**Nombre del Algoritmo:** Averaged One-Dependence Estimators (AODE)

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

**Delimitación del Problema:**

El algoritmo Averaged One-Dependence Estimators (AODE) aborda el problema de clasificación supervisada al mejorar la simplicidad y rapidez del modelo de Naive Bayes (NB), mientras relaja su estrictiva suposición de independencia condicional entre atributos. AODE considera dependencias de un atributo respecto a otro, manteniendo un equilibrio entre eficiencia computacional y precisión.

**Comprensión del Problema:**

El problema a resolver es cómo clasificar correctamente un conjunto de datos de entrada basándose en atributos que puedan tener ciertas dependencias mutuas. En el modelo Naive Bayes, se asume independencia condicional entre atributos, lo que puede ser una limitación cuando esta suposición no se cumple. AODE extiende esta idea promediando modelos que permiten dependencias de un atributo respecto a otros.

**Clasificación del Problema:**

* Aprendizaje Supervisado: Clasificación en función de un conjunto de datos etiquetados.
* Probabilístico: Utiliza distribuciones de probabilidad para inferir etiquetas.
* Ampliación de Naive Bayes: Permite una dependencia limitada entre atributos.

**Viabilidad Computacional:**

AODE es computacionalmente más intensivo que Naive Bayes debido a la consideración de múltiples dependencias, pero sigue siendo eficiente comparado con modelos más complejos como redes bayesianas completas. Es adecuado para conjuntos de datos pequeños a medianos.

**Contexto del Problema:**

AODE se aplican en áreas de clasificación donde:

* Existen dependencias leves entre atributos.
* Se busca mejorar la precisión de Naive Bayes sin incurrir en el alto costo computacional de modelos más complejos.

Ejemplos:

* Diagnósticos médicos
* Clasificación de texto.
* Análisis de riesgo financiero.

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

* Entrada
  + Un conjunto de datos etiquetados 𝐷 = {(𝑥1, 𝑦1),…,(𝑥𝑛,𝑦𝑛)} donde 𝑥𝑖 son vectores de características y 𝑦𝑖 son etiquetas.
* Estructura:
  + Modelos probabilísticos que consideran una dependencia limitada (one-dependence).

**Evaluación de Complejidad:**

* Entrenamiento
  + El costo de estimar probabilidades es 𝑂 (𝑛⋅𝑑²) donde 𝑛 es el número de instancias y 𝑑 es el número de atributos.
* Predicción
  + El tiempo de predicción es 𝑂(𝑑²), ya que se promedian estimadores dependientes.

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

**Contexto general del Problema:**

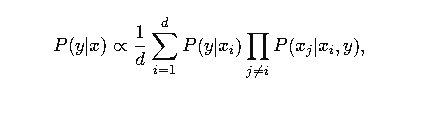
En aplicaciones prácticas, la suposición de independencia condicional de Naive Bayes rara vez se cumple. Sin embargo, métodos que modelan dependencias completas entre atributos son computacionalmente costosos. AODE equilibra simplicidad y precisión al modelar una dependencia limitada y promediar múltiples estimadores.

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

1. Conjuntos de datos:
   * Consta de instancias con atributos 𝑋={𝑥1,…,𝑥𝑑} y etiquetas 𝑌
2. Estructura de Datos:
   * Tablas de frecuencias para calcular probabilidades condicionales.
   * Matrices de dependencias de un atributo con respecto a otro.

**Formalización del problema:**

El objetivo es predecir 𝑦 para una instancia dada 𝑥 maximizando la probabilidad condicional promedio:



donde:

* P(y∣xi​) es la probabilidad de 𝑦 dado el atributo 𝑥𝑖.
* P(xj∣xi,y) es la probabilidad de 𝑥𝑗 condicionado en 𝑥𝑖 y 𝑦

**Análisis de la Dinámica del Problema:**

1. Fase de entrenamiento:
   * Calcular frecuencias relativas para estimar probabilidades marginales y condicionales.
2. Fase de Predicción:
   * Calcular promedios ponderados de estimadores dependientes

**Evaluación de la Complejidad del Contexto:**

1. Ventajas
   * Relaja la suposición de independencia condicional de NB.
   * Mejora la precisión en escenarios prácticos.
2. Desventajas
   * Incremento en costo computacional respecto a NB.
   * Requiere memoria adicional para almacenar dependencias**.**

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

**Casos límite:**

* Conjuntos de datos con pocos atributos: El beneficio respecto a NB es mínimo.
* Datos escasos: Puede no estimar correctamente las probabilidades condicionales.

**Excepciones:**

* Cuando los atributos son realmente independientes, NB puede superar a AODE

**Interacción con otros sistemas:**

* Puede integrarse con técnicas de preprocesamiento como selección de atributos.
* Compatible con validación cruzada para ajustar hiperparámetros.

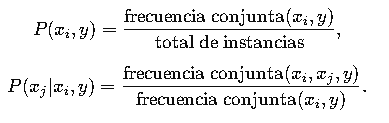
**Especificación Formal:**

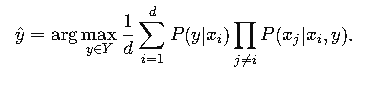
**Definición de las entradas y salidas:**

* Entradas:
  + 𝐷= {(𝑥1, 𝑦1), ..., (𝑥𝑛, 𝑦𝑛)}: Datos de entrenamiento
  + 𝑋𝑡𝑒𝑠𝑡: vector de atributos para predicción.
* Salidas:
  + Etiquetas 𝑌^ predichas para 𝑋𝑡𝑒𝑠𝑡.

**Definición formal del problema mediante funciones:**

1. Estimación de probabilidades



1. Predicción:

**Relaciones y Restricciones:**

* **Relaciones:** 
  + Cada atributo depende de otro y de la etiqueta 𝑦.
* **Restricciones:**
  + Requiere datos suficientes para estimar probabilidades conjuntas.

**Invariantes y propiedades:**

* Invariante principal: Los promedios de estimadores individuales siempre consideran dependencias de un atributo.
* Propiedades:
  + Mejora en precisión frente a NB.
  + Robustez frente a datos con dependencias leves.

**Control de Flujo:**

1. Preprocesamiento:
   * Normalización de los datos
   * Transformación para manejar valores categóricos.
2. Entrenamiento
   * Calcular 𝑃(𝑥𝑖,𝑦) y 𝑃(𝑥𝑗∣𝑥𝑖,𝑦) para cada par de atributos.
3. Predicción:
   * Evaluar P(𝑦 ∣𝑥test) usando la fórmula de promedios dependientes.
   * Seleccionar la etiqueta con mayor probabilidad.

**Pseudocodigo del Algoritmo:**

Algoritmo AODE

// Entradas:

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

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

// nueva\_instancia: Vector[m], datos de la instancia a clasificar.

Funcion ConstruirAODE(datos\_entrenamiento, etiquetas)

Definir probabilidades\_clase como Diccionario // Almacena P(C)

Definir probabilidades\_condicionales como Diccionario // Almacena P(A|C) y P(A|C, A\_k)

// Calcular probabilidades de clase (P(C))

Para cada clase en etiquetas

probabilidades\_clase[clase] <- Contar(etiquetas == clase) / tamaño(etiquetas)

FinPara

// Calcular probabilidades condicionales simples y dependientes (P(A|C) y P(A|C, A\_k))

Para cada atributo\_i en datos\_entrenamiento

Para cada atributo\_j en datos\_entrenamiento

Si atributo\_i != atributo\_j Entonces

Para cada clase en etiquetas

Contar ocurrencias de (atributo\_i, atributo\_j | clase)

Calcular P(A\_i | C) y P(A\_i | C, A\_j)

Almacenar en probabilidades\_condicionales

FinPara

FinSi

FinPara

FinPara

Retornar (probabilidades\_clase, probabilidades\_condicionales)

FinFuncion

Funcion Clasificar(nueva\_instancia, probabilidades\_clase, probabilidades\_condicionales)

Definir puntuaciones como Diccionario // Guardará puntuaciones por clase

// Iterar por cada clase

Para cada clase en probabilidades\_clase

puntuacion <- log(probabilidades\_clase[clase])

Para cada atributo\_i en nueva\_instancia

Para cada atributo\_j en nueva\_instancia

Si atributo\_i != atributo\_j Entonces

puntuacion <- puntuacion + log(P(nueva\_instancia[atributo\_i] | clase, nueva\_instancia[atributo\_j]))

FinSi

FinPara

FinPara

puntuaciones[clase] <- puntuacion

FinPara

// Devolver la clase con la puntuación más alta

clase\_predicha <- ClaseConMaximaPuntuacion(puntuaciones)

Retornar clase\_predicha

FinFuncion

// Proceso principal

Escribir "Iniciando construcción del modelo AODE..."

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

Definir etiquetas como Vector[n] // Cargar etiquetas

Definir nueva\_instancia como Vector[m] // Cargar datos a clasificar

probabilidades\_clase, probabilidades\_condicionales <- ConstruirAODE(datos\_entrenamiento, etiquetas)

Escribir "Modelo construido. Clasificando nueva instancia..."

clase\_predicha <- Clasificar (nueva\_instancia, probabilidades\_clase, probabilidades\_condicionales)

Escribir "La clase predicha para la instancia es: ", clase\_predicha

FinAlgoritmo

**Casos de usos Posible:**

* Clasificación de Texto:
  + Identificar el género de un texto (ej., novela, poesía) basándose en métricas lingüísticas.
* Detección de Spam:
  + Clasificar correos electrónicos como "spam" o "no spam" usando palabras clave y métricas adicionales.
* Predicción de Diagnósticos Médicos:
  + Diagnosticar enfermedades basándose en síntomas, pruebas de laboratorio y antecedentes médicos.
* Clasificación de Imágenes:
  + Asignar etiquetas a imágenes utilizando características como colores, texturas y bordes.
* Sistemas de Recomendación:
  + Clasificar productos o contenido relevante para un usuario basado en su historial de preferencias.
* Análisis de Sentimientos:
  + Clasificar reseñas de usuarios como positivas, negativas o neutras en aplicaciones comerciales.