EDA

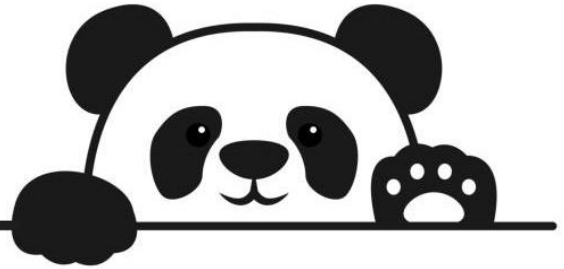
EDA - Exploratory Data Analysis

1. **Data Collection**: Gather the dataset from various sources (e.g., CSV, database, APIs).
2. **Data Cleaning**: Handle missing values, duplicates, and inconsistencies in the data.
3. **Data Transformation**: Convert data types, create new features, or normalize/scale data if needed.
4. **Data Visualization**: Use plots (histograms, boxplots, scatter plots, etc.) to understand distributions, relationships, and patterns.
5. **Statistical Summary**: Calculate summary statistics like mean, median, mode, standard deviation, and correlations.
6. **Outlier Detection**: Identify and handle outliers that might skew analysis or modeling.
7. **Feature Selection**: Identify important features and drop irrelevant ones.
8. **Correlation Analysis**: Analyze relationships between features using correlation matrices or heatmaps.

The scientific stack

- Special packages in python for EDA:
  - **NumPy**: implementation of multi-dimensional arrays in Python.
  - **pandas**: Tabular data manipulation (table data).
  - **matplotlib**: A cornerstone for data visualization in Python.
  - **seaborn**: Smart visualizations of tabular data.

Pandas

Panel +Data = 

- Library for dealing with tables.
- Install in conda: conda install pandas
- Import pandas as pd

Pandas

Pandas is a powerful library for data manipulation and analysis in Python. It provides data structures for efficiently storing large datasets and tools for working with them in a variety of ways. Some of the main features of pandas that are useful for EDA include:

- Reading and writing data: pandas provides functions for reading and writing data from a variety of formats, including CSV, Excel, and SQL databases.
- Handling missing values: pandas provides functions for identifying and handling missing values in a dataset.
- Data cleaning: pandas provides functions for cleaning and formatting data, such as converting data types and removing duplicates.
- Data visualization: pandas integrates with the Matplotlib library for data visualization, allowing you to create a variety of plots and charts to visualize your data.
- Data aggregation and grouping: pandas provides functions for performing aggregation and grouping operations on your data, such as calculating the mean or sum of a group of values.

Series

A 1-dimensional data structure, like a column in a table or a simple array, holding labeled data.

```
0    3.14159
1    2.71828
2    1.00000
3   -1.00000
4    0.00000
dtype: float64
```

```
0        Aurora
1         Belle
2    Cinderella
dtype: object
```

DataFrame

- The most useful part of pandas library.
- 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns.

```python
import pandas as pd
data = {'name': ['Shir', 'Alon'],
        'age': [20, 25],
        'height': [160, 175]}
df = pd.DataFrame(data)
df
```

	name	age	height
0	Shir	20	160
1	Alon	25	175

DataFrame basic tools

head()

- Present the first rows (default=5).
- Useful for quickly testing if your object has the right type of data in it.

```
df.head()
```

	longitude	latitude	housing_median_age	total_rooms
0	-114.31	34.19	15.0	5612.0
1	-114.47	34.40	19.0	7650.0
2	-114.56	33.69	17.0	720.0
3	-114.57	33.64	14.0	1501.0
4	-114.57	33.57	20.0	1454.0

DataFrame basic tools

describe()

- Print descriptive statistics such as count, mean, min, max, std and more.

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
count	17000.000000	17000.000000	17000.000000	17000.000000	17000.000000
mean	-119.562108	35.625225	28.589353	2643.664412	539.410824
std	2.005166	2.137340	12.586937	2179.947071	421.499452
min	-124.350000	32.540000	1.000000	2.000000	1.000000
25%	-121.790000	33.930000	18.000000	1462.000000	297.000000
50%	-118.490000	34.250000	29.000000	2127.000000	434.000000
75%	-118.000000	37.720000	37.000000	3151.250000	648.250000
max	-114.310000	41.950000	52.000000	37937.000000	6445.000000

DataFrame basic tools

info()

- Print a summary of the dataframe.
- Including information about the index dtype and columns, non-null values and memory usage.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           17000 non-null  float64
 1   latitude            17000 non-null  float64
 2   housing_median_age  17000 non-null  float64
 3   total_rooms         17000 non-null  float64
 4   total_bedrooms      17000 non-null  float64
 5   population          17000 non-null  float64
 6   households          17000 non-null  float64
 7   median_income       17000 non-null  float64
 8   median_house_value  17000 non-null  float64
dtypes: float64(9)
memory usage: 1.2 MB
```

Basic Series Tools

Methods:

S_name.head()	show the top (5 by default) results
S_name.tail()	show the end (5 by default) results
S_name.describe()	gives statistical data about the series

Attributes:

S_name.values –	a list of the values in the series
S_name.index –	information about the index
S_name.dtype –	the data type of the values in the series. “Object” is used for strings

Basic Series Tools

S_name.sum() – sum of elements
S_name.product() – multiplication
S_name.mean () – average

S_name.add(n)
S_name.sub(n)
S_name.div(n)
S_name.floordiv(n)

Will calculate each value

Usefull Series Tools

sort_values: Sorts the Series by its values, ascending or descending.

is_unique: Returns True if all elements in the Series are unique.

ndim: Returns the number of dimensions (always 1 for Series).

shape: Returns a tuple showing the number of elements (rows,) in the Series.

size: Returns the total number of elements in the Series.

Usefull Series Tools

sort_values: Sorts the Series by its values, ascending or descending.

is_unique: Returns True if all elements in the Series are unique.

ndim: Returns the number of dimensions (always 1 for Series).

shape: Returns a tuple showing the number of elements (rows,) in the Series.

size: Returns the total number of elements in the Series.