DMLA 2022 - Plataformas para Machine Learning

Parte 2: Analisis de datos en Pandas

Francisca Cattan y Nicolás Alvarado

Pontificia Universidad Católica de Chile

Octubre, 2022

Vamos por más...

- Análisis exploratorio en Python
- Manipulación de DataFrames
- Preparación de datos parte 1: datos faltantes
- Unión y combinación

Qué es el análisis exploratorio (en Python)?

- Asume la existencia de los datos.
- Consiste principalmente en utilizar librerías para:
  - Limpiar y transformar los datos.
  - Explorar distintas dimensiones de los datos.
  - Calcular estadísticas de los datos.
  - Visualizar los datos.
  - Construir modelos predictivos preliminares.
- Para todo esto (y más), está Pandas.

Introducción a Pandas

¿Para qué usar Pandas?

Introducción a Pandas

¿Para qué usar Pandas?

- Manipular, analizar y visualizar datos.
- Consiste en dos estructuras principales: Series y DataFrame.

DataFrames

Un DataFrame es una tabla creada por pandas, por ejemplo:

DataFrames

Un DataFrame es una tabla creada por pandas, por ejemplo:

	area	pop
California	423967	38332521
Texas	695662	26448193
New York	141297	19651127
Florida	170312	19552860
Illinois	149995	12882135

DataFrames

La tabla anterior se generó de la siguiente forma:

```
1  import numpy as np
2  import pandas as pd
3  from IPython.display import display
```

DataFrames

La tabla anterior se generó de la siguiente forma:

```
1  import numpy as np
2  import pandas as pd
3  from IPython.display import display
```

Podemos crear o manipular datos,

```
1  area = pd.Series({'California': 423967,
2  'Texas': 695662, 'New York': 141297,
3  'Florida': 170312, 'Illinois': 149995})
4  pop = pd.Series({'California': 38332521,
5  'Texas': 26448193, 'New York': 19651127,
6  'Florida': 19552860, 'Illinois': 12882135})
7  data = pd.DataFrame({'area':area, 'pop':pop})
8  data
```

DataFrames

La ejecución de la última línea del código anterior entregará una visualización de este DataFrame, como si fuera una tabla de datos

```
8 data
```

	area	pop
California	423967	38332521
Texas	695662	26448193
New York	141297	19651127
Florida	170312	19552860
Illinois	149995	12882135

DataFrames

- Estadísticos de los datos: ”describe” y “value_counts”.
- Filtrado o proyección de datos: ”loc”.
- Creación de nuevos campos: ”apply”.

DataFrames

- Uno de los principales problemas con los datos es que “no están”.
- Muchos procedimientos no funcionan directamente con valores faltantes.
- Pandas provee gran cantidad de mecanismos para:
  - Detectar valores faltantes
  - Eliminar valores faltantes (filas y/o columnas)
  - Llenar valores faltantes

DataFrame

Exploremos un poquito:

DataFrame

Exploremos un poquito:

```
1  data['area']
```

DataFrame

Exploremos un poquito:

```
1  data['area']
```

Que arroja lo anterior?

DataFrame

Exploremos un poquito:

```
1  data['area']
```

Que arroja lo anterior?

```
California    423967
Texas         695662
New York      141297
Florida       170312
Illinois      149995
Name: area, dtype: int64
```

DataFrame

Equivalente a lo anterior sería usar data.area. Y si usamos data['Texas':'Florida']

DataFrame

Equivalente a lo anterior sería usar data.area. Y si usamos data['Texas':'Florida'] obtenemos:

	area	pop
Texas	695662	26448193
New York	141297	19651127
Florida	170312	19552860

DataFrame

- Que pasa con data[1:3]?
- Que pasa con data.loc[:'Texas',:'area']?
- Que pasa con data.iloc[:4,:1]?

DataFrame

Un problema típico es la exploración de diferentes fuentes.

DataFrame

Un problema típico es la exploración de diferentes fuentes.

- Cuando todo está en un DataFrame, la cosa fluye, pero la mayoría de las veces, tenemos más de uno
- Pandas entrega varios mecanismos para enfrentar esto.

DataFrame

Un problema típico es la exploración de diferentes fuentes.

- Cuando todo está en un DataFrame, la cosa fluye, pero la mayoría de las veces, tenemos más de uno
- Pandas entrega varios mecanismos para enfrentar esto.

Podemos enfrentarlo usando concatenación.

DataFrame

Tipos de contatenación

- Simple.
- Join externo.
- Join interno.
- Merge.

Resumen!

- Pandas: Herramienta flexible que nos sirve para (casi) todo.
- Manipulación de DataFrames: Operaciones intuitivas y flexibles.
- Combinación de DataFrames: Concatenación y joins.