A Scalable Data Science Platform built on Open Source Technologies with Application of Predictive Analytics on Acute Respiratory Distress Syndrome disease

Vaggelis Chaniotakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisors:
Prof. *Dimitris Plexousakis*
Prof. *Manolis Tsiknakis*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

A Scalable Data Science Platform built on Open Source Technologies with Application of Predictive Analytics on Acute Respiratory Distress Syndrome disease

Thesis submitted by
Vaggelis Chaniotakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Vaggelis Chaniotakis

Committee approvals: DIMITRIOS PLEXOUSAKIS Digitally signed by DIMITRIOS PLEXOUSAKIS Date: 2021.03.05 12:49:01 +02'00'
_____
Dimitris Plexousakis
Professor, Thesis Supervisor, UOC

Manolis Tsiknakis Digitally signed by Manolis Tsiknakis Date: 2021.03.05 13:56:56 +02'00'
_____
Manolis Tsiknakis
Professor, Thesis Advisor, Committee Member, HMU

_____
Kostas Magoutis
Associate Professor, Committee Member, UOC

Departmental approval: _____
Polyvios Pratikakis
Assistant Professor, Director of Graduate Studies, UOC

Heraklion, March 2021

A Scalable Data Science Platform built on Open Source Technologies with Application of Predictive Analytics on Acute Respiratory Distress Syndrome disease

Abstract

The continuous growth of high volumes of biomedical data in healthcare generates significant challenges for their efficient management. This need has made inevitable the adoption of big data infrastructures and relevant techniques from healthcare organizations, in order for them to efficiently explore the wealth of real-world data generated with the objective to improve the quality of healthcare services. In the healthcare industry, various big data sources, that are characterized by heterogeneity, exist. These include hospital information systems (HIS) and medical records of patients (EHRs), results of laboratory procedures and examinations residing in relevant information systems (Laboratory Information Systems - LIS), data from continuous patient monitoring (e.g. in an Intensive Care Unit - ICU) and data from smart devices, such as wearables. Also, very big data sets are generated from genomics-related clinical and research work. Regarding genomics, the rate of growth over the last decade has also been truly astonishing, with the total amount of sequence data produced doubling approximately every seven months [55]. This data requires efficient management and analysis in order to derive meaningful and actionable information.

In developing such solutions, a range of challenges and complications associated with each step of the pipeline for handling such healthcare big data sets need to be addressed. These can only be resolved by using high-quality computing solutions for big data analysis. Especially in the current situation of the COVID-19 pandemic, complications that might occur after the onset of this disease are really important. An important such complication is Acute Respiratory Distress Syndrome (ARDS), which is a serious respiratory condition with high mortality and associated morbidity. A large number of basic and clinical studies have demonstrated that early diagnosis and intervention are key to improving the survival rate of patients with ARDS [56]. Therefore, there is a pressing need for the development and clinical testing of predictive models for ARDS events, which might improve the clinical diagnosis or the management of ARDS.

In the present thesis, we focused on two distinct objectives; namely a) to design a scalable data science platform, built on open source technologies, and b) to exploit the platform and publically available big healthcare datasets to develop machine learning models for predicting acute respiratory distress syndrome (ARDS) events through commonly available parameters, including baseline characteristics and clinical and laboratory parameters.

This thesis is divided into two main parts. The first part presents and analyzes in detail all the procedures, materials, and methods adopted to develop this big data management platform. We report on the complications and difficulties that arise in creating and using such systems with large biomedical datasets, such as the MIMIC-III dataset. The second part of the thesis describes how we exploit this clinical database, to perform an evaluation study of our platform on a real world clinical scenario for ARDS. The objective of the study was to develop and evaluate a novel application of machine learning models for predicting acute respiratory distress syndrome (ARDS. We employ random forests and logistic regression algorithmic models, trained on patient health record data for the early prediction and diagnosis of ARDS. Our approach achieves better results in all metrics that are based on AUC, when compared to relevant published efforts using the MIMIC III dataset to develop predictive models of ARDS. Specifically, both of our algorithmic models outperform in ARDS prediction, with 10-fold cross validated Random Forest being dominant, according to AUC (95.1%), Accuracy (98.0%), Specificity (98.62%) and Sensitivity (96.25%).

Μια Επεκτάσιμη Πλατφόρμα Επιστήμης Δεδομένων, βασισμένη σε Τεχνολογίες Ανοιχτού Κώδικα με Εφαρμογή Προγνωστικής Ανάλυσης για τη νόσο του Συνδρόμου Οξείας Αναπνευστικής Δυσχέρειας

Περίληψη

Η συνεχής ανάπτυξη μεγάλου όγκου βιοϊατρικών δεδομένων στην υγειονομική περίθαλψη δημιουργεί σημαντικές προκλήσεις για την αποτελεσματική τους διαχείριση. Αυτή η ανάγκη έκανε αναπόφευκτη την υιοθέτηση μεγάλων υποδομών δεδομένων και σχετικών τεχνικών από οργανισμούς υγειονομικής περίθαλψης, προκειμένου να εξερευνήσουν αποτελεσματικά τον πλούτο των δεδομένων του πραγματικού κόσμου που δημιουργούνται με στόχο τη βελτίωση της ποιότητας των υπηρεσιών υγείας. Στη βιομηχανία υγειονομικής περίθαλψης, υπάρχουν διάφορες μεγάλες πηγές δεδομένων, που χαρακτηρίζονται από ετερογένεια. Αυτές περιλαμβάνουν νοσοκομειακά συστήματα πληροφοριών (HIS) και ιατρικά αρχεία ασθενών (EHRs), αποτελέσματα εργαστηριακών διαδικασιών και εξετάσεων που βρίσκονται σε σχετικά συστήματα πληροφοριών (Laboratory Information Systems - LIS), δεδομένα από συνεχή παρακολούθηση ασθενών (π.χ. σε μία μονάδα εντατικής θεραπείας - ΜΕΘ) και δεδομένα από έξυπνες συσκευές, όπως φορητά. Επίσης, πολύ μεγάλα σύνολα δεδομένων δημιουργούνται από κλινικές και ερευνητικές εργασίες που σχετίζονται με τη γονιδιωματική. Όσον αφορά τη γονιδιωματική, ο ρυθμός ανάπτυξης κατά την τελευταία δεκαετία ήταν επίσης πραγματικά εκπληκτικός, με τον συνολικό αριθμό δεδομένων αλληλούχισης που παράγονται να διπλασιάζεται περίπου κάθε επτά μήνες [55]. Αυτά τα δεδομένα απαιτούν αποτελεσματική διαχείριση και ανάλυση προκειμένου να εξάγουν ουσιαστικές και εφαρμόσιμες πληροφορίες.

Κατά την ανάπτυξη τέτοιων λύσεων πρέπει να αντιμετωπιστεί μια σειρά από προκλήσεις και επιπλοκές που συνδέονται με κάθε βήμα του σχεδιασμού συστημάτων για την διαχείριση τέτοιων μεγάλων συνόλων δεδομένων υγειονομικής περίθαλψης. Αυτές μπορούν να επιλυθούν μόνο χρησιμοποιώντας υψηλής ποιότητας υπολογιστικές λύσεις για ανάλυση μεγάλων δεδομένων. Ειδικά στην τρέχουσα κατάσταση της πανδημίας COVID-19, οι επιπλοκές που μπορεί να εμφανιστούν μετά την έναρξη αυτής της ασθένειας στη ζωή του ανθρώπου είναι πραγματικά σημαντικές. Μια σημαντική τέτοια επιπλοκή είναι το σύνδρομο οξείας αναπνευστικής δυσχέρειας (ARDS), το οποίο είναι μια σοβαρή αναπνευστική κατάσταση με υψηλή θνησιμότητα και σχετική νοσηρότητα. Ένας μεγάλος αριθμός βασικών και κλινικών μελετών έχουν δείξει ότι η έγκαιρη διάγνωση και παρέμβαση είναι καθοριστικής σημασίας για τη βελτίωση του ποσοστού επιβίωσης των ασθενών με ARDS. Επομένως, υπάρχει επιτακτική ανάγκη για την ανάπτυξη και κλινική δοκιμή προγνωστικών μοντέλων για συμβάντα ARDS, τα οποία θα μπορούσαν να βελτιώσουν την κλινική διάγνωση ή τη διαχείριση του ARDS.

Στην παρούσα διατριβή, εστιάσαμε σε δύο διαφορετικούς στόχους: συγκεκριμένα α) να σχεδιάσουμε μια επεκτάσιμη πλατφόρμα διαχείρισης μεγάλου όγκου δεδομένων, βασισμένοι σε τεχνολογίες ανοιχτού κώδικα, και β) να εκμεταλλευτούμε την πλατφόρμα και δημόσια διαθέσιμα μεγάλα σύνολα κλινικών δεδομένων προκειμένου να αναπτύξουμε μοντέλα μηχανικής μάθησης για την πρόβλεψη συμβάντων οξείας αναπνευστικής δυσχέρειας (ARDS) μέσω κοινώς διαθέσιμων παραμέτρων, συμπεριλαμβανομένων των βασικών χαρακτηριστικών και των κλινικών και εργαστηριακών παραμέτρων.

Η διατριβή χωρίζεται σε δύο κύρια μέρη. Το πρώτο μέρος παρουσιάζει και αναλύει λεπτομερώς όλες τις διαδικασίες, τα υλικά και τις μεθόδους που υιοθετήθηκαν για την ανάπτυξη αυτής της πλατφόρμας διαχείρισης μεγάλων δεδομένων. Εστιάσαμε στις επιπλοκές και τις δυσκολίες που προκύπτουν κατά τη δημιουργία και τη χρήση τέτοιων συστημάτων σε μεγάλα βιοϊατρικά δεδομένα, όπως το σύνολο δεδομένων MIMIC-III. Το δεύτερο μέρος αυτής της διατριβής, περιγράφει τον τρόπο με τον οποίο χειριστήκαμε αυτήν την κλινική βάση δεδομένων, για να πραγματοποιήσουμε μια μελέτη αξιολόγησης της πλατφόρμας μας, σε ένα πραγματικό κλινικό σενάριο για το ARDS. Ο στόχος της μελέτης μας ήταν να αναπτύξουμε και να αξιολογήσουμε μια νέα εφαρμογή αλγοριθμικών μοντέλων, Random Forest και Logistic Regression, που εκπαιδεύτηκαν σε δεδομένα σχετικά με την υγεία των ασθενών, για την πρώιμη διάγνωση και πρόβλεψη του ARDS. Η προσέγγιση μας επιτυγχάνει καλύτερα αποτελέσματα σε όλες τις μετρήσεις, σε σύγκριση με σχετικές δημοσιευμένες προσπάθειες που επίσης χρησιμοποιούν τη βάση δεδομένων MIMIC III για την ανάπτυξη προγνωστικών μοντέλων για ARDS. Συγκεκριμένα, και τα δύο αλγοριθμικά μοντέλα μας έχουν καλύτερη απόδοση στην πρόβλεψη ARDS, με κυρίαρχο το Random Forest με 10-fold cross validation, σύμφωνα με την περιοχή κάτω από την καμπύλη AUC (95,1%), την ακρίβεια (98,0%), την ειδικότητα (98,62%) και την ευαισθησία (96,25%).

Ευχαριστίες

Θα ήθελα να ευχαριστήσω από καρδιάς τον καθηγητή Βιοϊατρικής Πληροφορικής και Ηλεκτρονικής Υγείας, στο Ελληνικό Μεσογειακό Πανεπιστήμιο, κύριο Μανώλη Τσικνάκη ως σύμβουλο της διατριβής μου και συνεργαζόμενο Ερευνητή του Εργαστηρίου Υπολογιστικής Βιο-Ιατρικής, Ινστιτούτου Πληροφορικής, ΙΤΕ, ο οποίος με τις καινοτόμες ιδέες και το πλούσιο όραμα του, στάθηκε αρωγός σε αυτή την προσπάθεια, συμβάλλοντας σημαντικά στην διεξαγωγή της μελέτης και ολοκλήρωσης αυτής της εργασίας. Επίσης σε συνεργασία με τον επόπτη καθηγητή Επιστήμης Υπολογιστών και της διατριβής μου, στο Πανεπιστήμιο Κρήτης, κύριο Δημήτρη Πλεξουσάκη, Διευθυντή του Ινστιτούτου Πληροφορικής, ΙΤΕ, τους ευχαριστώ θερμά για την πλούσια υποστήριξη τους και παροχή σε υπολογιστικούς πόρους για τη συγκεκριμένη εργασία. Χωρίς εκείνους δεν θα ήταν δυνατή η διεξαγωγή όλων αυτών των πειραματικών διαδικασιών και κατ' επέκταση η ολοκλήρωση της επεκτάσιμης πλατφόρμας διαχείρισης μεγάλου όγκου δεδομένων που δημιουργήσαμε. Ακόμη, θέλω να ευχαριστήσω ιδιαιτέρως τον επισκ. καθηγητή του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης, κύριο Χαρίδημο Κονδυλάκη και τον κύριο Λευτέρη Κουμάκη, συνεργαζόμενους Ερευνητές του Εργαστηρίου Υπολογιστικής Βιο-Ιατρικής, Ινστιτούτου Πληροφορικής, ΙΤΕ για την συνεπίβλεψη τους σε αυτή την εργασία και τις πολύτιμες συζητήσεις που είχαμε, τις συμβουλές και την καθοδήγηση τους. Δε μπορώ να παραλείψω βέβαια τον αν. καθηγητή του Τμήματος Επιστήμης Υπολογιστών, του Πανεπιστημίου Κρήτης, κύριο Κώστα Μαγκούτη, διδάσκοντα σε συναφές μάθημα με τον τομέα εξειδίκευσης που επέλεξα, όπως επίσης και συνεργαζόμενο Ερευνητή του Ινστιτούτου Πληροφορικής, ΙΤΕ. Μπορώ να πω ότι το μάθημα του και συγκεκριμένα ο τρόπος που μου μεταλαμπάδευσε το περιεχόμενο, επισφράγισε την επιλογή μου να ασχοληθώ με Distributed and Scalable data store and management τεχνολογίες και υπηρεσίες. Αποτέλεσε καθοριστική έμπνευση για εμένα και τον ευχαριστώ θερμά για αυτό. Επιπλέον, θέλω να ευχαριστήσω εγκάρδια, τον ακαδημαϊκό σύμβουλο μου, Πρόεδρο στο Τμήμα Επιστήμης Υπολογιστών, του Πανεπιστημίου Κρήτης, καθηγητή κύριο Αντώνιο Αργυρό, επίσης συνεργαζόμενο Ερευνητή του ΙΠ, ΙΤΕ, για την αμέριστη κατανόηση και εμπιστοσύνη που έδειξε στα πρώτα μου βήματα, πιστεύοντας σε εμένα και την όρεξη που είχα για αυτό το μεταπτυχιακό. Ειλικρινά, ήταν τιμή μου να συνεργαστώ και να έχω δίπλα μου όλους αυτούς τους καταξιωμένους ανθρώπους του Ινστιτούτου Πληροφορικής, ΙΤΕ. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, Δημήτριο και Βασιλεία, τον αδερφό μου Φίλιππο, την Δάφνη και να τους αποδώσω φόρο τιμής για την τεράστια δύναμη ψυχής που έδειξαν παραμένοντας από την πρώτη στιγμή στο πλάι μου, όπως επίσης και δύο εγκάρδιους φίλους μου, Νικόλαο και Μιχαήλ, δίνοντας μου στήριξη και ελπίδα στις δύσκολες στιγμές που συνάντησα σε αυτό το όμορφο ταξίδι.

στους γονείς μου

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Challenges and Complications

Nowadays, with the advent of computer systems and its potential, the digitization of all clinical exams and medical records in the healthcare systems has become a standard and widely adopted practice [1]. Therefore, there is increased interest in developing big data technology in healthcare and biomedicine to manage massive collections of heterogeneous health datasets, such as electronic health records and sensor data, which are increasing dramatically. Furthermore, large scale data analytics can improve patient outcomes and personalized care, while reducing medical spending. Nevertheless, in the biomedical field, data volume is increasingly growing, and traditional methods cannot manage it efficiently. There are still challenges of Big Data analytics in healthcare systems that need to be identified. These challenges are categorized by volume which refers to high amounts of data, variety which emphasizes that data comes under different sources and formats, velocity which means that data is generated at a rapid pace and veracity which means that accurate and applicable data originates from trustable sources. These challenges are often encountered in management, analysis and storage of biomedical data and efforts to handle these growing datasets has stretched the limits of traditional healthcare information technology systems. Another characteristic of big data is its variability which indicates variations that occur in the data rates. An additional important aspect of big data infrastructures is complexity. Complexity arises from the fact that big data is often produced through various origins, which implies that many operations are being performed over the data. These operations include identifying relationships, cleaning and transforming data flowing from different sources (ICU mechanical ventilators, home mechanical ventilators, smart device sensors, etc.) (Figure 1.1).

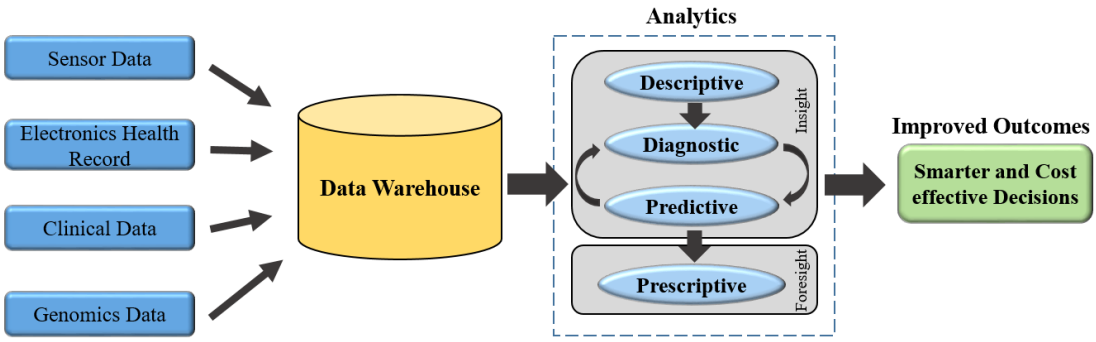

Figure 1.1: Workflow of Big data Analytics. Data warehouses store massive amounts of data generated from various sources. This data is processed using analytic pipelines to obtain smarter and affordable healthcare options [1]

Moreover, quantifying patient health and predicting future outcomes is a significant aspect in biomedical research. According to literature review [9, 12], if a patient's condition changes, physiological parameters (such as heart rate, blood pressure, respiratory rate, etc.) will change at varying degrees, too. Especially when we have to handle time series data, the difficulty of management increases if we consider that all data must be interconnected in some logical way. For instance, each patient in the hospital has a unique identification code, each hospitalization for each patient also has a unique identification code, each admission to the intensive care unit also has a unique identification code, as well as all values and physiological parameters recorded for all these unique patient codes have unique identification codes. Therefore, we realize that many different values and measurements for many patients, who are hospitalized for a long time in hospitals and in particular in intensive care units of these hospitals, with the frequency of their hospitalization constantly increasing, generate more and more new data. This data must be accurate in order to be applicable and able to identify clearly a clinical situation. The management of this data reveals at the same time the fact that we have to handle the challenges of variety, volume, velocity and veracity of this data. Thus, we perceive that even in big data infrastructures, complications may occur.

Studies [2, 52] have indicated that ARDS is a highly heterogeneous syndrome that may be composed of several distinct sub-phenotypes. Heterogeneity in population implies heterogeneity in relationships between explanatory variables and other

variables within data parts, posing serious challenges in building predictive models attempting to identify a common explanatory data pattern associated with an outcome. All of the above are necessary steps of a proper preprocessing and purification of data in order to reach the ultimate goal which is their modeling in machine learning algorithms for knowledge mining.

1.2 Significance of ARDS

Acute respiratory distress syndrome (ARDS) is a life-threatening disease, characterized by acute onset of hypoxia and pulmonary infiltrates, and incited by conditions such as sepsis, pneumonia, trauma and blood transfusion [3, 4, 5]. ARDS causes diffuse lung inflammation which leads to increased pulmonary vascular permeability, pulmonary edema, and alveolar epithelial injury [3]. According to relevant epidemiological investigations, the in-hospital mortality rate of ARDS is as high as 40% [4]. The acute respiratory distress syndrome (ARDS) was defined in 1994 by the American-European Consensus Conference (AECC) [5] and it is diagnosed based on three criteria: acute onset, moderate to severe impairment of oxygenation and bilateral lung infiltrates of a non-cardiac origin on chest x-ray or tomographic (CT) scan. The severity of the ARDS is defined by the degree of hypoxemia, which is calculated as the ratio of arterial oxygen tension to fraction of inspired oxygen (PaO2/FiO2). ARDS can be characterized as mild ($200 \leq$ (PaO2/FiO2) ≤ 300), moderate ($100 \leq$ (PaO2/FiO2) < 200) or severe ((PaO2/FiO2) < 100), which carries a mortality rate of 45%, as clarified by the Berlin definition of ARDS [5]. Determining the PaO2/FiO2 requires arterial blood gas (ABG) analysis. To calculate the PaO2/FiO2 ratio, the PaO2 is measured in mmHg and the FiO2 is expressed as a decimal between 0.21 and 1. As an example, if a patient has a PaO2 of 100 mmHg while receiving 80 percent oxygen, then the PaO2/FiO2 ratio is 125 mmHg (i.e., 100 mmHg/0.8). The PaO2/FiO2 ratio and positive end-expiratory pressure (PEEP) (>=5 cmH2O) are valuable clinical measures of the patient's respiratory status while receiving supplemental oxygen. It enables bedside clinicians to monitor the degree of hypoxemia, quickly detect early progression of respiratory failure, and intensify treatment.

1.3 ARDS Relation with COVID-19 Pandemic

Especially in the current situation of the COVID-19 pandemic, complications that could occur after the onset of this disease in human's life are really important and one of the most dangerous of these complications is ARDS [3, 6, 7]. As we explained above, ARDS is an important cause of morbidity and mortality worldwide. It may be developed after a direct injury to the lungs as aspiration, trauma or pneumonia (one of its consequences of COVID-19) or an indirect injury to other parts of the body as sepsis or pancreatitis. Specifically, severe COVID-19 presents viral pneumonia from severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) infection leading to ARDS [3]. According to this study [3], COVID-19 ARDS is diagnosed when someone with confirmed COVID-19 infection meets the Berlin 2012 ARDS diagnostic criteria of (i) acute hypoxaemic respiratory failure; (ii) presentation within 1 week of worsening respiratory symptoms; (iii) bilateral airspace disease on chest x-ray, computed tomography (CT). In 2020, there were approximately 100 million confirmed cases of people infected with the virus, including approximately 2 million deaths [53]. The most worrying point, however, is not the total number of infected people but the high rate at which this virus is transmitted to the human community. ARDS develops in 42% of patients presenting with COVID-19 pneumonia, and 61–81% of those requiring intensive care [3]. The cases of people infected with COVID-19 virus are increasing dramatically. As we can see in the charts below, the transmission rate in Figure 1.2 and Figure 1.4 and the deaths in Figure 1.3 and Figure 1.5 are quite high in relation to time.

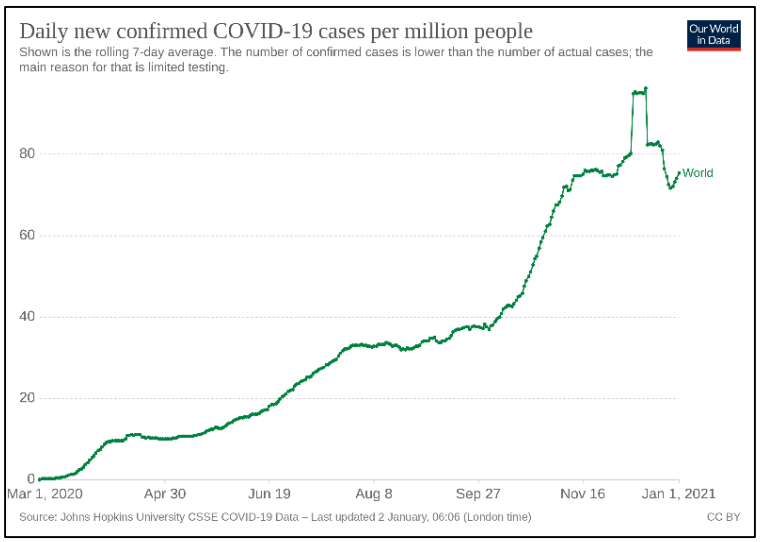

Figure 1.2: Daily New Confirmed COVID-19 cases.

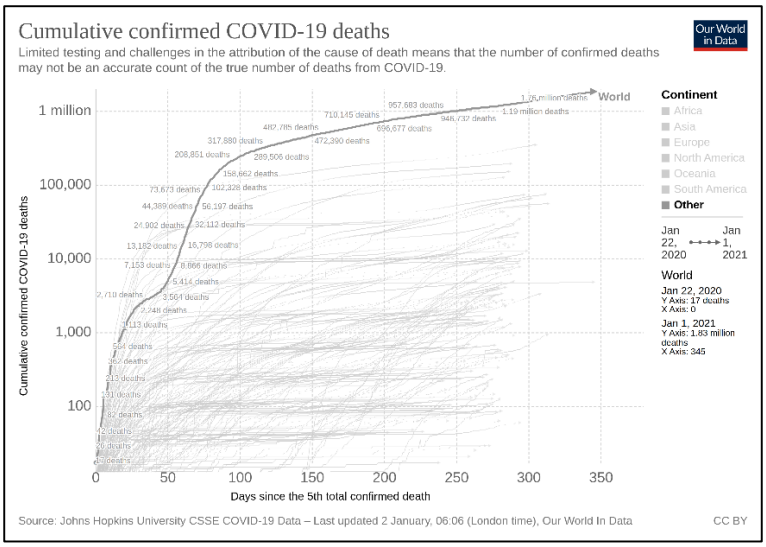

Figure 1.3: Cummulative Confirmed COVID-19 deaths.

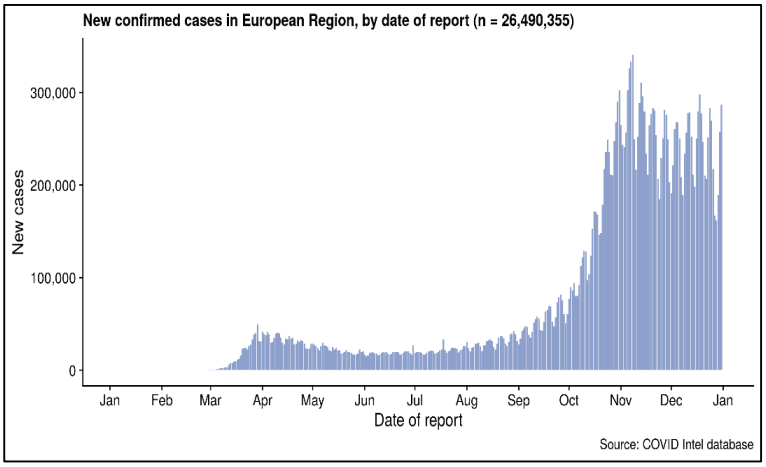

Figure 1.4: New Confirmed COVID-19 cases in European Region.

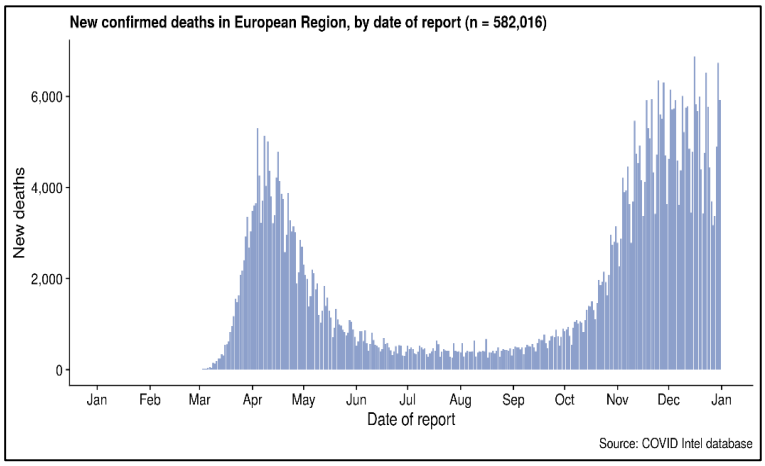

Figure 1.5: Daily New Confirmed COVID-19 deaths in European Region.

1.4 Significance of Scalable Infrastructures for Big Data Predictive Analysis

Early identification and management of ARDS can limit the relapse of lung disease and significantly improve patient outcomes [51]. The difficulty in analyzing and predicting ARDS outcomes originates from the fact that it is a highly heterogeneous

condition. ARDS involves the interaction of multiple risk factors, various vital signs, symptoms, past and current conditions [51]. This difficulty grows, as the volume of heterogeneous data that needs to be processed and analyzed to give us useful information grows. Therefore, identifying patients with COVID-19 related or not related ARDS is not an easy task. One reason is the large number of different types of data to be analyzed, structured or unstructured, that exist in the scientific community. Another reason is that, although every detail for analysis is biomedical data, the majority of the useful information in this field is not clear and obvious. Clinical data is not always well defined. Significant parts of the data are coded with specific numbers that define specific clinical situations. ICD-9 diagnosis codes [8] and ItemID codes [8] that identify mechanically ventilated patients with their laboratory measurements and their charts values, contribute significantly on identification of severity class of ARDS patients. There are study cases [2, 9, 10, 11] where Artificial Intelligence (AI) models are developed for ARDS identification and the authors omit to describe how their data were retrieved from big data sources and how they were actually managed in order to get used from predictive models. It is important when presenting the result and conclusion of a study, to list the main points of the recipe of this result or even difficulties that were encountered trying to reach it. Building an AI system for early identification of ARDS based on large volumes of data, hides complications and some specific implementation processes which are good to get known to the scientific community of data management and analysis. This tactic would help more researchers and analysts in the future to be able to reproduce such clinical scenarios in order to improve them and achieve even better results. We encountered such complications in our own work, using the MIMIC-III Clinical database [8], in order to examine, analyze and export ARDS related data.

It is worth noting that another major challenge in this area is data integration. More specifically there is a large need to integrate the data that is obtained for each patient into one system, as that will allow for fast data analysis, and give clinicians all the information they need to treat their patients in a perfect way. However, most of the time data is coded due to the patients' privacy rules, making it necessary to be decoded and normalized.

Early management of a disease requires early identification. To our knowledge, to date, there are a few reliable ways [9, 10] to anticipate which patients are likely to develop ARDS. Improved predictive validity is needed to enable reliable early identification and management of patients at risk for ARDS. All the above facts make the immediate identification of patients with ARDS a high priority of the scientific community. This action, however, presupposes the rapid processing and analysis of biomedical data by stable, scalable and fault tolerant systems governed by simplicity. Concluding that it is not trivial to develop AI models based on big data sources, we aimed that a systematic approach is required. Consequently, we need a scalable system for processing and analyzing large volumes of diverse data.

1.5 Our Approach

To this direction, in this thesis, we focus on deploying a scalable open source based platform that enables the development of machine learning predictive models for the early identification of patients with ARDS. We used this clinical data analysis scenario in order to evaluate our platform's infrastructure in real world use cases. We present an overview of our solution in this area and we focus on building a scalable architecture, using open source components. We identify the main problems occurring in such a big data infrastructure and we report experiences and solutions proposed. Moreover, we produced a complete workflow from defining the research question, retrieving the data using our scalable platform, preprocessing, building and training the model and eventually testing it. It was necessary to rely on well supported and documented open source, large scale data management infrastructures, for achieving quality data analysis results without being burdened from high financial costs and waste of time. Our platform may handle stream and batch data processing of clinical scenarios with fast, fault tolerant methods and support the development of machine learning models for the early identification of patients with life threatening diseases, such as ARDS.

As a big data test case for our infrastructure we used the last version (1.4) of

MIMIC-III database [8]. The raw size of this dataset is approximately 50 GB. It gave us the opportunity to explore the biomedical data in depth and develop specific cleaning and preprocessing methods, as a response to the problems listed above. Furthermore, we explored the relation of these parameters and focused on the identification of P/F and S/F [9, 12, 13, 14] ratio in combination with PEEP, according to literature review [2, 5, 9, 10, 15, 16, 17]. Using our infrastructure in building AI models for big data analysis, we developed algorithms for prediction of ARDS disease based mostly on various noninvasive parameters [9] in order to provide medical staff with the early and accurate knowledge of disease diagnosis. We used specific machine learning algorithms and a cross validation method to evaluate our predictive model based on the integrated data that retrieved from our infrastructure. More specifically, we first used WEKA [18] in order to see the potentials and the perspective of our techniques in our data and then we ran Random Forest with cross validation of 10 folds and Logistic Regression algorithms with specific tuning parameters. In comparison with recently relevant works and solutions on ARDS, we accomplished excellent performance in prediction results. To the best of our knowledge there is no other solution to enable uninterrupted integration and execution of modeling, using real world big data and make tests on real biomedical challenges like ARDS, outperforming on existing solutions.

Overall, the remaining of this thesis is structured as follows: Section 2 mentions the related work on other biomedical and healthcare big data infrastructures as well as review of efforts in developing predictive models of ARDS. Section 3 considers the background of the technologies that we used to build our platform's infrastructure and we explain the materials and methods of its architecture in detail. Moreover, we mention our experiences in the configuration procedure and all the challenges in addition with complications that we met. Section 4 describes the datasets we used and all the preprocessing and cleaning methods as a significant part of our work and summarizes the data analysis evaluation and the performance results. Finally, Section 5 concludes this work, noting future prospects.

Chapter 2

Related Work

In this section, we briefly survey various categories of related work on big data infrastructures and review of efforts in developing predictive models of ARDS.

2.1 Other Solutions on Big Data Infrastructures

Jacob McPadden, et al., in 2018 [11], demonstrated the implementation of a data science platform built on open source technology within a large, academic healthcare system and describe two computational healthcare applications built on this platform. According to the authors, their Hadoop based infrastructure provides a robust analytics platform where healthcare and biomedical research data can be analyzed in near real-time for precision medicine and computational healthcare use cases. They also report that several limitations exist in data science platforms like this, noting that it requires substantial technical expertise to use them to their full potential.

Jagreet Kaur et al., in 2018 [19], proposed a generic architecture for enabling AI based healthcare analytics platform by using open source technologies. They tried to show the importance of applying AI based predictive and prescriptive analytics techniques in the Health sector. They provided a systematic approach to support fast growing data of people with severe diseases. Their proposed architecture can support Artificial intelligence based healthcare analytics by providing batch and stream computing, extendable storage solution and query management.

Ankita Sharma et al., in 2019 [20], presented a Hadoop-based big data framework

(called BHARAT) integrating non-invasive magnetic resonance imaging (MRI), MR spectroscopy (MRS) as well as neuropsychological test outcomes to identify early diagnostic biomarkers of Alheimer's Disease. The proposed framework is partitioned into four major components, namely (1) Data Normalization, (2) Data Management, (3) Data Storage, and (4) Data Processing. They describe big data challenges in AD research and specifically regarding the large data size, the feature extraction in heterogeneous data, classification and missing values.

Van-Dai Ta et al., in 2016 [21], proposed a generic architecture for big data healthcare analytic by using open source technologies, including Hadoop, Apache Storm, Kafka and NoSQL Cassandra. Thy concluded that the combination of high throughput publish, subscribe messaging for streams, distributed real-time computing, and distributed storage system can effectively analyze a huge amount of health care data coming with a rapid rate.

Wullianallur Raghupathi and Viju Raghupathi, in 2014, have proposed "Big data analytics in healthcare: promise and Potential" [22]. In this paper the authors proposed the potential of big data analytics in healthcare. The paper provides an overview of big data analytics for healthcare practitioners and researchers, noting that still remain challenges to overcome.

Naoual El Aboudi et al., in 2018 [23], proposed an extensible big data architecture based on both stream computing and batch computing in order to enhance further the reliability of healthcare systems by generating real-time alerts and making accurate predictions on patient health condition. Based on the proposed architecture, a prototype implementation has been built for healthcare systems in order to generate real-time alerts. The suggested prototype is based on spark and MongoDB tools.

Mariam Kiran et al., in 2015 [24], presented an implementation of the lambda

architecture design pattern to construct a data-handling backend on Amazon EC2 [25]. This paper combines ideas from database management, cost models, query management and cloud computing to present a general architecture that could be applied in any given scenario where affordable online data processing of Big Datasets is needed. Authors had foreseen that the current industry would focus of using Spark SQL have aided further faster processing reducing some of the weaknesses of the Hadoop processing model [26].

Most of the above related works are referring to generic architectures and platforms without a real use case to prove their value. Nevertheless, with the push for population-wide research initiatives such as the COVID-19 ARDS [3, 6, 7] and the mortality of already well know ARDS, that will rely on large, complex, relational data, institutions need to develop systems that can adequately scale to handle the data inflow and provide sufficient capacity for analytic needs. Despite this fact, any new approaches must be mindful to the privacy and reliability requirements associated with healthcare data. Therefore, we present a use case that highlights the architecture and implementation of our biomedical data science platform and enables scalable, integrated, fault tolerant and attentive to privacy healthcare analytics. These strategies imply current best practices for data management, system integration, and distributed computing, maintaining a high level of credibility and fault tolerance.

2.2 Review of Efforts in Developing Predictive Models of ARDS

Pengcheng Yang et al., in 2020 [9], proposed a new method for identifying the acute respiratory distress syndrome disease based on noninvasive physiological parameters. According to their study, arterial gas blood is required in order to define the ratio of partial pressure of arterial oxygen to fraction of inspired oxygen (PaO2/FiO2 ratio) for ARDS prediction. They used the MIMIC-III database and they proposed an algorithm based on non-invasive physiological parameters which

helps in the estimation of P/F levels to aid in the ARDS disease diagnosis. They applied machine learning methods in co-operation with specifically feature selection filters in order to study more accurately the correlation in plenty of noninvasive parameters from patients that leads to the identification of ARDS disease. Moreover, they used cross-validation techniques on their machine learning methods in order to measure and approve the performance of their algorithms for various feature subsets. More specifically, they used XGBoost which is a gradient boosted tree model with 10-fold cross validation and they achieved satisfying results on the performance of ARDS identification, with the sensitivity of 84.03%, the specificity of 87.75% and the AUC of 0.9128. As part of feature extraction they discriminated some risk factors that contribute significantly to any ARDS prediction model. According to the Berlin Definition [5], they stated positive end-expiratory pressure (PEEP) >=5 cmH2O and PaO 2/FiO 2 ratio (P/F ratio) <= 300 mmHg as the most important criteria for ARDS classification. They categorized these criteria in three states of severity, namely, mild (200 < arterial oxygen partial pressure (PaO2)/ fraction of inspired oxygen (FiO2) (P/F) <=300), moderate (100 < P/F <= 200), and severe (P/F <= 100), according to the level of oxygenation index (P/F). They mainly used blood gas analysis in order to measure PaO2 that contributes to the P/F value for the ARDS severity evaluation. According to all the above, their selection criteria contained patients with P/F < 300 on the first day of entering the ICU, patients older than 16 years old, with 48h minimum LOS (Length of Stay) in the ICU and mechanically ventilated at some time during their presence in the ICU.

Sidney Le et al., in 2020 [10], developed and evaluated an application of gradient boosted tree models trained on patient health record data for the early prediction of ARDS. They used MIMIC-III database for their analysis and they created XGBoost gradient boosted tree models in order to achieve early ARDS prediction. They extracted clinical variables and numerical representations of radiology reports as data source to their models and applied 10-fold cross validation. They followed specific methods and selection criteria in order to feed their model with important and useful features. More specifically, they included patients with at least 18 years of age, following the Berlin Definition [5]. According to this and the co-occurrence

of two parameters, (1) Positive end Expiratory Pressure (PEEP) >= 5 cmH 2O and (2) PaO 2/FiO2 ratio (P/F ratio) <= 300 mmHg, they examined the patient data. According to their results, their classifier demonstrated AUROC performance of 0.843, 0.858, 0.810, and 0.790 for early ARDS prediction on the test set at 0 hours, 12 hours, 24 hours, and 48 hours prior to onset, respectively.

The same time period, in 2020 [27], Elizabeth Sanchez et al., studied about Acute Respiratory distress syndrome (ARDS). They created a predictive model using baseline characteristics in order to identify patients at high risk of having severe ARDS. The selection criteria according to the Berlin Definition [5] included the ratio PaO2/FiO2 <= 100 mmHg. Moreover, FiO2, and positive end-expiratory pressure (PEEP) were categorized by the authors as useful variables to predict persistent severe ARDS. They used random forest and regularized logistic regression with an L1 penalty [Least Absolute Shrinkage and Selection Operator (LASSO)] techniques in order to identify predictive variables of persistent severe ARDS. They presented their results concluding that PaO2:FiO2, FiO2 and positive end-expiratory pressure (PEEP) at enrollment were useful predictive variables.

Wang et al. [2], in 2019, studied about ARDS identification for enhanced machine learning predictive performance. They used MIMIC-III database from which they extracted adult patients (age >= 18 years old). According to the authors, ICD-9 diagnosis codes and procedure codes that are used for the identification of mechanically ventilated patients are very important factors in ARDS prediction. They extracted PaO2, FiO2, and PEEP from the dataset and they used Berlin Definition criteria [5], setting as basic parameters PaO2/FiO2 ratio ≤ 300 and PEEP >= 5 cmH2O. They used much more clinical variables in case of analysis, i.e. mean airway pressure (MAP), PaCO2, tidal volume, platelet count, total bilirubin; minimum of sodium, glucose, albumin, hematocrit, systolic blood pressure (SBP); maximum of temperature, heart rate, white blood cell (WBC) count, creatinine. They developed predictive models, including gradient boosted machine (GBM) and random forest (RF) with cross validation as part of parameters' tuning.

Xue-Shu Yu et al. [29], in 2019, studied about risk factors for acute respiratory distress syndrome (ARDS) and found out that lung heart pressure index is a one of them. They used MIMIC-III database from where they selected ARDS patients who had undergone mechanical ventilation for more than 48 hours (using structured query language SQL queries). They collected demographics and useful variables via data extraction such as age, sex, ethnicity (white, black, other), body mass index (BMI), smoking status, ARDS severity (according to the Berlin definition), disease severity scores (Sequential Organ Failure Assessment [SOFA]), vital signs (MAP, respiratory rate [RR], heart rate [HR], Pao2/Fio2), laboratory values (pH, lactate, red cell volume distribution width [RDW]) and ventilator parameters (lung-heart pressure index (LHPI, [100%*DP/MAP]), DP, MP, platform pressure (Pplat). The primary outcome of their study was 30-day mortality from the date of ICU admission. In order to achieve accurate predictions, they used random forest and logistic regression models with 10-fold cross validation, resulting in ARDS identification and presenting the mortality of ARDS in patients. Their study showed that the LHPI was a powerful prognostic indicator of 30-day mortality in ARDS patients, and its predictive discrimination was better than that of driving pressure DP and mechanical pressure MP.

Emilia Apostolova et al., in 2019 [51], used a combination of free-text and structured data in order to create an Acute Respiratory Distress Syndrome (ARDS) analytics model. They used MIMIC-III database deriving patients-specific contextual ARDS risk factors, making use of deep-learning methods on ICD and free-text clinical data. They extracted structured data from the first 24 hours of admission, such as vital signs and lab results, building an ARDS patient prediction model and an ARDS patient mortality prediction model. The age of patients that attempted to predict ARDS was above 18 years old with ICD-9 codes for severe acute respiratory failure and use of continuous invasive mechanical ventilation. The structured data included in this analysis consists of anion gap (anion gap), albumin, bands, bicarbonate, bilirubin, creatine, chloride, glucose, hematocrit, hemoglobin, lactate, platelet, potassium, partial thromboplastin time (ptt), international normalized ratio (inr), prothrombin time (pt), sodium, bun, white blood cell count (wbc), heart rate (heart rate), systolic blood pressure (sysbd), diastolic blood

pressure (diasbp), mean blood pres- sure (mean bp), respiratory rate (resperate), body temperature (tempc), peripheral capillary oxygen saturation (spo2), body mass index (bmi), gender, age, urine output. All variables were measured over the first 24 hours of ICU admission, because according to the authors it has been reported that ARDS develops at a median of 30 hours after hospital admission. They referred to a variety of ICD9 codes that were utilized in their study and explained the significance of these codes in the core of the study.

Jianfeng Xie et al. [28], in 2018, studied about acute respiratory distress syndrome prediction (ARDS) in order to establish a modified ARDS prediction score (MAPS) helping clinicians in the early recognition of ARDS in patients who need to be admitted to the ICU. They used data from 13 tertiary hospitals in China. The main risk factor that used in their selection criteria was patients with PaO2/FiO2 <= 300 mmHg and PEEP ≥5 cmH2O. They used univariate and multivariate logistic regression models in order to make accurate predictions, resulting in various statistics about patients and concluding that MAPS discriminated patients who developed ARDS from those who did not, with an area under the curve (AUC) of 0.809.

Chapter 3

Platform Architecture and Deployment

This section provides an overview of the procedures described in the adopted methods and our platform's architecture, which are visually summarized in Figure 3.6.

3.1 Methods and Categories

Big data technologies have received great attention due to their successful handling of high volume data compared to traditional approaches. Big data frameworks support all kinds of data, structured, semi structured, and unstructured data, while providing several features. Those features include predictive model design and big data mining tools that allow better decision making process through the selection of relevant information.

Big data processing is characterized by two categories: batch processing and stream processing. The first category, batch processing, is based on analyzing data over a specified period of time and it is mainly used when there are no constraints regarding the response time. Specifically, this category aims to process high volume of data by collecting and storing batches to be analyzed and generate results at a fast pace. The second category, stream processing, is preferable for applications that require real-time feedback.

Batch computing requires ingesting all data before processing it in a specified time. For several years, Mapreduce [30] represents a widely adopted solution in the field

of batch processing. It operates by splitting data into small pieces that are distributed to multiple nodes in order to produce intermediate results. Once data processing by nodes is finished, outcomes will be aggregated in order to generate the final results.

As regards, stream computing in real applications such as healthcare, a high quantity of data is generated continuously. When the need for real time stream processing increases, data analysis takes into consideration continuous manner of data to change over time and being trained in these data changes, manages them accordingly. Indeed, storing large quantities of data for further processing may be challenging in terms of memory resources. Moreover, real applications tend to produce noisy data which contain missing values along with redundant features, making data analysis complicated, as it requires significant computational time [23]. Stream processing reduces this computational load by performing simple and fast computations for small amounts of data, spending only a few seconds in computations.

3.2 Background and Materials

This section presents essential background information related to the topic described in this thesis and we explain the main concepts. In order to contribute and support the informatics needs for the next generation of computational health research, novel approaches to data storage and analysis are necessary.

3.2.1 Apache Spark Ecosystem

Fortunately, several applications have emerged that begin to address the key challenges in big data processing, such as distributed data storage and scalable processing capacity. One example is the Apache Spark framework, which contains a set of open source modules designed specifically for these tasks [31, 32]. The

goal of these platforms is to create a central repository, called data lake, which can store raw data in its native format for later search, retrieval, and analysis. However, researchers and clinicians in the healthcare region looking to leverage modern big data architectures, are faced with particular challenges in implementation and little guidance or evidence on the use of these platforms in parallel with production environments.

Apache Spark is a unified open source and cluster computing analytics engine for large scale data processing [32]. More specifically, it is an open source analytics engine used for big data which is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries, and streaming. The main feature of Apache Spark is its in-memory cluster computing that increases the processing speed of an application. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. It can handle both batches as well as real-time analytics and data processing workloads. Apache Spark started in 2009 as a research project at the University of California, Berkeley. Researchers were looking for a way to speed up processing jobs in Hadoop systems. It is based on Hadoop MapReduce [26, 30, 33, 34] and it extends the MapReduce [30] model to efficiently use it for more types of computations, such as interactive queries and stream processing that mentioned above. Spark provides native bindings for the Java, Scala, Python, and R programming languages. In addition, it includes several libraries to support build applications for sequential querying (SQL), machine learning [MLlib], stream processing [Spark Streaming], and graph processing [GraphX]. Apache Spark consists of Spark Core and a set of libraries. Spark Core is the heart of Apache Spark and it is responsible for providing distributed task transmission, scheduling, and I/O functionality.

Spark_Core

Spark Core is the base engine for large scale parallel and distributed data processing. Further, additional libraries which are built on the top of the core allows diverse workloads for streaming, SQL, and machine learning. It is responsible for memory management and fault recovery, scheduling, distributing and monitoring

jobs on a cluster and interacting with storage systems.

Spark Streaming

Spark Streaming is the component of Spark which is used to process real-time streaming data. Thus, it is a useful addition to the core Spark API. It enables high-throughput and fault-tolerant stream processing of live data streams.

Spark SQL

Spark SQL is a new module in Spark which integrates relational processing with Spark's functional programming API. It supports querying data either via SQL or via the Hive Query Language. For those who are familiar with RDBMS, Spark SQL is an easy transition from their earlier tools where they can extend the boundaries of traditional relational data processing.

Mllib (Machine Learning)

MLlib stands for Machine Learning Library. Spark MLlib is used to perform machine learning in Apache Spark.

We decided to use Apache Spark against other technologies such as the aforementioned and widely used from related works [1, 20, 21, 35, 36] Hadoop, because of specific features that spark grants us. We mentioned the most important components of it above, concluding with the six main ones that are depicted in Figure 3.1:

- **Speed**
  Spark runs much faster than Hadoop MapReduce for large scale data processing. It is also able to achieve this speed through controlled partitioning.

- **Powerful Caching**

  Simple programming layer provides powerful caching and disk persistence capabilities.

- **Deployment**

  It can be deployed through Mesos [37], Hadoop via YARN [38], or Spark's own cluster manager.

- **Real-Time**

  It offers Real-time computation and low latency because of in-memory computation.

- **Polyglot**

  Spark provides high-level APIs in Java, Scala, Python, and R. Spark code can be written in any of these four languages. It also provides a shell in Scala and Python which is extremely helpful in cases of server or cluster deployment.

- **Scalable**

  It can be easily paralleled to any server or cluster of servers.

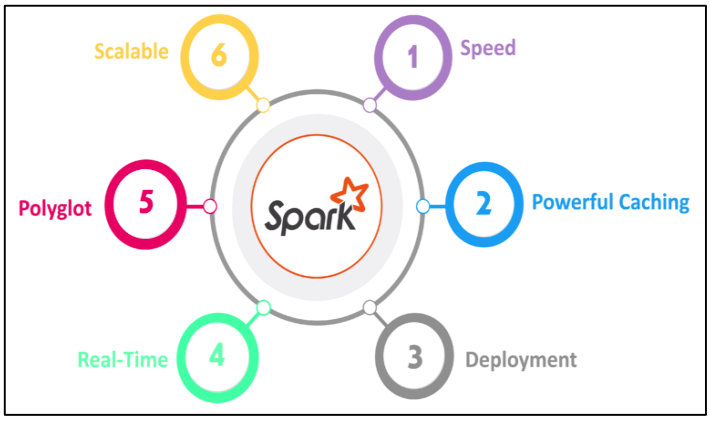

Figure 3.1: Apache Spark most important characteristics.

3.2.2 Apache Kafka and Zookeeper

Apache Kafka and Zookeeper [39] are two open source tools that work together to serve primarily stream processing scenarios. Apache Kafka [39] is a distributed, scalable, high performance messaging system that was developed for collecting and delivering high volumes of log data with low latency. We can observe a Kafka messaging queue workflow in Figure 3.2. Being open source means that it is essentially free to use and has a large network of users and developers who contribute towards updates, new features and offering support for new users. Kafka [40] is designed to run in a "distributed" environment, which means that it runs across several (or many) servers, leveraging the additional processing power and storage capacity that this brings. Kafka was originally created at LinkedIn [41], where it played a part in analyzing the connections between their millions of professional users in order to build networks between people. It was given open source status and passed to the Apache Foundation – which coordinates and oversees development of open source software – in 2011.

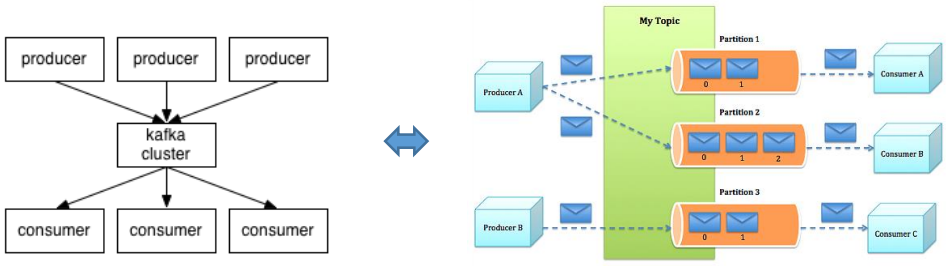

Figure 3.2: Kafka Messaging Queue Workflow.

ZooKeeper is a top-level software developed by Apache that acts as a centralized service and is used to maintain naming and configuration data and to provide flexible and robust synchronization within distributed systems. As we can observe in Figure 3.3, ZooKeeper keeps track of status of the Kafka cluster nodes and it also keeps track of Kafka topics and partitions. The data within ZooKeeper is divided across multiple collections of nodes and this is how it achieves its high availability and consistency. In case a node fails, ZooKeeper can perform instant

failover migration; e.g. if a leader node fails, a new one is selected in real-time by polling within an ensemble. A client connecting to the server can query a different node if the first one fails to respond. Kafka uses ZooKeeper to manage the cluster. ZooKeeper is used to coordinate the brokers/cluster topology.

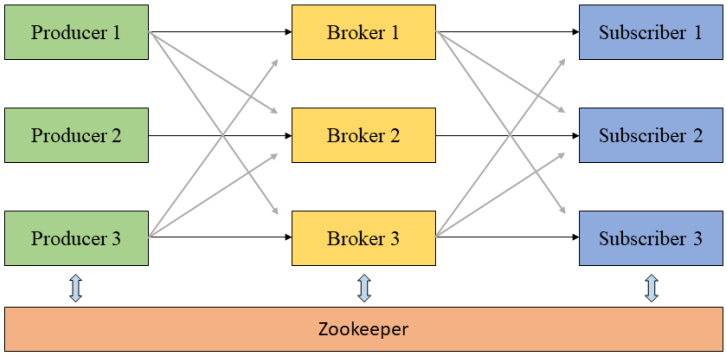

Figure 3.3: Kafka ZooKeeper Architecture.

3.3 System Configuration

This section presents the system configuration and build, regarding the hardware and the software parts, summarizing the important points of each one through experimental evaluation results.

3.3.1 Hardware Configuration

In the context of tests and experiments about the scalability of our platform we used a desktop personal computer with a 64bit Intel® Core™ i7 CPU at 3,60 GHz with total 4 processing cores, 8 threads, 16 GB Memory and 1 TB of storage, running Linux Ubuntu 18.04. The Apache Spark platform was deployed on a single node research laboratory Server at FORTH Computational Bio-Medicine Laboratory, running Linux Ubuntu 18.04. This server has a 64bit Intel® Xeon® CPU at 2,60 GHz (3,50 GHz Max) with total of 24 processing cores, 48 threads, 250 GB Memory and approximately 50 TB of storage. Moreover, Apache Kafka with

Zookeeper was deployed on the same server.

3.3.2 Software Configuration

We configured Apache Spark appropriately to run every piece of code distributed, making full use of all available resources. To be more specific we divided our available resources, setting 12 cores and 60 GB memory on each worker node as we can see at Figure 3.4 below. In this figure we can also observe how spark environment's interior components interact and cooperate with each other. Spark uses a master-slave architecture with one coordinator and four distributed workers. The central coordinator is called Driver. The Driver communicates with a number of distributed workers called executors. Driver and its executors compose a Spark application. A Spark application runs on a set of machines or processors using a service called Cluster Manager.

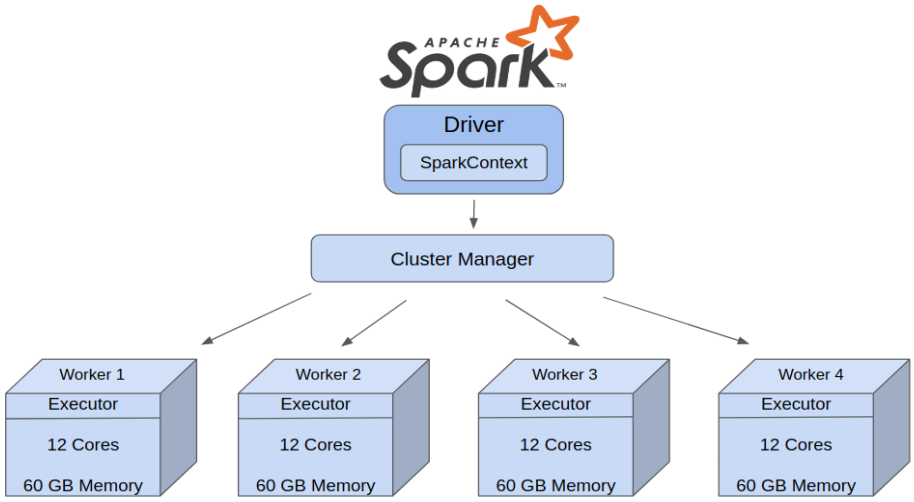

Figure 3.4: Parallelization and System Configuration of Apache Spark.

Our architecture is based and shares the most characteristics of Lambda architecture [24]. Lambda is one of the most discussed architecture patterns in the data science space that is designed to address robustness, scalability and fault tolerance of big data systems. It contains batch layer that has two major tasks: (a) managing

historical data, (b) computing results (ML) and speed layer that manages near real time data and provides results in a low-latency. (see Figure 3.15)

Figure 3.5: Apache Kafka with Zookeeper and Spark integration.

Kafka is the best performing framework for queueing slightly large messages and CPU load. The Direct (D)Stream integration with Spark means that messages coming from Event Producers, are being transferred directly from the Kafka server to the Spark workers (see Figure 3.5). It is also worth noting that Kafka is not intended for handling large file sizes (>1 Mb) in terms of velocity.

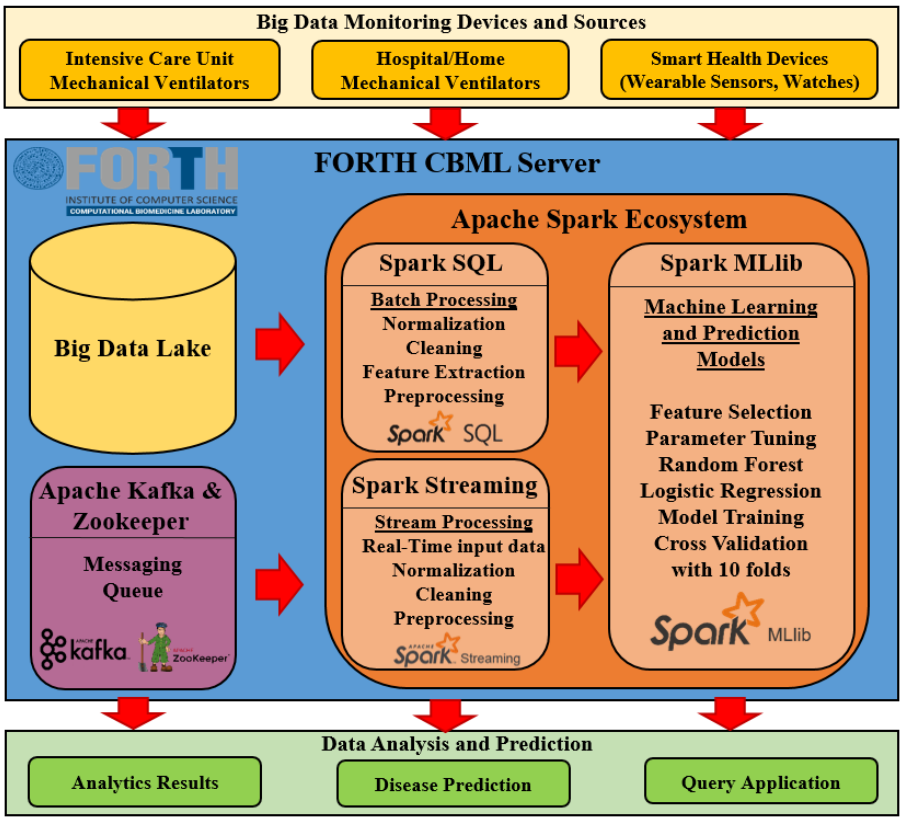

Figure 3.6: Architecture of our Integrated Scalable Platform for Big Data Analytics.

3.3.3 Experimental Evaluation of System Configurations on Query Application

Initially, we created a plan with the available resources. Consequently, combining our knowledge in SQL and Scala language we created a complete query (subsystem) application in Spark SQL where the user may ask questions in any large database such as MIMIC-III and get answers in a much shorter time than he would get in a conventional Postgresql. The advantage of our query application is that it takes full benefit of the possibilities offered by Spark in terms of parallelization and cooperation of all available resources of a given server or even a cluster of servers, with the feeling of fault tolerance that governs spark. It is also noteworthy that very easily a user with good knowledge in information systems management, can configure the spark and parallelize the available resources depending on their needs and the materials available. We can realize how important the scalability of such a platform is by observing the execution times (Duration) in Figures [3.7 – 3.12], where the same query appears to run on a personal computer (Figure 3.7 – 3.9) with few resources (Table 3.1, System Conf. 1) and respectively to run on the FORTH CBML server (Figure 3.10 - 3.12) with much more resources (Table 3.1, System Conf. 2). Execution time was captured through the spark Web UI which allows us to monitor our application's status and resource consumption in real time, providing us with a wealth of useful information. This fact proves that our platform which is built on open-source technologies, is scalable and capable to serve the management and processing of big data with speed and stability, depending on the available computing resources.

Desktop personal computer vs **FORTH CBML Server**						
System Conf.	**Workers**	**Cores**	**Memory (GB)**	**Memory per Executor**	**Execution time (min)**	**Stages: Succeeded/Total**
1	2	8	12	6	25	41/56
2	4	48	240	60	7	41/56

Table 3.1: Desktop personal computer and FORTH CBML Server configurations

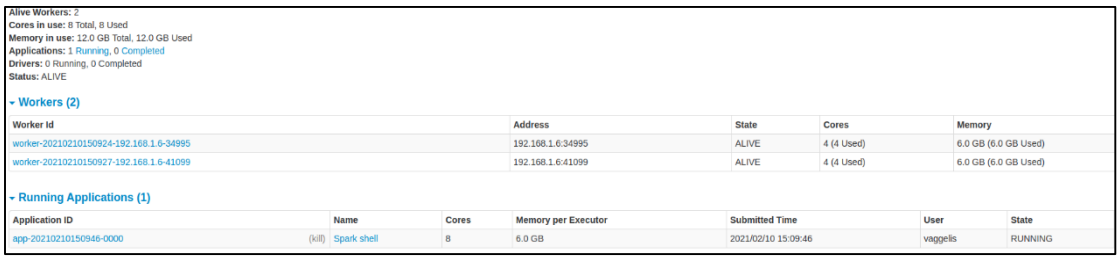

Figure 3.7: Spark Web UI, system configuration on personal computer.

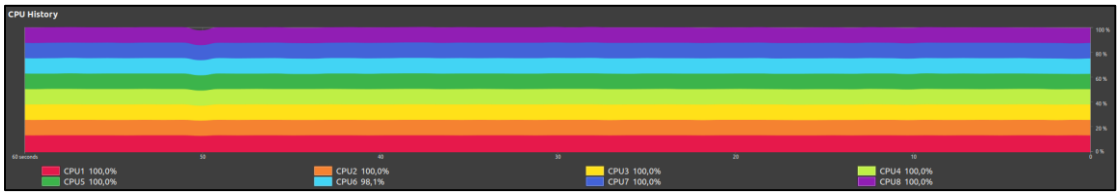

Figure 3.8: CPU usage of personal computer.

Figure 3.9: Execution time on personal computer.

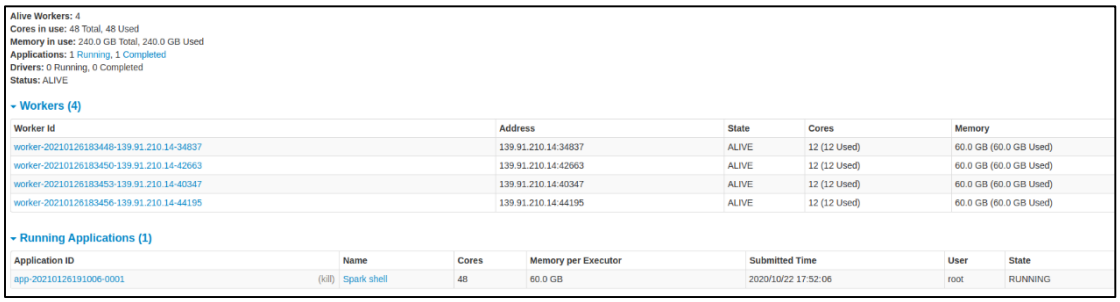

Figure 3.10: Spark Web UI, system configuration on FORTH CBML server.

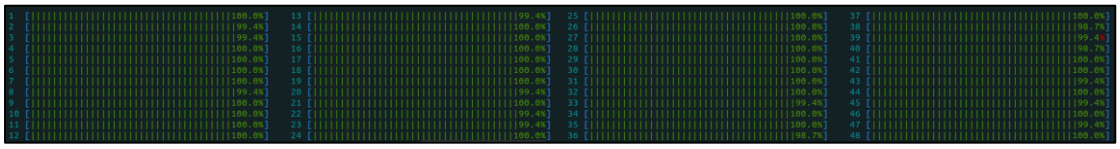

Figure 3.11: CPU usage on FORTH CBML Server.

Figure 3.12: Execution time on FORTH CBML Server.

3.4 Data Processing Clinical Scenarios

This section presents the main points of our platform's architecture, summarizing the complications and the prospects. Our architecture depicts two different big data processing clinical scenarios. To take advantage of the infrastructure speed we use one of the low level APIs that Spark provides us, which uses resilient distributed datasets (RDDs). In both scenarios of clinical data processing, we used Scala language because of its performance advantage over Python. The name Scala comes from the English phrase "scalable language", which states that it is designed to grow in line with the needs of its users. Dataframes and especially RDDs perform better in Scala because they are executed directly on JVM, avoiding a significant communication (Python-JVM) time overhead.

3.4.1 Batch Processing Clinical Scenario

The first clinical scenario is called batch processing. In this scenario we get the data from sources like mechanical ventilators of ICU, hospital rooms, homes and data from smart health devices such as wearable sensors, smart watches etc. Data is stored in the available storage space called the data lake. Then we load our data from the data lake in Spark and using Spark SQL we start the procedure of normalization, cleaning, feature extraction and generally the preprocessing. When the data have been preprocessed and all the significant features for analysis have been extracted successfully, enter the Spark MLlib and the machine learning process begins. The data are divided into input features and output class values and enter in the respective machine learning algorithm where the predictive model is

built. This model will give us the analytics and the prediction for the respective disease. In our work and specifically in the spark application we created in order to evaluate the analytics efficiency in our data and predict patients with ARDS, we used Random Forest with 10-fold cross validation and Logistic Regression algorithms. Moreover, we may run dynamic queries on patient clinical data at any time through the query (subsystem) application that our infrastructure supports upon Spark SQL. We present below a workflow on Figure 3.13 and some examples of the queries that a clinician might run in both clinical scenarios in order to perform a statistical analysis and clinical questions on this dataset:

- ARDS related queries:
    - Give me <u>all the</u>/<u>specific</u> distinct ARDS cases in newborn patients and demographics information about them.
    - Give me <u>all the</u>/<u>specific</u> distinct Acute Respiratory Failure cases in all patients.
    - Give me <u>all the</u>/<u>specific</u> admissions where patients have PEEP >= 5.
    - Give me <u>all the</u>/<u>specific</u> admissions where patients have PaO2 and FiO2.
    - Give me <u>all the</u>/<u>specific</u> Heart Rates from table chartevents grouped by icustay_id, etc.

- General MIMIC-III related queries:
    - Give me the number of patients who died while the patients were in the hospital and who survived
    - Give me the maximum length of stay in the ICU for specific patients
    - Give me the maximum length of stay in the ICU for each patient
    - Give me the maximum length of stay in the ICU for each patient where the maximum length of stay is < 10 days
    - Give me the number of male and female patients

Figure 3.13: Batch Processing Clinical Scenario.

3.4.2 Stream Processing Clinical Scenario

The second clinical scenario is called stream processing. In this scenario we get data from the same sources, however the data follow a different "path". Specifically, new data coming in real time, get into Apache Kafka under the ZooKeeper support and are being queued and processed as topics. Every topic from Kafka gets into Spark and specifically into Spark Streaming where it can then be processed and follow the same procedures as in the first scenario (cleaning, preprocessing, etc.) depending on the algorithms we run. After this stage, we can also take as output real time analytics, prediction of a disease and make dynamic queries on data. Therefore, we have created a complete integration as we can observe the workflow on Figure 3.14.

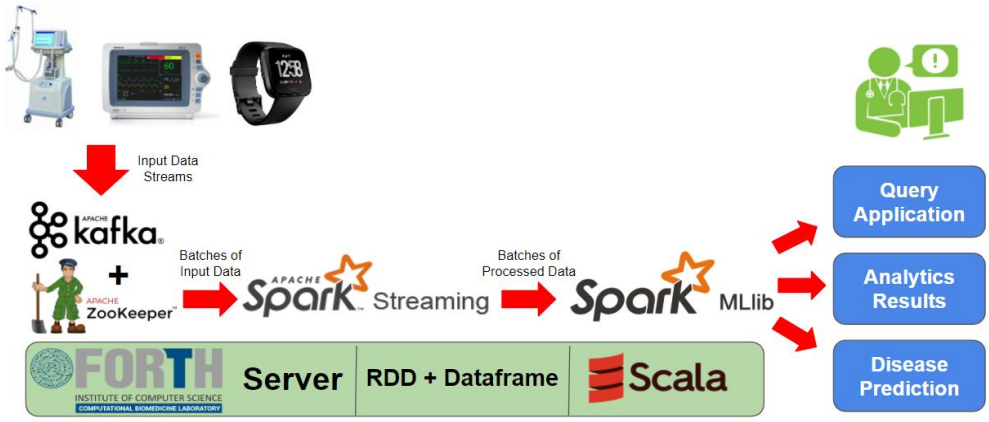

Figure 3.14: Stream Processing Clinical Scenario.

3.4.3 Complications and Experiences

Regarding the batch processing part, we made a first allocation and parallelization of resources and started running the first queries on the data in order to test our infrastructure and collect useful information for data analytics use. We noticed some technical issues regarding the configuration of Spark when we started to scale up using our large database for analysis. However, apache spark has a large supporting community and we managed to solve any problem without spending much time to search for the solution.

Regarding the stream processing part, we had difficulty in connecting Apache Kafka with Spark. We did this in order to be able to send real-time data to our system and have them filtered and managed first by Kafka and then in queue form to get into Spark. Unfortunately, there is not much community in Apache Kafka that has dealt with the specific issue, as there is also not much community in Apache Flink [44, 45] and Apache Druid [42]. Apache Flink is an open source system for processing streaming and batch data. It is an excellent work in the field of data management and can compete with Apache Spark in individual use cases. We tried to connect Apache Kafka with Flink to see if it is better than Apache Spark in some cases (performance, stability, community support), however the difficulties we encountered in its configuration and communication with Kafka, made us leave it out of our architecture. We performed some use cases individually on it, but the Kafka-Flink connection procedure cost us much time ending up with unsolved problems so we skipped it. Druid [43] is an open source database that is most often used for powering use cases where real-time ingest, fast query performance, and high uptime are important. As such, Druid is commonly used for powering GUIs of analytical applications, or as a backend for highly-concurrent APIs that need fast aggregations. We installed and tried to set up Apache Druid in order to test it on streaming data and see if it fits our needs, however it still has poor community support, so we did not let any other configuration difficulties cost more valuable time to us. After all, our scope was to create a system that is simple to use and at the same time stable. At Spark-Kafka connection, we encountered some difficulties

which did not bother us much because fortunately Apache Spark has a large, rich community and support as well as excellent documentation as we described above. It is very important when we work with open source technologies with which we want to deploy a scalable infrastructure, to have well written documentation and support to rely on.

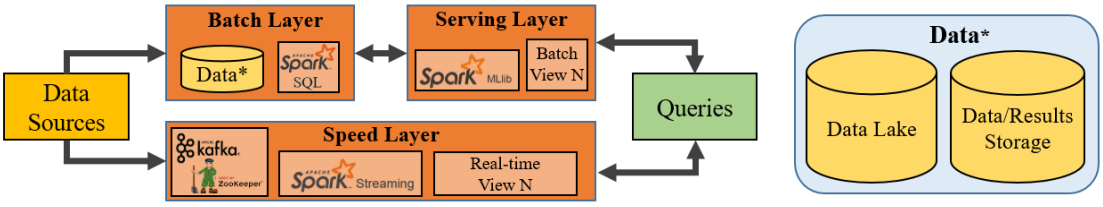

Figure 3.15: Lambda Architecture.

Data Sources: Data can be obtained from a variety of sources, which can then be included in the Lambda Architecture for analysis. **Batch Layer**: This component saves all data coming into the system as batch views in preparation for indexing. **Serving Layer**: This layer incrementally indexes the latest batch views to make them query by end users. **Speed Layer**: This layer complements the serving layer by indexing the most recently added data not yet fully indexed by the serving layer. **Query**: This component is responsible for submitting end user queries to both the serving layer and the speed layer and consolidating the results.

Chapter 4

Using the Architecture to Build Predictive Models

In this section we describe the real world datasets used in our experiments and the data analysis application build procedure. We present all the complications and challenges that we managed to handle eventually with specific techniques and methods. We summarize this chapter with our experimental evaluations and results.

4.1 Data Selection and Sources

As we mentioned above, we needed a large enough dataset to build the biomedical application of predictive analysis in order to evaluate our platform and validate our study. Consequently, we used MIMIC-III (Medical Information Mart for Intensive Care - III) [8, 49] clinical database, which is a large, freely available database comprising information relating to patients admitted to critical care units at a large tertiary care hospital. The MIMIC-III clinical database captures over a decade of intensive care unit (ICU) patient stays at Beth Israel Deaconess Medical Center. An individual patient might be admitted to the ICU multiple times over the years, and even within a single hospital stay could be moved in and out of the ICU multiple times. This is a fact that generates many non-unique identification codes regarding the patient's activity. As we can observe in Figure 4.1, data includes vital sign measurements obtained at the bedside, demographics, medications, laboratory measurements and test results, records of arterial blood gas levels observations and notes charted by care providers, fluid balance, procedure codes, diagnostic codes, imaging reports, hospital length of stay, survival data, and other clinical variables. The raw data in MIMIC-III, with size of 50 GB, provide fine-grained timestamps

for each laboratory measurement and recorded vital sign. However, most measurements are infrequent (e.g. blood tests of interest may be run every few hours at most), meaning each variable's raw time-series is quite sparse [46] and this is a fact that generates many missing values. Each measurement in the MIMIC-III database is associated with a unique ItemID, as specified by the original EHR software. These raw ItemIDs are not robust to changes in software or human data entry practices. For example, "HeartRate" may be recorded under ItemID 211 (using CareVue EHR systems before 2008) or under ItemID 220045 (using MetaVision EHR software after 2008). We thus developed a manually curated clinical taxonomy designed to group semantically equivalent ItemIDs together into more robust "clinical aggregate" features. These aggregate representations reduce overall data missingness and the presence of duplicate measures. Therefore, we have included as much as possible all the coded values that interest us. Appendix D (Tables A.4, A.5), details the proposed clinical taxonomy about the MIMIC-III encoded features. Ventilator settings were documented by respiratory therapists at intubation and as ventilator settings were adjusted. International Classification of Diseases, Ninth Revision (ICD-9) codes were documented for specific diseases as required by hospital staff on patient discharge. Each row associated with one ItemID (e.g. 212) corresponds to an instantiation of the same measurement (e.g. heart rate). We used ICD-9 and ItemID codes in order to filter and clean our data (see Appendix D).

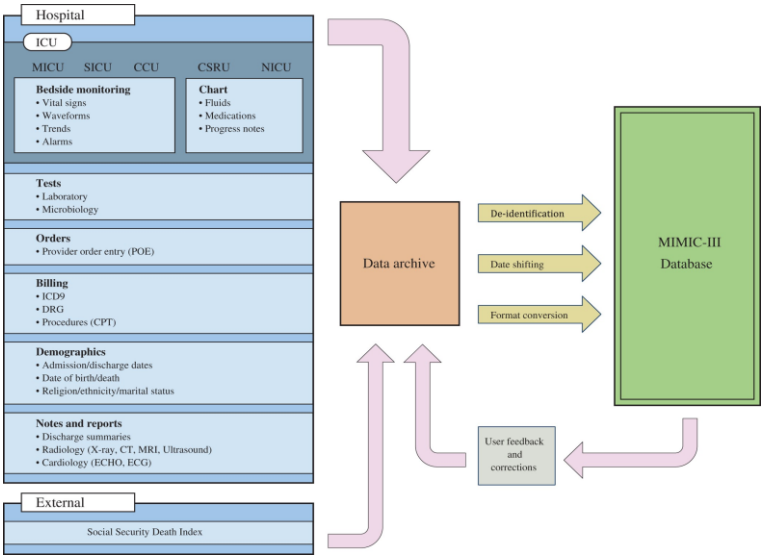

Figure 4.1: Overview of the MIMIC-III critical care database.

Furthermore, we used FitBitChargeHR dataset [47] from Kaggle [48] data science repository. The file contains one year of human activity such as calories, steps, distance in meters, floors, minutes sitting, minutes of moderate activity, minutes of intense activity as well as the calories burned for the activities. The data was gathered with a Fitbit Charge HT fitness tracker and every observation regards one day. We multiplied and used this data in conjunction with MIMIC-III data to implement and simulate real world stream processing scenarios.

4.2 Data Extraction for Class Analysis

The majority of the useful information in this dataset is not clear and obvious. Significant parts of the data are coded with specific numbers. As we mentioned above, ICD-9 diagnosis codes [8] and ItemID codes [8] that identify mechanically ventilated patients with their laboratory measurements and their charts values, contribute significantly on identification of severity class of ARDS patients. More specifically, PaO2, FiO2, PEEP and HR information were extracted from charted data using ICD-9 and ItemID codes. WBC (White Blood Cell), lactate and other useful ARDS related values were extracted from laboratory measured data. Time series include specific time points of ARDS onset which are defined based on Berlin criteria [5], i.e. PaO2/FiO2 ratio $\leq$ 300 with PEEP at least 5 cmH2O or SpO2/FiO2 $\leq$ 200 [9, 12, 13, 14]. The observed vital signs and laboratory measurements after the identified diagnosis time are extracted and features constructed as class-defining variables in our modelling including diastolic blood pressure, mean airway pressure (MAP), respiratory rate, systolic blood pressure, temperature, PH, platelet, blood cultures, tidal volume, GCS, height, weight, BMI, PaO2/FiO2 (P/F), SpO2/FiO2 (S/F) and some demographic variables. (see Appendix B, C, D)

4.3 Data Extraction for Predictive Modeling

The features that we considered to build the predictive model include:

1. Vital signs from chart measurements: heart rate, respiratory rate, body temperature, systolic blood pressure, diastolic blood pressure, mean arterial pressure, oxygen saturation, tidal volume.

2. Laboratory measurements: white blood cell count, hematocrit, lactate, creatinine, bicarbonate, pH, INR, blood gas measurements (partial pressure of arterial oxygen, fraction of inspired oxygen, and partial pressure of arterial carbon dioxide).

3. Other chart measurements: motor, verbal, and eye sub-score of Glasgow Coma Scale (GCS).

4. Demographic indicators as potential risk factors: gender, age, ethnicity, etc.

   (see Appendix B, C, D)

4.4 Data Preprocessing Methods

This section describes all the adopted data preprocessing, cleaning, filtering and normalization methods, in addition with solutions to the challenges arose.

4.4.1 Data Cleaning and Filtering

Primarily, we comprehended the nature of the data and their peculiarities, investigating in depth the large number of parameters and their varied content. As we explained above, there are specific codes that need to be decoded and clarified in order to access the data that really interest us. The primary purpose for starting the cleaning and preprocessing of the data was to separate the useful details we were looking for in this database. Regarding the feature selection section, in this study we extracted a variety of information and physiological parameters of the patients, consulted and guided mainly by the clinical doctor we collaborated with

and the relevant study [9]. Subsequently, in next steps we validated the feature selection and the dynamic of our data preprocessing methods using WEKA [18]. To get the information we needed from the data we used the query (subsystem) application that we built (described above in the architecture section) and gathered all the necessary details. According to the literature review [9, 10] we found the specific codes that correspond to ARDS related risk factors. Each query asks the database for the specific codes (ItemID, ICD9) we are looking for in the specific files and then returns the answer accompanied by all the necessary information that we want. In our case, this information is called SUBJECT_ID, HADM_ID and ICUSTAY_ID which were accompanied by associated values such as vital signs, laboratory measurements, demographics, etc. SUBJECT_ID refers to patients, HADM_ID refers to hospital admissions and ICUSTAY_ID refers to ICU stays. Once we collected this information, we then proceeded with the process of cleaning the data from missing and not useful values. In the process of collection of physiological parameters from patients, we have found that some parameters were missing, possibly because they were recorded at a lower frequency, however, this fact would result in an imbalance in the dataset. Fortunately, this issue had already been handled successfully from [9], therefore, in any case we had to balance the missing data, we used the technique of imputation and oversampling which they suggested and explained in detail, filtering at the same time the fields with information that interested us mostly. Then, after categorizing the data based on their content, we proceeded with the rest pre-processing procedure on them which revealed various problems that usually arise in the processing and management of time series data.

4.4.2 Data Normalization and Solutions to Challenges

We will focus on the main problem that preoccupied us and cost us much time and computing resources in data normalization procedure. Since we split and categorized our significant ARDS related values into pieces, per patient, hospitalization and ICU stay, facilitating the process of retrieving them through a query at any moment, it was time to put these pieces together in order to create a

single data set which we would then split into training and test set with all the necessary information and features that we would like to give to the machine learning algorithms of the predictive model. To achieve this, we had to find some common points in these pieces that would form the links, the so-called "keys" in the databases field. In time series data, of course, we rarely see unique keys when the dataset contains hospitalization IDs and patient IDs. This is because each patient is connected to a machine, whether it is in the intensive care unit, the hospital room, or a room in a home. This machine records information (SpO2, Heart rate, PaO2, FiO2, etc.) over time, so we had many different values for different timestamps for the same patient. Therefore, it was inevitable to join all the pieces we wanted in non-unique keys, taking into account the size of the tables with millions of rows. At this point we met the biggest challenge of our work, (see Figure 4.2), trying to solve this issue with the available resources.

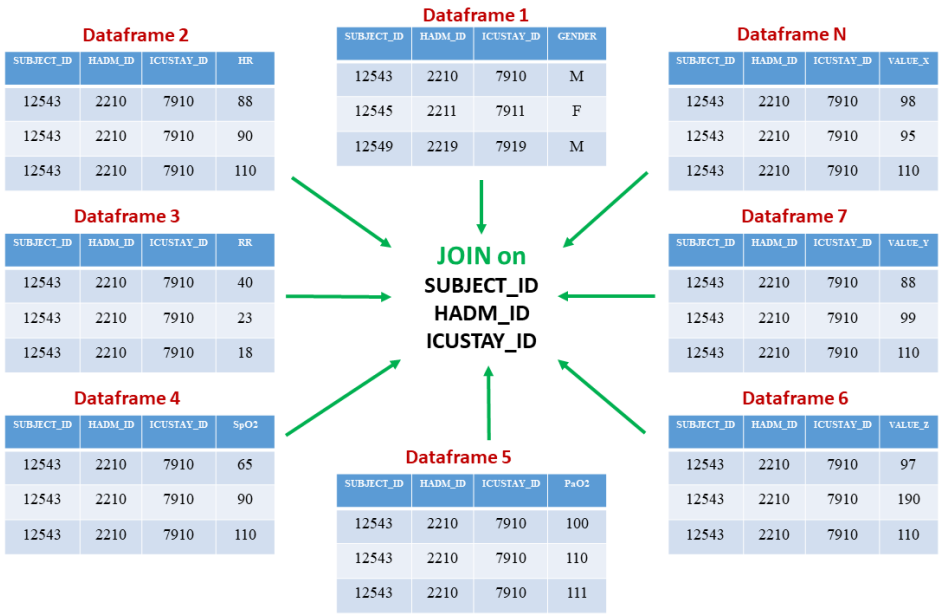

Figure 4.2: Join attempt on non-unique keys.

This cost us mainly in random access memory because as we mentioned above, Spark loads the data that is to be processed into memory. It was not a trivial situation and we still had to deal with it and continue with the preprocessing procedure of our data. There is no simple way to achieve this when it comes to large

volumes of data and in particular non-unique keys. It is worth noting that none of the related works mentioned this important fact, so there was no officially recorded solution. Even with the resources we had available in combination with the parallelism offered by spark, when we had to combine information from multiple tables with non-unique keys, the system was too late to answer our questions and in some cases was impossible to finish the join procedure. Nevertheless, we found a kind of solution in order to continue and finish our work. No matter how much memory we had at our disposal the problem would not be solved so easily. Therefore, we had to adapt to this situation and handle it. At this point we decided that it was necessary to consider one more parameter in the filtering and pre-processing methods of our data. We investigated our available data further to see if we can find any other common factor in these pieces besides the obvious ones that the MIMIC-III dataset officially states. After exhausting investigation inside the data, we found another element called “CHARTTIME” and we managed to handle the aforementioned problem by setting this new parameter as the only unique sub-key. 'CHARTTIME' records the time at which an observation was made, and is usually the closest proxy to the time the data was actually measured. It is essentially a unique timestamp for each patient's records. We set specific time limits in which we singled out and filtered the data that had a common timestamp, taking into account the year, month, day and time, rarely including the minutes and not at all the seconds. As a result, we re-categorized the patients who met the selection criteria we set and managed to join our data by creating the final data file that we would use in our model to split for train and test set in order to run the machine learning algorithms. This file contained all the necessary features and values for the build of our machine learning model. Below in Figure 4.3, we present a flowchart outlining a small part of our patient selection and filtering methods, detailing the number of patients that eventually get inside the predictive model.

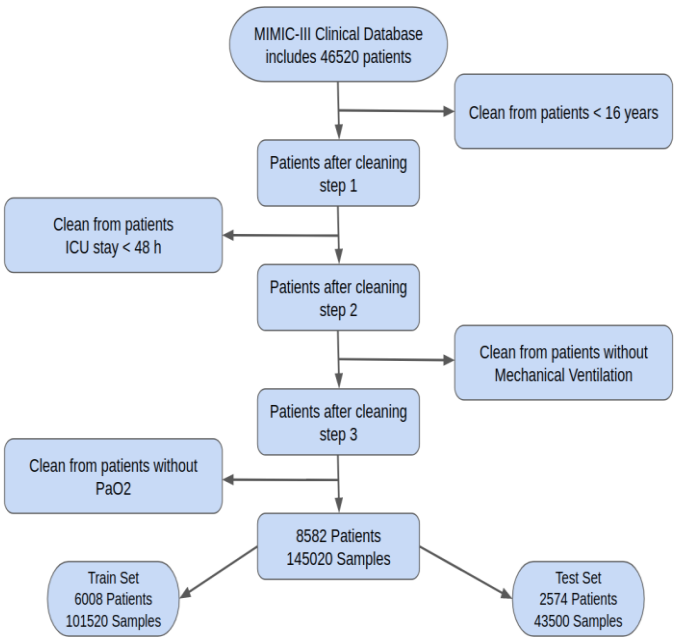

Figure 4.3: Flow diagram for patient selection.

4.5 Classification Algorithms

This section describes all the classification and prediction analysis algorithms in addition with parameters tuning methods used.

4.5.1 Prediction Class Analysis

Machine learning algorithms are generally divided into two categories: supervised and unsupervised learning algorithms. Supervised learning algorithms are used to uncover the relationship between variables of interest and one or more target outcomes. For supervised problems, the target outcome(s) must be known. Unsupervised learning algorithms are used to uncover naturally occurring patterns or groupings in the data, without targeting a specific outcome. In our case, we have to handle a supervised problem where we target two categorical outcomes. 1 for positive and 0 for negative in each case of ARDS severity. More specifically, we used this method for 4 different cases of severity in order to rebound at 4 classes which we set as output in our prediction model. Each case-class as we can see in

Table 4.1 represents the severity of ARDS in patients. The first class means "mild", the second class means "moderate", the third class means "severe" and the fourth class means "non ARDS", i.e. no risk. The most common applications of data science to critical care problems are predictive models using supervised learning algorithms. In this study, as we can see in Figure 4.4, we designed a complete pipeline model that combined feature selection (from a given number of total features) with multiple classification algorithms, used a 10-fold cross-validation model, trained classifiers for different feature subsets, and selected the optimal combination of feature subsets and classifiers, accomplishing the early identification of the ARDS. We used random forest and logistic regression machine learning algorithms in order to build and validate our predictive model with the available data of MIMIC-III, splitting them in train (70%) and test (30%) sets.

ARDS Severity	Class
Mild	1
Moderate	2
Severe	3
None	4

Table 4.1: ARDS severity classes.

4.5.2 Random Forest

Random forests are ensembles of decision trees. They combine many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions. Spark.mllib [54] supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. It implements random forests using the existing decision tree implementation. Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data. To make a prediction on a new instance, a random forest must aggregate the predictions from its set of

decision trees. As regards the classification method that we used, the majority vote wins. Each tree's prediction is counted as a vote for one class. The label is predicted to be the class which receives the most votes. The most important parameters that we used and tuned in order to improve the performance of our model are the number of trees and the maximum depth of each tree in the forest. Increasing the number of trees will decrease the variance in predictions, improving the model's test-time accuracy and training time increases roughly linearly in the number of trees. Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train, it is acceptable to train deeper trees when using random forests than when using a single decision tree, because one tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest). We experimented with many different cases of the same algorithm, changing some important parameters each time, in order to conclude at the optimal one that would give us the best results. More specifically, we experimented setting different values at the max depth of the tree, the number of the trees and the number of folds at cross validation step.

4.5.3 Logistic Regression

Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. It can be used to solve under classification type machine learning problems. Classification involves looking at data and assigning a class (or a label) to it. Usually there are more than one classes, when there are two classes (0 or 1) it identifies as Binary Classification. In spark.ml [54] logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression. Our Logistic Regression model builds a Binary Classifier model to predict ARDS based on historical data of patients. Specifically, we built a Logistic Regression model by experimenting and finally defining the number of max iterations, the number of regression parameters and the number of elastic Net parameters.

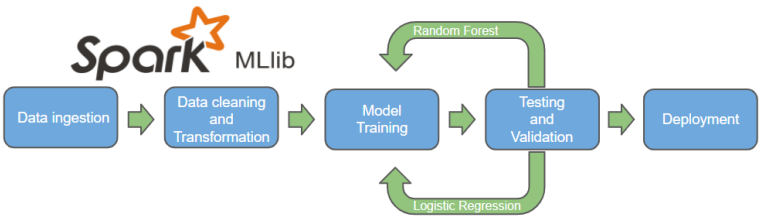

Figure 4.4: Machine learning pipeline model.

4.6 Evaluation and Results

According to the diagnostic definition of ARDS disease, P/F ≤ 300 means ARDS. According to this standard, each sample is divided into positive and negative results. Table 4.2 describes the relationship between the actual category and the prediction category. A common metric used to evaluate the accuracy of a Logistic Regression and Random Forest model with binary classification is Area Under the Curve (AUC). We measured the classification performance based on the average of AUC, the accuracy, sensitivity and specificity.

Predicted Class	**Actual Class**	
	Positive (P/F≤300)	**Negative (P/F>300)**
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Table 4.2: The relationship between actual categories and prediction results.

We identified 8582 patients who met our inclusion criteria from a total of 46520 patients enrolled in the MIMIC-III database. As we can observe in Figure 4.3, there were 6008 patients (101520 data points) in the training set and 2574 patients (43500 data points) in the test set. The patient demographics and characteristics that utilized in this study are shown in Table 4.3 and in Appendix B, C, D in Tables A.2, A.3, A.4 and A.5. Table 4.3 summarizes the demographic information of patients that we used. The training set has a consistent patient distribution with the test set. In the training set, the patients were hospitalized in various intensive care

units: CSRU (Cardiac Surgery Recovery Unit), MICU (Medical Intensive Care Unit), SICU (Surgical Intensive Care Unit), TSICU (Trauma Surgical Intensive Care Unit), and CCU (Coronary Care Unit) and the average age of patients was 65 years. The majority of the patients were male. Appendix C in Table A.3 summarizes the physiological parameters of patients classified in the training and test sets. For the training set and the test set, the two datasets were randomly grouped and had a common distribution. According to [9] and WEKA [18] feature selection methods and rankings, SpO2 was clearly the most relevant parameter. Furthermore, SpO2, S/F, FiO2, and PEEP, were also highly relevant features. Using the training dataset, the 10-fold cross validation methods were used to evaluate the performance of the random forest algorithm. Moreover, we used the same data in order to evaluate the logistic regression algorithm and the test sets were completely independent of the data of model training.

Demographic Variables
Age(year)
Gender (Male – Female)
BMI(kg/m2)
Length of stay in ICU (days)
ICU type (CSRU, MICU, SICU, TSICU, CCU)
Admission type (Emergency, Elective)
Ethnicity (White, Asian, Black, Hispanic, Other)

Table 4.3: Patient demographics in training and test sets.

The common features in relation to [10] were many, so we considered that we could make a comparative study of the two most relevant works [9, 10] in relation to our own. In Table 4.4 we present the similar features of [9, 10] with our work and in Table 4.5 we present the most in common features, that we used in order to perform our machine learning experiments, of our work with [9] which is the closest use case to ours. As we can observe, most of the features are common, therefore we can compare our performance results with both of these works [9, 10].

Mutual features between two related works [10], [9] and our work		
[10]	**[9]**	**Our work**
Age	Age	Age
PEEP	PEEP	PEEP
FiO2	FiO2	FiO2
Creatinine	GCS (eye)	GCS (eye)
Blood Culture	GCS (motor)	GCS (motor)
Diastolic BP	TV / Kg	TV / Kg
Fluid Bolus	GCS (Verbal)	GCS (Verbal)
GCS	GCS	GCS
Heart Rate	Heart Rate	Heart Rate
INR	Gender (Female)	Gender (Female)
Lactate	Gender (Male)	Gender (Male)
Mean Air Pressure	Mean Air Pressure	Mean Air Pressure
Organ Dysfunction	Peak Pressure	Peak Pressure
PP	Plateau Pressure	Plateau Pressure
Platelets	OSI	OSI
Respiratory Rate	Respiratory Rate	Respiratory Rate
SpO2	SpO2	SpO2
Temperature	Temperature	Temperature
Urine Output	Nidbp	Nidbp
WBC	Minute Ventilation Volume	Minute Ventilation Volume
pH	Nisbp	Nisbp
Systolic Blood Pressure	Nimbp	Nimbp
Antibiotics	BMI	BMI
PaO2	PaO2	PaO2
Bilirubin	S/F	S/F
PaO2/FiO2 (P/F)	P/F	P/F

Table 4.4: Mutual features among [10], [9] and our work.

Most in common features between [9] and our work
SpO2
PaO2
FiO2
Heart Rate
Temperature
Tidal Volume
PEEP
Nbps
GCS
GCS Verbal
P/F
First careunit
Last careunit
Ethnicity
Admission type
Height
Weight
Dbsource

Table 4.5: Most in common features between [9] and our work.

We observe the AUC results of the two algorithms in Figure 4.5 and their comparison with the dominant algorithm (XGBoost) of [9] is shown in Figure 4.6. The AUC of the Random Forest (0.95) is higher than the AUC of Logistic Regression (0.93) under the same feature set and both of them are higher than AUC of [9] XGBoost (0.91). Moreover, in Table 4.6 we may observe the AUC, Accuracy, Specificity and Sensitivity results of RF and LR algorithms that arise from the calculations of confusion matrix (RF) and confusion matrix (LR) that we can see in Table 4.7 and Table 4.8 respectively. Based on the results, we show that the overall performance of our two algorithms exhibits significantly better performance with respect to the closest related work [9] algorithm.

Algorithm	**Results**			
	AUC	**Accuracy**	**Specificity**	**Sensitivity**
Random Forest	95,1 %	98,0 %	98,62 %	96,25 %
Logistic Regression	93,31 %	95,0 %	98,62 %	90,63 %

Table 4.6: Identification results of two algorithms on test sets.

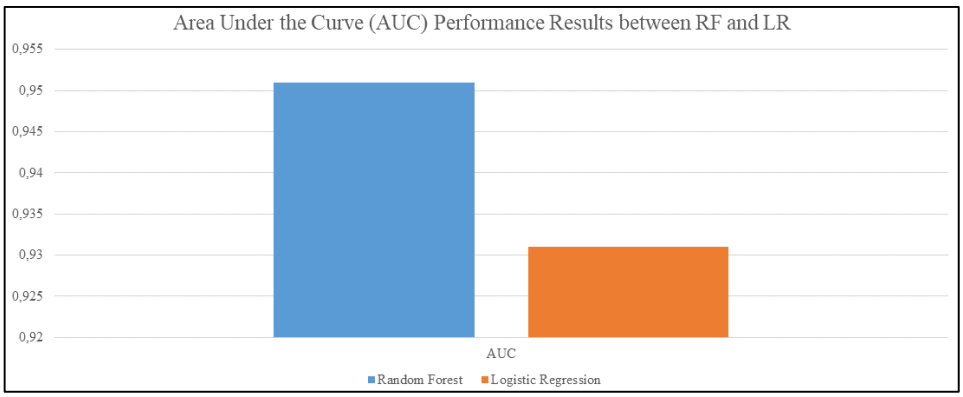

Figure 4.5: AUC Performance results between RF and LR.

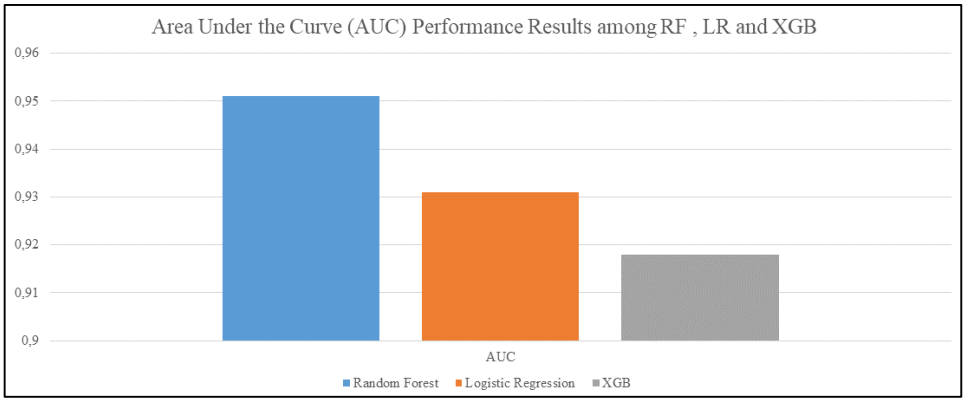

Figure 4.6: AUC Performance results among RF, LR and XGB [9].

Confusion Matrix (RF)		**Actual Class**			
		Class 1	**Class 2**	**Class 3**	**Class 4**
Predicted Class	**Class 1**	20723	392	0	101
	Class 2	0	19776	592	0
	Class 3	0	661	12893	0
	Class 4	598	0	0	3571

Table 4.7: Confusion Matrix (Random Forest).

Confusion Matrix (LR)		**Actual Class**			
		Class 1	**Class 2**	**Class 3**	**Class 4**
Predicted Class	**Class 1**	19398	1292	0	425
	Class 2	540	19067	169	0
	Class 3	0	1116	12438	0
	Class 4	1687	0	0	2482

Table 4.8: Confusion Matrix (Logistic Regression).

In this case study, we analyzed the classification ability of two different algorithms, as we can see in Table 4.9, Table 4.10 and Figure 4.7, Figure 4.8 respectively. In the algorithmic evaluation, we compare the experimental results of both algorithms which reflect the great ability of the Random Forest algorithm to mine information from different aspects. Therefore, according to the results, Random Forest with 10-fold cross validation and specific parameters tuned, achieves the best results under the selected dataset.

Classification Results (RF)	**Random Forest**			
	Class 1	**Class 2**	**Class 3**	**Class 4**
Accuracy	98,12 %	97,19 %	97,85 %	98,79 %
Specificity	98,66 %	98,43 %	98,52 %	98,89 %
Sensitivity	97,20 %	94,94 %	95,61 %	97,25 %

Table 4.9: Classification results per class (RF)

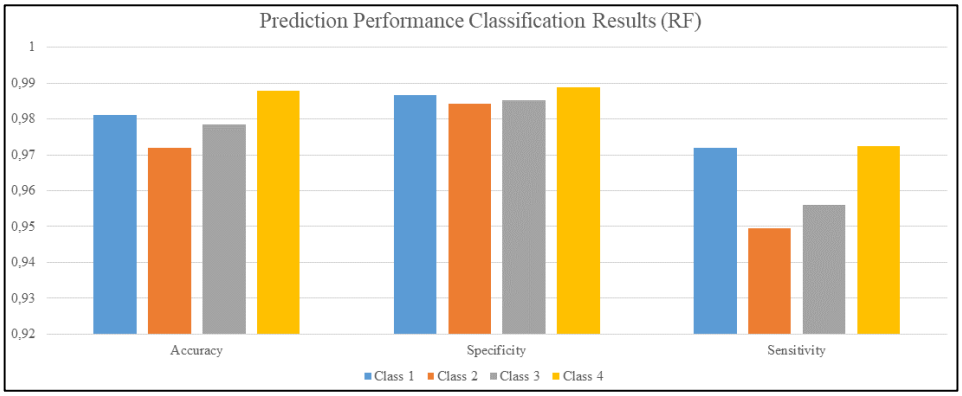

Figure 4.7: Prediction performance classification results (RF).

Classification Results (LR)	**Logistic Regression**			
	Class 1	**Class 2**	**Class 3**	**Class 4**
Accuracy	93,12 %	94,48 %	97,65 %	96,19 %
Specificity	95,19 %	97,98 %	97,35 %	96,79 %
Sensitivity	89,70 %	88,79 %	98,66 %	85,38 %

Table 4.10: Classification results per class (LR)

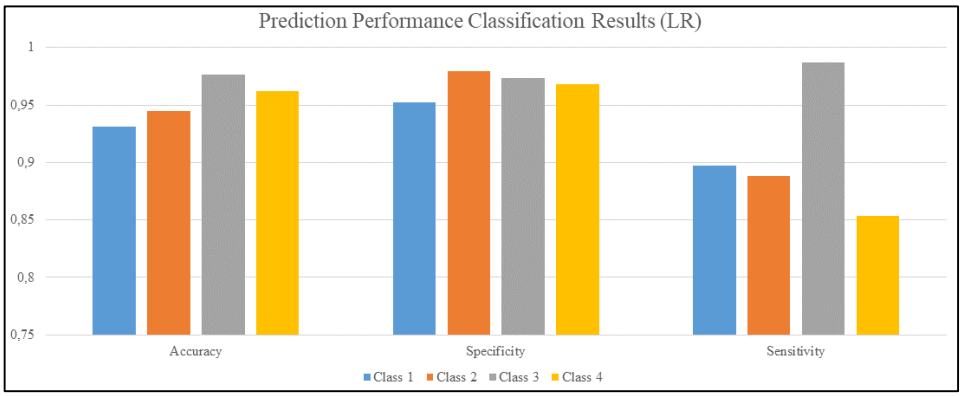

Figure 4.8: Prediction performance classification results (LR).

4.7 Discussion

Our approach achieves significantly better results in all metrics that are based on AUC, when compared to relevant published efforts that also use the MIMIC III database to develop predictive models of ARDS. Our prediction performance results between our two deployed algorithms (RF and LR) and XGBoost (XGB) [9] are depicted in Figure 4.9 and as we can see in Table 4.11 and Figure 4.10 we outperform among [9] and [10] which also uses the XGBoost (XGB) algorithm. Our distinction in the results originates from the preprocessing in data management procedure, the feature selection and the different algorithms we used. In addition, we tuned specific parameters in both of our algorithms and we concluded that the random forest prediction model performs significantly better than logistic regression and the related works [9, 10], because of the fine-tuned number and depth of trees in combination with 10-folds cross validation.

Sampling , Splitting and Algorithms used on MIMIC-III data with Results			
	[10]	**[9]**	**Our work**
Number of samples	9001	8702	8582
Split of Data	Train: 90 % Test: 10 %	Train: 75 % Test: 25 %	Train: 70 % Test: 30 %
Algorithms Used	XGBoost	XGBoost	Random Forest
AUC	90,5 %	91,28 %	95,1 %
Accuracy	82,5 %	85,89 %	98,0 %
Specificity	82,3 %	87,75 %	98,62 %
Sensitivity	80,06 %	84,03 %	96,25 %

Table 4.11: Sampling, splitting and algorithms used on MIMIC-III with performance results of best cases.

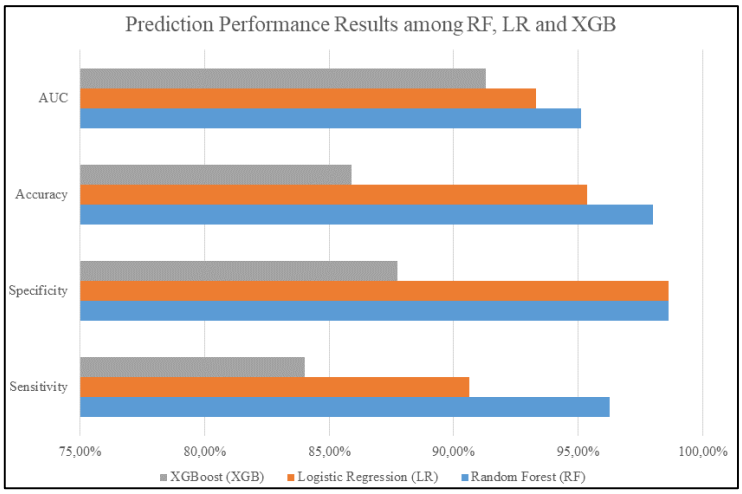

Figure 4.9: Prediction performance results among RF, LR and XGB [9].

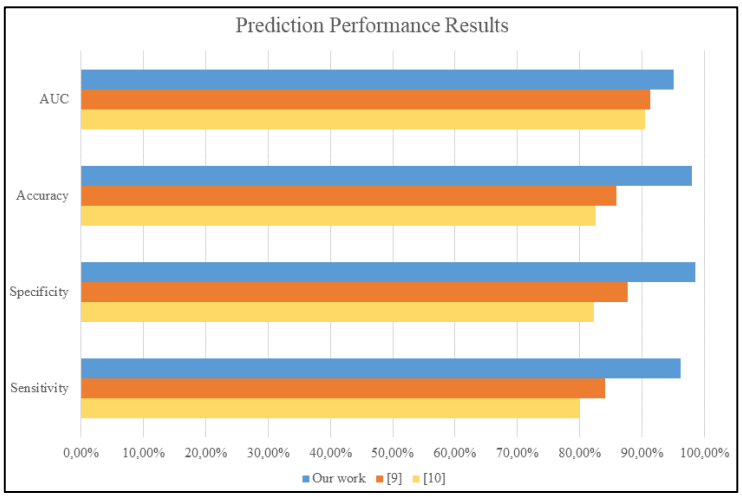

Figure 4.10: Prediction Performance Results between [9], [10] and our work.

Chapter 5

Conclusion and Future Work

In this thesis, we present a scalable data science platform built on open source technologies, accompanied by a biomedical data analysis application about Acute Respiratory Distress Syndrome disease (ARDS). We underlined the main challenges and complications about infrastructures and data analysis in our use case, managing to handle and explain in detail any issues that arose. In the first part of this thesis, we presented our platform's architecture and the main clinical scenarios that may serve. These scenarios depict a batch and stream processing data flow respectively. We analyzed the most significant parts of this platform with all the adopted methods, giving insights on its scalability and utilization techniques.

As a result of conducting this study for this thesis, we conclude in the first part, that scalable systems and infrastructures, built on open source technologies, have the potential to manage big data processing scenarios, however they also have limitations. We refer mainly to the technical problems that arose in the construction of the platform and then in development and evaluation of the application of predictive models. Time series big data need special handling in the preprocessing procedure, especially in cases where the data is biomedical and we aim to predict a clinical situation accurately. Consequently, we propose the adoption of open source technologies for the construction of scalable infrastructures and systems, provided that they have rich supporting community and documentation for time and financial benefit.

At the second part of this thesis, we explained how we used our scalable platform to build prediction models of ARDS, giving insights about the data sources utilized and mainly data selection of MIMIC-III clinical database. We focused on the

preprocessing methods, highlighting technical data analysis issues that arise from such a large dataset, with such a heterogeneous clinical situation that needs to be managed. Our approach handled the complications that arose with specific methods, that the related works mentioned above, do not report in their study cases. Moreover, we experimentally evaluated our data analysis application of ARDS, using machine learning algorithms, comparing our results with the closest related work [9, 10] use cases. The overall classification effects of our Random Forest model was better than our Logistic Regression model and outperforms the related works [9, 10] XGBoost model. Concluding, our approach, specifically of random forest with fine tuning of parameters, achieves significantly better results, when compared to relevant published efforts that also use the MIMIC III database to develop predictive models of ARDS. Of course, it is worth noting that specific and valuable risk factors of ARDS, in addition with feature selection information that we consulted from [9] and WEKA [18] results, highly contributed in this comparison study.

As future work, we plan to evaluate our platform on new use cases using MIMIC-IV [50] clinical database that was released a few time ago. Furthermore, we intend to extend the existing functions of our platform in order to be able to manage multimodal data sets, including signals, images (MRI, CT, etc.), videos and genomics data. Eventually, we do not neglect the fact that our platform requires substantial technical expertise to use it clinical staff to its full potential. Therefore, we aim to create a graphical user interface (GUI) that will enable data visualization with charts, in addition with data management features. This action will eliminate the above functional limitation of the platform and upgrade the value of the query application, making it easier to be utilized by doctors and clinical staff.

Bibliography

[1] Dash, Sabyasachi, et al. "Big data in healthcare: management, analysis and future prospects." Journal of Big Data 6.1 (2019): 1-25.

[2] Wang, Tony, et al. "Using latent class analysis to identify ARDS sub-phenotypes for enhanced machine learning predictive performance." *arXiv preprint arXiv:1903.12127* (2019).

[3] Gibson, Peter G., Ling Qin, and Ser Hon Puah. "COVID-19 acute respiratory distress syndrome (ARDS): clinical features and differences from typical pre-COVID-19 ARDS." *Med J Aust* 213.2 (2020): 54-56.

[4] Rezoagli, Emanuele, Roberto Fumagalli, and Giacomo Bellani. "Definition and epidemiology of acute respiratory distress syndrome." *Annals of translational medicine* 5.14 (2017).

[5] Force, ARDS Definition Task, et al. "Acute respiratory distress syndrome." *Jama* 307.23 (2012): 2526-2533.

[6] Fan, Eddy, et al. "COVID-19-associated acute respiratory distress syndrome: is a different approach to management warranted?." *The Lancet Respiratory Medicine* (2020).

[7] Ferguson, Niall D., Tài Pham, and Michelle Ng Gong. "How severe COVID-19 infection is changing ARDS management." (2020): 1-3.

[8] Johnson, Alistair EW, et al. "MIMIC-III, a freely accessible critical care database." *Scientific data* 3.1 (2016): 1-9.

[9] Yang, Pengcheng, et al. "A new method for identifying the acute respiratory distress syndrome disease based on noninvasive physiological parameters." *PloS one* 15.2 (2020): e0226962.

[10] Le, Sidney, et al. "Supervised machine learning for the early prediction of acute respiratory distress syndrome (ARDS)." *Journal of Critical Care* 60 (2020): 96-102.

[11] McPadden, Jacob, et al. "A scalable data science platform for healthcare and precision medicine research." arXiv preprintarXiv:1808.04849 (2018).

[12] Pisani, Luigi, et al. "Risk stratification using SpO 2/FiO 2 and PEEP at initial ARDS diagnosis and after 24 h in patients with moderate or severe ARDS." *Annals of intensive care* 7.1 (2017): 1-10.

[13] Brown, Samuel M., et al. "Nonlinear imputation of Pao2/Fio2 from Spo2/Fio2 among patients with acute respiratory distress syndrome." *Chest* 150.2 (2016): 307-313.

[14] Rice, Todd W., et al. "Comparison of the SpO2/FIO2 ratio and the PaO2/FIO2 ratio in patients with acute lung injury or ARDS." *Chest* 132.2 (2007): 410-417.

[15] Zhang, Zhongheng, and Hongying Ni. "Prediction model for critically ill patients with acute respiratory distress syndrome." *PloS one* 10.3 (2015): e0120641.

[16] Neto, Ary Serpa, et al. "Mechanical power of ventilation is associated with mortality in critically ill patients: an analysis of patients in two observational cohorts." *Intensive care medicine* 44.11 (2018): 1914-1922.

[17] Taoum, Aline, Farah Mourad-Chehade, and Hassan Amoud. "Early-warning of ARDS using novelty detection and data fusion." *Computers in biology and medicine* 102 (2018): 191-199.

[18] https://www.cs.waikato.ac.nz/ml/weka/

[19] Kaur, Jagreet, and Kulwinder Singh Mann. "AI based healthcare platform for real time, predictive and prescriptive analytics using reactive programming." *Journal of Physics: Conference Series*. Vol. 933. No. 1. IOP

Publishing, 2017.

[20] Sharma, Ankita, et al. "BHARAT: an integrated big data analytic model for early diagnostic biomarker of Alzheimer's disease." *Frontiers in neurology* 10 (2019): 9.

[21] Ta, Van-Dai, Chuan-Ming Liu, and Goodwill Wandile Nkabinde. "Big data stream computing in healthcare real-time analytics." *2016 IEEE international conference on cloud computing and big data analysis (ICCCBDA)*. IEEE, 2016.

[22] Raghupathi, Wullianallur, and Viju Raghupathi. "Big data analytics in healthcare: promise and potential." *Health information science and systems* 2.1 (2014): 1-10.

[23] Benhlima, Laila. "Big data management for healthcare systems: architecture, requirements, and implementation." *Advances in bioinformatics* 2018 (2018).

[24] Kiran, Mariam, et al. "Lambda architecture for cost-effective batch and speed big data processing." *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015.

[25] https://aws.amazon.com/ec2/

[26] https://hadoop.apache.org/

[27] Sanchez, Elizabeth, et al. "Persistent severe acute respiratory distress syndrome for the prognostic enrichment of trials." *PloS one* 15.1 (2020): e0227346.

[28] Xie, Jianfeng, et al. "A modified acute respiratory distress syndrome prediction score: a multicenter cohort study in China." *Journal of thoracic disease* 10.10 (2018): 5764.

[29] Yu, Xue-Shu, et al. "Lung-heart pressure index is a risk factor for acute respiratory distress syndrome (ARDS): A machine learning and propensity

score-matching study." (2019).

[30] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters." (2004).

[31] Zaharia, Matei, et al. "Spark: Cluster computing with working sets." *HotCloud* 10.10-10 (2010): 95.

[32] https://spark.apache.org/

[33] Borthakur, Dhruba. "The hadoop distributed file system: Architecture and design." *Hadoop Project Website* 11.2007 (2007): 21.

[34] Shvachko, Konstantin, et al. "The hadoop distributed file system." *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. Ieee, 2010.

[35] Rallapalli, Sreekanth, and R. R. Gondkar. "Apache Spark and Hadoop Based Big Data Processing System for Clinical Research." *International Journal of Applied Engineering Research* 13.10 (2018): 7488-7492.

[36] Kouanou, Aurelle Tchagna, et al. "An optimal big data workflow for biomedical image analysis." *Informatics in Medicine Unlocked* 11 (2018): 68-74.

[37] Hindman, Benjamin, et al. "Mesos: A platform for fine-grained resource sharing in the data center." *NSDI*. Vol. 11. No. 2011. 2011.

[38] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

[39] https://kafka.apache.org/

[40] Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." *Proceedings of the NetDB*. Vol. 11. 2011.

[41] https://www.linkedin.com/

[42] https://druid.apache.org/

[43] Yang, Fangjin, et al. "Druid: A real-time analytical data store." *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014.

[44] https://flink.apache.org/

[45] Carbone, Paris, et al. "Apache flink: Stream and batch processing in a single engine." *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36.4 (2015).

[46] Wang, Shirly, et al. "Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii." *Proceedings of the ACM Conference on Health, Inference, and Learning*. 2020.

[47] https://www.kaggle.com/alketcecaj/one-year-of-fitbit-chargehr-data

[48] https://www.kaggle.com/

[49] https://physionet.org/content/mimiciii/1.4/

[50] https://physionet.org/content/mimiciv/0.4/

[51] Apostolova, Emilia, et al. "Towards reliable ARDS clinical decision support: ARDS patient analytics with free-text and structured EMR data." AMIA Annual Symposium Proceedings. Vol. 2019. American Medical Informatics Association, 2019.

[52] Zhang, Zhongheng. "Identification of three classes of acute respiratory distress syndrome using latent class analysis." PeerJ 6 (2018): e4592.

[53] https://covid19.who.int/

[54] https://spark.apache.org/mllib/

[55] Stephens, Zachary D., et al. "Big data: astronomical or genomical?." PLoS

biology 13.7 (2015): e1002195.

[56] Ding, Xian-Fei, et al. "Predictive model for acute respiratory distress syndrome events in ICU patients in China using machine learning algorithms: a secondary analysis of a cohort study." Journal of translational medicine 17.1 (2019): 1-10.

Appendix

A. Review of solutions using MIMIC-III

Year	Publ.	Risk factors	Features	Methods
2020	[9]	(PaO2/FiO2 ratio <= 300), End-Expiratory Pressure (PEEP) >=5 cmH2O, mild (200 < arterial oxygen partial pressure (PaO2)/ fraction of inspired oxygen (FiO2) (P/F) <=300), moderate (100 < P/F <= 200), and severe (P/F <= 100), according to the level of oxygenation index (P/F)	Age, PEEP, FiO2, GCS (eye), GCS (motor), GCS (Verbal), TV / Kg, GCS, Heart Rate, Gender, Peak Pressure, Plateau Pressure, Respiratory Rate, OSI, SpO2, Temperature, Nidbp, Minute Ventilation Volume, BMI, Pa02, S/F, P/F	XGBoost with cross validation
2020	[10]	Positive end Expiratory Pressure (PEEP) >= 5 cmH 2O, PaO 2/FiO 2 ratio (P/F ratio) <= 300 mmHg	Age, PEEP, FiO2, Creatinine, Blood Culture, Diastolic BP, Systolic Blood Pressure, Fluid Bolus, GCS, Heart Rate, INR, Lactate, MAP, Organ Dysfunction, PP, Platelets, Respiratory Rate, SpO2, Temperature, Urine Output, WBC, pH, Antibiotics, PaO2, Bilirubin, PaO2/FiO2 (P/F)	XGBoost gradient boosted tree models with 10-fold cross validation
2019	[2]	PaO2/FiO2 ratio ≤ 300, PEEP >= 5 cmH2O	BMI, means of bicarbonate, plateau pressure, mean airway pressure (MAP), PaCO2, tidal volume, platelet count, total bilirubin; minimum of sodium, glucose, albumin, hematocrit, systolic blood pressure (SBP); maximum of temperature, heart rate, white blood cell (WBC) count, creatinine	Gradient Boosted Machine (GBM) , Random Forest (RF) with 5-fold cross validation
2019	[29]	PaO2/FiO2 ratio ≤ 300 (Berlin Definition [5])	Age, sex, DP, MAP, Pao2/Fio2, SOFA, RR, BMI, RDW, Ph and, ethnicity, BMI, smoking, SOFA, heart rate, laboratory values (pH,	Logistic regression, Random forest with 10-fold cross

			lactate, RDW), and ventilator parameters (LHPI, driving pressure, mechanical power, platform pressure)	validation
2019	[51]	Berlin Definition risk factors	anion gap (aniongap), albumin, bands, bicarbonate, bilirubin, creatine, chloride, glucose, hematocrit, hemoglobin, lactate, platelet, potassium, partial thromboplastin time (ptt), international normalized ratio (inr), prothrombin time (pt), sodium, bun, white blood cell count (wbc), heart rate (heartrate), systolic blood pressure, diastolic blood pressure, mean blood pressure, respiratory rate, body temperature, peripheral capillary oxygen saturation (spo2), body mass index (bmi), gender, age, urine output	Gradient Boosting Machine (GBM) model

Table A.1: Review of solutions using MIMIC-III.

B. Static variables and description

Static Variable	**Description**
Age	Patient Age
Gender	Patient Gender
Ethnicity	Patient Ethnicity
Insurance	Patient Insurance Type
Admittime	Hospital Admission Time
Dischtime	Discharge Time
Intime	ICU admission time
Outtime	ICU discharge time
Admission_type	Type of hospital admission
First_careunit	First ICU the patient was cared for

Table A.2: Static variables and description names.

C. Features used in predictive modeling and query subsystem

Category	Name
Vital	Heart Rate
	Systolic Blood Pressure
	Diastolic Blood Pressure
	Mean Airway Pressure
	Respiratory Rate
	Temperature
Blood Gas	SpO2
	PCO2
	PO2
	FIO2
	Bicarbonate
	Tidal volume
	Oxygen saturation
Hematology	PTT
	INR
	Platelet
	Hematocrit
	PT
	WBC
GCS	Total
	GCS Motor
	GCS Verbal
	GCS Eye
Blood Chemistry	Anion Gap
	Albumin
	Bilirubin
	Creatinine
	Glucose
	Lactate
	pH
Demographics	Age
	Gender
	BMI

Table A.3: Features used in predictive modeling and query application.

D. Identification codes for ARDS related risk factors and features from MIMIC-III

ICD-9 Code	Name
5279, 51881	Acute respiratory failure
5274, 51851	Acute respiratory failure following trauma and surgery
8702, 769	Respiratory distress syndrome in newborn
5133, 5063	Other acute and subacute respiratory conditions

Table A.4: ICD9-Codes with names.

ItemID Code	Name
490, 779	PaO2
646, 220277	SpO2
190, 223835, 3422	FiO2
220045	Heart Rate
8368, 8440, 8441, 8555, 220180, 220051	Diastolic BP
50827, 50828, 51237	INR
50813	Lactate
456, 52, 6702, 443, 220052, 220181, 225312	MAP
615, 618, 220210, 224690	Respiratory Rate
51, 442, 455, 6701, 220179, 220050	Systolic BP
223762, 676, 50825	Temperature C
50820	PH
51300, 51301	WBC
198	GCS
50800, 50801, 50802, 50803, 50804, 50805, 50806, 50807, 50808, 50809, 50810, 50811, 50812, 50813, 50814, 50815, 50816, 50817, 50818, 50819, 50820, 50821	Blood Cultures
223761, 678	Temperature F
828	Platelets
220179	Non Invasive Blood Pressure systolic
220180	Non Invasive Blood Pressure diastolic
220181	Non Invasive Blood Pressure mean
220051	Arterial Blood Pressure diastolic
220052	Arterial Blood Pressure mean

50826	Tidal Volume
507	PIP
543	Plateau Pressure
3259, 6078	MV
50819, 505	PEEP
198	GCS Total
227012, 226757, 454, 223901	GCS Motor
227014, 226758, 723, 223900	GCS Verbal
227011, 226756, 184, 220739	GCS Eyes
226730, 920, 1394, 4187, 3486, 3485, 4188, 226707	Height
3580, 3693, 226512, 220739	Weight

Table A.5: ItemID Codes with names of patient physiological parameters and characteristics.