University of Alberta

ON THE APPLICATION OF MULTI-CLASS CLASSIFICATION IN PHYSICAL THERAPY RECOMMENDATION

by

Jing Zhang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2012

Edmonton, Alberta

Examine Committee

Osmar R. Zaiane, Department of Computing Science

Douglas Gross, Department of Physical Therapy

Vadim Bulitko, Department of Computing Science

Jeremy Beach, Department of Medicine

Abstract

Selecting appropriate rehabilitation treatments for injured workers has been a challenging task for clinicians and health care funders. Currently, clinicians are unable to identify the optimal treatment for a patient with absolute confidence and are looking for assistance from other research fields, such as Machine Learning.

This thesis aims at building a knowledge-based clinical decision support system using machine learning algorithms. We have found that with proper data preprocessing, the RIPPER algorithm can extract meaningful, editable and interpretable decision rules from a severely imbalanced multi-class clinical dataset.

Moreover, the extracted rule set is integrated into our prototype Work Assessment Triage Tool (WATT), a web-based online decision support system. It has an easy-to-use interface and provides useful recommendations to help clinicians make better decisions.

Table of Contents

List of Tables

List of Figures

List of Publications

- Jing Zhang, Douglas Gross, and Osmar Zaiane. On the application of multi-class classification in physical therapy recommendation (submitted). *IEEE International Conference on Data Mining*, 2012
- Gross DP, Zhang J, Steenstra I, Cooper J, Barnsley S, Haws C, McIntosh G, Amell T, and Zaiane O. Development of a computer-based decision-support tool using machine-learning strategies for selecting appropriate rehabilitation interventions. *Odense International Forum XII Primary Care Research on Back Pain*, 2012
- Gross DP, Zhang J, Steenstra I, Cooper J, Barnsley S, Haws C, McIntosh G, Amell T, and Zaiane O. Development of a computer-based decision-support tool using machine-learning strategies for selecting appropriate rehabilitation interventions. *BMJ Occupational and Environmental Medicine*, 2012

Chapter 1

Introduction

Work-related musculoskeletal injuries have been responsible for the long-time lay off and high cost in health care among injured workers [12][36][43][34][8][48][42]. While the majority of individuals with such conditions return to work quickly, a small minority suffers from delayed recovery and this has become heavy burdens on their personal, social and economic lives [9][27]. Ideally, effective rehabilitation programs for those most at risk would be identified. This is a classification process which involves assigning claimants to appropriate rehabilitation programs that lead to successful return-to-work (RTW) based on their clinical and work-related characteristics. However, current referral for rehabilitation program follows a trial-and-error fashion since clinicians are not completely confident of identifying the best intervention which patients will respond to among various options [24] and thus the rate of successful RTW among those injured workers is not high and could be further improved.

As mentioned, most injured worker return to work quickly. For those whose recovery have not occurred within 4 to 6 weeks, further assessment followed by an intervention is necessary to avoid chronic pain and disability. Various interventions are available including physical conditioning or functional restoration programs [52], worksite-based interventions [53], and interdisciplinary biopsychosocial rehabilitation (i.e. chronic pain management programs) [37] [23]. If a patient going through an intervention successfully returns to work within a pre-determined time,

the intervention is considered as appropriate with a positive outcome. Otherwise, it is considered as a failure with a negative outcome and the patient has to go through subsequent assessments and interventions. Although it is possible that multiple rehabilitation programs can lead to return-to-work for a patient, we cannot find out what they are since it is impossible to let a patient go through multiple programs at the same time in order to observe the outcomes. We could let a patient go through a sequence of different programs, but each time the patient completes a program, he's not the same person anymore since his health conditions may have changed due to the previous rehabilitation. Therefore, an important assumption in this thesis is that for each patient there exists only one appropriate program. If a patient is correctly categorized into the true appropriate program, he will absolutely return to work. Otherwise, there will be no successful RTW. Under the assumption above, we could determine a patient's return-to-work status in advance based on the classification result.

The main purpose of this thesis is to construct a classification model using machine learning that categorizes an injured worker into his own appropriate rehabilitation program. Machine Learning (ML) algorithms are capable of mining useful patterns in various forms from empirical data (i.e., past records of patients including their characteristics, rehabilitation program and program outcome) to generate class descriptions to distinguish different categories of objects (i.e., clinical characteristics that fit a certain type of rehabilitation programs). If the model could achieve good classification performance (e.g., higher classification accuracy or other measurements than human experts'), it would facilitate the rehabilitation recommendation process and improve the success rate of RTW.

This model will be included into an easy-to-use Clinical Decision Support System (CDSS), which is receiving attentions from an increasing number of researchers in clinical practice. Computerized CDSS has shown the potential to augment human clinical decisions [50][51].

Most of the current CDSS utilize a knowledge base (rules or guidelines) manually constructed by human experts. Machine learning algorithms are capable of constructing a knowledge base automatically without human inputs and have the

potential of discovering something the human experts have overlooked. Together with the experts' own expertise which could be injected into the knowledge base, a CDSS equipped with a model generated from ML algorithms would become a powerful assistant in the intervention selection process.

1.1 Thesis Statements

This dissertation elaborates on building a clinical decision support system (CDSS) using machine learning algorithms. We aim at addressing the following statements:

- TS1: Machine learning algorithms could create meaningful, editable and interpretable models (e.g. a set of Disjunctive Normal Form (DNF) rules) as the knowledge base of the clinical decision support system.
- TS2: Previous methods handling class imbalance in classification are only validated on binary datasets. Their effectiveness is unknown for multi-class datasets. Combinations with class decomposition may be more effective.
- TS3: The rule-based classification model could provide multiple recommendations. The quality of a recommendation could be measured by the quality of the underlying rules through different metrics. Confidence and Chi-Square could be two promising measurements.

1.2 Thesis Contributions

In this dissertation, we build a web-based decision support system that incorporates a knowledge base (consisting a set of classification rules generated from machine learning algorithms). This approach combines the advantages of the knowledge-based (KB-DSS) and the non-knowledge-based DSS (NKB-DSS) together:

- We use machine learning algorithms to generate a classification model efficiently without the need for expert input (an advantage of NKB-DSS).

- The machine learning algorithms can mine meaningful rules. Some might have not been discovered by the domain experts before (another advantage of NKB-DSS).

- By using rule-based machine learning algorithms, we obtain a classification model consisting of a set of Disjunctive Normal Form (DNF) rules which are interpretable and editable by the clinicians (an advantage of KB-DSS).

During the process of constructing our knowledge base (a multi-class classification model), we have encountered severe class imbalance and overlap in our data. In order to mitigate their negative impacts, we have compared and analyzed various data preprocessing methods and provide insights of how popular methods in literature work on real life clinical datasets. We have found that the combination of SMOTE + Numeric-Variable-Encoding + Supervised Discretization + Tomek Links/NCR + RIPPER can produce meaningful recommendation rules as evaluated by our domain expert and have shown promising classification potential in independent test evaluations.

Additionally, we present our prototype system Work Assessment Triage Tool (WATT). Based on the individual and clinical characteristics of a worker, WATT provides multiple rehabilitation recommendations. All recommendations can be sorted based on different criteria and the supporting rules of each recommendation can be displayed to explain why such a recommendation is made. The system provides a recommendation pool for the clinicians to choose from so that they can make better decisions and possibly increase the efficiency of the decision-making process.

1.3 Thesis Organization

The rest of this dissertation is organized as follows. In chapter 2 we briefly introduce the clinical background of rehabilitation treatment in physical therapy, the concept and property of (clinical) decision support system, and machine learning. In chapter 3, we analyze the datasets we are using along the timeline of this research. Chapter

3 also includes the details of the evaluation metrics in our experiments. Chapter 4 introduces the machine learning algorithms, data preprocessing and visualization methods in our experiments. Chapters 5 and 6 detail two different sets of experiments design, report their results and analysis respectively. Finally, we conclude the overall contributions and future study of this thesis in Chapter 7.

Chapter 2

Background and Related Works

2.1 Rehabilitation process and classification of injured workers

Figure 2.1 shows a continuum care model for injured workers (provided by Workers' Compensation Board-Alberta). In general each injured worker receives a return-to-work (RTW) assessment (including personal and clinical characteristics) after a few weeks of recovery therapy. The physician and case manager make a rehabilitation recommendation based on the assessment result, however, the case manager has the right to override the physicians decision. The feature of each program is listed as follows [23]:

- "other" intervention (prog0): No rehabilitation or a single service provider. Basically it means a patient′s health condition does not require any treatment or no treatments could help the patient return to work.
- Complex Service (prog3): Comprehensive pain management program for patients with chronic pain and multiple complex barriers to RTW.
- Provider Site Based Service (prog4): Interdisciplinary rehabilitation at a designated rehabilitation facility.

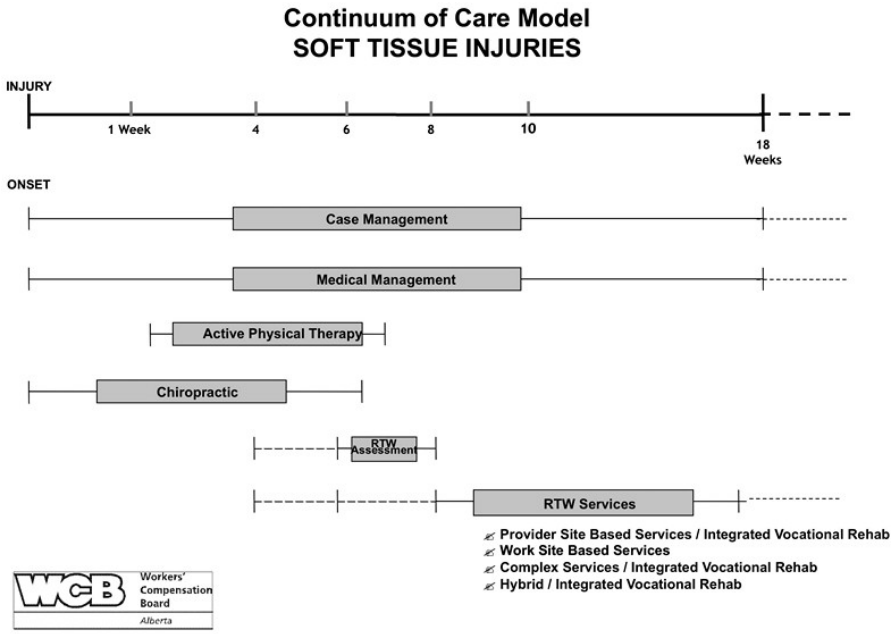

Figure 2.1: WCB-Alberta Continuum of Care Model for the Management of Claimants with Soft-Tissue Disorders (2010 version)

- Work Site Based Service (prog5): The intervention takes place at a worksite instead of at a rehabilitation facility.
- Hybrid (prog6): A hybrid program of prog4 and prog5.

The classification of the injured workers is to map a patient R to a rehabilitation program y:

$$R : (f_1, f_2, ..., f_n) \rightarrow y \in \{prog0, prog3, prog4, prog5, prog6\}$$

where y is a correct class for R iff

$$\exists \text{ A clinician or case manager recommended } y \text{ for } R$$

$$and$$

$$R_{\text{subjected to } y} \text{ RTW in a pre-determined time } t.$$

In this thesis, the pre-determined time t is defined as the 30th day after a patient's admission to assessment. f_1 to f_n stands for the characteristics of a patient.

2.2 Clinical Decision Support System

Decision Support Systems (DSS) are computer-based information systems that assist people to make decisions more effectively and efficiently. Although there is no universally accepted definition of DSS [4], the followings are the common features that DSS share:

- DSS are designed to assist, rather than replace, the decision-makers. They should allow the decision makers to inject their own insights into the decision process.
- They incorporate models in various formats (such as a knowledge base or intelligent algorithms) as well as the data.
- They should improve the quality of the decision with an easy-to-use access.

Clinical Decision Support Systems (CDSS) are specifically designed for decision makers in health care to improve tasks such as medical diagnosis or clinical prognosis.

CDSS are usually categorized into two types: Non-Knowledge based and Knowledge based CDSS. The knowledge-based CDSS are built on a set of IF-THEN rules and associations [7]. These rules are usually provided by domain experts [40][10] and decisions are made by testing instances against the rules. The system allows the users to edit the knowledge base with their own insight. The Non-Knowledge-based CDSS incorporate intelligent algorithms (i.e., machine learning) to extract useful patterns from past experiences or in the clinical data. The clinicians are less willing to use it since the extracted patterns are usually not interpretable [7]. A good combination is to use rule-based machine learning algorithms to generate rule-based models as the knowledge base. In this way, we can utilize the advantage of the algorithms while providing model interpretability and editing.

2.3 Machine Learning

2.3.1 Introduction

Machine Learning aims at designing algorithms to analyze and induce knowledge from empirical data. Such knowledge is applied in many tasks such as classification, regression and clustering.

2.3.2 Supervised Learning

The data used by machine learning algorithms are represented using a set of features. Those features can be either continuous or discrete. If a known label (or a pre-specified output value) is given to each instance in the dataset, the learning process is called supervised learning.

Classification

Classification is part of the supervised learning problem. In classification, machine learning algorithms learn from pre-labeled data to discriminate instances from different classes. Given a test instance, the trained model categorizes the instance by assigning a class label to it.

Regression

Regression is similar to classification except that it learns from data with known output which is continuous rather than discrete (class label).

2.4 Related Work

Researchers have been developing clinical decision support systems to assist decision making in various clinical situations including acute care and chronic disease management [51][50], drug prescribing and management [32], the diagnosis of low back pain [40] and lumber spine imaging modalities [10]. These systems

share a common ground that they are based on a knowledge base (such as a set of guidelines or rules) manually constructed by human experts without the participation of intelligent computer algorithms. Usually clinicians tend to not fully trust non-knowledge-based systems since the evidence generated by many computing algorithms is not interpretable [7].

Regarding the class imbalance problem, researchers in machine learning and data mining have been working on various solutions including resampling techniques (oversampling the minority class examples [41][14] and undersampling the majority class examples [38]), recognition-based learning (one class learning) [35] and cost-sensitive learning [20]. However, class imbalance may not be the only reason that deteriorates the classifier′s performance since Batista [44] have reported that for linearly separable dataset the classifier′s performance is not susceptible to imbalance. Class overlap, however, is actually another factor that hinders the classifier′s performance. A variety of methods can be applied to solve this problem including Edited Nearest Neighbor (ENN) [57], Nearest Neighbor Cleaning Rule (NCR) [39] and Tomek Links [1]. These methods were originally proposed as under-sampling techniques but can also be used as data cleaning methods. Combinations of the oversampling techniques and data cleaning methods have also been proposed to balance the dataset and clean up the overlaps to form better defined data clusters [6]. There are other variations of the above methods. Borderline SMOTE [30] only samples on the minority class data on the borderline since only the borderline data defines the boundary between different class clusters. SMOTEBoost [15] is a combination of SMOTE and Boost. Instead of assigning larger weights on hard examples (usually the minority class data), SMOTEBoost samples on the minority classes to change the distribution of the dataset. ENN and NCR are actually derived from Condensed Nearest Neighborhood (CNN) [31]. CNN tends to remove redundant majority class data and is considered as an under-sampling technique. Table 2.1 provides an brief overview of methods proposed by various researchers.

Methods	References	Summary
Random Oversampling	[40]	Simple duplication of minority class examples
SMOTE	[14]	Generates synthetic data for minority classes
Borderline SMOTE	[30]	Variation of SMOTE (only sample on borderline minority data)
SMOTEBoost	[15]	Combination of SMOTE and Boosting
Condensed Nearest Neighborhood (CNN)	[31]	Removes redundant majority class data (undersampling)
Edited Nearest Neighbor (ENN)	[57]	Remove noisy data based on K-NN comparison
Nearest Neighbor Cleaning Rule (NCR)	[39]	Variation of ENN (only remove majority class data)
Tomek Links	[1]	Remove borderline data or noise
SMOTE + Tomek Link	[6]	Combination of Oversampling and data cleaning
SMOTE + ENN	[6]	Combination of Oversampling and data cleaning
Recognition-based Learning	[35]	Learn only the minority classes
Cost-sensitive Learning	[20]	Provide cost information to bias the classifier (in a good way)
AdaBoost.NC	[55]	Handling multi-class imbalance without class decomposition

Table 2.1: Overview of methods for solving class imbalance

Chapter 3

Data Analysis

The data in our research are extracted from a Canadian provincial compensation database at Workers Compensation Board (WCB) Alberta. At the beginning of this research, the data were not finalized and kept evolving until late December 2011. Therefore, we have 3 datasets in total and they are briefly explained as follows. Different datasets are used in different stages of this research.

3.1 Preliminary Dataset (June 2011 - Sept. 2011)

This dataset consists of only 100 records. Almost half of them have missing values in a large portion of variables. Therefore, it is not meaningful to use this dataset to train a classification model. But it did provide useful knowledge about the nature of the variables.

3.2 Second dataset (Sept. 2011 - Dec. 2011)

This dataset is much more evolved (with more features and instances) in comparison of the preliminary dataset. It consists of 5007 data and is used for the separate model approach (detailed in Chapter 5). Since it is possible that the different injuries could influence the clinicians' decision-making, we divide the dataset into three parts based on the patients′ injury type and analyze them separately. The three types of

injury are Leg Injury (LEFS), Arm Injury (DASH) and Low Back Injury (OMPQ).

These three datasets suffer from severe class imbalance as the original dataset does. Table 3.1 to 3.3 show the class distribution of these datasets respectively.

Class	prog0	prog3	prog4	prog5	prog6
num of records	680	18	374	4	75

Table 3.1: Class Distribution of LEFS

Class	prog0	prog3	prog4	prog5	prog6
num of records	1097	13	584	17	138

Table 3.2: Class Distribution of DASH

Class	prog0	prog3	prog4	prog5	prog6
num of records	857	12	628	8	152

Table 3.3: Class Distribution of OMPQ

3.3 Final dataset (Jan. 2012 - May 2012)

The final dataset consists of 14,484 records of claimants undergoing Return-To-Work (RTW) assessment and program. It has 200 features including the individual and clinical characteristics collected during the assessment and is our major focus in this research. As we mentioned in Chapter 1, not all records have a successful RTW result. If an intervention was unsuccessful, the claimant would have to go through subsequent interventions. In order to train a classification model that predicts successful interventions, the machine learning algorithms need to learn from only the successful records in the data instead of all of them. In other words, the algorithms should mine the relationship between the patients' characteristics and

their rehabilitation program in records with positive outcome. Therefore, we extracted 4876 successful records for the algorithms to learn a positive classification model (In fact, there are more records with successful RTW status but those with a large number of missing values are removed in data cleaning). The rest of the data consisting of unsuccessful records (without missing data severely) are used to train a negative model. The positive and negative models are explained in Section 7.3.

The extracted dataset is also highly skewed as shown in Table 3.4.

Class	prog0	prog3	prog4	prog5	prog6
num of records	1828	84	2286	96	582

Table 3.4: Class Distribution of the Final Dataset

Chapter 4

Machine Learning algorithms and Data Preprocessing Methods

In this chapter, we introduce the system design, the algorithms and data preprocessing methods we are using.

4.1 Machine Learning Algorithms

In this section, we briefly describe the following algorithms including Naive Bayes, C4.5, Associative Classifier, RIPPER and Support Vector Machine. We would like to compare algorithms from different categories on handling a multi-class classification problem.

4.1.1 Naive Bayes

The naive Bayes algorithm [49] applies the Bayes theorem to compute the likelihood that an unseen object belongs to a certain class C_i (i = 1, 2, ..., k) given the attribute values in the object. The algorithm assigns the object to the class with the maximum likelihood. It relies on a naive assumption that given the class label, all attributes are mutually independent. Although the assumption seems over-simplified, it has shown its competitiveness in many practical situations.

Equation 4.1 shows that given a test case *t* with feature values of f_1 to f_n, we can use Bayes rule to calculate the probability that *t* belongs to class C_i (i = 1, 2, ..., k) as follows.

$$p(C_i|f_1, f_2, f_3, ..., f_n) = \frac{p(C_i)p(f_1, f_2, f_3, ..., f_n|C_i)}{p(f_1, f_2, f_3, ..., f_n)} \tag{4.1}$$

Based on the assumption of conditional independence, Equation 4.1 can be represented by Equation 4.2 as follows.

$$p(C_i|f_1, f_2, f_3, ..., f_n) = \frac{p(C_i)\prod_{j=1}^{n} p(f_j|C_i)}{p(f_1, f_2, f_3, ..., f_n)} \tag{4.2}$$

Then according to the maximum a posteriori (MAP) decision rule, the test case *t* is classified by Equation 4.3 as follows.

$$classification(t) = argmax\, p(C_i)\prod_{j=1}^{n} p(f_j|C_i) \tag{4.3}$$

4.1.2 C4.5

C4.5 [46] is a well-known decision tree algorithm introduced to produce explicit models that users can interpret. Figure 4.1 shows a decision tree that describes a mortgage model. Each node in the tree represents a selected feature and the tree branches out based on the values in the node. The leaf node represents a class. A new instance is classified by testing the feature at each node and following the branch of the tree based on the observed value in the instance. This process repeats until the instance reaches the leaf node and it is assigned to the class represented by the leaf.

The decision tree is formed by choosing the most useful feature as the internal node recursively. Specifically, the algorithm calculates the information gain from splitting the data on each feature and the one with the highest information gain is chosen. This criterion is applied recursively from the root node to all internal nodes and terminates when a node contains instances from only one class.

C4.5 algorithm also has a pruning stage to simplify the tree and reduce the probability of overfitting the training data. The pruning is done by replacing an internal node by a leaf node in its subtree if such replacement results in an expect error rate that is no greater than that without pruning.

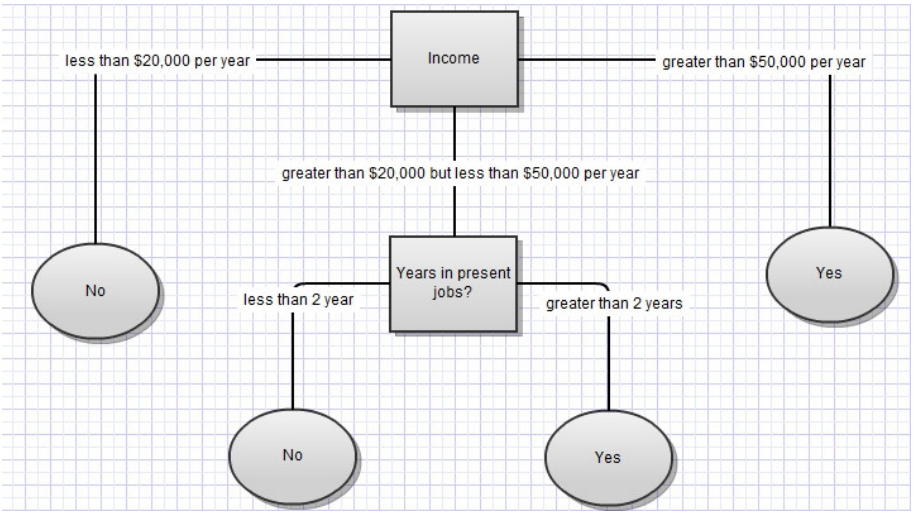

Figure 4.1: A Decision Tree Example

4.1.3 Associative Rule Classification By Category (ARC-BC)

Associative rule classification is derived from the association rule mining techniques. Instead of mining all possible rule associations, it only discovers associations between a set of features and a class label. We use the Associative Rule Classification-By Category (ARC-BC) [58] algorithm in this research. ARC-BC mines associative classification rules in each category separately and therefore does not overlook the minority classes when class imbalance is present. The rules that pass through a local minimum support threshold σ are gathered and grouped into different categories based on the associated class label. These rules form the final classifier. A given test instance could be covered by multiple rules with different class labels. To make the final decision, for each prediction, the algorithm calculates the average rule confidence of all rules supporting it. The prediction with the highest average rule confidence prevails.

4.1.4 RIPPER

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [16] is a classification rule learner that generates Disjunctive Normal Form (DNF) classification

rules. Each classification rule has a form of

$$r : (Rule\ Antecedent) \rightarrow y$$

The left hand side of the rule (Rule Antecedent) is a test of attribute conjunction while the right side of the rule is the class label. A rule r covers an instance t if t's attribute values satisfy the rule antecedent. In the following example, only $I2$ is covered by Rule r.

- r: $(Age > 30) and (Income > 6000) and (Status = Married) \rightarrow Mortgage = Yes$

Several instances:

- $I1$: $(Age = 29) and (Income = 7500) and (Status = Single)$
- $I2$: $(Age = 32) and (Income = 6100) and (Status = Married)$
- $I3$: $(Age = 35) and (Income = 4500) and (Status = Divorced)$

For a binary classification problem, the RIPPER algorithm generates classification rules in the following way:

1. Choose the smaller class as the positive class.
2. Set the other class as the negative (default) class.
3. Learn the rule set that separate the positive class from the default class.

Figure 4.2 presents how the RIPPER algorithm build a rule set in step 3. The RIPPER algorithm builds up the rule set by keeping to add a new rule at a time until it meets the stop criterion: either there are no positive examples left or the description length of the rule set after adding the new rule is d bits (usually 64 bits) larger than the previous smallest description length. Each time a new rule is extracted, all the examples covered by this rule are eliminated from the data.

The RIPPER algorithm builds a single rule in the following steps:

1. Split currently uncovered examples into a growing and pruning set.

```
Rule Set S=ø
While # of Pos> 0:
    RandomPartition(Pos, Neg) → GrowSet(Pos, Neg) and PruningSet(Pos, Neg)
    Rule=GrowRuleOn(GrowSet)
    Rule=PruneRuleOn(PruningSet)
    If MeetStopCriterion == True:
        Return S
    Else:
        Add Rule into S
        Eliminate both Positive and Negative Examples covered by Rule
Return S
```

Figure 4.2: RIPPER algorithm (Reproduced from Figure 1 in [16])

2. On the growing set, it starts with an empty rule (an rule with no antecedent).

3. Add a new condition into the rule antecedent as long as this addition maximize FOIL's information gain criterion [45].

4. Repeat Step 3 until no negative examples from the growing set are covered by this rule.

5. Prune this immediately on the pruning set.

To prune a rule, RIPPER deletes any final sequence of rule antecedents that maximizes v, which is defined as

$$v = \frac{p - n}{p + n}$$

The notation p and n stands for the number of positive and negative examples in the pruning set covered by this rule respectively.

RIPPER algorithm includes an optimization process as a postprocessing stage to improve the original rule set. It searches for possible variants of r by:

- Adding new conditions to extend the original rule

- Growing a new rule to replace the original rule

Each time a rule variant is generated, RIPPER compare the original rule set with the one using the variant instead of r. The rule set that minimizes minimum description length (MDL) principle [47] is then chosen. This process is repeated for each rule in the original rule set until all rules have been processed.

For multi-class problem:

1. RIPPER sort the classes in ascending order based on the class size.
2. It chooses the smallest class as the positive class and the rest is considered as the negative class.
3. A rule set for the positive class is learned.
4. Repeat step 2 and 3 for the next smallest class.

The rules generated from the RIPPER algorithm are ranked in ascending order based on the number of examples in the class. An unknown instance is tested against the rules in that order. The first rule that covers the test instance "fires" to make the classification and the testing phase ends.

However, since our CDSS need to provide multiple recommendations for clinicians to choose, we make the following modifications on the default RIPPER algorithm and refer to it as the alternate RIPPER algorithm (ARIPPER) in the rest of the thesis. For a given test instance, instead of firing the first rule that covers it, all covering rules are gathered. If the predictions of these rules are not consistent, ARIPPER groups rules with the same prediction together. In this way, the system outputs multiple recommendations, each with several underlying rules as evidence supporting it.

For a decision support system, it is sufficient to provide multiple recommendations for user to choose. To facilitate the users' decision-making process, the recommendations could be ranked based on their quality. The quality of a recommendation is measured by the strength of the supporting rules combined as a whole. To compare the strength of each group, we use the following four different measurements:

- Highest Average Rule Confidence **(HARC)**: HARC is adopted from the ARC-BC algorithm. It calculates the average rule confidence of all rules supporting a prediction.

- Single Rule with Highest Confidence **(SRHC)**: SRHC is a variation of the confidence-based measurement. Instead of measuring the average confidence like HARC, it only looks at the single rule confidence and uses the one with the highest confidence as a recommendation's quality.

- Highest Weighted Chi-Square **(HWCS)**: HWCS is a measurement adopted from CMAR, i.e., Classification based on Multiple Association Rules [2]. It calculates the weighted rule Chi-Square value of all rules supporting each recommendation. The weighted Chi-Square measure is defined by Equation 4.4 in CMAR [2]:

$$Weighted\chi^2 = \sum_{k=1}^{n} \frac{\chi^2\chi^2}{max\chi^2} \tag{4.4}$$

where *n* is the number of rules in a group. χ^2 stands for the Chi Square value of a single rule. $max\chi^2$ represents the upper bound of χ^2 and is defined in Equation 4.5 in CMAR [2].

$$max\chi^2 = (min\{sup(P), sup(c)\} - \frac{sup(P)sup(c)}{|T|})^2|T|e \tag{4.5}$$

where $|T|$ stands for the total number of data instance in the training data. For each rule R: $B \rightarrow c$, sup(B) and sup(c) stand for the support of the rule body B and the support of the class label c respectively. Additionally,

$$\begin{aligned} e = & \frac{1}{sup(p)sup(c)} + \frac{1}{sup(p)(|T| - sup(c)} \\ & + \frac{1}{(|T| - sup(P)sup(c)} + \frac{1}{(|T| - sup(P)(|T| - sup(c)} \end{aligned}$$

- Single Rule with Highest Chi-Square **(SRHCS)**: SRHCS looks at the Chi-Square of each single rule and uses the one with the highest Chi-Square value as the quality of a recommendation.

4.1.5 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm for data classification based on statistic learning theory [54]. It was first proposed by Vapnik [17] in 1995. It has shown good performance in fields such as text classification and image recognition. Given a set of labeled two-class instances $(x_1, y_1), (x_2, y_2)..., (x_n, y_n)$ where y belongs to $\{1, -1\}^n$, SVM aims to learn a hypothesis that maps **X** to **Y** by solving the following optimization problem:

$$\underset{\mathbf{w},b,\xi}{min} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

subject to

$$y_i(\mathbf{w}^T\phi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0.$$

C is the penalty parameter of the error term. ϕ is the function that maps **X** to a higher dimensional space. ξ_i is the slack variable which measures misclassification degree of x_i. Then SVM finds a linear separable hyperplane that separates the binary class data with the maximal margin in that space. The optimal hypothesis given by Equation 4.6.

$$f(\mathbf{X}) = \sum_{i=1}^{n} y_i\alpha_i K(\mathbf{X}_i, \mathbf{X}) + b \tag{4.6}$$

where $K(\mathbf{X}_i, \mathbf{X}) = (\phi(\mathbf{X}_i), \phi(\mathbf{X}))$ is called the Kernel function. The hypothesis can vary based on the kernel function used.

4.2 Data Preprocessing Methods

The class distribution in the dataset is severely imbalanced as shown in Chapter 3. Normally, with the presence of class imbalance the classifier may be biased towards the majority classes while the minority classes may be insufficient for learning.

Possible solutions for this circumstance can be organized into three categories: cost-sensitive learning, recognition-based learning and resampling. Since it is not

possible for us to obtain the cost of misclassification during the project, we focus on the last two techniques. Recognition-based learning is applied through the RIPPER algorithm. Resampling techniques are used in the data preprocessing.

We use a variety of known resampling techniques including 1) over-sampling: Random Oversampling [41] and SMOTE [14] and 2) data cleaning (undersampling) methods: Tomek Links [1], Edited Nearest Neighbor (ENN) [57], Neighbor Cleaning Rule (NCR) [39]. We use these methods not only to mitigate the class imbalance but also to clean up the noise (such as majority class examples invading the minority classes and vice versa) in the data space.

However, since these methods are only validated in the literature on the binary datasets, their effectiveness is unknown for multi-class imbalance problems [55]. So we apply the resampling techniques directly on the dataset (detailed in Chapter 5 and 6) as well as combining class decomposition with the resampling techniques (detailed in Chapter 6). We also compare the class decomposition approach with a new method called AdaBoost.NC [55]. The details of each method are described in the following sections.

4.2.1 Oversampling

Random Oversampling

Random Oversampling (RO) [41] simply replicates data from minority classes to balance the class distribution.

Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [14] stands for Synthetic Minority Oversampling Technique. It manipulates the feature values of minority data examples that are nearest neighbors to each other in order to form new synthetic minority class data. It generalizes the data space and thus can avoid overfitting to some extent.

4.2.2 Data Cleaning Methods

Tomek Links

If two data examples from different classes are the 1 nearest neighbors to each other, they form a Tomek Link (TL) [1]. And either both of them are borderline points or one of them is a noise invading the data space of the other class. Generally, both points in a Tomek Link can be removed.

Edited Nearest Neighbor Rule

Edited Nearest Neighbor (ENN) Rule [57] is an undersampling method that removes data examples whose class label differs from that of at least two of its three nearest neighbors.

Nearest Cleaning Rule

NCR [39] also finds each data example whose class label differs from the class of at least two of its three nearest neighbors. But unlike ENN, NCR aims at preserving the minority data by doing the following removal: If the target example belongs to the majority class, remove it. Otherwise, remove its nearest neighbors which belong to the majority class.

SMOTE + Tomek Links

The main reason for this combination is that the synthetic data from minority class might invade the majority class too deeply and with the cleaning of Tomek Links, we could avoid potential overfitting. This approach was first proposed by Batista [5] in Bioinformatics.

SMOTE + NCR

This combination shares the same reason as SMOTE + Tomek Links. Since NCR only removes data points from the majority classes, this combination aims at making more space for both original and synthetic minority class examples.

SMOTE + ENN

This combination also shares the same reason as SMOTE + Tomek Links. However, ENN is likely to remove more data than Tomek Links and NCR do since its data removal criterion is less strict.

Random Oversampling + Tomek Links/NCR/ENN

Although random oversampling does not cause the invasion problem of SMOTE, it is possible that the original data has class overlap and require data cleaning. So is the data after random oversampling.

4.2.3 AdaBoost.NC

AdaBoost.NC [55] is a new classification ensemble using negative correlation learning. It introduces diversity in the ensembles by exploiting ambiguity terms. It is similar to the AdaBoost [3] technique proposed by Freund and Schapire: it builds classifiers sequentially and introducing weights to training examples. Instead of simply assigning weights to hard examples, AdaBoost.NC introduces correlation information (obtained by measuring the difference among the current classifiers) into weights. It is considered as a cost-sensitive solution and claims that it is able to effectively handle multi-class imbalance without class decomposition [55].

4.2.4 Feature Selection, Encoding and Discretization

Feature Selection

Since the system has a limit of 30 variables as mentioned in Section 5.1, a feature selection process is required. Before we apply any automatic feature selection methods, we consulted the experts from the Department of Physical Therapy to check each variable and eliminate those that are absolutely irrelevant from the perspective of clinical practice. Further feature selection is done by the combination of Correlation-based Feature Subset Evaluation [29] and Linear Forward Search [26]. It is briefly described as follows:

1. Suppose we have an initial dataset with n features $F_1, F_2..., F_n$
2. Sort the features according to their individual predictive ability to the class label:$F_1^{'}, F_2^{'}..., F_n^{'}$
3. Select the top K features from the sorted features as a target set:$F_1^{'}, F_2^{'}..., F_K^{'}$
4. Choose a subset of the target set with the highest predictive ability to the class.

In general, this feature selection method favors an individual feature that is highly correlated with the class but much less correlated with other features.

Feature Encoding

Some of the features can be interpreted as either nominal or ordinal variables. Thus, by using different encoding approaches, two types of datasets are created. They are denoted as Nominal-Variable-Encoding and Numeric-Variable-Encoding. In Numeric-Variable-Encoding, these variables are encoded as numeric variables to preserve the order information while in Nominal-Variable-Encoding we simply treat them as nominal variables. Note that for the second dataset we only used Numeric-Variable-Encoding while for the final dataset both encoding methods are applied.

Feature Discretization

Since many machine learning algorithms work better with discrete features, numeric features are sometimes discretized using supervised discretization based on Fayyad & Iranis Minimum Description Length (MDL) methods [25]. This method utilizes a heuristic algorithm to minimize entropy for discretizing a continuous variable into multiple intervals.

4.2.5 Data Visualization

Data visualization provides a visual representation of data so that one can get a deeper understanding of the data in a more intuitive way. However, since it is difficult or even impossible to visualize high dimensional data (over 3 dimensionalities), a data compression process that reduces data dimensionality is required. In this paper, we utilize the Principal Component Analysis (PCA) [33] to achieve the goal of data compression. PCA reduces dimensionality by projecting n-dimensional data onto a k-dimensional space $(n > k)$ which minimizes the projection error. In this thesis, k is set to 2. The compression is done by using the PCA analysis embedded in Weka [28] while the visualization is done using Python Orange [19].

Chapter 5

System Requirements and Evaluation

5.1 System Requirements

The system we are developing has the following requirements:

- The classification model should be interpretable. The users should be able to see the evidence supporting the recommendations made by the system. Rule-based algorithms are more desirable (i.e, ARIPPER).
- The system should provide multiple predictions with support evidence (e.g., supporting rules or guidelines) so the users can choose the most appropriate one under different considerations.
- The system should include a limited number of variables (preferably no greater than 30 as required in the system's specification).
- The system should use web-based technology to provide easy and ubiquitous access on devises equipped with a modern browser.

5.2 Evaluation

To measure the effectiveness of our CDSS, a control experiment in clinical practice will be required. Specifically, we will need two groups of patients, one receiving only clinicians' recommendations of rehabilitation program while the other taking recommendations from clinicians with the assistance of the CDSS. By comparing the outcome of both group (success rate of return-to-work in one round of assessment and program), we could determine if the CDSS is making any difference. However, such kind of evaluation is not feasible at current stage because we need to wait for enough new patients and the completion of their rehabilitation programs. This will take up 3 to 5 years in the future.

Therefore, at current stage we evaluate the system by measuring the classification capability of the model and compare it with the human experts' classification performance. Specifically, a dataset will be split into a training and test set. The model is created on the training set and is then tested on the test set. The classification capability of this model is represented by how the model classifies instances of patients in the test set. To compare with the human experts, we need to compute how well human experts classify patients as a baseline. Currently, the measurement for the baseline is limited due to the following reasons:

- For a past record of patient with positive outcomes, the associated program is considered to be the true label of this record.
- However, for a past record of patient with negative outcome, the true label of this record is unknown.
- A confusion matrix cannot be formed and measurements based on the matrix are not available.

The only option available for the baseline is the successful rate of return-to-work, which is defined as

$$Successful\ rate = \frac{number\ of\ records\ with\ positive\ outcome}{number\ of\ all\ records}.$$

Since we can determine the outcome of a program based on the classification result, the successful rate of return-to-work can be redefined as

$$Successful\ rate = \frac{number\ of\ correctly\ classified\ instances}{number\ of\ all\ instances}.$$

This is the same definition as classification accuracy and is used to measure the classification model for the purpose of comparison with the baseline.

As mentioned in Section 5.1, the system should provide multiple predictions. However, for each instance in the test set, there is only one associated label. Since the goal is to build a decision SUPPORT system, the system is not supposed to make the final decision for the clinician by choosing one program from multiple predictions. But without a final decision, we cannot measure the classification model by comparing the final prediction with the true label. To resolve this issue, we propose a measurement called potential classification accuracy (PT) for ARIPPER defined as follows:

$$PT = \frac{\sum_{i=1}^{N} C(\mathbf{P}_i, l_i)}{N}$$

where $\mathbf{P}_i$ stands for the set of all predictions made by the model for test instance i, l_i represents the true label of instance i and N is the total number of instances in the test set. Function $C(\mathbf{P}_i, l_i)$ is defined as

$$C(\mathbf{P}_i, l_i) = \begin{cases} 1 & \text{if } l_i \in \mathbf{P}_i, \\ 0 & \text{otherwise} \end{cases} \quad i = 1, 2, ..., N.$$

This means that if the true label of a test instance matches with one of the predictions, we count it as a successful classification. Therefore, PT represents the highest classification accuracy a model can possibly achieve if the system is asked to make a final decision itself. The higher PT is, the higher possibility the model includes the right recommendations in its multiple recommendations. We are assuming that the clinician is capable of making the right decision if the true appropriate recommendation is included in the predictions. We can compute the quality of each prediction and narrow down the recommendation pool by choosing only those rank at the top based on the quality value. Different quality measurements (e.g., HARC, SRHC, HWCS and SRHCS) can lead to different potential classification accuracy.

Although we can only use classification accuracy to compare with the baseline, for completion we do include other measurements including Sensitivity, Specificity and their geometric mean G-Mean. Table 5.1 shows the confusion matrix for a two-class problem.

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

Table 5.1: Confusion Matrix For a Two-class Classification Problem

- **Sensitivity**: Sensitivity is defined as $\frac{TP}{TP+FN}$. In medical diagnosis, sensitivity describes the proportion of actual positive examples that are correctly identified.
- **Specificity**: Specificity is defined as $\frac{TN}{TN+FP}$. In medical diagnosis, specificity measures the proportion of actual negative examples that are correctly identified.
- **Geometric Mean (G-mean)**: There is a trade-off between sensitivity and specificity. G-mean is a more harmonic measurement defined as

$$\sqrt{sensitivity * specifity}.$$

5.2.1 Human Baseline

The human baseline here is derived from the final dataset. Out of 14484 records, there are 8522 successful recommendations made by physicians (with or without missing data in the record). Therefore, the current human baseline is 58.8% (8522/14484). Sensitivity, specificity and G-Mean are not applicable when analyzing the whole dataset, but we can analyze them on the extracted 4876 records. The total number of the physician's success is 3970 out of 4876, which gives a classification accuracy of 81.4%. Table 5.2 shows the analysis on the physician's performance on the extracted data with sensitivity, specificity and G-Mean.

	Sensitivity	Specificity	G-Mean
Prog0	0.75	0.91	0.83
Prog3	0.75	0.99	0.86
Prog4	0.86	0.85	0.85
Prog5	0.89	0.99	0.94
Prog6	0.81	0.97	0.89
Overall	0.81	0.95	0.88

Table 5.2: Physicians' Performance On Extracted Records

Chapter 6

Experiment Design and Evaluation for Separated Models Approach

6.1 Experiment Design

There are three types of injuries in our dataset including Leg injury (LEFS), Back Injury (OMPQ) and Arm Injury (DASH). It is possible that the type of injury may influence the decision of the clinicians. Therefore, we separate the second dataset based on the injury type and build three classification models.

As shown in Chapter 3, these datasets suffer from severe class imbalance. Since some of the minority classes have so few examples, a train-test split only makes it more difficult for algorithms to learn anything useful. Therefore, for each dataset, we use all the data as training set and apply the following preprocessing on it. Independent test sets can be acquired in later stages of the research in the future.

At this stage of our research, we only applied the combination of SMOTE + Tomek Links to mitigate the class imbalance and remove the potential class overlaps. The reason that we only use SMOTE is that some of the minority classes have so few examples and SMOTE can at least generate synthetic data to generalize the data space and possibly provide more information for machine learning algorithms to learn. On the other hand, Random Oversampling only replicates the original data without introducing anything new and is therefore less useful.

6.1.1 Data preprocessing

For each injury dataset, we apply the same data preprocessing as follows. To mitigate the class imbalance, we sample on the minority classes. Instead of simply balancing the data into a uniform distribution, we use a progressive sampling approach to change the class distribution:

1. Choose one minority class and fix the rest.
2. Increase the size of the selected class by a certain percentage P.
3. Train a RIPPER classifier on the sampled dataset. If the true positive rate of the selected class increases significantly (By significant, we mean that the increase is greater than 2%), undo the sampling and repeat step b with a larger percentage P and step c. However, if the size of the sampled class is greater than that of the largest class in the dataset (exceptions will be explained in details) or the increase is less than 2%, stop the sampling process.
4. Choose P as the final sampling percentage.

After sampling, we use feature selection and discretization to further process the data. Tomek Links Cleaning method is then applied to clean up the data space. Data points from different classes that form a Tomek Links are considered as marginal or noisy points, and generally can be removed. Principal Component Analysis (PCA) was then used to visualize the dataset. The details of the cleaning process are stated as follows:

1. Extract each pair of classes.
2. Identify the Tomek Links between these two classes.
3. Visualize the data using the first two components in PCA.
4. Remove noise or borderline points based on visualization (not necessarily removing both points in the link). If such cleaning improves the overall performance of the model, merge the cleaned up classes back to the whole dataset. Otherwise, undo the cleaning.

5. Repeat step 1 to 4 until all possible pairs of classes is processed.

We remove data points in the Tomek Links under different conditions:

- Data points from one class invade the other (but not in the opposite direction). These points are considered as noise and should be removed.
- Class overlap happens at their borderline. Remove the overlap points from both classes.
- A large portion of both classes are overlapped with each other. Preserve the one with higher interest (i.e., lower misclassification cost) but remove the other.

After sampling and cleaning, we can train models on the processed dataset.

6.2 Results and Discussions

6.2.1 Analysis of the separated models approach

After the sampling process, each dataset has a different class distribution as shown in Figure 6.1, 6.2 and 6.3.

Then PCA is applied to visualize the sampled dataset as shown from Figure 6.4 to 6.6. Note that for the sake of readability, all the data visualization in this thesis is only performed on a 10% stratified subset of the original dataset.

We can see that there are class overlaps between almost each pair of classes. Tomek Links is applied to clean up the mess. Figure 6.7 to 6.9 visualize the datasets after cleaning that are ready for training purpose.

6.2.2 Discussion

For each injury dataset, 4 classifiers are trained and validated in 10-fold cross validations as shown in Table 6.1 to 6.12.

Although the performance of these algorithms is very promising in the 10-fold cross validation, independent test evaluations are still required to examine their true

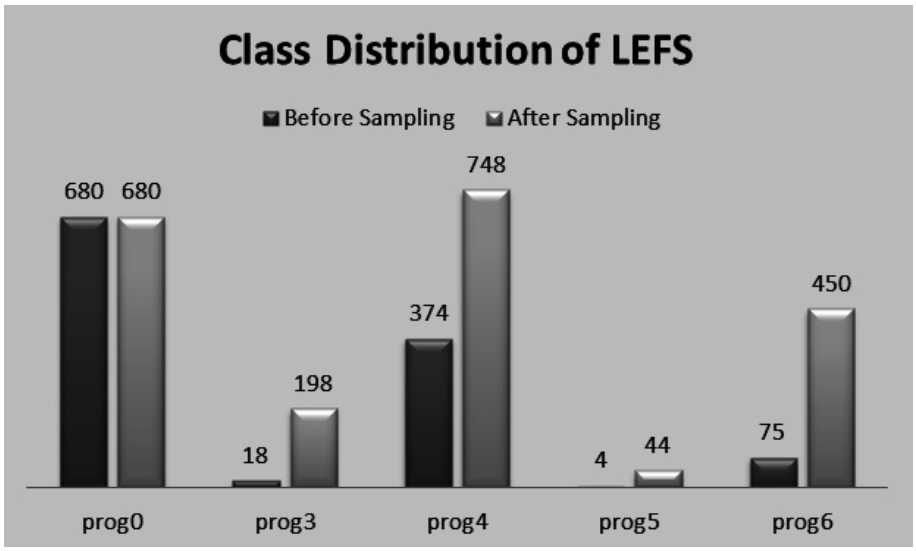

Figure 6.1: Class Distribution of LEFS Before and After Sampling

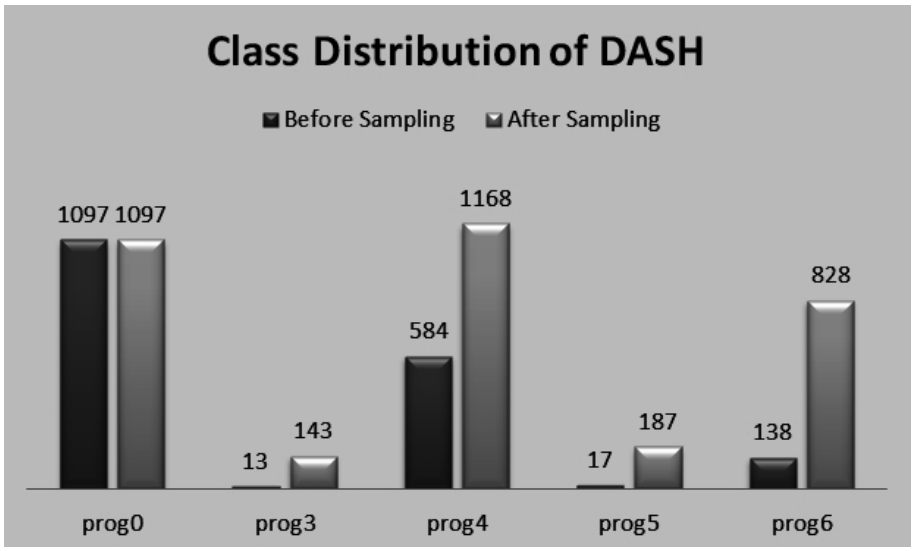

Figure 6.2: Class Distribution of DASH Before and After Sampling

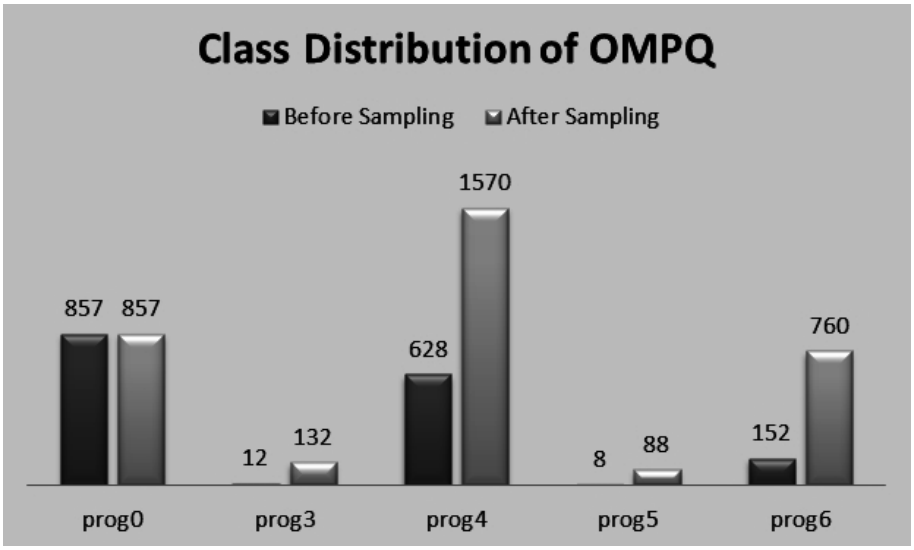

Figure 6.3: Class Distribution of OMPQ Before and After Sampling

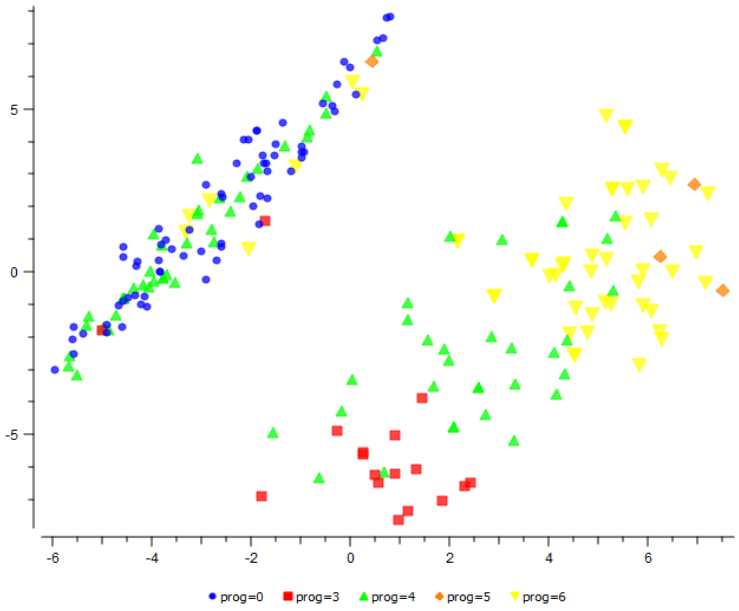

Figure 6.4: Visualization of LEFS after Sampling

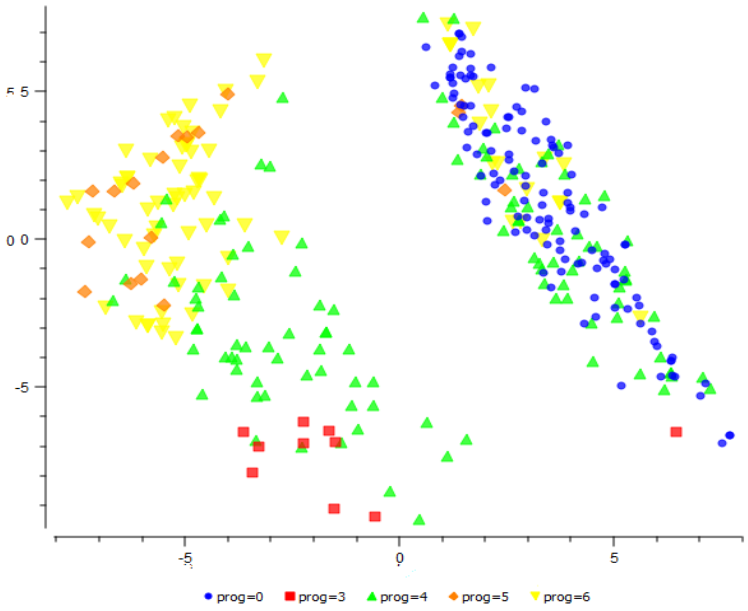

Figure 6.5: Visualization of DASH after Sampling

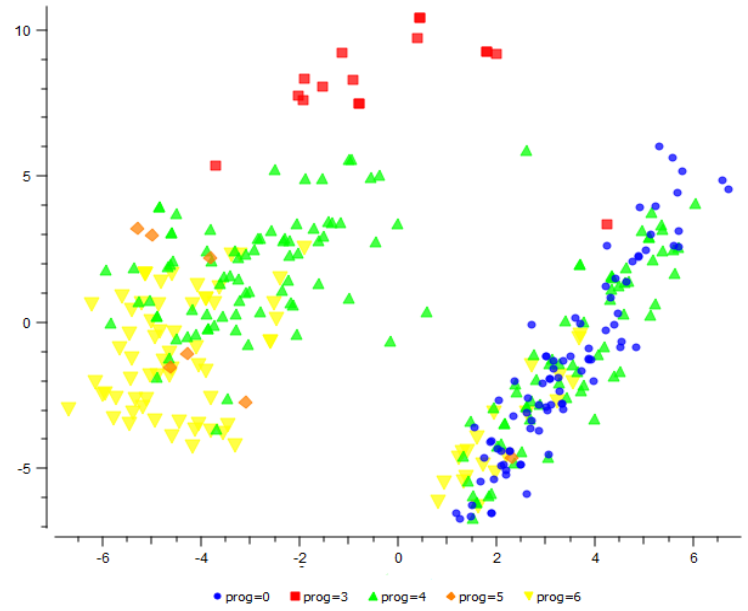

Figure 6.6: Visualization of OMPQ after Sampling

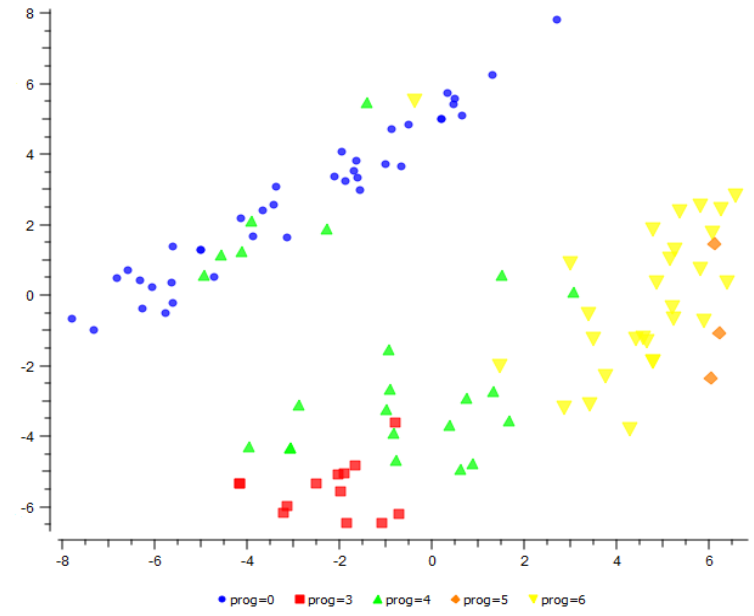

Figure 6.7: Visualization of LEFS after Tomek Links Cleaning

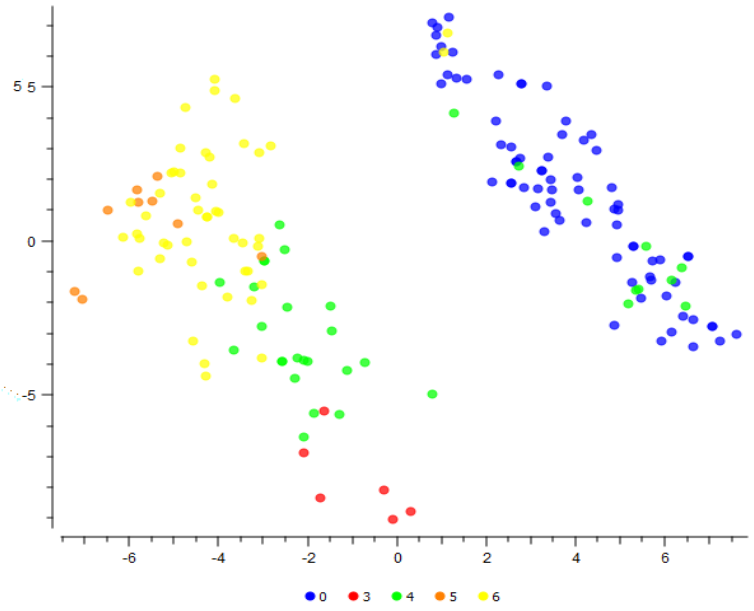

Figure 6.8: Visualization of DASH after Tomek Links Cleaning

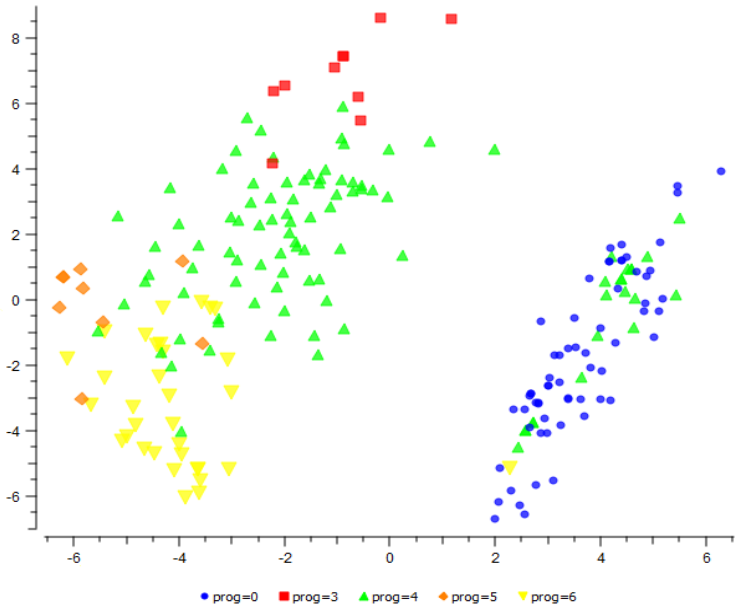

Figure 6.9: Visualization of OMPQ after Tomek Links Cleaning

classification ability. However, we were unable to get more minority class data for our research at that time. Independents test dataset will be provided in the future for additional evaluations.

	Sensitivity	Specificity	G-Mean
Prog0	0.943	0.867	0.904
Prog3	1.000	0.996	0.998
Prog4	0.681	0.963	0.810
Prog5	0.909	0.997	0.952
Prog6	0.911	0.978	0.944
Overall	0.863	0.966	0.913

Table 6.1: 10-Fold Cross Validation (RIPPER-LEFS)

	Sensitivity	Specificity	G-Mean
Prog0	1.000	0.867	0.931
Prog3	1.000	0.999	0.999
Prog4	0.635	0.995	0.795
Prog5	0.909	1.000	0.953
Prog6	0.972	0.968	0.970
Overall	0.884	0.971	0.926

Table 6.2: 10-Fold Cross Validation (Naive Bayes-LEFS)

	Sensitivity	Specificity	G-Mean
Prog0	0.818	0.858	0.838
Prog3	0.990	0.975	0.982
Prog4	0.642	0.922	0.769
Prog5	0.909	0.996	0.952
Prog6	0.887	0.969	0.927
Overall	0.802	0.951	0.873

Table 6.3: 10-Fold Cross Validation (C4.5-LEFS)

	Sensitivity	Specificity	G-Mean
Prog0	0.997	0.865	0.929
Prog3	0.917	1.000	0.957
Prog4	0.710	0.983	0.835
Prog5	0.886	1.000	0.941
Prog6	0.972	0.995	0.983
Overall	0.896	0.974	0.934

Table 6.4: 10-Fold Cross Validation (SVM-LEFS)

	Sensitivity	Specificity	G-Mean
Prog0	0.957	0.858	0.906
Prog3	0.937	0.992	0.964
Prog4	0.592	0.980	0.762
Prog5	0.802	0.988	0.890
Prog6	0.925	0.957	0.941
Overall	0.846	0.961	0.902

Table 6.5: 10-Fold Cross Validation (RIPPER-DASH)

	Sensitivity	Specificity	G-Mean
Prog0	1.000	0.869	0.932
Prog3	0.984	0.996	0.990
Prog4	0.661	0.988	0.808
Prog5	0.931	0.998	0.964
Prog6	0.949	0.990	0.970
Overall	0.896	0.974	0.935

Table 6.6: 10-Fold Cross Validation (Naive Bayes-DASH)

	Sensitivity	Specificity	G-Mean
Prog0	0.960	0.824	0.890
Prog3	0.730	0.993	0.852
Prog4	0.547	0.967	0.727
Prog5	0.733	0.987	0.851
Prog6	0.903	0.960	0.931
Overall	0.817	0.954	0.883

Table 6.7: 10-Fold Cross Validation (C4.5-DASH)

	Sensitivity	Specificity	G-Mean
Prog0	1.000	0.861	0.928
Prog3	0.429	1.000	0.655
Prog4	0.671	0.962	0.803
Prog5	0.509	1.000	0.713
Prog6	0.967	0.958	0.962
Overall	0.854	0.963	0.907

Table 6.8: 10-Fold Cross Validation (SVM-DASH)

	Sensitivity	Specificity	G-Mean
Prog0	0.827	0.839	0.833
Prog3	1.000	1.000	1.000
Prog4	0.723	0.892	0.803
Prog5	1.000	0.999	1.000
Prog6	0.904	0.976	0.939
Overall	0.810	0.952	0.878

Table 6.9: 10-Fold Cross Validation (RIPPER-OMPQ)

	Sensitivity	Specificity	G-Mean
Prog0	1.000	0.871	0.933
Prog3	0.991	0.999	0.995
Prog4	0.766	0.985	0.868
Prog5	1.000	1.000	1.000
Prog6	0.919	0.980	0.949
Overall	0.888	0.972	0.929

Table 6.10: 10-Fold Cross Validation (Naive Bayes-OMPQ)

	Sensitivity	Specificity	G-Mean
Prog0	0.924	0.840	0.881
Prog3	0.946	0.992	0.969
Prog4	0.721	0.933	0.820
Prog5	0.986	0.996	0.991
Prog6	0.828	0.980	0.901
Overall	0.827	0.957	0.889

Table 6.11: 10-Fold Cross Validation (C4.5-OMPQ)

	Sensitivity	Specificity	G-Mean
Prog0	0.999	0.871	0.933
Prog3	0.928	1.000	0.963
Prog4	0.807	0.943	0.872
Prog5	0.814	1.000	0.902
Prog6	0.819	0.998	0.904
Overall	0.880	0.970	0.924

Table 6.12: 10-Fold Cross Validation (SVM-OMPQ)

Chapter 7

Experiment Design and Evaluation for Single Model Approach

On the final dataset, we decided to build a single model instead of three. There are two reasons:

- Although the minority classes in the final dataset have more data instances, separating the dataset would result in the data insufficiency as mentioned in Chapter 5.
- Using a single model can simplify the prototype system since we only need to integrate one model instead of three.

7.1 Building A Single Model

To build a single classification model, we simply ignore the factor of injury by removing injury-related features since they are mutually exclusive and cause a large number of missing values if grouped together. In this approach, the dataset is split into training and test set because the minority classes in this dataset can afford such action. Since the data preprocessing methods are only validated on binary datasets and might be ineffective on the multi-class datasets, we divide our experiments into two parts: Direct Approach and Decomposition Approach.

7.1.1 Direct Approach

In this section, we present the direct approach. In short, we apply sampling and other data preprocessing methods directly on the multi-class dataset. Since we have two sampling methods, three cleaning methods and two types of datasets as well as whether to use discretization, 18 variations of the experiments are designed in the direct approach. Note that for nominal-encoding dataset there is no need to do discretization, which is the reason that we have 18 variations instead of 24. The following experiments share the same procedure as the experiments in Chapter 5. The only differences are the underlying methods and datasets. Table 7.1 lists all the experiments in the direct approach.

Two aspects that should be noted are:

- There is no need to apply discretization on the Nominal-Encoding dataset.
- The Tomek Links method in Chapter 5 is applied manually. However, it requires many steps and takes a long time. For all the experiments in Chapter 6, all the methods are applied automatically through computer programs.

We also include the AdaBoost.NC method in direct approach because it claims to be effective to handle multi-class imbalance without class decomposition. To use AdaBoost.NC, we first balance the dataset by sampling on the minority classes using Random Oversampling as suggested by the author. That is to say, after sampling, the dataset has a uniform distribution. We then apply AdaBoost.NC with different algorithms as base learners on the sampled data.

7.1.2 Decomposition Approach

Instead of applying any methods on the multi-class dataset directly, we first decompose the dataset into several binary datasets. The number of binary datasets varies depending on the decomposition strategy (either One-vs.-All or One-vs.-One).

Most of the binary datasets are still imbalanced and require sampling. We sample on the relative minority class in each binary dataset. The size of the minority class after sampling should be close to but smaller than that of the majority class.

Sampling	Dataset	Discretization	Cleaning
SMOTE	Numeric	Yes	ENN
SMOTE	Numeric	Yes	NCR
SMOTE	Numeric	Yes	TL
SMOTE	Numeric	No	ENN
SMOTE	Numeric	No	NCR
SMOTE	Numeric	No	TL
RO	Numeric	Yes	ENN
RO	Numeric	Yes	NCR
RO	Numeric	Yes	TL
RO	Numeric	No	ENN
RO	Numeric	No	NCR
RO	Numeric	No	TL
SMOTE	Nominal	No	ENN
SMOTE	Nominal	No	NCR
SMOTE	Nominal	No	TL
RO	Nominal	No	ENN
RO	Nominal	No	NCR
RO	Nominal	No	TL

Table 7.1: Experiments in Direct Approach

Each binary dataset can produce a binary classifier. To make the final decision, one needs to combine all the individual predictions. In the One-vs.-One approach, we use the voting strategy that the prediction receiving the most votes prevails. In the One-vs.-All approach, one usually accepts the prediction with the highest prediction probability. However, since the dataset is imbalanced, using such strategy would still make a biased prediction towards the majority class. Thus, we use the imbalance rate [55] to combine the probability prediction of all binary classifiers: we take the product of prediction probability and the imbalance rate of its corresponding class as a final weight. The test instance belongs to the class with the highest weight. The imbalance rate of a class is defined as the inverse of the proportion of instances in that class over the whole population in the dataset. We further divide the experiments into the following two categories:

- Class decomposition + SMOTE/RO + NCR/ENN/Tomek Links + Discretization/No Discretization + Numeric-Variable Encoding/ Nominal-Variable Encoding (OVA: One-vs-All): In this approach we first decompose the dataset into 5 binary datasets. Each binary dataset contains the data from one positive class and all other classes are considered as one negative class.

  We use SMOTE/Random Oversampling to sample on the minority class in each binary dataset. Then we apply NCR/ENN/Tomek Links as a data cleaning method to clean the data space. After the cleaning, five binary classifiers are created using different learning algorithms. To make a prediction for an unknown instance, each classifier generates a probability of that instance belonging to the positive class and we combine them with imbalance rate mentioned above.

- Class decomposition + SMOTE/RO + NCR/ENN/Tomek Links + Discretization/No Discretization + Numeric-Variable Encoding / Nominal-Variable Encoding (OVO: One-vs-One): this approach is almost the same as the OVA approach except for two parts: 1) We divide the training set into 10 binary datasets and 2) we use voting when combining the predictions from each binary classifiers.

7.2 Results and Discussions

7.2.1 Analysis of the single model approach

Direct Approach

Since all the experiments share the same procedure, we only provide the details of one experiment. The rest are summarized together in the analysis.

SMOTE + Tomek Links + Discretization + Numeric-Variable-Encoding: we first sampled on the minority class using SMOTE. The final sampling percentage obtained for each minority class is 900%, 900% and 300% respectively. Figure 7.1 shows the class distribution before and after the sampling.

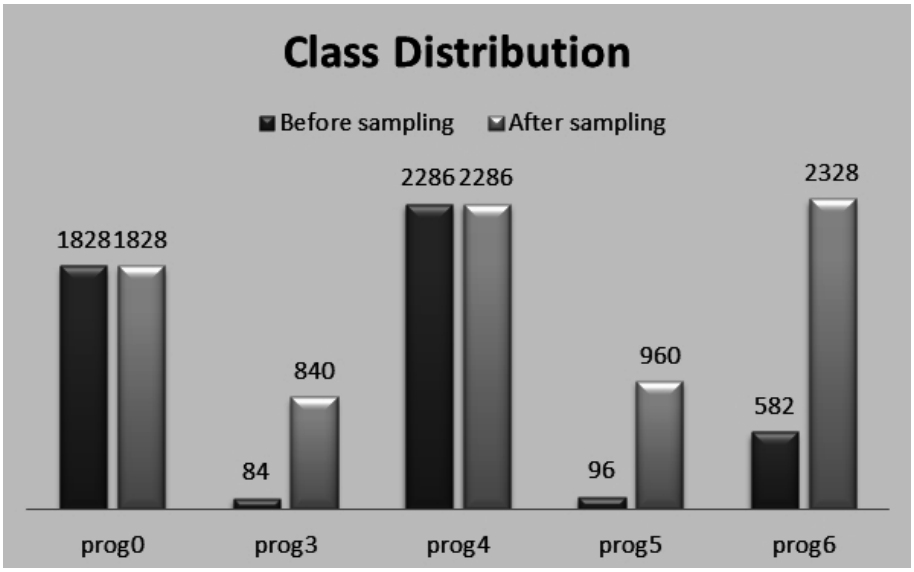

Figure 7.1: Class Distribution Before and After Sampling-Final Dataset

We can visualize the sampled dataset using PCA with the first two components as shown in Figure 7.2. We can see that on the left side prog 3 has a minor overlap with prog 6 while prog 5 have deeply invaded prog 6. On the right side, prog 0 and 4 are completely mixed together.

We apply Tomek Links Cleaning on the sampled dataset. Figure 7.3 visualizes the dataset after Tomek Links cleaning. Table 7.2 shows the 10-fold cross validation on the cleaned-up dataset using RIPPER algorithm.

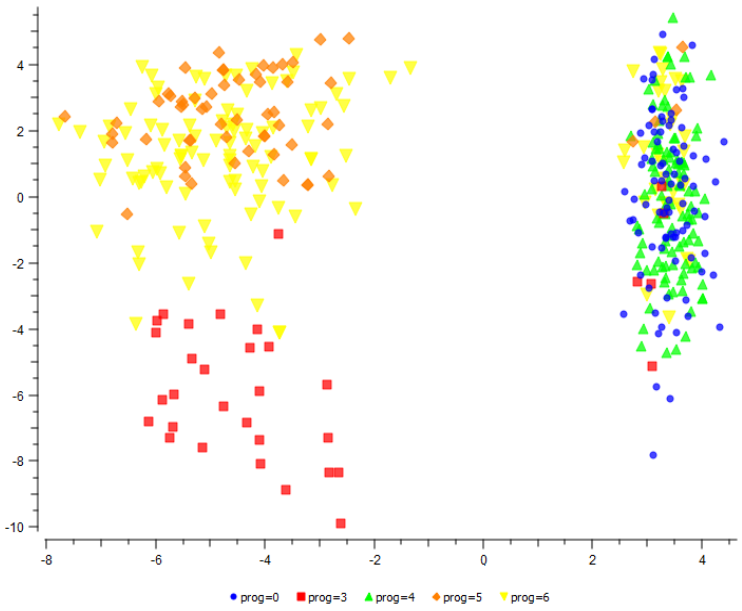

Figure 7.2: Dataset Visualization After Sampling

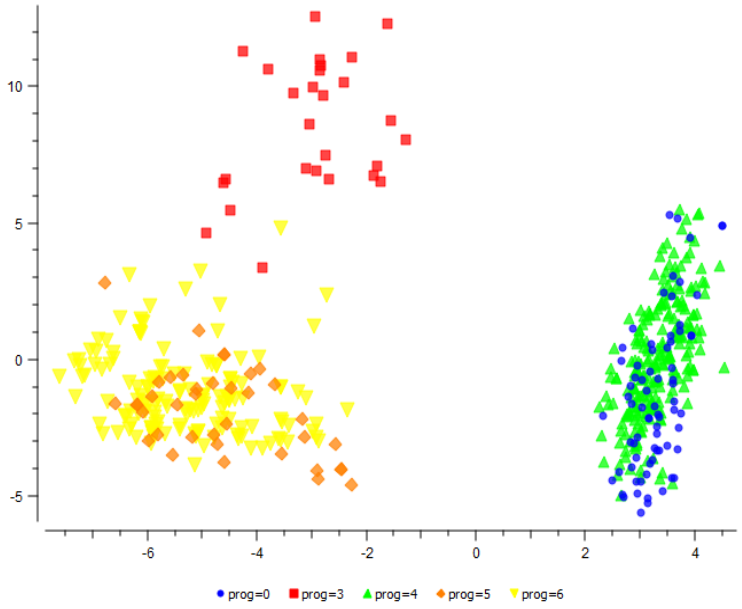

Figure 7.3: Dataset Visualization After Cleaning

	Sensitivity	Specificity	G-Mean
Prog0	0.259	0.964	0.500
Prog3	0.976	0.999	0.822
Prog4	0.929	0.822	0.874
Prog5	0.840	0.991	0.912
Prog6	0.963	0.988	0.975
Overall	0.845	0.961	0.901

Table 7.2: 10-Fold Cross Validation On the Cleaned-up Final Dataset (RIPPER)

The only drawback here is the low performance on class 0. There are two possible reasons to explain the low performance on prog0:

- Since prog0 and prog4 are mixed with each other all the time, it is possible the features we use cannot effectively separate them (currently we have 30 features).
- We can see from Figure 7.2 that a large portion of prog0 is removed and the rest of it (those useful examples) may not be sufficient for training model.

Thus, we sample on prog0 as a possible solution. Since the feature selection process is data dependent, it is possible that we can select new and effective features to separate prog0 and 4. Sampling on prog0 may cause further overlapping between prog0 and 4. But those points will be removed later as noise while the useful examples will be reinforced. We choose to sample 60% on prog0 and apply the same procedures above. 19 features are selected and the visualization using PCA is shown in Figure 7.4.

Clearly, we can see that part of prog0 is now separable from prog4. We then apply the Tomek Links cleaning on the new dataset. Figure 7.5 visualizes the dataset after the cleaning. Those points from prog0 mixing with prog4 are removed while those separable remain in the data space.

We then train a model using the RIPPER algorithm on the cleaned-up dataset. This model has a better performance on prog0 as shown in Table 7.3.

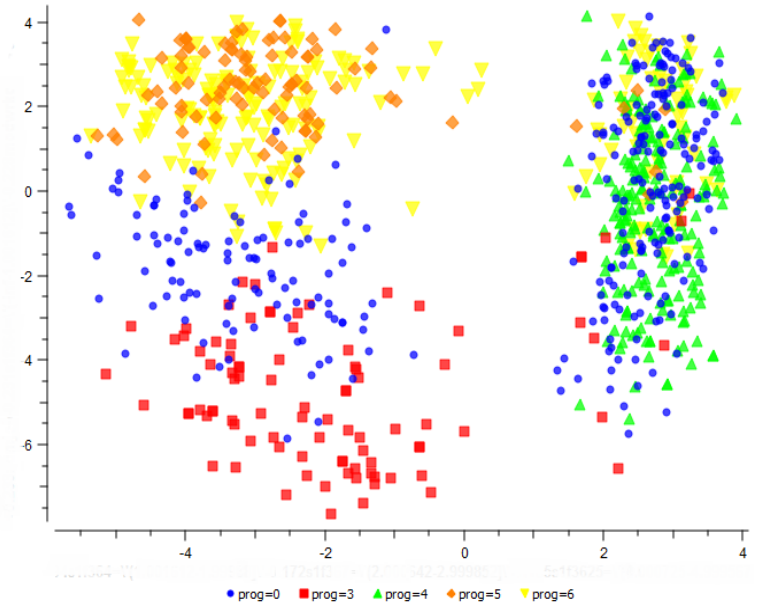

Figure 7.4: Dataset Visualization After Sampling (prog0 sampled)

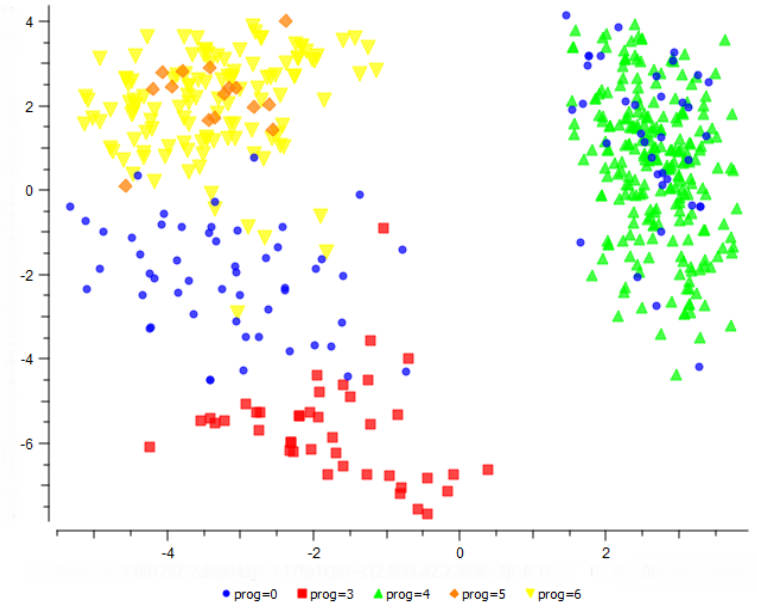

Figure 7.5: Dataset Visualization After Cleaning(prog0 sampled)

	Sensitivity	Specificity	G-Mean
Prog0	0.606	0.979	0.770
Prog3	0.973	0.994	0.983
Prog4	0.958	0.890	0.923
Prog5	0.804	0.979	0.887
Prog6	0.907	0.969	0.938
Overall	0.865	0.966	0.914

Table 7.3: 10-Fold Cross Validation On the Cleaned-up Data (Class 0 Sampled) Using RIPPER

Although the model works well in 10-fold cross validation as shown in Table 7.3, the evaluation is biased. An evaluation on the test set is required. Table 7.4 shows the evaluation on the test set using RIPPER algorithm. We also trained another two classifiers using Naive Bayes and C4.5 (SVM is not presented here since the training time is too long). Their evaluations on the test are shown in Table 7.5 and 7.6.

	Sensitivity	Specificity	G-Mean
Prog0	0.557	0.577	0.567
Prog3	0.000	0.996	0.000
Prog4	0.629	0.591	0.609
Prog5	0.000	1.000	0.000
Prog6	0.034	0.993	0.185
Overall	0.508	0.877	0.668

Table 7.4: Evaluation On the Test Set (RIPPER)

For other experiments in direct approach, we follow the same evaluation process. Each model is evaluated using 10-fold cross validation and on the test set. We summarize the evaluation on the test set for all the experiments using direct

	Sensitivity	Specificity	G-Mean
Prog0	0.071	0.961	0.261
Prog3	0.000	1.000	0.000
Prog4	0.969	0.069	0.260
Prog5	0.000	1.000	0.000
Prog6	0.000	1.000	0.000
Overall	0.482	0.870	0.647

Table 7.5: Evaluation On the Test Set (Naive Bayes)

	Sensitivity	Specificity	G-Mean
Prog0	0.426	0.695	0.544
Prog3	0.111	0.990	0.332
Prog4	0.664	0.498	0.575
Prog5	0.000	0.998	0.000
Prog6	0.155	0.956	0.385
Overall	0.492	0.873	0.655

Table 7.6: Evaluation On the Test Set (C4.5)

approach (except for AdaBoost.NC) in Table 7.7. Since it is not practical to detail the performance of each model in each experiment like our example experiment, we only report three measurements as shown in the last three columns in Table 7.7 with the form (N3/N5/Accuracy). N3 and N5 stands for the number of test instances from prog3 and 5 respectively that are correctly classified. There are 9 test instances in total in these two classes respectively. The accuracy here is the overall accuracy (in percent) on the whole test set.

Sampling	Dataset	Discretize	Cleaning	Bayes	C4.5	RIPPER
SMOTE	Numeric	Yes	ENN	0/0/48.4	0/0/49.4	0/0/48.6
SMOTE	Numeric	Yes	NCR	0/0/47.5	4/0/48	1/0/45.1
SMOTE	Numeric	Yes	TL	0/0/48.2	1/0/49.2	0/0/50
SMOTE	Numeric	No	ENN	8/2/32	1/0/32.2	3/2/32.4
SMOTE	Numeric	No	NCR	8/1/30.5	5/1/39.5	4/1/44.3
SMOTE	Numeric	No	TL	8/2/28.9	2/0/40.4	4/2/45.3
RO	Numeric	Yes	ENN	5/4/33.2	0/2/37.5	2/1/33.4
RO	Numeric	Yes	NCR	5/5/36.1	1/2/42.4	2/2/31.8
RO	Numeric	Yes	TL	5/4/36.3	0/1/42.2	3/1/39.3
RO	Numeric	No	ENN	7/6/31.8	3/1/36.7	2/0/34.2
RO	Numeric	No	NCR	9/6/33.2	4/1/45.7	5/1/46.9
RO	Numeric	No	TL	9/6/32.4	3/1/42	2/0/41.2
SMOTE	Nominal	No	ENN	5/2/43.6	3/1/45.5	2/0/42.2
SMOTE	Nominal	No	NCR	7/2/40.4	4/3/45.3	6/0/39.8
SMOTE	Nominal	No	TL	7/2/43.6	2/1/44.9	3/0/46.3
RO	Nominal	No	ENN	7/5/34.2	1/0/35.7	3/1/30.1
RO	Nominal	No	NCR	8/5/35.9	1/1/42.8	4/3/45.3
RO	Nominal	No	TL	8/5/34.6	1/0/40.4	1/2/38.9

Table 7.7: Evaluation On the Test Set (All Experiments)

AdaBoost.NC + Numeric-Variable-Encoding/Nominal-Variable-Encoding: We use naive Bayes and C4.5 as the base learner in this approach. Table 7.8 and 7.9 show the evaluation on the test set using base learner Naive Bayes. Table 7.10 and 7.11 show the evaluation on the test set using base learner C4.5. The overall classification accuracies for these four experiments are 32.4%, 28.6%, 37.9% and 37.5% respectively.

	Sensitivity	Specificity	G-Mean
Prog0	0.169	0.934	0.169
Prog3	1	0.856	1
Prog4	0.41	0.779	0.41
Prog5	0.444	0.724	0.444
Prog6	0.293	0.872	0.293
Overall	0.317	0.829	0.317

Table 7.8: Evaluation On the Test Set (Numeric-Variable-Encoding, Naive Bayes)

	Sensitivity	Specificity	G-Mean
Prog0	0.142	0.951	0.368
Prog3	0.889	0.889	0.889
Prog4	0.424	0.784	0.576
Prog5	0.667	0.674	0.670
Prog6	0.241	0.867	0.458
Overall	0.309	0.827	0.506

Table 7.9: Evaluation On the Test Set (Nomial-Encoding, Naive Bayes)

Decomposition Approach

For the decomposition approach, instead of displaying all the results, we only present those that are representative. Table 7.12 shows the results using OVA de-

	Sensitivity	Specificity	G-Mean
Prog0	0.317	0.833	0.317
Prog3	0.555	0.949	0.555
Prog4	0.393	0.815	0.393
Prog5	0	0.983	0
Prog6	0.793	0.633	0.793
Overall	0.408	0.852	0.408

Table 7.10: Evaluation On the Test Set (Numeric-Variable-Encoding, C4.5)

	Sensitivity	Specificity	G-Mean
Prog0	0.268	0.823	0.469
Prog3	0.667	0.908	0.778
Prog4	0.380	0.776	0.543
Prog5	0.222	0.931	0.455
Prog6	0.569	0.716	0.638
Overall	0.363	0.841	0.552

Table 7.11: Evaluation On the Test Set (Nominal-Variable-Encoding, C4.5)

composition while Table 7.13 shows the results using OVO decomposition. The results here follows the same format in Table 7.7. In some experiments the minority classes are completely misclassified and have low overall classification accuracy. We omit their results by filling the corresponding cells with "X".

Sampling	Dataset	Discretize	Cleaning	Bayes	RIPPER
SMOTE	Numeric	Yes	ENN	8/4/35.6	3/2/43.2
SMOTE	Numeric	Yes	NCR	9/4/28.3	3/3/37.7
SMOTE	Numeric	Yes	TL	8/4/34.6	2/2/37.1
SMOTE	Numeric	No	ENN	X	X
SMOTE	Numeric	No	NCR	X	X
SMOTE	Numeric	No	TL	X	X
RO	Numeric	Yes	ENN	9/6/23.9	3/0/47.3
RO	Numeric	Yes	NCR	9/6/17.2	4/0/13.5
RO	Numeric	Yes	TL	9/6/16.8	3/1/12.9
RO	Numeric	No	ENN	X	X
RO	Numeric	No	NCR	X	X
RO	Numeric	No	TL	X	X
SMOTE	Nominal	No	ENN	X	X
SMOTE	Nominal	No	NCR	X	X
SMOTE	Nominal	No	TL	X	X
RO	Nominal	No	ENN	9/6/9.8	1/0/39.5
RO	Nominal	No	NCR	9/6/6.5	2/2/31.1
RO	Nominal	No	TL	9/6/6.9	2/3/12.9

Table 7.12: Evaluation On the Test Set (OVA-All Experiments)

Sampling	Dataset	Discretize	Cleaning	Bayes	RIPPER
SMOTE	Numeric	Yes	ENN	9/4/40.6	4/1/47.9
SMOTE	Numeric	Yes	NCR	9/4/36.9	4/1/34.0
SMOTE	Numeric	Yes	TL	7/3/41.6	0/0/48.9
SMOTE	Numeric	No	ENN	X	X
SMOTE	Numeric	No	NCR	X	X
SMOTE	Numeric	No	TL	X	X
RO	Numeric	Yes	ENN	X	X
RO	Numeric	Yes	NCR	X	X
RO	Numeric	Yes	TL	X	1/1/43.2
RO	Numeric	No	ENN	X	X
RO	Numeric	No	NCR	X	X
RO	Numeric	No	TL	X	X
SMOTE	Nominal	No	ENN	X	X
SMOTE	Nominal	No	NCR	X	X
SMOTE	Nominal	No	TL	X	X
RO	Nominal	No	ENN	9/4/32.8	1/0/42.6
RO	Nominal	No	NCR	9/4/19.3	2/0/35.2
RO	Nominal	No	TL	9/4/35.2	0/0/45.8

Table 7.13: Evaluation On the Test Set (OVO-All Experiments)

7.2.1.1 Direct Approach For ARIPPER

Clearly, we can see from Table 7.7 to 7.13 that none of the experiments outperforms the human baseline on overall classification accuracy, regardless of whether or not to use class decomposition. However, in the direct approach, we are using the default setting RIPPER setting of firing the first rule that covers the test instance. To see how it works under the alternate setting (recommending multiple programs and pick the most appropriate one as the prediction), we did further evaluations only for ARIPPER. Table 7.14 shows the evaluation on the test set using ARIPPER."DS" is short for Discretize while PT is short for potential. PT means that if any of the predictions matches with the true label, we count it as a correct prediction. It is the maximum classification accuracy this model can achieve if given an appropriate prediction selection criteria. Some cells in Column WCS and SRHCS are filled with $'$-$'$ because the results are not applicable due to the zero division problem when calculating the Chi Square statistics.

Note that in rules generated from RIPPER algorithm, there is a default rule with empty rule body and neither confidence nor Chi Square is applicable. So for each test instance, we assign to it both the selected prediction and the default prediction. If either of them matches with the true label, we count it as a correct prediction.

7.2.2 Discussions

Based on the results shown above, we can see that none of the models (using default algorithms) outperforms the human baseline on the test set in the direct approach, although they all did well in the 10-fold cross validation. The class decomposition approach did not make any difference. However, the naive Bayes algorithm is working better than other algorithms in predicting minority class examples. The AdaBoost.NC method also shows its efficacy in classifying minority classes. But it is difficult to ensure good classification results on both the majority and minority classes at the same time. This is a tradeoff with the presence of class imbalance.

The nature of the rehabilitation program is also somewhat responsible for the misclassification between the majority classes (according to the confusion matrix,

Sampling	Dataset	DS	CL	HARC	SRHC	WCS	SRHCS	PT
SMOTE	Numeric	Yes	ENN	54.30	53.48	41.39	42.01	76.23
SMOTE	Numeric	Yes	NCR	75.20	71.72	61.27	62.29	87.29
SMOTE	Numeric	Yes	TL	72.95	72.13	48.15	47.74	78.27
SMOTE	Numeric	No	ENN	37.91	37.91	39.14	39.14	44.87
SMOTE	Numeric	No	NCR	53.28	53.28	52.66	52.66	54.92
SMOTE	Numeric	No	TL	56.35	56.35	56.56	56.56	56.96
RO	Numeric	Yes	ENN	60.45	60.65	–	–	72.34
RO	Numeric	Yes	NCR	59.84	58.61	–	–	68.03
RO	Numeric	Yes	TL	68.65	68.65	–	–	73.56
RO	Numeric	No	ENN	51.64	51.64	–	–	53.27
RO	Numeric	No	NCR	67.83	67.63	–	–	70.90
RO	Numeric	No	TL	53.89	53.89	–	–	54.92
SMOTE	Nominal	No	ENN	46.93	46.31	43.65	43.65	50.82
SMOTE	Nominal	No	NCR	45.90	45.69	46.52	46.52	50.82
SMOTE	Nominal	No	TL	60.24	60.24	59.43	59.43	61.27
RO	Nominal	No	ENN	46.93	36.88	–	–	38.52
RO	Nominal	No	NCR	63.24	63.24	–	–	63.93
RO	Nominal	No	TL	51.43	51.43	–	–	53.07

Table 7.14: Evaluation On the Test Set (ARIPPER)

not shown). As we are told by the experts from WCB and Department of Physical Therapy, prog0 and prog4 are similar to each other. A large portion of people receiving prog4 actually does not need prog4. But in order to make sure that people do return to work, they are assigned with prog4 in the end. Additionally, prog6 is a hybrid program of prog4 and prog5, which makes it even more complicated in classification. The data visualization in Chapter 5 and 6 also confirms this issue as we can see the overlap between these classes.

ARIPPER shows different performance on the test evaluation. As shown in Table 7.14, we have 9 out of 18 results with potential above the human baseline. These results indicate that we may still have a chance to apply certain extraction methods to select predictions and make the system more accurate than the baseline. The top performance is given by the combination of SMOTE + Numeric-Variable-Encoding + Discretization + NCR. It achieves 3 highest classification accuracies out of 4 measurements. The one in the second place is SMOTE + Numeric-Variable-Encoding + Discretization + TL. It achieves the highest accuracy by using SRHCS and second place using HARC. Additionally, we can randomly fix 3 columns out of the first 4 and vary the rest to find the best choice in that column. For example, by fixing column 2 to 4, we compare the SMOTE and the Random Oversampling method. Row 1 is compared to Row 7 because they have the same value from col 2 to 4 but sampling method varies. Three out of five measurements are comparable including HARC, SRHCS and Potential. The Random Oversampling method gets 2 wins while SMOTE gets only 1. In this way, we make all possible comparisons and accumulate the wins. The one with the most wins prevails. Table 7.15 shows the statistics of the wins for each column. We can see that the best choice for each column are SMOTE, Numeric-Variable-Encoding, Discretization and TL respectively. This is in fact the second best combination. So the combination of each column does not necessary produce the best combination, but it is close.

We can examine the prediction selection criteria from column 5 to 8 in a similar way. For each row, we compare the value from column 5 to 8 and get the column index with the maximum accuracy. Table 7.16 shows the statistics of the accumulated wins. We found that HARC is the best approach for selecting predictions.

SMOTE	15
Random Oversampling	11
Numeric-Variable-Encoding	14
Nominal-Variable-Encoding	1
Discretize	21
No discretize	3
Tomek Link	12
NCR	10
ENN	0

Table 7.15: Statistics For Column 1 to 4

HARC	SRHC	WCS	SRHCS
10	1	0	0

Table 7.16: Statistics For Column 5 to 8

The only problem with the evaluation here is that we still have two predictions for each test instance because of the empty rule body of the default rule. Currently we haven't found a proper way to overcome this problem with RIPPER. Since ARC-BC does not have the empty rule problem, we have trained an ARC-BC classifier on the dataset processed by the combination of SMOTE + Numeric-Variable-Encoding + Discretization + Tomek Links and test it on the test set. The result is shown in Table 7.17.

	Sensitivity	Specificity	G-Mean
Prog0	0.132	0.958	0.355
Prog3	0.588	0.458	0.519
Prog4	0.856	0.274	0.484
Prog5	0.105	0.969	0.319
Prog6	0.069	0.952	0.256
Overall	0.471	0.747	0.593

Table 7.17: Evaluation On the Test Set (Numeric-Variable-Encoding, ARC-BC)

ARC-BC algorithm is also not working well on this one test. Besides, the number of rules extracted by ARC-BC is much greater than that produced by RIPPER since RIPPER rules are more compact. Currently our prototype system implements the RIPPER rules trained from the combination of SMOTE + Numeric-Variable-Encoding + Discretization + Tomek Links since the rules are considered to be very meaningful from clinical perspective (more details in Section 7.3). As a decision SUPPORT system, a high "potential" should be sufficient since the clinician is the one who makes the final decision. To further evaluate the system, we need to do additional validations in real clinical settings.

7.3 Prototype System WATT

In this section, we introduce our prototype CDSS Work Assessment Triage Tool (WATT). It is a web application that is accessible through browsers (JavaScript enabled) on all platforms. The tool integrates both the positive and negative model generated by the RIPPER algorithm, specifically, generated from the combination of SMOTE + Numeric-Variable-Encoding + Discretization + Tomek Links + RIPPER but on different data. The positive model is built using all 4876 records (training and test sets are merged back together) while the negative model is built using records with unsuccessful RTW status. Table 7.18 shows the 10-fold cross validation of the RIPPER algorithm in the experiment that generates our rule set.

	Sensitivity	Specificity	G-Mean
Prog0	0.75/0.615	0.91/0.980	0.83/0.776
Prog3	0.75/0.936	0.99/0.992	0.86/0.964
Prog4	0.86/0.976	0.85/0.876	0.85/0.925
Prog5	0.89/0.757	0.99/0.993	0.94/0.867
Prog6	0.81/0.963	0.97/0.985	0.89/0.974
Overall	0.81/0.891	0.95/0.973	0.88/0.931

Table 7.18: 10-Fold Cross Validation Using RIPPER to Generate Rules Comparing to the Physicians' Performance on Extracted Records (Physicians/RIPPER)

Figure 7.6 shows a screen capture of the tool interface. The tool integrates 20 features including: admjob, modwrkcd, Prencode, curwork, Diag-Grp, ACTOADMC, Occupationa, VAS, the following SF36 items: (2, 4, 5, 7, 12, 14, 18, 21, 25) and an additional feature funAbility provided by the domain expert externally. The meaning of each feature is explained in the following list. All the decision support rules are presented in the Appendix I.

- admjob: job attachment and working status at RTW assessment (whether the worker had a job to return to)

- modwrkcd: availability of modified work
- Prencode: National Occupational Classification Code [13]
- curwork: currently working or not
- Diag-Grp: ICD9 diagnostic code which describes the nature of the injuries
- ACTOADMC: calendar days between injury to assessment admission
- Occupationa: the 'Occupation' item from the Pain Disability Index (PDI) [11]. PDI is a self-report questionnaire that measures the degree of different aspects of your life are disrupted due to chronic pain. 'Occupation' in PDI measures the rate of disability on a 0-10 scale of occupational activities.
- VAS: Pain Visual Analogue Scale (VAS) out of 10 [18]. VAS is a measurement of the pain severity level on a scale of 0-10.
- SF-36: This is a health outcome measure including vitality, physical functioning, bodily pain, general health perceptions, physical role functioning, emotional role functioning, social role functioning, and mental health [56].
- funAbility: Whether the worker demonstrate pre-accident functional abilities.

The tool works like a questionnaire. To use the tool to make the recommendations for a certain claimant, one simply selects a value in each of the drop-down menus or enters a value in the textbox according to the characteristics of that claimant and then press the Predict button. Recommendations made by the system will be presented under the form. Each recommendation has the following attributes:

- The name of the rehabilitation program recommended.
- The prediction of the program duration if the claimant took that program (this is done by linear regression).
- The average rule confidence of the underlying rules supporting a recommendation.

Figure 7.6: WATT Interface

- The number of rules supporting a recommendation.
- The user can also view the details of the underlying rules by pressing the Rules button.

These features provide a way to measure the quality of the recommendations so the users can make a decision with more confidence.

7.3.1 Program Duration Prediction

The duration of a rehabilitation program is an important factor to be considered when making recommendations. Therefore, we trained a linear regression model with 20 features including the 19 features extracted in earlier section as well as the predicted program label. This linear regression model predicts the duration of the rehabilitation program based on the clinical characteristics of the patient and the program recommended.

7.3.2 Resolving Rule Conflicts

Since both the positive and negative models are integrated into the system, it is possible that a given instance is covered by both positive and negative rules making contradictory recommendations. There may not always be negative recommendations, but when there are, they may be in conflict with the positive rules. For example, Figure 7.7 shows a situation where conflict happens for a patient. The positive model makes three recommendations while the negative model makes two. The rehabilitation program Complex is in the middle of this conflict. To resolve any possible conflicts, users can compare different metrics of the recommendations or view the specific rules to decide to accept or reject a recommendation. Currently our domain expert suggests the user cancelling out the predictions involved in conflicts.

7.3.3 Discussion

We have presented our prototype system at a meeting with rehabilitation experts from WCB and discussed the classification rules with a broader team on March 6th, 2012. During the demo of our system, several patient records are presented as test cases. The experts were invited to make a recommendation for these patients. It turned out that they did not agree with each other. However, once we showed the recommendations with supporting evidence made by our system, they all converged and reached an agreement on one recommendation as proposed by our system. This demo has illustrated the potential of our prototype system in facilitating the rehabilitation recommendation process. The classification rules were also evaluated by the experts and were considered as meaningful in clinical practice. We plan to do additional evaluation by measuring each rule on a scale to quantify how meaningful a rule is in the future.

Prediction from positive rules	Duration	Confidence	Rules Number	Rules
Consider vocational rehab/no further rehab	0 days	0.96	1	Rules
Complex (Chronic Pain Management Program) • Rule1: (admjob = 0) and (1<=s1f3612<=2) • Rule2: (admjob = 0) and (1<=s1f3967<=2) • Rule3: (0<=s1f3967<=1) and (1<=s1f3614<=2)	22 days	0.86	3	Rules
Provider-based (Functional Restoration Program)	24 days	0.84	1	Rules

Prediction from negative rules	Duration Confidence		Rules Number	Rules
Not Hybrid (Functional Restoration Program with Integrated Workplace Component)	---	1	1	Rules
Not Complex (Chronic Pain Management Program) • Rule1: (s1f3625>=2) and (s1f3625<=3) and (Diag_Grp = 3) • Rule2: (s1f3967>=1) and (s1f3967<=2) and (Diag_Grp = 3)	---	1	2	Rules

Figure 7.7: Conflict Case

Chapter 8

Conclusion

8.1 Conclusions

In this dissertation we designed and implemented a prototype of our decision support system, as well as tackled the multiple-class imbalance problem encountered in building the system′s knowledge base using various data preprocessing and machine learning algorithms. In summary:

1. In Chapter 2 we introduced the background of the rehabilitation treatment for injured workers in our research. We then described the concept of the decision support system (DSS) and its clinical application (CDSS). We further elaborated the differences between different CDSS categories and then moved on to the machine learning concept that is fundamental for building the knowledge base of the CDSS.

2. We explained the timeline, datasets and evaluation metrics of our research in Chapter 3. We briefly analyzed the statistics of each dataset we had and explained the multi-class imbalance problem and the limitations of the evaluation. Since the meaningful evaluation for the human baseline is currently limited to classification accuracy, we are currently stuck with it when doing comparison.

3. Chapter 4 states the system requirements for our prototype system and intro-

duces a variety of machine learning algorithms and data preprocessing methods. Each algorithm is briefly introduced through a concrete example and modifications to any algorithm are also specified. We also include the introduction of the feature selection, discretization and data visualization methods in this chapter.

4. In Chapter 5, we explained the separated model approach on the second dataset. Due to the lack of data in the minority classes, we have to use all the data in the training process instead of following the standard train-test split process. We applied the combination of SMOTE + Tomek Links on each injury dataset and then trained different classifiers using RIPPER, C4.5 SVM and naive Bayes algorithms. The classifiers are evaluated through 10-fold cross validation and have shown promising classification performance. However, without independent test evaluations, it is impossible to tell the general classification ability on unseen examples. Therefore we must wait for additional data to test the generalization ability of the separated models in the future.

5. Finally, in Chapter 6 we addressed our thesis statements. On the final dataset, we aimed at building a single classification model regardless of the injury type. All the following experiments are done on a 90%-10% train-test split of the dataset. In the direct approach, we apply 18 possible variations of data preprocessing combinations on the training set. All algorithms are then applied on each of these processed datasets. The corresponding classification models are evaluated through 10-fold cross validation and tested on the test set. Although the cross-validation looks promising, their performance on the test set is unfortunately no match to the human baseline on classification accuracy. However, the rules extracted from the combination of SMOTE + Numeric-Variable-Encoding + Discretization + Tomek Links + RIPPER are considered as meaningful rules and are now integrated into the prototype system for further evaluation on clinical samples. Additionally, we redo all the above experiments with class decomposition since the data cleaning meth-

ods are only validated on binary datasets. Unfortunately, The classification accuracy of all models on the test set is still lower than the human baseline. Therefore, class decomposition does not improve the overall classification models. It is worth noticing that naive Bayes algorithm makes good predictions of the minority classes with or without class decomposition. ARIPPER is then tested in another set of experiments. By using HARC and SHRCS measurement, the ARIPPER model can actually achieve over 71% accuracy with the potential of 87% with preprocessing combination of SMOTE + Numeric-Variable-Encoding + Discretization + NCR.

8.2 Summary of Contributions

This MSc dissertation makes the following contributions:

1. By using machine learning algorithms to extract decision rules, we combine the advantage of both the knowledge-based DSS and non-knowledge-based DSS together. The rule-based machine learning algorithm is able to extract editable and interpretable rules and may also mine facts that are not discovered by domain experts.

2. We have compared and analyzed a variety of methods and algorithms to tackle the multi-class imbalance problem encountered during the knowledge extraction process. We have found that models generated from the combination of SMOTE + Numeric-Variable-Encoding + Discretization + Tomek Links/NCR with ARIPPER can produce higher classification accuracy than the human baseline on the test set. The one with Tomek Links also produces meaningful recommendation rules as evaluated by our domain expert.

3. Our prototype decision support system Work Assessment Triage Tool (WATT), which integrates our rule-based model, provides a recommendation pool (both positive and negative recommendations) to facilitate the clinical process.

8.3 Future Study

In this dissertation we have compared and analyzed a variety of methods and algorithms to tackle multi-class imbalance problem in classification. There are certain limitations and much could be done to extend the proposed work.

1. Due to the class imbalance, it is difficult for machine learning algorithms to learn models that make good predictions on both the minority and majority classes at the same time. The tradeoff is inevitable. We hoped that we could get more data for the minority classes in later stage of this research. Moreover, an analysis of the misclassification cost between different classes would be beneficial as well.

2. The criterion for a successful treatment can vary and in result the datasets for positive and negative modeling can be different. Selecting an appropriate criterion for this issue is worth investigating. We would also like to know how the case managers make decisions since they have the power to override the clinicians′.

3. The major misclassification happened between prog0/4 and prog5/6. The rehabilitation programs represented by the class label in each pair are similar to each other in nature. A deeper analysis of how the human experts differentiate them would provide insight for developing new strategies in tackling the classification problem.

4. For the separated models based on injury type, additional independent tests are required in the future if the data become available. We would apply all the 18 variations of experiments presented in single model approach to each injury dataset.

5. In the single model approach with ARIPPER, three combinations have relatively high potential in classification accuracy. Currently, we are using only the confidence-based and chi-square-based criteria to select the appropriate predictions, and there is still room for improving the models to reach their

own potential by using other criteria. Additionally, the final decision consists of two predictions including a default prediction of the largest class. To narrow it down to only one prediction, we could:

(a) Train a set of binary classifiers. Based on the final two predictions, the corresponding binary classifier is used to classify the instance.

(b) Use other rule-based algorithms that do not generate default rule with empty rule body. And we could try more experiments with the ARC-BC algorithm.

6. Currently, we are focusing on the positive model. In fact, it is also possible to integrate the negative model in the evaluation. Based on certain criteria, we can use negative predictions to cancel out some positive predictions and narrow down the decision pool.

Bibliography

[1] Two modifications of cnn. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-6(11):769 –772, nov. 1976.

[2] *CMAR: accurate and efficient classification based on multiple class-association rules*, 2001.

[3] Peter L. Bartlett and Yoav Freund. Adaboost is consistent. In *In Advances in Neural Information Processing Systems*, 2006.

[4] A. Basnet. *Formulation of Decision Support Systems (DSS) Architectures for Highway Toll Plazas: A Thesis in Computer Science*. University of Massachusetts Dartmouth, 2006.

[5] Gustavo E. A. P. A. Batista, Ana L. C. Bazzan, and Maria Carolina Monard. Balancing training data for automated annotation of keywords: a case study, 2003.

[6] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[7] E.S. Berner. *Clinical Decision Support Systems: Theory And Practice*. Health Informatics. Springer, 2007.

[8] Martin BI, Deyo RA, and et al Mirza SK. Expenditures and health status among adults with back and neck problems. *JAMA*, 299:656–64, 2008.

[9] Martin BI, Deyo RA, Mirza SK, and et al. Expenditures and health status among adults with back and neck problems. *JAMA: The Journal of the American Medical Association*, 299(6):656–664, 2008.

[10] C Craig Blackmore, Robert S Mecklenburg, and Gary S Kaplan. Effectiveness of clinical decision support in controlling inappropriate imaging. *J Am Coll Radiol*, 8(1):19–25, 2011.

[11] Pollard CA. Preliminary validity study of the pain disability index. *Percept Mot Skills*, 59, 1984.

[12] Health Canada. Economic burden of illness in canada, 1998. 2002.

[13] Human Resources Development Canada. National occupational classification: Occupational descriptions. 2001.

[14] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[15] Nitesh V. Chawla, Ar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: improving prediction of the minority class in boosting. In *In Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*, pages 107–119, 2003.

[16] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[18] NICOLA CRICHTON. Information point: Visual analogue scale (vas). *Journal of Clinical Nursing*, 10:697–706, 2001.

[19] Toma Curk, Janez Demar, Qikai Xu, Gregor Leban, Uro Petrovi, Ivan Bratko, Gad Shaulsky, and Bla Zupan. Microarray data mining with visual programming. *Bioinformatics*, 21:396–398, February 2005.

[20] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[21] Gross DP, Zhang J, Steenstra I, Cooper J, Barnsley S, Haws C, McIntosh G, Amell T, and Zaiane O. Development of a computer-based decision-support tool using machine-learning strategies for selecting appropriate rehabilitation interventions. *Odense International Forum XII Primary Care Research on Back Pain*, 2012.

[22] Gross DP, Zhang J, Steenstra I, Cooper J, Barnsley S, Haws C, McIntosh G, Amell T, and Zaiane O. Development of a computer-based decision-support tool using machine-learning strategies for selecting appropriate rehabilitation interventions. *BMJ Occupational and Environmental Medicine*, 2012.

[23] Gross DP, Zhang J, Steenstra I, Cooper J, Barnsley S, Haws C, McIntosh G, Amell T, and Zaiane O. Development of a computer-based decision-support tool using machine-learning strategies for selecting appropriate rehabilitation interventions. 2012.

[24] Haldorsen EM. The right treatment to the right patient at the right time. *Occup Environ Med 2003*, 60:235–236, 2003.

[25] Fayyad and Irani. Multi-Interval discretization of Continuous-Valued attributes for classification learning. pages 1022–1027, 1993.

[26] Martin Gutlein, Eibe Frank, Mark Hall, and Andreas Karwath. Large-scale attribute selection using wrappers. In *CIDM*, pages 332–339, 2009.

[27] N.M. Hadler. *Occupational musculoskeletal disorders*. Raven Press, 1993.

[28] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[29] Mark A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, April 1999.

[30] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-SMOTE: A new Over-Sampling method in imbalanced data sets learning. pages 878–887. 2005.

[31] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[32] Brian Hemens, Anne Holbrook, Marita Tonkin, Jean Mackay, Lorraine Weise-Kelly, Tamara Navarro, Nancy Wilczynski, R Brian Haynes, and the CCDSS Systematic Review Team. Computerized clinical decision support systems for drug prescribing and management: A decision-maker-researcher partnership systematic review. *Implementation Science*, 6(1):89, 2011.

[33] Gordana Ivosev, Lyle Burton, and Ron Bonner. Dimensionality reduction and visualization in principal component analysis. *Analytical Chemistry*, 80(13):4933–4944, 2008.

[34] Rapoport J, Jacobs P, Bell NR, and Klarenbach S. Refining the measurement of the economic burden of chronic diseases in canada. *Chronic Dis Can*, 25:13–21, 2004.

[35] Nathalie Japkowicz. Concept-learning in the presence of between-class and within-class imbalances. In *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '01, pages 67–77, London, UK, UK, 2001. Springer-Verlag.

[36] Power JD, Perruccio AV, Desmeules M, Lagace C, and Badley EM. Ambulatory physician care for musculoskeletal disorders in canada. *J Rheumatol*, 33:133–9, 2006.

[37] K. Karjalainen, A. Malmivaara, M. Van Tulder, R. Roine, M. Jauhiainen, H. Hurri, and B. Koes. Multidisciplinary biopsychosocial rehabilitation for subacute low back pain among working age adults. *Cochrane Database Syst Rev*, 2, 2003.

[38] S. B. Kotsiantis and P. E. Pintelas. Mixture of expert agents for handling imbalanced data sets, 2003.

[39] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, AIME '01, pages 63–66, London, UK, UK, 2001. Springer-Verlag.

[40] Lin Lin, Paul Jen-Hwa Hu, and Olivia R. Liu Sheng. A decision support system for lower back pain diagnosis: Uncertainty management and clinical evaluations. *Decision Support Systems*, 42(2):1152 – 1169, 2006.

[41] Charles X. Ling and Chenghui Li. Data mining for direct marketing: Problems and solutions. In *KDD*, pages 73–79, 1998.

[42] Hadler NM. Occupational musculoskeletal disorders. 3rd ed. *Philadelphia, PA: Lippincott Williams and Wilkins*, 2005.

[43] Coyte PC, Asche CV, Croxford R, and Chan B. The economic cost of musculoskeletal disorders in canada. *Arthritis Care Res*, 11:315–25, 1998.

[44] Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Maria C. Monard. Class imbalances versus class overlapping: An analysis of a learning system behavior, 2004.

[45] J. Quinlan and R. Cameron-Jones. Foil: A midterm report. In *Machine Learning: ECML-93*, pages 1–20. Springer, 1993.

[46] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[47] J.R. Quinlan. Mdl and categorical theories (continued). In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 464–470. Citeseer, 1995.

[48] Logan R and Reeder P. Occupational injuries and diseases in canada, 19962005. *Ottawa: Human Resources and Social Development Canada*, 2007.

[49] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*.

[50] Pavel Roshanov, Shikha Misra, Hertzel Gerstein, Amit Garg, Rolf Sebaldt, Jean Mackay, Lorraine Weise-Kelly, Tamara Navarro, Nancy Wilczynski, R Brian Haynes, and the CCDSS Systematic Review Team. Computerized clinical decision support systems for chronic disease management: A decision-maker-researcher partnership systematic review. *Implementation Science*, 6(1):92, 2011.

[51] Navdeep Sahota, Rob Lloyd, Anita Ramakrishna, Jean Mackay, Jeanette Prorok, Lorraine Weise-Kelly, Tamara Navarro, Nancy Wilczynski, R Brian Haynes, and the CCDSS Systematic Review Team. Computerized clinical decision support systems for acute care management: A decision-maker-researcher partnership systematic review of effects on process of care and patient outcomes. *Implementation Science*, 6(1):91, 2011.

[52] Frederieke Schaafsma, Eva Schonstein, Karyn M Whelan, Eirik Ulvestad, Dianna Theadora Kenny, and Jos H Verbeek. Physical conditioning programs for improving work outcomes in workers with back pain. *Cochrane Database Syst Rev*, (1):CD001822, 2010.

[53] S.H. van Oostrom, M.T. Driessen, HC De Vet, R.L. Franche, E. Schonstein, P. Loisel, W. van Mechelen, and J.R. Anema. Workplace interventions for preventing work disability. *Cochrane Database Syst Rev*, 2, 2009.

[54] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, 2000.

[55] Shuo Wang and Xin Yao. The effectiveness of a new negative correlation learning algorithm for classification ensembles. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 1013–1020, Washington, DC, USA, 2010. IEEE Computer Society.

[56] Gandek B Ware JE. The sf-36 health survey: development and use in mental health research at the iqloa project. *International Journal of Mental Health*, 23, 1994.

[57] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, 2(3):408–421, July 1972.

[58] Osmar R. Zaïane and Maria-Luiza Antonie. Classifying text documents by associating terms with text categories. In *Proceedings of the 13th Australasian database conference - Volume 5*, ADC '02, pages 215–222, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[59] Jing Zhang, Douglas Gross, and Osmar Zaiane. On the application of multi-class classification in physical therapy recommendation (submitted). *IEEE International Conference on Data Mining*, 2012.

Appendix A

Final Rules in WATT

A.1 Positive Rules

1. (funAbility=1) ⇒ Consider immediate return
2. (admjob=1) and (2≤s1f364≤3) and (s1f367≥3) and (1≤VASa≤2) and (Prencode=7) ⇒ Worksite-Based Program
3. (admjob=1) and (4≤s1f3614≤5) and (s1f367≥3) and (s1f3612≥3) ⇒ Worksite-Based Program
4. (admjob=1) and (2≤s1f364≤3) and (s1f367≥3) and (Prencode=7) and (2≤s1f362≤3) and (2≤s1f365≤3) ⇒ Worksite-Based Program
5. (admjob=1) and (3≤s1f3614≤4) and (s1f364=2) and (2≤s1f362≤3) ⇒ Worksite-Based Program
6. (admjob=1) and (3≤s1f3621≤4) and (s1f364≥3) ⇒ Worksite-Based Program
7. (admjob=1) and (4≤s1f3614≤5) and (s1f364=2) ⇒ Worksite-Based Program
8. (admjob=1) and (2≤s1f364≤3) and (s1f3621=4) ⇒ Worksite-Based Program

9. (admjob=1) and (134≤ACTOADMC≤197) and (2≤s1f3612≤3) and (s1f364=2) and (s1f367≥3) ⇒ Worksite-Based Program

10. (admjob=1) and (2≤s1f362≤3) and (s1f364≥3) ⇒ Worksite-Based Program

11. (admjob=1) and (3≤s1f3614≤4) and (1≤s1f3612≤2) and (s1f367≥3) ⇒ Worksite-Based Program

12. (admjob=1) and (3≤s1f3621≤4) and (s1f3614=4) ⇒ Worksite-Based Program

13. (admjob=1) and (2≤s1f364≤3) and (50≤ACTOADMC≤90) and (s1f367≥3) and (3≤s1f3618≤4) ⇒ Worksite-Based Program

14. (admjob=1) and (2≤s1f364≤3) and (s1f3614≥5) ⇒ Worksite-Based Program

15. (admjob=1) and (7≤OCCUPATIONa≤8) and (s1f364=2) and (modwrkcd=0) ⇒ Worksite-Based Program

16. (admjob=1) and (4≤s1f3614≤5) and (modwrkcd=1) and (3≤OCCUPATIONa≤4) ⇒ Worksite-Based Program

17. (admjob=1) and (2≤s1f364≤3) and (s1f3618=3) ⇒ Worksite-Based Program

18. (admjob=1) and (3≤s1f3614≤4) and (s1f3621=4) and (3≤s1f362≤4) ⇒ Worksite-Based Program

19. (admjob=1) and (4≤s1f3618≤5) and (s1f364≥3) ⇒ Worksite-Based Program

20. (admjob=1) and (2≤s1f3612≤3) and (s1f3614≥5) ⇒ Worksite-Based Program

21. (admjob=1) and (4≤s1f3614≤5) and (2≤s1f364≤3) and (4≤s1f3618≤5) ⇒ Worksite-Based Program

22. (admjob=1) and (4≤s1f3614≤5) and (1≤s1f3625≤2) ⇒ Worksite-Based Program

23. (admjob=1) and (3≤s1f3618≤4) and (50≤ACTOADMC≤90) and (s1f3621=4) ⇒ Worksite-Based Program

24. (admjob=1) and (3≤s1f3614≤4) and (s1f367≥3) and (3≤s1f362≤4) and (modwrkcd=1) ⇒ Worksite-Based Program

25. (admjob=1) and (VASa=1) and (4≤s1f3625≤5) ⇒ Worksite-Based Program

26. (admjob=1) and (0≤ACTOADMC≤180) and (curwork=1) and (modwrkcd=0) and (2≤s1f365≤5) ⇒ Complex

27. (admjob=0) and (0≤s1f365≤1) and (7≤VASa≤8) ⇒ Complex

28. (admjob=0) and (1≤s1f3612≤2) ⇒ Complex

29. (admjob=0) and (660≤ACTOADMC≤1375) and (3≤s1f362≤4) ⇒ Complex

30. (admjob=0) and (4≤s1f362≤5) ⇒ Complex

31. (admjob=0) and (1≤s1f367≤2) ⇒ Complex

32. (admjob=0) and (4≤s1f3625≤5) and (Prencode=6) and (s1f3612=2) ⇒ Complex

33. (admjob=0) and (2≤s1f3618≤3) and (2≤s1f367≤3) and (1≤s1f364≤2) ⇒ Complex

34. (0≤s1f367≤1) and (1≤s1f3614≤2) ⇒ Hybrid

35. (admjob=1) and (2≤s1f365≤3) and (modwrkcd=1) ⇒ Hybrid

36. (admjob=1) and (2≤s1f367≤3) and (modwrkcd=1) ⇒ Hybrid

37. (admjob=1) and (2≤s1f3614≤3) and (modwrkcd=1) ⇒ Hybrid

38. (admjob=1) and (modwrkcd=2) and (Prencode=7) and (2≤s1f3614≤3) and (197≤ACTOADMC≤404) ⇒ Hybrid

39. (admjob=1) and (3≤s1f3621≤4) and (s1f3612≥3) ⇒ Hybrid

40. (admjob=1) and (Prencode=7) and (modwrkcd=2) and (2≤s1f367≤3) and (s1f3612≥3) ⇒ Hybrid

41. (admjob=1) and (2≤s1f362≤3) and (modwrkcd=1) ⇒ Hybrid

42. (admjob=1) and (2≤s1f3612≤3) and (modwrkcd=1) ⇒ Hybrid

43. (admjob=1) and (modwrkcd=2) and (Prencode=7) and (2≤s1f3612≤3) and (s1f3621=4) ⇒ Hybrid

44. (admjob=1) and (modwrkcd=2) and (Prencode=7) and (1≤s1f364≤2) and (s1f3614=2) ⇒ Hybrid

45. (admjob=1) and (4≤s1f362≤5) and (s1f3612≥3) ⇒ Hybrid

46. (admjob=1) and (modwrkcd=2) and (Prencode=7) and (2≤s1f3612≤3) and (s1f364=2) ⇒ Hybrid

47. (admjob=1) and (4≤s1f3621≤5) and (modwrkcd=1) ⇒ Hybrid

48. (admjob=1) and (modwrkcd=2) and (Prencode=7) and (2≤s1f3614≤3) and (90≤ACTOADMC≤134) ⇒ Hybrid

49. (admjob=1) and (modwrkcd=2) and (6≤OCCUPATIONa≤7) and (Prencode=7) and (3≤s1f362≤4) ⇒ Hybrid

50. (admjob=1) and (modwrkcd=2) and (2≤s1f362≤3) and (4≤s1f3618≤5) ⇒ Hybrid

51. (admjob=1) and (modwrkcd=2) and (2≤s1f362≤3) and (197≤ACTOADMC≤404) ⇒ Hybrid

52. (admjob=1) and (1≤s1f3612≤2) and (s1f362=3) and (modwrkcd=1) ⇒ Hybrid

53. (admjob=1) and (7≤OCCUPATIONa≤8) and (s1f362≥5) ⇒ Community Single Service Provider

54. (s1f365≥3) and (curwork=1) and (s1f364≥3) and (0≤OCCUPATIONa≤1) ⇒ Consider vocational rehab/no further rehab

55. (admjob=0) and (0≤s1f364≤1) and (0≤s1f3614≤1) and (0≤s1f365≤1) ⇒ Community Single Service Provider

56. (s1f365≥3) and (curwork=1) and (VASa=3) and (s1f3618≥5) and (50≤ACTOADMC≤90) ⇒ Community Single Service Provider

57. (s1f364≥3) and (s1f3614=4) and (0≤OCCUPATIONa≤1) ⇒ Consider vocational rehab/no further rehab

58. (ACTOADMC≥2152) ⇒ Consider vocational rehab/no further rehab

59. (1375≤ACTOADMC≤2152) ⇒ Consider vocational rehab/no further rehab

60. (660≤ACTOADMC≤1375) and (0≤s1f364≤1) and (modwrkcd=0) ⇒ Community Single Service Provider

61. (s1f365≥3) and (curwork=1) and (50≤ACTOADMC≤90) and (s1f3618≥5) and (Prencode=7) ⇒ Community Single Service Provider

62. (s1f3614≥5) and (s1f3621=4) and (s1f367≥3) and (0≤ACTOADMC≤50) ⇒ Community Single Service Provider

63. Default ⇒ Provider-based (Functional Restoration Program)

A.2 Negative Rules

1. (curwork=1) and (s1f3612≥2) and (s1f3612≤3) ⇒ Not Hybrid

2. (curwork=1) and (s1f365≥2) and (s1f365≤3) ⇒ Not Hybrid

3. (curwork=1) and (s1f3613≥1) and (s1f3613≤2) ⇒ Not Hybrid

4. (curwork=1) and (s1f367≥2) and (s1f367≤3) ⇒ Not Hybrid

5. (curwork=1) and (s1f3613≥2) and (s1f3613≤3) ⇒ Not Hybrid

6. (s1f3621≥5) and (s1f3621≤6) and (Diag-Grp=7) ⇒ Not Complex

7. (ACTOADMC≥946) and (ACTOADMC≤3275) and (Diag-Grp=7) and (s1f365≥1) and (s1f365≤2) ⇒ Not Complex

8. (VASa≥9) and (VASa≤10) and (ACTOADMC≥946) and (ACTOADMC≤3275) ⇒ Not Complex

9. (s1f3625≥2) and (s1f3625≤3) and (Diag-Grp=3) ⇒ Not Complex

10. (s1f3618≥1) and (s1f3618≤2) and (Diag-Grp=7) ⇒ Not Complex

11. (s1f367≥2) and (s1f367≤3) and (Diag-Grp=3) ⇒ Not Complex

12. (VASa≥9) and (VASa≤10) and (ACTOADMC≥375) and (ACTOADMC≤946) ⇒ Not Complex

13. (s1f3621≥5) and (s1f3621≤6) and (Diag-Grp=3) ⇒ Not Complex

14. (ACTOADMC≥946) and (ACTOADMC≤3275) and (Diag-Grp=7) and (s1f367≥2) and (s1f367≤3) ⇒ Not Complex

15. (s1f365≥1) and (s1f365≤2) and (Diag-Grp=7) ⇒ Not Complex

16. (s1f367≥1) and (s1f367≤2) and (Diag-Grp=3) ⇒ Not Complex

17. (OCCUPATIONa≥8) and (OCCUPATIONa≤9) and (Diag-Grp=7) ⇒ Not Complex

18. (s1f3621≥6) and (ACTOADMC≥946) and (ACTOADMC≤3275) and (s1f3612=2) ⇒ Not Complex

19. (Diag-Grp=8) and (admjob=0) ⇒ Not Other (not prog0)

20. (curwork=1) and (Diag-Grp=6) ⇒ Not Other (not prog0)