A reference architecture for using big data and machine learning to improve patient care with the Open Source Trusted Analytics Platform (TAP)

An In-depth technical discussion and accompanying code can be found at:
https://github.com/ProKarma-Inc/TAP-readmissions-reduction

1. The Problem

To promote quality health care for Americans and reduce hospital readmissions, the Affordable Care Act (Section 3025) established the Hospital Readmissions Reduction Program (HRRP). The program instructs the Centers for Medicare & Medicaid Services (CMS) to penalize hospitals that have higher-than-expected readmissions for specific clinical conditions. In addition to complying with HRRP, hospitals have other financial incentives for reducing statistically high readmission rates too — they incur unnecessary costs when patients are readmitted for conditions that could have been addressed during the patient's initial stay. As a result, hospitals need a reliable and efficient way to reduce readmission rates.

2. Executive Summary

The availability of large patient datasets and powerful computational resources gives data scientists the ability to find meaningful patterns in patient readmission data. With this information, hospitals can build predictive models to identify patients most at-risk for readmissions. This enables practitioners to apply an appropriate intervention before the patient is discharged, ultimately reducing the risk that he or she will return for readmission. To address this growing challenge, ProKarma's Analytics team built a reference architecture for using big data and machine learning to improve patient care with the Open Source Trusted Analytics Platform (TAP). This document highlights the following:

Solution Overview: TAP packages standard open source tools such as Cloudera Hadoop, Apache Spark, Jupyter Notebook, Docker and Cloud Foundry to create an integrated platform to allow data scientists to quickly develop predictive models from large datasets and then deploy those models for use in applications.

Reference Architecture: Multiple data sources can be combined to train a predictive model with high accuracy on assessing likelihood of patient readmission. The model can be deployed as a service on TAP so that its predictions can be consumed by other applications (see Figure 1).

Reference Implementation: ProKarma's specific implementation of the reference architecture is based on the open source MIMIC-III medical dataset, complete with detailed explanations, walkthrough and code available at https://github.com/ProKarma-Inc/TAP-readmissions-reduction .

Adoption Plan for a New Implementation: This explores how any hospital can use the reference architecture to create an implementation with its own data and existing applications on TAP.

Reducing Readmissions Case Study: A case study (which inspired this reference architecture) demonstrating how Intel helped a large hospital group successfully implement a readmissions reduction program using big data and machine learning.

3. Using the Reference Architecture

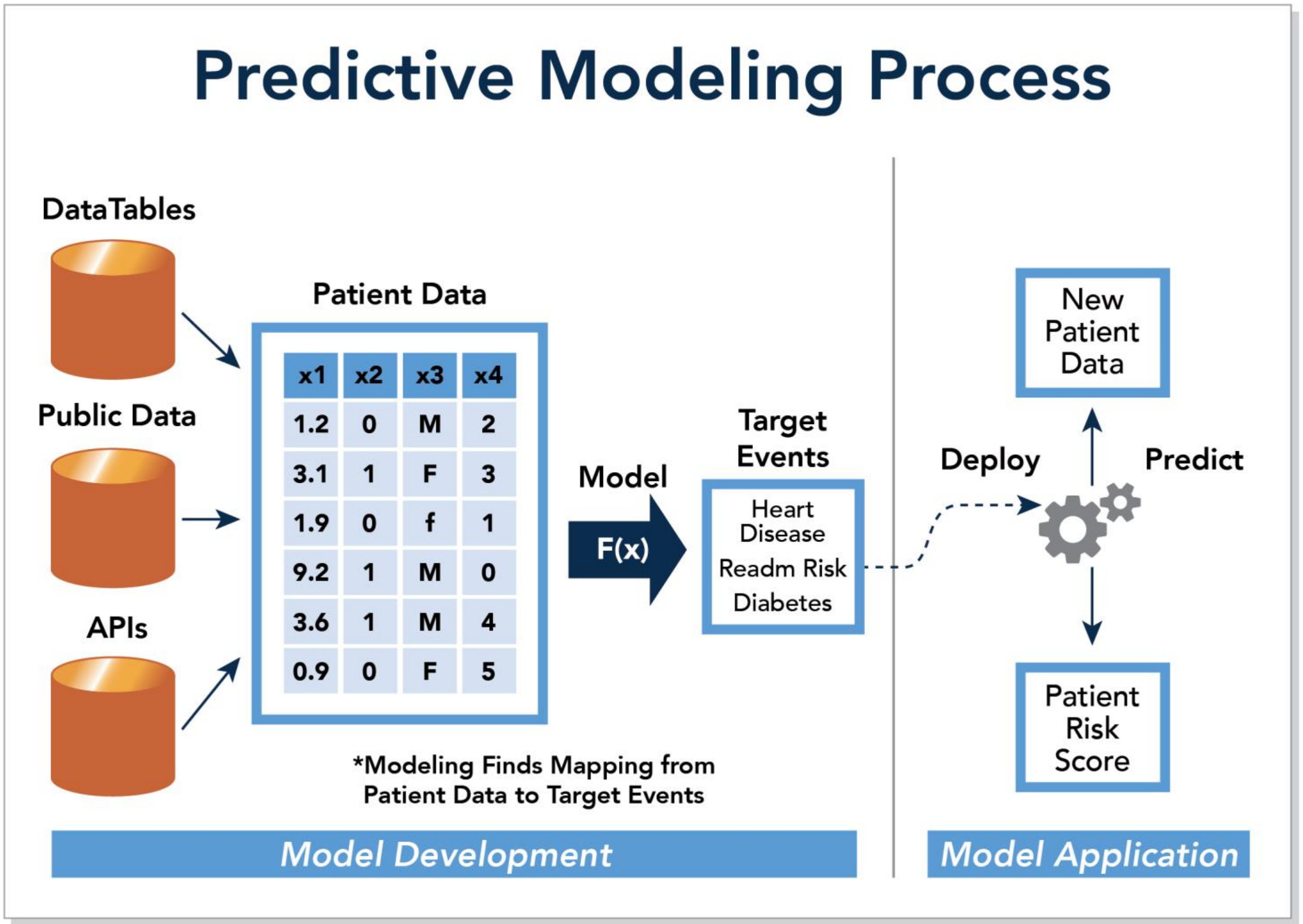

Figure 1: Reference architecture for creation and deployment of a predictive model to assess patient readmission risk.

Figure 1 illustrates the reference architecture that uses a predictive modeling process where patient electronic medical record (EMR) data is combined with multiple data sources, such as census and socio-economic data, to form a rich picture of a patient. With the right inputs, data scientists can create predictive models that learn the relationships between patient data and their propensity for different conditions, e.g. heart disease or risk of early readmission.

The reference architecture is meant to be customized to a specific hospital, creating a unique implementation. The reference implementation (see Section 4 and the GitHub repository listed) includes downloadable code and detailed documentation. Specific discussion and considerations for a hospital's individual implementation can be found in Section 5.

In general, once a predictive model has been created and validated, it can be deployed as a cloud-based service that allows the model's predictions to be consumed by other applications. For example, discharge planning software can pass a list of patient IDs to the model and receive a score that indicates the readmission risk for each patient. Once high-risk patients have been identified, their EMRs and discharge plans can be evaluated to address any appropriate risk factors. The model serves as a cognitive aid to assist hospital staff in identifying high-risk patients who may have otherwise gone unnoticed.

4. What the Solution Contains — The Reference Implementation

The reference implementation (Figure 2) utilizes the core TAP technologies, e.g. Cloudera Hadoop (CDH), Apache Spark, Jupyter Notebook, Docker and Cloud Foundry in the following process:

1. Historical data and patient records are stored in the CDH cluster.
2. Jupyter notebooks are created in Docker containers which enables data scientists to conduct collaborative and reproducible analysis.
3. Apache Spark — the big data computing engine — is utilized on the CDH cluster to analyze large distributed datasets.
4. Apache Spark's Machine Learning Library is used to train and validate the predictive model.
5. Cloud Foundry is used to package the predictive model and deploy it in the TAP cloud as an API service.
6. Cloud Foundry is also used to create an application that displays prediction results in a visual manner to facilitate comprehension by medical staff.

See the GitHub repository at https://github.com/ProKarma-Inc/TAP-readmissions-reduction for more details.

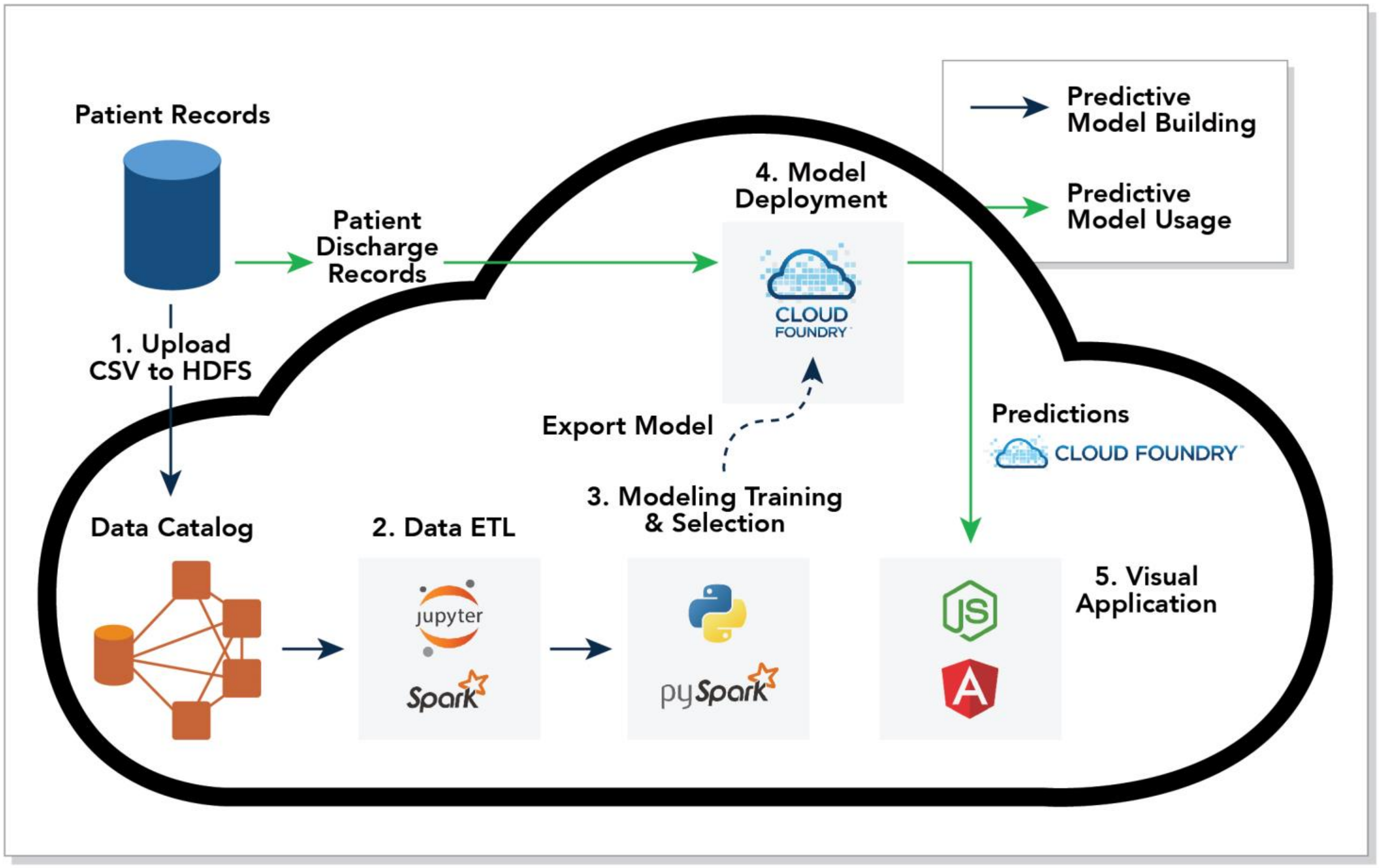

Figure 2: Logical relationship and key technologies used in this reference implementation.

5. Adoption Roadmap — Creating a New Implementation

This adoption roadmap outlines how creating a unique implementation of the reference architecture differs from the specifics of the reference implementation in Section 4.

The adoption roadmap for this solution consists of five essential steps:

1. Identify the organization's relevant data and load it into the TAP cluster.
2. Explore, process and engineer features for use in predictive modeling.
3. Pick performance criteria and train a predictive model accordingly.
4. Deploy the predictive model as an API that can be used by another application.
5. Build an application that incorporates the model predictions into the workflow.

1. Identify the organization’s relevant data and load it into the TAP cluster.

At a minimum, a hospital’s admission records are required to identify which patients were readmitted within a given time frame. Other sources of data, such as demographic information, electronic chart data and comorbidity records, can further enrich the patient readmission data, boosting model performance. Choosing which data to incorporate into a model can be as much a creative effort as it is an investigative one, and should be treated as an iterative development process. Subject matter experts (SMEs), IT professionals and data scientists should be involved in this initial phase. Data can be loaded and stored in TAP in many forms that lend themselves to the individual preferences and needs of an IT and analytics team. CSV files and Hive tables are two storage formats that facilitate using big data tools, such as Apache Spark.

2. Explore, process and engineer features for use in predictive modeling.

This step involves creating the data pipeline that takes the data from its original source and prepares it for modeling. This step includes conducting exploratory data analysis (EDA) to learn the structure of the data, cleaning any dirty or missing data and identifying what data fields will be useful for modeling. A significant effort in this step is feature engineering – the creation of new data features from pre-existing ones. Feature engineering is best done when SMEs, IT professionals and data scientists come together to brainstorm and discuss novel ideas and courses of action. In most cases, socioeconomic, demographic data and low-dimensional features, such as height, weight and age contain the majority of the predictive signal.

3. Pick a performance criteria and train a predictive model accordingly.

No model is perfect and tradeoffs must be made since some patients will be incorrectly flagged by the model and other high-risk patients will be missed entirely. Therefore, an analysis must be done that considers the cost of false positives and false negatives – administering unnecessary medical care or missing a high-risk patient – and the benefit of true positives, reducing readmissions. The stakeholders in this discussion are the healthcare professionals, operations planning staff and data scientists. With appropriate performance criteria the data scientist can train and validate a model according to the specified criteria. The area under the receiver operating characteristic curve (AUC-ROC) is best suited for delivering optimal results on this particular problem.

4. Deploy the predictive model as an API that can be used by another application.

Once a model has been created it must be deployed in a format that enables other applications to consume the predictions. From a development and integration standpoint, the simplest way to do this is to package the model as a service where it can be called like a REST API. Creating the model as a service gives application developers tremendous flexibility in how they choose to utilize the model, allowing them to only worry about the input and output of the model and ignore the internal details of how the model works.

5. Build an application that incorporates the model predictions into the workflow.

With the model available as a service, it is necessary to incorporate the prediction results into an existing workflow to give medical practitioners a way to easily use the model's predictions. Since the output of a model is just a number representing a risk score for readmission, this reference architecture builds a lightweight app that allows visualizations of the predictions and patient data. It is often helpful to provide contextual data to go with a given prediction so hospital staff can view the model's prediction within the broader context of the overall population. The model-as-a-service approach is modular and flexible enough to incorporate into different applications as needed to fit an organization’s requirements.

6. Solution Background — A Case Study in Reducing Readmissions

This reference architecture was inspired by a pilot program implemented by Intel and Cloudera. The program set out to use predictive analytics to reduce readmission rates within a large hospital group.

Intel data scientists took historical patient data and combined it with socioeconomic data, such as housing prices and health services, in the surrounding area. With this enriched dataset they trained a random forest predictive model that enabled doctors to pinpoint which patients were a high readmission risk. With this information, hospital staff administered additional care to identify any shortcomings in the treatment and discharge plan, thereby reducing overall readmission rates.

By using the predictions from the analysis, the hospital group received the following benefits:

1. Reduced 6,000 occurrences of patient readmission.
2. Avoided $4 million in potential Medicare penalties.
3. Saved approximately $72 million in medical costs.
4. Improved hospital ratings by lowering readmission rate and increasing patient satisfaction.
5. Achieved more efficient utilization of resources by focusing on high-risk patients.

One of the unintended benefits of implementing this solution was more efficient utilization of resources. Specifically, the increased quality of care provided to the identified high-risk patients during their initial visit freed up resources that enabled the hospital group to help an increase of 300-500 percent of patients.

Implementing the reference architecture on TAP allows the solution to be quickly developed and customized for an individual hospital or healthcare organization. The implementation of the model as a RESTful service allows it to be easily consumed by any third party applications.

Conclusion

This reference implementation serves as a blueprint for any healthcare organization to use TAP to quickly adopt the above-described solution, and begin reaping the same benefits.

For more information about deploying and/or using TAP and the reference architecture described in this paper, please contact:

Michael Hood	Tipton Loo	Julie Maas
Lead Data Scientist	VP Analytics	Business Development Manager
Analytics Group, ProKarma	Analytics Group, ProKarma	Analytics Solutions Group, Intel
mhood@prokarma.com	tloo@prokarma.com	Julie.maas@intel.com