CSCI 2980 Master's Project Report

Applied Machine Learning to

Healthcare Predictive Analytics

Abhishek Dutta

Department of Computer Science, Brown University

Advisor: Prof. Ugur Cetintemel

Spring (May, 2018)

Abstract

From fall 2017, I've been working with Prof. Ugur Cetintemel in a project about applying machine learning models to improve healthcare. Current medical devices are overly sensitive to noise and variations in time series signals and suffer from high false alarm rates – even simple, normal patient movements and sensor/cable artefacts can trigger high-priority clinical alarms. Thus, their ability to accurately and meaningfully alert providers about urgent, clinically significant situations is limited.

Our goal is to apply computational methods to analyze large and varied clinical datasets in order to generate models that can accurately identify, differentiate, and possibly predict clinically significant alarms directly from clinical datastreams. We use two machine learning models to understand machine generated signals and create a high precision model to correctly predict highly significant alarms of patients.

This report first introduces these two systems, and then discusses the motivation and use cases for both the models. Then it states the details about the decision and implementation, as well as the performance testing of both models. In the last part of the report, I also proposed some observation and works that can be done in the future.

The data was received from the clinical experts at the Rhode Island Hospital. The two semester long research project was divided into two tasks:

1. Machine generated alarms
2. Clinically significant Human Annotated Alarms

Table of Contents

1. Introduction

1.1 SVM and other Machine Learning Models

In machine learning, support vector machines (SVMs) are supervised learning schemes with learning algorithms that parse the data used for classification and other research analysis. Given a set of training and test samples, each labeled as belonging to one class or the other. The SVM training algorithm builds a model that assigns new examples to one class or the other. Hence, a non-probabilistic binary linear classifier is used for this purpose. The model maps all the data points in a high dimensional space and the support vectors try to maximize the margin of separation to identify different class labels. The test samples are then mapped into the same high dimensional space and predicted to fall in a class based on which side of the margin they belong.

When data does not have labels, supervised learning is not possible, and an unsupervised learning mechanism is required. It uses a clustering mechanism to form clusters of the data, and then map new test samples to these clusters. The support vector clustering algorithm categorizes unlabeled data, and is one of the most commonly used approach in real world scenarios.

1.2 LSTMs, CNNs and other Deep Learning Models

There are multiple Deep Learning Models used in this project. In this project, our focus is to use Long/short-term memory networks (LSTMs) and Convolutional neural networks (CNNs).

Long/short-term memory networks (LSTMs) are a special type of RNNs which unlike typical units, do not perform activation, allowing the same signal to move back in the network for an arbitrarily long period of time. This special characteristic of LTSMs distinguishes it from other Deep Learning Models and it aids in representing a time series. It provides a connection between past and present that is sensitive to both past and recent events. The LTSMs are most effective in capturing semantic nuances like in machine translation or for accuracy in probabilistic models. There's a theorem which states that

LTSMs are Turing complete and in principle can represent any computation which is very fascinating to note.

Convolutional neural networks (CNNs) are very good for scenarios in which data is sampled periodically in one or more dimensions, like audio signals in one dimension, and images in two dimensions. The CNNs are defined by convolving a number of layers with an activation function whose output predicts the next layer in the network. Furthermore, the results of this output layer are then aggregated into a layer of fewer cells where each cell averages the results of several units in the previous layer. This aids the next layer to contain fewer cells as the network progresses.

In our project, a hybrid model of LSTMs and CNNs are built to take advantage of both the scenarios. This provides a robust model which is more sophisticated to predict alarms with higher precision.

2. Motivation and Design

2.1 Objective

The objective of this project is broadly categorized in the following three stages:

1. *Study and apply base system alarms (i.e., learn the model used by existing systems)*
2. *Study and learn data associated with high priority alarms (i.e., learn then filter monitor data for true red alarms),*
3. *Study and learn the clinical significance and severity of alarms from clinical correlate data (i.e., learn and associate monitor data with clinical outcomes).*

2.2 Data Description and Format

2.2.a. Original JSON files:

The original JSON data consists of multiple json objects, each json object is an observation of live patient signals at a given timestamp processed by patient bedside monitors and acquired by research "listener" hardware/software devices. Each timestamp is a json object. Based on the data emission characteristics of the patient monitors studied (Philips Intelivue systems; e.g., 1-lead ECG at 250Hz sent out as 64 datapoints / 0.256s, which translates to 256 datapoints / 1.024s = 250Hz), patient physiologic data are exported as .json objects.

Example 1: timestamped waveform data (ECG at 250Hz; Pleth SpO2 at 125Hz)

{"timestamp": "2017-05-26T02:48:20.680000", **"ECG"**: [0.11499999999999488, -
0.10000000000000853, -0.21500000000001052, -0.2400000000000091, -
0.22500000000000853, -0.20000000000000995, -0.16500000000000625, -
0.1250000000000071, -0.07000000000000739, -0.03500000000003695, -
0.015000000000007674, -0.005000000000009663, -7.105427357601002e-15,
0.009999999999990905, 0.014999999999993463, 0.024999999999991473,
0.024999999999991473, 0.024999999999991473, 0.02999999999999403,
0.03999999999999204, 0.03999999999999204, 0.04999999999999005,
0.04999999999999005, 0.05499999999999261, 0.05499999999999261,
0.06499999999999062, 0.06999999999999318, 0.07499999999999574,
0.07999999999999119, 0.0899999999999963, 0.09499999999999176,
0.09999999999999432, 0.10499999999998977, 0.11499999999999488,
0.1249999999999929, 0.1349999999999909, 0.13999999999999346,
0.14999999999999147, 0.15499999999999403, 0.15499999999999403,
0.16499999999999204, 0.1799999999999926, 0.18499999999999517,
0.19499999999999318, 0.2049999999999912, 0.2149999999999963,
0.22499999999999432, 0.24499999999999034, 0.2599999999999909,
0.2749999999999915, 0.2849999999999895, 0.29999999999999005,
0.3199999999999932, 0.34499999999999176, 0.3649999999999949,
0.38999999999999346, 0.4099999999999895, 0.4399999999999906,
0.45999999999999375, 0.47999999999998977, 0.4999999999999929,
0.5099999999999909, 0.529999999999994, 0.549999999999999], "Heart Rate": null,
"Respiration Rate": null, "alarms": null, "Airway": {"Respiration Rate": null, "etCO2": null},
"Pleth": [2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048,
2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048, 2048,
2048, 2048, 2048, 2048, 2048], "SpO2": null, "Non-invasive Blood Pressure": {"mean": null,
"systolic": null, "diastolic": null}, "qos": 0}

Example 2: timestamped alarm and numeric data (approx. 1Hz), nulls for data not present

{"timestamp": "2017-05-26T02:48:21.992000", "ECG": null, "Heart Rate": null, "Respiration
Rate": null, **"alarms"**: {"Alarm_T_0": {"source": "NOM_PULS_OXIM_SAT_O2", "state":
"AL_SILENCED_RESET", "code": "NOM_EVT_SENSOR_DISCONN", "type":
"MED_PRI_T_AL", "string": "SpO"}, "Alarm_P_0": {"source":
"NOM_ECG_CARD_BEAT_RATE", "state": 0, "code": "NOM_EVT_LO", "type":
"LOW_PRI_P_AL", "string": ""}}, "Airway": {"Respiration Rate": null, "etCO2": null}, "Pleth":
null, "SpO2": null, "Non-invasive Blood Pressure": {"mean": null, "systolic": null, "diastolic":
null}, "qos": 0}

2.2.b. Alarm CSV files:

The alarm files contain all the red(high priority) ECG, PPG(SpO2) and NIBP(blood pressure) time-stamped alarms of clinical interest for this research program, extracted with Splunk from the original .json files and including only the alarms of interest to the research program.

Example:

_time	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(alarm	values(duration)
2017-05-25T05:28:27.088-0400												hypoxia		10.24
2017-05-25T05:28:38.352-0400													hypoxia	52.224
2017-05-25T05:29:32.624-0400												hypoxia		5.12
2017-05-25T09:52:54.992-0400					tachycardia									55.296
2017-05-25T09:53:51.312-0400			sev.tachycardia											112.64
2017-05-25T09:58:09.360-0400					tachycardia									0
2017-05-25T09:58:10.384-0400			sev.tachycardia											9.248
2017-05-25T10:00:32.720-0400					tachycardia									21.536

2.3 Splunk Database Migration

Splunk Database Connect is a solution to work with databases from Splunk. It aids in quickly integrating structured data sources with the Splunk real-time machine data collection which is used by the clinical experts to process machine generated signals of patients. This is a very useful tool for doctors and it supports multiple platforms like DB2/Linux, MemSQL, MySQL, AWS Aurora, Microsoft SQL Server, Oracle, PostgreSQL, and few others.

3. Implementation Details

3.1 Machine generated alarms

The machine generated signals from devices that are connected to patients is parsed by clinical experts using Splunk. The machine generated signals generate multiple false alarms and our task in the project is to reduce these false alarms and correctly predict real alarms.

3.2 Human Annotations

Human clinician expert annotations of datastream window between **[*t* - 5 min, *t* + 5 min]** around timestamp ***t*** of each alarm included in the *alarms.csv* file. For each subwindow (approx. 60 sec each), a window annotation is applied manually.

3.2.a. Annotation details

The decision process for human annotations is made as follows:

1. Review and analyze the recorded physiologic waveforms (Pleth/SpO2 and/or ECG)
    1.1 **Pleth/SpO2 waveform**:
        - interpretable (not too noisy, recognizable as waveforms)
        - not interpretable (noisy or other)
        - off (no signal)
        - disconnected (*e.g.,* true device flatline, as opposed to clinical asystole)

- **quality-of-signal** (QoS from Pleth/SpO2 processing by UCSF svd code):
  - correct (*i.e.,* reviewer assessment of Pleth/SpO2 waveform quality in agreement w/ QoS value)
  - incorrect (*i.e.,* reviewer assessment of Pleth/SpO2 waveform quality in **dis**agreement w/ QoS value)
  -

1.2 **EKG** waveform:

- interpretable (not too noisy, recognizable as waveforms)
- not interpretable (noisy or other)
- off (no signal)
- disconnected (*e.g.,* true device flatline, as opposed to clinical asystole)
- **clinical significance**:*
  - clinically significant (improvement)
  - clinically significant (deterioration)
  - no clinical significance
  - indeterminate clinical significance
- **clinical severity**:*
  - emergent
  - urgent
  - non-urgent
  - indeterminate

2. *Review **numeric values** (HR, RR, SBP/DBP/MAP) for 10-minute window surrounding red alarm for clinical correlation.

Here PPG(Pleth) is more important compared to ECG, because it reflects how blood is perfusing the tissues (the reverse is actually more important: even if the ECG looks "bad," if the Pleth/SpO2 waveform looks good, the ECG is probably artefact/noise).

The annotations files consist of 2 parts:

- Window annotations (in the first part)
- Alarm annotations (in the second part)

3.2.b. Window annotations

Each record in window annotations has a **start** and an **end** timestamp that defines a window expanded around corresponding alarm which is annotated in the second part. There are two kinds of window annotations:

- **In x00-##.yyyy-mm-dd_ekgAnnotations.txt files:**
  - EKG window annotations:
    - EKG INTERPRETABLE
    - EKG NOT INTERPRETABLE

        - EKG OFF
        - EKG DISCONNECT

Example 1:
2017-05-25T20:55:23.459,2017-05-25T20:55:28.747,EKG INTERPRETABLE
2017-05-25T20:55:30.193,2017-05-25T20:55:34.215,EKG INTERPRETABLE
2017-05-25T20:54:35.417,2017-05-25T20:54:39.936,EKG NOT Interpretable
2017-05-25T20:54:59.867,2017-05-25T20:55:23.233,EKG NOT Interpretable

- **In x00-##.yyyy-mm-dd_ppgAnnotations.txt files:**
    - PPG window annotations:
        - PPG INTERPRETABLE
        - PPG NOT INTERPRETABLE
        - PPG OFF
        - PPG DISCONNECT
    - QoS window annotations:
        - QoS CORRECT
        - QoS NOT CORRECT

Example 2:
2017-05-25T05:24:02.230,2017-05-25T05:24:08.811,PPG INTERPRETABLE,QoS NOT Correct
2017-05-25T05:27:41.538,2017-05-25T05:27:47.322,PPG INTERPRETABLE,QoS CORRECT
2017-05-25T05:27:36.069,2017-05-25T05:27:38.871,PPG NOT Interpretable,QoS CORRECT
2017-05-25T05:27:38.871,2017-05-25T05:27:41.492,PPG NOT Interpretable,QoS NOT Correct

3.2.c. Alarm annotations

The significance / severity annotations for all red alarms will be at the end of the annotations.txt files. It contains two kinds of annotations:

- Significance annotations:
    - Clin. SIGNIFICANT
    - Clin. INDETERMINATE
    - Clin. NOT SIGNIFICANT

- Severity annotations:
    - EMERGENT
    - URGENT
    - NON-URGENT
    - INDETERMINATE

The annotated combination of Significance and Severity are typically:

- SIGNIFICANT and URGENT
- INDETERMINATE and INDETERMINATE
- but can also be: [SIGNIFICANT and EMERGENT] and [SIGNIFICANT and NON-URGENT] (and other permutations...but less likely)

Example:
2017-05-25T05:28:27.088,Clin. SIGNIFICANT PPG alarm (URGENT)
2017-05-25T05:28:38.352,Clin. SIGNIFICANT PPG alarm (URGENT)
2017-05-25T14:24:02.256,Clin. SIGNIFICANT EKG alarm (URGENT)
2017-05-25T20:55:49.960,Clin. INDETERMINATE EKG alarm (INDETERMINATE)

3.3 Data Preprocessing

The data preprocessing stage is crucial for the project as the data is required to be consistent and in the correct format. This is done in order to parse the data correctly and it also aids to build a precise Machine Learning Model.

The clinical experts had provided us the human annotated files of different frequencies (5Hz Mean and 2Hz Median). There were 5 files and each one had an alarm with a time window of 5 minutes before and after the alarm was raised.

There are the following stages of preprocessing:

3.3.a. Merging:

The 5 files were merged together to have all the data at one source. The data was sorted and data containing only unique timestamp values were filtered.

3.3.b. Extracting good quality data:

This model relies on consistent and reliable data. The rows with mean QoS (Quality of Signal) value of 1 was extracted for this model.

3.3.c. Processing Human Annotated alarms:

The major task of this model is to process human annotated alarms which the clinical experts had provided us. The rows with a comment in the alarm field was suspected to be an alarm. Hence, for the sake of simplicity, the alarm field was changed to 1 for an alarm that was raised and 0 for no alarm raised. The data comprised of 5 alarms which were significant, and others were non-significant alarms for each type of frequency.

3.3.d. Experimenting with window size:

The window size around alarms was adjusted to fine tune the model. The window sizes of 5, 10, 50, 150, 300 rows were tried respectively.

3.4 HealthCare Predictive Model using SVM

Our initial hypothesis was to use a supervised learning method like SVM to understand and process the data. It was used to check how well the algorithm can learn from the data that is present. It was used to set a benchmark that was required to formulate a base foundation for the model.

Alternatively, we also used Deep Learning Models to build a sophisticated framework by taking advantages of SVM and achieve a higher precision for the model.

The data which is present has a very high percentage of no alarms and very low percentage of actual alarms that were raised. There were 5 alarms out of 13,946 rows of data.

Hence, a different approach was required to tackle this issue. The method used was the One-Class Classification (OCC) technique. It is a special case of supervised classification techniques. Here, the negative examples are absent during training but the negative samples may appear during the testing stage.

The data with maximum labels as 0 (no alarm) is equivalent to having no labels. This indicates that the labels are not providing any additional information for the Machine Learning Model to learn. It would rather lead to memorizing a specific pattern from the data. Therefore, it can be considered as an unsupervised learning technique which is like an outlier detection scheme. The information is present but no prior information about class labels (as seen in the case of supervised learning techniques).

The learning algorithm can be considered as unsupervised, but the main objective of OCC is classification. The OCC technique is very useful in this scenario due to very high difference in number of values (alarms Vs no alarms). The OCC mechanism can be used build a discriminative model like One-Class SVM (used in this project) or a generative models like Autoencoders.

4. Experiments and Analysis

4.1 Case analysis

This project required analysis of both machine generated alarms and human annotated significant alarms that were reported by clinical experts. This was done locally for testing purposes and for privacy of the patients' data that was received from the doctors. The model required more investment of time on the data preprocessing (aggregating timestamps to a frequency of 1second and QoS shifting) stage in order to get better results. There were a lot of experiments to understand the best features for model selection and parameter tuning. The Machine Learning model uses a Single Class SVM Classifier to predict machine generated alarms. Since, the data set is big, it takes some time to run this on a local server with Jupyter Notebook. Hence, a sample of the entire data was used for testing purposes. The data extracted resembled the actual data and comprised of nearly 90% of the data with 100,000 and 125,000 samples of data.

The first part of this project was to accurately predict the alarms that were generated by the machines. The final part of the project was to understand the data better and build a more sophisticated model to predict the human annotated alarms.

4.2 Experiment results of the Model

There were separate experiments carried out for machine generated alarms and human annotated alarms.

4.2.a. Machine Generated Alarms:

Here is how accuracy changes with respect to splitting of data between training and testing sets:

100,000 data set:

70% Train + 30% Test:
Accuracy of correctly predicting alarms: 2075/2149 = 96.5565379246%
Accuracy of correctly predicting no alarms: 15026/27852 = 53.9494470774%

Percentage of correct prediction: 17101/30001 = 57.0014332856%
Percentage of incorrect prediction: 12900/30001 = 42.9985667144%

75% Train + 25% Test :
Accuracy of correctly predicting alarms: 905/966 = 93.685300207 %
Accuracy of correctly predicting no alarms: 12463/24035 = 51.8535469108 %

Percentage of correct prediction: 13368/25001 = 53.4698612056 %
Percentage of incorrect prediction: 11633/25001 = 46.5301387944 %

80% Train + 20% Test :
Accuracy of correctly predicting alarms: 899/949 = 94.7312961012 %
Accuracy of correctly predicting no alarms: 9196/19052 = 48.2678983834 %

Percentage of correct prediction: 10095/20001 = 50.4724763762%
Percentage of incorrect prediction: 9906/20001 = 49.5275236238 %

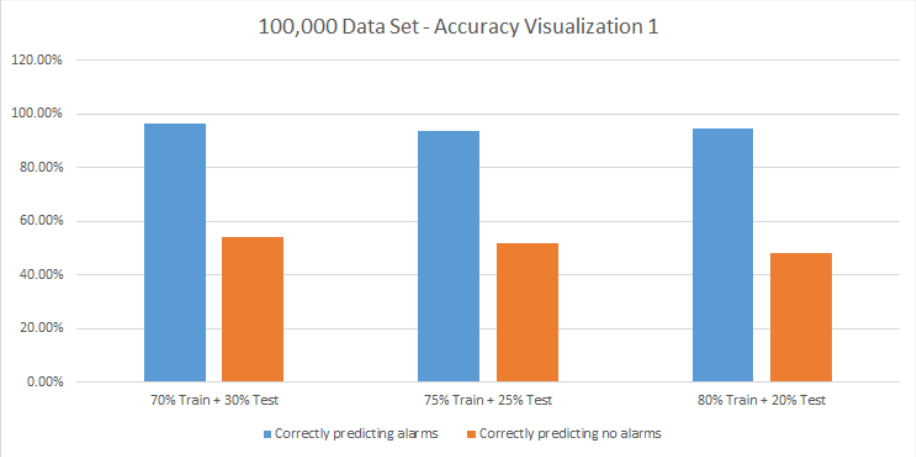

Fig. 1. Accuracy Visualization – 1: Correctly predicting alarms Vs Correctly predicting no alarms

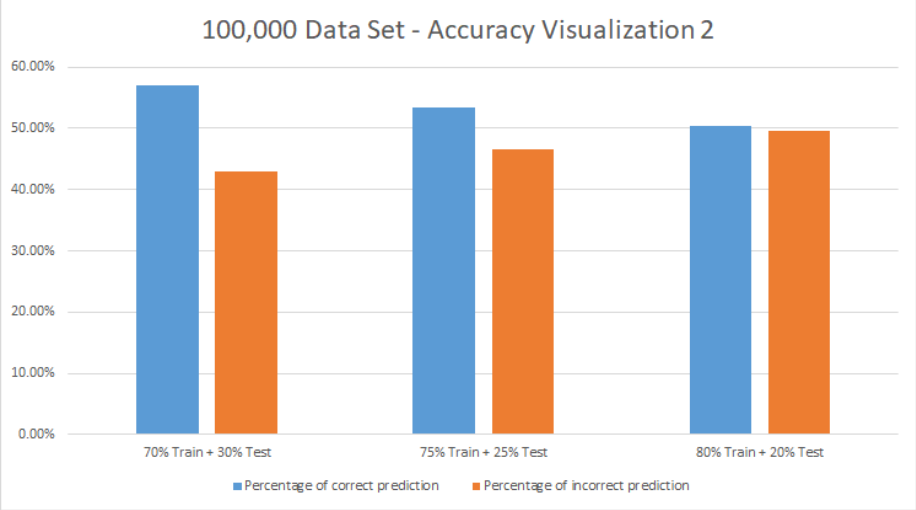

Fig. 2. Accuracy Visualization – 2: Percentage of correct prediction Vs Percentage of incorrect prediction

125000 data set:

70% Train + 30% Test:
Accuracy of correctly predicting alarms: 4932/4985 = 98.9368104313%
Accuracy of correctly predicting no alarms: 16579/32516 = 50.9872062984%

Percentage of correct prediction: 21511/37501 = 57.3611370363%
Percentage of incorrect prediction: 15990/37501 = 42.6388629637%

75% Train + 25% Test:
Accuracy of correctly predicting alarms: 4561/4584 = 99.4982547993%
Accuracy of correctly predicting no alarms: 13139/26667 = 49.2706341171%

Percentage of correct prediction: 17700/31251 = 56.638187578%
Percentage of incorrect prediction: 13551/31251 = 43.361812422%

80% Train + 20% Test:
Accuracy of correctly predicting alarms: 4032/4036 = 99.9008919722%
Accuracy of correctly predicting no alarms: 12282/20965 = 58.5833532077%

Percentage of correct prediction: 16314/25001 = 65.2533898644%
Percentage of incorrect prediction: 8687/25001 = 34.7466101356%

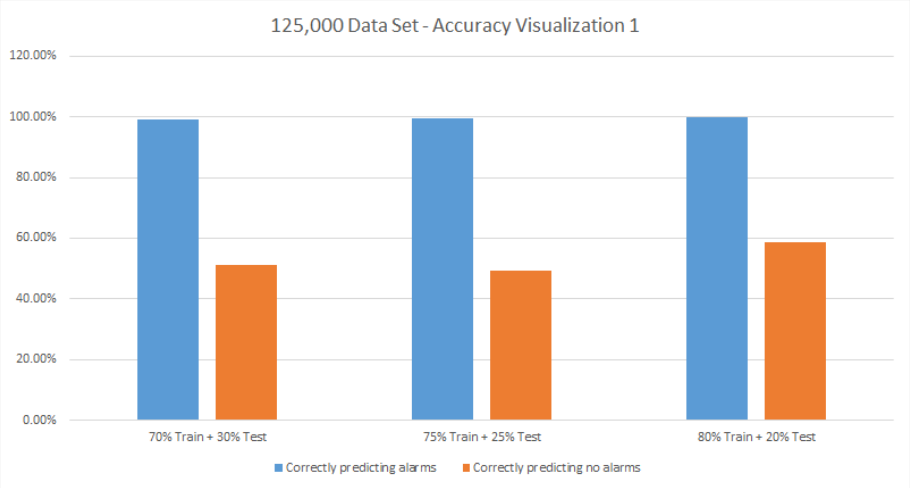

Fig. 3. Accuracy Visualization – 1: Correctly predicting alarms Vs Correctly predicting no alarms

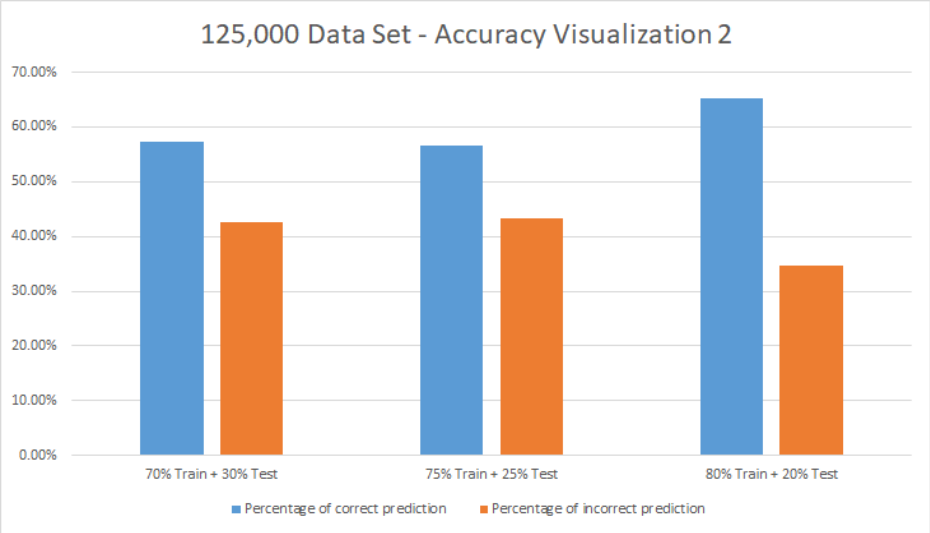

Fig. 4. Accuracy Visualization – 2: Percentage of correct prediction Vs Percentage of incorrect prediction

There were best results observed for the 125,000 data sample with 80% Training set + 20% Testing set split with an **accuracy** of **99.90%**.

We are currently trying to work on better feature selection & extraction to improve our results further. We would then change our focus towards high priority alarms.

4.2.b. Human Annotated High Priority Alarms:

The human annotated high priority alarms required many tests to achieve high precision model. The data was sampled into different window sizes (5, 10, 50, 150, 300 rows respectively). The data was split between training and testing set (60% and 75% respectively for each sample). The best results of **99.8335%** was achieved for window size of 300 rows with 60% split between training and testing set.

Window Size	5		10		50		150		300	
Split between Training and Testing Data	60%	75%	60%	75%	60%	75%	60%	75%	60%	75%
Number of Correctly Predicted True Alarms	0	0	0	0	0	0	0	0	0	0
Number of True Alarms	2	1	2	1	2	1	2	1	2	1
Number of Correctly Predicted False Alarms	19	13	39	25	199	125	599	360	1199	683
Number of False Alarms	19	13	39	25	199	125	599	375	1199	750
Accuracy of Correctly Predicted True Alarms	0	0	0	0	0	0	0	0	0	0
Accuracy of Correctly Predicted False Alarms	1	1	1	1	1	1	1	0.96	1	0.910667
Total Correctly Predicted Alarms	19	13	39	25	199	125	599	360	1199	683
Total Incorrectly Predicted Alarms	2	1	2	1	2	1	2	16	2	68
Total Number of Predictions	21	14	41	26	201	126	601	376	1201	751
Accuracy of Correctly Predicted Alarms	0.904762	0.928571	0.95122	0.961538	0.99005	0.992063	0.996672	0.957447	0.998335	0.909454
Accuracy of Incorrectly Predicted Alarms	0.095238	0.071429	0.04878	0.038462	0.00995	0.007937	0.003328	0.042553	0.001665	0.090546
F1-score	0.859524	0.89418	0.927439	0.942685	0.9851	0.988111	0.995011	0.975659	0.997503	0.951312
AUC	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.48	0.5	0.455333

Fig. 5. Experiments that were carried out along with their results

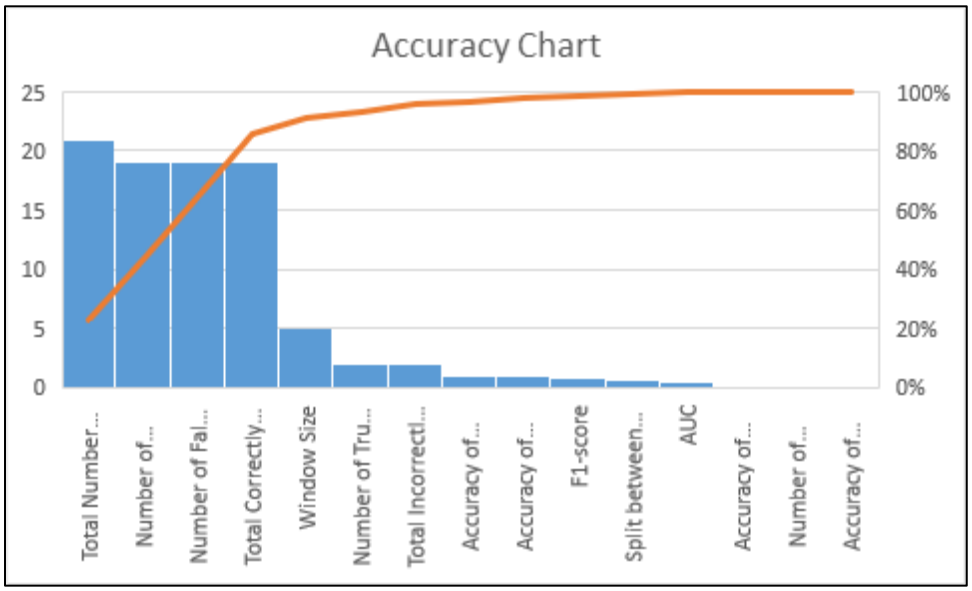

Fig. 6. Accuracy Chart of all metrics used to compute accuracy of the model

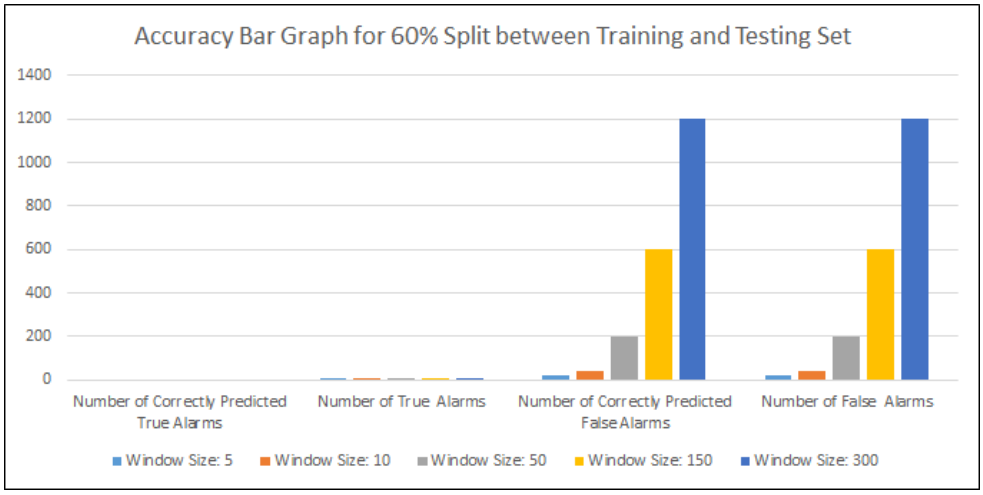

Fig. 7. Accuracy bar graph of predictions for true and false alarms for 60% split between training and testing set

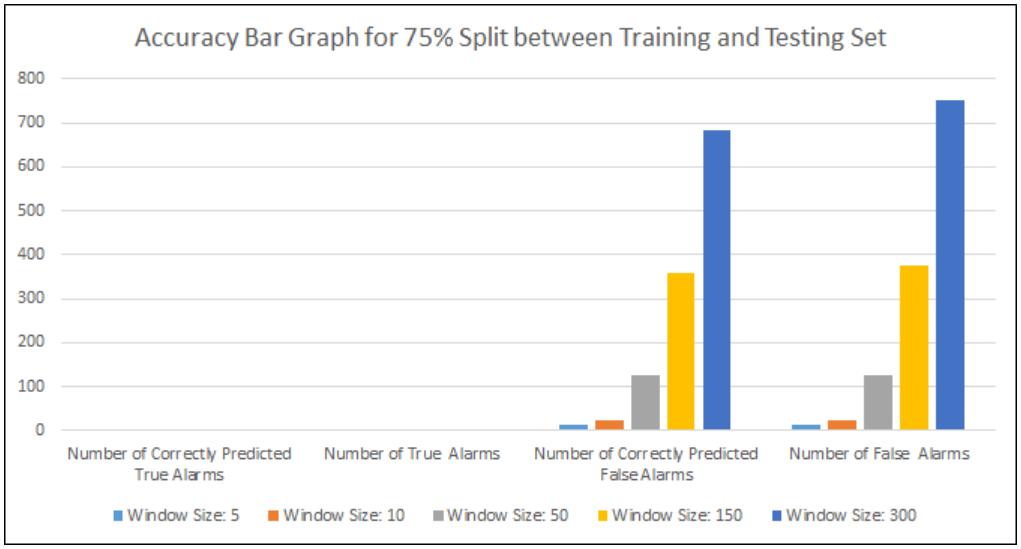

Fig. 8. Accuracy bar graph of predictions for true and false alarms for 75% split between training and testing set

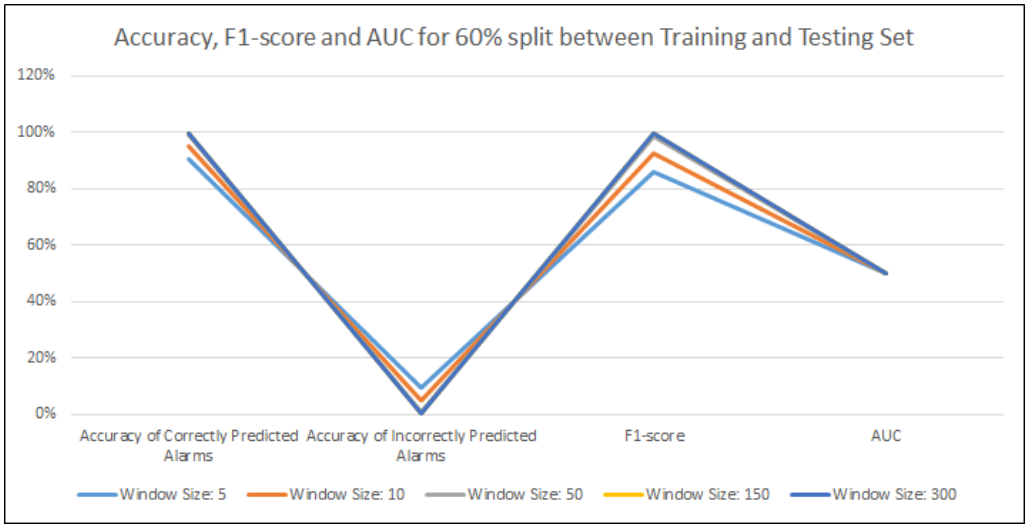

Fig. 9. Accuracy, F1-score and Area Under the Curve (AUC) FOR 60% split between Training and Testing Set

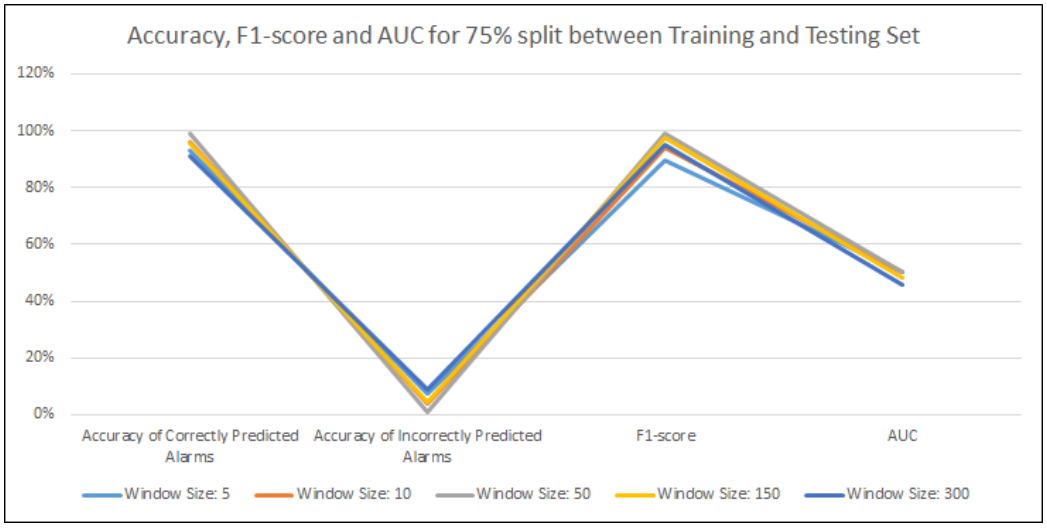

Fig. 10. Accuracy, F1-score and Area Under the Curve (AUC) FOR 75% split between Training and Testing Set

5. Future Works

We can refine the predictive model further by performing the following:

1) Find correlations between data to model clinical significance and severity of alarms.
2) Learn and associate monitor data with clinical outcomes to improve it further.
3) Reduce false positives further by training with more data samples and features.
4) Correctly predict true alarms by increasing window size to more than 5 minutes.
5) There were many NULL values that could be dealt by interpolation between waveforms to fill holes. The gap between holes isn't fixed which makes it challenging.

6. References

[1] https://www.wikipedia.org/

[2] https://stackoverflow.com/

[3] http://scikit-learn.org

[4] https://splunkbase.splunk.com

[5] https://stats.stackexchange.com/

[6] https://docs.python.org

[7] https://www.tutorialspoint.com/python