# STAT 636: Final Project

Katie Hill, Asmita Nagila, Jerrin Wiley

2023-12-06

```
library(dplyr)
library(ggplot2) #Plots for visualization
library(glmnet)
library(corrplot) #Correlation Heatmap
library(pROC) #Roc Curve
library(e1071)
library(caret)
```

## Project Data Source

https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success

Instances: 4424

Features: 36

Goal: Classify Students into 3 categories: Dropout, Enrolled, and Graduate

## Data Import

Import Data

```
data <- read.csv("data.csv",sep=";")
attach(data)
```

```
# Suppressed for pdf output
#head(data)
```

Check for Missing Values

```
which(is.na(data))
```

```
## integer(0)
```

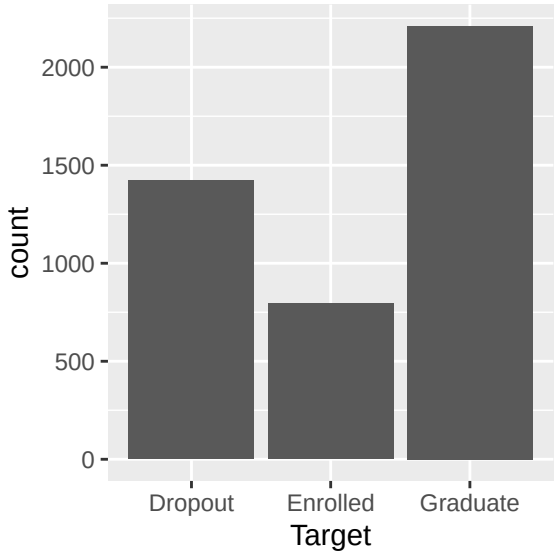
No line is missing data

Convert Target values to numerical values. Dropout = 1, Enrolled = 2, Graduate = 3

```
model_data <- data
model_data$Target <- as.numeric(factor(Target))
```

## Data Exploration

```
ggplot(data=data)+geom_bar(mapping=aes(x=Target))
```




Build function to graph distribution of feature by target

```
FeatureVsTarget = function(feature,name){
  print(ggplot(data=data)+
          geom_boxplot(mapping=aes(x=reorder(Target,feature,FUN=median),y=feature))+xlab("Target")+ylab
}
```

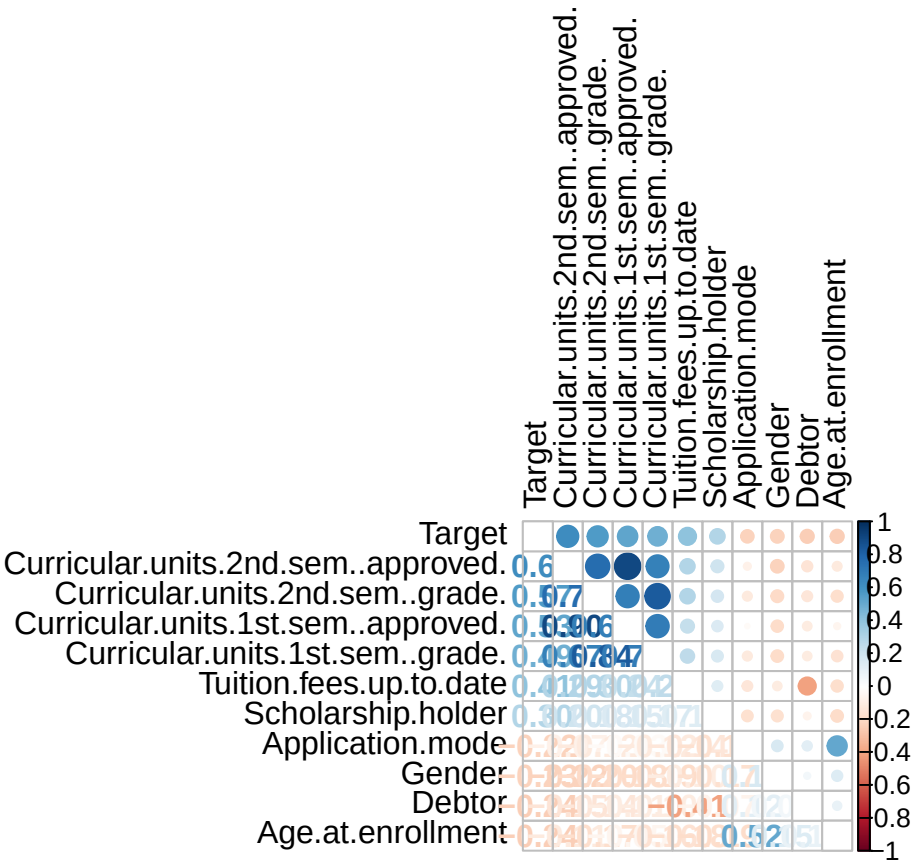
Plot Features

```
# Suppressed for pdf output. These plots were used to explore data relationships

#data_col <- colnames(model_data)
#for(i in 1:ncol(model_data)){
#  FeatureVsTarget(model_data[,i], data_col[i])
#}
```

```
numericVars <- which(sapply(model_data, is.numeric)) #index vector numeric variables
numericVarNames <- names(numericVars) #saving names vector for use later on
cat('There are', length(numericVars), 'numeric variables')
```

```
## There are 37 numeric variables
```

```
all_numVar <- model_data[, numericVars]
cor_numVar <- cor(all_numVar, use="pairwise.complete.obs")

cor_sorted <- as.matrix(sort(cor_numVar[,'Target'], decreasing = TRUE))

 #select highest correlations
CorHigh <- names(which(apply(cor_sorted, 1, function(x) abs(x)>0.2)))
cor_numVar <- cor_numVar[CorHigh, CorHigh]

corrplot.mixed(cor_numVar, tl.col="black", tl.pos = "lt")
```




## Data Split

70% train, 30% split (Will not be used for feature selection, k-fold cross-validation will be)

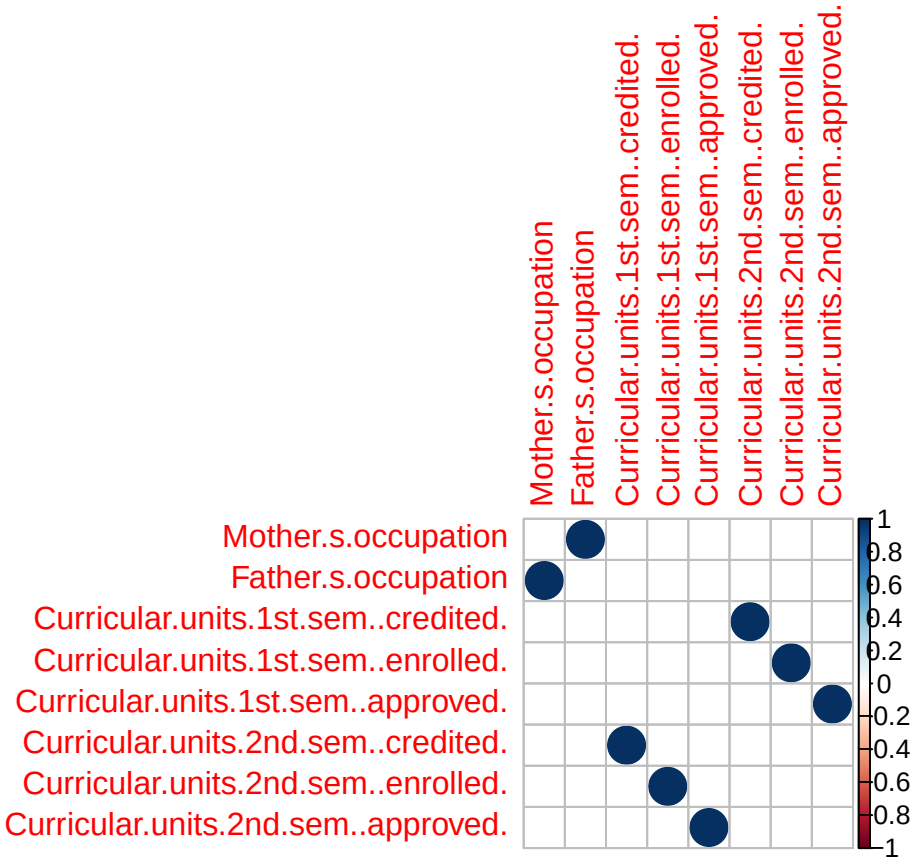
```
sample <- sample(c(TRUE,FALSE),nrow(data),replace=TRUE,prob=c(0.7,0.3))
train <- data[sample,]
test <- data[!sample,]
```

## Data Prep

Considering dropping highly correlated variable to counteract their possible dependent relationship. First, find correlations over 0.85 within X

```
x <- all_numVar
data_cor <- cor(x)

#Create function to find correlations over 0.85
corr_find <- function(x){
  return (ifelse(x>=0.85,ifelse(x==1,0,1),0))
}

#high_corr <- data_cor[colSums(abs(data_cor))>0.85,]
data_cor[data_cor==1]<-0
data_cor[data_cor>=0.85]<-1
data_cor[data_cor<0.85]<-0
high_corr<-as.data.frame(data_cor)
high_corr <- high_corr[colSums(abs(high_corr),na.rm=TRUE)>0]
```

```
high_corr <- high_corr[rowSums(abs(high_corr))>0,]
corrplot(as.matrix(high_corr))
```




Next, for all pairs find the feature with the lowest correlation to Target

- mother.s.occupation vs father.s.occupation = 0.0056 vs 0.0019
  - father.s.occupation
- Curricular.units.1st.sem..credited. vs Curricular.units.2nd.sem..credited. = 0.0481 vs 0.0540
  - Curricular.units.1st.sem..credited.
- Curricular.units.1st.sem..enrolled. vs Curricular.units.2nd.sem..enrolled. =0.1560 vs 0.1758
  - Curricular.units.1st.sem..enrolled
- Curricular.units.1st.sem..approved. vs Curricular.units.2nd.sem..approved. = 0.5291 vs 0.6242
  - Curricular.units.1st.sem..approved.

```
abs(cor_sorted["Mother.s.occupation",])
```

```
## Mother.s.occupation
##         0.005628565
```

```
abs(cor_sorted["Father.s.occupation",])
```

```
## Father.s.occupation
##         0.001898935
```

```r
abs(cor_sorted["Curricular.units.1st.sem..credited.",])
```

```
## Curricular.units.1st.sem..credited.
##                          0.04814971
```

```r
abs(cor_sorted["Curricular.units.2nd.sem..credited.",])
```

```
## Curricular.units.2nd.sem..credited.
##                          0.05400381
```

```r
abs(cor_sorted["Curricular.units.1st.sem..enrolled.",])
```

```
## Curricular.units.1st.sem..enrolled.
##                           0.155974
```

```r
abs(cor_sorted["Curricular.units.2nd.sem..enrolled.",])
```

```
## Curricular.units.2nd.sem..enrolled.
##                           0.1758468
```

```r
abs(cor_sorted["Curricular.units.1st.sem..approved.",])
```

```
## Curricular.units.1st.sem..approved.
##                           0.5291233
```

```r
abs(cor_sorted["Curricular.units.2nd.sem..approved.",])
```

```
## Curricular.units.2nd.sem..approved.
##                           0.6241575
```

Drop lowest correlated of each pair

```r
dropFeatures <- c("Mother.s.occupation","Father.s.occupation","Curricular.units.1st.sem..credited.","Cu
model_data <- model_data[,!names(model_data) %in% dropFeatures]
```

Create test and train set from cleaned data

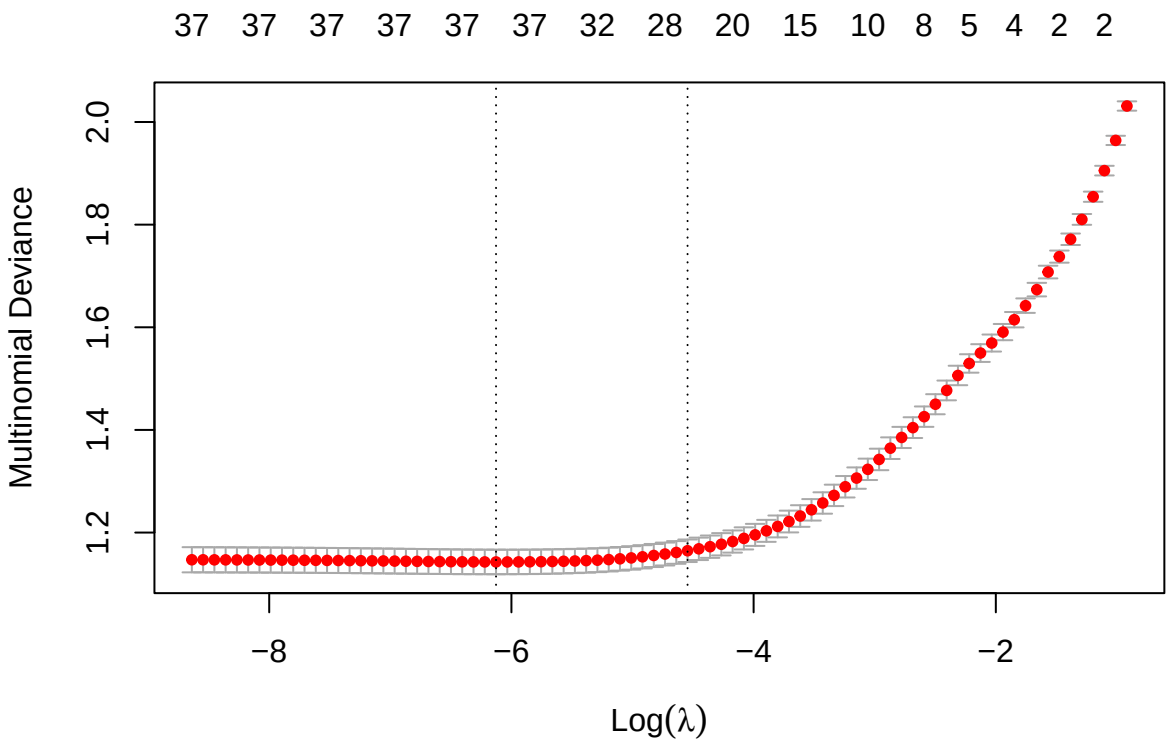
```r
sample2 <- sample(c(TRUE,FALSE),nrow(model_data),replace=TRUE,prob=c(0.7,0.3))
train_clean <- data[sample2,]
test_clean <- data[!sample2,]
```

## Feature Selection - LASSO model

For the Lasso model, we will use the `glmnet` package. The model will be trained on the training set and evaluated on the test set.

```r
# Lasso Model Fitting

# Preparing the data for the Lasso model: matrix format is required for glmn

X_train <- model.matrix(~ . - 1, data = train[, -which(colnames(train) == "Target")])
Y_train <- train$Target

# Fit the Lasso model using cross-validation
set.seed(1217) #for reproducibility
cv_model <- cv.glmnet(X_train, Y_train, family = "multinomial", type.multinomial = "grouped")

# Determine the best lambda (regularization parameter)
best_lambda <- cv_model$lambda.min
```

```r
# Fit the final model on the training data using the best lambda
final_model <- glmnet(X_train, Y_train, family = "multinomial", lambda = best_lambda, type.multinomial =

# View the model coefficients
# Suppressed for pdf output
#coef(final_model, s = best_lambda)

plot(cv_model)
```




Testing the LASSO model

```r
# Preparing the test data in matrix format
X_test <- model.matrix(~ . - 1, data = test[, -which(colnames(test) == "Target")])
Y_test <- test$Target

# Making predictions on the test data
predictions <- predict(final_model, newx = X_test, s = best_lambda, type = "response")

# Converting predictions to the same format as 'Y_test' for comparison
predicted_classes <- apply(predictions, 1, which.max)

# Evaluate the model performance
confusion_matrix <- table(Predicted = predicted_classes, Actual = Y_test)
print(confusion_matrix)
```

```
##           Actual
## Predicted Dropout Enrolled Graduate
```

```
##          1     357       69       22
##          2      40       72       35
##          3      64      107      604
```

```
# Calculating overall accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy: ", accuracy)
```

```
## Accuracy:  0.7540146
```

## Support Vector Machine (SVM) model

```
# Convert target variable to a facor
train$Target <- as.factor(train$Target)

# Fitting SVM to the training data
# Using radial basis kernel (can use others like linear, polynomial)
#for reproducibility
set.seed(1217)
svm_model <- svm(Target ~ ., data = train, method = "C-classification", kernel = "radial")

# Summary of the model
summary(svm_model)
```

```
##
## Call:
## svm(formula = Target ~ ., data = train, method = "C-classification",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1720
##
##  ( 551 625 544 )
##
##
## Number of Classes:  3
##
## Levels:
##  Dropout Enrolled Graduate
```

Testing the SVM

```
# Making predictions on the test data
svm_predictions <- predict(svm_model, newdata = test)

#Convert predictions to factor
svm_predictions <- factor(svm_predictions, levels = levels(train$Target))

#Ensure test target is a factor w/ same levels
test$Target <- factor(test$Target, levels = levels(train$Target))
```

```r
svm_predictions <- predict(svm_model, newdata = test)

# Confusion Matrix to evaluate performance
confusionMatrix(svm_predictions, test$Target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Dropout Enrolled Graduate
##   Dropout      333       43       15
##   Enrolled      43       83       22
##   Graduate      85      122      624
##
## Overall Statistics
##
##                Accuracy : 0.7591
##                  95% CI : (0.7356, 0.7816)
##     No Information Rate : 0.4825
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5929
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Dropout Class: Enrolled Class: Graduate
## Sensitivity                  0.7223         0.33468          0.9440
## Specificity                  0.9362         0.94207          0.7080
## Pos Pred Value               0.8517         0.56081          0.7509
## Neg Pred Value               0.8693         0.86498          0.9314
## Prevalence                   0.3365         0.18102          0.4825
## Detection Rate               0.2431         0.06058          0.4555
## Detection Prevalence         0.2854         0.10803          0.6066
## Balanced Accuracy            0.8293         0.63837          0.8260
```

### Test using cleaned test and train sets for SVM

```r
# Convert target variable to a factor
train_clean$Target <- as.factor(train_clean$Target)

# Fitting SVM to the training data
# Using radial basis kernel (can use others like linear, polynomial)
#for reproducibility
set.seed(1217)
svm_model <- svm(Target ~ ., data = train_clean, method = "C-classification", kernel = "radial")

# Summary of the model
summary(svm_model)
```

```
##
## Call:
## svm(formula = Target ~ ., data = train_clean, method = "C-classification",
##     kernel = "radial")
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1779
##
##  ( 560 663 556 )
##
##
## Number of Classes:  3
##
## Levels:
##  Dropout Enrolled Graduate
```

```
# Making predictions on the test data
svm_predictions <- predict(svm_model, newdata = test)

#Convert predictions to factor
svm_predictions <- factor(svm_predictions, levels = levels(train$Target))

#Ensure test target is a factor w/ same levels
test$Target <- factor(test$Target, levels = levels(train$Target))

svm_predictions <- predict(svm_model, newdata = test)

# Confusion Matrix to evaluate performance
confusionMatrix(svm_predictions, test$Target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Dropout Enrolled Graduate
##   Dropout      354       33       12
##   Enrolled      39      109       16
##   Graduate      68      106      633
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.7778, 0.8209)
##     No Information Rate : 0.4825
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6645
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Dropout Class: Enrolled Class: Graduate
## Sensitivity                  0.7679         0.43952          0.9576
## Specificity                  0.9505         0.95098          0.7546
```

```
## Pos Pred Value              0.8872         0.66463          0.7844
## Neg Pred Value              0.8898         0.88474          0.9503
## Prevalence                  0.3365         0.18102          0.4825
## Detection Rate              0.2584         0.07956          0.4620
## Detection Prevalence        0.2912         0.11971          0.5891
## Balanced Accuracy           0.8592         0.69525          0.8561
```

The method using SVM with the data set that removed attributes that were highly correlated with each other. This was done to prevent those features from having undue weight on the model.