Model Selection to Predict User Knowledge Level

Jianghui Wen, Zhiyin Shi

***Abstract*—The application of logistic regression and tree models is widely used for classification and segmentation. In this study, to help institutions have a better understanding of students' knowledge status in a subject, we applied multinomial logistic regression, proportional odds logit regression, classification tree, and random forest on user knowledge modeling dataset to classify students to 4 different knowledge levels: very low, low, middle, and high based on their degree of study time for goal object materials, degree of repetition number of user for goal object materials,degree of study time of user for related objects with goal object,exam performance of user for related objects with goal object, and exam performance of user for goal objects. Eventually, institutions can provide different course options for different knowledge level students to register.**

I. Introduction

Online education is becoming increasingly popular in recent years, educational technology companies like Coursera and Big Data University provide massive online courses and change the way people learn. User modeling, which describes the process of building up and modifying a conceptual understanding of the user[1], has been widely applied by such institutions and organizations to understand their prospective students and thus be able to customize and develop the courses.

User modeling in online learning environment is composed of static and dynamic data[2]. Static data refers to features such as username, password and other physical characteristics, while dynamic data represents students knowledge on certain materials such as the degree of study time, repetition number of reading and exam performance of user for goal material. In this study, we aim to understand the effect of those dynamic patterns and how we can utilize such data to classify students into different user knowledge levels (UNS), and eventually be able to design customized courses based on this classification.

In this study, we explored through a dynamic user knowledge dataset and predicted the user knowledge level (UNS) with five dynamic features using regression and tree-based models. We fitted our dataset with four models, multinomial regression, proportional odds logit, decision tree and random forest. Within each model class, we performed cross-validation to find the optimal parameter set. Finally, we evaluated and compared the best model from each model class, and made suggestions on the best model for predicting UNS with the select features.

II. Data Description

The user knowledge modeling dataset is retrieved from UCI machine learning repository. The dataset describes students knowledge status towards to the subject of Electrical DC Machines[4]. The raw dataset is clean and ready to use. The dataset contains variables on 403 students, with already normalized and splited training and test dataset by a ratio of 258:145. The dataset includes five independent features: STG, SCG, STR, LPR, and PEG; and one four-levels response variable: UNS. Detailed dataset and feature information is listed in the following:

Data Set Characteristics:	Multivariate	**Number of Instances:**	403	**Area:**	Computer
Attribute Characteristics:	Integer	**Number of Attributes:**	5	**Date Donated**	2013-06-26
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	56686

Table 2.1

Features' name	Information
STG	The degree of study time for goal object materials.
SCG	The degree of repetition number of user for goal object materials.
STR	The degree of study time of user for related objects with goal object.
LPR	The exam performance of user for related objects with goal object.
PEG	The exam performance of user for goal objects.

Table 2.2 Features (Independent variables)

Classes	The total number for each class
Very low	50
Low	129
Middle	120
High	102

Table 2.3 Response (Dependent variables)

Before applying any statistical learning method, we would like to explore into the features and the correlations first. Since here are only 5 features, we can still manage to generate a pairwise plot to show the relationship between every pair of features in our dataset. The plot is shown in Figure 2.1. As we can see, each pair of features is hardly correlated and we can assume they are independent to each other.

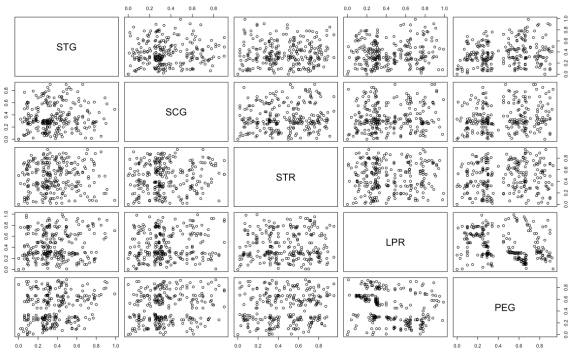

Figure 2.1 Feature Pairwise Plot

III. Methodology

A. Overview

In this study, we aim to search for the best model to predict ordinal response User Knowledge Level with five features related to user dynamic data using regression and tree-based

models. Our dataset is divided into training and testing by a ratio of 258 : 145.

The method is specific for this specific dataset only, it is not guaranteed that our final model will yield accurate prediction for other UNS related dataset, since our sample size is small and features available for different problems may vary. We fitted our train dataset with four models, multinomial regression, proportional odds logistic regression, decision tree and random forest. Within each model class, we performed cross-validation to find the optimal parameter set. Finally, we predicted the test dataset, and evaluated the fitness of each model based on misclassification test error rates and log loss.

B. Multinomial Logistic Regression

We fitted a full multinomial logistic regression model on train dataset with all independent variables and ignored any interactions, since our features showed adequate independence. Next, we applied step function to the full multinomial model in order to find the optimal combination of features which lead to the lowest AIC. With the optimal model we selected from stepwise-method, we made prediction on test dataset and evaluated the model goodness of fit based on misclassification error rate and log loss.

C. Proportional Odds Logit

Proportional odds logit model is designed for the data with ordered categorical outcomes. The dependent variable UNS in our dataset has natural orders. It is a considerable model to apply on the user knowledge dataset. First, we reordered the UNS classes from low to high: Very Low, Low, Middle, High. Then, we fitted the full proportional odds logit regression model with all features. Next, after selecting significant variables by comparing the AIC from step() function, we build the final proportional logit regression model with significant variables SCG, STR, LPR and PEG. Finally, we predicted the test dataset with the optimal model from last step and computed its error rates.

D. Classification Decision Tree

Tree-based models involves segmenting the feature space into a number of non-overlapping subspaces and assigning each subspace with a prediction value, so every observation falls into the same subspace will have the same response. In this dataset, our response is categorical and one of the following: very low, low, medium and high. We predict the response for each observation based on the most commonly occurring class it belongs to. Next, we grow a large tree using recursive binary partitioning until a stopping criterion is met. Misclassification error rate is used as the splitting criterion, which is simply the fraction of the training observations in that region that do not belong to the common class[3].

A large tree may yield a good prediction but also increase the risk of overfitting, thereby tree pruning is applied to trade-off between classification error rate and tree complexity. In our case, we pruned our tree using 6-fold cross validation (since 6 is a multiple of our training sample size) and selected the optimal parameter set for our final classification tree model, which will be compared with models from other model classes later.

E. Random Forest

Decision tree described in previous section suffers from high variance, especially when the dataset is large. To mitigate the variance volatility, random forest is introduced which involves growing a number of trees with different number of predictors and average them to produce a single low variance tree. As a first step, we build a number of trees on bootstrapped training samples. Next, for each tree we randomly choose m = sqrt(p) predictors to train the model, where p is the total number of predictors, according to the common practice. By selecting a subset of predictors to train our model, the trees in our bag will no longer be highly correlated, so we can think random forest as a tree decorrelation process. Finally, we combine all the trees to obtain a single low-variance tree.

In this study, we first applied 6-fold cross validation to select parameter m, the number of predictors to be chosen for each tree. Then we trained a random forest model which grows 500 trees in total.

F. Error Measures

In this project, we use two error measures, misclassification rate and log loss. Misclassification error rate is the fraction of observations in that region that do not belong to the common class.
Log Loss is also a common measure of error in classification problems, it is defined as following:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\ln(p_{ij}).$$

where N is the number of objects to classify, M is the number of classes, yij is 1 if object i belongs to class j, and 0 otherwise, pij is predicted probability that object i belongs to class j.

IV. Result

A. Multinomial Logistic Regression

First, we fit the full multinomial logistic regression model on train dataset. This full model resulted in AIC = 82.03 and residual deviance = 46.83. The summary from multinom() function in R is showsn in Table A.1.

```
multinom(formula = UNS ~ STG + SCG + STR + LPR + PEG, data = uns.training,
    Hess = TRUE, method = "class")

Coefficients:
         (Intercept)        STG         SCG        STR       LPR        PEG
Low         78.57295 -0.4992804  -4.1057329  -8.506013 -39.30303 -127.35111
Middle      39.10377  2.3863912   0.6933342  -5.273197 -20.60977  -44.58053
Very Low   129.25850  5.0057595 -21.9472339 -20.515917 -82.83290 -276.80730

Std. Errors:
         (Intercept)       STG       SCG      STR       LPR      PEG
Low         18.64044  5.237707  4.666492 4.694097  9.715870 38.88792
Middle      11.35121  4.187757  3.742525 3.831596  7.127002 12.63146
Very Low    35.11194 11.221725 11.924218 9.677018 26.703602 95.10127

Residual Deviance: 46.03233
AIC: 82.03233
```

Table A.1. Summary of Full Multinomial Logistic Regression Model

We cannot assess the goodness of fit of the full model with comparing AIC and residual deviance values of other models with different predictor combination. Thus, we applied stepwise-forward method and called the function step() in R, in order to select the optimal multinomial model with lowest AIC. From the stepwise method result, the multinomial model with 4 predictors SCG, STR, LPR, and PEG resulted in the lowest AIC, which equals to 77.599, and was chosen by us as the optimal multinomial logistic regression model. The summary returned by multinom() of the optimal model is shown in Table A.2.

```
multinom(formula = UNS ~ SCG + STR + LPR + PEG, data = uns.training,
    Hess = TRUE, method = "class")

Coefficients:
         (Intercept)         SCG        STR       LPR        PEG
Low         75.13094  -4.4306708  -7.935034 -37.11023 -119.05918
Middle      40.00667   0.7374325  -5.082486 -20.14013  -45.09244
Very Low   130.00301 -24.1390313 -20.647141 -83.02074 -273.14232

Std. Errors:
         (Intercept)       SCG      STR       LPR      PEG
Low         16.36956  4.631763 4.504982  8.569799 33.11847
Middle      11.39990  3.717587 3.710436  6.751478 12.56627
Very Low    33.30302 11.763231 9.434851 25.363650 86.91911

Residual Deviance: 47.59966
AIC: 77.59966
```

Table A.2. Summary of Optimal Multinomial Logistic Regression Model

Next, we predicted the UNS classes of test dataset using the optimal model from previous step. This resulted in misclassification of 4 out 145 observations, which is equivalent to 3% misclassification rate. The details of confusion matrix is shown in the Table A.3. The log loss for the final is 0.069, which suggests a very good classification model.

Table A.3 Confusion Matrix of Optimal Multinomial Logistic Regression Model Test

	Very Low	Low	Middle	High
Very Low	26	0	0	0
Low	0	45	1	0
Middle	0	1	31	2
High	0	0	0	39

B. Proportional Odds Logit

The summary of full proportional odds logit regression model shows that AIC is 76.98. The details of the coefficients are shown in the Table B.1.

```
polr(formula = UNS.ordered ~ STG + SCG + STR + LPR + PEG, data = uns.training,
    Hess = TRUE, method = "logistic")

Coefficients:
      Value Std. Error t value
STG  0.7116      2.032  0.3501
SCG  5.2306      1.795  2.9144
STR  3.6791      1.605  2.2925
LPR 17.8878      3.094  5.7814
PEG 55.5886      9.039  6.1498

Intercepts:
             Value   Std. Error t value
Very Low|Low 19.5733  3.4698     5.6410
Low|Middle   31.5743  5.1801     6.0953
Middle|High  48.4146  8.0885     5.9856

Residual Deviance: 60.97596
AIC: 76.97596
```

Table B.1. Summary of Full Proportioanl Odds Logit Model

Next, we applied forward stepwise method to choose a good model with significant variables. The optimal proportional odds logit model includes 4 variables: SCG, STR, LPR, and PEG. The AIC of the final model is about 75.10. Details of the summary are shown in Table B.2.

```
polr(formula = UNS.ordered ~ SCG + STR + LPR + PEG, data = uns.training,
    Hess = TRUE, method = "logistic")

Coefficients:
     Value Std. Error t value
SCG  5.206      1.789   2.911
STR  3.707      1.606   2.308
LPR 17.758      3.038   5.845
PEG 55.308      8.932   6.192

Intercepts:
             Value   Std. Error t value
Very Low|Low 19.2614  3.3040     5.8298
Low|Middle   31.1964  5.0036     6.2348
Middle|High  47.9203  7.8637     6.0939

Residual Deviance: 61.09955
AIC: 75.09955
```

Table B.2. Summary of Optimal Proportional Odds Logit Model

We then predicted the UNS classes on test dataset with this final optimal model. The confusion matrix shows that 3 out of 145 observations are misclassified and the misclassification rate is 0.02. Confusion matrix details are shown in Table B.3. The final proportional logit regression model predicted the test dataset very well, with log loss of 0.053.

	Very Low	Low	Middle	High
Very Low	26	0	0	0
Low	0	45	1	0
Middle	0	1	32	1
High	0	0	0	39

Table B.3. Confusion Matrix of Optimal Proportional Odds Logit Model

C. Classification Decision Tree

First, we fitted a deep tree model to our training dataset and this resulted in 10 terminal nodes, the structure of the tree is shown in Figure C.1.

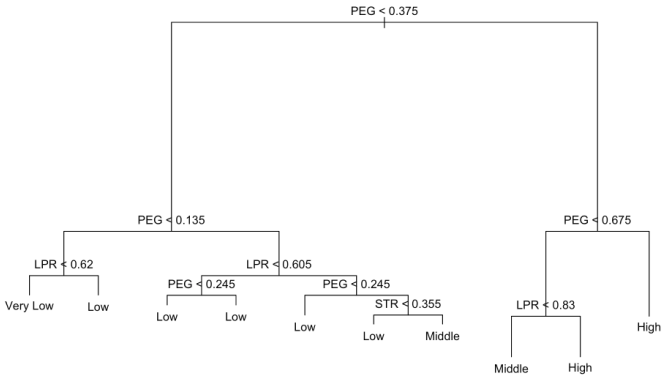

Figure C.1. Single deep tree.

The variables that actually used for this tree model are PEG, LPR and STR. The result is pretty good, with classification error rate of 3.9%, which means only 10 out of 258 data points in train dataset are misclassified with this model.

Next, we prune the tree in the hope of improved results. Now, lets first choose the optimal complexity parameter through 6-fold cross validation, and see if the pruned tree with the best alpha value can yield a better test error than the original tree above. The CV error, measured in deviance, against number of terminal nodes and various complexity parameter values are shown in Figure C.2 and Figure C.3 respectively. We can see from the plots that number of nodes = 10 and α = 0 yields the best result. Cross-validation suggests us to stick with the original unpruned tree, which has lowest misclassification error.

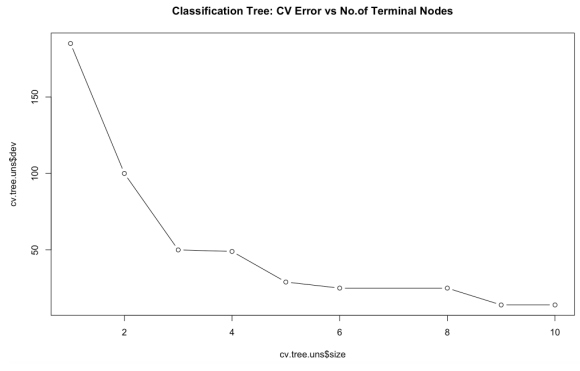

Figure C.2. CV Error vs. No. of Terminal Nodes.

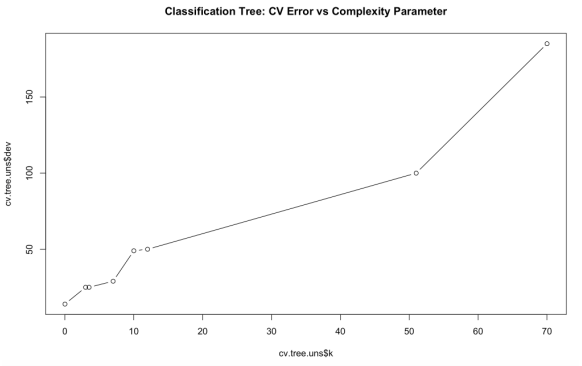

Figure C.3. CV Error vs. Complexity Parameter.

Applying this tree model on test dataset results in log loss = 1.764 and misclassification of 11 out 145 observations, which is equivalent to 7.6% misclassification error rate. This value doubled the error rate in training dataset, but is still a good result. The confusion matrix is shown in Table C.1

	High	Low	Middle	Very Low
High	39	0	1	0
Low	0	43	4	3
Middle	0	3	29	0
Very Low	0	0	0	23

Table C.1. Confusion Matrix for Classification Tree

D. Random Forest

The classification tree with 10 nodes has resulted in a pretty good prediction, but will random forest perform even better? We will fit the training dataset with Random Forest model and check its performance. First, we performed a 6-fold cross-validation to select the optimal number of predictors to train each tree in Random Forest model. The result is shown in Figure D.1. From the CV, we will choose m = 2 as our number of predictors selected to train each tree in RF.

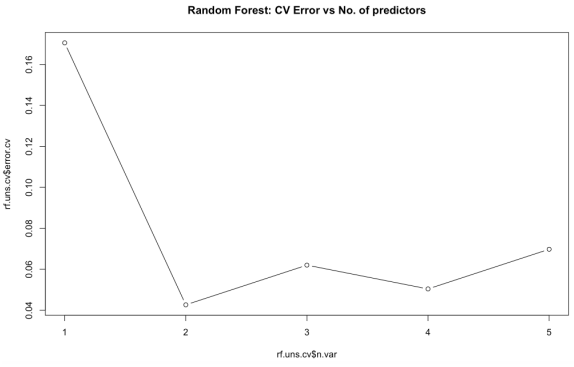

Figure D.1. CV Error vs. Number of Predictors

Now, we fit our train dataset with RF model with number of trees = 500 and m = 2. The confusion matrix on the train dataset is shown in Table D.1. 18 out of 258 train data points are misclassified, which is about 7% misclassification error rate.

	High	Low	Middle	very_low
High	62	0	1	0
Low	0	79	6	6
Middle	1	3	81	0
very_low	0	1	0	18

Table D.1. Confusion Matrix for RF Train

Next, we predict our test dataset using the RF model above. The result is represented in a confusion matrix, as shown in Table D.2.

	High	Low	Middle	Very Low
High	39	0	0	0
Low	0	44	3	3
Middle	0	2	31	0
very_low	0	0	0	23

Table D.2. Confusion Matrix for RF Test

The random forest model with m = 2 results in 5.5% misclassification error rate and 0.286 log loss.

E. Summary of All Models and Recommendation

In the previous steps, we selected the optimal model in each model class using forward-stepwise and cross-validation methods, applied the optimal model to predict UNS classes in test dataset, and computed the misclassification error rates and log loss values. In this section, we put all results together, compare these models and select the best model to predict User Knowledge Level.

The visualization of error rates of all models is shown in Figure E.1. The detailed numeric values are shown in Table E.1. As we can see, the misclassification error rate and sum of log loss are consistent in measuring model goodness of fit. For our dataset specifically, proportional odds yields the lowest error rate, and single decision tree yields in the highest.

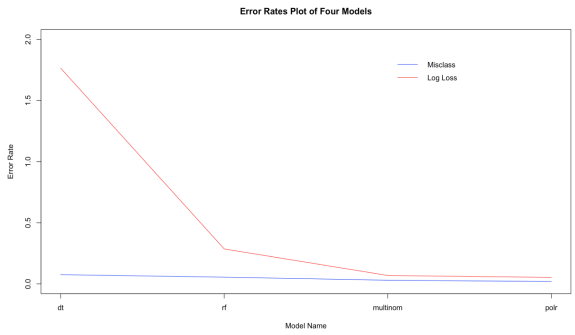

Figure E.1. Error Rates of Different Models

	Decision Tree	Random Forest	Multinom	Proportional Odds
Misclassification	0.076	0.055	0.03	0.02
Log Loss	1.764	0.286	0.069	0.053

Table E.1. Error Rates of Different Models

This result is very reasonable and matches our hypothesis at the beginning. Tree-based models are extremely useful when the feature-response relationship is highly complex and the feature space is huge, but in the expense of losing accuracy. Random forest, which aggregated many trees, performed better than the single decision tree, but still not as competitive as the other two models. On the other hand, logistic regression models are more accurate but are computational expensive when the dataset grows large.

In our case, the sample size was small thereby every model class was equally competitive in terms of computation speed. When comparing in terms of accuracy, the logistic regression models outweighed tree-based models significantly. When narrowing the scope to the former class, proportional odds performed better than multinomial model, even though the classification boundaries for proportional odds model are always parallel and less flexible. This is because our response in this case is well ordered from 'Very Low' to 'High', and proportional odds logit model is a great choice for such ordinal class responses.

In conclusion, we suggest the optimal model to classify User Knowledge Level to be proportional odds logit model, which resulted in misclassification test error rate of 2% and sum of log loss of 0.053, the best among all four model classes.

V. Conclusion

In this study, we performed cross-validation to find the optimal parameter set in tree models and selected best feature combination model using forward stepwise method in regression models. Essentially, we used the test classification rate and log loss to compare and evaluate the performance of four best models. From our study, we see that tree-based models did not perform as good as logistic regression models when the response was well ordered and feature space was relatively small. At the end, we suggest that the best model to predict this dataset is proportional odds logit regression model, with the lowest misclassification rate of 0.02 and lowest log loss score of 0.0527.

VI. Limitations and Future Suggestions

1. Our training and testing dataset contains 258 and 145 observations respectively. The dataset is very small and may not represent whole population.

2. Despite our current selection of logistic regression models are already very accurate, We may also want to apply cross-validation technique to these models in the future with different combinations of polynomials and see if the model can be further improved.

3. For the random forest model, we took the advantage of small dataset, so it was not computationally expensive to grow large number of trees such as 500. As the dataset becomes huge, it is no longer easy to grow many trees, other models and computation tools should be considered.

References

[1] User Modeling: https://en.wikipedia.org/wiki/User_modeling
[2] H. T. Kahraman, Sagiroglu, S., Colak, I., *Developing intuitive knowledge classifier and modeling of users' domain dependent data in web*, Knowledge Based Systems, vol. 37, pp. 283-295, 2013.

[3] James, Witten, Hastie, Tibshirani*ISLR*.
[4] Data source: https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling

VII. Appendix

A. R Code

0. Import Data

```
#Import Data
dataPath <- "/Users/JaneShi/Desktop/MSCA31008/Project"

uns.train <- read.csv(paste(dataPath, "uns_training.csv", sep = "/"),
                      header = TRUE, sep = ",")
uns.test <- read.csv(paste(dataPath, "uns_testing.csv", sep = "/"),
                      header = TRUE, sep = ",")
#Rename Column Values
library(plyr)
uns.train$UNS <- revalue(uns.train$UNS, c('High' = 'High', 'Low' = 'Low',
                         'Middle' = 'Middle', 'very_low' = 'Very Low'))
```

1 Functions for Response Transformation and LogLoss Computation

```
log.loss <-  function(actual, pred){
    eps <- 1e-15
    if (!is.matrix(pred)) pred <- t(as.matrix(pred))
    if (!is.matrix(actual)) actual <- t(as.matrix(actual))
    nr <- nrow(pred)
    pred <- apply(pred, c(1,2), function(x) max(min(x, 1-10^(-15)), 10^(-15)))
    score <- -sum(actual*log(sweep(pred, 1, rowSums(pred), FUN="/")))/nr
    return(score)
}

response.matrix <- function(Y)
{
    y_res <- matrix(NA,nrow = 0,ncol = 4)
    for (i in 1:length(Y))
    {
        if (Y[i] == "High")
            y_res <- rbind(y_res,c(1,0,0,0))
        if (Y[i] == "Low")
            y_res <- rbind(y_res,c(0,1,0,0))
        if (Y[i] == "Middle")
            y_res <- rbind(y_res,c(0,0,1,0))
        if (Y[i] == "Very Low")
            y_res <- rbind(y_res,c(0,0,0,1))
    }
    return (y_res)
}
```

2. Multinomial Logistic Regression

```
uns.multinomial <- multinom(UNS ~ STG+SCG+STR+LPR+PEG, data=uns.training, Hess= TRUE, method="class")
```

```
## # weights:  28 (18 variable)
## initial  value 357.663945
## iter  10 value 84.418495
## iter  20 value 31.346775
## iter  30 value 28.785519
## iter  40 value 27.932325
## iter  50 value 26.709746
## iter  60 value 24.812505
## iter  70 value 23.144098
## iter  80 value 23.106160
## iter  90 value 23.069791
## iter 100 value 23.016163
## final  value 23.016163
## stopped after 100 iterations
```

```
summary(uns.multinomial)
```

```
## Call:
## multinom(formula = UNS ~ STG + SCG + STR + LPR + PEG, data = uns.training,
##     Hess = TRUE, method = "class")
##
## Coefficients:
##          (Intercept)        STG         SCG        STR       LPR
## Low         78.57295 -0.4992804  -4.1057329  -8.506013 -39.30303
## Middle      39.10377  2.3863912   0.6933342  -5.273197 -20.60977
## Very Low   129.25850  5.0057595 -21.9472339 -20.515917 -82.83290
##                PEG
## Low      -127.35111
## Middle    -44.58053
## Very Low -276.80730
##
## Std. Errors:
##          (Intercept)       STG       SCG       STR       LPR      PEG
## Low         18.64044  5.237707  4.666492 4.694097  9.715870 38.88792
## Middle      11.35121  4.187757  3.742525 3.831596  7.127002 12.63146
## Very Low    35.11194 11.221725 11.924218 9.677018 26.703602 95.10127
##
## Residual Deviance: 46.03233
## AIC: 82.03233
```

Using step() to select variables to build a good model or calculate p values to select significant varaibles using the following ctable <- coef(summary(uns.polr)) p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2 ctable <- cbind(ctable, "p value" = p)

```
#step(uns.multinomial)
```

The result shows a good fitted model including predictors SCG,STR,LPR, PEG, with AIC=77.5996

```
uns.multinomial.model <- multinom(UNS~SCG+STR+LPR+PEG, data=uns.training, Hess= TRUE, method="class")
```

```
## # weights:  24 (15 variable)
## initial  value 357.663945
## iter  10 value 90.958628
## iter  20 value 31.387774
## iter  30 value 29.221481
## iter  40 value 28.281752
## iter  50 value 26.824243
## iter  60 value 24.666243
## iter  70 value 23.895758
## iter  80 value 23.854259
## iter  90 value 23.825782
## iter 100 value 23.799831
## final  value 23.799831
## stopped after 100 iterations
```

3.Proportional Odds logit regression

```
library(MASS)
UNS.ordered <- ordered(uns.training$UNS, levels=c("Very Low","Low","Middle","High"))
uns.polr <- polr(UNS.ordered~STG + SCG + STR + LPR + PEG, data=uns.training, Hess= TRUE, method="logistic")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary.uns.polr <- summary(uns.polr)
```

Using step() to select variables

```
#step.polr <- step(uns.polr)
```

The good fitted model including predictors SCG,STR,LPR,PEG, with AIC= 75.09955

```
uns.polr.model <-  polr(UNS.ordered ~ SCG + STR+ LPR + PEG, data = uns.training, Hess = TRUE, method="logistic")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summry.uns.polr.model <- summary(uns.polr.model)
```

Holdout for proportional odds logistic regression

```
uns.polr.test <- predict(uns.polr.model, newdata= uns.testing, type="class")
grid.table(table(uns.testing$UNS, uns.polr.test))
```

4. Classification Tree

```
library(tree)
#Train Classification Tree Model
tree.uns <- tree(UNS ~ ., data = uns.train)
summary(tree.uns)
```

```
##
## Classification tree:
## tree(formula = UNS ~ ., data = uns.train)
## Variables actually used in tree construction:
## [1] "PEG" "LPR" "STR"
## Number of terminal nodes:  10
## Residual mean deviance:  0.2096 = 51.99 / 248
## Misclassification error rate: 0.03876 = 10 / 258
```

```
cv.tree.uns <- cv.tree(tree.uns, FUN = prune.misclass, K = 6)
cv.tree.uns
```

```
## $size
## [1] 10  9  8  6  5  4  3  2  1
##
## $dev
## [1]  17  17  25  25  34  47  49 100 183
##
## $k
## [1] -Inf  0.0  3.0  3.5  7.0 10.0 12.0 51.0 70.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"          "tree.sequence"
```

```
plotCVClassTree1 <- plot(cv.tree.uns$size, cv.tree.uns$dev, type = "b", main = "Classification Tree: CV Error vs No.of Terminal Nodes")
```

```
plotCVClassTree2 <- plot(cv.tree.uns$k, cv.tree.uns$dev, type = "b", main = "Classification Tree: CV Error vs Complexity Parameter")
```

```
#Predict the test dataset and Compute test errors
tree.uns.pred <- predict(tree.uns, uns.test[, -6], type = "class")

llClassTree <- log.loss(response.matrix(uns.test$UNS), predict(tree.uns, uns.test[, -6]))

library(grid)
library(gridExtra)
cmClassTree <- grid.table(table(tree.uns.pred, uns.test[, 6]))
```

5. Random forest

```
suppressWarnings(library(randomForest))
```

```
#CV to select the best m (Number of Predictors)
rf.uns.cv <- rfcv(uns.train[, -6], uns.train[, 6], cv.fold = 6, step = 0.9)
plotCVRF <- plot(rf.uns.cv$n.var, rf.uns.cv$error.cv, type = "b", main = "Rand
om Forest: CV Error vs No. of predictors")

#Fit train dataset with RF
set.seed(1222711)
rf.uns <- randomForest(UNS ~ ., data = uns.train, mtry = 2, importance = TRUE)

cmRFTrain <- grid.table(table(rf.uns$predicted, uns.train[, 6]))

#Predict test dataset and Compute test errors
rf.uns.pred <- predict(rf.uns, newdata = uns.test[, -6])

llRF <- log.loss(response.matrix(uns.test$UNS), predict(rf.uns, newdata = uns.
test[, -6], type = 'prob'))

cmRFTest <- grid.table(table(rf.uns.pred, uns.test[, 6]))
```

6 Comparison

```
library(grid)
library(gridExtra)

#Combine all error rates
misclass <- c(dt=0.076, rf=0.055, multinom=0.03, polr=0.02)
logloss <- c(dt=1.764, rf=0.286, multinom=0.069, polr=0.053)
labels <- c('dt', 'rf', 'multinom', 'polr')

#Aggregate as a table
tb <- rbind(misclass, logloss)
colnames(tb) <- c('Decision Tree', 'Random Forest',
                  'Multinom', 'Proportional Odds')
rownames(tb) <- c('Misclassification', 'Log Loss')

grid.table(tb)
```

```
#Plot
plot(1:4, misclass, type = 'l', col = 'blue', ylim = c(0, 2),
     xaxt = "n", main = 'Error Rates Plot of Four Models',
     xlab = 'Model Name', ylab = 'Error Rate')
lines(1:4, logloss, col = 'red')
legend(3, 1.9, c('Misclass', 'Log Loss'), lty = c(1, 1),
       lwd=c(1,1),col=c('blue','red'), box.lty = 0)
axis(1, at=1:4, labels = labels)
```