DataSci 400

lesson 3: exploring data

Seth Mottaghinejad

today's agenda

- exploratory data analysis
- `pandas` data types
- statistical and visual exploration
  - univariate summaries
  - bivariate summaries
- dealing with missing values

exploratory data analysis

- **EDA** is a very important step **prior to** machine learning
- we say **garbage in, garbage out** to emphasize importance of EDA
- the goal is to know your data and fix any irregularities
    - make sure column types are correct
    - make sure missing values are properly flagged
    - make sure the distribution of columns match what we expect
    - any other **sanity checks** we deem appropriate
- EDA steps can depend on context (domain) to some extent

`pandas` column types

column type	description
`object`	text data or data with non-numeric characters
`int64/32/16/8`	integer numbers
`float64/32/16`	floating point numbers (numbers with decimals)
`bool`	True/False values (also called **binary**)
`datetime64`	date and time values
`timedelta[ns]`	differences between two datetimes
`category`	**finite** (and usually small) list of text values

about column types

- it's important to remember with column types that appearances are not the only thing: **what type a column should have depends on how it's used** by us in our analysis
- data scientists are usually **not** the people in the organization in charge of storing and managing data, they are the **end users**, as such their expectations of column types may differ
- data is often read from a variety of sources with **varying schema structure**, and when extracting data from these sources, **data types are not automatically inherited**

some examples

- a timestamp column by default inherits the `object` type because it contains non-numeric characters: needs to converted to `datetime` **explicitly** (using `pd.to_datetime`)
- a `category` column by default inherits the `object` type and needs to be explicity converted to `category` when doing so is appropritate (when the categories are **few and well-defined**)
- the choice of `int64` vs `int32` etc. affects storage and precision
- a categorical is often **encoded** using integers, which means `pandas` reads it as `int` and we have to convert it to `object` or `catgory`

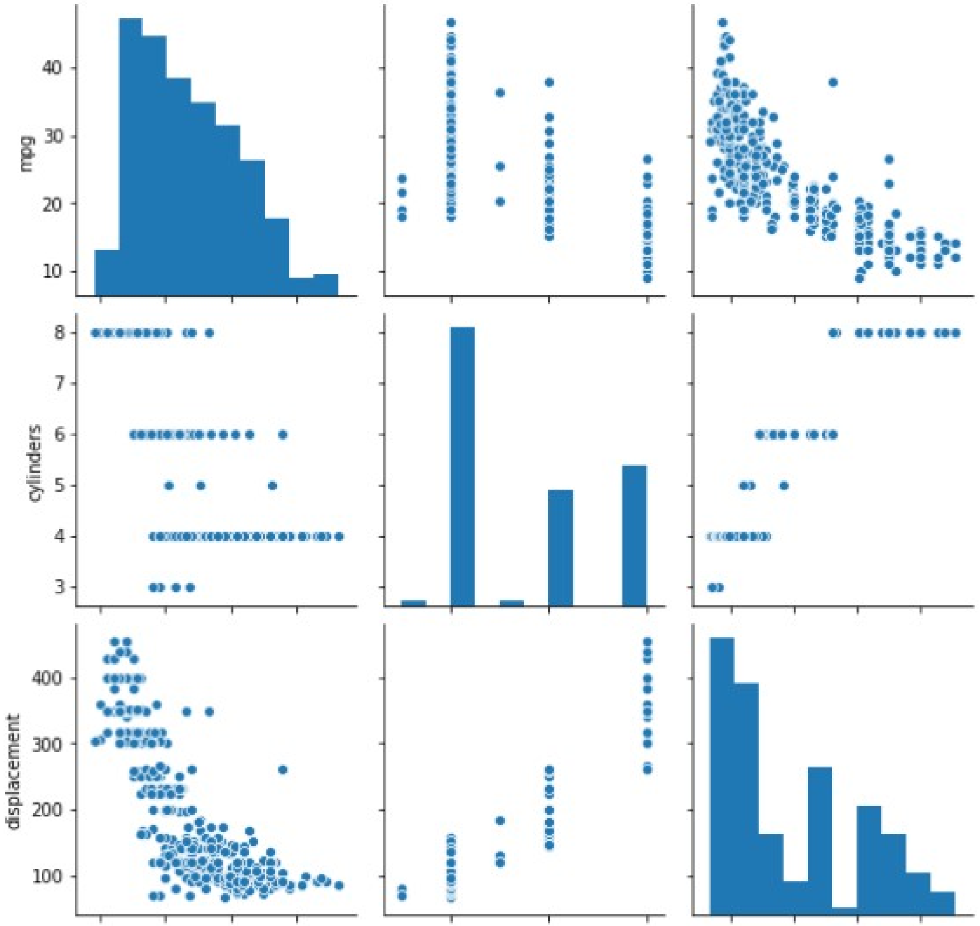
[lab time]()

many datasets come with a **data dictionary**, which is just a description of the data and the fields in it, and it can guide us when we are deciding what the appropriate data types are

- go to the page for [adult income]() dataset website on the **UCI Machine Learning Repository**
- based on column descriptions, specify what types each column will have if we read the data using `pandas`, and propose what the column type **should be** if we need to change it
- propose some ways to explore each column with stats and charts

scatter plot matrix

- a quick way to explore all numeric columns at once
- do this on a **sample** of the data if data size is large
- may need to transform columns with **extreme values** so plots aren't **skewed**, by **trimming** or using a **log transform**



dealing with missing values

- also called **null values** or **NAs**
- missing values are **very common** in most real-world data
- missing values can be generated if we try to do **problematic computations**, such as 0/0 or $\log(x)$ where $x < 0$
- if missing value is not necessarily randomly distributed, it can **bias** the machine learning later
- replacing missing values with a reasonable value is called **missing value imputation**, and it can be a simple or complex process

the end