Capstone Project 2:
Early Prediction of Sepsis from Clinical Data
Milestone Report 1

Introduction

Sepsis is a highly dangerous condition where the body's response to infection or foreign sources cause further damage, from organ system damage, to septic shock, to death. According to the CDC, approximately 1.7 million people develop sepsis, while 270,000 die from it in the USA. Internationally, the WHO estimates that 30 million people suffer from sepsis, with a death rate of 6 million each year. In addition, the costs of diagnosing and treating sepsis in US hospitals alone costs up to $24 billion dollars. There is a very strong need for new methods to detect sepsis early and accurately. This is a significant need for hospitals and patients alike who are greatly affected by this disease. Creating a precise and accurate model that can predict early onset of sepsis would save lives, and hospitals from expending valuable resources. Predicting it too early consumes limited hospital resources, while predicting it too late makes improving sepsis outcomes incredibly challenging.

To create such a model, I will be using data made available through Physionet.org. Physionet is a platform that provides publically available datasets and offers yearly challenges to solving complex problems through using such data. It is very similar to Kaggle.com, however the primary differences are that Physionet focuses around clinical and medical data, and most importantly, the datasets are MESSY. They have not been preprocessed for typical statistical analysis and machine learning applications. Physionet's 2019 Challenge is to use clinical data provided to assess early detection of Sepsis. I plan to participate in this challenge and will use such data to create my models.

The data consists of 5000 PSV files, each containing a patient's clinical data starting at the patient's admission to the ICU. Clinical data is collected hourly and is represented row-wise, with features including heart rate, temperature, white blood cell count, and many more.

We are trying to predict, based on the available clinical data, whether a patient will develop sepsis. My approach will thus be to wrangle the data into a single dataframe preprocessed appropriately for supervised learning. Because this is a Time-Series problem, Time-Series analysis and resources will be implemented through adding features that appropriately capture significant trends since the data points can absolutely relate to each other. This is my initial plan, which I'm certain will develop, evolve, and change throughout this project.

My deliverables will include the code/algorithms to access my models, as well as a paper and slides summarizing my thought process throughout this project.

Exploratory Data Analysis/Data Cleansing

The initial steps I took for exploring the data are as follows:

- I checked to see how many of the 5000 patients ended up testing positive for Sepsis
- I created a visual representation of the hour at which these patients got diagnosed with sepsis, with t = 0 being when the patient gets admitted to the ICU
- I observed where missing values are prevalent

Each patient file includes 40 features that can be put into three major categories. Vital Signs, Laboratory Values, and Demographic information. Below, I show the categories and their corresponding features:

- Vital signs
    - HR: Heart Rate (beats per minute)
    - O2Sat: Pulse oximetry, (%)
    - Temp: Temperature (Deg C)
    - SBP: Systolic BP (mm Hg)
    - MAP: Mean arterial pressure (mm Hg)
    - DBP: Diastolic BP (mm Hg)
    - Resp: Respiration rate (breaths per minute)
    - EtCO2: End tidal carbon dioxide (mm Hg)

- Laboratory Values
    - BaseExcess: Measure of excess bicarbonate, (mmol/L)
    - HCO3: Bicarbonate, (mmol/L)
    - FiO2: Fraction of inspired oxygen, (%)
    - pH: potential Hydrogen
    - PaCO2: Partial pressure of carbon dioxide from arterial blood, (mm Hg)
    - SaO2: Oxygen saturation from arterial blood, (%)
    - AST: Aspartate transaminase, (IU/L)
    - BUN: Blood urea nitrogen, (mg/dL)
    - Alkalinephos: Alkaline phosphatase, (IU/L)
    - Calcium: (mg/dL)
    - Chloride: (mmol/L)
    - Creatinine: (mg/dL)
    - Bilirubin_direct: (mg/dL)
    - Glucose: Serum glucose, (mg/dL)
    - Lactate: Lactic acid, (mg/dL)
    - Magnesium: (mmol/dL)
    - Phosphate: (mg/dL)

    - Potassium: (mmol/L)
    - Bilirubin_total: Total bilirubin, (mg/dL)
    - TroponinI: Troponin I, (ng/mL)
    - Hct: Hematocrit, (%)
    - Hgb: Hemoglobin, (g/dL)
    - PTT: partial thromboplastin time, (seconds)
    - WBC: Leukocyte count (count*10^3/μL)
    - Fibrinogen: (mg/dL)
    - Platelets: (count*10^3/μL)

- Demographics
    - Age: Years (100 for patients 90 or above)
    - Gender: Female (0) or Male (1)
    - Unit1: Administrative identifier for ICU unit (MICU)
    - Unit2: Administrative identifier for ICU unit (SICU)
    - HospAdmTime: Hours between hospital admit and ICU admit
    - ICULOS: ICU length-of-stay (hours since ICU admit)

In addition, each patient file also includes a feature called ‘SepsisLabel’, which, per row, contains the value 0 if the patient has not been diagnosed with Sepsis, and value 1 if/when they do. In other words, negative patients will have this column filled only with 0s, while positive patients start out with 0s, which eventually convert to 1s. Through this, I wrote a for loop saying if a patient’s ‘SepsisLabel’ column contains the value 1, append the psv file to a list, and then use the len function to see how many of the 5000 patients tested positive. From this we can see that there are 279 positive patients, making this a highly imbalanced dataset. This means it will be imperative to monitor not just accuracy from our models, but also precision, recall, and area under the ROC curve.

The number of rows for each patient file are not uniform. Some contain as little as 20 rows, while others contain rows as many as 200. This simply shows that some patients have longer stays in the ICU than others.

Among the positive patients, I wanted to see how quickly they were able to be diagnosed. Therefore I made a bar plot showing the number of patients who were diagnosed within their first day in the ICU, those within the second day, and so on. The results from this can be seen below in **Figure 1**:

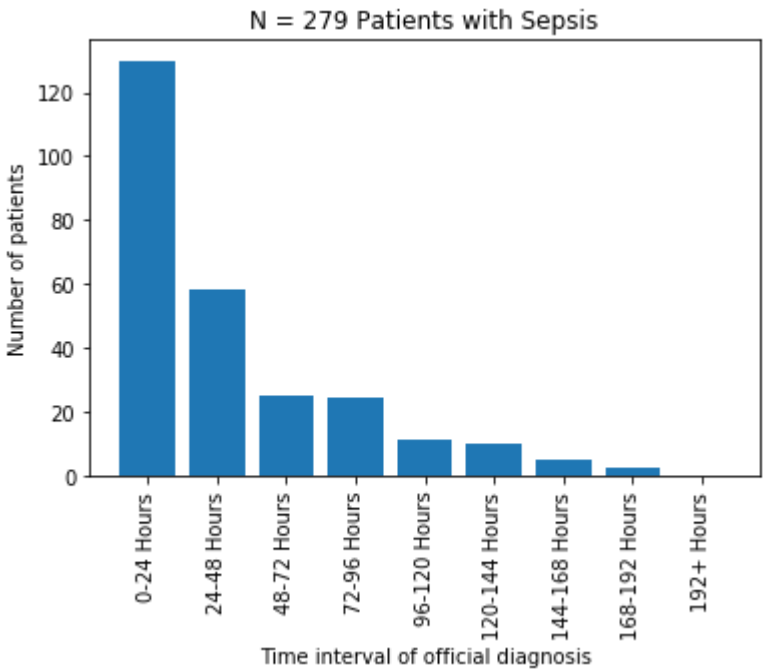

Figure 1: Time interval of official diagnosis

We see that only a little over 120 patients get diagnosed within the first 24 hours, while others don't receive confirmation until days later, elucidating the need for stronger ways to improve early prediction of sepsis.

As mentioned previously, these patient files are messy, due to the inconsistent missing values present among them. I handle the missing values in the Vital Sign columns by filling with the mean, because these features are more commonly collected hourly, meaning less frequent missing values, so such imputation methods like filling with the mean can suffice. The missing values in the Laboratory Value columns tend to be much more abundant, as in many cases, only one or two values are real. This is likely due to the Lab values being collected much less frequently at the hospitals where the data originate. Therefore, the missing values from these columns are handled through forward and backward filling.

To visualize the effects of my imputation techniques, I created histograms with both the original and imputed datasets overlapping each other, for each feature from the vital signs and lab values, the resulting distributions maintain their normal shape and do not cause major skews, which is a good indication that this can be an effective method for filling the missing values. Examples of the generated overlapping distributions can be seen in **Figures 2 and 3**.

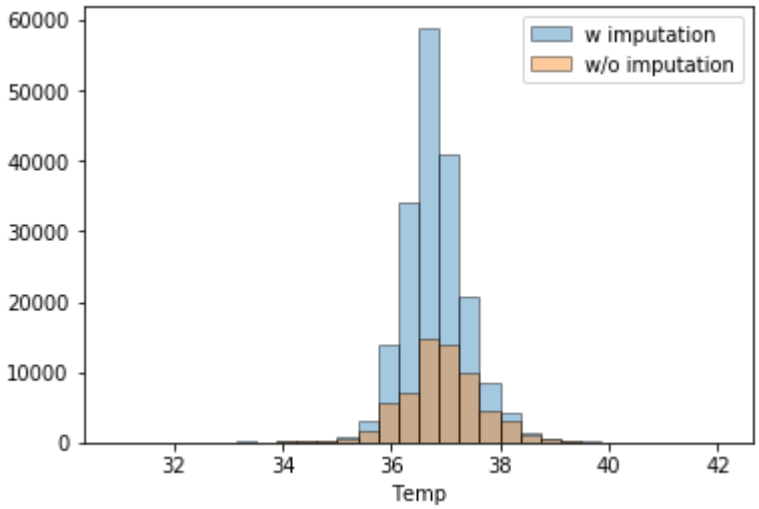

Figure 2: Distribution of all values from the Temperature feature

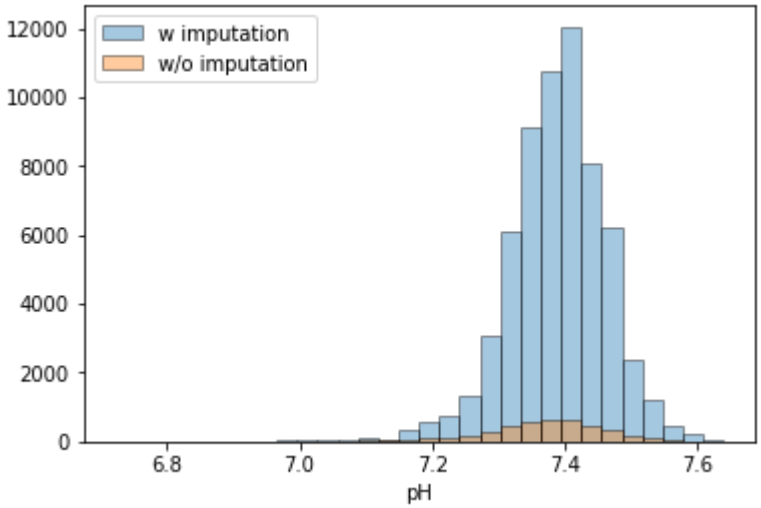

Figure 3: Distribution of all values from the pH feature

Since the PSV files are divided, I want to concatenate them all into a single dataframe. This facilitates parsing through the different patients, specifically by adding an additional column (patientID), whose value would be unique to each individual patient. Per individual patient, the only missing values left are those initially present in columns in its entirety.

When further preparing the dataset, I want to observe, even after these imputation methods, how many missing values are present among patients. I aim to use a **grouped forward selection** process to observe the effects of features on model performance, therefore I aim to start with models that only taking in the Vital Signs and Demographics. With that in mind, I want to observe the number of real values present in the Vital Signs columns shown in **Figure 4**, as well as the number of each missing columns per patient, shown in **Figure 5**.

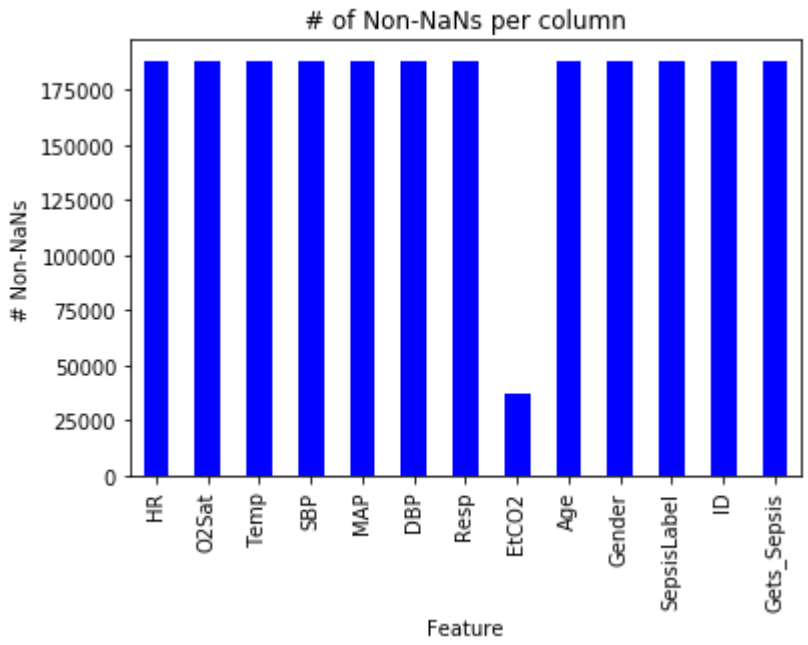

Figure 4: Number of real values, per feature, across all patients

```
Number of patients with entirely missing columns from the HR feature: 1
Number of patients with entirely missing columns from the O2Sat feature: 0
Number of patients with entirely missing columns from the Temp feature: 10
Number of patients with entirely missing columns from the SBP feature: 2
Number of patients with entirely missing columns from the MAP feature: 17
Number of patients with entirely missing columns from the DBP feature: 4
Number of patients with entirely missing columns from the Resp feature: 16
Number of patients with entirely missing columns from the EtCO2 feature: 4216
Number of patients with entirely missing columns from the Age feature: 0
Number of patients with entirely missing columns from the Gender feature: 0
Number of patients with entirely missing columns from the SepsisLabel feature: 0
Number of patients with entirely missing columns from the ID feature: 0
Number of patients with entirely missing columns from the Gets_Sepsis feature: 0
```

Figure 5: Number of patients with entirely missing columns from all Vital Sign features

From these figures, we can see that despite imputation methods, 84% (4216/5000) of patients do have fully missing value columns from the EtCO2 feature. Moving forward, it would be wise to remove the EtCO2 column from the dataset, when creating our first round of models. That way, when I use the .dropna() command of the dataframe, much fewer patients will be removed and valuable data can be preserved. From doing this, I preserve 4956 patients, and end up, at this point, with a dataframe with no missing values, ready to be used for binary classification models, which will be the next step.