Identificar hongos venenosos con aprendices de reglas

Cada año, muchas personas enferman y algunas incluso mueren por ingerir hongos silvestres venenosos. Dado que muchos hongos son muy similares entre sí en apariencia, en ocasiones, incluso los recolectores de hongos experimentados se envenenan.

A diferencia de la identificación de plantas dañinas, como el roble venenoso o la hiedra venenosa, no existen reglas claras como "hojas de tres, déjenlas" para identificar si un hongo silvestre es venenoso o comestible.

Para complicar las cosas, muchas reglas tradicionales como "los hongos venenosos tienen colores brillantes" brindan información peligrosa o engañosa. Si hubiera reglas simples, claras y consistentes para identificar hongos venenosos, podrían salvar las vidas de los recolectores.

Como una de las fortalezas de los algoritmos de aprendizaje de reglas es el hecho de que generan reglas fáciles de entender, parecen ser una opción adecuada para esta tarea de clasificación. Sin embargo, las reglas solo serán tan útiles como precisas.

Paso 1 - recopilación de datos

Para identificar las reglas para distinguir los hongos venenosos, utilizaremos el conjunto de datos Mushroom de Jeff Schlimmer de la Universidad Carnegie Mellon. El conjunto de datos sin procesar está disponible de forma gratuita en el repositorio de aprendizaje automático de la UCI (http://archive.ics.uci.edu/ml).

El conjunto de datos incluye información sobre 8124 muestras de hongos de 23 especies de hongos con agallas (*guilled*) que figuran en la guía de campo de hongos norteamericanos de la Sociedad Audubon (1981). En la guía de campo, cada una de las especies de hongos se identifica como "definitivamente comestible", "definitivamente venenosa" o "probablemente venenosa y no se recomienda comerla". Para los fines de este conjunto de datos, el último grupo se combinó con el grupo "definitivamente venenosa" para formar dos clases: venenosa y no venenosa. El diccionario de datos disponible en el sitio web de la UCI describe las 22 características de las muestras de hongos, incluidas características como la forma del sombrero, el color del sombrero, el olor, el tamaño y el color de las agallas, la forma del tallo y el hábitat.

Esta práctica utiliza una versión ligeramente modificada de los datos de los hongos. Si planeas seguir el ejemplo, descarga el archivo mushrooms.csv adjunto.

Paso 2 - exploración y preparación de los datos

Comenzamos utilizando read.csv() para importar los datos para nuestro análisis. Dado que las 22 características y la clase de destino son nominales, estableceremos stringsAsFactors = TRUE para aprovechar la conversión automática de factores:

```
> mushrooms <- read.csv("mushrooms.csv", stringsAsFactors = TRUE)
```

La salida del comando str(mushrooms) indica que los datos contienen 8124 observaciones de 23 variables, tal como se describe en el diccionario de datos. Si bien la mayor parte de la salida de str() no tiene nada destacable, vale la pena mencionar una característica. ¿Observas algo peculiar sobre la variable veil_type en la siguiente línea?

```
> str(mushrooms)
$ veil_type               : Factor w/ 1 level "partial": 1 1 1 1 1 1 1 1 1 1 ...
```

Si te pareció extraño que un factor tenga solo un nivel, estás en lo correcto. El diccionario de datos enumera dos niveles para esta característica: parcial y universal; sin embargo, todos los ejemplos en nuestros datos se clasifican como parciales. Es probable que este elemento de datos se haya codificado incorrectamente de alguna manera. En cualquier caso, dado que el tipo de velo no varía entre muestras, no proporciona ninguna información útil para la predicción. Eliminaremos esta variable de nuestro análisis utilizando el siguiente comando:

```
> mushrooms$veil_type <- NULL
```

Al asignar NULL al vector veil_type, R elimina la característica del frame de datos de hongos.
Antes de continuar, deberíamos echar un vistazo rápido a la distribución de la variable de tipo de hongo en nuestro conjunto de datos:

```
> table(mushrooms$type)

  Edible poisonous
    4208      3916
```

Alrededor del 52 por ciento de las muestras de hongos son comestibles, mientras que el 48 por ciento son venenosas. Para los fines de este experimento, consideraremos que las 8214 muestras en los datos de hongos son un conjunto exhaustivo de todos los hongos silvestres posibles. Esta es una suposición importante porque significa que no necesitamos mantener algunas muestras fuera de los datos de entrenamiento para fines de prueba. No estamos

tratando de desarrollar reglas que cubran tipos imprevistos de hongos; simplemente estamos tratando de encontrar reglas que representen con precisión el conjunto completo de tipos de hongos conocidos. Por lo tanto, podemos construir y probar el modelo en los mismos datos.

Paso 3 - entrenamiento de un modelo en los datos

Si entrenáramos un clasificador ZeroR hipotético en estos datos, ¿qué predeciría? Dado que ZeroR ignora todas las características y simplemente predice el modo del objetivo, en lenguaje sencillo, su regla indicaría que “todos los hongos son comestibles”. Obviamente, este no es un clasificador muy útil porque dejaría a un recolector de hongos enfermo o muerto durante casi la mitad de las muestras de hongos. Nuestras reglas deberán funcionar mucho mejor que este punto de referencia para brindar asesoramiento seguro que pueda publicarse. Al mismo tiempo, necesitamos reglas simples que sean fáciles de recordar.

Dado que las reglas simples aún pueden ser útiles, veamos cómo se desempeña un aprendiz de reglas muy simple en los datos de hongos. Con este fin, aplicaremos el clasificador 1R, que identificará la característica única que es la más predictiva de la clase objetivo y usará esta característica para construir una regla.

Usaremos la implementación de 1R que se encuentra en el paquete OneR de Holger von Jouanne-Diedrich en la Universidad de Ciencias Aplicadas de Aschaffenburg. Este es un paquete relativamente nuevo, que implementa 1R en código R nativo para mayor velocidad y facilidad de uso. Si aún no tienes este paquete, puedes instalarlo con el comando install.packages("OneR") y cargarlo escribiendo library(OneR).

1R classification rule syntax

Using the `OneR()` function in the `OneR` package

Building the classifier:

```
m <- OneR(class ~ predictors, data = mydata)
```

- `class` is the column in the `mydata` data frame to be predicted
- `predictors` is an R formula specifying the features in the `mydata` data frame to use for prediction
- `data` is the data frame in which `class` and `predictors` can be found

The function will return a OneR model object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test)
```

- `m` is a model trained by the `OneR()` function
- `test` is a data frame containing test data with the same features as the training data used to build the classifier

The function will return a vector of predicted class values.

Example:

```
mushroom_classifier <- OneR(type ~ odor + cap_color,
                            data = mushroom_train)
mushroom_prediction <- predict(mushroom_classifier,
                               mushroom_test)
```

Figura 1: Sintaxis de la regla de clasificación de 1R.

Al igual que C5.0, la función OneR() utiliza la sintaxis de fórmula de R para especificar el modelo que se va a entrenar. El uso del tipo de fórmula ~ . con OneR() permite que nuestro primer aprendiz de reglas considere todas las características posibles en los datos de hongos al predecir el tipo de hongo:

```
> mushroom_1R <- OneR(type ~ ., data = mushrooms)
```

Para examinar las reglas que creó, podemos escribir el nombre del objeto clasificador:

```
> mushroom_1R
```

```
Call:
OneR.formula(formula = type ~ ., data = mushrooms)

Rules:
If odor = almond   then type = edible
If odor = anise    then type = edible
If odor = creosote then type = poisonous
If odor = fishy    then type = poisonous
If odor = foul     then type = poisonous
If odor = musty    then type = poisonous
If odor = none     then type = edible
If odor = pungent  then type = poisonous
If odor = spicy    then type = poisonous

Accuracy:
8004 of 8124 instances classified correctly (98.52%)
```

Al examinar el resultado, vemos que se seleccionó la característica de olor para la generación de reglas. Las categorías de olor, como almendra, anís, etc., especifican reglas sobre si es probable que el hongo sea comestible o venenoso. Por ejemplo, si el hongo huele a pescado, a humedad, acre, picante o a creosota, es probable que sea venenoso. Por otro lado, se predice que los hongos con olores más agradables, como la almendra y el anís, y aquellos que no tienen olor alguno, serán comestibles.

Para los propósitos de una guía de campo para la recolección de hongos, estas reglas podrían resumirse en una simple regla general: “si el hongo huele poco apetitoso, entonces es probable que sea venenoso”.

Paso 4 - evaluación del desempeño del modelo

La última línea del resultado indica que las reglas predicen correctamente la comestibilidad de 8004 de las 8124 muestras de hongos, o casi el 99 por ciento. Sin embargo, cualquier cosa que no sea perfecta corre el riesgo de envenenar a alguien si el modelo clasificara un hongo venenoso como comestible.

Para determinar si esto ocurrió, examinemos una matriz de confusión de los valores predichos versus los reales. Esto requiere que primero generemos las predicciones del modelo 1R y luego comparemos las predicciones con los valores reales:

```
> mushroom_1R_pred <- predict(mushroom_1R, mushrooms)
> table(actual = mushrooms$type, predicted = mushroom_1R_pred)

           predicted
actual      edible poisonous
  edible      4208         0
  poisonous    120      3796
```

Aquí, podemos ver dónde fallaron nuestras reglas. Las columnas de la tabla indican la comestibilidad predicha del hongo, mientras que las filas de la tabla dividen los 4208 hongos realmente comestibles y los 3916 hongos realmente venenosos. Al examinar la tabla, podemos ver que, aunque el clasificador 1R no clasificó ningún hongo comestible como venenoso, sí clasificó 120 hongos venenosos como comestibles, lo que constituye un error increíblemente peligroso.

Teniendo en cuenta que el aprendiz utilizó solo una característica, lo hizo razonablemente bien; si evitas los olores desagradables cuando buscas hongos, casi siempre evitarás una visita al hospital. Dicho esto, no basta con estar cerca de la realidad cuando hay vidas en juego, por no mencionar que el editor de la guía de campo podría no estar contento con la perspectiva de una demanda cuando sus lectores se enfermen. Veamos si podemos agregar algunas reglas más y desarrollar un clasificador aún mejor.

Paso 5 - mejorar el rendimiento del modelo

Para un aprendiz de reglas más sofisticado, utilizaremos JRip(), una implementación basada en Java del algoritmo RIPPER. La función JRip() está incluida en el paquete RWeka, que le da a R acceso a los algoritmos de aprendizaje automático en la aplicación de software Weka basada en Java de Ian H. Witten y Eibe Frank.

> Weka es una popular aplicación gráfica de código abierto y con todas las funciones para realizar tareas de minería de datos y aprendizaje automático, una de las primeras herramientas de este tipo. Para obtener más información sobre Weka, consulta http://www.cs.waikato.ac.nz/~ml/weka/.

El paquete RWeka depende del paquete rJava, que a su vez requiere que el kit de desarrollo de Java (JDK) esté instalado en la computadora host antes de la instalación. Este paquete se puede descargar desde https://www.java.com/ e instalar siguiendo las instrucciones específicas para tu plataforma. Después de instalar Java, usa el comando

install.packages("RWeka") para instalar RWeka y sus dependencias, luego carga el paquete RWeka usando el comando library(RWeka).

Java es un conjunto de herramientas de programación disponibles sin costo, que permiten el desarrollo y uso de aplicaciones multiplataforma como Weka. Si bien alguna vez se incluyó de manera predeterminada en muchas computadoras, esto ya no es así. Desafortunadamente, puede ser complicado de instalar, especialmente en computadoras Apple. Si tienes problemas, asegúrate de tener la última versión de Java. Además, en Microsoft Windows, es posible que debas configurar correctamente tus variables de entorno como JAVA_HOME y verificar la configuración de PATH (busca en la web para obtener más detalles). En computadoras macOS o Linux, también puedes intentar ejecutar R CMD javareconf desde una ventana de terminal y luego instalar el paquete R rJava desde la fuente usando el comando R install.packages("rJava", type ="source"). Si todo lo demás falla, puedes probar una cuenta gratuita de Posit Cloud (https://posit.cloud/), que ofrece un entorno RStudio en el que Java ya está instalado.

Con rJava y RWeka instalados, el proceso de entrenamiento de un modelo JRip() es muy similar al de entrenamiento de un modelo OneR(), como se muestra en el cuadro de sintaxis que aparece a continuación. Este es uno de los beneficios agradables de la interfaz de fórmulas de R: la sintaxis es consistente en todos los algoritmos, lo que facilita la comparación de una variedad de modelos.

RIPPER classification rule syntax

Using the `JRip()` function in the `RWeka` package

Building the classifier:

```
m <- JRip(class ~ predictors, data = mydata)
```

- `class` is the column in the `mydata` data frame to be predicted
- `predictors` is an R formula specifying the features in the `mydata` data frame to use for prediction
- `data` is the data frame in which `class` and `predictors` can be found

The function will return a RIPPER model object that can be used to make predictions.

Making predictions:

```
p <- predict(m, test)
```

- `m` is a model trained by the `JRip()` function
- `test` is a data frame containing test data with the same features as the training data used to build the classifier

The function will return a vector of predicted class values.

Example:

```
mushroom_classifier <- JRip(type ~ odor + cap_color,
                            data = mushroom_train)
mushroom_prediction <- predict(mushroom_classifier,
                               mushroom_test)
```

Figura 2: Sintaxis de la regla de clasificación de RIPPER.

Entrenemos al aprendiz de reglas JRip() como hicimos con OneR(), permitiéndole encontrar reglas entre todas las características disponibles:

> mushroom_JRip <- JRip(type ~ ., data = mushrooms)

Para examinar las reglas, escribe el nombre del clasificador:

> mushroom_JRip

```
JRIP rules:
===========

(odor = foul) => type=poisonous (2160.0/0.0)
(gill_size = narrow) and (gill_color = buff) => type=poisonous (1152.0/0.0)
(gill_size = narrow) and (odor = pungent) => type=poisonous (256.0/0.0)
(odor = creosote) => type=poisonous (192.0/0.0)
(spore_print_color = green) => type=poisonous (72.0/0.0)
(stalk_surface_below_ring = scaly) and (stalk_surface_above_ring = silky) => type=poisonous (68.0/0.0)
(habitat = leaves) and (cap_color = white) => type=poisonous (8.0/0.0)
(stalk_color_above_ring = yellow) => type=poisonous (8.0/0.0)
 => type=edible (4208.0/0.0)

Number of Rules : 9
```

El clasificador JRip() aprendió un total de nueve reglas a partir de los datos de los hongos.

Una forma sencilla de leer estas reglas es pensar en ellas como una lista de instrucciones if-else, similar a la lógica de programación. Las primeras tres reglas se podrían expresar como:

- Si el olor es desagradable, entonces el tipo de hongo es venenoso
- Si el tamaño de las agallas es angosto y el color de las agallas es beige, entonces el tipo de hongo es venenoso
- Si el tamaño de las agallas es angosto y el olor es penetrante, entonces el tipo de hongo es venenoso

Finalmente, la octava regla implica que cualquier muestra de hongo que no haya sido cubierta por las siete reglas anteriores es comestible. Siguiendo el ejemplo de nuestra lógica de programación, esto se puede leer como:

- De lo contrario, el hongo es comestible

Los números junto a cada regla indican la cantidad de instancias cubiertas por la regla y un recuento de instancias mal clasificadas. Cabe destacar que no hubo muestras de hongos mal clasificadas utilizando estas ocho reglas. Como resultado, la cantidad de instancias cubiertas por la última regla es exactamente igual a la cantidad de hongos comestibles en los datos (N = 4208).

La siguiente figura proporciona una ilustración aproximada de cómo se aplican las reglas a los datos de hongos. Si imaginas que el óvalo grande contiene todas las especies de hongos, el aprendiz de reglas identifica características, o conjuntos de características, que separan los

segmentos homogéneos del grupo más grande. Primero, el algoritmo encontró un grupo grande de hongos venenosos que se distinguen únicamente por su olor desagradable.
Luego, encontró grupos más pequeños y específicos de hongos venenosos. Al identificar reglas de cobertura para cada una de las variedades de hongos venenosos, todos los hongos restantes eran comestibles.

Gracias a la Madre Naturaleza, cada variedad de hongos era lo suficientemente única como para que el clasificador pudiera lograr una precisión del 100 por ciento.

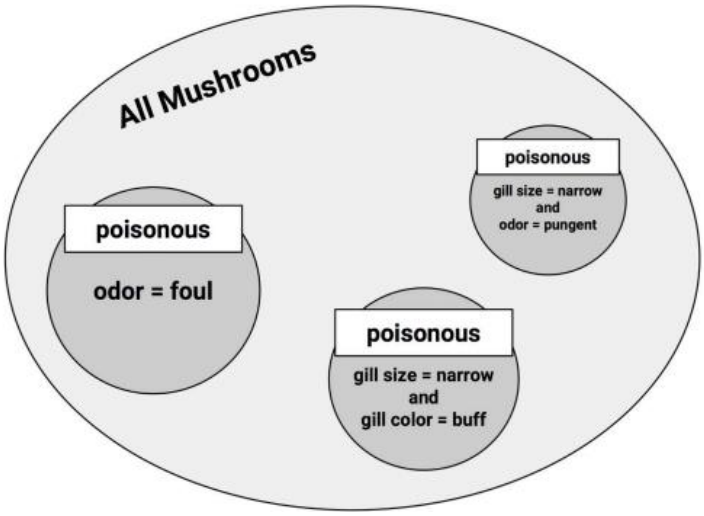

Figura 3: Un sofisticado algoritmo de aprendizaje de reglas identificó reglas para cubrir perfectamente todos los tipos de hongos venenosos.

Resumen del tema

Este tema abordó dos métodos de clasificación que utilizan los denominados algoritmos "codiciosos (o *ávidos*)" para dividir los datos según los valores de las características. Los árboles de decisión utilizan una estrategia de dividir y vencer (*conquistar*) para crear estructuras similares a diagramas de flujo, mientras que los aprendices de reglas separan y conquistan los datos para identificar reglas lógicas if-else. Ambos métodos producen modelos que se pueden interpretar sin conocimientos estadísticos.

Un algoritmo de árbol de decisión popular y altamente configurable es C5.0. Utilizamos el algoritmo C5.0 para crear un árbol para predecir si un solicitante de préstamo no pagará. Al utilizar opciones de aumento y errores sensibles al costo, pudimos mejorar nuestra precisión y evitar préstamos riesgosos que podrían costarle más dinero al banco.

También utilizamos dos aprendices de reglas, 1R y RIPPER, para desarrollar reglas para identificar hongos venenosos. El algoritmo 1R utilizó una sola característica para lograr una precisión del 99 por ciento en la identificación de muestras de hongos potencialmente fatales. Por otro lado, el conjunto de nueve reglas generadas por el algoritmo RIPPER más sofisticado identificó correctamente la comestibilidad de cada hongo.

Esto apenas roza la superficie de cómo se pueden utilizar los árboles y las reglas. En el siguiente tema, Pronóstico de datos numéricos: métodos de regresión, se describen técnicas conocidas como árboles de regresión y árboles de modelos, que utilizan árboles de decisión para la predicción numérica en lugar de la clasificación.

Más adelante veremos cómo se pueden utilizar las reglas de asociación (un pariente cercano de las reglas de clasificación) para identificar grupos de elementos en datos transaccionales. Por último, casi al final del curso revisaremos el tema, Cómo formar mejores aprendices, descubriremos cómo se puede mejorar el rendimiento de los árboles de decisión agrupándolos en un modelo conocido como bosque aleatorio, además de otras técnicas de modelado avanzadas que se basan en árboles de decisión.