Informatics for Health: Connected Citizen-Led Wellness and Population Health
R. Randell et al. (Eds.)

doi:10.3233/978-1-61499-753-5-181

Fast and Efficient Feature Engineering for Multi-Cohort Analysis of EHR Data

Michal OZERY-FLATO[1], Chen YANOVER, Assaf GOTTLIEB, Omer WEISSBROD, Naama PARUSH SHEAR-YASHUV, and Yaara GOLDSCHMIDT
Healthcare Informatics Department,
IBM Research - Haifa, Israel

Abstract. We present a framework for feature engineering, tailored for longitudinal structured data, such as electronic health records (EHRs). To fast-track feature engineering and extraction, the framework combines general-use plug-in extractors, a multi-cohort management mechanism, and modular memoization. Using this framework, we rapidly extracted thousands of features from diverse and large healthcare data sources in multiple projects.

Keywords. Feature engineering, electronic health records, longitudinal data.

1. Introduction

Feature engineering is the process of deriving informative features from data for a machine learning task. It requires tailoring to the source data, and is often assisted by domain knowledge. Following the increase in size and complexity of data, feature engineering has become a much more challenging, programmatically multifaceted, and time-consuming task. We describe a framework for enhancing the process of feature engineering from EHR data. This framework is based on extensive experience in analyzing such data, gathered during more than a decade of research on multiple disease areas and data sources.

2. Feature Extraction Framework

The basic elements of the proposed framework are *feature extractors* (shortened as *extractors*). Each extractor derives a set of features from the source data, or from features computed by other extractors. The features computed by an extractor are determined by a predefined, configurable set of parameters. Commonly-used extractors and their parameters are described in Section 2.1. The dependencies between different extractors are presented by a directed acyclic graph, termed *the extractors graph*. Typically, this graph has a single root, which represents the final output features matrix. The *extraction engine* orchestrates the entire extraction process; it instantiates and manages the extractors graph, and provides additional utilities that manage: (1) access to the data source; (2) parameters; (3) cached features; and (4) train-data statistics. The

[1] Corresponding author, Healthcare Informatics Department, IBM Research - Haifa, Haifa University Campus, Haifa, 3498825, Israel; E-mail: ozery@il.ibm.com.

input to the extraction engine defines both the extractors graph and the analyzed cohort. The latter is managed in a *cohorts repository*. Figure 1 illustrates the extraction framework and its components.

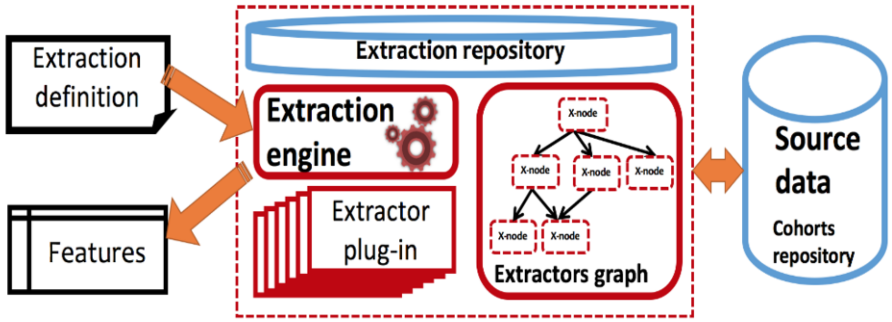

Figure 1. Overview of feature engineering framework.

2.1. Feature Extractors

Extractors are the building blocks of the feature extraction process. These plug-ins allow users to add new extractors without requiring a code change of the framework. Below we describe commonly used extractors, starting with those accessing a structured data source directly:

1. **Event extractor**: computes various aggregations on a sequence of events, where each event has a patient identifier, event name, time point/interval, and a value. A similar notion was previously described in [1].
2. **Numerical and Categorical Table extractor**: computes various aggregations on numerical and categorical data columns in a table, respectively. The input table has the following columns: patient identifier, time [optional], and one or more data columns.
3. **Time-to-Index-Date extractor**: computes various aggregations on the difference between date-columns in a table and the patient's index-date (index-dates will be discussed in Section 2.2).

For example, features like "average value of a <lab-test>" can be generated either by Event or Numerical Table extractors, and features like "indicator of a <diagnosis>" can be generated by either Event or Categorical Table extractors. The choice of extractor depends on the format of the source data. The Time-to-Index-Date extractor is used for computing "age" at index-date, or features like "time since first <diagnosis>". The sequence of events / data table, as well as the aggregation type and observation window, are configured by parameters. An SQL query is automatically built for each of these extractors, allowing the bulk computation to be performed in the database.

On top of the above-listed "atomic" extractors, the following extractors combine and manipulate features computed by other extractors:

1. **Group extractor**: concatenates output features of multiple extractor nodes.

2. **Family extractor**: executes a given extractor across a (configurable) list of parameter configurations and concatenates the resulting output features.
3. **Filter extractor**: filters out features (e.g., sparse features). The filtering type is configurable.
4. **Transformer extractor**: transforms features (e.g., algebraic transformations or second-tier aggregation of features). The transformation type is configurable.

One of the functionalities provided by the framework is the management of train-data statistics, to support extraction from new data, e.g., during the test phase. For example, an extractor that filters sparse features may keep statistics identifying the filtered features, so the same set of features will be filtered for new data; or an extractor that computes normalized features (e.g., weight percentile). Extractors that require train-data statistics are often *context-sensitive*, that is, features' vectors computed for the same individual may change for different cohorts. Extractors that depend on a context-sensitive extractor are expected to be context-sensitive as well.

2.2. Multi-Cohort Support

Often there is an interest in analyzing two or more related cohorts; for example, all patients having a disease; women who have the disease; and women in a certain age range having the disease. The framework manages all the cohorts in a *cohort repository*, which compactly represents each cohort by a list of quartets, termed *samples*. A sample is comprised of four fields: (1) patient identifier; (2) start-date; (3) index-date; and (4) end-date. The start-date and end-date bound and limit the overall time period for which data is analyzed for the patient. The index-date partitions this time period into two: (1) *baseline* period [start-date, index-date], and (2) the *follow-up* period [index-date, end-date]. In a prediction task, features are extracted from the baseline period, while the predicted outcome is extracted from the follow-up period. The framework allows for the same individual to appear more than once in a cohort, under the restriction that the corresponding samples have different index-dates.

2.3. Speeding Up and Tracking Feature Extraction

Data analysis, and feature engineering in particular, are typically an iterative process, where different iterations may involve overlapping feature sets and/or cohorts. Our framework provides an efficient modular memoization mechanism for the extraction process, which identifies and reuses previously computed features. The extraction engine automatically assigns each extractor node with an ID that uniquely maps to the node's configuration, cohort, and dependencies in the graph. This ID is then used for associating each extractor with its cached output features and train-data statistics. Multi-cohort analysis, including cross validation and bootstrapping, often involve sub-cohorts of a larger cohort. To optimize the extraction for sub-cohorts, the memoization of every extractor that is *not* context-sensitive is performed at the level of the larger cohort. Finally, the output features include the ID of the root extractor, which can be used for tracking the configuration of the entire extraction process, e.g., for reproducibility purposes. This ID should also be used in the test phase, so the framework can locate the train-data statistics of each extractor.

We extended the framework to support Big Data analysis using Apache Spark [2]. In this extension, the feature matrix is built in a distributed manner using Spark DataFrames, and the entire extraction process can be integrated into a Spark machine learning pipeline (MLlib [3]).

3. Framework Demonstration

We demonstrate the usefulness of our feature engineering framework on a real-world, de-identified (structured) claims dataset of 320K patients, for risk factor analysis across multiple cohorts. The considered risk was for urgent care visits during a one-year period. Previously, we effectively extracted thousands of features from this dataset for applications such as risk prediction [4] and contextual anomaly detection [5]. We evaluate the predictivity of the extracted features for the outcome and compare their importance for three chronic conditions: epilepsy, diabetes, and hypertension. We use the cohort of all patients as a reference.

The baseline period was defined as the first two years of the data (2002-2004) in all four cohorts. Each of the three chronic condition cohorts was defined by requiring that patients have at least one relevant diagnosis (i.e., hypertension, diabetes, or epilepsy) during the baseline period. The outcome was defined as having at least one urgent care visit during the one-year follow-up period (2004-2005). We extracted features from all the entities available in the data: diagnosis and procedure codes, pharmacy prescriptions, lab values, and patient demographics. We used HCUP-US Clinical Classifications Software (CCS) for grouping related ICD9 (diagnoses) and CPT (procedures) codes. We utilized the National Drug Code (NDC) Directory to cross-index each NDC to its corresponding generic drug components. Sparse features with less than 100 non-zero/valid values were filtered out. We imputed missing values in lab test features using the average of existing values and used Random Forest and its Gini-importance [6] for ranking the importance of features. The results of the analysis are summarized in Table 1. The extracted features were found to be informative in every cohort (out-of-bag AUC between 0.69 and 0.72). Inspection of the top-10 ranked features in each cohort revealed that risk factors for urgent care are dominated by utilization features, such as number of claims and number of prior urgent care visit. Risk factors not directly related to utilization included "Age" and features associated with the cohort (e.g., "Epilepsy; convulsion: count" in the epilepsy cohort).

4. Conclusion

We described a framework that enhances the process of feature engineering and extraction in a multi-cohort analysis of longitudinal structured data. Our framework offers: (1) efficient management of cohorts; (2) an extendible library of reusable feature extractors; (3) a modular memoization mechanism for accelerating the extraction of overlapping feature sets; and (4) a tracking mechanism for reproducibility of former extractions. The usability of this framework was successfully used in a variety of applications, including risk analysis [4] and detection of unexpected response to treatment [5].

EHRs contain longitudinal data on patients of various types: diagnoses, drug prescriptions, lab test results, medical procedures, and more. The dynamic and irregular nature of EHR data often requires engineering features that provide different summary statistics per patient, such as: the average value for each lab test result; or the proportion of days covered for each drug. Our extractors library enables simplified definition and reliable computation of such features for configurable time windows and cohorts. The repertoire of supported summary statistics is relatively large and is constantly growing.

One of the unique traits of the framework is its support of context-sensitive features. The framework provides infrastructure for managing train-data statistics for context-sensitive feature extractors. Moreover, its memoization mechanism, which optimizes extraction of sub-cohorts, automatically identifies context-sensitive extractors and properly handles their memoization.

We further supplemented our framework with a plug-in for Apache Spark [2] that extends our capabilities to analyze Big Data. Future work includes improving user experience via a graphical user interface that will further simplify the use of the framework, and an integrated visualization layer previously described in [5] that will allow inspection of defined cohorts as well as present longitudinal data of individual patients. Following all of the above, our framework offers an unprecedented, powerful means for analyzing and manipulating EHR data.

Table 1. Analysis results. #Positive = number of patients with positive outcome; TOP-10 = Non-utilization risk factors among top 10 ranked, excluding age

	Data statistics			Prediction model	
Cohort	Size	#Positive	#Features	AUC	TOP-10
All	320K	22.5K (7%)	949	0.69	1) Other upper respiratory infections: count
Hypertension	51.5K	5K (10%)	729	0.69	1)Essential hypertension: count 2)TRIG lab: average 3)CHOL lab: average
Diabetes	26K	2.5K (10%)	603	0.7	1)Diabetes mellitus w/o complication: count 2)TRIG lab: average 3)CHOL lab: average
Epilepsy	2K	0.27K (13%)	175	0.72	1)Epilepsy; convulsion: count 2) CHOL lab: average

References

[1] T. Tran, W. Luo, D. Phung, S. Gupta, S. Rana, R.L. Kennedy, A. Larkins, S. Venkatesh, A framework for feature extraction from hospital medical data with applications in risk prediction, BMC Bioinformatics. 15 (2014) 425. doi:10.1186/s12859-014-0425-8.

[2] Apache Software Foundation, Apache Spark, (2016). http://spark.apache.org/.

[3] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D.B. Tsai, M. Amde, S. Owen, others, MLlib: Machine Learning in Apache Spark, J. Mach. Learn. Res. 17 (2016) 1–7.

[4] H. Neuvirth, M. Ozery-Flato, J. Hu, J. Laserson, M.S. Kohn, S. Ebadollahi, M. Rosen-Zvi, Toward Personalized Care Management of Patients at Risk: The Diabetes Case Study, in: Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, New York, NY, USA, 2011: pp. 395–403. doi:10.1145/2020408.2020472.

[5] M. Ozery-Flato, L. Ein-Dor, N. Parush-Shear-Yashuv, R. Aharonov, H. Neuvirth, M.S. Kohn, J. Hu, Identifying and Investigating Unexpected Response to Treatment: A Diabetes Case Study, Big Data. 4 (2016) 148–159.

[6] L. Breiman, Random Forests, Mach. Learn. 45 (n.d.) 5–32. doi:10.1023/A:1010933404324.