Deusto

Aprobación de Créditos

Aprendizaje automático basado en Árboles de Decisión

Descripción del problema

- ¿Podemos predecir los préstamos que se van a conceder según algunas características de quien lo solicita?
  - El objetivo de este proyecto es utilizar la información financiera y otros datos asociados para determinar si un banco daría un préstamo a un solicitante.
  - Utilizaremos como referencia un dataset disponible en plataforma **Kaggle** (origen).

Descripción del DataSet

- loan_id: identificador del ejemplo
- no_of_dependents: número de personas a cargo del cliente [0, …, 5]
- education: nivel de educación [Graduate | Not Graduate]
- self_employed: trabajador autónomo [Yes | No]
- income_annum: ingresos anuales [200k, …, 9.90m]
- loan_amount: cantidad de préstamo [300k, …, 39.5m]
- loan_term: años de préstamo [2, .., 20]
- cibil_score: puntuación de crédito [300, …, 900]
- residential_assets_value: valor de activos residenciales
- commercial_assets_value: valor de activos comerciales
- luxury_assets_value: valor de activos de lujo
- bank_asset_value: valor de activos bancarios
- **loan_status (atributo a predecir): [Approved | Rejected]**

Se pide

- Elabora scripts en R para predecir la aprobación (o no) de préstamos
  - Procesa el dataset (reemplaza los datos que falten con el valor más común de esa columna).
  - Genera 10 árboles de decisión y calcula su precisión:
    - Divide los datos en entrenamiento (75%) y test (25%)
    - Limita la profundidad a 5
    - Realiza predicciones con los modelos
    - Calcula (y guarda) varias métricas de precisión: precisión total, precisión para cada clase, precisión considerando sólo la educación y precisión considerando sólo si es un trabajador autónomo

Se pide

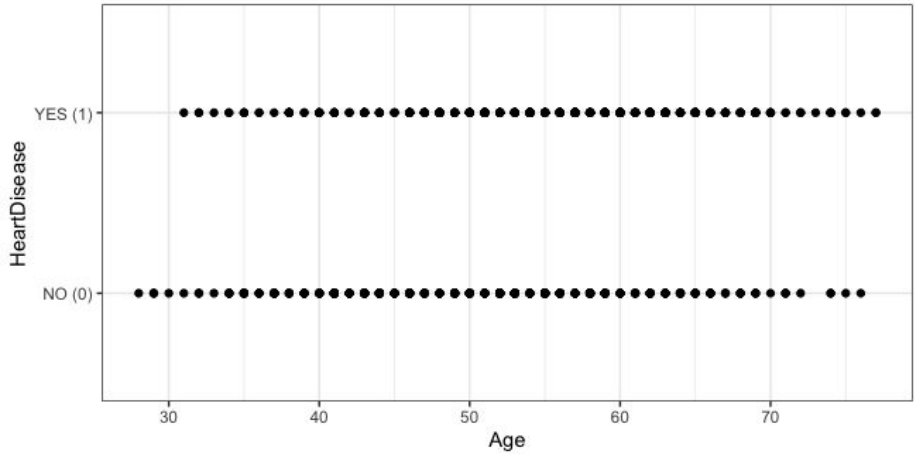
- Elabora scripts en R para predecir la aprobación (o no) de préstamos
  - Imprime la imagen del árbol con mayor precisión y genera las reglas que definen su comportamiento.
  - Identifica los 5 atributos más relevantes (por código).
  - Imprime una tabla con los resultados de las 10 ejecuciones y una fila ‘media’.
  - Añade el código necesario para responder a las cuestiones finales.

Analiza los datos

- Verifica la idoneidad del uso de todos los atributos del dataset
  - ¿Hay datos numéricos? ¿Pueden convertirse en rangos?
  - ¿El valor de un atributo es único para cada instancia?

Analiza los datos - Rangos

- Visualizar la relación entre los datos numéricos y el resultado
  - `ggplot(data, aes(x=Age, y=HeartDisease)) + geom_point()`

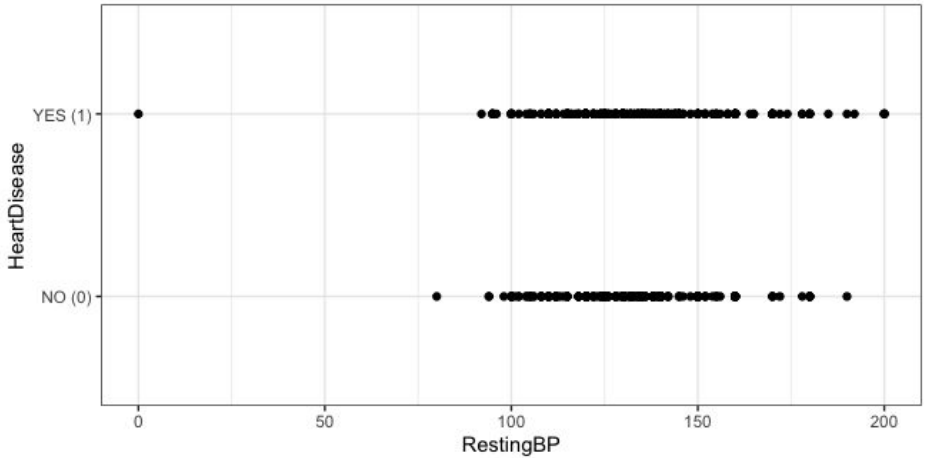


Este gráfico nos puede ayudar a establecer rangos.

En este caso, podrían definirse rangos de tamaño uniforme

Analiza los datos - Rangos

- Visualizar la relación entre los datos numéricos y el resultado
  - `ggplot(data, aes(x=RestingBP, y=HeartDisease)) + geom_point()`



Este gráfico nos puede ayudar a establecer rangos.

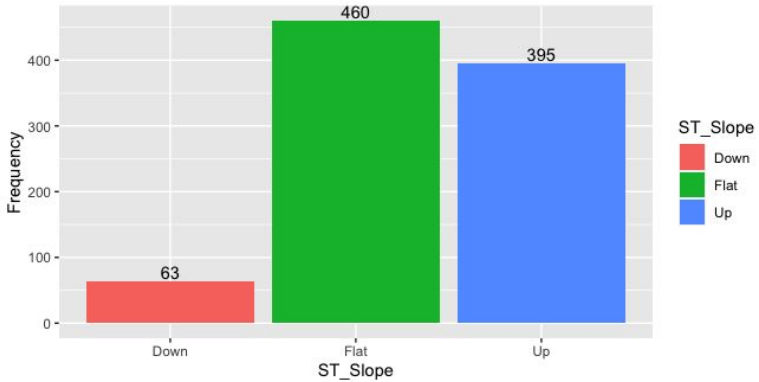
En este caso, el rango no debe ser uniforme ya que los ejemplos se concentran entre 100 y 175.

Analiza los datos - Rangos

- Se pueden crear rangos con umbrales constantes
  - `data$Cholesterol <- ifelse(data$Cholesterol <= 200, "Normal", "High")`
    - Esta sentencia modifica los valores numéricos asignando cadenas de texto diferentes para valores <= 200 y > 200.
    - La función `ifelse()` puede anidarse para crear definir más valores.
- Se pueden crear rangos con un tamaño uniforme
  - `data$Age <- cut(data$Age, breaks = 4)`
    - Esta sentencia particiona la columna Age en 4 rangos de tamaño uniforme en base a los valores máximo y mínimo.

Analiza los datos - Frecuencia

- Visualiza la distribución de las instancias para cada atributo
  - ```
    ggplot(data, aes(x=ST_Slope, fill=ST_Slope)) + geom_bar() +
        geom_text(stat="count", aes(label=..count..), vjust=-0.25) +
        labs(x = "ST_Slope", y = "Frequency")
    ```



Si la frecuencia de un valor es muy baja, puede distorsionar el modelo. Incluso generar errores durante la validación del modelo.
    ```

Crea un árbol de decisión

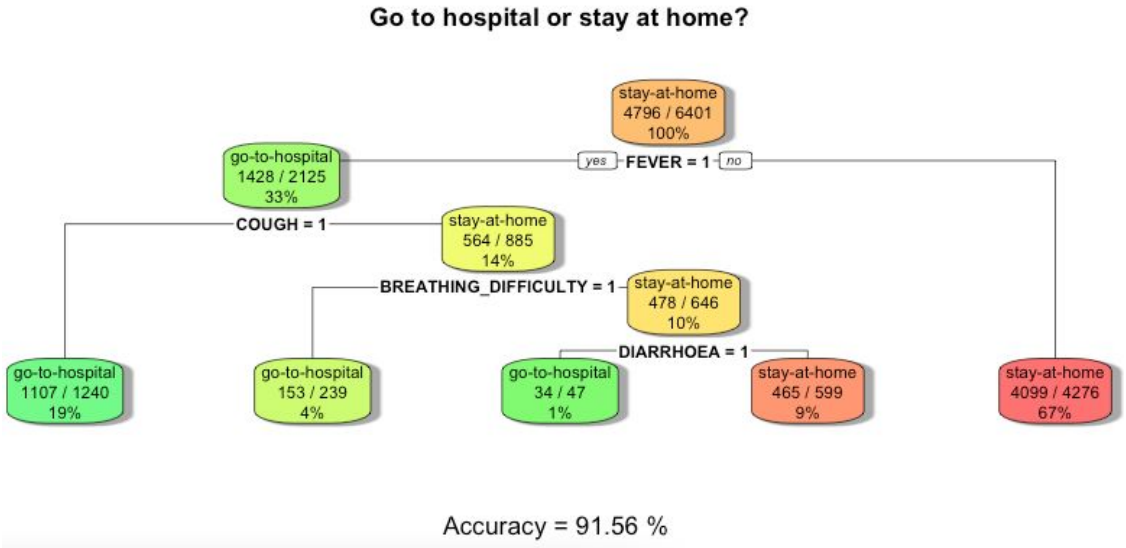
- Divide el dataset en entrenamiento (75%) y test (25%)

```
createDataPartition(y = data$Target, p = train_%, list = FALSE)
```

- Genera el árbol de decisión usando la función rpart():
  - `rpart(formula = ¿?, data = training_data)`
    - `formula=target~.` (usa todos los atributos del data.frame)
    - `formula=target~Att-1+Att-2+...+Att-N` (usa sólo algún atributo)
- Valida el modelo: `predict(model, test_data)`
  - Calcula la precisión del árbol usando la Matriz de Confusión

Crea un árbol de decisión

- Dibuja el árbol usando la función [rpart.plot](): `rpart.plot(model)`



Crea un árbol de decisión

- Genera las reglas usando la función [rpart.rules](): `rpart.rules(model)`

```
TARGET      go- sta                                                    cover
Go-to-hosp  [.90 .10] IF FEVER = 0 && BLOOD_EXPECTORATION = 1          0%
Go-to-hosp  [.89 .11] IF FEVER = 1 && COUGH = 1                        20%
   ...                    ...
Stay-at-h   [.04 .96] IF FEVER = 0 && BLOOD_EXPECTORATION = 0          66%
```

Responde a las siguientes preguntas

- Usando el mejor modelo, añade el código que permita responder a las siguientes cuestiones (justifica en la documentación cómo respondes a cada cuestión):
  - Generar una representación gráfica sobre porcentajes de valores actuales y predichos por el modelo por número de dependientes
  - Identifica clientes con el préstamo rechazado tal que, si cambiamos su educación de 'Not graduated' a 'Graduated', les hubiesen concedido el crédito.
  - Para cada cliente con el crédito rechazado, identifica los ingresos mínimos con los que le habrían aprobado el préstamo.
  - Para cada cliente con el crédito aprobado, identifica la máxima cantidad de préstamo que podría haber pedido, y aún haber sido aprobado.

Entrega

- Entrega el trabajo a través de la **tarea de ALUD**
  - Fecha de entrega: 12 de mayo
  - Completa el cuestionario de tiempos y dificultades
  - Formato: fichero .ZIP
- Evaluación - 10%
  - Ejecución correcta y libre de errores - 5%
  - Documentación de análisis de los resultados - 5%
- Esfuerzo individual
  - 10h. por persona