An End-To-End Platform for Creating an Ensemble Asynchrony Detection Algorithm for Mechanically Ventilated Patients

By

Gregory Rehm

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

In the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA DAVIS

Approved:

_____
Chen-Nee Chuah, Chair

_____
Nicholas Anderson

_____
Jason Adams

Committee in Charge

2017

ABSTRACT

Advances in medical data science have potential to allow existing systems to become more fault tolerant, adaptive to adverse conditions, and save untold numbers of lives. Mechanical ventilators in particular show incredible promise for improvement by utilizing the latest techniques in data science. A key problem in mechanical ventilation is the necessity for providers to continuously adjust the ventilator to patient respiratory demand. If adjustments are not made during times of patient discomfort or distress, pathologic interactions between the ventilator and patient called patient ventilator asynchronies (PVA) can occur. PVA, if left unchecked, can have a variety of adverse outcomes, including patient discomfort, lung injury, and potentially even death. Because of this, it is critical we detect PVA as soon as and as accurately possible. To perform PVA detection, we designed a nearly completely automated platform of hardware and software that is capable of capturing mechanical ventilator waveform data, processing it, and then performing detection for harmful asynchronies in a clinical environment. Capture of mechanical ventilator data relies on Raspberry Pi microcomputers that stream their data to a central server. Clinicians can then annotate data to generate machine learning based PVA detection algorithms that can determine if and what type of asynchrony occurs at any specific point in time. These algorithms and the process used to create them can serve a multitude of purposes. They may serve as an example to researchers on how to create clinical decision support platforms for high-frequency waveform data. And in the future, these algorithms may assist in greater automation of ventilator management, and the creation of robust clinical support decision systems, thus improving standards of care in the hospital.

ACKNOWLEDGMENTS

My thanks to my graduate committee for their guidance. Thank you Jason Adams for the time you've taken out of your busy day to answer my questions, and the guidance you've given me towards becoming a better scientist. Thank you Chen-Nee Chuah for handling all the administrative hassles that you do and always being available when I need feedback. Thank you Nick Anderson for helping our lab so immensely and for always being available to talk. Thank you to my family who has supported me through this process, and most of all thank you Jackie for always being there for me. I couldn't have done this without you.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I: Introduction

1.1 Rationale

The intensive care unit (ICU) is a data-rich, "information poor" environment with at present limited capability to provide decision support of high volume patient data for clinical or research use. Physiologic monitoring devices often use waveform data that are used by bedside care teams for diagnosis, prognostication, and assessing response to treatment, but are not routinely recorded in the electronic health record (EHR).[1] The rapid development of sophisticated instrumentation and supporting software by vendors could ameliorate this problem, but factors ranging from lack of interface standards to proprietary, cost-prohibitive data collection and storage mechanisms make platforms that utilize these rich data streams difficult to establish, deploy, and maintain, especially for research purposes. [2]

These limitations are well-illustrated through the use case of mechanical ventilation. Modern mechanical ventilators are sophisticated life support devices that regulate the delivery of pressure, flow, and supplemental oxygen for the management of patients with acute respiratory failure. The goal of mechanical ventilation is to improve patient oxygenation and relieve stress on the patient's respiratory system.[3,4] To better assist care providers, ventilators routinely display streaming waveform data representing pressure, flow, and volume (Figure 1) that are used by bedside clinicians to diagnose and manage patients in real time. However, for a variety of reasons, these data are not routinely captured to facilitate research.[2,5,6]

Lack of access to these data has limited the study of mechanical ventilation and pathologic patient-ventilator interactions such patient-ventilator asynchrony (PVA). PVA occurs when ventilator support is inadequate to meet patient demand, in terms of timing and/or quantity of support.[7] In small studies, PVA has been associated with a variety of adverse clinical

outcomes including prolonged duration of mechanical ventilation, longer ICU length of stay, higher rates of tracheostomy, patient discomfort, and increased sedative dosing.[7-9]It is also suggested from that severe PVA can lead to more ventilator induced pneumonia and increased mortality.[10-14] In another study, patients with severe ARDS had lower mortality when receiving a PVA preventative paralytic drug, compared to a population that did not receive the paralytic.[11] It has even been suggested that PVA could be an contributing cause to ICU related PTSD.[15] Although evidence suggests that PVA can be harmful to patient health, researchers have been limited in their ability to solidify a connection between PVA and other more life threatening complications. This is due in part to the fact researchers lack an easily accessible, and clinically validated platform to detect PVAs.[16-18]

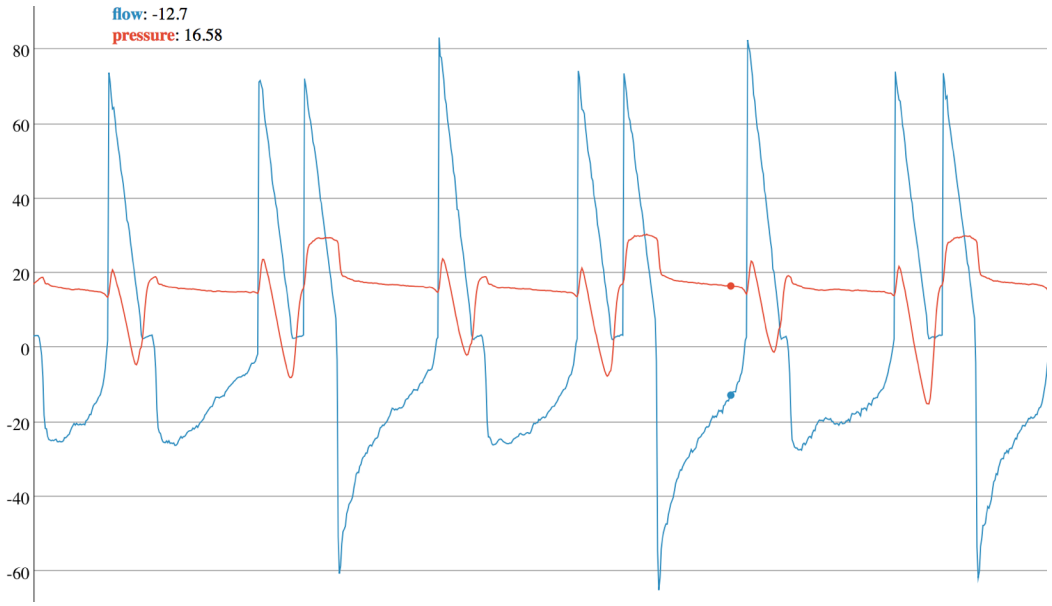

Figure 1: Flow-time (blue curve) and pressure-time (red curve) waveforms captured from the ventilator of a patient with severe acute respiratory distress syndrome. These waveforms are displayed in real time on ventilator user interfaces for diagnosis and management but are not routinely captured for secondary use or decision support.

A system to detect PVA could allow for major advances in both research and clinical care. New research questions about PVA and mechanical ventilation would be capable of being answered. Questions such as: “when does PVA occur most frequently,” “what are the

consequences of extended periods of PVA," and "what is the best way to provide alerts to doctors so that they can prevent PVA in their patients" will be capable of being answered. Furthermore, larger investigations can be performed answering the questions of whether or not PVAs lead to serious complications like ARDS and finalize the question about whether PVA's cause higher mortality in patients. This work has implications for the future of mechanical ventilation as well. One can envision an integration of these kinds of algorithms inside ventilators so that they can provide decision support and accurate, early detection of PVA.[19] These kinds of improved diagnostic support systems portend other benefits for clinicians: Physicians could be alerted remotely the moment PVA occurred, rather than only when they closely examined a patient. Such a system could lead to greater comfort for ventilated patients, lower usage of sedative, lower prevalence of ventilator induced lung injury, and could serve as a model for automating the detection of other deleterious events that are currently unavoidable in modern medicine.

Building a platform to detect PVAs is an expansive goal. To accomplish this, there were 3 independent software systems necessary for development. First a system of collecting and aggregating that data from ventilators to a central storage and analytics location was built. Some previous efforts to develop similar infrastructure have been successful, but external application can be limited by the fact a human needs to be actively involved in data collection, the monitoring device is so intrusive that it could affect patient care, or the use of high cost and proprietary products.[8,10,20-25] As a result we will developed our own infrastructure to solve these problems. To create robust algorithms for identifying PVA's we first needed to annotate sets of data, denoting and categorizing PVA and non-PVA events. To facilitate annotation, we constructed specialized software for annotating PVA occurrences in ventilator waveform data

(VWD). Finally, a computational model that is capable of identifying multiple types of PVA's was built. This model is capable of differentiating PVA events from normal breaths, and other abnormal breaths that are consequence of clinical artifact like suction procedures and coughing. The engineering of these components is enumerated in successive chapters of this thesis.

1.2 Background

Patient Ventilator Asynchrony

PVA is well-recognized as a major complication that can occur to a mechanically ventilated patient.[26] Despite its wide recognition there is still exist many questions surrounding PVA. These questions are often unable to be answered due to factors such as the unavailability of informatics infrastructure capable of collecting large amounts of quality data, and the tremendous amount of inter-patient, and inter-class variability that PVA takes in a clinical setting. This thesis does not attempt to give a comprehensive guide of all PVA subtypes found in the hospital, rather it attempts to give an understanding of two subtypes of PVA, double trigger asynchrony (DTA), and breath stacking asynchrony (BSA). Both of these types of PVA have been shown to be injurious to patients,[8,10,25,27,28] and we necessitate their accurate classification in retrospective VWD to determine how injurious they are and what clinical outcomes are associated with their rates, times of occurrence, and clinical state at occurrence, among other questions.

DTA is characterized by incomplete exhalation of inspired gas in between breaths due to sequential triggering of supported breaths, leading to little or no time for exhalation in between, a phenomenon referred to as dynamic hyperinflation, that is thought to injure the lung due to excessive distention of lung tissue. DTA is most recognizable by a characteristic double peak flow waveform where the first breath in a DTA sequence is not fully exhaled, and in the second

breath in sequence the patient exhales the amount of air received over the past two breaths. A canonical example of DTA can be found in Figure 2.

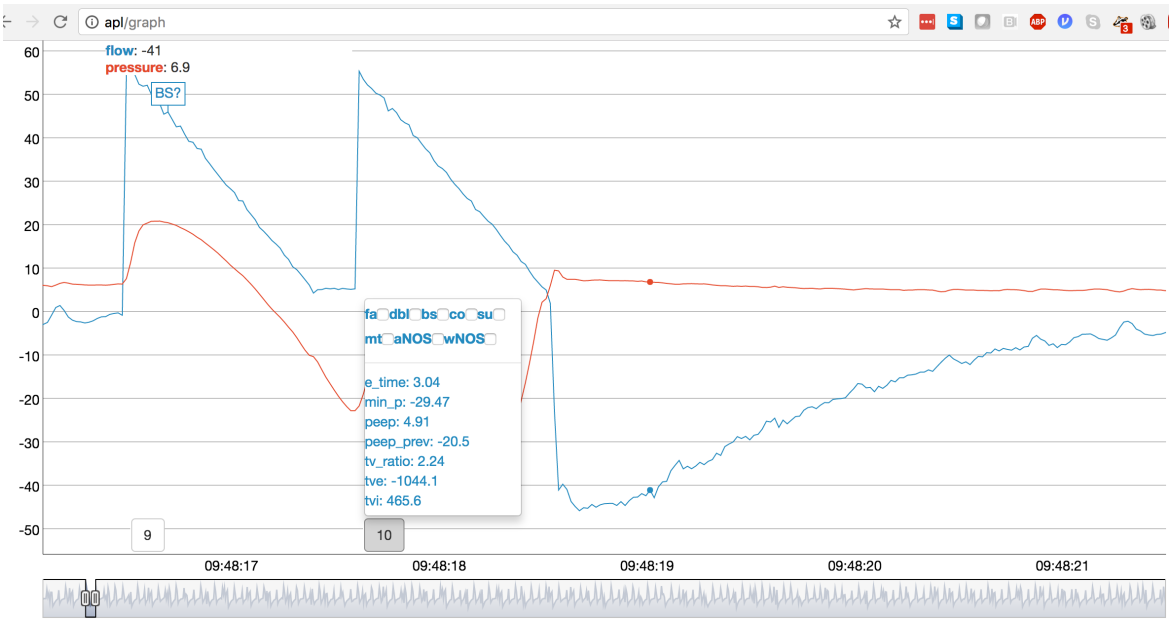

Figure 2: A canonical example of a DTA observation. In the first breath, the patient demands additional support beyond what the ventilator is programmed to deliver, resulting on patient-ventilator asynchrony. Ongoing inspiratory effort at the end of the first breath causes a new breath to be triggered without intervening exhalation, resulting in dynamic hyperinflation and potential lung injury. DTA, double trigger asynchrony

BSA is another PVA subtype characterized by incomplete exhalation between breaths that results in dynamic hyperinflation, and definitions of BSA and DTA vary across studies.[7,25,29,30] While these forms of PVA exist on a morphologic continuum, they differ mechanistically with DTA characterized by ongoing patient inspiratory effort at the end of a breath resulting in an immediate triggering of a second breath, and BSA resulting from inadequate time in between breaths often in the setting of a rapid respiratory rate, expiratory airflow obstruction, or both.[27,28] This mechanistic difference can be seen in Figure 3 where BSA is only a single breath sequencing resulting in dynamic hyperinflation, whereas DTA is a two breath sequence with the same result.

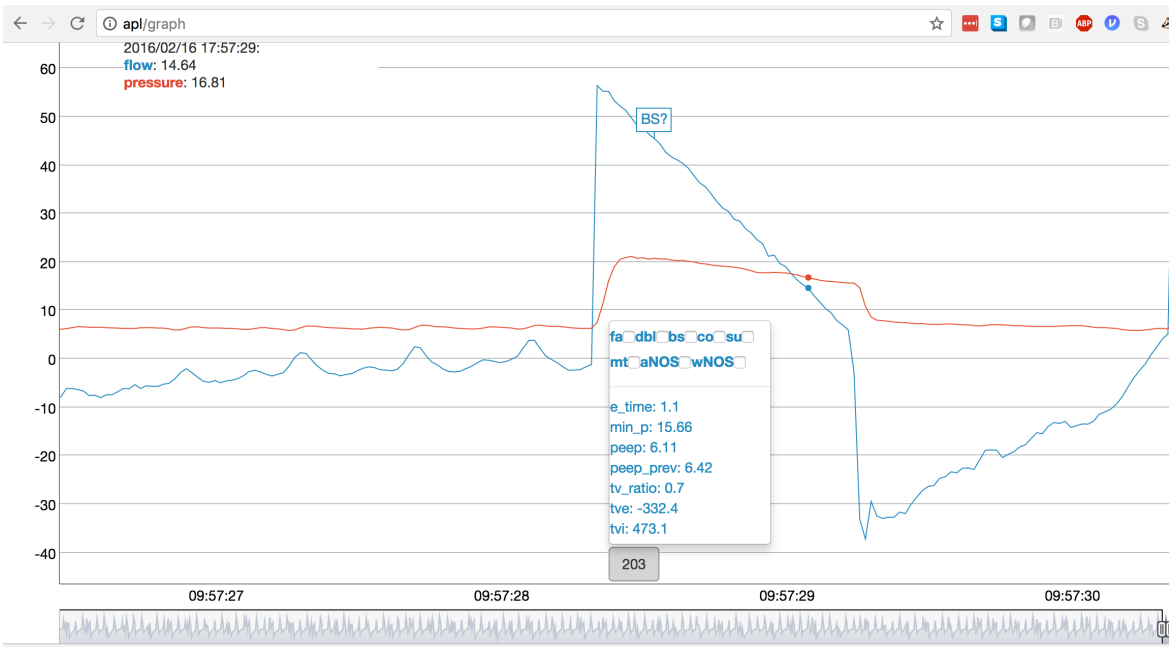

Figure 3: An example of BSA. In BSA, a patient attempts to exhale but expiratory time is too short to allow full exhalation in between successive breaths, resulting in dynamic hyperinflation. In this study, we define any morphologically normal breath where TVe/TVi < 0.9 and E-time > 0.3 seconds as BSA. In this case, the TVe/TVi is 0.7 and the E-time is 1.1 seconds, which we qualify as a breath stacking event. BSA, breath stacking asynchrony; TVe/TVi, expiratory divided by inspiratory tidal volume; E-time, expiratory time (seconds).

Non-PVA breaths make up the vast majority of patient breathing while mechanically ventilated. These breaths are generally indicative of normal, synchronous breathing, however there are frequent abnormal non-PVA breathing that must be accounted for. Some common types of abnormal, non-PVA breathing can be classified as clinical artifact, or breaths that are the result of normal, non-injurious clinical procedure like suction, or abnormal transitory breathing like cough. These types of breathing can look especially strange when viewed as VWD (Figure 4) and can potentially confuse a PVA detection algorithm. As a result, it was critical that we included both normal and abnormal non-PVA subtypes in any dataset we used to train classification algorithms.

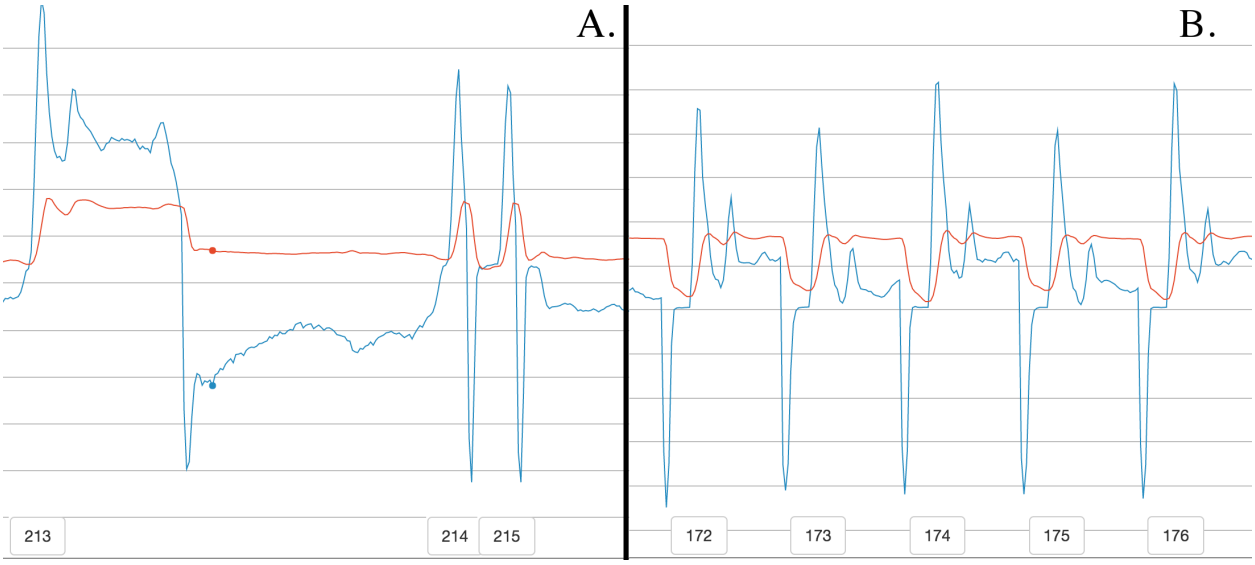

Figure 4: A. This image shows a normal breath at breath #213 followed by two coughs in quick succession. Cough can usually be visually identified by sharp inhalation and exhalation spikes in the flow waveform (blue). **B.** This waveform shows a series of suction events from breath #172 to #176. Suction artifact results from repeated triggering of breath delivery by the suction catheter, evidenced by multiple breaths delivered in succession without substantive exhalation in between breaths.[31]

Ventilator Waveform Data Acquisition

Like many other physiologically injurious events, immediate detection of PVA is possible via bedside examination, but detection can be delayed due to physician workload, the presence of signal noise or clinical artifacts such as coughing and suction events, or unfamiliarity with MV and PVA recognition [32] To ameliorate this problem, linking VWD to waveform data capture platforms would allow automated computer algorithms to perform the work of constantly monitoring patient breathing patterns, giving physicians the ability to instantly respond when a bout of PVA occurs.

Several efforts to develop waveform data acquisition infrastructure have been described, but these examples are limited in several important ways including the need for a human to be actively involved at multiple points in data collection; the intrusiveness of monitoring devices that could influence provider behavior or adversely affect patient care; or the use of high cost and proprietary products.[8,10,20-25] In an important proof of concept study for creating an open source,

reproducible data collection system, Howard demonstrated the ability to wirelessly gather data from PB-840 ventilators using a laptop computer and ad hoc Bluetooth and Wi-Fi local area networks. [33] Another important contribution was Howard's use of a Wi-Fi relay that could communicate with a remote laptop, presaging current efforts to use Raspberry Pi's as Wi-Fi relays. However, Howard's study relied on an ad hoc Wi-Fi network which limited the areas of the hospital where a researcher could enroll patients. There were also issues with Howard's system where only skilled users would be able to conduct study operations. As time passed technology has also changed, and Howard used the PalmPilot™, which is currently outdated.

Current technology utilizes internet of things (IoT) devices, such as Raspberry Pi's, as data conduits from medical devices.[34-38] OpenICE™ is another software platform aimed at demonstrating the feasibility of an open-source waveform data acquisition architecture using IoT devices. [24] There are however limitations to many of these solutions. Many solutions do not contain user guides or instruction manuals for reproducing their system. [37,38] OpenICE™ is feature rich, but does not have the capability for a researcher to passively store patient data. Instead, a researcher's personal computer must be on, and focused on storing the data at all times. This type of storage is called active storage, and is common in medical waveform storage.[18,22,24] Performing this type of data storage however, is impractical for a long term, continuous, multi-patient study. A major need is for passive data collection systems to replace active storage models, wherein a provider can simply attach IoT hardware to a medical device, and then allow the IoT device to automatically handle data collection. To our knowledge, there currently exists no fully functioning, inexpensive, open-source, research grade system that can provide passive storage capability for VWD.

Ventilator Waveform Data Annotation

After VWD is collected, it must be annotated to provide a ground truth for the development of any classification algorithm. Previous efforts to develop clinical annotation software like BEDA and ChronoViz have pioneered the ability to digest multiple streams of data, categorize time epochs, and output categorization results.[39,40] However, these packages lack specificity to the application of ventilator waveform analysis. In VWD analysis we find our time epoch is semi-regular, on a breath to breath level. Existing annotation packages do not have functionality to segregate time epochs by individual breaths, and performing this segregation using existing software would be overly laborious. Second, generic annotation platforms must be able to accept many different types of data for annotation. In MV, we only have two types of data necessary for visualization, flow and pressure. Understanding this, we can simplify a specialized annotation platform so we can annotate VWD much more quickly than we would be able to otherwise using more generic annotation software. Finally, it is recognized PVA annotation cannot be reliably performed by a clinician through solely visual inspection.[32] To provide a more reliable means of annotation we must instead display breath level metadata derived from the raw VWD. No annotation package exists to our knowledge that is able to split VWD on a breath by breath basis, derive breath level metadata, display it on-screen, and allow specific, clinically relevant, annotation of events.

Patient Ventilator Asynchrony Classification

For PVA classification algorithms, previous studies have relied on expert systems composed of heuristic logic, custom designed by clinicians.[17,25,30,41] These sometimes complex systems of rules can be accurate but have not always been rigorously verified using complete datasets of real patient data containing clinical artifact and other non-PVA breathing.[17,41] When

these systems have been rigorously verified, the logic embedded within the heuristic algorithm can be extremely complex and difficult to engineer.[30] To decrease the difficulty in engineering a PVA detection algorithm, we can instead turn to machine learning (ML).

ML has already yielded advances in other areas of healthcare analytics including the fields of ECG analysis,[42,43] cancer diagnosis,[44] photoplethysmogram (PPG) analysis,[45] early warnings of hemorrhagic shock,[46] and toxicology analysis,[47] among others. Most importantly, ML offers several advantages over heuristic classifiers: ML allows an algorithm to automatically derive detection rules from collected data instead of requiring access to the detailed clinical expertise required to define them manually; ML produces performant models that may be more accurate than logistic regression when used for classification tasks, especially when non-linear relationships exist between features and classes of interest; [48]; finally, ML may offer greater efficiency of model development especially in cases where signal noise, clinical artifacts, or high levels of intra-class variability may necessitate the creation of extensive heuristic logic to optimize performance. This is not to say that ML is not without its challenges: ML can be sensitive to data noise/corruption,[49] and can be affected by data imbalances.[50-52] Yet these hurdles are not debilitating: data noise can be compensated for by using outlier filtering algorithms and ensemble classifiers,[49] and data imbalances can be solved through usage of tools that create new synthetic samples for minority classes.[50] As a result, we believe that ML is worth a serious evaluation for creating VWD classification algorithms.

CHAPTER II: Designing Systems for Data Collection in the ICU

2.1 Design

The requirements demanded of a system to collect VWD were: (1) ensure temporally accurate data to preserve data linkage between collected VWD and EHR data.[53-55] (2) Enable reliable, continuous, and automated data collection from multiple mechanical ventilators simultaneously. (3) Must be nearly undetectable by providers so as not to influence patient care;[56,57] (4) include automated archival storage including backups, and (5) ensure ease of use by non-technical end users with data acquisition hardware and data management software. For collecting data, we chose Raspberry Pi (RPi) microcomputers. The RPi is a small, yet powerful computer that can be unobtrusively attached to a ventilator in a way that would not affect the level of patient care. For acquiring accurate time for all VWD we used hospital Network Time Protocol (NTP) servers to ensure all temporal data was accurate to a variance of 5 seconds.[58]

Since all VWD is written into a file on the RPi, archiving the data requires moving the files from the RPi to a central location. We first experimented with performing this process with graphical user interface (GUI) file transfer tools local to a clinician's laptop but found this process frequently resulted in lost or misattributed VWD. Since linkage of data to the patient is of critical importance when making temporal associations with EHR, we decided to pursue an automated solution of creating a web server to handle all archival transfer and storage. This way, we preclude the possibility of human error leading to data loss or misattribution. Furthermore, since the supervisor web application is able to automatically handle all necessary tasks for transferring of VWD from RPi's to a central collection point, study coordinators only need basic computer skill to collect large amounts of data.

Our final goal is to transfer VWD from the RPi to a database for analysis by tools like Python Pandas, Matplotlib, and Scikit-learn. [59-61] To do this we created a database storage plugin that automatically populates a database with the newly collected VWD. After all the VWD is collected and attributed using the supervisor application, we store it in a database so that it can be queried for further analytics.

2.2 Implementation

Data Collection.

The RPi's have two necessary hardware components for collecting VWD: an RS-232 to USB cable and a power cable. We wrote specifically designed software for capturing ventilator data in Python using only 228 lines of code. Initialization of data acquisition was simplified so that the RPi only required the input of the RPi power supply and the connection of the RS-232/USB cable to the PB-840 and RPi. Once connected, our software code automatically begins passive collection of VWD from the ventilator. To remain unobtrusive, RPi's can be attached to a mount located behind the ventilator monitor to remain unseen and not influence patient care (Figure 5).

Upon connecting to a ventilator via RS-232/USB cable (Figure 5), the software on the RPi performs the essential task of connecting to an NTP server. After connecting to NTP, a timestamp is written at the top of the file so that we can later feed-forward the time of each observation. This feed-forward rate is based on the 50 hertz, fixed sampling rate of the PB-840. If an NTP server is unable to be contacted data collection does not begin.

VWD is split into two columns, air flow and pressure. The PB-840 demarcates data by breaths, using BS (breath start) and BE (breath end) tokens in the data to respectively display

when a breath has started and ended. Data is stored locally on the RPi until transfer to an intermediate storage location. We initially attempted to collect data in a format such that a date timestamp was attached to each observation. It was believed that this system would be more accurate and require less post-processing than a feed-forward system. However, this method yielded computational race conditions in our software resulting in data corruption. As a result, we opted for the less computationally complex system of feed-forward timestamping.

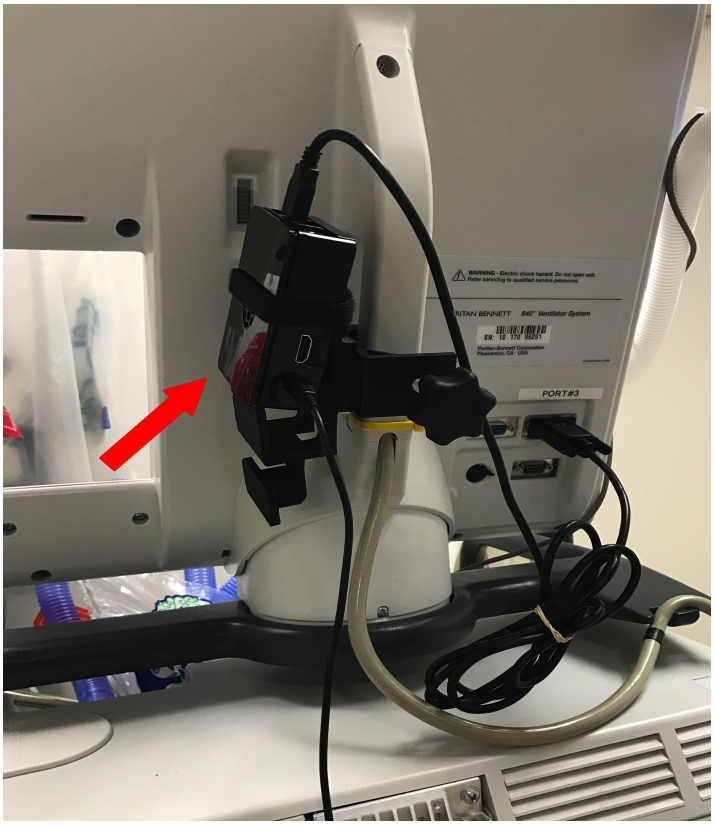

Figure 5: An image of the Raspberry Pi attached to the ventilator. The red arrow is pointing to the Raspberry Pi which is positioned behind the screen of the monitor. Because the RPi is attached behind the monitor it doesn't influence patient care, while still performing the task of data collection from the PB-840.

Data Aggregation

Management of the RPis and data aggregation is performed by our supervisor web application. It is a GUI for the following operations:

- *List* all VWD files on the RPis.
- *Transfer* select VWD files from the RPi to an intermediate storage location for further processing.
- *Backup* files on the RPis to prevent catastrophic data loss.
- *Delete* VWD files on the RPis once they have been transferred to another location.

Absent a full time study coordinator, the issue of data linkage can be a major problem in the ICU since an RPi be switched to a new patient before being cleaned of a previous patient's VWD. To solve this problem, new files are made every two hours and file names include date/time stamp. Then, in the supervisor application, we manually select files belonging to a specific patient by identifying the time on the ventilator described in the EHR and matching that to a corresponding file on the RPi. Although this process takes manual intervention and cross reference of the EHR to perform effectively, it also helps mitigate the problems of data misattribution, which could potentially force researchers to discard collected data.

Data Storage

To accomplish this, we created a plugin for the supervisor web application that automatically uploads VWD into a database after the VWD finishes transferring from an RPi. Using temporal associations encoded with the VWD, we can then analyze the VWD with specialized computing algorithms or combine it with EHR data sources to create a more complete, and data driven picture of patient state (Figure 6). Using this method, we have collected a dataset of over 450 patients with ventilation data. At the time of writing this comprises approximately 150 GB of VWD.

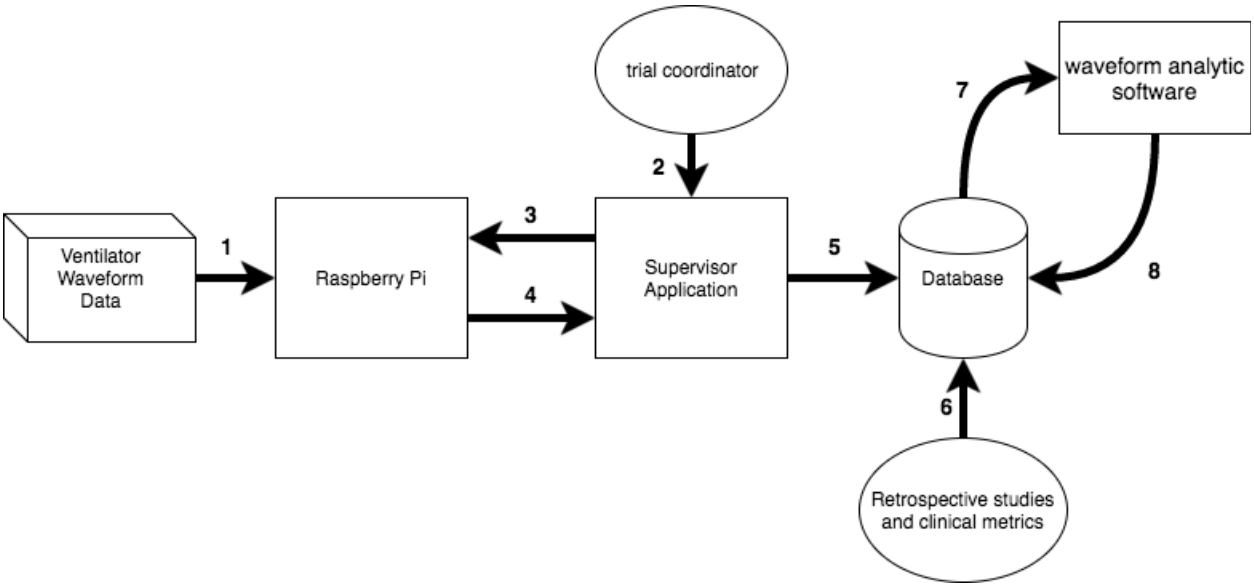

Figure 6: Schematic of our novel data acquisition and management infrastructure. 1.) Raw ventilator waveform data is sent to the Raspberry Pi via serial connection 2.) Trial coordinator visits supervisor application, and 3.) chooses raspberry pi to extract and files from. 4.) Files are to extracted from the raspberry pi, and labeled with a patient unique identifier 5.) All data is then translated to SQL and stored on a database. 6.) Retrospective studies or clinical improvement measurements can then be derived with the stored data. 7) VWF are processed using waveform analysis software to generate breath-by-breath physiologic metadata and detect relevant clinical events. 8) Time-stamped data derived from VWD processing are uploaded into the database for subsequent research and quality improvement studies.

Scalability and Adding New Devices

Should additional RPi devices be required for VWD collection, study coordinators will need some way of placing our software onto the brand new RPi's. To increase usability, initializing the individual RPi devices is meant to be simple. We have created images--or files containing our operating system-- that can store preconfigured details of our software. We can then flash the microSD card for the RPi with this image. This decreases the amount of steps necessary for RPi deployment. After flashing the microSD card, there is an additional step of configuring an NTP peer, which we accomplish from terminal or automatically using Ansible, an IT automation platform.[62]

2.3 Summary

We designed a hardware platform using Raspberry Pi microcomputers with the software necessary to collect, aggregate and store VWD for secondary use in research purposes. We have also demonstrated the platform's use and utility in the hospital by gathering data for over 450 patients. This system, and it's efficacy of data collection, especially in absence of a clinical trial coordinator, can provide an effective example at the development and operation of a research grade passive storage system that is able to collect VWD automatically upon being connected to a patient's ventilator

CHAPTER III: Annotating Clinically Relevant Data

3.1 Design

The Annotation PipeLine (APL) had the goal of being a generalizable VWD annotation software platform. To accomplish this, we set APL to have the following 4 requirements: (1) Must be able to graph VWD data and process/display breath level metadata. (2) The software must be fast in rendering and processing to maximize the time resources of clinician reviewers, and (3) provide some method of annotation that is reliable enough where different reviewers with a standardized methodology can come to similar categorization conclusions. (4) The system must be able to output categorization results in a standardized format.

For graphing of raw VWD data we settled on the Dygraphs software package to display relevant data.[63] Using Dygraphs' application program interface (API) we could sequentially number breaths and then display relevant metadata when requested. Another advantage of Dygraphs is its speed. When using ChronoViz we could not display files containing more than 100 breaths because the software would slow to the point of being unusable, whereas Dygraphs has the ability to render thousands of breaths at a time while maintaining responsiveness.

Real time annotation is what truly made APL unique compared to previous waveform annotation efforts, and is what enabled clinicians to make uniform judgements in categorizing asynchronous breaths by PVA type. Since clinicians cannot accurately identify all PVA subtypes by simple visual inspection[32] we instead added additional breath specific metadata to assist clinicians in making a judgment. This way clinicians could create classification criteria using metadata thresholds coupled with visual inspection, thereby generating more objective classifications than visual inspection alone. If a breath matches classification criteria for a certain PVA then it could be annotated with a PVA or clinical artifact subtype. Finally, when the

requisite breaths were annotated, the results were output to a standardized comma separated value (CSV) file so the classifications could be used to generate, train, and validate algorithms for PVA detection.

Figure 7: The user interface screen of a Puritan-Bennett 840 ventilator. This displays valuable information to providers such as the pressure (top waveform) and the flow (bottom waveform) of air to a patient during mechanical ventilation. On bottom options are available for modifying ventilator settings based on physiologic conditions and provider care preferences. On top summary statistics are available to providers so that current settings and conditions are displayed in a quickly understood format.

3.2 Implementation

Rendering

The first step in the process of annotation was uploading the raw VWD data from a ventilator to APL, after which APL processed the data into a format that was consumable to be graphed. APL then rendered the MV data to a form that mirrors how it would be seen by a clinician at the bedside (Figure 7). Initial rendering of pressure and flow data streams was performed by Dygraphs and then a second rendering action was performed by Dygraphs that places relative breath numbers in accordance to where each breath is graphically located. Finally,

a third rendering action was performed that made APL capable of displaying breath level metadata and performing real-time annotations when requested. When requested, APL displayed the following breath level metadata for identifying PVA and other clinical artifacts:

1. Tidal Volume Expired (tve) – the amount of air expired during a breath
2. Tidal Volume Inspired (tvi) – the amount of air inhaled during a breath
3. Tidal Volume Ratio (tv_ratio)– The ratio of expired tidal volume to inspired tidal volume for a breath
4. Expiratory Time (e_time) – The amount of time a patient spent on exhalation for a breath
5. Positive End Expiratory Pressure (peep) – Ventilator airway pressure at the end of expiration. This is also known as PEEP.
6. Previous Positive End Expiratory Pressure (peep_prev) – The last breath's PEEP
7. Minimum Pressure (min_p)– The minimum pressure reading in breath.

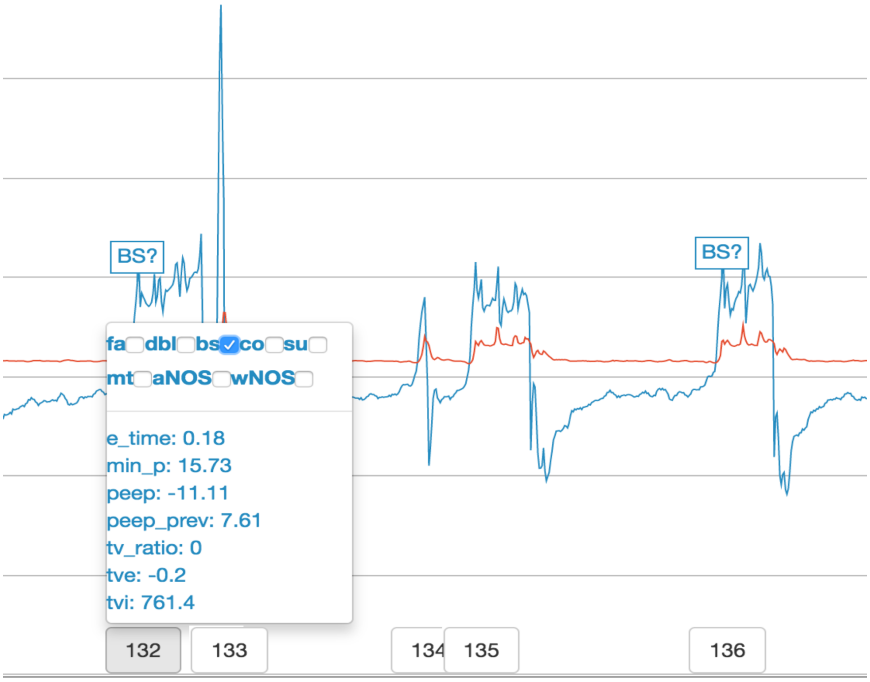

Figure 8: Here we display our novel interface for annotating ventilator waveform data (VWD). Breaths are labeled according to their number relative to other breaths in a file. By clicking on the breath number we can obtain summary statistics for the breath. We also obtain the ability to assign various annotation labels to a single breath. The Annotation PipeLine also offers helpful markers to help the user identify potential breath stacking asynchrony events. These markers facilitate user interaction when classification of a breath may be based on some statistic rather than an event easily discernable by human sight.

Annotation

Using APL, we were able to annotate 4 different PVA's and 4 clinical artifacts that manifested through mechanical ventilation. Once a reviewer had found a PVA they wished to annotate they selected the breath number assigned by APL for the breath, reviewed the metadata output, then assigned a classification by selecting a checkbox (Figure 8). The PVA's able to be annotated were: flow asynchrony (fa),[29] breath stacking asynchrony (bs),[25] double trigger (dbl),[21] and asynchrony not otherwise specified (aNos). The clinical artifacts we could annotate are: cough (co), suction (su),[64] multi-trigger (mt),[65] waveform not otherwise specified (wNos). Finally, when the reviewer was satisfied that they had annotated all breaths necessary, they input the range of breath numbers they desired processed. The processed breaths were output into a comma separated value (CSV) file so it can then be used for algorithm development, and validation.

3.3 Summary

APL served a vital role in the annotation of data collected from a ventilator so that we could develop novel algorithms for the detection of PVA. Using APL, we could visualize VWD and display metadata for each breath we wished to annotate. APL gave clinicians a visual, data-driven approach to perform VWD annotation, allowing for a more rapid, and accurate annotation process. When we combined this with our data collection and aggregation architecture we were given all the software components necessary to begin constructing algorithms for detecting PVA.

CHAPTER IV: Creating algorithms for PVA detection

4.1 Methods

Dataset Description and Feature Extraction

In the preceding chapter we discussed the software platform we designed to annotate breath level VWD so that we could generate a ground truth dataset for training and validating PVA detection algorithms. In our study, we use a dataset of VWD from 35 distinctive patients composed of a total of 9719 breaths. VWD was captured from the ventilator using the Raspberry Pi™-based data collection system. VWD files were manually analyzed by two ICU physicians to identify 300-350 breath regions of interest (ROI) where PVA was present, and each breath was manually annotated to determine the breath type. Disagreements in classification were adjudicated by a discussion between the clinicians before reaching a consensus. Each breath was then analyzed to derive clinically-relevant metadata described in Table 1, which could then be defined as features for use in ML model development. [30,66]

Variable Name	Units	Description
TVi	Milliliters/second	inspiratory tidal volume, defined as the integral of the flow-time curve values from BS to point where flow crosses 0 (x0).
TVe	Milliliters/second	expiratory tidal volume, defined as the integral of the flow-time curve values from x0 to BE
TVe/TVi	Unitless	the ratio of expiratory tidal volume to inspiratory tidal volume
I-time	Seconds	the time from BS to [x0 minus 1 time point]
E-time	Seconds	the time from x0 to BE
I:E ratio	Unitless	the ratio of the I-time to the E-time
RR	Unitless	instantaneous respiratory rate, defined as 60/breath time
PIF	Liters/minute	peak inspiratory flow, defined as the maximum positive flow recorded from BS to [x0 minus 1 time point]

PEF	Liters/minute	peak expiratory flow, defined as the most negative flow recorded from x0 to BE
PIP	cm H2O	peak inspiratory pressure, defined as the maximum recorded pressure from BS to [x0 minus 1 time point]
Mean flow from PEF	Milliliters	The mean flow observation from the point in time PEF occurred to the point where the breath terminated and a new one begins
ipAUC	cm H2O	the inspiratory pressure area under the curve, defined as the integral of the pressure-time curve from BS to [x0 minus 1 time point]
epAUC	cm H2O	the expiratory pressure area under the curve, defined as the integral of the pressure-time curve from x0 to BE
PEEP	cm H2O	positive end-expiratory pressure, defined as the average of the last 5 data points from the pressure-time curve of each breath
Paw	cm H2O	mean airway pressure
Dynamic Compliance (C_{dyn})	Unitless	The pulmonary compliance of the lung at any given point during a breath.[67] This measure is derived via : $C_{dyn} = \frac{TVi}{PIP-PEEP}$

Table 1: List of all metadata variables along with a description. These variables were all processed from raw ventilator waveform data and were evaluated as independent features to add to our PVA detection model.

Using all available metadata in modeling may result in lower performance or longer computation time, so we applied feature selection,[68] to reduce the dimensionality of our dataset, speed model training time, and improve model performance.[69,70] In general, feature selection can be performed through expert knowledge, or using computational methods. ML practitioners generally first seek expert opinion on a problem to perform feature selection[69,71,72] and many medically useful algorithms are informed by the inclusion of features defined by expert clinicians.[42,73,74] In contrast, computationally-based feature selection methods may be more reproducible and do not require repeated consultations with clinicians. In the present study, we compared expert opinion to feature selection driven by computational methods. Our experts

performed manual feature selection from variables defined in Table 1. Computational feature selection was conducted using multiple methods including the Chi-square test,[75] L1-regularization with a Support Vector Machine,[76], Recursive Feature Elimination,[77] PCA,[78] Linear Discriminant Analysis,[79] and Independent Component Analysis.[80]We present results of the Chi-square test, in conjunction with successive sensitivity analyses, due to its superior performance compared to the other methods when applied to our use case.

Classification Methodology

To detect PVA events, we developed a supervised ML model. For training and validating the model, two ICU physicians manually annotated each breath as one of 3 categories: non-PVA, double trigger asynchrony (DTA), or breath stacking asynchrony (BSA). Clinical artifacts in the VWD caused by suctioning of patient secretions and cough events were included in the category of non-PVA breaths. As shown in Table 2, we annotated 7039 instances of non-PVA breath, 752 instances of DTA, and 1928 instances of BSA.

Event Type	Number	Percentage
Normal (non-PVA and non-artifact)	6548	67.37
Cough	123	1.27
Suction	368	3.79
Non-PVA (normal + suction + cough)	7039	72.45
Double trigger (DTA)	752	7.74
Breath stacking (BSA)	1928	19.83

Table 2: Event types and rates. One thing to note from this table is how rare PVA actually is in our dataset. In total PVA only comprises 27.57% of all breaths annotated. Cough and suction is even more rare, representing slightly over 5% of all breaths annotated. PVA, patient ventilator asynchrony

Clinical artifacts like suction and cough were explicitly identified and included in the training and validation datasets because they share morphological characteristics with common forms of PVA that can result in false-positive PVA classification.[30] We thus considered it

essential to include artifacts in model development to decrease the false positive detection rate. With respect to PVA, we only targeted DTA and BSA in accordance with the scope of this thesis.

Because it is difficult to rely solely on visual inspection to classify PVA [32], we utilized additional heuristics to perform DTA and BSA classification and create a gold standard set of classified PVA observations for our supervised learning algorithm. Our clinicians used a heuristic algorithm that utilizes both visual inspection and clinical metadata for DTA classification, in order to better incorporate the types of DTA seen clinically[7,21,81]. We defined DTA as any non-artifact sequence of two breaths in rapid succession where the first breath is defined by $eTime \leq 0.3$ and $TVe/TVi \leq 0.25$, or when the first breath has $eTime \leq 0.3$, $TVe/TVi \leq 0.5$, and $TVe \leq 100\ ml$.

Existing definitions of BSA in the literature are inclusive of DTA, but may fail to identify less extreme events leading to dynamic hyperinflation, and do not necessarily distinguish between events resulting from ongoing inspiratory effort versus those associated with inadequate expiratory time.[7,8,25] In order to include a broader range of breath stacking events than existing definitions and distinguish them from DTA, we defined BSA is any non-artifact breath where $TVe/TVi < 0.9$ and $eTime > 0.3$ (Figure 3). Where the TVe/TVi threshold of < 0.9 was used to account for the inherent inaccuracy of the ventilator's flow sensor.

Despite enrichment for PVA and artifacts in our training and validation data sets, the relatively low proportion of abnormal breath types resulted in a class imbalance problem.[82,83] Imbalanced training sets can often be an obstacle to training accurate classifiers when learning methods assume a balanced distribution of classes.[84] To address this problem, we used "synthetic minority over-sampling technique" (SMOTE), which mitigates the class imbalance problem by

creating synthetic samples of minority class observations using the K-Nearest Neighbor algorithm to estimate where to construct new samples.[50] We used SMOTE with a 1:1 ratio for minority class to majority class observations, thereby creating the same number of DTA and BSA observations while keeping non-PVA observations static. Specifically, SMOTE created a dataset of 9719 DTA, 9719 BSA, and 9719 Non-PVA observations.

Model Evaluation

To perform supervised ML, we defined the classifier function *f* as follows: [85]

$$y = f(X)$$

Where X represents the input observations for each breath and *y* indicates the classification result. We defined X to be a matrix such that $X = \{x_1, x_2, \dots, x_n\}$ where each $x_i \in X$ corresponds with a single breath. Each x_i takes form $x_i = \{b_{i1}, b_{i2}, \dots, b_{im}\}$ where b_{ij} can be defined as the observed value of a feature for a specific breath. We define *y* as some 1-dimensional vector where $y = \{y_1, y_2, \dots, y_n\}$. When performing binary classification $y_i \in \{0,1\}$. In multi-class classification $y_i \in \{0,1, \dots, k\}$, for $k + 1$ different classification states in our problem.

To learn and evaluate our proposed model, we split our database into two parts; a training and a testing set .[70,85,86] The training set is used to develop a model while the testing set is used to evaluate each model's performance in a unique data set. Training each model involves a process of taking repeated calculations of expectation of a loss function $L\big(y, f(X)\big)$ that updates the function f to f^*.[87]

$$f^* = \arg \ min_f \ E_{y,X} \ L(y, f(X))$$

Here $E_{y,X}$ is the expectation function and through repeated minimizations of this, we eventually arrive at an optimal f^* to use for testing.

In our study, we utilized four well-known classifiers implemented within the Scikit-learn[60]: (i) Random Forest (RF), (ii) Multilayer Perceptron (MLP), (iii) Extremely Randomized Trees classifier (ERTC), and (iv) Gradient Boosted classifier (GBC). The RF classifier uses the classification and regression tree (CART) algorithm[88] to perform tree splitting and the cross-entropy criteria to minimize the impurity function[89]. The MLP uses backpropagation[90] with the *tanh* activation function and the cross-entropy loss function.[91]To improve model variance, tree splits in ERTC are performed randomly[92] and the *gini* criteria[88] defines which splits are best. Finally, we implemented the GBC to use deviance for its loss function.[87]

For model training and testing, we used cross patient learning to segregate specific patients into a training cohort, and others into a testing cohort. We performed this type of evaluation as compared to using the holdout method because, using holdout, a single patient's observations may become mixed into the training and testing sets, which may introduce bias and not generalize well to subsequent patients.[74,85,93] This bias can be caused by intra-patient waveform similarities resulting from static ventilation settings and other patient-specific physiologic factors. The training cohort was then used to parameterize our model, and the resulting model was validated using data from the withheld testing cohort. Classification metrics were evaluated in a leave-one-subject-out cross validation to avoid any circularity of using the same patient's data in both our training and testing sets.[94,95]In total, this yielded 35 kfolds for use, in correspondence to the number of patients in our dataset. In each kfold, the true positive, true negative, false positive, and false negative counts were saved and then analyzed later to give a final gauge of model performance across patients.

For evaluating the efficacy of our model, we used sensitivity and specificity as our primary metrics. Sensitivity and specificity are two traditional methods for validating alerts in medicine[96-99] and are calculated as follows:

$$sensitivity = \frac{True\ positives}{True\ positives + False\ negatives}$$

$$specificity = \frac{True\ negatives}{True\ negatives + False\ positives}$$

Sensitivity, also known as or *recall*, describes how effectively a model classified PVA when it was present, and is important for maximizing true-positive event detection.[100] Specificity was used to describe how well a model performed when classifying non-PVA breaths and is important for minimizing false-positive event detection. Optimizing each model's specificity was considered paramount to allow these or similar models to be translated to eventual clinical practice, in order to avoid potential alert fatigue resulting from frequent false positive alarms.[101]

Positive and negative predictive values were not used here since application of these metrics would require unbiased sampling to estimate the true rates of PVA and non-PVA in the general population of ventilated patients. Our use of manual ROI selection to enrich for representative samples of PVA and artifacts was deemed necessary after initial efforts at random sampling resulted in gross under-representation of non-normal breath classes given that the relative sparsity of these events, which tend to be interspersed amongst large regions of normal breaths in typical patient files.

4.2 Results

Classifying PVA versus Non-PVA: A Binary Classification

Computationally Finding the Optimal Feature Set

We first attempted to classify a given breath as PVA or non-PVA, and created a model to classify either BSA or DTA, but not both, and non-PVA breaths. It is known that reductions in feature space can minimize the problem complexity caused by the "curse of dimensionality" while potentially improving performance of the model.[69] We thus performed this reduction on our list of total features from *Table 1*. We also developed models using a set of expert-selected features for DTA and BSA detection. For DTA our expert features were the I:E ratio, E-time, TVe, and TVe/TVi. For BSA our expert features were the E-time and TVe/TVi. Note that we did not reduce the set of expert-derived features, since those features were known to be effective from previous study.[30] In the absence of expert input, we applied cross patient learning with the Chi-square test combined with successive sensitivity analyses to inform our feature selections. For speed purposes, we use an RF with 10 estimator trees. In this evaluation, we did not use SMOTE initially so that we could evaluate model performance without synthetic dataset additions. In the absence of SMOTE, our sensitivity analyses showed that sensitivity and specificity for the detection of BSA (Figure 4) and DTA (Figure 5) were improved with use of all available metadata features in the model. While sensitivity and specificity generally remained stable or improved with increasing feature number across both models, sensitivity for DTA classification remained poor in all tested conditions and specificity remained just over 90% (Figure 5).

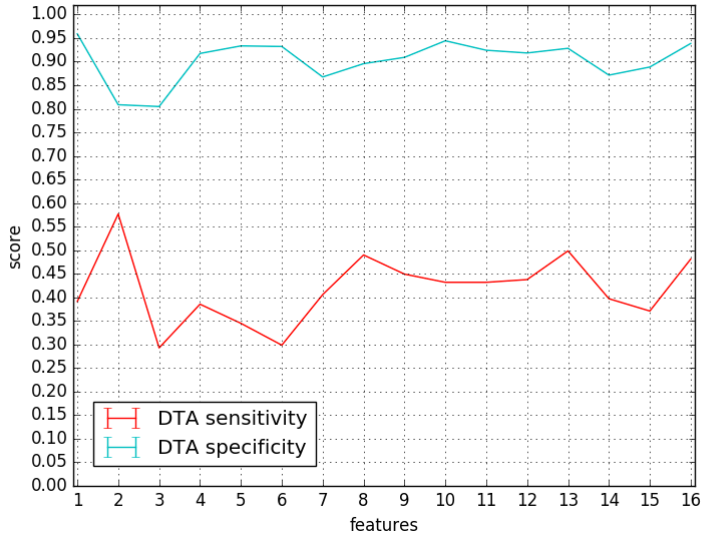

Figure 9: Chi-square sensitivity analysis for DTA without SMOTE. Our analysis suggests optimal model performance using all 16 metadata features. Note the low DTA sensitivity of this model when SMOTE is not used.. DTA: double trigger asynchrony, SMOTE: synthetic minority over-sampling technique.

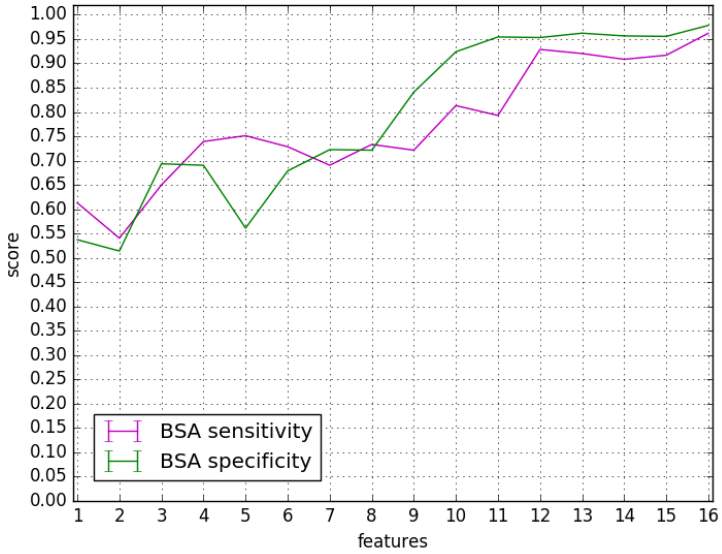

Figure 10**:** Chi-square test with sensitivity analysis detecting BSA. We found all 16 features to be the optimal number of features to be used. BSA, breath stacking asynchrony

Binary Classification Model Results

Given the overall poor DTA classification performance of our RF model without SMOTE in Figure 9 and from other experiments (see Appendix B, Figures B14-B17), we next compared

the performance of four different machine learning classifiers, with the addition of SMOTE, to determine if model performance would improve over our baseline RF model. However, even with the addition of SMOTE, DTA performance was disappointing when using metadata features, and the best performing classifier only achieved 70% sensitivity and 91% specificity. Expert-derived features improved DTA detection performance, but still leaves much room for improvement.

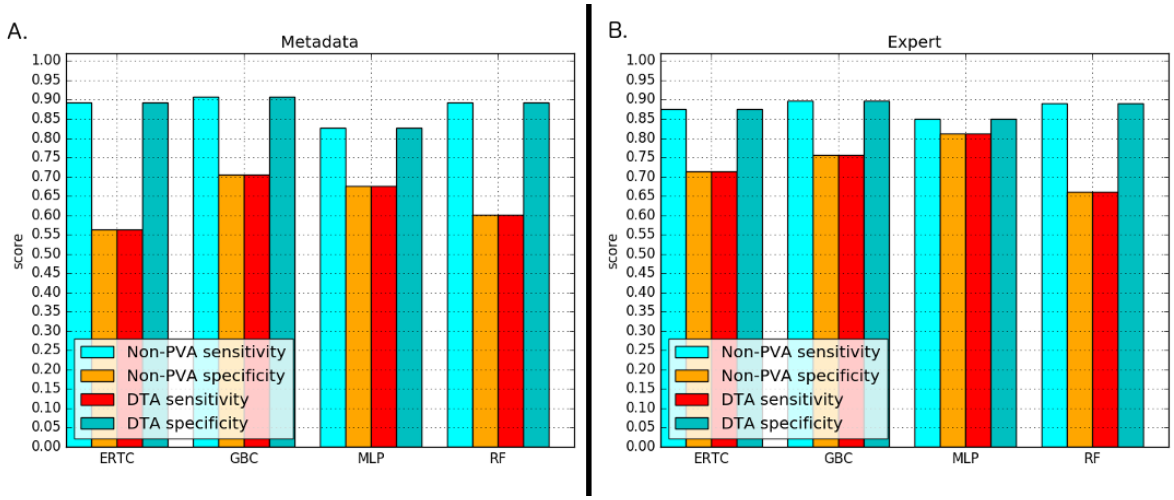

Figure 11: A. DTA detection model using all metadata features. GBC yields the best model with 70% sensitivity, while other classifiers yield inferior models. **B.** Using expert-derived features, DTA detection improves with all algorithms, but still could use improvement. DTA, double trigger asynchrony; ERTC; extremely randomized trees classifier; GBC, gradient boosted classifier; MLP, multi-layer perceptron; RF, random forest

BSA detection on the other hand, was excellent and performed equivalently using all metadata and expert features, with greater than 95% sensitivity and specificity in all but one of the four classifiers (Figure 12).

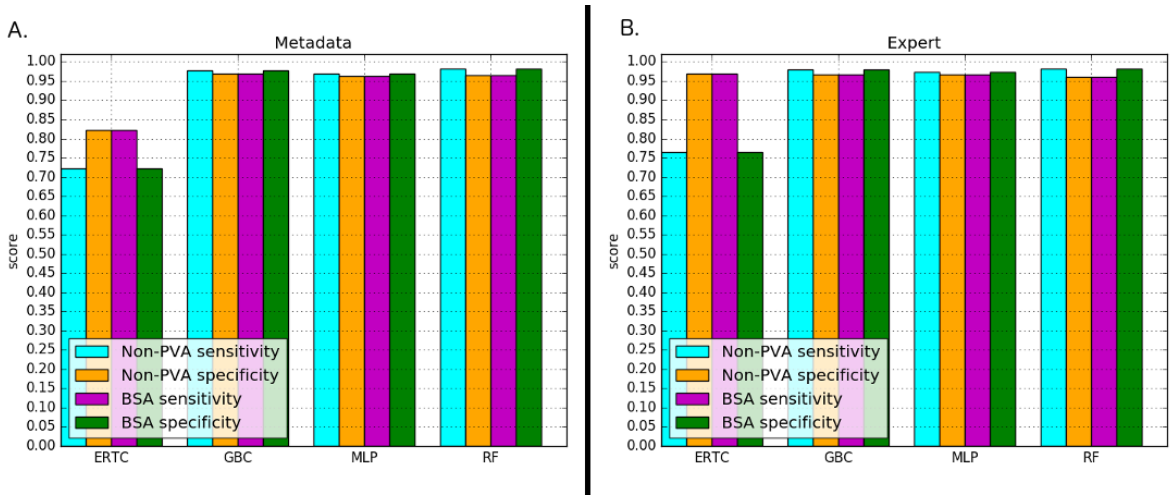

Figure 12: A. BSA detection using all metadata as features. Here, our feature set performs very well in all classifier algorithms with the exception of the ERTC. **B.** BSA detection using expert derived features. Here, our highest performing algorithms exhibit equivalent performance to the metadata model, while the ERTC improves its sensitivity score. BSA, breath stacking asynchrony; ERTC, extremely randomized trees classifier; GBC, gradient boosted classifier; MLP, mulit-layer perceptron; RF, random forest.

Cross-Patient Learning with Time Varying Features

As shown in previous sections, DTA classification models performed poorly using either all available metadata features or expert-derived features. Because DTA is a two breath sequence, we reasoned that model performance would improve if retrospective features were added to the models. To take this two breath sequence into account, we added *retrospective features* to our model, or features that look backwards at the metadata for a previous breath. This adds an additional 16 features to any possible model. A Chi-square sensitivity analysis showed that 21 features out of 32 metadata and retrospective features were most effective to identify DTA (see Appendix B Figure B18).

We next examined how inclusion of expert-derived retrospective features would compare to our mathematically-derived metadata feature set and created a new set of expert-derived features from the set of *retrospective features*, naming this set the "*retrospective expert features*." This set was comprised of TVe/TVi, the previous breath's TVe/TVi (TVe/TVi-

previous), and the previous breath's E-time (E-time previous). These features were directly derived from the heuristic algorithm our clinicians used to identify DTA.[30] In Figure 13 we show the results of the chi-squared model versus the expert model. We also attempted a similar experiment with BSA models but saw no improvement above using single breath features (see Appendix B Figure B20).

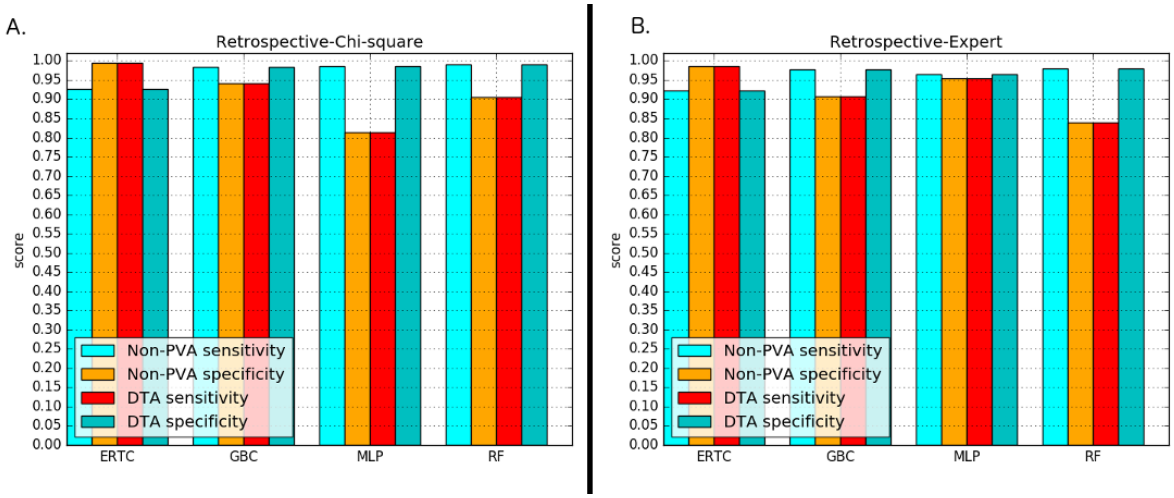

Figure 13: The use of a simplified dataset including retrospective, expert-derived features improves DTA sensitivity and specificity considerably. **A.** We use the set of 21 features chosen by Chi-square analysis for our DTA model. **B.** Model using the set of *retrospective expert features*. The features used here were TVe/TVi, TVe/TVi-previous, E-time-previous. DTA, double trigger asynchrony; TVe, expiratory tidal volume; TVi, inspiratory tidal volu me; TVe/TVi, expiratory divided by inspiratory tidal volume; E-time, total expiratory time (seconds). TVe/TVi-previous, the previous breath's TVe/TVi; E-time-previous, the previous breath's E-time. ERTC; extremely randomized trees classifier; GBC, gradient boosted classifier; MLP, multi-layer perceptron; RF, random forest

Classifying Multiple PVA Types: A Multiclass Classification

In Figure 13 we see that the DTA model with the best sensitivity is the one using *retrospective expert features* and the MLP classifier. So, to create a model that can detect both DTA and BSA events we used the 3 *retrospective expert features* as a starting point. Then to improve the specificity of our model, we added 4 additional features, which were guided by Chi-square sensitivity analyses: TVe-previous, RR, RR-previous, and RR-twice-previous. The RR-twice-previous feature is the RR of two breaths before the current breath. The final features used

in the multi-class classification model were thus: TVe:TVi, TVe:TVi-previous, E-time-previous, TVe-previous, RR, RR-previous, and RR-twice-previous.

For our final model, we created an ensemble of an ERTC, 6 GBC, and 3 MLP classifiers to detect both DTA and BSA. We did not use the RF in the ensemble because its DTA prediction sensitivity adversely affected the model. The ensemble performed its classification by using the prediction of the classifier with the highest prediction probability in the ensemble.[60]. We initially applied this ensemble approach without SMOTE, and although BSA detection performed well, DTA detection once again suffered from poor sensitivity performance (see Appendix B Figure B21). We next conducted the same experiment after applying SMOTE. Table 3 shows how the three models performed differently with regard to sensitivity. and specificity of detecting DTA and BSA.

Algorithm	Class	Accuracy	Sensitivity	Specificity
Ensemble	**Non-PVA**	0.971	0.9673	0.9806
	DTA	0.9742	0.9601	0.9754
	BSA	0.9793	0.9445	0.9879
ERTC	**Non-PVA**	0.7245	0.6744	0.856
	DTA	0.8693	0.9934	0.8589
	BSA	0.7683	0.5835	0.814
GBC	**Non-PVA**	0.9707	0.9692	0.9746
	DTA	0.9745	0.9335	0.9779
	BSA	0.9779	0.9445	0.9861
MLP	**Non-PVA**	0.954	0.9439	0.9806
	DTA	0.9576	0.9628	0.9572
	BSA	0.9678	0.9155	0.9807

Table 3: Descriptive statistics for the all classifiers run on the multiclass classification problem using SMOTE. ERTC: Extremely Randomized Trees classifier. GBC: Gradient Boosting classifier. MLP; Multi-layer Perceptron

4.3 Summary

In this chapter we have described the creation and validation of a machine learning classifier to detect both DTA and BSA, two of the the most studied types of PVA in existence.[8,21,25] We intended this study to not just be practical and effective, but also to serve as a model for detecting other types of PVA. Previous methods for detecting PVA have relied on heuristic models of rules custom crafted by clinicians. Machine learning in contrast, allows an algorithm to derive these same detection rules for us. Using computational tools like SMOTE we could overcome the problem of low sample sizes biasing our results. Through careful cross examination of feature sets, we could determine the best set of data to use for our feature sets. Through usage of ensemble classifiers, we could combine multiple algorithms that were good at detecting one type of PVA into a single classifier that combines the best attributes of its components. Our investigation tells us that expert clinician knowledge will still be relevant for creating machine learning models, given that we necessitated it's usage for creation of our final feature set. Usage of the Chi-square test was helpful in reducing our data dimensionality in many cases, but was still inferior to expert knowledge when constructing a final feature set for PVA detection. We believe all these lessons will be important for clinicians and researchers in the future.

CHAPTER V: CONCLUSION

In this thesis, we outlined the creation of a data collection system to collect VWD from mechanical ventilators, annotation software to create a ground truth, and an ensemble machine learning model for classifying two common, clinically-relevant PVA subtypes, BSA and DTA, thought to be detrimental to patient health.[8,10,25,41,102] Our data collection system serves as a use case for the development of automated, wireless research-focused data acquisition systems in complex healthcare environments. Our annotation system APL, presents a specialized, yet simple, annotation system to enable reviewers to make fast, accurate, and consistent retrospective categorizations of PVA's and clinical artifacts on retrospective VWD. Finally, we showed that addressing several issues common to ML model development in other fields resulted in excellent sensitivity and specificity for the classification of PVA in mechanically ventilated patients.

To address these issues of collection, aggregation and storage of high-frequency waveform data in the context of mechanical ventilation research, we developed an inexpensive and easy to use system that fully automates the end-to-end workflow for data acquisition, wireless communication, and storage except for a single, intentional human checkpoint to ensure proper data attribution. Our system enables temporally accurate waveform data collection and database storage using an unobtrusive physical presence to avoid influencing patient care or introducing observer bias, and requires limited training by study personnel. Furthermore, instructions are provided to enable researchers to use our platform at other institutions.

For annotating the collected waveform data, we discussed how clinicians were able to rapidly annotate raw ventilator waveform data and output it in a usable format using the Annotation PipeLine (APL). APL presents a specialized, yet simple, annotation system to enable

reviewers to make fast, accurate, and consistent retrospective categorizations of PVA's and clinical artifacts. To diminish potential of inter-rater variability, we developed a system of real-time annotation to allow reviewers to accessibly view breath level metadata, enabling them to make more data-driven judgements. Finally, all annotations were output into CSV format which could be easily consumed by algorithms to generate, train, or validate methods of detecting PVA's while filtering clinical artifacts. With APL our team annotated over 9,000 breaths for use in development of our PVA detection algorithm for breath stacking, and double-trigger.

We elucidated our steps in detail for creating an end-to-end platform for creating an ML classifier in order to highlight methodological principles for creating ML classification models for streaming waveform data in healthcare. This challenge is especially acute in the electronic era of medicine. As healthcare increasingly digitizes, high volume, streaming waveform analytic data is being made available to providers from a variety of sources. However, this wealth of data is often coupled with a poverty of information, where data are often used to generate alarms with such low specificity that providers are burdened by incessant false positive alarms that may seriously limit the clinical utility of classification algorithms.[96-98,101] Over time, these false positive alerts can cause provider alarm fatigue that makes it more likely that life-threatening physiologic changes in patients go unnoticed.[101] Similarly in MV, automated, ventilator-based alert systems that notify providers of abnormal breathing patterns often rely on simple thresholding and are subject to numerous false positive alerts. Creation of an ML classifier with high sensitivity and specificity for PVA detection can serve as an example, and basis for how to robustly create algorithms that are both clinically useful and do not contribute to alarm fatigue. Most importantly, for the case of mechanical ventilation, accurate PVA alarms could potentially

improve patient experience by decreasing discomfort, usage of sedatives, ventilator induced lung injury, and potentially even chance of death.[10,28]

Towards solving these problems in ML, we first addressed the class imbalance problem that resulted from PVA breaths being outnumbered by non-PVA breaths by using SMOTE to equilibrate the number of DTA, BSA, and non-PVA breaths. This class imbalance problem stands to potentially impact model development for use cases with similar event prevalence. In this regard, using SMOTE substantially improved our DTA detection performance. Second, we demonstrated that the combined use of expert feature selection and feature selection using statistical tools resulted in superior model performance compared to what either feature selection mechanism could accomplish alone. Finally, we demonstrated improved performance of our classifier by creating an ensemble classifier utilizing multiple different ML algorithms to generate a model with higher sensitivity and specificity than any single algorithm.

Use of SMOTE and ensemble methods may be particularly helpful for clinicians in the future. This thesis highlights that mechanical waveform data is highly imbalanced, with patients taking in range of 20,000-40,000 breaths over a given day, where the vast majority of waveform data is indicative of normal, synchronous breathing. When PVA does occur it can be used as the basis for a classification model, but spurts of PVA can be brief, and difficult to find.[30] SMOTE helps alleviate data paucity by creating synthetic samples of a minority class based on existing operations, and while it is not the only method to deal with class imbalance, is well represented in literature and has proven itself valuable in areas that also deal with data imbalances such as security and networking research.[51,52] Ensemble models can be useful as well for data that is noisy,[49] has highly complex decision boundaries,[103]and have been shown to create more accurate classifiers by combining many weaker classifiers.[104]In our use case, our ensemble model yields

sensitivity and specificity improvements for DTA classification compared the MLP classifier, while maintaining equivalent performance for BSA classification compared to a single GBC classifier.

There are several limitations with the platform we have discussed. Due to variance in network topology and security across institutions there may be technical issues that must be addressed before implementation of data collection. For example, our devices currently transfer data on the hospital clinical wireless network, but this might not be possible in cases where wireless signal availability is poor, or institutional barriers prohibit research devices on facility networks. In this latter case, a separate, free-standing wireless local area network may be required for implementation.[33]Furthermore, our data collection system is meant to be operated by users with minimal technical training, but end-to-end system setup remains complex, and needs to be performed with assistance from IT professionals with systems administration experience.

The work we have accomplished with our annotation software currently means we have a fully functional proof of concept, but additional work is required to make it into a truly generalizable VWD annotation package. To increase the speed of APL, further improvements would be reducing the already relatively low computational burden on-screen annotation requires. Another improvement would be adding additional PVA's and artifacts for annotation like failed trigger and ventilator disconnect. APL is currently only available for use using Puritan Bennet 840^{TM} data. We hope that by open sourcing APL, the wider community can assist in further development and increase the number of mechanical ventilators supported by APL.

For our ML classifier, the development of an optimized model depended on a mix of statistical and expert-derived approaches to feature selection that still requires substantial expert knowledge and it is possible that inclusion and selection of other clinical metadata in model

development may have resulted in a final model with different performance characteristics. Similarly, our selection of ML algorithms was not exhaustive and use of different algorithms or a more exhaustive hyperparameter search may have similarly affected model performance. Indeed, given the number of combinations of features and algorithms used, a thorough hyperparameter search may have proved overly time consuming. Furthermore, we must note the performance of BSA detection in our multi-class model was lower than performance in the binary BSA detection model. This is because of the strict classification boundaries that our clinicians used to differentiate DTA and BSA were not always learned correctly by our model in each kfold. Future work will examine trying to improve this classification boundary, where strict E-time and TVe/TVi requirements are necessary for correct DTA classification.

In summary, we have created an end-to-end platform capable of collecting ventilator data, annotating it, and then classifying it as either a patient injurious PVA subtype, or non-PVA event using machine learning. Our data collection architecture offers research teams a new avenue for accessing VWD data, and serves as a generalizable model for the ongoing development of systems to acquire, transmit, and store other patient data streams for secondary use. Our annotation software offers a new avenue for clinicians to quickly annotate PVA and other clinical artifact in VWD using a combination of visual inspection and mathematic analysis. Finally, our ensemble machine learning classifier can detect two types of clinically-relevant PVA- DTA, and BSA- with high levels of sensitivity and specificity in 35 patients receiving mechanical ventilation in the ICU. Our classifier's high sensitivity and specificity suggests that ML-based models may translate effectively to future PVA detection algorithms to improve the quality of care and clinical experience for patients receiving mechanical ventilation. Finally, our model and methodology may serve as a generalizable framework to guide future researchers in

the use of data analytics and ML to automate the classification of clinical events in patient-derived waveform data.

References

1. Halpern NA. Innovative designs for the smart ICU: part 1: from initial thoughts to occupancy. *Chest*. 2014;145(2):399-403. doi:10.1378/chest.13-0003.

2. Raghupathi W, Raghupathi V. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems 2014 2:1*. 2014;2(1):3. doi:10.1186/2047-2501-2-3.

3. Murias G, Lucangelo U, Blanch L. Patient-ventilator asynchrony. *Curr Opin Crit Care*. 2016;22(1):53-59. doi:10.1097/MCC.0000000000000270.

4. Epstein SK. How Often Does Patient-Ventilator Asynchrony Occur and What Are the Consequences? *Respiratory Care*. 2011;56(1):25-38. doi:10.4187/respcare.01009.

5. Auffray C, Balling R, Barroso I, et al. Making sense of big data in health research: Towards an EU action plan. *Genome Medicine 2016 8:1*. 2016;8(1):71. doi:10.1186/s13073-016-0323-y.

6. Murdoch TB, Detsky AS. The Inevitable Application of Big Data to Health Care. *JAMA*. 2013;309(13):1351-1352. doi:10.1001/jama.2013.393.

7. Thille AW, Rodriguez P, Cabello B, Lellouche F, Brochard L. Patient-ventilator asynchrony during assisted mechanical ventilation. *Intensive Care Med*. 2006;32(10):1515-1522. doi:10.1007/s00134-006-0301-8.

8. Chanques G, Kress JP, Pohlman A, et al. Impact of Ventilator Adjustment and Sedation–Analgesia Practices on Severe Asynchrony in Patients Ventilated in Assist-Control Mode*. *Critical Care Medicine*. 2013;41(9):2177-2187. doi:10.1097/CCM.0b013e31828c2d7a.

9. de Wit M, Pedram S, Best AM, Epstein SK. Observational study of patient-ventilator asynchrony and relationship to sedation level. *Journal of Critical Care*. 2009;24(1):74-80. doi:10.1016/j.jcrc.2008.08.011.

10. Blanch L, Villagra A, Sales B, et al. Asynchronies during mechanical ventilation are associated with mortality. *Intensive Care Med*. 2015;41(4):633-641. doi:10.1007/s00134-015-3692-6.

11. Papazian L, Forel J-M, Gacouin A, et al. Neuromuscular Blockers in Early Acute Respiratory Distress Syndrome. *http://dxdoiorg/101056/NEJMoa1005372*. 2010;363(12):1107-1116. doi:10.1056/NEJMoa1005372.

12. PARKER JC, HERNANDEZ LA, PEEVY KJ. Mechanisms of ventilator-induced lung injury. *Critical Care Medicine*. 1993;21(1):131.

13. DREYFUSS D, SAUMON G. Ventilator-induced Lung Injury. *American Journal of Respiratory and Critical Care Medicine*. December 2012. doi:10.1164/ajrccm.157.1.9604014.

14. Sundar S, Novack V, Jervis K. Influence of low tidal volume ventilation on time to extubation in cardiac surgical patients. *The Journal of …*. 2011.

15. Schmidt M, Banzett RB, Raux M, et al. Unrecognized suffering in the ICU: addressing dyspnea in mechanically ventilated patients. *Intensive Care Med*. 2014;40(1):1-10. doi:10.1007/s00134-013-3117-3.

16. Adams J, Lieng M, Kuhn B, et al. Automated Mechanical Ventilator Waveform Analysis of Patient-Ventilator Asynchrony. *Chest*. 2015;148(4):175A. doi:10.1378/chest.2281731.

17. Blanch L, Sales B, Montanya J, et al. Validation of the Better Care® system to detect ineffective efforts during expiration in mechanically ventilated patients: a pilot study. *Intensive Care Med*. 2012;38(5):772-780. doi:10.1007/s00134-012-2493-4.

18. Yeong Shiong Chiew, Pretty CG, Beatson A, et al. Automated logging of inspiratory and expiratory non-synchronized breathing (ALIEN) for mechanical ventilation. In: IEEE; 2015:5315-5318. doi:10.1109/EMBC.2015.7319591.

19. Kacmarek RM. The Mechanical Ventilator: Past, Present, and Future. *Respiratory Care*. 2011;56(8):1170-1180. doi:10.4187/respcare.01420.

20. Oliveira S, Portela CF, Santos MF. Pervasive universal gateway for medical devices. *Recent Advances in Electrical Engineering and Education Technologies (SCI 2014)*. 2014:205-210.

21. Mulqueeny Q, Ceriana P, Carlucci A, Fanfulla F, Delmastro M, Nava S. Automatic detection of ineffective triggering and double triggering during mechanical ventilation. *Intensive Care Med*. 2007;33(11):2014-2018. doi:10.1007/s00134-007-0767-z.

22. Szlavecz A, Chiew YS, Redmond D, et al. The Clinical Utilisation of Respiratory Elastance Software (CURE Soft): a bedside software for real-time respiratory mechanics monitoring and mechanical ventilation management. *BioMedical Engineering OnLine 2014 13:1*. 2014;13(1):140. doi:10.1186/1475-925X-13-140.

23. Pohlman MC, McCallister KE, Schweickert WD, et al. Excessive tidal volume from breath stacking during lung-protective ventilation for acute lung injury*. *Critical Care Medicine*. 2008;36(11):3019-3023. doi:10.1097/CCM.0b013e31818b308b.

24. Plourde J, Arney D, Goldman JM. OpenICE: An open, interoperable platform for medical cyber-physical systems. In: IEEE; 2014:221-221.

doi:10.1109/ICCPS.2014.6843734.

25. Beitler JR, Sands SA, Loring SH, et al. Quantifying unintended exposure to high tidal volumes from breath stacking dyssynchrony in ARDS: the BREATHE criteria. *Intensive Care Med*. 2016;42(9):1427-1436. doi:10.1007/s00134-016-4423-3.

26. Slutsky AS. Mechanical ventilation. American College of Chest Physicians' Consensus Conference. *CHEST Journal*. 1993.

27. Leatherman J. Mechanical Ventilation for Severe Asthma. *Chest*. 2015;147(6):1671-1680. doi:10.1378/chest.14-1733.

28. Gilstrap D, MacIntyre N. Patient–Ventilator Interactions. Implications for Clinical Management. *American Journal of Respiratory and Critical Care Medicine*. 2013;188(9):1058-1068. doi:10.1164/rccm.201212-2214CI.

29. Nilsestuen JO, Hargett KD. Using Ventilator Graphics to Identify Patient-Ventilator Asynchrony. *Respiratory Care*. 2005;50(2):202-234.

30. Adams JY, Lieng MK, Kuhn BT, Rehm GB. Development And Validation Of A Multi-Algorithm Analytic Platform (ventMAP) For The Automated Detection Of Off-Target Mechanical Ventilation. *A25 CRITICAL CARE …*. 2017.

31. Masry El A, Williams PF, Chipman DW, Kratohvil JP, Kacmarek RM. The impact of closed endotracheal suctioning systems on mechanical ventilator performance. *Respiratory Care*. 2005;50(3):345-353.

32. Colombo D, Cammarota G, Alemani M, et al. Efficacy of ventilator waveforms observation in detecting patient-ventilator asynchrony. *Critical Care Medicine*. 2011;39(11):2452-2457. doi:10.1097/CCM.0b013e318225753c.

33. Howard WR. Wireless on-demand and networking of Puritan Bennett 840 ventilators for direct data capture. *Respiratory Care*. 2007;52(11):1530-1541.

34. Prasad S, Mahalakshmi P. Smart Surveillance Monitoring System Using Raspberry PI and PIR Sensor. *Int J Comput Sci …*. 2014.

35. Milenkovic AM, Markovic IM, Jankovic DS, Rajkovic PJ. Using of Raspberry Pi for data acquisition from biochemical analyzers. In: IEEE; 2013:389-392. doi:10.1109/TELSKS.2013.6704405.

36. Dudas R, VandenBussche C, Baras A, Ali SZ, Olson MT. Inexpensive telecytology solutions that use the Raspberry Pi and the iPhone. *Journal of the American Society of Cytopathology*. 2014;3(1):49-55.

37. Gupta MSD, Patchava V, Menezes V. Healthcare based on IoT using Raspberry Pi. In: IEEE; 2015:796-799. doi:10.1109/ICGCIoT.2015.7380571.

38. Jassas MS, Qasem AA, Mahmoud QH. A smart system connecting e-health sensors and the cloud. In: IEEE; 2015:712-716. doi:10.1109/CCECE.2015.7129362.

39. Kim J, Snodgrass M, Pietrowicz M. BEDA: Visual analytics for behavioral and physiological data. ... *Visual Analytics in* ..... 2013.

40. Fouse A, Weibel N, Hutchins E, Hollan JD. *ChronoViz: a System for Supporting Navigation of Time-Coded Data*. ACM; 2011:299-304. doi:10.1145/1979742.1979706.

41. Mulqueeny Q, Ceriana P, Carlucci A, Fanfulla F, Delmastro M, Nava S. Automatic detection of ineffective triggering and double triggering during mechanical ventilation. *Intensive Care Med*. 2007;33(11):2014-2018. doi:10.1007/s00134-007-0767-z.

42. Kim J, Shin HS, Shin K, Lee M. Robust algorithm for arrhythmia classification in ECG using extreme learning machine. *BioMedical Engineering OnLine 2014 13:1*. 2009;8(1):31. doi:10.1186/1475-925X-8-31.

43. Zhanpeng Jin, Yuwen Sun, Cheng AC. Predicting cardiovascular disease from real-time electrocardiographic monitoring: An adaptive machine learning approach on a cell phone. In: IEEE; 2009:6889-6892. doi:10.1109/IEMBS.2009.5333610.

44. Kourou K, Exarchos TP, Exarchos KP, Karamouzis MV, Fotiadis DI. Machine learning applications in cancer prognosis and prediction. *Comput Struct Biotechnol J*. 2015;13:8-17. doi:10.1016/j.csbj.2014.11.005.

45. Li Q, Clifford GD. Dynamic time warping and machine learning for signal quality assessment of pulsatile signals. *Physiol Meas*. 2012;33(9):1491-1501. doi:10.1088/0967-3334/33/9/1491.

46. Convertino VA, Moulton SL, Grudic GZ, et al. Use of Advanced Machine-Learning Techniques for Noninvasive Monitoring of Hemorrhage. *Journal of Trauma and Acute Care Surgery*. 2011;71(1):S25-S32. doi:10.1097/TA.0b013e3182211601.

47. Blinova VG, Dobrynin DA, Finn VK, Kuznetsov SO, Pankratova ES. Toxicology analysis by means of the JSM-method. *Bioinformatics*. 2003;19(10):1201-1207. doi:10.1093/bioinformatics/btg096.

48. Fernández-Delgado M, Cernadas E, Barro S. Do we need hundreds of classifiers to solve real world classification problems. *J Mach Learn Res*. 2014.

49. Verbaeten S, Van Assche A. Ensemble Methods for Noise Elimination in Classification Problems. In: *Multiple Classifier Systems*. Vol 2709. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, Berlin, Heidelberg; 2003:317-325. doi:10.1007/3-540-44938-8_32.

50. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic Minority

Over-sampling Technique. *Journal of Artificial Intelligence Research*. 2002;16:321-357.

51. Raghuramu A, Pathak PH, Zang H, Han J, Liu C, Chuah C-N. Uncovering the footprints of malicious traffic in wireless/mobile networks. *Computer Communications*. 2016;95:95-107.

52. Crussell J, Stevens R, Chen H. *MAdFraud: Investigating Ad Fraud in Android Applications*. New York, New York, USA: ACM; 2014:123-134. doi:10.1145/2594368.2594391.

53. Jacobs I, Nadkarni V, Bahr J, et al. Cardiac arrest and cardiopulmonary resuscitation outcome reports: update and simplification of the Utstein templates for resuscitation registries. *Resuscitation*. 2004;63(3):233-249. doi:10.1016/j.resuscitation.2004.09.008.

54. Holman CDJ, Bass JA, Rosman DL, et al. A decade of data linkage in Western Australia: strategic design, applications and benefits of the WA data linkage system. *Aust Health Review*. 2008;32(4):766-777. doi:10.1071/AH080766.

55. Ludvigsson JF, Otterblad-Olausson P, Pettersson BU, Ekbom A. The Swedish personal identity number: possibilities and pitfalls in healthcare and medical research. *Eur J Epidemiol*. 2009;24(11):659-667. doi:10.1007/s10654-009-9350-y.

56. Chen LF, Vander Weg MW, Hofmann DA, Reisinger HS. The Hawthorne Effect in Infection Prevention and Epidemiology. *Infection Control & Hospital Epidemiology*. 2015;36(12):1444-1450. doi:10.1017/ice.2015.216.

57. McCambridge J, Witton J, Elbourne DR. Systematic review of the Hawthorne effect: New concepts are needed to study research participation effects. *Journal of Clinical Epidemiology*. 2014;67(3):267-277. doi:10.1016/j.jclinepi.2013.08.015.

58. Mills D, Burbank J, Kasch W. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. (Martin J, ed.). RFC Editor; 2010. doi:10.17487/rfc5905.

59. McKinney W. Data structures for statistical computing in python. In:; 2010.

60. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12(Oct):2825-2830.

61. Hunter JD. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*. 2007;9(3):90-95. doi:10.1109/MCSE.2007.55.

62. Heap M. Advanced Ansible. In: *Ansible*. Berkeley, CA: Apress; 2016:137-157. doi:10.1007/978-1-4842-1659-0_9.

63. Vanderkam D. *Dygraphs Javascript Charting Library*. 2006.

64. Siempos II, Vardakas KZ, Falagas ME. Closed tracheal suction systems for prevention of ventilator-associated pneumonia. *Br J Anaesth*. 2008;100(3):299-306. doi:10.1093/bja/aem403.

65. Al-Khafaji A, Manning HL. Inappropriate ventilator triggering caused by an in-line suction catheter. *Intensive Care Med*. 2002;28(4):515-519. doi:10.1007/s00134-002-1239-0.

66. Bishop CM. Pattern recognition. *Machine Learning*. 2006.

67. Best CH, Taylor NB. *The Physiological Basis of Medical Practice: A University of Toronto Text in Appliewd Physiology*. Williams & Wilkins Company; 1945.

68. Martín-González F, González-Robledo J, Sánchez-Hernández F, Moreno-García MN. Success/Failure Prediction of Noninvasive Mechanical Ventilation in Intensive Care Units. *Methods Inf Med*. 2016;55(3):234-241. doi:10.3414/ME14-01-0015.

69. Guyon I, Elisseeff A. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*. 2003;3(Mar):1157-1182.

70. Blum AL, Langley P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*. 1997;97(1-2):245-271. doi:10.1016/S0004-3702(97)00063-5.

71. Dash M, Liu H. Feature selection for classification. *Intelligent Data Analysis*. 1997;1(1-4):131-156. doi:10.1016/S1088-467X(97)00008-5.

72. Kotsiantis SB. Supervised Machine Learning: a Review of Classification Techniques

. 2007.

73. Sun Y, Chan KL, Krishnan SM. ECG signal conditioning by morphological filtering. *Computers in Biology and Medicine*. 2002;32(6):465-479. doi:10.1016/S0010-4825(02)00034-3.

74. Shoeb AH, Guttag JV. Application of machine learning to epileptic seizure detection. In:; 2010.

75. Jin X, Xu A, Bie R, Guo P. Machine Learning Techniques and Chi-Square Feature Selection for Cancer Classification Using SAGE Gene Expression Profiles. In: *Data Mining for Biomedical Applications*. Vol 3916. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, Berlin, Heidelberg; 2006:106-115. doi:10.1007/11691730_11.

76. Hastie T, Rosset S, Tibshirani R, Zhu J. The Entire Regularization Path for the Support Vector Machine. *Journal of Machine Learning Research*. 2004;5(Oct):1391-1415.

77. Guyon I, Weston J, Barnhill S, Vapnik V. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*. 2002;46(1-3):389-422. doi:10.1023/A:1012487302797.

78. Wold S, Esbensen K, Geladi P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*. 1987;2(1-3):37-52. doi:10.1016/0169-7439(87)80084-9.

79. Mika S, Ratsch G, Weston J, Scholkopf B, Mullers KR. Fisher discriminant analysis with kernels. In: IEEE; 1999:41-48. doi:10.1109/NNSP.1999.788121.

80. Lee T-W. Independent Component Analysis. In: *Independent Component Analysis*. Boston, MA: Springer, Boston, MA; 1998:27-66. doi:10.1007/978-1-4757-2851-4_2.

81. Chao DC, Scheinhorn DJ, Stearn-Hassenpflug M. Patient-Ventilator Trigger Asynchrony in Prolonged Mechanical Ventilation. *Chest*. 1997;112(6):1592-1599. doi:10.1378/chest.112.6.1592.

82. Kubat M, Holte RC, Matwin S. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*. 1998;30(2-3):195-215. doi:10.1023/A:1007452223027.

83. Wang S, Minku LL, Yao X. A Systematic Study of Online Class Imbalance Learning with Concept Drift. March 2017.

84. Japkowicz N. The class imbalance problem: Significance and strategies. *Proc of the Int'l Conf on Artificial …*. 2000.

85. Friedman JH. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery*. 1997;1(1):55-77. doi:10.1023/A:1009778005914.

86. Michie D, Spiegelhalter DJ, Taylor CC. Machine Learning, Neural and Statistical Classification. 1994.

87. Friedman JH. Greedy Function Approximation: A Gradient Boosting Machine on JSTOR. *Annals of statistics*. 2001. doi:10.2307/2699986.

88. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees*. Monterey: Wadsworth & Brooks; 1984.

89. Ross Quinlan J, Rivest RL. Inferring decision trees using the minimum description lenght principle. *Information and Computation*. 1989;80(3):227-248. doi:10.1016/0890-5401(89)90010-2.

90. LeCun YA, Bottou L, Orr GB, Müller K-R. Efficient BackProp. In: *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg; 1998:9-48.

doi:10.1007/978-3-642-35289-8_3.

91. Rubinstein R. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*. 1999;1(2):127-190. doi:10.1023/A:1010091220143.

92. Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. *Machine Learning*. 2006;63(1):3-42. doi:10.1007/s10994-006-6226-1.

93. Keijzer M, Babovic V. Genetic Programming, Ensemble Methods and the Bias/Variance Tradeoff – Introductory Investigations. In: *Genetic Programming*. Springer, Berlin, Heidelberg; 2000:76-90. doi:10.1007/978-3-540-46239-2_6.

94. Kriegeskorte N, Simmons WK, Bellgowan PSF, Baker CI. Circular analysis in systems neuroscience: the dangers of double dipping. *Nature Neuroscience*. 2009;12(5):535-540. doi:10.1038/nn.2303.

95. Zheng ZS, Reggente N, Lutkenhoff E, Owen AM, Monti MM. Disentangling disorders of consciousness: Insights from diffusion tensor imaging and machine learning. *Human Brain Mapping*. 2017;38(1):431-443. doi:10.1002/hbm.23370.

96. Blum JM, Kruger GH, Sanders KL, Gutierrez J, Rosenberg AL. Specificity Improvement for Network Distributed Physiologic Alarms Based on a Simple Deterministic Reactive Intelligent Agent in the Critical Care Environment. *J Clin Monit Comput*. 2009;23(1):21-30. doi:10.1007/s10877-008-9159-3.

97. Graham KC, Cvach M. Monitor Alarm Fatigue: Standardizing Use of Physiological Monitors and Decreasing Nuisance Alarms. *Am J Crit Care*. 2010;19(1):28-34. doi:10.4037/ajcc2010651.

98. Phillips J, Barnsteiner JH. Clinical Alarms: Improving Efficiency and Effectiveness. *Critical Care Nursing Quarterly*. 2005;28(4):317.

99. Imhoff M, Kuhls S. Alarm Algorithms in Critical Care Monitoring. *Anesthesia & Analgesia*. 2006;102(5):1525-1537. doi:10.1213/01.ane.0000204385.01983.61.

100. Lalkhen AG, McCluskey A. Clinical tests: sensitivity and specificity. *Contin Educ Anaesth Crit Care Pain*. 2008;8(6):221-223. doi:10.1093/bjaceaccp/mkn041.

101. Drew BJ, Harris P, Zègre-Hemsey JK, et al. Insights into the Problem of Alarm Fatigue with Physiologic Monitor Devices: A Comprehensive Observational Study of Consecutive Intensive Care Unit Patients. Tereshchenko LG, ed. *PLOS ONE*. 2014;9(10):e110274. doi:10.1371/journal.pone.0110274.

102. Beitler JR, Thompson BT, Matthay MA, et al. Estimating Dead-Space Fraction for Secondary Analyses of Acute Respiratory Distress Syndrome Clinical Trials. *Critical Care Medicine*. 2015;43(5):1026-1035. doi:10.1097/CCM.0000000000000921.

103. Polikar R. Ensemble based systems in decision making. *IEEE Circuits Syst Mag*. 2006;6(3):21-45. doi:10.1109/MCAS.2006.1688199.

104. Dietterich TG. Ensemble methods in machine learning. *Multiple classifier systems*. 2000.

Appendix A

The UCD VWD infrastructure is designed to allow anyone to use off the shelf commercial products to set up a fully functional research data acquisition system for collecting mechanical ventilator waveform data.

In this README, installation and usage instructions are provided. It is important to note that experienced, technical users should set the system up, but daily operation can be performed by almost anyone, regardless of technical skill.

A.1 Hardware Prerequisites

The following components will be necessary before attempting to install the UCD VWD infrastructure:

1. A Raspberry Pi device and power cable.
2. A DB-9 to USB serial null modem cable
3. An RS-232 optical isolator (some sites may not require an optical isolator)
4. A linux/OSX computer for the user's local machine. If Windows absolutely desired, then it must have Cygwin. Note that if Windows is used, ansible is not explicitly supported on Windows and may require an alternative solution (see sections on ansible later in this document).
5. A server to run the clinical supervisor application (CSA). As a note: CSA setup does not need to be performed immediately, and data can be collected without it. However, at scale the CSA is critical to ease of operation and ensuring proper data attribution.

A.2 PB-840 Ventilator Setup

The PB-840 ventilator will need to undergo several steps to enable data output from the RS-232 serial port.

On the PB-840 GUI, on the lower screen, click the icon on the far right side of the screen (Other Screens Icon). Go to the Communication Settings, select “Waveforms” for port 3, and change the configuration settings to the following values:

1. baud rate: 38400
2. data bits: 8
3. parity mode: 0

A.3 Installing dependencies

Once hardware dependencies are satisfied, and the ventilator is set up, the user will need to begin installing software on their local machine via the command line to set up the RPi devices. First, the user needs to install virtualenv. Using pip this can be done via:

```
pip install virtualenv
```

Then create a new virtual environment and activate it:

```
virtualenv venv
source venv/bin/activate
```

Next navigate to the root directory where `setup.py` is located. Run the following command:

```
pip install –e .
python setup.py develop
```

Now all dependent packages should be installed

A.4 Raspberry Pi Setup

Via image flashing

The first step in image flashing is to get an SD card reader/writer device. Hook it up to your computer and then insert the raspberry pi's SD card into the device.

You can obtain the disk image here.

From Debian

From a debian based machine download the image then run the following command:

```
dd if=ucdpv-jessie.img bs=4M of=/dev/sdb
```

After this step, remove the SD card from the device and insert it into the raspberry pi to boot.

From OSX

Setup is relatively the same as in Debian based machines here except the `of` input will be different:

```
dd if=ucdpv-jessie.img bs=4M of=/dev/rdisk2s2
```

Final configuration

Network Operations and NTP Setup

It is important to note that without a connection to an NTP server, ventilator waveform data collection will **NOT** commence. At UCD, IT Network Operations necessitated the RPi devices be located behind a secure firewall. If this is not pertinent to your institution, then this section is unnecessary. If this is the case at your institution, then internet access to public NTP servers may not be available.

If there are network restrictions present on the wifi network then specific NTP servers will need to be used. Go to `/etc/ntp.conf` and modify the file so that the NTP server addresses, specific to your institution, are used. Your local IT Network Operations professional can assist with this step. Configuration should look like:

```
server <host1 ip addr> iburst
server <host2 ip addr> iburst
```

Wifi

The wifi network being used will need to be modified in `/etc/wpa_supplicant/wpa_supplicant.conf`. Your local IT Network Operations professional can assist with this step.
Modify the file like so:

```
network={
    ssid=<WiFi SSID>
    psk=<WiFi password>
}
```

Then, on the command line, restart the networking service to gain wifi connectivity:

```
sudo service network restart
```

Ethernet

If using an Ethernet connection for the raspberry pi, no additional configuration is required here other than hooking an ethernet cable to the raspberry pi.

Via Ansible

If you have already setup your raspberry pi using NOOBS or some other tool and want to modify your raspberry pi to collect ventilator data then setup via ansible may be best (this assumes that you have elected not to use the disk image flashing methods detailed above).

Pre-setup

The requirements here are that a wireless network is set up and that the internal network address of this network is known. If the internal network address is not

known then you can type in the command:

```
ifconfig
```

The network address should look something similiar to `192.168.1.10`. Here take the first 3 numbers of this address and open the file `raspi/__init__.py`

```
nano raspi/__init__.py
```

Now enter the first 3 numbers in the address into the `lan_prefix` variable name. For example, if your address was `192.168.1.10` you would enter `lan_prefix = "192.168.1"`.

Raspberry Pi minimal setup

Since Ansible was installed in previous steps and the provisioning LAN is now setup, plug an ethernet cable from the router into the raspberry pi. Navigate to the `raspi/ansible` directory

```
cd raspi/ansible
```

and modify the file at `group_vars/prod`.
Here enter the production network's wireless SSID and password. Also enter the network's ntp server host ip addresses (your local IT Network Operations professional can assist with these steps). Finally, installation can proceed. To install all necessary software:

```
ansible-playbook -u pi --ask-sudo -i inventory/rpi_initial rpi_minimal_install.yml
```

When prompted for the password enter "raspberry" per standard setup. After ansible finishes, your raspberry pi will be able to collect ventilator waveform data!

Raspberry Pi Initial setup

Since Ansible was installed in previous steps and the provisioning LAN is now setup, plug an ethernet cable from the router into the raspberry pi. Navigate to the `raspi/ansible` directory

```
cd raspi/ansible
```

and modify the file at `group_vars/rpis`.
Here enter the production network's wireless SSID and password. Also enter the network's ntp server host ip addresses (your local IT Network Operations professional can assist with these steps). Finally, installation can proceed. To install all necessary software:

```
ansible-playbook -u pi --ask-sudo -i inventory/rpi_initial rpi_pre_hostname_install.yml
```

When prompted for the password enter "raspberry" per standard setup. Now your system will be minimally set up to extract ventilator data. However, the device will

not be able to interact with the CSA yet. Next steps involve setting up the CSA, and networking your Raspberry Pis so that they receive DNS addresses.

Raspberry Pi Final Setup

After the CSA is set up, the Raspberry Pis have DNS addresses, and configuration details for the `retriever` user are finished (see CSA setup below), you can complete the setup of the Raspberry Pi's. This will change all hostnames on the Raspberry Pis in accordance with your DNS profile, and will set up SSH links between the CSA and the Raspberry Pis:

```
ansible-playbook -u pi --ask-sudo -i inventory/<inventory file> rpi_install.yml
```

A.5 Clinicalsupervisor (CSA) Setup

Activate your virtual environment (see above), and navigate to `raspi/ansible`. Create a new inventory script using `inventory/ucd` as an example. Input the host DNS name under the `[clinicalsupervisor]` group (your local IT Network Operations professional can assist with these steps).

If the database plugin is desired for use, then the installer will need to modify the file `group_vars/clinicalsupervisor` and the variables `database_host` and `database_password`. To ensure these variables stay secret, it is highly recommended to use `ansible-vault` to encrypt this file so that secure information is not gained by unauthorized parties. Information on how to use `ansible-vault` is located here.

Static DNS

The CSA is able to communicate with the raspberry pis through either regular or static DNS. If for some reason the raspberry pis are unable to be listed in an institution DNS server or DNS is spotty then static DNS will need to be used. Static DNS addresses can be provided to the CSA through the `group_vars/clinicalsupervisor` file. If you are using static DNS but a raspberry pi is not listed, then regular DNS will be used. Static addresses can be added like this:

```
static_dns:
   hostname1: 192.168.1.5
   hostname2: 192.168.1.6
   ...
```

Restricting SSH Commands

The CSA should only communicate with the Raspberry Pis via a limited

set of commands. As a result, 3 new users are set up on the clinicalsupervisor and the Raspberry Pis: `backup`, `cleaner`, and `lister`.

`backup` is meant only to backup all data in the `/home/pi/Data` directory. `cleaner` is meant to perform deletions of files when necessary, and `lister` is designed to only list the files in the data directory.

As the sysadmin, you will need to create a passwordless, public/private keypair for the `retriever` user on the clinicalsupervisor host. You can do this via terminal

```
ssh-keygen
```

Then, take the public key for this key pair and paste it into `group_vars/rpis` for the `retriever_pub_key` variable. After pasting the public key it should look like this:

```
retriever_pub_key: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQChlZwB1+fMhvqVfP2ZV1pH8kH9rwpTYsBb
cvCeLZ/cfeScn91RI3M9eWhdTOWD1O4T5FhgWkCjaVbDqDjKFmknSGXa9NaIicMX8fSUZ
7Kda0PvfJBwwtewgS8uzhuxgXG2ltflh11W6c0c1sNI2XaGEZ7LlAE3bbkzP1PWvWCtqC8+s4
ZeSNDFE2K0GCJmckbb0xw4CNFoVHj10kCdD1z/vGCV1YKKmUn7WRYL2Rcpw7HIOlprz
HPhSgg2rda8GvqN0N8C9pbY+XMLiG0bU+iD8dgtg0h5gBBKNmicHp+SQQdtjlZcBtLYDDh
TRI5tAKvpHSFqc8PK+n1WAaMWeY3x retriever@foo
```

After performing this action you will need to re-update your Raspberry Pi devices with this public key.

OSX

```
ansible-playbook -i inventory/<inv file name> clinicalsupervisor_install_osx.yml
```

Debian

```
ansible-playbook -i inventory/<inv file name> clinicalsupervisor_install_debian.yml
```

A.6 UCD VWD System Usage

Raspberry Pi

To use the raspberry pis:

1. Get a micro-usb power cable, USB-to-RS232 serial cable, and optical isolator attachement if this device is necessary
2. Hook optical isolator attachment to primary serial port of PB-840 (if necessary)
3. Take the raspberry pi to the ventilator and hook the DB-9 cable to serial port 3 of the PB-840 ventilator and the USB side to the RPi.

4. Power the RPi with the power cable,
5. Ensure the ventilator is currently operating and actively displaying waveform data.

Once the ventilator is operating, the RPi will attempt to connect to NTP. Once connected to NTP, waveform data collection will begin.

Clinicalsupervisor (CSA)

There are several pieces of functionality that the CSA utilizes to perform its job.

1. Listing files
2. Backing up files
3. Deleting files
4. Attributing VWD to the correct patient

In each case the CSA asks for input of the raspberry pi name before continuing. The raspberry pi name is the DNS name for the device on the network (e.g. rpi16). If this has not been created but the raspberry pis are connected to the network and have a static ip address then a static DNS service hosted by the CSA can be used. For details about setting this up, go to the section on `Static DNS` in the installation guide.

Listing files

If the user desires to know the filenames on a raspberry pi then the `List` button on the top nav bar will allow this.

Backing up files

This will perform a backup of all files currently on the raspberry pi on the system server. Use the `Backups` button on the nav bar for this.

Deleting files

This should only be done if the files in question have been backed up or belong to a patient not qualified for study enrollment. If this action is completed, ALL mechanical ventilator waveform files on the raspberry pi will be deleted. Go to `Full Clean` for this.

Attributing VWD to the Correct Patient

If you desire to collect specific mechanical ventilator waveform data files and store them properly attributed to a particular patient on the system server, then click the `Enroll` button. First you will be asked to enter the raspberry pi name; after doing this, all files on the pi will be backed up. Then you will be asked to select all files belonging to a specific patient. After selecting the correct files, input a patient unique identifier (such as a study pseudo-ID) into the field at the bottom of the UI screen. After inputting a unique identifier and selecting submit, all selected file names will be appended with the unique patient identifier, moved to a unique identifier-specific folder on the system server, and selected files on the raspberry pi will be deleted.

A.7 Security

Given you are working in a hospital environment, security will be a concern about any system used. As a result, we have taken steps to harden this system against intrusion. While no security protocol is perfect, the steps outlined, if followed, will ensure your system poses minimal security risk to the broader hospital infrastructure. Note that users should discuss these security measures with institution-specific IT security, IT network operations, research IT, and local institutional review board personnel to ensure that institution-specific security measures are followed.

Raspberry Pi Hardening

User Passwords

It is **highly advised** to change the default user password for the Raspberry Pis. First, generate a password and hash it, via the steps outlined here. Then go to `group_vars/rpis` and modify the `pi_user_password` variable. Since you should not risk exposing a hash publically, encrypt the `group_vars/rpis` file using ansible-vault.

```
ansible-vault encrypt group_vars/rpis
```

Pi Storage

The Raspberry Pis are generally not vulnerable while they are attached to a ventilator in the intensive care unit given the near constant monitoring a patient receives there, but may be vulnerable to exploit in an intermediary storage location if located in a common storage area of the hospital. Together with hospital engineering, network, and security teams you **must** make a **conscious determination** on what is the best storage procedure for the Raspberry Pis and we recommend storing devices in a non-publicly accessible location.

Pi Networking

Raspberry Pi's should not be networked to communicate with anything via outbound connection. It is advised to perform firewalling at the network level to not allow outbound communications from the Raspberry Pis. We recommend working with institutional IT networking personnel to establish appropriate firewall protocols.

Protected Health Information (PHI) Capability

Currently, this system is not configured to be used with PHI data. This would require the modification of the system to either perform:

* resting encryption of all data
* immediate transfer of PHI data to a secure system server location without writing to disk on the raspberry pi

If this capability is desired you are free to modify this software under GPL.

Clinicalsupervisor (CSA) Hardening

Basics

The primary goal here is to ensure that an infection of one of the Raspberry Pi devices does not compromise the clinicalsupervisor where data resides.

1. Ensure the CSA resides on a health-system monitored server in a datacenter. This ensures only privileged users will have access to the computing device, and links the health of your server to the IT capabilities of your institution
2. Ensure that the CSA can only accept incoming communications from approved IP addresses. Do not allow the Raspberry Pis to communicate with the CSA, as there is no reason why they should need this functionality. This whitelisting of IP addresses can either take place on the host, or the network level.

Others

Security is a deep field, and it is difficult to enumerate all the things necessary to properly secure a server. We have chosen to remove some security details for the CSA like SSH configuration because we believe these things are better left in the hands of the user. There may be institution-specific security procedures or configurations that require modification of our existing architecture and so this documentation is meant as a framework to guide local iteration as needed. If there are any other security details you feel that we've missed, pull requests are always welcome :)

A.8 Software

Raspberry Pi

PB-840 get_serial.py

The `get_serial.py` script is the main software that completes the action of pulling data from the PB-840. It begins operation immediately after an NTP server is found and time is accurately updated on the system clock. After this, the software will search for a serial connection and if one is found, data will be pulled from the ventilator and saved to a file. If for some reason there is a break in data collection for longer than 10 seconds (such as when a patient is temporarily disconnected from the ventilator, or during prolonger apneas) the current file will be saved, closed, and a new file will be written once serial data are detected again. This is performed to maintain the temporal accuracy of data if an investigator elects to use a file timestamp-based feed-forward approach to timestamping individual data points within a file.

This software will run continuously in a loop as long as the RPi is turned on. If no data is available to be pulled, then no data will be written to file.

Appendix B

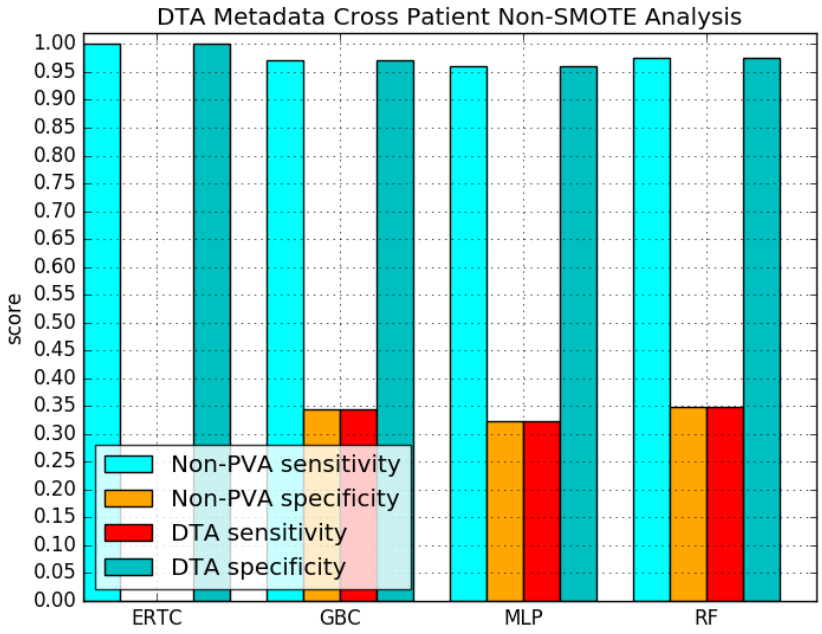

Figure B14: Binary DTA detection using metadata features without SMOTE.

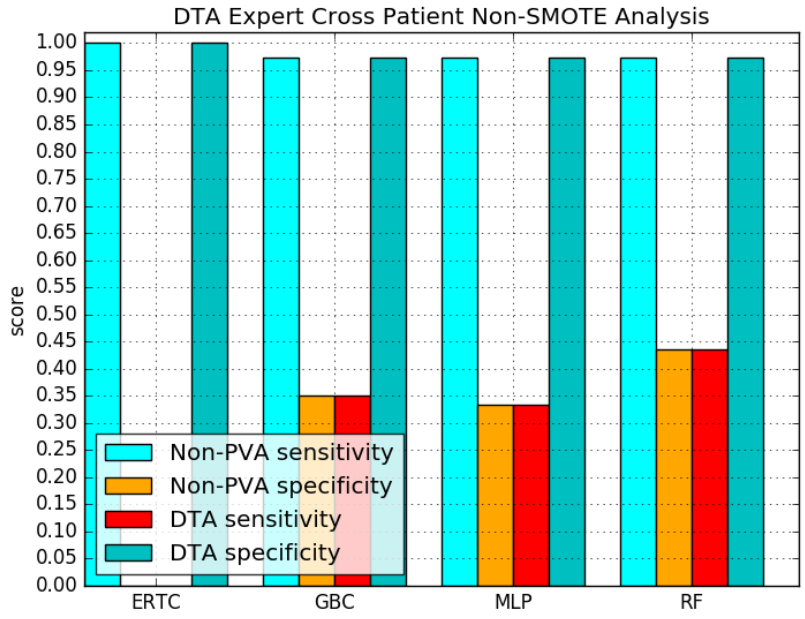

Figure B15: Binary DTA detection using expert feature without SMOTE.

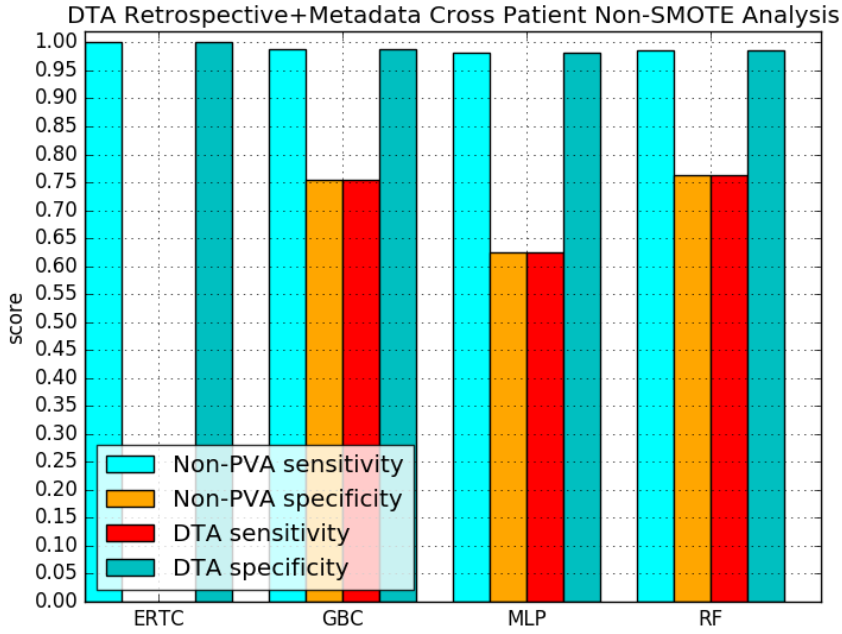

Figure B16: Binary DTA detection using retrospective and metadata features without SMOTE.

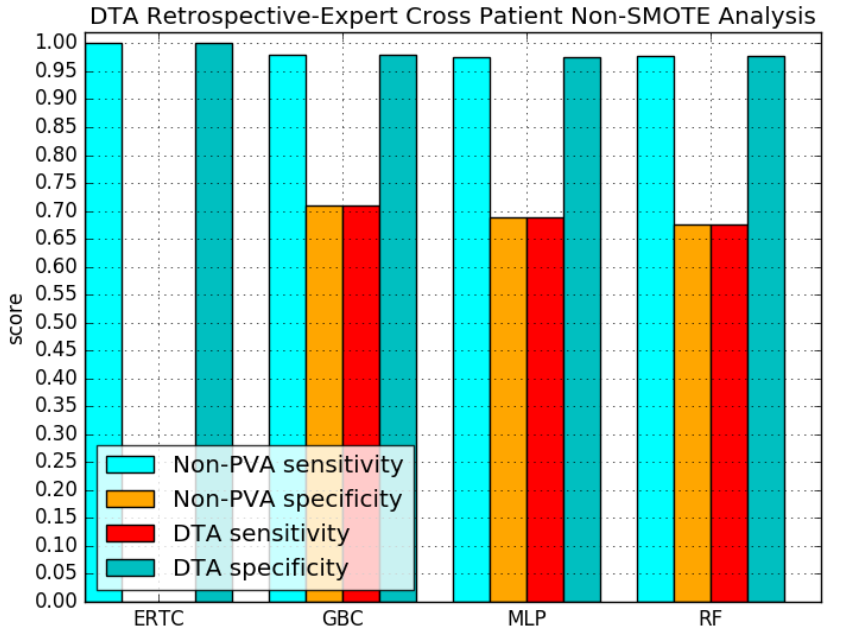

Figure B17: Binary DTA detection with expert retrospective features and run without SMOTE.

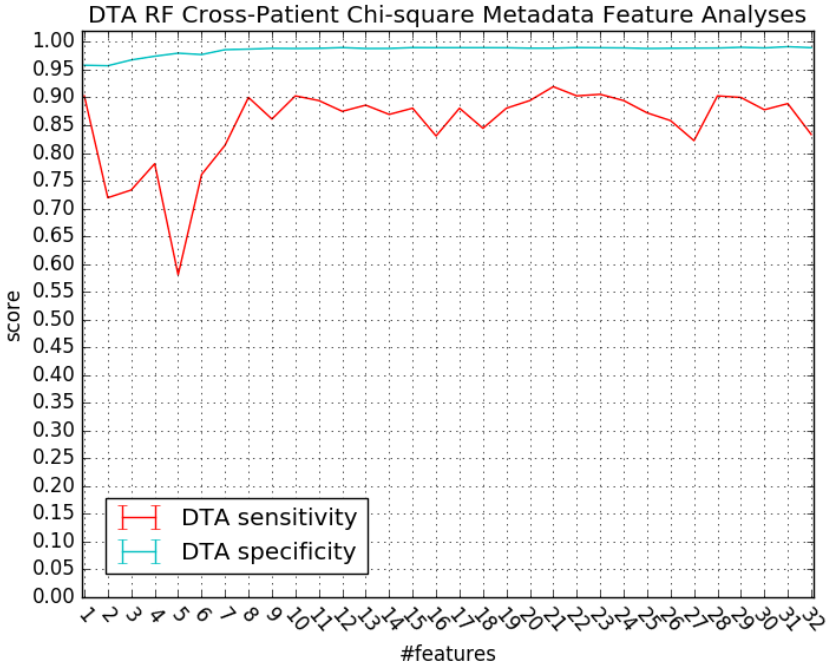

Figure B18: Chi-square sensitivity analysis for binary BSA detection using all retrospective and metadata features. We found 21 features was the optimal number of features here for a subset of DTA features derived from the retrospective and metadata features.

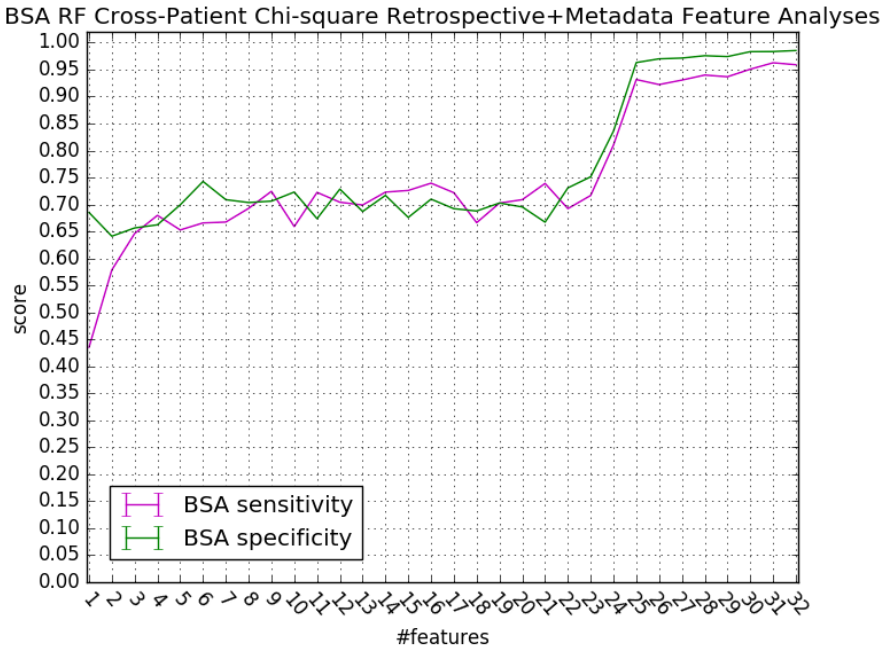

Figure B19: Chi-square sensitivity analysis for binary BSA detection using all retrospective and metadata features. Here we found the optimal number of features was 32, which is all possible features.

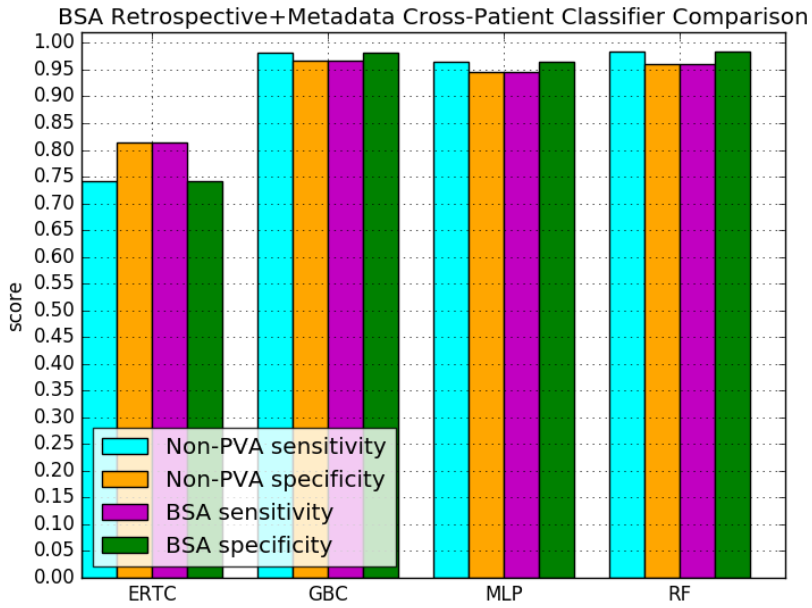

Figure B20: Binary BSA detection using all retrospective and metadata features. From this experiment we found the addition of the retrospective features did not improve our model above baseline performance of using all metadata.

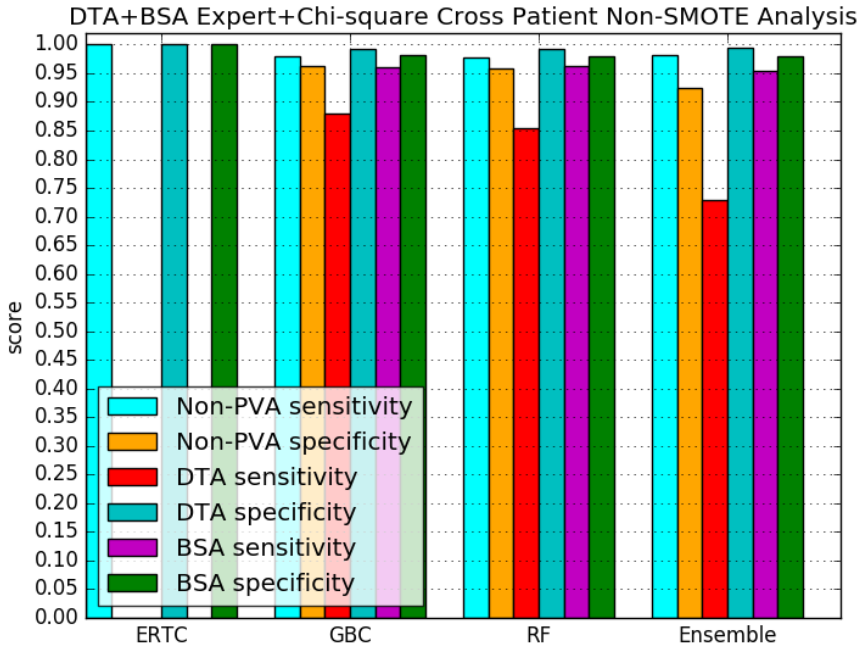

Figure B21: Figure details the results of running our final model without SMOTE. Most classifiers with exception of GBC suffer from poor DTA sensitivity, while BSA is relatively unaffected by lack of SMOTE. SMOTE, synthetic minority over-sampling technique; PVA, patient ventilator asynchrony; BSA, breath stacking asynchrony; DTA, double trigger asynchrony; ERTC, extremely randomized trees classifier; GBC, gradient boosted classifier; MLP, multi-layer perceptron