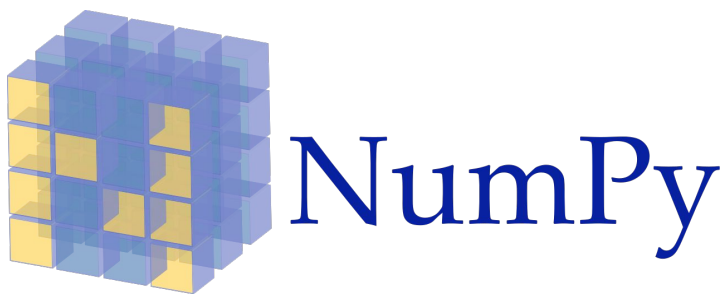
Platzi

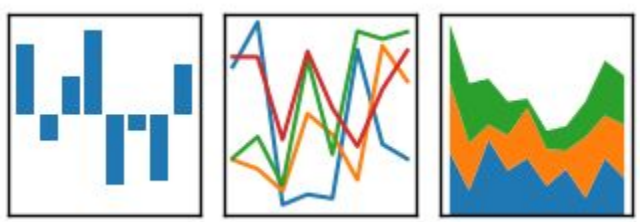
Ingeniería de Datos

David Aroesti

Introducción

Introducción





1. Introducción

¿Qué es la Ciencia de Datos?

¿Qué es la Ciencia de Datos?

- La Ciencia de Datos es la disciplina que se encarga de extraer conocimiento a partir de los datos disponibles.
  - En muchas ocasiones es necesario ser proactivo en la captura de datos
- Es multidisciplinaria:
  - Computer science:
    - Estructuras de datos
    - Algoritmos
    - Visualización
    - Big Data support
    - Programming
  - Estadística
    - Regresiones
    - Inferencias
  - Conocimiento del dominio
    - Preguntar lo correcto
    - Interpretar los resultados

¿Qué es la Ciencia de Datos?

- Este campo se auxilia de muchos otros campos:
  - Bases de datos
    - SQL y NoSQL
  - Análisis de texto y Procesamiento de lenguaje natural
  - Análisis numérico de datos y minado de datos
  - Análisis de redes
  - Visualización de datos
  - Machine Learning e Inteligencia Artificial
  - Análisis de señales digitales
  - Análisis de datos en la nube (Big Data)

Roles

- Data engineer
  - Se encarga de obtener los datos
  - Limpiarlos y estructurarlos para posterior análisis
  - Crear pipelines de análisis automatizado
  - Utilización de herramientas en la nube
  - Análisis descriptivo de los datos
  - Background: Software engineering
- Data scientist
  - Análisis matemático de los datos
  - Identificación de variables relevantes para el negocio
  - Generación de modelos predictivos y prescriptivos
- Machine Learning engineer
  - Creación de sistemas predictivos y prescriptivos de gran escala
  - Mantenimiento y ajuste del modelo

1. Introducción

Configuración del ambiente

Configuración del ambiente

Anaconda

Configuración del ambiente

Jupyter

Tipos de datos

Tipos de datos

- Primitivos
  - int, str, bool, float, hex, oct, datetime, objetos especiales
- Estructurados
  - Bases de datos
  - Data warehouses
- Semi estructurados
  - json APIs
  - Datos tabulares (csv, excel)
- No estructurados
  - HTML
  - Texto libre
  - Curriculum vitaes
  - Imágenes, audio, social media
  - Datos científicos
- Cualitativos vs cuantitativos
- Tiempo real vs históricos

Fuentes de datos

Fuentes de datos

- Web
- APIs
- Datasets
- System logs
- User analytics
- Sensor data
- Y muchos más...

1. Introducción

ETL

ETL

- Extract
  - Es el proceso de lectura de datos de diversas fuentes
    - Bases de datos de aplicaciones
    - CRM
    - Archivos CSV
    - Datasets publicos
- Transform
  - Es el proceso de limpieza de datos para su posterior inserción en un data warehouse
    - Limpieza
    - Estructurado
    - Enriquecimiento
- Load
  - Es el proceso de inserción de datos al data warehouse
    - Depende del tipo de solución que se haya escogido

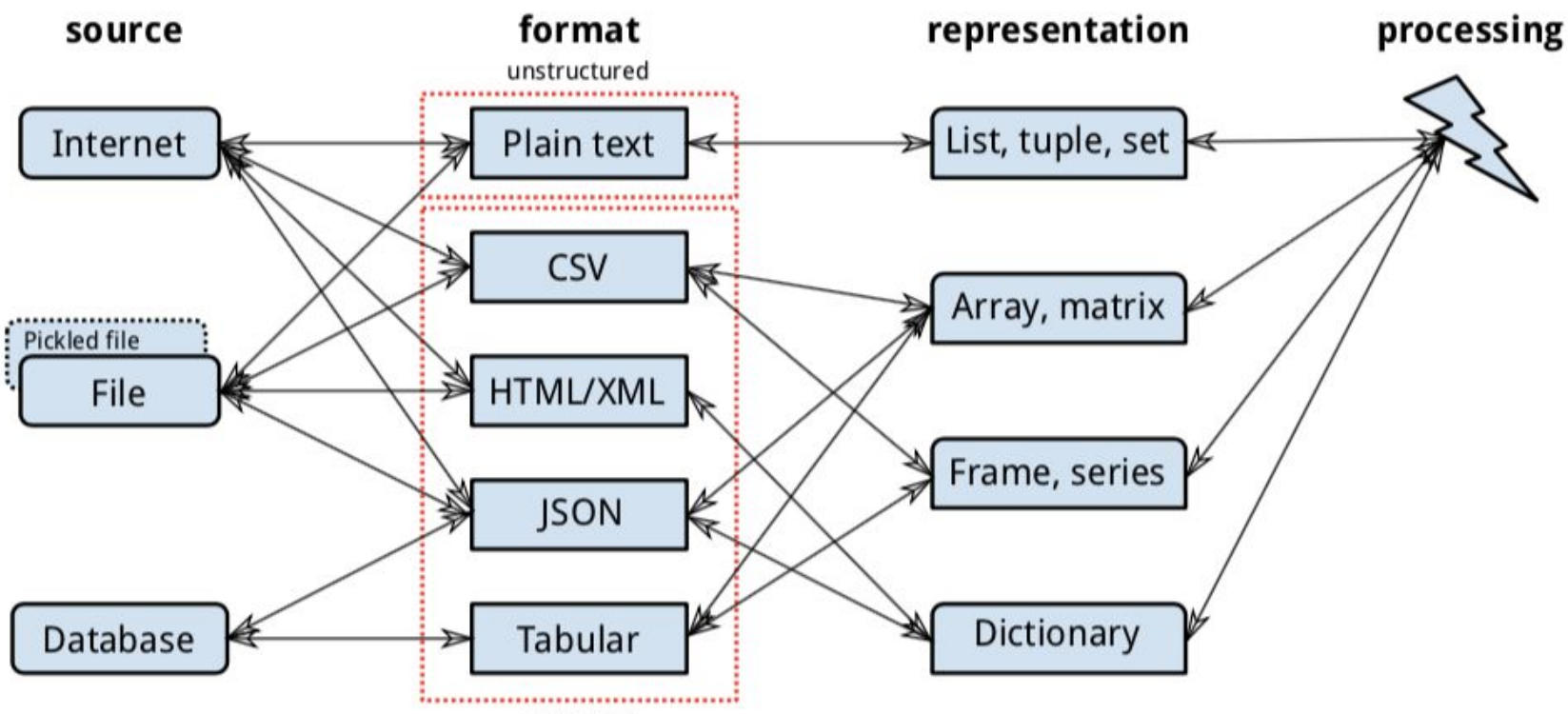
ETL


CRM
ERP
PIM
Operational System
Server
Flat Files
ETL
Extract
Transform
Load
Data Warehouse
Business Inteligence
Data Mining
Business Analytics
Big Data
Predictive Analysis
Reporting
Planning
Data Visualisation

2. Web scraping

Adquisición de datos

Adquisición de datos



2. Web scraping

Tecnologías web

Tecnologías web

- Internet es el sistema global de computadoras interconectadas a través de redes digitales que se comunican a través del protocolo TCP/IP
  - Privadas, públicas, académicas, negocios, gobiernos, etc.
  - Muchas aplicaciones
    - WWW (http), telefonía (voip), mail (pop3, imap), compartir archivos (ftp)
- La web es espacio de información en el cual documentos (y otros recursos web) se pueden accesar a través de URLs y vínculos (links). La comunicación se da a través del protocolo HTTP

Tecnologías web

- HTML
  - Estructura
- CSS
  - Presentación
- Javascript
  - Interactividad y cómputo
- Json
  - Transferencia de datos

Solicitudes a la web

Solicitudes a la web

- Para realizar solicitudes a la web en Python, se utiliza la librería requests
- Creado con el PEP20 en mente
  - En mi opinión, mejor que las alternativas (urrlib3)
- Permite utilizar varios verbos HTTP
  - GET, POST, PUT, DELETE, PATCH, OPTIONS

2. Web scraping

Extracción de información del HTML

BeautifulSoup

- Para manipular documentos HTML, en Python utilizamos BeautifulSoup
  - BeautifulSoup permite analizar gramaticalmente (“parsear”) el documento HTML para que lo podamos manipular programáticamente y podamos consultarlo
  - El documento se convierte en un árbol de nodos
  - Le realizamos consultas con selectores de CSS
- Ej.

```
soup = bs4.BeautifulSoup(response.text, 'html.parser')
soup.select('body')
```

Page Object Pattern

- Es un patrón que viene del mundo de las pruebas automatizadas
- Consiste en esconder los queries específicos que se utilizan para manipular un documento HTML detrás de un objeto que representa la página web
- Si estos queries se añaden directamente al código principal, el código es bastante frágil y arreglarlo se vuelve muy difícil
- Ej.

```
class WebPage:
    …

    @property
    def page_header(self):
        return soup.select(‘.some-query h1’)
```

Del scraping al crawling

Del scraping al crawling

- La diferencia entre scraping y crawling es relitivamente vaga
- El web crawling permite crear programas que naveguen la web de manera autónoma
  - Identifica los links, los guarda en una base de datos y crea un plan para continuar navegando
- Consejos:
  - Piensa muy bien qué tipo de datos quieres recolectar
  - Utiliza una base de datos para identificar vínculos que ya hayas visitado
  - Separa la parte del crawling (extracción de vínculos y generación de un plan de navegación) de la parte del scraping (extracción de información)
  - Incorpora un criterio para detener a tu crawler
  - Si puedes, utiliza programación paralela
    - `from requests import async`
  - No te metas en problemas legales (robots.txt)

Pandas

Introducción

- Pandas otorga facilidades para realizar “domado de datos” (data wrangling)
- Introduce dos estructuras de datos:
  - Series
    - Un array unidimensional el cual permite el acceso a sus valores a través de etiquetas (labels)
  - DataFrame
  - Un conjunto de Series que forman una tabla
    - Se puede acceder a los datos a través de etiquetas que representan filas y columnas
    - Muy similar a una hoja de cálculo o una tabla de base de datos

Introducción

- Estas estructuras de datos no deben verse como contenedores de datos
  - Son estructuras intermedias para:
    - Manipular datos
    - Manejar datos faltantes
    - Realizar operaciones aritméticas y booleanos
    - Operaciones similares a las bases de datos como merge y agregaciones
    - Leer datos de archivos y escribir a disco

Series

- Es un vector unidimensional con un índice
  - El índice lo podemos utilizar para tener acceso a través de etiquetas o posiciones
  - Los datos son homogéneos (sólo un tipo de datos)
- Se pueden crear a partir de cualquier secuencia: listas, tuplas, numpy arrays
  - Ej.
    series = pd.Series([1, 2, 3])

- Problemas:
  - La serie se comporta igual a una lista
    - Python duck typing
  - Una mejor aproximación utiliza diccionarios para definir el índice
    - Ej.
      series = pd.Series({‘a’: 1, ‘b’: 2, ‘c: 3})

DataFrames

- Es una tabla con filas y columnas etiquetadas
- Se pueden construir con arrays bidimensionales de numpy, lista de listas, lista de tuples, un diccionario de Python u otro DataFrame
    - Al igual que en las Series, cuando utilizamos un diccionario, sus llaves se convierten en el índice de las columnas
    - Cuando utilizamos un lista, podemos utilizar el parámetro opcional columns

Índices y selección

- Existen muchas formas de acceder a los datos
  - Dictionary like
    - Ej.

      ```
      df['col1'] # Regresa un DataSeries
      df[['col1', 'col3']] # Regresa un DataFrame
      ```

  - Numpy like
    - Ej.

      ```
      df.iloc[:] # fila
      df.iloc[:, :] # fila, columna
      ```

  - Label based
    - Ej.

      ```
      df.loc[:] # fila
      df.loc[:, :] # fila, columna
      ```

Data wrangling

- Esta es una de las actividades más importantes para todo los profesionales de los datos
- Consiste en limpiar, transformar y enriquecer el dataset para objetivos posteriores
  - Ej.
    - Inserción en una base de datos
    - Visualización
- Pandas es una de las herramientas más poderosas para realizar este ejercicio:
  - Nos permite generar transformaciones con gran facilidad
  - Nos permite trabajar rápidamente con datasets grandes
  - Nos permite detectar y reemplazar datos faltantes
  - Nos permite agrupar y resumir nuestros datos
  - Nos permite visualizar nuestros resultados

Datos faltantes

- Los datos faltantes son un hecho de la vida
- Pandas nos otorga distintas estrategias (todas con diferentes “tradeoffs”) para solucionar este problema
  - Eliminar el registro (fila o columna)
    - .isna()
    - .notna()
    - .dropna()
  - Reemplazar el registro
    - .fillna() # utiliza un valor sentinela
    - .ffill() # utiliza el último valor utilizado

Combinación de datasets

- Los estudios más interesantes suceden cuando combinamos distintas fuentes de datos
- Podemos utilizar concatenación o join
  - Ej.
    dataframe_1 = pd.DataFrame([...])
    dataframe_2 = pd.DataFrame([...])
    dataframe_3 = pd.concat([dataframe_1, dataframe_2)

- También podemos agregar registros con el método append
  - Ej.
    dataframe_3.append([...])

Agregaciones y agrupado

- Casi siempre queremos agrupar nuestros datos para resumirlos
- Para eso podemos utilizar agrupaciones
  - Ej.
    grouped_dataframe = other_df.groupby(‘column_name’)
    grouped_dataframe[‘some_col’].agg([‘mean’, ‘max’])

Introducción a los sistemas de datos

Procesamiento en bloque y en chorro

- Procesamiento en bloque lo utilizamos con datos históricos
  - Cosas que ya sucedieron
- Procesamiento en chorro lo utilzamos con datos en tiempo real
  - Cosas que están sucediendo en este momento
- El criterio principal a tener en cuenta:
  - ¿Cuál es el valor de una decisión temprana?

SQL vs NoSQL

- SQL vs NoSQL describe las ventajas y desventajas de diversas aproximaciones a sistemas de datos
- La discusión es más relevante en el mundo de las aplicaciones (web y mobile), donde dependiendo de la aplicación en mano, la decisión puede ser fundamental para el crecimiento de la app
- La verdad es que para los profesionales de los datos, especialmente los Ingenieros de Datos, es necesario saber ambos.

Big Data y la nube

¿Por qué y cuándo usar la nube?

- La nube contiene el mayor poder de cómputo al que podemos acceder
  - Nos permite resolver en segundos problemas que se tardarían días en nuestra laptop
    - Utilizamos miles de computadoras en paralelo para resolver un problema y sólo pagamos por lo que utilizamos
- Podemos crear pipelines que se liguen a los sistemas de nuestra empresa
- Diversas nubes ya ofrecen paquetes completos para el ciclo de datos:
  - Ej. Google Cloud
    - Dataflow
    - Pub/Sub
    - Cloud Storage
    - Dataprep
    - Datalab
    - BigQuery
    - Dataproc
    - Firestore
    - etc.