Creación de un modelo de árbol de clasificación

Ahora que tenemos una mejor comprensión del concepto detrás de los árboles de decisión, pongámoslo en práctica usando R. En esta pequeña práctica, usaremos una función de árbol de decisión basada en el algoritmo CART para resolver el problema que a continuación se enuncia. Nuestro objetivo es construir un modelo que prediga si una solicitud de permiso pasará por un proceso de revisión acelerada o no en función de las características de la solicitud.

Prediciendo las decisiones de permiso de construcción

A medida que exploramos los métodos del árbol de decisiones en el curso, usaremos un conjunto de datos del Departamento de Construcción y Seguridad en Los Ángeles, California. Este conjunto de datos contiene información sobre las decisiones de permisos de construcción tomadas por el departamento e incluye información sobre la naturaleza del proyecto y si el permiso fue aprobado a través de un proceso acelerado de un día o si fue marcado para una revisión más extensa por parte del personal del departamento.

Los contratistas, por supuesto, preferirían que la mayor cantidad posible de proyectos de construcción se encaminaran a través del proceso acelerado. Nuestra tarea es analizar los datos para determinar si hay características específicas de una solicitud de permiso que hacen que sea más probable que pase por el proceso de revisión acelerada.

El conjunto de datos que utilizaremos está disponible en el archivo permits.csv. El conjunto de datos incluye una variedad de datos de permisos para nuestro análisis:

- status es el estado actual de la solicitud de permiso. Puede tomar valores como Finalizado, Emitido, Caducado y otros códigos de estado.
- permitType contiene la naturaleza de las mejoras solicitadas. Puede tomar valores como Eléctrico, Modificación/Reparación de Edificios, Plomería, etc.
- permitSubtype es el tipo de edificio afectado por el permiso. Puede tomar valores como 1 o 2 Vivienda unifamiliar, Comercial, Apartamento, etc.
- initiatingOffice es la ubicación de la oficina del departamento que inició la solicitud del permiso.
- ZIP es el código postal de la dirección de la propiedad.
- Valuation es el valor tasado de la propiedad según los registros de impuestos.
- floorArea son los pies cuadrados del área de piso de la propiedad.
- numberUnits es el número de unidades residenciales en una propiedad de varias viviendas.
- stories representa el número de pisos del edificio.
- contractorState es el estado en el que se encuentra el contratista que solicita el permiso, si corresponde.

- licenseType es un campo que categoriza el tipo de licencia que posee el contratista, si corresponde.
- zone es la categoría de zonificación de la propiedad.
- year y month son el año y mes en que se procesó la solicitud de permiso, respectivamente.
- permitCategory es la variable que queremos predecir. Contiene el valor Plan Check o el valor No Plan Check.

Dado el problema y los datos proporcionados, estas son algunas de las preguntas que debemos responder:

¿Qué variables predicen mejor si una solicitud de permiso se acelerará o se marcará para una revisión adicional?

¿Qué tan bien podemos predecir si una solicitud de permiso se marcará para su revisión o no en función de las variables de predicción disponibles para nosotros?

Primero importamos y previsualizamos nuestros datos. Use la versión de R, 4.0.4.

```
> library(tidyverse)
> permits <- read_csv("permits.csv", col_types = "fffffffnnnnfffff")
> glimpse(permits)
```

```
Rows: 971,486
Columns: 15
$ status           <fct> Permit Expired, Permit Finaled, Permit Finaled, Permit Finaled, Permit Finaled, Permit Finaled, Permit Finaled, P...
$ permitType       <fct> Plumbing, Plumbing, Plumbing, Plumbing, Electrical, Plumbing, Electrical, Electrical, HVAC, Plumbing, Plumbing, H...
$ permitSubtype    <fct> 1 or 2 Family Dwelling, 1 or 2 Family Dwelling, Commercial, 1 or 2 Family Dwelling, 1 or 2 Family Dwelling, 1 or ...
$ permitCategory   <fct> No Plan Check, No Plan Check, No Plan Check, No Plan Check, No Plan Check, No Plan Check, No Plan Check, No Plan ...
$ initiatingOffice <fct> INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INTERNET, INT...
$ ZIP              <fct> 90046, 90004, 90021, 90029, 90039, 90039, 91406, 90043, 91335, 91604, 90004, 91436, 90064, 90046, 90047, 90210, 9...
$ valuation        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 1000, NA, 40000000, 700, NA, NA, 3600, NA, 1650000, 5...
$ floorArea        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 10920, NA, NA, NA, NA...
$ numberUnits      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 8, NA, NA, NA, NA, NA...
$ stories          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 4, NA, NA, NA, NA, NA...
$ contractorState  <fct> CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, CA, NA, CA, CA, NA, NA, CA, CA, CA, C...
$ licenseType      <fct> C36, C36, C36, C36, C10, C36, C10, C10, C20, C36, C36, B, C36, B, C20, B, B, A, B, C10, NA, C39, C36, NA, NA, C36...
$ zone             <fct> R1-1, R2-1, M2-2D, R1-1-HPOZ, R1-1, R1-1VL, R1-1, R1-1, [Q]C2-2D-CDO, R1-1, R2-1, (Q)C4-1L, R1-1-O, R1-1, R1-1, R...
$ year             <fct> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013,...
$ month            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
```

Según el resultado del comando glimpse(), vemos que nuestro conjunto de datos consta de 971,486 instancias y 15 características. Como mencionamos al comienzo del documento, la variable que estamos tratando de predecir (nuestra clase) es permitCategory.

Nuestro resultado también muestra que tenemos una serie de valores perdidos para algunas de nuestras características (indicadas como NA). Obtengamos un resumen estadístico de nuestro conjunto de datos para comprender mejor los problemas que podemos tener con los datos faltantes, los valores atípicos y el ruido. Para hacer esto, usamos la función summary().

```
> summary(permits)
```

```
          status                  permitType                  permitSubtype          permitCategory   initiatingOffice       ZIP
 Permit Finaled:644876   Electrical       :274356   1 or 2 Family Dwelling:542641   No Plan Check:646957   METRO   :289327   90045  : 25362
 Issued        :196696   Bldg-Alter/Repair:222644   Commercial            :248659   Plan Check   :324489   VAN NUYS:283862   90049  : 21111
 Permit Expired: 54706   Plumbing         :185189   Apartment             :161264   NA's         :    40   INTERNET:251721   91331  : 17270
 CofO Issued   : 43917   HVAC             : 96490   Onsite                : 12536                           WEST LA : 76451   91367  : 16631
 Permit Closed : 12832   Fire Sprinkler   : 38404   Special Equipment     :  5299                           SOUTH LA: 37615   90026  : 16109
 (Other)       : 18419   (Other)          :154363   (Other)               :  1047                           (Other) : 32470   (Other):874902
 NA's          :    40   NA's             :    40   NA's                  :    40                           NA's    :    40   NA's   :   101
   valuation           floorArea        numberUnits        stories       contractorState   licenseType          zone             year
 Min.   :        0   Min.   :-154151   Min.   :-147.0   Min.   : -3.0   CA     :809934   B      :327643   R1-1   :179475   2018   :175912
 1st Qu.:     2100   1st Qu.:     32   1st Qu.:   0.0   1st Qu.:   0.0   TN     :  3670   C10    :175364   R3-1   : 51635   2017   :169791
 Median :     8000   Median :    500   Median :   0.0   Median :   1.0   GA     :  3666   C36    :125550   RS-1   : 41478   2016   :156165
 Mean   :   153474   Mean   :   3869   Mean   :   1.8   Mean   :   1.6   WA     :  3597   C20    : 73022   R2-1   : 26992   2015   :148824
 3rd Qu.:    30000   3rd Qu.:   2180   3rd Qu.:   1.0   3rd Qu.:   2.0   FL     :  3236   C16    : 37949   RA-1   : 25430   2014   :132524
 Max.   :525000000   Max.   :1788210   Max.   : 910.0   Max.   :4654.0   (Other): 13663   (Other): 98788   (Other):644096   (Other):188230
 NA's   :602487      NA's   :888698    NA's   :927409   NA's   :891769   NA's   :133720   NA's   :133170   NA's   :  2380   NA's   :    40
     month
 4      : 92875
 3      : 91715
 8      : 84622
 10     : 83117
 1      : 82425
 (Other):536692
 NA's   :    40
```

El resultado resumido muestra que nos faltan datos para la mayoría de nuestras características. Este no es un problema para los algoritmos de árboles de decisión. Son capaces de manejar muy bien los datos faltantes sin necesidad de imputación por nuestra parte. Esto se debe a que, durante el proceso de partición recursivo, las divisiones se realizan basándose únicamente en los valores observados de una variable. Si una observación tiene un valor perdido para la variable que se está considerando, simplemente se ignora.

También observamos en el resultado de resumen que algunas de las características numéricas, como valuation y floorArea, tienen una amplia gama de valores y posibles datos atípicos. Con algunos de los enfoques de aprendizaje automático que hemos cubierto hasta ahora, estos serían problemáticos y tendrían que solucionarse. Ese no es el caso de los árboles de decisión. Son capaces de manejar de forma robusta valores atípicos y datos ruidosos.

Como puedes comenzar a ver, los árboles de decisiones requieren poco de nosotros en términos de preparación de datos.
Sin embargo, nuestras estadísticas resumidas señalan algunas inconsistencias lógicas con algunos de nuestros valores de características. Por ejemplo, vemos que el valor mínimo de floorArea es –154,151.
Este no es un valor razonable para los pies cuadrados de un edificio. También vemos problemas similares con los valores mínimos para las características de valuation, numberUnits y stories.

Si bien estas inconsistencias no son un problema para el algoritmo del árbol de decisiones, conducirán a reglas de decisión ilógicas si el árbol se usara para la toma de decisiones comerciales. Para resolver estas inconsistencias, simplemente las tratamos como datos faltantes estableciendo sus valores en NA.

```
> permits <- permits %>%
+ mutate(valuation = ifelse(valuation < 1, NA, valuation)) %>%
+ mutate(floorArea = ifelse(floorArea < 1, NA, floorArea)) %>%
```

```
+ mutate(numberUnits = ifelse(numberUnits < 1, NA, numberUnits)) %>%
+ mutate(stories = ifelse(stories < 1, NA, stories))
```

Las estadísticas de resumen también muestran que tenemos un problema con el valor máximo de la característica stories. Una búsqueda rápida en línea revela que el edificio más alto de Los Ángeles (el Wilshire Grand Center) tiene solo 73 pisos. Por lo tanto, tratamos cualquier valor superior a 73 como datos faltantes estableciendo el valor en NA.

```
> permits <- permits %>%
+ mutate(stories = ifelse(stories > 73, NA, stories))
> summary(select(permits, valuation, floorArea, numberUnits, stories))
```

```
   valuation            floorArea         numberUnits        stories      
 Min.   :        1   Min.   :      1   Min.   :  1.0    Min.   : 1.0  
 1st Qu.:     3000   1st Qu.:    397   1st Qu.:  1.0    1st Qu.: 1.0  
 Median :     9801   Median :   1296   Median :  1.0    Median : 2.0  
 Mean   :   164723   Mean   :   5105   Mean   :  5.6    Mean   : 1.8  
 3rd Qu.:    32700   3rd Qu.:   2853   3rd Qu.:  1.0    3rd Qu.: 2.0  
 Max.   :525000000   Max.   :1788210   Max.   :910.0    Max.   :63.0  
 NA's   :627686      NA's   :908545    NA's   :954847   NA's   :914258
```

Los algoritmos del árbol de decisiones hacen un gran trabajo al seleccionar qué características son importantes para predecir el resultado final y cuáles no. Por lo tanto, la selección de características como paso de preparación de datos no es necesaria. Sin embargo, para simplificar nuestra ilustración, utilicemos únicamente las características permitType, permitSubtype e iniciatingOffice como predictores del resultado final, que está representado por la característica permitCategory. Usando el comando select() del paquete dplyr, reducimos nuestro conjunto de datos a estas cuatro características:

```
> library(dplyr)
> permits <- permits %>%
+ select(
+ permitType,
+ permitSubtype,
+ initiatingOffice,
+ permitCategory
+ )
```

División de datos

La siguiente etapa de nuestro proceso es dividir nuestros datos en conjuntos de prueba y entrenamiento. Con la función sample(), dividimos nuestro conjunto de datos dividiendo el 80 por ciento de los datos originales como datos de entrenamiento y el 20 por ciento restante como datos de prueba.

```
> set.seed(1234)
> sample_set <- sample(nrow(permits), round(nrow(permits)*.80), replace = FALSE)
> permits_train <- permits[sample_set, ]
> permits_test <- permits[-sample_set, ]
> round(prop.table(table(select(permits, permitCategory))),2)

No Plan Check    Plan Check
         0.67          0.33
> round(prop.table(table(select(permits_train, permitCategory))),2)

No Plan Check    Plan Check
         0.67          0.33
> round(prop.table(table(select(permits_test, permitCategory))),2)

No Plan Check    Plan Check
         0.67          0.33
```

Entrenamiento de un modelo

Ahora estamos listos para construir nuestro modelo. Como mencionamos anteriormente, usaremos el algoritmo CART para resolver nuestro problema de ejemplo. El algoritmo CART se implementa en R como parte del paquete rpart. Este paquete proporciona una función rpart() con un nombre similar, que usamos para entrenar nuestro modelo. Esta función toma tres argumentos principales. La primera es la fórmula de predicción, que especificamos como permitCategory ~. lo que significa que nuestro modelo debe usar todas las demás variables del conjunto de datos como predictores de la variable permitCategory. El segundo argumento es el método, que especificamos como clase. Esto significa que estamos construyendo un árbol de clasificación. El argumento final es el conjunto de datos de entrenamiento que se utilizará para construir el modelo.

```
> library(rpart)
> permits_mod <- rpart(
+ permitCategory ~ .,
+ method = "class",
+ data = permits_train
+ )
```

Evaluación del modelo

Ahora que hemos entrenado nuestro modelo de árbol de decisiones, visualicémoslo. Para hacerlo, usamos la función rpart.plot() del paquete rpart.plot de nombre similar. Ver la figura 1.

```
> library(rpart.plot)
> rpart.plot(permits_mod)
```

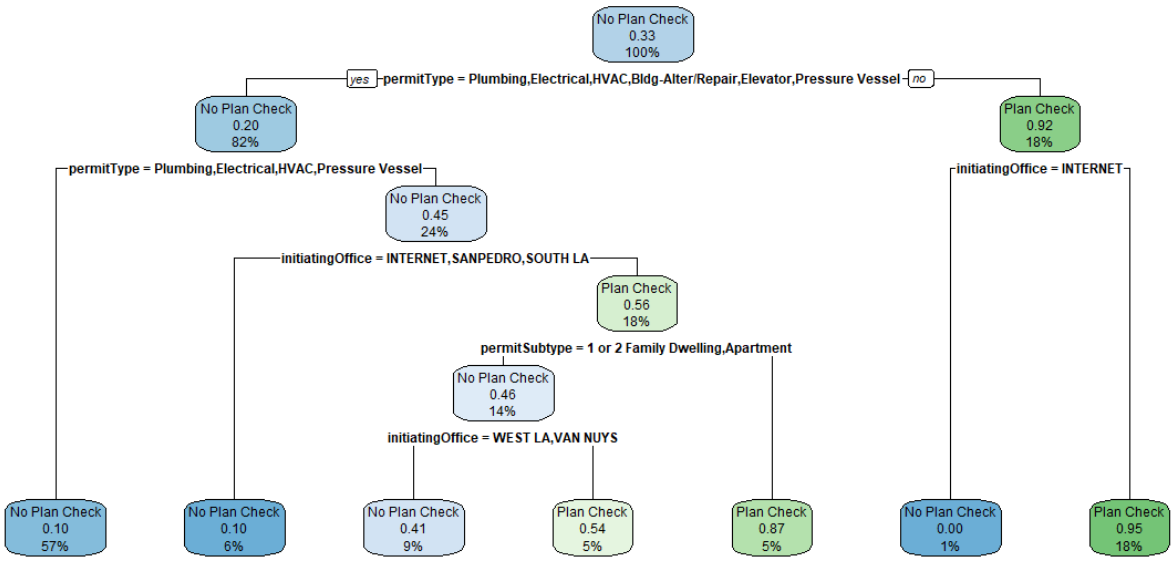

Figura 1: Visualización de un modelo de árbol de decisión usando la función rpart.plot() en R.

La estructura de un árbol de decisiones puede decirnos mucho sobre nuestros datos. Por ejemplo, el orden en el que se evalúan las características dentro del árbol es significativo. Nuestro árbol en particular comienza con una división por permitType en el nodo raíz. Esto nos dice que de las características que usamos en nuestro modelo, permitType es la más predictiva de nuestro resultado final. Cuanto más nos alejamos del nodo raíz, menos predictiva es una característica del resultado final.

Esto significa que después de permitType, initiatingOffice es la siguiente característica más predictiva, seguida de permitSubtype.

Además del orden en que se encuentran las características, los colores y las etiquetas de los nodos también son útiles para comprender nuestros datos. Recuerda que el nodo raíz de un árbol representa el conjunto de datos original antes de la primera división y que cada uno de los nodos subsiguientes (nodos de decisión y hoja) representa subparticiones del conjunto de datos original después de cada división anterior.

Al observar las etiquetas de cada nodo, aprendemos algo sobre cada una de las particiones que representan. Por ejemplo, las etiquetas del nodo raíz son: Sin verificación de plan (No Plan Check), 0.33 y 100%. La etiqueta inferior (100%) nos dice cuántos de nuestros datos originales representa la partición. El número del medio (0.33) nos dice la probabilidad de que una aplicación dentro de esta partición sea marcada para una revisión adicional (Plan Check). Debido a que esta probabilidad es menor que 0.5, el nodo se etiqueta como Sin verificación de plan (No Plan Check), que es la etiqueta superior del nodo.

Otra forma de leer esto es que, según todos nuestros datos, la probabilidad de que se acelere una nueva solicitud de permiso es del 67 por ciento, mientras que la probabilidad de que se marque para una revisión adicional es del 33 por ciento. Estos números son consistentes con los números de distribución de clases que obtuvimos anteriormente. Ahora, si seguimos las ramas más a la izquierda de nuestro árbol hasta el nodo de la hoja, aprendemos que la probabilidad de que se marque una nueva solicitud de permiso para una revisión adicional disminuye aún más, del 33 por ciento al 10 por ciento, si el permiso es para plomería, trabajo eléctrico, HVAC o con recipientes a presión.

Cuando se usa un árbol de decisión para la clasificación, los nodos y ramas del árbol ilustran el camino de decisión lógica que se puede tomar para clasificar datos previamente no clasificados. A medida que se encuentran nuevos datos, se evalúan contra criterios de división específicos en cada uno de los nodos de decisión, y se elige una ruta hasta que se encuentra un nodo terminal y se asigna una etiqueta. Las vías (o ramas) hacia la izquierda representan un acuerdo con los criterios de división, mientras que las vías hacia la derecha representan un desacuerdo con los criterios de división. Por ejemplo, la vía más a la derecha de nuestro árbol nos dice que si tenemos una nueva solicitud de permiso de construcción para la reparación de rociadores contra incendios (permitType = Fire Sprinkler) que no se inició a través de Internet (initiatingOffice! = INTERNET), entonces se marcará para revisión adicional (Plan Check).

Ahora, veamos cómo funciona nuestro modelo con este proceso en comparación con nuestra prueba. De manera similar a lo que hicimos anteriormente, pasamos el modelo (permits_mod) a la función predict() para clasificar los datos de prueba (permits_test), estableciendo el argumento de tipo en clase. Después de esto, creamos una matriz de confusión basada en nuestras predicciones y calculamos la precisión predictiva de nuestro modelo.

```
> permits_pred <- predict(permits_mod, permits_test, type = "class")
> permits_pred_table <- table(permits_test$permitCategory, permits_pred)
> permits_pred_table
              permits_pred
               No Plan Check Plan Check
  No Plan Check        121929       7357
  Plan Check            19054      45949
> sum(diag(permits_pred_table)) / nrow(permits_test)
[1] 0.8640278
```

Los resultados muestran que nuestro modelo tiene una precisión predictiva del 86.4 por ciento frente a los datos de la prueba. ¿Cómo podemos mejorar este desempeño? Hay varias cosas que me vienen a la mente. La primera es recordar que los algoritmos del árbol de decisión no son paramétricos. El rendimiento de los modelos no paramétricos puede mejorar

a medida que se consideran datos adicionales. Por lo tanto, podríamos ajustar la proporción de entrenamiento para probar los datos para nuestros datos existentes o podríamos recopilar datos adicionales. El segundo enfoque es considerar características adicionales para el modelo. Recuerda que usamos solo cuatro características en este modelo.