Universidad Nacional Autónoma de México

Facultad de Ciencias, 2024 - 1

Almacenes y Minería de Datos

Práctica 10. Clasificación

PostgresandoesoSQLazos

Integrantes

Adrian Aguilera Moreno	421005200
Marco Antonio Rivera Silva	318183583
Sebastían Alejandro Gutiérrez Medina	318287021
Israel Hernández Dorantes	318206604
Alejandra Ortega García	420002495
Luis Enrique García Gomez	315063880

Actividades.

i. Contesta lo siguiente:

- ¿Como definirías con tus propias palabras un Recursive Partitioning and Regression Tree?

  Se puede definir como un algoritmo de aprendizaje automático que se utiliza para resolver problemas de regresión y clasificación. Este algoritmo divide repetidamente el conjunto de datos en subconjuntos más pequeños y homogéneos en función de las características o variables predictoras, hasta que se alcanza un criterio de parada predefinido. Cada división se realiza de manera recursiva, creando un árbol de decisiones que se utiliza para predecir los valores de salida o las clases de nuevos datos.

- Investiga los paquetes de R: tidyverse, rpart, rpart.plot y caret.
  - **tidyverse**: Es una colección de paquetes orientados a la manipulación, importación, exploración y visualización de datos, la cual se utiliza en ciencia de datos. El uso de *tidyverse* permite facilitar el trabajo estadístico y la generación de trabajos reproducibles. Está compuesto de los paquetes: readr, dplyr, ggplot2, tibble, tidyr, purr, stringr y forcats.
  - **rpart**: Es una implementación de la metodología CART (Recursive PARTitioning), el cual se utiliza para construir árboles de decisión en análisis de datos. Este paquete proporciona la función principal "rpart()", la cual se utiliza para crear árboles de decisión basados en particiones recursivas, donde además proporciona funciones para visualizar y graficar los árboles de decisión generados.
  - **rpart.plot**: Este es una extensión del paquete *rpart* que se utiliza para visualizar los resultados de los árboles de decisión creados con el paquete *rpart*. Proporciona funciones y herramientas para representar gráficamente los árboles de decisión de una manera más elegante y fácil de interpretar; además permite resaltar características importantes del árbol, como las variables predictoras más relevantes y las reglas de decisión en cada nodo.
  - **caret**: Es un paquete muy útil para el aprendizaje automático y el análisis de datos. "caret" significa "classification and regression training" (entrenamiento de clasificación y regresión) y proporciona una amplia gama de funciones que facilitan el proceso de modelado predictivo. Algunas de sus funcionalidades son la unificación de diferentes algoritmos de aprendizaje automático, el preprocesamiento de datos, la evaluación y selección de modelos; y la visualización y análisis de los resultados de los modelos.

- ¿Que ventaja representa un Random Forest sobre los árboles de decisión?

  Un **Random Forest** tiene la capacidad de manejar datos faltantes y valores atípicos de manera más efectiva que los árboles de decisión individuales; ya que utiliza múltiples árboles y promedia sus resultados, lo cual reduce el impacto de los valores atípicos y a manejar los datos faltantes de una manera más robusta. Además combina sus predicciones para obtener un resultado final, lo cual permite reducir el sesgo y la varianza, resultando en una mayor precisión en las predicciones.

- Supongamos que al aplicar el algoritmo ID3 calculamos la ganancia de los *n* casos y la mejor ganancia se repite en 3 de ellos, ¿qué implica esto?

  Lo que implica es que hay múltiples instancias en el conjunto de datos que comparten las mismas características, las cuales son igualmente informativas para la clasificación. Para este caso, el algoritmo **ID3** podría seleccionar cualquiera de los 3 casos con la mejor ganancia como el nodo de división en el árbol de decisión, y la elección específica dependerá de cómo esté implementado el algoritmo, cómo es el orden de los datos o cómo es la forma en que se manejan los empates.

ii. Realiza los 7 pasos que resulven el siguiente problema:

Imagina que una amiga te solicita ayuda en su trabajo gerencial, su problema consta en la decisión siguiente, si tienen dos posibles diseños para su nueva linea de linea blanca.

- La primera opción tiene un 80 % de probabilidad de generar 70 % productos de su línea correctos y un 20 % de probabilidad de lograr 50 % de produtos correctos. Este diseño tiene un costo de $ 450,000.00.
- La segunda opción tiene una probabilidad del 70 % de lograr el 70 % de productos correctos y un 30 % del 50 % de prodcutos correctos. Su costo sería de $ 600,000.00.
- El costo de cada producto de línea blanca es de $ 100.00, si es correcto se vende en $ 250.00, mientras que si no es correcto no se puede vender, por tanto su costo precio seria $ 0.00.

¿Por qué campaña se debería optar?
Los pasos para solucionarlo deben ser:

- Enumerar las opciones
- Enlistar las alternativas de decisión y sus estados asociados a la misma.
- Agregar el árbol de decisión.
- Asignar la probabilidades a priori de cada estado.
- Calcular el beneficio de cada una de las ramas.
- Resolver el árbol de decisión de derecha a izquierda.
- Resolver la etapa anterior y exponer cual es el mejor resultado.

Se pueden apoyar en .R para realizar la solución, asi que deberan subir su archivo que utilizaron (si es que utilizan R) llamado **arbolDecision.R**.

Solución :

- Enumerar las opciones

(a) Opción 1 con un costo de $450,000.00.

(b) Opción 2 con un costo de $600,000.00.

- Enlistar las alternativas de decisión y sus estados asociados a la misma.

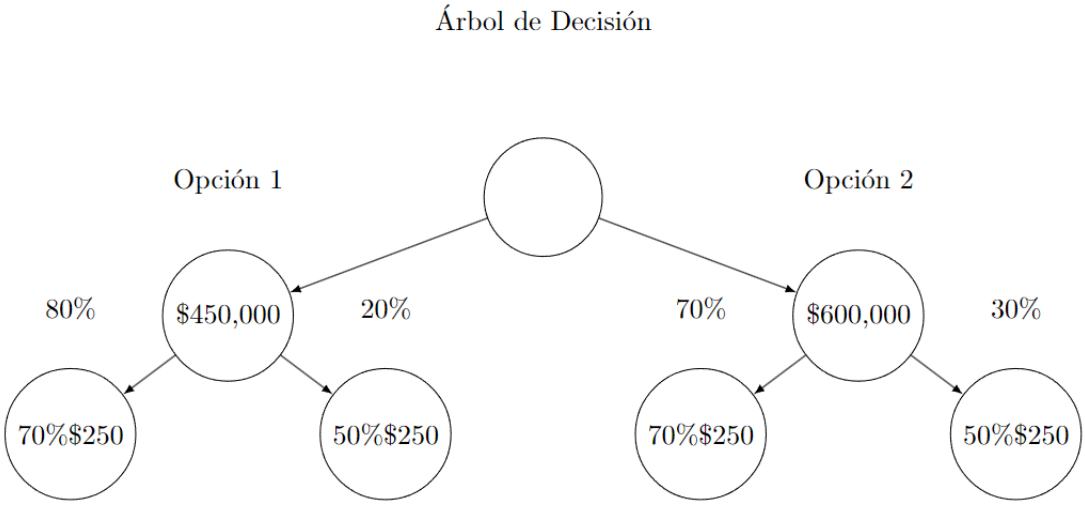
(a) **Alternativa 1**: Elegir Opción 1.

- Estado 1: 70% de productos correctos.
- Estado 2: 50% de productos correctos.

(b) **Alternativa 2**: Elegir Opción 2.

- Estado 3: 70% de productos correctos.
- Estado 4: 50% de productos correctos.

- Agregar el árbol de decisión.



- Asignar la probabilidades a priori de cada estado.

- Probabilidad Estado 1: 0.8
- Probabilidad Estado 2: 0.2
- Probabilidad Estado 3: 0.7

  - Probabilidad Estado 4: 0.3

- Calcular el beneficio de cada una de las ramas.

  - Beneficio Estado 1: (0.7 * $250) - $100 = $75
  - Beneficio Estado 2: (0.5 * $250) - $100 = $25
  - Beneficio Estado 3: (0.7 * $250) - $100 = $75
  - Beneficio Estado 4: (0.5 * $250) - $100 = $25

- Resolver el árbol de decisión de derecha a izquierda.

  - Para Opción 1: (0.8 * $75) + (0.2 * $25) = $65
  - Para Opción 2: (0.7 * $75) + (0.3 * $25) = $60

- Resolver la etapa anterior y exponer cual es el mejor resultado.

  La mejor elección es elegir la opción 1, ya que tiene un beneficio esperado mayor en comparación con la opción 2.

iii. Con el script arbolVino.R elabora un reporte detallado donde incluyas una basta explicación que es lo que hace, como lo hace y que resultados va generando cada línea o sentencia.

A continuación se enlista la explicación de cada uno de los pasos que se llevan a cabo en el script `arbolVino.R`:

- De la línea 2 a la línea 5 cargamos las librerías; `tidyverse`[1], `rpart`[2], `rpart.plot`[3], y `caret`[4].
- La línea 8 y 11 descarga archivos con los nombres `wine.data` y `wine.names`.
- La línea 14 realiza la lectura de las primeras 10 líneas en el archivo `wine.names`.

```
> readLines("wine.data", n = 10)
 [1] "1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065"
 [2] "1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050"
 [3] "1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185"
 [4] "1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480"
 [5] "1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735"
 [6] "1,14.2,1.76,2.45,15.2,112,3.27,3.39,.34,1.97,6.75,1.05,2.85,1450"
 [7] "1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290"
 [8] "1,14.06,2.15,2.61,17.6,121,2.6,2.51,.31,1.25,5.05,1.06,3.58,1295"
 [9] "1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045"
[10] "1,13.86,1.35,2.27,16,98,2.98,3.15,.22,1.85,7.22,1.01,3.55,1045"
```

- Guarda en la variable `vino` los valores guardados en `wine.data` asumiendo que estos se separan por comas y no hay encabezados.
- Devuelve los datos en `vino` para la línea 17. A continuación se muestran los primeros 26 valores en `vino`:

```
> vino
   V1    V2   V3   V4   V5  V6   V7   V8   V9  V10  V11   V12  V13  V14
1   1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29 5.64 1.040 3.92 1065
2   1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28 4.38 1.050 3.40 1050
3   1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81 5.68 1.030 3.17 1185
4   1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18 7.80 0.860 3.45 1480
5   1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82 4.32 1.040 2.93  735
6   1 14.20 1.76 2.45 15.2 112 3.27 3.39 0.34 1.97 6.75 1.050 2.85 1450
7   1 14.39 1.87 2.45 14.6  96 2.50 2.52 0.30 1.98 5.25 1.020 3.58 1290
8   1 14.06 2.15 2.61 17.6 121 2.60 2.51 0.31 1.25 5.05 1.060 3.58 1295
9   1 14.83 1.64 2.17 14.0  97 2.80 2.98 0.29 1.98 5.20 1.080 2.85 1045
10  1 13.86 1.35 2.27 16.0  98 2.98 3.15 0.22 1.85 7.22 1.010 3.55 1045
11  1 14.10 2.16 2.30 18.0 105 2.95 3.32 0.22 2.38 5.75 1.250 3.17 1510
12  1 14.12 1.48 2.32 16.8  95 2.20 2.43 0.26 1.57 5.00 1.170 2.82 1280
13  1 13.75 1.73 2.41 16.0  89 2.60 2.76 0.29 1.81 5.60 1.150 2.90 1320
14  1 14.75 1.73 2.39 11.4  91 3.10 3.69 0.43 2.81 5.40 1.250 2.73 1150
15  1 14.38 1.87 2.38 12.0 102 3.30 3.64 0.29 2.96 7.50 1.200 3.00 1547
16  1 13.63 1.81 2.70 17.2 112 2.85 2.91 0.30 1.46 7.30 1.280 2.88 1310
17  1 14.30 1.92 2.72 20.0 120 2.80 3.14 0.33 1.97 6.20 1.070 2.65 1280
18  1 13.83 1.57 2.62 20.0 115 2.95 3.40 0.40 1.72 6.60 1.130 2.57 1130
19  1 14.19 1.59 2.48 16.5 108 3.30 3.93 0.32 1.86 8.70 1.230 2.82 1680
20  1 13.64 3.10 2.56 15.2 116 2.70 3.03 0.17 1.66 5.10 0.960 3.36  845
21  1 14.06 1.63 2.28 16.0 126 3.00 3.17 0.24 2.10 5.65 1.090 3.71  780
22  1 12.93 3.80 2.65 18.6 102 2.41 2.41 0.25 1.98 4.50 1.030 3.52  770
23  1 13.71 1.86 2.36 16.6 101 2.61 2.88 0.27 1.69 3.80 1.110 4.00 1035
24  1 12.85 1.60 2.52 17.8  95 2.48 2.37 0.26 1.46 3.93 1.090 3.63 1015
25  1 13.50 1.81 2.61 20.0  96 2.53 2.61 0.28 1.66 3.52 1.120 3.82  845
26  1 13.05 2.05 3.22 25.0 124 2.63 2.68 0.47 1.92 3.58 1.130 3.20  830
```

- La línea 23 realiza la lectura de las primeras 10 líneas del archivo `wine.names`:

[1] Está compuesto por varios paquetes que facilitan el proceso de manipulación, visualización y análisis de datos en R.
[2] Se utiliza para construir árboles de decisión.
[3] Visualiza los árboles de decisión creados con `rpart`.
[4] Utilizado para entrenar modelos, por ejemplo los modelos de validación cruzada vistos en clase.

```
> readLines("wine.names", n = 10)
 [1] "1. Title of Database: Wine recognition data"
 [2] "\tUpdated Sept 21, 1998 by C.Blake : Added attribute information"
 [3] ""
 [4] "2. Sources:"
 [5] "   (a) Forina, M. et al, PARVUS - An Extendible Package for Data"
 [6] "       Exploration, Classification and Correlation. Institute of Pharmaceutical"
 [7] "       and Food Analysis and Technologies, Via Brigata Salerno, "
 [8] "       16147 Genoa, Italy."
 [9] ""
[10] "   (b) Stefan Aeberhard, email: stefan@coral.cs.jcu.edu.au"
>
```

- En la línea 26 copia la información de `wine.names` al archivo `wine_names.txt`. Mientras, en la línea 27 se muestra el contenido del archivo creado recientemente. A continuación mostramos algunas líneas en el archivo `wine_name.txt`:

```
1. Title of Database: Wine recognition data
      Updated Sept 21, 1998 by C.Blake : Added attribute information

2. Sources:
   (a) Forina, M. et al, PARVUS - An Extendible Package for Data
       Exploration, Classification and Correlation. Institute of Pharmaceutical
       and Food Analysis and Technologies, Via Brigata Salerno,
       16147 Genoa, Italy.

   (b) Stefan Aeberhard, email: stefan@coral.cs.jcu.edu.au
   (c) July 1991
3. Past Usage:

   (1)
   S. Aeberhard, D. Coomans and O. de Vel,
   Comparison of Classifiers in High Dimensional Settings,
   Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
   Mathematics and Statistics, James Cook University of North Queensland.
   (Also submitted to Technometrics).

   The data was used with many others for comparing various
   classifiers. The classes are separable, though only RDA
   has achieved 100% correct classification.
   (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
   (All results using the leave-one-out technique)

   In a classification context, this is a well posed problem
   with "well behaved" class structures. A good data set
   for first testing of a new classifier, but not very
   challenging.

   (2)
   S. Aeberhard, D. Coomans and O. de Vel,
   "THE CLASSIFICATION PERFORMANCE OF RDA"
   Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
   Mathematics and Statistics, James Cook University of North Queensland.
```

- En la línea 30 se muestra un resumen para `vino`:

```
> summary(vino)
       V1              V2              V3              V4              V5              V6              V7
 Min.   :1.000   Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60   Min.   : 70.00   Min.   :0.980
 1st Qu.:1.000   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20   1st Qu.: 88.00   1st Qu.:1.742
 Median :2.000   Median :13.05   Median :1.865   Median :2.360   Median :19.50   Median : 98.00   Median :2.355
 Mean   :1.938   Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49   Mean   : 99.74   Mean   :2.295
 3rd Qu.:3.000   3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50   3rd Qu.:107.00   3rd Qu.:2.800
 Max.   :3.000   Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00   Max.   :162.00   Max.   :3.880
       V8              V9              V10             V11             V12             V13             V14
 Min.   :0.340   Min.   :0.1300   Min.   :0.410   Min.   : 1.280   Min.   :0.4800   Min.   :1.270   Min.   : 278.0
 1st Qu.:1.205   1st Qu.:0.2700   1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938   1st Qu.: 500.5
 Median :2.135   Median :0.3400   Median :1.555   Median : 4.690   Median :0.9650   Median :2.780   Median : 673.5
 Mean   :2.029   Mean   :0.3619   Mean   :1.591   Mean   : 5.058   Mean   :0.9574   Mean   :2.612   Mean   : 746.9
 3rd Qu.:2.875   3rd Qu.:0.4375   3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170   3rd Qu.: 985.0
 Max.   :5.080   Max.   :0.6600   Max.   :3.580   Max.   :13.000   Max.   :1.7100   Max.   :4.000   Max.   :1680.0
```

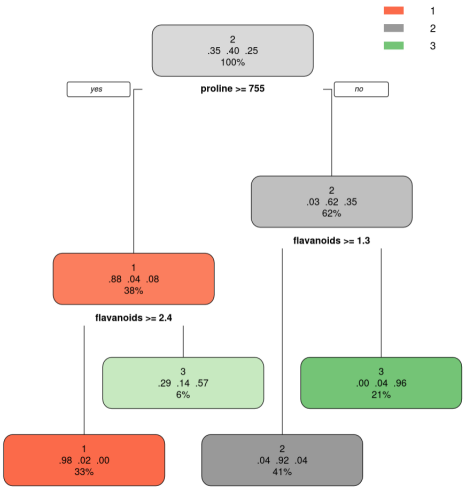
- En la línea 34; los nombres de las columnas se limpian, se convierten a minúsculas y se reemplazan espacios o barras inclinadas con guiones bajos. El primer nombre de columna se establece como

"tipo" (type).

- En la línea 44 los nombres de columna procesados se asignan a las columnas del conjunto de datos `vino`.
- En la línea 47 la columna "tipo" se convierte a un factor utilizando la función `mutate_at`.
- En la línea 50 establecemos la semilla para generar nuestras probabilidades aleatorias, esto con un valor de 1649. En la línea 51 creamos un conjunto de entranmiento `vino_entrenamiento` con el 70% de los datos.
- En la línea 54 se crea el conjunto de prueba `vino_prueba` tomando el complemento del conjunto de entrenamiento en el conjunto de datos completo, esto es un 25% de los datos.
- En la línea 57 se crea el primer árbol de decisión `arbol_1` modelando la variable `tipo` en función de todas las demás variables.
- En la línea 60 se imprime los detalles del primer árbol de decisión. A continuación se muestran estos:

```
> arbol_1
n= 125

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

1) root 125 75 2 (0.35200000 0.40000000 0.24800000)
  2) proline>=755 48  6 1 (0.87500000 0.04166667 0.08333333)
    4) flavanoids>=2.35 41  1 1 (0.97560976 0.02439024 0.00000000) *
    5) flavanoids< 2.35 7  3 3 (0.28571429 0.14285714 0.57142857) *
  3) proline< 755 77 29 2 (0.02597403 0.62337662 0.35064935)
    6) flavanoids>=1.265 51  4 2 (0.03921569 0.92156863 0.03921569) *
    7) flavanoids< 1.265 26  1 3 (0.00000000 0.03846154 0.96153846) *
```

- En la línea 63 se muestra el árbol anterior de manera gráfica, esto es

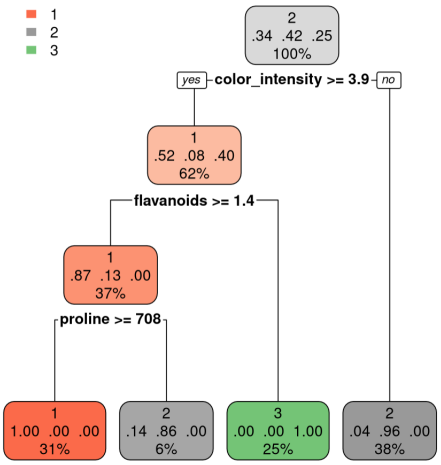


- Se realizan predicciones en el conjunto `vino_prueba` con el uso de `arbol_1` y se aloja en la variable `prediccion_1`.

- En la línea 69 se encuentra la matriz de confusión para que evaluemos al `arbol_1`. A continuación se muestra la matriz de confusión generada:

```
> confusionMatrix(prediccion_1, vino_prueba[["tipo"]])
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 15  0  0
         2  0 15  3
         3  0  6 14

Overall Statistics

               Accuracy : 0.8302
                 95% CI : (0.702, 0.9193)
    No Information Rate : 0.3962
    P-Value [Acc > NIR] : 1.106e-10

                  Kappa : 0.7444

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity             1.000   0.7143   0.8235
Specificity             1.000   0.9062   0.8333
Pos Pred Value          1.000   0.8333   0.7000
Neg Pred Value          1.000   0.8286   0.9091
Prevalence              0.283   0.3962   0.3208
Detection Rate          0.283   0.2830   0.2642
Detection Prevalence    0.283   0.3396   0.3774
Balanced Accuracy       1.000   0.8103   0.8284
>
```

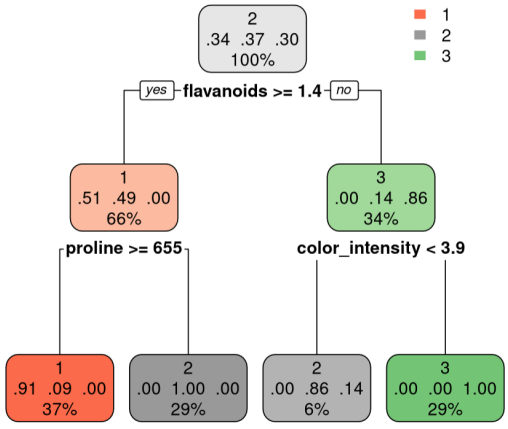
- De la línea 72 a la línea 76 se crean un nuevo árbol de decisión `arbol_2` con una semilla igual a 7439. Además, se realiza una nueva predicción y se aloja en `prediccion_2`. En la línea 79 se genera la gráfica que representa a `arbol_2` de la siguiente manera:



- En la línea 82 se encuentra la matriz de confusión para el `arbol_2`. A continuación se muestran los datos de la matriz de confusión generada recientemente:

```
> confusionMatrix(prediccion_2, vino_prueba_2[["tipo"]])
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 14  0  0
         2  1 21  0
         3  0  0 17

Overall Statistics

               Accuracy : 0.9811
                 95% CI : (0.8993, 0.9995)
    No Information Rate : 0.3962
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9713

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.9333   1.0000   1.0000
Specificity            1.0000   0.9688   1.0000
Pos Pred Value         1.0000   0.9545   1.0000
Neg Pred Value         0.9744   1.0000   1.0000
Prevalence             0.2830   0.3962   0.3208
Detection Rate         0.2642   0.3962   0.3208
Detection Prevalence   0.2642   0.4151   0.3208
Balanced Accuracy      0.9667   0.9844   1.0000
>
```

- De la línea 85 a la línea 90 encontramos un tercer árbol de decisión llamado `arbol_3` con una semilla de 8476, su gráfica se ve de la siguiente manera:



- En la línea 93 se genera la siguiente matriz de confusión (de `arbol_3`):

```
> confusionMatrix(prediccion_3, vino_prueba_3[["tipo"]])
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 15  4  0
         2  0 17  0
         3  0  0 17

Overall Statistics

               Accuracy : 0.9245
                 95% CI : (0.8179, 0.9791)
    No Information Rate : 0.3962
    P-Value [Acc > NIR] : 8.174e-16

                  Kappa : 0.8871

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.8095   1.0000
Specificity            0.8947   1.0000   1.0000
Pos Pred Value         0.7895   1.0000   1.0000
Neg Pred Value         1.0000   0.8889   1.0000
Prevalence             0.2830   0.3962   0.3208
Detection Rate         0.2830   0.3208   0.3208
Detection Prevalence   0.3585   0.3208   0.3208
Balanced Accuracy      0.9474   0.9048   1.0000
>
```

- En la línea 95 se define una función que de un conjunto de datos selecciona 70% de ellos para entrenamiento (una lista llamada de esta manera) y 30% para el conjunto de prueba.
- En la línea 105 se define la función `entrenar_arbol` que devuelve un árbol de decisión entrenado (de hecho una lista que lo contiene) y su respectiva predicción.
- En la línea 121 tenemos una función llamada `obtener_diagnostico` que devuelve la matriz de confusión para un árbol de decisión.
- En la línea 134 se crea la función `crear_arbol`, que devuelve una lista que contiene conjuntos de datos de entrenamiento y prueba, un árbol de decisión entrenado y su diagnóstico asociado (matriz de confusión). Esta última función realiza los pasos que estuvimos haciendo con {`arbol_1`, `arbol_2`, `arbol_3`} en conjunto.
- De la línea 144 a la 147 se invoca a la función `crear_arbol` con una semilla de 1986 y se devuelve la matriz de confusión siguiente:

```
> unarbol[["diagnostico"]]
$matriz
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 15  1  0
         2  1 18  2
         3  0  2 14

Overall Statistics

               Accuracy : 0.8868
                 95% CI : (0.7697, 0.9573)
    No Information Rate : 0.3962
    P-Value [Acc > NIR] : 1.535e-13

                  Kappa : 0.8287

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.9375   0.8571   0.8750
Specificity            0.9730   0.9062   0.9459
Pos Pred Value         0.9375   0.8571   0.8750
Neg Pred Value         0.9730   0.9062   0.9459
Prevalence             0.3019   0.3962   0.3019
Detection Rate         0.2830   0.3396   0.2642
Detection Prevalence   0.3019   0.3962   0.3019
Balanced Accuracy      0.9552   0.8817   0.9105

$mincp
  CP.mínimo CP.original Podar
1     0.005       0.005    NO

>
```