**MSDS 6372 Applied Statistics - Project** 2  
  
**Contributors:** Juan Edgar Nunez, Sterling Beason

# Introduction

The purposes of this project are to demonstrate the ability to perform Exploratory Data Analysis (EDA) and build a logistic regression model. Using a simple logistic regression model as a baseline, perform additional competing models to improve on prediction performance metrics. This project is based on the Global Longitudinal study of Osteoporosis in Women (GLOW). The first part of the analysis provides an answer to a question of relationship between the occurrence of a fracture and other variables in a subset of 500 observations of women in the United States. The goal of the first part of the analysis is to determine a mathematical equation that can be used to predict if a fracture will occur.

The second part of the analysis has the objective of identifying the best prediction model for predicting if a participant in the study will suffer a fracture, based on the risk factor data. The best model will be selected after producing and comparing four different models using various statistical methods. The analysis offers an attempt to provide the best statistical model to predict the participant’s fracture score.

## Data Description

The data for this analysis is included as part of the R’s APLORE3 package. The variables in this GLOW study are FRACTURE (yes or no), PRIORFRACTURE (prior fracture, yes or no), AGE (years), WEIGHT (kilograms), HEIGHT (cm), BMI (body mass index in kg/m2), PREMENO (premenopausal, yes or no), MOMFRAC (mother had hip fracture, yes or no) ARMASSIST (arms are needed to stand from chair, yes or no) SMOKE (former or current smoker, yes or no), and RATERISK (self-reported risk of fracture relative to others of same age, on a scale from 1 to 3).

Some of the variables were codified to facilitate their use in logistic regression. The following are codes and values sheet for variables in the GLOW study.

Variables in the GLOW Study’s code sheet

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Description | Codes/Values | Name |
| 1 | Subject Identification code | 1 to n | SUB\_ID |
| 2 | Study site ID code | 1 to 6 | SITE\_ID |
| 3 | Physician ID code | 128 individual codes | PHY\_ID |
| 4 | Any fracture in first year | 1 = Yes  0 = No | FRACTURE |
| 5 | History of prior fracture | 1 = Yes  0 = No | PRIORFRAC |
| 6 | Age at enrollment | Years | AGE |
| 7 | Weight at enrollment | Kilograms | WEIGHT |
| 8 | Height at enrollment | Centimeters | HEIGHT |
| 9 | Body mass index | Kg/m2 | BMI |
| 10 | Menopause before age 45 | 1 = Yes  0 = No | PREMNO |
| 11 | Mother had hip fracture | 1 = Yes  0 = No | MOMFRAC |
| 12 | Arms are needed to stand from a chair | 1 = Yes  0 = No | ARMASSIST |
| 13 | Former or current smoker | 1 = Yes  0 = No | SMOKE |
| 14 | Self-reported risk of fracture | 1 = Less than others at the same age  2 = Same as others at the same age  3 = Greater than others of the same age | RATERISK |
| 15 | Fracture risk score | Composite risk score | FRACSCORE |

## Analysis Objective 1:

**Restatement of Problem**

We want to predict the log of the odds of fracture for a woman in the United States, that is associated with Osteoporosis risk factors. For that purpose, we want to determine a mathematical equation that allow us to predict if a subject of this study will experience a fracture given the subject’s age, whether or not arms are needed to stand from a chair; the mother had hip fracture, the subject entered menopause before age 45, has history of prior fracture, self-reported risk of fracture (1, 2, or 3); and is the subject a former or current smoker.

For every unit risk factor gained the log of the odds of fracture increases by slope or coefficients ( ) of the, starting at the estimated intersect.

**Build and Fit the Model**

Raterisk = (1, 2, 3)

Note: z value is the estimated intersect divided by the standard error or the number of standard deviation the intercept is away from zero in the standard normal curve. This is the Wald’s test: if an estimate value, including the intersect is less than two standard deviations away from zero, it is not significantly significant, this can be confirmed by the p-value.

|  |
| --- |
| Model.6 <- glm( FRACTURE ~ FRACSCORE+RATERISK+PRIORFRAC ,family = binomial,data = glow5003) |

The variables WEIGHT, HEIGHT, and BMI were removed due to high Variance Inflation Facto (VIF) of 155.83, 19.052, 140.5 respectively, which indicated high multicollinearity; this could affect p-values making the model less reliable.

**Model:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model:** | Model.6 | | |  | | | |
| **Binary Dependent Variable:** | | FRACTURE |  | |  |  | | |
| **Independent Variables:** | |  |  | |  | Variable selection: forward stepwise | | |
| FRACSCORE,RATERISK,PRIORFRAC | |  |  | |  |  |  | |
| **Equation:** |  |  |  | |  |  |  | |
| Model.6 <- glm( FRACTURE ~ FRACSCORE+RATERISK+PRIORFRAC ,family = binomial,data = glow5003) | | | | | | | | |

### Model Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression Statistics: Model.6 for FRACTURE ( 3 variable(s), 6 removed by forward stepwise, n= 500 )** | | | | | | | |
| **R-Squared** | **Adj.R-Sqr.** | **RMSE** | **Mean** | **#Fitted** | **ROC area** | **AIC** |  |
| 0.082 | 0.068 | 0.413 | 0.250 | 500 | 0.707 | 524.1 |  |

### Parameters

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Coefficient Estimates: Model.6 for FRACTURE ( 3 variable(s), 6 removed by forward stepwise, n= 500 )** | | | | | | | |  |
| **Variable** | **Coefficient** | **Std.Err.** | **z statistic** | **P value** | **Lower95%** | **Upper95%** | **VIF** | **Exp.Coeff.** |
| Constant | -2.784 | 0.360 | -7.731 | 0.000 | -3.490 | -2.078 | 0.000 | 0.062 |
| FRACSCORE | 0.200 | 0.049 | 4.060 | 0.000 | 0.103 | 0.296 | 1.279 | 1.221 |
| RATERISK | 0.369 | 0.140 | 2.629 | 0.009 | 0.094 | 0.643 | 1.024 | 1.446 |
| PRIORFRAC | 0.437 | 0.263 | 1.664 | 0.096 | -0.078 | 0.952 | 1.306 | 1.548 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Analysis of Deviance: Model.6 for FRACTURE ( 3 variable(s), 6 removed by forward stepwise, n= 500 )** | | | | | | |
| **Source** | **Deg.Freedom** | **Deviance** | **P-value** |  |  |  |
| Regression | 3 | 46.197 | 0.000 |  |  |  |
| Residual | 496 | 516.138 |  |  |  |  |
| Null | 499 | 562.335 |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Comparative error statistics: Model.6 for FRACTURE ( 3 variable(s), 6 removed by forward stepwise, n= 500 )** | | | | | | | |
|  | **#** | **Mean** | **Model RMSE** | **Const RMSE** | **RMSE ratio** | **MSE(Brier score)** |  |
| FRACTURE | 500 | 0.250 | 0.413 | 0.433 | 0.954 | 0.171 |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Training Set Classification Accuracy With Cutoff Value = 0.5 : Model.6 for FRACTURE ( 3 variable(s), 6 removed by forward stepwise, n= 500 )** | | | | | | | | | |
| **Training results** | **Predict 0** | **Predict 1** | **Total** |  | **Fraction** | **Predict 0** | **Predict 1** | **Total** |  |
| Actual 0 | 358 | 17 | 375 |  | Actual 0 | 71.6% | 3.4% | 75.0% |  |
| Actual 1 | 112 | 13 | 125 |  | Actual 1 | 22.4% | 2.6% | 25.0% |  |
| Total | 470 | 30 | 500 |  | Total | 94.0% | 6.0% | 100.0% |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fraction correct | 74.2% |  | RMSE train | 0.413 |
| True positive rate | 10.4% |  | RMSE const | 0.433 |
| True negative rate | 95.5% |  | Cutoff value | 0.5 |

## Checking Assumptions

**ROC Plot**A close up of a map

Description automatically generated

A screenshot of a social media post

Description automatically generated

### 

### Residual Plots

A screenshot of a map

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Model:** | Model.5 |  |
| **Binary Dependent Variable:** | | FRACTURE |
| **Independent Variables:** | | |
| AGE,ARMASSIST,FRACSCORE,MOMFRAC,PREMENO,PRIORFRAC,RATERISK,SMOKE | | |

|  |
| --- |
| **Equation:** |
| Model.5 <- glm( FRACTURE ~ AGE+ARMASSIST+FRACSCORE+MOMFRAC+PREMENO+PRIORFRAC+RATERISK+SMOKE ,family = binomial,data = glow5003) |

### Model Metrics

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression Statistics: Model.5 for FRACTURE ( 8 variable(s), n= 500 )** | | | | | |  |  |
| **R-Squared** | **Adj.R-Sqr.** | **RMSE** | **Mean** | **#Fitted** | **ROC area** | **AIC** |  |
| 0.088 | 0.056 | 0.412 | 0.250 | 500 | 0.718 | 530.7 |  |
|  |  |  |  |  |  |  |  |

### Parameters

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Coefficient Estimates: Model.5 for FRACTURE ( 8 variable(s), n= 500 )** | | | | |  |  |  |  |
| **Variable** | **Coefficient** | **Std.Err.** | **z statistic** | **P value** | **Lower95%** | **Upper95%** | **VIF** | **Exp.Coeff.** |
| Constant | -4.402 | 2.985 | -1.475 | 0.140 | -10.252 | 1.448 | 0.000 | 0.012 |
| AGE | 0.028 | 0.051 | 0.548 | 0.584 | -0.072 | 0.129 | 18.215 | 1.029 |
| ARMASSIST | 0.231 | 0.489 | 0.473 | 0.636 | -0.727 | 1.189 | 4.959 | 1.260 |
| FRACSCORE | 0.067 | 0.242 | 0.277 | 0.782 | -0.408 | 0.542 | 30.614 | 1.069 |
| MOMFRAC | 0.479 | 0.393 | 1.217 | 0.223 | -0.292 | 1.250 | 1.731 | 1.614 |
| PREMENO | 0.182 | 0.280 | 0.649 | 0.516 | -0.367 | 0.730 | 1.077 | 1.199 |
| PRIORFRAC | 0.602 | 0.360 | 1.675 | 0.094 | -0.103 | 1.307 | 2.418 | 1.826 |
| RATERISK | 0.357 | 0.146 | 2.450 | 0.014 | 0.071 | 0.642 | 1.094 | 1.429 |
| SMOKE | -0.399 | 0.528 | -0.755 | 0.450 | -1.433 | 0.636 | 1.359 | 0.671 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Analysis of Deviance: Model.5 for FRACTURE ( 8 variable(s), n= 500 )** | | | | |
| **Source** | **Deg.Freedom** | **Deviance** | **P-value** |  |
| Regression | 8 | 49.611 | 0.000 |  |
| Residual | 491 | 512.724 |  |  |
| Null | 499 | 562.335 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Comparative error statistics: Model.5 for FRACTURE ( 8 variable(s), n= 500 )** | | | | | |  |
|  | **#** | **Mean** | **Model RMSE** | **Const RMSE** | **RMSE ratio** | **MSE(Brier score)** |
| FRACTURE | 500 | 0.250 | 0.412 | 0.433 | 0.952 | 0.170 |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Training Set Classification Accuracy With Cutoff Value = 0.5 : Model.5 for FRACTURE ( 8 variable(s), n= 500 )** | | | | | | | |  |
| **Training results** | **Predict 0** | **Predict 1** | **Total** |  | **Fraction** | **Predict 0** | **Predict 1** | **Total** |
| Actual 0 | 357 | 18 | 375 |  | Actual 0 | 71.4% | 3.6% | 75.0% |
| Actual 1 | 108 | 17 | 125 |  | Actual 1 | 21.6% | 3.4% | 25.0% |
| Total | 465 | 35 | 500 |  | Total | 93.0% | 7.0% | 100.0% |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fraction correct | 74.8% |  | RMSE train | 0.412 |
| True positive rate | 13.6% |  | RMSE const | 0.433 |
| True negative rate | 95.2% |  | Cutoff value | 0.5 |

**Code in Appendix A**

## Model Summaries:

|  |  |  |
| --- | --- | --- |
| **R Logistic Model For FRACTURE** | Model.5 | Model.6 |
| **Run Time** | 12/2/19 10:19 PM | 12/3/19 7:17 AM |
| **# Fitted** | 500 | 500 |
| **Mean** | 0.250 | 0.250 |
| **Standard Deviation** | 0.433 | 0.433 |
| **#Variables** | 8 | 3 |
| **RMSE** | 0.412 | 0.413 |
| **R-squared** | 0.088 | 0.082 |
| **AIC** | 530.724 | 524.138 |
| **Area Under ROC** | 0.718 | 0.707 |
| **Cutoff Value** | 0.500 | 0.500 |
| **Correct/TruePos/TrueNeg** | 75/14/95 | 74/10/95 |
| **Train/Test Conditions** | All Data | All Data |
|  |  |  |
| **Coefficients:** | Model.5 | Model.6 |
| **Constant** | -4.402 (0.140) | -2.784 (0.000) |
| **AGE** | 0.028 (0.584) |  |
| **ARMASSIST** | 0.231 (0.636) |  |
| **FRACSCORE** | 0.067 (0.782) | 0.200 (0.000) |
| **MOMFRAC** | 0.479 (0.223) |  |
| **PREMENO** | 0.182 (0.516) |  |
| **PRIORFRAC** | 0.602 (0.094) | 0.437 (0.096) |
| **RATERISK** | 0.357 (0.014) | 0.369 (0.009) |
| **SMOKE** | -0.399 (0.450) |  |

### Conclusion

The summary statistics and coefficient table for Model 6 and Model 5 are shown in the Model Metrics section. Model 5 was included to add a little complexity and compare to the logistic regression model of objective 1. R-squared and adjusted R-squared have very small value (0.082, 0.068). In addition, the coefficient estimates are close to zero, making them barely significant. Variables Weight and Height and BMI were removed from the model due to having elevated inflation factor (VIF) of 155.83, 19.052, 140.5 respectively, which is an indicator of significant multicollinearity. The classification table for model 6 and ROC chart for model 6 and model 5 are included and the default cutoff value is 0.5.

With this cutoff value the model achieves fraction correct = 74.2%, true positive rate 10.4%, and true negative rate 95.5%, very low true positive and false positive rates.

Previous models revealed that age does not appear to have a significant impact in predicting who will have a fracture for a subject with a prior fracture.

## Analysis Question 2

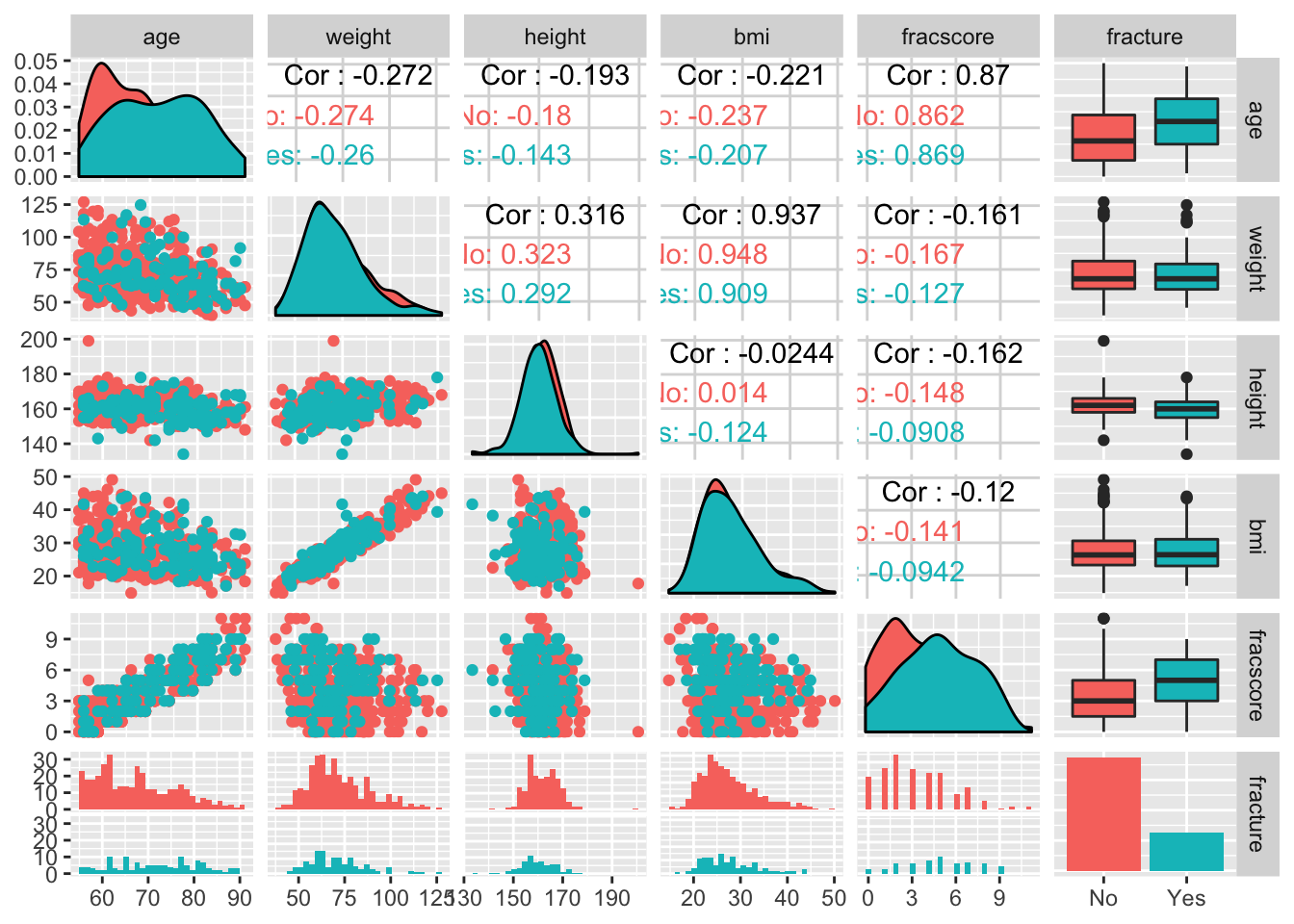
**Objective:**

With Analysis Question 1’s simple logistic regression as a baseline for performance, Analysis Question 2 will provide competing models in hopes of outperforming the simple logistic regression.

**Feature Selection:**

These models were built with either all the predictors available, only numeric, and/or predictors selected from the Random Forest or Multivariate Adaptive Regression (MARS) algorithm.

|  |  |  |
| --- | --- | --- |
| MARS:   * fracscore * height * raterisk * priorfrac * weight * bmi * momfrac | MARS (Numeric):   * fracscore * height * weight * bmi | Random Forest:   * bmi * weight * height * age * fracscore * raterisk   see appendix B for plot |

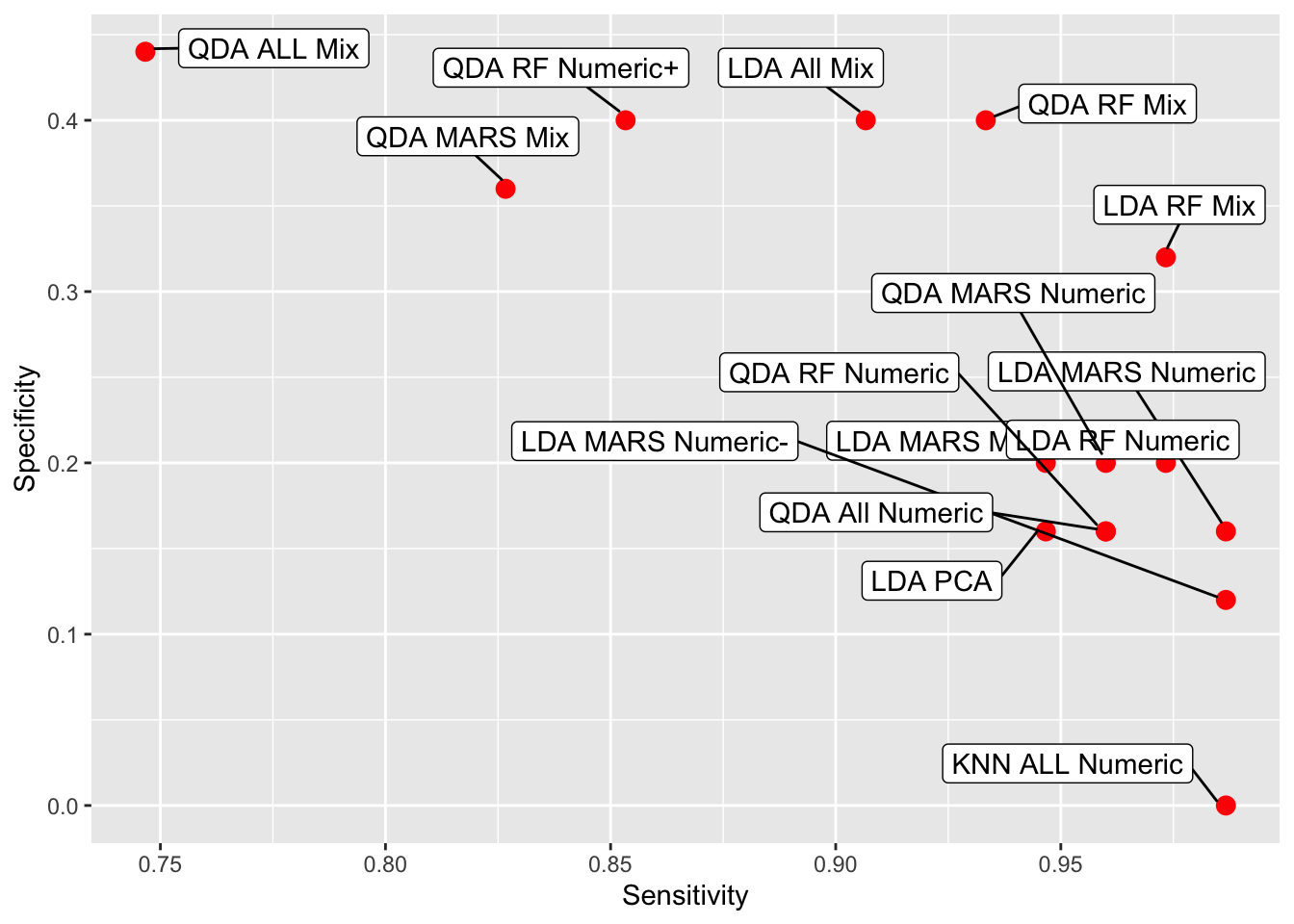


**Competing Models:**

In order to out perform the simple logistic regression model, the following models were used:

* LDA
* QDA
* K-nearest neighbors (kNN)
  + One kNN model was built with a k of 35 after 250 iterations of k selection. (Appendix B)

**Model Performance Summary**



| Model Name | Feature Selection | Numeric Only | Sensitivity | Specificity |
| --- | --- | --- | --- | --- |
| LDA All Mix | All | No | 0.9066667 | 0.40 |
| LDA MARS Mix | MARS | No | 0.9466667 | 0.20 |
| LDA MARS Numeric | MARS | Yes | 0.9866667 | 0.16 |
| LDA MARS Numeric- | MARS | Yes\* | 0.9866667 | 0.12 |
| LDA RF Mix | Random Forest | No | 0.9733333 | 0.32 |
| LDA RF Numeric | Random Forest | Yes | 0.9733333 | 0.20 |
| LDA PCA | All (PCA) | No | 0.9466667 | 0.16 |
| KNN ALL Numeric | All | Yes | 0.9866667 | 0.00 |
| QDA ALL Mix | All | No | 0.7466667 | 0.44 |
| QDA All Numeric | All | Yes | 0.9600000 | 0.16 |
| QDA RF Mix | Random Forest | No | 0.9333333 | 0.40 |
| QDA RF Numeric | Random Forest | Yes | 0.9600000 | 0.16 |
| QDA RF Numeric+ | Random Forest | Yes\*\* | 0.8533333 | 0.40 |
| QDA MARS Mix | MARS | No | 0.8266667 | 0.36 |
| QDA MARS Numeric | MARS | Yes | 0.9600000 | 0.20 |

\* Weight predictor dropped from model against MARS recommendations

\*\* Continuous interaction term was used in conjunction with other continuous predictors (weight\*fracscore)

**Conclusion**

The goal of objective 2 was to use continuous predictors only for the LDA and QDA models. While this in mind, the best continuous only predictors model was the QDA model with with predictors selected from random forest and one interaction term (sensitivity: 0.8533333, specificity: 0.40). The second best continuous only predictors model was QDA with predictors selected from the MARS algorithm (sensitivity: 0.9600000, specificity: 0.20). The model that had the best balance of sensitivity and specificity score that didn’t fit this criteria was QDA with all continuous and categorical predictors (sensitivity: 0.7466667, specificity: 0.44).

*Note: Objective 2 R-code/markdown in Appendix C.*

## Appendix A

writeLines(" -----Begin script glow5003.Model.6.12.03.03.28.47.r")

writeLines(" ")

modelDescription <- "Logistic regression model Model.6 for FRACTURE in data frame glow5003 "

writeLines(modelDescription)

variableSelection <- "Variable selection: forward stepwise "

writeLines(variableSelection)

testMethod <- "Out-of-sample test: none "

writeLines(testMethod)

#Loads the data

glow5003 <- read.table("C:/Users/edgar\_000/OneDrive/SMU/6372 Applied Statistics/Project 2/AppliedStatisticsProject2/Data/glow5003.csv", header = TRUE, sep = "," )

regData <- glow5003

regY <- glow5003$FRACTURE

nData <- nrow(regData)

regScript <- "glow5003.Model.6.12.03.03.28.47.r"

yName <- "FRACTURE"

# ----- setup for a model with no out-of-sample test

rowNumbers <- 1:nData

trainSet <- rowNumbers

trainData <- regData[trainSet,]

nTrain <- nData

nTest <- 0

Model.6 <- glm(FRACTURE ~ AGE + ARMASSIST + BMI + FRACSCORE + MOMFRAC + PREMENO + PRIORFRAC + RATERISK + SMOKE, family = binomial, data = trainData)

constantInd <- 1

Model.6.allvariables <- Model.6

# ----- before stepwise selection, fit the all-variable model and then remove rows with missing values of predictions or dependent variable

trainPred <- predict(Model.6.allvariables, trainData, se.fit= T, type = "response", interval = "prediction", level = .95)

trainErrors <- residuals(Model.6.allvariables, trainData, type = "response", interval = "prediction", level = .95)

trainTable <- na.omit(cbind(trainSet, regY[trainSet], trainPred$fit, trainPred$se.fit))

trainTable <- na.omit(cbind(trainTable,trainErrors))

trainSet <- trainTable[,1]

trainData <- regData[trainSet,]

writeLines(" ")

writeLines(paste("# rows with any missing values removed prior to stepwise selection = ", nTrain - dim(trainTable)[1] ))

writeLines(" ")

# ----- zero-variable logistic model as starting point for forward stepwise selection

nullModel <- glm(FRACTURE ~ 1, family=binomial, data = trainData)

Model.6 <- step(nullModel,scope=list(lower=formula(nullModel),upper=formula(Model.6.allvariables)),direction="forward", trace = 0)

writeLines(paste("# variables removed by forward stepwise selection = ", Model.6.allvariables$rank - Model.6$rank))

stepInfo <- paste(" ",Model.6.allvariables$rank - Model.6$rank,"removed by forward stepwise,")

regModel <- Model.6

modelName <- "Model.6"

trainPred <- predict(regModel, trainData, se.fit= T, type = "response", interval = "prediction", level = .95)

trainErrors <- residuals(regModel, trainData, type = "response", interval = "prediction", level = .95)

# ----- table of training-set actual and predicted values, residuals, deviance residuals and Pearson's residuals

# ----- deviance is the square of the deviance residual

trainTable <- cbind(na.omit(cbind(trainSet, regY[trainSet],trainPred$fit,regY[trainSet]-trainPred$fit)),residuals(regModel,type="deviance"),residuals(regModel,type="pearson"),hatvalues(regModel),cooks.distance(regModel))

# ----- more variables are created for later reference

nMissing <- nTrain - dim(trainTable)[1]

nTrain <- dim(trainTable)[1]

yValues <- trainTable[,2]

yFitted <- trainTable[,3]

RMSEtrain <- (var(trainErrors) \* (nTrain - 1) / nTrain + mean(trainErrors) ^ 2) ^ 0.5

RMSEdigits <- max(0, round(4 - log10(RMSEtrain))) # number of significant digits to use for rounding some statistics in tables

RMSEnull <- (var(yValues)\* (nTrain - 1) / nTrain) ^ 0.5

colnames(trainTable) <- c("Train Row#", "Actual", "Predicted", "Residual","Deviance.Res.", "Pearson.Res.", "Leverage", "Cooks D")

nVariables <- regModel$rank

modelInfo <- paste(modelName,"for",yName,"(",nVariables - constantInd,"variable(s),",stepInfo,"n=",nTrain,")")

# ----- build table of training set results for exporting to Excel

trainTableExport <- rbind(c("Train Row#", "Actual", "Predicted", "Residual", "Std.Res.", "AbsStdRes", "Leverage", "Cooks D"),trainTable)

colnames(trainTableExport) <- rep("",8)

residualDf <- regModel$df.residual

tValue <- qt(0.975, residualDf)

modelDf <- regModel$df.null - regModel$df.residual

modelDev <- regModel$null.deviance - regModel$deviance

modelRsquared <- 1 - regModel$deviance/regModel$null.deviance

modelAdjRsquared <- max(0,1 - (regModel$deviance + 2\*(modelDf + 1))/regModel$null.deviance)

modelMASE <- NA

writeLines(" ")

print(regModel$call)

writeLines(" ")

writeLines(paste("Regression statistics:",modelName,"for",yName,"(#variables =",nVariables-1,")"))

roccurve <- roc(yValues ~ yFitted, data = trainData)

logitStats <- t(matrix(c(modelRsquared,modelAdjRsquared,regModel$df.residual,RMSEtrain,mean(yValues),nTrain,nMissing,roccurve$auc,summary.glm(regModel)$aic)))

print(kable(logitStats, col.names = c("R-Sqr","Adj R-Sqr","Df","RMSE","Mean","#Fitted","#Missing","ROC area","AIC"),digits = c(3,3,0,4,4,0,0,3,3)))

writeLines("")

expCoeff <- exp(summary.glm(regModel)$coefficients[,1])

# ----- calculate variance inflation factors if the model includes a constant and has more than one independent variable

vifValues <- NA

if(grepl("Int", variable.names(regModel)[1]) & nVariables > 1){

vifValues <- c(0,vif(regModel))

}

coeffTable <- cbind(summary.glm(regModel)$coefficients[,c(1,2)],round(summary.glm(regModel)$coefficients[,c(3,4)],6),na.omit(confint.default(regModel, level =.95)),vifValues,expCoeff)

writeLines("Coefficient estimates:")

cDigits <- max(pmax(0,round(3 - log10(as.numeric(coeffTable[,2])),0)))

rownames(coeffTable) <- variable.names(regModel)

coeffTable <-gsub("\_","|",kable(coeffTable, col.names = c("Coeff","StdErr","z stat","P(>\_z\_)","Lower95%","Upper95%","VIF","ExpCoeff"),digits = c(cDigits,cDigits,3,3,cDigits,cDigits,1,3)))

coeffTable <- gsub("NA"," ",coeffTable)

colnames(coeffTable) <- NULL

print(coeffTable, row.names = F)

writeLines(" ")

writeLines("Correlation matrix of coefficient estimates ")

writeLines(" ")

print(cov2cor(vcov(regModel)))

# ----- build single-row table of error statistics in training period

errorStats <- t(matrix(c(nTrain,mean(trainTable[,2]),RMSEtrain,RMSEnull,RMSEtrain/RMSEnull,RMSEtrain^2)))

rownames(errorStats) <- yName

if(nTest>0){

# ----- add 2nd row with error statistics in test period

errorStats <- rbind(errorStats,c(nTest,mean(na.omit(testTable[,2])),RMSEtest,RMSEnulltest,RMSEtest/RMSEnulltest,RMSEtest^2))

rownames(errorStats) <- c("Train","Test")

}

writeLines(" ")

writeLines("Comparative error statistics:")

print(kable(errorStats, col.names = c("#","Mean","ModelRMSE","ConstRMSE","Model/Const","MSE(BrierScore)"),digits= c(0,4,4,4,2,4)))

writeLines(" ")

if(nTest>0){

writeLines("The test set constant model is based on the mean of the dependent variable in the test set.")

}

writeLines(" ")

cutoffValue <- 0.5

writeLines(" ")

writeLines(paste("Classification accuracy with cutoff value = ",cutoffValue))

writeLines("Training set: ")

trainClass <- table(yValues, fitted(regModel) > cutoffValue)

if(as.vector(dim(trainClass)[1])==1){

if(rownames(trainClass)[1] == "FALSE") {

trainClass <- rbind(trainClass,c(0,0))

} else {

trainClass <- rbind(c(0,0),trainClass)

}

}

if(as.vector(dim(trainClass)[2])==1){

if(colnames(trainClass)[1] == "FALSE") {

trainClass <- cbind(trainClass,c(0,0))

} else {

trainClass <- cbind(c(0,0),trainClass)

}

}

rownames(trainClass ) <- c("Actual: 0"," 1")

colnames(trainClass ) <- c("Predicted: 0"," 1")

trainPctCor <- round((trainClass [1, 1] + trainClass [2, 2]) / sum(trainClass ), 4)

trainTruePos <- round(trainClass [2, 2]/ sum(trainClass [2,]), 4)

trainTrueNeg <- round(trainClass [1, 1]/sum(trainClass [1,]), 4)

trainClass <- cbind(trainClass , Total = rowSums(trainClass ))

trainClass <- rbind(trainClass , Total = colSums(trainClass ))

trainStats <- trainClass

rownames(trainStats) <- rep(" ",3)

colnames(trainStats) <- rep(" ",3)

trainStatsExport <- rbind(c("Predict 0","Predict 1","Total","","","Predict 0","Predict 1","Total"),cbind(trainStats,matrix("",nrow=3,ncol=2),trainStats/trainStats[3,3]))

trainStatsExport <- cbind(c("Training results","Actual 0","Actual 1","Total"),trainStatsExport)

trainStats <- rbind(c("Predict 0","Predict 1","Total","","","Predict 0","Predict 1","Total"),cbind(trainStats,matrix("",nrow=3,ncol=2),round(trainStats/trainStats[3,3],3)))

trainStats <- cbind(c("Training results","Actual 0","Actual 1","Total"),trainStats)

trainStatsExport <- rbind(trainStatsExport,rep("",9),c("Fraction correct",trainPctCor,"","RMSE train","","","","",""))

trainStatsExport <- rbind(trainStatsExport,c("True positive rate",trainTruePos,"","RMSE const","","","","",""))

trainStatsExport <- rbind(trainStatsExport,c("True negative rate",trainTrueNeg,"","Cutoff value","","","","",""))

trainStatsExport[6:8,5] <- c(RMSEtrain,RMSEnull,cutoffValue)

trainStatsExport[1:4,6] <- c("Fraction","Actual 0","Actual 1","Total")

trainStats[1:4,6] <- c("Fraction","Actual 0","Actual 1","Total")

trainStatsValues <- trainStats[2:4,2:9]

rownames(trainStatsValues) <- c("Actual 0","Actual 1","Total")

print(kable(trainStatsValues,col.names=c("Predict 0","Predict 1","Total","","Fraction","Predict 0","Predict 1","Total")))

writeLines(" ")

trainStats <- rbind(c(paste("Training set classification accuracy with cutoff value = ",cutoffValue,":"),"","","","","","","",""),trainStats)

trainStats <- rbind(rep("",9),trainStats)

trainStatsExport <- rbind(c(paste("Training Set Classification Accuracy With Cutoff Value = ",cutoffValue,":",modelInfo),"","","","","","","",""),trainStatsExport)

trainStatsExport <- rbind(rep("",9),trainStatsExport)

classStats <- t(matrix(c(nTrain,trainPctCor,trainTruePos,trainTrueNeg)))

rownames(classStats) <- yName

if(nTest>0){

classStats <- rbind(classStats,c(nTest,testPctCor,testTruePos,testTrueNeg))

rownames(classStats) <- c("Train","Test")

}

writeLines(" ")

writeLines("Overall accuracy:")

print(kable(classStats, col.names = c("#","Correct","True Pos","True Neg"),digits = c(0,rep(3,3))))

writeLines(" ")

testTitle = " "

trainSetName = " "

testSetName = " "

mainTitle = paste(modelName," for ",yName,testTitle)

modelSize <- paste("n=", nTrain, ", #var=",length(regModel$coefficients)-1,", ")

layout(matrix(c(1,2,3,4),2,2))

plot(regModel, main = mainTitle, cex.main = 0.95)

layout(matrix(c(1,2),2,1))

yMin = min(na.omit(regY),yValues,yFitted)

yMax = max(na.omit(regY),yValues,yFitted)

plot(trainTable[,1],yValues,main=mainTitle, type = "o", col = "black",lty=1, xlab = NA , ylab= yName, cex.main = 0.95, ylim = c(yMin,yMax))

title(main = (paste(modelSize, trainSetName, "RMSE=",round(RMSEtrain,RMSEdigits))), line = 0.5, cex.main = 0.8)

points(trainTable[,1],yFitted,pch = 16, col = "red",cex=1)

lines(trainTable[,1],yFitted, col = "red",lty=3,cex=0.5)

barplot(yValues- yFitted,main=mainTitle,xlab = NA, ylab= "Residual", cex.main=0.95,ylim = c(-1,1))

title(sub = paste(modelSize, trainSetName, "RMSE=",round(RMSEtrain,RMSEdigits)), cex.sub = 0.8)

writeLines(" ")

trainPred <- predict(regModel,type=c("response"))

plot(roccurve, asp = NA,legacy.axes = TRUE,main = paste (mainTitle, " AUC = ", round(print(roccurve$auc),3)))

# ----- build a text string with the model equation

modelEquation <- paste("Model.6 <- glm(",as.character(formula(Model.6)[2])," ~ ",paste(names(coef(Model.6)), collapse='+'),",family = binomial,data = glow5003)")

modelEquation <- sub("(Intercept)+","",modelEquation,fixed=TRUE)

**============================================= MODEL 5 ==============================================**

writeLines(" -----Begin script glow5003.Model.5.12.02.21.58.32.r produced by RegressItLogistic version 2019.09.25 on CONSOLE001 at time 12.02.21.58.32 ")

writeLines(" ")

modelDescription <- "Logistic regression model Model.5 for FRACTURE in data frame glow5003 "

writeLines(modelDescription)

variableSelection <- "Variable selection: all "

writeLines(variableSelection)

testMethod <- "Out-of-sample test: none "

writeLines(testMethod)

#Loads the data

glow5003 <- read.table("C:/Users/edgar\_000/OneDrive/SMU/6372 Applied Statistics/Project 2/AppliedStatisticsProject2/Data/glow5002.csv", header = TRUE, sep = "," )

regData <- glow5003

regY <- glow5003$FRACTURE

nData <- nrow(regData)

regScript <- "glow5003.Model.5.12.02.21.58.32.r"

yName <- "FRACTURE"

# ----- setup for a model with no out-of-sample test

rowNumbers <- 1:nData

trainSet <- rowNumbers

trainData <- regData[trainSet,]

nTrain <- nData

nTest <- 0

Model.5 <- glm(FRACTURE ~ AGE + ARMASSIST + FRACSCORE + MOMFRAC + PREMENO + PRIORFRAC + RATERISK + SMOKE, family = binomial, data = trainData)

constantInd <- 1

stepInfo <- ""

regModel <- Model.5

modelName <- "Model.5"

trainPred <- predict(regModel, trainData, se.fit= T, type = "response", interval = "prediction", level = .95)

trainErrors <- residuals(regModel, trainData, type = "response", interval = "prediction", level = .95)

# ----- table of training-set actual and predicted values, residuals, deviance residuals and Pearson's residuals

# ----- deviance is the square of the deviance residual

trainTable <- cbind(na.omit(cbind(trainSet, regY[trainSet],trainPred$fit,regY[trainSet]-trainPred$fit)),residuals(regModel,type="deviance"),residuals(regModel,type="pearson"),hatvalues(regModel),cooks.distance(regModel))

# ----- more variables are created for later reference

nMissing <- nTrain - dim(trainTable)[1]

nTrain <- dim(trainTable)[1]

yValues <- trainTable[,2]

yFitted <- trainTable[,3]

RMSEtrain <- (var(trainErrors) \* (nTrain - 1) / nTrain + mean(trainErrors) ^ 2) ^ 0.5

RMSEdigits <- max(0, round(4 - log10(RMSEtrain))) # number of significant digits to use for rounding some statistics in tables

RMSEnull <- (var(yValues)\* (nTrain - 1) / nTrain) ^ 0.5

colnames(trainTable) <- c("Train Row#", "Actual", "Predicted", "Residual","Deviance.Res.", "Pearson.Res.", "Leverage", "Cooks D")

nVariables <- regModel$rank

modelInfo <- paste(modelName,"for",yName,"(",nVariables - constantInd,"variable(s),",stepInfo,"n=",nTrain,")")

# ----- build table of training set results for exporting to Excel

trainTableExport <- rbind(c("Train Row#", "Actual", "Predicted", "Residual", "Std.Res.", "AbsStdRes", "Leverage", "Cooks D"),trainTable)

colnames(trainTableExport) <- rep("",8)

residualDf <- regModel$df.residual

tValue <- qt(0.975, residualDf)

modelDf <- regModel$df.null - regModel$df.residual

modelDev <- regModel$null.deviance - regModel$deviance

modelRsquared <- 1 - regModel$deviance/regModel$null.deviance

modelAdjRsquared <- max(0,1 - (regModel$deviance + 2\*(modelDf + 1))/regModel$null.deviance)

modelMASE <- NA

writeLines(" ")

print(regModel$call)

writeLines(" ")

writeLines(paste("Regression statistics:",modelName,"for",yName,"(#variables =",nVariables-1,")"))

roccurve <- roc(yValues ~ yFitted, data = trainData)

logitStats <- t(matrix(c(modelRsquared,modelAdjRsquared,regModel$df.residual,RMSEtrain,mean(yValues),nTrain,nMissing,roccurve$auc,summary.glm(regModel)$aic)))

print(kable(logitStats, col.names = c("R-Sqr","Adj R-Sqr","Df","RMSE","Mean","#Fitted","#Missing","ROC area","AIC"),digits = c(3,3,0,4,4,0,0,3,3)))

writeLines("")

expCoeff <- exp(summary.glm(regModel)$coefficients[,1])

# ----- calculate variance inflation factors if the model includes a constant and has more than one independent variable

vifValues <- NA

if(grepl("Int", variable.names(regModel)[1]) & nVariables > 1){

vifValues <- c(0,vif(regModel))

}

coeffTable <- cbind(summary.glm(regModel)$coefficients[,c(1,2)],round(summary.glm(regModel)$coefficients[,c(3,4)],6),na.omit(confint.default(regModel, level =.95)),vifValues,expCoeff)

writeLines("Coefficient estimates:")

cDigits <- max(pmax(0,round(3 - log10(as.numeric(coeffTable[,2])),0)))

rownames(coeffTable) <- variable.names(regModel)

coeffTable <-gsub("\_","|",kable(coeffTable, col.names = c("Coeff","StdErr","z stat","P(>\_z\_)","Lower95%","Upper95%","VIF","ExpCoeff"),digits = c(cDigits,cDigits,3,3,cDigits,cDigits,1,3)))

coeffTable <- gsub("NA"," ",coeffTable)

colnames(coeffTable) <- NULL

print(coeffTable, row.names = F)

writeLines(" ")

writeLines("Correlation matrix of coefficient estimates ")

writeLines(" ")

print(cov2cor(vcov(regModel)))

# ----- build single-row table of error statistics in training period

errorStats <- t(matrix(c(nTrain,mean(trainTable[,2]),RMSEtrain,RMSEnull,RMSEtrain/RMSEnull,RMSEtrain^2)))

rownames(errorStats) <- yName

if(nTest>0){

# ----- add 2nd row with error statistics in test period

errorStats <- rbind(errorStats,c(nTest,mean(na.omit(testTable[,2])),RMSEtest,RMSEnulltest,RMSEtest/RMSEnulltest,RMSEtest^2))

rownames(errorStats) <- c("Train","Test")

}

writeLines(" ")

writeLines("Comparative error statistics:")

print(kable(errorStats, col.names = c("#","Mean","ModelRMSE","ConstRMSE","Model/Const","MSE(BrierScore)"),digits= c(0,4,4,4,2,4)))

writeLines(" ")

if(nTest>0){

writeLines("The test set constant model is based on the mean of the dependent variable in the test set.")

}

writeLines(" ")

cutoffValue <- 0.5

writeLines(" ")

writeLines(paste("Classification accuracy with cutoff value = ",cutoffValue))

writeLines("Training set: ")

trainClass <- table(yValues, fitted(regModel) > cutoffValue)

if(as.vector(dim(trainClass)[1])==1){

if(rownames(trainClass)[1] == "FALSE") {

trainClass <- rbind(trainClass,c(0,0))

} else {

trainClass <- rbind(c(0,0),trainClass)

}

}

if(as.vector(dim(trainClass)[2])==1){

if(colnames(trainClass)[1] == "FALSE") {

trainClass <- cbind(trainClass,c(0,0))

} else {

trainClass <- cbind(c(0,0),trainClass)

}

}

rownames(trainClass ) <- c("Actual: 0"," 1")

colnames(trainClass ) <- c("Predicted: 0"," 1")

trainPctCor <- round((trainClass [1, 1] + trainClass [2, 2]) / sum(trainClass ), 4)

trainTruePos <- round(trainClass [2, 2]/ sum(trainClass [2,]), 4)

trainTrueNeg <- round(trainClass [1, 1]/sum(trainClass [1,]), 4)

trainClass <- cbind(trainClass , Total = rowSums(trainClass ))

trainClass <- rbind(trainClass , Total = colSums(trainClass ))

trainStats <- trainClass

rownames(trainStats) <- rep(" ",3)

colnames(trainStats) <- rep(" ",3)

trainStatsExport <- rbind(c("Predict 0","Predict 1","Total","","","Predict 0","Predict 1","Total"),cbind(trainStats,matrix("",nrow=3,ncol=2),trainStats/trainStats[3,3]))

trainStatsExport <- cbind(c("Training results","Actual 0","Actual 1","Total"),trainStatsExport)

trainStats <- rbind(c("Predict 0","Predict 1","Total","","","Predict 0","Predict 1","Total"),cbind(trainStats,matrix("",nrow=3,ncol=2),round(trainStats/trainStats[3,3],3)))

trainStats <- cbind(c("Training results","Actual 0","Actual 1","Total"),trainStats)

trainStatsExport <- rbind(trainStatsExport,rep("",9),c("Fraction correct",trainPctCor,"","RMSE train","","","","",""))

trainStatsExport <- rbind(trainStatsExport,c("True positive rate",trainTruePos,"","RMSE const","","","","",""))

trainStatsExport <- rbind(trainStatsExport,c("True negative rate",trainTrueNeg,"","Cutoff value","","","","",""))

trainStatsExport[6:8,5] <- c(RMSEtrain,RMSEnull,cutoffValue)

trainStatsExport[1:4,6] <- c("Fraction","Actual 0","Actual 1","Total")

trainStats[1:4,6] <- c("Fraction","Actual 0","Actual 1","Total")

trainStatsValues <- trainStats[2:4,2:9]

rownames(trainStatsValues) <- c("Actual 0","Actual 1","Total")

print(kable(trainStatsValues,col.names=c("Predict 0","Predict 1","Total","","Fraction","Predict 0","Predict 1","Total")))

writeLines(" ")

trainStats <- rbind(c(paste("Training set classification accuracy with cutoff value = ",cutoffValue,":"),"","","","","","","",""),trainStats)

trainStats <- rbind(rep("",9),trainStats)

trainStatsExport <- rbind(c(paste("Training Set Classification Accuracy With Cutoff Value = ",cutoffValue,":",modelInfo),"","","","","","","",""),trainStatsExport)

trainStatsExport <- rbind(rep("",9),trainStatsExport)

classStats <- t(matrix(c(nTrain,trainPctCor,trainTruePos,trainTrueNeg)))

rownames(classStats) <- yName

if(nTest>0){

classStats <- rbind(classStats,c(nTest,testPctCor,testTruePos,testTrueNeg))

rownames(classStats) <- c("Train","Test")

}

writeLines(" ")

writeLines("Overall accuracy:")

print(kable(classStats, col.names = c("#","Correct","True Pos","True Neg"),digits = c(0,rep(3,3))))

writeLines(" ")

testTitle = " "

trainSetName = " "

testSetName = " "

mainTitle = paste(modelName," for ",yName,testTitle)

modelSize <- paste("n=", nTrain, ", #var=",length(regModel$coefficients)-1,", ")

layout(matrix(c(1,2,3,4),2,2))

plot(regModel, main = mainTitle, cex.main = 0.95)

layout(matrix(c(1,2),2,1))

yMin = min(na.omit(regY),yValues,yFitted)

yMax = max(na.omit(regY),yValues,yFitted)

plot(trainTable[,1],yValues,main=mainTitle, type = "o", col = "black",lty=1, xlab = NA , ylab= yName, cex.main = 0.95, ylim = c(yMin,yMax))

title(main = (paste(modelSize, trainSetName, "RMSE=",round(RMSEtrain,RMSEdigits))), line = 0.5, cex.main = 0.8)

points(trainTable[,1],yFitted,pch = 16, col = "red",cex=1)

lines(trainTable[,1],yFitted, col = "red",lty=3,cex=0.5)

barplot(yValues- yFitted,main=mainTitle,xlab = NA, ylab= "Residual", cex.main=0.95,ylim = c(-1,1))

title(sub = paste(modelSize, trainSetName, "RMSE=",round(RMSEtrain,RMSEdigits)), cex.sub = 0.8)

writeLines(" ")

trainPred <- predict(regModel,type=c("response"))

plot(roccurve, asp = NA,legacy.axes = TRUE,main = paste (mainTitle, " AUC = ", round(print(roccurve$auc),3)))

# ----- build a text string with the model equation

modelEquation <- paste("Model.5 <- glm(",as.character(formula(Model.5)[2])," ~ ",paste(names(coef(Model.5)), collapse='+'),",family = binomial,data = glow5003)")

modelEquation <- sub("(Intercept)+","",modelEquation,fixed=TRUE)

## Appendix B

Training Set Results for Objective 1: Model.6 and Training Set Results: Model.5 are in the attachment file: Training Set Results Model.5 for FRACTURE ( 8 variable(s), n= 500 )

|  |
| --- |
| Random Forest Feature Selection |
| kNN k Selection |

## Appendix C

See attachment “appendix\_c.(Rmd|html) for objective 2 analysis and R code.