Lenguajes de la ciencia de datos

Hay literalmente miles de lenguajes de programación diferentes con sus propias fortalezas y debilidades. Entonces, ¿qué lenguaje deberías aprender primero? La respuesta a esa pregunta depende en gran medida de sus necesidades, los problemas que está tratando de resolver, y para quién los está resolviendo. Primero y, ante todo, recomendamos considerar los lenguajes Python, R, SQL y LAN. Sin embargo, otros lenguajes como Scala, Java, C++, y Julia con características específicas también son populares.

El idioma que elijas aprender dependerá de las cosas que necesites lograr y los problemas que necesites resolver. Los problemas pueden estar relacionados con la empresa para la que trabajas, el puesto que desempeñas, y la antigüedad de tu solicitud actual.

Lenguaje Python

Es el lenguaje de programación más utilizado y popular en la industria de la ciencia de datos. Glassdoor informó que, en 2019, más del 75 % de los puestos de ciencia de datos enumerados incluían Python en sus descripciones de trabajo. Python es ideal porque utiliza una sintaxis clara y legible. Puedes desarrollar los mismos programas desde otros lenguajes con menos código usando Python.

Python es útil en muchas áreas, incluyendo ciencia de datos, IA y aprendizaje automático, desarrollo web, y dispositivos de Internet de las Cosas o IoT.

Python es un lenguaje de programación de alto nivel y propósito general que se puede aplicar a muchas clases diferentes de problemas. Tiene una gran biblioteca estándar que proporciona herramientas adecuadas para muchas tareas diferentes, incluidas, entre otras, bases de datos, automatización raspado web, procesamiento de texto, procesamiento de imágenes, aprendizaje automático, y análisis de datos. Para la ciencia de datos, puede utilizar las bibliotecas de computación científica de Python como Pandas, NumPy, SciPy, y Matplotlib. Para inteligencia artificial, cuenta con TensorFlow, PyTorch, Keras, y Scikit-learn. Python también se puede utilizar para el procesamiento de lenguaje natural, o NLP, utilizando el Natural Language Toolkit, o NLTK.

El lenguaje Python tiene un código de conducta ejecutado por la Python Software Foundation que busca garantizar la seguridad y la inclusión de todos en las comunidades de Python tanto en línea como en persona.

PyLadies es un grupo de mentoría internacional con un enfoque en ayudar a que más mujeres se conviertan en participantes activas y líderes en la comunidad de código abierto de Python.

Lenguaje R

Python es de código abierto, mientras que R es software libre. el software de código abierto y el software libre son gratuitos para usar. Ambos se refieren comúnmente al mismo conjunto de licencias. Por ejemplo, muchos proyectos de código abierto utilizan la Licencia Pública General, o GNU. Ambos apoyan la colaboración, y, en muchos casos, estos términos pueden usarse indistintamente, pero no todos, sus diferencias son, la Iniciativa de Código Abierto, o OSI, defiende el código abierto, mientras que la Fundación para el Software Libre, o FSF, define el software libre. El código abierto está más centrado en el negocio, mientras que el software libre está más centrado en un conjunto de valores.

Deberías aprender R porque es software libre. Puedes utilizar el lenguaje de la misma manera que contribuyes al código abierto. Además, permite el uso privado, el uso comercial y la colaboración pública.

Los estadísticos, matemáticos, y mineros de datos utilizan R para desarrollar software estadístico, gráficos y análisis de datos.

La sintaxis orientada a matrices de los lenguajes R facilita la traducción de matemáticas a código para estudiantes sin conocimientos de programación o con conocimientos mínimos de programación.

R tiene más de 15,000 paquetes publicados, lo que hace posible realizar Análisis exploratorio de datos complejos.

R se integra bien con otros lenguajes informáticos como C++, Java, C, .NET, y Python.

Usando R, operaciones matemáticas comunes como la multiplicación de matrices dan resultados inmediatos.

R tiene facilidades de programación orientada a objetos más fuertes que la mayoría de los lenguajes de computación estadística.

Lenguaje SQL

SQL es diferente de otros lenguajes de desarrollo de software porque es un lenguaje no procedimental. Su alcance se limita a la consulta y gestión de datos. Si bien no es un lenguaje de ciencia de datos, los científicos de datos lo utilizan regularmente porque es simple y poderoso. Es unos 20 años más antiguo que Python y R. Apareció por primera vez en 1974 y fue desarrollado en IBM. SQL fue diseñado para gestionar datos en Base de datos relacional.

Una base de datos relacional está formada por colecciones de tablas bidimensionales, por ejemplo, conjuntos de datos y hojas de cálculo de Excel. Cada una de estas tablas está formada entonces por un número fijo de columnas y cualquier número posible de filas.

Aunque SQL se desarrolló originalmente para su uso con bases de datos relacionales, debido a su omnipresencia y facilidad de uso, también se han desarrollado interfaces SQL para muchos repositorios NoSQL y de big data.

El lenguaje SQL se subdivide en varios elementos del lenguaje que incluyen cláusulas, expresiones, predicados consultas, y declaraciones.

Conocer SQL te ayudará a conseguir muchos trabajos diferentes en ciencia de datos, como analista de negocios y de datos. Este conocimiento también es imprescindible en la ingeniería de datos. Al realizar operaciones con SQL, se accede a los datos directamente sin necesidad de copiarlos por separado, lo que puede acelerar considerablemente las ejecuciones del flujo de trabajo.

SQL se comporta como un intérprete entre usted y la base de datos. SQL es un estándar del Instituto Nacional Estadounidense de Estándares, o ANSI, lo que significa que, si aprende SQL y lo usa con una base de datos, puede aplicar su conocimiento de SQL a muchas otras bases de datos fácilmente.

Hay muchas bases de datos SQL diferentes disponibles, incluyendo las siguientes: MySQL, IBM DB2, PostgreSQL, Apache Open Office Space, SQLite, Oracle, MariaDB, Microsoft SQL Server, y más. La sintaxis del SQL que escribas puede cambiar según el Sistema de administración de bases de datos relacionales que estés utilizando. Si desea aprender SQL, debe centrarse en una base de datos relacional específica.

Otros lenguajes

Scala, Java, C++ y Julia son probablemente los lenguajes de ciencia de datos más tradicionales. Sin embargo, JavaScript, PHP, Go, Ruby, Visual Basic y muchos otros han encontrado su lugar en la comunidad de ciencia de datos.

Java es un lenguaje de programación orientado a objetos de uso general y probado. Tiene una gran adopción en el ámbito empresarial y fue diseñado para ser rápido y escalable. Las aplicaciones Java se compilan en código de bytes y se ejecutan en la máquina virtual Java o JVM. Algunas herramientas de ciencia de datos notables creadas con Java incluyen Weka para minería de datos, Java ML para aprendizaje automático, Apache ML Lib, que hace que el aprendizaje automático sea escalable, y Deep Learning 4 para aprendizaje profundo.

Hadoop es otra aplicación de Java que gestiona el procesamiento y almacenamiento de datos para aplicaciones de big data que se ejecutan en sistemas agrupados.

Scala es un lenguaje de programación de propósito general que brinda soporte para la programación funcional y es un sólido sistema de tipos estáticos. El lenguaje Scala fue construido para abordar las deficiencias de Java. También es

interoperable con Java ya que se ejecuta en la JVM. El nombre Scala es una combinación de escalable y lenguaje. Este lenguaje está diseñado para evolucionar con los requisitos de sus usuarios. Para la ciencia de datos, el programa más popular creado con Scala es Apache Spark. Spark es un sistema informático en clúster rápido y de propósito general que proporciona API que facilitan la escritura de trabajos paralelos. Tiene un motor optimizado que soporta gráficos computacionales generales. Spark incluye Shark, que es un motor de consulta, MLlib para aprendizaje automático, GraphX para procesamiento de gráficos, y Spark Streaming. Fue diseñado para ser más rápido que Hadoop.

C++ es un lenguaje de programación de propósito general. Es una extensión del lenguaje de programación C, o C con clases. C++ mejora la velocidad de procesamiento, permite la programación del sistema, y proporciona un control más amplio sobre la aplicación de software. Muchas organizaciones que utilizan Python u otros lenguajes de alto nivel para el análisis de datos y tareas de exploración dependen de C++ para desarrollar programas que suministran datos a los clientes en tiempo real Para la ciencia de datos, TensorFlow es una popular biblioteca de aprendizaje profundo para Dataflow que fue creada con C++, y aunque C++ es la base de TensorFlow, se ejecuta en una interfaz de Python, por lo que los usuarios no necesitan conocimientos de C++ para usarlo.

MongoDB es una base de datos NoSQL para la Administración de datos que fue construida con C++. La IU es intuitiva. Y Caffe es un repositorio de algoritmos de aprendizaje profundo creado con C++ con enlaces Python y MATLAB. Una tecnología fundamental para la World Wide Web.

JavaScript es un lenguaje de propósito general que se extendió más allá del navegador con la creación de Node.js y otros enfoques del lado del servidor. JavaScript no está relacionado con el lenguaje Java. Para la ciencia de datos, sin duda, TensorFlow.js es la implementación más popular. TensorFlow.js hace posible el aprendizaje automático y el aprendizaje profundo en Node.js, así como en el navegador. TensorFlow.js también fue adoptado por otras bibliotecas de código abierto, incluidas Brain.js y MachineLearn.js. Otra implementación de JavaScript para la ciencia de datos es RJS. El proyecto RJS ha reescrito las especificaciones de álgebra lineal del lenguaje R en TypeScript. Esto sienta las bases para que proyectos futuros implementen marcos basados en matemáticas más potentes como NumPy y SciPy de Python.

TypeScript es un superconjunto de JavaScript.

Julia fue diseñada en el MIT para el análisis numérico de alto rendimiento y la ciencia computacional. Julia proporciona un desarrollo rápido como Python o R y al mismo tiempo produce programas que se ejecutan tan rápido como los programas C o Fortran. Se compila, lo que significa que el código Julia se ejecuta directamente en el procesador como código ejecutable. Llama a las bibliotecas C, Go, Java, MATLAB, R, Fortran y Python y tiene paralelismo refinado. Julia como lenguaje, fue escrito en 2012, pero tiene un gran potencial para su impacto futuro en la industria de la ciencia de datos. Y una gran aplicación de Julia para la ciencia de datos es

JuliaDB, que es un paquete para trabajar con conjuntos de datos grandes y persistentes.

Biblioteca para la ciencia de datos

Las bibliotecas de computación científica contienen módulos integrados que proporcionan diferentes funcionalidades, que puedes utilizar directamente, también se les llama marcos.

Pandas ofrece estructuras de datos y herramientas para la limpieza, manipulación, y análisis de datos efectivos. Proporciona herramientas para trabajar con diferentes tipos de datos. El instrumento principal de Pandas es una tabla bidimensional que consta de columnas y filas denominada marco de datos. Pandas también puede proporcionar una indexación sencilla para que puedas trabajar con tus datos.

NumPy se basan en matrices y arreglos, lo que le permite aplicar funciones matemáticas a los arreglos. Pandas está construido sobre NumPy. Utiliza métodos de visualización de datos para comunicarse con otros y mostrar resultados de análisis significativos. Estas bibliotecas le permiten crear gráficos, cuadros, y mapas.

MapPlotLib es la biblioteca más conocida para visualización de datos. Son populares para hacer gráficos y diagramas, y los gráficos son fácilmente personalizables.

Seaborn, se basa en Matplotlib. Esta biblioteca genera mapas de calor, Serie temporal y diagramas de violín.

Scikit-learn contiene herramientas para el modelado estadístico, que incluyen regresión, clasificación, agrupamiento, etc. Está construido sobre NumPy, SciPy, y MapPlotLib, y es sencillo comenzar a usarlo; Usted define el modelo y especifica los tipos de parámetros que desea utilizar para crear modelos de aprendizaje profundo

Keras le permite crear el modelo de aprendizaje profundo estándar Al igual que Scikit-Learn, la interfaz de alto nivel permite construir modelos de manera rápida y sencilla, puede funcionar utilizando unidades de procesamiento gráfico o GPU, pero en muchos casos, es necesario un entorno de nivel inferior para el aprendizaje profundo.

TensorFlow es un marco de bajo nivel que se utiliza en la producción a gran escala de modelos de aprendizaje profundo. Está diseñado para producción e implementación, pero puede resultar difícil de manejar para la experimentación.

PyTorch se utiliza para la experimentación, lo que facilita a los investigadores probar ideas.

Apache Spark es un marco de computación en clúster de propósito general que le permite procesar datos mediante clústeres de cómputo. Los datos se procesan en paralelo en más de un ordenador simultáneamente.

La biblioteca Spark tiene una funcionalidad similar a Pandas, NumPy, y Scikit-learn. Y los trabajos de procesamiento de datos de Apache Spark pueden realizarse en Python, R, Scala y SQL.

Actualmente existen muchas bibliotecas de Scala, y Scala se utiliza predominantemente en ingeniería de datos y ciencia de datos. Analicemos algunas bibliotecas que son complementarias a Spark.

Vegas es una biblioteca de Scala para visualizaciones de datos estadísticos. Con Vegas, se puede trabajar con archivos de datos y también con marcos de datos de Spark. Y para el aprendizaje profundo, se puede utilizar BigDL.

R tiene funcionalidad incorporada para aprendizaje automático y visualización de datos, pero también hay bibliotecas complementarias, ggplot2 es una biblioteca popular para visualización de datos en R. También puede utilizar bibliotecas que le permitan interactuar con Keras y TensorFlow.

Interfaces de programación de aplicaciones (API)

Permite la comunicación entre dos piezas de software. Por ejemplo, en un programa, tienes algunos datos y otros componentes de software. Utiliza la API para comunicarse mediante entradas y salidas sin saber qué sucede en el back-end.

El backend es la parte de un sistema informático (como una aplicación web o móvil) que se encarga de la lógica interna, el procesamiento de datos y la comunicación con la base de datos o servicios externos. Es lo que ocurre "detrás de cámaras", es decir, lo que el usuario no ve directamente, pero que hace que todo funcione correctamente.

La API solo se refiere a la interfaz, es la parte de la biblioteca que ves mientras contiene todos los componentes del programa.

Para entender mejor cómo funciona una API en una biblioteca, considera un ejemplo de la biblioteca Pandas. Pandas es un conjunto de componentes de software WHERE no todos los componentes están escritos en Python. En su programa, hay algunos datos y un conjunto de componentes de software. Puede utilizar la API de Pandas para procesar los datos comunicándose con otros componentes de software.

El componente de software en el backend puede ser el mismo, pero puede haber una API para diferentes idiomas, considera TensorFlow en el backend, escrito en C++ que puede usar APIs para otros lenguajes, como Python, JavaScript, C++, Java y Go, y así, la API es solo la interfaz.

Otras APIs desarrolladas por voluntarios para TensorFlow son Julia, MATLAB, R, Scala, TEE y muchas más. Entonces, las API REST son otro tipo popular de API. RE significa Representacional, S significa Estado, y T significa Transferencia. Te

permiten comunicarte a través de internet y aprovechar recursos como almacenamiento, datos, algoritmos de inteligencia artificial.

En la API REST, su programa es el cliente. La API se comunica con un servicio web al que puedes llamar a través de Internet. Aunque hay reglas sobre la comunicación, la entrada o solicitud, y la salida o respuesta.

Las API REST obtienen toda la información de la solicitud enviada por el cliente. La solicitud se envía mediante un mensaje HTTP que contiene un archivo JSON. El archivo contiene instrucciones sobre qué operación debe realizar el servicio web. Esta operación se transmite al servicio web a través de internet, y el servicio realiza la operación, además, se utiliza IA para mejorar el rendimiento. De manera similar, el servicio web devuelve una respuesta a través de un mensaje HTTP donde la información se devuelve mediante un archivo JSON, y esta información se transmite de vuelta al cliente.

Otro ejemplo de una API REST es Watson Speech-to-Text API. Esta API convierte voz en texto. En la llamada API, enviará una copia del archivo de audio a la API. Esto se llama solicitud de publicación. Luego, la API enviará la transcripción de texto de lo que está diciendo el individuo. En el back-end, la API realiza una solicitud GET. Y finalmente, veamos nuestro ejemplo final, la API de Watson Language Translator. Envía el texto que deseas traducir a Watson Language Translator API. La API traducirá el texto y te enviará la traducción de vuelta. Y en este caso, la API traduce del inglés al español.

Conjuntos de datos

Un conjunto de datos es una colección estructurada de datos. Los datos incorporan información representada como texto, números, o medios como imágenes, archivos de audio o video.

Un conjunto de datos tabulares comprende una colección de filas que contienen columnas que almacenan la información. Un formato de datos tabulares popular son los valores separados por comas o CSV. Un archivo CSV es un archivo de texto delimitado donde cada línea representa una fila y una coma separa los valores de datos.

Las estructuras de datos jerárquicas o de red se utilizan normalmente para representar relaciones entre datos. Los datos jerárquicos se organizan en un formato de árbol, mientras que los datos de red se almacenan como un gráfico.

Un conjunto de datos también puede incluir archivos de datos sin procesar, como imágenes o audio.

Existen muchas fuentes de datos abiertos en Internet, y se puede encontrar una lista completa de portales de datos disponibles en todo el mundo en el sitio web datacatalogues.org de la Open Knowledge Foundation. Las Naciones Unidas, la

Unión Europea, y muchas otras organizaciones gubernamentales e intergubernamentales mantienen repositorios de datos que proporcionan acceso a una amplia gama de información. En Kaggle, una popular comunidad en línea de ciencia de datos, puedes encontrar y contribuir con conjuntos de datos que podrían ser de interés general.

Compartir datos empresariales - Intercambio de activos de datos

IBM creó Data Asset Exchange, o DAX. DAX proporciona una colección seleccionada de conjuntos de datos abiertos, tanto de IBM Research como de fuentes confiables de terceros. Estos conjuntos de datos están listos para usarse en aplicaciones empresariales con una amplia variedad de tipos de aplicaciones, incluidas imágenes, video, texto y audio. DAX tiene como objetivo fomentar el intercambio de datos y la colaboración manteniendo los conjuntos de datos disponibles bajo un Acuerdo de Licencia de Datos Comunitarios.

DAX facilita que los desarrolladores comiencen a trabajar con conjuntos de datos porque proporciona un único lugar para acceder a conjuntos de datos únicos y de alta calidad de fuentes confiables como IBM Research. También proporciona cuadernos tutoriales que explican los conceptos básicos de limpieza de datos, preprocesamiento, y análisis exploratorio. Ciertos conjuntos de datos incluyen Notebooks avanzados que explican cómo realizar tareas más complejas como crear gráficos, entrenar modelos de Aprendizaje automático, integrar Aprendizaje profundo a través del intercambio de activos de modelos, y realizar análisis estadísticos y de Serie temporal. Tanto el intercambio de activos de datos como el intercambio de activos de modelos están disponibles en el sitio web de IBM Developer.

Con los Notebooks en Watson Studio, puedes acceder a IBM Cloud, crear un proyecto, y cargar Notebooks.

Modelos de aprendizaje automático - Aprender de los modelos para hacer predicciones

Los datos contienen mucha información que puede utilizarse para resolver ciertos tipos de problemas. Los enfoques tradicionales de análisis de datos pueden ser una persona que inspecciona manualmente los datos o un programa informático especializado que automatiza el análisis humano.

El aprendizaje automático, o ML, utiliza algoritmos, también conocidos como modelos, para identificar patrones en los datos. El proceso por el que el modelo aprende patrones de datos se llama entrenamiento. Un modelo entrenado puede hacer predicciones en IA y CTI; ciencia, tecnología e innovación.

Cuando se presentan nuevos datos al modelo, éste intenta hacer predicciones o tomar decisiones basadas en los patrones que ha aprendido de datos anteriores.

Los modelos de aprendizaje automático se pueden dividir en tres clases básicas: aprendizaje supervisado, aprendizaje no supervisado, y aprendizaje de refuerzo.

El tipo de aprendizaje automático más comúnmente utilizado es el aprendizaje supervisado. En el aprendizaje supervisado, un humano proporciona datos de entrada y resultados correctos. El modelo intenta identificar relaciones y dependencias entre los datos de entrada y la salida correcta. Este tipo de aprendizaje comprende dos tipos de modelos: regresión y clasificación.

Los modelos de regresión se utilizan para predecir un valor numérico o real.

Los modelos de clasificación se utilizan para predecir si cierta información o datos pertenecen a una categoría o clase.

En el aprendizaje no supervisado, los datos no son etiquetados por un humano. Los modelos deben analizar los datos y tratar de identificar patrones y estructuras dentro de los datos basándose en sus características. El agrupamiento es un ejemplo de este estilo de aprendizaje. Los modelos de agrupamiento se utilizan para dividir cada registro de un conjunto de datos en uno de un grupo similar.

El aprendizaje por refuerzo se inspira en la forma en que los seres humanos y otros organismos aprenden a través de la experiencia. En este enfoque, un modelo aprende cuál es la mejor secuencia de acciones que debe tomar, dadas ciertas condiciones del entorno, con el objetivo de maximizar una recompensa acumulada a lo largo del tiempo. Este tipo de aprendizaje ha demostrado un éxito notable en tareas complejas, como superar a los mejores jugadores humanos en juegos como Go, ajedrez y videojuegos de estrategia populares.

El aprendizaje profundo es un tipo especializado de aprendizaje automático. Se refiere a un conjunto general de modelos y técnicas que emulan vagamente la forma en que el cerebro humano resuelve una amplia gama de problemas. Se utiliza comúnmente para analizar lenguaje natural, tanto hablado como texto, imágenes, audio, y vídeo, para pronosticar datos de series de tiempo, y mucho más. El aprendizaje profundo ha tenido recientemente mucho éxito en estas y otras áreas y, por lo tanto, se está convirtiendo en una herramienta cada vez más popular e importante para la ciencia de datos.

Se requieren grandes conjuntos de datos etiquetados para entrenar un modelo, requiere un uso intensivo de recursos informáticos y, por lo general, requiere hardware especial para lograr tiempos de entrenamiento aceptables.

Hoy en día, es posible construir un modelo de inteligencia artificial (IA) basado en aprendizaje profundo desde cero o utilizar modelos preentrenados disponibles en repositorios públicos. Estos modelos se implementan mediante marcos de desarrollo populares como TensorFlow, PyTorch, Keras y ONNX, los cuales ofrecen interfaces (APIs) en lenguajes como Python, C++ y JavaScript. Además, existen entornos de desarrollo integrados (IDE) que facilitan el trabajo tanto en entornos locales (LAN) como en la nube (MIN). Los modelos preentrenados pueden descargarse desde repositorios conocidos como “zoológicos de modelos” (model zoos), ofrecidos por plataformas como TensorFlow Hub, PyTorch Hub y

ONNX Model Zoo. Estos recursos también incluyen modelos desarrollados por grupos de investigación académicos y empresas del sector tecnológico.

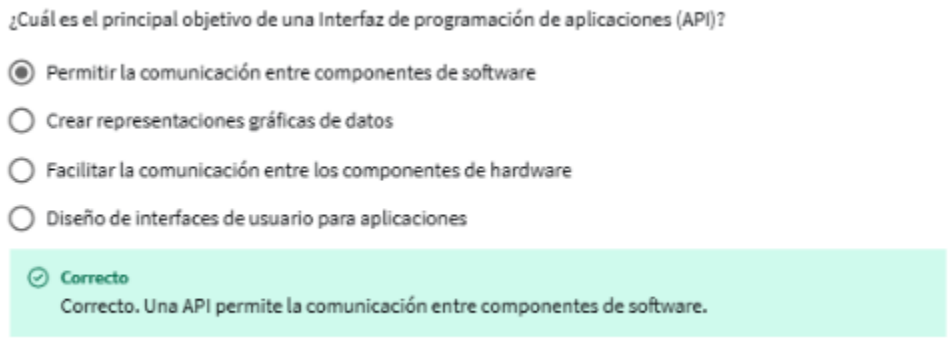
Para ilustrar el proceso, consideremos el caso de una aplicación que debe identificar objetos en imágenes. El primer paso consiste en recolectar y preparar los datos, lo cual incluye el etiquetado de imágenes con información relevante, como cuadros delimitadores alrededor de los objetos. Luego, se selecciona un modelo adecuado o se construye uno desde cero, y se entrena utilizando los datos preparados. Durante el entrenamiento, el modelo aprende a reconocer patrones y a identificar objetos con base en las etiquetas suministradas. Este proceso se repite y ajusta hasta alcanzar un rendimiento satisfactorio. Finalmente, el modelo entrenado se implementa e integra en la aplicación para su uso real.

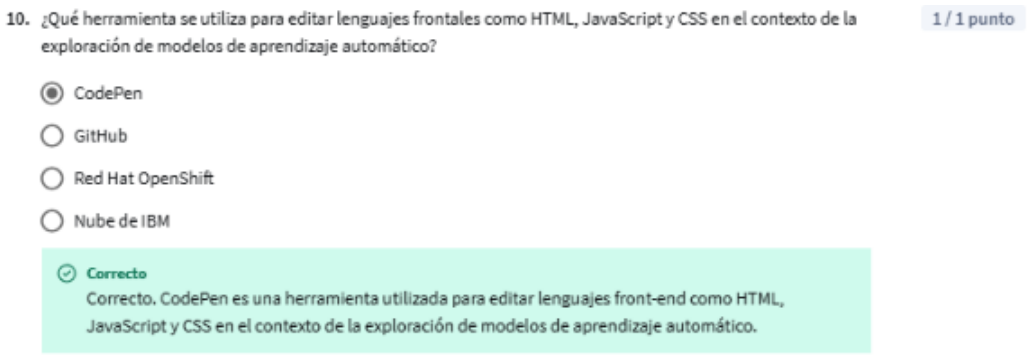
El Intercambio de Activos Modelo

Model Asset Exchange (MAX), disponible en la plataforma IBM Developer, es un repositorio gratuito y de código abierto que proporciona modelos de aprendizaje profundo listos para usar y personalizables. Entrenar un modelo desde cero requiere una gran cantidad de datos, tiempo, trabajo y recursos computacionales, lo cual puede extender significativamente el tiempo necesario para obtener valor (VALUE) del modelo. Para reducir este tiempo, es recomendable aprovechar modelos previamente entrenados, los cuales pueden ser utilizados directamente o requerir menos tiempo de ajuste. Estos modelos han sido desarrollados mediante un proceso riguroso que incluye etapas de investigación, evaluación, prueba, entrenamiento y validación. MAX ofrece microservicios diseñados para resolver problemas comerciales comunes, utilizando modelos previamente entrenados que pueden implementarse rápidamente tanto en entornos locales como en la nube. Todos los modelos de MAX están disponibles bajo licencias permisivas de código abierto, lo que facilita su uso en proyectos personales y comerciales con bajo riesgo legal.

Los modelos en MAX cubren una amplia variedad de dominios, como detección de objetos, clasificación de imágenes, procesamiento de audio, video y texto, reconocimiento de entidades con nombre, traducción de imágenes a texto y detección de poses humanas, entre otros. Cada microservicio de implementación incluye: un modelo de aprendizaje profundo previamente entrenado, código para el preprocesamiento de datos de entrada, código de posprocesamiento de las salidas del modelo, y una API pública estandarizada que permite su integración con aplicaciones. Estos microservicios se implementan como imágenes Docker de código abierto, lo que facilita su despliegue en máquinas locales o en entornos de nube pública, privada o híbrida. Las imágenes Docker se publican en GitHub y pueden personalizarse según las necesidades del usuario. Además, MAX aprovecha herramientas como Kubernetes para automatizar la implementación,

escalado y gestión de estos contenedores, siendo Red Hat OpenShift una de las plataformas Kubernetes empresariales más utilizadas, disponible en servicios como IBM Cloud, Google Cloud Platform, Amazon Web Services y Microsoft Azure.





Cuadernos Jupyter

Los Jupyter Notebooks son aplicaciones basadas en navegador que permiten crear y compartir documentos que integran código, ecuaciones, visualizaciones, texto narrativo, enlaces y otros elementos interactivos. Surgieron originalmente del proyecto IPython, enfocado en Python, pero con el tiempo se ampliaron para soportar otros lenguajes como Julia y R, dando origen al nombre “Jupyter” (JUlia, PYThon, R). Actualmente, son herramientas fundamentales en ciencia de datos, ya que permiten documentar experimentos y resultados de manera reproducible. Un notebook puede mostrar la salida del código —como gráficos, tablas o texto— directamente dentro del mismo archivo, y puede exportarse a formatos como PDF o HTML. La plataforma JupyterLab amplía las funcionalidades de los notebooks, permitiendo trabajar con múltiples archivos, consolas, terminales y componentes personalizados de forma integrada. JupyterLab admite numerosos formatos de archivo, incluyendo CSV, JSON, PDF, Vega y más, y es de código abierto.

Los cuadernos Jupyter pueden utilizarse tanto en la nube (por ejemplo, mediante servicios como IBM Skills Network Labs o Google Colab) como en entornos locales. Las plataformas en la nube no requieren instalación y ofrecen un entorno Jupyter completamente funcional, con soporte para múltiples lenguajes. Para uso local, Jupyter puede instalarse mediante pip install o descargarse como parte de la distribución Anaconda, una suite popular para ciencia de datos. Los notebooks funcionan a través de kernels, que son motores de ejecución responsables de procesar el código. Al abrir un cuaderno, el kernel correspondiente (como Python, R, Julia, Swift, entre otros) se inicia automáticamente y ejecuta las celdas de código. En entornos como Skills Network, varios kernels vienen preinstalados, facilitando el trabajo con diferentes lenguajes. En instalaciones locales, es posible añadir nuevos kernels desde la línea de comandos según las necesidades del proyecto.

Arquitectura de Jupyter

La arquitectura de Jupyter se basa en un modelo de dos procesos: un *cliente* y un *núcleo* (kernel). El cliente, generalmente una interfaz basada en navegador permite al usuario escribir y enviar código al núcleo para su ejecución. Por su parte, el núcleo ejecuta ese código y devuelve los resultados al cliente para su visualización, permitiendo una interacción fluida y dinámica. En el caso del Notebook de R, el navegador actúa como interfaz para interactuar con el núcleo de R. Todo el contenido de un notebook —incluyendo código, metadatos, resultados y visualizaciones— se guarda en un archivo con formato JSON y extensión. ipynb. Este archivo es administrado por el servidor de cuadernos (notebook server), que se encarga de guardar, cargar y gestionar los notebooks, mientras el núcleo ejecuta las celdas de código cuando el usuario lo solicita.

Jupyter también incorpora herramientas para exportar notebooks a otros formatos mediante nbconvert. Este proceso se compone de tres etapas: preprocesamiento, exportación y postprocesamiento. Por ejemplo, al convertir un notebook a HTML, primero el contenido es procesado por un preprocesador que lo adapta, luego un exportador convierte el archivo al nuevo formato, y finalmente un postprocesador aplica modificaciones finales para obtener el archivo HTML listo para ser visualizado en un navegador. Este flujo de trabajo hace posible compartir notebooks como páginas web, documentos PDF, presentaciones o scripts de código, aumentando su portabilidad y utilidad en entornos académicos y profesionales.

Entornos Anaconda Jupyter adicionales

Los cuadernos computacionales, como los Notebooks de Jupyter, integran código, resultados computacionales, texto explicativo y elementos multimedia en un único

documento, lo que los hace ideales para la ciencia de datos y el aprendizaje automático. Estos notebooks admiten múltiples lenguajes de programación —como Python y R— y son compatibles con entornos como JupyterLab y Visual Studio Code (VS Code). JupyterLab es una aplicación web de código abierto que extiende las capacidades de los Notebooks de Jupyter, ofreciendo una experiencia más flexible e integrada. Incluye bibliotecas clave de Python preinstaladas mediante Anaconda, como NumPy, Pandas y Matplotlib. Además, permite la creación de celdas de código y celdas Markdown, lo que facilita tanto la ejecución de scripts como la documentación de procesos de forma enriquecida.

Anaconda es una distribución gratuita y de código abierto para Python y R, muy popular en proyectos de ciencia de datos. Incluye más de 1.500 bibliotecas, así como herramientas gráficas como Anaconda Navigator, que permite gestionar entornos y paquetes sin necesidad de utilizar la línea de comandos. A través de Anaconda Navigator se pueden lanzar JupyterLab y VS Code fácilmente. VS Code, por su parte, es un editor de código multiplataforma que soporta múltiples lenguajes y funciones como depuración, autocompletado y ejecución de notebooks. En ambos entornos es posible crear y ejecutar Notebooks de Jupyter y R, seleccionar kernels, personalizar celdas y exportar los archivos resultantes en múltiples formatos. Esta flexibilidad facilita la documentación, colaboración y reproducibilidad de los experimentos en ciencia de datos.

Entornos Jupyter adicionales basados en la nube

Los cuadernos computacionales, como los Notebooks de Jupyter, combinan código, resultados computacionales, texto explicativo y elementos multimedia en un solo documento, lo que facilita su uso en proyectos de ciencia de datos. Entre los entornos basados en la nube más utilizados para trabajar con Notebooks destacan JupyterLite y Google Colaboratory (Google Colab). JupyterLite es una versión liviana de JupyterLab que se ejecuta directamente en el navegador, sin necesidad de un servidor dedicado. Esto permite desplegarlo como un sitio web estático, ideal para entornos educativos o demostrativos. Incluye bibliotecas populares como Altair, Plotly e ipywidgets para visualizaciones interactivas, y funciona con kernels como Python Pyodide y Pyolite, los cuales permiten ejecutar código Python completamente en el navegador.

Por otro lado, Google Colab es un entorno Jupyter gratuito alojado en la nube que se integra con Google Drive y GitHub. Los notebooks creados en Colab pueden compartirse fácilmente, ejecutarse sin configuración previa y aprovechar bibliotecas preinstaladas como Scikit-learn y Matplotlib. Esta plataforma permite a los usuarios subir, clonar o crear notebooks desde cero, y ejecutar código en celdas directamente desde el navegador. Google Colab es especialmente útil para desarrollar y probar proyectos de ciencia de datos de manera rápida y colaborativa. En conjunto, herramientas como JupyterLite y Google Colab amplían

las opciones para trabajar con Jupyter Notebooks, ofreciendo soluciones ligeras y accesibles sin necesidad de instalaciones locales.

Introducción a R y RStudio

Es una herramienta poderosa para el procesamiento y manipulación de datos, inferencia estadística, análisis de datos y algoritmos de aprendizaje automático.

R es utilizado principalmente por académicos, profesionales de la salud, y el gobierno. R admite la importación de datos de diferentes fuentes como archivos planos, bases de datos, web, y software estadístico como SPSS y Stata.

R es un lenguaje preferido por algunos científicos de datos porque las funciones de R son fáciles de usar. También es conocido por producir excelentes visualizaciones y contiene paquetes para manejar el análisis de datos sin la necesidad de instalar bibliotecas adicionales. Un entorno de desarrollo integrado popular para desarrollar y ejecutar el código fuente y los programas del lenguaje R es RStudio. Mejora y aumenta la productividad con el lenguaje R.

RStudio es un entorno de desarrollo integrado (IDE) ampliamente utilizado para trabajar con el lenguaje de programación R, especialmente en proyectos de ciencia de datos y estadística. Su interfaz está organizada en múltiples pestañas y paneles que permiten al usuario escribir, ejecutar y rastrear su código de forma eficiente.

Entre sus componentes principales se encuentran:

- Editor de código: Resalta la sintaxis y permite ejecutar directamente fragmentos de código.
- Consola: Permite la entrada y ejecución directa de comandos en R.
- Espacio de trabajo e historial: Muestra los objetos creados en la sesión actual y registra todos los comandos previamente ejecutados.
- Pestaña Archivos: Muestra el contenido del directorio de trabajo actual.
- Pestaña Gráficos: Almacena un historial visual de los gráficos generados y ofrece opciones para exportarlos en formatos como PDF o imagen.
- Pestaña Paquetes: Presenta una lista de paquetes R instalados y permite gestionar su activación o instalación.
- Pestaña Ayuda: Ofrece acceso a documentación sobre funciones, paquetes, recursos del lenguaje R y asistencia de RStudio.

Si se utiliza R como herramienta principal de análisis, existen varias bibliotecas muy populares dentro de la comunidad científica de datos, como:

- dplyr para manipulación de datos,

- stringr para manejo de cadenas de texto,
- ggplot2 para visualización de datos.
- caret para tareas de aprendizaje automático.

Además, para facilitar el aprendizaje, existen entornos virtuales de RStudio disponibles en plataformas educativas como laboratorios en línea, permitiendo a los estudiantes practicar sin necesidad de instalar software adicional o configurar un entorno local.

Trazado en RStudio

Con la creciente disponibilidad de datos, una de las tareas más importantes de un científico de datos es generar información útil mediante visualizaciones. R ofrece una variedad de paquetes de visualización que se pueden utilizar según las necesidades del análisis. Para instalar estos paquetes en tu entorno de R, se utiliza el comando install.packages("nombre_del_paquete").

Algunos paquetes comunes para visualización en R incluyen ggplot2, Plotly, Lattice y Leaflet. El paquete ggplot2 se utiliza ampliamente para crear visualizaciones como histogramas, gráficos de barras y diagramas de dispersión. Una de sus fortalezas es la posibilidad de agregar capas y componentes a una sola visualización. Plotly, por otro lado, es ideal para generar visualizaciones interactivas basadas en la web que pueden visualizarse directamente en el navegador o guardarse como archivos HTML. Lattice está diseñado para trabajar con conjuntos de datos multivariables complejos y permite generar gráficos de forma eficiente sin necesidad de demasiadas personalizaciones. Por último, Leaflet se utiliza para construir mapas interactivos.

R también cuenta con funciones integradas para crear gráficos simples. Por ejemplo, se puede utilizar la función plot() para generar un diagrama de dispersión que representa los valores frente a su índice. A este gráfico se le pueden agregar elementos como líneas y títulos para hacerlo más comprensible. Para añadir una línea, se especifica el tipo mediante el argumento type, y para el título se utiliza la función title().

La biblioteca ggplot2 permite crear visualizaciones más elaboradas al permitir combinar diversas capas. Por ejemplo, para crear un gráfico de dispersión con el conjunto de datos integrado mtcars, primero se carga la biblioteca con library(ggplot2). Luego, se llama a la función ggplot(), especificando mpg como eje X y wt como eje Y, y se agrega la función geom_point() para representar los puntos del gráfico. Sin esta función, el gráfico aparecería vacío.

Además, puedes mejorar el gráfico añadiendo un título principal con ggtitle() y modificando los nombres de los ejes mediante el argumento labs(). Esto permite crear visualizaciones más claras y significativas para el análisis.

Finalmente, dentro de un laboratorio práctico, también puedes utilizar la extensión ggally, que amplía las capacidades de ggplot2. Esta biblioteca facilita la combinación de objetos geométricos con datos transformados, simplificando la creación de visualizaciones complejas.

Visión general de Git/GitHub

No puedes hablar de Git y GitHub sin tener un entendimiento básico de lo que es el Control de versión. Un sistema de control de versiones le permite realizar un seguimiento de los cambios en sus documentos. Esto le permite recuperar fácilmente versiones anteriores de su documento si comete un error y hace que la colaboración con otros sea mucho más sencilla.

Git es un software gratuito y de código abierto distribuido bajo la nueva licencia pública general.

Git es un sistema de control de versiones distribuido, lo que significa que los usuarios en cualquier parte del mundo pueden tener una copia de su proyecto en su propia computadora. Cuando hayan realizado cambios, podrán sincronizar su versión con un servidor remoto para compartirla contigo Git no es el único sistema de Control de versión que existe, pero el aspecto distribuido es una de las principales razones por las que se ha convertido en uno de los sistemas de Control de versión más comunes disponibles.

Los sistemas de control de versiones permiten Control de versión de imágenes, documentos y varios tipos de archivos.

Puedes usar Git sin una interfaz web usando tu interfaz de línea de comandos, pero GitHub es uno de los servicios alojados en la web más populares para repositorios de Git. Otros incluyen GitLab, Bitbucket, y Beanstalk.

Antes de comenzar a trabajar con Git y GitHub, es importante conocer algunos términos esenciales:

- SSH (Secure Shell): Un protocolo para iniciar sesión de forma remota de manera segura entre computadoras.
- Repositorio: Es el espacio donde se almacenan los archivos de tu proyecto y donde Git realiza el control de versiones.
- Fork (tenedor): Es una copia de un repositorio existente, útil para proponer cambios sin afectar el original.
- Pull Request (solicitud de extracción): Es la forma en que solicitas que alguien revise y apruebe tus cambios antes de integrarlos al proyecto principal.
- Directorio de trabajo: Es la carpeta local donde se encuentran los archivos de tu proyecto asociados con un repositorio Git.

Comandos básicos de Git

Hay una serie de comandos de Git que se utilizan con frecuencia:

- git init: Inicializa un nuevo repositorio en el directorio local.
- git clone: Clona un repositorio existente desde un servidor remoto (como GitHub) a tu máquina local.
- git add: Mueve los archivos modificados al área de preparación (staging area).
- git status: Muestra el estado actual de los archivos en el directorio de trabajo y la zona de preparación.
- git commit: Guarda una instantánea de los archivos preparados en el historial del proyecto.
- git reset: Revierte cambios realizados en los archivos del directorio de trabajo.
- git log: Muestra el historial de commits (cambios) del proyecto.

Trabajo con ramas

- git branch: Crea ramas, que son entornos aislados para trabajar en nuevas características o correcciones.
- git checkout: Permite cambiar de rama dentro del repositorio.
- git merge: Fusiona ramas para integrar los cambios realizados en diferentes líneas de desarrollo.

Introducción a GitHub

A principios de los años 2000, el desarrollo del núcleo de Linux se gestionaba utilizando un sistema de control de versiones gratuito llamado BitKeeper. Sin embargo, en 2005 BitKeeper dejó de ser gratuito, lo cual representó un problema para la comunidad de desarrolladores de Linux. Como respuesta**,** Linus Torvalds**,** creador de Linux, lideró el desarrollo de un nuevo sistema de control de versiones: Git**.**

Git fue creado rápidamente, con las principales características definidas por un pequeño grupo de desarrolladores. Entre estas se incluyen: soporte robusto para el desarrollo no lineal (algo esencial dado que los parches de Linux llegaban a razón de 6,7 por segundo), desarrollo distribuido (cada programador puede tener una copia local del historial completo del proyecto), compatibilidad con sistemas y protocolos existentes, manejo eficiente de proyectos de gran tamaño, autenticación criptográfica del historial (para garantizar que todas las copias distribuidas estén sincronizadas), y estrategias de fusión personalizables (necesarias para flujos de trabajo complejos).

Git es un sistema de control de versiones distribuido orientado al seguimiento de cambios en el código fuente durante el desarrollo de software. Está diseñado para permitir la colaboración eficiente entre múltiples desarrolladores y para adaptarse bien a metodologías ágiles de desarrollo. A diferencia de los sistemas centralizados, donde los desarrolladores deben conectarse a un servidor central para obtener y subir cambios, Git permite que cada desarrollador trabaje desde una copia local completa del repositorio. Esto mejora la velocidad, la autonomía y la resiliencia del desarrollo colaborativo.

En un proyecto gestionado con Git, normalmente existe una rama principal (como main o master) que representa el estado listo para producción del código. Los equipos pueden trabajar en distintas ramas de manera simultánea, integrando los cambios a la rama principal una vez que estén validados. Además, Git ofrece un control centralizado de permisos y tareas, ideal para proyectos más grandes o en organizaciones estructuradas.

GitHub es una de las plataformas más populares para alojar repositorios Git en la web. Aunque se puede usar Git desde la línea de comandos o con clientes de escritorio (como el GitHub Desktop), GitHub proporciona una interfaz web muy útil para gestionar repositorios, revisar código, colaborar con otros desarrolladores, y hacer seguimiento de tareas.

GitHub es propiedad de Microsoft y ofrece cuentas gratuitas tanto para individuos como para organizaciones. A agosto de 2019, GitHub ya albergaba más de 100 millones de repositorios. En esta plataforma, un repositorio (o “repo”) es una estructura de datos donde se almacena el código fuente, junto con todo el historial de cambios y versiones del proyecto.

Además de GitHub, existen otras plataformas como GitLab, que ofrece una solución completa de desarrollo en una sola aplicación. GitLab proporciona repositorios Git junto con herramientas de colaboración, revisión de código, integración continua (CI) y entrega continua (CD). Estas herramientas permiten a los equipos trabajar de manera más fluida, desde la codificación hasta el despliegue de software, manteniendo siempre el control de versiones y la trazabilidad de los cambios.

Repositorios de GitHub

El repositorio es el núcleo de cualquier proyecto basado en Git. Contiene todo el código y archivos relacionados, como el archivo README (que describe el propósito del proyecto), archivos de licencia, y otros artefactos importantes. Al crear un repositorio, puedes elegir que sea público (visible para todos) o privado (accesible solo para usuarios autorizados).

Una vez creado, el repositorio tiene varias pestañas visibles en la interfaz de GitHub:

- Código: aquí se almacenan todos los archivos del repositorio. Inicialmente, solo puede contener un README o un archivo de licencia si así se indicó al momento de la creación.
- Issues: permite hacer seguimiento de tareas, errores o mejoras. Es una herramienta clave para la planificación colaborativa.
- Pull requests (Solicitudes de extracción): permiten proponer cambios al proyecto. Estas solicitudes son revisadas antes de que los cambios sean integrados en la rama principal.
- Projects: proporciona herramientas de gestión como tableros Kanban para organizar tareas.
- Wiki, Security y Insights (Análisis): ofrecen documentación colaborativa, opciones de seguridad y métricas de uso.
- Settings: permite personalizar el repositorio, incluyendo su nombre, visibilidad y permisos de acceso.

Trabajo con ramas en GitHub

Una rama en Git es una copia del estado actual del repositorio donde se pueden realizar cambios sin afectar directamente la rama principal (main o master). Esto permite desarrollar nuevas funciones, corregir errores o experimentar sin alterar el código estable.

Crear una nueva rama

1. Desde la interfaz del repositorio, haz clic en el menú desplegable de ramas.
2. Crea una nueva rama (por ejemplo, rama-secundaria) a partir de la rama principal.
3. Esta nueva rama contendrá una copia exacta del contenido de la rama principal en ese momento.

Agregar archivos en una rama

- Asegúrate de tener seleccionada la rama-secundaria.

- Haz clic en “Crear nuevo archivo”.
- Nombra el archivo, por ejemplo testchild.py, y añade contenido (como un print()).
- En la parte inferior, añade un mensaje de confirmación (commit) como createtestchild.py y haz clic en Confirmar nuevo archivo.
- Este archivo ahora forma parte de la rama-secundaria, pero no está en la rama principal.

Solicitudes de extracción (Pull Requests)

Una vez que el archivo fue probado y estás satisfecho con los cambios:

1. Haz clic en Comparar y Solicitud de extracción.
2. GitHub mostrará las diferencias entre la rama secundaria y la principal (por ejemplo, qué archivos se modificaron o agregaron).
3. Añade un título y comentario descriptivo para la solicitud.
4. Haz clic en Crear solicitud de extracción.
5. Otro miembro del equipo puede revisar la solicitud y, si todo está en orden, hacer clic en Fusionar solicitud de extracción y luego en Confirmar.

Después de la fusión:

- Puedes eliminar la rama secundaria si ya no se necesita.
- El archivo testchild.py ahora aparece también en la rama principal.