Instituto Tecnológico y de Estudios Superiores de Monterrey.

Campus Estado de México

Momento de Retroalimentación: Módulo 2 Implementación de una técnica de aprendizaje máquina sin el uso de un framework. (Portafolio Implementación)

ID3

Profesor

Jorge Adolfo Ramírez Uresti

Integrantes

Sebastián Espinoza Farías……. ITC | A01750311

Fecha de entrega: 05-09-2015

Para la primera entrega del portafolio de implementación se desarrolló el algoritmo ID3, sobre árboles de decisión. La implementación se realizó utilizando únicamente numpy y pandas.

El algoritmo fue probado con tres datasets distintos:

Tennis: Dataset utilizado en clase sobre la decisión de jugar un partido de tenis. Al ser muy pequeño y sencillo, el algoritmo mostró síntomas de overfitting, por lo que fue descartado como prueba final. Sin embargo, resultó útil para validar las primeras iteraciones del código.

Mushrooms: Dataset más complejo, con mayor número de entradas. Aunque al inicio parecía una mejor opción, también presentó overfitting evidente, por lo que no permitió evaluar adecuadamente la capacidad del algoritmo.

Cars: Dataset proveniente del repositorio UCI Machine Learning Repository, con mayor diversidad de clases en el atributo objetivo. Este conjunto no presentó overfitting evidente y mostró un desempeño más equilibrado, siendo finalmente el seleccionado para la entrega.

El algoritmo realiza de manera automática la división del dataset en entrenamiento y prueba, reservando un 20% de los datos para evaluación y un 80% para entrenamiento.

Una vez hecha la división, el programa genera dos archivos CSV:

- <dataset>_train.csv: contiene los datos utilizados para entrenar el árbol.
- <dataset>_test.csv: contiene los datos reservados para probar el modelo.

Esta funcionalidad permite visualizar claramente qué datos fueron usados como train y test.

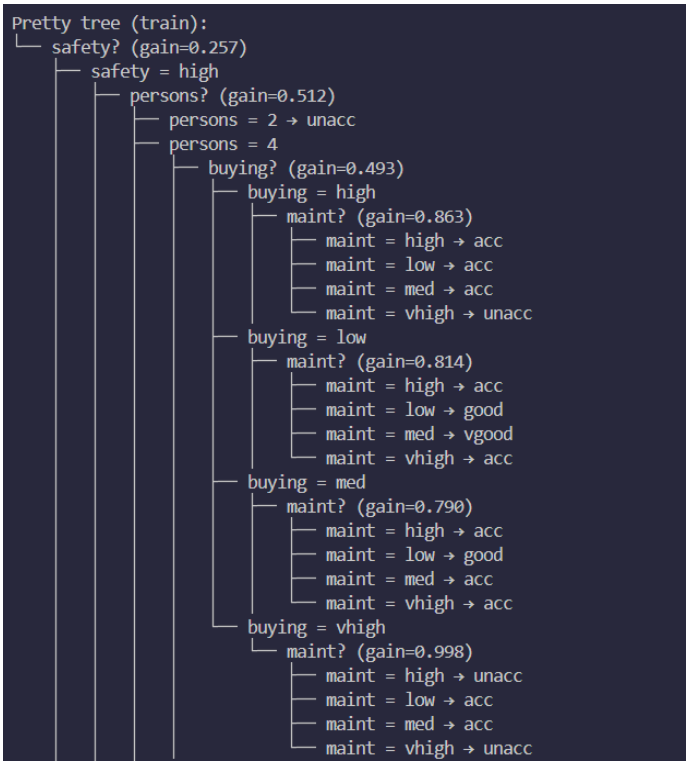
Resultados

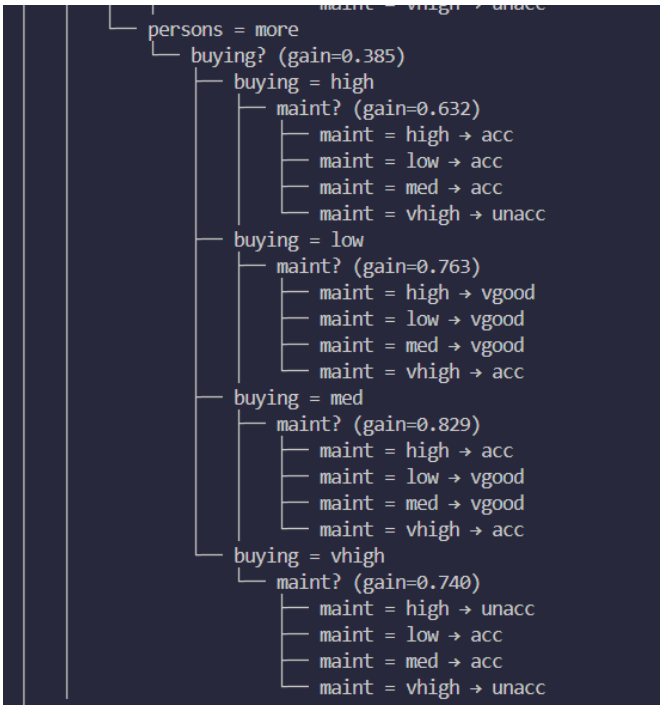
Cuando el algoritmo finaliza su ejecución, se imprime en la terminal información de utilidad para saber qué fue lo que el algoritmo realizó:

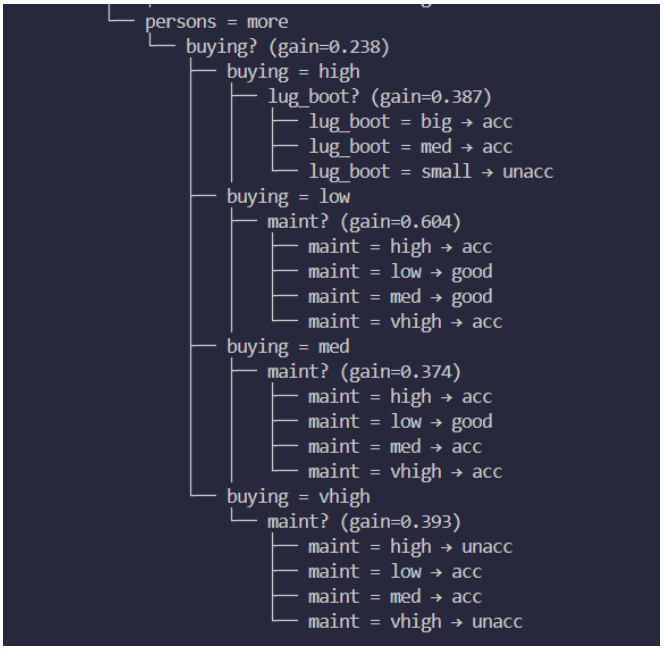
```
Saved splits:
  Train -> cars_train.csv  (rows: 1382)
  Test  -> cars_test.csv   (rows: 346)

Dataset: cars.csv
Train class counts:
 class
unacc    968
acc      307
good      55
vgood     52
Name: count, dtype: int64
Test class counts:
 class
unacc    242
acc       77
good      14
vgood     13
Name: count, dtype: int64
```

En la primera parte se indican los archivos generados automáticamente (cars_train.csv y cars_test.csv en este caso), junto con el número de filas que contiene cada uno. De esta manera se puede verificar la división del dataset en 80% para entrenamiento y 20% para prueba.

Posteriormente, se muestran las distribuciones de clases en los conjuntos de entrenamiento y prueba. Esta información es relevante porque confirma que el algoritmo mantiene la proporción de las clases al momento de dividir los datos (stratified split), lo que asegura que tanto el modelo de entrenamiento como la evaluación final se realizan sobre datos representativos.


Pretty tree (train):
safety? (gain=0.257)
safety = high
persons? (gain=0.512)
persons = 2 → unacc
persons = 4
buying? (gain=0.493)
buying = high
maint? (gain=0.863)
maint = high → acc
maint = low → acc
maint = med → acc
maint = vhigh → unacc
buying = low
maint? (gain=0.814)
maint = high → acc
maint = low → good
maint = med → vgood
maint = vhigh → acc
buying = med
maint? (gain=0.790)
maint = high → acc
maint = low → good
maint = med → acc
maint = vhigh → acc
buying = vhigh
maint? (gain=0.998)
maint = high → unacc
maint = low → acc
maint = med → acc
maint = vhigh → unacc


persons = more
buying? (gain=0.385)
buying = high
maint? (gain=0.632)
maint = high → acc
maint = low → acc
maint = med → acc
maint = vhigh → unacc
buying = low
maint? (gain=0.763)
maint = high → vgood
maint = low → vgood
maint = med → vgood
maint = vhigh → acc
buying = med
maint? (gain=0.829)
maint = high → acc
maint = low → vgood
maint = med → vgood
maint = vhigh → acc
buying = vhigh
maint? (gain=0.740)
maint = high → unacc
maint = low → acc
maint = med → acc
maint = vhigh → unacc



Se imprime también el árbol de decisión creado, lo que da una buena idea de que fue lo que interpretó y los valores de ganancia que hicieron que el árbol decidiera de esa manera.

```
Accuracy
  Train: 0.9008683068017366
  Test : 0.8930635838150289

Confusion Matrix (Test)
           pred_acc  pred_good  pred_unacc  pred_vgood
true_acc         69          4           4           0
true_good         3          7           0           4
true_unacc       12          2         226           2
true_vgood        4          2           0           7

Per-class Precision, Recall, F1 (Test)
Class     Precision   Recall      F1-score
-------------------------------------------
acc       0.784       0.896       0.836
good      0.467       0.500       0.483
unacc     0.983       0.934       0.958
vgood     0.538       0.538       0.538
```

Por último, imprime el accuracy del algoritmo y también las métricas adicionales:

- **Accuracy:**

Indica el porcentaje de instancias correctamente clasificadas sobre el total.

En este caso, el modelo alcanza aproximadamente 90% en entrenamiento y 89% en prueba, lo que sugiere un buen desempeño y una generalización adecuada, a diferencia de los otros dos datasets, que siempre salen con 100% de accuracy.

- **Matriz de confusión:**

Muestra en forma tabular cuántas instancias de cada clase fueron correctamente clasificadas (diagonal principal) y cuántas se confundieron con otras clases (valores fuera de la diagonal).

Esto permite identificar qué clases son más difíciles de predecir. Por ejemplo, en la tabla se observa que la clase “unacc” es clasificada con gran precisión, mientras que las clases “good” y “vgood” presentan más errores.

Métricas por clase (Precision, Recall, F1-score):

Precisión: Indica qué proporción de las predicciones asignadas a una clase son realmente correctas. Es un valor cercano al accuracy, pero un poco más específico, ya que se enfoca en la calidad de las predicciones positivas realizadas por el modelo, lo que es de utilidad para su evaluación de desempeño.

Recall: Mide qué proporción de los verdaderos elementos de una clase fueron identificados correctamente por el algoritmo. Esta métrica permite detectar qué tan bien el modelo reconoce una clase en particular y qué tantos falsos negativos se generan.

F1-score: Es la media entre precisión y recall. Combina ambas métricas en un solo valor balanceado, especialmente útil cuando existe un desbalance de clases, ya que ofrece una visión más justa del rendimiento del modelo en lugar de considerar solo una métrica individual.

Estas métricas fueron escogidas porque permiten evaluar el desempeño del algoritmo en cada clase por separado y no únicamente en promedio.

Análisis y conclusión final

Al implementar el algoritmo ID3 y probarlo con diferentes datasets, quedó claro que el tamaño y la complejidad de los datos influyen directamente en el desempeño y en el riesgo de que suceda el overfitting. Con el dataset de Tennis, el modelo siempre alcanzaba un 100% de accuracy, lo cual a primera vista podría parecer positivo, pero en realidad evidenciaba que el árbol era demasiado simple y estaba aprendiendo de memoria los pocos ejemplos disponibles.

El dataset de Mushrooms representó un reto mayor porque tenía más registros, pero aun así mostró síntomas de overfitting. Esto reforzó la idea de que no solo importa la cantidad de datos, sino también la variedad de clases y la forma en que están distribuidos.

Finalmente, el dataset de Cars resultó ser el más adecuado para la entrega. Gracias a que contiene más clases en el atributo objetivo, esto permitió evaluar de manera más realista la capacidad del algoritmo. Los resultados obtenidos, con un accuracy cercano al 90% tanto en entrenamiento como en prueba, muestran que el modelo no solo aprendió las reglas, sino que también logró aplicarlas correctamente a ejemplos nuevos, pero si cometió algunos errores, lo que me parece que es un modelo mucho más natural.

En conclusión, la implementación del ID3 fue una buena experiencia para entender cómo funciona un árbol de decisión desde cero. Más allá de los resultados, lo más valioso fue observar cómo se construye el árbol paso a paso, cómo se calculan las ganancias de información y, sobre todo, cómo distintas configuraciones de datos afectan el rendimiento del modelo, lo que me ayudó bastante a comprender el tema.

Dataset utilizado para el entrenamiento usando cars.csv

https://github.com/Sebastian-Espinoza-25/Machine-Learning-Implementation/blob/main/cars_train.csv

Dataset utilizado para la evaluación usando cars.csv

https://github.com/Sebastian-Espinoza-25/Machine-Learning-Implementation/blob/main/cars_test.csv