Paper ML09

Machine Learning for CDISC Specification Recommendations

S. Preston Burns, Rho, Durham, US
Brandon Welch, Rho, Durham, US

ABSTRACT

The creation of SDTM and ADaM variable specifications is a painful requirement for enjoying the benefits of CDISC standardization. Demanding large amounts of both statistician and programmer time, it is an expensive process that grows more costly as the quantity of required metadata increases. The design phase of the dataset specification process, in particular, appears ripe for automation. Previously, Rho began work to leverage historical specification data to generate recommendations for programmers entering ADaM specifications for new studies. Here we build on the text mining methods introduced in the previous work by applying creative data preparation, feature development, and machine learning models to the predictive task. These developments allow us to incorporate more inputs, and the interactions between them, in the recommendation of variable specifications.

INTRODUCTION

Creating sound specifications for SDTM and/or ADaM data sets is vital. However, depending on the programmer's setup (Microsoft ® Excel worksheets are a popular choice), the task requires manually editing and entering metadata. This approach of entering the data in a worksheet is error prone, laborious and lends itself to inefficiencies. Many of the specification definitions are highly repetitive (e.g., treatment emergent adverse event definition span almost all studies) and require low levels of critical thinking and subject matter expertise, making them ripe for automation. In a previous paper, Rho began looking at ways to leverage historical data to provide definition recommendations to programmers to save them time. We evaluated the frequency of use and various forms of string similarity as metrics to rank definitions and provide recommendations. The results showed promise and provided justification for further investigation [1]. However, our previous attempts used one piece of information at a time. A robust solution needed to incorporate multiple inputs and determine optimal weights for these inputs and the interactions between them. Machine learning was the natural next step.

In order to generate recommendations, we needed to rank potential definitions by the likelihood that a programmer would select them. To do this we employed *supervised* machine learning. In supervised machine learning, a model is trained to predict an outcome based on a set of inputs. In our situation, the outcome was whether or not a programmer would select the definition. By observing examples from historical metadata where definitions were or were not selected, the model was able to assign probabilities of selection to new definitions it hadn't seen before. Those are the probabilities we ranked to get our recommendations.

In this paper, we both outline and evaluate an experimental process for developing and providing variable specifications using machine learning.

OVERVIEW

The process we followed can be broken down into three main steps: data cleaning, data transformation, and modeling. The paper will follow this structure. It is visually depicted in Figure 1.

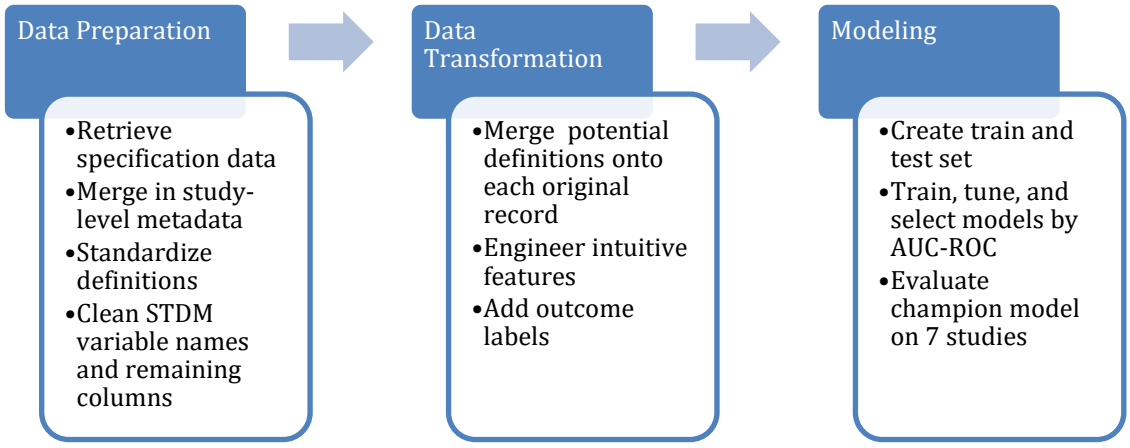

FIGURE 1. Process overview

DATA CLEANING

RETRIEVING THE DATA

To prepare these examples, we started by extracting all variable specifications from the last 12 years for 36 ADaM datasets (details in appendix A) from our metadata database. We retrieved the dataset name, variable name, study id, and study sponsor for each of these specifications. Using a second data source, we determined the Therapeutic Area and Indication for 71% of these studies, added that information to our dataset, and dropped the studies without it. While we lost some studies, this additional information seemed potentially helpful to the model in making predictions. The final extracted data contained 8,720 variables spread over 199 studies. There were 21,768 definitions that only occurred once across all of these variables. The bad news is, that's out of 46,425 total definitions. Yep, that's right, 47% of the definitions were completely unique and couldn't have been easily populated from historical data.

NOT AS UNIQUE AS YOU MIGHT THINK

One might expect the project to end right there – what's the use in trying to recommend past definitions for new studies, if the new ones have never been seen before? Well, here are three main points that make this less of an impossibility and more of a problem:

1) There's a wide variability in the uniqueness of variables

   Even with CDISC standards, there are some unique variables for almost every study and these account for a large amount of the uniqueness, while other variables like TRTEMFL are consistent across almost every study.

2) We don't need to predict every variable

   This isn't about perfection; it's about reducing workload. We will never get every variable, but if we can get at the repeated variables that make up the vast majority of studies, we will reap large efficiency gains.

3) They are not as unique as they seem

   Consider these two definitions for the variable TRTEMFL:

   "Y" if ADSL.TRTSDT <= ADAE.ASTDT <= ADSL.TRTEDT + 3 missing otherwise

   ="Y" if TRTSDT <= ADAE.ASTDT <= ADSL.TRTEDT + 3 =missing otherwise

   It's easy for us to see that those are all really the same definition, but the model would have no idea. In order for the model to know this too, we'll have to replace these with a single standard definition.

STANDARDIZING DEFINITIONS

Two main strategies were employed to standardize the definitions:

1) String Clustering
2) Regular Expressions

For the string clustering we used two very conservative algorithms implemented in the data munging application openRefine: Fingerprint and N-Gram Fingerprint [2]. These methods reduce text strings to their essential pieces and then use these representations to group similar strings together. A host of custom regular expressions ranging from replacing SAS® functions and expressions to recoding all forms of yes to Y were employed subsequently. Combined, these methods reduced the percentage of single-instance definitions from 47% to 36%. The variable and sponsor names were cleaned using similar techniques.

DATA TRANSFORMATION

THE PROPER FORMAT

With the data cleaned, we now needed to transform the data into a format conducive to generating recommendations. To replicate a definition being chosen from all possible options, we created a new dataset where each row represented a variable name and a pair of studies: the study that was being recommended for and a study that offered a potential definition. If 15 possible definitions had occurred for a variable name, then each study/variable name combination (a row from the original dataset) was replaced with 15 rows of paired studies and definitions. This is depicted below in figure 2:

Study	Variable Name	Historical Definition
A	AGEU	DM.AGEU
B	AGEU	= ‘years’
C	AGEU	DM.AGEU

Study	Variable Name	Historical Definition	Potential Definition	Potential Definition Study
A	AGEU	DM.AGEU	DM.AGEU	A
A	AGEU	DM.AGEU	= ‘years’	B
A	AGEU	DM.AGEU	DM.AGEU	C
B	AGEU	= ‘years’	DM.AGEU	A
B	AGEU	= ‘years’	= ‘years’	B
B	AGEU	= ‘years’	DM.AGEU	C
⋮				

FIGURE 2. Data transformation

Merging in the potential recommendations caused the dataset to go from 46,505 rows to 1,850,938 rows.

CREATING FEATURES

With our data in the proper format, it was time to derive meaningful features. “Features” is a machine learning term for the pieces of information the model uses to predict the outcome. Some of these features were relative in nature, like “Same Sponsor” or “Same Indication” and represented situations where the study we were recommending for and the recommended study both had the same sponsor and indication. Others were simply features of the recommended definition, like the recommended definitions’ popularity in the past. The creation of these relevant features, and the removal of irrelevant ones, made it easier for the model to learn how to recommend good definitions. The code for the final features is provided below, with *seed* representing the study recommended for, and *reco* the one recommended. For the rest of the paper, we will use these abbreviations for brevity.

Feature	Python Code
Popularity	`data.groupby(['reco_varname','reco_definition'])['reco_definition'] .transform('count')`
Same Sponsor	`np.where(data['seed_sponsor'] == data['reco_sponsor'], 1, 0)`
Same Therapeutic Area	`np.where(data['seed_thera_area'] == data['reco_therea_area'], 1, 0)`
Same Indication	`np.where(data['seed_indication'] == data['reco_indication'], 1, 0)`
Contains Variable Name in Definition	`[x[0].lower() in x[1] for x in zip(data['seed_varname'], data['reco_definition'])]`
Sponsor	`Transformed into dummy coded columns by pd.get_dummies(data)`

TABLE 1. Python code for engineered features

CHOOSING A TARGET

Remember, before we can start training a supervised model to assign definition probabilities, we must give it labeled examples. Right now our dataset contains information about each recommended definition and its relationship to the study and variable its being recommended for, but it doesn’t know which definitions are correct. In the future, these labels could be determined by which recommended definition the user selects from a dropdown, but for this initial training, we needed a stand in. A simple (1) if the reco definition is the same as the seed definition, and (0) if not, was sufficient. However, to avoid cheating, all rows where the reco study and the seed study were the same were removed. (Single study variables that were lost to this correction were accounted for in the final evaluation metrics). In real life, the model will only have access to definitions from past studies, so the training should be consistent with that. Building on the sample dataset from figure 2, you can see the new outcome column below in figure 3. The first row was deleted as expected since the recommended definition was from the same study.

Study	Variable Name	Historical Definition	Potential Definition	Potential Definition Study	Correct Definition
~~A~~	~~AGEU~~	~~DM.AGEU~~	~~DM.AGEU~~	~~A~~	~~1~~
A	AGEU	DM.AGEU	= 'years'	B	0
A	AGEU	DM.AGEU	DM.AGEU	C	1
B	AGEU	= 'years'	DM.AGEU	A	0
B	AGEU	= 'years'	= 'years'	B	1
B	AGEU	= 'years'	DM.AGEU	C	0
⋮					

FIGURE 3. Addition of outcome variable

MODELING

With the data curated, it was finally time to build machine learning models.

ENSURING GENERALIZABILITY

Before we started building the models we split the data into a training (80%) dataset and a test (20%) dataset. The training set was used for determining model parameters and hyperparameters using five-fold cross-validation. Final evaluation of each model was then performed on the test dataset to assess their recommendation performance and ability to generalize to unseen data [3].

CHOOSING AND TRAINING A MODEL

Developing a machine learning pipeline can be a time-intensive process. We chose to automate this by using the Python package TPOT which uses a genetic algorithm to find an optimal pipeline [4]. Area under the receiver operating characteristic curve (AUC-ROC) was used to determine the best model. A basic logistic regression model and the univariate popularity feature were included in the evaluation as baselines for comparison.

MODEL PERFORMANCE

	Popularity	Logistic Regression (Baseline)	Random Forest (TPOT)
AUC - ROC	.865	.893	.978

TABLE 2. Performance metrics on test dataset

A random forest model [5] was chosen by the TPOT algorithm to be the champion model. It exhibited a large improvement in ranking ability over both baseline comparators. This may be due to the fact that random forest models inherently determine simple interactions in the data, while a logistic regression would require explicit specification. For example, in the logistic model, having the same sponsor is always going to increase the likelihood that a recommendation will be selected. However, this isn't necessarily true; if a certain sponsor does wildly different studies all the time, trying to carry over definitions from old studies may be more harmful than helpful. The random forest model, on the other hand, will discern and build that nuanced relationship between the same sponsor and sponsor features on its own.

EVALUATION

While AUC was useful for selecting a model, we wanted the final evaluation to replicate a real world setting. To do this we selected 7 studies that the model was not trained on and generated variable specifications recommendations for the 36 target datasets. We evaluated the effectiveness of our models on the studies by calculating the percentage of variable definitions that the number one recommendation matched.

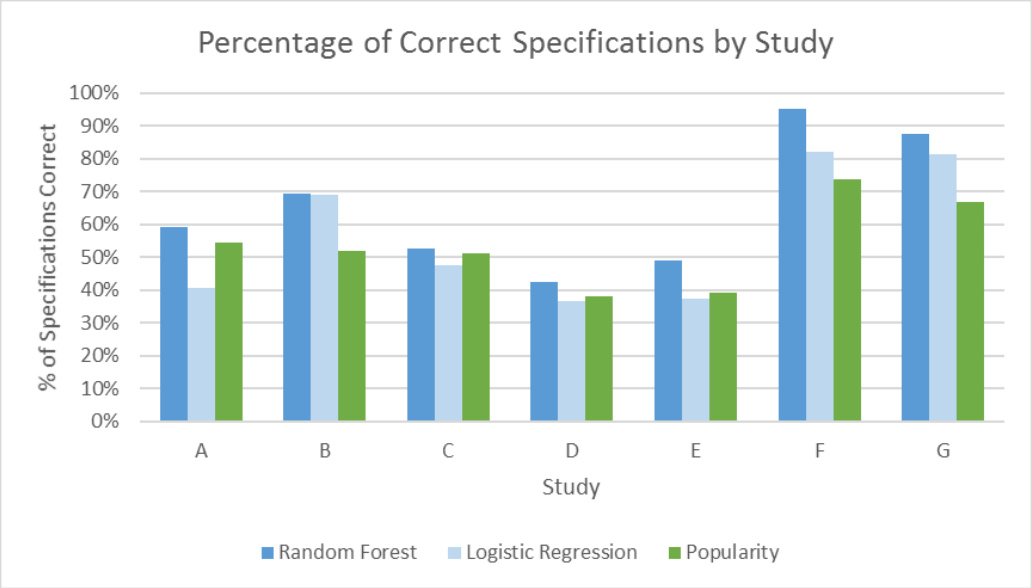

FIGURE 4. Percentage of correct variable specifications by study

As depicted in Figure 4, the machine learning models' recommendation performance hovered between 40-70% for the majority of studies, and exceled on two of the studies, correctly predicting 95% of the definitions on study F. Popularity was a strong recommender as expected, but the champion model was able to successfully leverage the additional information for noticeably better recommendations on every study except study C. This study was for a new sponsor and indication, so it makes sense that the methods would perform similarly given the absence of historical study-level information.

Next, we split the results up by dataset to evaluate how the models performed across different datasets.

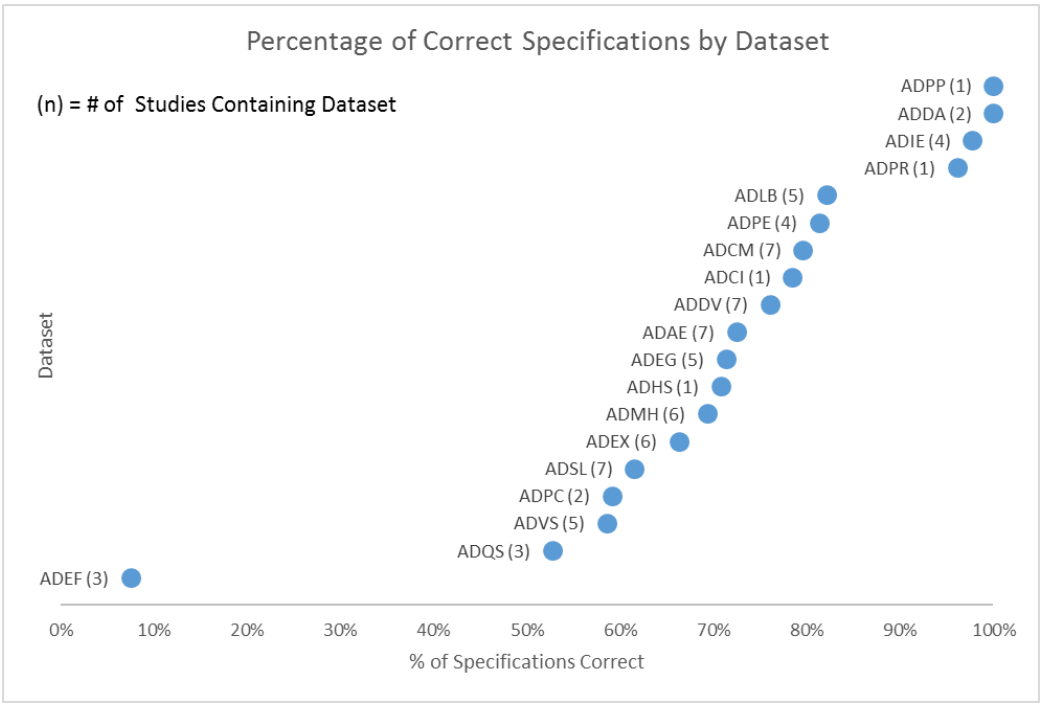

FIGURE 5. Percentage of correct variable specifications by dataset

From Figure 5, it is intuitive that ADEF had the fewest correct definitions. ADEF contains clinical trial efficacy endpoints which vary considerably by study; an oncology study's endpoints are defined much differently than an ophthalmology study. The study level ADSL data set falling slightly above 50 is intuitive too. This data set contains common variables such as RACE and AGE, which are pulled directly from SDTM, but also contains treatment definitions which vary by study. The higher scoring

data sets such as ADAE excel due to a high proportion of common definitions like AEREL (adverse event causality) which is typically defined similarly across studies and is directly pulled from SDTM.

DEPLOYMENT

There are a variety of ways this process could be productionalized, but creating a web service seems like a natural choice. To begin the process, a client would simply pass the seed definition's sponsor, variable name, dataset name, therapeutic area and indication to the web service via an API. The web service would then repeat the data transformation portion of the process described here: pair up the request with all the potential definition recommendations, derive the specified features, add on the outcome labels, and then use the chosen model to assign selection probabilities to each potential definition. These definitions would then be sorted by their probabilities and sent back to the client for displaying to the user.

FUTURE WORK

There is a lot of room for improvement here. Additional features, more careful modeling, and expansion of definition standardization methods, are all possible while maintaining the same broad process. Visual examination of the recommendation results revealed to us that the model is fairly conservative with its predictions, so there is a lot of room for further definition standardization work in particular. Additionally, a different model selection or optimization metric that is designed specifically for ranking like Normalized Discounted Cumulative Gain (NDCG) may assist with optimizing the model [6]. Ultimately, however, performance is limited by the fact that recommended definitions are only historical. In order to handle definitions that are more unique to a study, knowledge closer to that of the programmers would be required. This could come from CDISC metadata or text mining additional data sources.

The evaluation itself also has potential for clearer results. We only counted a model's definition recommendations as correct if the top recommendation was right. In reality, the programmer would be presented with the top 3 or more, and if the right definition was near the top, it'd be just as convenient as being in the first position. Additionally, definitions had to be nearly identical to be counted as correct. In practice, a "wrong" definition that requires only a few edits is still useful and time-saving for a programmer. Having a statistical programmer sit down and evaluate the use of this tool while programming their specifications could reveal that we are underestimating or overestimating our model.

CONCLUSION

Ultimately, the experiment was successful. We were able to improve ranking performance on repetitive variable specification definitions by applying a machine learning approach. While the model struggled on definitions and datasets that were more study-specific, those are exactly the kind of problems that deserve statistical programmer time; there's no need for them to be writing definitions that this model can handle with relative ease. While there is immense potential for improvement, the current workflow is already providing the majority of correct variable specifications for many studies. That's an efficiency opportunity worth further pursuit.

REFERENCES

[1] Stephen M. Noga, Brandon Welch, and Jeff Abolafia (2018). A LearnedApproach to CDISC Specifications. PHUSE EU Connect 2018. Retrieved from https://www.lexjansen.com/phuse/2018/ml/ML09.pdf

[2] Owen Stephens (2018). OpenRefine: Clustering In Depth. Retrieved from https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth

[3] Ioannis Tsamardinos, Amin Rakhshani, and Vincenzo Lagani. (2014). Performance-Estimation Properties of Cross-Validation- Based Protocols with Simultaneous Hyper-Parameter Optimization. International Journal on Artificial Intelligence Tools. 24. 10.1007/978-3-319-07064-3_1.

[4] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore (2016). Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. *Proceedings of GECCO 2016*, pages 485-492.

[5] L. Breiman (2001). Random Forests. Machine Learning, 45(1), 5-32.

[6] Hamed Valizadegan et al. (2009). Learning to Rank by optimizing NDCG Measure. Advances in Neural Information Processing Systems 22, pages 1883-1891. Retrieved from https://papers.nips.cc/paper/3758-learning-to-rank-by-optimizing-ndcg-measure

RECOMMENDED READING

Chapter 5. (Machine Learning Basics) from Deep Learning by Goodfellow et al. This is a fantastic and concise introduction to machine learning.

APPENDIX A
The following 36 ADaM datasets were include in this analysis:

- ADAE
- ADSL
- ADEFF
- ADLB
- ADCM
- ADEG
- ADMH
- ADVS
- ADQS
- ADVE
- ADPE
- ADEX
- ADMC
- ADDV
- ADPD
- ADIE
- ADDA
- ADPR
- ADEF
- ADDS
- ADEC
- ADCI
- ADPK
- ADPC
- ADPM
- ADZL
- ADXP
- ADHS
- ADPP
- ADMB
- ADMS
- ADSV
- ADCO
- ADYA
- ADYO
- ADYT

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Preston Burns
Rho
2635 E NC Hwy 54
Durham, NC 27713
Work Phone: (919) 595-1547
Email: Preston_Burns@rhoworld.com
GitHub: pburnsdata