INTRODUCCION AL ANALISIS MULTIVARIADO

Lab. No.6 - Métodos basados en árboles

ARBOLES DE DECISION

1. Cargue los paquetes `rpart` , `rattle` , `DT` , `adabag` y `randomForest` .
2. Cargue los datos de la base `Credit.Rdata` . Vea la cantidad de filas y columnas de `base2` .
3. Muestre 1000 registros usando la función `datatable` , de tal forma que aparezcan 10 registros a la vez, de la siguiente forma:

```
datatable(head(base2,1000), options = list(pageLength = 10, scrollX = TRUE))
```

4. Divida el archivo de datos en dos: uno para entrenar llamado `train` (80%) y el otro para validación llamado `test` (20%). Para reproducir los resultadas, use una semilla en 10, pero antes debe correr la siguiente instrucción una sola vez: `RNGkind(sample.kind = "Rounding")` .
5. Obtenga las dimensiones de la base de entrenamiento y de la base de validación.
6. Use la función `rpart` para generar el árbol con parámetros por defecto. Use todas las variables de la base de entrenamiento como predictores y cliente como respuesta.

```
mod1 = rpart(cliente ~ ., method="class", data=train)
```

7. Obtenga una representación gráfica del árbol de decisión con la función `fancyRpartPlot` . Escoja los colores con `palettes` :

```
fancyRpartPlot(mod1 , palettes=c("Greens", "Reds"))
```

- Cuántos nodos terminales tiene este árbol?
- Cuál es la profundidad de este árbol?
- Cuántos individuos hay en el nodo terminal de la izquierda? Haga una base que seleccione a todos los individuos de ese nodo y vea cuántos elementos tiene. Luego saque el porcentaje que representa esa cantidad del número de individuos de la base de entrenamiento. Compare ese porcentaje con el que da el nodo.
- Obtenga la proporción de buenos que hay en ese nodo a partir de la bae que hizo anteriormente. Compare con lo que dice en el nodo.

8. Haga la clasificación manualmente del primer registro de la base de validación. Observe si el árbol lo clasifica como bueno o malo y compárelo con la clasificación real de esta persona.

9. Haga la clasificación de todos los registros de la base de predicción usando la función `predict` con `newdata=test` y `type="class"` . Ponga los resultados en una base nueva. Verifique que la primera persona es clasificada como “bueno”.
10. Agregue la clasificación obtenida para los registros como una nueva colmuna de la base de predicción.
11. Haga una tabla para comparar la clasificación original con la obtenida mediante el árbol.
12. Haga nuevamente un árbol cambiando algunos parámetros:

- Número mínimo de observaciones para que un nodo se pueda dividir: $minsplit = 5\% * 3556 = 177$.
- Número mínimo de observaciones que debe tener un nodo para ser considerado como terminal: $minbucket = 2\% * 3556 = 71$.
- Profundidad máxima del árbol (con el nodo raíz contabilizado como 0): $maxdepth = 6$.
- Parámetro de complejidad: $cp = 0.004$.

13. Obtenga la representación gráfica.

- ¿Cuántos nodos termiales tiene este árbol?
- Cuál es la profundidad de este árbol?

14. Obtenga manualmente la clasificación del registro 164 de la base de validación. Hágalo con el árbol original y con el segundo árbol. Compare los resultados.

- Haga la clasificación automática para encontrar en qué clase se clasificó este individuo.

15. Obtenga nuevamente la clasificación de todos los registros de la base de predicción y haga nuevamente la tabla cruzada.
16. Para ilustrar el uso de variables ordinales, se va a usar la variable **vivienda** sin declararla como ordinal y luego se declarará como ordinal. Haga un árbol usando como predictores **vivienda** y **antig_laboral**. Muestre el árbol.
17. Declare la variable vivienda como ordinal de la siguiente forma:

```
factor(train$vivienda2, ordered = TRUE, levels = c("padres","alquilada","propia", "contrato priv
        ado","otros")
```

18. Obtenga el árbol nuevamente y compárelo con el anterior.

- Elimine la variable vivienda2 de train para futuros ejercicios.

BAGGING

19. Usando la misma base de entrenamiento (train) se realizará una agregación de bootstrap para predecir el tipo de cliente usando árboles de decisión.

- Haga un modelo clasificación de train, usando la función `bagging` de la librería `adabag` . Indique `method = "class"` y `mfinal = 19` , este es el número de muestras de bootstrap que se usarán para construir los árboles de decisión.

20. Extraiga el primer árbol generado con `mod5$trees[[1]]` .

- Use ese árbol para hacer la clasificación de los clientes de la base de validación con `predict(mod5$trees[[1]],test,type="class")` .
- Use cada árbol para hacer la clasificación de los clientes de la base de validación. Almacene las predicciones en una matriz con 19 columnas.
- Vea en qué clase se clasificó el segundo cliente en los 19 árboles.
- Encuentre la moda de los 19 árboles para cada cliente. Use la función `mlv` de la librería `modeest` . Puede usar esta función dentro de un `apply` por filas (indicando 1).

21. Obtenga la clasificación automática de los clientes de la base de validación con la función `predict` , indicando al final `$class` . Compare los resultados con los obtenidos en el punto anterior.

- Haga la tabla de confusión.

BOSQUES ALEATORIOS

22. Usando de entrenamiento (train) se realizará un bosque aleatorio para predecir el tipo de cliente.

- Haga un modelo clasificación de train, usando la función `randomForest` de la librería `randomForest` . Indique `method = "class"` y `ntree = 100` , este es el número de árboles de decisión. El número de variables que se usa en cada árbol está determinado por el parámetro `mtry` . Se usa el default para `mtry = sqrt(p)` , donde p es el número de variables usadas en general.

23. Obtenga la clasificación automática de todos los clientes de test con la función `predict` .

- Haga la tabla de confusión.

POTENCIACION (BOOSTING)

24. Usando de entrenamiento (train) se aplicará el algoritmo de potenciación para predecir el tipo de cliente.

- Haga un modelo clasificación de train, usando la función `boosting` de la librería `adabag` . Indique `boos = TRUE` y `mfinal = 50` , este es el número de árboles de decisión o número de iteraciones.

25. Obtenga la clasificación automática de todos los clientes de test con la función `predict` indicando `$class`

- Haga la tabla de confusión.

26. Verifique la forma en que la función está dando peso a cada árbol. Primero obtenga los pesos de cada árbol (α_j) con `mod7$weights` .

- Use el `predict` de cada árbol con la base de validación de esta forma: `predict(mod7$trees[[1]],test)[,2]>0.5` . Se usa la segunda columna porque en ella están las probabilidades de “malo” que en este caso es el éxito.
- Use cada árbol para hacer la clasificación de los clientes de la base de validación. Almacene las predicciones en una matriz con 50 columnas llamada `tab` .
- Convierta los resultados de la clasificaicón: FALSE en -1 y TRUE en 1 y llámelo `tab1` .
- Observe cómo ha sido clasificado el primer cliente de la base de validación en los 50 árboles.
- Multiplique la primera fila de `tab1` por el vector de pesos.
- Clasifique al cliente como “bueno” si ese resultado es negativo y como “malo” si es positivo.

- Multiplique toda la matriz `tab1` por el vector de pesos y clasifique los clientes usando el mismo criterio.
- Compare esta clasificación con la obtenida anteriormente de forma automática.