precision, recall and K-Fold cross-validation. The rules based algorithm obtained a lower score than would be expected, in respect to the recall metric in the K-Fold cross-validation. However it was possible to observe that the geo-location based algorithm obtained good results in every metric meaning it was very effective in the detection of fraudulent distributions and it is expected to perform well with new data since it obtained a good result in the K-Fold cross-validation metric.

5

Conclusion and Future Work

Contents

5.1 Conclusion

It was shown that the healthcare system is subjected to numerous fraudulent actions committed by a variety of entities. The most common types of fraud were presented and discussed in section 2. From analysing the literature, two algorithms (geo-location based and medical rule based) were chosen to incorporate the designed solution. An architecture for a possible solution was constructed to meet the goals described in subsection 1.1. A novel approach was taken by designing and implementing a data generator capable of creating data to be used in a plethora of algorithms. The detection algorithms were tested with a variety of metrics, including accuracy, precision, recall, and K-Fold cross validation. The obtained results were satisfactory, with the rule based algorithm having a lower score in the K-Fold cross validation and the geo-location based algorithm scoring a good result meaning on average the system performed well.

In conclusion it was seen that there are many algorithms to be used when solving the problem of detecting fraud in the healthcare system, as much as there are different types of fraud. Therefore it was important to use algorithms that detected distinct types of fraud in order to broaden the portion of fraud the system can predict and, consequently, making it more robust.

5.2 Future Work

In this section some possible improvements to the solution are presented.

- Transform the system into a modular design with each algorithm being a module, as to allow new algorithms to be implemented and added independently, in a simple way, as well as adding the accompanying data generator code for that specific algorithm;
- Implement more algorithms to broaden the detection capabilities of the system;
- Discover more information about how the prescriptions are structured as to create a more authentic dataset;
- Use real data to test and perfect the detection algorithms and to create a more realistic dataset;
- Design and implement a system to present the data in a visual way, plotting graphics to represent the distributions and clustering the prescriptions according to the type of fraud in a graphical way, to allow for a better understanding of the obtained results;

Bibliography

[1] Y. Yang, M. Manoharan, and K. S. Barber, "Modelling and analysis of identity threat behaviors through text mining of identity theft stories," in *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*. IEEE, 2014, pp. 184–191.

[2] E. H. Humaid and T. Barhoum, "Water consumption financial fraud detection: a model based on rule induction," in *Information and Communication Technology (PICICT), 2013 Palestinian International Conference on*. IEEE, 2013, pp. 115–120.

[3] P. K. Panigrahi, "A framework for discovering internal financial fraud using analytics," in *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*. IEEE, 2011, pp. 323–327.

[4] A. Shen, R. Tong, and Y. Deng, "Application of classification models on credit card fraud detection," in *2007 International Conference on Service Systems and Service Management*. IEEE, 2007, pp. 1–4.

[5] Z. Yongbin, Y. Fucheng, and L. Huaqun, "Behavior-based credit card fraud detecting model," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009.

[6] S. Liu, L. M. Ni, and R. Krishnan, "Fraud detection from taxis' driving behaviors," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 1, pp. 464–472, 2014.

[7] V. Rawte and G. Anuradha, "Fraud detection in health insurance using data mining techniques," in *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*. IEEE, 2015, pp. 1–5.

[8] M. Ahmed and M. Ahamad, "Combating abuse of health data in the age of ehealth exchange," in *Healthcare Informatics (ICHI), 2014 IEEE International Conference on*. IEEE, 2014, pp. 109–118.

[9] M. Suleiman, R. Agrawal, C. Seay, and W. Grosky, "Data driven implementation to filter fraudulent medicaid applications," in *IEEE SOUTHEASTCON 2014*. IEEE, 2014, pp. 1–8.

[10] U. Srinivasan and B. Arunasalam, "Leveraging big data analytics to reduce healthcare costs," *IT Professional*, vol. 15, no. 6, pp. 21–28, 2013.

[11] J. Li, K.-Y. Huang, J. Jin, and J. Shi, "A survey on statistical methods for health care fraud detection," *Health care management science*, vol. 11, no. 3, pp. 275–287, 2008.

[12] H. He, J. Wang, W. Graco, and S. Hawkins, "Application of neural networks to detection of medical fraud," *Expert Systems with Applications*, vol. 13, no. 4, pp. 329–336, 1997.

[13] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedreschi, "A classification-based methodology for planning audit strategies in fraud detection," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 175–184.

[14] G. J. Williams and Z. Huang, "Mining the knowledge mine," in *Australian Joint Conference on Artificial Intelligence*. Springer, 1997, pp. 340–348.

[15] W.-S. Yang, "A process pattern mining framework for the detection of health care fraud and abuse," Ph.D. dissertation, National Sun Yat-sen University, 2003.

[16] A. Tagaris, G. Konnis, X. Benetou, T. Dimakopoulos, K. Kassis, N. Athanasiadis, S. Rüping, H. Grosskreutz, and D. Koutsouris, "Integrated web services platform for the facilitation of fraud detection in health care e-government services," in *2009 9th International Conference on Information Technology and Applications in Biomedicine*. IEEE, 2009, pp. 1–4.

[17] W.-S. Yang and S.-Y. Hwang, "A process-mining framework for the detection of healthcare fraud and abuse," *Expert Systems with Applications*, vol. 31, no. 1, pp. 56–68, 2006.

[18] A. Tagaris, P. Mnimatidis *et al.*, "Implementation of a prescription fraud detection software using rdbms tools and atc coding," in *2009 9th International Conference on Information Technology and Applications in Biomedicine*. IEEE, 2009, pp. 1–4.

[19] N. Christiannini and J. Shawe-Taylor, "Support vector machines and other kernel-based learning methods," 2000.

[20] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275–300, 2004.

[21] Q. Liu and M. Vasarhelyi, "Healthcare fraud detection: A survey and a clustering model incorporating geo-location information," in *29th world continuous auditing and reporting symposium*, 2013.

[22] S. Zhu, Y. Wang, and Y. Wu, "Health care fraud detection using nonnegative matrix factorization," in *Computer Science & Education (ICCSE), 2011 6th International Conference on*. IEEE, 2011, pp. 499–503.

[23] "Frobenius norm," http://mathworld.wolfram.com/FrobeniusNorm.html, accessed: 2016-01-18.

[24] "List of the top 200 drugs of 2016," http://www.pharmacy-tech-test.com/top-200-drugs.html, accessed: 2016-08-22.

[25] "Pharmaceutical market january 2016," Internal study conducted by hmR - Health Market Research.

[26] "Survey with information regarding the distance between residence and workplace," http://forum.autohoje.com/off-topic/91243-distancia-de-casa-ao-trabalho-4.html, accessed: 2016-09-21.

[27] M. Kirlidog and C. Asuk, "A fraud detection approach with data mining in health insurance," *Procedia-Social and Behavioral Sciences*, vol. 62, pp. 989–994, 2012.

[28] "Amazon web services website," https://aws.amazon.com/, accessed: 2016-10-15.

[29] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.