Paper SAS4463-2020

Explanatory Machine Learning Model Plots for Epidemiological and Real-World Evidence Studies

David Olaleye, SAS Institute

ABSTRACT

For real-world evidence (RWE) and epidemiological studies, comparative effectiveness research (CER) provides actionable information for drug regulatory and health care decision-making. CER relies on white box statistical and machine learning (ML) models for estimating treatment effect and drug safety evaluation. Black box ML models are also powerful for generating better predictions, but how to interpret the model results is not straightforward. How should ML model results be presented to regulators to assure them that the results are accurate, fair, and unbiased? How should ML model findings be presented to end users to overcome the stigma of black box model bias? We provide a standardized interpretability plots framework for evaluating and explaining patient-level ML models using observational health-care data. The paper shows how to use SAS® Cloud Analytic Services action sets and model-agnostic interpretability tools available in SAS® Visual Data Mining and Machine Learning to explain the functional relationship between model features and target outcome variable. In addition to using the partial dependence and individual conditional expectation plots, we present some use cases and example code to demonstrate how plots that represent time-varying and non-time dependent variables interaction, cohort-period-feature effect, gender-age group stratification, beneficial and untoward effects of drug exposure, drug-disease interactions, and confounding-by-indication effect can be used to explain ML model results.

INTRODUCTION

Machine learning (ML) algorithms are increasingly being used in real-world clinical settings to study a variety of clinical outcomes such as readmission, healthcare resource utilization, adverse drug reactions, treatment pathways, polypharmacy, and medication adherence. For real-world evidence (RWE) and comparative effectiveness research (CER), study questions and hypotheses to be tested often require application of causal inference analysis that goes beyond mere prediction. For purposes of estimation and interpretation of feature effect estimates, CER often relies on white box ML models such as logistic and linear regression, or Cox proportional hazards model for time-to-event modeling. Black box ML models such as gradient boosting, random forests, naïve Bayes, and neural networks are underutilized in CER, even though they generate predictions with much better accuracy than the classic regression models. This is partly due to lack of straightforward interpretation of the model features, unlike the estimates obtained from decision tree or logistic regression models that are interpretable.

Recent enhancements to black box ML algorithms in the area of model interpretability now make it possible to build predictive and causal inference models using real-world data from insurance claim databases and electronic health records. Model interpretability is important in pharmaceutical and healthcare settings where regulators, drug safety reviewers, and decision makers need to ascertain that the model results are accurate, reliable, and free of bias. More importantly, these models should reflect transparency and minimize the possibility of producing erroneous or unintended results that might result in potential bias against a subgroup of the population. In the book

"Interpretable Machine Learning" by Christopher Molnar (2019), he defined explainable ML models as 'methods and models that make the behavior and predictions of ML models to be understandable to humans." Therefore, model interpretability dictates that ML models provide sufficient information that will help end-users and the target audience understand (1) the rules resulting in the decision to select one model as the champion model over other competing models, (2) the dynamics between input features and output predictions, (3) how best to interpret important features selected by the model, and (4) how each feature contributes to or partially influences model predictions. Partial dependence (PD) plots, individual conditional expectation (ICE) plots, local interpretable model-agnostic explanations (LIME), and temporal pattern discovery plots (Norten et. al. 2010) are examples of visual, model-agnostic tools that can be used to explain black box ML models.

Another important consideration with ML model interpretability is 'evidence generation'. This term is synonymous with the concept used in quality control process settings. ML models are not immune to data quality issues. Data incompleteness is one major source of bias that can compromise model interpretation, especially with real-world data. For any ML modeling task, the following two interpretation issues are noteworthy:

- holding data source and question constant, different ML models can produce different features' importance results and different model parameter estimates; and

- holding ML algorithm constant, different runs of the same black box model but with different autotuning of model hyper-parameters can generate models with better accuracy but at the expense of model interpretability.

Therefore, the following model attribute checks are considered important to make ML models digestible for consumers:

- model reliability - for evidence produced by an ML model to be deemed reliable, the evidence should be independent of the modeling method or apparatus used. In other words, modelers should expect to see identical results from multiple ML algorithms that perform similar analyses on the same data when applied to the same question.

- model replicability and reproducibility – results generated from a black box model can be deemed reliable if the results are replicable and reproducible. For example, for real-world evidence studies, evidence generated from analyses performed against administrative insurance claims from one large insurance company database might be strengthened if it can be replicated on claims data from another large insurance commercial database, particularly for a well-studied problem such as hospital readmissions.

- model generalizability - points to the value of external databases for generalizing results beyond the model training database that produced the results. Being able to evaluate the performance of a model that was trained on one database by observing its discriminative accuracy when applied to a different database provides some level of confidence. One thing that makes a data scientist happy is when identical analyses are performed against different databases and the models show consistent similar results (Schuemie et al. 2018; Schuemie, Ryan, et al. 2018).

In this paper, we show how to use SAS Cloud Analytic Services ML procedures and action sets and model-agnostic interpretability tools available in SAS Visual Data Mining and Machine Learning to explain the functional relationship between model input features and output predictions. We also present a model-agnostic visual tool called post-prediction

model features association check (PMFAC) plot for post hoc ML model fit assessment and explanation of features' association. PMFAC is a model prediction checking tool that can be used to check whether the fitted model is consistent with observed data and to check for reproducibility of model predictions under different sampling and modeling scenarios. The assumption is that the sample distribution of model predictions using the observed data should be similar to that observed in the replicate samples drawn from the same data. The generated plots from the tool often present useful information that can be used to inspect the association dynamics between model input features and output predictions such as detecting variable interactions or association patterns that might suggest gender-age group effect, beneficial effects of treatment, or drug-disease interactions.

The paper is organized as follows: a brief description of the data used in the paper, presentation of SAS Visual Data Mining and Machine Learning model results and interpretability plots, description of the PMFAC tool, and illustrative examples of using the PMFAC tool to investigate the association between model input features, including assessing the impact of the HbA1c test, and 30-day hospital readmission likelihood. HbA1c is used as a marker for diabetes care for individuals identified as having a diagnosis of diabetes mellitus. As many research studies have shown, measurement and monitoring of HbA1c have been found to be associated with a reduction in readmission rates amongst hospitalized diabetic patients (Strack et. al. 2014). The version of SAS® code used to create the PMFAC tool is presented in the Appendix of the paper. The final version of the code will be made available on SAS GitHub.

METHODS

The data used for this exercise was downloaded from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008). Additional information about the data set and a full description of its features are available on the web site (see the provided link), and from the published research article (Strack et al. 2014) that used the data to study the impact of HbA1c measurement on hospital readmission rates. Table 1 provides brief information about the data and the model input variables.

Table 1: List of features and their descriptions in the initial data set (the data set is also available at the web site of Datamining and Biomedical Informatics Lab at VCU).

Study parameters	Description
Population data	The unit of analysis is an encounter per patient with the first encounter for the patient as the primary admission and used to determine whether the patient was readmitted within 30 days.
Outcome event	30-day readmission. Coded as "readmitted," if the patient was readmitted within 30 days of discharge or "otherwise" (covers both readmission after 30 days and no readmission).
Primary model input variable	HbA1c status – consisting of eight groups of encounters: (1) normal HbA1c with diabetic medications change, (2) normal HbA1c and no diabetic medications change, (3a) HbA1c performed, result is greater than 8%, and change in diabetic medications, (4) HbA1c performed, result is greater than 8%, and no change in medications, (5) borderline HbA1c and change in diabetic medications, (6) borderline HbA1c with no change in diabetic medications, (7) no HbA1c test performed but change in diabetic medications , and (8) no HbA1c performed with no change in diabetic medications.

Covariates	55 features describing the diabetic encounters, demographics, diagnoses, diabetic medications, number of visits in the year preceding the encounter, and payer information.
Total observations (train, validate, test)	N=88291 observations divided into three groups:- training (42912), validation (28606), and test (16773).

Source: http://www.cioslab.vcu.edu/

MACHINE LEARNING MODEL INTERPRETABILITY TOOLS

ML model interpretability is important for trust, transparency, and informing human decision-making. Because ML models often act on imperfect real-world data and opaque learning algorithms that might produce unintended results, the ability for end-users and the target audience to adopt them for decision-making largely depends on their reliability, generalizability, and reproducibility. ML model interpretability methods can be categorized into the following two categories:

- global explanations provide an understanding of the model's behavior across the entire data set
- local explanations provide an understanding of the model's behavior in a small region.

Across these two categories, the explanations can be further classified into the following three areas, depending on where interpretation is needed in the model development workflow:

- model features' discovery plots (hindsight) – this help determine and select which features or variables are the best candidates to include in the main predictive model. The challenge here is the likelihood of 'hindsight bias' – the tendency for model features to exaggerate the extent to which a past event could have been predicted beforehand.
- model features' explanatory plots (insight) – these are posterior or post-prediction model plots that can be used to gain accurate and deep insight and understanding of the inner workings of the model, as well as highlight the different impacts that different model features exert on model predictions. They can be used to perform post-prediction what-if-scenario analysis impact of one variable on the target variable holding other model features' values constant. Example plots include PD, ICE, Shapley's, PMFAC tool, etc.
- mode features' revelatory plots (foresight) – these are also post-prediction model plots that tend to forecast or reveal how certain events predicted by the model might unfold over a period of time, and what actions (impacting features) can be undertaken to influence such outcomes. Example plots include age-period-cohort effect, risk scorecards, etc.

MODEL STUDIO AND SAS VISUAL DATA MINING AND MACHINE LEARNING

Model Studio is an integrated visual environment that provides a suite of analytic data mining tools to facilitate end-to-end data mining analysis. The data mining tools supported in Model Studio are designed to take advantage of the SAS® Viya® programming and cloud processing environments to deliver and distribute analytic model data mining champion models, score code, and results.

SAS Visual Data Mining and Machine Learning is a web-based collection of analytic tools and data mining best-practice templates that provides an integrated comprehensive

visual data mining and machine learning approach to a wide variety of analytic data mining problems. It enables the user to visually assemble, configure, build, and compare data mining models and pipelines for a wide range of analytic data mining tasks. SAS Visual Data Mining and Machine Learning also provides different types of model interpretability tools that are very useful for explaining and interpreting ML models. These techniques are model-agnostic and can be applied to any model that is generated by a supervised learning node. Available plots include Partial Dependence (PD), Local Interpretable Model-Agnostic Explanations (LIME), Individual Conditional Expectation (ICE), and Kernel SHAP. Shapley values are used to determine the relative importance of each variable to a given observation's prediction. For additional information, visit the SAS Help Center (https://support.sas.com/en/documentation.html).

POST-PREDICTION MODEL FEATURES ASSOCIATION CHECKPLOT (PMFAC) TOOL

The ability to reproduce and replicate the results generated from a black box model performed on one data source in another similar but independent data source might help to increase the confidence level of that black box model. Validation using multiple data sources helps to identify potential or unanticipated issues with black box models particularly if they are data-related or are model fitting issues. When another independent data source is not available for use for validation, a pseudo-external validation method is the post-prediction model check of results based on the raw data and employing the same functional model to score the replicate samples generated from the raw data. Why is this method appealing? White box models such as regression models employ known or estimated distribution to fit the model, the same cannot be assumed for black box models that use complex algorithms to generate predictions.

The intuitive appeal of this method is that it provides a test platform and standard test statistics to assess the fit of the predictive model (regardless of whether it is a white box or black box model), as well as explore ways in which the model might be lacking. Test statistics such as the mean, median, standard error, min, max, etc. summarize the parametric or functional distribution profile of the test variable of interest. Therefore, a check of the fitted model on the test variable might help to reveal aspects of the raw data that were not well-predicted by the model, since one would expect to see similarity in the distribution profile of the test statistics for the test variable in the raw data as well as in the replicate samples generated from the data. The approach described here is not used to perform hypothesis tests on the test variable, nor is it being used to reject or accept the validity of the model fit, rather it is an intuitive approach to better understand the limits of the black box model with regard to its reliability and applicability in real-world case scenarios.

PMFAC tool is a post-prediction model-agnostic technique tool for checking the sensitivity of the model fit in relation to the observed data, particularly the sensitivity of the model predictions to the influence of model input features. The tool first checks whether the fitted ML model is consistent with the observed data and generates diagnostic plots for visualizing the results. The tool then compares the observed data with each replicate sample drawn from the observed data to see if there are any large and systematic differences. Large discrepancies between the test statistics for the observed data and those of the replicate samples might suggest a possible model misfit. The tool also provides plots to investigate the behavior of the ML model predictions based on (1) the observed data and (2) the replicates generated from the whole sample. Therefore, the plots can reveal anomalous features' interactions within the replicate samples that might not have been detected given the real data. Further, it provides post hoc model fitting checking plots that

can be used to examine the behavior of the model input variables and the target outcome variable, especially if different ML algorithms present different representations of the relationship.

RESULTS

MODEL STUDIO: MODEL INTERPRETABILITY PLOTS

Figure 1 presents the Model Studio pipeline for the readmission prediction task. Model Studio enables the user to build automated or user-customized model pipelines. The figure shows a pipeline with three autotuned black box ML models (neural network, gradient boosting, and random forest models) and two white box ML models (decision tree and forward logistic models). The objective here is to compare the model results and model interpretability plots across the different ML algorithms.

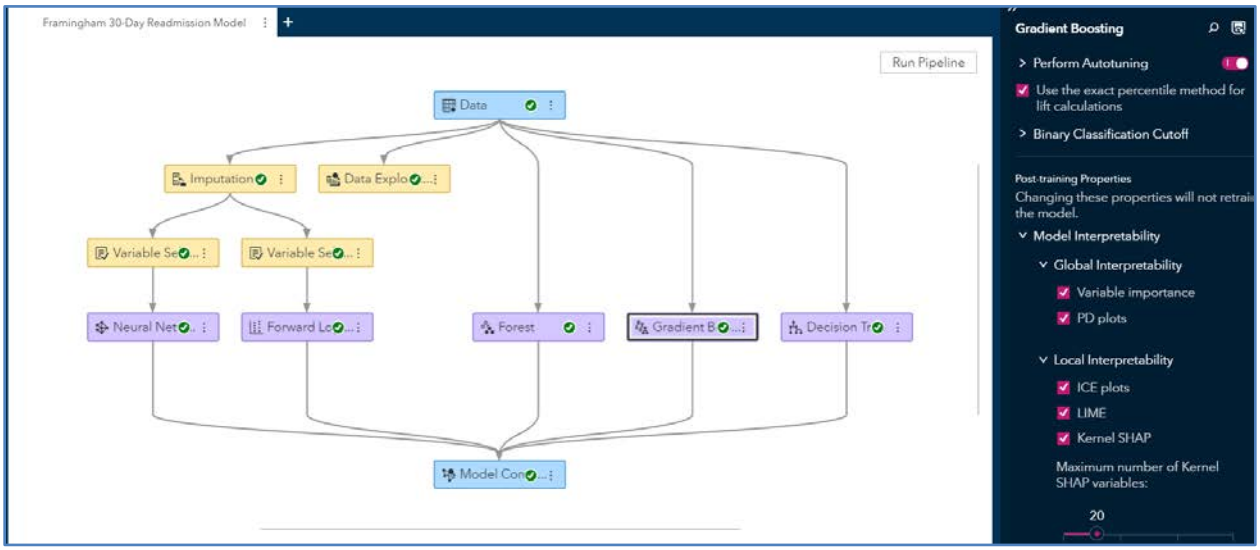


Figure 1: Model Studio Pipeline for 30-Day Readmission Prediction Model

The champion model for this project is Forest. (See Table 2.) The model was chosen based on the KS (Youden) for the Test partition (0.13). 86.13% of the Test partition was correctly classified using the Forest model.

Table 2: Model Comparison Results

Champion	Name	KS (Youden)	Misclassification Rate	Gain	Average Squared E...
★	Forest	0.1254	0.1387	0.7498	0.1189
	Neural Network	0.1174	0.1394	0.6604	0.1191
	Forward Logistic Regression	0.1170	0.1389	0.7060	0.1185
	Gradient Boosting	0.1071	0.1387	0.5074	0.1202
	Decision Tree	0.0741	0.1379	0.3659	0.1203

The five most important factors are age, number of medications, discharge disposition-id, number of inpatient admissions, and number of diagnoses. Misclassification rates were similar for all the models (revolve around 0.139).

Figure 2 presents the partial dependence model interpretability plots for the number-of-inpatient model feature and 30-day readmission probability for forest, gradient boost, and logistic models. The plot shows the relationship between number of inpatient admissions and the predicted target variable, averaging out the effects of the other inputs. It displays values for number of inpatient admissions, as well as a heat map that shows the distribution of observations with the attribute on the X axis and the corresponding average prediction for the target variable on the Y axis. For the random forest model, the highest average target prediction is 0.39 and occurs when number of inpatient admissions = 6; the lowest average target prediction is 0.08 and occurs when number of inpatient admissions = 0. For the gradient boost and logistic models, the observed relationship shows the highest average target predictions to be 0.13 and 0.54, and they occur when number of inpatient admissions = 2 and 11, respectively.

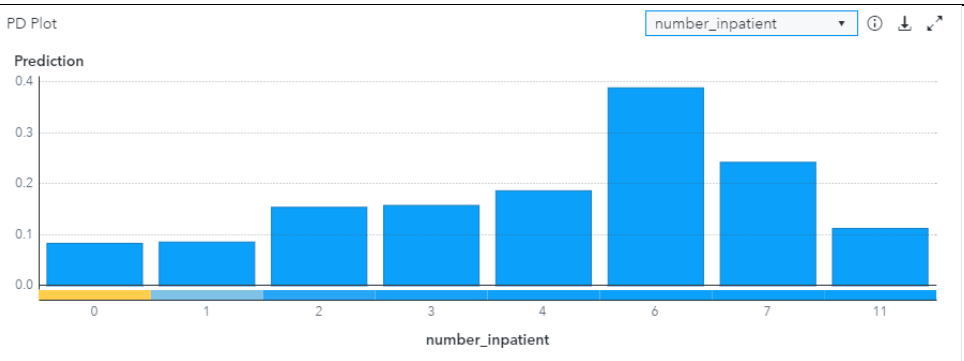
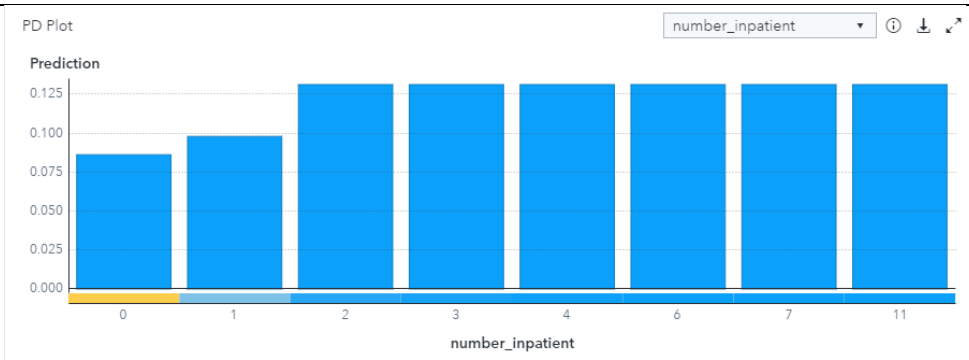
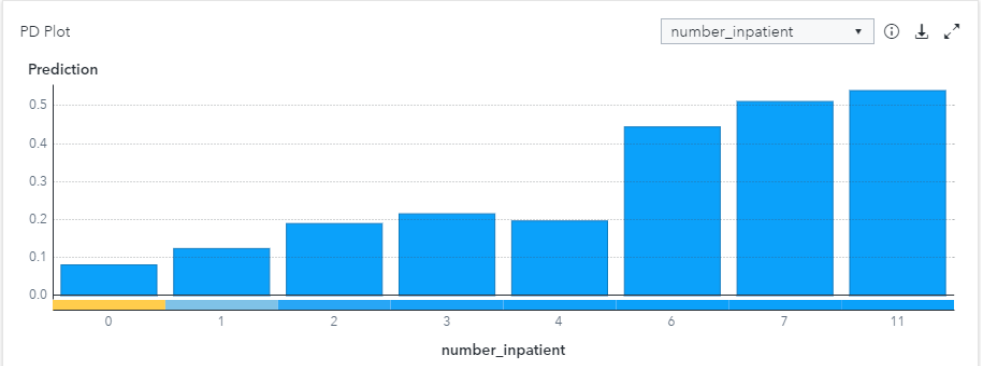
Forest	
Gradient Boost	
Logistic	

Figure 2: PD plots for number of inpatient model feature and predicted 30-day readmission probability for random forest, gradient boost, and logistic regression models.

Figure 3 shows the overlay plot of PD and ICE, showing the relationship between number_inpatient and the predicted target for each individual observation. For each individual observation, the corresponding ICE plot displays values of number of inpatient admissions on the X axis and the corresponding prediction for the target variable on the Y axis, holding the other inputs constant at their values for each observation. In contrast to the gradient boost model, both the forest and logistic models tend to show similar association patterns for the selected patients given their 30-day risk of readmission.

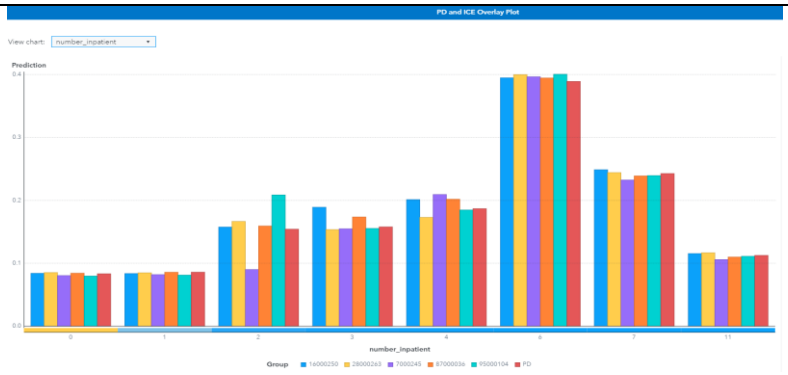
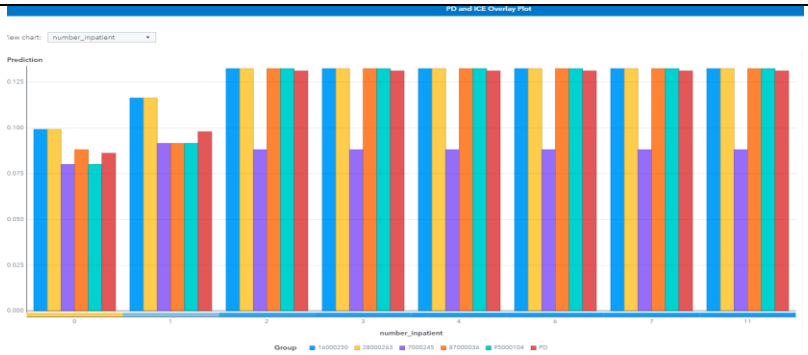
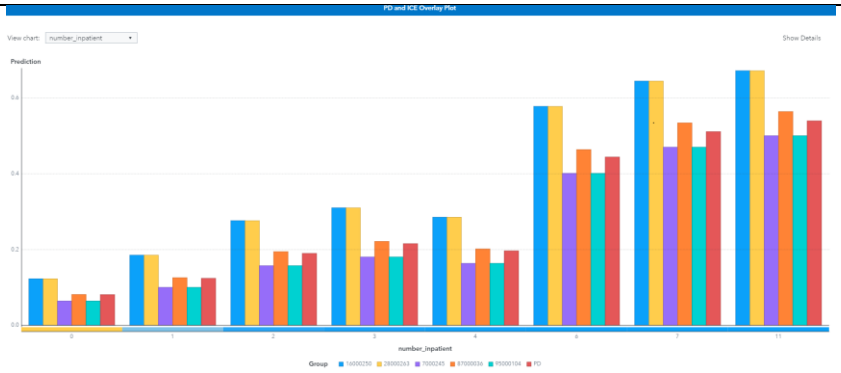
Forest	
Gradient Boost	
Logistic	

Figure 3: PD and ICE Overlay plots for the number of inpatient model feature and predicted 30-day readmission probability for forest, gradient boost, and logistic regression models.

PMFAC TOOL SAS® MACRO PROGRAM

As a post hoc model fit checking tool, the PMFAC macro program can be used to examine the predictions of a machine learning model based on the observed data and the replicates generated from the data. The associated plots can be used to examine the relationship between one or more model inputs and the model predictions. Unlike the PD plot, the PMFAC plot is not based on the unique values of the plot variable fitted against

values of the complementary/explanatory variables. It is also worth mentioning that the PMFAC tool is not intended to be used as a statistical hypothesis testing tool. Rather, we are using the tool to evaluate post hoc ML model predictions and to examine the behavior of the ML model in two testing scenarios. Specifically, to check whether the association observed between the model outcome variable and input variables based on observed data is also found (or to what extent they differ) in the replicate samples generated from the observed data. The PMFAC macro program thus provides one plausible approach to check for data and algorithm bias in ML models. The idea is to generate multiple replicate samples from the training data (or from the partitioned data sets) and then compare each fitted replicate sample with the fitted training data and check whether there are any large and systematic differences.

One of the model fit plots generated using the PMFAC tool is the plot showing the post-predictive distribution check of the fitted model outcome or dependent variable. This is shown in Figure 4 for a gradient boost ML model. No extreme p-values are observed, which might suggest the notion that the predicted results from the replicate samples are similar to the actual observations and that the model fits the data.

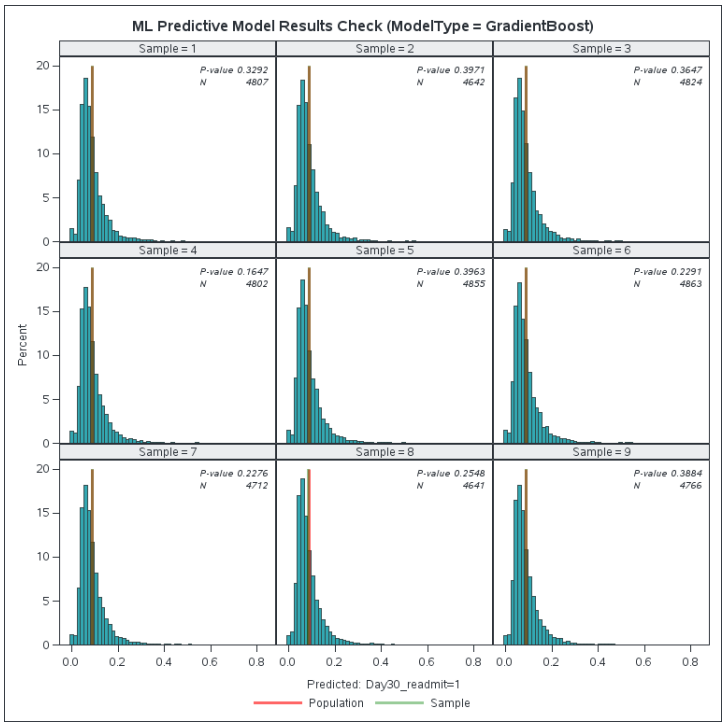


Figure 4: PMFAC post-predictive distribution check plot for predicted 30-day readmission probability (gradient boost model).

PMFAC TOOL: ONE-VARIABLE PLOT

The PMFAC tool provides post hoc model fitting and plot checking that can be used to examine the behavior of model input variables and model predictions, especially if different ML algorithms present different representations of the relationship. Figure 5 shows a one-variable PMFAC plot for the marginal effect of the binned number of inpatient admissions on average readmission risk. While the readmission risk profile shows a somewhat similar pattern for both gradient boosting and random forest models, the latter model predicts a slightly greater mean readmission probability than the former. Interestingly, this association plot looks closely similar to the PD plot for forest and gradient

boost models (see Figure 2) for the same variable generated by the Model Studio model interpretability tool.

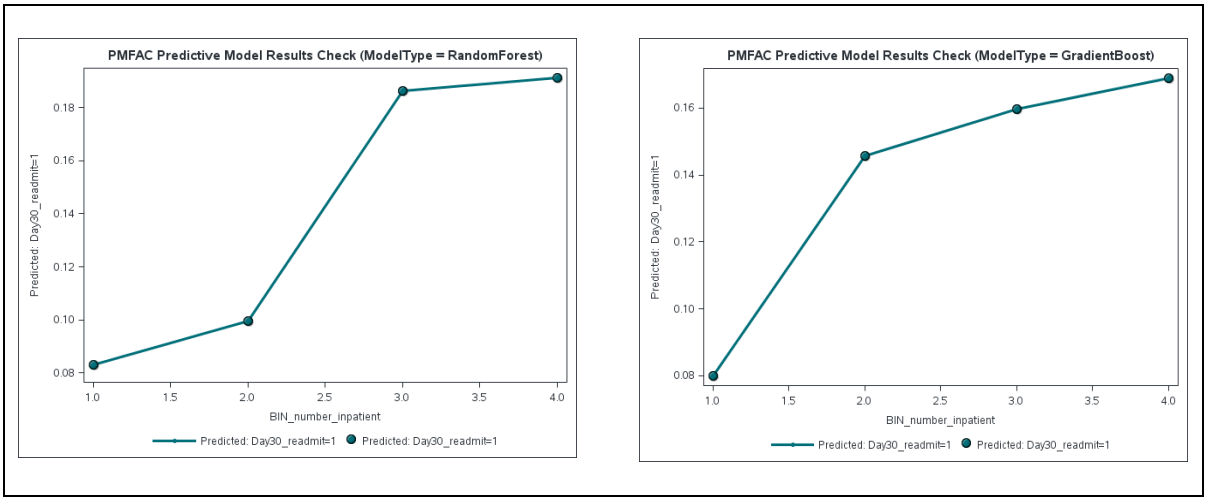


Figure 5: Predicted 30-day readmission probability PMFAC plot by number of inpatient admissions (binned) and machine learning model type.

ML models use different algorithms to generate predictions. The question is: which one should the data modeler select to interpret the association between readmission probability and a model input feature, especially if they present contrasting views of the association? As shown in Figure 6, the PMFAC tool provides visual plots of the model predictions based on observed data in contrast with the replicate samples generated from the data for both random forest and gradient boost models. What is striking is the consistency of the association pattern observed across the replicate samples generated from the observed data.

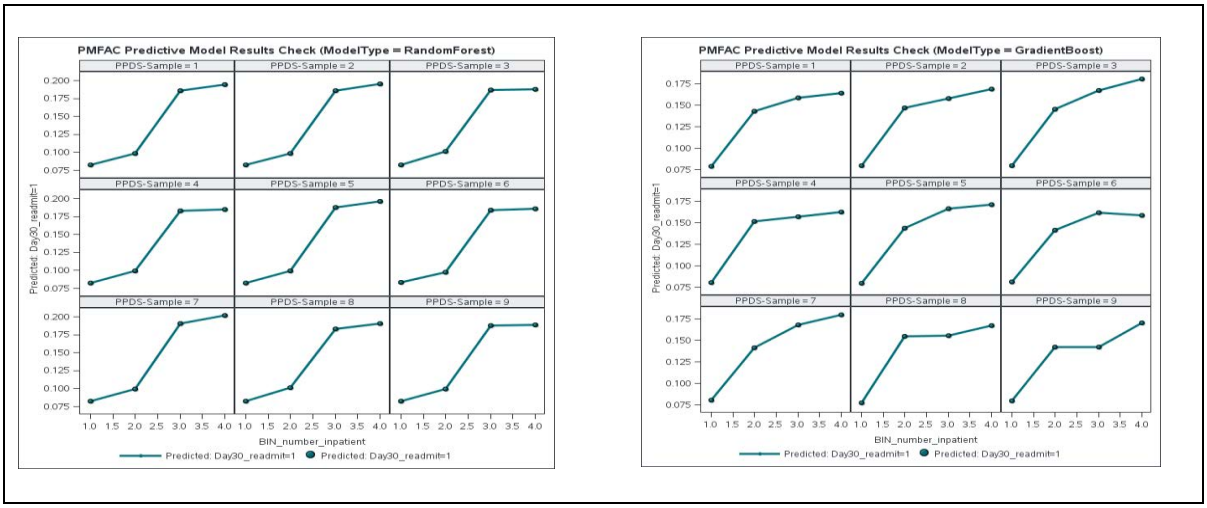


Figure 6: PMFAC model prediction check for predicted 30-day readmission probability by inpatient admissions (binned) based on the observed data and replicate samples of the observed data for random forest and gradient boost models.

PMFAC TOOL: TWO-VARIABLE PMFAC PLOTS

The two-variable PMFAC plot for the association between the number of inpatient admissions and controlling for a primary diagnosis of diabetes encounter on readmission

probability is presented in Figure 7. Based on the observed data, the two-variable PMFAC association shows some type of interaction between inpatient admission and primary diabetes diagnosis. Both black box models suggest the propensity for increased inpatient hospitalization for patients with a primary diabetes diagnosis relative to those without.

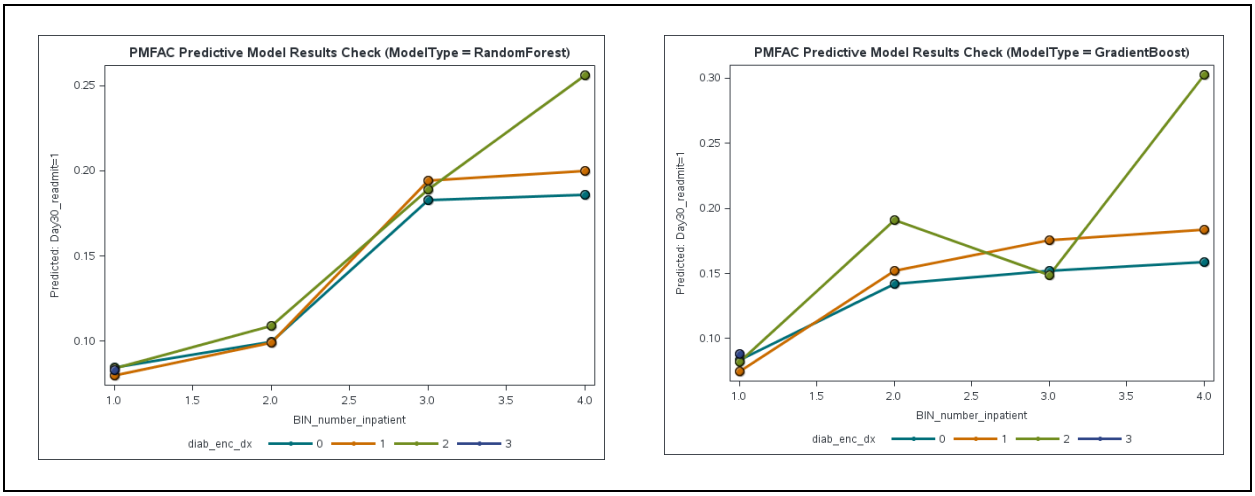


Figure 7: Two-variable PMFAC plot of predicted 30-day readmission probability for inpatient admissions (binned) controlling for a primary diabetes diagnosis encounter.

When the two-variable plot is evaluated at the replicate sample level (see Figure 8), the interaction appears to be either quantitative or qualitative in nature. Thus, by evaluating model predictions with the PMFAC tool across the replicate samples, we gain additional information about readmission risk that would have been missed if we had only looked at the one-way variable PMFAC plot.

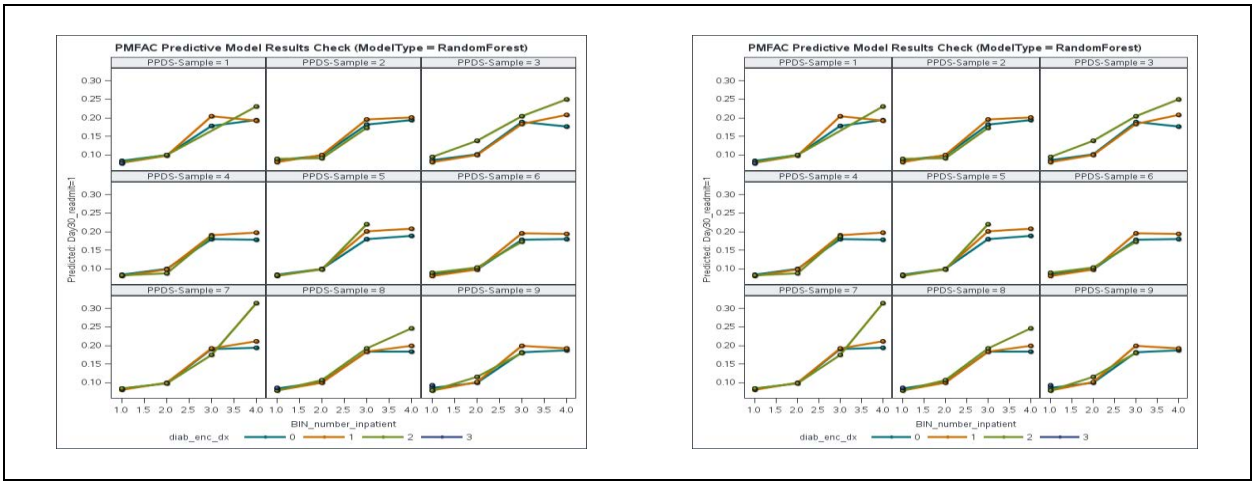


Figure 8: PMFAC model prediction check for predicted 30-day readmission probability by inpatient admissions (binned) and primary diabetes diagnosis based on the observed data and replicate samples of the observed data.

PMFAC TOOL: ASSOCIATION PATTERNS DISCOVERY PLOTS

The PMFAC tool can also be used for open-ended pattern discovery in a large patient records repository such as the administrative claims database. The graphical plot summarizes and contrasts model predictions across multiple ML models based on the observed data and replicate samples of the observed data. With real-world data, possible

features' association patterns might include age-gender interaction, disease severity, beneficial and untoward effects of drug exposure, or detection of the presence of variable confounders in a propensity score matching task. Figures 9 displays gradient boost model predictions for (a) the association involving age and gender and (b) the association involving the HbA1C test with and without prescription change and insulin use.

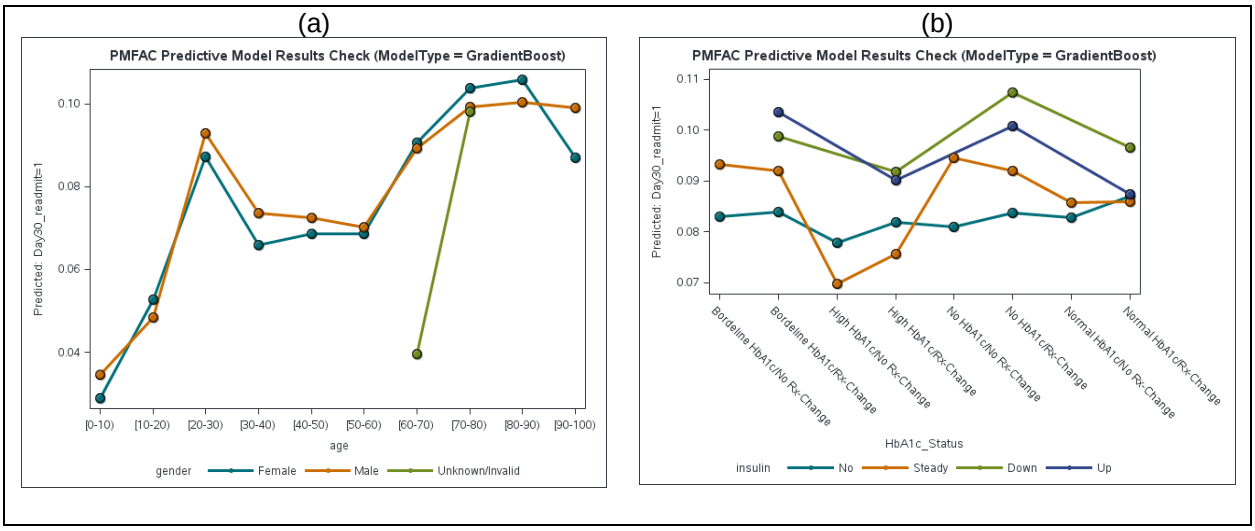


Figure 9: PMFAC tool for open-ended association patterns discovery- (a) age-gender features' association and (b) HbA1c test status with or without prescription change and insulin use.

CONCLUSION

Interpretable and explainable machine learning modules are now standard features in many ML applications. The ability to explain and interpret ML model outputs in ways that human beings can comprehend promotes transparency and trust in these models. Standard interpretability tools such as PD, ICE, and LIME available in SAS Visual Data Mining and Machine Learning and Model Studio enable data scientists to understand the inner workings of the ML models they build, reveal input variable dependency and interactions, and show how changes in model input variables can influence model predictions. We presented another post hoc model prediction checking tool, PMFAC, that can be used to check for consistency and reproducibility of ML results when replicate samples generated from the data are compared with the results generated using the real data. Generated PMFAC plots can help to reveal or dig deeper to find interactions among model variables and unusual combinations of model input features that might yield important findings.

REFERENCES

Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. https://christophm.github.io/interpretable-ml-book/.

Schuemie, M. 2018. "Empirical confidence interval calibration for population-level effect estimation studies in observational healthcare data." Proc. Natl. Acad. Sci. U.S.A. 115 (11): 2571–7.

Schuemie, M. J., P. B. Ryan, G. Hripcsak, D. Madigan, and M. A. Suchard. 2018. "Improving reproducibility by using high-throughput observational studies with empirical calibration." Philos Trans A Math Phys Eng Sci 376 (2128).

Naitee T. 2019. "Statistical Interactions in a Clinical Trial." Therapeutic Innovation & Regulatory Science 2018, Vol. 52(1) 14-21.

Strack et al. 2014. "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records," BioMed Research International, vol. 2014, Article ID 781670

Friedman, J.H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." The Annals of Statistics, 29, pp. 1189–1232.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2014). "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation." arXiv:1309.6392v2.

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Website Federal Reserve Bank of St. Louis. 2012. "Economic Research." Accessed November 7, 2012. http://research.stlouisfed.org.

ACKNOWLEDGMENTS

The author would like to thank SAS reviewers who provided manuscript review and editing support, Heather Weinstein, Robert Collins, and Hiwot Tesfaye for content review of the paper.

RECOMMENDED READING

- Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. https://christophm.github.io/interpretable-ml-book/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Olaleye
SAS Institute
David.Olaleye@sas.com

APPENDIX

PMFAC TOOL SAS® MACRO PROGRAM

The steps to use the PMFAC tool and generate the PMFAC plots are outlined below. Use cases of the PMFAC tool on real-world data are presented in the Results section of the paper.

1. The first step is to generate replicate samples from the observed data. Note that there are multiple ways of generating random samples, but for illustration purposes, we used the simple random sampling method.

```
/***Initiate CAS session ***/
   cas ;
   caslib _all_ assign;

   /* Create a CAS engine libref to save the output data sets */
   %let caslibname = mycaslib;

   /* Define a CAS engine libref for CAS in-memory data tables */
   libname mycaslib cas caslib=casuser;

   /* Specify the data set names */

   %let sasdata=%str(mysaslib.diab_cas_enctrs);
   %let casdata=%str(mycaslib.diab_cas_enctrs);
   %let grid=9;

   data &casdata.;
       length obsid grid_obsid 8;
       do obsnum=1 to n;
             set &sasdata. point=obsnum nobs=n;
             grid_obsid = ceil(&grid. * ranuni(1234));
             obsid=obsnum;
             if _error_ then abort;
             output;
       end;
       stop;
   run;
```

2. Optionally, create a partition variable in the resulting table if the modeling objective requires using a training/validation data set.

3. Optionally, perform the binning task on interval-level variables (or high-cardinality variables) that can be used as plot variables.

```
   /*************************************************************/
   /* Partition the data into training and validation           */
   /*************************************************************/
   proc partition data=&casdata partition samppct=60;
   by &target grid_obsid;
   output out=&casdata. copyvars=(_ALL_);
   run;

   proc binning data=&casdata. numbin=4 method=cutpts; *method=bucket;
   input number_inpatient / numbin=4 cutpts(1, 2, 3) LEVEL=INTERVAL;
   input number_diagnoses / numbin=4 cutpts(3, 6, 9) LEVEL=INTERVAL;
```

```
input time_in_hospital / cutpts(3, 6, 9) LEVEL=INTERVAL;
output out= &casdata. copyvars=(_ALL_);
run;
```

4. Fit the machine learning model of choice and use the "OUTMODEL=<CAS-libref.>data-table" option to save the ML model properties (in this example, a gradient boosting model) to a CAS table for scoring the replicate samples.

5. Determine the test statistics of interest based on the observed data. For the binary level target variable, the test quantities might include the average (mean) and the sample standard deviation (sd) or standard error (se). For the interval level target variable, additional test parameters might include the maximum (max) and the minimum value (min).

```
/*****************************************************************/
/* GRADIENT BOOSTING MACHINES predictive model                   */
/*****************************************************************/
proc gradboost data=&casdata. ntrees=100 intervalbins=20 maxdepth=5
               outmodel=mycaslib.gb_model_runtime VII=2;
  input &interval_inputs. / level = interval;
  input &class_inputs. / level = nominal;
  target &target / level=nominal;
  *partition rolevar=_partind_(train='1' validate='0');
  id obs_id;
  code file="&sascodedir./gb_select_score.sas";
  SAVESTATE RSTORE=mycaslib.gb_store_code_runtime;
  output out=mycaslib.gb_scored_ds_runtime copyvars=(_ALL_);
  ods output FitStatistics=addinrwe.gb_fitstats_runtime
VariableImportance=addinrwe.gb_VarImp_runtime;
run;

data addinrwe.gb_scored_ds_runtime;
    length Model_Type $15;
    set mycaslib.gb_scored_ds_runtime;
    Model_Type='GradientBoost';
run;
```

6. Run the PMFAC plot macro to score each replicate and store the scored samples in a data set. The tool also stores the model fit statistics for each replicate, which can be used for a post hoc model fitting diagnostics check.

7. Calculate the test statistics of choice, which for this illustration are the average and standard error of mean of the target variable based on the real data, and the test parameters (mean and standard error of the predicted target value) for each replicate sample. Assuming that the posterior predictive distribution of the predicted scores follows a normal distribution, a z-score test statistic and associated p-value are obtained for checking whether the test parameter for each replicate sample differs from that of the test statistic obtained from the real data. The observed p-value (depending on how extreme the value is) for a replicate sample might suggest the extent that the predicted results differ from or are similar to the actual observations.

8. Finally, a paneled histogram plot (for all replicates) is produced that shows the distribution of predicted scores for the observational unit (in this case, a diabetic patient), overlaid with line graphs for the observed and estimated test parameters for both the whole population data and each replicate sample. The p-value and N test parameters are overlaid on the plot for each sample. (See Figure 6.)

```
***The PMFAC Plot Replicate Samples Generator****;

%macro gen_pmfacplot_samples(mlproc=, mlname=, mlmodel=, targetvar=,
targetvar_value=, patid=);

proc sql noprint;
%if %sysfunc(exist(mycaslib.&mlmodel._scored_samples_all)) %then %do;
    drop table mycaslib.&mlmodel._scored_samples_all;
%end;

%if %sysfunc(exist(addinrwe.&mlmodel._fitstats_samples_all)) %then
%do;
    drop table addinrwe.&mlmodel._fitstats_samples_all;
%end;
quit;

%let j=0;
%do j=1 %to &grid.;

proc &mlproc. data=&casdata.(where=(grid_obsid=&j.))
inmodel=mycaslib.&mlmodel._model_runtime;
 output out=mycaslib.&mlmodel._outppdsample_&j. copyvars=(_ALL_);

%if (%kupcase(&mlname.) eq NNET) %then %do;
    ods output ScoreInfo=&mlmodel._fitstats_sample_&j.;
%end;
%else %do;
    ods output FitStatistics=&mlmodel._fitstats_sample_&j.;
%end;
run;

proc append base=addinrwe.&mlmodel._fitstats_samples_all
    data=&mlmodel._fitstats_sample_&j. force;
run;
%end;

%let k=0;

data addinrwe.&mlmodel._scored_samples_all;
set
%do k=1 %to &grid.;
    %if %sysfunc(exist(mycaslib.&mlmodel._outppdsample_&k.))  %then
%do;
          mycaslib.&mlmodel._outppdsample_&k.
    %end;
%end;
;
run;

data addinrwe.&mlmodel._scored_ds_runtime;
    length Model_Type $15;
    set mycaslib.&mlmodel._scored_ds_runtime;
    Model_Type="&mlname.";
run;

    /** ML model fit check starts here **/
```

```
proc hpsummary data=addinrwe.&mlmodel._scored_ds_runtime;
    var &target.;
    output out=&mlmodel._stats mean=mean_pop max=max_pop min=min_pop
stderr=se_pop;
run;

data _null_;
    set &mlmodel._stats;
    call symputx('mean_pop',mean_pop);
    call symputx('min_pop',min_pop);
    call symputx('max_pop',max_pop);
    call symputx('se_pop',se_pop);
run;

proc hpsummary data=addinrwe.&mlmodel._scored_samples_all;
    class grid_obsid;
    var p_&targetvar.&targetvar_value;
          types grid_obsid;
    output out=&mlmodel._sample_stats n=n mean=mean max=max min=min
stderr=se;
run;

data &mlmodel._scored_samples_check;
if _n_=1 then set &mlmodel._stats;
merge
addinrwe.&mlmodel._scored_samples_all(in=in1 keep=grid_obsid &patid.
p_&targetvar.&targetvar_value)
    &mlmodel._sample_stats end=eof;
by grid_obsid;

 if first.grid_obsid then do;
    zscore = (mean - mean_pop)/se_pop;
    pvalue = pdf('NORMAL', zscore);
 end;
else call missing(n, mean, min, max, se, zscore, pvalue);
run;

proc sql noprint;
    select min(p_&targetvar.&targetvar_value) as min_plotvalue,
          max(p_&targetvar.&targetvar_value) as max_plotvalue
    into :min_plotvalue trimmed, :max_plotvalue trimmed
    from &mlmodel._scored_samples_check;
quit;

ods graphics on / width=8in;

title "ML Predictive Model Results Check (ModelType = &mlname)";
proc sgpanel data=&mlmodel._scored_samples_check;
  panelby grid_obsid / columns=3 rows=3;
  histogram p_&targetvar.&targetvar_value / TRANSPARENCY=0.2 ;
  lineparm x=&mean_pop. y=0 slope=. / lineattrs=(color=red
thickness=3) LEGENDLABEL="Population"   NOEXTEND TRANSPARENCY=0.4;
  lineparm x=mean y=0 slope=. / lineattrs=(color=green thickness=3)
LEGENDLABEL="Sample" NOEXTEND     TRANSPARENCY=0.6;

  inset pvalue n / position=topright textattrs=(style=italic);
  colaxis min=&min_plotvalue. max=&max_plotvalue.;
```

```
label
    grid_obsid="Sample"
    pvalue='P-value'
    n='N';
run; quit;
title;

%mend gen_pmfacplot_samples;

%gen_pmfacplot_samples(mlproc=gradboost,
    mlname=GradientBoost,
    mlmodel=gb,
    targetvar=Readmission,
    targetvar_value=1,
    patid=patient_nbr );
```

Note: the macro requires the following input parameters:
Mlproc = SAS VIYA procedure name for the ML model (for example, gradboost or forest or nnet)
Mlname = name to use to identify the ML model on the plot (for example, GradientBoost)
Mlmodel = name to use to identify or reference the temporary ML data sets (for example, gb or fm)
Targetvar = name of the observed target variable to be modeled (for example, readmission)
targetvar_value = modeled value of the observed target variable (for example, 1)
patid = patient identifier (for example, patient_nbr)

Note: the macro program is a working prototype for this presentation and subject to modification. Final code will be published in GitHub.

PMFAC PLOT SAS® MACRO PROGRAM

```
/******POST-PREDICTION DISTRIBUTION (PPD) PLOTS- THE %PPDPLOT MACRO*****/

%macro PMFACPLOT(dataset_all=,
          dataset_sample=,
          targetvar=,
          targetvar_value=,
          PPDVars=,
          PPDVars_CLM=,
          outPPD=);

proc sql noprint;
    %if %sysfunc(exist(&outPPD._all)) %then %do;
          drop table &outPPD._all;
    %end;
    %if %sysfunc(exist(&outPPD._sample)) %then %do;
          drop table &outPPD._sample;
    %end;
quit;

data _null_;
    length PPDVar1 PPDVar2 $50 PPDVars $100 numPPDVars 8;
    PPDVars = kstrip(compbl("&PPDVars"));
    numPPDVars = kcountw(PPDVars);
    PPDVar1 = kstrip(kscan(ktrim(PPDVars),1));
    PPDVar2 = kstrip(kscan(ktrim(PPDVars),2));

    call symput('PPDVAR1', kstrip(PPDVAR1));
    call symput('PPDVAR2', kstrip(PPDVAR2));
    call symputx('numPPDVars', numPPDVars);
run;
%put PPDVar1 = &PPDVar1;
%put PPDVar2 = &PPDVar2;
%put numPPDVars = &numPPDVars;

/*Compute average predicted value for entire training dataset */
proc summary data = &dataset_all.;
    class &PPDVar1. &PPDVar2.;
    output out=&outPPD._all
    %if &numPPDVars = 1 %then %do; (where=(_type_ = 1)) %end;
    %if &numPPDVars = 2 %then %do; (where=(_type_ = 3)) %end;

    mean(p_&targetvar.&targetvar_value) = p_&targetvar._Mean
    stderr(p_&targetvar.&targetvar_value) =p_&targetvar._stderr
    LCLM(p_&targetvar.&targetvar_value) =  p_&targetvar._LCLM
    UCLM(p_&targetvar.&targetvar_value) =  p_&targetvar._UCLM ;

run;

proc sql noprint;
    select min(p_&targetvar._Mean) as min_plotvalue,
max(p_&targetvar._Mean) as max_plotvalue,
    min(p_&targetvar._LCLM) as min_plotvalue_LCLM, max(p_&targetvar._UCLM)
```

```
as max_plotvalue_UCLM
    into :min_plotvalue trimmed, :max_plotvalue trimmed,
          :min_plotvalue_LCLM trimmed, :max_plotvalue_UCLM trimmed
    from &outPPD._all;
quit;

    /* if treated as categorical/nominal variables */
proc sgplot data=&outPPD._all;
    series  y=p_&targetvar._Mean x=&PPDVar1. / markers
markerattrs=(size=5px)
    %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
          ;
    %if %kupcase(&PPDVars_CLM.) = Y %then %do;
    scatter y=p_&targetvar._Mean x=&PPDVar1. /
yerrorlower=p_&targetvar._LCLM
    yerrorupper=p_&targetvar._UCLM DATASKIN=CRISP
          %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
    ;
    yaxis min=&min_plotvalue_LCLM. max=&max_plotvalue_UCLM.;
    %end;

    %if %kupcase(&PPDVars_CLM.) = N %then %do;
          scatter y=p_&targetvar._Mean x=&PPDVar1. / DATASKIN=CRISP
          %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
          ;
          yaxis min=&min_plotvalue. max=&max_plotvalue.;
    %end;

    *label grid_obsid='PPDS-Sample';
run; quit;
title;

/*Compute average predicted value for each sample replicate */
proc summary data = &dataset_sample.;
    by grid_obsid;
    class &PPDVar1. &PPDVar2.;

    output out=&outPPD._sample
    %if &numPPDVars = 1 %then %do; (where=(_type_ = 1)) %end;
    %if &numPPDVars = 2 %then %do; (where=(_type_ = 3)) %end;

    mean(p_&targetvar.&targetvar_value) = p_&targetvar._Mean
    stderr(p_&targetvar.&targetvar_value) =p_&targetvar._stderr
    LCLM(p_&targetvar.&targetvar_value) =  p_&targetvar._LCLM
    UCLM(p_&targetvar.&targetvar_value) =  p_&targetvar._UCLM ;

run;

proc sql noprint;
    select min(p_&targetvar._Mean) as min_plotvalue,
max(p_&targetvar._Mean) as max_plotvalue,
          min(p_&targetvar._LCLM) as min_plotvalue_LCLM,
max(p_&targetvar._UCLM) as                    max_plotvalue_UCLM
    into :min_plotvalue trimmed, :max_plotvalue trimmed,
           :min_plotvalue_LCLM trimmed, :max_plotvalue_UCLM trimmed
    from &outPPD._sample;
quit;
```

```
/* if treated as categorical/nominal variables */
proc sgpanel data=&outPPD._sample;
    panelby grid_obsid / columns=3 rows=3;

    series  y=p_&targetvar._Mean x=&PPDVar1. / markers
markerattrs=(size=5px)
    %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
    ;
    %if %kupcase(&PPDVars_CLM.) = Y %then %do;
          scatter y=p_&targetvar._Mean x=&PPDVar1. /
yerrorlower=p_&targetvar._LCLM
    yerrorupper=p_&targetvar._UCLM DATASKIN=CRISP
          %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
          ;
          rowaxis min=&min_plotvalue_LCLM. max=&max_plotvalue_UCLM.;
    %end;

    %if %kupcase(&PPDVars_CLM.) = N %then %do;
          scatter y=p_&targetvar._Mean x=&PPDVar1. / DATASKIN=CRISP
                %if %eval(&numPPDVars = 2) %then %do; group=&PPDVar2. %end;
          ;
          rowaxis min=&min_plotvalue. max=&max_plotvalue.;
    %end;
    label grid_obsid='PPDS-Sample';
run; quit;
title;

%mend PMFACPLOT;

%LET mlmodel=gb;

** ONE-WAY PMFAC PLOT - One Categorical variable***;
%PMFACPLOT(dataset_all=%str(mycaslib.&mlmodel._scored_ds_runtime),
    dataset_sample=%str(mycaslib.&mlmodel._scored_samples_all),
    targetvar=readmission,
    targetvar_value=1,
    PPDVars=HbA1c_Status,
    PPDVars_CLM=N,
    outPPD=outPD_&mlmodel._plot_onevar);

** TWO-WAY PMFAC PLOT - Two Categorical variables***;
%PMFACPLOT(dataset_all=%str(mycaslib.&mlmodel._scored_ds_runtime),
    dataset_sample=%str(mycaslib.&mlmodel._scored_samples_all),
    targetvar=readmission,
    targetvar_value=1,
    PPDVars diabetesMed HbA1c_Status,
    PPDVars_CLM=N,
    outPPD=outPD1_&mlmodel._plot_twovars);

Note: macro input parameters required are:
dataset_all = the scored data set based on the entire training sample
dataset_sample = the scored replicate samples generated by the above macro
program
targetvar = target outcome variable to be modeled, e.g. readmission
targetvar_value = the target event for the outcome variable to be predicted
PPDVars = list of input features to use as plot-variables (the macro only
supports one or two plot-variables at this time)
```

PPDVars_CLM = option to add confidence limits to the line plots for the x-axis plot-variable (N=no, Y=yes)
outPPD = macro name for the plot-variables data set. May be used in other SAS applications to generate enhanced plots.

Note: the macro program is a working prototype for this presentation and subject to modification. Final code will be published in GitHub.