**Nombre del Algoritmo:**  C4.5 algorithm

**Descripción del Problema Computacional para la cual el algoritmo es una solución.**

**Delimitación del Problema:**

El algoritmo C4.5 resuelve problemas de clasificación en aprendizaje supervisado, donde el objetivo es predecir una etiqueta de clase para instancias basadas en un conjunto de características. C4.5 extiende el algoritmo ID3 al manejar valores continuos, datos incompletos y realizar poda en los árboles generados para mejorar la generalización.

**Comprensión del Problema:**

El problema abordado es cómo construir un modelo que pueda categorizar datos desconocidos dividiendo el espacio de características en subespacios homogéneos en términos de la clase objetivo. C4.5 utiliza un enfoque basado en árboles de decisión y selecciona atributos para dividir datos en función de una métrica de ganancia de información.

**Clasificación del Problema:**

* Aprendizaje Supervisado: Utiliza datos etiquetados.
* Clasificación: Categorización de instancias en clases predefinidas.
* Árbol de Decisión: Modelo de estructura jerárquica para dividir datos.

**Viabilidad Computacional:**

C4.5 es viable computacionalmente para conjuntos de datos pequeños y medianos. Sin embargo, para grandes conjuntos de datos, su complejidad puede aumentar debido a la búsqueda exhaustiva de la mejor división para cada nodo del árbol.

**Contexto del Problema:**

C4.5 es ampliamente utilizado en áreas como:

* Diagnóstico médico.
* Sistemas de recomendación.
* Predicción de riesgos financieros.
* Análisis de comportamiento del cliente.

**Análisis de Datos y Estructura del Problema:**

* Entrada: Conjunto de datos etiquetados 𝐷={(𝑥1,𝑦1),…,(𝑥𝑛,𝑦𝑛)} donde 𝑥𝑖 son vectores de características y 𝑦𝑖 son etiquetas.
* Estructura de datos: Árbol de decisión jerárquico donde cada nodo representa un atributo y las hojas representan etiquetas de clase.

**Evaluación de Complejidad:**

* Entrenamiento: La complejidad es 𝑂(𝑛⋅𝑑⋅log(𝑛)) donde 𝑛 es el número de instancias y 𝑑 el número de atributos.
* Predicción: La complejidad depende de la profundidad del árbol, típicamente 𝑂(log(𝑛)).

**Análisis del Contexto del Problema para la cual el algoritmo es una solución**.

**Contexto general del Problema:**

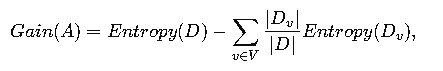
El algoritmo aborda problemas donde se requiere una clasificación precisa y comprensible. La salida de un árbol de decisión es fácilmente interpretable y puede ser utilizada directamente para toma de decisiones.

**Identificación de los conjuntos y estructuras de datos:**

1. Conjunto de datos:
   * Instancias con atributos discretos o continuos.
   * Etiquetas de clase asociadas.
2. Estructura:
   * Árbol binario o multinivel con nodos y hojas que representan divisiones de datos y etiquetas.

**Formalización del problema:**

Dado un conjunto de datos 𝐷 con 𝑛 instancias y 𝑑 atributos, el objetivo es construir un árbol de decisión 𝑇 que minimice el error de clasificación. Cada nodo 𝑡 𝑇 se define seleccionando el atributo 𝐴 que maximiza la ganancia de información:



Donde Dv es el subconjunto de 𝐷 dividido por el valor 𝑣 del atributo 𝐴

**Análisis de la Dinámica del Problema:**

1. Selección de atributos:
   * Identificar el mejor atributo usando la ganancia de información.
2. División de Datos:
   * Crear subárboles con subconjuntos de datos basados en el atributo seleccionado.
3. Poda:
   * Reducir la complejidad del árbol eliminando ramas redundantes.

**Evaluación de la Complejidad del Contexto:**

* Ventajas:
  + Soporte para datos continuos.
  + Manejo de valores faltantes.
  + Salida interpretable.
* Desventajas:
  + Propenso al sobreajuste sin poda adecuada.
  + Requiere tiempo considerable para construir árboles grandes.

**Identificación de los casos límites y excepciones:**

**Casos límite:**

* Conjuntos de datos con muchos valores faltantes.
* Datos con alta dimensionalidad.

**Excepciones:**

* Cuando los datos están altamente correlacionados, puede generar árboles redundantes.

**Interacción con otros sistemas:**

* Puede integrarse con sistemas de preprocesamiento para discretizar atributos continuos.
* Compatible con técnicas de ensamblado como Random Forest.

**Especificación Formal:**

**Definición de las entradas y salidas:**

* Entradas:
  + Conjunto de datos 𝐷: 𝑛 instancias con 𝑑 atributos.
  + Parámetros: Umbral mínimo para dividir nodos.
* Salida:
  + Un árbol de decisión 𝑇 que clasifica instancias.

**Definición formal del problema mediante funciones:**

* Entrenamiento:



donde BuildTree selecciona el atributo 𝐴 con mayor ganancia de información y divide 𝐷 subconjuntos.

* Predicción:



Donde 𝑥 es una nueva instancia y 𝑇 es el árbol generado.

**Relaciones y Restricciones:**

* Relaciones:
  + Cada atributo en 𝐷 puede ser considerado para una división.
* Restricciones:
  + La ganancia de información debe superar un umbral para justificar la división.

**Invariantes y propiedades:**

* Invariante principal: Cada nodo divide los datos en subconjuntos homogéneos respecto a la clase objetivo.
* Propiedades:
  + El árbol siempre tiene al menos un nodo raíz y varias hojas.

**Control de Flujo:**

1. Preprocesamiento:

* Manejo de datos faltantes.
* Normalización o discretización de atributos continuos.

1. Entrenamiento:

* Construcción del árbol dividiendo nodos en función de la ganancia de información.
* Poda para reducir sobreajuste.

1. Predicción:

* Recorrer el árbol desde la raíz hasta una hoja para determinar la clase.

**Pseudocodigo del Algoritmo**

Algoritmo C4.5

// Entradas:

// datos\_entrenamiento: Matriz[n][m], donde n es el número de instancias y m-1 es el número de atributos.

// etiquetas: Vector[n], clases esperadas de las instancias.

Funcion ConstruirArbol(datos\_entrenamiento, etiquetas)

Si todas las etiquetas son de la misma clase Entonces

Retornar CrearHoja (etiquetas[0]) // Nodo hoja con la clase única.

FinSi

Si datos\_entrenamiento está vacío o atributos están vacíos Entonces

Retornar CrearHoja(ClaseMayoritaria(etiquetas)) // Hoja con la clase más común.

FinSi

// Elegir el mejor atributo usando ganancia de información normalizada

mejor\_atributo <- SeleccionarMejorAtributo(datos\_entrenamiento, etiquetas)

// Dividir datos basándose en el mejor atributo

nodos\_hijos <- Diccionario

Para cada valor en ValoresUnicos(mejor\_atributo, datos\_entrenamiento)

subconjunto\_datos <- FiltrarDatos(datos\_entrenamiento, mejor\_atributo, valor)

subconjunto\_etiquetas <- FiltrarEtiquetas(datos\_entrenamiento, etiquetas, mejor\_atributo, valor)

Si subconjunto\_datos está vacío Entonces

nodos\_hijos[valor] <- CrearHoja(ClaseMayoritaria(etiquetas))

Sino

nodos\_hijos[valor] <- ConstruirArbol(subconjunto\_datos, subconjunto\_etiquetas)

FinSi

FinPara

// Crear nodo de decisión

Retornar CrearNodoDecision(mejor\_atributo, nodos\_hijos)

FinFuncion

// Proceso principal

Escribir "Construyendo árbol de decisión usando C4.5..."

Definir datos\_entrenamiento como Matriz[n][m] // Cargar datos

Definir etiquetas como Vector[n] // Cargar etiquetas

arbol\_decision <- ConstruirArbol(datos\_entrenamiento, etiquetas)

Escribir "Árbol de decisión construido con éxito."

FinAlgoritmo

Funcion SeleccionarMejorAtributo(datos\_entrenamiento, etiquetas)

mejor\_atributo <- NULL

mejor\_ganancia <- -INFINITO

Para cada atributo en datos\_entrenamiento

ganancia <- CalcularGananciaNormalizada(datos\_entrenamiento, etiquetas, atributo)

Si ganancia > mejor\_ganancia Entonces

mejor\_ganancia <- ganancia

mejor\_atributo <- atributo

FinSi

FinPara

Retornar mejor\_atributo

FinFuncion

**Casos de Usos Posibles:**

* Clasificación de Riesgo Crediticio:
  + Identificar si un cliente es de bajo o alto riesgo basado en atributos como ingresos, historial crediticio y deudas.
* Diagnóstico Médico:
  + Clasificar pacientes en diferentes categorías de enfermedades basándose en síntomas y pruebas de laboratorio.
* Predicción de Comportamiento del Cliente:
  + Predecir si un cliente comprará un producto dependiendo de su historial de compras, edad, y ubicación.
* Detección de Fraudes:
  + Clasificar transacciones bancarias como "fraudulentas" o "no fraudulentas" basándose en patrones de datos históricos.
* Segmentación de Mercado:
  + Dividir a los consumidores en grupos basados en datos demográficos y comportamientos.
* Clasificación de Texto:
  + Determinar la categoría de un artículo (noticias, deportes, tecnología) en función de las palabras clave.