COGS 108 Week 3 A04/A07

Oct 16, 2023

Slides Adapted from Previous Quarter

Before we start...

slido

Have you started Assignment 1 Yet?

ⓘ Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

AGENDA FOR TODAY

01 LOGISTICS

02 COGNITIVE PERSPECTIVE

03 DISCUSSION LAB 2

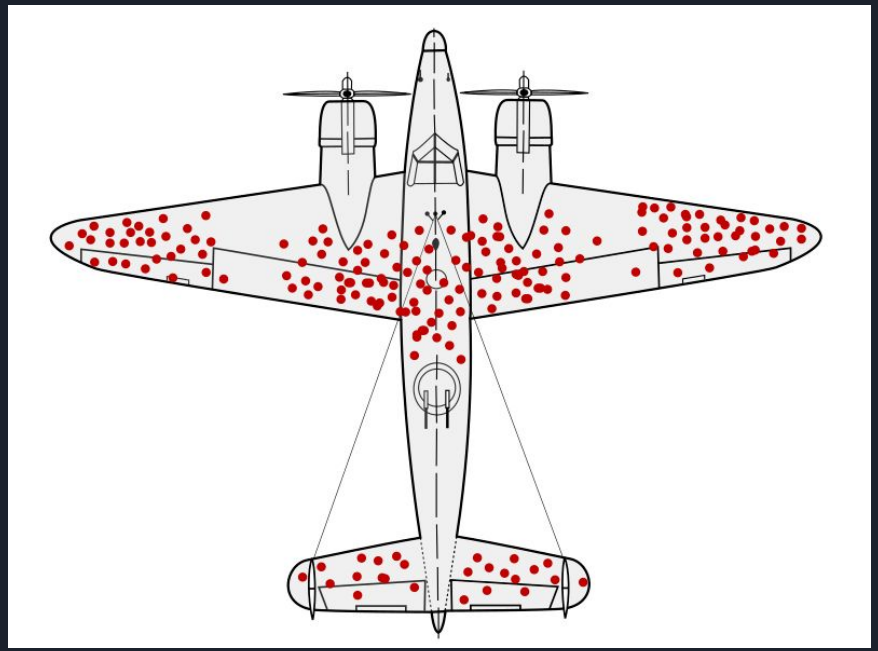
LOGISTICS

DUE DATES

- Quiz 2 is due Oct 16, 11:59PM (TONIGHT)
- Assignment 1 is due Oct 18, 11:59PM (Wednesday)
- Discussion lab 2 is due Oct 20, 11:59PM (Friday)
- PLEASE SUBMIT YOUR GITHUB ID

COGNITIVE Perspectives on DATA Science

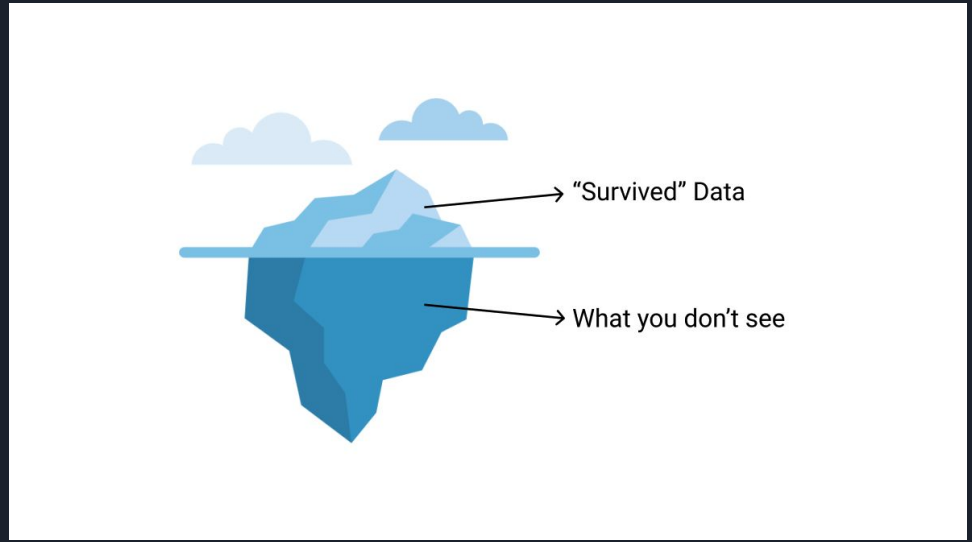
SURVIVORSHIP BIAS



During World War II, researchers from the non-profit research group the Center for Naval Analyses were tasked with a problem. They needed to reinforce the military's fighter planes at their weakest spots. To accomplish this, they turned to data. They examined every plane that came back from a combat mission and made note of where bullets had hit the aircraft. Based on that information, they recommended that the planes be reinforced at those precise spots.

SURVIVORSHIP BIAS

Say you begin working on a data set. You have created your features and reached a decent accuracy on your modelling task. But maybe you should ask yourself if that is the best result you can achieve. Have you tried looking for more data? Maybe adding weather forecast data to the regular sales variables that you use in your models would help you to forecast your sales better.



DISCUSSION LAB 2

DATA WRANGLING & DATA CLEANING

DATA WRANGLING

Data wrangling in python deals with the below functionalities:

1. **Data exploration:** In this process, the data is studied, analyzed and understood by visualizing representations of data.
2. **Dealing with missing values:** Most of the datasets having a vast amount of data contain missing values of *NaN, they are needed to be taken* care of by replacing them with mean, mode, the most frequent value of the column or simply by dropping the row having a *NaN* value.
3. **Reshaping data:** In this process, data is manipulated according to the requirements, where new data can be added or pre-existing data can be modified.
4. **Filtering data:** Some times datasets are comprised of unwanted rows or columns which are required to be removed or filtered

Ref: https://www.geeksforgeeks.org/data-wrangling-in-python/

NumPy and Pandas

import numpy as np

Import pandas as pd

pd.read_csv()

PANDAS	NUMPY
When we have to work on **Tabular data**, we prefer the p*andas* module.	When we have to work on **Numerical data**, we prefer the n*umpy* module.
The powerful tools of pandas are **Data frame and Series.**	Whereas the powerful tool of *numpy* is **Arrays.**
Pandas consume **more memory**.	*Numpy* is **memory efficient.**
Pandas has a better performance when a number of rows is **500K or more.**	*Numpy* has a better performance when number of rows is **50K or less.**
Indexing of the *pandas* series is **very slow** as compared to *numpy* arrays.	Indexing of *numpy* Arrays is **very fast**.
Pandas offer a have2d table object called **DataFrame.**	*Numpy* is capable of providing **multi-dimensional arrays.**

PANDAS OPERATIONS

How to read csv files into a pandas df:
pd.read_csv('https://data.sample/source.cvs')

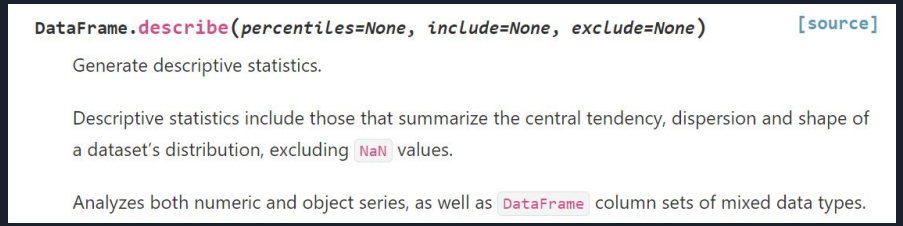
To explore the data frame:

dataframeName.head(n)

- Returns n rows of the dataframe, default 5

dataframeName.describe()

- Generates descriptive statistics like value counts, mean, standard deviation for an entire data frame
- Can also be used for a single column using df['columnName'].describe()



`DataFrame.describe(percentiles=None, include=None, exclude=None)` [source]

Generate descriptive statistics.

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values.

Analyzes both numeric and object series, as well as `DataFrame` column sets of mixed data types.

PANDAS OPERATIONS

data.iloc[0] # first row of data frame

data.iloc[1] # second row of data frame

data.iloc[-1] # last row of data frame

data.iloc[:,0] # first column of data frame

data.iloc[:,1] # second column of data frame

.iloc selections - position based selection

data.iloc[<row selection], <column selection>]

Integer list of rows: [0,1,2]	*Integer list of columns: [0,1,2]*
Slice of rows: [4:7]	*Slice of columns: [4:7]*
Single values: 1	*Single column selections: 1*

PANDAS OPERATIONS

To print a list of column names use the function:

- list(dataframeName)

Column names can also be renamed to more appropriate names using:

- df['smoking','alcohol','gambling',
- 'skydiving','speeding', 'cheated',
- 'steak', 'steak_preference','gender',
- 'age', 'income', 'education', 'region']
- ['name1', ‘name2’, ‘name3’]

PANDAS OPERATIONS: CLEANING

Syntax: *Pandas.isnull("DataFrame Name") or DataFrame.isnull()*

Parameters: *Object to check null values for*

Return Type: *Dataframe of Boolean values which are True for NaN values*

isnull()

null_rows = dataframeName.isnull().any(axis=1).sum()

- Returns number of rows with null values in the dataframe

PANDAS OPERATIONS: CLEANING

To drop rows or columns with null values we use:

- dropna()

DataFrame.dropna(*axis=0*, *how=_NoDefault.no_default*, *thresh=_NoDefault.no_default*, *subset=None*, *inplace=False*)

Example to drop any row or column with a null value:

- dataframeName = dataframeName.dropna(how='all')

Dropping rows and/or columns is generally useful if our data contains a large amount of null values; rows with missing values may not be valuable to us

Parameters: **axis** : *{0 or 'index', 1 or 'columns'}, default 0*

Determine if rows or columns which contain missing values are removed.

- 0, or 'index' : Drop rows which contain missing values.
- 1, or 'columns' : Drop columns which contain missing value.

Changed in version 1.0.0: Pass tuple or list to drop on multiple axes. Only a single axis is allowed.

how : *{'any', 'all'}, default 'any'*

Determine if row or column is removed from DataFrame, when we have at least one NA or all NA.

- 'any' : If any NA values are present, drop that row or column.
- 'all' : If all values are NA, drop that row or column.

thresh : *int, optional*

Require that many non-NA values. Cannot be combined with how.

subset : *column label or sequence of labels, optional*

Labels along other axis to consider, e.g. if you are dropping rows these would be a list of columns to include.

inplace : *bool, default False*

Whether to modify the DataFrame rather than creating a new one.

Returns: **DataFrame or None**

DataFrame with NA entries dropped from it or None if `inplace=True`.

PANDAS OPERATIONS: CLEANING

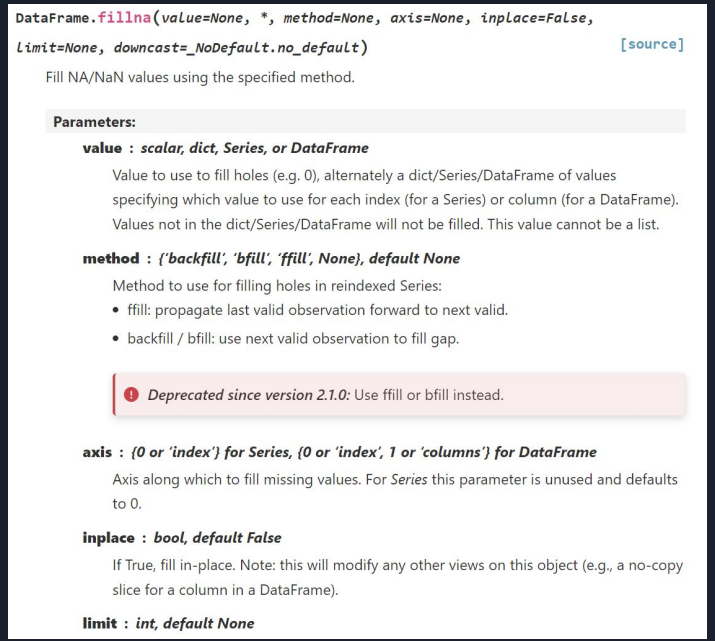
To fill null values in a data set we use:

- fillna()

Example to fill value with value of the previous observation:

- dataframeName = dataframeName.fillna(method='bfill')

We use fillna when we want to save observations and don't have many null values present. It's also useful if we already have a good idea of what to fill it with like a mean or median value



`DataFrame.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=_NoDefault.no_default)` [source]

Fill NA/NaN values using the specified method.

Parameters:

value : *scalar, dict, Series, or DataFrame*

Value to use to fill holes (e.g. 0), alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled. This value cannot be a list.

method : *{'backfill', 'bfill', 'ffill', None}, default None*

Method to use for filling holes in reindexed Series:

- ffill: propagate last valid observation forward to next valid.
- backfill / bfill: use next valid observation to fill gap.

Deprecated since version 2.1.0: Use ffill or bfill instead.

axis : *{0 or 'index'} for Series, {0 or 'index', 1 or 'columns'} for DataFrame*

Axis along which to fill missing values. For *Series* this parameter is unused and defaults to 0.

inplace : *bool, default False*

If True, fill in-place. Note: this will modify any other views on this object (e.g., a no-copy slice for a column in a DataFrame).

limit : *int, default None*

THANKS!

Questions on Campuswire or office hours

Office hours: Tue/Thu, 4-5 PM