DAT-4253 LM 6 CLASSIFICATION TREES

AUTHOR
Aaron Younger

Business Understanding

Derek Anderson is an institutional researcher at a major university. The university has set a goal to increase the number of first-year freshmen students who graduate within four years by 20% in five years. Derek is asked by his boss to create a model that would flag any freshmen student who has a high likelihood of not being able to graduate within four years to help with early intervention.

Data Understanding

- Sex of Student (Male or Female).
- White; Whether the student is Caucasian.
- HS GPA; The students high school GPA.
- SAT; The students SAT Score.
- GPA; The Students College GPA.
- College Parent; Whether the student had parents that went to college.
- Grad; Whether that student graduated in four years. This is the Dependent Variable, 1 = they graduated in four years 0 = they did not graduate in four years.

```r
## Libraries
library(readxl)
library(tidyverse)
library(DataExplorer)
library(e1071)
library(dlookr)
library(caret)
library(rpart)
library(rpart.plot)
library(gains)
library(pROC)
```

EDA

```r
library(readxl)
Graduate_Data <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "Graduate_Data")
View(Graduate_Data)

library(readxl)
Graduate_data_score <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "Graduate_Score")
View(Graduate_data_score)

Graduate_Data <- Graduate_Data %>%
  select(-GPA)
view(Graduate_Data)



Graduate_data_score <- Graduate_data_score %>%
  select(-GPA)
view(Graduate_data_score)

Graduate_Data <- Graduate_Data %>%
  rename(HS_GPA = 'HS GPA')

Graduate_Data <- Graduate_Data %>%
  rename(College_Parent = 'College Parent')

Graduate_data_score <- Graduate_data_score %>%
  rename(HS_GPA = 'HS GPA')

Graduate_data_score <- Graduate_data_score %>%
  rename(College_Parent = 'College Parent')
```

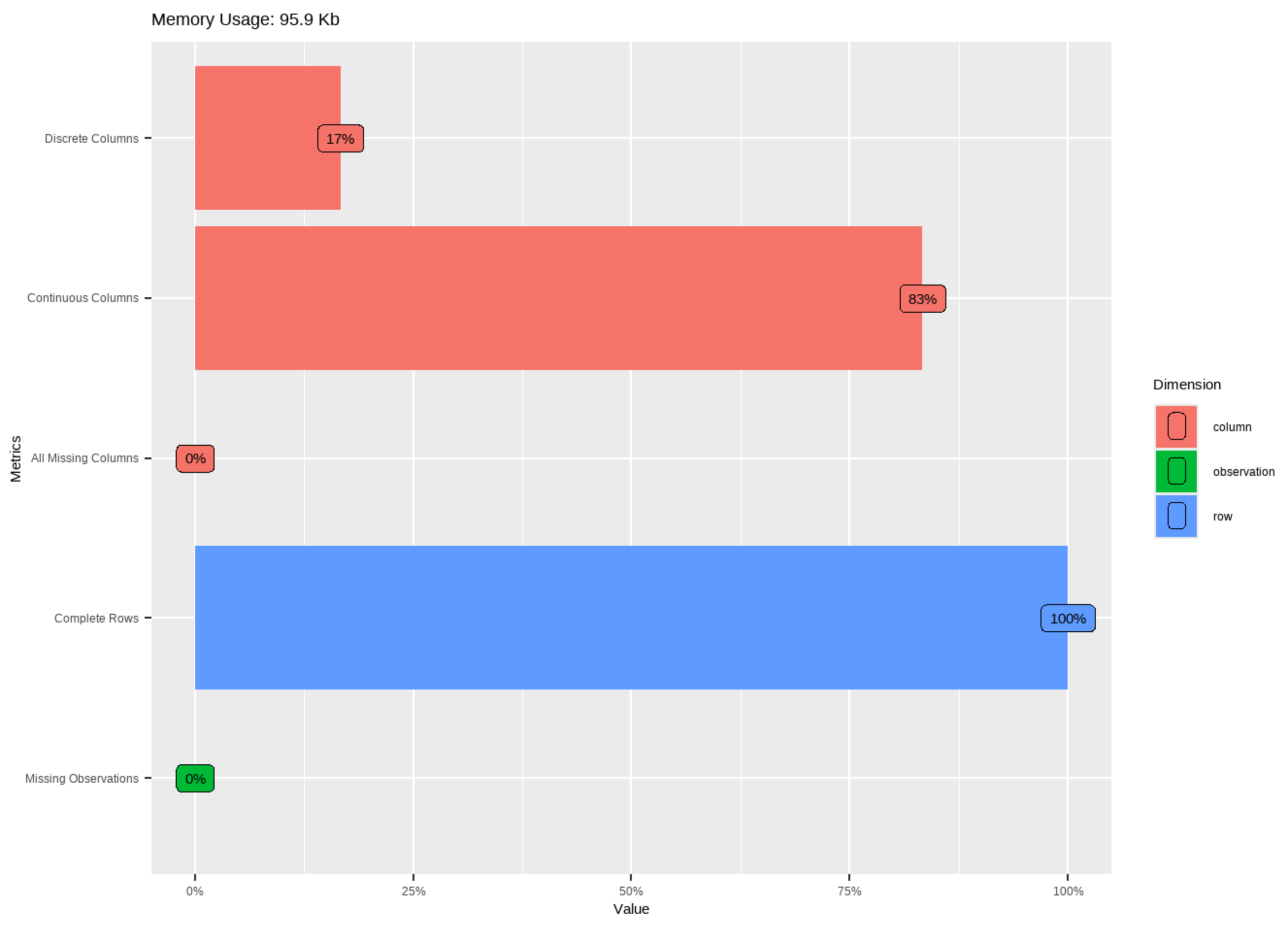
```
Graduate_Data %>% head()
```

```
# A tibble: 6 × 6
  Sex   White HS_GPA   SAT College_Parent  Grad
  <chr> <dbl>  <dbl> <dbl>          <dbl> <dbl>
1 F         1   4.14  1410              1     1
2 M         1   3.3   1260              1     1
3 F         0   4.3    950              0     1
4 M         1   4.29  1290              1     1
5 F         1   4.2   1350              1     1
6 F         0   3.86  1350              1     1
```

```
Graduate_Data %>% tail()
```

```
# A tibble: 6 × 6
  Sex   White HS_GPA   SAT College_Parent  Grad
  <chr> <dbl>  <dbl> <dbl>          <dbl> <dbl>
1 F         1   3.36  1250              1     1
2 M         1   3.18  1400              1     1
3 M         0   3.82  1230              1     0
4 F         0   3.74  1140              0     0
5 M         1   3.75  1260              1     1
6 M         0   3.08   950              0     0
```

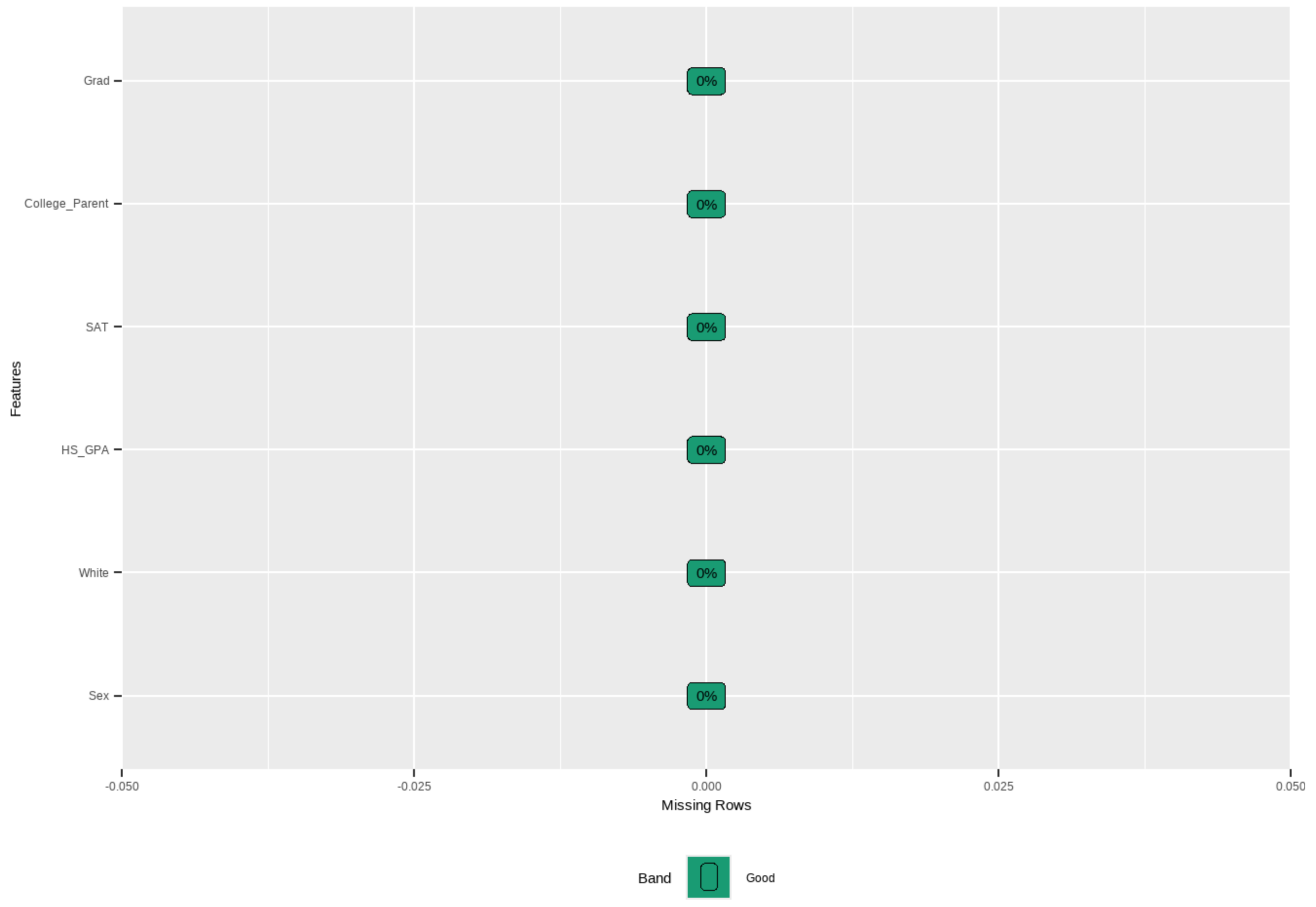
```
Graduate_Data %>% plot_intro
```



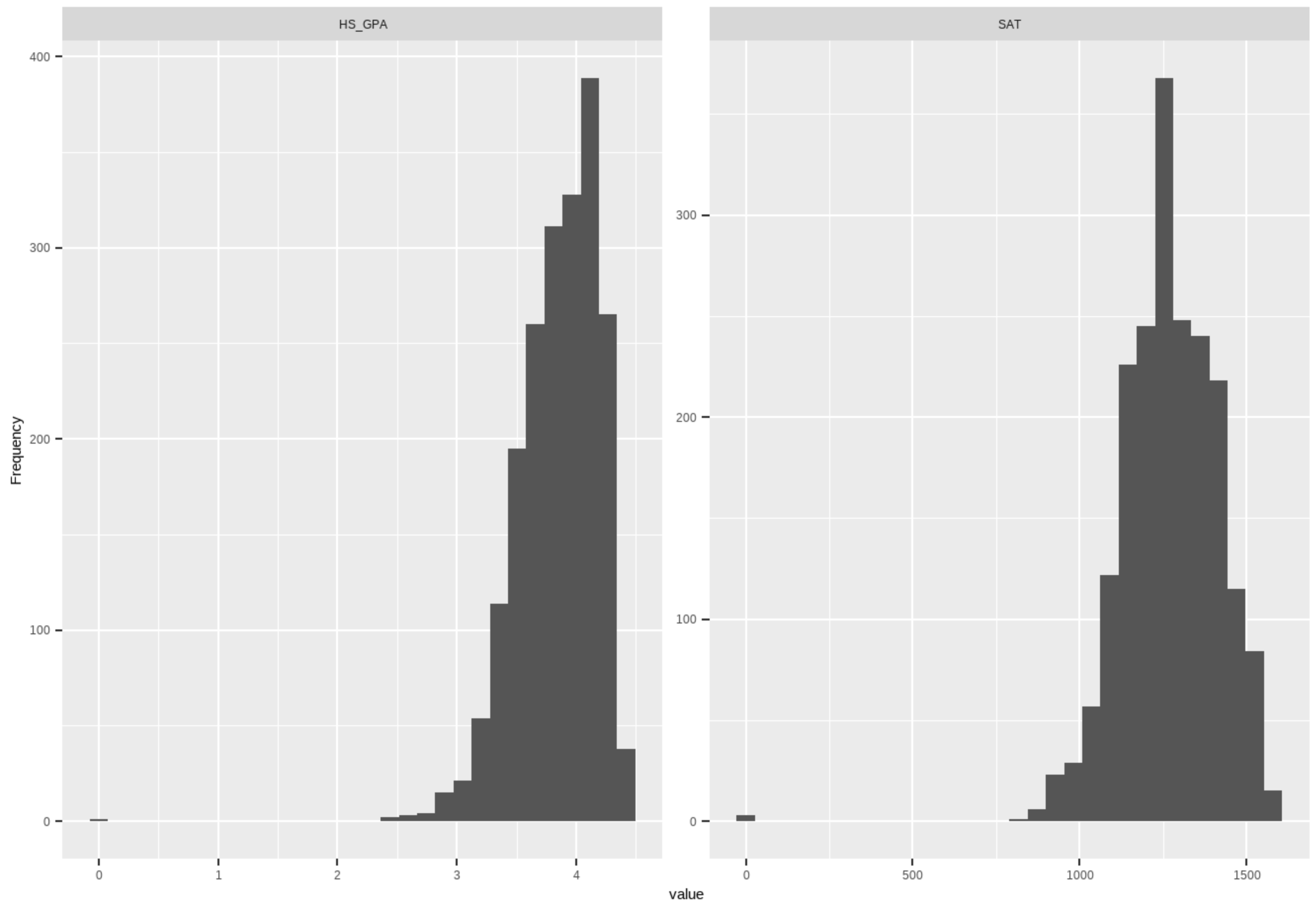
```
Graduate_Data %>% str()
```

```
tibble [2,000 × 6] (S3: tbl_df/tbl/data.frame)
 $ Sex           : chr [1:2000] "F" "M" "F" "M" ...
 $ White         : num [1:2000] 1 1 0 1 1 0 1 0 1 1 ...
 $ HS_GPA        : num [1:2000] 4.14 3.3 4.3 4.29 4.2 3.86 3.75 3.32 3.86 3.91 ...
 $ SAT           : num [1:2000] 1410 1260 950 1290 1350 1350 1180 1180 970 1490 ...
 $ College_Parent: num [1:2000] 1 1 0 1 1 1 1 1 1 1 ...
 $ Grad          : num [1:2000] 1 1 1 1 1 1 1 1 1 1 ...
```

```
Graduate_Data %>% plot_missing()
```



```
Graduate_Data %>% plot_histogram()
```



```
skewness(Graduate_Data$HS_GPA)
```

```
[1] -1.377024
```

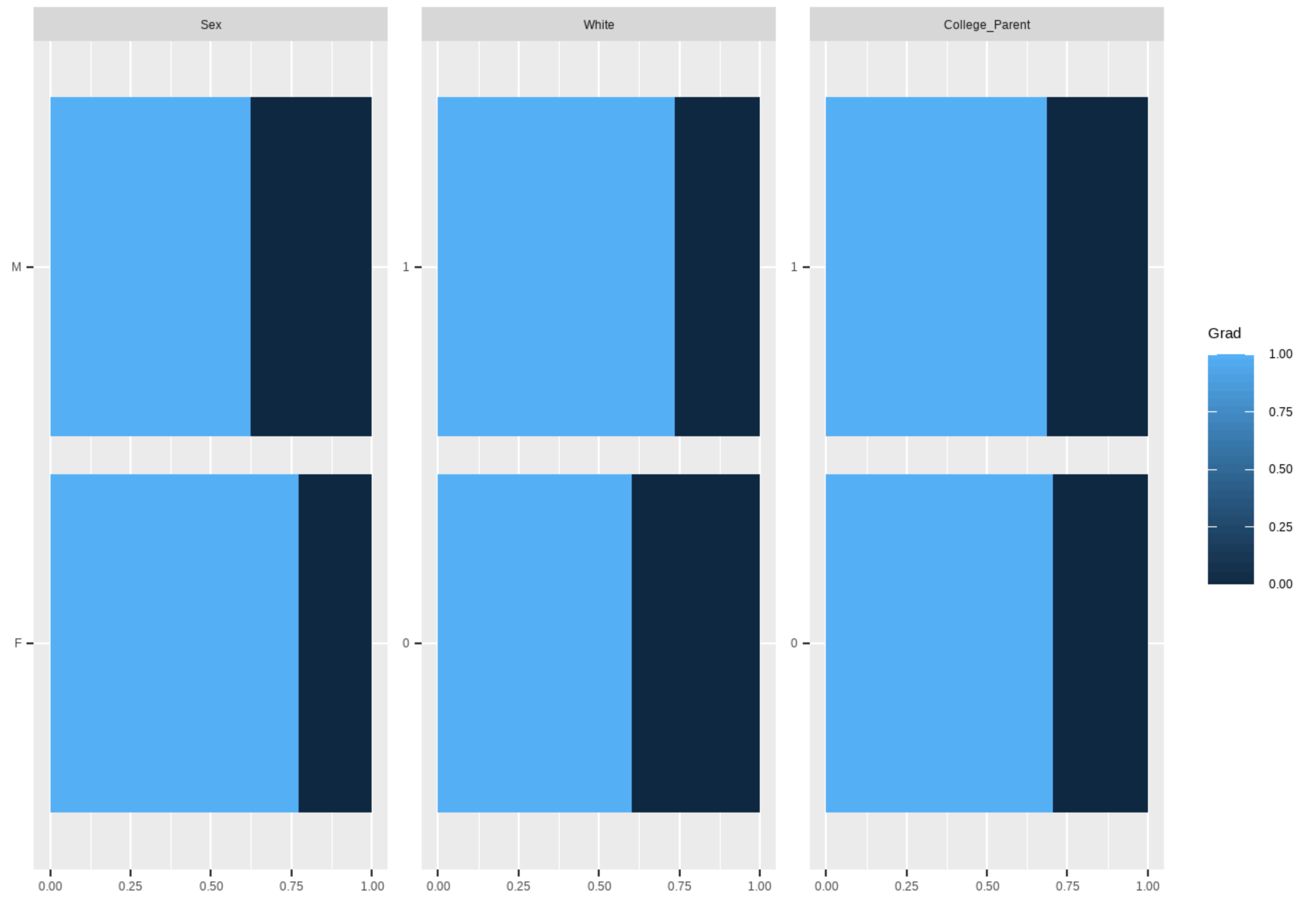
```
skewness(Graduate_Data$GPA)
```

```
[1] NaN
```

```
skewness(Graduate_Data$SAT)
```

```
[1] -1.150264
```

```
Graduate_Data %>% plot_bar(by="Grad")
```



```r
dlookr::diagnose_outlier(Graduate_Data)
```

```
# A tibble: 5 × 6
  variables     outliers_cnt outliers_ratio outliers_mean with_mean without_mean
  <chr>                <int>          <dbl>         <dbl>     <dbl>        <dbl>
1 White                    0           0              NaN      0.680        0.680
2 HS_GPA                  25           1.25             2.69   3.85         3.87
3 SAT                     16           0.8            719.  1272.        1276.
4 College_Pare…            0           0              NaN      0.686        0.686
5 Grad                     0           0              NaN      0.692        0.692
```

```r
boxplot(Graduate_Data$SAT, plot = FALSE)$out
```

```
 [1] 880 880 900 850   0 900   0 910 880   0 840 860 910 910 900 880
```

```r
range(Graduate_Data$SAT) ## Some students have scored 0 on their SAT ??
```

```
[1]    0 1580
```

```r
range(Graduate_Data$GPA)
```

```
[1]  Inf -Inf
```

```
range(Graduate_Data$`HS GPA`)
```

```
[1]  Inf -Inf
```

```
Graduate_Data %>%
  filter(SAT == 0) %>%
  select(SAT, Grad)
```

```
# A tibble: 3 × 2
    SAT  Grad
  <dbl> <dbl>
1     0     0
2     0     1
3     0     1
```

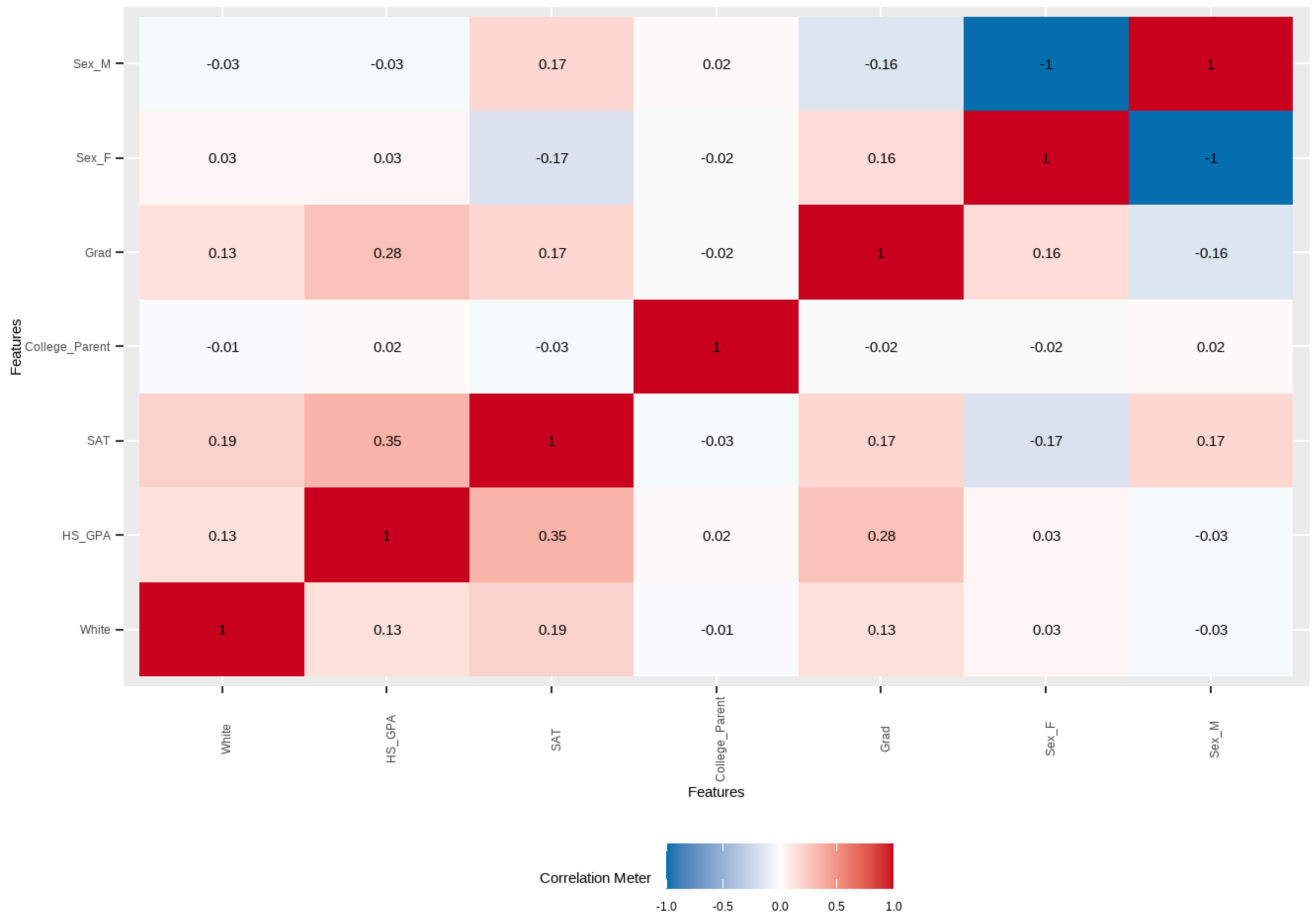
```
## Imbalanced Dataset, More 1's than zeros, model will most likely favor predicting ones.
cat("Proportion of Dependent Variable \n")
```

```
Proportion of Dependent Variable
```

```
prop.table(table(Graduate_Data$Grad))
```

```

    0     1
0.308 0.692
```

```
Graduate_Data %>% plot_correlation() ## The GPA variables have the highest positive correlation
         to students graduating in four years
```



	White	HS_GPA	SAT	College_Parent	Grad	Sex_F	Sex_M
Sex_M	-0.03	-0.03	0.17	0.02	-0.16	-1	1
Sex_F	0.03	0.03	-0.17	-0.02	0.16	1	-1
Grad	0.13	0.28	0.17	-0.02	1	0.16	-0.16
College_Parent	-0.01	0.02	-0.03	1	-0.02	-0.02	0.02
SAT	0.19	0.35	1	-0.03	0.17	-0.17	0.17
HS_GPA	0.13	1	0.35	0.02	0.28	0.03	-0.03
White	1	0.13	0.19	-0.01	0.13	0.03	-0.03

Comments on EDA:
This dataset contains no missing values. The numeric values are relatively symmetrical all having slight left skewness. The dependent variable is seen in all categorical variables. Outliers are present in the variables HS GPA, SAT, and GPA however the outliers do not significantly affect the mean so were kept in as observations. The dataset is imbalanced, the dependent variable has majority 1's which will be something to reconcile in modeling. Both HS GPA and GPA have the highest positive correlation to the students graduating in four years. It is important to note that for modeling, the model is supposed to be detecting students that are unlikely to graduate in four years, which is the 0 value of the dependent variable.

Data Preperation

```
Graduate_Data$Sex <- as.factor(Graduate_Data$Sex)
Graduate_Data$White <- as.factor(Graduate_Data$White)
Graduate_Data$College_Parent <- as.factor(Graduate_Data$College_Parent)
Graduate_Data %>% str()
```

```
tibble [2,000 × 6] (S3: tbl_df/tbl/data.frame)
 $ Sex           : Factor w/ 2 levels "F","M": 1 2 1 2 1 1 2 1 1 2 ...
 $ White         : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 2 1 2 2 ...
 $ HS_GPA        : num [1:2000] 4.14 3.3 4.3 4.29 4.2 3.86 3.75 3.32 3.86 3.91 ...
 $ SAT           : num [1:2000] 1410 1260 950 1290 1350 1350 1180 1180 970 1490 ...
 $ College_Parent: Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 2 2 2 ...
 $ Grad          : num [1:2000] 1 1 1 1 1 1 1 1 1 1 ...
```

```
Graduate_data_score$Sex <- as.factor(Graduate_data_score$Sex)
Graduate_data_score$White <- as.factor(Graduate_data_score$White)
Graduate_data_score$College_Parent <- as.factor(Graduate_data_score$College_Parent)
Graduate_data_score %>% str()
```

```
tibble [3 × 5] (S3: tbl_df/tbl/data.frame)
 $ Sex           : Factor w/ 2 levels "F","M": 1 1 2
 $ White         : Factor w/ 2 levels "0","1": 2 2 1
 $ HS_GPA        : num [1:3] 4 2.91 3.58
 $ SAT           : num [1:3] 1260 1090 1210
 $ College_Parent: Factor w/ 1 level "1": 1 1 1
```

Comments on Data Preparation:
Factors were assigned to sex, white, and college parent so they would be treated as levels as opposed to numeric or character values by the model.

Modeling

Partition Data

```
set.seed(1)
my_index <- createDataPartition(Graduate_Data$Grad, p=0.7, list=FALSE)
trainset <- Graduate_Data[my_index,]
testset <- Graduate_Data[-my_index,]

## The train and test split by depvar are very close which is good
cat("test set split by depvar \n")
```

```
test set split by depvar
```

```
prop.table(table(trainset$Grad))
```

```

        0         1 
0.3042857 0.6957143 
```

```
cat("Train Set split by depvar \n")
```

```
Train Set split by depvar
```

```
prop.table(table(testset$Grad))
```

```
        0         1 
0.3166667 0.6833333 
```

Comments on Partitioning:
Before modeling the dataset is partitioned into a 70/30 split so the data can be trained then tested. A set.seed of 1 was also given for reproducibility of the model results. The proportion of the dependent variable was very close between the trainset and testset.

Grow Full Tree

```
set.seed(1)
full_tree <- rpart(as.factor(Grad) ~., data=trainset, method="class", control=
         rpart.control(minsplit=1, minbucket=1, cp=0, maxdepth=30))

cat("Full Tree cp table\n")
```

```
Full Tree cp table
```

```
print(full_tree$cptable, digits = 3)
```

```
         CP nsplit rel error xerror   xstd
1  0.044601      0    1.0000  1.000 0.0404
2  0.008216      2    0.9108  0.920 0.0394
3  0.007042      5    0.8850  0.969 0.0401
4  0.004695     11    0.8404  0.972 0.0401
5  0.003912     20    0.7981  0.986 0.0402
6  0.003521     23    0.7864  1.016 0.0406
7  0.003130     45    0.6995  1.026 0.0407
8  0.002347     50    0.6831  1.031 0.0407
9  0.001878    112    0.5352  1.075 0.0412
10 0.001761    117    0.5258  1.059 0.0410
11 0.001565    165    0.4085  1.075 0.0412
12 0.001408    205    0.3451  1.110 0.0415
13 0.001174    210    0.3380  1.174 0.0421
14 0.000939    357    0.1596  1.195 0.0422
15 0.000782    407    0.1033  1.223 0.0425
16 0.000587    453    0.0657  1.228 0.0425
17 0.000000    481    0.0446  1.235 0.0425
```

```
cat("Unweighted Variable Importance\n")
```

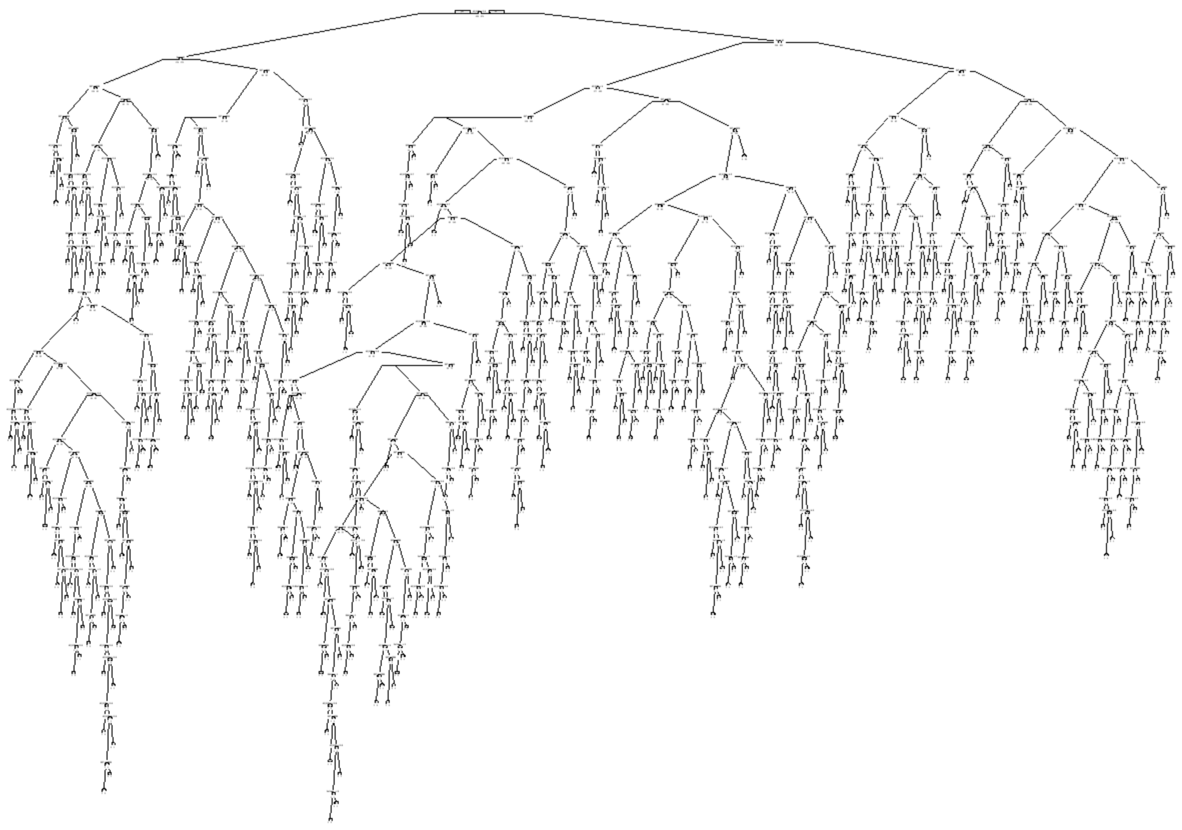
```
Unweighted Variable Importance
```

```
print(caret::varImp(full_tree))
```

```
                 Overall
College_Parent 106.78949
```

```
HS_GPA          387.29607
SAT             389.82908
Sex              40.39708
White           100.62474
```

```
## Print full tree
prp(full_tree,
    type = 1,
    extra = 1,
    under = TRUE)
```



```
## Confusion Matrix
uf_predicted_class <- predict(full_tree, testset, type = "class")
pred <- factor(as.character(uf_predicted_class), levels = c("0","1"))
ref <- factor(as.character(testset$Grad), levels = c("0","1"))
uf_conf_matrix <- caret::confusionMatrix(pred, ref, positive="0") ## The positive value is 0
         because the business problem is interested in students that did not graduate in four
         years
uf_conf_matrix
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction   0   1
         0  68 118
         1 122 292

               Accuracy : 0.6
                 95% CI : (0.5596, 0.6395)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.0705

 Mcnemar's Test P-Value : 0.8465

            Sensitivity : 0.3579
            Specificity : 0.7122
         Pos Pred Value : 0.3656
         Neg Pred Value : 0.7053
             Prevalence : 0.3167
         Detection Rate : 0.1133
   Detection Prevalence : 0.3100
      Balanced Accuracy : 0.5350

       'Positive' Class : 0
```

```r
## F1_Score
uf_F1 <- with(as.list(uf_conf_matrix$byClass),
                 (2*(`Pos Pred Value` * Sensitivity)) / (`Pos Pred Value` + Sensitivity))
cat("F1 Score: ", uf_F1, "\n")
```

```
F1 Score:  0.3617021
```

Comments on Full Tree:
The first model made was a full classification tree. The full tree was printed along with a cptable and variable importance. The CP table helps decide how much to prune the full classification tree. The table shows model complexity along with error rate. The best pruned tree is usually the CP with the lowest xerror. Variable importance helps in determining which variables are important, in this model SAT, HS GPA, and College Parent are important to the dependent variable. A confusion matrix and F1 score was printed to help determine how good the model is.

Confusion Matrix and F1 Score comments
Note: The positive class is 0
- Accuracy (0.6133) – 61.33% of predictions were correct.
- 95% CI (0.5731–0.6525) – The true accuracy likely lies between 57.31% and 65.25%. - No Information Rate (0.6933) – Shows the proportion of the non-“positive” class (majority class), which makes up 69.33% of the data.
- P-Value [Acc > NIR] (1.0000) – Indicates the model’s accuracy is not significantly better than simply guessing the majority class.
- Kappa (0.107) – Shows very weak agreement between predicted and actual classes beyond random

chance.
- McNemar's Test (p = 0.4702) – No significant difference between the types of misclassifications (false positives vs. false negatives).
- Sensitivity (0.4022) – The model correctly identified 40.22% of the positive class.
- Specificity (0.7067) – The model correctly identified 70.67% of the negative class.
- Pos Pred Value (Precision, 0.3776) – Of all cases predicted as positive, 37.76% were actually positive. - Neg Pred Value (0.7277) – Of all cases predicted as negative, 72.77% were actually negative.
- Prevalence (0.3067) – The positive class makes up 30.67% of the dataset.
- Detection Rate (0.1233) – Only 12.33% of all samples were correctly identified as belonging to the positive class.
- Detection Prevalence (0.3267) – About 32.67% of cases were predicted as positive, regardless of correctness. - Balanced Accuracy (0.5545) – Averaging sensitivity and specificity, the model correctly identifies both classes only slightly better than random (55%).
- F1 Score (0.3895) – Shows the model's balance between precision and recall is quite weak, meaning it struggles to identify positive cases accurately and consistently.

It is important to note that a full tree has a tendency to overfit the dataset and can therefore produce high accuracy. It is important to prune the true and compare performance metrics.

Best Pruned Tree; Manual (For my own exploration and curiosity)

```
cat("Full Tree cp table\n")
```

```
Full Tree cp table
```

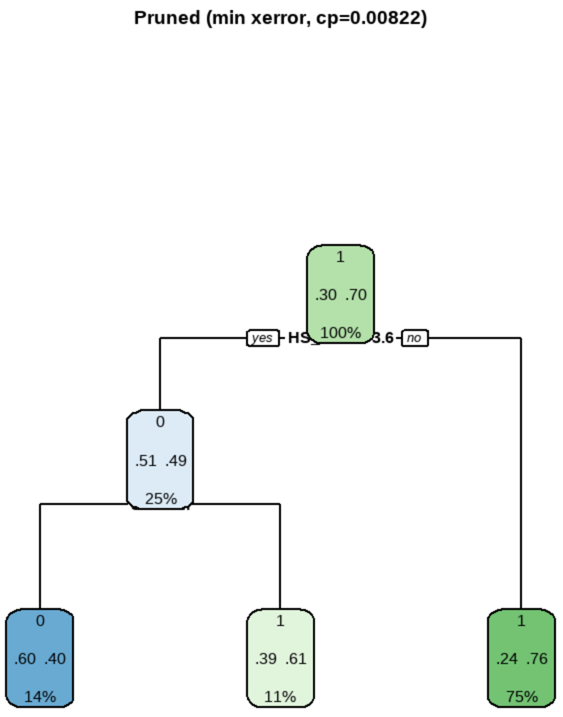
```
print(full_tree$cptable, digits = 3)
```

```
         CP nsplit rel error xerror   xstd
1  0.044601      0    1.0000  1.000 0.0404
2  0.008216      2    0.9108  0.920 0.0394
3  0.007042      5    0.8850  0.969 0.0401
4  0.004695     11    0.8404  0.972 0.0401
5  0.003912     20    0.7981  0.986 0.0402
6  0.003521     23    0.7864  1.016 0.0406
7  0.003130     45    0.6995  1.026 0.0407
8  0.002347     50    0.6831  1.031 0.0407
9  0.001878    112    0.5352  1.075 0.0412
10 0.001761    117    0.5258  1.059 0.0410
11 0.001565    165    0.4085  1.075 0.0412
12 0.001408    205    0.3451  1.110 0.0415
13 0.001174    210    0.3380  1.174 0.0421
14 0.000939    357    0.1596  1.195 0.0422
15 0.000782    407    0.1033  1.223 0.0425
16 0.000587    453    0.0657  1.228 0.0425
17 0.000000    481    0.0446  1.235 0.0425
```

```
cp_min <- full_tree$cptable[which.min(full_tree$cptable[,"xerror"]), "CP"]
tree_min  <- prune(full_tree, cp = cp_min)
print(tree_min)
```

```
n= 1400 

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 1400 426 1 (0.3042857 0.6957143)  
  2) HS_GPA< 3.645 351 173 0 (0.5071225 0.4928775)  
    4) Sex=M 200  81 0 (0.5950000 0.4050000) *
    5) Sex=F 151  59 1 (0.3907285 0.6092715) *
  3) HS_GPA>=3.645 1049 248 1 (0.2364156 0.7635844) *
```

```
rpart.plot(tree_min, type=2, extra=104, fallen.leaves=TRUE,
           main = sprintf("Pruned (min xerror, cp=%.5f)", cp_min))
```



```
pred_min <- predict(tree_min, testset, type = "class")
cm_min <- caret::confusionMatrix(pred_min, factor(testset$Grad, levels = c(0,1)), positive =
        "0")
cm_min
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  42  37
         1 148 373

               Accuracy : 0.6917
                 95% CI : (0.653, 0.7284)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 0.3481

                  Kappa : 0.1551

 Mcnemar's Test P-Value : 6.097e-16

            Sensitivity : 0.2211
            Specificity : 0.9098
         Pos Pred Value : 0.5316
         Neg Pred Value : 0.7159
             Prevalence : 0.3167
         Detection Rate : 0.0700
   Detection Prevalence : 0.1317
      Balanced Accuracy : 0.5654

       'Positive' Class : 0
```

Comments about Best Pruned Tree by lowest xerror:
I wanted to see how this version of the best pruned tree compared to the best tree caret makes.

```
myCtrl1 <- trainControl(method = "cv", number = 10)

trainset$Grad <- as.factor(trainset$Grad)

bp_tree <- train(Grad ~.,  data = trainset,
                                 method = "rpart",
                                 trControl = myCtrl1,
                                 tuneGrid = NULL)

cat("unweighted best pruned cptable\n")
```

```
unweighted best pruned cptable
```

```
bp_tree$finalModel$cptable
```

```
           CP nsplit rel error
1 0.044600939      0 1.0000000
2 0.008215962      2 0.9107981
```

```
cat("Variable Importance\n")
```

```
Variable Importance
```

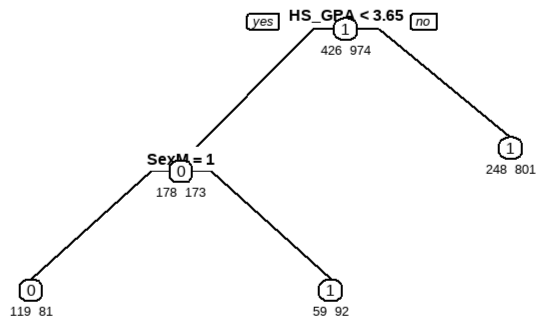
```
print(caret::varImp(bp_tree))
```

```
rpart variable importance

               Overall
HS_GPA          100.00
SexM             62.78
SAT              56.54
White1           22.42
College_Parent1   0.00
```

```
cat("\nTREE DIAGRAM WITH NODE COUNTS\n")
```

```

TREE DIAGRAM WITH NODE COUNTS
```

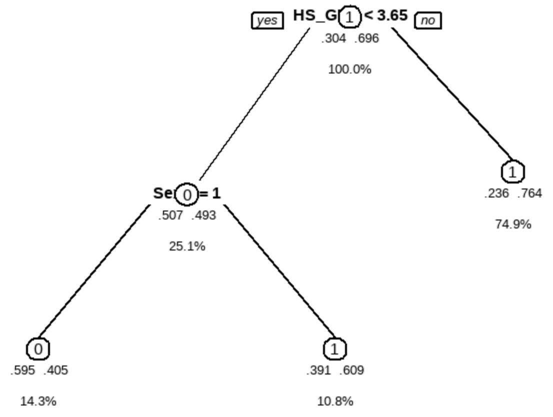
```
prp(bp_tree$finalModel, type = 1, extra = 1  , under = TRUE, digits=3)
```



```
cat("\nTREE DIAGRAM SHOWING PROBABILTIES AND NODE PROPORTION\n")
```

```
TREE DIAGRAM SHOWING PROBABILTIES AND NODE PROPORTION
```

```
prp(bp_tree$finalModel, type = 1, extra = 104, under = TRUE, digits=3)
```



```
ubp_predicted_class <- predict(bp_tree, testset, type="raw")
ubp_CM <- caret::confusionMatrix(ubp_predicted_class,
                                 as.factor(testset$Grad), positive = "0")
cat("Confusion Matrix at Default Cutoff Value\n")
```

```
Confusion Matrix at Default Cutoff Value
```

```
ubp_CM
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 42 37
         1 148 373

               Accuracy : 0.6917
                 95% CI : (0.653, 0.7284)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 0.3481

                  Kappa : 0.1551

 Mcnemar's Test P-Value : 6.097e-16

            Sensitivity : 0.2211
            Specificity : 0.9098
         Pos Pred Value : 0.5316
         Neg Pred Value : 0.7159
             Prevalence : 0.3167
         Detection Rate : 0.0700
   Detection Prevalence : 0.1317
      Balanced Accuracy : 0.5654

       'Positive' Class : 0
```

```r
ubp_F1 <- with(as.list(ubp_CM$byClass),
                 (2*(`Pos Pred Value` * Sensitivity)) / (`Pos Pred Value` + Sensitivity))
cat("F1 Score: ", ubp_F1, "\n")
```

```
F1 Score:  0.3122677
```

Comments on best pruned tree:
For the best pruned tree HS_GPA and SAT were the variables with the most importance to the dependent variable.

Note: the positive class was the 0 class

- Accuracy (0.7183) – 71.83% of predictions were correct.
- 95% CI (0.6805 – 0.7540) – The true accuracy likely lies between 68.05% and 75.40%.
- No Information Rate (0.6933) – Shows the proportion of the non-“positive” class (majority class), which makes up 69.33% of the data.
- P-Value (0.0989), The model’s accuracy is slightly higher but not statistically significant compared to simply guessing the majority class.
- Kappa (0.2317) – Shows weak agreement between predicted and actual classes beyond random chance.
- McNemar’s Test (p = 4.419e-12) – Indicates a significant difference between the types of misclassifications (false positives vs. false negatives).

- Sensitivity (0.2935) – The model correctly identified 29.35% of the positive class.
- Specificity (0.9062) – The model correctly identified 90.62% of the negative class.
- Pos Pred Value (Precision, 0.5806) – Of all cases predicted as positive, 58.06% were actually positive.
- Neg Pred Value (0.7436) – Of all cases predicted as negative, 74.36% were actually negative.
- Prevalence (0.3067) – The positive class makes up 30.67% of the dataset.
- Detection Rate (0.0900) – Only 9.00% of all samples were correctly identified as belonging to the positive class.
- Detection Prevalence (0.1550) – About 15.50% of cases were predicted as positive, regardless of correctness.
- Balanced Accuracy (0.5999) – Averaging sensitivity and specificity, the model correctly identifies both classes slightly better than random (60%).
- F1 Score (0.3899) – Shows the model's balance between precision and recall is quite weak, meaning it struggles to identify positive cases accurately and consistently.

Compared to the full tree this model has a higher accuracy and performs better overall. However this model has low sensitivity meaning the model struggles to predict the positive class.

Weighted Best Tree

```
class_counts <- table(trainset$Grad)

cat("Class Counts (n) for Admitted from the Training Dataset")
```

```
Class Counts (n) for Admitted from the Training Dataset
```

```
class_counts
```

```

  0   1 
426 974 
```

```
cat("\nClass Total:", sum(class_counts), "\n")
```

```

Class Total: 1400
```

```
wts <- ifelse(trainset$Grad == 1,
       (0.5 * sum(class_counts)) / class_counts[2],
       (0.5 * sum(class_counts)) / class_counts[1])
```

```
set.seed(1)
w_bp_tree <- train(as.factor(Grad)~., data = trainset, method = "rpart", trControl = myCtrl1,
         tuneGrid = NULL, weights = wts)

cat("Weighted Best Pruned Tree cp Table\n")
```

```
Weighted Best Pruned Tree cp Table
```

```
w_bp_tree$finalModel$cptable
```

```
          CP nsplit rel error
1 0.24271915      0 1.0000000
2 0.04460094      1 0.7572809
```

```
cat("\nWeighted Variable Importance\n")
```

```

Weighted Variable Importance
```

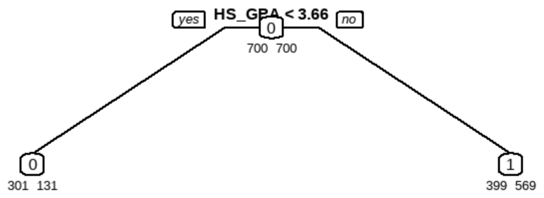
```
print(caret::varImp(w_bp_tree))
```

```
rpart variable importance

                Overall
HS_GPA           100.00
SexM              60.57
SAT               50.14
White1            27.46
College_Parent1    0.00
```

```
cat("\nTREE DIAGRAM WITH NODE COUNTS\n")
```

```

TREE DIAGRAM WITH NODE COUNTS
```

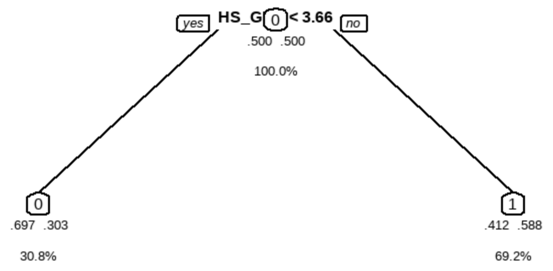
```
prp(w_bp_tree$finalModel, type = 1, extra = 1  , under = TRUE, digits=3)
```



```
cat("\nTREE DIAGRAM SHOWING PROBABILTIES AND NODE PROPORTION\n")
```

```
TREE DIAGRAM SHOWING PROBABILTIES AND NODE PROPORTION
```

```
prp(w_bp_tree$finalModel, type = 1, extra = 104, under = TRUE, digits=3)
```



```r
wb_predicted_class <- predict(w_bp_tree, testset, type = "raw")
wb_CM <- caret::confusionMatrix(wb_predicted_class, as.factor(testset$Grad), positive = "0")
cat("\nCONFUSION MATRIX AT DEFAULT CUTOFF VALUE\n")
```

```
CONFUSION MATRIX AT DEFAULT CUTOFF VALUE
```

```r
wb_CM
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  77  71
         1 113 339

               Accuracy : 0.6933
                 95% CI : (0.6547, 0.73)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 0.316142

                  Kappa : 0.2467
```

```
 Mcnemar's Test P-Value : 0.002506

            Sensitivity : 0.4053
            Specificity : 0.8268
         Pos Pred Value : 0.5203
         Neg Pred Value : 0.7500
             Prevalence : 0.3167
         Detection Rate : 0.1283
   Detection Prevalence : 0.2467
      Balanced Accuracy : 0.6160

       'Positive' Class : 0
```

```
wb_F1 <- with(as.list(wb_CM$byClass),
                 (2*(`Pos Pred Value` * Sensitivity)) / (`Pos Pred Value` + Sensitivity))
cat("F1 Score: ", wb_F1, "\n")
```

```
F1 Score:  0.4556213
```

Comments on best weighted pruned tree:
For the best weighted pruned tree HS_GPA and SAT still remain the variables with the most importance to the dependent variable.

Note: The positive class was the 0 class

- Accuracy (0.6267) – 62.67% of predictions were correct.
- 95% CI (0.5866 – 0.6655) – The true accuracy likely lies between 58.66% and 66.55%.
- No Information Rate (0.6933) – Shows the proportion of the non-“positive” class (majority class), which makes up 69.33% of the data.
- P-Value (0.9998) – The model’s accuracy is not significantly better than simply guessing the majority class.
- Kappa (0.2254) – Shows weak agreement between predicted and actual classes beyond random chance.
- McNemar’s Test (p = 6.139e-09) – Indicates a significant difference between the types of misclassifications (false positives vs. false negatives).
- Sensitivity (0.6304) – The model correctly identified 63.04% of the positive class.
- Specificity (0.6250) – The model correctly identified 62.50% of the negative class.
- Pos Pred Value (Precision, 0.4265) – Of all cases predicted as positive, 42.65% were actually positive.
- Neg Pred Value (0.7927) – Of all cases predicted as negative, 79.27% were actually negative.

- Prevalence (0.3067) – The positive class makes up 30.67% of the dataset.
- Detection Rate (0.1933) – 19.33% of all samples were correctly identified as belonging to the positive class.
- Detection Prevalence (0.4533) – About 45.33% of cases were predicted as positive, regardless of correctness.
- Balanced Accuracy (0.6277) – Averaging sensitivity and specificity, the model correctly identifies both classes about 62.8% of the time, showing modest overall balance.
- F1 Score (0.5088) – Shows the model's balance between precision and recall is moderate, meaning it identifies positive cases somewhat consistently but still with room for improvement.

Select Final Model

```
extract_metrics <- function(cm, f1) {
  c(Accuracy           = unname(cm$overall["Accuracy"]),
    Kappa              = unname(cm$overall["Kappa"]),
    Sensitivity        = unname(cm$byClass["Sensitivity"]),
    Specificity        = unname(cm$byClass["Specificity"]),
    `Pos Pred Value`   = unname(cm$byClass["Pos Pred Value"]),
    Prevalence         = unname(cm$byClass["Prevalence"]),
    `Detection Rate`   = unname(cm$byClass["Detection Rate"]),
    `Balanced Accuracy`= unname(cm$byClass["Balanced Accuracy"]),
    F1                 = f1)}

metrics_table <- data.frame(
  Full_Tree     = extract_metrics(uf_conf_matrix, uf_F1),
  Unweighted_Pruned = extract_metrics(ubp_CM, ubp_F1),
  Weighted_Pruned   = extract_metrics(wb_CM, wb_F1))

metrics_table <- tibble::rownames_to_column(metrics_table, var = "Metric")
knitr::kable(metrics_table, digits = 3, caption = "MODEL PERFORMANCE COMPARISON")
```

MODEL PERFORMANCE COMPARISON

Metric	Full_Tree	Unweighted_Pruned	Weighted_Pruned
Accuracy	0.600	0.692	0.693
Kappa	0.070	0.155	0.247
Sensitivity	0.358	0.221	0.405
Specificity	0.712	0.910	0.827
Pos Pred Value	0.366	0.532	0.520
Prevalence	0.317	0.317	0.317
Detection Rate	0.113	0.070	0.128

Metric	Full_Tree	Unweighted_Pruned	Weighted_Pruned
Balanced Accuracy	0.535	0.565	0.616
F1	0.362	0.312	0.456

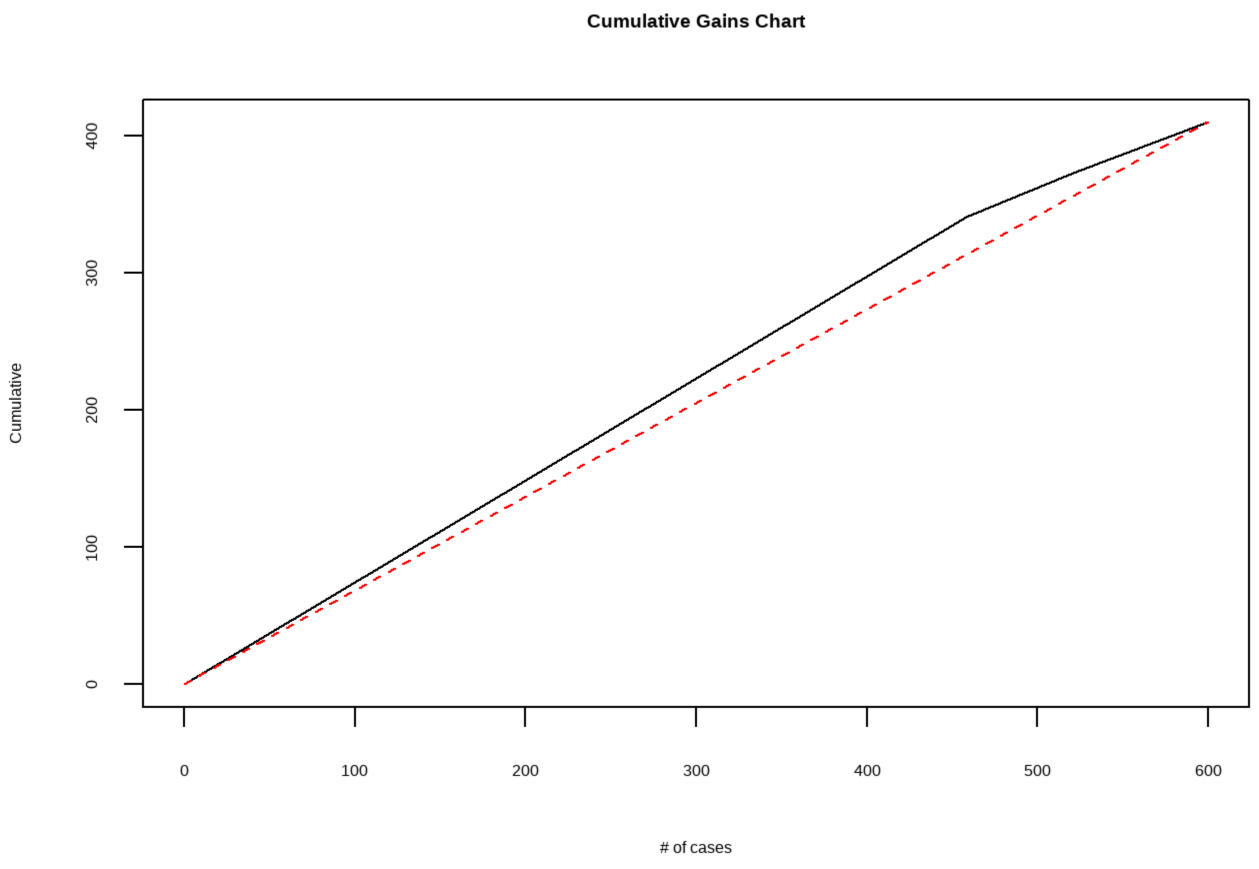
Model of selection is the **Weighted Pruned** Due to its balance of sensitivity and specificity along with having the highest balanced accuracy and F1 score among the models.

Evaluation

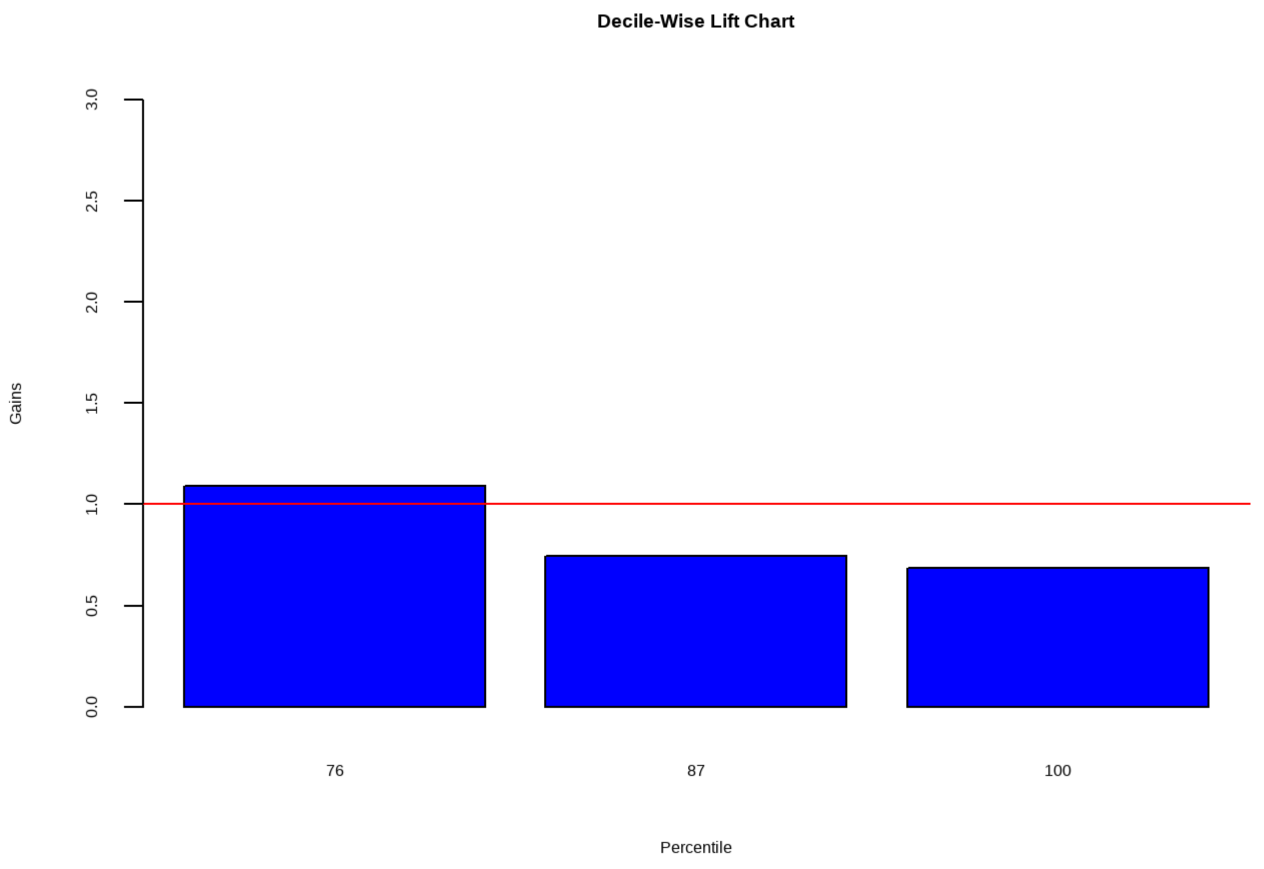
```
# convert the depvar back to numeric to plot
testset$Grad <- as.numeric(as.character(testset$Grad))

predicted_prob <-  predict(bp_tree, testset, type = "prob")

# gains table
gains_table <- gains(testset$Grad, predicted_prob[,2])
gains_table
```

```
Depth                            Cume   Cume Pct                      Mean
 of             Cume     Mean    Mean   of Total    Lift    Cume     Model
File     N        N      Resp    Resp       Resp   Index    Lift     Score
-------------------------------------------------------------------------
  76   458      458      0.74    0.74      83.2%     109     109      0.76
  87    63      521      0.51    0.72      91.0%      74     105      0.61
 100    79      600      0.47    0.68     100.0%      69     100      0.40
```

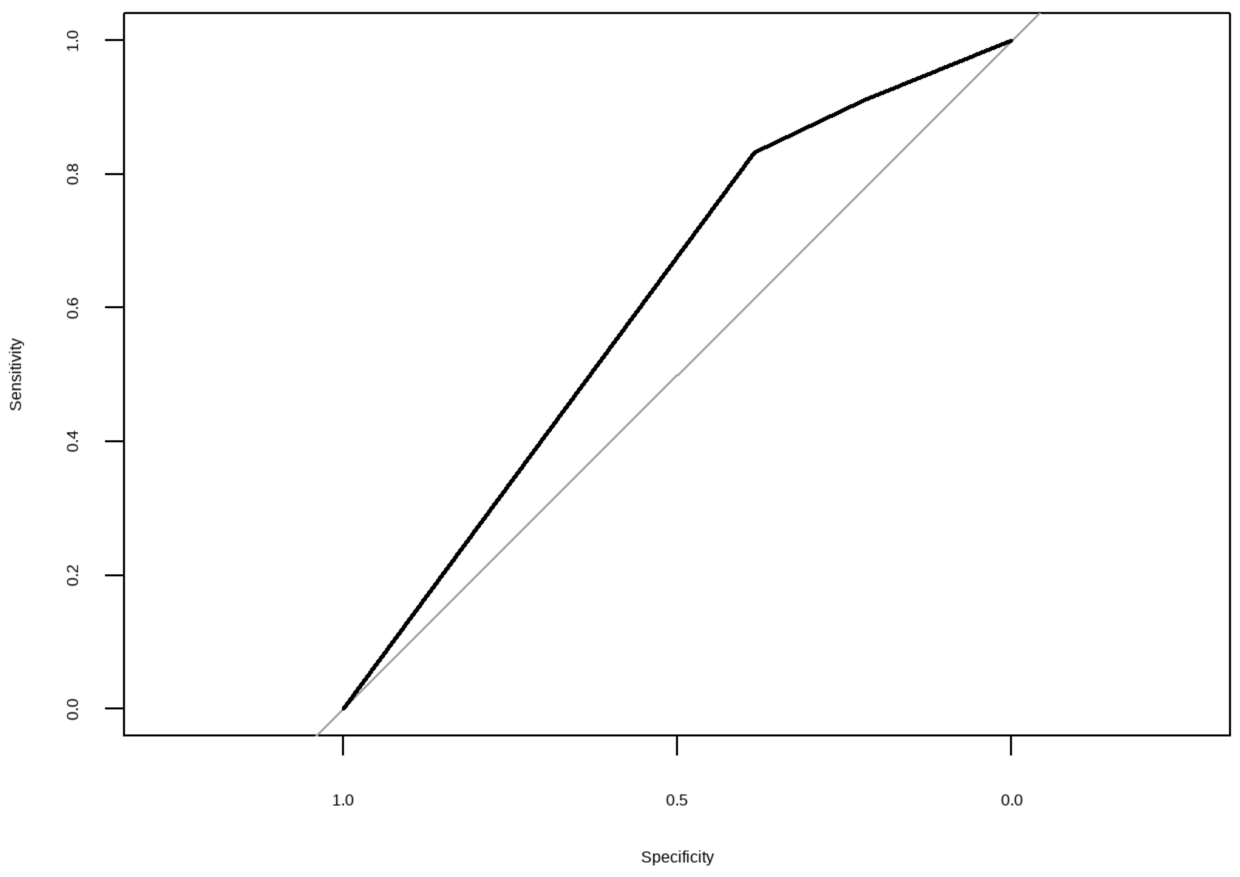
```
# cumulative gains chart
plot(c(0, gains_table$cume.pct.of.total*sum(testset$Grad)) ~ c(0, gains_table$cume.obs), xlab =
         '# of cases', ylab = "Cumulative", type = "l", main="Cumulative Gains Chart")
lines(c(0, sum(testset$Grad))~c(0, dim(testset)[1]), col="red", lty=2)
```



```
#Decile-Wise Lift Chart
barplot(gains_table$mean.resp/mean(testset$Grad), names.arg=gains_table$depth,
        xlab="Percentile", ylab="Gains", ylim=c(0,3),
                col="blue", main="Decile-Wise Lift Chart")
abline(h=c(1),col="red")
```



```
#ROC with AUC
roc_object <- roc(testset$Grad, predicted_prob[,2])
plot.roc(roc_object)
```



```
auc(roc_object)
```

```
Area under the curve: 0.6092
```

Comments on Evaluation:
The gains table shows how well the model ranks observations by predicted probability of the positive class. It shows that most of the predictive power is in the top of predicted probabilities. The Cumulative lift chart shows our model does perform better than random chance as the black line is above the red dashed line. The Decile Wise lift chart shows that the first decile is above 1 which shows the top segments of data contain proportionally more positive cases. The AUC value shows that the model has discrimination power.

Deployment

```
score_data_prob <- as.data.frame(predict(w_bp_tree, Graduate_data_score, type="prob"))

scored_opt <- cbind(Graduate_data_score, score_data_prob)
knitr::kable(scored_opt, align = "c")
```

Sex	White	HS_GPA	SAT	College_Parent	0	1
F	1	4.00	1260	1	0.4122848	0.5877152

Sex	White	HS_GPA	SAT	College_Parent	0	1
F	1	2.91	1090	1	0.6968730	0.3031270
M	0	3.58	1210	1	0.6968730	0.3031270

Comments on Deployment:
The model on the scoring data ranked two of the three students likely to not complete college in four years. The main reason is that both of these students HS GPA was below 3.87 which in the model showed that these students likely would not complete college in four years.