PANDAS.

Date ....

~~Attributes & imp functionality~~

Pandas is built-on Numpy and Matplotlib.

(1) .head()
(2) .info()
(3) .describe() # numerical values.
(4) .values. # values [in the form of 2d Array]
(5) .columns # column names
(6) .index # rows names
(7) .shape # (row, col) tuple

SORTING and SUBSETTING

(1) .sort_values("col_name"), ascending = False)

.sort_values(["col_name1", "col_name2"]).

(2) dogs['name']

$\underbrace{\text{dogs}}_{Df}[\underbrace{[}_{\text{outer}}\underbrace{\text{'name', 'breed'}}_{\text{Inner square brackets are list of column names}}]]$

outer
Sbar subsetting the desired col" from Df.

dogs[['name', 'breed']], ascending = [True, False])

Spiral

filtering

condition

Date ....

dogs[dogs["height_cm"] > 50]

dogs[dogs["breed"] == "Labrador"] — α

Y M D

dogs[dogs["dob"] < "2015-01-01"]

is_lab = α

is_brown = dogs["color"] == "Brown"

dogs[is_lab & is_brown].

Subsetting using .isin()

.isin()

is_black_or_brown = dogs["color"].isin(["Black", "Brown"])

New Columns in pandas

dogs["height_m"] = dogs["height_cm"] / 100

print(dogs)

dogs["bmi"] = dogs["weight_kg"]/dogs["height_m"]**2

Date .... #2

Multiple Manipulations.

SUMMARY STATISTICS.

~~mean~~

1. dog["height_cm"].mean().
2. .median()
3. .mode()
4. .min()
5. .max()
6. .var()
7. .std()
8. .sum().
9. .quantile()
10. .agg()
11. .cumsum(), .cummax(), .cummin(), .cumprod

a list not a single number

We can also get the summary statistics for date values.

dogs["dob"].min().   dog["dob"].max()

AGG method

```
def pct30(column):
    column.quantile(0.3)
```

→ 30% of data on col^n.

dogs["weight_kg"].agg(pct30)

dogs[["weight_kg", "height_cm"]].agg(pct30)

def pct40(col): → 40%
    col. quantile(0.4)

Date ....

dog["weight"]. agg([pct30, pct40])

dog["weight"]. cumsum()

Dropping Duplicate values

vet_visits. drop_duplicates(subset = "name").

vd = vet_visits. drop_duplicates(subset = ['name', 'breed'])

print(vd).

To count.

unique_dogs["breed"]. value_counts().

unique_dogs["breed"]. value_counts(sort = True)

Proportions

unique_dogs["breed"]. value_counts(normalize = True)

Spiral

Groupby / Grouped Summary Statistics

Date ....

dogs.groupby("color")["weight_kg"].mean()

Df

the coloumn to groupby on

dogs.groupby("color")["weight_kg"].agg([min, max, sum])

dogs.groupby(["color", "breed"]).["kg"].mean()

dogs.groupby(["color", breed"]).[["kg", "cm"]].mean().

Pivot Tables

Groupby to pivot Table

dogs.groupby("color")["weight kg).mean()

dogs.pivot_table(values = "weight_kg", index = "color")

performance for sum statistics

to be group by

```
dogs.pivot_table(values = "weight_kg",
                 index = "color",
                 aggfunc = np.median)
```

Date ....

```
dogs.pivot_table(values = "weight_kg",
                 index = "color",
                 aggfunc = [np.mean, np.median])

dogs.groupby(["color", "breed"])["weight_kg"]
    .mean()

dogs.pivot_table(values = "weight_kg",
                 index = "color",
                 columns = "breed").
```

When there are missing values (NaNs) in pivot table

```
dogs.pivot_table(values = "weight_kg",
                 index = "color",
                 columns = "breed",
                 fill_value = 0,
                 margin = True)
```

mean of all except the filled_values / NaNs

Spiral

EXPLICIT

Date ....

INDEXES.

```
dogs_ind = dogs.set_index("name").
```

for Removing the index.

```
dogs_ind.reset_index().
```

Dropping an index.

```
dogs_ind.reset_index(drop=True)
```

↓

drop the ~~value~~ index
entirely removing
that index/columns.

WHY INDEXING?

Indexing makes subsetting easier

~~Index~~ values in the index don't need to be unique.

Multi-level indexes
a.k.a hierarchical indexes

Date ....

```
dogs_ind3 = dogs.set_index(["color", "breed"])
```

Subsetting with outer level

```
dog_ind3.loc[["Lab", "Chihuahua"]]
```

Subsetting with inner levels with a list of tuples.

```
dog_ind3.loc[('Lab', 'brown'), ('Chihua', 'Tan')]
```

Sorting by index values.

```
dogs_ind3.sort_index()
```

~~dogs_ind3~~

Controlling sort_index

```
dogs_ind3.sort_index(level=["color", "breed"],
                     ascending=[True, False]).
```

Now we have 3 problems

Date ....

1. Index values are just data and storing data in multiple forms confuses with my/our brain.

2. It also violates "tidy data" principles

normal tabular data

3. plus need to learn two syntaxes

4. I would suggest not to use but knowing about it is better. It helps in reading other peoples code some might find it useful.

SLICING AND SUBSETTING WITH .loc and .iloc

→ Sorting the index before slicing.

```
dogs_srt = dogs.set_index(["breed", "color
           sort_index()
```

Sp

Slicing at the outer index level

Date ....

dog_srt.loc["Chow chow": "Poodle"]

↓ it is included

specifying the index values

for inner index level : list of Tuples.

dog-srt.loc["Tan": "Grey"].

↓

wrong approach

dog-srt.loc[("Lab","Brown"): ("Schnauzer", Grey)].

Slicing columns

dog-srt.loc[:, "name": "height_cm"]

mixing both slicing for rows as well as columns

dogs_srt.loc[("Lab","brown"): ("Sch", grey), "name": "height"]

Spiral

Index Slicing can be ~~tempo~~
Imp when it comes to

Date ....

Dates.

dogs = dogs.set_index("DOB").sort_index()

So now, I can directly slice the data using dates // partial dates.

dogs.loc["2014":"2016"]
→ included.

dogs.iloc[2:5, 1:4]
→ final values are not included

Working with pivot tables

dog_height_bybreed_vs_color = dog_pack.pivot_table(
"height", index = "breed",
columns = "color")

dhbbvc.loc["chow chow":"poodle"]

dhbvc . mean (axis = "index")
→ rows

Spiral

Visualizing data.

Date ....

import matplotlib.pyplot as plt.

dog_pack["height_cm"].hist().

plt.show().

bins:

dog_pack["height_cm"].his(bins=5).

plt.show().

BAR PLOT.

avg_weight_by_breed = dog_pack.groupby("~~color~~ breed")["Weight_kg"].mean().

avg_weight_by_breed.plot(kind="bar",
title="Mean Weight by dog breed")

plt.show().

line plots-

Date ....

sully.plot(x= "date", y= "weight", kind= "line")

o/p. plt.show()

sully.plot (x= "date", y= "weight", kind= "line",
rot = 45)

plt.show()

the x axis labels rotating it 45° to read easily.

SCATTER PLOT {rel" b/w two numeric variables}

sully.dogpack(x= "height_cm", y= "weight",
kind ='scatter")

plt.show()

LAYERING PLOTS {one plot on top of other plot}.

dog-pack [dog-pack["sex"] == "F"]["height"].hist()

dog-pack [dog-pack ["sex"] == "M"]["height"].hist()

Spiral

plt. legend(["F", "M"])

Date ....

plt.show()

for Transparency

dog-pack[["sex"]=="F"]["height"].hist(alpha=0.7)

dog-pack[["sex"]=="M"]["height"].hist(alpha=0.7)

plt. legend(["F", "M"])

plt.show().

MISSING VALUES.

Detecting missing values

dogs. isna().

dogs.isna().any() → # In columns.

dogs.isna().sum()

Removing Missing Values

Date ....

dogs. dropna()

Replacing missing values

dogs. fillna(0) .

Creating dataframes:-

my_dict = { "key1" : value1,
"key2" : value2,
"key3 : value3 }.

my.dict = { "title" : "Charlotte's Web",
"author" : "E.B. White,"
"published" : 1952
}

pd. DataFrame(list_of_dicts).

DICTIONARY OF LISTS - By Column

pd. Dataframes.

Spiral

Reading & Writing CSV

Date ....

```
# new_dogs = pd.read_csv("new_dogs.csv")

new_dogs.to_csv("new_dogs_with_bmi.csv").
```