MFM Datasets Lessons Learned

This is an informal collection of some of the lessons I learned while working on the MFM Datasets project.

## Data is King

Machine and Deep Learning algorithms are only as good as the data they are provided, and poor data will invariably cause improper conclusions. In our case, I feel we did a pretty good job of cleaning our variables and performing Extended Data Analysis on most of the fields. However, when the algorithms predicted that Inoxy\_incrdose and Intratocolytix were top predictors, it took a long time for us to realize that these were very Site-dependent. I personally could have spent more time understanding the data fields and how their inter-relationships. Crosstab queries can be very helpful for this.

Similarly, cleaning / organizing the data is very important, but must be done somewhat carefully. Some of the ML algorithms (such as SVM, citation) are sensitive to large differences in magnitude. So scaling of the data makes sense in this regard. We may have gone too far, however, in that we removed most of the continuous variables and recast them as Categorical / Ordinal. There were two examples where in the case of the Gradient Boosting algorithm that Continuous variables appeared to outperform similar Ordinal versions: Age & BMI. In later versions of our data, we ran GB with both a Continuous and Ordinal versions of these variables, and both of the Continuous ones were more likely to be represented in the “top predictor” lists. More controlled experiments would need to be run to determine if there is any validity to this trend. One possible confounding factor could be the way these Ordinal variables were defined – the “high\_Age” variable, for example was segregated based on common obstetrical age ranges rather than fixed bins. I’m not sure if this could have an affect on the algorithm performance or not.

## Missing/Unknown data is difficult

There was a lot of Missing or Unknown fields in our dataset which we handled by imputing to the Median, which is considered one of the best available methods. However, imputing Estimated Bloodloss (EBL) to the median, produced a value of 350 ml, which is “Normal”. So while this did not affect us directly since we used it as a target, some of these women likely had high Bloodloss that wasn’t recorded as such. This may have confused the algorithms since this subset were labeled as Negative when they should have been Positive.

By contrast, the Venkatesh paper simply threw out a the data where their sole target was missing, high Estimated Bloodloss. This method also poses it’s own set of issues, but doesn’t suffer from mislabeling.

Alternatively, we could have tried using ML & DL methods to impute Missing data. There are examples of using both Autoencoders and simple Multi-layer Perceptrons to impute missing data. (Citations needed).

## More data isn’t always helpful

In general, ML and DL algorithms produce better predictions when they have more data to work with. We found a couple exceptions in our case, however.

* Surprisingly, the Gradient Boosting algorithm usually performed somewhat better when using undersampling (1:1) as compared to class-based sample-weighting, which used all of the data but with the minority class weighted higher than the majority class.
* Oversampling, specifically using the SMOTE algorithm, did not produce better predictions than class weighting. We did not try a combination of random under and oversampling, however, so that may produce better results.
* When creating the trans\_loss target (see below) we combined Transfusion with Estimated Bloodloss (EBL) to get more positive cases. However, our prediction results were much worse that EBL alone.

## Choose target carefully

The choice of the prediction target is as important as data preparation. And clinical significance of a target variable, while a good starting point, should be revisited often. In our case, the transfus\_yes variable, while very important clinically, doesn’t appear to work as a target variable in this dataset because of the institutional differences in when transfusion is chosen and possibly how the data is recorded.

### Multipart targets are especially tricky

Also, care must be taken when using multipart, or combined variables as the target. In our case, the “trans\_loss” variable was defined as “transfus\_yes” OR “Estimated Bloodloss >= 1000ml”. In the same manner, “transfus\_yes” itself was defined as “Postransfus” OR “Bloodproduct”. Combining these three variables in this way, while reasonable, does require a strong understanding of what the actual clinical practice is in defining them and how well they are coded by the institutions. I’m not sure we had enough data to make an informed decision in this case. The intra-site experiments seem to indicate a wide variation in how and when transfusion is performed.

Complicating this target selection was the fact that all three of these variables had large numbers of Missing/Unknown values (see above discussion). While this doesn’t eliminate these fields from consideration as a targets, it probably requires more statistical analysis in order to make valid conclusions. [re-word?]

Finally, we didn’t run any of our models with just the “Estimated Bloodloss” target until very late in the process. With a multipart target like this, it probably makes sense to run the algorithms against the individual parts to see how they perform. In our case, we might have seen that EBL was performing very well and that the transfus\_yes portion of trans\_loss was dragging it down.

## Combining sites can produce sub-group affects

It appears that sub-group affects, namely the Sitenum in our case, caused major problems with our dataset. This seems like a difficult problem to solve in general since we need as much data as possible, but combining multiple sites introduces additional issues. This possibly could have been handled earlier in the process with some cross-site extended data analysis. We may have seen the large disparity in the application of transfusion between the different sites and therefore avoided that target, or controlled for site in some manner.

It seems like this is could be an issue for any analysis using the CSL dataset. And it could extend to many other clinical studies that make use of data from multiple sites without having consistent standards for when certain procedures are performed. [examples?]

[See if Rich wants to add anything]

## Use multiple statistics for imbalanced data

Our chosen targets were all highly imbalanced, with a positive to negative ratio of at least 14:1. The traditional use of Accuracy and ROC\_AUC(Receiver Operating Characteristic Area Under the Curve) can be very misleading in the case of imbalanced data. While these statistics can still be useful for comparison purposes and as a cross-check, the following statistics that emphasis the predictive performance of the Positive class should be favored.

* Precision Recall Area Under the Curve (PR\_AUC ) : Similar to ROC\_AUC, this statistic is more consistent than ROC\_AUC for imbalanced data [citation needed]
* Matthews Correlation Coefficient (MCC): A calculation that used all quadrants of the Confusion Matrix, this statistic is often cited as the most robust for handling imbalanced data.
* F1, F2, F-beta:
* Precision/Recall/Specificity: These statistics calculate different portions of the Confusion Matrix and provide insight into the quality of the prediction. Recall/Sensitivity is especially useful for imbalanced dataset since it is sensitive to False-Negatives which are can be critical in a clinical setting.

Also, the clinical “cost” of miss-prediction should also be considered when choosing a statistic. In our case, the “cost” of a False Negative where a woman is at high risk for an adverse outcome isn’t flagged could be much more detrimental than one who flagged as a False Positive and given special care that isn’t necessary. Obviously, this depends on the specific intervention used. [example?]

## Hyperparameter / algorithm tuning

### When tuning ML hyperparameters, start with defaults

Many of the machine learning algorithms, especially the best-performing ensemble, tree-based algorithms such as Gradient Boosting and Random Forest, produce reasonable results with the default hyperparameters. This is dependent on the quality of the input data, however. Also, the defaults can run much slower due to the algorithm needing to work harder to find a minimum. However, the default are often a better starting point than trying to guess which hyperparameters to start tuning.

### Be careful of stochastic variability (check term)

Machine and Deep Learning algorithms can be sensitive to random/stochastic variation. [citation] A couple techniques to ameliorate this include cross-validation and averaging over many initial random seeds. This is especially useful in identifying top predictor variables given their sensitivity.

### Using Optuna (or other) can save effort if done correctly

Using a third-party algorithm tuning application (e.g. Optuna)[reference] can both save time and produce a better result. It can be coded to try multiple algorithms (both machine and deep learning), perform hyperparameter tuning and reduce stochastic variability by running cross-validation or multiple runs with different random seeds. While the initial learning curve can be a bit steep, use of this type of tool can both help optimize the algorithm and increase confidence in the final results.

### “Top Predictor” variables are difficult to assess

* + Predictions are based on specific model and vary per run
  + Need to run with multiple random seeds similar to determining Confidence Interval
  + Use of Shapely values (via SHAP software) can provide insight

## Specific MFM comments

### Not clear that feature selection helps in our case

We initially spent a lot of time performing feature selection using Cramer and Theil, and also trying to use Cohorts. While this was a useful in that it provided insight into the associations among our variables, and was a good cross-check against our results, our top algorithms performed better against all the data fields. In our case, this was not an especially high number (191 for the Pre-Intra set), which did not require high compute resources. A different dataset may very well need this type of preprocessing.

### Deep Learning needs additional research

The two Deep Learning networks we tried produced middling results: better than Logistic Regression, but worse than Random Forest and Gradient Boosting. However, due to the higher coding requirements and lack of experience at the time, we only scratched the surface of what is possible. The first thing to do would be better tuning of the neural network hyperparameters, such as number of layers, neurons per layer, activation function, loss function, number of epochs and learning rate. Very little was done in this regard and could greatly improve the prediction performance. At the time I did not know that a tool like Optuna can be used to automate the tuning of Deep Learning algorithms and could save quite a bit of time.

### Consider reproducing other results as a starting point

In hindsight, I believe it may have been worthwhile to attempt to reproduce the Venkatesh results on our own given the similarity of our goals. The advantages of starting with this approach, if successful, would have provided us with an initially, reasonable first-cut from which to build. It may also have reduced and shortened our learning curve with this data. From there we could have branched off and experimented with everything we did, but it would have provided a point of comparison.

The negatives of this starting approach are that it would have been difficult completely reproduce the results given the data we had, and it may have caused us to “lock in” on their method of analysis. It also may have been more difficult to publish since, to me, there seems to be an over-emphasis to produce “novel” results.

## Conclusion

Overall, this was a fascinating and useful project to work on … [more]