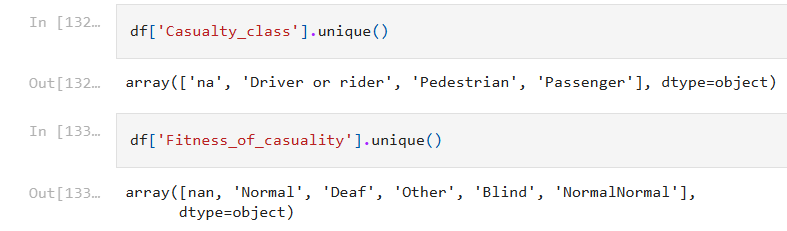
Week 1 Class Notes

Objective: This document includes class notes: **summaries** of everything we learned, as well as **questions and answers** during the sessions, as well as any **resources** and links shared by students or otherwise. Also **assignments** and due dates. Other stuff might be included as well.

# Session 5

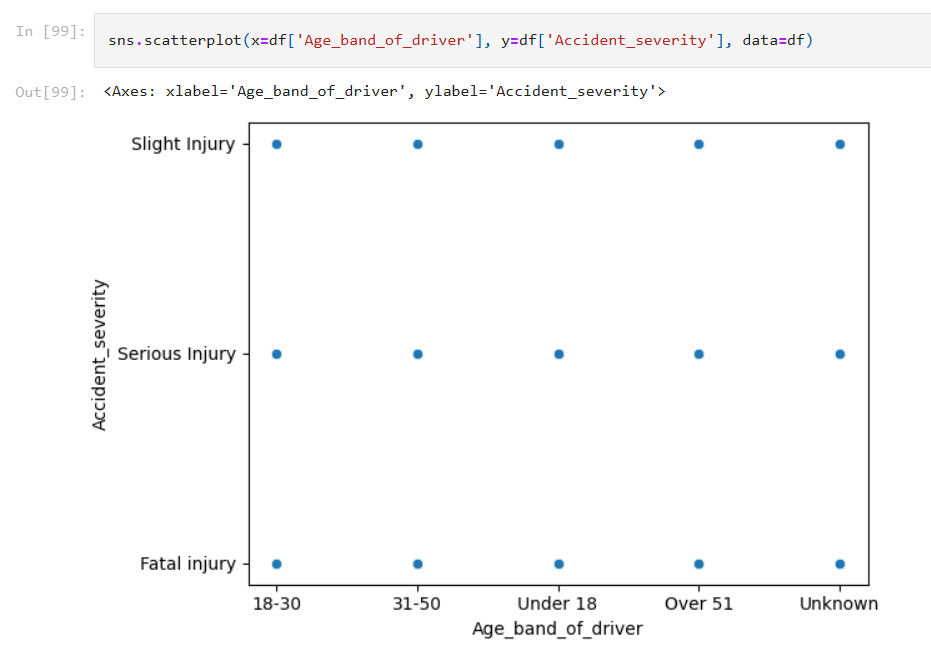
## The Good and Bad Examples

### Good Example. Inspecting categorical column values for NULL / Unknowns, one by one:



### 

### Bad Example. Using Scatter Plot for Categorical vs Categorical:



Better: x=Categorical, y=Numerical

plt.figure(figsize=(12, 6))

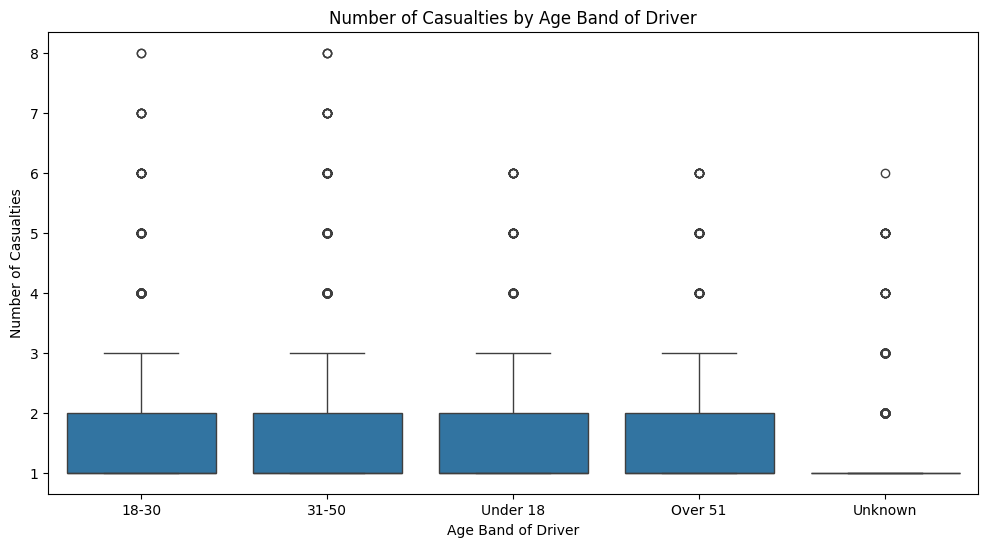
sns.boxplot(x='Age\_band\_of\_driver', y='Number\_of\_casualties', data=df1)

plt.title('Number of Casualties by Age Band of Driver')

plt.xlabel('Age Band of Driver')

plt.ylabel('Number of Casualties')

plt.show()



However, this ordinal categorical variable needs to be sorted!

# Session 4

Data Visualization.

**Patterns** emerge when we have lots of high-quality representative data. If we have too little, non-representative, or low-quality data, no useful observation can be made.

Good **Visualizations** make it easy for the eye to see patterns.

**Uni-variate**: a pattern may be seen in how **one variable** changes over the population.

* Histogram or Boxplot
* Bar chart

**Bi-variate**: a relation may be seen between **two variables**.

* Scatter plot
* Line chart

**Multi-variate**: some relations and patterns are specific to some **categories** only. Thus, we need to also visualize across categories.

* Heatmap
* Radar Chart

**Charts:**

* **Histogram**: distribution over a numerical variable (discrete or continuous)
* **Bar Chart**: distribution over a categorical variable (nominal or ordinal)
* **Pie**: distribution over a categorical variable (Bar Chart is almost always preferred)
* **Line**: showing changes over time (e.g., prices vs time)
* **Scatter plot**: numerical vs numerical
  + **Hexbin**: scatter plot with density
* **Heatmap**: matrix of relations

**Libraries**:

* **Matplotlib**: low-level plotting
* **Seaborn**: high-level statistical plots (uses matplotlib)
* Plotly, Bokeh, ..etc.: Interactive visuals (zoom, pan, hover)

# Session 3

Pandas Getting Started Tutorials (1 hour): <https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html>

This **Thursday**

* Two tasks (notebooks + dataset): you have 4 hours to finish them.
  + No Google. No ChatGPT. No Copilot.
    - you can use the documentation
    - you cannot use StackOverFlow
  + ~~No slides and no notebooks~~
* Last 1 hour: quiz is 60 questions. Topics are: Python + Introduction to Data Science (slides of this week)

**Boolean Mask**:

* Example:
  + mask = df[“age”] > 30
  + df[mask]
* Example (with negation):
  + mask = df.isnull()
  + df[~mask]

**Difference between** loc and iloc:

* iloc is positional (0 is first, 1 is second, regardless of index)
* loc is on the index (may coincide with position, may be datetime)

**Axis:**

* axis=0 means "apply the function to each column." (i.e., row-wise operation)
* axis=1 means "apply the function to each row." (i.e., column-wise operation)

**Handle missing values:**

* find missing values: df.isnull().sum()
* Drop columns (if proportion of missing data is higher than a threshold, say 60%)

df1 = df1.drop('Service\_year\_of\_vehicle', axis=1)

* Drop rows (else)

df1.dropna(inplace=True)

* Impute values (estimate with: mean, mode, regression, nearest neighbors, …etc.)

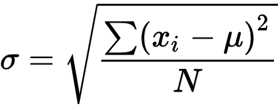
df1['Educational\_level\_missing'] = df1['Educational\_level'].isna()

df1[‘Educational\_level'].fillna('Unknown', inplace=True)

**Pandas Groupby Aggregations:** <https://pandas.pydata.org/docs/getting_started/intro_tutorials/06_calculate_statistics.html#aggregating-statistics-grouped-by-category>

* **Split** the data into groups
* **Apply** a function to each group independently
* **Combine** the results into a data structure

**Outlier detection methods (Z-score and IQR):**

* Z-Score: <https://www.investopedia.com/thmb/YYS3WeHvPn9AmDMqV2sABQKd2tU=/1500x0/filters:no_upscale():max_bytes(150000):strip_icc()/HtsEmpirical-1-d8d33ce1b8a64870a8cd0d67f1d8bf0d.png>
* Standard Deviation Formula:
* 
* Proportions in Normal distribution: <https://www.scribbr.de/wp-content/uploads/2023/01/Standard-normal-distribution.webp>
* Boxplot: <https://www.scribbr.com/wp-content/uploads/2022/01/interquartile-range.png>
* Boxplot on top of Histogram (Normal Distribution): <https://www.researchgate.net/publication/353410712/figure/fig1/AS:1048732418203648@1627048701501/Removal-of-outliers-using-IQR-method.png>

**How to choose between IQR and z-score?**

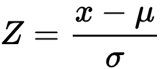
You may compare analysis result using both methods, and pick whichever works

* IQR is often more conservative (considers more data as outliers). IQR is usually a safe choice
* Z-score is a good fit for normally distributed data

**Handling Outliers**:

* Remove them
  + if you suspect they are noise/errors or
  + if influence the analysis badly
* Separate analysis
  + if you suspect there is a story behind it
* Transform the data non-linearly so they are no longer outliers

**Transformation**:

* **Numerical:**
  + **Scaling** to ranges: [0 to 1] OR [-1 to 1]
    - essential for ML
      * KNN
      * faster and more stable convergence in gradient descent
    - essential for fair comparison between values
      * The number $10 million in house prices is a lot
      * Similarly, the number 20 in number of rooms is a lot
      * After scaling, we find big values (regardless of scale) are close to 1
      * Also, we find small values (regardless of scale) are close to 0
    - MaxScaler
    - MinMaxScaler
  + **Normalization**
    - error: slides mention they are the same as scaling.
    - It is actually the change of distribution of the data using a non-linear transformation (e.g., exponential → normal distribution)
    - after normalization, less data may be considered outliers
    - Example PowerTransform in sklearn
  + **Standardization**
    - Linear transformation of data, usually, using z-score, such that values are centered, i.e., they have and .
    - 
    - Use: StandardScaler in sklearn
    - Often a required step in applying statistical techniques, such as z-score method of outlier removal, hypothesis testing, and others.
* **Categorical:**
  + **Integer encoding**: encode as integers.
    - LabelEncoder
    - Red, Green, Blue → 1, 2, 3
  + OrdinalEncoder: encode as integers (and preserves order)
    - Medium, Low, High → 2, 1, 3
  + **One-hot encoding**: vector of zeros, with only 1 one
    - pd.get\_dummies
    - Red, Green, Blue → [

[1, 0, 0],

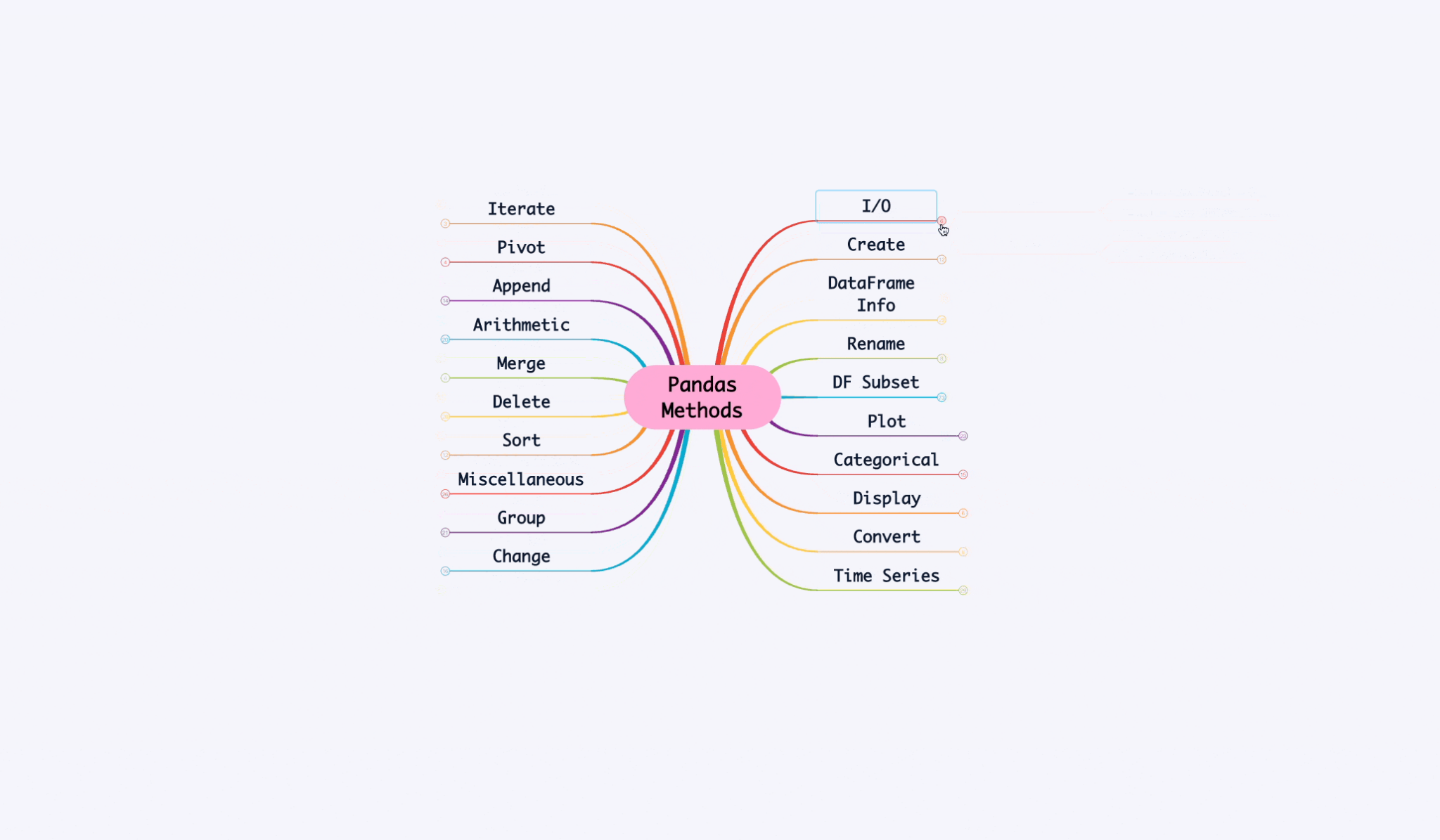
[0, 1, 0],

[0, 0, 1]

]

**Useful References:**

* Pandas CookBook: <https://pandas.pydata.org/docs/user_guide/cookbook.html>
* Pandas MindMap: <https://xmind.ai/share/ugVH30g4>

[](https://xmind.ai/share/ugVH30g4)

# Session 2

Python Tutorials: <https://github.com/HassanAlgoz/python>

T5 Courses:

1. Introduction to Data Science (EDA)
   1. Pandas
2. Foundation of Machine Learning (Predictive Analytics)
   1. Scikit-learn
3. Advanced Machine Learning (more algorithms)
   1. Scikit-learn (carrot)
4. Deep Learning (AI: Neural Networks)
   1. Keras (TensorFlow)
5. Time series Forecasting (Recurrent Neural Networks)
6. Computer Vision (Convolutional Neural Networks)
7. Natural Language Processing (traditional)
   1. Spacy, NLTK, scikit-learn
8. Transformers (Architecture NN)
   1. HuggingFace Transformers
9. Large Language Models (LLM)
10. Advanced Deep Learning
11. Machine Learning Operations (MLOps)
    1. Docker, Flask
12. Capstone Project

Everyweek, on Thursday we have: Project (task) + Quiz (theory: MCQ).

Distributions:

* Central Tendency, Variations, and Skewness
* Comparing distributions
  + (Skewed) Distribution of Death by Age: [https://miro.medium.com/v2/resize:fit:1400/1\*jW9Zwvr5pPmVZcdQ72Gtiw.jpeg](https://miro.medium.com/v2/resize:fit:1400/1*jW9Zwvr5pPmVZcdQ72Gtiw.jpeg)
  + (Normal, Same Mean) Distribution of IQ Scores (Male and Female): <https://qph.cf2.quoracdn.net/main-qimg-e3a75fc210c089f8562259e7a91cc646.webp>
  + (Normal, Different Mean) Distribution of Heights (Male, Female and NBA Players): <https://distributionofthings.com/human-height/>

**Correlation** does not imply causation. Correlation is a requirement for establishing causation, but not the only one.

* Example: Coffee Consumption and IQ
* Example: Drug X and Memory

**Selection Bias** example (n=10 shows correlation, n=100 shows no correlation!)

* Sample size (bigger is better; n=30 is minimum)
* Bias in selection
  + I select the people who are enrolled in a memorization program; when I want to prove that it is my drug is effective, without mentioning the program
* Diversity in characteristics related to the variable
  + Ineffective diversity: vary eye-colors, hair-shape, height, or weight; when the measured factor is IQ-level, or whether they like Hot or Cold coffee.
  + Effective diversity: weight, and height, when determining whether a person is athletic or not.

**Randomized Controlled Trials (RCTs)** are the golden standard in medical studies.

**Placebo** is given for the *control group*, whereas the *treatment group* is given the actual drug.

Pandas getting started tutorials: <https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html>

# Session 1

In this session, we discussed things like:

**Data Capture** the fact that we need to measure things, if we want to make use of them. If we don’t know what to measure, or don’t capture that, we will never be able to predict or derive insights in the first place.

**Data Ingestion** the ETL (Extract: get it from sources, Transform: normalize it, Load: store it in a destination).

**Data Source**. Could come from sensors, surveys, websites, and all kinds of sources.

An **API** (Application Programming Interface), is one which is used by other programs. Contrary to UI (User Interface) which is used by humans.

When a program (e.g., Twitter’s website) provides an API, they are serving your program. You can ask for the data you want, or even, other services, using their well-curated API.

**Web Scraping**, on the other hand, is having a program fetch the data directly from the webpage. You see, websites are built with HTML. If you read the HTML, you read the webpage. But, it is often hard to get your hands on the data you need in this way.

Using APIs is like using the front door to access the data. Using Web Scrapingis like using the window, it is awkward, but sometimes necessary.

Direct database access gives the most control and visibility. However, we rarely have this access. And thus, need to integrate with third-party services through their APIs.

Platforms (like Twitter) offer their API and charge money for it. Whether on a per-request basis, or on the amount of data, or other considerations.

**Data Processing** includes cleaning the data of missing, erroneous, or duplicate data.

It also includes **Normalizing** the data (transforming it to a common format).

**Data Storage**: many database types for structured and unstructured data.

**Cloud:** is about having a managed, easy to use, hardware as a service. Think of it like an Operating System (OS) which you can ask for **resources** (either **Compute Power** or **Storage Space**), and not worry about what machines, or how many machines, it needs to allocate for you. It also abstracts away maintenance, backups, scaling, and other details.

**Exploratory Data Analysis**; aims at describing things, rather, than judging.

**Predictive Analytics** is about filling the missing information; based on known details. It is usually about the future. It can be general or detailed filling.

Git: Version Control System (VCS) -- Client

GitHub: Platform -- Server