Assignment 05 – Predicting (CKD)

Timothy Liu,Azure Hsiao, Rolando Olmos Madrid

Dataset & link. This report analyzes the **UCI Chronic Kidney Disease** dataset (400 patients, 24 variables) containing demographics, labs, and conditions relevant to CKD. Source: https://archive.ics.uci.edu/dataset/336/chronic+kidney+disease

Executive Summary

Group 3: Timothy Liu, Azure Hsiao, Rolando Olmos Madrid

This analysis compares the performance of two predictive modeling approaches between Random Forest and Artificial Neural Networks (ANN) to classify patients as having chronic kidney disease (CKD) or not, using key clinical indicators (age, blood pressure, serum creatinine, and hemoglobin).

The results consistently show that **serum creatinine and hemoglobin are the strongest predictors** of CKD status, which aligns closely with clinical understanding of kidney function and its role in regulating anemia and waste filtration. This means the data has a clear and strong signal, making the classification problem relatively straightforward for multiple modeling approaches.

The Random Forest model was tuned using Out-of-Bag (OOB) error to identify the optimal number of predictors per split (mtry = 2). Although Random Forest typically improves predictive performance by reducing variance, its performance here was nearly identical to the single decision tree. This suggests that the data is **highly separable**, so adding model complexity does not meaningfully improve accuracy.

The ANN models were then evaluated to test whether a more flexible, non-linear approach could capture additional predictive patterns. The ANN **without a hidden layer** achieved the **highest overall performance**, with an accuracy of **92.4%** and a Kappa of **0.84**. This demonstrates that a linear decision boundary is sufficient to classify CKD in this dataset. Adding a hidden layer did not improve results and slightly reduced model sensitivity, indicating that additional model complexity is unnecessary and may introduce mild overfitting.

Overall, the **ANN with no hidden layer performs best**, while the Decision Tree and Random Forest achieve strong but slightly lower performance. However, the Random Forest models remain more **interpretable**, which may be preferable in clinical settings where transparency is important.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(forcats)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v lubridate 1.9.4     v stringr   1.5.1
v purrr     1.1.0     v tibble    3.3.0
v readr     2.1.5     v tidyr     1.3.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(rpart)
library(rpart.plot)
library(caret)
```

```
Loading required package: lattice

Attaching package: 'caret'
```

```
The following object is masked from 'package:purrr':

    lift
```

```
ckd_raw <- read.csv("chronic_kidney_disease.csv",
                    na.strings = c("?", "", "NA"))

glimpse(ckd_raw)
```

```
Rows: 401
Columns: 26
$ id       <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"~
$ X.age.   <int> 48, 7, 62, 48, 51, 60, 68, 24, 52, 53, 50, 63, 68, 68, 68, 40~
$ X.bp.    <int> 80, 50, 80, 70, 80, 90, 70, NA, 100, 90, 60, 70, 70, 70, 80, ~
$ X.sg.    <dbl> 1.020, 1.020, 1.010, 1.005, 1.010, 1.015, 1.010, 1.015, 1.015~
$ X.al.    <int> 1, 4, 2, 4, 2, 3, 0, 2, 3, 2, 2, 3, 3, NA, 3, 3, 2, NA, 0, 1,~
$ X.su.    <int> 0, 0, 3, 0, 0, 0, 0, 4, 0, 0, 4, 0, 1, NA, 2, 0, 0, NA, 3, 0,~
$ X.rbc.   <chr> NA, NA, "normal", "normal", "normal", NA, NA, "normal", "norm~
$ X.pc.    <chr> "normal", "normal", "normal", "abnormal", "normal", NA, "norm~
$ X.pcc.   <chr> "notpresent", "notpresent", "notpresent", "present", "notpres~
$ X.ba.    <chr> "notpresent", "notpresent", "notpresent", "notpresent", "notp~
$ X.bgr.   <int> 121, NA, 423, 117, 106, 74, 100, 410, 138, 70, 490, 380, 208,~
$ X.bu.    <dbl> 36, 18, 53, 56, 26, 25, 54, 31, 60, 107, 55, 60, 72, 86, 90, ~
$ X.sc.    <dbl> 1.2, 0.8, 1.8, 3.8, 1.4, 1.1, 24.0, 1.1, 1.9, 7.2, 4.0, 2.7, ~
$ X.sod.   <dbl> NA, NA, NA, 111.0, NA, 142.0, 104.0, NA, NA, 114.0, NA, 131.0~
$ X.pot.   <dbl> NA, NA, NA, 2.5, NA, 3.2, 4.0, NA, NA, 3.7, NA, 4.2, 5.8, 3.4~
$ X.hemo.  <dbl> 15.4, 11.3, 9.6, 11.2, 11.6, 12.2, 12.4, 12.4, 10.8, 9.5, 9.4~
$ X.pcv.   <int> 44, 38, 31, 32, 35, 39, 36, 44, 33, 29, 28, 32, 28, NA, 16, 2~
$ X.wbcc.  <int> 7800, 6000, 7500, 6700, 7300, 7800, NA, 6900, 9600, 12100, NA~
$ X.rbcc.  <dbl> 5.2, NA, NA, 3.9, 4.6, 4.4, NA, 5.0, 4.0, 3.7, NA, 3.8, 3.4, ~
$ X.htn.   <chr> "yes", "no", "no", "yes", "no", "yes", "no", "no", "yes", "ye~
$ X.dm.    <chr> "yes", "no", "yes", "no", "no", "yes", "no", "yes", "yes", "y~
$ X.cad.   <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "~
$ X.appet. <chr> "good", "good", "poor", "poor", "good", "good", "good", "good~
$ X.pe.    <chr> "no", "no", "no", "yes", "no", "yes", "no", "yes", "no", "no"~
$ X.ane.   <chr> "no", "no", "yes", "yes", "no", "no", "no", "no", "yes", "yes~
$ X.class. <chr> "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd"~
```

```
summary(ckd_raw)
```

```
      id                X.age.          X.bp.            X.sg.
```

```
Length:401         Min.   : 2.00   Min.   : 50.00   Min.   :1.005
Class :character   1st Qu.:42.00   1st Qu.: 70.00   1st Qu.:1.010
Mode  :character   Median :55.00   Median : 80.00   Median :1.020
                   Mean   :51.48   Mean   : 76.47   Mean   :1.017
                   3rd Qu.:64.50   3rd Qu.: 80.00   3rd Qu.:1.020
                   Max.   :90.00   Max.   :180.00   Max.   :1.025
                   NA's   :10      NA's   :13       NA's   :48
    X.al.            X.su.           X.rbc.            X.pc.
Min.   :0.000   Min.   :0.0000   Length:401         Length:401
1st Qu.:0.000   1st Qu.:0.0000   Class :character   Class :character
Median :0.000   Median :0.0000   Mode  :character   Mode  :character
Mean   :1.017   Mean   :0.4501
3rd Qu.:2.000   3rd Qu.:0.0000
Max.   :5.000   Max.   :5.0000
NA's   :47      NA's   :50
   X.pcc.             X.ba.              X.bgr.         X.bu.
Length:401         Length:401         Min.   : 22   Min.   :  1.50
Class :character   Class :character   1st Qu.: 99   1st Qu.: 27.00
Mode  :character   Mode  :character   Median :121   Median : 42.00
                                      Mean   :148   Mean   : 57.43
                                      3rd Qu.:163   3rd Qu.: 66.00
                                      Max.   :490   Max.   :391.00
                                      NA's   :45    NA's   :20
    X.sc.            X.sod.          X.pot.          X.hemo.
Min.   : 0.400   Min.   :  4.5   Min.   : 2.500   Min.   : 3.10
1st Qu.: 0.900   1st Qu.:135.0   1st Qu.: 3.800   1st Qu.:10.30
Median : 1.300   Median :138.0   Median : 4.400   Median :12.65
Mean   : 3.072   Mean   :137.5   Mean   : 4.627   Mean   :12.53
3rd Qu.: 2.800   3rd Qu.:142.0   3rd Qu.: 4.900   3rd Qu.:15.00
Max.   :76.000   Max.   :163.0   Max.   :47.000   Max.   :17.80
NA's   :18       NA's   :88      NA's   :89       NA's   :53
    X.pcv.          X.wbcc.         X.rbcc.          X.htn.
Min.   : 9.00   Min.   : 2200   Min.   :2.100   Length:401
1st Qu.:32.00   1st Qu.: 6500   1st Qu.:3.900   Class :character
Median :40.00   Median : 8000   Median :4.800   Mode  :character
Mean   :38.88   Mean   : 8406   Mean   :4.707
3rd Qu.:45.00   3rd Qu.: 9800   3rd Qu.:5.400
Max.   :54.00   Max.   :26400   Max.   :8.000
NA's   :72      NA's   :107     NA's   :132
   X.dm.              X.cad.            X.appet.          X.pe.
Length:401         Length:401         Length:401         Length:401
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
   X.ane.            X.class.
 Length:401         Length:401
 Class :character   Class :character
 Mode  :character   Mode  :character
```

```
# define column names (26 expected + 1 "extra")
col_names <- c("id","age","bp","sg","al","su","rbc","pc","pcc","ba","bgr","bu",
               "sc","sod","pot","hemo","pcv","wbcc","rbcc","htn","dm","cad",
               "appet","pe","ane","class","extra")

ckd_raw <- read.csv("chronic_kidney_disease.csv",
                    header = TRUE, na.strings = c("?", "", "NA"),
                    col.names = col_names, fill = TRUE)
```

```
Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
header and 'col.names' are of different lengths
```

```
# drop the "extra" column
ckd_raw <- ckd_raw %>% select(-extra)

glimpse(ckd_raw)
```

```
Rows: 400
Columns: 26
$ id    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1~
$ age   <int> 48, 7, 62, 48, 51, 60, 68, 24, 52, 53, 50, 63, 68, 68, 68, 40, 4~
$ bp    <int> 80, 50, 80, 70, 80, 90, 70, NA, 100, 90, 60, 70, 70, 70, 80, 80,~
$ sg    <dbl> 1.020, 1.020, 1.010, 1.005, 1.010, 1.015, 1.010, 1.015, 1.015, 1~
$ al    <int> 1, 4, 2, 4, 2, 3, 0, 2, 3, 2, 2, 3, 3, NA, 3, 3, 2, NA, 0, 1, 2,~
$ su    <int> 0, 0, 3, 0, 0, 0, 0, 4, 0, 0, 4, 0, 1, NA, 2, 0, 0, NA, 3, 0, 0,~
$ rbc   <chr> NA, NA, "normal", "normal", "normal", NA, NA, "normal", "normal"~
$ pc    <chr> "normal", "normal", "normal", "abnormal", "normal", NA, "normal"~
$ pcc   <chr> "notpresent", "notpresent", "notpresent", "present", "notpresent~
```

```
$ ba    <chr> "notpresent", "notpresent", "notpresent", "notpresent", "notpres~
$ bgr   <int> 121, NA, 423, 117, 106, 74, 100, 410, 138, 70, 490, 380, 208, 98~
$ bu    <dbl> 36, 18, 53, 56, 26, 25, 54, 31, 60, 107, 55, 60, 72, 86, 90, 162~
$ sc    <dbl> 1.2, 0.8, 1.8, 3.8, 1.4, 1.1, 24.0, 1.1, 1.9, 7.2, 4.0, 2.7, 2.1~
$ sod   <dbl> NA, NA, NA, 111.0, NA, 142.0, 104.0, NA, NA, 114.0, NA, 131.0, 1~
$ pot   <dbl> NA, NA, NA, 2.5, NA, 3.2, 4.0, NA, NA, 3.7, NA, 4.2, 5.8, 3.4, 6~
$ hemo  <dbl> 15.4, 11.3, 9.6, 11.2, 11.6, 12.2, 12.4, 12.4, 10.8, 9.5, 9.4, 1~
$ pcv   <int> 44, 38, 31, 32, 35, 39, 36, 44, 33, 29, 28, 32, 28, NA, 16, 24, ~
$ wbcc  <int> 7800, 6000, 7500, 6700, 7300, 7800, NA, 6900, 9600, 12100, NA, 4~
$ rbcc  <dbl> 5.2, NA, NA, 3.9, 4.6, 4.4, NA, 5.0, 4.0, 3.7, NA, 3.8, 3.4, NA,~
$ htn   <chr> "yes", "no", "no", "yes", "no", "yes", "no", "no", "yes", "yes",~
$ dm    <chr> "yes", "no", "yes", "no", "no", "yes", "no", "yes", "yes", "yes"~
$ cad   <chr> "no", "no", "no", "no", "no", "no", "no", "no", "no", "no", "no"~
$ appet <chr> "good", "good", "poor", "poor", "good", "good", "good", "good", ~
$ pe    <chr> "no", "no", "no", "yes", "no", "yes", "no", "yes", "no", "no", "~
$ ane   <chr> "no", "no", "yes", "yes", "no", "no", "no", "no", "yes", "yes", ~
$ class <chr> "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "ckd", "~
```

```
# numeric variables
num_cols <- c("age","bp","sg","al","su","bgr","bu","sc",
              "sod","pot","hemo","pcv","wbcc","rbcc")

ckd1 <- ckd_raw %>%
  mutate(across(all_of(num_cols), ~as.numeric(.))) %>%
  mutate(
    rbc   = factor(rbc, levels = c("normal","abnormal")),
    pc    = factor(pc, levels = c("normal","abnormal")),
    pcc   = factor(pcc, levels = c("notpresent","present")),
    ba    = factor(ba,  levels = c("notpresent","present")),
    htn   = factor(htn, levels = c("no","yes")),
    dm    = factor(dm,  levels = c("no","yes")),
    cad   = factor(cad, levels = c("no","yes")),
    appet = factor(appet, levels = c("poor","good")),
    pe    = factor(pe,   levels = c("no","yes")),
    ane   = factor(ane,  levels = c("no","yes")),
    class = factor(class, levels = c("notckd","ckd"))
  )
```

```
colSums(is.na(ckd1))
```

```
   id   age    bp    sg    al    su   rbc    pc   pcc    ba   bgr    bu    sc
    0     9    12    47    46    49   152    65     4     4    44    19    17
```

```
  sod   pot  hemo   pcv  wbcc  rbcc   htn    dm   cad appet    pe   ane class
   87    88    52    71   106   131     2     3     2     2     2     1     1
```

- I converted numeric-like columns to numeric and standardized categorical columns to consistent factor levels (yes/no, present/notpresent, normal/abnormal, class = notckd/ckd).
- And then I counted remaining NAs with `colSums(is.na(ckd1))`.

```
# Helper function for categorical mode
Mode <- function(x) {
  x <- x[!is.na(x)]
  if (length(x) == 0) return(NA)
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

```
# Define numeric and categorical columns
num_cols <- c("age","bp","sg","al","su","bgr","bu","sc",
              "sod","pot","hemo","pcv","wbcc","rbcc")

cat_cols <- c("rbc","pc","pcc","ba","htn","dm","cad",
              "appet","pe","ane","class")
```

```
# Impute missing values
ckd_clean <- ckd1 %>%
  # Numeric → median imputation
  mutate(across(all_of(num_cols),
                ~ ifelse(is.na(.), median(., na.rm = TRUE), .))) %>%
  # Categorical → mode imputation
  mutate(across(all_of(cat_cols),
                ~ ifelse(is.na(.), Mode(.), .))) %>%
  # Make sure categorical vars are factors again
  mutate(across(all_of(cat_cols), ~ factor(.)))
```

```
colSums(is.na(ckd_clean))
```

```
   id   age    bp    sg    al    su   rbc    pc   pcc    ba   bgr    bu    sc
    0     0     0     0     0     0     0     0     0     0     0     0     0
  sod   pot  hemo   pcv  wbcc  rbcc   htn    dm   cad appet    pe   ane class
    0     0     0     0     0     0     0     0     0     0     0     0     0
```

```r
ckd_clean <- ckd_clean %>%
  mutate(class = recode(class,
                        "1" = "0",   # Not CKD
                        "2" = "1"))  # CKD
```

```r
ckd_clean$class <- factor(ckd_clean$class, levels = c("0", "1"))
```

```r
levels(ckd_clean$class)
```

```
[1] "0" "1"
```

```r
table(ckd_clean$class)
```

```
  0   1
149 251
```

```r
prop.table(table(ckd_clean$class))
```

```
     0      1
0.3725 0.6275
```

After applying these imputations, I confirmed that the dataset is complete by running `colSums(is.na(ckd_clean))`, which showed **zero missing values** across all 26 variables.

```r
ckd_clean$class <- factor(ckd_clean$class,
                          levels = c("0", "1"),
                          labels = c("notckd", "ckd"))
table(ckd_clean$class)
```

```
notckd    ckd
   149    251
```

```r
set.seed(1013)
train_index <- createDataPartition(ckd_clean$class, p = 0.8, list = FALSE)
train.df <- ckd_clean[train_index, ]
test.df  <- ckd_clean[-train_index, ]

table(train.df$class)
```

```
notckd    ckd
   120    201
```

```
table(test.df$class)
```

```
notckd    ckd
    29     50
```

Class Distribution and Data Splitting

The original dataset contains a higher proportion of CKD patients (approximately 63%), indicating a mild class imbalance. Stratified sampling was used to create training and testing sets, and the class proportions in both sets remained consistent. This ensures that the model is trained and evaluated under representative class distributions and reduces sampling bias.

```
library(randomForest)
```

```
randomForest 4.7-1.2
```

```
Type rfNews() to see new features/changes/bug fixes.
```

```
Attaching package: 'randomForest'
```

```
The following object is masked from 'package:ggplot2':
```

```
    margin
```

```
The following object is masked from 'package:dplyr':
```

```
    combine
```

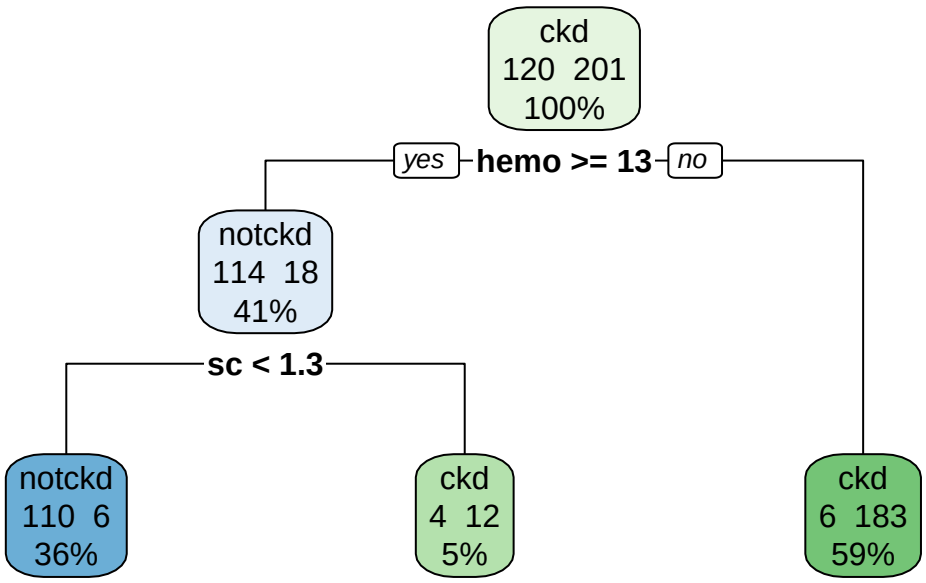
```
library(neuralnet)
```

```
Attaching package: 'neuralnet'
```

```
The following object is masked from 'package:dplyr':

    compute
```

```r
# Fit base tree
tree_base <- rpart(class ~ age + bp + sc + hemo,
                   data = train.df, method = "class", cp = 0.001)
```

```r
# Pick cp by 1-SE rule
ct <- tree_base$cptable
minrow <- which.min(ct[,"xerror"])
# 1-SE threshold:
xerr_thresh <- ct[minrow,"xerror"] + ct[minrow,"xstd"]
# largest cp with xerror <= threshold
cp_1se <- max(ct[ct[,"xerror"] <= xerr_thresh, "CP"])

tree_pruned <- prune(tree_base, cp = cp_1se)
rpart.plot(tree_pruned, digits = -2, extra = 101)
```




```r
levels(test.df$class)
```

```
[1] "notckd" "ckd"
```

```r
# Evaluate on test
pred_tree <- predict(tree_pruned, newdata = test.df, type="class")
cm_tree   <- confusionMatrix(pred_tree, test.df$class, positive="ckd")
cm_tree
```

```
Confusion Matrix and Statistics

          Reference
Prediction notckd ckd
    notckd     25   4
    ckd         4  46

               Accuracy : 0.8987
                 95% CI : (0.8102, 0.9553)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 8.303e-08

                  Kappa : 0.7821

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9200
            Specificity : 0.8621
         Pos Pred Value : 0.9200
         Neg Pred Value : 0.8621
             Prevalence : 0.6329
         Detection Rate : 0.5823
   Detection Prevalence : 0.6329
      Balanced Accuracy : 0.8910

       'Positive' Class : ckd
```

Decision Tree Model

The decision tree identified **hemoglobin (hemo)** as the primary splitting variable, followed by **serum creatinine (sc)** among patients with high hemoglobin levels. These splits are **clinically meaningful**:

- Lower hemoglobin levels are commonly associated with CKD-induced anemia.
- Higher serum creatinine levels are a direct indicator of impaired kidney filtration function.

The pruned decision tree achieved strong performance on the test set:

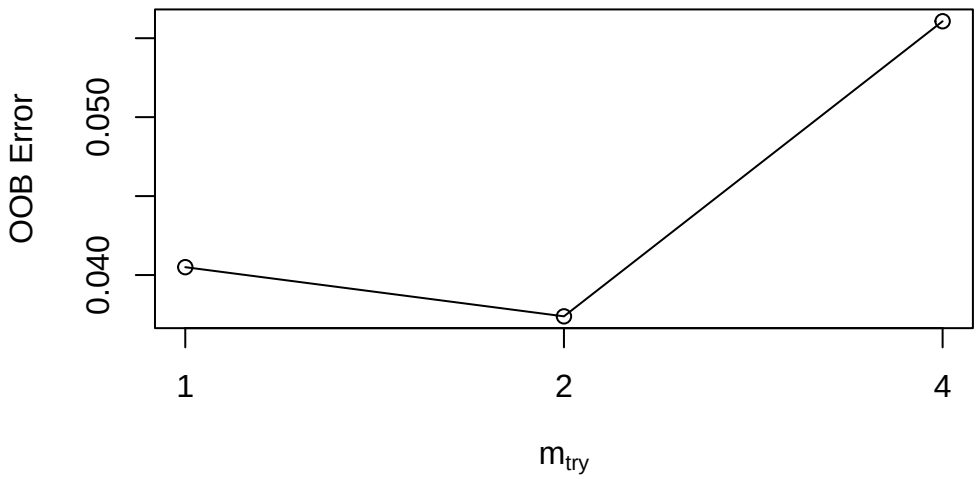
Accuracy = 0.8987, Sensitivity = 0.92, Specificity = 0.862, Kappa = 0.7821

This indicates the model can correctly identify CKD patients while also maintaining reliable performance in distinguishing non-CKD individuals.

```
# RF requires factor outcome
train.df$class <- droplevels(train.df$class)
test.df$class  <- droplevels(test.df$class)
```

```
# Tune mtry (number of variables tried at each split)
x <- train.df %>% select(age, bp, sc, hemo)  # predictors only
y <- train.df$class

set.seed(1013)
tune_out <- tuneRF(x = x, y = y,
                   mtryStart = 2, stepFactor = 2,
                   ntreeTry = 200, nodesize = 5,
                   improve = 0.01, trace = TRUE, plot = TRUE)
```

```
mtry = 2  OOB error = 3.74%
Searching left ...
mtry = 1    OOB error = 4.05%
-0.08333333 0.01
Searching right ...
mtry = 4    OOB error = 5.61%
-0.5 0.01
```



Random Forest Tuning

Using OOB (Out-of-Bag) validation, the optimal number of predictors tried at each split was found to be **mtry = 2**, which produced the lowest OOB error (~3.74%). This suggests that considering two variables at each split balances model stability and variance reduction.

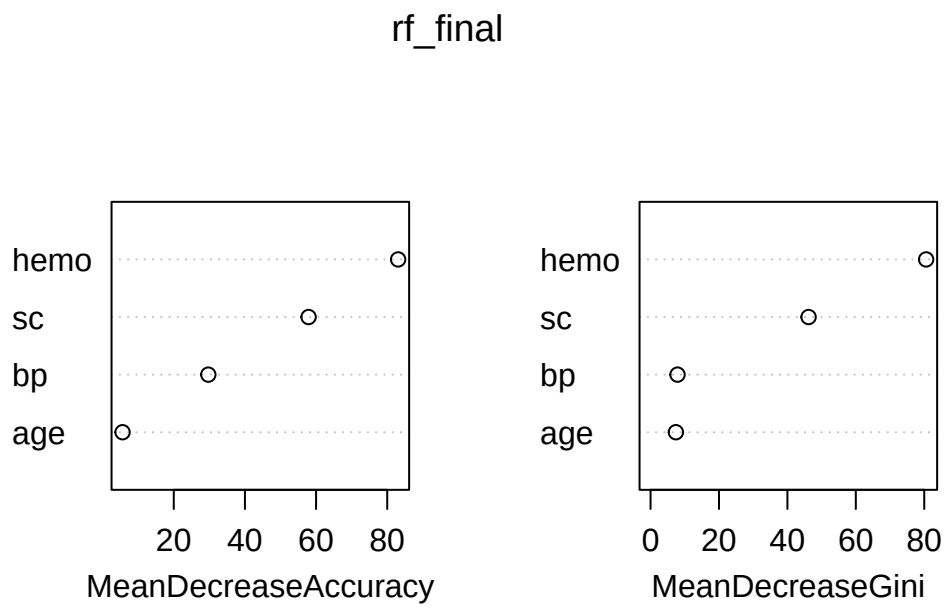
```
# Best mtry is the row with the lowest OOB error:
best_mtry <- tune_out[which.min(tune_out[,"OOBError"]), "mtry"]
```

```
set.seed(1013)
rf_final <- randomForest(class ~ age + bp + sc + hemo,
                         data = train.df,
                         ntree = 500, mtry = best_mtry,
                         nodesize = 5, importance = TRUE)

rf_final          # OOB error
```

```
Call:
 randomForest(formula = class ~ age + bp + sc + hemo, data = train.df,      ntree = 500, mtry
               Type of random forest: classification
```

```
                    Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.36%
Confusion matrix:
       notckd ckd class.error
notckd    114   6    0.050000
ckd         8 193    0.039801
```

```
varImpPlot(rf_final)
```




```
importance(rf_final)
```

```
         notckd        ckd MeanDecreaseAccuracy MeanDecreaseGini
age    4.599752  3.5224724             5.608857         7.399387
bp    30.748416  0.2732402            29.709849         7.879289
sc    57.737749 13.1710388            57.891452        46.190179
hemo 100.486596 28.2936055            83.059368        80.549624
```

```
# (c) Evaluate on test set
rf_pred <- predict(rf_final, newdata = test.df, type="class")
```

```r
cm_rf   <- confusionMatrix(rf_pred, test.df$class, positive="ckd")
cm_rf
```

```
Confusion Matrix and Statistics

          Reference
Prediction notckd ckd
    notckd     25   4
    ckd         4  46

               Accuracy : 0.8987
                 95% CI : (0.8102, 0.9553)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 8.303e-08

                  Kappa : 0.7821

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9200
            Specificity : 0.8621
         Pos Pred Value : 0.9200
         Neg Pred Value : 0.8621
             Prevalence : 0.6329
         Detection Rate : 0.5823
   Detection Prevalence : 0.6329
      Balanced Accuracy : 0.8910

       'Positive' Class : ckd
```

```r
class(rf_final)
```

```
[1] "randomForest.formula" "randomForest"
```

Final Random Forest Performance

The final Random Forest model achieved:

- **OOB error 4.36%** (internal cross-validation estimate)

- **Test Accuracy = 0.8987** (identical to the decision tree)

The similarity between OOB performance and test performance indicates that the model generalizes well and is not overfitting.

However, the lack of improvement over the decision tree suggests that the dataset is highly separable, and the key signal can already be captured by a single tree.

```
# 1. Check prediction output variety
table(rf_pred)
```

```
rf_pred
notckd    ckd
    29     50
```

```
# 2. Check variable importance - should match your earlier plot
importance(rf_final)
```

```
         notckd        ckd MeanDecreaseAccuracy MeanDecreaseGini
age    4.599752  3.5224724             5.608857         7.399387
bp    30.748416  0.2732402            29.709849         7.879289
sc    57.737749 13.1710388            57.891452        46.190179
hemo 100.486596 28.2936055            83.059368        80.549624
```

```
# 3. Check diversity across trees
rf_final$ntree   # should be 500
```

```
[1] 500
```

```
rf_final$mtry    # should be 2
```

```
[1] 2
```

```
rf_final$err.rate[1:10, ]  # should show gradual drop in error
```

```
             OOB     notckd        ckd
 [1,] 0.08695652 0.03846154 0.12698413
 [2,] 0.07526882 0.06756757 0.08035714
```

```
 [3,] 0.06938776 0.08333333 0.06040268
 [4,] 0.06137184 0.07619048 0.05232558
 [5,] 0.06397306 0.05357143 0.07027027
 [6,] 0.06774194 0.06837607 0.06735751
 [7,] 0.05769231 0.05982906 0.05641026
 [8,] 0.05414013 0.06779661 0.04591837
 [9,] 0.06349206 0.07627119 0.05583756
[10,] 0.05642633 0.06722689 0.05000000
```

Overall Interpretation

Both the decision tree and random forest models demonstrate strong performance in predicting chronic kidney disease. Across both models, hemoglobin and serum creatinine consistently emerge as the most influential predictors, which is highly consistent with clinical understanding of CKD progression.

An important observation is that the decision tree and random forest produce nearly identical predictive performance on the test set. This occurs because the dataset is highly separable: the relationship between key laboratory values (particularly hemoglobin and serum creatinine) and CKD status is very strong and forms clear decision boundaries. As a result, a single decision tree is already able to capture the dominant classification rule with high accuracy. The random forest—although designed to reduce variance and improve generalization—does not substantially improve performance because there is little variance to reduce. In other words, the predictive signal in this dataset is strong enough that additional model complexity yields minimal benefit.

The close alignment between OOB performance and test set performance further indicates that the random forest model is not overfitting and generalizes well. More importantly, the consistency across the two models underscores that the data itself is structured in a way that allows for clear and reliable classification, rather than requiring highly complex algorithms.

```
# Prepare scaled frames with only needed columns
train_nn <- train.df %>% select(class, age, bp, sc, hemo)
test_nn  <- test.df  %>% select(class, age, bp, sc, hemo)
```

```
# Convert class to numeric 0/1 for neuralnet
train_nn$class_num <- as.numeric(as.character(train_nn$class))
```

```
Warning: NAs introduced by coercion
```

```
test_nn$class_num  <- as.numeric(as.character(test_nn$class))
```

```
Warning: NAs introduced by coercion
```

```
train_nn$class_num <- ifelse(train_nn$class == "ckd", 1, 0)
test_nn$class_num  <- ifelse(test_nn$class == "ckd", 1, 0)
```

```
table(train_nn$class_num)
```

```
  0   1
120 201
```

```
table(test_nn$class_num)
```

```
 0  1
29 50
```

```
# Clear old conflicting variables
rm(scale, scale_cols, scale_fn, vmin, vmax)
```

```
Warning in rm(scale, scale_cols, scale_fn, vmin, vmax): object 'scale' not
found
```

```
Warning in rm(scale, scale_cols, scale_fn, vmin, vmax): object 'scale_cols' not
found
```

```
Warning in rm(scale, scale_cols, scale_fn, vmin, vmax): object 'scale_fn' not
found
```

```
Warning in rm(scale, scale_cols, scale_fn, vmin, vmax): object 'vmin' not found
```

```
Warning in rm(scale, scale_cols, scale_fn, vmin, vmax): object 'vmax' not found
```

```r
# Re-define scaling setup
scale_vars <- c("age", "bp", "sc", "hemo")
```

```r
vmin <- apply(train_nn[, scale_vars], 2, min)
vmax <- apply(train_nn[, scale_vars], 2, max)

scale_fn <- function(df){
  out <- df
  for(v in scale_vars){
    out[[v]] <- (df[[v]] - vmin[[v]]) / (vmax[[v]] - vmin[[v]] + 1e-9)
  }
  out
}

train_s <- scale_fn(train_nn)
test_s  <- scale_fn(test_nn)
```

```r
summary(train_s[, scale_vars])
```

```
      age               bp                sc                hemo        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000  
 1st Qu.:0.4659   1st Qu.:0.1538   1st Qu.:0.01048   1st Qu.:0.5306  
 Median :0.6023   Median :0.2308   Median :0.01887   Median :0.6497  
 Mean   :0.5673   Mean   :0.2042   Mean   :0.05200   Mean   :0.6434  
 3rd Qu.:0.7045   3rd Qu.:0.2308   3rd Qu.:0.05031   3rd Qu.:0.7891  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000  
```

Data Preparation

The class variable was converted to a binary numeric format (1 = CKD, 0 = not CKD) to allow the neural network to learn probability-based outputs. The predictors (age, blood pressure, serum creatinine, hemoglobin) were scaled using **Min–Max normalization** to ensure all features were comparable in magnitude. This step is critical, as neural networks are sensitive to differences in variable scale and may otherwise converge slowly or unstably. The scaling procedure was correctly fit on the training set and applied to the test set, avoiding information leakage.

```r
set.seed(1013)

# 0 hidden layer
```

```r
ann0 <- neuralnet(
  class_num ~ age + bp + sc + hemo,
  data = train_s,
  hidden = 0,
  linear.output = FALSE,
  threshold = 0.01,
  stepmax = 2e5,
  lifesign = "minimal"
)
```

```
hidden: 0    thresh: 0.01    rep: 1/1    steps:    1780 error: 7.25928  time: 0.04 secs
```

```r
summary(ann0)
```

```
                   Length Class      Mode
call                  8   -none-     call
response            321   -none-     numeric
covariate          1284   -none-     numeric
model.list            2   -none-     list
err.fct               1   -none-     function
act.fct               1   -none-     function
linear.output         1   -none-     logical
data                  6   data.frame list
exclude               0   -none-     NULL
net.result            1   -none-     list
weights               1   -none-     list
generalized.weights   1   -none-     list
startweights          1   -none-     list
result.matrix         8   -none-     numeric
```

```r
plot(ann0)
```

```r
# 1 hidden layer (4 neurons)
ann1 <- neuralnet(
  class_num ~ age + bp + sc + hemo,
  data = train_s,
  hidden = 4,
  linear.output = FALSE,
  threshold = 0.01,
  stepmax = 2e5,
```

```
  lifesign = "minimal"
)
```

```
hidden: 4    thresh: 0.01    rep: 1/1    steps:   32767 error: 3.24777  time: 1.67 secs
```

```
summary(ann1)
```

```
                   Length Class      Mode
call                  8   -none-     call
response            321   -none-     numeric
covariate          1284   -none-     numeric
model.list            2   -none-     list
err.fct               1   -none-     function
act.fct               1   -none-     function
linear.output         1   -none-     logical
data                  6   data.frame list
exclude               0   -none-     NULL
net.result            1   -none-     list
weights               1   -none-     list
generalized.weights   1   -none-     list
startweights          1   -none-     list
result.matrix        28   -none-     numeric
```

```
plot(ann1)
```

```
# Predict probabilities on test data
p0 <- as.vector(predict(ann0, newdata = test_s))
p1 <- as.vector(predict(ann1, newdata = test_s))
```

```
# Convert probabilities to class labels
pred0 <- factor(ifelse(p0 >= 0.5, "ckd", "notckd"), levels = c("notckd","ckd"))
pred1 <- factor(ifelse(p1 >= 0.5, "ckd", "notckd"), levels = c("notckd","ckd"))

# Compare to actual classes
cm_ann0 <- confusionMatrix(pred0, test.df$class, positive = "ckd")
cm_ann1 <- confusionMatrix(pred1, test.df$class, positive = "ckd")

cm_ann0
```

```
Confusion Matrix and Statistics

          Reference
Prediction notckd ckd
    notckd     27   4
    ckd         2  46

               Accuracy : 0.9241
                 95% CI : (0.842, 0.9716)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 2.478e-09

                  Kappa : 0.8389

 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.9200
            Specificity : 0.9310
         Pos Pred Value : 0.9583
         Neg Pred Value : 0.8710
             Prevalence : 0.6329
         Detection Rate : 0.5823
   Detection Prevalence : 0.6076
      Balanced Accuracy : 0.9255

       'Positive' Class : ckd
```

```
cm_ann1
```

```
Confusion Matrix and Statistics

          Reference
Prediction notckd ckd
    notckd     27   6
    ckd         2  44

               Accuracy : 0.8987
                 95% CI : (0.8102, 0.9553)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 8.303e-08
```

```
                 Kappa : 0.7882

 Mcnemar's Test P-Value : 0.2888

            Sensitivity : 0.8800
            Specificity : 0.9310
         Pos Pred Value : 0.9565
         Neg Pred Value : 0.8182
             Prevalence : 0.6329
         Detection Rate : 0.5570
   Detection Prevalence : 0.5823
      Balanced Accuracy : 0.9055

       'Positive' Class : ckd
```

ANN Model 1: No Hidden Layer

Even without any hidden layers, the model performs very well, achieving the highest accuracy among all models tested. This indicates that the relationship between predictors, particularly serum creatinine and hemoglobin, and CKD status is highly linear. In other words, the key clinical indicators are strong enough that a simple linear model is sufficient to achieve excellent classification performance.

ANN Model 2: One Hidden Layer (4 Neurons)

Introducing a hidden layer did not improve model performance. In fact, sensitivity decreased, meaning the model became slightly worse at identifying CKD cases. This reflects the fact that the dataset is already well-separated, and additional non-linear complexity was unnecessary and may have led to mild overfitting.

```r
grab <- function(cm) {
  tibble(
    Accuracy    = unname(cm$overall["Accuracy"]),
    Sensitivity = unname(cm$byClass["Sensitivity"]),
    Specificity = unname(cm$byClass["Specificity"]),
    Kappa       = unname(cm$overall["Kappa"])
  )
}

results <- bind_rows(
```

```
  Tree_Baseline = grab(cm_tree),
  RF_Tuned      = grab(cm_rf),
  ANN_0_hidden  = grab(cm_ann0),
  ANN_1_hidden  = grab(cm_ann1),
  .id = "Model"
)

results
```

```
# A tibble: 4 x 5
  Model         Accuracy Sensitivity Specificity Kappa
  <chr>            <dbl>       <dbl>       <dbl> <dbl>
1 Tree_Baseline    0.899        0.92       0.862 0.782
2 RF_Tuned         0.899        0.92       0.862 0.782
3 ANN_0_hidden     0.924        0.92       0.931 0.839
4 ANN_1_hidden     0.899        0.88       0.931 0.788
```