CSCI E-106:Assignment 8

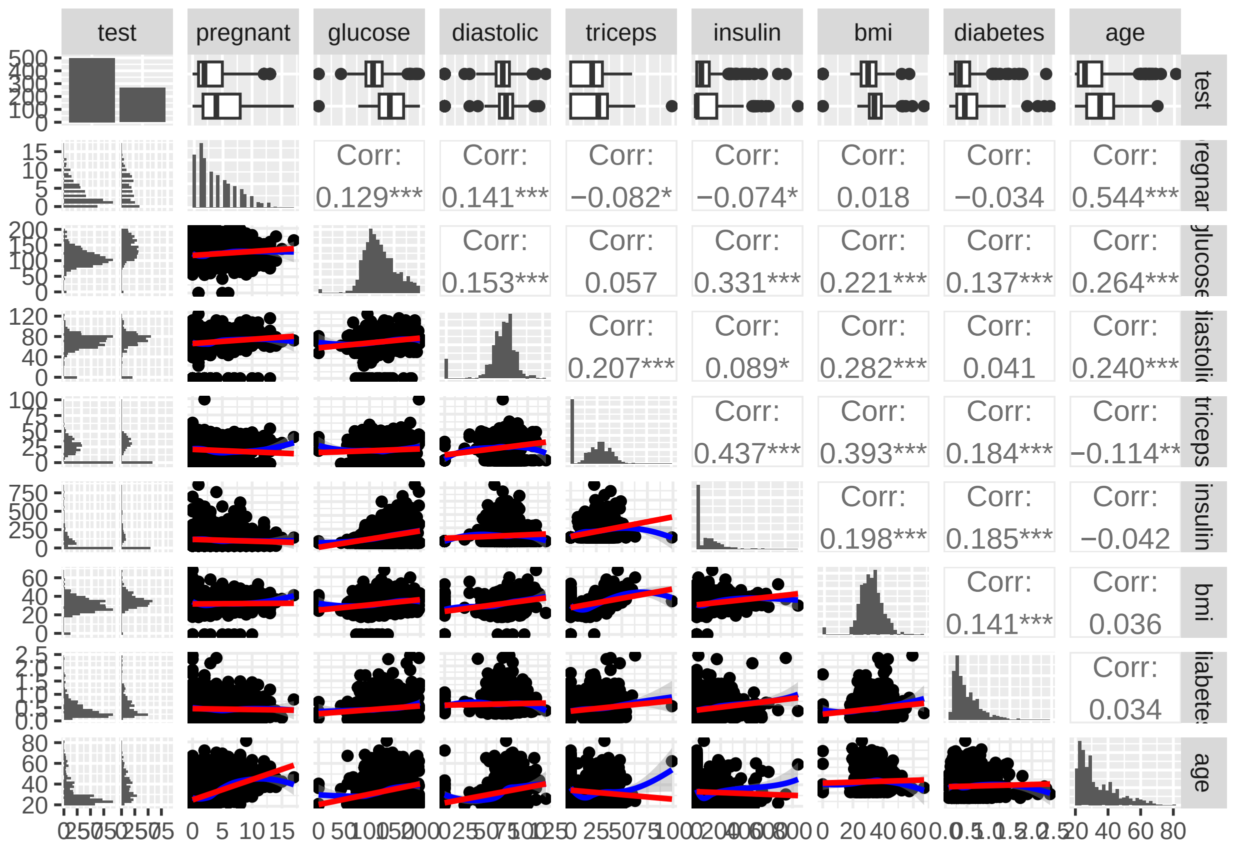
Problem 1

Use the pima data(copy and paste the following command into R console:library(faraway);data(pima). The pima dataset consists of 768 female Pima Indians. We want to predict the diabetes test result from the other predictors.

a-) Plot the test results against each of the predictors. Identify impossible values of the predictors and replace them with the missing value code. Plot the data again and comment on the relationships. (10 pts)

```r
data("pima")
pima$test = as.factor(pima$test)
levels(pima$test) <- c("negative","positive")
summary(pima)
```

```
##     pregnant         glucose        diastolic         triceps
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     insulin          bmi           diabetes           age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##       test
##  negative:500
##  positive:268
##
##
##
##
```

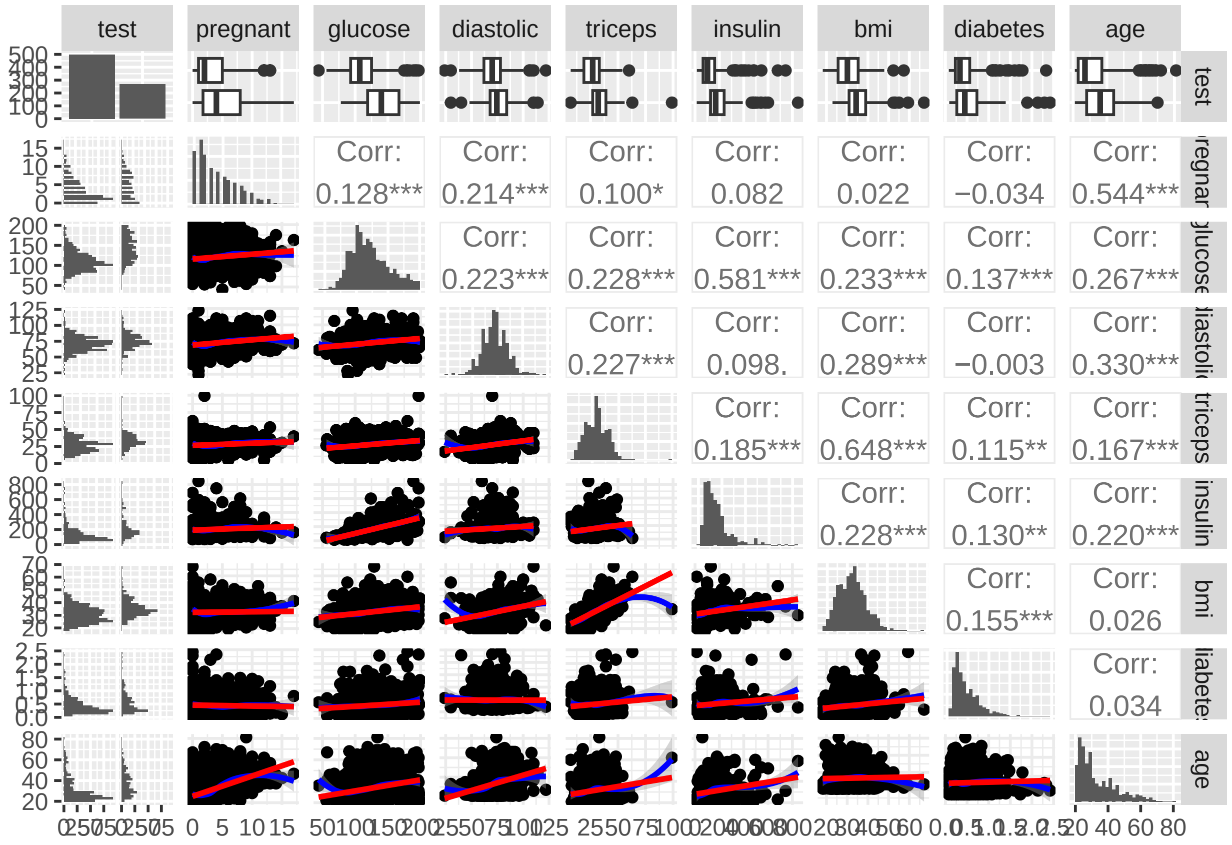


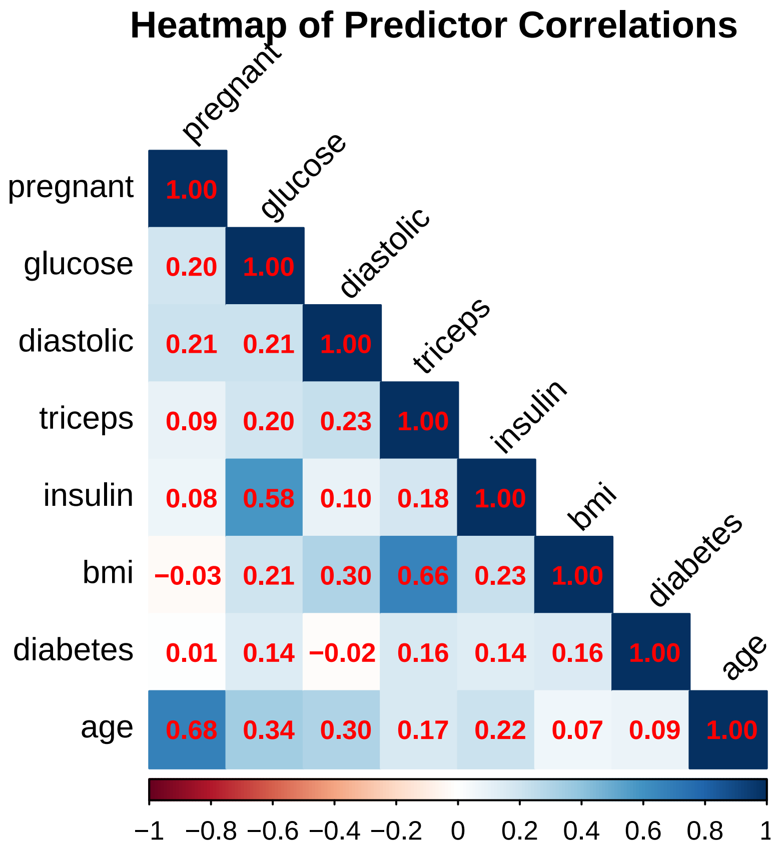
```
zero_counts <- sapply(pima, function(x) sum(x == 0))
zero_counts
```

```
##  pregnant   glucose diastolic   triceps   insulin       bmi  diabetes       age 
##       111         5        35       227       374        11         0         0 
##      test 
##         0
```

```
impossible_predictors <- c("glucose", "triceps", "bmi", "insulin", "diastolic")
pima[impossible_predictors] <- lapply(pima[impossible_predictors], function(x) {
  x[x == 0] <- NA
  return(x)
})
summary(pima)
```

```
##     pregnant         glucose        diastolic         triceps     
##  Min.   : 0.000   Min.   : 44.0   Min.   : 24.00   Min.   : 7.00  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.00   1st Qu.:22.00  
##  Median : 3.000   Median :117.0   Median : 72.00   Median :29.00  
##  Mean   : 3.845   Mean   :121.7   Mean   : 72.41   Mean   :29.15  
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:36.00  
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
##                   NA's   :5       NA's   :35       NA's   :227    
##     insulin            bmi           diabetes           age       
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0780   Min.   :21.00  
##  1st Qu.: 76.25   1st Qu.:27.50   1st Qu.:0.2437   1st Qu.:24.00  
##  Median :125.00   Median :32.30   Median :0.3725   Median :29.00  
##  Mean   :155.55   Mean   :32.46   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:190.00   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##  NA's   :374      NA's   :11
##        test
##  negative:500
##  positive:268
##
##
##
##
##
```





Upon removing the impossible values from the pima dataset, we examine the relationships the scatterplot and correlation matrix reveals. There are a handful of significant correlations amongst the predictor variables, specifically between `insulin` and `glucose`, `bmi` and `triceps`, and `age` and `pregnant`.

b-) Delete all the missing values and create 70% of the data for train data set and use remaining data (30% of data) for test data set (use set.seed(1023)). (10 pts)

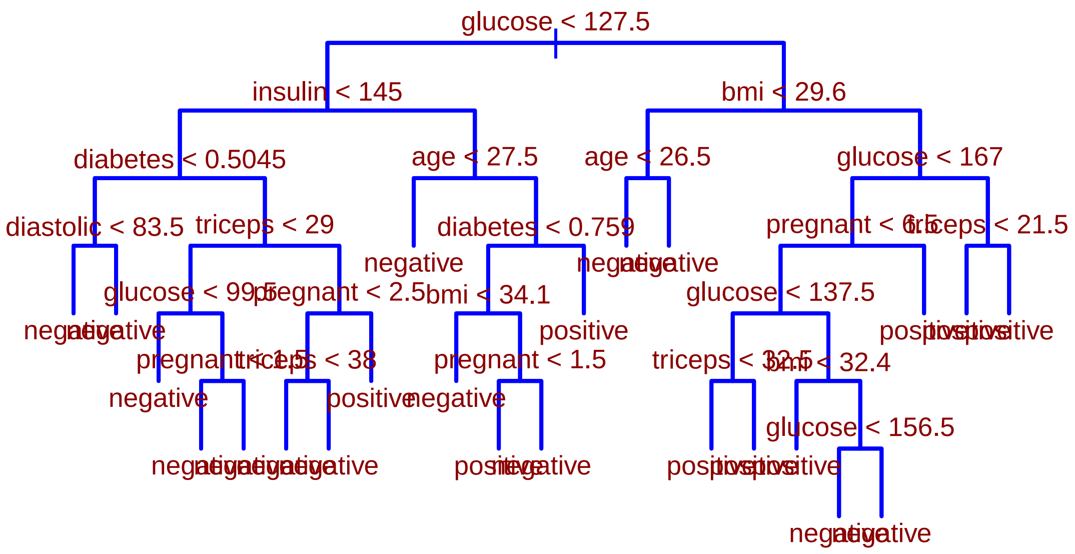
```
pima_clean <- na.omit(pima)
set.seed(1023)
DataSplit<-createDataPartition(y = pima_clean$test, p = 0.7, list = FALSE)
training_data <- pima_clean[DataSplit,]
testing_data <- pima_clean[-DataSplit, ]
```

c-) Fit the default tree model with the result of the diabetes test as the response and all the other variables as predictors using the training set. Make a nice plot of the tree and comment on the shape of the tree. Is it broad or deep? (20 points)

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

```
##
## Classification tree:
## tree(formula = test ~ ., data = training_data)
## Number of terminal nodes:  23
## Residual mean deviance:  0.3712 = 93.54 / 252
## Misclassification error rate: 0.09091 = 25 / 275
```

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



We see that the training error rate is 9%. A small deviance indicates a tree that provides a good fit to the (training) data. The residual mean deviance reported is simply the deviance divided by $n - |T_0|$, which in this case is $275 - 23 = 252$. The most important predictor of `test` appears to be `glucose`, since the first branch differentiates the training data observations between those with a glucose higher than 127.5 and lower than 127.5. From the visualization of the tree, it seems both deep and broad given the amount of variables that play a role on the `test` results.

d-) What fraction of cases in the test set are correctly classified according to this model? (10 pts)

```
tree.pred.train <- predict(pima.tree.mod, training_data, type = "class")
table(tree.pred.train, training_data$test)
```

```
##
## tree.pred.train negative positive
##        negative      175       15
##        positive        9       76
```

```
accuracy <- mean(tree.pred.train == training_data$test)
accuracy
```

```
## [1] 0.9127273
```

$\approx$ 91% of the cases in the test set are correctly classified according to this model on the training data.

e-) Create the confusion matrix and comment on the model performance (10 pts)

```
confusion_matrix_train <- confusionMatrix(tree.pred.train, training_data$test, mode="prec_recall", posi
confusion_matrix_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative      175       15
##   positive        9       76
##
##                Accuracy : 0.9127
##                  95% CI : (0.8729, 0.9433)
##     No Information Rate : 0.6691
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7996
##
##  Mcnemar's Test P-Value : 0.3074
##
##               Precision : 0.8941
##                  Recall : 0.8352
##                      F1 : 0.8636
##              Prevalence : 0.3309
##          Detection Rate : 0.2764
##    Detection Prevalence : 0.3091
##       Balanced Accuracy : 0.8931
##
##        'Positive' Class : positive
##
```

First, we breakdown the Confusion Matrix:

- True Negatives (TN): 175 instances were correctly predicted as negative.
- False Positives (FP): 9 instances were incorrectly predicted as positive.
- False Negatives (FN): 15 instances were incorrectly predicted as negative.
- True Positives (TP): 76 instances were correctly predicted as positive.

Next, we dive into the Performance Metrics:

1. **Accuracy (91%)**: This is the percentage of correct predictions. A high accuracy here (92%) suggests the model is performing well on the training data. However, accuracy alone may not fully capture performance, especially in imbalanced datasets.
2. **Kappa (0.7996)**: Kappa measures the agreement between predicted and actual classes, adjusted for chance. A Kappa value of 0.80 is quite strong, indicating good predictive performance beyond random chance.

Then, we examine the Class Specific Metrics:

1. **Precision (0.8941)**: Precision indicates that when the model predicts positive, it is correct about 89.4% of the time. This is a high value, meaning the model has a relatively low rate of false positives.
2. **Recall (0.8352)**: Recall measures the model's ability to identify actual positives. Here, it captures about 83.5% of positive cases. This indicates the model is doing well at detecting true positives but may miss some (false negatives).
3. **F1 Score (0.8636)**: The F1 score, which balances precision and recall, is 86.4%. This is a high value, indicating that both precision and recall are strong.
4. **Balanced Accuracy (0.8931)**: This is the average of sensitivity and specificity, accounting for imbalances in class distribution. A balanced accuracy of 89.3% suggests that the model performs consistently well across both classes.

Overall, the model shows strong performance on the training data, with high accuracy, balanced error rates, and a good balance between precision and recall. However, these metrics are based on the training data and may not reflect how well the model will generalize to new data.

f-) Create the confusion matrix on the test data set and comment on model performance (10 pts)

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

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction negative positive
##   negative       60       14
##   positive       18       25
##
##                Accuracy : 0.7265
##                  95% CI : (0.6364, 0.8048)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.09981
##
##                   Kappa : 0.4
##
##  Mcnemar's Test P-Value : 0.59588
##
##               Precision : 0.5814
##                  Recall : 0.6410
##                      F1 : 0.6098
##              Prevalence : 0.3333
##          Detection Rate : 0.2137
##    Detection Prevalence : 0.3675
##       Balanced Accuracy : 0.7051
##
##        'Positive' Class : positive
##
```

First, we breakdown the Confusion Matrix:

- True Negatives (TN): 60 instances were correctly predicted as negative.
- False Positives (FP): 18 instances were incorrectly predicted as positive.
- False Negatives (FN): 14 instances were incorrectly predicted as negative.
- True Positives (TP): 25 instances were correctly predicted as positive.

Next, we dive into the Performance Metrics:

1. **Accuracy (72.65%)**: This indicates that approximately 72.659% of all predictions were correct. However, this metric alone can be misleading in the presence of class imbalance.
2. **95% Confidence Interval (0.6364, 0.8048)**: This interval suggests that we can be 95% confident that the true accuracy lies between approximately 0.6364 and 0.8048.
3. **No Information Rate (NIR) (0.6667)**: This is the accuracy one would expect from a model that predicts the majority class (in this case, negative). The model's accuracy is slightly higher than the NIR, but not significantly so (as indicated by the p-value).
4. **P-Value [Acc > NIR] (0.09981)**: This p-value indicates that the model's accuracy is not statistically significantly better than random guessing or simply predicting the majority class.
5. **Kappa (0.4)**: Kappa measures the agreement between predicted and actual classifications, adjusted for chance. A Kappa value of 0.4 indicates a moderate level of agreement beyond chance, but it suggests room for improvement.

Then, we examine the Class Specific Metrics:

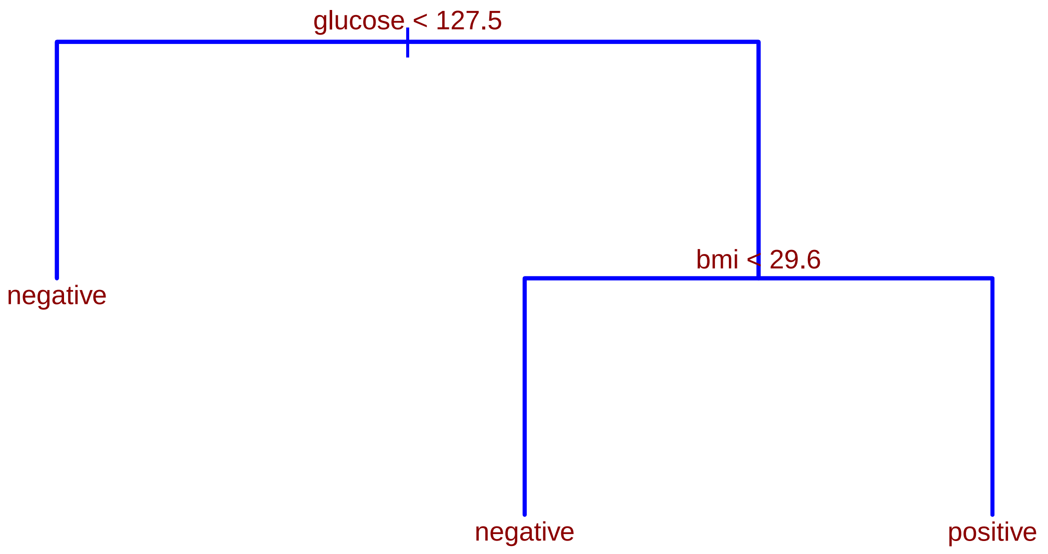
1. **Precision (0.5814)**: This means that when the model predicts positive, it is correct 58% of the time. While better than random guessing, it indicates that there are still many false positives.
2. **Recall (0.6410)**: This shows that the model identifies 64.10% of all actual positive cases. The model could improve its sensitivity to capture more positives.
3. **F1 Score (0.6098)**: The F1 score, the harmonic mean of precision and recall, is 60.98%. This suggests a balance between precision and recall, but both could be improved.
4. **Prevalence (0.3333)**: About one-third of the cases in the test set are positive.

5. **Detection Rate (0.2137)**: This indicates the proportion of actual positives that were detected (true positives), showing that the model could detect only about 21.37% of positive cases.
6. **Detection Prevalence (0.3675)**: This shows the proportion of all cases that the model predicted as positive, suggesting that the model is over-predicting the positive class.
7. **Balanced Accuracy (0.7051)**: This metric averages the sensitivity (recall) and specificity, indicating that the model has balanced performance across both classes, though still needing improvement.

The model exhibits moderate performance, with an accuracy of around 73%. While the model performs better than random guessing, the lack of statistical significance and the Kappa value indicate that there is considerable room for improvement. Specifically, the precision and recall suggest that the model struggles to identify positive cases effectively.

g-) Use cross-validation to select the optimal tree size. For the selected model, check the performance on the test data set. Compare agains part f. (30 points)

```
cv.pima.tree <- cv.tree(pima.tree.mod, FUN = prune.misclass) # Use cross-validation to determine the op
optimal_size <- cv.pima.tree$size[which.min(cv.pima.tree$dev)] # Identify the optimal tree size based o
pruned_tree <- prune.misclass(pima.tree.mod, best = optimal_size) # Prune the tree to the optimal size
plot(pruned_tree, type = "uniform", col = "blue", lwd = 2, main = "Pruned Regression Tree for PIMA Outc
text(pruned_tree, pretty = 0, cex = 0.8, col = "darkred")
```



```
tree.pred.prund.test <- predict(pruned_tree, testing_data, type = "class")

confusion_matrix_prund_test <- confusionMatrix(tree.pred.prund.test, testing_data$test, mode = "prec_re
confusion_matrix_prund_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative       61       14
##   positive       17       25
##
##                Accuracy : 0.735
##                  95% CI : (0.6455, 0.8123)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.06868
##
```

```
##                   Kappa : 0.4151
##
##  Mcnemar's Test P-Value : 0.71944
##
##               Precision : 0.5952
##                  Recall : 0.6410
##                      F1 : 0.6173
##              Prevalence : 0.3333
##          Detection Rate : 0.2137
##    Detection Prevalence : 0.3590
##       Balanced Accuracy : 0.7115
##
##        'Positive' Class : positive
##
```

Table 1.1: Full vs Pruned Tree Performance on Pima Test Data

Metric	Full Tree	Pruned Tree
Accuracy	0.7265	0.735
95% Confidence Interval	(0.6364, 0.8048)	(0.6455, 0.8123)
No Information Rate	0.6667	0.6667
P-Value	0.09981	0.06868
Kappa	0.4	0.4151
Mcnemar's Test P-Value	0.59588	0.71944
Precision	0.5814	0.5952
Recall	0.6410	0.6410
F1 Score	0.6098	0.6173
Prevalence	0.3333	0.3333
Detection Rate	0.2137	0.2137
Detection Prevalence	0.3675	0.3590
Balanced Accuracy	0.7051	0.7115

In the pruned tree, we observe slightly higher accuracy, precision, recall, F1 score, and balanced accuracy, indicating that the pruned tree has a better generalization on the test data.