Overview of Introduction to Data Science for an Applied Data Science Course

Introduction to Data Science:

Definition and significance of data science in various industries.

The role of a data scientist and the data science lifecycle.

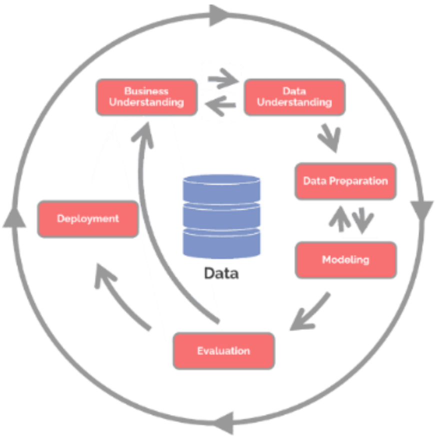
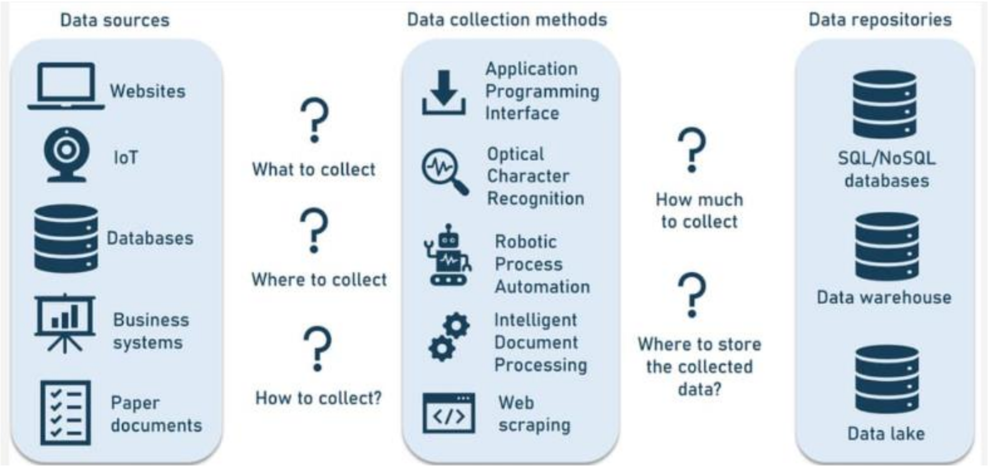


Figure : Data Science Project Life Cycle

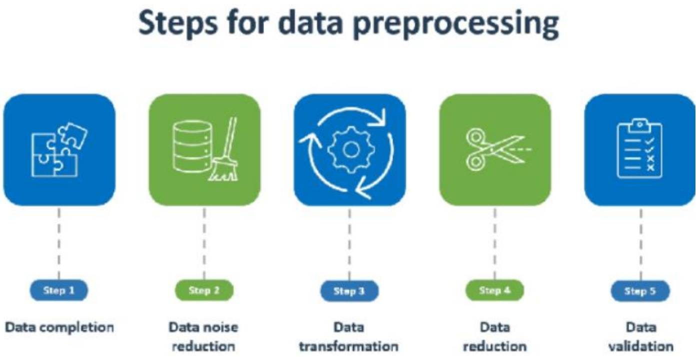
DATA SCIENCE PROJECT LIFECYCLE

Data Collection and Preprocessing:

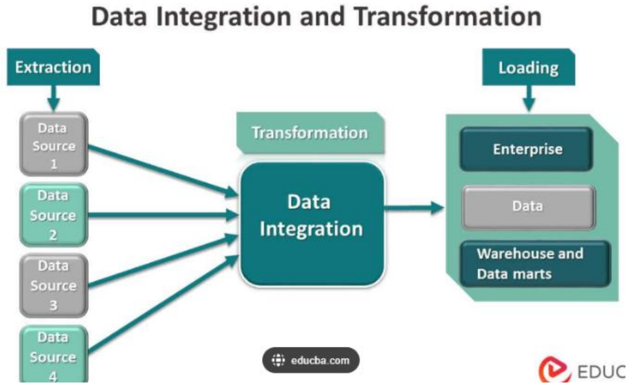
Methods for collecting data from various sources.



Data cleaning and preprocessing techniques to handle missing values, outliers, and inconsistencies.

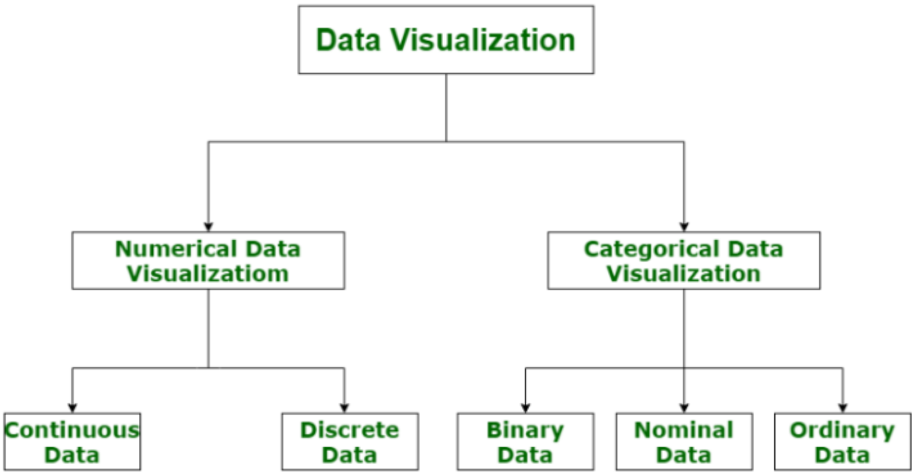


Data integration and transformation processes.



Exploratory Data Analysis (EDA):

Techniques for summarizing and visualizing data.


Data Visualization

Numerical Data Visualization

Categorical Data Visualization

Continuous Data

Discrete Data

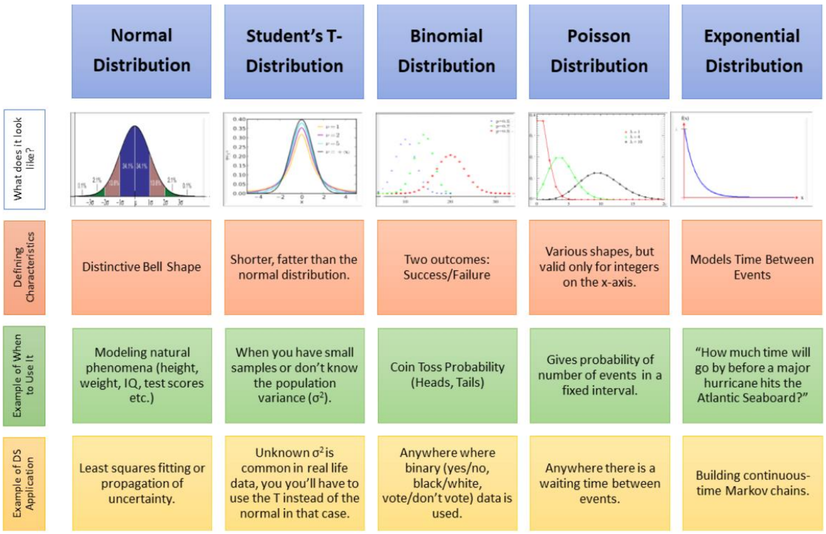
Binary Data

Nominal Data

Ordinary Data

Identifying patterns, trends, and anomalies in datasets.

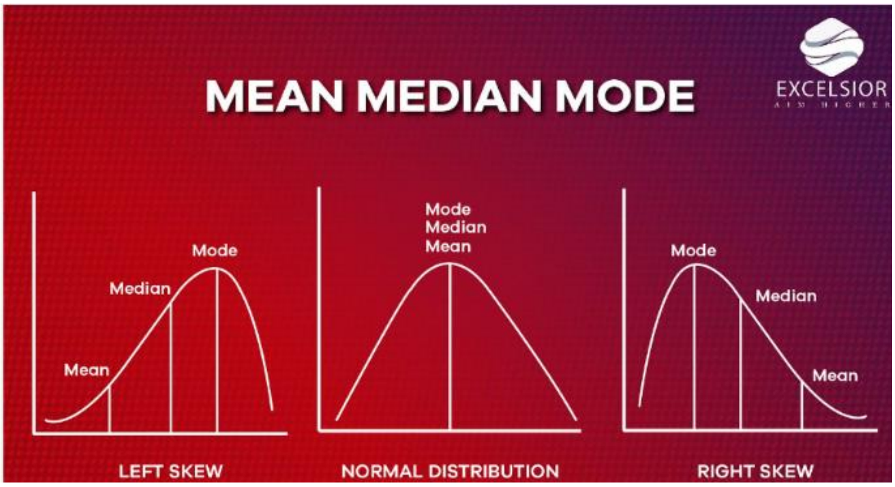
Use of statistical methods to explore and understand data distributions.



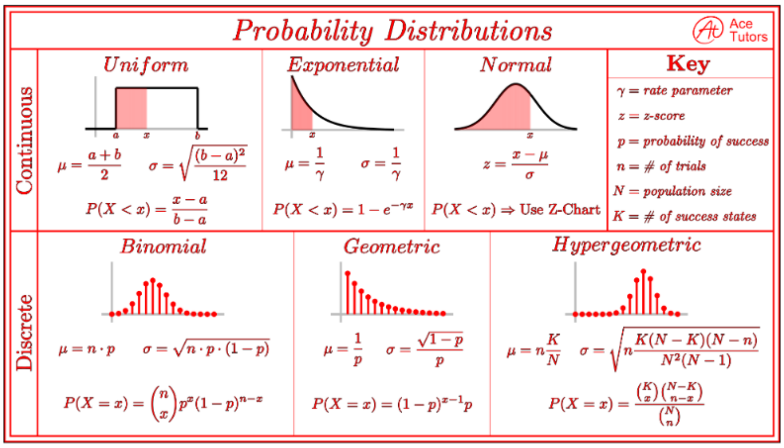
	Normal Distribution	Student's T-Distribution	Binomial Distribution	Poisson Distribution	Exponential Distribution
What does it look like?					
Defining Characteristics	Distinctive Bell Shape	Shorter, fatter than the normal distribution.	Two outcomes: Success/Failure	Various shapes, but valid only for integers on the x-axis.	Models Time Between Events
Example of When to Use It	Modeling natural phenomena (height, weight, IQ, test scores etc.)	When you have small samples or don't know the population variance (σ^2).	Coin Toss Probability (Heads, Tails)	Gives probability of number of events in a fixed interval.	"How much time will go by before a major hurricane hits the Atlantic Seaboard?"
Example of DS Application	Least squares fitting or propagation of uncertainty.	Unknown σ^2 is common in real life data, you you'll have to use the T instead of the normal in that case.	Anywhere where binary (yes/no, black/white, vote/don't vote) data is used.	Anywhere there is a waiting time between events.	Building continuous-time Markov chains.

Statistical Foundations:

Descriptive statistics: mean, median, mode, variance, and standard deviation.



Probability theory and distributions.



Probability Distributions

Ace Tutors

Continuous

Uniform

$\mu = \frac{a+b}{2}$ $\sigma = \sqrt{\frac{(b-a)^2}{12}}$

$P(X < x) = \frac{x-a}{b-a}$

Exponential

$\mu = \frac{1}{\gamma}$ $\sigma = \frac{1}{\gamma}$

$P(X < x) = 1 - e^{-\gamma x}$

Normal

$z = \frac{x-\mu}{\sigma}$

$P(X < x) \Rightarrow$ Use Z-Chart

Key

γ = rate parameter

z = z-score

p = probability of success

n = # of trials

N = population size

K = # of success states

Discrete

Binomial

$\mu = n \cdot p$ $\sigma = \sqrt{n \cdot p \cdot (1-p)}$

$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$

Geometric

$\mu = \frac{1}{p}$ $\sigma = \frac{\sqrt{1-p}}{p}$

$P(X = x) = (1-p)^{x-1} p$

Hypergeometric

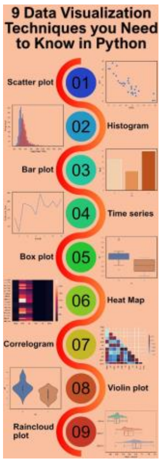
$\mu = n\frac{K}{N}$ $\sigma = \sqrt{n\frac{K(N-K)(N-n)}{N^2(N-1)}}$

$P(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$

Data Visualization:

Principles of effective data visualisation.

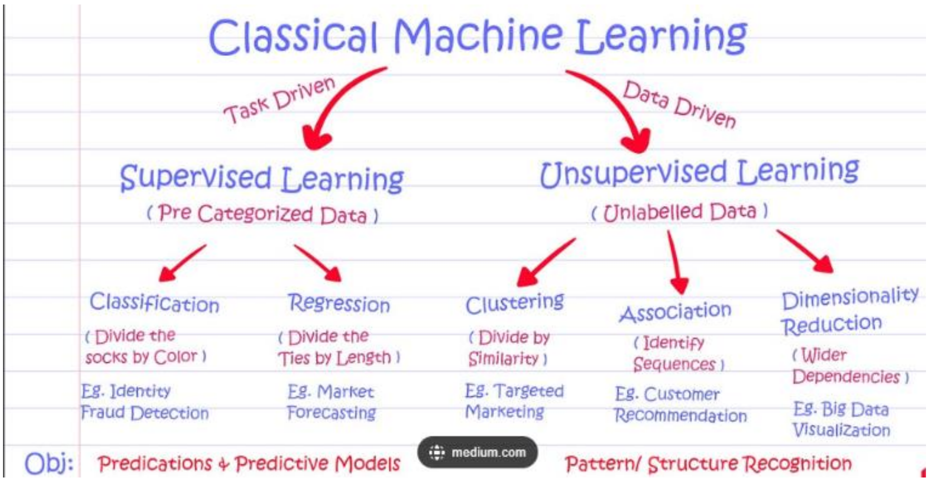
Tools and libraries for creating visualisations (e.g., Matplotlib, Seaborn, Tableau).



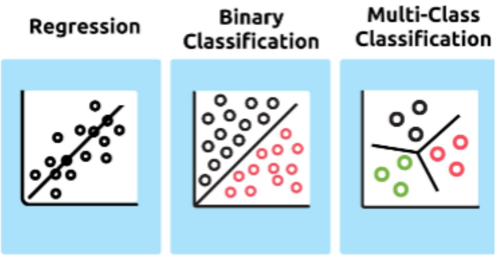
Designing and interpreting various types of plots, charts, and dashboards.

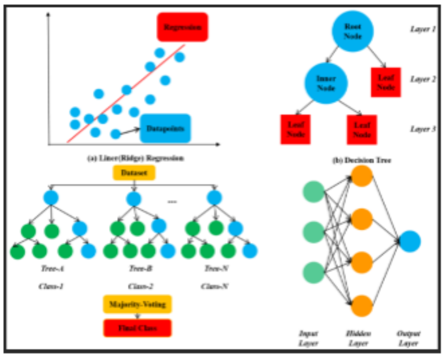
Introduction to Machine Learning:

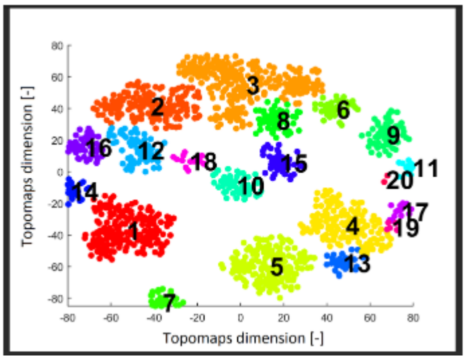
Basic concepts and types of machine learning: supervised, unsupervised, and reinforcement learning.



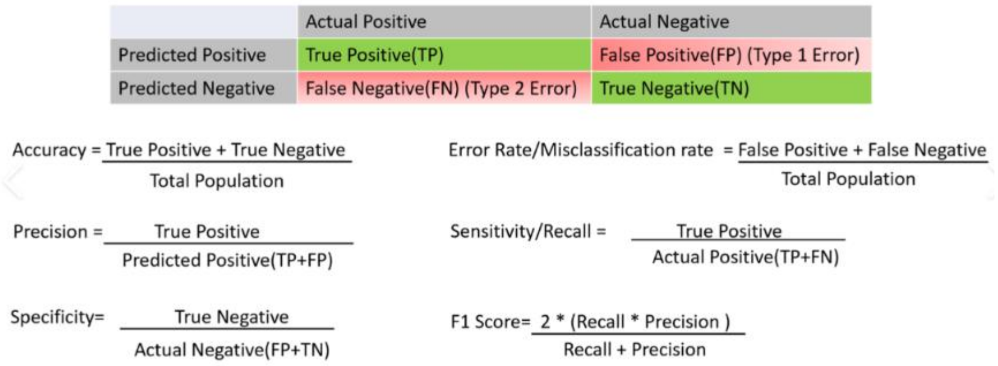
Key algorithms: regression, classification, clustering, and decision trees.







Model evaluation and validation techniques.



	Actual Positive	Actual Negative
Predicted Positive	True Positive(TP)	False Positive(FP) (Type 1 Error)
Predicted Negative	False Negative(FN) (Type 2 Error)	True Negative(TN)

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total Population}}$$

$$\text{Error Rate/Misclassification rate} = \frac{\text{False Positive + False Negative}}{\text{Total Population}}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{Predicted Positive(TP+FP)}}$$

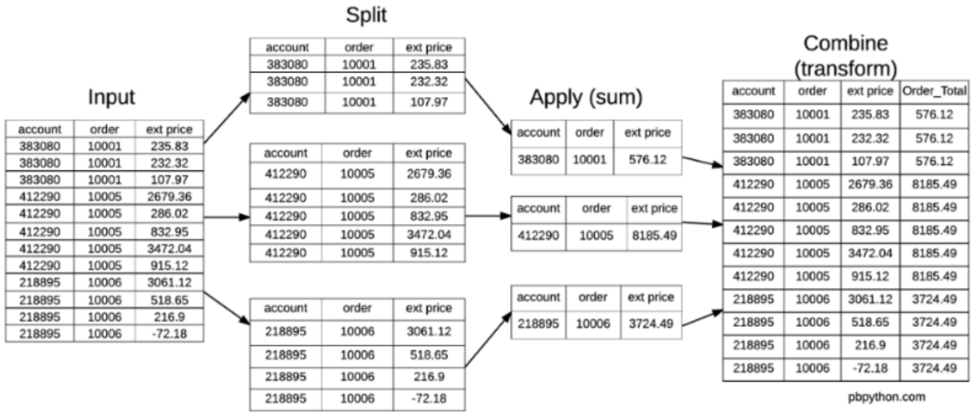
$$\text{Sensitivity/Recall} = \frac{\text{True Positive}}{\text{Actual Positive(TP+FN)}}$$

$$\text{Specificity} = \frac{\text{True Negative}}{\text{Actual Negative(FP+TN)}}$$

$$\text{F1 Score} = \frac{2 * \text{(Recall * Precision)}}{\text{Recall + Precision}}$$

Data Wrangling and Transformation:

Techniques for manipulating and transforming data using libraries such as Pandas.



Input

account	order	ext price
383080	10001	235.83
383080	10001	232.32
383080	10001	107.97
412290	10005	2679.36
412290	10005	286.02
412290	10005	832.95
412290	10005	3472.04
412290	10005	915.12
218895	10006	3061.12
218895	10006	518.65
218895	10006	216.9
218895	10006	-72.18

Split

account	order	ext price
383080	10001	235.83
383080	10001	232.32
383080	10001	107.97

account	order	ext price
412290	10005	2679.36
412290	10005	286.02
412290	10005	832.95
412290	10005	3472.04
412290	10005	915.12

account	order	ext price
218895	10006	3061.12
218895	10006	518.65
218895	10006	216.9
218895	10006	-72.18

Apply (sum)

account	order	ext price
383080	10001	576.12

account	order	ext price
412290	10005	8185.49

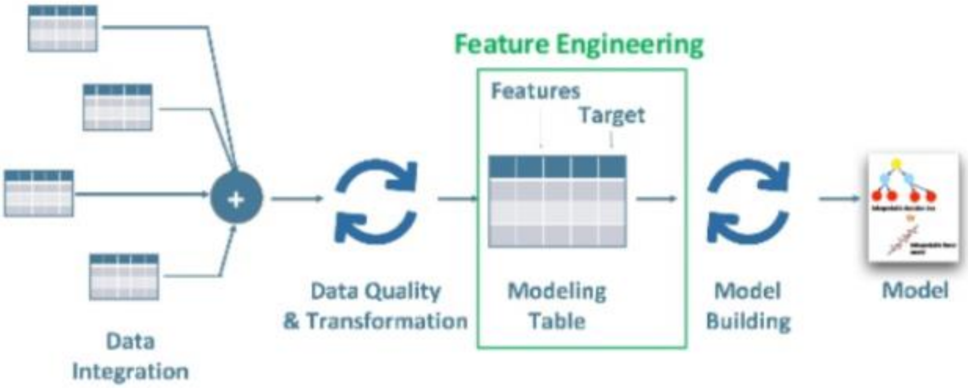
account	order	ext price
218895	10006	3724.49

Combine (transform)

account	order	ext price	Order_Total
383080	10001	235.83	576.12
383080	10001	232.32	576.12
383080	10001	107.97	576.12
412290	10005	2679.36	8185.49
412290	10005	286.02	8185.49
412290	10005	832.95	8185.49
412290	10005	3472.04	8185.49
412290	10005	915.12	8185.49
218895	10006	3061.12	3724.49
218895	10006	518.65	3724.49
218895	10006	216.9	3724.49
218895	10006	-72.18	3724.49

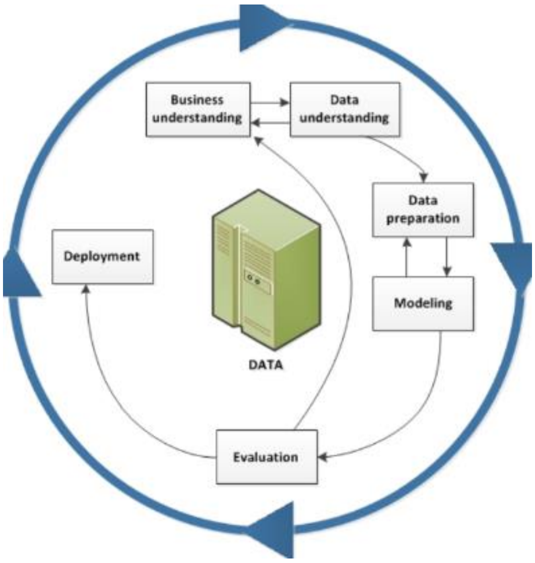
pbpython.com

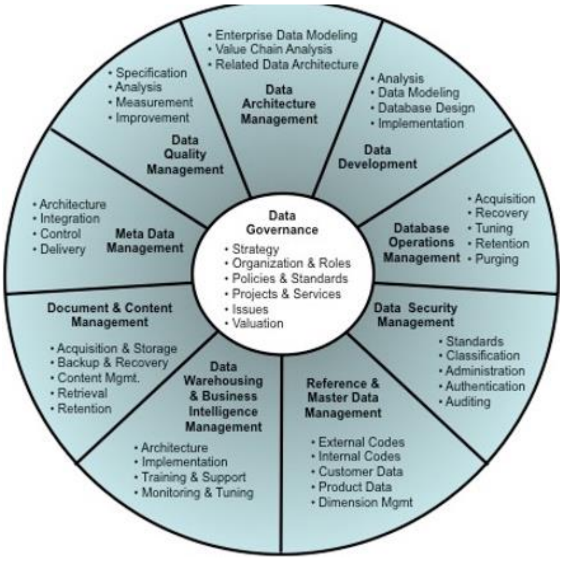
Feature engineering and selection methods to improve model performance.



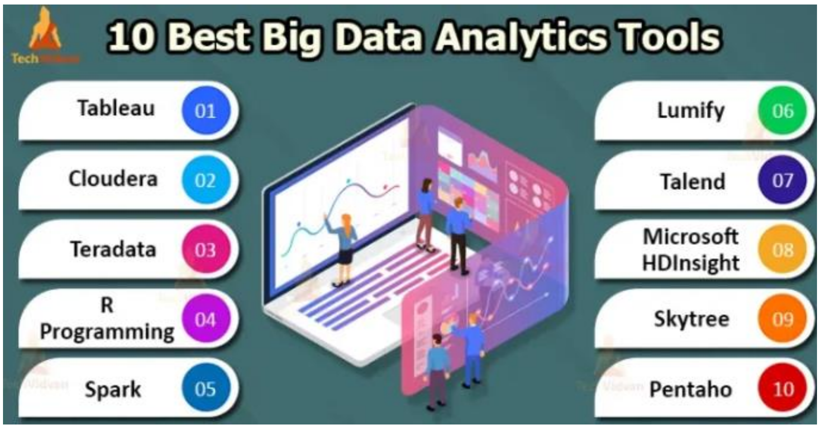
Big Data Technologies:

Overview of big data and its significance.





Introduction to Hadoop, Spark, and other big data tools.

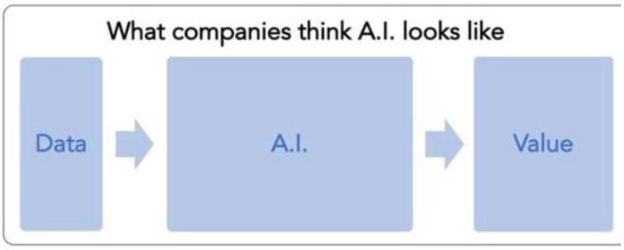


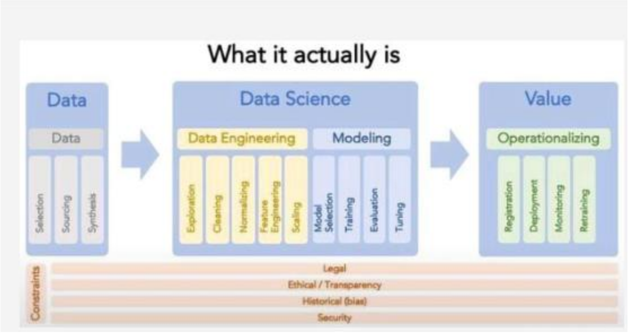
Processing and analysing large-scale datasets.

Ethics and Data Privacy:

Ethical considerations in data science.

Big Data Ethics
Privacy Protection
Responsible
Fair Data Usage
Vast amount of Data
Transparency
Ethical Handling
Minimizing Biases





Data privacy laws and regulations.

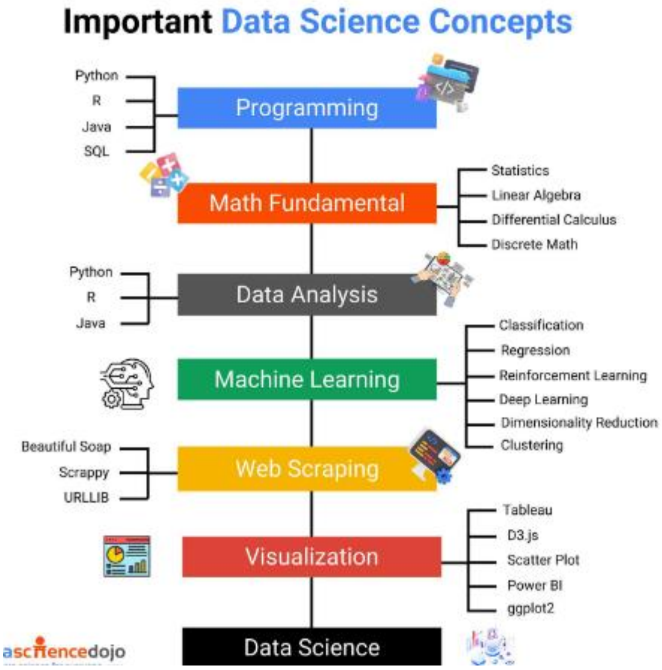
Best practices for responsible data handling and analysis.

Applied Data Science Projects:

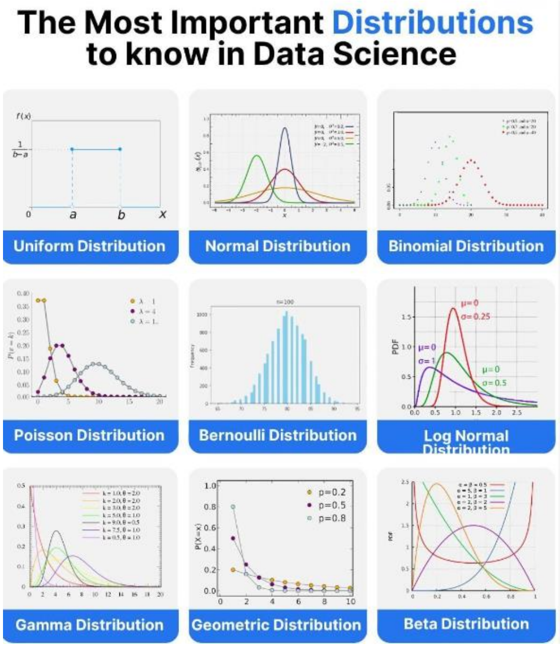
Hands-on projects and case studies to apply learned concepts.

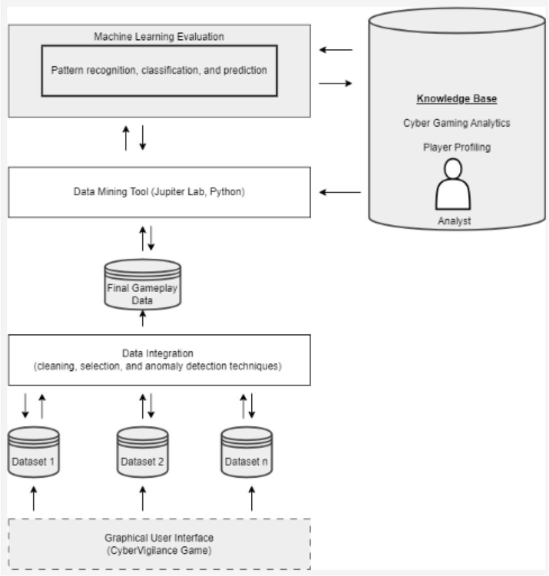
Working with real-world datasets to solve practical problems.

Collaboration and presentation of findings.



Basic Statistics and Data Processing




Machine Learning Evaluation
Pattern recognition, classification, and prediction
Knowledge Base
Cyber Gaming Analytics
Player Profiling
Analyst
Data Mining Tool (Jupiter Lab, Python)
Final Gameplay Data
Data Integration
(cleaning, selection, and anomaly detection techniques)
Dataset 1
Dataset 2
Dataset n
Graphical User Interface
(CyberVigilance Game)

Kaggle is an online community and platform for data scientists and machine learning practitioners. It provides users with resources and tools to build and hone their data science and machine learning skills. Key features of Kaggle include:

Competitions: Users can participate in machine learning competitions where they solve real-world problems and compete for prizes. These competitions often provide datasets and a leaderboard to track progress.

Datasets: Kaggle hosts a vast collection of public datasets, which users can access and use for analysis, model building, and learning.

Kernels (Notebooks): Kaggle provides a cloud-based coding environment called Kernels, where users can write and run code in Python or R without needing to install any software on their local machines!

Learning: Kaggle offers courses and tutorials on various topics in data science and machine learning, allowing users to learn at their own pace.

Community: Users can engage with a global community of data scientists, participate in discussions, share code, and collaborate on projects.

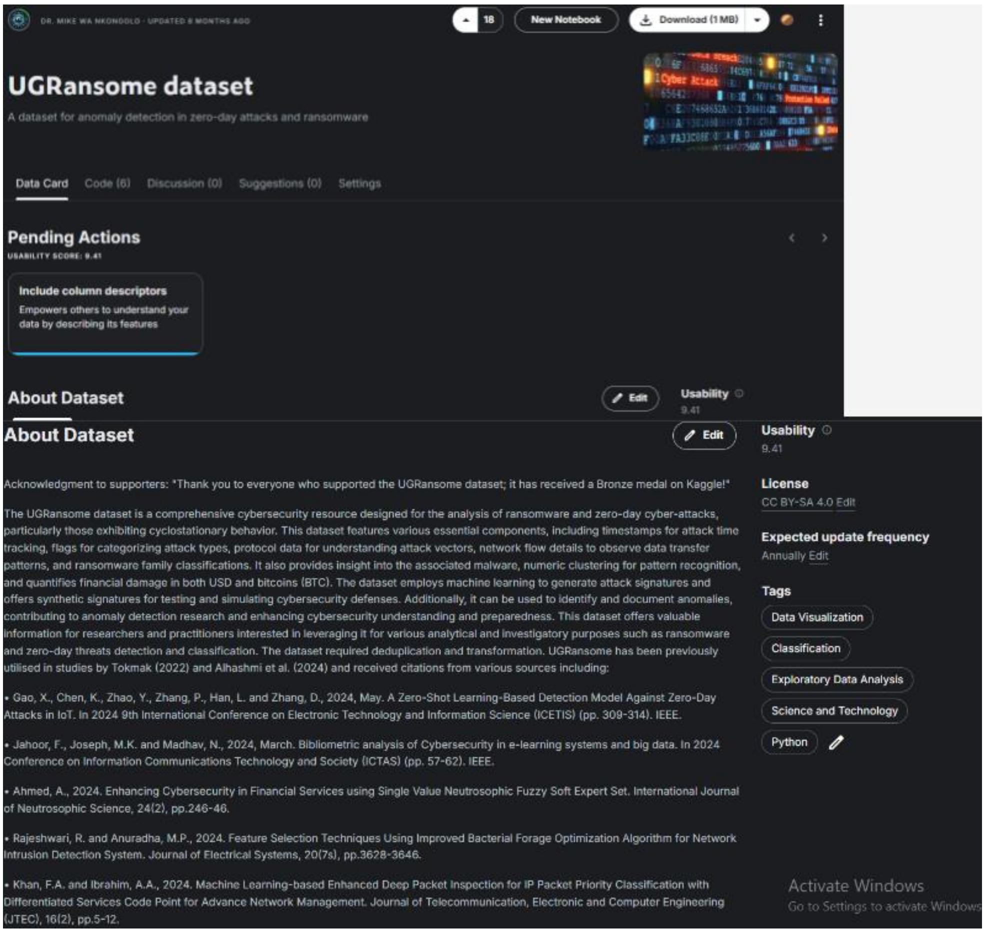
Kaggle is widely used for both educational purposes and professional development, offering a platform to learn, practice, and demonstrate skills in data science and machine learning.

The code is also available on Kaggle.

Definition: A DataFrame is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure in the **pandas library of Python**. It is similar to a table in a database or an Excel spreadsheet, where data is arranged in rows and columns.

Exploratory data analysis (EDA): packages in Python is extremely beneficial for gaining insights into the dataset, cleaning and preparing the data, and guiding the feature engineering and modeling process. These packages provide a range of tools for summarizing, visualizing, and understanding the data, making them essential for any data science project.

The UGRansome dataset:



UGRansome dataset

A dataset for anomaly detection in zero-day attacks and ransomware

Pending Actions

USABILITY SCORE: 9.41

Include column descriptors

Empowers others to understand your data by describing its features

About Dataset

Usability

9.41

About Dataset

Usability

9.41

Acknowledgment to supporters: "Thank you to everyone who supported the UGRansome dataset; it has received a Bronze medal on Kaggle!"

The UGRansome dataset is a comprehensive cybersecurity resource designed for the analysis of ransomware and zero-day cyber-attacks, particularly those exhibiting cyclostationary behavior. This dataset features various essential components, including timestamps for attack time tracking, flags for categorizing attack types, protocol data for understanding attack vectors, network flow details to observe data transfer patterns, and ransomware family classifications. It also provides insight into the associated malware, numeric clustering for pattern recognition, and quantifies financial damage in both USD and bitcoins (BTC). The dataset employs machine learning to generate attack signatures and offers synthetic signatures for testing and simulating cybersecurity defenses. Additionally, it can be used to identify and document anomalies, contributing to anomaly detection research and enhancing cybersecurity understanding and preparedness. This dataset offers valuable information for researchers and practitioners interested in leveraging it for various analytical and investigatory purposes such as ransomware and zero-day threats detection and classification. The dataset required deduplication and transformation. UGRansome has been previously utilised in studies by Tokmak (2022) and Alhashmi et al. (2024) and received citations from various sources including:

- Gao, X., Chen, K., Zhao, Y., Zhang, P., Han, L. and Zhang, D., 2024, May. A Zero-Shot Learning-Based Detection Model Against Zero-Day Attacks in IoT. In 2024 9th International Conference on Electronic Technology and Information Science (ICETIS) (pp. 309-314). IEEE.

- Jahoor, F., Joseph, M.K. and Madhav, N., 2024, March. Bibliometric analysis of Cybersecurity in e-learning systems and big data. In 2024 Conference on Information Communications Technology and Society (ICTAS) (pp. 57-62). IEEE.

- Ahmed, A., 2024. Enhancing Cybersecurity in Financial Services using Single Value Neutrosophic Fuzzy Soft Expert Set. International Journal of Neutrosophic Science, 24(2), pp.246-46.

- Rajeshwari, R. and Anuradha, M.P., 2024. Feature Selection Techniques Using Improved Bacterial Forage Optimization Algorithm for Network Intrusion Detection System. Journal of Electrical Systems, 20(7s), pp.3628-3646.

- Khan, F.A. and Ibrahim, A.A., 2024. Machine Learning-based Enhanced Deep Packet Inspection for IP Packet Priority Classification with Differentiated Services Code Point for Advance Network Management. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 16(2), pp.5-12.

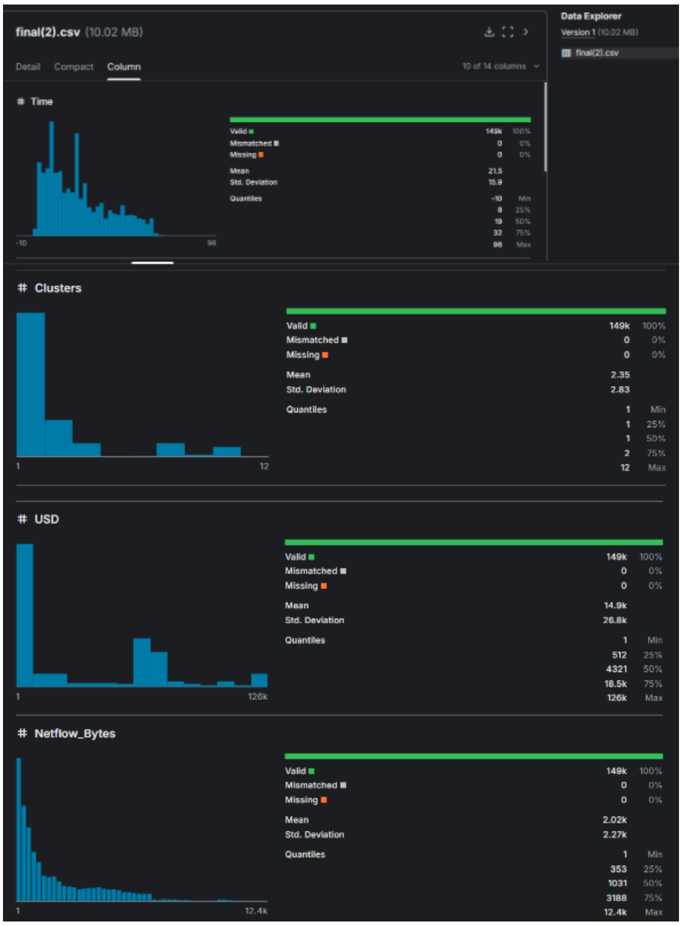
License

CC BY-SA 4.0 Edit

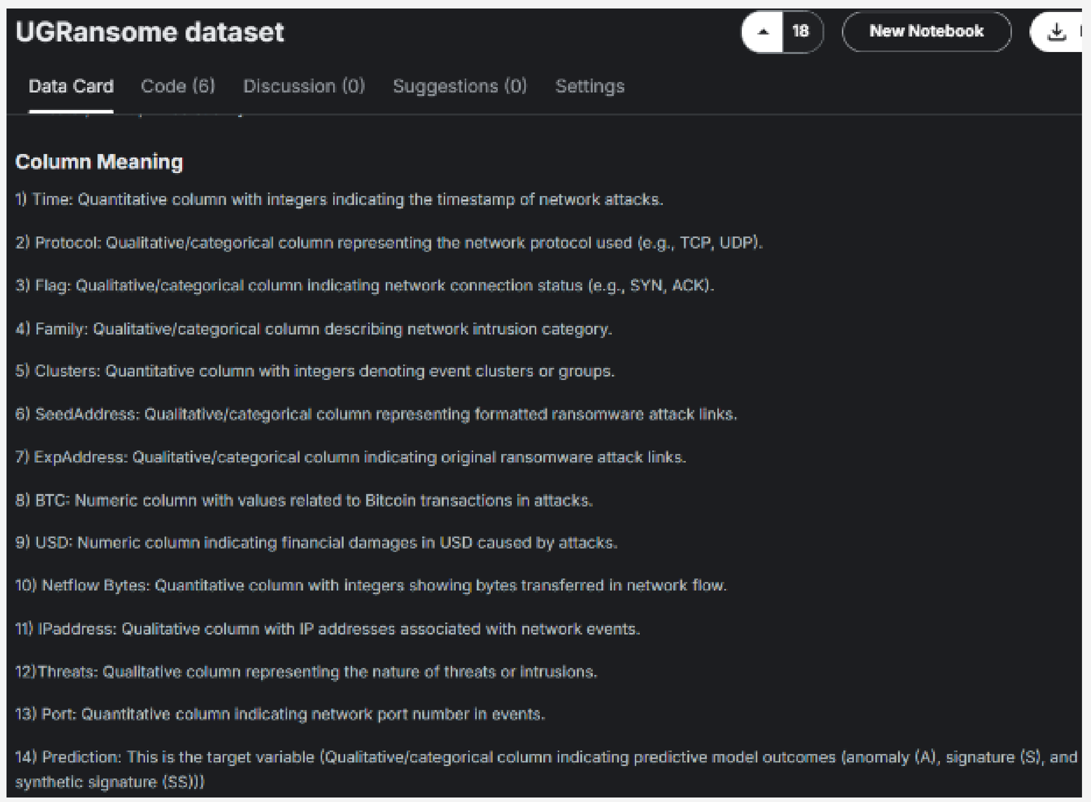
Expected update frequency

Annually Edit

Tags



```
# Time
Valid           149k  100%
Mismatched         0    0%
Missing            0    0%
Mean            21.5
Std. Deviation  15.9
Quantiles        -10   Min
                   8   25%
                  19   50%
                  32   75%
                  96   Max

# Clusters
Valid           149k  100%
Mismatched         0    0%
Missing            0    0%
Mean            2.35
Std. Deviation  2.83
Quantiles          1   Min
                   1   25%
                   1   50%
                   2   75%
                  12   Max

# USD
Valid           149k  100%
Mismatched         0    0%
Missing            0    0%
Mean           14.9k
Std. Deviation 26.8k
Quantiles          1   Min
                 512   25%
                4321   50%
               18.5k   75%
                126k   Max

# Netflow_Bytes
Valid           149k  100%
Mismatched         0    0%
Missing            0    0%
Mean           2.02k
Std. Deviation 2.27k
Quantiles          1   Min
                 353   25%
                1031   50%
                3188   75%
               12.4k   Max
```



UGRansome dataset

Column Meaning

1) Time: Quantitative column with integers indicating the timestamp of network attacks.

2) Protocol: Qualitative/categorical column representing the network protocol used (e.g., TCP, UDP).

3) Flag: Qualitative/categorical column indicating network connection status (e.g., SYN, ACK).

4) Family: Qualitative/categorical column describing network intrusion category.

5) Clusters: Quantitative column with integers denoting event clusters or groups.

6) SeedAddress: Qualitative/categorical column representing formatted ransomware attack links.

7) ExpAddress: Qualitative/categorical column indicating original ransomware attack links.

8) BTC: Numeric column with values related to Bitcoin transactions in attacks.

9) USD: Numeric column indicating financial damages in USD caused by attacks.

10) Netflow Bytes: Quantitative column with integers showing bytes transferred in network flow.

11) IPaddress: Qualitative column with IP addresses associated with network events.

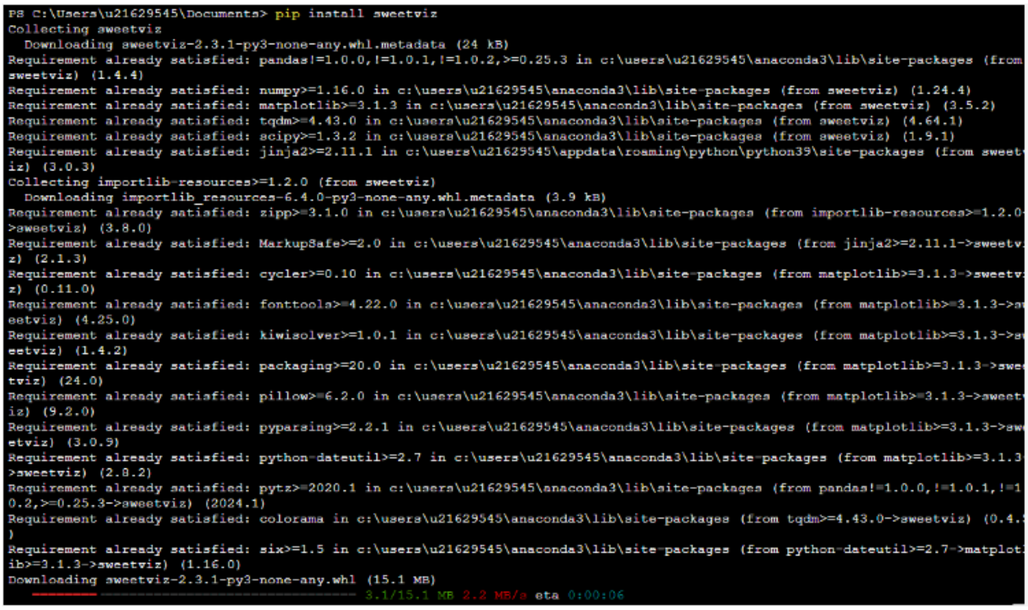
12)Threats: Qualitative column representing the nature of threats or intrusions.

13) Port: Quantitative column indicating network port number in events.

14) Prediction: This is the target variable (Qualitative/categorical column indicating predictive model outcomes (anomaly (A), signature (S), and synthetic signature (SS)))

To use a module (e.g., Sweetviz), you need to install it first. You can do this using pip in the terminal.

Command to install Sweetviz: pip install sweetviz



```
PS C:\Users\u21629545\Documents> pip install sweetviz
Collecting sweetviz
  Downloading sweetviz-2.3.1-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,>=0.25.3 in c:\users\u21629545\anaconda3\lib\site-packages (from
sweetviz) (1.4.4)
Requirement already satisfied: numpy>=1.16.0 in c:\users\u21629545\anaconda3\lib\site-packages (from sweetviz) (1.24.4)
Requirement already satisfied: matplotlib>=3.1.3 in c:\users\u21629545\anaconda3\lib\site-packages (from sweetviz) (3.5.2)
Requirement already satisfied: tqdm>=4.43.0 in c:\users\u21629545\anaconda3\lib\site-packages (from sweetviz) (4.64.1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\u21629545\anaconda3\lib\site-packages (from sweetviz) (1.9.1)
Requirement already satisfied: jinja2>=2.11.1 in c:\users\u21629545\appdata\roaming\python\python39\site-packages (from sweet
iz) (3.0.3)
Collecting importlib-resources>=1.2.0 (from sweetviz)
  Downloading importlib_resources-6.4.0-py3-none-any.whl.metadata (3.9 kB)
Requirement already satisfied: zipp>=3.1.0 in c:\users\u21629545\anaconda3\lib\site-packages (from importlib-resources>=1.2.0-
>sweetviz) (3.8.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\u21629545\anaconda3\lib\site-packages (from jinja2>=2.11.1->sweetv
z) (2.1.3)
Requirement already satisfied: cycler>=0.10 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->sweetv
z) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->sw
eetviz) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->sw
eetviz) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->swe
tviz) (24.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->sweet
iz) (9.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3->swe
etviz) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\u21629545\anaconda3\lib\site-packages (from matplotlib>=3.1.3
>sweetviz) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\u21629545\anaconda3\lib\site-packages (from pandas!=1.0.0,!=1.0.1,!=1
0.2,>=0.25.3->sweetviz) (2024.1)
Requirement already satisfied: colorama in c:\users\u21629545\anaconda3\lib\site-packages (from tqdm>=4.43.0->sweetviz) (0.4.
)
Requirement already satisfied: six>=1.5 in c:\users\u21629545\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplot
ib>=3.1.3->sweetviz) (1.16.0)
Downloading sweetviz-2.3.1-py3-none-any.whl (15.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/15.1 MB 2.2 MB/s eta 0:00:06
```

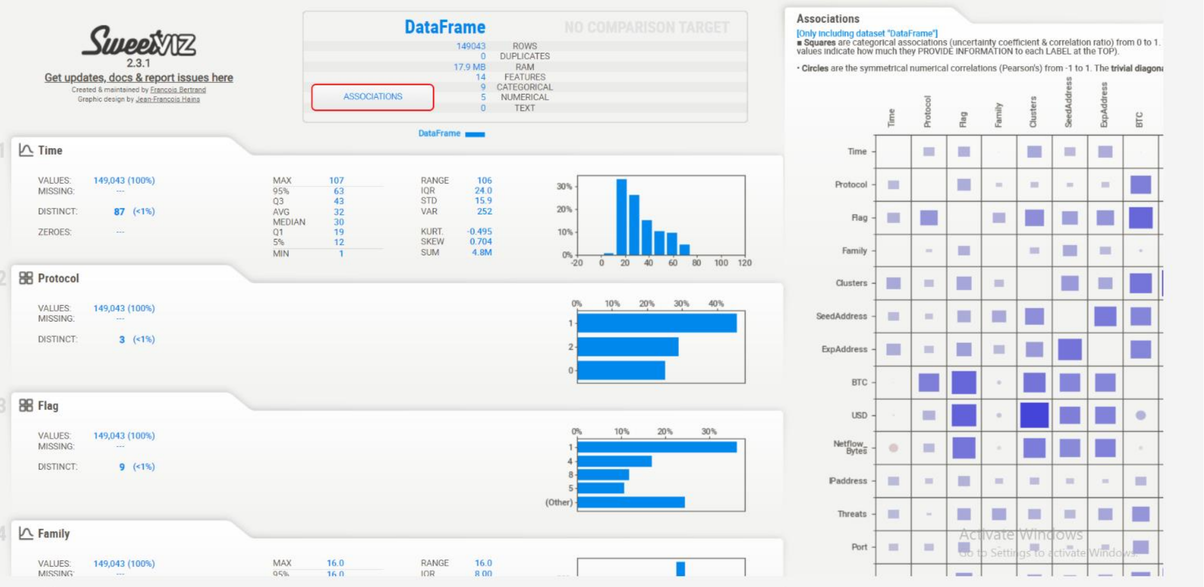
Sweetviz is EDA module that generates a highly detailed, visual analysis of the data and provides a comprehensive visual report of the dataset (please do not incorporate vusualizations generated by Sweetviz in your assignment. Only using it to understand your data).

Code

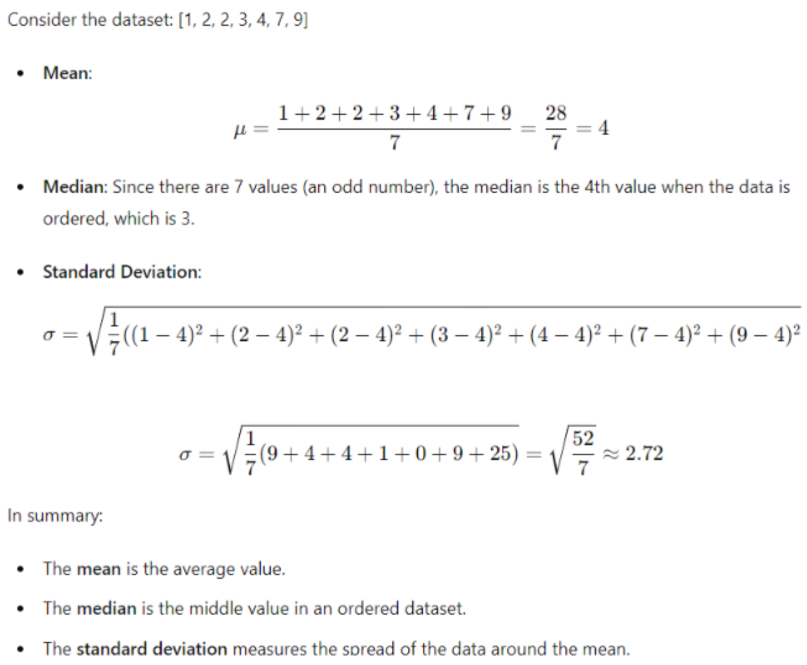
import sweetviz as sv

report = sv.analyze(UGRansome)

report.show_html('report.html')



```
Sweetviz 2.3.1
Get updates, docs & report issues here
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

DataFrame                NO COMPARISON TARGET
   149043   ROWS
        0   DUPLICATES
  17.9 MB   RAM
       14   FEATURES
        9   CATEGORICAL
        5   NUMERICAL
        0   TEXT
ASSOCIATIONS
DataFrame

1 Time
VALUES:    149,043 (100%)    MAX     107    RANGE   106
MISSING:   ---               95%      63    IQR     24.0
                             Q3       43    STD     15.9
DISTINCT:  87  (<1%)         AVG      32    VAR     252
                             MEDIAN   30
ZEROES:    ---               Q1       19    KURT.   -0.495
                             5%       12    SKEW    0.704
                             MIN       1    SUM     4.8M

2 Protocol
VALUES:    149,043 (100%)
MISSING:   ---
DISTINCT:  3  (<1%)

3 Flag
VALUES:    149,043 (100%)
MISSING:   ---
DISTINCT:  9  (<1%)

4 Family
VALUES:    149,043 (100%)    MAX     16.0   RANGE   16.0
MISSING:   ---               95%     16.0   IQR     8.00

Associations
[Only including dataset "DataFrame"]
■ Squares are categorical associations (uncertainty coefficient & correlation ratio) from 0 to 1.
values indicate how much they PROVIDE INFORMATION to each LABEL at the TOP).
• Circles are the symmetrical numerical correlations (Pearson's) from -1 to 1. The trivial diagona
```



Consider the dataset: [1, 2, 2, 3, 4, 7, 9]

- **Mean**:

$$\mu = \frac{1+2+2+3+4+7+9}{7} = \frac{28}{7} = 4$$

- **Median**: Since there are 7 values (an odd number), the median is the 4th value when the data is ordered, which is 3.

- **Standard Deviation**:

$$\sigma = \sqrt{\frac{1}{7}((1-4)^2 + (2-4)^2 + (2-4)^2 + (3-4)^2 + (4-4)^2 + (7-4)^2 + (9-4)^2}$$

$$\sigma = \sqrt{\frac{1}{7}(9+4+4+1+0+9+25)} = \sqrt{\frac{52}{7}} \approx 2.72$$

In summary:

- The **mean** is the average value.
- The **median** is the middle value in an ordered dataset.
- The **standard deviation** measures the spread of the data around the mean.

These metrics—**mean, median, and standard deviation**—provide important insights into the characteristics of a dataset:

Mean:

Central Tendency: The mean gives you an idea of the average value in the dataset.

Overall Level: It indicates the overall level of the values in the dataset.

Effect of Outliers: Since the mean is sensitive to outliers, a few extremely high or low values can significantly affect it. This can be useful to know if the dataset contains outliers.

Median:

Central Tendency: Like the mean, the median gives you an idea of the central tendency of the dataset.

Middle Value: The median is the middle value when the data is ordered, so it represents the 50th percentile.

Resilience to Outliers: The median is not affected by outliers, making it a better measure of central tendency when the dataset contains extreme values.

Standard Deviation:

Spread: The standard deviation tells you how spread out the values in the dataset are around the mean.

Variability: A low standard deviation means the values are close to the mean, indicating low variability. A high standard deviation means the values are spread out over a wider range, indicating high variability.

Data Consistency: It helps to understand the consistency of the data. For example, in quality control processes, a high standard deviation might indicate inconsistent product quality.

What These Metrics Tell You About a Dataset

Central Tendency and Typical Value: The mean and median provide information about the central tendency or typical value of the data.

- If the mean and median are close, the dataset is likely symmetrically distributed.
- If the mean is significantly higher or lower than the median, the dataset may be skewed.

Skewness:

- If the mean is greater than the median, the data might be right-skewed (positively skewed).
- If the mean is less than the median, the data might be left-skewed (negatively skewed).

Variability and Consistency:

- A low standard deviation indicates that the data points are close to the mean, suggesting low variability and high consistency.
- A high standard deviation indicates that the data points are spread out over a larger range, suggesting high variability and low consistency.

Practical Examples

Income Data: In income data, the mean might be higher than the median if there are a few individuals with very high incomes (right-skewed distribution). The median might give a better sense of a "typical" income.

Test Scores: For test scores of a class, if the standard deviation is low, it indicates that most students scored similarly. A high standard deviation would indicate a wide range of scores, with students scoring very differently from each other.

Quality Control: In manufacturing, a low standard deviation of product dimensions might indicate high precision in the manufacturing process, whereas a high standard deviation might indicate issues with the process consistency.

Understanding these metrics helps in making informed decisions, identifying patterns, and detecting anomalies in the data.

A correlation matrix is a table showing the correlation coefficients between many variables. Each cell in the table shows the correlation between two variables. The value is in the range of -1 to 1.

- 1 indicates a strong positive correlation: as one variable increases, the other variable increases.
- -1 indicates a strong negative correlation: as one variable increases, the other variable decreases.
- 0 indicates no correlation: the variables do not have a linear relationship.

Why Use a Correlation Matrix?

- Identify Relationships: It helps in identifying the relationships between different variables in a dataset.
- Feature Selection: It can be used in feature selection to identify highly correlated features and reduce redundancy.
- Data Understanding: It provides a quick overview of how variables are related to each other, which is essential for data exploration and understanding.

Lecture Code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from sklearn.preprocessing import StandardScaler

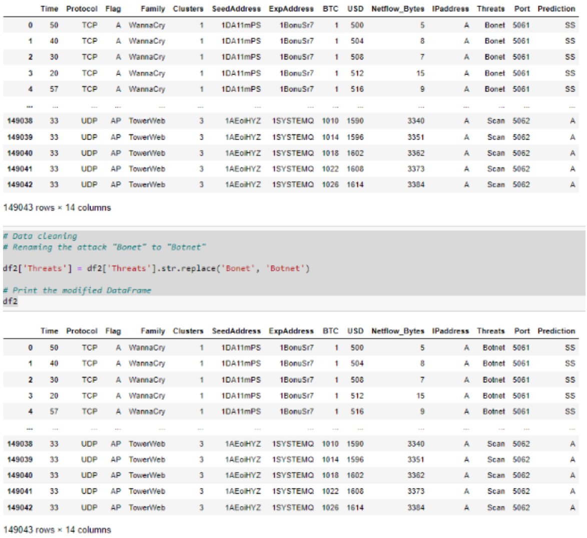
# Phase 1: Data Collection (e.g., UGRansome dataset)

df2 = pd.read_csv(r'C:/Users/u21629545/Downloads/archive (8)/final(2).csv')

df2

df2.columns = ['Time','Protocol','Flag','Family','Clusters','SeedAddress','ExpAddress','BTC','USD','Netflow_Bytes','IP address','Threats','Port','Prediction']

df2



```
        Time Protocol Flag    Family Clusters SeedAddress ExpAddress  BTC   USD Netflow_Bytes IPaddress Threats Port Prediction
     0    50      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   500             5         A   Bonet 5061         SS
     1    40      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   504             8         A   Bonet 5061         SS
     2    30      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   508             7         A   Bonet 5061         SS
     3    20      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   512            15         A   Bonet 5061         SS
     4    57      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   516             9         A   Bonet 5061         SS
   ...   ...      ...  ...       ...      ...        ...        ...   ...   ...           ...       ...     ...  ...        ...
149038    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1010  1590          3340         A    Scan 5062          A
149039    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1014  1596          3351         A    Scan 5062          A
149040    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1018  1602          3362         A    Scan 5062          A
149041    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1022  1608          3373         A    Scan 5062          A
149042    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1026  1614          3384         A    Scan 5062          A

149043 rows × 14 columns
```

```
# Data cleaning
# Renaming the attack "Bonet" to "Botnet"

df2['Threats'] = df2['Threats'].str.replace('Bonet', 'Botnet')

# Print the modified DataFrame
df2
```

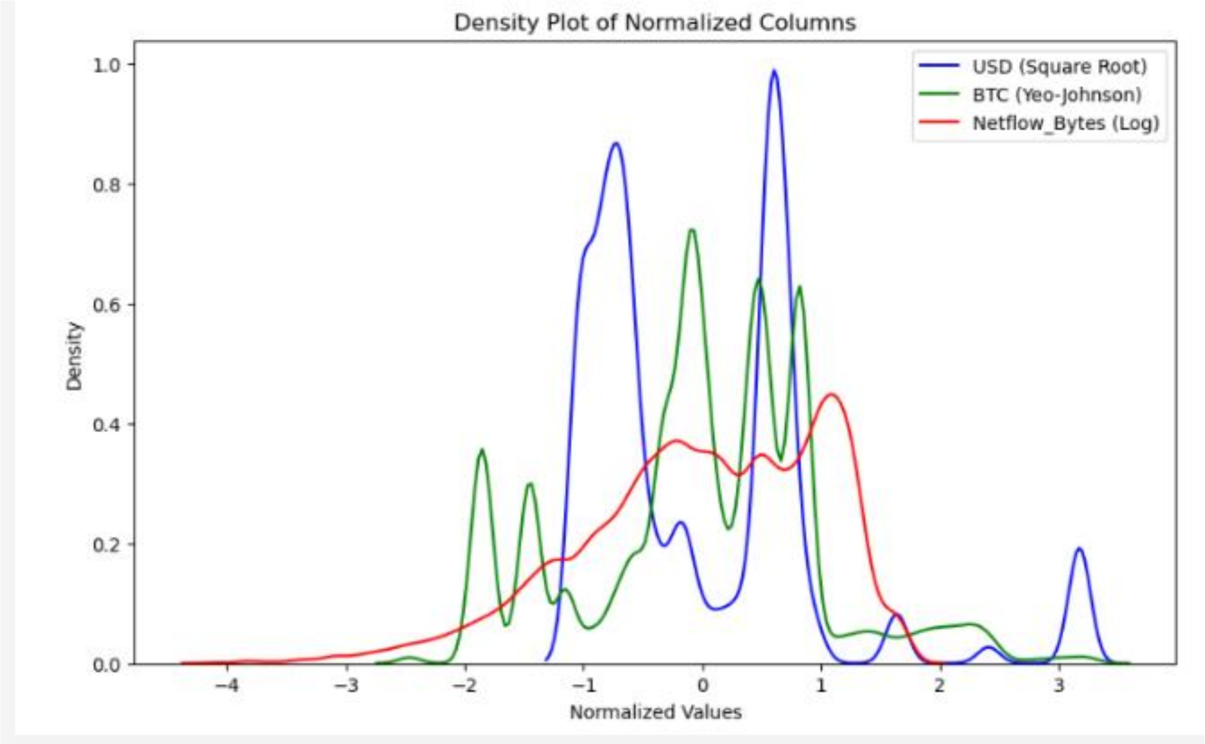
```
        Time Protocol Flag    Family Clusters SeedAddress ExpAddress  BTC   USD Netflow_Bytes IPaddress Threats Port Prediction
     0    50      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   500             5         A  Botnet 5061         SS
     1    40      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   504             8         A  Botnet 5061         SS
     2    30      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   508             7         A  Botnet 5061         SS
     3    20      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   512            15         A  Botnet 5061         SS
     4    57      TCP    A  WannaCry        1   1DA11mPS   1BonuSr7     1   516             9         A  Botnet 5061         SS
   ...   ...      ...  ...       ...      ...        ...        ...   ...   ...           ...       ...     ...  ...        ...
149038    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1010  1590          3340         A    Scan 5062          A
149039    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1014  1596          3351         A    Scan 5062          A
149040    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1018  1602          3362         A    Scan 5062          A
149041    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1022  1608          3373         A    Scan 5062          A
149042    33      UDP   AP  TowerWeb        3   1AEoiHYZ  1SYSTEMQ  1026  1614          3384         A    Scan 5062          A

149043 rows × 14 columns
```

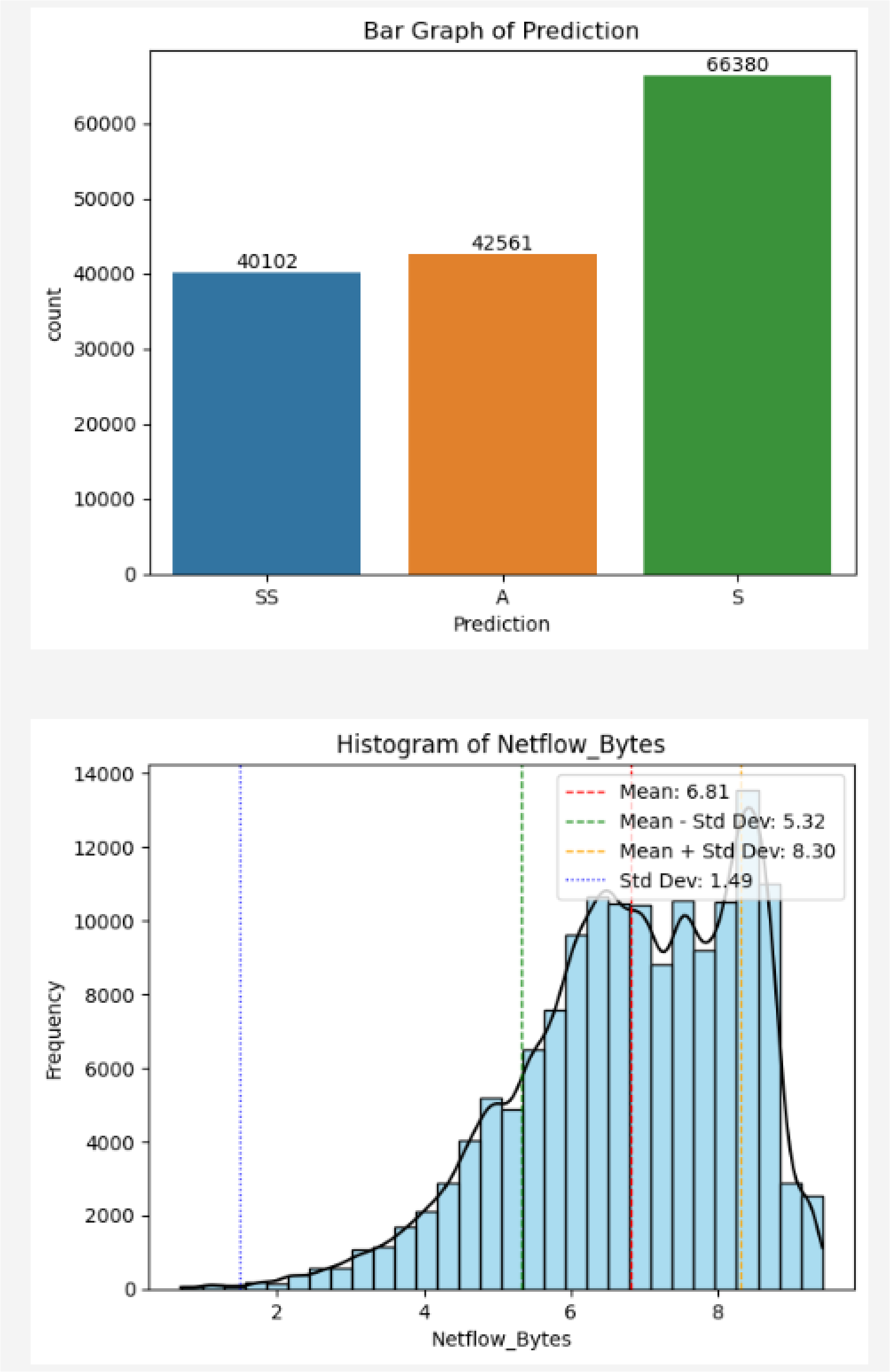
```
 # Data cleaning

# Renaming the attack "Bonet" to "Botnet"

df2['Threats'] = df2['Threats'].str.replace('Bonet', 'Botnet')

# Print the modified DataFrame

df2

# Phase 2: Data Preparation (feature engineering and data transformation)

# --- Drop all duplicate rows --- #

df2 = df2.drop_duplicates()

# --- Remove negative values from time/timestamp feature --- #

df2['Time'] = df2['Time'] + 11

# --- Math transformations to reduce skewness --- #

# --- Log transformation applied to column NETFLOW_BYTES --- #

# A log transformation involves taking the natural logarithm (base e) of each data point in a particular column or feature.

#Logarithmic transformations are often used to reduce the impact of extreme values (outliers) and make the data conform more

#closely to a normal distribution. They are particularly useful when dealing with positively skewed data,

#where the tail of the distribution is elongated on the right side.


#The np.log() function is a common way to perform a logarithmic transformation in Python.

#The + 1 added to the data points is often used to avoid issues with taking the logarithm of zero or negative values.

#It's a common practice to add a small constant like 1 to the data before applying the logarithm.

#By applying a log transformation to a feature, you're essentially compressing the range of values in that feature,

#which can help in cases where the data exhibits a rightward skew, making it more suitable for certain types of analysis

#or modeling techniques that assume normally distributed data.

df2['Netflow_Bytes'] = np.log(df2['Netflow_Bytes']+1)


# --- Square root transformation applied to columns USD ---#
```

```
#Square Root Transformation: A square root transformation involves taking the square root of each data point in the
#specified column. In this case, it's applied to the 'USD' column.
#Square root transformations are a type of mathematical transformation used to mitigate the impact of right-skewed data.
#Just like logarithmic transformations, square root transformations can help make the data more symmetric and closer to
#a normal distribution.
#The np.sqrt() function is used to calculate the square root.
#By applying a square root transformation to the 'USD' column, the code is attempting to make the data distribution less skewed
#and more suitable for certain statistical analyses or modeling techniques that assume normally distributed data or
#require data to be more symmetric. It's a common technique used in data preprocessing to improve the quality of data for
#analysis or modeling
df2['USD'] = np.sqrt(df2['USD'])
# --- Yeo Johnson transformation applied to columns BTC--#


#Yeo-Johnson transformation is being applied to the 'BTC' column in the DataFrame (df2['BTC']).
#This transformation is used to modify the data in the 'BTC' column to make its distribution more normalized or symmetric
#The Yeo-Johnson transformation is a mathematical transformation technique used to modify the distribution of data.
#It can be applied to both positive and negative values and is more versatile than some other transformations like the Box-Cox transformation.
#The transformation is performed using the stats.yeojohnson() function from a library like SciPy


df2['BTC'], _ = stats.yeojohnson(df2['BTC'])


#--PLOTING TRANSFORMED DATA--#
fig, ax = plt.subplots(figsize=(10, 6))
# Plot the transformed 'USD' column
ax.hist(df2['USD'], bins=50, alpha=0.5, color='blue', label='USD (Square Root)')
```

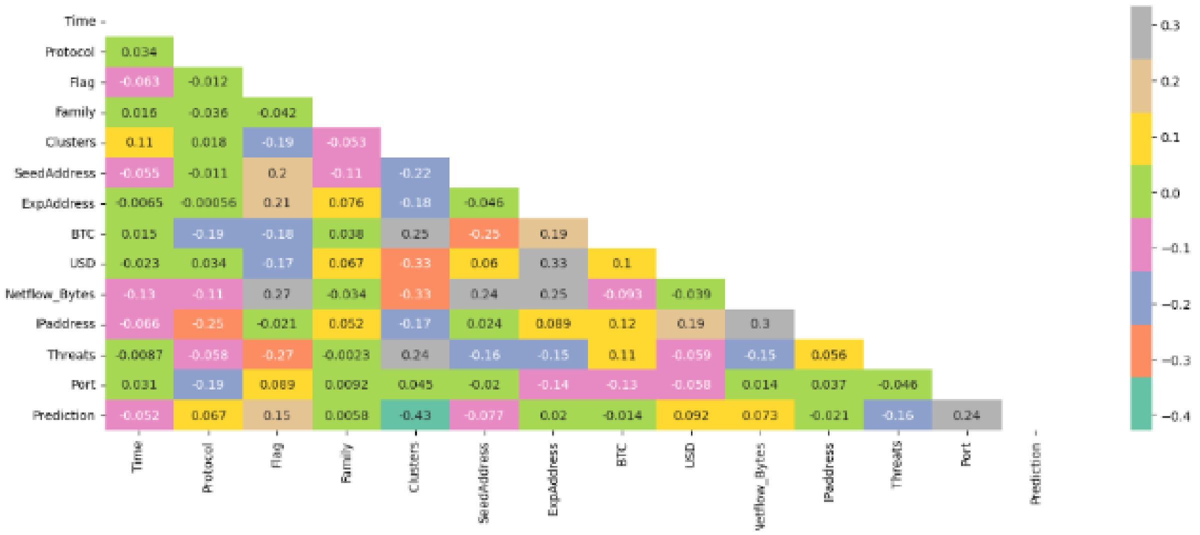
```
# Plot the transformed 'BTC' column
ax.hist(df2['BTC'], bins=50, alpha=0.5, color='green', label='BTC (Yeo-Johnson)')
# Plot the transformed 'Netflow_Bytes' column
ax.hist(df2['Netflow_Bytes'], bins=50, alpha=0.5, color='red', label='Netflow_Bytes (Log)')
# Add labels and a legend
ax.set_xlabel('Transformed Values')
ax.set_ylabel('Frequency')
ax.set_title('Distribution of Transformed Columns')
ax.legend()
# Show the plot
plt.show()



# Create a figure and axis for the plot
fig, ax = plt.subplots(figsize=(10, 6))
# Create a StandardScaler instance
# The StandardScaler is a common preprocessing technique used in machine learning and data analysis.
#It is used to standardize or normalize the features of a dataset by scaling them such that they have a mean of 0 and a standard
#deviation of 1.

#Standardizing the features is useful because it makes different features more directly comparable, especially in algorithms
#that are sensitive to the scale of the input data, such as many machine learning algorithms.
#In the code provided, scaler is created as an instance of the StandardScaler class, which can then be used to standardize
#the specified columns in the df2 DataFrame using the fit_transform method, as seen in the subsequent code
scaler = StandardScaler()
# Normalize each column's features
df2_normalized = df2.copy()
```

```
df2_normalized[['USD', 'BTC', 'Netflow_Bytes']] = scaler.fit_transform(df2[['USD', 'BTC', 'Netflow_Bytes']])

# Plot the density of the normalized 'USD' column

sns.kdeplot(df2_normalized['USD'], color='blue', label='USD (Square Root)', ax=ax)

# Plot the density of the normalized 'BTC' column

sns.kdeplot(df2_normalized['BTC'], color='green', label='BTC (Yeo-Johnson)', ax=ax)

# Plot the density of the normalized 'Netflow_Bytes' column

sns.kdeplot(df2_normalized['Netflow_Bytes'], color='red', label='Netflow_Bytes (Log)', ax=ax)

# Add labels and a legend

ax.set_xlabel('Normalized Values')

ax.set_ylabel('Density')

ax.set_title('Density Plot of Normalized Columns')

ax.legend()

# Show the plot

plt.show()
```



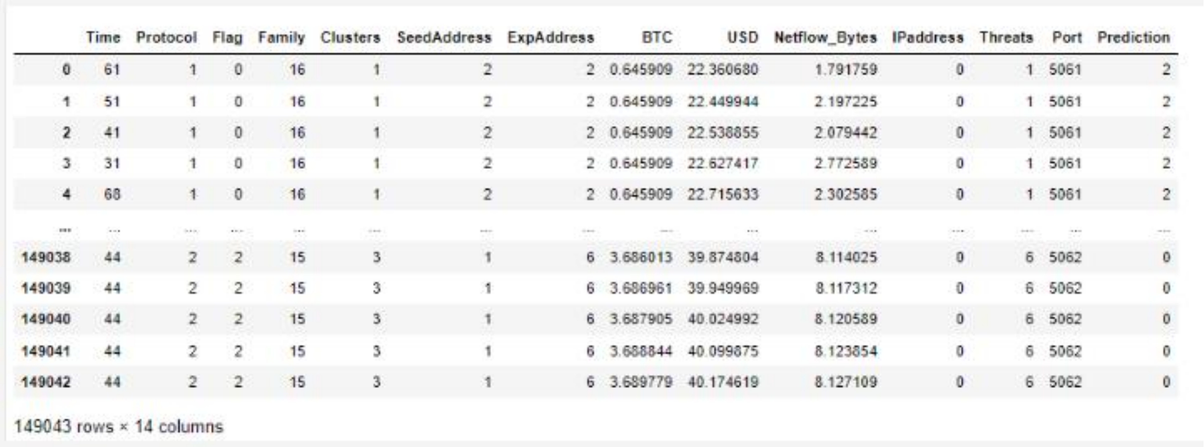

Bar Graph of Prediction
66380
60000
50000
42561
40102
40000
count
30000
20000
10000
0
SS
A
S
Prediction
Histogram of Netflow_Bytes
14000
12000
10000
8000
6000
4000
2000
0
Frequency
Mean: 6.81
Mean - Std Dev: 5.32
Mean + Std Dev: 8.30
Std Dev: 1.49
2
4
6
8
Netflow_Bytes

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

# df2 is your dataframe

plt.figure(figsize=(17, 6))

corr = df2.corr(method='spearman')

my_m = np.triu(corr)

sns.heatmap(corr, mask=my_m, annot=True, cmap="Set2")

plt.show()

correlation_matrix = df2.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

plt.show()
```



```
#The preprocessing module in scikit-learn provides various tools and techniques for preprocessing your data before

#feeding it into machine learning models.

#This preprocessing is crucial to improve the quality of your data and the performance of your models.

from sklearn import preprocessing #pip install scikit-learn
```

```
#The code segment uses scikit-learn's LabelEncoder to transform categorical variables into numerical values.

#Each categorical column, such as 'Protocol,' 'Flag,' 'Family,' 'SeedAddress,' 'ExpAddress,' 'IPaddress,' 'Threats,' and

#'Prediction,' is encoded into unique numeric labels.

#This preprocessing step is essential for machine learning algorithms, as they typically require numerical input data

#instead of categorical labels.


lab_encoder = preprocessing.LabelEncoder()                # transformation of categorical to numeric

df2['Protocol'] = lab_encoder.fit_transform(df2['Protocol'])

df2['Flag'] = lab_encoder.fit_transform(df2['Flag'])

df2['Family'] = lab_encoder.fit_transform(df2['Family'])

df2['SeedAddress'] = lab_encoder.fit_transform(df2['SeedAddress'])

df2['ExpAddress'] = lab_encoder.fit_transform(df2['ExpAddress'])

df2['IPaddress'] = lab_encoder.fit_transform(df2['IPaddress'])

df2['Threats'] = lab_encoder.fit_transform(df2['Threats'])

df2['Prediction'] = lab_encoder.fit_transform(df2['Prediction'])

df2
```



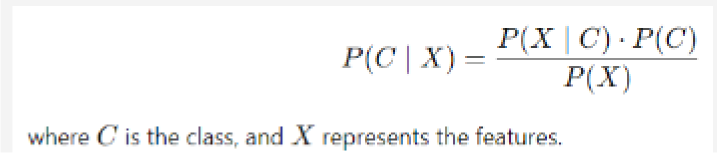
	Time	Protocol	Flag	Family	Clusters	SeedAddress	ExpAddress	BTC	USD	Netflow_Bytes	IPaddress	Threats	Port	Prediction
0	61	1	0	16	1	2	2	0.645909	22.360680	1.791759	0	1	5061	2
1	51	1	0	16	1	2	2	0.645909	22.449944	2.197225	0	1	5061	2
2	41	1	0	16	1	2	2	0.645909	22.538855	2.079442	0	1	5061	2
3	31	1	0	16	1	2	2	0.645909	22.627417	2.772589	0	1	5061	2
4	68	1	0	16	1	2	2	0.645909	22.715633	2.302585	0	1	5061	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
149038	44	2	2	15	3	1	6	3.686013	39.874804	8.114025	0	6	5062	0
149039	44	2	2	15	3	1	6	3.686961	39.949969	8.117312	0	6	5062	0
149040	44	2	2	15	3	1	6	3.687905	40.024992	8.120589	0	6	5062	0
149041	44	2	2	15	3	1	6	3.688844	40.099875	8.123854	0	6	5062	0
149042	44	2	2	15	3	1	6	3.689779	40.174619	8.127109	0	6	5062	0

149043 rows × 14 columns

Supervised and Ensemble Learning

Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence between features. It is often used for classification tasks and works particularly well for large datasets.

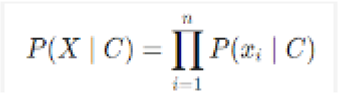
Bayes' Theorem: Naive Bayes uses Bayes' theorem to calculate the probability of a class given the features



$$P(C \mid X) = \frac{P(X \mid C) \cdot P(C)}{P(X)}$$

where C is the class, and X represents the features.

Naive Assumption: The "naive" part refers to the assumption that all features are independent given the class. This simplifies the computation of P not stretchy left parenthesis X | C not stretchy right parenthesisP(X|C) to:



$$P(X \mid C) = \prod_{i=1}^{n} P(x_i \mid C)$$

where xi are the individual features.

Types: There are different types of Naive Bayes classifiers based on the type of features:

- **Gaussian Naive Bayes**: Assumes that features follow a Gaussian distribution.
- **Multinomial Naive Bayes**: Used for discrete features like word counts in text classification.
- **Bernoulli Naive Bayes**: Used for binary/boolean features.

Strengths:

- Simple and easy to implement.
- Works well with high-dimensional data.

Weaknesses:

The assumption of feature independence is often unrealistic, which can affect performance.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It finds the hyperplane that best separates different classes in the feature space.

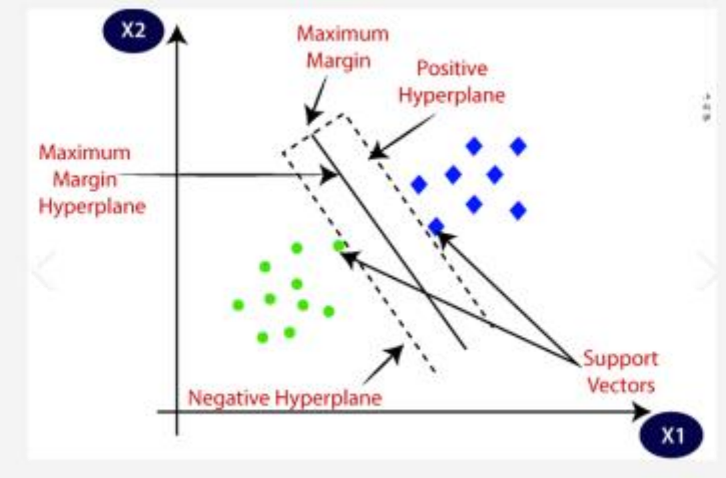
- **Hyperplane**: SVM tries to find the optimal hyperplane that maximizes the margin between two classes. The margin is the distance between the hyperplane and the nearest data points from each class, which are called support vectors.
- **Kernel Trick**: SVM can be extended to handle non-linear relationships using the kernel trick, which transforms the data into a higher-dimensional space where a linear hyperplane can be used to separate the classes.
- **Regularization**: SVM includes a regularization parameter that controls the trade-off between maximizing the margin and minimizing classification error.

Strengths:

- Effective in high-dimensional spaces.
- Can handle non-linear boundaries using kernels.

Weaknesses:

- Can be computationally expensive for large datasets.
- Requires careful tuning of hyperparameters.



Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve classification and regression performance. It uses the following principles:

- **Bootstrap Aggregating** (Bagging): Random forests build multiple decision trees using different subsets of the training data. Each tree is trained on a random sample of the data with replacement (bootstrap sample).
- **Random Feature Selection**: When splitting nodes in a tree, random forests consider a random subset of features. This reduces correlation between trees and improves generalization.
- **Voting/Averaging**: For classification, the final prediction is made by majority voting from all decision trees. For regression, it is the average of predictions from all trees.

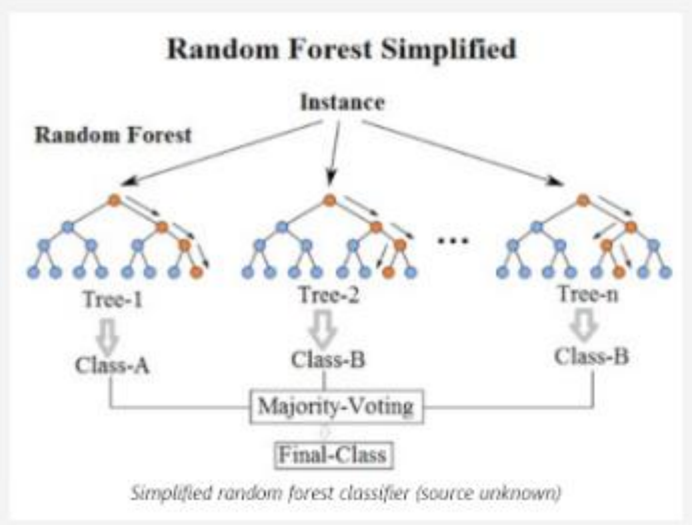
Strengths:

- Handles both classification and regression tasks.
- Reduces overfitting compared to individual decision trees.

- Can handle large datasets with many features.

Weaknesses:

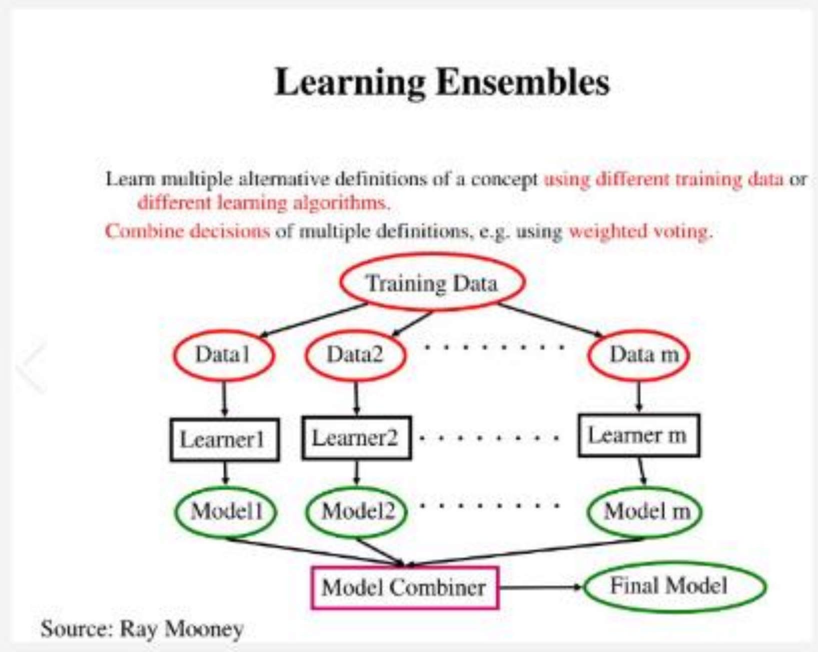
- Can be computationally intensive and slower to predict than individual decision trees.
- Less interpretable than a single decision tree.



Simplified random forest classifier (source unknown)

Ensemble Learning

Ensemble Learning is a machine learning paradigm that combines multiple models to produce a better overall performance than individual models. The idea is to leverage the strengths of different models and mitigate their weaknesses.



Learning Ensembles

Learn multiple alternative definitions of a concept using different training data or different learning algorithms.

Combine decisions of multiple definitions, e.g. using weighted voting.

Source: Ray Mooney

Strengths:

- Often improves accuracy and robustness compared to individual models.
- Can handle a wide variety of data and problems.

Weaknesses:

- Can be complex to implement and tune.
- May require more computational resources.

In summary:

Naive Bayes is a probabilistic classifier based on independence assumptions.

SVM is a powerful classifier that finds optimal decision boundaries.

Random Forest is an ensemble of decision trees that improves performance through averaging and randomization.

Ensemble Learning combines multiple models to improve performance and generalization.

Types of Ensemble Methods:

Bagging (Bootstrap Aggregating): Builds multiple models (e.g., decision trees) using different subsets of the data. The final prediction is an aggregation of predictions from all models. Random Forest is a popular bagging method.

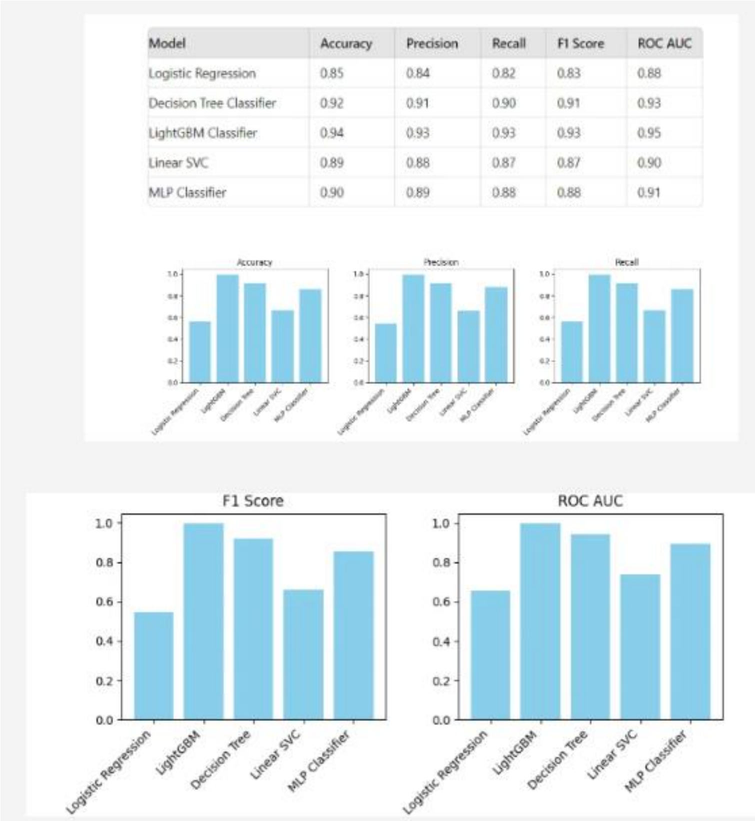
Boosting: Sequentially builds models, where each new model corrects errors made by the previous ones. Examples include AdaBoost and Gradient Boosting Machines (GBM).

Stacking: Combines multiple models (base learners) and uses another model (meta-learner) to aggregate their predictions. The base learners might use different algorithms, and the meta-learner combines their outputs.

Unsupervised Learning and Deep Learning

Recaputilation: Lazypredictor and LazyRegressor model for supervised learning and ROC evaluation metric.

See this link, it teachs you how to plot the ROC value using the UGRansome dataset https://medium.com/@samson.sabu/blog-post-ransomware-analysis-using-machine-learning-and-deep-learning-a-comprehensive-study-with-a48dfe024dbf



Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
Logistic Regression	0.85	0.84	0.82	0.83	0.88
Decision Tree Classifier	0.92	0.91	0.90	0.91	0.93
LightGBM Classifier	0.94	0.93	0.93	0.93	0.95
Linear SVC	0.89	0.88	0.87	0.87	0.90
MLP Classifier	0.90	0.89	0.88	0.88	0.91

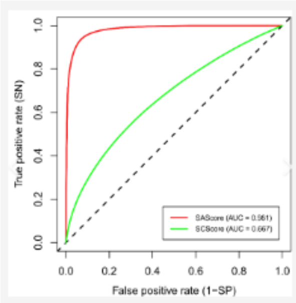
ROC stands for **Receiver Operating Characteristic**. It's a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.:

True Positive Rate (TPR) or Sensitivity: This is plotted on the y-axis. It represents the proportion of actual positives correctly identified by the model (e.g., the percentage of cryptojacking cases correctly detected).

False Positive Rate (FPR): This is plotted on the x-axis. It represents the proportion of actual negatives incorrectly identified as positives (e.g., the percentage of normal cases incorrectly flagged as cryptojacking).

The ROC curve shows the trade-off between sensitivity and specificity (1 - FPR). A model with a curve closer to the top-left corner indicates a better performance, as it shows a higher true positive rate with a lower false positive rate.

The area under the ROC curve (AUC) is often used as a single metric to summarize the model's performance. An AUC of 1.0 represents a perfect model, while an AUC of 0.5 suggests no discrimination, akin to random guessing.



Unsupervised learning for image processing involves analyzing and extracting patterns from images without using labeled data. Unlike supervised learning, where models are trained on labeled datasets (i.e., images with corresponding labels), unsupervised learning works with unlabeled data, discovering hidden structures and relationships in the data. Here are some common techniques and their applications in image processing:

Clustering

Technique: Clustering algorithms like K-means, hierarchical clustering, and DBSCAN group similar images or regions within images into clusters based on pixel intensity, color, texture, or other features.

Applications:

- Image segmentation (dividing an image into meaningful segments)
- Organizing large image datasets by grouping similar images together
- Anomaly detection in images by identifying clusters of unusual patterns

Dimensionality reduction

Technique: Techniques like Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and autoencoders reduce the dimensionality of image data while preserving important information.

Applications:

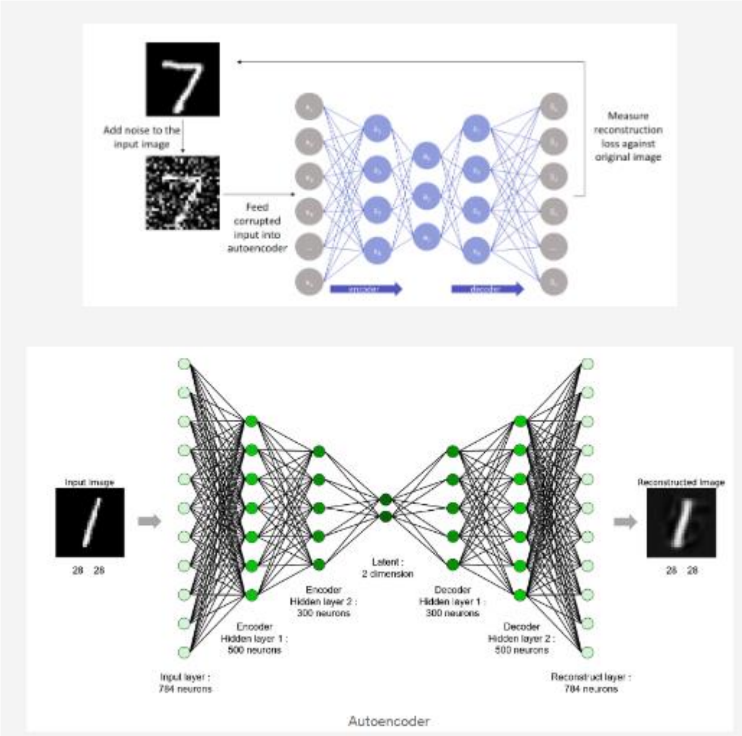
- FImage compression by reducing the size of image data
- Visualization of high-dimensional image data in 2D or 3D space
- Feature extraction for downstream tasks like classification or clustering

Autoencoders

Technique: An autoencoder is a type of neural network used to learn efficient representations of data, often for the purpose of dimensionality reduction or feature learning. It consists of an encoder that compresses the image into a latent space and a decoder that reconstructs the image from this space.

Applications:

- Image denoising by training autoencoders to reconstruct clean images from noisy ones
- Anomaly detection by comparing the input image with its reconstruction
- Generating new images by sampling from the latent space


Add noise to the input image
Feed corrupted input into autoencoder
Measure reconstruction loss against original image
encoder
decoder
Input image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed image
28 28

Autoencoder

Generative models

Technique: Models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) generate new images that resemble the original data by learning the underlying distribution of the image data.

Applications:

- Image synthesis to create new, realistic images
- Data augmentation to generate additional training data for supervised learning tasks
- Style transfer by altering the appearance of images to match a desired style

Self-organizing maps (SOMs)

Technique: SOMs are a type of neural network that map high-dimensional image data onto a lower-dimensional grid while preserving the topological structure of the data.

Applications:

- Visualizing patterns in image datasets
- Clustering similar images
- Dimensionality reduction for feature extraction

Benefits

- **No Label Dependency**: Unsupervised learning doesn't rely on labeled data, making it useful in scenarios where labeling is expensive or impractical.
- **Exploratory Analysis:** It helps in exploring and understanding the underlying structure of image data, leading to insights that may not be apparent with supervised learning.

- **Feature Learning**: It can discover useful features or representations of images that can be used in other tasks like classification, object detection, or segmentation.

Unsupervised learning for image processing is a powerful approach for tasks where labeled data is scarce, expensive, or unavailable. It is widely used in exploratory data analysis, feature extraction, and anomaly detection in various image processing applications.

We will implement unsupervised deep learning techniques, specifically using autoencoders with both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), including Long Short-Term Memory (LSTM) networks.

Breakdown of the components in our architecture

Load the MNIST dataset. The MNIST dataset is a classic dataset used for training various image processing systems. It consists of handwritten digits from 0 to 9.

Normalize the data. Normalization scales the data to a standard range, usually [0, 1], which helps improve the performance and convergence of neural networks.

Reshape data for CNN (28x28x1) and RNN (28 timesteps, 28 features):

- For CNN, the MNIST images are reshaped to have a single channel (grayscale) with dimensions 28x28.
- For RNN, the data is reshaped to 28 timesteps, each with 28 features. This transformation considers each row of the image as a timestemp and each pixel in that row as a feature.

Build CNN and RNN based autoencoder models with encoder & decoder:

- CNN-based autoencoder. Uses convolutional layers in the encoder and decoder parts to learn spatial features from the images.
- RNN-based autoencoder. Uses RNN layers (or LSTM layers) to capture temporal dependencies in the image data. In this case, LSTM autoencoders are also mentioned, which use LSTM cells for encoding and decoding.

Reconstruct images using both models. After training, the autoencoders will reconstruct the input images. The CNN autoencoder will output reconstructed images based on learned spatial features, and the RNN autoencoder will output images based on learned temporal features.

Plot original and reconstructed images:

- **Display original**: Shows the original MNIST images.
- **Display CNN reconstructed**: Shows the images reconstructed by the CNN autoencoder.
- **Display RNN reconstructed**: Shows the images reconstructed by the RNN autoencoder.

Show(): This command displays the plots.

This architecture involves training autoencoders with both CNN and RNN (LSTM) components, which is a common deep learning approach for image reconstruction and anomaly detection tasks.

Deep neural networks (DNNs) are a class of machine learning models composed of multiple layers of interconnected nodes (neurons) that learn to transform input data into output predictions. Different architectures of DNNs are designed to handle various types of data and tasks. Here are three common types of DNN architectures: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory networks (LSTMs).

Convolutional neural networks (CNNs)

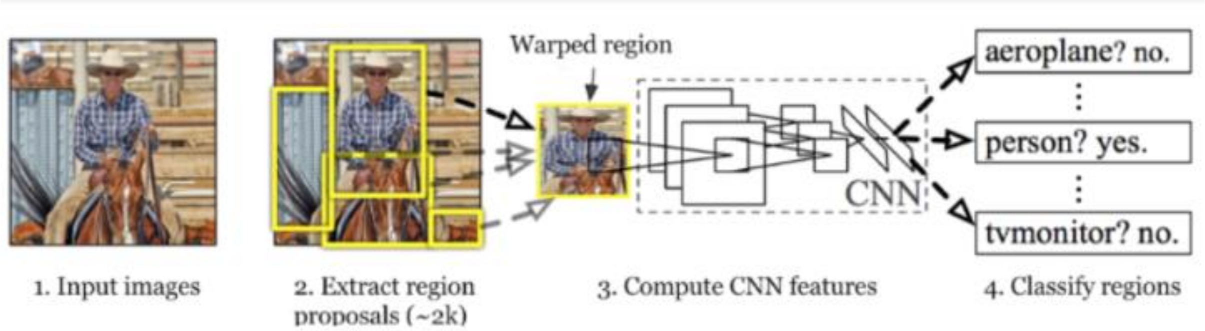
Purpose: CNNs are specialized for processing grid-like data, such as images.

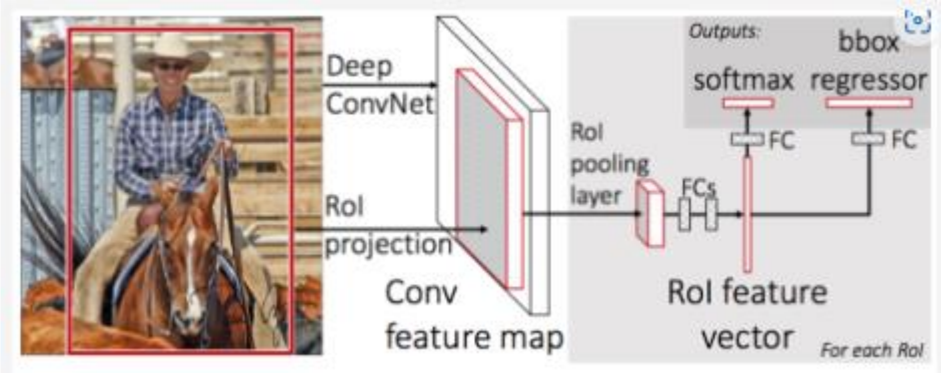
Key Components:

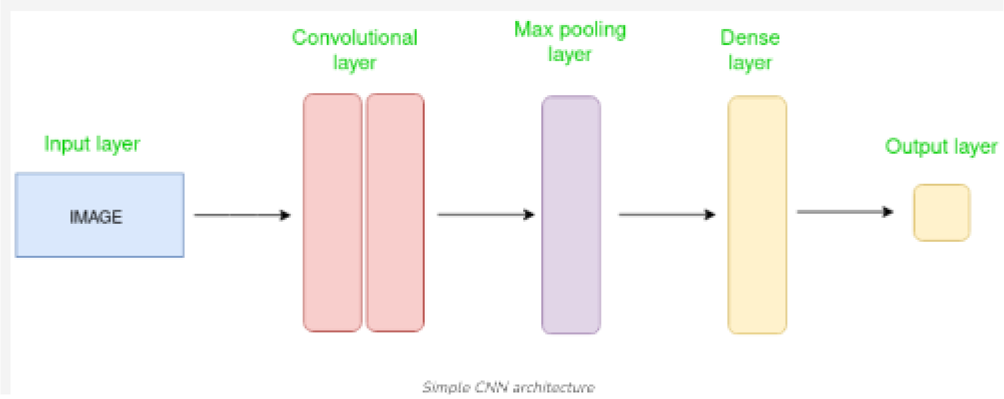
- **Convolutional Layers**: Apply convolutional filters to detect features like edges, textures, and patterns in the input image. These filters slide over the image, producing feature maps.
- **Pooling Layers**: Reduce the spatial dimensions of the feature maps, retaining important information while reducing computational complexity. Max pooling is a common pooling technique.
- **Fully Connected Layers**: Neurons in these layers are fully connected to all activations in the previous layer, and they often serve as the final layers in a CNN, where classification or regression is performed.

Applications:

- Image classification (e.g., identifying objects in images)
- Object detection (e.g., detecting and localizing objects within an image)
- Image segmentation (e.g., dividing an image into regions with different objects)
- Video analysis and image generation (e.g., GANs)

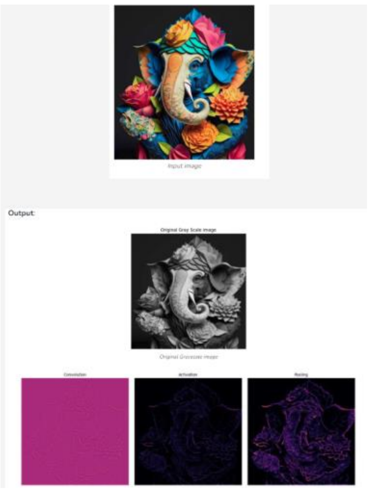






Simple CNN architecture

The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected/dense layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.



Recurrent neural networks (RNNs)

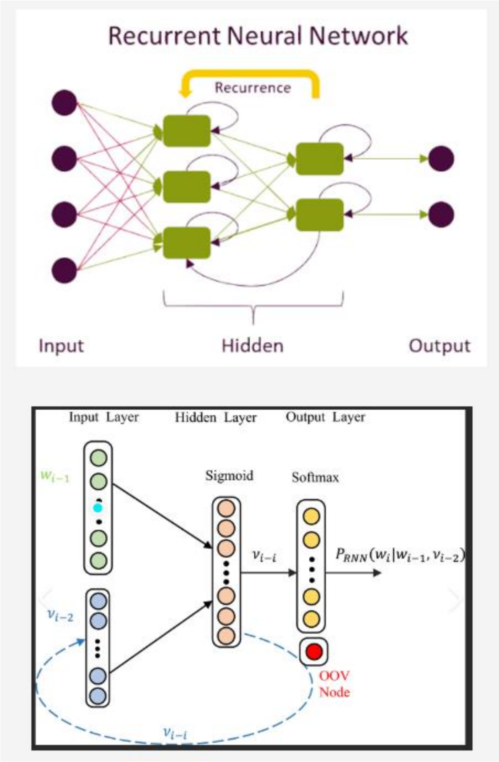
Purpose: RNNs are designed for sequential data, where the order of inputs matters, such as time series, text, or speech.

Key Components:

- **Recurrent Layers**: In an RNN, the output from the previous time step is fed back into the network along with the new input. This allows the network to maintain a memory of previous inputs, making it suitable for tasks involving sequences.
- **Hidden States**: The hidden state in an RNN captures information about previous inputs in the sequence. It gets updated at each time step, influencing the output.

Applications:

- Natural language processing (NLP) tasks like text generation, sentiment analysis, and machine translation
- Time series forecasting (e.g., predicting stock prices or weather patterns)
- Speech recognition and generation
- Sequential data classification (e.g., activity recognition in sensor data)



Long short-term memory networks (LSTMs)

Purpose: LSTMs are a specialized type of RNN designed to better capture long-term dependencies in sequential data.

Key Components:

Memory Cell: The core of an LSTM is the memory cell, which retains information over time. It is regulated by three gates:

- Input Gate: Controls how much of the new input to store in the memory cell.
- Forget Gate: Decides what information to discard from the memory cell.
- Output Gate: Determines the output of the LSTM at the current time step based on the memory cell's state.

Gating Mechanism: The gates use sigmoid functions to control the flow of information, ensuring that relevant information is retained and irrelevant information is discarded.

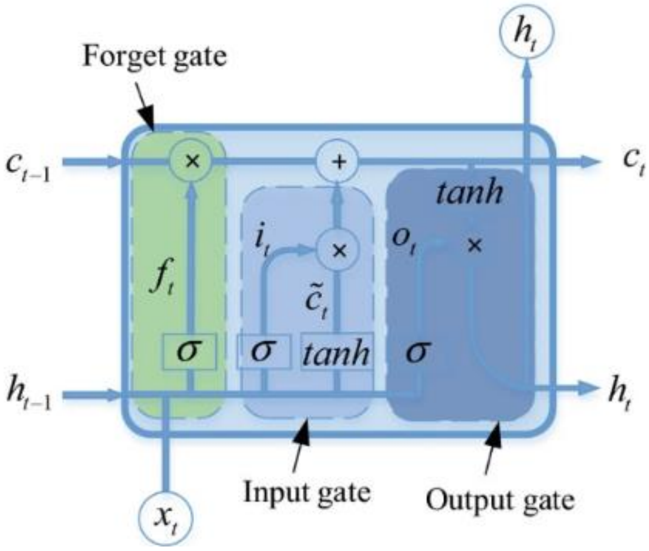
Applications:

Long-term sequence prediction (e.g., language modeling, text generation)

Machine translation (e.g., translating sentences from one language to another)

Speech synthesis and recognition

Video analysis and anomaly detection in sequential data



Key Differences and Applications

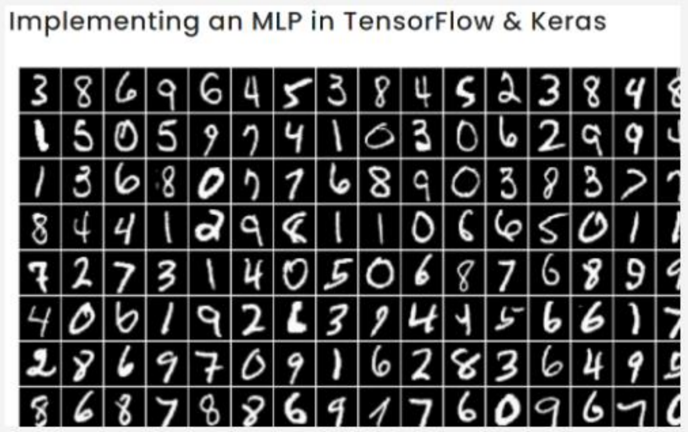
CNNs: Best suited for spatial data like images and videos, focusing on local feature detection through convolution.

RNNs: Ideal for sequential data where the order of inputs is crucial. RNNs are suitable for tasks where previous context influences the current output, such as in text or time series.

LSTMs: A type of RNN that excels at capturing long-term dependencies in sequences, making it particularly effective for tasks like language modelling and speech recognition.

Each of these architectures has unique strengths and is applied in different contexts, depending on the nature of the data and the task at hand.

MNIST Dataset for Handwriting Recognition



Implementing an MLP in TensorFlow & Keras

To reconstruct images using both CNN and RNN models, we'll modify the architecture of each model to include a decoder component. This allows the model to reconstruct the input images from the encoded representation.

Code for image reconstruction using CNN and RNN

1. Load the MNIST dataset

2. Normalize the data

3. Reshape data for CNN (28x28x1) and RNN (28 timesteps, 28 features)

4. Build:

   CNN, RNN based autoencoder model with encoder & decoder

5. Reconstruct images using both models

6. Plot original and reconstructed images

   6.1. Display original

   6.2. Display CNN reconstructed

   6.3. Display original

7. Display RNN reconstructed

8. Show()

Explanation

CNN Autoencoder:

- The CNN model first encodes the input image into a lower-dimensional representation using a convolutional layer followed by max pooling.
- It then reconstructs the image using upsampling and convolutional layers.

RNN Autoencoder:

- The RNN model treats each row of the image as a sequence and encodes it into a fixed-size vector.
- It then reconstructs the image by repeating the encoded vector and using another RNN layer to decode it back into the original shape.

Training:

- Both models are trained to minimize the reconstruction loss, which measures the difference between the original and reconstructed images.

Visualization:

- After training, we visualize the original and reconstructed images for both CNN and RNN models.

How to run

Install Required Libraries: Ensure you have TensorFlow installed in your environment

pip install tensorflow

Review the results. The script will display the original images along with their reconstructions from both the CNN and RNN models. You can compare how well each model reconstructs the images.

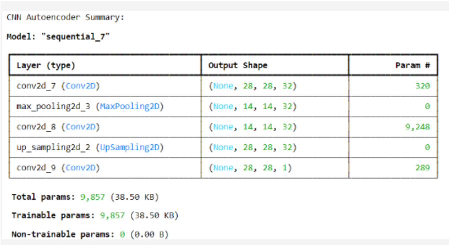
This will give you a clear comparison of how CNNs and RNNs handle image reconstruction.

cnn_autoencoder.summary(): Prints a summary of the CNN autoencoder model, including layer types, output shapes, and the number of parameters.

rnn_autoencoder.summary(): Prints a summary of the RNN autoencoder model, showing similar details.

lstm_autoencoder.summary(): Prints a summary of the LSTM autoencoder model.

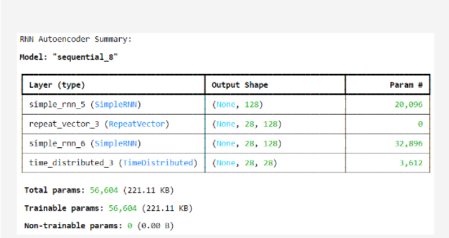
This will give you a detailed look at the number of layers, layer types, and other important characteristics of each model.



```
CNN Autoencoder Summary:

Model: "sequential_7"
```

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_8 (Conv2D)	(None, 14, 14, 32)	9,248
up_sampling2d_2 (UpSampling2D)	(None, 28, 28, 32)	0
conv2d_9 (Conv2D)	(None, 28, 28, 1)	289

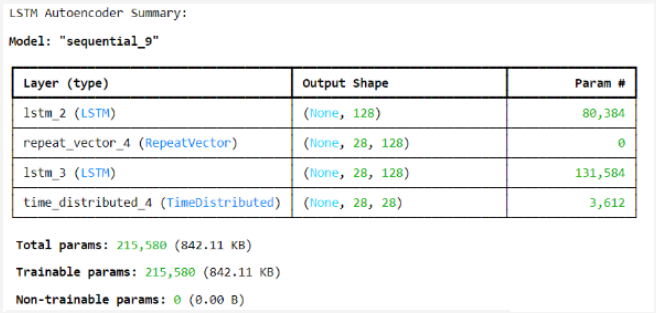
```
Total params: 9,857 (38.50 KB)

Trainable params: 9,857 (38.50 KB)

Non-trainable params: 0 (0.00 B)
```



```
RNN Autoencoder Summary:

Model: "sequential_8"
```

Layer (type)	Output Shape	Param #
simple_rnn_5 (SimpleRNN)	(None, 128)	20,096
repeat_vector_3 (RepeatVector)	(None, 28, 128)	0
simple_rnn_6 (SimpleRNN)	(None, 28, 128)	32,896
time_distributed_3 (TimeDistributed)	(None, 28, 28)	3,612

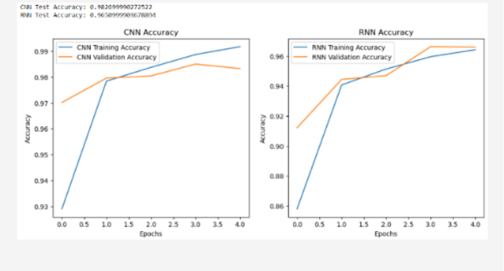
```
Total params: 56,604 (221.11 KB)

Trainable params: 56,604 (221.11 KB)

Non-trainable params: 0 (0.00 B)
```



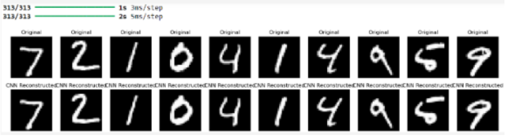
```
LSTM Autoencoder Summary:

Model: "sequential_9"
```

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 128)	80,384
repeat_vector_4 (RepeatVector)	(None, 28, 128)	0
lstm_3 (LSTM)	(None, 28, 128)	131,584
time_distributed_4 (TimeDistributed)	(None, 28, 28)	3,612

```
Total params: 215,580 (842.11 KB)

Trainable params: 215,580 (842.11 KB)

Non-trainable params: 0 (0.00 B)
```



```
CNN Test Accuracy: 0.982699990272522
RNN Test Accuracy: 0.9650999903678894
```



```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

To include LSTM in the image reconstruction task, **we'll modify the RNN model to use LSTM layers instead**. LSTMs are better suited than simple RNNs for capturing long-term dependencies, which might lead to better reconstruction performance.

Code for Image Reconstruction Using CNN, RNN, and LSTM

To include LSTM in the image reconstruction task, we'll modify the RNN model to use LSTM layers instead. LSTMs are better suited than simple RNNs for capturing long-term dependencies, which might lead to better reconstruction performance.

1. Load the MNIST dataset

2. Normalize the data

3. Reshape data for CNN (28x28x1) and RNN (28 timesteps, 28 features)

4. Build:

CNN, RNN based autoencoder model with encoder & decoder

4.1. Build LSTM autoencoder with encoder & decoder

5. Reconstruct images using both models

6. Plot original and reconstructed images

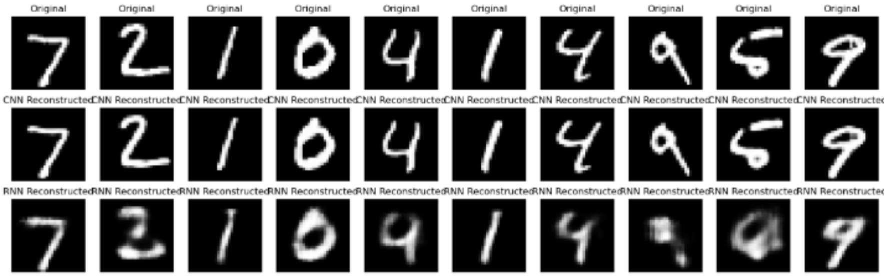
6.1. Display original

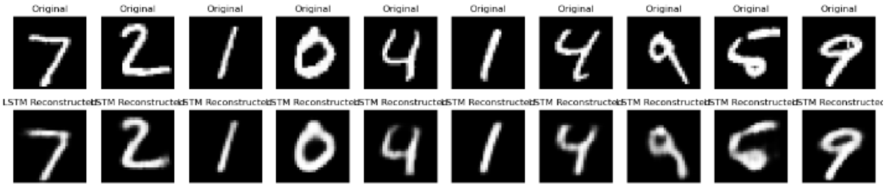
6.2. Display CNN reconstructed

6.3. Display original

7. Display RNN reconstructed

8. Show()





Note: Use the uploaded code to practice.

See the K-Means code for unsupervised learning

Natural Language Processing and Computational Lexicography

The study of natural language processing (NLP) encompasses a wide array of essential concepts and tools aimed at understanding and processing human language. Key components include **lexicons**, which are vital **resources containing words or phrases categorized with their sentiment or other linguistic attributes**. These lexicons serve as foundational tools for sentiment analysis, a crucial task in NLP that involves assessing the emotional tone conveyed by text. Techniques like **bag of words and tokenization** are fundamental preprocessing steps that **break down text into manageable units**, allowing for analysis and modeling. Moreover, **part of speech tags and named entity recognition** enable the identification and categorization of words based on their **grammatical roles or as specific entities** like names or locations within text data.

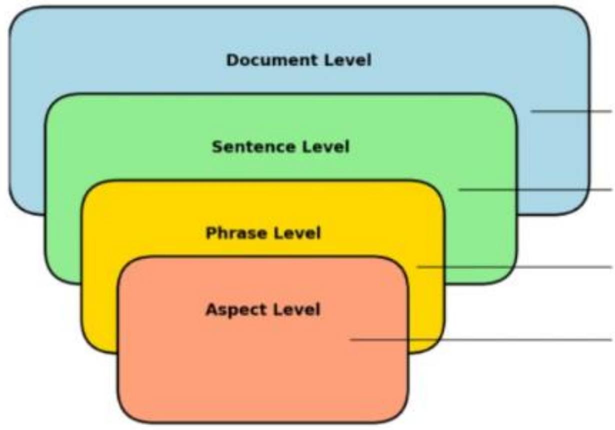
In practical applications, **NLP frameworks such as the Natural Language Toolkit** (NLTK) and indexed techniques like **BM25** play pivotal roles. NLTK provides a comprehensive **suite of libraries** and algorithms for NLP tasks, facilitating tasks from tokenization to syntactic analysis. Meanwhile, **BM25 indexing enhances search efficiency by ranking documents based on relevance to a query** within preprocessed patterns. Addressing challenges like **polysemic dilemmas**, **where words have multiple meanings depending on context, and managing stop words, common but non-informative words, are critical to refining NLP models and improving accuracy in tasks like text translation and sentiment analysis**.

Sentiment analysis itself relies on specialized lexicons like Sentistrength, VADER, SentiWordNet, Liu and Hu lexicon, AFINN-111 , and SentiWordNet, each tailored to capture nuances in sentiment expressed through text, supporting more nuanced and accurate analysis of textual data.

Computational lexicography is the field that applies computational techniques to the **creation, analysis, and management of dictionaries and lexical resources**. It combines principles from linguistics, computer science, and lexicography to automate the process of collecting, organizing, and defining words and their meanings. This includes tasks such as **word sense disambiguation**, identifying **part-of-speech tags**, analyzing **word usage patterns**, and developing algorithms that can process large corpora of text to extract and update dictionary entries efficiently.

In computational lexicography, tools like machine learning, natural language processing (NLP), and databases are employed to construct and maintain digital dictionaries, often in multiple languages. **These resources are essential for various applications, such as language learning, machine translation, and text analysis**.

The **integration of automated methods such as Large Language Models (LLM) and GenAI** allow for the rapid updating of dictionaries, the inclusion of new words, and the identification of word relationships in vast datasets, ultimately improving the accuracy and usability of **lexical resources in the digital age.**



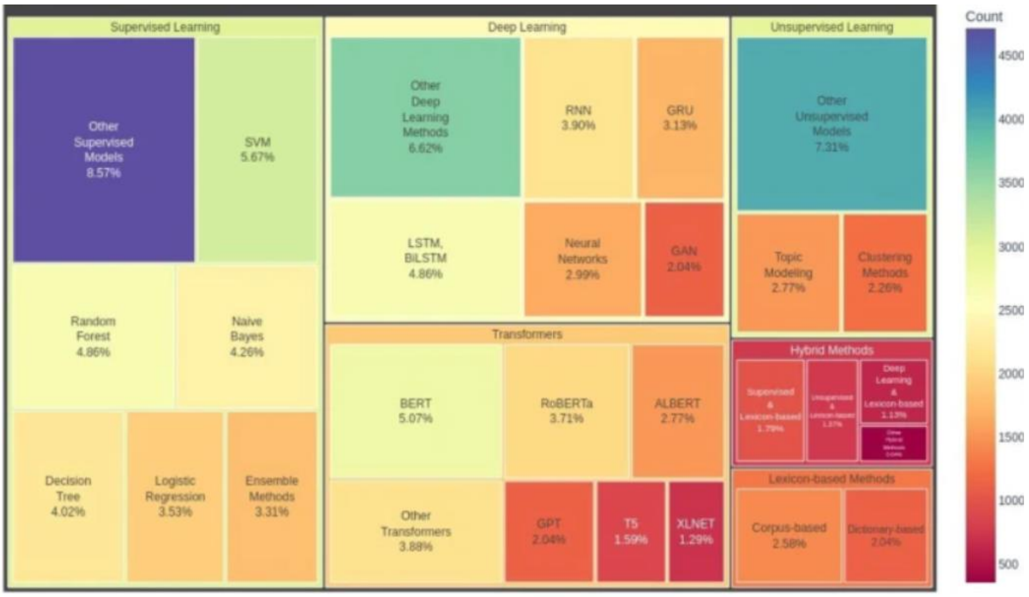
Evaluates an entire document as a single data point.
Useful for understanding overall sentiment of lengthy texts like reviews or articles.

Breaks down the text into individual sentences, assessing the sentiment of each.
Captures varying sentiments within a document.

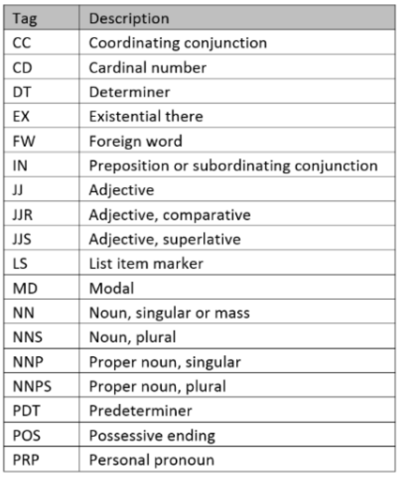
Examines sentiments within specific phrases or segments of sentences.
Identifies and classifies sentiment-bearing phrases.

Focuses on specific aspects or features of a product or service mentioned in the text.
Provides a detailed sentiment breakdown.

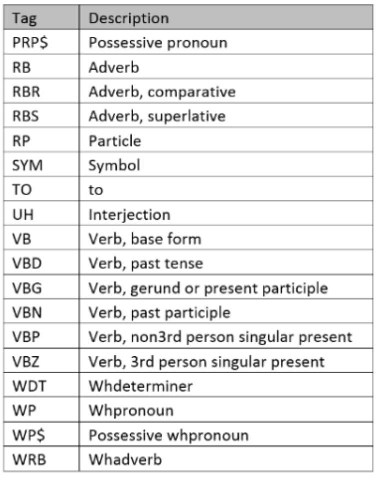
Levels of sentiment analysis



Sentiment analysis approaches



Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun



Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

Please, read the following research papers, to familiarize yourself with the NLP jargon:

Nkongolo Wa Nkongolo, M., 2023. News Classification and Categorization with Smart Function Sentiment Analysis. International Journal of Intelligent Systems, 2023(1), p.1784394. https://doi.org/10.1155/2023/1784394

Mohammed, I. and Prasad, R., 2024. Lexicon dataset for the Hausa language. Data in Brief, 53, p.110124. https://doi.org/10.1016/j.dib.2024.110124

Recommended reading:

Adeliyi, T.T., Oluwadele, D., Igwe, K., Aroba, O.J. (2023). Analysis of road traffic accidents severity using a pruned tree-based model. International Journal of Transport Development and Integration, Vol. 7, No. 2, pp. 131-138. https://doi.org/10.18280/ijtdi.070208

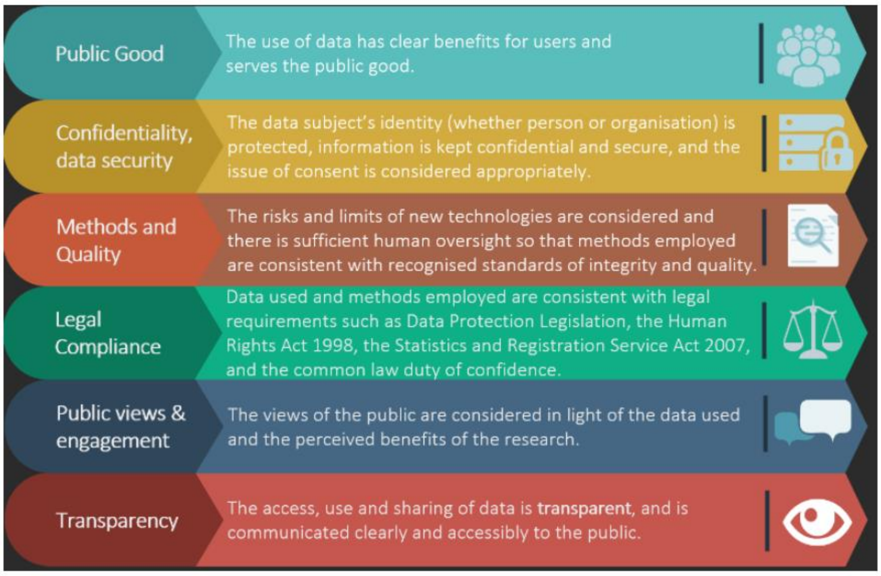
Ethics

Definition of Ethics: Overview of ethical principles and their importance in decision-making.

- **Privacy**: Protecting individuals' personal data from unauthorized access or misuse.
- **Bias and Fairness**: Ensuring data and algorithms do not disproportionately disadvantage any group.
- **Transparency**: Being clear about how data is collected, processed, and how decisions are made by algorithms.
- **Accountability**: Taking responsibility for the impact of data-driven decisions and ensuring that harm is minimized.
- **Security**: Safeguarding data from breaches and ensuring the confidentiality and integrity of information.

Ethics in Data Science refers to the application of moral principles and values to the processes and practices involved in collecting, analyzing, and using data. It ensures that data science activities are conducted in a way that respects individual rights, promotes fairness, maintains transparency, and avoids harm. Key ethical considerations include:

Ethics in data science aims to balance innovation and societal good with legal and moral responsibilities, preventing misuse of data and ensuring trust in data-driven technologies.

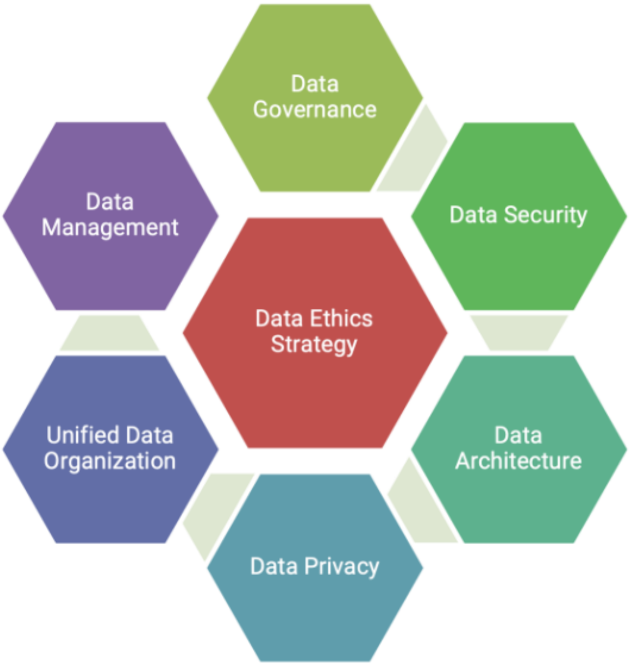


Public Good	The use of data has clear benefits for users and serves the public good.
Confidentiality, data security	The data subject's identity (whether person or organisation) is protected, information is kept confidential and secure, and the issue of consent is considered appropriately.
Methods and Quality	The risks and limits of new technologies are considered and there is sufficient human oversight so that methods employed are consistent with recognised standards of integrity and quality.
Legal Compliance	Data used and methods employed are consistent with legal requirements such as Data Protection Legislation, the Human Rights Act 1998, the Statistics and Registration Service Act 2007, and the common law duty of confidence.
Public views & engagement	The views of the public are considered in light of the data used and the perceived benefits of the research.
Transparency	The access, use and sharing of data is **transparent**, and is communicated clearly and accessibly to the public.

Relevance to Data Science: Discuss why ethics is critical in data-driven practices, particularly due to the influence of data science on society (e.g., AI, automation).

Examples of Ethical Issues: Misuse of personal data, biased algorithms, and lack of transparency.

2. Key ethical principles in data science



Fairness: Ensuring that data models and algorithms treat all individuals or groups fairly without bias (e.g., gender, race, socio-economic status).

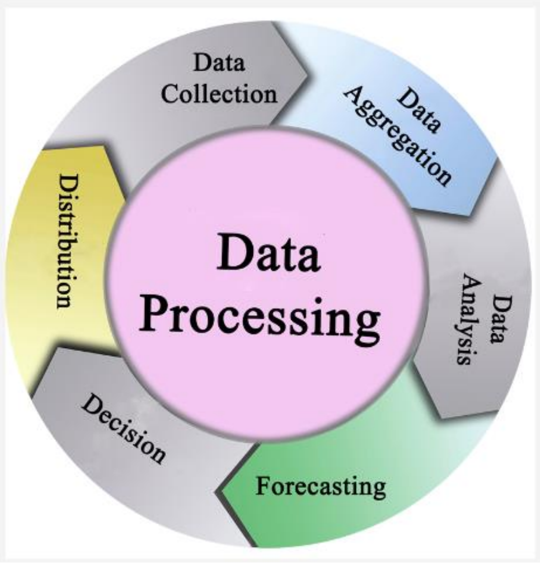
Transparency: Data scientists must provide transparency in their methods, data sources, and algorithms.

Accountability: Who is responsible when a data model causes harm? The need for accountability mechanisms for decisions made by AI or ML models.

Privacy: Understanding privacy laws and ensuring that personal data is collected, processed, and stored with proper consent and security measures.

Security: Ethical responsibilities in protecting data from breaches and ensuring the confidentiality, integrity, and availability of data.

3. Bias in data collection and modelling

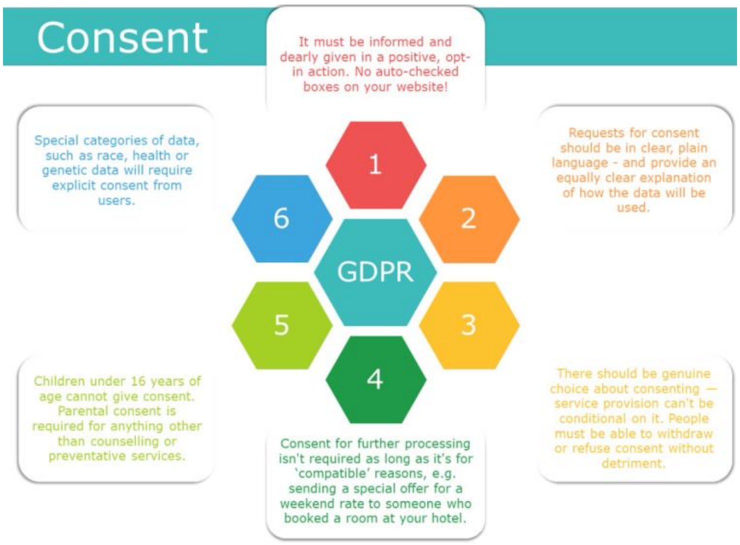


Types of Bias: Selection bias, confirmation bias, algorithmic bias, and measurement bias.

Impact of Bias: The real-world consequences of biased algorithms, such as unfair hiring practices, biased sentencing in the judicial system, or unequal loan approval rates.

Mitigating Bias: Approaches to identify and mitigate bias, such as diverse data collection, algorithm auditing, and fairness metrics.

4. Privacy and consent in data usage



It must be informed and dearly given in a positive, opt-in action. No auto-checked boxes on your website!

Requests for consent should be in clear, plain language - and provide an equally clear explanation of how the data will be used.

There should be genuine choice about consenting — service provision can't be conditional on it. People must be able to withdraw or refuse consent without detriment.

Consent for further processing isn't required as long as it's for 'compatible' reasons, e.g. sending a special offer for a weekend rate to someone who booked a room at your hotel.

Children under 16 years of age cannot give consent. Parental consent is required for anything other than counselling or preventative services.

Special categories of data, such as race, health or genetic data will require explicit consent from users.

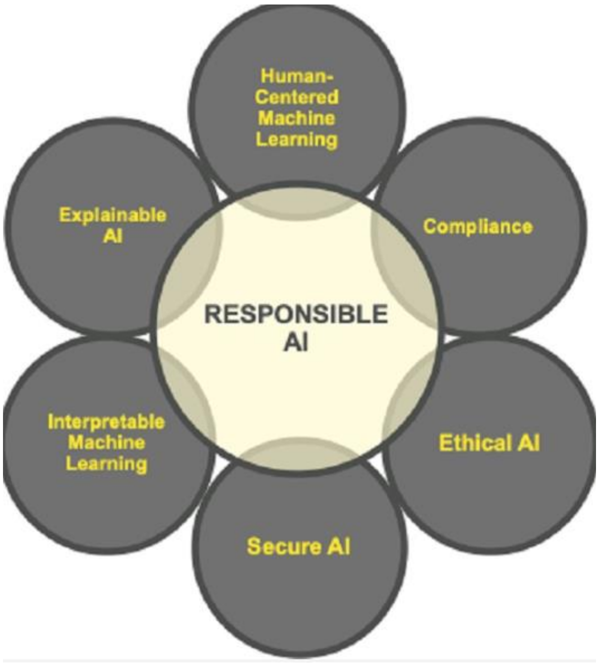
Understanding Personal Data: Define personal, sensitive, and anonymized data. Discuss the ethical considerations for handling each type.

Informed Consent: The importance of obtaining clear, informed consent from users before collecting or using their data.

Data Ownership: Who owns the data? Rights of individuals versus organizations.

Privacy Laws and Regulations: Overview of global privacy laws such as GDPR (General Data Protection Regulation) in Europe, CCPA (California Consumer Privacy Act), HIPAA (Health Insurance Portability and Accountability Act), and their implications for data science projects.

5. Ethical considerations in AI and machine learning


Human-Centered Machine Learning
Explainable AI
Compliance
RESPONSIBLE AI
Interpretable Machine Learning
Ethical AI
Secure AI

Explainability and Interpretability: Ethical importance of building models that are interpretable and understandable to non-experts.

Autonomous Decision-Making: Challenges with autonomous systems making decisions without human oversight (e.g., self-driving cars, automated hiring).

Ethical AI Frameworks: Overview of existing frameworks, such as AI ethics principles from Google, Microsoft, and other organizations.

Bias in Machine Learning Models: How biased data can lead to biased predictions, and ways to address these challenges.

6. Case studies in data science ethics

(These case studies will be use in the Exam theory section)

Case Study 1: Cambridge Analytica and Facebook Data Scandal: Discuss how user data was exploited for political purposes without informed consent, leading to ethical violations.

In 2018, it was revealed that Cambridge Analytica, a political consulting firm, had harvested personal data from millions of Facebook users without their informed consent. This data was used to create detailed psychological profiles, which were then employed to influence voter behavior in political campaigns, including the 2016 U.S. presidential election and the Brexit referendum. The scandal highlighted significant ethical violations concerning user privacy and consent. Facebook faced substantial backlash for its role in allowing third-party access to user data, leading to increased scrutiny over data protection practices and the implementation of stricter privacy regulations.

Case Study 2: Amazon's Biased Hiring Algorithm: Explore the ethical implications of building hiring algorithms that discriminated against women.

In 2018, Amazon discontinued an experimental AI recruitment tool after discovering it was biased against women. The algorithm, trained on resumes submitted over a ten-year period, favored male candidates for technical roles, effectively penalizing resumes that included the word "women" or referenced women's colleges. This incident underscores the ethical implications of deploying AI in hiring processes without addressing inherent biases in training data, leading to discriminatory outcomes.

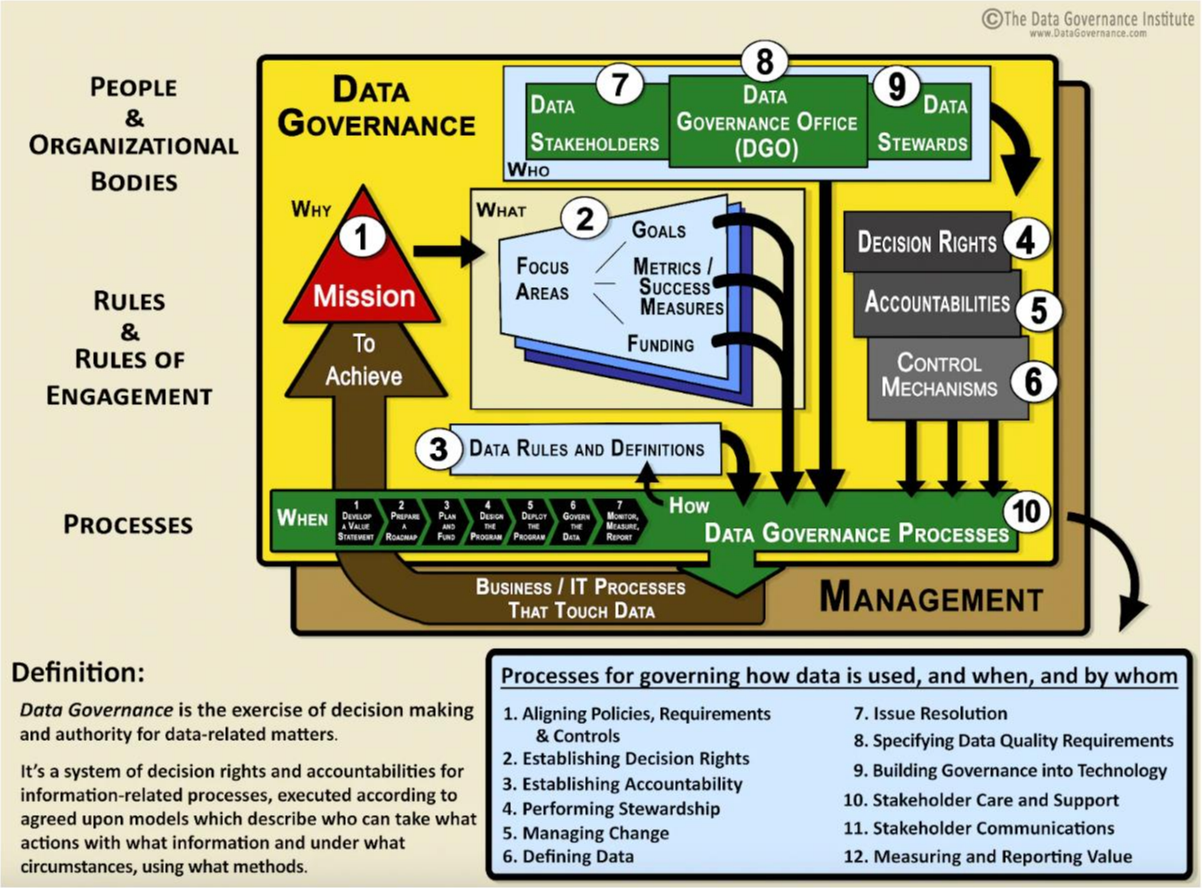
Case Study 3: Healthcare Algorithms: Ethical considerations in healthcare AI, such as algorithms misdiagnosing or prioritizing treatment unfairly based on biased data.

The integration of AI in healthcare has raised ethical concerns, particularly when algorithms trained on biased data result in misdiagnoses or unequal treatment recommendations. For instance, studies have shown that some healthcare algorithms prioritize care for certain demographic groups over others, not based on medical need but due to biases present in the training data. These disparities can exacerbate existing health inequalities, emphasizing the necessity for transparency, rigorous validation, and continuous monitoring of AI systems in healthcare to ensure equitable patient care.

Case Study 4: Predictive Policing: The ethical concerns of using AI in law enforcement and the risk of amplifying systemic bias.

Predictive policing involves using AI to forecast criminal activity and allocate law enforcement resources accordingly. However, these systems often rely on historical crime data that may reflect existing societal biases, leading to disproportionate targeting of marginalized communities. This raises ethical concerns about perpetuating systemic discrimination and infringing on civil liberties. Critics argue that without careful design and oversight, predictive policing can reinforce unjust practices rather than promote fair and effective law enforcement.

7. Ethical data science governance and guidelines



Definition:

Data Governance is the exercise of decision making and authority for data-related matters.

It's a system of decision rights and accountabilities for information-related processes, executed according to agreed upon models which describe who can take what actions with what information and under what circumstances, using what methods.

Processes for governing how data is used, and when, and by whom

1. Aligning Policies, Requirements & Controls
2. Establishing Decision Rights
3. Establishing Accountability
4. Performing Stewardship
5. Managing Change
6. Defining Data
7. Issue Resolution
8. Specifying Data Quality Requirements
9. Building Governance into Technology
10. Stakeholder Care and Support
11. Stakeholder Communications
12. Measuring and Reporting Value

Data Governance Frameworks: Key components of effective data governance, including roles, policies, and standards for ethical data use.

Corporate Social Responsibility (CSR): How organizations can integrate ethical practices into their data science strategies?

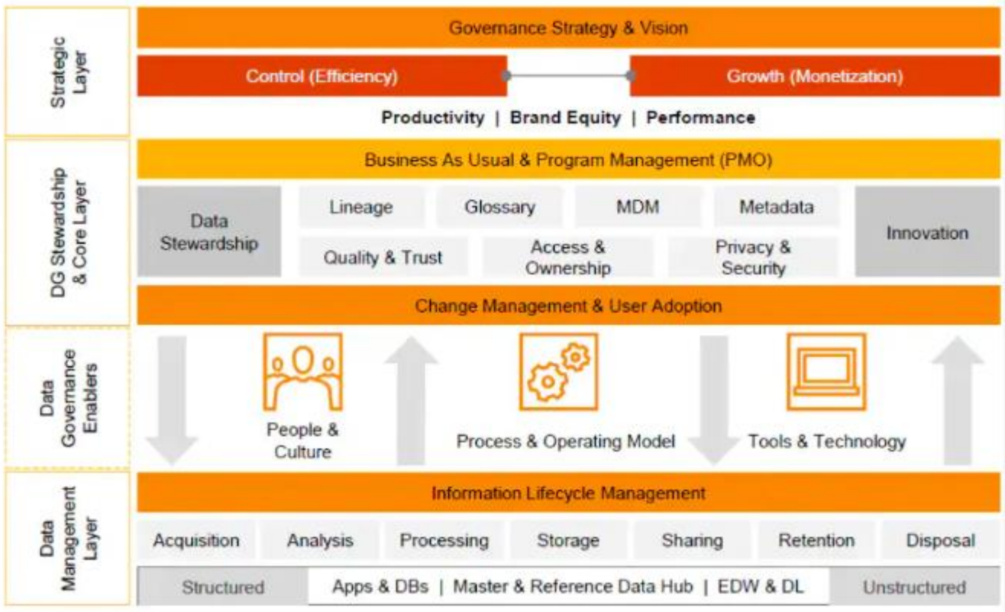


Figure 1: PwC's Data Governance Framework

8. Ethical challenges in big data and data-driven decision making

- **The Role of Big Data:** Ethical considerations when working with massive datasets, such as the risk of re-identifying individuals in anonymized datasets.
- **Decision-Making in Automated Systems**: Ethical implications of allowing automated systems to make decisions without human intervention, especially in critical domains like healthcare or finance.
- **Surveillance and Data Exploitation**: The balance between data collection for societal benefits (e.g., public health) and the risk of mass surveillance.

9. Ethics in research and publication

- **Research Integrity**: Ethical standards for publishing data science research, including transparency in methodology, reproducibility, and avoidance of cherry-picking results.
- **Bias in Peer Review and Publication**: Ethical concerns about bias in research publication, the influence of funding sources, and the importance of diversity in research.

10. Tools and techniques for ethical data science

- **Bias Detection Tools**: Introducing tools and techniques to detect and mitigate bias in datasets and algorithms (e.g., IBM AI Fairness 360).
- **Privacy-Preserving Techniques**: Discuss techniques like differential privacy, federated learning, and encryption methods that allow analysis without compromising individual privacy.

- **Ethical AI Toolkits**: Overview of toolkits designed to support ethical AI development (e.g., Google’s What-If Tool).

11. Conclusion and future trends

- **Emerging Trends**: Discuss new challenges on the horizon (e.g., ethics in deepfakes, AI in military applications, and brain-computer interfaces).
- **Responsibilities of Data Scientists**: Reinforce the ethical responsibility that data scientists bear in shaping society through the tools and models they develop.
- **The Path Forward**: Think critically about how you can contribute to ethical practices in your own work.

12. In the exam

You will debate real-world ethical dilemma in data science, such as the use of facial recognition technology in public spaces or the ethical use of AI/XAI in healthcare. You will submit your answer on click for this section of the exam.

A dataset will be given (known/unknown), and you will have to use codes developped in class to solve the problem. You will submit your notebook.ipynb for the pratical section of the exam.

You will have multiple choice, and True/False questions to assess your theoretical understanding.

The practical section: 80%

The theoretical section: 20%

Open Internet without AI tool like GPT

Content posted on Clickup will be hidden when you will be writting. Kindly prepare your notes.

Content posted on Clickup is used to set the exam.