Intelligent Medical Decision Support for Predicting Patients at Risk in Intensive Care Units

Dissertation

for the award of the degree
"Doctor of Philosophy" Ph.D.
Faculty of Mathematics and Computer Science
of the Georg-August-Universität Göttingen

within the doctoral program Ph.D-program in Computer Science (PCS)
of the Georg-August University School of Science (GAUSS)

submitted by

Araek Sami Tashkandi
from Jeddah, Saudi Arabia

Göttingen
in October 2020

Thesis Committee

First supervisor	Prof. Dr. Lena Wiese Institute for Computer Science, Goethe-Universität Frankfurt
Second supervisor	Prof. Dr. Ramin Yahyapour Institute for Computer Science, Georg-August-Universität Göttingen

Members of the Examination Board

Reviewer	Prof. Dr. Lena Wiese Institute for Computer Science, Goethe-Universität Frankfurt
Second Reviewer	Prof. Dr. Ramin Yahyapour Institute for Computer Science, Georg-August-Universität Göttingen

Further members of the Examination Board

Prof. Dr. Dagmar Krefting
Institute for Medical Informatics, Georg-August-Universität Göttingen

Prof. Dr. Burkhard Morgenstern
Institute for Microbiology and Genetics, Department for Bioinformatics,
Georg-August-Universität Göttingen

Prof. Dr. Ulrich Sax
Institute for Medical Informatics, Georg-August-Universität Göttingen

Prof. Dr. Armin Schmitt
Breeding Informatics Group, Department of Animal Sciences,
Georg-August-Universität Göttingen

Date of the oral examination: 27.11.2020

Acknowledgement

I would like to express my sincere gratitude and appreciation to all the people who support me to deliver this Ph.D. thesis. I thank God for the success he delivers me to.

Firstly, a big thanks and appreciation to my wonderful advisor Prof. Lena Wiese to whom I'm incredibly thankful. Without your precious support, valuable supervision, and endless motivations, it would not be possible to deliver this notable thesis. I am lucky to have you as a supervisor. A special thanks to my second supervisor Prof. Ramin Yahyapour. Thanks for your insightful guidance and feedback throughout this Ph.D. research.

Heartfelt thanks to my lovely family, who have been by my side to get me through the Ph.D. process successfully. I could not have done it without you, your motivations, and standing by me with the most difficult and challenging times. My first and forever love my lovely parents; I am profoundly grateful for your prayers and constant encouragement that give me the strength to continue working hard and success. My husband, the love of my life, and my forever supporter, thank you from the depth of my heart. My two angels, my sons Elias and Adam, hope I make you proud of your Mom; I love you.

I am sincerely grateful to the University of Jeddah and King Abdulaziz University, who made this Ph.D. journey possible by the financial support.

Abstract

Early detection of at-risk patients has great importance in Intensive Care Units (ICUs) to improve patient healthcare and save patients' life outcomes. The severity of illness scores have been used for predicting the patients' risk of mortality. However, their poor accuracy is a weakness. Thus, Machine Learning (ML) models are exploited for decision support for this goal. Some challenges have to be overcome to achieve accurate predictions of the risk of mortality – for instance, finding important medical measurements or features that influence the prediction. Imbalanced class distribution is a major obstacle (i.e., the number of patients with risk is much less than the patients without), which produces the so-called accuracy paradox problem.

Researchers of the related work applied ML models and different methods in order to handle those challenges. However, the important details and comparison between different methods are still missing. Hence, this thesis presents an overview of implementing the main building block of this medical decision support. It leverages the ensemble ML model, the Gradient Boosting Decision Tree (GBDT). The GBDT shows its performance even with the imbalanced data. Moreover, this thesis provides detailed steps for implementing the model and for pre-processing the data. Comparisons between different ML models, methods of feature selection, and handling imbalanced data are provided and tested on a real-world ICU dataset. Furthermore, an efficient cluster-based under-sampling method to handle imbalanced data is implemented.

Predicting the risk of mortality in the related work is generic (i.e., for patients with different diseases). Some works are done on predicting mortality based on patients similarity on a large number of features (which has a weakness of High computational time and complexity). In this thesis, an approach to avoid this computational complexity and for optimizing the prediction accuracy of predicting the risk is represented and implemented. This approach is based on mortality prediction for similar patients with the same disease classification.

This thesis work is compared to the related works and the commonly used severity of illness score and verified on another ICU dataset. The result shows the significant performance improvement over the severity scores and the related works and the high accuracy on the other dataset. Moreover, the achieved result – specifically, the high prediction performance of the critical cases of patients at risk (i.e., the rare cases of the minority class) – is promising. Area under the curve (AUC) of 0.956 is achieved.

Contents

List of Figures

List of Tables

Acronyms

PSM Patient Similarity Metric

EMR Electronic Medical Record

DBMS Database Management System

NoSQL Not only SQL

SQL Structured Query Language

ICU Intensive Care Unit

KNN K-nearest neighbor

LR Logistic Regression

DT Decision Tree

GBDT Gradient Boosting Decision Tree

SVM Support Vector Machine

ROC Receiver Operator Characteristic Curve

PRC Precision Recall Curve

AUC Area under the ROC Curve

AUPRC Area under the PRC

ML Machine Learning

ICD-9 International Classification of Diseases-9

ICD International Classification of Diseases

IMDSS Intelligent Medical Decision Support System

MIMIC Medical Information Mart for Intensive Care

MIMIC-III Medical Information Mart for Intensive Care-III

MIMIC-II Medical Information Mart for Intensive Care-II

SMOTE Synthetic Minority Over-sampling Technique

1

Introduction

This chapter presents the problem statement of the current issues in this thesis's research field, which will be addressed. It provides a list of the research questions that the thesis aims to answer. Then, it gives a summary of the thesis contributions to handle the described problems. It shows the author's publications of the thesis intermediate results. Finally, it represents the structure of the thesis.

Contents

1.1 Problem and Motivation

With advanced health information technology for electronically collecting patients' data through different sources, a vast amount of medical data has been available. In Intensive Care Units (ICUs), massive medical data is generated on an hourly basis and stored in electronic medical records (EMRs). These data contain laboratory values, vital signs, demographics, and others. To protect patients' life, doctors should keep monitoring these massive measurements. They are overwhelmed with this vast amount of data, and they might overlook some critical measures which lead to sudden death (which is a serious problem the ICU's patients suffer).

A crucial task that can maximize patient health care and can help to minimize the number of sudden deaths in ICUs is the identification of patients with acute health risks. The medical intervention should be in the right moment to save a patient life. Thus, early assessment of a patient's risk of death is essential.

The ICU patients with life-threatening illnesses whom continuous monitoring generates a large overwhelmed amount of data need a computerized system that helps in proactive care [73]. This task is accomplished by implementing an Intelligent Medical Decision Support System (IMDSS) that predicts the risk of death for the patients. This system harnesses the information wealth of this vast amount of data of EMRs to support medical decision making. This system analyzes patient data collected regularly during the patient's ICU stay to give a prediction or an alarm of patient death risk. As a result, proactive medical interaction can be achieved to save patient life.

The accuracy of prediction is one of the most critical characteristics of this intelligent system. The medical data's technical issues as the curse of dimensionality, missing values (sparsity), and class imbalance problems are significant challenges when implementing the system. For more details, see (Lee and Yoon 2017; Johnson et al. 2016). These issues are accuracy factors to this IMDSS. Moreover, there are other accuracy factors that affect the prediction performance of this system, as described in the following:

- The ML model and its parameters: The intelligence of this system comes from its main building block, which is a Machine Learning ML model. The ML model is trained on the medical data to learn how to accurately predict the patient's case. The selection of the ML model and the values of the parameters have a significant impact on system performance.

- Curse of dimensionality: is when high dimensional data causes many issues such as it makes the classifier decision boundaries difficult. We have to find which features (i.e., medical measurements data from EMR) contribute most to the prediction to increase the model accuracy. Feature selection is one approach to handle high dimensional data.

- Class Imbalance: As stated by (Li et al. 2010): "In medical data sets,

data are predominantly composed of "normal" samples with only a small percentage of "abnormal" ones, leading to the so-called class imbalance problems.". An imbalanced real-world ICU dataset contains a majority percentage of the patients that survived and a minority percentage of the patients that died. The issue of imbalanced class distribution causes the classifier to be extremely biased towards the majority class and discounting the minority class. Nevertheless, the minority class is the class of interest.

Different Machine Learning ML models can be leveraged to implement this intelligent system. The selection of the predictive model influences the system accuracy. Researches have been using the advances of ML to develop such an IMDSS for predicting the risk of mortality for ICU patients (Ghassemi et al. 2015; Luo et al. 2016).

Implementing this system requires considerable effort and many steps and decisions.

Researches have been using the advances of ML to develop such an IMDSS for predicting the risk of mortality for ICU patients (see related work Chapter 2). They consider some methods to overcome these problems; however, there are still issues to solve, for example:

- The selection of the model parameters was ambiguous. The selection of the model parameters in case of imbalanced data is based on optimizing the accuracy (the so-called accuracy paradox problem occurs) or another metric.
- Different methods were used for handling the accuracy factors and the medical data issues that we previously mentioned. Less attention has been giving to study and compare various accuracy factors and different performance optimization methods.
- Some studies for predicting the risk of death were for specific diseases, and others were in generic (homogeneous data). No effort has been made to compare the two approaches to show, which gives higher accuracy.

In mortality prediction and health prediction in general besides ML models, the patient similarity is another concept to mention. Identifying similar patient cases to the new patient helps in predicting different clinical outcomes for medical decision support. Patient similarity analysis can be applied to different applications. However, the central focus of the hospitals the ICUs need patient similarity-based personalized-predictive modeling [73]. There are much research works on leveraging patient similarity for implementing health or mortality prediction. In this thesis, both concepts (i.e., ML models, and patient similarity) are used. The ML models are used to implement the predictive model and similarity between patients is used in the KNN model and clustering under-sampling method. Moreover, to optimize the computational burden problem of patient

similarity on large dimensions, I implement an approach of similarity solely on one feature (ICD code).

1.2 Research Questions

In this thesis, we want to answer the following questions:

I Accuracy of Prediction:

- What are the main factors that affect the accuracy of predicting patients at risk in general? And which factors affect prediction models?
- Does the ML model selection affect the accuracy of the prediction? If yes, which ML model can give the highest performance for risk prediction?
- How does the selection of the features or the predictor variables affect the accuracy of prediction?

II Effect of Dataset on Prediction:

- How does the imbalanced dataset affect the accuracy of model prediction? And to what extent?
- Does focusing on a specific disease (i.e., filtering patients by disease code) improve the accuracy of the risk prediction? Or is a heterogeneous dataset (i.e., patient with different disease codes) enough?

III Performance Optimization:

- What are the different performance optimizations that can be done?
- Will ML models improve the prediction accuracy than the standard severity scores for patient's risk prediction?
- Considering the run time and the accuracy, what is the best combination of the ML model and the feature selection method?

1.3 Thesis Contributions

To handle the problems described in Section 1.1, and to provide support in the existing gaps, this thesis makes the following contributions:

- Implementing this system requires considerable effort and many steps and decisions. It gives a first level overview of model deployment steps for this IMDSS system. Figure 1.1 represents this overview that also summarizes the contributions of this thesis, which are discussed in the following.

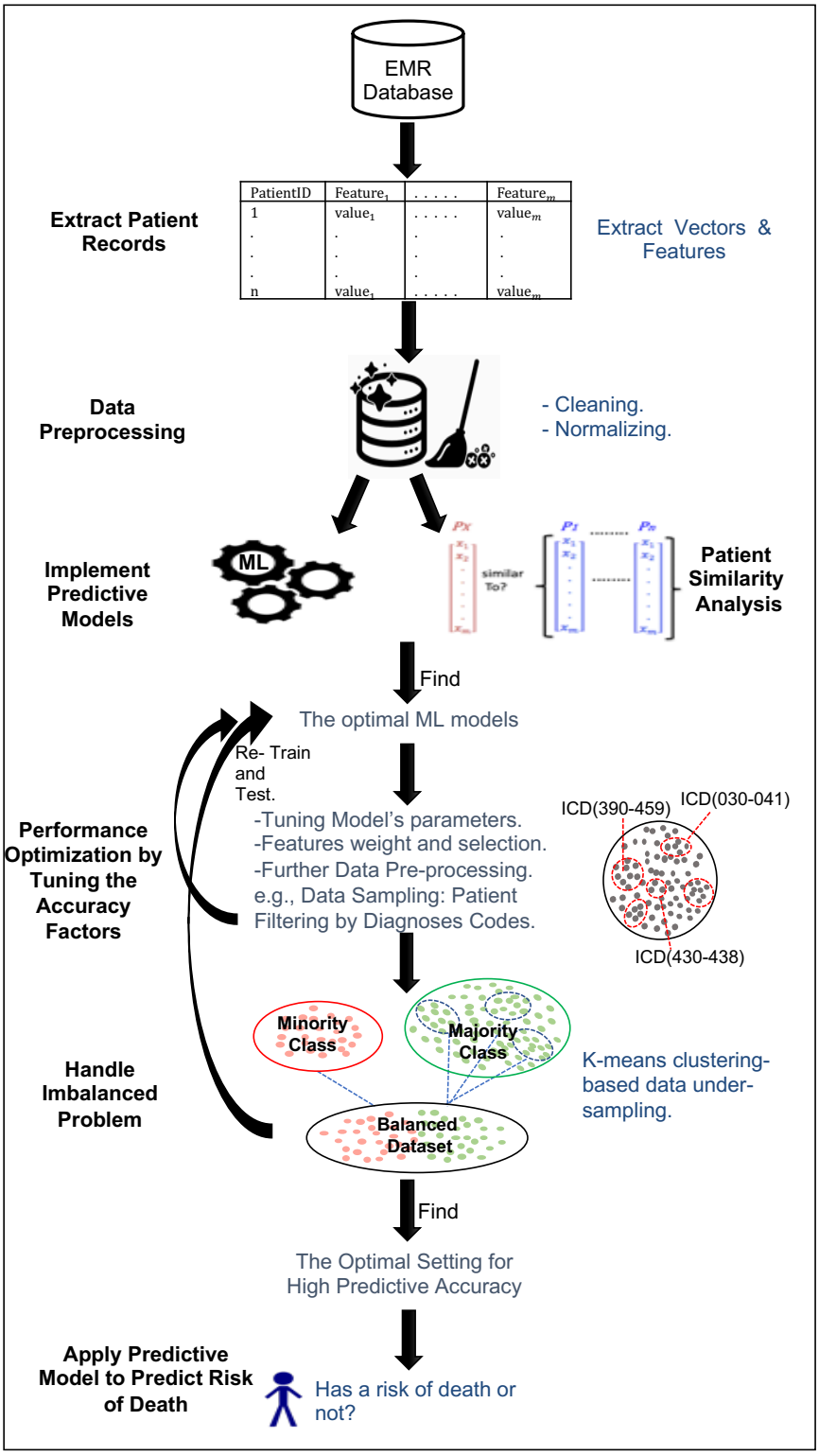


Figure 1.1: Overview of Model Deployment Steps for IMDSS System for Predicting Risk of Death

- It provides a comprehensive study of different accuracy factors that affect the prediction performance of this system. This was done partially or ambiguously in the previous literature. I aim to reach the optimal setting for accurate prediction. I conduct a practical performance comparison for different models and different performance optimization methods. The results allow making statements in a broader context in contrast to the un-detailed works provided in most other research works in this field.
  - I develop different ML models and compare their performances. I compare the prediction performance of seven ML models such as Logistic Regression, Gradient Boosted Decision Tree, and K-Nearest Neighbors.
  - I discuss in detail the selection of the ML models' parameters. I compare the selections that optimize the accuracy and other metrics.
  - I compare different feature selection methods for handling high dimensionality. Filter and wrapper approaches are tested.
  - I test some data sampling methods for handling the imbalanced data problem. Some under-sampling and over-sampling methods are tested and compared.
- A big real-world medical dataset is used to test these approaches. The patients' data are extracted from the MIMIC-III database [40]. The models are applied on data of the first 24 hours in the ICU stay to predict the in-hospital risk of death for ICU patients. As previously stated, many studies use this publicly available dataset. Furthermore, the developed code is available in the code hosting platform GitHub (`https://github.com/Araek/Mortality-Prediction`). Thus, my work serves as a benchmark.
- After extracting several features (predictor variables), feature selection methods are applied to consider only the most efficient features. I use the filter and wrapper approaches. Rather than relying on a large number of features (that causes computational cost) to improve accuracy as what Purushotham *et al.* [68] did, I use the features that contribute most to the prediction to increase the accuracy.
- The imbalanced data affect the performance of the ML models. However, our implemented model GBDT had a significantly higher performance than other tested models, even on the imbalanced data without any optimization (area under the curve (AUC)=0.859). Moreover, it outperforms the prediction performance of some of the previous studies on similar versions of our used dataset [23, 59, 50, 55]. Furthermore, it significantly outperforms the standard severity scores as SAPS and SOFA.

- I develop a clustering-based data under-sampling method to overcome the critical imbalanced data problem. The best performance (AUC = 0.956) is achieved by under-sampling the data by k-means (which took only a few seconds).

- I implement the approach of mortality prediction for similar patients (with the same disease classification). As discussed previously some works are done on predicting mortality based on data of similar patients on a large number of features. Rather than finding similarity between patients in many features that have high computational time and complexity, I build the ML model on top of similar patients based only on one feature, which is the ICD. This avoids the complexity and the computational burden of similarity calculation of a large number of features. In addition, it improves the accuracy of the predictive model. Furthermore, it outperforms the previous works done on implementing the ML model on top of patient similarity of large features [50, 48]. Thus, we leverage the similarity and ML model.

1.4 Thesis Impact

This Ph.D. research work achieved a good amount of publications in journals, conferences, and workshops [78, 77, 88, 79, 80, 72]. During the accomplishment of this thesis, intermediate results have been published in the following peer-reviewed conference proceedings and journals. In addition, the author contributed to some related work.

- **Araek Tashkandi**, Lena Wiese. A Hybrid Machine Learning Approach for Improving Mortality Risk Prediction on Imbalanced Data, Proceedings of the 21st International Conference on Information Integration and Web-based Applications and Services (iiWAS), pages 83-92, ACM, 2019.

- **Araek Tashkandi**, Lena Wiese. Intelligent Medical Decision Support System for Predicting Patients at Risk in Intensive Care Units, Proceedings of the PRE-ICIS SIGDSA SYMPOSIUM, Association for Information Systems Special Interest Group on Decision Support and Analytics (SIGDSA), AIS eLibrary, 2019.

- **Araek Tashkandi**, Ingmar Wiese, Lena Wiese. Efficient in-database patient similarity analysis for personalized medical decision support systems, Big Data Research Journal, volume 13, pages 52-64, Elsevier, 2018.

- Ingmar Wiese, Nicole Sarna, Lena Wiese, **Araek Tashkandi**, Ulrich Sax. Concept acquisition and improved in-database similarity analysis for medical data, Distributed and Parallel Databases Journal, volume 37, pages 297–321, Springer, 2018.

- Nicole Sarna, **Araek Tashkandi**, Lena Wiese. Patient Similarity Analysis for Personalized Health Prediction Models (abstract), Proceedings of the European Conference on Data Analysis (ECDA), 2018.
- **Araek Tashkandi**, Lena Wiese. Leveraging patient similarity analytics in personalized medical decision support system (abstract). Proceedings of the Learning, Knowledge, Data, Analytics (LWDA), 2017 FGDB Database Workshop, page 125, CEUR-WS, 2017.

1.5 Thesis Structure

The thesis is structured as follows. Chapter 2 "Related Work" provides a survey of related approaches. The researches that have been done for mortality prediction either by applying patient similarity or by ML models are surveyed and compared. Chapter 3 "Background" presents the approaches for health prediction (i.e., patient similarity and ML models) and for the implementation. It includes a description of the used dataset and the extraction and pre-processing steps. It defines the different metrics for evaluating the predictive performance of the models in specific in case of imbalanced data problem. Moreover, it represents the factors for performance optimization of the predictive model, which are discussed in the following chapters. Chapter 4 "The predictive Model" provide a detailed definition of different ML models that can be used for mortality prediction and implementing the ML models. Furthermore, it discusses the models' weaknesses and strengths, crucial models' parameters, and tuning those parameters to select the optimal values. It provides a comparative analysis of different ML models for the task at hand and chooses the best-performing candidate. Chapter 5 "Performance Optimization" shows the result of different data preprocessing and feature selection methods for performance optimization. It defines and implements a data sampling method by filtering patients by diagnoses code for improving the model prediction accuracy. Chapter 6 "Handle Imbalanced Classes" provides an overview of this problem of imbalanced classes. It describes different data sampling methods (under-sampling and over-sampling) and provides the conducted experiment to compare them. Furthermore, it reveals our implemented clustering-based under-sampling approach that helps to achieve high predictive performance. In the end, it applies clustering-based under-sampling after filtering patients by diagnoses code for handling imbalanced classes. Chapter 7 "Verification" presents verification of our approach against the severity of illness scores and against another large dataset. Chapter 8 "Conclusion" discusses the outcomes of this thesis in the context of the related work to represent its contributions. Moreover, it summarizes the findings and works that have been done to answer the thesis's research questions. Finally, it discusses some of the future works.

2

Related Work

The following chapter provides a survey of related works for the two approaches for implementing patient health risk prediction. The review is started with an overview of utilizing patient similarity for mortality prediction and other health prediction purposes. Then an overview of using Machine Learning (ML) models for mortality prediction is provided. For each approach, a summary is provided with a literature comparison. In the end, a review of some research works on handling imbalanced data by clustering-based under-sampling and by ensembl ML models is provided.

Contents

2.1 Health Prediction by Patient Similarity

A considerable amount of research work has discussed the topic of patient similarity analysis, which is an approach for health prediction. Different patient similarity metrics and different predictive models are employed. Besides all of this, the aims of deploying patient similarity analysis vary from disease diagnoses to mortality prediction. We will present in this section some of the related works for analyzing patient similarity first for mortality prediction and then for other predictive approaches.

2.1.1 Patient Similarity for Mortality Prediction

Predictive models for mortality prediction by Lee et al. [49]

They implement three predictive models for mortality prediction: logistic regression (LR), support vector machine (SVM), and decision tree (DT). AdaBoost is applied to improve the predictive performance of the DT. 10-fold cross-validation was used to train and test the models. The predictive performance was measured for each predictive model by AUC. 9269 is the data size of the first ICU admissions of all adult patients from MIMIC-II.

They show that the worst predictive model is DT, while the improved DT by AdaBoost resulted in the best performance. Moreover, LR and SVM resulted in similar high performance to AdaBoost. They claim that their results are comparable to SOI scores, whereas unlike these scores (using diagnoses and procedures next to demographic and clinical data), their models only use clinical and demographic data.

Personalized data-driven decision support system for mortality prediction Lee et al. [50]

Similar to [49], they apply LR and DT models next to death counting. However, this work aims to improve the predictive performance of [49] by implementing personalized mortality prediction by deploying cosine-similarity-based metric. By defining the most similar patients to each patient in the medical measurements of the first 24 hours in the ICU, 30-day mortality prediction models were the result.

They show that the best prediction performance was achieved by LR, whereas the worst by DT. A good predictive performance can be achieved by death counting (only among 60 to 100 similar patients). Their main hypotheses related to higher prediction performance being achieved by analyzing only similar patients were approved. Moreover, they claim that their approach of patient similarity metric outperformed the well-known ICU severity of illness scores.

Patient Based Predictive Modeling framework for ICU mortality prediction by Morid et al. [59]

The belief that local approximation of similarity-based method is best fit for complex health prediction problems rather than general approximation models is the basis of their framework. The K-Nearest Neighbor (KNN) of a Similarity-based classification approach is utilized to find similar patients to a current one. The time-series data that are extracted from the first 48 hours of the ICU stays are used, unlike [50]. The required output is mortality prediction before hospital discharge. They approve that feature weighting has a significant contribution to improve prediction accuracy for the high dimensional mortality prediction (i.e., which might involve as minimal 40 predictors). In their framework, the wrapper approach Gradient Descent is used to iteratively update the features.

Their method outperforms the non-similarity method for ICU mortality prediction (i.e., the severity of illness scores).

Patient Similarity vs. Predictive Models by Hoogendoorn et al. [32]

Two different approaches are used and compared for mortality prediction, the Patient similarity approach, and the predictive model approach. The K-Nearest Neighbor (KNN) is selected for computing similarity between the patients. Euclidean distance is computed next to Keogh lower bounds and a penalty for the unmatched features. The logistic regression (LR) model with 50 features is compared to this nearest neighbor approach with 132 features. They found that the predictive accuracy and the run-time of KNN are worse than the ones of LR. However, that might be linked to the number of features.

2.1.2 Patient Similarity for Different Predictive Approaches

Patient Similarity for Medical Prognosis by Wang et al. [85]

This work from the healthcare analytics research group of IBM develops SimProX system for providing prognosis to predict the patients' future health. It stands for SimProX (Similarity-based Prognosis with eXperts' inputs). It is based on patient similarity and expert/physician feedback. The key component is similarity assessment, whereas physician feedback is only to check if the retrieved cohort of similar patients is really similar to the queried patient.

The patient vectors are embedded into an intrinsic space by the Local Spline Regression (LSR). After that, the Euclidean distance is used to measure the pairwise distance between patient vectors. The SimProX is evaluated by comparing its precision, recall, accuracy and F-measure against Locally Linear Embedding (LLE), Laplacian Embedding (LE), Principal Component Analysis (PCA), and simple Euclidean Distance (ED). The result shows that SimProX's performance outperforms the others.

Patient Similarity for Medical Prognosis by Sun et al. [76]

The similarity between patients is defined for prognosis prediction of occurrence of the acute hypotensive episode (AHE). They develop Locally Supervised Metric Learning (LSML) that automatically adjusts the importance of the predictor variables. It is based on Mahalanobis distance.

The evaluation of their developed system is by measuring the accuracy of classification, retrieval, and prognosis performance. The authors approve the result of [64] that the performance in terms of classification and retrieval of the supervised similarity measure outperforms the unsupervised approaches.

Patient Similarity for Medication Plan by Panahlazar et al. [64]

The patient similarity is utilized for predicting treatment plan for Heart Failure (HF) patients. Supervised and unsupervised clustering approaches cluster patients that have a good response to HF therapy. The k-means and hierarchical clustering are used for unsupervised clustering, and then each cluster has a label of the most frequent medication plan. The class variable of the medication plan is used for supervised clustering. Finally, the patient similarity is measured between a queried patient and the clusters by Mahalanobis distance. Finding the cluster a patient is most similar to, the response to the HF therapy can be predicted, and then the medication recommendation can then be decided.

The performance of their approach is evaluated against specificity, sensitivity, F1, accuracy, and AUC. The supervised clustering approach has superior performance followed by the hierarchical, then the k-means approach. Further validation with a larger dataset is required for their result since their used data set is relatively small, which had N=1386 patients.

Patient Similarity for Discharge Diagnoses by Gottlieb et al. [24]

Rather than using the diagnoses information of EHR in the patient similarity method such as [85], this research work utilized the patient similarity method with some information from patient records to infer the discharge diagnoses. Ten patient similarity measurements are calculated. Two similarity measurements are for ICD code, using the coding hierarchy and the co-occurrences with computing Jaccard score. The rest is to measure similarity for other hospitalization information such as medical history, blood test, age, gender, and ECG. Different methods are used as Euclidean distance for the blood test similarity and Boolean values 0 and 1 for gender similarity. These similarity measures are normalized to the range [0, 1] and combined.

2.1.3 Summary

The related work leverages different methods of patient similarity and different types of information from EHR for various predictive approaches. There is a

general agreement by these works on the outperforming of patient similarity-based methods over the general severity of illness scores on mortality prediction for ICU patients. The reason behind that is accurate mortality prediction should consider the behavior for the specific patient rather than a general approximation.

Table 2.1 compare the surveyed works on the purpose of using patient similarity, the used patient similarity metric, and the evaluation metric along with the implementation tools that are used. In this thesis, the focus will shed light on applying patient similarity in KNN approach for mortality prediction and in clustering under-sampling for handling the imbalanced class distribution problem. Different patient similarity metrics will be used and compared — various evaluation metrics are used to evaluate the performance.

So far the focus is mainly to measure the performance of prediction accuracy. A focus on improving computational performance still seems to be missing. The previous works face the computational burden of calculating patient similarity on high dimensional data. We leverage patient similarity solely on one feature (ICD International Classification of Diseases code) to optimize the accuracy of mortality prediction. Moreover, to avoid the computational burden of similarity calculating between a large number of features.

2.2 Health Prediction by Machine Learning Models

Machine Learning (ML) Models is another approach for predicting patient health. It has been used for achieving high prediction accuracy. In the following, we will present a review of some of the works that use ML Models for predicting the risk of death.

Personalized data-driven decision support system for mortality prediction Lee et al. [50]

They use Logistic Regression (LR) and Decision Trees (DT) for predicting 30 days mortality. No particular feature selection method was used. For handling the imbalanced class distribution problem, the 10-fold cross-validation incorporated stratified sampling. The highest AUC is equal to 0.830.

Patient Based Predictive Modeling framework for ICU mortality prediction by Morid et al. [59]

They use k-Nearest Neighbor for mortality prediction using data collected during the first 48 hours of ICU admission. They extracted 36 time-series features. However, the average of the feature's value per two hours is calculated to have in total 24 values of each feature. Moreover, they have four statics features. The wrapper approach of Gradient Descent was used for feature weight. They

Study	Patient Similarity for	Patient Similarity Metric	Evaluation Metrics	Implementation Tool
Sun *et al.* [76]	Prognosis prediction of occurrence of Acute Hypotensive Episode (AHE)	Locally Supervised Metric Learning (LSML), Mahalanobis distance	Accuracy of: Classification, Retrieval, and Prognosis	Developed system
Wang *et al.* [85]	Disease prognosis	Euclidean distance	Classification performance: precision, Recall, Accuracy, and F-measure	Developed SimProX system
Morid *et al.* [59]	Mortality prediction	Classification with K-Nearest Neighbor algorithm	Classification performance: precision, Recall, and F-measure	Developed PPMF Framework
Lee *et al.* [50]	Mortality prediction	Cosine Similarity metric	Accuracy: Area under ROC curve; area under precision-recall curve	R
Hoogendoorn *et al.* [32]	Mortality prediction	K-Nearest Neighbor, Euclidean distance	AUC for: -Influence of the number of patients upon the predictions accuracy and the computation time. -Earliest Prediction Time	Python
Panahiazar *et al.* [64]	Medication plan	Supervised and unsupervised clustering approaches with Mahalanobis distance	Sensitivity, specificity, F-measure, and accuracy	Developed framework
Gottlieb *et al.* [24]	Predict the eventual discharge diagnoses	Jaccard, Euclidean distance and others	AUC and F-measure	MATLAB

Table 2.1: Literature Comparison of Using Patient Similarity for Mortality Prediction

did not handle the imbalanced data problem. They provide the best F-measure of 0.66.

A multivariate time-series modeling approach to severity of illness assessment and forecasting in ICU with sparse, heterogeneous clinical data by Ghassemi et al. [23]

They used Lasso logistic regression and L2 linear kernel Support Vector Machine. From MIMIC-II, they extracted a total of 313,461 notes from nursing, physicians, labs, and radiology recorded before the patient's first discharge from ICU. They predict in-hospital mortality (i.e., before discharge) and 1-year post-discharge mortality. They had a class imbalance problem where the in-hospital mortality rate was 10.9%. To handle this issue in the training set, they had a minimum 70%/30% ratio between the negative and the positive classes by randomly sub-sampling the negative class. The best AUC of in-hospital mortality is 0.812.

Predicting ICU mortality risk by grouping temporal trends from a multivariate panel of physiologic measurements by Luo et al. [55]

They proposed an unsupervised feature learning algorithm for analyzing patterns in clinical time-series data. The authors aim to improve the interpretability and accuracy of the predictive model. They introduced Subgraph Augmented Non-negative Matrix Factorization (SANMF) to convert the ICU time series data into a graph representation. Then, from frequent subgraphs, they extract the temporal trends of the physiologic variables. Non-negative matrix factorization discovers a group of patients on those trends (by grouping those trends). The resulted trend groups are the features to train the predictive model.

They use the Logistic Regression model for mortality risk prediction for predicting within 30-days mortality (including in-hospital mortality or after ICU discharge mortality). The patient data were extracted from MIMIC-II of the period between 12 and 24 hours after ICU admission. The dataset was skewed, which has 788 to 7075 for the suffered patients to surviving patients, respectively. However, they didn't handle the imbalanced data problem. Their model had an AUC =0.848.

Multitask learning and benchmarking with clinical time series data by Harutyunyan et al. [28]

They developed four prediction tasks: in-hospital mortality, physiologic decompensation, length of stay (LOS), and phenotype classification. They stated it is a public benchmark suite derived from the publicly available MIMIC-III database. They extracted 17 clinical variables from the first 48 hours of the ICU stay.

They compare the performance of these models: Logistic regression (LR), Standard LSTM, channel-wise LSTM, deep supervision, multitask standard LSTM, and multitask channel-wise LSTM. They extracted 17 predictor variables, but the number of the extracted features from them was not mentioned. In-hospital mortality based on the first 48 hours of an ICU stay gives the best AUC of 0.870 by multitask channel-wise LSTM. While in this thesis approach, we use less data, which is only the first 24 hours data of the ICU stay. We get higher AUC than them with handling imbalanced data.

Benchmarking deep learning models on large healthcare datasets by Purushotham et al. [68]

They compare the performance of Super Learner models and Deep Learning models. They extracted three sets of features based on features used in SAPS-II score and low missing rate. No specific feature selection methods were used. With 24 hours of ICU stay dataset, they achieved with the set 'A' an AUC of 0.8673, with dataset 'B' an AUC of 0.8730. With the dataset 'C,' which is the largest set (136 features), they have AUC of 0.9410. The best performance they achieved is with the most significant number of features. This comes with the computational cost of an hour.

Our model GBDT reaches with imbalanced dataset AUC of 0.859 and with the balanced dataset (with K-means under-sampling) AUC =0.956. The best performance they achieved is with the largest number of features (which took an hour) while in our work, the best performance we achieved by under-sampling the data by k-means (which took few seconds).

Mortality prediction in intensive care units (ICUs) using a deep rule-based fuzzy classifier by Davoodi and Moradi [15]

They proposed Deep Rule-Based Fuzzy System (DRBFS) to predict in-hospital mortality for the ICU patients. The stacked generalization principle inspires their proposed model, which is the Deep Takagi-Sugeno-Kang Fuzzy Classifier (D-TSK-FC) [92]. They leverage its strengths and overcome its weaknesses by using fuzzy clustering. The supervised fuzzy clustering technique is employed for fuzzy rule generation. They aim to build a model suitable for big data and mixed attribute variables.

Medical features were extracted from the first 48 hours of the ICU stay. No particular feature extraction method is used to overcome high dimensionality. They only compute the significance of an attribute towards the clustering process [2]. They evaluate their model against different ML models, including Decision Trees (DT), Naïve Bayes (NB), and Deep Belief Network (DBN) on MIMIC-III dataset. The random under-sampling is used to handle the imbalanced data. Their model outperforms the other by AUC=0.739.

Real-time mortality prediction in the Intensive Care Unit by Johnson and Mark [38]

They evaluate the ML models logistic regression (LR), logistic regression with an L1 regularization penalty using the Least Absolute Shrinkage and Selection Operator (LASSO), logistic regression with an L2 regularization penalty (L2), and Gradient Boosting Decision Trees (GBDT). They extracted the patients' data of ICU stays from MIMIC-III. A total of 148 features from physiologic and laboratory measurements were extracted. No further feature selection methods were used. They conducted two experiments that differ in the time window used for data extraction. In the first experiment "benchmarking experiment", the time window was fixed to 24 hours after ICU admission. In the second experiment "real-time experiment", the time window set to a random time during the patient's ICU stay (was varied from 4 to 24 hours).

We will compare our work to their first experiment since we have a fixed time window of 24 hours. They compare the ML models with a set of the severity of illness scores for predicting in-hospital mortality from the first 24 hours of a patient's ICU stay. The GBDT model outperformed the other models and the severity of illness scores by AUC of 0.927. However, no details are provided about the selection of the hyperparameters of GBDT.

2.2.1 Summary

Researches have been working on this topic. In Table 2.2, we compare the previously discussed works based on the accuracy factors: the used ML models, the size of the extracted features and the size of the time window, feature selection methods (to find the optimal feature set after extracting the initial feature set), handling of the imbalanced data problem, and the best-achieved AUC for mortality prediction using similar datasets to ours MIMIC-III.

Researchers have been using the advances of ML to develop an IMDSS for predicting the risk of mortality for ICU patients. They use a different set of ML models from the simple k-Nearest Neighbor to the complex deep learning model. They extracted different variable and feature sizes, and most of them extract a sequence of time-series features from the different periods (e.g., a value from every 3 hours). In most of the related work, the number of the extracted predictor variables was mentioned. However, not all of them specify the final number of features. The time window was either the first 24 hours of the ICU stay or the first 48 hours. In general, after the initial feature set extraction, there was no further use of different feature selection methods to find the optimal feature set. In some works, they use one method for further feature selection. Handling the imbalanced data was either ignored or commonly treated by the low efficient randomly under-sampling. Afterward, when they use a technique for handling imbalanced data or for feature selection, they didn't compare it to other methods. Finally, we find that the works that achieved high AUC were

either because of using more features or a larger time window than ours (e.g., extracted data from the first 48 hours of the ICU stay rather than from the first 24 hours). In this thesis, there is a dedicated chapter for each factor of these accuracy factors (i.e., the ML model, feature selection, and handling the imbalanced data). Chapter 4, Chapter 5, and Chapter 6.

Johnson *et al.* [39] state the difficulty of reproducing the studies that have been done on mortality prediction using MIMIC dataset. Thus, considering this difficulty in this thesis, I do not reproduce the exact datasets or settings of the previous work to benchmark. However, I compare their work and best-achieved performance to this thesis work and its best performance in order to find an optimal setting for highly accurate prediction on this dataset.

Study	Machine Learning Models	Feature size and Time Window	Feature Selection Method	Handle Imbalanced Data	Best Performance
Lee *et al.* [50]	LR and DT	76 features from 24 hours	None	cross-validation incorporated stratified sampling	AUC= 0.830
Morid *et al.* [59]	kNN	868 features from 48 hours	wrapper approach (Gradient Descent)	None	F-measure= 0.66
Luo *et al.* [55]	LR	54 variables and 100 features, from 24 hours	non-negative matrix factorization	None	AUC= 0.848
Ghassemi *et al.* [23]	LR and SVM	313,461 notes, time prior ICU discharge	topic modeling for dimensionality reduction	randomly sub-sampling the negative class	AUC= 0.812
Davoodi and Moradi [15]	Deep Rule-Based Fuzzy Classifier	29 variables the feature size not specified, from 48 hours	significance of an attribute towards the clustering process [2]	random under-sampling	AUC= 0.739

Purushotham *et al.* [68]	Super Learner models and Deep Learning models	136 features from 24 hours	None	None	AUC= 0.941
Harutyunyan *et al.* [28]	LR and LSTM-based models	17 variables the feature size not specified, from 48 hours	None	None	AUC= 0.870
Johnson and Mark [38]	LR, LASSO, L2, and GBDT	148 features from 24 hours	None	None	AUC= 0.927

Table 2.2: Literature Comparison of Using ML models for Mortality Prediction

2.3 Handling Imbalanced data by Clustering-based Under-sampling and Ensemble Models

Applications of ML in medical use cases require high reliability of the models. In particular, the models have to be able to handle the class imbalance problem. The investigation of this issue is one of the major focuses of our work in this thesis. To overcome the imbalanced class problem, we rely on the data sampling method – specifically clustering-based under-sampling and ensemble ML.

As shown previously in Table 2.2 that the works on mortality prediction either ignore handling imbalanced data or use the weak randomly under-sampling. Thus, here we will review the works that have been done on applying clustering-based under-sampling in general.

Many related approaches have applied the clustering-based under-sampling technique to class-imbalanced data. We survey the most significant of them here. Lin *et al.* [53] proposed a clustering-based under-sampling method based on K-means. They set the number of the majority class clusters equal to the minority class size. Then, the selection of the majority class representatives follows two strategies: using the cluster centers or using the nearest neighbors of the cluster centers. Ofek *et al.* [61] also used the clustering approach for under-sampling. They aim to consider both computational cost and predictive performance. They cluster the minority class instances and select for each cluster a similar instance number from the majority class. From each cluster, all the minority instances are included, but only some instances from the majority class included. Thus, the number of instances of both classes are the same. The

included majority class instances have to be within a specific distance from the cluster centroid. This distance is the cluster's bound, which equals the distance from the cluster centroid to the farthest minority instance in that cluster.

Tsai *et al.* [81] propose an integrated clustering-based under-sampling method with instance selection algorithms. Affinity Propagation (AP) algorithm is used to cluster the majority class instances just as guidance for K-means to select the k value since it does not require the number of clusters. Then, three different instance selection algorithms are used individually for comparison to select instances from each cluster of the majority class. Finally, the resulted reduced dataset is combined with the minority class instances. Kumar *et al.* [44] use K-means clustering for under-sampling the majority class. First, they eliminate the weak or noisy instances from the majority class. They find the most influencing attributes or features by Correlation-Based Feature Subset Selection method (CFS) then remove ranges of the weak attributes relating to that feature. Second, they combine the majority class's resulted subset with the minority class to be clustered by K-means to remove the most misclassified instances (from both majority and minority sets). Finally, they use C4.5 as the learning algorithm.

Lin *et al.* [53] used a similar clustering under-sampling approach as us (i.e., K-means and the cluster centers are the representatives of the majority class). However, they did not empower it by the ensemble ML model. Similarly, [74] apply k-means in conjunction with KNN for text classification. A comparison of these works is provided in Table 2.3.

Ensemble ML models are another approach to handle imbalanced datasets. Haixiang *et al.* [26] and Galar *et al.* [19] give a survey of the ensemble methods that are used for imbalanced class problem. The ensemble-based models are usually combined either with data re-sampling methods or a cost-sensitive strategy to learn from imbalanced data. In particular, in the recent review done by Haixiang *et al.* [26] there are 218 papers that proposed ensemble models for imbalanced data out from the 527 reviewed articles. The only two papers that used ensemble-based GBDT combined it with the cost-sensitive approach. The high performance of our approach relies on the clustering-based under-sampling in conjunction with an ensemble ML model (GBDT). To the best of our knowledge, there is so far no other extensive research proposing and analyzing GBDT with clustering-based under-sampling for imbalanced data. Moreover, from Table 2.3, these previous research works mainly focus on the clustering-based under-sampling for pre-processing the dataset; then, the learning process from the data is done by applying ordinary ML models or used boosting which is not the gradient boosting but rather the Adaboost. Tsai *et al.* [81] used the boosting ensemble approach but didn't specify the boosting is Adaboost or gradient boosting.

Study	Clustering Method	Number of Clusters	Selection of the Class Representatives	The used ML model
Lin *et al.* [53]	K-means	number of the majority class clusters equal to the minority class size	using the cluster centers or using the nearest neighbors of the cluster centers	MLP, Decision Tree, Random Forest and AdaBoost
Ofek *et al.* [61]	K-means	cluster the minority class instances	from each cluster sample the same number of the minority class instances the majority class instances that are within the cluster's bound	ML
Tsai *et al.* [81]	Affinity Propagation algorithm and K-means	Affinity Propagation algorithm decides the number of clusters	The genetic algorithm, IB3 and DROP3 algorithms	hamming clustering and bagging and boosting ensembles for (C4.5 decision tree, KNN, naive Bayes and MLP)
Kumar *et al.* [44]	K-means	two clusters	the representatives are the remaining instances after removal of weak instances related to the selected features by (CFS) and of the misclassified instances from both majority and minority sets	C4.5

Table 2.3: Literature Comparison of Using Clustering-based Under-sampling for Handling Imbalanced Dataset

3

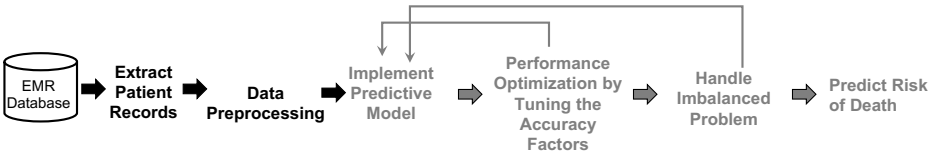
Background

This chapter presents the theoretical foundation and the background knowledge of this thesis topic. First, it introduces the approaches for health prediction, which are patient similarity analysis and ML models. Then, it represents the various alternatives for implementing the predictive models. It describes the used dataset and the selected predictor variables. This chapter also discusses the pre-processing and transformation of the data. Furthermore, the different performance metrics for evaluating the accuracy of the prediction are defined. It concludes by providing performance improvement techniques.

Contents

3.1 Scope of the Chapter

In this Chapter, the first steps of model deployment for predicting the risk of death are represented. First, we will introduce the medical database from where the EMRs of the patient data are extracted. Then, we will define the medical measurements (i.e., the predictor variables and features) that are selected from the patient records. Patient data need to be prepared for similarity analysis and for applying the ML predictive model. Thus, we describe the pre-processing steps we made.



3.2 Approaches for Health Prediction

3.2.1 Machine Learning Models for Health Prediction

Predicting mortality or death risk can be seen as a classification task. It is a binary classifier for two classes; either a patient has a death risk (the positive class with label "1"), or a patient has no risk to death (the negative class with label "0"). Different supervised learning algorithms are employed for this task. In this thesis, I will use some ML models as Logistic Regression and Gradient Boosting Decision Tree (GBDT). The Chapter 4 will describe them.

To make it clear many research papers in this topic call it "mortality prediction," in this paper, we prefer to call it "mortality risk prediction." The reason, because there has been no mortality prediction of 100% accuracy even by ML model, and there are many factors that affect death even after the prediction takes place. However, we use them exchangeably to refer to predicting the risk of mortality.

3.2.2 Patient Similarity-based for Health Prediction

Leveraging patient similarity analysis for health prediction is a case-based reasoning. The health prediction as diagnosis or prognosis of patient x is based on similar previously patient cases to x.

3.2.2.1 Basic Definitions

Before diving into the definition of similarity metrics, there are some basic definitions that have to be known for better understanding.

For a set X the following can be defined:

- **Distance:** A function $d : X \times X \to \mathbb{R}$ that takes two elements in the set X and produces a real number is called a distance or dissimilarity function if it satisfies the following [51, 16] :

  I **Non-negativity:** $d(x, y) \geq 0$.

  II **Symmetry:** $d(x, y) = d(y, x)$.

  III **Reflexivity:** $d(x, x) = 0$.

  IV **Triangle inequality:** $d(x, y) \leq d(x, z) + d(z, y)$.

  Triangle inequality is the factor for efficiency. It assures that all the distance measures describe a distance of the shortest path between two points.

- **Similarity:** A function $s : X \times X \to \mathbb{R}$ is called a similarity function if it is [16]:

  I **Non-negative:** $s(x, y) \geq 0$.

  II **Symmetric:** $s(x, y) = s(y, x)$.

  III **For all $\boldsymbol{x, y \in X}$:** $s(x, y) \leq s(x, x)$, with equality if and only if $x = y$.

  Obtaining a distance (dissimilarity) d from a similarity s is by: $d = 1 - s$.

3.2.2.2 Distance- and Similarity-based Patient Similarity Metrics PSMs

Selecting the PSM for patient similarity analysis is an important decision. Deza *et al.* [16] state: "There are many similarities used in Data Analysis; the choice depends on the nature of data and is not an exact science". Hence, the selection of the distance or similarity metric does not depend on the application area (e.g., medicine), but relies on the data for which the metric is implemented.

The EMR data types are varied: structured, semi-structured or unstructured [36]. These different data formats affect the selection of the implementation approach (as will be discussed in the following chapter) and the selection of the distance metric.

In this thesis, structured EMR data is used. Furthermore, the structured data comes in different formats like numerical, binary, and categorical values. Moreover, the size of the data and the dimensions might affect the selection of the metric.

We made a comparison between some distance metrics on different factors by the K-nearest neighbors model. We consider the computational efficiency (i.e., the required time for computing a similarity or distance metric between the patients of the full dataset). Not all the metrics respond similarly with high dimensional data. Furthermore, the metric might have effects on the accuracy of the prediction model. Thus, we also consider this in comparison. Here we will define some distance and similarity metrics.

- **Cosine Similarity:**

  The cosine similarity metric is defined as shown in the following Equation 3.1.

$$cos(x, y) = \frac{x \cdot y}{\| x \|_2 \cdot \| y \|_2} . \tag{3.1}$$

  x and y are the vectors of two different patients $x \neq y$, $\cdot$ is the dot product and $\| \ \|_2$ is the Euclidean vector magnitude. Translating the dot product and the magnitude into calculations on the vector elements x_i and y_i results in the following Equation 3.2.

$$cos(x, y) = \frac{\sum_{i=1}^{m} x_i y_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \sqrt{\sum_{i=1}^{m} y_i^2}} . \tag{3.2}$$

  The patient data of n patients with m features are represented in an m-dimensional vector space: The patient vectors $N = P_1, P_2, ..., P_n$ each consist of features $x_1, x_2, ..., x_m$. Thus, m is the size of the feature set, while n is the size of the patient set. i ranges over all the features m. Cosine similarity between two patient vectors in the vector space is the cosine of the angle between them. It gives a result value between -1 and 1. Two patients are very similar if the result is 1 and are very different if the result yields -1 [50].

  Cosine distance is defined by $1 - cos\theta$. There are many implementations of cosine similarity in patient similarity analysis for different health prediction areas [50, 21, 52].

  Lee *et al.* [50] select cosine-similarity for the patient similarity metric. The continuous predictor values are normalized to the range between 0 and 1. Then, cosine similarity is used for computing the similarity between the continuous predictors in the patient vectors. However, cosine similarity is not used for the categorical variables.

  Garcelon *et al.* [21] compute the TF-IDF weight for each feature in the patient vector. The weight was the product of TF and IDF. TF is the number of feature occurrences for the patient divided by the total number of features. IDF is the logarithm of n divided by the number of patients who have this feature. Cosine similarity is used to measure the distance between these vectors.

- **Euclidean Distance:**

  Euclidean distance is one of the most common distance measures. In m-dimensional space, two vectors x and y the Euclidean distance between these two vectors [51] as shown in the following Equation 3.3.

$$d(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \ . \tag{3.3}$$

It is the squared distance of each dimension or feature (where i ranges over all the features m) of these two vectors and takes the square root of the sum of squared difference.

The Euclidean distance is used for patient similarity analysis, for instance, [66, 85, 32, 24]. Wang *et al.* [85] embed the patient vectors into an intrinsic space by the Local Spline Regression (LSR). After that, the Euclidean distance is used to measure the pairwise distance between patient vectors.

Hoogendoorn *et al.* [32] select the K-Nearest Neighbor (k-NN) for patients similarity. Euclidean distance is computed next to Keogh lower bounds and a penalty for the unmatched features.

- **Jaccard Distance:**

  Jaccard similarity is defined by Equation 3.4 from [16]:

$$s(x, y) = \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} y_i^2 - \sum_{i=1}^{m} x_i y_i} \ . \tag{3.4}$$

  It is determined by dividing the dot product of the two patient vectors by the subtraction of the sums of summation of their squared vectors from the dot product of the two patient vectors. Therefore, the corresponding Jaccard distance is as follows Equation 3.5:

$$\begin{aligned} d(x, y) &= 1 - \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} y_i^2 - \sum_{i=1}^{m} x_i y_i} \\ &= \frac{(\sum_{i=1}^{m} x_i - y_i)^2}{\sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} y_i^2 - \sum_{i=1}^{m} x_i y_i} \ . \end{aligned} \tag{3.5}$$

  Jaccard distance is one of the different patient similarity metrics Gottlieb *et al.* used [24]. For each feature of a patient vector, a similarity measurement is calculated. Jaccard distance is used to measure the similarity between patient ICD codes as follows: the co-occurrences of an ICD pair among all the patients are computed. For each pair, the Jaccard score is evaluated. The binary case of Jaccard is called Tanimoto similarity defined by Equation 3.6 :

$$s(X, Y) = \frac{\mid X \cap Y \mid}{\mid X \cup Y \mid} \ . \tag{3.6}$$

- **Mahalanobis Distance:**

  Mahalanobis Distance is defined as follows [16] Equation 3.7 :

$$\| x - y \|_A = \sqrt{(detA)^{\frac{1}{n}}(x-y)A^{-1}(x-y)^T} \, . \tag{3.7}$$

  Where x and y are the patient vectors, n is the size of the patient set. A is the covariance matrix.

  Sun *et al.* [76] develop Locally Supervised Metric Learning (LSML) that automatically adjusts the importance of the predictor variables. It is based on Mahalanobis distance.

  Panahiazar *et al.* [64] analyzed patient similarity through implementing supervised and unsupervised clustering approaches with Mahalanobis distance. Mahalanobis distance is used to measure the similarity between a queried patient and the patient clusters.

3.2.2.3 Computing Patient Similarity

Here is a description of how similarity or distance between patient vectors is computed. If we have n patients each with m features, they can be represented by a matrix in Table 3.1:

$VectorID$	$Feature_1$	$Feature_2$	$\cdots$	$Feature_m$
1	$value_1$	$value_2$	$\cdot$	$value_m$
2	$value_1$	$value_2$	$\cdot$	$value_m$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$value_1$	$value_2$	$\cdot$	$value_m$

Table 3.1: Patient and Feature Matrix

The distance between Patient vectors can be measured by the previous PSMs and can be visualized by the follows matrix of distances Figure 3.1:

$$
\begin{array}{c}
\\
1 \\
2 \\
3 \\
\vdots \\
n-1 \\
n
\end{array}
\begin{array}{c}
\begin{array}{cccccc}
1 & 2 & 3 & \cdots & n-1 & n
\end{array} \\
\begin{pmatrix}
0 & d_{1,2} & d_{1,3} & d_{1,..} & d_{1,n-1} & d_{1,n} \\
d_{2,1} & 0 & d_{2,3} & d_{2,..} & d_{2,n-1} & d_{2,n} \\
d_{3,1} & d_{3,2} & 0 & d_{3,..} & d_{3,n-1} & d_{3,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
d_{n-1,1} & d_{n-1,2} & d_{n-1,3} & d_{n-1,..} & 0 & d_{n-1,n} \\
d_{n,1} & d_{n,2} & d_{n,3} & d_{n,..} & d_{n,n-1} & 0
\end{pmatrix}
\end{array}
$$

Figure 3.1: Distance Matrix

As discussed previously in Section 3.2.2.1, the distance is symmetric (i.e. the distance between P_1 and P_2 is equal to the distance between P_2 and P_1 where $d_{1,2} = d_{2,1}$) see Figure 3.1. Therefore, the upper and lower triangular are symmetric. Moreover, the diagonal line in the matrix represents the elements of the distance between the patient vector with itself (i.e., reflexivity). In case of measuring the distance, they will have zeros; otherwise they will have ones in case of similarity.

For computing, the similarity or distance between patients, the upper, or the lower matrix part is enough. Thus, rather than $n \times n$ distances need to be computed, only $\frac{1}{2}n(n-1)$ distances or similarities need to be computed. However, this amount of computation becomes significant with big patient data.

For finding the similar patients to patient x, we should measure the distance between x and all the other patients. That is $n-1$ similarity or distance calculations. For classification by KNN model (in Section 4.6) and for clustering by K-means (in Section 6.3.2), all the pairwise distances are computed. In the KNN model, to find the nearest neighbors to patient x, the distance between patient x and all the other patients are calculated, then the top k nearest neighbors are selected. In K-means clustering, to assign a patient data point to its nearest cluster, all pairwise distances between the cluster centroids and the patient data point need to be calculated.

3.2.3 Approaches for Implementing The Predictive Model and Patient Similarity Analysis

The high adoption of electronic medical records (EMRs) increases the interest and the value of analyzing medical data. EMR data are usually extracted from different systems, then integrated and pre-processed in a dedicated tool, and afterward, analysis is executed [69]. Since the EMR data come from different sources with different formats, the extracted data could be structured, semi-structured, or unstructured [36]. Various analytic tools can be used to analyze EMR data and to build the predictive model. Wang *et al.* [86] state that depending on the data type and the analysis purpose, the big data analysis in health care can be divided into three components: Hadoop Map/Reduce, stream computing, and in-database analytics. The high-performance stream

data processing is to predict the likelihood of illegal events such as healthcare fraud detection. For analyzing the EMR data, we have Hadoop Map/Reduce (or in general data analysis tools and platforms) and in-database analytics.

Data Analysis tools and Platforms:

Data analytics platforms and tools such as Hadoop, Mahout, or R are one approach to use for implementation. In the previous Chapter 2, we mentioned some implementations by different tools. Current implementations for the predictive ML model and patient similarity analysis use different tools for data pre-processing and analysis. Most of the time, in this approach, the database system where the patient EMRs are stored is not used as a part of the analysis workload. The EMR data is extracted from the Database Management System (DBMS), then the analysis is processed outside the data warehouse. Lee *et al.* [50] use R language to develop patient similarity analysis and predictive models. All calculations and analyses were done with R, while the EMR data was obtained from an Oracle SQL Developer database using SQL. Gottlieb *et al.* [24] use Matlab to implement the patient similarity analysis. Works on optimizing patient similarity analysis for medical data like [5] commonly use the power of data analysis tools – but thereby paying the latency cost of accessing and processing the EMR data outside the (DBMS). Using external data mining tools for analysis and computation, data access, as well as pre-processing and transformation, causes several data management issues [62]. Here we did not consider the time we lose for exporting and transforming the data out the DBMS and accessing the output in different tools.

This approach poses some challenges. The open-source big data analytic tools (e.g., Hadoop) provide advances in analysis and scalability (when running in distributed systems) however pose challenges for healthcare [69]. They require high programming skills and technical support, which is uncommon to obtain in the healthcare end-users. Moreover, security and privacy are significant drawbacks. The mathematical packages to develop the predictive model as R and Matlab provides a variety of different ready to use algorithms. However, they do not scale to a large data set.

In this thesis, we use the RapidMiner platform. RapidMiner is a data science platform that supports data analysis and visualization tools [57, 58]. By RapidMiner, the time and effort in pre-processing the data in a second tool are saved. It provides the required data processing and ML algorithms for predictive model deployment. Furthermore, it supports database connection where we can access our data in our database without a need for exporting and no latency of outside access from the DBMS. Furthermore, for non-relational data RapidMiner provides text analysis. Through its Radoop extension, it can scale for Big data analysis. Hofmann *et al.* [31] give two examples of medical data mining by RapidMiner carpal tunnel syndrome and diabetes. Examples of using RapidMiner for analyzing medical data of ICU are Van *et al.* [83] and Parreco *et al.*

[67]. For this study, we use RapidMiner studio version 9.2 Educational edition [57]. It used as a tool for model development, training, testing, and performance evaluation. The computation takes place in Windows 10, Intel i5-7300U, CPU 2.70 GHz, RAM 32 GB, x64-based processor.

Database Management Systems:

Another approach for implementing EMRs analysis in specific patient similarity analysis is to use a DBMS. Either relational DBMSs or NoSQL databases are a valid choice. However, the type of extracted data decides the choice of the database system. The SQL database is well-suited for structured and relational data, whereas the NoSQL database is perfect for non-relational and unstructured data.

NoSQL databases (see [89] for a comprehensive survey) are used for analyzing unstructured data. For instance, Abdelrahman *et al.* [1] use the NoSQL database Neo4j for analyzing health care data. Some current works are applying patient similarity analysis on the unstructured health record data such as medical notes. Unstructured medical data usually have poor quality: "unstructured data is highly variable and all too often incorrect" [69]. Moreover, pre-processing the unstructured data for analyzing the similarity requires much effort. Extracting medical terms from unstructured data requires much effort and medical knowledge. Identifying and extracting the medical notes from examination reports is done manually by Chan *et al.* [9]. Wells *et al.* [87] mention different problems of analyzing unstructured medical data. Difficulties include grammatical errors, various interpretations of a specific phrase dependent on the content, and the acronyms and abbreviations.

Relational Databases have well-defined standards that assure full integrity and availability of data. In our case, the data of the EMRs are structured (e.g., diagnoses, laboratory values, and medications). Hence, we find the SQL database is the best fit. Furthermore, we argue that analyzing structured data is more accurate and does not require as much effort for pre-processing as the unstructured one. The column store DBMS is the fastest based on other authors' opinion: "Data mining researchers have also shown that SQL on a parallel, columnar database could be a candidate for Big Data analytics" [75].

In-database approaches have many advantages. They eliminate the cost of pre-processing the data and analysis in different tools and avoid data management problems. Ordonez [63, 87] discuss many benefits of performing data analysis inside a DBMS, but the main one is avoiding the data export bottleneck from the data warehouse. Wells *et al.* [87] discuss many pros and cons of in-database health care data analysis. On the other hand, some weaknesses of an in-database approach might be that the data format and query language are limited to a specific range. Moreover, based on the best of our knowledge, there is no current implementation of predictive health model on in-database.

Summary

Developing a solution for analyzing structured data might not be valid to be applied to unstructured ones and vice versa. Nevertheless, Johnson *et al.* [37] state that it is nontrivial integrating data from different medical devices into a single data management system. The reason lies in the lack of standardization among the medical devices and the various data formats. Furthermore, as already mentioned, the selection of different data analytics depends on the data type and the analysis purpose [86].

Thus, in-database analysis cannot be applied to all of these heterogeneous and various data types. Implanting predictive analysis in DBMS is for a specific data type. SQL DBMS is for the structured data type, and NO-SQL DBMS is for the un-structured one. The data analysis platform Hadoop is scalable for big data and supports parallel analysis of both unstructured and structured data. However, it can be challenging in healthcare data analysis because of a lack of technical support and its required high programming skills, which is not available in typical healthcare user [69].

RapidMiner supports analysis for the two data types structured and unstructured. Furthermore, RapidMiner is identified as leader in advanced analytics platforms by Gartner Magic Quadrant for Data Science and Machine Learning Platforms for the sixth year in a row because of its highest score for the Ability to Execute. It doesn't require high programming skills. Thus, it would be easy for medical staff with basic programming skills to optimize patient care. Based on a review on Gartner peer insights by a director of data research and analytics in the healthcare industry: "Easy to use data science tool, Straight forward tool with good functionality" [57]. Moreover, the performance bottleneck of exporting the dataset outside a DBMS and pre-processing and analysis in a different tool is eliminated. All the workload can be carried out inside RapidMiner.

3.3 Dataset

For health prediction purposes, various medical data has to be extracted and analyzed. The selection of the medical measurements is based on the intention of the prediction. For instance, for diagnoses prediction, the feature selection depends on the disease we are looking to diagnose. In this paper, we use the real-world critical care database Medical Information Mart for Intensive Care (MIMIC) [40]. The data is collected from patients admitted to critical care units at the Beth Israel Deaconess Medical Center in Boston, Massachusetts in June 2001 to October 2012. It is a publicly available, widely used, and de-identified dataset. We use the latest version of MIMIC, which is MIMIC-III. MIMIC-III comprises over 61,000 hospital admissions to critical care units of 53,423 adult admissions and 7870 neonate admissions with thousands of medical data.

MIMIC-III was collected from different sources: archives from critical care

information systems, hospital electronic health record databases, and Social Security Administration Death Master File. From the archives of two critical care information systems (CareVue and MetaVision), the clinical data of the critical care were collected. These data include the time-stamped hourly collected physiological measurements as heart rate and other notes and medication data. From the hospital and laboratory health record databases, these data were collected: demographic data and in-hospital mortality, laboratory results, discharge report, and billing information. Finally, from Social Security Administration Death Master File, the dates of out-of-hospital mortality were obtained.

3.3.1 Selection of the Predictor Variables

In our study, we are interested in predicting the risk of mortality for the adult patient (aged 15 years or above). Thus, only the data of the adult patient admissions to the different critical care units are extracted. The data of the neonate admissions are not included. Our medical measurements (i.e., the predictor variables) selection is inspired by Lee *et al.* [50]. Time series sequences were obtained. Some of the predictor variables are sampled every 6 hours to produce time-series features. In total, there are 74 features from the first 24 hours in the ICU stay (see Table 3.3). Furthermore, age, gender, and ICD code were also extracted.

The value that we want the model to predict is if the patient has a risk of in-hospital mortality. Thus, the value of the in-hospital mortality flag is also extracted from MIMIC for test purposes. As a result, these extracted data can help us build the model to predict the risk of in-hospital mortality after the first 24 hours of ICU stay.

3.3.2 Data Pre-processing Transformation and Normalization

Next, I discuss the data normalization process. Scaling of variables is one of the aspects you come across when applying distance metrics since distance depends on scale. Data with different units have to be scaled for implementing the distance metrics. Moreover, variables with different ranges will have different weights in the distance metric. The enormous range variable has the most weight to the distance metric. Hence, scaling the variable range gives every variable equal weight and make all the variable equally contributed to the distance metric.

Normalization transforms the data into a scale of smaller range. Normalization by scaling the values to a specific range is commonly applied [50, 32] to ensure equal contribution of all predictor variables to the PSM calculation. In our case, we treat all the predictor variables in the dataset to have an equal impact on the prediction and the similarity (i.e., no such a predictor variable

Predictor variables	Feature extracted	Time window
Vital signs (heart rate, mean blood pressure, systolic blood pressure, Spo2, body temperature, and spontaneous respiratory rate)	Min and Max	From each non-overlapping 6-hour period during the first 24 hours
Lab variables (blood urea nitrogen, hematocrit, white blood cell count, serum glucose, serum HCO3, serum potassium, serum sodium, and serum creatinine.)	Min and Max	From the first 24 hours
Categorical variables (use of mechanical ventilation, receipt of vasopressor therapy)	Binary	From the first 24 hours
Glasgow Coma Scale	Min	From each non-overlapping 6-hour period during the first 24 hours
Urinary output	Sum	From each non-overlapping 6-hour period during the first 24 hours

Table 3.3: The Selected Predictor Variables and the Extracted Features

has more impact on the similarity or the prediction). However, they have different units and ranges. The continuous numerical predictors, such as the vital signs and lab test results, were normalized into the range $[0, 1]$. The result value of our used PSM is in the range -1 (denoting the minimum similarity) and 1 (denoting the maximum similarity).

We use the min-max normalization method [27] as shown in Equation (3.8).

$$v_i'' = \frac{v_i - \min_x}{\max_x - \min_x} \cdot (\max_{\text{new}} - \min_{\text{new}}) + \min_{\text{new}} \tag{3.8}$$

In this equation, x is a numeric predictor variable with m observed feature values $v_1, v_2, ..., v_m$. $\max_x$ and $\min_x$ are the minimum and maximum values of the predictor variable x. Hence, the normalization method in Equation (3.8) maps a value v_i of x to v_i'' within the new range; we want to scale to the new range $\max_{\text{new}}$ and $\min_{\text{new}}$, which are 1 and 0 respectively. However, the relation among the original data values of x are preserved. Replacing, $\max_{\text{new}}$ and $\min_{\text{new}}$ with their values 1 and 0 gives the following Equation (3.9).

$$v_i'' = \frac{v_i - \min_x}{\max_x - \min_x} \ . \tag{3.9}$$

Another alternative approach for normalizing the data for the distance metric is the re-weighting of the variables. Some implementations of this approach in medical prediction are [59, 21].

3.4 Evaluating the Predictive Performance

To evaluate the performance, we select different parameters with the models besides the different selected features. Test dataset is used to test how generalizable our model is. For an imbalanced dataset, we should use other metrics besides accuracy as recall and precision.

3.4.1 Accuracy Metrics

The model performance can be measured by different accuracy metrics such as accuracy, recall, precision, F-measure, AUC. We will explain some of these metrics and what they mean in our study content. In our classifier, we have two classes to predict: the positive class (the class of interest, i.e., the patients with risk of mortality) and the negative class (the survived patients). First, we need to know the output of our classifier that can be described by the Confusion Matrix, which are True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). A TP is a patient who is truly predicted to be in risk, FP is a patient who is incorrectly predicted to be in risk, TN is a patient who is correctly predicted as survived, and FN a patient that is incorrectly predicted as survived.

- **Accuracy:** is the ratio of the total true predictions (for both the patients with mortality risk (TP) and survived patients (TN)) to the all predictions made by the model (see Equation 3.10):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \ . \tag{3.10}$$

- **Recall:** is also called the True Positive Rate (TPR) and the Sensitivity. It is a fraction of the patients that are correctly predicted with risk over all the patients who have the risk. It measures the efficiency of the model of predicting the entire group of patients with a risk of mortality. This formula in Equation 3.11 calculates it:

$$Recall = \frac{TP}{TP + FN} \ . \tag{3.11}$$

- **Precision:** for our model, it is the fraction of the patients who are truly predicted with mortality risk over all the patients that are predicted with mortality risk. The higher the precision is, the lower the number of incorrectly predicted patients with risk. This formula in Equation 3.12 calculates it:

$$Precision = \frac{TP}{TP + FP} \ . \tag{3.12}$$

- **F-Measure:** is the f1 score that combines the Precision and the Recall to give an average of them. As a result, this metric measures the efficiency of the predictive model for predicting both the patient with mortality risk and without risk (see Equation 3.13):

$$F1 \ score = \ 2 \cdot \ \frac{(Precision \cdot Recall)}{(Precision + Recall)} \ . \tag{3.13}$$

A trade-off occurs between accuracy metrics such as between Recall and Precision. Thus, we should consider which of the evaluation metric is the most worth for our predictive model. We mostly care to correctly predict patients with mortality risk (i.e., TP). Moreover, a low number of FNs is more crucial than a low number of FPs. A high Recall has a high priority for our system.

3.4.2 Visualize the Performance by Curves

To measure the performance of a classifier, we can use the graphical plot Receiver Operator Characteristic Curve (ROC) [17]. The ROC graph is a two-dimensional graph in which the X-axis is False Positive Rate (FPR), and the Y-axis is the True Positive Rate (TPR). The FPR is the ratio of negatives incorrectly classified as positives (FP) to the total negatives regardless of classification. The FPR is calculated by the ratio FPR= FP/FP+TN. The TPR is the ratio of the positives correctly classified to the total positives. The TPR is defined in the previous section 3.4.1 by the recall. First, the Receiver Operator Characteristic Curve (ROC) graph shows the model performance at all classification thresholds by plotting the relation between TPR and FPR. Then, the Area Under the ROC Curve (AUC) aggregates the performance among all the classification thresholds. Precision Recall Curve (PRC) [14].

In PR space, one plot Recall on the x-axis and Precision on the y-axis. The recall is the same as TPR, whereas Precision is the fraction of correct positive predictions among all positively predicted examples. From this curve we can calculate the Area Under the PRC (AUPRC).

The larger the value of AUC and AUPRC, the more accurate the model performance is. For the imbalanced dataset, the good metrics are AUC and F-measure. In evaluating models for mortality risk prediction, maximizing the AUC is the goal that researchers are typically seeking to [39]. Thus, in this thesis, AUC is the main accuracy metric that we will try to maximize.

3.5 Performance Improvements

In the Section 4.7, we find the poor performance our models has on predicting the risk of death. Even though the accuracy was high, the models fail in predicting the risk (low Recall) — the reason laid on a data problem (imbalanced class distribution). Therefore, we should improve the predictive performance of the models for accurate mortality prediction. Brink *et al.* [8] mentioned three techniques for achieving better model accuracy: tuning the model parameters, selecting a subset of features, and pre-processing the data. In the following Subsections, we will define them and discuss their different methods. To improve the predictive performance, we test the approaches of performance improvements in the next Chapters. In the end, we will combine the optimal parameter values and the outperformed methods from these performance optimization factors. This combination helps us to find the best setting for our models and data for improving predictive performance.

3.5.1 Tuning the Model Parameters

Each of the used Machine Learning models is configured by specific parameters. These tuning parameters control how the algorithm uses training data to build a

model. There are no standard best values of these model parameters. In general, the optimal value of these parameters is entirely dependent on the type, and the structure of the used dataset and on the problem needs to be solved. The parameter values impact the predictive performance of the model. Thus, we should do cautiously selecting.

We discussed the crucial parameters of each of the selected ML models in Chapter 4. We find the effect of those parameters on the model performance. Furthermore, we implement a Grid search to find the optimal values of the parameters for our used case. In the following techniques of performance optimization, we will use these values.

3.5.2 Selecting a Subset of Features

In the big data age, it is common that the used dataset for prediction is a high dimensional data. A large number of features might include a noise that causes difficult knowledge discovery and hard to find the important feature that helps in prediction. The high dimensional dataset not only slows the training process but also makes finding the optimal solution harder. This problem is referred to *curse of dimensionality*. Thus, further feature selection after the first feature extraction step plays an important role. It is not obvious to know the effect of the features on the model. The feature that contributes most to the prediction label is the one we need. Therefore, we should carefully search for the features that build the most general and accurate model.

In Chapter 4, we process the model training on the original dataset with the complete set of features (i.e., the initial extracted set). It takes a good time for training, and the predictive accuracy was not good. Thus, we have made further feature selection to find the optimal set to improve the predictive accuracy (for the result, see Chapter 5). In the following Subsections, we will present three approaches for feature selection to select a subset of features from the initial set to build a good predictor. There are three main approaches for feature selection; wrapper, filter, and embedded approaches. Each of these approaches has different methods with pros and cons that will be discussed in the following.

3.5.2.1 Feature Selection by Wrapper Approach

The wrapper method wraps a ML model inside it as a black box to evaluate a subset of the features. The ML model is trained on the feature subset, and it scores them according to their predictor power [65, 42]. There are two methods of the wrapper approaches which are backward elimination and forward selection.

The backward elimination is searching for the features by beginning with the full set of features and removing one feature at a time. With each deletion of the feature, the model performance is evaluated. The feature that gives the lowest

performance decrease will be deleted. This process continues until a decrease in model performance occurs.

The forward selection begins with an empty set of features and then adds one feature at a time. With each addition of a feature, the performance is estimated by cross-validation. Only the feature that by its inclusion gives a high-performance improvement is added to the selected list. It adds the feature that gives the highest increase of the model performance. The iteration is stopped when no increase in the model performance occurs.

There are advantages and disadvantages of these methods. Backward elimination is extremely expensive but can easily find the interacting features. On the other hand, the forward selection is a computational advantage since building a classifier model with few features in the data is much faster [42]. With a large size of features, the backward elimination long run time makes it in-feasible for practice.

3.5.2.2 Feature Selection by Filter Approach

Unlike the wrapper approach, the filter method selects the features independently of the learning machine [25]. It selects the features based on a statistical score about their correlation to the predicted value. The features are filtered based on a metric we defined.

CHI-squared is one metric of the filter-based feature selection method. We calculate the chi-square metric between each feature and the target (i.e., the value we want to predict). If the relationship between the feature and the target variable is independent, then the feature is not important; if it is dependent, then the chi-square is high, and the feature is very important. Then, we only select the features with the highest chi-squared values.

3.5.2.3 Feature Selection by Embedded Methods

There are some ML algorithms that have built-in feature selection methods. Embedded methods use ML algorithms that have integrated feature selection methods.

In this thesis, the used ML model, the GBDT has a built-in feature selection method. Furthermore, a great advantage we have after the GBDT constructed is the earning of a list of feature importance (i.e., feature importance estimation). GBDT assign importance score for features. The feature importance score indicates the usefulness (i.e., the more use) of the feature in making a critical decision with the decision trees (this feature is essential for building the boosted tree model).

The relative importance or weight of features is calculated by considering the influence of features in splinting to improve the squared error. Relative influences I_j of individual features x_j in a tree T approximated by a surrogate measure in Equation 3.14 from [18]

$$\widehat{I}_j^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 1(\upsilon_t = j) \tag{3.14}$$

Where t is non-terminal nodes of J terminal node in T tree. The υ_t is the splitting variable associated with node t, while $\hat{\imath}_t^2$ is the corresponding improvement in squared error resulted from the split.

The relative influence of a feature for the set of the decision trees in GBDT $\{T_m\}_1^M$ is the average over all the trees (see Equation 3.15) from [18]

$$\widehat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \widehat{I}_j^2(T_m) \tag{3.15}$$

The feature weight scales from zero to infinity. The assigned weight of zero indicates the no importance of this feature since it was not used for any split in the trees. The larger the weight is, the more important the feature. However, in Table 5.2 we also provide the top-20 of the scaled weight from 0 to 1.

We will use the GBDT model to select a feature subset that has the highest importance score. The top-20 features are listed in Table 5.2.

3.5.3 Pre-processing the Data

Real-World datasets usually are not ready for directly applying Machine Learning models. They have to be pre-processed and cleaned. The problems that a dataset contains affect the performance of the ML models, for instance, missing values. In our dataset, we remove all the missing values. Furthermore, imbalanced class distribution causes the model to be biased toward the majority class, which results in low predictive accuracy. This is what we face when we use the imbalanced MIMIC-III dataset for evaluating our classifier models(see Section 4.7). An overview of imbalanced classes problem and handling it with data sampling approaches are in Chapter 6.

Using normalized or un-normalized data is a decision to take when pre-processing the data. Some ML models require data normalization, and others are not requiring. In this thesis, we compare different ML models, where some models require data normalization. We normalized our dataset to range from 0 and 1. Brink *et al.* [8] state: it is better to let the ML algorithm figure out the relative weights of features rather than forcing particular feature weight. Different feature value range will have different weight. Therefore, normalizing features to a specific range makes sure all features are considered equally. However, we also test the un-normalized data to find the best for optimizing the predictive performance. The comparison between the using normalized or un-normalized dataset is in Chapter 5.

4

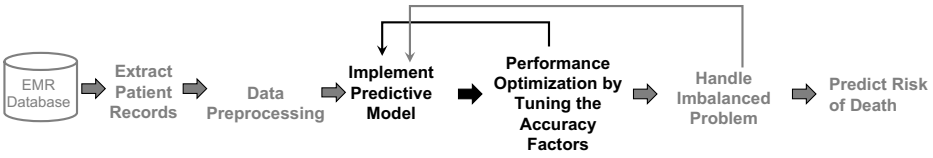
The Predictive Model

The following chapter lays the different Machine Learning (ML) models for health prediction driven by EMR. First, it gives a comparison between 7 ML models to select the initial group of models. Then, for the initial models' group, it provides the algorithm behind the models. The important parameters for each model are represented. These parameters have a main effect on model performance. Thus, the tuning of these parameters and their performance effects are discussed. In tuning the parameters of the models, the optimal value of the parameters is found using a grid search with cross-validation. Furthermore, the models' strengths and weaknesses are provided. In the end, a detailed comparison of the models is represented to select the optimal ML model.

Contents

4.1 Scope of the Chapter

In this chapter, we will address the step of implementing the predictive model and selecting the optimal model for predicting death's risk. All the essential details of implementation are represented. Tuning the model's parameters is also discussed in particular, which is one of the integral accuracy factors of the next step "performance optimization by tuning the accuracy factors".



4.2 Comparing Models Performance

Several ML models can be used for predicting the risk of death. To select the initial group of models for further tests, then to find the optimal one, we compare the performance of 7 models. We compare the performance of the ensemble ML model (the GBDT), and some of the commonly used models in predicting the risk of mortality in our dataset. The models we compare are Logistic Regression (LR), Decision Tree (DT), K-nearest neighbor (KNN), Naïve Bayes (NB), Support Vector Machine (SVM), and Random Forests. We set the models' parameters to the default values in RapidMiner. We test the models by 10-fold cross-validation and compare their performance in predicting the risk of mortality in our dataset (see Figure 4.1).

The value of AUC =0.5 indicates a random guesses predictor, while the higher value of the AUC indicates better model discrimination. We select the models that have an excellent performance in Figure 4.1 for further analysis. The models that had the highest AUC values are GBDT, LR, Naïve Bayes, and KNN. However, even though Naïve Bayes gives a good AUC of 0.729 (from Figure 4.1), its accuracy was low 29.77%. Thus, we do not include it for further tests. We include DT instead of Naïve Bayes, for further comparison to the best model GBDT that based on DT. In the following sections, we will define the selected models and tune their parameters to get the best performance out of them. Because of the imbalanced data and accuracy paradox problems, we should implement parameter optimization by grid search with optimizing the AUC, and not rely on the accuracy. The main performance criteria in cross-validation while testing the model is set to AUC. However, for all the models, we compare optimal parameter selection by optimizing both criteria (i.e., accuracy and AUC). In the last section (Section 4.7), we will compare the best-obtained models' performance in detail with different metrics. As a result, we will find the best model for our intended prediction.

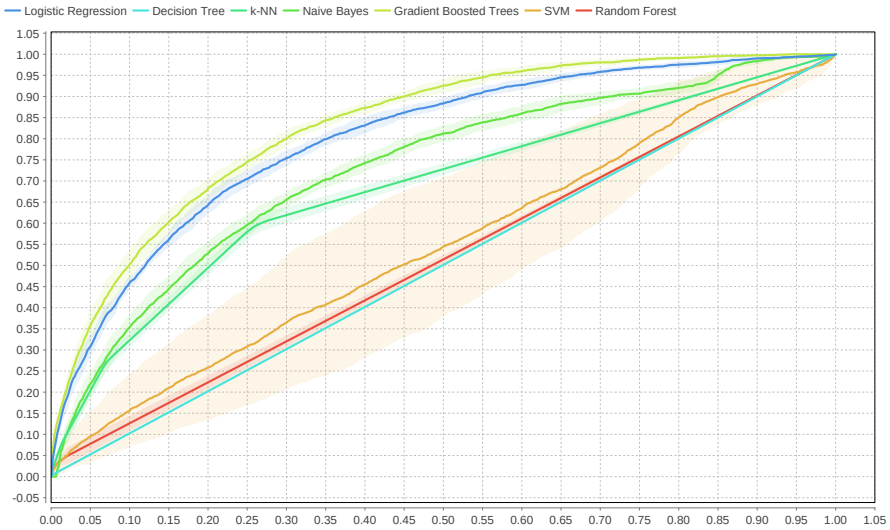


Figure 4.1: Compare the AUC Performance of Different ML Models

4.3 Logistic Regression

Regression models are used for classification. Logistic Regression LR will be described here as it is described by Aurélien [22]. It estimates the probability of an instance that belongs to a particular class. The instance belongs to the positive class when the probability is greater than 50%; otherwise, it belongs to the negative one.

Logistic Regression is based on the Linear Regression model Equation 4.1. It simply computes the weighted sum of the input plus the bias term θ_0. The probability value is $\hat{p}$, n is the feature size, $x_1, x_2, ..., x_n$ are the features and $\theta_1, \theta_2, ..., \theta_n$ are the feature weights.

$$\hat{p} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \ . \tag{4.1}$$

This can be represented by factorized form as the following Equation 4.2 which represents LR model estimated probability. (h_θ) is the hypothesis function that uses the model parameter vector (θ) which includes the bias term (θ_0) and the feature weights $(\theta_1, \theta_2, ..., \theta_n)$. x is the feature vector that contains $(x_1, x_2, ..., x_n)$ where (x_0) always equal to 1. The transpose of (θ) is (θ^T). Unlike to Linear Regression where is only the dot product of $(\theta^T.x)$, the LR gives the logistic of this result $(\sigma(\theta^T.x))$.

$$\hat{p} = h_\theta(x) = \sigma(\theta^T.x) \ . \tag{4.2}$$

Logistic (or logit) is noted by ($\sigma(.)$) which is a sigmoid function. The result of the logistic function is a number between 0 and 1. It defines by Equation 4.3.

$$\sigma(t) = \frac{1}{1 + e^{(-t)}} \ . \tag{4.3}$$

After estimating the probability $\hat{p}$, the prediction value $\hat{y}$ is found as follows:

$$\hat{y} = \begin{cases} 0 \text{ if } \hat{p} < 0.5 & , \\ 1 \text{ if } \hat{p} \geq 0.5 & . \end{cases}$$

That ($\hat{y}$) is 0 when ($\hat{p}$) is less than 0.5 and is 1 when ($\hat{p}$) is greater than or equal to 0.5. This because ($\sigma(t) < 0.5$) when ($t < 0$), and ($\sigma(t) \geq 0.5$) when ($t \geq 0$). Therefore, an instance belongs to the positive class when the predicted value ($\hat{y}$) is 1 and ($\theta^T.x$) is positive, while it belongs to the negative class when the predicted value ($\hat{y}$) is 0 and ($\theta^T.x$) is negative.

Important Parameters

The weight or coefficients and intercept parameters are learned by the model. The main parameter in the linear model is the regularization parameter. Assigning a small value to the regularization parameter develops a simple model. Logistic Regression can be prone to over-fitting with high dimensional features. Regularization gives a more reasonable decision boundary that prevents over-fitting. To use regularization with Logistic Regression, we need to add this term in Equation 4.4:

$$\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \ . \tag{4.4}$$

It penalizing the parameters ($\theta_1, \theta_2, ..., \theta_n$) from being to large. Applying regularization and keeping the parameters small when fitting high order polynomial with a large number of parameters will prevent over-fitting by giving a more reasonable decision boundary for separating the positive and negative samples.

Tuning LR Parameters

Using regularization increases the bias of a model. This can help in case the model over-fitting the training data (i.e., high variance). However, high bias will cause under-fitting, which produces a poor performance for the training and test dataset. Regularization adds a new parameter to LR, which is lambda. For making a decision on using regularization or not and for finding the optimal lambda value, we evaluate the accuracy and AUC to find which provides a higher prediction performance.

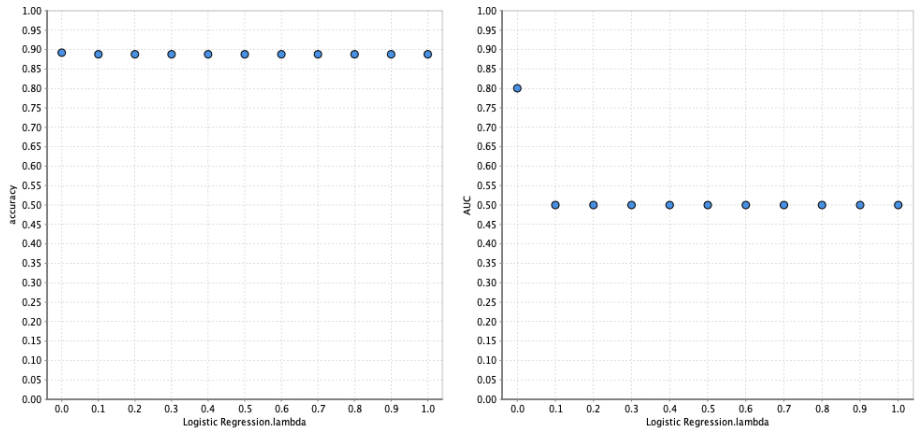


Figure 4.2: Testing Different Lambda of the LR to Optimize Accuracy and to Optimize AUC

- **main criterion Accuracy:** We test LR with regularization and without while setting the main performance measure to accuracy. Using regularization gives an accuracy of 0.888, and not using regularization gives an accuracy of 0.892. The results are close to each other hence a decision is not obvious. Thus, we search for the best fit lambda value. lambda of 0 gives the highest accuracy 89.23 % +/− 0.28% and AUC 0.801 +/− 0.011. Figure 4.2 shows the result. Lambda of 0 means no regularization is applied. Therefore, no regularization is the decision. However, the accuracy measures of the other lambda values are almost the same at around 0.888.
- **main criterion AUC:** We test the LR's regularization parameter with AUC as the main performance criterion. Using regularization gives AUC of 0.500 and not using regularization gives AUC of 0.801. Unlike the accuracy results, the AUC values for both decisions were way different, and the decision is easier to take. It is clear that not using regularization is the right way to go. All the lambda values give AUC of 0.500 and only lambda of 0 gives AUC of 0.801 (see Figure 4.2).

Strength and Weaknesses

In general, linear models are fast in training and in predicting. LR performs well with sparse data, unlike KNN. Moreover, it is a best choice for large datasets because of its scalability for very large datasets (i.e., large sample size or a large number of features). "In particular, it performs well when the number of features is large compared to the number of samples" [60]. However, it is not the best choice for low dimensional space. It is easy to understand how the prediction is made by the formula represented by Equation 4.3.

4.4 Decision Tree

Decision Tree DT is used for classification and regression tasks. The DT develops a decision by learning a sequence of if/else questions [60].

Important Parameters

The parameters that affect the performance are splitting criterion, max depth of the tree, and minimum samples needed to make a split [8].

- **Splitting Criterion:** This parameter is the criterion value which decides the selection of the attributes for splitting. The split value has to optimize the selected criterion. The splitting criterion can have one of these values information gain, gain ratio, and Gini index [70].
  - **Information gain:** Information gain relies on splitting the dataset on an attribute that decreases the entropy (i.e., values similarity or homogeneity). For splitting in a DT, we need to have the highest information gain. Information gain of splitting a training set S on a feature a_i is calculated in Equation4.5:

$$InformationGain(a_i, S) = Entropy(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{\mid \sigma_{a_i = v_{i,j}} S \mid}{\mid S \mid} \cdot Entropy(y, \sigma_{a_i = v_{i,j}} S) \tag{4.5}$$

    Where $\mid \sigma_{a_i = v_{i,j}} S \mid$ is subset of S with $a_i = v_{i,j}$. Summing up the fractions of the result of partitioning S by values $v_{i,j}$ from the attribute a_i produces the weighted average. The Entropy of the target feature y that is resulted from splitting by attribute a_i is calculated as follows Equation 4.6:

$$Entropy(y, S) = - \sum_{c_j \in dom(y)} \frac{\mid \sigma_{y = c_j} S \mid}{\mid S \mid} \cdot \log_2 \frac{\mid \sigma_{y = c_j} S \mid}{\mid S \mid} \tag{4.6}$$

    Where $\frac{|\sigma_{y=c_j} S|}{|S|}$ is the fraction of data examples from a class label c_j from the target feature y.
  - **Gain ratio:** The gain ratio is the normalized value of the information gain see Equation 4.7:

$$GainRatio(a_i, S) = \frac{InformationGain(a_i, S)}{Entropy(a_i, S)} \tag{4.7}$$

– **Gini index:** Gini index measures the differences between the probability distributions of the target attribute's values (see Equation 4.8):

$$Gini(y, S) = 1 - \sum_{c_j \in dom(y)} \left(\frac{\mid \sigma_{y=c_j} S \mid}{\mid S \mid} \right)^2 \tag{4.8}$$

- **Max Depth of Tree:** It indicates the maximum depth of the tree. The deeper the tree is, the complex the model to be and subject to the overfitting.
- **Minimum Samples Needed to Make a Split:** It indicates the minimum required number of samples to do a split in a node.

Tuning DT Parameters

We test tuning the parameters that affect the performance: splitting criterion, max depth of tree, and minimum samples needed to make a split. We find the optimal parameters to optimize accuracy and AUC. For this test, we use 10 fold cross-validation on the dataset of 32.635 patients.

- **main criterion Accuracy:** First, we will search for the optimal parameters to optimize accuracy. We test the three splitting criterion approaches: gain ratio, information gain, and the gini index. The max depth of the tree was the same 20. The result is represented by Figure 4.3. As a result, we find that the gain ratio gives the highest accuracy among the other splitting criterion, accuracy of 88.76%.

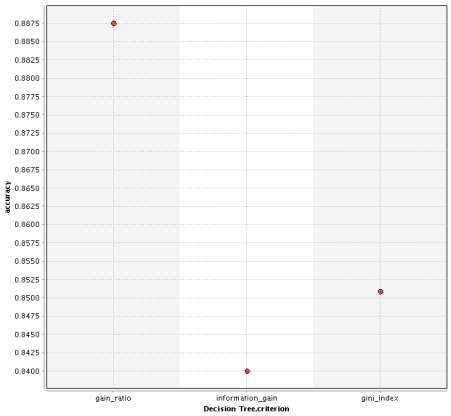


Figure 4.3: Testing Different Splitting Criterion of the DT to Optimize Accuracy

We test the max depth of tree from a range of 1 until 100 with 10 steps linear scale. The splitting criterion was set to gain ratio. Max depth of

41 has the highest accuracy of the DT. Max depth of 41 gives accuracy of 88.80% as shown in Figure 4.4.

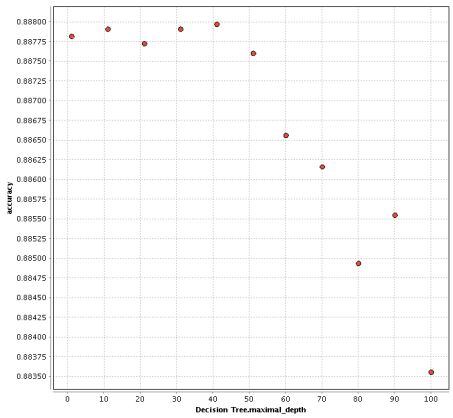


Figure 4.4: Testing Different Max Depth of the DT to Optimize Accuracy

Finally, the minimum samples needed to make a split is also tested. The range of 1 to 100 with 10 steps in linear scale is used. The splitting criterion was set to gain ratio. We set the max depth to 41, which is the optimal depth we found in the previous test. The minimum size of 11 produce the highest accuracy of 88.84% see Figure 4.5

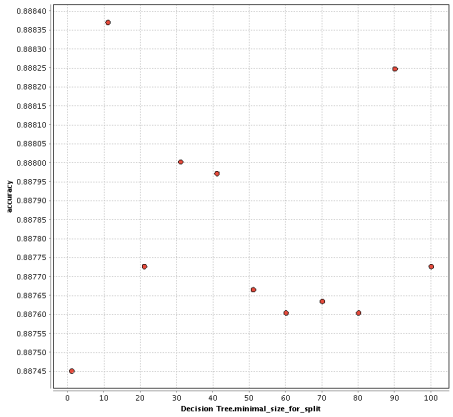


Figure 4.5: Testing Different Minimum Size for Splitting of the DT to Optimize Accuracy

- **main criterion AUC:** Second, we will search for the optimal parameters to optimize AUC. We test the splitting criterion parameter, where we set the max depth to 20 and the others to default. From the three splitting criterion approaches, the information gain gives the highest AUC of 0.601

+/− 0.016 (see Figure 4.6). The interesting result is that gain ratio gives the lowest AUC while it gives the highest accuracy. This is because of the imbalanced dataset we use and the accuracy paradox problem that we will discuss in Section 4.7. The high accuracy we got from the splitting criterion "gain ratio" is a fake accuracy because of the imbalanced dataset, which does not necessarily assure a high AUC.

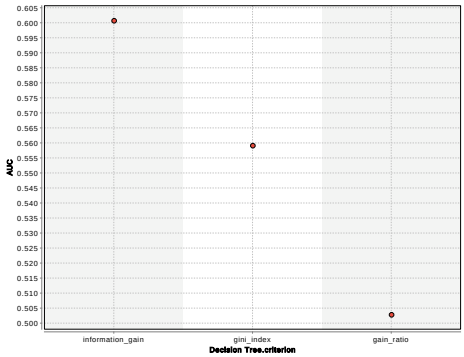


Figure 4.6: Testing Different Splitting Criterion of the DT to Optimize AUC

We test the max depth of the tree from a range of 1 until 100 with ten steps linear scale. The splitting criterion was set to information gain and the other parameters to default values. Max depth of 11 gives the highest AUC of 0.719 +/− 0.019 (see Figure 4.7). Moreover, it optimizes the other metrics in comparison to optimizing accuracy. For instance, Recall 17.78% but in cost lowering precision 41.31%.

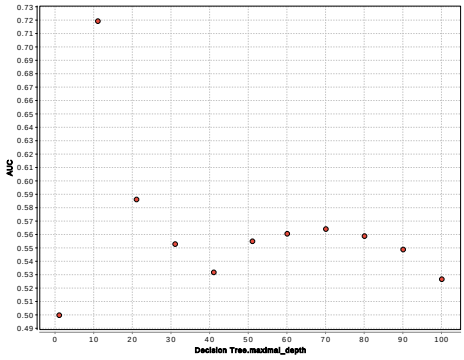


Figure 4.7: Testing Different Max Depth of the DT to Optimize AUC

At the end, the minimum samples needed to make a split is tested. The range of 1 to 100 with 10 steps in a linear scale is used. We set the splitting criterion and the max depth to the optimal values we find in the

previous tests and the others to the default values. From the result in Figure 4.8, the minimum size for split 100 is the best AUC by 0.756 +/− 0.015. Recall 16.36% and precision 45.07%, accuracy 88.35%. Since the best value is the highest value (i.e., 100) in the selected value range then testing higher values might improve AUC even more.

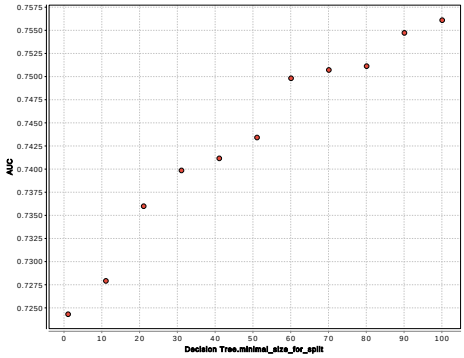


Figure 4.8: Testing Different Minimum Size for Splitting of the DT to Optimize AUC

Strength and Weaknesses

Kotsiantis [43] gives a review of the DT algorithm. The advantages of DT are that it needs simple data preparation and does not require feature scaling or centering. Moreover, it is considered to be a white box model rather than a black box model as random forest and neural network models [22]. Its decisions are easy to interpret (i.e., easy to know which rules and calculations contributed to such a prediction).

4.5 Gradient Boosting Decision Tree

First of all, boosting refers to hypothesis boosting that is "any ensemble method that can combine several weak learners into a strong learner"[22]. The popular boosting methods are AdaBoost (short term of Adaptive Boosting) and Gradient Boosting. AdaBoost sequentially builds better predictor based on the previous one by adjusting (i.e., increasing) the weight of the misclassified training instances. Similar to AdaBoost, Gradient Boosting in every iteration builds a better predictor by correcting its predecessor. However, it is not based on updating the weight of the instances. It is based on fitting to the *residual errors* of the previous predictor.

Multiple Decision Trees are ensembled to produce a more powerful model. Two ensemble models that are built out of multiple decision trees are Random Forest and Gradient Boosting Decision Tree GBDT.

Important Parameters

The more complex the algorithm is, the more numerous the tuning parameters are. In our case, the GBDT model is the one that has the largest number of tuning parameters.

The important parameters of GBDT are the number of trees, max depth of the tree, learning rate, splitting criterion, and minimum samples needed to make a split [8].

Because the GBDT is built out of ensemble DT, the parameters are mainly similar. Similarly to the DT, the GBDT has the parameters splitting criterion, max depth of the tree, and minimum samples needed to do a split. However, it has other parameters, which are the main parameters, the number of trees, and the learning rate. The learning rate is the degree of mistakes correction that each tree is allowed to do of the previous trees. These two main parameters have an inverse relationship. The lower the learning rate, the larger the number of learning trees needed to build a model. We should pay attention to the model complexity and overfitting that may be caused by increasing the number of trees.

The maximum depth is usually set to very low to reduce the complexity of each tree, often no deeper than five splits.

Tuning GBDT Parameters

We search for the optimal values of the parameters by grid research. We test both the optimal values to optimize the accuracy metric and the AUC metric.

- **main criterion Accuracy:** First, we will find the optimal values to optimize accuracy. We will test the main parameters: the number of trees and the learning rate. We test the number of learning trees with the range between 1 and 200 with ten steps. The other parameters are set to be the same for all the 10 folds cross-validation iterations. The learning rate was 0.1, and the maximum depth of the tree was 5. Figure 4.9 shows the accuracy of different number of learning trees where 180 is the best accuracy of 88.9 % +/− 0.41 %. It gives AUC of 0.863. However, we have to consider the low learning rate we set in this test. The low learning rate causes the need for more learning trees.

  Then we test the learning rate, which ranges from 0.1 to 1.0 with 10 steps. From the previous result Figure 4.9, we find that the accuracy values of learning trees (100, 120, 140, and 160) were close to the one of 180. After 180, the accuracy was getting lower as by 200. Therefore, we select 100 as the number of learning trees. The lower the learning rate is, the more computational time and the more iterations are required. We find that learning rate of 0.19 gives the best accuracy 88.78 % +/− 0.38% and AUC 0.858 +/− 0.006 as shown by Figure 4.10.

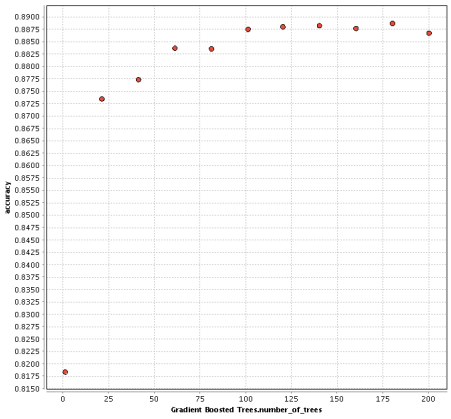


Figure 4.9: Testing Different Number of Learning Trees of the GBDT to Optimize Accuracy

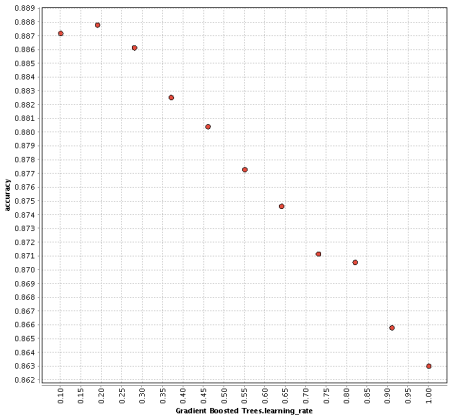


Figure 4.10: Testing Different Learning Rate of the GBDT to Optimize Accuracy

We test both the number of learning trees and the learning rate. All the combinations between the learning rate from 0.1 to 1.0 and the number of trees from 1 to 200 are tested. There are 11 variations for the learning rate and 11 variations for the learning trees. Each combination is tested by 10-fold cross-validation. The result is $11 \times 11 = 121$ models are trained and evaluated in 10 folds. The optimal result was reached with the learning rate of 0.1, and with the number of trees equals to 200. This gives accuracy 88.94 % +/− 0.50 % and AUC of 0.860 +/− 0.007. Figure 4.11 shows the accuracy of the different number of trees with the selected range of learning rate. However, the accuracy of 0.19 learning rate with 160 number of trees is almost similar to the best one. In general, a low learning rate causes fewer corrections for each tree added to the model. Therefore, the smaller the learning rate is the more trees are required to be added to the model. Moreover, the small max depth of the tree can affect this need for more learning trees.

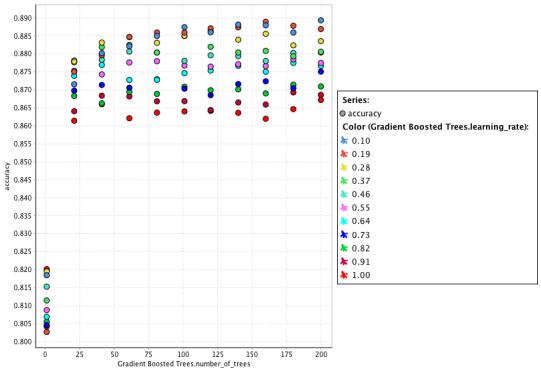


Figure 4.11: Testing Different Learning Rate with Different Number of Trees of the GBDT to Optimize Accuracy

- **main criterion AUC:** Second, we will find the optimal values to optimize the AUC metric. The number of learning trees is tested with the range between 1 and 200 with 10 steps. The other parameters are set to be the same for all the 10 folds cross-validation iterations. The learning rate was 0.1, and the maximum depth of the tree was 5. The result in Figure 4.12 shows that 180 trees give the highest AUC of 0.863 +/− 0.008.

  Then we test the learning rate which ranges from 0.1 to 1.0 with 10 steps. The amount of learning trees was set to 180 and maximum depth of tree was set to 5. The result is shown in Figure 4.13. Learning rate of 0.1 gives higher AUC = 0.861 +/− 0.008.

  Finally, since we find that the optimal parameters by optimizing AUC give higher predictive performance than by optimizing accuracy, we test this heavy computation. We test all the combinations between the number

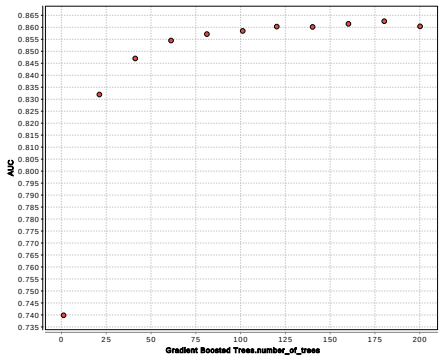


Figure 4.12: Testing Different Number of Learning Trees of the GBDT to Optimize AUC

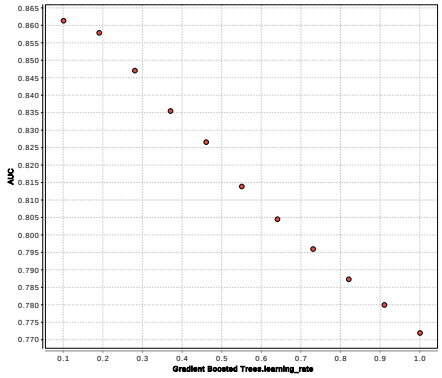


Figure 4.13: Testing Different Learning Rate of the GBDT to Optimize AUC

of learning trees, the learning rate, and the maximum depth of a tree to optimize the AUC metric. We test those value ranges: the learning rate from 0.1 to 1.0, the number of trees from 1 to 200, and the maximum depth of tree from 1 to 20. There are 11 variations for the learning rate, 11 variations for the learning trees, and 11 variations for the maximum depth. Each combination is tested by 10-fold cross-validation. The result is $11 \times 11 \times 11 = 1331$ models are trained and evaluated in 10-fold. The optimal predictive performance of AUC was reached with a learning rate of 0.1, the number of trees equals 140, and a maximum depth of 5. This gives the highest AUC of 0.862 +/− 0.009. The accuracy 88.72 % +/− 0.59 %, precision 49.87%, and recall 43.59%. The relation between the learning rate and the amount of trees is represented in Figure 4.14. It shows that the AUC is low with a large learning rate and higher with a small learning rate and a big number of trees. Moreover, it shows the maximum AUC reached by the 10-fold cross-validations is by 140 trees and 0.1 learning

rate. The relation between the learning rate and the maximum depth is represented by Figure 4.15.

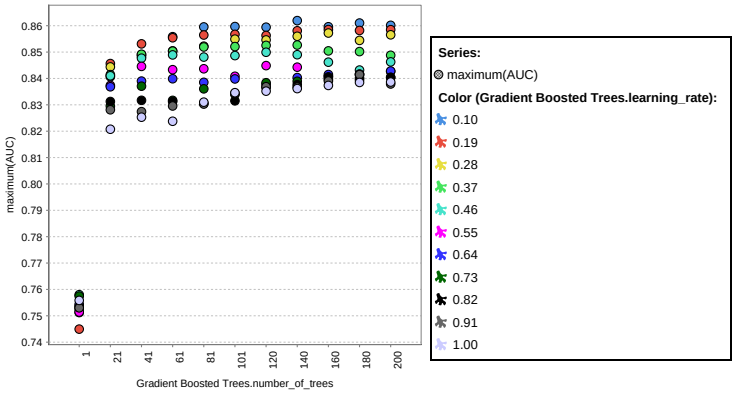


Figure 4.14: Testing Different Learning Rate with Different Number of Trees of the GBDT to Optimize AUC

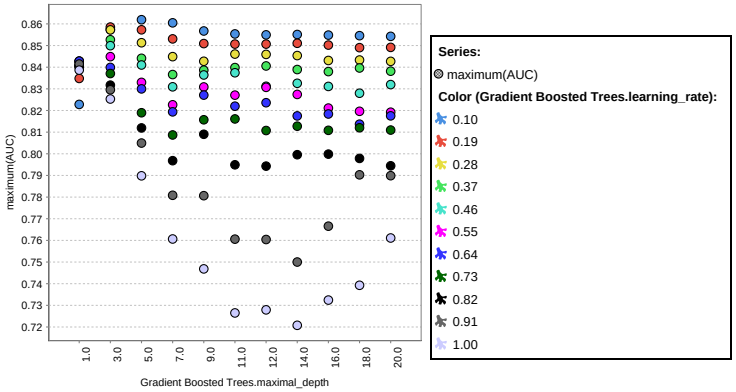


Figure 4.15: Testing Different Learning Rate with Different Maximum Depth of Trees of the GBDT to Optimize AUC

Thus, we fixed the three primary parameters to the optimal values: learning rate = 0.1, the number of trees = 140, and a maximum depth = 5. Then, we optimize the other parameters, minimum rows from 10 to 200, and the sample rate from 0.1 to 1.0, all in linear ten steps. The minimum rows of 143 and the sample rate of 0.82 provide the highest AUC 0.865 +/− 0.007 (see Figure 4.16). The other metrics are also improved; accuracy 88.25%, precision 47.78%, recall 49.06%, and f_measure 48.36%. Figure 4.16 shows that low sample rates give low AUC.

To summarize, tuning the parameters by optimizing the AUC provides better performance than by optimizing the accuracy. Optimizing the primary and the

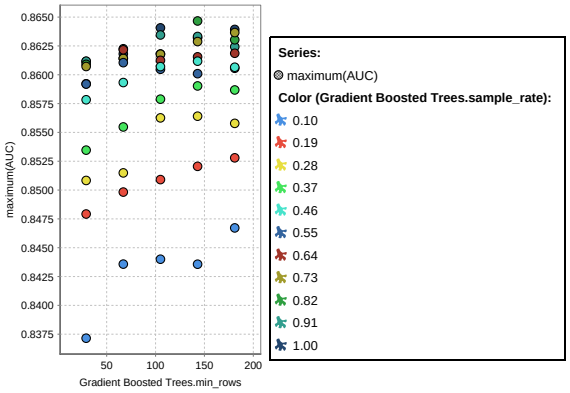


Figure 4.16: Testing Different Minimum Rows with Sample Rates of the GBDT to Optimize AUC

other parameters improve the GBDT performance with the default parameter values from AUC of 0.859 to 0.865.

Strength and Weaknesses

GBDT is one of the most powerful and widely used supervised learning models. Its main weakness is that it requires careful tuning of the parameters. Moreover, it may take a long time for training. Similarly to the DT, the GBDT does not require scaling features. Furthermore, it works well, even with a mixture of binary and continuous features, however, not with high-dimensional sparse data.

4.6 K-Nearest Neighbor for Patient Similarity-based Health Prediction

K-nearest neighbor KNN is one of the simplest machine learning methods. It is one of the top 10 data mining algorithms listed by [90]. KNN is another approach for analyzing patient similarity for health prediction. It is described by [35]:

- **Supervised learning algorithm:** It refers to the predicted class label of a test instance from the labeled training data. From the input patient vectors with class labels, the KNN method can predict the class label for an unseen example (i.e., a new patient).
- **Non-parametric learning algorithm:** KNN algorithm has no dependency on parameter. The parameters are not fixed in advance. Thus, no assumptions are made on the shape of the decision boundary. However,

this property causes a performance reduction with a dataset that has a large feature number.

- **Instance-based learning algorithm:** The prediction for a new instance x is from the training instances. That is why KNN is called a lazy learner. It does not build a training model for generalization. However, to estimate a class label for a new instance or test instance (i.e., patient x), the KNN learner compares it to all the training instances and find the nearest neighbors that help with prediction.

Prediction of the class label for a new instance x is from the training instances. A user-defined a positive integer k, it identifies the k nearest neighbor to x from which the predicted class of x is assigned. Hence, distances or similarities between x and all the training instances are computed. Various distance metrics can be used, for instance, the discussed ones in Section 3.2.2.2. To estimate the predicted value to x a conditional probability for each class j in the k set is calculated as follows Equation 4.9 [35]:

$$Pr(Y = j \mid X = x_0) = \frac{1}{k} \sum_{i \in N_0} I(y_i = j) \; . \tag{4.9}$$

This gives a fraction of points with class value j. Where Y is the class label and x_0 is the test instance that we want to predict its class label. N_0 represents the k nearest neighbors to x_0. $I()$ is an indicator function that returns 1 when the argument is true and 0 otherwise. For each value of the class label occurs in the nearest neighbor instances (N_0), the predicted value pr is calculated. The predicted class value of x is the class with the highest probability Pr among N_0 (i.e., the class to which the majority of k instances belong).

Another formula of Equation 4.9 to predict the class label of a test instance is given by [90] and called Majority Voting:

$$Pr(Y = j \mid X = x_0) = argmax_{y_i} \sum_{i \in N_0} I(y_i = j) \; . \tag{4.10}$$

Important Parameters

Some critical key choices and issues affect the performance of KNN [90]:

- **The value of k nearest neighbour:** "The choice of K has a drastic effect on the KNN classifier obtained" [35]. The choice of k value affects the performance of KNN. Small k gives low bias but very high variance classifier. The opposite with large k the classifier gives high bias and low variance. There is no a rule of thumb of the k value. For avoiding tied votes of the binary labels among N_0, it is helpful to choose k odd. However, the decision of the exact value of k depends on the data.

- **The approach to combine the class labels:**

  After estimating the probability of each class label occurring in the set of k nearest neighbors instances by Equation 4.9, a decision has to be made to select the predicted class label. The general approach is to take the majority vote (i.e., select the predominant class label). A weakness of this approach occurs when the nearest neighbors are widely in their distance, and the closest neighbors have the reliable indicator to the class predicted value. The class label that commonly occurs will affect the predicted value. Therefore, to assure that the nearest neighbors affect more than the distant ones assign a weight to the neighbors' contribution or vote. Its distance weights the vote of each neighbor. The weight factor is: $w_i = 1/d(x_0, x_i)^2$ the weight of the x_i vote is the reciprocal of the squared distance between the test instance x_0 and x_i. This approach is by Equation 4.11:

$$Pr(Y = j \mid X = x_0) = \frac{1}{k} \sum_{i \in N_0} w_i \times I(y_i = j) \; . \tag{4.11}$$

  This approach is less sensitive to the choice of k.

- **The choice of distance metric:**

  Hu *et al.* [33] examine the effect of the used distance method on the classification performance of KNN. They test four distance methods Euclidean, Cosine, Chi-square, and Minkowsky on medical datasets. The classification accuracy is tested on three different feature data types: numerical, categorical and mixed. They find that Chi-square distance function outperforms the other distance functions over all the different data types. In specific, with the mixed data type the other distance functions perform worst. However, this is not a silver bullet to which is the best distance metric. The selection of the best fit distance metric should be based on the data.

Different values of these factors can be tested to find the best choice. Finding the lowest test error rate sheds light on the right decision. However, using test data for this test purpose cause overfitting. Hence, cross-validation is one approach to use. In this approach, a subset of the training dataset is used for testing.

To predict the class of a test instance x the KNN classifier identifies the nearest observations to x by measuring the distance. Therefore, similar to any distance metric the scale of the variables is essential in KNN. Large-scale variables have more significant effects on the distance measure and then on the KNN classifier.

In the previous approach, Section 3.2.2.2 to predict an output class of patient x, we need to compute the distance between x and all the other patients in the dataset. Then the output data of similar patients are used with one of a

predictive model to predict such an output. However, in KNN after calculating all the similarities or distances between x and all the other patients in the training dataset, a majority vote of the k nearest neighbor is deciding the predicted value.

Tuning KNN Parameters

We use a dataset of 32.635 patients. For KNN model we test different parameters of distance metrics and the value of k. First, we test the distance metric parameter with 10 fold cross-validation. Different distance metrics are used, but all with the same k. In this example, we test $k = 5$. From Figure 4.17 you can see the different performance KNN model has with different distance metrics.

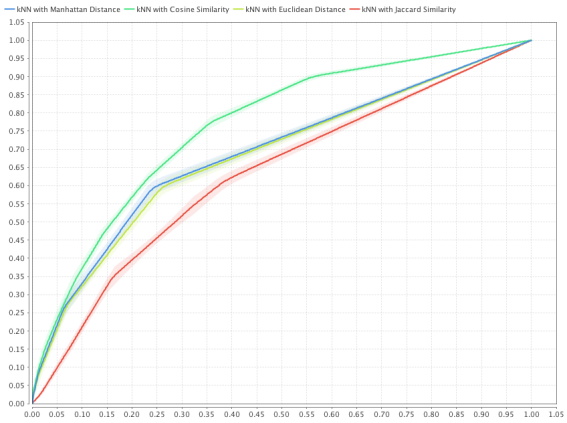


Figure 4.17: KNN with Different Distance Metrics

- **main criterion Accuracy:** We test k parameter for a specific distance metric Euclidean Distance to optimize accuracy. To find the best k we test with 10 fold cross validation the value from 1 to 50 with linear scaled 1,6,11,16,21,26,30,35,40,45,50. Figure 4.18 shows the performance difference of different k values. As a result the optimal k is $k = 21$ that gives 87.02 % accuracy.

  Then, we test the parameter of the approach to combine the class labels—the distance metric set to the Euclidean distance and k value set to $k = 21$. The only thing we change is the way to decide the predicted class label. We compare the two different approaches: the majority vote and the weighted vote. The approach of the weighted vote is selected by Grid optimize parameters. However, the accuracy measures of using the two approaches were almost the same which is 88.84%.

- **main criterion AUC:** We search for the optimal k value to optimize AUC. We use the same range and linear steps for k as the previous test.

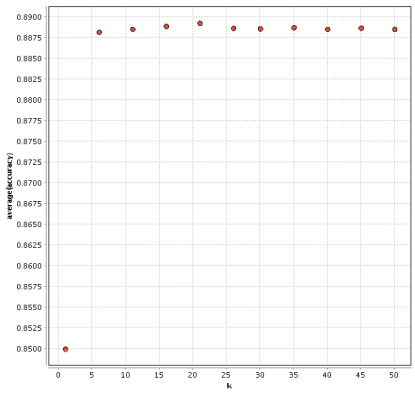


Figure 4.18: KNN with Different k Values with Euclidean Distance to Optimize Accuracy

Figure 4.19 shows the performance difference of different k values. As a result, the optimal k is $k = 50$ that gives AUC 0.777 +/− 0.009. The other metrics accuracy 88.86 %, precision 79.05%, a very low recall 0.90%, and f_measure 1.78%. Imbalanced data affect the ML model's parameter values. For instance, KNN with imbalanced class distribution dataset with increasing the K the predicting of the minority class is lowered (i.e., the larger the k is the less probability of minority class accrues as neighbors and more majority class instances as neighbors).

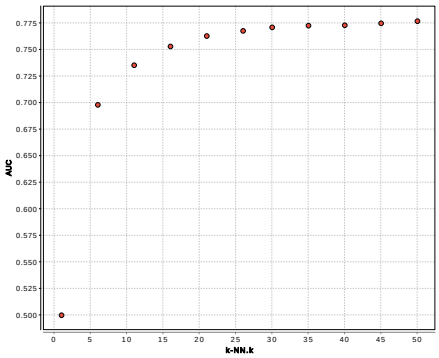


Figure 4.19: KNN with Different k Values with Euclidean Distance to Optimize AUC

The last parameter we test the tuning effect is the approach to combine the class labels. We select the same distance metric (the Euclidean distance) and the same k value $k = 50$ with 10-folds cross-validation. The approach of the weighted vote is selected by Grid optimize parameters. The accuracy of both approaches was almost the same. However, looking more in

detail of the performance result, we find that differences were in AUC and the Precision and Recall of predicting the positive class (i.e., predicting mortality). The majority vote gives AUC of 0.777% and 70.07 precision and 0.85% recall. The weighted vote improves the performance where it gives 0.778% AUC, 1.15% recall, and precision 77.78%. The reason lies back to a problem we have in our dataset (imbalanced class distribution). In the following chapter, we will discuss the problem in detail. We previously mentioned that the class label that commonly occurs would affect the predicted value. In our dataset, the survival class is the common one. Thus, using the majority vote with a larger k value is not a good choice in our case.

Strength and Weaknesses

The KNN algorithm is easy to implement. However, a drawback of KNN is that its performance is affected by the increases in the feature size. "This decrease in performance as the dimension increases is a common problem for KNN"[35]. In the high dimensional space (i.e., large feature number), there are very few neighbors near to any test instance. There is a small difference between the nearest neighbors and the farthest neighbors. Therefore, KNN will be slow to find the nearest neighbor. This deterioration in the performance of KNN is a fact of the non-parametric approaches, which has poor performance with large feature number. This problem is called the curse of dimensionality.

Another drawback is the need for a large memory where all the training data need to be stored since the decision on prediction is based on all the training data instances. Furthermore, the test phase is costly. It is a lazy instance-based learner that no generalization model needs to be built – the classification is estimated for each test instance. Thus, the classifying phase to predict the class of an instance is computationally expensive. This requires pairwise distance computations to all the training instances, which is costly with a large training dataset. However, some methods exist to avoid pairwise distance computations to all the training instances. The goal is to help reduce the computational cost without affecting the classifying accuracy.

4.7 Choosing the Optimal ML Model

In the previous, we test the four models with tuning different parameters. In this Section, we compare the accuracy of the four models. The highest accuracy of the optimal parameters (that found with main criterion was accuracy) of each model is represented in Figure 4.20.

Figure 4.20 shows that the models have high accuracy. Moreover, they have almost the same predictive accuracy. This suspicious result makes us look closer to the test result.

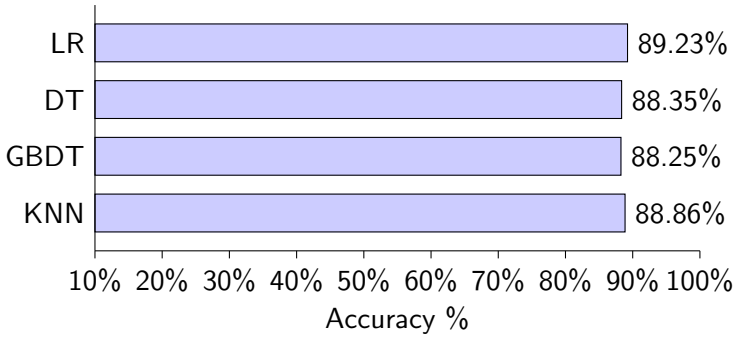


Figure 4.20: Compare Models Accuracy

Performance Metric	LR	DT	GBDT	KNN
Accuracy	89.23%	88.35%	88.25%	88.86%
AUC	0.801	0.756	0.865	0.778
Precision	58.63%	45.07%	47.78%	77.78%
Recall	13.41%	16.36%	49.06%	1.15%

Table 4.1: Compare Models Performance

We retest the four models with the optimal parameters we found from the previous sections (when AUC was the main criterion). Besides the accuracy metric, we calculate different performance metrics. The result are given in Table 4.1 and Figure 4.21

We find that the models were successful in predicting survival cases rather than the death cases. The high predictive accuracy was a sign for overall prediction of the majority class which is the survived case. This situation where the higher accuracy metric is not an indicator of an excellent classifier performance is called *Accuracy paradox* [82]. It is paradoxical when accuracy is not the good metric for the predictive model.

The used dataset MIMIC-III from 32,635 include 28,974 alive patients and only 3,661 dead patients. The ratio of the instances of Class-1 (survived patient) to Class-2 (died patient) is 89:11. This problem of imbalanced class distribution causes the classifier to be extremely biased toward the majority class. As a result, the models' high accuracy was obtained by predicting all instances as a majority class. Thus, it is the model accuracy in predicting most of the dominant class instances and discounting the accuracy in predicting the minor class ones. Nevertheless, the minority class (i.e., the died patient) is the positive class which is the class of interest (i.e., we focus on predicting this class).

We can conclude from this case that having a highly accurate model is not enough indication of a useful model. Valverde-Albacete *et al.* [82] state that a predictive classifier model with low accuracy may have higher predictive power than a model with high accuracy. In particular, they stress that this is applied

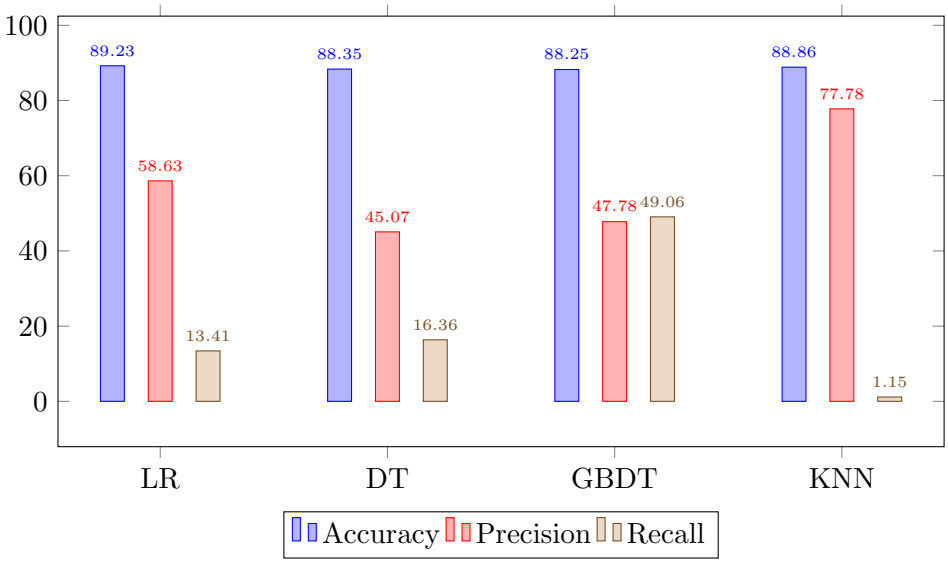


Figure 4.21: Compare Models Prediction Performance

to the highly *imbalanced* or *skewed* training data where the classifier produces a highly accurate result by assigning all the cases to the majority class. For instance, even though DT and KNN have high accuracy, they have a very low Recall (that measures how often a positive class instance is truly predicted as a positive one). Therefore, we should use other metrics to evaluate our models beside accuracy. Hoens *et al.* and Chawla [30, 10] state that the predictive accuracy is inappropriate when data is imbalanced. They recommend alternative metrics to evaluate the classifier performance on the imbalanced dataset. Hoens *et al.* [30] recommend balanced accuracy, ROC curves, Precision and Recall, and F-measure. Chawla [10] recommends ROC curves, Precision and Recall, and Cost-sensitive Measures (cost curves and cost matrix). He *et al.* [29] state that accuracy is sensitive to the class distribution, while Precision and Recall are not. Therefore, from now on, we will use Recall, Precision, AUC, and F-measure as classification performance metrics. Thus, also considering AUC in selecting the features was a better metric than accuracy. The models' AUC are shown in Figure 4.22.

Looking to Figure 4.21 and the Table 4.1, we find that GBDT has the highest and the best performance trade off through all the metrics (accuracy, AUC, Precision and Recall). In specific, for the Recall which we consider a critical metric the GBDT has the highest value. Moreover, comparing to DT and KNN the LR has a higher AUC and Recall values. The DT give the random guessing value in AUC and the lowest Recall value. The KNN also has a low Recall and a low AUC values in comparison with GBDT and LR.

Keep in mind that GBDT and LR give this high performance without any

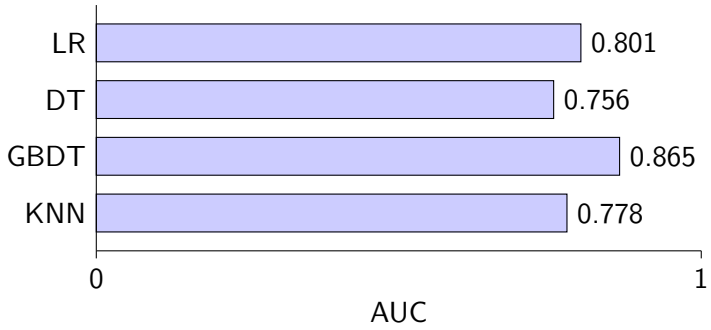


Figure 4.22: Compare Models AUC

performance optimization regarding the data pre-processing (to solve the imbalanced data problem) and without further feature selection. Therefore, For the next tests of performance optimization, we will consider GBDT and LR which we expect they give high performance. In contrast, we hypothesized that DT and KNN would not provide significant improvements.

5

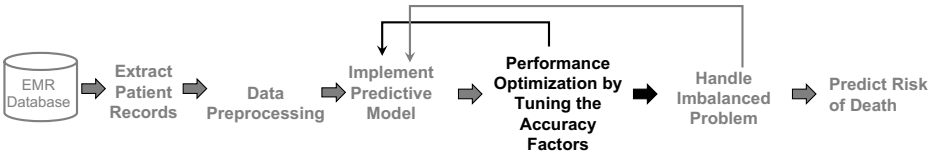
Performance Optimization

This chapter presents different approaches for performance optimization. It discusses the effect of normalized and un-normalized data on prediction performance. It shows the practical implementation of different feature selection methods to find the optimal subset of the features. Filter, wrapper, and embedded methods are applied. In the end, it proposes an approach for performance optimization by filtering the patients by the diagnoses code.

Contents

5.1 Scope of the Chapter

In this chapter, we will represent the step of performance optimization by tuning the accuracy factors. Different approaches are described as further data pre-processing, feature selection, and filtering patients by their diagnoses codes. For every optimization, we will apply the predictive ML model to test the effect of this performance optimization.



5.2 Data Pre-processing Normalized vs. Un-normalized Data

We test if the normalization of the data affects the accuracy of prediction. We test different data Pre-processing approaches. First, the normalized and un-normalized data are tested with the models LR and GBDT. GBDT with un-normalized data results AUC 0.860 +/− 0.011 while the normalized data results AUC 0.860 +/− 0.005. The summarized metrics are shown by Figure 5.1.

LR with un-normalized data gives AUC 0.797 +/− 0.009 while the normalized data results AUC 0.802 +/− 0.012. The other accuracy metrics are shown by Figure 5.2.

The observed result from these tests of the models in both normalized and un-normalized data is that the normalized data require a Little longer time than the un-normalized. The difference between the accuracy metrics of the two data types of the both models is not big. However, in general, the accuracy metrics Recall, F-Measure and AUC are improved with the normalized data.

Besides the binary categorical attributes in our dataset, there are two categorical attributes with more than two values: ICD codes and the ICU service_type. In the previous test, we exclude the text service_type from the data and include the numerical un-normalized ICD codes even in the normalized dataset. Here we transform these categorical variables and add them to the normalized data to test their effect on the models' performance. First, we normalized the ICD codes to the range [0,1] and add it to the normalized data and test GBDT: accuracy: 88.56%, precision: 49.00%, recall: 43.91% , f_measure: 46.21%. In comparison to the previous test of the GBDT with normalized data with un-normalized ICD, the metrics Precision improved Little but the Recall reduced by 4.14808%. The AUC: 0.860 +/− 0.006. Second, we test the service_type attribute. It has four values (MICU, SICU, CCU, CSRU); we transform them into numerical 1,2,3,4, then we do range transformation of [0,1]. We add this

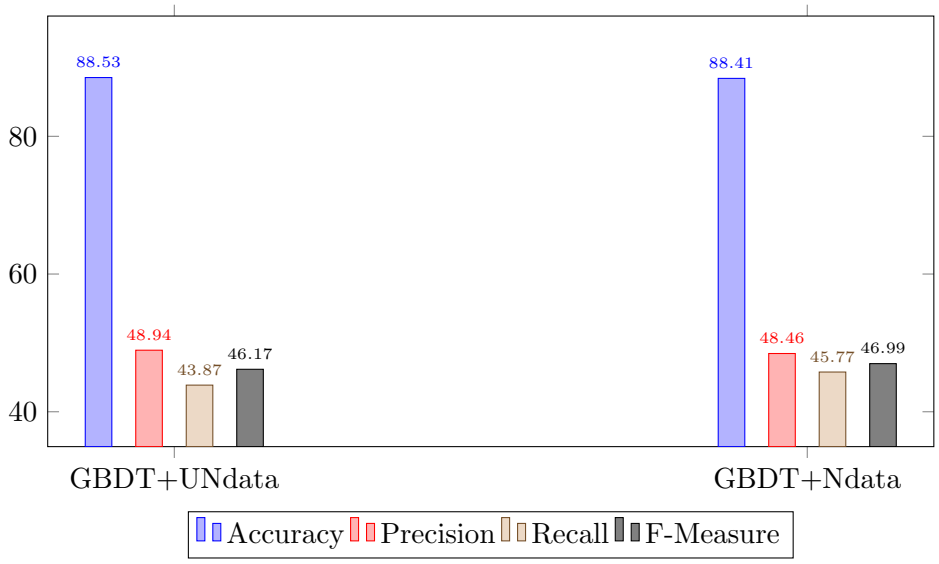


Figure 5.1: GBDT with Un-Normalized and Normalized Data

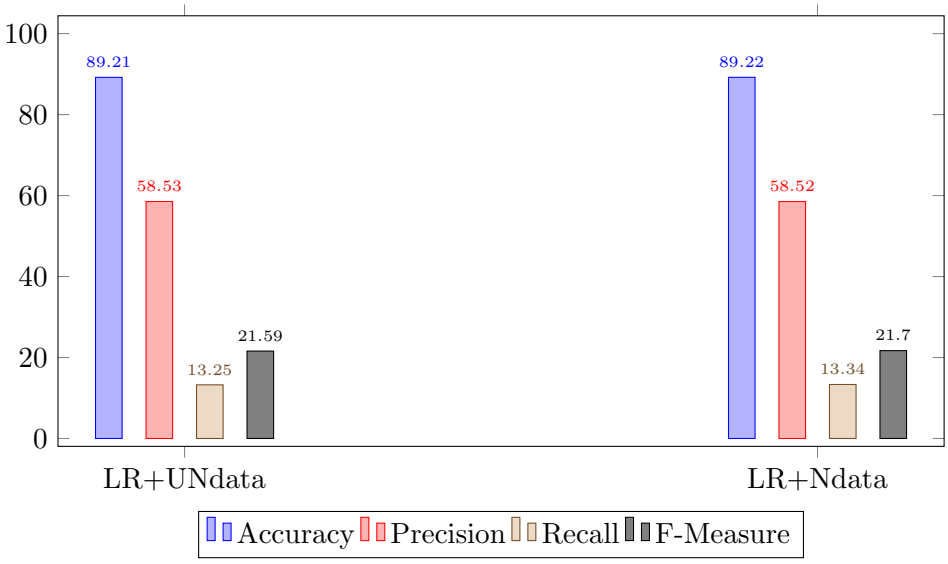


Figure 5.2: LR with Un-Normalized and Normalized Data

attribute to the normalized data with the un-normalized ICD and run GBDT: accuracy: 88.25%, precision: 47.66%, recall: 45.88% , f_measure: 46.67%. AUC: 0.860 +/- 0.005 only the Recall improved a little.

5.3 Result of Feature Selection Methods

We test the two feature selection methods Forward selection and backward elimination for the both models LR and GBDT. The data pre-processing affects the predictive performance; thus, we test on both the normalized data (except the ICD code) and the un-normalized data. Hence we have done model parameters optimization; we use the optimal parameters we found.

5.3.1 Filter Selection by Chi Squared

We implement the GBDT on the dataset that contains the TOP20 features from the Chi Squared weight (see Table 5.1). The GBDT parameters are set to the optimal parameters values that were found to optimize the AUC. The resulted metrics are accuracy of 86.97%, AUC of 0.846 +/− 0.007, precision of 42.98%, recall of 48.42%, and f_measure of 45.47%. The resulted AUC is lower than the AUC of implementing the GBDT with the complete set of features.

5.3.2 Forward Selection

Forward selection method (defined in Section 3.5.2.1) is used to select the optimal features in two approaches for optimizing the accuracy and AUC of the performance metric.

- **main criterion Accuracy:**

  We test Forward Selection approach for LR and GBDT on not normalized data. The models' parameters are set to the optimal values we found to optimize the accuracy. For LR the selected attributes were 9 attributes in total: urine_12h, rr_12h_min, sbp_12h_min, sbp_24h_min, spo2_18h_min, spo2_24h_min, bicarbonate_min, service_type, and vent. It gives AUC 0.748 +/− 0.014. It takes 24 minutes. For GBDT only 7 attributes were selected: urine_18h, hr_18h_min, mbp_24h_min, wbc_min, glucose_min, bun_min, and creatinine_max. It gives AUC 0.750 +/− 0.016. It needs 9 hours and 31 minutes, which is much longer than the required time by LR. All the result are represented in Figure 5.3.

  Then, we test Forward selection approach with the normalized data (except the ICD code) for LR and GBDT. For GBDT, 8 attributes are selected urine_18h, urine_24h, rr_18h_max, hr_24h_min, mbp_18h_min, sbp_6h_min, bun_min, and ccreatinine_min. The result AUC equals to 0.752 +/− 0.015. The result are shown in Figure 5.4. LR selects 17 attributes:

Feature	Weight
Blood urea nitrogen_min	1206.72
Blood urea nitrogen_max	1109.48
Serum HCO3_min	979.23
Serum HCO3_max	765.85
Spontaneous respiratory rate_18h_min	650.84
Spontaneous respiratory rate_12h_min	641.34
Sodium_max	609.99
ICD_code	582.77
Spontaneous respiratory rate_6h_min	571.23
Spontaneous respiratory rate_24h_min	561.54
Systolic blood pressure_24h_min	547.43
Heart rate_24h_max	506.01
Age	465.82
Heart rate_24h_min	430.77
Spo2_24h_min	418.36
Glasgow Coma Scale_min	392.76
Systolic blood pressure_18h_min	342.89
Heart rate_18h_max	336.49
Systolic blood pressure_6h_min	327.11
Use of mechanical ventilation	320.33

Table 5.1: Top-20 Features Weight by Chi-Squared.

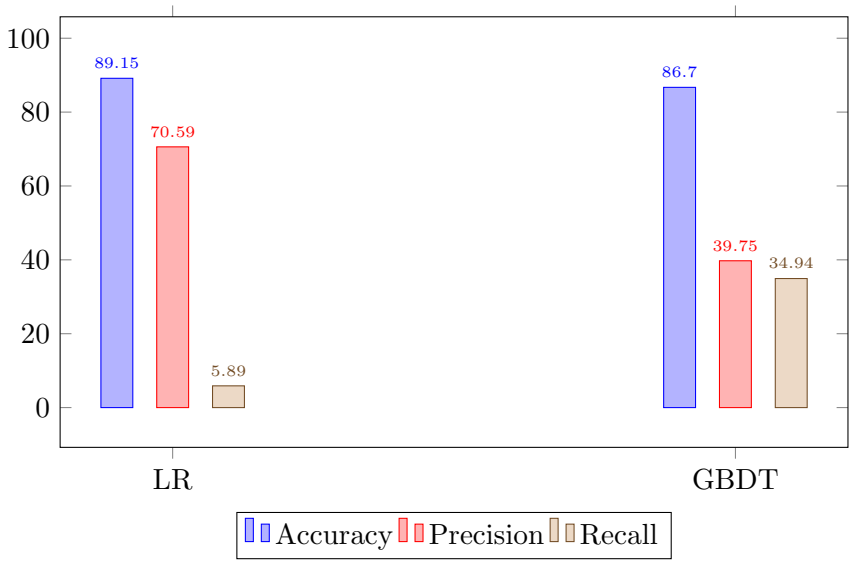


Figure 5.3: Forward Selection with Un-Normalized data

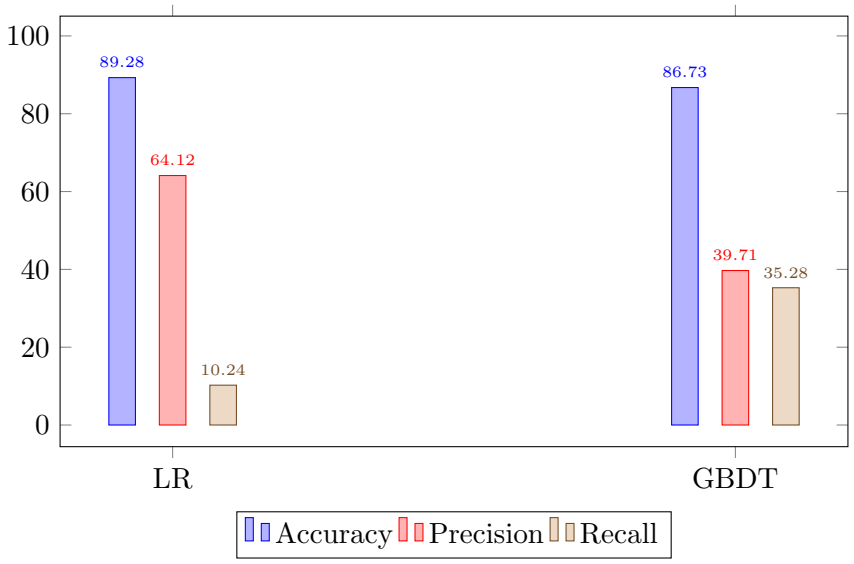


Figure 5.4: Forward Selection with Normalized data

urine_18h, urine_24h, mbp_12h_max, rr_24h_max, mbp_12h_min, mbp_24h_min, rr_12h_min, sbp_24h_min, spo2_12h_min, spo2_24h_min, temperature_6h_min, glucose_max, bicarbonate_min, bun_min, age, vent, and gcs. It has AUC of 0.778 +/− 0.013. LR takes much less time than GBDT. LR takes only 39 minutes, while GBDT takes 10 hours and 29 minutes.

- **main criterion AUC** :

  In the previous tests of forward selection to optimize accuracy, we find that using normalized data provides higher AUC than un-normalized data. Thus, for this forward selection to optimize AUC, we will use only the normalized data.

  The GBDT's parameters are set to the optimal values we found to optimize the AUC. The main criterion for the model performance evaluation with different features is AUC. GBDT selects 16 features: urine_18h, hr_24h_max, rr_18h_min, sbp_12h_min, spo2_18h_min, temperature_6h_min, wbc_min, bicarbonate_min, sodium_max, bun_min, creatinine_max, age, vent, gcs, vasopressor, and icd_code. The result metrics are accuracy = 87.31%, AUC= 0.859 +/− 0.013, precision= 44.56%, recall= 52.10%, and f_measure = 47.95%. It takes 22 hours and 32 minutes.

  Forward Selection with LR when main criterion is AUC selects 31 features: urine_12h, urine_18h, hr_24h_max, hr_6h_max, rr_24h_max, temperature_12h_max, hr_18h_min, rr_12h_min, rr_6h_min, sbp_6h_min, spo2_18h_min, spo2_6h_min, temperature_24h_min, temperature_6h_min, hematocrit_min, hematocrit_max, wbc_max, glucose_min, bicarbonate_min, potassium_min,

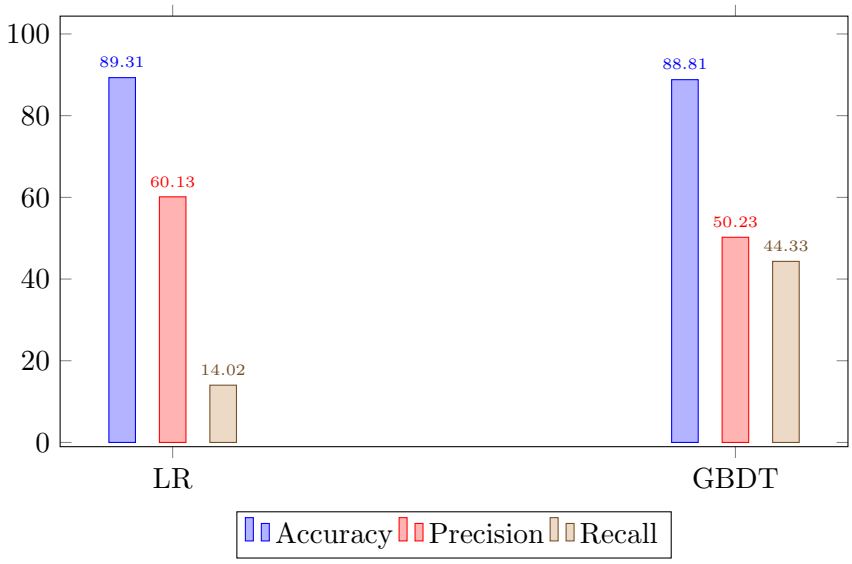


Figure 5.5: Backward Elimination with Un-Normalized data

potassium_max, sodium_min, sodium_max, bun_min, bun_max, ccreatinine_min, age, vent, gcs, vasopressor, icd_code. The performance metrics are 89.13% accuracy, 0.802 +/− 0.014 AUC, 57.74% precision, 12.08% recall, and 19.90% f_measure. It needs much less time than GBDT, only 1 hour and 10 minutes.

5.3.3 Backward Elimination

Backward Elimination method (defined in Section 3.5.2.1) is used to select the optimal features in two approaches for optimizing the accuracy and AUC as the performance metric.

- **main criterion Accuracy:** We test Backward Elimination approach with the not normalized data for LR and GBDT. In the case of Backward Elimination with LR two attributes were eliminated: mbp_6h_max and icd_code. It gives AUC 0.803 +/− 0.011. With GBDT only one attribute was eliminated equal spo2_18h_max. It gives AUC 0.860 +/− 0.009. LR takes 45 minutes, while GBDT takes 5 hours and 38 minutes. The results are shown in Figure 5.5.

  Then, we test Backward elimination approach with the normalized data (except the ICD code) for LR and GBDT. The LR eliminates 6 attributes: urine_24h, hr_18h_max, spo2_12h_max, temperature_24h_min, sodium_max, and icd_code. It produces AUC of 0.798 +/− 0.010. GBDT eliminates only one attribute: hr_6h_max with AUC of 0.859 +/− 0.006.

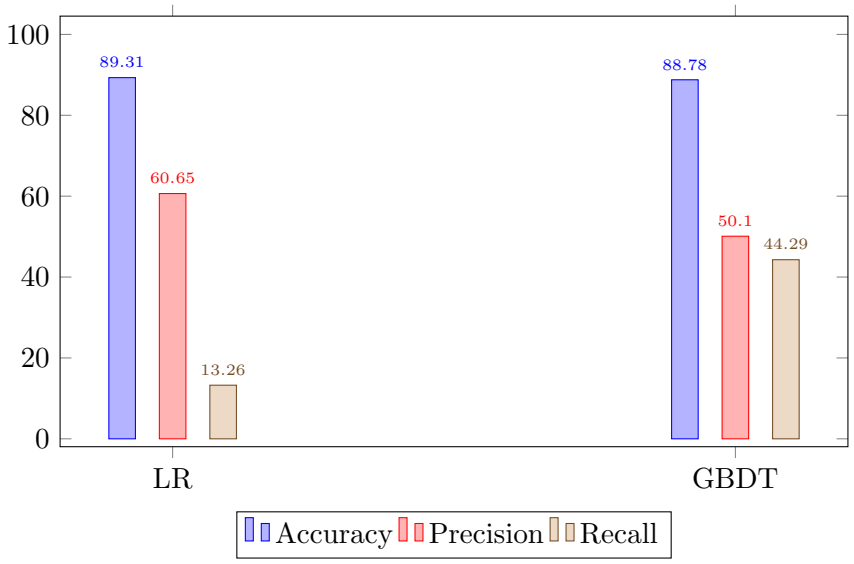


Figure 5.6: Backward Elimination with Normalized data

This test takes 5 hours and 33 minutes for GBDT and 1 hour and 37 minutes for LR. The result is represented in Figure 5.6.

- **main criterion AUC:**

  GBDT with backward elimination to optimize AUC eliminates only one feature wbc_max. The GBDT's parameters are set to the values that optimize AUC. The main criterion to stop eliminating features is decreased AUC. These tests resulted in an accuracy of 88.30%, AUC of 0.864 +/− 0.006, the precision of 48.06%, recall of 49.55%, and f_measure of 48.70%. The selection takes 6 hours and 20 minutes.

  LR with backward elimination to optimize AUC eliminates those three features hr_18h_max, hr_18h_min, and creatinine_min. This elimination gives accuracy of 89.23%, AUC of 0.803 +/− 0.005, precision of 58.80%, recall of 13.48%, and f_measure of 21.90%. It needs 59 minutes, which is much less time than GBDT.

5.3.4 Embedded Feature Selection Method of GBDT

Embedded feature selection means no extra feature selection process has to be implemented since the feature selection is made during GBDT model implementation (see Section 3.5.2.3). Thus, no extra time needed only the required time to implement the GBDT, which is only a few minutes. Moreover, this shows the interpretability of our used GBDT model.

Our constructed GBDT provides a list of feature importance. The top-20 features are listed in Table 5.2. It contains almost all the forward selected

Feature	Relative Importance	Scaled Importance
Urinary output_18h	581.73	1.00
ICD_code	466.13	0.801
Blood urea nitrogen_min	463.23	0.796
Bicarbonate (serum HCO3)_min	334.72	0.575
Use of mechanical ventilation	316.58	0.544
Sodium_max	259.37	0.446
Glasgow Coma Scale	248.11	0.426
Age	228.18	0.392
Urinary output_6h	202.69	0.348
Heart rate_24h_max	202.10	0.347
Systolic blood pressure_24h_min	165.77	0.285
Blood urea nitrogen_max	146.17	0.251
White blood cell_min	132.71	0.228
Serum creatinine_max	126.97	0.218
Respiratory rate_6h_min	121.67	0.209
Body temperature_6h_min	116.16	0.200
Serum glucose_min	111.06	0.191
Spo2_24h_min	105.49	0.181
Respiratory rate_18h_min	96.95	0.167
Respiratory rate_24h_max	91.81	0.158

Table 5.2: Top20 Features by GBDT.

features when the main criterion is AUC. Except it includes vasopressor feature and the minimum value of Systolic blood pressure not from the 12h but 24h, and the minimum value of spo2 not from the 18h but 24h. We conclude from this that feature selection to optimize AUC leads to the optimal features and performance than optimizing accuracy - in specific, with the case of imbalanced data. Moreover, it shows the efficiency of the forward selection in comparison to backward elimination.

We implement the GBDT with only the top-20 features selected by the model. We set the parameters to the optimal values to optimize the AUC: learning rate = 0.1, the number of trees = 140, the maximum depth = 5, minimum rows =143, and the sample rate=0.82. The result metrics are accuracy = 87.85%, AUC= 0.862 +/− 0.008, precision= 46.35%, recall= 50.31% , and f_measure= 48.16%. In comparison to implementing GBDT with the complete set of features 74, the AUC is decreased by only 0.3%, and the recall improved by 2.55%. In general, we can say that the result of implementing the model with only the top-20 features is highly competitive to the result of the complete set of features. Training and testing the GBDT with the top-20 features achieves AUC of 0.862, which is very close to the achieved AUC of 0.865 of training the model with the complete set of features. Moreover, the time to train and test the model with

the top-20 features takes less time than with the completer set of features. With the top-20, it takes 1 minute and 35 seconds, while the complete set of features takes 5 minutes and 10 seconds.

We implement the LR with only the top20 features selected by the GBDT. accuracy: 89.27% , AUC = 0.790 +/− 0.009, precision= 61.49%, recall= 11.72%, and f_measure= 19.68%. In comparison to implementing LR with the complete set of features 74, the AUC is decreased by 1.4%, and the recall decreased by 12.6%. In general, we can say that LR works better with the complete set of features.

5.3.5 Summary

To summarize, from the previous tests of feature selection methods on both data transformations and both models, we find these results: in general, the Forward Selection with normalized dataset gives higher Recall for the both models than with the un-normalized data. The GBDT with not normalized data Forward Selection is costly with respect to computation power and time, which takes around 10 hours while Backward elimination needs less than 10 hours. The GBDT has a higher Recall and Precision in Backward Elimination in comparison to Forward Selection. However, Forward Selection uses only 7 attributes, while Backward elimination uses the original large number of features minus 1 attribute. Similar to LR, it has higher Recall and Precision in Backward Elimination than Forward Selection. Therefore, the Backward Elimination gives high Recall and Precision and in reasonable computation time but needs high dimensional features while Forward Selection uses really few features but needs high computation time and gives good Recall and Precision. Here we see a trade-off between the number of features and the prediction accuracy. Furthermore, we find that the models work differently with different features [12].

We compare all the implemented feature selection methods in terms of time cost and the performance results in 5.3. Feature selection by optimizing AUC costs much time than optimizing accuracy. However, it produces a better prediction performance. Backward selection by optimizing AUC requires less time than forward selection by optimizing AUC. Moreover, it provides higher AUC than the forward selection. However, forward selection uses much fewer features and produces highly competitive AUC. In general, the time required to implement feature selection on un-normalized data is less than the one for normalized data. The time to complete forward selection for GBDT is much higher than to complete backward selection. However, it is the opposite situation for LR. Overall, GBDT needs much longer time: more than the double of the time required by LR.

It is not only the features or the data format or feature selection method that affect the prediction performance but also the model itself. It is not only the used features or the data format that affect the feature selection method and prediction performance but also the used predictive model itself. The models

Feature Selection Approach	Time Cost	AUC
Filter Selection by Chi Squared+Normalized data	00:00:01	
(Top-20)+LR	00:00:04	0.768
(Top-20)+GBDT	00:01:30	0.846
Forward Selection (optimizing Accuracy)		
Un-normalized data+LR	00:24:00	0.748
Un-normalized data+GBDT	9:31:00	0.750
Forward Selection (optimizing Accuracy)		
Normalized data+LR	00:39:00	0.778
Normalized data+GBDT	10:29:00	0.752
Forward Selection (optimizing AUC)		
Normalized data+LR	1:10:00	0.802
Normalized data+GBDT	22:32:00	0.859
Backward Elimination (optimizing Accuracy)		
Un-normalized data+LR	00:45:00	0.803
Un-normalized data+GBDT	5:38:00	0.860
Backward Elimination (optimizing Accuracy)		
Normalized data+LR	1:37:00	0.798
Normalized data+GBDT	5:33:00	0.859
Backward Elimination (optimizing AUC)		
Normalized data+LR	00:59:00	0.803
Normalized data+GBDT	6:20:00	0.864
Embedded Feature Selection Method of GBDT+Normalized data	00:05:10	
(Top-20)+LR	00:00:05	0.790
(Top-20)+GBDT	00:01:35	0.862

Table 5.3: Compare Feature Selection Time Cost and Prediction Performance.

react differently with the features.

At the end of implementing feature selection by testing on both the normalized data and un-normalized ones, we are still not able to achieve high predictive performance. The highest Recall we got is 44.33 of GBDT with Backward Elimination. In comparison to the original Recall GBDT has without feature selection 41.82, it is only improved by 5.83%. Moreover, for LR the improvement in Recall is only by 4.45%. This low-performance improvement in predicting the critical cases is due to the imbalanced data we have. However, the performance result of using the selected features from the embedded method of GBDT is highly competitive to the result of the complete feature set.

5.4 Data Sampling with Patient Filtering by Diagnoses Code

As we find in subsection 3.5.2.3 in Table 5.2 that ICD has a high importance in making a decision of risk of death in GBDT. It is the second most important feature for prediction. Thus, predicting mortality of patients with similar ICD codes will help to increase predictive performance. Therefore, I implement the approach of mortality prediction for similar patients (i.e., with the same disease classification). We filter the patients by the ICD code.

Some works are done on predicting mortality based on data of similar patients on a large number of features (which has a weakness of High computational time and complexity). I implement the approach of mortality prediction by applying the ML model (on top of similar patients based only on one feature, which is the ICD) with the same disease classification. It avoids these problems of similarity calculation of a large number of features. It improves the accuracy of the predictive model. Furthermore, it outperforms the previous works done on implementing the ML model on top of patient similarity of large features [50, 48].

In MIMIC-III, the ICD codes are ICD-9 represented by six characters in length. The decimal point is between the third and fourth digit. We are interested in the first level of ICD, which is represented in the first three digits. Therefore, we kept only the first left three digits and excluded the rest.

The top three codes across hospital admissions for patients aged 16 years and above, as found by Johnson *et al.* [40], were 414.01 by 7.1%, 038.9 by 4.2%, and 410.71 by 3.6%.

We have a dataset of 32635 patients. We filter the patients by the classification list of the ICD code, as shown in Figure 5.7.

5.4.1 Filtering the Group of the Highest Occurrence Code

We compare the performance of the GBDT with filtering the patient by a specific group of ICD codes (390-459), which are the highest occurrence codes in the dataset (11,272 patients). The result is compared with testing GBDT on the same dataset size and same classes ratio 92:8 with randomly selected ICD codes (stratified dataset and the ICD codes not filtered). We use this same size and class ratio dataset for fair comparison because the original dataset is larger and has a different class ratio. The GBDT with group (390-459) of ICD code gives AUC 0.894 +/− 0.014 while GBDT random ICD codes gives AUC 0.844 +/− 0.021. The result in Figure 5.9. Selecting a specific classification group of ICD codes makes the metrics highly improved in specific the Recall. However, it is not yet good accuracy. The reason is that this limited size of the dataset still has the main problem of the imbalanced dataset. Even though this ICD group of (390-459) has the highest mortality occurrences as shown in Figure 5.8 from

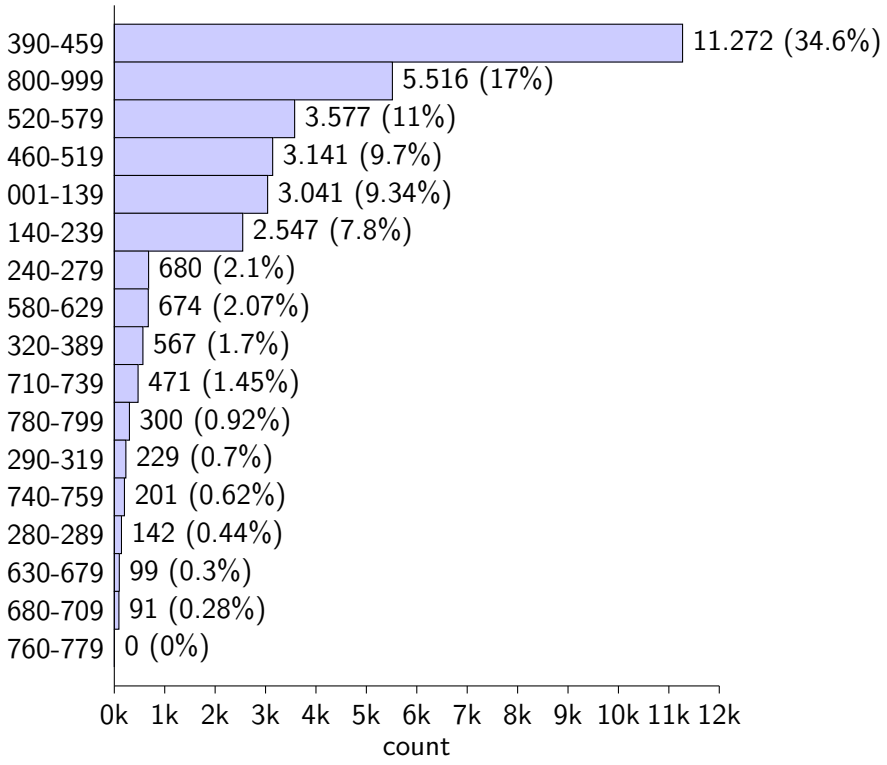


Figure 5.7: ICD Codes Distribution Through out the Dataset

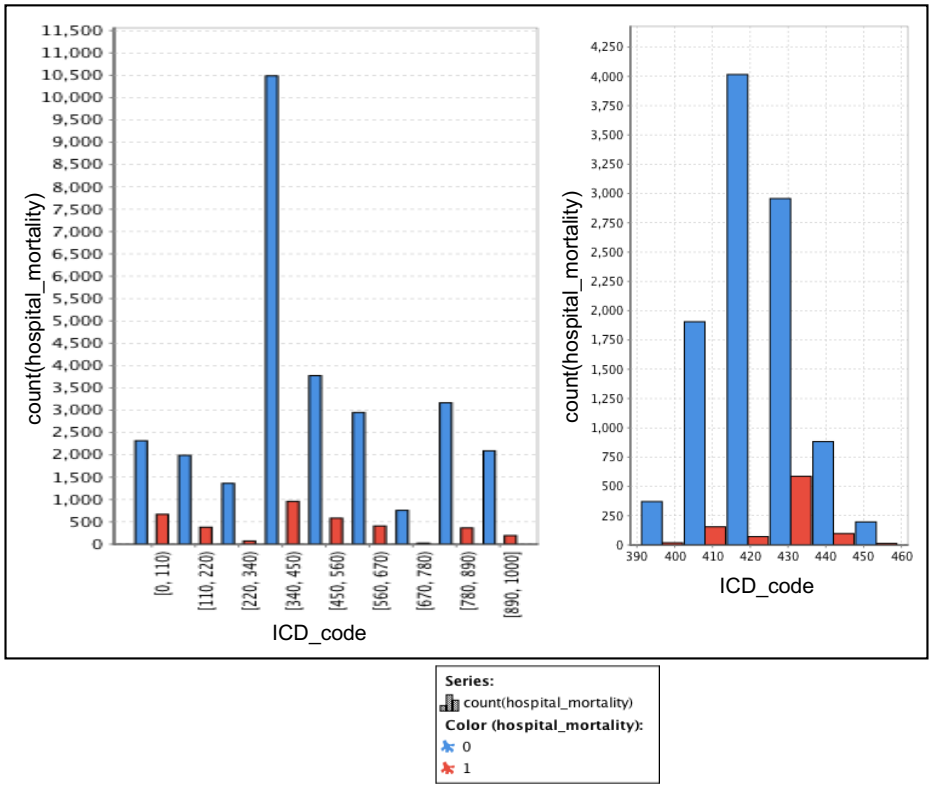


Figure 5.8: Counting Mortality According to all the ICD Codes and According to ICD Codes of (390-459)

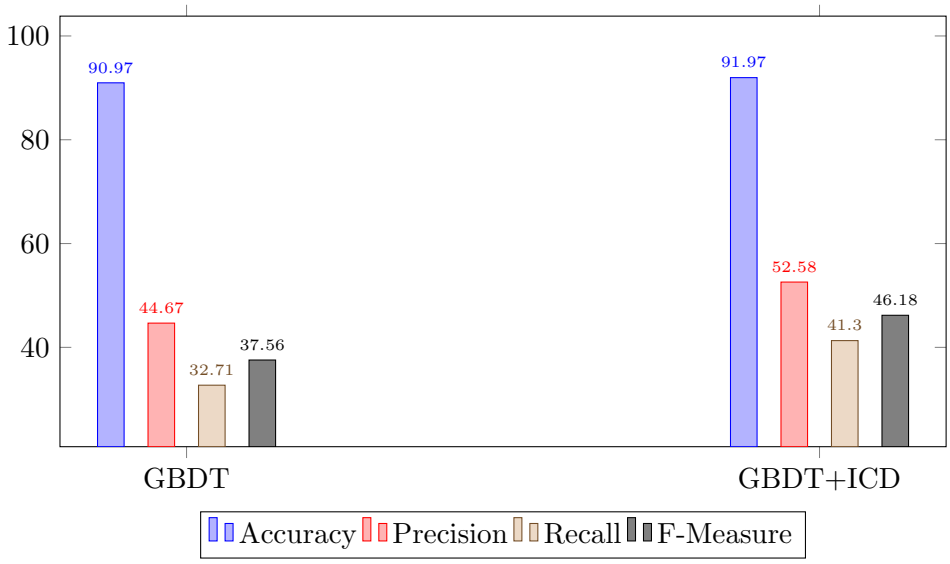


Figure 5.9: Compare GBDT with and without Filtering Patients by Specific ICD Group of (390-459)

11,272 patients, there are only 942 suffered and 10,330 live (i.e., a ratio of 92:8).

5.4.2 Filtering the Group of the Highest Mortality Occurrences

The ICD group of (390-459) has the highest mortality occurrences, as shown in Figure 5.8. From 11,272 patients, there are only 942 dead and 10,330 live. However, this group includes different diseases. Therefore, we look further for smaller groups of ICD codes of diseases that have a higher similarity. Look closer to the ICD codes group of (390-459); the highest mortality rate is on (428-434), which are 437 patients. The smaller and similar classification group is Other forms of heart disease (420-429) and Cerebrovascular disease (430-438). The group of (420-429) contains 2,945 patients, where 179 suffered, and 2766 survived. The (430-438) group contains 2,194 patients, 440 suffered, and 1754 survived: the ratio is 80:20. The (430-438) group with GBDT produces AUC of 0.857 +/− 0.028. The result in Figure 5.10 shows that the smaller ICD group improves the metrics. We should consider that the (430-438) group has a better ratio of 80:20 regarding suffered to survived patients than the larger group 92:08. Thus, we create a dataset with the same size and ratio 80:20 with random ICD codes and test the GBDT on it. This random ICD codes dataset produces AUC of 0.810 +/− 0.034. The result is presented in Figure 5.11.

Among the whole dataset, the highest mortality rate is in the ICD group of (035-041) that are 592 patients. This ICD group is included in the (001-139) classification group of Infectious and Parasitic Diseases. The small and similar group that (035-041) belong to is other bacterial diseases (030-041)—for

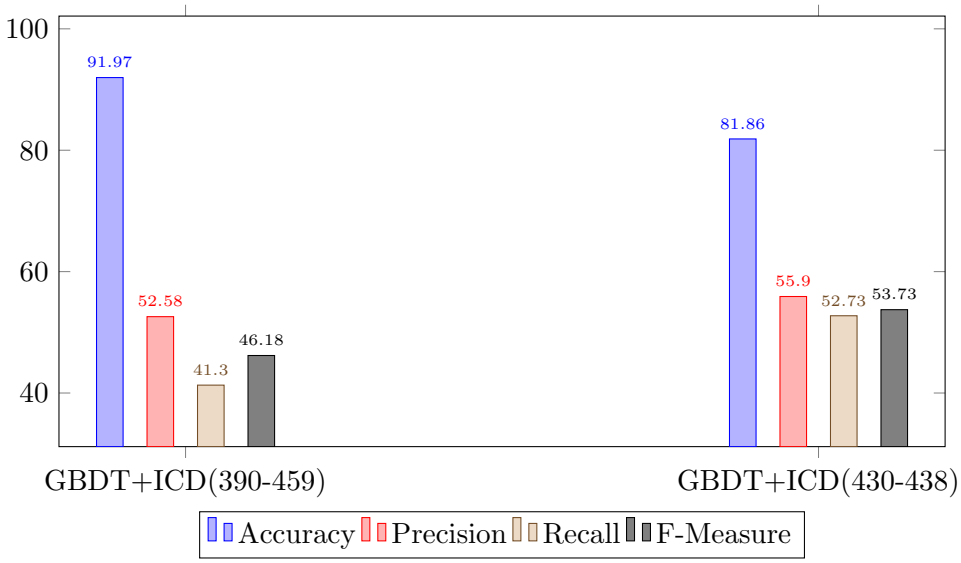


Figure 5.10: Compare GBDT with Filtering Patients by Large ICD Group of (390-459) and by Smaller and Similar Group of (430-438)

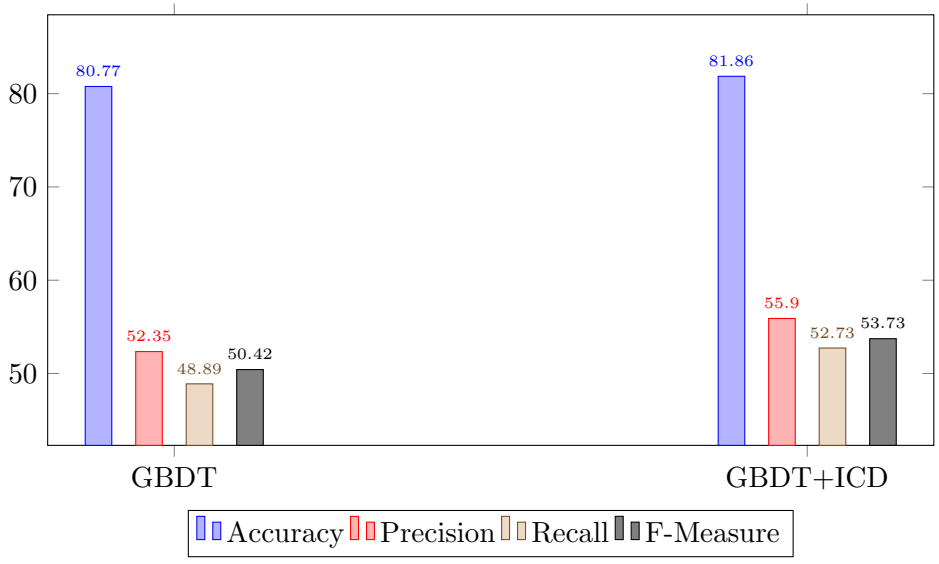


Figure 5.11: Compare GBDT with and without Filtering Patients by Specific ICD Group of (430-438)

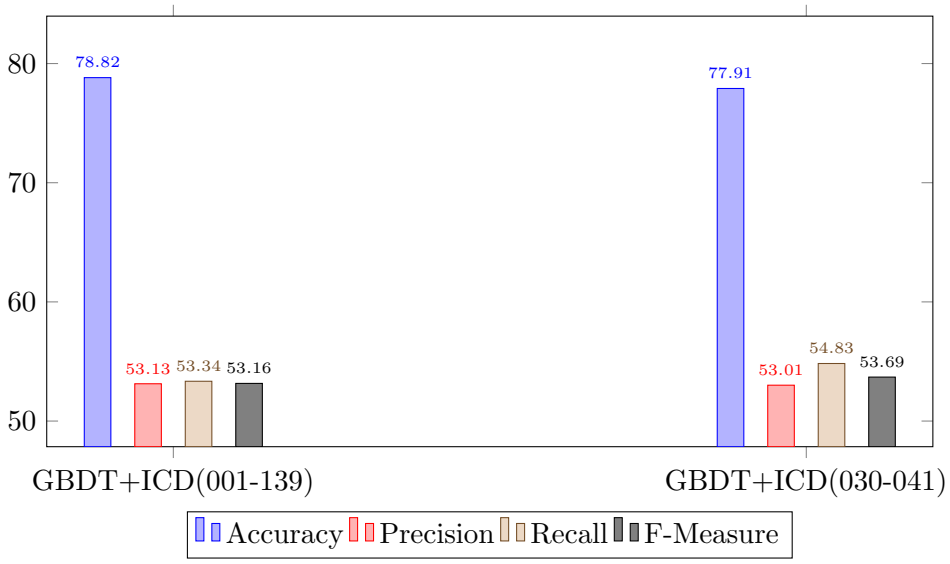


Figure 5.12: Compare GBDT with Filtering Patients by Specific ICD Group of (001-139) and (030-041)

instance, Meningococcal infection and different types of septicemia. Then we select only that group of (030-041) and apply GBDT. The group of (030-041) ICD codes contain 2,517 patients: 593 are suffered and 1924 survived (ratio 76:24). We test GBDT on this dataset and on another dataset of the main classification group (001-139) ICD codes (with the ratio 77:23). The dataset of ICD group of (030-041) produces AUC of 0.798 +/− 0.030, the dataset of ICD group of (001-139) gives AUC of 0.795 +/− 0.022. The result is presented in Figure 5.12. The similar group of diseases (030-041) gives higher AUC, Recall and F-Measure than the main larger ICD group (001-139).

So far, from the previous tests we find that the prediction performance of GBDT is improved by filtering the patients by ICD groups.

5.4.3 Feature Selection after Filtering by the Diagnoses Code

The result can be improved by feature selection based on a specific disease. Thus, finding the features' importance is based on a particular disease. However, that will limit the generalization of the model.

To have a disease dependent prediction of mortality, we execute a feature selection process on a group of patients with a specific disease. The resulted features should improve the accuracy of mortality prediction for that specific disease. Forward Selection and GBDT on normalized data with only the most occurrence ICD group of 390-459 that include 11,272 patients selects 16 attributes: urine_6h, urine_18h, urine_24h, rr_24h_max, spo2_6h_max, temperature_6h_max, mbp_6h_min, spo2_24h_min, glucose_min, glucose_max, bun_max, age, vent, gcs,

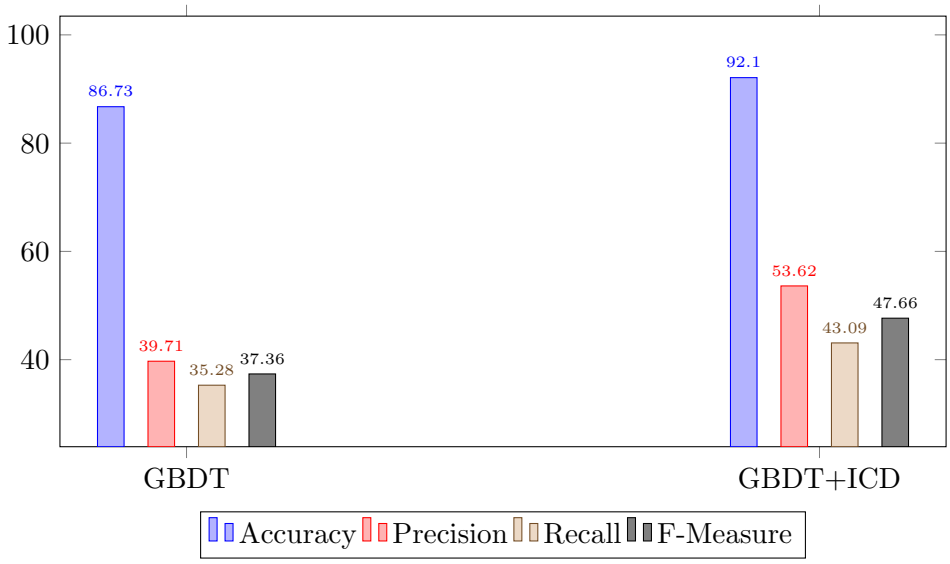


Figure 5.13: Compare GBDT and Forward Selection with and without Filtering Patients by Specific ICD Group of (390-459)

vasopressor, icd_code. Result AUC 0.889 +/− 0.016.

Recall that the previous GBDT with forward selection without focusing on specific ICD group selected 8 attributes: urine_18h, urine_24h, rr_18h_max, hr_24h_min, mbp_18h_min, sbp_6h_min, bun_min, and ccreatinine_min. AUC equals to 0.752 +/− 0.015. Using specific ICD group with Feature selection improves the all the accuracy metrics. The result is showed in Figure 5.13. However, still the imbalanced problem kept the performance low.

5.4.4 Summary

In summary, we filter the patient by specific ICD code group, either the group that has the highest occurrence in the dataset (390-459) or the group that has the highest mortality percentage (030-041) and (430-438). Those datasets have a different imbalanced ratio. For a fair comparison, we compare GBDT on those datasets with datasets that have the same size and class ratio but with random ICD. We find imbalanced class distribution still affects the prediction performance. In general, implementing the GBDT model on top of those data of patients with specific ICD group improves the prediction performance.

Implementing GBDT on the complete dataset without filtering the ICD gives AUC of 0.865 while implementing GBDT on the dataset of patients with the specific ICD codes (390-459) gives AUC of 0.894. Even though the class ratio is highly imbalanced of the dataset with the specific ICD codes (92:8) than the complete dataset (89:11), the prediction performance is optimized.

Focusing on a smaller group of ICD codes (i.e., for a specific disease) from

the main classification list of that ICD codes group gives a higher prediction of patients with a risk of mortality (i.e., higher sensitivity or recall). For instance, when we implement mortality prediction for patients with ICD codes (430-438), which is a smaller group inside the main list of (390-459) codes, the accuracy metrics improved. Thus, the smaller the classified group of ICD, the more similar the patients are, which leads to a higher prediction of mortality risk.

The practical application of this approach can be achieved by providing the ICD code of a patient. Then, this patient will be filtered to the other patients with ICD codes that belong to the same group of this patient's ICD. Finally, the GBDT model will be applied to that group of patients with similar ICD codes to predict the mortality risk of this patient.

6

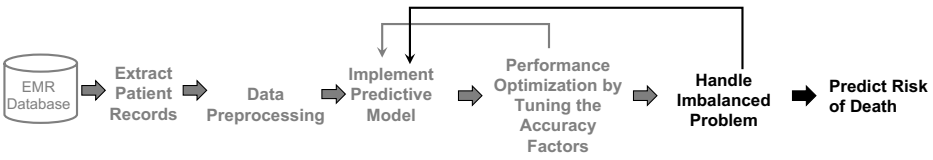
Handle Imbalanced Classes

This chapter discusses the imbalanced data problem. It presents under-sampling and oversampling approaches to handle this problem. It shows a detailed practical application of different methods of each approach. Moreover, the effective clustering-based under-sampling method proposed in the thesis to solve the imbalanced class distribution is described and applied. Furthermore, it applies clustering-based under-sampling to handle the imbalanced data after filtering the patients by the diagnosis codes.

Contents

6.1 Scope of the Chapter

In this chapter, we will represent the crucial step of handling the imbalanced problem. Different methods for handling this problem are defined and practically tested with the predictive ML model.



6.2 Overview

A dataset with one class much more frequent than the other is called *imbalanced dataset* or *a dataset with imbalanced classes* [60]. When χ is an imbalanced dataset, χ_{min} and χ_{maj} are the subsets of the minority and majority classes, respectively; the balancing ratio (BR) of the dataset χ is calculated by Equation 6.1 [46]:

$$BR_{\chi} = \frac{\mid \chi_{min} \mid}{\mid \chi_{maj} \mid} . \tag{6.1}$$

The $\mid \chi_{min} \mid$ and $\mid \chi_{maj} \mid$ are the number of the minority and the majority class instances, respectively. The smaller the balancing ratio is, the more imbalance the data gets. The imbalance ratio (IR) of the dataset χ is the opposite of the balanced ratio. It is calculated in the following Equation 6.2 [54, 3]:

$$IR_{\chi} = \frac{\mid \chi_{maj} \mid}{\mid \chi_{min} \mid} . \tag{6.2}$$

As an example of imbalanced data, predicting spam on a dataset where the amount of non-spam emails are larger than spam ones. This imbalanced problem our dataset has and any ICU real dataset for mortality will have is where the proportion of positive cases (mortality cases) is smaller than the negative cases (survival cases). This unbalanced dataset problem plays an important role in performance. It causes bias to the prediction model towards the more common class (i.e., the majority class) and low performance in predicting the target class (i.e., the minority class). However, in reality, it is a normal situation to occur where the events in a dataset often have an un-similar or different frequency. Our used dataset of MIMIC-III has this imbalanced class distribution problem with the balancing ratio of the two classes instances $BR_{\text{MIMIC-III}} = 0.11$, and the imbalance ratio of $IR_{\text{MIMIC-III}} = 8.9$. The total death cases in our data

are 3,661 out of 32,635. Thus, the dataset consists of 89% from the majority class instances, while only 11% from the minority class instances.

The approaches to handle this problem are categorized into three categories (according to Galar *et al.* [20]): algorithm level approaches, data level approaches, and cost-sensitive learning methods. The data level approach covers the data sampling methods (to balance the dataset), which can further be divided into over-sampling and under-sampling methods. The algorithm level approach develops an algorithm that adapts to the characteristics of the imbalanced data. The cost-sensitive learning method is a hybrid of both data and algorithm level approaches with different classification costs of the classes.

The data level approach to overcome the imbalanced data problem is the commonly used one with low risk. This approach is sampling the data to create a new training dataset by balancing the minority and majority classes to overcome the problem of imbalanced class distribution. Resampling techniques are over-sampling, under-sampling, or a hybrid approach of them. Hoens *et al.* [30] discuss several techniques to overcome this problem of imbalanced data. Moreover, Batista *et al.* [6] give a study of several methods to overcome the imbalanced dataset problem, e.g., the SMOTE method is invented by Chawla *et al.* [11]. Under-sampling removes samples from the majority class that might discard useful information; this is why it has to be done carefully. Thus, we implement a clustering-based method for under-sampling that carefully selects the representatives of the majority class from the clusters.

The ratio of the classes has to be managed to balance the data by using the sampling methods. Thus, we re-implement the ML models and the previous tests of feature selection on both normalized and un-normalized data (from Chapter 5) with sampling the dataset. We will test different sampling methods (over-sampling and under-sampling).

6.3 Data Under-sampling Approaches to Handle Imbalanced Classes

Under-sampling refers to the fact that only some instances of the majority class are chosen for the training data set. We re-sampled the imbalanced data to the ratio of 1:1 by using two different approaches. In the both approaches, we select the complete instances of the minority class and use an approach to under-sampling the majority class. The two approaches to under-sampling the majority class are random under-sampling and K-means clustering based under-sampling.

6.3.1 Random Under-sampling

The first approach for the 1:1 ratio of a balanced dataset, we select all the instances of the minority class (positive class) and we randomly select from the

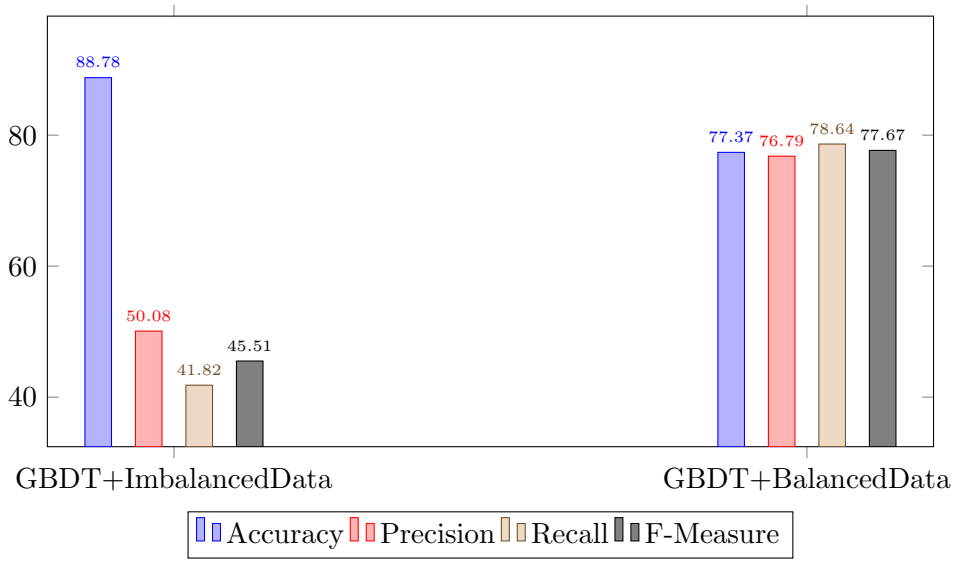


Figure 6.1: GBDT with Balanced Dataset and with The Imbalanced Dataset on Normalized Data

majority class the same instance number of the minority class. This is considered to be a random under-sampling of the majority class.

We test the random under-sampling of the majority class with the 1:1 ratio, first of all, with the models without the feature selection. We want to compare the effect of balancing the classes on the GBDT; thus, we compare with the original data and with the balanced data. We select the fully normalized data (i.e., also the normalized ICD codes) because this is the dataset that we used in the previous test. The GBDT with balanced dataset results AUC is 0.854 +/− 0.016 while GBDT with the imbalanced dataset produces 0.859 +/− 0.009. The result is represented in Figure 6.1.

We find that the balanced training dataset significantly improves Precision, Recall, and F-Measure. The accuracy metric should be ignored because we know that the accuracy of the imbalanced dataset is fake (because of the problem of bias accuracy to the majority class, which has a higher occurrence).

Next, we test feature selection approaches with the two models on the balanced dataset 1:1. Moreover, we test both the un-normalized dataset and the normalized dataset.

The Forward Selection on the LR with balanced dataset of 1:1 ratio on un-normalized data selects: urine_12h, urine_18h, temperature_6h_min, potassium_min, bun_max, creatinine_max. The result AUC is 0.720 +/− 0.023. With normalized data LR selects 17 attributes: urine_12h, urine_18h, hr_6h_max, hr_6h_min, mbp_6h_min, rr_12h_min, rr_6h_min, temperature_6h_min, hematocrit_min, bicarbonate_min, bun_min, bun_max, age, vent, gcs, vasopressor, icd_code. The result AUC is 0.783 +/− 0.022. The results of these tests are

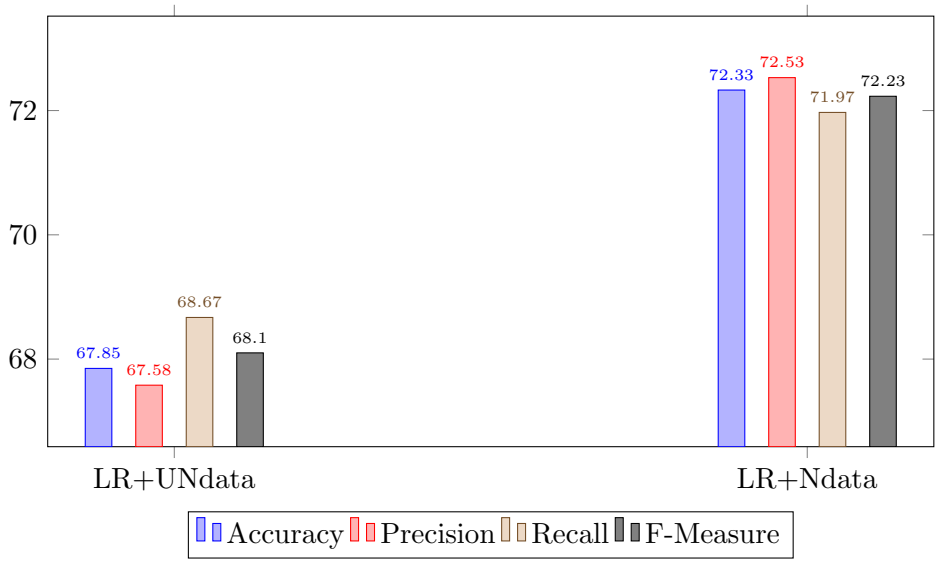


Figure 6.2: Forward Features Selection with Balanced Dataset and LR

summarized in Figure 6.2. Forward selection with LR on the normalized dataset takes 7 minutes and 58 seconds while on the un-normalized dataset takes 5 minutes and 42 seconds. Even though with normalized data it takes longer to finish, the performance is significantly improved. In comparison with un-normalized data the normalized data makes the AUC improved by 8.38% and the Recall improved by 4.69%.

The Forward Selection on the GBDT with balanced dataset of 1:1 ratio on normalized data selects 10 attributes: urine_18h, hr_6h_max, sbp_6h_max, rr_18h_min, spo2_18h_min, bun_min, age, vent, gcs, icd_code. It produces a good AUC of 0.837 +/− 0.017. In comparison to the same previous test setting with LR the GBDT improves the AUC by 6.67% and the Recall by 15%. With un-normalized data GBDT selects 14 attributes: urine_12h, hr_18h_max, hr_6h_max, mbp_6h_max, rr_24h_min, hematocrit_max, potassium_max, sodium _min, bun_min, creatinine_max, age, vent, gcs, icd_code. It gives AUC of 0.835 +/− 0.014. The results of these tests are summarized in Figure 6.3. The normalized data gives higher Recall by 2.96%.

Even though that GBDT has the highest Recall and higher metrics values still, the LR has a better trade-off between the performance metrics (i.e., the Recall, Precision, and F-Measure).

In the previous chapter 5 in feature selection Section 5.3, we find that feature selection on an imbalanced normalized dataset gives higher prediction performance than on imbalanced un-normalized dataset. Here we test forward selection on a normalized and un-normalized balanced dataset. We observe the same result the normalized dataset provides higher prediction performance than

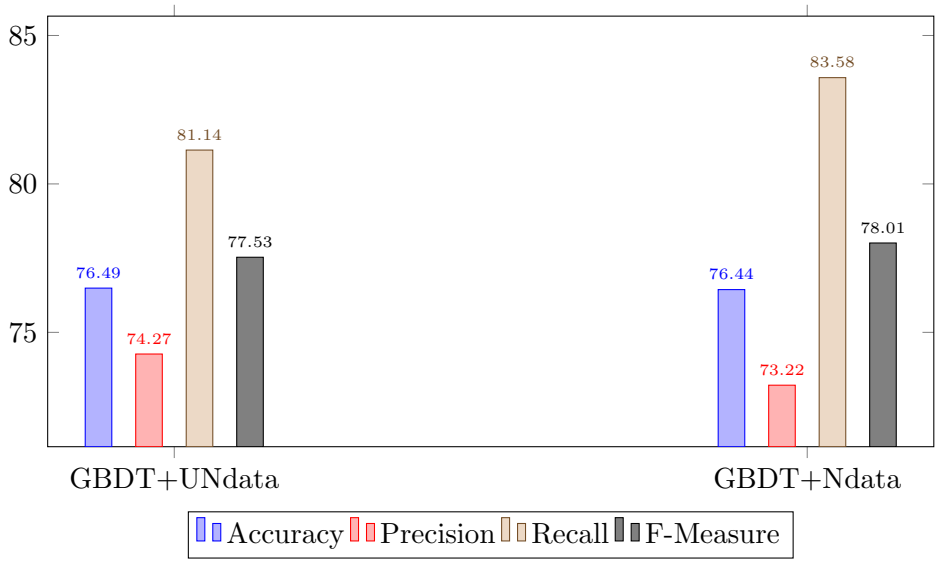


Figure 6.3: Forward Features Selection with Balanced Dataset and GBDT

un-normalized data. Therefore, in the following tests, we will continue our performance optimization tests only on the normalized dataset.

Now we want to test the Backward Elimination on the LR and GBDT with balanced dataset of 1:1 ratio on normalized data. LR eliminates 4 attributes: hr_18h_max, sbp_12h_min, sbp_24h_min, spo2_24h_min. It has AUC 0.794 +/− 0.017. It takes 15 minutes and 20 seconds. The GBDT eliminates 1 attribute: sodium_min with AUC is 0.852 +/− 0.012. It takes 2 hours and 15 minutes. The both results are visualized in Figure 6.4. The GBDT still the winner.

In conclusion, balancing the dataset significantly improved the predictive performance for the model itself and the features selection. For instance, in the case of Forward Selection with GBDT with the normalized dataset, balancing improved the Recall by 81.27% and the AUC by 10.6986%. The same with LR with the normalized dataset, balancing is fully improved the AUC.

Moreover, using a balanced dataset affects the selected features in the model itself and in comparison to the other model too. In the Forward Selection with normalized data, balancing the dataset makes the selected features similar between the models. The selected features by GBDT in Forward Selection or Backward Elimination without the balanced dataset are more different than with the balanced one.

Furthermore, we can summarize from all of the previous tests that the normalized data even though it requires longer run time, it gives higher predictive performance than the un-normalized one. The same with the GBDT, it requires long run time but in general it provides higher predictive performance than LR.

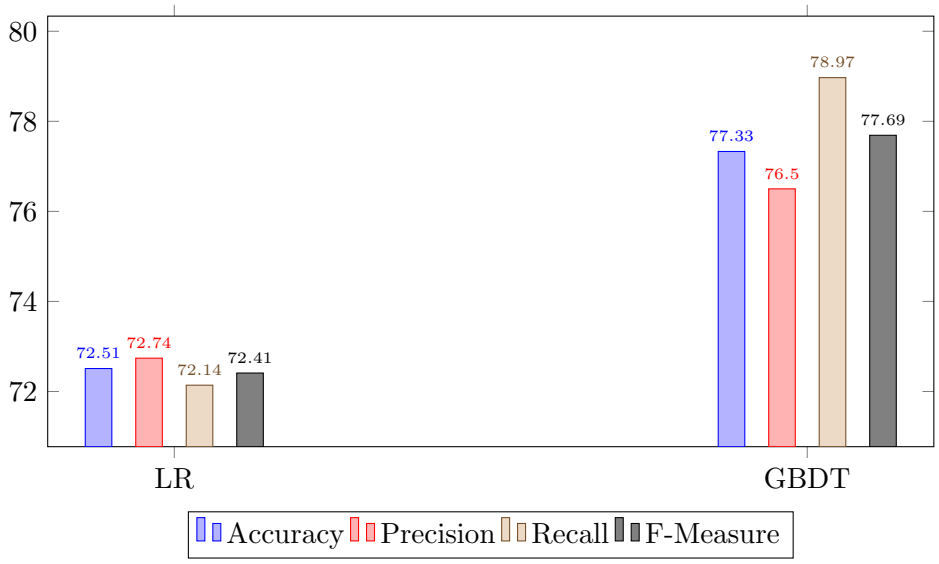


Figure 6.4: Backward Elimination Features Selection with Balanced Dataset and LR and GBDT on Normalized Data

6.3.2 K-Means Clustering-based Under-sampling

The second approach for under-sampling the dataset is K-means clustering under-sampling. K-means clustering under-sampled the majority class (negative class) into k clusters. The initial cluster centers are determined by using K-means++ algorithm [4]. To balance the distribution of the classes to a 1:1 ratio, we need to select from the majority class clusters the same instance number of the minority class. Different approaches are tested to select the instances from the clusters.

The k value is either =3,661 (the size of the minority class) or the optimal k value. To find the optimal k, we test different amounts of clusters k = 10, 100, 500, 900, and 1,830 (that is, half of the minority class). We evaluate the clusters that are created from different amounts of k of clusters by the Davies Bouldin (DB) index [13]. The DB index is a ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within a cluster is the standard deviation of the distance between the cluster center (centroid) and all the samples of this cluster. The separation between two clusters is the distance between their centroids (see Equation 6.3 from [13]).

$$R_{ij} = \frac{S_i + S_j}{M_{ij}} \tag{6.3}$$

Cluster similarity measure or cluster separation measure R_{ij} compute the average similarity between cluster i and j. S_i is the dispersion of cluster i and S_j is the dispersion of cluster j. M_{ij} is the distance between the vectors of

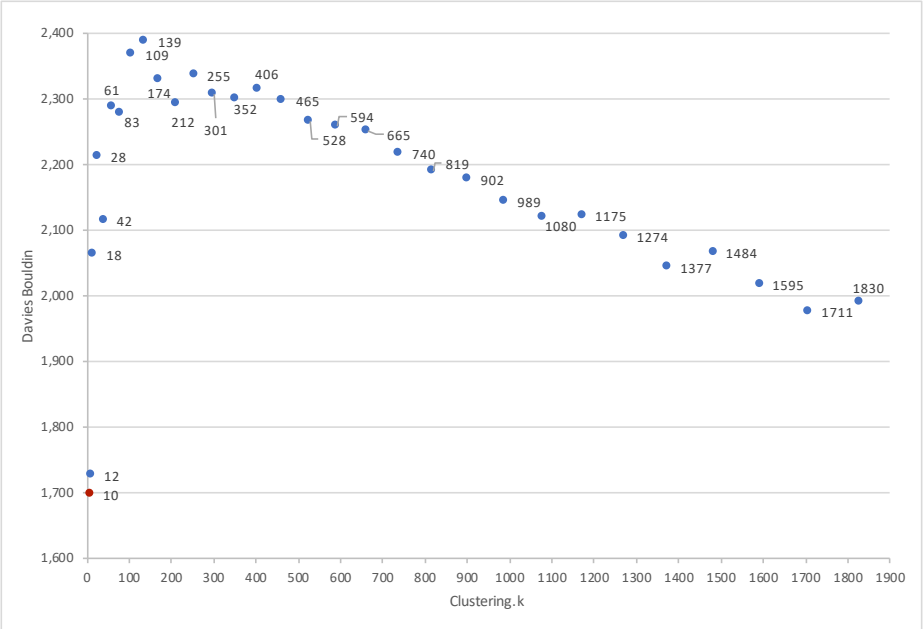


Figure 6.5: Davies Bouldin Index for Different Numbers of Clusters.

cluster i and j. Then, the average of the similarity measures $\bar{R}$ between the clusters is calculated as followed (see Equation 6.3) [13].

$$\bar{R} = \frac{1}{N} \sum_{i=1}^{N} R_i \tag{6.4}$$

When N is the number of the clusters, R_i is equivalent to the maximum of R_{ij} when $i \neq j$. The average of the similarity measures $\bar{R}$ between the clusters is the DB index. Minimizing this average of the similarity of the clusters (i.e., the lowest DB index) produces the most proper clustering. The result of the DB index for different k values from 10 to 1,830 is shown in Figure 6.5. We find that k=10 has the lowest (i.e., the best) DB index.

The selection of the majority class representatives is a critical point during the under-sampling process. The approaches for selecting the majority class representatives from the K-means++ clusters are:

- Cluster centroids: The number of the clusters (the k value) equals the size of the minority class. Then, only the centroids of the clusters are used as representatives for this class.
- Random sampling: The cluster number k=10; recall that 10 was approved to be the optimal cluster number for our data. Equally sized subsets are randomly selected from each cluster where the size of these subsets in

total equals the size of the minority class. In this approach, we use a small number of clusters k=10 than the k=size of minority class to reduce the run time of clustering. Moreover, an equal subset from each cluster represents the clusters equally.

- Top1 centroids' nearest neighbor: The number of the clusters k equals to the size of the minority class. For each cluster, we calculate the distance between the cluster's centroid and the cluster's points (by Euclidean distance). Afterward, from each cluster, we select the Top1 nearest neighbor to the cluster centroid. In the first approach, we select cluster centroids as representative of the majority class. The cluster centroids are artificial records since they are defined by averaging the data points of that cluster. In this approach, we select real patient records rather than the artificial ones of the centroids.

- TopN centroids' nearest neighbors: The number of clusters equal to the optimal K=10. After that, the Euclidean distance is calculated between the centroids and the cluster points. Then from each cluster, we select the TopN nearest neighbors to the cluster centroid. Where TopN = size of the minority class/number of clusters k. In this approach, we use a small number of clusters k=10 than the k=size of minority class to reduce the run time of clustering. Moreover, the clusters are equally and appropriately represented by the centroids' neighbors.

The resulting under-sampled data of the majority class from these different approaches are combined with the minority class (positive class) before starting the classification. We compare the K-means++ under-sampling method with the different approaches for selecting the majority class representatives on GBDT by 10-fold cross-validation (see Figure 6.6) .

From the results in Figure 6.6, we find that the approach of selecting the TopN nearest neighbors to the centroid, in general, outperforms all the other approaches. Only the recall of another method (selecting the cluster centroids) is higher by 3.78%. Predicting the positive class of patients at risk of death is crucial in our case – and hence we consider the approach with the highest recall the most appropriate for us. Nonetheless, the approach of selecting the TopN nearest neighbors to the centroid (with a small k) is a great competitor to the method of selecting the cluster centroids. It has a good balance between the accuracy metrics and a short computational time (a few seconds) in comparison to the long time (more than 7 hours) of the approach with the highest recall.

Unexpectedly, the approach that combines K-means++ and random sampling outperforms the method of choosing the Top1 nearest neighbors. The reason might correspond to the optimal k value that the hybrid method of K-means++ and random sampling used.

We compare the random under-sampling method (Section 6.3.1) with the best approach of K-means under-sampling. The result is presented in Figure 6.7

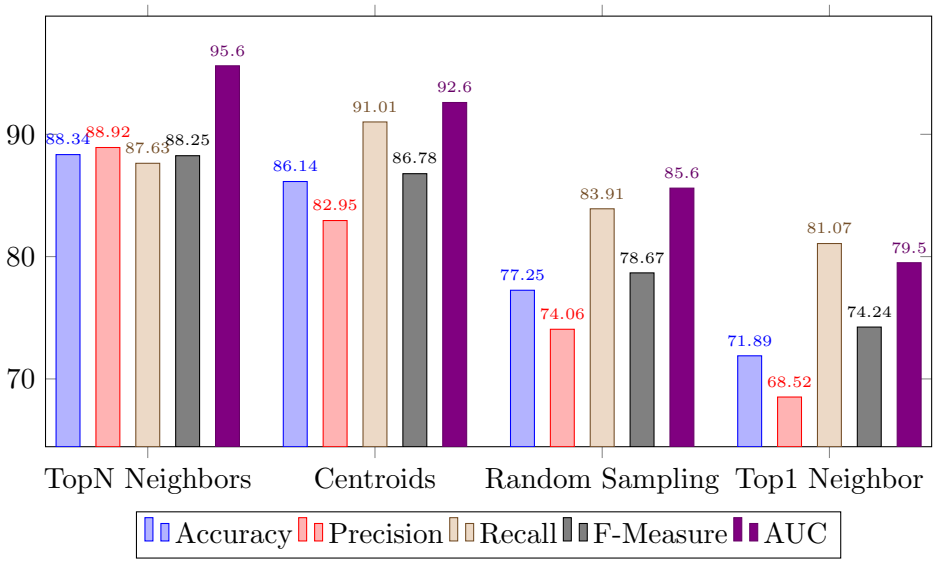


Figure 6.6: Comparison of K-means++ Under-sampling with Different Approaches for Selecting the Majority Class Representatives.

and Figure 6.8. In general, we find K-Means clustering based under-sampling for the majority class totally outperforms the random under-sampling. The K-means under-sampling improves all the accuracy metrics. The accuracy improved by 14%, and the recall improves with K-means by 11.43%. The K-means improves the AUC by 11.94% and the AUPRC by 10.44%. The reason for these performance improvements could return to the main disadvantage of random under-sampling, where we lose potentially relevant information from the omitted samples. However, by the k-means cluster, we are able to keep more relevant information (i.e., more variety) of the majority class.

One drawback of K-means is the long run time with large k (i.e., when k= the size of the minority class). The k-means clustering of the whole majority class of 28,974 takes 7 hours and 11 minutes. However, with small k (e.g., the optimal k value =10) the clustering run time was only 2 seconds (see Table 6.1).

Moreover, another way to improve the run time of K-means is to apply K-means only on the dataset of the features that are selected by forward feature selection on GBDT on the normalized dataset. This dataset contains only eight attributes. K-means with k=size of minority class on this dataset takes 1 hour and 10 minutes and 13 seconds. Then we use this dataset (i.e., only the centroids 3661 samples). The result metrics are accuracy: 86.46%, precision: 84.69%, recall: 89.02%, f_measure: 86.79%, AUC: 0.944 +/- 0.009, and AUPRC: 0.939 +/- 0.008. In comparison to K-means on the whole majority class, this approach is competitive. It saves more than 50% of the runtime by reducing from 7 hours to only 1 hour (see Table 6.1). The accuracy is minor affected by 2.5%, and the recall by 1.4% and AUC and AUPRC is almost not affected.

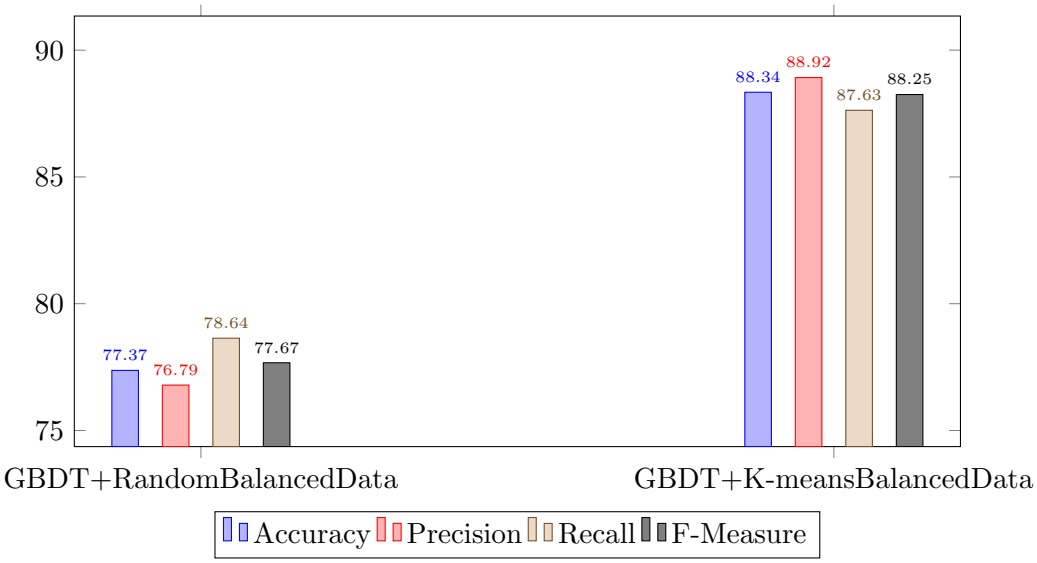


Figure 6.7: Compare GBDT with Random Under-sampled Balanced Dataset and with The K-means Under-sampled Balanced Dataset on Normalized Data

Figure 6.8: The AUC and AUPRC of The GBDT with Random Under-sampled Balanced Dataset and with The K-means Under-sampled Balanced Dataset on Normalized Data

K-means Clustering Approach	Time Cost
k=size of minority class	7:11:00
k=the optimal value (10)	00:00:02
k=size of minority class and Subset of Features (by Forward selection)	1:10:13

Table 6.1: Compare Time Cost of Different K-means Clustering Approaches

In conclusion, the k-means clustering for under-sampling the majority class improves the predictive accuracy more than the random under-sampling of the majority class. The main drawback of K-means++ -its long run time- can be improved. The approaches we use to improve the run time save much of the time and improve the accuracy. K-means on dataset with the forward selection features is competitive to the k-means on the complete set of features.

The cluster number k and the approach to select the majority class representatives (from the K-means++ clusters) are crucial influencers on the model accuracy. Selecting the nearest neighbors to the centroids works best with a small k (i.e., we choose TopN). Whereas, with a significantly larger k value selecting the centroids is the better choice than choosing the Top1 nearest neighbor.

In general, all the approaches that used K-means++ clustering-based under-sampling with different methods for selecting the majority class representatives are significantly improving the prediction accuracy of the GBDT on imbalanced data (in comparison to Figure 4.21).

6.4 Data Over-sampling Approaches to Handle Imbalanced Classes

Another approach to solving the problem of imbalanced class distribution is oversampling the minority class. Over-sampling is generating new samples from the minority class. We re-sampled the imbalanced data to the ratio of 1:1 and other ratios by using SMOTE up-sampling approach. In the different ratios, we select the complete instances of the majority class and use different up-sample sizes of SMOTE approach to over-sampling the minority class.

6.4.1 SMOTE Over-sampling

For over-sampling the majority class we use Synthetic Minority Over-sampling Technique (SMOTE) [11]. In SMOTE the minority class is over-sampled by generating synthetic examples from each sample in the minority class. The k nearest neighbors of each sample in the minority class are defined. Based on the required over-sampling size, a set of neighbors from the k nearest neighbors are randomly chosen. Then the synthetic samples from a sample are generated by first calculating the difference between the feature vector of the selected sample and its nearest neighbor. Second, multiply this difference by a random number

(between 0 and 1). Then, add the resulted number to the sample's feature vector.

Applying oversampling with cross-validation has to be done carefully; otherwise, it is suspected to overfitting. In under-sampling, the minority class instances (the critical cases the patients with risk) are not changed; only the majority class instances are reduced. However, with oversampling, the minority class samples are duplicated by similar instances. This causes similar instances occurring in the training and the testing of the cross-validation. Thus, the oversampling has to be done only in the training dataset.

Blagus and Lusa [7] conducted a practical study of two ways to apply cross-validation with different methods of oversampling and under-sampling. They stated that there are two ways to use under-sampling and oversampling with cross-validation either before cross-validation or during the training process of cross-validation. They compare the results of the two approaches. They find that the model prediction performance with oversampling is significantly different between the two ways of applying the oversampling with cross-validation. However, the performance was identical between the two ways of applying the under-sampling with cross-validation.

Thus, here we will test the two different approaches of applying the oversampling method (SMOTE) with the cross-validation for both models (LR and GBDT) on the normalized dataset. There are some parameters of the SMOTE that have to be set: the number of neighbors, nominal change rate, round integer, equalize classes, normalize, and up-sampling size. Furthermore, the feature optimization of SMOTE is applied for the two approaches of using it with cross-validation. The result of the two ways of implementing oversampling with cross-validation are presented in the following:

- **Over-sampling the complete dataset then apply cross-validation:**

  The minority class in the entire dataset (training and test data) is oversampled then apply the cross-validation. Because of the small size of the minority class, we select a small size of neighbors. The number of neighbors set to k=5. The equalize classes parameter is set to true (i.e., the produced dataset is balanced with classes ratio 1:1). The minority class is over-sampled to equalize the majority class 28,897, where the total dataset is 57,794 patients. The LR with over-sampled data by SMOTE takes 5 minutes and 24 seconds to complete. It results in AUC of 0.810 +/− 0.006. The GBDT takes 10 minutes to finish. It produces AUC of 0.980 +/− 0.001. The result is represented in Figure 6.9. Equalized classes with SMOTE produce a significant performance improvement for GBDT in all the metrics more than balancing the classes by random under-sampling of majority class and having 1:1 ratio. It is clear that so far GBDT outperforms LR. Therefore, for the next tests, we will use GBDT.

  We test the previous test of GBDT and SMOTE with different SMOTE

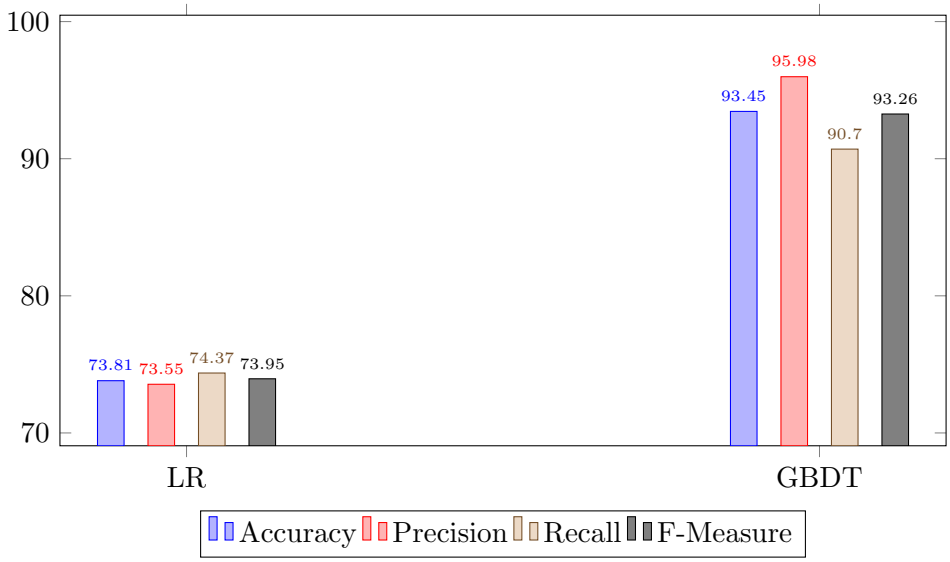


Figure 6.9: LR and GBDT with SMOTE Equalized Classes

parameters. First, we test without round integers (it rounds integer attributes to the next integer) and observe that it gives the same result.

Second, after testing equalize classes in SMOTE, we test different up-sampling size (i.e. different number of samples of minority class), which are 2000, 5000, 10000, and 15000. The over-sample size of 5000, makes the total dataset is 37,548 (dead 8651, live 28897) and gives AUC of 0.937 +/− 0.002. The over-sample size of 10000 produces a dataset size of 42,548 (dead 13,651, live 28897) and gives AUC of 0.960 +/− 0.003. The over-sample size of 15000 produces a total dataset of 47,548 (dead 18651, live 28897) and gives AUC of 0.970 +/− 0.003. The comparison of the metrics result is represented in Figure 6.10.

In the previous test, we used the number of neighbors k=5, so we test parameters optimization of a different number of neighbors from 5 to 50 together with testing different up-sampling size 2000 to 30000—the round integer set to false and the change rate to 0.0. K=5 and up-sampling size of 30000 lead the highest accuracy. The resulted metrics are accuracy 93.96%, AUC 0.982 +/- 0.001, precision 96.89%, recall 91.71% and f_measure: 94.23%. The result is summarized in Figure 6.12. It shows that regardless of the neighbors' size, the larger the up-sampling size is, the higher the accuracy is.

Then, we test different nominal change rate of 0.0 to 1.0 with different up-sampling size 5000 to 30000 and k=5 (see Figure 6.13). The best AUC 0.983 +/- 0.002 is reached by a change rate of 0.7 and up-sample size of 30000. The resulted metrics are accuracy: 93.99%, precision: 96.73%,

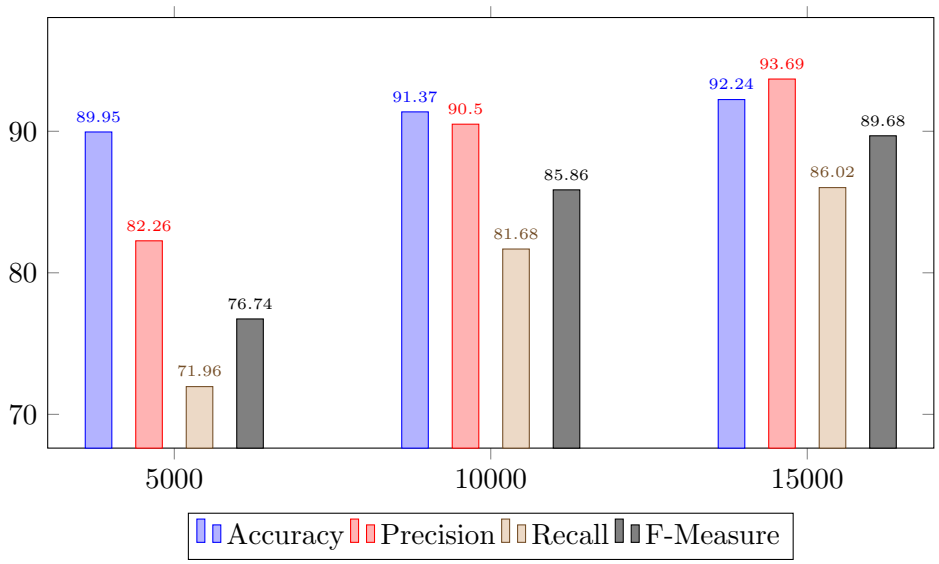


Figure 6.10: GBDT with SMOTE Different Over-sampling Sizes

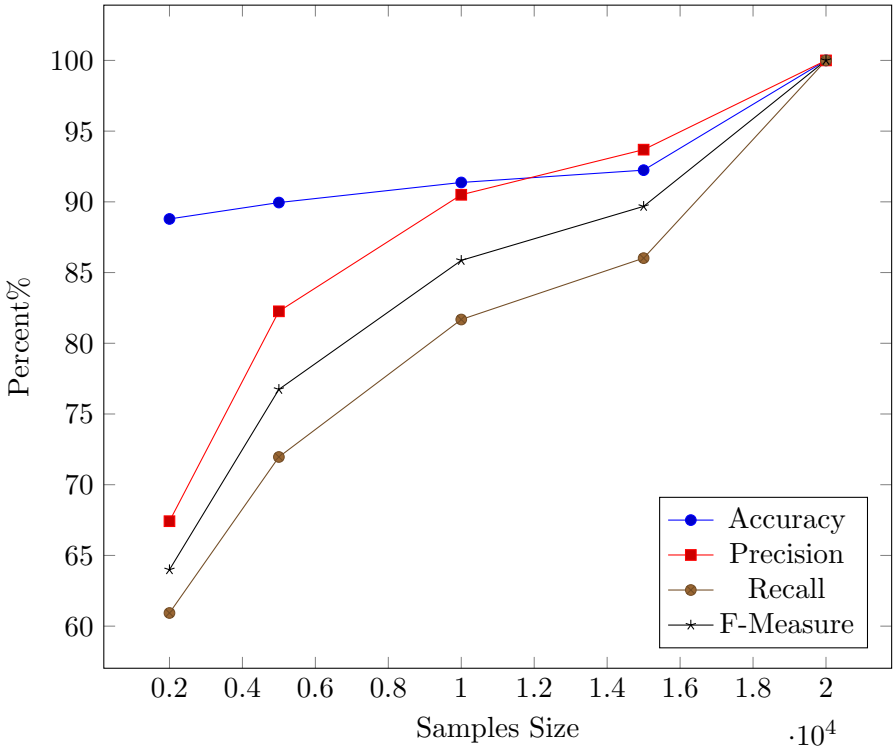


Figure 6.11: GBDT with SMOTE Different Over-sampling Sizes

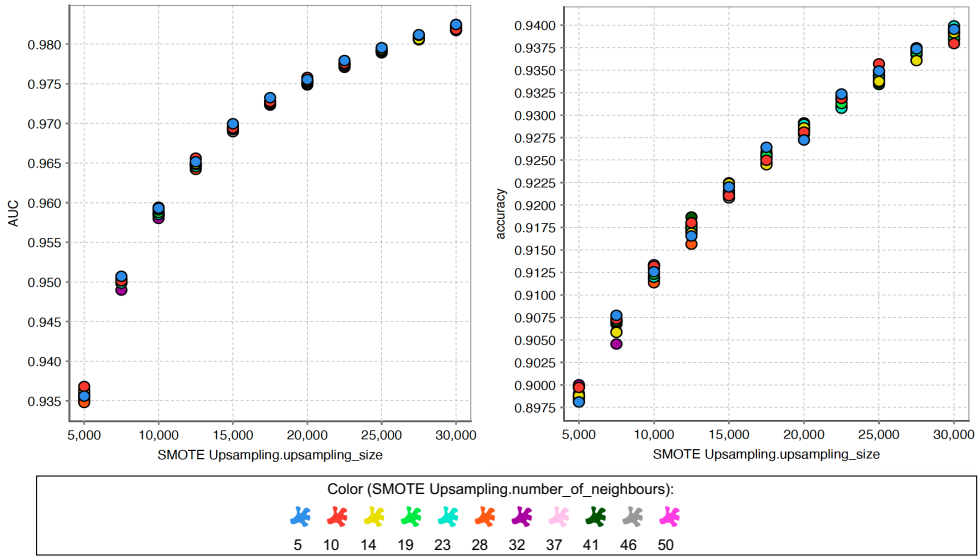


Figure 6.12: Testing SMOTE with Different Number of Neighbours and Up-sampling Sizes to Optimize AUC and Accuracy

recall: 91.92% and f_measure: 94.26%. The larger the up-sampling size, the higher the accuracy is, and the higher the AUC is (see Figure 6.14).

I test the equalize classes parameter—either true or false value. The equalize classes give higher accuracy by only 0.56%. The equalize classes give 93.44% and not equalize classes has 92.92%. The round integer parameters false is the best.

To summarize the finds, the larger the minority class up-sampling size, the higher the accuracy metrics are (overfitting). The change rate of 0.7 is a good choice for our case. The best performance achievement is with small neighbors' size of k=5 and a large up-sample size of 30000. Equalize classes is also a good decision with SMOTE. This approach of applying cross-validation with SMOTE oversampling causes an overoptimistic error or overfitting. This is what we experienced in the previous tests, where we have optimistic results. Over-optimism occurs because of the similarity between the test and the training sets. Oversampling applied in both sets, which produces some similar patterns that occur in both the training and test sets, causing overoptimism [71]. Moreover, during SMOTE's parameter optimization, we experienced over-fitting. The larger the up-sample size, the more similar data examples added to the training set, which cause over-fitting.

- **Over-sampling the training set at each iteration of cross-validation:** The SMOTE oversampling is implemented at each iteration of the cross-

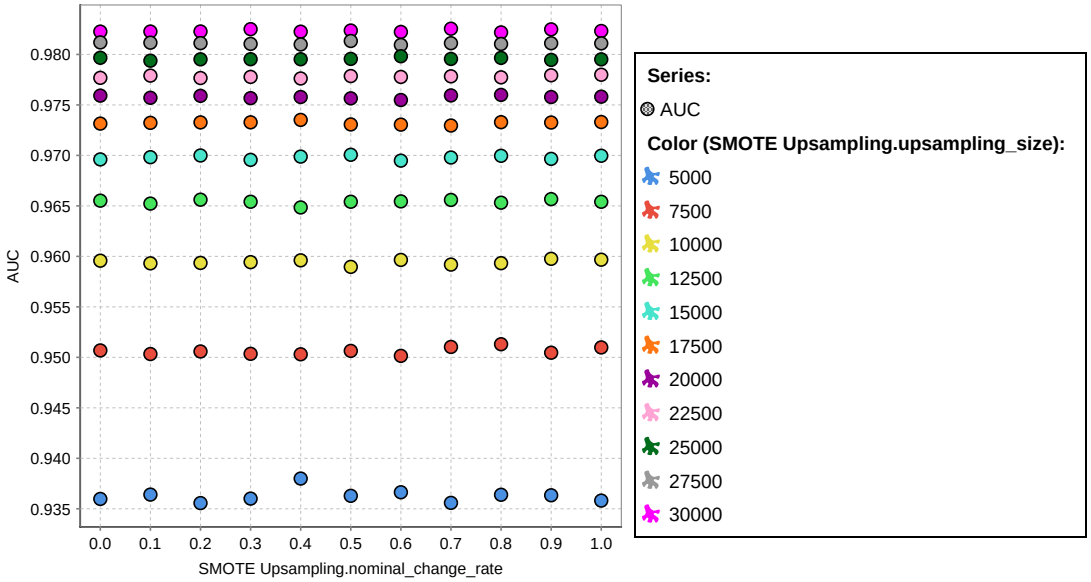


Figure 6.13: Testing SMOTE with Different Up-sampling Sizes and Different Nominal Change Rate to Optimize AUC

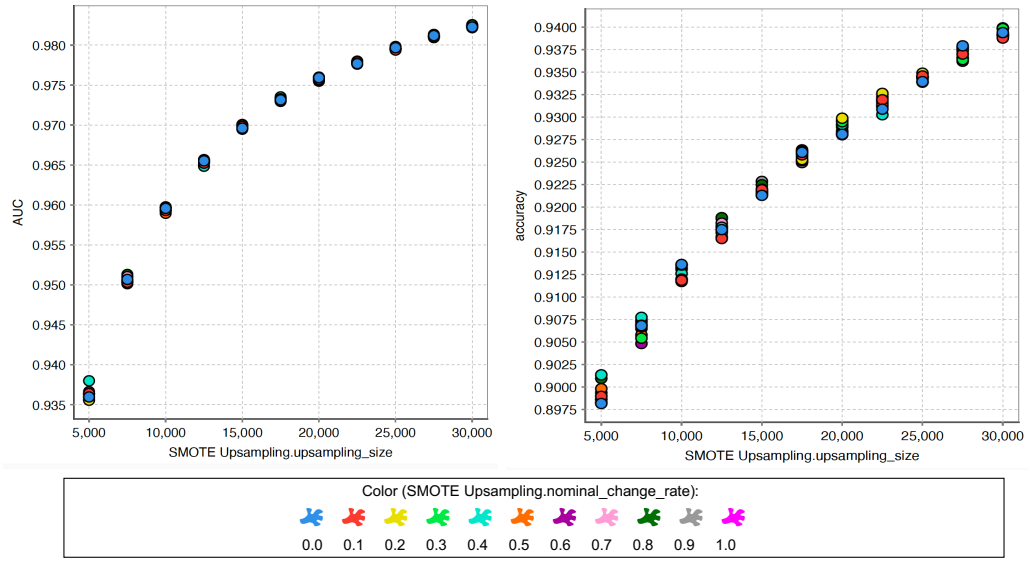


Figure 6.14: Testing SMOTE with Different Up-sampling Sizes and Different Nominal Change Rate to Optimize AUC and Accuracy

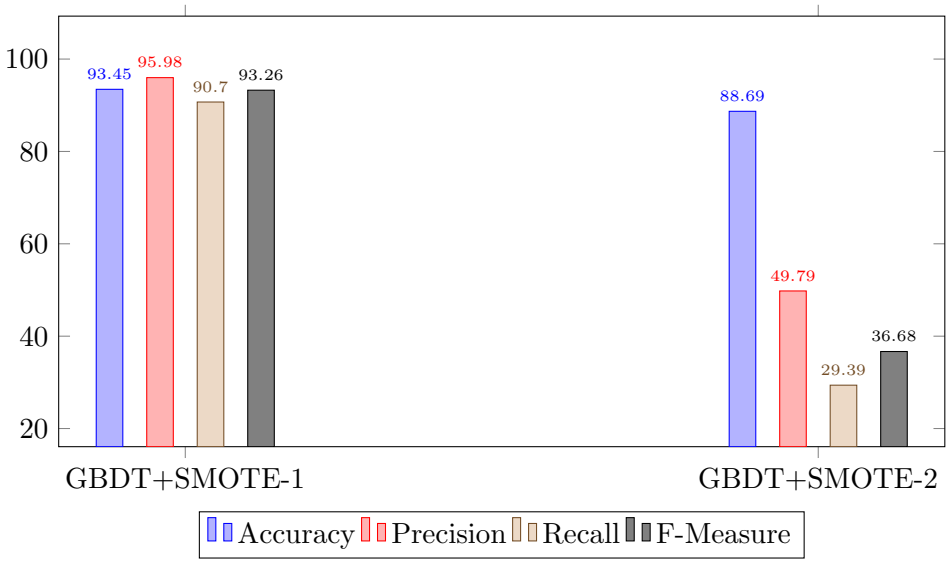


Figure 6.15: GBDT with the Two Approaches of Applying SMOTE with Cross-validation (Equalized Classes)

validation only on the minority class in the training set. Oversampling of the dataset during the cross-validation only on the training set makes a proper evaluation [71]. This approach is computationally costly than the first approach. However, it avoids overoptimism that occurs in the first approach. We test GBDT by cross-validation with implementing SMOTE during the cross-validation in the training process. SMOTE parameters are number of neighbors k=5, change rate = 0.0 and equalize classes. It takes 1 hour and 17 minutes and 25 seconds. The AUC is 0.835 +/− 0.009. We compare the previous approach with this approach in Figure 6.15. Where GBDT+SMOTE-1 is the first approach and GBDT+SMOTE-2 is this approach.

We optimize the parameters of SMOTE with this approach to implement it with cross-validation. For equalized classes (i.e., the same ratio of both classes), we test parameters optimization of a different number of neighbors from 5 to 50 with equalized classes. The round integer is set to false and the change rate to 0.0. k=5 produces the best AUC. Then, we test different nominal change rate with the original rate of 0.0 to 1.0 with equalized classes (i.e., balanced dataset) where k=5. The selected change rate by optimize parameters Grid that produces the highest AUC is 0.4 which gives accuracy: 88.58%, AUC: 0.836 +/- 0.012 , precision: 48.44%, recall: 27.70%, f_measure: 35.17%.

Then, we test GBDT with different SMOTE's up-sampling size (i.e. different number of samples of minority class), which are 5000, 10000, and

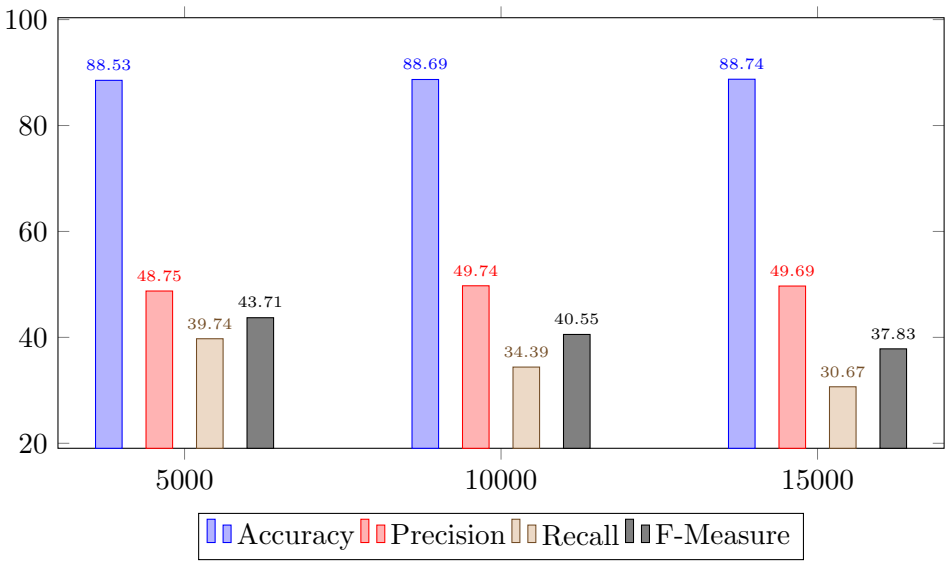


Figure 6.16: GBDT with SMOTE Different Up-sampling Sizes

15000 where k=5 (see Figure 6.16) and the change rate =0.0 (since change rate 0.4 approve the optimal for equalized classes). The resulted AUC are with 5000 the AUC=0.851 +/- 0.010, with 10000 the AUC=0.846 +/- 0.009 and with 15000 the AUC=0.843 +/- 0.010.

Then we test different up-sampling sizes together with different change rates. Since this test is costly, we run a smaller range of values. For the up-sampling from the previous test, we find that 5000 produces the highest AUC, then we test a range of values less than 5000 from 500 to 10000 (500, 2400, 4300, 6200, 8100, and 10000). The change rate ranges from 0.0 to 1.0 and k=5 and round integer is set to false (see Figure 6.18). The best achieved accuracy is with up-sample size 8100 and change rate 0.4. In general, we find that 500 up-sample size gives the highest AUC with all the change rates, while 10000 gives the lowest AUC (see Figure 6.17). Moreover, the best AUC has reached with the up-sample size of 500 samples and a change rate of 0.5, which is 0.860. However, this doesn't make any improvement to the AUC resulted from applying GBDT alone without SMOTE (which is 0.865).

Furthermore, unlike the previous implementation of SMOTE with cross-validation (i.e., applying SMOTE on all the dataset before cross-validation) were increasing the up samples, the AUC increases ((see Figure 6.14), here increases the up samples not increase the AUC (see Figure 6.18).

We test different up-sampling size together with a different number of neighbors. We test the same range of up-sampling values from the previous

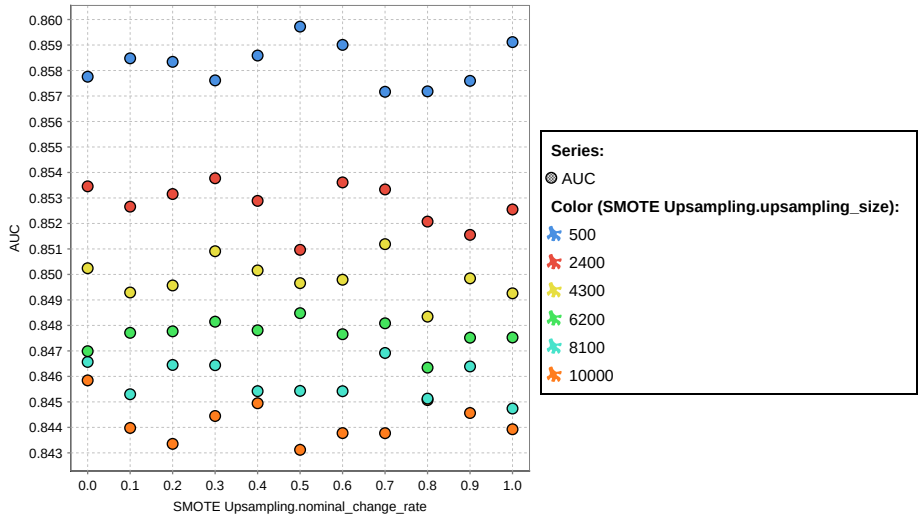


Figure 6.17: Testing SMOTE with Different Nominal Change Rate and Different Up-sampling Sizes to Optimize AUC

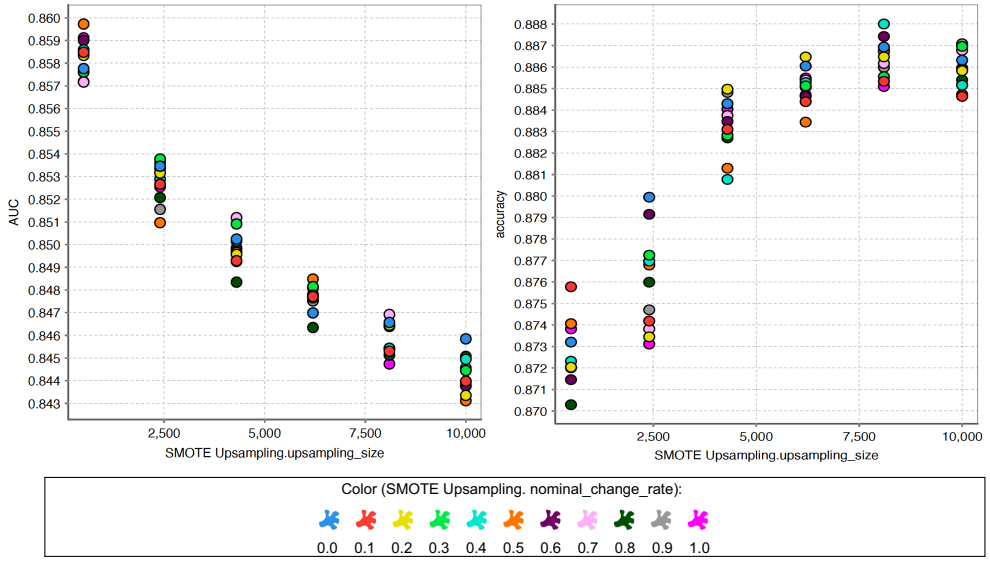


Figure 6.18: Testing SMOTE with Different Nominal Change Rate and Different Up-sampling Sizes to Optimize AUC and Accuracy

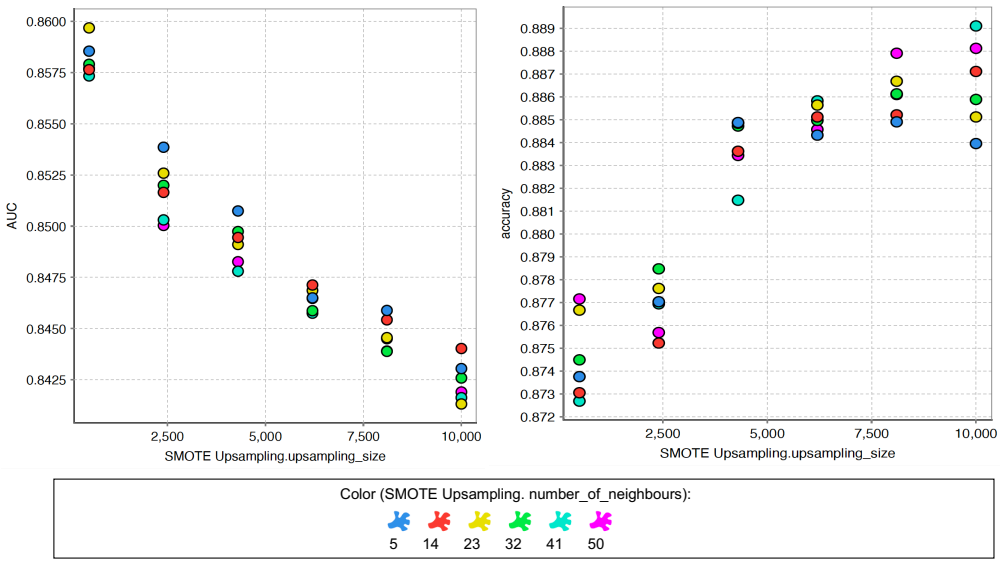


Figure 6.19: Testing SMOTE with Different Number of Neighbors and Different Up-sampling sizes to Optimize AUC and Accuracy

test from 500 to 10000 (500, 2400, 4300, 6200, 8100, and 10000) for the up-sampling. The change rate =0.5. The best AUC reached with upsample size 500 and k=23 (see Figure 6.19) and the best Accuracy with k=41 and upsample size 10000 (see Figure 6.19). Moreover, as the previous test with increasing the size of the up-sample, the accuracy increases (see Figure 6.12), which is explained by the imbalanced ratio is increasing, and the accuracy is not the correct metric to consider. However, increasing the up-sample size does not increase the AUC as the previous approach (see Figure 6.12 and Figure 6.19).

To summarize, applying SMOTE oversampling in each iteration of cross-validation only on the training set does not cause overoptimism results. For instance, optimizing the up-sample size with the change rate increases the AUC with increasing the size of the up-sample in the first approach where SMOTE oversampling applied on both the training and test sets then apply cross-validation (see Figure 6.14). However, optimizing the up-sample size with the change rate does not increase the AUC with increasing the size of up-sample in this approach where SMOTE oversampling applied only on the training set in each iteration of cross-validation (see Figure 6.18). Thus, this approach does not cause over-fitting, but it doesn't optimize the prediction performance—the best achieved AUC 0.860 while the best AUC with standalone GBDT model is 0.865.

We compare our approach (in Section 6.3.2) to oversampling SMOTE method by cross-validation. The result is shown in Figure 6.20. We find that our

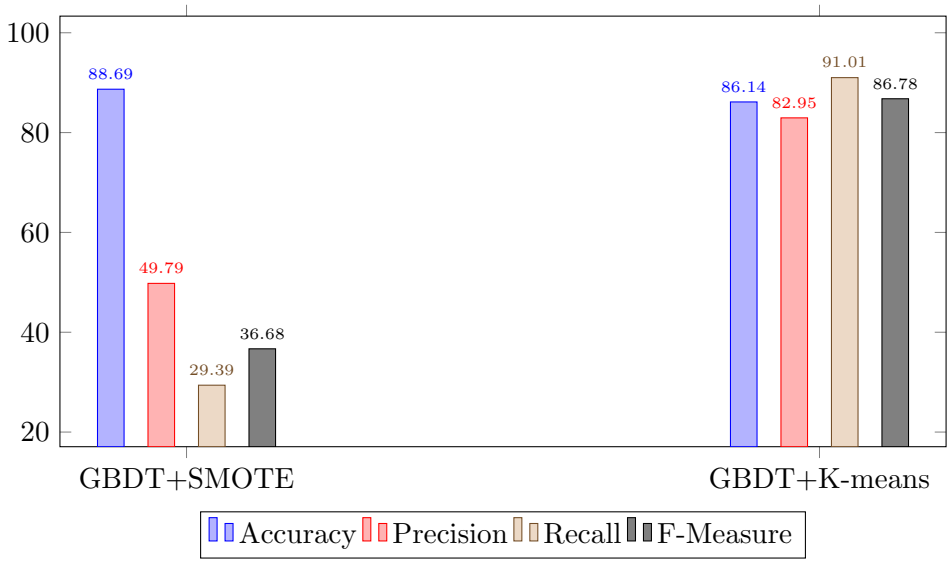


Figure 6.20: GBDT with SMOTE and K-means Equalized Classes

approach of clustering-based under-sampling significantly outperforms SMOTE oversampling.

The implementation of the cross-validation for the oversampling method should be done correctly to avoid overoptimism.

6.5 Handle Imbalanced Classes after Patient Filtering by Diagnoses Code

First of all, we test K-means clustering under-sampling with the approach of mortality prediction of patients with similar disease code classification. Thus, we handle the imbalanced classes after filtering the patient by diagnosis codes. After filtering the patient with ICD (390-459) codes, we apply K-means clustering with the top-N nearest neighbors to the centroids where k=10. The GBDT parameters are set to the parameters that optimize AUC. The accuracy metrics are significantly improved in comparison to the approach of filtering the patient by ICD codes without clustering under-sampling (see the comparison in Figure 6.21). Without under-sampling the AUC equals to 0.894 +/− 0.014, while with under-sampling AUC is 0.963 +/− 0.011. Regardless the fake accuracy of imbalanced data, all the accuracy metrics are also improved: accuracy = 89.54%, AUC = 0.963 +/− 0.011, precision= 89.07%, recall= 90.23%, and f_measure = 89.61%.

Next, we test the affect of data sampling method SMOTE on GBDT with dataset of specific ICD patients group. ICD (390-459) with SMOTE equal classes produces AUC of 0.883 +/− 0.014, accuracy: 91.00%, precision 45.56%,

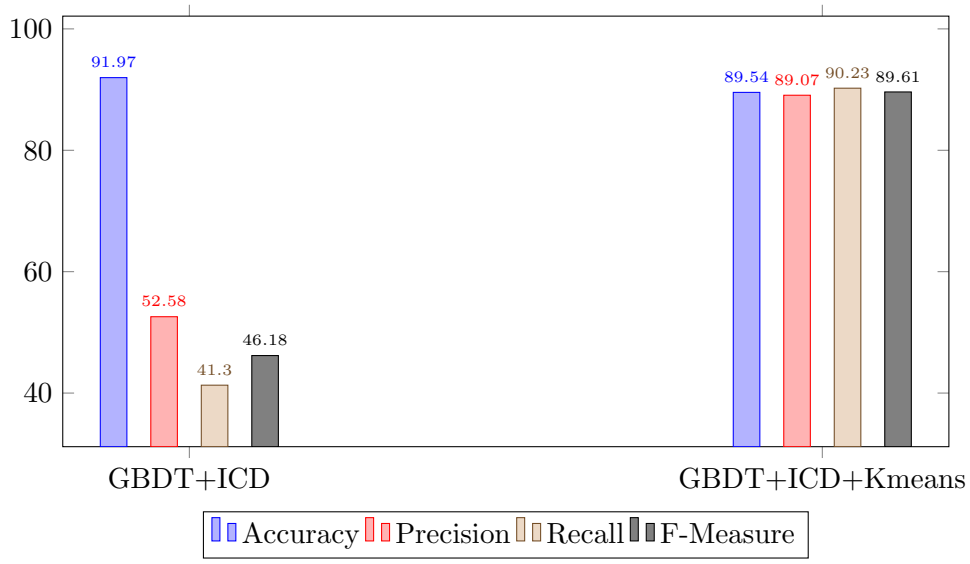


Figure 6.21: Compare GBDT and Filtering Patients by Specific ICD Group of (390-459) with and without K-means Clustering Under-sampling

recall 38.33%, and f_measure 41.56%. We also optimize SMOTE parameters with different up sampling size, number of neighbors and nominal change rate. The optimal parameter values produce better performance than equalize classes – which are upsample size of 500, nominal change rate of 0.2 and number of neighbors 5 produces AUC of 0.894 +/- 0.011, accuracy of 91.29%, precision of 48.04%, recall: 49.67%, and f_measure of 48.60%. In comparison to GBDT with specific ICD patients group (390-459) without SMOTE (see Figure 5.9) AUC of 0.894 +/− 0.014, using SMOTE improves the Recall by 8.37 and the F-Measure by 2.42. However, the precision metric does not improve. Result is in Figure 6.22. As a result the under-sampling by k-means clustering provide better performance with specific ICD codes than SMOTE oversampling.

6.6 Summary

Imbalanced class distribution is a major problem that affects the performance of predicting the risk of death. To handle this problem, we test different data sampling methods: random under-sampling, K-means clustering-based under-sampling, and SMOTE over-sampling. The best method was K-means clustering-based under-sampling.

A significant drawback of under-sampling is that it might remove some useful information. Random under-sampling (that is, choosing training instances from the majority class at random) is one approach for under-sampling the majority class that has this problem. Therefore, the under-sampling has to be done

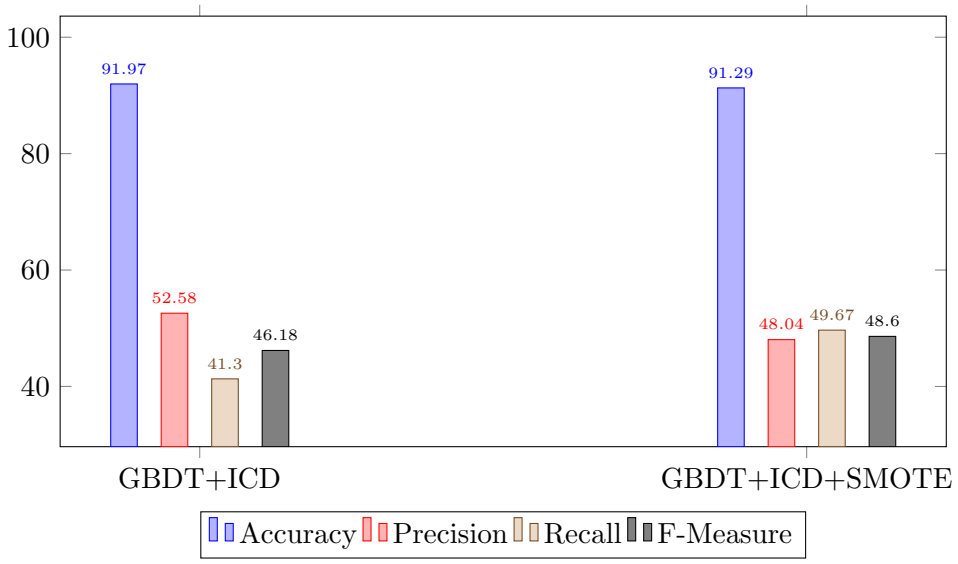


Figure 6.22: Compare GBDT and Filtering Patients by Specific ICD Group of (390-459) with and without SMOTE Over-sampling

carefully. In our approach (i.e., k-means under-sampling), we analyze the deployment of clustering algorithms prior to under-sampling; this approach avoids the deletion of important samples that occurs with random under-sampling. Moreover, we test different approaches to select the majority class representatives from the clusters. The best approach for predictive accuracy is selecting the TopN centroids' nearest neighbors when k equals the optimal value we found for our dataset. It significantly optimizes our model GBDT predictive performance from AUC=0.865 to AUC=0.956.

7

Verification

In this Chapter, we will verify the performance and the generalization of the approach of this thesis. First, we will compare the performance we achieve in mortality risk prediction against the commonly used severity scores on our used dataset. Then, we will use another big ICU dataset for testing our selected ML model for predicting the risk of death. Furthermore, we will test our implemented data under-sampling method to overcome the imbalanced classes and to improve the prediction accuracy on this dataset.

Contents

7.1 Our Approach Vs. Severity Scores

The severity of illness scores are models for mortality risk prediction such as Acute Physiology, Age, Chronic Health Evaluation (APACHE)III [41], Simplified Acute Physiology Score (SAPS) [45], (SAPS)II [47], and Sequential Organ Failure Assessment (SOFA) score [84]. Those sets of severity scores are calculated on the MIMIC-III data for mortality prediction [38]. Table 7.1 gives a comparison between a set of the severity of illness scores and the model of this thesis.

	AUC
SOFA	0.739
SAPS	0.758
SAPS II	0.809
APACHE III	0.784
This thesis model	0.956

Table 7.1: Comparison of this Thesis's Trained Predictive Model with the Severity of Illness Scores on MIMIC-III Dataset.

SOFA had AUC of 0.739, SAPS had AUC of 0.758, SAPSII had AUC of 0.809 and APACHEIII had AUC of 0.784. The approach of this thesis for mortality prediction outperforms the severity of illness scores. We had a high AUC measure of 0.956. Even without handling the imbalanced class distribution our basis ML model still gives a higher AUC (0.865) than the severity scores' AUC.

7.2 Test Our Approach with Another Dataset

We want to verify the performance of our selected ML model and our implemented data sampling method to overcome the imbalanced class problem. To this end, we use a dataset from GOSSIS (Global Open Source Severity of Illness Score) of MIT (Massachusetts Institute of Technology).

GOSSIS provides an extensive database of critical care data from many different intensive care units (ICUs) from Argentina, Australia, New Zealand, Sri Lanka, Brazil, and more than 200 hospitals in the United States. The dataset we have from GOSSIS is used in the challenge of mortality prediction in Women in Data Science (WiDS) 2020 conference. It has 91,713 patient records data from the first 24 hours stay in ICU. It has 186 variables containing identifiers, demographics, vitals, and lab data. In specific, it has APACHE severity score variables.

For using the dataset to test our approach, we had to clean and re-process it. First of all, we remove the identifier variables: encounter_id, hospital_id, icu_id, and patient_id. We remove the features that have a higher rate of missing

values than 50% and the highly correlated features with more than 0.9. The correlation is measured by Pearson correlation coefficient. The final dataset has 108 variables.

In the following subsections, we will test this dataset with the selected ML model and the data sampling method.

7.2.1 Test The ML models

We compare the performance of the seven models that were compared in Chapter 4. We compare the performance of the Logistic Regression (LR), Decision Tree (DT), K-nearest neighbor (KNN), Naïve Bayes (NB), Support Vector Machine (SVM), and Random Forests (RF) against GBDT. However, SVM is excluded because it can not handle missing values. For fair comparison purpose, the models' hyperparameters are not optimized; the default values were used. The result is to find if GBDT gives the highest accuracy for mortality prediction on this dataset, as we had with our dataset extracted from MIMIC-III. The models' performance in predicting the risk of death is compared by 10-fold cross-validation and represented in Table 7.2.

Models	Accuracy	Precision	Recall	F-measure	AUC
LR	92.37%	64.15%	27.25%	37.52%	0.875
DT	91.95%	76.92%	9.85%	17.38%	0.777
GBDT	91.73%	52.29%	49.48%	50.80%	0.884
KNN	87.31%	11.81%	7.23%	8.63%	0.571
NB	83.89%	30.16%	65.79%	41.35%	0.850
RF	92.37%	85.66%	13.95%	23.98%	0.866

Table 7.2: Compare the Performance of Different ML Models

From Table 7.2, we find that we have the same problem of accuracy paradox we had with our dataset extracted from MIMIC-III. The dataset has imbalanced class distribution; out of 91,713 patients there are 83,798 survived patients, and only 7915 suffered patients. Furthermore, we see that out of the box without hyperparameters optimization, GBDT gives the best performance in specific, regarding the main criteria metric, which is the AUC.

7.2.2 Test The Data Under-sampling Method

As can be seen in the previous Section, the dataset is imbalanced, which causes low prediction metrics. Thus, we can handle this problem and improve the prediction accuracy by the proposed K-means under-sampling clustering method of this thesis. We use the best-performing approach we found in Section 6.3.2 to select the majority class representatives (i.e., the TopN centroids' nearest neighbors).

K-means cannot handle missing values; we had to replace the missing values by average values. We didn't run any search for the best number of clusters k, since the default value already shows good performance. We use the default k, which is 5. Then, we select the TopN nearest neighbors to this k. TopN, in this case, is the size of the minority class/k. We combined the resulting data of the majority class with the data of the minority class. This dataset is used to test the GBDT with 10-fold cross-validation and it improves the AUC by 11.88% (see Figure 7.1). Moreover, we compare the time cost to run GBDT on the complete dataset with clustering-based under-sampling and without (see Table 7.3).

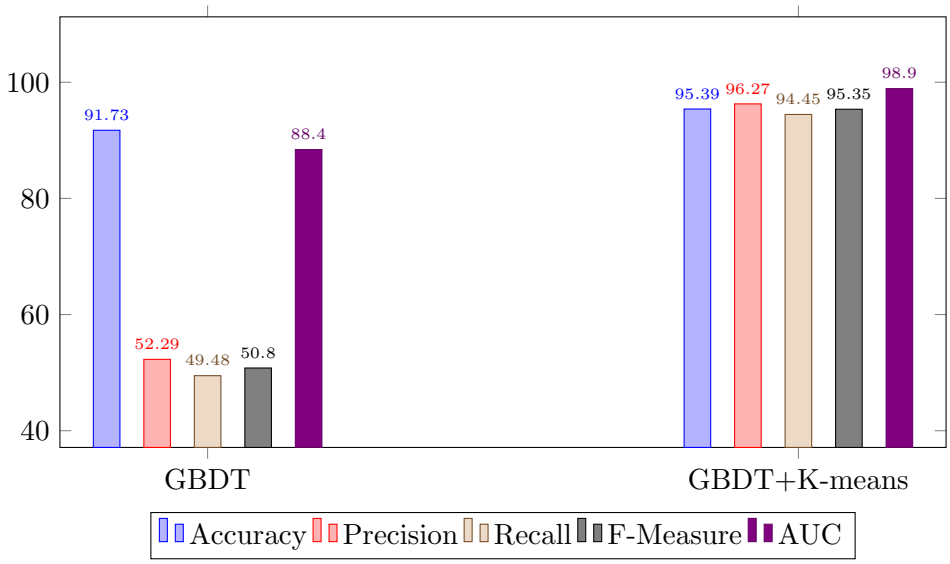


Figure 7.1: GBDT with and without K-means Clustering Under-sampling

Approach	Time Cost
GBDT without under-sampling	28:11:00
GBDT with clustering under-sampling	00:03:23

Table 7.3: Compare Time Cost of GBDT with and without Clustering Under-sampling

7.2.3 Results Summary

We verify this thesis approach for using GBDT model to predict mortality risk and k-means clustering under-sampling method to handle imbalanced data on another larger ICU dataset. Our approach shows its high capability. We did not do models' hyperparameters optimization to test their usability and perfor-

mance without fitting them to that specific dataset. Even without parameters optimization, the performance of the GBDT model and the K-means undersampling method was outstanding.

8

Conclusion

This final chapter summarizes the findings and contributions of this thesis. It discusses this thesis approach in the context of related work to show its strengths and contributions to this research field. It summarizes this thesis work and findings to answer its research questions.

Contents

8.1 Discussion

In Table 8.1, we compare the work of this thesis to the related work discussed in Chapter 2 (those implementing mortality prediction by ML models using similar datasets to ours). We compare the used ML models, the selected feature size, and the time window from which the features are selected. Furthermore, we compare the work on performance improvement factors mentioned in Section 3.5. For instance, the feature selection methods that are used to find the optimal feature set after the initial feature set extraction, handling the imbalanced data problem, and the best performance achieved by using similar datasets to ours (based on MIMIC-III). In comparison to the related work, this thesis has the following advantages:

- Researchers have been using a different set of ML models for predicting the risk of mortality for ICU patients. They either use one model or compare different models. We compare the performance of seven ML models. In particular, we find the outstanding performance of the ensemble model, the GBDT, which is not commonly used by them. In addition, we discuss in detail tuning the hyperparameters of the selected models which has not been done by other authors (see Chapter 4). Moreover, we optimize the hyperparameters in two cases to optimize accuracy and AUC metrics. This shows that optimizing the AUC gives a clear guide for optimal parameter value selection.

- The predictor variables and extracted feature sizes differ between the works, and they might get larger than 100. The works that use a small variables size do not specify the final size of the extracted features [15, 28]. The time window is either a small one, 24 hours, or a larger one 48 hours. In this thesis, we use less than 100 features and a small time window of 24 hours. Moreover, the extracted sequence of time-series features is from each 6-hour period, while some of them use a smaller period (e.g., a value from every 3 hours). Furthermore, we mention the information about the extracted variables and features (see Section 3.3.1).

- After the initial feature set extraction, we implement and compare different feature selection methods (see Section 5.3). Furthermore, we test two cases for optimization metrics (i.e., optimizing accuracy and optimizing AUC) and provide the details of the methods. In some of the related work, there was no use of feature selection methods to find the optimal feature set. In other work, only one method was used for feature selection.

- One of the significant contributions of this thesis is handling the imbalanced class distributions of the MIMIC dataset. Even though the effect of imbalanced data on the ML model's performance is recognized, some of the researchers of the related work ignore it. Moreover, the works that

handle it use the low performance random under-sampling and do not try other methods. In this thesis, a dedicated chapter of handling imbalanced data is provided (see Chapter 6), where different data sampling methods are compared. We implement an efficient clustering-based under-sampling method to create a training dataset which is used to train and optimize the prediction performance of the ML model. Furthermore, we find that this clustering under-sampling method significantly outperformed the random under-sampling method.

- Our best achieved AUC notably outperforms the AUC achieved in related work. The best obtained AUC of the prediction accuracy ranges from 0.7 to 0.9. The two best-performing AUCs from the related work [68, 38] have almost twice as many features as we use. The third best AUC of 0.870 [28] has a more extended time window (i.e., 48 hours) than us. Therefore, we can say that our approach contributes to this research field of mortality prediction on the MIMIC dataset by providing an optimal setting for achieving a high prediction accuracy. Moreover, our approach shows its capability on another ICU dataset (see Chapter 7).

- We mentioned in Section 3.4.2 that the related work on mortality prediction usually tries to maximize the AUC in evaluating their models. However, they didn't mention which metric they use for the other evaluation purposes. In this thesis, we evaluate the accuracy, recall, precision, and f-measure beside the AUC in all the steps for model deployment and performance optimization. We compare tuning the models' parameters by optimizing accuracy and AUC. Furthermore, the feature selection methods are tested for optimizing both accuracy and AUC. For evaluating the models, we measure all of the five mentioned accuracy metrics. In general, we find with imbalanced data accuracy is absolutely not the right metric to use in evaluation. Following the AUC measure provides guidance in all the steps for model deployment. After all, we had a high AUC measure of 0.956. Even without handling the imbalanced class distribution, our basis ML model still give a high AUC 0.865.

- The MIMIC data providers reviewed and tried to reproduce the work that has been done on mortality prediction on the MIMIC data (Johnson *et al.* [39]). They find that the reproduction of the studies is challenging and cannot be guaranteed. They state the demand for providing the details of the used methods for building the dataset (e.g., data pre-processing, variable selection)—moreover, the need to provide open code for public benchmark purposes. Thus, in this thesis, we consider this reproducibility problem; hence we provide the required methods' and models' details and the open code. Therefore, this thesis is an available public benchmark.

Study	Machine Learning Models	Feature size and Time Window	Feature Selection Method	Handle Imbalanced Data	Best Performance
This Thesis	LR, DT, KNN, NB, GBDT, SVM, and RF	74 features from 24 hours	Filter approach (weight by Chi-Squared), wrapper approaches (Forward Selection and Backward Elimination), and Embedded method by GBDT	Clustering-based under-sampling and compared with other methods	AUC= 0.956
Lee *et al.* [50]	LR and DT	76 features from 24 hours	None	cross-validation incorporated stratified sampling	AUC= 0.830
Morid *et al.* [59]	kNN	868 features from 48 hours	wrapper approach (Gradient Descent)	None	F-measure= 0.66
Luo *et al.* [55]	LR	54 variables and 100 features, from 24 hours	non-negative matrix factorization	None	AUC= 0.848
Ghassemi *et al.* [23]	LR and SVM	313,461 notes, time prior ICU discharge	topic modeling for dimensionality reduction	randomly sub-sampling the negative class	AUC= 0.812
Davoodi and Moradi [15]	Deep Rule-Based Fuzzy Classifier	29 variables the feature size not specified, from 48 hours	significance of an attribute towards the clustering process [2]	random under-sampling	AUC= 0.739

Purushotham *et al.* [68]	Super Learner models and Deep Learning models	136 features from 24 hours	None	None	AUC= 0.941
Harutyunyan *et al.* [28]	LR and LSTM-based models	17 variables the feature size not specified, from 48 hours	None	None	AUC= 0.870
Johnson and Mark [38]	LR, LASSO, L2, and GBDT	148 features from 24 hours	None	None	AUC= 0.927

Table 8.1: Comparison of this Thesis Approach to the Related Work on Mortality Prediction

8.2 Summary

The research questions of this thesis were specified in Chapter 1, which are focused on three main points accuracy of prediction, the effect of the dataset on prediction, and performance optimization. We summarize the work of this thesis that has been done to answer these questions.

- **Research Question Regarding Accuracy of Prediction:** the main factors that affect the accuracy are introduced in Section 1.1, which are the ML model and its parameters, curse of dimensionality, and class imbalance problem. These factors are explored and discussed in the following chapters Chapter 4, Chapter 5, and Chapter 6. One major question in this point is the effect of the ML model on the accuracy and which model gives the highest performance for mortality risk prediction and it is answered by Chapter 4. It provides the way for implementing the main building block of the system, which is the ML models. We tested different ML models to find the optimal one; the models' performance was different with respect to their capability to deal with the imbalanced dataset. The models' prediction performance is severely affected due to the imbalanced class distributions. The imbalanced data has a major effect on the model accuracy since it causes the model to be biased about predicting the majority class, which is not our target. We find the selection of the ML model is an important decision to take in mortality prediction because of the imbalanced class problem. Furthermore, the model's parameters selection and

the accuracy metric to optimize during the selection are absolute factors on the model's prediction accuracy. Out of the seven models, we find that GBDT is the optimal one. In Chapter 5, we discuss and provide practical test of the effect of the predictor variables and the feature selection on the accuracy. In Chapter 6, we discuss the major effect of the imbalanced data on prediction accuracy.

- **Research Question Regarding Effect of Dataset on Prediction:** it is about the effect of a dataset on the accuracy of the model prediction. In section 5.2, we test the effect of using the normalized and un-normalized data on the prediction accuracy. We find that the accuracy metrics are improved with the normalized data. In addition, we find in Chapter 4 the significant effect of the dataset problem of the imbalanced class distribution on the model accuracy. Imbalanced data not only affects the model's performance but also affects the selection of the model parameters and the selection of the predictor features. For this reason, we had to consider different accuracy metrics during the selections; and re-implement them on the balanced data. Moreover, for performance optimization we consider handling imbalanced problem (see the following point "Performance Optimization").

  One question asked in this point "Does focusing on a specific disease (i.e., filtering patients by disease code) improve the accuracy of the risk prediction? Or is a heterogeneous dataset (i.e., patient with different disease codes) enough?" is answered by Section 5.4. We find that the diagnoses code ICD is the second high importance feature in making a decision of risk of death in GBDT (see Table 5.2). We hypothesise that predicting mortality of patients with similar ICD codes will help to increase predictive performance. Thus, we filter the patients by the diagnoses code. Then implement the ML model for predicting the risk of mortality on the patients with similar diagnoses code. Even though the resulted sampled datasets with similar ICD codes have a different imbalanced ratio, we find the prediction performance improved. For instance, the complete dataset without filtering the ICD codes has an imbalance ration IR=8.9, while the dataset of patients with the specific ICD codes (390-459) has IR=10.96. Implementing the GBDT on the dataset of patients with this specific ICD codes gives AUC of 0.894, which improves the AUC of implementing the GBDT on the dataset with unfiltered ICD codes (AUC = 0.865) even with the higher imbalanced ratio.

- **Research Question Regarding Performance Optimization:** We search for the different performance optimizations that can be done to improve the prediction accuracy of the ML model. We work for performance optimization by tuning the accuracy factors we defined. The performance optimization techniques we follow are introduced by Section 3.5:

tuning the model parameters, selecting a subset of features, and further pre-processing the data.

For tuning the model parameters, in Chapter 4, we defined the critical parameters for each ML model. Then we use a grid search to find the optimal values for those parameters. We find that when we were searching for the optimal value by optimizing the accuracy metric, the accuracy metrics were not improved better than the resulted accuracy metrics from using the default values of the parameters. Thus, we grid search for the optimal value while optimizing the AUC. It helps us to find the optimal values to improve performance. As a result, tuning the parameters by optimizing the AUC provides better performance than by optimizing the accuracy.

For selecting a subset of features, in Chapter 5 in Section 5.3, we implement different feature selection methods to find the subset of the features that optimizes the prediction performance. The filter, wrapper, and embedded methods were implemented. The filter method was not as good as the wrapper methods. Our optimal ML model, the GBDT, has higher performance on the features selected by backward elimination in comparison to the forward selection (of the wrapper approach). Furthermore, the time needed to implement forward selection with GBDT is longer than to perform backward elimination. However, forward selection uses much fewer features than Backward elimination. Filter and wrapper feature selection methods were not able to achieve high prediction performance because of the class imbalance problem. However, the performance result of using the selected features from the embedded method of GBDT is highly competitive to the result of the complete feature set. This result also answers our question regarding the best combination of the ML model and the feature selection method considering the run time and the accuracy.

For further pre-processing the data, we consider the major problem of our dataset, which is the class imbalance. Therefore, in Chapter 6, we test different data sampling methods to overcome this problem; over-sampling and under-sampling methods. Over-sampling adds more similar instances to the minority class. In this sense, the chance of overfitting the model to the minority class is increasing. Under-sampling balances the dataset by reducing the size of the majority class. Moreover, it keeps the instances from the minority class intact since it doesn't change them (i.e., our important target class). We find that the works that have been done on mortality prediction either ignore the imbalanced problem or apply random under-sampling, which has a major weakness by random elimination of the majority class instances, and hence losing useful information in the majority class. Thus, in this thesis, to avoid losing information on the majority class and retain as many useful and informative samples as pos-

sible, we apply clustering-based under-sampling. The ML model trained on the balanced data resulted from using clustering-based under-sampling. The results prove that when the imbalanced class distributions problem is treated by the proposed clustering-based under-sampling, models' performance to predict risk mortality significantly improved.

We asked if ML models will improve the prediction accuracy beyond the standard severity scores for patient's risk prediction. An evaluation of another dataset and verification of the approach of this thesis is conducted in Chapter 7. In section 7.1, we compare our implemented ML model, the GBDT against the calculated severity scores on our used dataset. We find that our ML model significantly outperforms the severity of illness scores in predicting the risk of mortality. Moreover, it performs well on the other dataset.

In the end, this thesis aims to provide an overview and implementation of the predictive model for intelligent medical decision support for predicting the risk of death. The main goal to achieve was to provide an optimal setting for accurately predicting the risk of death. It provides the required steps for data pre-processing, feature selection, and handling the imbalanced class problem for implementation and performance optimization. It proposes an approach that combined the ensemble ML model with clustering-based under-sampling for significantly higher predictive accuracy in predicting the risk of mortality. Our used dataset is highly imbalanced, with a very low balance ratio of 0.11. The implemented ML the GBDT has an outstanding performance even before handling the imbalance problem (AUC = 0.865). After handling imbalanced data with the proposed cluster-based under-sampling GBDT has great performance (AUC=0.956).

We provide a detailed comparison to the related work on mortality prediction; this thesis contributes by providing many detailed works on different methods, and our approach performance outperformed them. In comparison to the commonly used severity scores, our approach highly exceeded their performance. Furthermore, applying our approach to another dataset shows its capability and high performance. Therefore, the optimal setting for predicting mortality risk to support medical decisions is achieved.

In addition, an existing problem in this research field is the demand of the detailed methods and open code for public benchmarking [39]. In this thesis, we do our best to present all the needed details for data pre-processing, feature selection, and model implementation and its parameter selection for reproducibility purposes. Moreover, our code is open for public benchmarking.

8.3 Future Work

After accomplishing this thesis, one major step we look forward to is implementing its approach for predicting the risk of mortality in a real medical clinic. There are critical requirements that have to be considered.

For medical staff to trust our prediction, the model has to be understandable. This means it should be easy to understand the reasons behind a decision of risk. Our model, the GBDT, fulfills this requirement. Unlike the black box models as the neural network or deep learning models, GBDT is an interpretable model. GBDT provides ranked features with importance to predict the risk of mortality (for instance, see Table 5.2). Moreover, the model to make a decision it builds a decision tree with branches which medical staff can also follow to understand the reason behind a decision.

Another requirement is usability. We consider using a framework that doesn't require advanced programming skills which simplified our mortality risk prediction approach. RapidMiner has an easy to understand interface. Furthermore, a review of RapidMiner by a healthcare data analytics director provided in Gartner peer insights approves its simplicity (see Section 3.2.3). Moreover, RapidMiner provides medical support (see [56]). Most importantly, RapidMiner offers database connection, which is one option to usable clinical implementation of our approach by RapidMiner. Thus, a clinical database can be connected to it, and then our approach can be easily implemented (i.e., directly run the code we provide).

Moreover, the short run time is an important requirement. We implement in-hospital mortality risk prediction after the first 24 hours of ICU stays. Thus, when all the medical measurements (i.e., features) of the 24 hours ICU stay are collected the prediction should be ready. Our model GBDT takes for the complete dataset with 10-fold cross-validation around 5 minutes. Moreover, the clustering-based under-sampling took for clustering the complete dataset only 2 seconds.

However, other requirements should be considered in a real application for that critical medical field. For instance, a crucial requirement is patient data privacy and security. This requirement needs to be addressed and studied in detail. Federated learning is an option to deal with this privacy issue. By federated learning, a ML model with a central server can be applied to decentralized data scattered among isolated medical institutions, hospitals, or devices while keeping all the sensitive data where they belong and considering the privacy concerns (for more details see [91]). Xu *et al.* [91] summarizes some application of healthcare tasks in the federated learning setting, such as predicting mortality and ICU stay time by Huang *et al.* [34].

One major weakness of the commonly used severity of illness scores to predict the risk of mortality is the generic mortality prediction. They use a set of medical measurements (i.e., features) to predict the mortality risk of all the

patients regardless of the diseases they have. The risk of mortality prediction after filtering patients by specific ICD codes is presented in Section 5.4. We find that disease-specific mortality prediction increases the prediction accuracy. The patients that have a specific classification in ICD codes will have a similar disease with similar symptoms, which helps for predicting the risk. Furthermore, to have a disease dependent prediction of mortality, we execute a feature selection process on a group of patients with a specific disease to find the optimal features (see Section 5.4.3). We find that using the selected features for predicting mortality on those patients with the particular ICD codes increase the prediction accuracy. Thus, further research on that field would be beneficial. Moreover, a practical application of automatic patient filtering by ICD codes and automatic use of the optimal features is absolutely a great advancement. Then, automatically only the features that are found to provide an optimal risk prediction of that specific ICD codes will be used for predicting mortality risk for that patient.

Bibliography

[1] Samir E Abdelrahman and Bruce E Bray. Frequency tree clustering for icu mortality analytics using graph databases. In *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*, pages 813–817. IEEE, 2016.

[2] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[3] Adnan Amin, Sajid Anwar, Awais Adnan, Muhammad Nawaz, Newton Howard, Junaid Qadir, Ahmad Hawalah, and Amir Hussain. Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. *IEEE Access*, 4:7940–7957, 2016.

[4] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia,USA, 2007. Society for Industrial and Applied Mathematics.

[5] Mohammadhossein Barkhordari and Mahdi Niamanesh. ScaDiPaSi: an effective scalable and distributable MapReduce-based method to find patient similarity on huge healthcare networks. *Big Data Research*, 2(1):19–27, 2015.

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

[7] Rok Blagus and Lara Lusa. Joint use of over-and under-sampling techniques and cross-validation for the development and assessment of prediction models. *BMC bioinformatics*, 16(1):363, 2015.

[8] Henrik Brink, Joseph W Richards, Mark Fetherolf, and Beau Cronin. *Real-world machine learning*. Manning Shelter Island, NY, 2017.

[9] Lawrence WC Chan, Ying Liu, Tao Chan, Helen KW Law, SC Cesar Wong, Andy PH Yeung, KF Lo, SW Yeung, KY Kwok, William YL Chan, et al. Pubmed-supported clinical term weighting approach for improving inter-patient similarity measure in diagnosis prediction. *BMC medical informatics and decision making*, 15(1):43, 2015.

[10] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.

[11] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[12] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Icml*, volume 1, pages 74–81, 2001.

[13] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, PAMI-1(2):224–227, 1979.

[14] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[15] Raheleh Davoodi and Mohammad Hassan Moradi. Mortality prediction in intensive care units (icus) using a deep rule-based fuzzy classifier. *Journal of biomedical informatics*, 79:48–59, 2018.

[16] Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2016.

[17] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38, 2004.

[18] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[19] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[20] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions*

on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4):463–484, 2012.

[21] Nicolas Garcelon, Antoine Neuraz, Vincent Benoit, Rémi Salomon, Sven Kracker, Felipe Suarez, Nadia Bahi-Buisson, Smail Hadj-Rabia, Alain Fischer, Arnold Munnich, et al. Finding patients using similarity measures in a rare diseases-oriented clinical data warehouse: Dr. warehouse and the needle in the needle stack. *Journal of biomedical informatics*, 73:51–61, 2017.

[22] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[23] Marzyeh Ghassemi, Marco AF Pimentel, Tristan Naumann, Thomas Brennan, David A Clifton, Peter Szolovits, and Mengling Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 446–453, 2015.

[24] Assaf Gottlieb, Gideon Y Stein, Eytan Ruppin, Russ B Altman, and Roded Sharan. A method for inferring medical diagnoses from patient similarities. *BMC medicine*, 11(1):194, 2013.

[25] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[26] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[27] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[28] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):96, 2019.

[29] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, 21(9):1263–1284, 2008.

[30] T Ryan Hoens and Nitesh V Chawla. Imbalanced datasets: from sampling to classifiers. In Haibo He and Yunqian Ma, editors, *Imbalanced learning: foundations, algorithms, and applications*, chapter 3, pages 43–59. John Wiley & Sons, 2013.

[31] Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2016.

[32] Mark Hoogendoorn, Ali el Hassouni, Kwongyen Mok, Marzyeh Ghassemi, and Peter Szolovits. Prediction using patient comparison vs. modeling: A case study for mortality prediction. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 2464–2467. IEEE, 2016.

[33] Li-Yu Hu, Min-Wei Huang, Shih-Wen Ke, and Chih-Fong Tsai. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1):1304, 2016.

[34] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics*, 99:103–291, 2019.

[35] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[36] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.

[37] Alistair EW Johnson, Mohammad M Ghassemi, Shamim Nemati, Katherine E Niehaus, David A Clifton, and Gari D Clifford. Machine learning and decision support in critical care. *Proceedings of the IEEE*, 104(2):444–466, 2016.

[38] Alistair EW Johnson and Roger G Mark. Real-time mortality prediction in the intensive care unit. In *AMIA Annual Symposium Proceedings*, volume 2017, pages 994–1003, 2017.

[39] Alistair EW Johnson, Tom J Pollard, and Roger G Mark. Reproducibility in critical care: a mortality prediction case study. In *Machine Learning for Healthcare Conference*, pages 361–376, 2017.

[40] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3, 2016.

[41] William A Knaus, Douglas P Wagner, Elizabeth A Draper, Jack E Zimmerman, Marilyn Bergner, Paulo G Bastos, Carl A Sirio, Donald J Murphy, Ted Lotring, Anne Damiano, et al. The apache iii prognostic system: risk prediction of hospital mortality for critically iii hospitalized adults. *Chest*, 100(6):1619–1636, 1991.

[42] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[43] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[44] N Santhosh Kumar, K Nageswara Rao, A Govardhan, K Sudheer Reddy, and Ali Mirza Mahmood. Undersampled k-means approach for handling imbalanced distributed data. *Progress in Artificial Intelligence*, 3(1):29–38, 2014.

[45] JR Gall Le, P Loirat, F Nicolas, C Granthil, F Wattel, R Thomas, P Glaser, P Mercier, J Latournerie, P Candau, et al. Use of a severity index in 8 multidisciplinary resuscitation centers. *Presse medicale (Paris, France: 1983)*, 12(28):1757–1761, 1983.

[46] Tuong Le, Hoang Le Son, Minh Thanh Vo, Mi Young Lee, Sung Wook Baik, et al. A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset. *Symmetry*, 10(7):250, 2018.

[47] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. *Jama*, 270(24):2957–2963, 1993.

[48] Joon Lee. Patient-specific predictive modeling using random forests: an observational study for the critically ill. *JMIR medical informatics*, 5(1):e3, 2017.

[49] Joon Lee, Joel A Dubin, and David M Maslove. Mortality prediction in the ICU. In *Secondary Analysis of Electronic Health Records*, pages 315–324. Springer, 2016.

[50] Joon Lee, David M Maslove, and Joel A Dubin. Personalized mortality prediction driven by electronic medical data and a patient similarity metric. *PloS one*, 10(5):e0127428, 2015.

[51] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[52] Li Li, Wei-Yi Cheng, Benjamin S Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P Bottinger, and Joel T Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science translational medicine*, 7(311):311ra174–311ra174, 2015.

[53] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409:17–26, 2017.

[54] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013.

[55] Yuan Luo, Yu Xin, Rohit Joshi, Leo Celi, and Peter Szolovits. Predicting icu mortality risk by grouping temporal trends from a multivariate panel of physiologic measurements. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 42–50, 2016.

[56] Ingo Mierswa and Ralf Klinkenberg. Rapidminer. URL: `https://rapidminer.com/industry/healthcare/` . Accessed on: June.04,2020.

[57] Ingo Mierswa and Ralf Klinkenberg. Rapidminer studio. Version 9.2. URL: `https://rapidminer.com` . Accessed on: Jan.15,2019.

[58] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, 2006.

[59] Mohammad Amin Morid, Olivia R Liu Sheng, and Samir Abdelrahman. PPMF: A patient-based predictive modeling framework for early ICU mortality prediction. *arXiv preprint arXiv:1704.07499*, 2017.

[60] Andreas C Müller, Sarah Guido, et al. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[61] Nir Ofek, Lior Rokach, Roni Stern, and Asaf Shabtai. Fast-cbus: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing*, 243:88–102, 2017.

[62] Carlos Ordonez. Data set preprocessing and transformation in a database system. *Intelligent Data Analysis*, 15(4):613–631, 2011.

[63] Carlos Ordonez. Can we analyze big data inside a DBMS? In *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, pages 85–92. ACM, 2013.

[64] Maryam Panahiazar, Vahid Taslimitehrani, Naveen L Pereira, and Jyotishman Pathak. Using EHRs for heart failure therapy recommendation using multidimensional patient similarity analytics. *Studies in health technology and informatics*, 210:369–373, 2015.

[65] Rattanawadee Panthong and Anongnart Srivihok. Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm. *Procedia Computer Science*, 72:162–169, 2015.

[66] Yoon-Joo Park, Byung-Chun Kim, and Se-Hak Chun. New knowledge extraction technique using probability for case-based reasoning: application to medical diagnosis. *Expert Systems*, 23(1):2–20, 2006.

[67] Joshua P Parreco, Antonio E Hidalgo, Alejandro D Badilla, Omar Ilyas, and Rishi Rattan. Predicting central line-associated bloodstream infections and mortality using supervised machine learning. *Journal of critical care*, 45:156–162, 2018.

[68] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134, 2018.

[69] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[70] Lior Rokach and Oded Maimon. Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer, 2005.

[71] Miriam Seoane Santos, Jastin Pompeu Soares, Pedro Henrigues Abreu, Helder Araujo, and Joao Santos. Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]. *IEEE Computational Intelligence Magazine*, 13(4):59–76, 2018.

[72] Nicole Sarna, Araek Tashkandi, and Lena Wiese. Patient similarity analysis for personalized health prediction models (abstract). In *European Conference on Data Analysis (ECDA)*, 2018.

[73] Anis Sharafoddini, Joel A Dubin, and Joon Lee. Patient similarity in prediction models based on health data: a scoping review. *JMIR medical informatics*, 5(1):e7, 2017.

[74] Jia Song, Xianglin Huang, Sijun Qin, and Qing Song. A bi-directional sampling based on k-means method for imbalance text classification. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–5. IEEE, 2016.

[75] KT Sridhar. Modern column stores for big data processing. In *International Conference on Big Data Analytics*, pages 113–125. Springer, 2017.

[76] Jimeng Sun, Daby Sow, Jianying Hu, and Shahram Ebadollahi. A system for mining temporal physiological data streams for advanced prognostic decision support. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1061–1066. IEEE, 2010.

[77] Araek Tashkandi, Ingmar Wiese, and Lena Wiese. Efficient in-database patient similarity analysis for personalized medical decision support systems. *Big data research*, 13:52–64, 2018.

[78] Araek Tashkandi and Lena Wiese. Leveraging patient similarity analytics in personalized medical decision support system. In *Learning, Knowledge, Data, Analytics (LWDA), 2017 FGDB Database Workshop*, page 125. CEUR-WS, 2017.

[79] Araek Tashkandi and Lena Wiese. A hybrid machine learning approach for improving mortality risk prediction on imbalanced data. In *The 21st International Conference on Information Integration and Web-based Applications and Services (iiWAS)*, pages 83–92. ACM, 2019.

[80] Araek Tashkandi and Lena Wiese. Intelligent medical decision support system for predicting patients at risk in intensive care units. In *PRE-ICIS SIGDSA Symposium, Association for Information Systems Special Interest Group on Decision Support and Analytics (SIGDSA)*. AIS eLibrary, 2019.

[81] Chih-Fong Tsai, Wei-Chao Lin, Ya-Han Hu, and Guan-Ting Yao. Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Information Sciences*, 477:47–54, 2019.

[82] Francisco J Valverde-Albacete and Carmen Peláez-Moreno. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox. *PloS one*, 9(1):e84217, 2014.

[83] Sven Van Poucke, Zhongheng Zhang, Martin Schmitz, Milan Vukicevic, Margot Vander Laenen, Leo Anthony Celi, and Cathy De Deyne. Scalable predictive analysis in critically ill patients using a visual open data analysis platform. *PloS one*, 11(1):e0145791, 2016.

[84] J-L Vincent, Rui Moreno, Jukka Takala, Sheila Willatts, Arnaldo De Mendonça, Hajo Bruining, CK Reinhart, PeterM Suter, and Lambertius G Thijs. The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure, 1996.

[85] Fei Wang, Jianying Hu, and Jimeng Sun. Medical prognosis based on patient similarity and expert feedback. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1799–1802. IEEE, 2012.

[86] Yichuan Wang, LeeAnn Kung, and Terry Anthony Byrd. Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, 126:3–13, 2018.

[87] Brian J Wells, Kevin M Chagin, Amy S Nowacki, and Michael W Kattan. Strategies for handling missing data in electronic health record derived data. *eGEMs*, 1(3):1035, 2013.

[88] Ingmar Wiese, Nicole Sarna, Lena Wiese, Araek Tashkandi, and Ulrich Sax. Concept acquisition and improved in-database similarity analysis for medical data. *Distributed and Parallel Databases*, 37(2):297–321, 2018.

[89] Lena Wiese. *Advanced Data Management for SQL, NoSQL, Cloud and Distributed Databases*. DeGruyter, 2015.

[90] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[91] Jie Xu and Fei Wang. Federated learning for healthcare informatics. *arXiv preprint arXiv:1911.06270*, 2019.

[92] Ta Zhou, Fu-lai Chung, and Shitong Wang. Deep tsk fuzzy classifier with stacked generalization and triplely concise interpretability guarantee for large data. *IEEE Transactions on Fuzzy Systems*, 25(5):1207–1221, 2016.