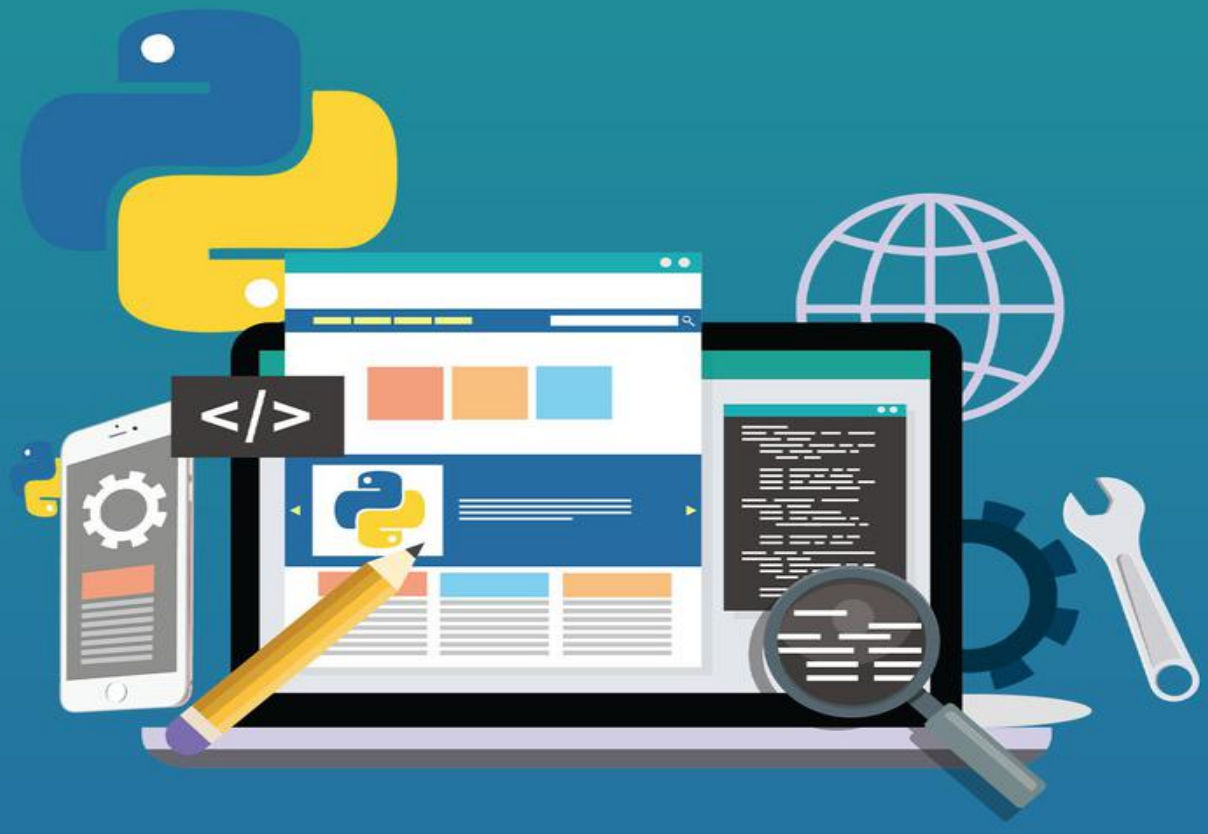
PYTHON

Curso Intensivo Paso a Paso Sobre Cómo Elaborar Fácilmente su Primer Proyecto de Ciencia de Datos Desde Cero en Menos de 7 Días. Incluye Ejercicios Prácticos



EMILIANO EMIGDIO

Tabla de Contenido

Python

Curso Intensivo Paso a Paso Sobre Cómo Elaborar Fácilmente su Primer Proyecto de Ciencia de Datos Desde Cero en Menos de 7 Días. Incluye Ejercicios Prácticos

EMILIANO EMIGDIO

Índice de contenidos

Introducción

Cuando la ciencia de los datos surgió por primera vez, sólo estaba en manos de los científicos y de algunos contables atrevidos. Esto es muy comprensible porque, sin llegar a vislumbrar realmente en qué consiste la ciencia de los datos, es fácil asumir que es para los fuertes de corazón y los que disfrutan resolviendo problemas "aburridos". Pero estamos en 2020, y el furor por la ciencia de datos está en todas partes. Esto sólo puede significar una cosa: que es muy importante para el mundo en el que vivimos actualmente. Cada vez se demandan más científicos de datos; cada día se construyen más tecnologías para ayudar al concepto de ciencia de datos. Pero, ¿qué es exactamente lo que hace que la ciencia de datos sea tan importante para la persona o la organización del siglo XXI? Sencillo: los datos.

Las personas, las organizaciones y los países necesitan datos, independientemente del nivel en que se encuentren. Las estadísticas son necesarias para medir el desarrollo, hacer informes de progreso y muchas otras cosas, dondequiera que nos dirijamos. Esto nos lleva a preguntarnos: ¿qué es la ciencia de los datos? Aunque no tenga una definición concreta que sea generalmente aceptable, porque se ha convertido en un fenómeno global, es una materia interdisciplinar que comprende tres áreas distintas pero que se solapan. Estas áreas son la estadística, la informática y los conocimientos especializados.

Un científico de datos será entonces un experto que puede modelar y resumir conjuntos de datos, diseñar y utilizar algoritmos para almacenar, procesar y visualizar eficazmente los datos obtenidos, al tiempo que es capaz de formular las preguntas adecuadas y poner las respuestas en contexto. La realidad, sin embargo, es que los mejores científicos de datos de hoy en día trabajan en equipo. Debido a la variedad de habilidades que se necesitan en este campo, es raro encontrar a quienes han perfeccionado todos los conjuntos de habilidades. Así que, si quieres aprender ciencia de datos, no pasa nada por no tener todavía todas las habilidades necesarias. Un resumen perfecto de la cartera de un científico de datos será: captura de datos, análisis de datos y presentación de datos. La ciencia de los datos implica el uso de técnicas matemáticas más avanzadas, estadísticas y big data.

Los científicos de datos no son sólo las personas "aburridas" que se limitan a sentarse delante de sus ordenadores para hacer números; son los que pueden responder a preguntas con más preguntas aún, ayudarnos a tomar mejores decisiones con la información que tenemos, crear sugerencias de opciones basadas en las elecciones anteriores, hacer que los robots vean objetos, y un montón de cosas más. De hecho, la ciencia de los datos se encuentra literalmente en cualquier concepto, ideología o industria que exista hoy en día (lo que sea), de modo que casi no podemos mirar a ninguna parte sin sentir sus efectos. La ciencia de los datos ayuda a compartir las experiencias desconcertantes que obtenemos de la tecnología hoy en día. Cuando decimos que la ciencia de los datos es lo que nos ayuda a entender y aceptar lo que consideramos nuestra realidad hoy en día, no es más que la verdad.

Aunque el concepto de Ciencia de Datos (el proceso de cuantificación y comprensión de las estadísticas) es relativamente nuevo, los principios y las matemáticas que lo sustentan siempre han existido. Así que sería estupendo abordar la ciencia de los datos no como un dominio de conocimiento totalmente nuevo, sino como un camino a través del cual puedes aplicar los conocimientos que ya tienes.

Puede que no lo sepas, pero de un modo u otro, has aplicado la ciencia de datos a una o dos de tus actividades diarias. Por ejemplo, cuando utilizas los motores de búsqueda para buscar algo. En algún momento, te hará sugerencias sobre algunas alternativas. Esos términos alternativos se obtienen gracias a la ciencia de datos. Cuando un médico hace un pronóstico, una forma de saber que tu bulto no es canceroso es a través de la ciencia de datos.

Este libro, en primer lugar, no sólo pretende ofrecer una introducción sencilla y útil a la ciencia de los datos, sino también mostrarle la importancia que tiene la ciencia de los datos en nuestra vida cotidiana. No sólo sabrás cómo responder a las preguntas planteadas, sino que también estarás seguro de dónde se pueden emplear. Así, sea cual sea el campo en el que te encuentres, ya sea para predecir el rendimiento de las acciones, optimizar los clics de los anuncios online, informar de los resultados de las elecciones o cualquier campo en el que se necesiten datos (que está en todas partes), podrás destacar con un mejor conocimiento y saber hacer de la ciencia de los datos.

Este libro está hecho para ser una herramienta que, en primer lugar, armonice la ciencia de los datos con Python. Busca conectar los puntos entre estos dos conceptos informáticos interrelacionados y hacerlos uno. Te destacará un millón y más de razones por las que aprender ciencia de datos con Python es una de las mejores maneras de hacerlo y por qué deberías aprovechar lo que aporta.

Python se implementó por primera vez en 1989 y se considera un lenguaje de programación muy fácil de usar y de aprender para codificadores de nivel básico y aficionados. Se trata de un lenguaje de programación de alto nivel, utilizado habitualmente para fines generales. Fue desarrollado originalmente por Guido van Rossum en el "Center Wiskunde & Informatica (CWI), Países Bajos", en la década de 1980 y presentado por la "Python Software Foundation" en 1991. Se considera ideal para las personas que se inician en la programación o en la codificación y necesitan comprender los fundamentos de la programación. Esto se debe al hecho de que Python se lee casi igual que el inglés. Por lo tanto, se necesita menos tiempo para entender cómo funciona el lenguaje y se puede centrar en el aprendizaje de los fundamentos de la programación.

Python es un lenguaje interpretado que admite la gestión automática de la memoria y la programación orientada a objetos. Este lenguaje de programación, muy intuitivo y flexible, puede utilizarse para codificar programas como algoritmos de aprendizaje automático, aplicaciones web, minería y visualización de datos o desarrollo de juegos.

Capítulo 1: Fundamentos de Python para la ciencia de datos

¿Qué es la ciencia de los datos?

La ciencia de los datos es una reunión de diferentes instrumentos, interfaces de datos y cálculos con estándares de IA (algoritmos) para encontrar patrones ocultos a partir de datos brutos. Estos datos se guardan en grandes almacenes de distribución de datos empresariales y se utilizan en enfoques inventivos para crear valor empresarial.

Un examinador de datos (analista) y un científico de datos son únicos. Un analista intenta procesar el historial de datos y aclarar lo que está sucediendo. En cambio, un investigador de datos necesita diferentes cálculos propulsados de IA (algoritmos de aprendizaje automático) para un evento de una ocasión específica utilizando el análisis.

Python y su historia

Python es un lenguaje de programación traducido de alta calidad y útil en todo el mundo. Desarrollado por Guido van Rossum y publicado por primera vez en 1991, la Fundación Python hace hincapié en la claridad del código aprovechando al máximo el espacio crítico. Su lenguaje está desarrollado y diseñado con metodología de objetos para permitir a los ingenieros de software componer código claro y lógico para proyectos de pequeña y gran escala.

Python se desarrolló por primera vez a finales de la década de 1980 como sucesor del lenguaje ABC. Python 2.0, descargado en 2000, presentaba novedades, como los conceptos de degradación y un marco de recolección de basura, adecuado para recoger los ciclos de referencia. Python 3.0, descargado en 2008, supuso una notable modificación del lenguaje, y gran parte del código de Python 2 no se ejecuta sin modificar en Python 3. El diseñador del lenguaje, Guido van Rossum, fue el único encargado de comprometerse hasta julio de 2018, pero ahora comparte la gestión como una persona más en una junta de cinco miembros.

Características únicas y filosofía

Python es un lenguaje de programación versátil que admite la programación orientada a objetos (POO) y otros lenguajes de programas informáticos prácticos. Inicialmente, no fue diseñado para la ciencia de los datos, pero como campo, los profesionales comenzaron a utilizarlo para el análisis de datos, y se convirtió en una prioridad para la ciencia de los datos. Muchos estándares diferentes se refuerzan utilizando expansiones, incluyendo un plan por contrato y la programación racional. Asimismo, incluye objetivos de nombres dinámicos (autoría tardía), que vinculan los nombres de técnicas y variables durante las operaciones del sistema. La biblioteca estándar tiene dos módulos que actualizan dispositivos útiles adquiridos de Haskell y ML estándar.

A diferencia de incorporar la mayor parte de su utilidad en su núcleo, Python fue concebido para ser profundamente escalable.

Python avanza hacia una puntuación y una estructura menos complejas y menos mezcladas, al tiempo que permite a los ingenieros tomar decisiones sobre su enfoque de la codificación. Al contrario de lo que se dice, "en Perl hay más de un enfoque", Python entiende que "debe haber uno, e idealmente un enfoque claro para hacerlo". Alex Martelli, de la Python Software Foundation y autor del libro Python, dice que "describir algo como "agudo" no es un cumplido para la cultura Python".

Los ingenieros de Python han tratado de mantener una distancia estratégica con respecto a los avances iniciales y de encerrar los parches en partes innecesarias de CPython que ofrezcan mínimos aumentos de velocidad a costa de la claridad. Cuando la velocidad es importante, un ingeniero de software de Python puede transferir las capacidades básicas de sincronización a extensiones escritas en dialectos. Por ejemplo, C, o utilizar PyPy, uno en el nombre del compilador de tiempo. También se puede acceder a Cython, que interpreta el contenido de Python en C y hace llamadas directas a la API de nivel C al traductor de Python.

El progreso de Python ha mejorado mucho con el proceso de la Propuesta de Mejora de Python (PEP). Esto incluyó la recopilación de los comentarios de la comunidad sobre los problemas y el registro de las decisiones sobre el marco de trabajo de Python. El estilo de codificación de Python se incluye en el PEP 8. Los PEP excelentes son calificados y evaluados por la comunidad de Python y el tablero de mando de Python.

La mejora del lenguaje se compara con el progreso en el uso de los informes de CPython. La lista de correo, Python-dev, es una discusión esencial sobre la evolución del lenguaje. Los problemas específicos se discuten en el depurador Roundup mantenido en Python.org. El desarrollo se llevó a cabo inicialmente en un repositorio de código fuente Mercurial autoabastecido, hasta que Python se trasladó a GitHub en enero de 2017.

Los descartes abiertos de CPython están disponibles en tres tipos, que determinan cuánto se incrementa el número de personalización.

Las variaciones hacia atrás son aquellas en las que se requiere que el código se rompa y debe transferirse de forma natural. La parte inicial del número de configuración aumenta. Estas vacunas son poco frecuentes. Por ejemplo, la versión 3.0 se descargó ocho años después de la 2.0.

Las tomas grandes o "estándar" se parecen a un reloj e incluyen nuevas características. La segunda parte del número de formulario aumenta. Todas las variantes principales admiten correcciones de errores mucho después de su lanzamiento.

Los rechazos por corrección de errores no nuevos se producen a intervalos regulares y tienen lugar cuando se ha corregido un número suficiente de errores ascendentes desde la última descarga. Las vulnerabilidades de seguridad también se definen en estos descartes: la tercera y última parte del número del formulario aumenta.

Muchas descargas alfa y beta también se descargan como un vistazo y para probarlas antes de las descargas finales. Aunque hay un calendario desagradable para cada exención, a menudo se pospone si la clave no está lista. El equipo de progreso de Python comprueba el estado del código ejecutando un enorme conjunto de pruebas unitarias durante la actualización y utilizando el sistema de unión ininterrumpida BuildBot. La comunidad de ingenieros de Python también ha contribuido con más de 86.000 módulos de programación. La verdadera conferencia escolar de Python es la PyCon. También hay excelentes programas de formación en Python, por ejemplo, Pyladies.

Aplicaciones de Python

Python es conocido por su naturaleza ampliamente útil que lo hace relevante en prácticamente todos los espacios de avance de la programación. Python puede ser utilizado en una plétora de formas para la

mejora; hay especificando territorios de aplicación donde Python puede ser aplicado.

Aplicaciones web

Podemos utilizar Python para crear aplicaciones web. Da bibliotecas para tratar con las convenciones de la web, por ejemplo, HTML y XML, JSON, manejo de correo electrónico, demanda, sopa hermosa, Feedparser, y así sucesivamente. Además, hay Frameworks. Por ejemplo, Django, Pyramid, Flask, y así sucesivamente para estructurar y desarrollar aplicaciones electrónicas. Algunas mejoras significativas son PythonWikiEngines, PythonBlogSoftware, y así sucesivamente.

Aplicaciones GUI de escritorio

Python ofrece una biblioteca Tk-GUI para crear UI en aplicaciones basadas en Python. Otra valiosa caja de herramientas incluye wxWidgets, Kivy, y es utilizable en algunos escenarios. El Kivy es bien conocido por las aplicaciones multitáctiles comp singulares.

Desarrollo de software

Python es útil para programar procesos avanzados. Funciona como un lenguaje de ayuda y puede ser utilizado para la fabricación de control y el tablero, pruebas, y así sucesivamente.

Científico y numérico

Python es la corriente principal y se utiliza generalmente en el cálculo lógico y numérico. Algunas bibliotecas y paquetes útiles son SciPy, Pandas, IPython, etc. SciPy es una biblioteca utilizada para la colección de paquetes de diseño, ciencia y aritmética.

Aplicaciones empresariales

Python se utiliza para fabricar aplicaciones empresariales, como los marcos de ERP y de negocios en línea. Tryton es una etapa de aplicación de estado anormal.

Aplicación basada en la consola

Puede utilizarse para aplicaciones basadas en la asistencia. Por ejemplo: IPython.

Aplicaciones basadas en audio o vídeo

Python es excelente para realizar diversas tareas y puede utilizarse para crear aplicaciones multimedia. Algunas de las aplicaciones auténticas son cplay, TimPlayer, etc.

Aplicaciones empresariales

Python puede ser utilizado para hacer aplicaciones que pueden ser utilizadas dentro de una empresa o una organización. Algunas aplicaciones en curso son Tryton, OpenERP, Picalo, etc.

Aplicaciones de las imágenes

Utilizando Python, se pueden crear algunas aplicaciones para una imagen. Varias aplicaciones incluyen VPython, Gogh, y imgSeek.

Por qué Python para realizar análisis de datos

Se pueden utilizar diferentes lenguajes de programación para la ciencia de datos (por ejemplo, SQL, Java, Matlab, SAS, R, y algunos más), pero Python es el más favorecido por los investigadores de datos entre los diferentes lenguajes de programación de esta lista. Python tiene algunas características excepcionales, incluyendo:

- Python es sólido y básico con el objetivo de que sea cualquier cosa menos difícil ganar competencia en el lenguaje. No tienes que preocuparte

por su estructura lingüística si eres un aficionado. Su sintaxis es similar a la de la escritura inglesa; por eso es un lenguaje de programación fácil de usar.

- Python es compatible con casi todas las plataformas, como Windows, Mac y Linux.
- Dispone de múltiples estructuras de datos con las que se pueden simplificar fácilmente los cálculos complejos.
- Python es un lenguaje de programación de código abierto que permite a los científicos de datos obtener bibliotecas y códigos predefinidos para realizar sus tareas.
- Python puede realizar la visualización de datos, la investigación de datos y el control de datos.
- Python sirve diferentes bibliotecas innovadoras para algoritmos y cálculos lógicos. Se pueden realizar diferentes cálculos lógicos complejos y de IA utilizando este lenguaje de forma eficaz en una estructura de frases moderadamente básica.

Capítulo 2: Configuración del entorno de Python para la ciencia de los datos

Ahora que has decidido trabajar con el código Python para ayudarte a hacer algo de tu propia programación y codificación, es el momento de ponerte a trabajar con la instalación de esto en tu ordenador. No podrás hacer todo el trabajo o usar cualquiera de los lenguajes de codificación si todas las diferentes partes y archivos que vienen con el código Python no están en tu computadora. Nos tomaremos un tiempo para ver cómo puedes instalar Python en tu ordenador, sin importar el sistema operativo con el que quieras trabajar.

Para estos pasos, vamos a asumir que estás obteniendo Python de www.python.org. Hay otros recursos donde puedes conseguir este lenguaje de programación, pero este suele ser el más fácil porque va a tener todos los archivos y extensiones que necesitas para hacer funcionar Python, y todos ellos son de uso gratuito. Otras fuentes pueden proporcionar algunas características más y otras cosas que necesitas, pero no siempre funcionan bien, o pueden faltar algunas de las partes que necesitas. Así que, vamos a sumergirnos y ver cómo podemos configurar las carpetas de Python con nuestro ordenador.

Instalación en Mac OS X

La primera opción que vamos a ver cuando queremos añadir Python a nuestro sistema operativo es el Mac OS X. Esta es una opción popular cuando se trata del sistema operativo de un ordenador, y va a funcionar muy bien con algunas de las codificaciones que decidamos hacer con Python. Sin embargo, es necesario comprobar el sistema porque algunas de las versiones de Python van a ser incluidas automáticamente en este sistema operativo. Para ver qué versión del programa Python se encuentra en su sistema, incluirá el siguiente código:

Python - V

Esto te mostrará la versión que tienes, por lo que aparecerá un número. También puedes elegir instalar Python 3 en este sistema si lo deseas, y no es necesario desinstalar la versión 2.X en el ordenador. Para comprobar la instalación de la versión 3.X, sólo tienes que abrir la aplicación de terminal y escribir el siguiente mensaje:

Python3 - V

Por defecto en OS X no se instalará Python 3 en absoluto. Si quieres usar Python 3, puedes instalarlo usando alguno de los instaladores que hay en Python.org. Este es un buen lugar para ir porque instalará todo lo que necesitas para escribir y ejecutar tus códigos con Python. Tendrá el shell de Python, las herramientas de desarrollo IDLE y el intérprete. A diferencia de lo que ocurre con Python 2.X, estas herramientas se instalan como una aplicación estándar en la carpeta de Aplicaciones.

Poder ejecutar el IDLE y el shell de Python va a depender de la versión que elijas y de algunas de tus preferencias personales. Puedes usar los siguientes comandos para ayudarte a iniciar el shell y las aplicaciones de IDLE:

- Para Python 2.X sólo hay que teclear "Idle"
- Para Python 3.X, basta con escribir "Idle3"

Instalación en un sistema Windows

Un sistema Windows también es capaz de trabajar con Python. Sin embargo, no va a haber una versión de este programa en Windows, simplemente porque el sistema Windows tiene su propio lenguaje de programación con el que se puede trabajar. Esto no significa que estés limitado a usar sólo ese lenguaje, pero sí significa que vas a tener que pasar algún tiempo instalando Python y eligiendo la versión de Python que te gustaría usar.

Hay algunos pasos que puedes utilizar para asegurarte de que puedes instalar el programa Python en tu sistema. Puede parecer un poco intimidante cuando empiezas, pero verás que estos pasos sólo llevan unos minutos, así que no es tan malo como parece. Algunos de los pasos que

debes seguir para asegurarte de que Python se instala correctamente en tu ordenador con Windows son:

1. Para empezar, es el momento de visitar la página oficial de descarga de Python y luego tomar el instalador de Windows. Usted es capaz de elegir la versión de Python que le gustaría trabajar. Su defecto de esto va a darle la versión de 32 bits del lenguaje, pero usted puede ir a través y haga clic en la versión de 64 bits si esto es lo que usted necesita para su sistema informático.
2. Ahora puedes pasar y pulsar el botón derecho del instalador y permitir que se ejecute como Administrador. Va a haber un punto que le da dos opciones. Usted querrá ir con el botón que permite personalizar la instalación.
3. En la siguiente pantalla, debe asegurarse de buscar en la parte de Características Opcionales y hacer clic en todas las casillas que están allí. Cuando esas casillas estén rellenadas, puede hacer clic en el botón Siguiente.
4. Mientras está en la parte de las Opciones Avanzadas, puede elegir la ubicación donde le gustaría instalar Python. Haz clic en Instalar y espera unos minutos para que la instalación termine. Luego cierre el instalador cuando haya terminado.
5. En este punto, puede configurar la variable PATH al sistema para asegurarse de que va a incluir todos los directorios que vienen con los paquetes y para asegurarse de que todos los otros componentes que usted necesita para mostrar también. Los pasos que puedes seguir para hacer que todo esto suceda incluyen:
    a. Comience esta parte abriendo su Panel de Control. Esto se hace fácilmente haciendo clic en su barra de tareas y luego escribiendo "Panel de control". Haz clic en el icono que aparece.
    b. Mientras estás en esta parte del proceso, es el momento de hacer una búsqueda del Entorno. Luego puede hacer clic en la parte que le permite editar las Variables de Entorno del Sistema. Desde aquí, puede hacer clic en el botón que le permite entrar en las Variables de Entorno.
    c. Diríjase a la sección de Variables de Usuario. Puede elegir editar la variable PATH que ya está ahí para su uso, o puede decidir crear una nueva.

d. Si ves que este sistema no te proporciona una variable para el PATH, entonces es el momento de crear una propia. Puedes crearla haciendo clic en NUEVO. Pon el nombre de la variable PATH y añádela a los directorios que quieras. Cierra todos los diálogos del Panel de Control y haz clic en Siguiente para terminar.

6. En este punto, puede abrir el símbolo del sistema en su ordenador Windows. Esto se hace haciendo clic en su menú de inicio y luego ir al sistema de Windows y luego en el símbolo del sistema. Escriba la palabra "python". Esto asegurará que el intérprete de Python se cargue para usted.

En este punto, el programa va a estar configurado y listo para usar en su sistema Windows. Usted puede optar por abrir las otras partes del sistema también para asegurarse de que todas las partes que usted necesita están en un solo lugar, y entonces es el momento de escribir cualquier código que desee cuando sea el momento adecuado.

¿Cómo instalar Python en su sistema Linux?

Ahora que hemos sido capaces de explorar cómo instalar Python en su ordenador Windows y su Mac OS X, es el momento de pasar a algunos de los pasos que puede utilizar para obtener este lenguaje instalado en su sistema Linux también. Hay muchas personas y programadores que se están pasando al sistema Linux, por lo que hay que dedicar algo de tiempo a aprender a utilizarlo para nuestras necesidades.

Lo primero que hay que hacer aquí es ver si hay una versión de Python 3 ya en su sistema. Puedes abrir el símbolo del sistema en Linux y luego ejecutar el siguiente código:

$ Python3 - - versión

Si estás en Ubuntu 16.10 o más reciente, entonces es un proceso sencillo para instalar Python 3.6. Sólo tienes que utilizar los siguientes comandos:

$ Sudo apt-get update

$ sudo apt-get install Python3.6

Si estás confiando en una versión más antigua de Ubuntu o en otra versión, entonces es posible que quieras trabajar con el PPA deadsnakes, u

otra herramienta, para ayudarte a descargar la versión 3.6 de Python. El código que necesitas para hacer esto incluye:

$ Sudo apt-get install software-properties-common

$ Sudo add-apt repository ppa: deadsnakes/ppa

Suoda apt-get update

$ Sudo apt-get install python3.6

Lo que hay que recordar aquí es que si has pasado algún tiempo trabajando con algunas de las otras distribuciones que vienen con Linux, es probable que tu sistema vaya a tener una versión de Python 3 ya instalada en él. Si no lo ves aquí, puedes optar por utilizar el gestor de paquetes de la distribución. O puedes seguir los pasos que tenemos arriba para ayudarte a instalar cualquier versión de Python que quieras antes de usar el programa.

Entender el intérprete en Python

La instalación estándar de Python, cuando lo haces en python.org, va a contener documentación, información sobre la licencia, y tres archivos principales para ejecutar que te ayudan a desarrollar y luego ejecutar los scripts que corren en python. Estos incluyen el intérprete de Python, IDLE, y Shell.

El primero es el intérprete de Python. Es importante porque es el responsable de ejecutar los scripts que decidas escribir. El intérprete puede convertir los archivos de script . py en instrucciones y luego los procesa según el tipo de código que escribas en el archivo.

También está el IDLE de Python. Esto se conoce como el entorno integrado de desarrollo y aprendizaje. Va a contener todas las herramientas que se necesitan para desarrollar tus programas en Python. Encontrarás herramientas para la depuración, el editor de texto, y el shell con esto. Dependiendo de la versión de Python que elijas, el IDLE puede ser extenso o bastante básico. También puedes elegir tu propio IDLE si hay otra versión que te guste más. A mucha gente le gusta encontrar nuevos editores de texto porque piensan que el de Python no tiene las características adecuadas, pero el de Python está bien para los códigos que vamos a hacer, así que no es necesario elegir uno diferente.

Capítulo 3: Aprendizaje automático con Scikit-Learn y cómo encaja con la ciencia de los datos

Scikit-Learn es una versátil biblioteca de Python que resulta útil para construir proyectos de ciencia de datos. Esta poderosa biblioteca le permite incorporar el análisis de datos y la minería de datos para construir algunos de los modelos más sorprendentes. Es predominantemente una biblioteca de aprendizaje automático, pero también puede satisfacer sus necesidades de ciencia de datos. Hay muchas razones por las que diferentes programadores e investigadores prefieren Scikit-Learn. Dada la exhaustiva documentación disponible en línea, es mucho lo que puedes aprender sobre Scikit-Learn, lo que facilitará mucho tu trabajo, incluso si no tienes experiencia previa. Sin dejar nada al azar, la API es eficiente, y la biblioteca es una de las bibliotecas de Python más coherentes y despejadas que se pueden encontrar en la ciencia de datos.

Al igual que muchas prolíficas bibliotecas de Python, Scikit-Learn es un proyecto de código abierto. Hay varias herramientas disponibles en Scikit-Learn que le ayudarán a realizar tareas de minería y análisis de datos fácilmente. Anteriormente en el libro, hemos mencionado que algunas bibliotecas de Python se pueden cortar a través de diferentes dimensiones. Esta es una de ellas. Cuando se aprende sobre las bibliotecas principales de Python, siempre es importante que se entienda que se pueden implementar a través de diferentes dimensiones.

Scikit-Learn está construido sobre Matplotlib, SciPy y NumPy. Por lo tanto, el conocimiento de estas bibliotecas independientes le ayudará a obtener una experiencia más fácil usando Scikit-Learn.

Usos de Scikit-Learn

¿Cómo ayuda Scikit-Learn a su curso de análisis de datos? El análisis de datos y el aprendizaje automático están entrelazados. A través de Scikit-Learn, puedes implementar los datos en tus proyectos de aprendizaje automático de las siguientes maneras:

Clasificación

Las herramientas de clasificación son algunas de las herramientas básicas en el análisis de datos y el aprendizaje automático. A través de estas herramientas, se puede determinar la categoría adecuada necesaria para los datos, especialmente para los proyectos de aprendizaje automático. Un buen ejemplo de cómo se utilizan los modelos de clasificación es la separación de los correos electrónicos de spam de los legítimos.

Utilizando Scikit-Learn, algunos de los algoritmos de clasificación con los que se encontrará incluyen el bosque aleatorio, los vecinos más cercanos y las máquinas de vectores de apoyo.

Regresión

Las técnicas de regresión en Scikit-Learn requieren la creación de modelos que identifiquen de forma autónoma las relaciones entre los datos de entrada y los de salida. A partir de estas herramientas, es posible realizar predicciones precisas, y quizás podamos ver la mejor ilustración de este enfoque en los mercados financieros o las bolsas de valores. Los algoritmos de regresión más comunes utilizados en Scikit-Learn son Lasso, la regresión de cresta y las máquinas de vectores de apoyo.

Agrupación

El clustering es un enfoque de aprendizaje automático en el que los modelos crean de forma independiente grupos de datos con características similares. Mediante el uso de clusters, se pueden crear varios grupos de datos a partir de un amplio conjunto de datos. Muchas organizaciones acceden a datos de clientes de diferentes regiones. Utilizando algoritmos de clustering, estos datos pueden ser agrupados según las regiones. Algunos de los algoritmos importantes que debe aprender son el desplazamiento de la media, la agrupación espectral y K-means.

Selección de modelos

En la selección de modelos, utilizamos diferentes herramientas para analizar, validar, comparar y contrastar, y finalmente elegir las condiciones ideales que utilizarán nuestros proyectos de análisis de datos en

funcionamiento. Para que estos módulos sean eficaces, podemos mejorar aún más su precisión utilizando enfoques de ajuste de parámetros como las métricas, la validación cruzada y los protocolos de búsqueda en la red.

Reducción de la dimensionalidad

En su forma bruta, muchos conjuntos de datos contienen un elevado número de variables aleatorias. Esto crea un enorme problema para los fines analíticos. A través de la reducción de la dimensionalidad, es posible reducir los desafíos esperados al tener tales variables en el conjunto de datos. Si, por ejemplo, está trabajando en visualizaciones de datos y necesita asegurarse de que el resultado es eficiente, una buena alternativa sería eliminar por completo los valores atípicos. Para ello, algunas técnicas que podría emplear en Scikit-Learn incluyen la factorización de matrices no negativas, el análisis de componentes principales y la selección de características.

Preprocesamiento

En la ciencia de datos, las herramientas de preprocesamiento utilizadas en Scikit-Learn ayudan a extraer características únicas de grandes conjuntos de datos. Estas herramientas también ayudan a la normalización. Por ejemplo, estas herramientas son útiles cuando se necesita obtener características únicas de los datos de entrada, como los textos, y utilizar las características para fines analíticos.

Representación de datos en Scikit-Learn

Si está trabajando individualmente o en equipo en un modelo de aprendizaje automático, el conocimiento de Scikit-Learn le ayudará a crear modelos eficaces. Antes de empezar a trabajar en cualquier proyecto de aprendizaje automático, es obligatorio realizar un curso de actualización sobre la representación de datos. Esto es importante para que puedas presentar los datos de manera que tus ordenadores o modelos los comprendan fácilmente. Recuerda que el tipo de datos que alimentes al ordenador afectará al resultado. Scikit-Learn se utiliza mejor con datos tabulares.

Datos tabulares

Las tablas son simples representaciones bidimensionales de unos datos. Las filas de una tabla identifican las características únicas de cada elemento dentro del conjunto de datos. Las columnas, por su parte, representan las cantidades o cualidades de los elementos que se quieren analizar del conjunto de datos. En nuestra ilustración para esta sección, utilizaremos el famoso conjunto de datos Iris. Por suerte para usted, Scikit-Learn viene con el conjunto de datos Iris cargado en su biblioteca, por lo que no necesita utilizar enlaces externos para cargarlo. Importarás este conjunto de datos a tu entorno de programación utilizando la biblioteca Seaborn como un DataFrame en Pandas.

El conjunto de datos Iris viene precargado en Scikit-Learn, por lo que llevarlo a su interfaz no debería ser un problema. Cuando haya terminado, la salida debería darle una tabla cuyas columnas incluyen lo siguiente:

- longitud_de_sépalos
- ancho_de_sépalos
- longitud_de_pétalos
- ancho_de_pétalos
- especies

Podemos deducir mucha información de este resultado. Cada fila representa una flor individual bajo observación. En este conjunto de datos, el número de filas infiere el número total de flores presentes en el conjunto

de datos de Iris. En Scikit-Learn, no utilizaremos el término filas, sino que nos referiremos a ellas como muestras. Basándonos en esta afirmación, se deduce que el número de filas en el conjunto de datos Iris se identifica como n_muestras.

En el mismo sentido, las columnas del conjunto de datos Iris proporcionan información cuantitativa sobre cada una de las filas (muestras). Las columnas, en Scikit-Learn, se identifican como características; por lo tanto, el número total de columnas en el conjunto de datos Iris se identificará como n_características.

Lo que hemos hecho hasta ahora es proporcionar la explicación más sencilla de una tabla de Scikit-learn utilizando el conjunto de datos Iris.

Matriz de características

A partir de los datos que hemos obtenido del conjunto de datos Iris, podemos interpretar nuestros registros como una matriz o una matriz bidimensional. Si elegimos utilizar la matriz, lo que tenemos es una matriz de características.

Por defecto, las matrices de características en Scikit-Learn se almacenan en variables identificadas como x. Utilizando los datos de la tabla anterior para crear una matriz de características, tendremos una matriz bidimensional que asume la siguiente forma [n_muestras, n_características]. Como estamos introduciendo arrays, esta matriz será, en la mayoría de los casos, parte de un array en NumPy. Alternativamente, también se puede utilizar Pandas DataFrames para representar la matriz de características.

Las filas en Scikit-Learn (muestras) aluden a objetos singulares que están contenidos en el conjunto de datos objeto de observación. Si, por ejemplo, estamos tratando con datos sobre flores según el conjunto de datos Iris, nuestra muestra deberá ser de flores. Si se trata de estudiantes, las muestras tendrán que ser estudiantes individuales. Las muestras se refieren a cualquier objeto bajo observación que pueda ser cuantificado en la medición.

Las columnas en Scikit-Learn (características) aluden a observaciones descriptivas únicas que utilizamos para cuantificar las muestras. Estas observaciones deben ser de naturaleza cuantitativa. Los valores utilizados en las características deben ser valores reales, aunque en algunos casos, puede encontrar datos con valores discretos o booleanos.

Matrices de destino

Ahora que entendemos lo que es la matriz de características (x), y su composición, podemos dar un paso más y ver las matrices de destino. Las matrices objetivo también se denominan etiquetas en Scikit-Learn. Por defecto, se identifican como (y).

Una de las características distintivas de las matrices objetivo es que deben ser unidimensionales. La longitud de un array objetivo es n_muestras. Encontrará arrays objetivo en la serie Pandas o en arrays NumPy. Un array objetivo debe tener siempre etiquetas o clases discretas, y los valores deben ser continuos si se utilizan valores numéricos. Para empezar, es conveniente aprender a trabajar con arrays objetivo unidimensionales. Sin embargo, esto no debe limitar su imaginación. A medida que avance en el análisis de datos con Scikit-Learn, se encontrará con estimadores avanzados que pueden soportar más de una matriz objetivo. Esto se representa como una matriz bidimensional, en la forma [n_muestras, n_objetivos].

Recuerde que existe una clara distinción entre las matrices de destino y las columnas de características. Para ayudarle a entender la diferencia, tenga en cuenta que las matrices objetivo identifican la cantidad que necesitamos observar del conjunto de datos. Desde nuestro conocimiento de la estadística, las matrices objetivo serían nuestras variables dependientes. Por ejemplo, si se construye un modelo de datos a partir del conjunto de datos de Iris que pueda utilizar las mediciones para identificar las especies de flores, la matriz objetivo en este modelo sería la columna de especies.

Los siguientes diagramas permiten distinguir mejor el vector objetivo y la matriz de características:

Diagrama de un vector objetivo

Diagrama de una matriz de características

Entender la API

Antes de empezar a utilizar Scikit-Learn, debería tomarse un tiempo y aprender sobre la API. Según el documento de la API de Scikit-Learn, los siguientes principios son la base de la API de Scikit-Learn:

Inspección

Debe mostrar todos los valores de los parámetros en uso como atributos públicos

Consistencia

Debe utilizar un número limitado de métodos para sus objetos. De este modo, todos los objetos utilizados deben tener una interfaz común y, para facilitarle el trabajo, asegúrese de que la documentación es sencilla y coherente en todos los casos.

Jerarquía de objetos limitada

Los únicos algoritmos que deben utilizar cadenas básicas de Python son los que pertenecen a las clases de Python.

Valores predeterminados sensatos

En el caso de los modelos que necesitan parámetros específicos exclusivos para su uso, la biblioteca Scikit-Learn definirá automáticamente los valores por defecto aplicables

Composición

Dada la naturaleza de las tareas de aprendizaje automático, la mayoría de las tareas que realice se representarán como secuencias, especialmente en

relación con los principales algoritmos de aprendizaje automático.

¿Por qué es importante entender estos principios? Son la base sobre la que se construye Scikit-Learn; por lo tanto, le facilitan el uso de esta biblioteca de Python. Todos los algoritmos que se utilizan en Scikit-Learn, especialmente los algoritmos de aprendizaje automático, utilizan la API de estimadores para su implementación. Debido a esta API, puedes disfrutar de la consistencia en el desarrollo para diferentes aplicaciones de aprendizaje automático.

Capítulo 4: Aplicación del aprendizaje automático mediante la biblioteca Scikit-Learn

Para entender cómo se utiliza la biblioteca Scikit-Learn en el desarrollo de un algoritmo de aprendizaje automático, vamos a utilizar el conjunto de datos "Sales_Win_Loss del repositorio Watson de IBM" que contiene datos obtenidos de la campaña de ventas de un proveedor mayorista de piezas de automóvil. Construiremos un modelo de aprendizaje automático para predecir qué campaña de ventas será ganadora y cuál tendrá pérdidas.

El conjunto de datos puede ser importado usando Pandas y explorado usando técnicas de Pandas como "head (), tail (), y dtypes ()". Las técnicas de trazado de "Seaborn" se utilizarán para visualizar los datos. Para procesar los datos el "preprocesamiento" de Scikit-Learn. LabelEncoder ()" y "train_test_split ()" para dividir el conjunto de datos en un subconjunto de entrenamiento y otro de prueba.

Para generar predicciones a partir de nuestro conjunto de datos, se utilizarán tres algoritmos diferentes, a saber, "Clasificación de vectores de soporte lineal y clasificador del vecino más cercano". Para comparar el rendimiento de estos algoritmos se utilizará la técnica de la biblioteca Scikit-Learn "accuracy_score". La puntuación del rendimiento de los modelos se puede visualizar utilizando Scikit-Learn y la visualización "Yellowbrick".

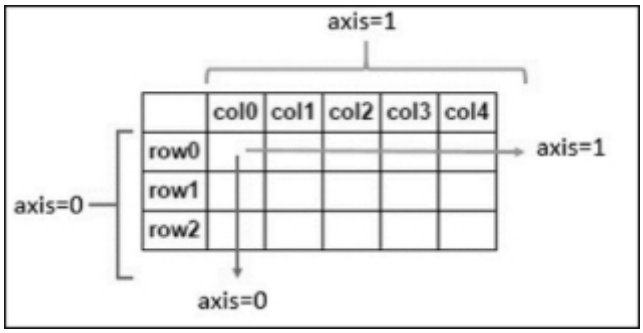
Importar el conjunto de datos

Para importar el "conjunto de datos Sales_Win_Loss del repositorio Watson de IBM", el primer paso es importar el módulo "Pandas" utilizando "import pandas as pd".

A continuación, aprovechamos una variable url como: "https://community.watsonanalytics. com/wpcontent/uploads/2015/04/ WA_Fn-UseC_-Sales-Win-Loss.csv" para almacenar la URL desde la que se descargará el conjunto de datos.

Ahora, se utilizará la técnica "read_csv () as sales_data = pd. read_csv(url)" para leer el archivo "csv o valores separados por comas" anterior, que es suministrado por el módulo Pandas. El archivo csv se convertirá entonces en un marco de datos de Pandas, con la variable de retorno como "sales_data", donde se almacenará el marco.

Para los nuevos usuarios de 'Pandas', la técnica "pd. read csv ()" en el código mencionado anteriormente generará una estructura de datos tabular llamada "marco de datos", donde un índice para cada fila está contenido en la primera columna, y la etiqueta/nombre para cada columna en la primera fila son los nombres de columna iniciales adquiridos del conjunto de datos. En el fragmento de código anterior, la variable "datos de ventas" da lugar a una tabla como la que se muestra en la imagen siguiente.



En el diagrama anterior, las "fila0, fila1, fila2" representan el índice de un registro individual, y las "col0, col1, col2" representan los nombres de columnas individuales o características del conjunto de datos.

Con este paso, habrás almacenado con éxito una copia del conjunto de datos y lo habrás transformado en un marco "Pandas".

Ahora, utilizando la técnica "head () as Sales_data. Head ()", los registros del marco de datos pueden mostrarse como se muestra a continuación para tener una "sensación" de la información contenida en el conjunto de datos.

	opportunity number	supplies subgroup	supplies group	region	route to market	elapsed days in sales stage	opportunity result
0	1641984	Exterior Accessories	Car Accessories	Northwest	Fields Sales	76	Won
1	1658010	Exterior Accessories	Car Accessories	Pacific	Reseller	63	Loss
2	1674737	Motorcycle Parts	Performance & Non-auto	Pacific	Reseller	24	Won
3	1675224	Shelters & RV	Performance & Non-auto	Midwest	Reseller	16	Loss

OBJ

Exploración de datos

Ahora que tenemos nuestra propia copia del conjunto de datos, que se ha transformado en un marco de datos "Pandas", podemos explorar rápidamente los datos para comprender qué información puede decirse, puede obtenerse de ellos, y en consecuencia planificar un curso de acción.

En cualquier proyecto de ML, la exploración de datos suele ser una fase muy crítica. Incluso una exploración rápida del conjunto de datos puede ofrecernos información significativa que podría perderse fácilmente de otra manera, y esta información puede proponer preguntas significativas que luego podemos intentar responder utilizando nuestro proyecto.

Aquí se utilizarán algunas librerías de Python de terceros para ayudarnos con el procesamiento de los datos, de manera que podamos utilizar eficientemente estos datos con los potentes algoritmos de Scikit-Learn. La misma técnica "head ()" que utilizamos para ver algunos registros iniciales del conjunto de datos importados en la sección anterior puede utilizarse aquí. De hecho, "(head)" es capaz de hacer mucho más que mostrar los registros de datos y personalizar la técnica "head ()" para mostrar sólo los registros seleccionados con comandos como "sales_data.head(n=2)". Este comando mostrará selectivamente los dos primeros registros del conjunto de datos. A simple vista, es obvio que columnas como "Región" y "Grupo de suministros" contienen datos de cadena, mientras que columnas como "Resultado de la oportunidad", "Número de la oportunidad", etc. están compuestas por valores enteros. También se puede ver que hay identificadores únicos para cada registro en la columna "Número de oportunidad".

Del mismo modo, para mostrar los registros seleccionados de la parte inferior de la tabla, se puede utilizar "tail() as sales_data.tail()".

Para ver los diferentes tipos de datos disponibles en el conjunto de datos, se puede utilizar la técnica de Pandas "dtypes() as sales_data.dtypes". Con esta información, se pueden listar las columnas de datos disponibles en el marco de datos con sus respectivos tipos de datos. Podemos averiguar, por ejemplo, que la columna "Subgrupo de suministros" es un tipo de datos "objeto" y que la columna "Tamaño del cliente por ingresos" es un "tipo de datos entero". Por lo tanto, tenemos una comprensión de las columnas que contienen valores enteros o datos de cadena.

Visualización de datos

En este punto, hemos terminado con los pasos básicos de exploración de datos, por lo que no intentaremos construir algunos gráficos atractivos para representar la información visualmente y descubrir otras narrativas ocultas de nuestro conjunto de datos.

De todas las bibliotecas de Python disponibles que proporcionan características de visualización de datos; "Seaborn" es una de las mejores opciones disponibles, por lo que vamos a utilizar la misma. Asegúrese de que el módulo de gráficos de Python proporcionado por "Seaborn" ha sido instalado en su sistema y está listo para ser utilizado. Ahora siga los siguientes pasos, genere el gráfico deseado para el conjunto de datos:

Paso 1-Importe el módulo "Seaborn" con el comando "import seaborn as sns".

Paso 2-Importar el módulo "Matplotlib" con el comando "import matplotlib.pyplot as plt".

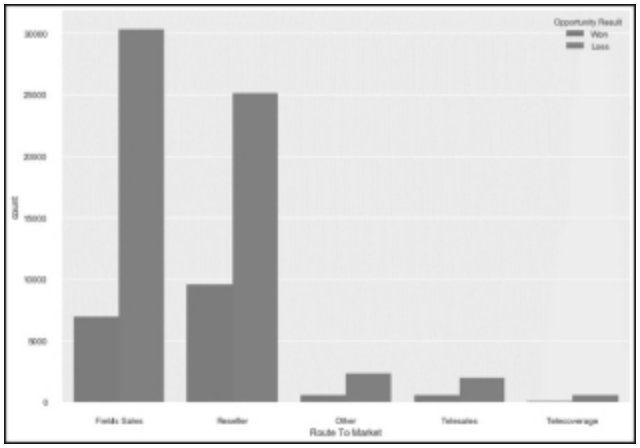
Paso 3-Para establecer el "color de fondo" del gráfico como blanco, utilice el comando "sns.set(style="whitegrid", color_codes=True)".

Paso 4-Para establecer el "tamaño del gráfico" para todos los gráficos, utilice el comando "sns.set(rc={'figure.figsize':(11.7,8.27)})".

Paso 5-Para generar un "countplot", utilice el comando "sns.countplot('Route To Market', data=sales_data, hue = 'Opportunity Result')".

Paso 6-Para eliminar los márgenes superior e inferior, utilice el comando "sns.despine(offset=10, trim=True)".

Paso 7-Para visualizar el gráfico, utilice el comando "plotplt.show()".



Recapitulación rápida-Los módulos "Seaborn" y "Matplotlib" fueron importados primero. A continuación, se utilizó la técnica "set()" para definir las distintas características de nuestro gráfico, como el estilo y el color del mismo. El fondo del gráfico se definió como blanco utilizando el fragmento de código "sns.set(style= "whitegrid", color codes= True)". A continuación, el tamaño del gráfico se definió mediante el comando "sns.set(rc= {'figure.figsize':(11.7,8.27)})" que define el tamaño del gráfico como "11.7px y 8.27px".

A continuación, se utilizó el comando "sns.countplot('Route To Market',data= datos de ventas, hue='Opportunity Result')" para generar el gráfico. La técnica "countplot()" permite la creación de un gráfico de recuento, que puede exponer múltiples argumentos para personalizar el gráfico de recuento según nuestras necesidades. Como parte del primer argumento de "countplot()", el eje X se definió como la columna "Route to Market" del conjunto de datos. El siguiente argumento se refiere a la fuente del conjunto de datos, que sería el marco de datos "sales_data" que importamos anteriormente. El tercer argumento es el color de los gráficos de barras que se definieron como "azul" para la columna "ganada" y "verde" para la columna "perdida".

Preprocesamiento de datos

A estas alturas, debería tener una comprensión clara de la información disponible en el conjunto de datos. Desde el paso de exploración de datos, hemos establecido que la mayoría de las columnas de nuestro conjunto de datos son "datos de cadena", pero "Scikit-Learn" sólo puede procesar datos numéricos. Afortunadamente, la biblioteca de Scikit-Learn nos ofrece muchas formas de convertir los datos de cadena en datos numéricos, por ejemplo, la técnica "LabelEncoder()". Para transformar las etiquetas categóricas del conjunto de datos, como "won" y "loss", en valores numéricos, utilizaremos la técnica "LabelEncoder()".

Observemos las imágenes siguientes para ver lo que intentamos conseguir con la técnica "LabelEncoder()". La primera imagen contiene una columna etiquetada como "color" con tres registros: "Rojo", "Verde" y "Azul". Utilizando la técnica "LabelEncoder()", el registro de la misma columna "color" puede convertirse en valores numéricos, como se muestra en la segunda imagen.

	Color
0	Red
1	Green
2	Blue

	Color
0	1
1	2
2	3

Comencemos ahora el verdadero proceso de conversión. Utilizando la técnica "fit transform()" dada por "LabelEncoder()", las etiquetas de la columna categórica como "Route To Market" pueden ser codificadas y convertidas en etiquetas numéricas comparables a las mostradas en los diagramas anteriores. La función "fit transform()" requiere etiquetas de

entrada identificadas por el usuario y, en consecuencia, devuelve etiquetas codificadas.

Para saber cómo se realiza la codificación, veamos rápidamente un ejemplo. El ejemplo de código siguiente constituye datos de cadena en forma de una lista de ciudades como ["París", "París", "Tokio", "Ámsterdam"] que se codificarán en algo comparable a "[2, 2, 1,3]".

Paso 1-Para importar el módulo requerido, utilice el comando "from sklearn import preprocessing".

Paso 2-Para crear el objeto codificador de etiquetas, utilice el comando "le = preprocessing.LabelEncoder()".

Paso 3-Para convertir las columnas categóricas en valores numéricos, utilice el comando

"valor_codificado = le.fit_transform(["París", "París", "Tokio", "Ámsterdam"])"

"print(valor_codificado) [1 1 2 0]"

¡Y ahí lo tienes! Acabamos de convertir nuestras etiquetas de datos de cadena en valores numéricos. El primer paso fue importar el módulo de preprocesamiento que ofrece la técnica "LabelEncoder()". A continuación, el desarrollo de un objeto que representa el tipo "LabelEncoder()". Luego se utilizó la función "fit_transform()" del objeto para distinguir entre las distintas clases de la lista ["París", "París", "Tokio", "Ámsterdam"] y dar salida a los valores codificados de "[1 1 20]".

¿Ha observado que la técnica "LabelEncoder()" ha asignado los valores numéricos a las clases en orden alfabético según la letra inicial de las mismas, por ejemplo a "(A)msterdam" se le ha asignado el código "0", a "(P)aris" el código "1" y a "(T)okyo" el código "2"?

Creación de subconjuntos de entrenamiento y prueba

Para conocer las interacciones entre las distintas características y cómo éstas influyen en la variable objetivo, hay que entrenar un algoritmo de ML en una colección de información. Para ello, debemos dividir el conjunto de datos completo en dos subconjuntos. Un subconjunto servirá como conjunto de datos de entrenamiento, que se utilizará para entrenar nuestro algoritmo para construir modelos de aprendizaje automático. El otro subconjunto

servirá como conjunto de datos de prueba, que se utilizará para comprobar la precisión de las predicciones generadas por el modelo de aprendizaje automático.

Capítulo 5: Estructuras de datos

Python se basa en tres estructuras de referencia: tuplas, listas y diccionarios. Estas estructuras son en realidad objetos que pueden contener otros objetos. Tienen utilidades muy diferentes y permiten almacenar información de todo tipo.

Estas estructuras tienen una serie de características comunes:

- Para extraer uno o varios objetos de una estructura, siempre utilizamos el botón []
- Para las estructuras indexadas numéricamente (tuplas y listas), las estructuras están indexadas a 0 (la primera posición es la posición 0)

Las tuplas

Se trata de una estructura que agrupa múltiples objetos en orden indexado. Su forma no es modificable (inmutable) una vez creada y se define mediante paréntesis. Sólo tiene una dimensión. En una tupla se puede almacenar cualquier tipo de objeto. Por ejemplo, si se quiere crear una tupla con diferentes objetos, se utiliza

tup1 = (1, Verdadero, 7.5.9)

También se puede crear una tupla utilizando la función tuple (). El acceso a los valores de una tupla se realiza mediante la clásica indexación de estructuras. Así, si queremos acceder al tercer elemento de nuestra tupla, utilizamos

En []: tup1 [2]

Fuera []: 7,5

Las tuplas pueden ser interesantes porque requieren poca memoria. Por otro lado, se utilizan como salidas de funciones que devuelven varios valores. Las tuplas como estructuras son objetos. Tienen métodos que son limpios. Estos son pocos para una tupla:

En []: tup1.count (9)

Fuera []: 1

A menudo preferimos las listas que son más flexibles.

Listas

La lista es la estructura de referencia en Python. Es modificable y puede contener cualquier objeto.

Creación de una lista

Creamos una lista utilizando corchetes:

list1 = [3,5,6, True]

También puede utilizar la función list (). La estructura de una lista es editable. Tiene muchos métodos:

.append (): añade un valor al final de la lista

.insert (i, val): inserta un valor en el índice i

.pop (i): recupera el valor del índice i

.reverse (): invierte la lista

.extend (): extiende la lista con una lista de valores

Nota-Todos estos métodos modifican la lista, el equivalente en términos de código clásico sería el siguiente:

liste1.extend (list2)

equivalente a

lista1 = lista1 + lista2

Las listas tienen otros métodos que incluyen:

.index (val): devuelve el índice del valor val

.count (val): devuelve el número de apariciones de val

.remove (val): elimina la primera aparición del valor val de la lista

Extraer un elemento de una lista

Como hemos visto anteriormente, es posible extraer un elemento utilizando los corchetes:

lista1 [0]

A menudo nos interesa la extracción de varios elementos. Se hace mediante el uso de los dos puntos:

lista1 [0: 2] o lista1 [: 2]

En este ejemplo, vemos que este sistema extrae dos elementos: el elemento indexado en 0 y el indexado en la posición 1. Así que tenemos como regla que i: j va del elemento I incluido en el elemento j no incluido. He aquí otros ejemplos:

Extraer el último elemento

lista1 [-1]

Extraer los 3 últimos elementos list1 [-3: -1] o list1 [-3:]

Un ejemplo concreto:

Supongamos que queremos crear una lista de países. Estos países están ordenados en la lista según su población. Intentaremos extraer los tres primeros y los tres últimos.

En []: country_list = ["China", "India", "Estados Unidos", "Francia", "España", "Suiza"]

En []: print (lista_de_países [: 3])

['China', 'India', 'Estados Unidos']

En []: print (lista_de_países [-3:])

['Francia', 'España', 'Suiza']

En []: country_list.reverse ()

print (liste_pays)

['Suiza', 'España', 'Francia', 'Estados Unidos', 'India', 'China']

Las listas de comprensión

Son listas construidas de forma iterativa. Suelen ser muy útiles porque son más eficientes que el uso de bucles para construir listas. He aquí un ejemplo sencillo:

En []: list_init = [4,6,7,8]

list_comp = [val ** 2 para val en list_init si val% 2 == 0]

La lista comp_list permite almacenar los elementos pares de list_init puestos al cuadrado.

Tendremos:

En []: print (lista_comp)

[16,36,64]

Esta noción de lista de comprensión es muy eficaz. Evita el código inútil (bucles en una lista) y tiene un mejor rendimiento que la creación de una lista de forma iterativa. También existe en los diccionarios, pero no en las tuplas que son inmutables. Podremos utilizar las listas de comprensión en el marco de la manipulación de tablas de datos.

Cadenas-Listas de caracteres

Las cadenas en Python están codificadas por defecto (desde Python 3) en Unicode. Puedes declarar una cadena de caracteres de tres maneras:

string1 = "Python para el científico de datos"

string2 = 'Python para el científico de datos'

string3 = "" "Python para el científico de datos" ""

La última permite tener cadenas en varias líneas. La mayoría de las veces utilizaremos la primera. Una cadena es en realidad una lista de caracteres, y podremos trabajar sobre los elementos de una cadena como sobre los de una lista:

En []: imprimir (cadena1 [: 6])

print (cadena1 [-14:])

print (cadena1 [3:20 p.m.])

Las cadenas de datos pueden transformarse fácilmente en listas:

```
En []: # separamos los elementos usando espacio
list1 = chaine1.split ()
imprimir (lista1)
['Python', 'for', 'the', 'Data', 'Scientist']
En []: # unimos los elementos con espacio
cadena1bis = "" .join (lista1)
print (chaine1bis)
```

Diccionarios

Los diccionarios constituyen una tercera estructura central para desarrollar en Python. Permiten el almacenamiento clave-valor. Hasta ahora, hemos utilizado elementos basados en la indexación numérica. Así, en una lista, se accede a un elemento utilizando su posición list1 [0]. En un diccionario, accederemos a un elemento utilizando una clave definida al crear el diccionario. Definimos un diccionario con llaves:

```
dict1 = {"cle1": valor1, "cle2": valor2, "cle3": valor3}
```

Esta estructura no requiere ninguna homogeneidad de tipo en los valores. A partir de ella, podemos tener una lista como valor1, un booleano como valor2, y un entero un valor3.

Para acceder a un elemento de un diccionario, utilizamos:

```
En []: dict1 ["cle2"]
Salida []: valor2
```

Para mostrar todas las claves de un diccionario, utilizamos

```
En []: dict1.keys
Out []: ("cle1", "cle2", "cle3")
```

Para mostrar todos los valores de un diccionario, utilizamos

```
En []: dict1.items ()
Out []: (valor1, valor2, valor3)
```

Se puede modificar o añadir fácilmente una clave a un diccionario:

```
En []: dict1 ["clave4"] = valor4
```

También puede eliminar una clave (y el valor asociado) en un diccionario:

```
En []: del dict1 ["cle4"]
```

En cuanto tengas más experiencia en Python, utilizarás más diccionarios. Al principio, tendemos a favorecer los diccionarios de listas porque suelen ser más intuitivos (con indexación numérica). Sin embargo, los pythonistas más expertos se darán cuenta rápidamente de la utilidad de los diccionarios. En particular, podremos almacenar tanto los datos como los parámetros de un modelo de forma muy sencilla. Además, la flexibilidad del bucle for de Python se adapta muy bien a los diccionarios y los hace muy eficaces cuando están bien construidos.

Programación

Las condiciones

Una condición en Python es muy sencilla de implementar; es una palabra clave. Como se mencionó antes, el lenguaje Python se basa en la indentación de su código. Utilizaremos un desplazamiento para esta sangría con cuatro espacios. Afortunadamente, herramientas como Spyder o Jupyter notebooks generarán automáticamente esta sangría.

Aquí está nuestra primera condición, que significa: si a es verdadero, entonces muestra "es verdadero":

```
si a es True:
print ("es verdad")
```

No hay salida de la condición; es la sangría la que nos permitirá manejarla. Generalmente, también nos interesa el complemento de esta condición; para ello utilizaremos else:

```
si a es True:
print ("es verdad")
Si no:
print ("no es cierto")
```

Podemos tener otro caso; si nuestra variable a no es necesariamente un booleano, utilizamos elif:

```
si a es True:
print ("es verdad")
elif a es False:
print ("está mal")
Si no:
print ("no es un booleano")
```

Los bucles

Los bucles son elementos centrales de la mayoría de los lenguajes de programación. Python no rompe esta regla. Sin embargo, hay que tener mucho cuidado con un lenguaje interpretado como Python. En efecto, el tratamiento de los bucles es lento en Python, y lo utilizaremos en bucles con pocas iteraciones. Evitamos crear un bucle que se repita miles de veces en las líneas de un array de datos. Sin embargo, podemos utilizar un bucle sobre las columnas de una tabla de datos hasta unas decenas de columnas.

El bucle for

El bucle de Python tiene un formato algo específico; es un bucle sobre los elementos de una estructura. Vamos a escribir:

```
para elem en [1, 2]:
print (elem)
```

Este trozo de código le permitirá mostrar 1 y 2. Así, el iterador del bucle (elem en nuestro caso) toma los valores de los elementos de la estructura en la segunda posición (después del in). Estos elementos pueden estar en diferentes estructuras, pero en general se preferirán las listas.

Funciones de rango, cremallera y enumeración

Estas tres funciones son muy útiles; permiten crear objetos específicos que pueden ser útiles en su código para sus bucles. La función range () se utiliza para generar una secuencia de números, empezando por un número dado o 0 por defecto y hasta un número no incluido:

```
En []: print (list (range (5)))
[0, 1, 2, 3, 4]
En []: print (list (range (2,5)))
[2, 3, 4]
En []: print (list (range (2,15,2)))
[2, 4, 6, 8, 10, 12, 14]
```

Vemos aquí que el objeto rango creado puede ser fácilmente transformado en una lista con la lista ().

En un bucle, esto da:

```
para i en el rango (11):
imprimir (i)
```

Las funciones zip y enumerar son también funciones útiles en los bucles y utilizan listas.

La función enumerar () devuelve el índice y el elemento de una lista. Si tomamos nuestra lista de países utilizada anteriormente:

En []: para i, en enumerar (lista_de_países):

imprimir (i, a)

1. Suizo
2. España
3. Francia
4. Estados Unidos
5. India
6. China

La función zip permitirá enlazar muchas listas e iterar simultáneamente elementos de estas listas.

Si, por ejemplo, queremos incrementar simultáneamente los días y el tiempo, podemos utilizar

En []: para el día, el tiempo en zip (["lunes", "martes"], ["bonito", "malo"]):

print ("% s, hará% s"% (day.capitalize (), weather))

El lunes, será agradable

El martes, será malo

En este código, utilizamos zip () para tomar un par de valores en cada iteración. La segunda parte es una manipulación de las cadenas de caracteres. Si una de las listas es más larga que la otra, el bucle se detendrá en cuanto llegue al final de una de ellas.

Por ejemplo, podemos vincular la enumeración y la cremallera en un código:

En []: para i, (día, tiempo) en enumerar (zip (["lunes", "martes"], ["bueno", "malo"]):

print ("% i:% s, hará% s"% (i, day.capitalize (), meteo))

0: El lunes, será agradable

1: El martes, será malo

Vemos aquí que i es la posición del elemento i.

Nota-Reemplazar en una cadena.

Bucle while

Python también permite utilizar un bucle while () que es menos utilizado y se parece mucho al bucle while que podemos cruzar en otros lenguajes. Para salir de este bucle, podemos utilizar una estación con una condición. Atención, debemos incrementar el índice en el bucle, a riesgo de encontrarnos en un caso de bucle infinito.

Podemos tener:

```
i = 1
mientras i <100:
i + = 1
si i> val_stop:
romper
imprimir (i)
```

Este código añade uno a i en cada bucle y se detiene cuando i alcanza val_stop, es decir, 100.

Nota-El incremento en Python puede tomar varias formas i = i + 1 o i + = 1.

Ambos enfoques son equivalentes en términos de rendimiento; se trata de elegir el que más le convenga.

Capítulo 6: Algoritmos y modelos de ciencia de datos

Los algoritmos utilizados en la ciencia de datos pueden dividirse en varias categorías, principalmente el aprendizaje supervisado, el aprendizaje no supervisado y, hasta cierto punto, el aprendizaje semisupervisado.

Como su nombre indica, el aprendizaje supervisado cuenta con la ayuda de la interacción humana, ya que el científico de datos debe proporcionar la entrada y la salida para obtener un resultado de las predicciones que se realizan durante el proceso de entrenamiento. Una vez completado el entrenamiento, el algoritmo utilizará lo aprendido para aplicarlo a datos nuevos pero similares.

Nos vamos a centrar en este tipo de algoritmos de aprendizaje. Sin embargo, hay que tener en cuenta que su finalidad se divide en función de los problemas que deben resolver. Principalmente hay dos categorías distintas, a saber, la regresión y la clasificación. En el caso de los problemas de regresión, su objetivo es un valor numérico, mientras que en la clasificación, es una clase o una etiqueta. Para que quede más claro, un ejemplo de tarea de regresión es determinar el valor medio de las casas en una ciudad determinada. Una tarea de clasificación, en cambio, consiste en tomar ciertos datos, como la longitud de los pétalos y de los sépalos, y, a partir de esa información, determinar cuál es la especie de una flor.

Teniendo esto en cuenta, empecemos por hablar de los algoritmos de regresión y de cómo trabajar con ellos.

Regresión

En la ciencia de los datos, muchas tareas se resuelven con la ayuda de técnicas de regresión. Sin embargo, una regresión también puede clasificarse en dos ramas diferentes, que son la regresión lineal y la regresión logística. Cada una de ellas se utiliza para resolver problemas diferentes; sin embargo, ambas son una opción perfecta para los análisis de predicción debido a la gran precisión de los resultados.

El propósito de la regresión lineal es dar forma a un valor de predicción a partir de un conjunto de variables no relacionadas. Esto significa que si

necesitas descubrir la relación entre una serie de variables, puedes aplicar un algoritmo de regresión lineal para que haga el trabajo por ti. Sin embargo, éste no es su uso principal. Los algoritmos de regresión lineal se utilizan para tareas de regresión. Ten en cuenta que la regresión logística no se utiliza para resolver problemas de regresión, como su nombre indica. En cambio, se utiliza para tareas de clasificación.

Dicho esto, vamos a empezar por implementar un algoritmo de regresión lineal en el conjunto de datos de viviendas de Boston, que está disponible de forma gratuita e incluso se incluye en la biblioteca Scikit-learn. Este conjunto de datos contiene 506 muestras, con 13 características y un objetivo de tipo numérico. Vamos a dividirlo en dos secciones, un conjunto de entrenamiento y otro de pruebas. No hay reglas inamovibles en cuanto a la proporción de la división; sin embargo, en general se acepta que es mejor mantener el conjunto de entrenamiento con una distribución de datos del 70% al 80%, y luego guardar el 20% al 30% para el proceso de prueba.

Vecinos más cercanos (K-Nearest Neighbors)

Este algoritmo es uno de los más fáciles de trabajar; sin embargo, puede resolver algunos de los problemas de clasificación más difíciles. El algoritmo de k-próximo más cercano puede utilizarse en varios escenarios que requieren desde la compresión de datos hasta el procesamiento de datos financieros. Es uno de los algoritmos de aprendizaje automático supervisado más utilizados, y debería hacer todo lo posible para practicar su técnica de implementación.

La idea básica del algoritmo es el hecho de explorar la relación entre dos observaciones de entrenamiento diferentes. Por ejemplo, las llamaremos x e y, y si se tiene el valor de entrada de x, ya se puede predecir el valor de y. La forma en que funciona es calculando la distancia de un punto de datos en relación con otros puntos de datos. En función de esta distancia, se selecciona el punto k más cercano y se asigna a una clase concreta.

Para demostrar cómo implementar este algoritmo, vamos a trabajar con un conjunto de datos mucho más grande que antes; sin embargo, no utilizaremos todo lo que contiene. Una vez más, vamos a recurrir a la biblioteca Scikit-learn para acceder a un conjunto de datos conocido como

MNIST handwritten digits dataset. Se trata de una base de datos que contiene aproximadamente 70.000 imágenes de dígitos escritos a mano, que se distribuyen en un conjunto de entrenamiento con 60.000 imágenes y un conjunto de prueba con 10.000 imágenes. Sin embargo, como ya se ha mencionado, no vamos a utilizar todo el conjunto de datos porque eso llevaría demasiado tiempo para esta demostración. En su lugar, nos limitaremos a 1000 muestras. Empecemos:

```
En: from sklearn.utils import shuffle
from sklearn.datasets import
from sklearn.cross_validation import train_test_split
importar pepinillo
mnist = pickle.load(open("mnist.pickle", "rb" ))
mnist.data, mnist.target = shuffle(mnist.data, mnist.target)
```

Como es habitual, primero importamos el conjunto de datos y las herramientas que necesitamos. Sin embargo, notará un paso adicional aquí, a saber, la serialización del objeto. Esto significa que convertimos un objeto a un formato diferente para que pueda ser utilizado más tarde, pero también revertirlo a su versión original si es necesario. Este proceso se conoce como decapado, y es por ello que tenemos importado el módulo pickle, aparentemente fuera de lugar. Esto nos permitirá comunicar objetos a través de una red si es necesario. Ahora, vamos a cortar el conjunto de datos hasta que tengamos sólo 1000 muestras:

```
mnist.data = mnist. data[:1000]
mnist.target = mnist.target[:1000]
X_train, X_test, y_train, y_test = train_test_split(mnist.data,
mnist.target, test_size=0.8, random_state=0)
En: from sklearn.neighbors import KNeighborsClassifier
# KNN: K=10, medida por defecto de la distancia euclidiana
clf = KNeighborsClassifier(3)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Ahora veamos el informe con las métricas de precisión como antes:

```
En: from sklearn.metrics import classification_report
print (classification_report(y_test, y_pred))
```

Y aquí están los resultados:

Fuera:

precisión	retirada	f1-score	soporte	
0.0	0.68	0.90	0.78	79
1.0	0.66	1.00	0.79	95
2.0	0.83	0.50	0.62	76
3.0	0.59	0.64	0.61	85
4.0	0.65	0.56	0.60	75
5.0	0.76	0.55	0.64	80
6.0	0.89	0.69	0.77	70
7.0	0.76	0.83	0.79	76
8.0	0.91	0.56	0.69	77
9.0	0.61	0.75	0.67	87
promedio / total	0.73	0.70	0.70	800

Los resultados no son los mejores; sin embargo, sólo hemos implementado el algoritmo "en bruto" sin realizar ningún tipo de operaciones de preparación que limpien y denoten los datos. Afortunadamente, la velocidad de entrenamiento fue excelente, incluso en este nivel básico. Recuerde que, cuando se trabaja con algoritmos supervisados o con cualquier otro algoritmo, siempre se está cambiando la precisión por la velocidad de procesamiento o viceversa.

Máquinas de vectores de apoyo

La SVM es uno de los algoritmos de aprendizaje supervisado más populares debido a su capacidad para resolver tanto problemas de regresión como de clasificación. Además, tiene la capacidad de identificar los valores atípicos. Se trata de un algoritmo de ciencia de datos completo que no puede perderse. ¿Qué tiene de especial este algoritmo?

En primer lugar, las máquinas de vectores soporte no necesitan mucha potencia de procesamiento para mantener la precisión de la predicción. Este algoritmo, sin embargo, está en una liga propia, y no tendrá que preocuparse demasiado por sacrificar la velocidad de entrenamiento por la precisión o al revés. Además, las máquinas de vectores soporte pueden

utilizarse para eliminar parte del ruido mientras se realizan las tareas de regresión o clasificación.

Este tipo de algoritmo tiene muchas aplicaciones en el mundo real, y por eso es importante que entiendas su implementación. Se utiliza en el software de reconocimiento facial, la clasificación de textos, el software de reconocimiento de escritura a mano, etc. Sin embargo, el concepto básico en el que se basa consiste simplemente en la distancia entre los puntos más cercanos en los que se selecciona un hiperplano a partir del margen entre una serie de vectores de soporte. Hay que tener en cuenta que lo que se conoce como hiperplano aquí es el objeto que divide el espacio de información a efectos de clasificación.

Para poner toda esta teoría en la aplicación, vamos a apoyarnos una vez más en la biblioteca Scikit-learn. El algoritmo se implementará de forma que se demuestre la precisión de la predicción en el caso de la identificación de billetes reales. Ya hemos mencionado que las máquinas de vectores de soporte son eficaces cuando se trata de la clasificación de imágenes. Por lo tanto, este algoritmo se adapta perfectamente a nuestros objetivos. Lo que tenemos que resolver en este ejemplo es un simple problema de clasificación binaria, ya que necesitamos entrenar al algoritmo para que determine si el billete es válido o no.

La factura se describirá mediante varios atributos. Tenga en cuenta que, a diferencia de los demás algoritmos de clasificación, una máquina de vectores de soporte determina su límite de decisión definiendo la distancia máxima entre los puntos de datos más cercanos a las clases relevantes. Sin embargo, no buscamos limitar la decisión, sólo queremos encontrar la mejor. Los puntos más cercanos de esta mejor decisión son lo que denominamos vectores de soporte. Dicho esto, vamos a importar un nuevo conjunto de datos y varias herramientas:

```
importar numpy como np
importar pandas como pd
importar matplotlib.pyplot como plt
dataset = pd.read_csv ("bank_note.csv")
```

Como siempre, el primer paso es aprender más sobre los datos con los que estamos trabajando. Aprendamos cuántas filas y columnas tenemos y luego obtengamos los datos de las cinco primeras filas solamente:

```
print (dataset.shape)
print (dataset.head())
```

Este es el resultado:

	Desviación	Skewness	Curtosis	Entropía	Clase
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.454590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Ahora tenemos que procesar esta información para establecer los conjuntos de entrenamiento y prueba. Esto significa que tenemos que reducir los datos a atributos y etiquetas solamente:

x = dataset.drop ('Clase', eje = 1)

y = conjunto de datos ['Clase']

El propósito de este código es almacenar los datos de la columna como la variable x y luego aplicar la función drop para evitar la columna de la clase para que podamos almacenarla dentro de una variable 'y'. Al reducir el conjunto de datos a una colección de atributos y etiquetas, podemos empezar a definir los conjuntos de datos de entrenamiento y de prueba. Dividamos los datos como hemos hecho en todos los ejemplos anteriores. A continuación, vamos a empezar a implementar el algoritmo.

Necesitamos Scikit-learn para este paso porque contiene el algoritmo de la máquina de vectores de soporte, y por lo tanto podemos acceder fácilmente a él sin requerir fuentes externas.

```
from sklearn.svm import SVC
svc_classifier = SVC (kernel = 'linear')
svc_classifier.fit (x_train, y_train)
pred_y = svc.classifier.predict(x_test)
```

Por último, tenemos que comprobar la precisión de nuestra implementación. Para este paso, vamos a utilizar una matriz de confusión, que actuará como una tabla que muestra los valores de precisión del rendimiento de la clasificación. Veremos un número de verdaderos positivos, verdaderos negativos, así como de falsos positivos y falsos negativos. A partir de estos valores se determina el valor de precisión. Dicho esto, echemos un vistazo a la matriz de confusión y luego imprimamos el informe de clasificación:

```
from sklearn.metric import confusion_matrix
print (matriz_de_confusión (y_test, pred_y)
```

Este es el resultado:
[[160 1]
[1 113]]
Puntuación de precisión: 0,99
Ahora veamos el conocido informe de clasificación:
from sklearn.metrics import classification_report
print (classification_report(y_test, y_pred))
Y aquí están los resultados del informe:

precisión	Recall	f1-score	Soporte	
0.0	0.99	0.99	0.99	161
1.0	0.99	0.99	0.99	114
promedio / total	0.99	0.99	0.99	275

Basándonos en todas estas métricas, podemos determinar que hemos obtenido una precisión muy alta con nuestra implementación de las máquinas de vectores de soporte. Una puntuación de 0,99 es casi lo mejor que se puede conseguir; sin embargo, siempre se puede mejorar.

Capítulo 7: Agregación de datos y operaciones de grupo

Esto representa la primera parte de la agregación y la agrupación utilizando Pharo DataFrame. Esto sólo manejará la funcionalidad básica como la agrupación de una serie de datos utilizando los valores de una serie separada del tamaño correspondiente y el uso de funciones de agregación a las estructuras de datos agrupados.

Las próximas iteraciones se ocuparán de la funcionalidad ampliada en función de los escenarios previstos. Es probable que la implementación cambie a algo optimizado.

Definición de marco de datos

Esto representa hojas de cálculo como estructuras de datos que ofrecen una API para limpiar, cortar y analizar datos.

En caso de que quiera leer más sobre el proyecto DataFrame, debe tener en cuenta la documentación.

Dividir-Aplicar-Combinar

La división-aplicación-combinación es una técnica en la que se clasifica una determinada tarea en partes manejables y luego se integran todas las partes.

La agregación y agrupación de datos facilita la elaboración de resúmenes para su análisis y visualización. Por ejemplo, cuando se calculan los valores medios o se crea una tabla de recuentos. Este es un paso que se adhiere al procedimiento de dividir-aplicar-combinar.

1. Separe los datos en secciones según un procedimiento determinado.
2. Utiliza la función para cada clúster de forma independiente.
3. Combinar los resultados mediante una estructura de datos.

Aplicación

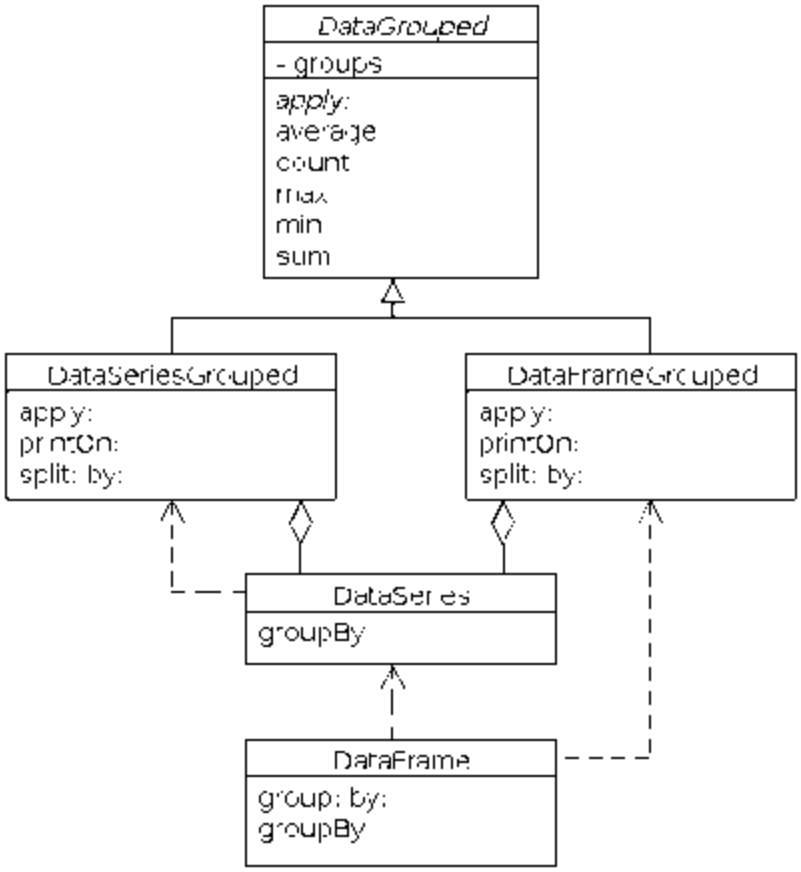
En esta parte, descubrirá cómo se implementa la función de agrupación y agregación. En caso de que no quiera estos detalles, puede pasar a la siguiente parte.

Tomemos, por ejemplo, este mensaje que se ha enviado al objeto firstSeries:

```
firstSeries groupBy: secondSeries.
```

Una vez enviado este mensaje, firstSeries definirá un objeto de DataSeriesGrouped, que divide firstSeries en varias subseries dependiendo de los valores de secondSeries.

La colección de subseries se guarda posteriormente como un objeto de DataSeries cuyas claves equivalen a los valores especiales de secondSeries y los valores almacenan las subseries de firstSeries. De este modo, coincidirá con cada uno de esos valores especiales.



Esto significa que los grupos representan DataSeries que contienen claves que coinciden con los valores únicos contenidos en una cadena en la que se reconoce el marco de datos. Cuando el marco de datos está categorizado por una sola columna, esa columna se elimina del marco de datos antes de la agrupación. Por lo tanto, esto elimina la duplicación de datos porque los mismos valores se conservarán como claves.

En el caso de DataSeriesGrouped, cada subsistema se conectará a un escalar, y todos los escalares posteriores se fusionarán en un DataSeries. En el caso de DataFrameGrouped, incluirá el bloque en cada columna de cada cuadro de subdirectorio y mostrará la tabla escalar final como el nuevo DataFrame.

La combinación se realiza con el uso de mensajes. Requiere un bloque como argumento y lo utiliza en cada valor de la cadena del grupo y luego lo integra en una nueva estructura de datos.

Las funciones de agregación más comunes, como la media, el mínimo y el máximo, proporcionan mensajes más pequeños. En la siguiente iteración, estos mensajes son útiles y actúan como atajos.

```
average
   self apply: [ :each   each average ].
```

Sin embargo, estos mensajes llevarán las implementaciones optimizadas de las agregaciones comparadas porque es necesario que estas funciones sean eficientes en tiempo y memoria.

Examinemos las series de agrupación.

El ejemplo más sencillo de utilizar este operador groupBy es clasificar los valores de una serie utilizando valores del mismo tamaño.

```
bill :  tips column:
sex :  tips column:
```

El resultado de la consulta anterior será un objeto. Este objeto separará la factura en dos series.

Porque muchas veces es necesario clasificar las series de grupos que se asemejan a las columnas de un mismo marco de datos. Hay un atajo útil.

¿Cómo se agrupan los marcos de datos?

Además del método abreviado para clasificar las columnas. El DataFrame tiene un método para clasificar una de sus columnas.

La respuesta de la consulta anterior será un objeto de DataFrameGrouped, manteniendo dos marcos de datos diferentes para fumadores y no fumadores.

La columna del fumador se eliminará de los marcos de datos anteriores porque sus valores se mantendrán como claves dentro de un objeto DataFrameGrouped. Además, los diferentes grupos de fumadores y no fumadores permitirán la reconstrucción completa de la columna de fumadores cuando sea necesario.

Las funciones de agregación representan las que aceptan diferentes entradas y muestran un valor escalar que suma los valores de esa serie en particular. Se trata de funciones estadísticas como min, max, stdev y muchas más.

Una vez combinados los datos, se puede utilizar la función de agregación para obtener la estructura de datos integrada que suma los datos originales.

```
grouped := tips group: #total_bill by: #day.
grouped apply: [ :each | each average round: 2].
```

Dado que la agrupación se está haciendo a una columna de DataFrame por una columna separada. El resultado será un objeto DataSeries.

Como se ha dicho antes, el DataGrouped presenta atajos para las funciones de agregación popularmente aplicadas como count, sum, min y max. Por el momento, se trata de atajos, pero en el futuro se ejecutarán las agregaciones optimizadas que se utilizarán más rápidamente.

Una vez que el cuadro de datos ha sido agrupado en un objeto DataFrameGrouped, también podemos aplicar una función de agregación a ese objeto. DataFrameGrouped implementa el mensaje de aplicación: para que la función se aplique a cada columna de cada cuadro de datos hijo, produciendo el valor incremental. Estos pasos se combinan en un nuevo marco de datos.

El resultado de esta consulta será un cuadro de datos que contendrá el número de celdas vacías de cada columna, correspondientes a las filas "Hombre" y "Mujer".

	total_bill	tip	smoker	day	time	size
Female	87	87	87	87	87	87
Male	157	157	157	157	157	157

Capítulo 8: Códigos y ejercicios prácticos para utilizar Python

Haremos algunos ejercicios diferentes de Python aquí para que puedas divertirte un poco y tener una mejor idea de cómo usar los diferentes temas de los que hemos hablado en esta guía para tu beneficio. Hay un montón de programas que puedes usar cuando escribes en Python, pero este te dará una buena idea de cómo escribir códigos y cómo usar los ejemplos de los que hemos hablado en esta guía en la codificación real.

Creación de una bola 8 mágica

El primer proyecto que vamos a ver aquí es cómo crear tu propia bola 8 mágica. Esto funcionará como una bola 8 mágica normal, pero estará en el ordenador. Puedes elegir cuántas respuestas quieres que estén disponibles para los que usen el programa, pero vamos a centrarnos en que aparezcan ocho respuestas para el usuario en un orden aleatorio, para que obtengan algo diferente cada vez.

Configurar este código es más fácil de lo que crees. Tómate un tiempo para estudiar este código y luego escríbelo en el compilador. El código que necesitas usar para crear un programa que incluya tu propia bola 8 mágica incluirá:

```
# Importar los módulos
importar sys
Importar al azar
ans = Verdadero
mientras que ans:
question = raw_input("Haz una pregunta a la bola 8 mágica: (pulsa intro para salir)")
respuestas = random.randint(1,8)
si pregunta == ""
sys.exit()
elif respuestas ==1:
print("Es cierto")
elif respuestas == 2:
```

```
print("Outlook good")
elif respuestas == 3:
print("Puede confiar en ello")
elif answers == 4:
print("Vuelve a preguntar más tarde"
)
elif respuestas == 5:
print("Concéntrate y vuelve a preguntar")
elif answers == 6:
print("Respuesta confusa, inténtalo de nuevo.")
elif answers == 7:
print("Mi respuesta es no")
elif respuestas == 8:
print("Mis fuentes dicen que no")
```

Recuerde, en este programa, elegimos ir con ocho opciones porque es una bola 8 mágica, y eso tiene el mayor sentido. Pero si quiere añadir más opciones, o trabajar en otro programa que sea similar y tenga más opciones, entonces sólo tendrá que seguir añadiendo más de la sentencia elif para conseguirlo. Este es un buen ejemplo de cómo usar la sentencia elif de la que hablamos antes y puede darnos algo de práctica sobre cómo usarla. También puedes experimentar un poco con el programa para ver qué tan bien funciona y hacer los cambios que creas necesarios para ayudarte a obtener los mejores resultados.

Cómo hacer un juego del ahorcado

El siguiente proyecto que vamos a ver es la creación de tu propio juego del ahorcado. Este es un gran juego para crear porque tiene muchas de las diferentes opciones de las que hemos hablado a lo largo de esta guía y puede ser una gran manera de conseguir algo de práctica en los diversos temas que hemos visto. Vamos a ver cosas como un bucle presente, algunos comentarios, y más, y esta es una buena manera de trabajar con algunas de las declaraciones condicionales que aparecen también.

Ahora, usted puede estar mirando este tema y pensar que va a ser difícil trabajar con un juego del ahorcado. Va a tener un montón de partes que van juntas, ya que la persona hace una conjetura y el programa trata de averiguar lo que está pasando, si las conjeturas son correctas, y cuántas

oportunidades tiene el usuario para hacer estas conjeturas. Pero usar muchas de las diferentes partes de las que ya hemos hablado en esta guía puede ayudarnos a escribir este código sin ningún problema. El código que necesitas usar para crear tu propio juego del ahorcado en Python incluye:

```
# importando el módulo de tiempo
tiempo de importación
#acogida del usuario
Nombre = raw_input("¿Cuál es su nombre?")
print("Hola, + nombre, "¡Hora de jugar al ahorcado!")
imprimir("
"
#espera 1 segundo
time.sleep(1)
print("Empieza a adivinar...")
time.sleep(.05)
#aquí ponemos el secreto
palabra = "secreto"
#crea una variable con un valor vacío
conjeturas = ''
#determinar el número de vueltas
vueltas = 10
#crear un bucle while
#comprobar si las vueltas son más que cero
mientras que los giros > 0:
#hacer un contador que empiece por cero
fallido = 0
#para cada carácter de la palabra secreta
para el coche en la palabra:
#ver si el personaje está en la suposición de los jugadores
si char en suposiciones:
#imprime entonces el carácter
imprime el carácter,
si no
# si no se encuentra, imprime un guión
imprimir "_",
# y aumentar el contador de fallos con uno
fallido += 1
```

```
#si el fallo es igual a cero
#print You Won
si falla == 0:
print("Has ganado")
#salir de la secuencia de comandos
Descanso
imprimir
# pide al usuario que adivine un carácter
guess = raw_input("adivinar un carácter:")
#poner a los jugadores a adivinar
conjeturas += conjetura
# si la conjetura no se encuentra en la palabra secreta
si supongo que no en word:
#El contador de vueltas disminuye con 1 (ahora 9)
gira -= 1
#impresión errónea
print("Equivocado")
# cuántas vueltas quedan
Print("Tienes," + giros, 'más conjeturas')
#si las vueltas son iguales a cero
si los giros == 0
#print "You Lose"
```

Este es un pedazo de código más largo, especialmente cuando se compara con la Bola 8 Mágica que hicimos anteriormente, pero respira profundamente, y revisa todo para ver lo que reconoces que hay. Esto no es tan malo como parece, y gran parte de él son comentarios para ayudarnos a ver lo que está pasando en algunas de las diferentes partes del código. Esto hace que sea más fácil de usar para nuestras propias necesidades y puede asegurar que sabemos lo que está pasando en las diferentes partes.

Cómo crear su propio algoritmo K-Means

Ahora que hemos tenido algo de tiempo para ver algunos juegos y ejemplos divertidos que puedes hacer con la ayuda del código de Python, vamos a tomar un momento para ver algunas de las cosas que puedes hacer con Machine Learning e inteligencia artificial con tu codificación. Hemos pasado algún tiempo hablando de cómo se puede trabajar con estos y

algunas de las diferentes partes del código, así como la forma en que Python va a trabajar con la idea de Machine Learning. Y ahora, vamos a tomar esa información y crear uno de nuestros algoritmos de Machine Learning para trabajar también.

Antes de trabajar en un código para esto, necesitamos echar un vistazo a lo que significa este clustering de k-means. Este es un algoritmo básico que funciona bien con el aprendizaje automático y va a ayudarle a reunir todos los datos que tiene en su sistema, los datos que no están etiquetados en el momento, y luego los pone todos juntos en su pequeño grupo de un clúster.

La idea de trabajar con este tipo de cluster es que los objetos que caen dentro del mismo cluster, ya sea que haya sólo dos o más, van a estar relacionados entre sí de alguna manera u otra, y no van a ser tan similares a los puntos de datos que caen en los otros clusters. La similitud aquí va a ser la métrica que querrá utilizar para mostrarnos la fuerza que hay en la relación entre los dos.

Cuando se trabaja en este algoritmo en particular, va a ser capaz de formar algunos de los clusters que necesita de los datos, sobre la base de la similitud de los valores de los datos que tiene. Tendrás que ir y darle un valor específico para K, que será el número de clusters que te gustaría usar. Lo mejor es tener al menos dos, pero el número de estos clusters con los que se trabaje dependerá de la cantidad de datos que se tengan y de cuántos se ajusten al tipo de datos con los que se esté trabajando.

Con esta información en mente y un buen antecedente de para qué se va a utilizar el algoritmo K-means, es hora de explorar un poco más sobre cómo escribir sus propios códigos y hacer un ejemplo que funcione con K-means. Esto nos ayuda a practicar un poco con el Aprendizaje Automático y nos da la oportunidad de practicar algunas de nuestras nuevas habilidades en Python.

```
importar numpy como np
importar matplotlib.pyplot como plt
def d(u, v):
diff = u - v
return diff.dot(diff)
def coste(X, R, M):
coste = 0
para k en xrange(len(M)):
para n en xrange(len(X)):
```

coste += R[n,k]*d(M[k], X[n])
coste de retorno

Después de esta parte, vamos a tomar el tiempo para definir su función para que sea capaz de ejecutar el algoritmo de k-means antes de trazar el resultado. Esto va a terminar con un gráfico de dispersión donde el color representará la cantidad de miembros que están dentro de un clúster en particular. Lo haríamos con el siguiente código:

```
def graficar_k_medias(X, K, max_iter=20, beta=1.0):
N, D = X.shape
M = np.zeros((K, D))
R = np.ones((N, K)) / K
# inicializar M al azar
para k en xrange(K):
M[k] = X[np.random.choice(N)]
ancho_de_rejilla = 5
altura_de_la_rejilla = max_iter / anchura_de_la_rejilla
random_colors = np.random.random((K, 3))
plt.figure()
costes = np.zeros(max_iter)
para i en xrange(max_iter):
# movido la trama dentro del bucle for
colores = R.dot(colores_aleatorios)
plt.subplot(ancho_cuadrícula, alto_cuadrícula, i+1)
plt.scatter(X[:,0], X[:,1], c=colors)
# paso 1: determinar las asignaciones / responsabilidades
# ¿es esto ineficiente?
para k en xrange(K):
para n en xrange(N):
R[n,k] = np.exp(-beta*d(M[k], X[n])) / np.sum( np.exp(-beta*d(M[j], X[n])) para j en xrange(K) )
# paso 2: recalcular las medias
para k en xrange(K):
M[k] = R[:,k].dot(X) / R[:,k].sum()
costes[i] = coste(X, R, M)
si i > 0:
si np.abs(costes[i] - costes[i-1]) < 10e-5:
romper
```

```
plt.show()
```

Observe aquí que tanto la M como la R van a ser matrices. La R se va a convertir en la matriz porque mantiene 2 índices, el k y el n. M también es una matriz porque va a contener los K vectores individuales de dimensión D. La variable beta va a controlar el grado de dispersión de los miembros de los clusters y se conocerá como hiperparámetro. A partir de aquí, vamos a crear una función principal que creará clusters aleatorios y luego llamará a las funciones que ya hemos definido anteriormente.

```
def main():
# Asumir que 3 significa
D = 2 # para que podamos visualizarlo más fácilmente
s = 4 # separación para poder controlar la distancia entre las medias
mu1 = np.array([0, 0])
mu2 = np.array([s, s])
mu3 = np.array([0, s])
N = 900 # número de muestras
X = np.zeros((N, D))
X[:300, :] = np.random.randn(300, D) + mu1
X[300:600, :] = np.random.randn(300, D) + mu2
X[600:, :] = np.random.randn(300, D) + mu3
# ¿Qué aspecto tiene sin agrupación?
plt.scatter(X[:,0], X[:,1])
plt.show()
K = 3 # por suerte, ya lo sabemos
plot_k_means(X, K)
# K = 5 # ¿Qué pasa si elegimos una K "mala"?
# plot_k_means(X, K, max_iter=30)
# K = 5 # ¿Qué pasa si cambiamos beta?
# plot_k_means(X, K, max_iter=30, beta=0.3)
si __name__ == '__main__':
main()
```

Sí, este proceso va a tomar algún tiempo para escribir aquí, y no siempre es un proceso fácil cuando se trata de trabajar a través de las diferentes partes que vienen con Machine Learning y cómo puede afectar a su código. Pero cuando haya terminado, usted será capaz de importar algunos de los datos que su empresa ha estado recogiendo, y luego determinar cómo se compara utilizando el algoritmo de K-means también.

Capítulo 9: Funciones y módulos en Python

Cuando trabajas con un lenguaje como Python, habrá ocasiones en las que necesitarás trabajar con algo que se conoce como función. Estas funciones van a ser bloques de código reutilizable que utilizarás para realizar tus tareas específicas. Pero cuando defines una de estas funciones en Python, necesitamos tener una buena idea de los dos tipos principales de funciones que se pueden utilizar, y cómo funciona cada una de ellas. Los dos tipos de funciones que están disponibles aquí se conocen como incorporadas y definidas por el usuario.

Las funciones incorporadas son las que vendrán automáticamente con algunos de los paquetes y bibliotecas que están disponibles en Python. Sin embargo, vamos a dedicar nuestro tiempo a trabajar con las funciones definidas por el usuario porque son las que el desarrollador creará y utilizará para los códigos especiales que escriba. En Python, sin embargo, una cosa que hay que recordar, no importa el tipo de función con la que estés trabajando, es que todas ellas serán tratadas como objetos. Esto es una buena noticia porque puede hacer que sea mucho más fácil trabajar con estas funciones en comparación con otros lenguajes de codificación.

Built-in Functions

abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

Las funciones definidas por el usuario, de las que hablaremos en la siguiente sección, serán importantes y pueden ampliar parte del trabajo que estamos haciendo. También tenemos que echar un vistazo a algunos de los trabajos que somos capaces de hacer con las funciones incorporadas. La

lista anterior incluye muchas de las que se encuentran en el lenguaje Python. Tómate un tiempo para estudiarlos y ver lo que son capaces de hacer para ayudar a hacer las cosas.

¿Por qué son tan importantes las funciones definidas por el usuario?

Para mantenerlo simple, un desarrollador va a tener la opción de escribir algunas de sus propias funciones, conocidas como funciones definidas por el usuario, o puede ir y tomar prestada una función de otra biblioteca, una que puede no estar directamente asociada con Python. Estas funciones a veces nos van a proporcionar algunas ventajas, dependiendo de cómo y cuándo queramos utilizarlas en el código. Algunas de las cosas que debemos recordar cuando trabajemos con estas funciones definidas por el usuario, para comprender mejor cómo funcionan, serán:

Estas funciones van a estar formadas por bloques de código que son reutilizables. Es necesario escribirlas sólo una vez. Luego puedes utilizarlas tantas veces como necesites en el código. Incluso puede hacer uso de las funciones definidas por el usuario en algunas de sus otras aplicaciones también.

Estas funciones también pueden ser muy útiles. Puedes usarlas para ayudarte con cualquier cosa que quieras, desde escribir una lógica específica en los negocios hasta trabajar en utilidades comunes. También puedes modificarlas en función de tus propias necesidades, para que el programa funcione correctamente.

El código suele ser amigable para los desarrolladores, fácil de mantener y bien organizado. Esto significa que es capaz de apoyar el enfoque para el diseño modular.

Puede escribir este tipo de funciones de forma independiente. Además, las tareas de su proyecto se pueden distribuir para un rápido desarrollo de la aplicación, si es necesario.

Una función definida por el usuario que esté bien pensada puede facilitar el proceso de desarrollo de una aplicación.

Ahora que sabemos un poco más sobre los fundamentos de una función definida por el usuario, es el momento de ver algunos de los diferentes argumentos que pueden venir con estas funciones antes de pasar a algunos de los códigos que se pueden utilizar con este tipo de función.

Opciones para los argumentos de las funciones

Cada vez que estés listo para trabajar con este tipo de funciones en tu código, encontrarás que tienen la capacidad de trabajar con cuatro tipos de argumentos. Estos argumentos y los significados detrás de ellos serán predefinidos, y el desarrollador no siempre va a ser capaz de cambiarlos. En su lugar, el desarrollador va a tener la opción de utilizarlos, pero también seguir las reglas que hay con ellos. Usted tiene la opción de añadir un poco a las reglas para hacer que las funciones trabajen de la manera que usted desea. Como dijimos antes, hay cuatro tipos de argumentos con los que puedes trabajar, y estos incluyen:

- **Argumentos por defecto**: En Python, vamos a encontrar que hay una manera diferente de representar los valores por defecto y la sintaxis para los argumentos de sus funciones. Estos valores por defecto van a indicar que el argumento de la función va a tomar ese valor si no tienes un valor para el argumento que pueda pasar por la llamada de la función. La mejor manera de averiguar dónde estará el valor por defecto es buscar el signo de igualdad.
- **Argumento obligatorio**: Estos son los tipos de argumentos que serán obligatorios para la función en la que se está trabajando. Estos valores necesitan pasar y ser pasados en el orden y número correcto, ya sea cuando la función es llamada o cuando el código no podrá ejecutarse de la manera correcta.
- **Argumentos de la palabra clave**: Estos van a ser los argumentos que van a poder ayudar con la llamada a la función dentro de Python. Estas palabras clave van a ser las que mencionemos a través de la llamada a la función, junto con algunos de los valores que pasarán por ésta. Estas palabras clave serán mapeadas con el argumento de la función para que seas capaz de identificar todos los valores, incluso si no mantienes el mismo orden cuando el código es llamado.
- **Argumentos variables**: El último argumento que vamos a ver aquí es el número variable de argumentos. Esta es una buena para trabajar cuando no estás seguro de cuántos argumentos van a ser necesarios para el código que estás escribiendo para pasar la función. De lo contrario, usted puede utilizar esto para diseñar su código donde cualquier número de argumentos se puede pasar, siempre y cuando han sido capaces de pasar cualquier requisito en el código que se establece.

Escribir una función

Ahora que tenemos una idea un poco mejor de cómo son estas funciones, y algunos de los tipos de argumentos que están disponibles en Python, es hora de que aprendamos los pasos que se necesitan para lograr todo esto. Habrá cuatro pasos básicos que podemos utilizar para hacer que todo esto suceda, y es realmente hasta el programador de lo difícil o simple que le gustaría que esto sea. Empezaremos con algunos de los básicos, y luego se puede ir a través y hacer algunos ajustes según sea necesario. Algunos de los pasos que necesitamos tomar para escribir nuestras propias funciones definidas por el usuario se dan a continuación.

- Declare su función. Tendrás que usar la palabra clave 'def', y luego hacer que el nombre de la función venga justo después de ella.
- Escriba los argumentos. Estos deben estar dentro de los dos paréntesis de la función. Termina esta declaración con dos puntos, para seguir el protocolo de escritura adecuado en este lenguaje.
- Añade las sentencias que el programa debe ejecutar en este momento.
- Terminar la función. Puedes elegir si quieres hacerlo con una sentencia return.

Un ejemplo de la sintaxis que usarías cuando quieres hacer una de tus propias funciones definidas por el usuario es:

```
def userDefFunction (arg1, arg2, arg3, ...):
declaración del programa1
declaración del programa2
declaración del programa3
....
Vuelve;
```

Trabajar con funciones puede ser una gran manera de asegurar que tu código se va a comportar de la manera que te gustaría. Asegurarse de que lo configuras de la manera adecuada también puede ser muy importante. Hay muchas veces en las que las funciones saldrán y servirán para algún propósito, así que tomarse el tiempo ahora para aprender a usarlas puede ser muy importante para el éxito de tu código.

Módulos de Python

Los módulos se componen de definiciones y sentencias de programa.

Un ejemplo es un archivo llamado config.py, que se considera un módulo. El nombre del módulo sería config. Los módulos se utilizan para ayudar a dividir los programas grandes en archivos más pequeños, manejables y organizados, así como para promover la reutilización del código.

Ejemplo:

Creación del primer módulo

Inicie el IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

Def add(x, y):(Este es un programa para sumar dos números y devolver el resultado)

Resultado: x+y

Resultado de la devolución

Módulo de importación

La palabra clave "importar" se utiliza para importar.

Ejemplo:

Importar primero

El operador punto puede ayudarnos a acceder a una función siempre que conozcamos el nombre del módulo.

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

first.add(6,8)

Declaración de importación en Python

La sentencia import puede utilizarse para acceder a las definiciones de un módulo mediante el operador punto.

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

importar matemáticas

print("El valor PI es", math.pi)

Importación con cambio de nombre

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

importar math como h

print("El valor de PI es-",h.pi)

En este caso, h es nuestro módulo matemático renombrado que ahorra tiempo de escritura en algunos casos. Cuando renombramos, el nuevo nombre se convierte en el válido y reconocido en lugar del original.

De... declaración de importación Python

Es posible importar nombres particulares de un módulo, en lugar de importar todo el módulo.

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

from math import pi

print("El valor de PI es-", pi)

Importar todos los nombres

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

de math import*

print("El valor de PI es-", pi)

En este contexto, estamos importando todas las definiciones de un módulo concreto. Sin embargo, es una norma alentada, ya que puede dar lugar a duplicados no vistos.

Ruta de búsqueda de módulos en Python

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

importar sys

sys.path

Python busca en todas partes, incluido el archivo .sys.

Recarga de un módulo

Python sólo importará un módulo una vez, aumentando la eficiencia en la ejecución.

print("Este programa fue ejecutado") import mine

Código de recarga

Ejemplo:

Iniciar IDLE.

Vaya al menú Archivo y haga clic en Nueva ventana.

Escribe lo siguiente:

mina de importación

mina de importación

mina de importación

mine.reload(mine)

Función integrada de Python Dir()

Para descubrir los nombres contenidos en un módulo, utilizamos la función incorporada dir().

Sintaxis

dir(nombre_del_módulo)

Paquete Python

Los archivos en Python contienen módulos, y los directorios se almacenan en paquetes. Un solo paquete en Python contiene módulos similares. Por lo tanto, los diferentes módulos deben colocarse en diferentes paquetes de Python.

Capítulo 10: La ciencia de los datos y la nube

La ciencia de los datos es una mezcla de muchos conceptos. Para convertirse en un científico de datos, es importante tener algunos conocimientos de programación. Aunque no conozcas todos los conceptos de programación relacionados con la infraestructura, pero tener conocimientos básicos de conceptos de informática es imprescindible. Debes instalar en tu ordenador los dos lenguajes de programación más comunes y más utilizados, es decir, R y Python. Con la analítica avanzada en constante expansión, la Ciencia de los Datos sigue extendiendo sus alas en diferentes direcciones. Esto requiere soluciones de colaboración como el análisis predictivo y los sistemas de recomendación. Las soluciones de colaboración incluyen herramientas de investigación y cuadernos integrados con el control de código fuente. La ciencia de los datos también está relacionada con la nube. La información también se almacena en la nube. Así pues, esta lección le ilustrará con algunos datos sobre los "datos en la nube". Así que vamos a entender qué significa la nube y cómo se almacenan los datos y cómo funciona.

¿Qué es la nube?

La nube puede describirse como una red global de servidores, cada uno de los cuales tiene diferentes funciones únicas. Para estudiar la nube es necesario entender las redes. Las redes pueden ser agrupaciones simples o complejas de información o datos.

Red

Como se ha especificado anteriormente, las redes pueden tener un simple o pequeño grupo de ordenadores conectados o grandes grupos de ordenadores conectados. La red más grande puede ser Internet. Los grupos pequeños pueden ser redes locales domésticas como el Wi-Fi y la red de área local que está limitada a ciertos ordenadores o localidad. Hay redes compartidas

como las de medios de comunicación, páginas web, servidores de aplicaciones, almacenamiento de datos e impresoras y escáneres. Las redes tienen nodos, donde un ordenador se denomina nodo. La comunicación entre estos ordenadores se establece mediante el uso de protocolos. Los protocolos son las reglas intermedias establecidas para un ordenador. Protocolos como HTTP, TCP e IP se utilizan a gran escala. Toda la información se almacena en el ordenador, pero resulta difícil buscar la información en el ordenador cada vez. Esta información suele almacenarse en un Centro de Datos. El Centro de Datos está diseñado de tal manera que está equipado con seguridad de apoyo y protección para los datos. Dado que el coste de los ordenadores y del almacenamiento ha disminuido sustancialmente, muchas organizaciones optan por hacer uso de varios ordenadores que funcionan juntos y que se quieren escalar. Esto difiere de otras soluciones de escalado como la compra de otros dispositivos informáticos. La intención detrás de esto es mantener el trabajo de forma continua incluso si un ordenador falla; el otro continuará la operación. También es necesario escalar algunas aplicaciones en la nube. Echando un vistazo a algunas aplicaciones informáticas como YouTube, Netflix y Facebook que requieren cierto escalado. Rara vez experimentamos que estas aplicaciones fallen, ya que han montado sus sistemas en la nube. Hay un clúster de red en la nube, donde muchos ordenadores están conectados a las mismas redes y realizan tareas similares. Se puede denominar como una única fuente de información o un único ordenador que gestiona todo para mejorar el rendimiento, la escalabilidad y la disponibilidad.

Ciencia de datos en la nube

Todo el proceso de la ciencia de datos tiene lugar en la máquina local, es decir, en un ordenador o portátil proporcionado al científico de datos. El ordenador o el portátil tienen lenguajes de programación incorporados y algunos requisitos previos más instalados. Esto puede incluir lenguajes de programación comunes y algunos algoritmos. Posteriormente, el científico de datos tiene que instalar el software y los paquetes de desarrollo pertinentes según su proyecto. Los paquetes de desarrollo pueden instalarse mediante gestores como Anaconda o similares. También se puede optar por instalarlos manualmente. Una vez que se instalan y se entra en el entorno de desarrollo, comienza el primer paso, es decir, el flujo de trabajo, en el que

su acompañante son sólo los datos. No es obligatorio realizar la tarea relacionada con Data Science o Big data en diferentes máquinas de desarrollo. Comprueba las razones que hay detrás de esto:

1. El tiempo de procesamiento necesario para llevar a cabo las tareas en el entorno de desarrollo falla debido a un fallo en la potencia de procesamiento.
2. Presencia de grandes conjuntos de datos que no pueden ser contenidos en la memoria del sistema del entorno de desarrollo.
3. Los resultados deben ser dispuestos en un entorno de producción e incorporados como un componente en una gran aplicación.
4. Se aconseja utilizar una máquina que sea rápida y potente.

Los científicos de datos exploran muchas opciones cuando se enfrentan a estos problemas; hacen uso de máquinas locales o de máquinas virtuales que se ejecutan en la nube. El uso de máquinas virtuales y clústeres de autoescalado tiene varias ventajas, como que pueden ampliarse y descartarse en cualquier momento en caso de que sea necesario. Las máquinas virtuales se personalizan de forma que satisfagan las necesidades de potencia de cálculo y almacenamiento de cada uno. El despliegue de la información en un entorno de producción para empujarla en una gran tubería de datos puede tener ciertos desafíos. Estos desafíos deben ser entendidos y analizados por el científico de datos. Esto se puede entender teniendo una idea de las arquitecturas de software y los atributos de calidad.

Arquitectura de software y atributos de calidad

Los arquitectos de software desarrollan un sistema de software basado en la nube. Estos sistemas pueden ser un producto o un servicio que depende del sistema informático. Si se está construyendo un software, la tarea principal incluye la selección del lenguaje de programación adecuado que se va a programar. Se puede cuestionar el propósito del sistema, por lo que hay que tenerlo en cuenta. El desarrollo y el trabajo con la arquitectura del software deben ser realizados por una persona altamente cualificada. La mayoría de las organizaciones han comenzado a implementar un entorno de nube eficaz y fiable utilizando la computación en nube. Estos entornos de nube se

despliegan en varios servidores, almacenamiento y recursos de red. Esto se utiliza en abundancia debido a su menor coste y a su alto retorno de la inversión.

El principal beneficio para los científicos de datos o sus equipos es que utilizan el gran espacio de la nube para explorar más datos y crear casos de uso importantes. Se puede lanzar una función y hacer que se pruebe al segundo siguiente y comprobar si añade valor o no es útil para llevarla adelante. Toda esta acción inmediata es posible gracias a la computación en nube.

Compartir Big Data en la nube

El papel del Big Data también es vital cuando se trata de la nube, ya que facilita el seguimiento y el análisis de los datos. Una vez establecido esto, el big data crea un gran valor para los usuarios.

La forma tradicional era procesar los datos por cable. Esta técnica dificultaba el intercambio de información entre el equipo. Los problemas habituales eran la transferencia de grandes cantidades de datos y la colaboración de los mismos. Aquí es donde la computación en nube empezó a sembrar su semilla en el mundo competitivo. Todos estos problemas se eliminaron gracias a la computación en la nube y, poco a poco, los equipos pudieron trabajar juntos desde diferentes lugares y también en el extranjero. Por lo tanto, la computación en la nube es muy vital tanto en la Ciencia de los Datos como en el Big data. La mayoría de las organizaciones hacen uso de la nube. Por ejemplo, algunas empresas que utilizan la nube son Swiggy, Uber, Airbnb, etc. Utilizan la computación en la nube para compartir información y datos.

Gobernanza de la nube y los grandes datos

Trabajar con la nube es una gran experiencia, ya que reduce el coste de los recursos, el tiempo y los esfuerzos manuales. Pero se plantea la cuestión de cómo las organizaciones se ocupan de la seguridad, el cumplimiento y la gobernanza. La regulación de los mismos es un reto para la mayoría de las empresas. No se limita a los problemas de Big data, sino que trabajar con la nube también tiene sus problemas relacionados con la privacidad y la seguridad. Por lo tanto, es necesario desarrollar una fuerte política de gobernanza en sus soluciones en la nube. Para asegurarse de que sus soluciones en la nube son fiables, robustas y gobernables, debe mantenerla como una arquitectura abierta.

Necesidad de herramientas de nube de datos para ofrecer un alto valor de los datos

La demanda de un científico de datos en esta época está aumentando rápidamente. Se encargan de ayudar a las organizaciones grandes y pequeñas a desarrollar información útil a partir de los datos o del conjunto de datos que se les proporciona. Las grandes organizaciones tienen datos masivos que necesitan ser analizados continuamente. Según informes recientes, casi el 80% de los datos no estructurados que reciben las organizaciones están en forma de redes sociales, correos electrónicos, es decir, Outlook, Gmail, etc., vídeos, imágenes, etc. Con el rápido crecimiento de la computación en la nube, los científicos de datos lidian con varias cargas de trabajo nuevas que provienen de IoT, AI, Blockchain, Analytics, etc. Pipeline. Trabajar con todas estas nuevas cargas de trabajo requiere una plataforma estable, eficiente y centralizada en todos los equipos. Con todo esto, existe la necesidad de gestionar y registrar los nuevos datos, así como los documentos heredados. Una vez que a un científico de datos se le asigna una tarea, y tiene el conjunto de datos para trabajar, debe poseer las habilidades adecuadas para analizar los volúmenes cada vez mayores a través de las tecnologías de la nube. Tienen que convertir los datos en ideas útiles que sean responsables de elevar el negocio. El científico de datos tiene que construir un algoritmo y codificar

el programa. La mayoría de las veces utilizan el 80% de su tiempo para recopilar información, crear y modificar datos, limpiarlos si es necesario y organizarlos. El 20% restante se utiliza para analizar los datos con una programación eficaz. Esto exige la necesidad de contar con herramientas específicas en la nube que ayuden a los científicos de datos a reducir su tiempo de búsqueda de la información adecuada. Las organizaciones deberían poner a disposición de sus respectivos científicos de datos nuevos servicios y herramientas en la nube para que puedan organizar los datos masivos rápidamente. Por lo tanto, las herramientas en la nube son muy importantes para que un científico de datos pueda analizar grandes cantidades de datos en un período más corto. Ahorrará tiempo a la empresa y ayudará a construir modelos de datos sólidos y robustos.

Capítulo 11: Minería de datos

La minería de datos es un proceso elaborado en el que el analista busca e identifica correlaciones, patrones y anomalías dentro de un conjunto de datos determinado. A partir de su investigación, los resultados se utilizan para el análisis predictivo. Hay varias técnicas que pueden utilizarse para interpretar la información obtenida mediante la minería de datos. En muchos entornos, los informes de minería de datos ayudan a tomar decisiones importantes sobre los ingresos, la gestión de los costes, la mitigación de riesgos, la gestión de las relaciones con los clientes y otras empresas, y mucho más. En la ciencia de los datos, la minería de datos también se denomina descubrimiento de conocimientos.

La minería de datos implica el uso de varias herramientas de diversos campos, como la inteligencia artificial y la estadística. También se necesitan algunos conocimientos de gestión de bases de datos para aprender a analizar mejor los conjuntos de datos. La minería de datos es un tema importante, ya que las técnicas utilizadas sirven de apoyo a otras disciplinas y aplicaciones. Los modelos de recomendación utilizados en el aprendizaje automático y los algoritmos de los motores de búsqueda son algunas de las principales áreas en las que la minería de datos se utiliza con frecuencia.

Tipos de minería de datos

La minería de datos está estrechamente relacionada con el aprendizaje automático. Los modelos de aprendizaje automático recopilan datos que se utilizan para realizar pruebas autónomas, de manera que, a largo plazo, los modelos puedan tomar sus propias decisiones. La minería de datos consiste en recopilar esta información con un propósito específico. Desde la relación del aprendizaje automático, podemos considerar la minería de datos en términos de los procesos de aprendizaje. Esto nos da el aprendizaje supervisado y no supervisado. Veamos esto en profundidad:

Aprendizaje supervisado

Los modelos de aprendizaje supervisado se construyen para la clasificación y la predicción. Es un proceso elaborado que se centra en un resultado

predeterminado. En el aprendizaje supervisado, el objetivo del modelo de aprendizaje es determinar el valor de cualquier observación.

Los filtros de spam, por ejemplo, determinan si tus correos electrónicos son legítimos o spam, y los clasifican en consecuencia. Así que, en primer lugar, los filtros de spam predecirán la naturaleza de cada correo electrónico, y luego los enviarán a la carpeta de spam en lugar de a la bandeja de entrada.

A continuación se presentan algunos de los modelos analíticos que se utilizan en la minería de datos supervisada y que usted aprenderá con el tiempo:

- **Regresión lineal**

La regresión lineal en la minería de datos es un modelo que anticipa el valor de las variables continuas en un conjunto de datos, independientemente del número de entradas de datos. Por ejemplo, en el negocio inmobiliario, los agentes de la propiedad utilizan una serie de factores para predecir el valor de un anuncio. Algunos de los factores que tienen en cuenta son el código postal, el año de construcción de la casa, el número de dormitorios, el número de cuartos de baño, el tamaño del recinto y otras instalaciones o servicios disponibles.

- **Regresión logística**

La regresión logística se centra en la predicción de la probabilidad de que se produzca una variable específica. Al igual que la regresión lineal, también se realiza con independencia del número de entradas de datos independientes. Un ejemplo de aplicación es el sector de los préstamos financieros. Los prestamistas determinan la solvencia de un prestatario para establecer si incumplirá los préstamos utilizando factores como sus ingresos, su edad, su calificación crediticia y muchos otros factores personales.

- **Árboles de regresión (clasificación**)

Los árboles de regresión se utilizan en la minería de datos para predecir resultados de variables específicas y continuas. En función de los datos disponibles, estos modelos se diseñan para generar reglas específicas que crean grupos para las variables que comparten similitudes. El valor predicho de una observación, por tanto, se convierte en el grupo al que se asigna, basándose en las características inherentes.

- **Series temporales**

El análisis de series temporales para la extracción de datos consiste en predecir variables independientes. Estos modelos pueden utilizarse para predecir cómo cambia la demanda en función de diferentes factores. Por ejemplo, en el sector minorista, la demanda puede ser una función del tiempo porque la demanda de los clientes cambia con las estaciones. Con esta información, la dirección puede tomar medidas y reponer sus tiendas en consecuencia para no sufrir problemas de inventario.

- Redes neur**onales**

Se trata de modelos analíticos de datos cuyo funcionamiento toma mucho de las operaciones del cerebro humano. Las redes neuronales se estructuran en torno al cerebro y a la red de neuronas. Las redes neuronales en la minería de datos dependen de las señales. Este proceso es el preferido en muchas organizaciones porque ofrece resultados casi al instante. Un buen ejemplo de minería de datos en uso a través de redes neuronales es en los coches autoconducidos. Estos coches están entrenados para leer el tráfico y otros estímulos de su entorno inmediato para poder tomar decisiones en fracciones de segundo sobre cómo llegar de un punto a otro.

Aprendizaje no supervisado

A diferencia de los modelos de aprendizaje supervisado, las tareas de aprendizaje no supervisado se modelan para entender y describir los datos de manera que puedan identificar patrones y tendencias dentro de los datos. Uno de los mejores ejemplos de esto son los sistemas de recomendación. Dado que los usuarios son diversos, estos modelos rastrean los patrones de los usuarios, identificando recomendaciones personalizadas que pueden ser

utilizadas por los responsables de la toma de decisiones para mejorar la experiencia que tienen los clientes.

A continuación se presentan algunos de los modelos analíticos que se utilizan en la minería de datos no supervisada y que aprenderá con el tiempo:

- **Agrupación**

Los modelos de agrupación se construyen para identificar datos similares y agruparlos. Estos modelos son ideales en situaciones en las que se utilizan datos complejos para definir un único resultado.

- Análisis de **componentes principales**

En muchos casos, nos encontramos con conjuntos de datos en los que la correlación entre las nuevas variables y las variables de entrada no es explícitamente obvia. Aquí es donde resulta útil el análisis de componentes principales. Lo que ocurre aquí es que el modelo registrará la misma información de las variables de entrada y las nuevas variables, pero no con las mismas variables. La idea aquí es mejorar la precisión y la utilidad de los datos utilizados en la minería de datos supervisada reduciendo las variables consideradas.

- **Análisis de asociación**

Este modelo se utiliza para determinar los artículos que pueden emparejarse. También se denomina análisis de la cesta de la compra. Uno de los mejores lugares donde se puede encontrar esto en acción es en el supermercado. En muchos casos, emparejan algunos artículos y los distribuyen uniformemente dentro de la tienda. La idea es asegurarse de que usted pueda ver y comprar más artículos y, de paso, aumentar su volumen de ventas.

Aplicaciones de minería de datos

La minería de datos se lleva a cabo en prácticamente todas las industrias en este momento. Los datos están en el centro de la mayoría de las decisiones que toman las empresas, y para obtener todos los datos que necesitan, es

importante que inviertan en alguna forma de minería de datos. La profundidad del proceso es irrelevante, siempre y cuando obtengan los datos que necesitan para satisfacer sus necesidades y, al mismo tiempo, cumplan con las regulaciones establecidas por diferentes organizaciones y convenios de los que son parte, como el GDPR, que describen los procedimientos, procesos y requisitos para el manejo de datos.

A continuación se presentan algunos de los casos en los que la minería de datos tiene lugar en diferentes organizaciones e industrias de nuestro entorno:

- **Filtración de spam**

Hoy en día recibimos muchos correos electrónicos, la mayoría de los cuales son perjudiciales de una forma u otra. El correo basura es un dolor de cabeza para todos. Todos los principales proveedores de servicios de correo electrónico que utilizamos cuentan con técnicas de filtrado de spam para proteger a los usuarios de ataques de phishing, malware y cualquier otro problema que pueda surgir de dicho correo.

Utilizando técnicas de minería de datos, sus sistemas monitorizan todos los correos electrónicos en el proceso de aprendizaje de las características de los correos maliciosos. Este informe se utiliza entonces para alertar a los diferentes aparatos de seguridad existentes y marcar los correos electrónicos como spam. Este es un aspecto del aprendizaje automático que aprenderás en una fase avanzada, y podrás construir tus propios filtros de spam. Además de detectar los mensajes de spam, existen mecanismos que van más allá y filtran los mensajes a la carpeta de spam antes de que lleguen a la bandeja de entrada del usuario.

- La **bioinformática en la** industria **sanitaria**

En el sector sanitario, la extracción de datos ayuda a los expertos médicos a determinar la posibilidad de que un paciente corra el riesgo de padecer determinadas enfermedades. Esta información se obtiene de una base de datos de los factores de riesgo más comunes. Además, también examinan la composición genética, los antecedentes familiares y los datos demográficos, que se modelan en torno a características únicas, de modo que es posible identificar ciertos problemas de salud antes de que se conviertan en un brote completo.

- **Mecanismos antifraude**

El mercado financiero está lleno de oportunidades para los estafadores. Desde los usuarios que no cierran sus cuentas hasta los que utilizan servicios públicos de Internet gratuitos y dispositivos compartidos para acceder a detalles importantes sobre sus cuentas, muchos usuarios son vulnerables ahí fuera. Por ello, las instituciones financieras se han encargado de crear y desplegar sistemas de extracción de datos que puedan detectar y prevenir fácilmente las transacciones ilegales.

La informática forense es un departamento de las finanzas que está ganando popularidad con el tiempo. Esto se debe a que los defraudadores también se benefician de los avances tecnológicos y son cada vez más inteligentes. El tipo de extracción de datos que se lleva a cabo en las instituciones financieras es sigiloso, de manera que los clientes disfrutan de una increíble protección sin saberlo.

Una de las técnicas que utilizan es el seguimiento de los hábitos de gasto de los clientes. Con esta información, los modelos de detección y prevención del fraude son alertados en caso de que se produzca una anomalía en el comportamiento de las transacciones. Algunos modelos retienen los pagos o bloquean las transacciones hasta que el cliente pueda verificarlas.

- Programas de **recomendación**

Los sistemas de recomendación son muy populares hoy en día en el comercio online. Cada vez que se visita una tienda online, aparecen recomendaciones mientras se navega por su página. Además, cuando ves un producto online, recibes recomendaciones sobre otros productos que serían ideales para comprar junto al que estás mirando.

Cuando se introdujeron los sistemas de recomendación, no eran tan eficaces como ahora. Sin embargo, con el tiempo estos sistemas se han integrado con modelos de aprendizaje automático para hacerlos más precisos y adecuados a sus necesidades. Esto es posible gracias al comportamiento predictivo del consumidor.

Los modelos de aprendizaje automático que se utilizan suelen controlar el comportamiento de los clientes de todo el mundo. Utilizando algoritmos únicos, son capaces de crear grupos de clientes diferentes en función de sus

hábitos de compra. Hoy en día, algunos de los principales minoristas del sector no se limitan a utilizar modelos de minería de datos, sino que construyen sus propios modelos que se adaptan a sus necesidades. Así es como Amazon ha conseguido convertirse en líder del comercio minorista online.

La extracción de datos permite a los minoristas comprender mejor las experiencias de los clientes. A través de estos datos, son capaces de mejorar las experiencias de los clientes, y esta es una de las razones por las que tienen clientes fieles que les compran siempre. Otro ejemplo de empresa que ha utilizado eficazmente los sistemas de recomendación es Netflix. La necesidad de un sistema de recomendación preciso era tan importante para su modelo de negocio que ofrecieron un millón de dólares a los desarrolladores que construyeran un algoritmo que mejorara su sistema de recomendación. Finalmente, consiguieron un sistema que ofrecía una mejora del 8%. A lo largo de los años, han perfeccionado el modelo para ofrecer resultados aún más precisos.

- **Calificaciones de crédito y** gestión de **riesgos**

Cada vez que quiera pedir un préstamo a un banco o a cualquier otra institución financiera, ésta debe determinar primero su solvencia. Esto ayuda al prestamista a determinar si eres un prestatario de riesgo y, en función de tu perfil de riesgo, el tipo de interés que te cobrarán si aceptan concederte el préstamo.

C o nclusión

Es un gran momento para ser un científico de datos. Cada día que pasa, siempre hay algo nuevo en este campo. ¿Quieres convertirte pronto en un científico de datos? Bueno, hay algunas cosas que puedes hacer para asegurarte de que estás absolutamente preparado para lo que te espera. Sí, esto no tiene nada que ver con más códigos. Relájate. Aquí tienes algunos de nuestros consejos para asegurarte de que tienes una maravillosa carrera por delante.

Plan

Es probable que te sientas muy cómodo al abordar algunas formas de la ciencia de los datos después de terminar este libro. Si siente esa emoción, entonces está en el camino correcto. Sin embargo, sigue siendo necesario planificar bien antes de adentrarse en ella.

Hay muchas cosas que deberá comprender antes de poder alcanzar sus objetivos, y sin el plan adecuado, esto será prácticamente imposible. Así que asegúrate de tener una lista de control antes de dar el salto. Créame, será importante a largo plazo.

La planificación también garantizará que sepas qué aspectos de la ciencia de los datos te gustan más. Saber esto te asegurará que sabes lo que te interesa, así como tus puntos fuertes y débiles de cara a la nueva carrera.

Leer más

En este mundo de la ciencia de los datos, la lectura seguirá siendo una forma importante de asegurarse de seguir avanzando. Gracias a Internet, hay muchos libros que pueden ayudarte a conseguir tu objetivo. Así que asegúrate de estar siempre leyendo y explorando cosas y conceptos nuevos. Lo único constante en el sector de la ciencia de los datos es que las cosas cambian constantemente cada día que pasa. Así que asegúrate de estar al tanto de todo.

Buscar la comunidad profesional

Afortunadamente para usted, hay muchos expertos por ahí que pueden considerarse expertos en todo lo relacionado con la minería de datos. Aunque este libro le ofrece una guía para principiantes, a medida que vaya profundizando, tendrá que reunir algo más de experiencia. La experiencia que se puede obtener leyendo libros es limitada. Al final tendrá que buscar a los expertos en la materia. Ellos proporcionan algunos consejos valiosos, que probablemente no encontrará en ningún libro. Así que ya está.

Practicar todos los días

No puedes convertirte en un gran científico de datos sin ponerte a trabajar. Tendrás que trabajar para desarrollarte en el campo cada día. La verdad es que con la ciencia de datos siempre hay algo nuevo que aprender, y seguramente te harás un mundo de bien practicando. ¿Sabes qué es lo mejor de practicar? Que se hace más fácil. Pronto te habrás acostumbrado y estarás preparado para seguir adelante con cada año que pase.

Dar el salto

Por fin estás preparado. ¿Tienes los pies fríos? Ha llegado el momento de dar el salto. Ya has invertido mucho en aprender el oficio, y es hora de iniciar tu carrera en la ciencia de los datos. ¿Y si eres el propietario de una organización que busca formas de asegurarse de que su empresa obtiene lo mejor de la ciencia de los datos y lo que ofrece? Entonces. También es el momento de dar el salto y asegurarse de que su empresa sigue recogiendo los frutos de la integración de la ciencia de datos en su negocio.

Consigue un trabajo y sigue haciendo contactos

Hay mucho más que explorar ahí fuera. Sigue trabajando y amplía tus horizontes más allá de la ciencia de los datos. Todo lo que necesitas es una medida de confianza. Una vez que la tengas, el mundo se pondrá a tus pies.

El siguiente paso es empezar a poner en práctica parte de la información que hemos abordado en esta guía. Como empresa, si aún no ha comenzado

a recopilar datos de diversas fuentes, ya sea en línea, en las redes sociales, de los clientes que compran en su sitio o más, entonces ya se está quedando atrás. Sólo una vez recopilada esa información podrá comenzar el verdadero trabajo de ordenar todos esos datos y descubrir la información que se esconde en su interior. Esta guía ha dedicado algún tiempo a examinar este proceso y todos los pasos que puede dar para que la ciencia de los datos, con la ayuda del lenguaje de codificación Python, trabaje para usted.

Esta guía nos ha proporcionado una gran cantidad de información sobre la ciencia de los datos, sobre cómo funciona, el aprendizaje automático, el lenguaje Python, e incluso algunos de los ejemplos de cómo se puede poner todo esto junto y realmente hacer que todo funcione. A menudo, la ciencia de los datos suena difícil y demasiado duro para trabajar, pero esta guía nos ha mostrado algunos de los pasos prácticos que podemos tomar para poner todo junto.

Cuando finalmente esté listo para tomar algunos de los datos que ha estado acumulando, y esté emocionado por hacer que todo esto funcione para usted en términos de proporcionar un mejor servicio al cliente, y realmente ver algunos buenos resultados en las decisiones que toma para su negocio, asegúrese de revisar esta guía para ayudarle a comenzar con Python para la ciencia de datos.

Al final, tienes razón. Mantén la cabeza alta. Estar centrado y aprender más sobre la ciencia de los datos resultará ser la mejor decisión que habrás tomado este año.