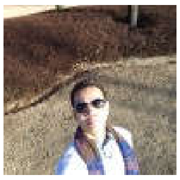
Time Series Analysis | Physionet 2019 Early Prediction of Sepsis

2. June 2019

Source: Deep Learning on Medium



Mun Kar Kin

Jun 2

Introduction

I am currently taking part in the PhysioNet/Computing in Cardiology Challenge (2019) for the early prediction of sepsis. According to the website, sepsis is a life-threatening condition that occurs when the body's response to infection causes tissue damage, organ failure, or death (Singer *et al.*, 2016). Some of the symptoms of sepsis include fit, convulsion rashes, heavy breathing and so forth. Early detection and antibiotic treatment of sepsis are critical for improving sepsis outcomes, where each hour of delayed treatment has been associated with roughly an ***4–8%*** increase in mortality (Kumar *et al.*, 2006; Seymour*et al.*, 2017). Hence, successful predictions of sepsis may potentially save many lives. For more information about the competition, visit

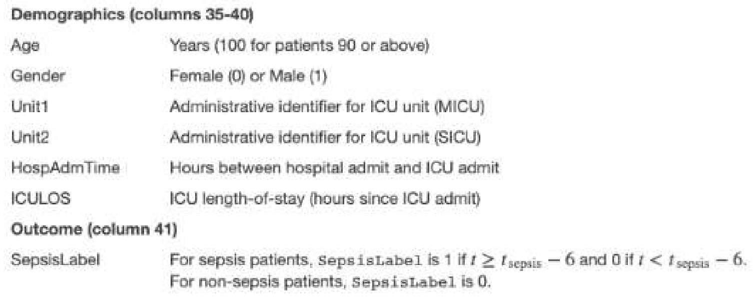
Early Prediction of Sepsis from Clinical Data: the PhysioNet/Computing in Cardiology Challenge 2019
We ask participants to design and implement a working, open-source algorithm that can, based only on the clinical data…
physionet.org

The article contains three main parts.

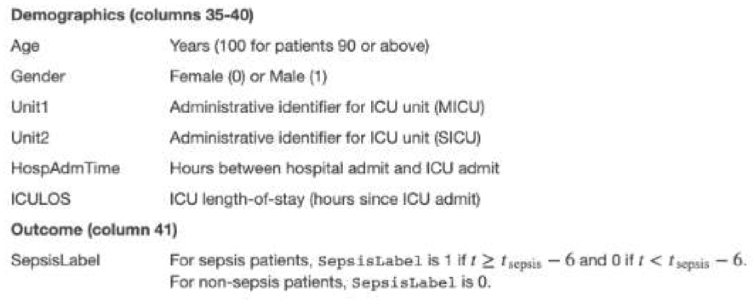
1. ***Explanatory Data Analysis (EDA)***
2. ***Missing Data Imputation using Gaussian Mixture Models (GMM)s***
3. ***Deploying a Deep Learning Model (Stacked LSTM with time distributed fully connected layers)***

This post is meant to provide a quick and dirty implementation of the entire pipeline and start identifying potential areas of improvement as Andrew Ng always preaches, *" we first implement something quick and dirty, and then we iterate "*.

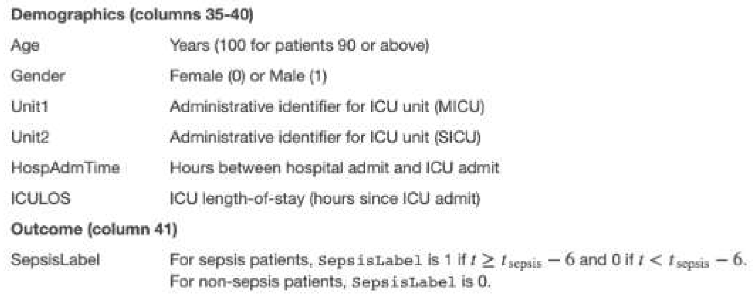
Explanatory Data Analysis (EDA)



Demographics (columns 35-40)	
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)
Outcome (column 41)	
SepsisLabel	For sepsis patients, `SepsisLabel` is 1 if $t \geq t_{\text{sepsis}} - 6$ and 0 if $t < t_{\text{sepsis}} - 6$. For non-sepsis patients, `SepsisLabel` is 0.



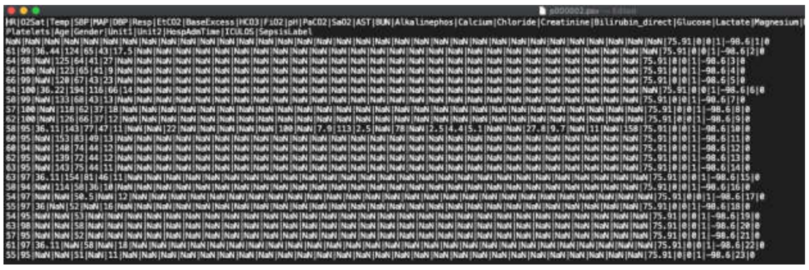
Demographics (columns 35-40)	
Age	Years (100 for patients 90 or above)
Gender	Female (0) or Male (1)
Unit1	Administrative identifier for ICU unit (MICU)
Unit2	Administrative identifier for ICU unit (SICU)
HospAdmTime	Hours between hospital admit and ICU admit
ICULOS	ICU length-of-stay (hours since ICU admit)
Outcome (column 41)	
SepsisLabel	For sepsis patients, `SepsisLabel` is 1 if $t \geq t_{\text{sepsis}} - 6$ and 0 if $t < t_{\text{sepsis}} - 6$. For non-sepsis patients, `SepsisLabel` is 0.



Demographics (columns 35-40)	
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)
Outcome (column 41)	
SepsisLabel	For sepsis patients, `SepsisLabel` is 1 if $t \geq t_{\text{sepsis}} - 6$ and 0 if $t < t_{\text{sepsis}} - 6$. For non-sepsis patients, `SepsisLabel` is 0.

Reference of all the features in the dataset

The data used in the competition is from the ICU patients in separate hospital systems. There are a total of *40336 patients*, with readings taken at an hourly basis. The number of **time-dependant hours** in a patient may be vastly different for each patient, varying in the range from about 10 hours of data per patient to a few hundred hours per patient. For example, the data for one of the patient may take the following form:

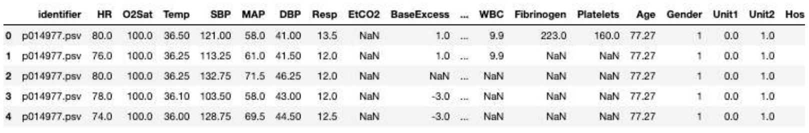


```
HR|O2Sat|Temp|SBP|MAP|DBP|Resp|EtCO2|BaseExcess|HCO3|FiO2|pH|PaCO2|SaO2|AST|BUN|Alkalinephos|Calcium|Chloride|Creatinine|Bilirubin_direct|Glucose|Lactate|Magnesium|
Platelets|Age|Gender|Unit1|Unit2|HospAdmTime|ICULOS|SepsisLabel
NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|1|0
61|99|36.44|124|65|43|17.5|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|2|0
64|98|NaN|125|64|41|27|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|3|0
56|100|NaN|123|65|41|9|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|4|0
66|99|NaN|120|67|43|23|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|5|0
94|100|36.22|194|116|66|14|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|6|0
58|99|NaN|133|68|43|13|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|7|0
57|100|NaN|118|62|37|18|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|8|0
62|100|NaN|126|66|37|12|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|9|0
58|95|36.11|143|77|47|11|NaN|NaN|22|NaN|NaN|NaN|NaN|NaN|100|NaN|7.9|113|2.5|NaN|78|NaN|2.5|4.4|5.1|NaN|NaN|27.8|9.7|NaN|11|NaN|158|75.91|0|0|1|-98.6|10|0
60|95|NaN|153|83|49|13|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|11|0
60|94|NaN|148|74|44|12|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|12|0
62|95|NaN|139|72|44|12|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|13|0
63|95|NaN|143|75|44|11|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|14|0
63|97|36.11|154|81|46|11|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|15|0
58|94|NaN|114|58|36|10|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|16|0
54|97|NaN|NaN|50.5|NaN|12|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|17|0
55|97|36|NaN|52|NaN|16|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|18|0
54|95|NaN|NaN|53|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|19|0
63|98|NaN|NaN|58|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|20|0
57|95|NaN|NaN|52|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|21|0
61|97|36.11|NaN|58|NaN|18|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|22|0
55|95|NaN|NaN|51|NaN|11|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|NaN|75.91|0|0|1|-98.6|23|0
```

Figure 1: Sample dataset for one of the patient

Notice that the dataset contains many missing values indicated by NaN. It is common to have a dataset from the real world with more missing values than real data points. This is an issue that we will address later on.

Since the datasets contain 40336 independent files, it is helpful to compress all of the data points in a csv file with an "identifier" to indicate the origin of the data points. Hence, the dataset now takes the shape of (m, n), where m = 1552210 and n = 42.



	identifier	HR	O2Sat	Temp	SBP	MAP	DBP	Resp	EtCO2	BaseExcess	...	WBC	Fibrinogen	Platelets	Age	Gender	Unit1	Unit2	Hos
0	p014977.psv	80.0	100.0	36.50	121.00	58.0	41.00	13.5	NaN	1.0	...	9.9	223.0	160.0	77.27	1	0.0	1.0	
1	p014977.psv	76.0	100.0	36.25	113.25	61.0	41.50	12.0	NaN	1.0	...	9.9	NaN	NaN	77.27	1	0.0	1.0	
2	p014977.psv	80.0	100.0	36.25	132.75	71.5	46.25	12.0	NaN	NaN	...	NaN	NaN	NaN	77.27	1	0.0	1.0	
3	p014977.psv	78.0	100.0	36.10	103.50	58.0	43.00	12.0	NaN	-3.0	...	NaN	NaN	NaN	77.27	1	0.0	1.0	
4	p014977.psv	74.0	100.0	36.00	128.75	69.5	44.50	12.5	NaN	-3.0	...	NaN	NaN	NaN	77.27	1	0.0	1.0	

Figure 2: Dataset Compressed into Comma Separated Values (csv) format

As aforementioned, the dataset is full of missing values. It might be helpful to visualize the number of missing values for each feature in the form of a graph sorted in ascending order.

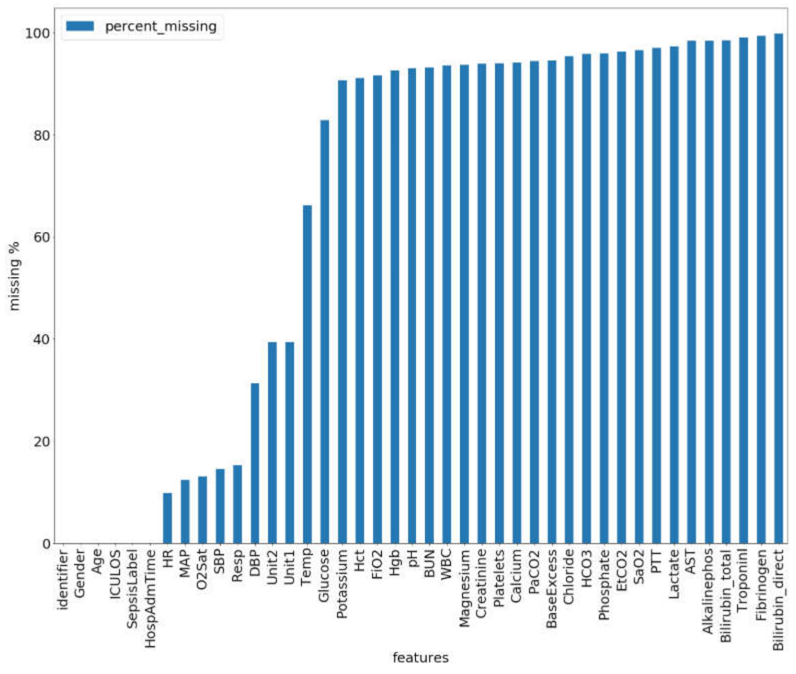


Figure 3: % of missing values for each f

Some of the features such as Bilirubin_direct, Fibrinogen, and so forth has 99% of missing values! This certainly makes it difficult for any model to learn the distribution of the data with so much of missing values. Let's visualize the correlation between each feature at a patient selected at random, and at the same time visualizing the number of missing values in the form of a heatmap.

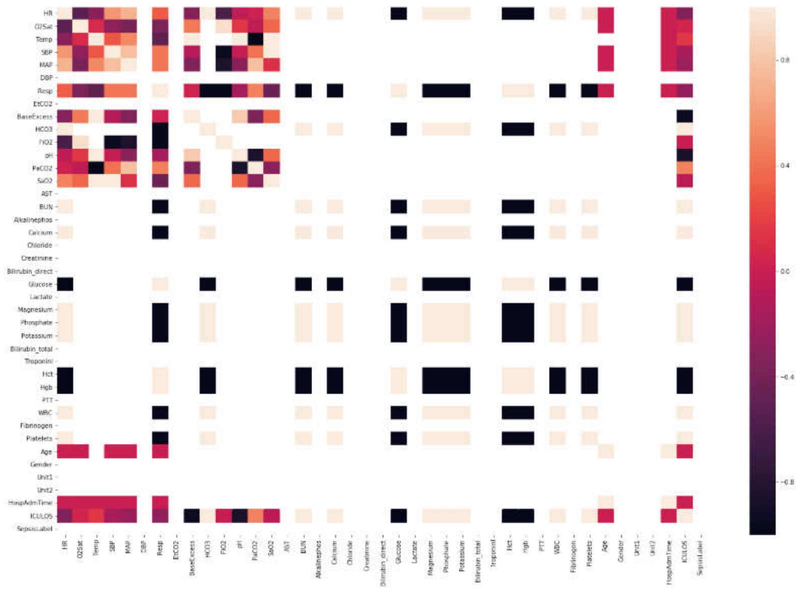


Figure 4: Correlation Between Each Feature for a Randomly Selected Patient

All the white spaces indicate that the features are missing. Based on the heatmap, it seems that most of the features on the upper left-hand corner may be put to good use. Besides that, the features are not very correlated. Hence, none of the features are found to be particularly redundant at this point in time. Let us then move on by visualizing the correlation between each feature with all 40336 patients combined.

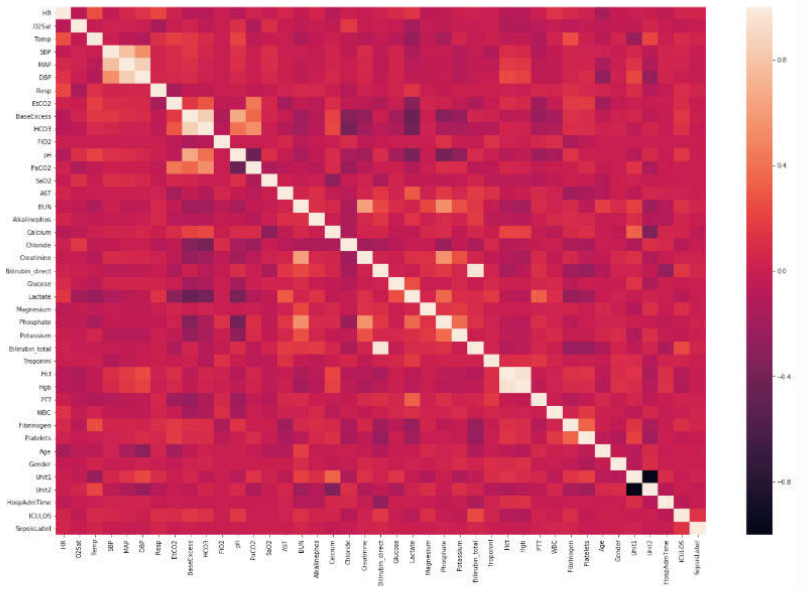


Figure 5: Correlation Between Each Feature for the Entire Dataset

It is now more evident that the features are uncorrelated.

It is usually helpful to visualize the distribution of the dataset to look out for any potential biases. Taking the dataset as a whole, the distribution is as follow, where 1.8% of all data points have a SepsisLabel == 1.

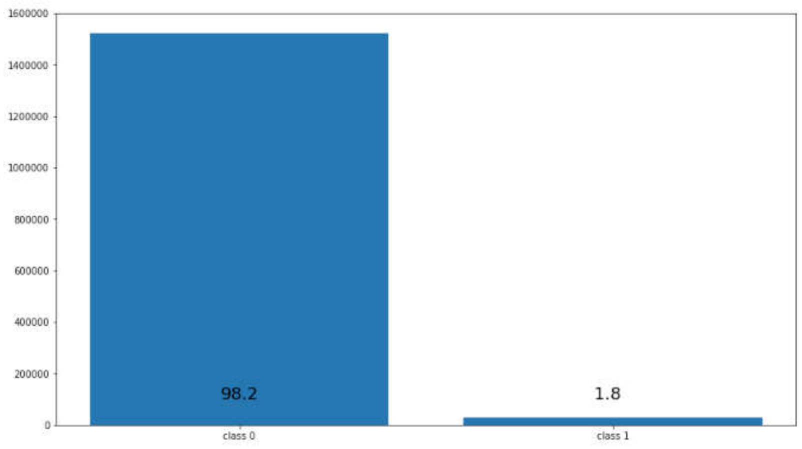


Figure 6

However, the distribution may not be an accurate representation of the problem we are dealing with as we are trying to predict sepsis as per patient. Hence, let's reformalize the problem slightly differently. We let *SepsisLabel == 1* if the patient will eventually suffer from sepsis within his/her time frame (may not suffer from sepsis in the beginning), and *SepsisLabel == 0* otherwise. Hence, the distribution now takes on the following form, with 7.27% of the patients in our dataset that will suffer from sepsis eventually.

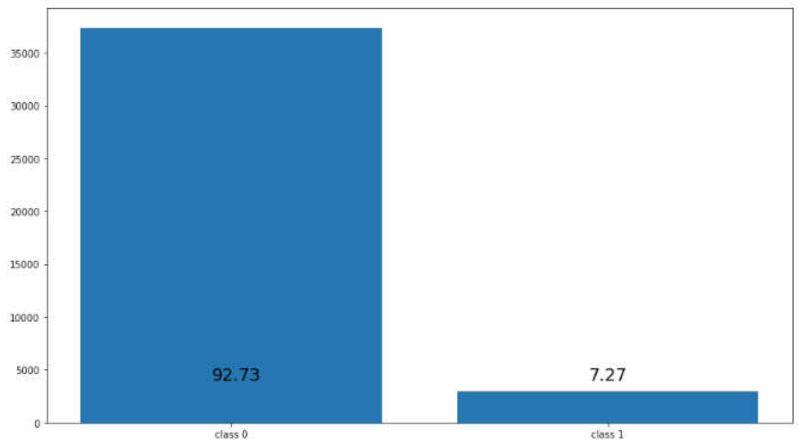
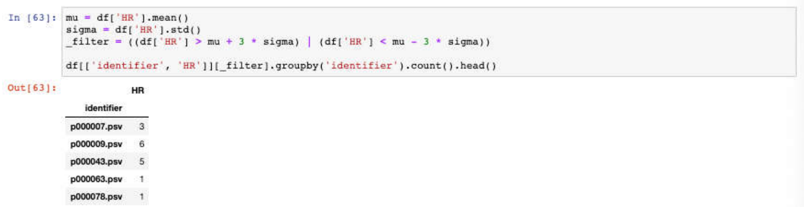


Figure 7

It is now obvious that is dataset is highly biased towards patients who will never suffer from sepsis within his/her time frame in the hospital. Techniques such as stratification will be required to ensure that the distribution of data remains consistent across the training, testing and cross-validation set.

Outliers are usually an important indication of the target label. Outliers could be due to an erroneous sensor reading, or it could provide very useful information for the class label classification. We define an outlier for a feature to be a value that is 3 standard deviations away from the mean of the feature. Let's visualize the outliers for one of the feature "HR".



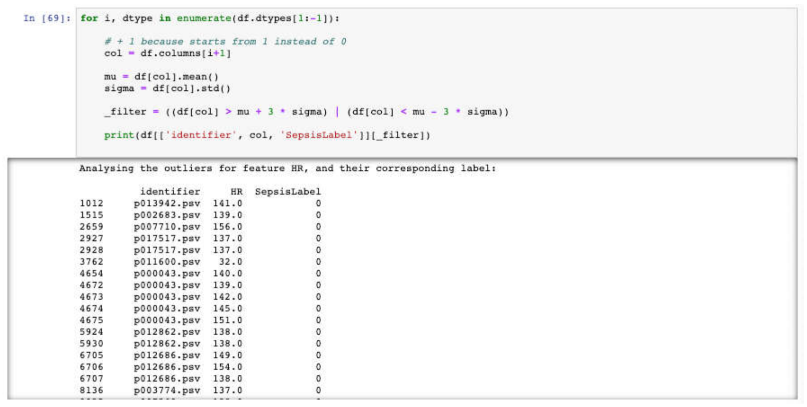
```
In [63]: mu = df['HR'].mean()
         sigma = df['HR'].std()
         _filter = ((df['HR'] > mu + 3 * sigma) | (df['HR'] < mu - 3 * sigma))

         df[['identifier', 'HR']][_filter].groupby('identifier').count().head()
```

Out[63]:

identifier	HR
p000007.psv	3
p000009.psv	6
p000043.psv	5
p000063.psv	1
p000078.psv	1

Figure 8: Visualizing the Number of Outliers of Each Patient

Most of the patients have at least one or more outliers. It is helpful to visualize the values of the outliers and the corresponding label.



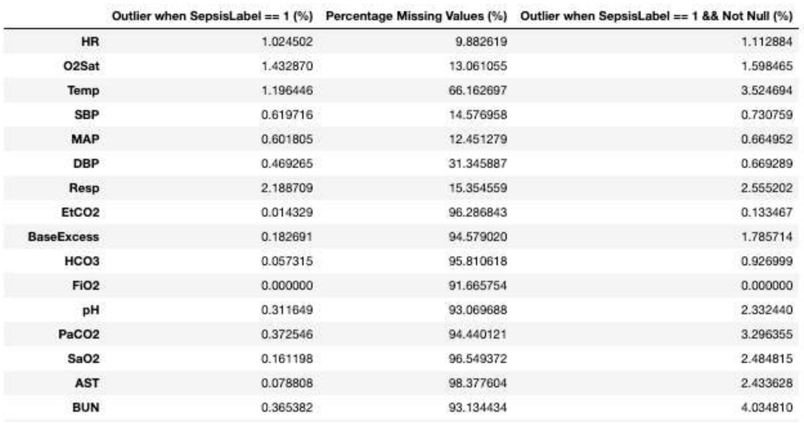
```
In [69]: for i, dtype in enumerate(df.dtypes[1:-1]):

             # + 1 because starts from 1 instead of 0
             col = df.columns[i+1]

             mu = df[col].mean()
             sigma = df[col].std()

             _filter = ((df[col] > mu + 3 * sigma) | (df[col] < mu - 3 * sigma))

             print(df[['identifier', col, 'SepsisLabel']][_filter])
```

```
Analysing the outliers for feature HR, and their corresponding label:

         identifier     HR  SepsisLabel
1012    p013942.psv  141.0            0
1515    p002683.psv  139.0            0
2659    p007710.psv  156.0            0
2927    p017517.psv  137.0            0
2928    p017517.psv  137.0            0
3762    p011600.psv   32.0            0
4654    p000043.psv  140.0            0
4672    p000043.psv  139.0            0
4673    p000043.psv  142.0            0
4674    p000043.psv  145.0            0
4675    p000043.psv  151.0            0
5924    p012862.psv  138.0            0
5930    p012862.psv  138.0            0
6705    p012686.psv  149.0            0
6706    p012686.psv  154.0            0
6707    p012686.psv  138.0            0
8136    p003774.psv  137.0            0
```

Figure 9: Visualizing the Outliers in the dataset

The information shows that the outliers are consistently around the same range. Typically, the data points are out of the ordinary, for example taking on values of 99999.999 among a pool of readings in the range of hundredths, that could be a potential sensor fault.

It is also often helpful to visualize the important indicators in the form of a table. In certain cases, one might be able to extract useful information from the dataset.



	Outlier when SepsisLabel == 1 (%)	Percentage Missing Values (%)	Outlier when SepsisLabel == 1 && Not Null (%)
HR	1.024502	9.882619	1.112884
O2Sat	1.432870	13.061055	1.598465
Temp	1.196446	66.162697	3.524694
SBP	0.619716	14.576958	0.730759
MAP	0.601805	12.451279	0.664952
DBP	0.469265	31.345887	0.669289
Resp	2.188709	15.354559	2.555202
EtCO2	0.014329	96.286843	0.133467
BaseExcess	0.182691	94.579020	1.785714
HCO3	0.057315	95.810618	0.926999
FiO2	0.000000	91.665754	0.000000
pH	0.311649	93.069688	2.332440
PaCO2	0.372546	94.440121	3.296355
SaO2	0.161198	96.549372	2.484815
AST	0.078808	98.377604	2.433628
BUN	0.365382	93.134434	4.034810

Figure 10: A truncated table showing different indicators

In the table above (in the form of a panda DataFrame), the first column tells us the percentage of outliers that are present when the patient is suffering from sepsis. A high value indicates that the feature is highly important for the prediction. The column in the center indicates the percentage of missing values, and it is here for comparison and visualizing purposes. Lastly, the last column tells us the percentage of outliers that are present when the patient is suffering from sepsis, conditioned on only taking into account the rows with no missing values. This is important as the dataset is full of missing values.

The table above is telling us a lot of information that might be useful. There are several observations that can be made from the first column, for example,

1. If a feature value of ICULOS is an outlier, there is a 14% chance that the patient might suffer from a Sepsis
2. If a feature value of HospAdmTime is an outlier, there is a 3% chance that the patient might suffer from a Sepsis
3. etc …

However, this may not be an accurate representation because when the sepsis is positive, the corresponding feature could be a null value. Hence, the third column takes care of this. Essentially, the third column is telling us that, whenever the patient suffers from sepsis, and that the corresponding feature is not null, what is the percentage of that happening? Hence several observations can be made, for example,

1. If a feature value of Bilirubin_direct is an outlier, the patient will have sepsis for 6% of the time (this time with the null values taken care of)
2. Also, for example, notice that ICULOUS has 0 % missing values. Hence, the first column and the third column has the same value, which makes sense because the feature does not have null values.

HospAdmTime is an indicative measure of the hours between hospital admit and ICU admits, and ICULOS is nothing but the ICU the length-of-stay. The data is telling us that the longer one stays in the hospital, the higher the risk of eventually developing sepsis. Intuitively, this makes sense. Hence, it gives further confidence to the visualization techniques performed. There are many other important factors that can be derived from the table but is not explained here for simplicity.

This whole time we have been visualizing the data in the form of numbers. Let's convert the data into pretty diagrams. Previously, we have mentioned that ICULOS could be an important indicative factor to the eventual development of sepsis. Indeed, when we visualize the distribution of the feature according to their individual class labels, we could a clear distinguishing point from the graph.

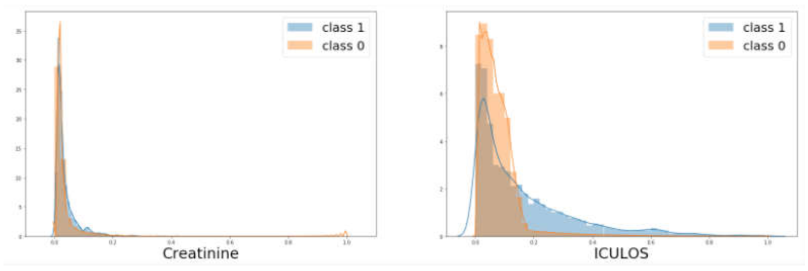


Figure 11: Visualization of Features of the dataset according to class labels

Recall that from the table in figure 10, feature MAP has one of the lowest values of the number of outliers present conditioned on taking only the count of values that is not null, visually, it is also relatively difficult to differentiate between the two classes.

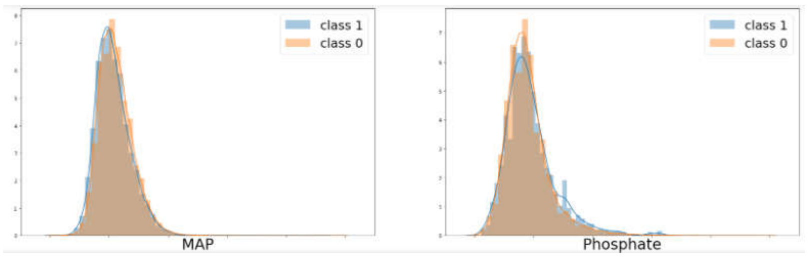


Figure 12: Visualization of Features of the dataset according to class labels

There are a total of 41 features but only 4 charts are shown for clarity.

Next, let’s visualize the time series dependency of each of the features of at patient level. A patient that will eventually develop sepsis, and a patient who does not develop sepsis for his/her entire time in the hospital is visualized below.

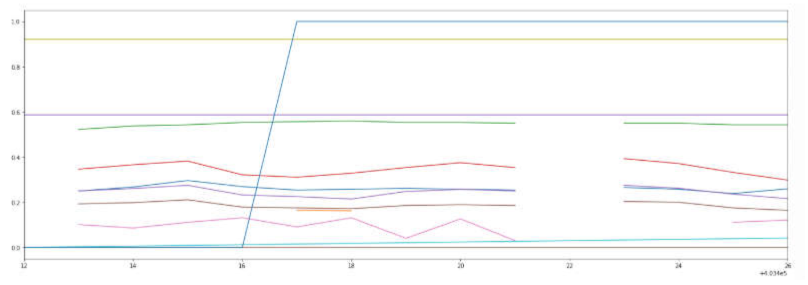


Figure 13: Visualizing of all features of a randomly selected patient in the form of a graph (the blue line that has a sharp transition of value from 0 to 1 indicates the time horizon in which the patient transition into sepsis.

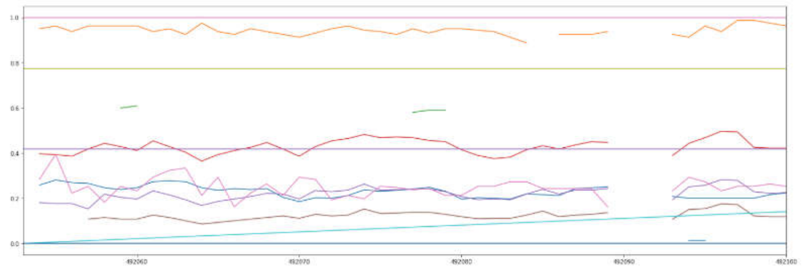
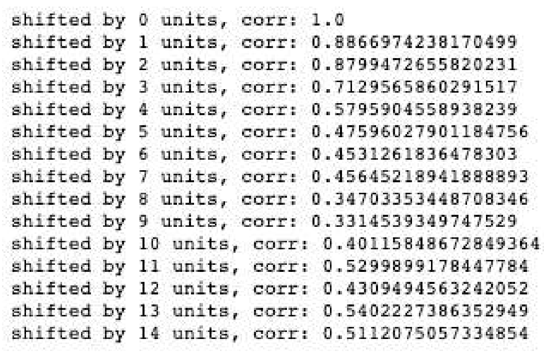


Figure 14: Visualizing of all features of a randomly selected patient in the form of a graph. (this time, the patient does not suffer from sepsis)

Notice that the patient who does not suffer from sepsis eventually has a noisier graph, whereas the patient who will eventually suffer from sepsis has a graph that is rather stagnant. Hence, if it is possible for us to differentiate the classes, it is also possible to deploy machine learning algorithms to model the behavior.

It is also helpful to find out how far the time dependency of the data persists. This can be done by calculating the correlation between the original time series data and a shifted version of the time series data.



```
shifted by 0 units, corr: 1.0
shifted by 1 units, corr: 0.8866974238170499
shifted by 2 units, corr: 0.8799472655820231
shifted by 3 units, corr: 0.7129565860291517
shifted by 4 units, corr: 0.5795904558938239
shifted by 5 units, corr: 0.47596027901184756
shifted by 6 units, corr: 0.4531261836478303
shifted by 7 units, corr: 0.45645218941888893
shifted by 8 units, corr: 0.34703353448708346
shifted by 9 units, corr: 0.3314539349747529
shifted by 10 units, corr: 0.40115848672849364
shifted by 11 units, corr: 0.5299899178447784
shifted by 12 units, corr: 0.4309494563242052
shifted by 13 units, corr: 0.5402227386352949
shifted by 14 units, corr: 0.5112075057334854
```

Figure 15: The correlation between the original time-dependent data of a patient to a shifted version of the data

This gives us a rough idea of the number of lookback hours for training models such as LSTMs. The correlation certainly does not for too long according to the information in figure 15.

Missing Data Imputation

There are many ways to deal with missing values. If the number of missing values is negligible, then one can simply treat the missing values by filling it with the mean of the feature for a continuous variable, or the highest occurring category of a feature for categorical variables.

However, since the dataset has a high number of missing values, treating the missing values using the method as aforementioned may cause serious biases in the model that we are developing, and will certainly overfit to those values are they are all constants. Hence, a probabilistic model will be more applicable in this case.

A Gaussian Mixture Model (GMM) can be used for missing value imputation.

For more information about the implementation of GMM, visit the following link.

In Depth: Gaussian Mixture Models
The -means clustering model explored in the previous section is simple and relatively easy to understand, but its…
jakevdp.github.io

Simply speaking, a GMM is a linear combination of Gaussian distributed variables, and it takes on the following form:

$$p(x) = \pi_1 N(x|\mu_1, \Sigma_1) + \pi_2 N(x|\mu_2, \Sigma_2) + \ldots + \pi_K N(x|\mu_K, \Sigma_K)$$

where $\sum_{i=1}^{K} \pi_i = 1$

$$p(x) = \sum_{i=1}^{K} \pi_i N(x|\mu_i, \Sigma_i)$$

Figure 16: The magic behind GMM (extracted from my lecture notes), K = number of components

The challenge is solving for the parameters in the statistical model, which is more mathematically involved. Expectation-Maximization (EM) is the go-to method to find the maximum likelihood estimate of the parameters of the model.

The data imputation is modeled as such. Since the primary goal is to find SepsisLabel, for any missing data, the missing values are modeled from distributions of two Gaussian, which may look like the following:

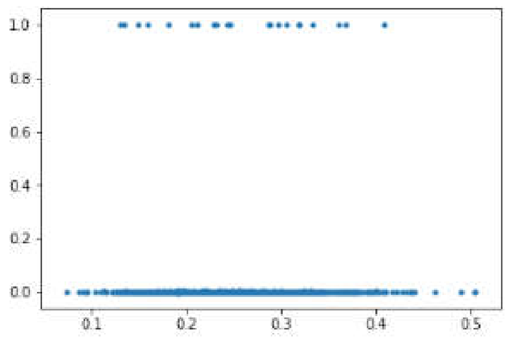


Figure 17: Visualization of GMM (think of having two Gaussian distributions within the two clusters of data in the diagram above)

Recall that there is about ***1.798 %*** of Class 1 labels in the entire dataset. It can be (and has been) verified that the distribution of data generated from the model does follow the same distribution.

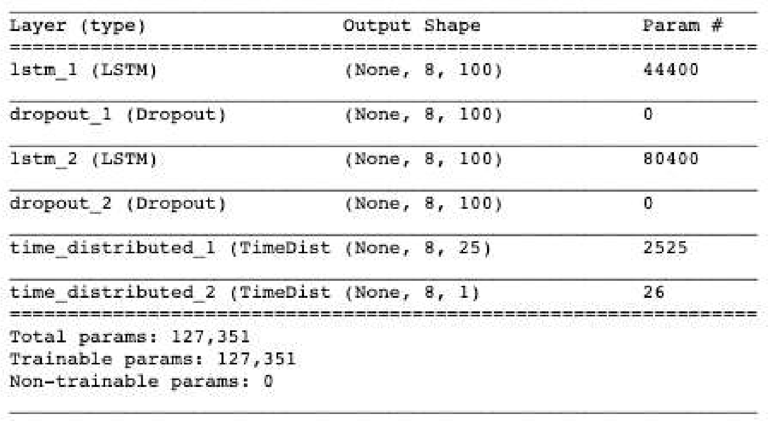
Note that since we are only interested in density estimation, it is possible to use higher values of K (more components) to generate better distribution. However, I would like to generate random values based on the distribution of the feature corresponding to the *SepsisLabel* to ensure consistency. Experiments have shown that even with ***K* = 2**, the data distribution can be captured as well.

For quick and dirty implementation, only the features with less than 15% of missing values are selected to be used for prediction.

Deploying Deep Learning Models

Since the rise of deep learning in 2012 after winning an ImageNet classification competition with a deep convolutional neural network, there are plenty of papers that are spawned in the domain. Deep learning models have been proven to be very useful to model different problems. For starters, since this is a time-dependent problem, the holy grail of time-series prediction is screaming Long Short Term Memory (LSTM).

For quick and dirty implementation, the following model is built with Keras.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 8, 100)            44400
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 100)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 8, 100)            80400
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 100)            0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 8, 25)             2525
_________________________________________________________________
time_distributed_2 (TimeDist (None, 8, 1)              26
=================================================================
Total params: 127,351
Trainable params: 127,351
Non-trainable params: 0
_________________________________________________________________
```

Figure 18: Deep Learning Model Architecture

To provide a more visually oriented diagram, the model is better visualized in the diagram below.

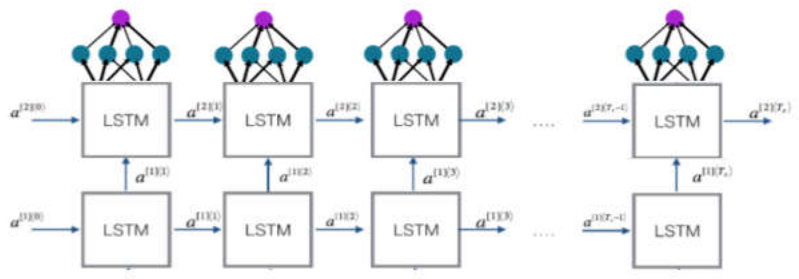
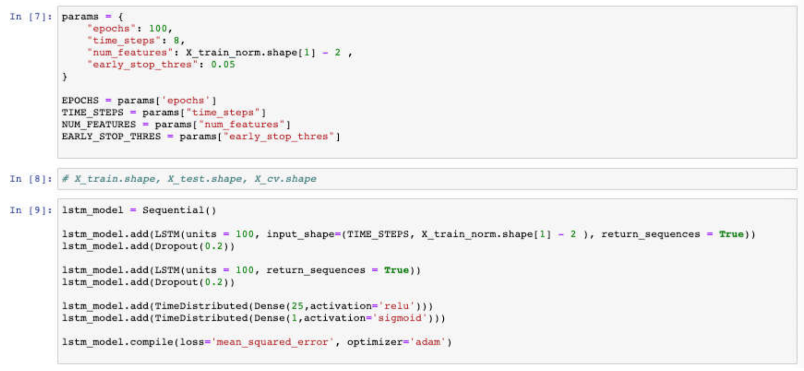


Figure 19: This model is generated by playing lego using the images online and the courses I have taken. A prediction of sepsis will be produced at each time step.

However, before training the model, there is just one more thing to do, data preprocessing.

1. Dataset is to be split into training, testing and cross-validation set with a ratio of ***60:20:20***, each taking on the same distribution as indicated in figure 7. The split is done with respect to patients, not to the total nuber of data points. (Each set should contain roughly ***7%*** of patients who will eventually suffer from sepsis). This can be achieved by applying the stratification method when splitting patients. **Note: The dataset should not be split based on the size of the dataset as it will not make sense to separate and randomize the time dependencies of each patient. It is very tempting to apply *train_test_split() directly*, but it is not correct.** Alternatively, one can use train_test_split() to split the patients instead of the dataset.
2. The training set is scaled to a range of between ***(0, 1)***. The same parameters used to scale the training set must be used to scale the testing and cross-validation set. This is because we are not supposed to have the knowledge on unforeseen data points in the testing and cross-validation set.

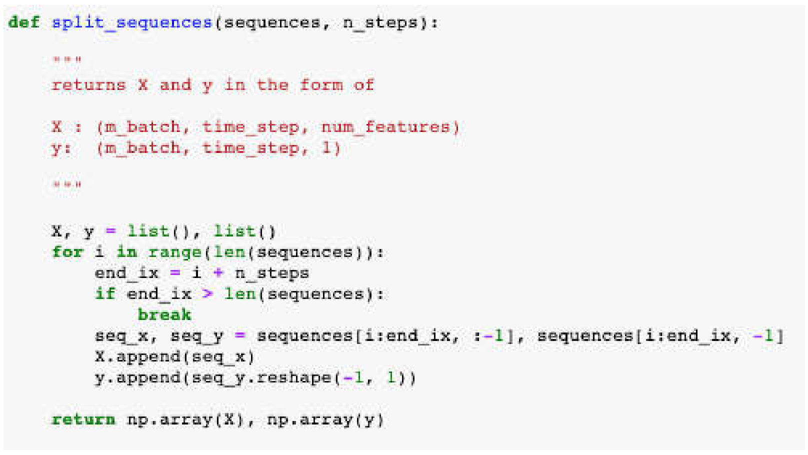
For a teaser, the model can easily be implemented using Keras with the code snippet below.



```
In [7]: params = {
            "epochs": 100,
            "time_steps": 8,
            "num_features": X_train_norm.shape[1] - 2 ,
            "early_stop_thres": 0.05
        }

        EPOCHS = params['epochs']
        TIME_STEPS = params["time_steps"]
        NUM_FEATURES = params["num_features"]
        EARLY_STOP_THRES = params["early_stop_thres"]

In [8]: # X_train.shape, X_test.shape, X_cv.shape

In [9]: lstm_model = Sequential()

        lstm_model.add(LSTM(units = 100, input_shape=(TIME_STEPS, X_train_norm.shape[1] - 2 ), return_sequences = True))
        lstm_model.add(Dropout(0.2))

        lstm_model.add(LSTM(units = 100, return_sequences = True))
        lstm_model.add(Dropout(0.2))

        lstm_model.add(TimeDistributed(Dense(25,activation='relu')))
        lstm_model.add(TimeDistributed(Dense(1,activation='sigmoid')))

        lstm_model.compile(loss='mean_squared_error', optimizer='adam')
```

Figure 20: Keras implementation of the model

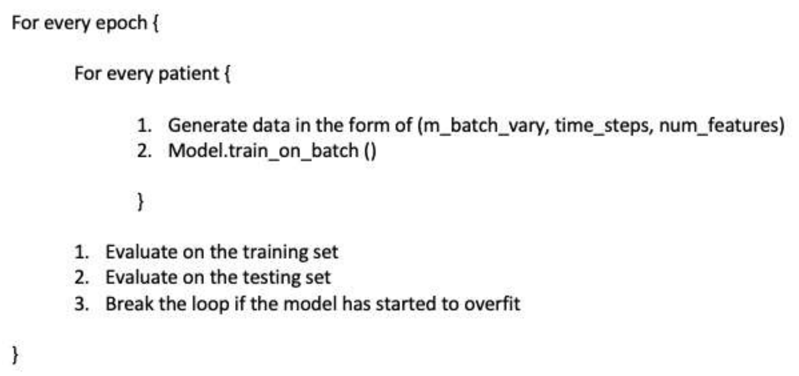
The Keras framework expects the input to be in the form of ***(m_batch, time_steps, num_features)***. This can be achieved using the helper function below.



```
def split_sequences(sequences, n_steps):
    """
    returns X and y in the form of

    X : (m_batch, time_step, num_features)
    y:  (m_batch, time_step, 1)

    """

    X, y = list(), list()
    for i in range(len(sequences)):
        end_ix = i + n_steps
        if end_ix > len(sequences):
            break
        seq_x, seq_y = sequences[i:end_ix, :-1], sequences[i:end_ix, -1]
        X.append(seq_x)
        y.append(seq_y.reshape(-1, 1))

    return np.array(X), np.array(y)
```

Figure 21: Data preparation into the right format. The function expects the entire data points of a single patient, and output the processed data in the desired format.

However, the batch size of each patient is varying. This can be solved by using the ***Model.train_on_batch()*** method of in the Keras framework. The pseudocode takes the following form (the implementation is done in Python).



```
For every epoch {

        For every patient {

                1. Generate data in the form of (m_batch_vary, time_steps, num_features)
                2. Model.train_on_batch ()

                }

        1. Evaluate on the training set
        2. Evaluate on the testing set
        3. Break the loop if the model has started to overfit

}
```

Figure 22: Pseudocode to train the model

The accuracy of the model can, of course, reach more than 90 % easily. This is due to the fact that the model is highly biased towards class 0. However, as Andrew Ng always preaches, " we first implement something quick and dirty, and then we iterate ".

The evaluation metrics should, of course, be changed. The competition does have its customized evaluation metrics, which rewards classifiers that predict sepsis between 12 hours and 3 hours before the patient suffers from sepsis and penalizes classifiers which do not predict sepsis or more than 12 hours before sepsis happens.

The code will be uploaded once the competition is over. ***Stay tuned*** to more posts about better uses cases of the cross-validation set to find out the optimum parameters (Bayesian Optimization, Random Searching, etc..) and better deep learning models such as using Attention, Encoders for feature extraction, or even Generative Models such as GANs to generate more data to reduce the biases towards class 0.

Early Detection of Sepsis Using Physiological Data Source: Deep Learning on Medium karan sindwaniJul 5What is Sepsis ?Sepsis is a potentially life-
5. July 2019
Similar post

Machine learning y pacientes críticos Original article can be found here (source): Artificial Intelligence on Medium Machine learning y pacientes
14. April 2020
Similar post

Doctors Use Big Data to Cut Sepsis Down to Size
Source: Communications of the ACM - Artificial Intelligence By National Institute of
19. February 2020
Similar post

«

« I AM VERY MUCH EXCITED TO SHARE THIS WITH YOU ALL THAT I HAVE BECOME A REDHAT ENTERPRISES LINUX 8…

TECHNICAL LEARNING OUTCOME—DAY 8 TO DAY 22 »