TP Machine Learning

Introducción a la Inteligencia Artificial

Agustín Díaz

Aldana Zarate

Dataset: Glass

Durante este trabajo se estará trabajando con diferentes tipos de vidrios. La cantidad de instancias o muestras de ellos son 214, sin valores nulos. Tendremos 10 atributos en total, de los cuales solo 9 son significativos ya que el primero es un Id#, y este solo es un número parte de una secuencia ordenada creciente de números naturales. Los atributos de los vidrios son:

- RI: índice de refracción
- Na: sodio
- Mg: magnesio
- Al: aluminio
- Si: silicona
- K: potasio
- Ca: calcio
- Ba: boro
- Fe: hierro

Cada uno de los vidrios puede ser de 7 clases diferentes. En la muestra estudiada, la distribución es la siguiente:

Tipo de vidrio	Cantidad de instancias
Building windows float processed	70
Building windows non-float processed	76
Vehicle windows float processed	17
Vehicle windows non-float processed	0
Containers	13
Tableware	9
Headlamps	29

Observando esta distribución, podemos ver que las primeras dos clases tienen considerablemente mayor representación que el resto, y que la cuarta clase no tiene representación en absoluto. Esta distribución de ejemplares con tales extremos muy probablemente afecte los resultados del modelo elegido.

Modelo

Entrenaremos un árbol de decisión para que sea un modelo clasificador para los distintos tipos de vidrio en base a los atributos de los mismos.

Entrenamiento

Del dataset total se utilizaron 10/12 para entrenamiento y 2/12 para testing, particionados de manera aleatoria. Se utilizó el método de *k-fold cross validation* con k=10 para determinar los mejores parámetros para entrenar a nuestro modelo completo. Se usó una proporción 80/20 para elegir el conjunto de entrenamiento (y de este analizar algunas métricas notables) y de testeo de entre estos conjuntos originales.

En el script R adjunto se puede ver que se hicieron pruebas de podado variando el parámetro cp, obteniendo métricas sobre la accuracy de los modelos resultantes. Luego se agrupan las métricas en base al cp y se busca la mejor accuracy para obtener el parámetro definitivo para el modelo completo. Tomamos esto como referencia ya que no es el objetivo principal el clasificar correctamente alguna clase en particular.

Luego de esta etapa de preprocesamiento, los parámetros para el podado con mejores métricas que se obtuvieron son:

Método	cp
Gini	0.05
Information	0.02

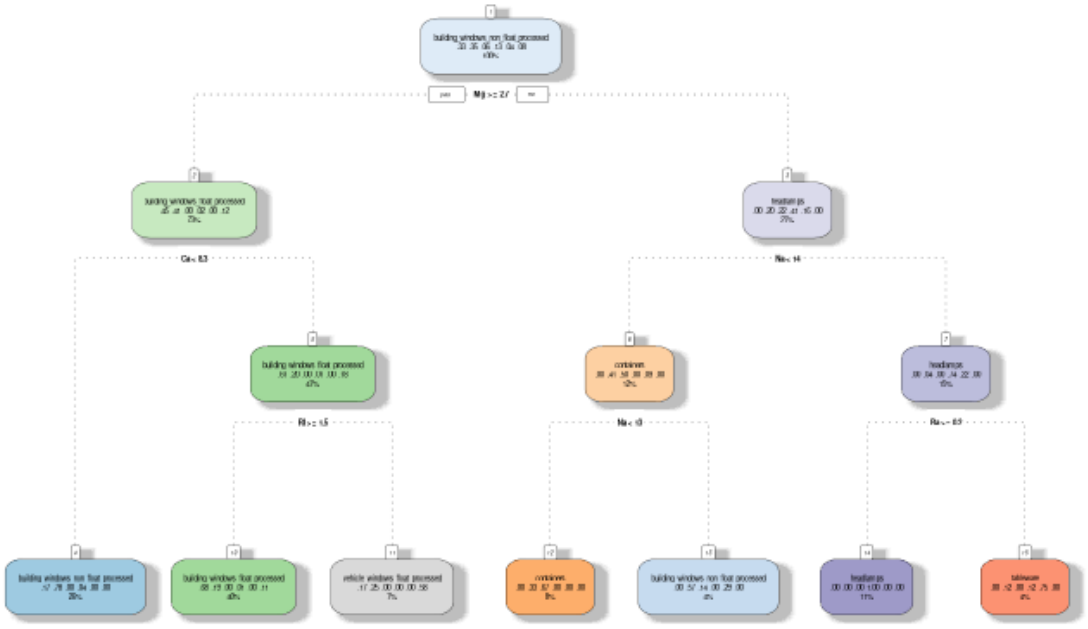
Para entrenar todos los modelos utilizamos todos los atributos disponibles.

Método: Ganancia de información

Ejecutando la línea:

```
fancyRpartPlot(fittedPrunedModel, caption = NULL)
```

Se obtiene el árbol de decisión siguiente:



En el archivo **arbol_information.pdf** se puede observar el árbol de decisión generado con mayor detalle. La imagen anterior es solo para tener una referencia. Además, ejecutando `summary(fittedPrunedModel)` se puede observar cómo se fue generando el árbol.

Veamos distintas métricas sobre las predicciones con los datos de testing

Accuracy

Obtuvimos una accuracy general de 0.54285714, la cual no es aceptable

Precision y Recall

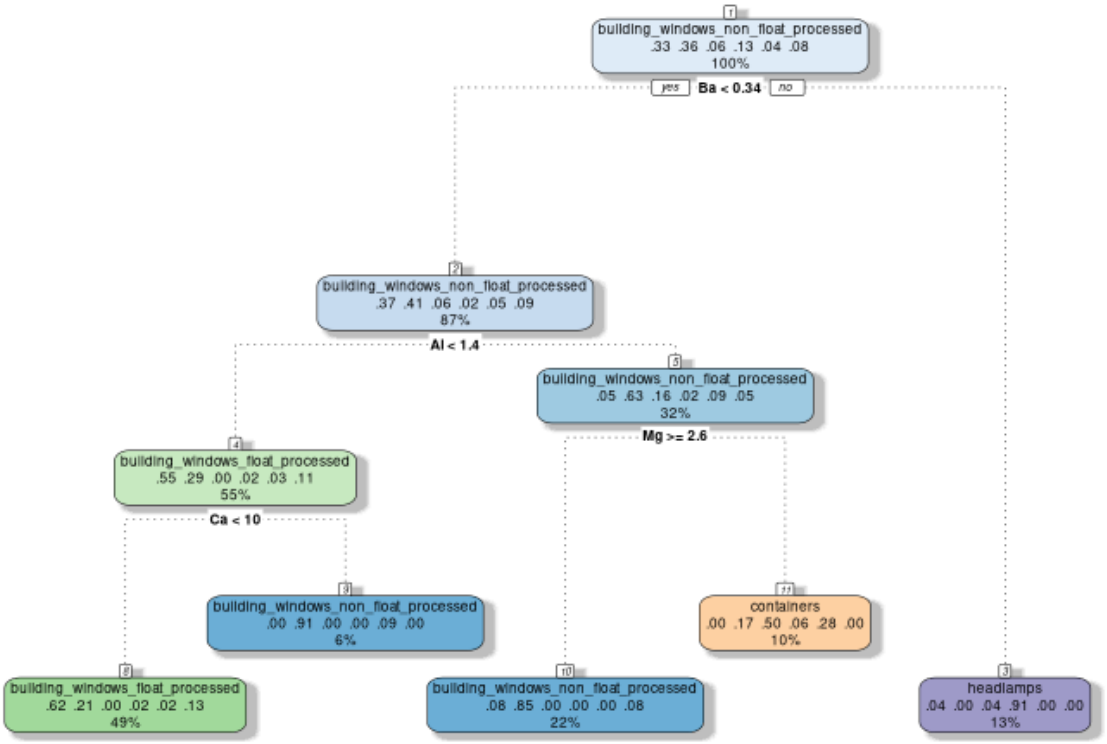
Obtuvimos una precision promedio de 0.629798 y recall promedio de 0.6678322, lo cual es ligeramente aceptable.

Specifity

Obtuvimos una specifity promedio de 0.8945707, la cual es bastante aceptable.

Método: Gini

En el archivo **arbol_gini.pdf** se puede observar el árbol de decisión generado con mayor detalle. La imagen a continuación es solo para tener una referencia:



Veamos distintas métricas sobre las predicciones con los datos de testing

Accuracy

Con este modelo obtenemos una accuracy en general de 0.6388888889, lo cual no nos resulta muy aceptable.

Precision y Recall

En promedio, nos queda una precision de 0.4285714, lo cual dado en términos generales tampoco es un número muy aceptable.

Por el lado del recall, nos queda en promedio 0.5, lo cual tampoco es muy destacable.

Specifity

La specifity promedio nos dio del 0.9150782, lo cual es un valor muy aceptable.

Conclusiones

Dada la naturaleza de la muestra estudiada, los resultados fueron apenas aceptables.

Comparemos los métodos:

	Accuracy	Precision	Recall	Specifity
Ganancia de Información	0.54285714	0.629798	0.6678322	0.8945707
Gini	0.6388888889	0.4285714	0.5	0.9150782

Si analizamos la precisión en ambos métodos, vemos que usando el método de ganancia de información se destacan las clases tableware y headlamps, y usando Gini, las clases headlamps y building_windows_float_processed.

Normalmente los modelos tienen prioridades sobre las métricas que le dan más relevancia al momento de elegir el modelo final. Por ejemplo, si estamos haciendo un diagnóstico de una enfermedad grave, es peor tener falsos negativos que falsos positivos. Dado que el objetivo no es clasificar alguna clase en particular, podríamos quedarnos con el método de Gini, el cual nos da mayor accuracy. Pero, algo llamativo es que comparando las demás métricas (salvando specifity que son muy similares), resalta más el método de ganancia de información.

Como conclusión final, ambos modelos no alcanzan las expectativas necesarias para ser modelos de clasificación confiables. Se podrían buscar distintas alternativas para conseguir un modelo funcional, aquí enumeramos algunas:

- Conseguir más cantidad de datos clasificados: es muy costoso ya que requiere mucho trabajo manual, de maquinaria y profesionales
- Generar más cantidad de datos clasificados: hay que ver el caso particular de este área de aplicación sobre el estudio de vidrio, pero muchas veces se pueden encontrar maneras de amplificar nuestro conjunto de datos (por ejemplo, si tenemos un clasificador de perros o gatos en base a foto, podríamos duplicar nuestro conjunto de datos espejando las fotos)
- Conocimiento de dominio: se pueden aprovechar estudios profesionales sobre patrones físicos, químicos o empíricos que pueden aportar a entrenar al clasificador.
- Transferencia del conocimiento: si ya existe algún modelo similar, depende del modelo, se podrá aplicar una transferencia de conocimiento para aprovechar el entrenamiento del modelo similar. Esto es muy común para redes neuronales.
- Probar otros modelos y arquitecturas: se pueden probar otros modelos como redes neuronales, support vector machine, random forest, K-Nearest-Neighbour; cada uno con sus flavours o arquitecturas particulares

Debido al pequeño tamaño del dataset puede darse el caso de overfitting, ya que es muy fácil memorizar conjuntos de datos pequeños. Sin embargo es poco probable por las siguientes consideraciones:

- Se utilizó k-fold cross validation para obtener los parámetros de podado

- Los modelos entrenados con distintos parámetros de podados cercanos a 0 (es decir sin podar) no generan cambio significativo en las métricas generales
- El modelo está beneficiado de un podado para eliminar las divisiones de los últimos niveles, que harían que el modelo memorice en vez de que aprenda a generalizar sobre los datos.