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

**DATA WRANGLING**

Data wrangling is the process of organizing, cleaning, and transforming raw data into a usable format. It's also known as data cleaning, scrubbing, or remediation.

Why is data wrangling important?

* It ensures data is reliable and complete
* It helps make data more consumable and useful for analytics or machine learning
* It helps ensure that data is accurate and ready for meaningful analysis

Steps of data wrangling

1. **Discovery**: The initial step in the data wrangling process
2. **Structuring**: Processing and restructuring raw data to make it useful
3. **Cleaning**: Removing errors, outliers, and inconsistencies from the data
4. **Enriching**: Adding new information to existing data sets to enhance their value
5. **Validating**: Verifying the consistency and quality of the data
6. **Publishing**: Delivering the final output of the wrangling efforts

Examples of data wrangling

* Removing duplicate entries
* Standardizing inputs, such as phone numbers
* Deleting empty cells or rows
* Merging third-party data, such as weather information, with shipment data



Data wrangling is the process of transforming and structuring data from one raw form into a desired format with the intent of improving data quality and making it more consumable and useful for analytics or machine learning.

Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics.

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*careof 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
5. **Other:** After dealing with the raw dataset with the above functionalities we get an efficient dataset as per our requirements and then it can be used for a required purpose like data analyzing, [machine learning,](https://www.geeksforgeeks.org/getting-started-machine-learning/) [data visualization](https://www.geeksforgeeks.org/what-is-data-visualization-and-why-is-it-important/), [model training](https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/) etc.



**DATA EXPLORATION**

Data exploration definition: Data exploration refers to the initial step in data analysis in which data analysts use data visualization and statistical techniques to describe dataset characterizations, such as size, quantity, and accuracy, in order to better understand the nature of the data.

Data exploration is the process of analyzing data to find patterns and relationships. It's the first step in data preparation and analysis.

Steps in data exploration

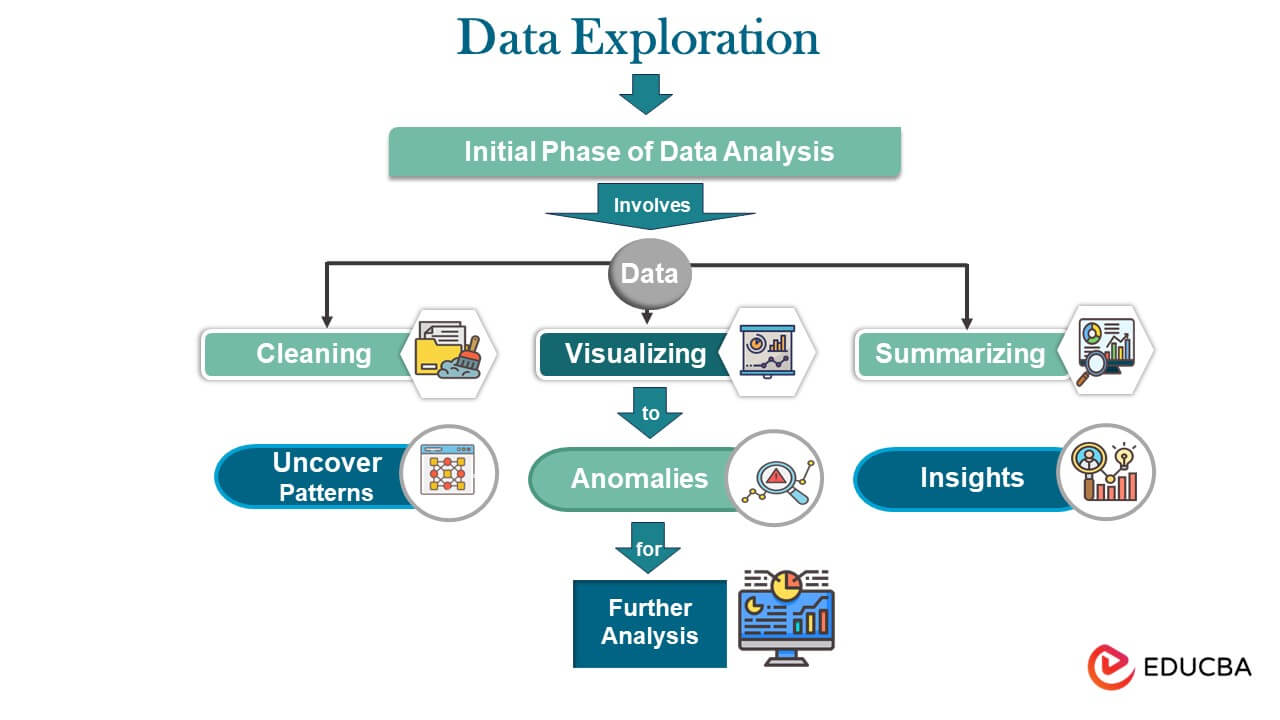
1. **Data collection**: Collect data from different sources.
2. **Data cleaning**: Remove irrelevant or incorrect data points.
3. **Data profiling**: Get familiar with the data by generating summary statistics.
4. **Data visualization**: Use tools like histograms, scatter plots, and box plots to identify patterns, correlations, and outliers.
5. **Exploratory data analysis (EDA)**: Analyze the data to find underlying structures, test hypotheses, and identify important variables.

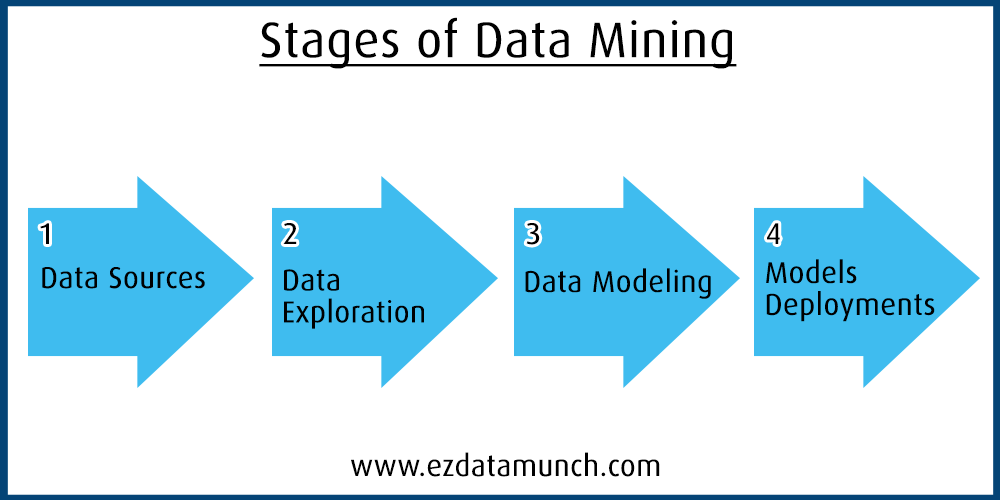
Benefits of data exploration

* Helps identify outliers, errors, and fraud
* Helps detect relationships between variables
* Helps understand the nature of the data
* Helps decide which model or algorithm to use in subsequent steps

Tools used in data exploration

* **Matplotlib**: A tool for analyzing large sets of data quickly
* **Descriptive statistics**: A statistical analysis that provides information about the distribution, central tendencies, and variability of the data
* **Correlation analysis**: A technique that reveals the degree of association in a set of data





**What is Data Exploration?**

Data exploration is the initial step in data analysis where you dive into a dataset to get a feel for what it contains. It's like detective work for your data, where you uncover its characteristics, patterns, and potential problems.

**Why is it Important?**

Data exploration plays a crucial role in data analysis because it helps you **uncover hidden gems** within your data. Through this initial investigation, you can start to identify:

* **Patterns and Trends:** Are there recurring themes or relationships between different data points?
* **Anomalies:** Are there any data points that fall outside the expected range, potentially indicating errors or outliers?

**How Data Exploration Works?**

1. **Data Collection:** Data exploration commences with collecting data from diverse sources such as databases, [APIs](https://www.geeksforgeeks.org/what-is-an-api), or through web scraping techniques. This phase emphasizes recognizing data formats, structures, and interrelationships. Comprehensive data profiling is conducted to grasp fundamental statistics, distributions, and ranges of the acquired data.
2. **Data Cleaning:** Integral to this process is the rectification of outliers, inconsistent data points, and addressing missing values, all of which are vital for ensuring the reliability of subsequent analyses. This step involves employing methodologies like standardizing data formats, identifying outliers, and imputing missing values. Data organization and transformation further streamline data for analysis and interpretation.
3. **Exploratory Data Analysis (EDA):**This [EDA](https://www.geeksforgeeks.org/what-is-exploratory-data-analysis) phase involves the application of various statistical tools such as[box plots,](https://www.geeksforgeeks.org/box-plot) [scatter plots](https://www.geeksforgeeks.org/scatter-plots-in-r-language),[histograms,](https://www.geeksforgeeks.org/histograms-in-r-language) and distribution plots. Additionally, correlation matrices and descriptive statistics are utilized to uncover links, patterns, and trends within the data.
4. **Feature Engineering:** [Feature engineering](https://www.geeksforgeeks.org/what-is-feature-engineering)focuses on enhancing prediction models by introducing or modifying features. Techniques like data normalization, scaling, encoding, and creating new variables are applied. This step ensures that features are relevant and consistent, ultimately improving model performance.
5. **Model Building and Validation:** During this stage, preliminary models are developed to test hypotheses or predictions. [Regression](https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning), classification, or [clustering techniques](https://www.geeksforgeeks.org/different-types-clustering-algorithm) are employed based on the problem at hand. [Cross-validation methods](https://www.geeksforgeeks.org/cross-validation-machine-learning) are used to assess model performance and generalizability.

**Steps involved in Data Exploration**

Data exploration is an iterative process, but there are generally some key steps involved:

**Data Understanding**

* **Familiarization:** Get an overview of the data format, size, and source.
* **Variable Identification:** Understand the meaning and purpose of each variable in the dataset.

**Data Cleaning**

* **Identifying Missing Values:** Locate and address missing data points strategically (e.g., removal, imputation).
* **Error Correction:** Find and rectify any inconsistencies or errors within the data.
* **Outlier Treatment:** Identify and decide how to handle outliers that might skew the analysis.

[**Exploratory Data Analysis (EDA)**](https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/)

* **Univariate Analysis:** Analyze individual variables to understand their distribution (e.g., histograms, boxplots for numerical variables; frequency tables for categorical variables).
* **Bivariate Analysis:** Explore relationships between two variables using techniques like scatterplots to identify potential correlations.

**Data Visualization**

* **Creating Visualizations:** Use charts and graphs (bar charts, line charts, heatmaps) to effectively communicate patterns and trends within the data.
* **Choosing the Right Charts:** Select visualizations that best suit the type of data and the insights you're looking for.

**Iteration and Refinement**

* **Iterate:**As you explore, you may need to revisit previous steps.
* **Refinement:**New discoveries might prompt you to clean further, analyze differently, or create new visualizations.

**Importance of Data Exploration**

* **Trend Identification and Anomaly Detection:**Data exploration helps uncover underlying trends and patterns within datasets that might otherwise remain unnoticed. It facilitates the identification of anomalies or outliers that could significantly impact decision-making processes. Detecting these trends early can be critical for businesses to adapt, strategize, or take preventive measures.
* **Ensuring Data Quality and Integrity:** It is essential for spotting and fixing problems with data quality early on. Through the resolution of missing values, outliers, or discrepancies, data exploration guarantees that the information used in later studies and models is accurate and trustworthy. This enhances the general integrity and reliability of the conclusions drawn.
* **Revealing Latent Insights:** Often, valuable insights might be hidden within the data, not immediately apparent. Through visualization and statistical analysis, data exploration uncovers these latent insights, providing a deeper understanding of relationships between variables, correlations, or factors influencing certain outcomes.
* **Foundation for Advanced Analysis and Modeling:** Data exploration sets the foundation for more sophisticated analyses and modeling techniques. It helps in selecting relevant features, understanding their importance, and refining them for optimal model performance. Without a thorough exploration, subsequent modeling efforts might lack depth or accuracy.
* **Supporting Informed Decision-Making:** By revealing patterns and insights, data exploration empowers decision-makers with a clearer understanding of the data context. This enables informed and evidence-based decision-making across various domains such as marketing strategies, risk assessment, resource allocation, and operational efficiency improvements.
* **Adaptability and Innovation:**In a rapidly changing environment, exploring data allows organizations to adapt and innovate. Identifying emerging trends or changing consumer behaviors through data exploration can be crucial in staying competitive and fostering innovation within industries.
* **Risk Mitigation and Compliance:**In sectors like finance or healthcare, data exploration aids in risk mitigation by identifying potential fraud patterns or predicting health risks based on patient data. It also contributes to compliance efforts by ensuring data accuracy and adhering to regulatory requirements.

**Example of Data Exploration**

* **Finance:** Detecting fraudulent activities through anomalous transaction patterns. In the financial domain, data exploration plays a pivotal role in safeguarding institutions against fraudulent practices by meticulously scrutinizing transactional data. Here's an elaborate exploration:
* **Anomaly Detection Techniques:**Data exploration employs advanced anomaly detection algorithms to sift through vast volumes of transactional data. This involves identifying deviations from established patterns, such as irregular transaction amounts, unusual frequency, or unexpected locations of transactions.
* **Behavioral Analysis:**By analyzing historical transactional behaviors, data exploration discerns normal patterns from suspicious activities. This includes recognizing deviations from regular spending habits, unusual timeframes for transactions, or atypical transaction sequences.
* **Pattern Recognition:**Through sophisticated data exploration methods, financial institutions can uncover intricate patterns that might indicate fraudulent behavior. This could involve recognizing specific sequences of transactions, correlations between seemingly unrelated accounts, or unusual clusters of transactions occurring concurrently.
* **Machine Learning Models:**Leveraging machine learning models as part of data exploration enables the creation of predictive fraud detection systems. These models, trained on historical data, can continuously learn and adapt to evolving fraudulent tactics, enhancing their accuracy in identifying suspicious transactions.
* **Real-time Monitoring:**Data exploration facilitates the development of real-time monitoring systems. These systems analyze incoming transactions as they occur, swiftly flagging potentially fraudulent activities for immediate investigation or intervention.
* **Regulatory Compliance:**Data exploration aids in ensuring regulatory compliance by detecting and preventing fraudulent activities that might violate financial regulations. This helps financial institutions adhere to compliance standards while safeguarding against financial crimes.

**Benefits of Data Exploration**

* **Fraud Mitigation:** By proactively identifying and addressing fraudulent activities, financial institutions can minimize financial losses and protect their customers' assets.
* **Enhanced Security:** Data exploration enhances the security infrastructure of financial systems, bolstering confidence among customers and stakeholders.
* **Operational Efficiency:** Identifying and mitigating fraud through data exploration streamlines operational processes, reducing the resources expended on investigating and rectifying fraudulent incidents.

**Applications of Data Exploration**

**Business Intelligence and Analytics:** Companies across sectors can apply data exploration techniques to extract insights from their datasets. For instance:

* **Retail:**Analyzing sales data to optimize inventory management and forecast demand.
* **Manufacturing:**Identifying production inefficiencies or predicting equipment failures through data analysis.
* **Marketing:**Understanding customer behavior for targeted and personalized marketing campaigns.

**Healthcare and Medicine:** Utilizing data exploration methods in healthcare can lead to various applications:

* **Disease Prediction:**Analyzing patient data to predict and prevent diseases based on risk factors.
* **Treatment Optimization:** Identifying effective treatments or therapies by analyzing patient response data.

**Financial Sector:** Besides detecting fraudulent activities, data exploration in finance includes:

* **Risk Assessment:**Assessing investment risks by analyzing market data and economic indicators.
* **Portfolio Management:** Optimizing investment portfolios based on historical performance and market trends.

**E-commerce and Customer Experience:**Data exploration techniques play a crucial role in:

* **Customer Personalization:** Analyzing browsing and purchasing patterns to personalize recommendations.
* **Supply Chain Optimization:**Optimizing inventory and logistics by analyzing demand and supply data.

**Predictive Maintenance in Industries:** Using data exploration in industries to:

* **Avoid Downtime:**Predict equipment failures by analyzing machine sensor data in real-time.
* **Optimize Maintenance:** Schedule maintenance tasks based on predictive analytics, reducing operational costs.

**Risk Management and Compliance:** Across sectors like finance, healthcare, and more:

* **Compliance Checks:**Ensuring adherence to regulatory standards by identifying data discrepancies or anomalies.
* **Fraud Prevention:**Beyond finance, detecting fraudulent activities in insurance or cybersecurity domains using similar data exploration techniques.

**DATA CLEANING**

Data cleaning is the process of identifying and fixing errors, inconsistencies, or missing values in data. It's also known as data cleansing or data scrubbing.

Why is data cleaning important?

* It improves the quality and reliability of data
* It ensures that only high-quality data is used in analysis
* It helps to make security decisions

What does data cleaning involve?

* Removing duplicates or invalid entries
* Filling in missing values
* Standardizing formats
* Cross-checking information
* Adding more details
* Converting data types
* Ensuring overall consistency
* Keeping data in a unified form

How is data cleaning done?

* Data cleaning can be done interactively using data wrangling tools
* It can also be done through batch processing using scripts or a data quality firewall

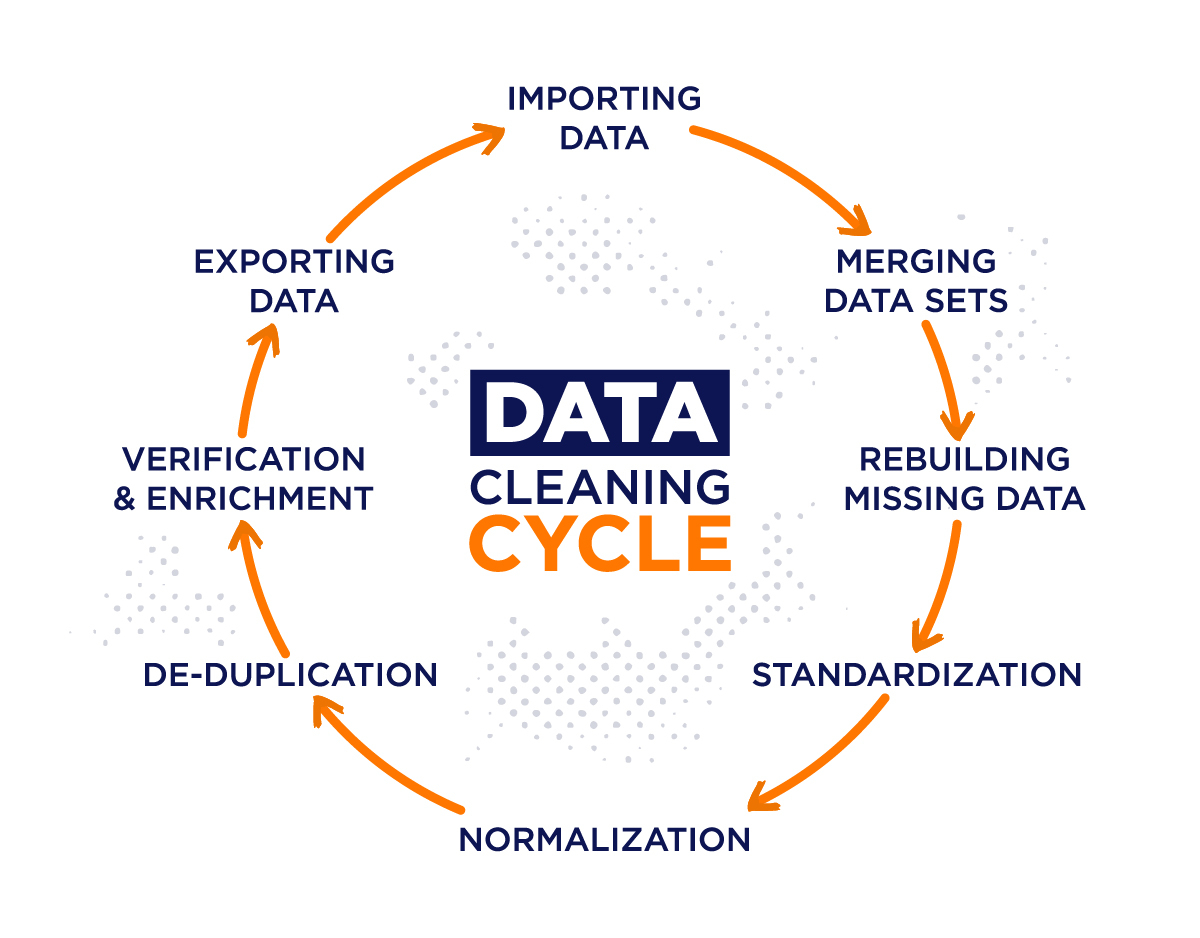
What tools can be used for data cleaning?

* **DataCleaner**: An open-source data quality and data cleansing software
* **OpenRefine**: An open-source (free) data cleaning tool that allows users to convert data between formats

Some common techniques for data cleaning include:

* Removing duplicate data
* Eliminating unnecessary data
* Ensuring overall consistency
* Normalization, which is the process of scaling or transforming data to a common range or format.

What is data cleaning? Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.



**ML | Overview of Data Cleaning**

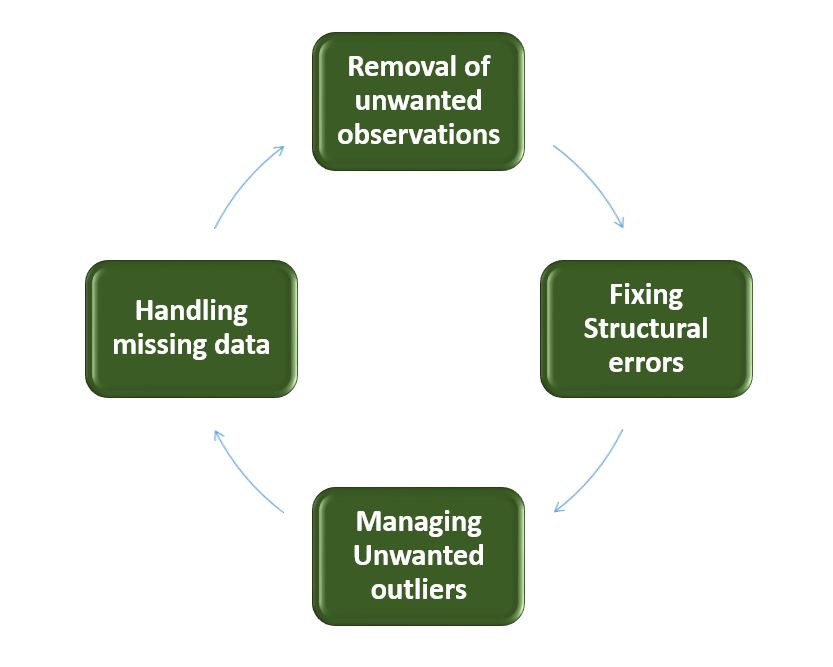
Data cleaning is a important step in the [machine learning (ML)](https://www.geeksforgeeks.org/machine-learning/) pipeline as it involves **identifying and removing any missing duplicate or irrelevant data**. The goal of data cleaning is to **ensure that the data is accurate, consistent and free of errors as raw data is often noisy, incomplete and inconsistent which can negatively impact the accuracy of model and its reliability of insights derived from it.** Professional data scientists usually invest a large portion of their time in this step because of the belief that

***“Better data beats fancier algorithms”***

Clean datasets also helps in [EDA](https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/) that enhance the interpretability of data so that right actions can be taken based on insights.

**How to Perform Data Cleanliness?**

The process begins by thorough understanding data and its structure to identify issues like missing values, duplicates and outliers. Performing data cleaning involves a systematic process to identify and remove errors in a dataset. The following are essential steps to perform data cleaning.



*Data Cleaning*

* **Removal of Unwanted Observations**: Identify and remove irrelevant or redundant (unwanted) observations from the dataset. This step involves analyzing data entries for duplicate records, irrelevant information or data points that do not contribute to analysis and prediction. Removing them from dataset helps reducing noise and improving the overall quality of dataset.
* **Fixing Structure errors:**Address structural issues in the dataset such as inconsistencies in data formats or variable types. Standardize formats ensure uniformity in data structure and hence data consistency.
* **Managing outliers:**Outliers are those points that deviate significantly from dataset mean. Identifying and managing outliers significantly improve model accuracy as these extreme values influence analysis. Depending on the context decide whether to remove outliers or transform them to minimize their impact on analysis.
* **Handling Missing Data:** To handle missing data effectively we need to impute missing values based on statistical methods, removing records with missing values or employing advanced imputation techniques. Handling missing data helps preventing biases and maintaining the integrity of data.

Throughout the process documentation of changes is crucial for transparency and future reference. Iterative validation is done to test effectiveness of the data cleaning resulting in a refined dataset and can be used for meaningful analysis and insights.

**Python Implementation for Database Cleaning**

Let’s understand each step for Database Cleaning, using [titanic dataset](https://media.geeksforgeeks.org/wp-content/uploads/20250114171103408125/Titanic-Dataset.csv). Below are the necessary steps:

* Import the necessary libraries
* Load the dataset
* Check the data information using df.info()

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

*# Load the dataset*

df = pd.read\_csv('titanic.csv')

df.head()

**Output**:

PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked

0 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S

1 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1 0 PC 17599 71.2833 C85 C

2 3 1 3 Heikkinen, Miss. Laina female 26.0 0 0 STON/O2. 3101282 7.9250 NaN S

3 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1 0 113803 53.1000 C123 S

4 5 0 3 Allen, Mr. William Henry male 35.0 0 0 373450 8.0500 NaN S

**Data Inspection and Exploration**

Let’s first understand the data by inspecting its structure and identifying missing values, outliers and inconsistencies and check the duplicate rows with below python code:

df.duplicated()

**Output**:

0 False

1 False

2 False

3 False

4 False

...

886 False

887 False

888 False

889 False

890 False

Length: 891, dtype: bool

**Check the data information using df.info()**

df.info()

**Output**:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

From the above data info we can see that Age and Cabin have an **unequal number of counts**. And some of the columns are categorical and have data type objects and some are integer and float values.

**Check the Categorical and Numerical Columns.**

*# Categorical columns*

cat\_col = [col **for** col **in** df.columns **if** df[col].dtype == 'object']

print('Categorical columns :',cat\_col)

*# Numerical columns*

num\_col = [col **for** col **in** df.columns **if** df[col].dtype != 'object']

print('Numerical columns :',num\_col)

**Output**:

Categorical columns : ['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked']

Numerical columns : ['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare']

**Check the total number of Unique Values in the Categorical Columns**

df[cat\_col].nunique()

**Output**:

Name 891

Sex 2

Ticket 681

Cabin 147

Embarked 3

dtype: int64

**Removal of all Above Unwanted Observations**

Duplicate observations most frequently arise during data collection and Irrelevant observations are those that don’t actually fit with the specific problem that we’re trying to solve.

* Redundant observations alter the efficiency to a great extent as the data repeats and may add towards the correct side or towards the incorrect side, therefore producing useless results.
* Irrelevant observations are any type of data that is of no use to us and can be removed directly.

**Now we have to make a decision according to the subject of analysis which factor is important for our discussion.**

As we know our machines don’t understand the text data. So we have to either drop or convert the categorical column values into numerical types. Here we are dropping the Name columns because the Name will be always unique and it hasn’t a great influence on target variables. For the ticket, Let’s first print the 50 unique tickets.

df['Ticket'].unique()[:50]

**Output**:

array(['A/5 21171', 'PC 17599', 'STON/O2. 3101282', '113803', '373450',

'330877', '17463', '349909', '347742', '237736', 'PP 9549',

'113783', 'A/5. 2151', '347082', '350406', '248706', '382652',

'244373', '345763', '2649', '239865', '248698', '330923', '113788',

'347077', '2631', '19950', '330959', '349216', 'PC 17601',

'PC 17569', '335677', 'C.A. 24579', 'PC 17604', '113789', '2677',

'A./5. 2152', '345764', '2651', '7546', '11668', '349253',

'SC/Paris 2123', '330958', 'S.C./A.4. 23567', '370371', '14311',

'2662', '349237', '3101295'], dtype=object)

From the above tickets, we can observe that it is made of two like first values ‘A/5 21171’ is joint from of ‘A/5’ and  ‘21171’ this may influence our target variables. It will the case of **Feature Engineering**. where we derived new features from a column or a group of columns. In the current case, we are dropping the “Name” and “Ticket” columns.

**Drop Name and Ticket Columns**

df1 = df.drop(columns=['Name','Ticket'])

df1.shape

**Output**:

(891, 10)

**Handling Missing Data**

Missing data is a common issue in real-world datasets and it can occur due to various reasons such as human errors, system failures or data collection issues. Various techniques can be used to handle missing data, such as imputation, deletion or substitution.

Let’s check the missing values columns-wise for each row using df.isnull() it checks whether the values are null or not and gives returns boolean values and sum() will sum the total number of null values rows and we divide it by the total number of rows present in the dataset then we multiply to get values in i.e per 100 values how much values are null.

round((df1.isnull().sum()/df1.shape[0])\*100,2)

**Output**:

PassengerId 0.00

Survived 0.00

Pclass 0.00

Sex 0.00

Age 19.87

SibSp 0.00

Parch 0.00

Fare 0.00

Cabin 77.10

Embarked 0.22

dtype: float64

We cannot just ignore or remove the missing observation. They must be handled carefully as they can be an indication of something important.

* The fact that the value was missing may be informative in itself.
* In the real world we often need to make predictions on new data even if some of the features are missing!

As we can see from the above result that Cabin has 77% null values and Age has 19.87% and Embarked has 0.22% of null values.

So, it’s not a good idea to fill 77% of null values. So we will drop the Cabin column. Embarked column has only 0.22% of null values so, we drop the null values rows of Embarked column.

df2 = df1.drop(columns='Cabin')

df2.dropna(subset=['Embarked'], axis=0, inplace=**True**)

df2.shape

**Output**:

(889, 9)

Imputing the missing values from past observations.

* Again “missingness” is almost informative in itself and we should tell our algorithm if a value was missing.
* Even if we build a model to impute our values we’re not adding any real information. we’re just reinforcing the patterns already provided by other features. We can use **Mean imputation** or **Median imputations** for the case.

**Note:**

* Mean imputation is suitable when the data is normally distributed and has no extreme outliers.
* Median imputation is preferable when the data contains outliers or is skewed.

*# Mean imputation*

df3 = df2.fillna(df2.Age.mean())

*# Let's check the null values again*

df3.isnull().sum()

**Output**:

PassengerId 0

Survived 0

Pclass 0

Sex 0

Age 0

SibSp 0

Parch 0

Fare 0

Embarked 0

dtype: int64

**Handling Outliers**

Outliers are extreme values that deviate significantly from the majority of the data. They can negatively impact the analysis and model performance. Techniques such as clustering, interpolation or transformation can be used to handle outliers.

To check the outliers we generally use a [box plo](https://www.geeksforgeeks.org/box-plot/)t. A box plot is a graphical representation of a dataset’s distribution. It shows a variable’s median, quartiles and potential outliers. The line inside the box denotes the median while the box itself denotes the [interquartile range (IQR)](https://www.geeksforgeeks.org/interquartile-range-iqr/). The box plot extend to the most extreme non-outlier values within 1.5 times the IQR. Individual points beyond the box are considered potential outliers. A box plot offers an easy-to-understand overview of the range of the data and makes it possible to identify outliers or skewness in the distribution.

**Let’s plot the box plot for Age column data.**

**import** **matplotlib.pyplot** **as** **plt**

plt.boxplot(df3['Age'], vert=**False**)

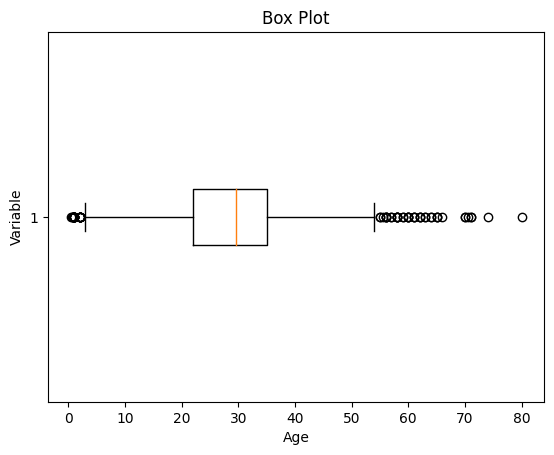
plt.ylabel('Variable')

plt.xlabel('Age')

plt.title('Box Plot')

plt.show()

**Output**:



*Box Plot*

As we can see from the above Box and whisker plot, Our age dataset has outliers values. The values less than 5 and more than 55 are outliers.

*# calculate summary statistics*

mean = df3['Age'].mean()

std = df3['Age'].std()

*# Calculate the lower and upper bounds*

lower\_bound = mean - std\*2

upper\_bound = mean + std\*2

print('Lower Bound :',lower\_bound)

print('Upper Bound :',upper\_bound)

*# Drop the outliers*

df4 = df3[(df3['Age'] >= lower\_bound)

& (df3['Age'] <= upper\_bound)]

**Output**:

Lower Bound : 3.705400107925648

Upper Bound : 55.578785285332785

Similarly, we can remove the outliers of the remaining columns.

**Data Transformation**

Data transformation involves converting the data from one form to another to make it more suitable for analysis. Techniques such as normalization, scaling or encoding can be used to transform the data.

**Data validation and verification**

Data validation and verification involve ensuring that the data is accurate and consistent by comparing it with external sources or expert knowledge.

For the machine learning prediction we separate independent and target features. Here we will consider only **‘Sex’ ‘Age’ ‘SibSp’, ‘Parch’ ‘Fare’ ‘Embarked’**only as the independent features and **Survived**as target variables because PassengerId will not affect the survival rate.

X = df3[['Pclass','Sex','Age', 'SibSp','Parch','Fare','Embarked']]

Y = df3['Survived']

**Data formatting**

Data formatting involves converting the data into a standard format or structure that can be easily processed by the algorithms or models used for analysis. Here we will discuss commonly used data formatting techniques i.e. Scaling and Normalization.

**Scaling**

* Scaling involves transforming the values of features to a specific range. It maintains the shape of the original distribution while changing the scale.
* Particularly useful when features have different scales, and certain algorithms are sensitive to the magnitude of the features.
* Common scaling methods include Min-Max scaling and Standardization (Z-score scaling).

**Min-Max Scaling**: Min-Max scaling rescales the values to a specified range, typically between 0 and 1. It preserves the original distribution and ensures that the minimum value maps to 0 and the maximum value maps to 1.

**from** **sklearn.preprocessing** **import** MinMaxScaler

*# initialising the MinMaxScaler*

scaler = MinMaxScaler(feature\_range=(0, 1))

*# Numerical columns*

num\_col\_ = [col **for** col **in** X.columns **if** X[col].dtype != 'object']

x1 = X

*# learning the statistical parameters for each of the data and transforming*

x1[num\_col\_] = scaler.fit\_transform(x1[num\_col\_])

x1.head()

**Output**:

Pclass Sex Age SibSp Parch Fare Embarked

0 1.0 male 0.271174 0.125 0.0 0.014151 S

1 0.0 female 0.472229 0.125 0.0 0.139136 C

2 1.0 female 0.321438 0.000 0.0 0.015469 S

3 0.0 female 0.434531 0.125 0.0 0.103644 S

4 1.0 male 0.434531 0.000 0.0 0.015713 S

**Standardization (Z-score scaling):**Standardization transforms the values to have a mean of 0 and a standard deviation of 1. It centers the data around the mean and scales it based on the standard deviation. Standardization makes the data more suitable for algorithms that assume a Gaussian distribution or require features to have zero mean and unit variance.

Z = (X - μ) / σ

Where,

* X = Data
* μ = Mean value of X
* σ = Standard deviation of X

**Data Cleansing Tools**

Some data cleansing tools**:**

* **OpenRefine**: A powerful open-source tool for cleaning and transforming messy data. It supports tasks like removing duplicate and data enrichment with easy-to-use interface.
* **Trifacta Wrangler:** A user-friendly tool designed for cleaning, transforming and preparing data for analysis. It uses AI to suggest transformations to streamline workflows.
* **TIBCO Clarity:**A tool that helps in profiling, standardizing and enriching data. It’s ideal to make high quality data and consistency across datasets.
* **Cloudingo:** A cloud-based tool focusing on de-duplication, data cleansing and record management to maintain accuracy of data.
* **IBM Infosphere Quality Stage:** It’s highly suitable for large-scale and complex data.

**Advantages and Disadvantages of Data Cleaning in Machine Learning**

**Advantages:**

* **Improved model performance**: Removal of errors, inconsistencies and irrelevant data helps the model to better learn from the data.
* **Increased accuracy**: Helps ensure that the data is accurate, consistent and free of errors.
* **Better representation of the data**: Data cleaning allows the data to be transformed into a format that better represents the underlying relationships and patterns in the data.
* **Improved data quality:** Improve the quality of the data, making it more reliable and accurate.
* **Improved data security:** Helps to identify and remove sensitive or confidential information that could compromise data security.

**Disadvantages:**

* **Time-consuming**: It is very time consuming task specially for large and complex datasets.
* **Error-prone:** It can result in loss of important information.
* **Cost and resource-intensive:** It is resource-intensive process that requires significant time, effort and expertise. It can also require the use of specialized software tools.
* **Overfitting:** Data cleaning can contribute to overfitting by removing too much data.

**DATA FILTERING**

Data filtering is the process of choosing a smaller part of your data set and using that subset for viewing or analysis. Filtering is generally (but not always) temporary – the complete data set is kept, but only part of it is used for the calculation.

Data filtering is the process of selecting relevant information from a large set of data using specific criteria. It can be used to improve the quality of data, save time, and make analysis more efficient.

How it works Define criteria for analysis, Select tools for filtering, Apply criteria to the data, and Display only the rows that meet the criteria.

When it's useful

* Data filtering can be used in databases, spreadsheets, and programming languages
* It can help you find important data and get rid of unnecessary information
* It can help you extract meaningful insights from large datasets

Some techniques for filtering data

* **Rule-based filtering**: Uses rules to select relevant data
* **Range filtering**: Filters data by range
* **Conditional filtering**: Filters data based on conditions

Examples of filtering data

* In a database, you can filter data by property, operator, and value
* In Excel, you can filter data in a table or range

**DATA MERGING**

Data merging is the process of combining two or more data sets into a single, unified database. It involves adding new details to existing data, appending cases, and removing any duplicate or incorrect information to ensure that the data at hand is comprehensive, complete, and accurate.

Data merging is the process of combining multiple data sets into a single database. It can involve adding new details, removing duplicates, and ensuring the data is accurate.

Steps in data merging

1. **Check data quality**: Ensure the data is accurate and free of errors.
2. **Check data compatibility**: Ensure the data sets have the same structure and format.
3. **Identify duplicates**: Remove any duplicate records.
4. **Align data**: Make sure the data is aligned correctly.
5. **Merge data**: Combine the data sets into a single database.

Challenges in data merging

* **Data quality**: Merging low-quality data can introduce errors and inaccuracies.
* **Data compatibility**: Merging data sets with different structures can be complex.
* **Duplicate records**: If not handled properly, merging can introduce duplicate records.
* **Large datasets**: Merging large data sets can be challenging.
* **Ambiguous keys**: If data sets have similar keys or identifiers, it can be difficult to merge them.

Types of data merging

* **Conditional merge**: Used when merging incomplete data sets.
* **Many-to-many merge**: Occurs when neither data set has a unique value for a common variable.
* **Appending**: Adding observations from one data set to another data set.

**DATA RESHAPING**

**Reshape a Pandas DataFrame using stack,unstack and melt method**

Pandas use various methods to reshape the dataframe and series. Reshaping a Pandas DataFrame is a common operation to transform data structures for better analysis and visualization. The stack method pivots columns into rows, creating a multi-level index Series. Conversely, the unstack method reverses this process by pivoting inner index levels into columns. On the other hand, the melt method is used to transform wide-format data into a long-format, making it suitable for various analytical tasks. Let’s see about some of that reshaping method.

**Importing the Dataset**

To download the dataset used in this article click [here](https://media.geeksforgeeks.org/wp-content/uploads/nba.csv).

*# import pandas module*

**import** **pandas** **as** **pd**

*# making dataframe*

df = pd.read\_csv("https://media.geeksforgeeks.org/wp-content/uploads/nba.csv")

*# it was print the first 5-rows*

print(df.head())

**Output:**

**Name Team Number Position Age Height Weight College Salary**  
0 Avery Bradley Boston Celtics 0.0 PG 25.0 6-2 180.0 Texas 7730337.0  
1 Jae Crowder Boston Celtics 99.0 SF 25.0 6-6 235.0 Marquette 6796117.0  
2 John Holland Boston Celtics 30.0 SG 27.0 6-5 205.0 Boston University NaN  
3 R.J. Hunter Boston Celtics 28.0 SG 22.0 6-5 185.0 Georgia State 1148640.0  
4 Jonas Jerebko Boston Celtics 8.0 PF 29.0 6-10 231.0 NaN 5000000.0

**Reshape DataFrame in Pandas**

Below are the three methods that we will use to reshape the layout of tables in [Pandas](https://www.geeksforgeeks.org/python-pandas-dataframe/):

* Using Pandas stack() method
* Using unstack() method
* Using melt() method

**Reshape the Layout of Tables in Pandas Using stack() method**

The stack() method works with the MultiIndex objects in DataFrame, it returns a DataFrame with an index with a new inner-most level of row labels. It changes the wide table to a long table.

*# import pandas module*

**import** **pandas** **as** **pd**

*# making dataframe*

df = pd.read\_csv("nba.csv")

*# reshape the dataframe using stack() method*

df\_stacked = df.stack()

print(df\_stacked.head(26))

**Output:**

0 Name Avery Bradley  
 Team Boston Celtics  
 Number 0.0  
 Position PG  
 Age 25.0  
 Height 6-2  
 Weight 180.0  
 College Texas  
 Salary 7730337.0  
1 Name Jae Crowder  
 Team Boston Celtics  
 Number 99.0  
 Position SF  
 Age 25.0  
 Height 6-6  
 Weight 235.0  
 College Marquette  
 Salary 6796117.0  
2 Name John Holland  
 Team Boston Celtics  
 Number 30.0  
 Position SG  
 Age 27.0  
 Height 6-5  
 Weight 205.0  
 College Boston University  
dtype: object

**Reshape a Pandas DataFrame Using unstack() method**

The unstack() is similar to stack method, It also works with multi-index objects in dataframe, producing a reshaped DataFrame with a new inner-most level of column labels.

*# import pandas module*

**import** **pandas** **as** **pd**

*# making dataframe*

df = pd.read\_csv("nba.csv")

*# unstack() method*

df\_unstacked = df\_stacked.unstack()

print(df\_unstacked.head(10))

**Output**:

**Name Team Number Position Age Height Weight College Salary**   
0 Avery Bradley Boston Celtics 0.0 PG 25.0 6-2 180.0 Texas 7730337.0   
1 Jae Crowder Boston Celtics 99.0 SF 25.0 6-6 235.0 Marquette 6796117.0   
2 John Holland Boston Celtics 30.0 SG 27.0 6-5 205.0 Boston University NaN   
3 R.J. Hunter Boston Celtics 28.0 SG 22.0 6-5 185.0 Georgia State 1148640.0   
4 Jonas Jerebko Boston Celtics 8.0 PF 29.0 6-10 231.0 NaN 5000000.0   
5 Amir Johnson Boston Celtics 90.0 PF 29.0 6-9 240.0 NaN 12000000.0   
6 Jordan Mickey Boston Celtics 55.0 PF 21.0 6-8 235.0 LSU 1170960.0   
7 Kelly Olynyk Boston Celtics 41.0 C 25.0 7-0 238.0 Gonzaga 2165160.0   
8 Terry Rozier Boston Celtics 12.0 PG 22.0 6-2 190.0 Louisville 1824360.0   
9 Marcus Smart Boston Celtics 36.0 PG 22.0 6-4 220.0 Oklahoma State 3431040.0

**Reshape the Layout of Tables in Pandas Using melt() method**

The [melt()](https://www.geeksforgeeks.org/python-pandas-melt/) in Pandas reshape dataframe from wide format to long format. It uses the “id\_vars[‘col\_names’]” to melt the dataframe by column names.

*# import pandas module*

**import** **pandas** **as** **pd**

*# making dataframe*

df = pd.read\_csv("nba.csv")

*# it takes two columns "Name" and "Team"*

df\_melt = df.melt(id\_vars=['Name', 'Team'])

print(df\_melt.head(10))

**Output:**

**Name Team variable value**  
0 Avery Bradley Boston Celtics Number 0.0  
1 Jae Crowder Boston Celtics Number 99.0  
2 John Holland Boston Celtics Number 30.0  
3 R.J. Hunter Boston Celtics Number 28.0  
4 Jonas Jerebko Boston Celtics Number 8.0  
5 Amir Johnson Boston Celtics Number 90.0  
6 Jordan Mickey Boston Celtics Number 55.0  
7 Kelly Olynyk Boston Celtics Number 41.0  
8 Terry Rozier Boston Celtics Number 12.0  
9 Marcus Smart Boston Celtics Number 36.0

**Reshape a Pandas DataFrame using stack,unstack and melt method – FAQs**

**What does the stack() and unstack() Function Do on a DataFrame?**

***Stack()****:*

* *stack() is a function in pandas that “compresses” a level in the DataFrame’s columns to produce a longer, narrower DataFrame (or Series if the resulting structure is one-dimensional). It essentially pivots the column labels into the row index, creating a multi-level index.*

***Unstack()****:*

* *unstack() performs the inverse operation of stack(). It pivots a level of the (possibly hierarchical) row index into the column axis, producing a wider and shorter DataFrame. This is useful for unstacking grouped or aggregated data.*

**How to Reshape a Pandas DataFrame**

*Reshaping a DataFrame can be achieved through several functions including pivot(), melt(), stack(), and unstack(), each serving different reshaping needs:*

* ***Pivot*** *is used to create a new derived table out of a given one.*
* ***Melt*** *is used to transform or reshape data from wide format to long format.*
* ***Stack/Unstack*** *as described above for moving column index levels to row index levels and vice-versa.*

**Difference Between Melt and Unstack in Pandas**

* ***Melt****: This function is used to transform a DataFrame from a wide format to a long format. It can be seen as ‘unpivoting’ a DataFrame, turning columns into rows.*
* ***Unstack****: This function moves information from the index to the columns. It is typically used on DataFrames with multi-level indexes to reshape them into a more readable form.*

**What does melt() Do in Python?**

*The melt() function in pandas transforms a DataFrame from a wide format to a long format. The id\_vars specify columns to use as identifier variables, while value\_vars specify columns to unpivot. If value\_vars is not specified, all non-id\_vars will be unpivoted.*

**Difference Between Stacking and Unstacking**

* ***Stacking****: Converts DataFrame columns into a new level of index, making the DataFrame taller and narrower (more rows, fewer columns).*
* ***Unstacking****: Converts a level of index into DataFrame columns, making the DataFrame shorter and wider (fewer rows, more columns).*

**DATA AGGREGATION**

**Grouping and Aggregating with Pandas**

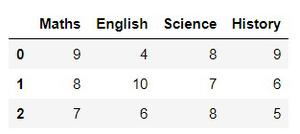
Grouping and aggregating will help to achieve data analysis easily using various functions. These methods will help us to the group and summarize our data and make complex analysis comparatively easy.

**Creating a sample dataset of marks of various subjects.**

* Python

|  |
| --- |
| # import module  **import** pandas as pd    # Creating our dataset  df **=** pd.DataFrame([[9, 4, 8, 9],                     [8, 10, 7, 6],                     [7, 6, 8, 5]],                    columns**=**['Maths',  'English',                             'Science', 'History'])    # display dataset  print(df) |

**Output:**



**Aggregation in Pandas**

Aggregation in pandas provides various functions that perform a mathematical or logical operation on our dataset and returns a summary of that function. Aggregation can be used to get a summary of columns in our dataset like getting sum, minimum, maximum, etc. from a particular column of our dataset. The function used for aggregation is agg(), the parameter is the function we want to perform.

Some functions used in the aggregation are:

***Function Description:***

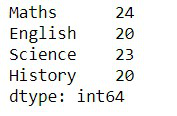
* *sum()         :Compute sum of column values*
* *min()          :Compute min of column values*
* *max()         :Compute max of column values*
* *mean()       :Compute mean of column*
* *size()          :Compute column sizes*
* *describe()  :Generates descriptive statistics*
* *first()          :Compute first of group values*
* *last()          :Compute last of group values*
* *count()       :Compute count of column values*
* *std()           :Standard deviation of column*
* *var()           :Compute variance of column*
* *sem()         :Standard error of the mean of column*

**Examples:**

* The sum() function is used to calculate the sum of every value.
* Python

|  |
| --- |
| df.sum() |

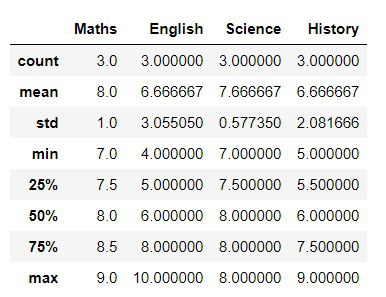
**Output:**



* The describe() function is used to get a summary of our dataset
* Python

|  |
| --- |
| df.describe() |

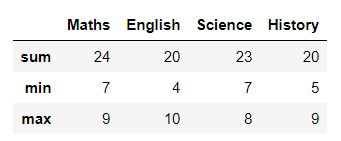
**Output:**



* We used agg() function to calculate the sum, min, and max of each column in our dataset.
* Python

|  |
| --- |
| df.agg(['sum', 'min', 'max']) |

**Output:**



**Grouping in Pandas**

Grouping is used to group data using some criteria from our dataset. It is used as split-apply-combine strategy.

* Splitting the data into groups based on some criteria.
* Applying a function to each group independently.
* Combining the results into a data structure.

**Examples:**

We use groupby() function to group the data on “Maths” value. It returns the object as result.

* Python

|  |
| --- |
| df.groupby(by**=**['Maths']) |

**Output:**

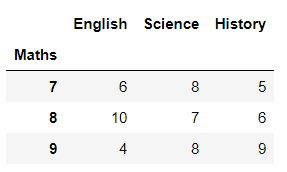
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000012581821388>

Applying groupby() function to group the data on “Maths” value. To view result of formed groups use first() function.

* Python

|  |
| --- |
| a **=** df.groupby('Maths')  a.first() |

**Output:**

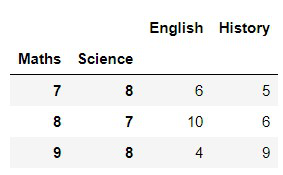


First grouping based on “Maths” within each team we are grouping based on “Science”

* Python

|  |
| --- |
| b **=** df.groupby(['Maths', 'Science'])  b.first() |

**Output:**



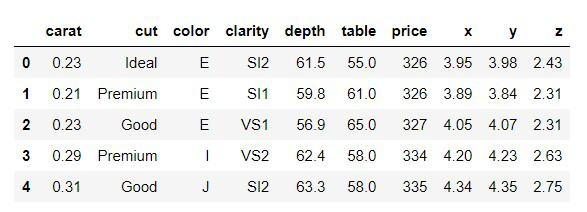
**Implementation on a Dataset**

Here we are using a dataset of diamond information.

* Python

|  |
| --- |
| # import module  **import** numpy as np  **import** pandas as pd    # reading csv file  dataset **=** pd.read\_csv("diamonds.csv")    # printing first 5 rows  print(dataset.head(5)) |

**Output:**



* We group by using cut and get the sum of all columns.
* Python

|  |
| --- |
| dataset.groupby('cut').sum() |

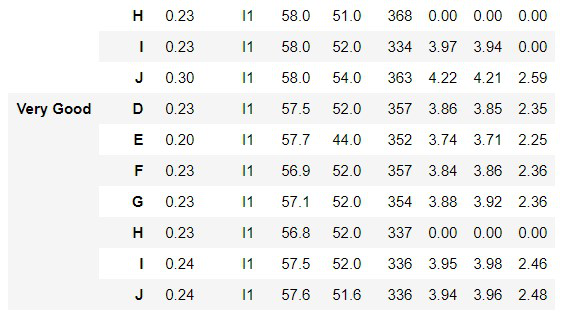
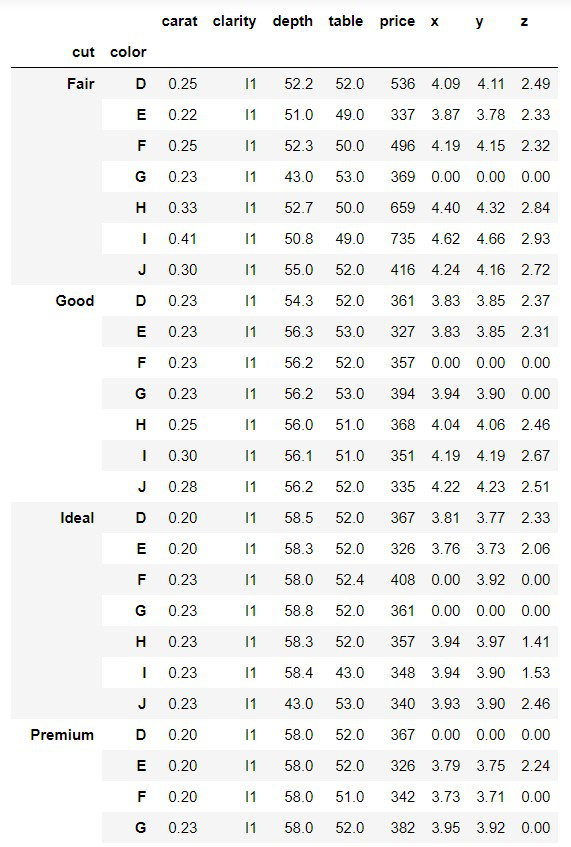
**Output:**



* Here we are grouping using cut and color and getting minimum value for all other groups.
* Python

|  |
| --- |
| dataset.groupby(['cut', 'color']).agg('min') |

**Output:**



* Here we are grouping using color and getting aggregate values like sum, mean, min, etc. for the price group.
* Python

|  |
| --- |
| # dictionary having key as group name of price and  # value as list of aggregation function  # we want to perform on group price  agg\_functions **=** {      'price':      ['sum', 'mean', 'median', 'min', 'max', 'prod']  }    dataset.groupby(['color']).agg(agg\_functions) |

**Output:**



We can see that in the prod(product i.e. multiplication) column all values are inf, inf is the result of a numerical calculation that is mathematically infinite