Model Comparison of Machine Learning Techniques to detect signs of Penetrating Abdominal and Pelvic Injuries using Base JMP® and R Integration

Melvin Alexander, Analytician

ABSTRACT

This presentation takes data from a medical radiology case study of Saksobhavivat et al. (2016) designed to detect signs from Multidetector Computed Tomography (MDCT) imaging in diagnosing and treating traumatic penetrating abdominal and pelvic injuries (PAPI).

Machine Learning techniques of Random Forests, Extreme Gradient Boosting, Stepwise Logistic, and Penalized Regression from the Base JMP® and R environment were used to compare these statistical models to determine the strongest signs as indicators of diagnosing PAPI following penetrating abdominal injury.

Although some of these tools are only available in JMP Pro®, the same (and more) capabilities are also available to Base JMP users.

I will demonstrate the R and Base JMP® integration to create several machine learning statistical models, some of which are not even available in JMP Pro®.

Base JMP® connects to R via JSL to execute R commands and exchange data. This presentation will demonstrate JMP and R integration that allows users to take advantage of the powerful capabilities in both tools.

Results of the image analyses helped radiologists and clinicians discriminate patients requiring surgery, observation, or non-operative management.

INTRODUCTION

Penetrating abdominal and pelvic injuries (PAPI) are uncommon, potentially life-threatening, trauma injuries resulting from stab wounds, gunshot wounds (GSW), or other types.

Multidetector computed tomography (MDCT), along with advances in other medical imaging technologies has made this method one of the major modes of emergency management of PAPI for injury detection and severity, replacing the need for conducting unnecessary surgical explorations and physical examination of traumatic PAPI patients.

Determining the accuracy of MDCT, when compared to the gold-standard surgical findings, has been an important and challenging area of study. Multiple studies have looked at the overall accuracy of triple contrast CT versus single-contrast CT assessment of penetrating torso or abdominopelvic trauma, especially as it pertains to detection of PAPI.

This presentation uses of JMP® and R Integration to identify key signs that are indicators of PAPI from MDCT using modern machine learning techniques.

MATERIALS AND METHODS

The University of Maryland Medical Center's Institutional Review Board (IRB) approved the prospective observational study. The written waiver of informed consent complied with the Health Insurance Portability and Accountability (HIPPA) regulations by the IRB.
Triple contrast (oral, rectal, and intravenous) MDCT has been commonly used as the primary means of evaluating penetrating abdominal and pelvic resulting from gunshot or stab wounds. Contrast media increases visibility of internal abdominal structures in CT imaging. Few studies have reported with high accuracy that triple contrast CT predicts the need for surgical treatment of penetrating abdominal and pelvic injuries.

CT images of 171 patients underwent MDCT imaging for surgery (77/171, 45.0%) or clinical follow-up (94/171, 55.0%) between October 2011 – April 2013 at the University of Maryland Medical Center's (UMMC) Shock Trauma Center. The images were interpreted by three independent radiologists, (one attending radiologist and two secondary readers). Each radiologist interpreted each patient scan and recorded findings on dedicated worksheets (Figure 2), blind to each other's imaging, clinical data, or patient's management outcomes.

Sixteen signs have been cited in the medical literature as key signs indicating PAPIs. Direct signs that indicated GastroIntestinal (GI) injury included GI wall discontinuity (Q7), subjective GI wall thickening (Q8), intramural air (Q4), bleeding into GI lumen (Q14), leakage of enteric contrast material (Q6), visible leakage of any GI content (Q5, if enteric contrast was not present at the injury site), and visible penetrating wound track (Q15 outlined by hemorrhage, air, and/or ballistic fragments) that extended up to the GI wall.

Indirect CT signs that were also evaluated included any evidence of peritoneal violation (Q1), retroperitoneal violation (Q2), free intraperitoneal/retroperitoneal gas *adjacent to* the GI injury site (Q3a), free intraperitoneal/retroperitoneal gas *remote to* the GI injury site (Q3b), peritoneal thickening or enhancement (Q12), co-existing penetrating injuries to intraperitoneal solid organs (Q13), free intraperitoneal fluid (Q9), mesenteric hematoma (Q10), and active mesenteric hemorrhage (Q11). The 17th overall CT diagnosis of GI injury (CToverall) rated the degree of overall confidence for the presence (or absence) of a PAPI. All signs used a 5-point confidence scale (1-definitely absent, 2-may be present but unlikely, 3-unequivocal, 4-likely present, 5-definitely present).

Cross-validation (i.e., the approach of avoiding overfitting that leads to poor prediction responses) divided the full dataset (513 observations of Table 1) into training and test data tables using an 80:20 split. Training data (411 or 80% of all observations, **Validation** = 0) built the regression models. Test data (102 or 20% of the remaining observations, **Validation** = 1) assessed the predictive accuracy of the regression models from training data with confusion matrices of the correct and misclassifications.

To integrate R and base JMP, follow these steps (Appendix 3 describes the Base SAS® and R interface):

1. Open the full data table and create the Train and Test subsets.

```
dt = Data Table( "pbi 11 for Mel Sep_26_2019" ) ;
/* Training data table */
dt << Select Where( :Validation == 0 ) <<
Subset(Output Table( "trainData" ),
columns(:surgery, :clinBI,:Q1, :Q2, :Q3a, :Q3b, :Q4,  :Q5, :Q6, :Q7,
 :Q8, :Q9, :Q10, :Q11, :Q12, :Q13, :Q14, :Q15, :Ctoverall ) );
```

```
 /* Test data */
 dt << Select Where( :Validation == 1 ) <<
Subset(Output Table( "testData" ),
columns(:surgery, :clinBI, :Q1, :Q2, :Q3a, :Q3b, :Q4, :Q5, :Q6, :Q7,
:Q8, :Q9, :Q10, :Q11, :Q12, :Q13, :Q14, :Q15, :Ctoverall  ) );

 dttrain = Data Table ("trainData") ;
 dttest = Data Table( "testData" ) ;
```

2. Create the R Environment that points JMP to the location where R is installed and send the Train and Test data tables as R objects:

```
/* create Environmental Varaible for R 3.5.0 */
Set Environment Variable( "R_HOME", "C:\Program Files\R\R-3.5.0" ) ;
R Init();
R Send ( dttrain );
R Send ( dttest) ;
R Send ( dt );
```

3. Submit the R code inside JMP's **R Submit ()** command as shown below:

```
R Submit(
"\[
     ###########
      R code  ;
     ###########
  ]\"
 );
    R term() ;
```

4. View the JMP log to see the output and results

Figure 1: Image from Saksobhavivat et al. [1] of Penetrating Abdominal Pelvic Injury (PAPI) (Reprinted with permission from the European Society of Radiology)

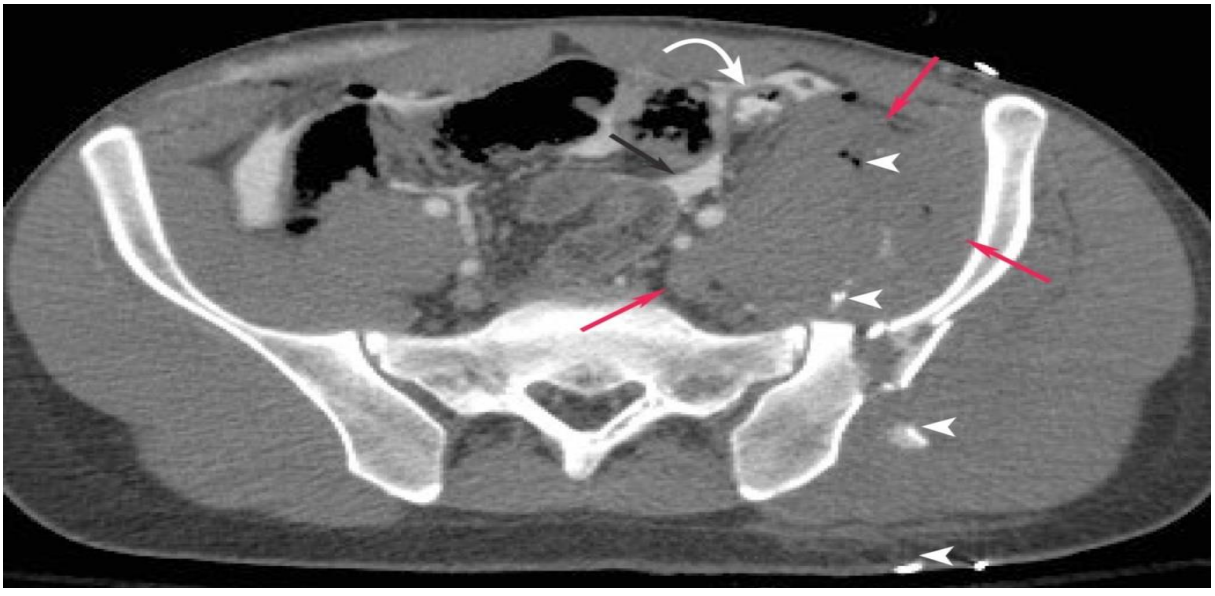


Figure 1 of Saksobhavivat et al. [1] showed an Axial (top-down) view of a gunshot wound to the left pelvis with rectal contrast material extravasation. Extravasation was where contrast media leaked into surrounding tissues. Reformatted CT images demonstrated a wound tract (Q15, arrowheads) outlined by hematoma, bullet and bone fragments. There was both a descending colon (Q13, curved arrow) and jejunal wall (Q8, white arrows) thickening. Rectal contrast material (Q6, red arrows) extravasation was seen throughout the peritoneum.

Hematoma is swelling of clotted blood within tissues. The jejungal wall makes up 20 percent of the small intestines and is used to evaluate the small bowel during follow-through evaluation (a.k.a. SBFT).

The arrows and arrow heads point to certain signs that radiologists saw that help diagnose patients and determine effective medical treatments.

Base JMP users connect well with R as data science principles continues to grow. JMP and R integration makes available more cutting-edge, machine learning tools like Random Forests, Extreme Gradient Boosting (XGBOOST), Penalized LASSO and Stepwise Logistic regression. This provides a more affordable alternative, at less expense, than having JMP PRO®. See Hastie, Tibshirani, Friedman [8] for more information about the various machine learning methods.

Figure 2 Worksheet with the Ordinal Scale used by independent radiologists to record data (Reprinted with permission from the Author)

Worksheet used to record data by All Readers for the Prospective Evaluation of Penetrating Bowel Injuries Study (Note: Standard Reader was the Attending Radiologist on call when the patient was admitted. All readers were blind to each other's findings and patient outcomes)

Radiologist Worksheet – to be completed on ***all*** patients with penetrating trauma to the torso ***at the time of initial scan interpretation***.

Radiologist Initials: _____
Date: _____
Time: _____
MRN: _____

Scan technique:
Was the scan performed with **IV** contrast? Y N
Was the scan performed with **oral** contrast? Y N
Was the scan performed with **rectal** contrast? Y N
If no, briefly explanation: _____

Are superficial entry/exit point(s) visible on CT? Y N
If yes, how many entry/exit point(s) are visible on CT? 1 2 3 4 5 more:____

Using the 5 point scale below, rate your confidence for the presence of the following findings:
1-definitely ***not*** present 2 – may be present but unlikely 3 – equivocal 4 – presence is likely 5 – finding is definitely present

1. Peritoneal violation	1	2	3	4	5	
2. Retroperitoneal violation	1	2	3	4	5	
3a. Free gas at the wound tract	1	2	3	4	5	
3b. Free gas away from the wound tract	1	2	3	4	5	
4. Intramural gas	1	2	3	4	5	
5. Leakage of luminal contents	1	2	3	4	5	
6. Leakage of oral/rectal contrast	1	2	3	4	5	
7. Discontinuity of bowel wall	1	2	3	4	5	
8. Bowel wall thickening (>4mm)	1	2	3	4	5	
9. Free fluid (without obvious source)	1	2	3	4	5	
10. Mesenteric hematoma	1	2	3	4	5	
11. Active bleeding in the mesentery	1	2	3	4	5	
12. Signs of peritonitis 13. Signs of solid organ injury 14. Intraluminal bleeding into bowel lumen 15. Wound tract extending up to the bowel	1	2	3	4	5	

Rate your degree of overall confidence, for presence or absence of a bowel injury, on a 5 point scale (1 – definitely no bowel injury, 2 – possible but unlikely presence of bowel injury, 3 – equivocal, 4 – presence of bowel injury is likely, 5 – definite bowel injury is present)

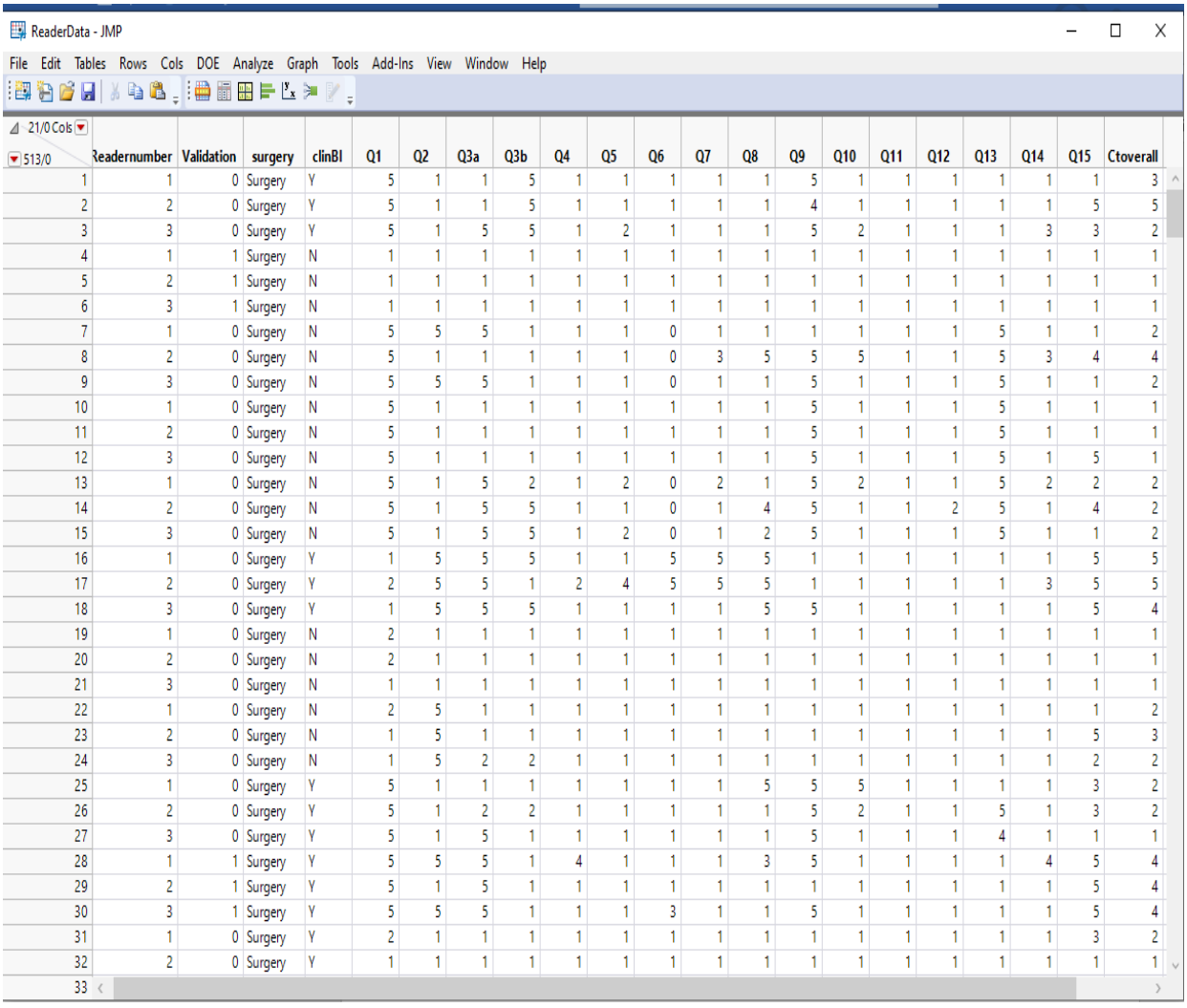
1 2 3 4 5

Do you think there is a bowel injury? Y N Unsure
Do you recommend surgery for bowel injury? Y N Unsure

The ordinal scale in Figure 2 was adopted because:

- past reviews of the literature only related signs to binary response outcomes (1= PBI presence, 0= PBI absence)
- it provided an improved measurement scale that extended beyond nominal, two-level outcomes of previous studies
- the scale was like scales I helped develop in other studies
- careful training was given to readers so that they could use the rating scale to score CT scans consistently and correctly (using reference images of "known" signs that appeared on images) in order to assure inter-rater reliability as reported by Alexander [4]
- this scale was useful for assessing reproducibility or variability among of image readers.

Table 1: Selected Records of the ReaderData JMP Data Table for Analysis



ReaderData - JMP

File Edit Tables Rows Cols DOE Analyze Graph Tools Add-Ins View Window Help

21/0 Cols

513/0

	Readernumber	Validation	surgery	clinBI	Q1	Q2	Q3a	Q3b	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Ctoverall
1	1	0	Surgery	Y	5	1	1	5	1	1	1	1	1	5	1	1	1	1	1	1	3
2	2	0	Surgery	Y	5	1	1	5	1	1	1	1	1	4	1	1	1	1	1	5	5
3	3	0	Surgery	Y	5	1	5	5	1	2	1	1	1	5	2	1	1	1	3	3	2
4	1	1	Surgery	N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	2	1	Surgery	N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	3	1	Surgery	N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	0	Surgery	N	5	5	5	1	1	1	0	1	1	1	1	1	1	5	1	1	2
8	2	0	Surgery	N	5	1	1	1	1	1	0	3	5	5	5	1	1	5	3	4	4
9	3	0	Surgery	N	5	5	5	1	1	1	0	1	1	5	1	1	1	5	1	1	2
10	1	0	Surgery	N	5	1	1	1	1	1	1	1	1	5	1	1	1	5	1	1	1
11	2	0	Surgery	N	5	1	1	1	1	1	1	1	1	5	1	1	1	5	1	1	1
12	3	0	Surgery	N	5	1	1	1	1	1	1	1	1	5	1	1	1	5	1	5	1
13	1	0	Surgery	N	5	1	5	2	1	2	0	2	1	5	2	1	1	5	2	2	2
14	2	0	Surgery	N	5	1	5	5	1	1	0	1	4	5	1	1	2	5	1	4	2
15	3	0	Surgery	N	5	1	5	5	1	2	0	1	2	5	1	1	1	5	1	1	2
16	1	0	Surgery	Y	1	5	5	5	1	1	5	5	5	1	1	1	1	1	1	5	5
17	2	0	Surgery	Y	2	5	5	1	2	4	5	5	5	1	1	1	1	1	3	5	5
18	3	0	Surgery	Y	1	5	5	5	1	1	1	1	5	5	1	1	1	1	1	5	4
19	1	0	Surgery	N	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	2	0	Surgery	N	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	3	0	Surgery	N	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	1	0	Surgery	N	2	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
23	2	0	Surgery	N	1	5	1	1	1	1	1	1	1	1	1	1	1	1	1	5	3
24	3	0	Surgery	N	1	5	2	2	1	1	1	1	1	1	1	1	1	1	1	2	2
25	1	0	Surgery	Y	5	1	1	1	1	1	1	1	5	5	5	1	1	1	1	3	2
26	2	0	Surgery	Y	5	1	2	2	1	1	1	1	1	5	2	1	1	5	1	3	2
27	3	0	Surgery	Y	5	1	5	1	1	1	1	1	1	5	1	1	1	4	1	1	1
28	1	1	Surgery	Y	5	5	5	1	4	1	1	1	3	5	1	1	1	1	4	5	4
29	2	1	Surgery	Y	5	1	5	1	1	1	1	1	1	1	1	1	1	1	1	5	4
30	3	1	Surgery	Y	5	5	5	1	1	1	3	1	1	5	1	1	1	1	1	5	4
31	1	0	Surgery	Y	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	2
32	2	0	Surgery	Y	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
33																					

Table 1 shows a subset of selected columns from full data table by the three readers.

Figure 3: Importance Plot of the Random Forest

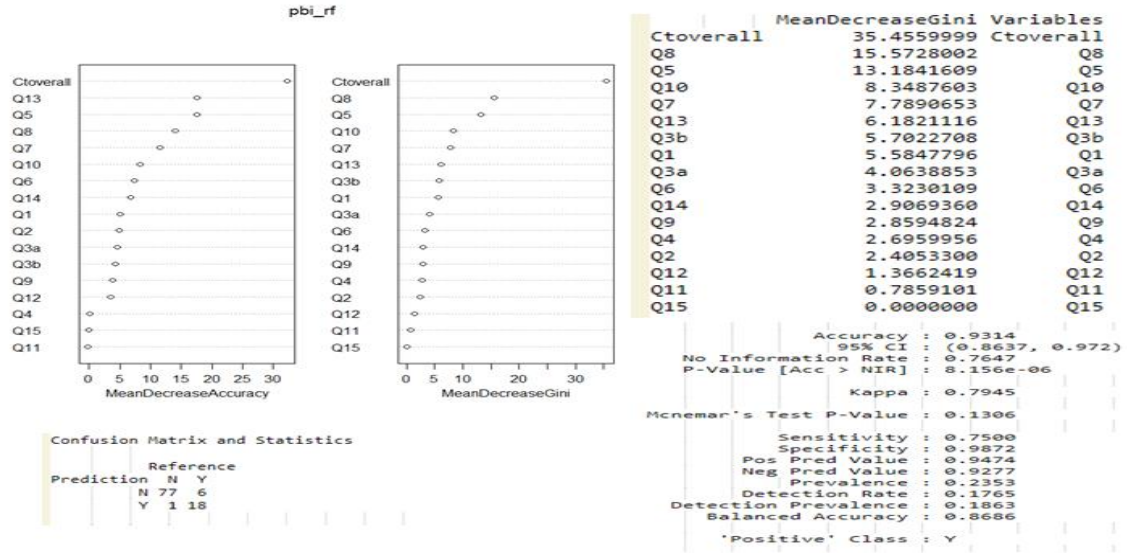


Figure 3 shows the Variable importance plot for pbi_rf (PAPI response variable) identifying the top six variables (Ctoverall, Q5, Q7, Q8, Q10, and Q13) based on the Model Accuracy (MeanDecreaseAccuracy) and Gini (MeanDecreaseGini) values. The table on the right listed variables in decreasing order of importance based on a measure (MeanDecreaseGini) for node or sign impurity.

The Confusion Matrix showed the predictive performance (Accuracy of 0.9314 or 93.14%) of the Random Forest Model on the Test dataset.

Figure 4: Decision Tree from The Decision Tree from R

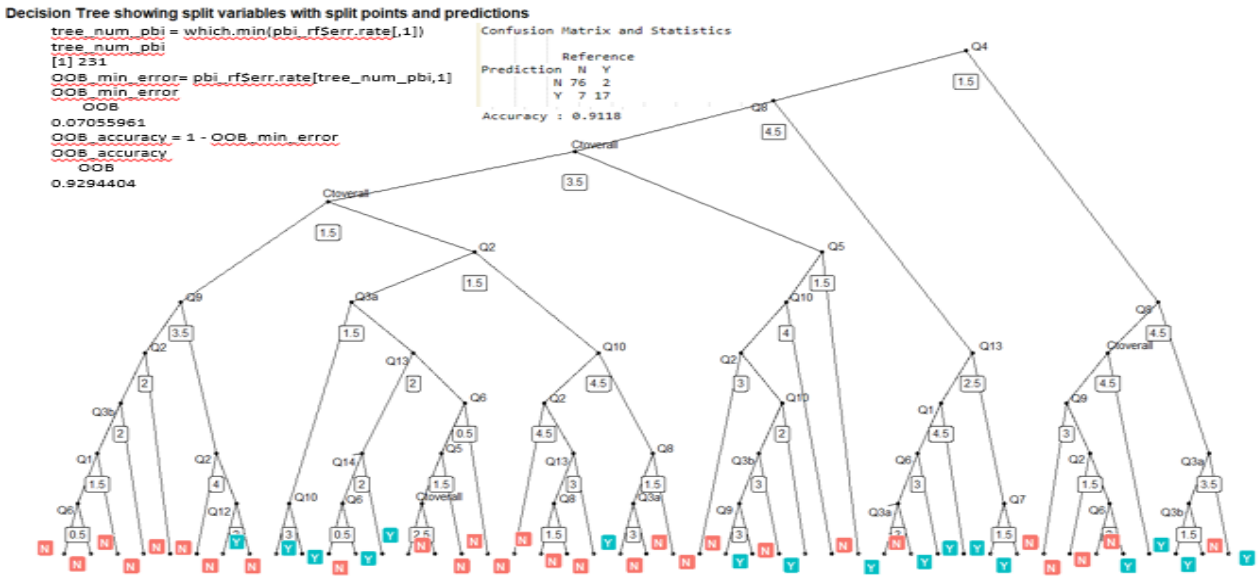


Figure 4 shows The Decision Tree with the smallest OOB error (tree number 231) had an out-of-bag prediction accuracy of 0.9294 (92.94%). The top three splitting variables in the decision tree are Q4, Q8, and Ctoverall. The Prediction Accuracy on the Test data was 0.9118 (91.2%).

Figure 5: Cross-validation Error (Binomial Deviance) by Log(lambda) Plot and Regression Coefficients from the LASSO Model, assessing the model accuracy against the Validation Test Data from the R GLMNET package

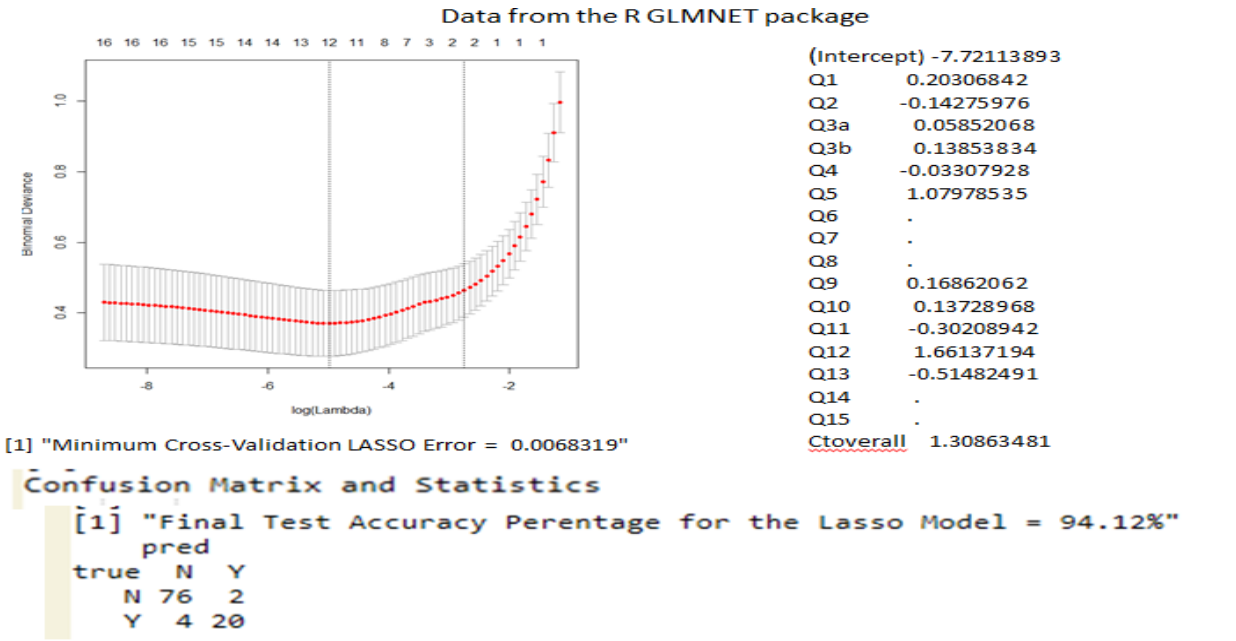


Figure 5 plots the cross-validation error according to the log of lambda. The left dashed vertical line indicates that the log of the optimal value of lambda is approximately -5, which is the one that minimizes the prediction error. This lambda value gave the most accurate model percentage of 94.12%.

Figure 6: Nominal Logistic on the Test data resulting from JMP's Stepwise platform

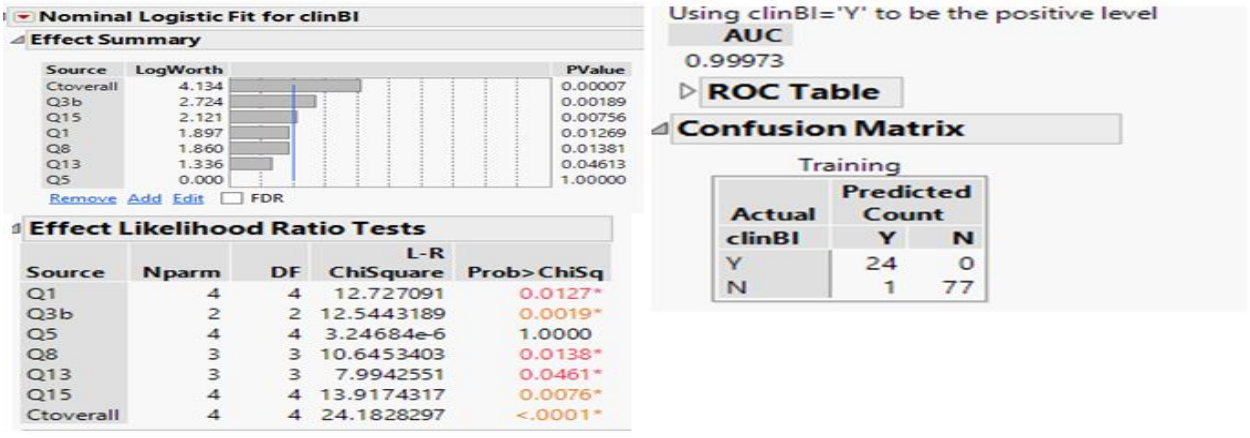
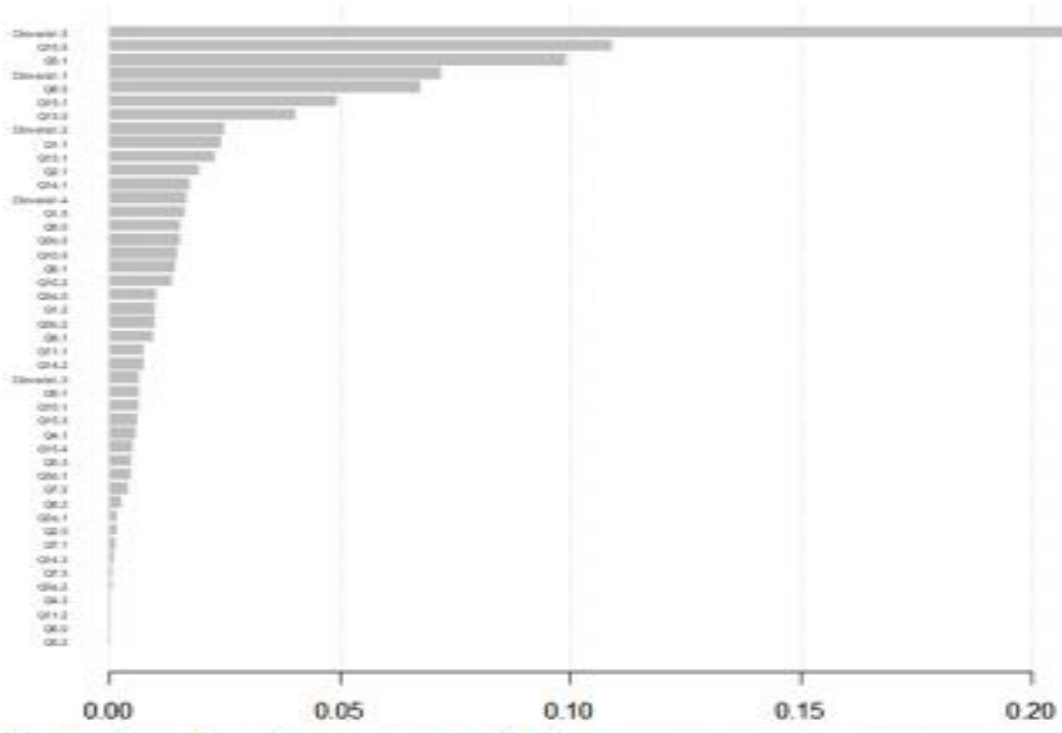


Figure 6 shows that Ctoveral, Q1, Q3b, Q8, Q13, and Q15 were the strongest signs with a prediction accuracy of 99.02% (100 * [24 + 77]/[24 + 77+ 1 + 0] = 100* 0.990196).

Figure 7: Relative Importance Variables from One Hot Encoding of XGBoost




```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 77  6
         1  1 18

               Accuracy : 0.9314
                 95% CI : (0.8637, 0.972)
    No Information Rate : 0.7647
    P-Value [Acc > NIR] : 8.156e-06

                  Kappa : 0.7945

 Mcnemar's Test P-Value : 0.1306

            Sensitivity : 0.7500
            Specificity : 0.9872
         Pos Pred Value : 0.9474
         Neg Pred Value : 0.9277
              Precision : 0.9474
                 Recall : 0.7500
                     F1 : 0.8372
             Prevalence : 0.2353
         Detection Rate : 0.1765
   Detection Prevalence : 0.1863
      Balanced Accuracy : 0.8686

       'Positive' Class : 1
```

Figure 7 shows that Ctoverall.5, Ctoverall.1 (Ctoverall ratings 5 and 1), Q15.5 (Q15 rating 5), Q5.1 (Q5 rating 1), and Q8.5 (Q8 rating 5) were the strongest signs with a prediction accuracy of 93.14% (100 * [77+18]/[77 + 18 + 1 + 6] = 100* 0.9314).

Table 2. Model Comparisons on the Test Data

Model	Confusion Matrix	Accuracy	Top Predictor Signs
LASSO (from glmnet of Appendix 1)	Pred true N Y N 76 2 Y 4 20	[1] "Final Test Accuracy Percentage for the Lasso Model = 94.12%"	Q1, Q2, Q3a, Q3b, Q4, Q5, Q9, Q10, Q11, Q12, Q13, Ctoverall
Random Forest (from randomForest of Appendix 1)	Reference Prediction N Y N 77 6 Y 1 18	[1] 0.9313725 or 93.14%	Ctoverall, Q8, Q5, Q10, Q7, Q13
XGBoost (Appendix 2 using One-Hot-Encoding)	Reference Prediction 0 1 0 77 6 1 1 18	Accuracy: 0.9314 or 93.14%	Ctoverall.5, Q15.5, Q5.1, Ctoverall.1, and Q8.5
Decision Tree (from rpart of Appendix 1)	Reference Prediction N Y N 76 2 Y 7 17	[1] "Final Test Accuracy Percentage for the Decision Tree Model = 91.18%"	Q4, Q8, Ctoverall, Q2, Q5, Q3
Stepwise Logistic (from JMP)	**Actual Predicted Count** **clinBI Y N** Y 24 0 N 1 77	0.990196 or 99.02%	Ctoverall, Q3b, Q15, Q1, Q8, Q13, Q9

CONCLUSION:

According to Table 2, the Stepwise Logistic model performed the best on the Test Data with a prediction accuracy of 99.02%. The next best models were the penalized LASSO, Random Forest, XGBOOST and the Rpart Decision Tree with prediction accuracies of 94.12%, 93.14%, 93.14%, and 91.18%, respectively.

The Stepwise Logistic model had a simpler mathematical form, was easier to interpret than the other models, and because of parsimony, was chosen as the most practical model to deploy. The strongest predictors from the Stepwise Logistic Model were: Ctoverall, Q1, Q3b, Q8, Q9, Q13, and Q15. Signs Q4, Q2, and Q10 are other indicators that radiologists and clinicians should especially notice also. The most accurate CT sign to indicate the need for

surgery for GI injury after penetrating torso trauma was Q8-GI wall thickening. Although Saksobhavivat et al. [1] found that the combination of Q8-GI wall thickening, Q6-leakage of GI content, and Q15-wound tract extending up to the bowel wall increased diagnostic accuracy, the signs of Ctoverall, Q5, Q8, Q10, and Q13 from the LASSO, Random Forest, XGBOOST, and Stepwise Logistic models provide additional indicators that radiologists should look for when viewing CT scans for PAPI.

This example showed how Model comparisons between different machine learning algorithms, excluding the advanced set of tools available in JMP Pro, can be performed using Base JMP and R integration. Appendix 3 described how Base SAS and R integration can be made. Integrating SAS and JMP with open source technologies like R gives analysts, data scientists, and statisticians many more advanced options to explore, gain insights, and make discoveries from data.

REFERENCES:

1. Saksobhavivat N, Shanmuganathan K, Boscak A, et al. (2016). "Diagnostic accuracy of triple-contrast multi-detector computed tomography for detection of penetrating gastrointestinal injury: a prospective study." *European Radiology*. 26(11): 4107-4120.
2. Ghumman, Z., Monteiro, S, et al. (2020). "Accuracy of Preoperative MDCT in Patients With Penetrating Abdominal and Pelvic Trauma." Canadian Association of Radiologist Journal, 2020. https://journals.sagepub.com/doi/full/10.1177/0846537119888375 (accessed 02/20/2020).
3. Alexander, M., (2016). "Exploring JMP®'s Image Visualization Tools in Medical Diagnostic Applications." SESUG 2016 Conference Proceedings, https://analytics.ncsu.edu/sesug/2016/RV-154_Final_PDF.pdf (accessed 06/27/2020).
4. Alexander, M (2014). "Using JMP® and R Integration to Assess Inter-rater Reliability in Diagnosing Penetrating Abdominal Injuries from MDCT Radiological Imaging." https://community.jmp.com/docs/DOC-6674 (accessed 04/27/2016).
5. Wei, X. (2012). "%PROC_R: A SAS Macro that Enables Native R Programming in the Base SAS Environment." *Journal of Statistical Software*, v.46, Code Snippet 2, https://www.jstatsoft.org/article/view/v046c02. (accessed 06/27/2020).
6. Alexander, M., (2019). "A Random Forest Example of the Boston Housing Data using the Base SAS® and the PROC_R macro in SAS® Enterprise Guide." Oct 23, 2019, *Proccedings of the 2019 SouthEast SAS® Users Group (SESUG) Conference*, Williamsburg, VA, October 20-22, 2019, https://www.lexjansen.com/sesug/2019/SESUG2019_Paper-158_Final_PDF.pdf (accessed 06/27/2020).
7. Hummell R (2019). "JMP® Synergies: Using JMP® and JMP® Pro With Python and R." https://www.jmp.com/en_us/whitepapers/jmp/jmp-synergies-using-jmp-and-jmp-pro-with-python-and-r.html (accessed 02/20/2020).
8. Hastie, T., Tibshirani, R., and Friedman, J., (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition, 12th Printing.* Springer-Verlag: New York, NY.

CONTACT INFORMATION

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

Melvin Alexander
Phone: (410) 458-7129
E-mail: melvin.alexander@verizon.net

Appendix 1: JMP JSL Script to Perform JMP and R integration

```
dt = Data Table( "pbi 11 for Mel Sep_26_2019" ) ;
/* Subset selected columns from full data table */
dt << Subset(
      Output Table( "ReaderData" ),
      Selected Rows( 0 ),
      columns( :Validation,
          :Readernumber, :surgery, :clinBI, :Q1, :Q2, :Q3a, :Q3b, :Q4,
            :Q5, :Q6, :Q7, :Q8, :Q9, :Q10, :Q11, :Q12, :Q13, :Q14,
            :Q15, :Ctoverall ) );

/* Training data table */

dt << Select Where( :Validation == 0 ) <<
  Subset(
      Output Table( "trainData" ),
            columns(
            :surgery, :clinBI,:Q1, :Q2, :Q3a, :Q3b, :Q4, :Q5, :Q6, :Q7, :Q8,
            :Q9, :Q10, :Q11, :Q12, :Q13, :Q14, :Q15, :Ctoverall ) );

/* Test data */
dt << Select Where( :Validation == 1 ) <<
  Subset(
      Output Table( "testData" ),
            columns(
            :surgery, :clinBI, :Q1, :Q2, :Q3a, :Q3b, :Q4,
            :Q5, :Q6, :Q7, :Q8, :Q9, :Q10, :Q11, :Q12,
            :Q13, :Q14, :Q15, :Ctoverall ) );


dttrain = Data Table ("trainData") ;
dttest = Data Table( "testData" ) ;

/* create Environmental Variable for R 3.5.0 */
Set Environment Variable( "R_HOME", "C:\Program Files\R\R-3.5.0" ) ;
R Init(); /* searches for R installed on the machine, initializes or opens it
*/
R Send ( dttrain ); /* send the training data table to R */
R Send ( dttest) ; /* send the test data table to R */
R Send ( dt ); /* send the full data to R */
R Submit(
"\[
```

```
ptm <- proc.time()
# turn JMP data tables (dttrain, dttest) into the R data frames (xtrain
xtest)
# assign new column names for training and test data frames
colnames_train_test <- c("surgery", "clinBI","Q1","Q2","Q3a","Q3b",
            "Q4","Q5", "Q6", "Q7", "Q8","Q9", "Q10", "Q11", "Q12", "Q13",
            "Q14", "Q15", "Ctoverall")

xtrain <- as.data.frame(dttrain)

#colnames(xtrain) <- colnames_train_test
# Change the variables into the forms for Random
# Forest classification with the transform function
head(xtrain)
xtrain <- transform(
  xtrain,
  surgery=as.factor(surgery),  BI_numeric=as.factor(clinBI),
  Q1=as.integer(Q1), Q2=as.integer(Q2), Q3a=as.integer(Q3a),
  Q3b=as.integer(Q3b), Q4=as.integer(Q4), Q5=as.integer(Q5),
  Q6=as.integer(Q6), Q7=as.integer(Q7), Q8=as.integer(Q8),
  Q9=as.integer(Q9), Q10=as.integer(Q10), Q11=as.integer(Q11),
  Q12=as.integer(Q12), Q13=as.integer(Q13), Q14=as.integer(Q14),
  Q15=as.integer(15), Ctoverall=as.integer(Ctoverall))
xtest <- as.data.frame(dttest)
xtest <- transform(
  xtest,
  surgery=as.factor(surgery), BI_numeric=as.factor(clinBI),
  Q1=as.integer(Q1), Q2=as.integer(Q2), Q3a=as.integer(Q3a),
  Q3b=as.integer(Q3b), Q4=as.integer(Q4), Q5=as.integer(Q5),
  Q6=as.integer(Q6), Q7=as.integer(Q7), Q8=as.integer(Q8),
  Q9=as.integer(Q9), Q10=as.integer(Q10), Q11=as.integer(Q11),
  Q12=as.integer(Q12), Q13=as.integer(Q13), Q14=as.integer(Q14),
  Q15=as.integer(15), Ctoverall=as.integer(Ctoverall) )
# assign new column names
#colnames(xtest) <- colnames_train_test
head(xtest)


nopred <- ncol(xtrain) - 2
nopred

my_model <- as.factor(clinBI) ~ Q1 + Q2 + Q3a + Q3b +
       Q4 + Q5 + Q6 +
       Q7 + Q8 + Q9 + Q10 + Q11 + Q12 + Q13 +
       Q14 + Q15 + Ctoverall

my_model

# load randomForest and ggplot2 libraries
library (randomForest)
library (caret)
library(e1071)
library (ggplot2)
set.seed(924)


trainData <- xtrain
```

```
testData <- xtest
tail(trainData)
#oob.err=double(nopred)
#test.err=double(nopred)



pbi_rf = randomForest(my_model, data=trainData, ntree=500, importance=T,
proximity=T)
table(predict(pbi_rf), trainData$clinBI)

print(pbi_rf)

x11()
plot(pbi_rf)

importance(pbi_rf)
var.imp <- data.frame(importance(pbi_rf,
           type=2))
# make row names as columns
var.imp$Variables <- row.names(var.imp)
var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),]
#var.imp[order(var.imp$MeanDecreaseAccuracy,decreasing = T),]
x11()
varImpPlot(pbi_rf)

pbiPred = predict(pbi_rf, newdata=testData)
table(testData$clinBI, pbiPred)

# S3 method for randomForest
# from https://www.rdocumentation.org/packages/randomForest/versions/4.6-
14/topics/predict.randomForest
pbi_pred_rf = predict(pbi_rf, newdata=testData, type="response",
  norm.votes=TRUE, predict.all=TRUE, proximity=TRUE, nodes=TRUE)
#pbi_pred_rf

#table(testData$clinBI,pbi_pred_rf)

#x11()
#plot(pbi_rf, testData$clinBI)

CM = table(pbiPred, testData$clinBI)
CM
accuracy = (sum(diag(CM)))/sum(CM)

accuracy

# make predictions and classification variables the same class
# so that we get the performance evaluation statistics of the
# confusion matrix

pbiPred = as.factor(pbiPred)
#testData$clinBi <- as.factor(testData$clinBI)
confusionMatrix(pbiPred, testData$clinBI, positive='Y')

# Calculate the size of each of the data sets:
data_set_size_train <- floor(nrow(trainData))
```

```
data_set_size_test <- floor(nrow(testData))
# Generate a random sample of "data_set_size" indexes
indexes_train <- sample(1:nrow(trainData), size = data_set_size_train)
indexes_test <- sample(1:nrow(testData), size = data_set_size_test)

# Assign the data to the correct sets
training <- trainData[indexes_train,]
validation1 <- testData[indexes_test,]

#import the package

# Validation set assessment #1: looking at confusion matrix
prediction_for_table <- predict(pbi_rf,validation1[,2:19])
#prediction_for_table
#validation1
table(observed=validation1$clinBI,predicted=prediction_for_table)

x11()
# Validation set assessment #2: ROC curves and AUC
# Needs to import ROCR package for ROC curve plotting:
library(ROCR)
# Calculate the probability of new observations belonging to each class
# prediction_for_roc_curve will be a matrix with dimensions data_set_size x
number_of_classes
prediction_for_roc_curve <- predict(pbi_rf,validation1[,2:19],type="prob")
#prediction_for_roc_curve
# Use pretty colours:
pretty_colours <- c("#F8766D","#00BA38")
# Specify the different classes
classes <- levels(validation1$clinBI)

# For each class
for (i in 1:2)
{
 # Define which observations belong to class[i]
 true_values <- ifelse(validation1[,2]==classes[i],1,0)
 # Assess the performance of classifier for class[i]
 pred <- prediction(prediction_for_roc_curve[,i],true_values)
 perf <- performance(pred, "tpr", "fpr")
 if (i==1)
 {
     plot(perf,main="ROC Curve",col=pretty_colours[i])
          legend(x = "bottomright",
          col = c("#F8766D","#00BA38"), lty = 1, lwd = 3,
          legend = c("N", "Y"))
 }
 else
 {
     plot(perf,main="ROC Curve",col=pretty_colours[i],add=TRUE)
          legend(x = "bottomright",
          col = c("#F8766D","#00BA38"), lty = 1, lwd = 3,
          legend = c("N", "Y"))
 }
 # Calculate the AUC and print it to screen
 auc.perf <- performance(pred, measure = "auc")
 print(auc.perf@y.values)
}
```

```
 # Calculate the AUC and print it to screen
 auc.perf <- performance(pred, measure = "auc")

# from https://www.r-bloggers.com/a-small-introduction-to-the-rocr-package/
# get the overall accuracy and plot it
acc.perf = performance(pred, measure = "acc")

plot(acc.perf)

# grab the maximum accuracy and cutoff
ind = which.max( slot(acc.perf, "y.values")[[1]] )
acc = slot(acc.perf, "y.values")[[1]][ind]
cutoff = slot(acc.perf, "x.values")[[1]][ind]
print(c(accuracy= acc, cutoff = cutoff))
print(paste("Final Test Accuracy Percentage for the Decision Tree Model
=",sprintf("%1.2f%%", 100*(acc))))

# https://stackoverflow.com/questions/28443834/how-to-plot-a-roc-curve-using-
rocr-package-in-r-with-only-a-classification-con
#tables <- lapply(seq(0, 1, .1), function(x) table(validation1$clinBI,
factor(predict(pbi_rf, type="response") >= x, levels=c(F, T))))
#tables[[1]]

#str(pbi_rf)

# from https://stats.stackexchange.com/questions/2344/best-way-to-present-a-
random-forest-in-a-publication

to.dendrogram <- function(dfrep,rownum=1,height.increment=0.1){

  if(dfrep[rownum,'status'] == -1){
    rval <- list()

    attr(rval,"members") <- 1
    attr(rval,"height") <- 0.0
    attr(rval,"label") <- dfrep[rownum,'prediction']
    attr(rval,"leaf") <- TRUE

  }else{##note the change "to.dendrogram" and not "to.dendogram"
    left <- to.dendrogram(dfrep,dfrep[rownum,'left
daughter'],height.increment)
    right <- to.dendrogram(dfrep,dfrep[rownum,'right
daughter'],height.increment)
    rval <- list(left,right)

    attr(rval,"members") <- attr(left,"members") + attr(right,"members")
    attr(rval,"height") <- max(attr(left,"height"),attr(right,"height")) +
height.increment
    attr(rval,"leaf") <- FALSE
    attr(rval,"edgetext") <- dfrep[rownum,'split var']
    #To add Split Point in Dendrogram
    #attr(rval,"edgetext") <- paste(dfrep[rownum,'split
var'],"\n<",round(dfrep[rownum,'split point'], digits = 2),"=>", sep = " ")
  }

  class(rval) <- "dendrogram"
```

```
  return(rval)
}

#find which number of trees providing the lowest error
# number of trees with lowest error rate
#which.max(pbi_rf$votes)

#pbi_rf$err.rate
which.min(pbi_rf$err.rate[,1])

# print confustion matrix and prediction accuracy
pbi_rf$confusion[1:2,1:2]
#print(paste("Final Test Accuracy Percentage for the Decision Tree Model
=",sprintf("%1.2f%%", 100*(1-max(pbi_rf$err.rate[,1])))))


#plot.tree <- getTree(pbi_rf, 1)
#plot.tree
plot.tree <- getTree(pbi_rf,k=which.min(pbi_rf$err.rate[,1]), labelVar=TRUE)
plot.tree

x11()
d2 <- to.dendrogram(plot.tree)
plot(d2,center=TRUE,leaflab='none',edgePar=list(t.cex=1,p.col=NA,p.lty=0))

library("rpart") ## recursive partitioning
m <- rpart(my_model, data = trainData,
           method = "class")

print(m)
printcp(m)

#plotcp(m)

pred = predict(m, type="class")
#pred
#pred <- predict(object = m, newdata = validation1, type = "class")
# from https://stackoverflow.com/questions/39620287/how-to-create-a-
confusion-matrix-for-a-decision-tree-model
pred1 <- predict(m, validation1, type="class")
pred1
confusionMatrix(validation1$clinBI, pred1, positive='Y')
str(pred)
summary(pred)

library("rpart.plot")
rpart.plot(m)

#table(training,pred)

## From "Plotting trees from Random Forest models with ggraph" 16Mar2017
## https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph

library(dplyr)
library(ggraph)
library(igraph)
```

```
#get tree number with the minimun OOB random forest error rate, tree_num =
which.min(pbi_rf$err.rate[,1])
tree_num_pbi = which.min(pbi_rf$err.rate[,1])

## From "Plotting trees from Random Forest models with ggraph" 16Mar2017
## https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph

tree_func <- function(final_model,
                      tree_num) {

  # get tree by index
  tree <- randomForest::getTree(final_model,
                                k = tree_num_pbi,
                                labelVar = TRUE) %>%
    tibble::rownames_to_column() %>%
    # make leaf split points to NA, so the 0s won't get plotted
    mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))

  # prepare data frame for graph
  graph_frame <- data.frame(from = rep(tree$rowname, 2),
                            to = c(tree$`left daughter`, tree$`right
daughter`))

  # convert to graph and delete the last node that we don't want to plot
  graph <- graph_from_data_frame(graph_frame) %>%
    delete_vertices("0")

  # set node labels
  V(graph)$node_label <- gsub("_", " ", as.character(tree$`split var`))
  V(graph)$leaf_label <- as.character(tree$prediction)
  V(graph)$split <- as.character(round(tree$`split point`, digits = 2))

  # plot
  x11()
  plot <- ggraph(graph, 'dendrogram', circular = FALSE) +
    theme_bw() +  geom_edge_link() + geom_node_point() +
    geom_node_text(aes(label = node_label), na.rm = TRUE, repel = TRUE) +
    geom_node_label(aes(label = split), vjust = 2.5, na.rm = TRUE,
    fill = "white") +  geom_node_label(aes(label = leaf_label,
    fill = leaf_label), na.rm = TRUE, repel = TRUE, colour = "white",
    fontface = "bold", show.legend = FALSE) +
    theme(panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          panel.background = element_blank(),
          plot.background = element_rect(fill = "white"),
          panel.border = element_blank(),
          axis.line = element_blank(),
          axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 10))

  print(plot)
}
```

```
tree_func(final_model = pbi_rf, tree_num_pbi)

# use glmnet library to generate penalized lasso regression
# http://www.sthda.com/english/articles/36-classification-methods-
essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-
elastic-net/
# lasso regression forces regression coefficients of less contributive
variables to zero.
# Only the most significant variables remain in the final model.

library(glmnet)

x <- model.matrix(my_model , trainData )[,-1]
'%ni%'<-Negate('%in%') # '%ni%' uses the %Negate function to denote do not
include
y <- ifelse(trainData$clinBI == "Y", 1, 0)
#glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)

glmmod <- glmnet(x, y=as.factor(trainData$clinBI), alpha=1,
family="binomial")
glmmod

coef(glmmod)

cv.lasso <- cv.glmnet(x, y=as.factor(trainData$clinBI), alpha = 1, family =
"binomial")
# Plot the cross-validation error according to the log of lambda. The left
dashed
# vertical line indicates that the log of the optimal value of lambda is
# approximately -5, which is the one that minimizes the prediction error.
# This lambda value will give the most accurate model
x11()
plot(cv.lasso)

# Display the cross-validation error according to the log of lambda.

cv.lasso$lambda.min
print(paste("Minimum Cross-Validation LASSO Error = ",sprintf("%1.7f%%",
cv.lasso$lambda.min)))


# Find the value of lambda that adjusts the amount of coefficient shrinkage
and
# gives the simplest model that lies within one standard error of the optimal
lambda value.
cv.lasso$lambda.1se

# Use lambda.min as the best lambda that gives the regression coefficients:

c <- coef(cv.lasso, cv.lasso$lambda.min)
c
inds<-which(c!=0)
variables<-row.names(c)[inds]
variables<-variables[variables %ni% '(Intercept)']
# list the contributory variables of the LASSO model that gives the best
solution
variables
```

```
# Make predictions on the test data by computing the final model using
# lambda.min and then assess the model accuracy
# against the test data, fitting the model using lambda = lambda.1se
# Final model with lambda.min

lasso.model <- glmnet(x, y, alpha = 1, family = "binomial",
                      lambda = cv.lasso$lambda.min)
# Make prediction on test data
x.test <- model.matrix(my_model, testData)[,-1]
probabilities <- lasso.model %>% predict(newx = x.test)
predicted.classes <- ifelse(probabilities > 0.5, "Y", "N")
# Model accuracy
observed.classes <- testData$clinBI
mean(predicted.classes == observed.classes)

print(paste("Final Test Accuracy Percentage for the Lasso Model
=",sprintf("%1.2f%%", 100*mean(predicted.classes == observed.classes))))

#table(predicted.classes, testData$clinBI)

table(true = testData$clinBI, pred = predicted.classes)

proc.time() - ptm

]\"
);

R term();
```

Appendix 2: JMP JSL Script to Perform JMP and R integration for XGBOOST:

```
// Create one hot encoding using the Make Indicator Columns function in JSL
// Subset the pbi data
Data Table( "pbi 11 for Mel Sep_26_2019" ) << Subset(
      Output Table( "Subset of pbi 11 for Mel Sep_26_2019" ),
      columns(

      :Validation, :BI_numeric, :Q1, :Q2, :Q3a, :Q3b, :Q4, :Q5, :Q6, :Q7,
      :Q8,:Q9, :Q10, :Q11, :Q12, :Q13, :Q14, :Q15, :Ctoverall ) ) ;

dt = DataTable("Subset of pbi 11 for Mel Sep_26_2019");

// Make sure all signs Q1-Ctoverall are set to numeric-nominal starting at
column 3
colList = dt << get column names(numeric);

For(i=3,i<=n items(colList), i++,
column(dt, colList[i]) << data type( Numeric)  << Set Modeling Type
(Nominal);
);

// Use the Make Indicator Columns function to create one hot encoded
variables used in
// the R XGBOOST package for each sign
```

```
dt <<  Make Indicator
Columns( columns( {:Q1,:Q2, :Q3a, :Q3b,:Q4, :Q5, :Q6, :Q7, :Q8, :Q9, :Q10, :Q
11, :Q12, :Q13, :Q14, :Q15, :Ctoverall} ), append column name(1), include
missing(1) );


// The Regex function replaces underscores ("_") with periods (".") for one
hot
// encoded variables used by R
For(i = 3, i <= N Col(dt), i++,
    Column(dt, i) << set name(Regex(Column(dt, i) << get name, "_", ".",
GLOBALREPLACE)) );

dt << Set Name ("One Hot Encode from JMP to R") ;
dt = Data Table( "One Hot Encode from JMP to R" ) ;

/* Training data from one-hot-encoded transformed columns*/
dt << Select Where( :Validation == 0 ) ;

dt<<
  Subset(
      Output Table( "trainohe" ),
columns(
            :BI_numeric, :Q1.1, :Q1.2, :Q1.3, :Q1.4, :Q1.5,
            :Q2.1, :Q2.2, :Q2.3, :Q2.4, :Q2.5, :Q3a.1,
            :Q3a.2, :Q3a.3, :Q3a.4, :Q3a.5, :Q3b.1, :Q3b.2,
            :Q3b.3, :Q3b.4, :Q3b.5, :Q4.1,  :Q4.2,    :Q4.3,
            :Q4.4,  :Q4.5, :Q5.1, :Q5.2, :Q5.3, :Q5.4, :Q5.5,
            :Q6.0,  :Q6.1, :Q6.2, :Q6.3, :Q6.5, :Q6.4, :Q7.1,
            :Q7.2,  :Q7.3, :Q7.4, :Q7.5, :Q8.1, :Q8.2, :Q8.3,
            :Q8.4,  :Q8.5, :Q9.1, :Q9.2, :Q9.3, :Q9.4, :Q9.5,
            :Q10.1, :Q10.2, :Q10.3,:Q10.4, :Q10.5, :Q11.0,
            :Q11.1, :Q11.2, :Q11.3, :Q11.4, :Q11.5, :Q12.1,
            :Q12.2, :Q12.3, :Q13.1, :Q13.2, :Q13.3, :Q13.4,:Q13.5,
            :Q14.0, :Q14.1, :Q14.2, :Q14.3, :Q14.4, :Q14.5,
            :Q15.1, :Q15.2, :Q15.3, :Q15.4, :Q15.5, :Ctoverall.1,
            :Ctoverall.2, :Ctoverall.3, :Ctoverall.4, :Ctoverall.5 ) );

/* Test data */

dt << Select Where( :Validation == 1 ) <<
  Subset(
      Output Table( "testohe" ),
columns(
            :BI_numeric, :Q1.1, :Q1.2, :Q1.3, :Q1.4, :Q1.5,
            :Q2.1, :Q2.2, :Q2.3, :Q2.4, :Q2.5, :Q3a.1,
            :Q3a.2, :Q3a.3, :Q3a.4, :Q3a.5, :Q3b.1, :Q3b.2,
            :Q3b.3, :Q3b.4, :Q3b.5, :Q4.1,  :Q4.2,    :Q4.3,
            :Q4.4,  :Q4.5, :Q5.1, :Q5.2, :Q5.3, :Q5.4, :Q5.5,
            :Q6.0,  :Q6.1, :Q6.2, :Q6.3, :Q6.5, :Q6.4, :Q7.1,
            :Q7.2,  :Q7.3, :Q7.4, :Q7.5, :Q8.1, :Q8.2, :Q8.3,
            :Q8.4,  :Q8.5, :Q9.1, :Q9.2, :Q9.3, :Q9.4, :Q9.5,
            :Q10.1, :Q10.2, :Q10.3,:Q10.4, :Q10.5, :Q11.0,
            :Q11.1, :Q11.2, :Q11.3, :Q11.4, :Q11.5, :Q12.1,
            :Q12.2, :Q12.3, :Q13.1, :Q13.2, :Q13.3, :Q13.4,:Q13.5,
            :Q14.0, :Q14.1, :Q14.2, :Q14.3, :Q14.4, :Q14.5,
```

```
            :Q15.1, :Q15.2, :Q15.3, :Q15.4, :Q15.5, :Ctoverall.1,
            :Ctoverall.2, :Ctoverall.3, :Ctoverall.4, :Ctoverall.5
      ) );

dttrainBoost = Data Table ("trainohe") ;
dttestBoost = Data Table( "testohe" ) ;
/* create Environmental Variable for R 3.5.0 */
Set Environment Variable( "R_HOME", "C:\Program Files\R\R-3.5.0" ) ;
R Init();
R Send ( dttrainBoost );
R Send ( dttestBoost) ;
R Send ( dt );
R Submit(
"\[
# turn JMP data tables (dttrainBoost, dttestBoost) into the R data frames
# (xtrainBoost xtestBoost)
# assign new column names for training and test data frames
colnames_train_test_Boost <- c("BI_numeric","Q1","Q2","Q3a","Q3b",
            "Q4","Q5", "Q6", "Q7", "Q8","Q9", "Q10", "Q11",
            "Q12", "Q13", "Q14", "Q15", "Ctoverall")

xtrainBoost <- as.data.frame(dttrainBoost)

#colnames(xtrain) <- colnames_train_test
# Change the variables into the forms for Random
# Forest classification with the transform function
head(xtrainBoost)
xtrainBoost <- transform(
  xtrainBoost,
BI_numeric=as.numeric(BI_numeric),
Q1.1=as.numeric(Q1.1), Q1.2=as.numeric(Q1.2), Q1.3=as.numeric(Q1.3),
Q1.4=as.numeric(Q1.4),Q1.5=as.numeric(Q1.5), Q2.1=as.numeric(Q2.1),
Q2.2=as.numeric(Q2.2),Q2.3=as.numeric(Q2.3),Q2.4=as.numeric(Q2.4),
Q2.5=as.numeric(Q2.5),Q3a.1=as.numeric(Q3a.1),Q3a.2=as.numeric(Q3a.2),Q3a.3=a
s.numeric(Q3a.3), Q3a.4=as.numeric(Q3a.4),
Q3a.5=as.numeric(Q3a.5),Q3b.1=as.numeric(Q3b.1),
Q3b.2=as.numeric(Q3b.2), Q3b.3=as.numeric(Q3b.3),
Q3b.4=as.numeric(Q3b.4),Q3b.5=as.numeric(Q3b.5),
Q4.1=as.numeric(Q4.1), Q4.2=as.numeric(Q4.2), Q4.3=as.numeric(Q4.3),
Q4.4=as.numeric(Q4.4),Q4.5=as.numeric(Q4.5), Q5.1=as.numeric(Q5.1),
Q5.2=as.numeric(Q5.2), Q5.3=as.numeric(Q5.3),
Q5.4=as.numeric(Q5.4), Q5.5=as.numeric(Q5.5), Q6.0=as.numeric(Q6.0),
Q6.1=as.numeric(Q6.1),Q6.2=as.numeric(Q6.2), Q6.3=as.numeric(Q6.3),
Q6.5=as.numeric(Q6.5), Q6.4=as.numeric(Q6.4),
Q7.1=as.numeric(Q7.1), Q7.2=as.numeric(Q7.2), Q7.3=as.numeric(Q7.3),
Q7.4=as.numeric(Q7.4), Q7.5=as.numeric(Q7.5), Q8.1=as.numeric(Q8.1),
Q8.2=as.numeric(Q8.2), Q8.3=as.numeric(Q8.3),   Q8.4=as.numeric(Q8.4),
Q8.5=as.numeric(Q8.5), Q9.1=as.numeric(Q9.1), Q9.2=as.numeric(Q9.2),
Q9.3=as.numeric(Q9.3), Q9.4=as.numeric(Q9.4), Q9.5=as.numeric(Q9.5),
Q10.1=as.numeric(Q10.1),Q10.2=as.numeric(Q10.2), Q10.3=as.numeric(Q10.3),
Q10.4=as.numeric(Q10.4), Q10.5=as.numeric(Q10.5),Q11.0=as.numeric(Q11.0),
Q11.1=as.numeric(Q11.1), Q11.2=as.numeric(Q11.2), Q11.3=as.numeric(Q11.3),
Q11.4=as.numeric(Q11.4), Q11.5=as.numeric(Q11.5), Q12.1=as.numeric(Q12.1),
Q12.2=as.numeric(Q12.2), Q12.3=as.numeric(Q12.3), Q13.1=as.numeric(Q13.1),
Q13.2=as.numeric(Q13.2), Q13.3=as.numeric(Q13.3),
```

```
Q13.4=as.numeric(Q13.4), Q13.5=as.numeric(Q13.5), Q14.0=as.numeric(Q14.0),
Q14.1=as.numeric(Q14.1),Q14.2=as.numeric(Q14.2), Q14.3=as.numeric(Q14.3),
Q14.4=as.numeric(Q14.4), Q14.5=as.numeric(Q14.5),
Q15.1=as.numeric(Q15.1), Q15.2=as.numeric(Q15.2), Q15.3=as.numeric(Q15.3),
Q15.4=as.numeric(Q15.4),Q15.5=as.numeric(Q15.5),
Ctoverall.1=as.numeric(Ctoverall.1),Ctoverall.2=as.numeric(Ctoverall.2),
Ctoverall.3=as.numeric(Ctoverall.3), Ctoverall.4=as.numeric(Ctoverall.4),
Ctoverall.5=as.numeric(Ctoverall.5) )
xtestBoost <- as.data.frame(dttestBoost)
xtestBoost <- transform(
  xtestBoost,
BI_numeric=as.numeric(BI_numeric),
Q1.1=as.numeric(Q1.1), Q1.2=as.numeric(Q1.2), Q1.3=as.numeric(Q1.3),
Q1.4=as.numeric(Q1.4),Q1.5=as.numeric(Q1.5), Q2.1=as.numeric(Q2.1),
Q2.2=as.numeric(Q2.2),Q2.3=as.numeric(Q2.3),Q2.4=as.numeric(Q2.4),
Q2.5=as.numeric(Q2.5),
Q3a.1=as.numeric(Q3a.1),Q3a.2=as.numeric(Q3a.2),Q3a.3=as.numeric(Q3a.3),
Q3a.4=as.numeric(Q3a.4), Q3a.5=as.numeric(Q3a.5),Q3b.1=as.numeric(Q3b.1),
Q3b.2=as.numeric(Q3b.2), Q3b.3=as.numeric(Q3b.3),
Q3b.4=as.numeric(Q3b.4),Q3b.5=as.numeric(Q3b.5),
Q4.1=as.numeric(Q4.1), Q4.2=as.numeric(Q4.2), Q4.3=as.numeric(Q4.3),
Q4.4=as.numeric(Q4.4),Q4.5=as.numeric(Q4.5), Q5.1=as.numeric(Q5.1),
Q5.2=as.numeric(Q5.2), Q5.3=as.numeric(Q5.3),
Q5.4=as.numeric(Q5.4), Q5.5=as.numeric(Q5.5), Q6.0=as.numeric(Q6.0),
Q6.1=as.numeric(Q6.1),Q6.2=as.numeric(Q6.2), Q6.3=as.numeric(Q6.3),
Q6.5=as.numeric(Q6.5), Q6.4=as.numeric(Q6.4),
Q7.1=as.numeric(Q7.1), Q7.2=as.numeric(Q7.2), Q7.3=as.numeric(Q7.3),
Q7.4=as.numeric(Q7.4), Q7.5=as.numeric(Q7.5), Q8.1=as.numeric(Q8.1),
Q8.2=as.numeric(Q8.2), Q8.3=as.numeric(Q8.3),   Q8.4=as.numeric(Q8.4),
Q8.5=as.numeric(Q8.5), Q9.1=as.numeric(Q9.1), Q9.2=as.numeric(Q9.2),
Q9.3=as.numeric(Q9.3), Q9.4=as.numeric(Q9.4), Q9.5=as.numeric(Q9.5),
Q10.1=as.numeric(Q10.1),Q10.2=as.numeric(Q10.2), Q10.3=as.numeric(Q10.3),
Q10.4=as.numeric(Q10.4), Q10.5=as.numeric(Q10.5),Q11.0=as.numeric(Q11.0),
Q11.1=as.numeric(Q11.1), Q11.2=as.numeric(Q11.2), Q11.3=as.numeric(Q11.3),
Q11.4=as.numeric(Q11.4), Q11.5=as.numeric(Q11.5), Q12.1=as.numeric(Q12.1),
Q12.2=as.numeric(Q12.2), Q12.3=as.numeric(Q12.3), Q13.1=as.numeric(Q13.1),
Q13.2=as.numeric(Q13.2), Q13.3=as.numeric(Q13.3),
Q13.4=as.numeric(Q13.4), Q13.5=as.numeric(Q13.5), Q14.0=as.numeric(Q14.0),
Q14.1=as.numeric(Q14.1),Q14.2=as.numeric(Q14.2), Q14.3=as.numeric(Q14.3),
Q14.4=as.numeric(Q14.4), Q14.5=as.numeric(Q14.5),
Q15.1=as.numeric(Q15.1), Q15.2=as.numeric(Q15.2), Q15.3=as.numeric(Q15.3),
Q15.4=as.numeric(Q15.4),Q15.5=as.numeric(Q15.5),
Ctoverall.1=as.numeric(Ctoverall.1),Ctoverall.2=as.numeric(Ctoverall.2),
Ctoverall.3=as.numeric(Ctoverall.3), Ctoverall.4=as.numeric(Ctoverall.4),
Ctoverall.5=as.numeric(Ctoverall.5) )
# assign new column names
#colnames(xtest) <- colnames_train_test
head(xtestBoost)

nopredBoost <- ncol(xtrainBoost) - 2
nopredBoost
str(xtrainBoost)

#my_modelB <- as.factor(BI_numeric) ~ Q1 + Q2 + Q3a + Q3b +
#       Q4 + Q5 + Q6 +
#       Q7 + Q8 + Q9 + Q10 + Q11 + Q12 + Q13 +
#       Q14 + Q15 + Ctoverall
```

```
#my_modelB

# http://jamesmarquezportfolio.com/get_up_and_running_with_xgboost_in_r.html
# https://rpubs.com/awanindra01/xgboost
# https://www.infopulse.com/blog/the-solution-to-binary-classification-task-
using-xgboost-machine-learning-package/

library(xgboost)
library(Matrix)
library(caret)
library("dplyr")

train        <- xtrainBoost
test         <- xtestBoost
train.label  <- xtrainBoost[,"BI_numeric"]
test.label   <- xtestBoost[,"BI_numeric"]

# Create sparse matrices and perform One-Hot Encoding to create dummy
# variables
dtrain  <- sparse.model.matrix(as.factor(BI_numeric) ~., data=train)
dtest   <- sparse.model.matrix(as.factor(BI_numeric) ~., data=test)

train_matrix = xgb.DMatrix(data = as.matrix(dtrain), label = train.label)
#test_matrix <- list(sparse.model.matrix(BI_numeric ~ .-1, data = train),
label)
test_matrix = xgb.DMatrix(data = as.matrix(dtest), label = test.label)

# View the number of rows and features of each set
dim(dtrain)
head(dtrain)
dim(dtest)
head(dtest)

# Set our hyperparameters
# objective = "binary:logistic"`: we will train a binary classification
# model ;
# `max_depth = 3`: the trees won't be deep, because our case is very simple ;
# `nthread = 2`: the number of cpu threads we are going to use;

param <- list(objective   = "binary:logistic",
              eval_metric = "error",
              max_depth   = 3,
              eta         = 0.1,
              gamma      = 1,
              colsample_bytree = 0.5,
              min_child_weight = 1)
# try different tuning parameters
# try 1: gamma = 0, eta = 0.05 ; # try 2: gamma = 0, eta = 0.01;
# try 3: gamma = 1, eta = 0.01 ; # original:  gamma=1 eta = 0.1
# gamma 0.01, eta = 0.01
set.seed(1)

bst_model = xgb.train(params = param, data = train_matrix,
                      nrounds = 500)
bst_model
str(bst_model)
```

```
#feature importance
imp = xgb.importance(colnames(train_matrix), model = bst_model)
xgb.plot.importance(main="Relative Importance Variables from One Hot Encoding
of XGBoost",imp)
imp

# Train the XGBoost classifer
bst_model2 = xgb.train(
  params=param, data=train_matrix, nrounds=500, nthreads=1,
  early_stopping_rounds=10,
watchlist=list(val1=train_matrix,val2=test_matrix), verbose=2 )
bst_model2

dt <- xgb.model.dt.tree(colnames(train_matrix), bst_model2)
dt

#prediction & confusion matrix
p = predict(bst_model, newdata = test_matrix)
p

p2 = predict(bst_model2, newdata = dtest)
p2

# See https://www.youtube.com/watch?v=woVTNwRrFHE

nc <- length(unique(train.label))
nc
head(p2)
#p2pred <- matrix(p2, nrow=nc, ncol = length(p2)/nc) %>%

# plot the training and test error for each iteration. The best iteration
# has the smallest gap between the training and test errors to avoid
# overfitting

ev <- data.frame(bst_model2$evaluation_log)
x11()
plot(main= "Plot of the Training and Test Errors for each XGBOOST iteration",
ev$iter, ev$val1_error, col = "blue", xlab="Iteration (ev$iter)",
ylab="Training/Test Errors")
lines(ev$iter, ev$val2_error, col = "red")
legend("bottomleft", legend=c("Training Error (ev$val1_error)", "Test Error
(ev$val2_error)"),
       col=c("blue", "red"), text.col=c("blue", "red"), lty=1:2, cex=0.8)

#head(p2pred)

test$BI_numeric
test$predicted = ifelse(p2 > 0.5,1,0)
test$predicted
xgb_pred2_class <- ifelse (as.numeric(p2) > 0.5,1,0)
xgb_pred2_class
#p2$predicted
table(test$predicted, test$BI_numeric)

confusionMatrix(factor(xgb_pred2_class), factor(test$BI_numeric),
                  mode = "everything", positive='1')
```

```
]\"
);

R term();
```

Appendix 3: To integrate R and base SAS, follow these steps described by Alexander [6]:

1. Download the PROC_R macro code from Wei [5].
2. Save the file in a location (e.g., "P:\My SAS Files\pbi Random Forest R Script 13Feb2020.sas").
3. Open the code and update the path of R executable file in the code below.

```
add the location path where R version 3.5.0 is installed ;
%macro quit(rpath=%str(C:\Progra~1\R\R-3.5.0\bin\R.exe));
```

   Here the rpath points to the location where the R executable is saved on the same machine where SAS is installed. Be sure to install the R library packages used for the analysis.
4. Open Base SAS and call the PROC_R3 macro, a modification of Xin Wei's PROC_R macro (Wei[5]) that executes R code in base SAS. The Proc_R3_fgname.sas program (with the %PROC_R3 macro) suppresses macro variables &fgname and &fgsw from being printed to the SAS log that avoids errors when running the program.
5. Run R inside SAS environment. See the SAS program below:

```
%include "P:\My SAS Files\Proc_R3_RF.sas "  ;
%Proc_R3(SAS2R = dttrain dttest, R2SAS=) ;
/*  dttrain and dttest  saved as SAS datasets*/
 cards4 ;
/*****************************************************************/
  Insert the R code between the R Submit("\[…]\"); command in
  Appendix 1
/*****************************************************************/

dev.off()
;;;;
%quit;
```

The macro variables are described as follows:

SAS2R - specifies the names of SAS datasets converted into R data frames or objects. The dataset can be single file names or multiple files names separated by spaces.
R2SAS - specifies the names of R data frames or objects that convert into SAS datasets. These objects can be single file names or multiple file names separated by spaces.