


Early Detection of Sepsis from Real-Time Patient Vitals

CSE 6250 Spring 2018
Adam Hedges, Hazel John



Motivation for advanced warning of sepsis infections

More than 1.5 million people acquire a sepsis infection each year in the United States alone*

More than 250,000 people die from sepsis each year in the United States*

Roughly 1 in 3 patients who die in the hospital have a sepsis infection*

Due to the nature of the infection, it can be very difficult to diagnose with sufficient lead time

*Statistics captured from <https://www.cdc.gov/sepsis/>



A machine learning/big data approach to predictive classification

Our experiment is based on the *InSight* predictive model, developed by Dr. Thomas Desautels et al (reference available in the paper)

We intend to test the efficacy of using real-time patient vital signs, captured from hospital EHR systems, to build a predictive classification model for the onset of a septic infection

Our model has been developed against the MIMIC-III database, available openly through the MIT Lab for Computational Physiology

Apache Spark 2.3, Spark SQL, Spark DataFrames, and Hadoop are used to pre-process data files from the MIMIC-III database and generate feature sets

scikit-learn is used to validate, train, and test machine learning classifiers against these feature sets

Flow of data

1) Load MIMIC-III data files

Load the PATIENTS, ICUSTAYS, CHARTEVENTS, PRESCRIPTIONS, and MICROBIOLOGYEVENTS CSV files

Populate Apache Spark DataFrames with Scala case classes for each record type

Filter DataFrames using Spark SQL, then persist to disk

2) Generate features

Index dates are calculated based on the suspicion of infection

The relevant patient vitals are captured at 1-, 2-, 4-, 6-, and 8-hour prediction windows

Resulting files are persisted to disk in LIBSVM format

3) Classification

Feature files for each prediction window are split into training and testing sets

The scikit-learn Python library provides the random forest classifier used for predictive modeling

Results are evaluated via standard scoring metrics and matplotlib charting



Working with Spark

Spark DataFrames, consisting of Scala case classes, provide an excellent method for dealing with extremely large data sets

Familiar languages like SQL can be used to work with Spark DataFrames, providing a streamlined development experience

Spark manages distributed cluster jobs, so that processes affecting large data files can be run efficiently on commodity hardware



Working with Spark

Example: Filtering and persisting CHARTEVENTS

```
/* Filter registered data table */
val chartevents_filtered = ss.sql("SELECT HADM_ID, SUBJECT_ID, ICUSTAY_ID, ITEMID, CHARTTIME, VALUENUM " +
    "FROM CHARTEVENTS WHERE CHARTEVENTS.ITEMID IN " + vitals +
    "AND CHARTEVENTS.ERROR = 0 " +
    "AND VALUENUM IS NOT NULL " +
    "AND CHARTEVENTS.ICUSTAY_ID IN ( " +
    "SELECT ICUSTAY_ID FROM ICUSTAY_IDS)")

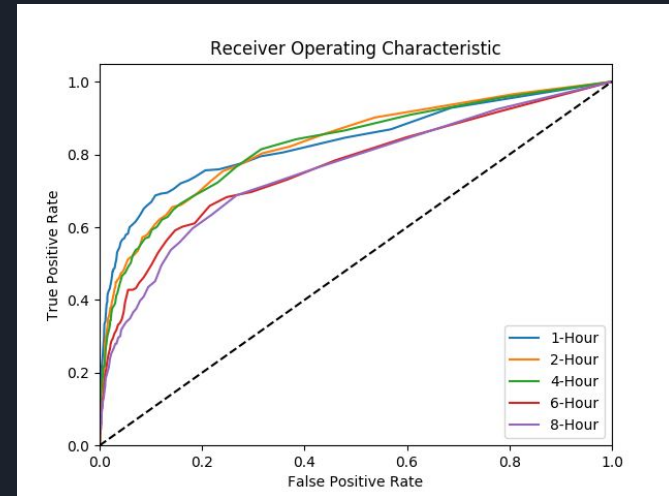
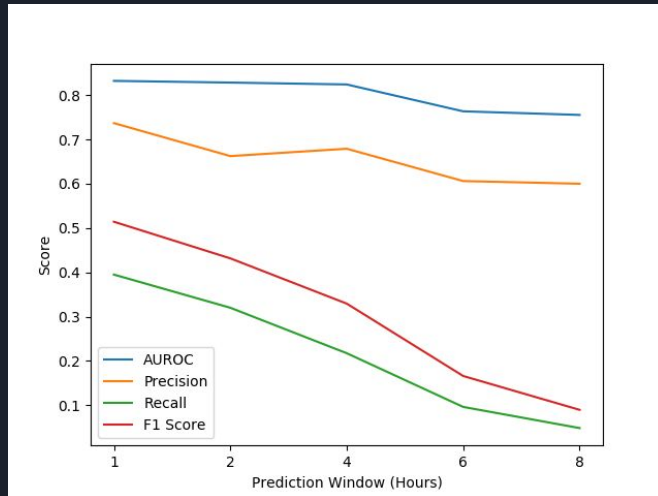
/* Convert to RDD */
val chartevents = chartevents_filtered.rdd.map(row => ChartEvents(row.getInt(1), row.getInt(0), row.getInt(2),
    row.getInt(3), row.getTimestamp(4), row.getDouble(5)))

/** Store to reduce processing time in subsequent runs */
ParquetUtils.saveDataFrameAsParquet(ss, chartevents.toDF(), saveDir+"/chartevents")
```

Classification results

At 1- and 2-hours prior to the suspicion of infection, predictive outcomes are relatively strong

A noticeable decrease in accuracy can be seen greater than 2-hours prior





Conclusion

There is always something to improve

Feature selection and construction is much more important than the classification method

Predictive classifiers have been used successfully in practice, though a key hurdle can often be acceptance of the medical staff themselves