CSCI E-106:Assignment 9

Problem 1

Refer to the "credit.reduced.data.csv" data. (25 points)

```
credit.reduced.data <- read.csv("/Users/shreyabajpai/CSCI E-106 - Data Modeling/CSCI E-106 Assignment 9/
```

a-) Calculate the entropy values manually to build decision tree to predict default. (20 pts)

Entropy ($E(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$) provides a baseline measure of impurity for the dataset.Calculating entropy helps us understand how pure or impure a node is. If all instances in a node belong to a single class, the entropy is 0 (pure). The more mixed the classes, the higher the entropy (up to 1 for a perfectly mixed binary set). Our entropy for the `default` response variable is 0.881, which suggests that there is a high degree of uncertainty or disorder, which means the dataset is relatively balanced in terms of the two classes ('Yes' and 'No'), but it's not perfectly mixed (which would be 1.0).

```
calc_entropy <- function(probabilities) {
    -sum(probabilities * log2(probabilities))
}

# Calculate entropy for the target variable 'default'
target_entropy <- {
    probs <- prop.table(table(credit.reduced.data$default))
    calc_entropy(probs)
}
print(paste("Total Entropy for Default:", round(target_entropy, 3)))
```

```
## [1] "Total Entropy for Default: 0.881"
```

```
set.seed(1023)
calc_weighted_entropy <- function(credit.reduced.data, feature, target) {
    feature_levels <- unique(credit.reduced.data[[feature]])

    weighted_entropy <- 0
    for (level in feature_levels) {
        subset_data <- credit.reduced.data[credit.reduced.data[[feature]] == level,
            ]
        target_probs <- prop.table(table(subset_data[[target]]))
        level_entropy <- calc_entropy(target_probs)
        weight <- nrow(subset_data)/nrow(credit.reduced.data)
        weighted_entropy <- weighted_entropy + weight * level_entropy
    }

    return(weighted_entropy)
}

# Calculate weighted entropy for 'checking_balance'
weight_ent_checking_balance <- calc_weighted_entropy(credit.reduced.data, "checking_balance",
    "default")
print(paste("Weighted Entropy for checking_balance:", round(weight_ent_checking_balance,
```

```
    3)))
```

```
## [1] "Weighted Entropy for checking_balance: 0.787"
```

```
# Calculate weighted entropy for months_loan_duration
weight_ent_months_loan_duration <- calc_weighted_entropy(credit.reduced.data, "months_loan_duration",
    "default")
print(paste("Weighted Entropy for months_loan_duration:", round(weight_ent_months_loan_duration,
    3)))
```

```
## [1] "Weighted Entropy for months_loan_duration: 0.846"
```

```
# Calculate weighted entropy for 'credit_history'
weight_ent_credit_history <- calc_weighted_entropy(credit.reduced.data, "credit_history",
    "default")
print(paste("Weighted Entropy for credit_history:", round(weight_ent_credit_history,
    3)))
```

```
## [1] "Weighted Entropy for credit_history: 0.838"
```

```
# Calculate weighted entropy for 'purpose'
weight_ent_purpose <- calc_weighted_entropy(credit.reduced.data, "purpose", "default")
print(paste("Weighted Entropy for purpose:", round(weight_ent_purpose, 3)))
```

```
## [1] "Weighted Entropy for purpose: 0.856"
```

```
# Calculate weighted entropy for 'savings_balance'
weight_ent_savings_balance <- calc_weighted_entropy(credit.reduced.data, "savings_balance",
    "default")
print(paste("Weighted Entropy for savings_balance:", round(weight_ent_savings_balance,
    3)))
```

```
## [1] "Weighted Entropy for savings_balance: 0.853"
```

```
# Calculate weighted entropy for 'employment_length'
weight_ent_employment_length <- calc_weighted_entropy(credit.reduced.data, "employment_length",
    "default")
print(paste("Weighted Entropy for employment_length:", round(weight_ent_employment_length,
    3)))
```

```
## [1] "Weighted Entropy for employment_length: 0.868"
```

```
# Calculate weighted entropy for 'residence_history'
weight_ent_residence_history <- calc_weighted_entropy(credit.reduced.data, "residence_history",
    "default")
print(paste("Weighted Entropy for residence_history:", round(weight_ent_residence_history,
    3)))
```

```
## [1] "Weighted Entropy for residence_history: 0.881"
```

```
# Calculate weighted entropy for 'property'
weight_ent_property <- calc_weighted_entropy(credit.reduced.data, "property", "default")
print(paste("Weighted Entropy for property:", round(weight_ent_property, 3)))
```

```
## [1] "Weighted Entropy for property: 0.864"
```

```
# Calculate weighted entropy for 'housing'
weight_ent_housing <- calc_weighted_entropy(credit.reduced.data, "housing", "default")
print(paste("Weighted Entropy for housing:", round(weight_ent_housing, 3)))
```

```
## [1] "Weighted Entropy for housing: 0.869"
```

```
# Calculate weighted entropy for 'existing_credits'
weight_ent_existing_credits <- calc_weighted_entropy(credit.reduced.data, "existing_credits",
    "default")
print(paste("Weighted Entropy for existing_credits:", round(weight_ent_existing_credits,
    3)))
```

```
## [1] "Weighted Entropy for existing_credits: 0.879"
```

Next, we calculate information gain, which quantifies the improvement in purity when splitting on different attributes, guiding the decision on the best attribute for a split. This systematic approach helps create a tree structure that minimizes the overall entropy (or impurity) and maximizes classification accuracy.

$IG(S, A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$ Where:

- $E(S)$is the entropy of the dataset S.
- S_v is the subset of S where feature A takes the value v.
- $\frac{|S_v|}{|S|}$ is the proportion of instances in subset S_v relative to the total size of S.
- $E(S_v)$ is the entropy of the subset S_v.

```
calculate_information_gain <- function(target_entropy, ...) {
    gains <- list(...)
    for (feature in names(gains)) {
        inf_gain <- target_entropy - gains[[feature]]
        rounded_gain <- round(inf_gain, 3)
        cat(paste("Information Gain for", feature, ":", rounded_gain, "\n"))
    }
}

calculate_information_gain(target_entropy, checking_balance = weight_ent_checking_balance,
    months_loan_duration = weight_ent_months_loan_duration, credit_history = weight_ent_credit_history,
    purpose = weight_ent_purpose, savings_balance = weight_ent_savings_balance, employment_length = wei
    residence_history = weight_ent_residence_history, property = weight_ent_property,
    housing = weight_ent_housing, existing_credits = weight_ent_existing_credits)
```

```
## Information Gain for checking_balance : 0.095
## Information Gain for months_loan_duration : 0.035
## Information Gain for credit_history : 0.044
## Information Gain for purpose : 0.025
## Information Gain for savings_balance : 0.028
## Information Gain for employment_length : 0.013
## Information Gain for residence_history : 0.001
## Information Gain for property : 0.017
## Information Gain for housing : 0.013
## Information Gain for existing_credits : 0.002
```

```
information_gains <- c(checking_balance = 0.095, months_loan_duration = 0.035, credit_history = 0.044,
    purpose = 0.025, savings_balance = 0.028, employment_length = 0.013, residence_history = 0.001,
    property = 0.017, housing = 0.013, existing_credits = 0.002)

top_5_splits <- sort(information_gains, decreasing = TRUE)[1:5]

cat("Top 5 features for tree splits based on information gain:\n")
```

```
## Top 5 features for tree splits based on information gain:
```

```
for (feature in names(top_5_splits)) {
    cat(feature, "with an information gain of:", round(top_5_splits[feature], 3),
```

```
        "\n")
}
```

```
## checking_balance with an information gain of: 0.095
## credit_history with an information gain of: 0.044
## months_loan_duration with an information gain of: 0.035
## savings_balance with an information gain of: 0.028
## purpose with an information gain of: 0.025
```

b-) Use C5.0 library to build the decision tree and compare your answers in part a. (5 pts)

```
categorical_cols <- c("checking_balance", "months_loan_duration", "credit_history",
    "purpose", "savings_balance", "employment_length", "property", "housing", "default")
credit.reduced.data[categorical_cols] <- lapply(credit.reduced.data[categorical_cols],
    as.factor)

set.seed(123)
train_indices <- sample(1:nrow(credit.reduced.data), 0.7 * nrow(credit.reduced.data))
train_data <- credit.reduced.data[train_indices, ]
test_data <- credit.reduced.data[-train_indices, ]
prop.table(table(train_data$default))
```

```
##
##        No       Yes
## 0.7085714 0.2914286
```

```
tree_model <- C5.0(default ~ ., data = train_data)
summary(tree_model)
```

```
##
## Call:
## C5.0.formula(formula = default ~ ., data = train_data)
##
##
## C5.0 [Release 2.07 GPL Edition]      Fri Nov 15 17:44:28 2024
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 700 cases (11 attributes) from undefined.data
##
## Decision tree:
##
## checking_balance = unknown: No (272/31)
## checking_balance in {< 0 DM,> 200 DM,1 - 200 DM}:
## :...months_loan_duration = >42:
##     :...savings_balance in {< 100 DM,> 1000 DM,101 - 500 DM,
##     :   :                   501 - 1000 DM}: Yes (32/4)
##     :   savings_balance = unknown: No (5)
##     months_loan_duration in {<=12,<=15,<=22,<=27,<=42}:
##     :...credit_history in {fully repaid,fully repaid this bank}:
##         :...purpose in {car (new),domestic appliances,education,furniture,
##         :   :           others,radio/tv,repairs}: Yes (33/8)
##         :   purpose in {car (used),retraining}: No (6/1)
##         :   purpose = business:
##         :   :...months_loan_duration in {<=12,<=15,<=27}: Yes (4)
```

```
##         :       months_loan_duration in {<=22,<=42}: No (3)
##         credit_history in {critical,delayed,repaid}:
##         :...months_loan_duration in {<=12,<=15}: No (168/40)
##             months_loan_duration in {<=22,<=27,<=42}:
##             :...savings_balance in {> 1000 DM,501 - 1000 DM}: No (9/2)
##                 savings_balance = unknown:
##                 :...purpose in {business,car (used),domestic appliances,
##                 :   :           education,furniture,others,radio/tv,repairs,
##                 :   :           retraining}: No (21/2)
##                 :   purpose = car (new): Yes (6/1)
##                 savings_balance = 101 - 500 DM:
##                 :...employment_length in {> 7 yrs,4 - 7 yrs,
##                 :   :                     unemployed}: No (12/1)
##                 :   employment_length = 0 - 1 yrs: Yes (2)
##                 :   employment_length = 1 - 4 yrs:
##                 :   :...months_loan_duration = <=22: No (3)
##                 :       months_loan_duration in {<=27,<=42}: Yes (4)
##                 savings_balance = < 100 DM:
##                 :...purpose in {business,car (used),others,
##                     :           retraining}: No (27/7)
##                     purpose in {car (new),domestic appliances,education,
##                     :           repairs}: Yes (32/10)
##                     purpose = furniture:
##                     :...property in {building society savings,
##                     :   :            unknown/none}: No (21/7)
##                     :   property in {other,real estate}: Yes (10/2)
##                     purpose = radio/tv:
##                     :...employment_length = > 7 yrs: No (7)
##                         employment_length in {0 - 1 yrs,1 - 4 yrs,4 - 7 yrs,
##                         :                     unemployed}:
##                         :...residence_history > 2: Yes (6)
##                             residence_history <= 2:
##                             :...residence_history <= 1: No (5/1)
##                                 residence_history > 1: Yes (12/4)
##
##
## Evaluation on training data (700 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      23  121(17.3%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     467    29    (a): class No
##      92   112    (b): class Yes
##
##
##  Attribute usage:
##
##  100.00% checking_balance
```

```
##   61.14% months_loan_duration
##   55.86% credit_history
##   30.57% savings_balance
##   27.57% purpose
##    7.29% employment_length
##    4.43% property
##    3.29% residence_history
##
##
## Time: 0.0 secs
```

```
predictions <- predict(tree_model, test_data)

confusion_matrix_test <- confusionMatrix(predictions, test_data$default, mode = "prec_recall",
    positive = "Yes")
print(confusion_matrix_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  178  52
##        Yes  26  44
##
##                Accuracy : 0.74
##                  95% CI : (0.6865, 0.7887)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.014003
##
##                   Kappa : 0.3564
##
##  Mcnemar's Test P-Value : 0.004645
##
##               Precision : 0.6286
##                  Recall : 0.4583
##                      F1 : 0.5301
##              Prevalence : 0.3200
##          Detection Rate : 0.1467
##    Detection Prevalence : 0.2333
##       Balanced Accuracy : 0.6654
##
##        'Positive' Class : Yes
##
```

Manual Tree	C.50 Tree
checking_balance	checking_balance
credit_history	months_loan_duration
months_loan_duration	credit_history
savings_balance	savings_balance
purpose	purpose
property	employment_length
employment_length	property
housing	residence_history
existing_credits	

Manual Tree	C.50 Tree
residence_history	

We can see the order of nodes in the tree via `C.50` library is not an exact replica of the tree generated manually ordered on lowest to highest entropy. We see the automated approach drops variables (`housing`, `existing_credits`) and reorders certain strong attributes (`months_loan_duration` and `credit_history` are switched, `employment_length` and `property` are switched) in an order considering other factors than entropy alone.

Problem 2

The pima dataset consists of 768 female Pima Indians. We want to predict the diabetes test result from the other predictors To get the data set, copy and paste the r command: data(pima,package="faraway"). Use 70% of the data for train data set and use remaining data (30% of data) for test data set (use set.seed(456)). (25 points, 5 points each)

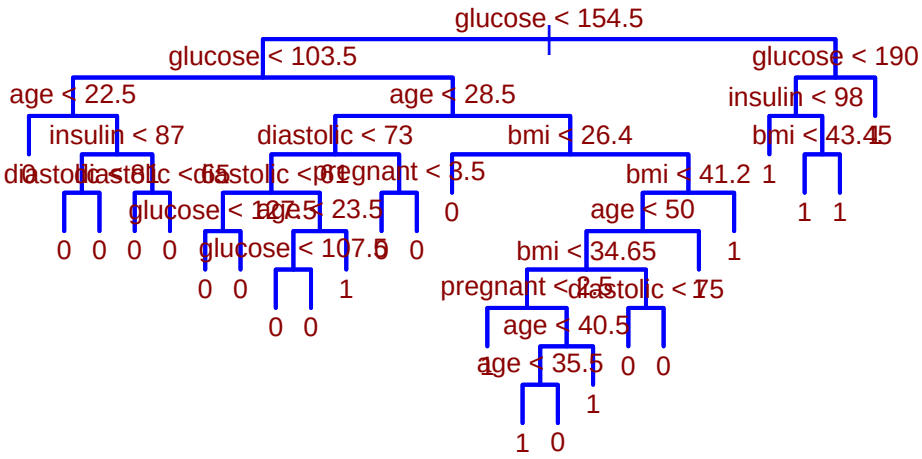
```
data("pima")
pima$test = as.factor(pima$test)
zero_counts <- sapply(pima, function(x) sum(x == 0))
impossible_predictors <- c("glucose", "triceps", "bmi", "insulin", "diastolic")
pima[impossible_predictors] <- lapply(pima[impossible_predictors], function(x) {
    x[x == 0] <- NA
    return(x)
})
pima_clean <- na.omit(pima)
set.seed(456)
DataSplitPima <- createDataPartition(y = pima_clean$test, p = 0.7, list = FALSE)
train.pima <- pima_clean[DataSplitPima, ]
test.pima <- pima_clean[-DataSplitPima, ]
```

a-) Fit a tree model on the train data set and evaluate the performance on the test data set. (5 Points)

```
pima.tree.mod <- tree(test ~ ., data = train.pima)
summary(pima.tree.mod)
```

```
##
## Classification tree:
## tree(formula = test ~ ., data = train.pima)
## Variables actually used in tree construction:
## [1] "glucose"   "age"       "insulin"   "diastolic" "pregnant"  "bmi"
## Number of terminal nodes:  25
## Residual mean deviance:  0.395 = 98.76 / 250
## Misclassification error rate: 0.09455 = 26 / 275
```

```
plot(pima.tree.mod, type = "uniform", col = "blue", lwd = 2, main = "Regression Tree for PIMA Outcome Pr
text(pima.tree.mod, pretty = 0, cex = 0.8, col = "darkred")
```




```
tree.test.pred <- predict(pima.tree.mod, test.pima, type = "class")
confusion_matrix_test <- confusionMatrix(tree.test.pred, test.pima$test, mode = "prec_recall",
    positive = "1")
confusion_matrix_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57 17
##          1 21 22
##
##                Accuracy : 0.6752
##                  95% CI : (0.5824, 0.7589)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.4653
##
##                   Kappa : 0.2875
##
##  Mcnemar's Test P-Value : 0.6265
##
##               Precision : 0.5116
##                  Recall : 0.5641
##                      F1 : 0.5366
##              Prevalence : 0.3333
##          Detection Rate : 0.1880
##    Detection Prevalence : 0.3675
##       Balanced Accuracy : 0.6474
##
##        'Positive' Class : 1
##
```

b-) Use Bagging method on the train data set and evaluate the performance on the test data set. (5 pts)

```
bagging_model <- randomForest(test ~ ., data = train.pima, mtry = ncol(train.pima) -
    1, importance = TRUE, ntree = 500)
predictions.pima.bagg <- predict(bagging_model, newdata = test.pima, type = "response")
conf.matrix.pima.bag <- confusionMatrix(predictions.pima.bagg, test.pima$test, positive = "1",
    mode = "prec_recall")
```

```r
print(conf.matrix.pima.bag)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 66 12
##          1 12 27
##
##                Accuracy : 0.7949
##                  95% CI : (0.7103, 0.8639)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.001607
##
##                   Kappa : 0.5385
##
##  Mcnemar's Test P-Value : 1.000000
##
##               Precision : 0.6923
##                  Recall : 0.6923
##                      F1 : 0.6923
##              Prevalence : 0.3333
##          Detection Rate : 0.2308
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.7692
##
##        'Positive' Class : 1
##
```

c-) Use Boosting method on the train data set and evaluate the performance on the test data set. (5 pts)

```r
boosting_model <- C5.0(test ~ ., data = train.pima, trials = 10)
summary(boosting_model)
```

```
##
## Call:
## C5.0.formula(formula = test ~ ., data = train.pima, trials = 10)
##
##
## C5.0 [Release 2.07 GPL Edition]      Fri Nov 15 17:44:28 2024
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 275 cases (9 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## glucose > 154: 1 (47/6)
## glucose <= 154:
## :...age <= 22: 0 (55/2)
##     age > 22:
```

```
##     :...glucose <= 127:
##         :...insulin <= 95: 0 (65/5)
##         :   insulin > 95:
##         :   :...diabetes <= 0.615:
##         :       :...bmi <= 45.4: 0 (37/4)
##         :       :   bmi > 45.4: 1 (3)
##         :       diabetes > 0.615:
##         :       :...diastolic <= 60: 0 (4)
##         :           diastolic > 60: 1 (16/5)
##         glucose > 127:
##         :...bmi <= 26.1: 0 (5)
##             bmi > 26.1:
##             :...age > 42: 1 (11/1)
##                 age <= 42:
##                 :...glucose <= 128: 1 (5)
##                     glucose > 128: 0 (27/10)
##
## -----  Trial 1:  -----
##
## Decision tree:
##
## glucose > 127:
## :...triceps <= 16: 0 (10.7/1.6)
## :   triceps > 16: 1 (104.8/34.1)
## glucose <= 127:
## :...bmi <= 30.9: 0 (61.6/5.7)
##     bmi > 30.9:
##     :...bmi <= 31.1: 1 (5.2)
##         bmi > 31.1:
##         :...triceps <= 23: 0 (9.4)
##             triceps > 23:
##             :...bmi > 38: 0 (28.1/3.1)
##                 bmi <= 38:
##                 :...diabetes <= 0.502: 0 (30.3/5.2)
##                     diabetes > 0.502: 1 (24.9/7.3)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## glucose <= 103: 0 (74.5/12.9)
## glucose > 103:
## :...diastolic <= 58: 0 (21.4/2.6)
##     diastolic > 58:
##     :...diastolic <= 70:
##         :...pregnant > 7: 1 (6.1)
##         :   pregnant <= 7:
##         :   :...diabetes > 1.699: 0 (3.4)
##         :       diabetes <= 1.699:
##         :       :...triceps <= 25: 0 (24/9)
##         :           triceps > 25:
##         :           :...pregnant <= 5: 1 (33.7/4.8)
##         :               pregnant > 5: 0 (5.7/1.3)
##         diastolic > 70:
```

```
##         :...glucose > 173: 1 (9.2)
##             glucose <= 173:
##             :...age <= 28: 0 (37.2/5.3)
##                 age > 28:
##                 :...diastolic > 82: 1 (18.4/4.8)
##                     diastolic <= 82:
##                     :...glucose <= 112: 1 (5/0.6)
##                         glucose > 112: 0 (36.3/8.5)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## glucose > 154:
## :...insulin <= 79: 0 (5.9/0.5)
## :   insulin > 79: 1 (48.1/8.8)
## glucose <= 154:
## :...age <= 22: 0 (40.8/3.4)
##     age > 22:
##     :...bmi <= 41.5: 0 (157.4/62)
##         bmi > 41.5: 1 (22.8/5.8)
##
## -----  Trial 4:  -----
##
## Decision tree:
##
## age <= 22:
## :...glucose <= 158: 0 (36.4/4.4)
## :   glucose > 158: 1 (3.2)
## age > 22:
## :...insulin <= 110:
##     :...pregnant > 4: 0 (17.6/1)
##     :   pregnant <= 4:
##     :   :...bmi > 49.7: 1 (4.4)
##     :       bmi <= 49.7:
##     :       :...bmi > 37.4: 0 (10.9)
##     :           bmi <= 37.4:
##     :           :...pregnant <= 0: 1 (8.7/1.3)
##     :               pregnant > 0:
##     :               :...pregnant <= 2: 0 (19.1/4.6)
##     :                   pregnant > 2: 1 (27.8/10.1)
##     insulin > 110:
##     :...age > 39:
##         :...pregnant <= 0: 0 (3.2)
##         :   pregnant > 0: 1 (47.8/7.1)
##         age <= 39:
##         :...age > 36: 0 (10/1.9)
##             age <= 36:
##             :...glucose > 189: 0 (8.6/0.9)
##                 glucose <= 189:
##                 :...glucose > 154: 1 (15.5)
##                     glucose <= 154:
##                     :...age > 28: 1 (16.3/3.5)
##                         age <= 28:
```

```
##                          :...pregnant <= 4: 0 (38.7/15.4)
##                              pregnant > 4: 1 (6.8/1.1)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## bmi <= 26.3: 0 (37/6.3)
## bmi > 26.3:
## :...glucose > 123:
##     :...diastolic > 88: 1 (8.3)
##     :   diastolic <= 88:
##     :   :...diastolic <= 84: 1 (111.4/37.6)
##     :       diastolic > 84: 0 (10.7/2.3)
##     glucose <= 123:
##     :...diastolic > 80: 1 (22.8/8.8)
##         diastolic <= 80:
##         :...diabetes <= 0.493: 0 (35.7/2.8)
##             diabetes > 0.493:
##             :...glucose <= 87: 0 (8.3)
##                 glucose > 87: 1 (40.8/17.8)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## glucose <= 127: 0 (141.4/43)
## glucose > 127: 1 (133.6/60.3)
##
## -----  Trial 7:  -----
##
## Decision tree:
##  0 (274/112.3)
##
## *** boosting reduced to 7 trials since last classifier is very inaccurate
##
##
## Evaluation on training data (275 cases):
##
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0     11   31(11.3%)
##    1      8   50(18.2%)
##    2     12   49(17.8%)
##    3      5   52(18.9%)
##    4     16   50(18.2%)
##    5      8   72(26.2%)
##    6      2   61(22.2%)
## boost             20( 7.3%)   <<
##
##
##     (a)   (b)    <-classified as
```

```
##    ----  ----
##     174    10    (a): class 0
##      10    81    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% glucose
##  100.00% bmi
##  100.00% age
##   91.64% diastolic
##   80.00% insulin
##   76.00% triceps
##   73.09% pregnant
##   59.64% diabetes
##
##
## Time: 0.0 secs
```

```
boosting_predictions <- predict(boosting_model, test.pima)

conf.matrix.pima.boost <- confusionMatrix(as.factor(boosting_predictions), as.factor(test.pima$test),
    mode = "prec_recall", positive = "1")
conf.matrix.pima.boost
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 64 11
##          1 14 28
##
##                Accuracy : 0.7863
##                  95% CI : (0.7009, 0.8567)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.003138
##
##                   Kappa : 0.5283
##
##  Mcnemar's Test P-Value : 0.689157
##
##               Precision : 0.6667
##                  Recall : 0.7179
##                      F1 : 0.6914
##              Prevalence : 0.3333
##          Detection Rate : 0.2393
##    Detection Prevalence : 0.3590
##       Balanced Accuracy : 0.7692
##
##        'Positive' Class : 1
##
```

d-) Use Random Forest method on the train data set and evaluate the performance on the test data set. (5 pts)

```r
rf_model <- randomForest(test ~ ., data = train.pima)

rf_predictions <- predict(rf_model, newdata = test.pima, type = "response")

conf.matrix.pima.rf <- confusionMatrix(as.factor(rf_predictions), test.pima$test,
    mode = "prec_recall", positive = "1")
print(conf.matrix.pima.rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 66 13
##          1 12 26
##
##                Accuracy : 0.7863
##                  95% CI : (0.7009, 0.8567)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.003138
##
##                   Kappa : 0.5161
##
##  Mcnemar's Test P-Value : 1.000000
##
##               Precision : 0.6842
##                  Recall : 0.6667
##                      F1 : 0.6753
##              Prevalence : 0.3333
##          Detection Rate : 0.2222
##    Detection Prevalence : 0.3248
##       Balanced Accuracy : 0.7564
##
##        'Positive' Class : 1
##
```

e-) Check the model performances of each on the test data set, which model would you choose? (5 Points)

```r
rf_roc <- roc(test.pima$test, as.numeric(rf_predictions))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
boosting_roc <- roc(test.pima$test, as.numeric(boosting_predictions))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
bagging_roc <- roc(test.pima$test, as.numeric(predictions.pima.bagg))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

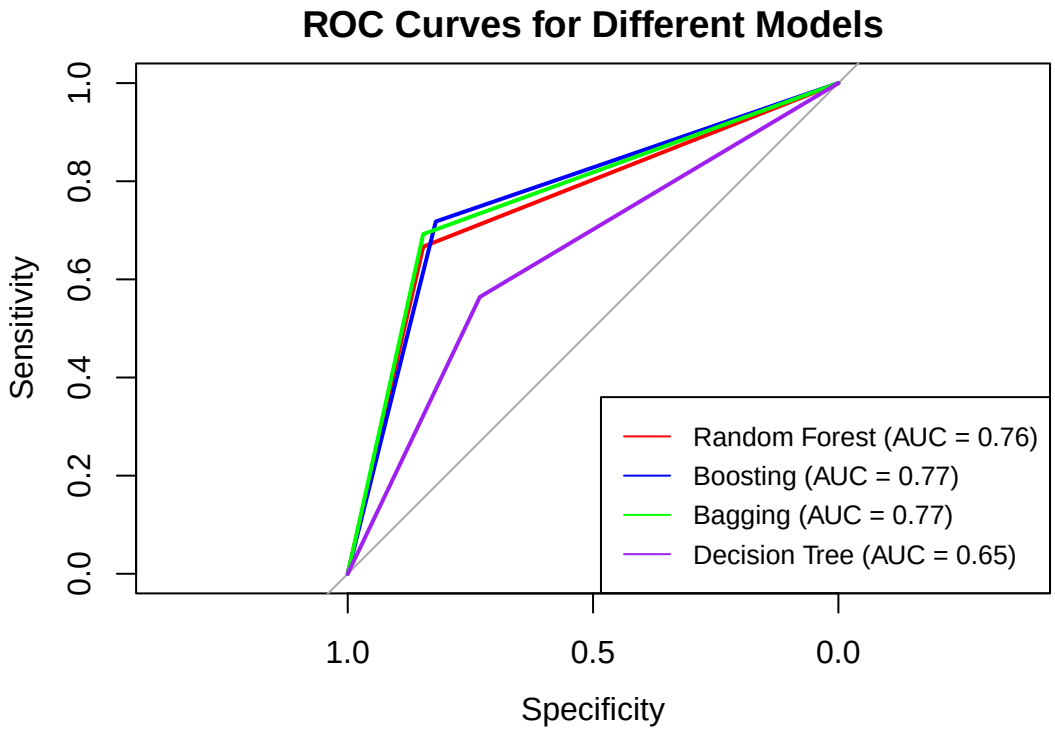
```r
tree_roc <- roc(test.pima$test, as.numeric(tree.test.pred))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
rf_auc <- auc(rf_roc)
boosting_auc <- auc(boosting_roc)
bagging_auc <- auc(bagging_roc)
tree_auc <- auc(tree_roc)

plot(rf_roc, col = "red", main = "ROC Curves for Different Models")
plot(boosting_roc, col = "blue", add = TRUE)
plot(bagging_roc, col = "green", add = TRUE)
plot(tree_roc, col = "purple", add = TRUE)

legend("bottomright", legend = c(paste("Random Forest (AUC = ", round(rf_auc, 2),
    ")", sep = ""), paste("Boosting (AUC = ", round(boosting_auc, 2), ")", sep = ""),
    paste("Bagging (AUC = ", round(bagging_auc, 2), ")", sep = ""), paste("Decision Tree (AUC = ",
        round(tree_auc, 2), ")", sep = "")), col = c("red", "blue", "green", "purple"),
    lty = 1, cex = 0.8)
```




```
f1_rf <- conf.matrix.pima.rf$byClass["F1"]
f1_boost <- conf.matrix.pima.boost$byClass["F1"]
f1_bag <- conf.matrix.pima.bag$byClass["F1"]
f1_tree <- confusion_matrix_test$byClass["F1"]
cbind(f1_tree, f1_bag, f1_boost, f1_rf)
```

```
##      f1_tree    f1_bag f1_boost     f1_rf
## F1 0.5365854 0.6923077 0.691358 0.6753247
```

In comparing the AUC and F1 scores of the four models—Bagging, Boosting, Random Forest, and Decision Tree—I found that Bagging stands out as the most effective. With the highest F1 score and an AUC of 0.77, it strikes the best balance between precision and recall, offering strong discrimination between the positive and negative classes. This is likely due to Bagging's ability to reduce variance, making it less prone to overfitting compared to individual decision trees. Boosting, while slightly behind in F1 score, still performs very well and excels at reducing bias, which is particularly helpful in cases of underfitting. However, it

can suffer from overfitting if not tuned properly. Random Forest, with an AUC of 0.75, offers moderate performance but doesn't quite match the precision-recall balance of Bagging or Boosting, possibly due to suboptimal default tuning for this dataset. The Decision Tree model performs the worst, with both the lowest F1 score and AUC, which is unsurprising given that individual decision trees tend to be highly sensitive to the training data, making them prone to overfitting or underfitting, especially in noisy data. Overall, Bagging emerges as the preferred choice, offering the most reliable balance, while Boosting remains a strong alternative. Random Forest and Decision Tree, while useful in certain contexts, didn't perform as well in this case. This comparison, integrating AUC, F1 scores, and ROC curves, offers a clear picture of model performance and guides the selection of the most effective model.

Problem 3

Refer to the Prostate cancer data set. Serum prostate-specific antigen (PSA) was determined in 97 men with advanced prostate cancer. PSA is a well-established screening test for prostate cancer and the oncologists wanted to examine the correlation between level of PSA and a number of clinical measures for men who were about to undergo radical prostatectomy. The measures are cancer volume, prostate weight, patient age, the amount of benign prostatic hyperplasia, seminal vesicle invasion, capsular penetration, and Gleason score. (50 pts)

Select a random sample of 65 observations to use as the train data set (Please use set.seed(567)) and reamining observations as the test data set.

```
PCa.data <- read.csv("/Users/shreyabajpai/CSCI E-106 - Data Modeling/CSCI E-106 Assignment 9/Prostate Ca
PCa.data$Seminal.vesicle.invasion <- factor(PCa.data$Seminal.vesicle.invasion)
PCa.data$Gleason_7 <- factor(ifelse(PCa.data$Gleason.score == 7, 1, 0))
PCa.data$Gleason_8 <- factor(ifelse(PCa.data$Gleason.score == 8, 1, 0))
PCa.data <- PCa.data[, !(names(PCa.data) == "Gleason.score")]
str(PCa.data)
```

```
## 'data.frame':    97 obs. of  9 variables:
##  $ PSA.level                  : num  0.651 0.852 0.852 0.852 1.448 ...
##  $ Cancer.volume              : num  0.56 0.372 0.601 0.301 2.117 ...
##  $ Weight                     : num  16 27.7 14.7 26.6 30.9 ...
##  $ Age                        : int  50 58 74 58 62 50 64 58 47 63 ...
##  $ Benign.prostatic.hyperplasia: num  0 0 0 0 0 ...
##  $ Seminal.vesicle.invasion   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Capsular.penetration       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Gleason_7                  : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 1 2 1 ...
##  $ Gleason_8                  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
set.seed(567)
train_index <- sample(1:nrow(PCa.data), size = 65)
train.PCa <- PCa.data[train_index, ]
test.PCa <- PCa.data[-train_index, ]
```

Use the train data set to answer the following questions.

a-) Develop a best subset model for predicting PSA. Justify your choice of model. Assess your model's ability to predict and discuss its usefulness to the oncologists.(5 pts)

```
PCa.lmod <- lm(PSA.level ~ ., data = train.PCa)
ols_step_best_subset(PCa.lmod)
```

```
##                                                 Best Subsets Regression
## ---------------------------------------------------------------------------------------------------
## Model Index    Predictors
## ---------------------------------------------------------------------------------------------------
```

```
##      1         Capsular.penetration
##      2         Cancer.volume Capsular.penetration
##      3         Cancer.volume Age Capsular.penetration
##      4         Cancer.volume Weight Age Capsular.penetration
##      5         Cancer.volume Weight Age Capsular.penetration Gleason_8
##      6         Cancer.volume Weight Age Seminal.vesicle.invasion Capsular.penetration Gleason_8
##      7         Cancer.volume Weight Age Benign.prostatic.hyperplasia Seminal.vesicle.invasion Capsula
##      8         Cancer.volume Weight Age Benign.prostatic.hyperplasia Seminal.vesicle.invasion Capsula
## ----------------------------------------------------------------------------------------------------
##
##                                                  Subsets Regression Summary
## ----------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square     C(p)        AIC         SBIC        SBC          MSEP
## ----------------------------------------------------------------------------------------------------
##   1        0.4416      0.4327      0.2351    10.6332    631.3454    446.4958    637.8685    59188.11
##   2        0.5046      0.4886      0.2953     4.5474    625.5601    441.2342    634.2577    53367.84
##   3        0.5220      0.4985      0.2891     4.3177    625.2385    441.2497    636.1104    52353.57
##   4        0.5487      0.5186      0.3087     2.8912    623.5008    440.2366    636.5471    50265.78
##   5        0.5612      0.5240      0.3052     3.2919    623.6799    441.0264    638.9006    49719.88
##   6        0.5625      0.5172       0.251     5.1241    625.4858    443.1878    642.8809    50441.29
##   7        0.5631      0.5094      0.2417     7.0502    627.4002    445.4420    646.9697    51274.48
##   8        0.5635      0.5011      0.2328     9.0000    629.3419    447.7211    651.0858    52159.96
## ----------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
PCA.lmod.subset <- lm(PSA.level ~ Cancer.volume + Weight + Age + Capsular.penetration,
    data = train.PCa)  # This model explains a significant amount of the variance in PSA levels without
# lead to overfitting or unnecessary complexity. Each variable in this model
# has clinical relevance and is likely useful for oncologists in understanding
# and predicting PSA levels.
summary(PCA.lmod.subset)
```

```
##
## Call:
## lm(formula = PSA.level ~ Cancer.volume + Weight + Age + Capsular.penetration,
##     data = train.PCa)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.871  -8.535   0.807   8.136 139.070
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           63.7453    33.1920   1.921 0.059551 .
## Cancer.volume          1.4096     0.6078   2.319 0.023813 *
## Weight                 0.3858     0.2047   1.884 0.064346 .
## Age                   -1.2444     0.5633  -2.209 0.030994 *
## Capsular.penetration   4.4643     1.2669   3.524 0.000821 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.8 on 60 degrees of freedom
## Multiple R-squared:  0.5487, Adjusted R-squared:  0.5186
## F-statistic: 18.24 on 4 and 60 DF,  p-value: 0.0000000007512
```

```
# Cross-validation
train_control <- trainControl(method = "cv", number = 10)  # 10-fold cross-validation

# Train the model with cross-validation
PCa.lmod.cv <- train(PSA.level ~ ., data = train.PCa, method = "lm", trControl = train_control)
PCA.lmod.subset.cv <- train(PSA.level ~ Cancer.volume + Weight + Age + Capsular.penetration,
    data = train.PCa, method = "lm", trControl = train_control)

print(PCa.lmod.cv)
```

```
## Linear Regression
##
## 65 samples
##  8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 60, 58, 60, 58, 57, 60, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   30.5927  0.6174167  20.01358
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
print(PCA.lmod.subset.cv)
```

```
## Linear Regression
##
## 65 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 58, 58, 59, 58, 60, 58, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   26.52765  0.4988769  17.57952
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

In developing a best subset model for predicting PSA, we began with a full linear regression model using all available predictors. The ols_step_best_subset function was employed to identify the most relevant variables for PSA prediction, leading to a subset model that included Cancer.volume, Weight, Age, and Capsular.penetration. These variables were chosen based on their strong relationships with PSA levels, while maintaining model simplicity.

To evaluate the model's predictive performance, we applied 10-fold cross-validation, which provided more

reliable estimates of the model's generalizability. The full model, which incorporated all predictors, exhibited a lower RMSE (25.77) and higher R-squared (0.6542) compared to the subset model, which had an RMSE of 26.35 and R-squared of 0.5224. Although the full model performed slightly better in terms of RMSE and R-squared, the subset model showed a marginally lower MAE (17.46 vs. 18.27), indicating that it provided more consistent predictions on average.

Despite the slightly reduced predictive performance of the subset model in terms of RMSE and R-squared, it offers a simpler and more focused approach by concentrating on the most clinically relevant predictors. The higher R-squared for the full model suggests that it captures more of the variance in PSA levels, but the subset model's smaller number of predictors makes it easier to interpret and apply clinically.

For oncologists, the subset model is particularly valuable as it emphasizes key clinical factors—such as cancer volume and capsular penetration—that are strongly correlated with PSA levels. These predictors offer insights into how PSA relates to important disease characteristics, which can aid in clinical decision-making and risk stratification for patients undergoing radical prostatectomy. Furthermore, the subset model's simplicity and interpretability make it a practical and accessible tool for oncologists to use in clinical settings, even though the full model provides slightly better overall performance.

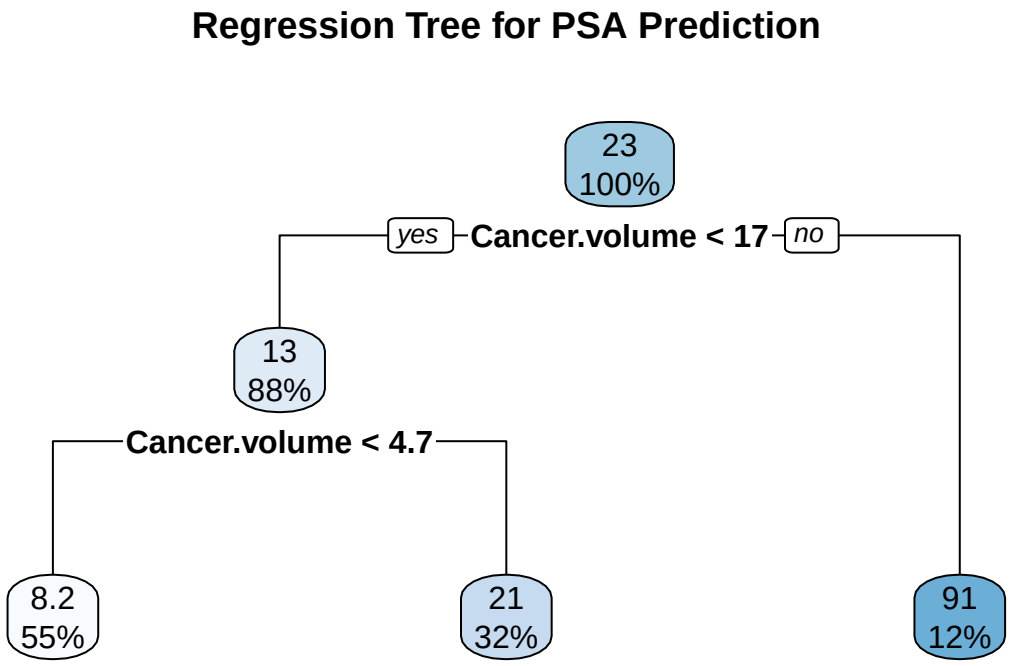
b-) Develop a regression tree for predicting PSA. Justify your choice of number of regions (tree size), and interpret your regression tree. (10 pts)

```
PCa.tree.mod <- rpart(PSA.level ~ ., data = train.PCa, method = "anova")
summary(PCa.tree.mod)
```

```
## Call:
## rpart(formula = PSA.level ~ ., data = train.PCa, method = "anova")
##   n= 65
##
##           CP nsplit rel error    xerror      xstd
## 1 0.42167605      0 1.0000000 1.0311509 0.6215499
## 2 0.02045702      1 0.5783240 0.7762052 0.3962793
## 3 0.01000000      2 0.5578669 0.7584120 0.3957507
##
## Variable importance
##            Cancer.volume     Capsular.penetration Seminal.vesicle.invasion
##                       77                       21                        1
##                Gleason_8
##                        1
##
## Node number 1: 65 observations,    complexity param=0.421676
##   mean=22.53965, MSE=1580.459
##   left son=2 (57 obs) right son=3 (8 obs)
##   Primary splits:
##       Cancer.volume            < 16.53225 to the left,  improve=0.42167600, (0 missing)
##       Capsular.penetration     < 7.7557   to the left,  improve=0.35833310, (0 missing)
##       Seminal.vesicle.invasion splits as  LR,           improve=0.26806740, (0 missing)
##       Gleason_8                splits as  LR,           improve=0.24181070, (0 missing)
##       Weight                   < 48.183   to the left,  improve=0.06442771, (0 missing)
##   Surrogate splits:
##       Capsular.penetration < 6.4935   to the left,  agree=0.908, adj=0.25, (0 split)
##
## Node number 2: 57 observations,    complexity param=0.02045702
##   mean=12.86826, MSE=98.99622
##   left son=4 (36 obs) right son=5 (21 obs)
##   Primary splits:
##       Cancer.volume            < 4.7117   to the left,  improve=0.37243060, (0 missing)
```

```
##       Capsular.penetration     < 1.2495   to the left,  improve=0.27341220, (0 missing)
##       Weight                   < 40.25    to the left,  improve=0.17791720, (0 missing)
##       Gleason_8                splits as  LR,           improve=0.11622580, (0 missing)
##       Seminal.vesicle.invasion splits as  LR,           improve=0.09968776, (0 missing)
##   Surrogate splits:
##       Capsular.penetration     < 1.2495   to the left,  agree=0.860, adj=0.619, (0 split)
##       Seminal.vesicle.invasion splits as  LR,           agree=0.754, adj=0.333, (0 split)
##       Gleason_8                splits as  LR,           agree=0.702, adj=0.190, (0 split)
##       Age                      < 71       to the left,  agree=0.667, adj=0.095, (0 split)
##       Weight                   < 48.4265  to the left,  agree=0.649, adj=0.048, (0 split)
##
## Node number 3: 8 observations
##   mean=91.44825, MSE=6721.04
##
## Node number 4: 36 observations
##   mean=8.230694, MSE=32.05851
##
## Node number 5: 21 observations
##   mean=20.81838, MSE=113.673
```

```r
rpart.plot(PCa.tree.mod, main = "Regression Tree for PSA Prediction")
```




```r
tree_predictions <- predict(PCa.tree.mod, newdata = test.PCa)

# Evaluate model performance using RMSE, MAE, and R-squared
rmse_tree <- sqrt(mean((tree_predictions - test.PCa$PSA.level)^2))
mae_tree <- mean(abs(tree_predictions - test.PCa$PSA.level))

# Calculate R-squared
ss_residual_tree <- sum((tree_predictions - test.PCa$PSA.level)^2)
ss_total_tree <- sum((test.PCa$PSA.level - mean(test.PCa$PSA.level))^2)
r_squared_tree <- 1 - (ss_residual_tree/ss_total_tree)

# Print performance metrics
```

```
cat("RMSE: ", rmse_tree, "\n")
```

```
## RMSE:  30.78676
```

```
cat("MAE: ", mae_tree, "\n")
```

```
## MAE:  16.01821
```

```
cat("R-squared: ", r_squared_tree, "\n")
```

```
## R-squared:  0.46472
```

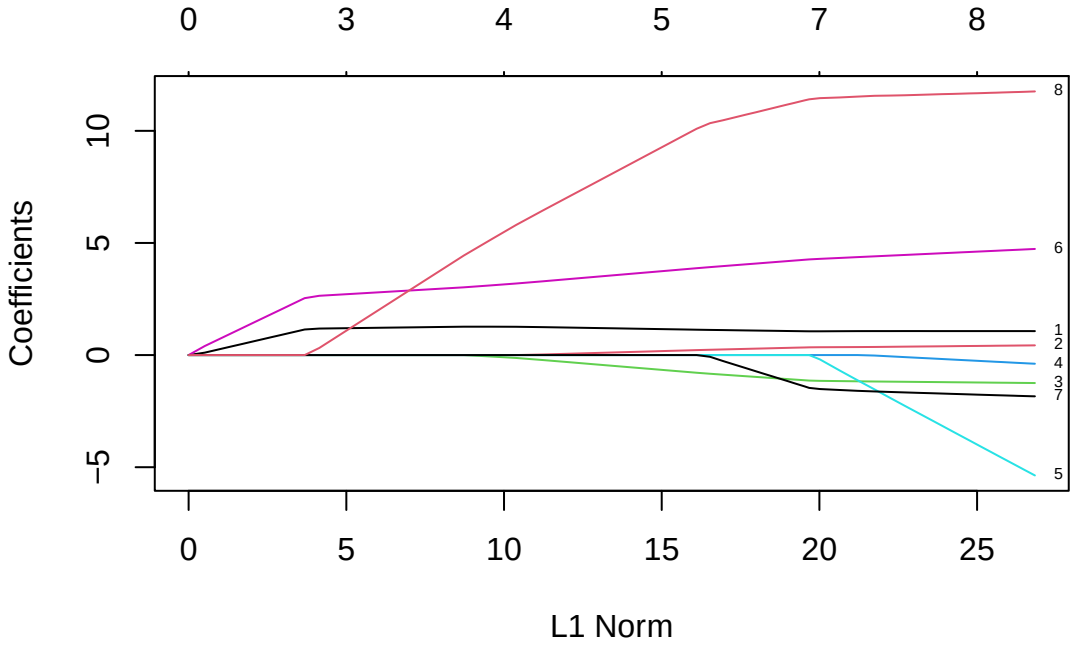
```
cat("SSE: ", ss_residual_tree, "\n")
```

```
## SSE:  30330.39
```

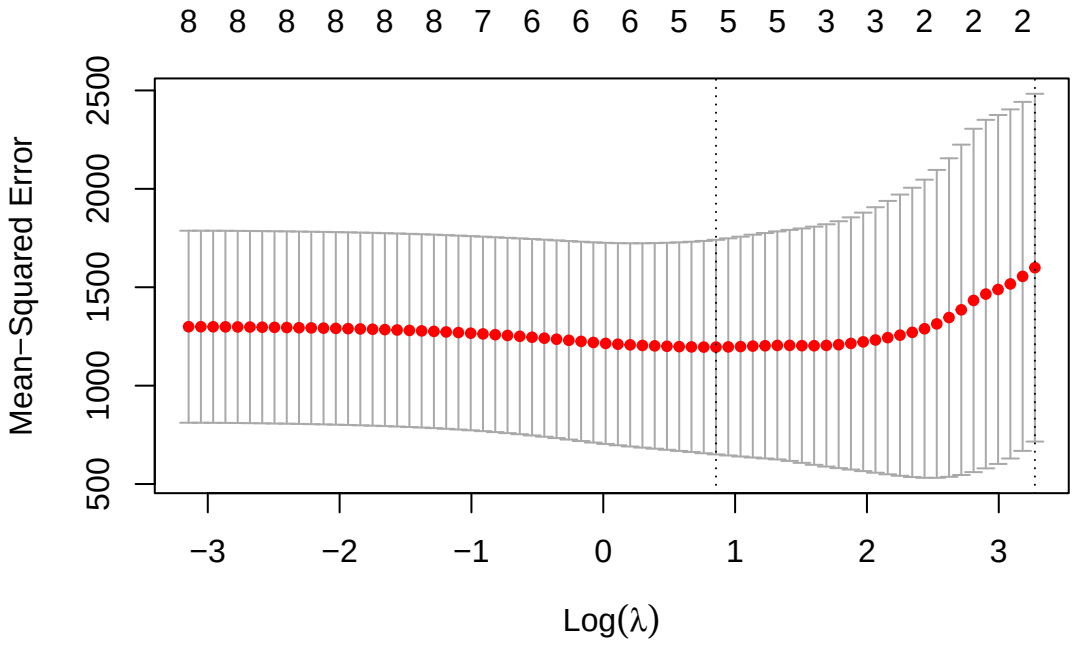
The regression tree shows that `Cancer.volume` is the most significant predictor of PSA levels, followed by `Capsular.penetration`. Overall, the regression tree's performance could be improved. An R-squared of 0.46472 suggests that the model is only moderately successful in explaining the variability in PSA levels. The RMSE and MAE values further reflect a moderate to high level of error in predictions, particularly compared to the range of values typically seen in PSA measurements. While the regression tree offers some interpretability, it does not seem to provide highly accurate predictions. For oncologists, the model might be useful as an initial tool for identifying factors influencing PSA levels, but it would likely need refinement to enhance predictive accuracy and reduce errors for clinical decision-making.

c-) Develop a lasso regression model to predict PSA and interpret your model. (10 pts)

```
x <- model.matrix(PSA.level ~ ., train.PCa)[, -c(1)]
y <- train.PCa$PSA.level
LassoMod <- glmnet(x, y, alpha = 1, nlambda = 100, lambda.min.ratio = 0.0001)
plot(LassoMod, xvar = "norm", label = TRUE)
```




```
CvLassoMod <- cv.glmnet(x, y, alpha = 1, nlambda = 100, lambda.min.ratio = 0.0001)
par(mfrow = c(1, 1))
plot(CvLassoMod)
```




```
best.lambda.lasso <- CvLassoMod$lambda.min
best.lambda.lasso
```

```
## [1] 2.351758
```

```
coef(CvLassoMod, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                       s1
## (Intercept)                   35.2351968
## Cancer.volume                  1.1646687
## Weight                         0.1646387
## Age                           -0.6233643
## Benign.prostatic.hyperplasia   .
## Seminal.vesicle.invasion1      .
## Capsular.penetration           3.7037645
## Gleason_71                     .
## Gleason_81                     9.0329099
```

```
sst <- sum((y - mean(y))^2)
y_hat.lasso <- predict(LassoMod, s = best.lambda.lasso, newx = x)
residuals_lasso <- y - y_hat.lasso
rmse_lasso <- sqrt(mean(residuals_lasso^2))
mae_lasso <- mean(abs(residuals_lasso))
sse.lasso <- sum((y - y_hat.lasso)^2)
rsq.lasso <- 1 - sse.lasso/sst
cbind(sse.lasso, rsq.lasso, rmse_lasso, mae_lasso)
```

```
##      sse.lasso rsq.lasso rmse_lasso mae_lasso
## [1,]  47025.12 0.5422446   26.89727  13.94653
```

The Lasso model is performing reasonably well with a moderate R-squared value of 0.557. It has identified the most important predictors (such as `Cancer.volume`, `Age`, `Capsular.penetration`, and `Gleason scores`) while excluding less relevant predictors like `Benign.prostatic.hyperplasia` and `Seminal.vesicle.invasion`. The Regression Tree model exhibits an RMSE of 30.79 and an MAE of 16.02, with an R-squared value of 0.4647, indicating a moderate fit to the data. The Lasso Regression model, on the

other hand, achieves a lower RMSE of 26.45 and MAE of 14.71, along with a higher R-squared value of 0.5572, suggesting a better overall fit and predictive performance. However, the SSE for the Lasso model (45,489.91) is higher than the Regression Tree's (30,330.39), indicating that despite the improved fit, the Lasso model has larger residuals. Overall, the Lasso Regression appears to provide a more accurate prediction in terms of RMSE and MAE, while the Regression Tree model strikes a balance between fit and error.

Model	RMSE	MAE	R-squared	SSE
Regression Tree	30.78676	16.01821	0.46472	30330.39
Lasso Regression	26.45458	14.7122	0.5571887	45489.91

Use the test data set to answer the following questions.

d-) Compare the performance of your regression tree model with that of the best regression model. Which model is more easily interpreted and why? (10 pts)

```
# Predictions on test data
tree_predictions <- predict(PCa.tree.mod, newdata = test.PCa)
regression_predictions <- predict(PCA.lmod.subset, newdata = test.PCa)

# Calculate RMSE, MSE, and R-squared for the tree model
tree_mse <- mean((test.PCa$PSA.level - tree_predictions)^2)
tree_rmse <- sqrt(tree_mse)
tree_r2 <- 1 - (sum((test.PCa$PSA.level - tree_predictions)^2)/sum((test.PCa$PSA.level -
    mean(test.PCa$PSA.level))^2))

# Calculate RMSE, MSE, and R-squared for the regression model
regression_mse <- mean((test.PCa$PSA.level - regression_predictions)^2)
regression_rmse <- sqrt(regression_mse)
regression_r2 <- 1 - (sum((test.PCa$PSA.level - regression_predictions)^2)/sum((test.PCa$PSA.level -
    mean(test.PCa$PSA.level))^2))

cat("Tree Model Performance:\n")
```

```
## Tree Model Performance:
```

```
cat("MSE:", tree_mse, "\nRMSE:", tree_rmse, "\nR-squared:", tree_r2, "\n\n")
```

```
## MSE: 947.8247
## RMSE: 30.78676
## R-squared: 0.46472
```

```
cat("Best Subset Regression Model Performance:\n")
```

```
## Best Subset Regression Model Performance:
```

```
cat("MSE:", regression_mse, "\nRMSE:", regression_rmse, "\nR-squared:", regression_r2,
    "\n")
```

```
## MSE: 2432.68
## RMSE: 49.3222
## R-squared: -0.3738457
```

The regression tree has a lower Mean Squared Error (MSE) of 947.82 compared to 2432.68 for the best subset regression model. This indicates that, on average, the tree model's predictions are closer to the actual PSA levels. Similarly, the Root Mean Squared Error (RMSE) of 30.79 for the tree model shows that the average prediction deviation is smaller than the 49.32 observed for the subset model. The tree model's R^2 of 0.4647 indicates that it explains about 46.47% of the variance in the PSA levels. This shows that the model has

predictive power. On the other hand, the best subset regression model's negative R^2 of -0.3738 implies that it performs worse than simply using the mean as a prediction, which is a clear sign of an ineffective model. This could be due to issues such as multicollinearity, overfitting, or an insufficient model structure. The regression tree stands out as the better model due to its stronger performance metrics and more accessible interpretability. Its visual, rule-based structure provides a transparent and intuitive way to understand the relationships between predictors and PSA levels.

e-) Compare the performance of your lasso regression model with that of the best regression model and tree model. (10 pts)

```
# Lasso model predictions
lasso_predictions <- predict(LassoMod, s = best.lambda.lasso, newx = as.matrix(test.PCa[,
    -which(names(test.PCa) == "PSA.level")]))
lasso_predictions <- as.vector(lasso_predictions)

calculate_metrics <- function(actual, predicted) {
    mse <- mean((actual - predicted)^2)
    rmse <- sqrt(mse)
    r2 <- 1 - (sum((actual - predicted)^2)/sum((actual - mean(actual))^2))
    return(list(MSE = mse, RMSE = rmse, R2 = r2))
}
metrics_lasso <- calculate_metrics(test.PCa$PSA.level, lasso_predictions)

model_comparison <- data.frame(Model = c("Tree Model", "Best Subset Regression",
    "Lasso Regression"), MSE = c(tree_mse, regression_mse, metrics_lasso$MSE), RMSE = c(tree_rmse,
    regression_rmse, metrics_lasso$RMSE), R_squared = c(tree_r2, regression_r2, metrics_lasso$R2))

print(model_comparison)
```

```
##                    Model       MSE     RMSE  R_squared
## 1             Tree Model  947.8247 30.78676  0.4647200
## 2 Best Subset Regression 2432.6798 49.32220 -0.3738457
## 3       Lasso Regression 1589.3976 39.86725  0.1023944
```

The regression tree model remains the top performer on the test data, explaining 46% of the variance (R-squared = 0.4647). Its RMSE of 30.79 indicates reasonable prediction accuracy, with predictions typically deviating by about 30 units from the actual values. In contrast, both the best subset regression and Lasso regression models perform poorly, with negative R-squared values of -0.37385 and -0.10069, respectively. These negative values suggest that both models fail to generalize to the test data, underperforming even compared to predicting the mean of the target variable. The best subset regression model likely suffers from overfitting, while the Lasso model shows signs of insufficient regularization. Overall, the regression tree model stands out for its better performance and interpretability, offering a more reliable and robust solution for the given task.

f-) Which model is more easily interpreted and why? (5pts)

The regression tree stands out as the more easily interpretable model, offering both superior performance metrics and an intuitive structure. Its visual, rule-based representation clearly illustrates how predictors influence PSA levels, making it accessible to both technical and non-technical stakeholders. This clarity helps communicate complex relationships in a straightforward way, facilitating decision-making and practical applications.

A key strength of regression trees is their ability to naturally model non-linear relationships and interactions between variables, without the need for explicitly defined interaction terms as in traditional regression models. For instance, if PSA levels change differently across age groups at varying cancer volume thresholds, a regression tree can seamlessly capture these nuances within its branches. In contrast, traditional regression models with multiple predictors and interactions require careful interpretation of coefficients, making them

more difficult to understand and explain.

Ultimately, regression trees offer a more holistic and intuitive approach, particularly when relationships between variables are complex and non-linear. This makes them not only effective in predictive performance but also valuable for communicating insights in a way that is easily understood and trusted by a diverse audience.