Wenyi (Alex) Yan – Logistic Regression

Import data & Check Class Bias

```
###### Import Data ######
HOS=read.csv(file="~/desktop/employees_v2.csv")

###### Check Class Bias ######
table(HOS$IS_TURNOVER)
```

```
   0    1
4954 2472
```

Clearly, there is a class bias, a condition observed when the proportion of events is much smaller than proportion of non-events. So we must sample the observations in approximately equal proportions to get better models.

Create Training and Test sample

```
###### Create Training and Test Sample ######
# Create Training Data
HOS1=HOS[which(HOS$IS_TURNOVER==1),]  #all 1"s
HOS0=HOS[which(HOS$IS_TURNOVER==0),]  #all 0's
set.seed(100)  #for repeatability of samples
HOS1_TrainingRows=sample(1:nrow(HOS1),0.7*nrow(HOS1)) #1's for training
HOS0_TrainingRows=sample(1:nrow(HOS0),0.7*nrow(HOS1))  #0'S for training. pick as many 0's as 1's
Training1=HOS1[HOS1_TrainingRows,]
Training0=HOS0[HOS0_TrainingRows,]
TrainingData=rbind(Training1, Training0)  #row bind the 1's and 0's
# Create Test Data
Test1=HOS1[-HOS1_TrainingRows, ]
Test0=HOS0[-HOS0_TrainingRows, ]
TestData=rbind(Test1, Test0)  # row bind the 1's and 0's
```

One way to address the problem of class bias is to draw the 0’ s and 1’ s for the TrainingData (development sample) in equal proportions. In doing so, I put rest of the HOS data not included for training into TestData (validation sample). As a result, the size of development sample would be smaller that validation, which is okay.

Compute information value to find out important variables

```r
###### Compute information value to find out important variables ######
install.packages('smbinning')
install.packages('tcltk')
library(smbinning)
library(tcltk)
# segregate continuous and factor variables
factor_vars=c("PEER_GROUP_CAT_ABBR","PARTNERSHIP_TYPE","SKILL_CATEGORY_ABBR")
continuous_vars=c("FTE", "SENIORITY", "STAFF_COUNT", "UTO_COUNT","PTO_COUNT","TARDY_COUNT","TOTAL_HOURS"
iv_df=data.frame(VARS=c(factor_vars, continuous_vars), IV=numeric(22))  # init for IV results
# compute IV for categoricals
for(factor_var in factor_vars){
  smb=smbinning.factor(TrainingData, y="IS_TURNOVER", x=factor_var)  # WOE table
  if(class(smb) != "character"){ # heck if some error occured
    iv_df[iv_df$VARS == factor_var, "IV"]=smb$iv
  }
}
# compute IV for continuous vars
for(continuous_var in continuous_vars){
  smb=smbinning(TrainingData, y="IS_TURNOVER", x=continuous_var)  # WOE table
  if(class(smb) != "character"){  # any error while calculating scores.
    iv_df[iv_df$VARS == continuous_var, "IV"]=smb$iv
  }
}
iv_df <- iv_df[order(-iv_df$IV), ]  # sort
iv_df
```

```
                      VARS     IV
10             TOTAL_HOURS 4.0713
11                  OT_HRS 1.1932
12             LEAKAGE_HRS 1.0749
22           WEEKEND_HOURS 0.6809
21           WEEKEND_COUNT 0.6727
18 CRITICAL_STAFFING_COUNT 0.3661
5                SENIORITY 0.3094
3      SKILL_CATEGORY_ABBR 0.2533
8                PTO_COUNT 0.2448
1      PEER_GROUP_CAT_ABBR 0.0609
7                UTO_COUNT 0.0534
4                      FTE 0.0381
2         PARTNERSHIP_TYPE 0.0358
15             TRAIN_COUNT 0.0219
14            CANCELL_COUNT 0.0195
6              STAFF_COUNT 0.0000
9              TARDY_COUNT 0.0000
13               EXTRA_HRS 0.0000
16       TIMEOFF_REQ_COUNT 0.0000
17   EXCUSED_TIMEOFF_COUNT 0.0000
19     HIGH_STAFFING_COUNT 0.0000
20   OPEN_SHIFT_PICK_COUNT 0.0000
```

The smbinning::smbinning function converts a continuous variable into a categorical variable using recursive partitioning. I first converted them to categorical variables and then, captured the information values for all variables in iv_df.

Here we take variables whose IV are bigger than 0.3 (strong predictors)

Build Logit Models and Predict

```
###### Build Logit Models and Predict ######
LogitMod=glm(IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + LEAKAGE_HRS + WEEKEND_HOURS + WEEKEND_COUNT + CRITICAL
predicted=plogis(predict(LogitMod, TestData))  # predicted scores
###### Decide on optimal prediction probability cutoff for the model ######
install.packages('InformationValue')
library(InformationValue)
optCutOff=optimalCutoff(TestData$IS_TURNOVER, predicted)[1]
#=> 0.87
```

The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve the prediction of 1' s, 0' s, both 1' s and 0' s and reduce the misclassification error.

Model Diagnostics

```
###### Model Diagnostics ######
# GOF test
summary(LogitMod)

Call:
glm(formula = IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + LEAKAGE_HRS +
    WEEKEND_HOURS + WEEKEND_COUNT + CRITICAL_STAFFING_COUNT +
    SENIORITY, family = binomial(link = "logit"), data = TrainingData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3925  -0.3352   0.0088   0.4852   3.1146

Coefficients:
                          Estimate Std. Error z value             Pr(>|z|)
(Intercept)              1.9366869  0.0959282  20.189 < 0.0000000000000002 ***
TOTAL_HOURS             -0.0036161  0.0001319 -27.415 < 0.0000000000000002 ***
OT_HRS                   0.0058753  0.0007553   7.779  0.00000000000000731 ***
LEAKAGE_HRS              0.0001520  0.0007781   0.195               0.8451
WEEKEND_HOURS           -0.0052616  0.0018019  -2.920               0.0035 **
WEEKEND_COUNT            0.0892370  0.0215228   4.146  0.00003381038695697 ***
CRITICAL_STAFFING_COUNT  0.0156701  0.0071965   2.177               0.0294 *
SENIORITY                0.0129745  0.0082054   1.581               0.1138
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4796.6  on 3459  degrees of freedom
Residual deviance: 2188.5  on 3452  degrees of freedom
AIC: 2204.5

Number of Fisher Scoring iterations: 6
```

```
# remove insignificant x variables
LogitMod2=glm(IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + WEEKEND_HOURS + WEEKEND_COUNT + CRITICAL_STAFFING_COUNT
summary(LogitMod2)
```

```
Call:
glm(formula = IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + WEEKEND_HOURS +
    WEEKEND_COUNT + CRITICAL_STAFFING_COUNT, family = binomial(link = "logit"),
    data = TrainingData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4215  -0.3367   0.0087   0.4956   3.0839

Coefficients:
                          Estimate Std. Error z value            Pr(>|z|)
(Intercept)              1.9817932  0.0918470  21.577 < 0.0000000000000002 ***
TOTAL_HOURS             -0.0035994  0.0001293 -27.841 < 0.0000000000000002 ***
OT_HRS                   0.0057468  0.0007541   7.620   0.0000000000000253 ***
WEEKEND_HOURS           -0.0053671  0.0017908  -2.997              0.00273 **
WEEKEND_COUNT            0.0909348  0.0213515   4.259   0.0000205402260032 ***
CRITICAL_STAFFING_COUNT  0.0161275  0.0072038   2.239              0.02517 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4796.6  on 3459  degrees of freedom
Residual deviance: 2191.0  on 3454  degrees of freedom
AIC: 2203

Number of Fisher Scoring iterations: 6
```

```
# VIF
library(car)
vif(LogitMod2)
```

```
            TOTAL_HOURS                  OT_HRS           WEEKEND_HOURS           WEEKEND_COUNT
               2.811806                2.180309               79.580558               82.446737
CRITICAL_STAFFING_COUNT
               1.763324
```

```
# Remove WEEKEND_HOURS
LogitMod3=glm(IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + WEEKEND_COUNT + CRITICAL_STAFFING_COUNT, data=TrainingD
summary(LogitMod3)
vif(LogitMod3)
```

```
Call:
glm(formula = IS_TURNOVER ~ TOTAL_HOURS + OT_HRS + WEEKEND_COUNT +
    CRITICAL_STAFFING_COUNT, family = binomial(link = "logit"),
    data = TrainingData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1004  -0.3442   0.0154   0.4906   3.2838

Coefficients:
                          Estimate Std. Error z value             Pr(>|z|)
(Intercept)              2.0003245  0.0916553  21.824 < 0.0000000000000002 ***
TOTAL_HOURS             -0.0035834  0.0001285 -27.880 < 0.0000000000000002 ***
OT_HRS                   0.0058091  0.0007587   7.657   0.0000000000000191 ***
WEEKEND_COUNT            0.0280947  0.0033416   8.408 < 0.0000000000000002 ***
CRITICAL_STAFFING_COUNT  0.0150267  0.0072198   2.081               0.0374 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4796.6  on 3459  degrees of freedom
Residual deviance: 2200.0  on 3455  degrees of freedom
AIC: 2210

Number of Fisher Scoring iterations: 6

 TOTAL_HOURS                  OT_HRS           WEEKEND_COUNT CRITICAL_STAFFING_COUNT
    2.798183                2.205408                2.053822                1.775645
```

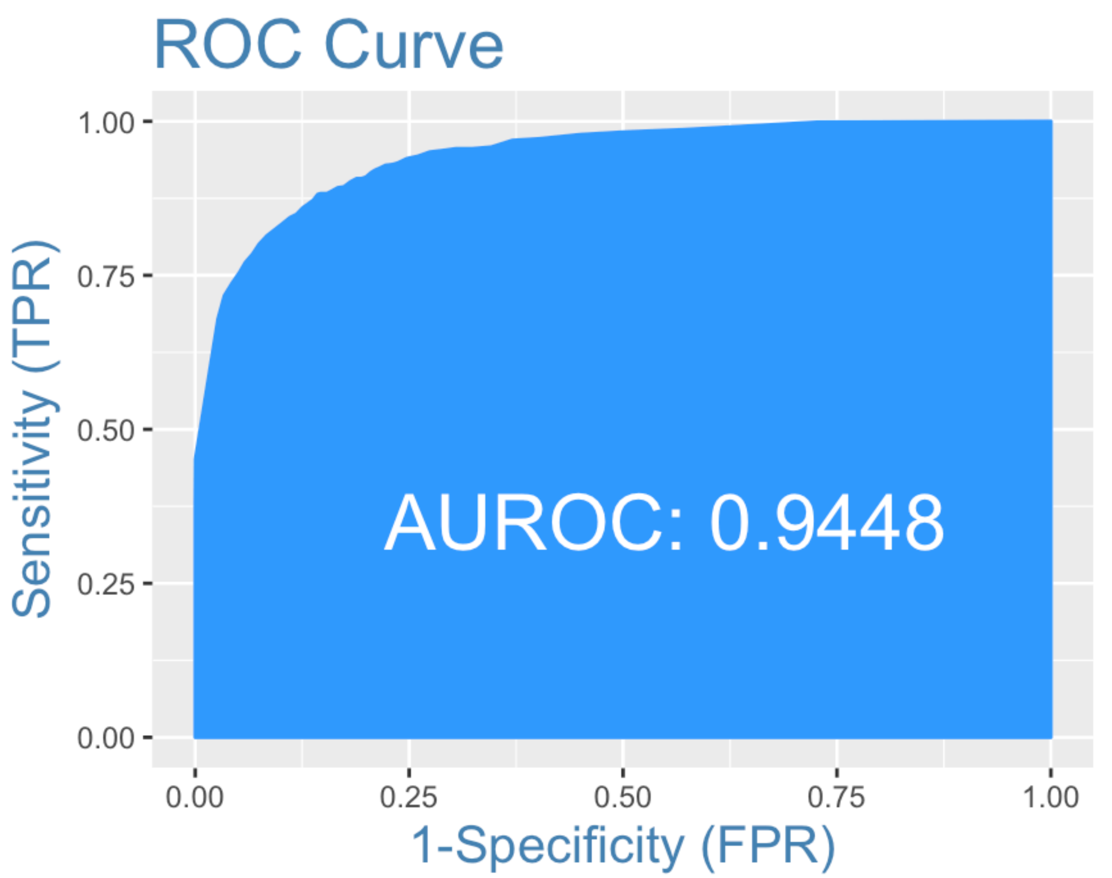
```
# Predicted scores for the final model
predicted=plogis(predict(LogitMod3, TestData))  # predicted scores

# Misclassification Error
misClassError(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
```

```
[1] 0.0814
```

Misclassification error is the percentage mismatch of predicts vs actuals, irrespective of 1’ s or 0’ s. This error value here is small enough which means it’ s a good model.

```
# ROC
plotROC(TestData$IS_TURNOVER, predicted)
```



Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. Greater the area under the ROC curve, better the predictive ability of the model. The above model has area under ROC curve 94.48%, which is pretty good.

```
# Concordance
Concordance(TestData$IS_TURNOVER, predicted)
```

```
$Concordance
[1] 0.9450228
```

In simpler words, of all combinations of 1-0 pairs (actuals), *Concordance* is the percentage of pairs, whose scores of actual positive’ s are greater than the scores of actual negative’ s. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model.

The above model with a concordance of 94.5% is indeed a good quality model.

```
# Specificity and Sensitivity
sensitivity(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
specificity(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
> sensitivity(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
[1] 0.7169811
> specificity(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
[1] 0.9649504
```

Sensitivity (or True Positive Rate) is the percentage of 1' s (actuals) correctly predicted by the model, while, specificity is the percentage of 0' s (actuals) correctly predicted.

Here we see that they are all high enough.

Confusion Matrix

```
# Confusion Matrix
confusionMatrix(TestData$IS_TURNOVER, predicted, threshold = optCutOff)
```

```
     0   1
0 3111 210
1  113 532
```

The columns are actuals, while rows are predicts.

In conclusion, our fitting model is LogitMod3:

LogitMod3= 2.0003 - 0.0036 TOTAL_HOURS + 0.006OT_HRS + 0.028 WEEKEND_COUNT + 0.015 CRITICAL_STAFFING_COUNT