UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

Inteligencia Artificial

Ing. Juancarlos Santana Huamán
jsantana@ucss.edu.pe

¿Qué es la gestión de datos de IA?

- La gestión de datos de IA es la práctica de emplear inteligencia artificial (IA) y machine learning (ML) en el ciclo de vida de la gestión de datos. Los ejemplos incluyen la aplicación de IA para automatizar u optimizar la recopilación de datos, la limpieza de datos, el análisis de datos, la seguridad de datos y otros procesos de gestión de datos.
- Tanto la IA tradicional basada en reglas como los modelos de IA generativa más avanzados pueden ayudar con la gestión de datos.
- Las empresas modernas poseen grandes cantidades de datos sobre todo, desde transacciones financieras e inventario de productos hasta registros de empleados y preferencias de los clientes. Las organizaciones que emplean estos datos para informar la toma de decisiones e impulsar iniciativas comerciales pueden obtener beneficios significativos sobre sus competidores.

¿Qué es la gestión de datos de IA?

- Sin embargo, el desafío proviene de hacer que estos grandes conjuntos de datos sean lo suficientemente precisos, confiables y accesibles para que las personas los utilicen en la práctica.
- Las herramientas de IA y ML pueden ayudar a las organizaciones a emplear sus datos optimizando tareas como la integración de fuentes de datos, la limpieza de datos y la recuperación de datos. Como resultado, las empresas pueden tomar más decisiones basadas en datos.
- La gestión de datos de IA también ayuda a las organizaciones a construir las canalizaciones de datos de alta calidad que necesitan para entrenar y desplegar sus propios modelos de IA y algoritmos de machine learning.

Herramientas de gestión de datos de IA

- Muchos tipos de herramientas de administración de datos—como soluciones de almacenamiento de datos, herramientas de integración de datos, herramientas de master data management, soluciones de gobierno y otras—ahora incorporan capacidades de ML e IA. Estas herramientas pueden utilizar tanto algoritmos de IA tradicionales como sistemas de IA generativa.
  - **Los sistemas tradicionales** de IA realizan tareas específicas basadas en reglas—por ejemplo, un sistema de gestión de bases de datos que categoriza automáticamente los datos según criterios predefinidos.
  - **Los sistemas de IA generativa**, como Microsoft Copilot, Llama de Meta e IBM Granite™, responden al lenguaje natural y crean contenido original. Por ejemplo, un sistema de gestión de bases de datos con un modelo de lenguaje extensos (LLM) integrado puede crear resúmenes de datos y aceptar consultas en inglés sencillo en lugar de SQL.

Casos de uso de gestión de datos de IA

Descubrimiento de datos

- Las organizaciones de hoy en día trabajan con una gran cantidad de datos, que llegan al negocio de múltiples fuentes diferentes, en múltiples formatos. Estos datos son manejados por varios usuarios y terminan dispersos en nubes públicas y privadas, sistemas de almacenamiento on-premises e incluso endpoints personales de los empleados.
- Puede ser difícil realizar un seguimiento y administrar centralmente todos estos datos, lo que plantea dos problemas.
- En primer lugar, una organización no puede emplear un conjunto de datos si no sabe que el conjunto de datos existe.
- En segundo lugar, estos "datos ocultos" no descubiertos ni gestionados plantean riesgos de seguridad. Según el Informe del costo de una filtración de datos de IBM, un tercio de las violaciones de datos implican datos ocultos. Estas filtraciones cuestan 5.27 millones de dólares de promedio—un 16% más que el costo promedio global de las filtraciones.
- IA y ML pueden automatizar muchos aspectos del descubrimiento de datos, otorgando a las organizaciones más visibilidad y control sobre todos sus activos de datos.

Casos de uso de gestión de datos de IA

- Ejemplos de IA en el descubrimiento de datos
  - Las herramientas de descubrimiento de datos impulsadas por IA pueden escanear automáticamente dispositivos de red y repositorios de almacenamiento de datos, indexando nuevos datos casi en tiempo real.
  - Las herramientas automatizadas de clasificación de datos pueden etiquetar nuevos datos en función de reglas predefinidas o modelos de machine learning. Por ejemplo, la herramienta podría clasificar cualquier número de nueve dígitos en el formato XXX-XX-XXXX como un número de seguro social de Estados Unidos.
  - Los LLMs y otras herramientas de procesamiento de lenguaje natural pueden extraer datos estructurados de fuentes de datos no estructurados, como por ejemplo, extraer los datos de contacto y la experiencia previa de candidatos a un puesto de trabajo a partir de currículos en documentos de texto con distintos formatos.

Casos de uso de gestión de datos de IA

Data quality

- Los datos erróneos pueden causar más problemas que la falta total de datos. Si los datos de una organización son incompletos o inexactos, entonces las iniciativas comerciales y los modelos de IA creados sobre esos datos también serán deficientes.
- Las herramientas de IA y ML pueden ayudar a identificar y corregir errores en los datos de la organización, lo que significa que los usuarios no necesitan hacer el largo trabajo de limpieza manual de datos. La IA también puede trabajar más rápido y detectar más errores que un usuario humano.

Casos de uso de gestión de datos de IA

- Ejemplos de IA en la limpieza de datos
  - Las herramientas de preparación de datos habilitadas para IA pueden realizar comprobaciones de validación y marcar o corregir errores, como formato inadecuado y valores irregulares. Algunas herramientas de preparación de datos impulsadas por IA también pueden convertir los datos al formato adecuado, como convertir notas de reuniones no estructuradas en tablas estructuradas.
  - Los generadores de datos sintéticos pueden proporcionar missing values y llenar otros huecos en los conjuntos de datos. Estos generadores pueden emplear modelos de machine learning para identificar patrones en los datos existentes y generar puntos de datos sintéticos de gran precisión.
  - Algunas herramientas de master datan management (MDM) pueden emplear IA y ML para detectar y corregir errores y duplicados en registros críticos. Por ejemplo, fusionar dos registros de clientes con el mismo nombre, dirección y detalles de contacto.
  - Las herramientas de observabilidad de los datos impulsadas por IA pueden generar automáticamente registros de linaje de datos para que las organizaciones puedan rastrear quién usa los datos y cómo cambian con el paso del tiempo.

Casos de uso de gestión de datos de IA

Accesibilidad a los datos

- Los silos de datos impiden que muchas organizaciones aprovechen todo el valor de sus datos. IA y ML pueden optimizar los esfuerzos de integración de datos, reemplazando los repositorios de silos con estructuras de datos unificadas. Los usuarios de toda la organización pueden acceder a los activos de datos que necesitan cuando los necesitan.
- Ejemplos de IA en el acceso a datos
  - Las herramientas de integración de datos habilitadas para IA pueden detectar automáticamente las relaciones entre diferentes conjuntos de datos, lo que permite a la organización conectarlos o fusionarlos.
  - Las herramientas de gestión de metadatos con capacidades de IA pueden ayudar a automatizar la creación de catálogos de datos al generar descripciones de activos de datos basadas en etiquetado y clasificación.
  - Las bases de datos y los catálogos de datos con interfaces impulsadas por LLM pueden aceptar y procesar comandos de lenguaje natural, lo que permite a los usuarios encontrar activos de datos y productos sin escribir código personalizado o SQL queries. Algunas interfaces impulsadas por LLM también pueden ayudar a los usuarios a refinar consultas, enriquecer conjuntos de datos o sugerir puntos de datos relacionados.
  - Los motores de consulta habilitados para IA pueden emplear algoritmos de machine learning para mejorar el rendimiento de la base de datos mediante el análisis de patrones de carga de trabajo y la optimización de la ejecución de consultas.

Casos de uso de gestión de datos de IA

Data security

- Existe un caso comercial que justifica priorizar la seguridad de los datos. La filtración de datos promedio le cuesta a una organización 4.88 millones de dólares entre pérdida de negocios, tiempo de inactividad del sistema, daño a la reputación y esfuerzos de respuesta, según el Informe del costo de una filtración de datos.
- IA y ML pueden ayudar a aplicar políticas de seguridad, detectar filtraciones y bloquear actividades no autorizadas.
- Ejemplos de IA en la seguridad de datos
  - Las herramientas de prevención de pérdida de datos impulsadas por IA pueden detectar automáticamente información de identificación personal (PII) y otros datos confidenciales, aplicar controles de seguridad y marcar o bloquear el uso no autorizado de esos datos.
  - Las herramientas de detección de amenazas basadas en anomalías, tales como el análisis del comportamiento de usuarios y entidades (UEBA) y la detección y respuesta de endpoints (EDR), emplean algoritmos de IA y ML para monitorear la actividad de la red. Detectan desviaciones sospechosas de la norma, como una gran cantidad de datos que se trasladan repentinamente a una nueva ubicación.
  - Los LLMs pueden ayudar a las organizaciones a generar e implementar políticas de gobernanza de datos. Por ejemplo, en un sistema de control de acceso basado en roles (RBAC), un LLM puede ayudar al equipo de seguridad a describir los diferentes tipos de roles y sus permisos. El LLM también podría ayudar a convertir estas descripciones de roles en reglas para un sistema de gestión de identidad y acceso.
  - Las herramientas de detección de fraude habilitadas para IA pueden emplear IA y ML para analizar patrones y detectar transacciones anormales.

Beneficios de la gestión de datos de IA

Aprovechar todo el valor del big data para las empresas

- En el Reporte de gestión de la IA y la información de AvePoint, el 64% de las organizaciones encuestadas dijeron que gestionaban al menos un petabyte de datos.1 En perspectiva, eso equivale a aproximadamente 9 cuatrillones de bits de información. Y gran parte viene en formatos no estructurados, como archivos de texto, imágenes y video.
- Todos estos datos pueden ser una bendición para los científicos de datos, pero es imposible gestionar manualmente datos tan complejos en cantidades tan masivas. Las herramientas de IA y ML pueden hacer que estos datos sean utilizables mediante la automatización de tareas críticas como el descubrimiento, la integración y la limpieza.
- Cuando los datos están limpios y son accesibles, las organizaciones pueden emplearlos para proyectos avanzados de analytics de datos, como una iniciativa de análisis predictivos que emplea datos históricos para pronosticar tendencias futuras en el gasto de los consumidores.
- Las tecnologías de IA también pueden hacer que los datos sean más accesibles para los usuarios sin experiencia en ciencia de datos. Los catálogos de datos fáciles de usar con interfaces de bases de datos impulsadas por LLM y visualizaciones automatizadas permiten que más usuarios de toda la empresa empleen datos para fundamentar sus decisiones.

Beneficios de la gestión de datos de IA

Impulsando las iniciativas de IA

- El 59% de los directores ejecutivos (CEOs) encuestados por IBM Institute for Business Value creen que el beneficio competitivo de una organización en el futuro depende de tener la IA generativa más avanzada. Para crear y desplegar esos modelos de IA, las organizaciones necesitan flujos constantes de datos buenos y limpios.
- Al agilizar la gestión de datos, las herramientas de IA ayudan a crear los canales de datos fiables y de alta calidad que las organizaciones necesitan para capacitar sus propios modelos de IA y ML. Y como estos modelos se pueden entrenar con los datos de la empresa, se pueden entrenar para realizar tareas y resolver problemas específicos de la empresa y sus clientes.

Beneficios de la gestión de datos de IA

Usar los datos sin dejar de cumplir

- Las herramientas de seguridad y gobernanza basadas en IA ayudan a prevenir ciberataques y filtraciones de datos, que pueden resultar costosos. También permiten a las empresas emplear los datos que tienen cumpliendo con las regulaciones de privacidad y protección de datos como GDPR y el Estándar de Seguridad de Datos de la Industria de Tarjetas de Pago (PCI-DSS).
- Según el Institute for Business Value, el 57% de los CEOs afirma que la seguridad de datos es una barrera para adoptar IA generativa. El 45% dice que la privacidad de los datos también es una barrera. Estas barreras pueden ser especialmente desafiantes en industrias altamente reguladas, como la atención médica y las finanzas.
- La gestión de datos habilitada por IA puede ayudar aplicando automáticamente las protecciones adecuadas y las políticas de uso de datos. De esa manera, solo los usuarios autorizados pueden acceder a los datos y solo pueden usarlos de la manera que lo permitan las regulaciones de la industria y la política de la empresa.
- Los generadores de datos sintéticos también pueden ayudar generando conjuntos de datos que reflejen con precisión las tendencias generales, al tiempo que eliminan datos personales confidenciales que una organización podría no tener permitido usar de ciertas maneras.

¿Qué es el Decision Tree o árbol de decisiones en IA?

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Un decision tree (o árbol de decisión) es una técnica de modelado predictivo en la cual se utiliza una estructura de árbol para representar decisiones y sus consecuencias. El árbol de decisión divide un conjunto de datos en subconjuntos más pequeños en función de las características de los datos. Cada nodo del árbol representa una característica (atributo), cada rama representa una decisión basada en esa característica y cada hoja representa el resultado de la decisión tomada.
- Los árboles de decisión se utilizan en la ciencia de datos y el aprendizaje automático como una técnica de clasificación y predicción. En la clasificación, los árboles de decisión se utilizan para asignar una etiqueta o clasificación a un conjunto de datos en función de las características de los mismos. En la predicción, se utilizan para predecir valores numéricos de una variable objetivo. Los algoritmos de árboles de decisión son a menudo utilizados en problemas de clasificación y se han utilizado en numerosos campos, como medicina, finanzas, marketing, entre otros.
- Es una técnica popular debido a que es fácil de entender y se puede visualizar de manera gráfica, lo que facilita la interpretación de los resultados del análisis. En resumen, el decision tree es una técnica en el aprendizaje automático que permite la creación de modelos predictivos utilizando una estructura de árbol de decisiones para la clasificación y predicción de datos.

¿Cómo funcionan los árboles de decisión en machine learning?

- Los árboles de decisión son un tipo de aprendizaje automático supervisado destinado a realizar predicciones en función de un conjunto de preguntas a las que el sistema ha de responder y realizar una predicción.

¿Cómo funcionan los árboles de decisión?

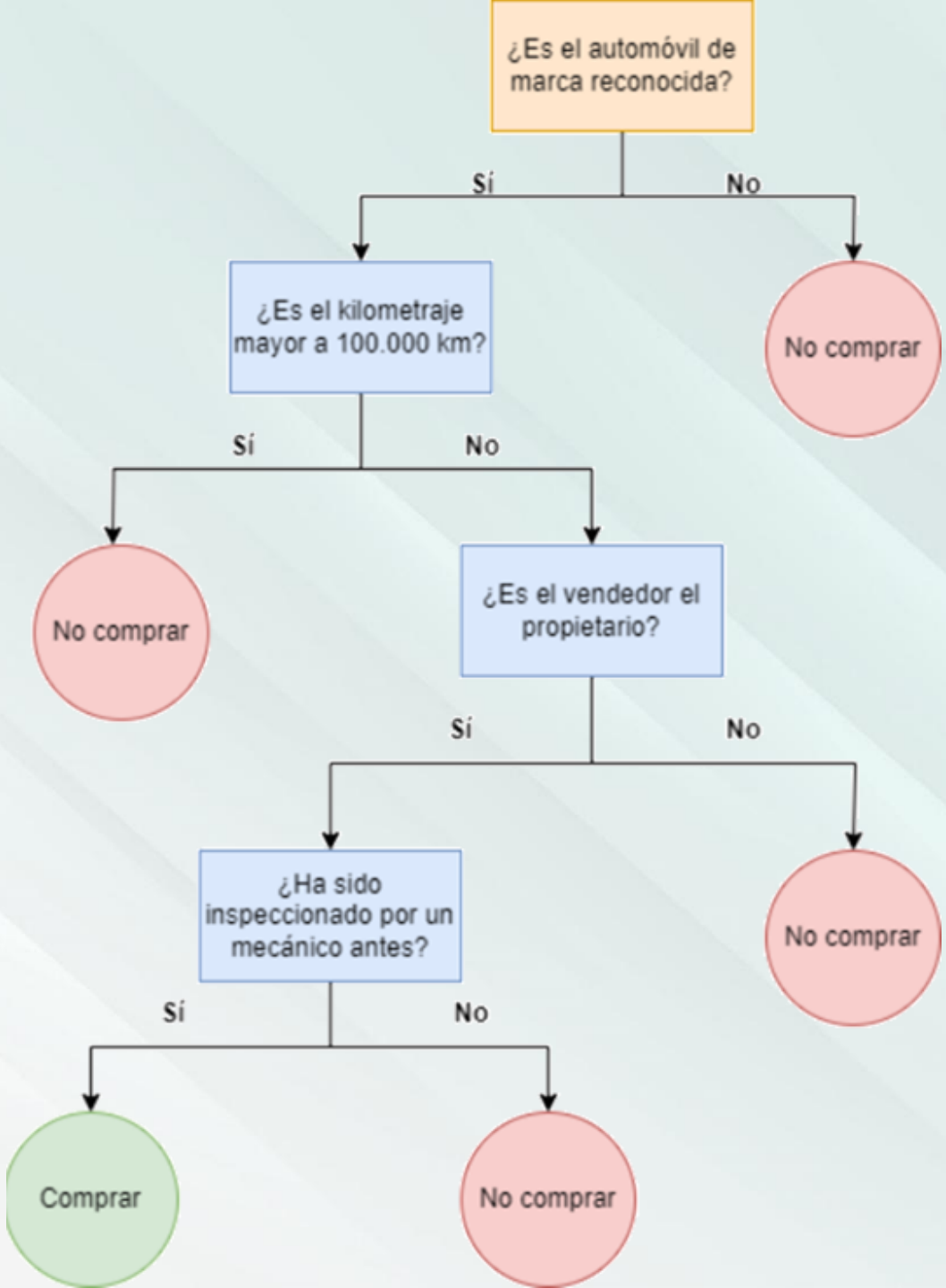
- Podemos asemejar el funcionamiento de un árbol de decisión al de la simple mecánica del juego infantil “veo, veo”.
- Imagina que queremos que un ordenador aprenda a distinguir entre diferentes tipos de frutas, como manzanas y naranjas. El árbol de decisión empezaría con una pregunta, como “¿Es la fruta redonda?” Si la respuesta es sí, entonces la computadora podría preguntar “¿Es la fruta roja?” Si la respuesta es sí de nuevo, podría decidir que la fruta es una manzana. Si la respuesta es no, podría ser una naranja. Si la fruta no es redonda, el sistema formularía otra pregunta a raíz de otra característica, como “¿Es la fruta de color naranja?”
- El árbol de decisión sigue haciendo preguntas y tomando decisiones basadas en las respuestas hasta que clasifica correctamente todas las frutas en grupos.

¿Cómo funcionan los árboles de decisión?

- **Nodos raíz y nodos de decisión.** Así pues, un árbol de decisión es parecido a un árbol. A la base del árbol se le llama nodo raíz, a partir del cual nacen los nodos de decisión, que representan posibles preguntas por las que se quiere segmentar toda la muestra. La formulación de estas preguntas dependerá de las variables que hayamos identificado y que queramos utilizar para segmentar el conjunto de datos.
- **Divisiones y subnodos.** A estos nodos de decisión les corresponden una, dos o más respuestas, llamadas divisiones o particiones. Cuantas más particiones o respuestas posibles tengan un nodo de decisión, más subnodos de decisión se crearán, y mayor será la complejidad del árbol.
- A medida que avance al árbol y sus ramificaciones, el número de nodos de decisión se irán reduciendo y el conjunto de datos se irán clasificando en subconjuntos más homogéneos basándonos en ciertos criterios.
- **Hojas.** El objetivo del árbol de decisión es que el sistema de inteligencia artificial vaya cribando poco a poco el conjunto de datos y los categorice con precisión y rigor. Así pues, el árbol de decisión termina en hojas, es decir, respuestas únicas sin particiones para todos los nodos de decisión.
- **Poda.** A medida que va ramificándose el árbol en particiones, es necesario ir filtrando nodos para evitar que crezca en exceso con información que no nos es relevante para nuestros propósitos. Esta depuración del modelo de inteligencia artificial, también llamada en inglés “pruning”, se realiza durante el diseño del árbol de decisiones, así como después de completar todo el árbol. Es una forma de simplificar el modelo, hacerlo más útil y evitar que se sobreajuste a los datos del entrenamiento.

¿Para qué sirven los árboles de decisión?

- Los árboles de decisión en la Inteligencia Artificial (IA) se utilizan para diversas aplicaciones, aquí te menciono algunas de las más comunes:
  - **Clasificación de datos**: Los árboles de decisión pueden clasificar datos en diferentes categorías o clases basándose en ciertos criterios.
  - **Predicciones**: Su aplicación más útil y extendida es la de hacer predicciones sobre valores numéricos. Por ejemplo, pueden predecir el precio de una casa basándose en características como su tamaño, ubicación, número de habitaciones, etc.
  - **Detección de anomalías**: En ciberseguridad, se pueden utilizar para identificar actividades sospechosas (patrones de uso irregular) en una red, como intentos de piratería o intrusiones.
  - **Aprendizaje automático supervisado**: Los árboles de decisión son un tipo de aprendizaje automático supervisado destinado a realizar predicciones en función de un conjunto de preguntas a las que el sistema ha de responder y realizar una predicción.
  - **Resolución de problemas de regresión o de clasificación**: Un árbol de decisión es una técnica que emplea algoritmos en forma de árbol para enseñar a las máquinas a tomar decisiones y, por tanto, a resolver problemas de regresión o de clasificació.
  - Creación de modelos predictivos precisos y fiables: Como resultado, obtenemos modelos predictivos precisos y fiables.
- En resumen, los árboles de decisión son una herramienta poderosa en el campo de la IA que permite una representación clara y fácil de entender de la toma de decisiones.


¿Es el automóvil de marca reconocida?
Sí
No
¿Es el kilometraje mayor a 100.000 km?
No comprar
Sí
No
No comprar
¿Es el vendedor el propietario?
Sí
No
¿Ha sido inspeccionado por un mecánico antes?
No comprar
Sí
No
Comprar
No comprar

Ventajas y desventajas de los árboles de decisión

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Los árboles de decisión en machine learning son una técnica muy útil y popular por las siguientes razones:
  - **No requieren mucho tiempo de preprocesado y limpieza de datos**: A diferencia de algunos otros algoritmos de aprendizaje automático, los árboles de decisión pueden manejar datos con diferentes tipos de variables (categóricas y numéricas) sin necesidad de una preparación extensa. Trabajar con árboles de decisión es muy eficiente, ya que no se ha de perder mucho tiempo antes de elaborar uno.
  - **No importa si el dataset tiene lagunas**: Los árboles de decisión pueden manejar conjuntos de datos incompletos, es decir, conjuntos de datos que tienen valores faltantes o lagunas.
  - **Es sencillo de explicar visualmente a otras personas**: Los árboles de decisión son fáciles de entender y visualizar, incluso para personas que no tienen formación específica en el área. Puedes representar un árbol de decisión como un diagrama que muestra las decisiones tomadas en cada nodo y cómo se llega a una conclusión para facilitar que otras personas comprendan el modelo que estás entrenando.

Ventajas y desventajas de los árboles de decisión

- Sin embargo, debemos ser conscientes de que los árboles de decisión tienen sus limitaciones y desventajas, como las siguientes:
  - **Inestabilidad**: Los cambios en los datos o la presencia de ruido pueden hacer que la estructura del árbol varíe enormemente. En esencia, los árboles son sensibles a pequeñas variaciones en los datos de entrenamiento o a la presencia de ruido. Es decir, si los datos cambian un poco o si hay errores en los datos, la estructura del árbol resultante puede verse muy alterada, haciendo que la toma de decisiones sea menos rigurosa según las predicciones.
  - **Lleva más tiempo entrenar un modelo usando un árbol de decisión**: Aunque los árboles de decisión son relativamente rápidos en la etapa de predicción, pueden llevar más tiempo entrenar el modelo, especialmente si el conjunto de datos es grande o si el árbol crece demasiado. Esto se debe a que el algoritmo necesita considerar todas las posibles divisiones y características para construir el árbol, lo que puede ser computacionalmente costoso.
  - **Sobreajuste (overfitting)**: Los árboles de decisión tienen una alta capacidad para adaptarse a los datos de entrenamiento, lo que conduce a una situación de overfitting. El sobreajuste ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y captura el ruido o las características específicas de esos datos en lugar de aprender patrones generales que se puedan generalizar a nuevos datos. Así pues, cuando a un modelo sobreentrenado lo alimentamos con información de un conjunto de datos diferentes, ofrece muchas respuestas incorrectas y un rendimiento ínfimo.

Tipos de árboles de decisión

- Diferenciamos los siguientes árboles de decisión según su enfoque:

De clasificación

- Este tipo de árbol se utiliza cuando la variable objetivo es categórica, es decir, pertenece a un conjunto discreto de clases o categorías. El árbol de clasificación divide el conjunto de datos en función de las características para clasificar las instancias en diferentes categorías. Por ejemplo, puede clasificar correos electrónicos como spam o no spam, pacientes como enfermos o sanos, o clientes como compradores o no compradores.

De regresión

- Un árbol de regresión se utiliza cuando la variable objetivo es numérica o continua, es decir, puede tomar cualquier valor dentro de un rango específico. El árbol de regresión divide el conjunto de datos en función de las características para predecir un valor numérico. Por ejemplo, puede predecir el precio de una casa en función de su tamaño, la cantidad de habitaciones, la ubicación, etc. o predecir la temperatura máxima diaria en función de variables meteorológicas como la humedad, la presión atmosférica y la velocidad del viento.

¿Cómo se crea el Decision Tree o árbol de decisiones de IA?

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

- Crear un árbol de decisiones en Inteligencia Artificial (IA) implica varios pasos. Aquí te dejo un resumen de cómo se puede hacer:
  - **División de los datos**: La técnica comienza dividiendo los datos en subconjuntos más pequeños. La variabilidad se reduce al mismo tiempo que aumenta la homogeneidad dentro de cada subgrupo.
  - **Selección de variables**: En este paso, el modelo selecciona automáticamente las mejores variables para realizar la partición utilizando medidas como entropía e índice Gini.
  - **Continuación del proceso**: El proceso continúa hasta que todos los nodos sean puros o hasta alcanzar alguna condición límite previa establecida por el usuario (por ejemplo, profundidad del árbol).
  - Aplicación del modelo: Cuando el modelo está entrenado, puede aplicarse a nuevos conjuntos de datos entrantes para hacer predicciones precisas.
  - **Poda**: A medida que va ramificándose el árbol en particiones, es necesario ir filtrando nodos para evitar que crezca en exceso con información que no nos es relevante para nuestros propósitos. Esta depuración del modelo de inteligencia artificial, también llamada en inglés “pruning”, se realiza durante el diseño del árbol de decisiones, así como después de completar todo el árbol.
  - **Validación y evaluación del modelo**: Una vez que el árbol de decisiones ha sido construido, es importante validar y evaluar su rendimiento. Esto puede implicar el uso de técnicas como la validación cruzada.
  - **Implementación del modelo**: Finalmente, una vez que estás satisfecho con el rendimiento del modelo, puedes implementarlo para hacer predicciones en nuevos datos.
- Es importante mencionar que existen varias bibliotecas y herramientas que pueden ayudarte a crear árboles de decisiones en IA, como Scikit-learn en Python, rpart en R, y otros.

Crear un árbol de decisiones usando el algoritmo ID3 con un ejemplo resuelto

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

¿Qué es el algoritmo ID3?

- El algoritmo ID3 (Dicotomizador Iterativo 3) es uno de los más antiguos y utilizados para crear árboles de decisión a partir de un conjunto de datos. Emplea el concepto de entropía y ganancia de información para seleccionar el mejor atributo para dividir los datos en cada nodo. La entropía mide la incertidumbre o aleatoriedad de los datos, y la ganancia de información cuantifica la reducción de la incertidumbre lograda al dividir los datos en un atributo específico. El algoritmo ID3 divide recursivamente el conjunto de datos según los atributos con mayor ganancia de información hasta que se cumple un criterio de parada, lo que genera un árbol de decisión que puede utilizarse para tareas de clasificación.

Comprensión del algoritmo ID3:

- El algoritmo ID3 utiliza el concepto de entropía y ganancia de información para construir un árbol de decisión. La entropía mide la incertidumbre o aleatoriedad de un conjunto de datos, mientras que la ganancia de información cuantifica la reducción de entropía lograda al dividir los datos en un atributo específico. El atributo con la mayor ganancia de información se selecciona como nodo de decisión del árbol.

Pasos para crear un árbol de decisiones utilizando el algoritmo ID3:

- **Paso 1: Preprocesamiento de datos:** Limpiar y preprocesar los datos. Gestionar los valores faltantes y convertir las variables categóricas en representaciones numéricas si es necesario.
- **Paso 2: Selección del nodo raíz:** Calcule la entropía de la variable objetivo (etiquetas de clase) con base en el conjunto de datos. La fórmula para la entropía es:

$$Entropy(S) = -\sum (p\_i \times \log_2(p\_i))$$

- Donde:
  - p_i es la proporción de instancias que pertenecen a la clase i.

Pasos para crear un árbol de decisiones utilizando el algoritmo ID3:

- **Paso 3: Cálculo de la ganancia de información:** Para cada atributo del conjunto de datos, calcule la ganancia de información al dividir el conjunto de datos en función de dicho atributo. La fórmula para la ganancia de información es:

$$Information\ Gain(S, A) = Entropy(S) - \sum\left((|S\_v|/|S|) \times Entropy(S\_v)\right)$$

- Donde:
  - S_v ∈ es el subconjunto de instancias para cada valor posible del atributo A y $|S\_v|$ ∈ es el número de instancias en ese subconjunto.
- **Paso 4: Seleccionar el mejor atributo:** elija el atributo con la mayor ganancia de información como nodo de decisión para el árbol.
- **Paso 5: Dividir el conjunto de datos:** divida el conjunto de datos según los valores del atributo seleccionado.
- **Paso 6: Repetir el proceso:**
  Repetir recursivamente los pasos 2 a 5 para cada subconjunto hasta que se cumpla un criterio de detención (por ejemplo, la profundidad del árbol alcanza un límite máximo o todas las instancias de un subconjunto pertenecen a la misma clase).

Ejemplo resuelto:

- Ilustremos el algoritmo ID3 con un ejemplo sencillo de clasificación para jugar al tenis según las condiciones meteorológicas. Consideremos el siguiente conjunto de datos:

Clima	Temperatura	Humedad	Ventoso	¿Juegas al tenis?
Soleado	Caliente	Alto	FALSO	No
Soleado	Caliente	Alto	Verdadero	No
Nublado	Caliente	Alto	FALSO	Sí
Lluvioso	Leve	Alto	FALSO	Sí
Lluvioso	Fresco	Normal	FALSO	Sí
Lluvioso	Fresco	Normal	Verdadero	No
Nublado	Fresco	Normal	Verdadero	Sí
Soleado	Leve	Alto	FALSO	No
Soleado	Fresco	Normal	FALSO	Sí
Lluvioso	Leve	Normal	FALSO	Sí
Soleado	Leve	Normal	Verdadero	Sí
Nublado	Leve	Alto	Verdadero	Sí
Nublado	Caliente	Normal	FALSO	Sí
Lluvioso	Leve	Alto	Verdadero	No

Ejemplo resuelto:

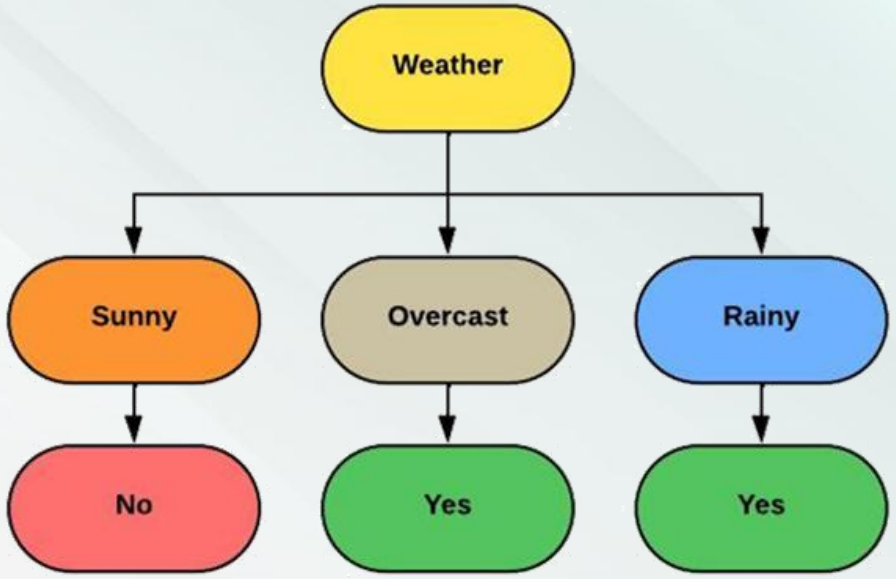
- **Paso 1: Preprocesamiento de datos:** el conjunto de datos no requiere ningún preprocesamiento, ya que está en un formato adecuado.
- **Paso 2: Cálculo de la entropía:** para calcular la entropía, primero determinamos la proporción de instancias positivas y negativas en el conjunto de datos:
  - Casos positivos (Jugar al tenis = Sí): 9
  - Instancias negativas (Jugar al tenis = No): 5

$$Entropy(S) = -\sum(p\_i \times \log_2(p\_i))$$

  - Entropía(S) = -(9/14) * log2(9/14) – (5/14) * log2(5/14) ≈ 0,940
- **Paso 3: Cálculo de la ganancia de información:** Calculamos la ganancia de información para cada atributo (clima, temperatura, humedad, viento) y elegimos el atributo con la mayor ganancia de información como nodo raíz.
  - Ganancia de información (S, Clima) = Entropía (S) – [(5/14) * Entropía (Soleado) + (4/14) * Entropía (Nublado) + (5/14) * Entropía (Lluvioso)] ≈ 0,246
  - Ganancia de información (S, Temperatura) = Entropía (S) – [(4/14) * Entropía (Caliente) + (4/14) * Entropía (Suave) + (6/14) * Entropía (Fría)] ≈ 0,029
  - Ganancia de información (S, Humedad) = Entropía (S) – [(7/14) * Entropía (Alta) + (7/14) * Entropía (Normal)] ≈ 0,152
  - Ganancia de información (S, Windy) = Entropía (S) – [(8/14) * Entropía (Falso) + (6/14) * Entropía (Verdadero)] ≈ 0,048

Ejemplo resuelto:

- **Paso 4: Seleccionar el mejor atributo:** El atributo “Clima” tiene la mayor ganancia de información, por lo que lo seleccionamos como el nodo raíz de nuestro árbol de decisiones.
- **Paso 5: División del conjunto de datos:** Dividimos el conjunto de datos en función de los valores del atributo “Clima” en tres subconjuntos (Soleado, Nublado, Lluvioso).
- **Paso 6: Repetir el proceso:** Dado que el atributo "Clima" no tiene valores repetidos en ningún subconjunto, dejamos de dividir y etiquetamos cada nodo hoja con la clase mayoritaria de ese subconjunto. El árbol de decisión se verá como se muestra a continuación:



Código Python para crear un árbol de decisiones utilizando el algoritmo ID3:

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

```
import pandas as pd
import numpy as np
import random

# Define the dataset
data = {
    'Weather': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast',
'Rainy'],
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High'],
    'Windy': [False, True, False, False, False, True, True, False, False, False, True, True, False, True],
    'Play Tennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
}

df = pd.DataFrame(data)

def entropy(target_col):
    elements, counts = np.unique(target_col, return_counts=True)
    entropy_val = -np.sum([(counts[i] / np.sum(counts)) * np.log2(counts[i] / np.sum(counts)) for i in range(len(elements))])
    return entropy_val
```

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

```
def information_gain(data, split_attribute_name, target_name):
    total_entropy = entropy(data[target_name])
    vals, counts= np.unique(data[split_attribute_name], return_counts=True)
    weighted_entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(data.where(data[split_attribute_name]==vals[i]).dropna()[target_name]) for i in
range(len(vals))])
    information_gain_val = total_entropy - weighted_entropy
    return information_gain_val

def id3_algorithm(data, original_data, features, target_attribute_name, parent_node_class):
    # Base cases
    if len(np.unique(data[target_attribute_name])) <= 1:
        return np.unique(data[target_attribute_name])[0]
    elif len(data) == 0:
        return np.unique(original_data[target_attribute_name])[np.argmax(np.unique(original_data[target_attribute_name], return_counts=True)[1])]
    elif len(features) == 0:
        return parent_node_class
    else:
        parent_node_class = np.unique(data[target_attribute_name])[np.argmax(np.unique(data[target_attribute_name], return_counts=True)[1])]
        item_values = [information_gain(data, feature, target_attribute_name) for feature in features]
        best_feature_index = np.argmax(item_values)
        best_feature = features[best_feature_index]
        tree = {best_feature: {}}
        features = [i for i in features if i != best_feature]
        for value in np.unique(data[best_feature]):
            value = value
            sub_data = data.where(data[best_feature] == value).dropna()
            subtree = id3_algorithm(sub_data, data, features, target_attribute_name, parent_node_class)
            tree[best_feature][value] = subtree
        return tree
```

UNIVERSIDAD CATÓLICA SEDES SAPIENTIAE

```
def predict(query, tree, default = 1):
    for key in list(query.keys()):
        if key in list(tree.keys()):
            try:
                result = tree[key][query[key]]
            except:
                return default
            result = tree[key][query[key]]
            if isinstance(result, dict):
                return predict(query, result)
            else:
                return result

def train_test_split(df, test_size):
    if isinstance(test_size, float):
        test_size = round(test_size * len(df))
    indices = df.index.tolist()
    test_indices = random.sample(population=indices, k=test_size)
    test_df = df.loc[test_indices]
    train_df = df.drop(test_indices)
    return train_df, test_df
```

```
train_data, test_data = train_test_split(df, test_size=0.2)

def fit(df, target_attribute_name, features):
    return id3_algorithm(df, df, features, target_attribute_name, None)

def get_accuracy(df, tree):
    df["classification"] = df.apply(predict, axis=1, args=(tree, 'Yes'))
    df["classification_correct"] = df["classification"] == df["Play Tennis"]
    accuracy = df["classification_correct"].mean()
    return accuracy

tree = fit(train_data, 'Play Tennis', ['Weather', 'Temperature', 'Humidity', 'Windy'])
accuracy = get_accuracy(test_data, tree)
print("Decision Tree:")
print(tree)
print("Accuracy:", accuracy)
```

Limitaciones del algoritmo ID3 y los árboles de decisión:

- Si bien los árboles de decisión y el algoritmo ID3 ofrecen varias ventajas, también tienen algunas limitaciones que deben tenerse en cuenta antes de usarlos en ciertos escenarios:
  - **Sobreajuste**: Los árboles de decisión son propensos al sobreajuste, especialmente cuando se vuelven demasiado profundos o complejos. El sobreajuste ocurre cuando el árbol captura ruido o fluctuaciones aleatorias en los datos de entrenamiento, lo que resulta en un rendimiento deficiente con datos no visualizados.
  - **Inestabilidad**: Pequeños cambios en los datos pueden generar diferentes estructuras de árbol, lo que hace que los árboles de decisión sean menos estables. Una pequeña variación en los datos podría causar una división en un atributo o umbral diferente, lo que podría afectar a todo el árbol.
  - **Incapacidad para capturar relaciones lineales**: Los árboles de decisión no son adecuados para capturar relaciones lineales entre variables. Dividen los datos en regiones distintas, lo que dificulta la representación de patrones lineales.
  - **Sesgo hacia atributos con más niveles**: Los atributos con más niveles o categorías tienden a generar mayor ganancia de información simplemente por tener más divisiones posibles. Esto puede sesgar el árbol de decisión hacia dichos atributos, incluso si no son los más informativos.
  - **Falta de robustez al ruido**: los árboles de decisión pueden ser sensibles a los datos ruidosos, ya que podrían crear divisiones basadas en ruido o valores atípicos que no se generalizan bien a datos nuevos.

Limitaciones del algoritmo ID3 y los árboles de decisión:

- **Dificultad para manejar variables continuas**: El algoritmo ID3 y los árboles de decisión básicos están diseñados para manejar variables categóricas. Para las variables continuas, se requiere preprocesamiento para convertirlas en intervalos discretos o utilizar otros algoritmos como CART (árboles de clasificación y regresión).
- **Crecimiento exponencial del tamaño del árbol**: Los árboles de decisión pueden crecer rápidamente, especialmente al trabajar con grandes conjuntos de datos o un gran número de características. Esto puede generar árboles complejos y difíciles de interpretar.
- **Expresividad limitada**: si bien los árboles de decisión pueden representar límites de decisión simples, pueden tener dificultades para capturar relaciones complejas en los datos.
- **Dificultad para gestionar valores faltantes**: El algoritmo ID3 no gestiona adecuadamente los valores faltantes. Es necesario utilizar métodos de imputación u otros algoritmos para gestionar los datos faltantes.
- **Desequilibrio de clases**: los árboles de decisión pueden tener dificultades para lidiar con distribuciones de clases desequilibradas en los datos, potencialmente favoreciendo a la clase mayoritaria y teniendo un desempeño deficiente en la clase minoritaria.

- A pesar de estas limitaciones, los árboles de decisión y el algoritmo ID3 siguen siendo ampliamente utilizados y aún pueden ser eficaces en diversos escenarios, especialmente al combinarse con técnicas como la poda, métodos de conjunto (p. ej., bosques aleatorios, potenciación de gradiente) o al emplearse como parte de procesos de aprendizaje automático más sofisticados. Comprender las fortalezas y debilidades de los árboles de decisión ayuda a los científicos de datos y profesionales del aprendizaje automático a tomar decisiones informadas sobre su uso en diferentes aplicaciones.

Clasificación J48 (algoritmo C4.5) en pocas palabras

- Vivimos en un mundo de tecnología, e internet ha abierto las puertas a un vasto conocimiento e investigación al alcance de todos. La innovación y los avances continuos han abierto muchas puertas, lo que resulta en una enorme cantidad de datos útiles y significativos en todos los aspectos de nuestra vida. La pregunta es qué hacemos con estos datos, cómo los aprovechamos y cómo profundizamos en ellos para comprender su significado. Aquí es donde entra en juego el poder del aprendizaje automático y la inteligencia artificial. Estas técnicas ofrecen alternativas inteligentes al análisis de grandes volúmenes de datos. Mediante el desarrollo de algoritmos rápidos y eficientes, y modelos basados en datos para el procesamiento de datos en tiempo real, el aprendizaje automático puede producir resultados y análisis precisos.
- El algoritmo J48 es uno de los algoritmos de aprendizaje automático más utilizados para examinar datos de forma categórica y continua. El algoritmo C4.5 (J48) se utiliza principalmente en diversos campos para la clasificación de datos, por ejemplo, la interpretación de datos clínicos para el diagnóstico de enfermedades coronarias, la clasificación de datos de gobernanza electrónica y muchos más.

Clasificación J48 y su árbol de decisiones

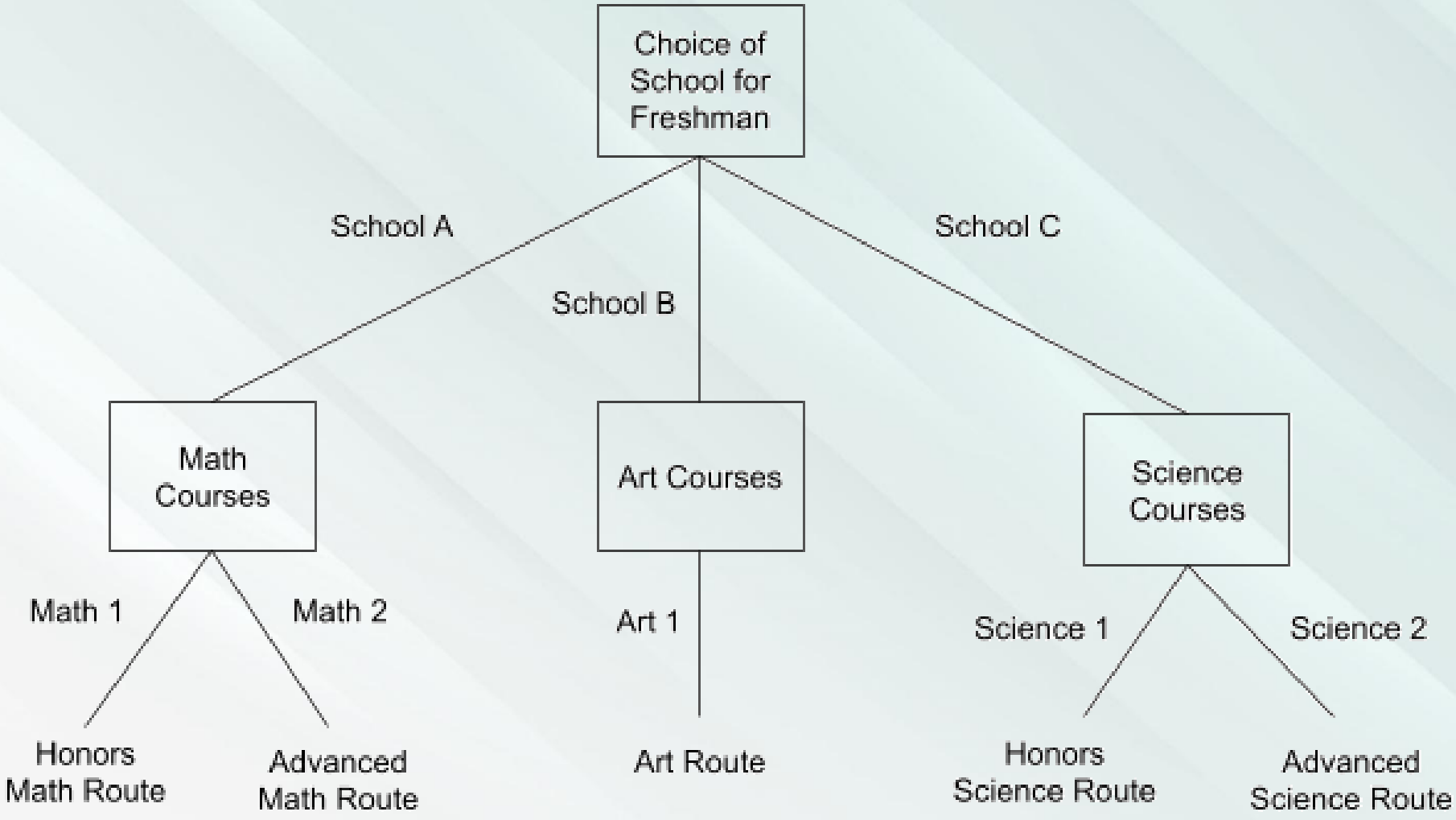
Algoritmo C4.5/J48

- El algoritmo C4.5 es un algoritmo de clasificación que genera árboles de decisión basados en la teoría de la información. Es una extensión del algoritmo ID3 de Ross Quinlan, también conocido en Weka como J48 (J de Java). Los árboles de decisión generados por C4.5 se utilizan para la clasificación, por lo que a menudo se le denomina clasificador estadístico.
- La implementación J48 del algoritmo C4.5 ofrece numerosas funciones adicionales, como la contabilización de valores faltantes, la poda de árboles de decisión, rangos continuos de valores de atributos, la derivación de reglas, etc. En la herramienta de minería de datos WEKA, J48 es una implementación Java de código abierto del algoritmo C4.5. J48 permite la clasificación mediante árboles de decisión o reglas generadas a partir de ellos.
- Este algoritmo construye árboles de decisión a partir de un conjunto de datos de entrenamiento, de forma similar al algoritmo ID3, utilizando el concepto de entropía de la información. Los datos de entrenamiento son un conjunto $S=\{s_1, s_2, \ldots\}$ de muestras ya clasificadas. Cada muestra si consiste en un vector p-dimensional $(x_1, i, x_2, i, \ldots, xp, i)$, donde xj representa los valores o características de los atributos de la muestra correspondiente, así como la clase a la que pertenece. Para obtener la máxima precisión de clasificación, el mejor atributo para dividir es aquel con la mayor información.

Clasificación J48 y su árbol de decisiones

- En cada nodo del árbol, el algoritmo C4.5 selecciona el atributo de los datos que divide mejor su conjunto de muestras en subconjuntos, enriquecidos en una u otra clase. El criterio de división es la ganancia de información normalizada, que se calcula a partir de la diferencia de entropía. Se selecciona el atributo con la mayor ganancia de información normalizada para tomar la decisión. A continuación, el algoritmo C4.5 recurre sobre las sublistas particionadas utilizando un enfoque de "divide y vencerás" y crea un árbol de decisión basado en el algoritmo voraz.
- En un ejemplo más concreto y de alto nivel, el algoritmo trabaja con el árbol de decisión. Para ello, analiza los datos de los estudiantes que ya han cursado dichos cursos y los utiliza para construir un modelo que predice qué cursará un futuro estudiante según su elección de universidad al comenzar su primer año. Tomamos nuestra lista de estudiantes y la dividimos aleatoriamente en conjuntos de datos. Con cada conjunto de datos, generamos un conjunto de ponderaciones que predicen la trayectoria del estudiante y, a continuación, seleccionamos el conjunto de datos que predice con mayor precisión qué cursará.

Clasificación J48 y su árbol de decisiones



Clasificación J48 y su árbol de decisiones

El árbol de decisiones

- Los nodos internos de este árbol de decisión representan los diferentes atributos, y las ramas entre los nodos indican los posibles valores que estos atributos pueden tener en las muestras observadas, mientras que los nodos terminales indican el valor final (clasificación) de la variable dependiente. El atributo predicho es la variable dependiente y los demás atributos del árbol son las variables independientes del conjunto de datos.
- Al construir un árbol, J48 ignora los valores faltantes; el valor de ese elemento puede predecirse a partir de los valores de los atributos de los demás registros. La idea básica es dividir los datos en rangos según los valores de los atributos presentes en la muestra de entrenamiento.
- Algunos casos de uso básicos de este algoritmo son:
  - Este algoritmo crea una lista para todas las muestras de una clase.
  - Evalúa características para evaluar cualquier ganancia de información, si no es posible ninguna ganancia, entonces este algoritmo crea un nodo más arriba en el árbol utilizando el valor esperado de la clase.
  - Si este algoritmo encuentra una clase desconocida que no ha visto hasta ahora, entonces crea un nodo más arriba en el árbol utilizando el valor esperado.

Clasificación J48 y su árbol de decisiones

- Los árboles de decisión se utilizan para delinear los procesos de toma de decisiones. Se trata de un clasificador que actúa como un diagrama de flujo, similar a la construcción de un árbol, para representar modelos de asociación. Los árboles de decisión se utilizan para categorizar instancias, ordenándolas desde el origen hasta un pequeño nodo hoja. Cada nodo especifica un análisis de la instancia, y cada división corresponde a uno de los posibles beneficios de este atributo.
- Divide un conjunto de datos en subconjuntos cada vez más pequeños y se desarrolla de forma incremental. El resultado final es un árbol con nodos de decisión y nodos hoja. Cada nodo de decisión tiene dos o más divisiones, y el nodo hoja representa una asociación o decisión. El nodo de decisión superior del árbol, que corresponde al mejor predictor, se denomina nodo de origen.

Clasificación J48 y su árbol de decisiones

Poda de árboles

- El sobreajuste es un riesgo importante con estos árboles de decisión, por lo que es vital validar el modelo antes de implementarlo en el conjunto de prueba. A medida que este árbol de decisión crece, tiende naturalmente a sobreajustarse a los datos. La poda es un proceso mediante el cual se selecciona el árbol más grande y más generalizable, y se eliminan todas las ramas por debajo de ese nivel. Esto mejora significativamente el rendimiento con nuevos datos.
- La herramienta WEKA para J48 ofrece varias opciones para la poda de árboles. En caso de un posible sobreajuste, la poda puede utilizarse como herramienta de resumen. En otros algoritmos, la clasificación se realiza recursivamente hasta que cada hoja sea pura; es decir, la clasificación de los datos debe ser lo más perfecta posible. Este algoritmo genera las reglas a partir de las cuales se genera la identidad de esos datos. El objetivo es la generalización progresiva de un árbol de decisión hasta alcanzar un equilibrio entre flexibilidad y precisión.

Clasificación J48 y su árbol de decisiones

Condiciones previas a la poda y de terminación

- El problema con la poda es que es bastante ineficiente, ya que se realiza después de construir el árbol de decisión, pudiendo estar sobreajustado. Por lo tanto, la siguiente mejora de este algoritmo sería podar el árbol durante su construcción. El concepto de prepoda consiste en finalizar el algoritmo antes de que el árbol de decisión sea demasiado grande o se convierta en un árbol completamente desarrollado. Esto evita el riesgo de sobreajuste y evita que los árboles de decisión se vuelvan demasiado complejos. Los objetivos principales de la prepoda de un árbol son detener el algoritmo si todas las muestras pertenecen a la misma clase o si todos los valores de los atributos son iguales. Sin embargo, estos dos puntos no son comunes y no abordan todos los aspectos del sobreajuste. Por lo tanto, existen tres condiciones de finalización que deben aplicarse al construir un árbol de decisión: la primera es detener el algoritmo si el tamaño de la muestra es demasiado pequeño, la segunda es limitar la profundidad del árbol para evitar que crezca demasiado, y la tercera es no dividir un nodo en ramas si no se reduce lo suficiente el error de clasificación. Con estas tres condiciones de terminación, el algoritmo J48 se vuelve mucho más eficiente y preciso ya que evita el sobreajuste del árbol de decisión.

Limitaciones

- Limitaciones de los árboles de decisión
- Si bien estos árboles de decisión son muy útiles para analizar y clasificar los datos, también presentan algunas desventajas asociadas con la poda. La poda reduce la complejidad del clasificador final y, por lo tanto, mejora la precisión predictiva al reducir el sobreajuste.
- Algunas de las cuestiones asociadas a estos árboles de decisión son:
  - El árbol en sí es computacionalmente costoso en cada nodo; cada candidato que divide el conjunto de datos debe ordenarse antes de poder encontrar su mejor nivel.
  - Además, los algoritmos de poda pueden resultar costosos, ya que es necesario industrializar y comparar innumerables subárboles candidatos.
  - La medición de la entropía consume muchos recursos.
  - El algoritmo del árbol de decisión produce un árbol de tamaño colosal.
  - Los tamaños de ejemplo minúsculos de un conjunto de entrenamiento plantean un desafío importante para los árboles de decisión, ya que la cantidad de instancias de entrenamiento obtenibles disminuye exponencialmente a medida que el árbol se divide.
  - Los criterios de decisión son rígidos en el sentido de que al final sólo se puede seleccionar un nodo.
  - Los árboles de decisiones profundos que abarcan innumerables niveles también pueden abarcar innumerables atributos, lo que genera valores de atributos atípicos.

Limitaciones

Limitaciones del algoritmo J48

- El algoritmo J48 se basa en árboles de decisión, por lo que presenta algunos de los problemas asociados a estos. A continuación, se presentan algunas deficiencias de este algoritmo:
  - **Ramas vacías**: La construcción de árboles con valores significativos es uno de los pasos importantes para la generación de reglas mediante el algoritmo J48. Sin embargo, estos valores no contribuyen a la creación de clases para las tareas de clasificación, sino que amplían y complican el árbol.
  - **Ramas insignificantes**: El número de atributos distintos seleccionados producen el mismo número de divisiones potenciales para construir un árbol de decisión. Sin embargo, no todas son significativas para las tareas de clasificación. Estas ramas insignificantes no solo reducen la usabilidad de los árboles de decisión, sino que también generan el problema del sobreajuste.
  - **Sobreajuste**: El sobreajuste ocurre cuando la visualización de un algoritmo obtiene información con atributos excepcionales. Esto causa muchas fragmentaciones, es decir, nodos estadísticamente poco importantes, en la distribución del proceso. Normalmente, el algoritmo J48 construye árboles y hace crecer sus ramas "lo suficientemente profundo como para clasificar perfectamente los ejemplos de entrenamiento", ya que este enfoque funciona mejor con datos sin ruido; sin embargo, la mayoría de las veces esta estrategia sobreajusta los ejemplos de entrenamiento con datos ruidosos. Actualmente, hay dos estrategias que se utilizan ampliamente para evitar este sobreajuste en el aprendizaje de árboles de decisión. La primera es que, si un árbol crece, se detenga su crecimiento antes de que alcance el punto máximo de clasificación precisa de los datos de entrenamiento. La segunda estrategia es dejar que el árbol se sobreajuste a los datos de entrenamiento y luego podarlo posteriormente.

Optimizaciones

Mejoras de J48 utilizando un mejor esquema de entropía

- El punto clave en el ensamblaje de árboles de decisión es la elección del mejor atributo para determinar el nivel del nodo actual. Para cada valor del conjunto de entrenamiento, se toma la mejor entropía calculada para determinar la ubicación del nodo. Los conjuntos de entrenamiento en general presentan algunas limitaciones, como: el conjunto de entrenamiento contiene una o más muestras que pertenecen a la misma clase, el conjunto de entrenamiento no contiene muestras y el conjunto de entrenamiento contiene muestras que pertenecen a una combinación de clases. Debido a estas limitaciones, el árbol de decisión resultante puede no representar con precisión la generalización; por lo tanto, se han ideado diferentes maneras de calcular la ganancia de entropía con mayor precisión. El algoritmo J48 puede mejorarse con un mejor cálculo de la ganancia de entropía.

Optimizaciones

Mejoras de J48 mediante el uso de clasificaciones híbridas no lineales

- La forma más común de construir árboles de decisión es mediante partición descendente. El conjunto de entrenamiento se utiliza recursivamente para encontrar una división lineal que maximice la ganancia de entropía. El árbol obtenido mediante este proceso suele ser demasiado grande y sobreajusta los datos, por lo que se poda examinando cada nodo intermedio y evaluando la utilidad de reemplazarlo con una hoja.
- Tras la poda, el conjunto local de casos de entrenamiento para un nodo hoja puede ser bastante amplio, y tomar solo la clase mayoritaria puede generalizar demasiado el árbol. La lógica J48 puede extenderse a partir del algoritmo básico de aprendizaje del árbol de decisión, de modo que, en lugar de la regla de la mayoría, se pueda utilizar un tipo de modelo diferente en cualquiera de las hojas. La decisión de reemplazar una hoja simple por un modelo alternativo se toma durante la pospoda. Los clasificadores alternativos de hojas se introducen principalmente durante la fase de pospoda. Las estimaciones de error se comparan antes de reemplazar un nodo hoja. Los algoritmos híbridos resultantes combinan los sesgos de la inducción descendente de árboles de decisión basada en la entropía y los respectivos modelos alternativos de hojas.

Optimizaciones

Mejoras de J48 mediante Meta Learning

- En la minería de datos, generalmente es necesario establecer los parámetros que utiliza el algoritmo para obtener el mejor modelo y los mejores resultados posibles. Los experimentos muestran un aumento sustancial en la precisión cuando se utilizan los parámetros correctos. Sin embargo, existe un problema asociado al ajuste de los parámetros de la mayoría de los algoritmos de minería de datos. Esta tarea puede implicar un alto costo computacional para encontrar los parámetros óptimos o, de lo contrario, correr el riesgo de basarse en suposiciones que podrían sesgar los resultados.
- La mayoría de los algoritmos y herramientas de minería de datos actuales requieren configuración previa. En otras palabras, los usuarios deben proporcionar valores adecuados para los parámetros con antelación para obtener buenos resultados o modelos; por lo tanto, deben poseer cierta experiencia para encontrar la configuración correcta. Para resolver este problema, la minería de datos permite aprender de ejecuciones anteriores de los algoritmos y así mejorar la selección futura de parámetros según el comportamiento previo del algoritmo.
- El metaaprendizaje es el estudio de métodos basados en principios que aprovechan el metaconocimiento para obtener modelos y soluciones eficientes mediante la adaptación del aprendizaje automático y la minería de datos. El modelo de árbol de decisión tiene parámetros que influyen en el grado de poda. Al podar árboles, se puede optimizar la eficiencia computacional y la precisión de clasificación del modelo.

Ejercicio 1: Clasificación de Frutas (D43)

- Clasificar frutas según su color y tamaño
- Datos de Entrenamiento (NO aleatorios):

Fruta	Color (1=Rojo, 2=Verde, 3=Amarillo)	Tamaño (1=Pequeño, 2=Mediano, 3=Grande)	Tipo
Manzana	1	2	Manzana
Manzana	2	2	Manzana
Manzana	1	3	Manzana
Naranja	3	2	Naranja
Naranja	3	3	Naranja
Plátano	3	3	Plátano
Plátano	3	2	Plátano
Uva	1	1	Uva
Uva	2	1	Uva

Resolución en Python

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

# Datos estructurados correctamente
X = [
    [1, 2], [2, 2], [1, 3],  # Manzanas
    [3, 2], [3, 3],          # Naranjas
    [3, 3], [3, 2],          # Plátanos
    [1, 1], [2, 1]           # Uvas
]

y = [
    'Manzana', 'Manzana', 'Manzana',
    'Naranja', 'Naranja',
    'Plátano', 'Plátano',
    'Uva', 'Uva'
]
```

```
# Crear y entrenar el modelo D43
modelo = DecisionTreeClassifier(criterion='entropy',
max_depth=3)
modelo.fit(X, y)

# Visualizar el árbol
plt.figure(figsize=(12, 8))
tree.plot_tree(modelo,
               feature_names=['Color', 'Tamaño'],
               class_names=['Manzana', 'Naranja', 'Plátano', 'Uva'],
               filled=True)
plt.show()

# Predecir una nueva fruta
nueva_fruta = [[3, 1]]  # Color amarillo, tamaño pequeño
prediccion = modelo.predict(nueva_fruta)
print(f"Predicción: {prediccion[0]}")
```

Ejercicio 2: Diagnóstico Médico (J48/C4.5)

- Objetivo: Diagnosticar enfermedades basado en síntomas

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Datos médicos realistas (no aleatorios)
datos_medicos = {
    'fiebre': [1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'tos': [1, 1, 0, 1, 0, 1, 1, 0, 0, 1],
    'dolor_cabeza': [0, 1, 0, 1, 1, 0, 1, 0, 0, 1],
    'edad': [25, 45, 30, 60, 35, 50, 28, 42, 33, 55],
    'enfermedad': ['Gripe', 'Gripe', 'Sano', 'COVID', 'Migraña',
                   'COVID', 'Migraña', 'Sano', 'Sano', 'COVID']
}

df = pd.DataFrame(datos_medicos)
print("Datos médicos:")
print(df)
```

Ejercicio 2: Diagnóstico Médico (J48/C4.5)

```
# Preparar datos
X = df[['fiebre', 'tos', 'dolor_cabeza', 'edad']]
y = df['enfermedad']

# Dividir en entrenamiento y prueba (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Modelo J48 (C4.5) - Usando entropía
modelo_j48 = DecisionTreeClassifier(
    criterion='entropy',  # Esto implementa C4.5/J48
    max_depth=4,
    min_samples_split=2,
    min_samples_leaf=1
)
```

```
# Entrenar
modelo_j48.fit(X_train, y_train)

# Evaluar
y_pred = modelo_j48.predict(X_test)
precision = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo: {precision:.2f}")

# Hacer predicción
nuevo_paciente = [[1, 1, 0, 40]]  # Fiebre, tos, sin dolor
cabeza, 40 años
diagnostico = modelo_j48.predict(nuevo_paciente)
print(f"Diagnóstico probable: {diagnostico[0]}")
```

UNIVERSIDAD
CATÓLICA
SEDES SAPIENTIAE