Desarrollo de contenido

Unidad 1

Programación para Análisis de Datos

Ingeniería de Software y Datos

IU Digital de Antioquia
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

Programación para Análisis de Datos

Presentación general de la asignatura

¡Bienvenido al curso Programación para Análisis de Datos!

En la era digital, la capacidad de extraer, analizar y aprovechar grandes volúmenes de datos se ha convertido en una habilidad crucial para cualquier profesional. Este curso te guiará a través de metodologías avanzadas y herramientas prácticas para transformar datos en información valiosa, que pueda impulsar decisiones estratégicas.

La primera unidad de este curso está diseñada para que domines la metodología CRISP-DM, un estándar en la minería de datos que te permitirá abordar proyectos de análisis de manera estructurada y eficiente. Además, aprenderás técnicas de Web Scraping utilizando herramientas como BeautifulSoup, Selenium y Scrapy para extraer datos de la web de manera automatizada. Estas habilidades son fundamentales en un entorno donde los datos en línea son cada vez más abundantes y relevantes para la toma de decisiones.

En la segunda unidad te prepararás para manejar y optimizar los procesos de desarrollo de *software*. Aprenderás a utilizar GitHub para el control de versiones, lo que es esencial para trabajar en equipo y asegurar la integridad de tus proyectos. También te introducirás en DevOps, una metodología que combina el desarrollo y las operaciones para acelerar la entrega de aplicaciones y servicios.

Por último, en la tercera unidad, te enfocarás en la optimización de procesos de desarrollo, explorando herramientas como GitHub Actions para la automatización y técnicas para la puesta en producción de servicios. Este enfoque te permitirá no solo proponer soluciones efectivas, sino también implementarlas de manera ágil y eficiente.

A lo largo del curso, desarrollarás habilidades técnicas y estratégicas que son altamente demandadas en el mercado laboral actual. Cada unidad del curso está diseñada para ofrecerte un aprendizaje práctico, con ejercicios y proyectos que simulan escenarios reales.

Te invitamos a participar activamente y a aprovechar al máximo esta oportunidad para impulsar tu carrera en el mundo de la analítica de datos.

Objetivos de aprendizaje

Aplicar técnicas algorítmicas para identificar y detallar estadísticamente los datos y su obtención.

Resultados de aprendizaje

Desarrollo algoritmos para la analítica de datos con uso de técnicas estadísticas de validación y generalidad sobre los datos.

Pregunta orientadora

En la era de la analítica de datos es fundamental garantizar que tus proyectos o en los que participas sean escalables, integrales y susceptibles de ser automatizados para el despliegue de productos y servicios. La escalabilidad asegura que los proyectos puedan crecer y adaptarse a medida que los conjuntos de datos aumentan en tamaño y complejidad. La integralidad implica abordar todas las facetas del análisis de datos, desde la adquisición y limpieza de datos hasta la implementación de modelos y la presentación de resultados. La automatización, por su parte, permite agilizar y optimizar los procesos, liberando recursos y reduciendo errores humanos.

En este contexto, la pregunta que orientará la reflexión de este curso es: ¿Qué tan seguro puedes estar de que tus proyectos de análisis de datos son escalables, integrales y susceptibles de ser automatizados para el despliegue de productos y servicios?

Esta pregunta propone mantener la atención para evaluar la capacidad y la viabilidad a largo plazo de los proyectos de análisis de datos en cualquier organización. La respuesta a esta pregunta proporciona información valiosa sobre la capacidad de una empresa para adaptarse a las demandas cambiantes del mercado y aprovechar al máximo el potencial de sus datos.

Mapa del curso

Programación para Análisis de Datos.

UNIDAD 1 Metodologías de análisis y herramientas de extracción de datos.

- Metodología CRISP-DM.
- Técnicas de web scraping (Beautifulsoup, Selenium, Scrapy).

UNIDAD 2 Herramientas y flujos de trabajo de desarrollo.

- Manejo y control de versiones por medio de Github.
- Optimización de procesos de desarrollo (DevOps).

UNIDAD 3 Optimización de procesos de desarrollo.

- Automatizacion (Github Actions).
- Puesta en producción (despliegue de servicios).

Cronograma de actividades del curso

Actividad de aprendizaje	Evidencia de aprendizaje	Semana	Ponderación
AA. Actividad de conocimientos previos.	EA. Prueba de conocimientos previos.	Semana 1	0 %
AA1. Metodologías de análisis y herramientas de extracción de datos.	EA1. Análisis y herramientas de extracción de datos.	Semanas 2 y 3	25 %
AA2. Herramientas y flujos de trabajo de desarrollo.	EA2. Aplicación de herramientas y flujos de trabajo.	Semanas 4 y 5	30 %
AA3. Optimización de procesos de desarrollo.	EA3. Optimización procesos de desarrollo.	Semanas 6 y 7	30 %
AA4. Actividad de cierre.	EA4. Proyecto integrador (propuesta de negocio).	Semana 8	15 %
Total			**100 %**
Actividad de refuerzo			**0 %**

Unidad 1. Metodologías de análisis y herramientas de extracción de datos

Introducción

En esta unidad desarrollaremos dos aspectos fundamentales que constituyen la base del análisis de datos moderno: las metodologías de análisis y las herramientas de extracción de datos, esenciales para cualquier profesional que desee transformar datos en información valiosa y accesible.

Específicamente, abordaremos la Metodología CRISP-DM (Cross-Industry Standard Process for Data Mining o Proceso Estándar para la Minería de Datos en Diversas Industrias), un modelo es ampliamente reconocido y utilizado en la industria por su enfoque estructurado y adaptable, dado que guía a los analistas de datos desde la comprensión del negocio hasta el despliegue de modelos de análisis. CRISP-DM es un método probado para orientar sus trabajos de minería de datos. Consta de seis fases principales: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. A lo largo de esta unidad, discutiremos en detalle cada una de estas fases, proporcionando ejemplos y mejores prácticas para su implementación efectiva.

En cuanto a las herramientas de extracción de datos, nos enfocaremos específicamente en las técnicas de web scraping, para que desarrolles habilidades que te permitan obtener datos de la web de manera automatizada. Aprenderemos a utilizar tres poderosas herramientas de scraping: BeautifulSoup, Selenium y Scrapy, efectivos para integrar datos extraídos de la web en tus análisis y obtener así resultados más completos y detallados.

Esta unidad sentará las bases para el resto del curso, proporcionándote las herramientas y técnicas necesarias para abordar proyectos de análisis de datos con confianza y precisión. ¡Presta mucha atención!

Resultados de aprendizaje de la *Unidad 1*

- Comprende la metodología CRISP-DM, aplicándola en propuestas de análisis de datos e identificando y ejecutando cada una de sus fases para estructurar y gestionar proyectos de manera efectiva.
- Entiende las diferentes librerías desde Python para realizar web scraping, extrayendo datos estructurados de sitios web estáticos mediante la navegación y búsqueda en documentos HTML y XML, aplicando selectores CSS (Cascading Style Sheets), que traduce Hojas de Estilo en Cascada, y técnicas de Parsing de datos, a partir de BeautifulSoup, Selenium y Scrapy.

Cronograma de actividades de la *Unidad 1*

Actividad de aprendizaje	Evidencia de aprendizaje	Semana	Ponderación
AA. Actividad de conocimientos previos.	EA. Prueba de conocimientos previos.	Semana 1	0 %
AA1. Metodologías de análisis y herramientas de extracción de datos.	EA1. Análisis y herramientas de extracción de datos.	Semanas 2 y 3	25 %
Total			**25 %**

Unidad 1. Actividad de aprendizaje 1. Metodologías de análisis y herramientas de extracción de datos

La metodología de CRISP-DM es muy popular y utilizada para guiar proyectos de minería de datos y análisis de datos. Desde su creación se ha convertido en un marco de referencia estándar en la industria para el desarrollo de proyectos de análisis de datos.

En el desarrollo temático de esta primera unidad iremos reconociendo su enfoque práctico y estructurado para el desarrollo de proyectos de análisis de datos. Al dividir el proceso en fases, podemos ayudar a los equipos a gestionar y estructurar proyectos de manera efectiva, lo que facilita la colaboración entre los diferentes roles y disciplinas involucradas en un proyecto de análisis de datos. Cada una de estas fases es crucial para garantizar que el proyecto no solo se enfoque en la construcción de modelos predictivos o descriptivos, sino también en alinearlos con los objetivos estratégicos del negocio.

Tema 1. Metodología CRISP-DM

CRISP-DM (Cross-Industry Standard Process for Data Mining) es una metodología estándar para el desarrollo de proyectos de minería de datos. Fue creada para proporcionar un marco estructurado que facilite la comprensión, ejecución y reproducción de proyectos de minería de datos en diversas industrias.

La metodología CRISP-DM es fundamental en proyectos de minería de datos debido a su estructura clara y flexible, que guía a los equipos desde la comprensión del problema de negocio hasta la implementación de soluciones basadas en datos.

Esta metodología estandariza el proceso de minería de datos en seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Al seguir estas fases, las organizaciones pueden asegurar una gestión efectiva del ciclo de vida del proyecto, facilitando la colaboración entre diferentes departamentos y profesionales.

Además, CRISP-DM es reconocida por su aplicabilidad en diversas industrias y tipos de proyectos, lo que permite su adaptación a diferentes contextos y necesidades específicas. Esta flexibilidad es esencial en un campo tan dinámico como la minería de datos, donde las tecnologías y las técnicas evolucionan constantemente. También promueve una documentación rigurosa y detallada, lo cual es vital para la reproducibilidad y la mejora continua de los procesos analíticos.

¿Sabías qué?

La metodología CRISP-DM fue desarrollada en 1996 por un consorcio de empresas europeas, incluyendo SPSS Inc. (ahora IBM), NCR Corporation, Daimler-Benz (ahora Daimler AG) y OHRA, como parte de un proyecto financiado por la Comisión Europea. El proyecto buscaba estandarizar y mejorar el proceso de minería de datos para hacerlo más accesible y eficiente.

1. Fases de CRISP-DM

En CRISP-DM, el proceso estándar para la minería de datos en diversas industrias se organiza en seis fases principales, cada una diseñada para guiar un aspecto específico del proyecto de minería de datos. A continuación, te presentamos una representación gráfica de las fases de CRISP-DM:

Figura 1. Fases CRISP-DM

Fase 1. Comprensión del negocio

En esta fase se establece una comprensión clara de los objetivos y requisitos del negocio y se definen los criterios de éxito. Se conocen los objetivos del negocio, y esos objetivos se convierten en un problema de minería de datos bien definido. Esta comprensión inicial asegura que el proyecto esté alineado con las necesidades y expectativas del negocio.

Al involucrar a las partes interesadas se comprende completamente el contexto empresarial y cómo se pueden aplicar las soluciones de minería de datos.

Definición de objetivos del negocio

La definición de los objetivos del negocio es un paso ineludible en la fase de comprensión del negocio dentro del ciclo de vida de un proyecto de minería de datos. Este proceso asegura que todos los esfuerzos de análisis y modelado estén alineados con las metas estratégicas del negocio, proporcionando un enfoque claro y direccional para el proyecto. A continuación, te contamos cómo definir los objetivos del negocio de manera efectiva:

- Identificación de problemas y objetivos clave: es el primer paso en cualquier proyecto de minería de datos.
- Evaluación de la situación: implica un análisis detallado de los recursos disponibles, el conocimiento previo y los requisitos necesarios para abordar el proyecto de minería de datos.
- Desarrollo del plan del proyecto: guiará la ejecución del proyecto de minería de datos desde el inicio hasta la finalización.

Caso de studio

Contexto del problema de E-Shop IUD

El E-Shop IUD ha observado una tasa de abandono de carritos de compra significativamente alta, lo que impacta negativamente en las ventas y en la satisfacción del cliente. Identificar los factores que contribuyen a este fenómeno y predecir cuándo es probable que un cliente abandone su carrito puede ayudar a implementar estrategias efectivas para reducir esta tasa.

¿Cuál es la solución óptima para el problema de minería de datos de E-Shop IUD?

Desarrollar un modelo que prediga las recomendaciones de productos basándose en el historial de compras de los clientes.

Implementar el modelo CRISP-DM para desarrollar un sistema predictivo que permita identificar los factores que influyen en el abandono de carritos de compra y predecir cuándo es probable que ocurra, con el fin de aplicar estrategias de retención y recuperación de ventas.

Requerimientos del negocio:

- Identificar los factores potenciales que influyen en el abandono de carritos, como precios, tiempos de carga, opciones de envío y experiencia del usuario.
- Definir los objetivos de reducción de abandono de carritos y mejorar la tasa de conversión.
- Recopilar datos históricos de la actividad de los clientes, incluyendo información sobre productos añadidos al carrito, tiempos de navegación y datos demográficos.
- Realizar un análisis exploratorio de los datos para identificar patrones y tendencias.
- Limpiar y transformar los datos, eliminando inconsistencias y datos faltantes.
- Seleccionar y generar características relevantes que pueden influir en el abandono de carritos.
- Entrenar y validar los modelos con los datos recopilados.

Fase 2. Comprensión de los datos

Durante esta fase se recopilan y exploran los datos disponibles para el proyecto. Los equipos analizan la calidad de los datos, identifican los problemas y realizan una exploración inicial para comprender la estructura y el contenido de los datos. Esta fase establece una base sólida para las siguientes etapas del proceso.

Para comprender los datos se deben considerar las siguientes etapas:

- Recolección de datos iniciales: al recolectar y explorar los datos disponibles se obtiene una visión clara de su calidad y características.
- Descripción de los datos: esta etapa implica la presentación detallada y el análisis inicial de los datos recolectados para entender sus características básicas.
- Exploración de los datos: es una etapa crítica que implica un análisis detallado y exhaustivo de los datos recolectados.
- Verificación de la calidad de los datos: implica la identificación y corrección de problemas como valores faltantes, duplicados y datos inconsistentes.

Caso de estudio

Volvamos al caso de la empresa de comercio electrónico E-Shop IUD y enfoquémonos en comprender los datos asociados a ella.

Lo primero que debemos hacer es la recopilación inicial de datos:

- Datos de historial de compras de los clientes (transacciones).
- Información demográfica de los clientes.
- Datos de navegación en el sitio web (páginas visitadas, tiempo en cada página, etc.).

Cuando tengamos esa información, podemos proceder con el análisis de datos, para ello podemos:

- Analizar la distribución de compras por categoría de producto.
- Identificar patrones de compra (como los productos comprados juntos).
- Verificar la calidad de los datos (valores nulos, inconsistencias, duplicados).

¿Qué se debe hacer en E-Shop IUD para comprender los datos?

Verificar cuántos clientes compran productos electrónicos y luego accesorios relacionados.

Fase 3. Preparación de los datos

En esta fase se limpian, transforman y preparan los datos para el modelado. Esto puede incluir tareas como la limpieza de datos incorrectos o faltantes, la selección de variables relevantes, la transformación de datos en un formato adecuado y la integración de datos de diferentes fuentes. Una preparación de datos efectiva es importante para construir modelos de minería de datos precisos y confiables.

Aquí se detallan los aspectos clave para esta fase:

- Selección de datos: implica identificar y elegir aquellos que son pertinentes para el análisis específico que se desea realizar.
- Limpieza de datos: es fundamental para garantizar la calidad y fiabilidad de los datos utilizados en el análisis. Uno de los aspectos clave de esta etapa es el manejo de valores faltantes, duplicados y errores.

- Construcción de datos: en esta etapa se lleva a cabo la derivación de nuevas variables y la transformación de datos con el objetivo de enriquecer y preparar los datos para el análisis.
- Integración de datos: en esta etapa se aborda la combinación de datos procedentes de diversas fuentes para crear un conjunto unificado y completo que permita un análisis integral.
- Formateo de datos: esta etapa se enfoca en la estructuración y organización de los datos de manera que sean adecuados para el análisis posterior, asegura que los datos estén en el formato correcto y cumplan con los requisitos específicos de las técnicas de modelado y herramientas de análisis utilizadas.

Caso de estudio

¿Cómo preparar los datos para atender la necesidad que tenemos en E-Shop IUD?

Debemos comenzar con la **selección de datos,** para ello podemos filtrar datos relevantes para el modelo, como ID de cliente, ID de producto, fecha de compra, cantidad, etc.

Luego, procedemos con la **limpieza de datos**, en la que podemos realizar las siguientes acciones:

- Manejar valores nulos en las transacciones.
- Eliminar duplicados.
- Normalizar los datos (por ejemplo: categorías de productos en un formato uniforme).

Continuamos con la **construcción de datos**, para ello podemos crear características derivadas, como *total gastado por cliente* o *frecuencia de compra por categoría*.

Finalmente realizamos la **integración de datos**, en la que debemos combinar datos de diferentes fuentes (transacciones, demografía, navegación).

¿Qué se debe hacer en E-Shop IUD?

Crear una tabla de datos que incluya el historial de compras de cada cliente, junto con sus características demográficas y datos de navegación.

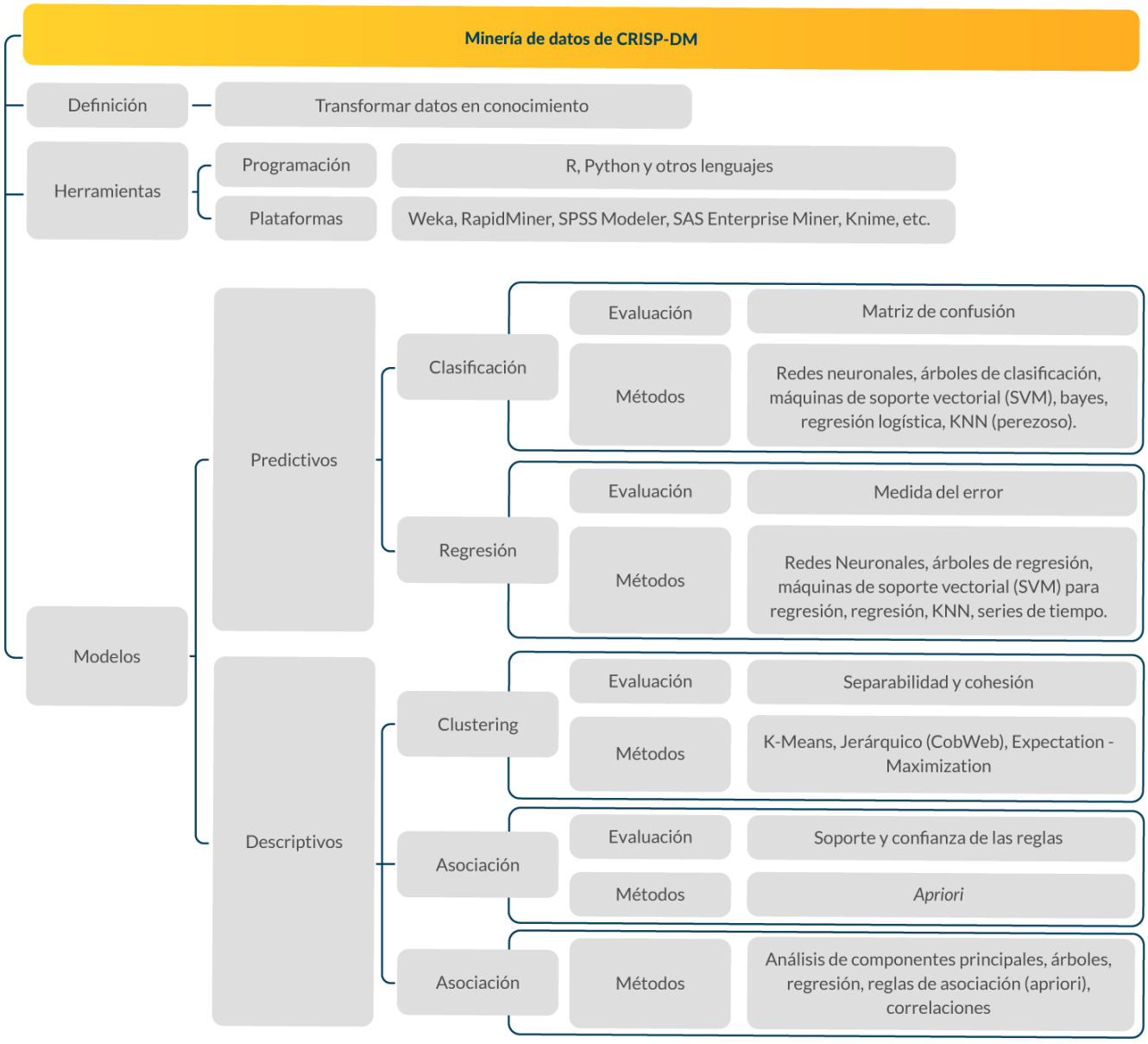
Fase 4. Modelado

Durante esta fase se construyen modelos predictivos o descriptivos utilizando técnicas como regresión, clasificación y otras técnicas de aprendizaje automático y estadísticas y se ajustan sus parámetros para mejorar su rendimiento.

Se hace la identificación y evaluación de los algoritmos y métodos más apropiados para abordar el problema específico que se está enfrentando. Esto implica considerar diversos factores, como la naturaleza de los datos disponibles, los objetivos del proyecto y las métricas de evaluación relevantes.

Los algoritmos y técnicas para esta fase de modelado deben ser capaces de manejar la estructura y características de los datos de manera efectiva. Por ejemplo, para problemas de clasificación, podrían considerarse algoritmos como SVM (Support Vector Machines), árboles de decisión o redes neuronales, dependiendo de la complejidad y dimensionalidad de los datos. Para problemas de regresión, podrían evaluarse modelos como regresión lineal, regresión logística o modelos de ensamble similares a Random Forest.

Figura 2. Modelos de análisis CRISP-DM



Fuente: adaptado de MathWorks (s. f.)

Además de la elección del algoritmo, es necesario considerar técnicas de preprocesamiento de datos como la normalización, la reducción de dimensionalidad y la selección de características para optimizar el rendimiento del modelo y asegurar la generalización de datos nuevos. La selección de técnicas de modelado en CRISP-DM proporciona el fundamento para construir modelos efectivos que puedan convertir datos en conocimientos accionables para la organización o proyecto en cuestión, sin embargo, estas técnicas de aprendizaje automático serán tema de desarrollo en el curso de machine learning.

Diseño de pruebas

El diseño de pruebas es una etapa fundamental dentro de la fase de modelado, donde se verifica la capacidad del modelo para generalizar y predecir con precisión datos no vistos. Este proceso comienza con la creación de conjuntos de entrenamiento y prueba que son representativos de la población de datos. El conjunto de entrenamiento se utiliza para ajustar el modelo, mientras que el conjunto de prueba se reserva para evaluar su desempeño.

Durante el diseño de pruebas se aplican técnicas como la validación cruzada para evaluar la robustez del modelo frente a diferentes particiones de los datos. Esto ayuda a detectar problemas de sobreajuste (*overfitting*) o subajuste (*underfitting*) y permite ajustar los parámetros del modelo de manera adecuada. Además, se implementan métricas de evaluación como precisión, recall, F1-score, entre otras, dependiendo del tipo de problema que se esté abordando (clasificación, regresión, etc.), para cuantificar el rendimiento del modelo de manera objetiva.

El diseño de pruebas efectivo en CRISP-DM garantiza que el modelo final seleccionado sea capaz de generalizar bien a datos nuevos y no vistos, proporcionando confianza en las conclusiones y recomendaciones derivadas del análisis de datos**.**

Las etapas claves en el diseño de pruebas son:

- Construcción del modelo: en esta etapa se implementan las técnicas de modelado seleccionadas previamente para abordar el problema específico de minería de datos.
- Evaluación del modelo: es la etapa donde se analiza la capacidad predictiva y la eficacia general del modelo construido. Durante esta etapa se utilizan técnicas y métricas específicas para medir la precisión, el rendimiento y la generalización del modelo en datos nuevos o no vistos.

Caso de estudio

Para el caso de E-Shop, en la **selección de técnicas de modelado**, se debe desarrollar algoritmos de recomendación, como filtrado colaborativo o modelos basados en contenido.

Para **aplicar modelos** se podría:

- Entrenar un modelo de filtrado colaborativo utilizando matrices de usuario-producto.
- Entrenar un modelo basado en contenido utilizando características de productos y clientes.

Cuando lo anterior esté listo se procederá con la **calibración de parámetros**, es decir, el ajuste de los parámetros del modelo para mejorar sus medidas de desempeño.

¿Cómo realizar el modelado en E-Shop?

Podríamos entrenar un modelo de filtrado colaborativo que recomiende productos basados en compras anteriores de clientes similares.

Fase 5. Evaluación

En esta fase los modelos construidos se evalúan en función de los criterios establecidos en la fase de comprensión del negocio. Se realizan pruebas de rendimiento y se comparan los modelos para seleccionar el más adecuado. Los modelos también se validan utilizando datos independientes para asegurar su generalización y robustez. En esta fase se realiza:

- Revisión del proceso: permite identificar mejoras potenciales y errores en el desarrollo del proyecto de minería de datos. Durante esta parte de la etapa se analizan exhaustivamente los resultados obtenidos del modelo construido en la fase anterior, evaluando su capacidad para cumplir con los objetivos de negocio y las expectativas iniciales.
- Determinación de los próximos pasos: tras la evaluación exhaustiva del modelo y la revisión del proceso, es crucial determinar los próximos pasos

estratégicos para asegurar la efectividad y el éxito continuo del proyecto de minería de datos.

Caso de estudio

Si fuésemos a **evaluar modelos para el problema de E-Shop**, sería recomendable:

- Medir la precisión del modelo utilizando un conjunto de datos de validación.
- Utilizar métricas para la medición de desempeño de las variables categóricas y continuas.

Además, sería necesario **decidir los próximos pasos**:

- Si el modelo cumple con los objetivos del negocio, proceder al despliegue.
- Si no, iterar sobre las fases anteriores (ajustar el modelo, recolectar más datos, etc.).

¿Qué debemos hacer en esta fase para el caso de E-Shop?

Evaluar el modelo de recomendaciones midiendo cuántas de las recomendaciones realizadas coinciden con las compras reales posteriores de los clientes.

Fase 6. Despliegue

En la fase final del proceso CRISP-DM se implementan los modelos seleccionados en el entorno de producción, se presentan los resultados del análisis de datos a las partes interesadas y se integran en los sistemas existentes de la organización. Además, se desarrollan estrategias para monitorear y mantener los modelos en producción para garantizar su efectividad continua.

Veremos unos aspectos importantes de esta fase:

- Planificación del despliegue: en esta etapa se desarrolla un plan detallado para la implementación de los modelos o soluciones derivadas del análisis de datos. Esto puede incluir la definición de recursos necesarios, el cronograma de implementación y la asignación de responsabilidades.
- Monitoreo y mantenimiento: son aspectos importantes una vez que un modelo de minería de datos se despliega en producción. El seguimiento del

rendimiento del modelo implica la recolección continua de datos sobre su comportamiento operativo y su precisión predictiva.

- Generación de informes: en el contexto de la minería de datos, se usa para comunicar eficazmente los resultados y las conclusiones obtenidas a las partes interesadas. Estos informes deben ser claros, concisos y adaptados al público objetivo, que puede incluir desde ejecutivos y tomadores de decisiones hasta analistas y equipos técnicos.
- Documentación de experiencias y mejores prácticas: los participantes aprenderán a planificar y ejecutar el despliegue de soluciones de minería de datos de manera efectiva en entornos de producción. Se enfocarán en garantizar la calidad y el rendimiento continuo del modelo a través del monitoreo, la actualización y la comunicación efectiva de los resultados a las partes interesadas.

Caso de estudio

La **implementación del modelo** en E-Shop IUD implica:

- Integrar el modelo de recomendaciones en el sitio web de E-Shop.
- Mostrar recomendaciones personalizadas en las páginas de producto y en el carrito de compras.

Además, es necesario**:**

- Monitorear el rendimiento del modelo en producción.
- Actualizar el modelo regularmente con nuevos datos de compras.

¿Cómo se puede implementar el modelo en E-Shop IUD?

A través de la implementación de un sistema en el sitio web que muestre tres productos recomendados en la página del carrito de compras basados en el historial de compras del cliente y datos de navegación.

¿Cómo monitorear el modelo que se implemente en E-Shop IUD?

- Midiendo el impacto en las ventas y el valor promedio de las órdenes.
- Realizando A/B tests para comparar la efectividad de las recomendaciones personalizadas vs. recomendaciones genéricas.

Ten en cuenta que…

Aquí finalizamos la exploración de las 6 fases de CRISP-DM, las cuales proporcionan una guía estructurada y sistemática para gestionar proyectos de minería de datos desde el inicio hasta la implementación, asegurando que los objetivos del negocio se cumplan y se maximice el valor de los datos.

Ciclo de vida del proyecto de minería de datos

En el ciclo de vida de un proyecto de minería de datos, las iteraciones y la retroalimentación son fundamentales para garantizar la mejora continua y la adaptación a medida que avanza el proyecto.

Las iteraciones implican repetir las fases del ciclo (comprensión, preparación, modelización, evaluación e implementación) con el objetivo de refinar y mejorar tanto los datos como los modelos desarrollados. Durante cada iteración se recopilan nuevos datos, se ajustan los modelos existentes según los hallazgos de evaluaciones anteriores, y se exploran diferentes técnicas o algoritmos para optimizar los resultados.

La retroalimentación juega un papel importante en momento de proporcionar información de las partes interesadas y de los usuarios finales, así como de los resultados obtenidos en la implementación del modelo en producción. Esta retroalimentación se utiliza para validar y ajustar las decisiones tomadas durante el ciclo de vida del proyecto. Además, permite corregir errores, mejorar la precisión de los modelos y asegurar que los objetivos del negocio se estén cumpliendo de manera efectiva. Así, las iteraciones y la retroalimentación no solo facilitan la mejora continua del proyecto, sino que también aseguran que las soluciones de minería de datos sean relevantes y efectivas en el contexto empresarial.

Tema 2. Técnicas de web scraping (Beautifulsoup, Selenium y Scrapy)

Antes de revisar con profundidad las técnicas de web scraping, es importante explorar una herramienta que usaremos en el desarrollo de esta unidad.

¿Qué es Google Colaboratory?

Google Colaboratory, comúnmente conocido como Google Colab, es una herramienta de desarrollo proporcionada por Google que permite escribir y ejecutar código en Python directamente en el navegador. Es especialmente popular en la comunidad de ciencia de datos y aprendizaje automático por su simplicidad y facilidad de uso, además de ser gratuito y accesible sin necesidad de configuraciones complicadas.

Google Colab funciona sobre una plataforma de Jupyter Notebooks, que son documentos que pueden contener tanto código ejecutable como texto enriquecido, también párrafos, ecuaciones, imágenes y gráficos. Los Notebooks de Colab se ejecutan en la nube, lo que permite a los usuarios aprovechar el poder computacional de los servidores de Google sin la necesidad de contar con hardware potente localmente.

Ten en cuenta que…

El desarrollo de las actividades lo vamos a realizar en Jupyter Notebook y Google Colab y, para ello, es necesario que tengas claro lo siguiente:

- Jupyter Notebook es una aplicación web que permite crear y compartir documentos que contienen código, ecuaciones, visualizaciones y texto narrativo. Es ampliamente utilizado en ciencia de datos, análisis de datos, machine learning y educación.
- Google Colab, por su parte, es un servicio gratuito de Google que permite ejecutar Jupyter Notebooks en la nube. Ofrece acceso a recursos de computación gratuitos como GPUs y TPUs, facilitando el desarrollo y la

ejecución de proyectos de machine learning y análisis de datos sin necesidad de configurar un entorno local.

Ahora sí, exploremos el web scraping.

Las técnicas de web scraping son métodos utilizados para extraer información de sitios web de manera automatizada. Este proceso implica el uso de herramientas y bibliotecas como BeautifulSoup, Selenium y Scrapy para navegar, seleccionar y recuperar datos estructurados de páginas web. BeautifulSoup facilita la extracción de datos mediante el análisis del HTML, Selenium permite interactuar con páginas dinámicas simulando la navegación del usuario, y Scrapy ofrece una plataforma robusta para la recopilación de datos a gran escala a través de spiders configurables. Estas técnicas son esenciales para la recopilación de datos en proyectos de análisis, investigación de mercado, monitoreo de precios y más, permitiendo a los desarrolladores automatizar la recolección de grandes volúmenes de información de manera eficiente y precisa.

1. Generalidades sobre web scraping

El web scraping es una técnica utilizada para extraer datos de sitios web de manera automatizada. Consiste en el uso de programas o scripts que navegan por páginas web y recuperan información específica contenida en ellas, como texto, imágenes y otros elementos HTML. Esta práctica permite recopilar grandes volúmenes de datos de manera rápida y eficiente, lo cual es útil para diversas aplicaciones como análisis de datos, investigación de mercado, monitoreo de precios y creación de bases de datos. Sin embargo, es importante considerar los aspectos legales y éticos del web scraping, respetando las políticas de uso de los sitios web y la privacidad de los datos.

La legalidad del web scraping es un tema complejo y depende de varios factores, incluyendo la jurisdicción, el propósito del scraping y las políticas del sitio web en cuestión. En general, scraping de datos de un sitio web sin permiso puede violar los términos de servicio del sitio y potencialmente infringir derechos de autor o derechos de privacidad. Algunos países tienen leyes específicas que regulan el acceso y uso de datos en línea, lo que puede incluir

sanciones para el scraping no autorizado. Sin embargo, el web scraping es legal si se realiza con el consentimiento del propietario del sitio o si los datos están disponibles públicamente sin restricciones legales. Es crucial que los practicantes de web scraping comprendan y respeten las regulaciones aplicables y consideren obtener el permiso explícito cuando sea necesario.

Ética y políticas de uso de los sitios web

Cuando se realiza web scraping, es fundamental adherirse a la ética y las políticas de uso de los sitios web. Las políticas de uso de un sitio web, a menudo detalladas en sus términos de servicio (ToS) o en su archivo robots.txt, establecen las reglas sobre cómo se puede interactuar con el contenido del sitio. El archivo robots.txt, ubicado en la raíz del sitio web, indica qué partes del sitio son accesibles para los bots y crawlers. Ignorar estas directrices no solo puede ser considerado una infracción ética, sino también una violación legal.

Ética y legalidad están estrechamente relacionadas en el web scraping. Éticamente, es importante respetar la propiedad intelectual de los sitios web, no sobrecargar los servidores con solicitudes excesivas, y utilizar los datos obtenidos de manera responsable y conforme a los objetivos declarados. En la práctica, esto significa que se deben evitar prácticas como el scraping agresivo que puede afectar el rendimiento del sitio web, el uso de datos para actividades maliciosas o engañosas, y la ignorancia de las restricciones explícitas establecidas por el propietario del sitio.

En resumen, los practicantes de web scraping deben actuar con integridad y respeto, asegurándose de cumplir con las normativas y estándares de uso de los sitios web para mantener la confianza y la legalidad en sus actividades de scraping.

2. Conceptos básicos para explorar web scraping

HTML y estructura de páginas web

La estructura básica de HTML (HyperText Markup Language) define cómo se organiza y presenta el contenido en una página web. Consiste en elementos fundamentales que ayudan a estructurar y formatear el contenido de manera semántica y visualmente coherente. Un documento HTML comienza típicamente con la declaración <!DOCTYPE html> que especifica la versión de HTML que se utiliza, seguido por el elemento <html> que encierra todo el contenido de la página. Dentro del <html>, se encuentran dos secciones principales: <head> y <body>. El <head> contiene metainformación sobre el documento, como el título de la página, enlaces a hojas de estilo CSS, scripts JavaScript y otros metadatos. Mientras que el <body> contiene el contenido visible de la página, estructurado mediante elementos como , <p>, <h1> a <h6> para párrafos y encabezados respectivamente, <a> para enlaces, <img> para imágenes, y muchos otros elementos según las necesidades de diseño y contenido. Esta estructura básica proporciona un marco flexible para crear páginas web interactivas y bien organizadas.

Navegación y selección de elementos

La navegación y selección de elementos en HTML se realiza principalmente utilizando selectores CSS y XPath. Los selectores CSS permiten seleccionar elementos basados en sus atributos, clases o identificadores, utilizando sintaxis familiar a los estilos de CSS. Por ejemplo, para seleccionar todos los elementos <a> dentro de un con la clase "menu", se utiliza el selector div.menu a.

Por otro lado, XPath proporciona una forma más precisa y flexible de navegar por el DOM (Document Object Model) de HTML. Utiliza rutas para seleccionar nodos en un documento XML o HTML, permitiendo seleccionar elementos según su posición relativa, contenido o atributos específicos. Un ejemplo de XPath sería //div[@id='content']/p, que selecciona todos los elementos <p> dentro de un con el id "content".

Ambos métodos son fundamentales para interactuar con los elementos de una página web durante procesos como web scraping o manipulación dinámica de contenido. La elección

entre CSS y XPath depende de la complejidad de la estructura HTML y las necesidades específicas del desarrollo o automatización que se esté realizando.

Las herramientas de desarrollo del navegador son indispensables para cualquier desarrollador web, ya que facilitan la inspección, depuración y optimización de sitios y aplicaciones web. Estas herramientas, accesibles mediante teclas de acceso rápido o haciendo clic derecho en cualquier parte de una página web y seleccionando "Inspeccionar" o "Inspect", proporcionan una interfaz interactiva que muestra el DOM (Document Object Model), los estilos CSS aplicados, las solicitudes de red y más.

3. Herramientas de web scraping

Comparación entre BeautifulSoup, Selenium y Scrapy

Al realizar web scraping, es esencial seleccionar la herramienta adecuada según las necesidades específicas del proyecto. BeautifulSoup, Selenium y Scrapy son tres de las herramientas más populares, y cada una ofrece ventajas particulares. Veamos:

- **BeautifulSoup** es una biblioteca de Python que facilita la extracción de datos de archivos HTML y XML, es ideal para proyectos pequeños a medianos debido a su simplicidad y facilidad de uso, pero no maneja bien el contenido dinámico.
- **Selenium** es un framework para automatizar navegadores web, es capaz de interactuar con contenido dinámico, lo que lo hace ideal para páginas web que requieren interacción con JavaScript, pero puede ser más lento y consume más recursos en comparación con otras herramientas.
- **Scrapy** es un framework completo para web scraping que permite la extracción de datos a gran escala. Está diseñado para ser eficiente y rápido, permitiendo manejar tareas complejas de scraping y crawling. Scrapy es muy potente y escalable, pero puede requerir una curva de aprendizaje más pronunciada en comparación con BeautifulSoup.

En resumen, la elección de la herramienta depende del tipo y la complejidad del proyecto: BeautifulSoup para scraping simple y estático, Selenium para contenido dinámico y páginas interactivas, y Scrapy para grandes proyectos y necesidades avanzadas de scraping.

BeautifulSoup

Es una biblioteca de Python utilizada para extraer datos de archivos HTML y XML. Proporciona métodos sencillos y potentes para navegar, buscar y modificar el árbol de análisis del documento (parse tree), lo que la hace especialmente útil para el web scraping. BeautifulSoup permite convertir datos semiestructurados en datos estructurados de una manera sencilla y eficiente.

Instalación de BeautifulSoup

Para instalar BeautifulSoup, puedes usar pip, el gestor de paquetes de Python. La instalación incluye tanto BeautifulSoup como el parser lxml que se recomienda para un mejor rendimiento y manejo de errores. Puedes realizar el proceso de instalación y configuración de BeautifulSoup siguiendo el proceso que te presentamos a continuación:

Código en Python

```bash
pip install beautifulsoup4 lxml
```

Fin código

Configuración de BeautifulSoup

Una vez instalado, BeautifulSoup se puede configurar importando la biblioteca en tu script de Python y cargando el contenido HTML que deseas analizar.

Código en Python

```python
from bs4 import BeautifulSoup
import requests

# Obtener el contenido HTML de una página web
url = 'https://www.example.com'
response = requests.get(url)
html_content = response.content

# Crear una instancia de BeautifulSoup
soup = BeautifulSoup(html_content, 'lxml')
```

Fin código

Sintaxis básica y uso de BeautifulSoup

La sintaxis básica de BeautifulSoup incluye métodos para buscar y navegar por el árbol de análisis del documento.

1. Encontrar elementos:

Puedes usar `find` y `find_all` para buscar elementos HTML específicos:

Código en Python

```python
# Encontrar el primer elemento <a>
first_link = soup.find('a')

# Encontrar todos los elementos <a>
all_links = soup.find_all('a')
```

Fin Código

2. Navegar por el árbol de análisis:

Puedes navegar a través de los nodos hermanos, padres y descendientes:

Código en Python

```python
# Obtener el padre del primer enlace
parent = first_link.parent

# Obtener todos los hijos de un elemento <div>
div = soup.find('div')
children = div.find_all(recursive=False)
```

Fin Código

3. Acceder a atributos y contenido de texto:

Puedes extraer atributos y texto de los elementos:

Código en Python

```python
# Extraer el atributo 'href' del primer enlace
link_href = first_link['href']

# Extraer el texto del primer enlace
link_text = first_link.get_text()
```

Fin código

4. **Uso de selectores CSS:**

También puedes usar selectores CSS para buscar elementos:

Código en Python

```python
# Encontrar todos los elementos con la clase 'example'
elements = soup.select('.example')
```

Fin código

Ejemplo completo

Verás en este momento un ejemplo completo que muestra cómo usar BeautifulSoup para extraer datos de una página web:

Código en Python

```python
from bs4 import BeautifulSoup
import requests

# URL de la página web a analizar
url = 'https://www.example.com'

# Realizar la solicitud HTTP para obtener el contenido de la página
response = requests.get(url)

# Crear una instancia de BeautifulSoup con el contenido HTML
soup = BeautifulSoup(response.content, 'lxml')

# Encontrar todos los enlaces en la página
links = soup.find_all('a')

# Imprimir los atributos 'href' y el texto de cada enlace
for link in links:
    href = link.get('href')
    text = link.get_text()
    print(f'Texto: {text}, URL: {href}')
```

```

Fin código

## Navegación y búsqueda en el DOM

BeautifulSoup facilita la navegación y búsqueda en el Document Object Model (DOM) de documentos HTML y XML mediante métodos como find() y find_all() que permiten localizar elementos específicos, y select() que utiliza selectores CSS. La biblioteca también ofrece herramientas para acceder a los contenidos y atributos de los nodos, así como para navegar por la jerarquía del DOM mediante propiedades como .parent, .children, .next_sibling y .previous_sibling. Estos métodos y propiedades permiten extraer y manipular datos de manera eficiente y estructurada, simplificando el proceso de web scraping.

## Extracción de datos

La extracción de datos con BeautifulSoup implica el uso de métodos y propiedades para acceder y obtener información específica de un documento HTML o XML. Utilizando métodos como find() y find_all() para localizar elementos y select() para aplicar selectores CSS, se puede extraer el texto de los nodos con .text o .get_text(), y obtener atributos de los elementos con .get(). Este proceso permite recolectar datos estructurados y no estructurados de manera eficiente, facilitando su posterior análisis o almacenamiento en bases de datos o archivos.

¡Atrévete a poner en práctica lo aprendido! Realiza el primer reto formativo de la unidad y aplica tus conocimientos y habilidades en BeautifulSoup. ¡No te pierdas esta oportunidad de aprender haciendo!

## Reto formativo 1

Propósito: con el ejercicio práctico, aprenderás a utilizar BeautifulSoup, una poderosa biblioteca de Python para web scraping, incluyendo su instalación, sintaxis básica, métodos
```

de navegación y búsqueda en el DOM, así como técnicas para extraer y manipular datos de una página web. Además, te permitirá ampliar los conocimientos adquiridos en un contexto real de web scraping.

Instrucciones:

- Ingresa a un Jupyter Notebook Colab.
- Sigue las instrucciones de la practica descargando el siguiente ejercicio: https://drive.google.com/file/d/1fVjGRdqVLDvAQLfRSgpq7rQqY9B11Uvt/view?usp=drive_link
- Utiliza BeautifulSoup para extraer los datos de una página web que elijas.
- Aplica los conceptos aprendidos en un escenario real de web scraping que elijas utilizando BeautifulSoup.

Selenium

Selenium es una suite de herramientas para la automatización de navegadores y aplicaciones web, pero también es muy útil para tareas de web scraping que requieren interacción con elementos dinámicos y JavaScript. Selenium soporta varios lenguajes de programación como Python, Java, C#, Ruby y más, permitiendo la creación de scripts que controlan un navegador web como si un usuario estuviera interactuando con él.

Instalación de Selenium

La instalación de Selenium es fundamental para aquellos que desean automatizar pruebas en aplicaciones web. Utilizando bibliotecas específicas según el lenguaje de programación elegido, se puede facilitar la ejecución.

Para instalar Selenium en Python, puedes usar `pip`, el gestor de paquetes de Python:

Código Python
```bash
pip install selenium
```

```
Fin Código
```

Además, necesitarás descargar el WebDriver correspondiente al navegador que planeas usar (Chrome, Firefox, etc.). Por ejemplo, para Chrome necesitarás [ChromeDriver:](https://sites.google.com/a/chromium.org/chromedriver/downloads) .

Configuración de Selenium

Una vez que hayas instalado Selenium y descargado el WebDriver, necesitarás configurarlo en tu script. Aquí te mostramos cómo hacerlo para Chrome:

1. 1. Descargar ChromeDriver:** Aasegúrate de que el ChromeDriver está en tu `PATH` o proporciona la ruta directamente en tu script.
2. 2. Configuración del WebDriver en Python:

Código Python
```python
from selenium import webdriver

# Ruta al ChromeDriver descargado
driver_path = '/ruta/al/chromedriver'

# Inicializar el WebDriver
driver = webdriver.Chrome(executable_path=driver_path)
```
Fin Código

Sintaxis básica y uso

Selenium permite realizar diversas acciones en el navegador. A continuación, te mostramos algunos ejemplos básicos:

1. Abrir una página web:

   Código Python
   ```python
   driver.get('https://www.ejemplo.com')
   ```
   Fin Código

2. Encontrar elementos:

   Código Python
   ```python
   # Encontrar un elemento por su nombre
   element = driver.find_element_by_name('q')

   # Encontrar un elemento por su ID
   element = driver.find_element_by_id('searchInput')

   # Encontrar elementos por su clase
   elements = driver.find_elements_by_class_name('example-class')

   # Encontrar elementos usando selectores CSS
   element = driver.find_element_by_css_selector('.class_name')
   ```
   Fin Código

3. Interacción con elementos:
   Código Python
   ```python
   # Escribir en un campo de texto
   element.send_keys('Texto de búsqueda')

   # Hacer clic en un botón
   button = driver.find_element_by_css_selector('button[type="submit"]')
   button.click()
   ```
   Fin Código

4. **Esperas:**

Para manejar elementos dinámicos que pueden no estar inmediatamente disponibles, es útil la acción utilizar esperas:

Código Python
```python
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Espera hasta que el elemento esté presente
element = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, 'element_id'))
)
```
Fin Código

5. **Cierre del navegador:**

Código Python
```python
driver.quit()
```
Fin Código

Automatización del navegador

Selenium permite la automatización completa del navegador, simulando la interacción de un usuario real con una página web. Esto incluye navegar a diferentes URL, rellenar y enviar formularios, hacer clic en botones y enlaces, e incluso desplazarse por páginas largas. Selenium es capaz de manejar elementos dinámicos mediante el uso de esperas implícitas y explícitas, garantizando que las interacciones se realicen solo cuando los elementos están presentes y listos. Su capacidad para automatizar navegadores hace que Selenium sea una herramienta poderosa para tareas de pruebas automatizadas y web scraping en aplicaciones que dependen de JavaScript y requieren una interacción compleja con la interfaz de usuario.

Esperas y manejo de elementos dinámicos

En Selenium, el manejo de elementos dinámicos son variables que determinan la interacción con páginas web que cargan contenido de manera asincrónica. Las esperas implícitas hacen que Selenium espere un tiempo determinado antes de lanzar una excepción si un elemento no se encuentra disponible inmediatamente. Por otro lado, las esperas explícitas permiten definir condiciones específicas bajo las cuales Selenium debe esperar, como la presencia de un elemento o su estado de visibilidad, antes de continuar con la ejecución del script. Esto asegura que las interacciones con la página web sean fiables y que los scripts no fallen debido a la carga tardía de los elementos.

¡Pon en práctica lo aprendido! Realiza el segundo reto formativo de la unidad.

Reto formativo 2

Propósito: realizar un ejercicio práctico para aplicar los conocimientos adquiridos en un contexto real de web scraping utilizando Selenium, una herramienta poderosa para la automatización del navegador en Python, incluyendo su instalación, configuración, métodos para la navegación web y la interacción con elementos de la página.

Instrucciones:

- Ingresa a un Jupyter Notebook Colab.
- Sigue las instrucciones de la practica descargando el siguiente ejercicio: https://drive.google.com/file/d/1apVbcMjxU4-3-3XoTqHLKt-hqC_g_3mV/view?usp=drive_link
- Utiliza Selenium para automatizar la extracción de datos de una página web que elijas.
- Aplica los conceptos aprendidos en un escenario práctico de web scraping que elijas utilizando Selenium.

Scrapy

Es un framework de scraping web y crawling utilizado para extraer datos de sitios web de manera rápida, eficiente y escalable. Desarrollado en Python, Scrapy permite definir spiders que navegan por las páginas web y extraen información estructurada que puede ser

almacenada en varios formatos como JSON, CSV o bases de datos. Es especialmente útil para proyectos grandes y complejos debido a su capacidad de manejar solicitudes asíncronas y seguir enlaces en múltiples páginas.

Instalación

La instalación de Scrapy es esencial para aquellos que desean realizar web scraping de manera eficiente y estructurada. Se puede lograr fácilmente utilizando herramientas como pip para Python, y una vez configurado, Scrapy permite navegar por sitios web de forma programática.

Para instalar Scrapy, necesitas tener Python y pip instalados en tu sistema. Luego, puedes instalar Scrapy usando pip:

Código Python
```sh
pip install scrapy
```
Fin código

Configuración

1. Crear un proyecto Scrapy:

Primero, crea un nuevo proyecto Scrapy utilizando el comando `scrapy startproject` seguido del nombre del proyecto.

Código Python
```sh
scrapy startproject nombre_proyecto
```
Fin Código

Esto creará una estructura de directorios con los archivos necesarios.

2. Defineir un Spider:

Dentro de tu proyecto, define un spider que se encargará de navegar y extraer datos de las páginas web. Los spiders se colocan en el directorio `spiders`.

Código Python
```sh
cd nombre_proyecto
scrapy genspider nombre_spider dominio.com
```
Fin código

Sintaxis bBásica y Uuso

Un spider básico en Scrapy podría verse así:

Código Python
```python
import scrapy

class MiSpider(scrapy.Spider):
    name = "mi_spider"
    start_urls = [
        'http://ejemplo.com',
    ]

    def parse(self, response):
        for titulo in response.css('h2::text'):
            yield {'titulo': titulo.get()}

        for siguiente in response.css('a.next::attr(href)'):
            yield response.follow(siguiente, self.parse)
```
Fin Código

Desglose de sintaxis básica:

1. Importar Scrapy**: `import scrapy` es necesario para acceder a las funcionalidades de Scrapy.
2. Definir la clase del Spider: `class MiSpider(scrapy.Spider)` define un nuevo spider heredando de `scrapy.Spider`.
3. Nombre del Spider: `name = "mi_spider"` da un nombre único a tu spider.
4. URLs iIniciales: `start_urls` es una lista de URLs desde donde el spider empezará a rastrear.
5.Método `parse`: Este método procesa cada respuesta obtenida de las URLs iniciales. Utiliza selectores CSS para extraer datos y sigue enlaces para navegar a otras páginas.

Ejecución del Spider

Para ejecutar el spider, utiliza el siguiente comando desde la raíz del proyecto:

Código Python
```sh
scrapy crawl mi_spider
```
Fin Código

Creación de un proyecto Scrapy

Para iniciar un proyecto en Scrapy, primero asegúrate de tener Python y pip instalados. Comienza abriendo tu terminal y ejecutando el comando `scrapy startproject nombre_proyecto`, donde "nombre_proyecto" es el nombre que le darás a tu proyecto. Este comando creará una estructura de directorios organizada que incluye carpetas y archivos esenciales para tu proyecto, como `spiders`, donde definirás tus spiders, y archivos de configuración como `settings.py`. Una vez creado el proyecto, navega al directorio del proyecto con `cd nombre_proyecto` y podrás comenzar a definir tus spiders utilizando el comando `scrapy genspider nombre_spider dominio.com`, que genera un archivo base para

tu spider dentro del directorio `spiders`. Este enfoque estructurado facilita la gestión y escalabilidad de tus proyectos de web scraping.

¡Realiza el último reto formativo de la unidad!

Reto formativo 3

Propósito: Realizar un ejercicio práctico para aplicar los conocimientos adquiridos en un contexto real de web scraping utilizando Scrapy, un framework de web scraping para Python, incluyendo su instalación, configuración, y cómo crear y ejecutar proyectos de Scrapy para extraer datos de diferentes fuentes web.

Instrucciones:

- Ingresa a un Jupyter Notebook Colab.
- Sigue las instrucciones de la practica descargando el siguiente ejercicio: https://drive.google.com/file/d/1diE5uLUie24Undgm5bNYKfk3U9WcKZfQ/view?usp=drive_link
- Utiliza Scrapy para extraer datos de una o varias páginas web que elijas.
- Aplica los conceptos aprendidos en un escenario práctico de web scraping que elijas utilizando Scrapy.

4. Manejo y almacenamiento de datos

Para entender por qué es crucial estudiar el manejo y almacenamiento de datos en el contexto del scraping, hay que reconocer que la calidad y disponibilidad de los datos son fundamentales para cualquier proyecto de extracción y análisis de información web. A medida que exploramos cómo Scrapy, BeautifulSoup o Selenium pueden obtener datos valiosos de la web, es esencial asegurar que estos datos se almacenen de manera estructurada y segura. Además, la limpieza y transformación de estos datos son pasos críticos para garantizar su integridad y utilidad en análisis posteriores. Este conocimiento te permite obtener datos de manera efectiva, gestionarlos de manera eficiente y prepararlos para su análisis y aplicación práctica en diversos contextos.

Limpieza y transformación de datos

La limpieza y transformación de datos son procesos fundamentales en cualquier proyecto de análisis de datos, incluyendo el web scraping. Estos procesos aseguran que los datos obtenidos estén libres de errores, inconsistencias y sean estructurados de manera adecuada para análisis posteriores. La limpieza de datos involucra la identificación y corrección de valores incorrectos, valores faltantes o duplicados, así como la normalización de formatos para facilitar su procesamiento uniforme. Por otro lado, la transformación de datos implica la modificación o derivación de nuevas variables a partir de los datos existentes, como la conversión de formatos, cálculos de nuevas métricas o agregaciones necesarias para el análisis específico del proyecto. Estos procesos son esenciales para garantizar la calidad y utilidad de los datos antes de su utilización en modelos analíticos o reportes finales.

Almacenamiento de datos

Después de recolectar, limpiar y transformar los datos, es vital contar con un método efectivo para guardarlos de manera segura y accesible. Existen varias opciones para almacenar datos extraídos, dependiendo de la cantidad de datos, la frecuencia de actualización y los requisitos específicos del proyecto.

Una opción común es el almacenamiento en archivos como CSV o JSON, que son formatos simples y ampliamente compatibles que permiten almacenar datos estructurados de manera fácilmente legible. Esto es útil para proyectos más pequeños o donde la portabilidad y la simplicidad son prioritarias.

Otra alternativa es el almacenamiento en bases de datos, como SQLite o MongoDB. Estas bases de datos permiten almacenar grandes volúmenes de datos de manera eficiente y ofrecen capacidades avanzadas para consultas y manipulación de datos. SQLite es ideal para aplicaciones de tamaño medio que no requieren un servidor de base de datos, mientras que MongoDB es una opción robusta para grandes volúmenes de datos no estructurados o semiestructurados.

Además, el uso de sistemas de almacenamiento en la nube, como Amazon S3 o Google Cloud Storage, proporciona escalabilidad y acceso global a los datos, facilitando la colaboración y el procesamiento distribuido en entornos de análisis de datos complejos.

Buenas prácticas de scraping

Las buenas prácticas en el scraping de datos son esenciales para realizar extracciones de manera ética, eficiente y efectiva. A continuación, te presentamos algunas buenas prácticas que te serán de utilidad:

1. Conocer y respetar los términos de servicio y las políticas de cada sitio web objetivo. Esto incluye revisar si permiten el scraping y bajo qué condiciones. Además, es recomendable limitar la frecuencia de las solicitudes para no sobrecargar los servidores del sitio, utilizando tiempos de espera entre cada petición para simular el comportamiento humano.
2. Identificar claramente el bot mediante un User-Agent apropiado y personalizado. Esto facilita la comunicación con los administradores del sitio en caso de problemas y ayuda a mantener la transparencia sobre el origen de las solicitudes. Es fundamental también manejar correctamente las sesiones y las cookies, especialmente en sitios que requieren autenticación o conservación de estado.
3. Respetar las directrices del archivo robots.txt del sitio es otra norma ética importante, ya que este archivo indica qué secciones deben ser accedidas o no por bots. Esto ayuda a evitar problemas legales y técnicos al tiempo que se respeta la voluntad del propietario del sitio.
4. Evitar impactar negativamente en el rendimiento del sitio web objetivo, evitando el scraping en momentos de alta demanda o carga. Monitorear y manejar los errores también es esencial para asegurar la estabilidad de los scripts de scraping y la confiabilidad de los datos extraídos.
5. Cumplir con las leyes y regulaciones de protección de datos aplicables, especialmente al manejar información personal o sensible. Esto puede incluir obtener consentimiento explícito cuando sea necesario y asegurarse de no infringir los derechos de privacidad de los usuarios.

Siguiendo estas buenas prácticas se promueve un scraping responsable que no solo cumple con las normativas legales y éticas, sino que también fomenta relaciones positivas con los propietarios de los sitios web y minimiza los riesgos de bloqueo o sanciones.

Optimización y rendimiento

La optimización y el rendimiento en el contexto del web scraping son fundamentales para mejorar la eficiencia y la efectividad de las extracciones de datos. Para comenzar, es diseñar scripts de scraping que sean lo más eficientes posible en términos de recursos computacionales y tiempo de ejecución. Esto implica optimizar la selección de elementos HTML mediante selectores específicos y directos, evitando consultas redundantes o innecesarias que puedan ralentizar el proceso.

Además, es recomendable implementar técnicas de paralelización para procesar múltiples solicitudes de manera simultánea. Esto se puede lograr utilizando bibliotecas como concurrent.futures en Python o empleando el multiprocessing para ejecutar tareas en paralelo. Al distribuir la carga de trabajo entre varios hilos o procesos, se puede acelerar significativamente el tiempo total de extracción de datos, aprovechando al máximo los recursos disponibles.

Otro aspecto importante es el manejo adecuado de los tiempos de espera y los intervalos entre solicitudes. Ajustar estos tiempos de manera inteligente puede ayudar a evitar ser detectado como un bot y ser bloqueado por los sistemas de defensa del sitio web objetivo. Utilizar tiempos de espera aleatorios y ajustables puede simular el comportamiento humano y reducir las posibilidades de ser identificado como un scraper automatizado.

Además de la optimización técnica, es importante considerar la optimización del diseño del sistema de scraping. Esto incluye el uso eficiente de la memoria y la gestión adecuada de los datos extraídos, evitando la sobrecarga de almacenamiento o la pérdida de información relevante.

Manejo de errores y excepciones

El manejo de errores y excepciones en el web scraping para garantizar la estabilidad y fiabilidad de los scripts automatizados. Dado que el scraping implica interactuar con recursos externos y sitios web que pueden cambiar estructuras o estar sujetos a

interrupciones, es imprescindible implementar estrategias robustas para manejar posibles errores.

Una práctica común es capturar excepciones específicas que pueden surgir durante el scraping, como errores de conexión, tiempo de espera excedido, o cambios en la estructura de la página web. Utilizando bloques try-except, es posible detectar estos errores y manejarlos adecuadamente, por ejemplo, realizando nuevas tentativas de conexión o ajustando la lógica de extracción para adaptarse a cambios en el sitio web.

Además, es recomendable implementar mecanismos para registrar y monitorear errores durante la ejecución del script. Esto permite identificar y corregir rápidamente problemas recurrentes, mejorando así la robustez del proceso de scraping a lo largo del tiempo.

Otro aspecto importante del manejo de errores es la gestión de excepciones relacionadas con la legalidad y la ética del scraping. Es esencial respetar las políticas de uso de los sitios web objetivo y manejar de manera adecuada los casos de bloqueo o restricción de acceso. Esto puede incluir la implementación de mecanismos para ajustar dinámicamente los tiempos de espera y las solicitudes, así como el uso de proxies o herramientas de rotación de IP para evitar ser detectado y bloqueado por los sitios web.

Cierre de la *Unidad 1*

¡Buen trabajo! A lo largo de esta unidad, hemos explorado temas fundamentales en el campo del análisis de datos y la extracción de información: la metodología CRISP-DM y las técnicas de web scraping utilizando herramientas como BeautifulSoup, Selenium y Scrapy.

Juntos hemos recorrido un camino significativo al aprender las bases de la metodología CRISP-DM y las técnicas de web scraping. Estas habilidades son esenciales para cualquier profesional en el campo del análisis de datos y la inteligencia artificial. No obstante, el viaje no termina aquí.

En las siguientes unidades de este curso y en cursos posteriores, te sumergirás en temas aún más profundos y emocionantes. Explorarás técnicas avanzadas de modelado y análisis predictivo, aprenderás a manejar grandes volúmenes de datos con tecnologías de Big Data y descubrirás cómo aplicar machine learning para resolver problemas complejos en diversas industrias.

Nos vemos en la *Unidad 2*.

Referencias de imágenes

- Figura 1. SketchBubble (s. f.). Crisp DM PowerPoint and Google Slides Template. https://www.sketchbubble.com/en/presentation-crisp-dm.html

- Figura 2. MathWorks (s. f.). Cómo funciona Machine Learning. https://la.mathworks.com/discovery/machine-learning.html

Bibliografía

- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. y Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide.* The CRISP-DM Consortium. https://mineracaodedados.wordpress.com/wp-content/uploads/2012/12/crisp-dm-1-0.pdf

- Kouzis, D. (2018). *Learning Scrapy*. Packt Publishing. https://github.com/gaudimark/books/blob/master/Programm/python/Learning%20Scrapy.pdf

- Lawson, R. (2015). *Web scraping with Python: Using BeautifulSoup and Scrapy*. Packt Publishing. https://github.com/MishRanu/scrapy-practice/blob/master/Richard%20Lawson-Web%20Scraping%20with%20Python-Packt%20Publishing%20(2015).pdf

- Mitchell, R. (2015). *Web scraping with Python: Collecting data from the modern web*. O'Reilly Media. https://github.com/boydfd/books/blob/master/seeing/underway/OReilly.Web.Scraping.with.Python.Collecting.Data.from.the.Modern.Web.1491910291.pdf

- Russell, M. A. (2013). *Mining the social web: Data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more.* O'Reilly Media. https://www.webpages.uidaho.edu/~stevel/504/mining-the-social-web-2nd-edition.pdf

- Shearer, C. (2000). The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing, 5*(4), 13-22. https://mineracaodedados.wordpress.com/wp-content/uploads/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf

- Vanden Broucke, S. y Baesens, B. (2018). *Practical web scraping for data science: Best practices and examples with Python. Apress*. https://github.com/Apress/practical-web-scraping-for-data-science

- Wirth, R. y Hipp, J. (2000). *CRISP-DM: Towards a standard process model for data mining*. https://cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf

IU Digital de Antioquia

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

www.**iudigital**.edu.co