Creating Machine Learning Datasets with Process Data in Steel Plants

Juan G. Sagasti[1]

[1]AUSTRALTEK LLC
800 Old Pond Rd St 706K, Bridgeville, PA
jsagasti@australtek.com

Keywords: Data Set, ETL, Machine Learning, Feature Engineering, Hot Rolling, Melt Shop

INTRODUCTION

Machine Learning algorithms applied to automation process data have several use cases in the steel industry such as defect prediction, smart sampling, anomaly detection, cobble prevention, and others.
This paper presents a collection of techniques developed for a large steel manufacturer in the US to curate the data coming from the automation layer in order to be usable in ML algorithms. Some of the processes involved are: Traceability, label identification, sample definition, feature extraction, time-based or length-based data aggregation, missing data handling and partial data import strategies.

MOTIVATION

For quality and efficiency reasons steel plants usually record most of the data generated by their automation layers during the several stages of the steel process. The data is stored for long periods of time from 2 to 10 years in different formats and structures. These data repositories are traditionally used for investigating customer claims -*defect forensics*- or perform studies to improve the plant's efficiency.

As stated in previous works [1] [2] a large number of efficiency and quality problems of the steel industry can be addressed using existing process data and AI algorithms -in particular Machine Learning ones- but the effectiveness of these techniques is highly dependent on a high quality and homogeneous dataset.

Because industrial plants are always evolving adding new product types, new equipment and new sensors, the dataset is not static but it grows with time. A modularized and easy to maintain data import *process* needs to be implemented to keep up with new data structures and data points. This process is generically called ETL (Extract – Transform – Load)

DEFINING THE SAMPLE OBJECT

A Machine Learning algorithm operate on a single 2D dataset that is composed of rows and columns similar to a spreadsheet. The rows are called *samples* and the columns are called *features*. If the algorithm is *supervised*, one of the features -the one that carries the property that the algorithm will predict- is called *label*

Depending on the problem that the ML algorithm is trying to solve, a decision must be taken to define the *sample object* -the data unit that corresponds to one row- this decision is one of the first that must be taken and is based on data availability and required accuracy of the predicted values. A table of typical sample object definitions for different steel related ML problems is shown below:

Problem	Type	Sample Object
Prediction of defect in coils	Supervised Binary Classification	Longitudinal sample of 1 meter of sheet
Prediction of defect in slabs	Supervised Binary Classification	Longitudinal slice of 5 cm of slab
Estimation of Heat Temperature	Supervised Regression	A heat
Thickness of ladle refractory	Supervised Regression	A ladle
Detection of Outliers	Unsupervised K-Means	A coil / slab
Prediction of Cobbles	Supervised Binary Classification	Slice of 5 minutes of process data
Prediction of Breakout	Supervised Binary Classification	Slice of 20 seconds of process data
Detection of Hot Spots	Supervised Binary Classification	Collection of simultaneous 2D images

DATA SOURCE TYPES

A typical data set is obtained from multitude of data sources, one useful classification of them is by the structure type, depending on this the extraction and transformation steps require more or less work. Data sources can be classified into:

Relational data
These data sources are the typical SQL databases. They represent objects, their attributes and their relationships. Its main data representation paradigm is a table in which each row maps to a physical or logical identifiable object such as a Heat, Sample, Billet or Slab. The columns of these tables will be typically mapped into features on the final dataset.

Time Based data
These data sources are also known as historians. They store the evolution in time of scalar variables. Its main representation is a timestamped sequence of values, one for each variable tracked. The values are typically aggregated, grouped by the sample object before entering into the final dataset.

Length Based data
This type of data structure is very common on rolling mills. Is analog to Time Based data but instead of timestamps the data is organized around the pair [Product Number, Length]. Most quality-related ML problems require sample objects of this type and if it is not available a procedure must be put in place to generate it using the ubiquitous Time Based data.

The following figure shows a Time Based temperature profile of a coil transformed into Length Based using the metallurgical length as index. We start with the curves for one coil $Temperature(t)$ and $Length(t)$, and as $L = Length(t)$ is usually a monotonic function, the inverse function $t = Time(Length)$ can be obtained. By composing both we obtain:

$$Temperature(L) = Temperature\big(Time(L)\big)$$

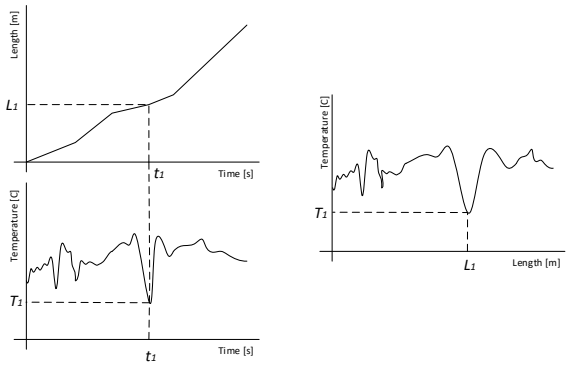

Figure 1 - Conversion from time domain to length domain

Other Rich Data
Other types of rich data such as pictures and 3D profiles are typically stored as binary chunks using keys for traceability/identification. The ETL process must implement some form of feature extraction / data aggregation to use this data in the final dataset.

LINKING DATA SOURCES

In the case of quality related problems, the dataset usually is constructed from several data sources that are linked through *product traceability*, a concept that is important on its own merit and can be defined as the ability to document all physical transformations that a material went through during its process until is delivered to the customer.

One of the distinctive features of the steel processing is that it starts with a large homogeneous quantity of steel (heat) and progressively generate smaller sub-products (slabs, coils/plates, etc.) from it. Data that represents that

process tend to be organized in hierarchical layers and when that structure is flattened to build the dataset, the parent data (in our case the Heat data) is repeated in all related samples.

AGGREGATING DATA

The granularity of the available data can exceed the scope of the sample object and in those cases more than one data points correspond to one sample. In these cases, the *subsamples* of the set need to be aggregated to be included in to the dataset. For example, consider a sample object that represents a slice of 10 cm of slab and a time-based high-speed "Caster Mold Level" signal. For each slice there will be thousands of mold level sub-samples. A valid approach in this case is to calculate average, minimum, maximum and standard deviation for the subsample set and use those features in the final data set.

The process of aggregating requires the calculation of functions that are applied to all the subsamples of the set and that return one or more metrics that represent the set. Examples of generic aggregation functions are: sum, average, standard deviation, minimum and maximum. Non-generic domain specific aggregation functions can be for example: number of peaks, maximum time between two events, components of FFT.

The following example shows an oversimplified temperature estimation algorithm, the sample object is defined as a period between actual temperature measurements (lance) during the LMF process and the goal is to predict the current bath temperature. Since we have subsamples of 1 second for electric energy, lid position and argon flowrate we need to aggregate them in different ways to generate useful features:

Electric Energy: The aggregate is generated by a subtraction between last "Energy" value and first "Energy" val*ue*
Argon Volume: The aggregate is generated with a numeric integral of "Argon flowrate".
Lid Open: The aggregate is generated as the number of seconds the lid was open on the period
Delta T : The aggregate is the temperature difference, note that in this case the data source does not provide subsamples.

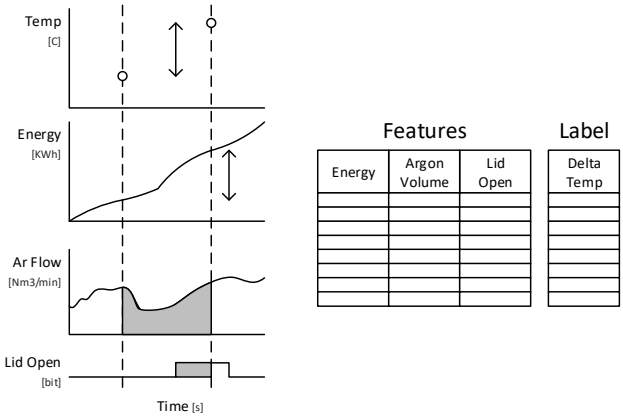

Figure 2 - Aggregation techniques in LMF temperature prediction

As a side note for this example: Once the training dataset is created and the model is trained, a real time sample is applied to the trained model to obtain a temperature estimation. The real time sample must have the same features as the training dataset and for that we need to perform the same data aggregation but this time the subsample set will be taken from the previous actual temperature taken to current time.

DATA CLEANING

As mentioned before, the quality of the data set affects the algorithm performance so steps must be taken to clean the data coming from the data sources before merging them into the final dataset.

The following is a list of typical data quality problems found in steel plant data sources according to [3] and their potential solutions:

Wrong Values

Data coming from instruments or data-entries must be filtered to eliminate values that are not physically possible, and that would affect the effectiveness of the ML algorithm. These values appear when instruments are broken, disconnected or coming from not properly sanitized manual data-entries.

It is recommended that these values are marked as N/A (not available) or NULL by a cleaning procedure right after the extraction phase, this approach allows for better modularization and maintenance in the long term. For most physical measurements a valid range can be obtained from instrument data sheets and/or process engineering practices, but in many cases the assumption that the distribution is normal or multi-normal holds and a simple histogram of values per column can be used to detect which features are more prone to generate these errors and adjust the procedure accordingly.

Types of Missing Data

As samples are built from more than one data source using traceability to link them together, it is possible that there is no matching record in all the data sources, yielding several features as NULL

Another source of missing data is instruments/automation recording absurd values that are filtered by the acquisition systems as described in the previous paragraph, leaving the values as NULL.

Yet another reason for missing data is the inexistence of the underlying value, consider a single stand reversing rolling mill where most product types require 7 passes but some of them require only 5 passes, the features associated with passes 6 and 7 will be NULL for those coils.

Most Machine Learning algorithms cannot work with datasets that have NULL values so actions are needed in order to fix them. The following is a summary of possible actions

Eliminate Samples

Ignoring samples with incomplete features is a valid approach as long as it does not introduce a selection bias in the problem that we are trying to solve and there are enough samples left for training the algorithm.

An example of a selection bias: In a hypothetical hot mill one can be tempted to eliminate all coils that don't have a value for "downcoiler temperature" feature, but taking that action would eliminate all unfinished coils from the dataset making the dataset completely useless for some uses such as detecting cobbles.

Eliminate Features

As a general rule, all features must be included in the dataset, but not all features contribute equally to the ML problem at hand as each feature portrays a potential *discriminative power*. This discriminative power has two components: One is independent of the ML problem and is related to the variance of the column, the less variance the less power, the extreme case being all samples having the same value for the feature yielding no discriminative power at all. The other component is the correlation with the label or ML problem itself, for example it is very unlikely that the feature "Scrap Basket Weight" affects the prediction of "Tensile Strength "

As a rule of thumb, only consider eliminating features affected by missing values if they don't have significant discriminative power.

Single Imputation

The last alternative is to fill the empty spots with calculated values in an effort to save the useful data while not affecting the dataset validity. This approach is supported by many software packages that offer a multitude of options such as Custom Substitution, Replace with Mean, Replace with Median, Replace with Mode and more complex multivariate algorithms such as Probabilistic PCA, a method described in [4].

The basic methods described are dependent on the Machine Learning problem to solve, it is advisable that if a dataset will be used for more than one problem, a version with missing data is kept and the data cleaning processes are performed on a per problem basis.

FEATURE ENGINEERING AND SELECTION

Steel plants have plenty of process data that can be directly converted to features and as a general rule all data should be considered valuable. In certain cases, we should consider using our domain knowledge to add synthetic features to improve discrimination and overcome ML algorithms limitations, and in other cases we must consider removing unnecessary features to improve training time and overall effectiveness.

Feature Engineering

The raw data coming from the instruments can be expanded by creating additional features with non-linear calculations among them. Consider the case of timestamps on a Reheating Furnace where we are trying to predict the final temperature. Instead of using timestamps we can "help" the decision algorithm by calculating the

difference in seconds between entering and exiting each furnace zone, although this operation is linear and would be eventually found in some form by a decent ML algorithm, it won't hurt to add it as a feature simplifying the dataset and saving calculation time.

Using the same example let's assume the slab entry temperature is the outside ambient temperature and we are not measuring it. We know that the timestamp carries valuable information, namely that there is a 40 C gap between summer and winter months but any ML algorithm will have a hard time figuring that out because is a non-linear function. We can create a new feature "ambient temperature" that express that difference either by linking a local weather database or just a lookup table with average temperatures per month.

Another source of additional features is the previous history. Machine Learning rely on finding an optimal estimation function $y = f(x)$ where x is the set of inputs of a sample, and each sample is considered independent i.e. there is no state being carried out from previous samples. In sample objects that are defined by consecutive periods of time, it is very common to add features from previous samples, or aggregations of them.

As an example of domain-specific feature engineering, consider the case of a breakout warning system that uses hi-resolution cameras on the caster platform to determine if a mold breakout is occurring by analyzing the glare. As with most image related algorithms the individual pixels of the image ROI are too numerous to be considered features and instead some type of domain specific indicators need to be created. In this case the individual pixel brightness is compared with a threshold and the resulting pixels are counted weighed with the distance to the mold walls providing a synthetic feature that yields high discriminative power.

Feature Selection

In an industrial plant features tend to be highly correlated between each other so a lot of redundancy is being provided by the data sources, making ML training inconvenient and slow. There are several techniques to reduce the dimensionality of the ML problem without using domain-specific knowledge. PCA analysis is one of the most used but when applied to our typical use cases it comes at the price of creating artificial features that do not have physical sense, hindering the analysis that sometimes is one important take-away of a successful ML project.

Another method suggested by [5] is "Forward Greedy Selection" that consists on starting with one feature and adding the most performant ones by evaluating them on the ML problem. That method has the advantage of creating a ranking of meaningful features that can be evaluated by process engineers to find the underlying physical model of the problem at hand.

ETL ARCHITECTURE

Division by Data Source and Domain

As mentioned above, the generation of a data set suitable for ML problems needs to be thought as a process and not a one-time task. The dataset generation action takes time and usually stresses out the plant databases. The process must involve procedures for gathering data from different data sources and those procedures need to be prepared to be modified and executed when new features are included or recalculated. It is advisable that the generation procedures are maintained separately by data source or by domain in order to execute only the ones that have changed when regenerating the dataset.

The following diagram represents the extraction and cleaning procedures installed in a large steel plant that generate a dataset that is used for different ML problems

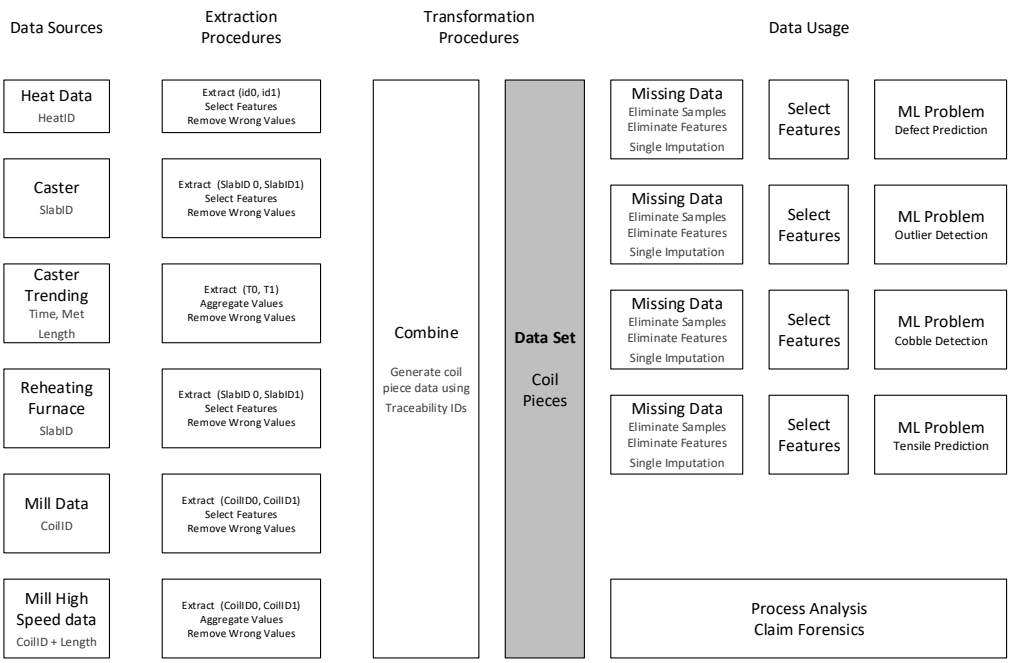

Figure 3 - Sample ETL architecture

In the example, if a new feature "Carbon Equivalent" needs to be included in the dataset, only the Heat Data procedure needs to be modified and executed.

Division by Time or IDs

Also, on the use cases mentioned the number of samples of the dataset can be very large and require a parallelizable architecture, in our previous example instead of running the procedures through all the historical samples, we can divide the samples by year/month chunks and run them in parallel. It is even better if the data sources offer consecutive unique IDs for the sample objects (in this case coilID – SlabID - Length) to avoid time differences in different data sources.

The final dataset itself can also be divided in chunks so depending on the ML problem we can select the amount and age of the data we are feeding it to.

CONCLUSION

In this paper we presented several design challenges that affect the ETL architecture for generating a dataset for Machine Learning projects for steel plants, such as multiple data sources linked by not always perfect traceability, possibility of using the same dataset for multiple problems, and the fact that a ranked feature set provides very useful information. We explored these challenges on the light of several steel related use cases and provide a description of a ETL architecture implemented in one large steel producer in the US.

REFERENCES

1. Santanu Ghorai ; Anirban Mukherjee ; M. Gangadaran ; Pranab K. Dutta: "Automatic Defect Detection on Hot-Rolled Flat Steel Products", IEEE Transactions on Instrumentation and Measurement, Volume 62 Issue 3. March 2013.
2. Sagasti J., Vazzana E., Coleman P., Rummel S.- Use of Machine Learning to assess defects in hot rolled coils. AISTech 2017 Proceedings
3. Erhard Rahm, Hong Hai Do. Data Cleaning: Problems and Current Approaches. University of Leipzig, Germany. http://www.betterevaluation.org/sites/default/files/data_cleaning.pdf
4. Tipping M. Bishop C. Probabilistic Principal Component Analysis. Royal Statistical Society, Series B 21(3), 611–622.
5. Shai Shalev, Shwartz, Shai Ben-David. Understanding Machine Learning: From Theory to Algorithms. 2014 Edition, 310-315